MULTI-CLASS SVM OPTIMIZATION USING MCE TRAINING WITH APPLICATION TO TOPIC IDENTIFICATION

Timothy J. Hazen

MIT Lincoln Laboratory
Lexington, Massachusetts, USA

ABSTRACT

This paper presents a minimum classification error (MCE) training approach for improving the accuracy of multi-class support vector machine (SVM) classifiers. We have applied this approach to topic identification (topic ID) for human-human telephone conversations from the Fisher corpus using ASR lattice output. The new approach yields improved performance over the traditional techniques for training multi-class SVM classifiers on this task.

Index Terms— SVM classifiers, MCE training, topic identification.

1. INTRODUCTION

Support vector machines (SVMs) have recently been used used in a variety of text and speech classification problems including topic identification, speaker identification, and language identification [3, 11]. However, the traditional method of training multi-class SVMs has limitations. Multi-class SVMs are typically created by independently training individual *one-vs.-all* classifiers for each class. Thus, for a classification problem with M classes, the traditional training procedure individually optimizes M different 2-class classifiers instead of globally optimizing a single M-class classifier. As a result, the scores of the different 2-class SVMs may not be well calibrated for comparison within an M-class classifier.

To address the short-comings of traditional multi-class SVMs, a variety of methods have been proposed. For example, Crammer and Singer [4] have presented a extension to soft-margin training for the multi-class SVM scenario. Platt [13] addressed the calibration issue by fitting sigmoid functions to the outputs of the SVM classifiers to generate scores that approximated probabilities. The sigmoid functions were fitted to held-out data not used during the SVM training. Brummer [2] has alternatively used logistic regression classifiers trained on held-out data to calibrate the output scores of different classifiers. In our previous work [10], we successfully applied Brummer's calibration approach to a multi-class SVM system for topic identification (topic ID). In this paper, we present a minimum classification error (MCE) approach for calibrating multi-class SVM classifiers that does not require a held-out data set as was required in our previous work, but instead can use an efficient leave-one-out procedure applied directly to the training data.

Concerns about the soft-margin optimization approach that is typically used for training multi-class SVMs have also been raised in previous papers. For example, Arenas-García and Pérez-Cruz [1] argued for the use of empirical risk minimization optimization of multi-class SVMs instead of soft-margin optimization. Gao *et al* [7] have introduced a maximum figure of merit (MFoM) approach for optimizing the detection performance of SVM-based topic detection systems. In this paper, we further investigate the use of MCE training of multi-class SVMs as an alternative to soft margin training.

In our previous work, we have used MCE training to learn a weighting vector for rescaling the input feature space for either naive Bayes [9] or SVM classifiers [10]. In this paper, we also revisit the use of this technique in conjunction with our new MCE approach for training the SVM parameters themselves.

This work was sponsored by the Air Force Research Laboratory under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

2. MCE TRAINING OF SUPPORT VECTOR MACHINES

2.1. Overview of Multi-Class SVMs

Traditional SVM training finds a hyperplane which maximally separates positive and negative training tokens in a vector space [15]. SVMs are commonly trained with soft margin optimization which allows training tokens to fall within the separation margin of the decision hyperplane with some penalty.

In its standard form, an SVM is a 2-class classifier. To create a multi-class SVM for a problem with M classes, one would typically produce a one-vs.-all SVM classifier for each class m with the following scoring function for a test vector $\vec{x}$:

$$S(\vec{x}, m) = -b_m + \sum_{\forall i} \alpha_{i,m} K(\vec{v}_i, \vec{x}) \tag{1}$$

Here, each vector $\vec{v}_i$ is a unique training token or *support vector* from the full collection of training tokens covering all classes. Each $\alpha_{i,m}$ value represents the learned *support vector* weight for the training token i for the SVM classifier for class m. For notational simplicity, the $\alpha_{i,m}$ values here absorb the ± 1 valued class labels $y_{i,m}$ which are often included in the SVM expression as presented in much of the literature. The b_m value represents a decision boundary offset value for the SVM scoring function. The function $K(\vec{v}, \vec{x})$ is a *kernel function* for comparing the vectors $\vec{v}$ and $\vec{x}$. While many kernel functions are possible, the primary optimization algorithm discussed in this paper is independent of the choice of kernel function.

Because of the geometric constraints imposed on the SVM learning process, the positive training tokens for class m will have positive valued support vector weights for $\alpha_{i,m}$, while the negative training tokens will have negative valued support vector weights. Furthermore, if the full collection of training vectors is divided into the set of positive tokens for class m, V_m^+, and the set of negative tokens for class m, V_m^-, the following equality also holds:

$$\sum_{\forall i: \vec{v}_i \in V_m^+} \alpha_{i,m} = \sum_{\forall i: \vec{v}_i \in V_m^-} -\alpha_{i,m} = a_m \tag{2}$$

Here, we have introduced the term a_m, which we will refer to as the SVM scaling factor for class m. We can further define a set of prescaled support vector weights as:

$$\omega_{i,m} = \alpha_{i,m}/a_m \tag{3}$$

We should also note that pulling the scale factor a_m out of the summation in Equation 2 yields this constraint:

$$\sum_{\forall i:\vec{v}_i \in V_m^+} \omega_{i,m} = \sum_{\forall i:\vec{v}_i \in V_m^-} -\omega_{i,m} = 1 \tag{4}$$

Next we can define the weighted sum of kernel scores for the positive and negative support vectors independently as:

$$R^+(\vec{x},m) = \sum_{\forall i:\vec{v}_i \in V_m^+} \omega_{i,m} K(\vec{v}_i,\vec{x}) \tag{5}$$

$$R^-(\vec{x},m) = \sum_{\forall i:\vec{v}_i \in V_m^-} \omega_{i,m} K(\vec{v}_i,\vec{x}) \tag{6}$$

From here, we define the full raw SVM score $R(\vec{x},m)$ to be:

$$R(\vec{x},m) = R^+(\vec{x},m) + R^-(\vec{x},m) \tag{7}$$

Using these definitions we can rewrite Equation 1 as:

$$S(\vec{x},m) = -b_m + a_m R(\vec{x},m) \tag{8}$$

Using this interpretation of the SVM expression, the discriminative capabilities of the individual 2-class SVMs created for each class m are captured by the set of class specific support vector weights $\omega_{i,m}$ contained within $R(\vec{x},m)$, while the calibration of the full multi-class SVM is captured by the settings of the scale factors, a_m, and decision thresholds, b_m.

2.2. MCE Training of SVM Parameters

To optimize our multi-class SVM system, we use the minimum classification error (MCE) training approach [12]. We begin by defining a misclassification measure:

$$\mathcal{M}(\vec{x}) = F(\vec{x},\{m \neq m_C\}) - S(\vec{x},m_C) \tag{9}$$

Here, $S(\vec{x},m_C)$ represents the classifier score for vector $\vec{x}$ for the correct class m_C, and $F(\vec{x},\{m \neq m_C\})$ is a function of the scores for all of the incorrect classes defined as follows:

$$F(\vec{x},\{m \neq m_C\}) = \frac{1}{\eta}\log\left[\frac{1}{M-1}\sum_{\forall m \neq m_C}\exp(\eta S(\vec{x},m))\right] \tag{10}$$

In this expression, all competing hypotheses contribute to the misclassification measure with the highest scoring competitors contributing the most. The η term is a posterior scaling factor for controlling the relative weighting of the incorrect classes.

The misclassification measure is then mapped by a sigmoid loss function onto the $[0,1]$ continuum as follows:

$$\ell(\vec{x}) = \frac{1}{1+\exp(-\beta\mathcal{M}(\vec{x}))} \tag{11}$$

Here, β represents the slope of the sigmoid function. The loss function approximates the classification error, thus $\ell(\vec{x})$ should approach zero for test vectors that are classified correctly and approach one for test vectors that are incorrectly classified.

The loss function can be differentiated with respect to the individual parameters of the SVM classifier. The partial derivatives of the loss function with respect to these parameters are used to update these parameters with the goal of minimizing the average value of the loss function over the training set. We will refer to a process that optimizes only the SVM scale and threshold parameters, a_m and b_m, as *calibration*. If we additionally train the prescaled support vector weights $\omega_{i,m}$, we will refer to this as *full optimization*.

While SVM training often selects a sparse subset of training vectors to be actual support vectors, we will simplify our derivation by assuming that all training vectors are used within the SVM expression in Equation 1. In this case, the non-support vectors simply carry a support vector weight of zero. It should be noted that the MCE optimization allows training vectors that start with an initial weight of zero to obtain a non-zero weight during the optimization process.

For MCE training we must first compute the partial derivatives of the loss function $\ell(\vec{x})$ with respect to each parameter. The expressions of these derivatives for each a_m, b_m and $\omega_{i,m}$ parameter are:

$$\frac{\partial\ell(\vec{x})}{\partial a_m} = \begin{cases} -\beta\ell(\vec{x})\left(1-\ell(\vec{x})\right)R(\vec{x},m) & \text{if } m = m_C \\ \gamma_m\beta\ell(\vec{x})\left(1-\ell(\vec{x})\right)R(\vec{x},m) & \text{if } m \neq m_C \end{cases} \tag{12}$$

$$\frac{\partial\ell(\vec{x})}{\partial b_m} = \begin{cases} \beta\ell(\vec{x})\left(1-\ell(\vec{x})\right) & \text{if } m = m_C \\ -\gamma_m\beta\ell(\vec{x})\left(1-\ell(\vec{x})\right) & \text{if } m \neq m_C \end{cases} \tag{13}$$

$$\frac{\partial\ell(\vec{x})}{\partial \omega_{i,m}} = \begin{cases} -\beta\ell(\vec{x})\left(1-\ell(\vec{x})\right)a_m K(\vec{v}_i,\vec{x}) & \text{if } m = m_C \\ \gamma_m\beta\ell(\vec{x})\left(1-\ell(\vec{x})\right)a_m K(\vec{v}_i,\vec{x}) & \text{if } m \neq m_C \end{cases} \tag{14}$$

Here, the γ_m parameters are posterior-like weights over the incorrect classes as defined by:

$$\gamma_m = \frac{\exp(\eta S(\vec{x},m))}{\sum_{\forall m_i \neq m_C}\exp(\eta S(\vec{x},m_i))} \tag{15}$$

As $\eta \to \infty$ then $\gamma_{m_I} \to 1$ for the best scoring incorrect class m_I, and $\gamma_m \to 0$ for all other incorrect classes.

In our system, the learning algorithm performs iterative batch updating of the parameters, i.e., the partial derivatives of the loss function are averaged over all N_X vectors in the optimization set and the parameters are then updated once at the end of each pass through the optimization data. The form of the updates are as follows:

$$a_m' = a_m - \frac{\epsilon}{N_X}\sum_{\forall\vec{x}}\frac{\partial\ell(\vec{x})}{\partial a_m} \tag{16}$$

$$b_m' = b_m - \frac{\epsilon}{N_X}\sum_{\forall\vec{x}}\frac{\partial\ell(\vec{x})}{\partial b_m} \tag{17}$$

$$\omega_{i,m}' = \omega_{i,m} - \frac{\epsilon}{N_X}\sum_{\forall\vec{x}}\frac{\partial\ell(\vec{x})}{\partial \omega_{i,m}} \tag{18}$$

Here ϵ is a learning rate parameter. We further regularize the collection of a_m and b_m terms after each iteration through the training data to adhere to these constraints:

$$\textstyle\sum_{\forall m} a_m = c_A \quad \text{and} \quad \sum_{\forall m} b_m = c_B \tag{19}$$

Here, c_A and c_B represent the sums of the initial values of the a_m and b_m terms before MCE training begins. We also regularize the

support vector weights to adhere to the constraints given in Equation 4 and we impose the constraint that the weights for the positive training tokens of a class must be non-negative while the weights for the negative training tokens must be non-positive.

During full optimization, we first optimize the a_m and b_m parameters while leaving the $\omega_{i,m}$ weights fixed, and then we optimize the $\omega_{i,m}$ weights while leaving the a_m and b_m parameters fixed. We iteratively alternate between these two optimization steps until convergence is reached.

2.3. Leave-One-Out Scoring for SVMs

When computing the loss $\ell(\vec{x})$ of a token $\vec{x}$ during the MCE optimization process, $\vec{x}$ would ideally be a token that is held-out or unseen during the initial training process. This is most easily achieved by optimizing over a set of data that is held out during the initial SVM training. A drawback of this approach is that less data is then available to train the initial SVM parameters. The *leave-one-out* training approach could provide a more efficient use of the training data by allowing all but the one held-out data point to be used for constructing the classification models. While leave-one-out training can be prohibitively expensive in terms of computation, we use a leave-one-out approximation to SVM scoring that can be run in a computationally efficient manner.

In leave-one-out training we must remove a specific training vector $\vec{v}_j$ from the training set, adjust the trained models to account for the removal of this vector, and then assess the accuracy of the model by computing its loss $\ell(\vec{v}_j)$ on the held out vector. Our approach approximates the score of an SVM model by removing the training vector as a support vector and renormalizing the weights of the remaining support vectors to account for the lost weight from the held-out vector. In the case that the held-out training vector $\vec{v}_j$ is a positive support vector for class m this requires that we adjust the computation of $R^+(\vec{v}_j, m)$ as follows:

$$R^+(\vec{v}_j, m) = \sum_{\forall i: i \neq j, \vec{v}_i \in V_m^+} \frac{\omega_{i,m}}{1 - \omega_{j,m}} K(\vec{v}_i, \vec{v}_j) \qquad (20)$$

Of course, this expression is not valid for the degenerate case where $\omega_{j,m} = 1$. In our experiments we have yet to encounter this case, but should this occur one could conceivably approximate $R^+(\vec{v}_j, m)$ by assigning uniform weights to the remaining collection of positive training vectors for class m. For the case where the held-out vector $\vec{v}_j$ is a negative support vector for class m, we similarly adjust the computation of $R^-(\vec{v}_j, m)$ as follows:

$$R^-(\vec{v}_j, m) = \sum_{\forall i: i \neq j, \vec{v}_i \in V_m^-} \frac{\omega_{i,m}}{1 + \omega_{j,m}} K(\vec{v}_i, \vec{v}_j) \qquad (21)$$

Using this leave-one-out approach to SVM model scoring, we can compute approximate values for $R(\vec{v}, m)$ and $\ell(\vec{v})$, thus allowing every training vector $\vec{v}$ in the training set to be used as an unseen observation during MCE training.

2.4. Training the Kernel Function

In the experiments in this work, we use a linear kernel function, i.e., $K(\vec{v}, \vec{x}) = \vec{v} \cdot \vec{x}$. In previous work we have added an additional feature weight to the input feature vector which serves to increase or decrease the contribution of different elements in the feature vector [10]. In this case, the kernel function becomes:

$$K(\vec{v}, \vec{x}) = \vec{v} \cdot (\vec{\lambda} * \vec{x}) \qquad (22)$$

Here the $*$ operator is simply a component-wise multiplication and $\vec{\lambda}$ is initialized as a vector of ones (and hence does not alter the initial SVM training). After SVM training is complete, MCE training can be used to optimize $\vec{\lambda}$. We described this process in our previous paper [10]. We have modified the n-fold jackknife training approach from our earlier work to adhere to the leave-one-out training paradigm described in Section 2.3. In this work, MCE training of the kernel function feature weights is performed before MCE optimization of the SVM parameters when both techniques are applied.

3. EXPERIMENTAL RESULTS

3.1. Data Set

Our experiments used the English Phase 1 portion of the Fisher Corpus containing 10-minute-long telephone conversations between two people. Each conversation discussed one topic chosen from a set of 40 prespecified topics [6]. For this work, the corpus was subdivided into three subsets: (1) a 553 hour recognizer training set containing 3104 calls, (2) a 244 hour topic ID training set with 1375 calls, and (3) a 226 hour topic ID test set with 1372 calls.

3.2. Speech Recognition Details

In our ASR-based experiments, a network, or *lattice*, of speech recognition hypotheses is generated for every audio segment from both conversation sides of every call. Within each lattice the posterior probability is computed for each hypothesized word. An *expected count* for each word within a call is then computed by summing the posterior scores over all instances of each word over all lattices.

For ASR we have used the MIT SUMMIT speech recognition system [8]. The system's acoustic models were trained using a standard maximum-likelihood approach on the full 553 hour recognition training set without any form of speaker normalization or adaptation. For language modeling, the system uses a basic trigram language model with a 31.5K word vocabulary trained using the transcripts of the recognizer training set. This system performs recognition faster than real time (on a current PC) but word error rates can be high (typically over 40%).

3.3. Experimental Details

In our experiments, every call is represented by a single feature vector containing the expected counts of the words observed during the call. Proper normalization of this feature vector is required for optimal topic ID performance. In this work we examine two normalization techniques: (1) term frequency/inverse document frequency normalization (TF-IDF) and (2) term frequency/log likelihood ratio (TF-LLR) normalization. For TF-IDF, we use a modified version of this standard technique in which the inverse document frequency component is estimated from the lattice term counts [16].

For TF-LLR normalization, the expected counts for each vocabulary word w are first converted into relative frequencies f_w, which are then normalized by the square root of the global *a prior* likelihood $P(w)$ of observing w (as estimated over all calls in the training set) as follows:

$$x_w = f_w / \sqrt{P(w)} \qquad (23)$$

When using a linear kernel function, TF-LLR yields a linear approximation to a log likelihood ratio within the SVM [3].

In our experiments we use the LIBSVM software package [5] for initial training of the SVM models. It is worth noting that our

training data in these experiments is fully separable (i.e. the error rate over the full training set is 0%). We use settings of $\beta = 10$ and $\eta = 100$ for MCE training, though we have found that performance is not overly sensitive to these settings and a wide range of β and η yield similar results. The learning rate ϵ is initially set to 1, but is adaptively adjusted using a modified version of the RPROP algorithm [14].

3.4. Results

Table 1 shows the topic ID performance of our approach on the test set. Performance is reported with two metrics: (1) the classification error rate (CER) for a 40-class closed-set topic classification evaluation, and (2) the equal error rate (EER) from the detection-error trade-off curve for a topic detection evaluation. The first column of the table shows the four different types of feature normalization and weighting used in our experiments. Here TF-LLR and TF-IDF are compared, both with and with-out additional MCE training of the feature weighting vector $\vec{\lambda}$. The second column of the table shows which style of our new MCE training approach is used. Here, performance of standard soft margin SVM training is compared against our MCE calibration and MCE full optimization techniques.

In the table, we observe that our new MCE approach for SVM calibration and optimization yields improved accuracy over the baseline SVM system using either TF-LLR or TF-IDF normalization. Next, we can observe that previously introduced method for MCE training of the feature weights also provides significant performance improvements over the baseline SVM system using either TF-LLR and TF-IDF normalization. Finally, when we combine MCE optimization of the SVM parameters with MCE training of the feature weights, we see additional modest improvements in performance.

Overall, the best performing SVM system used TF-LLR normalization with MCE training of both the feature weights and the SVM parameters. We also note that the TF-LLR normalization approach yields better results than the traditional TF-IDF normalization after our MCE optimization techniques are applied. The results from the best SVM system presented here are similar to results obtained with the best naive Bayes system using MCE trained feature weights that we have previously reported [9].

Table 1. Topic ID performance on Fisher call data from ASR output using various forms of SVM classifier optimization.

Normalization & Feature Weighting	MCE Training of SVM Parameters	CER (%)	EER (%)
TF-LLR	None	12.5	2.84
TF-LLR	Calibration	10.3	2.45
TF-LLR	Optimization	10.2	2.15
TF-LLR + MCE	None	8.6	2.21
TF-LLR + MCE	Calibration	8.6	2.19
TF-LLR + MCE	Optimization	8.1	2.06
TF-IDF	None	11.4	2.84
TF-IDF	Calibration	10.7	2.92
TF-IDF	Optimization	9.8	2.33
TF-IDF + MCE	None	9.9	2.40
TF-IDF + MCE	Calibration	9.5	2.41
TF-IDF + MCE	Optimization	9.1	2.33

4. SUMMARY

In this paper we have presented an MCE-based optimization approach for improving the accuracy of a multi-class SVM system for classification problems. We have applied this approach to topic ID for human-human telephone conversations using the lattice output from an ASR system. We have found that MCE-based optimization of SVM-based systems yields better accuracy than traditional SVM training techniques within our topic ID systems. Although we have applied these techniques to topic ID, we believe they may also be applied to other multi-class classification problems as well.

5. REFERENCES

[1] J. Arenas-García and F. Pérez-Cruz, "Multi-class support vector machines: A new approach," in *Proc. ICASSP*, Hong Kong, April 2003.

[2] N. Brummer and D. van Leeuwan, "On calibration of language recognition scores," in *Proc. The Speaker and Language Recognition Workshop*, San Juan, June 2006.

[3] W. Campbell, *et al*, "Support vector machines for speaker and language recognition," *Computer Speech & Lang.*, vol. 20, no. 2-3, pp. 210-229, 2006.

[4] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265-292, Dec. 2001.

[5] C.-C. Chang and C.-J. Lin, "LIBSVM – A Library for Support Vector Machines," available online at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[6] C. Cieri, D. Miller, and K. Walker, "The Fisher corpus: A resource for the next generation of speech-to-text," in *Proc. Int. Conf. on Lang. Resources and Eval.*, Lisbon, May 2004.

[7] S. Gao, W. Wu, C.-H. Lee and T.-S. Chua, "A MFoM learning approach to robust multiclass multi-label text categorization," in *Proc. ICML*, Banff, July 2004.

[8] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Lang.*, vol. 17, no. 2-3, pp. 137-152, 2003.

[9] T. J. Hazen and A. Margolis, "Discriminative feature weighting using MCE training for topic identification of spoken audio recordings," in *Proc. ICASSP*, Las Vegas, April 2008.

[10] T. J. Hazen and F. Richardson, "A hybrid SVM/MCE training approach for vector space topic identification of spoken audio recordings," in *Proc. Interspeech*, Brisbane, Sep. 2008.

[11] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. of Euro. Conf. on Machine Learning*, Chemnitz, April 1998.

[12] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Processing*, vol. 40, no. 12, Dec. 1992.

[13] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, pp. 61-74, MIT Press, 1999.

[14] M. Riedmiller and H. Braun, "RPROP - A fast adaptive learning algorithm," in *Proc. Int. Symposium on Computer and Information*, Antalya, Turkey, 1992.

[15] V. Vapnik, *Statistical Learning Theory,* John Wiley and Sons, New York, 1998.

[16] J. Wintrode and S. Kulp, "Confidence-based techniques for rapid and robust topic identification of conversational telephone speech", in *Proc. Interspeech*, Brighton, England, 2009.