Few-Shot Learning with Meta Metric Learners

Yu Cheng[†], **Mo Yu**[†], **Xiaoxiao Guo**[†], **Bowen Zhou**[‡]
[†]AI Foundations Lab, IBM Research [‡]JD.COM

Abstract

Existing few-shot learning approaches are based on either meta-learning or metric-learning, which would suffer if the tasks have varying numbers of classes and/or the tasks diverge significantly. We propose **meta metric learning** to deal with the limitations of the existing few-shot learning approaches. Our meta metric learning approach consists of two components, task-specific learners that exploit metric learning to handle flexible numbers of classes across tasks, and a meta-learner that learns and specifies the metrics in the task-specific learners. We test our approach in the new realistic few-shot setting with more diverse tasks and flexible class numbers and obtains superior performance.

1 Introduction

Few-shot learning [10, 8, 7] was proposed to learn classifiers for new classes with only a few training examples per class. The few-shot learning idea has recently been combined with the deep learning techniques in two main lines of researches: (1) learning metric/similarity from multiple few-shot learning tasks with deep networks (metric learning)[6, 14]; and (2) learning a meta-model on multiple few-shot learning tasks, which could be then used to predict model parameters given a new few-shot learning task (meta learning) [4, 13, 11].

Both metric-learning and meta-learning based few-shot learning have achieved successes in the so-called "k-shot, N-way" scenario, in which each few-shot learning task has the same N number of classes and each class has k training instances. However, such simplified "N-way" setting is far from the realistic scenario in many real-world few-shot learning applications. Existing *meta-learning* approaches build on the "N-way" simplification to let the meta-learner predict same-shaped parameter matrices of homogeneously-structured task-specific classifiers (e.g. the weights of task-specific neural networks). On the other hand, the *metric-learning* approaches allow different number of classes per task, but they usually require all the tasks coming from a single domain to ensure that a uniform metric works for all the tasks. However, because of the variety of tasks, the optimal metric will also vary across tasks. The learned task-invariant metric would fail if the tasks diverge.

We propose **meta metric learning** to deal with the limitations of existing few-shot learning approaches. Our meta metric learning approach consists of two components, task-specific learners that exploit metric learning to handle flexible numbers of classes across tasks, and a meta learner that learns and specifies the metrics in the task-specific learners. The meta learner that operates across tasks uses an LSTM-based architecture to discover good parameters of gradient decent in task-specific metric learners. The task-specific metric learners exploit the state of the art metric learning few-shot learning approach, Matching Networks [14], to parameterize the task metrics using the weight prediction from the meta-learner. Our meta metric learner model is able to handle unbalanced classes in meta-train and meta-test sets as the usage of Matching Network as well as to generate task-specific metrics leveraging the weight prediction of the meta-learner given task instances. Our meta metric learning approach combines the merits of both meta-learning and metric-learning approaches to address realistic few-shot learning problems where the tasks could have various numbers of classes and come from different domains.

2 Meta Metric Learner for Few-Shot Learning

The meta metric learning training algorithm is shown in Algorithm 1. Following the notation in [13], the data consists of three parts, $\mathscr{D}_{meta-train}$, $\mathscr{D}_{meta-valid}$ and $\mathscr{D}_{meta-test}$, each $\mathscr{D}$ is a set of few-shot learning tasks. Generally, in real applications, the number of classes in $\mathscr{D}_{meta-train}$ is different from that in $\mathscr{D}_{meta-test}$. Our proposed model, meta-metric learner, consists of two components: (1) the base learner, matching network [14], $M(\cdot;\theta)$, which is trained as a task-specific classifier on every few-shot task sampled from $\mathscr{D}_{meta-train}$, $\mathscr{D}_{meta-valid}$ or $\mathscr{D}_{meta-test}$. The matching networks can tackle class-unbalanced few-shot learning problems; (2) an LSTM-based meta-learner $R(\cdot;\Theta)$ [13], which guides the training of base metric-learner in order to achieve task-specific metric given a few-shot learning task.

Algorithm 1 Meta-Training Algorithm for the Meta Metric Learner

```
Require: Meta-train set $\mathscr{D}_{meta-train}$
 1: Initialize meta-Learner $R$ with parameters randomly initialized $\Theta_0$
 2: for $d = 1, n$ do
 3:     $D^d_{train}$, $D^d_{test} \leftarrow$ random sampled dataset from $\mathscr{D}_{meta-train}$
 4:     Initialize a matching network $M(\cdot;\theta_0)$, with $\theta_0 \leftarrow c_0$
 5:     Divide $D^d_{train}$ to a support set $D^d_S = \{(\hat{x}_t, \hat{y}_t)\}$ and a standard training set $D^d_T = \{(x_t, y_t)^T_{t=1}\}$
 6:     for $t = 1, T$ do
 7:         $x_t, y_t \leftarrow$ random batch sampled from $D^d_T$
 8:         $\mathcal{L}_t \leftarrow \mathcal{L}(M(x_t, D^d_S; \theta_{t-1}), y_t)$
 9:         $c_t \leftarrow R([\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t]); \Theta_{d-1}), \theta_t \leftarrow c_t$
10:     end for
11:     $\mathcal{L}_{test} \leftarrow 0$
12:     for $x, y \in D^d_{test}$ do
13:         $\mathcal{L}_{test} \leftarrow \mathcal{L}_{test} + \mathcal{L}(M(x, S; \theta_T), y)$
14:     end for
15:     Updating $\Theta_d$ using $\nabla_{\Theta_{d-1}}\mathcal{L}_{test}$
16: end for
17: Return $\Theta_n$
```

Meta-Learner Training on $\mathscr{D}_{\mathbf{Meta-Train}}$ At time step t, the meta metric learner $R(\cdot;\Theta)$, which is an LSTM with parameter Θ in this work, receives its input as the stochastic gradient of the base learner on the t-th training instance, concatenated with the loss made by the base learner at t-th step; and its cell state vector c_t corresponds to the base learner's parameters θ_t updated at the t-th step. Therefore, after the meta-learner passed a few-shot learning task with T training instances in total, we have a base learner $M(\cdot;c_T)$ guided by the meta-learner.

The meta-learner's parameters Θ are trained on $\mathscr{D}_{meta-train}$ to optimize the test performance of the base learners $M(\cdot;c_T)$ given the training data of the sampled few-shot tasks. Specifically, we iteratively sample a task d with two data sets from $\mathscr{D}_{meta-train}$ (line 2): D^d_{train} that would be used to train the task-specific base learner $M(\cdot;\theta)$ (line 4-11, see the next paragraph for details) and D^d_{test} that would be used to evaluate the performance of the learned $M(\cdot;\theta_T)$ (line 12-15). Finally, we use the loss of θ_T evaluated on D^d_{test}, $\mathcal{L}_test$, to update the parameter of the meta learner $R(\cdot;\Theta)$ via gradient descent (line 16).

Base-Learner Training (for both Meta-Training and Meta-Testing) Each few-shot learning task from $\mathscr{D}_{meta-train}$ or $\mathscr{D}_{meta-test}$ contains a training set D_{train}, which can be used to train the base learners. With a meta-learner $R(\cdot;\Theta)$ and D_{train}, we first divide D_{train} into two subsets (line 5): a supporting set D_S and a standard training set D_T. In every iteration, we sample training instance (x, y) from D_T. We compare x with the instances in D_S with the metric learned by $M(\cdot;\theta)$, and select the label $y' = M(x, S; \theta)$ that has the class instance $x' \in S$ closest to x (line 8). During meta-training, the meta learner takes the loss on the prediction $\mathcal{L}(y', y)$ and the gradient of the loss as inputs to update the parameters of the base learner (line 9). Similar training process during meta-testing is shown in Algorithm 2 (line 5-10).

Dealing with the One-Shot Difficulty with Auxiliary Data When using the meta-learner to update the parameters of base learner on a few-shot training set D_{train} from $\mathscr{D}_{meta-test}$, we need to divide D_{train} to a support set D_S and a standard training set D_T, in order to compute the loss and its gradient. This means that each few-shot training set D_{train} must contain more than one instance for each class label, i.e. the method could only directly handle k-shot problems with $k \geq 2$.

Algorithm 2 Meta-Testing Algorithm for the Meta Metric Learner

```
Require: $k$-shot task $(D_{train}, D_{test}) \in \mathscr{D}_{meta-test}$, auxiliary set $\mathscr{D}_{aux}$ (optional), meta metric learner $R(\cdot;\Theta)$
 1: Initialize a matching network $M(\cdot;\theta_0)$, initialize $\theta \leftarrow c_0$
 2: for $(D'_{train}, D'_{test})$ randomly sampled from $\{(D_{train}, D_{test})\} \cup \mathscr{D}_{aux}$ do
 3:     if $D'_{train}$ is a one-shot task then continue
 4:     Divide $D^d_{train}$ to a support set $D^d_S = \{(\hat{x}_t, \hat{y}_t)\}$ and a standard training set $D^d_T = \{(x_t, y_t)^T_{t=1}\}$
 5:     $\theta_0 \leftarrow \theta$
 6:     for $t = 1, T$ do
 7:         $x_t, y_t \leftarrow$ random batch sampled from $D^d_T$
 8:         $\mathcal{L}_t \leftarrow \mathcal{L}(M(x_t, D^d_S; \theta_{t-1}), y_t)$
 9:         $c_t \leftarrow R([\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t]); \Theta), \theta_t \leftarrow c_t$
10:     end for
11:     $\theta \leftarrow \theta_T$
12: end for
13: return $\theta$
```

In order to handle one-shot learning problems in our model, we propose to borrow instances from other tasks to update the parameters of the base learner. Specifically, we construct a set of auxiliary tasks (see examples in Sec. 3), $\mathscr{D}_{aux}$, to augment the original D_{train}. This makes the meta-testing essentially a multi-task learning process, as shown in line 2, Algorithm 2.

3 Experiments and Results

Datasets. 1) **Omniglot**: a public few-shot learning benchmark. The data comes with a standard split of 30 training alphabets with 964 classes and 20 evaluation alphabets with 659 classes. Each of these was hand drawn by 20 different people. The large number of classes (characters) with 20 data per class. 2) **Sentence Classification Service (SCS)**: The data is from an on-line service which trains and serves text classification models for different clients for their business purposes. The evaluation tasks contain data from 10 clients, where the clients all provide travel guide/assistant services. The number of classes per clients range from 10 to 28. Note that for both benchmarks the meta-training and meta-testing data could come from different domains (alphabets or clients).[1]

We created few-shot learning tasks from both the benchmarks: we split the data with 1) 50%, 20%, and 30% classes (**5 vs. 3 split**) and 2) 30%, 20%, and 50% classes (**3 vs. 5 split**) for meta-training, meta-validation and meta-testing. As a result, the meta-learners or the metric models are trained and tested with tasks containing different numbers of labels. The previous k-shot,N-way meta-learner could be still applied to case 1), by setting dummy labels during meta-testing thus make the meta-training/testing work on the same number of classes again. However the case 2) is challenging for previous meta-learners since the number of classes in meta-train is smaller than meta-test.

Baseline Models. The first baseline is our own implementation of Matching Network with the fully-conditional embedding (FCE) [14]. The Second baseline is Meta-Learner LSTM. We implemented our own version according to [13].

CNN architectures. We exploit two different CNN architecture used in all the methods: 1) for text, we used a simple yet powerful CNN [5] as the embedding function, which consists of a convolution layer and a max-pooling operation over the entire sentence for each feature map. The model uses multiple filters with varying window sizes $h(h = 3, 4, 5)$. Word embedding are initialized with 100-dimensional Glove embeddings trained on 6B corpus from [12]; 2) for image, the 2D CNN architecture in [14] is used, which consists of a stack of modules, a 3×3 convolution with 64 filters followed by batch normalization, a ReLu non-linearity and 2×2 max-pooling.

Hyper-parameters. There are several hyper-parameters required for meta-learner, matching network and CNNs. All of them are tuned on the meta-validation set.

Standard Few-Shot Learning Results (Meta-Testing with Single Tasks) First we demonstrate the effectiveness of our meta metric learner in the standard few-shot learning settings on Sentence Classification Service and Omniglot datasets. Since no auxiliary set $\mathscr{D}_{aux}$ is available in this setting,

[1]An interesting extension of such multi-domain setting is to automatically detect relevant domains from huge number of arbitrary domains [2, 15] and we leave this to future work.

Table 1: Average classification accuracies on SCS and Omniglot in the heterogeneous setting.

Model	SCS				Ominiglot			
	5 vs. 3 split		**3 vs. 5 split**		**5 vs. 3 split**		**3 vs. 5 split**	
	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot
Matching Network	59.57%	68.91%	48.74%	57.31%	96.50%	97.39%	94.14%	95.26%
Meta-learner LSTM	60.15%	69.06%	-	-	96.54%	97.45%	-	-
Meta Metric-learner	**61.27%**	**69.58%**	**50.56%**	**59.02%**	**97.38%**	**98.47%**	**95.69%**	**96.24%**

we need to perform $k * 2$-shot learning ($k = 1, 2$). For each class, we equally split its samples into the support set and the standard training set. We evaluate our models on both the *5 vs. 3* and the *3 vs. 5* settings. We use all classes in $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$, which is different from k-shot,N-way setting. For both splits, the validation set is used to adjust the hyper-parameters. To have a fair comparison, all the baselines are trained with $k * 2$ samples per class according to their own recipes.

Table 2: Average accuracy on the 1-shot SCS tasks with the usage of auxiliary data.

Model	Additional Data	5 vs. 3 split, 1-shot
Matching Network	Y	53.16%
Meta-learner LSTM	N	56.98%
Meta Metric-learner	Y	**57.92%**

For SCS, all 10 tasks are used in the evaluation and the results are shown in Table 1. All the results are measured after 10 runs. In all for settings, our model outperforms the two baselines (note that meta-learner LSTM does not work for the 3 vs. 5 split), showing that (1) our model could better handle the varying numbers of labels case compared to meta-learner LSTM; and (2) by learning task-specific metric, our model could outperform the standard matching network.

On Omniglot set, we randomly choose 20 tasks from the whole tasks for evaluation. Same setting and data split are used as in SCS. The results are reported after 15 runs and described in Table 1. Similar trend is observed: the performance of meta metric-learners are better than others.

One-Shot Learning with Auxiliary Task Data We show the results of our meta-metric learner when there is auxiliary set $\mathscr{D}_{aux}$ available. According to Algorithm 2, the base learner is trained by the meta-learner with data from multiple tasks (including the target task data from $\mathscr{D}_{meta-test}$ and the auxiliary task data $\mathscr{D}_{aux}$). Therefore the meta-learner could handle the 1-shot learning.

We conduct the experiment on the SCS data, and tested the models in the 5 vs. 3 split setting so that the baseline meta-learner can be compared as well. Since all the 10 clients' data are about travel guide service, we believe that their data belong to similar domains so they could share some common metric space. As a result, for each task from a client, we use the data from the rest 9 clients tasks as the additional auxiliary data during meta-testing. The baseline matching network is trained iteratively with one auxiliary task data each time. Since the tasks from different clients usually have different number of class labels, meta-learner LSTM can not take the additional data and could be only used to train the base learner with the target task data.

The results are shown in Table 2. Meta Metric-learners achieve the best accuracies over all the methods. Note that even with the help of additional sources, the performance of matching network is not better than meta-learner LSTM, but our meta-learning approach significantly improves the accuracy. This observation confirms that our approach does help to learn a better metric compared to standard matching network training.

4 Conclusion

We proposed the meta metric learner for few-shot learning, which uses the meta-learning method to guide the gradient optimization in matching networks. The method takes several advantages of both meta-learning and metric-learning, and is able to handle unbalanced classes as well as to generate task-specific metrics. Our idea is general and can be also employed to concurrent few-shot works [3, 9, 15, 1]. We will explore these directions in the future.

References

[1] Anonymous. A simple neural attentive meta-learner. *International Conference on Learning Representations*, 2018.

[2] Aviad Barzilai and Koby Crammer. Convex multi-task learning by clustering. In *Artificial Intelligence and Statistics*, pages 65–73, 2015.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[4] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[5] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[6] Gregory Koch. *Siamese neural networks for one-shot image recognition*. PhD thesis, University of Toronto, 2015.

[7] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[8] Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[9] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[10] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 464–471. IEEE, 2000.

[11] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *arXiv preprint arXiv:1703.00837*, 2017.

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[13] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6, 2017.

[14] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[15] Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Robust task clustering for deep many-task learning. *arXiv preprint arXiv:1708.07918*, 2017.