Semi-Supervised Few-Shot Learning with Prototypical Networks

Rinu Boney and Alexander Ilin
The Curious AI Company
{rinu,alexilin}@cai.fi

Abstract

We consider the problem of semi-supervised few-shot classification (when the few labeled samples are accompanied with unlabeled data) and show how to adapt the Prototypical Networks [10] to this problem. We first show that using larger and better regularized prototypical networks can improve the classification accuracy. We then show further improvements by making use of unlabeled data.

1 Introduction

Few-shot learning addresses the problem of learning new concepts quickly, which is one of the important properties of human intelligence. In few-shot classification, the task is to adapt a classifier to previously unseen classes from just a few examples [5, 4, 11, 8]. This skill is useful in many practical applications since data annotation is laborious and a training set can be rather small. Recent works have focused on casting few-shot learning as a meta-learning problem, in which a model is trained on a variety of tasks to adapt quickly using a small number of training examples [1, 7, 8, 6]. Thus, the system learns to adapt to new problems using few labeled samples based on the knowledge transferred from its previous experience with related problems.

In some real-world problems, the tasks to which the learning system needs to adapt may contain both (few) labeled and (many) unlabeled examples, the problem known as semi-supervised learning. For example, a common feature of photo management applications is automatic organization of images based on limited interactive supervision from the user. The classes relevant to a specific user are likely to be different from the classes in publicly available image datasets such as ImageNet, thus there is a need for adaptation. The user can facilitate the adaptation by labeling a few images by personal preferences. In this application, the learning system also has access to lots of unlabeled images and it can make use of those to improve the classification accuracy. This is the problem of semi-supervised few-shot classification that we consider in this paper. We show that a model called Prototypical Networks [10] can be extended to handle unlabeled data and we demonstrate noticeable improvement on few-shot classification tasks.

2 Prototypical Networks

2.1 Supervised learning

In few-shot classification, the learning system needs to solve a classification task using a data set $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_T, y_T)\}$ with T labeled examples, where $\mathbf{x}_j$ denote inputs and y_j are the corresponding labels. To solve the few-shot classification, Prototypical Networks (PN) compute a representation of the inputs $\mathbf{x}$ using an embedding function g parameterized with $\boldsymbol{\gamma}$: $\mathbf{z} = g(\mathbf{x}, \boldsymbol{\gamma})$. Each class k is represented in the embedding space by a prototype vector which is computed as the

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

mean vector of the embedded inputs for all the examples S_k of the corresponding class k:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_j, y_j) \in S_k} g(\mathbf{x}_j, \boldsymbol{\gamma}) \,. \tag{1}$$

The distribution over predicted labels y for a new sample is computed using softmax over negative distances to the prototypes in the embedding space:

$$p(y = k | \mathbf{x}, \{\mathbf{c}_k\}) = \frac{\exp(-d(\mathbf{z}, \mathbf{c}_k))}{\sum_{k'} \exp(-d(\mathbf{z}, \mathbf{c}_{k'}))} \,. \tag{2}$$

The training process iterates between the following steps. A subset of classes is randomly selected to formulate one training task. The support set S and query set $Q = \{(x_1, y_1), ..., (x_V, y_V)\}$ are created by sampling examples from the selected classes. The prototypes are estimated using the support set S and (1). The parameters $\boldsymbol{\gamma}$ are updated so as to improve the likelihood computed on the query set:

$$\sum_{(\mathbf{x}_j, y_j) \in Q} \log p(y = y_j | \mathbf{x}_j, \{\mathbf{c}_k\}) \,,$$

which is computed using (2) with the estimated prototypes.

2.2 Semi-supervised learning

In the semi-supervised scenario, test tasks contain both labeled and unlabeled samples. The prototypes can first be estimated with the labeled data using (1). Then, the probability distributions of the missing labels can be computed with (2) and the prototypes can be re-estimated:

$$\mathbf{c}_k = \frac{\sum_j p(y = k | \mathbf{x}_j) \mathbf{z}_j}{\sum_j p(y = k | \mathbf{x}_j)} \,, \tag{3}$$

where the summation is done for all the samples. $p(y = k|\mathbf{x}_j)$ is either 0 or 1 for the labeled samples and is computed using (2) for the unlabeled samples. Iterating this re-estimation process corresponds to parameter estimation in a mixture-of-Gaussian model using the EM-algorithm. In practice, we do only one update (3) as we did not observe further improvements with more iterations. We also use an approximation of (3) by binarizing $p(y|\mathbf{x})$ so that the most probable class has the probability one and the remaining classes have the probability zero, which corresponds to hard assignment similar to the one used in the K-means algorithm. The PN model can be either trained normally using only labeled examples, or the prototypes can be re-estimated with the unlabeled samples also in the training procedure. In practice, both strategies yielded similar results in our experiments.

3 Experiments

3.1 Synthetic data set

To test the proposed algorithm, we created a synthetic data set which is fast to experiment with. The data set consists of a set of two-dimensional classification tasks with two classes, in which the optimal decision boundary is a sine wave (see Fig. 1). The amplitude A of the optimal decision boundary varies across tasks within $[0.1, 5.0]$ and the phase ϕ varies within $[0, 2\pi]$. The first dimension of the data samples is drawn uniformly from $[-5, 5]$ and the second dimension is computed as

$$x_2 = A \sin(x_1 + \phi) + c$$

where c is a noise term with the Laplace distribution with the mean ± 2 (depending on the class) and the scale parameter 0.5. We sampled 100 tasks for training and 1000 tasks for testing.

In this experiment, we use a fully connected network with two hidden layers of size 40 with ReLU nonlinearity as the embedding network. Examples of the decision boundaries produced by PNs on test tasks are shown in Fig. 1. Note that even for a small number of training examples the PN decision boundaries resemble the sine wave, thus the knowledge is transferred between tasks. The classification errors on test tasks depending on the number of unlabeled samples are shown in Table 1. One can see from the results that the PN accuracy improves with the use of unlabeled data. However, the improvement plateaus and the results do not improve much after adding more than 100 data points.

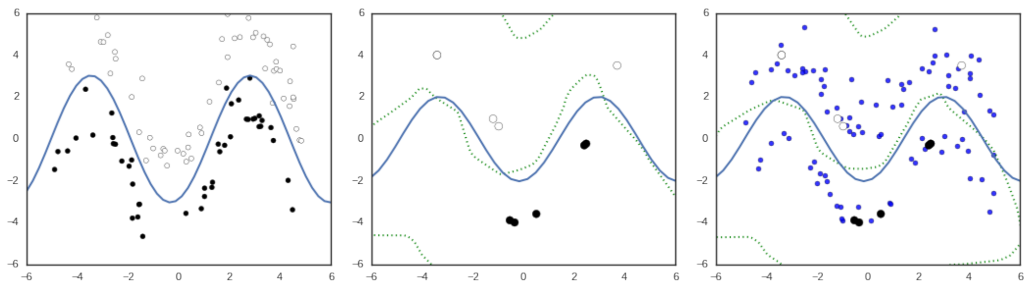

Figure 1: Left: Example task from the sine data set. The black dots correspond to samples of one class and the white dots correspond to samples from the other class. The optimal decision boundary is shown with the blue line. Middle: Example of adaptation to a test task from 10 labeled samples. Right: Example of adaptation to a test task from the same 10 labeled samples and 100 unlabeled samples. The blue dots correspond to unlabeled samples.

Table 1: Classification error on sine data set with 10 labeled samples and n extra samples.

n	labeled	unlabeled
0	6.38	6.38
10	4.97	5.54
100	2.98	4.70
1000	1.68	4.57

3.2 miniImagenet

We tested the PN in the semi-supervised scenario on the miniImagenet recognition task proposed in [11]. The dataset involves downsampled 84x84 images from 64 training classes, 12 validation classes, and 24 test classes from Imagenet. We use the same split as [8]. We follow the experimental setup which involves N-way K shot classification, similarly to [11] that is every task at test time contains N classes with K labeled examples from each class. For the semi-supervised case, we also assume the existence of M unlabeled samples per class at test time. Following previous works, we use 15 test samples per class for each task during training and for evaluation. We evaluate the model over 2400 tasks which involve the 24 classes reserved for testing.

One challenge with the miniImagenet data set is that it requires rather complex features but it contains relatively little amount of data. Therefore, preventing overfitting becomes an important issue. Most previous works [11, 8, 2, 10] used conventional convolutional networks with a small number of layers to prevent overfitting. Those networks may not be expressive enough to capture relevant complex features. In PNs [10], the model was also regularized by having more classes at training time than at test time (e.g., 30-way training and 5-way testing). In our experiments, we used a Wide Residual Network [12] as the embedding network. It was a network of depth 16 and a widen factor of 6. We had a 8×8 pooling with a stride of 4 at the end to obtain embeddings of dimensionality 384. The network was regularized with dropout with rate of 0.3. We used the Adam optimizer [3] with a learning rate of 0.01 for training. We perform early stopping to prevent overfitting.[1] We trained the model with 30 classes during training for 1-shot classification and 20 classes during training for 5-shot classification, similarly to the original PN paper. We tried varying the number of classes during training and observed that it had much smaller impact on the results compared to the observations in [10]. This suggests that tuning this regularization parameter is less important for the proposed architecture.

The results are presented in Table 2. One can observe that using the Wide ResNets to learn the embedding space yields noticeable improvements of the classification accuracy compared to the baseline methods. The improvements are more significant for the 20-way classification. The last row in Table 2 presents the results of the PN tested in the semi-supervised scenario: The prototypes are

[1]Residual Networks are typically trained using stochastic gradient descent with momentum and we expect better results by doing the same and fine-tuning the hyperparameters.

Table 2: Average classification accuracy (with 95% confidence intervals) on miniImagenet. The comparison numbers for the 20-way testing are from [6].

	5-way testing		20-way testing	
Model	1-shot	5-shot	1-shot	5-shot
fine-tuning baseline	28.86 ± 0.54	49.79 ± 0.79	–	–
nearest-neighbor baseline	41.08 ± 0.70	51.04 ± 0.65	–	–
Meta-LSTM [8]	43.44 ± 0.77	60.60 ± 0.71	16.70 ± 0.23	26.06 ± 0.25
Matching nets [11]	46.6	60.0	17.31 ± 0.22	22.69 ± 0.20
MAML [2]	48.70 ± 1.84	63.11 ± 0.92	16.49 ± 0.58	19.29 ± 0.29
ARC [9]	49.14	–	–	–
Meta Networks [7]	49.21 ± 0.96	–	–	–
PN [10]	49.42 ± 0.78	68.20 ± 0.66	–	–
Meta-SGD [6]	50.47 ± 1.87	64.03 ± 0.94	17.56 ± 0.64	28.92 ± 0.35
Resnet PN (ours)	$\mathbf{51.69 \pm 0.42}$	$\mathbf{69.57 \pm 0.64}$	$\mathbf{23.35 \pm 0.28}$	$\mathbf{41.10 \pm 0.25}$
Resnet PN, re-estimate (ours)	$\mathbf{54.05 \pm 0.47}$	$\mathbf{70.92 \pm 0.66}$	$\mathbf{23.59 \pm 0.31}$	$\mathbf{41.23 \pm 0.26}$

Table 3: Average classification accuracy (with 95% confidence intervals) on miniImagenet for the semi-supervised scenario. M is the number of unlabeled samples per class available at test time.

	5-way testing	
M	1-shot	5-shot
15	54.05 ± 0.47	70.92 ± 0.66
30	54.70 ± 0.46	71.86 ± 0.59
60	55.66 ± 0.46	72.21 ± 0.55
120	55.67 ± 0.45	72.55 ± 0.52

re-estimated at test time using both labeled samples and the inputs $\mathbf{x}$ from the test set. In Table 3, we show how the number of unlabeled examples at test time affects the classification accuracy of the trained PN. The results indicate that more unlabeled samples yields better performance, however, the improvement plateaus very quickly with the increase of the number of unlabeled samples. This agrees with the results obtained on the synthetic data reported in Section 3.1.

4 Conclusion

In this paper, we proposed an extension of the Prototypical Networks for few-shot semi-supervised classification. The preliminary results indicate that the proposed extension can improve the few-shot accuracy by using unlabeled data. However, the improvement plateaus quickly with the increase of the number of unlabeled data, which suggests that the algorithm may be further improved. We continue investigating other possible alternatives.

Acknowledgments

We would like to thank our colleagues from The Curious AI Company for fruitful discussions.

References

[1] Finn, C., Abbeel, P., and Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

[2] Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. (2017b). One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*.

[3] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[4] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

[5] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, **350**(6266), 1332–1338.

[6] Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-SGD: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*.

[7] Munkhdalai, T. and Yu, H. (2017). Meta networks. *arXiv preprint arXiv:1703.00837*.

[8] Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

[9] Shyam, P., Gupta, S., and Dukkipati, A. (2017). Attentive recurrent comparators. *arXiv preprint arXiv:1703.00767*.

[10] Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

[11] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., *et al.* (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

[12] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.