KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Design and validation of low-complexity methods for resolving spike overlap in neuronal spike sorting

Ir. Jasper Wouters

Supervisors:
Prof. dr. ir. Alexander Bertrand
Prof. dr. Fabian Kloosterman

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2020

Design and validation of low-complexity methods for resolving spike overlap in neuronal spike sorting

Ir. Jasper WOUTERS

Examination committee:
Prof. dr. ir. Hendrik Van Brussel, chair
Prof. dr. ir. Alexander Bertrand, supervisor
Prof. dr. Fabian Kloosterman, supervisor
Prof. dr. Vincent Bonin
Prof. dr. ir. Sabine Van Huffel
Prof. dr. ir. Marc Van Hulle
Dr. Pierre Yger
(Institut de la Vision, Inserm)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2020

© 2020 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Ir. Jasper Wouters, Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

There is an end to every journey. This thesis is the end of my four year during doctoral program. It have been four crazy years with many highs and lows. Would I recommend following a doctoral program to other people? Definitely yes. Would I take a second one myself? Definitely no. I have learned many things about engineering and even a little bit about neuroscience, but mostly I have learned a lot about myself. During this journey I have had the pleasure to collaborate with and meet many interesting people. Without these people I would not be standing here today.

I would like to thank prof. dr. ir. Alexander Bertrand and prof. dr. Fabian Kloosterman for giving me the opportunity to work on their project and to be part of their respective research groups. Alexander, you have been a true inspiration to me. I must admit that during the first months of our collaboration, your knowledge and creativity were somewhat overwhelming. I was unsure about whether I would be able to make a difference, let alone where I would make that difference. However, thanks to your easy going leadership and trust, some of that knowledge trickled through and I was able to make a modest contribution of my own. This would not have been possible without your supervision. Fabian, thank you for providing some context to my work. The experiments that you and your lab are conducting are beyond imagination. You are without any doubt the most versatile researcher I have ever met. You have changed my initial and naive definition of "a psychologist" for the better.

Let me also thank the members of the examination committee for their time and effort to review this manuscript. I have really appreciated your critical mindset which has resulted in many valuable edits to the final manuscript. Prof. dr. Bonin and prof. dr. ir. Van Huffel, thank you for being part of the committee from the beginning on, your input throughout the entire project has been most valuable. Prof. dr. ir. Van Hulle and dr. Yger, thank you for sharing your insights and comments on the project. Although we only got to discuss my research at a late stage, your input and feedback have made a big difference.

Finally, I would like to thank prof. dr. ir. Van Brussel for his effort in chairing the examination committee.

A big thanks also goes out to the technical staff at STADIUS: Aldona, Elsy, Ida, John, Maarten and Wim. Thank you for your various support. I hope that in my future job I will be equally well supported for those tasks that I know nothing about by such kind people as you.

Some say a doctoral program is a lonely journey. This has not at all been the case for me thanks to the *cheerful* BIOMED people. Maybe, if it would have been lonelier, this thesis would have had some additional chapters, but I don't regret any of the time spent with my amazing/amazink colleagues. Rob, you are a man of many talents. Not only are you an award winning scientist, but your work on science communication is widely recognized. You taught me that black shouldn't be black, you enlightened me with Roboto, and above all you have been a true friend. I am sorry for the numerous times that I didn't make use of wide-screen slides. Abhi, never stop being amazed. You have been one of my best buddies around the group. Let me know when you are available for our Ladakh trip! Alexander CD, I will never forget my first and only gay experience that you got me in to, son. Amalia, I like your fighting spirit. Keep on making good music and science. Bori, although you are a respected professor by now, you will forever be remembered as the centaur of B00.26 that can dodge sharks using telepathy. Carolina, if I think about BIOMED, I think about you. It is so good to have you back. I hope we can get the most out of our final months as colleagues given the current circumstances. Cem, I have really enjoyed our coffee breaks and I am looking forward to eating iskender kebab together. Dorien, I will never give up on your hand stand. I will always be there to support you. Dries, thank you for your friendship, humour, and for the free Burger King you once bought me. Griet, thanks to you I made peace with my past as a jumper. I am proud that I am part of a community that brought forward such a great person and scientist as you. Jonathan D, I will die a happy person if I can hold your wheel for at least ten minutes. I truly cherish our time together in Berlin, those were nice days. Jonathan M, thanks for the numerous hours of ping pong practice. You are a true athlete, but unfortunately for sports, you are also smart. Laure, although you have been long gone, you have not been forgotten. I hope you still rock your bass. Mario, you are one of the most unique persons I have ever met. It is okay to be who you are. Martijn, we almost became a guild of our own, but then I had to write this thesis. Michiel, I envy your volleybal skills, thanks for helping us year after year with bringing the trophy home. Neetha, if BIOMED were a family, you would be my favourite sister. Nico, without any doubt you are the most dedicated colleague around. Science will survive for at least another decade thanks to people like you. Ofelie, thank you for the many nice cakes and

friendship. Ik eb kweetnie hoeveel respect vur jun. Pooya, thanks to you the average IQ of B00.26 has taken a significant step forward. Your quantum brain, will without doubt solve the travelling salesman problem, but don't forget to step off the train in Leuven. Simon VE, your lyrical and scientific skills are A+. Thomas DC, father of the group, your rock solid attitude and cool made me sufficiently confident for taking the bigger steps in life. Christos, direct descendent from the fathers and mothers of science, you have a bright future ahead of you. Tim, as the kakhiel of our group, you are often misunderstood. Never stop fooling other people. Thomas S, since you have joined our group, summers have become so much drier. Nick, prophet of deep learning, thank you for sharing some of those insights. Margot, you are my favourite badminton partner. Thank you for taking such great care of Kevin. Lieven, you made me realize that science and religion can go hand in hand. Thank you for that. John, I was never able to beat you in terms of being at the office early. I am sure your hard work will make a big impact for the remote healthcare system that you envision. Kaat, I have really enjoyed the many talks we had. You are one of the kindest persons that I have ever met. Simon G, the field of AAD is lucky to have the full attention of such a talented researcher as you are. Your humble attitude is an inspiration. Servaas, thank you for introducing me to the wonderful world of rock climbing. We have many more challenges ahead of us. Amir, thank you for all of our software-related discussions, they were of much added value to my work. Adrian, we haven't seen or heard from each other in a while, but you are still on my mind. I hope we get to meet again one day. Stijn, thank you for finally paying for your sandwich in a timely manner. Nicolas, Elisabeth, Laura, Miguel, Joran, mostly due to the working from home policy, we never really got to know each other all that well. You are the future of BIOMED and I know you will ensure the continuation of this great group. If I have forgotten you here, I will blame it on baby related sleep deprivation. Many have come, many have left. You have all been great colleagues without exception.

If only the above people were the ones that I had many interesting discussions and coffee with, my productivity would have been only minimally affected. Unfortunately, there are many more great people around STADIUS. Just to name one: Jeroen. We go way back. This is likely the end of the professional side of our ten year long friendship. I hope that we can share many more moments in the future. To minimize the risk of forgetting even more people, let me thank all of the people from Marc's group, Toon's group, Lieven's group, Bart's group, Panos' group and Johan's group for the many nice moments. A special shout-out goes to Randy and Nicco for coping with my snoring in Calgary. I blame it on that dusty carpet.

I would also like to thank my colleagues at NERF for the many interesting lab

meetings that provided me with a lot of context. Thank you Chae Young, Fred, Hanna, JJ, Kasia, Lies, Marine, Rik, Davide and others.

Life is more than work. Fortunately, after work I can come home to my wonderful family and friends. I consider myself incredibly fortunate to have so many wonderful people around me.

Een speciaal bedankje gaat uit richting Ward. Bedankt om de laatste maanden zo een loyale looppartner te zijn. Bij deze wil ik mij excuseren voor het oeverloze gezeur omtrent de thesis.

Ik wil ook mijn moeder en vader bedanken. Jullie hebben mij jarenlang op talloze vlakken ondersteund, niet in het minst financieel. Zonder mijn vader zijn atypische manier van opvoeden, was ik nooit zo ver bergop geraakt. 'Pap'. Ik denk dat ik mijn doorzettingsvermogen deels daaraan te danken heb. Bedankt moeder, wij stromen nog altijd over van de liefde die jij ons gegeven hebt (en nog steeds geeft). Ook in het bijzonder dank voor de talloze uren die jij gespendeerd hebt aan het maken van het maatpak voor de doctoraatsverdediging. Ik wil ook mijn zussen, schoonbroers en nichtjes bedanken. Ik voel me goed als jullie in de buurt zijn.

Last but not least wil ik mijn meisjes bedanken. Lore, mijn favoriete moment van de dag is om bij jou en Jozefien thuis te komen. Bedankt om deel uit te maken van ons warme gezinnetje. Mijn leven voelt geslaagd louter door jullie toedoen en dat helpt echt om de moeilijke dagen in perspectief te plaatsen. Dankzij jullie kan ik dat tikkeltje meer. Ik ben er zeker van dat dit doctoraatseinde voor ons het begin is van een mooie toekomst.

– Jasper Wouters, September 2020

Abstract

Despite many neuroscientific breakthroughs, it remains largely unknown how brain activity supports cognition. Obtaining such a fundamental understanding of the brain has the potential to foster new clinical applications aimed at improving functional deficits resulting from a 'malfunctioning' brain. The identification of causal relationships between brain activity and cognition depends on the real-time characterization of neural activity. By implanting extracellular electrodes into the brain, potential changes can be recorded, which are a reflection of the ongoing neural activity. Such recorded voltage traces contain, among other neural activity signals and patterns, information about the action potentials of the neurons that are close to the implanted electrodes, which is also referred to as the multi-unit activity.

Although the multi-unit activity holds information about the activity of individual neurons, this information is not readily available. Moreover, the multi-unit activity often contains a mixture of action potentials, also referred to as spikes, from several neurons. In this work, we focus on the development and validation of signal processing algorithms to extract individual spike times from multi-unit activity recordings and group those spike times according to their putative neurons. The algorithms that perform this specific extraction task are generally known as spike sorting algorithms. The resulting single-neuron spike trains are usually further processed to decode the information that is encoded within the spike train. Although spike sorting and decoding hold great promise towards a better understanding of the brain, recent research in spike sorting has resulted in computationally intensive approaches that are not suitable for real-time use. On the other hand, available spike sorting algorithms intended for online use are often limited in terms of coping with realistic experimental conditions, e.g., the occurrence of overlapping spikes, under which they fail.

The first point of focus of this work is the development of a spike sorting methodology with low computational complexity, which explicitly accounts for overlapping spikes. We take a common threshold-based approach where a single-neuron spike train is obtained through the use of a linear finite impulse response filter applied to the multi-unit activity, followed by a comparison of the filter output against a threshold value. Such a low-complexity sorting architecture is useful in the context of online and/or embedded sorting. We propose a novel class of filter design methods for the threshold-based sorting architecture that are based on signal-to-peak-interference ratio (SPIR) optimality. This new class of filters enables threshold-based spike sorting, as compared to earlier approaches based on signal-to-noise ratio (SNR) optimality or signal-to-interference-plus-noise ratio (SINR) optimality, which are insufficiently

discriminative or are faced with practical limitations. We show that the proposed methodology outperforms existing methods on an extensive set of validation data. Furthermore, we shown that SPIR-optimality is related to the theory of support vector machines, such that the SPIR-optimal filter can be interpreted as a maximum-margin matched filter that can be useful in other pattern recognition tasks where the computational complexity is restricted.

Related to the design of optimal linear filters for use in spike sorting, we present a study that is aimed at better understanding the need for linear filter design regularization. We propose a data-driven regularization technique that is shown to outperform the state-of-the-art. Furthermore, the proposed methodology makes use of an interpretable hyperparameter, which aids in the regularization tuning process. The proposed regularization technique is then integrated with SPIR-optimal filter design, where it is shown to have the additional property of transforming the filter design into an unconstrained optimization problem.

The second point of focus in this work is related to the validation of spike sorting algorithms. Such validation requires the availability of ground truth multi-unit activity recordings, i.e., recordings for which the individual spike times of one or more spike trains are completely known. State-of-the art approaches for the generation of such ground truth data are either expensive, and/or require expert knowledge, or result in unrealistic recordings. In this work we present a user-friendly tool for the creation of ground truth multi-unit activity recordings. The graphical tool implements a hybrid ground truth model, which enables the transformation of real extracellular recordings into ground truth recordings. As such, realistic ground truth data can be easily obtained without the need for costly procedures or complex computational modelling theory. Therefore, besides its usefulness for validating spike sorting algorithms during development, such data can also be leveraged by spike sorting users for algorithm selection and parameter tuning to improve spike sorting performance for their specific recording setting. The transformation is based on the availability of manually verified initial spike sorting results. The tool comes with additional routines to aid the quantification of spike sorting performance on such ground truth data. Furthermore, the tool has been integrated within a wider spike sorting ecosystem to accelerate its adoption. A case study is presented in which its usefulness for the spike sorting practice is demonstrated.

As a third point of focus, we propose an innovative method for resolving spike overlap directly in the feature space, as opposed to other strategies that are based on the design of linear filters that depend on the availability of an initial clustering. The methodology consists of the design of a specialized spike embedding in which overlap behaves as a linear operation. The proposed methodology holds great promise for a future clustering-based spike sorting pipeline that can handle various signal characteristics that are often encountered in the practice, such as spike overlap, bursting, and drift. Through the elimination of the filtering related complexities from the pipeline, this approach could be considered as an alternative pipeline that enables low-complexity spike sorting capable of resolving spike overlap.

Beknopte samenvatting

Ondanks vele neurowetenschappelijke doorbraken, blijft het grotendeels ongekend hoe hersenactiviteit aanleiding geeft tot cognitie. Het bekomen van een fundamenteel inzicht in de werking van het brein, kan tot nieuwe klinische toepassingen leiden die gericht zijn op het verbeteren van functionele tekorten als gevolg van een 'slecht werkend' brein. De identificatie van causale verbanden tussen hersenactiviteit en cognitie vereist de *real-time* characterizatie van hersenactiviteit. Door extracellulaire elektroden in de hersenen te implanteren, kunnen veranderingen in de spanningspotentialen worden geregistreerd die een weerspiegeling zijn van de neurale activiteit. Dergelijke spanningssignalen bevatten, naast andere neurale patronen, informatie over de actiepotentialen van neuronen in de omgeving van de geïmplanteerde elektroden, ook wel multi-neuron activiteit genoemd.

Hoewel de multi-neuron activiteit informatie bevat over de activiteit van individuele neuronen, is deze informatie niet direct beschikbaar. De multi-neuron activiteit bevat dikwijls een mengsel van actiepotentialen, ook wel spanningspieken of spikes genoemd, van verschillende neuronen. In dit werk richten we ons op de ontwikkeling en validatie van signaalverwerkingsalgoritmen die de individuele tijdstippen waarop neuronen een spike genereren, extraheren uit de multi-neuron activiteit en vervolgens deze tijdstippen groeperen per individueel neuron. De algoritmen die deze specifieke extractietaak uitvoeren, staan algemeen bekend als spike-sorteringsalgoritmen. De resulterende tijdsinformatie over de activiteit van individuele neuronen, wordt meestal verder verwerkt om de cognitieve informatie te decoderen die geëncodeerd is in deze tijdsinformatie. Hoewel spike-sortering en decodering veelbelovend zijn voor het bekomen van een beter inzicht in de werking van de hersenen, zijn de huidige algoritmen veelal ongeschikt voor real-time gebruik omwille van de hoge rekencomplexiteit. Algortimen die wel bedoeld zijn voor real-time gebruik, zijn vaak onvoldoende robuust voor gebruik in realistische experimentele condities, zoals het voorkomen van overlappende spikes wanneer twee of meerdere neuronen gelijktijdig actief zijn.

Het eerste aandachtspunt van dit werk is de ontwikkeling van een methodologie met een lage rekencomplexiteit voor het sorteren van spikes, terwijl expliciet rekening wordt gehouden met overlappende spikes. Er wordt een gekende filter-en-vergelijk-aanpak gevolgd, waarbij de activiteit van een individueel neuron wordt verkregen door het toepassen van een lineair filter met eindige impulsrespons op de multi-neuron activiteit, gevolgd door het vergelijken van de filteruitgang met een eenvoudige drempelwaarde. Dergelijke sorteringsarchitectuur met lage complexiteit is nuttig in de context van online en/of ingebedde sortering. Voor deze sorteringsarchitectuur stellen we een nieuwe klasse van filterontwerpmethoden voor die gebaseerd zijn op signaal-tot-piek-interferentieverhouding (SPIR) optimaliteit. Deze nieuwe klasse van filters maakt een filter-en-vergelijk-aanpak mogelijk. Dit in tegenstelling tot eerdere benaderingen op basis van signaal-ruisverhouding (SNR) optimaliteit of signaal-tot-

interferentie-plus-ruisverhouding (SINR) optimaliteit, die onvoldoende discriminatief zijn voor het beoogde gebruik en/of worden geconfronteerd met praktische beperkingen. We tonen aan dat de voorgestelde methodologie beter presteert dan de bestaande methoden door middel van een uitgebreide validatie. Verder wordt aangetoond dat SPIR-optimaliteit gerelateerd is aan de theorie van *support vector machines*, zodat het SPIR-optimale filter kan worden geïnterpreteerd als een filter met een maximale marge dat kan worden gebruikt voor andere patroonherkenningsdoeleinden waarbij de computationele complexiteit beperkt is.

Gerelateerd aan het ontwerp van optimale lineaire filters voor gebruik bij het sorteren van spikes, presenteren we een expliciete studie die is gericht op het beter begrijpen van de behoefte aan regularisatie van dergelijke lineaire filterontwerpen. We stellen een datagedreven regularisatietechniek voor die beter presteert dan de state-of-the-art. Bovendien maakt de voorgestelde methodologie gebruik van een interpreteerbare hyperparameter, die helpt bij het instellen van de regularisatie. De voorgestelde regularisatietechniek wordt vervolgens geïntegreerd in het SPIR-optimaal filterontwerp, waar dit leidt tot de transformatie van het filterontwerp in een optimalisatieprobleem zonder randvoorwaarden.

Het tweede aandachtspunt in dit werk is gerelateerd aan de validatie van spike- sorteringsalgoritmen. Zulke validatie vereist de beschikbaarheid van multi-neuron validatie-activiteit waarvan de onderliggende tijdsinformatie van één of meerdere neuronen volledig gekend is. De huidige methoden voor het genereren van zulke data zijn kostelijk en/of vereisen expertenkennis om te resulteren in realistische signalen. In deze thesis presenteren we een gebruiksvriendelijke tool voor het genereren van multi-neuron validatie-activiteit. De grafische tool implementeert een hybride model, dat de transformatie van echte extracellulaire potentialen naar validatie-activiteit mogelijk maakt. Als zodanig kan een realistische validatie-activiteit worden verkregen zonder gebruik te maken van complexe computationele modelleringstheorie. Dergelijke validatie-activiteit kan naast het valideren van spike-sorteringsalgoritmen, ook worden benut door gebruikers van dergelijke algoritmen om de algoritme-selectie en de parameterinstelling te sturen met als doel het verbeteren van de sorteringsperformantie binnen hun specifieke experimentele context. De transformatie is gebaseerd op de beschikbaarheid van handmatig geverifieerde sorteringsresultaten. De tool wordt geleverd met extra routines om de sorteerresultaten op dergelijke validatie-activiteit te kwantificeren. Bovendien is de tool geïntegreerd in een breder ecosysteem van sorteringsalgoritmen om het gebruik van de tool te bevorderen. Er wordt een casestudy gepresenteerd waarin het nut van onze tool voor gebruikers van sorteringsalgoritmen wordt aangetoond.

Als derde aandachtspunt stellen we een innovatieve methode voor om overlappende spikes rechstreeks in de kenmerkenruimte te behandelen, dit in tegenstelling tot andere strategieën die gebaseerd zijn op het ontwerp van lineaire filters die de beschikbaarheid van initiële clusteringresultaten veronderstellen. De methodologie bestaat uit het ontwerp van een gespecialiseerde kenmerkenruimte waarin overlap zich gedraagt als een lineaire operatie. De voorgestelde methodologie is veelbelovend voor een toekomstige op clustering gebaseerde spike-sorteringspijplijn die kan omgaan met verschillende signaalkenmerken die vaak in de praktijk worden aangetroffen. Door de filtergerelateerde complexiteit uit de pijplijn te verwijderen, zou deze benadering beschouwd kunnen worden als een alternatieve pijplijn met lage computationele complexiteit die het sorteren van spikes mogelijk maakt in de context van overlappende spikes.

List of Abbreviations

ASIC application-specific integrated ciruit.

BMI brain-machine interface.

BOTM Bayes optimal template matching.

DTM discriminative template matching.

FIR finite impulse response.

FP false positive.

FPGA field-programmable gate array.

GAN generative adversarial network.

GEVD generalized eigenvalue decomposition.

GUI graphical user interface.

HD high-density.

IQR interquartile range.

KS KiloSort.

MAP maximum a posteriori.

MLP multi-layer perceptron.

PCA principal component analysis.

PSNR peak-signal-to-noise ratio.

ReLU rectified linear unit.

SC Spyking Circus.

SIC subtractive interference cancellation.

SINR signal-to-interference-plus-noise ratio.

SNR signal-to-noise ratio.

SPIR signal-to-peak-interference ratio.

SVM support vector machine.

TP true positive.

ZCA zero-phase component analysis.

List of Symbols

$\mathcal{A}_i$	Pattern source activation times for pattern source i, generalization of $\mathcal{T}_i$.
β_i^s	Least squares template fitting factor for spike at sample time s of neuron i.
$\delta[k]$	Kronecker delta function, i.e., $\delta[k] = 1$ if $k = 0$ and $\delta[k] = 0$ if $k \neq 0$.
E^c	Template energy on channel c.
η	Interference threshold tuning parameter in interference covariance heuristic.
f_c	Cut-off frequency of a high-pass filter.
$\mathbf{f}_i$	Finite impulse response filter coefficients associated with a neuron i.
$\mathbf{f}_i^{\mathrm{MF}}$	Matched filter coefficients associated with a neuron i.
$\mathbf{f}_i^{\mathrm{CSO}}$	Convex SPIR-optimal filter coefficients associated with a pattern source i.
$\mathbf{f}_i^{\mathrm{SB}}$	SPIR-based filter coefficients associated with a pattern source i.
$\mathbf{f}_i^{\mathrm{SO}}$	SPIR-optimal filter coefficients associated with a pattern source i.
$\mathbf{f}_i^{\mathrm{WMF}}$	Weighted matched filter coefficients associated with a pattern source i.
f_s	Sampling frequency.
γ	Interference threshold fraction.
$\mathbf{h}^c[k]$	Hybrid extracellular recording at sample time k.
$\mathbf{I}$	Identity matrix.

j^s Random spike jitter map.

K Target pattern output response.

L Temporal delay line length.

l_i^β Lower bound on template fitting factor for the neuron i.

u_i^β Upper bound on template fitting factor for the neuron i.

$\mathcal{N}$ Set of sample times at which there is no neuronal activity.

N Number of electrodes on a neural recording device.

N_i Number of electrodes on a neural recording device that are relevant for the retrieval of the single-unit spike train of a neuron i.

$\mathbf{n}[k]$ Background noise signal at time k.

$\bar{\mathbf{n}}[k]$ Temporal delay line extension of $\mathbf{n}[k]$.

$\mathbf{p}_i[k]$ Pattern source signal from pattern source i.

$\boldsymbol{\pi}_i$ Pattern or template of a pattern source i, generalization of $\boldsymbol{\tau}_i$.

ψ Subspace regularization power fraction.

$\mathbf{R}_{\mathbf{zz}}$ Covariance matrix of a multi-dimensional signal $\mathbf{z}[k]$.

$\hat{\mathbf{R}}_{\mathbf{zz}}$ Covariance matrix estimate of a multi-dimensional signal $\mathbf{z}[k]$.

$\mathbf{s}^c[k]$ Extracellular recording from which the initial units have been subtracted at sample time k.

σ_i Parameter controlling for the signal-to-noise ratio of the reinserted unit associated with neuron i in the hybrid recording.

$\mathcal{T}$ Set of sample times that are used for the training of a linear filter (e.g., for the estimation of signal covariance), not to be confused with $\mathcal{T}_i$, which denotes the set of spike times used for the estimation of a spike template.

T Duration in samples of the extracellular recording.

$\mathcal{T}_i$ Example spike times of a neuron i that can be used for, e.g., template estimation.

T_i Detection threshold for threshold-based pattern recognition associated with pattern source i.

$\boldsymbol{\tau}_i$	Spike template estimate for a neuron i.
$\mathbf{X}$	Extracellular recording matrix, containing T rows, where each row corresponds to $\mathbf{x}[k]$.
$x^c[k]$	Extracellular recording observation on channel c at sample time k.
$\bar{\mathbf{x}}^c[k]$	Temporal delay line from $x^c[k]$ of length L.
$\mathbf{x}[k]$	Extracellular recording at sample time k.
$\mathbf{x}_i[k]$	Extracellular recording at sample time k for the channels that are relevant for the retrieval of the single-unit spike train of a neuron i.
$\overline{\mathbf{x}}_i[k]$	Temporal delay line extension of $\mathbf{x}_i[k]$.
$y_i[k]$	Finite impulse response filter output at sample time k.
ζ	Target safe zone tuning parameter in interference covariance heuristic.

Contents

Abstract v

Beknopte samenvatting vii

List of Abbreviations x

List of Symbols xiii

Contents xv

List of Figures xxi

List of Tables xxix

1 Spike sorting primer 1

1.1 Neuronal action potentials 1

1.2 Tapping into neural activity 4

1.3 Spike sorting 7

1.4 Violation of the stationarity assumption 10

1.5 Resolving spike overlap 11

1.5.1 Threshold-based spike sorting 11

1.5.2 Opportunities for real-time sorting 13

1.5.3 Non-discriminative matched filter problem 13
1.6 General challenges in spike sorting 15
1.6.1 Lack of ground truth data 15
1.6.2 Coping with multiple stationarity violations 15
1.7 Thesis outline 16
1.8 Appendix 18
1.8.1 Spike sorting with bursting neurons 18
1.8.2 Spike sorting with electrode drift 18

2 Discriminative template matching with suppression of interfering spikes 19

2.1 Introduction 20
2.2 Threshold-based spike sorting pipeline 23
2.2.1 Threshold-based spike sorting 23
2.2.2 Matched filtering-based template matching 26
2.3 Discriminative template matching 27
2.3.1 Filter design for SPIR maximization 27
2.3.2 Illustrative example 29
2.3.3 Estimation of the interference covariance matrix 31
2.4 Validation 34
2.4.1 Description of validation data set 34
2.4.2 Validation parameters 36
2.4.3 Signal-to-peak-interference maximization 36
2.4.4 Spike sorting performance 37
2.4.5 Robustness to noise 41
2.5 Discussion and Conclusion 43
2.6 Appendix 45
2.6.1 Derivation of the optimal filter 45

2.6.2 Robustness against target leakage 47

3 Data-driven regularization for template matching in spike sorting 49

3.1 Introduction 50

3.2 Template matching 51

3.3 Regularization 52

3.3.1 Use the signal+noise covariance matrix 54

3.3.2 Filter design in PCA subspace 54

3.3.3 Filter design in template & PCA subspace 55

3.4 Experiments 56

3.5 Conclusion 57

4 SHYBRID 59

4.1 Introduction 60

4.2 Hybrid ground-truth model 64

4.3 GUI for hybrid data generation 68

4.3.1 Input data 69

4.3.2 Visualizing the template 71

4.3.3 Inspecting template fit 72

4.3.4 Relocating the unit 73

4.3.5 Auto hybridization 76

4.3.6 Importing / exporting templates 76

4.3.7 Automatic tools for spike sorting algorithm assessment . 76

4.3.8 SpikeInterface integration 78

4.4 Case study: comparing spike sorting algorithms and tuning parameters 78

4.5 Discussion and Conclusion 83

4.6 Appendix 84

4.6.1 Auto hybridization fitting factor bounds 84
4.6.2 Auto hybridization random unit relocation 85
4.6.3 External template import 85
4.6.4 Automatic merging 86

5 Multi-pattern recognition through maximization of signal-to-peak-interference ratio 91

5.1 Introduction 92
5.2 Problem statement 94
5.3 Filter design review 95
5.3.1 Matched filtering 95
5.3.2 Matched filtering with source weighting 97
5.3.3 SPIR-based filtering 98
5.4 Non-convex SPIR-optimal filtering 100
5.4.1 Definition of SPIR-optimal filtering 100
5.4.2 Choice of K 101
5.4.3 Bounds on γ 102
5.5 Convex SPIR-optimal filtering 103
5.5.1 Definition of the convex SPIR-optimal filter 103
5.5.2 SVM interpretation 105
5.6 Subspace projection regularization 108
5.7 Case study: max-SPIR filters for neural spike sorting 110
5.8 Conclusion and Discussion 118

6 Resolving spike overlap in the feature space 121

6.1 Introduction 122
6.2 Problem statement 123
6.3 Method 124

6.4 Data augmentation 126
6.5 Initial experiments 127
6.6 Improved spike overlap model 130
6.7 Conclusion and discussion 133

7 Conclusion **135**

7.1 Contributions 135
7.2 Open challenges 139

Bibliography **145**

List of publications **159**

List of Figures

1.1 Schematic representation of a neuron where the major parts involved in the generation of an action potential are indicated. [adapted from Quasar Jarosz – CC SA 3.0 license] 2

1.2 Transmembrane potential over time at the axon initial segment during an action potential period. The different phases of the action potential are indicated on the graph. [Wikimedia Commons – CC SA 3.0 license] 3

1.3 The extracellular potential changes at various locations that are indicated by the black dots (and referenced at infinity) are shown in response to an action potential of the visualized neuron model. 5

1.4 Example voltage traces from a multi-channel extracellular recording made with a high-density neural probe with an electrode pitch of 20 µm. The voltage traces have undergone a high-pass filtering with a cut-off frequency of 300 Hz. The horizontal axis represents time, whereas along the vertical axis the voltage traces from several recording channels are shown. The brief troughs in the voltage traces are the extracellular reflections of neuronal action potentials, also known as spikes. 7

1.5 A generic unsupervised approach for solving the spike sorting problem consists of three major processing steps: 1. spike snippets are extracted from the extracellular recording through spike detection, 2. every spike snippet is projected into a single-unit promoting feature space, 3. a clustering analysis is performed in the feature space to identify single-unit spike trains. [adapted from [33]] 9

1.6 Schematic representation of a spike sorting pipeline as introduced in Section 1.3 (blue) that is augmented with template matching blocks (orange) for resolving spike overlap. 12

1.7 Detailed schematic representation of a matched filtering approach for resolving spike overlap where each filter output is processed independently from one another. 12

1.8 Detailed schematic representation of an iterative matched filtering approach for resolving spike overlap when the individual matched filters are not sufficiently discriminative. 14

1.9 Thesis organization overview. 16

2.1 A schematic representation of the *threshold-based spike sorting* paradigm. For every sortable neuron, a separate processing pipeline needs to be implemented. Every pipeline includes preprocessing, template matching and thresholding, and generates single-unit activity at its output. A detailed view of such a pipeline (bottom) reveals the finite impulse response (FIR) template matching filters that are considered in this chapter. . 24

2.2 **a** Drawing of the simulated setup. The setup contains three neocortical layer 5b morphologically detailed pyramidal cells (cell 1: blue, cell 2: green, cell 3: red) and a 128-channel high density probe of which only the electrode locations are shown by black dots. **b** 35ms of simulated data for 20 out of 128 channels. Times at which the simulated neurons generate spikes are highlighted by the active cell's corresponding color. From this it can be seen that each individual neuron is active three times in the given data slice. **c** Matched filter-based template matching output of the three filters trained on the individual templates which were extracted using the ground truth spike times. **d** Ground truth spike times for each cell. **e** Discriminative template matching output of the three filters each trained using all three templates. 30

2.3 Graphical representation of the proposed algorithm for estimating the interference covariance matrix needed for the discriminative template matching filter design. **1.** The prior knowledge required for the algorithm exists of a set of example spike times for the target neuron. **2.** A matched filter is computed and applied to the recording, using the spike template as filter coefficients. The effect of background noise on the filter output is depicted by bold signal segments. **3.** In order to prevent leakage of spikes from the target neuron into the interference covariance matrix estimate, a target safe zone is determined. Each peak that falls within this target safe zone is excluded for the estimation of the interference covariance matrix. **4.** Based on the filter output statistics the noise floor is estimated. **5.** By combining both the target safe zone and the noise floor, the interference threshold is calculated. **6.** All peaks that have their maximum above the interference threshold and outside the target safe zone are used for estimating the interference covariance matrix. The signal segments involved in the estimation of the interference covariance matrix are shaded. 32

2.4 The output of both the matched-filtering based template matching (blue) and the discriminative template matching with interference suppression (green) filter applied to in-vivo recorded data is shown. Both filter outputs are normalized with respect to their standard deviation. The ground truth spike time of the target neuron in this segment, is indicated by a red asterisk. The double arrows indicate the signal-to-peak-interference ratio for both outputs. 37

2.5 **a** The spike sorting sensitivity and precision for all analyzed ground truth neurons are shown. For each neuron four different algorithms were tested: template matching threshold-based (blue), discriminative template matching with interference suppression threshold-based (green), Bayes optimal template matching [31] (red), and Spyking Circus [137] (yellow). **b** Summary of precision (blue) and sensitivity (green) over all neurons for the four methods. The error bars indicate a 95 percent confidence interval. 38

2.6 Sensitivity and precision for discriminative template matching and Bayes optimal template matching under adverse noise conditions. If no spikes are detected, the precision metric is undefined. By convention we depict an undefined precision by a cross marker located at zero. 42

2.7 Sensitivity and precision for discriminative template matching under adverse noise conditions using only one discriminative template matching filter and threshold trained on 5 dB PSNR data. 43

3.1 Power along every principal component (red) for the spatio-temporally expanded recording data of neuron 1 and neuron 2. The cumulative normalized power (dashed blue) is also shown and the ψ-fraction point (see Section 3.3.2) is indicated by the black arrow. 53

3.2 The F_1-score is shown as a measure of spike sorting performance for every ground truth neuron used. Three different regularization methods for calculating the template matching filters are compared: template & PCA subspace regularization (green), PCA subspace regularization (red), and spatial-only regularization (blue). 58

4.1 **A**: The probe grid model assumes that all electrodes (green dots) are located on intersections of a grid. The distance between grid lines along a certain axis is assumed to be constant. Not every intersection has to be populated by an electrode (grey dots). **B**: Moving the template over the probe consist of shifting the template over the grid. The black arrows represent shifting a template one position to the left. At the right side of the probe extrapolated waveforms (blue dots and arrows) are introduced into the template, while on the left side, a part of the template is lost. Missing channels that are shifted onto working electrodes contain a value obtained through interpolation (orange dots and arrows). 67

4.2 Spike templates (and recording snippets) are visually organized according to the channel geometry of the recording device. The different rows and columns in the visualization correspond to the rows and columns of the recording device. The waveform duration (horizontal axes) is controlled by the user. The waveform amplitude (vertical axes) is automatically scaled to provide a clean visualization. The relative waveform amplitude with respect to the recording noise is available from the SNR that is calculated and shown for every spike template. 70

4.3 **A**: Template temporal window length given in milliseconds. **B**: Altering the zero force fraction changes the spatial extent of the template. 72

4.4 For every recording chunk (shown in red) that is used for the template estimation, an optimal template scaling factor is determined. The *inspect template fit* view allows the user to assess how good of a model the scaled template (shown in blue) is for the current cluster. The organization of the visualization is identical to Figure 4.2. The red profile at the bottom of the plotting area is a visual representation of the ordered scaling factors for all recording chunks. The fine dashed line (black) indicates the unit fitting factor, i.e., $\beta_i^s = 1$. 74

4.5 A unit's spike template can be moved over the probe by using the arrow controls on the left panel. The average spike rate on every channel is indicated by the color in which the channel is plotted. Moving a template into a silent region will likely result in a hybrid recording that is easier to sort. The organization of the visualization is identical to Figure 4.2. 75

4.6 Example templates (blue) of hybrid units for both the SC-informed and KS-informed hybrid data are shown. The two templates in the top row originate from hybrid units that are recovered (i.e., F_1-score > 0.9) by both spike sorting algorithms (see Table 4.1 and 4.2 for numerical details). The bottom row contains two templates from hybrid units that were only partially recovered by both algorithms. An example optimally scaled spike signal chunk (red) is shown behind each template to allow for a visual assessment of the relative noise levels with respect to the template power. The individual unit template plots are generated through the SHYBRID *export plot* functionality. 80

4.7 **A**: For both the SC- and KS-informed hybrid ground-truth data, the number of hybrid single-unit spike trains that are recovered by the different spike sorting algorithms is shown. The total number of hybrid ground-truth spike trains, i.e, the maximum number that could have been recovered, is shown in green. **B**: For both the SC-and KS-informed hybrid ground-truth data, the average number of cluster merges over the single-unit spike trains is shown for the different spike sorting algorithms that are used. The error bars indicate the standard deviation. **C** and **D**: Spike sorting performance metrics for the SC- and KS-informed hybrid data, respectively. As performance metrics the F_1-score, precision and recall are shown. The error bars indicate the standard deviation across the recovered single-unit spike trains. 81

5.1 Qualitative comparison between an SNR- and SPIR-optimal filter output. The SPIR-optimal filter has an increased SPIR compared to the SNR-optimal one, at the acceptable cost of a decrease in SNR, enabling a more robust filter-and-threshold-based pattern recognition for the SPIR-optimal filter output. 99

5.2 The shifted sigmoid weighting function only "behaves" as a step function for values of K that are sufficiently large. For $K = 10$ (green) and $\gamma = 0.1$, the shifted sigmoid does not reach zero for any $s \in [0, \gamma K]$. For $K = 1000$ (orange) and $\gamma = 0.1$ the shifted sigmoid is a good approximation to a step function. 102

5.3 A geometric interpretation of the convex SPIR-optimal optimization problem in two dimensions ($\mathbf{f} \in \mathbb{R}^2$). The shaded area represents the margin area. The green dot represents the pattern of interest $\boldsymbol{\pi}_i$. The other dots represent the training points $\mathbf{x}_k$, where the orange dots represent the support vectors. The equality constraint (5.23b) is satisfied if the separating line (in black) is tangent to the half ρ-circle (in blue). 108

5.4 The effect on the spike sorting performance of the choice of regularization subspace power fraction is studied. The average precision and recall for all analysed filter design methods are shown for both the non-interfering (top) and interfering (bottom) group. 117

5.5 The relative dimensionality of the filter design subspace is shown in function of the regularization subspace power fraction. . . . 118

6.1 3 orthogonal viewpoints on a 3-dimensional feature space with 3 neurons. **Left**: Overlapping spikes in a (linear) principal component feature map do not form separable clusters and do not have a clear relationship with the single-neuron clusters. **Right**: The proposed neural network feature map is capable of resolving overlapping spikes. 125

6.2 The proposed neural network feature map is implemented as a multi-layer perceptron with two hidden layers. 127

6.3 On the horizontal axis the adjusted Rand index (with random jitter for readability) is depicted and on the vertical axis the center prediction error is shown. Every blue dot represents the feature map performance for a pair of test neurons. 129

6.4 Extended architecture that is used in the context of the improved model. 132

6.5 On the horizontal axis the adjusted Rand index is shown and on the vertical axis the center prediction error is shown. Every blue dot represents the feature map performance for a pair of test neurons. 133

List of Tables

4.1 Spike sorting performance metrics for the three separate spike sorting runs (SC, KS, KS more clusters) applied on the SC-informed hybrid ground-truth data. For every ground-truth unit in every spike sorting run, the F_1-score, precision, recall and number of cluster merges is shown. Rows with bold typesetting indicate the ground-truth units that were recovered as single-unit spike trains. This table is generated by SHYBRID. 88

4.2 Spike sorting performance metrics for the three separate spike sorting runs (SC, KS, KS more clusters) applied on the KS-informed hybrid ground-truth data. For every ground-truth unit in every spike sorting run, the F_1-score, precision, recall and number of cluster merges is shown. Rows with bold typesetting indicate the ground-truth units that were recovered as single-unit spike trains. This table is generated by SHYBRID. 89

5.1 Filter-and-threshold-based spike sorting performance in terms of precision and recall for the ground-truth neurons in the non-interfering group. Every row corresponds to a neuron, whereas the grouped pairs of columns correspond to the different filter design methods presented in this chapter. 112

5.2 Filter-and-threshold-based spike sorting performance in terms of precision and recall for the ground-truth neurons in the interfering group. Every row corresponds to a neuron, whereas the grouped pairs of columns correspond to the different filter design methods presented in this chapter. The performance metric values shown in red correspond to values lower than 65%, whereas the values shown in green correspond to values higher than 80%. 113

Chapter 1

Spike sorting primer

1.1 Neuronal action potentials

We are our brain. Although this statement, which is the title of a recent popular science book, is a neuro-essentialist view on the role of the brain, the brain is generally believed to be a protagonist in our existence. It is known that the brain is responsible for various function including sensation, perception, decision making, learning, memory, language, and many more [138]. The brain is part of the larger nervous system, which is spread throughout the body, innervating our organs and tissues. Roughly speaking, the nervous system can be seen as an information processing and transmission system, coordinating many vital processes.

To perform its specific function, the nervous system is built from specialized cells. One such type of cells are known as neurons [17]. Neurons are information processing and transmission enabling cells. It is believed that complex behaviour is supported by neuronal activity that arises from organizing such neurons in large networks. The human brain contains approximately 70 to 80 billion neurons [127], where each neuron has on average between 1000 and 10000 connections, also referred to as synapses, with other neurons.

A single neuron, as schematically depicted in Figure 1.1, can be interpreted as an input-output system: its dendrites can receive synaptic input from other neurons, while the axon is considered the output stage that enables the transmission of information to other neurons. A central concept in neuronal physiology is the generation and propagation of action potentials, which is essentially the transformation of the neuronal input to output.

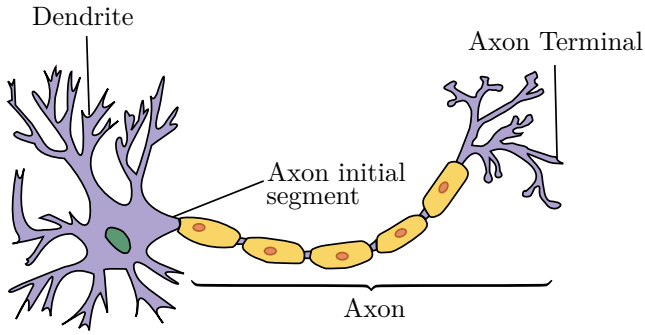

Figure 1.1: Schematic representation of a neuron where the major parts involved in the generation of an action potential are indicated. [adapted from Quasar Jarosz – CC SA 3.0 license]

When a neuron is in its resting state, i.e., when it does not receive any input from other neurons, a negative membrane potential difference is maintained between the inside and outside of the cell. This potential difference or transmembrane potential is due to an ionic charge imbalance between the intracellular and extracellular medium through selective permeability of ion channels. A typical value for this resting potential is around -70 mV [24].

When a neuron receives synaptic input at its dendrites, the ionic charge imbalance can be altered by an exchange of ions between the intracellular and extracellular medium through ligand-gated channels. Those channels open in response to chemical messengers or neurotransmitters, e.g., produced by nearby neurons, binding to the ion channel's receptors. A change in the charge imbalance will alter the transmembrane potential. A transmembrane potential that becomes more negative than the resting potential is referred to as hyperpolarization, whereas a potential that becomes less negative than the resting potential is referred to as depolarization.

Once a depolarizing transmembrane potential near the axon initial segment, which is rich in voltage-gated ion channels, crosses a specific voltage threshold, an action potential is initiated. An action potential is a brief local variation in the transmembrane potential as is shown in Figure 1.2. The action potential is initiated due to the opening of voltage-gated Na^+ channels that lead to an influx of Na^+ into the cell, which further depolarizes the local transmembrane potential, eventually resulting in a positive transmembrane potential and an inactivation of the Na^+ channels. Next, voltage-gated K^+ channels open and K^+ flows out of the cell leading to a local repolarization and eventually a hyperpolarization

of the transmembrane potential. Because of this hyperpolarizing overshoot, the cell is less likely to generate another action potential in a short period after the occurrence of an action potential. This period is referred to as the refractory period. Once the K^+ channels close, the transmembrane potential will return to its resting state.

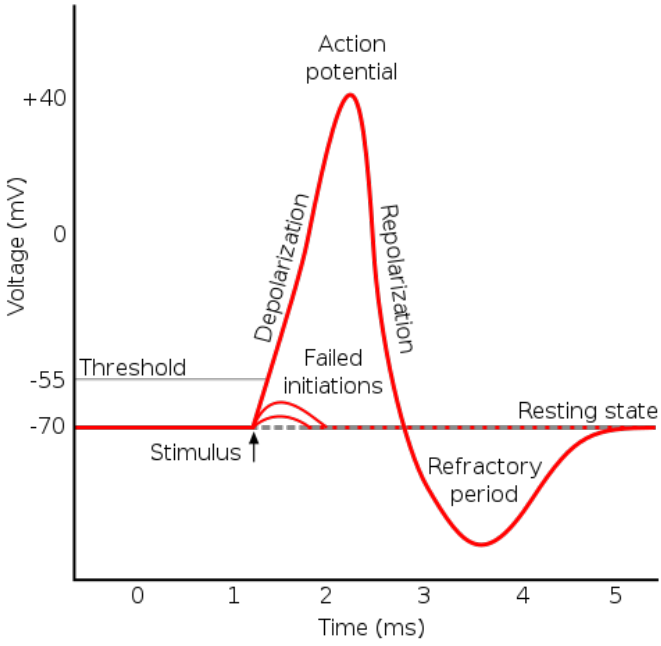

Figure 1.2: Transmembrane potential over time at the axon initial segment during an action potential period. The different phases of the action potential are indicated on the graph. [Wikimedia Commons – CC SA 3.0 license]

Although the action potential has been introduced as a local change in transmembrane potential, the strong local depolarization triggers the opening of voltage-gated ion channels in nearby segments of the axon. This induces a chain reaction that results in the propagation of the action potential through the axon to its terminals and under certain conditions from the cell body back to the dendrites (backpropagating action potential). The axon terminals are coupled with other neuron's dendrites through the synapses. If an action potential reaches the axon terminals, neurotransmitters are released that, as explained above, can alter the charge imbalance and transmembrane potential of the neighbouring neurons.

1.2 Tapping into neural activity

Given that neurons are fundamental building blocks of the brain, truly understanding the brain would thus mean understanding the role of each individual neuron within its neuronal network. The activity of a neuron can be quantified in terms of the generation of action potentials. We say a neuron is active when it fires an action potential, because through the generation of action potentials it can change the state of the larger neuronal network in which it is embedded. Measuring the activity of neurons through the measurement of their action potentials is a cornerstone of electrophysiology.

The quantification of action potential occurrences of individual neurons is essential for experimental neuroscientists and their quest towards a fundamental understanding of the brain [78] [58] [5]. Furthermore, the retrieval of action potential occurrences of individual neurons is ideally performed in a real-time and online fashion, because it enables closed-loop neuroscience experiments [22] based on single-unit activity, which is essential for understanding the causality between the activity of individual neurons and cognition. Access to such online single-neuron or single-unit activity is also relevant in the clinic, e.g., for identifying the target brain region for deep brain stimulation in Parkinson's disease patients [52]. Single-unit activity, among other signal modalities, can also be used as an input to a brain-machine interface (BMI) for restoring and enhancing neural functions in patients [89].

Techniques that are capable of measuring the activity of a single neuron, are the intracellular patch-clamp recording [32] and the extracellular single-unit recording using micro-wire electrodes implanted in the brain in close proximity to a neuron of interest [49], which measure the activity of only one single neuron at a time resolution (~0.1 ms) that can represent individual action potentials, also known as spikes, in great detail. To better emphasize action potentials, the extracellular recording is typically high-pass filtered. In the remainder of this chapter, we will assume that such a high-pass filtering has been applied to the extracellular recording.

The reason that action potentials can also be measured in the extracellular medium, is due to the fact that an action potential is generated by an exchange of ionic currents between the intracellular and extracellular medium through voltage-gated ion channels (see Section 1.1). Given that the extracellular potentials originate from a distance-weighted sum of contributions from such transmembrane currents [11], an abrupt change in the electric field local to the ion channels that are involved in the generation of an action potential is expected. Figure 1.3 shows the extracellular potential in response to an action potential at different points in the extracellular medium that are surrounding

the spiking neuron. One can see from Figure 1.3 that the effect of the action potential on the extracellular electric field quickly decays with distance. For the case of multiple active neurons, their effects on the extracellular electric field add up linearly. For a detailed exposition on the biophysics underlying the extracellular action potential, we refer to Linden et al. [68].

Note that the simulated extracellular potentials from Figure 1.3 are noiseless. In real recordings, an extracellular spike is embedded in noise and is only detectable within a small region around the neuron. Recording noise typically consists of neural activity from far-away neurons and noise that originates from the measurement electronics.

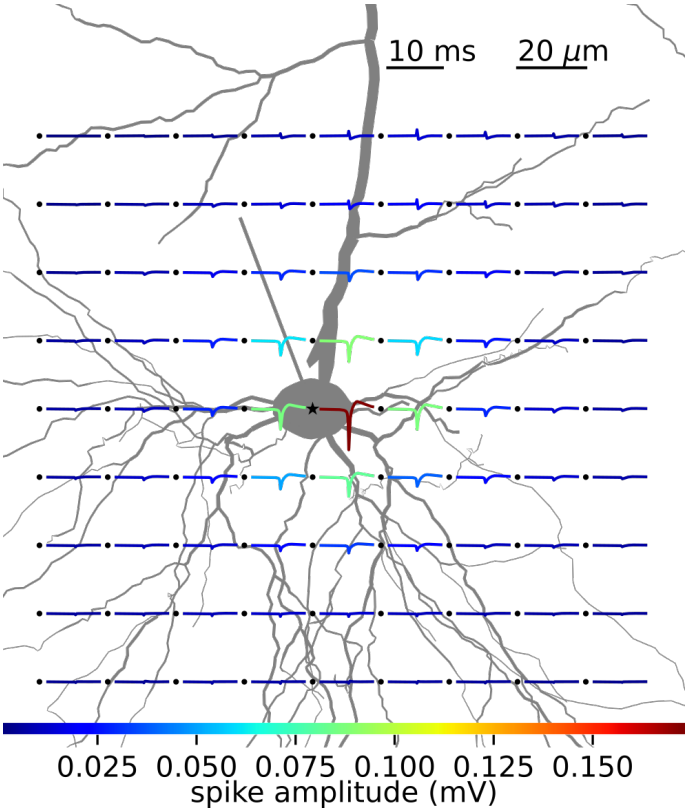

Figure 1.3: The extracellular potential changes at various locations that are indicated by the black dots (and referenced at infinity) are shown in response to an action potential of the visualized neuron model.

Although the techniques discussed in the previous paragraphs enable the

recording of single-unit activity, the techniques result in a minimal yield in terms of the number of neurons they can record from. To improve the recording yield, the size of the extracellular electrode can be increased in order to sample a larger area of the extracellular electric field. Such a larger electrode can pick up spikes from multiple neurons leading to so-called multi-unit recordings. It is to be noted that the electrode size can not be made arbitrarily big. Indeed, the spike waveform is only a local perturbation of the electric field, such that its recorded amplitude will decrease with an increasing electrode size due to spatial averaging. The choice of electrode size and other electrode related characteristics have a big impact on the quality of the extracellular recording [125], but are considered outside the scope of this work. We assume that the recordings used in this work have been obtained with electrodes that are optimally designed for measuring multi-unit activity.

To increase the spatial extent of the multi-unit recording and the number of neurons that are recorded from, multiple multi-unit recording electrodes can be organized in an array-like structure and implanted as such. The geometry and distance between the different recording electrodes are important parameters that affect the spatial extent and neuronal yield [53]. Recent progress in high-density recording array design and fabrication enables recording from thousands of neurons simultaneously through the use of high-density arrays [54] [20] featuring hundreds of recording channels that are concurrently sampled. To even further increase the spatial extent and unit yield of multi-unit activity recordings, recently, recordings have been obtained from multiple high-density electrode arrays simultaneously [115]. An example multi-channel multi-unit recording is shown in Figure 1.4 where neuronal spikes can be visually distinguished from the background noise.

Given now are electrodes that can record the spiking activity of groups of neurons that are in close proximity to those electrodes. Recordings obtained from such an electrode thus contain a mixture of spiking activity from these close by neurons. When single-unit activity is desired, a source separation problem has to be solved. This post-processing strategy of extracting single-unit spike trains, i.e., spike trains from individual neurons, from multi-unit recordings is referred to as spike sorting [67]. From Figure 1.4, one can already notice that there are repetitive patterns present in the recording data which are important for spike sorting.

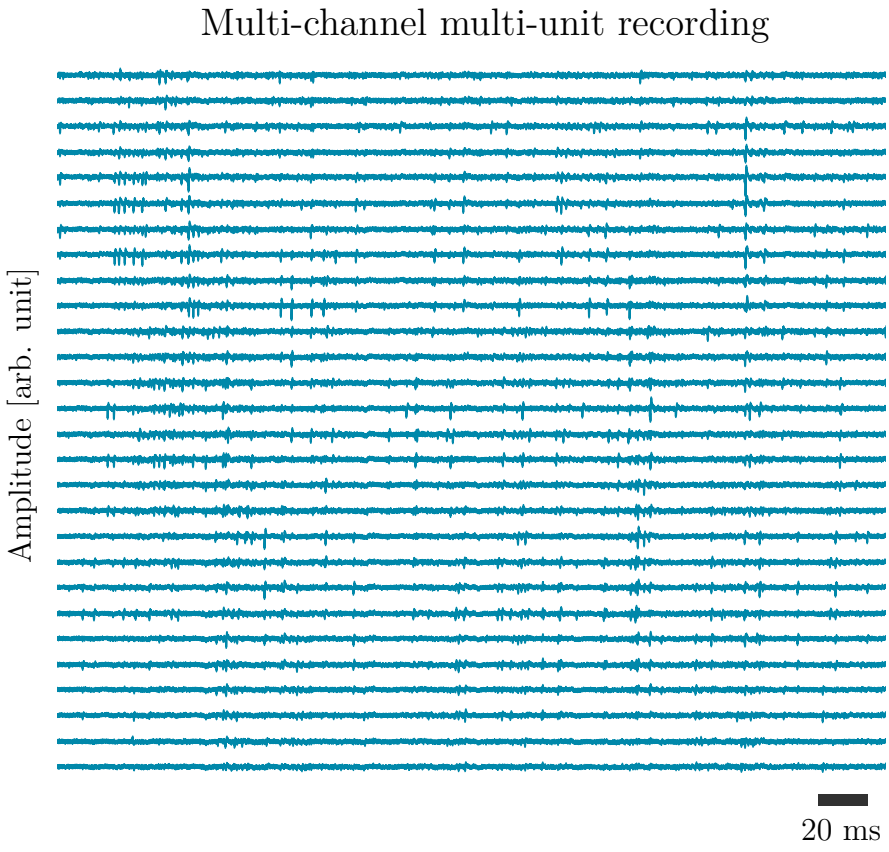

Figure 1.4: Example voltage traces from a multi-channel extracellular recording made with a high-density neural probe with an electrode pitch of 20 µm. The voltage traces have undergone a high-pass filtering with a cut-off frequency of 300 Hz. The horizontal axis represents time, whereas along the vertical axis the voltage traces from several recording channels are shown. The brief troughs in the voltage traces are the extracellular reflections of neuronal action potentials, also known as spikes.

1.3 Spike sorting

As mentioned in the previous section, multi-channel multi-unit extracellular recordings have the potential to sample the spiking activity from thousands of neurons simultaneously. However, the spiking activity from the individual neurons that contribute to the recording are not readily available from the multi-

unit recording. To solve the neuronal source separation problem inherent to the multi-unit recording, i.e., extracting single-unit spike trains from such recordings, the recordings have to be subjected to a processing algorithm known as spike sorting. A reliable recovery of the spike trains of individual neurons, enables a wide range of analysis [38] that can advance our understanding of the brain. Because of its fundamental role, spike sorting has received considerable attention over the last decades [85] [111] [59] [94] [55] [103] [86] [66] [53] [21] [64] [137]. Furthermore, spike sorting algorithms for online use have also been widely investigated [2] [108] [105] [83] [28] [82] [129] [65]

Those solutions to the spike sorting problem are based on two assumptions that enable the separation of neuronal spiking activity in extracellular recordings:

1. The spike waveform of a specific neuron is uniquely distinguishable from the spike waveforms of other neurons. (**uniqueness assumption**)
2. Multiple spike occurrences of the same neuron are constant throughout time. (**stationarity assumption**)

However, for a given recording, the number of neurons that have been recorded from and their respective spike waveforms are unknown. Therefore, the spike sorting problem has to be approached as an unsupervised machine learning.

Multiple factors make that the uniqueness assumption is well approximated in practice. The spike waveform is essentially determined by the morphology of the neuron and the distribution of its ion channels that are involved in the generation of action potentials [7] as we discussed in the previous section. Furthermore, the recorded spatio-temporal spike waveform is modulated by the relative position and orientation of the neuron with respect to the recording electrodes as shown in Figure 1.3. If spike sorting fails, usually it is because the stationarity assumption is violated. We will elaborate on conditions that violate the stationarity assumption in Section 1.4. For the remainder of this section we will assume that the spike sorting assumptions are both satisfied.

Given its unsupervised nature, spike sorting is generally approached as a clustering problem. A generic approach to such a clustering problem, which forms the basis from which specific spike sorting algorithms are derived, is shown in Figure 1.5. In order to retrieve the spike times of the individual neurons that were recorded from, the multi-unit extracellular recording is presented at the input of the pipeline where it is first subjected to a spike detection algorithm. The spike detection results in temporal spike snippets. Those snippets are then projected into a low dimensional feature space as shown in Figure 1.5 that allows for the identification of the different neurons that are recorded from. Finally, the cluster analysis then reveals the single-unit spike clusters from which the

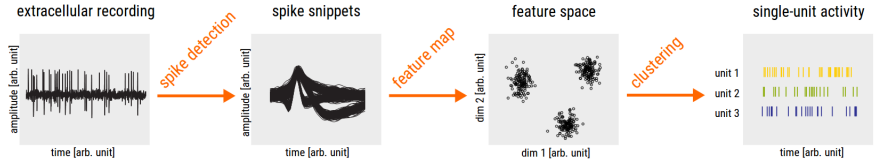

Figure 1.5: A generic unsupervised approach for solving the spike sorting problem consists of three major processing steps: 1. spike snippets are extracted from the extracellular recording through spike detection, 2. every spike snippet is projected into a single-unit promoting feature space, 3. a clustering analysis is performed in the feature space to identify single-unit spike trains. [adapted from [33]]

single-unit spike trains can be trivially constructed. The uniqueness assumption means that there exists a feature map that allows single-unit clustering in the feature space. The stationarity assumption is what leads to the clustering of spike waveforms from the same neuron. However, note that the signal-to-noise ratio (SNR) of the recording also impacts cluster separability in the feature space.

The specific implementation of spike detection algorithm, feature map and clustering algorithm, and, how these different blocks interact, is what differentiates different spike sorting algorithms from each other. The development of different spike sorting implementations is usually driven by the availability of novel algorithms or progress in recording hardware (e.g., increase in the number of concurrent recording channels and their density). For an overview and comparison between different implementations, we refer to the following review papers: Gibson et al. [33], Rey et al. [98], and, Lefebvre et al. [66].

For the sake of a simpler explanation, the algorithmic pipeline from Figure 1.5 is shown to act on a single extracellular recording electrode. However, the use of densely spaced arrays of recording electrodes is what enables recording single-unit activity from many neurons over a larger area. Furthermore, additional spatial information, i.e., information related to the correlation between signals from different electrodes, that can be exploited during spike sorting becomes available when using such electrode arrays. The generic pipeline is applicable to multi-channel recordings through the use of spatio-temporal snippets and a "divide-and-conquer" scheme for clustering [74] [117]. Typically, the spike detection is still performed on a per-channel basis in multi-channel recordings. For every detected spike, a spatio-temporal rather than a temporal snippet is extracted from the recording. A spatio-temporal snippet is a collection of

concurrent temporal snippets as recorded on multiple electrodes. The spatial window for a specific spike, i.e., the subset of recording electrodes for that spike, is usually centred around the channel on which the spike reaches its maximum energy and includes neighbouring channels on which the spike is also present. Irrelevant channels from that specific spike's point of view are not included in the spatial window. Given that every snippet is centred on a certain channel, the clustering is typically performed on a per-electrode basis. Only snippets centred on a certain electrode are projected into a feature space, where they are then clustered into single-unit clusters and this procedure is repeated for each channel separately.

1.4 Violation of the stationarity assumption

Many implementations of the generic spike sorting pipeline that was presented in the previous section are only robust in case the uniqueness and stationarity assumption are satisfied. However, the stationarity assumption is easily violated in practice. Three common causes have been identified that contribute to this violation [98]:

1. A **bursting neuron**, i.e., a neuron that fires multiple spikes consecutively, will exhibit a varying spike waveform shape and amplitude throughout such a burst.

2. Relative movement between the brain tissue and recording electrode, also referred to as **electrode drift**, will result in non-stationary spike waveforms.

3. The occurrence of two or more spikes from nearby neurons firing near-simultaneously in time will lead to **overlapping spikes** snippets. Such overlapping spikes snippets will typically map to points in the feature space that are unrelated to the clusters that represent the well isolated spikes of the neurons that contribute to the overlapping spikes snippet.

It is to be noted that under one or more of the above causes, the clustering pipeline does not necessarily break down completely. For example, although overlapping spikes snippets cannot be resolved to their single-unit contributions, non-overlapping (in the spatio-temporal sense) spike snippets present in the same recording can still be sorted. Not accounting for bursting neurons and electrode drift will typically lead to an overclustering, i.e., more clusters are retrieved than the number of neurons from which we record spikes. In practice, this overclustering is usually corrected during a manual cluster merging process.

This manual curation step is time consuming and does not scale well with the ever increasing number of neurons that can be recorded from simultaneously. Because of their negative impact on spike sorting performance and applicability, the causes of violation have been studied intensively and (partial) solutions have been adopted in practice. For more information on the automatic sorting of bursting neurons and/or electrode drift compensation we refer to Appendix 1.8.1 and 1.8.2, respectively. In the remainder of this Chapter we will mainly focus on techniques for resolving overlapping spikes. We will also discuss the opportunities and limitations of these techniques, which will form the starting point of the research presented in this thesis.

1.5 Resolving spike overlap

1.5.1 Threshold-based spike sorting

The available solutions for solving the *overlapping spikes* problem are based on the use of single-unit spike templates. The popularity of approaches based on spike templates can be understood from the fact that the near-simultaneous firing of several neurons results in a linear superposition of its spike waveforms. Therefore, spike overlap can be modelled as the sum of scaled time-shifted spike templates, such that template matching techniques can be used to retrieve the activation times of individual neurons in the context of spike overlap. We can differentiate between sparse signal reconstruction template matching approaches [93] [90] [27] and matched filtering approaches [74] [31].

A schematic representation of the matched filtering approach is shown in Figure 1.6. The generic clustering-based pipeline that was introduced in Section 1.3 is augmented here with a template estimation and matched filtering block. Given such an augmented pipeline, spike sorting is performed in three distinct phases:

1. The clustering-based pipeline is applied to the recording and results in a collection of single-unit clusters, but fails to resolve overlapping spikes snippets.

2. For every single-unit cluster, a spike template is estimated by combining the spike time information obtained in phase 1 and the extracellular recording. Subsequently, a matched filter is derived for every single-unit template.

3. The matched filters are then applied to the extracellular recording. Based on the matched filter outputs, the single-unit spike trains are retrieved.

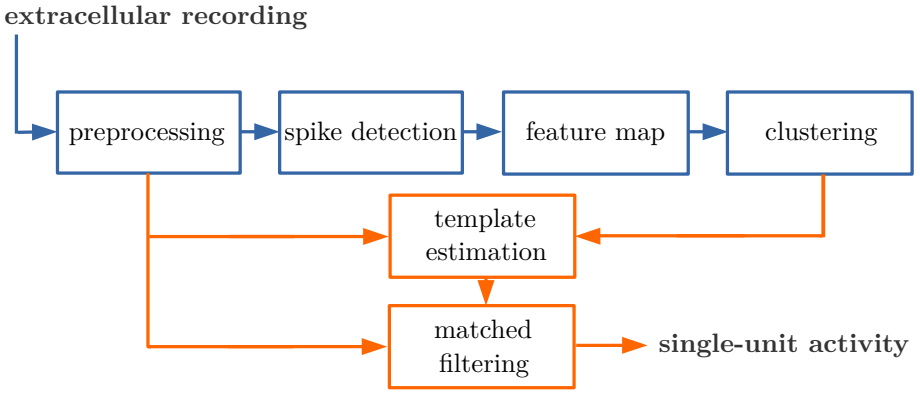

Figure 1.6: Schematic representation of a spike sorting pipeline as introduced in Section 1.3 (blue) that is augmented with template matching blocks (orange) for resolving spike overlap.

Qualitatively speaking, the matched filter emphasizes at its output the occurrence of its corresponding template in the input extracellular recording. When the matched filter succeeds at only emphasizing the corresponding template, each matched filter output can be analysed, independently from the other filter outputs, to decide whether or not the corresponding template is present at the filter input. A schematic representation of this approach is shown in Figure 1.7. When a simple threshold operation is used as a decision device, we refer to this pipeline as a *threshold-based spike sorting* pipeline.

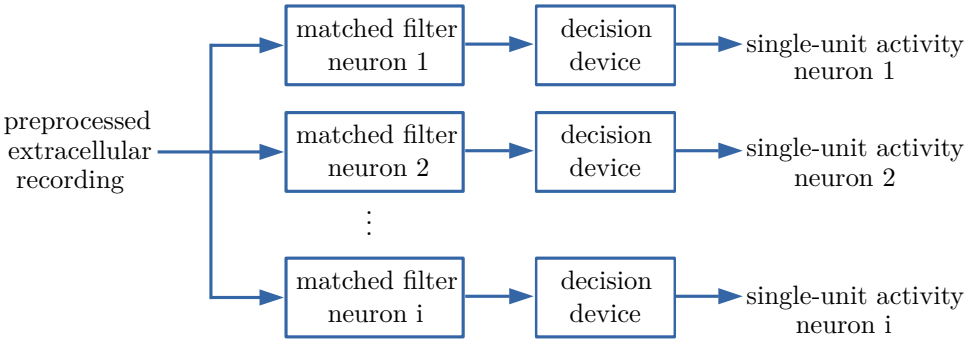

Figure 1.7: Detailed schematic representation of a matched filtering approach for resolving spike overlap where each filter output is processed independently from one another.

1.5.2 Opportunities for real-time sorting

The matched filters can also be applied to data that was unseen during their design. For example, the pipeline presented in Figure 1.7 can be applied in the context of online spike sorting [30]. The first two phases as presented in Section 1.5.1 are initially performed on so-called training data. During online sorting, only the third phase is executed. For online spike sorting, the computational complexity and algorithmic delay of the spike sorting approach matter. Typically, threshold-based spike sorting filters require only linear operations and have a low and deterministic delay. Another advantage of using the threshold-based approach for online spike sorting is that it explicitly accounts for overlapping spikes. Due to its interesting properties, recently, several threshold-based spike sorting hardware implementations have been proposed [39] [88] [71] [123] [128]. Such neural processors can extract single-unit spike time information without the need for an external computer during sorting.

In experimental neuroscience, such a spike sorting neural processor can be used in closed-loop stimulation experiments, where a predictable delay is essential for timely stimulation. Furthermore, a spike sorting that is performed prior to the transmission of the data results in an effective data compression. When only spike times and their respective neuron identities are of interest on the receiver side, the required energy budget for data transmission can be drastically reduced if the spike sorting results are available on the transmitter side. The energy consumption of a neural interface is essential in the case of wireless neural implants. Neural processors can also be used as part of unobtrusive BMIs that depend on the availability of single-unit activity information. Although, threshold-based spike sorting is promising for use in online spike sorting and hardware implementations thereof, current implementations suffer from a fundamental shortcoming that has been overlooked.

1.5.3 Non-discriminative matched filter problem

The matched filters that are in use in spike sorting [74] [31] are typically not sufficiently discriminative, i.e., they also emphasize certain other pattern occurrences, besides the occurrence of their corresponding template. Such interfering patterns are typically spike waveforms from close-by neurons. When interfering patterns are present, the approach presented in Figure 1.7 will lead to false detections.

Therefore, a more complex approach is often encountered in offline spike sorting that is shown in Figure 1.8. Here, all filter outputs are processed together. When, certain filter outputs indicate the presence of a pattern, that pattern is

assigned to the spike template, and hence neuron, to which it is most similar. In order to resolve overlapping spikes using this scheme, the assigned spike template is subtracted from the input data. After this subtraction, the filter-and-subtract process is repeated until no more pattern occurrences are detected. Note that this additional iterative subtraction results in a non-deterministic delay and additional complexity. Furthermore, such an iterative process is a greedy rather than an optimal approach, which comes with its own failure modes leading to false detections.

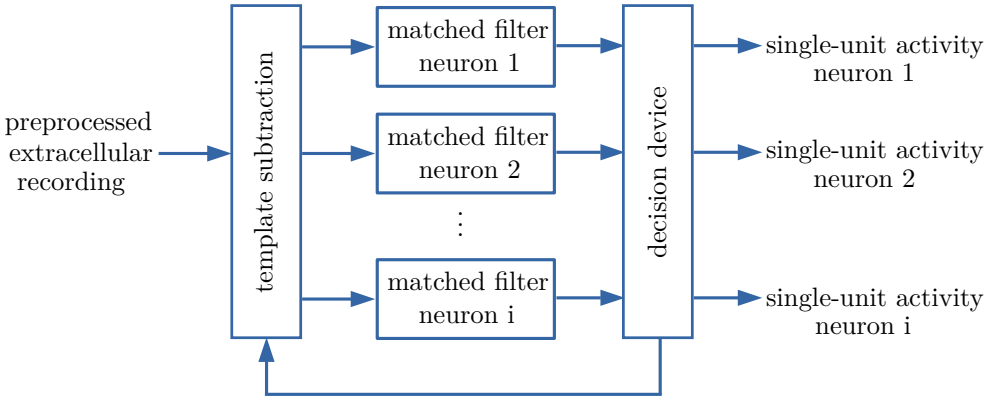

Figure 1.8: Detailed schematic representation of an iterative matched filtering approach for resolving spike overlap when the individual matched filters are not sufficiently discriminative.

The threshold-based spike sorting pipeline from Figure 1.7 is favourable, both in terms of computational complexity and algorithmic delay, when compared to the iterative pipeline presented in Figure 1.8. Because of its importance to online spike sorting, matched filter design methods have been proposed that promise to improve the discriminative character of the matched filter for use in spike sorting [100] [126] [30]. Unfortunately, the improved filter design is based on unrealistic prior knowledge assumptions, such that they are not used in practice.

In Chapter 2, 3 and 5 of this thesis, we develop novel discriminative matched filter design methods for threshold-based spike sorting with realistic prior knowledge assumptions.

1.6 General challenges in spike sorting

Besides the remaining challenges related to threshold-based spike sorting, we would also like to draw the attention to two general open challenges in spike sorting. Although the scope of these challenges is much broader than online spike sorting, we will also provide (partial) solutions to these general challenges in this thesis.

1.6.1 Lack of ground truth data

Spike sorting development has suffered from a lack of ground truth data for validation purposes. Moreover, the lack of a golden standard in spike sorting makes it difficult to assess the true capabilities of a certain spike sorter. However, recently spike sorting validation has received more attention [12] [13] and has resulted in an ever increasing spike sorting validation set [72]. Despite the recent progress, it is unlikely that a single spike sorter outperforms all the others under all conditions. Therefore, the choice for a certain spike sorter is likely to be still driven by personal taste rather than spike sorting performance given a specific experimental recording set up.

In Chapter 4 we present a graphical tool for the generation of ground truth data that is tailored towards a desired recording set up.

1.6.2 Coping with multiple stationarity violations

Many of the techniques to cope with the causes of stationarity violations from Section 1.4 have been implemented in spike sorting packages that are available to the public (see Section 1.5.1 and Appendix 1.8.1 and 1.8.2). However, to the best of our knowledge no algorithm is available that can cope with all three violating mechanisms simultaneously. Intuitively, we can get some understanding of why an integrated solution for all violating mechanisms is non-trivial. Drift robustness very much depends on location independent signal characteristics [43], whereas bursting robustness typically requires such location or spatial information [53]. Furthermore, template matching for resolving spike overlap is based on the usage of prototypical waveforms, which are not straightforwardly compatible with changing signal statistics. Although adaptive template matching approaches exist, they are very unlikely to be able to track fast changing signal statistics.

In Chapter 6 we propose a novel approach for resolving spike overlap in the feature space. Because this approach eliminates the need for matched filters,

and could potentially be extended with bursting and electrode drift robustness, it is a promising direction for an integrated sorting approach.

1.7 Thesis outline

A schematic overview of the organization of this manuscript is provided in Figure 1.9.

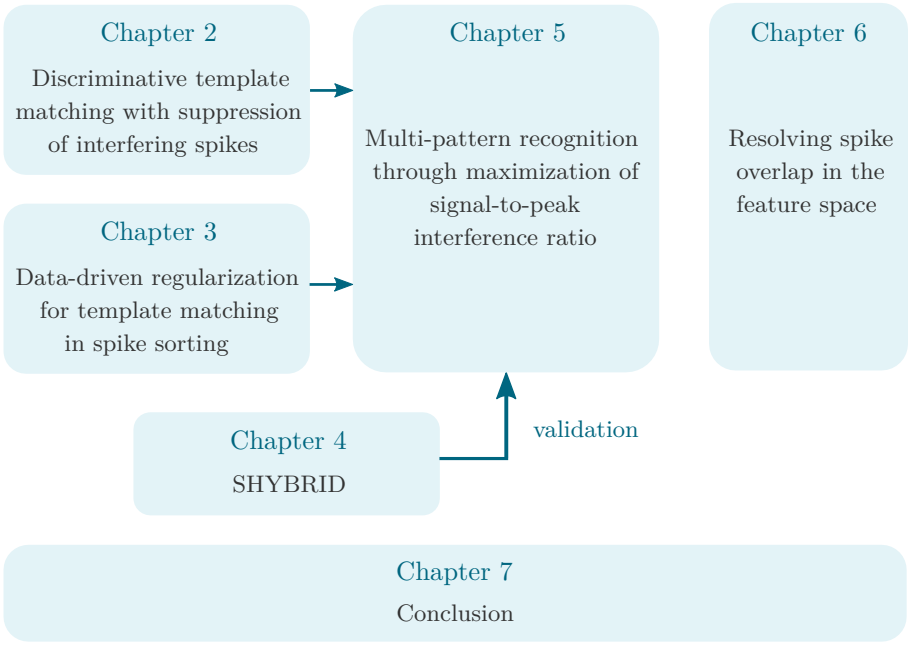

Figure 1.9: Thesis organization overview.

In Chapter 2 we propose a novel discriminative filter design method to leverage the low-complexity threshold-based spike sorting architecture. The proposed filter design method is based on the concept of signal-to-peak-interference ratio (SPIR) optimality, which is a natural expression of the desired filter output behaviour for use in threshold-based sorting. Furthermore, a SPIR-optimal filter design approach requires a minimum of prior information, unlike other existing discriminative filter design methods which suffer from unrealistic prior knowledge assumptions. At first, SPIR-optimality is approached from a heuristic point of view. The heuristic approach is validated on an initial publicly available ground truth data set. In chapter 5 we propose true SPIR-optimal filter design

methods that fix the practical limitations related to the heuristic approach. The true SPIR-optimal filter design methodologies are validated on an extensive hybrid data set.

Chapter 3 focusses on the regularization of linear filter design for use in spike sorting. Here, regularization entails the addition of extra information to the filter design process in order to obtain a more reliable result. The regularization of linear filter design for use in spike sorting has not been explicitly studied, yet it is a crucial concept that can heavily affect the spike sorting performance when not properly understood. We provide some insights into the need for a regularization, and we also propose a regularization approach that is in line with the underlying problem causing the need.

In Chapter 4 we take a unique approach where we focus on the generation of ground truth data that is in line with the recording setting (i.e., the combination of all factors that ultimately determine the extracellular recording characteristics) of individual users. The approach is based on a hybrid ground truth scheme. In a hybrid scheme, simulated spike waveforms are embedded in real recordings. In this work we rely on a data-driven model for the generation of spike waveforms, rather than using simulations from complex biophysically detailed neuron models. We also present a graphical tool for the generation of such hybrid ground truth data. Essentially, this tool facilitates the transformation of real extracellular recordings into ground truth recordings. Such hybrid data is shown to be useful for experimentalists in a case study that we present in Chapter 4. Its usefulness for developers is implicitly shown in the case study that is presented in Chapter 5.

In Chapter 6 we propose a novel approach for spike sorting where spike overlap is resolved in the feature space. Although the work in this chapter is preliminary, the proposed methodology is promising in the context of delivering a spike sorting solution that can cope with all three violating mechanisms simultaneously, as we will discuss in Chapter 7. This approach could also be extended into a low-complexity method through network pruning strategies. Furthermore, in Chapter 7 we conclude our work and discuss the remaining challenges related to the presented material. We also provide some pointers that could inspire future research and developments.

1.8 Appendix

1.8.1 Spike sorting with bursting neurons

Several types of solutions for the *bursting* problem have been proposed. A first type of solution consists of using features that are robust to bursting, mainly by emphasizing spatial location, rather than the temporal waveform. Feature maps have been proposed that relate to or approximate the location of the putative neurons [99] [43] [51] or that make use of features derived from relative rather than absolute multi-channel amplitude information [53]. A second type of solution consists of defining generative models for the extracellular recording, in which bursting statistics (inter-spike time dependence on amplitude scaling) are incorporated [91] [25]. Spike times are then recovered through a maximum a posteriori (MAP) estimation. The downside of such an approach is that the inter-spike interval statistics can not be reliably used as spike train quality metrics after spike sorting. A third approach consists of an automatic merging of clusters [21], which is equivalent to a manual curation with fixed rules and decision thresholds.

1.8.2 Spike sorting with electrode drift

For coping with *electrode drift*, a first strategy consists of quantifying the drift prior to spike sorting and correcting for it [53]. Such an approach is limited by the rather poor spatial resolution of the extracellular recording in terms of sampling the complete underlying electric field, allowing for only a limited drift correction. A second strategy that is often employed is that of adaptive clustering, either driven by parametric models [8] [14] [110] or non-parametric approaches [105] [61]. A recent notable approach involves reordering time segments based on similarity between the segments, prior to adaptive clustering [87]. By doing so, the algorithm can track both slow and fast changes in signal statistics due to drift through adaptive clustering. The downside of relying on segment reordering is that it is not suitable for online spike sorting.

Chapter 2

Discriminative template matching with suppression of interfering spikes

This chapter has been published as *Jasper Wouters, Fabian Kloosterman, and Alexander Bertrand. "Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes." Journal of neural engineering 15.5 (2018): 056005.* Jasper Wouters has co-developed the methodology, conducted the experiments and has written the manuscript. Compared to the publication, minor textual and notational changes have been implemented for better integration in this thesis.

Currently, the use of spike sorting is mainly limited to an offline usage, where spikes are sorted after the data acquisition has been completed. In this chapter, we propose a discriminative template matching algorithm for threshold-based spike sorting on high-density extracellular data. Such threshold-based spike sorting has a low and deterministic algorithmic delay, allowing for fast online spike sorting. At its core, threshold-based spike sorting is driven by linear filters. The proposed discriminative template matching filter design algorithm optimizes the output signal-to-peak-interference ratio in a data-driven fashion, assuming the template of the target spike is available. The latter allows the filter to suppress the spikes of interfering neurons and to resolve spike overlap. The data-driven filter design algorithm requires only templates of the target

neurons of interest, which can be retrieved, e.g., through a prior clustering on an initial recording. The proposed discriminative template matching filters are validated on in-vivo ground truth data and are shown to provide single-unit activity with good accuracy using a simple thresholding operation on the filter outputs. The low algorithmic complexity allows for computationally cheap and fast spike sorting. Also the proposed filters are guaranteed to be stable and have a deterministic delay. These characteristics make the proposed filter design method a valuable building block for online spike sorting, thereby enabling unit activity-based real-time and closed-loop experiments for high-density neural recordings.

2.1 Introduction

Neurons communicate with each other through all-or-nothing action potentials, also known as spikes, which can be recorded with an extracellular electrode placed in the vicinity of the neurons. An electrode will generally pick up spikes from several neurons near the electrode. Therefore, to decode brain processes, neuroscientists have to identify and sort these spikes according to their underlying neuronal origins, a source separation process which is often referred to as spike sorting [67].

Currently, spike sorting is often performed as a post-processing step once a complete extracellular recording is available. This allows only correlative conclusions to be drawn from the spike sorting results and behavioural data. When performing online spike sorting [2] [61] as part of a closed-loop experiment in which neural processes can be altered through stimulation, causal relationships could be discovered between spiking patterns and behavioural outcomes. The wide availability of such an algorithm would be a game changer for experimental neuroscientists.

Recently, on the hardware side, there has been a shift from conventional extracellular recording devices with a few electrodes to high-density (HD) neuroprobes with more than thousand electrodes and electrode pitches of around 20 µm [69] [95] [54]. Such HD probes enable recording from the entire depth of a rodent brain. For these new probes, conventional spike sorting algorithms [111] [94] [55] [103] fail, because the computational complexity scales supralinear with the number of electrodes. On top of that, the cluster analysis will suffer from the "*curse of dimensionality*" [113], a general phenomenon that occurs when clustering in high-dimensional spaces.

To overcome these computational problems, a number of new spike sorting algorithms have been developed [86] [53] [21] [137] for sorting HD probe data.

All of these algorithms are developed to perform real-time spike sorting on HD extracellular data. Important to note however, is that they are real-time with respect to the offline processing time, meaning that they can process a finished recording in approximately the same time as the duration of the recording. For closed-loop stimulation experiments in which the stimulation depends on the outcome of the spike sorting process, an online real-time deadline has to be met which is dependent on the specific process under investigation [22]. For instance an experimental study of synaptic plasticity requires stimulation to occur within 2 ms, the synaptic integration time scale, after a target neuron generated an action potential [53].

Many conventional spike sorting algorithms also struggle with handling overlapping spikes. The reason for their failure is that spikes are classified based on features used during the clustering analysis [33]. When features are extracted from a segment containing, e.g., two overlapping spikes from different neurons, the features are unlikely to be related to the features extracted from single spike segments from either neuron, leading to spurious clusters. To solve this problem, recent algorithm pipelines have been augmented with a final template matching [1] [74] [90] [27] [31] phase. Prototypical spike templates are in practice extracted from an initial spike sorting, where only non-overlapping spikes are taken as the input. Each extracted spike template is then applied to the data as a linear matched filter. The outputs of these matched filters are then used to decide on the final cluster assignment for spikes in the recording.

Such linear filtering-based template matching is a particularly interesting approach for online spike sorting. For each sortable unit in the extracellular recording, a linear filter is trained which responds to the spatio-temporal spike footprint of that unit [29]. Such filters can be applied to the data in a streaming fashion. This implies that a spike can be sorted before its contribution in the extracellular recording has faded away. Classification is then done by thresholding the different filter outputs. Another advantage of using linear filters is their low computational complexity. The only computations involved in applying such filters are scalar multiplications of recording samples in a finite shift register and a summation over the different products. This makes linear filters extremely fit for fast (hardware) implementation and online real-time application.

Recent work on template matching-based spike sorting [31] [137] uses only information from target spikes in training the template matching filters. This leads to the problem of multiple filters responding to a single spike, when their target spike pattern shows some similarity with the spike pattern of the active neuron. This makes it difficult to determine whether these simultaneous responses are caused by a single spike triggering multiple filter responses, or by multiple overlapping spikes each triggering a single filter response. To solve this

problem, both methods apply an iterative scheme where they assign the spike to the neuron of which the corresponding filter response is maximal and then subtract the spatio-temporal template of this neuron from the data. The data from which the template is subtracted is iteratively filtered again (each time followed by another spike template subtraction) until all filter responses on this part of the data are below the detection threshold. A consequence of such an iterative scheme is that it takes multiple iterations over the same data. This causes an unpredictable delay in the processing, which might lead to missing real-time deadlines. Even if there is just one spike active, a second pass over the data is necessary to confirm that the exceeded thresholds in other filter outputs are not caused by overlapping spikes.

Another potential issue arises directly from relying on the identification of the maximal filter response for assigning a spike to its putative neuron. Since the filters are subjected to noise, the filter outputs might be distorted and as such the filter with the maximal response is not always the one that corresponds to the firing neuron, leading to a misclassification of that spike.

To overcome the aforementioned issues related to iterative subtraction schemes, linear filters can be designed which simultaneously maximize the response to the target neuron, while suppressing interfering spikes from neighboring neuron cells, as well as general background noise [100] [126] [30]. The method proposed in this work is in line with the design goal of the cited methods. However, the previous algorithms were presented prior to the availability of HD probe data, and were shown to be unable to sufficiently differentiate between the neurons using non-HD data and a simple threshold. [29].

So we will investigate the use of discriminative template matching filters for HD extracellular recordings, which are strongly tuned towards a single unit, such that a significant filter response is generated only when the corresponding neuron is active. Such an approach removes the necessity of iterative filtering schemes, it enables threshold-based spike sorting, and it allows for a controllable processing delay.

The optimal filter design will be based on the generalized eigenvalue decomposition (GEVD) [35] of pairs of estimated signal and interference covariance matrices. Furthermore a practical offline scheme for estimating these covariance matrices will also be proposed, such that the optimality of the filters is aligned with the ultimate goal: maximizing spike sorting accuracy. Basically this comes down to an implicit optimal weighting of the different interfering sources in the interference covariance matrix to maximize the *signal-to-peak-interference ratio* (SPIR). We also show that discriminative template matching filters are robust against target leakage, i.e., the target spike structure is present in the interference covariance matrix.

Our work differs from other discriminative template matching design algorithms as follows: **1.** From an optimization point of view, our filter design method is based on maximizing the signal-to-peak-interference ratio (SPIR) objective function as opposed to signal-to-interference-plus-noise ratio (SINR) or signal-to-noise ratio (SNR) optimal filters. **2.** The interference covariance is estimated in a data-driven way as opposed to template-based approaches, reducing the dependence on the initial clustering stage and the availability of templates of non-target spikes or other transient noise sources. **3.** The algorithm presented in this work is validated on in-vivo recorded HD probe data, which are rich in spatial information, such that multi-channel filtering approaches become more powerful.

The remainder of this chapter is built up as follows. In Section 2.2 a model for online spike sorting, namely *threshold-based spike sorting*, used throughout this chapter is presented and matched filtering-based template matching is reviewed. In section 2.3 our *discriminative template matching* filter design method is derived. Next the proposed method is validated on HD in-vivo recorded ground truth data in Section 2.4. Finally the results are interpreted and conclusions are drawn in Section 2.5.

2.2 Threshold-based spike sorting pipeline

This section will first conceptually introduce the threshold-based spike sorting pipeline, followed by the derivation of a matched filter that can be used as the pipeline's core template matching filter. **Mathematical notation:** Scalars are represented by non-bold letters a or A, vectors by bold lowercase letters $\mathbf{b}$ and matrices by bold uppercase letters $\mathbf{C}$. $\mathbf{C}^T$ denotes the matrix transpose of $\mathbf{C}$ and $\mathbf{C}^{-1}$ denotes the matrix inverse of $\mathbf{C}$.

2.2.1 Threshold-based spike sorting

Consider a HD extracellular neural probe with N electrodes, or channels, as shown in Figure 2.1. Such a probe is lowered into the brain so that it can record spikes from neurons in the vicinity of the electrodes. Spikes from a specific neuron will typically be picked up by multiple electrodes close to that neuron.

Consider $\mathbf{x}[k]$ to be an N-dimensional vector containing the observations from the N-channel neural probe at sample time k. To extract the activity of a neuron of interest i, usually only a subset of the channels in $\mathbf{x}[k]$ is useful. Therefore, we introduce for each neuron i the N_i-channel signal $\mathbf{x}_i[k]$ consisting

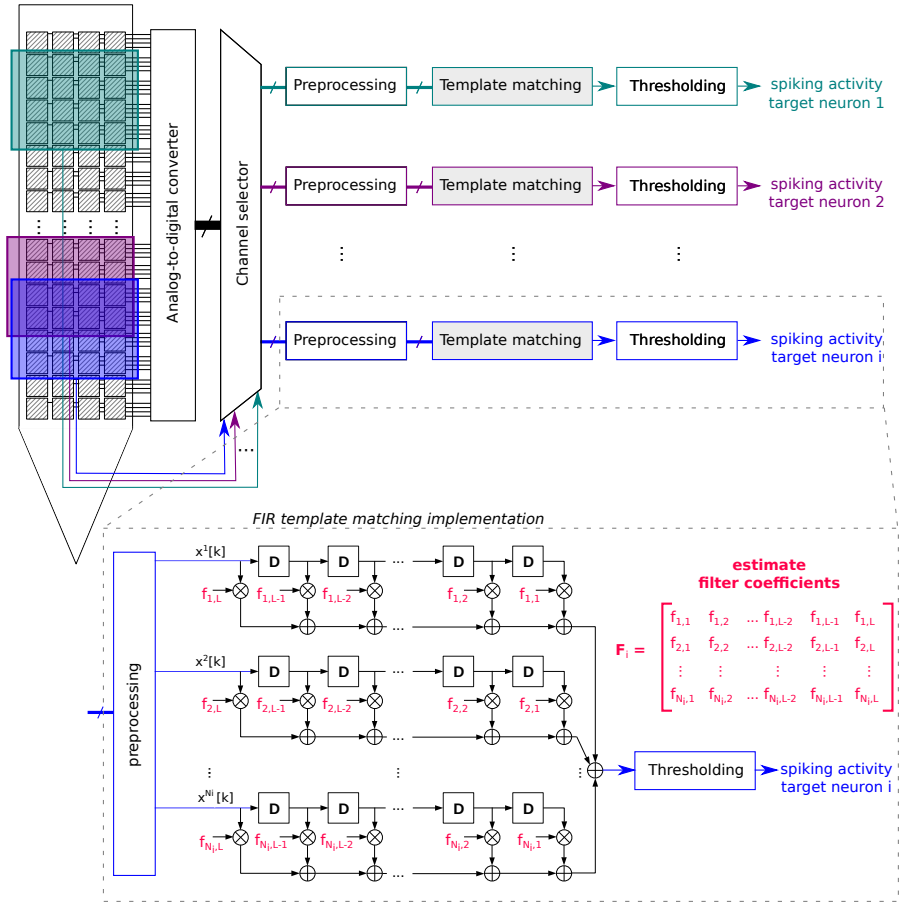

Figure 2.1: A schematic representation of the *threshold-based spike sorting* paradigm. For every sortable neuron, a separate processing pipeline needs to be implemented. Every pipeline includes preprocessing, template matching and thresholding, and generates single-unit activity at its output. A detailed view of such a pipeline (bottom) reveals the finite impulse response (FIR) template matching filters that are considered in this chapter.

of N_i channels of $\mathbf{x}[k]$ that contain useful information for isolating the activity of neuron i. Selecting a subset of channels is represented in Figure 2.1 by the *channel selector* block. Some practical guidelines on how this subset of channels can be selected will be given in Section 2.4.

Consider $x_i^c[k]$ to be the observation of the cth channel of $\mathbf{x}_i[k]$. Without loss of generality, we assume $x_i^c[k]$ to be zero-mean for each c, which is generally achieved through a prior high-pass filtering on the data. This high-pass filtering is schematically represented by the *preprocessing* blocks in Figure 2.1. Define $\overline{\mathbf{x}}_i^c[k] = \begin{bmatrix} x_i^c[k] & x_i^c[k-1] & \dots & x_i^c[k-L+1] \end{bmatrix}^T$ as the L-dimensional time-lagged vector describing a delay line containing the current and $L-1$ previous samples of $x_i^c[k]$ at time k. By stacking all these delay lines for every channel of $\mathbf{x}_i[k]$, we obtain the vector $\overline{\mathbf{x}}_i[k] = \begin{bmatrix} \overline{\mathbf{x}}_i^1[k]^T & \dots & \overline{\mathbf{x}}_i^{N_i}[k]^T \end{bmatrix}^T \in \mathbb{R}^{LN_i}$.

The goal of this chapter is to design a linear spatio-temporal filter $\mathbf{f}_i \in \mathbb{R}^{LN_i}$ for each detectable neuron i in the recording, such that spike sorting can be done via a simple thresholding operation on the individual filter outputs $y_i[k] = \mathbf{f}_i{}^T \overline{\mathbf{x}}_i[k]$, without using iterative schemes for handling overlapping spikes. Note that $\mathbf{f}_i{}^T \overline{\mathbf{x}}_i[k]$ corresponds to a filter-and-sum operation containing N_i finite impulse response (FIR) filters, each with L taps. Spike sorting based on simple thresholding of $y_i[k]$ will be referred to as *threshold-based spike sorting*. Each filter will work independent of one another and all filters can be executed in parallel. The neuron corresponding to a filter is also referred to as the *target neuron* of that filter.

An FIR filter can be implemented with delay elements, adders and multipliers as shown in the schematically detailed bottom part of Figure 2.1. Such filters are guaranteed to be stable, they have a predictable processing delay, and have a low computational complexity allowing for fast online processing. The actual filter design boils down to choosing the involved channels (electrodes) and the number of delay elements, and estimating the filter coefficients such that the filter has the desired behaviour. Section 2.2.2 and 2.3 will focus on designing template matching filters for threshold-based spike sorting. The presented filter design algorithms are meant for offline use, meaning that these algorithms estimate the filter coefficients from initially available recording data, after which the online filter can be applied to streaming data. However, these algorithms are candidates for adaptive implementations, such that the filter coefficients can be corrected for varying signal statistics in an online fashion.

2.2.2 Matched filtering-based template matching

In the remainder of this chapter it is assumed that at least the templates of the neurons of interest (i.e. the neurons for which we want to retrieve the single-unit spike times) are known. In practice this prior knowledge can be extracted directly from the recording using a clustering-based initial spike sorter to perform an initial coarse clustering, possibly on only a subset of data. Such an initial spike sorter will also give access to a set of example spike times (used to estimate the template) for each neuron of interest.

Because all filters operate in parallel and independently from one another, as can be seen from Figure 2.1, the mathematics will be derived for a single neuron of interest i.

As the SNR at the output of the filter has a direct impact on the classification accuracy in a threshold-based spike sorting paradigm, it is important to properly design the filter in order to maximize its overall output SNR. In standard matched filtering-based template matching [56], the filter coefficients $\mathbf{f}_i$ are chosen to be the spatio-temporal template of neuron i, yielding a so-called matched filter design. This template $\boldsymbol{\tau}_i$ is an LN_i-dimensional vector and can be estimated by averaging over multiple aligned example spikes generated by neuron i:

$$\mathbf{f}_i^{\mathrm{MF}} = \boldsymbol{\tau}_i = \frac{1}{|\mathcal{T}_i|} \sum_{k \in \mathcal{T}_i} \overline{\mathbf{x}}_i[k], \tag{2.1}$$

where $\mathcal{T}_i$ is the set that contains all discrete example spike times of neuron i, that have been chosen such that the example spike waveforms are mutually temporally aligned and $|\mathcal{T}_i|$ denotes the cardinality of the set $\mathcal{T}_i$. Alternatively, a median operator can be used to estimate a template, which is less sensitive to outliers compared to averaging. The output $y_i^{\mathrm{MF}}[k]$ of the matched filter $\mathbf{f}_i^{\mathrm{MF}}$ will have a large response when a spike of neuron i is presented at its input at time k.

It is noted that a matched filter only optimizes the output SNR if the background noise is uncorrelated both in space and time. Therefore, a pre-whitening operation [92] [31] is performed on the signals in order to obtain a white background noise without spatio-temporal structure. The zero-phase component analysis (ZCA) [57] whitening matrix $\mathbf{W}_i$ is given by the square root of the inverse of the spatio-temporal covariance matrix of the background noise:

$$\mathbf{W}_i = \mathbf{R}_{\overline{\mathbf{n}}_i \overline{\mathbf{n}}_i}^{-\frac{1}{2}} = E\left\{\overline{\mathbf{n}}_i[k] \cdot \overline{\mathbf{n}}_i[k]^T\right\}^{-\frac{1}{2}}, \tag{2.2}$$

with $\overline{\mathbf{n}}_i[k]$ the additive background noise. Since $\overline{\mathbf{n}}_i[k]$ is unknown, in practice $\mathbf{R}_{\overline{\mathbf{n}}_i\overline{\mathbf{n}}_i}$ is estimated from $\overline{\mathbf{x}}_i[k]$ at times k when no spikes are detected, i.e., where only baseline noise is observed. The square root can be computed using, e.g., a Cholesky factorization or an eigenvalue decomposition on $\mathbf{R}^{-1}_{\overline{\mathbf{n}}_i\overline{\mathbf{n}}_i}$.

Applying the whitening matrix to the data will reduce the spatio-temporal structure of the background noise, but it will also alter the spike waveform, and as such the template. Therefore matched filtering-based template matching using whitened data can be expressed as:

$$\begin{aligned} y_i[k] &= \left(\mathbf{W}_i\boldsymbol{\tau}_i\right)^T \left(\mathbf{W}_i\overline{\mathbf{x}}_i[k]\right) \\ &= \left(\mathbf{R}^{-1}_{\overline{\mathbf{n}}_i\overline{\mathbf{n}}_i}\boldsymbol{\tau}_i\right)^T \overline{\mathbf{x}}_i[k]. \end{aligned} \tag{2.3}$$

From the above equation it can be seen that the whitening can be combined with template matching using a single FIR filter $\mathbf{f}_i^{\mathrm{MF}} = \mathbf{R}^{-1}_{\overline{\mathbf{n}}_i\overline{\mathbf{n}}_i}\boldsymbol{\tau}_i$.

The noise covariance matrix will likely be dominated by the baseline noise from far-away neurons in which case a pre-whitening in combination with a matched filter will obtain its optimal SNR by reducing the baseline noise. This improvement in SNR is unlikely to influence the threshold-based spike sorting accuracy, because false detections mainly arise from sparse transient noise events and spikes from close-by neurons. Therefore, the matched filter as described in this section, and as used in many spike sorting algorithms, is often not optimal for threshold-based spike sorting. The problem with the matched filter used for threshold-based spike sorting is that the spatio-temporal structure of a spike template might not be sufficiently discriminative to generate a high output variance *only* when neuron i is firing.

2.3 Discriminative template matching

2.3.1 Filter design for SPIR maximization

Consider again the goal of this chapter: the design of a linear FIR template matching filter for each detectable neuron in a HD probe recording, such that single-unit activity can be obtained by a simple thresholding operation on the individual template matching filter outputs.

Conceptually the optimal filter for threshold-based spike sorting, would be the filter where spurious input peaks are suppressed towards the same level as the

filter's response to background noise, while peaks resulting from activity of the target neuron are still significantly higher. Such a filter design should not focus on optimizing overall output SNR, but on maximizing *signal-to-peak-interference ratio* (SPIR) instead. A filter optimizing the SPIR can be thought of as a pre-whitening with a matched filter, which is designed using a weighted noise covariance estimate.

To increase the threshold-based spike sorting accuracy we have to extend the definition of noise: *noise should also include spikes from other neurons*, as is done in [100] [126] [30]. Furthermore, some noise sources are more severely affecting spike sorting accuracy than others, and as such should be treated differently. The optimal filter for threshold-based spike sorting should have a maximal response to the target neuron, while having a minimal response to the most important interfering sources (e.g. spikes from nearby neurons). Mathematically speaking, this means that the ratio between the output signal powers in both of these cases should be maximized, i.e.,

$$\mathbf{f}_i^{\mathrm{DTM}} = \underset{\mathbf{f}}{\operatorname{argmax}} \frac{\left(\mathbf{f}^T \boldsymbol{\tau}_i\right)^2}{\sum_l p^l E\left\{\left(\mathbf{f}^T \bar{\mathbf{i}}_i^l[k]\right)^2\right\}}, \tag{2.4}$$

where $p^l \in \mathbb{R}$ are weights that denote the importance of each interfering source l, and where $\bar{\mathbf{i}}_i^l[k]$ is the signal generated by the interfering source l (e.g., sharing significant spatio-temporal structure with the spikes generated by the target neuron i), also referred to as peak interferer l. As such these peak interferers might influence the threshold-based spike sorting accuracy. The filter given by (2.4) thus maximizes the filter response to the spike template, while maintaining a minimal response to the interfering sources.

We define the weighted interference covariance matrix as $\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} = \sum_l p^l E\left\{\bar{\mathbf{i}}_i^l[k]\bar{\mathbf{i}}_i^l[k]^T\right\}$, describing the spatio-temporal structure of the interfering sources. (2.4) can then be rewritten as:

$$\mathbf{f}_i^{\mathrm{DTM}} = \underset{\mathbf{f}}{\operatorname{argmax}} \frac{\mathbf{f}^T \boldsymbol{\tau}_i \boldsymbol{\tau}_i^T \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} \mathbf{f}}. \tag{2.5}$$

It is noted that the solution of (2.4) and (2.5) is only defined up to a scaling.

Assume for now that we have such an interference covariance estimate $\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}$ available which leads to a filter output with optimal SPIR. A direct-form solution for the optimization problem in (2.5) is:

$$\mathbf{f}_i^{\mathrm{DTM}} = \mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}^{-1}\boldsymbol{\tau}_i. \tag{2.6}$$

For a proof of (2.6) see Appendix 2.6.1. A key ingredient in the above filter design is the interference covariance matrix. The estimation of the interference covariance matrix is the subject of Section 2.3.3.

The type of filter discussed in this section will be referred to as a *discriminative template matching (DTM)* filter. The filter design method allows for optimally discriminating between two different spatio-temporal structures as described by a pair of covariance matrices. In the context of this work the pair of covariance matrices will consist of the target spike covariance matrix (here replaced by a rank-1 matrix based on the template $\boldsymbol{\tau}_i$) and the estimated covariance matrix of signal segments who otherwise cause spurious peaks on the filter output.

2.3.2 Illustrative example

A small neural ensemble containing three cells [42] was simulated using line-source modelling with LFPy (version 1.1.2) [68] and the results are shown in Figure 2.2. It is noted that this is purely for illustrative purposes, and this example should not be seen as a realistic case or validation of our method (we refer to Section 2.4 for a quantitative validation). For each neuron i the matched filter coefficients $\mathbf{f}_i^{\mathrm{MF}}$ are calculated according to (2.1). The matched filter outputs $y_i^{\mathrm{MF}}[k] = {\mathbf{f}_i^{\mathrm{MF}}}^T\overline{\mathbf{x}}_i[k]$ are shown in Figure 2.2**c** and are color coded according to the template's putative neuron.

One can observe from Figure 2.2**c** that the filter outputs of all three matched filtering-based template matching filters are very similar. This is due to the fact that all three templates have a similar spatial distribution. These filter outputs are not suitable for spike sorting. Even a greedy iterative approach will fail, because cell 1 has a significantly lower spike energy in the extracellular recording. In iterative schemes for resolving overlap each filter output is normalized with respect to its corresponding template energy. This will relatively boost the response of cell 2 and cell 3 on the filter output of cell 1, leading to all spikes being classified as belonging to cell 1.

In Figure 2.2**e** the color coded filter outputs of the proposed discriminative template matching filters are shown. The filter coefficients $\mathbf{f}_i^{\mathrm{DTM}}$ are calculated by setting $\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} = \sum_{j\neq i}\hat{\mathbf{R}}_{\boldsymbol{\tau}_j\boldsymbol{\tau}_j} + \alpha\mathbf{I}$ in (2.6) with $j \in \{1,2,3\}$, i.e., the interference covariance matrix is built from the templates of the two non-target spikes. A scaled version of the identity matrix $\mathbf{I}$ is added to each interference covariance matrix to make sure all interference covariance matrices are invertible.

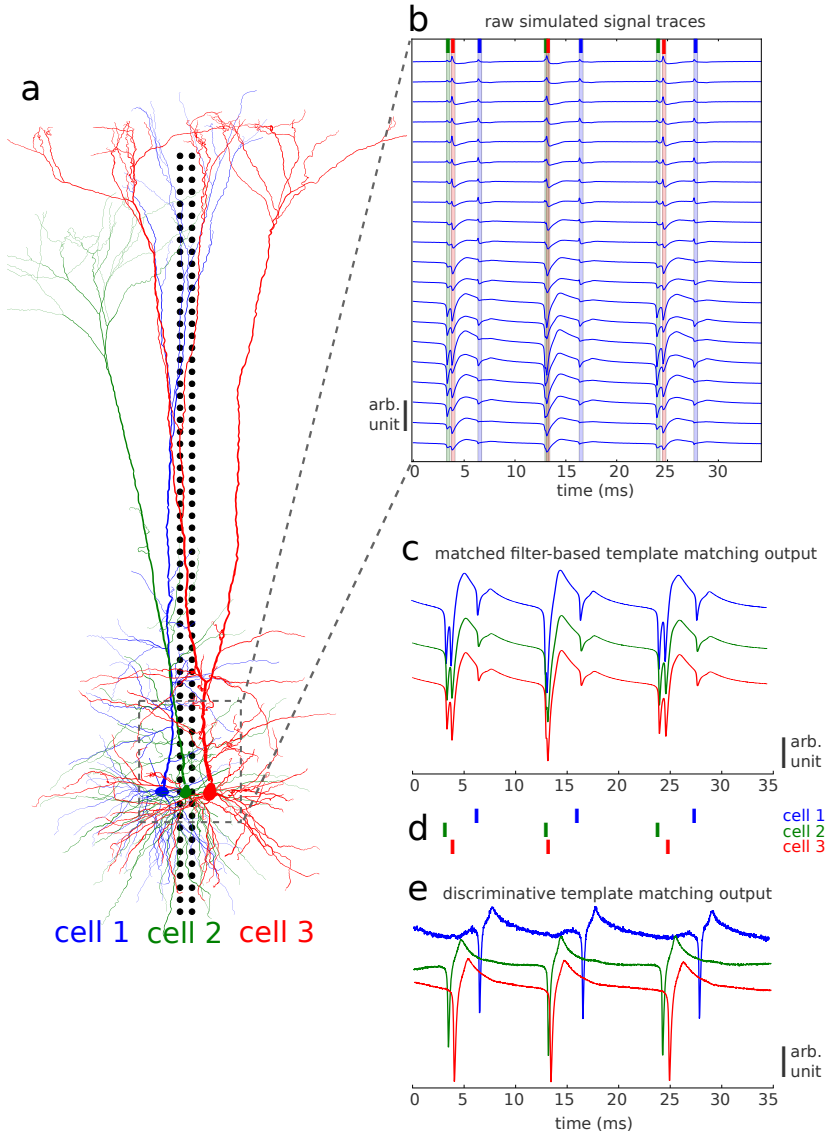

Figure 2.2: **a** Drawing of the simulated setup. The setup contains three neocortical layer 5b morphologically detailed pyramidal cells (cell 1: blue, cell 2: green, cell 3: red) and a 128-channel high density probe of which only the electrode locations are shown by black dots. **b** 35ms of simulated data for 20 out of 128 channels. Times at which the simulated neurons generate spikes are highlighted by the active cell's corresponding color. From this it can be seen that each individual neuron is active three times in the given data slice. **c** Matched filter-based template matching output of the three filters trained on the individual templates which were extracted using the ground truth spike times. **d** Ground truth spike times for each cell. **e** Discriminative template matching output of the three filters each trained using all three templates.

Thresholding the discriminative template matching outputs directly solves the spike sorting problem.

The threshold-based spike sorting capability of the discriminative template matching might seem to come at a cost. Comparing the filter outputs in Figure 2.2**c** with the filter outputs in Figure 2.2**e**, it can be seen that the filter outputs in Figure 2.2**e** are more noisy. The increased output response to background noise is acceptable here, because the background baseline noise will not cause the detection of false positives in the low noise case simulated here. We argue that the optimal spike sorting filter must focus on attenuating non-target spikes and other interfering transients while maintaining a decent response to the target spike. A quantitative validation of the proposed filter design method will be given in Section 2.4, where we will also investigate the influence of adverse noise conditions on the spike sorting performance.

2.3.3 Estimation of the interference covariance matrix

In this section a method will be proposed for estimating an interference covariance matrix $\mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i}$, which will capture the most harmful interfering sources, and hence put a larger emphasis on these in the optimization of (2.5). This matrix can then be used to design the optimal filter coefficients given by (2.6) to isolate the spikes of a single neuron while suppressing interfering spikes of other neurons or interfering transients which might cause false positives otherwise.

The interference covariance matrix estimation algorithm consists of six steps, as visualized in Figure 2.3, which have to be performed for each detectable neuron i separately. This algorithm is operated a-priori on a representative subset of the data, which we refer to as the training set (it is noted that – for offline spike sorting – the training set can in principle contain the complete data set). It uses classic template matching to compute reasonable estimates of the interference covariance matrices, which at their turn will then be used to design (better) discriminative template matching filters:

1. **Required prior knowledge**: it is assumed that a representative spike template for the target neuron is available. In practice this template is estimated by using a clustering-based spike sorting algorithm which can identify a subset of the spikes of the target neuron contained in the training set. The template is then estimated by averaging over this subset of target spikes (see (2.1)). This interference covariance estimation algorithm will also use the a priori known spike times of the target neuron.

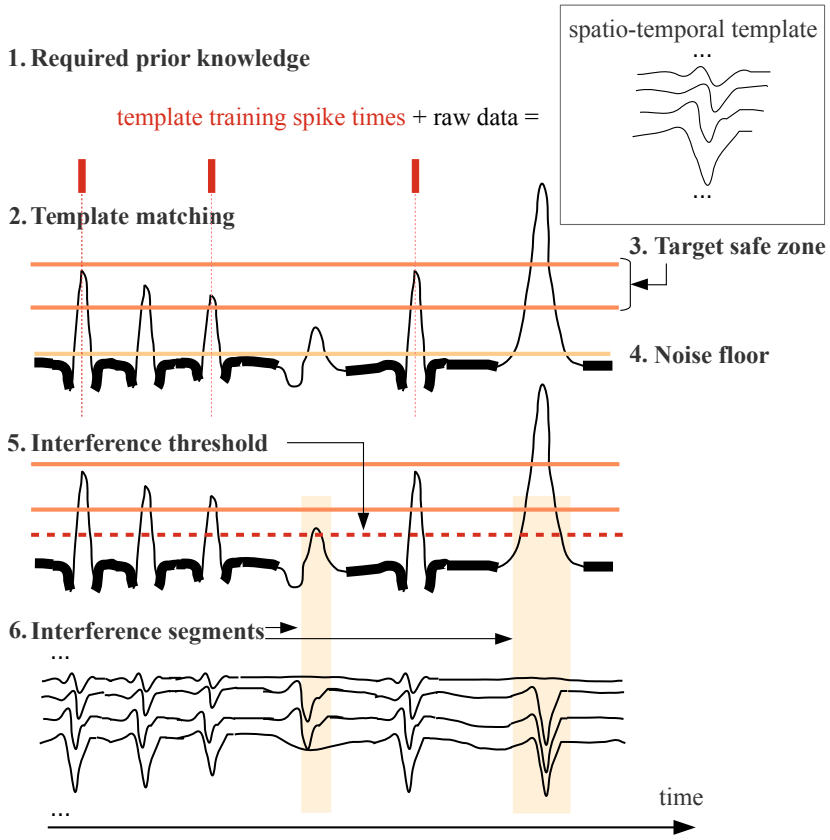

Figure 2.3: Graphical representation of the proposed algorithm for estimating the interference covariance matrix needed for the discriminative template matching filter design. **1.** The prior knowledge required for the algorithm exists of a set of example spike times for the target neuron. **2.** A matched filter is computed and applied to the recording, using the spike template as filter coefficients. The effect of background noise on the filter output is depicted by bold signal segments. **3.** In order to prevent leakage of spikes from the target neuron into the interference covariance matrix estimate, a target safe zone is determined. Each peak that falls within this target safe zone is excluded for the estimation of the interference covariance matrix. **4.** Based on the filter output statistics the noise floor is estimated. **5.** By combining both the target safe zone and the noise floor, the interference threshold is calculated. **6.** All peaks that have their maximum above the interference threshold and outside the target safe zone are used for estimating the interference covariance matrix. The signal segments involved in the estimation of the interference covariance matrix are shaded.

2. **Matched filtering**: The available target spike template is then used for a matched filtering-based template matching pass over the training set data.

3. **Target safe zone**: We assume that it is possible that several of the target spikes in the training set have not been identified in the prior clustering phase of step 1. Therefore, to avoid a too strong leakage of the spatio-temporal structure of the target spike covariance into the interference covariance estimate, a *target safe zone* is defined. This target safe zone determines the amplitude range within which the output signal of the matched filter in step 2 can be considered to be a target spike with sufficient confidence. The target safe zone is calculated from the known spike times in step 1. Using the available target neuron spike times, the maximum output response of the matched filter is determined in a finite window around each available spike time. Next the maximum $p_{\max}$ and minimum $p_{\min}$ over all these output response maxima are computed. The target safe zone is defined as the interval $((1-\zeta)p_{\min}, (1+\zeta)p_{\max})$ with, e.g., $\zeta = 0.1$.

4. **Noise floor**: The following step consists of estimating the amplitude of the baseline noise n_{est} in the output response of the matched filter over the training set data. This is done by calculating the median of the absolute value of the output response where all 2 ms windows around known spike times are ignored in the calculation to further reduce the estimation bias.

5. **Interference threshold**: The interference threshold T_{int} is then calculated as $T_{\text{int}} = \eta p_{\min} + (1-\eta)n_{\text{est}}$ with, e.g., $\eta = 0.5$. Data segments corresponding to time instants k_j where the matched filter response crosses the interference threshold and where the maximum filter response of the crossing event does not fall into the target safe zone, are classified as interference data segments. Cases exist where no interference is detected and the matched filter-based template matching is sufficiently discriminative already.

6. **Interference covariance**: With the interference data segments $\bar{\mathbf{i}}_i[k_j] = \overline{\mathbf{x}}_i[k_j]$ at time k_j available, one can estimate the covariance matrix $\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}$ characterizing the spatiotemporal coherence of the interfering events:

$$\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} = \sum_j \bar{\mathbf{i}}_i[k_j]\bar{\mathbf{i}}_i[k_j]^T. \tag{2.7}$$

This interference covariance matrix can together with the target spike template be used in (2.6) to find the filter coefficients for discriminative template matching. Due to the use of an interference threshold, only data segments with the most

relevant interference activity are included, allowing the filter design to mainly focus on the cancellation of these harmful sources. This interference threshold thus implicitly determines the weights p^l in (2.4). If all data segments without target spike would be used to estimate the interference covariance matrix, the filter design would "waste" its degrees of freedom on the cancellation of noise that is not harmful anyway. In that case, it would optimize the average SNR, but not the SPIR.

We assume that the interference covariance estimation algorithm is not perfect, and that some target spike information might be included in the interference covariance matrix estimate. However, in Appendix 2.6.2 it is shown that the filter design method can cope with such target spike leakage, and as such that imperfect interference covariance estimates are not necessarily a big concern.

If interference is sufficiently attenuated in the discriminative template matching filter output, one can recalculate the interference threshold as explained in step 5 (possibly with a different value for η) on the discriminative matching output and use that threshold as a spike detection threshold.

2.4 Validation

Discriminative template matching with an interference covariance matrix computed using the algorithm from Section 2.3.3 is compared to matched filtering-based template matching (with pre-whitening) as in Section 2.2.2 and Bayes optimal template matching (BOTM) with an iterative subtraction scheme (SIC) [31], which is a recent advanced template matching algorithm. The above three template matching methods are also compared to Spyking Circus [137], a full-blown spike sorting package with a greedy iterative final template matching stage. This final comparison allows us to contrast the template matching results with spike sorting results that are closer to what one would obtain in practice. Note that both [31] and [137] are non-single-shot methods, and as such have a non-deterministic processing delay.

2.4.1 Description of validation data set

The algorithms under examination are assessed on in-vivo extracellular ground truth data. Two different types of ground truth data are used for validation:

1. **Paired recordings** [80] are obtained by closely positioning an extracellular silicon probe to a neuron which is targeted by a juxtacellular

micro-pipette. The signal measured by the micro-pipette can be used to extract ground truth spike times, because it contains high SNR peaks at times when the targeted neuron fires an action potential. For a spike of the targeted neuron to be visible on the extracellular probe, the distance between the micro-pipette and the silicon probe should be less than 50 µm.

2. **Hybrid data** [103] is obtained from in-vivo HD recordings. Each recording is spike sorted using some available spike sorting algorithm. An operator then manually selects the spike clusters which consist of single-unit activity only. The spikes belonging to some verified single-unit cluster are then subtracted from the original recording and migrated to another spatial region resulting in a so-called hybrid recording. Since the timestamps of the newly injected spikes are known in the hybrid recording, these recordings can be used as ground truth data sets.

The specific datasets used for this work are available online [1,2].

Fifteen neurons were selected from this data for further analysis. The selection criteria used are: the existence of interfering sources for the considered neuron and a lower bound on the number of spikes (> 50 spikes) for that neuron. The spike count lower bound was chosen such that sufficient spikes are available for the estimation of templates. All templates are calculated according to (2.1), where all $\mathcal{T}_i$ are obtained from ground truth spike time information. From all recordings the first five minutes of data are analyzed.

The channel selector for each neuron passes only data from electrodes within a circular region with a radius of 100 µm around the electrode which shows the highest voltage deflection for spikes of that neuron. The exact number of electrodes N_i involved in calculating a covariance matrix is ultimately determined by the probe layout used to make the recording containing the neuron under investigation. The delay line length L for all filters is chosen such that samples from the last 1 ms can be stored, which is usually sufficient to capture a spike waveform.

Prior to the filter design as well as the actual filtering, the data is minimally preprocessed. Preprocessing of the raw extracellular data is kept limited because online algorithms require a limited processing delay. Each channel is separately bandpass filtered between 300 and 10 000 Hz using a linear phase finite impulse response filter. This filter operation extracts the multi-unit activity from the raw data and suppresses low-frequency signal content and high-frequency measurement noise.

[1] Paired recording data: http://www.kampff-lab.org/validating-electrodes/
[2] Hybrid data: http://phy.cortexlab.net/data/sortingComparison/datasets/

2.4.2 Validation parameters

We examined spike sorting sensitivity and precision for each combination of validation neuron and algorithm. *Sensitivity* is defined as the number of recovered target spikes (true positives) over the total number of ground truth spikes for that target neuron. A related measure that is important in spike sorting is precision. *Precision* is defined as the number of recovered target spikes (true positives) over all recovered events, including false detection. The importance of precision comes directly from the definition of spike sorting: unraveling multi-unit spike activity into single unit spike activity. A precision lower than 1 indicates the presence of non-target spikes/events in the spike sorting output.

For threshold-based spike sorting, sensitivity and precision depend on the value of the threshold applied at the filter output. To make it possible to compare different methods, an optimal threshold is defined for each filter output. The optimal threshold is chosen by sweeping the threshold and calculating the sensitivity and precision for each position of the threshold. Because of the importance of precision in spike sorting, the optimal threshold is chosen from all thresholds with a precision > 0.9 that maximizes the sum of the sensitivity and precision. When no thresholds are available that give a precision > 0.9, the optimal threshold is chosen as the threshold that maximizes only the precision. BOTM comes with an integrated optimal threshold. In the context of BOTM we will use this dedicated threshold, unless stated otherwise.

2.4.3 Signal-to-peak-interference maximization

Before showing quantitative results, the key mechanism of the discriminative template matching filter, namely maximizing the SPIR at its output, is graphically validated for in-vivo recorded data. For this purpose a segment of both a matched filter output and a discriminative template matching output is shown in Figure 2.4. The target neuron spiked once over the duration of the depicted segment, this spike time is indicated by the red asterisk.

For this segment the matched filter output (blue) has a clear peak when the target neuron is active. Besides the peak coinciding with the neuron being active, two more minor peaks can be found in the matched filter output later in time. The discriminative template matching filter output (green) also has a significant peak when the target neuron spikes. Apart from this activity-related peak, there are no other clear peaks in the filter output. This observation is also reflected by the increase in SPIR for the depicted segment between the matched filtering output and the discriminative template matching output. This increase

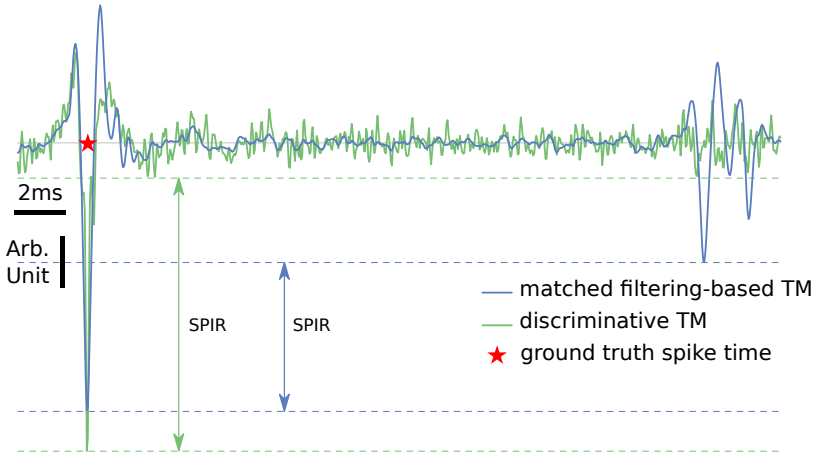

Figure 2.4: The output of both the matched-filtering based template matching (blue) and the discriminative template matching with interference suppression (green) filter applied to in-vivo recorded data is shown. Both filter outputs are normalized with respect to their standard deviation. The ground truth spike time of the target neuron in this segment, is indicated by a red asterisk. The double arrows indicate the signal-to-peak-interference ratio for both outputs.

in SPIR is beneficial for the accuracy and robustness of the threshold-based spike sorting scheme.

From Figure 2.4 it can also be seen that the increase in SPIR comes at a cost. This cost is an increase of the baseline noise output response (as was already pointed out in Section 2.3.2). Up to some extent this increase of the baseline noise output response will not harm the threshold-based spike sorting performance.

2.4.4 Spike sorting performance

Figure 2.5**a** shows the spike sorting sensitivity and precision for each validation neuron for the four different methods under investigation. First threshold-based spike sorting using matched filters (blue) is compared to threshold-based spike sorting using discriminative template matching (green). It can be observed that the use of discriminative template matching filters generally outperforms the use of matched filter in terms of the sum of sensitivity and precision. In terms of sensitivity and precision separately the use of discriminative template matching

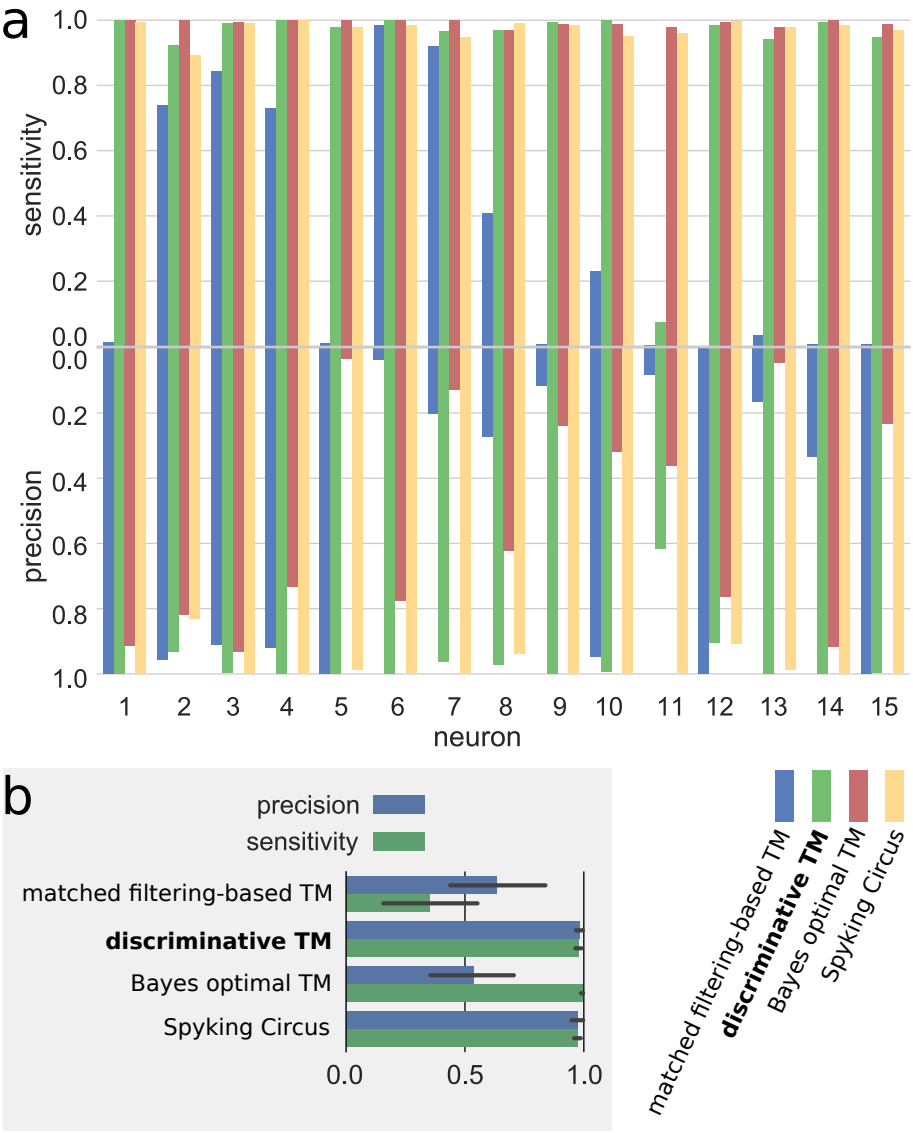

Figure 2.5: **a** The spike sorting sensitivity and precision for all analyzed ground truth neurons are shown. For each neuron four different algorithms were tested: template matching threshold-based (blue), discriminative template matching with interference suppression threshold-based (green), Bayes optimal template matching [31] (red), and Spyking Circus [137] (yellow). **b** Summary of precision (blue) and sensitivity (green) over all neurons for the four methods. The error bars indicate a 95 percent confidence interval.

outperforms matched filtering, with the exception of neuron 12 which has a notably higher precision. However this precision is only reached at an almost zero sensitivity, making this matched filter practically useless. This initial comparison confirms that a combination of matched filtering-based template matching and a simple threshold in the context of spike sorting is of no practical value.

Next, discriminative template matching is compared to BOTM. BOTM (red) slightly outperforms discriminative template matching in terms of sensitivity (Figure 2.5**b**), but its precision is considerably lower than discriminative template matching. Two mechanisms are responsible for the difference in performance between these two algorithms: the fact that different thresholds are selected and the way interference is handled, which will be further discussed below.

As mentioned before, discriminative template matching employs a threshold focused on optimizing the precision, whereas the BOTM threshold seems to favor sensitivity maximization. This difference in threshold makes it difficult to draw strong conclusion from this comparison. Therefore, the performance of a modified version of BOTM is determined. This modified version uses detection thresholds which are calculated in the same way as the thresholds used for discriminative template matching. This experiment results in an average sensitivity of 53.3% and a precision of 71.8%. Although the precision increases substantially (cf. Figure 2.5**b**), it is still low compared to the discriminative template matching results. Also, BOTM's sensitivity with this new threshold is roughly half compared to the original BOTM algorithm. This result indicates that another mechanism is responsible for the performance difference found on our dataset.

A more relevant mechanism to discuss is the way in which both algorithms handle interference. BOTM assumes that the templates of all spiking neurons in the recording are known. From these known templates it either builds compound templates (which is computationally expensive) or makes use of an iterative subtraction scheme to resolve spike overlap. However, in practice it often happens that not all spiking neurons' templates are known, e.g., multi-unit activity clusters do not result in useful templates. As such, when the wave shape of an interfering neuron is not known, the template matching procedure will likely assign spikes of such unaccounted for neurons to faulty neuronal sources. BOTM also does not take into account impulsive noise events which are likely to be missed while calculating the interference covariance matrix and as such can also trigger false detections. Since our experiments are conducted on in-vivo data, we don't have full knowledge about all active neurons and their corresponding templates. Our proposed algorithm has the practical advantage that it only assumes partial knowledge about the activity of the neurons for which single-unit spike times have to be retrieved, because the interference is

handled in a data-driven way. This fundamental difference is also an explaining factor for the difference in precision between BOTM and discriminative template matching.

Next, threshold-based spike sorting using discriminative template matching is compared to Spyking Circus (yellow). Spyking Circus is a state-of-the-art spike sorting pipeline, which employs a greedy template matching algorithm (with iterative template subtraction) for resolving overlapping spikes. To allow for a fair comparison between both methods, the manual curation phase of Spyking Circus is replaced by ground truth assisted cluster merging. From Figure 2.5**a** it can be seen that our discriminative template matching algorithm performs very similar to Spyking Circus in terms of both sensitivity and precision, except for neuron 11.

The reason for the failure of threshold-based spike sorting using discriminative template matching on neuron 11 comes from the fact that the response of the interfering neurons to the initial matched filter (see step 2 of the interference covariance matrix estimation algorithm in Subsection 2.3.3) by accident falls within the target spike safe zone. This leads to a discriminative template matching filter which does not put any effort in minimizing this specific peak interferer. This problem can be alleviated during the filter design phase by re-estimating the interference covariance matrix on the intermediate discriminative template matching outputs in the training set. As the initial discriminative template matching filter alters the response to the interfering neuron compared to the original matching filter (as both are based on different filter design paradigms), it will not fall in the spike safe zone anymore with a high probability. As a result, the interfering neuron will now be included in the interference covariance matrix for the re-estimation of the discriminative template matching filter. Such an approach leads for neuron 11 to a spike sorting sensitivity and precision above 95 percent. For the other neurons, applying the interference covariance estimation on the initial matched filter output proved sufficient.

The Spyking Circus performance results also require us to reflect on the BOTM performance. The Spyking Circus results indicate that more prior clustering information is potentially available within the data, than the prior clustering information (i.e., available ground truth spike times) we used to estimate templates of the interfering neurons for BOTM. However, curating all prior clustering information is labour intensive in practice and as such not desirable. Also, there is no guarantee that the prior clustering stage can resolve all interference as single-unit clusters, which are required to estimate interfering templates.

2.4.5 Robustness to noise

In this section the effect of noise on both discriminative template matching and BOTM is investigated. In this section we will focus on data from ground truth neuron 3. This neuron is chosen for two reasons. The first reason is because both algorithms perform equally well on the original data (Figure 2.5), indicating sufficient prior knowledge was available for BOTM to assign non-target spikes to their non-target neuronal sources. As such we can fairly compare the performance of both algorithms for different noise levels. The second reason is that the original data from ground truth neuron 3 has a low noise level, such that we have a lot of flexibility to artificially control the noise conditions (to control SNR, we can only add noise, but not remove existing noise).

For this experiment we added noise of different peak-signal-to-noise ratios (PSNR) to the original data before training the different template matching filters. The PSNR (not to be confused with SPIR) is defined as the peak power of the template divided by the noise power. The noise that we added was spatially correlated pink noise. The spatial correlation matrix and the frequency domain slope were extracted from an in-vivo probe recording. The spatial correlation matrix was obtained by calculating the sample correlation matrix of noise-only data segments between the desired number of neighbouring electrodes. The temporal correlation was modeled by fitting a high-pass filtered $f^{-\gamma}$ curve to the frequency spectrum of single probe electrode recordings. This procedure resulted in $\gamma = \frac{6}{5}$. The phases of the different frequency components were randomly generated by sampling a uniform distribution. As such, the synthetically generated noise was similar to experimental noise. Before adding the synthetic noise to the data it was scaled w.r.t. the template peak power of the original data to obtain the desired PSNR. As shown in Figure 2.6, the PSNR of the synthetic noise ranges from -20 dB to 5 dB.

The performance differences between discriminative template matching and BOTM in this experiment (Figure 2.6), are likely due to different thresholds being used. When the template estimate energy is dominated by noise (-20 to -10 dB), BOTM will subtract this energy from the filter output, making it nearly impossible to detect spikes. On the other hand the result obtained with the discriminative template matching should be interpreted with care. The threshold applied to the filter output relies on prior information. Although both template matching algorithms use this prior information to estimate relevant templates, discriminative template matching also uses this information to determine the threshold. In practice this prior information will not be available at very high noise levels.

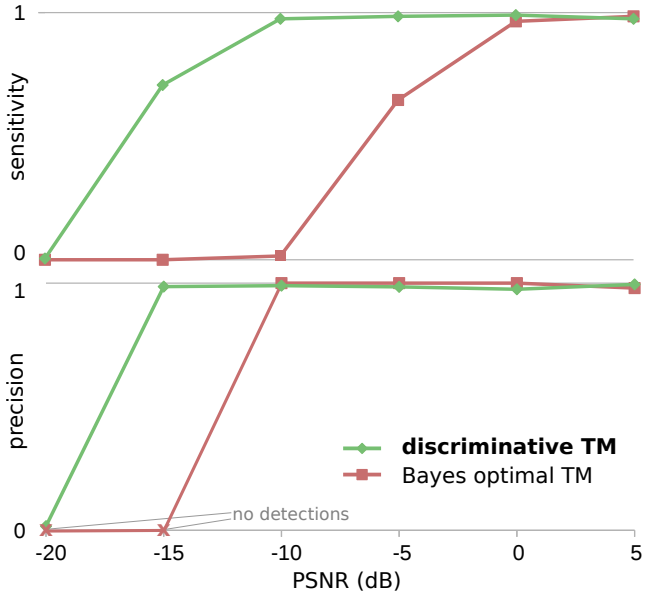

Figure 2.6: Sensitivity and precision for discriminative template matching and Bayes optimal template matching under adverse noise conditions. If no spikes are detected, the precision metric is undefined. By convention we depict an undefined precision by a cross marker located at zero.

The message we want to convey here is that the interference covariance estimation algorithm is capable of also catching the noise covariance. This is due to the fact that the interference covariance matrix is estimated on data instead of templates. The interference segments thus contain the sum of interference and background noise. As such, at very high noise levels the interference covariance matrix will be dominated by the background noise.

We conducted another experiment where we trained a discriminative template matching filter on data with a 5 dB PSNR. We then applied this filter to data with a lower PSNR (4 to -20 dB). This resulted in a sensitivity and precision as shown in Figure 2.7. The sensitivity and precision for this experiment were obtained using a constant threshold. The fixed threshold was determined on the 5 dB PSNR training data. Again, this indicates that our filter design method is capable of extracting the background noise covariance and that the filter is also robust to changes in background noise levels.

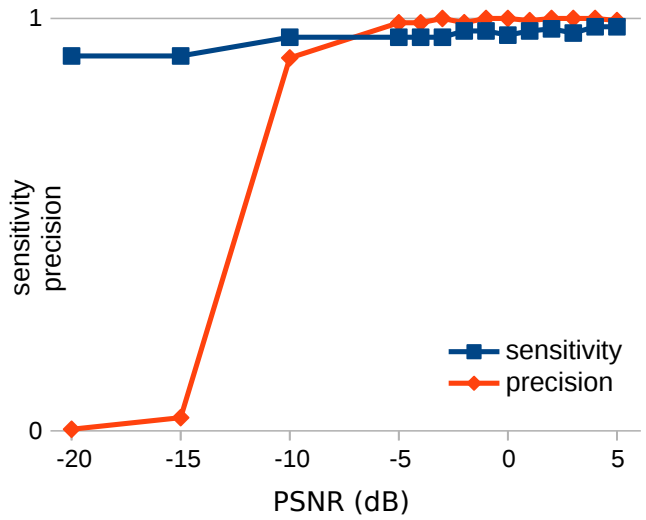

Figure 2.7: Sensitivity and precision for discriminative template matching under adverse noise conditions using only one discriminative template matching filter and threshold trained on 5 dB PSNR data.

2.5 Discussion and Conclusion

This work focused on a data-driven discriminative template matching filter design method for the threshold-based spike sorting pipeline, which is promising for online spike sorting and closed-loop neuroscience. The idea behind threshold-based spike sorting is that single-unit activity can be obtained from a simple thresholding operation on the output of each template matching filter. Such a pipeline has a low computational complexity and a predictable delay. However, the template matching filters have to sufficiently discriminate between the target neuron and interfering neurons for the threshold-based spike sorting to work.

It was demonstrated that a matched filtering-based template matching approach is not suitable for use in threshold-based spike sorting. Matched filtering methods have to rely on iterative schemes to extract single-unit activity from the template matching filter outputs. Therefore, the discriminative template matching method was introduced which was shown to outperform the matched filtering approach in terms of threshold-based spike sorting performance.

We also compared the performance of our data-driven discriminative template matching procedure to Bayes optimal template matching with an iterative subtraction scheme. It was demonstrated that discriminative template matching

outperforms Bayes optimal template matching on the analyzed dataset, because discriminative template matching can handle incomplete prior clustering information. We've also shown that our method can handle different background noise levels, because the interference covariance is estimated in a data-driven way. Although good spike sorting results were obtained at high noise levels in our experiments, it should be noted that it is unlikely that one can create template matching filters for such low SNR spikes in the first place, due to the difficulty of extracting the necessary prior clustering information for high noise levels.

A limitation of most template matching algorithms is that in case of consistent spike overlap between spikes from different neurons, the initial clustering phase will not be able to retrieve the required prior information for our filter design method. In this case our method is deemed to fail in retrieving the spike times of those specific neurons.

Another potential source of failure are neurons with spike templates that are equal up to a scaling factor. In such a case the discriminative template matching filter will not be able to suppress the interference generated by a neuron with similar template. A possible solution to this problem is to integrate an upper and lower threshold for detection on the filter output.

For bursting neurons, our filtering approach is straightforwardly applicable, if the bursting neuron is correctly identified during the initial clustering stage. Depending on the specific bursting variability between spikes of the target neuron, one can estimate a single template or use higher-rank templates (Appendix 2.6.1). If the bursting neuron is split during the initial clustering phase, one has to rely on additional post-processing to merge the detection outputs of all filters related to that neuron.

When comparing the single-shot threshold-based spike sorting performance with the performance of a widespread spike sorting package like Spyking Circus [137], it was shown that both approaches have a similar spike sorting performance. These results indicate that discriminative template matching is a viable building block for threshold-based spike sorting packages. Such threshold-based spike sorting algorithms have the advantage of algorithmic simplicity and a low (and deterministic) input-output delay, making it suitable for online closed-loop neuroscience experiments.

The power of discriminative template matching lies in the way of estimating the so-called interference covariance matrix, which is an important ingredient of the filter design method. The interference covariance matrix estimation algorithm implicitly weights the different interfering sources, taking into account whether or not the segments cause spurious peaks on a matched filtering template matching

output, and as such are classified as peak interferers. The discriminative template matching filter can be thought of as optimizing the SPIR, instead of the SNR as is the case for a matched filter.

An approach also trying to suppress interfering spikes was already proposed in [100] [126] [30]. However, this template matching technique did not make it into today's spike sorting packages. One possible reason for this is given by [29], where the filters are argued to be less robust to noise, because they are designed under stronger constraints. We believe that the advent of HD probes results in increased spatial information and as such brings new opportunities for template matching filters with an extra constraint on suppression of interference. Discriminative template matching differs from approaches like [100] mainly in the way the interference is estimated. Instead of a template-based approach, the interference in our filter design method is estimated in a data-driven way. Such a data-driven approach removes the necessity of a complete prior spike sorting, i.e., only example spike times of target neurons of interest are required as an input. Another advantage of our approach over template-based approaches is that different interfering sources are automatically weighted based on their harmfulness.

Another advantage of threshold-based spike sorting is that it could be a candidate for on-probe spike sorting. The need for on-probe sorting might arise from an ever increasing channel count on HD probes together with a limited data transfer bandwidth, making it a challenging task to transfer raw data from a probe to, e.g., a computer. The simple architecture of a finite impulse response filter and the low complexity of a single thresholding operation, can be cheaply implemented in hardware. Recently, active CMOS probes [95] have been developed, already having integrated electronics on probe. On-probe sorting also eliminates the transfer delay, which is beneficial for use in closed-loop experiments.

2.6 Appendix

2.6.1 Derivation of the optimal filter

The proposed filter $\mathbf{f}_i^{\mathrm{DTM}}$ for threshold-based single-unit activity detection of neuron i is the solution to the optimization problem defined below:

$$\max_{\mathbf{f}} \frac{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} \mathbf{f}}, \tag{2.8}$$

where

$$\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i} = \boldsymbol{\tau}_i\boldsymbol{\tau}_i^T \tag{2.9}$$

is the positive semidefinite spatio-temporal covariance matrix of the deterministic spike template waveform and $\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}$ is the positive definite spatio-temporal peak-interference covariance matrix for neuron i. The filter satisfying (2.8) thus optimizes the signal-to-peak-interference ratio at its output. To derive a closed-form solution, (2.8) is rewritten as:

$$\max_{\mathbf{f}} \mathbf{f}^T\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\mathbf{f} \quad \text{s.t.} \quad \mathbf{f}^T\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\mathbf{f} = K, \tag{2.10}$$

with $K \in \mathbb{R}$ an arbitrary constant. One can easily see that maximizing a fraction is equal to maximizing the numerator while keeping the denominator constant. To solve this constrained optimization problem, the method of Lagrange multipliers [10] is used. The Lagrangian of this optimization problem is given by:

$$\mathcal{L}(\mathbf{f}, \lambda) = \mathbf{f}^T\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\mathbf{f} - \lambda\left(\mathbf{f}^T\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\mathbf{f} - K\right). \tag{2.11}$$

The solution to the optimization problem is then found by setting the partial derivative w.r.t. $\mathbf{f}$ equal to zero and solving it for $\mathbf{f}$.

$$\frac{\partial\mathcal{L}}{\partial\mathbf{f}} = 2\left(\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i} - \lambda\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\right)\mathbf{f} = \mathbf{0}. \tag{2.12}$$

Rearranging (2.12) reveals that (2.8) reduces to a generalized eigenvalue problem [35], where the optimal filter coefficients correspond to a generalized eigenvector:

$$\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\mathbf{f} = \lambda\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\mathbf{f}. \tag{2.13}$$

By combining (2.10) and (2.13), we get

$$\mathbf{f}^T\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\mathbf{f} = \lambda\mathbf{f}^T\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\mathbf{f} = \lambda K. \tag{2.14}$$

This shows that (2.10) is maximized if λ is maximal, i.e., if $\mathbf{f}$ is chosen equal to the generalized eigenvector corresponding to the maximal generalized eigenvalue λ, i.e., the solution of (2.13) for maximal lambda.

In this work the template covariance matrix is a rank-1 matrix by definition as shown in (2.9). This rank-1 structure makes that there is only one generalized eigenvector which has a non-zero eigenvalue, which then also leads to a closed-form solution. This can be seen from substituting (2.9) in the generalized eigenvalue problem (2.13):

$$\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}^{-1}\boldsymbol{\tau}_i\boldsymbol{\tau}_i^T\mathbf{f} = \lambda\mathbf{f}. \tag{2.15}$$

It is noted that the left-hand side of (2.15) consists of a scaled version of the vector $\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}^{-1}\boldsymbol{\tau}_i$ scaled by the scalar $\boldsymbol{\tau}_i^T\mathbf{f}$. As a result, the closed-form solution of (2.8) is:

$$\mathbf{f}_i^{\mathrm{DTM}} = \mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}^{-1}\boldsymbol{\tau}_i. \tag{2.16}$$

It is noted that the GEVD-based approach also allows for the use of higher-rank template covariance matrices $\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}$. The optimal filter can then be found by solving (2.13) using a generic generalized eigenvalue problem solver [35]. This might be beneficial in the case of neurons with non-stationary spike waveforms, e.g. bursting neurons.

2.6.2 Robustness against target leakage

Assuming that the interference covariance estimation algorithm is imperfect, at some point, target spike segments will leak into the interference covariance matrix estimate. Such a faulty interference covariance matrix $\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}$ can be modeled as:

$$\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} = \mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} + c\,\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}, \tag{2.17}$$

with $c \in \mathbb{R}$.

The discriminative template matching filter for the pair of matrices $\left(\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}, \hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\right)$ is the solution to the following optimization problem:

$$\mathbf{f}_i^{\mathrm{DTM}} = \underset{\mathbf{f}}{\operatorname{argmax}} \frac{\mathbf{f}^T\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\mathbf{f}}{\mathbf{f}^T\left(\mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i} + c\,\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}\right)\mathbf{f}}. \tag{2.18}$$

Through algebraic manipulation of (2.18), it is shown that the discriminative template matching filter for $\left(\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}, \hat{\mathbf{R}}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\right)$ is equal to the discriminative template matching filter for $\left(\mathbf{R}_{\boldsymbol{\tau}_i\boldsymbol{\tau}_i}, \mathbf{R}_{\bar{\mathbf{i}}_i\bar{\mathbf{i}}_i}\right)$:

$$
\begin{aligned}
&\underset{\mathbf{f}}{\operatorname{argmax}} \frac{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}}{\mathbf{f}^T \left(\mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} + c\, \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \right) \mathbf{f}} \\
&= \underset{\mathbf{f}}{\operatorname{argmin}} \frac{\mathbf{f}^T \left(\mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} + c\, \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \right) \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}} \\
&= \underset{\mathbf{f}}{\operatorname{argmin}} \left(\frac{\mathbf{f}^T \mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}} + c \right) \\
&= \underset{\mathbf{f}}{\operatorname{argmin}} \frac{\mathbf{f}^T \mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}} \\
&= \underset{\mathbf{f}}{\operatorname{argmax}} \frac{\mathbf{f}^T \mathbf{R}_{\boldsymbol{\tau}_i \boldsymbol{\tau}_i} \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i} \mathbf{f}}.
\end{aligned}
\tag{2.19}
$$

We can conclude that target spike leakage into the interference covariance matrix, as modeled by (2.17), has no influence on the calculation of discriminative template matching filter coefficients. In practice, the filter coefficients will slightly differ because the leakage covariance will be calculated on a per spike basis including the background noise. This additional background noise will slightly change the relative weights between the interfering spike covariance and the background noise within $\hat{\mathbf{R}}_{\bar{\mathbf{i}}_i \bar{\mathbf{i}}_i}$.

Chapter 3

Data-driven regularization for template matching in spike sorting

This chapter has been published as *Jasper Wouters, Fabian Kloosterman, and Alexander Bertrand. "A data-driven regularization approach for template matching in spike sorting with high-density neural probes." 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany. IEEE, 2019.* Jasper Wouters has co-developed the methodology, conducted the experiments and has written the manuscript. Compared to the publication, minor textual and notational changes have been implemented for better integration in this thesis.

Optimal pre-whitened template matching filters that are used in spike sorting typically suffer from ill-conditioning. In this chapter, we investigate the origin of this ill-conditioning and the way in which it influences the resulting filters. Two data-driven subspace regularization approaches are proposed, and those are shown to outperform a regularization approach used in recent literature. The comparison of the methods is based on ground truth data that are recorded in-vivo.

3.1 Introduction

Extracellular microelectrodes are often used for studying brain dynamics at the level of individual neurons. Such microelectrodes can measure, among other signals, the extracellular potential changes caused by action potentials, also known as spikes, from nearby neurons. Typically, multiple electrodes are used for a given experiment and their mutual spatial organisation is matched to the targeted brain region. A popular electrode configuration used in hippocampal studies is the wire tetrode [81], reflecting the region's shallow cell layer. When studying cortical areas, probes [77] are more popular because they allow to measure along the entire cortical multilayer cell structure.

Single electrodes tend to pick up spikes from multiple neurons, while multiple electrodes pick up the spikes from a specific neuron. On top of that, multiple neurons can be simultaneously active. To study the dynamics of the *individual* neuron, one has to solve this neuronal cocktail party problem, i.e., unmixing the recorded spiking activity of the individual neurons that are embedded in the recording. A multitude of algorithmic pipelines have been proposed to achieve such unmixing. These pipelines are commonly referred to as spike sorting [67] algorithms.

A typical spike sorting algorithm consists of the following steps [33]: after *preprocessing* the recorded data a *spike detection* is performed, then *features* are extracted from those detected spikes, a *clustering* algorithm is applied to the extracted features, and finally a *template matching* phase is conducted for classifying the spikes according to their putative neurons.

Template matching is used here for classification [74] because it has the ability to resolve overlapping spikes. Template matching is also interesting for real-time spike sorting, e.g., for a neural prosthesis based on single-cell activity [136]. Although template matching is the final step in the algorithmic pipeline, the template matching filters are applied to the preprocessed data directly. As such they are suitable for streaming data applications after an initial training phase.

In this chapter we will study a failure mechanism in the template matching filter design, namely that of the ill-conditioned noise covariance matrix. Most spike sorting algorithms propose pragmatic solutions that seem to ignore the source of the ill-conditioning. The template matching filters obtained through such pragmatic approaches are suboptimal. We will propose a meaningful regularization approach, which is in line with the underlying failure mechanism.

The remainder of this chapter is organized as follows: in Section 3.2 template matching is formally presented and the ill-conditioning failure mechanism is studied. Section 3.3 proposes two data-driven regularization approaches that

can overcome the ill-conditioning. In Section 3.4 the proposed methods are validated. Finally Section 3.5 concludes this chapter with a brief discussion.

3.2 Template matching

Consider $\mathbf{x}_i[k] \in \mathbb{R}^{N_i}$ to be a selection of N_i channels of the high-pass filtered extracellular recording sampled at discrete time k, where typically only the channels are included that are relevant to isolate the spikes of a specific target neuron i. To represent the spatio-temporal L-tap delay line at time k the following stacked vector $\bar{\mathbf{x}}_i[k] \in \mathbb{R}^{LN_i}$ is constructed:

$$\bar{\mathbf{x}}_i[k] = \left[\mathbf{x}_i[k]^T \;\; \mathbf{x}_i[k-1]^T \;\; \ldots \;\; \mathbf{x}_i[k-L+1]^T\right]^T, \tag{3.1}$$

where L should be at least as long as the spike length of the target neuron.

In this chapter we focus only on the template matching filter design for spike sorting. Therefore, we assume that for every sortable neuron i we have some example spike times $\mathcal{T}_i$ available. In practice these example spike times are obtained from the spike sorting clustering phase.

Without loss of generality, in the remainder of this chapter we will focus on a single neuron, hence the neuron-related subscript is dropped. The spike template $\boldsymbol{\tau}$ for the neuron of interest can be calculated as follows:

$$\boldsymbol{\tau} = \underset{s\in\mathcal{T}}{\text{median}}\; \bar{\mathbf{x}}[s], \tag{3.2}$$

where the median is applied in an element-wise fashion. The median operator was chosen because it provides robustness against outliers in the training data. The classical optimal template matching filter [31] [130] that can be used for the classification of spikes is the solution to the following optimization problem:

$$\max_{\mathbf{f}} \frac{\mathbf{f}^T\boldsymbol{\tau}\boldsymbol{\tau}^T\mathbf{f}}{\mathbf{f}^T\mathbf{R}_{\overline{\mathbf{nn}}}\mathbf{f}}, \tag{3.3}$$

with $\mathbf{R}_{\overline{\mathbf{nn}}} \in \mathbb{R}^{LN_i\times LN_i}$ the spatio-temporal noise-only covariance matrix, i.e., $\sum_{k\in\mathcal{N}} \bar{\mathbf{x}}[k]\,\bar{\mathbf{x}}[k]^T$ with $\mathcal{N}$ containing all the sample times in which there is no spiking activity. The closed-form solution to (3.3) is given by [31] [130]:

$$\mathbf{f} = \mathbf{R}_{\overline{\mathbf{nn}}}^{-1}\boldsymbol{\tau}. \tag{3.4}$$

The actual spike classification is based on the filter output $\mathbf{f}^T\bar{\mathbf{x}}[k]$. This corresponds to a matched filter which also performs an implicit pre-whitening,

thereby maximizing the signal-to-noise ratio at the target neuron spikes. Within the scope of this chapter we will classify the spikes based on thresholding the squared template matching filter output. This approach is most in line with the objective function (3.3), and it gives a good indication about the quality of the obtained filters for use in spike sorting. However, in practice more advanced classification schemes based on, e.g., multiple thresholds and iterative filtering exist [137].

The filter given by (3.4) only exists if the spatio-temporal covariance matrix is invertible. The matrix is invertible if and only if it's null space is trivial, i.e., $\text{Null}\,(\mathbf{R}_{\overline{\mathbf{nn}}}) = \{\mathbf{0}\}$ [63]. If the covariance matrix is non-invertible, then it has a null space different from the zero-vector in which case, the denominator in (3.3) can be made arbitrarily small by choosing $\mathbf{f}$ "close" to that null space. Assuming that the template is not perfectly orthogonal to that null space, the objective function can be made arbitrarily large, without taking into account the template itself.

In practice most covariance matrices will be numerically invertible, if sufficient data are used for their estimation. This can be explained by the fact that the signals under investigation originate from experimental recordings. The recording electronics will typically introduce low power white noise which will lead to matrices that are full rank, therefore invertible. However, the power dominant spatio-temporal neural data typically have a low rank structure, especially when using modern high-density probes with spatial oversampling. Under the low power white noise assumptions, those matrices will often be ill-conditioned, i.e., they contain an eigenspace with corresponding eigenvalues that are several orders of magnitude smaller than the eigenvalues related to the neural data. As such the denominator in (3.3) can be made very small by pushing $\mathbf{f}$ towards this eigenspace (not arbitrarily small as was the case for the non-invertible case), thereby maximizing the objective function. Here, again there is a high probability that the filter response to the template is not dominant. Such a filter cannot emphasize the spiking activity of its target neuron, hence it is useless in a spike sorting context.

3.3 Regularization

Figure 3.1 shows the eigenvalues of $\mathbf{R}_{\overline{\mathbf{nn}}}$, i.e., the signal power across the different principal components in a principal component analysis (PCA) [120], from two of the spatio-temporally expanded in-vivo recordings used in Section 3.4. Most signal power is restricted to a few PCA components, indicating that the data are ill-conditioned. As explained in the previous section, applying (3.4)

without proper regularization will lead to filters that are practically useless for spike sorting purposes. Another need for regularization arises when only a limited number of samples are used for estimating the covariance matrix. In the remainder of this chapter we will expand on the topic of regularization within the context of template matching-based spike sorting.

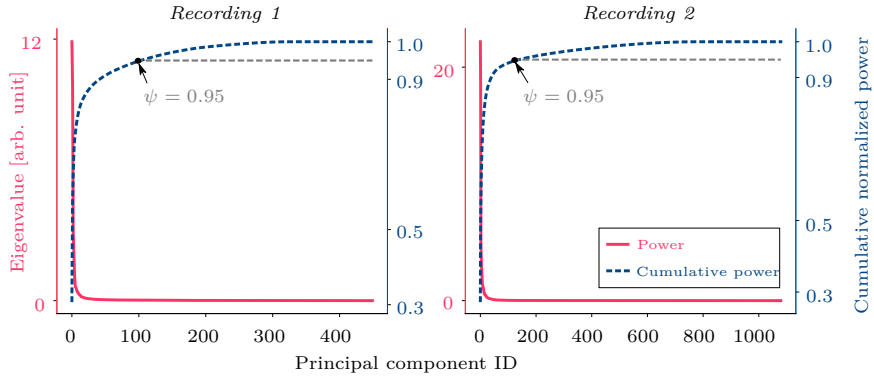

Figure 3.1: Power along every principal component (red) for the spatio-temporally expanded recording data of neuron 1 and neuron 2. The cumulative normalized power (dashed blue) is also shown and the ψ-fraction point (see Section 3.3.2) is indicated by the black arrow.

In recent spike sorting literature [137], the ill-conditioning is practically tackled by only considering the spatial covariance for the design of the spatio-temporal matched filter. On top of that a diagonal loading of the covariance matrix is performed to further improve the condition. This approach can be summarized by the following optimization problem which has the same form as (3.3):

$$\max_{\mathbf{f}} \frac{\mathbf{f}^T \boldsymbol{\tau}\boldsymbol{\tau}^T \mathbf{f}}{\mathbf{f}^T \left(\begin{bmatrix} \mathbf{R_{nn}} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{R_{nn}} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{R_{nn}} \end{bmatrix} + c\mathbf{I} \right) \mathbf{f}}, \tag{3.5}$$

where $\mathbf{R_{nn}} \in \mathbb{R}^{N_i \times N_i}$ is the spatial noise-only covariance matrix (based on $\mathbf{x}[k]$ instead of $\bar{\mathbf{x}}[k]$) and c is the diagonal loading constant which is a fixed parameter and data independent. By using this practically regularized filter design formulation the existing temporal covariance in the signal is ignored. The filters that result from this approach are suboptimal detectors from a spatio-temporal point of view. We will use this *spatial-only* regularization approach for benchmarking our proposed methods.

3.3.1 Use the signal+noise covariance matrix

From a matched filtering point of view there is no reason to exclude spiking activity samples from the noise covariance matrix estimate. One can show that including the template from the neuron of interest in the noise covariance does not change the optimal filter (see Section 2.6.2 for proof):

$$\arg\max_{\mathbf{f}} \frac{\mathbf{f}^T \boldsymbol{\tau}\boldsymbol{\tau}^T \mathbf{f}}{\mathbf{f}^T \left(\mathbf{R}_{\overline{\mathbf{nn}}} + \alpha \boldsymbol{\tau}\boldsymbol{\tau}^T\right) \mathbf{f}} = \arg\max_{\mathbf{f}} \frac{\mathbf{f}^T \boldsymbol{\tau}\boldsymbol{\tau}^T \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\overline{\mathbf{nn}}} \mathbf{f}}, \tag{3.6}$$

where α is an arbitrary weight factor. Therefore, we propose to use the signal+noise covariance matrix instead of the noise-only covariance matrix, i.e., we replace $\mathbf{R}_{\overline{\mathbf{nn}}}$ with $\mathbf{R}_{\overline{\mathbf{xx}}} = \sum_k \bar{\mathbf{x}}[k] \bar{\mathbf{x}}[k]^T$. This effectively increases the number of available samples used for the estimation of the covariance matrix, thereby making it invertible without regularization. This choice of covariance matrix also better describes the true underlying noise structure, which does include spikes from other (interfering) neurons[1]. The optimal filter design objective function we will consider is then:

$$\max_{\mathbf{f}} \frac{\mathbf{f}^T \boldsymbol{\tau}\boldsymbol{\tau}^T \mathbf{f}}{\mathbf{f}^T \mathbf{R}_{\overline{\mathbf{xx}}} \mathbf{f}}. \tag{3.7}$$

The objective function in (3.7) still suffers from ill-conditioning due to the low rank structure of the neural signal. To solve for this we propose a regularization approach that involves the design of the filter in a lower dimensional subspace.

3.3.2 Filter design in PCA subspace

We define a PCA subspace that accounts for a ψ-fraction of the total energy in the recorded (preprocessed) signal. A typical value would be $\psi = 0.95$. The subspace is readily available through the eigendecomposition of the covariance matrix:

$$\mathbf{R}_{\overline{\mathbf{xx}}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T, \tag{3.8}$$

where $\mathbf{U}$ contains the eigenvectors in its columns. Without loss of generality, we assume that the diagonal matrix $\boldsymbol{\Lambda}$ has all of its diagonal elements λ_i arranged in a descending order. The subspace is spanned by the columns of $\mathbf{U}_\psi$:

$$\mathbf{U}_\psi = [\mathbf{u}_1 \dots \mathbf{u}_k] \text{ with } k = \max\left\{ l \;\middle|\; \frac{\sum_{i=1}^{l} \lambda_i}{\operatorname{tr}(\boldsymbol{\Lambda})} < \psi \right\}, \tag{3.9}$$

[1] In Chapter 2 it has been shown that a more discriminative filter is obtained if $\mathbf{R}_{\overline{\mathbf{nn}}}$ is replaced with an interference covariance matrix that is *only* built from segments containing interfering spikes. In this case, the matrix may again become non-invertible, which is resolved by using the PCA preprocessing as explained in Sections 3.3.2 and 3.3.3.

where $\mathbf{u}_j$ denotes the jth column of $\mathbf{U}$. Note that the dimensionality of the subspace is defined by the underlying data, hence the term data-driven regularization. The template matching filter $\mathbf{f}_\psi$ operating in this reduced subspace is the solution to the following optimization problem:

$$\max_{\mathbf{f}_\psi} \frac{\mathbf{f}_\psi^T \mathbf{U}_\psi^T \boldsymbol{\tau}\boldsymbol{\tau}^T \mathbf{U}_\psi \mathbf{f}_\psi}{\mathbf{f}_\psi^T \mathbf{U}_\psi^T \mathbf{R}_{\overline{\mathbf{xx}}} \mathbf{U}_\psi \mathbf{f}_\psi}. \tag{3.10}$$

Transforming this optimal filter back to the original space results in the first subspace regularized filter:

$$\mathbf{f}_1 = \mathbf{U}_\psi \left(\mathbf{U}_\psi^T \mathbf{R}_{\overline{\mathbf{xx}}} \mathbf{U}_\psi\right)^{-1} \mathbf{U}_\psi^T \boldsymbol{\tau}. \tag{3.11}$$

3.3.3 Filter design in template & PCA subspace

The above method assumes that the template is well-approximated in the filter design subspace. For $\psi = 0.95$ it is very likely that the template is indeed well-approximated in the PCA subspace, but there is no absolute guarantee. To ensure that the template is well-approximated within the subspace we propose to use a subspace spanned by the template itself, which is then extended with orthogonal basis vectors until a ψ-fraction of the variance is captured. Consider the template data direction:

$$\mathbf{u}_{\boldsymbol{\tau}} = \boldsymbol{\tau} / \left\|\boldsymbol{\tau}\right\|_2. \tag{3.12}$$

The power that is associated with this direction is:

$$\lambda_{\boldsymbol{\tau}} = \mathbf{u}_{\boldsymbol{\tau}}^T \mathbf{R}_{\overline{\mathbf{xx}}} \mathbf{u}_{\boldsymbol{\tau}}. \tag{3.13}$$

Let $\text{Proj}_{\mathbf{u}_{\boldsymbol{\tau}}} \bar{\mathbf{x}}[k] = \mathbf{u}_{\boldsymbol{\tau}} \mathbf{u}_{\boldsymbol{\tau}}^T \bar{\mathbf{x}}[k]$ be the projection of $\bar{\mathbf{x}}[k]$ onto the direction $\mathbf{u}_{\boldsymbol{\tau}}$. We define the new data vector $\bar{\mathbf{x}}_{-\boldsymbol{\tau}}[k] = \bar{\mathbf{x}}[k] - \text{Proj}_{\mathbf{u}_{\boldsymbol{\tau}}} \bar{\mathbf{x}}[k] = \left(\mathbf{I} - \mathbf{u}_{\boldsymbol{\tau}} \mathbf{u}_{\boldsymbol{\tau}}^T\right) \bar{\mathbf{x}}[k]$ which is by definition orthogonal to $\mathbf{u}_{\boldsymbol{\tau}}$, i.e., it does not contain any variance in the template direction. The covariance matrix associated with $\bar{\mathbf{x}}_{-\boldsymbol{\tau}}[k]$ is then given by:

$$\mathbf{R}_{-\boldsymbol{\tau}} = \left(\mathbf{I} - \mathbf{u}_{\boldsymbol{\tau}} \mathbf{u}_{\boldsymbol{\tau}}^T\right) \mathbf{R}_{\overline{\mathbf{xx}}} \left(\mathbf{I} - \mathbf{u}_{\boldsymbol{\tau}} \mathbf{u}_{\boldsymbol{\tau}}^T\right)^T. \tag{3.14}$$

From the eigendecomposition (again assuming that the diagonal matrix $\boldsymbol{\Lambda}_{-\boldsymbol{\tau}}$ has all of its diagonal elements $\lambda_{-\boldsymbol{\tau}\,i}$ arranged in a descending order) of this covariance matrix:

$$\mathbf{R}_{-\boldsymbol{\tau}} = \mathbf{U}_{-\boldsymbol{\tau}} \boldsymbol{\Lambda}_{-\boldsymbol{\tau}} \mathbf{U}_{-\boldsymbol{\tau}}{}^T, \tag{3.15}$$

we can extract an orthogonal basis for the subspace that captures the most variance, while it is also guaranteed to contain the template:

$$\mathbf{T}_\psi = \begin{bmatrix} \mathbf{u}_{\boldsymbol{\tau}} & \mathbf{u}_{-\boldsymbol{\tau}\,1} \dots \mathbf{u}_{-\boldsymbol{\tau}\,j} \end{bmatrix}. \tag{3.16}$$

The subspace dimensionality of $j+1$ is then defined in a data-driven fashion by the following expression:

$$j = \max \left\{ l \;\middle|\; \frac{\lambda_{\boldsymbol{\tau}} + \sum_{i=1}^{l} \lambda_{-\boldsymbol{\tau} i}}{\operatorname{tr}(\boldsymbol{\Lambda})} < \psi \right\}. \tag{3.17}$$

The second subspace regularized filter is then given by:

$$\mathbf{f}_2 = \mathbf{T}_\psi \left(\mathbf{T}_\psi^T \mathbf{R}_{\overline{\mathbf{x}}\overline{\mathbf{x}}} \mathbf{T}_\psi \right)^{-1} \mathbf{T}_\psi^T \boldsymbol{\tau}. \tag{3.18}$$

3.4 Experiments

We validate the two proposed regularization methods on 28 ground truth neurons. These ground truth neurons are sourced from publicly available paired recordings [80] and hybrid recordings [103]. The preprocessing consists only of a high-pass filtering with a cutoff frequency $f_c = 300$ Hz (note that no explicit prewhitening is performed here). For every neuron a spatial region with size N was chosen such that all electrodes within a 100 µm radius from the electrode with the maximum template signal deflection for that neuron are included in that spatial region. The discrete temporal spike window length is set to $L = 30$, which is equivalent to a 1 ms window. From every recording the first 3 min of data are used for the analysis. The training data $\mathcal{T}$ include the ground truth spike times from the first 90 s. Validation is done on the remaining 90 s.

The filters that are used for benchmarking our two proposed regularized filter design methods are the solutions to the optimization problem given by (3.5). The free parameter was chosen to be $c = 10^{-18}$ as given by [137]. For our data this parameter was of very little use, because the spatial-only regularization was always dominant over the diagonal loading. As suggested before $\psi = 0.95$ was chosen for both our proposed regularized filter design methods given by (3.11) and (3.18), such that the low rank approximated recording does not deviate too much from the original recording.

The three different regularized filter design methods are validated through their spike sorting capability, that is obtained from thresholding their respective squared filter outputs. For every combination of neuron and regularized filter design, the spike sorting capability is summarized by the maximal F_1-score across all thresholds. This score reflects the ability of a filter to extract only the activity of the neuron of interest and is depicted in Figure 3.2. These results show an increased spike sorting performance in favour of both the proposed methods when compared to the spatial-only regularization. The subspace

regularization that includes the template seems to slightly outperform the subspace regularization without explicit use of the template.

Using a Friedman test with $\alpha = 0.05$, there is a statistically significant difference in F_1-score between the different studied methods, $X^2(2) = 24.768$, $p \ll 0.05$. Post hoc analysis with Wilcoxon signed-rank tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $p < \frac{\alpha}{m} = 0.017$, with $m = 3$ indicating the number of comparisons. There are statistically significant differences in F_1-score between the spatial-only regularization and the PCA subspace regularization ($Z = -4.0145$, $p \ll 0.017$), and between the spatial-only and the template & PCA subspace regularization (Z = -4.0145, $p \ll 0.017$). However, there is no significant difference between the PCA subspace regularization and the template & PCA subspace regularization ($Z = -2.3541$, $p = 0.018$), despite a perceived superiority of the template & PCA subspace method. However, this may be due to the overly conservative choice for Bonferroni correction, as the p-value almost reaches the significance level of $\alpha = 0.017$.

3.5 Conclusion

A data-driven regularization approach for template matching filter design in spike sorting is proposed. The approach consists of designing the actual filter in a subspace for which the dimensionality is derived from the data. Subspace regularization can be interpreted as data denoising combined with a diagonal loading. This approach is shown to significantly outperform the spatial-only regularization approach in our simplified spike sorting framework based on a single thresholding. More spike sorting regularization approaches exist in literature, but all of them are based on prior fixed parameters, e.g., a fixed diagonal loading parameter, or a fixed size subspace. The integration of our new approach in a fully-featured spike sorting framework might lead to a better spike discrimination and therefore less time spent on manual curation.

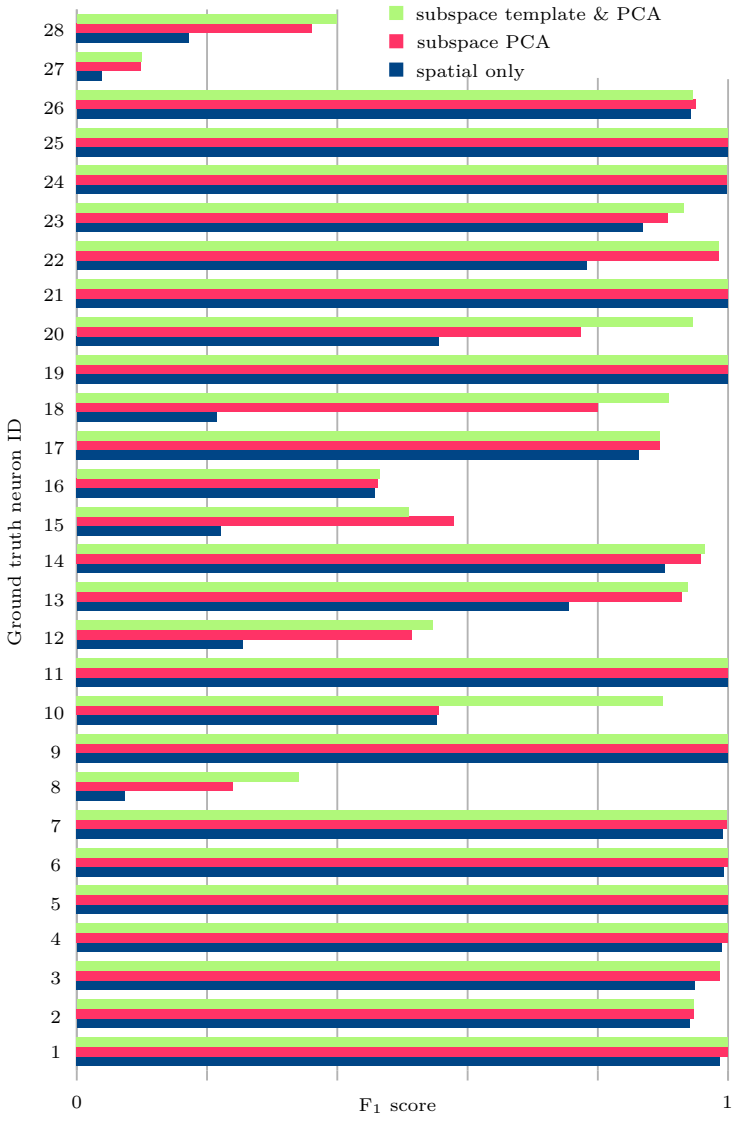

Figure 3.2: The F_1-score is shown as a measure of spike sorting performance for every ground truth neuron used. Three different regularization methods for calculating the template matching filters are compared: template & PCA subspace regularization (green), PCA subspace regularization (red), and spatial-only regularization (blue).

Chapter 4

SHYBRID

This chapter has been published as *Jasper Wouters, Fabian Kloosterman, and Alexander Bertrand. "SHYBRID: A graphical tool for generating hybrid ground-truth spiking data for evaluating spike sorting performance." Accepted for publication in Neuroinformatics (2020).* Jasper Wouters has co-developed the open-source software package, conducted the experiments and has written the manuscript. Compared to the journal publication, minor textual and notational changes have been implemented for a better integration in this thesis.

Over the last decades, many spike sorting algorithms have been published. In an effort to guide a user towards a specific spike sorting algorithm, given a specific recording setting (i.e., brain region and recording device), we provide an open-source graphical tool for the generation of hybrid ground-truth data in Python. Hybrid ground-truth data is a data-driven modelling paradigm in which spikes from a single unit are moved to a different location on the recording probe, thereby generating a virtual unit of which the spike times are known. The tool enables a user to efficiently generate hybrid ground-truth datasets and make informed decisions between spike sorting algorithms, fine-tune the algorithm parameters towards the used recording setting, or get a deeper understanding of those algorithms.

4.1 Introduction

After more than a century of brain research, and despite many breakthroughs, it is still largely unknown how brain activity gives rise to cognition. To aid researchers in getting a better understanding of brain function, many techniques have been developed to look at neuronal activity dynamics. Although recent techniques, such as calcium imaging, are getting increasingly popular, electrophysiology is still indispensable because it has the highest available temporal resolution among other available techniques. Electrophysiological recordings allow researchers to look at brain activity at a time scale that reveals neuronal action potentials in great detail. To increase the spatial extent of such electrophysiological recordings, recently recording hardware has undergone a drastic evolution leading to high-density neural probes [54] [70].

Most commonly, extracellular recordings are performed by inserting an array of electrodes into the brain. Electrodes that are in close proximity to a neuron will pick up a transient potential, referred to as a spike, when the neuron fires an action potential. Such action potentials are believed to be the main mechanism of communication between neurons. By measuring spikes from several neurons, one can try to get a better understanding of information processing in neuronal circuits. However, typically many neurons surround a single electrode, leading to spikes from several neurons to be picked up by the same electrode. On the other hand, in high-density recordings a spike from a single neuron is picked up by multiple electrodes. To resolve the mixture of spikes from several neurons and extract the spike times of the individual neurons embedded in the recordings, a spike sorting [67] [33] algorithm can be applied to the electrophysiological recording. A reliable recovery of the spike trains of individual neurons, enables a wide range of analysis [38] that ultimately advance our understanding of the brain [78] [58] [5]. Furthermore, a better understanding of the brain at the level of single neurons can create clinical impact [52] [109].

Over the last two decades a myriad of spike sorting [94] [105] [31] [103] [86] [53] [21] [137] [131] algorithms have been published. The development of these spike sorting algorithms has been driven by both the availability of novel computational methods [9] [102], as well as the ongoing evolution in the channel count and density of recording equipment [54] [70]. This has led to a broad spectrum of spike sorting algorithms, each with their own (dis)advantages, and often with only subtle differences in their underlying mechanisms. As a result, spike sorting users are often left pondering which algorithm they should use to get the spike sorting job done on their specific data.

The question which algorithm and parameters to use in a particular setting, usually boils down to the question which spike sorting package is the most

performing in terms of spike sorting accuracy and computational complexity. Unfortunately, the question related to the spike sorting accuracy is a hard one to answer, due to the lack of thorough comparative studies. In the past, initiatives for spike sorting validation have been set up [26] [16], but they have not led to a systematic validation of recent spike sorting algorithms. One of the reasons for the lack of such comparative studies is the absence of an extensive collection of ground-truth data. Two main paradigms for generating ground-truth data are in use: paired recordings [80] [3] [50] and simulation-based ground-truth data [44] [15] [68] [40] [12]. Both paradigms have their limitations as will be discussed next.

In paired recordings a single neuron is isolated (preferably in vivo) using a high signal-to-noise single cell approach, such as intracellular or juxtacellular recordings, that provides an accurate record of the targeted cell's spike times. Another recording device, e.g., a neural probe, is then lowered into the brain in close vicinity to the isolated neuron. This probe recording is analysed with a spike sorting algorithm, which can be partially validated using the ground-truth spike times obtained from the intracellularly or juxtacellularly recorded cell. The problem with paired recordings is that they involve an expensive, time consuming procedure. The procedure often fails, as it is very difficult to register the activity of the cell on both recording devices. For the paired recording to succeed, a very precise mutual positioning of both recording devices is required. Note that paired recordings have also been performed in vitro in the context of spike sorting validation [137], which could alleviate some of the experimental difficulties when compared to the in-vivo procedure.

It also remains unclear whether an algorithm's spike sorting accuracy for data with a specific recording setting is indicative for its accuracy using data recorded under a different setting. The recording setting is defined by both the brain region of interest as well as the recording equipment. Differences between brain regions, e.g., the differences in cell type distribution and cell density, will affect the extracellular recording content. Also different experimental protocols can cause changes in the local spiking activity. Differences in recording equipment, such as different probe channel layouts and channel densities, will naturally lead to a different spike feature space, i.e., the information that is used to assign detected spikes to single-unit spike clusters. The data acquisition system that is used will add electronics noise to the recording. The characteristics of such electronics noise can vary between different models of acquisition systems, or even between different acquisition systems of the same model. All of the above factors will influence the final recording, and as such will also affect the spike sorting results and performance. The recording setting intrinsics thus prevent us from generalizing spike sorting accuracy obtained on paired recordings.

As opposed to paired recordings, ground-truth data generated from computational simulations are more flexible in coping with changing recording settings. The downside of simulated data is that they often lack the richness of real recordings. To ameliorate this problem, background noise can be extracted from real recordings [116], while spikes that are generated from computational neuronal models are superimposed. Although such an approach improves the richness of the data, obtaining realistic simulations that are representative for a specific recording setting requires modelling expert knowledge. This is especially the case if realistic spike timing and scaling statistics are of interest. If realistic variations among spikes from different neurons are desired, morphologically detailed models have to be acquired, a process which is non-trivial and time-consuming. However, several large efforts have been undertaken to provide the community with a public collection of biophysically detailed neuron models [73] [96] [37].

Here, we present an open-source graphical tool which allows spike sorting users to transform their *own* recordings into ground-truth data in an efficient and controlled fashion. The ground truth generation paradigm which is used in our tool, is the so-called hybrid data paradigm that has been used for the validation of some spike sorting algorithms [103] [86]. The generation of such ground-truth data is a delicate process, which at least requires visual feedback on the generated ground truth and ideally some user interaction to make the key decisions during the generation process. Hybrid data does not rely on computational models, nor does it suffer from the complexity of obtaining paired recordings. Hybrid data generation can be seen as a data-driven modelling approach: spike templates, spike timings and scalings are directly extracted from the data. As such, the ground-truth data that is generated is automatically tailored to the user's recording setting. This ground-truth data can then be used to make an informed decision on which spike sorting algorithm is most suitable for a specific recording setting, fine-tune the algorithm parameters towards the used recording setting, or get a deeper understanding of those algorithms.

The tool takes as an input the raw data and initially curated spike sorting results, e.g., obtained from manual cluster cutting. As a first step a template is calculated for every single-unit spike train. Next, the user visually inspects the template fit for a selection of spikes in the corresponding spike train. If the scaled template model is deemed reasonable, the user moves the entire unit, i.e., the combination of the spike train and its corresponding template, to a different spatial location on the probe. The final step of relocating the unit is key for the relocated unit to be considered as a valid ground-truth unit. For the original unit, one can not make this ground-truth assumption. Indeed, in the initial spike sorting results, one can easily classify a spike cluster as originating from a single neuron, but it is a lot harder to show that all spikes from the specific

neuron are present in the cluster. Such missed detections (false negatives) in the initial sorting might be erroneously interpreted as false positives when the original recording and initial spike sorting are used for the validation of other (and potentially better) spike sorters. By changing the spatial location of the hybrid unit with respect to the original unit, this problem is bypassed because the spatial footprint of the hybrid unit is different now from the spatial footprint of the original unit, and as such potential false negatives in the initial sorting are very unlikely to interfere with the hybrid unit in terms of spike sorting.

A main advantage of the presented tool is the visual feedback that is provided throughout the hybrid data generation. The visual feedback allows a user to closely monitor and control the hybrid data quality. For example, visual feedback of the spike rate of every individual channel allows a user to control the spike sorting difficulty of the hybrid unit. Inserting a hybrid unit into a silent spatial region of the probe will likely be easier to sort than inserting a hybrid unit in a very active spatial region. The tool is also capable of fully automatic hybrid ground-truth data generation for use cases where less control is acceptable. An efficient method is included to validate spike sorting results obtained on the generated hybrid ground-truth data. The tool is easy to use and freely available[1], which will enable a broad audience to generate recording setting-specific ground-truth data for improving their spike sorting related research.

In this work we will focus on generating hybrid data from high-density neural probe recordings. However, the tool is by no means limited to neural probes, it can be used without any modification on various other multi-channel recording devices, e.g., microelectrode arrays (MEAs) [75] [74]. This work as a whole might also be useful outside of the field of neuronal spike sorting. The validation of sEMG [76] decomposition algorithms [47] [34], i.e., algorithms for sorting motor unit action potentials that are measured using electrode arrays placed on the skin on top of muscle tissue, might also benefit from the availability of hybrid ground-truth data.

The hybrid ground-truth model is discussed in depth in the following section. In Section 4.3 a graphical tool is introduced that implements this hybrid model. Section 4.4 contains a case study where ground-truth data generated with the user interface is used to compare and study spike sorting algorithms. Finally, the proposed method and results are discussed in Section 4.5.

[1]The tool is available on `https://github.com/jwouters91/shybrid`.

4.2 Hybrid ground-truth model

Before introducing the graphical tool for spiking hybrid ground-truth data generation, we will first introduce and formalize our hybrid ground-truth model. The basic idea behind the generation of hybrid data [103] [86] is to remove single-unit spikes from one location of the probe and re-introduce them at another location with an additional temporal offset. We thus create a fictitious neuron of which the spike waveforms are modelled as scaled spike templates estimated from the initial spike sorting results. This unit is then injected at another location on the probe to prevent false negatives (i.e. spikes from the original neuron that were not identified) in the initial spike sorting results from being wrongly interpreted as false positives when later using the hybrid data for evaluating spike sorting performance. This spatial migration is sufficient for decoupling the fictitious neuron from its donor neuron for spike sorting purposes. This approach depends on the availability of initial spike sorting results for the given recording, containing manually verified single-unit clusters. In this way ground-truth data, which has both realistic spike and noise related statistics, can be made without modelling effort.

The input high-pass filtered extracellular recording that will serve as a basis for the hybrid ground-truth data is denoted by the matrix $\mathbf{X} \in \mathbb{R}^{T \times N}$, with T the duration in samples of the recording and N the number of recording channels. The vector $\mathbf{x}^c \in \mathbb{R}^T$ contains the elements from the cth column of $\mathbf{X}$, i.e., it contains the data recorded on channel c. A single sample from channel c at discrete time k is represented by $x^c[k]$. Let's consider a single channel signal slice $\bar{\mathbf{x}}^c[k] = [x^c[k-K] \ \ldots \ x^c[k+K]]^T \in \mathbb{R}^L$ centered around k, where $L = 2K + 1$ denotes the slice length, i.e., the so-called temporal window size in samples.

Initial spike sorting results are required for the estimation of the spike templates. The initial spike sorting results consist of a set of spike times $\mathcal{T}_i$ for every known single-unit cluster i. Consider the spike snippets matrix $\mathbf{S}_i^c$ which contains in its columns all spike snippets $\bar{\mathbf{x}}^c[s] \ \forall s \in \mathcal{T}_i$. The spike template $\boldsymbol{\tau}_i^c$ of a single-unit cluster i on channel c can then be estimated as follows:

$$\boldsymbol{\tau}_i^c = \text{med } \mathbf{S}_i^c, \tag{4.1}$$

where the median operator acts on each row of the spike snippets matrix, such that $\boldsymbol{\tau}_i^c \in \mathbb{R}^L$.

Next, for every channel c, the template energy $E^c = \boldsymbol{\tau}_i^{cT} \boldsymbol{\tau}_i^c$ is calculated and the maximum energy $E^{\max} = \max_c E^c$ over all channels is determined. The channels in the template that contain relatively little energy, are considered to

be noise channels and are zero-forced based on the following maximum energy criterion:

$$\forall c \mid E^c < \alpha E^{\max} \ : \ \boldsymbol{\tau}_i^c \leftarrow \mathbf{0}, \tag{4.2}$$

where α is the so-called zero force fraction (e.g., $\alpha = 0.03$). This zero-forcing results in a more localized spike template.

Next, the optimal template scaling for every spike snippet $s \in \mathcal{T}_i$ used during the template estimation is calculated. This template scaling will be used both when subtracting and inserting units during the hybridization process. The optimal scaling is given by the least squares fitting factor β_i^s and minimizes the following least squares optimization problem:

$$\min_{\beta_i^s} \sum_c \left\| \bar{\mathbf{x}}^c [s] - \beta_i^s \boldsymbol{\tau}_i^c \right\|_2^2 . \tag{4.3}$$

The solution to (4.3) is given by the following expression:

$$\beta_i^s = \frac{\sum_c \bar{\mathbf{x}}^c [s]^T \boldsymbol{\tau}_i^c}{\sum_c \boldsymbol{\tau}_i^{cT} \boldsymbol{\tau}_i^c}. \tag{4.4}$$

Before inserting the hybrid spikes into the recording, for every spike time used during the template estimation, the fitted template is subtracted from the data. Although this step is not strictly necessary, this subtraction will prevent the total energy in the final hybrid recording from increasing. The subtracted signal $\mathbf{s}^c [k]$ is given by:

$$\begin{aligned} \mathbf{s}^c [k] = \mathbf{x}^c [k] - \\ \sum_i \sum_{s \in \mathcal{T}_i} \sum_{m=-K}^{K} \delta [k - (s+m)] \, \beta_i^s \boldsymbol{\tau}_i^c [m] , \end{aligned} \tag{4.5}$$

where $\delta [k]$ is the Kronecker delta, i.e., $\delta [k] = 1$ if $k = 0$ and $\delta [k] = 0$ if $k \neq 0$.

The next step in the process of generating the hybrid ground truth is to change the spatial location of the spike template. In order to facilitate this spatial migration, some assumptions about the probe geometry have to be made. For every probe a rectangular grid model is assumed (non-rectangular probes are supported, but will be internally treated as rectangular grids, see below). The grid model requires that all recording electrodes for a certain probe lie on the intersection of an imaginary rectangular grid, as depicted in Figure 4.1**A**. However, not every intersection point has to be occupied by an electrode,

thereby allowing to model probes with broken or missing electrodes, or probes with a non-rectangular electrode grid. Furthermore, the model assumes that the horizontal distance between the intersection points is constant. Also the vertical distance between intersection points has to be constant, but this vertical distance can differ from the horizontal constant distance.

From the above assumptions a probe graph model can be built. For every empty location on the grid a missing electrode is added to the graph model. In such a graph model every electrode is aware of its neighbouring (missing) electrodes, such that the template can be easily shifted in every direction on the probe.

A template shift is characterized by a number of moves along the horizontal axis in a certain direction and a number of moves along the vertical axis in a certain direction. For the horizontal axis moving the template to the right is represented by a positive number and moving to the left by a negative number. Moving up is represented by a positive number of moves, while moving down has a negative sign. The shifted template $\boldsymbol{\tau}_i^{c,(x,y)}$ is obtained from shifting horizontally by x and shifting vertically by y. Figure 4.1**B** gives a schematical representation of $\boldsymbol{\tau}_i^{c,(-1,0)}$. The black arrows in the figure indicate the spatial pseudo-permutation, note how extrapolated waveforms enter the shifted template (here by template we mean the set $\{\boldsymbol{\tau}_i^c \; \forall \; c\}$) on the right side of the probe and how template information is dropped on the left side of the probe. Template extrapolation is performed to gracefully handle spikes at the edge of the probe, that might otherwise cause edge effect artefacts in case zero-padding is used rather than extrapolation. The extrapolated waveforms result from averaging over the neighbouring working electrodes (see blue discs in Figure 4.1**B**) and an equal amount of fictitious electrodes containing all zeros to guarantee a major attenuation of the extrapolated waveforms when compared to their working neighbours. Missing channels that are shifted onto working electrodes contain a value obtained through interpolation (see orange discs in Figure 4.1**B**). The interpolated value equals the average over all working neighbouring channels, as is schematically represented in Figure 4.1**B** by the orange arrows.

Typically, the set of scaling parameters β_i^s contains various outliers, e.g., caused by fitting the template on a segment that is deformed by the presence of overlapping spiking activity from other neurons. To prevent outliers in β_i^s from propagating into the generated ground-truth data, only a subset of $\mathcal{T}_i$ should be considered for reinsertion. This subset is derived by choosing a lower bound l_i^β and upper bound u_i^β for the template scaling factor β_i^s and setting $\beta_i^s = 0$ for all the spike times $s \in \mathcal{T}_i$ for which $\beta_i^s \notin \left[l_i^\beta, u_i^\beta\right]$, thereby effectively discarding those spikes from $\mathcal{T}_i$ for insertion.

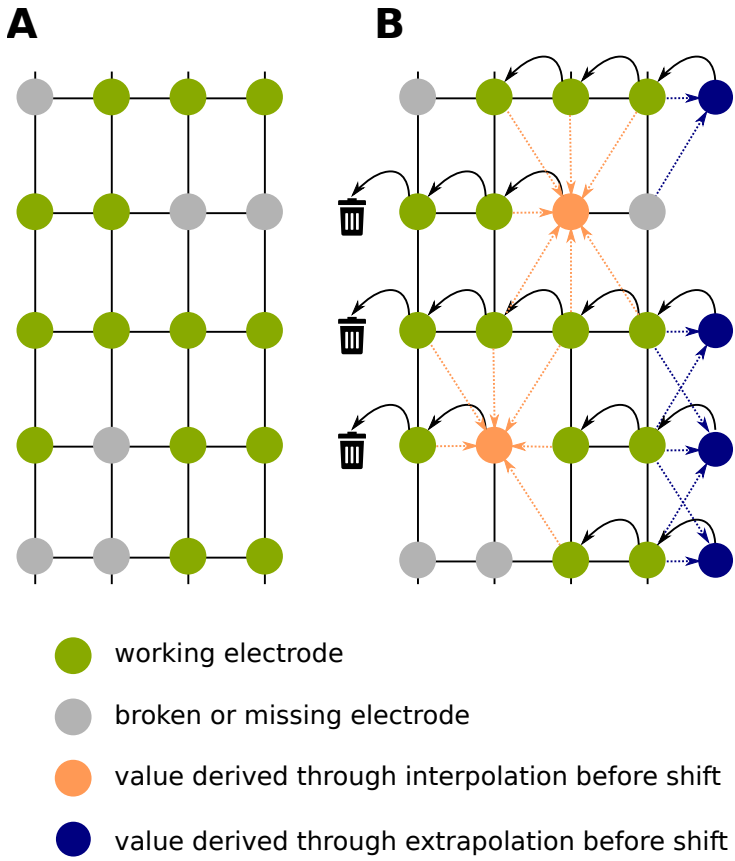

Figure 4.1: **A**: The probe grid model assumes that all electrodes (green dots) are located on intersections of a grid. The distance between grid lines along a certain axis is assumed to be constant. Not every intersection has to be populated by an electrode (grey dots). **B**: Moving the template over the probe consist of shifting the template over the grid. The black arrows represent shifting a template one position to the left. At the right side of the probe extrapolated waveforms (blue dots and arrows) are introduced into the template, while on the left side, a part of the template is lost. Missing channels that are shifted onto working electrodes contain a value obtained through interpolation (orange dots and arrows).

Prior to the insertion of the ground truth spikes in the recording, a temporal within-sample-time jittering is applied to the template for every $s \in \mathcal{T}_i$. This jittering is obtained by first upsampling the template by a factor of 10, then applying a random within-sample-time shift sampled from a uniform distribution to the upsampled template, and finally, downsampling the template to the original sampling frequency. This temporal jittering models the fact that the occurrence of spikes are not phase locked with the sample clock of the analog-to-digital converter. The jitter operator is represented by the non-linear function $j^s : \mathbb{R}^L \mapsto \mathbb{R}^L$, where the dependency on s is required to guarantee that for a certain spike time s all samples of the template are jittered by the same amount and because the random time shift is different for each $s \in \mathcal{T}_i$. Finally, the hybrid data can be mathematically expressed as follows:

$$\mathbf{h}^c[k] = \mathbf{s}^c[k] + \sum_i \sigma_i \sum_{s \in \mathcal{T}_i} \sum_{m=-K}^{K} \delta[k - (s + T_{\text{offset}} + m)]\, \beta_i^s j^s \left(\boldsymbol{\tau}_i^{c,(x,y)}[m] \right), \tag{4.6}$$

where we introduced an additional time offset T_{offset} to all remaining spike times before re-inserting them at the new location in order to decorrelate them with possible spike residues at the original location. σ_i is an optional user-defined scaling parameter to control the signal-to-noise-ratio (SNR) of the resulting hybrid unit associated with the neuron i. The default value is set to $\sigma_i = 1$.

4.3 GUI for hybrid data generation

The generation of ground-truth data is a delicate process, which at least requires visual feedback on the generated ground truth and ideally some user interaction to make the key decisions during the generation process. For this reason we developed a Python-based graphical user interface (GUI) called SHYBRID (spike hybridizer) to support spike sorting users and developers with the generation of ground-truth data, so that they gain more insight into how spike sorting algorithms behave on certain data. This section elaborates on the typical usage flow of the tool. The following topics will be discussed in the different subsections: which input data to provide, how to visualize and adapt the template, why and how to choose bounds on template scaling, and what to consider when moving a unit across a probe. We will also touch on functionality to import and export templates for additional flexibility. Finally, we will discuss the tool's functionality for the automatic assessment of spike sorting algorithms and its integration with a recent spike sorting framework. This section only

gives an overview of the above concepts, more detailed usage-related information can be found in the user manual that is provided with the tool[2].

4.3.1 Input data

The user interface depends on four different inputs:

1. an extracellular multi-channel recording in binary format
2. the recording's probe file
3. initial spike sorting results
4. a parameter file

The provided *binary recording* is assumed to be high-pass filtered. This high-pass filtering is standard practice in spike detection and spike sorting pipelines, as it is needed to attenuate the power dominant low frequency oscillations present in raw neural recordings. Without proper filtering, it would be difficult to reliably extract spike templates from the recording, which are the key building blocks for hybrid data generation. Only binary recordings that encode the signal samples using a signed data type are supported.

The application's centerpiece is an intuitive data visualization widget. Signal snippets are plotted as if they are drawn on the probe (see Figure 4.2). This visualization requires information about the geometry of the probe as described in the *probe file* [104] in a structured text format. This file format is used by some spike sorting algorithms already, so there are many probe files available for various neural recording devices. The probe file can also be used to inform the tool about broken/bad channels that are present in the recording, and are to be ignored during the hybridization process. Note that the tool is only compatible with single shank probe files. Some probe files also provide a probe graph. SHYBRID will ignore any graph present in the probe file and will build its own. This graph is built from the geometry related grid assumptions as introduced in Section 4.2.

The initial *spike sorting results* are a collection of manually curated single-unit clusters. A cluster consists of a unique integer identifier and the cluster's corresponding spike times (given in samples). The initial spike sorting results can be loaded using either a custom *CSV*-file or by supplying a path to the folder containing the *phy* compatible data [104] in the *template-gui* format[3]

[2]SHYBRID documentation: `https://shybrid.readthedocs.io/`

[3]Please consult the phy documentation for more information about the template-gui format.

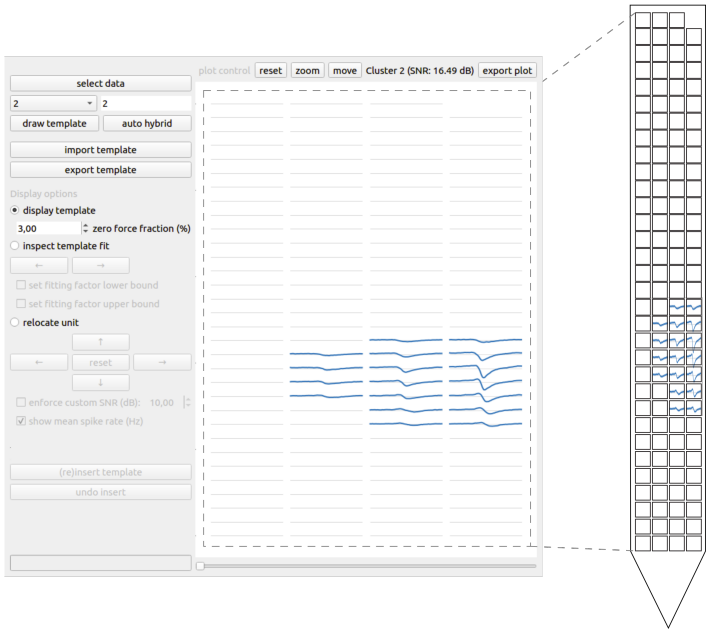

Figure 4.2: Spike templates (and recording snippets) are visually organized according to the channel geometry of the recording device. The different rows and columns in the visualization correspond to the rows and columns of the recording device. The waveform duration (horizontal axes) is controlled by the user. The waveform amplitude (vertical axes) is automatically scaled to provide a clean visualization. The relative waveform amplitude with respect to the recording noise is available from the SNR that is calculated and shown for every spike template.

(i.e., a de facto standard for spike sorting results). The CSV-file has to consist of two columns, where the first column contains the cluster identifier and the second column contains a corresponding discrete spike time. From a CSV-file all clusters are loaded into the application. When using the phy compatible initial spike sorting results, only clusters that are labeled as “good” will be loaded.

Finally, the *parameter file* links all of the above files together as illustrated in Listing 4.1. On top of that, it contains additional recording-related parameters that are needed by the application, such as the recording’s sampling frequency and data type. The parameter file has to have the same file name as the binary recording file, and requires a *.yml* extension, indicating that its content is structured using the *YAML* format.

```
---
# parameters used by SHYBRID
# recording data related
data:
  # sampling frequency
  fs: 25000
  # recording data type
  dtype: int16
  # recording matrix to linear storage
  # conversion order:
  # row-major (C) or colum-major (F)
  order: C
  # path to probe file
  probe: /path/to/probe.prb

# initial spike sorting results
clusters:
  # path to initial sorting csv
  csv: /path/to/clusters.csv
...
```

Listing 4.1: An example parameter file in the YAML format.

4.3.2 Visualizing the template

After loading the input data, a spike template can be drawn for a single-unit spike cluster of choice. Details on how the template is estimated can be found in Section 4.2. Two important parameters that affect the template estimate can be altered in the tool as shown in Figure 4.3.

The first parameter that has to be explicitly chosen is the temporal window length. Changing this window length will change the number of consecutive samples on every channel around a single spike occurrence as used for template estimation and visualization. For generating hybrid data, this window length should be sufficiently large. Within the temporal window, the signal on every channel of the template should return to zero on both sides, this is to ensure that the complete spike template is caught in the given window. Choosing an appropriate window size will limit residual artefacts that occur during the process of hybrid data generation.

The second template related parameter that can be altered is the zero force fraction. Usually a spike template as seen on a probe is spatially sparse, i.e., some channels of the template are dominated by spiking activity, whereas others are dominated by noise. To eliminate any effect of these noise channels further down the pipeline, the tool tries to zero-force the noise channels. Typically these noisy channels are considerably lower in terms of signal energy. The software

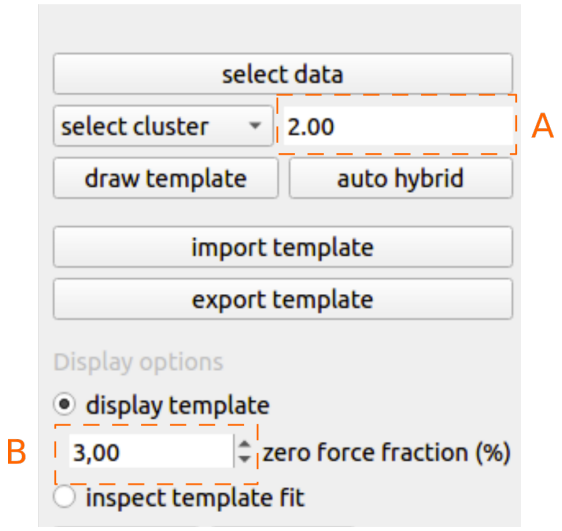

Figure 4.3: **A**: Template temporal window length given in milliseconds. **B**: Altering the zero force fraction changes the spatial extent of the template.

determines the signal energy for every channel of the template separately. From these energies the maximum energy is then determined. All channels that contain less energy than the given zero force fraction of the maximum template channel energy, are explicitly set to zero. As such the spatial extent of the template is determined from the data, instead of relying on an a-priori chosen spatial extent. The zero force fraction is set to 3% by default. Depending on the SNR of the recording, this fraction might have to be adjusted to reflect the true spatial footprint of the spike template. Increasing the zero force fraction (e.g., in case of a recording with low SNR) will eventually lead to more channels being set to zero.

After the template estimation process, the template is visualized as shown in Figure 4.2. Only non-zero forced channels are emphasized (in blue) in the visualization.

4.3.3 Inspecting template fit

A second step in the generation of hybrid data is to inspect how well a template fits the underlying data from which it was estimated. This is important to verify whether the scaled template model that is used for the generation of hybrid data is valid for the given cluster. For every data chunk that was used for the

estimation of the template, an optimal template scaling factor is determined that minimizes the squared error between the data chunk and the scaled template (see Section 4.2).

In the *inspect template fit* display mode, the scaled template (blue) is plotted on top of a signal chunk (red) containing a spike that was used during the template estimation process (see Figure 4.4). By using the left and right arrow buttons (or the bottom slider) the user can browse through all the data chunks that were used for the estimation of the template. In this way the user can assess whether the scaled template sufficiently models the underlying data. To avoid that the user has to go exhaustively through all the chunks, the data chunk browsing order is determined by the fitting factor β_i^s, where moving the slider to the right corresponds to an increasing fitting factor.

The reason that we opted for this ordering is that often, both very low and high fitting factors are likely to be the reflection of a spurious template fit. Such a spurious template fit is typically caused by either false positives in the provided spike sorting results or are due to spatio-temporally overlapping spikes. Because of the ordering that is in use here, bounds can be easily identified and set on the fitting factors. The red profile at the bottom of the plotting area (see Figure 4.4) is a visual representation of the ordered fitting factors for all data chunks. A user is adviced to start from the lowest scaling factor and move upwards until a nicely fitting signal chunk is identified. This chunk can then be set as the lower bound. After choosing a lower bound, the user can start from the maximum fitting factor and repeat the previous procedure in a downwards fashion. Only the spikes within the optional bounds will be considered during the hybridization process. This feature allows the user to have precise control over the scaling factor range, preventing unrealistic scaling factors from entering the hybrid data.

Because overlapping spikes will often cause unrealistic fitting factors, one might think that applying bounds on the fitting factors might exclude those interesting overlapping spike times from the hybridization process. However, since the spike template is spatially moved before the hybrid spikes are reinserted, it is very likely that new spatio-temporal overlap arises in the region to which the template is shifted.

4.3.4 Relocating the unit

Once the validity of the scaled template model for the active cluster has been assessed, the unit has to be relocated on the probe. This action is fundamental for the generation of hybrid data, as mentioned before. Moving the unit along the probe also allows for the reintroduction of overlapping spikes, i.e., by moving

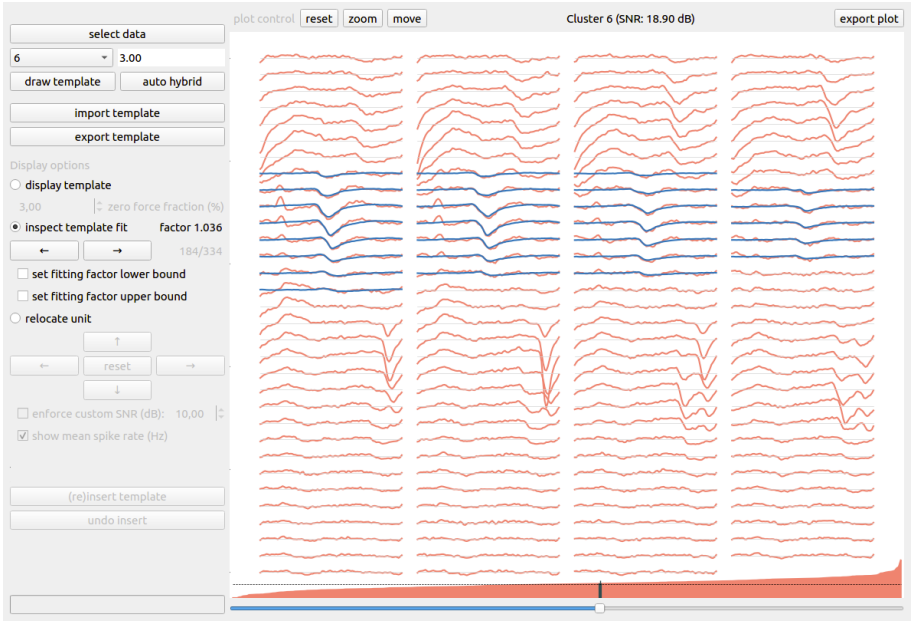

Figure 4.4: For every recording chunk (shown in red) that is used for the template estimation, an optimal template scaling factor is determined. The *inspect template fit* view allows the user to assess how good of a model the scaled template (shown in blue) is for the current cluster. The organization of the visualization is identical to Figure 4.2. The red profile at the bottom of the plotting area is a visual representation of the ordered scaling factors for all recording chunks. The fine dashed line (black) indicates the unit fitting factor, i.e., $\beta_i^s = 1$.

the fictitious neuron to a location on the probe where there is a high spiking activity. To aid the user in choosing where to move the unit to, the average spike rate on every electrode is calculated and represented visually in the *relocate unit* view. By moving the unit to a *busy* region of the probe, it is more likely that spike overlap occurs for the hybrid neuron. If a unit is moved to a *silent* region, the hybrid spikes are more likely to be easily separable. This allows a user to control the difficulty level for the spike sorting algorithm under test. In our illustrative example, the unit that is shown in Figure 4.4 is moved to a region with lower average spiking activity as shown in Figure 4.5. The user can also control the difficulty level by enforcing a custom peak-SNR for the resulting hybrid unit. The peak-SNR is defined in Section 4.6.3. By default, the peak-SNR is displayed corresponding to the default value $\sigma_i = 1$ in (4.6),

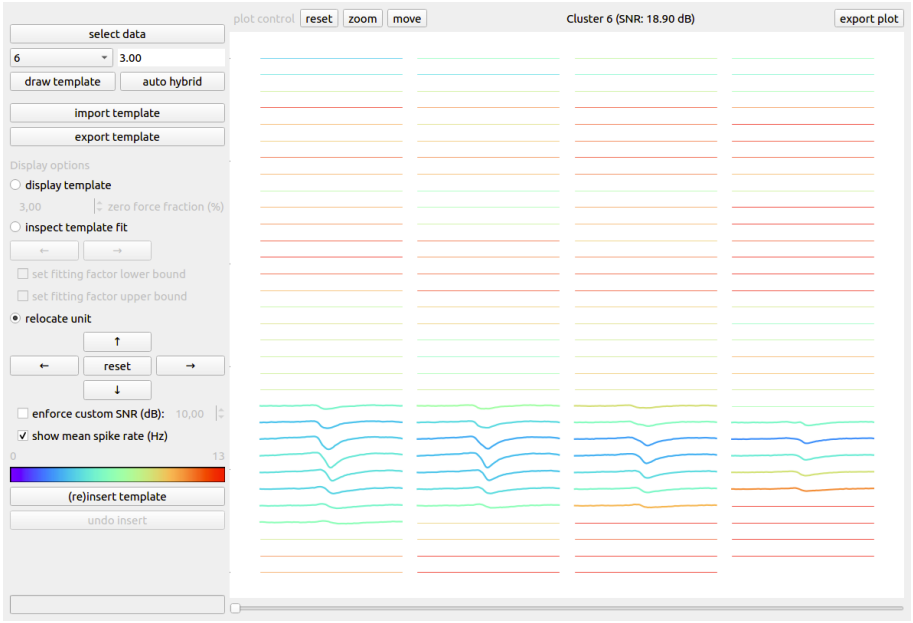

Figure 4.5: A unit's spike template can be moved over the probe by using the arrow controls on the left panel. The average spike rate on every channel is indicated by the color in which the channel is plotted. Moving a template into a silent region will likely result in a hybrid recording that is easier to sort. The organization of the visualization is identical to Figure 4.2.

which means the original amplitude of the spike template is kept.

After a unit is moved from its original spatial location, it has to be reinserted into its new location. A unit reinsertion first subtracts the original fitted templates from the recording after which the spikes within the template scaling factor bounds are reinserted at the new location. The spike times at which the scaled shifted templates are inserted are equal to the original spike times offset by an arbitrary constant that equals two times the temporal window length. As discussed in Section 4.2, this offset is added to prevent the residual artefact after subtraction from interfering with the hybrid cluster template. This reinsertion will also create a CSV-file that keeps track of the hybrid ground-truth spike times. Note that the reinsertion is immediately applied to the provided recording and overwrites the binary data file. A reinsertion can always be undone if the resulting hybrid unit is not satisfactory, e.g., when the provided custom SNR is deemed unrealistic after inspecting the hybrid spike chunks.

4.3.5 Auto hybridization

For long recordings containing dozens of manually curated single-unit clusters, the full user guided hybridization process is a lengthy procedure. Although we believe it is advisable to have a user making the key decisions, we also provide an auto-hybridization function which loops over all provided single-unit clusters. For every cluster, conservative bounds on the template fitting factor are chosen and the corresponding unit is moved to another spatial location on the probe that is randomly chosen. All of this happens automatically, at the cost of reduced control on the resulting hybrid data. The details on the automatic bounds selection and template movement can be found in Section 4.6.1 and 4.6.2, respectively.

4.3.6 Importing / exporting templates

To provide increased flexibility it is also possible to import external spike templates into a recording. This feature allows for the creation of ground-truth data without the need for a prior spike sorting. The external templates can be either hand crafted or exported from other recordings that do have some prior spike sorting information that allows for the estimation of spike templates. Details about the importing procedure can be found in Section 4.6.3.

4.3.7 Automatic tools for spike sorting algorithm assessment

Since the main objective of this work is to get a better insight into spike sorting performance, the tool comes with methods for comparing the hybrid data spike sorting results to the hybrid ground-truth spike times. This functionality consists of two important methods, that will be discussed in more detail starting from the next paragraph. First, the functionality to perform automatic cluster merging on the spike sorting results is discussed. Second, the spike sorting performance metrics routine is discussed.

Automatic cluster merging

Ideally, a spike sorting algorithm outputs a single cluster for each single-unit spike train present in a recording. In practice however, spikes from the same neuron are often split over multiple clusters (referred to as overclustering) or a single cluster contains spikes from multiple neurons. Experimentalists usually prefer overclustering, because the manual curation which then consist of cluster

merging is more straightforward than the process of splitting clusters into several single-unit clusters. Therefore, modern spike sorting algorithms are typically tuned towards overclustering.

If an algorithm under investigation is known to have a preference for overclustering, it is necessary to perform a merging step prior to calculating the spike sorting performance metrics. For this purpose an automatic ground truth assisted merging framework is implemented to bypass this manual merging step in the assessment of a spike sorting algorithm. Since we do not provide an automatic splitting framework, algorithms that are tuned towards overclustering are favoured in terms of final spike sorting accuracy. We believe this limitation is acceptable, as it reflects the user preference towards overclustering. The algorithmic details of the merging framework are given in Section 4.6.4.

Performance metrics calculation

A fast implementation is provided for the calculation of spike sorting related performance metrics given the hybrid ground-truth data. The provided method is capable of generating tables such as Table 4.1 and Table 4.2. Prior to the calculation of the metrics, for every hybrid spike train an automatic cluster merging step is performed as explained above. After this merging step, the following metrics are calculated for every ground-truth spike train:

- The *recall* or sensitivity, which is defined as $\frac{TP}{P}$, with TP the number of true positives, i.e., the number of spikes that are correctly associated with the ground-truth spike train and P the number of positives, i.e., the total number of spikes in the ground-truth spike train.
- The *precision*, which is defined as $\frac{TP}{TP+FP}$, with FP the false positives, i.e, spikes that are wrongly associated with the ground-truth spike train.
- The F_1-score is also calculated, which is the harmonic average of the recall and precision: $F_1 = 2\frac{\text{precision.recall}}{\text{precision+recall}}$.
- The number of clusters that were automatically merged, prior to calculating the above performance metrics.

The implementation is based on Python set structures. A spike train is modelled as a set containing the spike times at which the underlying neuron is active. A Python set is essentially a hash table. A hash table allows to check whether an element is present in the structure in $O(1)$, i.e., the time it takes is independent of the number of spike times that are present in the set. This particular

implementation based on hash tables allows for a very fast calculation of the above metrics. To cope with spike time misalignment between the ground truth and the actual spike sorting results, every spike time in the set structure is extended by a user-defined window.

4.3.8 SpikeInterface integration

SpikeInterface [13] is an open-source software stack, that was designed to promote the interoperability between different neural recording systems and spike sorting software. This goal is reached by implementing a unified access model for both neural recordings and spike sorting. SpikeInterface also provides functionality for ground truth validation and curation that can be applied to their unified data access model. In order to further improve the user experience of SHYBRID, a SHYBRID-SpikeInterface integration is provided. This integration has the following advantages:

- Extend the native file format compatibility of SHYBRID with the file formats that are supported in the SpikeInterface ecosystem. This is both applicable to the neural recordings, as well as the initial spike sorting format.
- SHYBRID hybrid data can be easily sorted using the various spike sorters that are supported in the SpikeInterface ecosystem.
- Besides the SHYBRID validation routine, one can also analyse the hybrid data spike sorting results and perform a ground truth validation using SpikeInterface.

4.4 Case study: comparing spike sorting algorithms and tuning parameters

In this section the performance of two spike sorting algorithms on a specific recording is analysed. The two spike sorting algorithms considered here are Spyking Circus (SC) [137] and KiloSort (KS) [86]. The donor recording used here is part of a paired recordings dataset (2015_09_03_Cell.9.0) [80], which is commonly used for validating spike sorting algorithms [21] [137] [131].

The donor recording is given to both spike sorting algorithms prior to the hybridization process. We generate two hybrid data sets from this donor

recording, where one is informed by the SC spike sorting (with default parameter settings) results and the other by the KS spike sorting (with default parameter settings, where *number of clusters* = 256) results. This will allow us later on to identify a potential bias towards the algorithm from which the spike sorting results are used during the hybridization process. The spike sorting results from both algorithms are manually curated using the phy template GUI [104]. This curation process consists of a manual cluster merging and assessing whether or not the cluster consist of single-unit activity. Two hybrid data sets are then generated, following the steps described in Section 4.3, from the single-unit spike sorting results obtained during the manual curation of both algorithms. The total number of injected hybrid ground-truth units is 27 for the SC-informed hybrid data and 15 for the KS-informed hybrid data. There are no overlapping spike trains between the SC-informed and KS-informed data. Figure 4.6 shows the spike templates of four hybrid units. This figure contains templates from both SC-informed and KS-informed hybrid data. Visual comparison shows that units that are easily recovered (see next paragraph for the definition) during the final spike sorting, are likely to have a higher SNR compared to units that were not recovered during this spike sorting.

After hybridization, both hybrid data sets are again sorted using both algorithms (again with default parameter settings). Because this time the spike sorting is performed on ground-truth data, a ground truth assisted automatic merging can be performed after the spike sorting, as explained in Section 4.3.7. These automatically merged spike sorting results are then compared to the ground-truth labels. The automatic merging and ground truth comparison result in Table 4.1 and 4.2. The rows in those tables that have a bold typesetting indicate ground-truth units that have been properly recovered, i.e., the F_1-score for those units is greater than 0.9. Although this definition is somewhat arbitrary, it allows us to focus on the spike sorting performance metrics for units that have been successfully recovered.

The spike sorting performance metrics for the recovered units are also summarized in Figure 4.7. Figure 4.7**A** shows that for the SC-informed hybrid data SC recovers 18/27 units (dark blue bar) and KS recovers 10/27 units (red bar). One can also see that for the KS-informed hybrid data SC recovers 7/15 units and KS recovers 5/15 units. For both the SC and KS-informed hybrid data, the average number of automatic cluster merges is higher for SC than for KS (see Figure 4.7**B**), and the average F_1-score is higher for KS (red bar) than for SC (dark blue bar) as can be seen from Figure 4.7**C** and 4.7**D**.

SC recovers 66.7% of the hybrid units for the SC-informed data and 46.7% of hybrid units for the KS-informed data. The average F_1-score for SC on the SC-informed data equals 97.0%, and 95.3% on the KS-informed data. This information might indicate the existence of a bias towards the initial spike

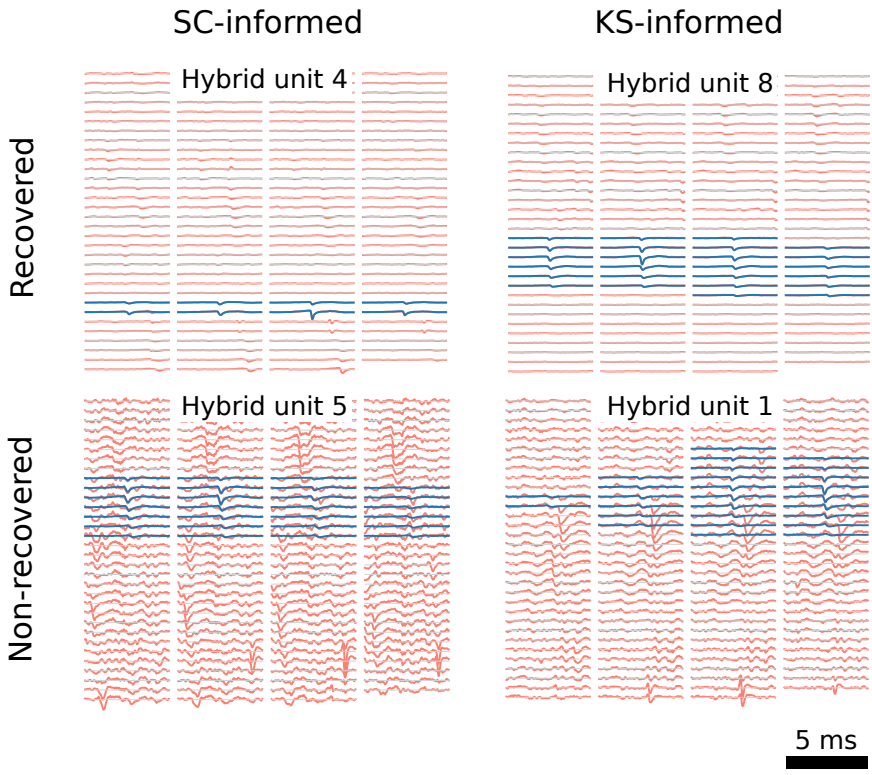

Figure 4.6: Example templates (blue) of hybrid units for both the SC-informed and KS-informed hybrid data are shown. The two templates in the top row originate from hybrid units that are recovered (i.e., F_1-score > 0.9) by both spike sorting algorithms (see Table 4.1 and 4.2 for numerical details). The bottom row contains two templates from hybrid units that were only partially recovered by both algorithms. An example optimally scaled spike signal chunk (red) is shown behind each template to allow for a visual assessment of the relative noise levels with respect to the template power. The individual unit template plots are generated through the SHYBRID *export plot* functionality.

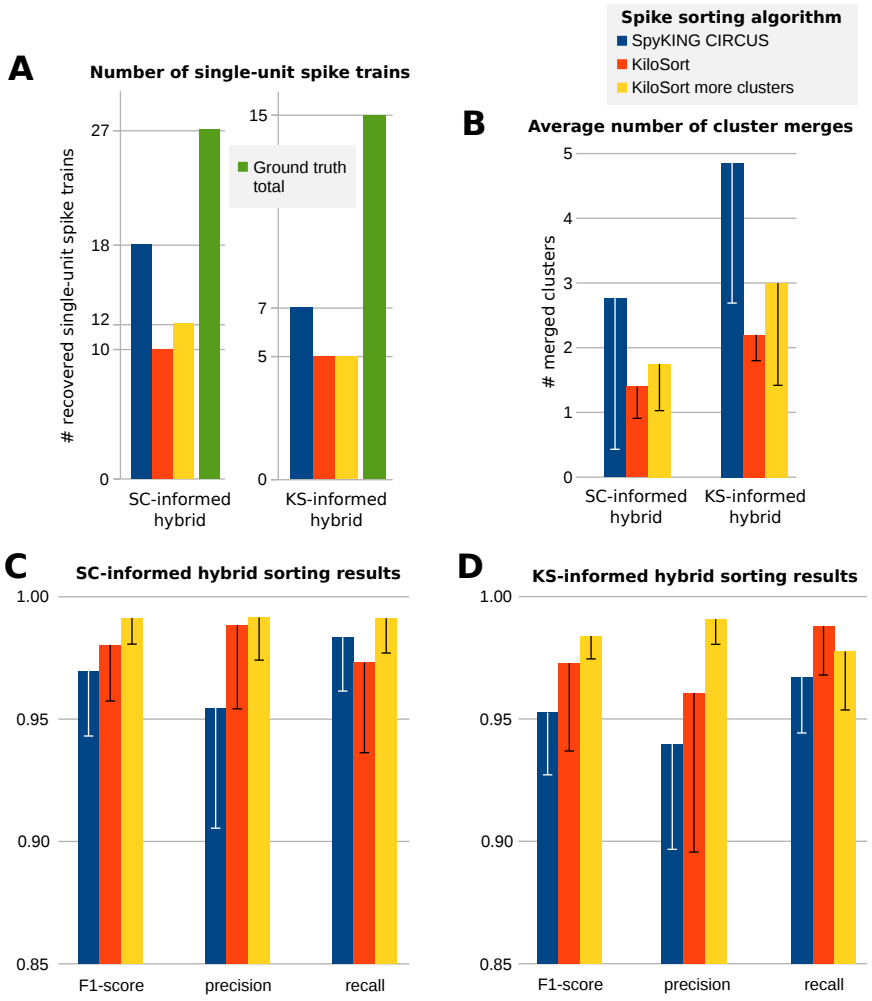

Figure 4.7: **A**: For both the SC- and KS-informed hybrid ground-truth data, the number of hybrid single-unit spike trains that are recovered by the different spike sorting algorithms is shown. The total number of hybrid ground-truth spike trains, i.e, the maximum number that could have been recovered, is shown in green. **B**: For both the SC-and KS-informed hybrid ground-truth data, the average number of cluster merges over the single-unit spike trains is shown for the different spike sorting algorithms that are used. The error bars indicate the standard deviation. **C** and **D**: Spike sorting performance metrics for the SC- and KS-informed hybrid data, respectively. As performance metrics the F_1-score, precision and recall are shown. The error bars indicate the standard deviation across the recovered single-unit spike trains.

sorter, when SC is used for that purpose. On the other hand, KS recovers 37.0% of the hybrid units for the SC-informed data and 33.3% for the KS-informed data. The average F_1-score for KS on the SC-informed data equals 98.0%, and 97.2% on the KS-informed data. From this information we cannot conclude the existence of a bias towards the initial spike sorter, when KS is used for this purpose. Apart from the potential existence of a bias, the above results might also be partially explained by differences in spike sorting difficulty between the two different hybrid data sets.

From these first observations we can conclude that SC returns more single-unit clusters compared to KS for this specific data. However, the spike sorting accuracy is higher for the single-unit clusters retrieved by KS. The average KS unit also consists of fewer merged sub-clusters, indicating that less effort has to be spend on the manual curation. Based on these identified differences, we can differentiate between the two algorithms for this specific recording setting and choice of spike sorting parameters depending on the intended further use of the spike sorting results. In case these spike sorting results are further used for the estimation of neural population dynamics, we argue that having a higher number of single units at a slightly reduced spike sorting performance is favourable (motivated by [121]), as such favouring SC for this application. However, when these spike sorting results are used for investigating the exact role of a single neuron, e.g., investigating its spatial tuning [78], the quality of the sorting is more important than the quantity of retrieved single units. As such, favouring KS over SC in this context. More generally speaking, depending on which sorting characteristics are favourable for a specific application, an informed choice can be made between spike sorting algorithms by using hybrid data.

A key difference between both spike sorting algorithms compared here, is that KS requires the user to define the number of clusters prior to the actual spike sorting. The KS spike sorting results discussed above, used a value of 256 for the number of clusters. To try to increase the number of recovered single-unit spike trains for KS, the number of clusters is chosen as 512 , while keeping the default setting for all other parameters. The results of this second KS run are represented in Figure 4.7 by the yellow bars.

As can be seen from Figure 4.7**A**, there is no direct evidence that KS with more clusters recovers more single-unit spike trains. Only two additional single-unit spike trains are recovered under the new number of clusters setting for the SC-informed data. However, the F_1-score for the recovered single-unit spike trains does increase even further, as compared to the previous spike sorting runs, as can be seen from Figure 4.7**C** and 4.7**D**. As can be seen from this example, it is non-trivial to predict the effect of changing spike sorting related parameters on the spike sorting performance.

4.5 Discussion and Conclusion

In this work we first formally introduced the hybrid ground-truth model. We presented a graphical tool to aid the creation of such hybrid ground-truth data. This graphical spike hybridizer for extracellular recordings or SHYBRID makes the creation of hybrid data very accessible to experimenters and tries to improve a user's spike sorting experience. Because of the visual approach, a user can easily verify the key hybrid model assumptions and monitor the quality of the resulting hybrid recordings. Besides the data generation aspect, functionality is provided for automatic cluster merging and very time-efficient spike sorting performance metric calculation through hash functions.

A case study was conducted where the performance of two different spike sorting algorithms on a hybridized recording was investigated. From the spike sorting results obtained on the hybrid ground truth, clear differences in spike sorting characteristics between the two algorithms could be identified for this data. Such objectively identified differences can help a user to make an informed decision about which algorithm to use for a certain application, depending on the targeted algorithmic performance characteristics. During the case study the effect of changing spike sorting related parameters was investigated. It was demonstrated that the effect of adapting a parameter that controls the total number of spike clusters does not necessarily have the anticipated effect of recovering more single-unit spike trains. Hybrid ground-truth data allows for an objective quantification of the effect of specific parameter settings. By studying the effect of different parameter settings, a deeper understanding of spike sorting algorithms is promoted.

The use of hybrid ground-truth data should lead to a higher spike sorting performance and less time spent on manual curation when applied to similar future data. This is especially interesting for chronic recordings, where the measurement equipment is kept in the same brain region over multiple weeks to months. In such experiments a multitude of recordings is available from the same region. Especially for such chronic experiments, the initial hybridization effort might be worth the investment. It is needless to say that this tool is also valuable for the community of spike sorting developers. By sharing generated hybrid data within the community, a large body of extracellular ground-truth data can be obtained for future benchmarks.

Almost simultaneously with the release of SHYBRID, the MEArec [12] testbench simulator for ground-truth extracellular spiking data was released. The development of this tool is illustrative for the need of spiking ground-truth data for validating and better understanding spike sorting algorithms. MEArec generates its ground-truth data from computational simulations, using complex

biophysically detailed models. Such a modelling approach enables full control over complex phenomena such as, e.g., bursting activity, drift or spatio-temporal synchrony. However, this high level of control implies that users have to acquire expert technical know-how in order to generate meaningful data, in particular when the aim is to generate ground-truth data that is representative for a specific experiment and/or recording set-up. Therefore, MEArec seems mostly suited for, e.g., spike sorting developers or modelling experts rather than spike sorting users.

SHYBRID does not require such a deep level of technical modelling expertise to generate meaningful ground-truth data. Furthermore, the ground-truth data is fully generated from an actual recording, which makes it intrinsically tailored to the recording setting of the user. The ease of use of SHYBRID comes at the expense of reduced control over certain recording-related characteristics, as compared to MEArec. Bursting cells can be included in the hybrid ground truth data, if they are present in the prior spike sorting results. Note that such SHYBRID bursting cells are only a simple amplitude modulation model of bursting, and do not account for template shape modulation that might occur for true bursting cells. Spatio-temporal overlap arises naturally when relocating units during the hybridization process, but synchrony can not be enforced. Drift simulation is not supported in SHYBRID for the time being, because it does not arise naturally from the presented hybrid model. A drift simulation framework could, technically speaking, be added to our software in a future update, but this will come at the cost of increased complexity for the user. SHYBRID and MEArec are complementary tools designed with the same end goal, i.e., improving spike sorting results, but both targeted at a different primary audience.

4.6 Appendix

4.6.1 Auto hybridization fitting factor bounds

The calculation of the fitting factor bounds during the automatic hybridization is based on robust statistics, which are commonly used for the detection and removal of outliers. The automatic bounds selection is rather conservative, i.e., it is likely that quite a few good spikes are excluded from the hybridization when using the automated approach.

Consider $\mathcal{B}_i = \{\log_{10}\ \beta_i^s \mid s \in \mathcal{T}_i\}$ which is the set of the logarithm of the fitting factors (see Section 4.2) for a certain neuron i. The logarithm is used to be able to also remove close to zero fitting factors based on simple statistics. Given $\mathcal{B}_i$,

the first and third quartile are calculated, denoted by Q_1 and Q_3 respectively. From those quartile values the interquartile range (IQR) is calculated as $IQR = Q_3 - Q_1$. From those statistics the bounds are calculated:

$$l_i^{\beta} = 10^{Q_1 - \frac{3}{4}IQR}, \tag{4.7}$$

and

$$u_i^{\beta} = 10^{Q_3 + \frac{3}{4}IQR}, \tag{4.8}$$

where the IQR scaling factor (i.e. $\frac{3}{4}$) was determined experimentally.

4.6.2 Auto hybridization random unit relocation

During the automatic hybridization, a random unit relocation is calculated for every neuron. For this relocation, only a shift in the y-direction is considered. The random shift is determined by drawing a y-position on the probe grid model (see Section 4.2) from a discrete uniform distribution. This random y-position is the y-position to which the channel with the maximal deflection in the spike template is shifted to. In this way we avoid that the complete template is shifted off the probe. The actual shift can then be calculated as the random y-position minus the y-position of the channel with maximal deflection in the original template. A minimum shift of two channels is enforced, to make sure that the re-inserted unit is sufficiently separable from the original unit.

4.6.3 External template import

When an external template is imported, there are no spike times available, neither is the scaling known. The spike occurrences are modeled as a poisson point process. The inter-spike interval Δ_{ISI} is then modelled by drawing from an exponential distribution:

$$p(\Delta_{ISI}, \lambda) = \lambda \exp\left(-\lambda \Delta_{ISI}\right), \tag{4.9}$$

where λ represents the desired spike rate. Every inter-spike interval sample $\hat{\Delta}_{ISI}$ is enforced to last at minimum the user-defined refractory period Δ_{min}:

$$\hat{\Delta}_{ISI} \leftarrow \max\left(\hat{\Delta}_{ISI}, \Delta_{min}\right). \tag{4.10}$$

The actual simulated discrete spike times s_{sim} are obtained by calculating the cumulative sum over the inter-spike interval samples. Those spike times are then discretized by multiplying them with the recording sampling frequency and rounding each product to its nearest integer. This gives rise to a set of discrete spike times $\mathcal{T}_{\text{ext}} = \{s_{\text{sim}}\}$.

The template scaling is derived from the user-defined desired peak-signal-to-noise ratio ($\text{PSNR} = 10 \log_{10} \frac{P_{\text{peak}}}{P_{\text{noise}}}$). The scaling factor is calculated as follows:

$$\beta_{\text{ext}} = \sqrt{\frac{P_{\text{noise}} 10^{\frac{\text{PSNR}}{10}}}{P_{\text{peak}}}}, \tag{4.11}$$

with P_{peak} equal to the square of the peak absolute value over all channels of the external template and P_{noise} equal to a robust estimate (based on the median absolute deviation) of the noise variance of the channel on which the template reaches its peak absolute value.

The hybrid data generated from an external template can then be described as follows:

$$\mathbf{h}^c_{\text{ext}}[k] = \mathbf{x}^c[k] + \sum_{s \in \mathcal{T}_{\text{ext}}} \sum_{m=-K}^{K} \delta[k - (s+m)] \, \beta_{\text{ext}} \boldsymbol{\tau}^{c,(x,y)}_{\text{ext}}[m], \tag{4.12}$$

where $\boldsymbol{\tau}^{c,(x,y)}_{\text{ext}}$ denotes the imported external template at channel c. Note that the template temporal window is derived from the external template directly. The external template is assumed to match the sampling frequency of the recording data that is being hybridized.

4.6.4 Automatic merging

The merging framework for a specific ground-truth spike train consists of the following steps:

1. Compute the correspondence between the ground-truth spike train and all automatically recovered spike clusters in terms of precision and recall. More information on those performance metrics can be found in Section 4.3.7.

2. Sort all clusters on descending precision, such that the cluster with the highest fraction of true spike times is on top of the list.

3. Merge the ordered clusters together in a top-down fashion, i.e. starting from the cluster with the highest precision, as long as the merge operation increases the F_1-score of the new cluster that contains all previously merged clusters.

Initially, the merging of clusters with a high precision will increase the sensitivity, at only a very small drop in precision. Such a merging will likely lead to an increase in F_1-score. At a certain point, clusters will start containing significant amounts of false positives that will notably decrease the precision of the merged cluster. This decrease will then result in a decreasing F_1-score. The proposed approach tries to find the combination of clusters with maximal F_1-score, without explicitly having to consider all possible combinations, preventing a combinatorial explosion from happening.

	Spyking Circus				KiloSort				KiloSort more clusters			
unit	F_1-*score*	*precision*	*recall*	# *clusters*	F_1-*score*	*precision*	*recall*	# *clusters*	F_1-*score*	*precision*	*recall*	# *clusters*
1	0.845	0.856	0.834	1	0.840	0.816	0.866	1	0.521	0.411	0.709	2
2	**0.994**	**0.991**	**0.996**	**1**	0.568	0.646	0.506	2	0.522	0.469	0.589	3
3	**0.963**	**0.972**	**0.953**	**5**	**0.973**	**1.000**	**0.945**	**1**	**0.966**	**0.936**	**0.997**	**2**
4	**0.995**	**1.000**	**0.990**	**1**	**0.996**	**1.000**	**0.992**	**1**	**1.000**	**1.000**	**1.000**	**1**
5	0.235	0.274	0.206	1	0.339	0.258	0.494	1	0.310	0.184	0.989	1
6	**0.963**	**1.000**	**0.928**	**1**	0.610	0.977	0.443	1	0.298	0.175	0.990	1
7	**0.946**	**0.897**	**1.000**	**1**	0.505	0.368	0.802	2	0.467	0.343	0.729	1
8	0.287	0.168	1.000	1	0.108	0.060	0.553	1	0.115	0.061	0.979	1
9	0.090	0.053	0.277	2	0.110	0.107	0.113	2	0.171	0.194	0.153	4
10	**0.973**	**0.953**	**0.993**	**1**	0.460	0.300	0.986	2	0.456	0.297	0.986	3
11	**0.993**	**0.987**	**1.000**	**1**	0.700	0.592	0.857	1	**0.997**	**0.993**	**1.000**	**2**
12	**0.991**	**0.995**	**0.988**	**5**	**0.997**	**1.000**	**0.994**	**1**	**0.999**	**1.000**	**0.998**	**2**
13	**0.992**	**0.984**	**1.000**	**1**	0.813	0.686	1.000	1	0.756	0.643	0.917	1
14	0.879	0.855	0.905	2	0.448	0.312	0.796	2	0.835	0.760	0.926	1
15	0.015	1.000	0.008	1	**1.000**	**1.000**	**1.000**	**2**	**1.000**	**1.000**	**1.000**	**1**
16	**1.000**	**1.000**	**1.000**	**1**	0.132	0.071	0.978	3	0.496	0.370	0.756	1
17	**0.972**	**0.950**	**0.995**	**1**	0.218	0.138	0.514	1	0.410	0.259	0.990	2
18	**0.975**	**0.956**	**0.996**	**5**	**0.998**	**0.998**	**0.997**	**1**	**0.999**	**1.000**	**0.997**	**2**
19	0.130	0.070	1.000	1	0.442	0.311	0.762	1	0.281	0.170	0.821	1
20	**0.916**	**0.854**	**0.988**	**2**	0.615	0.505	0.786	1	**0.986**	**1.000**	**0.972**	**1**
21	0.787	0.657	0.982	2	0.886	0.926	0.849	1	0.432	0.275	1.000	1
22	**0.981**	**0.963**	**1.000**	**2**	0.847	0.734	1.000	1	0.727	0.571	1.000	1
23	**0.946**	**0.955**	**0.937**	**9**	**0.959**	**1.000**	**0.917**	**1**	**0.989**	**0.986**	**0.993**	**3**
24	0.834	0.808	0.862	3	**0.945**	**1.000**	**0.896**	**2**	**0.973**	**0.997**	**0.951**	**1**
25	**0.902**	**0.821**	**1.000**	**6**	**0.999**	**1.000**	**0.997**	**2**	**0.999**	**1.000**	**0.997**	**2**
26	**0.974**	**0.980**	**0.967**	**4**	**0.997**	**1.000**	**0.994**	**2**	**0.992**	**0.986**	**0.999**	**3**
27	**0.979**	**0.970**	**0.988**	**1**	**0.939**	**0.886**	**0.999**	**1**	**0.996**	**1.000**	**0.991**	**1**

Table 4.1: Spike sorting performance metrics for the three separate spike sorting runs (SC, KS, KS more clusters) applied on the SC-informed hybrid ground-truth data. For every ground-truth unit in every spike sorting run, the F_1-score, precision, recall and number of cluster merges is shown. Rows with bold typesetting indicate the ground-truth units that were recovered as single-unit spike trains. This table is generated by SHYBRID.

	Spyking Circus				**KiloSort**				**KiloSort more clusters**			
unit	F_1*-score*	*precision*	*recall*	# *clusters*	F_1*-score*	*precision*	*recall*	# *clusters*	F_1*-score*	*precision*	*recall*	# *clusters*
1	0.298	0.192	0.660	1	0.106	0.056	0.992	3	0.139	0.139	0.140	3
2	0.678	0.516	0.988	1	0.643	0.474	1.000	2	0.601	0.430	1.000	3
3	0.283	0.165	0.998	1	0.702	0.546	0.984	1	0.440	0.415	0.469	2
4	0.084	0.070	0.105	8	0.047	0.043	0.053	13	0.090	0.065	0.143	29
5	**0.928**	**0.915**	**0.941**	**4**	0.893	0.991	0.813	1	0.364	0.263	0.590	9
6	**0.936**	**0.928**	**0.943**	**8**	**0.962**	**0.976**	**0.948**	**2**	**0.983**	**0.981**	**0.985**	**5**
7	**0.952**	**0.911**	**0.997**	**1**	0.263	0.154	0.903	3	0.629	0.481	0.909	4
8	**0.999**	**0.998**	**1.000**	**3**	**0.999**	**1.000**	**0.998**	**2**	**0.988**	**0.978**	**0.998**	**1**
9	0.433	0.337	0.604	2	0.789	1.000	0.652	2	0.613	0.622	0.604	4
10	**0.962**	**0.974**	**0.951**	**6**	**0.999**	**0.999**	**0.999**	**2**	**0.997**	**0.994**	**0.999**	**2**
11	0.808	0.971	0.693	2	0.633	0.465	0.991	1	0.631	0.537	0.764	1
12	0.615	0.481	0.854	2	0.685	0.527	0.978	1	0.683	0.519	1.000	2
13	**0.920**	**0.869**	**0.978**	**6**	**0.907**	**0.832**	**0.997**	**3**	0.866	0.965	0.785	1
14	0.800	0.993	0.670	1	**0.997**	**0.997**	**0.997**	**2**	**0.980**	**1.000**	**0.961**	**4**
15	**0.971**	**0.983**	**0.960**	**6**	0.702	0.542	0.997	1	**0.972**	**1.000**	**0.945**	**3**

Table 4.2: Spike sorting performance metrics for the three separate spike sorting runs (SC, KS, KS more clusters) applied on the KS-informed hybrid ground-truth data. For every ground-truth unit in every spike sorting run, the F_1-score, precision, recall and number of cluster merges is shown. Rows with bold typesetting indicate the ground-truth units that were recovered as single-unit spike trains. This table is generated by SHYBRID.

Chapter 5

Multi-pattern recognition through maximization of signal-to-peak-interference ratio

This chapter has been submitted for publication as *Jasper Wouters, Panagiotos Patrinos, Fabian Kloosterman, and Alexander Bertrand. "Multi-pattern recognition through maximization of signal-to-peak-interference ratio with application to neural spike sorting".* Jasper Wouters has co-developed the methodology, conducted the experiments and has written the manuscript. Compared to the submitted manuscript, minor textual and notational changes have been implemented for better integration in this thesis.

In this chapter, we propose three novel linear filter design methods for use in a multi-pattern recognition task with overlapping patterns and strong peak interferers. The recognition is based on a linear filter-and-threshold approach, which is particularly interesting when the task has to be performed in a computationally constrained environment. The first method optimizes the signal-to-peak-interference ratio (SPIR) of the filter output, where the focus is on minimization of the post-filtering peak interference instead of the pre-filtering peak interference as in existing methods. The second and third method are convex approximations of the first method and are shown to be closely related to support vector machines, which establishes a natural link between SPIR-optimal filtering and the maximum margin matched filter. The proposed methods only require a template of the target patterns as prior knowledge and do not require the training data to be labelled. An extensive case study is presented in the context of neural spike sorting, in which the proposed approaches are shown

to significantly outperform existing filter-and-threshold approaches for spike sorting.

5.1 Introduction

A long-standing method used in pattern recognition is the matched filter [122]. The essence of the matched filter is that it maximizes the signal-to-noise ratio (SNR) at its output, possibly after a pre-whitening of the data. Qualitatively speaking, this means that a high instantaneous filter output power is desired when the pattern of interest, also referred to as the template, is present at the input of the filter, and a low output power is desired when only noise is present at the filter input. Such a filter enables the detection and localization of several occurrences of the pattern of interest in the input sequence, e.g., by comparing the filter output to a threshold. The matched filter-and-threshold approach is the optimal detector in the case of additive Gaussian noise [45]. Because of (and despite) its simplicity, the matched filter is still common practice in many application fields that require detection methods, e.g., telecommunications [106] [107], astrophysics [79] [84], and biomedical science [19] [135]. Because of its minimal computational footprint, the linear matched filter-and-threshold approach for pattern recognition is especially relevant for real-time pattern recognition in computationally constrained environments, e.g., for real-time brain activity pattern recognition in neural implants [136] [123].

In this work we focus on the recognition of multiple patterns in a signal, i.e., the detection and localization of patterns originating from multiple sources, while suppressing interfering sources and discriminating between the occurrence of patterns from different sources. A well-known example of such a multi-pattern recognition task is the neural spike sorting problem [33, 67, 94, 103]. Spike sorting is the process of detecting neuronal action potentials or 'spikes' in an extracellular brain recording and assigning each detected spike to the neuron that presumably generated that spike. A naive approach for solving this problem consists of designing a matched filter for every distinct pattern source, i.e., for every detectable neuron. However, such a naive approach is known to produce filters that are incapable of properly discriminating between distinct pattern sources [30]. The lack of discriminating capability can be explained by the fact that the non-target pattern sources are usually only sparsely active in time in many applications. As such, those interfering sources are typically weakly weighted in the estimated noise statistics. To cope with this problem, signal-to-interference-plus-noise ratio (SINR) optimal filters have been proposed [30] [100] [126]. For such filters a trade-off can be made between the suppression of the noise floor and (known) interfering sources. This trade-

off potentially gives rise to filters that have better discrimination properties. However, choosing the trade-off is non-trivial in practice and therefore there are no guarantees in terms of discrimination properties. The pattern recognition performance then strongly depends on a manual tuning by an expert during the actual filter design process.

Signal-to-peak-interference ratio (SPIR) optimal filters were proposed in Chapter 2 to improve the discriminating capability of the matched filter. The idea behind the SPIR-optimal filter is to only focus on suppressing inputs that cause interfering peaks in the filter output. As such, no degrees of freedom are allocated on trying to attenuate inputs that are not contributing to spurious output peaks. When training such filters from a classical matched filtering point of view, interference segments have to be identified prior to the filter design based on heuristics [131], which suffer from errors. Furthermore, even if these errors would not exist, all the degrees of freedom in such filter designs are spent on maximally suppressing the interference peaks that are observed prior to the filter design, while the resulting filtering operation itself could possibly introduce new spurious peaks at its output. Therefore, we argue that these methods actually only use a proxy for the SPIR based on statistics collected at the filter *input*, and hence do not truly optimize the SPIR at the filter *output*. A naive attempt to fix this could be to re-compute the heuristic in an iterative fashion based on knowledge of the output of the filter design in the previous step. However, there would be no convergence guarantees towards a SPIR-optimal filter.

In this work we propose three optimal filter design methods that optimize the SPIR of the filter output directly without labelling interfering segments at the pre-filtering stage. Therefore, these filters are anticipated to be more suitable for multi-pattern recognition when using a filter-and-threshold approach. The first method minimizes a power loss function, which is a generalization of the classical matched filter. The second method is a convex approximation of the former to avoid problems with local minima. The third method is an alternative convex approximation which exploits additional information that is available from the convex constraint. Besides guarantees on global optimality, the convex reformulation provides interesting parallels with support vector machines (SVM). These parallels provide additional insights into the filter design mechanics and its resulting robustness, leading to a maximum margin matched filter interpretation.

The outline of the chapter is as follows. We formalize the problem statement of multi-pattern recognition in Section 5.2 and review matched filtering approaches that have been applied to this problem in Section 5.3. In Section 5.4 we introduce the novel non-convex SPIR-optimal filter design method and in Section 5.5 we present two convex reformulations thereof. In Section 5.6 we review an

alternative regularization approach for the proposed filter design methods, after which we benchmark the proposed methods in an extensive case study of neural spike sorting in Section 5.7. Finally, in Section 5.8 we further discuss the main results and draw conclusions.

5.2 Problem statement

Consider the N-dimensional signal $\mathbf{x}[k] \in \mathbb{R}^N$ at sample time k which could contain the samples collected at N sensors at time k (spatial filtering) or a delay line of N samples of a single sensor (temporal filtering), or a combination of both (spatio-temporal filtering). Without loss of generality we assume that $E\{\mathbf{x}\} = \mathbf{0}$, i.e, the signal is zero-mean along its different dimensions. The signal $\mathbf{x}[k]$ is modelled as

$$\mathbf{x}[k] = \sum_{i=1}^{M} \mathbf{p}_i[k] + \mathbf{n}[k], \tag{5.1}$$

where the M different $\mathbf{p}_i$ are so-called (sparse) pattern sources which have an 'on-off' characteristic and where $\mathbf{n}$ is the noise term absorbing all sources that are not pattern sources. A pattern source $\mathbf{p}_i[k]$ is characterized by the deterministic pattern $\boldsymbol{\pi}_i \in \mathbb{R}^N$ and is defined in the following way:

$$\mathbf{p}_i[k] = \sum_{a \in \mathcal{A}_i} \delta[k-a] * \mathbf{\Pi}_i[k], \tag{5.2}$$

with $\delta[k]$ being the Kronecker delta, i.e., $\delta[k] = 1$ if $k = 0$ and $\delta[k] = 0$ for all $k \neq 0$. $\mathcal{A}_i$ is the set of sparse pattern occurrence times for the pattern source i. $\mathbf{\Pi}_i[k]$ is a finite support signal defined such that $\mathbf{\Pi}_i[0] = \boldsymbol{\pi}_i$ where $\boldsymbol{\pi}_i$ is the known deterministic pattern of interest. The content of $\mathbf{\Pi}_i[k]$ for $k \neq 0$ may contain time-shifted versions of $\boldsymbol{\pi}_i$ in case the vector $\mathbf{x}[k]$ consists of a sample delay line. For the sake of an easy exposition, we will make abstraction of the exact content of $\mathbf{\Pi}_i[k]$ for $k \neq 0$.

When solving the multi-pattern recognition task, one wants to retrieve $\mathcal{A}_i$ for the pattern source of interest i, which we will refer to in this context as the *target* pattern source, given $\mathbf{x}[k]$ and the target pattern $\boldsymbol{\pi}_i$ or an estimate thereof. In this work a linear filter-and-threshold approach is taken for estimating $\mathcal{A}_i$ from the signal:

$$\hat{\mathcal{A}}_i = \left\{ k \mid \left(\mathbf{f}_i^T \mathbf{x}[k]\right)^2 > T_i \right\}, \tag{5.3}$$

where $\mathbf{f}_i \in \mathbb{R}^N$ denotes the linear filter, and T_i is a user-defined detection threshold. For every target pattern source i, a separate linear filter has to be designed and applied to the data in a parallel fashion where the other pattern sources are then treated as interferers, which should not cause the filter output to exceed the threshold T_i. In the remainder of this work we will focus on the design of such linear filters $\mathbf{f}_i$ that enable *filter-and-threshold-based pattern recognition* as given by (5.3).

5.3 Filter design review

In this Section we review several filter design methods that have been used in the context of filter-and-threshold-based pattern recognition. The different filter design approaches are all presented from an optimization point of view. This uniform presentation has the advantage that the different methods can be easily compared among each other.

5.3.1 Matched filtering

A classical filter design method that is used for filter-and-threshold-based pattern recognition is the matched filter, which is also referred to as the SNR-optimal filter. The matched filter $\mathbf{f}_i^{\mathrm{MF}}$ is given by the following optimization problem:

$$\mathbf{f}_i^{\mathrm{MF}} \sim \underset{\mathbf{f}}{\operatorname{argmax}} \frac{(\mathbf{f}^T \boldsymbol{\pi}_i)^2}{\frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} (\mathbf{f}^T \mathbf{n}[k])^2}, \tag{5.4}$$

where the numerator contains the filter output power when the target pattern $\boldsymbol{\pi}_i$ is present at the input and the denominator contains the average filter output power in response to the noise term $\mathbf{n}[k]$, effectively optimizing the output SNR. The training data is represented by $\mathcal{T}$ which is the set of sample times that are available during training of the filters. Note that $\mathbf{f}_i^{\mathrm{MF}}$ is only defined up to an arbitrary scaling, which is why we use '$\sim$' instead of '$=$' in (5.4).

We can also consider all pattern sources, except for target pattern occurrences, to be noise sources. When defining all non-target pattern sources and shifted versions of the target pattern as noise, we can adjust (5.4) to be

$$\mathbf{f}_i^{\mathrm{MF}} \sim \underset{\mathbf{f}}{\operatorname{argmax}} \frac{(\mathbf{f}^T \boldsymbol{\pi}_i)^2}{\frac{1}{|\mathcal{T} \setminus \mathcal{A}_i|} \sum_{k \in \mathcal{T} \setminus \mathcal{A}_i} (\mathbf{f}^T \mathbf{x}[k])^2}. \tag{5.5}$$

Note that this filter design requires complete knowledge about the target pattern source activity $\mathcal{A}_i$ in the training data, rendering it of lesser practical significance.

However, it was shown in Chapter 2 that the presence of the pattern of interest in the denominator does not affect the filter solution, and as such it was suggested in Chapter 3 that (5.5) can be rewritten as

$$\mathbf{f}_i^{\text{MF}} \sim \underset{\mathbf{f}}{\text{argmax}} \frac{(\mathbf{f}^T \boldsymbol{\pi}_i)^2}{\frac{1}{|\mathcal{T}|}\sum_{k\in\mathcal{T}} (\mathbf{f}^T \mathbf{x}[k])^2}, \tag{5.6}$$

which simplifies the filter design process greatly, by not having to compensate for the target pattern source and training the filter on the original acquired data immediately without knowledge of the activation times of the target pattern in the training data. It is well known that a closed-form solution to (5.6) is:

$$\mathbf{f}_i^{\text{MF}} = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1} \boldsymbol{\pi}_i, \tag{5.7}$$

where $\hat{\mathbf{R}}_{\mathbf{xx}} = \frac{1}{|\mathcal{T}|}\sum_{k\in\mathcal{T}} \mathbf{x}[k]\mathbf{x}[k]^T$ is the estimate of the signal covariance matrix. This solution is equivalent to a pre-whitening of the data followed by a matched filtering with the pre-whitened target pattern. In the special case where $\mathbf{x}$ is white over its different dimensions, i.e., $\hat{\mathbf{R}}_{\mathbf{xx}} = \mathbf{I}$ with $\mathbf{I}$ the identity matrix, the matched filter boils down to its simplest solution $\mathbf{f}_i^{\text{MF}} = \boldsymbol{\pi}_i$.

In many practical applications $\hat{\mathbf{R}}_{\mathbf{xx}}$ is ill-conditioned, or even rank deficient. Therefore, a diagonal loading can be applied to the covariance matrix estimate prior to inversion:

$$\mathbf{f}_i^{\text{MF}} = \left(\hat{\mathbf{R}}_{\mathbf{xx}} + C\mathbf{I}\right)^{-1} \boldsymbol{\pi}_i, \tag{5.8}$$

where C is a hyper-parameter that controls the condition number of the resulting sum matrix.

We will now transform (5.6) in an equivalent problem that will serve as the basis for the later algorithms introduced in this chapter. Note that the maximum of (5.6) can be found by fixing the numerator to a constant value K and minimizing the denominator (note that this also resolves the non-uniqueness problem related to optimizing the ratio (5.6)). When also including the diagonal loading introduced in (5.8), we arrive to the following equivalent constrained optimization problem

$$\begin{aligned} \mathbf{f}_i^{\text{MF}} = &\underset{\mathbf{f}}{\text{argmin}} \frac{1}{|\mathcal{T}|}\sum_{k\in\mathcal{T}} (\mathbf{f}^T \mathbf{x}[k])^2 + C\|\mathbf{f}\|_2^2 \\ &\text{subject to} \quad (\mathbf{f}^T \boldsymbol{\pi}_i)^2 = K \end{aligned}, \tag{5.9}$$

where $K \in \mathbb{R}_{>0}$ is an arbitrary strictly positive number, that controls the pattern of interest's output response power. Note that the diagonal loading in (5.8) is equivalent to a Tikhonov regularization in the above reformulation.

Hence, the diagonal loading in (5.8) pushes the solution to the minimal norm solution for $C \to \infty$. This minimal norm solution is the target pattern $\boldsymbol{\pi}_i$ itself, as can be straightforwardly seen from (5.8). Although such matched filters have been used in the context of filter-and-threshold-based pattern recognition, they were shown to be insufficiently discriminative [30] for the target pattern, because they typically focus on minimizing the baseline noise, but fail to attenuate sparse peaks, e.g., caused by the other pattern sources or peak-interference sources, hence leading to false positive threshold crossings.

5.3.2 Matched filtering with source weighting

In an attempt to improve the filter-and-threshold-based pattern recognition performance, variants of the matched filter have been proposed [30] [100] [126], where the different additive sources in (5.1) are weighted individually. By using separate weights for the different sources, the aim is to improve the discriminating capabilities of the filter. Such a weighted matched filter $\mathbf{f}_i^{\text{WMF}}$ results from the following optimization problem (cf. (5.9)):

$$\begin{aligned} \mathbf{f}_i^{\text{WMF}} = \underset{\mathbf{f}}{\text{argmin}} \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \big(& w_n \cdot (\mathbf{f}^T \mathbf{n}[k])^2 \\ & + \sum_{j=1}^{M} w_j \cdot (\mathbf{f}^T \mathbf{p}_j[k])^2 \big) + C \|\mathbf{f}\|_2^2 \\ & \text{subject to} \quad (\mathbf{f}^T \boldsymbol{\pi}_i)^2 = K, \end{aligned} \tag{5.10}$$

where w_n is the weight associated with the filter output power in response to the noise source $\mathbf{n}$. w_j are the weights associated with the filter output power in response to the pattern source $\mathbf{p}_j[k]$. Note that this method assumes complete knowledge about all the pattern instances that are present in the training data and requires an estimate of the covariance matrix of the noise $\mathbf{n}$, which can be estimated during segments where none of the pattern sources are active. Furthermore, no concrete methods are proposed for choosing the values of the different weights, as such these weights should be seen as hyper-parameters that should be optimized based on a cross-validation scheme, leading to an $M + 2$-dimensional grid-search, which quickly becomes infeasible for increasing M. While (5.10) is an interesting generalization of (5.9), all the aforementioned issues make it hard to use it in practice.

5.3.3 SPIR-based filtering

The concept of SPIR-optimal filter design was proposed in Chapter 2 in an attempt to improve the discriminating capabilities of the filters $\mathbf{f}_i$ with respect to their target patterns $\boldsymbol{\pi}_i$. The main idea is to only allocate degrees of freedom in the filter design to signal segments that cause spurious peaks in the output during inactivity of the target pattern. SPIR-optimal filter design is also intended as a practical method, i.e., it does not require complete prior knowledge about different pattern sources in the mixture (as in Subsection 5.3.2) and works with a minimal set of hyper-parameters.

The qualitative differences between the SNR-optimal and SPIR-optimal filter are highlighted in Figure 5.1. The top graph shows the SNR-optimal filter output power, where two major peaks can be distinguished, of which only one peak reflects the occurrence of the target pattern. The bottom graph shows the SPIR-optimal filter output power. Here, only one dominant peak is present, while the previous spurious peak has been attenuated at the cost of an increase in overall noise power. The SPIR-optimal filter output shown in the bottom graph is suitable for filter-and-threshold-based pattern recognition, whereas the top graph does not allow for a robust filter-and-threshold-based discrimination between target and non-target pattern occurrences.

The SPIR-based filter design proposed in Chapter 2 results from solving the following optimization problem[1]:

$$\begin{aligned} \mathbf{f}_i^{\mathrm{SB}} = \underset{\mathbf{f}}{\operatorname{argmin}} \; & \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} w_k \cdot (\mathbf{f}^T \mathbf{x}[k])^2 + C\|\mathbf{f}\|_2^2 \\ \text{subject to} \quad & (\mathbf{f}^T \boldsymbol{\pi}_i)^2 = K, \end{aligned} \quad , \tag{5.11}$$

which is a generalization of (5.9) by the introduction of the binary weights $w_k \in \{0, 1\}$, which effectively select a subset of training samples when designing the filter.

In Chapter 2, the following heuristic for the estimation of w_k was proposed (slightly simplified here for illustrative purposes):

1. Design a classical matched filter according to (5.9).
2. Apply the matched filter to the data and identify segments for which the output power exceeds γK with $\gamma \in [0, 1]$.

[1] Note that the optimization problem (5.11) can be written as a maximization of a ratio, similar to (5.6)-(5.9), which explains the term SPIR.

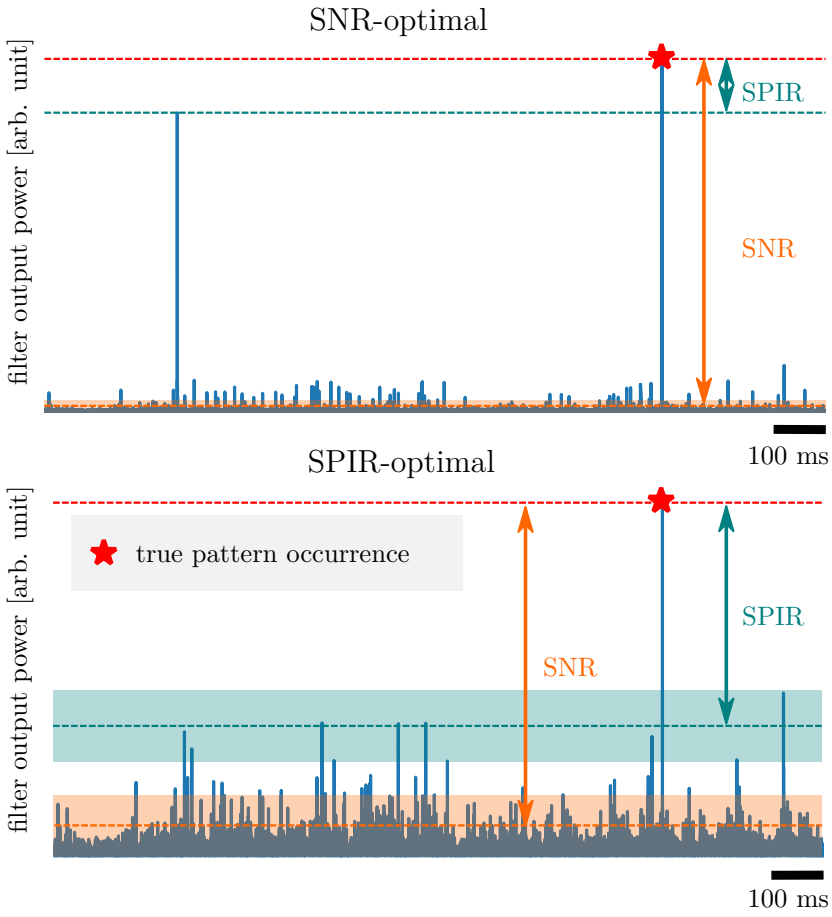

Figure 5.1: Qualitative comparison between an SNR- and SPIR-optimal filter output. The SPIR-optimal filter has an increased SPIR compared to the SNR-optimal one, at the acceptable cost of a decrease in SNR, enabling a more robust filter-and-threshold-based pattern recognition for the SPIR-optimal filter output.

3. For every sample time k for which the output power threshold is exceeded, set $w_k = 1$. Otherwise, $w_k = 0$.

4. Compute the SPIR-based filter (5.11).

The method was shown capable of outperforming the matched filter in a simple treshold-based pattern recognition task. However, this heuristic procedure (as

well as any other alternative heuristic) may make wrong decisions in selecting the peak interference segments. Furthermore, we will argue in the next section that – even without such errors – the filter design (5.11) is not truly SPIR-optimal when the weights w_k are a-priori fixed.

5.4 Non-convex SPIR-optimal filtering

5.4.1 Definition of SPIR-optimal filtering

The filter design in (5.11) does not optimize the SPIR of the filter output directly, i.e., there is no mechanism that prevents samples $\mathbf{x}[k]$ with $w_k = 0$ from generating a peak response in the filter output. Note that the weights w_k in the heuristic procedure explained in Subsection 5.3.3, are determined based on the peak interferers in the output of a *different* filter than the final filter that is generated by (5.11). In principle, the optimal binary weights in (5.11) should be determined based on the interference peaks in the output of the filter $\mathbf{f}$ *itself*, which creates a chicken-and-egg situation. A straightforward – yet naive – fix would be to compute (5.11) multiple times, where the weights w_k are every time determined by the peak interference in the output of the previous filter. However, this would require a human expert to terminate this iteration, since there are no guarantees on convergence towards a workable solution of this ad-hoc process. Furthermore, none of the resulting filters would be truly SPIR-optimal as the selection of interfering segments would always be based on a different filter than the one eventually used. To cope with these fundamental limitations we propose a truly SPIR-optimal filter design method, which inherently and automatically chooses the weights w_k based on the peak interference in the filter *output* signal. This makes the weights dependent on $\mathbf{f}$ itself, thereby creating an interaction between the filter coefficients in $\mathbf{f}$ and the weights w_k that define the filter. By including this chicken-and-egg situation in the optimization problem itself, we remove the need for an ad-hoc iterative design, while convergence towards a truly SPIR-optimal filter can be guaranteed.

To acquire a truly SPIR-optimal filter $\mathbf{f}_i^{\mathrm{SO}}$, we replace the prior weights w_k in (5.11) by a weighting function g that depends directly on the filter output power:

$$\boxed{\begin{array}{c}\mathbf{f}_i^{\mathrm{SO}} = \underset{\mathbf{f}}{\operatorname{argmin}} \dfrac{1}{|\mathcal{T}|} \displaystyle\sum_{k\in\mathcal{T}} g\left((\mathbf{f}^T\mathbf{x}[k])^2\right)(\mathbf{f}^T\mathbf{x}[k])^2 + C\|\mathbf{f}\|_2^2 \\ \text{subject to} \quad (\mathbf{f}^T\boldsymbol{\pi}_i)^2 = K\end{array}}, \tag{5.12}$$

where the weighting function $g : \mathbb{R} \mapsto \mathbb{R}$ is given by

$$g(s) = \frac{1}{1 + e^{-(s-\gamma K)}}, \tag{5.13}$$

where $\gamma \in [0, 1]$. This weighting function is a shifted sigmoid and is a differentiable approximation of the step function. This function approximately implements a binary weighting, which is in line with the use of binary weights in (5.11). As a result, samples with a power smaller than γK at the output of the filter $\mathbf{f}$ are not treated as peak interference and are ignored in the filter design of $\mathbf{f}$. Note that K in (5.13) is the same as in the constraint of (5.12), i.e., γK determines the fraction of the output power in response to the target pattern. We will explain later on (Subsection 5.4.3) how γ can be chosen to steer the filter design towards a predefined desired SPIR.

By making use of a weighting function g rather than prior weights w_k, the resulting filter is truly SPIR optimal. The price that one has to pay is that as opposed to (5.9), (5.10) and (5.11), the filter design method in (5.12) has no closed-form solution. Therefore, one has to use an optimization solver, e.g., based on gradient-descent, to obtain a non-convex SPIR-optimal filter. Depending on the solver that is used, convergence to a solution can be guaranteed. However, because (5.12) is non-convex, the resulting filter might only coincide with a local minimum of the objective function. In Section 5.5, we will propose a convex approximation of (5.12) to avoid such local minima, and which allows to use efficient solvers for convex optimization.

Note that only limited prior knowledge is needed for designing a SPIR-optimal filter. Only the target pattern $\boldsymbol{\pi}_i$ is required (as opposed to the method described in Section 5.3.2). Furthermore, the filter design is controlled by two hyper-parameters: the interpretable interference fraction γ and the regularization hyper-parameter C. In Section 5.6 we will apply the regularization scheme from Chapter 3 to this filter design, that will lead to a regularization hyper-parameter that can be more easily tuned and interpreted. Note that K is not truly a hyper-parameter, but an arbitrary constant that has little influence on the SPIR itself, as long as K is chosen large enough, as explained in the next subsection.

5.4.2 Choice of K

We did not include an extra hyper-parameter in (5.13) to control the slope of the shifted sigmoid. This can be implicitly controlled through the parameter K, which is an arbitrary constant (see also (5.9)) defining the filter output power in response to the target pattern. By choosing a large K, the rise time of the

sigmoid can be made arbitrarily short, i.e., relative[2] to the length of the interval $[0, K]$ that determines the scale of the output signal. If K is large enough, the sigmoid will sufficiently 'behave' as a step function (in practice $K = 1000$ works well, $K = 10$ is too small as the unshifted sigmoid raises from 0 to 1 in the interval $[-5, 5]$, as is shown in Figure 5.2).

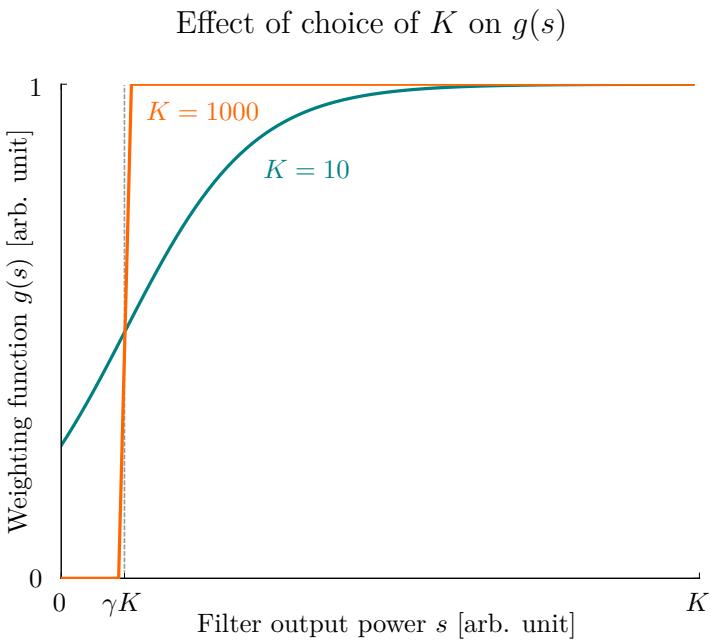

Figure 5.2: The shifted sigmoid weighting function only "behaves" as a step function for values of K that are sufficiently large. For $K = 10$ (green) and $\gamma = 0.1$, the shifted sigmoid does not reach zero for any $s \in [0, \gamma K]$. For $K = 1000$ (orange) and $\gamma = 0.1$ the shifted sigmoid is a good approximation to a step function.

5.4.3 Bounds on γ

As was mentioned before, γ is bounded by nature, i.e., $\gamma \in [0, 1]$. Here, we will further narrow down this interval. In all aforementioned filter design optimization problems, we explicitly control the filter output power for the target pattern to be K. Therefore, we are able to define an explicit interference

[2]Note that the rise time is fixed in absolute sense, but should be viewed relative to the interval $[0, K]$.

threshold γK in (5.13). By combining both the target power and interference threshold, we can define the desired SPIR (in dB):

$$\text{desired SPIR} = 10 \log \frac{K}{\gamma K} = -10 \log \gamma. \tag{5.14}$$

We refer to the above metric as *desired* SPIR, because there is no guarantee that all interferers are pushed below the interference threshold during the optimization process due to the finite degrees of freedom in the filter design. From Figure 5.1, it is clear that the SPIR will by definition always be smaller than the SNR. Since the matched filter (5.9) is SNR-optimal, we thus require that the desired SPIR $\leq \text{SNR}_{\text{MF}}$, where SNR_{MF} is the SNR at the output of the matched filter. From this inequality a lower bound on γ can be derived:

$$\gamma \geq \frac{1}{10^{\frac{\text{SNR}_{\text{MF}}}{10}}}. \tag{5.15}$$

Although no theoretical narrowing down of the upperbound is provided, it goes without saying that a γ close to one is considered bad practice, because this might lead to very low SPIR as can be seen from (5.14). This low SPIR will likely deteriorate the filter-and-threshold-based pattern recognition performance. Therefore, without further proof, we suggest as a pragmatic upper bound $\gamma = 0.5$.

5.5 Convex SPIR-optimal filtering

In this section we propose a convex reformulation of (5.12). A convex reformulation has the advantage that convergence can be guaranteed to the *global* optimum of the objective function rather than to a local optimum and that efficient convex optimization solvers can be used.

5.5.1 Definition of the convex SPIR-optimal filter

We define the convex SPIR-optimal filter $\mathbf{f}_i^{\text{CSO'}}$ as the solution to the following optimization problem:

$$\boxed{\begin{aligned} \mathbf{f}_i^{\text{CSO'}} = \underset{\mathbf{f}}{\text{argmin}} \; & \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} r\left(\left(\mathbf{f}^T \mathbf{x}[k]\right)^2 - \gamma K\right) + C\|\mathbf{f}\|_2^2 \\ & \text{subject to} \quad \mathbf{f}^T \boldsymbol{\pi}_i = \sqrt{K} \end{aligned}}, \tag{5.16}$$

where $r(p) = \max\{0, p\}$ is the rectified linear unit (ReLU). The ReLU can be interpreted as a binary weighting of p, i.e., $r(p) = 0 \cdot p$ if $p \leq 0$, otherwise $r(p) = 1 \cdot p$. With $p \leftarrow \left(\mathbf{f}^T\mathbf{x}\right)^2 - \gamma K$, a zero weight is applied to all training data points $\mathbf{x}$ for which $\left(\mathbf{f}^T\mathbf{x}\right)^2 \leq \gamma K$. The ReLU $r(p)$ is a convex function and its convexity is preserved under composition with another convex function, such as $\left(\mathbf{f}^T\mathbf{x}\right)^2 - \gamma K$. The nonnegative weighted sum over convex functions is also convex [10], thereby proving that the cost function in (5.16) is convex, given that $C \geq 0$. The set of points satisfying the power constraint in (5.12) do not form a convex set and can as such never lead to a convex optimization problem. Therefore, a linear constraint is used instead in (5.16), such that the feasible set of solutions is convex. The optimization problem in (5.16) is thus a convex optimization problem, because it has both a convex objective function and convex feasible set. Note that this convex reformulation is not a true power optimization any more, because the cost function sums over the binary weighted difference $\left(\mathbf{f}^T\mathbf{x}\right)^2 - \gamma K$, rather than $\left(\mathbf{f}^T\mathbf{x}\right)^2$ as in (5.12).

Because of the usage of a linear constraint in (5.16), rather than a quadratic constraint, the sign of the filter output response to the target pattern $\boldsymbol{\pi}_i$ is known a-priori. We can exploit this knowledge by applying an interference threshold to the filter output directly instead of the filter output power, i.e.,

$$\min_{\mathbf{f}} \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} r\left(\mathbf{f}^T\mathbf{x}[k] - \sqrt{\gamma K}\right) + C\|\mathbf{f}\|_2^2 \quad \text{subject to} \quad \mathbf{f}^T\boldsymbol{\pi}_i = \sqrt{K}, \tag{5.17}$$

thereby not "wasting" degrees of freedom during the filter design on peak interferers or pattern sources that create a negative response at the filter output. In an attempt to focus more on the attenuation of larger interfering peaks compared to smaller interfering peaks, we reintroduce the squaring operator outside of the ReLU, leading to an alternative convex SPIR-optimal filter $\mathbf{f}_i^{\text{CSO"}}$ problem formulation:

$$\boxed{\mathbf{f}_i^{\text{CSO"}} = \underset{\mathbf{f}}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \left(r\left(\mathbf{f}^T\mathbf{x}[k] - \sqrt{\gamma K}\right)\right)^2 + C\|\mathbf{f}\|_2^2 \quad \text{subject to} \quad \mathbf{f}^T\boldsymbol{\pi}_i = \sqrt{K}} \tag{5.18}$$

Again, it can be easily shown that the alternative cost function in (5.18) is convex. The squared ReLU $(r(p))^2$ is a convex function. It is known that convexity is preserved under composition with an affine transformation [10], i.e, $\left(r\left(\mathbf{f}^T\mathbf{x} - \sqrt{\gamma K}\right)\right)^2$ is convex if $(r(p))^2$ is convex. Again, the nonnegative

weighted sum over convex functions is also convex, thereby proving that the cost function in (5.18) is convex, given that $C \geq 0$. Note that by introducing auxiliary variables, (5.18) can be expressed as a quadratic program which can be solved reliably by standard interior point methods (this is derived in Section 5.5.2, i.e, in equation (5.21)). The convex reformulation in (5.18) is also not a true filter output power optimization, because the interference threshold $\sqrt{\gamma K}$ is subtracted from the filter output prior to squaring. It is also not a true power optimization because the filters under design do not attempt to attenuate negative peak interferers at all (as opposed to (5.16)). This means that more degrees of freedom can be exploited to suppress the peak interferers that really count, i.e., the ones generating positive peaks at the filter output. A logical consequence of not considering negative peak interferers during the filter design is that the filter-and-threshold-based pattern recognition using (5.18) has to be performed on the filter output amplitude directly:

$$\hat{\mathcal{A}}_i = \left\{ k \mid \mathbf{f}_i^T \mathbf{x}[k] > T_i' \right\}, \tag{5.19}$$

rather than on the filter output power as originally described in (5.3). Note that for applications where the target pattern is often superimposed onto strong negative peak interferers, the use of (5.18) may result in a large amount of missed detections (false negatives), i.e., resulting in a reduction in recall.

5.5.2 SVM interpretation

Besides guarantees on convergence to the global optimum, the convex SPIR reformulation in (5.18) provides an interesting SVM interpretation for SPIR-optimal filtering, leading to a maximum-margin interpretation of the SPIR criterion. Reconsider the convex reformulation (5.18) of the SPIR-optimal filter design problem, where the objective function is multiplied with $|\mathcal{T}|$:

$$\begin{aligned} \underset{\mathbf{f}}{\operatorname{argmin}} \sum_k \left(r \left(\mathbf{f}^T \mathbf{x}_k - \gamma \sqrt{K} \right) \right)^2 + C' \|\mathbf{f}\|_2^2 \\ \text{subject to} \quad \mathbf{f}^T \boldsymbol{\pi}_i = \sqrt{K} \end{aligned}, \tag{5.20}$$

with $C' = C|\mathcal{T}|$ and where the sample time k is moved to the subscript to be consistent with notation in the field of SVMs. By introducing the slack variables

e_k an equivalent optimization problem can be formulated:

$$\begin{aligned} \min_{\mathbf{f},\mathbf{e}} \sum_k e_k^2 + C'\|\mathbf{f}\|_2^2 \\ \text{subject to} \quad \mathbf{f}^T\boldsymbol{\pi}_i = \sqrt{K} \\ \mathbf{f}^T\mathbf{x}_k - \gamma\sqrt{K} \leq e_k \quad \forall k \\ e_k \geq 0 \quad \forall k \end{aligned} , \tag{5.21}$$

where we now optimize over both $\mathbf{f} \in \mathbb{R}^N$ and $\mathbf{e} \in \mathbb{R}^{|\mathcal{T}|}$, i.e., the collection of all e_k $\forall k \in \mathcal{T}$. Next, we define the so-called threshold t and margin ρ in terms of $\sqrt{K}$ and γ:

$$\begin{aligned} \sqrt{K} &= t + \rho \\ \gamma\sqrt{K} &= t - \rho \ . \end{aligned} \tag{5.22}$$

Reparametrizing the optimization problem in terms of t and ρ then results in:

$$\min_{\mathbf{f},\mathbf{e}} \sum_k e_k^2 + C'\|\mathbf{f}\|_2^2 \tag{5.23a}$$

$$\text{subject to} \quad \mathbf{f}^T\boldsymbol{\pi}_i - t = \rho \tag{5.23b}$$

$$y_k\left(\mathbf{f}^T\mathbf{x}_k - t\right) \geq \rho - e_k \quad \forall k \tag{5.23c}$$

$$e_k \geq 0 \quad \forall k, \tag{5.23d}$$

with $y_k = -1$ $\forall k$, which in a context of SVMs introduces an implicit "ground truth" label which is in our formulation the same for each sample.

This problem is equivalent to L2-SVM [62] except for the fact that all labels are negative and that an extra linear constraint (5.23b) is added. The extra equality constraint leads to the loss of one degree of freedom when compared to regular L2-SVM, i.e., here we don't optimize over the threshold t (both t and ρ are fixed by fixing K and γ). This difference is essential for the SPIR-optimal filter to work. To better appreciate this statement, remember that regular SVM is a supervised learning method. However, the filter design problem addressed in this chapter can be viewed as an extreme case of positive-unlabeled (PU) learning [23], i.e., we have only one labelled training sample, namely the target pattern $\boldsymbol{\pi}_i$ which represents the 'positive' class ($y = 1$). All other training samples are unknown, because they contain a mixture of noisy target and non-target pattern occurrences, and pure noise samples. In our SPIR formulation,

all of those samples apparently are assigned a negative ground truth label y_k, which we know is only partially true.

So one may wonder how this method can work, given the class imbalance and erroneous training labels. This is due to the additional equality constraint compared to regular SVM. The a-priori fixed target pattern $\boldsymbol{\pi}_i$ response is key in fighting class imbalance. Understanding the robustness against erroneous (or absence of) training labels, requires some geometrical elaboration on that same equality constraint. As shown graphically in Figure 5.3 for the case where $\mathbf{f} \in \mathbb{R}^2$, the equality constraint (5.23b) forces the pattern of interest $\boldsymbol{\pi}_i$ to be on the border of the margin ρ. To satisfy the equality constraint the separating hyperplane that is orthogonal to $\mathbf{f}_i$ has to be tangent to a half hypersphere with radius $\frac{\rho}{\|\mathbf{f}_i\|}$ and center point $\boldsymbol{\pi}_i$. The optimization problem thus consists of rotating the hyperplane on the hypersphere, trying to minimize the number of support vectors, i.e., training samples violating the predetermined margin $\frac{\rho}{\|\mathbf{f}_i\|}$, thereby driving as many e_k's close or equal to zero as possible. The support vectors are represented in Figure 5.3 as orange dots. Consider now the support vectors surrounding the target pattern in Figure 5.3 to represent occurrences of the target pattern in the training data. If the noise is spherically symmetric, rotating the filter hyperplane along the hypersphere, will not lead to a decrease in cost related to the target pattern occurrence support vectors. Therefore, the optimization algorithm will have to focus on the other support vectors to effectively decrease the cost, hence explaining the robustness against erroneous training labels.

Through this link between SPIR-optimal filtering and SVM, we can state that SPIR-optimal filtering can be viewed as the maximum margin counterpart of matched filtering. To the best of our knowledge, SPIR-filtering is the first true maximum margin approach to matched filtering. Similar approaches exist [101], but they have been formulated in a supervised setting, rather than the more practically relevant PU-learning setting that is more closely related to matched filtering. Another interesting result of the link between SVM and SPIR-optimal filtering, is the possibility to design a kernel variant of the SPIR-optimal filter design method similar to [48]. This kernel method has the potential of separating target pattern occurrences from the other samples in a non-linear way. Because the evaluation of kernel-SPIR-optimal filter requires for every data point the evaluation of the so-called kernel function over all its support vectors, it is not a low complexity filter and as such, it is outside of the scope of this work.

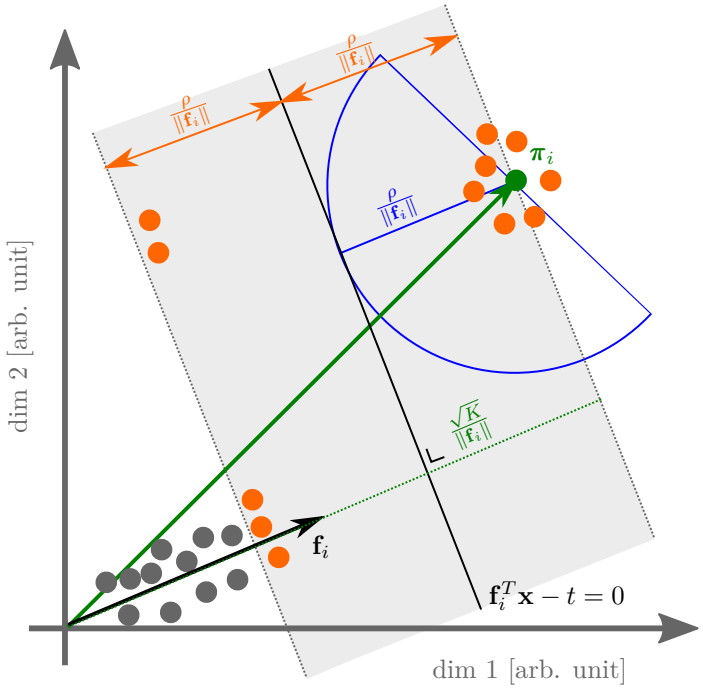

Figure 5.3: A geometric interpretation of the convex SPIR-optimal optimization problem in two dimensions ($\mathbf{f} \in \mathbb{R}^2$). The shaded area represents the margin area. The green dot represents the pattern of interest $\boldsymbol{\pi}_i$. The other dots represent the training points $\mathbf{x}_k$, where the orange dots represent the support vectors. The equality constraint (5.23b) is satisfied if the separating line (in black) is tangent to the half ρ-circle (in blue).

5.6 Subspace projection regularization

In this Section, we briefly review the concept of subspace projection regularization introduced in Chapter 3, which allows to avoid a tedious tuning of the Tikhonov regularization parameter C in, e.g, (5.9), (5.12), (5.16) and (5.18). Generally speaking, the regularization of a learning problem serves to shrink the solution space compared to the unregularized solution space $\mathbb{R}^N$. The main idea here is to project the data into a lower-dimensional subspace that is orthogonal to certain undesired directions. Excluding these undesired directions altogether is a stronger way of regularizing than, e.g., Tikhonov regularization,

where certain solutions are discouraged, but never explicitly excluded.

Assume the existence of a meaningful regularization subspace that is spanned by orthogonal basis vectors represented by the columns of the matrix $\mathbf{U} \in \mathbb{R}^{N \times P}$ where typically $P \ll N$. A subspace projection regularized optimal filter $\mathbf{f}$ is obtained through the following steps:

1. Construct the training data $\mathbf{x}'[k] = \mathbf{U}^T \mathbf{x}[k]$ by projecting the original training data onto the regularization subspace.
2. Design an optimal filter $\mathbf{f}' \in \mathbb{R}^P$ by training on $\mathbf{x}'$, while setting $C = 0$.
3. The optimal filter $\mathbf{f} = \mathbf{U}\mathbf{f}'$ is retrieved through the backprojection into the original data space.

Consider for example the matched filter design problem (5.9), which can be reformulated as a subspace projection regularized matched filter design problem through the application of the above steps, as follows:

$$\begin{aligned} \mathbf{f}_i^{\mathrm{MF}} = \mathbf{U} \operatorname*{argmin}_{\mathbf{f}'} \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} ({\mathbf{f}'}^T \mathbf{U}^T \mathbf{x}[k])^2 \\ \text{subject to} \quad ({\mathbf{f}'}^T \mathbf{U}^T \boldsymbol{\pi}_i)^2 = K \end{aligned} . \tag{5.24}$$

Besides the advantage of having explicit control over the solution subspace, another advantage of the subspace projection regularization, is that the optimization problem is formulated in a P-dimensional space, which leads to a reduction in computational effort given that $P \ll N$.

In Chapter 3, the following subspace design is proposed for the regularization of matched filters in the context of spike sorting. The first dimension in the subspace is explicitly set to the target pattern:

$$\mathbf{u}_{\boldsymbol{\pi}_i} = \boldsymbol{\pi}_i / \left\| \boldsymbol{\pi}_i \right\|_2 . \tag{5.25}$$

Having the target pattern explicitly in the subspace is important to assure that the resulting filter can respond to target pattern occurrences. $\mathbf{U}$ is then expanded with $P-1$ principal components of the recording data after removing the direction $\mathbf{u}_{\boldsymbol{\pi}_i}$ from the data to guarantee that $\mathbf{U}$ describes an orthogonal basis. The dimensionality of the subspace is controlled by capturing a predetermined fraction of the total signal power in the recording. Note that the use of this subspace performs an implicit denoising of the data prior to the filter design. We refer to Chapter 3 and [132] for more details.

Another advantage of this specific subspace projection regularization is that the structure of the subspace transforms the convex SPIR-optimal filter design

problem into an unconstrained problem. If the filter is designed in this specific subspace, the constraint is reduced to an equation in a single unknown, and can be easily satisfied as follows:

$$\mathbf{f'}^T \mathbf{U}^T \boldsymbol{\pi}_i = f'_1 \|\boldsymbol{\pi}_i\|_2 = \sqrt{K} \iff f'_1 = \frac{\sqrt{K}}{\|\boldsymbol{\pi}_i\|_2}, \tag{5.26}$$

where f'_1 represents the first element of the filter vector $\mathbf{f'}$ and is completely determined by the equality constraint. As such we have one degree of freedom less in the optimization problem. Consider $\mathbf{f'} = \begin{bmatrix} f'_1 & \mathbf{g}^T \end{bmatrix}^T$ with $\mathbf{g} \in \mathbb{R}^{P-1}$. The convex SPIR-optimal filter design problem, e.g., (5.18), can then be written as

$$\mathbf{f}_i^{\text{CSO"}} = f'_1 \mathbf{u}_{\boldsymbol{\pi}_i} + \mathbf{U}_{-\boldsymbol{\pi}_i} \mathbf{g}_i^{\text{CSO"}}, \tag{5.27}$$

with

$$\mathbf{g}_i^{\text{CSO"}} = \underset{\mathbf{g}}{\operatorname{argmin}} \sum_k \left(r \left(\mathbf{g}^T \mathbf{U}_{-\boldsymbol{\pi}_i}^T \mathbf{x}_k + f'_1 \mathbf{u}_{\boldsymbol{\pi}_i}^T \mathbf{x}_k - \sqrt{\gamma K} \right) \right)^2, \tag{5.28}$$

Where $\mathbf{U}_{-\boldsymbol{\pi}_i} \in \mathbb{R}^{N \times (P-1)}$ is the regularization subspace minus the target pattern direction. Note that the filter design problem in (5.28) is a convex unconstrained smooth optimization problem.

5.7 Case study: max-SPIR filters for neural spike sorting

In this section we will compare the novel SPIR-optimal filter design methods proposed in Section 5.4 and 5.5 and compare them to the matched filter in a filter-and-threshold-based multi-pattern recognition task, namely neural spike sorting [33]. Spike sorting is the process of detecting action potentials or 'spikes' generated by neurons in an extracellular multi-channel brain recording and assigning each spike to its putative source neuron. In a typical spike sorting pipeline [137], the algorithm can be divided in two stages: first a clustering algorithm is applied to all detected spike waveforms, where the cluster centroids serve as templates. In the second stage those templates are used to design matched filters to detect spikes from single neurons while resolving spike overlap. This second stage can be implemented as a filter-and-threshold-based spike sorting where the spikes from non-target neurons are treated as interfering

pattern sources. Filter-and-threshold-based sorting was proposed in Chapter 2 as a candidate for on-probe spike sorting, because of its low computational complexity. Closely related to the neural spike sorting problem is the surface electromyography (sEMG) decomposition problem [76] [47] [34]. In the field of sEMG decomposition, the goal is to assign recorded motor-unit action potentials to their putative motor units.

In this analysis we will only focus on the filter-and-threshold-based spike sorting stage, i.e., we assume that we have already identified a spike template of interest $\boldsymbol{\pi}_i$ for which we will design a filter-and-threshold-based pattern recognition filter. We use an extracellular ground truth data set, i.e., a neural recording for which the spike times of a fraction of the neurons are known. Because of this knowledge, the spike templates of the ground truth neurons can be trivially retrieved, by averaging over all the spikes of a specific neuron (in practice such ground truth is not available, in which case $\boldsymbol{\pi}_i$ should be computed as a cluster centroid during the initial spike clustering phase). The ground truth recording is generated by the SHYBRID open-source tool introduced in Chapter 4 and [133] in combination with SpikeInterface [13] providing access to a Neuropixel probe [54] dataset with 374 recording channels that is publicly available [114]. Note that the public dataset contains an extracellular recording that has been already preprocessed. The preprocessing consists of a common average referencing and high-pass filtering with a cut-off frequency of 300 Hz. These preprocessing steps are a standard practice in spike sorting.

The subset of channels that are relevant to neuron i are collected in the N_i-channel signal $\mathbf{r}_i[k] \in \mathbb{R}^{N_i}$ with N_i the number of relevant electrodes for sorting the neuron i. Note that the spike pattern is spatially sparse, such that we can ignore most of the 374 recording channels. Here we define the spatial region of interest by considering all electrodes within a 100 µm radius of the electrode with the largest absolute peak in the template $\boldsymbol{\pi}_i$. Because the spike template is a spatio-temporal pattern, rather than a spatial pattern, we define a spatio-temporal delay line $\mathbf{x}[k] = [\mathbf{r}_i[k]^T \ \mathbf{r_i}[k-1]^T \ \dots \ \mathbf{r}_i[k-L+1]^T]^T$ such that $N = LN_i$ with $L = 30$ corresponding to a 1 ms temporal window, given that the sampling frequency $f_s = 30$ kHz. For every neuron we consider one minute of recordings.

$\mathbf{f}_i^{\mathrm{MF}}$ as in (5.9)	
precision	recall
1.000	1.000
1.000	0.997
0.977	1.000
0.988	0.994
0.955	0.967
0.909	1.000
1.000	0.998
0.995	0.991
0.999	0.996
1.000	1.000
0.912	0.974
1.000	0.989
0.910	0.862
0.998	0.996
0.979	0.954
0.999	0.987
0.967	0.974
1.000	1.000
0.976	0.804
1.000	1.000
0.950	0.987
0.937	0.982
1.000	1.000
1.000	1.000
0.999	0.989
0.958	1.000
0.954	0.994
0.982	1.000
0.998	1.000
0.977 ± 0.03	**0.981 ± 0.043**

$\mathbf{f}_i^{\mathrm{SO}}$ as in (5.12)	
precision	recall
1.000	1.000
1.000	0.994
0.989	1.000
0.994	1.000
0.948	0.985
1.000	1.000
1.000	0.998
0.987	1.000
1.000	0.996
1.000	1.000
0.951	0.991
1.000	0.989
0.928	0.882
0.993	0.996
0.979	0.981
1.000	0.987
0.982	0.984
1.000	1.000
1.000	0.941
1.000	1.000
0.975	0.983
0.949	0.990
1.000	1.000
1.000	1.000
1.000	0.991
0.958	1.000
0.954	0.995
0.982	1.000
1.000	1.000
0.985 ± 0.021	**0.989 ± 0.024**

$\mathbf{f}_i^{\mathrm{CSO'}}$ as in (5.16)	
precision	recall
1.000	1.000
1.000	0.988
0.989	1.000
0.994	1.000
0.953	0.982
1.000	1.000
1.000	0.995
0.987	1.000
1.000	0.995
1.000	1.000
0.966	0.983
1.000	0.989
0.876	0.932
0.991	0.996
0.992	0.989
1.000	0.986
0.989	0.981
1.000	1.000
1.000	0.980
1.000	1.000
0.966	0.990
0.968	0.982
1.000	1.000
0.996	1.000
0.999	0.992
0.958	1.000
0.958	0.994
0.982	1.000
1.000	1.000
0.985 ± 0.026	**0.992 ± 0.013**

$\mathbf{f}_i^{\mathrm{CSO''}}$ as in (5.18)	
precision	recall
1.000	1.000
1.000	0.982
0.977	1.000
0.994	1.000
0.920	0.928
1.000	1.000
0.990	0.954
1.000	0.986
0.925	0.851
1.000	0.993
0.935	0.983
1.000	0.989
0.899	0.821
0.969	0.971
0.981	0.971
0.959	0.912
0.929	0.916
1.000	1.000
1.000	1.000
1.000	1.000
0.885	0.907
0.975	0.886
1.000	1.000
0.968	0.996
0.992	0.985
1.000	1.000
0.911	0.944
0.972	0.995
0.988	0.992
0.971 ± 0.035	**0.964 ± 0.049**

Table 5.1: Filter-and-threshold-based spike sorting performance in terms of precision and recall for the ground-truth neurons in the non-interfering group. Every row corresponds to a neuron, whereas the grouped pairs of columns correspond to the different filter design methods presented in this chapter.

$\mathbf{f}_i^{\mathrm{MF}}$ as in (5.9)	
precision	recall
0.778	1.000
0.848	0.921
0.556	**0.536**
0.258	0.948
0.022	1.000
0.819	0.984
0.406	**0.457**
0.684	0.720
0.711	0.873
0.788	0.766
0.564	0.663
0.522	1.000
0.180	0.821
0.833	0.992
0.529	**0.558**
0.595	0.995
0.593	0.981
0.488	**0.494**
0.667	0.923
0.677	**0.608**
0.399	0.821
0.567 ± 0.220	**0.812 ± 0.190**

$\mathbf{f}_i^{\mathrm{SO}}$ as in (5.12)	
precision	recall
0.778	1.000
0.935	0.946
0.889	0.857
0.376	0.914
0.023	1.000
0.829	0.969
0.451	**0.455**
0.797	0.733
0.988	1.000
0.893	0.771
0.762	**0.560**
0.522	1.000
0.321	**0.607**
0.833	0.992
0.671	0.777
0.596	0.992
0.731	0.823
0.532	**0.527**
0.732	0.888
0.696	0.663
0.455	0.911
0.658 ± 0.238	**0.828 ± 0.176**

$\mathbf{f}_i^{\mathrm{CSO'}}$ as in (5.16)	
precision	recall
0.778	1.000
0.966	0.943
0.958	0.821
0.949	0.966
0.050	1.000
0.828	0.966
0.430	**0.553**
0.814	0.933
1.000	1.000
0.823	0.843
0.926	**0.602**
0.522	1.000
0.852	0.821
0.833	0.992
0.767	0.770
0.812	0.785
0.801	0.919
0.615	**0.527**
0.827	0.853
0.735	0.660
0.557	0.842
0.754 ± 0.220	**0.847 ± 0.152**

$\mathbf{f}_i^{\mathrm{CSO''}}$ as in (5.18)	
precision	recall
1.000	1.000
0.873	0.728
0.893	0.893
1.000	1.000
1.000	1.000
0.763	0.831
0.699	**0.501**
0.934	0.947
0.975	1.000
0.854	0.848
0.812	0.651
0.522	1.000
1.000	1.000
0.977	0.992
0.751	0.867
0.856	0.847
0.912	0.848
0.770	**0.531**
0.832	0.785
0.802	0.733
0.832	0.781
0.860 ± 0.121	**0.847 ± 0.153**

Table 5.2: Filter-and-threshold-based spike sorting performance in terms of precision and recall for the ground-truth neurons in the interfering group. Every row corresponds to a neuron, whereas the grouped pairs of columns correspond to the different filter design methods presented in this chapter. The performance metric values shown in red correspond to values lower than 65%, whereas the values shown in green correspond to values higher than 80%.

The ground truth recording that is used in this work contains the full ground truth spike information of 50 neurons. For each of these 50 neurons, we design four filters (the matched filter $\mathbf{f}_i^{\text{MF}}$ (5.9), the non-convex SPIR-optimal filter $\mathbf{f}_i^{\text{SO}}$ (5.12) and the convex SPIR-optimal filters $\mathbf{f}_i^{\text{CSO'}}$ (5.16) and $\mathbf{f}_i^{\text{CSO"}}$ (5.18)), and compare their filter-and-threshold-based spike sorting performance.

We use the regularization subspace design from Chapter 3, with a subspace that contains the template $\boldsymbol{\pi}_i$ and that captures 90% of the total signal power (further on we will analyse the influence of this factor). The initial value for γ is chosen such that the initial desired SPIR of the filters corresponds to 10 dB. In order to ensure a sufficient amount of effective training samples, γ is dynamically changed during the filter design procedure. If less than 5000 interference threshold crossing are attained at the output of a filter, the filter design is iteratively repeated with a decreased γ. This practice is meant to ensure a well-conditioned training for cases in which there is little or no interference. Note that by decreasing γ, the SPIR-optimal filter converges towards the matched filter in case of little or no interference. The number of threshold crossings of 5000 corresponds to an average minimum of 25 effective training samples per degree of freedom in the filter design.

We quantify the filter-and-threshold-based spike sorting performance in terms of both precision and recall, which are defined in the following way:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}, \tag{5.29}$$

and

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{5.30}$$

For every neuron i and for each filter individually, we select the threshold T_i that leads to the maximal F_1-score [118], where F_1 is defined as follows:

$$\text{F}_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{5.31}$$

In other words we are comparing the best possible performances that can be obtained for every filter in terms of F_1. Prior to comparing the different filter design methods, we split the neurons in two different groups based on the precision of the matched filter solution. Neurons with a precision > 0.9 are assumed to not be affected by peak interferers (29 neurons), whereas those with a precision ≤ 0.9 are expected to be more affected (21 neurons). We thus expect SPIR filters to show a larger benefit in the second ('interfering') group, while achieving similar performance as a matched filter in the first ('non-interfering' group).

The threshold-based spike sorting performances for the non-interfering and interfering group are summarized in Table 5.1 and 5.2, respectively. We don't consider the heuristic SPIR-based filter design results in this comparison, because it performs poorly on the complex spike sorting data considered here with an overall average recall of 53% and a precision of only 40% (without ad-hoc

iteration). The SPIR-based filtering fails due to the fundamental limitations discussed in Section 5.3.3. We performed Wilcoxon signed-rank tests with $\alpha = 0.05$ to test for statistically significant performance differences between the different methods. A Bonferroni correction is applied to correct both for the six pairwise comparisons (between the different filter types) and two metrics (recall and precision), resulting in a significance level of $p < \frac{\alpha}{6 \cdot 2} = 0.00417$.

For the non-interfering neurons, we do not expect a benefit from using SPIR-optimal filters as the non-interfering neurons are selected as neurons on which the matched filter performs well. A statistical analysis shows that the SPIR-optimal filters can also provide satisfying results for such neurons. Indeed, comparing the matched filter $\mathbf{f}_i^{\mathrm{MF}}$ recall (0.981 ± 0.043 (average $\pm$ standard deviation)) with non-convex SPIR-optimal filter $\mathbf{f}_i^{\mathrm{SO}}$ recall (0.989 ± 0.024) results in p=0.00763, with convex SPIR-optimal filter $\mathbf{f}_i^{\mathrm{CSO'}}$ recall (0.992 ± 0.013) results in p=0.0219, and with convex SPIR-optimal filter $\mathbf{f}_i^{\mathrm{CSO''}}$ recall (0.964 ± 0.049) results in p=0.0129, such that we can state that no statistically significant differences are found in terms of recall for the above comparisons. When comparing the recall between the different proposed SPIR-optimal filter design methods, i.e, $\mathbf{f}_i^{\mathrm{SO}}$ and $\mathbf{f}_i^{\mathrm{CSO'}}$ with p=0.875, $\mathbf{f}_i^{\mathrm{SO}}$ and $\mathbf{f}_i^{\mathrm{CSO''}}$ with p=0.000681, and, $\mathbf{f}_i^{\mathrm{CSO'}}$ and $\mathbf{f}_i^{\mathrm{CSO''}}$ with p=0.000538, a significant difference in recall is identified between the usage of $\mathbf{f}_i^{\mathrm{SO}}$ and $\mathbf{f}_i^{\mathrm{CSO''}}$ on the one hand, and $\mathbf{f}_i^{\mathrm{CSO'}}$ and $\mathbf{f}_i^{\mathrm{CSO''}}$ on the other hand, twice in disadvantage of $\mathbf{f}_i^{\mathrm{CSO''}}$. In terms of precision no significant differences are found between the pairwise combinations of filter design methods. For reasons of clarity, we will not elaborate on the quantitative details of the precision analysis for the non-interfering group. The interested reader can find the performance metrics and their averages in Table 5.1. For the non-interfering examples we can thus conclude that there is no significant difference between the methods, except for a significant difference between the recall when comparing between the SPIR-optimal filter design methods. We argue that because there is no harmful interference in these examples, $\mathbf{f}_i^{\mathrm{CSO''}}$ can not benefit from neglecting negative sources during the filter design, such that it can only suffer from potential destructive interference leading to the observed significant decrease in recall. Note that the practical implication of this significance is limited. The average difference in recall when comparing those methods is only around 2.5 %. Based on the results presented in this paragraph, we can conclude that the use of SPIR-optimal filters is a viable alternative to the classical matched filter for the group of non-interfering neurons.

We now turn our attention to the interfering neurons, where by design we expect to see an improvement between the precision of the matched filter and of the SPIR-optimal filters. Comparing the matched filter $\mathbf{f}_i^{\mathrm{MF}}$ precision (0.567 $\pm$ 0.220), with non-convex SPIR-optimal filter $\mathbf{f}_i^{\mathrm{SO}}$ precision (0.658 ± 0.238) results in p=0.000196, with convex SPIR-optimal filter $\mathbf{f}_i^{\mathrm{CSO'}}$ precision (0.754

$\pm$ 0.220) results in p=0.000196, and with convex SPIR-optimal filter $\mathbf{f}_i^{\text{CSO''}}$ precision (0.860 $\pm$ 0.121) results in p=0.000120, such that we see that the anticipated effect is statistically backed. When comparing between the SPIR-optimal filter design methods, $\mathbf{f}_i^{\text{SO}}$ and $\mathbf{f}_i^{\text{CSO'}}$ with p=0.00185, $\mathbf{f}_i^{\text{SO}}$ and $\mathbf{f}_i^{\text{CSO''}}$ with p=0.000892, and, $\mathbf{f}_i^{\text{CSO'}}$ and $\mathbf{f}_i^{\text{CSO''}}$ with p=0.0187, the convex SPIR filters are shown to outperform the non-convex SPIR solution. When performing the pairwise comparisons for the recall, no significant effects are noted (again we omit the quantitative details, and refer to Table 5.2). Hence, the SPIR-optimal filters can obtain a significantly higher precision, while preserving the recall. The average improvement in precision between the different filter design methods in the order that they were presented in this work, is around 10 %, as can be seen from Table 5.2. When comparing the use of matched filtering in threshold-based spike sorting with the use of convex SPIR-optimal filtering $\mathbf{f}_i^{\text{CSO''}}$, an improvement of almost 30 % (in absolute sense) in average precision is noted. Besides the statistically significant results, the convex SPIR-optimal filters $\mathbf{f}_i^{\text{CSO''}}$ "transform" the threshold-based spike trains for 8 interfering neurons into single-unit spike trains, according to our arbitrary bound of precision > 0.9. This is an increase of 16% in detected single-unit spike trains for the entire data.

Next we will study the effect of the choice of regularization subspace power fraction on the spike sorting performance. The power fraction is the fraction of the total signal power that is accounted for in the regularization subspace, and as such, it serves as the regularization hyper parameter in this work. In the previous analysis we chose a power fraction of 0.9 (i.e., 90 %) because the resulting subspace is believed to provide a good trade-off between preserving and emphasizing the important signal characteristics, while providing a major reduction in dimensionality and computational complexity of the filter design. We can indeed verify from Figure 5.4 that the average spike sorting performance (both in terms of recall and precision) in the non-interfering group for a power fraction of 0.9 is near perfect, and approximately coincides with the spike sorting performance when using 0.95 or 0.97 as a regularization power fraction. When considering the interfering group one can see that for the recall a similar conclusion can be drawn, i.e., the difference in average recall between 0.9, 0.95 and 0.97 is only marginal. However, for the precision we see that by making use of a power fraction of 0.95 compared to 0.9 a couple of percent points could have been won. Also in terms of precision, we see that for a power fraction of 0.97 the regularizing effect starts breaking down again. This is to be expected because a higher power fraction can be interpreted as a lesser regularization, due to the fact that the solution subspace is less constrained. From Figure 5.4 it also becomes clear that the usage of a power fraction of 0.7 or 0.8 leads to reduced spike sorting performance in all cases. Purely based on spike sorting performance, 0.95 would be the better choice for use as a regularization power

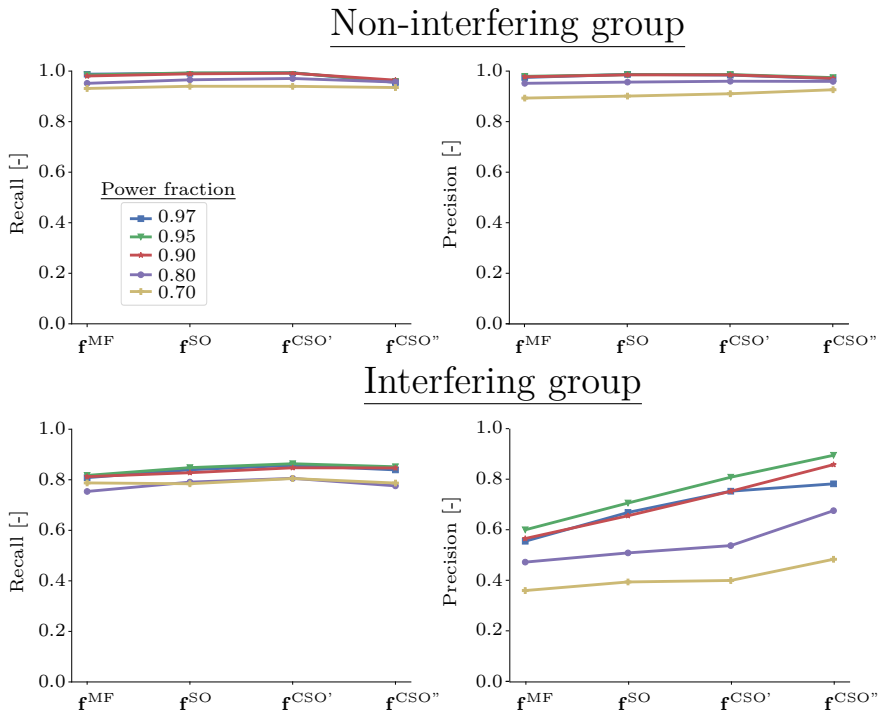

Figure 5.4: The effect on the spike sorting performance of the choice of regularization subspace power fraction is studied. The average precision and recall for all analysed filter design methods are shown for both the non-interfering (top) and interfering (bottom) group.

fraction. However, by also taking into account the relative dimensionality, i.e., the fraction between the dimensionality of the regularization subspace and the original problem subspace, we argue that the use of 0.9 is a reasonable practical choice. From Figure 5.5 one can see that a reduction of almost 70 % is obtained in terms of the dimensionality of the filter design subspace. By using a power fraction of 0.95 a reduction of only 45 % is obtained. This difference in relative dimensionality will have a big practical impact in terms of memory usage and computational time.

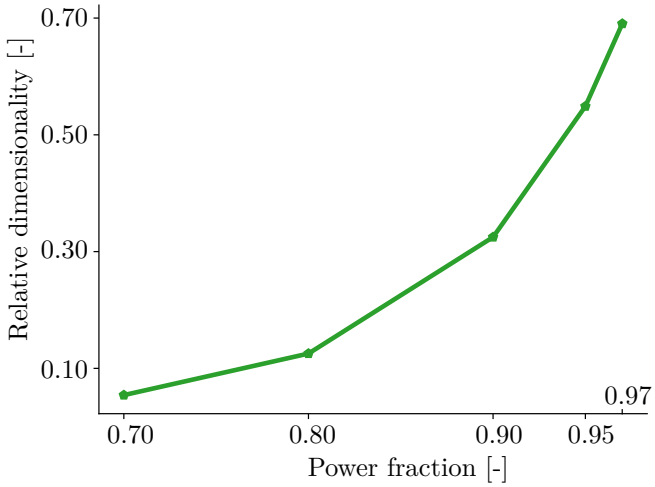

Figure 5.5: The relative dimensionality of the filter design subspace is shown in function of the regularization subspace power fraction.

5.8 Conclusion and Discussion

We have presented three novel filter design methods for use in filter-and-threshold-based multi-pattern recognition. The first SPIR-optimal filter design method was proposed to cope with the limitations of heuristic SPIR-optimal filter design methods. Two convex approximations of the first method were then proposed, which come with explicit guarantees on global optimality. The second convex reformulation was shown to be closely related to the theory of SVMs, and can be viewed as an extreme case of PU learning. Through this relationship with SVMs we concluded that this convex SPIR-optimal filter can be viewed as a maximum margin reformulation of the matched filter. Besides this conclusion, the relationship with SVMs provided a geometrical insight into the mechanics of the proposed filter design methods. Also, the proposed methods require no prior knowledge (except for the template of the target pattern), which make them particularly interesting for practical use, in which complete knowledge about the signal statistics is rarely available.

We have benchmarked the proposed filter design methods against the use of classical matched filters in the context of filter-and-threshold-based spike sorting. We were able to show that the proposed methods all outperform the matched filtering approach. The difference in precision between non-convex SPIR and convex SPIR-optimal filtering might be attributed to the discovery of a local rather than global optimal solution in the non-convex case. Another important

difference is that convex SPIR-optimal filters $\mathbf{f}_i^{\text{CSO''}}$ can also make use of the sign in the filter output, i.e., interferers that lead to a negative output response are harmless, whereas for a power optimization, these interferers might become dominant again. Whether or not the sign of the filter output is exploited could explain the difference in average precision between $\mathbf{f}_i^{\text{CSO'}}$ and $\mathbf{f}_i^{\text{CSO''}}$. The results from these experiments indicate that the convex SPIR-optimal filters are promising candidates for replacing the matched filter in filter-and-threshold-based pattern recognition tasks. Another advantage when comparing non-convex SPIR to convex SPIR optimization, is that convex SPIR optimization showed considerably faster convergence. During this case study we have also investigated the use of a subspace projection regularization approach, which has the ability to transform the convex SPIR-optimal filter design problem into an unconstrained optimization problem. Furthermore, because of the use of this regularization approach, both proposed filter design methods had a minimal set of interpretable hyper-parameters. Interpretable hyper-parameters are interesting in general, because they can be chosen in an informed manner, rather than a brute-force grid search type of experiment.

Although the proposed methods were shown to significantly improve the filter-and-threshold-based pattern recognition performance for spike sorting, the results showed that there is still room for improvement for many of the so-called interfering neurons. From an SVM point of view, we might conclude that for many of the interfering neurons, the target spike pattern and non-target spike patterns are not linearly separable. As already mentioned, the convex SPIR-optimal filter design method can be straightforwardly converted into a non-linear kernel method. Such a kernel SPIR-optimal filter has the potential to separate non-linearly separable data at the cost of an increase in computational complexity.

Chapter 6

Resolving spike overlap in the feature space

Part of this chapter has been published as *Jasper Wouters, Fabian Kloosterman, and Alexander Bertrand. "A neural network-based spike sorting feature map that resolves spike overlap in the feature space." 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain.* Jasper Wouters has co-developed the methodology, conducted the experiments and has written the manuscript. Compared to the publication, the manuscript has been extended with an improved model of spike overlap and experiments related to this improved model.

Many spike sorting pipelines rely on a clustering algorithm that groups the spikes coming from the same neuron in a pre-defined feature space. However, classical spike sorting algorithms fail when spike overlap, i.e., the near-simultaneous occurrence of two or more spikes from different neurons, is present in the recording. In such cases, the overlapping spikes segment ends up in a seemingly random position in the feature space and is not assigned to the correct cluster. This problem has been addressed before by extending the sorting algorithm with a template matching post-processor. In this chapter, a novel approach is presented to resolve spike overlap directly in the feature space. To this end, a neural network feature map is presented, that generates spike embeddings (feature vectors) that behave as a linear superposition in the feature space in the case of spike overlap. Its performance is quantified on semi-synthetic data obtained through a data augmentation procedure applied to real neural recordings.

6.1 Introduction

When studying the brain, an often used method to explore brain activity is the acquisition and processing of extracellular recordings. Extracellular recordings can be obtained from measurement electrodes that are lowered into the brain. If the implanted electrodes are in close proximity to neurons, the electrodes can pick up the spike waveforms generated by these neurons. To get a better understanding of micro-scale brain network dynamics, extracellular recordings are often processed to extract the spike times of individual neurons from which the spikes were recorded. This process of detecting and assigning spikes to their putative neurons is known as *spike sorting* [67].

Spike sorting is an unsupervised machine learning problem that is based on the assumption that all spikes from the same neuron exhibit the same spatio-temporal waveform, and that every neuron has a unique spatio-temporal spike pattern when compared to other neurons. The spike sorting problem is typically solved through the use of a classical clustering pipeline [33]. This pipeline roughly consists of a preprocessing, spike detection, feature extraction, and clustering phase. The preprocessing block minimally consists of a high-pass filter, that filters out the low-frequency energy that is unrelated to the spiking activity. The spike detection block then selects only recording segments that have spiking activity. Next, a feature vector is calculated for every spike segment, such that finally the clustering algorithm can isolate dense clusters of spikes that are believed to belong to the same neuron.

Such a clustering approach typically suffers from the overlapping spikes problem [90]. This problem arises when a detected spike segment consists of the activity of multiple neurons. Typically, such segments containing overlapping spikes will lead to points in the feature space, that are unrelated to the clusters that represent the well isolated spikes of the neurons that contribute to the overlapping spikes segment. They appear at seemingly random positions in the feature space, depending on the amount of overlap and the relative spike times of both neurons. It is believed that spike overlap can not be resolved in the feature space [16]. As such, the classical pipeline has been augmented with a template matching post-processing step in many spike sorting implementations [30] [31] [137] [131] [134]. Alternatively, iterative model-based algorithms [90] [27] [86] have been developed to cope with spike overlap. Typically, in such extended pipelines, first the clustering approach is taken, after which heuristics are used to identify clusters that contain the activity of individual neurons. Next, spike templates are estimated from those single-neuron clusters. Those templates are then used to design matched or discriminative filters that have the capability to resolve overlapping spikes. Augmenting the classical pipeline in such a way, leads to a considerable increase in complexity.

In this chapter we present a neural network-based [119] feature map that, contrary to popular belief, is capable of resolving spike overlap directly in the feature space, hence, resulting in a pipeline with overall reduced computational complexity. In Section 6.2 the overlapping spikes problem is formalized. Section 6.3 presents the neural network feature map capable of resolving overlapping spikes for an initial simplified data model. In Section 6.4 a data augmentation procedure that is used for the generation of training and testing data is presented. Section 6.5 summarizes the experimental results for the initial simplified data model. In section 6.6 we train and validate a feature map using an improved data model that accounts for within-cluster scaling, distortion and sample-time jitter. Finally, Section 6.7 concludes this chapter by discussing the proposed methods.

6.2 Problem statement

Consider the high-pass filtered extracellular multi-channel brain recording $\mathbf{x}[k] \in \mathbb{R}^N$ at sample time k with N the number of recording sites of interest. Typically, only compact subsets of recording sites are processed simultaneously in spike sorting because the neuronal spike is a spatially compact pattern. For modern probes $N \ll N_{\text{tot}}$, where N_{tot} is the total number of recording sites of the recording device. A spatio-temporal recording snippet of duration L containing an entire neuronal spike waveform can be represented as an NL-dimensional vector $\bar{\mathbf{x}}[k] = \left[\mathbf{x}\,[k]^T \;\; \mathbf{x}\,[k-1]^T \;\; \ldots \;\; \mathbf{x}\,[k-L+1]^T\right]^T$, which captures a spike waveform between sample time $k-L+1$ and k. After spike detection, we store the values of $\bar{\mathbf{x}}[k]$ at those sample indices k that correspond to a spiking event, where k is selected such that the largest peak of each spiking event is aligned across the extracted snippets $\bar{\mathbf{x}}[k]$. This results in a set of M extracted snippets represented as NL-dimensional vectors $\left\{\bar{\mathbf{x}}^1 \ldots \bar{\mathbf{x}}^M\right\}$, each containing an aligned spatio-temporal representation of a spike waveform. Since each neuron has a signature spatio-temporal waveform, the snippets $\bar{\mathbf{x}}$ that capture spikes from the same neuron i will be close to each other in the $\mathbb{R}^{NL}$ space and their average can be viewed as a spike template for neuron i. In the remaining of this chapter we will mostly drop the time index k in $\bar{\mathbf{x}}[k]$.

During spike sorting, the aligned snippets are mapped to a lower-dimensional feature space to accommodate the clustering stage. Consider the feature extraction $f : \mathbb{R}^{NL} \to \mathbb{R}^P$, with P the dimensionality of the feature space. In many spike sorting pipelines f is a linear function, e.g., based on principal component analysis on all aligned spike snippets. The implication of such a linear feature map is that $f(\mathbf{s}_i + \mathbf{s}_j) = f(\mathbf{s}_i) + f(\mathbf{s}_j)$ for any $\mathbf{s}_i, \mathbf{s}_j \in \mathbb{R}^{NL}$. Let $\mathbf{s}_i$

and $\mathbf{s}_j$ denote the spike waveform of neuron i and neuron j, respectively. When using a linear feature map, it is expected that overlapping spike snippets of the form $\bar{\mathbf{x}} = \mathbf{s}_i + \mathbf{s}_j$ would form a new cluster of which the centroid is equal to the sum of the centroids of the clusters associated to neuron i and j (cfr. the red, blue and purple cluster in the right panel of Figure 6.1). If this were to be the case, it would be straightforward to resolve spike overlap. Unfortunately, this is not the case in practice as this requires the main peaks of the spikes $\mathbf{s}_i$ and $\mathbf{s}_j$ to be perfectly aligned in time (remember that all snippets are aligned through their maximum peak value).

In reality, spike overlap can be better modelled as $\bar{\mathbf{x}}[k] = \mathbf{s}_i[k-m]+\mathbf{s}_j[k-n]$ with $m, n \in \mathbb{Z}$ being probabilistic sample offsets, and where $\mathbf{s}_i[k]$ and $\mathbf{s}_j[k]$ correspond to the *mutually aligned* isolated spike snippets of neuron i and j, respectively. Next we introduce the operator $\oplus$ to denote a sum with probabilistic sample offsets, which abstracts away the actual values for m and n. The shifted sum can now be rewritten as

$$\mathbf{s}_i \oplus \mathbf{s}_j = \mathbf{s}_i[k-m] + \mathbf{s}_j[k-n]. \tag{6.1}$$

Because of the introduction of the time shift, $f(\mathbf{s}_i \oplus \mathbf{s}_j) \neq f(\mathbf{s}_i) + f(\mathbf{s}_j)$ for the linear feature map f, if at least one of the two sample offset is non-zero, thereby rendering the proposed approach for the identification of overlapping spikes clusters in the feature space useless as shown in the left panel of Figure 6.1. In this chapter we aim to design a non-linear feature map g for which $g(\mathbf{s}_i \oplus \mathbf{s}_j) \approx g(\mathbf{s}_i) + g(\mathbf{s}_j)$, i.e., the linearity property is preserved despite the random sample offsets in the overlapping spike. Such a feature map g would be capable of resolving spike overlap, because the relation between clusters can be derived by simply adding pairs of cluster centroids, as is shown for the cluster centroids of neuron one and neuron two in the right panel of Figure 6.1, where those cluster centroids sum up to the cluster containing randomly shifted overlapping spikes of neuron one and neuron two.

6.3 Method

To obtain the desired feature behaviour, we train a neural network using the following custom cost function:

$$\sum_{\substack{i,j \in \mathcal{T} \\ i \neq j}} \left[\|g(\mathbf{s}_i) + g(\mathbf{s}_j) - g(\mathbf{s}_i \oplus \mathbf{s}_j)\|_2^2 + \frac{\kappa}{\|g(\mathbf{s}_i) - g(\mathbf{s}_j)\|_2^2} \right], \tag{6.2}$$

where the sum is taken over all pairs of spikes (both corresponding to different neurons) that are in the training set $\mathcal{T}$. When minimizing the cost function

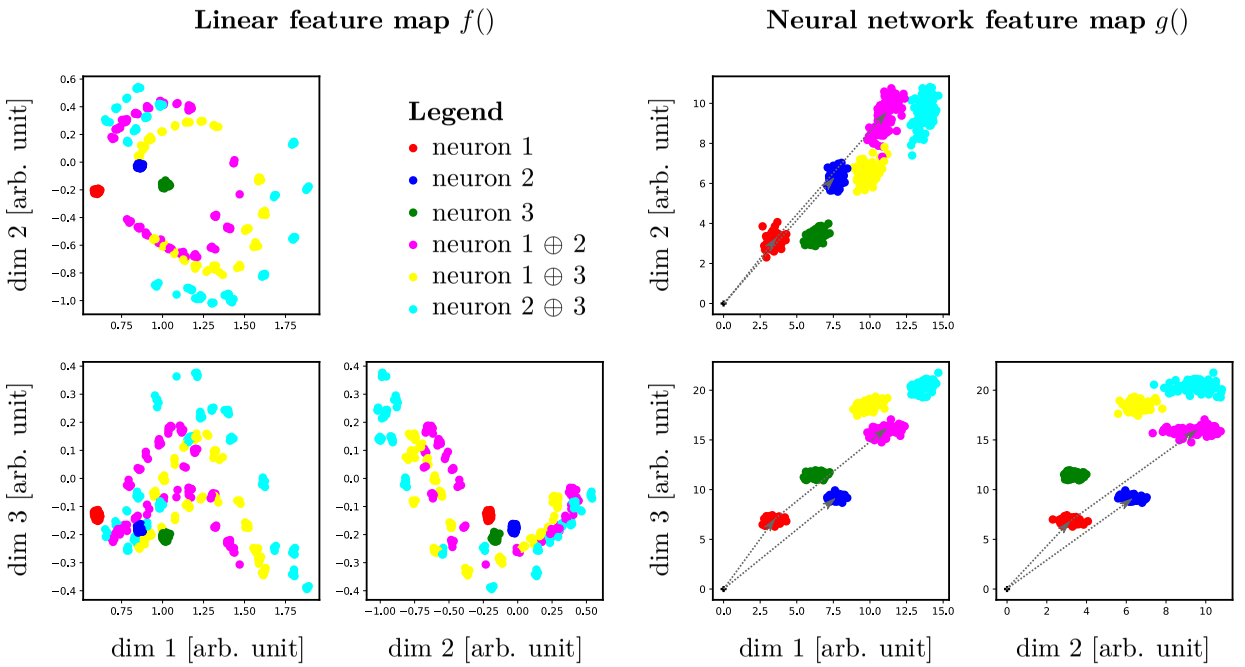

Figure 6.1: 3 orthogonal viewpoints on a 3-dimensional feature space with 3 neurons. **Left**: Overlapping spikes in a (linear) principal component feature map do not form separable clusters and do not have a clear relationship with the single-neuron clusters. **Right**: The proposed neural network feature map is capable of resolving overlapping spikes.

over g, the first term in the cost function will decrease when the feature acts as if it is a linear operator applied to a linear combination, i.e., the behaviour that is desired to resolve overlap in the feature space. The second term is added to try to increase the distance between spikes from different neurons, and hence their separability, in the feature space and this will as well prevent the network from converging to the trivial solution during training. The hyperparameter κ associated with the second term can be altered to trade off the desired linearity with the cluster separability.

Notice that the cost function in (6.2) does not explicitly control for the intra-class variability, which is an important characteristic for a feature map intended for clustering. In this chapter the intra-class variability is controlled through on the one hand limiting the degrees of freedom by choosing an appropriate neural network architecture and on the other hand by applying regularization during the training phase. In this chapter early stopping regularization is used [18], where thirty percent of the training data is used as a validation set for this purpose. If the validation cost does not decrease over a certain number of epochs, the training process is stopped to prevent the network from overfitting on the training data. We set the number of epochs associated with the early stopping equal to ten.

In Figure 6.2 the neural network architecture that is used in combination with (6.2) is shown. The architecture consists of a multi-layer perceptron (MLP) [119] with two hidden layers. The network takes an NL-dimensional[1] snippet $\bar{\mathbf{x}}$ as input, with $N = 10$ and $L = 30$. The first and second hidden layer consist of 600 and 60 hidden perceptrons, respectively. The network outputs a three dimensional feature vector for every data chunk at its input. All layers use a rectifier activation function.

6.4 Data augmentation

The training and testing data used in this chapter are obtained through a data augmentation procedure similar to [64]. First, 131 single-neuron spike templates are calculated from a publicly available in-vivo Neuropixels [54] brain recording data set that has been spike sorted [114]. For each of the 131 neurons, a spatio-temporal spike template is calculated by averaging over the spike snippets of that neuron where a neuron-specific spatial region of ten neighbouring channels is selected. Next, the single-neuron templates are randomly split into two sets,

[1] The choice of N and L is related to the spatial resolution of the electrode array and the sampling frequency, respectively. The dimensions of the proposed network were designed for a neural probe with an electrode pitch of 20 µm and a sampling frequency of 30 kHz.

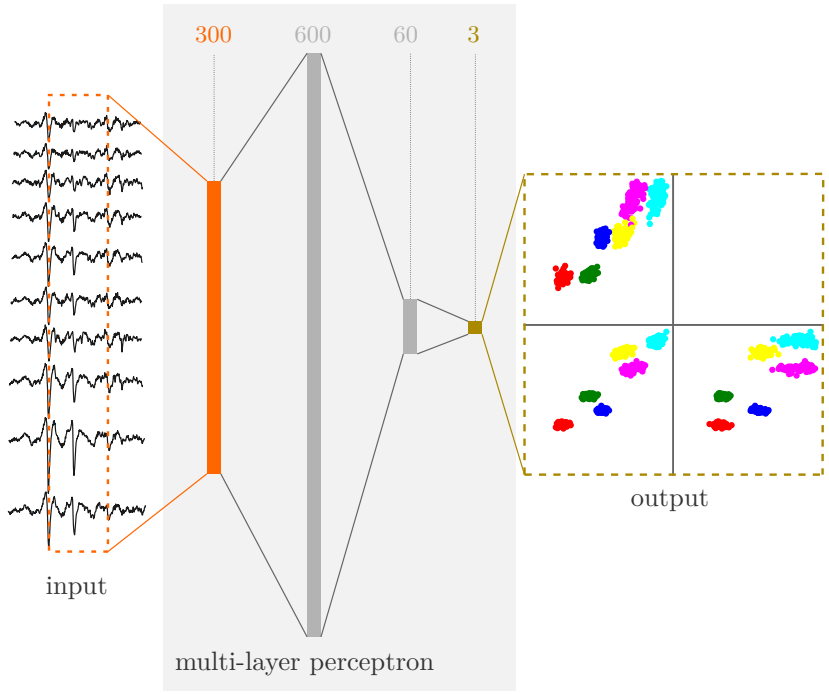

Figure 6.2: The proposed neural network feature map is implemented as a multi-layer perceptron with two hidden layers.

one containing 88 templates used for generating the training data and 43 used for generating the testing data. For both the set of training and testing templates separately, principal components are derived that account for 99 percent of the template energy. Both sets are then projected onto their respective principal component subspaces. Along each principal component an independent skew normal distribution [6] is fitted. Finally, synthetic training and testing spike templates are constructed by sampling from the estimated distributions in each principal component. The templates obtained through data augmentation have a duration of 82 samples.

6.5 Initial experiments

From the training distributions obtained in the previous section, 50 000 different pairs of neuronal spikes are generated to form the training set $\mathcal{T}$, i.e, the summation in (6.2) runs over 50 000 training samples. For each pair (i, j), 3

snippets are created: $\mathbf{s}_i$, $\mathbf{s}_j$ and $\mathbf{s}_i \oplus \mathbf{s}_j$. $\mathbf{s}_i$ and $\mathbf{s}_j$ are created by adding Gaussian white noise to the two synthetic spikes, aligning both of them according to their largest peak, and cropping them in the time dimension to the desired window size of $L = 30$ samples. $\mathbf{s}_i \oplus \mathbf{s}_j$ is obtained by randomly time-shifting the template of neuron i with respect to the template of neuron j, after which white noise is added and the summed waveforms are aligned based on the largest peak of the mixed waveform and cropped to $L = 30$ samples. The relative shift is sampled from a discrete uniform distribution where the minimum and maximum shift are -5 and 5, respectively. The synthetic snippets have an average signal-to-noise ratio of 20 dB. The network presented in Section 6.3 is trained using the Adam optimization algorithm [60]. The hyperparameter $\kappa = 10$ is chosen through a grid-search, but the performance on the training data is not very sensitive to the exact choice of κ, even when considering differences of several orders of magnitude.

After training, the performance on the test set of the neural network feature is quantified in terms of two metrics: 1. the adjusted Rand index [97], which is a cluster performance metric that is equal to one for perfect clustering and zero when the clustering performs equal to chance, and 2. the center prediction error

$$e_{i,j} = \frac{\|\mathbf{c}_{i\oplus j} - (\mathbf{c}_i + \mathbf{c}_j)\|_2}{\frac{1}{3}(\sigma_{i\oplus j} + \sigma_i + \sigma_j)}, \tag{6.3}$$

i.e., the distance in the feature space between the true overlapping spikes cluster centroid $\mathbf{c}_{i\oplus j} \in \mathbb{R}^P$ and the predicted centroid by taking the sum of the isolated cluster centroids $\mathbf{c}_i + \mathbf{c}_j$, normalized with respect to the intra-class variabilities $\sigma_{i\oplus j}$, σ_i and σ_j, which are the average distances between points in the feature space and their corresponding centroid.

The test data contains 500 pairs of synthetic test templates. For every pair of test templates two sets of 100 isolated spikes are derived by adding white noise to the templates and 100 snippets containing spike overlap with a random offset similar to the offset used in the training set are generated. The synthetic training snippets also have an average signal-to-noise ratio of 20 dB. The neural network feature map is then applied to every group of 300 snippets and a k-means clustering [4] with k = 3 is applied for all 500 test pairs. The performance metrics for all test cases are summarized in Figure 6.3. Because 483 pairs have an adjusted Rand index equal to one, random jitter is added to the horizontal axis to make the graph more readable (hence, explaining test pairs with a depicted Rand index slightly bigger than one). As one can see in Figure 6.3 the majority of the test pairs is situated in the bottom right corner. For those test pairs, unseen during training, the neural network feature map generalizes very well. The feature allows for near perfect clustering for those pairs, even using a very simple clustering method such as k-means. Secondly, only a small

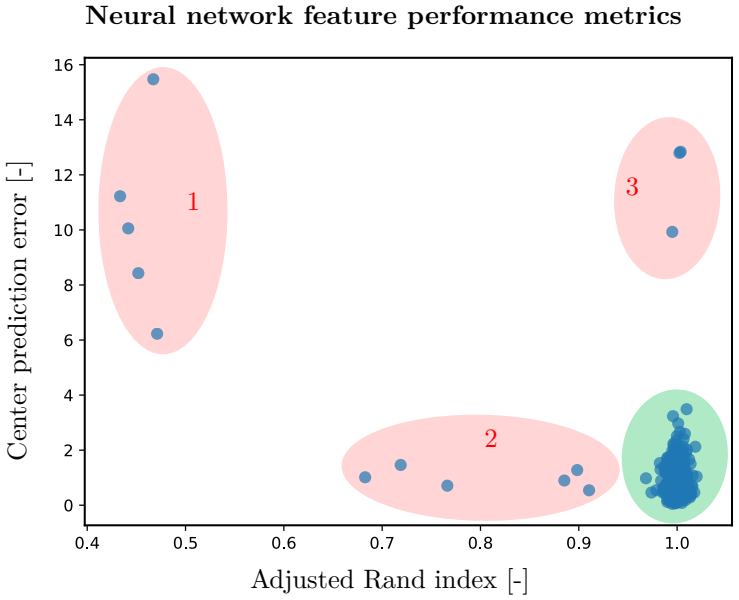

Figure 6.3: On the horizontal axis the adjusted Rand index (with random jitter for readability) is depicted and on the vertical axis the center prediction error is shown. Every blue dot represents the feature map performance for a pair of test neurons.

center prediction error is made, indicating the feature's capability to resolve spike overlap in the feature space.

In Figure 6.3 we also highlighted three groups of pairs for which the feature map performs suboptimal. For the pairs in *group 1* the single-neuron clusters are very close to each other in the feature space, combined with the fact that the cluster containing all overlapping spikes is elongated. Because of this, the single-neuron clusters are grouped together and the overlapping spikes cluster is split in two by the clustering algorithm, hence explaining both the poor Rand index and center prediction error for those pairs. This issue might be resolved by using a more advanced clustering approach, e.g., a density based clustering algorithm [102]. For the pairs in *group 2*, one of the single-neuron clusters is close to the overlapping spikes cluster, as such, spikes belonging to the overlapping spikes cluster are assigned to that single-neuron cluster by the clustering algorithm. Because only a minor fraction of pairs is wrongly assigned, the center prediction error is still within acceptable bounds. This issue might also be resolved by using an alternative clustering algorithm. The pairs in *group 3* show perfect clustering, but a big center prediction error. For

this small fraction of pairs, the spikes of one of the neurons are not capable of fully exciting the neural network. This behaviour is characterized in the feature space by having one of the single-neuron clusters with at least one dimension equal to zero. For the dimensions in the feature space that are not properly excited, the desired linear behaviour is not obtained.

6.6 Improved spike overlap model

During testing in the previous paragraph, the spikes from a certain neuron were modelled as the sum of their template and Gaussian white noise. This is a simplified model of single-unit spike waveforms. Furthermore, during training and testing, only overlap between spikes from different neurons that reach their maximum energy on the same channel was considered. In this section we consider an improved model of single-unit spike waveforms and spike overlap during both training and testing. The improved model during training is reflected by the following improved cost function:

$$\sum_{\substack{i,j\in\mathcal{T}\\ i\neq j}} \Bigg[\left\| g(\mathbf{s}_i) + g(\mathbf{s}_j) - g(\mathbf{s}_i \oplus \mathbf{s}_j) \right\|_2^2 + \kappa_1 \exp\left(- \left\| g(\mathbf{s}_i) - g(\mathbf{s}_j) \right\|_2^2 \right) + \kappa_2 \left(\left\| g(\mathbf{s}_i) - g(\mathbf{s}_i') \right\|_2^2 + \left\| g(\mathbf{s}_j) - g(\mathbf{s}_j') \right\|_2^2 \right) + \kappa_3 \left(\left\| g(\mathbf{s}_i) - g(\mathbf{s}_i^d) \right\|_2^2 + \left\| g(\mathbf{s}_j) - g(\mathbf{s}_j^d) \right\|_2^2 \right) \Bigg], \tag{6.4}$$

where κ_1, κ_2 and κ_3 are tuning parameters.

The first two terms in (6.4) have the same purpose as the terms in (6.2) with respect to the desired behaviour of the feature map. The second term in the improved cost function, i.e., the κ_1-term, is slightly adjusted compared to the initial cost. We opted for an exponential function rather than a reciprocal, because the exponential is bounded on the interval that is considered here, which leads to an improved training stability.

Considering that $\mathbf{s}_i'$ and $\mathbf{s}_j'$ are amplitude-scaled variations of $\mathbf{s}_i$ and $\mathbf{s}_j$, respectively, and each individual snippet contains independent additive noise, the κ_2-term exposes a more realistic model of a single-unit cluster to the training process. This improved model thus includes within-cluster amplitude scaling that is encountered in practice. As such, κ_2 can be seen as the hyper parameter that explicitly controls for the intra-class variability. The amplitude

scaling factors are sampled from a uniform random distribution between 0.8 and 1.2. These distribution bounds were chosen based on our experience with the generation of hybrid ground truth data.

$\mathbf{s}_i^d$ and $\mathbf{s}_j^d$ are distorted variations of $\mathbf{s}_i$ and $\mathbf{s}_j$, respectively, where each individual snippet contains independent additive noise. Every distorted spike snippet contains an additional spatio-temporally overlapping spike that reaches its maximum on a different channel. As such, the κ_3-term exposes overlap between spikes from different neurons that reach their maximum on different channels. The final term tries to shape the feature map such that it is robust with respect to such off-axis spike overlap.

As a consequence of the increase in complexity of the desired feature map behaviour, we extend the initial MLP with an additional layer between the first and second hidden layer of the initial architecture, resulting in a so-called *deep* neural network [119]. The dimensionality of the additional layer is set equal to the dimensionality of the input layer as shown in Figure 6.4. Furthermore, we replace the rectifier activation function in the output layer with a linear activation function. This choice of activation prevents the output from clipping negative values to zero, thereby effectively solving the *group 3* errors that were identified in Section 6.5.

We increase the size of the training data set to accommodate the deep architecture. To increase the available training data, we apply the data augmentation procedure from Section 6.4 to simulated templates [12] from biophysically detailed neuron models [96], using the same probe as in Section 6.4. We use a total of 634 neuron models of which one-third is reserved for the testing data augmentation. A total of 50 000 training templates are generated from which 1 000 000 training pairs (i, j) are sampled. Again, the training is regularized using early stopping regularization with a validation split of thirty percent.

The hyper parameters κ_1, κ_2 and κ_3 are explored during training through a grid-search experiment. The combination of parameters that leads to the best spike sorting performance and spike overlap resolution capability on a random sample of the training data is $\kappa_1 = 100$, $\kappa_2 = 1$ and $\kappa_3 = 0.1$. The network that results from training with these hyper parameters is then characterized on the test data set, similar to the characterization from Section 6.5. Additionally, here we perform a random scaling to every spike that is sampled from the same scaling distribution as is used during training. Then, a sub-sample time jitter is applied to every spike similar to the procedure described in Section 4.2 to model the randomness of the spike time with respect to the sample clock. Finally to every snippet (including the spike overlap snippets), a non-spatially aligned distorting spike is added with a five percent chance.

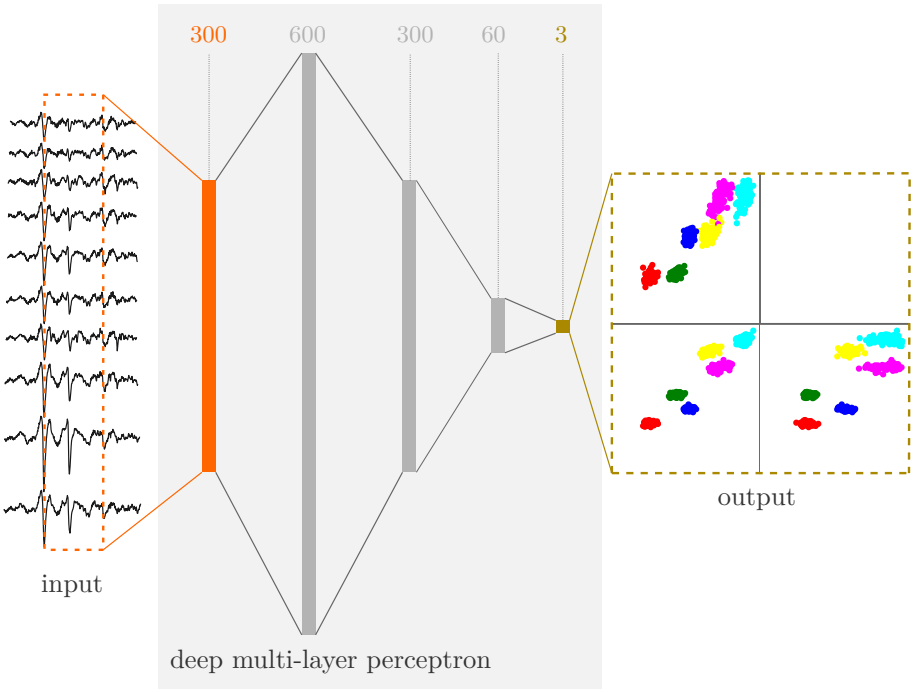

Figure 6.4: Extended architecture that is used in the context of the improved model.

The results from the characterization on 500 testing pairs are shown in Figure 6.5. In summary, the average adjusted Rand index is 0.951 and the average center prediction error is 0.410. One can see a reduction in *group 1* errors, which is likely due to the explicit κ_2-penalty that controls the intra-class variability and the shape of the resulting clusters. *Group 3* errors have also vanished due to the linear output activation. However, we see an increase in *group 2* errors, which is mainly due to a decreased isolation between clusters as a result of the more challenging model used for the generation of single-unit test clusters, i.e., including amplitude scaling, distortion and sub-sample time jitter.

When characterizing the performance of the feature map directly on the simulated templates from the test set, rather than on augmented templates derived from the simulated test templates as was done above, the average Rand index drops to 0.757 and the average center prediction error increases to 1.064. In contrast to the findings for the augmented test templates, here we note that the feature map does not generalize to the biophysically simulated templates. This observation indicates that the data augmentation procedure

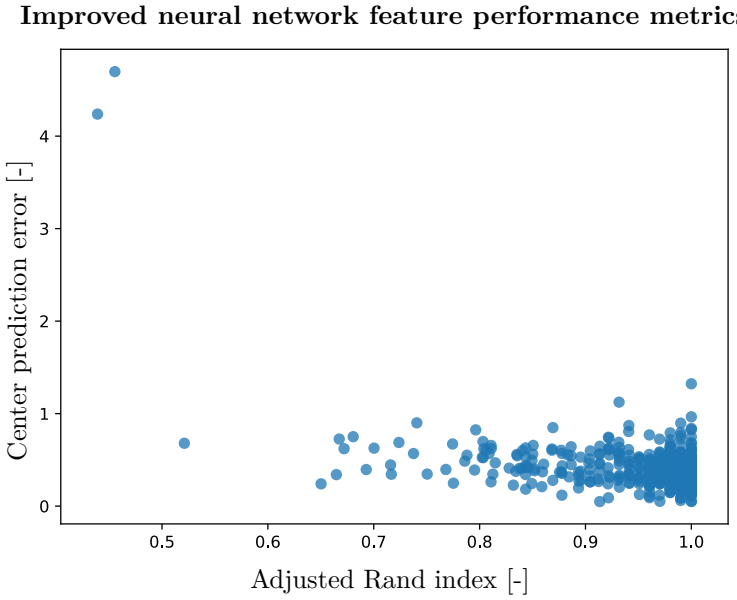

Figure 6.5: On the horizontal axis the adjusted Rand index is shown and on the vertical axis the center prediction error is shown. Every blue dot represents the feature map performance for a pair of test neurons.

is not sufficiently expressive to promote direct practical use of the proposed feature map.

6.7 Conclusion and discussion

In this chapter we have presented a novel approach for resolving spike overlap in the feature space through the design of a neural network feature map which was presented using a simplified model of overlap and single-unit clusters. Such a feature map can be straightforwardly plugged into a classical clustering pipeline. After performing the cluster analysis, overlapping spikes clusters can be identified as clusters of which the centroid is "close to" the vector addition of two other clusters. The overlapping spikes cluster is then believed to contain snippets that originate from the near-simultaneous action potential firing of two neurons that are associated with the two other clusters, thereby effectively resolving spike overlap. We have shown that the desired neural network behaviour for resolving spike overlap generalizes well on most of the augmented testing data. On a small fraction of the testing pairs, the feature map performed suboptimal. For

those pairs we have studied their failure mechanisms, and proposed solutions where possible. We then improved our data model and showed that the feature map that was trained on the improved model also generalizes on augmented test data.

However, when investigating the performance with respect to the non-augmented test data, we noticed that the spike sorting performance drops considerably. This is indicative of the limiting nature of the data augmentation procedure with respect to the practical use of the obtained feature map. Therefore, the primary focus of future research in this area should be on the development of data augmentation procedures that promote generalization towards biophysically detailed simulations and real recordings. Having such augmentation procedures available, the methods presented in this chapter can serve as a framework for the development and validation of feature maps capable of resolving spike overlap in the feature space.

Since the feature maps are trained using data from a specific type of probe, this feature map is believed to be probe dependent. Further research is also necessary to assess how well this feature map works on real recordings using the same probe, and whether it can be used with different recording probes as well.

When compared to template matching based approaches, our method is limited in terms of resolving the exact spike times of the individual neurons that fire near-simultaneously. For experiments where this limitation is acceptable, a much simpler spike sorting pipeline, capable of resolving spike overlap, is given in return. Combining the proposed feature map with an adaptive clustering algorithm is a promising and potentially powerful approach to real-time spike sorting, given the availability of an improved data augmentation procedure.

Chapter 7

Conclusion

7.1 Contributions

The state-of-the art techniques that are in use today for resolving spike overlap during spike sorting of extracellular neural recordings are based on iterative template matching procedures. Due to their iterative character, the processing delay is non-deterministic. Furthermore, the computational complexity is still relatively high and scales poorly with the number of templates due to interactions between the different templates that are involved in the matching procedure. For online spike sorting applications, e.g., in the context of closed-loop neuroscience, these iterative characteristics are to be avoided. To this extent, we focussed on the concept of *threshold-based spike sorting* in Chapter 2, which is a framework for spike sorting that is non-iterative and that does not involve any interaction between the different spike templates. The framework depends on the availability of discriminative linear filters that are derived from the spike templates. The use of linear filters ensures a low computational complexity, a low deterministic delay and a stable operation, hence, making the framework fit for online spike sorting, and potential embedded implementations thereof.

Although the idea of threshold-based spike sorting has been proposed prior to this thesis work, the lack of true discriminative template matching filters has prevented reliable use of the framework in practice. In Chapter 2 we proposed the novel signal-to-peak-interference ratio optimaltiy or *SPIR-optimality* criterion for the design of discriminative template matching filters. The optimality criterion is a natural expression of the desired qualitative threshold-based spike

sorting filter output behaviour for non-target signals. This behaviour comprises of a trade-off between baseline noise inflation and peak interferers suppression, where ideally this leads to a similar output power response to both baseline noise and peak interferers. Earlier work on discriminative template matching with suppression of interfering spikes is based on signal-to-interference-plus-noise ratio or SINR-optimality, which depends on unrealistic prior knowledge assumptions and many tunable hyper parameters in order to obtain the desired filter output behaviour as we pointed out in Section 5.3.2. Throughout this thesis we have developed the SPIR-optimal filter design methodology, among other contributions, to the point where it requires *minimal and realistic prior knowledge* and depends only on a minimal set of *interpretable hyper parameters* as was shown in Chapter 5.

In Chapter 2, we proposed a *heuristic scheme to identify peak interferers* for SPIR-optimal filter design. Although the feasibility of this scheme was shown in the context of spike sorting, we identified two limitations. A first limitation is the possibility of missing peak interferers due to the use of the so-called target safe zone. A second, more severe, limitation is that the heuristic works exclusively on the filter input, while being blind for the actual filter output, i.e., new interferers that were not accounted for during the design can become dominant after the filter design . An ad-hoc solution in the form of an iteration between the heuristic identification and filter computation during the filter design process was proposed, but this ad-hoc solution comes without convergence guarantees and requires expert intervention to terminate the filter design process, thereby hampering practical adoption.

Before tackling the limitations related to the heuristic SPIR-optimal filter design, we took a step back and turned our attention to the lesser studied, yet more general problem of the regularization of matched filters for use in spike sorting in Chapter 3. We identified the need for a regularization as a mechanism to cope with ill-conditioned covariance estimates during the filter design, which is an increasingly relevant problem in the context of high-density neural probes. Having gained a better understanding of the problem, we proposed a *data-driven regularization approach with an interpretable hyper parameter*. The approach consists of a regularization subspace projection followed by the design of the matched filter in a lower-dimensional space, which leads to a *computational advantage* when compared to, e.g., a diagonal loading. Besides a computational advantage, the regularization approach was shown to also improve the threshold-based spike sorting performance.

Another important realization from Chapter 3 is the recommendation to *include spiking activity segments into the noise covariance estimate*, as it can be shown that leakage of the target template into the noise covariance matrix causes no harm. An important consequence of this realization is that the target safe zone,

as introduced in Chapter 2, is not strictly necessary and can be ignored, thereby solving the issues related to the use of such a target safe zone as discussed earlier.

Prior to tackling the second limitation related to the heuristic SPIR-optimal filter design, in Chapter 4 we presented a *graphical tool for the generation of ground truth extracellular spiking data*. This tool implements a hybrid ground truth scheme, which enables the transformation of real recordings into ground truth recordings in a data-driven fashion. The tool requires some initial spike sorting proficiency from the user, rather than computational modelling experience as is the case for other ground truth paradigms. Therefore, this tool is a valuable resource for spike sorting users. The tool can help users to select the best spike sorting package for their specific recording setting, help them in tuning a specific algorithm to their specific needs or obtain a better understanding about the effect of certain algorithmic parameters on the spike sorting performance. Ideally, the use of hybrid ground-truth data leads to an improved spike sorting performance and/or less time spent on manual curation of the sorting results. Besides that primary audience, the tool is also useful for spike sorting developers for the validation of new techniques. The graphical tool is provided as an *open-source software* and has been integrated with SpikeInterface for improved compatibility within the spike sorting ecosystem. The tool also provides methods for efficient ground truth validation and presents an *automatic cluster merging* based on the available ground truth. The availability of an automatic cluster merging enables the assessment of the true potential of spike sorting packages that are not intended for fully automatic use.

In Chapter 5 we turned our attention to the limitation related to the lack of coupling between the interference identification and the actual filter output when using the heuristic from Chapter 2. As already envisioned in Chapter 2, the optimal filter for use in a threshold-based spike sorting setting is the SPIR-optimal filter. In Chapter 5 we then proposed a truly optimal filter design approach, where we *embedded the definition of peak-interference into the optimization problem* itself, as such providing an *explicit coupling between the interference identification and the actual filter output*. Due to this coupling, there is no closed-form filter solution available, such that one has to rely on available optimization solvers. Furthermore, this filter design approach inherits guarantees on the convergence towards a solution from such solvers. Unfortunately, due to the non-convex nature of the SPIR-optimality criterion, the solution could converge to a local optimum.

We then proposed two *convex reformulations of the SPIR-optimal filter design*. The advantage of these reformulations is that convergence to the global optimum is guaranteed. One of the convex reformulations was shown to be closely related to SVMs. As a consequence of this connection, the SPIR-optimal filter can be

interpreted as the *maximum-margin matched filter*. Furthermore, we provided a graphical proof concerning the robustness of such filters with respect to erroneous implicit labels from an SVM point of view during the training of such filters in the absence of a mechanism like the target safe zone from Chapter 2. All of the proposed filter designs from Chapter 5 require minimal prior knowledge in the form of the template of the target neuron, and depend on two interpretable hyper parameters: 1. the interference threshold fraction, for which a dynamic selection scheme was proposed, and, 2. the subspace regularization fraction.

We validated the use of SPIR-optimal filters in the context of spike sorting on an extensive hybrid ground truth data set. We have shown that the proposed convex SPIR-optimal filters outperform the SNR-optimal and non-convex SPIR-optimal filters for threshold-based spike sorting, both in the presence and absence of peak interferers. As such, the convex filters proposed in Chapter 5 are true threshold-based spike sorting enablers, not only in terms of sorting performance, but also due to their minimal prior knowledge requirements.

In Chapter 6 a novel and unconventional solution was proposed to resolve spike overlap. The approach consists of the design of a *spike embedding in which overlap behaves as a linear operator*. Furthermore, the embedding is constrained to reveal single-unit clusters. These two characteristics make that the embedding can be used to solve the spike sorting problem and overlapping spikes problem through an integrated approach in the feature space. Although the results related to the use of an overlap-resolving embedding for spike sorting are promising, the data augmentation approach that was used, does not allow an immediate application of the resulting embedding on real extracellular recordings. The proposed method is limited compared to matched filtering approaches in terms of resolving the exact occurrence of individual neuronal spikes for overlapping spikes. Yet, if this limitation is acceptable within a certain experimental setting, the embedding has the potential to resolve spike overlap using only the simple pipeline that was presented in Figure 1.5 and can be seen as an alternative low complexity approach to online spike sorting in the context of overlapping spikes.

In summary, in this thesis we have focussed on two low-complexity spike sorting architectures for resolving overlapping spikes, i.e.,

- **a threshold-based spike sorting architecture, and,**
- **a classical clustering-based spike sorting architecture.**

For these architectures we have contributed the fundamental building blocks that enable these architectures in the context of spike

overlap. For the threshold-based architecture we have developed

- **several innovative filter design methods based on SPIR optimality, and,**
- **an interpretable regularization scheme for use in such filter design.**

For the clustering-based architecture we have proposed

- **a spike embedding architecture based on a deep neural network, and,**
- **a custom loss function to find meaningful weights for that spike embedding architecture.**

Additionally, we have developed an open-source graphical tool to generate ground truth data to aid the validation of spike sorting algorithms during their development, as well as to guide users in selecting and tuning spike sorting algorithms for their specific recording setting.

7.2 Open challenges

Although the work presented in this thesis means a step forward for online spike sorting and potential embedded implementations thereof, some open challenges related to our work remain. In this section, we will give an overview of these challenges and we will sketch some rough ideas that can serve as a starting point for future research.

SPIR-optimal filtering

Although we've shown the potential for the use of SPIR-optimal filters in spike sorting, we've mostly ignored the fact that extracellular signal statistics can be non-stationary under certain conditions in practice, as was discussed in Section 1.4. To cope with varying signal statistics, an *adaptive* implementation of SPIR-optimal filters is to be investigated in order to cope with drift or bursting behavior. A possible initial approach could be to combine the work from Chapter 2 and 5 in the following way:

1. Determine an explicit interference covariance matrix on training data based on the activations obtained through the use of one of the SPIR-

optimal filter design methods from Chapter 5, rather than using the heuristic from Chapter 2.

2. Design a discriminative template matching filter based on that interference covariance matrix and the spike template of the target neuron.

3. During the actual filtering, every segment that generates an interference threshold crossing is integrated in the interference covariance matrix, e.g., by means of an exponential forgetting factor. Note that the inverse of a rank-one updated matrix can be efficiently calculated through the so-called Sherman-Morrison formula [41].

4. Also during the actual filtering, every segment that crosses the spike detection threshold can be used to update the template estimate.

5. Given the interference covariance and/or template estimate update, updated filters can be straightforwardly calculated given the formula's in Chapter 2.

Note that it remains unclear how the use of the activations determined from the methods in Chapter 5, especially when determined from the convex filter design methods, behave in the context of discriminative template matching filter design as proposed in Chapter 2. The above adaptive scheme does not involve complex numerical computations, and as such could be suitable for our intended online use.

Furthermore, for practical use of threshold-based spike sorting, more research has to be conducted related to the practical selection of a spike detection threshold. In our work, we have mostly shown the potential of such filters by determining the optimal working point by incorporating ground truth information into the threshold selection process. In practice, such ground truth is not available, although a partial labelling is assumed to be available from the template estimation process. Future research could thus investigate whether a threshold can be obtained that is close to the optimal threshold, given a filter output and the partial labelling information. Such a method could be based on a heuristic or can also be tackled as a machine learning problem.

When the above open challenges have been addressed, the work related to SPIR-optimal filtering could change its focus from research to development, i.e., focussing on making the filter design techniques accessible to a wider public. Because in SPIR-optimal filtering, the complexity is in the filter design process, rather than in actual filtering, an interesting model could be to provide the filter design as a cloud-based software service. For the actual (adaptive) filtering, several potential target platforms could be envisioned. For non-time critical spike sorting, a regular lab PC could be used for evaluating the filter outputs

and thresholding. For time critical lab experiments, a field-programmable gate array (FPGA) implementation in the data acquisition system could be provided [112]. Remember that the evaluation of the filter output consists only of linear operations, which can be efficiently implemented in hardware [124]. For wearable BMI-application, the filtering could be integrated into a low-power neural interface application-specific integrated ciruit (ASIC). Note that before an FPGA and ASIC implementation can be achieved, a fixed point implementation of SPIR-optimal filtering should be developed and validated.

To conclude the challenges related to SPIR-optimal filtering, we would like to point out that the low computational complexity of the SPIR-optimal filter is not a free lunch. It is important to realize that the spike sorting performance when using linear filters is ultimately bounded by its simple arithmetic and its limited degrees of freedom. Although we believe SPIR-optimal linear filtering is decent for threshold-based spike sorting, further advances in spike sorting performances likely require the use of non-linear signal processing techniques. Therefore, we believe it is worth pursuing the kernel SPIR-optimal filtering that was briefly mentioned in Chapter 5. The kernel SPIR-optimal filter can be interpreted as an infinite dimensional linear filter and thresholding, that is applied to a projection of the extracellular data into such an infinite dimensional space. Fortunately, through the application of the kernel trick [46], the computations do not explicitly take place in an infinite dimensional space.

Hybrid data generation

The current hybrid ground truth generation model that is provided to the public is limited in terms of modelling bursting and does not support probe drift. We envision further developments of the hybrid model in line with the original goal, i.e., a maximally data-driven approach.

An improved bursting model could fit a principal component to each spike to allow waveform variations, besides the pure template scaling. Such a fitting method is already available in the open-source back end of our tool, but it is disabled in the graphical user interface because experimentation showed that such a principal component fitting is prone to modelling overlapping spikes rather than unit waveform variations. This undesired behaviour can be explained by the use of relatively large temporal windows during hybridization. The impact of this undesired modelling behaviour should be minimized, e.g., by explicitly identifying a neuron as a bursting neuron prior to fitting the principal component. Shorter windows could also be used for determining and fitting such bursting principal component, but one should be careful not to introduce artefacts related to the use of small windows.

A data-driven drift model is envisioned that is inspired by biophysics. Without

providing mathematical details and without any guarantees related to its expressiveness, the outline for the data-driven modelling of a drifting single-unit template is as follows:

1. Determine the "location" of the neuronal source from its stable spatio-temporal spike template. For more details related to this task we refer to [51]. Given the source location, assume that the neuron behaves as a point source in that location.

2. For a new electrode location, consider the electrodes that are surrounding the new electrode location. Correct each single-channel template on the surrounding electrodes for its distance to the location of the neuron, thereby effectively projecting each channel back to the source space [68].

3. Interpolate in a linear fashion between the different point sources and project the interpolated point source to the new electrode location, obtaining a drifted single-channel template.

This scheme is to be repeated for every hybrid unit and electrode location separately. The background activity can be interpolated in the original recording space, because the above scheme converges to a simple interpolation for far-away sources. Although a drifting spike waveform can be potentially obtained in a data-driven fashion, the drifting electrode locations have to be artificially generated when assuming that the original recording is stable.

Another feature that might be of interest, especially in the context of spike overlap, is to implement an explicit control over the synchrony between different hybrid units. Note that such an approach means deviating from the data-driven spike time statistics, and as such might lead to more unrealistic hybrid scenario's.

Resolving spike overlap in the feature space

In Chapter 6 we proposed an innovative approach for resolving spike overlap in the feature space. Although the approach is not sufficiently mature for use in practice, the approach is promising with respect to the existence of an integrated spike sorting approach in the feature space that can cope with all practical spike sorting problems described in Section 1.4. The feature map could be trained with a cost function build on top of the proposed cost function. The extended cost function would have to include additional terms that enforce robustness in the feature map with respect to bursting and drift. These terms would be similar to the terms that have been described to control the variability within a single-unit cluster related to spike scaling and non-spatially aligned overlap.

As already mentioned, prior to extending this framework to improve robustness with respect to the various phenomena described above, the research focus should be on the development of data augmentation techniques that enable generalization of the feature map on real recordings. When including bursting and drift robustness into the problem, the data augmentation technique should also be aware of these phenomena. The simple data augmentation techniques used in Chapter 6 could be further improved in terms of the type of distributions that are fit to the principal component projection histograms. Another recent and interesting data augmentation approach that is worth exploring are generative adversarial networks (GANs) [36]. The resulting generator networks are in fact neural networks that have more descriptive power compared to our simple augmentation method, but are also more difficult to obtain.

Once the data augmentation problem has been overcome and a feature map that is useful in practice has been obtained, sparsity inducing terms on the feature map weights or the application of pruning techniques could be explored. These approaches will potentially lead to a significant reduction in feature map complexity, which will make them more suitable for online spike sorting.

As a final note related to this work, due to the scarcity of real extracellular ground truth data, very little information is available about the true level of spike overlap occurrence. When in practice two neurons are rarely active near-simultaneously, the overlapping spikes cluster for these two neurons will be hard to identify. This is an additional disadvantage of resolving overlap in the feature space, when compared to matched filtering approaches. Potentially this limitation can be overcome by using a one-hot encoding at the output of the neural network that encodes single-unit activity, although it is believed that such an approach would not be possible in a completely unsupervised setting, i.e., the feature map would have to be adjusted to a specific recording setting (using similar information as is exploited by the matched filter), e.g., by using transfer learning techniques. Such an approach would be very similar to the use of non-linear SPIR-optimal filter banks, where additional domain-specific prior knowledge would available. Ultimately, an integration of the different overlap resolving strategies that have been proposed in this work could elevate low-complexity spike sorting to unseen levels.

Bibliography

[1] Abeles, M., and Goldstein, M. H. Multispike train analysis. *Proceedings of the IEEE 65*, 5 (1977), 762–773.

[2] Aksenova, T. I., Chibirova, O. K., Dryga, O. A., Tetko, I. V., Benabid, A.-L., and Villa, A. E. An unsupervised automatic method for sorting neuronal spike waveforms in awake and freely moving animals. *Methods 30*, 2 (2003), 178–187.

[3] Allen, B. D., Moore-Kochlacs, C., Bernstein, J. G., Kinney, J., Scholvin, J., Seoane, L., Chronopoulos, C., Lamantia, C., Kodandaramaiah, S. B., Tegmark, M., et al. Automated in vivo patch clamp evaluation of extracellular multielectrode array spike recording capability. *Journal of neurophysiology* (2018).

[4] Arthur, D., and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 1027–1035.

[5] Aydin, Ç., Couto, J., Giugliano, M., Farrow, K., and Bonin, V. Locomotion modulates specific functional cell types in the mouse visual thalamus. *Nature communications 9*, 1 (2018), 1–12.

[6] Azzalini, A. *The skew-normal and related families*, vol. 3. Cambridge University Press, 2013.

[7] Bakkum, D. J., Radivojevic, M., Obien, M. E. J., Jaeckel, D., Frey, U., Takahashi, H., and Hierlemann, A. The axon initial segment drives the neuron's extracellular action potential. *bioRxiv* (2018), 266734.

[8] Bar-Hillel, A., Spiro, A., and Stark, E. Spike sorting: Bayesian clustering of non-stationary data. In *Advances in Neural Information Processing Systems* (2005), pp. 105–112.

[9] Blatt, M., Wiseman, S., and Domany, E. Superparamagnetic clustering of data. *Physical review letters 76*, 18 (1996), 3251.

[10] Boyd, S., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[11] Brette, R., and Destexhe, A. *Handbook of neural activity measurement*. Cambridge University Press, 2012.

[12] Buccino, A. P., and Einevoll, G. T. Mearec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity. *bioRxiv* (2019), 691642.

[13] Buccino, A. P., Hurwitz, C. L., Magland, J., Garcia, S., Siegle, J. H., Hurwitz, R., and Hennig, M. H. Spikeinterface, a unified framework for spike sorting. *BioRxiv* (2019), 796599.

[14] Calabrese, A., and Paninski, L. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods 196*, 1 (2011), 159–169.

[15] Camuñas-Mesa, L. A., and Quiroga, R. Q. A detailed and fast model of extracellular recordings. *Neural computation 25*, 5 (2013), 1191–1212.

[16] Carlson, D., and Carin, L. Continuing progress of spike sorting in the era of big data. *Current opinion in neurobiology 55* (2019), 90–96.

[17] Carpenter, R., and Reddi, B. *Neurophysiology: a conceptual approach*. CRC Press, 2012.

[18] Caruana, R., Lawrence, S., and Giles, C. L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems* (2001), pp. 402–408.

[19] Chanwimalueang, T., von Rosenberg, W., and Mandic, D. P. Enabling r-peak detection in wearable ecg: Combining matched filtering and hilbert transform. In *2015 IEEE International Conference on Digital Signal Processing (DSP)* (2015), IEEE, pp. 134–138.

[20] Chung, J. E., Joo, H. R., Fan, J. L., Liu, D. F., Barnett, A. H., Chen, S., Geaghan-Breiner, C., Karlsson, M. P., Karlsson, M., Lee, K. Y., et al. High-density, long-lasting, and multi-region electrophysiological recordings using polymer electrode arrays. *Neuron 101*, 1 (2019), 21–31.

[21] Chung, J. E., Magland, J. F., Barnett, A. H., Tolosa, V. M., Tooker, A. C., Lee, K. Y., Shah, K. G., Felix, S. H., Frank, L. M., and Greengard, L. F. A fully automated approach to spike sorting. *Neuron 95*, 6 (2017), 1381–1394.

[22] Ciliberti, D., and Kloosterman, F. Falcon: a highly flexible open-source software for closed-loop neuroscience. *Journal of Neural Engineering 14*, 4 (aug 2017), 045004.

[23] Claesen, M., De Smet, F., Suykens, J. A., and De Moor, B. A robust ensemble approach to learn from positive and unlabeled data using svm base models. *Neurocomputing 160* (2015), 73–84.

[24] Dégenètais, E., Thierry, A.-M., Glowinski, J., and Gioanni, Y. Electrophysiological properties of pyramidal neurons in the rat prefrontal cortex: an in vivo intracellular recording study. *Cerebral cortex 12*, 1 (2002), 1–16.

[25] Delescluse, M., and Pouzat, C. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. *Journal of neuroscience methods 150*, 1 (2006), 16–29.

[26] Einevoll, G. T., Franke, F., Hagen, E., Pouzat, C., and Harris, K. D. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology 22*, 1 (2012), 11–17.

[27] Ekanadham, C., Tranchina, D., and Simoncelli, E. P. A unified framework and method for automatic neural spike identification. *Journal of neuroscience methods 222* (2014), 47–55.

[28] Franke, F., Jäckel, D., Dragas, J., Müller, J., Radivojevic, M., Bakkum, D., and Hierlemann, A. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in neural circuits 6* (2012), 105.

[29] Franke, F., Jäckel, D., Dragas, J., Müller, J., Radivojevic, M., Bakkum, D., and Hierlemann, A. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in Neural Circuits 6*, December (2012), 1–7.

[30] Franke, F., Natora, M., Boucsein, C., Munk, M. H., and Obermayer, K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *Journal of computational neuroscience 29*, 1-2 (2010), 127–148.

[31] Franke, F., Quiroga, R. Q., Hierlemann, A., and Obermayer, K. Bayes optimal template matching for spike sorting–combining fisher discriminant analysis with optimal filtering. *Journal of computational neuroscience 38*, 3 (2015), 439–459.

[32] Furue, H., Katafuchi, T., and Yoshimura, M. In vivo patch-clamp technique. In *Patch-clamp analysis*. Springer, 2007, pp. 229–251.

[33] Gibson, S., Judy, J. W., and Marković, D. Spike sorting: The first step in decoding the brain. *IEEE Signal processing magazine 29*, 1 (2012), 124–143.

[34] Gligorijević, I., van Dijk, J. P., Mijović, B., Van Huffel, S., Blok, J. H., and De Vos, M. A new and fast approach towards sEMG decomposition. *Medical & biological engineering & computing 51*, 5 (2013), 593–605.

[35] Golub, G. H., and Van Loan, C. F. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[36] Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160* (2016).

[37] Gouwens, N. W., Berg, J., Feng, D., Sorensen, S. A., Zeng, H., Hawrylycz, M. J., Koch, C., and Arkhipov, A. Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature communications 9*, 1 (2018), 1–13.

[38] Grün, S., and Rotter, S. *Analysis of parallel spike trains*, vol. 7. Springer, 2010.

[39] Haas, A. M., Cohen, M. H., and Abshires, P. A. Real-time variance based template matching spike sorting system. In *2007 IEEE/NIH Life Science Systems and Applications Workshop* (2007), IEEE, pp. 104–107.

[40] Hagen, E., Ness, T. V., Khosrowshahi, A., Sørensen, C., Fyhn, M., Hafting, T., Franke, F., and Einevoll, G. T. Visapy: a python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of neuroscience methods 245* (2015), 182–204.

[41] Hager, W. W. Updating the inverse of a matrix. *SIAM review 31*, 2 (1989), 221–239.

[42] Hay, E., Hill, S., Schürmann, F., Markram, H., and Segev, I. Models of Neocortical Layer 5b Pyramidal Cells Capturing a Wide Range

of Dendritic and Perisomatic Active Properties. *PLoS Computational Biology 7*, 7 (jul 2011), e1002107.

[43] Hilgen, G., Sorbaro, M., Pirmoradian, S., Muthmann, J.-O., Kepiro, I. E., Ullo, S., Ramirez, C. J., Encinas, A. P., Maccione, A., Berdondini, L., et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports 18*, 10 (2017), 2521–2532.

[44] Hines, M. L., and Carnevale, N. T. The neuron simulation environment. *Neural computation 9*, 6 (1997), 1179–1209.

[45] Hippenstiel, R. D. *Detection theory: applications and digital signal processing*. CRC Press, 2001.

[46] Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics* (2008), 1171–1220.

[47] Holobar, A., and Zazula, D. Multichannel blind source separation using convolution kernel compensation. *IEEE Transactions on Signal Processing 55*, 9 (2007), 4487–4496.

[48] Huang, X., Shi, L., and Suykens, J. A. Asymmetric least squares support vector machine classifiers. *Computational Statistics & Data Analysis 70* (2014), 395–405.

[49] Hubel, D. H. Single unit activity in striate cortex of unrestrained cats. *The Journal of physiology 147*, 2 (1959), 226–238.

[50] Hunt, D. L., Lai, C., Smith, R. D., Lee, A. K., Harris, T. D., and Barbic, M. Multimodal in vivo brain electrophysiology with integrated glass microelectrodes. *Nature biomedical engineering* (2019), 1.

[51] Hurwitz, C., Xu, K., Srivastava, A., Buccino, A., and Hennig, M. Scalable spike source localization in extracellular recordings using amortized variational inference. In *Advances in Neural Information Processing Systems* (2019), pp. 4726–4738.

[52] Hutchison, W., Allan, R., Opitz, H., Levy, R., Dostrovsky, J., Lang, A., and Lozano, A. Neurophysiological identification of the subthalamic nucleus in surgery for parkinson's disease. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society 44*, 4 (1998), 622–628.

[53] Jun, J. J., Mitelut, C., Lai, C., Gratiy, S., Anastassiou, C., and Harris, T. D. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv* (2017), 101030.

[54] Jun, J. J., Steinmetz, N. A., Siegle, J. H., Denman, D. J., Bauza, M., Barbarits, B., Lee, A. K., Anastassiou, C. A., Andrei, A., Aydin, Ç., et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature 551*, 7679 (2017), 232–236.

[55] Kadir, S. N., Goodman, D. F., and Harris, K. D. High-dimensional cluster analysis with the masked em algorithm. *Neural computation* (2014).

[56] Kaneko, H., Suzuki, S., Okada, J., and Akamatsu, M. Multineuronal spike classification based on multisite electrode recording, whole-waveform analysis, and hierarchical clustering. *IEEE Transactions on Biomedical Engineering 46*, 3 (mar 1999), 280–290.

[57] Kessy, A., Lewin, A., and Strimmer, K. Optimal Whitening and Decorrelation. *The American Statistician 1305*, September (jan 2017), 0–0.

[58] Khatoun, A., Asamoah, B., and Mc Laughlin, M. Simultaneously excitatory and inhibitory effects of transcranial alternating current stimulation revealed using selective pulse-train stimulation in the rat motor cortex. *Journal of Neuroscience 37*, 39 (2017), 9389–9402.

[59] Kim, K. H., and Kim, S. J. Method for unsupervised classification of multiunit neural signal recording under low signal-to-noise ratio. *IEEE Transactions on biomedical engineering 50*, 4 (2003), 421–431.

[60] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[61] Knieling, S., Sridharan, K. S., Belardinelli, P., Naros, G., Weiss, D., Mormann, F., and Gharabaghi, A. An unsupervised online spike-sorting framework. *International journal of neural systems 26*, 05 (2016), 1550042.

[62] Koshiba, Y., and Abe, S. Comparison of l1 and l2 support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* (2003), vol. 3, IEEE, pp. 2054–2059.

[63] Lay, D. C. *Linear algebra and its applications*. Addison Wesley Publishing Company, 1994.

[64] Lee, J. H., Carlson, D. E., Razaghi, H. S., Yao, W., Goetz, G. A., Hagen, E., Batty, E., Chichilnisky, E., Einevoll, G. T., and Paninski, L. Yass: Yet another spike sorter. In *Advances in neural information processing systems* (2017), pp. 4002–4012.

[65] Lefebvre, B., Stimberg, M., Marre, O., and Yger, P. Online template-matching based spike sorting for microelectrode arrays with hundreds of channels. *Cellular Neuroscience Archive*, 31.

[66] Lefebvre, B., Yger, P., and Marre, O. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris 110*, 4 (2016), 327–335.

[67] Lewicki, M. S. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems 9*, 4 (1998), R53–R78.

[68] Lindén, H., Hagen, E., Leski, S., Norheim, E. S., Pettersen, K. H., and Einevoll, G. T. Lfpy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons. *Frontiers in neuroinformatics 7* (2014), 41.

[69] Lopez, C. M., Mitra, S., Putzeys, J., Raducanu, B., Ballini, M., Andrei, A., Severi, S., Welkenhuysen, M., Van Hoof, C., Musa, S., and Yazicioglu, R. F. A 966-electrode neural probe with 384 configurable channels in 0.13µm SOI CMOS. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)* (jan 2016), vol. 59, IEEE, pp. 392–393.

[70] Lopez, C. M., Putzeys, J., Raducanu, B. C., Ballini, M., Wang, S., Andrei, A., Rochus, V., Vandebriel, R., Severi, S., Van Hoof, C., et al. A neural probe with up to 966 electrodes and up to 384 configurable channels in 0.13μm soi cmos. *IEEE transactions on biomedical circuits and systems 11*, 3 (2017), 510–522.

[71] Luan, S., Williams, I., Maslik, M., Liu, Y., De Carvalho, F., Jackson, A., Quiroga, R. Q., and Constandinou, T. G. Compact standalone platform for neural recording with real-time spike sorting and data logging. *Journal of neural engineering 15*, 4 (2018), 046014.

[72] Magland, J. F., Jun, J. J., Lovero, E., Morley, A. J., Hurwitz, C. L., Buccino, A. P., Garcia, S., and Barnett, A. H. Spikeforest: reproducible web-facing ground-truth validation of automated neural spike sorters. *bioRxiv* (2020).

[73] Markram, H., Muller, E., Ramaswamy, S., Reimann, M. W., Abdellah, M., Sanchez, C. A., Ailamaki, A., Alonso-Nanclares, L., Antille, N., Arsever, S., et al. Reconstruction and simulation of neocortical microcircuitry. *Cell 163*, 2 (2015), 456–492.

[74] Marre, O., Amodei, D., Deshmukh, N., Sadeghi, K., Soo, F., Holy, T. E., and Berry, M. J. Mapping a complete neural population in the retina. *Journal of Neuroscience 32*, 43 (2012), 14859–14873.

[75] Maynard, E. M., Nordhausen, C. T., and Normann, R. A. The utah intracortical electrode array: a recording structure for potential brain-computer interfaces. *Electroencephalography and clinical neurophysiology 102*, 3 (1997), 228–239.

[76] Merletti, R., and Farina, D. *Surface electromyography: physiology, engineering, and applications*. John Wiley & Sons, 2016.

[77] Michon, F., et al. Integration of silicon-based neural probes and micro-drive arrays for chronic recording of large populations of neurons in behaving animals. *J Neural Eng 13*, 4 (2016), 046018.

[78] Moser, E. I., Kropff, E., and Moser, M.-B. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci. 31* (2008), 69–89.

[79] Nadathur, S., and Crittenden, R. A detection of the integrated sachs–wolfe imprint of cosmic superstructures using a matched-filter approach. *The Astrophysical Journal Letters 830*, 1 (2016), L19.

[80] Neto, J. P., Lopes, G., Frazão, J., Nogueira, J., Lacerda, P., Baião, P., Aarts, A., Andrei, A., Musa, S., Fortunato, E., et al. Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *Journal of neurophysiology 116*, 2 (2016), 892–903.

[81] Nguyen, D. P., et al. Micro-drive array for chronic in vivo recording: tetrode assembly. *JoVE*, 26 (2009).

[82] Nguyen, T. K. T., Navratilova, Z., Cabral, H., Wang, L., Gielen, G., Battaglia, F. P., and Bartic, C. Closed-loop optical neural stimulation based on a 32-channel low-noise recording system with online spike sorting. *Journal of neural engineering 11*, 4 (2014), 046005.

[83] Oliynyk, A., Bonifazzi, C., Montani, F., and Fadiga, L. Automatic online spike sorting with singular value decomposition and fuzzy c-mean clustering. *BMC neuroscience 13*, 1 (2012), 96.

[84] Othman, M. A. B., Belz, J., and Farhang-Boroujeny, B. Performance analysis of matched filter bank for detection of linear frequency modulated chirp signals. *IEEE Transactions on Aerospace and Electronic Systems 53*, 1 (2017), 41–54.

[85] Oweiss, K. G., and Anderson, D. J. Spike sorting: a novel shift and amplitude invariant technique. *Neurocomputing 44* (2002), 1133–1139.

[86] Pachitariu, M., Steinmetz, N. A., Kadir, S. N., Carandini, M., and Harris, K. D. Fast and accurate spike sorting of high-channel count probes with kilosort. In *Advances in Neural Information Processing Systems* (2016), pp. 4448–4456.

[87] Pachitariu, Marius. Kilosort2.

[88] Park, J., Kim, G., and Jung, S.-D. A 128-channel fpga-based real-time spike-sorting bidirectional closed-loop neural interface system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering 25*, 12 (2017), 2227–2238.

[89] Patil, P. G., and Turner, D. A. The development of brain-machine interface neuroprosthetic devices. *Neurotherapeutics 5*, 1 (2008), 137–146.

[90] Pillow, J. W., Shlens, J., Chichilnisky, E., and Simoncelli, E. P. A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings. *PloS one 8*, 5 (2013), e62123.

[91] Pouzat, C., Delescluse, M., Viot, P., and Diebolt, J. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a markov chain monte carlo approach. *Journal of neurophysiology 91*, 6 (2004), 2910–2928.

[92] Pouzat, C., Mazor, O., and Laurent, G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *Journal of Neuroscience Methods 122*, 1 (dec 2002), 43–57.

[93] Prentice, J. S., Homann, J., Simmons, K. D., Tkačik, G., Balasubramanian, V., and Nelson, P. C. Fast, scalable, bayesian spike identification for multi-electrode arrays. *PloS one 6*, 7 (2011).

[94] Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation 16*, 8 (2004), 1661–1687.

[95] Raducanu, B. C., Yazicioglu, R. F., Lopez, C. M., Ballini, M., Putzeys, J., Wang, S., Andrei, A., Welkenhuysen, M., van Helleputte, N., Musa, S., Puers, R., Kloosterman, F., van Hoof, C., and Mitra, S. Time multiplexed active neural probe with 678 parallel recording sites. In *2016 46th European Solid-State Device Research Conference (ESSDERC)* (sep 2016), vol. 2016-Octob, IEEE, pp. 385–388.

[96] Ramaswamy, S., Courcol, J.-D., Abdellah, M., Adaszewski, S. R., Antille, N., Arsever, S., Atenekeng, G., Bilgili, A., Brukau, Y., Chalimourda, A., et al. The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Frontiers in neural circuits 9* (2015), 44.

[97] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association 66*, 336 (1971), 846–850.

[98] Rey, H. G., Pedreira, C., and Quiroga, R. Q. Past, present and future of spike sorting techniques. *Brain research bulletin 119* (2015), 106–117.

[99] Rinberg, D., Davidowitz, H., and Tishby, N. Multi-electrode spike sorting by clustering transfer functions. In *Advances in neural information processing systems* (1999), pp. 146–152.

[100] Roberts, W. M., and Hartline, D. K. Separation of multi-unit nerve impulse trains by a multi-channel linear filter algorithm. *Brain Research 94*, 1 (aug 1975), 141–149.

[101] Rodriguez, A., Boddeti, V. N., Kumar, B. V., and Mahalanobis, A. Maximum margin correlation filter: A new approach for localization and classification. *IEEE Transactions on Image Processing 22*, 2 (2012), 631–643.

[102] Rodriguez, A., and Laio, A. Clustering by fast search and find of density peaks. *Science 344*, 6191 (2014), 1492–1496.

[103] Rossant, C., Kadir, S. N., Goodman, D. F., Schulman, J., Hunter, M. L., Saleem, A. B., Grosmark, A., Belluscio, M., Denfield, G. H., Ecker, A. S., et al. Spike sorting for large, dense electrode arrays. *Nature neuroscience 19*, 4 (2016), 634.

[104] Rossant, Cyrille. cortex-lab/phy.

[105] Rutishauser, U., Schuman, E. M., and Mamelak, A. N. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of neuroscience methods 154*, 1-2 (2006), 204–224.

[106] Salahdine, F., El Ghazi, H., Kaabouch, N., and Fihri, W. F. Matched filter detection with dynamic threshold for cognitive radio networks. In *2015 International Conference on Wireless Networks and Mobile Communications (WINCOM)* (2015), IEEE, pp. 1–6.

[107] Salama, U., Sarker, P. L., and Chakrabarty, A. Enhanced energy detection using matched filter for spectrum sensing in cognitive radio networks. In *2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)* (2018), IEEE, pp. 185–190.

[108] Santhanam, G., Sahani, M., Ryu, S. I., and Shenoy, K. V. An extensible infrastructure for fully automated spike sorting during online experiments. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2004), vol. 2, IEEE, pp. 4380–4384.

[109] Schwartz, A. B. Cortical neural prosthetics. *Annu. Rev. Neurosci. 27* (2004), 487–507.

[110] Shan, K. Q., Lubenov, E. V., and Siapas, A. G. Model-based spike sorting with a mixture of drifting t-distributions. *Journal of neuroscience methods 288* (2017), 82–98.

[111] Shoham, S., Fellows, M. R., and Normann, R. A. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of Neuroscience Methods 127*, 2 (aug 2003), 111–122.

[112] Siegle, J. H., López, A. C., Patel, Y. A., Abramov, K., Ohayon, S., and Voigts, J. Open ephys: an open-source, plugin-based platform for multichannel electrophysiology. *Journal of neural engineering 14*, 4 (2017), 045003.

[113] Steinbach, M., Ertöz, L., and Kumar, V. The challenges of clustering high dimensional data. In *New directions in statistical physics*. Springer, 2004, pp. 273–309.

[114] Steinmetz, N., Carandini, M., and Harris, K. D. “Single Phase3" and “Dual Phase3" Neuropixels Datasets.

[115] Steinmetz, N., Pachitariu, M., Stringer, C., Carandini, M., and Harris, K. Eight-probe Neuropixels recordings during spontaneous behaviors.

[116] Sukiban, J., Voges, N., Dembek, T. A., Pauli, R., Visser-Vandewalle, V., Denker, M., Weber, I., Timmermann, L., and Grün, S. Evaluation of spike sorting algorithms: Application to human subthalamic nucleus recordings and simulations. *Neuroscience 414* (2019), 168–185.

[117] Swindale, N. V., and Spacek, M. A. Spike sorting for polytrodes: a divide and conquer approach. *Frontiers in systems neuroscience 8* (2014), 6.

[118] Tharwat, A. Classification assessment methods. *Applied Computing and Informatics* (2018).

[119] Theodoridis, S. *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2015.

[120] Theodoridis, S., Pikrakis, A., Koutroumbas, K., and Cavouras, D. *Introduction to pattern recognition: a matlab approach*. Academic Press, 2010.

[121] Trautmann, E. M., Stavisky, S. D., Lahiri, S., Ames, K. C., Kaufman, M. T., O'Shea, D. J., Vyas, S., Sun, X., Ryu, S. I., Ganguli, S., et al. Accurate estimation of neural population dynamics without spike sorting. *Neuron 103*, 2 (2019), 292–308.

[122] Turin, G. An introduction to matched filters. *IRE transactions on Information theory 6*, 3 (1960), 311–329.

[123] Valencia, D., and Alimohammad, A. An efficient hardware architecture for template matching-based spike sorting. *IEEE transactions on biomedical circuits and systems 13*, 3 (2019), 481–492.

[124] Véstias, M., Duarte, R. P., de Sousa, J. T., and Neto, H. Parallel dot-products for deep learning on fpga. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (2017), IEEE, pp. 1–4.

[125] Viswam, V., Obien, M. E. J., Franke, F., Frey, U., and Hierlemann, A. Optimal electrode size for multi-scale extracellular-potential recording from neuronal assemblies. *Frontiers in neuroscience 13* (2019), 385.

[126] Vollgraf, R., and Obermayer, K. Improved optimal linear filters for the discrimination of multichannel waveform templates for spike-sorting applications. *IEEE Signal Processing Letters 13*, 3 (2006), 121–124.

[127] von Bartheld, C. S., Bahney, J., and Herculano-Houzel, S. The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting. *Journal of Comparative Neurology 524*, 18 (2016), 3865–3895.

[128] Wang, P. K., Pun, S. H., Chen, C. H., McCullagh, E. A., Klug, A., Li, A., Vai, M. I., Mak, P. U., and Lei, T. C. Low-latency single channel real-time neural spike sorting system based on template matching. *PloS one 14*, 11 (2019), e0225138.

[129] Weingärtner, S., Chen, X., Akçakaya, M., and Moore, T. Robust online spike recovery for high-density electrode recordings using convolutional compressed sensing. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)* (2019), IEEE, pp. 1015–1020.

[130] Wouters, J., Kloosterman, F., and Bertrand, A. Data-driven multi-channel filter design with peak-interference suppression for threshold-based spike sorting in high-density neural probes. In *Proc. IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP), Calgary, Alberta, Canada, Apr. 2018* (2018).

[131] Wouters, J., Kloosterman, F., and Bertrand, A. Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes. *Journal of neural engineering 15*, 5 (2018), 056005.

[132] Wouters, J., Kloosterman, F., and Bertrand, A. A data-driven regularization approach for template matching in spike sorting with high-density neural probes. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany* (2019), IEEE.

[133] Wouters, J., Kloosterman, F., and Bertrand, A. SHYBRID: A graphical tool for generating hybrid ground-truth spiking data for evaluating spike sorting performance. *bioRxiv* (2019), 734061.

[134] Wouters, J., Kloosterman, F., and Bertrand, A. Signal-to-peak-interference ratio maximization with automatic interference weighting for threshold-based spike sorting of high-density neural probe data. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)* (2019), IEEE, pp. 247–250.

[135] Wu, X., Chen, H., Gan, T., Chen, J., Ngo, C.-W., and Peng, Q. Automatic hookworm detection in wireless capsule endoscopy images. *IEEE transactions on medical imaging 35*, 7 (2016), 1741–1752.

[136] Yang, Y., Boling, S., and Mason, A. J. A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces. *IEEE transactions on biomedical circuits and systems 11*, 4 (2017), 743–754.

[137] Yger, P., Spampinato, G. L., Esposito, E., Lefebvre, B., Deny, S., Gardella, C., Stimberg, M., Jetter, F., Zeck, G., Picaud, S., et al. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *Elife 7* (2018), e34518.

[138] Zani, A., and Proverbio, A. M. Cognitive electrophysiology of mind and brain. In *The cognitive electrophysiology of mind and brain*. Elsevier, 2003, pp. 3–12.

List of publications

[1] Carpentier, L., Hemeryck, M., Wouters, J., Berckmans, D., Vranken, E., Norton, T., and Berckmans, D. Economical case study of the somo respiratory distress monitor in pigs. In *Proc. of Asian Conference on Precision Livestock Farming* (2016), pp. 1–5.

[2] Wouters, J., Kloosterman, F., and Bertrand, A. Data-driven multi-channel filter design with peak-interference suppression for threshold-based spike sorting in high-density neural probes. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP), Calgary, Alberta, Canada, Apr. 2018* (2018).

[3] Wouters, J., Kloosterman, F., and Bertrand, A. Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes. *Journal of neural engineering 15*, 5 (2018), 056005.

[4] Wouters, J., Kloosterman, F., and Bertrand, A. A data-driven regularization approach for template matching in spike sorting with high-density neural probes. In *Proc. of 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany* (2019), IEEE.

[5] Wouters, J., Kloosterman, F., and Bertrand, A. Signal-to-peak-interference ratio maximization with automatic interference weighting for threshold-based spike sorting of high-density neural probe data. In *Proc. of 9th International IEEE/EMBS Conference on Neural Engineering (NER)* (2019), IEEE, pp. 247–250.

[6] Wouters, J., Kloosterman, F., and Bertrand, A. A neural network-based spike sorting feature map that resolves spike overlap in the feature space. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP), Barcelona, Spain, May 2020* (2020), IEEE.

[7] Wouters, J., Kloosterman, F., and Bertrand, A. SHYBRID: A graphical tool for generating hybrid ground-truth spiking data for evaluating spike sorting performance. *Accepted for publication in Neuroinformatics* (2020).



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10, bus 2446
B-3001 Leuven
jasper.wouters@esat.kuleuven.be
https://www.esat.kuleuven.be/stadius/