Exploring Representation Learning for Flexible Few-Shot Tasks Supplementary Materials

Anonymous Author(s)
Affiliation
Address
email

A Flexible Few-Shot Toy Problem

In this section we give details on a toy problem that illustrates the challenges introduced by the flexible few-shot learning setting and the failures of existing approaches on this task. This simple model captures the core elements of our flexible few-shot tasks, including ambiguity, domain shift from training to test time, and the role of learning good representations. The primary limitation of this model is the fact that it is fully linear — in a more realistic FFSL task recovering a good representation from the data is significantly more challenging, and the data points will have a more complex relationship with the attributes as in our benchmark datasets.

Problem setup We define a flexible few-shot learning problem where the data points $\mathbf{x} \in \mathbb{R}^m$ are generated from binary attribute strings, $\mathbf{z} \in \{0,1\}^d$, with $\mathbf{x} = A\mathbf{z} + \boldsymbol{\zeta}$ for some matrix $A \in \mathbb{R}^{m \times d}$ with full column rank and noise source $\boldsymbol{\zeta}$. Thus, each data point $\mathbf{x}$ is a sum of columns of A with some additive noise.

We consider contexts that classify the examples as positive when two attributes are 1-valued, and negative otherwise (an AND attribute context). For the training episodes, the contexts depend only on the first $d_1 < d$ attributes. At test time, the episode contexts depend on the remaining $d - d_1$ attributes. The episodes are generated by sampling a context uniformly, and then uniformly sampling k data points with positive labels and k with negative labels.

Linear prototypical network Now, consider training a prototypical network on this data with a linear embedding network, $g(\mathbf{x}) = W\mathbf{x}$. Within each episode, the prototypical network computes the prototypes for the positive and negative classes,

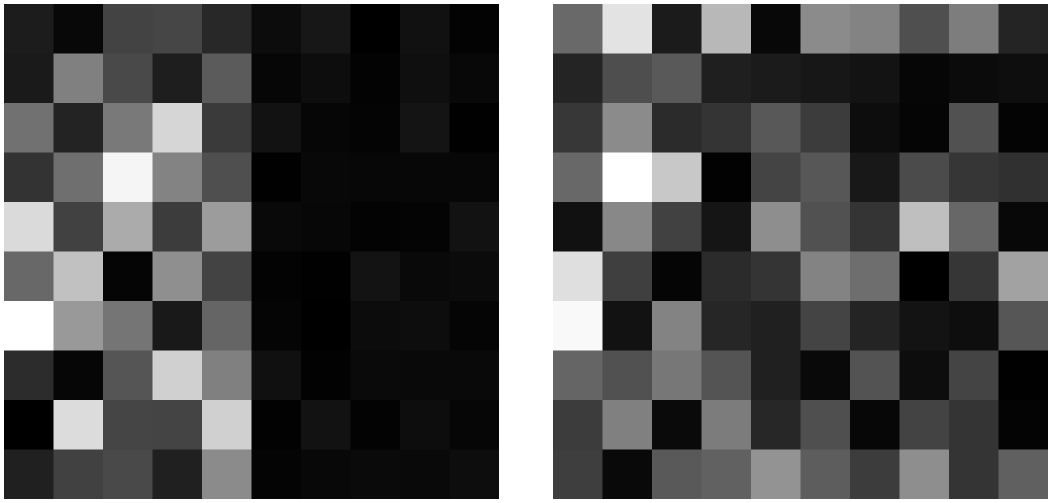
$$\mathbf{c}_j = \frac{1}{k} \sum_{\mathbf{x}_i \in S_j} g(\mathbf{x}_i), = \frac{1}{k} \sum_{\mathbf{x}_i \in S_j} \sum_{l=1}^{d} z_{il} W \mathbf{a}_l, \text{ for } j \in \{0,1\},$$

where S_j is the set of data points in the episode with label j, and $\mathbf{a}_l$ is the l^{th} column of the matrix A. Further, the prototypical network likelihood is given by,

$$p(y=0|\mathbf{x}) = \frac{\exp\left\{-\|W\mathbf{x} - \mathbf{c}_0\|_2^2\right\}}{\exp\left\{-\|W\mathbf{x} - \mathbf{c}_0\|_2^2\right\} + \exp\left\{-\|W\mathbf{x} - \mathbf{c}_1\|_2^2\right\}}.$$

The goal of the prototypical network is thus to learn weights W that lead to small distances between data points in the same class and large distances otherwise. In the flexible few-shot learning tasks, there is an additional challenge in that class boundaries shift between episodes. The context defining the boundary is unknown and must be inferred from the episode. However, with few shots (small k) there is ambiguity in the correct context — with a high probability that several possible contexts provide valid explanations for the labels.

Submitted to the 4th Workshop on Meta-Learning at NeurIPS 2020, Vancouver, Canada. Do not distribute.



(A) Flexible task learned features (B) Standard task learned features

Figure 1: Projecting data features into prototypical network embedding space (WA) for the linear toy problem. On the flexible task, the model destroys information from the test attributes to remove ambiguity at training time.

Fitting the prototypical network Notice that under our generative model, with $\mathbf{x} = W\mathbf{z} + \boldsymbol{\zeta}$ we have,

$$W\mathbf{x} - \mathbf{c}_j = WA(\mathbf{z} - \frac{1}{k}\sum_{\mathbf{z}_i \in S_j} \mathbf{z}_i) + \frac{1}{k}\sum_i W\boldsymbol{\zeta}_i + W\boldsymbol{\zeta}, \text{ for } j \in \{0, 1\}.$$

Notice that if $A(\mathbf{z} - \frac{1}{k}\sum_{\mathbf{z}_i \in S_j} \mathbf{z}_i) \in \text{Ker}(W)$ then the entire first term is zero. If $\mathbf{z} \in S_j$ then there is no contribution from the positive attribute features in this term. Otherwise, this term is guaranteed to have some contribution from the positive attribute features.

Therefore, if W projects to the linear space spanned by the positive attribute features then the model will be able to solve the episode without ambiguity. This suggests that the optimal weights are those that project to the set of features used in the training set — destroying all information about the test attributes which would otherwise introduce ambiguity.

We observed this effect empirically in Figure 1, where we have plotted the matrix $\text{abs}(WA)$. Each column of these plots represents a column of A mapped to the prototypical network's embedding space. The first 5 columns correspond to attributes used at training time, and the remaining 5 to those used at test time.

In the flexible task described above, the learned prototypical feature weights project out the features used at test time (the last 5 columns). As a result, the model achieves 100% training accuracy but only 51% test accuracy (chance is 50%). We also compared against a similar problem set up that resembles the standard few-shot learning setting. In this setting, the binary attribute strings may have only a single non-zero entry and each episode is a binary classification problem where the learner must distinguish between two classes. In this standard few-shot setting, the model is not forced to throw away test-time information and achieves 100% training accuracy and 99% test accuracy.

Settings for Figure 1 We use 10 attributes, 5 of which are used for training and 5 for testing. We use a uniformly random sampled $A \in \mathbb{R}^{30\times 10}$ and the prototypical network learns $W \in \mathbb{R}^{10\times 30}$. We use additive Gaussian noise when sampling data points with a standard deviation of 0.1. The models are trained with the Adam optimizer using default settings over a total of 30000 random episodes, and evaluated on an additional 1000 test episodes. We used $k = 20$ to produce these plots, but found that the result was consistent over different shot counts.

Table 1: **Attribute split for Celeb-A**

Train	5_o_Clock_Shadow	Black_Hair	Blond_Hair	Chubby
	Double_Chin	Eyeglasses	Goatee	Gray_Hair
	Male	No_Beard	Pale_Skin	Receding_Hairline
	Rosy_Cheeks	Smiling		
Val/Test	Bald	Bangs	Brown_Hair	Heavy_Makeup
	High_Cheekbones	Mouth_Slightly_Open	Mustache	Narrow_Eyes
	Sideburns	Wearing_Earrings	Wearing_Hat	Wearing_Lipstick
	Wearing_Necktie			

B Dataset split

We include the attribute split for Celeb-A in Table 1.