A Flexible Framework for Large Margin Mixture-of-Experts

Archit Sharma, Siddhartha Saxena, Dr. Piyush Rai

Indian Institute of Technology, Kanpur

Table of Contents

Introduction

Mixture-of-Experts

Mixture-of-Experts [2] is powerful framework, which essentially pools the effort of relatively simple experts to model a harder problem. There are two components for a mixture of experts framework:

- *Experts*: Local learner, usually simple, models a subset of input data.
- *Gating Network*: Maps the input to the expert.

These models are trained using Expectation Maximization (EM), as the input-expert assignment is not known.

A Generic Formulation of Mixture-of-Experts

In the most frequently occuring setting, we are a given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$. Assume K number of experts.

- *Gating Network*: Parameterized by $V = \{v_k\}_{k=1}^K$. We define expert assignment probability as $\pi_k(x) = f_k(x, V)$ such that $\sum_{k=1}^K f_k(x, V) = 1$ and $0 \leq f_k(x, V) \leq 1$.
- *Expert Network*: Parametrized by $W = \{w_k\}_{k=1}^K$. The prediction of k^{th} expert is given by $\hat{y}_k = g_k(x, w_k)$. For example, $g_k(x, w_k) = w_k^T x$. For the purposes of training, it is necessary to have a probabilistic interpretation for each expert.

The final prediction is given by $\hat{y} = h\big(\sum_{k=1}^K \pi_k(x) g_k(x, w_k)\big)$. Alternately, one can use the prediction from the most probable expert.

EM for Mixture-of-Experts

Input-Expert assignment not known. Introduce latent variable $Z = \{z_i\}_{i=1}^N$, where z_i is a one-hot vector representing expert assignment of x_i. The complete log-likelihood is given by

Complete Log-Likelihood

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

Note, $p(z_{ij} = 1|x_i j, V) = \pi_k(x_i)$ and $p(y_i|x_i, w_j)$ is the expert likelihood model.
We maximize the $\mathbb{E}_Z[\mathcal{L}(y, Z|X, V, W)]$, where $Z \sim p(Z|y, X, W^{(t)}, V^{(t)})$.

Benefits and Shortcomings

Benefits:

- Interpretable Non-Linear Learning
- Parametric, hence, fast at train and test time.
- Fully Bayesian treatment possible.

Shortcomings:

- Experts need to have a probabilistic interpretation, precludes powerful experts like SVMs and NNs.
- Inference of parameters can be slow due to multiple embedded iterative procedures.

We focus on the second shortcoming.

Leveraging Latent Variable Augmentations I: Bayesian SVM

- *Support Vector Machines* (SVM) [1] are extremely popular algorithms for binary classification
- Can be extended to regression, multi-class classification and non-linear learning.
- Hyperparamaters difficult to tune when using Kernels.
- Prone to overfitting.

The SVM objective can be written as

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

which needs to be minimized for estimating w.

Leveraging Latent Variable Augmentations I: Bayesian SVM

Bayesian SVM was formulated by Polson et al. [4]. They showed that

Bayesian SVM

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

This inspired that data augmentation scheme, which allows casts SVM objective into the following posterior maximization/estimation problem, something which is well studied by the Bayesians:

$$L(w, \gamma_i|\lambda, x_i) \propto p(w|\lambda)\prod_{i=1}^{N} L(y_i, \gamma_i|w, x_i)$$

$$L(y_i, \gamma_i|w, x_i) = \mathcal{N}(1 - y_i x_i^T w_i| - \gamma_i, \gamma_i)$$

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

Leveraging Latent Variable Augmentations II: Pólya-Gamma Augmentation

Consider the logistic regression likelihood model,

$$y_i \sim \text{Bern}(p_i)$$
$$p_i = \frac{1}{1 + \exp(-w^T x_i)}$$

Polya-Gamma augmentation [3, 5] augments latent variables β to give a Bayesian treatment to logistic regression. In particular, [3] shows that

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

where $p(\beta)$ represents the Pólya-Gamma distribution. This augmentation morphs the Logit likelihood model into a Gaussian likelihood model, allowing a Fully Bayesian/EM based treatment of the model.

Proposed Model Architectures

Flat Mixture-of-Experts

Bayesian SVMs are proposed as local learners in a Mixture of Experts model in this work. Four gating network architectures are experimented with:

- Softmax Gating Network
- Generative Gating Network
- Polya-Gamma augmented Softmax Gating Network
- Logistic Stick Break Prior Gating Network

The proposed models are trained using Expectation Maximization (EM). MCMC routines are easy to derive. Assume $W = \{w_i\}_{i=1}^{K}$, as the K Bayesian SVM experts.

Softmax Gating Network

The most naive architecture for Softmax Gating Network, as proposed in [2]. Assume $V = \{v_i\}_{i=1}^{K}$ softmax gating vectors. The probability of input x_i being assigned to expert j is given by

$$\pi_j(x_i) = \frac{\exp(v_j^T x_l)}{\sum_{l=1}^{K} \exp(v_l^T x_i)}$$

Major Drawback: *No closed form updates for the gating vector.*

Softmax Gating Network

EM Algorithm

E Step:

$$\eta_{ij} \leftarrow \exp(x_i^T v_j - 2\max(0, 1 - y_i x_i^T w_j))$$

$$\eta_{ij} \leftarrow \frac{\eta_{ij}}{\sum_{l=1}^{K} \eta_{il}}$$

$$\tau_{ij} \leftarrow |1 - y_i x_i^T w_j|^{-1}$$

M Step:

$$A_j \leftarrow diag(\frac{\eta_{1j}}{\tau_{1j}} \dots \frac{\eta_{Nj}}{\tau_{Nj}})$$

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

$$Iterate : v_j \leftarrow v_j - \alpha(\sum_{i=1}^{N} \left[\eta_{ij} - \frac{\exp(v_j^T x_i)}{\sum_{l=1}^{K} \exp(v_l^T x_i)}\right] x_i - \beta v_j)$$

Generative Gating Network

Generative gating network makes two major changes:

- *Gating network*: Each gate models the input (much like Gaussian Mixture Models). The gating parameters are $\{\alpha_k, \mu_k, \Sigma_k\}_{k=1}^K$ and $\pi_j(x_i) \propto \alpha_j \mathcal{N}(x_i|\mu_j, \Sigma_j)$.
- *Changed Objective*: The model now maximizes $\mathbb{E}[\log p(y, X, \gamma, z|\Theta)]$ instead of $\mathbb{E}[\log p(y, \gamma, z|X, \Theta)]$. Here, γ, z are latent variables corresponding to Bayesian SVM and input-expert assignment.

Drawbacks: The number of parameters has increased significantly. The objective proposed be solved is harder, and indirect to what we are interested in.

Generative Gating Network

EM Algorithm

E Step:

$$\eta_{ij} \leftarrow \exp(-2\max(0, 1 - y_i x_i^T w_j))\mathcal{N}(x_i|\mu_j, \Sigma_j)\alpha_j$$

$$\eta_{ij} \leftarrow \frac{\eta_{ij}}{\sum_{l=1}^{K} \eta_{il}}$$

$$\tau_{ij} \leftarrow |1 - y_i x_i^T w_j|^{-1}$$

M Step:

$$A_j \leftarrow diag(\frac{\eta_{1j}}{\tau_{1j}} \dots \frac{\eta_{Nj}}{\tau_{Nj}})$$

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

$$\alpha_j \leftarrow \frac{\sum_{i=1}^{N} \eta_{ij}}{N}, \ \mu_j \leftarrow \frac{\sum_{i=1}^{N} \eta_{ij} x_i}{\sum_{i=1}^{N} \eta_{ij}}, \ \Sigma_j \leftarrow \frac{\sum_{i=1}^{N} \eta_{ij}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{N} \eta_{ij}}$$

Pólya-Gamma Augmented Softmax Gating Network

We revert back to a softmax like construction, that is $V = \{v_k\}_{k=1}^K$ and $\pi_k(x) = \frac{\exp(x_k^w)}{\sum_{l=1}^K \exp(x^T w_L)}$. However, we introduce Pólya-Gamma augmentation, that is $\beta_{ij} \sim PG(1, 0)$ for all x_i and $j \in \{1, \dots K\}$. This augmentation can be used to "gaussianize" multinomial regression. We can get closed for updates for softmax gating networks using this. *Drawbacks*: Multiple augmentation, potentially unstable and more initialization dependent.

Pólya-Gamma Augmented Softmax Gating Network

EM Algorithm

E Step:

$$\eta_{ij} \leftarrow \exp(x_i^T v_j - 2\max(0, 1 - y_i x_i^T w_j))$$

$$\psi_{ij} \leftarrow x_i^T v_j - \log \sum_{l=1, l\neq j}^{K} \exp(x_i^T v_l)$$

$$\eta_{ij} \leftarrow \frac{\eta_{ij}}{\sum_{l=1}^{K} \eta_{il}},\ \tau_{ij} \leftarrow |1 - y_i x_i^T w_j|^{-1},\ \beta_{ij} \leftarrow \frac{1}{2\psi_{ij}} \tanh(0.5\psi_{ij})$$

M Step:

$$A_j \leftarrow diag(\frac{\eta_{nj}}{\tau_{nj}})_{n=1}^{N},\ \Omega_j \leftarrow diag(\beta_{nj}\eta_{nj})_{n=1}^{N}$$

$$\kappa_j^T \leftarrow [\eta_{nj}(\frac{1}{2} + \beta_{nj} \log \sum_{l=1, l\neq j}^{N} \exp(x_n^T \hat{v}_l))]_{n=1}^{N}$$

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

Logistic Stick Breaking Prior Network

LSBP is a non-parametric construction provides the ability to extract the "right" number of experts from the data itself. The set of parameters to be learnt are $V = \{v_i\}$. The expert assignment probability is defined as:

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

$$\nu_k(x_i) = \frac{1}{1 + \exp(-v_k^T x_i)}$$

As the name suggest, we again have a logistic likelihood defined in the gating network. We again introduce Pólya-Gamma latent variables $\beta_{ij} \sim PG(b, 0)$.

Logistic Stick Breaking Prior Gating Network

EM Algorithm

E Step:

$$\psi_{ij} \leftarrow x_i^T v_j,\ \nu_j(x_i) = \frac{1}{1+\exp(v_k^T x_i)}$$

$$\eta_{ij} \leftarrow \left[\exp(-2\max(0, 1-y_i x_i^T w_j))\right]\left[\nu_j(x_i)\prod_{l=1}^{j-1}(1-\nu_l(x_i))\right]$$

$$\eta_{ij} \leftarrow \frac{\eta_{ij}}{\sum_{l=1}^{K}\eta_{il}},\ \tau_{ij} \leftarrow |1-y_i x_i^T w_j|^{-1},\ \beta_{ij} \leftarrow \frac{1}{2\psi_{ij}}\tanh(0.5\psi_{ij})$$

M Step:

$$A_j \leftarrow diag(\frac{\eta_{nj}}{\tau_{nj}})_{n=1}^{N},\ \Omega_j \leftarrow diag(\beta_{nj}\sum_{l=j}^{K}\eta_{nl})_{n=1}^{N}$$

$$\kappa_j^T \leftarrow [\eta_{nj} - 0.5\sum_{l=k}^{K}\eta_{nl}]_{n=1}^{N}$$

$$w_j \leftarrow (X^T A_j X + \lambda I)^{-1}(\sum_{i=1}^{N}\eta_{ij}\frac{\tau_{ij}+1}{\tau_{ij}}y_i x_i),\ v_j \leftarrow (X^T\Omega_j X + \rho I)^{-1}X^T\kappa_j$$

Hierarchical Mixture-of-Experts

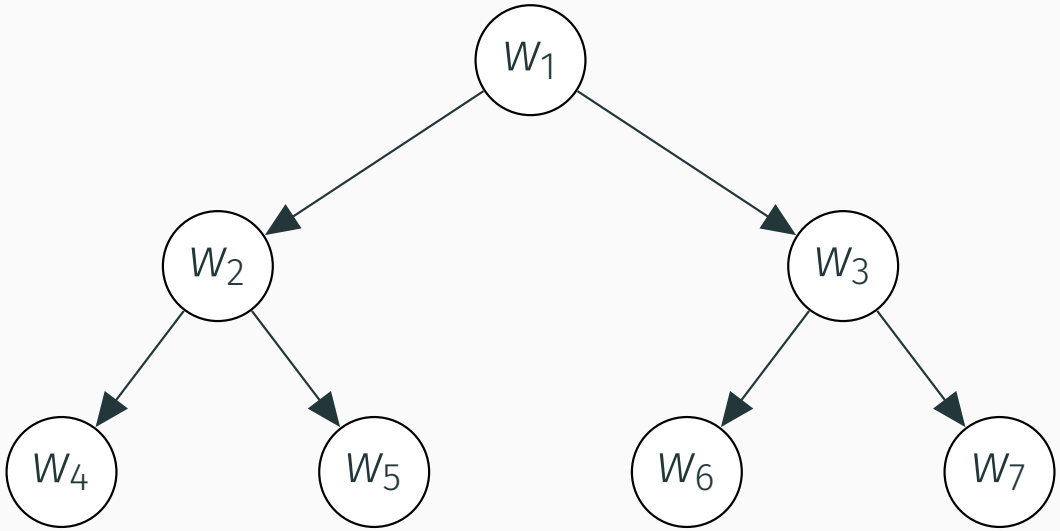

Figure 1: An instance of HMoE

Hierarchical Mixture-of-Experts

Predefine the structure of HMoE to be a full binary tree.

- *Expert Network*: The leaf nodes are the experts.
- *Gating Network*: The internal nodes decide the input-expert assignment. Essentially, the weight of the path to the leaf is the probability of that expert being chosen for the given input.

Choose all nodes, internal and leaf, to be Bayesian SVMs. Parametrized by $W = \{w_i\}_{i=1}^{7}$, where w_i denotes the weight vector associated with that node. Assume labels $y_i \in \{0, 1\}$ for all inputs x_i.

Hierarchical Mixture-of-Experts

For all internal nodes, we need to augment two Bayesian SVM latent variables $\gamma_{ij}^{(0)}, \gamma_{ij}^{(1)}$ for every example x_i. For all leaf nodes, we only need γ_{ij}. $z_{ij} = 1$ denotes the event that $j^{(th)}$ node was visited for x_i. The following recursive definitions hold:

$$p(z_{ij} = 1, \gamma | x_i, W) = p(j\%2, \gamma_{i\frac{j}{2}}^{j\%2} | w_2, x_i) p(z_{i\frac{j}{2}} = 1, \gamma_i | x_i, W)$$

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

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

Parameter Updates for HMoE

Updates

For all $w \in \{w_1, \ldots w_7\}$

$$w = (X^T A X + \lambda I)^{-1} (\sum_{i=1}^{N} b_i x_i)$$

$$A = \mathrm{diag}(a_1, a_2, \ldots a_n)$$

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

$$\tau_i^{(0)} = |1 + x_i^T w|, \ \tau_i^{(1)} = |1 - x_i^T w|$$

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

η_i denotes the sum of posterior expert assignment probabilities for experts arrived at by the following the left link from w. Similarly, η_i' is defined for experts from the right link.

Forward-Backward Algorithm

The recursive defintions and previously defined updates for parameters allow us to create a forward-backward pass algorithm for learning HMoE parameters.

- *Forward Pass*: This step is equivalent to E Step. In the forward pass, the posterior probabilities for each leaf node is computed, which is just the product of likelihoods of for each “link” on the path.
- *Backward Pass*: This is equivalent to M Step. The parameters are updated according to the above equations. The updates depend upon the posterior probabilities of experts arising from the left links and right links, which can be computed online in the backward pass.

Results

Some Results

Table 1: RBF-SVM: Support Vector Machine with RBF Kernel, **SS-ζ** (T=5): SS-softplus regression with Kmax = 20 and T = 5 [6], **GG**: Mixture of Bayesian SVM Experts with Generative Gating, **PG**: Mixture of Bayesian SVM experts with Polya-Gamma augmented Softmax gating Networks, **LSBP**: Logistic Stick Breaking Prior Network, **HMoE**: Hierarchical Mixture-of-Experts. *Mean error and Standard Deviation* is reported.

Data	RBF-SVM	SS-τ	GG*	LSBP*	PG*	HMoE*
b	10.85 (0.57)	11.89 (0.61)	**10.60 (0.41)**	11.53 (0.91)	25.45 (5.3)	15.59 (3.89)
w	10.73 (0.86)	11.69 (0.69)	9.41 (1.5)	19.1 (1.6)	**8.93 (0.31)**	12.19 (1.05)
i	2.84 (0.52)	**2.73 (0.53)**	3.1 (0.45)	3.65 (0.94)	10.43 (3.06)	6.49 (1.43)
bc	28.44 (4.52)	28.83 (3.40)	**21.04 (1.91)**	21.56 (3.01)	23.12 (4.07)	24.91 (4.55)

Table 2: b $\equiv$ banana, w $\equiv$ waveform, i $\equiv$ image, bc $\equiv$ breast cancer

Questions?

References i


C. Cortes and V. Vapnik.
Support-vector networks.
Mach. Learn., 20(3):273–297, Sept. 1995.


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


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

References ii

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

J. G. Scott and L. Sun.
Expectation-maximization for logistic regression.
arXiv preprint arXiv:1306.0040, 2013.

M. Zhou.
Softplus regressions and convex polytopes.
arXiv preprint arXiv:1608.06383, 2016.