Sparse Probability Regression by Label Partitioning

Shantanu Chakrabartty[1], Gert Cauwenberghs[1], and Jayadeva[2]

[1] Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218, USA
{shantanu,gert}@jhu.edu
[2] Department of Electrical Engineering,
Indian Institute of Technology,
Hauz Khas, New Delhi - 110058, INDIA
jayadeva@ee.iitd.ernet.in

Abstract. A large-margin learning machine for sparse probability regression is presented. Unlike support vector machines and other forms of kernel machines, nonlinear features are obtained by transforming labels into higher-dimensional label space rather than transforming data vectors into feature space. Linear multi-class logistic regression with partitioned classes of labels yields a nonlinear classifier in the original labels. With a linear kernel in data space, storage and run-time requirements are reduced from the number of support vectors to the number of partitioned labels. Using the partitioning property of KL-divergence in label space, an iterative alignment procedure produces sparse training coefficients. Experiments show that label partitioning is effective in modeling nonlinear decision boundaries with same, and in some cases superior, generalization performance to Support Vector Machines with significantly reduced memory and run-time requirements.

1 Introduction

Support Vector Machines [1, 2] derive their approximation power by mapping the training data vectors into a higher dimensional feature space, where a linear maximal margin separating hyper-plane can be found to efficiently discriminate between respective classes. Computation in this higher dimensional feature space is made feasible with the aid of reproducing Hilbert space kernels, which are equivalent to computing inner-products in these higher dimensions. Even though SVMs have demonstrated state-of-art classification performance, some challenges remain:

1. The efficiency of real-time classification for SVMs depends on the degree of sparsity as measured by the number of support vectors required for a given task. In most practical scenarios there exists significant overlap between class distributions which results in large numbers of support vectors for classification. For a given Bayes error rate, the number of error support vectors scales directly with the number of training points, and significantly outgrows the number of margin support vectors for very large datasets.

2. The complexity of SVM training scales with the square of the number of support vectors, and becomes prohibitively slow for huge datasets with significant class overlaps.
3. The choice of kernel is heuristic, and is governed by prior knowledge about the problem which usually is not available.

The first two problems can be attributed to the need to represent error vectors as support vectors in the dual formulation, assuming a non-linear kernel. It has been shown [3] that the same classification boundary can be obtained by optimizing a primal reformulation of the SVM cost function, leading to a reduction of the number of effective support vectors, although sparsity is not guaranteed and depends on the kernel used. Reduced Set Methods [4] project the decision surface onto a sparser kernel representation formed by a number of basis functions that are chosen based on an unconstrained non-convex optimization procedure. Relevance Vector Machines (RVM) [5] offer an alternative to obtaining sparse kernel expansions in a Bayesian setting, and have demonstrated same or better generalization performance with fewer 'relevance' vectors, although practical implementation has been limited to relatively small datasets.

We introduce the *Partitioned Label Machine* (PLM) to improve on sparsity of representation and generalization performance in large-margin kernel classification and probability regression. PLMs map the labels, rather than the data vectors, into a higher dimensional space. Through partitioning of the labels, the resulting partitioned classes can be linearly classified. Linear multi-class probability regression can then be used to map decision boundaries and combine hypotheses to form nonlinear classification decisions. The linear form gives rise to a sparse representation in the primal formulation, with an expansion that scales not with the number of training data, but with the number of label partitions. We show that label partitioning implies a nonlinear map similar to that implied in data space by a Mercer kernel in the SVM dual formulation. The PLM method is very general and easily extends from two-class to multi-class problems.

Committee machines [6], voting machines and mixture of experts [7–9] are based on similar lines of combining simple decision surfaces by voting/mixing. Linear weighting of simple hypotheses cannot result in a more complex hypothesis, and we show how PLMs model nonlinear decision surfaces even though the underlying classification functions are linear. SVMs are not combined by linear mixing; rather, partitioned subclasses combine nonlinearly through the competitive and self-normalizing functional form of multinomial logistic regression, applied once to all partitioned classes. Since the classification machine outputs class probabilities, partitioning of the labels can be accomplished in an iterative scheme similar to expectation maximization.

Section 2 describes the probability model used for PLM and provides its justification based on generalization performance. Section 3 formulates the problem in terms of KL-divergence and logistic kernel regression, and compares properties of label partitioning with those of Mercer kernels. Section 4 summarizes the experiments performed on PLMs and section 5 provides concluding remarks.

2 Model Selection

Partitioned Label Machines (PLMs) map an input feature vector $\mathbf{x} \in R^d$ onto one of the respective S label classes based on a compound selection criterion derived from partitioning of the label classes. The class decisions are based on choosing the class $i \in 1..S$ with the highest probability measure $P_i(\mathbf{x})$:

$$P_i(\mathbf{x}) = \Pr(i|\mathbf{x}) = \sum_j^K P_{ij}(\mathbf{x}) \tag{1}$$

obtained by pooling probability mass $P_{ij}(\mathbf{x})$ over K corresponding partitioned sub-classes j:

$$P_{ij}(\mathbf{x}) = \Pr(i, j|\mathbf{x}) = \frac{\exp(\mathbf{w}_{ij}.\mathbf{x} + b_{ij})}{\sum_p^S \sum_q^K \exp(\mathbf{w}_{pq}.\mathbf{x} + b_{pq})}. \tag{2}$$

A special case of interest is that of binary classification with decision function

$$y = \theta(P_1(\mathbf{x}) - 0.5) = \theta(\sum_j^K P_{1j}(\mathbf{x}) - 0.5) \tag{3}$$

where $\theta(\cdot)$ is the Heaviside function mapping onto two respective classes $\{1,0\}$ and the function $P_{1j}(\mathbf{x}) \in \mathcal{H}_s$ is given by

$$P_{1j}(\mathbf{x}) = \frac{\exp(\mathbf{w}_{1i}.\mathbf{x} + b_{1i})}{\sum_q \exp(\mathbf{w}_{0q}.\mathbf{x} + b_{0q}) + \sum_q \exp(\mathbf{w}_{1q}.\mathbf{x} + b_{1q})}. \tag{4}$$

The $\mathcal{P}$-dimension [10] $dim_{\mathcal{P}}$ of the class of functions $\mathcal{H}_s$ is bounded above by $d+1$. This can be easily verified by observing that $P_{1j}(\mathbf{x})$ forms a soft-max of K linear decision surfaces. Using results directly from [11] the scale sensitive dimension $fat_{\mathcal{H}}$ of the class of $\mathcal{H}$ of functions $P_1(\mathbf{x})$ in (3) is given by

$$fat_{\mathcal{H}}(\beta) \leq \frac{cK^2 d}{\beta^2} \log(1/\beta) \tag{5}$$

for some universal constant c and margin β. This shows that the upper-bound on complexity of decision space H is polynomial in the number of sub-classes K, suggesting poor generalization for large values of K. However, appropriately maximizing the margin β allows to control the complexity of the hypothesis class and hence the generalization ability. Equation (5) also suggests to adjust the margin β with with the number of partitions K as to maintain a fixed upper-bound on the complexity of $\mathcal{H}$.

3 Label Partitioning and Re-Estimation

For training the learning machine, we assume access to a training sequence $\mathbf{x}[n] \in R^d$ with labels (class memberships) $y[n] \in R^S$, where n denotes the data index, d the dimension of the input vectors, and S the number of classes. 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: $\sum_{i=1}^{S} y_i[n] = 1, y_i[n] \geq 0$ where S is the total number of classes.

Training could be formulated as minimization of the regularized empirical KL-divergence between the probabilities estimated by the learning machine $P_i[n] = \Pr(i|\mathbf{x}[n])$, and the label vectors $y_i[n]$:

$$H(\mathbf{W}) = \frac{1}{2}\sum_{i}^{S}\sum_{j}^{K}|\mathbf{w}_{ij}|^2 + C\sum_{n=1}^{N}\sum_{i=1}^{S} y_i[n]\log\frac{y_i[n]}{P_i[n]} \tag{6}$$

where $C \geq 0$ is a regularization constant, and $\mathbf{W} = \bigcup_{ij}(\mathbf{w}_{ij}, b_{ij})$ denotes the set of hyperplane parameters of the model.

The cost function (6) is non-convex for the probability model given by (2). To arrive at a convex optimization problem, we derive an auxiliary function upper-bounding the decrease in cost function $\delta H(\mathbf{W}, \mathbf{W}^*) = H(\mathbf{W}^*) - H(\mathbf{W})$, where $\mathbf{W}$ and $\mathbf{W}^*$ denote the current and the previous estimates of parameters. $\delta H(\mathbf{W}, \mathbf{W}^*)$ can be written as

$$\delta H(\mathbf{W}, \mathbf{W}^*) = \delta\Omega(\mathbf{W}, \mathbf{W}^*) + C\delta F(\mathbf{W}, \mathbf{W}^*) \tag{7}$$

with

$$\delta\Omega(\mathbf{W}, \mathbf{W}^*) = \frac{1}{2}\sum_{i}^{S}\sum_{j}^{K}[|\mathbf{w}_{ij}^*|^2 - |\mathbf{w}_{ij}|^2] \tag{8}$$

and

$$\delta F(\mathbf{W}, \mathbf{W}^*) = -\sum_{n}^{N}\sum_{i}^{S}\frac{y_i[n]\sum_j P_{ij}^*[n]}{\sum_j P_{ij}^*[n]}\log\frac{\sum_j P_{ij}^*[n]}{\sum_j P_{ij}[n]} \tag{9}$$

where

$$P_{ij}^*[n] = \frac{\exp(\mathbf{w}_{ij}^*.\mathbf{x}[n] + b_{ij}^*)}{\sum_s^S\sum_k^K \exp(\mathbf{w}_{sk}^*.\mathbf{x}[n] + b_{sk}^*)} \tag{10}$$

$$P_{ij}[n] = \frac{\exp(\mathbf{w}_{ij}.\mathbf{x}[n] + b_{ij})}{\sum_s^S\sum_k^K \exp(\mathbf{w}_{sk}.\mathbf{x}[n] + b_{sk})}. \tag{11}$$

We use the celebrated log-sum inequality to bound (9):

Lemma: For a sequence of non-negative numbers $\{a_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$, and for $a = \sum_{i=1}^n a_i$ and $b = \sum_{i=1}^n b_i$, the following inequality holds

$$\sum_{i}^{n} a_i\log\frac{a_i}{b_i} \geq a\log\frac{a}{b} \tag{12}$$

with equality iff $a_i/b_i = a/b$ for all i. The proof of lemma (12) is simple and is obtained by application of Jensen's inequality to the convex function $t\log(t)$.

Applying (12) to (9) the following expectation-maximization (EM) auxiliary function is obtained

$$A(\mathbf{W}, \mathbf{W}^*) = \frac{1}{2} \sum_{i,j} |\mathbf{w}_{ij}|^2 + C \sum_{n}^{N} \sum_{i,j} y^*_{ij}[n] \log \frac{P^*_{ij}[n]}{P_{ij}[n]} \tag{13}$$

where

$$y^*_{ij}[n] = y_i[n] \frac{P^*_{ij}[n]}{P^*_i[n]} \tag{14}$$

with $P^*_i[n] = \sum_l P^*_{il}[n]$. It is easy to verify that $A(\mathbf{W}, \mathbf{W}^*) \geq 0$. Minimizing the auxiliary function (13) is therefore equivalent to decreasing an upper bound on the cost function (6), and by subsequent iteration of computing the new labels based on previous estimates the procedure converges at least to a local minimum. The upper-bound $A(\mathbf{W}, \mathbf{W}^*)$ bears resemblance to conditional EM bounds in [12] within a regularization framework. One can directly see from the condition for equality in the above log-sum inequality that the solution converges to the global minimum if the following condition is satisfied:

$$\frac{P_{ij}[n]}{P_i[n]} = \frac{y_{ij}[n]}{y_i[n]}. \tag{15}$$

This condition, unfortunately cannot be strictly ensured by the probability measures $P_i(\mathbf{x})$ and therefore has to be satisfied to close approximation. The initialization sub-labels y_{ij} should therefore be chosen such that they can be easily classified using a large margin classifier which can be obtained by using disjoint partitioning or clustering methods like Gaussian mixture modeling.

The principle of mapping the a low-dimension label vector into higher dimensional space yields advantages similar to the 'kernel trick', in this case however one can work in the higher dimensional space directly using a linear kernel. Partitioning the classes into sub-classes yields easily separable classes which can then efficiently combined using multi-class probabilistic regression techniques, reviewed next.

3.1 Logistic Probability Regression

Optimization of the lower-bound in (13) amounts to regressing probabilities over $S \times K$ (partitioned) classes. Estimation of probabilities $P_{ij}[n]$, for partitions j of class i, from training data $\mathbf{x}[n]$ and partitioned labels $y^*_{ij}[n]$, is obtained using a regularized form of logistic probability regression. The model (2) assumes a multinomial logistic form

$$P_{ij}[n] = \exp(f_{ij}(\mathbf{x}[n])) / \sum_{s}^{S} \sum_{k}^{K} \exp(f_{sk}(\mathbf{x}[n])). \tag{16}$$

Primal Formulation:

In the primal formulation, the discriminant functions $f_{ij}(\mathbf{x})$ are expressed in the primal variables defining the coordinates of the hyperplane in feature space. In particular, for a *linear* kernel,

$$f_{ij}(\mathbf{x}) = \mathbf{w}_{ij}.\mathbf{x} + b_{ij}. \tag{17}$$

The objective function of logistic regression expresses regularized divergence (13) of the logistic model (16) in the form [13, 14]

$$H_1 = \sum_{i,j} \frac{1}{2} |\mathbf{w}_{ij}|^2 + C \sum_{n}^{N} [\sum_{i,j} y^*_{ij}[n] f_{ij}(\mathbf{x}[n]) + \log(\sum_{s,k} e^{f_{sk}(\mathbf{x}[n])})] \ . \tag{18}$$

Use of a linear kernel enables use of primal gradient related methods that directly optimizes (13). The advantage of the primal formulation with linear kernel is that the number of variables is fixed by the vector dimension d, and not by the number of training vectors N. In PLM, this implies that the number of terms in the expansion is proportional to the number of partitions, determined by the complexity of the task rather than the size of the data. Partitioning of the labels allows to use a linear kernel and yet model a nonlinear decision surface. Still, it may be advantageous to use a nonlinear Mercer kernel, or to resort to the dual representation otherwise and gain in terms of computational efficiency.

Dual Formulation:

Dual formulation of (18) yields a regularized kernel-based form of logistic regression [15, 14]. As with SVMs, dot products in the expression for $f_{ij}(\mathbf{x})$ in (16) convert into kernel expansions over the training data $\mathbf{x}[m]$ by transforming the data to feature space [4]

$$f_{ij}(\mathbf{x}) = \sum_{m} \lambda^m_{ij} \ K(\mathbf{x}[m], \mathbf{x}) + b_{ij}. \tag{19}$$

The parameters λ^m_{ij} in (19) are determined by minimizing a dual formulation of the objective function (18) obtained through the Legendre transformation, which for logistic regression takes the form of an entropy-based objective function [15]

$$H_2 = \sum_{i,j} [\frac{1}{2} \sum_{l}^{N} \sum_{m}^{N} \lambda^l_{ij} Q_{lm} \lambda^m_{ij} + C \sum_{m}^{N} (y^*_{ij}[m] - \lambda^m_{ij}/C) \log(y^*_{ij}[m] - \lambda^m_{ij}/C)] \tag{20}$$

with $Q_{lm} = K(\mathbf{x}[l], \mathbf{x}[m])$, to be minimized in the dual parameters subject to constraints

$$\sum_{m} \lambda^m_{ij} = 0 \tag{21}$$

$$\sum_{i,j} \lambda^m_{ij} = 0 \tag{22}$$

$$C(y^*_{ij}[m] - 1) \leq \lambda^m_{ij} \leq C y^*_{ij}[m], \qquad \forall \ i, j. \tag{23}$$

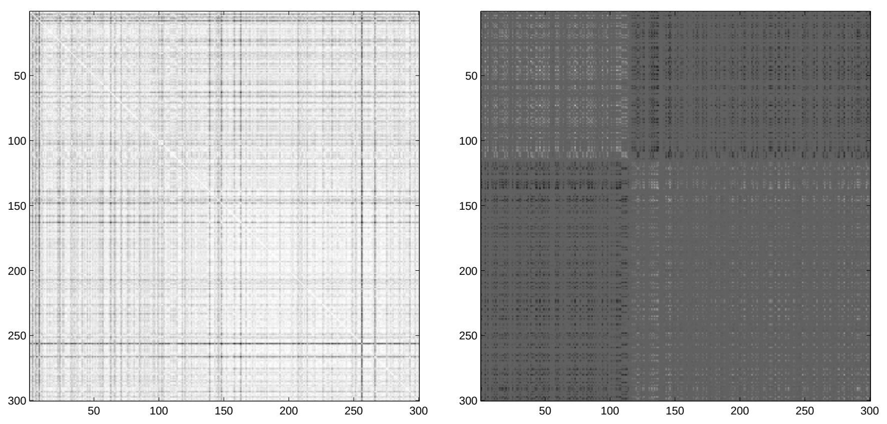


Fig. 1. *Kernel maps formed over UCI Pima-Indians database. (a) Linear Mercer kernel Q_{lm}. (b) Label "kernel" formed by inner-product of the coefficient vectors $\sum_{i,j} \lambda_{ij}^{l} \lambda_{ij}^{m}$ after training.*

Newton-Raphson based techniques can be used to solve the dual optimization problem (20) but exhibit slow convergence on account of the non-differentiability of the Shannon entropy factor in (20). *Gini*SVM [16] offers an approximate dual procedure based on a Gini quadratic form of entropy [17] to solve (20) efficiently. Like in soft-margin SVM classification, *Gini*SVM recasts the optimization problem into a quadratic programming (QP) problem, and produces a sparse kernel expansion.

In the following we assume the use of a linear kernel, and the distinction between primal and dual formulations becomes immaterial other than computational issues in the implementation.

3.2 Training Algorithm

The training algorithm can be summarized as follows:

1. Given the number of classes S, choose the number of sub-classes. This is determined by the constraints imposed by memory and computational resources for a specific application. Let the number of sub-classes be K. Therefore the total memory required scales with $K \times S$.
2. For each of the classes, partition the data vectors into disjoint K sub-classes using vector quantization or any clustering technique. This step is crucial to ensure that the equality (15) holds approximately.
3. Train probability regression with a linear kernel for K classes, to obtain $K \times S$ partition vectors $\mathbf{w}_{ij}$ and corresponding estimates $P_{ij}[n]$.
4. Re-estimate the new labels $y_{ij}^{*}[n]$ using (14) and retrain.

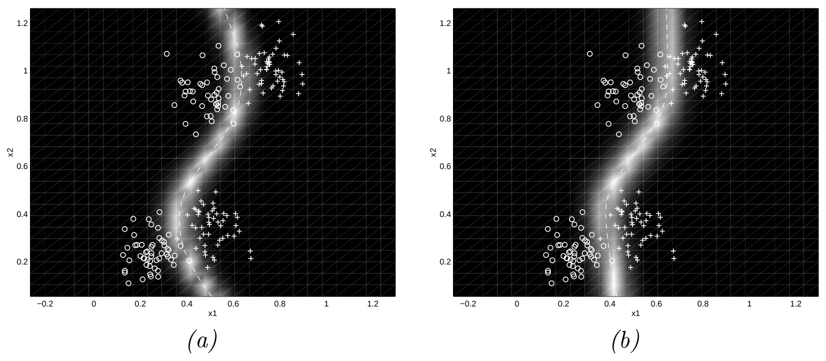


Fig. 2. *(a) 2-class decision surface obtained by SVM with a third order polynomial kernel. Support vectors in this case comprise 58% of the training data. (b) Decision surface obtained by PLM with linear kernel, and $K = 2$ partitions per class. Label partition vectors comprise 4% of the training data.*

In most cases two or three EM iterations are enough to obtain a good solution. It is also demonstrated in Section 4 that this re-estimation procedure prevents over-fitting which may occur when more sub-classes K are chosen than required for discrimination.

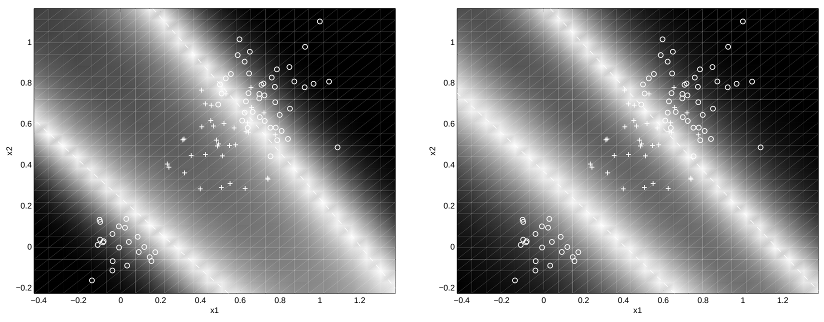
3.3 Correspondence between Label-Partitioning and Mercer Kernels

The similarity between using the partition method and using non-linear kernels can be observed through cost function (20) re-written as

$$H_2 = \frac{1}{2}\sum_{l}^{N}\sum_{m}^{N} Q_{lm}(\sum_{i,j}\lambda_{ij}^{l}\lambda_{ij}^{m}) + C\sum_{m}^{N}\sum_{i,j}(y_{ij}^{*}[m] - \frac{\lambda_{ij}^{m}}{C})\log(y_{ij}^{*}[m] - \frac{\lambda_{ij}^{m}}{C}). \quad (24)$$

The term $\sum_{i,j}^{M}\lambda_{ij}^{l}\lambda_{ij}^{m}$ could be interpreted as an inner product in a higher-dimensional 'label' space $\phi(\lambda^m).\phi(\lambda^l)$. This implies that even if the data kernel Q_{lm} is not powerful enough to model the desired non-linear decision boundaries, the kernel formed by inner-product of the coefficient label 'vectors' obtained by training augments the modeling power of PLM. Figure 1 compares the data kernel matrix Q_{lm} for a linear kernel with the PLM label 'kernel' matrix formed by the trained label coefficients $\sum_{i,j}\lambda_{ij}^{l}\lambda_{ij}^{m}$, using 300 training points from the UCI Pima-Indian dataset.

Unfortunately, unlike standard kernel machines the decision surface cannot be directly expressed in original label space. This necessitates explicit use of inner products of coefficients in partitioned label space, which is accomplished by means of the partition method of Section 3.



(a) (b)

Fig. 3. *(a) 2-Class decision surface formed by third order polynomial SVM with 20% support vectors. (b) 3-Partition PLM decision surface with 4% partitioned label vectors. Note that the solution obtained by SVM in this case is not a true large-margin solution in the original data-space, while PLM adjusts the margin locally by differentiating between clusters.*

4 Experiments and Results

The first set of experiments were performed with synthetic data generated using a mixture of Gaussians. The aim of the first set of experiments was to validate/observe the following:

1. PLM forms a decision surface similar to a full SVM solution but with much smaller number of partition labels than support vectors.
2. The large-margin decision surface does not change appreciably if larger number of sub-classes K are chosen than required. This illustrates that PLM does not over-fit in the scenario when more partition vectors are used than actually required. This is important because one cannot determine *a priori* the number of sub-classes necessary for good generalization.
3. The PLM formulation directly extends to the multi-class case, since the underlying probability model is multi-class (multinomial logistic).

Figures 2 illustrates the strong similarity in decision surface between PLM and conventional SVM classification, but with significantly fewer partition vectors $\mathbf{w}_{ij}$ than support vectors $\mathbf{x}[m]$. Figure 3 depicts an example where the SVM formulation does not produce large margin in data-space, because the true 'margin' of separation varies between pairs of clusters. Label partitioning allows to differentiate between clusters of data, and adapt the margin locally.

Figures 4 and 5 show the effect of choosing more partition vectors than actually required. One can observe in Figure 5 the effect of EM re-estimation on the large margin partitioned hyperplane, smoothing out artifacts of over-partitioning in Figure 4. Figure 6 shows probability contours obtained by applying the PLM method to the multi-class case.

The second set of experiments compares the performance of SVM with PLM on databases chosen from the UCI repository. For each of the selected datasets,

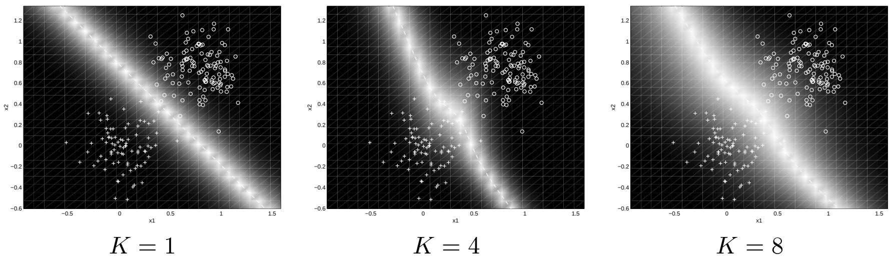


Fig. 4. *Decision surface obtained by increasing the total number of partitions before EM re-estimation.*

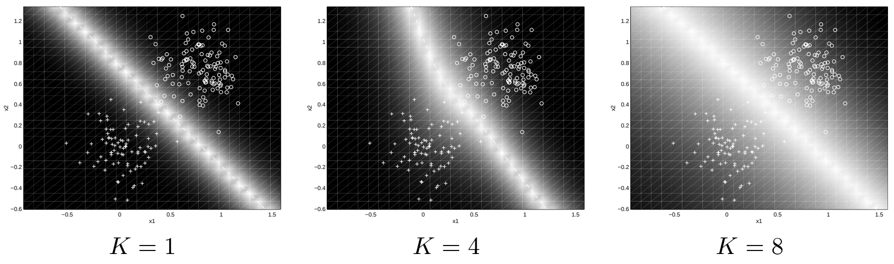
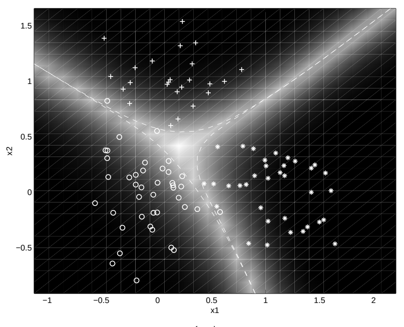


Fig. 5. *Decision surface obtained after EM re-estimation of partitions in Figure 4.*

a 10-fold cross-validation technique was used and the test errors were averaged over all the subsets. The size of the training set used N, the test error rates (Err) and the obtained number of support/partition vectors (#sv/#pv) are given in Table 1, which shows that PLM exhibits same or better generalization performance compared to binary soft-margin SVM, with improved sparsity in the representation.

Table 1. *Results on UCI data*

Database	N	SVM (Err)	PLM (Err)	SVM (#sv)	PLM (#pv)
Ionosphere	300	8.1%	7.2%	76	4
Pima Diabetes	650	21.6%	21.0%	284	4
Wisc Breast Cancer	600	3.0%	3.7%	81	6
Sonar	160	7.6%	5.7%	122	12



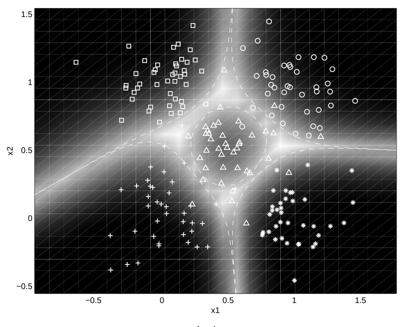


Fig. 6. *(a) Probability contours formed by using PLM for a 3-class problem with 9 partitions. (b) Probability contours formed by PLM for a 5-class problem with 10 partitions.*

5 Conclusions

We presented the *Partitioned Label Machine* (PLM) as a technique to obtain sparse classification and probability regression within the framework of large margin kernel methods and support vector machines. Advantages include:

1. PLM directly finds a decision surface subject to memory constraints imposed on the problem, expressed in number of label partitions.
2. The method is *non-parametric*, because there is no choice of kernel unlike support vector machines, and the only flexibility involves choosing the regularization constant C, and the number of sub-classes K.
3. PLMs provide comparable generalization performance in comparison with SVMs, but with much fewer partition labels than support vectors. Generalization performance of PLM is superior when the data is non-uniformly distributed, with variable margin in data space.
4. Because the algorithm is linear in each of the sub-classes, the training algorithm is fast, as only a single inner-product computation has to be performed to calculate the margin of any data vector.
5. The run-time classification is significantly reduced and only depends on the number of sub-classes K, which in turn can be fixed by memory constraints pre-specified during training.
6. The technique is very general and extends directly to multi-class problems.

Acknowledgment: This work was supported by the Catalyst Foundation, New York (*http://www.catalyst-foundation.org*), and National Science Foundation IIS-0209289.

References

1. Vapnik, V. *The Nature of Statistical Learning Theory,* New York: Springer-Verlag, 1995.

2. Schölkopf, B., Burges, C. and Smola, A., Eds., *Advances in Kernel Methods-Support Vector Learning,* MIT Press, Cambridge, 1998.
3. Osuna, E, and Girosi, F. "Reducing the Run-time Complexity in Support Vector Machines," *Advances in Kernel Methods-Support Vector Learning*, MIT Press, 1998.
4. Burges, C.J.C "Simplified Support Vector Decision Rules," *Proc.* 13th *Intl. Conf. on Machine Learning,* San Mateo, CA: Morgan Kaufmann, pp. 71-77, 1996.
5. Tipping, R. "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research,* vol. **1**, pp. 211-244, 2001.
6. Tresp, V. "A Bayesian Committee Machine," *Neural Computation,* vol. **12**, pp. 2719-2741, 2000.
7. Jacobs, R.A., Jordan, M.I., Nowlan, S.J and Hinton, G.E. "Adaptive Mixtures of Local Experts," *Neural Computation,* vol. **3** (1), pp. 79-87, 1991.
8. Collobert, R., Bengio, Y. and Bengio, S. "Scaling Large Learning Problems with Hard Parallel Mixtures," *IEEE Int. Conf. of Pattern Recognition: SVM workshop. (ICPR'2002),* Niagara Falls, 2002.
9. Kwok, J.T. "Support Vector Mixtures for Classification and Regression Problems," *Proc. of the International Conference on Pattern Recognition (ICPR),* pp. 255-258, Brisbane, Queensland, Australia, 1998.
10. Pollard, D. *Convergence of Stochastic Processes* New York: Springer-Verlag, 1984.
11. Bartlett, P "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network," *IEEE Transactions on Information Theory*, vol. **44** (2), March 1998.
12. Jebara, T. *Discriminative, Generative and Imitative Learning*, PhD Thesis, Media Laboratory, MIT, December 2001.
13. Wahba, G. *Support Vector Machine, Reproducing Kernel Hilbert Spaces and Randomized GACV*, Technical Report 984, Department of Statistics, University of Wisconsin, Madison WI.
14. 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.
15. Jaakkola, T. and Haussler, D. "Probabilistic kernel regression models," *Proc. Seventh International Workshop on Artificial Intelligence and Statistics,* 1999.
16. Chakrabartty, S. and Cauwenberghs, G. "Forward Decoding Kernel Machines: A hybrid HMM/SVM Approach to Sequence Recognition," *IEEE Int. Conf. of Pattern Recognition: SVM workshop. (ICPR'2002),* Niagara Falls, 2002.
17. Breiman, L. Friedman, J. H. et al. *Classification and Regression Trees*, Wadsworth and Brooks, Pacific Grove, CA, 1984.