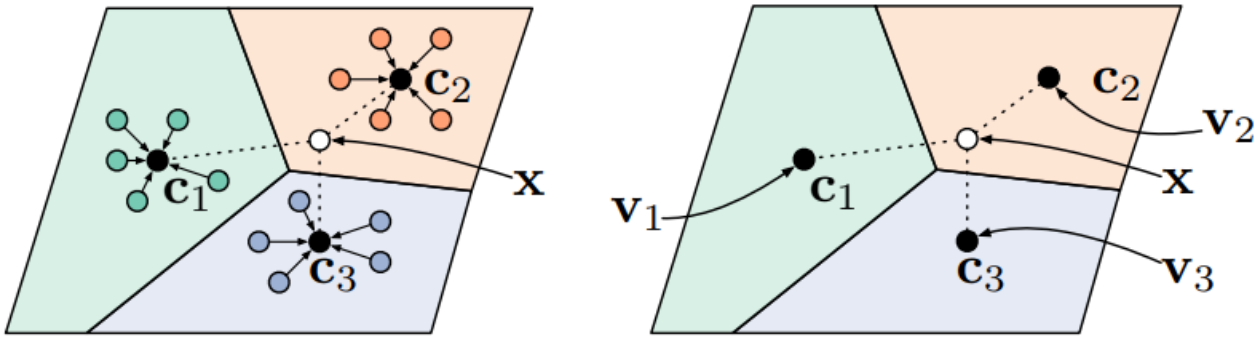
2. Prototypical Networks for Few-shot Learning

논문 : https://arxiv.org/pdf/1703.05175.pdf

요약

- feature representation space에 각 범주별 데이터의 feature vector mean에 모이도록 학습하는 Prototypical Network 제안
- Loss로 사용할 유사도 계산은 유클리드 거리가 코사인 유사도보다 좋은것을 경험적으로 알게됨



(a) Few-shot (b) Zero-shot

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

모델

모델 요약

- epoch 마다 각 범주별 학습데이터를 랜덤샘플링 후 모델을 통해 범주별 feature vector mean을 산출 후 softmax로 가장 가까운 범주를 정답으로 선택
- Loss Function : negative log-probability 최소화
- Optimizer : SGD

데이터/연산 형태 (알고리즘 1로 정리)

- 평범한 데이터 셋 X, Y pair 구조

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

- X는 이미지를 고려해서 D차원에 속함

$$\mathbf{x}_i \in \mathbb{R}^D$$

- 모델 f에서 M차원의 feature vector 연산

$$f_{\phi} : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

- 범주 별 중앙 점 c는 학습 데이터에서 랜덤 샘플링된 x들의 feature vector mean을 취함

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

$$\mathbf{c}_k \in \mathbb{R}^M$$

- feature vector간 거리 계산 함수 d.

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow [0, +\infty).$$

- 어떤 데이터가 입력됐을 때 가장 가까운 범주 중앙 점과의 거리 차이를 바탕으로 softmax 적용

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

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
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\phi}(\mathbf{x}_i)$ ▷ 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**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_{\phi}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_k)) \right]$ ▷ Update loss
  end for
end for

논문엔 feature vector mean을 클러스터의 중앙점으로 사용하는 것이 선형 모델이지만 비선형 정보를 학습할 수 있다는 것

(ImageNet Classification with Deep Convolutional Neural Networks , Going deeper with convolutions)

Euclidean distance가 Mixture Density Estimation을 만족하는 것을 증명하는 수식이 들어가 있음

(Clustering with Bregman Divergences)

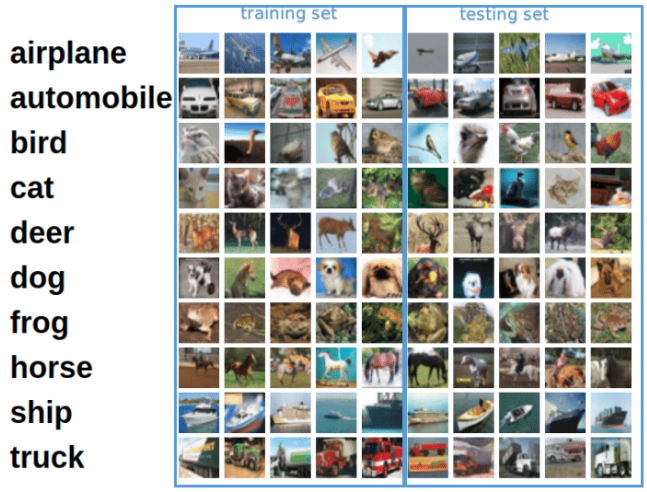
고려할 수 있는 것들

- Distance Metric
  - Euclidean Distance 대신 Cosine Distance를 사용해도 되나, mixture density estimation을 만족하지 않고 실험해보니 더 안좋아지더라
- Episode composition
  - M-way N-shot 방식(에피소드 별로 학습과 테스트셋이 분류되어 M개의 범주를 범주별 N개의 데이터로 학습/검증하는 방식)
  - 저자들이 경험적으로 알게 된것
    - 학습 시간에 테스트보다 더 큰 M 을 사용하는게 좋음
    - 학습 시간과 테스트에는 동일한 N을 사용하는게 좋음
- Zero-shot Learning
  - one-shot learning/few-shot learning과 다르게 메타정보가 추가되는데 메타정보를 feature vector로 연산하는 별도의 모델을 만들어서 처리하면 few-shot learning 때와 동일하게 학습/운용 할 수 있다고 함

실험

- 실험 방법
  - 기존의 제안 기법과 성능 비교 (결과 표 1)
    - Omniglot 데이터셋 사용
    - Matching Network 에서 제안한 학습기법과 논문이 제안한 학습기법을 비교하기위해 모델 구조 / 에피소드 구성 사용
  - 다른 제안 기법과 성능 비교 + 고려사항에 대한 성능 향상 확인 (결과 표 2, 그림 2)

- miniImageNet 데이터셋 사용 (input size 84x84의 100 class * 600 sample)



  - 에피소드 구성을 사용하되 학습 셋이 테스트 셋보다 더 많은 범주를 학습
    - 테스트 셋 : 5-way 5-shot (5개의 범주를 범주 별 5개의 이미지로 학습) / 5-way 1-shot
    - 학습 셋 : 5-way와 20-way를 둘다 실험
  - cosine distance와 euclidean distance를 비교 실험
- 다른 기법과 zero-shot learning 성능 비교 (표 3)
  - CUB-200 데이터셋 사용 (200 종류의 11,788 개의 새 사진)

Table 1: Few-shot classification accuracies on Omniglot.

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot	20-way Acc. 1-shot	20-way Acc. 5-shot
MATCHING NETWORKS [29]	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETWORKS [29]	Cosine	Y	97.9%	98.7%	93.5%	98.7%
NEURAL STATISTICIAN [6]	-	N	98.1%	99.5%	93.2%	98.1%
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	**98.8%**	**99.7%**	**96.0%**	**98.9%**

Table 2: Few-shot classification accuracies on *mini*ImageNet. All accuracy results are averaged ove 600 test episodes and are reported with 95% confidence intervals. *Results reported by [22].

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot
BASELINE NEAREST NEIGHBORS*	Cosine	N	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
MATCHING NETWORKS [29]*	Cosine	N	$43.40 \pm 0.78\%$	$51.09 \pm 0.71\%$
MATCHING NETWORKS FCE [29]*	Cosine	N	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
META-LEARNER LSTM [22]*	-	N	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	$\mathbf{49.42 \pm 0.78\%}$	$\mathbf{68.20 \pm 0.66\%}$

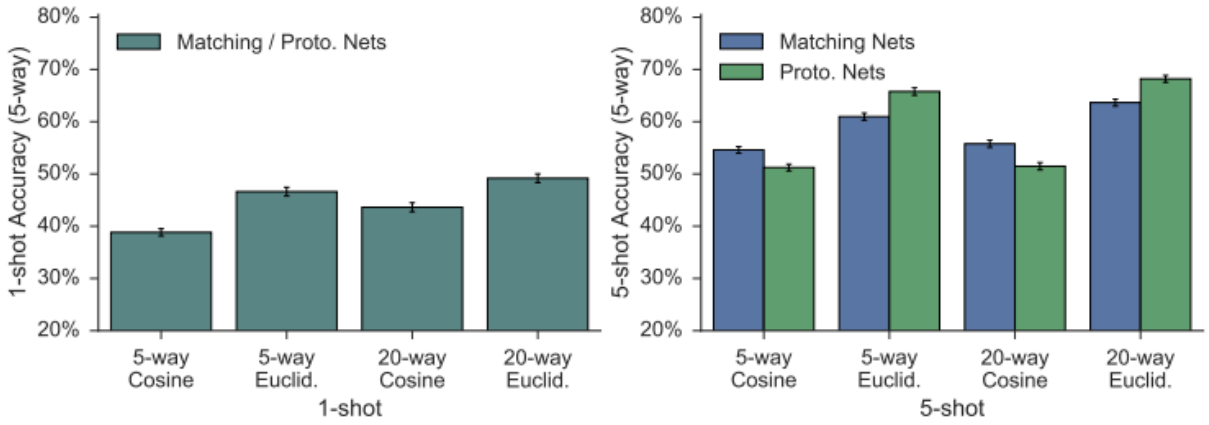


Table 3: Zero-shot classification accuracies on CUB-200.

Model	Image Features	50-way Acc. 0-shot
ALE [1]	Fisher	26.9%
SJE [2]	AlexNet	40.3%
SAMPLE CLUSTERING [17]	AlexNet	44.3%
SJE [2]	GoogLeNet	50.1%
DS-SJE [23]	GoogLeNet	50.4%
DA-SJE [23]	GoogLeNet	50.9%
PROTO. NETS (OURS)	GoogLeNet	**54.6%**