Forward Decoding Kernel Machines: A Hybrid HMM/SVM Approach to Sequence Recognition

Shantanu Chakrabartty and Gert Cauwenberghs

Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218, USA
{*shantanu,gert*}*@jhu.edu*

Abstract. Forward Decoding Kernel Machines (FDKM) combine large-margin classifiers with Hidden Markov Models (HMM) for Maximum a Posteriori (MAP) adaptive sequence estimation. State transitions in the sequence are conditioned on observed data using a kernel-based probability model, and forward decoding of the state transition probabilities with the sum-product algorithm directly produces the MAP sequence. The parameters in the probabilistic model are trained using a recursive scheme that maximizes a lower bound on the regularized cross-entropy. The recursion performs an expectation step on the outgoing state of the transition probability model, using the posterior probabilities produced by the previous maximization step. Similar to Expectation-Maximization (EM), the FDKM recursion deals effectively with noisy and partially labeled data.
We also introduce a multi-class support vector machine for sparse conditional probability regression, *Gini*SVM based on a quadratic formulation of entropy. Experiments with benchmark classification data show that *Gini*SVM generalizes better than other multi-class SVM techniques. In conjunction with FDKM, *Gini*SVM produces a sparse kernel expansion of state transition probabilities, with drastically fewer non-zero coefficients than kernel logistic regression. Preliminary evaluation of FDKM with *Gini*SVM on a subset of the TIMIT speech database reveals significant improvements in phoneme recognition accuracy over other SVM and HMM techniques.

1 Introduction

Sequence estimation is at the core of many problems in pattern recognition, most notably speech and language processing. Recognizing dynamic patterns in sequential data requires a set of tools very different from classifiers trained to recognize static patterns in data assumed *i.i.d.* distributed over time.

The speech recognition community has predominantly relied on Hidden Markov Models (HMMs) [1] to produce state-of-the-art results. HMMs are generative models that function by estimating probability densities and therefore require a large amount of data to estimate parameters reliably. If the aim is discrimination between classes, then it might be sufficient to model discrimination boundaries between classes which (in most affine cases) afford fewer parameters.

Recurrent neural networks have been used to extend the dynamic modeling power of HMMs with the discriminant nature of neural networks [2], but learning long term dependencies remains a challenging problem [3]. Typically, neural network training algorithms are prone to local optima, and while they work well in many situations, the quality and consistency of the converged solution cannot be warranted.

Large margin classifiers, like support vector machines, have been the subject of intensive research in the neural network and artificial intelligence communities [4, 5]. They are attractive because they generalize well even with relatively few data points in the training set, and bounds on the generalization error can be directly obtained from the training data. Under general conditions, the training procedure finds a unique solution (decision or regression surface) that provides an out-of-sample performance superior to many techniques.

Recently, support vector machines have been used for phoneme (or phone) recognition [7] and have shown very encouraging results. However, use of a standard SVM classifier implicitly assumes *i.i.d.* data, unlike the sequential nature of phones. Figure 1(b) depicts a typical vowel chart as used extensively in speech and linguistics, showing the location of different vowels with respect to the first two formants (resonant frequencies of the speech articulators). Due to inertia in articulation, speech production results in a smooth transition between vowels, and phones in general [8].

FDKM, introduced in [9], augments the ability of large margin classifiers to perform sequence decoding and to infer the sequential properties of the data. It performs a large margin discrimination based on the trajectory of the data rather than solely on individual data points and hence relaxes the constraint of *i.i.d.* data. FDKMs have shown superior performance for channel equalization in digital communication where the received symbol sequence is contaminated by inter symbol interference [9].

This paper applies FDKM to the recognition of phoneme sequences in speech, and introduces *Gini*SVM, a sparse kernel machine for regression of conditional probabilities as needed for training FDKM over large data. Finally, results of applying FDKM and *Gini*SVM on standard phoneme benchmark data such as TIMIT are included.

2 FDKM formulation

The problem of FDKM recognition is formulated in the framework of MAP (maximum a posteriori) estimation, combining Markovian dynamics with kernel machines. A Markovian model is assumed with symbols belonging to S classes, as illustrated in Figure 1(a) for $S = 3$. Transitions between the classes are modulated in probability by observation (data) vectors $\mathbf{x}$ over time.

2.1 Decoding Formulation

The MAP forward decoder receives the sequence $\overline{\mathbf{X}}[n] = \{\mathbf{x}[n], \mathbf{x}[n-1], \ldots, \mathbf{x}[1]\}$ and produces an estimate of the probability of the state variable $q[n]$ over all

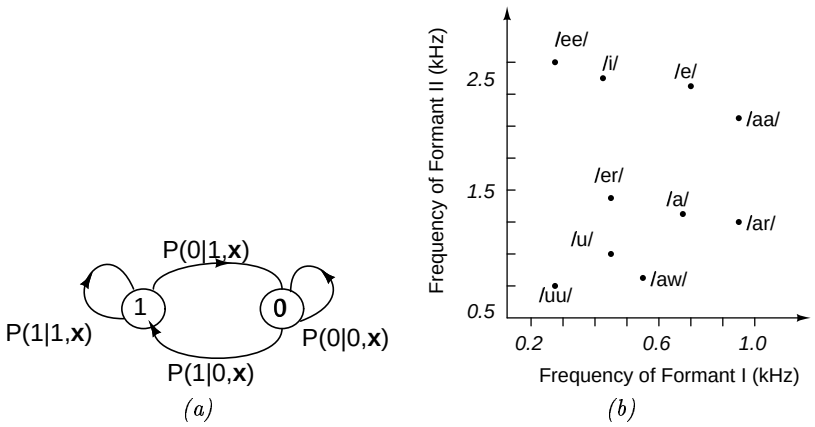

Fig. 1. *(a) Three state markov model, where transition probabilities between states are modulated by the observation vector* $\mathbf{x}$*. (b) Vowel chart showing the location of typical English vowels with respect to the first two formant frequencies. Because of inertial restriction on the articulators, the transition between phones is smooth and the trajectory induces a conditional distribution of a phone with respect to others.*

classes i, $\alpha_i[n] = P(q[n] = i \mid \overline{\mathbf{X}}[n], \mathbf{w})$, where $\mathbf{w}$ denotes the set of parameters for the learning machine. Unlike *hidden* Markov models, the states directly encode the symbols, and the observations $\mathbf{x}$ modulate transition probabilities between states [10]. Estimates of the posterior probability $\alpha_i[n]$ are obtained from estimates of local transition probabilities using the *forward-decoding* procedure [11, 10]

$$\alpha_i[n] = \sum_{j=0}^{S-1} P_{ij}[n]\ \alpha_j[n-1] \tag{1}$$

where $P_{ij}[n] = P(q[n] = i \mid q[n-1] = j, \mathbf{x}[n], \mathbf{w})$ denotes the probability of making a transition from class j at time $n-1$ to class i at time n, given the current observation vector $\mathbf{x}[n]$. The forward decoding (1) embeds sequential dependence of the data wherein the probability estimate at time instant n depends on all the previous data. An on-line estimate of the symbol $q[n]$ is thus obtained:

$$q^{\text{est}}[n] = \arg\max_i \alpha_i[n] \tag{2}$$

The BCJR forward-backward algorithm [11] produces in principle a better estimate that accounts for future context, but requires a backward pass through the data, which is impractical in many applications requiring real time decoding.

Accurate estimation of transition probabilities $P_{ij}[n]$ in (1) is crucial in decoding (2) to provide good performance. In [9] we used kernel logistic regression [12], with regularized maximum cross-entropy, to model conditional probabilities. A different probabilistic model that offers a sparser representation is introduced below.

2.2 Training Formulation

For training the MAP forward decoder, we assume access to a training sequence with labels (class memberships). For instance, the TIMIT speech database comes labeled with phonemes. Continuous (soft) labels could be assigned rather than binary indicator labels, to signify uncertainty in the training data over the classes. Like probabilities, label assignments are normalized: $\sum_{i=0}^{S-1} y_i[n] = 1, y_i[n] \geq 0$.

The objective of training is to maximize the cross-entropy of the estimated probabilities $\alpha_i[n]$ given by (1) with respect to the labels $y_i[n]$ over all classes i and training data n

$$H = \sum_{n=0}^{N-1} \sum_{i=0}^{S-1} y_i[n] \log \alpha_i[n] \tag{3}$$

To provide capacity control we introduce a regularizer $\Omega(\mathbf{w})$ in the objective function [6]. The parameter space $\mathbf{w}$ can be partitioned into disjoint parameter vectors $\mathbf{w}_{ij}$ and b_{ij} for each pair of classes $i, j = 0, \ldots, S-1$ such that $P_{ij}[n]$ depends only on $\mathbf{w}_{ij}$ and b_{ij}. (The parameter b_{ij} corresponds to the bias term in the standard SVM formulation). The regularizer can then be chosen as the L_2 norm of each disjoint parameter vector, and the objective function becomes

$$H = C \sum_{n=0}^{N-1} \sum_{i=0}^{S-1} y_i[n] \log \alpha_i[n] - \frac{1}{2} \sum_{j=0}^{S-1} \sum_{i=0}^{S-1} |\mathbf{w}_{ij}|^2 \tag{4}$$

where the regularization parameter C controls complexity versus generalization as a bias-variance trade-off [6]. The objective function (4) is similar to the primal formulation of a large margin classifier [5]. Unlike the convex (quadratic) cost function of SVMs, the formulation (4) does *not* have a unique solution and direct optimization could lead to poor local optima. However, a *lower bound* of the objective function can be formulated so that maximizing this lower bound reduces to a set of convex optimization sub-problems with an elegant dual formulation in terms of support vectors and kernels. Applying the convex property of the $-\log(.)$ function to the convex sum in the forward estimation (1), we obtain directly

$$H \geq \sum_{j=0}^{S-1} H_j \tag{5}$$

where

$$H_j = \sum_{n=0}^{N-1} C_j[n] \sum_{i=0}^{S-1} y_i[n] \log P_{ij}[n] - \frac{1}{2} \sum_{i=0}^{S-1} |\mathbf{w}_{ij}|^2 \tag{6}$$

with effective regularization sequence

$$C_j[n] = C\alpha_j[n-1] \; . \tag{7}$$

Disregarding the intricate dependence of (7) on the results of (6) which we defer to the following section, the formulation (6) is equivalent to regression of conditional probabilities $P_{ij}[n]$ from labeled data $\mathbf{x}[n]$ and $y_i[n]$, for a given outgoing state j.

2.3 Kernel Logistic Probability Regression

Estimation of conditional probabilities $\Pr(i|\mathbf{x})$ from training data $\mathbf{x}[n]$ and labels $y_i[n]$ can be obtained using a regularized form of kernel logistic regression [12]. For each outgoing state j, one such probabilistic model can be constructed for the incoming state i conditional on $\mathbf{x}[n]$:

$$P_{ij}[n] = \exp(f_{ij}(\mathbf{x}[n]))/\sum_{s=0}^{S-1}\exp(f_{sj}(\mathbf{x}[n])) \tag{8}$$

As with SVMs, dot products in the expression for $f_{ij}(\mathbf{x})$ in (8) convert into kernel expansions over the training data $\mathbf{x}[m]$ by transforming the data to feature space [13]

$$\begin{aligned} f_{ij}(\mathbf{x}) &= \mathbf{w}_{ij}.\mathbf{x} + b_{ij} \\ &= \sum_m \lambda_{ij}^m \ \mathbf{x}[m].\mathbf{x} + b_{ij} \\ &\xrightarrow{\Phi(\cdot)} \sum_m \lambda_{ij}^m \ K(\mathbf{x}[m],\mathbf{x}) + b_{ij} \end{aligned} \tag{9}$$

where $K(\cdot,\cdot)$ denotes any symmetric positive-definite kernel[1] that satisfies the Mercer condition, such as a Gaussian radial basis function or a polynomial spline [6, 14].

Optimization of the lower-bound in (5) requires solving M disjoint but similar sub-optimization problems (6). The subscript j is omitted in the remainder of this section for clarity. The (primal) objective function of kernel logistic regression expresses regularized cross-entropy (6) of the logistic model (8) in the form [14, 15]

$$H = -\sum_i \frac{1}{2}|\mathbf{w}_i|^2 + C\sum_m^N [\sum_i^M y_i[m] f_k(\mathbf{x}[m]) - \log(e^{f_1(\mathbf{x}[m])} + ... + e^{f_M(\mathbf{x}[m])})] \ . \tag{10}$$

The parameters λ_{ij}^m in (9) are determined by minimizing a dual formulation of the objective function (10) obtained through the Legendre transformation, which for logistic regression takes the form of an entropy-based potential function in the parameters [12]

$$H_e = \sum_i^M [\frac{1}{2}\sum_l^N \sum_m^N \lambda_i^l Q_{lm} \lambda_i^m + C\sum_m^N (y_i[m] - \lambda_i^m/C)\log(y_i[m] - \lambda_i^m/C)] \tag{11}$$

subject to constraints

$$\sum_m \lambda_i^m = 0 \tag{12}$$

[1] $K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}).\Phi(\mathbf{y})$. The map $\Phi(\cdot)$ need not be computed explicitly, as it only appears in inner-product form.

$$\sum_i \lambda_i^m = 0 \tag{13}$$

$$\lambda_i^m \leq C y_i[m] \tag{14}$$

Derivations to arrive at the dual formulation are provided in the Appendix.

There are two disadvantages of using the logistic regression dual directly:

1. The solution is non-sparse and all the training points contribute to the final solution. For tasks involving large data sets like phone recognition this turns out to be prohibitive due to memory and run-time constraints.
2. Even though the dual optimization problem is convex, it is not quadratic and precludes the use of standard quadratic programming (QP) techniques. One has to resort to Newton-Raphson or other nonlinear optimization techniques which complicate convergence and require tuning of additional system parameters.

In the next section a new multi-class probabilistic regression technique is introduced which closely approximates the logistic regression solution and yet produces sparse estimates. Like support vector machines, the resulting optimization is quadratic with linear constraints, for which several efficient techniques exist.

3 *Gini*SVM formulation

SVM classification produces sparse solutions but probability estimates they generate are biased. There are techniques that extend SVMs to generate probabilities by cross-validation using held out data [16], but it is hard to extend these to generation of multi-class probabilities.

*Gini*SVM produces a sparse solution by optimizing a dual optimization functional using a lower bound of the dual logistic functional. A quadratic ('Gini' [17]) index is used to replace entropy. The tightness of the bound provides an elegant trade-off between approximation and sparsity.

3.1 Huber Loss and *Gini* Impurity

For binary-class *Gini*SVM, 'margin' is defined as the extent over which data points are asymptotically normally distributed. Using the notation for asymptotic normality in [18], the distribution of distance z from one side of the margin for data of one class[2] is modeled by

$$F(z) = (1-\epsilon)\mathcal{N}(z,\sigma) + \epsilon\mathcal{H}(z) \tag{15}$$

where $\mathcal{N}(.,\sigma)$ represents a normal distribution with zero mean and standard deviation σ, $\mathcal{H}(.)$ is the unknown symmetrical contaminating distribution, and

[2] The distributions for the two classes are assumed symmetrical, with margin on opposite sides, and distance z in opposite directions.

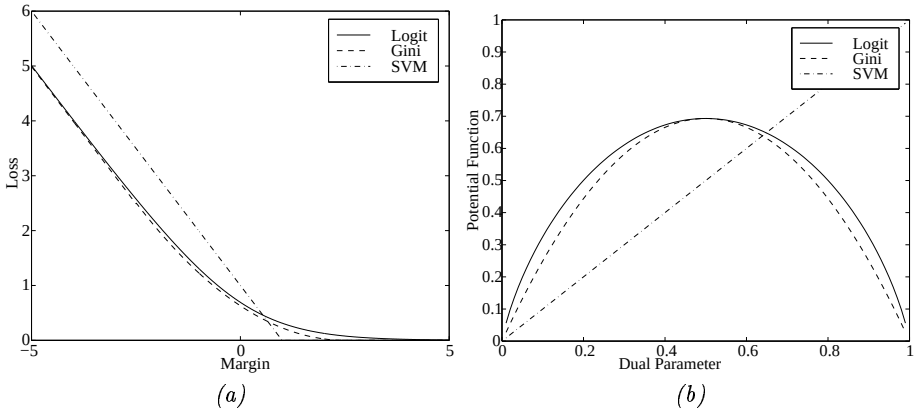

Fig. 2. *Comparison of logistic regression, GiniSVM and soft-margin SVM classification.* (a)*: Loss function in the primal formulation.* (b)*: Potential function in the dual formulation. The GiniSVM loss function and potential function closely approximate those for logistic regression, while offering the sparseness of soft-margin SVM classification.*

$0 \leq \epsilon \leq 1$. $\mathcal{H}(.)$ could, for instance, represent impulsive noise contributing outliers to the distribution.

Huber [18] showed that for this general form of $F(z)$ the most robust estimator achieving minimum asymptotic variance minimizes the following loss function:

$$g(z) = \begin{cases} \frac{1}{2}\frac{z^2}{\sigma^2} & ; |z| \leq k\sigma \\ k\frac{|z|}{\sigma} - \frac{1}{2}\sigma k^2 & ; |z| > k\sigma \end{cases} \tag{16}$$

where in general the parameter k depends on ϵ. For *Gini*SVM, the distribution $F(z)$ for each class is assumed one-sided ($z \leq 0$). As with soft-margin SVM, points that lie beyond the margin ($z > 0$) are assumed correctly classified, and do not enter the loss function ($g(z) \equiv 0$). The loss function asymptotically approaches the logistic loss function for a choice of parameters satisfying $k/\sigma = C$. The correspondence between the Huber and logistic loss functions is illustrated in Figure 2(a).

In the dual formulation, the Huber loss function (16) transforms to a potential function of the form

$$G(\lambda) = k\sigma\frac{\lambda}{C}(1 - \frac{\lambda}{C}) \tag{17}$$

where $C = k/\sigma$. The functional form of the potential (17), for $k\sigma = 2$, corresponds to the *Gini* entropy (or impurity function), used extensively in growing decision trees [17].

The *Gini* impurity potential (17) for *Gini*SVM offers a lower bound to the binary entropy potential for logistic regression. A tight bound is obtained for $k\sigma = 4\log 2$, scaling the *Gini* potential to match the extrema of the logistic potential, shown in Figure 2 (b). Since the *Gini* potential has a finite derivative

at the origin, it allows zero values for the parameters λ_i^m, and thus supports sparsity in the kernel representation, as for soft-margin SVM classification.

The principle of lower bounding the entropy potential function with the *Gini* impurity index, for a sparse kernel representation, can be directly extended to the multi-class case.

3.2 Multi-Class *Gini*SVM

Jensen's inequality ($\log p \leq p-1$) formulates a lower bound for the entropy term in (11) in the form of the multivariate *Gini* impurity:

$$1 - \sum_i^M p_i^2 \leq -\sum_i^M p_i \log p_i \tag{18}$$

where $0 \leq p_i \leq 1, \forall i$ and $\sum_i p_i = 1$. Both forms of entropy $-\sum_i^M p_i \log p_i$ and $1 - \sum_i^M p_i^2$ reach their maxima at the same values $p_i \equiv 1/M$ corresponding to a uniform distribution. As in the binary case, the bound can be tightened by scaling the *Gini* index with a multiplicative factor $\gamma \geq 1$, of which the particular value depends on M.[3] The *Gini*SVM dual cost function H_g is then given by

$$H_g = \sum_i^M [\frac{1}{2} \sum_l^N \sum_m^N \lambda_i^l Q_{lm} \lambda_i^m + \gamma C (\sum_m^N (y_i[m] - \lambda_i^m / C)^2 - 1)] \tag{19}$$

The convex quadratic cost function (19) with constraints in (11) can now be minimized directly using SVM quadratic programming decomposition methods like SMO [19] or by using incremental SVM techniques [20], details of which are discussed in [21]. The primary advantage of the technique is that it yields sparse solutions and yet approximates the logistic regression solution very well.

Figure 3 compares results of training a kernel probability model on three-class data, with the exact solution using logistic regression, and the approximate solution obtained by *Gini*SVM. Deviation between models is observed most clearly at decision regions far removed from training data. Figure 4 demonstrates that by minimizing the *Gini*SVM cost function (using a form of SMO [19, 21]) the true logistic regression cost function tends to decrease with it, although the effect of the approximation is apparent in the fluctuations at convergence.

4 Recursive FDKM Training

The weights (7) in (6) are recursively estimated using an iterative procedure reminiscent of (but different from) expectation maximization. The procedure involves computing new estimates of the sequence $\alpha_j[n-1]$ to train (6) based on estimates of P_{ij} using previous values of the parameters λ_{ij}^m. The training

[3] Unlike the binary case ($M = 2$), the factor γ for general M cannot be chosen to match the two maxima at $p_i = 1/M$.

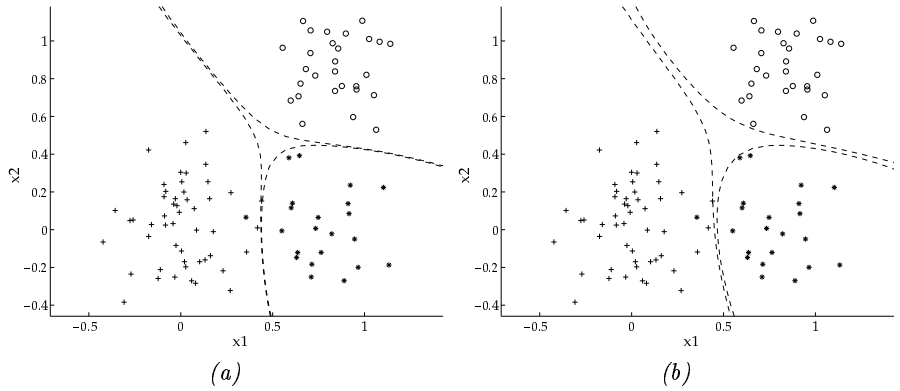

Fig. 3. *Equal (0.5) probability contours obtained from kernel conditional probability regression on a 3-class problem.* (a) *Exact logistic regression, and* (b) *approximate GiniSVM solution.*

proceeds in a series of epochs, each refining the estimate of the sequence $\alpha_j[n-1]$ by increasing the size of the time window (decoding depth, k) over which it is obtained by the forward algorithm (1).

The training steps are illustrated in Figure 5 and summarized as follows:

1. To bootstrap the iteration for the first training epoch ($k = 1$), obtain initial values for $\alpha_j[n-1]$ from the labels of the outgoing state, $\alpha_j[n-1] = y_j[n-1]$. This corresponds to taking the labels $y_i[n-1]$ as true state probabilities which corresponds to the standard procedure of using fragmented data to estimate transition probabilities.
2. Train logistic kernel machines, one for each outgoing class j, to estimate the parameters in $P_{ij}[n]$ $i, j = 1, .., S$ from the training data $\mathbf{x}[n]$ and labels $y_i[n]$, weighted by the sequence $\alpha_j[n-1]$.
3. Re-estimate $\alpha_j[n-1]$ using the forward algorithm (1) over increasing decoding depth k, by initializing $\alpha_j[n-k]$ to $y[n-k]$.
4. Re-train, increment decoding depth k, and re-estimate $\alpha_j[n-1]$, until the final decoding depth is reached ($k = K$).

4.1 Efficient Implementation

The training procedure of Figure 5 entails solving a full optimization problem for each iteration incrementing the decoding depth $k = 1, \ldots K$. This seems poor use of computational resources given that parameter estimates do not vary considerably. The overall training time can be reduced considerably by bootstrapping the optimization problem at each iteration k using the previous parameter values.

Updates to the regularization sequence $C_j[n] = C\alpha_j[n-1]$ at each iteration k affect the feasible region (14) for the optimization, so that the previous solution may be infeasible as a starting point for the next iteration. The solution to this

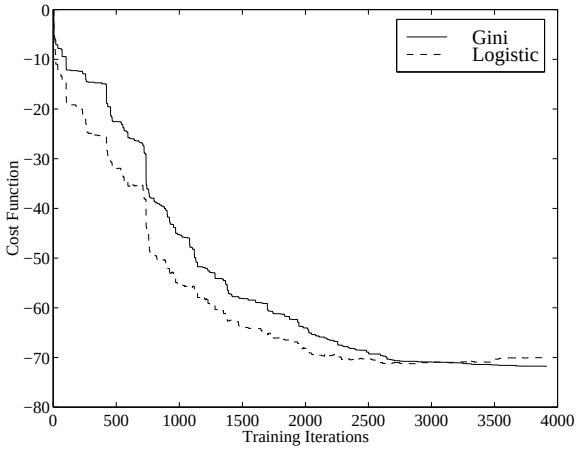

Fig. 4. *Minimization sequence of the GiniSVM dual cost function during SMO training. The logistic dual cost function is concurrently evaluated to illustrate the similarity between the two models.*

problem lies in the invariance of the probability model (8) to a common additive offset λ_*^m in the parameters λ_j^m. To bootstrap a feasible solution, the additive constants λ_*^m are chosen to project the previous solution to the current feasible space as defined by the new regularization sequence $C_j[n]$.

The performance of FDKM training depends on the final decoding depth K, although observed variations in generalization performance for large values of K are relatively small. A suitable value can be chosen *a priori* to match the extent of temporal dependency in the data. For phoneme classification in speech, for instance, the decoding depth can be chosen according to the length of a typical syllable.

5 Experiments and Results

5.1 *Gini*SVM experiments

To test how well *Gini*SVM performs as a multi-class classifier, we evaluated its performance on two standard speech datasets, the Paterson and Barney formant dataset and the Deterding dataset. Results are summarized in Table 1. In both cases, *Gini*SVMs yielded better or comparable out-of-sample classification performance compared to standard "one vs. one," "one vs. all" and other [22] multi-class SVM approaches.

5.2 Experiments with TIMIT data

FDKM performance was evaluated on the larger TIMIT dataset consisting of continuous spoken utterances, preserving sequential structure present between

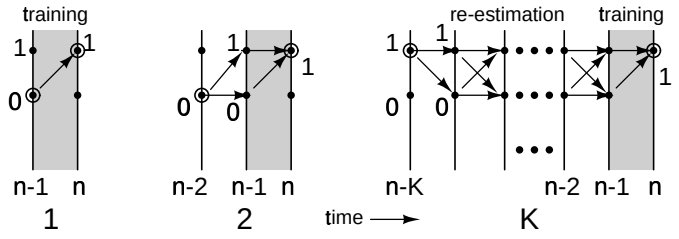

Fig. 5. *Iterations involved in training FDKM on a trellis based on the Markov model of Figure 1. During the initial epoch, parameters of the probabilistic model, conditioned on the observed label for the outgoing state at time $n-1$, of the state at time n are trained from observed labels at time n. During subsequent epochs, probability estimates of the outgoing state at time $n-1$ over increasing forward decoding depth $k = 1, \ldots K$ determine weights assigned to data n for training each of the probabilistic models conditioned on the outgoing state.*

Table 1. *Vowel Classification Results*

Machine	Dataset	Accuracy
SVM (one vs. all)	Paterson-Barney	79.1%
SVM (one vs. one)	Paterson-Barney	83.6%
*Gini*SVM	Paterson-Barney	86.5%
SVM (one vs. all)	Deterding	68.2%
SVM (one vs. one)	Deterding	68.4%
*Gini*SVM	Deterding	70.6%

labels (phones). The TIMIT speech dataset [23] consists of approximately 60 phone classes, which were first collapsed onto 39 phone classes according to standard folding techniques [24]. Training was performed on a subset of "sx" training sentences in the TIMIT corpus. The choice of "sx" sentences was motivated by their phonetic balance, i.e. each of the phone classes carries approximately the same frequency. For training we randomly selected 960 "sx" sentences spoken by 120 speakers, and for test set we randomly chose 160 "sx" sentences spoken by 20 speakers. The relative small size of the subset of the database was primarily governed by current limitations imposed by our SVM training software.

The speech signal was first processed by a pre-emphasis filter with transfer function $1 - 0.97z^{-1}$, and then a 25 ms Hamming window was applied over 10 ms shifts to extract a sequence of phonetic segments. Cepstral coefficients were extracted from the sequence, combined with their first and second order time differences into a 39-dimensional vector. Cepstral mean subtraction and speaker normalization were subsequently applied. Right and left context were added by concatenating previous and next segments to obtain a 117-dimensional feature vector [25]. The feature vectors were averaged over the duration of the

phone to obtain a single 117 dimensional feature vector for each phone utterance resulting in approximately 32,000 data points.

Evaluation on the test was performed using MAP forward decoding (2) and thresholding [26]. The decoded phone sequence was then compared with the transcribed sequence using Levenshtein's distance to evaluate different sources of errors shown in table 2. Multiple runs of identical phones in the decoded and transcribed sequences were collapsed to single phone instances to reflect true insertion errors.

Table 2 summarizes the results of the experiments performed with TIMIT. Note that training was performed over a small subset of the available data, and the numbers are listed for relative comparison purposes only. To calibrate the comparison, benchmark results from a standard Gaussian mixture triphone HMM model are included.

Table 2. *TIMIT Evaluation*

Machine	Accuracy	Insertion	Substitution	Deletion	Errors
HMM (6 mixture triphone)	60.6%	8.4%	30.1%	9.3%	47.8%
SVM (one vs all)	68.4%	4.9%	22.4%	9.2%	36.5%
SVM (one vs one)	69.1%	4.9%	21.6%	9.3%	35.9%
*Gini*SVM	70.3%	4.4%	21.6%	8.1%	34.1%
FDKM k=1, threshold = 0.25	71.8%	4.3%	21.3%	6.9%	32.5%
FDKM k=1, threshold = 0.1	71.4%	5.1%	21.5%	7.1%	33.7%
FDKM k=10, threshold = 0.25	73.2%	4.9%	19.6%	7.2%	31.7%
FDKM k=10, threshold = 0.1	72.9%	5.4%	20.1%	7.0%	32.5%

6 Conclusion

This paper addressed two problems in machine learning, and offered two solutions in conjunction: *Gini*SVM, and FDKM. *Gini*SVM was introduced as a sparse technique to estimate multi-class conditional probabilities with a large-margin kernel machine. Preliminary evaluations of *Gini*SVM show that it outperforms other multi-class SVM techniques for classification tasks. FDKM merges large-margin classification techniques into an HMM framework for robust forward decoding MAP sequence estimation. FDKM improves decoding and generalization performance for data with embedded sequential structure, providing an elegant tradeoff between learning temporal versus spatial dependencies. The recursive estimation procedure reduces or masks the effect of noisy or missing labels $y_j[n]$. Other advantages include a feed-forward decoding architecture that is very amenable to real-time implementation in parallel hardware [27].

References

1. L. Rabiner and B-H Juang, *Fundamentals of Speech Recognition,* Englewood Cliffs, NJ: Prentice-Hall, 1993.
2. Robinson, A.J., "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks*, vol. 5,No.2,March 1994.
3. Bengio, Y., "Learning long-term dependencies with gradient descent is difficult," *IEEE T. Neural Networks,* vol. 5, pp. 157-166, 1994.
4. Boser, B., Guyon, I. and Vapnik, V., "A training algorithm for optimal margin classifier," in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp 144-52, 1992.
5. Vapnik, V. *The Nature of Statistical Learning Theory,* New York: Springer-Verlag, 1995.
6. Girosi, F., Jones, M. and Poggio, T. "Regularization Theory and Neural Networks Architectures," *Neural Computation*, vol. **7**, pp 219-269, 1995.
7. Clark, P. and Moreno, M.J. "On the use of Support Vector Machines for Phonetic Classification," IEEE Conf. Proc., 1999.
8. Laderfoged, P. *A Course in Phonetics*, New York, Harcourt Brace Jovanovich, 2nd ed., 1982.
9. Chakrabartty, S. and Cauwenberghs, G. "Sequence Estimation and Channel Equalization using Forward Decoding Kernel Machines," *IEEE Int. Conf. Acoustics and Signal Proc. (ICASSP'2002)*, Orlando FL, 2002.
10. Bourlard, H. and Morgan, N., *Connectionist Speech Recognition: A Hybrid Approach,* Kluwer Academic, 1994.
11. Bahl, L.R., Cocke J., Jelinek F. and Raviv J. "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Inform. Theory*, vol. **IT-20**, pp. 284-287, 1974.
12. Jaakkola, T. and Haussler, D. "Probabilistic kernel regression models," *Proceedings of Seventh International Workshop on Artificial Intelligence and Statistics* , 1999.
13. Schölkopf, B., Burges, C. and Smola, A., Eds., *Advances in Kernel Methods-Support Vector Learning,* MIT Press, Cambridge, 1998.
14. Wahba, G. *Support Vector Machine, Reproducing Kernel Hilbert Spaces and Randomized GACV*, Technical Report 984, Department of Statistics, University of Wisconsin, Madison WI.
15. Zhu, J and Hastie, T., "Kernel Logistic Regression and Import Vector Machine," *Adv. IEEE Neural Information Processing Systems (NIPS'2001)*, Cambridge, MA: MIT Press, 2002.
16. Platt, J., "Probabilities for SV Machines," *Adv. Large Margin Classifiers,* Smola, Bartlett et al., Eds., Cambridge MA: MIT Press, 1999.
17. Breiman, L. Friedman, J. H. et al. *Classification and Regression Trees*, Wadsworth and Brooks, Pacific Grove, CA, 1984.
18. Huber, P.J., "Robust Estimation of Location Parameter," *Annals of Mathematical Statistics*, vol. 35, March 1964.
19. Platt, J. "Fast Training of Support Vector Machine using Sequential Minimal Optimization," *Adv. Kernel Methods,* Scholkopf, Burges et al., Eds., Cambridge MA: MIT Press, 1999.
20. Cauwenberghs, G. and Poggio, T., "Incremental and Decremental Support Vector Machine Learning," *Adv. IEEE Neural Information Processing Systems (NIPS'2000)*, Cambridge MA: MIT Press, 2001.

21. Chakrabartty, S. and Cauwenberghs, G. "Sequential Minimal Optimization for Kernel Probabilistic Regression," Research Note, Center for Language and Speech Processing, The Johns Hopkins University, MD, 2002.
22. Weston, J. and Watkins, C., "Multi-Class Support Vector Machines," Technical Report CSD-TR-9800-04, Department of Computer Science, Royal Holloway, University of London, May 1998.
23. Fisher, W., Doddington G. et al *The DARPA Speech Recognition Research Database: Specifications and Status.* Proceedings DARPA speech recognition workshop, pp. 93-99, 1986.
24. Lee, K.F. and Hon, H.W, "Speaker-Independent phone recognition using hidden markov models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 1641-1648, 1989.
25. Fosler-Lussier, E. Greenberg, S. Morgan, N., "Incorporating contextual phonetics into automatic speech recognition," *Proc. XIVth Int. Cong. Phon. Sci.*, 1999.
26. Wald, A. *Sequential Analysis*, Wiley, New York, 1947.
27. Chakrabartty, S., Singh, G. and Cauwenberghs, G. "Hybrid Support vector Machine/Hidden Markov Model Approach for Continuous Speech recognition," *Proc. IEEE Midwest Symp. Circuits and Systems (MWSCAS'2000)*, Lansing, MI, Aug. 2000.

Appendix: Dual Formulation of Kernel Logistic Regression

The regularized log-likelihood/cross entropy for kernel logistic regression is given by [14, 15]

$$H = \sum_k \frac{1}{2}|\mathbf{w}_k|^2 - C\sum_n^N[\sum_k^M y_k[n]f_k(\mathbf{x}[n]) - \log(e^{f_1(\mathbf{x}[n])} + ... + e^{f_M(\mathbf{x}[n])}] \ . \quad (20)$$

First order conditions with respect to parameters $\mathbf{w}_k$ and b_k in $f_k(\mathbf{x}) = \mathbf{w}_k.\mathbf{x}+b_k$ yield

$$\begin{aligned} \mathbf{w}_k &= C\sum_n^N[y_k[n] - \frac{e^{f_k(\mathbf{x}[n])}}{\sum_p^M e^{f_p(\mathbf{x}[n])}}]\mathbf{x}[n] \\ 0 &= C\sum_n^N[y_k[n] - \frac{e^{f_k(\mathbf{x}[n])}}{\sum_p^M e^{f_p(\mathbf{x}[n])}}] \ . \end{aligned} \quad (21)$$

Denote

$$\lambda_k^n = C[y_k[n] - \frac{e^{f_k(\mathbf{x}[n])}}{\sum_p^M e^{f_p(\mathbf{x}[n])}}] \quad (22)$$

in the first-order conditions (21) to arrive at the kernel expansion (9) with linear constraint

$$f_k(\mathbf{x}) = \sum_n \lambda_k^n K(\mathbf{x}[n], \mathbf{x}) + b_k \quad (23)$$

$$0 = \sum_n \lambda_k^n \ . \quad (24)$$

Note also that $\sum_k \lambda_k^n = 0$.

Legendre transformation of the primal objective function (20) in $\mathbf{w}_k$ and b_k leads to a dual formulation directly in terms of the coefficients λ_k^n [12]. Define $z_n = \log(\sum_p^M e^{f_p(\mathbf{x}[n])})$, and $Q_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Then (22) and (23) transform to

$$\sum_l Q_{nl}\lambda_k^l - \log[y_k[n] - \lambda_k^n/C] + b_k - z_n = 0 \tag{25}$$

which correspond to first-order conditions of the convex dual functional

$$H_d = \sum_k^M [\frac{1}{2}\sum_n^N \sum_l^N \lambda_k^n Q_{nl}\lambda_k^l + C\sum_n^N (y_k[n] - \lambda_k^n/C)\log(y_k[n] - \lambda_k^n/C)] \tag{26}$$

under constraints

$$\sum_n \lambda_k^n = 0 \tag{27}$$

$$\sum_k \lambda_k^n = 0 \tag{28}$$

$$\lambda_k^n \leq C y_k[n] \tag{29}$$

where b_k and z_n serve as Lagrange parameters for the equality constraints (27) and (28).