COMBINING FEATURE SETS WITH SUPPORT VECTOR MACHINES: APPLICATION TO SPEAKER RECOGNITION

Andrew O. Hatch[1,2]*, Andreas Stolcke*[1,3]*, and Barbara Peskin*[1]

[1]The International Computer Science Institute, Berkeley, CA, USA
[2]The University of California at Berkeley, USA
[3]SRI International, Menlo Park, CA, USA
{ahatch,barbara}@icsi.berkeley.edu, stolcke@speech.sri.com

ABSTRACT

In this paper, we describe a general technique for optimizing the relative weights of feature sets in a support vector machine (SVM) and show how it can be applied to the field of speaker recognition. Our training procedure uses an objective function that maps the relative weights of the feature sets directly to a classification metric (e.g. equal-error rate (EER)) measured on a set of training data. The objective function is optimized in an iterative fashion with respect to both the feature weights and the SVM parameters (i.e. the support vector weights and the bias values). In this paper, we use this procedure to optimize the relative weights of various subsets of features in two SVM-based speaker recognition systems: a system that uses transform coefficients obtained from maximum likelihood linear regression (MLLR) as features [1] and another that uses relative frequencies of phone n-grams [2, 3]. In all cases, the training procedure yields significant improvements in both EER and minimum DCF (i.e. decision cost function), as measured on various test corpora.

1. INTRODUCTION

In recent years, the field of speaker recognition has benefitted significantly from the use of SVMs. For example, in 2001, Campbell et al. demonstrated an SVM-based analog to the traditional approach of modeling cepstral features with Gaussian mixture models (GMMs) [4]. This approach has been widely adopted and currently forms one of the central components in a number of state-of-the-art speaker recognition systems. Other SVM-based approaches include work on modeling prosodic features [5]. In 2003, Campbell et al. introduced an SVM-based approach for using phone n-grams to train speaker models [2], which was subsequently extended in [3]. A similar approach has also been applied to word n-grams [6]. More recently, Stolcke et al. used MLLR-based features in an SVM to perform speaker recognition [1].

One issue common to each of these techniques is the question of how to design effective kernels for different types of feature sets (e.g. phone n-grams, cepstral features, etc.). A number of feature-specific kernels have been proposed in the literature. For example, Campbell et al. introduced a kernel for count-based features in [2], which tends to work well on SVM-based phone n-grams [2, 3]. However, for other, more abstract feature sets like the MLLR coefficients used in [1], choosing an appropriate kernel is far less straightforward. Similarly, the problem of incorporating or "combining" different features (or entire feature sets) in a single SVM can also pose a significant challenge.

In this paper, we focus on training linear kernels for two SVM-based systems: a system that uses MLLR-based features [1] and another that uses relative frequencies of phone n-grams [2, 3]. For the experiments in this paper, the MLLR features and the phone n-gram features are both divided into a small number of subsets. We then train relative weights for each subset of features and use the ensemble of weighted features to train SVM-based speaker models. We use an iterative approach to train the relative weights for the feature subsets, where we attempt to minimize the EER obtained on some training set.

This material is based upon work supported by the National Science Foundation under grant No. 0329258

2. KERNEL TRAINING

Given a kernel function, $K_\Theta(\cdot,\cdot)$, where $\Theta = \{\mu_1, \ldots, \mu_L\}$ is some set of kernel parameters, we would like to train Θ to minimize the EER that we obtain when $K_\Theta(\cdot,\cdot)$ is used to train SVM-based speaker models on a particular task and feature set. We use an iterative approach to train $\mathcal{A}$, $\mathcal{B}$, and Θ, where $\mathcal{A}$ and $\mathcal{B}$ represent the set of support vector weights and bias values (i.e. "alpha" and "beta" values) for a set of speaker models. Given some initial $\mathcal{A}$, we train Θ and $\mathcal{B}$ using an objective function that minimizes EER computed on some set of training data. Then we use the optimized Θ in $K_\Theta(\cdot,\cdot)$ to train new values for $\mathcal{A}$ and $\mathcal{B}$ (i.e. we train SVM-based speaker models). This process can be repeated until no further improvements in EER are achieved. The final, trained kernel function is then used for testing.

In the following subsections, we provide a set of definitions followed by a derivation of the objective function that we used to train Θ for a given $\mathcal{A}$. We also describe how $\mathcal{A}$ is initialized and briefly outline an approach for optimizing the objective function.

2.1. Definitions

Let $\mathcal{T} = \{T_1, \ldots, T_M\}$ be a set of M target speaker models, and let S_i be the corresponding speaker for model T_i. For every T_i, let $X_{i,\mathcal{A}}$ and $X_{i,\Theta}$ be data sets for training $\mathcal{A}$ and K_Θ, respectively. Both data sets are composed of positive and negative examples (i.e. feature vectors) for T_i. Let

$$y_i(x) = \begin{cases} 1, & x \in S_i \\ -1, & \text{otherwise.} \end{cases}$$

We define $f_i(x;\Theta,\mathcal{A},\mathcal{B})$ to be the SVM-based scoring function for speaker model T_i using kernel $K_\Theta(\cdot,\cdot)$:

$$f_i(x;\Theta,\mathcal{A},\mathcal{B}) = b_i \; + \sum_{x_{tr}\in X_{i,\mathcal{A}}} \alpha_i(x_{tr}) y_i(x_{tr}) K_\Theta(x_{tr},x), \\ 0 \le \alpha_i(x_{tr}) \le C \;\; \forall \;\; x_{tr} \in X_{i,\mathcal{A}} \quad (1)$$

In the above equation, b_i represents the bias term for the SVM, and $\alpha_i(x_{tr})$ represents the "alpha" weight applied to feature vector x_{tr}. The hyperparameter, C, represents the tradeoff between the regression and margin-maximization components in an SVM. As was mentioned earlier, we use $\mathcal{A}$ to represent the set of $\alpha_i(x_{tr})$ terms for all $x_{tr} \in X_{i,\mathcal{A}}$ and for all $i \in \{1,\ldots,M\}$. Similarly, $\mathcal{B} = \{b_1,\ldots,b_M\}$ represents the set of bias values for the speaker models. Note that equation (1) provides the general form for the scoring function in an SVM.

We use $J_i^+(\Theta,\mathcal{A},\mathcal{B})$ and $J_i^-(\Theta,\mathcal{A},\mathcal{B})$ to represent the classification error obtained on speaker model T_i for positive examples and for negative examples, respectively. These are defined as:

$$J_i^+(\Theta,\mathcal{A},\mathcal{B}) = \frac{1}{N_i^+} \sum_{\substack{x\in X_{i,\Theta} \\ s.t. \;\; x\in S_i}} \mathbf{1}\Big(f_i(x;\Theta,\mathcal{A},\mathcal{B}) < 0\Big),$$

$$J_i^-(\Theta,\mathcal{A},\mathcal{B}) = \frac{1}{N_i^-} \sum_{\substack{x\in X_{i,\Theta} \\ s.t. \;\; x\notin S_i}} \mathbf{1}\Big(f_i(x;\Theta,\mathcal{A},\mathcal{B}) > 0\Big),$$

where $\mathbf{1}(x)$ denotes the indicator function of x and where

$$N_i^+ = \sum_{x\in X_{i,\Theta}} \mathbf{1}(x \in S_i),$$

$$N_i^- = \sum_{x\in X_{i,\Theta}} \mathbf{1}(x \notin S_i).$$

In the above equations, the N_i^+ and N_i^- terms normalize the classification error for every positive (negative) trial by the total number of positive (negative) trials in $X_{i,\Theta}$.

2.2. Objective Function for Training Θ

Our approach for optimizing Θ is to minimize the EER computed on the training data given the current parameterization of $\mathcal{A}$ (i.e. the $\alpha_i(x)$ terms). We use a *model-normalized EER*, where the total error contributed by speaker model T_i is normalized by the number of trials in $X_{i,\Theta}$, and all models are assumed to have equal priors. This leads to the following optimization problem:

$$\min_{\Theta,\mathcal{B}} \frac{1}{2M} \sum_{i=1}^{M} \Big(J_i^+(\Theta,\mathcal{A},\mathcal{B}) + J_i^-(\Theta,\mathcal{A},\mathcal{B})\Big) \\ s.t. \;\; \sum_{i=1}^{M} J_i^+(\Theta,\mathcal{A},\mathcal{B}) = \sum_{i=1}^{M} J_i^-(\Theta,\mathcal{A},\mathcal{B}) \quad (2)$$

Note that in (2), we not only optimize over Θ, but also over $\mathcal{B}$ (i.e. we assume that the $\mathcal{B}$ terms are free parameters). For the experiments in this paper, we used the following modified version of the above optimization problem:

$$\min_{\Theta,\mathcal{B}} \frac{1}{2M} \sum_{i=1}^{M} \Big(J_i^+(\Theta,\mathcal{A},\mathcal{B}) + J_i^-(\Theta,\mathcal{A},\mathcal{B})\Big) \\ s.t. \;\; J_i^+(\Theta,\mathcal{A},\mathcal{B}) = J_i^-(\Theta,\mathcal{A},\mathcal{B}) \;\; \forall\, i \quad (3)$$

Here, instead of minimizing the model-normalized EER computed over the set of *all* speaker trials, we minimize the average of the individual *per-model* EERs. Since the latter minimization is simply a more constrained version of the original problem in (2), the minimization in (3) provides an upper bound on (2). We also note that (3) is generally easier to evaluate than (2). To see this, we can rewrite (2) as

$$\min_{\Theta,\mathcal{B}} \; EER\Big(\bigcup_{i=1}^{M} \bigcup_{x\in X_{i,\Theta}} \Big(f_i(x;\Theta,\mathcal{A},\mathcal{B}) \;,\; y_i(x) \;,\; i\Big)\Big), \quad (4)$$

and (3) as

$$\min_{\Theta} \frac{1}{M} \sum_{i=1}^{M} EER\Big(\bigcup_{x\in X_{i,\Theta}} \Big(f_i(x;\Theta,\mathcal{A},\mathcal{B}) \;,\; y_i(x) \;,\; i\Big)\Big). \quad (5)$$

In the above equations, $EER(S)$ represents a function that computes the *model-normalized* EER over some set S of score/label pairs, where each pair is annotated with model ID i to allow for model-normalization. Note that (4) requires us to compute the model-normalized EER over *all* speaker trials while simultaneously minimizing with respect to $\mathcal{B}$. On the other hand, in (5), the minimization over $\mathcal{B}$ is performed implicitly by computing EERs on a per-model basis. As a possible alternative to the objective function in (5), (one which we have opted not to use in this report), we note that $\mathcal{B}$ could also be kept fixed at every training iteration, in which case our objective function would take the form of (4), except that we only minimize over Θ.

2.2.1. Training Relative Weights for Feature Subsets

We are interested in training linear kernels for combining various subsets of features (i.e. we want to optimize the relative weight that we apply to each feature subset prior to training SVMs). For the experiments in this paper, we assume that the feature space is divided into L subsets. Thus, the kernel function, $K_\Theta(\cdot,\cdot)$, takes the following form

$$K_\Theta(x_{tr},x) = \sum_{\ell=1}^{L} \mu_\ell^2 \langle x_{tr,\ell} \;,\; x_\ell \rangle, \quad (6)$$

where μ_ℓ represents the weight assigned to feature subset ℓ. The feature vectors, x_{tr} and x, are defined as

$$x_{tr} = \Big[x_{tr,1}^T \;,\; x_{tr,2}^T \;,\; \cdots \;,\; x_{tr,L}^T\Big]^T,$$

$$x = \Big[x_1^T \;,\; x_2^T \;,\; \cdots \;,\; x_L^T\Big]^T.$$

Here, $x_{tr,\ell}$ and x_ℓ are the ℓth subsets of feature vectors x_{tr} and x, repectively.

2.3. Initializing $\mathcal{A}$

For the experiments in this paper, we initialized the $\alpha_i(x)$ values so that the SVM weight vector w_i for target model T_i is simply the difference between the average of the positive training examples and the average of the negative training examples for T_i. This is achieved by the following assignment:

$$\alpha_i(x_t) = \begin{cases} \frac{1}{\sum_{x\in X_{i,\mathcal{A}}} 1(x\in S_i)}, & x_t \in S_i \\ \frac{1}{\sum_{x\in X_{i,\mathcal{A}}} 1(x\notin S_i)}, & x_t \notin S_i. \end{cases}$$

2.4. Optimizing the Objective Function

The objective function for training Θ given the current parameterization of $\mathcal{A}$ is defined by plugging equations (1) and (6) into the optimization problem in (5). For the experiments in this paper, we optimized (5) with respect to Θ by performing a simple search over $[0,1]^L$. Our procedure for optimizing (5) is essentially an iterated grid search, where the search range is narrowed after every iteration. Since this paper is primarily focused on the problem of deriving an objective function for Θ (as opposed to a discussion of how to solve it), we will forego a detailed discussion on optimization. Instead, we focus in the following sections on describing the actual experiments that we performed.

3. EXPERIMENT I: PHONE N-GRAM SYSTEM

For our first experiment, we trained relative weights to combine feature subsets in an SVM-based phone n-gram system similar to the one described in [3]. This system uses relative frequencies of phone n-grams derived from an open-loop, lattice phone decoding as a basis for building feature vectors.

We used the following linear kernel developed by Campbell et al. in [2] to assign scaling factors to each relative frequency:

$$K(A,B) = \sum_{i=1}^{N} \frac{p(d_i|convSide_A)}{\sqrt{p(d_i|bkg)}} \frac{p(d_i|convSide_B)}{\sqrt{p(d_i|bkg)}}$$

Here, $p(d_i|convSide_A)$ and $p(d_i|bkg)$ represent the relative frequency of phone n-gram d_i within conversation side A and within the data for the background model, respectively. The above kernel simply normalizes each relative frequency by the square-root of the corresponding relative frequency in the background model. The resulting *scaled* relative frequencies are then used as features.

We used scaled relative frequencies based on various orders of phone n-grams to define 5 subsets of features. These included unigrams, bigrams, and the top 15000 trigrams, which were split into 3 subsets of 5000 trigrams each. To perform lattice phone decoding, we used the DECIPHER speech recognition system developed by SRI International. Our particular realization of DECIPHER uses a vocabulary of 46 phone units. Thus, the total size of the unigram and bigram subsets is 46 and $46^2 = 2116$, respectively. These subsets, combined with the 3 subsets of trigrams add up to a total of 17162 features.

3.1. Task and Data

For the phone n-gram system, we performed experiments on the 1-conversation training condition of the NIST 2003 Extended Data Task (a.k.a. SRE-03), which uses phases II and III of the Switchboard-2 corpus. Phases II and III consist of approximately 14000 conversation sides, which are split into 10 disjoint speaker sets. For our experiments, we used splits 1 through 5 to train a background model for splits 6 through 10 and vice versa. Further details on the phone n-gram system can be found in [3].

3.2. Training $\mathcal{A}$ and Θ

We used splits 9 and 10 as "foreground data" to define a set of *auxiliary* speaker models, $\mathcal{T} = \{T_1, \ldots, T_M\}$, which we used to train Θ for testing on splits 1 through 5. Splits 6 through 8 were used as the corresponding "background data" (i.e. the negative training examples in $X_{i,\mathcal{A}}$ and $X_{i,\Theta}$) for each of the auxiliary speaker models. Similarly, we used splits 4 and 5 as foreground data and splits 1 through 3 as background data to train Θ for testing on splits 6 through 10. We defined one auxiliary speaker model for every conversation side in the foreground data. Given a particular conversation side x_i which defines auxiliary speaker model T_i, x_i is used as the only positive training example in $X_{i,\mathcal{A}}$. The remaining conversation sides of speaker S_i (excluding x_i) were used as the positive trials in $X_{i,\Theta}$. We used the SVM^{light} package for all SVM training and scoring [7].

weights	SRE-03 EER%	SRE-03 DCF
uniform (baseline)	7.24	0.0291
iteration 1	7.60	0.0317
iteration 2	7.29	0.0299
iteration 3	**7.06**	**0.0287**
relative improvement	**2.5%**	**1.4%**

Table 1. EERs and minimum DCFs for the phone n-gram system on SRE-03

unigram	bigram	trigram-1	trigram-2	trigram-3
0.000	0.905	0.849	1.000	0.931

Table 2. final relative feature weights for each of 5 phone n-gram classes trained on SRE-03, splits 6 through 10

3.3. Results

Table 1 shows EERs and minimum DCF scores for our baseline phone n-gram system, where the relative weights are uniform, and for the first three iterations of kernel training. Note that neither the results nor the feature weights change significantly after the third iteration. As shown in table 1, the iterative training yields a relative improvement in EER of 2.5% over the baseline. The final feature weights for testing on splits 1 through 5 are shown in table 2 (note that these weights are virtually identical to those obtained for testing on splits 6 through 10). In table 2, we see that the final, trained relative weights are fairly uniform for all feature subsets except the phone unigrams, which have a weight of zero.

4. EXPERIMENT II: MLLR-SVM SYSTEM

For our second experiment, we trained a linear kernel to combine feature subsets for an MLLR-SVM system similar to the one described in [1]. The MLLR-SVM system uses speaker adaptation transforms from SRI's DECIPHER speech recognition system as features for speaker recognition. A total of 8 affine transforms are used to map the Gaussian mean vectors from speaker-independent to speaker-dependent speech models. The transforms are estimated using maximum-likelihood linear regression (MLLR), and can be viewed as a text-independent encapsulation of the speaker's acoustic properties. For every conversation side, we compute a total of 12480 transform coefficients, which are normalized to have unit variance and then used as features. The transform coefficients can be divided into 8 subsets of 1560 features each, where each subset corresponds to one of the following phone classes: *voiced stops*, *unvoiced stops*, *voiced fricatives*, *unvoiced fricatives*,

nasals, *retroflex phones* (e.g. /r/ and /er/), *low vowels* (e.g. /aa/, /ae/, /ah/) and *high vowels* (e.g. /eh/, /ey/, /ih/). Further information on the MLLR-SVM system can be found in [1].

4.1. Task and Data

For our experiments, we used a version of the MLLR-SVM system used by SRI International in the NIST 2005 Speaker Recognition Evaluation [1]. The system uses 3 disjoint sets for training: D_A, D_B, and a background set, D_{bkg}. Set D_A is composed of 3642 conversation sides taken from the Switchboard-2 data of 319 speakers. Set D_B is similarly composed, but uses a different set of Switchboard-2 speakers. Set D_{bkg} consists of 425 conversation sides taken from Switchboard-2 and 1128 conversation sides taken from the Fisher corpus. The Switchboard-2 conversation sides in D_{bkg} are selected so that no two conversation sides belong to the same speaker. Note that no speaker has data that appears in more than one set. Thus, the three sets can alternately be used for training and testing (i.e. jackknifing).

For testing on Switchboard-2 data, we defined a set of test-target speaker trials for both D_A and D_B. We used half of the conversation sides from every speaker in D_A and D_B to define separate 1-conversation target models. The target models were tested against all other conversation sides belonging to the same speaker (i.e. the true trials) and against one randomly chosen conversation side from *half* of the impostor speakers in the given set (i.e. the negative trials). Combining the trials from D_A and D_B gives us a total of 50769 positive and 609139 negative Switchboard-2 speaker trials. To evaluate the cross-task effectiveness of our kernel training algorithm, we also tested on the SRE-04 task.

4.2. Training $\mathcal{A}$ and Θ

We used D_{bkg} to provide the negative examples for training $\mathcal{A}$ for a given target speaker model, T_i. The speaker trials for training Θ on D_A and D_B were defined in the same way as the test speaker trials described above, except that we used the Switchboard-2 speakers in D_{bkg} to provide all of the negative examples. Thus, the speaker trials for training Θ were composed only of Switchboard-2 data. In all cases, training and testing were performed on disjoint speaker sets, so all results are fair (e.g. we used the training trials defined from D_B and D_{bkg} to train Θ for testing on D_A). As with the phone n-gram system, all SVM training and scoring was done with the SVM^{light} package [7].

4.3. Results

Table 3 shows EERs and minimum DCF scores for our baseline MLLR-SVM system, where the relative weights are uniform, and for the first three iterations of kernel training. Subsequent iterations achieved no significant improvements. As shown in table 3, the iterative training yields relative reductions in EER and minimum DCF of 4.6% and iterative training also achieves a 4.3% reduction in EER on SRE-04 (the same feature weights that we used to test on D_A were also used to test on SRE-04). Note that the EER reduction on SRE-04 represents an "out-of-task" result, since no SRE-04 data was used to train Θ. The final, relative feature weights (i.e. the μ values) for testing on D_A are shown in table 4 for each of the phone classes listed in section 4. Table 4 shows that the *low vowels* and *high vowels* are assigned the largest relative weights among the eight phone classes, while the *voiced stops* are assigned the smallest. The relative importance of vowels for speaker recognition is consistent with previous findings in the linguistics literature [8].

weights	Switchboard-2 data		SRE-04	
	EER%	DCF	EER%	DCF
uniform (baseline)	4.75	0.0175	9.84	0.0347
iteration 1	4.58	0.0166	9.42	0.0336
iteration 2	4.53	0.0163	9.42	0.0345
iteration 3	**4.53**	**0.0164**	**9.42**	**0.0342**
relative improvement	**4.6%**	**6.3%**	**4.3%**	**1.4%**

Table 3. EERs and minimum DCFs for the MLLR-SVM system

VS	UVS	VF	UVF	N	RP	LV	HV
0.407	0.668	0.714	0.619	0.709	0.610	1.000	0.926

Table 4. final relative feature weights for each of 8 MLLR transformation classes trained on D_B and D_{bkg}

5. DISCUSSION AND FUTURE WORK

Although the improvements described in this report are relatively modest, we consider the results to be quite encouraging—particularly given the very small number of parameters that were used (only 5 parameters for the phone n-grams and 8 for the MLLR-SVM). It's also worth noting that the baseline MLLR-SVM system is one of the best-performing systems for 1-conversation training that we are aware of. In light of this, we consider even the modest improvements obtained on the MLLR-SVM system to be highly encouraging.

One obvious extension to this work, which we plan to investigate, is the large-scale combination of multiple SVM-based systems—for example, the combination of cepstral, MLLR, phone n-gram, and prosodic features—within a single SVM. We would also like to experiment with gradient-based objective functions for training kernels (e.g. replacing the EER term in equation (5) with a "hinge-loss" function, which is differentiable). This would greatly improve the efficiency of the kernel training, and would allow us to train more weights. One particularly intriguing approach to gradient-based kernel optimization, which employs a hinge-loss function along with a margin-maximization component to prevent overfitting, is described by Lanckriet et al. in [9].

6. CONCLUSION

In this paper, we describe a general technique for optimizing the relative weights of feature sets in a support vector machine (SVM), and demonstrate its application to the field of speaker recognition. Our approach yields relatively small improvements in EER and DCF when we use it to optimize relative weights for a small number of n-gram classes in a phone n-gram system. When applied to a state-of-the-art MLLR-SVM system on Switchboard-2 data for a 1-conversation training task, our approach achieves improvements in EER and DCF of 4.6% and 6.3%.

7. REFERENCES

[1] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR Transforms as Features in Speaker Recognition," http://www.speech.sri.com/cgi-bin/run-distill?papers/eurospeech2005-mllr-spkr.ps.gz, to appear in Proc. of Interspeech, 2005.

[2] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "Phonetic speaker recognition with support vector machines," in *Advances in Neural Information Processing Systems 16*, 2003.

[3] A. Hatch, B. Peskin, and A. Stolcke, "Improved Phonetic Speaker Recognition Using Lattice Decoding," in *Proc. of ICASSP*, 2005.

[4] W. M. Campbell, "A Sequence Kernel and its Application to Speaker Recognition," in *Advances in Neural Information Processing Systems 14*, 2001.

[5] E. Shriberg, L. Ferrer, A. Venkataraman, and S. Kajarekar, "SVM Modeling of "SNERF-grams" for Speaker Recognition," in *Proc. of ICSLP*, 2004.

[6] S. Kajarekar et al., "SRI's 2004 NIST Speaker Recognition Evaluation System," in *Proc. of ICASSP*, 2005.

[7] T. Joachims, "Making large-scale SVM learning practical," in *Advances in kernel methods — support vector learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT-press, 1999.

[8] W. Labov, *Principles of Linguistic Change: Volume1, Internal Factors*, Blackwell, 1994.

[9] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan, "Learning the Kernel Matrix with Semidefinite Programming," in *Journal of Machine Learning Research*, 2004.