2018-2019 学年第二学期 COMP130137.01 《模式识别与机器学习》课程项目 Few-shot Learning

学号：17307130191 , 姓名：罗翔，贡献: 100%，签名：

1 Introduction

1.1 Motivation

Few-shot Learning (FSL) aims to learn classifiers from few examples per class just like humans learn new concepts with little supervision - e.g. a child can generalize the concept of "dolphin" from a glance of the TV show. Recently, deep learning has achieved great success in FSL via generalizing meta-models from a large number of meta-training tasks. Two major ways have been proposed: (1) metric-/similarity-based models, which learn the inter-task similarity (Vinyals et al. [2016]); and (2) optimization-based models, which predict either model parameters or parameter updates after receiving the input of gradients from a FSL task.

Most previous work has focused on the image domains, where all tasks are often sampled from one huge collection of data, such as ImageNet (Vinyals et al. [2016]), thus a common meta-model is often employed. In this project, the dataset comes from Yu et al. [2018]'s work, offering diverse kinds of tasks in Amazon product reviews.

1.2 Problem Definition

Since we focus on single metric-based FSL, the problem can be formulated as meta-training, where a small support set of N labeled examples $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ is given for each task and $x_i \in R^D$ is the D-dimensional feature vector of an example and y_i is the corresponding label. We need to learn a function c_S for each S, i.e. a mapping $S \to c_S(.)$, thus the prediction for an input x_i is $c_{S,\phi}(x_i)$ where ϕ is the parameters of networks.

1.3 Outline

We organized the paper by first introducing the model that we reproduced and comparing between them in Section 2. In Section 3, we will give detailed description of how does our model work and results are shown in Section 4. Related work will be referred in Section **??**.

2 Model and Strategy

Follow Yu et al. [2018]'s step, we also reproduce Vinyals et al. [2016]'s Matching Network and Snell et al. [2017]'s Prototypical Network in single metric-based FSL.

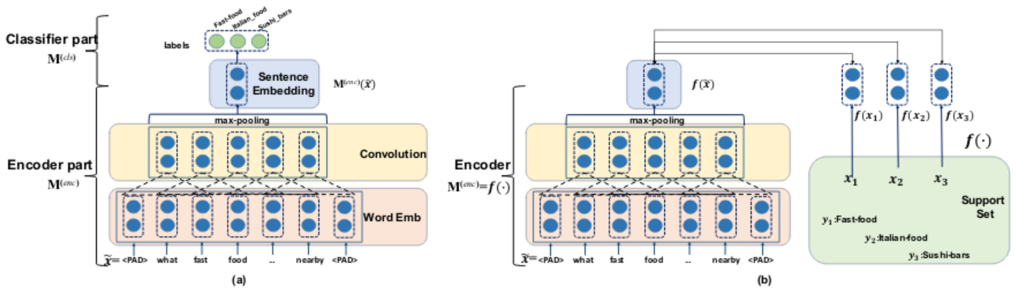


Figure 1: Matching Network

2.1 Matching Networks

Vinyals et al. [2016] first introduce the set-to-set framework, which means the network can produce sensible test labels for unobserved classes without any changes to the network. Their model in its simplest form computes $\widehat{y}$, the predicted output class for a given input unseen example $\widehat{x}$ as follows:

$$\widehat{y} = \sum_{i=1}^{k} a(\widehat{x}, x_i) y_i \tag{1}$$

where x_i, y_i are the samples and labels from the support set $S = \{(x_i, y_i)\}_{i=1}^{k}$. Clearly, eq. 1 essentially points out the output for a new class as a linear combination of the labels in the support set, and we can see eq. 1 subsumes KDE method. Another explanation for eq. 1 is where a acts as an attention mechanism and the y_i act as memories bound to the corresponding x_i, given an input, we "point" to the corresponding example in the support set, retrieving its label. Furthermore, It outperforms other attentional memory mechanisms due to its non-parametric nature, which can adapt to any new support set easily no matter how large the size of the support set is. The Fig. 1 from Yu et al. [2018]'s work vividly demonstrates the procedure.

2.1.1 The Attention Kernel

Equation 1 relies on choosing $a(.,.)$, the attention mechanism, which fully specifies the classifier. Vinyals et al. [2016] use the softmax over the cosine distance c, i.e.

$$a(\widehat{x}, x_i) = \frac{e^{c(f(\widehat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\widehat{x}), g(x_j))}} \tag{2}$$

with embedding functions f and g being appropriate neural networks (potentially with f = g) to embed $\widehat{x}$ and x_i. Snell et al. [2017] points that the the cosine distance may not be optimal measures when not doing one-shot learning, since the cosine similarity is not kinds of Bregman divergences. And it will increase the number of learnable parameters to further improve its performance.

Furthermore, Vinyals et al. [2016] introduced a novelty idea, Full Context Embedding, meaning that the embedding functions f and g are independent of each other to lift the feature space X to achieve maximum accuracy.

2.1.2 Training Strategy

To map a support set to a classification function $S \to c(\widehat{x})$, we modify the set-to-set paradigm augmented with attention, with the resulting mapping being of the form $P_\theta(y|x, S)$, where θ are the parameters of the model(embedding function). To form an 'episode' to compute gradients and update the model, we first sample a task L from the task set, then sample the support set S from L and the rest to be the query set B. The aim is to minimize the error predicting the label in the query set B conditioned on the support set S. The training objective is as follows:

$$\arg\max_{\theta} \; E_{L\in T}[E_{S\in L, B\in(L-S)}[\sum_{(x,y)\in B} \log P_\theta(y|x, S)]] \tag{3}$$

And the algorithm as follows:

Algorithm 1 Framework of Matching Network.

```
Input: The task set $T$, The common classifier, $c_S$.
Output: The loss $L$ for a randomly generated training episode.
1: Select a task for episode: $L \leftarrow Randomsample(T)$
2: Select the support set for task: $S \leftarrow Randomsample(L)$
3: The rest serve as the query set: $B \leftarrow L - S$
4: for (x, y) in B do
5:     Update loss: $L \leftarrow L + \log P_\theta(y|\widehat{x}, S)$
6: end for
```

2.2 Prototypical Networks

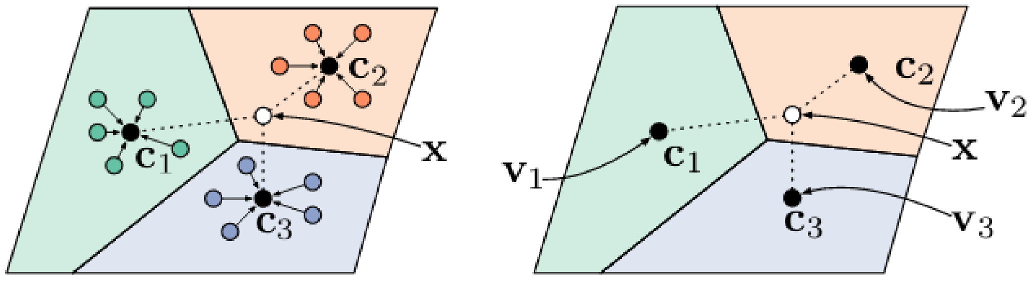


Figure 2: Prototypical Network

Snell et al. [2017] first introduced the concept of prototype, an M-dimensional representation $c_k \in R^M$, of each class through an embedding function $f_\phi : R^D \rightarrow R^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class:

$$c_k = \frac{1}{|S_k|} \sum_{(x,y)\in S_k} f_\phi(x_i) \tag{4}$$

Given a distance function $d : R^M \times R^M \rightarrow [0, +\infty)$, prototypical networks produce a distribution over classes for a query point x based on a softmax over distances to the prototypes in the embedding space:

$$p_\phi(y = k|x) = \frac{e^{-d(f_\phi(x),c_k)}}{\sum_{k'} e^{-d(f_\phi(x),c_{k'})}} \tag{5}$$

Learning proceeds by minimizing the negative log-probability $J(\phi) = -\log p_\phi(y = k|x)$ of true class k via SGD.

2.2.1 Comparison between Protopytical and Matching

Prototype networks use Euclidean distance $||\mathrm{z} - \mathrm{z}'||^2$ as the measures of distance, making the loss is fully differentiable, strictly convex function of the Legendre type. The computation can be viewed in terms of hard clustering on the support set, with one cluster per class and each support point assigned to its corresponding class cluster as the Fig. 2 from Snell et al. [2017]'s work shows. It has been verified for Bregman divergences that the cluster representative achieving minimal distance to its assigned points is the cluster mean, thus yielding optimal cluster representatives given the support set labels when a Bregman divergence is used.

And in the case of one-shot learning, matching networks and prototypical networks are equivalent. Clearly, we can see that prototypical networks are easier to train than matching networks to achieve better performance, since Vinyals et al. [2016] further introduce fully-conditional embedding (FCE)

Table 1: Accuracy of FSL

Model	Accuracy
Matching	0.906
Prototypical	0.846

which imposes an arbitrary ordering on the support set and bi-directional LSTM which dramatically slows the training speed, While prototypical networks can be improved by using some simple design choices.

Algorithm 2 Framework of Prototypical Network.

```
Input: The task set $T$, The common classifier, $c_S$.
Output: The loss $L$ for a randomly generated training episode.
1: Select a task for episode: $L \leftarrow Randomsample(T)$
2: Select the support set for task: $S \leftarrow Randomsample(L)$
3: The rest serve as the query set: $B \leftarrow L - S$
4: $S_k = [(x, y) \in S \;\&\; y = k]$
5: $c_k \leftarrow \frac{1}{|S_k|} \sum_{(x,y)\in S_k} f_\phi(x_i)$
6: for (x, y) in B do
7:     Update loss: $L \leftarrow L + \frac{1}{k|B|}[d(f_\phi(x), c_k) + \log \sum_{k'} e^{-d(f_\phi(x), c_k)}]$
8: end for
```

3 Experiments

In my implementation, there are three python files, __init__.py offers the preprocessing, model.py offers the two models and we train in the few shot.py. We used fastnlp to accelerate the preprocessing, and to make the procedure easier, we sample the support set randomly for each task during preprocessing instead of training. To be honest, due to my laziness, i didn't write codes to sample reasonable support set which may leads to all the point in the support set is 1 or -1, making it impossible to learn correctly. And due to the randomness, we result may vary greatly, and we think the initial test on the development set around 0.5 is a good sample. Again due to my laziness, I only test one task books instead of testing all the four tasks and get the average. But the results of two models all have great performance on development set and test set.

For sake of the speed, we do not incorporate LSTM to the embedding layer, and the CNN embedding layer can finish the training of 200 episodes less than 30 minutes, with good performance. For Matching Network, we use a embedding layer and only one layer of CNN and the classify according to the cosine similarity between the query point and the points in the support set. The difference in Prototypical Network is that we should maintain two support set (due to binary classification) and compute the mean point to represent the distribution of the support set and then compute the Euclidean distance just like KNN method to discriminate which class the query point belongs to.

During the training, we test on the development every 10 episodes and the result is available in 6

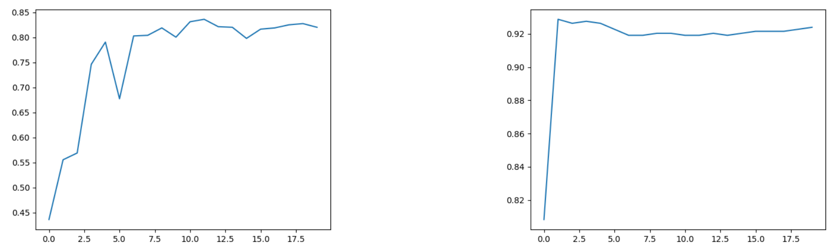
4 Result

Due to the deadline, we did not finish all the jobs and only test the one-way few-shot, but certainly our models is better than those of Yu et al. [2018] with average accuracy for the five-way 65.73 for Matching Network and 68.15 for Prototypical Network.

5 Acknowledgment

Thanks for my classmate Zuobai Zhang's great help in my implementation and through the understanding of the model. Happy to finish this work with only 48 hours(Sophomores be free from finals until 28th June!!!).

6 Appendix



(a) Prototypical Network (b) Matching Network

Figure 3: test on dev set every 10 epochs during training

References

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4080–4090, 2017. URL `http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning`.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638, 2016. URL `http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning`.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1206–1215, 2018. URL `https://aclanthology.info/papers/N18-1109/n18-1109`.