Advances in Speech Recognition Using Sparse Bayesian Methods

by

Jonathan Hamaker and Joseph Picone
Institute for Signal and Information Processing
Mississippi State University
Mississippi State, Mississippi

Abstract

The prominent modeling technique for speech recognition today is the hidden Markov model with Gaussian emission densities. They have suffered, though, from an inability to learn discriminative information and are prone to overfitting and overparameterization. Recent work on machine learning has moved toward models such as the support vector machine that automatically control generalization and parameterization as part of the overall optimization process. The support vector machine, however, requires ad hoc (and unreliable) methods to couple it to probabilistic speech recognition systems. In this work, we introduce the use of a probabilistic Bayesian learning machine termed the relevance vector machine as the core pattern recognition unit in a speech recognizer. The relevance vector machine system is compared to previous work using support vector machines and is found to outperform the support vector machine system in terms of both accuracy and sparsity on a continuous alphadigit task.

EDICS: 1-RECO

Please direct all correspondence to: Jonathan Hamaker, Institute for Signal and Information Processing, Mississippi State University, Box 9571, Mississippi State, MS 39762, Tel: (662) 325-8335, Fax: (662) 325-2298, email: hamaker@isip.msstate.edu.

I. INTRODUCTION

Computer speech recognition is typically framed as a pattern recognition problem which can be stated as follows: given a set of acoustic observations, $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$, and a set of models describing acoustic and linguistic patterns, we must determine which patterns were observed and, in doing so, determine which word sequence, $\boldsymbol{W} = \{W_1, W_2, \ldots, W_M\}$ was spoken. At the core of this problem is choosing amongst many different possible word sequences. This requires that we have some principled manner for directly comparing candidate transcriptions so that the "best" one may be chosen. Probabilistic modeling is a natural and very common comparison paradigm.

I.1 Probabilistic Acoustic Modeling

We can formulate the speech recognition problem as a probabilistic one where we want to find the word sequence, $\hat{\boldsymbol{W}}$, that is most probable given the acoustic observations, $\boldsymbol{O}$:

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}} \; P(\boldsymbol{W}|\boldsymbol{O}). \tag{1}$$

This *a posteriori* formulation gives us no way to apply information about the *a priori* probability of a word string. Thus, we use Bayes' rule to rewrite (1) as

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}} \; \frac{P(\boldsymbol{O}|\boldsymbol{W})P(\boldsymbol{W})}{P(\boldsymbol{O})} \tag{2}$$

where $P(\boldsymbol{O}|\boldsymbol{W})$ is the probability that the acoustic observations would be seen when a particular word sequence was spoken, $P(\boldsymbol{W})$ is the *a priori* probability of the word string $\boldsymbol{W}$ being spoken, and $P(\boldsymbol{O})$ is the *a priori* probability of the acoustic observation sequence occurring. $P(\boldsymbol{O})$ can be safely eliminated from (2) because the observation sequence, $\boldsymbol{O}$, is constant during the maximization. This yields

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}} \; P(\boldsymbol{O}|\boldsymbol{W})P(\boldsymbol{W}). \tag{3}$$

The terms in (3) are usually modeled separately. $P(\boldsymbol{W})$ is determined by a statistical *language model* which might take the form of a stochastic grammar or an N-gram language model [1,2]. $P(\boldsymbol{O}|\boldsymbol{W})$ is given by an *acoustic model*. It is the acoustic model that we will concentrate on in this paper. In most state-of-the-art recognition systems, the hidden Markov model (HMM) is used as the acoustic model [3]. The popularity of HMMs as a model of speech phenomena is owed to the HMMs ability to simultaneously model the temporal progression of speech (speech is usually seen as a "left-to-right" process) and the acoustic variability of the speech observations. The temporal variation is modeled via an underlying Markov process while the acoustic variability is modeled by an emission distribution at each state in the Markov chain. The most commonly used emission distribution is the Gaussian mixture model (GMM).

While the combination of HMMs and Gaussian mixture models (HMM/GMM) has been extremely successful, there are some key assumptions made that are not appropriate for speech modeling and that the techniques in this paper seek to overcome.

1. The HMM/GMM system makes assumptions about the parametric form of the underlying distribution which may lead to a poor match to the true underlying distribution.
2. The maximum likelihood (ML) approaches which are typically used to optimize HMM/GMM systems do not explicitly aim to improve the discriminative abilities of the model. In other words, the ML approach maximizes the probability of the correct model while implicitly ignoring the probability of the incorrect model. Ideally, the training approach should force the model toward in-class training examples while

simultaneously driving the model away from out-of-class training examples. Methods such as maximum mutual information [4,5] and minimum classification error [6] have been developed to incorporate discriminative training directly into the standard HMM/GMM framework. However, their success has been limited due primarily to their considerable computational costs [5].

I.2 Discriminative Modeling

The weaknesses of the HMM/GMM system have led researchers to seek models which mitigate some or all of them [6-9]. Hybrid connectionist systems which merge the power of artificial neural networks (ANNs) and HMMs have received a particularly large amount of attention from the research community in the past decade as an alternative to HMM/GMM systems [6-12]. The HMM/ANN hybrids have shown promise in terms of performance but have not yet found widespread use due to some serious problems. First, ANNs are prone to overfitting the training data if allowed to blindly converge. To avoid overfitting, a cross-validation set is often used to define a stopping point for the training set. This is wasteful of data and resources — a serious consideration in speech where the amount of labeled training data is very limited. ANNs also typically converge much slower than HMMs. Most importantly, the HMM/ANN hybrid systems have not shown the substantial improvements in recognition accuracy over HMM/GMM systems that would be necessary to force a paradigm shift in the research community. The approaches developed in this work draw significantly from the HMM/ANN work. However, we seek methods which are automatically immune to overfitting without the artificial imposition of a cross-validation set as well as methods which can automatically learn the appropriate model structure as part of the overall optimization process.

One such model that has come to the forefront of pattern recognition research is the support

vector machine (SVM) [13,14,15]. The support vector paradigm is based upon structural risk minimization where the learning process is posed as one of optimizing some *risk function*, $R(\alpha)$. The optimal learning machine is the one whose free parameters, α, are set such that the risk is minimized. This minimization is written as

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \; R(\alpha) = \underset{\alpha}{\operatorname{argmin}} \int Q(\boldsymbol{o}, y, \alpha) dP(\boldsymbol{o}, y) \tag{4}$$

where $Q(\boldsymbol{o}, y, \alpha)$ is a loss function which penalizes the mismatch between both the form and the parameterization of the learning machine and the true distribution; and $P(\boldsymbol{o}, y)$ is the true joint distribution of the observations and targets. Finding a minimum of the risk function is typically impossible due to the unknown distribution $P(\boldsymbol{o}, y)$. Instead, Vapnik [13] has formed an upper bound on the actual risk

$$R(\alpha) \leq R_{emp}(\alpha) + f(h) \tag{5}$$

which is related to the empirical risk (i.e. the training set error which can be measured) and a quantity, h, known as the Vapnik-Chervonenkis (VC) dimension. The VC dimension is a measure of the capacity of a learning machine to learn any training set and is typically closely related to the complexity of the learning machine's structure. With this result, we can guarantee both a small empirical risk (training error) and good generalization — an ideal situation for a learning machine.

In their most basic form SVMs use the SRM principle to impose an order on the optimization process by ranking candidate separating hyperplanes (C0, C1 and C2 in Figure 1) based on the margin they induce. For separable data, the optimal linear hyperplane is the one that maximizes the margin. The true power of the SVM, however, is how it deals with nonlinear class separating surfaces. Providing for a nonlinear decision region is accomplished using *kernels* [16]. Using this

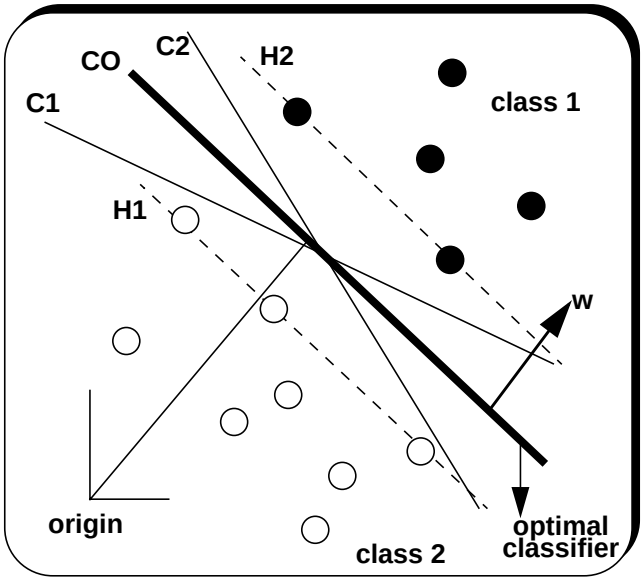

Figure 1. Hyperplanes C0-C2 achieve perfect classification (i.e. zero empirical risk). However, C0 is optimal in terms of generalization. The data points on the boundary in this case are called support vectors.

transformation of the data to a higher dimensional space by the function $\phi(\boldsymbol{o})$, kernels can define the dot product $\phi(\boldsymbol{o}_i) \cdot \phi(\boldsymbol{o}_j)$ in that higher dimensional space by

$$K(\boldsymbol{o}_i, \boldsymbol{o}_j) = \phi(\boldsymbol{o}_i) \cdot \phi(\boldsymbol{o}_j). \tag{6}$$

With this kernel function, the dot product in the high-dimensional space is computed without having to know the explicit form of $\phi(\boldsymbol{o})$.

It can be shown [13] that the SRM optimization process yields the decision function:

$$f(\boldsymbol{o}) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{o}, \boldsymbol{o_i}) + b. \tag{7}$$

where the sign of f can be used to classify examples as either in-class or out-of-class. This equation defines the SVM classifier. Only those training vectors, $\boldsymbol{o_i}$, that lie on the margin or in overlap regions have non-zero hyperparameters, α_i. In practice, the proportion of the training set

that becomes support vectors is small, making the classifier sparse. Consequently, the training process along with the training set directly optimize the complexity of the learning machine. Compare this to ANN systems where the structure of the model is typically defined *a priori*.

SVMs have had great success on static classification tasks (for example [17-20]). However, it is only recently, in the work of Ganapathiraju [9] and colleagues [21,22,23], that these techniques have been applied to continuous speech recognition. Ganapathiraju's work follows a hybrid approach combining techniques from the connectionist systems and segmental modeling systems [24,25]. It is the first to comprehensively address the problems associated with applying SVMs to continuous speech recognition. The experimental setup in our current work follows and is compared to that of Ganapathiraju as will be discussed in later sections.

While the SVMs provide an excellent classification paradigm, they suffer from two serious drawbacks that hamper their effectiveness in speech recognition. First, while sparse, the size of the SVM models (number of non-zero weights) tends to scale linearly with the quantity of training data. For a large speaker-independent corpus this effect becomes prohibitive. Second, the SVMs are binary classifiers which are only capable of producing a yes/no decision. In speech recognition this is an important disadvantage since there is significant overlap in the feature space which can not be modeled by a yes/no decision boundary. Further, the combination of disparate knowledge sources (such as linguistic models, pronunciation models, acoustic models, etc.) requires a method for combining the scores produced by each model so that alternate hypotheses can be compared. Thus, we require a probabilistic classification which reflects the amount of uncertainty in our predictions.

In the remainder of this work, we detail a Bayesian model due to Tipping [26,27] and termed the relevance vector machine (RVM) which is similar in form to the SVM but which addresses

these two problems. At the core of the RVM is a fully probabilistic model with an automatic relevance determination prior [28,29] over each model parameter. Thus, sparseness in the RVM model is explicitly sought in a probabilistic framework. The next section describes the Bayesian modeling approach that leads to the formulation of the RVM. Then we describe how the RVM is coupled into the HMM system for continuous speech recognition. Finally, we present experiments that compare the RVM system to an equivalent SVM system and describe their relative merits.

II. SPARSE BAYESIAN METHODS

In the speech problem defined earlier, the task of learning amounted to finding the values of the parameters in our model that best matched the training data. The hope was that, given sufficient training data, the model would generalize to unseen test sets. Implicit in this problem was choosing a model that was best suited to the speech task. We discussed three possible models thus far: the Gaussian mixture model-based HMMs; hybrid ANN/HMM systems; and SVMs. Embodied in this discussion are the two primary inference tasks of data modeling [28]. First, assuming that a particular model is true, we seek to infer the values for the parameters of the model that best fit the data at hand. This is exactly the training process used in the ANN hybrids — presume the ANN topology and proceed to use back propagation to find the optimal weights. The second level of inference is one that we have not addressed to this point and which is often ignored. That is the problem of inferring which model is most appropriate given the data at hand, i.e. model comparison.

A first-cut approach to model comparison might dictate that we simply choose the model that fits the data best — the maximum likelihood solution. However, a more complex model can always fit the data better. Jaynes [30] describes an extreme interpretation of this problem where we would always choose the so-called *Sure Thing* hypothesis, under which exactly the training set

and only the training set is possible. Though it is the maximum likelihood solution, the Sure Thing hypothesis is intuitively displeasing and is counter to our desire for a solution which generalizes. We avoid choosing the Sure Thing hypothesis by expressing an *a priori* preference for simpler solutions using a rather famous principle of modeling known as Occam's Razor.

MacKay [28,29] and others [31] have formalized this preference mechanism through the use of Bayesian methods. These provide a natural and quantitative embodiment of Occam's razor [29] as will be demonstrated shortly. Following MacKay [29], the first level of inference requires that we find the best-fit parameters. We can write this probabilistically as $P(\boldsymbol{w}|D, H_i)$, where $\boldsymbol{w}$ is the set of adjustable parameters, D is the data from which we will make all inferences, and H_i is the overall model of the world including the form of the model, etc. Using Bayes' rule, we can rewrite this as

$$P(\boldsymbol{w}|D, H_i) = \frac{P(D|\boldsymbol{w}, H_i)P(w|H_i)}{P(D|H_i)} \tag{8}$$

Gradient methods are typically applied to find a optimal setting of $\boldsymbol{w}$. The denominator, termed the *evidence* for the hypothesis H_i, is usually ignored during the first level of inference because it is not needed in finding the most probable parameter settings, $\hat{\boldsymbol{w}}$. This, by itself would constitute nothing more than maximum likelihood training which has long been a staple of speech modeling.

The second level of inference requires the comparison of competing hypotheses, H_1 and H_2, by finding which of $P(H_1|D)$ and $P(H_2|D)$ is a maximum. Setting this problem as a ratio of probabilities and using Bayes' rule gives

$$\frac{P(H_1|D)}{P(H_2|D)} = \frac{P(D|H_1)P(H_1)}{P(D|H_2)P(H_2)}. \tag{9}$$

If we assume that the competing hypotheses are *a priori* equiprobable (i.e. $P(H_1) = P(H_2)$),

then the best hypothesis is chosen by evaluating the evidence, $P(D|H_i)$. The evidence is computed by marginalization across the model parameters:

$$P(D|H_i) = \int P(D|\boldsymbol{w},H_i)P(\boldsymbol{w}|H_i)d\boldsymbol{w}. \quad (10)$$

It is usually impractical to compute the integration, so MacKay [28,29] prescribes an analytical approximation to the evidence computation. Under the assumption that the posterior probability in (8), $P(\boldsymbol{w}|D,H_i) \approx P(D|\boldsymbol{w},H_i)P(\boldsymbol{w}|H_i)$, is well-approximated by a Gaussian, the integrand in (10) can be assumed to have a strong peak at the most probable value of the parameters, $\hat{\boldsymbol{w}}$. The evidence can then be approximated by multiplication of the height of the integrand and the width of the posterior, $\Delta\boldsymbol{w}$. This is depicted in Figure 2.

The evidence is approximated by

$$P(D|H_i) \approx P(D|\hat{\boldsymbol{w}},H_i)P(\hat{\boldsymbol{w}}|H_i)\Delta\boldsymbol{w}. \quad (11)$$

where the term $P(D|\hat{\boldsymbol{w}},H_i)$ is the likelihood of the data given the best-fit parameter set and

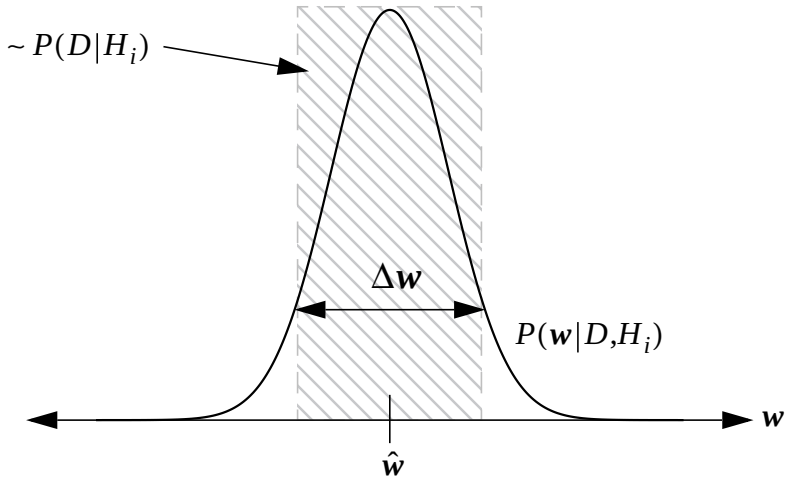

Figure 2. Evidence approximation for a single hypotheses. If the Gaussian assumption for the posterior, peaked about $\hat{\boldsymbol{w}}$, is not a good one then other methods must be employed. The width, $\Delta\boldsymbol{w}$, is the posterior uncertainty in our estimate of $\hat{\boldsymbol{w}}$ and can be determined by computing the error bars from the posterior.

$P(\hat{\boldsymbol{w}}|H_i)\Delta\boldsymbol{w}$ is a penalty on the range of $[0, 1]$ which measure of how well our posterior distribution on $\boldsymbol{w}$ fits with our prior specification. As shown in Figure 3, a more complex model would be expected to have a smaller prior probability for $\hat{\boldsymbol{w}}$, $P(\hat{\boldsymbol{w}}|H_i)$, than a less complex model and thus would be penalized more. This is precisely how the evidence embodies Occam's razor: all other things being equal, a less complex model is preferred. The evidence provides a natural trade-off between the best-fit likelihood and the Occam factor. This concept is closely related to other methods such as the Minimum Description Length [32] and the Bayesian Information Criteria [33] where the model is directly penalized by the number of parameters used. A similar idea was also seen in the SVM models which penalized models with too large a capacity (VC dimension) [16]. However, while the SVM models are forced to estimate the penalty via cross-validation schemes, Bayesian techniques automatically determine and apply the penalty in a fully probabilistic framework.

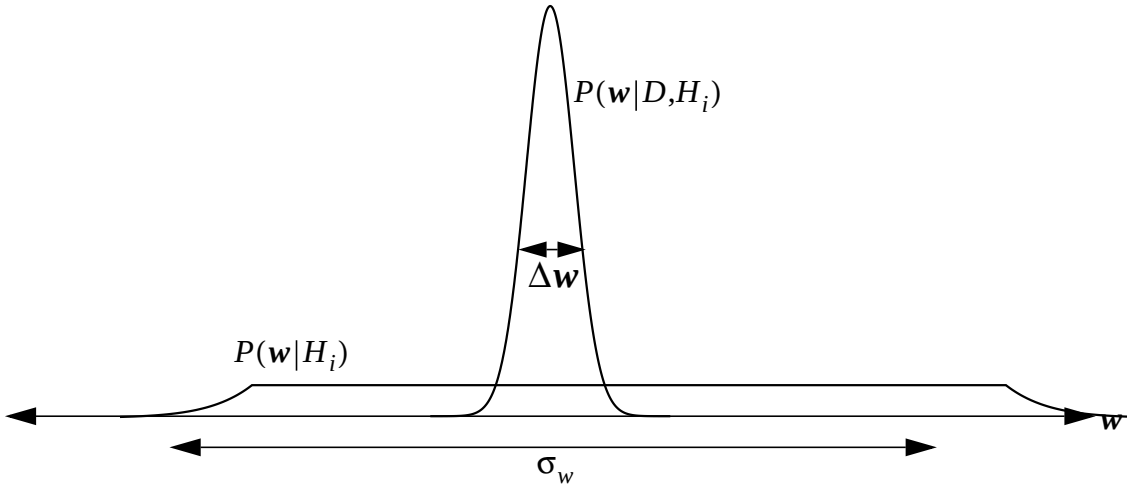

Figure 3. The prior distribution on the parameters in conjunction with the posterior distribution width determine the Occam factor. $\Delta w/\sigma_w$ determines the penalty incurred for choosing the model, H_i. A model with more parameters will tend to have a larger σ_w. Thus, the penalty for such a model will be larger. The evidence defines the trade-off between posterior likelihood and model complexity (generalization) in the Bayesian framework.

II.1 MacKay's Evidence Framework and Automatic Relevance Determination

MacKay [28] was the first to apply the evidence framework to regression and classification problems using ANNs. A brief summarization of his analysis is given now while an excellent primer on the detailed theory can be found here: [29]. Defining the training data set as $D = \{\boldsymbol{o}, \boldsymbol{t}\}$, our goal in neural network learning is to find the set of weights, $\boldsymbol{w}$, such that a global error term, $E_D(\boldsymbol{w})$, is minimized. Typically $E_D(\boldsymbol{w})$ takes the form of a squared error as in the sum squared error in back propagation

$$E_D(\boldsymbol{w}) = \sum_{j=1}^{|D|} \sum_i (t_i^{(j)} - y_i(\boldsymbol{o}^{(j)};\boldsymbol{w}))^2, \tag{12}$$

where $t_i^{(j)}$ is the i'th component of the j'th target output and $y_i(\boldsymbol{o}^{(j)};\boldsymbol{w})$ is the i'th output of the ANN when presented with training sample $\boldsymbol{o}^{(j)}$ when the weights are set to $\boldsymbol{w}$. To discourage overfitting, a weight decay or regularizer term may be added which penalizes large w_i, for example

$$E_W = \frac{1}{2}\sum_i w_i^2. \tag{13}$$

The objective function for learning thus becomes

$$M(\boldsymbol{w}) = \beta E_D + \alpha E_W. \tag{14}$$

We can give a probabilistic interpretation for (14) if we consider the neural network outputs to be perturbed by a Gaussian noise process so that

$$t_i^{(j)} = y_i(\boldsymbol{o}^{(j)};\boldsymbol{w}) + \varepsilon \tag{15}$$

where ε is a zero-mean Gaussian noise process with variance equal to $1/\beta$. Then,

$$p(t_i^{(j)} \big| \boldsymbol{w}, \beta, H) = N(t_i^{(j)} \big| y_i(\boldsymbol{o}^{(j)};\boldsymbol{w}), 1/\beta) \tag{16}$$

specifies a Gaussian distribution over $t_i^{(j)}$ with mean $y_i(\boldsymbol{o}^{(j)};\boldsymbol{w})$ and variance $1/\beta$. The total

probability of the data given the model (using the log of the sum-squared error condition) can then be written as

$$P(D|\boldsymbol{w}, \beta, H) = \frac{1}{Z_D(\beta)} e^{-\beta E_D}, \tag{17}$$

where $Z_D(\beta)$ is the Gaussian normalization term. Likewise, the log of the weight decay term, E_W, can be interpreted as a prior probability over the parameters so that

$$P(\boldsymbol{w}|\alpha, H) = \frac{1}{Z_W(\alpha)} e^{-\alpha E_W}. \tag{18}$$

With E_W as given in (13), $P(\boldsymbol{w}|\alpha, H)$ is a zero-mean Gaussian whose width is defined by $1/\alpha$. Finally, we have that

$$P(\boldsymbol{w}|D, \alpha, \beta, H) = \frac{P(D|\boldsymbol{w}, \beta, H) P(\boldsymbol{w}|\alpha, H)}{P(D|\alpha, \beta, H)}, \tag{19}$$

where, substituting (17) and (18) gives us

$$P(\boldsymbol{w}|D, \alpha, \beta, H) = \frac{\frac{1}{Z_D(\beta)} e^{-\beta E_D} \frac{1}{Z_W(\alpha)} e^{-\alpha E_W}}{P(D|\alpha, \beta, H)} = \frac{1}{Z_M} e^{-M(\boldsymbol{w})}. \tag{20}$$

It should be noted that binary and multi-class classification networks can be handled in a similar manner [29]. We simply replace the sum-square error function by a log-likelihood function, $G(\boldsymbol{w})$. The parameter, β, is not necessary in this case.

Application of (20) has the expected consequence that, by minimizing the objective function, (14), we are maximizing the probability of the weights given the constraints. Finding the most probable weights, however, is not the end of the problem. Two parameters, α and β, have been introduced which need to be estimated. Note that as α is increased, the probability distribution of the decay terms becomes peaked about zero and smoother distributions are favored. However, a value of α that is too large (i.e. too narrow a Gaussian) may limit the ability of the system to

model a complex data set. As α is decreased, more complex interpolants are allowed. Here we have the first application of the Occam factor as described above — we must find the α that provides sufficient flexibility to model the training data set without allowing so complex a model that overfitting is encouraged. A similar argument can be made for β.

Under typical statistical methods, we might turn to cross-validation to find suitable values for these two parameters, but Bayesian methods provide a natural and principled approach for estimating them using the available data. We can write down the probability of the two parameters given our state of knowledge as

$$P(\alpha, \beta|D, H) = \frac{P(D|\alpha, \beta, H)P(\alpha, \beta|H)}{P(D|H)}. \quad (21)$$

Note that $P(D|\alpha, \beta, H)$ is the evidence for α and β is the denominator in (19). Assuming we have no prior knowledge that would cause us to favor a particular value of α or β, we can find the optimal values for α and β by evaluating the evidence (if we did have prior knowledge, we would simply repeat the inference over α and β using the prior, $P(\alpha, \beta|H)$, similar to what was done in the optimization of $\boldsymbol{w}$. At some level of the inference, we will arrive at a point where our prior knowledge is too weak to apply and then we evaluate the evidence).

Unfortunately, a maximum for $P(D|\alpha, \beta, H)$ can not be found analytically in this case, so we proceed with an approximation due to MacKay [28] and Gull [31]. Under the assumption that the posterior distribution, (20), can be adequately approximated as a Gaussian, α and β can be updated as

$$\hat{\beta} = \frac{N - \gamma}{2E_D} \text{ and} \quad (22)$$

$$\hat{\alpha} = \frac{\gamma}{\sum_i (\hat{w}_i)^2}, \quad (23)$$

where N are the number of training points, $\hat{\boldsymbol{w}}$ are the most probable weights found by maximizing (20), γ is a measure of the number of parameters which are well-determined by the training data and is given by

$$\gamma = k - \alpha \mathrm{Trace}(\Sigma). \tag{24}$$

Σ is the covariance of the assumed posterior Gaussian and defines error bars on the parameters, $\boldsymbol{w}$. Σ is found by computing the negative inverse Hessian of the objective function given in (20). Iterative application of (20) and (21) provides optimal values for the system parameters, $\hat{\boldsymbol{w}}$, $\hat{\alpha}$, and $\hat{\beta}$, under the set of Gaussian assumptions.

For the explanation above, it was assumed that only one parameter, α, was used to control the complexity of all parameters in the system. In practice, we may want to group parameters of the system and control the complexity of each group separately. This requires little change to the above formulation. We now assume a Gaussian prior for each class, c, of parameters so that

$$P(\{w_i\}|\alpha_c, H) = \frac{1}{\prod Z_{W(c)}} e^{\left(-\sum_c \alpha_c E_{W(c)}\right)}, \tag{25}$$

where

$$E_{W(c)} = \sum_{i \in c} w_i^2 / 2 \tag{26}$$

and proceed with the optimization as above. In the extreme case, a control parameter, α_i, can be assigned to each weight, w_i. This extreme application of the Bayesian prior as a control parameter is known as the method of *automatic relevance determination* (ARD) [29]. It is so named because the prior over the input unit weights in a neural network can 'shut-off' those input dimensions which are irrelevant to the problem at hand. ARD is at the heart of the relevance vector machines that will be described next.

II.2 Relevance Vector Machines

All of the above analysis has been done in terms of neural network training. An application of the evidence framework to kernel machines is the relevance vector machine (RVM) [26,27]. As with SVMs, the RVMs are formed by defining a vector-to-scalar mapping as a weighted linear combination of basis functions,

$$y(\boldsymbol{o};\boldsymbol{w}) = w_o + \sum_{i=1}^{M} w_i K(\boldsymbol{o};\boldsymbol{o}_i). \tag{27}$$

The weights, $\boldsymbol{w}_i$, are the parameters to be tuned to produce an accurate model (under some appropriate measure) of the phenomena we desire to learn. At this stage, it is important to note the form of the kernel function, K. Since SVMs are optimizing a distance measure in the transform space, they require that the basis functions take the form of a so-called Mercer kernel [14] (i.e. a kernel which acts as a dot-product in some space). No such restriction is placed on the basis functions that can be employed by the RVM. However, the power demonstrated by SVMs gives compelling reason to pursue this special form of the basis function as a starting point.

We define (27) where there is one weight, w_i, associated with each training vector and $K(\boldsymbol{o};\boldsymbol{o_i})$ defines a kernel function (not necessarily a Mercer kernel). Due to the large number of parameters in this model — one per observation — we must guard against overfitting of the model to the training data. SVMs use the control parameter, C [15], to implicitly balance the trade-off between training error and generalization. RVMs take a Bayesian approach and explicitly define an ARD prior distribution over the weights

$$p(\boldsymbol{w}|\alpha) = \prod_{i=0}^{N} N\left(w_i|0, \frac{1}{\alpha_i}\right) = \frac{1}{\sqrt{(2\pi)^{N+1}|\boldsymbol{A}^{-1}|}} e^{-\frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{w}} \tag{28}$$

where we have defined $\boldsymbol{A} = diag(\alpha_o, \alpha_1, \ldots, \alpha_N)$. This prior acts to force weak components of

the model toward a weight of zero, thus finding only those inputs that are relevant to modeling the data at hand.

Each weight in the RVM model has an individual hyperparameter, α_i, that is iteratively reestimated as part of the optimization process. As the α_i grows larger, the prior on w_i becomes infinitely peaked around zero, forcing w_i to go to zero and, thus, contributing nothing to the summation in (27). This process automatically embodies the principle of Occam's Razor because it explicitly seeks the simplest model that satisfies the data constraints. In practice, the majority of the weights are pruned, resulting in an exceedingly sparse model with generalization abilities on par with SVMs [26, 34]. To complete the Bayesian specification of the model, we have to specify a prior probability over the α_i. In practice we use a non-informative (flat) prior to indicate a lack of preference [26].

With SVMs the form of (27) arises from the need to optimize the classification margin in a high-dimensional space. With RVMs, however, the goal is to directly model the posterior probability distribution. The posterior is, thus, formed by generalizing the linear model to a probability distribution with a sigmoid link function,

$$\sigma(y) = \frac{1}{1 + e^{-y}}, \tag{29}$$

and adopting the two-class Bernoulli distribution for $P(\mathrm{t}|\boldsymbol{o})$ to give

$$\mathrm{P}(\mathrm{t}_i|\boldsymbol{w}, \boldsymbol{o}_i) = [\sigma\{\mathrm{y}(\boldsymbol{o}_i;\mathbf{w})\}]^{\mathrm{t}_i} \; [1 - \sigma\{\mathrm{y}(\boldsymbol{o}_i;\mathbf{w})\}]^{1-t_i} \tag{30}$$

where $t_i \in \{0, 1\}$. Under the assumption that each data sample is drawn independently, the likelihood of the training data set can be written as

$$P(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{O}) = \prod_{n=1}^{N} \sigma_n^{t_n}(1 - \sigma_n)^{1-t_n} \tag{31}$$

where $\sigma_n = \sigma\{y(\boldsymbol{o}_n;\boldsymbol{w})\}$.

The objective of training is to find a parameter set which yields a model that is well-matched to the training data. In mathematical terms we want to find

$$(\hat{\boldsymbol{w}}, \hat{\alpha}) = \underset{\boldsymbol{w},\, \alpha}{\text{argmax}} \; p(\boldsymbol{w}, \alpha|\boldsymbol{t}, \boldsymbol{O}). \tag{32}$$

Using Bayes' rule and (31), we can form (32) as finding $\boldsymbol{w}$ and α that maximize

$$p(\boldsymbol{w}, \alpha|\boldsymbol{t}, \boldsymbol{O}) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \alpha, \boldsymbol{O})p(\boldsymbol{w}, \alpha|\boldsymbol{O})}{p(\boldsymbol{t}|\boldsymbol{O})}. \tag{33}$$

A closed form solution to this maximization is not possible so we use the iterative approximation used by MacKay [28] which was described earlier.

1. For a fixed α, find the locally most probable weights $\hat{\boldsymbol{w}}$. In other words, we want to find the $\boldsymbol{w}$ that maximizes $p(w|\boldsymbol{t}, \alpha, \boldsymbol{O})$. This is equivalent to maximizing $P(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{O})p(\boldsymbol{w}|\alpha)$. Taking the logarithm of this quantity and ignoring the scale factor on $p(\boldsymbol{w}|\alpha)$ which is a constant due to the fixed α we can write

$$\begin{aligned} L &= \log\{P(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{O})p(\boldsymbol{w}|\alpha)\} \\ &= \log\left[\prod_{n=1}^{N} \sigma_n^{t_n}(1-\sigma_n)^{1-t_n} \prod_{i=0}^{N} N\left(w_i|0, \frac{1}{\alpha_i}\right)\right] \\ &= \sum_{n=1}^{N} [t_n \log(\sigma_n) + (1-t_n)\log(1-\sigma_n)] - \frac{1}{2}\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} \end{aligned} \tag{34}$$

The gradient and Hessian of L are found by differentiating with respect to $\boldsymbol{w}$

$$\nabla_{\boldsymbol{w}} L = \Phi[\boldsymbol{t} - \boldsymbol{y}] - \boldsymbol{A}\boldsymbol{w} \tag{35}$$

$$\nabla_{\boldsymbol{w}}\nabla_{\boldsymbol{w}} L = -(\Phi^{\boldsymbol{T}}\boldsymbol{B}\Phi + \boldsymbol{A}), \tag{36}$$

where $\boldsymbol{B} = diag(\beta_1, \beta_2, \ldots, \beta_N)$ with $\beta_n = \sigma_n(1-\sigma_n)$, and Φ is the N x N+1 matrix

defined by $\Phi = [\phi(x_1), \phi(x_2), \ldots, \phi(x_N)]^T$ with

$$\phi(x_n) = [1, K(x_n, x_1), K(x_n, x_2), \ldots, K(x_n, x_N)]^T . \quad (37)$$

The Hessian defined in (36) is negative-definite everywhere, and therefore defines a unimodal, log-concave surface. We can, thus, use second-order Newton methods to solve for the $\boldsymbol{w}$ that maximizes L with an assuredness that the process will converge.

2. The Hessian is negated and inverted to give an approximation to the covariance of a Gaussian posterior over the weights, centered about $\hat{\boldsymbol{w}}$

$$\Sigma = (\Phi^{\boldsymbol{T}} \boldsymbol{B} \Phi + \boldsymbol{A})^{-1} \quad (38)$$

3. Using Σ and $\hat{\boldsymbol{w}}$ as the covariance and mean, respectively, of the Gaussian approximation, we can follow MacKay's approach [28] to update the $\{\alpha_i\}$ by

$$\alpha_i = \frac{\gamma_i}{\hat{w}_i^2}, \qquad \gamma_i = 1 - \alpha_i \Sigma_{ii} . \quad (39)$$

This iterative procedure is repeated until suitable convergence criteria are met. Central to this iterative method is the second-order Newton maximization of $P(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{O}) p(\boldsymbol{w}|\alpha)$ requiring an $O(N^3)$ inversion operation. As the quantity of training data increases, this becomes prohibitive. SVMs have a similar problem with scaling up that has been addressed through iterative refinement of the training set [35]. Current research is focusing on similar methods for RVMs [36,37].

II.3 RVM Training Refinements

The above procedure is an iterative reduction process. That is, initially each vector of the system is allocated one parameter. As the procedure continues, vectors are pruned from the model

when they are found to be irrelevant with respect the remaining parameters. Central to this iterative reestimation process is the computation of the inverse Hessian matrix. This operation requires the inversion of an MxM hessian matrix where M is initially set to the size of training set. For larger training sets (on the order of a few thousand), this computation is prohibitive both in time and in memory. In fact, initially in this work we were unable to operate on data sets larger than a few thousand training examples [34].

Tipping and Faul [38] have recently defined a constructive approach where the model begins with only a single parameter specified. All others are implicitly pruned. Parameters are then added to the system in a constructive fashion while still satisfying the original optimization function. The result of this new twist to the algorithm is that we are able to add a good bit more training data to our system — on the order of 10 thousand examples in training. However, care must be taken to insure convergence rates are reasonable. We have found that the model will often oscillate between a few local optima leading to slow convergence or even an inability to converge.

Despite our ability to increase the overall training size by approximately one order of magnitude, this iterative procedure does not completely solve the problem. For even larger problems as are typical in speech recognition, the full design matrix (or kernel matrix), Φ, will not fit in memory. We can still use the constructive approach but it requires the repeated recalculation of the full design matrix and is, again, prohibitive — now in time rather than memory. We are currently researching an approach to address this remaining issue [37].

III. EXPERIMENTS

RVMs have had significant success in several classification tasks [26]. These tasks have, however, involved relatively small quantities of static data. Speech recognition, on the other hand, involves processing a very large amount of temporally evolving signals. In order to gain insight

into the effectiveness of RVMs for speech recognition, we explored two tasks. We first experimented on the Deterding static vowel classification task which is a common benchmark used for new classifiers. Second, we applied the techniques described above to a complete small vocabulary recognition task. Comparison with SVM models are given below. For each task, the RVMs outperformed the SVM models both in terms of model sparsity and error rate. However, this comes at a significant up-front computational cost during training.

III.1 Deterding Vowel Data

In our first pilot experiment with speech data, we applied SVMs [9] and RVMs to a publicly available vowel classification task, Deterding Vowels [39]. This was a good data set to evaluate the efficacy of static classifiers on speech classification data since it has been used as a standard benchmark for several non-linear classifiers for several years. In this evaluation, the speech data was collected at a 10 kHz sampling rate and low pass filtered at 4.7 kHz. The signal was then transformed to 10 log-area parameters, giving a 10 dimensional input space. A window duration of 50 msec. was used for generating the features. The training set consisted of 528 frames from eight speakers and the test set consisted of 462 frames from a different set of seven speakers. The speech data consisted of 11 vowels uttered by each speaker in a h*d context. This data set is widely used for benchmarking non-linear classifiers. Though it appears to be a simple task, the small training set and significant confusion in the vowel data make it a very challenging task.

Table 1 shows the results for a range of nonlinear classification schemes on the Deterding vowel data. From the table, the SVM and RVM are both superior to nearly all of the other techniques. The RVM achieves performance rivaling the best performance reported on this data at 30% error rate while exceeding the error performance of SVMs and the best neural network classifier. Importantly, the RVM classifiers achieve superior performance to the SVM classifiers

Approach	Error Rate	# Parameters
K-Nearest Neighbor	44%	
Gaussian Node Network	44%	
SVM: Polynomial Kernels	49%	
SVM: RBF Kernels	35%	83 SVs
Separable Mixture Models	30%	
RVM: RBF Kernels	30%	13 RVs

Table 1: Performance comparison of SVMs and RVMs to other nonlinear classifiers on static vowel classification data [34].

while utilizing nearly an order of magnitude fewer parameters. While we do not expect the superior error performance to be typical (on pure classification tasks) we do expect the superior sparseness to be typical. This sparseness property is particularly important when attempting to build systems which are practical to train and test.

III.2 Coupling RVMs to HMM

The hybrid recognition architecture used in this work and shown in Figure 4 is a parallel of the SVM hybrid presented in [9]. Each phone-level classifier (either an SVM or RVM dichotomous classifier) is trained as a one-vs-all classifier. The classifiers are used to predict the probability of an acoustic segment. For the SVM hybrid, a sigmoid posterior fit is used to map the SVM distance to a probability [9]. The RVM output is naturally probabilistic so no link function is needed.

The HMM system is used to generate alignments at the phone level. Each phone instance is treated as one segment. Since each segment could span a variable duration, we divide the segment into three regions in a set ratio and construct a composite vector from the mean vectors of the three regions. In our experiments empirical evidence showed that a 3-4-3 proportion generally

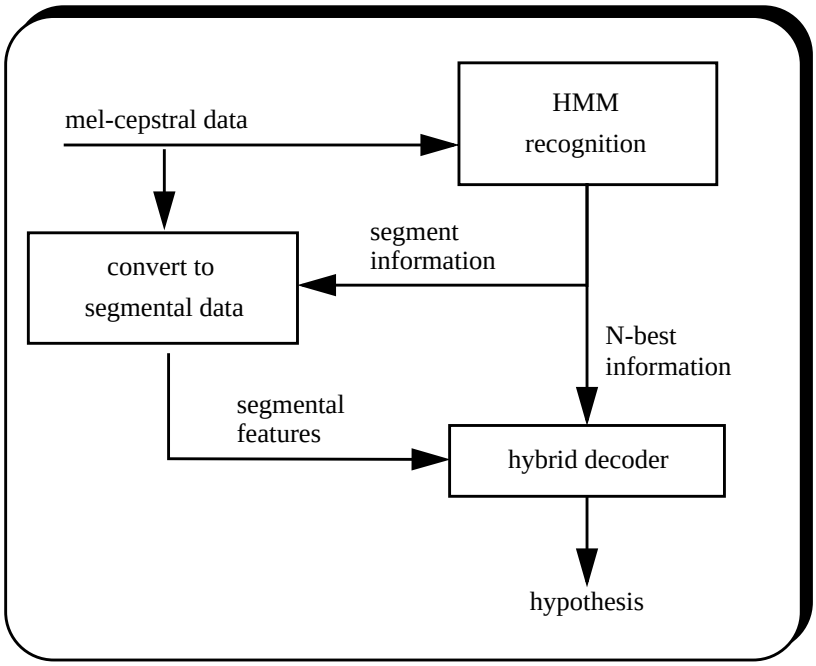

Figure 4. Flow graph for hybrid HMM/SVM and HMM/RVM systems [9].

gave optimal performance. Figure 5 shows an example for constructing a composite vector for a phone segment. The classifiers in our hybrid systems operate on composite vectors.

For decoding, the segmentation information is obtained from a baseline HMM system — a cross-word triphone system with 8 Gaussian mixtures per state. Composite vectors are generated for each of the segments and posterior probabilities are hypothesized that are used to find the best word sequence using the Viterbi decoder. The HMM system also outputs a set of N-best hypotheses. The posterior probabilities for each hypothesis are determined and the most likely entry of the N-best list is produced.

III.3 OGI Alphadigit Data

The performance of RVMs on the static classification of vowel data gave us good reason to expect the performance on continuous speech would be appreciably better than that of the SVM system in terms of sparsity and on par with the SVM system in terms of accuracy. Our initial tests

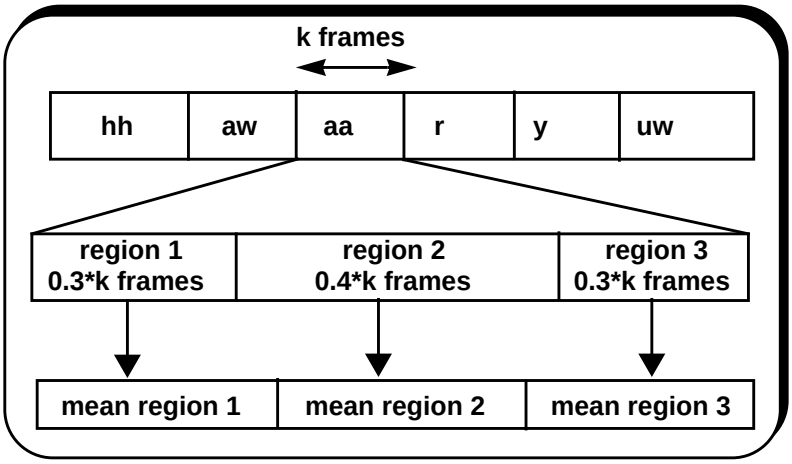

Figure 5. Composition of the segment level feature vector assuming a 3-4-3 proportion for the three sections.

of this hypothesis have been on a telephone alphadigit task. Recent work on both alphabet and alphadigit systems has taken a focus on resolving the high rates of recognizer confusion for certain word sets. In particular, the E-set (B, C, D, E, G, P, T, V, Z, THREE) and A-set (A, J, K, H, EIGHT). The problems occur mainly because the acoustic differences between the letters of the sets are minimal. For instance, the letters B and D differ primarily in the first 10-20 ms during the consonant portion of the letter [40].

The OGI Alphadigit Corpus [41] is a telephone database collected from approximately 3000 subjects. Each subject was a volunteer responding to a posting on the USEnet. The subjects were given a list of either 19 or 29 alphanumeric strings to speak. The strings in the lists were each six words long, and each list was “set up to balance phonetic context between all letter and digit pairs.” [41]. There were 1102 separate prompting strings which gave a balanced coverage of vocabulary and contexts. The training, cross-validation and test sets consisted of 51544, 13926 and 3329 utterances respectively, each balanced for gender. The data sets have been chosen to make them speaker independent.

The hybrid SVM and RVM systems have been benchmarked on the OGI alphadigit corpus

with a vocabulary of 36 words [41]. A total of 29 phone models, one classifier per model, were used to cover the pronunciations. Each classifier was trained using the segmental features derived from 39-dimensional frame-level feature vectors comprised of 12 cepstral coefficients, energy, delta and acceleration coefficients. The full training set has as many as 30k training examples per classifier. However, the training routines employed for the RVM models are unable to utilize such a large set as mentioned earlier. The training set was, thus, reduced to 10000 training examples per classifier (5000 in-class and 5000 out-of-class). The test set was an open-loop speaker independent set with 3329 sentences. The composite vectors are also normalized to the range -1 to 1 to assist in convergence of the SVM classifiers.

Both the SVM and RVM hybrid systems use identical RBF kernels with the width parameter set to 0.5. The trade-off parameter for the SVM system was set to 50. The parameters for this system were set using the optimal parameters found in Ganapathiraju's thesis [9]. The sigmoid posterior estimate for the SVM was constructed using a held-out set of nearly 14000 utterances.

The results of the RVM and SVM systems are shown in Table 2. The important columns to notice in terms of performance are the error rate, average number of parameters and testing time. In all three, the RVM system outperforms the SVM system. It achieves a slightly better error rate of 14.8% compared to 15.5%. This error rate is obtained in over an order of magnitude fewer

Approach	Word Error Rate	Avg # Parameters	Training Time	Testing Time
SVM: RBF Kernels	15.5%	994	3 hours	1.5 hours
RVM: RBF Kernels	14.8%	72	5 days	5 minutes

Table 2: Performance comparison of SVMs and RVMs on Alphadigit recognition data. The RVMs yield a large reduction in the parameter count while attaining superior performance.

parameters. This naturally translates to well over an order of magnitude better runtime performance. However, the RVM does require significantly longer to train. Since that added cost is all prior to runtime, we may be tempted to discount it as being unimportant. However, for larger tasks the RVM will not run at all as explained earlier. Thus, it is still an important research area and one we are currently addressing.

IV. CONCLUSIONS

This work is the first application of sparse Bayesian methods to continuous speech recognition. By using an automatic relevance determination mechanism, we are able to achieve state-of-the-art performance in extremely sparse models. Further, this is accomplished while maintaining a purely probabilistic framework. We also achieve performance better than the popular SVM kernel classifier while also using an order of magnitude fewer parameters for both a static classification task and a continuous speech task. However, this test-time efficiency comes at a large up front cost during training. Thus, most of our work at this point is focused on more efficient training schemes so that we can move to larger vocabulary tasks that would overwhelm our current training techniques. To this end, we have developed an iterative subset refinement approach which attempts to optimize the global criteria by locally optimizing the model on small subsets of the total training set. The subset models are incrementally used to generate a model of the full training set.

V. REFERENCES

1. F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, USA, 1997.
2. F. Jelinek, “Up From Trigrams! The Struggle for Improved Language Models,” *Proceedings of the 2nd European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1037-1040, Genova, Italy, September 1991.

3. L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-285, February 1989.

4. L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 49-52, Tokyo, Japan, October 1986.

5. P. Woodland and D. Povey, "Very Large Scale MMIE Training for Conversational Telephone Speech Recognition," *Proceedings of the 2000 Speech Transcription Workshop*, University of Maryland, MD, USA, May 2000.

6. S. Renals, *Speech and Neural Network Dynamics*, Ph. D. dissertation, University of Edinburgh, UK, 1990.

7. A. J. Robinson, *Dynamic Error Propagation Networks*, Ph.D. dissertation, Cambridge University, UK, February 1989.

8. J. Tebelskis, S*peech Recognition using Neural Networks*, Ph. D. dissertation, Carnegie Mellon University, Pittsburg, USA, 1995.

9. A. Ganapathiraju, *Support Vector Machines for Speech Recognition*, Ph.D. Dissertation, Mississippi State University, Mississippi State, Mississippi, USA, 2002.

10. M.D. Richard and R.P. Lippmann, "Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities", *Neural Computation*, vol. 3, no. 4, pp. 461-483, 1991.

11. H.A. Bourlard and N. Morgan, *Connectionist Speech Recognition — A Hybrid Approach*, Kluwer Academic Publishers, Boston, USA, 1994.

12. G.D. Cook and A.J. Robinson, "The 1997 ABBOT System for the Transcription of Broadcast News," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998.

13. V.N. Vapnik, *Statistical Learning Theory*, John Wiley, New York, NY, USA, 1998.

14. C. Cortes, V. Vapnik. Support Vector Networks, *Machine Learning*, vol. 20, pp. 293-297, 1995.

15. C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, http://svm.research.bell-labs.com/SVMdoc.html, AT&T Bell Labs, November 1999.

16. B. Schölkopf, C. Burges and A. Smola, *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA, USA, December 1998.

17. O. L. Mangasarian, W. Nick Street and W. H. Wolberg: ‘‘Breast cancer diagnosis and prognosis via linear programming", *Operations Research*, 43(4), pp. 570-577, July-August 1995.

18. Y. Le Cun, L.D. Jackel, L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, U.A. Muller, E. Sackinger, P. Simard, and V. Vapnik, “Learning algorithms for classification: A comparison on handwritten digit recognition,” in *Neural Networks: The Statistical Mechanics Perspective*, J.H. Kwon and S. Cho, eds., pp. 261--276, World Scientific, 1995.

19. T. Joachims, “Text Categorization with Supprt Vector Machines: Learning with Many Relevant Features,” *Technical Report 23, LS VIII, University of Dortmund*, Germany, 1997.

20. P. Clarkson and P. Moreno, "On the Use of Support Vector Machines for Phonetic Classification," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, USA, 1999.,

21. A. Ganapathiraju, J. Hamaker and J. Picone, “Support Vector Machines for Speech Recognition,” *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia, November 1998.

22. A. Ganapathiraju, J. Hamaker and J. Picone, “A Hybrid ASR System Using Support Vector Machines,” submitted to the *International Conference of Spoken Language Processing*, Beijing, China, October, 2000.

23. A. Ganapathiraju, J. Hamaker and J. Picone, “Hybrid HMM/SVM Architectures for Speech Recognition,” *Proceedings of the Department of Defense Hub 5 Workshop*, College Park, Maryland, USA, May 2000.

24. M. Ostendorf, V. Digalakis, and O. Kimball, "From HMM’s to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 360-378, 1996.

25. M. Ostendorf and S. Roukos, "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 37, pp. 1857-1867, 1989.

26. M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning*, vol. 1, pp. 211-244, June 2001.

27. M. Tipping, "The Relevance Vector Machine," in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen and K.-R. Muller, eds., pp. 652-658, MIT Press, 2000.

28. D. J. C. MacKay, *Bayesian Methods for Adaptive Models*, Ph. D. thesis, California Institute of Technology, Pasadena, California, USA, 1991.

29. D. J. C. MacKay, “Probable networks and plausible predictions --- a review of practical Bayesian methods for supervised neural networks,” *Network: Computation in Neural Systems*, 6, pp. 469-505, 1995.

30. E. T. Jaynes, "Bayesian Methods: General Background," *Maximum Entropy and Bayesian Methods in Applied Statistics*, J. H. Justice, ed., pp. 1-25, Cambridge University Press, Cambridge, UK, 1986.

31. S. F. Gull, "Bayesian Inductive Inference and Maximum Entropy," *Maximum Entropy and Bayesian Methods in Science and Engineering, vol. 1, Foundations*, G. J. Erickson and C. R. Smith, eds., Kluwer Publishing, 1988.

32. J. Rissanen, "Modeling by Shortest Data Description," *Automatica*, 14, pp. 465-471, 1978.

33. G. Schwartz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461-464, 1978.

34. J. Hamaker, J. Picone, and A. Ganapathiraju, "A Sparse Modeling Approach to Speech Recognition Based on Relevance Vector Machines," *Proceedings of the International Conference of Spoken Language Processing*, vol. 2, pp. 1001-1004, Denver, Colorado, USA, September 2002.

35. E. Osuna, R. Freund, and F. Girosi, “Support Vector Machines: Training and Applications,” *MIT AI Memo 1602*, March 1997.

36. A. Faul and M. Tipping, "Analysis of Sparse Bayesian Learning," *Proceedings of the Conference on Neural Information Processing Systems*, preprint, 2001.

37. J. Hamaker, *Sparse Bayesian Methods for Continuous Speech Recognition*, Ph.D. Dissertation, Mississippi State University, Mississippi State, Mississippi, USA, 2003.

38. M. E. Tipping and A. Faul, "Fast Marginal Likelihood Maximisation for Sparse Bayesian Models," submitted to *Artificial Intelligence and Statistics ’03*, 2003.

39. D. Deterding, M. Niranjan and A. J. Robinson, "Vowel Recognition (Deterding data)," Available at *http://www.ics.uci.edu/pub/machine-learning-databases/undocumented/connectionist-bench/vowel, 2000.*

40. P. Loizou and A. Spanias, "High-Performance Alphabet Recognition," *IEEE Transactions on Speech and Audio Processing*, pp. 430-445, 1996.

41. R. Cole, "Alphadigit Corpus v1.0," *http://www.cse.ogi.edu/CSLU/corpora/alphadigit*, Center for Spoken Language Understanding, Oregon Graduate Institute, USA, 1998.