interspeech 2008 incorporating SST 08

A Hybrid SVM/MCE Training Approach for Vector Space Topic Identification of Spoken Audio Recordings

Timothy J. Hazen and Fred Richardson

MIT Lincoln Laboratory
Lexington, Massachusetts, USA

Abstract

The success of support vector machines (SVMs) for classification problems is often dependent on an appropriate normalization of the input feature space. This is particularly true in topic identification, where the relative contribution of the common but uninformative function words can overpower the contribution of the rare but informative content words in the SVM kernel function score if the feature space is not normalized properly. In this paper we apply the discriminative minimum classification error (MCE) training approach to the problem of learning an appropriate feature space normalization for use with an SVM classifier. Results are presented showing significant error rate reductions for an SVM-based system on a topic identification task using the Fisher corpus of audio recordings of human-human conversations.

Index Terms: topic identification, topic spotting, MCE training, support vector machines

1. Introduction

Over the last ten years, support vector machines (SVMs) have become a popular choice of classifier for use in topic identification (topic ID) problems, achieving state-of-the-art results on various tasks [1]. The core training algorithm of an SVM classifier is well defined [2]. Thus, the ultimate success of SVMs for topic ID problems is often dependent on the creation and normalization of an appropriate input feature space and the selection of the SVM kernel function. In feature spaces based directly on word counts or relative word frequencies, the relative contribution of the common but uninformative function words can overpower the contribution of the rare but informative content words in the SVM kernel function score if the feature space is not normalized properly. Thus, in this paper we focus on the issue of feature normalization within an SVM classifier using a linear kernel. We will demonstrate how a minimum classification error (MCE) training algorithm for feature weighting can be adapted for use within a linear SVM approach to topic ID.

The work in this paper is motivated by our earlier success in applying an MCE training technique to the problem of learning feature weights within a naive Bayes approach to the topic ID problem [3]. In our earlier work we were able to separate the process of learning Bayesian statistics from the process of discriminatively learning feature weights. This allowed us to preserve the basic advantages of Bayesian learning while simultaneously providing significant accuracy improvements from a discriminative MCE training process.

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.

As we will show in Section 2, the SVM and naive Bayes classifier both use a linear projection matrix for discriminating between classes. Because of this fundamental similarity, we will show in Section 3 that our MCE approach to learning feature weights for a naive Bayes classifier can also be used with a linear SVM classifier. In Section 4 we demonstrate how our MCE training technique leads to significant improvements in the accuracy of an SVM system within topic ID experiments conducted on spoken audio files from the Fisher corpus.

2. Vector Space Topic Identification

2.1. The Naive Bayes Assumption

In traditional text-based topic ID systems, a document is represented as a sequence of words W. Probabilistic systems attempt to model the likelihood of W given a topic t, i.e. $P(W|t)$. Because of the difficulty in modeling this expression directly, a conditional independence (or naive Bayes) assumption between the words is typically applied. The use of this assumption results in the following log likelihood expression:

$$\log P(W|t) = \sum_{\forall w} c_w \log P(w|t) \tag{1}$$

Here c_w is the occurrence count of each unique vocabulary word w within W, and the sum is computed over all words in the full vocabulary of the classifier. The order of the words in W is thus completely ignored when using the naive Bayes assumption.

In our naive Bayes system we apply a hypothesis testing log likelihood ratio approach in which the score for any document/topic pair, W and t, can be expressed as:

$$S(W,t) = \log \frac{P(W|t)}{P(W|\bar{t})} = \sum_{\forall w} c_w \log \frac{P(w|t)}{P(w|\bar{t})} \tag{2}$$

Here, $\bar{t}$ refers to the hypothesis that the topic is not t.

2.2. The Vector Space Interpretation

The naive Bayes classifier is a specific instance of a larger set of classifiers known as linear (or vector space) classifiers. A standard linear classifier can be described in terms of the following matrix operation:

$$\vec{s} = \mathbf{R}\vec{x} \tag{3}$$

Here $\vec{x}$ is a feature vector describing the test data, R is a matrix representing the linear classifier, and $\vec{s}$ is a vector of classifier scores, one per topic. Borrowing from prior work in call routing, we will refer to the matrix R as the *routing matrix* [4].

The naive Bayes classifier is easily represented in this vector space interpretation. The feature vector $\vec{x}$ is simply comprised of the count information for each word in the vocabulary. The dimension of this vector, N_V, is the number of unique

words in the system's vocabulary. In the experiments presented in this paper, the count values are all normalized into relative frequency distributions as follows:

$$x_w = \frac{c_w}{|W|} \tag{4}$$

Thus, the L1 norm of $\vec{x}$ is always equal to one:

$$\sum_{\forall w} x_w = 1 \tag{5}$$

When using the naive Bayes approach, the routing matrix is populated with the estimated log likelihood ratios described in Equation 2. Thus, the routing matrix has dimension $N_T \times N_V$ (where N_T in the number of topics in the topic set) and the individual elements of the matrix R can be defined as:

$$r_{t,w} = log\frac{P(w|t)}{P(w|\bar{t})} \tag{6}$$

2.3. The SVM Approach

The support vector machine (SVM) approach finds a hyperplane which (if possible) maximally separates positive and negative training instances in some vector space. In its standard form, an SVM is a two-class classifier. Thus, for each topic t, we will produce a one-vs.-all (or in-topic vs. not-in-topic) SVM classifier with the following scoring function:

$$S(\vec{x}, t) = -b_t + \sum_{\forall i} \alpha_{i,t} K(\vec{v}_i, \vec{x}) \tag{7}$$

Here, each vector $\vec{v}_i$ is a unique relative frequency training instance from the full collection of training documents covering all topics. Each $\alpha_{i,t}$ value represents the learned *support vector* weight for the specific training instance i for the SVM classifier for topic t. The b_t value represents the decision boundary value for the SVM hyperplane projection. 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, linear kernel functions have typically been found to be sufficient for topic ID problems (i.e., the positive and negative training instances are fully separable by a single linear hyperplane in the original vector space). The linear kernel function is simply the dot product $\vec{v} \cdot \vec{x}$.

2.4. SVM Term Weighting

In practice, using the raw relative frequencies within the SVM approach is sub-optimal because the dot product $\vec{v} \cdot \vec{x}$ will be dominated by the most frequently used words, which are typically non-informative function words (e.g., articles, conjunctions, prepositions, etc.). To compensate for this effect a variety of term normalization techniques, such as inverse document frequency weighting, have been proposed for the purpose of giving greater weight to the less frequent words which are generally more information for topic ID. Applying term normalization is equivalent to using the following kernel function:

$$K(\vec{v}, \vec{x}) = (\vec{\phi} * \vec{v}) \cdot (\vec{\phi} * \vec{x}) \tag{8}$$

Here $\vec{\phi}$ is a term weighting vector, and the expression $(\vec{\phi} * \vec{v})$ generates a new vector whose value for each word (or feature) w is equal to $\phi_w v_w$.

In experiments, our best accuracies have come from weighting each feature w by the inverse square root of the feature's global frequency such that each element ϕ_w within $\vec{\phi}$ is:

$$\phi_w = 1/\sqrt{P(w)} \tag{9}$$

Here, $P(w)$ is a maximum a posteriori probability (MAP) estimate learned from the training data. As shown in [5], the use of this normalization within a linear kernel function results in an SVM whose scoring function can be viewed as a linear approximation of a probabilistic log likelihood ratio scoring function.

2.5. The SVM Routing Matrix

The kernel function of Equation 8 can be inserted into Equation 7 to yield this expression:

$$S(\vec{x}, t) = -b_t + \sum_{\forall i} \alpha_{i,t} (\vec{\phi} * \vec{v}_i) \cdot (\vec{\phi} * \vec{x}) \tag{10}$$

To view the SVM approach in the same form as Equation 3, we first convert Equation 10 into this equivalent form:

$$S(\vec{x}, t) = -b_t + \sum_{\forall i} \alpha_{i,t} (\vec{\phi} * \vec{\phi} * \vec{v}_i) \cdot \vec{x} \tag{11}$$

Next, let us define a vector $\vec{b}_t$ as an N_V dimension vector in which every dimension has the value of b_t. Using this definition in conjunction with the L1 norm constraint on $\vec{x}$ given in Equation 5, we can state:

$$\vec{b}_t \cdot \vec{x} = b_t \tag{12}$$

By substituting $\vec{b}_t \cdot \vec{x}$ for b_t, we can rewrite Equation 11 as:

$$S(\vec{x}, t) = \left(-\vec{b}_t + \sum_{\forall i} \alpha_{i,t} (\vec{\phi} * \vec{\phi} * \vec{v}_i) \right) \cdot \vec{x} \tag{13}$$

From this definition, it is easy to see that the SVM model for each topic reduces to a single projection vector. Furthermore, the collection of SVM trained vectors can be stacked to form an $N_T \times N_V$ dimension routing matrix R whose elements are:

$$r_{t,w} = -b_t + \sum_{\forall i} \alpha_{i,t} \phi_w^2 v_{i,w} \tag{14}$$

Here $v_{i,w}$ is the relative frequency of word w within vector $\vec{v}_i$.

3. MCE-Based Feature Weighting

3.1. MCE Algorithm

In our previous work [3], we have introduced an MCE training approach for learning weighting factors for each feature contained in the feature vector $\vec{x}$. To achieve this, we incorporate a new feature weighting vector $\vec{\lambda}$ into Equation 3 as follows:

$$\vec{s} = \mathbf{R}(\vec{\lambda} * \vec{x}) \tag{15}$$

Thus, each individual element λ_w in $\vec{\lambda}$ serves to rescale the contribution of the individual feature x_w.

We use a standard form of MCE training [6], where we begin by defining a misclassification measure:

$$M(\vec{x}) = F(\vec{x}, \bar{t}_C) - S(\vec{x}, t_C) \tag{16}$$

Here, $S(\vec{x}, t_C)$ represents the classifier score for the correct topic t_C and $F(\vec{x}, \bar{t}_C)$ is a function of the scores of all of the incorrect topics defined as follows:

$$F(\vec{x}, \bar{t}_C) = \frac{1}{\eta} \log \left[\frac{1}{N_T - 1} \sum_{\forall t \neq t_C} \exp(\eta S(\vec{x}, t)) \right] \tag{17}$$

In this expression, all competing hypotheses contribute to the misclassification measure with the highest scoring competitors contributing the most.

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

$$\ell(\vec{x}) = \frac{1}{1 + \exp(-\beta M(\vec{x}))} \tag{18}$$

Here, β represents the slope of the sigmoid function. The loss function will be close to zero for documents with large negative values of $M(\vec{x})$ and close to one for documents with large positive values of $M(\vec{x})$. This loss function can be differentiated with respect to the individual features weights and optimized via an iterative gradient descent algorithm. The partial derivative of the loss function $\ell(\vec{x})$ with respect to a specific feature weight λ_w is:

$$\frac{\partial \ell(\vec{x})}{\partial \lambda_w} = \beta \ell(\vec{x}) \left(1 - \ell(\vec{x})\right) \left(-r_{t_C,w} + \sum_{\forall t \neq t_C} \gamma_t r_{t,w}\right) x_w \tag{19}$$

Here, the γ_t's are posterior-like weights over the incorrect topics as defined by:

$$\gamma_t = \frac{\exp(\eta S(\vec{x}, t))}{\sum_{\forall t_i \neq t_C} \exp(\eta S(\vec{x}, t_i))} \tag{20}$$

In this expression, we can interpret the variable η as a posterior scaling factor. As $\eta \rightarrow \infty$ then $\gamma_{t_I} \rightarrow 1$ for the best scoring incorrect topic t_I, and $\gamma_t \rightarrow 0$ for all other topics.

In our system, the learning algorithm performs sequential updating as the trainer sweeps through the training vectors, i.e., the weights are updated immediately after each training vector is presented to the algorithm. The form of this update is:

$$\lambda'_w = \lambda_w - \epsilon \frac{\partial \ell(\vec{x})}{\partial \lambda_w} \tag{21}$$

Here ϵ is a learning rate parameter. We do not allow any λ_w to go negative in our experiments, and we further constrain $\vec{\lambda}$ by regularizing it as follows:

$$\sum_{\forall w} \lambda_w = N_V \tag{22}$$

Ideally, when computing the partial derivatives of $\ell(\vec{x})$, the routing matrix used in the calculation should be trained from data which excludes $\vec{x}$, thus allowing $\vec{x}$ to appear as unseen data in the eyes of the trainer. This can be accomplished via a *jackknifing* process over the training data. In our experiments, the topic ID training data was subdivided into ten partitions, and each vector $\vec{x}$ was scored using an R matrix that was trained from only the nine partitions of the data that did not include $\vec{x}$.

3.2. Hybrid SVM/MCE Training

When performing MCE training of the vector $\vec{\lambda}$, the SVM routing matrix R remains fixed. However, after the MCE training converges (or is otherwise halted), SVM training can be reapplied using an updated term weighting vector $\vec{\phi}$ in its linear kernel function. After a full run of MCE training, each individual term ϕ_w inside of $\vec{\phi}$ can be updated using this expression:

$$\phi'_w = \sqrt{\lambda_w} * \phi_w \tag{23}$$

Following the update, the SVM training is re-executed to learn a new set of support vector weights given the newly updated kernel function. The hope is that the improved weights learned from the MCE procedure will yield further improved performance from the SVM retraining. However, there is no guarantee that improvements will ensue as the MCE training objective (i.e. minimum average loss) is quite different from the SVM training objective function (i.e. maximum margin).

4. Experimental Results

4.1. Data Set

For the data set for our experiments we have used the English Phase 1 portion of the Fisher Corpus [7]. This corpus consists of 5851 recorded telephone conversations. During data collection, two people were connected over the telephone network and given instructions to discuss a specific topic for 10 minutes. Data was collected from a set of 40 different topics. Fixed prompts designed to elicit discussion on the topics were played to the participants at the start of each call. For our experiments the corpus was subdivided into four subsets:

1. Recognizer training set (3104 calls; 553 hours)
2. Topic ID training set (1375 calls 244 hours)
3. Topic ID development test set (686 calls; 112 hrs)
4. Topic ID evaluation test set (686 calls; 114 hrs)

We perform closed-set topic ID over the 40 topics in the Fisher corpus. We evaluate our approach on both the original text transcripts of the data as well as on the outputs of an automatic speech recognition system. The topic classifier is trained on the topic ID training set. The development set is used for optimizing training parameters and selecting training stopping criteria. Final results are reported on the evaluation test set.

4.2. 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 from both sides of the call.

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 specified above 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. Because this recognizer applies very basic modeling techniques with no adaptation, the system performs recognition faster than real time (on a current workstation) but top-choice word error rates can be high (typically over 40%).

4.3. SVM/MCE Training Details

In our experiments we use the SVMTorch software package for training of the SVM α and b values [2]. An initial set of 40 topic SVM models are trained over the full training set and merged into a single routing matrix. The same process is used to created 10 different SVM routing matrices for each jack-knifed partitioning of the training data used for MCE training. It is worth noting that our training data in all experiments is fully separable, i.e. the SVM error rate over the full training set is 0%. The feature weights λ_w are initially all set to a value of one, and the MCE training is then applied over the training data using the

Training Iterations	Topic Error Rate (%)	
	Initial SVM	SVM+MCE
0	14.1	9.6
1	11.5	8.5
2	10.6	8.2
3	10.1	8.3
4	9.8	8.6
5	9.9	8.6
6	10.0	8.9

Table 1: Closed-set topic ID error rate using the outputs from the ASR system on the development test set. Error rates are shown over multiple iterations of SVM/MCE hybrid training.

sequential training algorithm described earlier. We use test runs on the development test set to determine appropriate settings for the training parameters (β, η and ϵ) as well as the appropriate number of MCE passes to run per SVM/MCE iteration, and total number of iterations of SVM/MCE retraining to perform.

Table 1 shows the performance on the development test set when using the ASR outputs for the settings of $\beta = 10.0$, $\eta = 1000$, and $\epsilon = 0.1$. For these results 10 MCE passes through the data were run after each SVM retraining. Thus, the initial SVM error rate of 14.1% is improved to 9.6% after 10 MCE passes. The weights learned from the MCE process are then folded back into the SVM term weighting vector and the SVM is retrained. This yields the new SVM error rate of 11.5% which is reduced by MCE training to 8.5%, and so forth. Optimal performance on the development test set was achieved after MCE training of the second SVM retraining. Further iterations of the SVM/MCE hybrid training harm performance on the development set indicating over-tuning to the training set.

4.4. Back-End Calibration

In the fields of language ID and speaker ID, it has become standard practice to apply a back-end (BE) classification stage to the output scores produced by an initial classifier. The BE classifier is intended to *calibrate* the classifier scores, i.e. to compensate for estimation errors or biases inherent in the initial model. Our system uses a regression-based classifier from the FoCal toolkit [9]. The BE classifier takes a feature vector of 40 topic scores as its input, and outputs a set of 40 calibrated topic scores. It is trained using the held-out development test set.

4.5. Results

Table 2 shows the performance of our approach on the final test set under various conditions. The two columns represent the system performance when using the original text transcripts vs. when using the ASR lattice outputs. The top four rows show the performance of the SVM system both before and after the hybrid SVM/MCE training and when either using or not-using the back-end (BE) classifier. For comparison, the bottom four rows present results using the naive Bayes (NB) system discussed in our earlier work [3] (the results here are slightly different due to minor changes in the systems' training parameters).

The table shows that the MCE training algorithm produces significant improvements in both the SVM and naive Bayes systems. When no back-end classifier is used, the MCE algorithm produces relative error rate reductions between 26% and 52% over the various SVM and naive Bayes systems. Also, the use of back-end calibration is particularly helpful at improving the SVM performance when MCE training is not employed, producing error rate reductions of 35% in the text case and 26% in the ASR case. After MCE training, the back-end classifier only reduces the error rate of the SVM text-based system by 5% and the SVM ASR-based system by 7%. A similar trend is observed in the naive Bayes case. It can be argued that the MCE feature weight training produces scores with better calibration, though this is a topic that needs further investigation.

Classifier	Topic Error Rate (%)	
	Text	ASR
Initial SVM	10.9	11.2
Initial SVM + BE	7.1	8.3
SVM + MCE	5.4	8.3
SVM + MCE + BE	5.1	7.7
Initial NB	9.8	14.7
Initial NB + BE	9.3	13.1
NB + MCE	4.7	7.9
NB + MCE + BE	5.4	7.8

Table 2: Closed-set topic ID error rate for naive Bayes (NB) and SVM classifiers on the final test set under various conditions.

5. Summary

In this paper we have presented a new hybrid SVM/MCE approach for training SVM-based topic ID systems. We have applied this approach to topic ID for human-human telephone conversations using both text transcripts and ASR output. The new algorithm has significantly reduced the error rate of our SVM-based systems, bringing them on par with our pre-existing naive Bayes systems which are similarly trained with MCE.

6. References

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

[2] R. Collobert and S. Bengio, "SVMTorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143-160, 2001.

[3] 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.

[4] H.-K. J. Kuo, and C.-H. Lee, "Discriminative training of natural language call routers," in *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 1, pp. 24–35, Jan. 2003.

[5] W. Campbell, J. Campbell, D. Reynolds, D. Jones, and T. Leek. "Phonetic speaker recognition with support vectors machines," in *Proc. Neural Information Processing Systems Conf.*, Vancouver, pp. 1377–1384, Dec. 2003.

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

[7] 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.

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

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