Sparse Mixture of Experts

Benjamin Eysenbach
beysenba@cs.cmu.edu

William Guss
wguss@cs.cmu.edu

Chirag Gupta
chiragg@cs.cmu.edu

1 Introduction

Many problems of classification, prediction, and control operate on heterogeneous sources of data. For example, patients with a history of smoking are at risk for different sets of diseases than patients who have never smoked. Similarly, driving a car on a dry fall day is almost entirely distinct from driving in freezing sleet. For these sorts of data, smartly partitioning the data and using separate models for each partition has a number of potential benefits. First, if the data can be well approximated by a handful of simple models, then learning separate, simple models will likely be faster than learning a single complex model. Second, the partitions of a mixture model can be inspected, giving insight into how the model operates. Finally, we expect our model will generalize better on data drawn from a hierarchical generative model.

We focus on *sparse* mixture of experts (MoE) models. We propose a series of algorithms for optimizing this model in both the fixed data and inference setting. By introducing efficient algorithms for learning sparse mixture of experts models, our work allows users to exploit structure in data for efficiency and accuracy gains.

2 Background

Formally, we consider the setting where we have a fixed, heterogeneous dataset $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ and a set functions $\{f_j : 1 \leq j \leq C\}$ for performing the task at hand (classification, prediction, control). Our goal is to assign each datum $\mathbf{x}_i$ to exactly one function ("learner") f_j to maximize some objective over the entire dataset. To formalize this as an optimization problem, we introduce auxiliary variables $\alpha_{i,j}$ to indicate whether datum $\mathbf{x}_i$ belongs to function f_j:

$$\alpha_{i,j} \triangleq \mathbb{1}(\mathbf{x}_i \text{ assigned to } f_j)$$

We then introduce a convex loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ that measures the deviation from the function prediction and the true labels. With this notation in place, we cast our optimization problem as follows:

$$\begin{aligned} \min_{\alpha,\theta} & \sum_{i=1}^{N}\sum_{j=1}^{C} \alpha_{i,j}\mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i) && (1) \\ \text{s.t. } & \sum_{j=1}^{C} \alpha_{i,j} = 1 \qquad \forall\, i \in \{1\cdots N\} \\ & \alpha_{i,j} \in \{0,1\} \qquad \forall\, i \in \{1\cdots N\},\ j \in \{1\cdots C\} \end{aligned}$$

We note a few important properties of this problem. First, the problem is not convex. The objective is invariant to permutations of the cluster assignments, but interpolation between permutations will give an infeasible objective. Second, if we fix α, then our problem becomes convex. This observation motivates decoupling optimization into one step for obtaining α, and then a second step to perform convex optimization of θ.

NP-hardness. Without further assumptions, this problem is NP-hard even if $\mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i)$

Preprint. Work in progress.

is convex in θ. We show this by reduction to $k-$means which is known to be hard. Consider $\mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i) = \|x_i - \theta_j\|_2^2$ (it is independent of $\mathbf{y}_i$). This clearly reduces the problem to $k-$means.
Hence, the only hope is to solve this problem under assumptions. The eventual goal would be to identify conditions under which we can solve the problem. In this report, we empirically explore this model for some simple toy datasets and consider heuristics to obtain results in this setting.

3 Fixed Data Setting

In this section, we introduce two broad classes of algorithms for optimizing our problem: alternating minimization and direct optimization. One would notice that in this section we do not talk about inference, that is, predicting α for a new test point. Hence the term 'fixed-data'. This is in keeping with a focus on optimization that is more relevant for the course.

3.1 Alternating Minimization

One approach to solving our optimization problem is to alternate between optimizing the learner assignments α and optimizing the learner parameters θ. Algorithm 1 outlines the general approach. Noted that in Line 6, we compute $\mathcal{L}_{ij}$ for each pair (i,j), not just for ones where $\alpha_{ij} = 1$. We propose a number of methods for using these losses to assign data to experts. One naïve approach is to simply assign each datum to the learner under which it has lowest error. This approach is unstable: if no data are assigned to some learner, it is unclear how to update this learner in the next iteration. In classification settings, an additional problem is that each learner must obtain both positive and negative examples. Empirically, we find that both issues occur frequently enough to merit considerable discussion. We implement Algorithm 1 using four choices for the assignment function:

Algorithm 1 Alternating Minimization

```
1: randomly initialize α
2: while not converged do
3:     for j = 1 ··· C do
4:         Fit f_j to samples {(x_i, y_i) | α_{i,j} = 1}
5:         for i = 1 ··· N do
6:             Compute error L_{i,j} ← L(f_j(x_i; θ_j))
7:         end for
8:     end for
9:     α ← AssignFn(L, {f_1 ··· f_C})
10: end while
```

Argmin: Each sample is assigned to the model under which it has lowest error:

$$\alpha_i = \underset{j\in\{1\cdots C\}}{\arg\min}\, \mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i)$$

Softmax: This randomized approach samples the cluster assignment with probability inversely proportional to the error:

$$\alpha_i \sim \text{Categorical}(p_i) \quad \text{where} \quad p_{ij} = \frac{e^{-\mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i)/\tau}}{\sum_{l=1}^{C} e^{-\mathcal{L}(f_l(\mathbf{x}_i;\theta_l),\mathbf{y}_i)/\tau}}$$

Linear Sum Assignment (LAP): This approach solves an ancillary matching problem to assign samples to clusters. The solution to the linear sum assignment is an assignment to clusters where each cluster contains exactly N/C samples. We can solve this inner optimization problem efficiently using the Munkres algorithm [22].

Linear Program (LP): We solve a linear program in an inner loop of our optimization algorithm that minimizes the loss subject to constraints on the sizes of each cluster:

$$\min_{\alpha} \sum_{i=1}^{N}\sum_{j=1}^{C} \alpha_{i,j}\mathcal{L}_{i,j}$$

$$\text{s.t.} \sum_{j=1}^{C} \alpha_{i,j} = 1 \quad \forall\, i \in \{1\cdots N\} \quad \text{and} \quad \alpha_{i,j} \in [0,1] \quad \forall\, i \in \{1\cdots N\},\ j \in \{1\cdots C\}$$

We then appeal to the Birkhoff-von Neumann Theorem [5] to argue that the solution will have integer solutions. We solve this linear program using cvxopt [9].

The assignment methods are all related. LAP approach is a special case of the LP approach, where the lower and upper bound both equal N/C. In our experiments, we know that the correct cluster size is N/C, motivating the use of the LAP approach. In practice, when we do not know the correct size of each cluster, the LP approach gives us flexibility to allow different cluster sizes, while still enforcing constraints on the minimum and maximum sizes of each cluster. Similarly, the argmin is a generalization of the softmax approach, when the temperature goes to 0. Note that as the softmax temperature goes to infinity, we end up assigning samples to clusters uniformly at random.

3.2 Direct Optimization

An alternative paradigm for solution our sparse, mixture of experts model is to directly optimize a smooth, non-convex relaxation. Note that the original problem is non-smooth because of the integer constraints on α. One particular instantiation of this paradigm is to relax the integer constraints on α and optimize the following, weighted objective:

$$\min_{\theta,z} \quad \sum_{i=1}^{N}\sum_{j=1}^{C} \alpha_{i,j}(\tau)\mathcal{L}(f_j(\mathbf{x}_i;\theta_j),\mathbf{y}_i)$$

$$s.t. \quad \alpha_{i,j}(\tau) \triangleq \frac{\exp(-z_{i,j}/\tau)}{\sum_{k=1}^{C}\exp(-z_{i,k}/\tau)}$$

We can then use either first or second order methods to directly optimize this objective. Note that $\tau \in \mathbb{R}^+$ is a tuning parameter. We can iteratively solve this problem for decreasing values of τ, resulting in a sparse solution $\alpha(0)$ at convergence.

3.3 Experiments

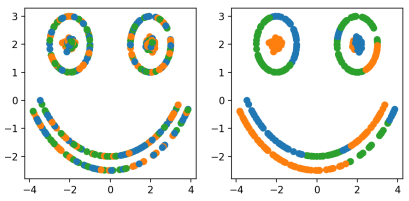
To study these two classes of algorithm, we applied them to two tasks: regression of (procedurally generated) piece-wise linear functions and classification of the "smile" dataset. We use linear regression experts for regression and RBF-SVM experts for classification. The main objective of these experiments is to illustrate how the naïve approach to alternating minimization results in empty clusters, whereas our proposed methods help avoid this.

Figure 1 summarizes our data and results. For regression (top row), we repeat each experiment k times, generating a new dataset for each trial. We set $k = 1000$ for the (fast) "argmin" and "softmax" approaches, and $k = 10$ for the other approaches. For classification, we use the same "smile" dataset for each trial, but randomize the initial assignment of data to clusters. The first column visualizes the dataset. Column 2 shows the training error, which is the L2 error for regression and the fraction of incorrect labels for classification. The third column shows the fraction of trials that did not return degenerate solutions (see discussion below). The last column reports the average wall clock time for each trial.

For the regression task, we observe that the baseline "argmin" approach is not *stable* (i.e., it results in empty clusters) on 20% of datasets. In contrast, all other approaches avoid empty clusters. As expected, the randomized "softmax" approach accrues high error because of its stochasticity. Note that the "LP" and "LAP" methods take substantially longer because they require solving an inner optimization problem.

For the classification task, we observe that the baseline "argmin" approach is unstable for about 25% of random seeds. Even though the "LP" and "LAP" approaches guarantee that each cluster contains at least L examples, they can still fail to converge if a they assign only positive (or only negative) examples to one cluster: it is unclear how to fit a classifier if you lack data from some class. This observation also explains why the "LP" and "LAP" approaches have high error. The softmax approach achieves a remarkably small training error. Because the RBF-SVM learners are quite expressive, they manage to classify the data, even if the cluster assignments are random.

Are clusters semantically meaningful? For regression, we recover one expert for each linear segment. For classification, we illustrate the cluster assignments in the to



Cluster assignments at random initialization (left) and convergence (right).

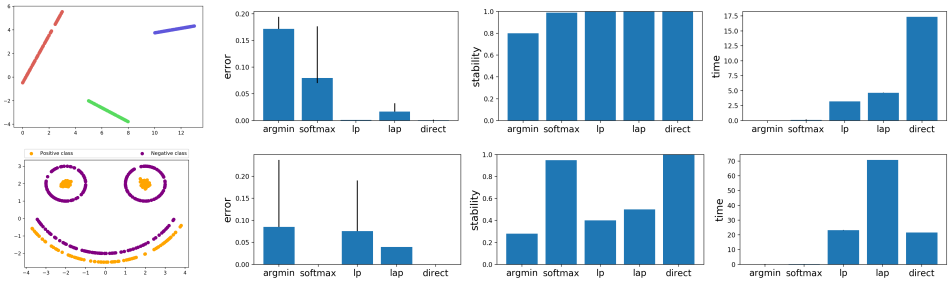


Figure 1: **Fixed Data Experiments**: We apply the alternating minimization algorithm to regression tasks (top) and a classification task (bottom). "Armin" refers to the baseline approach that simply assigns each datum to the cluster under which it has lowest loss. "Softmax" is a randomized variant, while "LP" and "LAP" solve inner optimization problems to avoid empty clusters. "Direct" is the direct optimization approach from § 3.2.

the right. The cluster assignments are random upon initialization, but do not converge to the expected clusters (one for each eye and one for the mouth), despite our model obtaining perfect accuracy. This observation contradicts our hypothesis that sparse MoE models would necessarily learn interpretable clusters.

4 Inference Setting

Up to this point we have examined how we can find a sparse mixture of experts for *fixed* data. However, these models cannot be readily applied to unseen data. We find that α_{ij} can directly memorize the labels. For example suppose that the number of clusters is equal to or larger than the number of classes. Then, a trivial solution that gets 100% training accuracy is that cluster j always returns class j, and α_{ij} is simply equal to 1 if point $\mathbf{x}_i$ belongs to class j, and 0 otherwise.
Thus, as is, we cannot hope to generalize to any test set. In this section, we address the problems associated with the inference setting by studying parametric and semi-supervised approaches to assigning new data to experts.

4.1 Parametric Approach

Our first approach assumes that α is a parametric function of the data. We formulate this approach as solving the following optimization problem:

$$\begin{aligned}
\min_{\theta} \quad & \sum_{i=1}^{N}\sum_{j=1}^{C} \alpha_\theta(\mathbf{x}_i)_j \mathcal{L}_{i,j} \\
\text{s.t.} \quad & \sum_{j=1}^{C} \alpha_\theta(\mathbf{x}_i)_j = 1 \quad \forall\, i \in [N] \\
& \alpha_\theta(\mathbf{x}_i)_j \in [0,1] \quad \forall\, i \in [N],\ j \in [C] \\
& L \le \sum_{i=1}^{N} \alpha_\theta(\mathbf{x}_i)_j \le U \quad \forall\, j \in [C]
\end{aligned}$$

We solve this problem via dual gradient ascent. We choose a functional form for α (in particular, a softmax of a linear function of the data) to structurally impose the constraint that each α be between 0 and 1, and that they sum to 1 for each datum $\mathbf{x}_j$. We then form the augmented Lagrangian by moving

the cluster size constraints into the objective:

$$\begin{aligned}\mathcal{L}(\theta, u, v) \triangleq & \sum_{i=1}^{N}\sum_{j=1}^{C}\alpha_\theta(\mathbf{x}_i)_j\mathcal{L}_{i,j} \\ & + \sum_{j=1}^{C} u_j\left(L - \sum_{i=1}^{N}\alpha_\theta(\mathbf{x}_i)_j\right) + \sum_{j=1}^{C} v_j\left(\sum_{i=1}^{N}\alpha_\theta(\mathbf{x}_i)_j - U\right) \\ & + \frac{\rho}{2}\sum_{j=1}^{C}\left[L - \sum_{i=1}^{N}\alpha_\theta(\mathbf{x}_i)_j\right]_+^2 + \sum_{j=1}^{C}\left[\sum_{i=1}^{N}\alpha_\theta(\mathbf{x}_i)_j - U\right]_+^2\end{aligned}$$

We perform dual gradient ascent on this objective by repeating the following steps, where the learning rate λ is a hyperparameter:

$$\begin{aligned}\theta^{(t)} &\leftarrow \arg\min_\theta \mathcal{L}(\theta, u^{(t-1)}, v^{(t-1)}) \\ u^{(t)} &\leftarrow u^{(t-1)} - \lambda\nabla_u\mathcal{L}(\theta^{(t)}, u, v^{(t-1)}) \\ v^{(t)} &\leftarrow v^{(t-1)} - \lambda\nabla_v\mathcal{L}(\theta^{(t)}, u^{(t-1)}, v)\end{aligned}$$

4.1.1 Extension to Nonconvex Function Approximators

We further consider extending the inference algorithm above to use nonconvex function approximators for the cluster assignments α and the learners f_j. Our algorithm is nearly the same as above. However, for computational efficiency, we do not solve the optimization over θ exactly, but rather take a single gradient step. This results in the following optimization algorithm:

$$\begin{aligned}\theta^{(t)} &\leftarrow \theta^{(t-1)} - \lambda\nabla_\theta\mathcal{L}(\theta, u^{(t-1)}, v^{(t-1)}) \\ u^{(t)} &\leftarrow u^{(t-1)} - \lambda\nabla_u\mathcal{L}(\theta^{(t)}, u, v^{(t-1)}) \\ v^{(t)} &\leftarrow v^{(t-1)} - \lambda\nabla_v\mathcal{L}(\theta^{(t)}, u^{(t-1)}, v)\end{aligned}$$

Taking inspiration from [25], we implement this algorithm as a standard Optimizer in Tensorflow. This tool readily allows non-expert users to solve constrained optimization problems.

4.2 Semi-Supervised Approaches

Alternately, we can consider semi-supervised approaches. These approaches rest on the following assumption: *locality in data-space implies locality in model-space.* In other words, heterogeneity data can be broken into non-heterogeneous clusters. This leads immediately to two algorithms. Both use sparse cluster assignments, which avoids the computational cost of training and evaluating each model across the whole dataset.

k-Means Assignment A natural approach is to assign models according to a k-means Voronoi partition of the data. Specifically, we first apply k-means++ [3] using the standard, objective-based clustering loss:

$$L(\mu) = \sum_{i=1}^{n}\min_{j=1,\ldots,k}\|\mathbf{x}_i - \mu_j\|^2.$$

We then train each model f_j on its particular cluster C_j:

$$\min_{\theta\in\mathbb{R}^{n\times C}}\sum_{k=1}^{n}\sum_{i\in C_j}\mathcal{L}(f_j(\mathbf{x}_i;\theta_j), \mathbf{y}_i)$$

We assign new examples to experts by choosing the model whose underlying cluster mean is closest. In all experiments in this paper, we use the Logistic Regression loss for $mathcalL$.

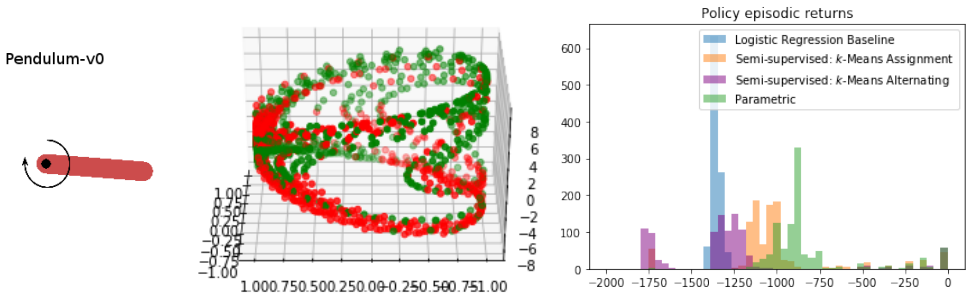


Figure 2: **Pendulum control task**: (left) illustration of the "Pendulum" control task; (center) expert demonstrations on the pendulum task, colored by the action taken at each step; (right) episodic returns of our algorithms ("semi-supervised-*" and "parametric") and a baseline ("logistic regression").

k-Means Initialized Alternating Minimization We first run the k-Means Assignment described above. After that, we proceed with the alternating minimization algorithm (Algorithm 1). For inference, we first set new cluster centers as the cluster means of the final learned assignment. Then, test points are assigned to the cluster with the closest center. We feel this could be improved in future work for the following reason: logistic regression tends to have very high confidence/probability on 'far-away' points. Consequently, during the reassignment phase of the alternating minimization algorithm, 'far-away' points are likelier to get assigned to incorrect clusters. By virtue of being 'far-away', these points affect the cluster means a lot leading to bad inference. One way to remedy this which we could not try was to make the means robust to outliers (for example use k-medians instead).

4.3 Experiments

4.3.1 Experimental Setup

In order to study the proposed inference setting algorithms, we apply them to three different tasks: the toy Smiley dataset, the standard MNIST handwritten digits dataset [19], and a behavioral cloning task on the OpenAI Gym[6] environment Pendulum-v0. These tasks exhibit varying degrees of heterogeneity in data space. For example, the Smiley dataset (Fig. 1, top left) can be explained by three simple models (namely, one for the mouth and two for the eyes). Likewise, the Pendulum behavioral cloning dataset (Fig. 2) can be explained by a collection of linear sub-policies.

For the Parametric approach, we used 10 experts, each of which was a linear model with a logistic loss. We trained to convergence, using Adam [18] with a step size of 1e-4 on each dataset. For k-means and k-means alternating, we again use 10 experts; each expert is a logistic regression model trained on each individual cluster using LBFGS. For k-means alternating, reassignment was performed using the argmin approach for a total of 20 iterations.

We evaluated the algorithms by computing the test accuracy on both Smiley and MNIST and the training accuracy on Pendulum. Figure 1 illustrates the results of our algorithms on these three tasks. To further evaluate the policies learned on the Pendulum task, we compute the reward achieved by each algorithm, shown in Figure 2.

4.4 Results

Table 1: **Inference Setting Experiments**. We show the test performance for four approaches to the inference setting. For the toy task and digit classification on MNIST, we report model accuracy on a held-out test set. For behavioral cloning on Pendulum, we report the training accuracy achieved.

	Smiley (test)	MNIST (test)	Pendulum (train)
Parametric	50.00%	91.08%	**76.11%**
Logistic Regression (baseline)	50.00%	92.32%	50.27%
k-Means Mixture	84.00%	**96.07%**	74.10%
k-Means Alternating Minimization	**86.16%**	94.28%	75.56%

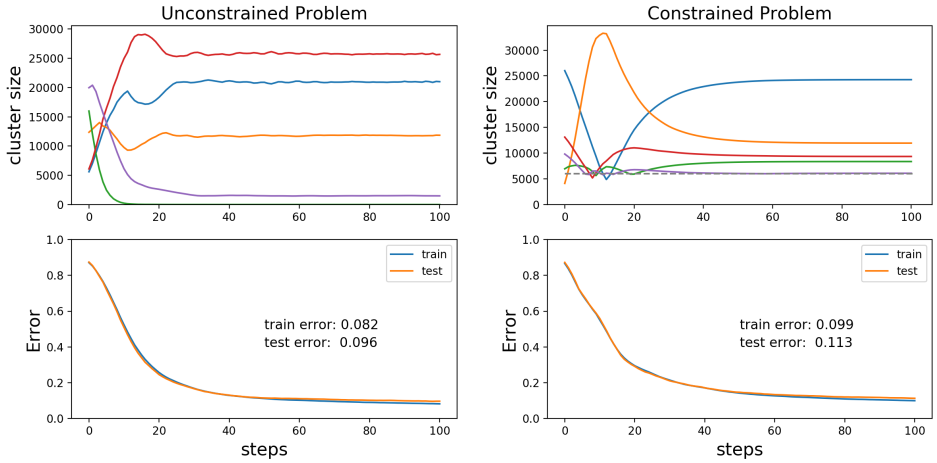


Figure 3: **Adding constraints**: We evaluate the parametric approach with and without constraints on the minimum cluster size. We use the MNIST dataset and learn a mixture of $C = 10$ experts. We observe that adding the constraint avoids empty clusters, but does not improve generalization.

We observe that, on all tasks, our algorithms perform well on unseen data, demonstrating that our approaches allow mixture of expert models to be applied in the inference setting.

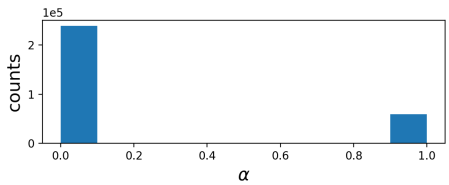
On the Smiley dataset, the "k-means alternating" approach performed best, while the parametric and baseline approaches which performed at chance. We hypothesize that the semi-supervised methods performed well because they exploit spatial locality of different clusters in the Smiley dataset. We suspect that the parametric model achieves low test accuracy because it does not enforce that nearby points be assigned to similar clusters.

For the MNIST task, all of the proposed inference mode models perform at a testing accuracy above 90%. The k-means approach performs the best, likely because the clusters found by k-means are well aligned with the class labels [26].

Finally, on the Pendulum task, each of the proposed algorithms outperforms the logistic regression baseline. While the logistic regression is able to capture the overall global policy (predicting counter-clockwise when z is negative and clockwise when z is positive), it fails to capture local structure of the dataset. As a proxy for test performance, we compute the episodic return for each of the learned policies. Figure 2 (right) shows that the parametric policy achieves the largest and most stable returns. Surprisingly, "k-means alternating" receives lower returns than the parametric approach, despite obtaining higher train accuracy We hypothesize that the noise in the expert data (e.g., incorrect actions) causes the "k-Means alternating" approach to overfit to the training set, obtaining low reward despite high training accuracy.

One benefit of the parametric approach is that we can easily add constraints. We additionally imposed the constraint that each cluster contain at least 10% of the datum. Figure 3 illustrates how, without the constraint, our parametric approach results in 2 of the 5 clusters becoming nearly empty. In contrast, adding the constraint ensured that all clusters remained non-empty. Surprisingly, imposing the constraint did not improve generalization. We suspect that this is an artifact of using a dataset that can be fairly well classified by a single expert. We hypothesize that the constrained solution would generalize better on more complex tasks, though we leave this to future work.

Finally, one concern with the parametric approach is that it does not explicitly enforce sparsity constraints. In fact, we find that the parametric approach implicitly learns a sparse solution. The figure to the right illustrates the predicted values of α, which are nearly all 0 or 1. If we had found that the parametric approach yielded non-sparse solutions, we could apply any of the assignment methods from Section 3.1 to obtain a sparse solution.



Parametric approach implicitly yields integer values for the cluster assignments, α.

5 Prior Work

The optimization problem we proposed is deeply related to the Mixture of Experts (MoE) problem. MoE attempts to learn a probability for a data-point to belong to a specific learner via a 'gating network' (e.g., [14, 8, 24]). The initial mixed-integer optimization problem (Eq. 1) from which we derive our convex relaxation is deeply related to mixture models, hierarchical control, and convex clustering. For example, [8] considered a mixture of SVMs and proposed to separate the optimization of the objective from the gating network in alternating steps. The gating network was trained to minimize the objective given the specific trained learners. After initial work by [14] in defining MoE models, substantial effort has been placed in developing sufficiently powerful models for both the gating networks and the underlying experts f_i. For example, [23, 10] use non-parametric stochastic processes, [8] consider parametric models such as SVMs, while [24, 13]. use neural networks. Likewise, several authors [23, 2] have proposed various constraints (graphical, hierarchical, sequential) on the configuration of experts.

6 Conclusion

This project explored the sparse mixture of experts model. Mixture models are broadly applicable to applications with heterogeneous data. We proposed a number of algorithms for *stably* performing optimization on this model. We explored how naïve alternating minimization can result in degenerate solutions, where some experts receive no training examples. We present a series of algorithms that prevent this degeneracy, by careful initialization, adding stochasticity, or solving an inner optimization problem. We further extend our algorithms to the inference setting, where we hope to obtain predictions on previously unseen data. Experiments illustrate that our models generally perform well in this setting. In closing, we hypothesize that the sparse mixture of experts model, solved via algorithms proposed above, will improve generalization, computational efficiency and interpretability, though we leave this to future work.

References

[1] A. Agarwal, H. Luo, B. Neyshabur, and R. E. Schapire. Corralling a band of bandit algorithms. *arXiv preprint arXiv:1612.06246*, 2016.

[2] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pages 7120–7129, 2017.

[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] D. Bahler and L. Navarro. Methods for combining heterogeneous sets of classifiers. In *17th Natl. Conf. on Artificial Intelligence (AAAI), Workshop on New Research Problems for Machine Learning*, 2000.

[5] G. Birkhoff. Three observations on linear algebra. *Univ. Nac. Tacuman, Rev. Ser. A*, 5:147–151, 1946.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[8] R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of svms for very large scale problems. In *Advances in Neural Information Processing Systems*, pages 633–640, 2002.

[9] J. Dahl and L. Vandenberghe. CVXOPT: A python package for convex optimization, 2008.

[10] M. P. Deisenroth and J. W. Ng. Distributed gaussian processes. *arXiv preprint arXiv:1502.02843*, 2015.

[11] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[12] S. Gross, A. Szlam, et al. Hard mixtures of experts for large scale weakly supervised vision. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5085–5093. IEEE, 2017.

[13] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[14] 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.

[15] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

[16] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed <today>].

[17] G. Karakoulas and R. Salakhutdinov. Semi-supervised mixture-of-experts classification. In *null*, pages 138–145. IEEE, 2004.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[20] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[21] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[22] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[23] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, pages 881–888, 2002.

[24] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[25] M. Wytock, S. Diamond, F. Heide, and S. Boyd. A new architecture for optimization modeling frameworks. In *Python for High-Performance and Scientific Computing (PyHPC), Workshop on*, pages 36–44. IEEE, 2016.

[26] R. Zhang and A. I. Rudnicky. A large scale clustering scheme for kernel k-means. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 289–292. IEEE, 2002.