How to Lose Confidence: Probabilistic Linear Machines for Multiclass Classification

Hui Lin, Jeff Bilmes

Department of Electrical Engineering
University of Washington
Seattle, WA 98195
{hlin,bilmes}@ee.washington.edu

Koby Crammer

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
crammer@seas.upenn.edu

Abstract

In this paper we propose a novel multiclass classifier called the probabilistic linear machine (PLM) which overcomes the low-entropy problem of exponential-based classifiers. Although PLMs are linear classifiers, we use a careful design of the parameters matched with weak requirements over the features to output a true *probability distribution* over labels given an input instance. We cast the discriminative learning problem as linear programming, which can scale up to large problems on the order of millions of training samples. Our experiments on phonetic classification show that PLM achieves high entropy while maintaining a comparable accuracy to other state-of-the-art classifiers.

Index Terms: multiclass classification, probability output, over-confident classifier, linear programming

1. Introduction

Classifiers based on exponential family functions (e.g., multilayer perceptrons, Gaussian classifiers and support vector machine with Gaussian kernels) are widely used in many areas such as speech signal processing and pattern recognition. Despite of their success, due to the nature of the exponential functions, these models are typically very confident in their prediction, putting most of the score or probability mass on a single label, excluding alternative labelings. This is even true when a given sample is far from the training data, so such classifiers are even confident when there is a training/test data mismatch (as is common in speech applications).

In cascaded systems, this property may cause drop in the overall performance since early-on all the labels but a very few are essentially ruled out, not allowing later stages of the system a good alternative label to choose from. For instances in HMM-based speech recognition where pruning based on the partial log-likelihood (e.g., the forward computation) or local posterior probabilities (e.g., hybrid ANN/HMM systems) is very common, concentrating too much probability mass on the top class will leave little accuracy for the alternative classes. Having a less confident but still accurate local (frame-by-frame) classifier, and leaving some probability available for the alternate local hypotheses (e.g., effectively expressing a soft ranking of the local labels) could make pruning algorithms work better. Note that, in general, it is **not** sufficient to exponentiate and then rescale such low-entropy probabilities, as sometimes they have such low entropy that the probability of the low probability candidates is essentially numerical noise. Another place that may benefit from less-confident acoustic classifiers is in the combination of an acoustic model and language model score, where Gaussian mixture models are usually used as the observation distributions, and where scaling is therefore necessary.

Still another application, which is the primary motivation of the work here, is the Vocal Joystick project. Combining machine learning, signal processing and human-computer interaction, the Vocal Joystick (VJ) project [1] is an application that allows voice-based control for moving a computer mouse cursor, or robotic arms. It specifically targets individuals with motor impairments, while many able-bodied people also enjoy using VJ for voice controlled drawings and multi-modal interaction. The VJ systems use vowel quality to control the motion direction and loudness to control the speed. For instance, 4-class vowel system uses four vowels to map to the four principle directions of up, down, left and right, and in 8-vowel system more precise diagonal directions are possible. To provide users with a high degree of control, VJ uses output probabilities as mixing weights to estimate vowel quality. An unwanted side effect of using classifiers that are based on exponential functions is that only a single direction of the pre-defined classes is usually dominant.

To overcome this problem, Malkin and Bilmes [2] proposed the ratio semi-definite classifiers (RSC) to obtain higher entropy posteriors on average, where ratios of polynomials are used instead of ratios of exponentials. Although RSCs were shown to work well in practice, they are learned using a non-convex optimization procedure which is often hard to optimize and only sub-optimal solutions can be found in practice. Furthermore, we felt that the accuracy of these classifiers could be further improved thus inspiring more research. Thus herein we present the *Probabilistic Linear Machines* (PLMs) where we replace the ratio between polynomials with simple linear terms, which, to start with, achieve even higher entropy posteriors. Intuitively, linear models usually gradually reduce probability values when transiting between classes, while the exponential posteriors (and even the greater-than-1^{st}-order polynomials of RSCs [2]) tend to have more dramatic and sudden changes resulting in higher entropy (see Fig. 1). This intuition is verified by our experimental results, where PLMs have high-entropy on average while remaining sufficient accuracy (see Sec. 5).

As suggested by its name, a PLM is essentially a linear classifier which is learned using linear programming (LP). Recent advances [3] in LP optimization resulted in algorithms that can scale up and efficiently solve problems with millions of variables or constraints, and are implemented as general purpose solvers like CPLEX [4] and MOSEK [5]. Note that ℓ_1 norm support vector machines [6, 7, 8] are also formulated as an LP. Our approach is distinguished by that fact that it incorporates multiclass support and probability output *simultaneously* in one

single LP optimization step.

SVMs were originally designed for binary classification. Only later they were extended to multiclass classification tasks, often in one of two ways: a reduction to binary classification or a direct approach. In the first case (some known as the "one-versus-rest" or "one-versus-one") the problem is first decomposed into a set of binary classification tasks, which later are combined to a single multiclass prediction (e.g., [9]). Platt et. al. [10] proposed to organize the binary problems in a directed acyclic graph. The second approach (e.g., by Crammer and Singer [11]) constructs a direct optimization formula. Our classifier also belongs to this latter category.

Unlike SVMs, PLMs produce calibrated probabilistic outputs and not general score values. Such posterior probabilities are very useful in many ways: they can be used to minimize the decision risk, compensate class priors, produce rankings, and are essential for model combinations since they are "normalized". There are several approaches to combine the large margin method with probabilistic outputs: Platt's [12] approach is the most notable example, where a sigmoid is applied to the output of the SVM. Platt also proposed a few methods to calibrate the parameters of the sigmoid. On the other hand, PLMs directly output probability distributions by incorporating probabilities in the learning optimization. Indeed, whether to convert non-probabilistic output into probabilities or directly learn the classifiers that output probabilities is still an open question [13], which we plan to further investigate in our future work.

2. Probabilistic Linear Machines

Our classifier produces probabilities as follows:

$$p(y|\mathbf{x}) = \frac{\mathbf{w}_y^T \phi(\mathbf{x})}{\sum_z \mathbf{w}_z^T \phi(\mathbf{x})} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input vector ,$y \in \{1, ..., K\}$ is a class labels, $\phi(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}_+^D$ is a mapping into non-negative feature values, and the parameters to learn are the weights $\mathbf{w}_y, y = 1, ..., K$. A valid probability distribution is obtained by constraining the value of $\mathbf{w}_y$ to be non-negative. The specific mapping $\phi(\cdot)$ we used is described in Sec. 5.

Note that $p(y|\mathbf{x})$ is a fractional linear function of both $\mathbf{w}_y$ and (as we will see) $\mathbf{x}$. Since a fractional linear function is quasi-convex [3], the learning of parameter $\mathbf{w}_y$ could have been optimized by solving a series of convex optimization problems. We, however, take an alternative approach, and formalize the learning problem using linear programming, where we further simplify the basic form: we pose the following element-wise simplex weight constraints,

$$\sum_{y=1}^{K} \mathbf{w}_y = \mathbf{1}_{D\times 1} \tag{2}$$

so that

$$p(y|\mathbf{x}) = \frac{\mathbf{w}_y^T \phi(\mathbf{x})}{\mathbf{1}_{1\times D} \phi(\mathbf{x})} = \frac{\phi(\mathbf{x})^T \mathbf{w}_y}{||\phi(\mathbf{x})||_1} . \tag{3}$$

where $\mathbf{1}_{D\times 1}$ is a vector in $\mathbb{R}^D$ with its elements equal to 1 and $||\cdot||_1$ represents ℓ_1 norm. This results in a distribution $p(y|\mathbf{x})$ which is a linear function in $\mathbf{w}_y$, while the probability properties are still preserved (i.e., $p(y|\mathbf{x}) \geq 0$, $\sum_y p(y|\mathbf{x}) = 1$). Moreover, without loss of generality, we assume that $\phi(\mathbf{x})$ is normalized such that $||\phi(\mathbf{x})||_1 = 1$, so that Eq. (3) can be written as

$$p(y|\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_y = \mathbf{w}_y^T \phi(\mathbf{x}) . \tag{4}$$

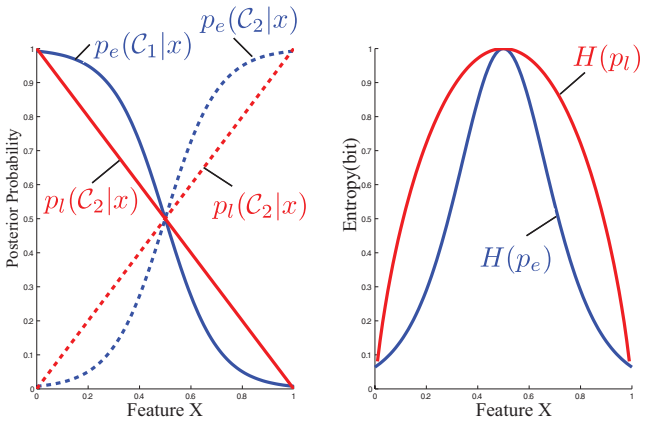

Figure 1: An illustration of larger entropy property of linear models comparing to exponential models. Consider a two class problem with classes $\mathcal{C}_1$ and $\mathcal{C}_2$, and a posterior distribution where $p(\mathcal{C}_1|x = 0) = p(\mathcal{C}_2|x = 1) = 1, p(\mathcal{C}_1|x = 0.5) = p(\mathcal{C}_2|x = 0.5) = 0.5$ and $p(\mathcal{C}_1|x = 1) = p(\mathcal{C}_2|x = 0) = 0$. When linear models are used, the posterior probability distribution typically has a form shown on the left figure in red ($p_l(\cdot|x)$), while exponential models usually end up with the distribution shown on the left figure in blue ($p_e(\cdot|x)$). The right figure shows the resulting entropy. As can be seen, the entropy of the exponential model is never greater than the linear model.

Specifically, the element-wise simplex constraints remove the unknown variable $\mathbf{w}_y$ from the denominators of Eq. (1), making it a linear function on $\mathbf{w}_y$. In the rest of the paper, we simplify the presentation and denote the model as follows:

$$p(y|\mathbf{x}) = \mathbf{x}^T \mathbf{w}_y = \mathbf{w}_y^T \mathbf{x} \tag{5}$$

We will revisit ϕ in Sec. 5.

Note that a similar trick was already used in the context of semidefinite probabilistic models [14]. Also, Cesa-Bianchi et. al. [15] suggest to force the output of linear classifier to be in the $[0, 1]$ range by assuming both the feature vector $\mathbf{x}$ and the weight vector $\mathbf{w}_y$ lie in a ball of radius 1. However, there is no simple and direct extension of this approach to multiclass problems.

As mentioned in Sec. 1, one benefit of having such linear forms is that, when a point moves between classes, linear models usually gradually reduce probability, while the exponential posterior tends to suddenly and rapidly change, and for most of the way, resulting in higher entropy. Fig. 1 illustrates one example. Another benefit, as we will see in the following section, is that we can use linear programming to obtain the global optimal solution of the learning problem.

3. Learning Methods

Given a training set $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1,...,N}$, we seek a set of parameters $\{\mathbf{w}_y\}_y$ that result in a good classifier. A good classifier should assign higher probability to correct classes than those to incorrect classes. Moreover, it is sufficient to maximize the *margin* between the probability for correct class $p(y_i|\mathbf{x}_i)$ and the incorrect ones $p(z|\mathbf{x}_i)$ where $z \neq y_i$.

We define the margin of the classifier on an example $\mathbf{x}_i$ as

$$m_i = p(y_i|\mathbf{x}_i) - \max_{z\neq y_i} p(z|\mathbf{x}_i) . \tag{6}$$

As in standard non-separable SVMs our learning algorithm

trades-off between maximizing the minimal margin and allowing some of the examples to be outliers, formalized as:

$$\max_{\eta,\{\mathbf{w}_y\},\{\xi_i\}} \eta - \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{7}$$

$$\text{subject to: } \mathbf{x}_i^T(\mathbf{w}_{y_i} - \mathbf{w}_z) \geq \eta - \xi_i, \quad \forall z \neq y_i,\ \forall i$$
$$1 \geq \xi_i \geq 0, \quad i = 1, ..., N$$
$$1 \geq \eta \geq 0$$
$$\mathbf{w}_y \succeq 0, \quad y = 1, ..., K$$
$$\sum_{y=1}^{K}\mathbf{w}_y = \mathbf{1}_{D\times 1}$$

where C is a constant that controls the trade-off between the classifier margin and the training errors, η represents the minimum margin on the training set and $\{\xi_i\}$ are slack variables. In this optimization problem, the unknown variables are $\{\mathbf{w}_y\}_y$, η, and $\{\xi_i\}_i^N$.

It is easy to verify that both the objective function and the constraints are linear in these variables. Therefore, the optimization problem in Eq. (7) is a linear programming (LP) problem, which can be solved using standard LP solvers like CPLEX [4] and MOSEK [5]. Note that LP has also been applied to the margin-based learning problem of linear SVMs [6, 7, 8], where the LP has similar forms as the LP proposed here. Our LP formulation of PLMs is distinguished from previous work since it incorporates multiclass support and probability output simultaneously in one single LP optimization problem, and therefore can be solved with a global optimum guarantee.

Algorithm 1 Active working set for solving LP 7

1: Input $(\mathbf{x}_i, y_i)_{i=1,...,N}, C, \epsilon, \mathcal{D}_c, c = 1, ..., M$
2: Initialize $\mathcal{A} \leftarrow \emptyset, \mathcal{D}_c = \mathcal{D}_1$
3: **repeat**
$$(\mathbf{w}^*, \eta^*, \xi^*) \leftarrow \underset{\mathbf{w}\geq 0, 1\geq\xi_i\geq 0, 1\geq\eta\geq 0}{\operatorname{argmax}} \eta - \frac{C}{N}\sum_{i=1}^{N}\xi_i$$
$$\text{subject to: } \mathbf{x}_i^T(\mathbf{w}_{y_i} - \mathbf{w}_z) \geq \eta - \xi_i, \quad \forall (i,z) \in \mathcal{A}$$
$$\sum_y \mathbf{w}_y = \mathbf{1}_{D\times 1}$$
5: $f \leftarrow$ `true`
6: **for** $i = 1, ..., N$ **do**
7: **if** $\min_z \mathbf{x}_i^T(\mathbf{w}_{y_i}^* - \mathbf{w}_z^*) < \eta^* - \xi_i^* - \epsilon$ **then**
8: $f \leftarrow$ `false`
9: **if** $i \in \mathcal{D}_c$ **then**
10: $\mathcal{A} \leftarrow \mathcal{A} \cup (i, \arg\max_{z\neq y_i} \mathbf{x}_i^T\mathbf{w}_z^*)$
11: **end if**
12: **end if**
13: **end for**
14: $\mathcal{D}_c \leftarrow$ next data chunk
15: **until** $f =$ `true`

4. Active working set algorithms

Although the LP formulation of PLM above can be solved with general purpose solvers, in practice they are not useful since these standard solvers often first load the entire optimization problem directly into memory. Thus machines with large amount of memory are required for solving even medium sized problems. We note that the size of the PLM formulation as a LP is linear in feature dimension D, the number of classes K, and the number of training samples N, yielding a total memory complexity of $\mathcal{O}(KDN)$. For the large scale problems that often arise in speech processing and recognition, hundreds of thousands of training examples are involved. Furthermore, the number of classes and the input feature dimensions are large enough to significantly increase in the problem size. Hence, standard solvers are not adequate. For example, MOSEK requires 64G memory to solve the 8-vowel classification task that we used (see Sec. 5). To overcome this problem, we develop an active working set algorithm (Algorithm 1).

The basic idea is that only a small fraction of the large number of constraints in Eq. (7) actually affects the optimal solution. The examples that are classified with large margins have no influence on the learned parameters. This is similar to the notion of support vectors in SVMs where examples that lie near the class decision boundary constitute a small portion of the entire training set. Furthermore, for each multiclass training example, often at most one additional class (besides the correct class) will affect the optimal solution. In other words, although we have $K-1$ constraints $\mathbf{x}_i^T(\mathbf{w}_{y_i} - \mathbf{w}_z) \geq \eta - \xi_i, \forall z \neq y_i$ for each training example i, only one of them is active in the final solution, i.e., the one corresponding to $\max_{z\neq y_i} p(z|\mathbf{x}_i)$, which we call the *active constraints*. Formally, active constraints are constraints that have positive Lagrange multipliers in the dual. Our active working set algorithm iteratively adds active constraints and removes inactive ones.

Specifically, we denote by $\mathcal{A}$ the current active training examples i and the corresponding active competing class z. We further reduce the memory complexity on each iteration by dividing the training set into M small chunks $\mathcal{D}_c, c = 1, ..., M$ and only use a single chunk in each iteration. We found that keeping the chunks class balanced improved the convergence rate of the final classifier (assuming the original training set is balanced).

We use a parameter denoted by ϵ to balance between training accuracy and training speed. When $\epsilon = 0$, the active working set algorithm finds exactly the same solution as solving the LP directly. In step 4, Algorithm 1 optimizes the objective of PLM (7) subject to a *subset* of the constraints. Then in steps 5-14, the algorithm verifies that all the non-active constraints are satisfied up to accuracy of ϵ. Constraints that are ϵ-violated are added to the active set. Clearly, the smaller the value of ϵ is, more constraints will be added to the active set, and the algorithm will require more memory and time. On the other hand, large values of ϵ yield only approximate solutions, which are obtained in less time and using less memory. We observed that the active working set algorithm often uses very small set of constraints (i.e., the size of $\mathcal{A}$) compared with the original problem.

5. Experiments

We evaluated our model on the Vocal Joystick Vowel Corpus [16] collected specifically for the VJ project. We created a training set from 21 recording sessions (2 speakers appear twice, although there is only partial overlap in their sounds), a development set of 4 speakers, and a test set of 10 speakers[1]. All speakers come from the earlier data collection efforts described by Kilanski et. al. [16] and capture the wide variability in human vowel production.

We tested two conditions: 4-vowel (æ, ɑ, u, i) classification and 8-vowel (with additional four vowels: a, o, ɨ, e) classification. For the 4-vowel case, there are approximately 275K training frames ($1,931$ utterances), and for the 8-vowel case, there are 550K frames ($3,867$ utterances). The development set

[1]The VJ corpus is available free online: `http://ssli.washington.edu/vj`

4-vowel	Accuracy (%)			Entropy		
Frames	1	3	7	1	3	7
PLM	97.4	98.2	98.5	1.97/0.02	1.97/0.02	1.97/0.02
RSC	95.7	97.5	98.1	0.85/0.43	1.13/0.31	0.83/0.37
MLP	97.4	98.2	98.6	0.66/0.40	0.62/0.38	0.31/0.29
Gaus.	97.2	95.2	93.2	0.07/0.19	0.08/0.21	0.06/0.19
8-vowel	Accuracy (%)			Entropy		
Frames	1	3	7	1	3	7
PLM	69.0	72.1	73.2	2.98/0.01	2.98/0.01	2.98/0.01
RSC	68.5	71.2	73.4	2.14/0.32	2.51/0.23	2.53/0.22
MLP	71.3	72.2	72.7	1.03/0.58	1.04/0.56	0.90/0.58
Gaus.	69.7	67.6	56.5	0.71/0.57	0.57/0.55	0.29/0.39

Table 1: Dev. set results, VJ Corpus. Entropies given as mean/dev.

4-vowel	Accuracy (%)			Entropy		
Frames	1	3	7	1	3	7
PLM	90.3	91.4	91.8	1.97/0.02	1.96/0.02	1.97/0.02
RSC	88.7	90.1	89.9	0.97/0.43	1.21/0.33	0.98/0.40
MLP	90.6	91.1	91.6	0.73/0.42	0.70/0.42	0.41/0.35
Gaus.	89.8	87.9	85.8	0.13/0.26	0.11/0.24	0.10/0.25
8-vowel	Accuracy (%)			Entropy		
Frames	1	3	7	1	3	7
PLM	61.6	63.4	63.8	2.98/0.01	2.98/0.01	2.98/0.01
RSC	62.1	63.2	63.4	2.17/0.31	2.52/0.21	2.55/0.21
MLP	67.2	67.8	68.4	1.08/0.58	1.11/0.58	0.97/0.61
Gaus.	61.9	60.7	54.3	0.69/0.57	0.54/0.53	0.31/0.41

Table 2: Test set results, VJ Corpus. Entropies are given as mean/dev.

was used to tune the parameters of the models, and the best parameters found in the development set were then applied to the test set to get the test set results. The dev set includes frames for 52K frames (385 utterances) 4-vowel task, and 109K frames (777 utterances) for 8-vowel task. The test set has 116K frames (716 utterances) for 4-vowel task and 236K (1432 utterances) frames for 8-vowel task. In these tasks, each utterance contains a single speaker uttering a single vowel. We used MFCCs with first-order deltas yielding vectors with 26 distinct features (frames were 25ms long with a 10ms shift) and we also varied the number of frames in the feature window as well.

We used MOSEK [5] as our base LP solver, and in the 8-vowel classification task we used the active working set algorithm introduced in Sec. 4 to overcome the memory issue when using the solver directly. The mapping function $\phi(\cdot)$ we used is $\phi(x_1, ..., x_n) = (1, \max(x_1, 0), \max(-x_1, 0), ..., \max(x_n, 0), \max(-x_n, 0))$ which ensures non-negativity and is very fast to compute. We compared our model to two exponential models, a 2-layer MLP and a Gaussian classifier using a single full-covariance Gaussian per class. The RSC [2] was also compared in our experiments.

The results of classification accuracy and the mean and standard deviation of entropies of posteriors over all frames in the development set are shown in Table 1, and the results for test set are shown in Table 2 [2]. For most of the classifiers, as the number of frames in the feature window grows, the classification accuracy increase, except for Gaussian classifier (perhaps because no regularization during training, such as shrinkage or l_2 on the Cholesky factorization, was used in this case).

For the 4-vowel task, PLM achieves the highest classification accuracy most of the time on both development and test set. It has similar classification accuracy performance to the MLP, and outperforms RSC and Gaussian classifiers.

On the 8-vowel task, the classification accuracies of PLM are no longer the best. One reason is that we use the active learning set algorithm in this task with $\epsilon = 0.001$, which yields only an approximate solution. Also, as the number of classes increase, better features (i.e., a different ϕ function) may be required for a linear classifier (we plan to pursue this in future work). Nevertheless, PLMs still have better performance than Gaussian classifiers, and also similar performances to RSC in classification accuracy, and much better entropy properties. Indeed, PLMs achieve high accuracy while achieving the highest average posterior entropy in all cases.

Acknowledgments: We thank Jonathan Malkin for sharing his VJ corpus setups. This work is supported by an ONR MURI grant (No. N000140510388) and an NSF grant (No. IIS-0326382).

[2] Note that the average entropy results for PLM in the 8-vowel case differ only at the 4^{th} digit of precision, so they look identical in the table.

6. References

[1] J. Bilmes, J. Malkin, X. Li, S. Harada, K. Kilanski, K. Kirchhoff, R. Wright, A. Subramanya, J. Landay, P. Dowden, and H. Chizeck, "The vocal joystick," in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Toulouse, France, 2006.

[2] J. Malkin and J. Bilmes, "Ratio semi-definite classifiers," in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, April 2008.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[4] Ilog, Inc., "Solver cplex," 2003. [Online]. Available: http://www.ilog.fr/products/cplex/

[5] "Mosek," 2006. [Online]. Available: http://www.mosek.com/

[6] P. Bradley and O. Mangasarian, "Massive data discrimination via linear support vector machines," *Optimization Methods and Software*, vol. 13, pp. 1–10, 2000.

[7] S. Sra, "Efficient Large Scale Linear Programming Support Vector Machines," *Machine Learning: ECML 2006*, vol. 4212, p. 767, 2006.

[8] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm Support Vector Machines," in *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Bradford Book, 2004.

[9] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, 2002.

[10] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," *Advances in Neural Information Processing Systems*, vol. 12, no. 3, pp. 547–553, 2000.

[11] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *The Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[12] J. Platt, "Probabilities for SV machines," *Advances in Neural Information Processing Systems*, pp. 61–74, 1999.

[13] A. Niculescu-Mizil and R. Caruana, "Obtaining calibrated probabilities from boosting," in *Proc. 21st Conference on Uncertainty in Artificial Intelligence (UAI'05), AUAI Press*, 2005.

[14] K. Crammer and A. Globerson, "Discriminative learning via semidefinite probabilistic models," in *Uncertainty in Artificial Intelligence*, 2006.

[15] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "Margin-based algorithms for information filtering," *Advances in Neural Information Processing Systems*, pp. 487–494, 2003.

[16] K. Kilanski, J. Malkin, X. Li, R. Wright, and J. Bilmes, "The Vocal Joystick data collection effort and vowel corpus," in *Interspeech*, Pittsburgh, PA, Sept. 2006.