Few-Shot Learning

Bandla Manikanta
SR-No : 21562
MTech CSA
manikantab@iisc.ac.in

1 Introduction

1.1 Few-shot Learning

Learning quickly is a hallmark of human intelligence, whether it involves recognising objects from a few examples or quickly learning new skills after just minutes of experience. Humans have a remarkable ability to quickly grasp new concepts from a very small number of examples or a limited amount of experience, leveraging prior knowledge and context. In contrast, traditional deep learning approaches treat each task independently and hence are often data inefficient and are prone to overfitting the model on each task. Few-shot classification is a task in which a classifier must be adapted to accommodate new classes not seen in training, given only a few examples of each of these classes.

1.2 Prototypical Networks

Prototypical networks[5] address few-shot learning by learning a metric space in which classification can be performed by computing distances to prototype representations of each class. Prototypical networks, is based on the idea that there exists an embedding space in which points form cluster around a single prototype representation for each class. In order to do this, we learn a non-linear mapping of the input into an embedding space using a neural network and take a class's prototype to be the mean of its support set in the embedding space. Classification is then performed for an embedded query point by simply finding the nearest class prototype.

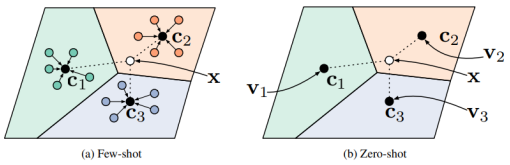

Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes c_k are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes c_k are produced by embedding class meta-data v_k. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_\phi(y = k|x) \propto \exp\left(-d(f_\phi(x), c_k)\right)$

1.3 Model-Agnostic Meta-Learning

The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. Meta-learning algorithm[6] that is general and model-agnostic, in the sense that it can be directly applied to any learning problem and model that is trained with a gradient descent procedure. The key idea underlying MAML method is to train the model's initial parameters such that the model has maximal performance on a new task

after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task.

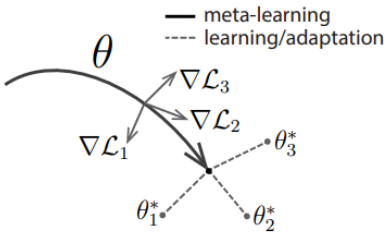

Figure 2: Diagram of model-agnostic meta-learning algorithm (MAML), which optimizes for a representation that can quickly adapt to new tasks.

2 Inspired Papers

My work is inspired by the papers "Protypical Networks for Few-Shot Learning" by Jake Snell, Kevin Swersky, and Richard S. Zemel, published at the 31st Conference on Neural Information Processing Systems (NIPS) in 2017 and "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" by Chelsea Finn, Pieter Abbeel Sergey Levine.

3 Literature Survey

Few-shot learning is a challenging problem in computer vision, which aims to classify unseen classes with only a few labelled examples per class. The metric-based approach has been widely used to tackle this problem by computing a similarity metric between the query image and a set of support images to predict the class label. In this literature survey, we will discuss the recent advances in metric-based few-shot learning.

Matching Networks (MNs) proposed by Vinyals et al. [4] (2016) is one of the pioneering works in the field of few-shot learning. MNs are based on the idea of learning a distance metric that can generalize to new classes. The authors used a simple k-nearest neighbour (KNN) algorithm to classify test images by computing the distance between the test image and the support set. However, this approach is computationally expensive, as it requires computing the distance between the test image and all support images.

Neighborhood Components Analysis (NCA)[2] learns a Mahalanobis distance to maximize K-nearest-neighbor's (KNN) leave-one-out accuracy in the transformed space. Salakhutdinov and Hinton extend NCA by using a neural network to perform the transformation. Large margin nearest neighbor (LMNN) classification [30] also attempts to optimize KNN accuracy but does so using a hinge loss that encourages the local neighborhood of a point to contain other points with the same label.

Another relevant few-shot learning method is the meta-learning approach proposed in Ravi and Larochelle [3]. The key insight here is that LSTM dynamics and gradient descent can be written in effectively the same way. An LSTM can then be trained to itself train a model from a given episode, with the performance goal of generalizing well on the query points. Matching networks and prototypical networks can also be seen as forms of meta-learning, in the sense that they produce simple classifiers dynamically from new training episodes; however the core embeddings they rely on are fixed after training.

4 Methodology

4.1 Prototypical Networks

4.1.1 Algorithm

We extract the embedding features of each input image by training an embedding architecture that contains four convolutional blocks. Each block comprises a 64 filter 3×3 convolution, batch normalization layer, a ReLU non-linearity and a 2×2 max-pooling layer. The embedding function $f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$ with learnable parameters ϕ, that converts each input image to M-dimensional embedding feature. A different set of training tasks, validation tasks and test tasks are formed by randomly selecting a subset of classes from the dataset. The classes present in each of the training tasks, validation tasks, and test tasks are disjoint. Each training task contains support set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$ and a query set $Q = \{(x_1, y_1), (x_2, y_2), ..., (x_P, y_P)\}$, where support set contains images from K-different classes with N different images from each class. The query set contains images from the same K classes but images are different from the support set. The prototype for each class is calculated as the mean of embedding features of images of that specific class in a task.

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

The embedding architecture learns the parameters ϕ such that the embedding feature space of each new task forms nice clusters, such that the class prototypes can be used as a representative of each class and a test image can be classified as a class whose prototype is closest. The distance metric used to find the closest prototype is Euclidean distance. The posterior probability of a query point x is calculated based on the softmax over distances to the prototypes in the embedding space.

$$p_\phi(y = k \mid x) = \frac{\exp\left(-d(f_\phi(x), c_k)\right)}{\sum_{k'} \exp\left(-d(f_\phi(x), c_{k'})\right)} \tag{2}$$

where $k^{'}$ refers to classes other than k.

The loss is calculated as $J(\phi) = -\log p_\phi(y = k/x)$ where k is the true class of x. The embedding architecture is trained on a different set of training tasks using an SGD optimiser. The validation tasks can be used for selecting better model. The testing tasks are used to evaluate our learned embedding architecture parameters.

4.1.2 Prototypical Networks as a Linear Model

The model in Equation(2) can be interpreted as a linear discriminant model with appropriate parameters. For a given x we assign it to class k if the prototype of class k is closer to $f_\phi(x)$.

$$d_k(x) = -||f_\phi(x) - c_k||^2 = -f_\phi(x)^\top f_\phi(x) + 2c_k^\top f_\phi(x) - c_k^\top c_k \tag{3}$$

We assign x to class k whose $d_k(x) > d_{k'}(x)\ \forall k^{'} \neq k$. The first term in Equation(3) is constant with respect to the class k, so it does not affect in deciding the class.

$$d_k(x) = 2c_k^\top f_\varphi(x) - c_k^\top c_k = w_k^\top f_\varphi(x) + b_k \tag{4}$$

where $w_k = 2c_k$ and $b_k = -c_k^\top c_k$. So the discriminant functions are linear.

4.2 Meta-Learning

4.2.1 Algorithm

We consider a model represented by a parametrised function f_θ with parameters θ. When adapting to a new task T_i, the model's parameters θ become $\theta_i^{'}$.

In our method, the updated parameter vector $\theta_i^{'}$ is computed using one or more gradient descent updates on task T_i. For example, when using one gradient update,

$$\theta_i^{'} = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta) \tag{5}$$

$$L_{T_i}(f_\theta) = \sum_{x(j), y(j) \sim T_i} y(j) \log f_\theta(x(j)) + (1 - y(j)) \log(1 - f_\theta(x(j))) \tag{6}$$

where $x(j)$ and $y(j)$ are an input/output pair sampled from task T_i.

The step size α may be fixed as a hyperparameter or metalearned. For simplicity of notation, we will consider one gradient update for the rest of this section, but using multiple gradient updates is a straightforward extension. The model parameters are trained by optimising for the performance of $f_{\theta_i'}$ with respect to θ across tasks sampled from $p(T)$. More concretely, the meta-objective is as follows:

$$\min_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'}) = \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \nabla_\theta L_{T_i}(f_\theta)}) \tag{7}$$

Note that the meta-optimization is performed over the model parameters θ, whereas the objective is computed using the updated model parameters $\theta^{'}$. In effect, our proposed method aims to optimise the model parameters such that one or a small number of gradient steps on a new task will produce maximally effective behaviour on that task. The meta-optimization across tasks is performed via stochastic gradient descent (SGD), such that the model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'}) \tag{8}$$

where β is the meta step size.

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

Figure 3: MAML algorithm for classification problems

5 Embedding function architecture

The MAML and Prototypical model follows the same architecture as the embedding function which has 4 modules with a 3 × 3 convolutions and 64 filters, followed by batch normalization, a ReLU nonlinearity. The Omniglot images are of size 28 × 28, so the dimensionality of the last hidden layer is 64.

The MAML model has an extra Fully connected layer at the end of the embedding function architecture of size as the number of classes(Way) in the support set of the task.

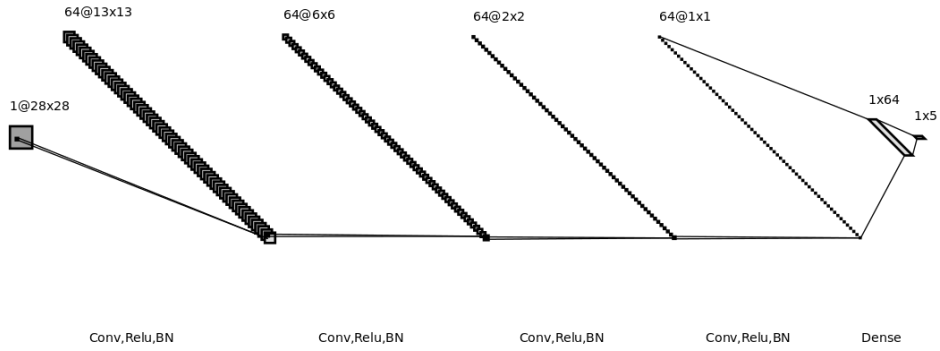

Figure 4: CNN architecture of Meta-learning for classification

6 Experiments

6.1 Datasets

Omniglot `https://github.com/brendenlake/omniglot` is a dataset of 1623 handwritten characters collected from 50 alphabets. After applying different augmentations on the dataset, divided the classes as a training set with 4112 classes, a validation set with 688 classes and a test set with 1692 classes.

6.2 Codes

6.2.1 Prototypical networks

I have used the PyTorch implementation of the paper from `https://github.com/orobix/Prototypical-Networks-for-Few-shot-Learning-PyTorch`. I trained the prototypical networks using Euclidean distance in the 1-shot and 5-shot scenarios with training episodes containing 60 classes and 5 query points per class. I have maintained the train shot that matches the test shot. But the way(Number of classes in the training set) is set as 60 for any underlying test set. I trained the model for 100 epochs, wherein each epoch trained over 100 randomly generated tasks from the train set. After each epoch, the model was validated over 100 randomly generated tasks from the validation set. Finally, the trained model is tested on 1000 randomly generated tasks from the test set. The results I have achieved by training the model from scratch are very close to what the authors have claimed in the paper.

I have also tested the trained model on checking how better the model trained on Omniglot is able to adapt to the MNIST dataset. The trained model was tested with a support set of tasks containing $N*K$ (K-shot N-way) examples and a query set containing 15 examples of each class. The model was able to adapt to the MNIST dataset and achieve 74% accuracy with just one example from each class.

Table 1: Few-shot classification accuracies of Prototypical networks on Omniglot dataset

Model	5-way Acc		20-way Acc	
	1-shot	5-shot	1-shot	20-shot
Prototypical Networks(Original)	98.8	99.7	96.0	98.9
Attached Git-repo implementation	98.2	99.5	94.7	98.4
Adaptation to MNIST	74.5	90.0	–	–

6.2.2 Meta-learning algorithm

For N-way, K-shot classification, each gradient is computed using a batch size of N∗K examples. For Omniglot, the 5-way convolutional model was trained with 1 gradient step with step size $\alpha = 0.4$ and a meta batch size of 32 tasks. The network was evaluated using 3 gradient steps with the same step size $\alpha = 0.4$. The 20-way convolutional MAML model was trained and evaluated with 5 gradient steps with step size $\alpha = 0.1$. During training, the meta batch size was set to 16 tasks. I have used the PyTorch implementation of the paper from `https://github.com/dragen1860/MAML-Pytorch`.

The model was trained for 40,000 epochs, wherein each epoch a random meta-task from the train set is taken and trained the model on that meta-task. The trained model was tested on a test dataset of Omniglot for 1000 epochs, wherein for each epoch a random task is taken and tested on that task.

Table 2: Few-shot classification accuracies on Omniglot

Model	5-way Acc		20-way Acc	
	1-shot	5-shot	1-shot	20-shot
MAML(Original)	98.7	99.9	95.8	98.9
Attached Git-repo implementation	94.4	98.3	84.7	92.7
Adaptation to MNIST	58.5	79.2	–	–

MAML model continues to improve with additional gradient steps, despite being trained for maximal performance after 5 gradient steps. But taking more gradient steps during meta-training will lead the model to overfit to that specific task, which cannot be better generalized to a new task.

Figure 5: The above plot depicts MAML accuracy with respect to the number of gradient steps at meta-testing. MAML model was able to adapt to test tasks with 2-3 gradient steps, but moving ahead with more gradient steps there is no significant improvement in accuracy.

7 Conclusion

I have given two different ways of solving the problem of the Few-shot learning classification problem. One is Prototypical networks which is based on the idea that we can represent each class by the mean of its examples in a representation space learned neural network. The other is the meta-learning method based on learning easily adaptable model parameters through gradient descent. The MAML meta-gradient update involves a gradient through a gradient. Computationally, this requires an additional backward pass through f to compute Hessian-vector products. Prototypical networks is far simpler, and results are slightly better than the meta-learning approach. MAML is

model-agnostic in the sense it can be directly applied to any learning problem and model that is trained with gradient descent procedures such as regression and classification problems. Though the Prototypical networks showed better results, MAML is very intuitive in the sense of learning the general representation of parameters during meta-training and adapting the parameters to the new task with few gradient updates during testing. in contrast, the parameters of the embedding function in Prototypical Networks remain unchanged once training is completed. The MAML approach is also further pursued by the paper LEO [7] improving the gradient descent to perform in low-dimensional parametric space. The paper was also shown to outperform the results of Prototypical Networks.

References

[1] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[2] Jacob Goldberger, Geoffrey E. Hinton, Sam T. Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2004.

[3] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[4] 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.

[5] Snell, J., Swersky, K., Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems* 2017.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning,*

[7] Francesca Bovet, Jean-Baptiste Cordonnier, and Michal Valko. META-LEARNING WITH LATENT EMBEDDING OPTIMIZATION. In Proceedings of the International Conference on Machine Learning, 2019.