A Hybrid SVM/HMM Acoustic Modeling Approach to Automatic Speech Recognition

J. Stadermann, G. Rigoll

Institute for Human-Machine Communication
Technische Universität München
Arcisstrasse 21, 80290 Munich, Germany
Phone:+49-89-289-{28319, 28541},
Email: {stadermann, rigoll}@mmk.ei.tum.de

Abstract

Acoustic models based on a NN/HMM framework have been used successfully on various recognition tasks for continuous speech recognition. Recently tied-posteriors have been introduced within this context. Here, we present an approach combining SVMs and HMMs using the tied-posteriors idea. One set of SVMs calculates class posterior probabilities and shares these probabilities among all HMMs. The number of SVMs is varied as well as the input context and the amount of training data. Applying a first implementation, results on the AURORA2 task show already a promising improvement of the word error rate compared to the baseline acoustic models.

1. Introduction

Today's acoustic models for automatic speech recognition are mostly based on *hidden Markov models* (HMMs) using Gaussian probability densities trained with a maximum likelihood criterion. However, there is also a long history of incorporating *neural networks* (NN) in the acoustic models because of their discriminative nature. One successful approach [1] applies the NN to estimate static phoneme posterior probabilities from a frame with fixed length and models the speech signal's time-dependencies with HMMs. An extension to this framework is shown in [2] where the posteriors are tied for all HMMs and the network topology as well as the HMM topology are varied.

Support vector machines (SVM) are another class of powerful algorithms derived from statistical learning theory and applicable to pattern recognition problems. Since SVMs also lack the ability to model time series (like NNs described above) the combination with HMMs is again an interesting solution. One implementation of a SVM/HMM system used for speech recognition is presented in [3] where the SVMs are trained on segment-level data with one-state HMMs. These segments generally occupy a varying number of frames so some sampling is necessary to generate a fixed-length input vector for the SVMs. The approach described in the following sections circumvents the problem of creating sampled segment-level data, but builds-up on the NN/HMM tied-posterior framework [4], thus allowing arbitrary HMM topologies. Furthermore, no additional HMM decoder with Gaussian models is necessary for the recognition phase (like it is in [3]), since our system does not need any external information about segments for decoding.

The remaining paper is organized as follows: Section 2 outlines the concept of SVMs, section 3 deals with the combination of SVMs and HMMs, section 4 presents details about our implementation of a SVM/HMM system, section 5 shows our results on the AURORA2 database and section 6 concludes the paper.

2. Support Vector Machines

In the first place the SVM tries to find an optimal classifier for a linear separable problem by *structural risk minimization* (SRM) [5]. SRM aims at finding an upper bound of the classifier's expected risk rather than minimizing the empirical risk that is just the expectation value of training errors. Figure 1 illustrates the difference: Minimizing the empirical risk could result in hyperplane A or B (or some other separating hyperplane) since each of them produces no training errors, but only SRM produces the bold hyperplane B that maximizes the margin between the two classes. Starting with N training samples $(i = 1, \ldots, N)$ with class labels y_i (either $+1$ or -1) the following equations hold:

$$\vec{x}_i \cdot \vec{w} + b \geq +1 \text{ for } y_i = +1 \quad (1)$$

$$\vec{x}_i \cdot \vec{w} + b \leq -1 \text{ for } y_i = -1 \quad (2)$$

Using Lagrangian multipliers α_i, maximizing the margin is a quadratic optimization problem that results in only a few training samples (called support vectors) affecting the hyperplane. Once the SVM is trained the classification can be done according to eq. 3 - determining the class by the sign of $y(t)$.

$$y(t) = \vec{x}(t) \cdot \vec{w} + b \text{ with } \vec{w} = \sum_{i=1}^{N_S} \alpha_i \hat{y}_i \vec{s_i} \qquad (3)$$

with N_S the number of support vectors $\vec{s_i}$ and $\hat{y}_i$ their class labels.

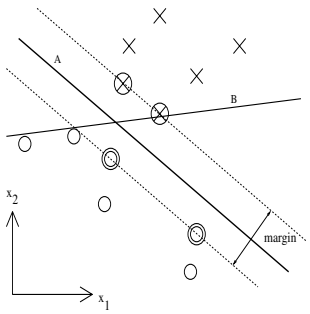

Figure 1: Linear separable problem with two classes and separating hyperplanes - the support vectors are circled

Since most of the real-world problems are not linearly separable (including phoneme or word classes), an extension to the algorithm is required. By introducing slack variables ξ_i the classification problem can be formulated as follows

$$\vec{x_i} \cdot \vec{w} + b \geq +1 - \xi_i \text{ for } y_i = +1 \qquad (4)$$

$$\vec{x_i} \cdot \vec{w} + b \leq -1 + \xi_i \text{ for } y_i = -1 \qquad (5)$$

$$\xi_i \geq 0 \qquad (6)$$

The SVM solution to this problem follows the same principle above. The only difference is that now an upper bound to the Lagrangian multipliers $0 \leq \alpha_i \leq C$ is set, where C is a parameter defining the penalty of classification errors during training.

The main reason of the success of SVMs is the ability to handle non-linear problems by mapping the feature space to some higher order space where the problem is again linear. Applying a *kernel* function $\mathrm{K}(.)$ to the dot product of feature vector and weight vector performs the actual mapping in the SVM equations. Due to the kernel function's property

$$\mathrm{K}(\vec{w}, \vec{x_i}) = \phi(\vec{w}) \cdot \phi(\vec{x_i}) \qquad (7)$$

the exact mapping rule to the higher order space does not need to be computed and eq.3 is transformed to

$$y(t) = \sum_{i=1}^{N_S} \alpha_i \hat{y}_i \mathrm{K}(\vec{s_i} \cdot \vec{x}(t)) + b \qquad (8)$$

From the choice of available kernel functions we have chosen the RBF-kernel $f(x) = -\gamma \exp|\vec{x} - \vec{s_i}|$ [3,5] motivated by initial experiments where linear kernels have not converged and sigmoid kernels have produced worse results.

3. SVM/HMM acoustic models

To combine the SVM's classification abilities with time-varying speech signals we introduce a SVM/HMM combination that transforms the SVM classification result to a class posterior probability that is then used in the HMM's likelihood computation.

The transformation of the SVMs' class distances to probabilities is done by applying a sigmoid function (eq. 9) to each of the SVM outputs. The sigmoid function's parameters A and B are estimated using the algorithm from [6]. The class posterior probability of class j given feature vector $\vec{x}$ can be written as

$$\Pr(j|\vec{x}) = \frac{1}{1 + A\exp(-y_j + B)} \qquad (9)$$

with y_j denoting the SVM output representing class j. The HMMs compute their state-dependent likelihoods by using the probabilities from all SVMs (*tied*-posteriors), the SVM/HMM combination is illustrated in figure 2. Thus, the likelihood is computed as follows:

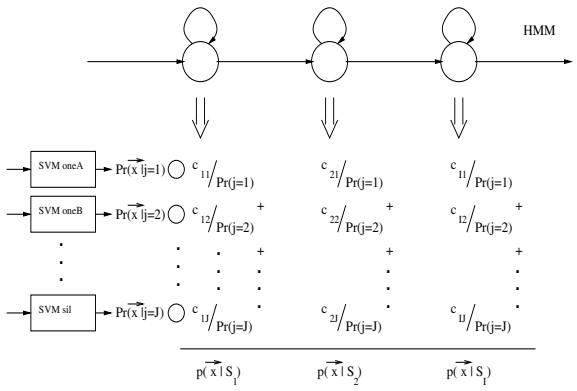

Figure 2: Likelihood generation in the hybrid SVM/HMM system

$$p(\vec{x}|S_i) \propto \sum_{j=1}^{J} c_{ij} \cdot \frac{\Pr(j|\vec{x})}{\Pr(j)} \qquad (10)$$

where $\Pr(j|\vec{x})$ denotes the posterior probability of class j given $\vec{x}$ (see above) and $\Pr(j)$ denotes the *a-priori* probability of class j that is estimated by the class' relative frequency in the training data. The mixture weights c_{ij} for each HMM state are trained on the complete data set (see section 4) using the standard Baum-Welch algorithm.

Similar to the NN/HMM tied-posterior system [4] we are able to adjust the HMM topology and the input context independently. The HMM topology consists of 11 whole-word HMMs with 16 states each and two silence models (`sil` and `sp`) with 3 states and 1 state, respectively. Regarding the input context, additional frames from past

and future time steps may be combined to one SVM input vector. One drawback compared to NN/HMM tied-posterior systems is that the posterior probabilities are not guaranteed to sum to 1 due to the independent sigmoid functions. If two SVMs produce similar high posterior probabilities an error might occur, but in most cases the HMMs should recover this kind of errors.

4. Experimental set up

The speech database used in our tests is the AURORA2 database presented in [7]. It contains a sub set of spoken digits and digit chains taken from the TI digits database added with different noise types at different signal-to-noise ratios (SNR). For our experiments we only use the clean-condition training set and compute feature vectors containing MFCC[1] features for the baseline system and RASTA-PLP features [8] for the various SVM/HMM systems. The SVM training is conducted using the SVM modules from the *Torch3* package [9].

Prior to feeding the concatenated feature vector into the SVM module (see section 3) a normalization step takes place to prevent numerical problems in the quadratic optimizer [3]. Instead of hard-limiting the range of value, we normalize the data to zero mean and unit variance, a method proven to be successful for NN training [4].

First we applied a word transcription of the training data to train one SVM for each word (plus additional SVMs for the two labels `sil` and `sp`). The SVMs themselves already produce good frame classification result (17.9% frame error rate on 1000 unseen training sentences if trained with 2000 sentences), but the SVM/HMM system is unable to distinguish between a single word ("one") and a two-word sequence ("one one") if the pause between the words is not recognized.

Therefore our next approach starts from a transcription based on HMM states. Then, new labels are formed by grouping the states of each word model in two halves. Figure 3 illustrates the process for the word *one*. Additionally, the labels for the two non-word models are grouped to one. Thus, we trained 22 SVMs for the 11 numerals and one SVM for *silence* and *short pause*. The

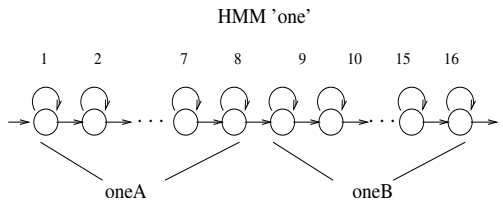

Figure 3: Alignment creation with two labels per word

frame error rate drops to 16.6% for the same 1000 sentences and the same amount of training data used for SVM training. More detailed results of the complete recognizer are given in the next section.

[1] mel-frequency cepstrum coefficients

5. Results

The baseline set up is taken from [7], the results are computed with 3 Gaussians per state (6 Gaussians for `sil` and `sp` models), input features are 13 MFCCs (including c_0) and the frame energy. The feature vector for the SVM training consists of 9 RASTA-PLP coefficients, the frame energy and the features' first and second derivatives.

System	NF	Test A	Test B	Test C
3/6 Gauss-8440	0	0%	0%	0%
MLP-TP-8440	7	-32%	-11%	-46%
13 SVM-2000	3	-58%	-35%	-80%
23 SVM-2000	3	-37%	-13%	-60%
23 SVM-3000	3	-4%	**+16%**	-17%
23 SVM-6000	3	**+3%**	**+24%**	-10%

Table 1: Acoustic models with the specified likelihood computation, relative deviation to the baseline result on the AURORA2 test sets

Table 1 shows the history of the experiments conducted on the AURORA2 task: We have started with 13 whole-word SVMs, have continued with 23 half-word SVMs (as described in section 4) and have increased the amount of data available for the SVM training. The figure after the dash in the first column denotes the number of training sentences taken from the 8440 available sentences in the AURORA2 clean training set. The second column (*NF - number of frames*) shows the number of frames that are composed to the SVM input vector. The figures from the last 3 columns denote the relative deviation of the word error rate to the baseline system (first line in table 1) on the 3 AURORA2 tests (positive numbers mark an improvement). For comparison we have also included a result from a NN/HMM tied-posterior system based on RASTA-PLP features that has performed well in a distributed speech recognizer if trained with the AURORA2 multi-condition training set [4] (second line in table 1).

Subway	Babble	Car	Exhibition
-20%	+30%	+11%	-24%

Table 2: Detailed results for the best SVM result from table 1 (Test A)

Deducing from table 1 the first observation to note is that the number of training sentences determines the system's performance. Second, a system trained with 13 SVMs on a word level transcription is worse than a system trained on a more detailed one (23 symbols, see

Restaurant	Street	Airport	Station
+24%	+10%	+32%	+28%

Table 3: Detailed results for the best SVM result from table 1 (Test B)

section 4). Regarding the different tests, we notice that the improvement on test A (known noise types) is worse than the gain on test B (unknown noise types) and despite the fact that RASTA-PLP features are used (which should cope with channel distortions) the result on test C - known and unknown noise filtered with a new channel characteristic - is worse than the baseline. A more detailed investigation of the results of test A and B is given in tables 2 and 3. Obviously, the quality of the SVM system is dependent on the noise characteristic. Test A's noise types *babble* and *car*[2] are better dealt with compared to the baseline, the noise types *subway* and *exhibition* are worse. The results of test B are always better than the baseline, the good result of speech distorted with *restaurant* noise corresponds well to the result with added *babble* noise from test A.

Summarizing these facts we believe that SVM-based acoustic models might be useful for noise environments with changing noise types (referring to the results of test B) and especially background speech. Of course, other methods of coping with noisy speech that mostly assume stationary noise can be combined with our system to further improve the result.

At present we have to train one SVM for each acoustical unit so a phoneme-based SVM system with about 50 SVMs seems impractical. On the other hand, the tied-posteriors framework is very modular and one might think of a combination of SVM classifiers with neural networks to estimate posteriors and HMMs to compose the acoustical models.

6. Conclusion

The SVM learning algorithm provides strong classification capabilities for binary decisions derived from statistical learning theory. The scope of this work focuses on the implementation of a SVM/HMM acoustic model exploring the advantages of both algorithms. To this extend, we have trained one SVM for each acoustical unit and have transformed the SVM's class distance to obtain an estimation for a class posterior probability. To improve the SVM decision we have extended the input vector by adding context frames. Next, the HMMs calculate their output densities using these tied-posteriors. Since the AURORA2 database is used for testing the system, we have designed whole word HMMs with two SVMs estimating the appearance of the first and last half of each word, respectively. Results on the AURORA2 tests A and B show a promising improvement of the word error rate over a Gaussian baseline system especially if dealing with speech signals distorted by background speech or unknown noise types. Future work will include the integration of SVMs in a NN/HMM tied-posterior system.

[2] The noise characteristics are presented in [7]

7. References

[1] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1994.

[2] J. Stadermann and G. Rigoll, "Comparing NN Paradigms in Hybrid NN/HMM Speech Recognition using Tied Posteriors," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, St. Thomas, U.S. Virgin Islands, Nov. 2003.

[3] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid SVM/HMM Architectures for Speech Recognition," in *Proc. of Neural Information Processing Systems (NIPS)*, 2000.

[4] J. Stadermann and G. Rigoll, "Flexible Feature Extraction and HMM Design for a Hybrid Distributed Speech Recognition System in Noisy Environments," in *Proc. ICASSP*, Hongkong, China, Apr. 2003.

[5] C. J. C. Burges, *A tutorial on support vector machines for pattern recognition*. Bell Laboratories, Lucent Technologies, 1998.

[6] J. C. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds. MIT Press, 2000, pp. 61–74.

[7] H. G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, 2000.

[8] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.

[9] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," IDIAP, Tech. Rep. 02-46, 2002. [Online]. Available: http://www.torch.ch