EE491A: Mixture of Bayesian SVM Experts

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

Abstract

We propose a novel extension of Bayesian Support Vector Machine (SVM) in a Mixture of Experts (MoE) setting, which can learn interpretable non-linear models efficiently. We also construct a novel gating network using Polya-Gamma Augmentation, which provides closed form updates when using a softmax based gating network. We compare the performance of the proposed framework on multiple binary classification tasks and show that the framework achieves competitive results.

Introduction

Support Vector Machines [1] (SVM) became the de-facto blackbox algorithm in machine learning due to their immense efficacy on binary classification tasks. The kernel trick [8] and fundamental extensions beyond binary classification [2, 10] 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 [7], 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

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

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 [7]. Minimizing (1) 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:

$$p(w|\lambda) = \mathcal{N}(0, \lambda^{-1}I) \tag{2}$$

$$L(y_i|w, x_i) = \exp(-2\max(1 - y_i x_i^T w, 0)) \tag{3}$$

$p(w|\lambda)$ represents the prior and $L(y_i|w, x_i)$ defines the pseudo-likelihood. The fundamental contribution of [7] was to decompose (3) into a location-scale mixture of normals. To be more precise, [7] shows that:

$$L(y_i|w, x_i) = \int_0^\infty \frac{1}{\sqrt{2\pi\gamma_i}} \exp(-\frac{1}{2}\frac{(1 + \gamma_i - y_i x_i^T w)^2}{\gamma_i}) d\gamma_i \tag{4}$$

(4) 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. 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 resort to Bayesian tools like **Expectation Maximization** (EM) or **Markov Chain Monte Carlo** (MCMC) methods, as has been show in [5, 7]. [5], in particular, goes onto show multiple extensions for Bayesian SVMs such as Multiclass SVMs, Nonlinear Kernel SVMs and Support Vector Regression. It is important to note that these extensions are simple to incorporate in the framework we have proposed.

Mixture of Experts [3] is a simple yet extremely effective way to learn non-linear models by combining locally linear models. In fact, the setup is general enough to incorporate non-linear "experts" as well. Mixture of experts model has been widely studied and used as well [12]. The idea behind mixture of experts, in its simplest form, is to break a non-linear problem into a set of linear 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 handles a linear subproblem and a *Gating Network* which assigns each input to an expert. A smooth introduction to the topic of mixture of experts is given in [4].

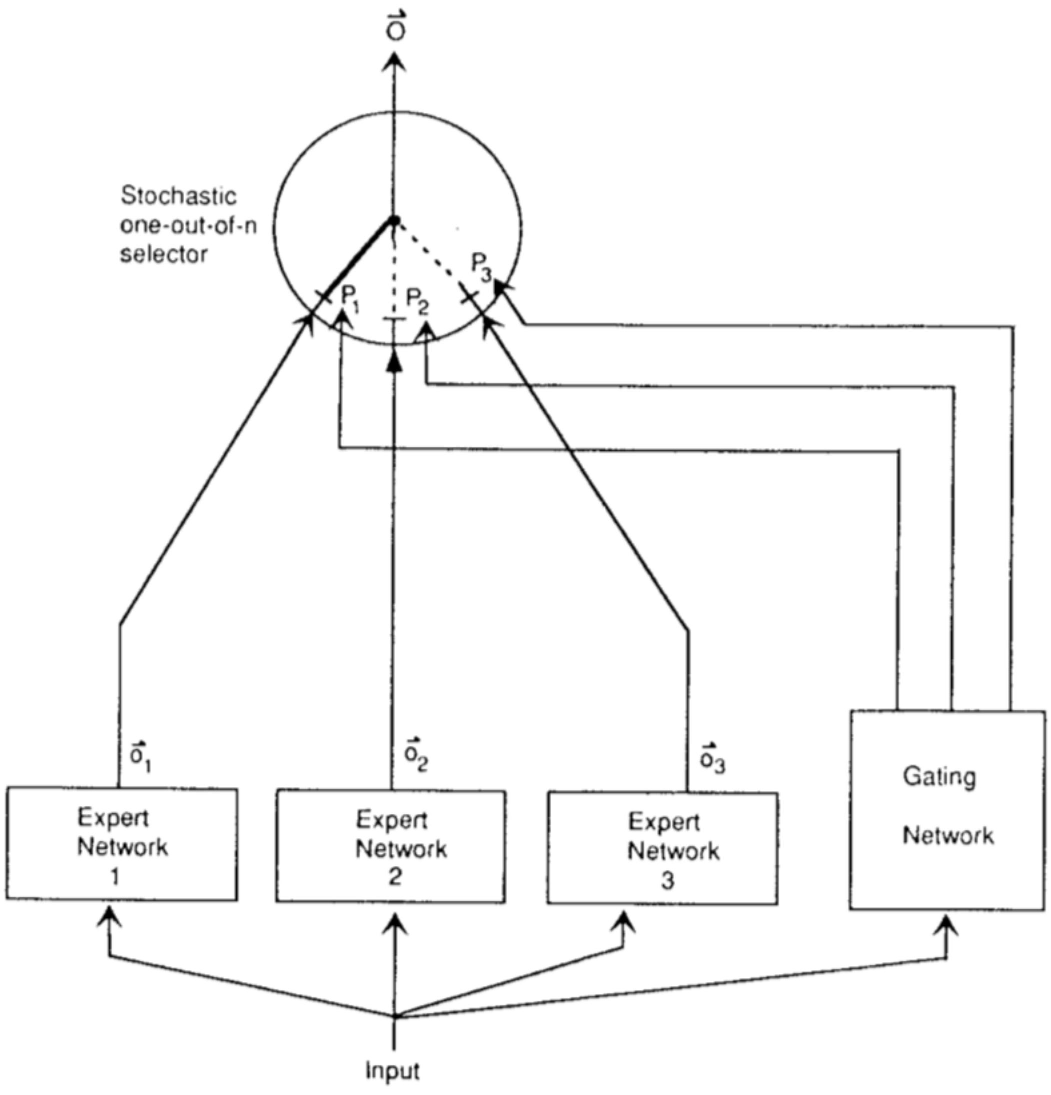

Figure 1: An illustration of a generic Mixture of Experts model [3]

The model in [3], proposed for non-linear regression, uses a softmax gating function to do a soft assignment of an input to an expert. The weighted/stochastic output of local linear expert/s is taken as the final output. Formally, for a mixture of K experts, we have a gating network $V = \{v_i\}_{i=1}^K$ and expert network $W = \{w_i\}_{i=1}^K$. The softmax value for each expert is given by $\pi_k(x_i) = \frac{\exp(v_k^T x_i)}{\sum_{l=1}^K \exp(v_l^T x_i)}$. The final prediction of the model is given by $\hat{y}_i = \sum_{k=1}^K \pi_k(x_i) w_k^T x_i$ or $\hat{y}_i = w_{\hat{k}}^T x_i, \hat{k} = \text{argmax}_k \, \pi_k(x_i)$ depending on the model choice. The latter is computationally cheaper, but can provide a biased prediction if the probabilities are uniformly distributed across experts. The model is trained by Expectation Maximization (EM). Assume a latent variable $Z = \{z_i\}_{i=1}^N$, which is a one-hot vector assigning each input to an expert. The objective to be optimized is given by

$$\mathcal{L}(y, Z|X, W, V) = \sum_{i=1}^{N} \log p(y_i, z_i|x_i, W) = \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1]\big(\log p(y_i|x_i, w_j) + \log p(z_{ij} = 1|x_i, V)\big) \tag{5}$$

(5) assumes that every input is assigned to exactly one expert. (Hence, the indicator function. This is a necessary assumption in EM, consequences and necessity of which have been discussed in last section of appendix

(15)). Here, the cluster-expert assignment is not known. Therefore we optimize $E_z[\mathcal{L}(y, Z|X, W)]$ where the expectation in iteration $t+1$ is computed with respect to $p(z|X, V^{(t)}, W^{(t)})$. Details can be referred to in [3,12].

We propose a novel framework which uses Bayesian SVMs as experts in the Mixture of Experts framework described earlier. SVMs (which are great at linear separation) of training data, cannot be directly incorporated into such a framework. However, Bayesian SVMs alleviate this problem. We also look at different gating networks. In particular, we formulate a novel 'Polya-Gamma augmented Softmax' gating network [6,9].

Model Architectures and Training Details

We propose a Mixture of Experts based model, where the experts are Bayesian SVMs. We initially look at the task of binary classification to justify such a construction. 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 Bayesian formulation makes SVMs more amenable to be included in other constructs, such as Mixture of Experts. In particular, Mixture of Experts has been very effective at constructing non-linear models which are computationally efficient and highly interpretable.
- As we will show in the later formulations, this model can be trained very efficiently using EM. In our later architectures, we show a formulation such that both E step and M step have closed form updates.

We look at three formulations in detail. These three formulations differ only in the architecture of the gating network. We derive the EM updates for all the three formulations. The derivations for EM updates are slightly tedious, therefore, they have been shifted to the appendix which prevents the discussion from getting clouted. MCMC rountines like Gibbs Sampling, though feasible to derive, are currently left from the discussion.

Naive Softmax Gating Network

For the first model, we follow [3] and construct a softmax gating network. The experts as stated before, are Bayesian SVMs. For a setting with K experts, $\theta_g = \{v_i\}_{i=1}^{K}$ denotes the gating network matrix, and $\theta_e = \{w_i\}_{i=1}^{K}$ denotes the experts weight matrix. There are two additional set of latent variables, which are supposed to be learnt during training: $\gamma = \{\gamma_{ij}\}_{i=1,j=1}^{i=N,j=K}$ and $Z = \{z_i\}_{i=1}^{N}$. Z represents the latent variable for input-expert assignment and γ represents the data augmentation for all Bayesian SVM experts, for each training example. These latent variables are learnt jointly is single EM loop.

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 [3], 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.

Generative Gating Network

The problem of having an iterative procedure (IRLS for softmax gates) within an iterative procedure (EM) was identified and solved in [11] 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 $p(j|x, \Theta) \propto \alpha_j p(x|j, \Theta)$ where α_j is the prior probability and $p(x|j, \Theta) = \mathcal{N}(\mu_j, \Sigma_j)$. Therefore, the new set of parameters for the gating network is $\theta_g = \{\alpha_k, \mu_k, \Sigma_k\}_{k=1}^{K}$.

- *Maximize* $\mathbb{E}_{p(\gamma,z|\Theta^{(t)},D)}[\log p(y,x,\gamma,z|\Theta)]$: 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 [11]. The derivations and the updates are provided in the Appendix section A.2.1.

This formulation achieves the goal of getting closed form updates for both the E and M step. However, there are two drawbacks for this approach:

- The number of parameters has increased drastically. We are using a Gaussian distribution to model the input for each expert. This implies that for each expert, we have to estimate a prior probability, an expert weight vector, a mean vector and a covariance matrix. The number of parameters are now $\mathcal{O}(KD^2)$, as opposed to $\mathcal{O}(KD)$, where D is the data dimensionality. The estimates for these parameters can be poor if the data available is less, or we are dealing with high dimensional data.

- The objective defined in this problem solves, in general, a much harder problem than is required. That is, we are aiming to model the $\{X, y\}$ rather than $y|X$. This problem is present in Generative Classification approaches as well, when compared to Discriminative Classification. Another associated problem is that the model assumes the data for each expert to be distributed as a Unimodal Gaussian.

The drawbacks listed above do not really show in the experimental results. However, we look at relatively simple problems in our experiments, and these problems might reflect in higher dimensional data, or data which is harder to classify. Therefore, we provide a new formulation which counters both the proposed problems, while preserving closed form updates for E and M steps.

Polya-Gamma Augmented Softmax Gating Network

Polya-Gamma augmentation was proposed in [6] to provide fully Bayesian inference for logistic regression models. Later, [9] shows an EM based routine to learn logistic regression models using the same data augmentation. In this formulation, we revert back to a softmax gating network, that is $\theta_g = \{v_i\}_{i=1}^{K}$. However, like Bayesian SVM, we augment latent variables from the standard polya-gamma distribution. This augmentation is motivated by the following result shown in [6]:

$$\frac{\left(e^{\psi}\right)^a}{\left(1+e^{\psi}\right)^b} = 2^{-b} e^{(a-b/2)\psi} \int_0^{\infty} e^{-\beta\psi^2/2} p(\beta) d\beta \tag{6}$$

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 Bernoulli likelihood (for $a = b = 1$) will become a Gaussian Likelihood Model. Thus, we can derive a EM routine which assumes β as latent variables.

Polya-gamma augmentation can be extended to multinomial likelihoods as well, as has been discussed in both [6, 9]. The softmax gating network is a special case of multinomial likelihood. In this formulation, each gate vector is augmented with a polya-gamma latent variable $\beta_{ij} \sim PG(1, 0)$ for each example. The exact details of how to convert softmax into pseudo-bernoulli likelihoods for each gating vector, and how to derive closed form updates for both E and M step using Expectation Conditional Maximization (ECM) are discussed in Appendix A.2.2.

The Polya-Gamma augmented Gating Networks are a general idea, independent of the Bayesian SVM experts. We obtain closed form updates, while preserving the softmax based structure of the gating network. Therefore, the number of parameters to be estimated is $\mathcal{O}(KD)$, which is significantly lesser than the parameters to be estimated in a generative gating network.

While this architecture, on paper, compensates the drawbacks associated with both the previous architectures, there are some concerns that need to be redressed. If using Polya-Gamma and Bayesian SVM in a Mixture of Experts simultaneously, we have three separate data augmentations. While preliminary testing of this setup seems promising, a thorough analysis for potential instability while scaling up the number of experts is mandated. A drawback, when compared to previous gating networks, is that the MCMC routines would be extremely slow as sampling from PG distribution is relatively inefficient (PG distribution is actually an infinite sum of scaled gamma distributions, sampling from which is computationally expensive).

Results

We evaluate the results of Binary Classification using Mixture of Binary SVM Experts on different datasets, particularly those that have been evaluated in [13]. The results shown here are preliminary, and do not represent a comprehensive analysis, as has been carried out in [13]. Nonetheless, the results sufficiently justify the utility of this construction, and validate further research on it. For most parts, we are interested in the results obtained using Generative Gating network and PG augmented Softmax Gating Network. A naive softmax network has a lot of hyperparameters, which can be cumbersome to tune. Thus, the former two architectures are favoured for evaluation.

A few top performing models are picked from the results in [13] on six binary classification datasets listed in the table. Only the lowest error rates are considered for all the models in this comparison.

Dataset	**LR**	**SVM**	**stack-ζ (T=5)**	**SS-ζ (T=5)**	**MBSVME-GG**	**MBSVME-PG**
banana(3)	43.38	10.28	27.45	11.28	**9.43**	17.72
breast cancer(10)	24.37	23.92	24.61	25.43	**18.19**	19.49
titanic(4)	21.69	21.7	22.01	21.49	**20.73**	21.31
waveform(22)	12.74	9.87	11.56	11.0	12.92	**8.14**
german(21)	21.93	20.79	20.75	21.77	19	**18.33**
image(19)	16.48	2.32	7.45	**2.2**	3.87	9.41

Table 1: **LR**: Logistic Regression, **SVM**: Support Vector Machine with RBF Kernel, **stack-ζ (T=5)**: stack-softplus regression with T=5 [13], **SS-ζ (T=5)**: SS-softplus regression with $K_{max} = 20$ and $T = 5$ [13], **MBSVME-GG**: Mixture of Bayesian SVM Experts with Generative Gating, **MBSVME-PG**: Mixture of Bayesian SVM experts with Polya-Gamma augmented Softmax gating Networks. For the datasets, the number in bracket shows the data dimensionality.

With the exception of Image dataset, Mixture of Bayesian SVM experts (either with generative gating or PG augmented softmax) achieve the lowest error rates. The numbers in the table justify the construction, and validate further inquiry into the performance of this framework. From the preliminary analysis, it seems that PG augmented gating network do better with higher dimensional data, while generative gating network do better with lower dimensional data. The possible reasoning for the same has been discussed earlier. However, in general, this analysis is far from comprehensive, and therefore, a much rigorous analysis of the framework should give more insight.

Discussion

The framework is computationally cheap to train, is easily interpretable and gives good performance on binary classification tasks. The hyperparameters for the model are very few, and easy to tune. However, there are a few limitations. Being a non-convex optimization, in particular, depending on the interaction of two networks optimizing different objectives, the initialization plays a significant role. This, in particular, is a concern with PG augmented Softmax Gating Network when used with low dimensional datasets (and sometimes even relatively higher dimensional dataset, such as Image). Thus, we need to look into the initialization of such a network. One such possible initialization can be done by pre-clustering the data (using K-means or GMM), which can be used to initialize the gating network.

There are multiple extensions possible for this framework. It can easily be extended to regression tasks using Bayesian SVRs or Kernel based Bayesian SVM experts. Multiclass Bayesian SVMs provide another direction of extension. This is a particularly interesting direction, as we can potentially replace the gating network by a Multiclass Bayesian SVM. We plan to explore these directions, along with a rigorous analysis and testing of the framework, in particular, Polya-Gamma augmented Softmax Gating Network.

References

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

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

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

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

[11] 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.

[12] 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.

[13] Mingyuan Zhou. Softplus regressions and convex polytopes. *arXiv preprint arXiv:1608.06383*, 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, $\gamma_i = \{\gamma_{i1}, \gamma_{i2}, \ldots \gamma_{iD}\}$ 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,

$$\begin{aligned} & p(\Theta, \gamma, z|D) \propto p(\Theta) p(y, \gamma, z|X, \Theta) \\ \Rightarrow & \log p(\Theta, \gamma, z|D) \propto \log p(\Theta) + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1](\log p(z_{ij} = 1|x_i, \theta_g) + \log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)) \\ \Rightarrow & \mathbb{E}[\log p(\Theta, \gamma, z|D)] \propto \log p(\Theta) + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{E}[z_{ij}](\log p(z_{ij} = 1|x_i, \theta_g) + \mathbb{E}[\log p(y_i, \gamma_{ij}|x_i, \theta_e, z_{ij} = 1)]) \end{aligned} \tag{7}$$

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 (3), we get:

$$\begin{aligned} \mathbb{E}[z_{ij}] &= p(z_{ij} = 1|x_i, y_i, \Theta^{(t)}) \\ &\propto p(y_i|x_i, z_{ij} = 1, \Theta_e^{(t)}) p(z_{ij}|x_i, \Theta_g^{(t)}) \\ &\propto \exp(-2\max(0, 1 - y_i x_i^T w_j^{(t)})) \exp(x_i^T v_j^{(t)}) \\ \Rightarrow \mathbb{E}[z_{ij}] &= \frac{\exp(x_i^T v_j^{(t)} - 2\max(0, 1 - y_i x_i^T w_j^{(t)}))}{\sum_{l=1}^{K} \exp(x_i^T v_l^{(t)} - 2\max(0, 1 - y_i x_i^T w_l^{(t)}))} = \eta_{ij} \end{aligned} \tag{8}$$

Now, onto the other expectation:

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

Here,

$$\log[\frac{1}{\sqrt{2\pi\gamma_{ij}}}\exp(-\frac{(1+\gamma_{ij}-y_iw_j^Tx_i)^2}{2\gamma_{ij}})] = \frac{-1}{2}\log 2\pi - \frac{1}{2}\log\gamma_{ij} - \frac{(1+\gamma_{ij})^2}{2\gamma_{ij}} - \frac{(y_iw_j^Tx_i)^2}{2\gamma_{ij}} + \frac{y_iw_j^Tx_i(1+\gamma_{ij})}{\gamma_{ij}}$$

We only care about the terms involving w_j, that is, only the last two terms. Therefore, we only care about the expectation $\mathbb{E}[\frac{1}{\gamma_{ij}}] = |1-y_ix_i^Tw_j^{(t)}|^{-1} = \tau_{ij}^{-1}$ [7]. Therefore, the $\mathbb{E}[\log p(y_i,\gamma_{ij}|x_i,\theta_e,z_{ij}=1)]$ can be replaced by $\frac{-(y_iw_j^Tx_i)^2+2y_iw_j^Tx_i}{2\tau_{ij}} + 2y_iw_j^Tx_i$ in (7) to get the final objective (note we are ignoring the terms not involving w_j). Replacing the appropriate expectations in (7), the final objective for the problem is:

$$\mathcal{L}(\Theta,\Theta^{(t)}) = \log p(\Theta) + \sum_{i=1}^N\sum_{j=1}^K \eta_{ij}\Big[\log\frac{\exp(v_j^Tx_i)}{\sum_{l=1}^K\exp(v_l^Tx_i)} - \frac{(y_iw_j^Tx_i)^2-2y_iw_j^Tx_i}{2\tau_{ij}} + 2y_iw_j^Tx_i\Big] \tag{9}$$

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,

$$\frac{\partial\mathcal{L}(\Theta,\Theta^{(t)})}{\partial w_j} = 0$$

$$\Rightarrow (\lambda w_j + \sum_{i=1}^N\frac{\eta_{ij}}{\tau_{ij}}(w_j^Tx_i)x_i) = \sum_{i=1}^N\eta_{ij}(\frac{y_ix_i}{\tau_{ij}} + y_ix_i) \tag{10}$$

$$\Rightarrow w_j = (X^TA_jX+\lambda I)^{-1}(\sum_{i=1}^N\eta_{ij}\frac{\tau_{ij}+1}{\tau_{ij}}y_ix_i) \tag{11}$$

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:

$$\begin{aligned}\frac{\partial\mathcal{L}(\Theta,\Theta^{(t)})}{\partial v_j} &= -\beta v_j + \sum_{i=1}^N\eta_{ij}x_i - \sum_{i=1}^N\sum_{j=1}^K\eta_{ij}\frac{\partial\log\sum_{l=1}^K\exp(v_l^Tx_i)}{\partial v_j}\\ &= \sum_{i=1}^N\Big[\eta_{ij} - \frac{\exp(v_j^Tx_i)}{\sum_{l=1}^K\exp(v_l^Tx_i)}\Big]x_i - \beta v_j\end{aligned} \tag{12}$$

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 [11], while the second approach uses the 'Polya-Gamma Data Augmentation' for the softmax weight vectors [6, 9].

A.2.1 Generative Gating Networks

The results derived in this section borrows some results from [11], 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,

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

$$\Rightarrow \mathbb{E}[\log p(y,x,\gamma,z|\Theta)] = \sum_{i=1}^N\sum_{j=1}^K\mathbb{E}[z_{ij}]\Big(\mathbb{E}[\log p(y_i,\gamma_{ij}|x_i,w_j)] + \log\alpha_j + \log p(x_i|\theta_g,z_{ij}=1)\Big) \tag{13}$$

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 (11):

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

The updates for the parameters of the gating network are borrowed from [11]. 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 [6, 9]. 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 $p(\gamma,\beta,z|\Theta^{(t)},X,y)$.

$$\begin{aligned}
\log p(y,\gamma,\beta,z|\Theta,X) &= \sum_{i=1}^N\sum_{j=1}^K \mathbb{1}[z_{ij}=1]\log p(y_i,z_{ij}=1,\beta_i,\gamma_{ij}|\Theta,x_i) \\
&= \sum_{i=1}^N\sum_{j=1}^K \mathbb{1}[z_{ij}=1]\Big(\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1) + \log p(\beta_{ij},z_{ij}=1|\Theta,x_i)\Big) \\
\Rightarrow \mathbb{E}[\log p(y,\gamma,\beta,z|\Theta,X)] &= \sum_{i=1}^N\sum_{j=1}^K \mathbb{E}[z_{ij}]\Big(\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] + \mathbb{E}[\log p(\beta_{ij},z_{ij}=1|\Theta,x_i)]\Big) \qquad (14)
\end{aligned}$$

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

$$\mathbb{E}[\mathbb{1}[z_{ij}=1]\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] = \mathbb{E}[z_{ij}]\mathbb{E}[\log p(y_i,\gamma_{ij}|\Theta,x_i,z_{ij}=1)] \qquad (15)$$

This, in general, is incorrect. But, (15) 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 [9], 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 (8) and (11) are directly applicable in this derivation. Now, $\beta_{ij}^{(t)} = \mathbb{E}[\beta_{ij}] = \frac{1}{2\psi_{ij}^{(t)}}\tanh\psi_{ij}^{(t)}$ [9]. The coupling of parameters mandates the usage of Expectation Conditional Maximization (ECM), when maximizing with respect to v_j [9]. 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,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial v_j} = \sum_{i=1}^N \eta_{ij}\Big(\frac{1}{2}x_i - \beta_{ij}^{(t)}(x_i^T v_j - \log \sum_{l=1,l\neq j}^K \exp(x_i^T \hat{v}_l))x_i\Big) = 0 \\ \sum_{i=1}^N \eta_{ij}(\frac{1}{2} + \beta_{ij}^{(t)} \log \sum_{l=1,l\neq j}^K \exp(x_i^T \hat{v}_l))x_i = (\sum_{i=1}^N \beta_{ij}^{(t)}\eta_{ij}x_i x_i^T)v_j \\ \Rightarrow v_j^{(t+1)} = (X^T\Omega_j X)^{-1}X^T\kappa_j \end{aligned} \tag{16}$$

where,

$$\kappa_j = [\eta_{1j}(\frac{1}{2} + \beta_{1j}^{(t)} \log \sum_{l=1,l\neq j}^N \exp(x_1^T \hat{v}_l)), \dots, \eta_{Nj}(\frac{1}{2} + \beta_{Nj}^{(t)} \log \sum_{l=1,l\neq j}^N \exp(x_N^T \hat{v}_l))]^T \tag{17}$$

$$\Omega_j = diag(\beta_{1j}^{(t)}\eta_{1j}, \dots \beta_{Nj}^{(t)}\eta_{Nj}) \tag{18}$$

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.