4. Matching Networks for One Shot Learning

논문 : https://arxiv.org/pdf/1606.04080.pdf

요약

- 거리기반 함수를 이용해 support data와 batch data(= query data)간의 유사성을 판단하는 one-shot learning
- Meta Learning에 Episodic training을 사용하여 일반화를 잘 할 수 있도록 학습하는 Meta learner를 만듦
- 모델 구조/손실 계산은 constraive learning과 비슷

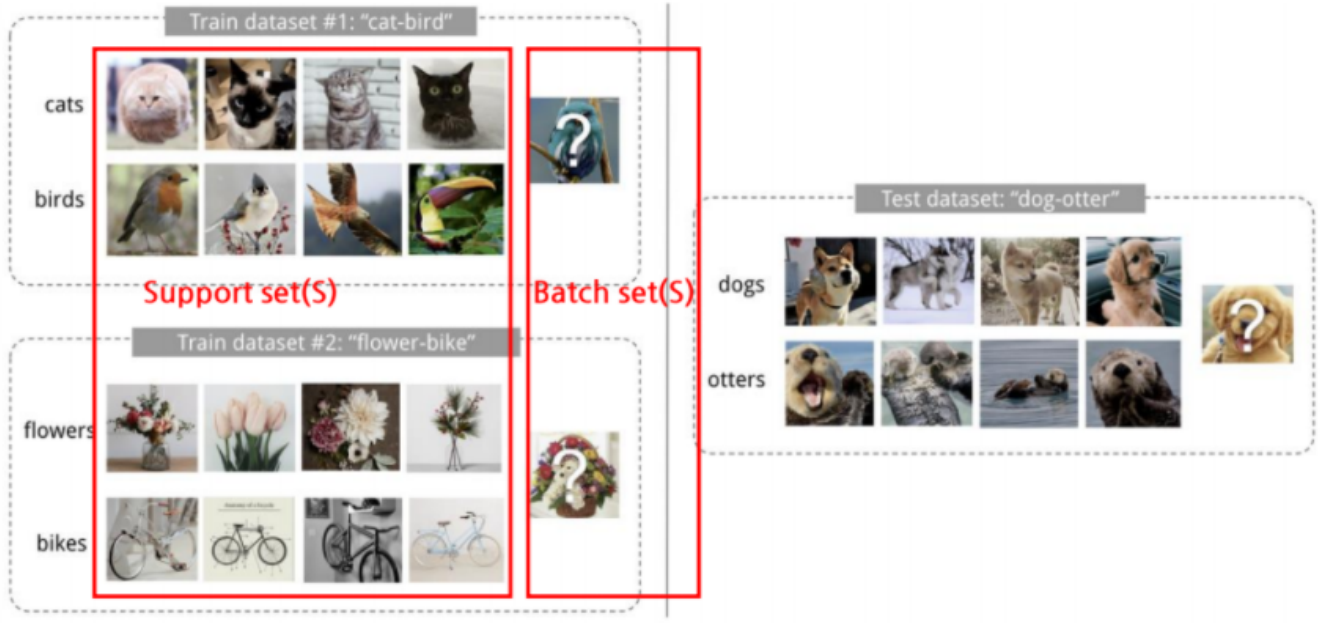
풀고자하는 문제

- M-way N-shot learning
  - M개의 class와 N개의 support data가 주어졌을 때 batch data(=query data)가 M개의 범주중 어디에 속하는가를 학습하는 문제
    - ex) 5개의 class가 주어지고 support data가 1개씩 주어지면, 5-way 1-shot learning
  - 일반적으로 사용되는 M과 N 값
    - M : 5 ~ 25
    - N : 1 ~ 5
  - 일반적인 문제해결 방식
    - Model-based
      - 학습한 Support data를 메모리에 저장한 뒤 batch data와 함께 모델 연산에 사용해 모델이 정답을 예측하는 기법
    - Metric-based (현 시점까지 리뷰한 논문들이 여기에 해당됨)
      - M*N개의 support data의 feature vector와 batch data를 거리계산 함수(코사인 유사도, 유클리드 거리)로 가장 근접한 class를 정답으로 선택
      - 모델은 다른 class의 feature vector를 잘 떨어드릴수있게 일반화하도록 학습
    - Optimization-based
      - M*N개의 support data로 모델이 학습을 해도 충분할수 있도록 모델의 파라미터 초기값을 최적화 하는 기법

Meta learning

	Model-based	Metric-based	Optimization-based
Key Idea	RNN; memory	Metric learning	Gradient Descent
How $P_\theta(y\|\mathbf{x})$ is modeled?	$f_\theta(\mathbf{x}, S)$	$\sum_{(\mathbf{x}_i, y_i)\in S} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i$ (*)	$P_{g_\phi(\theta, S^L)}(y\|\mathbf{x})$

- Episodic training
  - M-way N-shot learning의 테스트 데이터와 동일한 학습 데이터 구성을 맞추기 위해 논문이 제안한 기법
  - 데이터 구성
    - Episode 단위로 support data(M*N)와 batch data(B)가 주어짐
    - data는 superviesed learning과 동일하게 (x,y) 구성
  - Loss 산출
    - metric based meta learning의 경우 batch data와 support data간의 거리차를 Loss로 사용



모델 및 학습

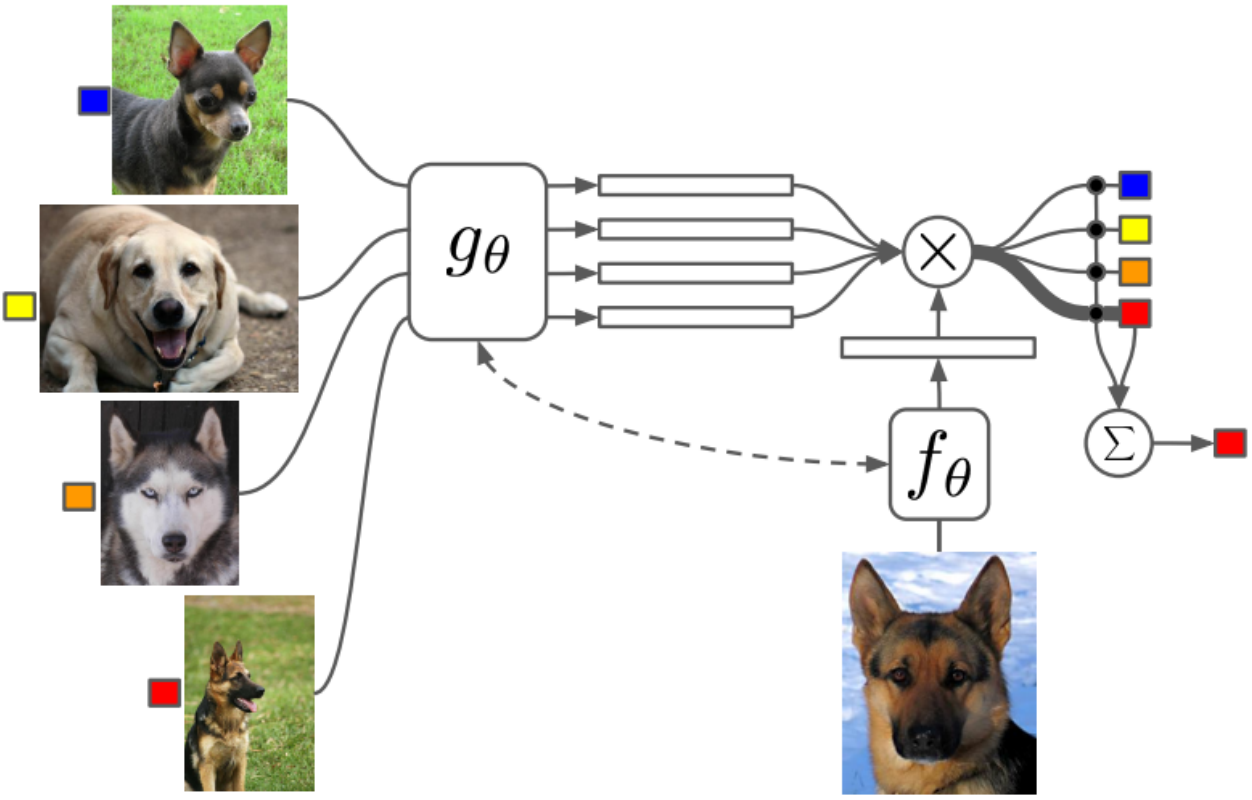


Figure 1: Matching Networks architecture

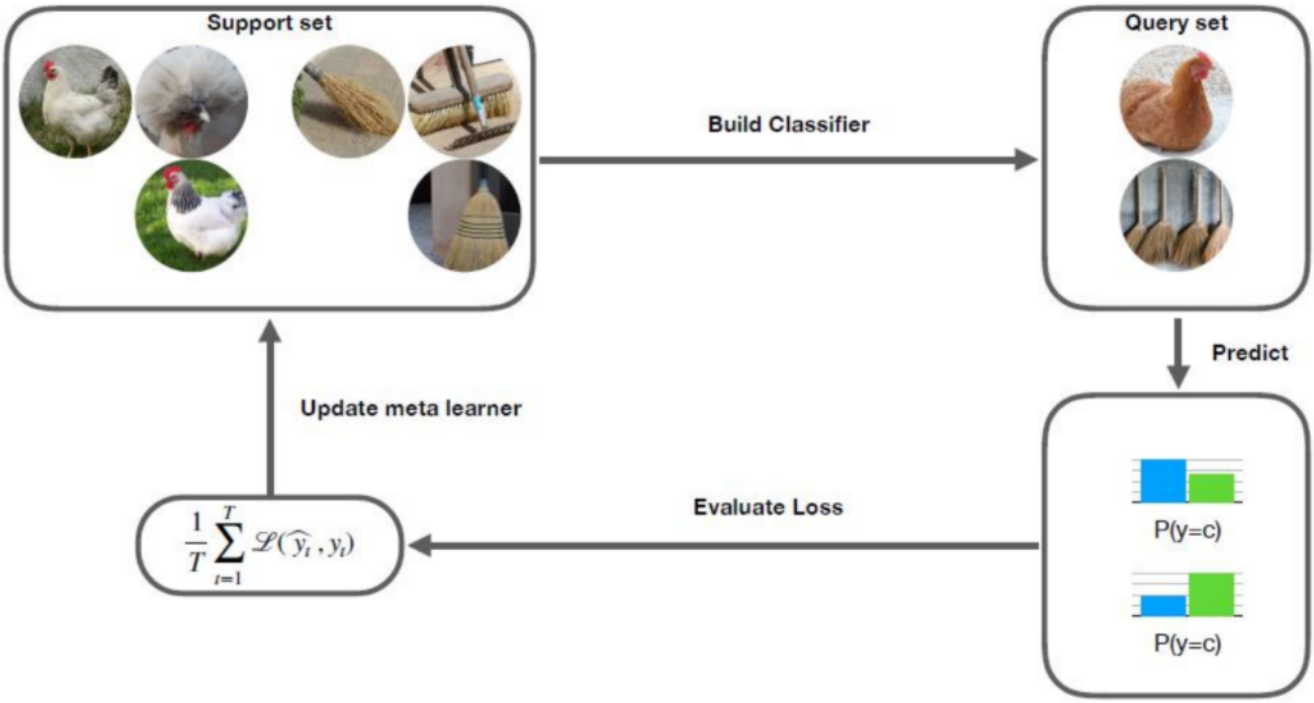
- 모델 (그림 1)
  - g, f 함수 : Task에 따라 supervised learning에서 사용하는 일반적인 모델
    - 이미지 분류 : EfficientNet, ResNet
    - 자연어 분류 : BERT
  - g와 f는 파라미터 공유를 해도되고 따로 학습시켜도됨
    - 논문에서는 모델 파라미터 공유
  - 모델은 attention machanism a를 사용해서 거리차이를 계산.

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

  - 논문은 attention kernel로 cosine similarity를 사용함 (LSTM, BERT의 Attention과 다름)

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

Episode training (Meta strategy)

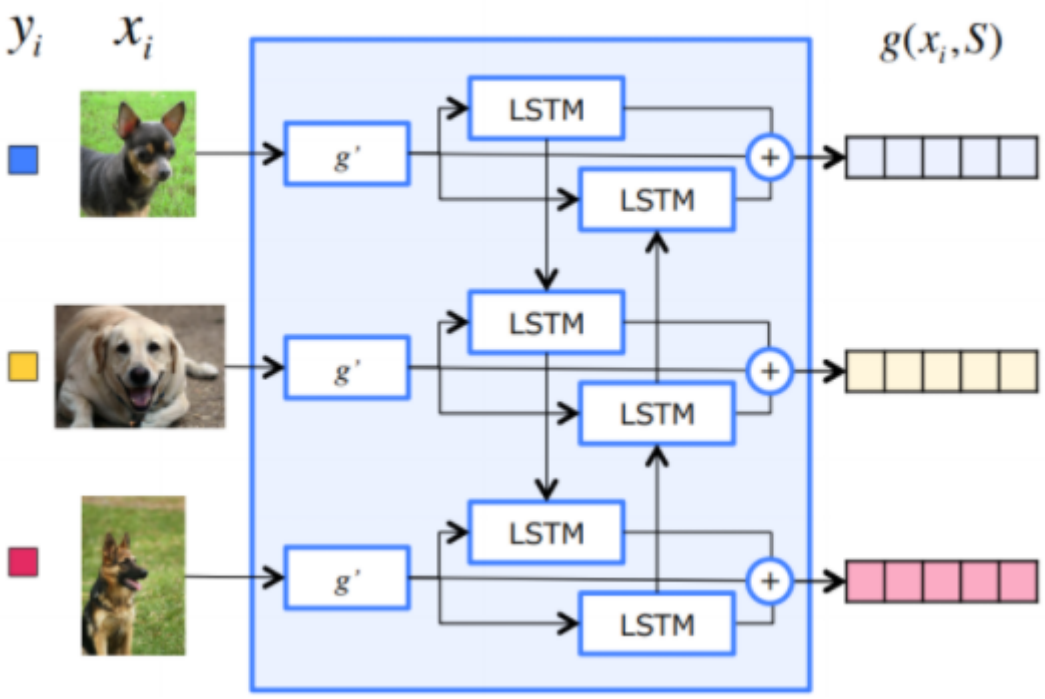


Meta-learning = Learning to learn

$$\theta = \arg\max_{\theta} E_{L\sim T}\left[E_{S\sim L, B\sim L}\left[\sum_{(x,y)\in B}\log P_\theta\left(y|x,S\right)\right]\right]$$

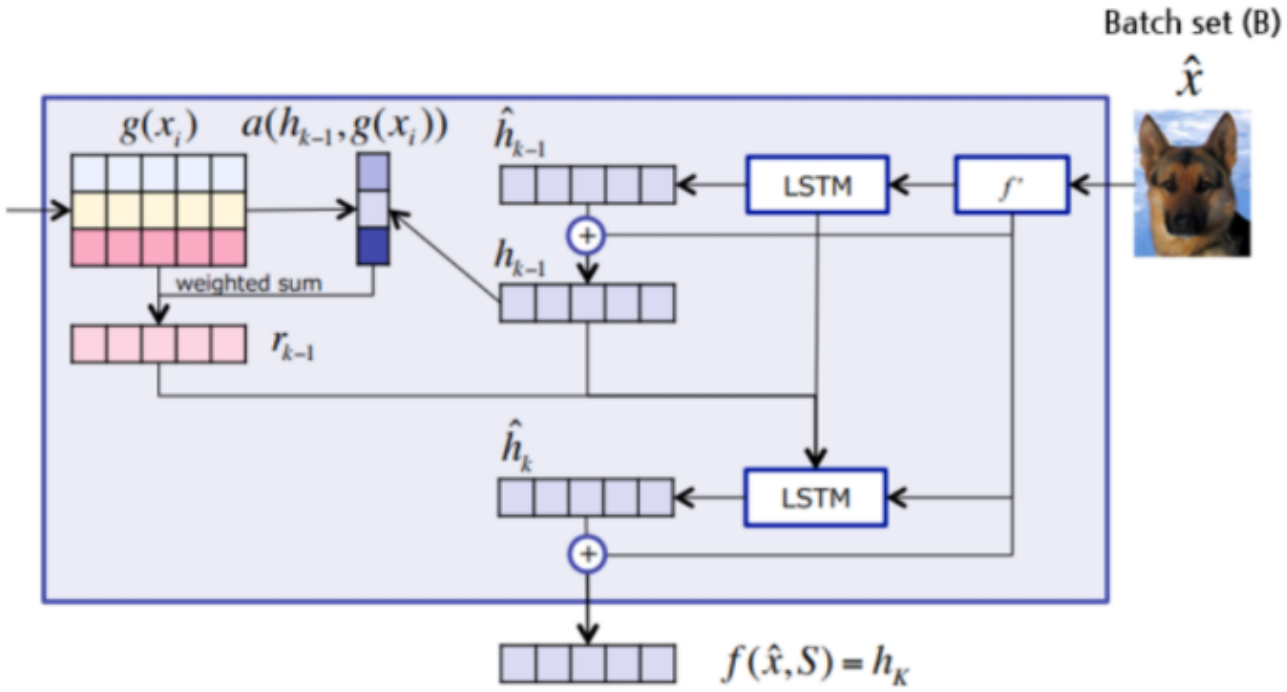
- 학습 (위 그림과 식)
  1. episode 마다 전체 데이터 T 에서 data L를 random sampling
     - random sampling은 class 기준으로 샘플링
  2. data L을 support data S와 batch data B로 분리
     - 뽑힌 class의 data에 대해서 random sampling
  3. 모델에 S와 B를 입력으로 B의 정답을 예측
  4. 예측값을 토대로 손실 계산 및 모델의 파라미터 업데이트
  5. 지정한 episode 만큼 1~4 반복

Support set (S)



$$\vec{h}_i, \vec{c}_i = \text{LSTM}(g'(x_i), \vec{h}_{i-1}, \vec{c}_{i-1})$$
$$\overleftarrow{h}_i, \overleftarrow{c}_i = \text{LSTM}(g'(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1})$$

$$g(x_i, S) = \vec{h}_i + \overleftarrow{h}_i + g'(x_i)$$



$$
\begin{aligned}
\hat{h}_k, c_k &= \text{LSTM}(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1}) \\
h_k &= \hat{h}_k + f'(\hat{x}) \\
r_{k-1} &= \sum_{i=1}^{|S|} a(h_{k-1}, g(x_i)) g(x_i) \\
a(h_{k-1}, g(x_i)) &= \text{softmax}(h_{k-1}^T g(x_i))
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
f(\hat{x}, S) &= \text{attLSTM}(f'(\hat{x}), g(S), K) \\
&= h_K
\end{aligned}
$$

Full Context Embedding

- Support data가 2개 이상인 경우((N>=2) - shot learning)와 같은 복잡한 문제에 적용할 수 있는 제안 기법
- 같은 class끼리 비슷한 결과값이 나올 수 있도록 bidirection-LSTM 결과 값을 support data의 feature vector로 사용
- batch data의 feature vector와 support data의 feature vector로 Attention machanism이 적용된 LSTM로 연산한 뒤 거리계산함수를 적용하는 방안을 제안

실험

- 데이터셋에 전체 class을 학습/테스트로 분리
  - 모델은 테스트 셋의 class를 학습하지 않음
- 실험 결과에 fine-tuning은 최적의 성능을 낼 수있도록 하이퍼퍼라미터를 튜닝한 것이라고 명시되있으나, 상세 튜닝값은 공개하지 않음
- 실험 종류
  - Ominiglot 데이터셋으로 기존 기법과 성능 비교 (표 1)
  - miniImageNet 데이터셋으로 기존 기법과 성능 비교 + Full Context Embedding이 의미있는가? (표 2, 3)
  - Penn Treebank (NLP 데이터)로 제안 기법 성능 확인 (표 4)

Model	Matching Fn	Fine Tune	5-way Acc		20-way Acc	
			1-shot	5-shot	1-shot	5-shot
PIXELS	Cosine	N	41.7%	63.2%	26.7%	42.6%
BASELINE CLASSIFIER	Cosine	N	80.0%	95.0%	69.5%	89.1%
BASELINE CLASSIFIER	Cosine	Y	82.3%	98.4%	70.6%	92.0%
BASELINE CLASSIFIER	Softmax	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	Cosine	N	82.8%	94.9%	–	–
CONVOLUTIONAL SIAMESE NET [11]	Cosine	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Cosine	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	Cosine	N	**98.1%**	**98.9%**	**93.8%**	98.5%
MATCHING NETS (OURS)	Cosine	Y	97.9%	98.7%	93.5%	**98.7%**

Table 1: Results on the Omniglot dataset.

Table 2: Results on *mini*ImageNet.

Model	Matching Fn	Fine Tune	5-way Acc	
			1-shot	5-shot
PIXELS	Cosine	N	23.0%	26.6%
BASELINE CLASSIFIER	Cosine	N	36.6%	46.0%
BASELINE CLASSIFIER	Cosine	Y	36.2%	52.2%
BASELINE CLASSIFIER	Softmax	Y	38.4%	51.2%
MATCHING NETS (OURS)	Cosine	N	41.2%	56.2%
MATCHING NETS (OURS)	Cosine	Y	42.4%	58.0%
MATCHING NETS (OURS)	Cosine (FCE)	N	44.2%	57.0%
MATCHING NETS (OURS)	Cosine (FCE)	Y	**46.6%**	**60.0%**

Table 3: Results on full ImageNet on *rand* and *dogs* one-shot tasks. Note that $\neq L_{rand}$ and $\neq L_{dogs}$ are sets of classes which are seen during training, but are provided for completeness.

Model	Matching Fn	Fine Tune	ImageNet 5-way 1-shot Acc			
			L_{rand}	$\neq L_{rand}$	L_{dogs}	$\neq L_{dogs}$
PIXELS	Cosine	N	42.0%	42.8%	41.4%	43.0%
INCEPTION CLASSIFIER	Cosine	N	87.6%	92.6%	**59.8%**	90.0%
MATCHING NETS (OURS)	Cosine (FCE)	N	**93.2%**	**97.0%**	58.8%	**96.4%**
INCEPTION ORACLE	Softmax (Full)	Y (Full)	$\approx 99\%$	$\approx 99\%$	$\approx 99\%$	$\approx 99\%$

Model	5 way accuracy		
	1-shot	2-shot	3-shot
Matching Nets	32.4%	36.1%	38.2%
Oracle LSTM-LM	(72.8%)	-	-