
# 딥러닝 학습 방법

## AI – 스쿨

2022. 06. 15.

# I

# 딥러닝 학습 방법

1. 모델 성능 향상 테크닉
2. MixlUp Training
3. Label smoothing
4. Model ensemble

## 모델 성능 향상 테크닉

(1) 딥 러닝 신뢰도 개선을 위한 모델 보정 기법

- 최근 딥 러닝은 다양한 산업 분야에서 사용되며 여러 성과를 보이고 있으나, 실제 어플리케이션에서 사용되기에는 신뢰성이 다소 부족함

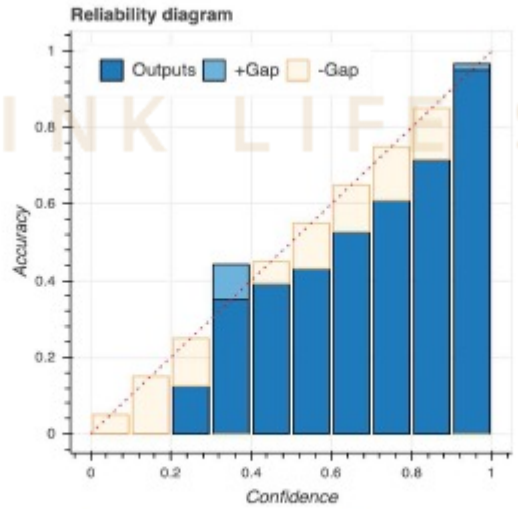
- 일반적인 딥 러닝 모델이 도출하는 정답에 대한 정확도(confidence score)는 실제 적용 환경에서 평가하는 정확도보다 높게 평가되는 경향 존재

- 자율주행, 의료, 금융 등 의사 결정이 건강과 경제에 큰 영향을 미치는 분야에서, 자체 평가 정확도는 높으나 실제로는 틀린 판단(High-confidence errors)이 발생하는 경우 치명적인 문제를 야기할 수 있음

## 모델 성능 향상 테크닉

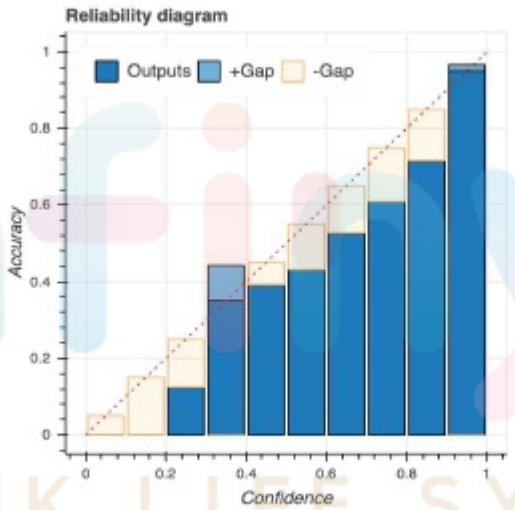
- 모델 보정(Model calibration)을 통해 딥 러닝 모델이 예측한 정답일 확률(confidence score)와 실제로 정답일 확률(accuracy)가 일치하도록 조정함으로써 모델의 신뢰도 문제를 개선할 수 있음

- 예측 확률과 정확도 간 관계를 나타내는 신뢰도 다이어그램을 이용하여 파악 가능




10개 구간(Bin)의 신뢰도 다이어그램(Reliability diagram)

## 모델 성능 향상 테크닉




10개 구간(Bin)의 신뢰도 다이어그램(Reliability diagram)

- 붉은 점선: 예측 확률과 정확도가 일치하는 상태(완전 보정, Perfect calibration)
- 붉은 점선보다 파란 막대가 작음: 예측 확률 > 정확도(과대확신, over-confidence)
- 붉은 점선이 파란 막대보다 작음: 예측 확률 < 정확도(과소확신, under-confidence)

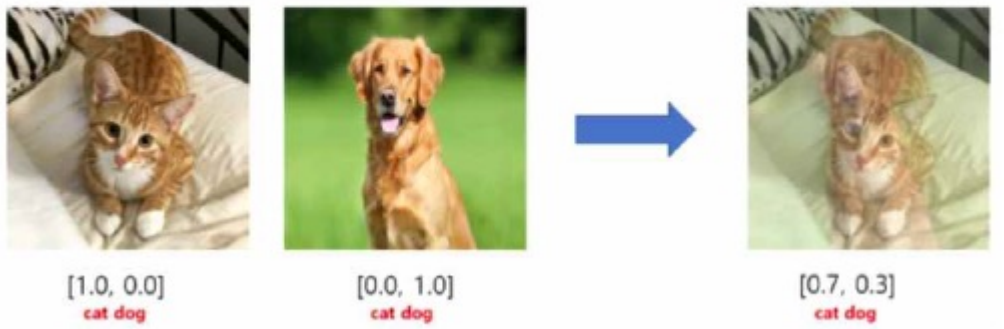
※ 이를 기반으로 분석하면 위 그래프는 '예측 확률이 0.3~0.4, 0.9~1.0인 구간을 제외하고는 과잉확신 양상이 나타나는 모델의 그래프'라고 해석할 수 있음

## MixUp Traing

- Data augmentation 기법의 일종으로 두 데이터와 라벨을 일정 비율로 섞어 새로운 데이터를 생성하는 방법론
- 학습 진행 시 랜덤하게 두 개 샘플(□▫□ □▫와 □▫□ □▫)을 골라 사용(□□□□□ )

$$\begin{aligned} \hat{x} &= \lambda x_i + (1-\lambda)x_j, \\ \hat{y} &= \lambda y_i + (1-\lambda)y_j, \end{aligned}$$

where $\lambda \in [0, 1]$ is a random number




[Mixup Augmentation 예시]

## MixUp Traing

- ex) 단순 학습 시 복합 패턴 분류 성능(baseline)은 72%, MixUp 시 92% 수준으로 20% 향상됨

| Accuracy | | Alexnet | VGGNet-16 | Resnet-18 |
|---|---|---|---|---|
| 단순 패턴 분류 | Baseline | 0.977 | 0.981 | 0.977 |
| | MixUp | 0.970 | 0.974 | 0.971 |
| 복합 패턴 분류 | Baseline | 0.715 | 0.678 | 0.738 |
| | MixUp | 0.872 | 0.964 | 0.928 |

※ 단순 패턴 분류 성능은 거의 유지, 복합 패턴 분류 성능은 대폭 향상

※ 20.11 대한산업공학회 출처 자료

## MixUp Traing

```
import numpy as np
mixupAlpha = 1.0 # MixUp 정도 조절값(alpha)

def mixupFunction(x, y) :
    lam = np.random.beta(mixupAlpha, mixupAlpha)
    batchSize = x.size()[0]
    index = torch.randerm(batchSize).cuda()
    # index = torch.randerm(batchSize).cpu()
    yA, yB = y, y[index]
    return mixedX, yA, yB, lam

def mixupCriterion(criterion, pred, yA, yB, lam) :
    return lam * criterion(perd, yA) + (1 - lam) * criterion(perd, yB)
```

```
# 실제 train 함수에서 사용되는 부분
# 빨간색 부분: 위에서 만든 mixup 함수가 호출됨
def train() :
    net.train()
    trainLoss = 0
    correct = 0
    tatal = 0

    for batch, idx, (inputs, targets) in enumerate(train_loader):
        inputs, tragets = input.to(device), tragets.to(device)
        inputs, tragetsA, targetsB, lam = mixupFunction(input, tragets)
        optimizer.zero_grad()
        output = net(inputs)
        loss = mixupCriterion(criterion, outputs, targets_a, targets_b, lam)
        loss.backward()
        _, predicted = outputs.max(1)

        total += targets.size(0)
        currentCorrect = (lam * predicted.eq(targtsA).sum().item() + (1-lam)
            * predicted.eq(targetsB).sum().item())
        correct += currentCorrect
        ....
    ...
```

## Label smoothing

- 데이터 정규화(Regularization) 기법 중 하나로 모델 일반화 성능 향상에 사용
- one-hot으로 표현되던 class label을 soft target으로 바꾸어 사용함

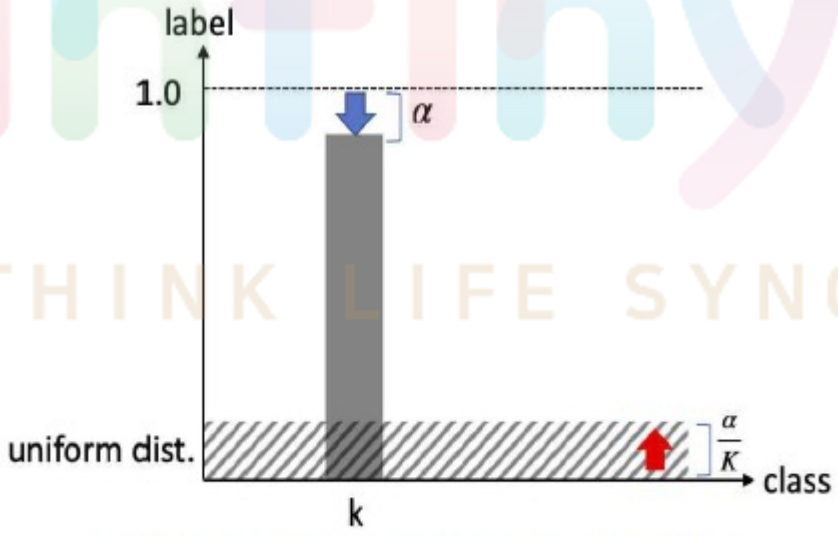
$$y_k^{LS} = y_k(1-\alpha) + \frac{\alpha}{K}$$

- $y_k^{LS}$: 라벨링벡터중 $k$ 번째값
- $\alpha$: hyperparameter

## Label smoothing

- ex) 4개 클래스를 분류하는 문제에서 레이블에 대한 표현형과 label smoothing 수행 예시는 다음과 같음

Hard target: □□□ □□ □□ □□ -> Soft target: □□□□□□□ □□□□□□ □□□□□□ □□□□□□




[라벨 스무딩은 다른 클래스를 균일 분포로 취급]

- 라벨 노이즈(Label noise, 라벨 실수)에 대해서도 라벨 스무딩 기법 적용 가능
- □□□□ 는 다른 클래스를 균일 분포(Uniform dist.)로 만듬으로써 라벨 노이즈를 사전 분포(Prior dist.)로 만듬

## Label smoothing

```
class LabelSmoothingCrossEntropy(nn.Module) :
    def __init__(self):
        super(LabelSmoothingCrossEntropy, self).__init__()
    def forward(self, y, targets, smoothing=0.1):
        confidence = 1. - smoothing
        log_probs = F.log_softmax(y, dim=-1) # 예측 확률 계산
        true_probs = torch.zeros_like(log_probs)
        # torch.zeros_like : 입력한 텐서와 크기를 동일하게 하면서 값을 0으로 채워짐
        true_probs.fill_(smoothing / (y.shape[1] - 1))
        true_probs.scatter_(1, targets.data.unsqueeze(1), confidence)
        # 정답 인덱스의 정답확률 confidence 로 변경
        # unsqueeze : 지정한 dimension 자리에 size가 1 빈공간 채워주면서 차원 확장
        return torch.mean(torch.sum(true_probs * -log_probs, dim=-1))
        # negative log likelihood
```

```
# 사용 예시
criterion = LabelSmoothingCrossEntropy()
```

## Model ensemble

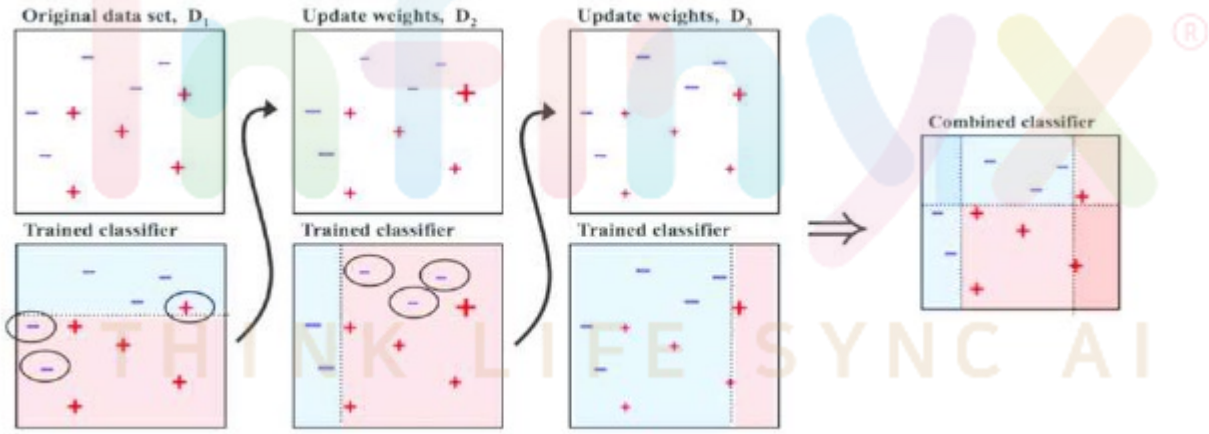
(1) Ensemble이란?
- 여러 모델을 결합하여 모델의 정확도를 향상시키는 기법
- 여러 개의 약 분류기(Weak Classifier)를 결합하여 강 분류기(Strong Classifier) 생성

(2) Ensemble 기법
- Bagging
  : Bootstrap Aggregation의 약자.
  데이터셋으로부터 여러 개의 샘플을 추출(bootstrap)하여 모델들을 학습시키며, 문제에 대한 모델들의 결과를 집계(aggregate)하여 최종 결과 도출

● Categorical Data: 모델 예측값 중 가장 많은 값을 최종값으로 도출(투표 방식)
● Continuous Data: 모델 예측값들의 평균을 최종값으로 도출

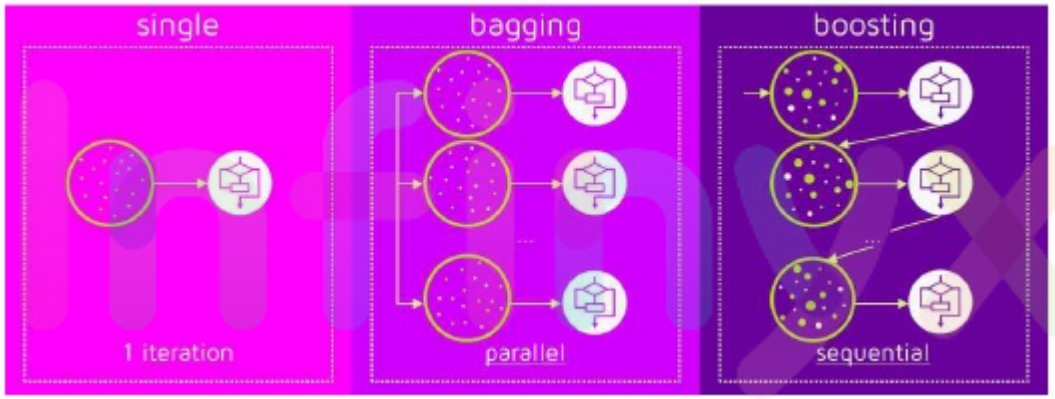
## Boosting

Boosting : 약 학습기들이 순차적으로 결합하여 문제 해결 규칙 형성. 잘못 분류된 데이터가 있을 경우 이에 집중하여 새로운 분류 규칙 만드는 단계를 반복함.




1) + 와 – 를 분류하는 문제에서, D1은 2/5지점을 가로로 나누어 분류함
2) 잘못 분류된 위쪽의 +와 아래쪽의 - 는가중치를 높이고,
   제대로 분류된 데이터는 가중치를 낮춤.
3) 가중치가 조정된 데이터를 분류하는 분류기(D2, D3)를 두는 절차를 반복.
4) D1, D2, D3 Classifier을 결합한 최종 Classifier은 +와 - 를 정확하게 분류

## Bagging 과 Boosting 차이




- 모델 학습 방식

● Bagging: 병렬 학습

● Boosting: 순차 학습. 한 모델의 학습과 결과가 다음 모델에 영향을 미침(가중치를 부여하여 다음 모델이 앞선 모델의 오답에 집중하도록 함)

- 특징

● Bagging: 빠른 학습 속도. 각 모델 학습이 독립적임

● Boosting: 느린 학습 속도. 일반적으로 성능이 좋으나 오버피팅이 발생할 수 있음

## 과제

6월 14일 ~ 6월 15일 실습한 학습 코드에 MixUP 라벨스무딩 적용해보기

1. MixUP 적용 해보기
2. 라벨스무딩 적용 해보기

제출 기간 : 6월 16일 오전 9시까지


감사합니다.
THINK LIFE SYNC AI
Infinyx