[논문 리뷰] iCaRL: Incremental Classifier and Representation Learning

diddu · 어제

논문

🔎 논문 리뷰

💡 iCaRL: Incremental Classifier and Representation Learning_논문 리뷰

🤸 cf) Knowledge Distillation 이란?

논문을 읽다가 distillation 개념에 대해 짚고 넘어가야할 것 같아서 찾아보았다.

> ***Knowledge Distillation(지식 증류)***란 쉽게 말해 학습된 모델로부터 지식을 추출하는 것이다.

Knowledge Distillation은 우리가 정확도를 최고로 올리기 위해 복잡하고 큰 모델을 만들었다 하더라도 실제 사용자가 쓰는 서비스에 적용하기엔 너무 무겁고 느려 적합하지 않기 때문에

등장했다.

ex) 3시간이 걸려 99%의 정확도를 내는 복잡한 T 모델 vs 3분이 걸려 90%의 정확도를 내는 단순한 S 모델

그렇다면 T와 S를 잘 활용하는 방법이 있지 않을까?
→ **"Knowledge Distillation"** : 복잡한 모델이 학습한 generaliztion 능력을 모델 S에 전달해주는 것을 말함

Distillation Loss

Knowledge Distillation을 이해하기 위해 Soft Label과 Distillation Loss에 대해 알아보자

> **Soft Label** (Soft-max Output, Dark knowledge)

일반적으로 이미지 클래스 분류 task에선 마지막 softmax layer로 클래스의 확률값을 출력한다. 10^{-6} 0.9 0.1 ... 이렇게 출력되는 값들이 모델의 지식이라고 볼 수 있다. 하지만 이 상태로는 값이 너무 작아서 반영하기 힘들기 때문에 아래 수식을 통해 값을 좀더 soft하게 변경한다.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

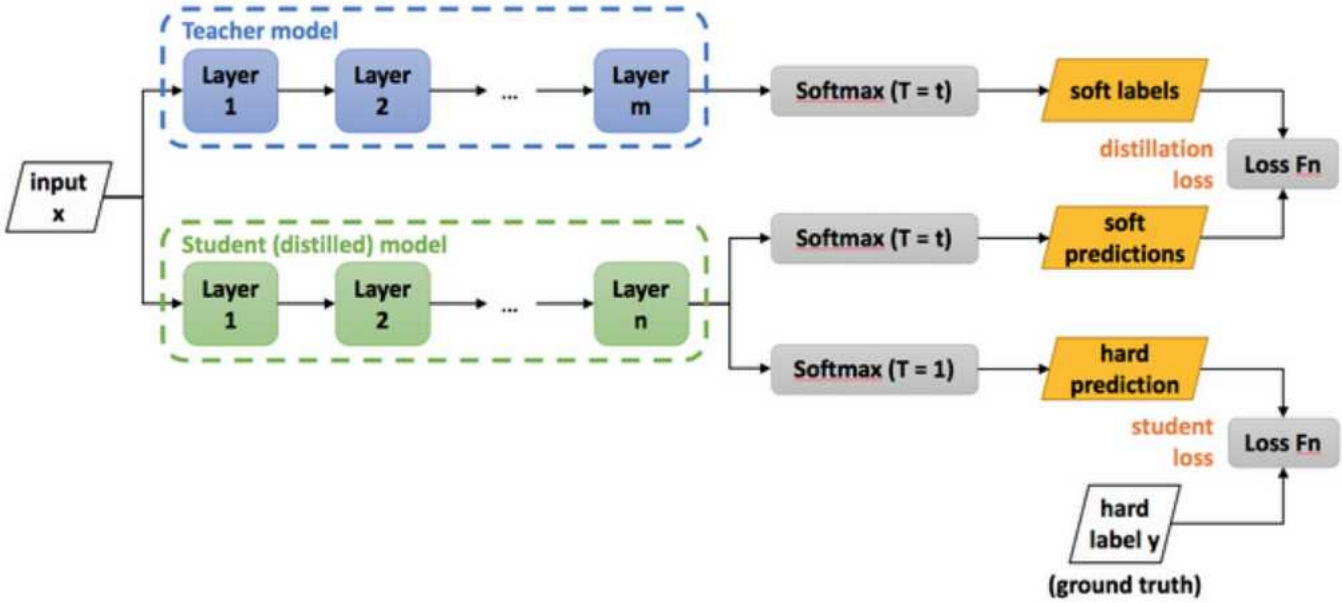
T값은 Temperature(온도)라고 하고 T값이 커지면 더 soft하게, T값이 작아지면 더 hard하게 나타낼 수 있다. 이렇게 나타낸 Soft Label은 복잡한 모델 T의 지식이라고 볼 수 있다. 이것을 S에게 넘기는 방법인 Distillation Loss에 대해 알아보자.

> **Distillation Loss**

$$L = \sum_{(x,y)\in\mathbb{D}} L_{KD}(S(x,\theta_S,\tau), T(x,\theta_T,\tau)) + \lambda L_{CE}(\hat{y}_S, y)$$

손실함수는 Distillation Loss와 Student Loss로 나눌 수 있다.

- L_{KD} = **Distillation Loss** : T의 soft labels와 S의 soft predictions를 비교하여 손실함수를 구성
- L_{CE} = **Student Loss** : 기존 이미지 분류에서 사용되는 Cross Entropy Loss



📌 Incremental Learning의 필요성

자율주행 자동차를 예로 들면, Image Detection을 할 때 필연적으로 학습 dataset에 포함되지 않은 다른 클래스의 물체가 등장할 것이고 이것을 추가로 학습해야한다. 이 때, 새로운 데이터를 포함한 dataset을 제작해서 처음부터 다시 학습하는 것은 비효율적이다.
따라서, 기존의 학습 내용은 유지하면서 새로운 정보만 추가로 학습하는 방법이 필요하다.

Incremental Learning에도 다양한 방법론들이 존재하는데 해당 논문은 iCaRL은 `Distillation + Memory` 방법으로 이전 task의 데이터를 소량 메모리로 두고 새로운 task를 학습할 때 사용하는 방법이다.
(이전 task 데이터의 일부 : Examplar)

📌 Introduction

해당 논문은 아래 세 가지를 만족하는 방법론을 제시한다.

1. **trainable** from a stream of data in which examples of **different classes occur at different times**
2. **should at any time provide a competitive multi-class classifier for the classes** observed so far
3. its **computational requirements and memory footprint should remain bounded**, or at least grow very slowly

> **** iCaRL*** : a practical strategy for simultaneously learning classifiers and a feature representation in the class-incremental setting
>
> - ***nearest-mean-of-exemplars rule***을 사용한 classificaiton
> - herding에 기반한 ***prioritized exemplar selection***
> - ***knowledge distillation***과 ***prototype rehearsal***을 사용한 representation learning

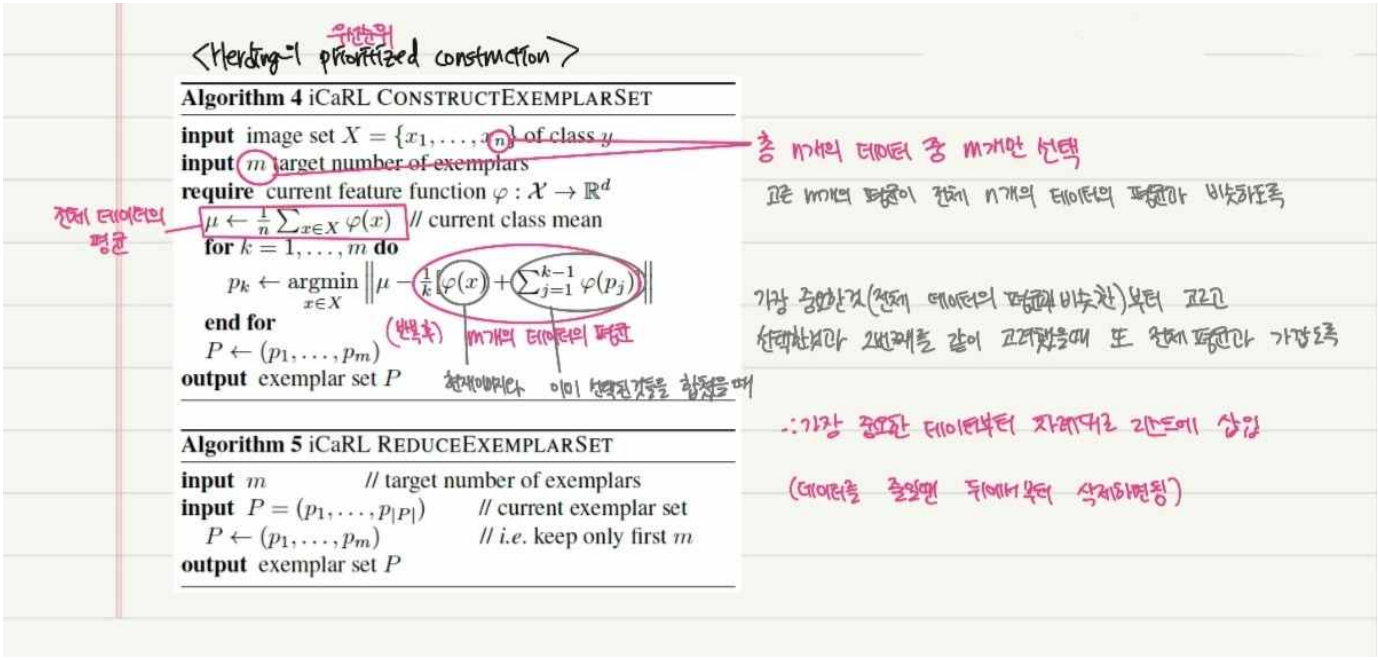
🎈 Examplar Management

기존에는 new task를 학습할 때, old class data 사용 X
→ 기존 지식을 잃어버리는 현상을 방지하기 위해 기존 data를 약간씩 sampling하여 examplar라는 미니 데이터셋을 구축
→ 활용

`"Examplar"`

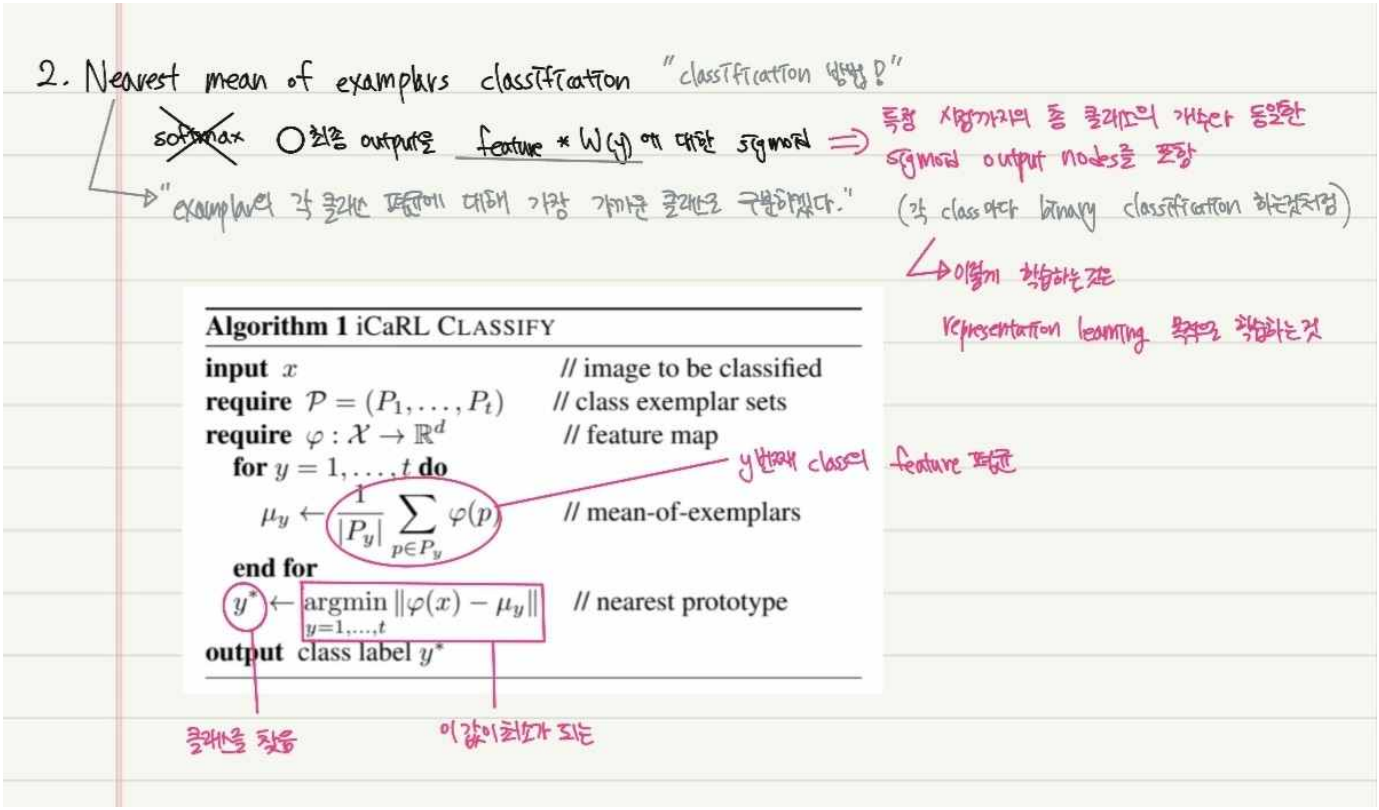
1. **모든 class의 data 개수를 동일하게**
   ∵ class imbalance 현상
2. **해당 class를 가장 잘 대표하는 데이터들이어야함**
   ∵ 적은 데이터만으로 학습해야하기 때문
3. **각 calss별로 해당 class를 잘 대표하는 순서대로 구성되어야함**
   ∵ 기존 class의 data를 메모리 유지를 위해 빼야하는데, 이때 중요도 순서대로

정렬돼있으면 뒤에서부터 제거하면 되기 때문



🎈 Nearest mean of examplars classification

일반적으로 이미지 분류를 진행할 때 softmax를 사용하지만 해당 논문에선 **Nearest-mean-of-examplars**를 사용한다. 즉, examplar의 각 클래스 평균에 대해 가장 가까운 클래스로 구분하겠다는 의미이다.



🎈 Representation Learning

1. 새로운 데이터와 기존 데이터를 합쳐 전체 dataset 구성
2. 전체 dataset에 대해 분류 결과를 임시로 저장
   (아직 학습 전이기 때문에 새로운 class로는 분류되지 않지만 새로운 데이터까지 모든 데이터를 입력으로 넣음 : distillation을 구하기 위해)

3. Representation learning

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
  $q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \left[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) \right.$$
$$\left. + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y) \log(1-g_y(x_i)) \right]$$

that consists of *classification* and *distillation* terms.

→ 학습에 필요한 dataset 구성

전체 dataset에 대해 분류 결과를 임시로 저장
이때 새로운 class로는 분류가 되지 X
하지만, 새로운 데이터까지 모든 데이터를 입력으로 넣음
why?! : distillation을 구하기 위해
(기존 네트워크가 학습한 지식을 유지, 잊지 않도록 하기 위함)

새로운: classification loss
기존: distillation loss
정답인 경우
정답이 아닌 경우
binary cross entropy loss

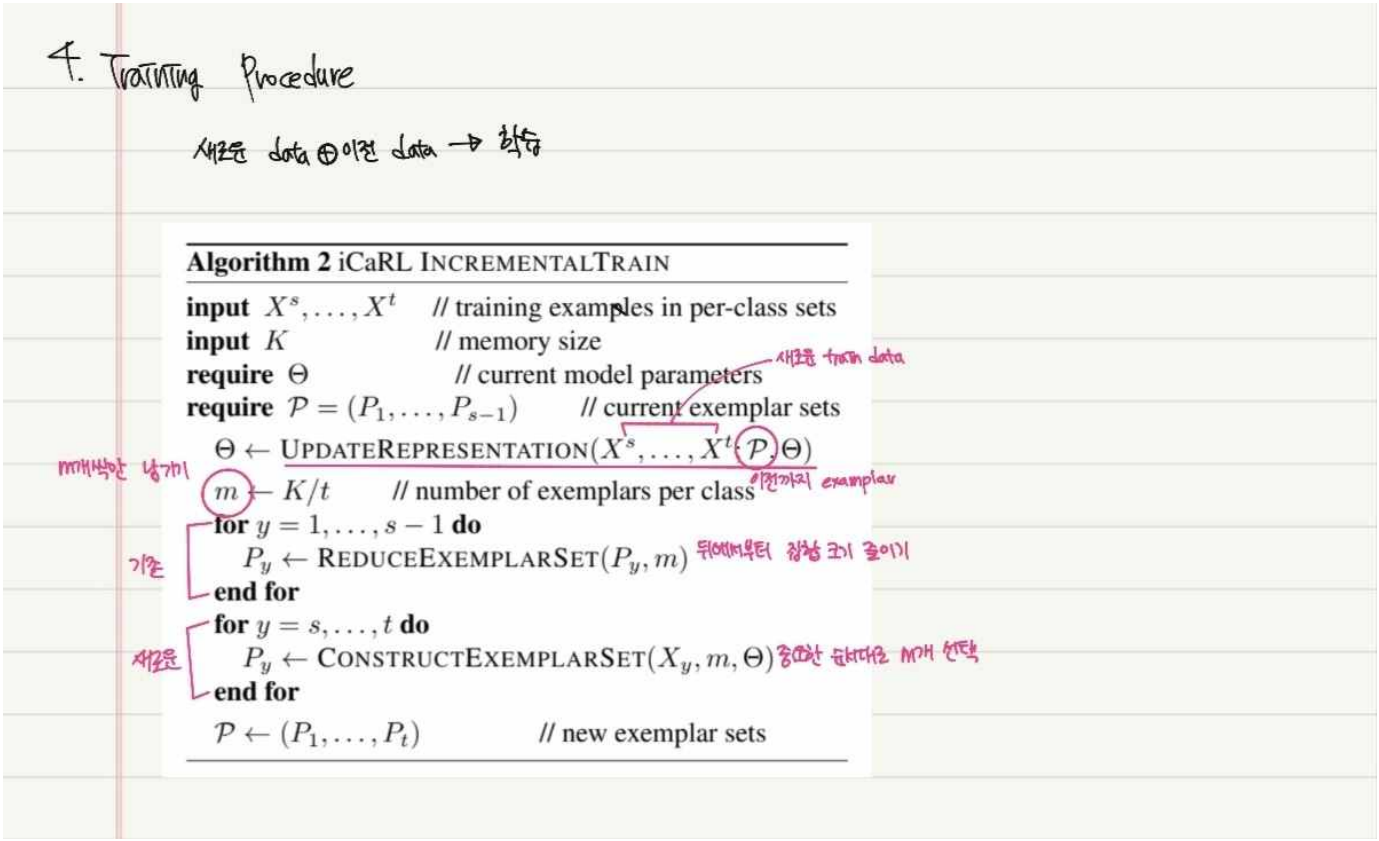
기존 네트워크가 알고있던 지식도 잊지 않기 위해
기존 네트워크의 출력 결과를 반영할 수 있도록 "distillation loss"

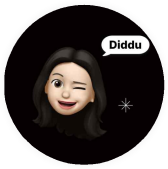
* 각 class를 개별적으로 binary classification하도록 학습
  - 네트워크의 parameter θ
  - feature extractor $\varphi: X \to R^d$
  - 각 class에 대한 분류결과 $g_y(x) = \frac{1}{(1+\exp(-w_y^T \varphi(x))}$

각각의 class마다 d-dimension의 w_y (weight vector)
얘를 학습시키는 것

실제 분류할땐 nearest-mean 으로 하긴
: $g_y(x)$를 사용하진 않음.
얘는 representation learning에 사용

🎈 Training Procedure





Diddu

다음 포스트
[논문 리뷰] PaLM-E: An Embodied Multimodal Language Model

이전 포스트
[논문 리뷰] End-to-End Multi-Task Learning with Attention

0개의 댓글

댓글을 작성하세요

댓글 작성