ExpertMatcher: Automating ML Model Selection for Clients using Hidden Representations

Vivek Sharma
MIT
vvsharma@mit.edu

Praneeth Vepakomma
MIT
vepakom@mit.edu

Tristan Swedish
MIT
tswedish@mit.edu

Ken Chang
MIT
kenchang@mit.edu

Jayashree Kalpathy-Cramer
MGH/Harvard Medical School
kalpathy@nmr.mgh.harvard.edu

Ramesh Raskar
MIT
raskar@mit.edu

Abstract

Recently, there has been the development of Split Learning, a framework for distributed computation where model components are split between the client and server (Vepakomma *et al.* , 2018b). As Split Learning scales to include many different model components, there needs to be a method of matching client-side model components with the best server-side model components. A solution to this problem was introduced in the ExpertMatcher (Sharma *et al.* , 2019) framework, which uses autoencoders to match raw data to models. In this work we propose an extension of ExpertMatcher, where matching can be performed without the need to share the client's raw data representation. The technique is applicable to situations where there are local clients and centralized expert ML models, but sharing of raw data is constrained.

1 Introduction

The Expert matching problem as described by Sharma *et al.* (2019) is to assign input data from a client to the most appropriate expert model without any information other than the input data itself. In essence, the goal is to assign input data based on its likelihood of being drawn from the distribution of the expert model training data. We are interested in assigning an expert model to a given sample such that an improved performance is achieved. The expert matching problem is well-studied: (Jacobs *et al.* , 1991; Jacobs, 1995, 1997), starting from the seminal work of Jacobs *et al.* (1991), where the authors trained a versatile blend of experts for speaker vowel recognition and utilized a gating system to determine which of the systems should be utilized for each sample. Other lines of work try to avoid using a gating function (Hinton *et al.* , 2015; Ahmed *et al.* , 2016) by training one oracle model. Aljundi *et al.* (2017) learns a gating function to make expert network assignments. More recently, with the ongoing advances in machine learning and big data, Sharma *et al.* (2019) proposed a landscape for the expert matching problem and shows that by using an Autoencoder (AE), it becomes possible to assign the most relevant expert model(s) given a sample data representation in a client-server setting. However, in their setting the client shares the raw data with the server.

For many reasons raw data cannot always be shared. First, there is oftentimes a need to protect the privacy of the individual from whom the data originated (such as patient health data). Further, data is a valuable resource and many clients prefer not to have their data open to the public. Lastly, if data is shared, there will be additional storage requirements which may be cumbersome and expensive. Motivated by this observation, our approach does not require sharing of raw data directly. Instead, independent autoencoders are trained on both the server and client sides and only the intermediate hidden representation of the data is shared for expert network(s) assignment.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

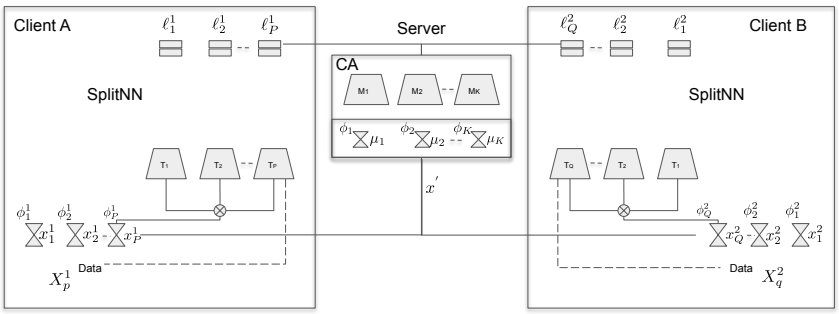

Figure 1: Pipeline of ExpertMatcher with hidden representations at the client and server for automatic model selection from a repository of server models based on their relevance to query data set hosted by the client. The matching is done based on encoded intermediate representations.

We evaluate our proposed method on six datasets, such as objects, text, digits, biological and sensor, namely STl-10 (Coates *et al.* , 2011), MNIST (LeCun *et al.* , 1998), HAR (Anguita *et al.* , 2013), Reuters (Lewis *et al.* , 2004), Non Line of Sight (Tancik *et al.* , 2018) and Diabetic Retinopathy (Graham, 2015). We are inspired from Sharma *et al.* (2019), and we experimentally show that we can achieve similar performance as Sharma *et al.* (2019) to match the task (coarse-Level) assignment and reasonably good performance for class (fine-grained) assignment in a Split Learning distributed computation framework (Vepakomma *et al.* , 2018b, 2019, 2018a).

2 Hidden Representation based ExpertMatcher

The expert matching problem aims to assign a given sample the best model at hand for a given task. Figure 2 shows the landscape of the Expert-Matching problem proposed by Sharma *et al.* (2019). In this work, we consider the no sharing scenario (row fourth in Figure 2), where the client and server do not share the data and model, but just the low-dimensional intermediate representation. Furthermore, within Figure 2 we consider (1) Resolution: coarse and fine level assignment; (2) Fusion: top-1; and (3) Metric: adhoc (i.e. cosine similarity).

	Resolution	Fusion	Metric
	Coarse | Fine	Top-1 | Top-K	Adhoc | Learn
Sharing	Data and weights are shared between client and server		
No Model Sharing	Share intermediate representation of server model		
No Data Sharing	Share processed representation of client data		
No Model & Data Sharing	Both client and server share intermediate representations		

Figure 2: Landscape of ExpertMatcher, Source: Sharma *et al.* (2019). This paper is relevant to the fourth setup where raw data or model is not shared between client and server, but rather some intermediate encoded representation is shared.

Now, we describe our proposed method to solve the ExpertMatcher problem without sharing raw data. In Sharma *et al.* (2019), the raw data is shared with the Server. In our proposed method, the Client shares only a encoded hidden representation for both Coarse-level (CA) and Fine-grained (FA) expert matching. Figure 1 and Figure 3 sketches the proposed method.

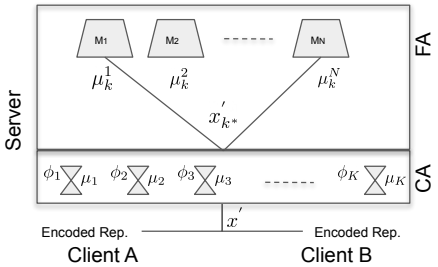

Figure 3: Overview of the coarse level dataset assignment (CA) and fine-level model assignment (FA) of the client's data in a hierarchical fashion. In contrast to Sharma *et al.* (2019), for both CA and FA we use maximum cosine-similarity as a metric for model assignment. Further, the client shares the encoder representation rather than the raw samples (Sharma *et al.* , 2019).

Implementation: We assume that we have K pre-trained expert networks on the centralized server, each of these networks has its corresponding pre-trained autoencoders (AE) ϕ_K trained on a task-specific dataset. Given the dataset on which the AE was trained on, we extract the hidden representations of the whole dataset and compute an average representation of the dataset $\mu_k \in \mathbb{R}^d, k \in \{1, \ldots, K\}$, where d is the feature dimension. Assuming the dataset consist of N object classes, we also compute the average representation of each class in the dataset $\mu_k^n \in \mathbb{R}^d, n \in \{1, \ldots, N\}, k \in \{1, \ldots, K\}$.

计算每一个用户，每一个类的平均特征向量

The clients (Client A and Client B) utilize a similar approach as the server, where the clients train their unique AE's one for each p^{th} and q^{th} datasets, Client A: $p \in \{1, \ldots, P\}$ and Client B: $q \in \{1, \ldots, Q\}$. Lets assume for Client A, the intermediate features extracted from a hidden layer

for a sample X_p^1 is given as $x_p^1 = \phi_p^1(X_p^1)$, similarly for Client B, $x_q^2 = \phi_q^2(X_q^2)$. For brevity, we denote the intermediate representation coming from any client as $x^{'}$.

For coarse assignment (CA) of clients data: hidden representation $x^{'}$, we assign an server AE, $k^* \in \{1, \ldots, K\}$ which has maximum cosine similarity of $x^{'}$ with μ_k.

For fine-grained (FA) of clients data: hidden representation $x^{'}$, we assign an expert network $M_n, n \in \{1, \ldots, N\}$ that has the maximum cosine similarity of $x^{'}_{k^*}$ with μ_k^n.

Finally after the assignment of the given sample to the model, one can easily train a SplitNN type architecture Vepakomma *et al.* (2018b, 2019, 2018a).

In the current setup, a weak level of privacy is preserved as the server does not have access to the raw client's data, but rather a very low dimensional encoded representation.

Note that there is a shortcoming of this approach. If the server has no AE model dedicated to the client data, the wrong assignment of client data takes place because of maximum cosine similarity criteria - this can be resolved by adding an additional model on the server that performs a binary classification: if the client data matches the server data or not.

3 Experiments

We present our evaluation of several different datasets. We first describe the dataset, metric, implementation details, followed by a thorough analysis of the proposed methods and comparison to Sharma *et al.* (2019) for both coarse assignment (CA) and fine-grained assignment (FA).

Datasets and Implementation Details. We demonstrate our method on challenging datasets which cover domains such as objects, text, digits, biological and raw sensors. The datasets are summarized in Table 1. For a fair comparison with Sharma *et al.* (2019), we use the same splits for Server, client A and Client B.

For all the image data types, we resize them to 28×28, and then flattening it to 784 dimensions. While for other datasets, we use 1D adaptive average pooling (`AdaptiveAvgPool1d`) to transform the input data to 784 dimensions. **AE:** All the AEs use a single-layer MLP encoder-decoder ($\mathbb{R}^{784} \to \mathbb{R}^{128} \to \mathbb{R}^{784}$). We use Adam optimizer for model training. We train the AE models for 45 epochs with a learning rate of 10^{-2}, which is then manually decreased by a factor of 10 every 15 epochs.

Note that, in this work, the server shares the pseudorandom number generator seeds with the clients to make the same random initialization for PyTorch model training (e.g. `torch.manual_seed(0)`). In future work, we plan to study the impact of random initialisation when the seed is not shared between server and clients.

Table 1: **Datasets**. "#S" denotes the number of samples, "#C" denotes the true number classes, and LC/SC is the class balance largest class (LC) to smallest class (SC).

	STL-10	MNIST	HAR	REUTERS	NLOS	DB	ALL
Type	Object	Digits	Sensors	Text	Sensor	Biological	
#C	10	10	6	4	3	3	
#S	13k	10k	10299	10k	45096	3540	
Dim.	32px	28px	561	2000	640×480	512px	
LC/SC (%)	10/10	11.35/8.92	19/14	43.12/8.14	33.33/33.33	33.33/33.33	
Server	6500	5000	5151	5000	22548	1770	
Client A	3250	2500	2574	2500	11274	885	22983
Client B	3250	2500	2574	2500	11274	885	22983

Evaluation Metric. For both CA and FA, we use maximum cosine similarity for making a model assignment (CA) / class assignment (FA) and then computing the accuracy between predicted class and target class.

Multiple Clients. In the current work, we consider two clients (Client A and Client B) for evaluation. However, the amount data available to each client is the same. In future work we plan to study the impact of the varying proportion of samples assigned to the clients for model training and assignment. Our idea is easily scalable to many more clients, and is not limited to the two client scenario presented here.

3.1 Coarse-level dataset assignment (CA)

In Table 2, we compare the results for the coarse level dataset assignment with Sharma *et al.* (2019). In case of Sharma *et al.* (2019), the authors use minimum reconstruction error (i.e. MSE) as the criteria for making the sample assignment when the client shares the raw data with the server.

These results suggest that autoencoders trained on non-overlapping datasets at the client or server side perform similarly to Sharma *et al.* (2019) - where the client shares the raw data with the server. Note we are not sharing our raw data - but only hidden representation. Further, we note that autoencoders are effective in learning the underlying representation of the dataset even if the client and server AE's are trained on different datasets. This is a surprising result, as training with non-overlapping data seems to lead to representations that are close via cosine similarity. In this work, we ensure models are initialized with the same seed, which encourages the models to converge to similar representations as long as the training samples are coming from the same underlying distribution. Otherwise, we expect the order of learned filters would not be preserved and cosine similarity would fail. We note that both Client A and Client B are 99% accurately assigned to their corresponding autoencoders.

Note that in the financial services industry, we have a similar situation where we are interested in finding an expert method for handling our specific problem at hand without loss of confidential information. A similar example dataset we have our experimental setup is REUTERS that contains newswire articles, which may be similar to financial analysts reports or other valuable text-based data that may not want to be shared with third parties. We think our approach to finding an expert model without sharing the raw data can easily be deployed to the financial industry where confidential information may not want to be shared in raw form. Sharing encoded representations provides some weak privacy, but also has other benefits such as lower bandwidth requirements leading to decreased latency and more efficient storage of data being shared between client and server.

Table 2: Coarse-level dataset assignment using cosine-similarity as the assignment metric for computing accuracy (%). Note that Sharma *et al.* (2019) use MSE loss as the assignment metric.

	MNIST	STL-10	HAR	REUTERS	NLOS	DB	Average
Client A (Sharma *et al.* , 2019)	100.0	100.0	100.0	99.64	99.92	96.49	99.34
Client A (**ours**)	99.36	99.93	99.80	96.36	99.59	100.0	99.30
Client B (Sharma *et al.* , 2019)	100.0	100.0	100.0	99.56	99.89	95.36	99.13
Client B (**ours**)	99.80	99.96	99.80	96.40	98.28	100.0	98.72

3.2 Fine-grained class assignment (FA)

In Table 3, we compare the results for the fine-grained class assignment with Sharma *et al.* (2019). It is interesting to observe that the autoencoder trained on the client-side and the server-side seems to perform well even if they are trained independently with different samples although coming from the same underlying distribution - this shows that autoencoders are effective at learning the class identity representation. A drop in performance for our method is expected, as the client's AE model has access to much smaller set of examples than the server.

4 Conclusion

In this work, we propose a novel ExpertMatcher based model selection where the raw data of remote clients are not shared with the central server. In contrast to the previous method, the server and client both train a model independently and the intermediate hidden representations are shared between them for either coarse or fine-grained matching respectively. We believe our approach to finding an expert model via hidden representations provides a path for a new paradigm of distributed machine learning architectures where sharing of raw data is not feasible.

Table 3: Fine-grained class assignment using cosine-similarity as the assignment metric for computing accuracy (%).

Dataset	#C	Client A	Client B
MNIST (Sharma *et al.* , 2019)	10	84.36	83.40
MNIST (**ours**)	10	71.4	71.64
NLOS (Sharma *et al.* , 2019)	3	71.78	71.26
NLOS (**ours**)	3	59.40	52.40
DB (Sharma *et al.* , 2019)	3	41.47	44.41
DB (**ours**)	3	44.30	53.33

Acknowledgements V. Sharma would like to thank Karlsruhe House of Young Scientists (KHYS) for funding his research stay at MIT.

References

Ahmed, Karim, Baig, Mohammad Haris, & Torresani, Lorenzo. 2016. Network of experts for large-scale image categorization. *Pages 516–532 of: European Conference on Computer Vision*. Springer.

Aljundi, Rahaf, Chakravarty, Punarjay, & Tuytelaars, Tinne. 2017. Expert gate: Lifelong learning with a network of experts. *Pages 3366–3375 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Anguita, Davide, Ghio, Alessandro, Oneto, Luca, Parra, Xavier, & Reyes-Ortiz, Jorge Luis. 2013. A public domain dataset for human activity recognition using smartphones. *In: Esann*.

Coates, Adam, Ng, Andrew, & Lee, Honglak. 2011. An analysis of single-layer networks in unsupervised feature learning. *Pages 215–223 of: Proceedings of the fourteenth international conference on artificial intelligence and statistics*.

Graham, Ben. 2015. Kaggle diabetic retinopathy detection competition report. *University of Warwick*.

Hinton, Geoffrey, Vinyals, Oriol, & Dean, Jeff. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jacobs, Robert A. 1995. Methods for combining experts' probability assessments. *Neural computation*, **7**(5), 867–888.

Jacobs, Robert A. 1997. Bias/variance analyses of mixtures-of-experts architectures. *Neural computation*, **9**(2), 369–383.

Jacobs, Robert A, Jordan, Michael I, Nowlan, Steven J, Hinton, Geoffrey E, *et al.* . 1991. Adaptive mixtures of local experts. *Neural computation*, **3**(1), 79–87.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick, *et al.* . 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Lewis, David D, Yang, Yiming, Rose, Tony G, & Li, Fan. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, **5**(Apr), 361–397.

Sharma, Vivek, Vepakomma, Praneeth, Swedish, Tristan, Chang, Ken, Kalpathy-Cramer, Jayashree, & Raskar, Ramesh. 2019. ExpertMatcher: Automating ML Model Selection for Users in Resource Constrained Countries. *In: NeurIPS workshop on Machine Learning for the Developing World (ML4D)*.

Tancik, Matthew, Satat, Guy, & Raskar, Ramesh. 2018. Flash photography for data-driven hidden scene recovery. *arXiv preprint arXiv:1810.11710*.

Vepakomma, Praneeth, Swedish, Tristan, Raskar, Ramesh, Gupta, Otkrist, & Dubey, Abhimanyu. 2018a. No Peek: A Survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*.

Vepakomma, Praneeth, Gupta, Otkrist, Swedish, Tristan, & Raskar, Ramesh. 2018b. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Vepakomma, Praneeth, Gupta, Otkrist, Dubey, Abhimanyu, & Raskar, Ramesh. 2019. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*.