Machine Learning (2019) 108:1369–1393
https://doi.org/10.1007/s10994-019-05811-4

A flexible probabilistic framework for large-margin mixture of experts

Archit Sharma[1] · Siddhartha Saxena[2] · Piyush Rai[2]

Received: 26 November 2018 / Revised: 2 April 2019 / Accepted: 8 May 2019 / Published online: 5 June 2019

Abstract
Mixture-of-Experts (MoE) enable learning highly nonlinear models by combining simple expert models. Each expert handles a small region of the data space, as dictated by the gating network which generates the (soft) assignment of input to the corresponding experts. Despite their flexibility and renewed interest lately, existing MoE constructions pose several difficulties during model training. Crucially, neither of the two popular gating networks used in MoE, namely the softmax gating network and hierarchical gating network (the latter used in the hierarchical mixture of experts), have efficient inference algorithms. The problem is further exacerbated if the experts do not have conjugate likelihood and lack a naturally probabilistic formulation (e.g., logistic regression or large-margin classifiers such as SVM). To address these issues, we develop novel inference algorithms with closed-form parameter updates, leveraging some of the recent advances in data augmentation techniques. We also present a novel probabilistic framework for MoE, consisting of a range of gating networks with efficient inference made possible through our proposed algorithms. We exploit this framework by using Bayesian linear SVMs as experts on various classification problems (which has a non-conjugate likelihood otherwise generally), providing our final model with attractive large-margin properties. We show that our models are significantly more efficient than other training algorithms for MoE while outperforming other traditional non-linear models like Kernel SVMs and Gaussian Processes on several benchmark datasets.

Keywords Probabilistic modelling · Mixture of experts · Bayesian SVMs

Archit Sharma and Siddhartha Saxena contributed equally to this work.

Editors: Karsten Borgwardt, Po-Ling Loh, Evimaria Terzi, Antti Ukkonen.

✉ Siddhartha Saxena
siddsax@cse.iitk.ac.in

Archit Sharma
architsh@google.com

Piyush Rai
piyush@cse.iitk.ac.in

1 Google AI Resident, Google Brain, Mountain View, CA, USA

2 IIT Kanpur, Kanpur, India

1 Introduction

Learning complex models by combining several simpler models is at the heart of ensemble methods. The Mixture-of-Experts (MoE) paradigm provides an attractive and principled, probabilistic framework for constructing such ensembles by combining simple "experts" via probabilistic mixture models (Yuksel et al. 2012; Masoudnia and Ebrahimpour 2014). Each expert in the MoE handles a small region of the data space where the data-to-expert assignment is controlled by a *gating network*. The gating network essentially learns a flat/hierarchical partitioning of the input space. The MoE paradigm has been successfully used to design nonlinear models by combining simpler linear expert models. Note however that the MoE setting does not restrict the experts to be linear models, e.g., each expert can be a Gaussian Process (Meeds and Osindero 2006).

As compared to the other dominant paradigm for nonlinear learning, i.e., kernel methods (Shawe-Taylor and Cristianini 2004), the MoE paradigm is often more appealing due to several reasons: (1) MoE has an inherently probabilistic formulation which enables a probabilistic/fully Bayesian treatment, (2) MoE based methods are faster at training and test time while also being efficient in terms of space since they do not have to deal with kernel matrices; and (3) MoE based methods provide nice interpretability—for example, a nonlinear classifier learned by a MoE can be thought of as a combination of several linear classifiers. Finally, MoE has also been gaining considerable attention lately in the design of very large neural networks (Shazeer et al. 2017) due to MoE's inherent property of *conditional* computation, i.e., only a part of the model is active for a given input.

Despite these appealing properties, the MoE architectures are known to be considerably difficult to train as they usually lack closed form parameter updates. Note that learning a MoE model requires learning the parameters of each expert in the mixture, as well as the parameters that define the gating network. A typical approach to learn MoE models is to formulate them as latent variable models and use the Expectation–Maximization (EM) (Jordan and Jacobs 1994) algorithm. In the EM algorithm for MoE, the latent variables denoting the input-to-expert (soft) assignments are inferred in the E step and the parameters defining the experts, and the gating network is updated in the M step. However, except for some special cases, the M step updates for MoE usually do not have a closed form and require iterative double loop procedures or iterative, recursive least squares solvers, which can be exhibit slow convergence. The lack of a closed form of parameter updates on MoE models can usually be (1) Due to the choice of gating network which is usually modeled by a softmax (in case we want a flat partitioning of the inputs), or modeled by a hierarchy of logistic models (in case we want a hierarchical partitioning, e.g., in hierarchical MoE models Jordan and Jacobs 1994; Bishop and Svenskn 2002); and (2) Due to the choice of the expert models which, for classification problems, are usually chosen to be logistic regression models (again lacking closed form parameter updates).

In this work, we present a probabilistic MoE framework which addresses the above issues in a principled manner. Our contributions can be summarized as follows:-

– We propose a rich suite of gating networks with appealing properties, for both flat and hierarchical MoE (Bishop and Svenskn 2002). More importantly, the proposed inference algorithms lead to efficient, closed-form parameter updates using variable augmentation techniques. While there do exist, some inference methods for flat MoEs that have closed form updates (Yuksel et al. 2012), they generally come with the cost of learning a huge number of parameters. While, to the best of our knowledge, this is the first work that offers closed form parameter updates for hierarchical MoE.

- One of our gating networks (logistic stick-breaking process based gating) can also learn the number of experts using a logistic stick-breaking prior (Ren et al. 2011), while still having simple closed form updates for the parameters, thanks to the variable augmentation scheme we employ.
- Our MoE framework leverages Bayesian linear support machines (SVM) (Polson et al. 2011) as constituent experts. Bayesian SVMs as experts are appealing due to two reasons: (1) The model naturally enjoys the large-margin properties of SVMs; and (2) Variable augmentation techniques can also be exploited to derive closed-form parameter updates for Bayesian linear SVM parameters. Crucially, the MoE framework with Bayesian linear SVM results in a nonlinear Bayesian SVM without employing the recently proposed kernelized extensions of Bayesian SVM (Henao et al. 2014), which are difficult to do inference on and are slow at training and test time. Our probabilistic framework can also be a viable alternative to Gaussian Process (GP) based nonlinear classification, which usually lacks conjugacy and is hard to do inference on Nickisch and Rasmussen (2008).
- We conduct extensive experiments to show that the use of closed-form updates lead to much faster training times compared to existing MoE frameworks whereas the use of Bayesian SVM as experts, leads to significant improvement in performance over competing methods for non-linear large margin models (Henao et al. 2014; Cotter et al. 2013; Wang and Zhu 2014; Zhu et al. 2011) as well as strong MoE baselines (Zhou 2016).

Our work lays out a novel and principled foundation to build MoE models, as well as it can be integrated into other more sophisticated models that require MoE as its building block, e.g., recent work on conditional computation in deep neural networks (Shazeer et al. 2017). Our inference algorithms with clean, closed-form updates are considerably more simple as compared to the existing inference methods for MoE.

2 Background

We first introduce some notation and provide a brief overview of the Mixture of Experts (Yuksel et al. 2012; Masoudnia and Ebrahimpour 2014) and Bayesian SVMs (Polson et al. 2011) which we use as experts in our proposed framework. We will focus on binary classification in which we assume we are given training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ and the goal is to learn a classifier that can predict the label $y_* \in \{-1, +1\}$ for a new input $\boldsymbol{x}_*$. We will denote the feature matrix by $\mathbf{X} \in \mathbb{R}^{N\times d}$ and ground truth labels by $\boldsymbol{y} \in \{-1, +1\}^N$. Although we present our framework focusing on binary classification, it can be extended naturally to multi-class classification or regression problems.

2.1 Mixture of experts

A Mixture-of-Experts (MoE) model is defined by a set of K experts, each of which is a probabilistic supervised learning model $\{p(y|\boldsymbol{x}, \boldsymbol{w}_k)\}_{k=1}^K$ parametrized by weights $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$, with $\boldsymbol{w}_k \in \mathbb{R}^d$. In addition, the framework is defined by a gating network parametrized by $\mathbf{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K]$, which defines the probabilities of assignment of an input $\boldsymbol{x}_n$ to each of the K experts. For example, the probability of input $\boldsymbol{x}_n$ being assigned to expert k is defined by a function $\pi_k(\boldsymbol{x}_n)$ where $\pi_k(\cdot)$ is defined in terms of the parameters $\mathbf{V}$ of the gating network. Note that $\sum_{k=1}^{K} \pi_k(\boldsymbol{x}_n) = 1$. The MoE defines the conditional probability of y_n given $\boldsymbol{x}_n$ as

$$p(y_n|\boldsymbol{x}_n, \mathbf{W}, \mathbf{V}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x}_n) \cdot p(y|\boldsymbol{x}_n, \boldsymbol{w}_k) \tag{1}$$

Different versions of MoE primarily differ in the choice of the experts and the gating network (Yuksel et al. 2012; Masoudnia and Ebrahimpour 2014 provide an excellent overview). The gating networks can learn a flat or a hierarchical partitioning of the input space (the latter being the case with *hierarchical* mixture of experts Bishop and Svenskn 2002). It is their plug-and-play nature that provides a considerable flexibility in designing MoE models. However, learning the gating network parameters also poses difficulty in MoE models, where lack of conjugacy in typically used softmax/logistic gating usually prevents from getting closed form updates for the gating network parameters (and one has to rely on expensive procedures such as inner-loop iterative recursive least square algorithms). In this work, we proposed a suite of gating networks, along with efficient inference algorithms based on variable augmentation schemes, which allows us to derive closed form parameter updates for the gating network parameters.

2.2 Bayesian support vector machines

We use a probabilistic formulation of support vector machines—the *Bayesian* SVM (Polson et al. 2011)—as experts in our proposed MoE framework. As we will show, this choice leads to a simple and seamless inference procedure for all the parameters of our MoE framework using latent variable augmentation strategies. At the same time, our framework also enjoys the large-margin properties of SVMs.

The Bayesian SVM is based on representing the hinge-loss objective in non-probabilistic SVM as a scale-mixture of Gaussians (Polson et al. 2011).

$$\exp(-2\max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x_i})) = \int_0^\infty \frac{1}{\sqrt{2\pi\gamma_i}} \cdot \exp\left(-\frac{(1 + \gamma_i - y_i \boldsymbol{w}^\top \boldsymbol{x_i})^2}{2\gamma_i}\right) d\gamma_i \tag{2}$$

Equation 2 enables writing the negative of the hinge-loss $\max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x_i})$ on each example $\boldsymbol{x}_i, y_i$ as a Gaussian likelihood when conditioned on an auxiliary latent variable γ_i with an inverse-Gaussian distribution. As shown in Polson et al. (2011), the likelihood when conditioned on γ_i, is of the form

$$p(y_i, \gamma_i|\boldsymbol{w}, \boldsymbol{x}_i) = \mathcal{N}(1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i| - \gamma_i, \gamma_i) \tag{3}$$

This is attractive since the Gaussian likelihood is conjugate with Gaussian priors or scale-mixture of Gaussian priors on the weight vector $\boldsymbol{w}$, which leads to simple, closed form posterior updates.

Note that although the Bayesian SVM formulation proposed in Polson et al. (2011) is applicable for linear SVMs, our MoE based formulation would naturally, allow it to be used for solving *nonlinear* classification problems without having to rely on kernel extensions which are usually much more difficult to do inference on, and are computationally very expensive (Henao et al. 2014) since they work with kernel matrices.

There have been other works to tackle this problem such as Williams and Seeger (2001) and Rahimi and Recht (2008). While such approximations can, to some extent, alleviate the scalability issue, our MoE based framework offers several other benefits beyond just scalability, such as better interpretability and improved inference algorithms. The proposed MoE framework can also be easily used with kernel-based experts (like Gaussian Process) and these kernel-approximation techniques can further help in scaling up the model.

3 A probabilistic framework for large-margin mixture of experts

Our Mixture-of-Experts (MoE) architecture leverages Bayesian SVM as its constituent experts. Assuming the number of experts to be K (which, as we will show in Sect. 4.2, can also be learned from data), our MoE formulation uses K Bayesian SVMs defined by the set of weight vectors $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$ where $\boldsymbol{w}_k$ denotes the weight vector of the kth Bayesian SVM expert.

We use $z_{ik} = 1$ to denote $\boldsymbol{z}_i = k$, i.e., the input $\boldsymbol{x}_i$ has been assigned to expert k. The expectations of the latent variables z_{ik} and γ_{ik}, given the current estimates $\hat{\mathbf{W}}$ and $\hat{\mathbf{V}}$ of MoE model parameters, can be computed in closed form

$$\begin{aligned}\eta_{ik} &= \mathbb{E}[z_{ik}] \propto p(y_i|\hat{\boldsymbol{w}}_k, \boldsymbol{x}_i, \gamma_{ik})p(z_{ik} = 1|\boldsymbol{x}_i, \hat{\mathbf{V}})\\ \tau_{ik} &= \mathbb{E}[\gamma_{ik}^{-1}] = |1 - y_i\hat{\boldsymbol{w}}_k^\top \boldsymbol{x}_i|^{-1}\end{aligned} \tag{4}$$

The form of $p(z_{ik} = 1|\boldsymbol{x}_i, \hat{\mathbf{V}})$ will depends on the form of gating network, discussed in Sects. 4.1–4.4.

We will use an Expectation–Maximization (EM) (Dempster et al. 1977) algorithm to learn the latent variables and the parameters. The E step will compute the expectation of the latent variables as defined in Eq. 4 and the M step will compute the point estimate of the MoE model parameters $\mathbf{W}$ and $\mathbf{V}$.

Denoting $\mathbf{Z} = [z_1, \ldots, z_N]$ as the input-to-expert assignment latent variables and $\boldsymbol{\gamma} = \{\gamma_{ik}\}_{i=1,k=1}^{N,K}$, the complete data log-likelihood of the model, as required to derive the EM algorithm will be

$$\mathcal{L}(\boldsymbol{y}, \mathbf{Z}, \boldsymbol{\gamma}|\mathbf{X}, \mathbf{W}, \mathbf{V}) = \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}(\log p(y_i, \gamma_{ik}|\boldsymbol{x}_i, \boldsymbol{w}_k) + \log p(z_{ik} = 1|\boldsymbol{x}_i, \mathbf{V})) \tag{5}$$

For brevity, we have ignored the prior terms on $\mathbf{W}, \mathbf{V}$. We use zero-mean Gaussian prior on the Bayesian SVM weight vectors $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$. The priors on the gating network parameters $\mathbf{V}$ will depend on the choice of the gating network. Also, there may be additional variable augmentations depending upon the gating network (which we discuss next), which will require to accordingly modify the gating network probability in Eq. 5.

The EM algorithm alternates between computing the expectations of the latent variables using Eq. 4 and using these expectations to maximize the complete data log-likelihood given by Eq. 5 to find the point estimates of the model parameters $\mathbf{W}$ and $\mathbf{V}$. Assuming Gaussian priors $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$ on the Bayesian SVM weight vectors $\boldsymbol{w}_k$ which act as regularization for our SVM classifiers, the Gaussian likelihood in Eq. 3, leads to the following maximum-a-posteriori (MAP) estimate

$$\hat{\boldsymbol{w}}_k = \left(\lambda\mathbf{I} + \sum_{i=1}^{N} \eta_{ik}\tau_{ik}\boldsymbol{x}_i\boldsymbol{x}_i^\top\right)^{-1}\left(\sum_{i=1}^{N} y_i(\tau_{ik} + 1)\eta_{ik}\boldsymbol{x}_i\right) \tag{6}$$

As can be seen from Eq. 6, the updates for the expert models resemble those computed in Polson et al. (2011), except the presence of η_{ik}. The presence of η_{ik} reweighs the examples for each expert, thus, ensuring that each expert models only a subspace of the input. The estimates of η_{ik} are computed using the gating network in the E-step of the EM algorithm. Thus, the subspace of input that an expert models depends on the structure of the gating network, that is, how we define $\pi_k(\boldsymbol{x}_i) = p(z_{ik} = 1|\boldsymbol{x}_i, \mathbf{V})$. In the next section, we discuss such different gating networks. Note, however, the MAP estimate for the experts is agnostic to the structure of the gating networks and is given by Eq. 6.

4 Gating networks

As discussed above, the choice of the gating network can have a major impact on the final model that is learnt. In this work, we propose three separate choices for flat gating networks: (1) Generative gating network (Sect. 4.1); (2) Polya-Gamma based Softmax gating network (Sect. 4.2); (3) Logistic stick-breaking process (LSBP) prior (Ren et al. 2011) based gating network (Sect. 4.2). We also discuss variable augmentation to learn Hierarchical Mixture-of-Experts (Sect. 4.4). Further, we show, all our gating networks admit a closed form parameter update which is a particularly appealing aspect from the point of view of computational efficiency.

4.1 Generative gating

The Generative Gating (**GG**) network, as the name suggest, models input-expert assignment through Gaussian Mixture Model like input modelling as shown below:

$$\pi_j(\boldsymbol{x}_i) = p(z_{ik} = 1|\boldsymbol{x}_i, \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \propto \alpha_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{7}$$

Here α_k represents the prior probability of expert k and $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ denote the parameters of the Gaussian likelihood of x_k being assigned to expert k. Thus, the parameters of the gating network are given by $\mathbf{V} = \{\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$. This gating network seamlessly embeds within the EM algorithm and the gating network parameters have closed form updated (similar to generative mixture models for data). The Gaussian distributions here can be replaced by some other distributions as well depending on the type of the inputs.

Using the input to expert assignment probabilities $\pi_j(\boldsymbol{x}_i)$ defined above, we can compute the expectation η_{ik} in (4). The estimates of parameters in $\mathbf{V}$ can be computed by maximizing the expected complete data log-likelihood, under the additional constraint that $\sum_{k=1}^{K} \alpha_k = 1$. The updates for gating network parameters are as follows:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \eta_{ik}\boldsymbol{x}_i}{\sum_{i=1}^{N} \eta_{ik}} \tag{8}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \eta_{ik}} \tag{9}$$

From practical perspectives, we prefer to approximate $\boldsymbol{\Sigma}_k$ using a diagonal covariance matrix, rather than the full covariance matrix (as implied by the expression above), which is computationally more efficient to compute (as the number of parameters to be estimated are linear in input data dimensionality as opposed to quadratic for a full covariance matrix). At the same time, we found the empirical performance of the diagonal covariance approximation to be comparable to that of a full covariance matrix. It is pertinent to mention that the generative gating, the simplest of our gating networks, and similar gating networks have been tried in other works as well (Meeds and Osindero 2006).

4.2 Pólya-gamma based softmax gating

The softmax gating is the most commonly used gating network for MoEs, dating back to the first work on MoEs (Jacobs et al. 1991) while also being used in recent works like (Shazeer et al. 2017) as well. However, these prior usages of softmax gating did not have closed form parameter updates and therefore, resort to iterative optimization. This makes our proposed Pólya-Gamma based Softmax Gating construction an appealing alternative.

We overcome the non-conjugacy of softmax gating through a latent variable augmentation scheme based on Pólya-gamma augmentation (**SG-PG**) (Polson et al. 2013) which leverages the following identity:

$$\frac{(e^{\psi})^a}{(1+e^{\psi})^b} = 2^{-b} e^{(a-\frac{b}{2})\psi} \int_0^\infty e^{-\beta\psi^2/2} p(\beta) d\beta \tag{10}$$

where $p(\beta)$ represents the density of $\beta \sim PG(b, 0)$ with PG denoting the Pólya-gamma distribution (Polson et al. 2013). This result inspires the data augmentation described in Scott and Sun (2013) to develop Gibbs sampling or EM based routines for logistic regression, which can be extended to softmax functions. For that, we introduce latent variables $\boldsymbol{\beta} = \{\beta_{ij}\}_{i=1,j=1}^{i=N,j=K}$. In the softmax gating network, we are interested in learning $\mathbf{V} = [\boldsymbol{v}_1, \dots \boldsymbol{v}_K]$. Note that this is equivalent to learning a multinomial classification model with data $\{\boldsymbol{x}_i, \boldsymbol{z}_i\}_{i=1}^N$, albeit the "labels" $\boldsymbol{z}_i$ are latent variables.

The traditionally defined softmax probability of the assignment of an input $\boldsymbol{x}_i$ to an expert k, without the latent variable augmentation, can be expressed as:

$$p(z_{ik} = 1|\mathbf{V}, \boldsymbol{x}_i) = (1 + \exp(-\psi_{ik}))^{-1} \tag{11}$$

where $\psi_{ik} = \boldsymbol{x}_i^T \boldsymbol{v}_k - \log \sum_{l=1,l\neq k}^K \exp(\boldsymbol{x}_i^T \boldsymbol{v}_l)$. This expresses the softmax probability as a Bernoulli likelihood, which can be decomposed by setting $a = b = 1$ in the PG identity. Using the augmentation results from Scott and Sun (2013), we get $\log p(z_{ik} = 1|\boldsymbol{x}_i, \beta_{ik}, \mathbf{V}) \propto \psi_{ik}/2 - \beta_{ik}\psi_{ik}^2/2$. This expression is substituted in the expression of the complete data log-likelihood in (5). The EM algorithm also requires the expectations of the PG latent variables, which luckily are available in closed form

$$\chi_{ik} = \mathbb{E}[\beta_{ik}] = 0.5\hat{\psi}_{ik}^{-1} \tanh(0.5\hat{\psi}_{ik}) \tag{12}$$

as derived in Polson et al. (2013). However, despite the augmentation, the parameter of $\mathbf{V}$ are coupled in the complete data log-likelihood due to coupling in ψ_{ik}. This is resolved by assuming the values of $\boldsymbol{v}_l, l \neq k$ to be fixed to their most recent estimates while estimating $\boldsymbol{v}_k$. This is equivalent to conducting an Expectation Conditional Maximization (ECM) routine. Assuming a Gaussian prior on $\boldsymbol{v}_k$, i.e., $\boldsymbol{v}_k \sim \mathcal{N}(0, \rho^{-1}\mathbf{I})$., the resulting parameter updates for $\boldsymbol{v}_k$ are given by:

$$\hat{\boldsymbol{v}}_k = (\mathbf{X}^\top \hat{\boldsymbol{\Omega}}_k \mathbf{X} + \rho\mathbf{I})^{-1}\mathbf{X}^\top(\hat{\kappa}_{1k}, \dots, \hat{\kappa}_{Nk})^\top \tag{13}$$

$$\hat{\kappa}_{ik} = \eta_{ik}(0.5 + \chi_{ik} \log \sum_{l=1,l\neq j}^{N} \exp(\boldsymbol{x}_i^\top \hat{\boldsymbol{v}}_l))$$

$$\hat{\boldsymbol{\Omega}}_k = \mathrm{diag}(\chi_{1k}\eta_{1k}, \dots, \chi_{Nk}\eta_{Nj}) \tag{14}$$

4.3 Logistic stick-breaking process gating

We present logistic stick-breaking gating network (Rigon and Durante 2017) based on the logistic stick-breaking prior (**LSBP**) (Ren et al. 2011). LSBP-Gating is a non-parametric gating network that partly relieves the machine learning practitioner from the need of tuning the number of experts as it "learns" the hyperparameter from the training data itself.

The LSBP construction specifies a large enough truncation level on the number of experts. The model prunes out unnecessary experts as warranted by the data, converging on the number of required experts. The gating probability for an input $\boldsymbol{x}_i$ is defined as:

$$\pi_k(\boldsymbol{x}_i) = \begin{cases} \alpha_1(\boldsymbol{x_i}) & \text{if} \quad k = 1 \\ \alpha_k(\boldsymbol{x_i}) \prod_{l=1}^{k-1}(1 - \alpha_l(\boldsymbol{x}_i)) & \text{if} \quad 1 < k < K-1 \\ \prod_{l=1}^{K-1}(1 - \alpha_l(\boldsymbol{x}_i)) & \text{if} \quad k = K \end{cases} \tag{15}$$

$$\alpha_k(\boldsymbol{x}_i) = \left(1 + \exp\left(-\nu_k^T \boldsymbol{x}_i\right)\right)^{-1} \tag{16}$$

This essentially means that a data point is assigned to the first expert with probability $\alpha_1(\boldsymbol{x_i})$ while for the rest of the gates, it is defined as the product of the probability that it has not 'yet' been assigned to an expert and the probability it will be assigned to the given gate. It is easy to check that this is a proper distribution whereas, from a computational perspective, we specify a fixed (but a large) number of gates. Each of the gates is assigned some (but not all) probability mass. The remaining probability mass is then assigned to the last gate/expert. This ensures that the probabilities still sum to unity i.e., $\sum_{k=1}^{K} \alpha_k(\boldsymbol{x}_i) = 1$, where K is the large number of experts/truncation level.

Intuitively, LSBP gating network *prunes* the provided set of experts to be used, because of its bias towards the initial set of experts. As an artifact of the Stick-Breaking Process used in this construction, which is one of the possible representations of Dirichlet Process, experts are considered in increasing order of their index, and experts down the queue are only considered if the previous experts were inadequate to explain the labels. This is in contrast to other gating functions, where all experts are treated equally and therefore, it becomes likely that an increasing number of experts will overfit to the training data. We can effectively consider the LSBP gating performing the role of a shrinkage prior. We demonstrate both the shrinkage by LSBP and the overfitting of GG gating networks with an increasing number of experts in our experiments.

The EM updates for LSBP gating can be derived, again by plugging in the gating probabilities into Eq 5. As was the case for PG–SG gating, we need data augmentation for over-coming non-conjugacy due to the softmax. Hence during the E-step, the expected value of the auxiliary variable is as follows:

$$\omega_{ik} = \left(2x_i^T \nu_k^{(t)}\right)^{-1} \tanh\left(0.5 x_i \nu_k^{(t)}\right) \sum_{i=k}^{K} \pi_k(\boldsymbol{x}_i) \tag{17}$$

Then in the M-Step, using the auxiliary variable and gating probabilities from the previous step, the weights of the logistic are updated as follows:

$$\nu_k^{(t+1)} = \left(\mathbf{X}^T \operatorname{diag}(\omega_{1k}, \omega_{2k}, \ldots, \omega_{nk}) + \Sigma_\nu^{-1}\right)^{-1} \left(\mathbf{X}^T \operatorname{diag}(\kappa_{1k}, \kappa_{2k}, \ldots, \kappa_{nk}) + \Sigma_\nu^{-1}\mu_\nu\right) \tag{18}$$

$$\boldsymbol{\Sigma}_\nu^{-1} = \lambda_{invreg}\mathbf{I}^{-1}, \quad \boldsymbol{\mu}_\nu \sim \mathcal{N}(0, \mathbf{I}) \tag{19}$$

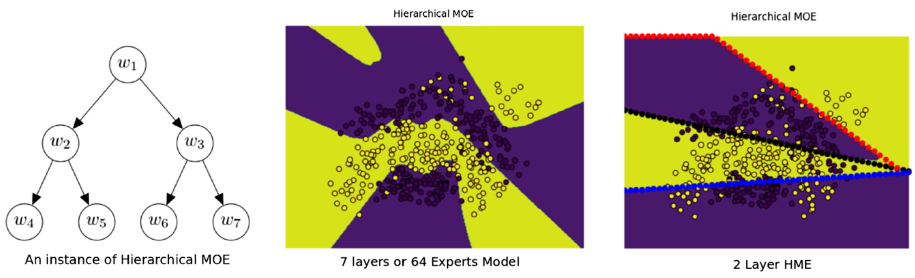

Fig. 1 (Left) **a** A diagrammatic representation of the structure of a 3-level Hierarchical MoE (right). **b** Final decision boundaries learnt by a 7-level Hierarchical MoE on banana dataset and (bottom). **c** Partitions learnt by different nodes of a 3-level Hierarchical MoE (black: root partition, red and blue: children partitions) (Color figure online)

$$\kappa_{ih} = \pi_{ik} - 0.5 \sum_{i=k}^{K} \pi_{ik} \tag{20}$$

4.4 Hierarchical gating (hierarchical mixture of experts)

We now shift our focus to Hierarchical MoEs (**HME**). First proposed in Jordan and Jacobs (1994), it assumes a hierarchical partitioning of the input space, providing an alternative to partitioning provided by the flat gating networks. Despite its success, an efficient algorithm with closed-form parameter updates has eluded Hierarchical MoEs (to the best of our knowledge). Interestingly, we can again leverage the latent-variable augmentations to construct HMEs which can be trained efficiently. The fundamental observation is that all internal nodes (non-leaf nodes) in a hierarchical MoE are essentially performing classification, and thus, can be constructed using Bayesian SVMs. While we focus on the construction of HMEs using Bayesian SVMs, HMEs can similarly be constructed using logistic models while using Pólya-Gamma augmentation.

We will restrict our discussion of Hierarchical MoEs to complete binary trees (of arbitrary depths), as was done in Jordan and Jacobs (1994). In these models, the internal nodes play the function of gating and the leaf nodes play the role of experts. Since we are using only binary nodes, we construct our HME using Bayesian SVMs. We can use multi-class Bayesian SVMs in case of non-binary trees.

Since, the notation for Hierarchical MoEs can get cumbersome, we restrict our discussion to a Hierarchical MoE shown in Fig. 1. Here, $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots \boldsymbol{w}_7]$ denote the weight vectors. Also, we assume the classification labels to be $y \in \{0, 1\}$. If an internal node predicts 0, left child is chosen for the next prediction, or else right. Let $z_{ij} = 1$ denote the event that node j was chosen in the prediction for $\boldsymbol{x}_i$. We can define the recursive probability rule (assume integer division for indices and % denotes the modulo operation)

$$p(z_{ik} = 1|\boldsymbol{x}_i, \mathbf{W}) \propto p\left(k\%2|\boldsymbol{x}_i, \boldsymbol{w}_{i\frac{k}{2}}\right) p(z_{i\frac{k}{2}} = 1|\boldsymbol{x}_i, \mathbf{W}) \tag{21}$$

Note that, $p(y|\boldsymbol{x}_i, \boldsymbol{w}_k) \propto \exp(-2\max(0, 1-2(y-1)\boldsymbol{w}_k^T \boldsymbol{x}_i))$ (Bayesian SVM) and $p(z_{i1} = 1|\boldsymbol{x}_i, \mathbf{W}) = 1$ (the root node is necessarily visited). The final prediction is the weighted combination of predictions by leaf nodes, weights being computed by normalizing $p(z_{ij} = 1|\boldsymbol{x}_i, \mathbf{W})$ for leaf nodes. When learning the model, we will assume z_{ij} to be a latent variable. For leaf nodes, we introduce γ_{ij} as latent variable for Bayesian SVM. However, for all internal nodes, we introduce two latent variables for every example $\boldsymbol{x}_i$, that is $\gamma_{ij}^0, \gamma_{ij}^1$. Internal nodes take both labels (albeit with different probabilities) unlike the leaf nodes which have a deterministic label from the training data. The likelihood function from Eq. 5 has to be changed to the following

$$\mathcal{L}(\boldsymbol{y}, \mathbf{Z}, \boldsymbol{\gamma}|\mathbf{X}, \mathbf{W}, \mathbf{V}) = \sum_{i=1}^{N}\sum_{k=4}^{7} z_{ik}(\log p(y_i, \gamma_{ik}|\boldsymbol{x}_i, \boldsymbol{w}_k) + \log p(z_{ik}=1, \boldsymbol{\gamma}|\boldsymbol{x}_i, \mathbf{W})) \tag{22}$$

Here, $p(z_{ik}=1, \boldsymbol{\gamma}|x_i, \mathbf{W}) \propto p(k\%2, \gamma_{i\frac{k}{2}}^{k\%2}|\boldsymbol{w}_{k/2}, \boldsymbol{x}_i) \times p(z_{i\frac{k}{2}}=1, \boldsymbol{\gamma}|\boldsymbol{x}_i, \mathbf{W})$. Now, we need to compute the following expectations in the E step (besides η_{ik} which retains its definition)

$$\tau_{ik}^j = \mathbb{E}\left[(\gamma_{ik}^j)^{-1}\right] = |1-(j-1)\boldsymbol{x}_i^\top \boldsymbol{w}_k|^{-1} \; (j=0,1, k \notin \text{leaf}) \tag{23}$$

τ_{ik} for leaf nodes retain their original definition. Solving the likelihood, we get the update $\boldsymbol{w}_k = (\mathbf{X}^\top \mathbf{A}_k \mathbf{X} + \lambda \mathbf{I})^{-1}\left(\sum_{i=1}^N b_{ik}\boldsymbol{x}_i\right)$

Here, $\mathbf{A}_k = \text{diag}(a_{1k}, \ldots a_{nk})$, $a_{ik} = \eta_{ik}\tau_{ik}^0 + \eta_{ik}'\tau_{ik}^1$ and $b_{ik} = \eta_{ik}'(1+\tau_{ik}^1) - \eta_{ik}(1+\tau_{ik}^0)$. For internal nodes, η_{ik} represents the posterior probability that $\boldsymbol{x}_i$ goes in the left subtree of node k which is equivalent to the sum of posterior probabilities of experts in its left subtree. Similarly, η_{ik}' represents the posterior probability of right subtree. As a sanity check, following this update for the leaf nodes is equivalent to the MAP estimate in Sect. 2.2.

4.5 Discussion on gating networks

With the construction discussed in the previous sections, we now have a *plug-and-play* setup, where we can switch our gating networks independent of our experts. While it is difficult to say apriori which network will perform better, some properties of these networks merit discussion. **GG**, a fairly robust construction, in part contributed to the success of MoEs (Yuksel et al. 2012; Meeds and Osindero 2006). The problem of having $\mathcal{O}(KD^2)$ parameters can be ameliorated by using diagonal co-variances to an extent as suggested earlier. However, our other gating network architectures (softmax, LSBP, HME) require far fewer parameters to be estimated, without further approximations.

Although the real benefit of augmentation schemes can be seen in **SG–PG** and **HME** constructions. Softmax gating networks previously relied on an IRLS optimization with an EM iteration, which is extremely expensive while Hierarchical models *never* had an algorithm with *closed form updates* and again relied on such iterative schemes for optimization. Both these constructions can be efficiently learned using variable augmentation schemes we have discussed.

We have also discussed a **LSBP** which provides a non-parametric construction to automatically prune experts, reducing the number of hyperparameters associated with the model and thereby relieving the machine learning practitioner of computational burden of tuning the number of experts. This is a significant (and desirable) modeling departure from the current machine learning models/algorithms which seem to be predicated upon an ever-increasing number of hyperparameters. However, there is a tradeoff here. **GG** has only K augmentations, **SG–PG** and LSBP have $2K$ augmentations while **HME** has $3K-2$ Augmentations per data point. More the number of augmentations, more the network becomes susceptible to poor initialization. As expectation–maximization only guarantees convergence to a local maximum (Balakrishnan et al. 2017), poor initialization becomes an issue with increasing augmentations. We can partly resolve this by re-running models with different random initializations. However, better analysis of initialization of augmented models will be a fruitful future endeavor.

5 Related work

Mixture of Expert (MoE) based models have had a rich history in machine learning, starting with the early works by Jacobs et al. (1991), Jordan and Jacobs (1994). An excellent survey can be found in Yuksel et al. (2012). Advances in MoE based models have been in several directions, primarily focusing on better gating networks, e.g., generative gating (Xu et al. 1995; Meeds and Osindero 2006), or in using more powerful/flexible experts, such as Gaussian Processes (Yuan and Neubauer 2009). However, these MoE formulations are difficult to do inference on and, moreover, the experts are limited to probabilistic models. The MoE classification setting is even more challenging because not only the gating network is difficult to learn, learning of the experts too is difficult due to non-conjugacy of the underlying model. The hierarchical MoE (HMoE) model offers a better interpretability than flat MoE. However, inference in HMoE is even more challenging, which has precluded its adoption by practitioners. Our proposed large-margin MoE framework with diverse choices for gating network and efficient closed-form inference in each of the cases is an attempt to address these issues.

From the perspective of MoE's ability to learn nonlinear models, our Bayesian SVM based MoE framework is similar in spirit to the work on kernel-based Bayesian SVMs proposed recently (Henao et al. 2014). However, their model does not admit as simple inference procedure as ours and is also much more computationally expensive as compared to our model. In our experiments, we also compare with this model and our results show that our model achieves much better classification accuracies, despite not using kernels. Therefore, our Bayesian SVM based MoE framework can also be an attractive method for Bayesian nonlinear large-margin classification, without having to use kernels.

6 Experiments

We present experimental results for our proposed MoE framework on several benchmark datasets and compare with various state-of-the-art baselines, both MoE based as well as Bayesian models designed for nonlinear classification. We retain our abbreviations for the four variants from the previous sections.

Firstly, we show a visualization of the decision boundaries learned by our MoE model (with generative gating network) on banana dataset as the number of experts is varied in Fig. 2. It is clear that a larger number of experts are able to generate increasingly non-linear decision boundaries.

6.1 Experimental settings

Our MoE set up consists of two main hyper-parameters. The number of experts for flat gating networks/the number of levels for hierarchical gating network and a regularization parameter for the weight vectors of the expert and the gating networks. Throughout this work, we use zero-mean Gaussian priors for our weight vectors which amounts to L2 regularization over the weight vectors for both gating networks and experts.

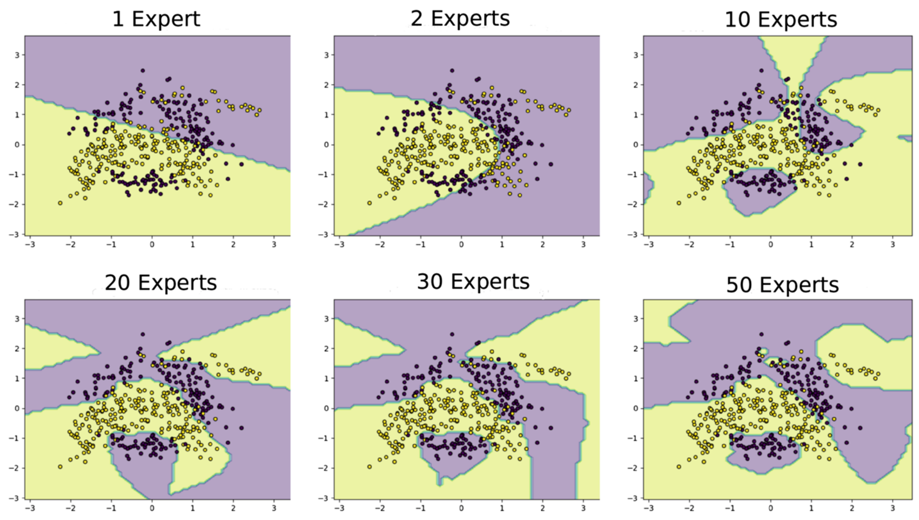

Fig. 2 A visualization of the decision boundaries learnt by our Gaussian Generative Gating model as the number of experts are increased from 1 to 50

Table 1 Description of all the datasets used in the experimentation section, where a hyphen under N_{test} indicates that the dataset did not come with a pre-defined train/test split

Dataset	N_{train}	N_{test}	Dimension
Banana	400	4900	2
Waveform	400	4600	21
Image	1300	1010	18
Breast cancer	200	77	9
Pima	768	–	8
Wisconsin	683	–	9
Sonar	208	–	60
Adult(a8a)	22,696	9865	123
IJCNN	49,990	91,701	22

The weight matrices for our models are initialized using random samples from $\mathcal{N}(0, I)$. We experimented with experts ranging from 2 to 20 for flat models while we varied the number of levels between 3 and 6 (equivalent to 4–32 experts) for hierarchical models. The regularization parameter was tuned between .01 to 100. Finally, for the number of iterations, we set a cap of 25 iterations for generative gating (GG) and 100 for other networks (as GG tends to converge much earlier than others), with a convergence threshold of .01 on the expected complete log-likelihood. In order to tune these hyper-parameters, we perform a simple grid search over some pre-specified values from the respective ranges (specified above) of the hyperparameters.

To ascertain the best combination of hyper-parameters, we perform a cross-validation over the training set, similar to the procedure in Henao et al. (2014). For datasets in the Table 3, we perform a fivefold validation, while for datasets in Table 2, we do a tenfold validation along the lines of Henao et al. (2014) and Zhou (2016) respectively. For the latter, we reuse the first ten pre-defined splits available here.[1] The dataset statistics are summarized in Table 1.

[1] https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets.

Table 2 Classification error rates (Part 1) (results in bold indicate the lowest mean error for the dataset)

Model	Banana	Waveform	Image	Breast cancer
LR	47.76 (4.38)	13.33 (0.59)	17.52 (1.05)	28.05 (3.68)
AMM	18.76 (4.09)	11.81 (1.13)	3.82 (0.87)	31.56 (0.87)
RVM	11.08 (0.69)	11.16 (0.72)	3.82 (0.59)	31.82 (4.66)
RBF-SVM	10.85 (0.57)	10.22 (0.86)	2.84 (0.52)	28.44 (4.52)
Stack-τ	33.21 (5.76)	12.25 (0.69)	7.97 (0.52)	27.92 (3.31)
SS-τ	11.89 (0.61)	11.69 (0.69)	2.73 (0.53)	28.83 (3.40)
OSSVM	12.32 (0.52)	14.31 (0.97)	5.17 (0.65)	28.94 (5.01)
BSVM	11.66 (.81)	11.97 (.36)	3.23 (.55)	30.00 (3.7)
GPC	10.70 (0.58)	32.95 (0.20)	**2.64** (0.3861)	31.68 (3.94)
GG	**10.60 (0.41)**	9.41 (1.5)	3.1 (0.45)	**21.04 (1.91)**
LSBP	11.53 (0.91)	10.49 (1.6)	3.65 (0.94)	21.56 (3.01)
SG–PG	16.23 (1.96)	**8.85 (0.15)**	5.52 (1.11)	21.17 (3.24)
HME	10.62 (0.15)	9.74 (.12)	4.82 (.72)	**21.04 (2.58)**

Our proposed models are in the lower half of the table. Wherever reported, numbers in bracket indicate standard deviation of the error

6.2 Baselines

We compare our model against various MoE and non-linear SVM baselines. We briefly describe these baselines below:

- *Sum-Stacked-Softplus* (SS-τ) classifier and *Stacked-Softplus* classifier (Stack-τ) (Zhou 2016): These are the present state-of-art MoE classification models that use a family of softplus functions, convolving countably infinite stacked gamma distributions. Here SS-τ consists of hierarchies (akin to our Hierarchical MoE), while Stack-τ is flat.
- *Adaptive Multi-Hyper-plane Machines (AMM)* (Wang et al. 2011): AMMs provide a framework to adapt the number of hyper-planes used for classification. It prunes the weights of the learnt model to improve computational efficiency at test time for SVMs.
- *Optimally Sparse SVM* (OSSVM) (Cotter et al. 2013): OSSVM provides an algorithm to sparsify the support vectors learnt for the data when doing non-linear classification with kernels for SVMs. It provides theoretical guarantees for the number of support vectors learnt by the algorithm.
- *Bayesian Non-Linear SVM* (BSVM) (Henao et al. 2014): Bayesian formulation of the SVM Hinge loss which uses a Gaussian Process Classifier. This is optimized using Expectation Conditional Maximization (ECM).
- *GP Classifier* (GPC): Standard off-the-shelf GP based classifier which uses a Gaussian ernel for non-linear classification.
- *Relevance Vector Machines* (RVM) (Tipping 2001): Probabilistic formulation of SVMs using generalized linear models for classification.
- *RBF-Kernel SVM* (RBF-SVM) and *Linear Regression* (LR): Standard baselines for comparison.

Table 3 Classification error rates (Part 2) (results in bold indicate the lowest mean error for the dataset)

Model	pima	wisconsin	sonar	adult(a8a)	IJCNN
LR	24.32 (7.11)	5.7475 (3.00)	22.5 (6.45)	15.00	8.00
AMM	34.21 (4.92)	13.06 (1.23)	16.25 (4.78)	14.88	**1.3**
RVM	24.67 (5.41)	3.67 (3.50)	19.44 (5.83)	14.95	1.29
RBF-SVM	24.22	3.07	11.54	14.6	6
Stack-τ	22.35 (10.25)	5.4375 (3.29)	20 (14.14)	15.00	6.64
SS-τ	21.05 (8.00)	5.51 (3.03)	22.5 (10.40)	15.02	2.24
OSSVM	24.73 (6.76)	13.6 (4.28)	28.5 (10.28)	14.75	1.8
BSVM	21.9	2.93	11.06	-	-
GPC	22.0	2.64	12.50	-	-
GG	18.7 (4.0)	**1.76 (1.1)**	6.36 (5.1)	**14.49**	2.02
LSBP	**17.7 (5.5)**	2.23	10.00 (6.5)	20.02	2.44
SG–PG	18.4 (2.9)	**1.76 (1.3)**	**4.86 (3.2)**	14.81	8.04
HME	20.6 (3.3)	**1.76 (1.1)**	8.5 (4.5)	14.93	4.36

Our proposed models are in the lower half of the table. Wherever reported, numbers in bracket indicate standard deviation of the error. Ones left blank meant that the model didn't scale on a 12 core 60 GBs memory system

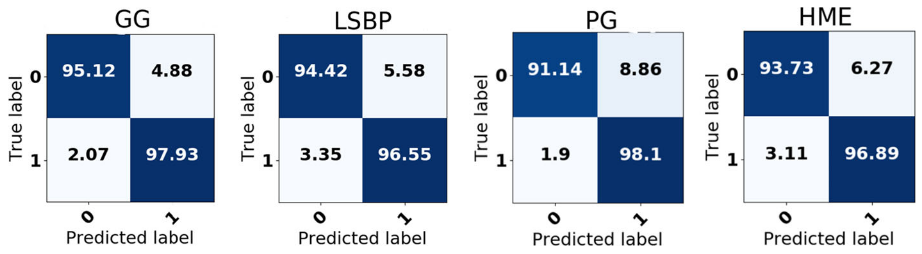

Fig. 3 Confusion matrices for our models on the image dataset (left to right: GG, LSBP, PG, HME)

The results, summarized in Tables 2 and 3 show that most of our models outperform the baselines, on all datasets except *image* and *IJCNN* (where **GG** and **LSBP** come close). Particularly notable is the performance of our models on the *breast cancer* and *Sonar* dataset where all our models outperform the baselines by a margin.

We have also benchmarked against the Small Variance Dirichlet Process Mixture SVM (M^2DPM) proposed by Wang and Zhu (2014), infinite-SVM (Zhu et al. 2011) and dpMNL (Shahbaba and Neal 2009). The comparisons with these models can be found in the "Appendix".

In Fig. 3, we provide a confusion matrix as an analysis of the of the errors made by different gating networks.

6.3 Automatic tuning of experts using LSBP

LSBP gating successfully prunes out the unnecessary experts without overfitting while GG tends to overfit as K increases. This is shown in the left plot in Fig. 4 which compares GG and LSBP, indicating the robustness of LSBP to increasing number of experts. For the graph on the right, we compared the distribution of training data assignment to the experts

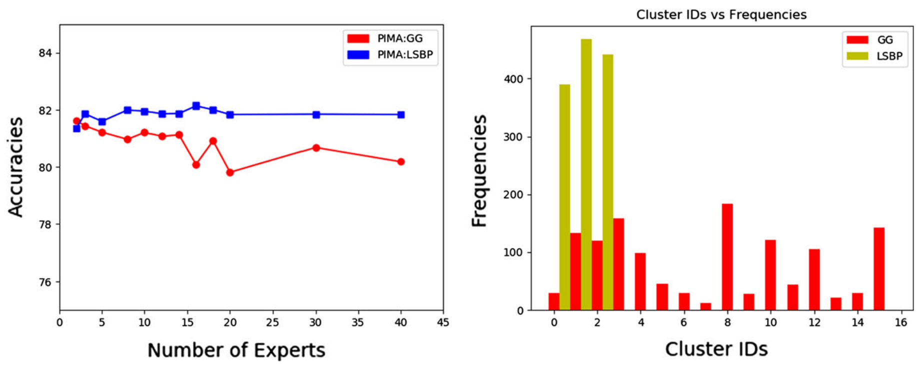

Fig. 4 *Left*: test accuracy versus # of experts in GG and LSBP models *Right*: number of points assigned to each expert in banana dataset with 10 experts

when we specify a large number of experts ($K = 16$). Here, LSBP gating model assigns the data points to only the first three experts whereas GG gating distributes data point among all the experts. This clearly shows the LSBP model's ability to do shrinkage and provide comparable performance with a smaller # of experts.

6.4 Timing experiments

6.4.1 Hierarchical MoE

We first compare our HME (VA-HME) with a traditional HME (BFGS-HME) formulation of the same structure, which approximately follows Jordan and Jacobs (1994). For the traditional HME, we have a softmax classifier for both internal and leaf nodes. Within the EM iterations, the weights of the nodes are optimized using L-BFGS. We carry out a fixed number of EM iterations (100). The experiment is repeated 20 times for every tree level, varied from 2 to 7. We report both real computation time, and CPU time. Note, both codes were implemented in python, using numpy and scipy. In particular, no effort was made to parallelize the code (although inbuilt library functions could have, due to which the difference in real time is observed). From the data, our HME model is nearly 5 times faster in real time and 2.5 times faster in computation time (Fig. 5).

6.4.2 Flat MoE

To emphasize the role of closed-form (CF) updates, we do a runtime comparison of different MoE models. We compare with GG + Logistic Regression (LR) Experts, where LR experts are optimized using L-BFGS. Similarly, we compare with Softmax Gating (SG) with Bayesian SVM experts, where SG is optimized using L-BFGS. Also, we do a comparison with SG+LR which does gradient descent (using Adam) directly on the incomplete likelihood.

Note, that our models with closed-form updates, which are implemented in Python + NumPy are at a significant disadvantage to the rest of these implementations, which have highly optimized C-based backend implementation (PyTorch for Adam, SciPy for L-BFGS). Yet, from Table 4, we can see that (1) our BSVM based MoE models are much faster than traditional LR based MoE that relies on iterative update (no closed-form updates), and (2) L-BFGS scales rather poorly.

Table 4 Running-time (in seconds) for different models as a function of number of experts on image dataset [Format: User Time (System Compute Time)]

Model/Experts	Optimization	4	16	64	128
GG+BSVM	CF+CF	2.81 (4.25)	5.54 (9.52)	19.08 (35.08)	32.65 (56.76)
LSBP+BSVM	CF+CF	7.44 (12.28)	22.50s (36.73)	55.40 (71.9)	97 (121.96)
SG+BSVM (Reg $=$ .1)	L-BFGS+CF	14.41 (17.25)	52.00 (56.99)	198.05 (223)	426.09 (488.36)
SG+BSVM (Reg $=$ .01)	L-BFGS+CF	14.50 (15.78)	57.32 (68.78)	222.46 (261.37)	394.01 (435.47)
GG+LR	CF+L-BFGS	5.25 (8.43)	16.12 (17.23)	60.89 (65.82)	141.05 (158.48)
SG+LR	Adam (MLE)	22.45 (.99)	41.59 (1.09)	93.47 (1.22)	114.87 (1.40)

Table 5 A comparison of training times of generative gating with several baseline models and varying data dimensionality

Dimensions	GG	GG+LR	SG+BSVM(0.1)	SG+BSVM (0.01)	Kernel-SVM
18	5.25 (8.00)	18.35 (15.53)	50.21 (57.5)	82.75 (111.22)	44.23 (1.53)
36	7.67 (11.24)	21.52 (15.72)	64.84 (79.54)	133.55 (172.84)	50.17 (1.64)
54	9.38 (13.63)	24.47 (18.96)	94.88 (110.51)	186.85 (229.42)	68.64 (1.64)
108	19.92 (26.23)	36.84 (25.68)	98.14 (100.81)	202.01 (229.92)	71.45 (2.00)

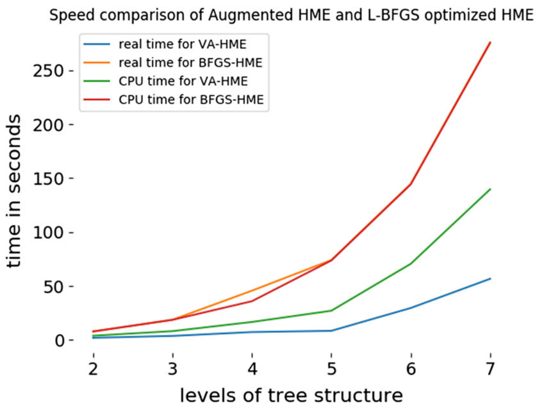

Fig. 5 Timing comparison of our proposed inference algorithm for HME and traditional EM-BFGS based algorithm for HME

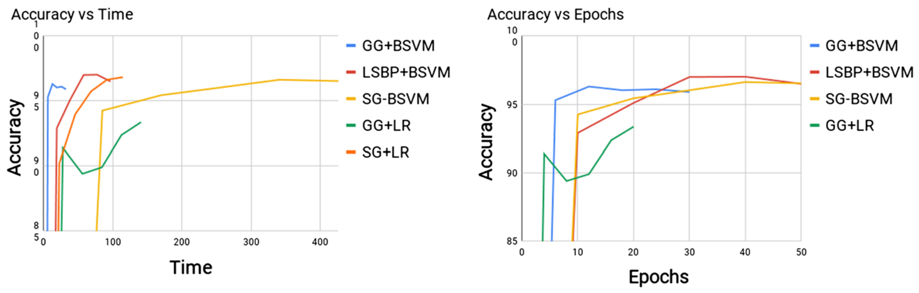

Fig. 6 Training accuracy as a function of time (left) and as a function of iterations (right) with 128 experts for all models

Next, we evaluate the training time of Generative Gating as the data dimensionality increases in Table 5. This shows how our models scale with dimensions when compared with other baselines and also with the standard Kernel-SVM. Our proposed models still present themselves as the computationally efficient alternative.

Finally, we compare the number of iterations required for different iterative algorithms to converge. As can be seen from Fig. 6 (right), our proposed models and the competing models have a comparable number of iterations needed to reach maximum accuracies (except for the Adam optimized SG+LR MoE model that needs a large number of iterations).

We also visualize the training accuracies as a function of computing time in Fig 6(left). This figure again illustrates the fact that both of our models can achieve higher accuracies in a lesser amount of time, which further strengthens the argument for closed form updates.

6.5 On role of Bayesian SVMs

We highlight the role of Bayesian SVMs. Particularly, we compare against Logistic Regression experts (GG+LR) instead of Bayesian SVM (GG+BSVM), and a non-closed-form L-BFGS update version of Softmax Gating which still uses Bayesian SVM (SG+BSVM). The results are summarized in Table 6. Bayesian SVMs consistently increase the accuracies when compared to Logistic Regression experts. SG + BSVM also seems to give similar performance to GG + BSVM, which indicates that Bayesian SVMs are responsible for the performance boost in our earlier experiments.

Table 6 Error rate comparison to show the significance of Bayesian SVM as experts (CF: Closed Form Updates) on Set I

Model	Optimization	Banana	Waveform	Image	Breast cancer
GG+BSVM	CF+CF	10.60 (0.41)	9.41 (1.5)	3.1 (0.45)	21.56 (3.01)
GG+LR	CF+L-BFGS	11.02 (0.75)	11.94 (0.42)	8.57 (0.61)	23.37 (2.96)
SG+BSVM	L-BFGS+CF	10.78 (0.36)	10.10 (0.32)	2.94 (0.48)	22.94 (5.59)

7 Conclusion

We have presented a novel mixture of experts based framework which uses large-margin classifier like SVM as experts by using the Bayesian formulation of SVM, and consists of a variety of gating networks, each of which admits a simple inference procedure with closed-form updates for the model parameters. One of the gating networks (LSBP) also allows learning the number of experts from data, while the hierarchical MoE offers more interpretability as compared to MoE with gating networks that rely on flat partitioning. Our framework, with each of the gating networks, also enjoys conjugacy in all of its components (both Bayesian SVM as well as the gating network), which makes it easy to do inference. In addition to providing a rich and flexible framework for learning MoE based models, as our experiments demonstrate, our framework is also competitive to Bayesian nonlinear classification models that use kernels (Henao et al. 2014).

Acknowledgements Piyush Rai acknowledges support from Visvesvaraya Young Faculty Research Fellowship from MEITY India.

A Appendix

A.1 MAP estimate for naive mixture of experts with Bayesian SVMs

Let $\boldsymbol{\Theta} = \{\theta_g, \theta_e\}$ where $\theta_g = \{\boldsymbol{v}_i\}_{i=1}^K$ represents the vectors for gating network of K experts and $\theta_e = \{\boldsymbol{w}_i\}_{i=1}^K$ represents the weight vectors for K experts (modelled using Bayesian SVMs), $\boldsymbol{\gamma}_i = \{\gamma_{i1}, \gamma_{i2}, \ldots \gamma_{iN}\}$ represent the per-example augmentations representation for the expert i and z_i is a one-hot vector denoting the allocation of example i to an expert. Note, $\mathcal{D} = \{\mathbf{X}, \boldsymbol{y}\}$ denotes the training data as usual.

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

$$\begin{aligned}
p(\boldsymbol{\Theta}, \boldsymbol{\gamma}, z|D) &\propto p(\boldsymbol{\Theta}) p(y, \boldsymbol{\gamma}, z|\mathbf{X}, \boldsymbol{\Theta}) \\
\log p(\boldsymbol{\Theta}, \boldsymbol{\gamma}, z|D) &\propto \log p(\boldsymbol{\Theta}) \\
&\quad + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1](\log p(z_{ij} = 1|x_i, \theta_g) + \log p(y_i, \boldsymbol{\gamma}_{ij}|x_i, \theta_e, z_{ij} = 1)) \\
\mathbb{E}[\log p(\boldsymbol{\Theta}, \boldsymbol{\gamma}, z|D)] &\propto \log p(\boldsymbol{\Theta}) \\
&\quad + \sum_{i=1}^{N}\sum_{j=1}^{K} \mathbb{E}[z_{ij}](\log p(z_{ij} = 1|x_i, \theta_g) + \mathbb{E}[\log p(y_i, \boldsymbol{\gamma}_{ij}|x_i, \theta_e, z_{ij} = 1)])
\end{aligned} \tag{24}$$

There is an implicit assumption in this: The posterior expectation of z_{ij} does not depend on $\boldsymbol{\gamma}$. In the current setup, we are using the softmax gating. We get:

$$\begin{aligned}\mathbb{E}[z_{ij}] &= p(z_{ij} = 1 | x_i, y_i, \boldsymbol{\Theta}^{(t)}) \\ &\propto p(y_i | x_i, z_{ij} = 1, \theta_e^{(t)}) p(z_{ij} | x_i, \theta_g^{(t)}) \\ &\propto \exp(-2 \max(0, 1 - y_i x_i^T w_j^{(t)})) \exp(x_i^T v_j^{(t)}) \\ \mathbb{E}[z_{ij}] &= \frac{\exp(x_i^T v_j^{(t)} - 2\max(0, 1 - y_i x_i^T w_j^{(t)}))}{\sum_{l=1}^K \exp(x_i^T v_l^{(t)} - 2\max(0, 1 - y_i x_i^T w_l^{(t)}))} = \eta_{ij} \end{aligned} \tag{25}$$

Now, onto the other expectation:

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

Here,

$$\begin{aligned}\log \left[\frac{1}{\sqrt{2\pi\boldsymbol{\gamma}_{ij}}} \exp\left(-\frac{(1 + \boldsymbol{\gamma}_{ij} - y_i w_j^T x_i)^2}{2\boldsymbol{\gamma}_{ij}} \right) \right] &= \frac{-1}{2} \log 2\pi - \frac{1}{2} \log \boldsymbol{\gamma}_{ij} - \frac{(1 + \boldsymbol{\gamma}_{ij})^2}{2\boldsymbol{\gamma}_{ij}} \\ &\quad - \frac{(y_i w_j^T x_i)^2}{2\boldsymbol{\gamma}_{ij}} + \frac{y_i w_j^T x_i (1 + \boldsymbol{\gamma}_{ij})}{\boldsymbol{\gamma}_{ij}}\end{aligned}$$

We only care about the terms involving w_j, that is, only the last two terms. Therefore, we only care about the expectation $\mathbb{E}[\frac{1}{\boldsymbol{\gamma}_{ij}}] = |1 - y_i x_i^T w_j^{(t)}|^{-1} = \tau_{ij}^{-1}$ (Polson et al. 2011). Therefore, the $\mathbb{E}[\log p(y_i, \boldsymbol{\gamma}_{ij} | x_i, \theta_e, \boldsymbol{z}_{ij} = 1)]$ 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 (24) to get the final objective (note we are ignoring the terms not involving w_j). Replacing the appropriate expectations in (24), the final objective for the problem is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)}) &= \log p(\boldsymbol{\Theta}) \\ &\quad + \sum_{i=1}^N \sum_{j=1}^K \eta_{ij} \Big[\log \frac{\exp(v_j^T x_i)}{\sum_{l=1}^K \exp(v_l^T x_i)} - \frac{(y_i w_j^T x_i)^2 - 2y_i w_j^T x_i}{2\tau_{ij}} + 2y_i w_j^T x_i \Big]\end{aligned} \tag{26}$$

We maximize this objective, that is, $\boldsymbol{\Theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})$. 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,

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

$$\Rightarrow \left(\lambda w_j + \sum_{i=1}^{N} \frac{\eta_{ij}}{\tau_{ij}} (w_j^T x_i) x_i\right) = \sum_{i=1}^{N} \eta_{ij} \left(\frac{y_i x_i}{\tau_{ij}} + y_i x_i\right) \tag{27}$$

$$\Rightarrow w_j = (\mathbf{X}^T \mathbf{A}_j \mathbf{X} + \lambda \mathbf{I})^{-1} \left(\sum_{i=1}^{N} \eta_{ij} \frac{\tau_{ij} + 1}{\tau_{ij}} y_i x_i\right) \tag{28}$$

where $\mathbf{X}$ 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:

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t)})}{\partial v_j} &= -\beta v_j + \sum_{i=1}^{N} \eta_{ij} x_i - \sum_{i=1}^{N} \sum_{j=1}^{K} \eta_{ij} \frac{\partial \log \sum_{l=1}^{K} \exp(v_l^T x_i)}{\partial v_j} \\ &= \sum_{i=1}^{N} \left[\eta_{ij} - \frac{\exp(v_j^T x_i)}{\sum_{l=1}^{K} \exp(v_l^T x_i)}\right] x_i - \beta v_j \end{aligned} \tag{29}$$

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 (Xu et al. 1995), while the second approach uses the ‘Polya-Gamma Data Augmentation’ for the softmax weight vectors (Polson et al. 2013; Scott and Sun 2013).

A.2.1 Generative gating networks

The results derived in this section borrows some results from Xu et al. (1995), and the previous section. We wish to maximize $\mathbb{E}_{p(\boldsymbol{\gamma}, z|\boldsymbol{\Theta}^{(t)}, D)}[\log p(y, x, \boldsymbol{\gamma}, z|\boldsymbol{\Theta})]$ with respect to $\boldsymbol{\Theta}$. Here,

$$\begin{aligned}\log p(y, x, \boldsymbol{\gamma}, z|\boldsymbol{\Theta}) &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1]\Big(\log p(y_i, \boldsymbol{\gamma}_{ij}|x_i, w_j) + \log \alpha_j + \log p(x_i|\theta_g, z_{ij} = 1)\Big) \\ \mathbb{E}[\log p(y, x, \boldsymbol{\gamma}, z|\boldsymbol{\Theta})] &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{E}[z_{ij}]\Big(\mathbb{E}[\log p(y_i, \boldsymbol{\gamma}_{ij}|x_i, w_j)] + \log \alpha_j + \log p(x_i|\theta_g, z_{ij} = 1)\Big)\end{aligned} \tag{30}$$

Now,

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

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 the update for w in (28):

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

The updates for the parameters of the gating network are borrowed from Xu et al. (1995). 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:

$$\alpha_j^{(t+1)} = \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)}}{N} = \frac{N_j}{N}$$

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)} x_i}{N_j}$$

$$\boldsymbol{\Sigma}_j^{(t+1)} = \frac{\sum_{i=1}^{N} \eta_{ij}^{(t)} (x_i - \boldsymbol{\mu}_j^{(t)})(x_i - \boldsymbol{\mu}_j^{(t)})^T}{N_j}$$

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 (Polson et al. 2013; Scott and Sun 2013). For every example and for every weight vector, we augment another set of latent variables $\beta_{ij} \sim PG(1, 0)$. 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 (which can easily be converted to MAP estimation). We want to maximize $\mathbb{E}[\log p(y, \boldsymbol{\gamma}, z, \beta | \boldsymbol{\Theta}, \mathbf{X})]$, where the expectation is with respect to $p(\boldsymbol{\gamma}, \beta, z | \boldsymbol{\Theta}^{(t)}, \mathbf{X}, y)$:

$$\begin{aligned}\log p(y, \boldsymbol{\gamma}, \beta, z | \boldsymbol{\Theta}, \mathbf{X}) &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1] \log p(y_i, z_{ij} = 1, \beta_i, \boldsymbol{\gamma}_{ij} | \boldsymbol{\Theta}, x_i) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{1}[z_{ij} = 1] \Big(\log p(y_i, \boldsymbol{\gamma}_{ij} | \boldsymbol{\Theta}, x_i, z_{ij} = 1) \qquad (31) \\ &\quad + \log p(\beta_{ij}, z_{ij} = 1 | \boldsymbol{\Theta}, x_i) \Big)\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\log p(y, \boldsymbol{\gamma}, \beta, z | \boldsymbol{\Theta}, \mathbf{X})] &= \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbb{E}[z_{ij}] \Big(\mathbb{E}[\log p(y_i, \boldsymbol{\gamma}_{ij} | \boldsymbol{\Theta}, x_i, z_{ij} = 1)] \qquad (32) \\ &\quad + \mathbb{E}[\log p(\beta_{ij}, z_{ij} = 1 | \boldsymbol{\Theta}, x_i)] \Big) \qquad (33)\end{aligned}$$

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

$$\mathbb{E}[\mathbb{1}[z_{ij} = 1] \log p(y_i, \boldsymbol{\gamma}_{ij} | \boldsymbol{\Theta}, x_i, z_{ij} = 1)] = \mathbb{E}[z_{ij}] \mathbb{E}[\log p(y_i, \boldsymbol{\gamma}_{ij} | \boldsymbol{\Theta}, x_i, z_{ij} = 1)] \qquad (34)$$

This, in general, is incorrect. But, factorization has been used in all the EM derivations carried out above (the factors are slightly different). The justification for this step is as follows (LHS $= \mathbb{E}[\mathbb{1}[z_{ij}=1]\log p(y_i,\boldsymbol{\gamma}_{ij}|\boldsymbol{\Theta},x_i,z_{ij}=1)]$):

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

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 Scott and Sun (2013), we get:

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

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

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

The discussion from the first section applies directly over here. The results for η and w are directly applicable in this derivation. Now, $\beta_{ij}^{(t)} = \mathbb{E}[\beta_{ij}] = \frac{1}{2\psi_{ij}^{(t)}}\tanh\psi_{ij}^{(t)}$ (Scott and Sun 2013). The coupling of parameters mandates the usage of Expectation Conditional Maximization (ECM), when maximizing with respect to v_j (Scott and Sun 2013). 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 $[0,0\ldots 0]^T$ throughout iterations. Therefore,

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

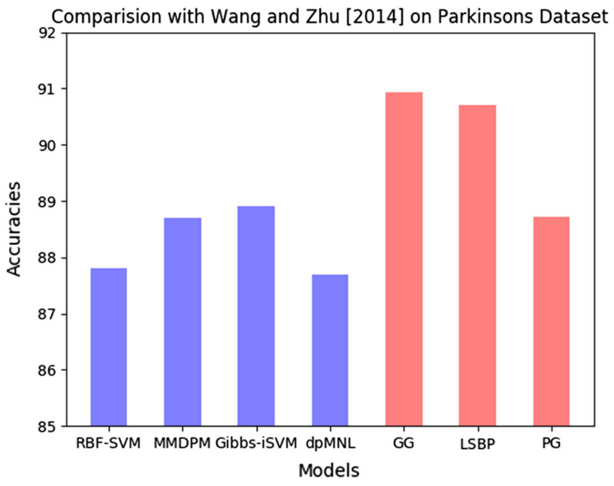

Fig. 7 Accuracies on the Parkinsons Dataset compared to different non-linear SVMs, Wang and Zhu (2014), Zhu et al. (2011) and Shahbaba and Neal (2009)

$$\sum_{i=1}^{N} \eta_{ij}\Big(\frac{1}{2} + \beta_{ij}^{(t)} \log \sum_{l=1,l\neq j}^{K} \exp(x_i^T \hat{v}_l)\Big)x_i = \Big(\sum_{i=1}^{N} \beta_{ij}^{(t)} \eta_{ij} x_i x_i^T\Big) v_j$$
$$\Rightarrow v_j^{(t+1)} = (\mathbf{X}^T \boldsymbol{\Omega}_j \mathbf{X})^{-1} \mathbf{X}^T \kappa_j \tag{35}$$

where,

$$\kappa_j = \left[\eta_{1j}\left(\frac{1}{2} + \beta_{1j}^{(t)} \log \sum_{l=1,l\neq j}^{N} \exp(x_1^T \hat{v}_l)\right), \ldots, \eta_{Nj}\left(\frac{1}{2} + \beta_{Nj}^{(t)} \log \sum_{l=1,l\neq j}^{N} \exp(x_N^T \hat{v}_l)\right)\right]^T \tag{36}$$

$$\boldsymbol{\Omega}_j = \mathrm{diag}(\beta_{1j}^{(t)} \eta_{1j}, \ldots \beta_{Nj}^{(t)} \eta_{Nj}) \tag{37}$$

As is clear, we have closed form updates for all steps and parameters.

A.3 Other classification models

We also consider the Small Variance Dirichlet Process Mixture SVM (M^2DPM) proposed by Wang and Zhu (2014), infinite-SVM (Zhu et al. 2011) and dpMNL (Shahbaba and Neal 2009). M^2DPM is an efficient version of infinite-SVM, which in turn, combines a Dirichlet process with Kernel-SVMs to learn the number of experts, as opposed to our approach of combining different experts for non-linear learning. In particular, infinite-SVM is closely related with the LSBP gating network. We use the Parkinson's disease dataset, which consists of 195 subjects that may or may not have the disease. We follow Wang and Zhu (2014) for reporting accuracies, where we have used a tenfold Cross Validation and report the mean and standard deviation. The results are reported in Fig. 7. As it can be seen, our LSBP gating with BSVM outperforms both M^2DPM and infinite-SVM learned by Gibbs sampling.

References

Balakrishnan, S., Wainwright, M. J., Yu, B., et al. (2017). Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, *45*(1), 77–120.

Bishop, C. M., & Svenskn, M. (2002). Bayesian hierarchical mixtures of experts. In: *UAI*.

Cotter, A., Shalev-Shwartz, S., & Srebro, N. (2013). Learning optimally sparse support vector machines. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (pp. 266–274).

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society Series B (Methodological)*, *39*, 1–38.

Henao, R., Yuan, X., & Carin, L. (2014). Bayesian nonlinear support vector machines and discriminative factor modeling. In: *Advances in neural information processing systems* (pp. 1754–1762).

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*(2), 181–214.

Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review*, *42*(2), 275–293.

Meeds, E., & Osindero, S. (2006). An alternative infinite mixture of Gaussian process experts. In: *Advances in neural information processing systems* (pp. 883–890).

Nickisch, H., & Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, *9*, 2035–2078.

Polson, N. G., Scott, S. L., et al. (2011). Data augmentation for support vector machines. *Bayesian Analysis*, *6*(1), 1–23.

Polson, N. G., Scott, J. G., & Windle, J. (2013). Bayesian inference for logistic models using pólya-gamma latent variables. *Journal of the American statistical Association*, *108*(504), 1339–1349.

Rahimi, A., & Recht, B. (2008) Random features for large-scale kernel machines. In: *Advances in neural information processing systems* (pp. 1177–1184).

Ren, L., Du, L., Carin, L., & Dunson, D. (2011). Logistic stick-breaking process. *Journal of Machine Learning Research*, *12*, 203–239.

Rigon, T., & Durante, D. (2017). Tractable Bayesian density regression via logit stick-breaking priors. ArXiv e-prints arXiv:1701.02969.

Scott, J. G., & Sun, L. (2013). Expectation–maximization for logistic regression. ArXiv preprint arXiv:1306.0040.

Shahbaba, B., & Neal, R. (2009). Nonlinear models using Dirichlet process mixtures. *Journal of Machine Learning Research*, *10*, 1829–1850.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. ArXiv preprint arXiv:1701.06538.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Wang, Z., Djuric, N., Crammer, K., & Vucetic, S. (2011). Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification. In: *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 24–32). ACM.

Wang, Y., & Zhu, J. (2014). *Small-variance asymptotics for dirichlet process mixtures of SVMs*. Palo Alto: AAAI.

Williams, C. K., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In: *Advances in neural information processing systems* (pp. 682–688).

Xu, L., Jordan, M. I., & Hinton, G. E. (1995). An alternative model for mixtures of experts. In: *Advances in neural information processing systems* (pp. 633–640).

Yuan, C., & Neubauer, C. (2009). Variational mixture of Gaussian process experts. In: *Advances in neural information processing systems* (pp. 1897–1904).

Yuksel, S. E., Wilson, J. N., & Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(8), 1177–1193.

Zhou, M. (2016). Softplus regressions and convex polytopes. ArXiv e-prints arXiv:1608.06383.

Zhu, J., Chen, N., & Xing, E.P. (2011). Infinite SVM: A dirichlet process mixture of large-margin kernel machines. In: *ICML*.