Wim De Vilder - filter

Verwijderen van 'ademhalingsruis' uit spraaksignalen

29 september 2014

Inhoudsopgave

1 Doelstelling **2**

2 Spraakdetectie (3 lessen) **2**

2.1 Voice Activity Detection 2

2.2 Classificatie 4

3 Features voor ademhalingsdetectie **6**

3.1 Zero crossing rate 6

3.2 Periodiciteit 6

3.3 Spectrale energie 8

3.4 Linear Predictive Coding 10

3.5 Tijdsinformatie 11

3.6 Cepstrale energie (optioneel) 12

4 Real-time implementatie (7 lessen) **12**

4.1 Bayesian classificatie (2 lessen) 13

4.2 Gaussian Mixture Models (5 lessen) 15

5 Demonstratie (4 lessen) **17**

5.1 Bonus feature: tijdsinformatie 17

1 Doelstelling

In deze moderne tijden moeten nieuwsuitzendingen gestroomlijnd en snel verlopen, wat grote inspanningen vergt van de huidige generatie nieuwslezers (zoals Wim De Vilder). Desondanks de trainingen die nieuwslezers krijgen, blijft het snelle tempo van de nieuwsuitzending er voor zorgen dat de nieuwslezer nog geregeld letterlijk naar adem moet happen. Tussen twee lange zinnen wordt er diep ingeademd, wat te horen is als een luid sissend geluid dat danig op de zenuwen kan werken van de TV-kijkers.

De vraag is of we met signaalverwerking Wim De Vilder en zijn collega's een handje zouden kunnen helpen. Kunnen we een 'Wim De Vilder - filter' maken dat automatisch de segmenten verwijdert waar Wim moet inademen en dat alleen de spraak doorlaat?

Daarom zoeken we in dit P&O project naar een spraakdetector die robuust is tegen 'ademhalingsruis'. Er zal onderzocht worden welke karakteristieken een typerend verschil geven tussen 'ademhaling' en 'spraak'. Die karakteristieken worden dan doormiddel van digitale signaalverwerkingstechnieken uit het audiosignaal geëxtraheerd om vervolgens te beslissen of een stukje geluid 'ademhaling' of 'spraak' voorstelt. Stukjes geluid die als 'ademhaling' worden gedetecteerd kunnen dan onderdrukt worden, de stukjes 'spraak' blijven onaangeroerd.

2 Spraakdetectie (3 lessen)

2.1 Voice Activity Detection

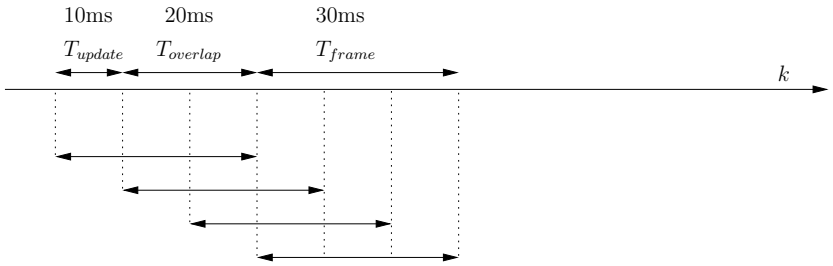
Voice Activity Detection (VAD) oftewel spraakdetectie is een probleem dat in vele toepassingen opduikt. Een bekend voorbeeld is de GSM. Aangezien de draadloze zender in je GSM een energievreter is, heb je er alle belang bij dat er alleen informatie wordt verstuurd als je effectief iets aan het zeggen bent. Door de zender uit te schakelen tijdens de periodes dat je aan het luisteren bent naar de partij aan de andere kant, moet je je GSM veel minder snel opladen. De beslissing om de zender al dan niet uit te schakelen gebeurt aan de hand van het signaal dat opgepikt wordt door de microfoon: stelt dit signaal spraak voor of is het achtergrond ruis.

Een eenvoudige manier om te beslissen of we al dan niet te maken hebben met spraak, maakt gebruik van het feit dat de bron van het spraaksignaal zich dichter bij de microfoon bevindt dan de bron(nen) van de achtergrond ruis. Wanneer de energie van het waargenomen signaal dus boven een bepaalde threshold komt, hebben we te maken met spraak. De energie op tijdstip k over een frame van N samples wordt berekend als

$$E[k] = \sum_{i=0}^{N-1} \left(x[k-i]\right)^2 . \tag{1}$$

Typisch wordt een audiosignaal verdeeld in frames van enkele tientallen milliseconden, eventueel met overlapping. Schematisch wordt dit weergegeven in figuur 1. Door de energie van een frame te vergelijken met een threshold T, wordt beslist of het frame spraak of achtergrondruis voorstelt.

Opdracht: Bouw een VAD gebaseerde spraakdetector in Matlab.

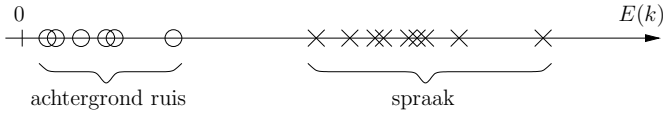


Figuur 1: Framing: opdelen van een signaal is stukjes

- Maak een aantal opnames waarbij je afwisselend spreekt en zwijgt. Zorg dat er wat achtergrondruis is (sprekende mensen, muziek,...). Adem ook een paar keer diep in zoals een nieuwslezer. Sla je files op in het *.wav* formaat en lees je audiobestanden in in Matlab. Enkele nuttige Matlab-commando's om met audiobestanden te werken zijn `wavrecord, wavread` en `wavwrite`. Meer informatie over hoe je met een bepaald Matlab-commando moet werken kan je bekomen met `help` <*commando*>.
- Verdeel je audiosignaal in frames zoals aangegeven in figuur 1. Zorg ervoor dat je de overlapping gemakkelijk kan instellen van 0 tot $T_{frame} - 1$.
- Bereken voor elk frame de energie en ga op zoek naar een goede threshold waarmee je spraak kan onderscheiden van achtergrondruis. Naast luisteren kan je ook plots gebruiken (`plot, subplot`) om de kwaliteit van je filter te controleren!

2.2 Classificatie

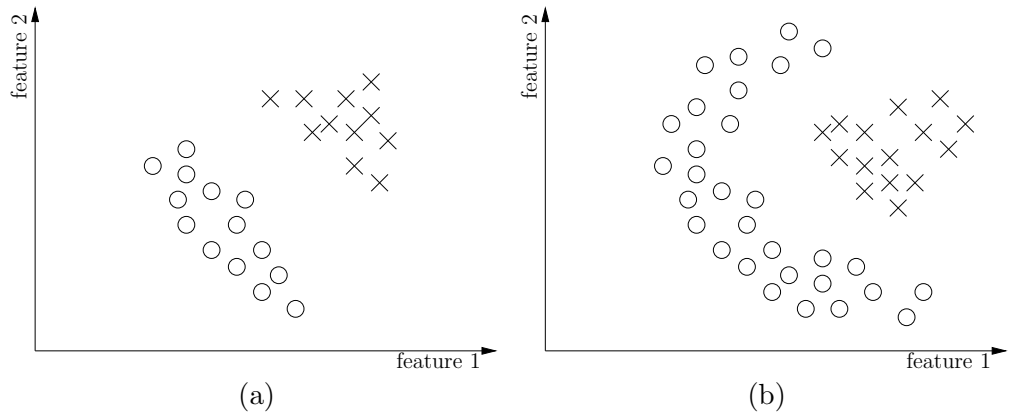
'Classificatie' is het op automatische wijze toekennen van objecten (bv. stukjes geluid) aan een set van klassen (bv. spraak, stilte,...). De beslissing tot welke klasse een object behoort gebeurt aan de hand van *features*, specifieke kenmerken die op één of andere manier geëxtraheerd worden uit het object. Een voorbeeld van een feature dat van toepassing kan zijn bij spraakdetectie is de energie van een frame, zoals je die in je VAD gebaseerde spraakdetector gebruikte. Als je zo'n feature weergeeft in een *feature-ruimte*, dan krijg je een situatie zoals weergegeven in figuur 2.



Figuur 2: 1-dimensionale feature-ruimte

Hieruit blijkt duidelijk dat je aan de hand van een threshold een onderscheid kan maken tussen spraak en achtergrondruis. Dit hebben we te danken aan de kwaliteit van het feature dat werd gebruikt: features met een goed onderscheidend vermogen zorgen ervoor dat de te detecteren klassen clusters vormen die duidelijk van elkaar gescheiden zijn in de feature-ruimte.

Naarmate de klassen die we willen detecteren sterker op elkaar gaan lijken of wanneer we geen specifiek feature kunnen extraheren met een goed onderscheiden vermogen, moeten we op zoek gaan naar een set van features die samen wel een onderscheidend vermogen hebben. We krijgen dan een meer-dimensionale feature-ruimte door een set van features als een vector te beschouwen. Een voorbeeld van 2-dimensionale feature-ruimtes is weergegeven in figuur 3, waarbij de objecten worden gekenmerkt door een vector [feature 1 , feature 2]^T.



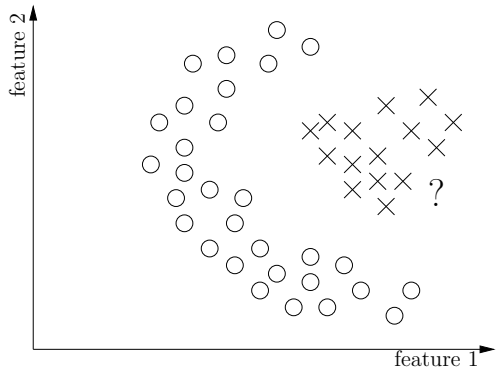
Figuur 3: 2-dimensionale feature-ruimtes

In deze feature-ruimtes merken we weer 2 duidelijk van elkaar onderscheiden clusters. De combinatie van de 2 features heeft dus een goed onderscheidend vermogen. Wanneer we slechts één van de features zouden hebben (projectie op een as), vloeien de clusters in elkaar en kunnen we ze niet van elkaar onderscheiden.

Alhoewel de objectklassen duidelijk van elkaar gescheiden clusters vormen in de feature-ruimte, kunnen we de klassen niet meer van elkaar onderscheiden door een simpele threshold te gebruiken. In figuur 3(a) zouden we nog kunnen proberen om de klassen te scheiden door een rechte, maar figuur 3(b) toont dat dit in het algemene geval niet altijd mogelijk is. We moeten op zoek gaan naar een manier om te beslissen tot welke cluster een bepaald punt in de feature-ruimte behoort.

Een oplossing bestaat erin om de classifier te *trainen*. Tijdens het trainen extraheren we de features van een set trainingsdata. De trainingsdata wordt dan met de hand geclassificeerd: we geven zelf aan tot welke klasse een bepaalde set features behoort. Wanneer we later met echte data te maken hebben, kunnen we op basis van de afstand tot de trainingspunten in de feature-ruimte beslissen tot welke klasse die nieuwe data behoort.

Figuur 4 maakt dit wat duidelijker. Met de bolletjes worden trainingspunten (feature-vectoren) weergegeven die bekomen werden met trainingsdata uit klasse A, met de kruisjes trainingspunten uit klasse B. Wanneer we nu een nieuw te classificeren object hebben dat via zijn features op de plaats van het vraagteken geplaatst wordt, dan kunnen we op basis van de afstand in de feature-ruimte relatief zeker zijn dat dit object tot klasse B behoort. Met andere woorden, het object wordt toegewezen aan dezelfde klasse als het trainingsobject dat het dichtst bij ligt (kortste Euclidische afstand) in de feature-ruimte.



Figuur 4: Classificatie aan de hand van trainingsdata

Opdracht: Maak een nieuwe versie van je VAD gebaseerde spraakdetector. Werk deze keer niet met een threshold, maar train je filter en classificeer op basis van kortste Euclidische afstand in de 1D feature-ruimte.

- Kan je het ontworpen filter robuuster maken tegen *outliers* (sporadische trainingspunten die nogal ver van hun cluster liggen of misschien zelfs foutief geclassificeerde trainingsdata)?
- Waarom moet je bij het testen van je filter verschillende data gebruiken voor de training en de test?
- Werkt het ontworpen filter goed voor verschillende sprekers ?

3 Features voor ademhalingsdetectie

In de volgende secties gaan we op zoek naar features of combinaties van features die ons toelaten om ademhalingsruis te onderdrukken. Dit kan gebeuren door een spraakdetectie te doen die robuust is tegen ademhalingsruis en alles wat geen spraak is te onderdrukken of door specifiek een detectie te doen van ademhalingsruis en die dan te onderdrukken.

- Maak een aantal opnames waarbij je de lokale nieuwtjes voorleest. Imiteer daarbij het ademhalingsgedrag van een doorsnee nieuwslezer.
- Maak ook opnames van een aantal *echte* nieuwslezers zoals Wim De Vilder. Je kan hiervoor materiaal vinden in de videozone van `http://www.deredactie.be`.
- Test voor alle features die verder aan bod komen hoe goed ze er in slagen om spraak van ademhaling te onderscheiden (classificatievermogen).
- Test de spreker-afhankelijkheid voor alle features die verder aan bod komen: werkt een filter dat getrained is met een bepaalde spreker ook voor een andere spreker ? Werkt een filter getrained met een mannelijke stem ook voor vrouwelijke stemmen, en omgekeerd ?
- Gebruik je ervaringen met de features om bepaalde combinaties van features uit te proberen waarmee een verbeterde classificatie kan bekomen worden. Het geheel is vaak beter dan de som van de delen !

3.1 Zero crossing rate

Het inademen van de nieuwslezer veroorzaakt een sissend geluid. De hoge frequenties die hiermee gepaard gaan zijn mogelijk typerend voor de ademhaling. Een eenvoudig feature dat gerelateerd is aan de hoogste frequentie aanwezig in een signaal, is de zero crossing rate: hoeveel keer per tijdseenheid verandert het signaal van teken.

3.2 Periodiciteit

Wanneer een spraaksignaal opgedeeld is in frames, dan komt een frame overeen met een bepaalde klank. Dergelijke klank heeft typisch een periodisch karakter. De periode van een klank komt overeen met de laagste frequentie die in het signaal aanwezig is.

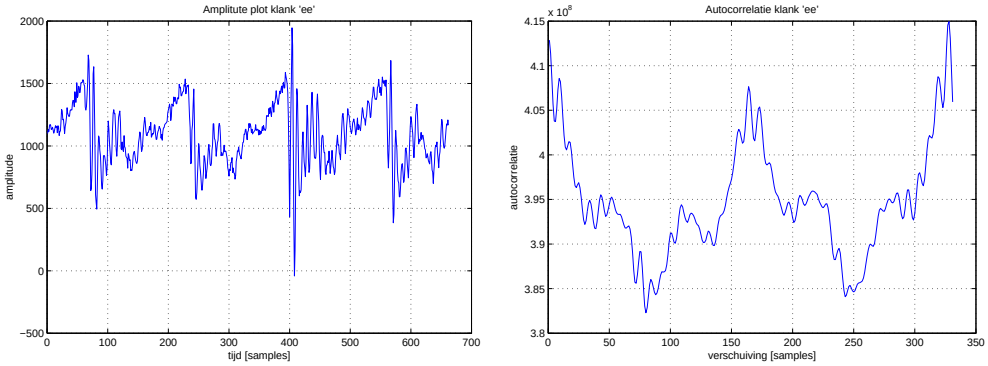
Om de periode uit een frame te extraheren kan een autocorrelatie gebruikt worden. De autocorrelatie-functie wordt gedefinieerd als

$$R_{xx}[i] = \frac{1}{N} \sum_{k=1}^{N} x[k]x[k+i] \tag{2}$$

en geeft aan hoe sterk een signaal lijkt op een verschoven versie van zichzelf. De index i waarvoor de autocorrelatie-functie een piekt, geeft aan hoeveel samples het signaal verschoven moet worden om een versie te krijgen die zo sterk mogelijk op het origineel lijkt. Het is dus een maat voor de periode van het signaal.

De periode van een spraaksignaal geeft de *pitch* weer van de stem: de frequentie waarmee de stembanden trillen. Voor mannen ligt de pitch typisch tussen 110Hz en 130Hz, voor vrouwen tussen 200Hz en 230Hz. Een pitch-waarde binnen deze intervallen zou er dus op kunnen wijzen dat een frame een stukje spraak voorstelt.

Een voorbeeld is weergegeven in figuur 5: links staat het amplitudeverloop voor de klank 'ee' met een zaagtand-achtige periode, rechts de autocorrelatie. De samplingfrequentie is 22050Hz. De eerste piek in de autocorrelatie bij een verschuiving van 165 samples wijst erop dat de periode van het signaal 165 samples is (en de 2de piek bij een verschuiving van 330 samples bevestigt dit). De pitch is hier dus ongeveer 22050/165=133Hz.



(a) (b)

Figuur 5: (a) Amplitude plot voor de klank 'ee' gedurende 30ms. (b) Autocorrelatie plot voor de klank 'ee'.

3.3 Spectrale energie

Energie werd reeds als feature gebruikt in de Voice Activity Detector uit hoofdstuk 2.

We kunnen echter nog een stapje verder gaan en op een andere manier naar ons signaal kijken. Tot nu toe hebben we het signaal altijd in het *tijdsdomein* beschouwd: op elk tijdstip de amplitude van het signaal. Wanneer we met signaalverwerking bezig zijn, krijgen we echter dikwijls een beter beeld van het signaal als we dit in het *frequentiedomein* bekijken.

Fourier toonde aan dat elk periodisch signaal kan ontbonden worden in een (mogelijk oneindige) som van sinussen en cosinussen, waarvan de frequenties veelvouden zijn van 1/periode van het signaal. Bij uitbreiding geldt hetzelfde voor alle signalen (ook niet-periodische): we kunnen stellen dat een niet-periodisch signaal een oneindige periode heeft en dus komen de frequenties van de sinussen en cosinussen in de ontbinding oneindig dicht bij elkaar liggen. Bovendien zijn een sinus en een cosinus op een faseverschuiving na dezelfde functie. Om een signaal te beschrijven in het frequentiedomein hebben we dus voldoende aan de amplitudes en de fases van de deelsinussen. In het frequentiedomein zien we bijgevolg de *frequentie-inhoud* van het signaal.

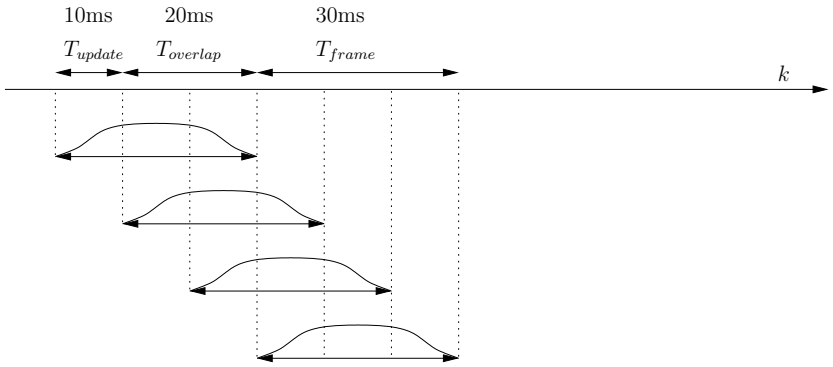
In Matlab kunnen we gemakkelijk omrekenen tussen het tijdsdomein en het frequentiedomein: om van het tijdsdomein naar het frequentiedomein te gaan gebruik je de functie `fft`, om van het frequentiedomein naar het tijdsdomein te gaan de functie `ifft`.

- Test dit eens uit op een blokvormige puls: `x = [zeros(1,10) ones(1,10) zeros(1,10)];` Kan je dit signaal transformeren naar het frequentiedomein en terug? Hoe ziet de frequentie-inhoud van dergelijk signaal eruit?

- Merk op dat `fft` en `ifft` de amplitude en fase van de sinussen niet apart geven maar als een fasor: een complex getal waarvan de norm gelijk is aan de amplitude en de hoek gelijk aan de fase. In Matlab kunnen de functies `abs` en `angle` van pas komen om de amplitude en fase uit de fasor te halen.

Voorgaand experiment brengt een probleem aan het licht wanneer we de frequentie-inhoud analyseren van een signaal dat is opgedeeld in frames. Je kan zo'n frame beschouwen als een vermenigvuldiging van het signaal met een (verschoven) blokpuls. Aangezien men kan aantonen dat een vermenigvuldiging in het tijdsdomein overeenkomt met een convolutie in het frequentiedomein, gaat het opdelen van het signaal in frames een vervorming veroorzaken in het frequentiedomein. Om dit zoveel mogelijk te vermijden zal men *windowing* toepassen: de frames worden vermenigvuldigd met een zacht verlopend window alvorens de frequentie-analyse te doen. Figuur 6 geeft hiervan een schematische voorstelling. Populaire window-functies zijn het Hamming window en het Hann(ing) window, die in Matlab respectievelijk met de commando's `hamming` en `hanning` berekend kunnen worden.

Wanneer we te maken hebben met spraaksignalen zijn we doorgaans niet geïnteresseerd in de fase-informatie aangezien ons gehoor hier niet gevoelig aan is. De amplitudes van de verschillende frequenties zijn echter zeer typerend voor het geluid dat we horen en bijgevolg interessante features. Een voor de hand liggend feature is dus om de amplitudes van de verschillende frequenties in een vector te duwen en dit als feature vector te gebruiken. Aangezien we echter evenveel frequenties krijgen als tijdsamples, zal dit een lange vector zijn met een hoog-dimensionale feature-ruimte tot



Figuur 6: Framing & Windowing: opdelen van een signaal is stukjes voor frequentie-analyse

gevolg. De complexiteit die hiermee gepaard gaat is eigenlijk zinloos aangezien ons gehoor niet die frequentie-resolutie heeft. Een beter feature wordt bekomen door het frequentiedomein op te delen in een aantal banden. Als feature-vector kan je dan de energie per band gebruiken.

Om de energie per band te bepalen kan je gebruik maken van het theorema van Parceval, dat zegt dat de energie van een frequentieband kan bepaald worden als de som van de kwadraten van de amplitudes in die band:

$$E(f_1, f_2) = \int_{f_1}^{f_2} |X(f)|^2. \tag{3}$$

Je kan het frequentiebereik op verschillende manieren in banden verdelen:

- Lineaire verdeling: alle frequentiebanden even breed.
- MEL verdeling: het frequentiebereik wordt getransformeerd naar de MEL-schaal met behulp van formule 4, waarop dan een lineaire verdeling wordt toegepast. Dit heeft tot gevolg dat frequentiebanden op hogere frequenties breder zijn, wat de frequentie-gevoeligheid van het menselijk gehoor beter nabootst.

$$m = 1000 \times log_2 \left(1 + \frac{f}{1000}\right) \tag{4}$$

3.4 Linear Predictive Coding

Wanneer je een klank bekijkt in het tijdsdomein, dan merk je dat dit typisch een periodisch signaal is (vooral bij de klinker-klanken). Met andere woorden: zodra we een periode kennen, kunnen we de rest van het signaal min of meer voorspellen. Dit is precies wat Linear Predictive Coding (LPC) doet: het huidige sample wordt geschreven als een lineaire combinatie van een reeks vorige samples plus een fout:

$$x[k] = \sum_{i=1}^{P} a_i x[k-i] + e[k]. \tag{5}$$

Om de coëfficienten $a_i,\ i = 1 \ldots P$ te bepalen die de fout e minimaliseren, schrijven we deze vergelijking voor verschillende samples:

$$\begin{array}{rcl} x[k] & = & a_1 x[k-1] + a_2 x[k-2] + \ldots + a_P x[k-P] + e[k] \\ x[k-1] & = & a_1 x[k-2] + a_2 x[k-3] + \ldots + a_P x[k-1-P] + e[k-1] \\ x[k-2] & = & a_1 x[k-3] + a_2 x[k-4] + \ldots + a_P x[k-2-P] + e[k-2] \\ \vdots & & \\ x[k-M] & = & a_1 x[k-M-1] + a_2 x[k-M-2] + \ldots + a_P x[k-M-P] + e[k-M] \end{array} \tag{6}$$

of in matrixvorm:

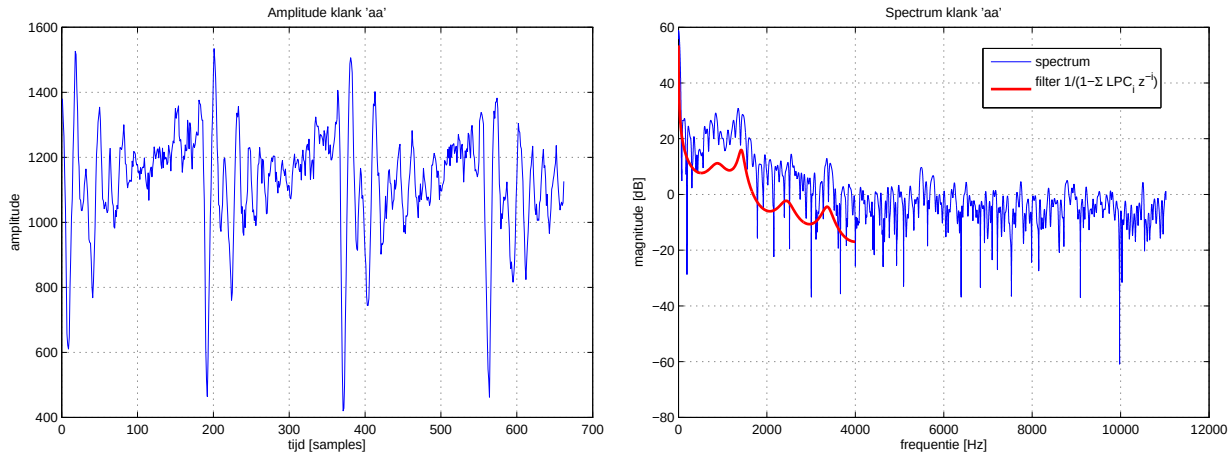
$$\mathbf{x} = \mathbf{X}\mathbf{a} + \mathbf{e}. \tag{7}$$

Op voorwaarde dat we voldoende vergelijkingen hebben, kunnen we de coëfficienten bepalen door het stelsel $\mathbf{x} = \mathbf{X}\mathbf{a}$ op te lossen via kleinste kwadraten methode: $\mathbf{a} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{x}$. Deze procedure is eenvoudig zelf te programmeren en ook geïmplementeerd in de Matlab functie `lpc`.

De coëfficienten-vector $\mathbf{a}$ is een mogelijke kandidaat om als feature-vector op te treden. Men kan namelijk aantonen dat deze coëfficienten de stand van de mond modelleren. Wanneer we een klank produceren sturen we een luchtstroom door onze mond die dan dienst doet als klankkast: een filter voor de luchtstroom afkomstig van de longen. Afhankelijk van de stand van de mond zal er een andere klank ontstaan. Het error-signaal $e[k]$ is dan het signaal dat de luchtstroom voorstelt. Hier kan eventueel ook nog een interessant feature uit geëxtraheerd worden. Wanneer dit signaal een pulserend verloop heeft (duidelijk periodisch is), dan wijst dit op het trillen van de stembanden en hebben we te maken met een stemhebbende klank. Als dit signaal geen pulserend verloop heeft (ruis), dan trillen de stembanden niet en hebben we te maken met een stemloze klank.

We kunnen dit filter-model voor het spraakkanaal ook in het frequentiedomein interpreteren. Het blijkt dan dat de coëfficienten $\mathbf{a}$ een filter voorstellen dat in het frequentiedomein de omhullende vormt voor het spectrum van het spraaksignaal. Een voorbeeld voor de klank 'aa' is weergegeven in figuur 7. LPC is dus nauw verbonden met een frequentie-analyse en we kunnen dan ook best de frames windowen om vervormingen tegen te gaan, zoals we bij de frequentie-analyses in sectie 3.3 deden. De interpretatie hiervan voor LPC in het tijdsdomein is dat de eerste samples in een frame moeilijk voorspeld kunnen worden aan de hand van de nullen die het frame voorafgaan. Door de windowing gaan die dan minder hard doorwegen bij de schatting van de coëfficienten.

De frequentiedomein interpretatie laat ook toe om onze voorspelling nog een beetje te verbeteren. Een spraaksignaal heeft typisch een verloop waarvan de omhullende afneemt met 20dB per decade (1ste orde verloop). De omhullende zou echter beter geschat worden als hij een vlak verloop zou



(a) (b)

Figuur 7: (a) Amplitude plot voor de klank 'aa' gedurende 30ms. (b) Frequentie spectrum en LPC filter voor de klank 'aa'. Op een verticale verschuiving na geeft het LPC filter de omhullende van het spectrum.

hebben. Daarom wordt dikwijls een pre-emphasis toegepast op het signaal alvorens de LPC-analyse te doen. Dit is een filteroperatie waarbij het signaal door een 1ste orde hoogdoorlaatfilter wordt gestuurd:

$$x_{pre}[k] = x[k] - hx[k-1], \qquad h = 0.9 \ldots 1. \tag{8}$$

3.5 Tijdsinformatie

Ook tijdsinformatie kan als feature aangewend worden. Het idee hierachter is dat de ademhaling van nieuwslezers gerelateerd is aan hun longinhoud. De momenten dat er wordt ingeademd vinden niet willekeurig plaats, maar wanneer de nieuwslezer in ademnood komt. De tijd die verstreken is sinds de vorige keer dat er werd ingeademd, is daarom een zinvol feature. Je kan nog een stap verder gaan en stellen dat de tijd die de nieuwslezer kan overbruggen gerelateerd is aan hoe diep hij inademde, wat dan weer gerelateerd is aan hoe luid en hoe lang het inademen de vorige keer werd waargenomen.

3.6 Cepstrale energie (optioneel)

Het filter-model voor het spraakkanaal (5), waarbij een luchtstroom $e[k]$ wordt gefilterd door een filter met coëfficienten $\mathbf{a}$, kan in het frequentiedomein voorgesteld worden als een vermenigvuldiging:

$$|X(f)|^2 = |A(f)|^2 \times |E(f)|^2, \tag{9}$$

met $X(f)$, $A(f)$ en $E(f)$ de frequentiedomeinvoorstellingen van respectievelijk het spraaksignaal $x[k]$, het filter met coëfficienten $\mathbf{a}$ en de luchtstroom $e[k]$. Met het LPC-model werd getracht om het filter $A(f)$, aan de hand van de filtercoëfficienten $\mathbf{a}$, uit het spraaksignaal te extraheren, aangezien dit als indicatie van de stand van de mond een interessant feature is om ademhaling te detecteren.

Een cepstrale analyse is een alternatieve manier om het filter A(f) te isoleren. Deze analyse wordt door de meeste spraakherkenners gebruikt om klanken te classificeren. Het voordeel ten opzichte van een LPC-analyse is dat de features die hieruit voortvloeien beter spreker-onafhankelijk zijn en beter te scheiden zijn in de feature-ruimte in vergelijking met LPC coëfficienten.

Om tot de definitie van het *cepstrum* te komen, nemen we de logaritme van vergelijking (9):

$$log(|X(f)|^2) = log(|A(f)|^2) + log(|E(f)|^2). \tag{10}$$

Het cepstrum bekomen we door deze vergelijking opnieuw in het frequentiedomein te bekijken. Men spreekt nu echter van het *quefrency domein* en men gebruikt niet de FFT maar de IFFT (beiden zijn equivalent, maar de IFFT geeft beter weer dat de eenheid van quefrency de seconde is):

$$IFFT(log(|X(f)|^2)) = IFFT(log(|A(f)|^2)) + IFFT(log(|E(f)|^2)). \tag{11}$$

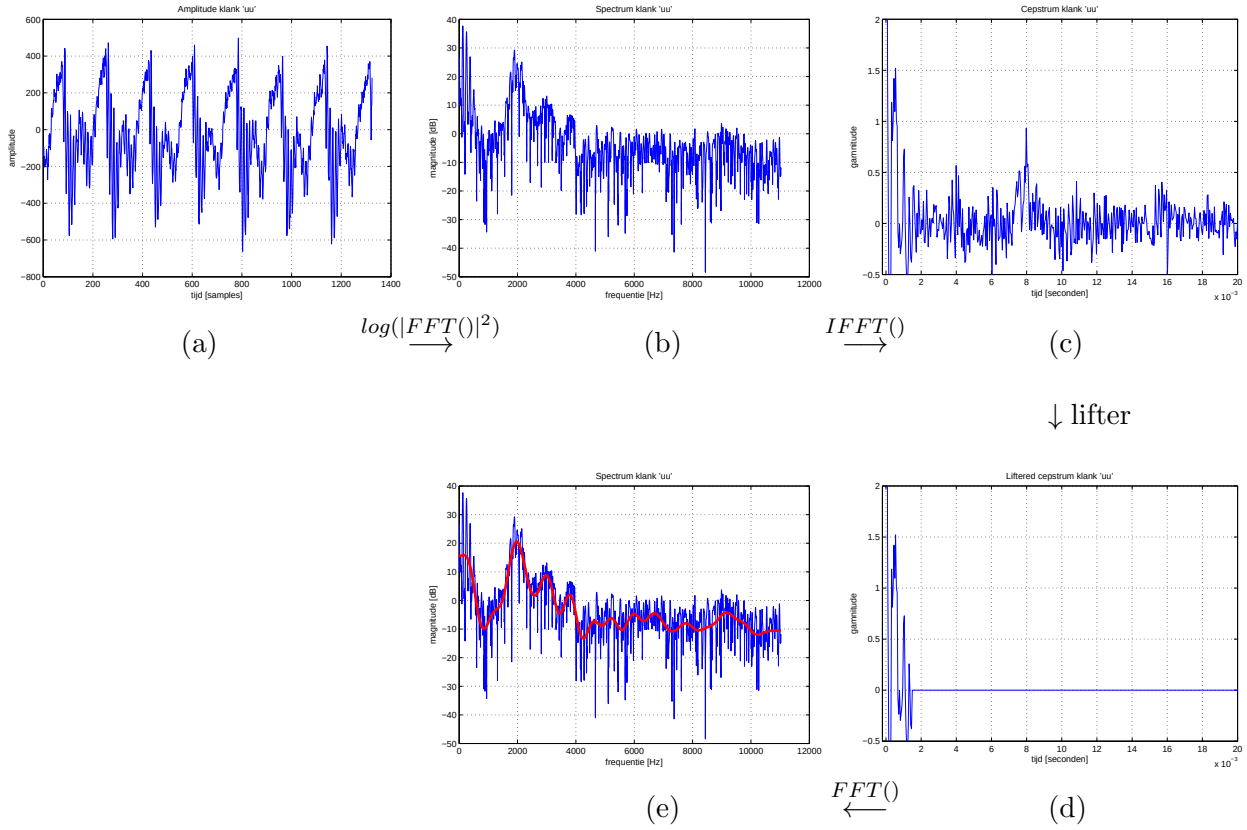
Aangezien $E(f)$ vooral hogerfrequente informatie bevat (trillen van de stembanden) dan $A(f)$, kunnen we $A(f)$ isoleren door enkel naar het eerste deel van de coefficienten van het cepstrum te kijken. Als we de informatie van de stembanden mee willen opnemen, gebruiken we alle coefficienten.

Dit is weergegeven in figuur 8 voor de klank 'uu'. Figuur 8(b) geeft het energie-spectrum weer in de gebruikelijke dB-schaal. Het cepstrum is niets anderes dan de IFFT hiervan en is weergegeven in figuur 8(c). De 'laagfrequente' informatie in de eerste $\pm$2ms heeft betrekking tot $A(f)$. De 'hoger frequente' informatie bevat $E(f)$. Merk hier de piek op 8ms in het quefrency domein: dit komt overeen met een frequentie van 125Hz, de pitch van het signaal die tot $E(f)$ behoort. Wanneer we ons beperken tot de laagfrequente informatie en alle cepstrale coëfficienten boven 1.5ms op nul zetten, dan merken we dat we de belangrijkste informatie nog altijd overgehouden hebben door terug over te schakelen naar het frequentiedomein: de overgehouden coëfficienten beschrijven nog altijd de globale vorm van het energie-spectrum en zijn dus typerend voor de klank.

Fishers linear discriminant analysis ?

4 Real-time implementatie (7 lessen)

Tot hiertoe werd de classificatie van een frame gedaan door de features te vergelijken met de features van alle trainings-frames. Naarmate we meer trainingsdata gaan gebruiken zal de complexiteit daardoor sterk toenemen. Daarom gaan we op zoek naar een manier om een frame te classificeren waarvan de complexiteit (tijd dat het in beslag neemt) onafhankelijk is van de hoeveelheid trainingsdata.



Figuur 8: (a) Amplitude plot voor de klank 'uu' gedurende 60ms. (b) Frequentie spectrum. (c) Cepstrum. (d) Cepstrum beperkt tot de coefficienten van de eerste 1.5ms. (e) De FFT van het gelimiteerd cepstrum geeft de trend van het spectrum weer.

4.1 Bayesian classificatie (2 lessen)

Bayesian classificatie maakt geen gebruik van de eigenlijke trainingsdata, maar kansverdelingen die de trainingsdata modelleren. De complexiteit wordt daardoor verschoven van de eigenlijke beslissing naar het opstellen van de modellen, wat vooraf kan gebeuren en waarvoor de hogere complexiteit dus van mider belang is. Centraal staat de regel van Bayes:

$$P(c|F) = \frac{P(F|c)P(c)}{P(F)}. \tag{12}$$

Hierin is $P(c)$ de kans dat een bepaalde klasse c optreedt, $P(F)$ de kans dat een bepaalde feature-vector F optreedt, $P(F|c)$ de kans dat feature-vector F optreedt als we weten dat dat deze vector een object van klasse c voorstelt en $P(c|F)$ de kans dat een object tot klasse c behoort als we een feature-vector F hebben.

Door de kans $P(c|F)$ uit te rekenen voor alle mogelijke klassen c kunnen we beslissen tot welke klasse een object met feature-vector F behoort door die klasse te nemen waarvoor $P(c|F)$ maximaal

is:

$$c_{best} = \underset{c \in C}{\operatorname{argmax}} P(c|F). \tag{13}$$

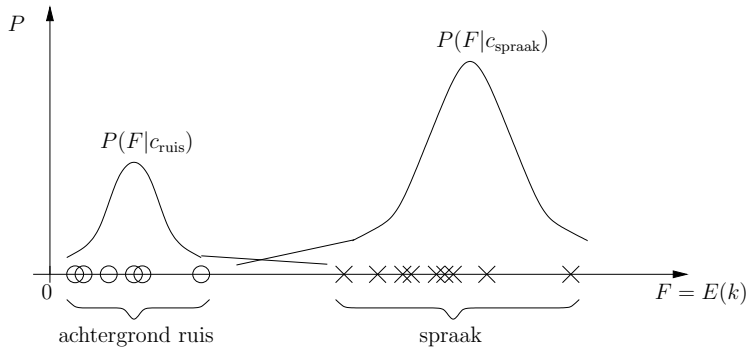
Met de regel van Bayes kan dit herleid worden tot

$$c_{best} = \underset{c \in C}{\operatorname{argmax}} \frac{P(F|c)P(c)}{P(F)}. \tag{14}$$

Tot slot kan opgemerkt worden dat $P(F)$ onafhankelijk is van de klasse en dus genegeerd kan worden: dit is slechts een scalering die voor alle klassen dezelfde is. Bayesian classificatie gebeurt dan aan de hand van de volgende formule:

$$c_{best} = \underset{c \in C}{\operatorname{argmax}} P(F|c)P(c). \tag{15}$$

Om dit toe te passen op de VAD gebaseerde spraakdetector uit sectie 2.2, kunnen we kijken naar figuur 9. We hebben 2 mogelijke klassen c: c_{spraak} en c_{ruis}. Om de classificatie te doen berekenen



Figuur 9: VAD gebaseerd op Bayesian classificatie

we voor elke klasse de 'kans' (het is geen echte kans, wegens het negeren van de scalering met $P(F)$ is de som niet gelijk aan 1):

$$\begin{aligned} \hat{c}_{\text{spraak}} &= P(F|c_{\text{spraak}})P(c_{\text{spraak}}) \\ \hat{c}_{\text{ruis}} &= P(F|c_{\text{ruis}})P(c_{\text{ruis}}) \end{aligned} \tag{16}$$

Als $\hat{c}_{\text{ruis}} > \hat{c}_{\text{spraak}}$ kunnen we beslissen dat we te maken hebben met ruis. Bayesian classificatie laat ook toe om meer zekerheid in te bouwen in de beslissing. Als we minstens $\alpha\%$ zekerheid willen hebben vooraleer we een frame als ruis beschouwen, moet volgende conditie voldaan zijn:

$$\frac{\hat{c}_{\text{ruis}}}{\hat{c}_{\text{spraak}} + \hat{c}_{\text{ruis}}} > \alpha. \tag{17}$$

De berekeningen in (16) om te beslissen of een frame spraak of ruis is, zijn op zich niet complex en kunnen snel uitgevoerd worden. De complexiteit werd verschoven naar het bepalen van de kansverdelingen uit de trainingsdata:

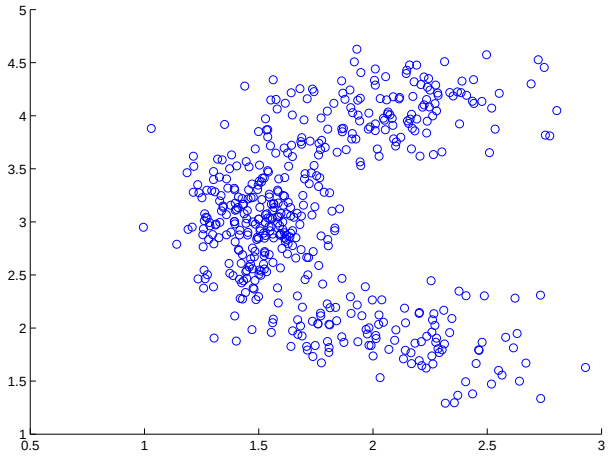
- $P(c_{\text{spraak}})$ en $P(c_{\text{ruis}})$ zijn eenvoudig te bepalen: hoeveel procent van de trainingsdata was spraak en hoeveel procent ruis.

- $P(F|c_{\text{spraak}})$ en $P(F|c_{\text{ruis}})$ zijn kansverdelingen die we moeten schatten. Voor de eenvoud gaan we ervan uit dat we te maken hebben met Gaussiaanse verdelingen. We gebruiken de trainingsdata om voor beide klassen het gemiddelde μ en de variantie σ^2 te berekenen.

Opdracht: Maak een VAD gebaseerde spraakdetector op basis van Bayesian classificatie.

4.2 Gaussian Mixture Models (5 lessen)

Naarmate de dimensie van de feature-ruimte hoger wordt, is het steeds minder aannemelijk dat de data Gaussiaans verdeeld is. Figuur 10 geeft een voorbeeld voor 2 dimensies. In dergelijke gevallen zullen we ons moeten behelpen met meer complexe modellen voor de kansverdelingen $P(F|c)$.



Figuur 10: Data die niet Gaussiaans verdeeld is.

Een model dat in vele toepassingen gebruikt wordt, is het Gaussian Mixture Model (GMM). Deze kansverdeling bestaat uit een gewogen som som van deelgaussianen:

$$f_{GMM}(\mathbf{x}) = \sum a_i f_i(\mathbf{x}), \qquad f_i(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i) \tag{18}$$

met a_i de gewichten en $f_i(\mathbf{x})$ de deelgaussianen met als gemiddelde de vector $\boldsymbol{\mu}_i$ en covariantie-matrix Σ_i.

Om de deelgaussianen te bepalen, moet de data verdeeld worden in kleinere groepjes waarvoor we wel een Gaussiaanse verdeling kunnen opstellen. Om de data te clusteren in k groepjes kan je het *k-means algoritme* gebruiken:

1. Selecteer (eventueel random) k datapunten. Dit worden de voorlopige centra van de clusters.
2. Bereken voor alle datapunten de afstand tot de centra van de clusters. Het datapunt behoort tot die cluster waarvoor de afstand tot het centrum het kortste is.
3. Bereken het gemiddelde voor elke cluster. Dit worden de nieuwe centra voor die clusters.

4. Als de centra van de clusters significant gewijzigd zijn, herhaal dan vanaf stap 2.

Zodra de data verdeeld is in clusters, fitten we op elke cluster i een meerdimensionale Gaussiaanse verdeling

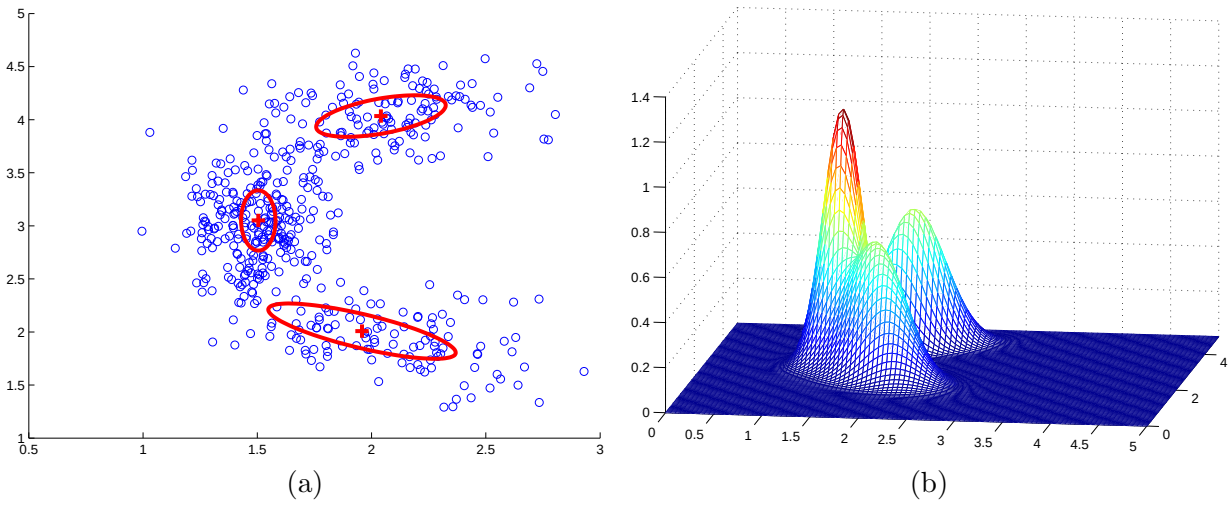
$$f_i(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right) \tag{19}$$

waarin N de dimensie van de verdeling is. De parameters $\boldsymbol{\mu}_i$ en Σ_i kunnen als volgt berekend worden uit de J datapunten $\mathbf{x}_j$ die tot cluster i behoren:

$$\boldsymbol{\mu}_i = \frac{1}{J}\sum_{j=1}^{J} \mathbf{x}_j \quad \text{en} \quad \Sigma_i = \frac{1}{J}\sum_{j=1}^{J} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T . \tag{20}$$

Tot slot worden de gewichten a_i bepaald aan de hand van het percentage van de datapunten dat in cluster i zit.

Figuur 11 toont het eindresultaat van het fitten van een GMM op de data uit figuur 10. Er werden 3 clusters gebruikt. In de praktijk zal je een beetje moeten experimenteren met het aantal clusters tot je merkt dat je GMM de data voldoende goed fit. In ons geval dus wanneer de Bayesian classificatie goed werkt.



Figuur 11: (a) Een GMM gefit op de data van figuur 10. De kruisjes geven $\boldsymbol{\mu}_i$ aan, de ellipsen $3\Sigma_i$. De verdeling met gemiddelde (1.5,3) heeft gewicht 0.46, de andere 2 hebben gewicht 0.27. (b) 3D weergave van het GMM.

Matlab voorziet in zijn Statistics Toolbox een meer geavanceerde manier om een Gaussian Mixture Model te schatten (op basis van het EM-algoritme). Om een GMM op te stellen aan de hand van gegeven data gebruik je `gmm = gmdistribution.fit(data,nbClusters);` om daarna de kans voor een bepaalde feature-vector `F` te berekenen aan de hand van `P = pdf(gmm,F);`

Opdracht: Bouw een ademhalingsdetector op basis Bayesian classificatie met GMMs. Beperk je tot de features uit sectie 3 die goede classificatie eigenschappen vertoonden. Experimenteer met het aantal clusters die nodig zijn om een goede performantie te krijgen. Vergelijk het classificatievermogen van de Bayesian classifier met die van de afstand-gebaseerde classifier.

5 Demonstratie (4 lessen)

5.1 Bonus feature: tijdsinformatie

Als je demo volledig klaar is en er is nog tijd over, kan je eens nadenken hoe je tijdsinformatie als feature kan gebruiken. Het idee hierachter is dat de ademhaling van nieuwslezers gerelateerd is aan hun longinhoud. De momenten dat er wordt ingeademd vinden niet willekeurig plaats, maar wanneer de nieuwslezer in ademnood komt. De tijd die verstreken is sinds de vorige keer dat er werd ingeademd, is daarom een zinvol feature. Je kan nog een stap verder gaan en stellen dat de tijd die de nieuwslezer kan overbruggen gerelateerd is aan hoe diep hij inademde, wat dan weer gerelateerd is aan hoe luid en hoe lang het inademen de vorige keer werd waargenomen.