AIP Project

Anmol Garg **20178**

Few Shot Image Classification

Prototypical Network and Matching Network

FewShot Classification

Most Deep Learning Networks require huge amount of data to train the model but human level intelligence tend to classify objects given only a single image, to solve this problem Few Shot Classification terminology is defined .

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. A naive approach, such as re-training the model on the new data, would severely overfit. While the problem is quite difficult, it has been demonstrated that humans have the ability to perform even one-shot classification, where only a single example of each new class is given, with a high degree of accuracy. Few-shot classification aims to learn a classifier to recognize unseen classes during training with very less labelled data.

Few shot image classification is achieved by three techniques as follows :-

1. **Initialization based methods** learning to finetune or to learn good model initialization (Model Agnostic Meta Learning → MAML and MAML++ algorithms)
2. **Distance metric learning based methods** learning to compare or differentiate on basis of distance between two feature vectors (Prototypical Networks and Matching Networks)
3. **Hallucination based methods** learning to augment data, basically to generate additional data using various data augmenting techniques or using GAN models

Here in this project I have tested and evaluated the Distance metric learning based method namely Prototypical Networks and Matching Networks on Omniglot dataset and produced the accuracy results.

1. Matching Networks : "A differentiable nearest-neighbours classifier"
2. Prototypical Netwroks : "Learning prototypical representations"

The N-shot,K-way task

The ability of an algorithm to perform few-shot learning is typically measured by its performance on n-shot, k-way tasks. These are run as follows :

1. A model is given a query sample belonging to a new , previously unseen class.
2. It is also given a support set, S ,consisting of n examples each from K different unseen classes
3. The algorithm then has to determine which of the support set classes the query belongs

OMNIGLOT DATASET:

Omniglot is a dataset of 1623 handwritten characters collected from 50 alphabets. There are 20 examples associated with each character, where each example is drawn by a different human subject. We follow the procedure of resizing the grayscale images to 28*28 and augmenting the character classes with rotations in multiples of 90 degrees. We use Kaggle dataset where it is divided into images_background and images_evaluation where different classes are there in each folder for training and testing.

Matching Networks

** *Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In Advances in Neural Information Processing Systems (NIPS), 2016*

Matching Networks first embed a high dimensional sample into a low dimensional space and then perform a generalised form of nearest-neighbours classification described by the equation below.

$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

Here prediction of the model, *y^*, is the weighted sum of the labels, *y_i*, of the support set, where the weights are a pairwise similarity function, *a(x^, x_i),* between the query example, x^, and a support set samples, *x_i*. The labels *y_i* in this equation are one-hot encoded label vectors.

The authors chose a straightforward softmax over cosine similarities in the embedding space as their attention function *a(x, x_i).* The embedding function they use for their few-shot image classification problems is a CNN which is differentiable and hence making the attention and Matching Networks fully differentiable which implies it is straightforward to fit the whole model end-to-end with typical methods such as stochastic gradient descent.

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i)))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j)))}$$

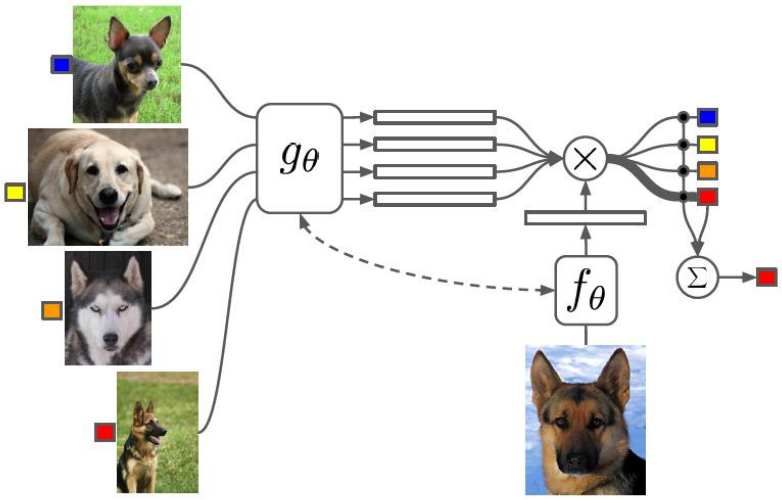


Figure 1: Matching Networks architecture

Full Context Embeddings

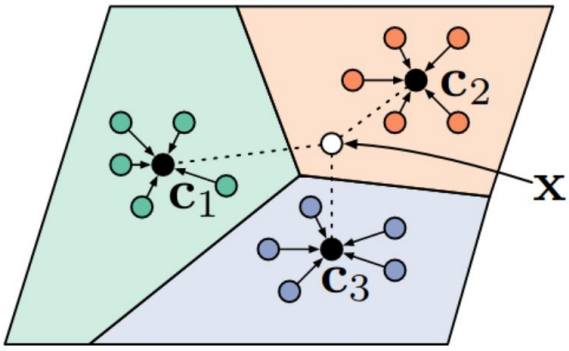
The myopic nature of the embedding functions is not optimal in the sense that each element of the support set *x_i* gets embedded by *g(x_i)* in a fashion that is independent of the rest of the support set and the query sample. Authors proposed that the embedding functions *f(x^)* and *g(x_i)* should take on the more general form *f(x^, S)* and *g(x_i, S)* where *S* is the support set. The reasoning behind this is that if two of the support set items are very close, e.g. we are performing fine-grained classification between dog breeds, we should change the way the samples are embedded to increase the distinguishability of these samples. A bidirectional Long-Short Term Memory (LSTM) to encode x_i in the context of the support set S, considered as a sequence.

Prototypical Networks

** Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems (NIPS), 2017**

Prototypical Networks, is based on the idea that there exists an embedding in which points cluster around a single prototype representation for each class. In order to do this, a non-linear mapping of the input is learnt 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.

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$$



Class prototypes c_i and query sample x.

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:

$$p_{\boldsymbol{\phi}}(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_{k'}))}$$

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

Algorithm implemented as given in paper

Algorithm 1 Training episode loss computation for Prototypical Networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss J for a randomly generated training episode.

$V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples
  $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_{k'})) \right]$ ▷ Update loss
  end for
end for

Difference between ProtoNetwork and MatchingNetwork

Distance metric

Another contribution of this paper is that the authors gave an argument to use euclidean distance over cosine distance in metric learning that also justifies the use of class means as prototypical representations. The key is to recognise that squared euclidean distance (but not cosine distance) is a member of a particular class of distance functions known as **Bregman divergences**.

It is known that centroid of the cluster that minimises the distance between each cluster point and centroid itself is just the mean of all the cluster points if the distance metric is Bregman Divergence.This centroid is the point that minimises the loss of information when representing a set of points as just a single point.

Episode composition

Episode during training method as implemented in matching network is to choose Nc classes and NS support points per class in order to match the expected situation at test-time. That is, if we expect at test-time to perform 5-way classification and 1-shot learning, then training episodes could be comprised of Nc = 5, NS = 1. In proto-network, it is shown to be extremely beneficial to train with a higher Nc, or "way", than will be used at test-time.

(difference of 2% has been seen for this episodic method with respect to one used in matching networks)

Results (Omniglot) :

ACCURACY	Distance metric	5-way(1 shot)	5-way(5 shot)	20-way(1 shot)	20-way(5 shot)
Matching Net.	Cosine	94.4	98.4	89.2	98.2
Prototypical Net.	Euclid.	95.8	99.3	92.4	98.3

ACCURACY	Distance metric	5-way(1 shot) (10 way trained)	5-way(5 shot) (10 way trained)	20-way(1 shot) (30 way trained)	20-way(5 shot) (30w trained)
Prototypical Net.	Euclid.	96.8	98.92	92.8	98.52