A Flexible Probabilistic Framework for Large-Margin Mixture-of-Experts

Report for EE392A

Archit Sharma(14129)
architsh@iitk.ac.in
Mentors: Dr. Piyush Rai, CSE and Dr. Ketan Rajawat, EE

Abstract

The Mixture-of-Experts provides a classical framework for doing non-linear learning in an interpretable fashion. The framework combines simple experts, who handle a subset of domain (governed by the gating network). However, the constraint of using probabilistic experts restricts the experts which can be leveraged in this setting. Moreover, with slightly sophisticated experts, the inference procedures become slow and complex, often requiring multiple embedded iterative procedure. In this work, we leverage recently developed latent-variable augmentation techniques to design efficient inference procedures for sophisticated Mixture-of-Experts. Our models yield competitive/better classification accuracies on various binary classification datasets.

1 Introduction

1.1 Mixture of Experts

Mixture of Experts [8] is a simple yet extremely effective way to learn non-linear models by combining locally linear models (the setup is general enough to incorporate non-linear experts as well, for example Gaussian Processes [11]). Mixture of experts model has been widely studied and used as well [10, 24]. The idea behind mixture of experts, in its simplest form, is to break a highly non-linear problem into a set of simpler problems, each of which is handled by an expert for that subset of input. Therefore, there are two fundamental components of Mixture of Experts: *Experts* which handle the simpler sub-problem and a *Gating Network* which assigns each input to an expert. A smooth introduction to the topic of mixture of experts is given in [12].

Mathematically, assume that we that our input x is in $\mathbb{R}^D$, and we have continuous labels y in $\mathbb{R}$. In the most frequent case, we define a gating network $V \in \mathbb{R}^{K\times D}$, where K is the number of experts, as $V = \{v_k\}_{k=1}^K$. Consequently, we defines the probability of choosing the expert k as $\pi_k(x) = f_k(x, V)$, where $f_k(x, V)$ is a valid probability, that is $0 \leq f_k(x, V) \leq 1$ and $\sum_{k=1}^K f_k(x, V) = 1\ \forall x \in \mathbb{R}^D$. Next, in the most frequent case, we define our experts $W \in \mathbb{R}^{D\times K}$ as $W = \{w_k\}_{k=1}^K$. For the simple linear case, the prediction is usually $\hat{y}_k = w_k^T x$. More generally, $\hat{y}_k = g_k(w_k, x)$. Note, the prediction for expert k cannot depend on the other experts. The final prediction is given by $\hat{y} = \sum_{k=1} \pi_k(x) g_k(w_k, x)$. Alternately, we can use predicted value from the most probable expert. The former is unbiased while the latter is cheap to compute.

Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, we generally employ **Expectation Maximization** (EM) to learn the parameters V, W. To this extent, we introduce the categorical latent variable $Z = \{z_i\}_{i=1}^N$. Here, z_i is a one-hot vector which denotes the expert making the prediction for x_i. The complete log-likelihood is defined as follows:

$$\mathcal{L}(y, Z|X, W, V) = \sum_{i=1}^{N} \log p(y_i, z_i|x_i, W) = \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1]\big(\log p(y_i|x_i, w_j) + \log p(z_{ij} = 1|x_i, V)\big) \quad (1)$$

Here, we employ the conventional notation $X = \{x_i\}_{i=1}^N, y = \{y_i\}_{i=1^N}$. Here, $p(y_i|x_i, w_j)$ denotes the probability/likelihood of label y_i, and $p(z_{ij} = 1|x_i, V)$ represents the probability of expert j which is simply $\pi_k(x)$. The equation above shows why our experts need to have a probabilistic interpretation, and why powerful experts like Support Vector Machines (SVM) [3] cannot be used directly. Next, we construct the expected

complete log-likelihood, as is traditional in EM, denoted by $\mathbb{E}_z[\mathcal{L}(y, Z|X, W, V)]$. Here, the expectation is taken with respect to $p(z|y, X, V^{(t)}, W^{(t)})$.

This concludes a generic overview of mixture-of-experts. The specific details depend upon the architectural choice of gating networks and experts. While there are a multitude of such choices possible for gating architectures, we can broadly divide them into two categories which gives rise to two distinct "mixture-of-experts" models: *Flat Mixture-of-Experts* and *Hierarchical Mixture-of-Experts* [9]. Representative images of both have been shown below:

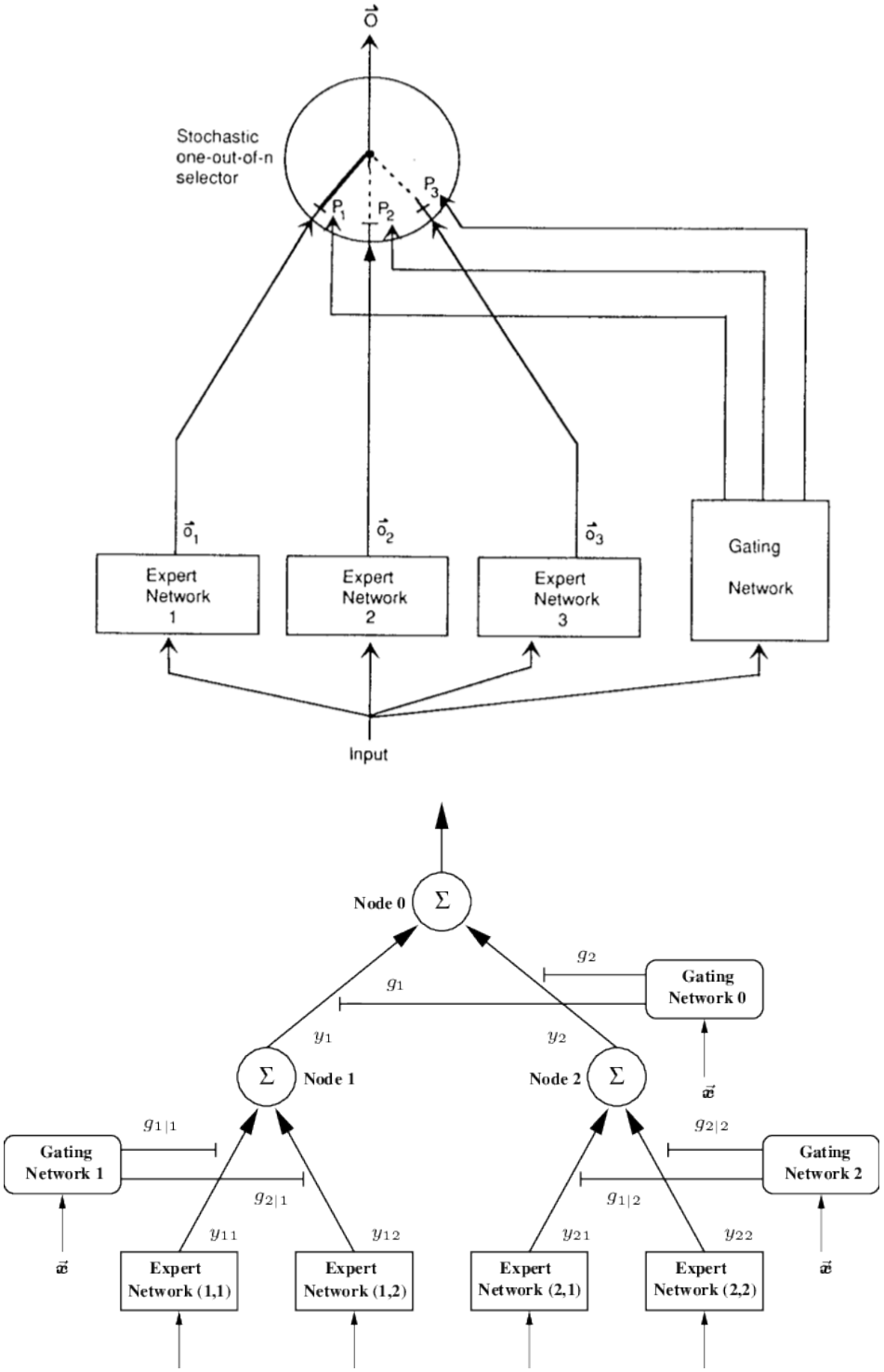

Figure 1: An illustration of a Flat Mixture-of-Experts model (top) [8] and a Hierarchical Mixture-of-Experts (bottom) [9]

Both of these models have been immensely popular. While effectively both represent similar mathematical formulations, the decision boundaries learnt by these two models are very different and thus depending upon the problem, one of them might perform better. More details on these architectures are provided later. While initially proposed for regression, these models have been adapted for classification as well [12, 24]. There are several reasons for the success of these models, especially when compared to kernel based approaches to non-linear learning [19, 21]:

- A probabilistic formulation makes a fully Bayesian treatment possible, for example [2].
- They are highly interpretable, unlike the several kernel based alternatives. The interpretability arises from the gating architectures, which informs us which "specific" expert is responsible for the prediction given the input.
- Being parametric models, they are faster at both train and test times. While training Kernel based models is hard, they usually scale in the size of training data at test time as well (though approximate inference is possible).

There are still a few shortcomings for mixture-of-experts models though.

- As alluded to earlier, the requirement that experts be probabilistic excludes powerful experts like SVMs and Neural Networks (NN). Approximate procedures might be possible, but this usually complicates and slows down inference. There are successful application with such non-probabilistic experts, for example the very recent [20]. However, such applications generally resort to heuristics for training.
- Even with probabilistic experts, having closed form updates in E and M steps is extremely rare. Even in rudimentary setups, such as a softmax gating network and probabilistic linear regressors as experts, the training procedure has to resort to iterative procedures, like Iteratively Reweighted least Squares (IRLS) or Gradient Descent, within the EM iterations [8]. A shining exception, developed to avoid this very limitation, is [23]. While we will discuss these models again in the upcoming sections, the embedded iterative procedures are not only slow, they increase the hyperparameters to be tuned and often converge sub-optimally.

In this work, we try to address a few of the shortcomings, in particular the second shortcoming. We leverage some of the advances in probabilistic formulation using latent-variable augmentation techniques, which are described next.

1.2 Gaussianizing Objectives using Latent-Variable Augmentation

Gaussian distribution is arguably the most recurring entity in Probabilistic Machine Learning, and Statistics in general. Besides being a useful modelling tool and having Central Limit Theorem to back it up, Gaussians have well understood properties, are easily manipulatable, are self-conjugate and often yield closed form solutions. Thus, we often try to manipulate likelihood functions, which are either intractable otherwise or inherently non-probabilistic, into Gaussian likelihoods using latent variable augmentation. One classic example is Gaussian Mixture Model (GMMs). Recall that to learn GMMs, latent variables are introduces such that likelihood for every data point becomes a Gaussian. This considerably simplifies the otherwise intractable inference. This technique of "Gaussianizing" is at the heart of [23], the first work to provide closed form updates in both E and M steps. They essentially switch to GMM like generative gating network which allows them to make the Gating Network conjugate with Probabilistic Linear Regression experts, which were modelled as Gaussians themselves. For our work, we will leverage two such recently introduced latent-variable augmentation techniques.

1.2.1 Bayesian Support Vector Machines

Support Vector Machines [3] (SVM) became the de-facto blackbox algorithm in machine learning due to their immense efficacy on binary classification tasks. The kernel trick [17] and fundamental extensions beyond binary classification [5, 21] perpetuated their popularity. However, there are some drawbacks to the SVM formulation. Being a discriminative model, there are no obvious ways to choose a kernel and tune its hyperparameters without cross validation, which can be computationally expensive. This difficulty, along with typical shortcomings of a discriminative model, can be countered easily with a Bayesian formulation. **Bayesian SVM**, proposed in [15], provided the aforementioned Bayesian formulation for SVMs using data augmentation.

Formally, assume a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$. Our aim is to learn a weight vector w such that the objective

$$\mathcal{L}(w, R) = \sum_{i=1}^{N} \max(1 - y_i x_i^T w, 0) + R(w) \tag{2}$$

is minimized. Here, $R(w)$ represents some regularizer. Assume $R(w) = \frac{\lambda||w||^2}{4}$, although the formulation extends to much more general regularizers, as described in [15]. Minimizing (2) is equivalent to finding the mode of $p(w|\lambda, \mathcal{D}) \propto p(w|\lambda) \prod_{i=1}^N L(y_i|w, x_i) \propto \exp(-2\mathcal{L}(w, R))$ for the following definitions:

$$p(w|\lambda) = \mathcal{N}(0, \lambda^{-1} I) \tag{3}$$

$$L(y_i|w, x_i) = \exp(-2\max(1 - y_i x_i^T w, 0)) \tag{4}$$

$p(w|\lambda)$ represents the prior and $L(y_i|w, x_i)$ defines the pseudo-likelihood. The fundamental contribution of [15] was to decompose (4) into a location-scale mixture of normals. To be more precise, [15] shows that:

$$L(y_i|w, x_i) = \int_0^\infty \frac{1}{\sqrt{2\pi\gamma_i}} \exp(-\frac{1}{2}\frac{(1 + \gamma_i - y_i x_i^T w)^2}{\gamma_i}) d\gamma_i \tag{5}$$

(5) inspires the data augmentation scheme. Essentially, γ_i is augmented for every data point x_i, which converts the likelihood function into a Gaussian distribution (hence, the "Gaussianization"). Therefore, define $L(y_i, \gamma_i|w, x_i) = \mathcal{N}(1 - y_i x_i^T w_i| - \gamma_i, \gamma_i)$. Now, with augmentation, we want to maximize/estimate the posterior $p(w, \gamma|\lambda, \mathcal{D}) \propto p(w|\lambda) \prod_{i=1}^N L(y_i, \gamma_i|w, x_i)$. Since γ itself is unknown, we again resort to tools like **Expectation Maximization** (EM) or **Markov Chain Monte Carlo** (MCMC) methods, as has been show in [13, 15]. [13], in particular, goes onto show multiple extensions for Bayesian SVMs such as Multiclass SVMs, Nonlinear Kernel SVMs and Support Vector Regression.

1.2.2 Polya-Gamma Augmentation

Bayesian analysis has long utilized the probit model for analysis of binary data. This can be attributed to a simple latent variable augmentation [1]. However, such a counterpart was missing for a long time for logistic regression models. Consider the following logistic regression model for binary data $y_i \in \{0, 1\}$ for feature vector $x_i \in \mathbb{R}^D$,

$$y_i \sim \text{Bern}(w_i) \tag{6}$$

$$w_i = \frac{1}{1 + \exp(-x_i^T w)} \tag{7}$$

where $w \in \mathbb{R}^D$ is the weight vector to be inferred. Due to the non-conjugacy of the likelihood model, a Bayesian analysis is extremely inefficient. To solve this problem, [14] introduced the Polya-Gamma latent-variable augmentation technique, which essentially transformed the logistic regression likelihood model to a Gaussian likelihood, thus enabling an efficient and full Bayesian treatment of Logistic Regression. Later, [18] showed how to use the Polya-Gamme latent-variable routine in an EM algorithm. The fundamental result shown by [14] was:

$$\frac{\left(e^\psi\right)^a}{\left(1 + e^\psi\right)^b} = 2^{-b} e^{(a-b/2)\psi} \int_0^\infty e^{-\beta\psi^2/2} p(\beta) d\beta \tag{8}$$

where $p(\beta)$ represents the density of $\beta \sim PG(b, 0)$ and PG stands for Polya-Gamma distribution. Clearly, if β is "known", the logit likelihood (for $a = b = 1$) will become a Gaussian Likelihood Model. Thus, we can derive a EM routine which assumes β as latent variables (alternately, a MCMC routine, in particular Gibbs Sampling, if required).

Leveraging the latent-variable augmentation schemes described above, we will now present some novel Mixture of Experts architectures, all of which have efficient inference routines. Our architectures will extensively leverage Bayesian SVMs (largely as experts). Thus, our work can also be interpreted as an alternate way to do non-linear learning with SVMs (in contrast to Kernel based methods). These are more efficient in both train and test time, and are highly interpretable

2 Model Architectures

We propose a Mixture of Experts (MoE) based model, where the experts are Bayesian SVMs. We restrict ourselves to binary classification tasks, however, extension to multi-class classification and regression should follow in a fairly similar manner. As a general motivation for this formulation, we provide the following reasons:

- Bayesian SVMs preserve the maximum margin separation properties which made SVMs popular. Thus, Bayesian SVMs can be effective linear models solving the binary classification task for a subset of the input assigned to it by the gating network.
- The mixture-of-experts formulation leverages the probabilistic interpretation of SVMs, thus allowing an ensemble of SVMs to be constructed in a systematic and interpretable fashion.
- The models can be trained very efficiently using EM (the detailed derivations have been shifted to the appendix). In our later architectures, we show a formulation such that both E step and M step have closed form updates.

Assuming the number of experts to be K (this can be learnt from the data itself) our MoE formulation uses K Bayesian SVMs defined by the set of weight vectors $W = \{w_i\}_{i=1}^{K}$ where w_k denotes the weight vector of the k^{th} Bayesian SVM expert. Following Eq. (5), given the latent variable $z_i \in \{1, \ldots, K\}$ denoting which expert the the input x_i has been assigned to, the Bayesian SVM conditional probability $p(y_i|x_i, z_i = k, \gamma_{ik})$ of the label y_i, given input x_i can be written as

$$p(y_i|x_i, w_k, \gamma_{ik}) = \mathcal{N}(1 - y_i w_k^T x_i| - \gamma_{ik}, \gamma_{ik}) \tag{9}$$

The expectations of the latent variables z_{ik} and γ_{ik}, given the current estimates $\hat{W}$ and $\hat{V}$ of MoE model parameters, can be computed in closed form [15] as follows:

$$\eta_{ij} = \mathbb{E}[z_{ij}] \propto p(y_i|\hat{w}_k, x_i, \gamma_{ij}) p(z_{ij} = 1|x_i, \hat{V}) \tag{10}$$

$$\tau_{ik} = \mathbb{E}[\gamma_{ik}^{-1}] = |1 - y_i \hat{w}_k^T x_i|^{-1} \tag{11}$$

Note that, in (10), the form of $p(z_{ik} = 1|x_i, \hat{V})$ will depend on the architecture of the gating network. The exact likelihood construction, updates for parameters and latent variables have been discussed in the respective sections (and appendices). Next, we will discuss the architectures for gating networks. The following discussion will be restricted to Flat Mixture of Experts, which will be followed up by a relevant construction of Hierarchical Mixture of Experts.

2.1 Flat Mixture of Experts

2.1.1 Naive Softmax Gating Network

For the first model, we follow [8] and construct a softmax gating network. The experts as stated before, are Bayesian SVMs. For a setting with K experts, $V = \{v_i\}_{i=1}^{K}$ denotes the gating network matrix. The probability of expert k is given by $\pi_k(x) = \frac{\exp(v_i^T x)}{\sum_{l=1}^{K} \exp(v_l^T x)}$ The detailed derivations, along with updates, are provided in Appendix A.1. There is one major drawback in this approach: *There are no closed form updates for gating network.* This point is emphasized in the derivation as well. This arises because of the use of softmax gating network, which couples all the gating vectors. This was identified in [8], which uses Iterative Reweighted Least Squares (IRLS) to solve for gating network. We propose to use the simpler gradient descent to solve for the gating network, the required gradient derivation for which has been provided. The problem is slightly alleviated by the fact that we do not have to train the softmax gating to convergence. For EM to converge, we only need to take a few gradients steps to make our 'softmax' network better. Thus, we only take a fixed number of steps along the direction of the gradient in every maximization step. Nonetheless, an iterative procedure within an iterative procedure is not a desirable feature of this model, as it increases the number of hyperparameters to tune.

2.1.2 Generative Gating Network

The problem of having an iterative procedure (IRLS for softmax gates) within an iterative procedure (EM) was identified and solved in [23] using the generative gates. In context of our formulation, we make the following changes:

- *Use generative gates*: Instead of using the softmax function to map the input to an expert, we use a generative model (similar in spirit to Gaussian Mixture Models) to decide to the expert. In particular, the probability of j^{th} expert being assigned any input x is given by $p(j|x, V) \propto \alpha_j p(x|j, V)$ where α_j is the prior probability and $p(x|j, V) = \mathcal{N}(\mu_j, \Sigma_j)$. Therefore, the new set of parameters for the gating network is $V = \{\alpha_k, \mu_k, \Sigma_k\}_{k=1}^K$.
- *Maximize* $\mathbb{E}_{p(\gamma,z|W^{(t)},V^{(t)},D)}[\log p(y, x, \gamma, z|V, W)]$: Maximizing the original objective will again require an iterative procedure for the M step. A new objective is proposed in this model, which when used, provides closed form updates for both the E and M step.

For a detailed discussion on why the above two steps are required, refer to [23]. The derivations and the updates are provided in the Appendix section A.2.1.

2.1.3 Pólya-Gamma Augmented Softmax Gating Network

We revert back to the softmax gating network. However, in this construction we will introduce Pólya-Gamma augmentation to the gating network vectors, which will allow us to get closed form updates for the weight vectors $V = \{v_k\}_{k=1}^K$, where each $v_i \in \mathbb{R}^D$. The expert assignment probability is again $\pi_k(x_i) = p(z_{ik} = 1|x_i, V) = \frac{\exp(v_k^T x_i)}{\sum_{l=1}^K \exp(v_l^T x_i)}$. For training input x_i, we will augment latent variables $\beta_{ij} \sim PG(0, 1)$ where PG stands for Pólya-Gamma distributions and $j \in \{1, \dots K\}$. Consequently, using the results from [14], we get

$$\log p(z_{ik} = 1|x_i, \beta_{ik}, V) \propto \frac{\psi_{ik}}{2} - \beta_{ik}\frac{\psi_{ik}^2}{2} \tag{12}$$

$$\mathbb{E}[\beta_{ik}] = 0.5\psi_{ik}^{-1}\tanh(0.5\psi_{ik}) \tag{13}$$

where $\psi_{ik} = x_i^T v_k - \log \sum_{l=1, l\neq k}^K \exp(x_i^T v_l)$. The detailed derivation has been given in section A.2.2.

2.1.4 Logistic Stick Breaking Prior Gating Network with Pólya-Gamma Augmentation

The fourth gating network is based on a logistic stick-breaking prior [16] (LSBP). One of the appealing properties of this construction for the gating network is that it enables us to learn the "right" number of experts from data and the stick breaking prior provides a non-parametric construction for the same [16]. Our LSBP construction specifies a large-enough truncation level on the number of experts and the model has the ability to determine the number of experts by automatically pruning the unnecessary experts, as warranted by the data. The LSBP gating network construction defines the probability of an input x_i assigned to an expert k as follows:

$$\pi_k(x_i) = \nu_k(x_i)\prod_{l=1}^{k-1}(1 - \nu_l(x_i)) \tag{14}$$

where $\nu_k(x_i) = \frac{1}{(1+\exp(-v_k^T x_i))}$. Note that the LSBP construction is somewhat akin to the softmax due to the way the stick weights $\nu_k(x_i)$ are defined. As a result, estimating the model parameter does not have a closed form, again requiring iterative methods. However, we can again leverage the Pólya-gamma latent variable augmentation described and used earlier. Therefore, for this construction, we again introduce a set of latent variables $\beta = \{\beta_{ik}\}_{i=1,k=1}^{i=N,k=K}$ where $\beta_{ik} \sim PG(b, 0)$. The set of parameters to be learned in the LSBP gating network are $V = \{v_i\}_{i=1}^K$, and

$$\log p(z_{ik} = 1|x_i, \beta_i, V) = \frac{v_j^\top x_i}{2} - \sum_{l=1}^{k-1}\frac{v_l^\top x_i}{2}\sum_{l=1}^{k}\beta_{il}\frac{(v_l^\top x_i)^2}{2} \tag{15}$$

This expression is substituted in the expression of the complete data log-likelihood. Just like the softmax case, here again, the required expectation of PG variables are available in closed form [14] and are given by

$$\chi_{ik} = \mathbb{E}[\beta_{ik}] = (2\hat{v}_k^T x_i)^{-1}\tanh(0.5\hat{v}_k^T x_i) \tag{16}$$

The parameter updates, resulting from maximization of expected complete data log-likelihood, are given by:

$$\begin{aligned}
\hat{v}_k &= (X^\top \Omega_k X + \rho I)^{-1} X^\top (\hat{\kappa}_{1k}, \dots, \hat{\kappa}_{Nk})^T \\
\hat{\kappa}_{ik} &= \eta_{ik} - 0.5\sum_{l=k}^{K}\eta_{il} \\
\hat{\Omega}_k &= \text{diag}(\chi_{1k}\sum_{l=k}^{K}\eta_{1l}, \dots, \chi_{Nk}\sum_{l=k}^{K}\eta_{Nl})
\end{aligned} \tag{17}$$

Here, ρ is the hyperparameter from the prior distribution on ν_k. The derivations largely resemble those performed in [16], and therefore are skipped here from the discussion for brevity.

2.2 Hierarchical Mixture of Experts

We will now shift our attention to Hierarchical Mixture of Experts (HMoE). HMoEs were first presented in [9], as an alternate to Generative Gating/Softmax gating based Flat Mixture of Experts. As is the convention, we will predefine the structure of the HMoE model to be full binary tree, as is the case in [9]. The leaf nodes in such a tree correspond to the "expert" nodes of a Flat MoE, where the rest of the tree acts as a gating network. There is a weight vector associated with every node in the tree. In this work, since we are dealing with binary classification tasks, every node of the tree essentially has a binary classification problem. The internal nodes decide which route to take (and hence the expert), whereas the final experts decide the final label. Ofcourse, all of this is done in a probabilistic fashion and therefore, each expert gets a weight depending upon the probability assigned to that path by the internal nodes. Since, all the nodes essentially have a binary classification task, this gives us an opportunity to convert all nodes in the tree to a Bayesian SVM! Up until now, we had restricted the Bayesian SVM formulation to the experts. However, in HMoE, we can even construct internal nodes/gating network using Bayesian SVMs. This Bayesian SVM construction would be valid even if the tree was not binary, as we can always use a multi-class version of Bayesian SVM [13].

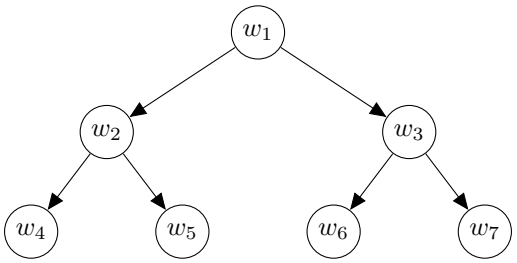

Figure 2: An instance of HMoE

The notation for HMoEs can get extremely cumbersome. Therefore, for simplicity of discussion assume that the HMoE has the structure as drawn in Figure 2. Also, assume that all classification labels are $\{0, 1\}$, contrary to the previous discussions where the labels were $\{-1, +1\}$. Here, w_1, w_2, w_3 play the role of "gating network", whereas w_4, w_5, w_6, w_7 are the experts. Here, $W = \{w_i\}_{i=1}^{7}$ play a dual role, where they identify the node in the tree as well as represent the weight vector. At test time, if an internal node predicts 0, we go left, otherwise we go right. If we do things in a probabilistic fashion, we get the following recursive probability rule for every expert [Note the divisions are assumed to be integer divisions for the rest of the discussion]:

$$p(z_{ij} = 1|x_i, W) \propto p(j\%2|x_i, w_{i\frac{j}{2}})p(z_{i\frac{j}{2}} = 1|x_i, W) \tag{18}$$

Here, $z_{ij} = 1$ denotes the event that j^{th} node was chosen along the path to an expert. Also, $p(y|x_i, w_j) \propto \exp(-2\max(0, 1 - 2(y-1)w_j^T x_i))$, where the final probability is computed after normalization with all other nodes at that level.

Clearly, the variable z_{ij} are unknown. Therefore, these are assumed to be latent for inference. Typically, a Bayesian SVM weight vector is augmented with one latent variable for every example. Here, every internal node of the tree is augmented with two latent variables per example, that is $\gamma_{ij}^0, \gamma_{ij}^{+1}$ for i^{th} training example and j^{th} weight vector where $j \in \{1, 2, 3\}$. This peculiarity happens because in all previous constructions the output label was known with certainity, whereas the "true" path for every example is latent, and can only be assigned certain probability (as is typical in EM). Therefore, the internal nodes predict both the labels with those probabilities and therefore, two augmentations are needed for every input in training data. Note, the leaf nodes only have one augmentation per training data point, that is γ_{ij} for $j \in \{4, 5, 6, 7\}$, because the true labels are known. Now, define

$$\gamma_i^{(k)} = \{\gamma_{ij}^p | p \in \{0, 1\}, j \in [2^k, 2^{k+1} - 1] \subset \mathcal{Z}\} \tag{19}$$

Essentially, $\gamma_i^{(k)}$ refers to the set of Bayesian SVM latent variables at level k, where root node is level 0. Now,

again the following recursive definitions hold:

$$p(z_{ij}=1,\gamma_i^{\lceil\log(\frac{j}{2})\rceil}|x_i,W)=p(j\%2,\gamma_{i\frac{j}{2}}^{j\%2}|w_2,x_i)p(z_{i\frac{j}{2}}=1,\gamma_i^{\lceil\log(\frac{j}{4})\rceil}|x_i,W) \tag{20}$$

$$\eta_{ij}=\mathbb{E}[z_{ij}]=p(z_{ij}=1|y_i,x_i,W^{(t)})\propto p(y_i|x_i,w_j^{(t)})p(z_{ij}=1|x_i,W^{(t)}) \tag{21}$$

Now, recall that expected complete log-likelihood function looks as the follows:

$$\begin{aligned}\mathcal{L}(y,Z,\gamma|X,W)&=\log p(y,\gamma,\beta,Z|X,W)\\&=\sum_{i=1}^{N}\sum_{j=4}^{7}\mathbb{1}[z_{ij}=1]\Big(\log p(y_i,\gamma_{ij}|w_j,x_i)+\log p(z_{ij}=1,\gamma_i^{(1)}|x_i,W)\Big)\end{aligned} \tag{22}$$

We can use the recursive formulas stated above to compute $\mathbb{E}_{z,\gamma}[\mathcal{L}]$. Before moving forward, let's look at the terms which depend on w_2.

$$\mathcal{L}(w_2)=\sum_{i=1}^{N}\eta_{i4}\log p(-1,\gamma_{i2}^{(0)}|x_i,w_2)+\eta_{i5}\log p(+1,\gamma_{i2}^{(1)}|x_i,w_2) \tag{23}$$

In fact, this can be generalized for all w_i. Let η_i denote the sum of η_{ij} for all expert nodes under that node on turning left (for example w_4 for w_2) and η_i' denote the sum of η_{ij} for all expert nodes on turning right. This definition holds for all w_i. Also define,

$$\tau_i^{(0)}=\mathbb{E}[\big(\gamma_i^{(0)}\big)^{-1}]^{-1}=|1+x_i^Tw| \tag{24}$$

$$\tau_i^{(1)}=\mathbb{E}[\big(\gamma_i^{(1)}\big)^{-1}]^{-1}=|1-x_i^Tw| \tag{25}$$

Such an abstraction simplifiest the results. Therefore,

$$\mathcal{L}(w)=\sum_{i=1}^{N}\sum_{i=1}^{N}\eta_i\log p(-1,\gamma_i^{(0)}|x_i,w)+\eta_i'\log p(+1,\gamma_i^{(1)}|x_i,w) \tag{26}$$

Now, solving this expression, we get the following updates (assuming a normal prior with 0 mean and covariance λI):

$$w=(X^TAX+\lambda I)^{-1}(\sum_{i=1}^{N}b_ix_i) \tag{27}$$

$$A=\text{diag}(a_1,a_2,\dots a_n) \tag{28}$$

$$a_i=\frac{\eta_i}{\tau_i^{(0)}}+\frac{\eta_i'}{\tau_i^{(1)}} \tag{29}$$

$$b_i=(\eta_i'-\eta_i)+(\frac{\eta_i'}{\tau_i^{(1)}}-\frac{\eta_i}{\tau_i^{(0)}}) \tag{30}$$

Note that these updates reduce to the standard updates in other architectures for the leaf/expert nodes as either $\eta_i'=0$ or $\eta_i=0$ for all leaf nodes.

With these updates, the algorithm is a forward pass-backward pass kind of algorithm. In the forwards, all the expectations are computed (E Step). In the backward pass, all the updates are executed (M Step). This is also reflected in the recursive nature of updates for the E Step.

3 Experiments and Results

We experiment on several benchmark binary classification datasets comparing with several state-of-the-art mixture of experts models. We consider 3 sets of benchmark datasets, based upon the recent relevant MoE models we compare our model with. A brief descriptions of these datasets, compared models and experiment settings are as follows:

- ***Set I:*** We first compare the results with the recently proposed MoE models, including the convex polytope machine (CPM), stack-softplus and sum-stacked-softplus classification models proposed by [25]. They propose a non-linear classifier by introducing a family of softplus functions, convolving countably infinite stacked gamma distributions, and report state-of-the-art results on several benchmark datasets, including

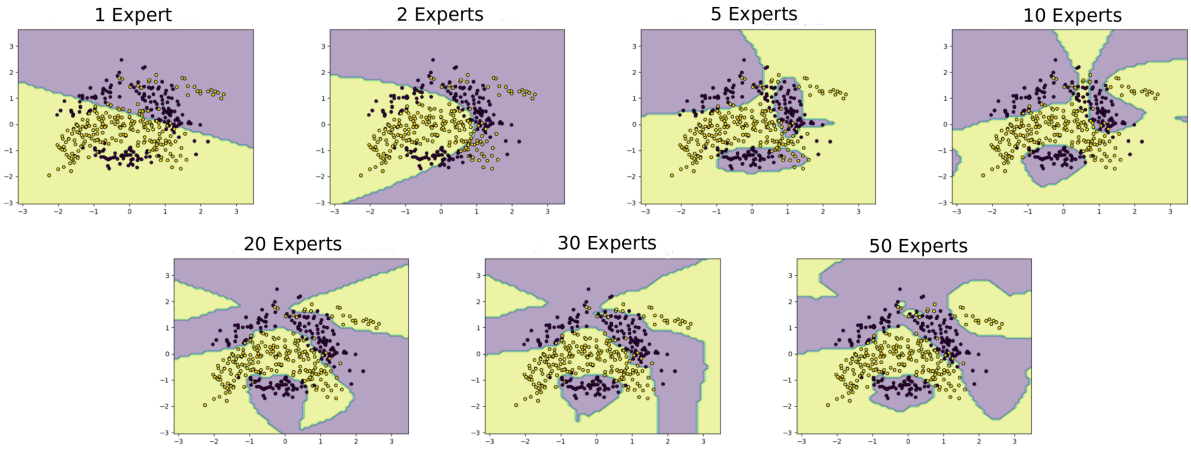

Figure 3: Decision Boundary learnt by our model as the number of experts are increased from 1 to 50 (left to right). Larger number of experts are able to generate increasingly non-linear decision boundary for the banana dataset.

Dataset	RBF-SVM	RVM	CPM	SS-τ	GG*	LSBP*	PG*	HMoE
banana	10.85 (0.57)	11.08 (0.69)	21.39 (1.72)	11.89 (0.61)	**10.60 (0.41)**	11.53 (0.91)	25.45 (5.3)	15.59 (3.89)
waveform	10.73 (0.86)	11.16 (0.72)	12.76 (1.17)	11.69 (0.69)	9.41 (1.5)	19.1 (1.6)	**8.93 (0.31)**	12.19 (1.05)
image	2.84 (0.52)	3.82 (0.59)	3.25 (0.41)	**2.73 (0.53)**	3.1 (0.45)	3.65 (0.94)	10.43 (3.06)	6.49 (1.43)
breast cancer	28.44 (4.52)	31.56 (4.66)	32.08 (4.29)	28.83 (3.40)	**21.04 (1.91)**	21.56 (3.01)	23.12 (4.07)	24.91 (4.55)

Table 1: Comparison on datasets with numbers reported by [25]

  - **banana**: This a 2-dimensional dataset that has a highly non-linear decision boundary. We qualitatively show how increasing number of expert are able to generate an increasingly non-linear decision boundary in Figure 3.
  - **Waveform, image, breast cancer** These datasets along with banana dataset are taken from [6] (as is done in [25]).

  These datasets have multiple predefined test and train splits. We use the first 10 splits and then report results on them. The results are summarized in Table 1.

- ***Set II*:** We also directly compare with a Bayesian nonlinear SVM model [7] on the datasets reported in their model. In their work, they propose a non-linear Bayesian SVM (BSVM) which use nonlinear kernels, as compared to our proposed Mixture-of-Experts model for classification. We also compare with the Gaussian Process based Classifier (GPC) as mentioned in their paper. For this set, the following datasets are considered:

  - **Pima Diabetes Dataset** It consists of Yes/No responses to the presence of diabetes in the patients based upon 8 features.
  - **Sonar Dataset** It consists of 208 instances where Sonar rays are bounced off ground to detect rocks and mines.
  - **Wisconsin Dataset** It consists of 682 instances of patients who are tested for breast cancer.

  We use the same experimental conditions as the authors, dividing the dataset into 10 parts for a 10-Fold Validation and then report the mean accuracies and std deviation.

- ***Set III:*** In [4], they proposed a technique to use an optimal number of support vectors while doing non-linear classification with Kernels. Hence they aim to reduce the test times of their model with lesser support vectors. While our model learns a set of sufficient statistics of the model and plugs them in at test times. We call their model OSSVM for brevity.

  - **Adult Dataset and IJCNN** They are the a8a and ijcnn1 dataset available on LIBSVM tools.

  Both of these dataset have predefined test and training datasets on which we report the test error as done in [4].

- ***Set IV*** Finally, we also consider the Small Variance Dirichlet Process Mixture SVM (M^2DPM) proposed by [22]. We use one of the two real datasets used by them, the Parkinsons disease dataset

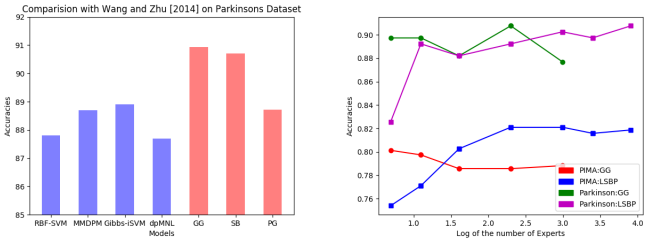

Figure 4: Left: Accuracies on the Parkinsons Dataset when compared with the Small Variance Dirichlet Process Mixture SVM (M^2DPM) proposed by [22] and allied models mentioned by them. Right : Plot showing that the Stick Breaking Prior is indeed learning the number of experts as the accuracy stabilizes if the number of experts is large enough. Legend format - Dataset:Model

which consists of 195 subjects that may or may not have the disease. For this too, we use a 10 Fold Cross Validation and report the mean and standard deviation

Model	Adult(a8a)	IJCNN
OSSVM	15	4
	14.75	**1.8**
SpSVM	14.6	6
	14.6	2.5
RSVM	14.65	8
	14.6	6
GG*	**14.49**	2.02
LSBP*	20.02	-
PG*	14.81	8.04
HMoE	15.30	4.36

Table 2: For OSSVM , SpSVM, RSVM we have two rows to report the errors with 1000 and 100 support vectors, upper and lower respectively. Their errors are reported by reading the graph in [4] hence may be off a little.

References

[1] James H Albert and Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

[2] Christopher M Bishop and Markus Svenskn. Bayesian hierarchical mixtures of experts. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann Publishers Inc., 2002.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[4] Andrew Cotter, Shai Shalev-Shwartz, and Nati Srebro. Learning optimally sparse support vector machines. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 266–274, 2013.

[5] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

Model	PIMA	Wisconsin	Sonar
RBF-SVM	24.22	3.07	11.54
GPC	22.01	2.64	12.50
BSVM	21.88	2.93	11.06
GG*	19.9 (4.68)	**1.7**	**7.73 (5.85)**
LSBP*	**17.7 (5.55)**	2.23	10.00 (6.54)
PG*	21.1 (3.9)	2.23	14.1 (7.02)

Table 3: Comparing with the GP Mixture model of [7]

[6] T. Diethe. 13 benchmark datasets derived from the UCI, DELVE and STATLOG repositories. https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets/, 2015.

[7] Ricardo Henao, Xin Yuan, and Lawrence Carin. Bayesian nonlinear support vector machines and discriminative factor modeling. In *Advances in Neural Information Processing Systems*, pages 1754–1762, 2014.

[8] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[9] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[10] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[11] Edward Meeds and Simon Osindero. An alternative infinite mixture of gaussian process experts. In *Advances in Neural Information Processing Systems*, pages 883–890, 2006.

[12] Hien D Nguyen and Faicel Chamroukhi. An introduction to the practical and theoretical aspects of mixture-of-experts modeling. *arXiv preprint arXiv:1707.03538*, 2017.

[13] Hugh Perkins, Minjie Xu, Jun Zhu, and Bo Zhang. Fast parallel svm using data augmentation. *arXiv preprint arXiv:1512.07716*, 2015.

[14] Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.

[15] Nicholas G Polson, Steven L Scott, et al. Data augmentation for support vector machines. *Bayesian Analysis*, 6(1):1–23, 2011.

[16] Lu Ren, Lan Du, Lawrence Carin, and David Dunson. Logistic stick-breaking process. *Journal of Machine Learning Research*, 12(Jan):203–239, 2011.

[17] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[18] James G Scott and Liang Sun. Expectation-maximization for logistic regression. *arXiv preprint arXiv:1306.0040*, 2013.

[19] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[21] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[22] Yining Wang and Jun Zhu. Small-variance asymptotics for dirichlet process mixtures of svms. 2014.

[23] Lei Xu, Michael I Jordan, and Geoffrey E Hinton. An alternative model for mixtures of experts. In *Advances in neural information processing systems*, pages 633–640, 1995.

[24] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[25] M. Zhou. Softplus Regressions and Convex Polytopes. *ArXiv e-prints*, August 2016.

A Appendix

A.1 MAP Estimate for a Mixture of Bayesian SVM Experts

Let $\Theta = \{\theta_g, \theta_e\}$ where $\theta_g = \{v_i\}_{i=1}^K$ represents the vectors for gating network of K experts and $\theta_e = \{w_i\}_{i=1}^K$ represent the weight vectors for K Bayesian SVMs, $\gamma_i = \{\gamma_{i1}, \gamma_{i2}, \dots \gamma_{iD}\}$ represent the augmented representation for the i^{th} Bayesian SVM construct and z represent the set of one-hot vectors allotting each input to an expert. $D = \{X, y\}$ denotes the training data as usual.

Here, γ and z act as the latent variables. As is the case with EM, instead of maximizing $p(\Theta|D)$, we maximize $\mathbb{E}[\log p(\Theta, \gamma, z|D)]$ where expectation is with respect to $p(\gamma, z|\Theta^{(t)}, D)$. Now,

$$
\begin{aligned}
p(\Theta, \gamma, z|D) &\propto p(\Theta)p(y, \gamma, z|X, \Theta) \\
\Rightarrow \log p(\Theta, \gamma, z|D) &\propto \log p(\Theta) + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1](\log p(z_{ij} = 1|x_i, \theta_g) + \log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)) \\
\Rightarrow \mathbb{E}[\log p(\Theta, \gamma, z|D)] &\propto \log p(\Theta) + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{E}[z_{ij}](\log p(z_{ij} = 1|x_i, \theta_g) + \mathbb{E}[\log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)]) \quad (31)
\end{aligned}
$$

There is an implicit assumption in this: The posterior expectation of z_{ij} does not depend on γ. In the current setup, we are using the softmax gating. Using (4), we get:

$$
\begin{aligned}
\mathbb{E}[z_{ij}] &= p(z_{ij} = 1|x_i, y_i, \Theta^{(t)}) \\
&\propto p(y_i|x_i, z_{ij} = 1, \Theta_e^{(t)})p(z_{ij}|x_i, \Theta_g^{(t)}) \\
&\propto \exp(-2\max(0, 1 - y_i x_i^T w_j^{(t)}))\exp(x_i^T v_j^{(t)}) \\
\Rightarrow \mathbb{E}[z_{ij}] &= \frac{\exp(x_i^T v_j^{(t)} - 2\max(0, 1 - y_i x_i^T w_j^{(t)}))}{\sum_{l=1}^{K}\exp(x_i^T v_l^{(t)} - 2\max(0, 1 - y_i x_i^T w_l^{(t)}))} = \eta_{ij} \quad (32)
\end{aligned}
$$

Now, onto the other expectation:

$$
\begin{aligned}
\mathbb{E}[\log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)] &= \int_0^\infty \log p(y_i, \gamma_{ij}|w_j, x_i, y_i)p(\gamma_{ij}|x_i, w_j^{(t)}, y_i)d\gamma_{ij} \\
&= \int_0^\infty \log[\frac{1}{\sqrt{2\pi\gamma_{ij}}}\exp(-\frac{(1 + \gamma_{ij} - y_i w_j^T x_i)^2}{2\gamma_{ij}})]p(\gamma_{ij}|x_i, w_j^{(t)}, y_i)d\gamma_{ij}
\end{aligned}
$$

Here,

$$
\log[\frac{1}{\sqrt{2\pi\gamma_{ij}}}\exp(-\frac{(1 + \gamma_{ij} - y_i w_j^T x_i)^2}{2\gamma_{ij}})] = \frac{-1}{2}\log 2\pi - \frac{1}{2}\log\gamma_{ij} - \frac{(1 + \gamma_{ij})^2}{2\gamma_{ij}} - \frac{(y_i w_j^T x_i)^2}{2\gamma_{ij}} + \frac{y_i w_j^T x_i(1 + \gamma_{ij})}{\gamma_{ij}}
$$

We only care about the terms involving w_j, that is, only the last two terms. Therefore, we only care about the expectation $\mathbb{E}[\frac{1}{\gamma_{ij}}] = |1 - y_i x_i^T w_j^{(t)}|^{-1} = \tau_{ij}^{-1}$ [15]. Therefore, the $\mathbb{E}[\log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)]$ can be replaced by $\frac{-(y_i w_j^T x_i)^2 + 2y_i w_j^T x_i}{2\tau_{ij}} + 2y_i w_j^T x_i$ in (31) to get the final objective (note we are ignoring the terms not involving w_j). Replacing the appropriate expectations in (31), the final objective for the problem is:

$$
\mathcal{L}(\Theta, \Theta^{(t)}) = \log p(\Theta) + \sum_{i=1}^{N}\sum_{j=1}^{K}\eta_{ij}\Big[\log\frac{\exp(v_j^T x_i)}{\sum_{l=1}^{K}\exp(v_l^T x_i)} - \frac{(y_i w_j^T x_i)^2 - 2y_i w_j^T x_i}{2\tau_{ij}} + 2y_i w_j^T x_i\Big] \quad (33)
$$

We maximize this objective, that is, $\Theta^{(t+1)} = \text{argmax}_\Theta\, \mathcal{L}(\Theta, \Theta^{(t)})$. Now, assume a zero mean gaussian prior for weight vectors of both softmax gating and expert weight vectors, that is, $p(w_i) \sim \mathcal{N}(0, \lambda^{-1}I)$ and $p(v_i) \sim \mathcal{N}(0, \beta^{-1}I)$. Note, $\frac{\partial \log p(w_i)}{\partial w_i} = -\lambda w_i$ and similarly, $\frac{\partial \log p(v_i)}{\partial v_i} = -\beta v_i$. Now, take derivatives of the objective $\mathcal{L}$ with respect to w_j, v_j. Therefore,

$$
\begin{aligned}
\frac{\partial\mathcal{L}(\Theta, \Theta^{(t)})}{\partial w_j} &= 0 \\
\Rightarrow (\lambda w_j + \sum_{i=1}^{N}\frac{\eta_{ij}}{\tau_{ij}}(w_j^T x_i)x_i) &= \sum_{i=1}^{N}\eta_{ij}(\frac{y_i x_i}{\tau_{ij}} + y_i x_i) \quad (34) \\
\Rightarrow w_j &= (X^T A_j X + \lambda I)^{-1}(\sum_{i=1}^{N}\eta_{ij}\frac{\tau_{ij} + 1}{\tau_{ij}}y_i x_i) \quad (35)
\end{aligned}
$$

where X represents the data matrix and $A_j = diag(\frac{\eta_{1j}}{\tau_{1j}}, \dots \frac{\eta_{Nj}}{\tau_{Nj}})$. As expected, we get a closed form update for mixture of Bayesian SVMs. However, we cannot get a closed form update for the softmax gating parameters, which needs to resort to iterative update methods. If using gradient descent, the following gradient is used:

$$\begin{aligned}\frac{\partial \mathcal{L}(\Theta, \Theta^{(t)})}{\partial v_j} &= -\beta v_j + \sum_{i=1}^{N} \eta_{ij} x_i - \sum_{i=1}^{N}\sum_{j=1}^{K} \eta_{ij} \frac{\partial \log \sum_{l=1}^{K} \exp(v_l^T x_i)}{\partial v_j} \\ &= \sum_{i=1}^{N} \Big[\eta_{ij} - \frac{\exp(v_j^T x_i)}{\sum_{l=1}^{K} \exp(v_l^T x_i)}\Big] x_i - \beta v_j \end{aligned} \tag{36}$$

A.2 Towards Closed Form Updates

Two approaches are discussed which will allow closed form updates for all the parameters of the model. The first approach changes the gating network from a discriminative softmax to a generative network [23], while the second approach uses the 'Polya-Gamma Data Augmentation' for the softmax weight vectors [14, 18].

A.2.1 Generative Gating Networks

The results derived in this section borrows some results from [23], and the previous section. We wish to maximize $\mathbb{E}_{\log p(\gamma, z|\Theta^{(t)}, D)}[p(y, x, \gamma, z|\Theta)]$ with respect to Θ. Here,

$$\begin{aligned}\log p(y, x, \gamma, z|\Theta) &= \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1]\Big(\log p(y_i, \gamma_{ij}|x_i, w_j) + \log \alpha_j + \log p(x_i|\theta_g, z_{ij} = 1)\Big) \\ \Rightarrow \mathbb{E}[\log p(y, x, \gamma, z|\Theta)] &= \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{E}[z_{ij}]\Big(\mathbb{E}[\log p(y_i, \gamma_{ij}|x_i, w_j)] + \log \alpha_j + \log p(x_i|\theta_g, z_{ij} = 1)\Big)\end{aligned} \tag{37}$$

Now,

$$\begin{aligned}\mathbb{E}[z_{ij}] &= p(z_{ij} = 1|x_i, y_i, \Theta^{(t)}) \\ &\propto p(y_i|x_i, z_{ij} = 1, \Theta^{(t)}) p(x_i|\Theta^{(t)}, z_{ij} = 1) p(z_{ij} = 1|\Theta^{(t)}) \\ &\propto \exp(-2\max(0, 1 - y_i x_i^T w_j^{(t)})) \mathcal{N}(x_i|\mu_j^{(t)}, \Sigma_j^{(t)}) \alpha_j^{(t)} \\ &= \frac{\exp(-2\max(0, 1 - y_i x_i^T w_j^{(t)})) \mathcal{N}(x_i|\mu_j^{(t)}, \Sigma_j^{(t)}) \alpha_j^{(t)}}{\sum_{l=1}^{K} \exp(-2\max(0, 1 - y_i x_i^T w_l^{(t)})) \mathcal{N}(x_i|\mu_l^{(t)}, \Sigma_l^{(t)}) \alpha_l^{(t)}} = \eta_{ij}^{(t)}\end{aligned}$$

The other expectation remains the same from the previous section. We can introduce a prior distribution on all the parameters to get a MAP estimate. The update for the experts remain the same (assuming the same gaussian prior on the weights). Referring to (35):

$$w_j^{(t+1)} = (X^T A_j^{(t)} X + \lambda I)^{-1} (\sum_{i=1}^{N} \eta_{ij}^{(t)} \frac{\tau_{ij}^{(t)} + 1}{\tau_{ij}^{(t)}} y_i x_i)$$

The updates for the parameters of the gating network are borrowed from [23]. The results do not assume any prior on the gating network parameters (which can be easily introduced). Note that there is an additional constraint that $\sum_{j=1}^{N} \alpha_j = 1$. The updates are as follows:

$$\begin{aligned}\alpha_j^{(t+1)} &= \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)}}{N} = \frac{N_j}{N} \\ \mu_j^{(t+1)} &= \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)} x_i}{N_j} \\ \Sigma_j^{(t+1)} &= \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)} (x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T}{N_j}\end{aligned}$$

where $N_j = \sum_{i=1}^{N} \eta_{ij}^{(t)}$. Thus, we get closed form updates in both the E and M steps.

A.2.2 Polya-Gamma Data Augmentation

Polya-Gamma augmentation is discussed in detail [14, 18]. For every example and for every weight vector, another latent variables $\beta_{ij} \sim PG(1,0)$ is augmented. Here, $PG(1,0)$ represents the Polya-Gamma distribution, and each of the K softmax weight vectors get an augmented latent variable for each of the example (that is NK latent variables). For simplicity of notation, we only consider the MLE optimization under EM (this can be easily converted to MAP estimation). We want to maximize $\mathbb{E}[\log p(y,\gamma,z,\beta|\Theta,X)]$, where the expectation is with respect to $p(\gamma,\beta,z|\Theta^{(t)},X,y)$.

$$\begin{aligned}
\log p(y,\gamma,\beta,z|\Theta,X) &= \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij}=1]\log p(y_i,z_{ij}=1,\beta_i,\gamma_{ij}|\Theta,x_i) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij}=1]\Big(\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1) + \log p(\beta_{ij},z_{ij}=1|\Theta,x_i)\Big) \\
\Rightarrow \mathbb{E}[\log p(y,\gamma,\beta,z|\Theta,X)] &= \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{E}[z_{ij}]\Big(\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] + \mathbb{E}[\log p(\beta_{ij},z_{ij}=1|\Theta,x_i)]\Big) \quad (38)
\end{aligned}$$

We have factorized the 'Expectation of a Product' into a 'Product of Expectation', that is

$$\mathbb{E}[\mathbb{1}[z_{ij}=1]\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] = \mathbb{E}[z_{ij}]\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] \quad (39)$$

This, in general, is incorrect. But, (39) has been used in all the EM derivations carried out above (the factors are slightly different). The justification for this step is as follows:

$$\begin{aligned}
\mathbb{E}[\mathbb{1}[z_{ij}=1]\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] &= \int \mathbb{1}[z_{ij}=1]\log p(y_i,\gamma_{ij}|w_j,x_i,z_{ij}=1)p(\gamma_{ij},\beta_i,z_i|\Theta^{(t)},X,y)d\gamma_{ij}d\beta_i dz_i \\
&= \int \mathbb{1}[z_{ij}=1]\log p(y_i,\gamma_{ij}|w_j,x_i,z_{ij}=1)p(\gamma_{ij},z_i|\Theta^{(t)},X,y)d\gamma_{ij}dz_i \\
&= \int p(z_{ij}=1|\Theta^{(t)},x_i,y_i)\log p(y_i,\gamma_{ij}|w_j,x_i,z_{ij}=1)p(\gamma_{ij}|\Theta^{(t)},x_i,y_i,z_{ij}=1)d\gamma_{ij} \\
&= \mathbb{E}[z_{ij}]\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)]
\end{aligned}$$

The other expectation factorizations (carried out in this and previous derivations) can be similarly justified. This factorization is possible because of the use of the indicator function. Now, let's consider the softmax without augmentation, that is

$$\begin{aligned}
p(z_{ij}=1|\theta_g,x_i) &= \frac{\exp(x_i^T v_j)}{\sum_{l=1}^{K}\exp(x_i^T v_l)} \\
&= \frac{\exp(x_i^T v_j - \log\sum_{l=1,l\neq j}^{K}\exp(x_i^T v_l))}{1+\exp(x_i^T v_j - \log\sum_{l=1,l\neq j}^{K}\exp(x_i^T v_l))} \\
&= \frac{\exp(\psi_{ij})}{1+\exp(\psi_{ij})}
\end{aligned}$$

Here, $\psi_{ij} = x_i^T v_j - \log\sum_{l=1,l\neq j}^{K}\exp(x_i^T v_l)$. Using augmentation results from [18], we get:

$$\begin{aligned}
\log p(z_{ij}=1|\theta_g,x_i,\beta_{ij}) &\propto \log[\exp(\frac{\psi_{ij}}{2})\exp(-\beta_{ij}\frac{\psi_{ij}^2}{2})] \\
&\propto \frac{\psi_{ij}}{2} - \beta_{ij}\frac{\psi_{ij}^2}{2}
\end{aligned}$$

Here, $\log p(\beta_i|\theta_g,x_i) = \log p(\beta_i)$ is ignored because it does not depend on any parameters. Therefore, the final objective can be written as

$$\mathcal{L}(\Theta,\Theta^{(t)}) = \sum_{i=1}^{N}\sum_{j=1}^{K}\eta_{ij}\Big(\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] + \frac{1}{2}\psi_{ij} - \mathbb{E}[\beta_{ij}]\frac{\psi_{ij}^2}{2}\Big)$$

The discussion from the first section applies directly over here. The results (32) and (35) are directly applicable in this derivation. Now, $\beta_{ij}^{(t)} = \mathbb{E}[\beta_{ij}] = \frac{1}{2\psi_{ij}^{(t)}}\tanh\psi_{ij}^{(t)}$ [18]. The coupling of parameters mandates the usage

of Expectation Conditional Maximization (ECM), when maximizing with respect to v_j [18]. In particular, when maximizing with respect to v_j, we assume $v_l, l \neq j$ to be fixed at their 'most recent estimates' (not necessarily $v_l^{(t)}$, as we would iterate over v_k from $k = 2$ to $k = K$). For identifiability, v_1 is fixed at $[0, 0 \dots 0]^T$ throughout iterations. Therefore,

$$\frac{\partial \mathcal{L}}{\partial v_j} = \sum_{i=1}^{N} \eta_{ij} \Big(\frac{1}{2} x_i - \beta_{ij}^{(t)} (x_i^T v_j - \log \sum_{l=1, l\neq j}^{K} \exp(x_i^T \hat{v}_l)) x_i \Big) = 0$$

$$\sum_{i=1}^{N} \eta_{ij} (\frac{1}{2} + \beta_{ij}^{(t)} \log \sum_{l=1, l\neq j}^{K} \exp(x_i^T \hat{v}_l)) x_i = \big(\sum_{i=1}^{N} \beta_{ij}^{(t)} \eta_{ij} x_i x_i^T \big) v_j$$

$$\Rightarrow v_j^{(t+1)} = (X^T \Omega_j X)^{-1} X^T \kappa_j \tag{40}$$

where,

$$\kappa_j = [\eta_{1j} (\frac{1}{2} + \beta_{1j}^{(t)} \log \sum_{l=1, l\neq j}^{N} \exp(x_1^T \hat{v}_l)), \dots, \eta_{Nj} (\frac{1}{2} + \beta_{Nj}^{(t)} \log \sum_{l=1, l\neq j}^{N} \exp(x_N^T \hat{v}_l))]^T \tag{41}$$

$$\Omega_j = diag(\beta_{1j}^{(t)} \eta_{1j}, \dots \beta_{Nj}^{(t)} \eta_{Nj}) \tag{42}$$

As is clear, we have closed form updates for all steps. Thus, the use of Polya-Gamma augmentation helps us achieve the necessary goals of closed form updates, significantly reducing the number of paramaters.