HSVM - A SVM TOOLKIT FOR SEGMENTED SPEECH DATA

André Stuhlsatz

Cognitive Systems Group,
Otto-von-Guericke University, 39016 Magdeburg, Germany
Laboratory for Pattern Recognition,
University of Applied Sciences, 40474 Duesseldorf, Germany,
andre.stuhlsatz@fh-duesseldorf.de

Abstract: This paper introduces add-on tools aligned with the Hidden-Markov-Model Toolkit (HTK) to use Support Vector Machines (SVM) in speech recognition. The resulting method and tool is named HSVM: **H**idden-Markov-Model Toolkit using **S**upport **V**ector **M**achines. Because SVMs have proven their generalization performance compared to the common maximum likelihood or MAP approach, a speech recognizer can profit from the use of SVMs for classifying the acoustic features. Given segmented speech data provided by a underlying HMM, the presented tools enable to post-classify acoustic features based on N-best-lists or recognition lattices. HSVM transforms the SVM predictions to probabilities and feeds them back into the recognition process in a offline fashion. Improvements using HSVM are presented for a phoneme classification on the TIMIT corpus as well as Wallstreet Journal Cambridge corpus.

1 Introduction

Hidden Markov Models (HMM) are the state-of-the-art of modern speech recognition architectures for modeling the temporal variability of speech. To predict from acoustic features $\boldsymbol{x} \in I\!R^n$, Maximum A Posteriori (MAP) classifiers are employed, i.e.

$$y^*(\boldsymbol{x}) := \underset{y\in\mathcal{Y}}{\operatorname{argmax}} \left[\frac{p(\mathbf{x}|\vartheta_y)P(\vartheta_y)}{p(\mathbf{x})}\right]. \tag{1}$$

The parameters ϑ_y are usually estimated from a set $\mathcal{X}_y$ of features $\boldsymbol{x}_m \in I\!R^n$ belonging to known classes $y \in \mathcal{Y}$ (words or phonemes) utilizing a Maximum Likelihood (ML) method:

$$\vartheta_y^*(\mathcal{X}_y) := \underset{\vartheta_y}{\operatorname{argmax}} \left[p(\mathcal{X}_y|\vartheta_y)\right]. \tag{2}$$

But this practical approach is not optimized with respect to a best generalization performance. Other methods are known to give superior generalization mainly on static data, namely Support Vector Machines [1]. Unlike the ML method that minimizes the empirical risk

$$\begin{aligned} R_{emp}(\vartheta) &:= \frac{1}{M}\sum_{m\in\mathcal{M}} L(\mathbf{z}_m,\vartheta) \\ \mathbf{z}_m &:= (\mathbf{x}_m, y_m) \in I\!R^n \times \mathcal{Y}, m \in \mathcal{M} := \{1,\ldots,M\} \end{aligned} \tag{3}$$

using an appropriate loss function L, the Support Vector Machine algorithm exerts influence on an upper bound of the actual risk

$$R_{act}(\vartheta) := \int\limits_{I\!R^n\times\mathcal{Y}} L(\mathbf{z},\vartheta)dP(\mathbf{z}) \tag{4}$$

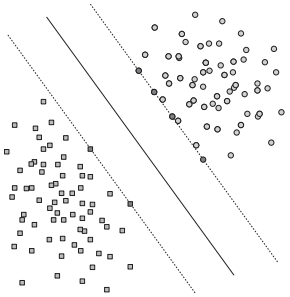


Figure 1 - Affine linear separating hyperplane in a generalized space called feature space. The dark points are the Support Vectors laying at the margin (dotted lines).

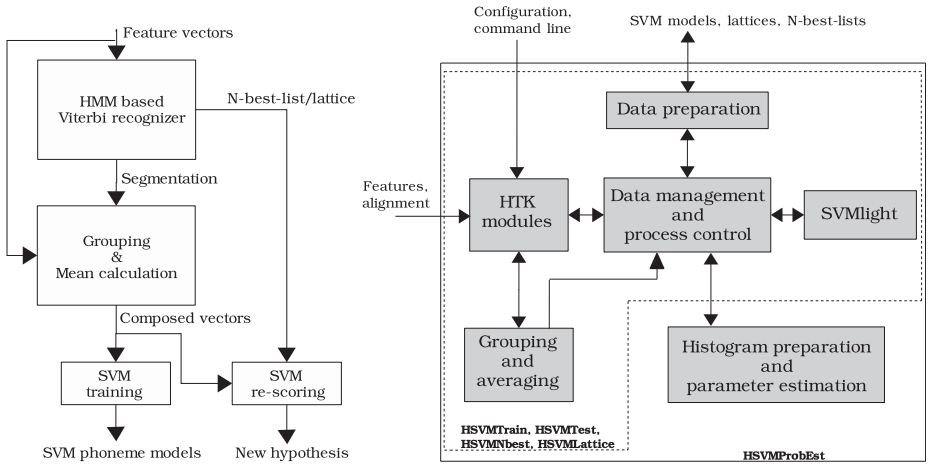
by controlling its complexity h, called VC-Dimension [1]. This is achieved by learning a linear decision function $y^* : I\!R^n \to \mathcal{Y} := \{-1, 1\}$,

$$y^*(\mathbf{x}) := \text{sign}\left[\beta + \sum_{m \in \mathcal{S}} \alpha_m^* y_m k(\mathbf{x}, \mathbf{x}_m)\right], \mathcal{S} := \{s \in \mathcal{M} \mid 0 < \alpha_s^* \leq \zeta\}, \beta \in I\!R \qquad (5)$$

from a set of given training examples $\mathbf{z}_m$, which minimizes the empirical risk and provides a maximum margin between two clusters and their class boundary (Fig. 1). The parameter ζ is free and controlls the ratio of margin and empirical error, and $k(\mathbf{x}, \mathbf{x}_m) = \langle \mathbf{x}, \mathbf{x}_m \rangle$. The vectors $\mathbf{x}_m \in I\!R^n$, $m \in \mathcal{S}$ are called Support Vectors (SV). Although, the SVM learns a linear classification in the input space $I\!R^n$, it is possible to learn a non-linear decision boundary using a non-linear transformation $\varphi : I\!R^n \to \mathcal{H}$ which maps the features into a high-dimensional space $\mathcal{H}$. However, one only has to know the kernel function evaluations $k(\mathbf{x}, \mathbf{x}_m) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}_m) \rangle$ without knowing φ explicitly. But for this purpose the kernel k has to satisfy the Mercer's condition [2] to be a scalar product in some $\mathcal{H}$. Commonly used kernels are the polynomial kernel $k(\mathbf{x}, \mathbf{x}_m) := (\langle \boldsymbol{x}, \mathbf{x}_m \rangle + 1)^d$ and the Gaussian radial basis function kernel $k(\mathbf{x}, \mathbf{x}_m) := \exp(-\|\mathbf{x} - \mathbf{x}_m\|^2 / 2\sigma^2)$. The SVM is particularly capable on small training-sets, where a small empirical error (3) cannot ensure a small actual risk (4), and it shows a better ability to generalize than empirical learning algorithms, e.g. Artificial Neural Networks (ANN) or MAP classifiers. Unfortunately, SVMs do not have the ability to model the temporal structure of speech, yet. Thus, a speech recognition architecture, which combines both methods (HMMs for modeling the temporal variability and SVMs for acoustic classification), may improve the recognition performance compared to a stand-alone HMM system. This paper introduces methods and software tools, namely HSVM: **H**idden-Markov-Model Toolkit using **S**upport **V**ector **M**achines, for training and testing SVM classifiers with segmented speech data, the estimation of probabilities from SVM outputs, as well as re-scoring N-best-lists and lattices utilizing the trained SVM classifiers. Recognition improvements using HSVM are presented for a phoneme classification on the TIMIT corpus as well as Wallstreet Journal Cambridge corpus [3].

2 What is HSVM?

HSVM provides an environment to investigate a combined system architecture consisting of a HMM based speech recognizer as well as SVM classifiers (Fig. 2). A underlying HMM recognizer architecture is needed, because of the inability of SVMs to model the temporal evolution. The segmentation information provided by the Viterbi decoder is used to compose incoming features to vectors of constant length. This is crucial for the training of static SVM classifiers



(a) Functional view. (b) The HSVM architecture.

Figure 2 - Scheme of HSVM including the HMM recognizer [4] and SVMlight [5].

and the SVM re-scoring of recognition hypotheses. HSVM is embedded into the system environment of the Hidden Markov Toolkit (HTK) [4]. HTK, developed at the Cambridge University Engineering Department (CUED), provides powerful tools to handle HMM models, mainly for speech recognition. The tools are written in the programming language C (like HSVM) and the modular design makes it possible to extend their functionality or to develop new tools. The SVM training and classification algorithms are implemented by SVMlight [5]. SVMlight is able to solve the SVM optimization problem efficiently. It handles several thousand training examples using a decomposition strategy and kernel caching.

3 HSVM in Depth

For the sake of brevity, in the following only the most important features of the tool box are described. All other options are described in more detail in [6].

4 Generic Properties

The command line invocation of a HSVM tool *foo* is similar to the general form for invoking a HTK tool [4]. The general syntax is

HSVMfoo [option] *SVMlist Files*

where *SVMlist* is a list-file consisting of all labels (e.g. phonemes or words) used for training, testing and re-scoring. The list-file has to be a simple text file with one label per line. *Files* are the parameterized feature data files which have to be in either native HTK format or Entropic Esignal ESPS format [4]. Explicitly, for every feature-file a label-file has to exist. A label-file contains the alignment information and a description of the data. Several file formats are defined for the label-files - HTK, ESPS, TIMIT and SCRIBE label-file formats [4]. Multiple training-files and label files can be handled using Script-Files (SF) and Master-Label-Files (MLF) [4]. After a tool HSVMfoo is invoked, it reads in all feature-files and label-files. HSVMfoo uses the information contained in these files to build composition vectors [3] which are then independent of the segment duration and are needed for the use of SVM classifiers (Fig. 3). The percentage of grouping can be controlled by the option -r *number of regions* $\%w_1, \%w_2, \ldots, \%w_i$. The

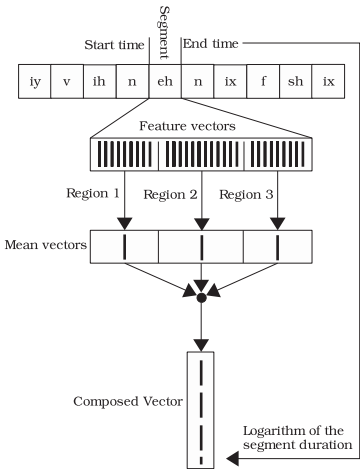


Figure 3 - Scheme of the composed vector building process.

numbers $\%w_1, \%w_2, \ldots, \%w_i$ are the specific weights denoted in percent. Internally, the composition vectors are stored in a dynamic hashtable, because it is not known in advance the order of the labels as well as how many times some label will occur in the descriptions. Utilizing a hashtable for storing also improves the average access time. In this context, it is important to note that HSVMfoo holds all composition vectors in memory to save time and that only the memory management of the OS decides when it is necessary to swap out the data on the harddisk. The data transfer from and to the hashtable, the HTK modules and SVMlight is managed by HSVMfoo's data managment and process control block (Fig. 2).

5 HSVMTrain

Syntax: **HSVMTrain** [option] *SVMlist trainFiles*

HSVMTrain implements the training of the SVM classifiers. After all *trainFiles* are preprocessed, HSVMTrain starts the training of the SVMs associated to the labels in *SVMlist* and the setting of the option -l *label*. If *label* is set, then HSVMTrain estimates one SVM model, else all SVM models are estimated as listed in *SVMlist*. A SVM model is estimated using the composition vectors associated to the label that is in process, currently. This label accounts for category $\{1\}$, while the remaining composition vectors establish the opposite set of class-membership $\{$-1$\}$. Therefore, it is called one versus all training, instead of one versus one training. Each trained SVM model will be saved as a file named *label*.mod. The option -t *type* sets the type of kernel function SVMlight will use. Defined types are 0: euclidean kernel $k(\mathbf{x}, \mathbf{x}_m) := \langle \mathbf{x}, \mathbf{x}_m \rangle$, 1: polynomial kernel $k(\mathbf{x}, \mathbf{x}_m) := (s\langle \mathbf{x}, \mathbf{x}_m \rangle + z)^d$, 2: Gaussian radial basis function kernel $k(\mathbf{x}, \mathbf{x}_m) := \exp(-\gamma \|\mathbf{x} - \mathbf{x}_m\|^2)$, 3: sigmoid kernel $k(\mathbf{x}, \mathbf{x}_m) := \tanh(s\langle \mathbf{x}, \mathbf{x}_l \rangle + z)$ and 4: user-defined kernel. The options -d *param*, -g *param*, -s *param*, -z *param* and -u *kern* set the parameters of the above kernel functions. In conjunction to the kernel choice, the parameter ζ, which justifies the ratio between the classification error and the range of separation, can be set by -c *zeta*.

6 HSVMProbEst

Syntax: **HSVMProbEst** [option] *SVMlist trainFiles*

After the SVMs are trained using HSVMTrain, the next step is the estimation of the posterior of class-membership $P(y|f(\mathbf{x}))$ from the SVM outputs $f(\mathbf{x}) := \beta + \sum_{m \in \mathcal{S}} \alpha_m^* y_m k(\mathbf{x}, \mathbf{x}_m)$. This is necessary for connecting HMMs and SVMs, because the underlying HMM system provides probabilities, whereas the SVM outputs $f(\mathbf{x})$ represent a kind of distance measure between a test pattern $\mathbf{x}$ and the decision boundary $\{\mathbf{x}|f(\mathbf{x}) = 0\}$. At this point, HSVMProbEst comes into play. HSVMProbEst estimates the posterior $P(y|f(\mathbf{x}))$ assuming that the SVM outputs $f(\mathbf{x})$ are Gaussian distributed with equal variances [7], i.e.

$$p(f(\mathbf{x})|y) := \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-[f(\mathbf{x}) - \mu_y]^2}{2\sigma^2}\right), y \in \{-1, 1\}. \tag{6}$$

Applying Bayes formula, we get the conditional probability that the vector $\mathbf{x}$ belongs to the positive class, whereas a and b depend on μ_y and σ:

$$P(y = 1|f(\mathbf{x})) = \frac{p(f(\mathbf{x})|y = 1)P(y = 1)}{\sum\limits_{y \in \{-1,1\}} p(f(\mathbf{x})|y)P(y)} = \frac{1}{1 + \exp(a \cdot f(\mathbf{x}) + b)}. \tag{7}$$

HSVMProbEst estimates the parameters a and b using a Minimum-Squared-Error (MSE) approach $\min_{a,b} \chi^2(a, b)$, assuming a estimated posterior $\hat{P}(y = 1|f(\mathbf{x}))$ scatters identically and independently Gaussian distributed around the assumed model $P(y = 1|f(\mathbf{x}))$ for every pattern $\mathbf{x}_m \in I\!R^n, m \in \mathcal{M} := \{1, \ldots, M\}$, i.e.

$$\chi^2(a, b) := \sum_{m \in \mathcal{M}} \left[\hat{P}(y = 1|f(\mathbf{x}_m)) - P(y = 1|f(\mathbf{x}_m))\right]^2. \tag{8}$$

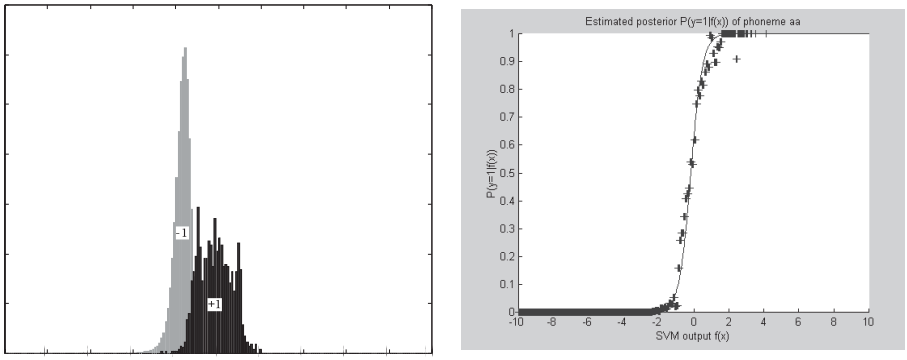
It can be shown that the minimization of the sum of squared deviations (8) is equivalent to the ML estimation. Thus, it is sufficient to obtain a ML estimation of a and b by solving $-\nabla\chi^2(a, b) = \mathbf{0}$. HSVMProbEst iterates a solution of this non-linear system by using a multi-dimensional Newton-method for non-linear equations. The approximation $\hat{P}(y = 1|f(\mathbf{x}))$ is obtained from the relative frequencies $\hat{P}(y)$ as well as $\hat{p}(f(\mathbf{x})|y)$ that is estimated by the use of histograms (Fig. 4). To determine a appropriate start point, HSVMProbEst uses a first-order Taylor-approximation of $P(y = 1|f)$. If the Newton-method does not converge (e.g. in the case of too few examples), HSVMProbEst falls back to the start parameters. After completing the estimation process, the parameters are saved in a file named *label*.prm.

7 HSVMTest

Syntax: **HSVMTest** [option] *SVMlist testFiles*

HSVMTest enables the evaluation of estimated SVM models and output probabilities using parameterized test data in *testFiles*. All label-files are composed to form a list of length T of sequential tuples $(\mathbf{x}_\tau, l_\tau)$, whereas $\tau \in \mathcal{T} := \{1, \ldots, T\}$ defines the rows within the list, $\mathbf{x}_\tau$ the composition vectors and $l_\tau \in \mathcal{L}$ the associated labels declared in the *SVMlist* $\mathcal{L}$ of length L. Both, SVM models and labels, are synonymously termed l_τ because a SVM model l_τ.mod corresponds unambiguously to the label l_τ in the description. Using the option -l *label*, a specific SVM model may be announced for testing. Else, all SVMs declared in the *SVMlist* will be tested. HSVMTest provides the Reference-Classification-Error (RCE), the total Classification-Error (CER) and optional the SVM-Hypotheses-Error (SHE) when setting the option -n. The RCE is the average of misclassified test vectors $\mathbf{x}_\tau$[1] using the SVM models l_τ and given the

[1] e.g., a test vector $\mathbf{x}_\tau$ is actually a member of class $\{1\}$, but is predicted by the SVM l_τ as a member of class $y^* = -1$



(a) Histogram for $p(f(\mathbf{x})|y)$. (b) Estimated posterior $P(y = 1|f(\mathbf{x}))$.

Figure 4 - Example estimates calculated with HSVMProbEst.

SVM output $y_l^*(\mathbf{x}_\tau), l \in \mathcal{L}$:

$$RCE := \frac{\sum\limits_{\tau \in \mathcal{T}} d(\tau, l_\tau)}{T}, \;\; d(\tau, l) := \begin{cases} 1 & : \; y_l^*(\mathbf{x}_\tau) = -1 \wedge l = l_\tau \\ 1 & : \; y_l^*(\mathbf{x}_\tau) = 1 \wedge l \neq l_\tau \\ 0 & : \; else \end{cases} . \tag{9}$$

The second value termed CER, is the average of misclassified test vectors $\mathbf{x}_\tau$ when all available SVM models $l \in \mathcal{L}$ are used to classify the vector $\mathbf{x}_\tau$:

$$CER := \frac{\sum\limits_{\tau \in \mathcal{T}} \sum\limits_{l \in \mathcal{L}} d(\tau, l)}{T \cdot L}. \tag{10}$$

Optional, the SHE determines the ratio of the number of wrong hypotheses against the total number T of predictions. Wrong hypothesis means, that $l(\mathbf{x}) := \text{argmax}_{l \in \mathcal{L}} \left[P_l(y = 1|f_l(\mathbf{x}))\right]$ from a committee of SVM models $l \in \mathcal{L}$ is not equal to the SVM model l_τ:

$$SHE := \frac{\sum\limits_{\tau \in \mathcal{T}} d(\tau)}{T}, \;\; d(\tau) := \begin{cases} 1 & : \; l(\mathbf{x}_\tau) \neq l_\tau \\ 0 & : \; else \end{cases} . \tag{11}$$

Utilizing the SHE, it is possible to evaluate the quality of the estimated output probabilities $P(y = 1|f(\mathbf{x}))$. Setting the option -x *t*, a heuristic to shrink the amount of training data corresponding to the category $\{-1\}$ will be applied. HSVMTest compares the average penalties

$$A_{l^*}(l) := -\frac{1}{|\mathcal{X}_{l^*}|} \sum_{x \in \mathcal{X}_{l^*}} \ln \left[P_l(y = 1|f_l(\mathbf{x}))\right] \tag{12}$$

for every SVM model $l \in \mathcal{L}$, whereas $\mathcal{X}_{l^*} := \{\mathbf{x}_\tau | \tau \in \mathcal{T} \wedge l_\tau = l^*\}$ denotes the set of composition vectors $\mathbf{x}_\tau$ belonging to the unique label $l^* \in \mathcal{L}$, specified by the option -l *label*. Every label $l \neq l^*$ is deleted from the given *SVMlist*, if it fulfills the condition $A(l_\tau) > t$. After reduction, HSVMTest writes the new *SVMlist* to the file named *label*.list (*label* $\equiv l^*$). This new list can be used to retrain a SVM with a smaller amount of training data. Additionally, a statistic of $A_{l^*}(l)\ \forall l \in \mathcal{L}$ can be written to the file *label*.stat via the option -z.

8 HSVMNbest

Syntax: **HSVMNbest** [option] *SVMlist testFiles*

The last step of using SVMs in combination with a HMM based speech recognizer is the re-scoring of hypotheses produced by the HMM recognizer. Re-scoring N-best-lists is one possible approach [8]. This is the task of the tool HSVMNbest. A N-best-list produced by the HTK Viterbi-Decoder consists of the N most likely recognition hypotheses of a test utterance according to the HMM score. All hypotheses are broken down into their descriptions and their segmentations, as well as their acoustic scores. HSVMNbest adds the scores $\ln\left[P(y=1|f(\mathbf{x}))\right]$ supplied by the SVM models linear weighted to the HMM scores:

$$score_{HMM/SVM} := \alpha \cdot score_{HMM} + \beta \cdot score_{SVM}. \tag{13}$$

The weights α and β are set via the option -s α β. After re-scoring the N-best hypotheses of the test data, HSVMNbest stores either the re-scored label-files *testFiles*.rec or a re-scored MLF. It is also possible by setting the -f option to save only the hypothesis of the N-best ones that has the maximum total score (termed 1-best hypothesis). Then, the performance of the combined system can be evaluated utilizing the HTK tool HResults [4].

9 HSVMLattice

Syntax: **HSVMLattice** [option] *dict SVMlist testFiles*

Unlike HSVMNbest, HSVMLattice re-scores label lattices produced using the HTK Viterbi-decoder [4]. The re-scoring of recognition lattices (which also include the hypotheses of the N-best-list) instead of N-best-lists was proposed in [3]. In the same manner, when re-scoring the N-best-lists, HSVMLattice re-scores the label lattices using a linear combination (13). The weights α and β are set via the option -s α β. After re-scoring the lattice, the best hypothesis is found by searching through the lattice for the best path[2], e.g. utilizing the HTK tool HLRescore [4]. Unlike the invocation of the other tools, HSVMLattice needs a dictionary *dict* as well as a lattice file for every given *testFile*. The re-scored lattices are saved as *testFile*.lat.

[2]The best path has the maximum total score.

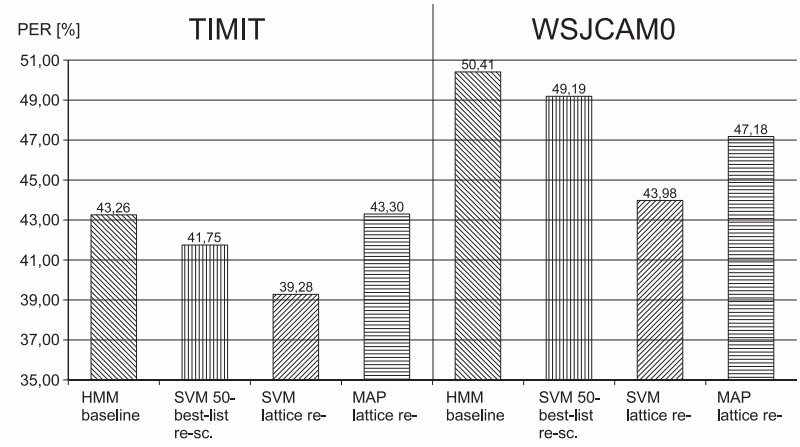


Figure 5 - Absolute phoneme error rates using HSVM.

10 Experimental Results using HSVM

Phoneme recognition results using HSVM were obtained on the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) and on the Wallstreet Journal Cambridge Corpus (WSJCAM0) (presented in [3]). The results on the test-set are summarized in (Fig. 5). After SVM N-best-list re-scoring, choosing the best hypothesis improved the phoneme error rate (PER) about 3.5% (TIMIT) and 2.4% (WSJCAM0) relating to the HMM baseline error. The re-scored lattices further improved the error rates about 9.2% PER (TIMIT) and 12.8% PER (WSJCAM0). For comparision, results using simple MAP classifiers instead of SVMs are also shown in (Fig. 5).

11 Conclusion

This paper presented a toolbox (HSVM) containing tools aligned with HTK for combining HMM based speech recognizer with Support vector Machines. The main features are:

- Training of SVMs with segmented speech data (HSVMTrain).
- Test and Evaluation of trained SVMs with segmented speech data (HSVMTest).
- SVM output probability estimation (HSVMProbEst).
- Offline N-best-list/lattice re-scoring (HSVMNbest/HSVMLattice).

Improved phoneme recognition rates are reported using HSVM indicating the generalization performance of SVMs to model the acoustic features of speech.

Reference

[1] Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag, 1995

[2] Courant, R., Hilbert, D.: Methods of Mathematical Physics. Interscience Publishers, Inc., 1953

[3] Stuhlsatz, A., Katz, M., Krüger, S.E., Meier, H.-G., Wendemuth, A.: Support vector machines for postprocessing of speech recognition hypotheses. Proceedings of International Conference on Telecommunications and Multimedia TEMU, 2006

[4] Young, S.: HTK. Cambridge University Engineering Department (CUED), 1999, http://htk.eng.cam.ac.uk

[5] Joachims, T.: Making Large-Scale SVM Learning Practical. Schölkopf, B., Burges , Smola, A. (ed.), Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999, SVMlight: http://cs.cornell.edu/People/tj/svm_light

[6] Stuhlsatz, A.: HSVM - Reference Manual, Technical Report, University of Magdeburg FEIT-IESK

[7] Platt, J.: Probalistic outputs for support vector machines and comparisons to regularized likelihood methods. Smola, A., Bartlett, P., Schlkopf, B., Schuurmans, D., editors, Advances in Large Margin Classifiers, MIT Press ,1999

[8] Ganapathiraju, A.: Support Vector Machines for Speech Recognition. PhD thesis, Mississippi State University, 2001