Multi-Task Learning via Task Multi-Clustering

Anonymous Authors[1]

Abstract

Multi-task learning has the potential to facilitate learning of shared representations between tasks, leading to better task performance. Some sets of tasks are related, and can share many features that are useful latent representations for these tasks. Other sets of tasks are less related, possibly sharing some features, but also competing on the representational resources of shared parameters. We propose to discover how to share parameters between related tasks and separate parameters between conflicting tasks, by learning a multi-clustering of the tasks. We present a mixture-of-experts model, where each cluster is an expert that extracts a feature vector from the input, and each task belongs to a set of clusters whose experts it can mix. In experiments on the CIFAR-100 MTL domain, multi-clustering outperforms a model that mixes all experts in accuracy and computation time. The results suggest that the performance of our method is robust to regularization that increases the model's sparsity when sufficient data is available, and can benefit from sparser models as data becomes scarcer.

1. Introduction

Multi-task learning (MTL) is the process of jointly learning to perform multiple tasks (Ruder, 2017). When different tasks are related to each other, we might expect joint learning to provide benefits such as achieving better performance at the tasks, requiring less data, and improving generalization and robustness. The reasons why MTL might lead to such benefits go beyond the mere availability of more training data or supervision signal: MTL can facilitate structured representation, i.e., have some parameters shared between related tasks. In this work, we focus on tasks that are related in the sense that they share featurizations of the input — i.e., latent representations — from which each task's output can be computed. A standard hypothesis is that learning models with shared parameters has the potential to lead to better performance.

However, when models for unrelated tasks are forced to share parameters, they compete on the same representational resources, which may hurt their performance. Ideally, we would learn models jointly for tasks that share useful features, and separately for tasks that do not; but this principle meets two obstacles. First, a pair of tasks may share some useful features but not others, so that a simple binary decision — whether to learn two tasks jointly or separately — may be too coarse. Second, it is often unclear how one could obtain prior information about the degree to which different tasks are related or competing in the structural sense.

In this work, we propose to overcome these obstacles by discovering a multi-clustering of the tasks, i.e., an assignment of each task to multiple clusters of tasks to which it is partially related. Each cluster corresponds to an "expert" that extracts a set of features from the input. A model for the task is then obtained by a mixture of the experts to which the task is clustered. Our gradient-based method jointly learns the experts, the multi-clustering, and the mixtures.

We present experimental results on the CIFAR-100 MTL domain suggesting that our method has both computational and performance benefits over existing methods with the same number of experts. The results further provide evidence that regularizing for model sparsity, which decreases the number of experts used by tasks, can improve performance when data is somewhat scarce. Experimenting with a range of regularization coefficients when data is more abundant indicates that performance is largely insensitive to small changes in that hyper-parameter, and that our method of selecting experts during evaluation outperforms simpler baselines.

In summary, this paper contributes: (1) a method for multi-task learning via task multi-clustering using discrete gating; (2) an evaluation of the computational and performance benefits of this method; and (3) evidence that the method's performance is robust to variations in the regularization hyper-parameter in the high-data regime, and can benefit from model sparsity in the low-data regime.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Adaptive & Multitask Learning workshop at the International Conference on Machine Learning (AMTL @ ICML). Do not distribute.

2. Related Work

Extensive research has been done in multi-task learning, seeking to improve task performance (Misra et al., 2016; Pinto & Gupta, 2017) and reduce the requirement of computational (Shazeer et al., 2017) and memory resources (Mallya & Lazebnik, 2018). As a result, many creative models and algorithms have been proposed, including ones that cluster tasks in a top-down fashion (Lu et al., 2017), learn relationships between tasks using matrix priors (Long et al., 2017), and more (Ruder, 2017).

Our model uses the mixture-of-experts (MoE) model for multi-task learning. Two related models are the cross-stitch network (Misra et al., 2016), where outputs from all experts are mixed together by a task-specific linear combination, and the routing network (Rosenbaum et al., 2017), where one expert per gating layer is selected, depending on the input and the task. Routing and cross-stitch networks can be viewed as two extremes of a spectrum, with the former making gating decisions that are hard (yes / no) and sparse (exactly one expert), and the latter performing soft and dense "gating". Our method enjoys the best of both approaches, by learning how many experts to select for each task, as well as task-specific linear combinations that mix the selected experts. This paper only studies networks with a single gating layer, but our method can be straightforwardly extended to multiple gating layers.

A similar approach is taken in Shazeer et al. (2017) in the single-task setting, where the gating layer selects a predetermined number of experts. In contrast, we learn a task-specific number of experts in the multi-task setting, which requires a different mechanism for expert selection (Section 3). During training time, our gating is a variant of dropout (Srivastava et al., 2014) in which entire experts are dropped out rather than single neurons, and the dropout probabilities are learned rather than fixed.

Another MoE model for the single-task setting is DeepMoE (Wang et al., 2018), which uses ReLU activation to encourage sparsity in input-dependent gating weights. In contrast, we make discrete gating choices using a probabilistic model that is conditioned only on the task. After training, we fix the set of experts that contribute to each task, facilitating efficient model deployment, and improving the model's interpretability (Section 4.3).

3. Task Multi-Clustering

A task is a tuple $\langle \mathcal{X}, \mathcal{Y}, P, \mathcal{L} \rangle$, with P a distribution over $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ a loss function. We say that a model $m : \mathcal{X} \to \mathcal{Y}$ achieves good performance if it has low expected loss $\mathbb{E}_{(x,y)\sim P}[\mathcal{L}(m(x), y)]$. In multi-task learning (MTL), we assume some distribution Q over a set of tasks $\mathcal{T}$, and a dataset of tuples (t, x, y), such that $t \sim Q$ and $(x, y) \mid t \sim P_t$. In our experiments, Q is the uniform distribution over N tasks.

We propose a mixture-of-experts (MoE) model for MTL in which the tasks in $\mathcal{T}$ are clustered into M experts. The binary gating variables $b_i(t) \in \{0, 1\}$ indicate whether task t is clustered into expert i. Intuitively, we would like a subset of the tasks $\mathcal{T}_i = \{t | b_i(t)=1\}$ to be clustered into expert i if these tasks, and only these tasks, share useful features $f_i(x)$ of the input x. The expert should learn to extract such features.

Given the expert featurizations $f_i(x)$ and the multi-clustering $b_i(t)$, we compute for each task t a task-specific representation $z(t, x)$ as a mixture of the experts into which it is clustered

$$z(t, x) = \sum_i b_i(t) w_i(t) f_i(x), \tag{1}$$

where $w_i(t)$ are the mixture weights. We learn a *task head* g mapping from the latent representation $z(t, x)$ to the output

$$m(t, x) = g(t, z(t, x)).$$

Our training objective is to minimize the expected loss between the predicted output and the true output

$$\min_{b,w,f,g} \mathbb{E}_{t\sim Q} \, \mathbb{E}_{(x,y)\sim P_t}[\mathcal{L}_t(m(t, x), y)]. \tag{2}$$

We note that, during both training and evaluation time, only the experts of the mixture (1) for which $b_i(t) = 1$ need to be called. Sparsity of the multi-clustering can therefore yield significant computational efficiency.

To perform gradient-based optimization of the objective (2), we can choose differentiable parametrizations of all w_i's, f_i's, and g as neural networks. Unfortunately, the multi-clustering indicators b_i are inherently discrete. To relax this modeling assumption, we introduce the probability $p_i(t)$ that expert i will be used for task t, and have $b_i(t) \sim \text{Bernoulli}(p_i(t))$. This makes $z(t, x; b)$ and $m(t, x; b)$ random variables, requiring special care in the optimization.

Because the discrete gating variables are not differentiable, in order to optimize the gating distribution, we can either relax it into a reparametrizable distribution using the Gumbel–softmax trick (Jang et al., 2016), or use the score-function trick to sample from the same gating distribution that we optimize. Sampling from a relaxed gating distribution generates gating values $b_i(t) \in (0, 1)$, and does not enjoy the computational benefits of sparsity during training time. We therefore choose the score-function trick

$$\nabla \, \mathbb{E}_{b\sim p(t)}[\mathcal{L}_t(m(t, x; b), y)] = \mathbb{E}_{b\sim p(t)} \left[\nabla \mathcal{L}_t + \mathcal{L}_t \sum_i \nabla \log p_i(b_i; t) \right], \tag{3}$$

where $p_i(b_i;t)$ is $p_i(t)$ if $b_i{=}1$, else $1{-}p_i(t)$. To estimate the gradient (3) in each optimizer step, we sample a single gating combination b for each element of the mini-batch.

We define the density of each task as the expected number of experts used in its computation, $k_t = \sum_i p_i(t)$, and refer to its complement $M - k_t$ as the sparsity. We desire sparser representations for faster training and evaluation, as long as task performance does not degrade much.

To our optimization objective, we add the expected task density, in our case the mean $\frac{1}{N}\sum_t k_t$, as a regularization term, namely L_1 regularization on p. The coefficient of this term, λ, can be viewed as the relative marginal cost of computing one expert and incurring a unit of loss. By tuning λ, we can control the trade-off between computation and loss.

During evaluation time, given an input x for task t, we wish to keep the same task density k_t as in training time. Moreover, efficient model deployment and computation may require a fixed, deterministic multi-clustering b. We therefore round k_t to the nearest integer, and pick the set of k_t most likely experts

$$\underset{b:\ \sum_i b_i(t)=k_t}{\arg\max} \sum_i b_i(t)p_i(t). \tag{4}$$

Other approaches to choosing the fixed multi-clustering b from the learned distribution p could be rounding the probabilities to 1 or 0, or sampling from them. However, rounding would induce a different expert density at evaluation time than in training time, which may create a covariate shift for the downstream network, and sampling from the gating distribution may not select the most likely experts, thus degrading performance.

As a result of our gating discretization technique (4), the model computation time is $O(k) \ll O(M)$, where regularization induces selection of only k experts of the available M. Mixing all experts would require $O(M)$ computation time.As the diversity of the set of tasks $\mathcal{T}$ increases, we must also increase the total number of experts M, but not the task density k, contributing to the advantage of the multi-clustering method over naive MoE.

4. Experiments

We evaluate our method on the CIFAR-100 MTL dataset, in which each task is to classify an image into one of 5 classes, with cross-entropy loss between the true class and the predicted class distribution. There are $N = 20$ tasks, corresponding to 20 non-overlapping super-classes (fish, flowers, etc.). Each image in the dataset is involved in exactly one task and has exactly one true label for that task.

The deep network architecture of each of the experts $f_i(x)$ is four sequential convolutional layers, each with 32 filters, ReLU activation, and 2×2 max pooling. Each task head $g(t, z)$ is a fully connected layer that takes the mixture of experts $z(t, x)$ and outputs classification logits. Two $N \times M$ matrices represent $w_i(t)$ and the logits of $p_i(t)$, with each element of w initialized by a standard Gaussian, and p initialized uniformly as 0 logits. We optimize the mean task loss (2) using Adam with an initial learning rate of 10^{-3}.

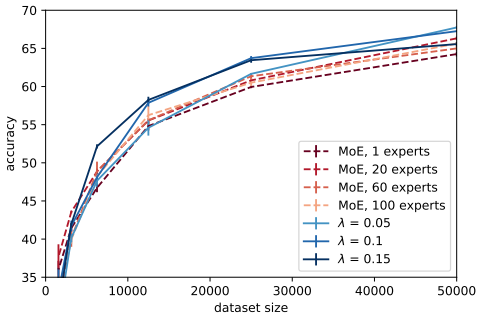

Figure 1. Accuracy of models trained with different amounts of data. Task multi-clustering with sufficiently large sparsity coefficient λ outperforms ungated mixture-of-experts, when sufficient data is available.

We present three experiments: (1) a comparison between different gating techniques in terms of performance, (2) a comparison between different ways to select experts during evaluation; and (3) a qualitative evaluation of the multi-clustering similarity between tasks.

4.1. Gating Evaluation

To compare with cross-stitch networks (Misra et al., 2016), we generalize their method to have more than 2 tasks and 2 experts by representing the mixture weights as an $N \times M$ matrix. This is equivalent to setting $b \equiv 1$ in our model, and compares sparse gating to dense, *ungated* MoE.

The results are summarized in Figure 1. With sufficient data, multi-clustering outperforms ungated MoE significantly. While MoE is largely insensitive to the number of experts in a wide range (20–100), multi-clustering benefits from sparser gating as the amount of data decreases, presumably due to a regularization effect. These results consistently outperform those reported in Rosenbaum et al. (2017) by a large margin: routing networks achieve 60% accuracy, whereas our model with $\lambda = 0.1$ achieves 68.5%.

4.2. Expert Selection during Evaluation

We turn to the question of how task-specific subsets of experts should be selected during evaluation, based on the learned distribution p. We wish to fix the multi-clustering after training to deploy a fixed model for each task with

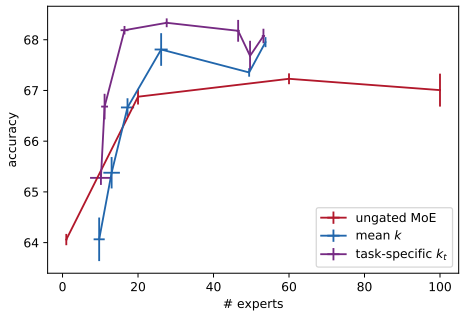

Figure 2. Performance over mean number of experts used during evaluation time on the full dataset of 50K images. Using task-specific selection of k_t, the number of experts per task, outperforms using mean k over all tasks and ungated mixture of all experts.

only the selected experts. This has the potential to make the model more computationally efficient and interpretable.

Prior works predefine the number of experts per task as a hyper-parameter k. In Shazeer et al. (2017), the k experts with the largest (noisy) weights are selected. In contrast, we choose the task-specific k_t to be the expected number of experts selected during training time for task t. We speculate that this choice of k_t reduces the covariate shift in the distribution of $z(t, x)$ experienced by the task head $g(t, z)$, thus improving its performance. Given k_t, we select the maximum-posterior subset of experts for the task, which is the k_t most likely experts.

We compare this method of selecting the final multi-clustering to choosing k to be mean expected number of experts over all tasks. The results, summarized in Figure 2, suggest that choosing task-specific k_t outperforms choosing the mean k. We speculate that the performance degradation is more pronounced in sparser multi-clustering because this increases the risk of having too few features for tasks that need more features than the average.

Figure 2 also compares the performance of ungated MoE as a function of the total number of experts M. Performance is largely insensitive to the number of experts, if there are sufficiently many, but forcing all M experts to be mixed does degrade performance. This suggests that tasks do benefit from our discrete multi-clustering formulation.

4.3. Task Similarity

After adding sparsity regularization with coefficient $\lambda = 0.1$, we find that some tasks have a strong preference for some experts, as depicted in Figure 3. The left maps show the probabilities of the M experts being used for the N tasks, with yellow values being close to 1, after training from 50K data points (top) and 12.5K data points (bottom). The maps on the right display the normalized correlation in gating

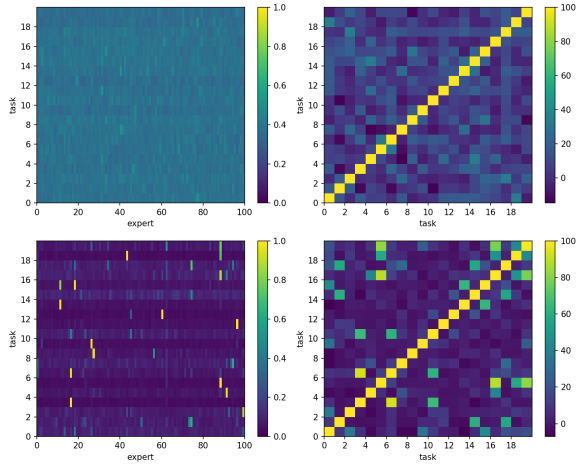

Figure 3. Gating representations for multi-clustering with sparsity regularization coefficient $\lambda = 0.1$, trained on 12.5K (top) and 50K (bottom) data points. Left: probabilities of selecting each expert for each task. Right: normalized correlation between the tasks.

probabilities between pairs of tasks (rows in the left maps). With the larger dataset, there is a clear correlation between tasks, such as task 3 and task 6, which are "food containers" and "household furniture", respectively. In contrast, with the smaller dataset, the correlations between tasks in terms of their gating probabilities are unclear.

5. Conclusion

In multi-task domains, tasks are often partially related, in the sense that they benefit from sharing some but not all of their latent representation with other tasks. We presented a method for discovering a multi-clustering of tasks into feature-extracting experts, such that each task uses a mixture of the experts it selected. Through these experts, the task shares each portion of the parameters in its model with a different cluster of partially related tasks.

The discovered structure of the multi-clustering facilitates more efficient learning, in terms of both task performance and computational efficiency. The model's sparsity allows evaluating only a subset of the experts per task, both during training and evaluation. By fixing the multi-clustering after training, we enable deployment of smaller models per task, which only involve the useful experts. Finally, with sufficient data, the model's performance is robust to significantly increasing the sparsity through regularization.

The single multi-clustering gating layer presented in this work can be extended to models with multiple such layers. This extension may require techniques that reduce the considerable variance of the score-function gradient estimator (3). Our method should further be evaluated on a variety of domains and architectures.

References

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Long, M., Cao, Z., Wang, J., and Philip, S. Y. Learning multiple tasks with multilinear relationship networks. In *Advances in Neural Information Processing Systems*, pp. 1594–1603, 2017.

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5334–5343, 2017.

Mallya, A. and Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3994–4003, 2016.

Pinto, L. and Gupta, A. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

Rosenbaum, C., Klinger, T., and Riemer, M. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

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

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Wang, X., Yu, F., Wang, R., Ma, Y.-A., Mirhoseini, A., Darrell, T., and Gonzalez, J. E. Deep mixture of experts via shallow embedding. *arXiv preprint arXiv:1806.01531*, 2018.