Project Proposal

Bao Pham

March 14, 2022

1 Introduction

Nowadays, machine learning has proven to be successful in data-intensive applications. However, when data is scarce, the performance of ML models is hampered. As a result, Few-Shot Learning (FSL) is proposed to tackle such a problem.

To introduce FSL, first recall the definition of machine learning:
A computer program is said to learn from experience E with respect to some classes of task T and performance measure P if its performance can improve with E on T measured by P [3, 7].

FSL [1] is a type of machine learning problems, where E contains a limited number of examples with supervised information of the target $\mathbf{T}$ [5,7]. Specifically, given a learning task T, FSL deals with a data set $D = \{D_{train}, D_{test}\}$ consisting of a training set $D_{train} = \{(x_i, y_i)\}_{i=1}^{I}$, where I is small, and a testing set $D_{test} = \{x^{test}\}$ [7]. A FSL algorithm searches a hypothesis space $\mathcal{H}$ to find the θ that parameterizes the best $h^* \in \mathcal{H}$ —its performance is measured by a loss function $L(\hat{y}, y)$, where $\hat{y} = h(x; \theta)$.

Fundamentally, in the supervised setting, given a hypothesis h the goal is to minimize its expected risk R which is the loss measured with respect to $p(x, y)$, the ground-truth joint probability distribution of input x and output y. The empirical risk of the training set D_{train} of I samples is then:

$$R_I(h) = \frac{1}{I} \sum_{i=1}^{I} l(h(x_i), y_i)$$

[1]Remark: When there is only one example with supervised information in E, FSL becomes one-shot learning. Additionally, if E has no examples for the target $\mathbf{T}$, FSL is then becomes a zero-shot learning problem [7].

However, due to I being small, empirical risk minimization is no longer reliable in FSL —$R_I(h)$ may be far from being a good approximation of the expected risk $R(h)$, and the resultant minimizer h_I overfits [1, 5–7].

To alleviate the problem of having an unreliable empirical risk minimizer in FSL supervised setting, prior knowledge is used. Specifically, according to [7], a FSL method use prior knowledge to either:

(a). Augment D_{train} and increase I to $\tilde{I}$, where $\tilde{I} \gg I$. Other machine learning models and algorithms can then be used on the augmented data to produce a more accurate empirical risk minimizer h_I.

(b). Constraint the complexity of $\mathcal{H}$ to produce a smaller hypothesis space $\tilde{\mathcal{H}}$ allowing D_{train} to be sufficient to learn a reliable h_I.

(c). Search for the θ which parameterizes the best hypothesis h^* in $\mathcal{H}$.

Altogether, this proposal aims to improve the robustness of an approach of the second FSL method type called prototypical network.

2 Prototypical Network

Prototypical network (ProtoNet) is an approach inspired from matching network proposed by Vinyals et al. [6]. It is worth noting both models utilize sampled mini-batches called episodes during training, where each episode is designed to mimic the few-shot task by sub-sampling classes and data points [5] —this use of episodic training (or meta-learning) enables models to be more robust.

Meanwhile, matching network meta-learns different embedding functions (f and g) for the training sample (support set) x_i and test sample (query set) x_{test}, and uses an attention mechanism to predict the classes for x_{test}. This approach can be thought as a weighted nearest-neighbor classifier applied within an embedding space [5–7]. In contrast, instead of comparing $f(x_{test})$ with each $g(x_i)$, ProtoNet compares $f(x_{test})$ with the class prototypes in D_{train} [5]. For class k, its prototype is computed as:

$$c_k = \frac{1}{N_k} \sum_{i=1}^{N_k} g_\phi(x_i)$$

where N_k x_i's are from class k and $g_\phi(\cdot)$ is an embedding function with learnable parameters ϕ.

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow [0, +\infty)$, ProtoNet produces a distribution over classses for a query point x_q based on a softmax over distances to the prototypes in the embedding space [5]:

$$p_\phi(y = k|x_q, S) = \frac{\exp(-d(g_\phi(x_q), c_k))}{\sum_{k'} \exp(-d(g_\phi(x_q), c_{k'}))}$$

The model is then learned through minimizing the negative log-probability $J(\phi) = -\log(p_\phi(y = k|x))$ of the true class k via stochastic gradient descent (SGD) [5]. Hence, the process of ProtoNet empirically leads to more stable results and reduces the computation cost in contrast to matching network [7].

3 Proposal

Learning in the presence of outliers is a challenge in machine learning. Since D_{train} is limited in the FSL setting, outliers in a small support set of N labeled examples $S = \{(x_1, y_1), ..., (x_N, y_N)\}$, where each $x_i \in \mathbb{R}^D$ is a $D-$dimensional feature vector of an example and $y_i \in \{1, ..., K\}$ is the corresponding label, can degrade prediction of the prototype c_k corresponding to S_k or the set of examples labeled with class k —the prototype representation of a certain class will be seriously drifted [9].

To alleviate such a problem, the prototype function c_k is proposed to be changed into a weighted average function [2]:

$$c_k = \frac{\sum_{i=1}^{N_k} g_\phi(x_i)\,\omega}{\sum_{m,n} \omega_{mn}}$$

where $\omega \in \mathbb{R}^{M \times M}$ and $g_\phi(x_i) \in \mathbb{R}^M$.

Additionally, given the loss function $J(\phi)$, it is possible to apply trimmed regularization as a way to handle outliers [8]. In other words, outlliers are handled by trimming observations (support samples in this case) with large residuals in terms of J: given a collection of n samples, $D = \{Z_1, ..., Z_n\}$, the problem is:

$$\underset{\phi \in \Omega,\, \omega \in \{0,1\}^n}{\text{minimize}} \sum_{i=1}^{n} \omega_i J(\phi, Z_i) \quad \text{s.t.} \sum_{i=1}^{n} \omega_i = n - h$$

[2] I think c_k can be furthered improve in a different way —I am still thinking about it although.

where Ω denotes the parameter space and this problem amounts to trimming h outliers as ϕ is learned [8].

To compare results, experimentation is proposed to be carried out as delineated in [2]. In this work, Mazumder et al. [2] proposes an inference method RNNP that uses a nearest neighbor prototype-based based evaluation procedure to improve robustness. Since the computation of prototypes unknowingly utilizes corrupted support examples, RNNP first generate N_u number of unlabeled hybrid features via combining features of support images using a proportion hyperparameter α:

$$x_u = \alpha \times x_i^{(k)} + (1 - \alpha) \times x_j^{(k)}$$

where x_u is the generated unlabeled hybrid feature, $x_i^{(k)}$ and $x_j^{(k)}$ are support samples of class k, where $i \neq j$ and $\alpha \in (0, 1)$ [2].

Using the class prototypes c_k as the initial centroids, soft k-means clustering is performed on a combined set of support image features, unlabeled hybrid features, and a corresponding query image feature [2]. Soft labels are assigned to the support and hybrid features, and a single query feature. The centroids are then updated using these soft labels for three iterations to obtain the refined class prototypes c_k^*. Since the corrupted support features are close to the non-corrupted ones, this process aims to reduce the influence of the corrupted labels on the class prototypes. Finally, $p_\phi(y = k, |x_q)$ is calculated.

$$p_\phi(y = k, |x_q) = \frac{\exp(-d(g_\phi(x_q), c_k^*))}{\sum_{k'} \exp(-d(g_\phi(x_q), c_{k'}^*))}$$

Altogether, the proposal aims to train a ProtoNet on the mini-ImageNet [6] and tiered-ImageNet [4] data sets with the proposed modifications: change c_k calculation and convert $J(\phi)$ to a trimmed loss problem. Then, apply RNNP [2] for the evaluation phase and compare the results.

References

[1] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[2] Pratik Mazumder, Pravendra Singh, and Vinay P. Namboodiri. Rnnp: A robust few-shot learning approach. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2663–2672, 2021.

[3] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[4] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, H. Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. *ArXiv*, abs/1803.00676, 2018.

[5] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc.

[6] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[7] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), jun 2020.

[8] Jihun Yun, Peng Zheng, Eunho Yang, Aurélie C. Lozano, and Aleksandr Y. Aravkin. M-estimation with the trimmed l1 penalty. *arXiv: Statistics Theory*, 2018.

[9] Junjie Zhu, Xiaodong Yi, Naiyang Guan, and Hang Cheng. Robust reweighting prototypical networks for few-shot classification. In *2020 6th International Conference on Robotics and Artificial Intelligence*, ICRAI 2020, page 140–146, New York, NY, USA, 2020. Association for Computing Machinery.