Few-shot learning for Digits Recognition

Yibing Liu

Research School of Computer Science,Australian National University
u6068252@anu.edu.au

Abstract. Digits recognition is a valuable problem and deep learning gets a big breakthrough in this area in the past decade. Yet, traditional supervised learning needs a large number of training data to get a good performance. In this paper, we would like to propose a prototypical network that can train neural network in small dataset and get good performance. The prototypical network will learn a k-means clustering which can classify non-linearly separable patterns. We use the shared weights network to pre-train the non-linear layer in the prototypical network to get a better performance. I choose cascading classifier as the baseline and the prototypical network performs much better than it when less training data are given.

Keywords: few-shot learning · deep learning · digits classification

1 Introduction

Based on research, children can recognize a new world after meet it once.[1] And also, a person can split Segway from other vehicles easily at one encounter.[2] So, it is easy to be concluded that humans have the ability to recognize a new object with one instance or few instances.[3] Although the deep learning technique has been successfully used in supervised pattern recognition and classification and gets state-of-the-art results in many problems, like the crowd recognition, yet, to get a good performance, the supervised learning need a large-scale of manually labeled training data by experts, while in many field this is nearly impossible.[4] With a small dataset, the model will be easily overfitted during training. So, we need to develop methods to solve lack of training data.

In some models, they solve lack of data by labeling data through a algorithm, like labeling topics by LDA algorithm, or collect data labeled by customers, like tweets from twitter, but the problem is that these two methods will introduce noises.[5,6] The noise is incorrectly labeled data. A model might want to analyze sentiment in twitter, so it use emoji in tweets as label of sentiment. In a tweet, its content shows positive sentiment, but it gives a crying emoji in the end. This introduces noise to the model and it will be hard to learn a model which can recognize the sentiment correctly. Therefore, to get a better performance, the noise needs to be modeled.[6].

The dataset I choose to work on is Optical Recognition of Handwritten Digits Data Set and it aims to let machine to classify images of handwritten digits by 44 persons. Digits' images are split into two parts: training data from 30 persons and test data from the other different 14 persons.[7] Digits recognition is a interesting problem in computer vision, and this well-formed dataset makes it easy to investigate the problem. Since each attribute is integer from 1 to 16, we don't need to re-normalize them. The test data is writer-independent which can make the overfitting caused by lack of training data more obvious. If the model has poor ability of generalization, it will perform worse in test dataset written by totally different people.

The prototypical network, proposed by J.Snell et.(2017), is a few-shot learning method attacks the overfitting which is a key problem caused by lack of data and the reason that make the model's ability of generalization poor.[8] The prototypical network assumes that there is a embedding space in which embeddings of digits cluster around a prototype that can represent the class of the digits.[8] The model maps embedding of digits to another embedding space with neural network and then it finds the nearest prototypes to the query.[8] To get a better mapping, we use Shared Weights Network to pre-train the Prototypical Network.[9]

2 Method

The digits dataset can be represented as $D = \{\boldsymbol{y_i}, \boldsymbol{x_i} | 0 \leq i \leq |D|\}$. $\boldsymbol{y_i}$ is label of the digit and $\boldsymbol{x_i}$ is its attributes. $\boldsymbol{x_i} \in \mathbb{Z}^{\mathbb{N}}$ is N-dimensional feature vector used as the input. $y_i \in \{0, ..., 9\}$ is a label representing the class to which the $\boldsymbol{x_i}$ belongs. The model will input the digits' images and then predict to which class it belongs in the output layer. Since the model's aim is to classify images, by minimizing difference between the predicted label and the true label, our model is trained. We use accuracy and macro f1 score to evaluate the performance of our model. Accuracy exactly shows the ratio of queries that are correctly classified, and macro f1 score comprehensively measures the precision and recall.

2.1 Prototypical Network

The prototypical network,proposed by J.Snell (2017), will split the digits training dataset into two parts.[8] D_t is normal training data and D_p is support vectors. We use D_p^k to denotes the set of digits labeled as class $k \in \{0, 1, ..., 9\}$ in D_p and $\boldsymbol{x_i^k} \in D_p^k$ represents a example in it. The model maps original N-dimensional features to M-dimensional space through a non-linear function with learnable parameters θ:

$$f_\theta : \mathbb{Z}^{\mathbb{N}} \leftarrow \mathbb{R}^{\mathbb{M}} \tag{1}$$

We map $\boldsymbol{x_i^k} \in D_p^k$ to M-dimensional space with Function (1) to get support points of class k. Then each prototype of class k, $\boldsymbol{p_k}$, is mean vector of support points:

$$\boldsymbol{p_k} = \sum_{i=1}^{|D_p^k|} f_\theta(\boldsymbol{x_i^k}) \tag{2}$$

When a query $\boldsymbol{x_i} \in D_t$ is given, the squared euclidean distance function *dist* is used to compute the distance between $\boldsymbol{x_i}$ and each prototype $\boldsymbol{p_k}$, and then use softmax to produce the distribution over classes for the query. The query will be classified to the class with maximum probability.

$$p(k|\boldsymbol{x_i}) = \frac{\exp\left(-dist(f_\theta(\boldsymbol{x_i}), f_\theta(\boldsymbol{p_k}))\right)}{\sum_{k'=1}^{9} \exp\left(-dist(f_\theta(\boldsymbol{x_i}), f_\theta(\boldsymbol{p_{k'}}))\right)} \tag{3}$$

To train the model, we need to minimize negative log-probability, $-log(p(k|\boldsymbol{x_i}))$, of true class k. Few-shot learning of prototypical network is shown in figure 1.

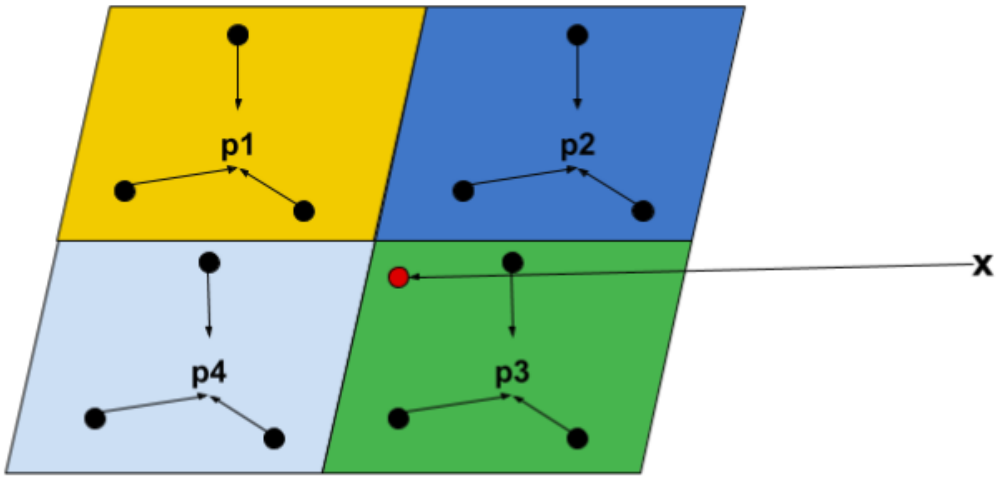

Fig. 1: Prototypical network for few-shot learning. $\boldsymbol{p1}$,$\boldsymbol{p2}$,$\boldsymbol{p3}$,and $\boldsymbol{p4}$ are prototypes of each class calculated by mean of support examples of corresponding class. $\boldsymbol{x}$ is query point which is classified to a cluster whose prototype is the closet to it.

2.2 Shared Weights Network

The shared weights model compresses and decodes the image, which means that the image is the input, compressed by several hidden layers and then symmetrically decoded by other hidden layers.[9] These hidden layers used to compress images share weights with symmetrical layers to decode images. The weight-sharing mechanism is implement by a simulator in which during back-propagation the weight are different and updated by the standard algorithm, and after that we average weights of symmetrical layers in the compressing and decoding process.[9] In figure 1, it is a shared weights network with one hidden layer. $\boldsymbol{W1}$ is weights matrix that maps input to hidden

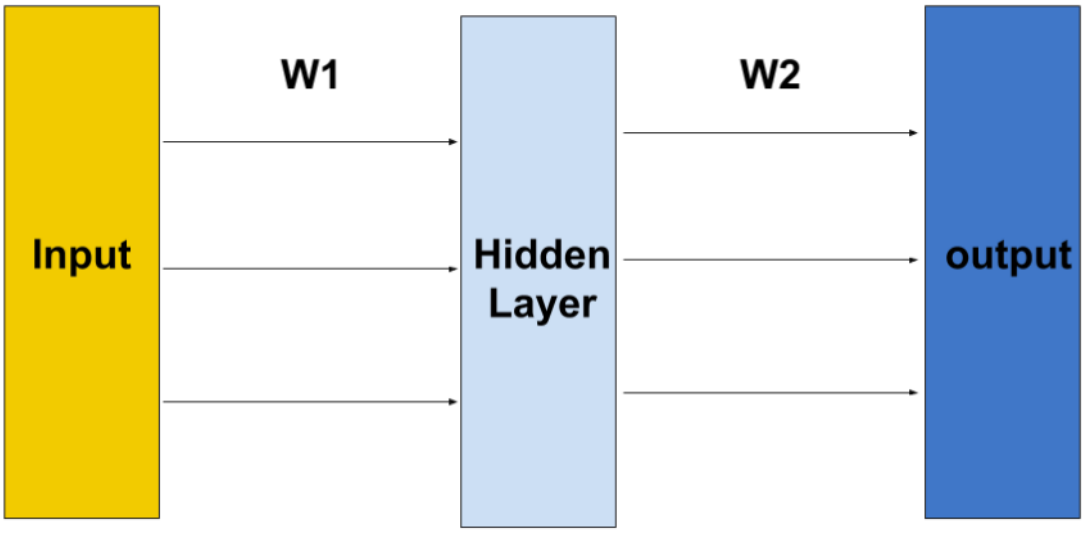

Fig. 2: Shared weights network with one hidden layer

layer and $\boldsymbol{W2}$ maps hidden layer to output layer. Connection between input and hidden layer shares weights with the one between hidden and output layer. So,the value of $\boldsymbol{W1}$ and $\boldsymbol{W2}$ is the same.

We use D to denote the collection of images used to train the shared weights network. $\boldsymbol{x} \in D$ is a image, and it is compressed by a function $f(\boldsymbol{x}) = \boldsymbol{x} * \boldsymbol{W_1^T} + \boldsymbol{b_1}$ and decoded by another function $g(\boldsymbol{h}) = \boldsymbol{h} * \boldsymbol{W_2^T} + \boldsymbol{b_2}$. After the compressed image is decoded, we minimize the distance between the original image $\boldsymbol{x}$ and the predicted one to train the shared weight network. So, the loss function is:

$$l(\boldsymbol{x}) = d(x, g(f(x))) \tag{4}$$

In Function (4), the d represents distance between two images and we choose euclidean distance to compute the difference between original image and the generated one. In prototypical network, when squared euclidean distance is used in Equation (3), it is a linear model.[8] Since the digits are not linearly separable, we add a non-linear layer to map the feature vector to a high dimensional space where the digits can be separated with linear model. So, to learn a better model, the shared weight network is used to pre-train the non-linear layer, and then we fine-tune the pre-trained non-linear in prototypical network.

2.3 Cascading Classifier

The cascading classifier is a multistage algorithm. It builds a cascade of a parametric linear model which learn rules and a non-parametric k-nearest neighbor (k-NN) model which learns exception rejected by rule.[10] It also divides the training data into two parts: D_r for linear model to learn rules and D_{exc} for K-NN to collect exceptions. We use cascading classifier as the baseline to compare with prototypical network with or without using shared weights to pre-train it when the training data is reduced.

3 Experiment

The Optical Recognition of Handwritten Digits Data Set has been split into two clusters. The one written by 30 persons is used as training data and another by 14 persons as writer-independent test data. The training data is divided into three part: 1934 examples as training set, 946 examples as support set, and the 943 as writer-dependent test set. The training set is used as D_t for prototypical network and D_r for cascading network, and D_p of prototypical and D_{exc} of cascading classifier is given support set. We test these two models with writer-independent test data. To test their performance when the training data is small, we gradually reduce the examples in training set. The D_t is used to pre-train the non-linear layer with shared weights network, and we would compare the performance of prototypical network with or without the pre-training process to see if shared weights network can learn a better

non-linear function to map images to a linear separable space. We refine D_p and D_{exc} to keep 50 examples for each class and in total there will be 500 examples. To see their performance when data set is small, we would reduce the training dataset, D_t and D_r, to contain 500 examples.

3.1 Result

Network \ Metrics	Accuracy	Macro F1
Cascading	0.8894	0.8887
Proto.	0.9360	0.9354
Proto. with share weights	0.8647	0.8631

Table 1: result of experiments

In Table 1, when the training data is reduced to contain 500 examples, the accuracy and macro f1 score of cascading network is 0.8894 and 0.8887, and for prototypical network without pre-training they are 0.9360 and 0.9354. We can see the performance of prototypical network is much better than the cascading network. When the training dataset contains 1934 examples, the cascading classifier's accuracy is nearly 96%.[10] When training dataset is small, it performs much worse. When we pre-train the prototypical network with shared weights network, the accuracy and f1 score reduce to 0.8647 and 0.8631 which is worse than the prototypical network only and even the cascading network.

3.2 Discussion

From the result we can see that, the prototypical network without pre-training beats the cascading network when the training dataset is small. And with a small training dataset, the cascading classifier performs less better compared to trained in a large dataset. The main reason is that the cascading network just trains a linear model which learns rules and gives exceptions to its K nearest neighbor classifier.[10] The K nearest neighbor classifier performs as a non-linear classifier, but it is not trainable. When training data is small, the linear model cannot learn a good rule, and its bad performance will influence K nearest neighbor which cannot learn by itself. The K nearest neighbor doesn't perform well when the number of exceptions increase because of poor generalization of linear model. The prototypical network can learn a better non-linear mapping than the cascading network even the training dataset is small. When the non-linear layer of prototypical network is pre-trained with shared weights network, the whole prototypical network's performance becomes worse. So, we can conclude that it learns a non-linear function which is not suitable to prototypical network. Yet, in another aspect, the result of prototypical network with pre-training shows that the pre-training process will influence the final performance of prototypical network. Although shared weights network is not competent to learn a better non-linear mapping, there are other networks we can explore.

4 Conclusion and future work

In this paper we talk about the few-shot learning which aims to train a model which can have art-of-state performance for pattern recognition and classification with small training dataset. We propose a prototypical network which have better performance than cascading network on digits recognition when the training dataset is small. Since the digits' images are not linearly separable, the prototypical network uses a non-linear function to learn all non-linearity. We want to use shared weights network to pre-train the non-linear function to get better performance. Yet, the experiments shows that the shared weights network cannot learn a better non-linear function to map the original image to a linear separable space. Therefore, we can explore other networks to find one which is competent. Although the prototypical network can use few data for each class to get good performance on digits recognition, yet to predict a existing class it still needs some training data belonging to it. If there is not training data for a class, how can we train a model to get good performance? This problem is called zero-shot learning. So, in the future we can improve the prototypical network for zero-shot learning. In the introduction section of the paper, we mention another way to solve lack of data by using coarse labeled data. Many previous works use distant supervision to model the noise in the coarse labeled data. If the coarse labeled data contain some examples that cannot be classified

to existing class, we can also find a way to make the model act good generalization on those new examples. This is called meta-learning. Our future works can also focus on improve the prototypical networks to know how to learn these new examples that doesn't belong to existing classes.

References

1. S.Carey, and E.Bartlett. Acquiring a single new word. Papers and Reports on Child Language Development,pp.17-29, 1978.

2. B. M. Lake, J. Gross R. Salakhutdinov, and J. B. Tenenbaum. One shot learning of simple visual concepts. In Proceedings of the 33rd Annual Conference of the Cognitive Science Society , 2011

3. G.Koch, R.Zemel,and R.Salakhutdinov,"Siamese Neural Networks for One-shot Image Recognition", in ICML Lille: Proceedings of the 32nd International Conference on Machine Learning, Lille, France, July 6-11, 2015.

4. L.Zhang, M.Shi, Q.Chen. Crowd counting via scale-adaptive convolutional neural network. In Proceedings of the Computer Vision and Pattern Recognition 2017, 2017.

5. D.Blei, A.Ng, and M.Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Resarch. pp.993-1022, 2003.

6. B.Felbo, A.Mislove, A.Sgaard, I.Rahwan, and S.Lehmann.Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In Proceeding of the Conference on Empirical Methods in Natural Language Processing, 2017.

7. D.Dua and E. Taniskidou. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2017

8. J.Snell, K.Swersky, R.Zemel.Prototypical networks for few-shot learning. In the Proceeding of Conference on Neural Information Processing Systems 2017, 2017.

9. T.Gedeon, J.Catalan, and J.Jin. Image Compression using Shared Weights and Bidirectional Networks.

10. E.Alpaydin and C.Kaynak. Cascading Classifiers. Kybernetika, 1998.