Hybrid Support Vector Machine / Hidden Markov Model approach for continuous speech recognition

Shantanu Chakrabartty, Guneet Singh and Gert Cauwenberghs
shantanu@bach.ece.jhu.edu, gsingh@ece.jhu.edu, gert@bach.ece.jhu.edu
Center for Language and Speech Processing(CLSP) and
Department of Electrical and Computer Engineering
The Johns Hopkins University
Baltimore, MD 21218 USA

Abstract— **A hybrid Support Vector Machine (SVM) and Hidden Markov Model (HMM) approach is proposed for designing continuous speech recognition systems. Using novel properties of SVMs and combining them with HMMs one can obtain models that map easily to hardware and leads to more modular and scalable design. The overall architecture of the proposed system is based on the MAP (maximum a posteriori) framework which offers a direct, feed-forward recognition model. The SVMs generate smooth estimates of local transition probabilities in the HMM, conditioned on the acoustic inputs. The transition probabilities are then used to estimate the global posterior probabilities of HMM state sequences. A parallel architecture that implements a simple speech recognition model in real-time is presented.**

I. Introduction

Speech recognition is a complicated problem when mapped to hardware especially when one is limited by hardware resources like silicon area on a chip. Therefore, one seeks a design approach which is flexible and scalable to meet hardware requirements of high and low complexity by trading off with the accuracy of the system. This however, requires insight into the working of the system so that desired results can be achieved by appropriate controlling of the design parameters.

Hidden Markov Model (HMMs) have been *the* principle behind all practical speech recognition systems. This is largely because they provide a unified framework on the basis of which the entire recognition system can be build . However with this scalable feature HMMs are plagued by an explosion in parameter space for which there is not enough data to estimate their values reliably, leading to problems of over-fitting and hence poor generalization .

Support Vector Machines (SVMs) on the other hand are appealing for our purpose mainly because

1. They generalize well even with relatively few data points in the training set, and bounds on the generalization error can be directly estimated from the training data.
2. The only parameter that needs to be chosen is a penalty term for misclassification which acts as a regularizer [9] and determines a trade-off between resolution and generalization performance. Hence we can control its learning ability.
3. The algorithm finds, under general conditions[1], a unique separating decision surface that provides the best out-of-sample performance.

By combining SVMs with HMMs we would want to get the best from both the worlds, the scalability of the HMM combined with the insight and better generalization ability of the SVM concept. The focus of this paper is to describe a system architecture based on the hybrid approach and provide an analog circuit implementation for it. Section II describes the overall speech recognition problem and fits the support vector formulation into it. Section III describes circuit implementation for the hybrid approach. Section IV describes some of the simulation results obtained using SpectreS and section V finally summarizes the inferences that can be drawn.

II. MAP formulation for Speech Recognition

Speech recognition problem involves finding out the most probable spoken sentence $\overline{M}$ given a sequence of acoustic vectors or feature vectors

$$\overline{M} = \arg\max_{M} P(M|\mathbf{X}, L, \theta) \tag{1}$$

where X is input acoustic vector sequence,L is language model ,θ represent parameters in the model M and $P(.)$ denotes a probability measure.

An HMM MAP(Maximum aposteriori) formulation decomposes the equation (1) into

$$P(M|\mathbf{X}, L, \theta) = \sum_{\Gamma_j} P(M, \Gamma_j|\mathbf{X}, L, \theta) \tag{2}$$

$$= \sum_{\Gamma_j} P(M|\mathbf{X}, L, \theta, \Gamma_j) P(\Gamma_j|\mathbf{X}, L, \theta) \tag{3}$$

where Γ_j is the jth path through the trellis of an HMM [1] which consists of sequence of states $\{q_j[1], q_j[2], ..., q_j[n]\}$, $q_k[n]$ representing occurance of state k at n^{th} stage of the trellis. The first term in (3) represents the probability of a state

[1]There can be pathological cases under which the problem of finding the support vectors is degenerate and a unique solution or any solution may not exist, but one can detect such case and therefore avoid it.

This work is supported by grants from The Catalyst Foundation, New York

Proc. 43rd IEEE Midwest Symp. on Circuits and Systems, Lansing MI, Aug 8-11, 2000
0-7803-6475-9/00/$10.00 ©IEEE 2000

sequence belonging to a sentence model M, which can be assumed to be a binary indicator function [2]. Using the first order markov assumption the resultant equation can then be solved by the forward recursion algorithm given by

$$\alpha_k[n] = \sum_j \alpha_j[n-1] P(q_k[n]|q_j[n-1], \mathbf{X}, L, \theta) \quad (4)$$

where α_k^n is the probability of being in state k at nth stage of the trellis $P(q_k[n]|X, L, \theta)$ [2]. The philosophy behind the hybrid approach is to generate the posterior probabilities $P(q_k[n]|q_j[n-1], \mathbf{X}, L, \theta)$ in equation (4) by using SVMs.

A. Support Vector Machines and Speech Recognition

In its basic form, an SVM classifies a pattern vector x based on the training data points x_ℓ and corresponding labels y_ℓ into classes $\{-1, +1\}$ based on the sign of y in

$$y = \sum_{\ell=1}^{L} \lambda_\ell \, y_\ell \, K(\mathbf{x}_\ell, \mathbf{x}) - b \quad (5)$$

where $K(\cdot,\cdot)$ is a symmetric positive-definite kernel function which can be freely chosen subject to fairly mild constraints [12]. Several widely used classifier functions reduce to special valid forms of kernel functions, like polynomial classifiers, multilayer perceptrons[2], and radial basis functions. The obtained support vectors are found to be almost invariant to the type of kernel function used [4], which indicates that the choice is not critical to classification performance.

The parameters λ_ℓ and b are determined by a linearly constrained quadratic programming problem [4], [7], which can be efficiently implemented by means of a sequence of smaller scale subproblem optimizations [11]. The key point for efficient implementation of the SVM is that typically only a small fraction of the λ_ℓ coefficients are non-zero. In other words, only a small set of *support vectors* ("prototypical" data points $\mathbf{x}_\ell$ with non-zero λ_ℓ) are needed in the classification of y.

For extending the concept to generate posterior probabilities for multiclass problems we can use different support vector machines for classifying each class and moderating their outputs by

$$\Pr(k) = g(y_k) \quad (6)$$

where $g(y_k) = \exp(y_k)/\sum_p \exp(y_p)$,y_k is the output of the k^{th} support vector classifier and $\Pr(k)$ is the probability measure of the kth output [10]. State transition and conditional class probabilities for equation (4) can be obtained by including the hidden variable *q[n-1]* along with the observations x into a concatenated feature vector x' presented to the SVM.

[2]with logistic sigmoidal activation function, for particular values of the threshold parameter only

TABLE I

System specification and sizing

D	Input Vector Dimension	6
L	Number of support vectors per conditional class	8
N	Total number of HMM states	16

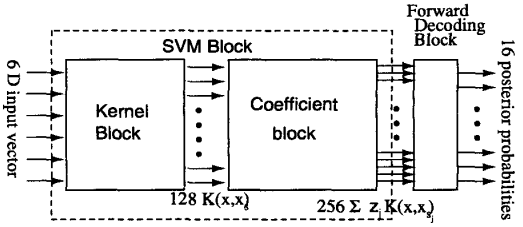

Fig. 1. *Block diagram of the overall system. The kernel block consists of 6 input stages corresponding to each dimension and 16x8x6 transistors storing support vectors. The Coefficient block consists of 16x16 transistors storing parameters* λ_ℓ.

III. Circuit Implementation

The block diagram of the overall system is shown in figure (1) and its size is specified by table I. The circuit generates probabilities of being in a particular state k $(\alpha_k), k = 1..16$ based on continuous valued input vectors.

The input vectors are presented to the SVM block which perform the support vector computation to generate unnormalized posterior probabilities Q given by

$$Q(q_k[n]|q_j[n-1], \mathbf{x}[n]) = \sum_{\ell=1}^{8} z_\ell^{jk} K(\mathbf{x}, \mathbf{x}_\ell^j) \quad i,j \in (1,\ldots,16) \quad (7)$$

where $z_\ell^{jk} = \lambda_\ell^{jk} y_\ell^{jk}$. A reasonable assumption we have used in the above equation is that the support vectors for each preconditioned input $q[n-1]$ in equation (4) will be similar. The constant b in equation (5) is absorbed by dedicating one extra constant component to the input vetor x. Therefore each output of the kernel block is reused by an equal number of parameters z_ℓ^{jk} in the coefficient block and leads to savings in silicon space.

1) SVM Block: The SVM block consists of two main functional parts namely the kernel block and the coefficient block shown in figure 1. Figure 2 shows the schematic of a cell in the kernel block. The kernel function that we have used in the circuit is a polynomial kernel of the type $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y})^2$ which is simpler to implement.

The support vectors in the SVM block and the parameters z_ℓ are stored on floating gate transistors shown by M6 in figure (2) which can be programmed by hot electron injection and

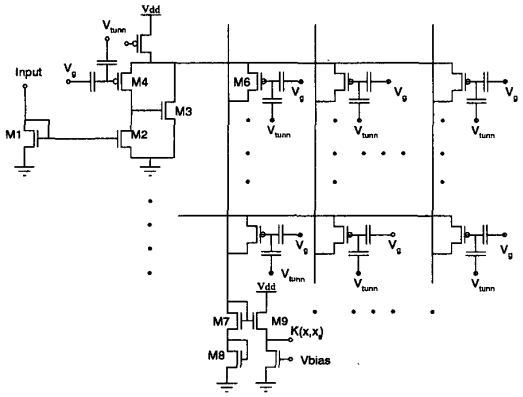

Fig. 2. *Schematic of the kernel sub-block of the SVM block.*

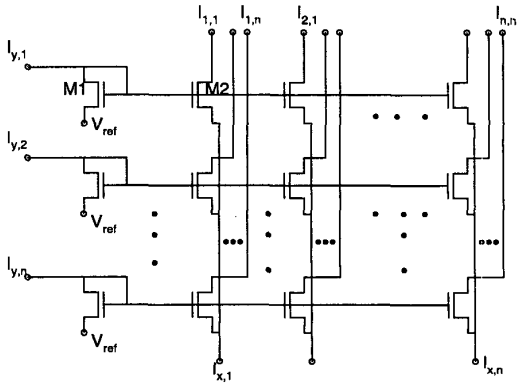

Fig. 3. *Schematic of the probability propagation circuit as given in [8].*

tunneling using terminals V_{tunn} and V_g [6].

The input to the kernel block as shown in figure 2 are presented as currents and the output is the scaled logarithm of of the input current. The block consists of a floating gate transistor M4 which tries to cancel the effects of κ variation of the support vector floating gate transistors. It uses a negative feedback loop transistor M3 which decreases the output impedance of the circuit and regulates the output voltage. This output voltage is then presented at the source of the floating gate transistor M6 where the support vectors are stored. The output current at the drain of the floating gate transistor M6 is then approximately equal to the product of the value stored at the floating gate to the value presented at the drain given by the sub-threshold equation for pMOS transistor in saturation

$$I = I_o W/L e^{-\kappa V_G/U_T} e^{V_s/U_T} \tag{8}$$

Schematic 2 also shows a squaring circuit M7-M9 that implements the $(.)^2$ function.

A similar circuit in coefficient block is used for the calculating the sum of product of the support vectors with the parameters λ_ℓ according to equation (7). The unnormalized probabilities Q forms the input to the forward decoding block which implicitly normalizes their values for probability generation.

2) Forward Decoding Block: The forward recursion given by is implemented using the probability propagation and decoding circuitry given by and a use of a low pass g_m-C filter to implement a delay element in continuous domain.

The basic element of this block consists the schematic shown in figure 3 where the output of each of the output lines is given by

$$I_{i,j} = I_z(I_{x,i}/I_x)(I_{y,j}/I_y) \tag{9}$$

with $I_x = \sum_{i=1}^{m} I_{x,i}, I_y = \sum_{j=1}^{n} I_{y,j}$ and $I_z = \sum_{i=1}^{m}\sum_{j=1}^{n} I_{i,j} = I_z$. More details about the circuit can be found in [8]. The output of this block are true probabilities as they are all normalized with respect to currents I_x and I_y.

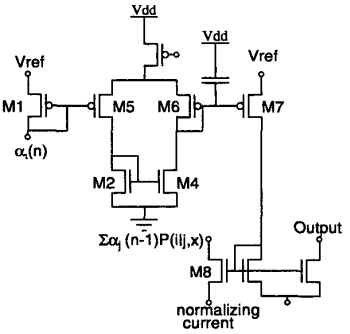

Fig. 4. *Schematic of the g_m-C filter used as a delay element in continuous mode architecture.*

Once the sum-product output I_x (proportional to $\alpha[n-1]$ is obtained the values of α have to be re-initialized according to equation (4). A continuous domain delay can be simulated by using a low pass filter. In the prototype implementation a g_m-C filter has been used as shown in the figure 4. Its a non-inverting differential amplifier with a capacitor at the output to form a gamma filter configuration.

IV. Experiments and Results

To gain insight into the working of the overall system we took a simple problem and applied the hybrid approach for pattern sequence classification. Given a long sequence of english text we want to learn how vowels and consonant patterns behave with respect to state sequences given by equation (4).

To impose hardware constraints to the above problem we would like to implement the above functionality using 2 dimension input vector and 2 support vectors per state. All the vowels and consonants were mapped to two dimensional vectors and clustered near each other. For example consonant *p* is represented by (0.8,0.2) and vowel *a* is represented by (0.2,0.8) vectors. For this particular problem we used a two state fully connected HMM model and trained it on the given 30000

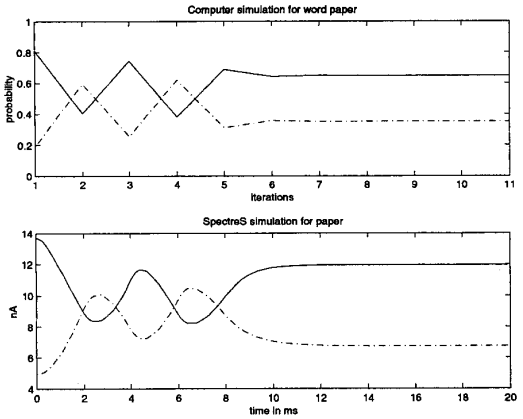

Fig. 5. *Computer and spectreS outputs of the system for the two state two dimensional system for the word* paper.

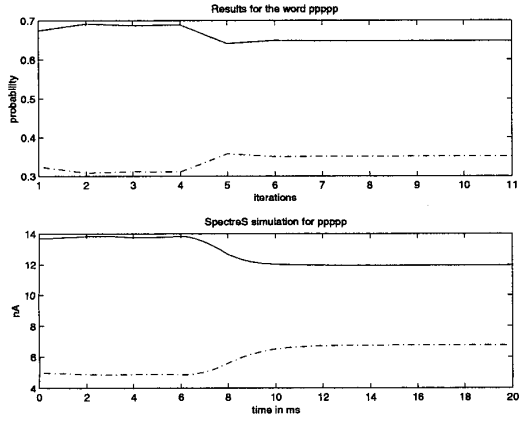

Fig. 6. *Computer and spectreS outputs of the system for the two state two dimensional system for the word* ppppp.

alphabet sequence using forward backward algorithm [13]. Then we extracted the most probable state sequence through this HMM using the viterbi algorithm to generate sequence of tuples of the form $\{q[n], q[n-1], x(n)\}$. Using a threshold value as a design parameter all the data points less than 1% count values were eliminated to generate sparse data and also reducing the size of the support vector training. With these data points we train four support vector machines each for previous state values $q[n-1]$ and then extract only the unbounded (essential) support vectors from each training routine. Because our circuit performs all computation in a single quadrant we biased all the values appropriately to obtain an architecture which conforms to the system/hardware requirement.

Figures (5) and (6) shows the result of system algorithm implemented in matlab and circuit simulation using the obtained values from the procedure in the previous section. The first graph shows plots for a valid word *paper* and the second one shows plots for an invalid word *ppppp*. The circuit simulation matches the smoothed out version of the matlab simulation of the algorithm and both of them produce results intuitive from theoretical arguments.

V. Extensions and Conclusions

The simple example illustrated in the above section can be extended to speech recognition using the system shown in figure (1). However the training procedure would be more sophisticated than the one described and forms a part of the future work.

In this paper we proposed a design approach using SVM and HMM and a circuit implementing the approach. We also applied it to a simple problem of pattern sequence recognition. In the context of speech recognition we can use a 16 state fully connected HMM to generate sequence of articulatory states which can then be used as training data for a support vector machine that can be mapped directly onto hardware.

References

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

[2] Y. Konig, *REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities in Transition-Based Speech Recognition,* Ph.D. Dissertation, Computer Science, UC Berkeley, 1996.

[3] F. Jelinek, *Statistical Methods for Speech Recognition,* MIT Press, Cambridge, 1999.

[4] V. Vapnik, *The Nature of Statistical Learning Theory,* Second Edition, Springer, 1999.

[5] G. Cauwenberghs and M. Bayoumi, *Learning on Silicon,* Kluwer Academic Publishers, Boston 1999.

[6] P. Hasler,B. Minch,J. Dugger and C. Diorio, *Adaptive Circuits and synapses using PFET Floating Gate Devices,* Learning on Silicon: Kluwer Academic Publishers, Boston 1999.

[7] B. Scholkopf, C. Burges and A.Smola, eds., *Advances in Kernel Methods-Support Vector Learning,* MIT Press, Cambridge 1998.

[8] H. Loeliger, F. Lustenberger, M. Helfenstein and F. Tarkoy, *Probability Propagation and decoding in Analog VLSI,* ISIT,IEEE, 1998.

[9] Girosi, F., Jones, M. and Poggio, T. "Regularization Theory and Neural Networks Architectures," *Neural Computation,* vol. 7, pp 219-269, 1995.

[10] Jaakkola, T. and Haussler, D., "Exploiting Generative Models in Discriminative Classifiers," to appear in *Adv. Neural Information Processing Systems (NIPS*98),* 1999.

[11] Osuna, E., Freund, R., and Girosi, F., "Training support vector machines: An application to face detection," in *Computer Vision and Pattern Recognition,* pp 130-136, 1997.

[12] Boser, B., Guyon, I. and Vapnik, V., "A training algorithm for optimal margin classifier," in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory,* pp 144-52, 1992.

[13] Baum, R. and Eagon, J.A., "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology," in *Bulletin of American Mathematical Society* , vol. 73, pp 360-63, 1967.