Large-Scale Kernel Logistic Regression for Segment-Based Phoneme Recognition

P. Karsmakers[a,1,*], K. Pelckmans[b], H. Van hamme[a], J.A.K. Suykens[a]

[a] *K.U.Leuven, Department of Electrical Engineering, B-3001 Heverlee, Belgium*
[b] *Uppsala University, Department of Information Technology, SE-751 05 Uppsala, Sweden*
[c] *K.H.Kempen (Association K.U.Leuven), Department IBW, B-2440 Geel, Belgium*

Abstract

This paper describes an efficient implementation of multi-class Kernel Logistic Regression (KLR) suitable for large-scale data sets which are common in speech recognition. Unlike the Support Vector Machine (SVM), KLR directly yields a-posteriori probabilities of membership for each of the classes and has a natural extension to the multi-class case. Compared to SVM, in our implementation of KLR the model sparseness is easy controllable. As a result much sparser models can be obtained while still having comparable classification accuracies. A second focus of the paper is the actual integration of the large-scale KLR algorithm into a segment-based speech recognizer architecture. We motivate and validate the model for its use in a phoneme recognition system and indicate that the use of unlabeled (universum) data in training the KLR model improves the final recognition performance.

Keywords: Learning systems, large-scale systems, nonlinear estimation, pattern classification, sequential decoding, speech recognition.

1. Introduction

In the statistical- and machine learning community one distinguishes between two main groups of classifiers, namely generative and discriminative ones [5]. The current state-of-the-art in speech recognition is based on Hidden Markov Models (HMMs) which may be categorized into the first category. Using Bayesian decision theory to determine the class membership, requires a-posteriori probabilities which in HMMs are approximated by class-conditional descriptions of likelihoods. Discriminative classifiers on the other hand focus directly on estimating the classification rule or the a-posteriori probabilities of the class labels conditioned on the input data. No attempt is made to model all probability rules

*Corresponding author, Kasteelpark Arenberg 10, B-3001, Heverlee,Belgium; Tel: +32 16 32 17 09; Fax: +32 16 32 19 86

Email addresses: peter.karsmakers@esat.kuleuven.be (P. Karsmakers), kp@it.uu.se (K. Pelckmans), hugo.vanhamme@esat.kuleuven.be (H. Van hamme), johan.suykens@esat.kuleuven.be (J.A.K. Suykens)

generating the data. In general, they are more robust in classifying data as generative ones, since less assumptions about the classes are made.

In the shadow of the success of Support Vector Machines (SVM) [66] another well-founded method in this category is (penalized) Logistic Regression (LR) [2],[26]. In this paper, particular emphasis is put on a non-linear "kernelized" variant called Kernel Logistic Regression (KLR) [30]. Compared to SVM, (K)LR yields a-posteriori probabilities of membership in each of the classes based on a maximum likelihood argument instead of direct risk minimization and has a well-described multi-class extension. In [76] it has been shown that kernel logistic regression has related properties, and is comparable to SVM when it comes to classification accuracy.

In an Automatic Speech Recognizer (ASR) context KLR might offer the following advantages: (i) it directly estimates a-posteriori probabilities which fit in the probabilistic framework utilized by the state-of-the-art ASR systems, (ii) the extension to the multi-class case is well described, which must be contrasted to the commonly used coding approach (see e.g., [55],[62]), (iii) learning is formulated as a convex optimization resulting in one unique solution, (iv) KLR provides a well-defined mechanism to control the bias-variance trade-off, even in high dimensional spaces [60] which means that compared to HMM methods, more features extracted from a speech signal might be used for classification, (v) any (customized) kernel, as long as it is positive semi-definite [1], can be used.

In order to successfully integrate KLR into an ASR framework the following issues are addressed: (i) Although there are fast and efficient implementations available (see e.g. [59], [34]), the classical dual formulation of KLR does not scale to large problems which are common in ASR. In contrast to SVM the optimization of the Kernel Logistic Regression objective does not lead to a sparse model; (ii) Speech signals have variable lengths, while KLR is most often used as a static classifier, meaning that the observations are assumed to be fixed-dimensional vectors; (iii) A speech signal corresponds to a sequence of words, and each word relates to an unknown portion of the speech signal. Thus, speech recognition is a sequence labeling problem, with unknown segmentation, while logistic regression is designed to predict a single label only.

Firstly, a solution is formulated to the problem of scalability. Different adaptations to the original KLR algorithm were performed to obtain sparseness such as in [76]. In this paper we describe a different practical technique, suited for large data sets, based on fixed-size Least Squares Support Vector Machines (LS-SVMs) [62], which we can use because KLR is related to a weighted version of LS-SVMs [31]. To solve the approximate problem we use the optimization algorithm described in [41] as a basis. In this part our contributions are:

- The extension of the optimization algorithm mentioned above to the natural multi-class KLR setting.
- The comparison of two different types of multi-class schemes for our approximate algorithm in terms of training complexity, classification performance, sparseness and evaluation speed.
- The introduction of a reduced memory implementation of our approximate algorithm.

All of these result in a practical and efficient classification algorithm which allows to easily manage the sparsity of the resulting model.

Secondly, by extracting an equal number of features from the variable-length speech signals commonly used static vector kernels such as the linear, polynomial, RBF, etc., can be employed. It is this approach that was used in this manuscript. However, note that the proposed algorithm can be extended in a straightforward manner to allow to work directly on variable-length sequences by means of using customized kernels (see e.g. [12]).

Thirdly, the sequence labeling problem is tackled using a segment-based speech recognition framework in which our approximate KLR algorithm is integrated. We chose this particular framework: (i) Since it has the potential to alleviate some drawbacks of conventional frame-synchronous HMM ASR systems. These systems assume conditional independence between the frames (vectors which describe short time windows of speech) which might be violated since adjacent frames, especially those within the same phonetic segment, often exhibit smooth dynamics and are highly correlated. This must be contrasted to the segment-based approach which makes fewer conditional independence assumptions between observation frames especially for those occurring within a phonetic segment [22]; (ii) Since in segment-based ASR typically larger speech windows are considered, possibly more features need to be extracted. Including additional features increases the dimensionality of the input space which can decrease performance of the traditional HMM systems while kernel-based classifiers such as KLR have a robust mechanism which generalizes well even in these high dimensional spaces.

In segment-based recognition experiments the task of segmenting the full speech utterance into smaller units emerges in parallel to classifying individual segments. For practical reasons this parallel task is usually split up into two dedicated components: (i) a unit (phoneme) component which hypothesizes an utterance given a set of assumably correct segments and (ii) a segmentation component which judges the segmentation. During decoding also incorrect segments which have no valid interpretation as lexical units (i.e., too long, too short, etc.) are possible. In the SVM literature these are called universum data points [69]. The segmentation component is trained such that these non-phonetic segments are assigned a low probability. Our contributions here are

- The integration and validation of our KLR implementation in a segment-based speech recognizer.
- The introduction of a technique which allows to incorporate information about incorrect segmentations. This technique is based on a regularization argument and does not add an extra "universum" model. We experimentally point out that this improves the final phoneme recognition rate.

In order to have a clear understanding of our algorithmic setup we now explain the notations used throughout the paper. Matrices and vectors are respectively denoted by capital and lowercase letters. Scalars and sets are uppercase letters. Random variables are denoted in boldface. Subscripts select parts of objects. In case differentiation is needed between

functions of the same type we also use a subscript, meaning should be clear from its context, e.g. f_c denotes the function linked to class c. If we select the ij-th element of a matrix then we use the notation Ω_{ij}. A shorthand notation for selecting a subset of rows from a matrix or vector is $X(\mathcal{B}_i)$ where $\mathcal{B}_i \in \mathbb{N}^l$ is a set of indices. The number of elements in a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$. A superscript (k) marks symbol differences between algorithmic iterations. Because of the multi-class structures we need a convenient way of representing data point and class indices at the same time for a certain vector as is done in [59]. For instance $w_{ci} \in \mathbb{R}^{CD}$ selects the i-th component linked to a model c. In case such a double indexed vector is used its components are ordered as $w = (w_{11}, \ldots, w_{1D}, \ldots, w_{C1}, \ldots, w_{CD})^T$ or in case a $\sim$ is used, e.g. $\tilde{p}$, component ordering is as follows $\tilde{p} = (\tilde{p}_{11}, \ldots, \tilde{p}_{C1}, \ldots, \tilde{p}_{1N}, \ldots, \tilde{p}_{CN})^T$. The notation "." indicates a full index set. For example $w_{c.}$ selects all components which correspond to the c index. Further we write, 1 the vector of all ones, 0 the vector of all zeros, I the identity matrix, the indicator function as $\mathcal{I}(y_i = j)$ equals 1 if y_i is equal to j otherwise it is 0. For a matrix X, $(\mathrm{diag}(X))_i = X_{ii}$ extracts the diagonal elements. For a vector x, $\mathrm{diag}(x)$ is the corresponding diagonal matrix. The same is done for matrices, $\mathrm{blockdiag}(X_1, ..., X_C)$ which represents a block-diagonal matrix with matrix blocks defined by X_i. In case of indicating sequences of ordered units we use underlined capitals such as $\underline{U} = (u_1, ..., u_{N_u})$. When context is clear indices are dropped.

This paper is organized as follows. In Section II some insights are given in the precise statistical setup of linear multi-class logistic regression, and its non-linear extension kernel logistic regression. Additionally a method is provided to incorporate universum data based on a regularization argument. In the next Section we give details about an approximate large-scale KLR implementation. Section IV formally describes the integration of our KLR algorithm into a segmental phoneme recognition framework. In Section V the approximate KLR algorithm is empirically verified on phoneme classification and recognition experiments on the `timit` data set. Finally, a conclusion is drawn in Section VI.

2. Modeling with multi-class kernel logistic regression

2.1. Linear multi-class logistic regression

At first, we give some insight in the precise statistical setup. Suppose we have a random variable $(\mathbf{X}, \mathbf{Y})$ taking values in $\mathbb{R}^D \times \{1, \ldots, C\}$ where D denotes the dimensionality of the input vectors, and C denotes the number of classes. The aim of logistic regression is to produce an estimate of the a-posteriori probability of membership in each of the classes for the given random vector. The multi-class logistic model is given as

$$\begin{cases} \Pr(\mathbf{Y} = 1 \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_1(x))}{\sum_{c=1}^{C} \exp(f_c(x))} \\ \Pr(\mathbf{Y} = 2 \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_2(x))}{\sum_{c=1}^{C} \exp(f_c(x))} \\ \vdots \\ \Pr(\mathbf{Y} = C \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_C(x))}{\sum_{c=1}^{C} \exp(f_c(x))}, \end{cases} \tag{1}$$

where $f = \{f_c\}_{c=1}^C, f_c : \mathbb{R}^D \to \mathbb{R}$ consists of C different functions. In the linear case we assume the functions can be written as

$$f_c(x) = w_{c.}^T x + w_{c0}, \quad c = 1, \ldots, C, \tag{2}$$

where $w \in \mathbb{R}^{CD}$, $w = (w_{11}, \ldots, w_{1D}, \ldots, w_{C1}, \ldots, w_{CD})^T$, $w_{c.}$ refers to the elements of w linked to class c and w_{c0} is the intercept term of class c. Note that this double indexing, as also explained in the introduction, will be repeatedly applied in the remainder of the text. For notational convenience in the sequel, unless noted otherwise, the x vectors are augmented with a 1 such that the intercept term is incorporated in the parameter vector $w_{c.}$ as the first component. Given this model, one can compute easily the class probabilities for a new sample $x^* \in \mathbb{R}^{D+1}$. If one is to single out a class belonging to x under the logistic model, the best one could do (achieving the Bayes risk), is the following prediction rule

$$\underset{c \in \{1,2,\ldots,C\}}{\arg\max} \ \Pr(\mathbf{Y} = c | \mathbf{X} = x^*; f_1, ..., f_C). \tag{3}$$

Now we are interested in how to infer the parameters (functions) of the logistic model given a set of observations of $(\mathbf{X}, \mathbf{Y})$. Specifically, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a set consisting of N i.i.d. samples from the above logistic model. The classical method for doing inference of the statistical model is based on maximizing the likelihood

$$\ell(f_1, ..., f_C) = \prod_{i=1}^N \Pr(\mathbf{Y} = y_i | \mathbf{X} = x_i; f_1, ..., f_C)^{\frac{1}{N}}, \tag{4}$$

over the unknown functions f_i. Since we will later consider functions which are very flexible, it was argued that a penalized version of maximum likelihood is more appropriate in [16]. The penalized negative log likelihood (PNLL) criterion is defined as follows [1]

$$\ell^\nu(f_1, ..., f_C) = -\frac{1}{N} \ln \left(\prod_{i=1}^N \Pr(\mathbf{Y} = y_i | \mathbf{X} = x_i; f_1, ..., f_C) \right) + \frac{\nu}{2} \sum_{c=1}^C w_{c.}^T w_{c.},$$

where ν is a regularization parameter, which controls the bias-variance trade-off [19], that is tuned using techniques such as cross-validation. Let us now specialize to the logistic multi-class model and the functions f_c:

$$\ell_{LR}^\nu(w) = \frac{1}{N} \sum_{c=1}^C \sum_{i \in \mathcal{D}_c} \left(-w_{c.}^T x_i + \ln(e^{w_{1.}^T x_i} + ... + e^{w_{C.}^T x_i}) \right) + \frac{\nu}{2} \sum_{c=1}^C w_{c.}^T w_{c.},$$

[1] For notational convenience we only used a single ν for all parameters including the intercept. The same reasoning holds for 2 parameter matrix $\Psi = \text{diag}\left((\nu', \ 1^T \nu)^T \right)$ and corresponding regularization term $\sum_{c=1}^C w_{c.}^T \Psi w_{c.}$. Resulting in a separate hyper-parameter ν' which regularizes the intercepts $w_{.0}$ and ν which controls the regularization of the other parameters.

where $\mathcal{D}_c$ is a subset of the full training data set $\mathcal{D}$ with data examples belonging to the class c such that $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_C$, $\mathcal{D}_i \cap \mathcal{D}_j = \varnothing, \forall i \neq j$ and if $y_i = c$ then $x_i \in \mathcal{D}_c$. In the sequel we use the shorthand notation

$$\tilde{p}_{ci} = \Pr(\mathbf{Y} = c | \mathbf{X} = x_i; w), \tag{5}$$

where $\tilde{p} = (\tilde{p}_{11}, \ldots, \tilde{p}_{C1}, \tilde{p}_{1N}, \ldots, \tilde{p}_{CN}), \tilde{p} \in \mathbb{R}^{CN}$ ($\sim$ indicates a specific type of component ordering which is different to that of e.g. w). Since $\ln(\sum_{c=1}^{C} e^{w_{c\cdot}^T x})$ is strictly convex (see [7]), the PNLL criterion for penalized logistic regression is strictly convex in the parameters w.

Notice that adding the penalization term as in (5) poses the following symmetrical constraint on the weights w (see [77])

$$\sum_{c=1}^{C} w_{c\cdot} = 0. \tag{6}$$

Note that this constraint (penalization) uniquely identifies the parameters w.

Remark 1. In case no penalization is used as in (4) the parameters w are not unique since if we add a common term w' the probability estimates of the former model will not change. In the literature it is often seen that authors make the parameters identifiable by choosing a reference class. This is done by setting $w_{c^{ref}\cdot} = 0$ where c^{ref} indicates a reference class. Now, the regression coefficient $w_{ci}, c \neq c^{ref}$ can be interpreted as the log odds ratio of class c versus the reference class c^{ref}. Unfortunately, the solution to this formulation depends on the specific choice of reference class [36].

2.2. Multi-class kernel logistic regression

In this section we describe a non-linear extension using kernels to previous models called Multi-class Kernel Logistic Regression (MKLR). Instead of using an appropriate *Representer Theorem* [37] for dealing with representing kernels, an optimization argument and associated primal-dual formulation is used. The latter formulation has the advantage that it can give insights on how to build approximate models in the primal space. In [31] it is shown that MKLR can be fit in terms of a scheme of iteratively re-weighted Least Squares-SVMs (irLS-SVM). Note that in [76] the relation of KLR to Support Vector Machines (SVM) is stated. Next we summarize the primal-dual formulation. The details are given in Appendix A.

The objective function in (5) can be advanced with a nonlinear extension to kernel machines where the inputs x are mapped to a high dimensional space. Define $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ as the feature map induced by a positive semi-definite kernel. With the application of the Mercer's theorem for the kernel matrix Ω as $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j), i, j = 1, \ldots, N$ it is not required to compute explicitly the nonlinear mapping $\varphi(\cdot)$ as this is done implicitly through the use of positive kernel functions K. Note that for notational convenience by writing $\varphi(x)$ a, by 1 extended version of the transformed non-augmented vector x is meant. In the experiments we will only use the radial basis function kernel (RBF-kernel)

$$K(x, x') = \exp(-||x - x'||_2^2 / \sigma^2), \tag{7}$$

where σ is a tuning parameter. However, note that also other kernels can be used as long as they are positive semi-definite [1]. In the kernel version of LR the C models are defined as in equation (1) but x is mapped to a high dimensional space, i.e. $f_c(x) = w_{c.}^T \varphi(x)$. Note that to simplify the notation a single mapping $\varphi(\cdot)$ is used for each class, but that class-dependent mappings $\varphi_c(\cdot)$ are also possible.

The minimization of objective function (5) can be reformulated as an iteratively re-weighted least squares (IRWLS) problem, which is also described in [45][55][76] . This gives the primal formulation at iteration k

$$\min_{s^{(k)}} \frac{1}{2}||Fs^{(k)} - \tilde{q}^{(k)}||^2_{W^{(k)}} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}), \tag{8}$$

where the definitions of weights matrix $W^{(k)}$, regressor vector $\tilde{q}^{(k)}$ and multi-class feature matrix F are respectively given in (A.6), (A.10), and (A.12) in Appendix A.

To conveniently write down a dual formulation of this problem we use the following notation for the multi-class kernel matrix $G = FF^T$. The solution to the MKLR problem can then be found by iteratively solving the linear system

$$\left(\frac{1}{\nu}G + (W^{(k)})^{-1}\right)\tilde{\alpha}^{(k)} = \tilde{q}^{(k)} + \frac{1}{\nu}G\tilde{\alpha}^{(k-1)}. \tag{9}$$

The probabilities of a new point x^* given by C different models can be predicted using (1) and

$$f_c(x^*) = w_{c.}^T\varphi(x^*) = \frac{1}{\nu}\sum_{i=1}^{N}\tilde{\alpha}_{ci}^{(L)}K(x_i, x^*), \tag{10}$$

where the $\tilde{\alpha}^{(L)}$ are the estimated parameters obtained at the final iteration L when minimizing objective function (8), and $\tilde{\alpha}^{(k)} = (\tilde{\alpha}_{11}^{(k)}, \tilde{\alpha}_{21}^{(k)}, \ldots, \tilde{\alpha}_{C1}^{(k)}, \ldots, \tilde{\alpha}_{1N}^{(k)}, \ldots, \tilde{\alpha}_{CN}^{(k)})^T$.

2.3. Multi-class coding schemes

Deriving the multi-class implementation as described above results in one large (all-at-once) learning problem without the use of a coding scheme. Besides the all-at-once scheme other possible multi-class implementations can be built by combining several independent binary classifiers via a common coding scheme approach. Some examples include minimum output coding, error correcting output codes [62], decision directed acyclic graphs (DAGs) [52], one-versus-all and one-versus-one output coding. The latter is often used in case of large scale problems since the large-scale problem is split up in several smaller ones. We therefor chose to compare this to compare the one-versus-one approach to the all-at-once multi-class scheme in terms of training complexity, classification performance, sparseness and evaluation speed.

We now briefly review the one-versus-one approach which was used in this manuscript. In one-versus-one output coding one uses $C(C-1)/2$ smaller classifiers, where each binary model discriminates between two opposite classes in a pairwise manner. To combine the individual classifier results to decide upon the class label several methods can

be used such as voting [25]. Since KLR generates probabilities for each pairwise classifier and we would like to obtain a-posteriori probabilities as described in (5) we opted to use a more complex, possibly better (see e.g. [14]), pairwise coupling combination scheme. This idea was first introduced in [24] were the a-posteriori probabilities $\tilde{p}$ are obtained by minimizing the Kullback-Leibler divergence between the pairwise probabilities $\mu_{c_1c_2}(x_i) = \Pr(\mathbf{Y} = c_1 | \mathbf{Y} = c_1 \text{ or } \mathbf{Y} = c_2, \mathbf{X} = x_i)$ and $\mu'_{c_1c_2}(x_i) = \frac{\tilde{p}_{c_1 i}}{\tilde{p}_{c_1 i} + \tilde{p}_{c_2 i}}$. In this text we used a related approach as is proposed in [71] (referred in that paper as second approach) which is simpler in both practical implementations and algorithmic analysis. Note that the latter technique is also used in the LIBSVM toolbox [40].

2.4. Inference with a universum

In some cases we are given a set of example data as well as non-example data that do not belong to any other class. The latter collection, called universum [69], allows one to model prior knowledge. We suppose the universum to contain data from the same domain as the problem at hand and expect it to represent meaningful information related to the classification task. More specifically, our segmental speech recognition setup is based on classifying fixed-length segmental feature vectors representing a phoneme which assumes a given segmentation of the phoneme utterance. During the evaluation phase the models are confronted with fixed-length segmental feature vectors which are generated using a bad segmentation. Hence, vectors from the same domain which do not belong to any phoneme class are generated. We expect to achieve better performance when this type of information is incorporated into the training process.

In the context of SVMs a framework that is an alternative to the large margin concept is introduced in [66]. In case of SVM one selects the function out of an equivalence class $\mathcal{G}$, which is a set of functions having the same labeling on the non-universum training data, that maximizes the margin. As an alternative one could measure the number of contradicting functions in the equivalence class. A data point x^* from the universum data makes a contradiction on the equivalence class $\mathcal{G}$ if in $\mathcal{G}$ there are two functions $f_1(x)$ and $f_2(x)$ such that $f_1(x^*) > 0$ and $f_2(x^*) < 0$. In [69] this idea of maximizing the number of contradictions is approximately implemented by penalizing the function outputs $f(x_i^*)$ for universum data which are far from the decision boundary more than those which are close it. As if only a small change in $f(x_i^*)$ will cause a contradiction on x_i^*. This corresponds to minimizing the distance of the universum data to the classifier boundary.

A similar approach can be used to extend our MKLR algorithm by adding the following term

$$\sum_{c=1}^{C} \sum_{i=1}^{|\mathcal{U}|} L(f_c(x_i)), \tag{11}$$

which favors maximal contradiction on universum data to the KLR objective in (8). The set $\mathcal{U}$ is called the universum and L denotes the loss function. KLR outcomes have a probabilistic interpretation, hence minimizing the distance of universum points to the decision boundary can be seen in this context along similar lines as maximizing the entropy of the MKLR model outcomes on the universum data. Maximum entropy is achieved when $\tilde{p}_{ci} = 1/C$ for

$c = 1, ..., C$. This is the case when all function values $f_c(x), \forall x \in \mathcal{U}$ are equal (see (1)). This requirement is only met when $f_c(x) = 0$, for $c = 1, ..., C$ which can be seen in Section 2.1 where it is remarked that the L_2 norm penalty in (8) implies the symmetric constraint $\sum_{c=1}^{C} w_c(x) = 0$ which can be written as $\sum_{c=1}^{C} f_c(x) = 0$ in case it is multiplied by x. Hence, minimizing $\sum_{c=1}^{C} \sum_{i=1}^{|\mathcal{U}|} L(f_c(x_i))$ results in maximizing the entropy and thus minimizing the distance of universum data to the decision boundary.

Several loss functions are possible. We choose here to use the L_2 loss function which gives:

$$\begin{aligned} \sum_{c=1}^{C} \sum_{i=1}^{|\mathcal{U}|} L(f_c(x_i)) &= \sum_{c=1}^{C} \sum_{i=1}^{|\mathcal{U}|} (f_c(x_i))^2 \\ &= \sum_{c=1}^{C} w_{c.}^T \sum_{i=1}^{|\mathcal{U}|} \left(\varphi(x_i) \varphi(x_i)^T \right) w_{c.}. \end{aligned} \tag{12}$$

Note that in case of SVMs, adding a regularizer term results in a maximization of the margin. In KLR adding regularization has similar properties (see [76]). Hence, by adding the term (12) we change the original objective for defining the decision boundary to a trade-off between the usual regularization on the training (non-universum) data and maximizing the entropy of the universum data. This trade-off is controlled by the hyper-parameter $\nu^{\mathcal{U}}$. In the Section 5 we experimentally indicate that the inclusion of universum data can improve the accuracy in terms of correct classifications and the likelihood of test data, over labeled data alone.

3. Large-scale implementation

This section describes a collection of practical insights which were crucial to give our implementation of KLR the characteristics needed for speech data sets.

3.1. Fixed-size approach: estimation in the primal weight space

In the following subsection we review how to estimate an approximation to the feature map $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ using the Nyström method [70]. The estimated eigenfunctions will then be used to solve the problem in primal space.

3.1.1. Approximation to the feature map

Suppose one takes a finite dimensional feature map (e.g. a linear kernel). Then one can equally well solve the primal as the dual problem. In fact solving the primal problem is more advantageous for larger data sets common in speech recognition because the dimension of the unknowns $w \in \mathbb{R}^{CD}$ is smaller compared to that of $\tilde{\alpha} \in \mathbb{R}^{CN}$. In order to work in the primal space using a kernel function other than the linear one, it is required to compute an explicit approximation of the nonlinear mapping $\hat{\varphi} : \mathbb{R}^D \to \mathbb{R}^M$, such that $\Omega_{ij} \approx \hat{\varphi}(x_i)^T \hat{\varphi}(x_j)$ with $M \ll N$. This leads to a sparse representation of the model when estimating in primal

space. Note that this approximation of the feature map is a preprocessing step and therefor the vectors x, $\varphi(x)$ nor $\hat{\varphi}(x)$ are augmented with a 1 in this section.

Explicit expressions for φ can be obtained by means of an eigenvalue decomposition of the kernel matrix Ω with entries $K(x, x')$. Given the integral equation $\int K(x, x')\phi_i(x)p(x)dx = \lambda_i\phi_i(x')$, with solutions λ_i and ϕ_i for a variable x with a probability density $p(x)$, we can write

$$\varphi(x') = \left(\sqrt{\lambda}_1\phi_1(x'), \sqrt{\lambda}_2\phi_2(x'), \ldots, \sqrt{\lambda}_{D_\varphi}\phi_{D_\varphi}(x')\right)^T. \tag{13}$$

such that $K(x, x') = \varphi(x)^T\varphi(x') = \sum_i \lambda_i\phi_i(x)^T\phi_i(x')$ holds.

Note that D_φ can be infinite when for instance an RBF kernel is used. The Nyström method [70] approximates the integral by means of a sample average

$$\lambda_i\phi_i(a) \approx \frac{1}{N}\sum_{l=1}^{N} K(x_l, a)\phi_i(x_l), \forall a \in E \subseteq \mathbb{R}^D. \tag{14}$$

Let $a = x_j$, in matrix notation one obtains the eigenvalue problem,

$$\Omega U = U\Lambda, \tag{15}$$

where $U = (u_1, \ldots, u_N)$ is the square matrix of eigenvectors of Ω, $u_i = ((u_i)_1, \ldots, (u_i)_M)^T$, $\Lambda = \mathrm{diag}(\lambda_1^s, \ldots, \lambda_N^s)$ is a diagonal matrix of nonnegative eigenvalues in decreasing order, and $\Omega_{ij} = K(x_i, x_j)$ are the elements of the kernel matrix. The eigenvalues λ_i and eigenfunctions ϕ_i from the continuous problem in expression (13) can be approximated by the sample eigenvalues λ_i^s and eigenvectors $u_i \in \mathbb{R}^N$ as

$$\lambda_i \approx \frac{1}{N}\lambda_i^s, \phi_i \approx \sqrt{N}u_i. \tag{16}$$

Substituting (16) in (14) results in an approximation of an eigenfunction evaluated in point $a \in E \subseteq \mathbb{R}^D$,

$$\hat{\phi}_i(a) = \frac{\sqrt{N}}{\lambda_i^s}\sum_{j=1}^{N}(u_i)_j K(x_j, a). \tag{17}$$

Based on this approximation, an expression for the entries of the approximated feature map $\hat{\varphi}_i : \mathbb{R}^D \to \mathbb{R}, i = 1, \ldots, N$ for a training point, or for any new point x^*, with $\hat{\varphi}(x^*) = (\hat{\varphi}_1(x^*), \ldots, \hat{\varphi}_N(x^*))^T$, is given as

$$\begin{aligned}\hat{\varphi}_i(x^*) &= \sqrt{\lambda_i}\hat{\phi}_i(x^*) \\ &= \frac{1}{\sqrt{\lambda_i^s}}\sum_{j=1}^{N}(u_i)_j K(x_j, x^*).\end{aligned} \tag{18}$$

Until now the entire training set of size N is used to compute the approximate feature map. Therefore $\hat{\varphi}(x)$ will yield at most N components, each one of which can be computed by (18) for all $x_i, i = 1, \ldots, N$. However, if we have a large scale problem, it has been

motivated [62] to use a subsample of $M \ll N$ data points to compute the $\hat{\varphi}(x)$. In this case, up to M components will be computed. Functions $f_c(x)$ will be approximated as follows

$$\hat{f}_c(x) = \sum_{i=1}^{M} \left(w'_{ci} \frac{1}{\sqrt{\lambda_i^s}} \sum_{k=1}^{M} (u_i)_k K(x_k, x) \right) + w_{c1}, \tag{19}$$

where $w'_{c(i-1)} = w_{ci}$ for $i = 2, \ldots, D_{\hat{\varphi}} + 1$ and λ_i^s and u_i the corresponding sample eigenvalues and eigenvectors of a subspace of size $M \times M$ of the kernel matrix Ω_c.

3.1.2. Sampling procedures

The selection of M components, which we will generally call prototype vectors (PVs), to obtain $\hat{\varphi}(x)$ is a key issue in obtaining a good Nyström approximate of the full kernel matrix. Note that in case these PVs coincidence with a subset of the training data, they might be interpreted as Support Vectors (SVs). In this work we evaluate 3 approaches: random selection, Renyi entropy maximization and k-center clustering. The first approach is the simplest choice where subsamples are selected randomly from the available training data examples. Instead of random selection one can resort to the use of active selection procedures. This can be important for large scale problems, as it is related to the underlying density distribution of the sample. In this sense, the optimality of this selection is related to the final accuracy of the model. One such active method is the second option which is a selection based on an entropy criterion which tries to find a uniform subset of the training data as is proposed in [62] [8]. We also compare with a third selection method based on k-center clustering. In [75] the observation is made that the Nyström low-rank approximation depends crucially on the quantization error induced by encoding the sample set with landmark points. This suggests that one can simply use the clusters obtained with a k-center (such as k-means) algorithm, which finds a local minimum of the quantization error. A simple and fast greedy algorithm which approximates the k-center clustering problem is proposed in [23] and called farthest-point clustering. This algorithm works as follows: pick an arbitrary point $\bar{p}_0$ as the center of the first cluster and add it to the center set $\mathcal{P}$. Then, for $i = 1$ to k, do the following: at step i, for every point x, compute its distance to the set $\mathcal{P}$, $d(x_j, \mathcal{P}) = \min_{\bar{p}' \in \mathcal{P}} ||x_j - \bar{p}'||$. Set $\bar{p}_j = x_j$ and let x_j be the point that is farthest from $\mathcal{P}$, i.e., the point for which $d(x_j, \mathcal{P}) = \max_x d(x, \mathcal{P})$. Add $\bar{p}_i$ to set $\mathcal{P}$. Report the points $\bar{p}_0, \bar{p}_1, ..., \bar{p}_{k-1}$ as the cluster centers. Each point is assigned to its nearest center. A fast implementation in [17] has a complexity of $O(N \log k)$.

Remark 2. Note that prototype vectors selected by the first two presented methods are a subset of the training samples the k-center method introduces an extra degree of freedom because now the prototype vectors do not necessarily have to coincidence with the training samples.

3.1.3. Selecting the number of prototype vectors

An important task in this framework is to determine the number of prototype vectors M which are used to build the FS-MKLR model. A large M gives more modeling flexibility

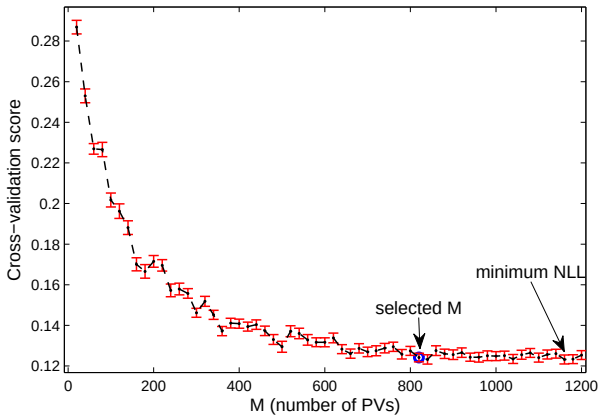


Figure 1: For a set of M values 10-fold cross-validation scores (based on a Negative Log Likelihood (NLL) criterion) are calculated for a tuned FS-MKLR model. On the figure both the mean and one tenth of the standard deviation of the 10 cross-validation measurements are given per M value. The final selected $M = 820$ is marked by the circle and was selected using the smallest value of M within one tenth of the standard deviation of the best cross-validation score (which is $M = 1,160$).

but slower evaluation times than a smaller M which gives less complex models but faster evaluation times. To determine an M a cross-validation score can be computed for all values of M up till $M = N$. For each different value for M, the hyper-parameters (e.g. regularization parameter ν, kernel band-width σ) are tuned using cross-validation. The best cross-validation score per M is stored and used to select the smallest value of M within one tenth of the standard deviation of the best cross-validation score (standard deviation was calculated from the different estimates computed using cross-validation). See [25] (Section 7.10, page 216) where a similar approach, referred to as the "one-standard-error", is given. However, exploring all possible values for M would be computationally too expensive for large-scale data sets. Therefore, a set of predefined values for M is chosen. Note that in order to exclude random effects in dividing the data the same cross-validation partitioning is kept for computing the cross-validation estimates for each value of M.

To illustrate this process, a subset of the `timit` training set ($N = 4,613$) is selected i.e. the examples belonging to phoneme classes /b/ and /d/ (the feature extraction is explained later in Section 5). Then for a set of M values 10-fold cross-validation scores (based on a Negative Log Likelihood (NLL) criterion) for a tuned FS-MKLR model are calculated. This is plotted in Figure 1 where both one tenth of the standard deviation and mean of the 10 cross-validation measurements (using tuned hyper-parameters) are given per M value. At a certain point adding more PVs does not improve the NLL based criterion which indicates that only a subset of the training points (called PVs) is needed to build a model. The final selected M is marked by the circle.

3.2. Relation to other methods

In the literature one often employs a greedy approach to obtain a sparser model than that of the original learning problem. The Import Vector Machine (IVM) [76], which is also

a sparse approximation to the KLR formalism, aims to minimize the negative log likelihood criterion iteratively by adding training samples to a subset of selected prototype vectors until the algorithm converges to some predefined value (as with M this parameter controls the sparseness of the model). Similar heuristics can be found for a sparse SVM algorithm in [35] and for sparse kernelized least-squares loss functions in [68].

The computation of those greedy methods requires to (re-)estimate the learning problem for all training samples (excluding the ones already selected as PV) in every iteration where the number of iterations equals the number of finally selected prototype (support) vectors. Although for most algorithms one can reuse information from previous iterations they still remain computationally expensive. Therefore they all implement a probabilistic speed-up which takes a random sample of the complete learning set (not already as PV selected training samples). In [33] the authors conclude by stating that the main problem of IVM is the computationally expensive estimation of the model parameters such that it is not applicable for large data sets.

In [56] a sparse KLR solution is achieved by replacing the penalization term in (8) by the L_1 norm of the dual unknowns $\tilde{\alpha}$. In [28] an adaptation of maximum likelihood estimation, instantiated on logistic regression is proposed. The model outputs proper conditional probabilities into a user-defined interval and is less precise elsewhere. Instead of selecting a parameter M one has to define the user-defined interval.

Note that LR as discussed in Section 2.1 can be derived from an entropy maximization framework as denoted [4]. In this context the logit models are known as MaxEnt models and entropy maximization is performed using moment constraints. In [72] a method is proposed to additionally formulate distribution constraints which are more suited for continuous features. In order to derive a convenient objective for optimization purposes the authors provide a solution by converting the input data to a higher dimensional space similar to the what is performed by the feature map $\varphi(\cdot)$ in our case. To achieve this, cubic-splines are used instead of kernels. Analogous to choosing a number of PVs, in their work the number of spline knots need to be specified. As in this text this number of knots is selected based on a validation set criterion. Our proposed framework has the advantage that it can be used with any positive semi-definite kernel function.

3.3. Solving the problem in primal space: Newton Trust Region approach

In this work the sparse kernel logistic regression problem is solved in the primal space and therefor one can reside to optimization algorithms used for linear logistic regression. In the literature there are many different methods for solving the logistic regression problem. All these optimizations are iterative procedures which can be distinguished by the trade-off being made between cost per iteration and speed of convergence.

Given the finite dimensional approximation to the feature map $\hat{\varphi}$, which is defined as in (17) where instead of N only $M \ll N$ samples are used to approximate φ, the problem in (8) can be solved in the primal weight space with unknown w using the Newton's based method which iteratively updates the weight vector $w^{(k)}$,

$$w^{(k)} = w^{(k-1)} + \tau s^{(k)}, \tag{20}$$

over k until convergence where τ is the step size and $w^{(k)}$ the vector of all parameters in the k-th iteration. In each iteration the step $s^{(k)} = -{H^{(k)}}^{-1} g^{(k)}$ can be computed where the gradient and Hessian are respectively defined as

$$g^{(k)} = \begin{pmatrix} \frac{\partial \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.}} \\ \vdots \\ \frac{\partial \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.}} \end{pmatrix},$$

and

$$H^{(k)} = \begin{pmatrix} \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.} \partial w^{(k)}_{1.}} & \cdots & \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.} \partial w^{(k)}_{C.}} \\ & \ddots & \\ \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.} \partial w^{(k)}_{1.}} & \cdots & \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.} \partial w^{(k)}_{C.}} \end{pmatrix}.$$

Applied to (8) this gives

$$g^{(k)} = \begin{pmatrix} \hat{\Phi}^T (\tilde{p}^{(k)}_{1.} - \tilde{r}^{(k)}_{1.}) + \nu w^{(k-1)}_{1.} \\ \vdots \\ \hat{\Phi}^T (\tilde{p}^{(k)}_{C.} - \tilde{r}^{(k)}_{C.}) + \nu w^{(k-1)}_{C.} \end{pmatrix}, \tag{21}$$

with $\tilde{r}$ as defined in (A.4) and $\hat{\Phi} = (\hat{\varphi}(x_1), ..., \hat{\varphi}(x_N))^T$, $\hat{\Phi} \in \mathbb{R}^{N \times (M+1)}$,

$$H^{(k)} = \begin{pmatrix} \hat{\Phi}^T T^{(k)}_{11} \hat{\Phi} + \nu I & \ldots & \hat{\Phi}^T T^{(k)}_{1C} \hat{\Phi} \\ & \ddots & \\ \hat{\Phi}^T T^{(k)}_{C1} \hat{\Phi} & \ldots & \hat{\Phi}^T T^{(k)}_{CC} \hat{\Phi} + \nu I \end{pmatrix}, \tag{22}$$

where

$$T^{(k)}_{ab} = \operatorname{diag}\left(t^{ab}_i, \ldots, t^{ab}_N\right), \tag{23}$$

with

$$t^{ab}_i = \begin{cases} a = b & \tilde{p}^{(k)}_{ai}(1 - \tilde{p}^{(k)}_{bi}) \\ a \neq b & -\tilde{p}^{(k)}_{ai} \tilde{p}^{(k)}_{bi}. \end{cases} \tag{24}$$

For large scale multi-class data this matrix might be too large to be stored. The size of the Hessian is proportional to C and the length of $\hat{\varphi}(\cdot)$ which is $D_{\hat{\varphi}} + 1$. Therefore certain iterative methods where the Hessian is always used in a product with a vector d are more appropriate here. This fact can be used to exploit the structure of the Hessian and therefor storage of the full Hessian is not needed. For Newton methods the conjugate gradient (CG) algorithm is mostly used for this purpose. Instead of performing all iterations until a stopping criterion is met, the loop can be interrupted earlier which is done in case of using truncated Newton methods as in [38]. Both issues mentioned above are tackled in the Newton Trust

Region algorithm described in [41]. This algorithm yielded the best performance compared to state-of-the-art alternatives. A good balance between convergence speed and cost per iteration is found in that low cost approximate Newton steps are taken in the beginning of the algorithm and full Newton directions at the end for fast convergence. We therefor used their optimization structure and extended it to the multi-class case.

At each iteration a second order Taylor approximate of ℓ_{LR}^{ν} subject to a trust region constrained is minimized which results in the following optimization problem

$$\begin{aligned} \hat{s}^{(k)} = &\arg\min_{s^{(k)}} g^{(k)T} s^{(k)} + \frac{1}{2} s^{(k)T} H^{(k)} s^{(k)} \\ &\text{such that } ||s^{(k)}||_2 < \Delta^{(k)}, \end{aligned} \tag{25}$$

at iterate $w^{(k)}$ and a trust region $\Delta^{(k)}$ to obtain a step direction $\hat{s}^{(k)}$. Then $w^{(k)}$ is updated according to (20) and $\Delta^{(k)}$ is updated by checking the ratio

$$\rho^{(k)} = \frac{\ell_{LR}^{\nu}(w^{(k+1)}) - \ell_{LR}^{\nu}(w^{(k)})}{g^{(k)T}\hat{s}^{(k)} + \frac{1}{2}\hat{s}^{(k)T} H^{(k)} \hat{s}^{(k)}}, \tag{26}$$

of the actual reduction in the function to the predicted reduction in the quadratic model as in [41]. The direction is accepted if $\rho^{(k)}$ is large enough:

$$w^{(k+1)} = \begin{cases} w^{(k)} + \hat{s}^{(k)} & \text{if } \rho^{(k)} > \eta_0 \\ w^{(k)} & \text{if } \rho^{(k)} \leq \eta_0, \end{cases} \tag{27}$$

where $\eta_0 > 0$ is some predefined value. Note that $\hat{s}^{(k)}$ differs from the Newton step $s^{(k)}$ given in (20) since it is calculated using the quadratic model in (25) which uses the trust region constraint.

In each iteration it is necessary to compute the probabilities $\tilde{p}^{(k)}$ using (1) where

$$\hat{f}_c(x) = w_{c.}^T \hat{\varphi}(x), \tag{28}$$

using (19) this can be written as

$$\hat{f}_c(x) = \frac{1}{\nu} \sum_{i=1}^{M} \tilde{\alpha}_{ci} K(x_i, x) + w_{c1}, \tag{29}$$

with

$$\tilde{\alpha}_{c.} = SUw'_{c.}, \tag{30}$$

with $S = \Lambda^{-\frac{1}{2}}$ where Λ is the diagonal matrix with the eigenvalues of a subspace (size $M \times M$) of Ω and $U = (u_1, \ldots, u_M)^T$ the matrix with corresponding eigenvectors.

The Fixed-Size MKLR (FS-MKLR) trust region algorithm is showed in Algorithm 1[1]. We terminate Algorithm 1 when the relative difference of the objective likelihood is smaller

[1]Note that in some cases it is possible that $\exp(f_c(x_i))$ reaches infinity in MATLAB. Especially in the tuning step were an automatic model selection framework is used, extreme hyper-parameter tuples occur. To circumvent this we simply add the following constraint to the function values: $f_c(x_i) \leq 300, \forall c, i$ as in (1),(28).

Algorithm 1 FS-MKLR

Input: training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$,$M$
Parameters: $w^{(k)}$
Output: probabilities $\Pr(\mathbf{X} = x_i|\mathbf{Y} = y_i; w^{opt}), i = 1, ..., N$ and w^{opt} is the converged parameter vector
Initialize: $k := 0$,$w^{(0)} = 0_{CM}$
Define: $g^{(k)}$ and $H^{(k)}$ according to resp. (21) (22).
Selection of M PVs using k-center clustering
compute features $\hat{\Phi}$ as in (17)
repeat
  $k := k + 1$
  compute $\Pr(\mathbf{X} = x_i|\mathbf{Y} = y_i; w^{(k-1)}), i = 1, ..., N$
  calculate $g^{(k)}$
  $\hat{s}^{(k)} = \arg\min_{s^{(k)}} g^{(k)T} s^{(k)} + \frac{1}{2} s^{(k)T} H^{(k)} s^{(k)}$ such that $||s^{(k)}|| \leq \Delta^{(k)}$ as in (25)
  compute $\rho^{(k)}$
  $w^{(k+1)} = w^{(k)} + \hat{s}^{(k)}$
  obtain $\Delta^{(k+1)}$
until convergence

than some threshold, or when the number of iterations reaches some maximum.

To solve the constrained minimization problem of (25) linear CG is used similarly as in [41]. The most CPU intensive task within linear CG is the product of the Hessian H and a vector $d = (d_{11}, \ldots, d_{1D}, \ldots, d_{C1}, \ldots, d_{CD})$ which can be efficiently implemented without explicitly storing the complete Hessian. Because the Hessian is structured as in (22) the resulting vector can be written as follows

$$\begin{pmatrix} \hat{\Phi}^T \left(T_{11}(\hat{\Phi} d_{1.}) + ... + T_{1C}(\hat{\Phi} d_{C.}) \right) + \nu d_{1.} \\ \vdots \\ \hat{\Phi}^T \left(T_{C1}(\hat{\Phi} d_{1.}) + ... + T_{CC}(\hat{\Phi} d_{C.}) \right) + \nu d_{C.} \end{pmatrix}, \tag{31}$$

where T_{ij} is defined as in (23). In a practical implementation first all vectors $h_{c.} = \hat{\Phi} d_{c.}$ are calculated and stored for $c = 1, ..., C$ then each component of Hd is calculated. Notice that a matrix-vector product in case of diagonal matrices as is with T_{C1} can be efficiently implemented as follows $h'_{i'} = (\mathrm{diag}(T_{ij}))_{i'} \, h_{ci'}$.

3.4. Very large data sets

For very large data sets, storage of the full $\hat{\Phi} \in \mathbb{R}^{N \times M}$ matrix can become problematic. The fastest implementation first stores the complete $\hat{\Phi}$ matrix in memory and then starts training in primal space. However, it is not necessary to fully store this matrix. At the cost of more processor time one can save memory by recomputing the $\hat{\Phi}$ in blocks at times it is necessary in the algorithm. It can be seen that $\hat{\Phi}$ appears at two different locations in

the algorithm. Firstly it is used in computing the gradient g in the main Newton loop and secondly to compute the matrix vector product Hd in the CG algorithm.

3.4.1. Gradient computation

Let N_b be a predefined maximum number of transformed input vectors which are allowed to be stored at once. Given N_b, the matrix X can be divided in Z blocks defined by the index sets $\mathcal{B}_i, \forall i = 1, \ldots, Z$, where $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$. Based on the fact that a matrix vector product can be decomposed as follows $\hat{\Phi}^T a = \hat{\Phi}(\mathcal{B}_1)^T a(\mathcal{B}_1) + \ldots + \hat{\Phi}(\mathcal{B}_Z)^T a(\mathcal{B}_Z)$, the gradient as defined in (21) can be computed as indicated by the pseudo code in 2. The features are now calculated in blocks and stored in a temporary matrix $\hat{\Phi}'^T$. In this case $\hat{\Phi}$ need not be fully stored at a cost of an increased computational cost due to recomputing features in $\hat{\Phi}'^T$.

Algorithm 2 Reduced memory implementation for gradient computation

Define: X is the matrix with input vectors, $\mathcal{B}_i$ is block index set, $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$ where Z equals the number of blocks, $a = \tilde{p} - \tilde{r}$
for $z = 1, ..., Z$ **do**
  compute features of $X(\mathcal{B}_z)$ and store in $\hat{\Phi}'$
  for $c = 1, ..., C$ **do**
    $g_{c.} := g_{c.} + \hat{\Phi}'^T a_{c.}(\mathcal{B}_z)$
  end for
end for
$g := g + \nu w$

3.4.2. Matrix-vector computation

Given the result in (31), a decomposition similar to that used in Algorithm 2, can be used to calculate the matrix-vector product Hd. This again reduces memory requirements to calculate the result. The pseudo code is given in Algorithm 3.

Algorithm 3 Reduced memory implementation for matrix-vector product Hd

Define: $X \in \mathbb{R}^{N \times D}$ is the matrix of test input vectors, $\mathcal{B}_k$ is block index set, $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$ where Z is the number of blocks, vector $d = (d_{11}, \ldots, d_{1D}, \ldots, d_{C1}, \ldots, d_{CD})$, and H according to (22).
for $z = 1, ..., Z$ **do**
  compute features of $X(\mathcal{B}_z)$ and store in $\hat{\Phi}'$
  for $c = 1, ..., C$ **do**
    $a_i = \sum_{j=1}^{C} (T_{cj})_{ii} (\hat{\Phi}' d_{j.})_i, \forall i$
    $h_{c.} := h_{c.} + \hat{\Phi}'^T a$
  end for
end for
$h := h + \nu d$

3.5. Computational complexity

The computational cost of the FS-MKLR algorithm (without the feature map approximation) is dominated by the matrix vector product Hd which has a complexity of $\mathcal{O}(CM(2N+C+1))$. This means that the matrix-vector product and hence the training procedure scales linearly in M and N and scales quadratically in C. A complete training run requires $k_1 k_2$ matrix-vector products. Where k_1 is the number of Newton iterations and k_2 is the number of CG iterations in Algorithm 1.

The complexity of calculating the Nyström approximation (with eigenvalue decomposition of the kernel matrix of size M) is $\mathcal{O}(M^3 + M^2 N)$. In case all features $\hat{\Phi}$ can be stored in memory this operation is only executed once independent of the number of classes. When the reduced memory implementation is used (see Algorithm 2, 3), the features are recomputed for each gradient and Hessian computation.

For a comparison with binary coupled classifiers we assume that the features fit in memory, an equal number of training samples N_c and an equal number of PVs M_c per class. The complexity term of FS-MKLR can then be written as $\mathcal{O}(C^2 M_c(2CN_c + C + 1)) = C^3(2M_c N_c + ((C+1)/C)M_c)$. For the pairwise approach, this reduces to $\mathcal{O}(1/2C(C-1)(2M_c(4N_c+3))) = 2(C^2-C)(2M_c N_c + (3/2)M_c)$. Thus, using a one-versus-one approach results in a reduction of computational costs inversely proportional to the number of classes.

If training time is considered the one-versus-one approach seems to be more suitable for practical use than FS-MKLR. However, if evaluation speed is considered then FS-MKLR is more interesting. A factor which dominates the evaluation speed is the number of unique PVs. Although a prototype vector may be associated to different binary classifiers, computer time is saved by calculating and storing all $K(x_i, x)$ first, where x_i are the selected unique prototype vectors and x is test data. Another factor is the total number of model parameters. This number will explode in case of one-versus-one coupled classifiers when the number of classes to consider increases. Due to this and the fact that FS-MKLR can produce sparser results with comparable accuracy as one-versus-one classifiers (see Section 5) the evaluation speed of FS-MKLR models is much faster opposed to one-versus-one models for problems with a large number of classes such as in phoneme recognition.

Remark 3. If the number of universum data examples is modest, adding the penalization term (12) does not have a major impact on the overall training complexity. If we assume a joint kernel the total costs are the calculation of the covariance matrix with complexity $\mathcal{O}(M|\mathcal{U}|^2 + M^2|\mathcal{U}|)$ (this is required only once at the beginning of the training procedure) and k_1 times $\mathcal{O}(CM^2)$ for computing the penalty (12).

4. Application of FS-MKLR in segment-based speech recognition

In this paper we choose to work in a segment-based context. In such systems variable-length segments are mapped to basic speech units. Compared to frame-synchronous ASR systems which extract features from short time intervals, segment-based systems use features computed using information from the complete segment. The reason why we adopted this frame-work is two-fold:

- The advantage of segmental over frame-based features is that they can include information about the correlation that exists among frames in the segment and its surroundings. The idea of segment-based modeling seems to have arisen around the early nineties in several different research groups [3], [39], [50]. Usually the false conditional independence assumption of Hidden Markov Models (HMMs) and their limitations in modeling segmental features especially duration were mentioned as the main motivation for developing these models [50].

- In order to describe a larger segment potentially more features are required than in standard frame-synchronous approaches. This motivates the use of kernel-based alternatives such as FS-MKLR and SVM which are known to generalize well even in high dimensional input spaces [60]. This gives the opportunity, opposed to the generative HMM approach, to build models with high predictive accuracy operating on a large set of features while having a relative small amount of data available.

After introducing some notation we will briefly review the formal statistical description of a speech recognition process.

Speech signals are first transformed to a sequence of vectors (called frames) $\underline{\mathrm{X}} = (x_1, \ldots, x_{N_x})$. Each frame contains features extracted from a time window of speech (e.g. 30 ms). Now, instead of recognizing the whole speech utterance at once (which is untractable in case of a large vocabulary) the process is decomposed into smaller jobs. Typically, a recognizer identifies a sequence of speech units which are later combined to words, sentences denoted by $\underline{\mathrm{W}} = (w_1, \ldots, w_{N_w})$. These sub-words are taken from an alphabet of basic units represented as $\mathcal{U}' = \{u'_1, \ldots, u'_C\}$, a sequence of basic units is noted as $\underline{\mathrm{U}} \in \mathcal{U}'^{N_u}, \underline{\mathrm{U}} = (u_1, \ldots, u_{N_u})$. Beforehand it is not known which frames belong to which unit. For this purpose a sequence of segments $\underline{\mathrm{S}} \in \mathcal{S}, \underline{\mathrm{S}} = (s_1, \ldots, s_{N_s})$ (with $\mathcal{S}$ the set of all possible segmentations of $\underline{\mathrm{X}}$) identifying the segmentation of the full frame sequence, which is used as a basis to recognize the units. A i-th segment respectively starts and ends in a frame with an index τ_{i-1} and τ_i. A unit u_i can have one or more segments depending on the model setup. Note the existence of different segmentations with the same corresponding unit sequence. For notational convenience we skip the random variables in the probability notation which is schematically summarized in Fig. 2.

The recognition process can now be formally described as a search for

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}, \underline{\mathrm{S}} | \underline{\mathrm{X}}), \tag{32}$$

where the sum over $\underline{\mathrm{U}}$ is only necessary in case of using multiple word pronunciations. In [50],[67] it is argued that depending on how (32) is decomposed, one can categorize segment-based recognizers into two broad classes: conditional and a-posteriori segment modeling formalism. We choose here to use the latter because it is naturally more appropriate for FS-MKLR (and other discriminative classifiers) to produce a-posteriori probabilities. Equation

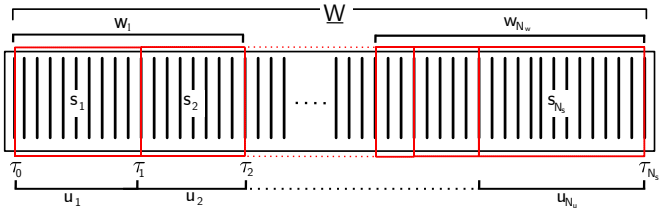


Figure 2: Time domain speech signals are first transformed to a sequence of vectors (called frames) $\underline{\mathrm{X}} = (x_1, \ldots, x_{N_x})$. These are presented in the figure by the black vertical lines. Each frame contains features extracted from a time window of speech (e.g. 30 ms). Now, instead of recognizing the whole speech utterance at once (which is untractable in case of a large vocabulary) the process is decomposed into smaller jobs. Typically, a recognizer identifies a sequence of phonemes (smallest distinguishable sounds) $\underline{\mathrm{U}} = (u_1, \ldots, u_{N_u})$ which are later combined to words, sentences denoted by $\underline{\mathrm{W}} = (w_1, \ldots, w_{N_w})$. Beforehand it is not known which frames belong to which phoneme. For this purpose a sequence of segments $\underline{\mathrm{S}} = (s_1, \ldots, s_{N_s})$ identifying the segmentation of the full frame sequence, which is used as a basis to recognize the phonemes. The i-th segment respectively starts and ends in a frame with an index τ_{i-1} and τ_i. Depending on the model setup a unit u_i is modeled by one or more segments. It should be clear that a certain sequence of phonemes can be identified by several different segmentations.

(32) can be decomposed as

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \frac{\Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}, \underline{\mathrm{S}}, \underline{\mathrm{X}})}{\Pr(\underline{\mathrm{X}})} \tag{33}$$

$$\approx \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}) \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}}) \sum_{\underline{\mathrm{S}}} \Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}) \tag{34}$$

where $\Pr(\underline{\mathrm{X}})$ can be ignored since it does not depend on $\underline{\mathrm{W}}, \underline{\mathrm{U}}$ nor $\underline{\mathrm{S}}$. It is also assumed that the basic units used are detailed enough such that all the information in $\underline{\mathrm{W}}$, which is relevant in determining a probability score on $\underline{\mathrm{S}}$ and $\underline{\mathrm{X}}$, is covered in $\underline{\mathrm{U}}$ ($\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{W}}) = \Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}})$). Using Bayes' rule we can rewrite $\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}})$ as

$$\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}) = \frac{\Pr(\underline{\mathrm{X}}) \Pr(\underline{\mathrm{U}}, \underline{\mathrm{S}}|\underline{\mathrm{X}})}{\Pr(\underline{\mathrm{U}})}, \tag{35}$$

where we again can ignore $\Pr(\underline{\mathrm{X}})$. Substituting this equation into (34) results in

$$\hat{\underline{\mathrm{W}}} \approx \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}) \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}}) \sum_{\underline{\mathrm{S}}} \frac{\Pr(\underline{\mathrm{S}}, \underline{\mathrm{U}}|\underline{\mathrm{X}})}{\Pr(\underline{\mathrm{U}})}. \tag{36}$$

The joint probability term $\Pr(\underline{\mathrm{S}}, \underline{\mathrm{U}}|\underline{\mathrm{X}})$ which can be found in (36) can be decomposed into,

$$\Pr(\underline{\mathrm{S}}, \underline{\mathrm{U}}|\underline{\mathrm{X}}) = \Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}, \underline{\mathrm{X}}) \Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}). \tag{37}$$

This factorization has some advantages:

- Dedicated statistical models can be trained for estimating the segmentation and the phoneme classification probabilities. Only the segmentation model has to be trained on all possible segmentations which can yield a reduction in training time (although we will indicate that using a technique based on a regularization argument to incorporate information of incorrect segmentations in the training process of the phoneme classification model leads to an improved recognition rate).

- The computational requirements of the recognition process can be reduced considerably by calculating the unit probabilities only for those candidate segmentations having a segmentation probability exceeding a predefined threshold.

4.1. Modeling the phoneme probability

The phoneme probability factor from (37) can be decomposed as follows

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}},\underline{\mathrm{X}}) = \prod_{i=1}^{N_u} \Pr(u_i|u_1,\ldots,u_{i-1},\underline{\mathrm{S}},\underline{\mathrm{X}}). \tag{38}$$

In case we assume a 2-gram phoneme language model and use segmental-based features, we can approximate further to

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}},\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_u} \Pr(u_i|u_{i-1}, f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})), \tag{39}$$

where τ_{i-1} and $\tau_i - 1$ respectively indicate the start and end indices of the i-th segment in sequence $\underline{\mathrm{X}}$ (see Figure 2).

Using Bayes' rule, the fact that features are only selected based on the frames of the current unit, and the assumption that we only condition on u_i and not on u_{i-1}, we can transform the latter approximate factor as follows

$$\begin{aligned}
\Pr(u_i|u_{i-1}, f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})) &= \frac{\Pr(u_i, f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})}{\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})} \\
&= \frac{\Pr(u_i|u_{i-1})\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_i,u_{i-1})}{\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})} \\
&\approx \frac{\Pr(u_i|u_{i-1})\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_i)}{\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))} \\
&\approx \frac{\Pr(u_i|u_{i-1})\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}),u_i)}{\Pr(f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))\Pr(u_i)} \\
&\approx \frac{\Pr(u_i|u_{i-1})\Pr(u_i|f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))}{\Pr(u_i)}.
\end{aligned} \tag{40}$$

Combining (39) and (40) gives

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}},\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_u} \frac{\Pr(u_i|u_{i-1})\Pr(u_i|f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))}{\Pr(u_i)}. \tag{41}$$

The denominator in (41) defines 1-gram probabilities. The first factor in the numerator defines a 2-gram language model. Note that this formula can be easily generalized to the N-gram case. In case we intend to model the second factor in the numerator by means of a kernel method using static kernels (e.g. RBF kernel), we have to tackle the technical problem that the segments are of different durations. Fortunately, it is quite straightforward to find the kind of segmental feature set that yields a segment classification rate similar to or better than those of HMM phoneme models. Such examples can be found in [21] and [11]. The mapping function f outputs a fixed length segmental feature vector which is used to relate segments to phonemes. One common type of segmental feature vector is a fixed length sampled version of the frame sequence corresponding to the considered segment. Also features describing the sufficient statistics of a segment given its frame sequence as used in [73] might be used. However, in this work we applied the technique as described in [11]. Here, each phoneme segment was broken into three regions using the ratio 3-4-3. Frames belonging to each of these regions were averaged resulting in three vectors. In addition, frames belonging to a window region of 50 ms width centered at the start and end of the phonetic segments were averaged. These 5 vectors plus a feature indicating the log-duration of the phoneme segment were stacked to one fixed length vector [2].

4.2. Modeling the segmentation probability

The segmentation model judges segmentations of a speech signal. Given the segmentation a unit sequence hypothesis can be generated using the phoneme model. There are several possibilities for estimating $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ (37). A possible solution is to execute a frame-based recognizer, retain the N best paths produced by it, and evaluate only these paths using the segmentation model [74]. In this case one can ignore the factor $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ by assuming that the segmentations proposed by the frame-based recognizer all have the same segmentation probability. Alternatively, in [39] the authors proposed to calculate the segmentation probability as a product of boundary probabilities, estimated using Multi-Layer Perceptrons (MLP). Instead of making a product in [43] one uses an additional MLP to estimate $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ given the boundary probabilities. A third option is to construct an estimate of $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ as follows

$$\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_u} \Pr(s_i|\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}), \tag{42}$$

[2] In case a phoneme segment contains only 1 frame then 3 copies of the same vector were used to represent the 3 phoneme regions. In case 2 frames are available they are assigned to the left and right region and there average is used to represent the mid-region.

where the equality holds if independence between different segments is assumed. To train such model one can construct an extra (also called anti-phone) class with universum data, i.e. data obtained from incorrect segmentations [21]. This extra class can then be used to build a dedicated binary classifier or to extend the phoneme classifier with an extra class. The universum training examples can be generated artificially over a manually segmented corpus [57] or in a corrective manner by providing the model with negative examples encountered during (mis)recognition [3].

This work focusses on building a FS-MKLR phoneme classifier, we therefor choose to use the likelihood scores, which we already have from our HMM recognizer, to estimate the segmentation probability. Note that other, possibly more suited, segment-based models could be used. We prefer this approach above that of using N best segmentations of the HMM recognizer because in this way the segmentation model as well as the phoneme model can jointly decide on the segmentation.

In case factor $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ is used in a maximization formulation, it can be replaced by $\Pr(\underline{\mathrm{X}}|\underline{\mathrm{S}})$ using Bayes' rule since $\Pr(\underline{\mathrm{X}})$ is constant and we assume $\Pr(\underline{\mathrm{S}})$ to be constant for all permitted $\underline{\mathrm{S}}$. A segment duration model in the form of a non-uniform $\Pr(\underline{\mathrm{S}})$ could refine the approach.

Using marginalization of the HMM recognizer likelihoods and by applying Bayes' rule again, we can formulate

$$\begin{aligned}\Pr(\underline{\mathrm{X}}|\underline{\mathrm{S}}) &= \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}, \underline{\mathrm{U}}|\underline{\mathrm{S}}) \\ &= \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}}) \Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}) \\ &\approx \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}}) \Pr(\underline{\mathrm{U}}), \end{aligned} \tag{43}$$

where factor $\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}})$ is approximated as $\Pr(\underline{\mathrm{U}})$. This might be used because we do not expect to gain much when conditioning the utterance sequence on the segmentation without knowledge of the acoustic vectors.

With (43) and incorporating the standard HMM independence assumptions, we can replace $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ by

$$\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_u} \sum_{c=1}^{C} \left(\Pr(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1}|s_i, u_i = u'_c) \Pr(u_i = u'_c) \right). \tag{44}$$

Note that $\Pr(\underline{\mathrm{U}})$ does not refer to the hypothesis which is searched for in maximization problem (36) but is used as a dummy variable to obtained the actual segmentation probability. It is therefor motivated to use unigram assumptions in this case which also simplifies the implementation of the search algorithm considerably.

4.3. Phoneme recognition

Substituting (41) and (44) into (37) gives the formulation of our phoneme recognition task

$$\hat{\underline{\mathrm{U}}} = \arg\max_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \prod_{i=1}^{N_u} \sum_{c=1}^{C} \left(\Pr(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1} | s_i, u_i = u_c') \Pr(u_i = u_c') \right) \frac{\Pr(u_i | f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1}))}{\Pr(u_i)} \Pr(u_i | u_{i-1}). \tag{45}$$

Note that in case of a path-algorithm as Viterbi to extract the most likely segmentation and corresponding phoneme sequence the sum over $\underline{\mathrm{S}}$ in (45) is replaced by the max operator over $\underline{\mathrm{S}}$.

In the experimental section the continuous speech recognition results are presented as phoneme error rates (PER) defined as

$$PER = \frac{N_I + N_D + N_S}{N_C} \tag{46}$$

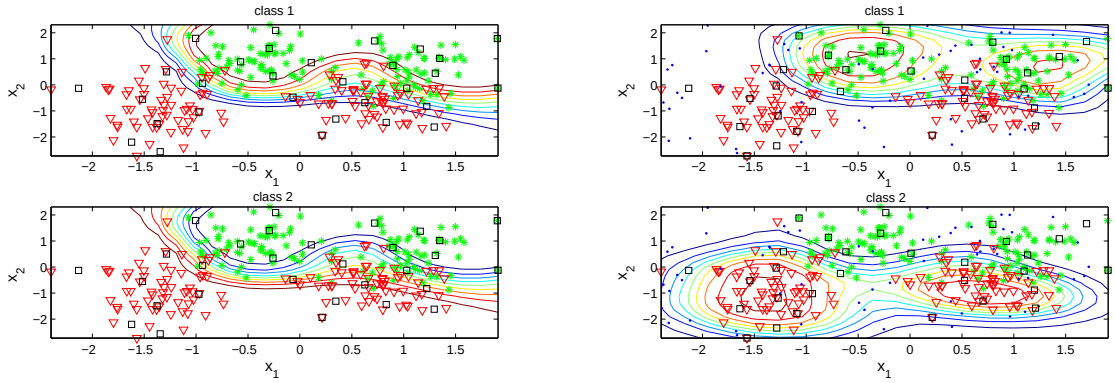
where N_I is the number of inserted phonemes, N_D is the number of deleted phonemes, N_S the number of substituted phonemes in the recognized phoneme sequence. N_C is the number of phonemes in the correct sequence.

Remark 4. Notice that in case a 1-gram language model is used, it would cancel out the denominator $\Pr(u_i)$ in (45). Hence, using it simplifies our model but deteriorates the PER as is observed in the experimental section.

Since there is information available about incorrect segmentations and since using the technique described in Section 2.4 to cope with universum data is not too computationally demanding we can easily incorporate the extra unlabeled data while constructing our phoneme model. Note that the number of phoneme model parameters is not increased by this technique which means that the time for evaluating new data remains the same as before. Besides discriminating the phonemes differently the unit probability factor now goes beyond its initial scope and can jointly decide on the segmentation together with the segment probability factor. Note that our approach differs from that of e.g. [21], [57] since we do not consider the universum data as an extra class (which would increase the model complexity) but rather use the universum information to define a criterion which decides where to locate the classification boundary of the phoneme model. In the next Section we experimentally indicate that adding information of the universum (incorrect segmentations) to build the phoneme classifier will improve the recognition rate.

5. Experiments and results

All FS-(M)KLR experiments in this section are carried out in MATLAB. We opted to use a separate hyper-parameter ν' which controls the regularization for all the intercept terms which was set to an arbitrary small value 10^{-9}. The hyper-parameters:



(a) (b)

Figure 3: In (a) the probability landscape of a tuned FS-MKLR on the Ripley data set is presented. When incorporating uniformly distributed noise (indicated with dots) as universum data into the model we obtained the result shown in (b). The first row presents the probability of the first class in function of the input values while the second row gives that of the second class. For both experiments an RBF kernel was used.

- ν : regularization parameter for model weight penalty
- σ^2 : RBF-kernel bandwidth

are tuned using 5-fold cross-validation with a NLL cost criterion and Nelder-Mead simplex [46] as the optimization algorithm. The hyper-parameter $\nu^{\mathcal{U}}$ which controls the universum penalty is additionally tuned using NLL (in Section 5.1) or PER (in Section 5.2) as a cost criterion.

5.1. Training with a universum penalty

First, we illustrate potential advantages of using the universum penalty explained in Section 2.4 using two initial experiments:

- Firstly, an FS-MKLR model is trained and tuned on Ripley's synthetic data [54] where we took uniformly distributed noise as universum data. Using the universum penalization term described in (12) we try to maximize the likelihood of the training data as well as minimizing the likelihood of the universum data as much as possible. The RBF-kernel is used. Without the information of the universum class we obtain a probability landscape for the first and second class respectively presented in the first and second row of Fig. 3a. In case we include the universum data and choose an appropriate tuning parameter $\nu^{\mathcal{U}}$, we obtain more compact probability landscapes as given in Fig. 3b. Note that the universum data is presented with dots.
- Secondly, instead of using random data universum examples should be collected to reflect information about the domain of our problem of interest. In the next experiment we build a binary classifier which has to discriminate between 2 letters selected from the `isolet` [65] data set. The remaining letters were used as universum data points.

Table 1: The table shows results for FS-MKLR and FS-MKLR$_{\mathcal{U}}$ using the same 200 PVs on a binary `isolet` classification problem. In case of FS-MKLR$_{\mathcal{U}}$ the other letters not selected as the binary couple were used as universum data. The hyper-parameters were tuned using the negative log likelihood as a criterion. Column err and llh represent respectively the misclassification rate and log likelihood on test data available for the 2 considered classes.

data	**FS-MKLR**		**FS-MKLR$_{\mathcal{U}}$**	
	$Err(\%)$	NLL	$Err(\%)$	NLL
N-M	11.93($\pm$0.70)	0.2326($\pm$0.0059)	**10.92**($\pm$**0.59**)	*0.227*($\pm$*0.0068*)
P-T	**5.00**($\pm$**0.00**)	0.2013($\pm$0.0046)	**5.00**($\pm$**0.00**)	*0.1442*($\pm$*0.0472*)
F-S	3.33($\pm$0.59)	0.0901($\pm$0.0022)	**3.00**($\pm$**0.75**)	*0.0801*($\pm$*0.0051*)
V-B	**4.33**($\pm$**0.37**)	0.122($\pm$0.0022)	4.50($\pm$0.75)	*0.1191*($\pm$*0.0028*)
B-D	**2.50**($\pm$**0.00**)	0.0776($\pm$0.0068)	**2.50**($\pm$**0.00**)	*0.0734*($\pm$*0.014*)

We selected 5 pairwise classification problems which produced the most confusion when considering the confusion matrix of a model obtained in a previous experiment. The RBF-kernel was used and 200 PVs were selected via the k-center method. Hyper-parameters σ^2,ν,$\nu^{\mathcal{U}}$ were tuned with NLL as the performance criterion. Results where averaged over 5 runs. The partitions as well as the selected PVs are kept constant to make a fair comparison possible between experiments with and without universum data. The results are shown in Table 1. For FS-MKLR$_U$ we observe that the misclassification rates in 4 out of 5 cases are similar or even better compared to FS-MKLR. If we consider the NLL score then FS-MKLR$_U$ has better scores for all cases.

5.2. Phoneme classification and recognition

To test the performance of FS-(M)KLR on speech data we used the `timit` database [64]. Training was performed on the 'sx' and 'si' training sentences. These create a training set with $3,696$ utterances from 168 different speakers. For testing we chose the full test set. It consists of $1,344$ utterances from 168 different speakers not included in the training set. All utterances contain labels indicating the phoneme identity and the starting and ending time of each phoneme. The standard Kai-Fu Lee mapping [42] was used, resulting in a set of 39 phonemes. For each of the experiments we converted the utterances from their waveform representation into a sequence of 36 dimensional observation vectors. These observation vectors were obtained by means of mutual information based discriminant linear transformation [13] on 24 MEL spectra (obtained using a window size of 25ms and shift of 10ms) and their first and second order time derivatives. If we refer to a state-of-the-art HMM phoneme recognizer it has the following configuration. In total 51 context-independent acoustical phoneme models (one model of silence is included) with 2 to 4 states (left-to-right models) where the number of states dependents on the average phoneme length. This gives a total of 141 states (from which 13 states are shared between 2 or 3 phonemes). In total $5,710$ Gaussians (average 131.2/state) were used. As explained in the introduction we chose to work using a static kernel. From this family we selected the RBF kernel (7) which will be

used in upcoming experiments since this kernel outperforms other alternatives as was seen in several previous publications [11],[18],[58].

5.2.1. Phoneme classification

Note that in [32], we already presented some experimental evidence for phoneme classification using FS-MKLR on the `timit` data set. However, several modifications to the FS-MKLR algorithm resulted in much higher accuracies which are reported here.

In case of phoneme classification we assume a given segmentation of speech utterances. The objective is then to find a statistical relation between the variable- length segments and phoneme labels. In order to use machine learning techniques such as K-Nearest Neighbors (KNN)[25], Linear Discriminant Analysis (LDA) [25], SVM (with static kernel) and FS-MKLR (with static kernel) on such variable-length segments, we must encode the waveform information as a fixed-length vector. We chose the same simple method of encoding the variable-length segment information in a vector of fixed length as in [11]. Each phoneme segment was broken into three regions in the ratio 3-4-3. The 36 dimensional vectors belonging to each of these regions were averaged resulting in three 36 dimensional vectors. In addition, the 36 dimensional vectors belonging to a window region centered at the end and start of the phonetic segments and with a 50 ms width were averaged, resulting in another two 36 dimensional vectors. One additional feature indicating the log-duration of the phoneme segment was added. This resulted in $142,910$ train and $51,681$ test examples with 181 dimensions.

In order to speed up our automatic tuning process we randomly selected a set of $20,000$ data points which is used for tuning the hyper-parameters using cross-validation. After tuning we used the full train set to train the algorithm using the obtained hyper-parameters. In case of FS-MKLR we ceiled the maximum number of PVs to $8,000$ and training on the full training set was performed using the reduced memory implementation (see 3.4) to make computation feasible on our computer. The SVM experiments are performed using the LIBSVM toolbox [40] where multi-class problems are encoded in a one-versus-one scheme.

The phonetic classification accuracies on the full `timit` test set using different classifiers are shown in Table 2. It is observed that the best result was obtained using SVM in case the pairwise coupling idea (SVM-pc) described as the second method in [71] is used to label unseen data. This gave an accuracy of 83.2% (using $89,563$ Support Vectors (SVs)) which is slightly higher than that of using max-wins voting with SVM. This indicates that the pairwise coupling might be a better choice. A similar conclusion was made in [14]. Note that this method requires a-posteriori probabilities which are only indirectly generated in case of SVM using the method described by [51]. To compare these results with our proposed algorithm we conducted an experiment using the same pairwise coupling idea on a set of binary FS-KLR models, we refer to this as FS-KLR-pc, and a similar amount of PVs (i.e. $89,000$). It was observed that a comparable accuracy of 82.8% was obtained. For the FS-MKLR experiment we tuned hyper-parameters (ν and σ^2) for $M \in \{6000, 7000, 8000\}$. We ceiled M to $8,000$ to keep training/tuning times acceptable. Using the process described in 3.1.3 $M = 8,000$ was selected which gave an accuracy of 81.9%. This is 1.3% worse than that of SVM-pc but only $8,000$ PVs are used compared to $89,563$ SVs resulting in much faster

Table 2: Classification accuracies on the `timit` full test set for different algorithms are shown. The column **acc** gives the percentage of correctly classified phoneme segments. FS-MKLR is the all-at-once learning algorithm as in (1). FS-KLR-pc couples binary FS-MKLR models using the pairwise coupling scheme as described in Section 2.3 as is also done in case of SVM-pc (in this case the a-posteriori probabilities are indirectly estimated by the idea described in [51]). The SVM table entry uses one-versus-one encoding with max-win voting. All kernel methods used a static RBF-kernel. Note that the FS-MKLR model is trained using the reduced memory implementation as defined in 3.4.

alg	**acc(%)**	$\mathbf{t}_{tr}$	$\mathbf{t}_{es}$
SVM-pc (89,563 SVs)	83.2	10h	2h
SVM (89,563 SVs)	82.9	3h	1h
FS-MKLR (8,000 PVs)	81.9	86h	0.5min
FS-KLR-pc (8,000 PVs)	80.5	50min	6min
FS-KLR-pc (89,000 PVs)	82.8	26h	57min
LDA	66.5	4.7min	0.03min
KNN (k=1)	67.8	-	11.5h

evaluations. The FS-MKLR classifier has a 1.4% better classification accuracy compared to FS-KLR-pc using the same PVs ($M = 8,000$). All kernel classifiers outperform a state-of-the-art HMM/GMM classification experiment which obtained a classification accuracy of 78.6% using a 1-gram model. When considering timing results note that the LIBSVM toolbox is implemented in C++ while FS-(M)KLR in MATLAB.

To compare the relationship of the FS-MKLR outputs to Bayes probabilities, we give in Fig. 4 the histogram of FS-MKLR outputs to the fraction of segments correct for each output that is in a given bin range. In fact, each data point in Fig. 4 represents the average probability of being correct over all the input data for the FS-MKLR outcomes in each of the possible activation bins. This fraction (i.e. the probability of being correct) ideally corresponds to the Bayes probability [15]. A perfect Bayes classifier should have a probability equal to the probability of being correct, which corresponds to the diagonal in the figure. From the figure it is seen that although SVM is not trained in a probabilistic framework it still approximates the Bayes probabilities rather well. The extremes fit nicely with the desired probabilities. In the middle range the Bayes probabilities are underestimates for the SVM outputs. In order to compute the probability scores, SVM uses 89,563 SVs. In contrast using the FS-MKLR model only uses 8,000 PVs and still estimates the Bayes probabilities (especially in the low range) quite nicely. In [6] a similar experiment is performed using Multi Layer Perceptrons (MLPs) but instead of segment classification, frame-wise classification was used on the DARPA `resource management` data set. The a-posteriori MLP probabilities did not approximate the Bayes probabilities as well as those presented in our experiment.

5.2.2. Phoneme recognition

Unlike the previous section we will now not rely on a fixed, known segmentation. In case of phoneme classification only correct segments are processed, with phoneme recognition so

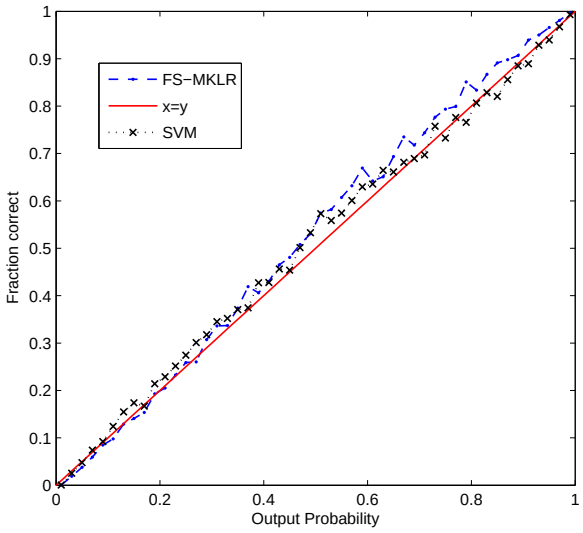


Figure 4: Histogram showing the relationship between FS-MKLR (using 8,000 PVs) and SVM outcome probabilities (using 89,563 SVs) and the percentage correct for each bin. The scores were computed on the independent `timit` full test set.

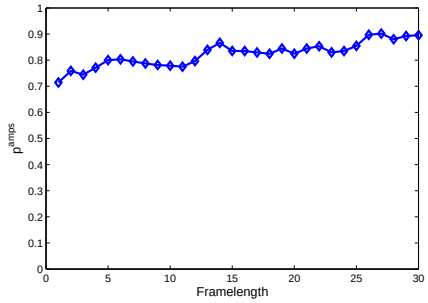
called universum segments are also considered. These are segments that are composed out of frames from different adjacent phonemes when a bad segmentation is employed. In order to explore some properties of the probability outcomes we first define the averaged maximum probability as

$$p^{amps} = \frac{1}{N}\sum_{i=1}^{N} \arg\max_{c}(\tilde{p}_{ci}), \tag{47}$$

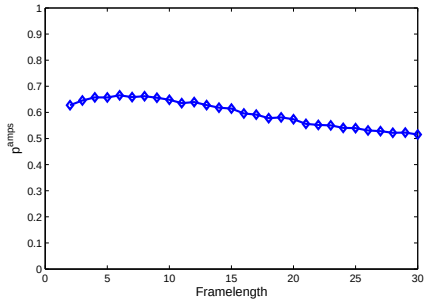
where N is the number of data samples. In Fig. 5(b) we investigate the averaged maximum probability scores for universum segments as a function of the segment length. As was expected, due to training the model only for classification (with given segmentation), in most cases the universum segment is assigned to a certain class with relatively high probability. From the figure we can also remark that for longer frames the model is less confident about the chosen phoneme class. The same trend is seen in Fig. 5(a) where the averaged maximum probabilities for non-universum data is plotted. Here the model is more certain about the class label in case of larger frame lengths. Intuitively, this seems reasonable because more frames are averaged in the preprocessing step in case of larger segments which possibly reduces the effect of noise in the data.

We next take universum information into account to build our phoneme model as was pointed out in Section 2.4. We tuned the hyper-parameter $\nu^{\mathcal{U}}$ on a small development set using Phoneme Error Rate (PER) as a cost criterion. In Table 3 we consider the maximal average phoneme probabilities for all universum segments for the FS-MKLR as well as the FS-MKLR$_{\mathcal{U}}$ model. As was expected it is seen that this value is reduced in case of incorporating universum information. This indicates that universum data is found to be less likely using the FS-MKLR$_{\mathcal{U}}$ compared to the FS-MKLR model.

Our phoneme recognition experiment uses the reference HMM recognizer model as de-



(a) FS-MKLR on non-universum data



(b) FS-MKLR on universum data

Figure 5: The figures show the averaged maximal probability score p^{amps} in function of the considered phoneme segment length for non-universum data from the `timit` full test set in subfigure (5a) and for universum data created using the `timit` full test set in (5b). In both cases $8,000$ PVs were used.

Table 3: The table shows the maximal averaged probability scores for universum data created using the `timit` full test set using the FS-MKLR model from previous section and a tuned FS-MKLR$_{\mathcal{U}}$ using the universum penalization term described in Section 2.4. In both cases $8,000$ PVs were used.

model	p^{amps}
FS-MKLR	0.60
FS-MKLR$_{\mathcal{U}}$	0.27

scribed in (4.2) as segmentation probability model and uses FS-MKLR or SVM to build (context-independent) phoneme models (trained on segmented utterances). The probabilities are combined as is stated in (45).

We proceed as follows

- Given an utterance, a set of possible segmentations, with minimal segment length set to 2 frames and maximal length equal to 30, is created.
- Next, all possible segments are scored by computing (45).
- Obtain the most likely segmentation using a Viterbi decoder.
- Finally, the decoded phoneme sequence was compared to reference transcriptions using Levenshtein's distance to evaluate different sources of errors.

Remark 5. Since the phoneme model favors longer phonemes (see Figure 5a) we added an empirically tuned offset factor (tuned on a small validation set selected from the training database) to the log probabilities to compensate for this effect. We also raised the segment probability factor $\Pr(\underline{S}|\underline{X})$ and 1-gram factor $\Pr(\underline{U})$ in (45) to an empirically tuned power of 0.02 (using the same small validation set).

Remark 6. Note that using the Kai-Fu Lee mapping to 39 phoneme classes implies that there are repeated phonemes in the reference sequence. For instance, consider a phoneme

sequence $(/ey/, /pcl/, /pau/)$ which will be transformed to $(/ey/, /pau/, /pau/)$. Since we choose to train our models (in case of the kernel-based alternatives) on the reduced set of 39 classes (hence there is no model for $/pcl/$), they cannot identify such a segmentation but instead are biased towards recognizing $(/ey/, /pau/)$ which generates a deletion error. We believe that ignoring this fact affects our results negatively in contrast to others which postpone the mapping to just before scoring the sequences. When considering the `timit` full test set this situation occurs in 94.2% of the cases for the silence class $/pau/$. It is therefor that multiple runs of identical phonemes in the decoded as well as the reference sequences were collapsed to single phoneme instances to reflect true insertion errors.

In Table 5 we present our phoneme recognition results obtained on the `timit` full test set. We notice that using the SVM phoneme model we have a PER which is only slightly better but has a much larger evaluation time than that of the FS-MKLR alternative. Without using a language model recognizing the phoneme sequences in the `timit` full test set using the SVM model required 121 against 2.6 hours in case FS-MKLR using $8,000$ PVs (Note that the state-of-the-art HMM needed 41 minutes to process the same data). No results for 1-gram language models are shown in Table 5 since these only had a negative impact on the recognition rate, consistent with the remark made in Section 4.3. Including universum data into the phoneme classification model improves the PER significantly. If no language model is used we have a 2.66% reduction in PER. In this case we have a recognition rate which is comparable to that of our state-of-the-art frame-based HMM using a 1-gram shown in Table 4. However, using N-grams does not result in as much performance improvement as is the case with the HMM architecture. Although the performance is consistently worse, for FS-MKLR and SVM phoneme models the same reduction ratio is found as in case of the HMM architecture. Investigating the cause of this phenomenon in case of using universum data into the FS-MKLR model is subject to future research. In case we neglect repeating phonemes in the reference sequence the scores are consistently 0.4% worse than those presented in Table 5.

Remark 7. Remark that the HMM recognizer in total uses $448,118$ unknowns and the FS-MKLR, without the segmentation probability factor, has $312,039$ unknowns.

6. Conclusion

We presented a practical and scalable implementation for Kernel Logistic Regression which exhibits characteristics such as (i) state-of-the-art non-linear classification performance, (ii) probabilistic interpretations of the model outcomes, (iii) a well-described multi-class approach, (iv) and easily extendable using a wide-range of positive semi-definite kernels (e.g. sequence kernels) that make it a valuable candidate to be used in speech recognition architectures. Experimental evidence gave the following conclusions:

- Adding universum data consistently improved the likelihood scores compared to building a model without including universum information.

Table 4: Phoneme recognition rates using our state-of-the-art reference HMM phoneme recognizer. Codes CI,LM,I,D,S and PER respectively signify context independent, language model, number of insertions, number of deletions, number of substitutions and phoneme error rate.

LM	(I,D,S) (%)	PER (%)
no LM	$(3.04, 9.23, 17.99)$	30.26
1-gram	$(2.97, 8.98, 17.01)$	28.97
2-gram	$(2.70, 7.65, 16.36)$	26.70
3-gram	$(3.17, 6.50, 15.96)$	25.62
4-gram	$(3.41, 6.16, 15.71)$	25.28

Table 5: Phoneme recognition rates using FS-MKLR, FS-MKLR$_{\mathcal{U}}$ and SVM as phoneme probability models were the SVM contains $89,563$ PVs and the FS-MKLR models use $8,000$ PVs. The column labels I,D,S respectively indicate the number of insertions, deletions and substitutions. The abbreviation LM stands for language model.

LM	**FS-MKLR**		**SVM**		**FS-MKLR$_{\mathcal{U}}$**	
	(I,D,S) (%)	PER (%)	(I,D,S) (%)	PER (%)	(I,D,S) (%)	PER (%)
no LM	$(4.69, 10.40, 16.02)$	31.10	$(4.13, 11.92, 15.42)$	31.47	$(4.68, 8.03, 16.10)$	28.81
2-gram	$(4.01, 9.14, 16.17)$	29.31	$(4.16, 9.52, 15.52)$	29.20	$(4.84, 7.04, 16.15)$	28.03
3-gram	$(4.09, 8.50, 15.74)$	28.32	$(4.82, 8.18, 15.14)$	28.13	$(4.36, 7.27, 15.70)$	27.32
4-gram	$(4.59, 7.80, 15.73)$	28.13	$(4.44, 8.60, 14.98)$	28.02	$(4.52, 7.05, 15.61)$	27.17

- In case of phoneme classification, SVM as well as the KLR all-at-once approximation called FS-MKLR outperformed a state-of-the-art HMM classifier.
- While having only a small decrease in terms of classification performance, FS-MKLR produced a model which was much sparser compared to the kernel based alternatives. This resulted in significantly faster evaluation times.
- Compared to FS-KLR-pc, FS-MKLR with the same amount of PVs $(8,000)$ had a 1.4% better classification performance.
- The estimated a-posteriori probabilities of FS-MKLR and SVM seem to match the Bayes probability fairly well.
- In case of phoneme recognition the use of universum data, based on a regularization argument, to build the phoneme model improved the final phoneme recognition rate without increasing the number of unknowns of the phoneme model.
- Without language model our segment-based approach has comparable results to that of the frame-synchronous state-of-the-art HMM recognizer using a 1-gram.

However, when we do include language models our method does not have the same PER reduction as is the case for the HMM recognizer. Investigating this mismatch will be the

subject of future research. We also believe there is room left for further improvement of the recognition rates. Firstly, instead of using a frame-based segmentation model we could use a segment-based model incorporating segmental features or features extracted at specific temporal landmarks (e.g. points of oral closure (or release), points of maximal constriction (or opening) in the vocal tract). Secondly, instead of using static kernels such as the RBF-kernel other types such as sequence kernels might be employed to operate directly on the variable-length segments.

Appendix A. Primal-dual formulation of MKLR

Appendix A.1. Gradient and Hessian

In this appendix a primal-dual formulation of the MKLR algorithm is stated. As will be shown next, MKLR can be solved in the Least Squares Support Vector Machines (LS-SVM) framework. This is achieved by first translating the minimization of objective function (5) to an iteratively re-weighted least-squares (IRWLS) equivalent. Then each locally weighted least-squares problem can be kernelized and phrased as a weighted LS-SVM problem. Hence, one can define an iterative scheme of re-weighted LS-SVMs which minimizes the MKLR objective.

By defining $A \in \mathbb{R}^{CN \times C(D+1)}$ as

$$A = \left(\begin{array}{c|c|c|c} A_1 & A_2 & \ldots & A_N \end{array} \right)^T, \tag{A.1}$$

where $A_i \in \mathbb{R}^{C \times C(D+1)}$ and

$$A_i = \begin{pmatrix} x_i^T & 0 & & 0 \\ 0 & x_i^T & & 0 \\ & & \ddots & \\ 0 & 0 & & x_i^T \end{pmatrix}. \tag{A.2}$$

we can write the formulas of the gradient and Hessian of the objective function (5) in a matrix notation which is convenient to reformulate to an IRWLS problem.

In iteration k this gives a gradient defined as

$$g^{(k)} = A^T(\tilde{p}^{(k)} - \tilde{r}) + \nu w^{(k-1)}, \tag{A.3}$$

where notation $\tilde{\cdot}$ denotes a certain component ordering as is explained in the introduction, and

$$\tilde{r}_{ci} = \mathcal{I}(y_i = c), \tag{A.4}$$

with $\mathcal{I}$ the indicator function. The k-th Hessian is given by

$$H^{(k)} = A^T W^{(k)} A + \nu I, \tag{A.5}$$

where $W^{(k)}$ is a block-diagonal weight matrix

$$W^{(k)} = \text{blockdiag}(W_1(k), ..., W_N(k)), \tag{A.6}$$

with the i-th block of a block-diagonal weight matrix $W^{(k)}$ defined as

$$W_i^{(k)} = \begin{pmatrix} \tilde{p}_{1i}^{(k)}(1 - \tilde{p}_{2i}^{(k)}) & -\tilde{p}_{1i}^{(k)}\tilde{p}_{2i}^{(k)} & \ldots & -\tilde{p}_{1i}^{(k)}\tilde{p}_{Ci}^{(k)} \\ -\tilde{p}_{2i}^{(k)}\tilde{p}_{1i}^{(k)} & \tilde{p}_{2i}^{(k)}(1 - \tilde{p}_{2i}^{(k)}) & & -\tilde{p}_{2i}^{(k)}\tilde{p}_{Ci}^{(k)} \\ & & \ddots & \\ -\tilde{p}_{Ci}^{(k)}\tilde{p}_{1i}^{(k)} & -\tilde{p}_{Ci}^{(k)}\tilde{p}_{2i}^{(k)} & \ldots & \tilde{p}_{Ci}^{(k)}(1 - \tilde{p}_{Ci}^{(k)}) \end{pmatrix}. \tag{A.7}$$

Note that this double indexing of a vector is explained in the introduction.

Remark that for each matrix column j we have $\sum_{i=1}^{N} W_{ij}^{(k)} = 1$ which makes the term $A^T W^{(k)} A$ positive semi-definite. Adding a positive constant ν to the diagonal results in a positive definite Hessian.

Appendix A.2. Iteratively re-weighted least squares

Newton's method computes in each iteration k the optimal step $s^{(k)opt}$ using the Taylor expansion of the objective function ℓ^{ν}_{LR}. This results in the following local optimization step

$$s^{(k)opt} = \arg\min_{s^{(k)}} \ \ell_{LR}(w^{(k-1)}) + (A^T(\tilde{p}^{(k)} - \tilde{r}) + \nu w^{(k-1)})^T s^{(k)} + \frac{1}{2} s^{(k)T}(A^T W^{(k)} A + \nu I) s^{(k)}. \tag{A.8}$$

By exchanging terms we can proof that (A.8) can be expressed as a iteratively re-weighted least squares problem (IRWLS) which can be written as

$$\min_{s^{(k)}} \frac{1}{2} ||As^{(k)} - \tilde{q}^{(k)}||^2_{W^{(k)}} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}), \tag{A.9}$$

where $\tilde{p}^{(k)}, A, W^{(k)}$ are respectively defined as in (5),(A.1),(A.6), and

$$\tilde{q}^{(k)} = (W^{(k)})^{-1}(\tilde{r} - \tilde{p}^{(k)}). \tag{A.10}$$

It turns out that the global regularization term in (5) is reflected in each step as a usual regularization term, resulting in a robust algorithm when ν is chosen appropriately.

Appendix A.3. Iteratively re-weighted least-squares support vector machines

Starting from equation (A.9) we include a feature map and introduce the error variable e, this results in

$$\begin{aligned} \min_{s^{(k)}, e^{(k)}} & \frac{1}{2} e^{(k)T} W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}) \\ & \text{such that } \tilde{q}^{(k)} = F s^{(k)} + e^{(k)}, \end{aligned} \tag{A.11}$$

where $F \in \mathbb{R}^{CN \times C(D_\varphi + 1)}$ is defined as

$$F = \left(\ F_1 \ \middle| \ F_2 \ \middle| \ \ldots \ \middle| \ F_N \ \right)^T, \tag{A.12}$$

where

$$F_i = \begin{pmatrix} \varphi(x_i)^T & 0 & & 0 \\ 0 & \varphi(x_i)^T & & 0 \\ & & \ddots & \\ 0 & 0 & & \varphi(x_i)^T \end{pmatrix}. \tag{A.13}$$

In the context of LS-SVMs this is called the primal problem. To conveniently write down a dual formulation of this problem we use the following notation for the multi-class kernel matrix $G = FF^T$.

The Lagrangian of the constrained problem as stated in (A.11) becomes

$$\mathcal{L}(s^{(k)}, e^{(k)}; \tilde{\alpha}^{(k)}) = \frac{1}{2} e^{(k)T} W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}) - \tilde{\alpha}^{(k)T}(F s^{(k)} + e^{(k)} - \tilde{q}^{(k)})$$

with Lagrange multipliers $\tilde{\alpha}^{(k)} \in \mathbb{R}^{NC}$. The first order conditions for optimality are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial s^{(k)}} = 0 \rightarrow s^{(k)} = \frac{1}{\nu} F^T \tilde{\alpha}^{(k)} - w^{(k-1)} \\ \frac{\partial \mathcal{L}}{\partial e^{(k)}} = 0 \rightarrow \tilde{\alpha}^{(k)} = W^{(k)} e^{(k)} \\ \frac{\partial \mathcal{L}}{\partial \tilde{\alpha}^{(k)}} = 0 \rightarrow F s^{(k)} + e^{(k)} = \tilde{q}^{(k)}. \end{cases} \tag{A.14}$$

This results in the following dual solution at iteration k

$$\left(\frac{1}{\nu} G + (W^{(k)})^{-1} \right) \tilde{\alpha}^{(k)} = \tilde{q}^{(k)} + F w^{(k-1)}. \tag{A.15}$$

The solution $w^{(k')}$ can be expressed in terms of $\tilde{\alpha}^{(k')}$ computed in the last iteration. Given the formula for $s^{(k)}$ in (A.14) and the update rule $w^{(k)} = w^{(k-1)} + s^{(k)}$ employed in the IRWLS scheme, one can write

$$w^{(k')} = \frac{1}{\nu} F^T \tilde{\alpha}^{(k')}. \tag{A.16}$$

The linear system in (A.15) can be solved in each iteration by substituting $w^{(k-1)}$ with $\frac{1}{\nu} F^T \tilde{\alpha}^{(k-1)}$ which then gives

$$\left(\frac{1}{\nu} G + (W^{(k)})^{-1} \right) \tilde{\alpha}^{(k)} = \tilde{q}^{(k)} + \frac{1}{\nu} G \tilde{\alpha}^{(k-1)}. \tag{A.17}$$

Probability $\Pr(\mathbf{Y} = y_* | \mathbf{X} = x^*; w)$ can be predicted by using (1) where $f_c(x^*) = w_{c.}^T \varphi(x^*) = \frac{1}{\nu} \sum_{i=1}^{N} \tilde{\alpha}_{ci} K(x_i, x^*)$.

Acknowledgment

The authors would like to thank colleagues Kris Demuynck for providing the scripts to score the phoneme sequences and Kris De Brabanter for many constructive discussions.

JS is a professor at the Katholieke Universiteit Leuven, Belgium. Research supported by: (i)Research Council KUL: GOA AMBioRICS, GOA MaNet, CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants;(ii) Flemish Government, FWO: PhD/postdoc grants, projects G.0452.04 (new quantum algorithms), G.0211.05 (Nonlinear), G.0226.06 (cooperative systems and optimization), G.0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC); IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; (iii) Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011) ; (iv) EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCOM (ICT-248940); (v) Contract Research: AMINAL; (vi) Other: Helmholtz: viCERP; ACCM; Bauknecht; Hoerbiger. The scientific responsibility is assumed by its authors.

References

[1] M. Aizerman, E. Braverman, L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition," *Automation and Remote Control*, Vol. 25, pp. 821-837, 1964.

[2] J.A. Anderson, V. Blair, "Penalized maximum likelihood estimation in logistic regression and discrimination", *Biometrika*, Vol. 1, pp. 123–136, 1982.
[3] S. Austin, G. Zavaliagkos, J. Makhoul, R. Schwartz, "Speech recognition using segmental neural nets", *In Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, U.S.A., Vol. 1, pp. 625–628, 1992.
[4] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra, "A maximum entropy approach to natural language processing", *Computational Linguistics*, Vol. 22(1), pp. 3971, 1996.
[5] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
[6] H.A. Bourlard, N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994.
[7] S. Boyd, L. Vandenberghe, *Convex optimitization*, Cambridge University Press, 2002. Available online at `http://www.stanford.edu/~boyd/cvxbook.html`.
[8] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, B. De Moor, "Optimized Fixed-Size Least Squares Support Vector Machines for Large Datasets", *Internal Report 08-193 ESAT-SISTA, K.U.Leuven, Belgium*, 2008.
[9] G.C. Cawley, "Leave-One-Out Cross-Validation Based Model Selection Criteria for Weighted LS-SVMs", *In Proc. of the International Joint Conference on Neural Networks (IJCNN-2006)*, Vancouver, Canada, pp. 1661–1668, 2006.
[10] N.V. Chawla, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, "Smote: Synthetic minority oversampling technique", *Journal Of Artificial Intelligence Research*, Vol. 16, pp. 321357, 2002.
[11] P. Clarkson, P.J. Moreno, "On the use of support vector machines for phonetic classification", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, Arizona, U.S.A, Vol. 2 , pp. 585–588, 1999.
[12] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," *In IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, U.S.A., pp. 413-416, 2007.
[13] K. Demuynck, *Extracting, Modelling and Combining Information in Speech Recognition*, Ph.D. thesis, Faculty of Engineering, K.U.Leuven, ESAT-PSI, Belgium, 2001.
[14] K. Duan, S.S Keerthi, "Which Is the Best Multiclass SVM Method? An Empirical Study", *In Proc. of the Sixth International Workshop on Multiple Classifier Systems*, Seaside, Oregon, U.S.A., pp. 278-285, 2005.
[15] R.O. Duda, P.E. Hart, *Pattern Recognition and Scene Analysis*, Wiley, 1973.
[16] P.P.B. Eggermont, V.N. LaRiccia, *Maximum Penalized Likelihood Estimation. Volume II: Regression*, Springer, 2001.
[17] T. Feder, D. Greene, "Optimal algorithms for approximate clustering". *In Proc. 20th ACM Symposium Theory of Computing*, Chicago, Illinois, U.S.A, pp. 434–444, 1988.
[18] A. Ganapathiraju, *Support Vector Machines For Speech Recognition*, PhD thesis, Mississippi State University, 2001.
[19] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, Vol. 4(1), pp. 1–58, 1992.
[20] M. Girolami "Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem", *Neural Computation*, Vol. 14(3), pp. 669–688, 2003.
[21] J.R. Glass, "A probabilistic framework for feature-based speech recognition", In Proc. of ICSLP, Philadelphia, Pennsylvania, U.S.A., pp. 2277-2280, 1996.
[22] J.R. Glass, "A probabilistic framework for segment-based speech recognition", *Computer Speech & Language*, Vol. 17 (2–3), pp. 137–152, 2003.
[23] T. Gonzalez, "Clustering to minimize the maximum intercluster distance", *Theoretical Computer Science*, Vol. 38, pp. 293–306, 1985.
[24] T. Hastie, R. Tibshirani, "Classification by pairwise coupling", *Advances in Neural Information Processing Systems*, Vol. 10, 1998.
[25] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2001.

[26] D.W. Hosmer, S. Lemeshow, *Applied Logistic Regression*, Wiley, 1989.
[27] C.W. Hsu, C.J. Lin, "A comparison of methods for multi-class support vector machines" *IEEE Transactions on Neural Networks*, Vol. 13, pp. 415–425, 2002.
[28] R. Hérault, Y. Grandvalet, "Sparse Probabilistic Classifiers", *In Proc. of the 24-th International Conference on Machine Learning*, Corvallis, Oregon, U.S.A., pp. 337–344, 2007.
[29] J.J. Hull, "A database for handwritten text recognition research", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16(5), pp. 550554, 1994.
[30] T. Jaakkola, D. Haussler, "Probabilistic kernel regression models", *In Seventh International Workshop on Artificial Intelligence and Statistics*", Fort Lauderdale, Florida, U.S.A., 1999.
[31] P. Karsmakers, K. Pelckmans, J. A. K. Suykens, "Multi-class kernel logistic regression: a fixed-size implementation", *In Proc. of the international joint conference in neural networks (IJCNN)*, Orlando, Florida, U.S.A, pp. 1756–1761, 2007.
[32] P. Karsmakers, K. Pelckmans, J. A. K. Suykens, H. Van hamme, "Fixed-Size Kernel Logistic Regression for Phoneme Classification", *In Proc. of INTERSPEECH*, Antwerpen, Belgium, pp. 78–81, 2007.
[33] M. Katz, M. Schaffner, E. Andelic, S. Krüger, A. Wendemuth "Sparse Kernel Logistic Regression for Phoneme Classification", *In Proc. of 10th International Conference on Speech and Computer (SPECOM)*, Patras, Greece, Vol. 2, pp. 523–526, 2005.
[34] S.S. Keerthi, K.B. Duan, S.K. Shevade, A.N. Poo, "A Fast Dual Algorithm for Kernel Logistic Regression", *Machine Learning*, Vol. 61, pp.151–165, 2005.
[35] S.S Keerthi, O. Chapelle, D. Decoste, P. Bennett, E. Parrado-hernndez, "Building Support Vector Machines with Reduced Classifier Complexity", *Journal of Machine Learning Research*, Vol. 7, pp. 1493–1515, 2006.
[36] Y. Kim, S. Kwon, S.H. Song, "Multiclass sparse logistic regression for classification of multiple cancer types using gene expression data", *Computational Statistics & Data Analysis*, Elsevier, 2006.
[37] G. Kimeldorf, G. Wahba, "Some results on Tchebycheffian spline functions", *Journal of Mathematics Analysis and Applications*, Vol. 33, pp. 82–95, 1971.
[38] P. Komarek, A. Moore, "Making Logistic Regression A Core Data Mining Tool With TR-IRLS", *In Proc. of the 5th International Conference on Data Mining*, Houston, Texas, U.S.A., pp. 685–688, 2005.
[39] H.C. Leung, I.L. Hetherington, V.W. Zue, "Speech recognition using stochastic segment neural networks", *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, U.S.A, Vol. 1, pp. 613-616, 1992.
[40] C.C. Chang, C.J. Lin, "LIBSVM : a library for support vector machines", *Software available at* `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.
[41] C.-J. Lin, R.C. Weng, S.S. Keerthi, Trust Region Newton Method for Large-Scale Logistic Regression, *Journal of Machine Learning Research*, Vol. 9, pp. 623–646, 2008.
[42] K.F. Lee and H.W. Hon, "Speaker-independent phone recognition using hidden Markov models", *In Proc. IEEE Transactions on Acoustics, Speech, and Signal Processing*, New York, New York, U.S.A., Vol. 37, no. 11, pp. 1641–1648, 1988.
[43] J.-P Martens, "A connectionist approach to continuous speech recognition", *In Proc. FORWISS/CRIM ESPRIT workshop*", Munich, Germany, pp. 26–33, 1994.
[44] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K.-R. Müller, "Invariant feature extraction and classification in feature spaces", *Advances in Neural Information Processing Systems*, Vol. 12, pp. 526–532, 2000.
[45] P. McCullagh, J.A. Nelder, *Generalized Linear Models, 2nd edition*, Chapman and Hall, London, 1989.
[46] J. A. Nelder, R. Mead, "A simplex method for function minimization", *Computer Journal*, Vol. 7, 308–313, 1965.
[47] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer, 1999.
[48] E. Osuna, R. Freund, F. Girosi, "Support vector machines: Training and applications", *AI Memo 1602*, Massachusetts Institute of Technology, 1997.
[49] G.E. Peterson, "Control methods used in a study of the vowels", *Journal of the Acoustical Society of America*, Vol. 24, pp. 175-184, 1952.

[50] M. Ostendorf, V.V. Digalakis, O. A. Kimball, "From HMMs to segment models: A unified view of stochastic modeling for speech recognition", *IEEE Transactions on Speech and Audio Processing*, Vol. 4(5), pp. 360-378, 1996.
[51] J.C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", *Advances in Large Margin Classifiers*, MIT Press, pp. 61–74, 1999.
[52] J.C. Platt, N. Cristianini, J. Shawe-Taylor, "Large margin DAGs for multiclass classification", *Advances in Neural Information Processing Systems*, Vol. 12, pp. 547-553, 2000.
[53] D. Price, S. Knerr, L. Personnaz, G. Dreyfus, "Pairwise nerual network classifiers with probabilistic outputs", *Neural Information Processing Systems*, Vol. 7, pp. 1109-1116, 1995.
[54] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.
[55] V. Roth, "Probabilistic Discriminative Kernel Classifiers for Multi-class Problems", *In Proc. of the 23rd DAGM-Symposium on Pattern Recognition*, Munich, Germany, pp. 246–253, Springer, 2001.
[56] V. Roth, "The generalized LASSO", *IEEE Transactions on Neural Networks*, Vol. 15, pp. 16-28, 2004.
[57] L. Tóth, A. Kocsor, "Telephone speech recognition via the combination of knowledge sources in a segmental speech model", *Acta Cybernetica*, Vol. 16, pp. 643-657, 2004.
[58] M. Schafföner, S.E. Krüger, E. Andelic, M. Katz, A. Wendemuth, "Limited training data robust speech recognition using kernel-based acoustic models", *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France, Vol. 1, pp. 1137–1140, 2006.
[59] M. Seeger, "Cross-Validation Optimization for Large Scale Structured Classification Kernel Methods", *Journal of Machine Learning Research*, Vol. 9, pp. 1147–1178, 2008.
[60] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
[61] J.A.K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", *Neural Processing Letters*, Vol. 9(3), pp. 293–300, 1999.
[62] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.
[63] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation", *Neurocomputing*, Vol. 48(1–4), pp. 85–105, 2002.
[64] TIMIT Acoustic-Phonetic Continuous Speech Corpus, *National Institute of Standards and Technology Speech Disc 1 -1.1*, NTIS Order No. PB91 -5050651996, 1990.
[65] C.J. Merz, P.M. Murphy, "UCI repository of machine learning databases", *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1998.
[66] V.N. Vapnik, *Statistical learning theory*, Wiley, New York, 1998.
[67] J. Verhasselt, I. Illina, J.-P. Martens, Y. Gong, J.-P. Haton, "Assessing the importance of the segmentation probability in segment-based speech recognition", *Speech Communications*, Vol. 24 (1), pp. 51–72, 1998.
[68] P. Vincent, Y. Bengio, "Kernel Matching Pursuit", *Machine Learning*, Vol. 48, pp. 165–187, 2002.
[69] J. Weston, R. Collobert, F. Sinz, L. Bottou and V. Vapnik, "Inference with the Universum", *In Proc. of International Conference on Machine Learning*, Pittsburgh, Pennsylvania, U.S.A., pp. 1009–1016, 2006.
[70] C.K.I Williams, M. Seeger "Using the Nyström Method to Speed Up Kernel Machines", *Advances in Neural Information Processing Systems*, Vol 13., pp. 682–688, 2001.
[71] T.-F. Wu, C.J. Lin, R.C. Weng, "Probability estimates for multi-class classification by pairwise coupling.", *Journal of Machine Learning Research*, Vol. 5, pp. 975–1005, 2004.
[72] D. Yu, L. Deng, A. Acero, "Using Continuous Features in the Maximum Entropy Model", *Pattern Recognition Letters*, Vol. 30 (8), 2009.
[73] D. Yu, L. Deng, A. Acero, "Hidden Conditional Random Field with Distribution Constraints for Phone Classification", *Interspeech*, Brighton, UK, pp. 676-679, 2009.
[74] G. Zavaliagkos, J. Zhao, R. Schwartz, J. Makhoul, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition" *IEEE Trans. Speech and Audio Proc.*, Vol. 2(1), pp. 151-159, 1994.

[75] K. Zhang, I.W. Tsang, J.T. Kwok, "Improved Nyström low-rank approximation and error analysis", *In Proc. of the 25th International Conference on Machine Learning*, Helsinki, Finland, pp. 1232–1239, 2008.

[76] J. Zhu, T. Hastie, "Kernel logistic regression and the import vector machine", *Advances in Neural Information Processing Systems*, Vol. 14, pp. 1081–1088, 2001.

[77] J. Zhu and T. Hastie, "Classification of gene microarrays by penalized logistic regression", *Biostatistics*, Vol. 5, pp. 427-444, 2004.