딥러닝 학습 방법

AI – 스쿨

2022. 06. 13.

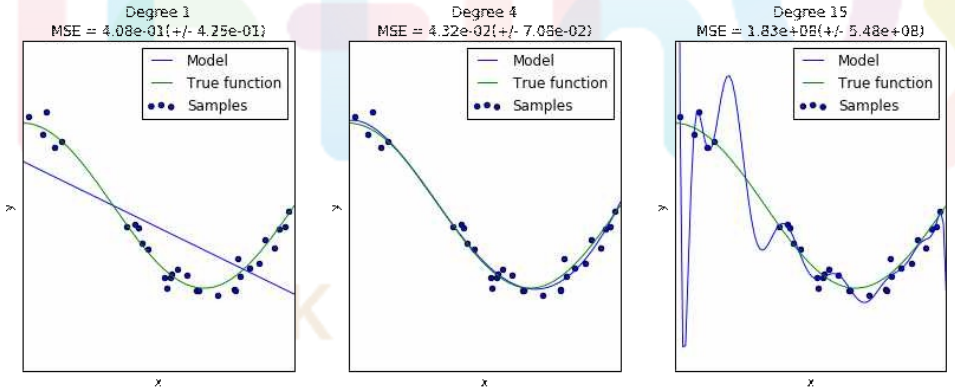
I

딥러닝 학습 방법

1. 6월 2주차 딥러닝 이론 정리 (복습)
2. 딥러닝 모델 선택 방법론
3. 딥러닝 모델 평가 지표

6월 2주차 내용 정리

- 부적합(Underfitting), 과적합(Overfitting)
  - 부적합(Underfitting) : train, test 데이터셋 모두에서 성능이 떨어지는 경우
  - 과적합(Overfitting) : train, valid 데이터셋 성능이 우수하나, test 에서 떨어지는 경우



[부적합, 이상적, 과적합일 때 나타나는 그림]

- 인공지능(파란선)의 목표는 True function과 유사해지는것이며, 이를 위해 Samples의 분포를 학습함
- 부적합일 때 인공지능은 Samples를 맞추는 정확도가 떨어지며, 과적합일 때는 Samples를 너무 잘 반영하여True function과 멀어짐
- True functioin와 유사하지 않을수록, 새로운 데이터에 대한 정확도가 떨어짐

6월 2주차 내용 정리

Gradient Vanishing/Exploding

- 기울기 소실(Gradient vanishing)및 기울기 폭주(Gradient exploding)
- 학습의 역전파 과정에서 미분 계산값이 소실되거나 발산되어 이상 가중치 발생

- 적용 가능한 해결 방안

● 활성함수변경
● 가중치 초기화: 학습 전 가중치 값설정.Xavier/He initialization 등
● 정규화(Normalization):출력값이정규화되도록 반환

6월 2주차 내용 정리

모델 학습 하이퍼파라미터(Hyperparameter)

(1) Loss function
- 모델이 도출한 output과 정답지 target의 차이를 수치화하는 함수
- 회귀분석에는 MSE(Mean Square Error), Classification에는Cross-Entropy

이진 분류문제는 Binary Cross-Entropy를 주로 사용
- loss 계산방식에 따라 Center loss, Triplet loss, Focal loss 등존재
- pytorch에서 제공 하는 기본 Loss function

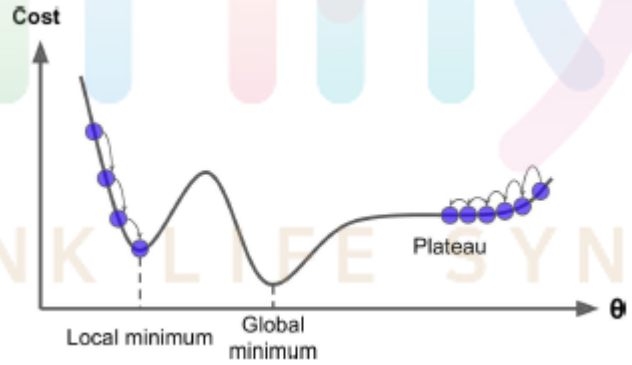
● https://pytorch.org/docs/stable/nn.html#loss-functions

※하이퍼파라미터:사용자가 직접설정하는 값

6월 2주차 내용 정리

모델 학습 하이퍼파라미터(Hyperparameter)

(2) Optimizer
- 가중치(weight)가 구성하는 공간을 탐색하여 모델 loss가 최저가 되는 지점(가중치값)을 찾아내는 알고리즘



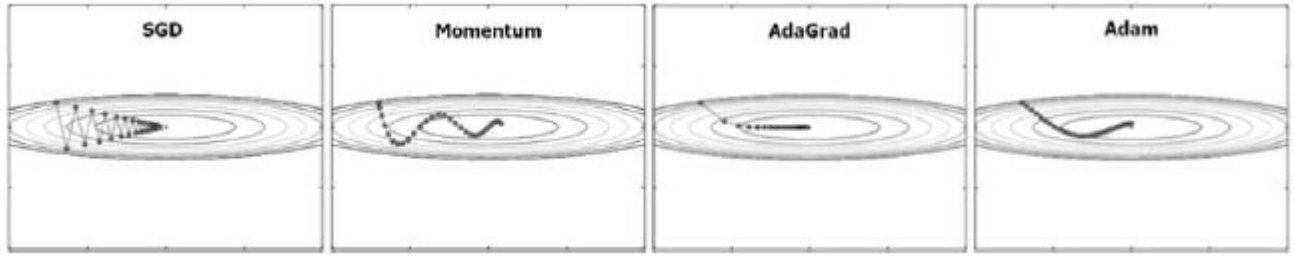
6월 2주차 내용 정리

모델 학습 하이퍼파라미터(Hyperparameter)

Optimizer

- 기본형은 SGD(Stochastic Gradient Descent)이며, 이를 활용한 기법으로 Momentum, AdaGrad(RMSprop), Adam이 대표적

- SGD: 경사하강법. 경사가 급할수록 가중치의변화가 큼
- Momentum:관성 개념접목. 경사의 완급에 따른 변화 정도가 차이남
- AdaGrad(RMSprop): 학습이 깊어질수록 변화량이 작도록 조정
- Adam: SGD 기반, Momentum과 AdaGrad의 특징을 모두 합함



- pytorch에서 제공하는 기본 Optimizer

- https://pytorch.org/docs/stable/optim.html

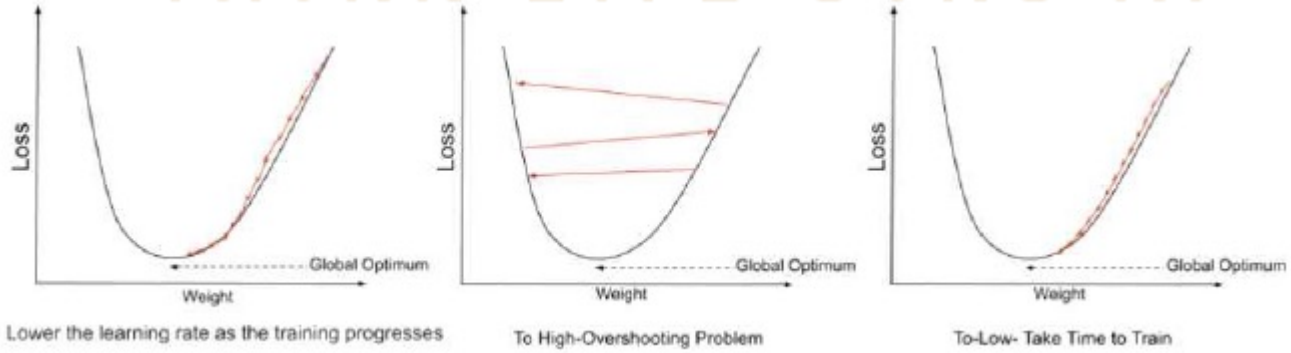
6월 2주차 내용 정리

모델 학습 하이퍼파라미터(Hyperparameter)

3. Learning rate

- Optimizer의 보폭을 조절할 수 있는 하이퍼파라미터
- Learning rate가 크면 Optimizer의 이동이 크고 작으면 작게 이동함
- 0.01을 기본으로 사용하며, 학습되는 양상에 따라 조절하며 테스트함

● 학습률이 큰 경우: Global minimum을 지나침으로써 학습이 발산될 수 있음
● 학습률이 작은 경우: 학습이 너무 오래 걸리거나 Local minimum에서 종료됨



6월 2주차 내용 정리

모델 학습 하이퍼파라미터(Hyperparameter)

4. 배치의 크기(batch size)

- Batch: 모델의 가중치 업데이트마다(학습 한 회당) 일괄 처리되는 데이터 집합
- Batch size: 한 번에 처리될 데이터의 개수
- batch size에 따라 stochastic, full batch, mini-batch로 구분
- 각각 batch size가 1개, 데이터 전체(n), 데이터 일부(1~n 중 사용자 선택)인 경우

● stochastic: 학습 시 소요되는 연산량 적음. 하나씩만 학습하므로 학습 효율 저조
● full batch: 많은 컴퓨팅 자원 소모. 데이터셋 전체를 학습하므로 양질의 학습 가능
● mini-batch: 절충 방식. 사용자의 컴퓨팅 환경에 맞춰 1회 학습량을 조절할 수 있으며 stochastic보다 효율적으로 학습 가능
- mini-batch를 주로 사용하며, 일반적으로 컴퓨터에서 가능한 최대 크기로 설정

6월 2주차 내용 정리

학습횟수, epoch 와 step

- 학습횟수: 보통 epoch의 횟수로 지정하며 사용자가 결정
- epoch: 전체 데이터셋에 대해 한 번 학습을 진행하는 것
- step: 한 epoch에 사용되는 mini-batch의 개수=데이터셋 구성 mini-batch 개수

6월 2주차 내용 정리

학습횟수, epoch 와 step

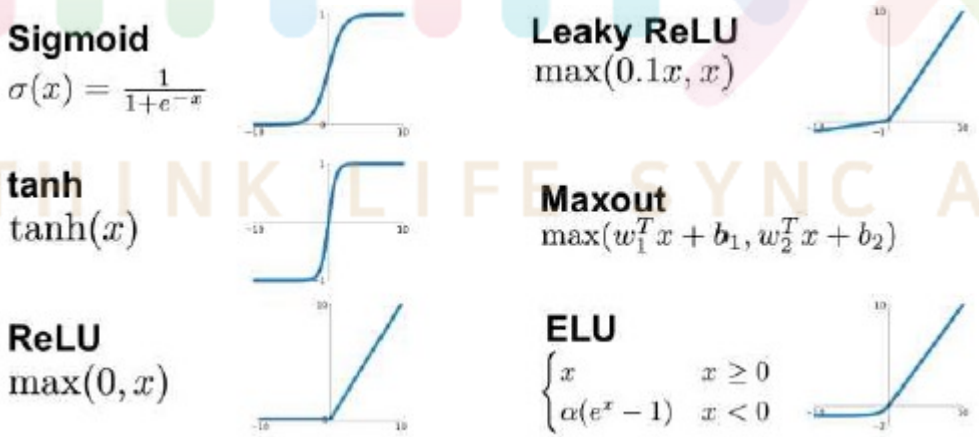
ex) 100개의 데이터를 가진 데이터셋에 대해, mini-batch 값은 5, epoch 10으로 설정하여 학습을 진행한다면,
- 1 epoch 동안 mini-batch는 20회(= 100 / 5) 호출됨
- 즉, 1 epoch는 20 step으로 구성
총 학습횟수는 10 epoch = 200 step(=10 * 20)

ex) 위 상황에서 mini-batch 크기가 10이라면,
- 1 epoch는 10 step으로 구성(= 100 / 10)
- 총 학습횟수는 100 step
※ 학습이 '반복' 수행되는 단위라는 점에서 epoch 또는 step을 iteration으로도 명칭.
단독으로 사용된 iteration은 보통 step을 의미

6월 2주차 내용 정리

활성함수(Activation function)

- 모델의 각 layer 단의 출력값을 조절하기 위해 사용
- 초기 함수인 sigmoid는 gradient vanishing 이슈 등이 발생해 잘 사용되지 않음
- 현재는 ReLU(Rectified Linear Unit)와 Leaky ReLU가 자주 사용됨



딥 러닝 학습 방법론

Model – image net zoo

- Tensorflow, PyTorch, MxNet 등에서 빠른 학습을 위해 모델을 모아 놓은 것
- 원하는 모델을 선택하여 사용(deploy)할 수 있음

TORCHVISION.MODELS

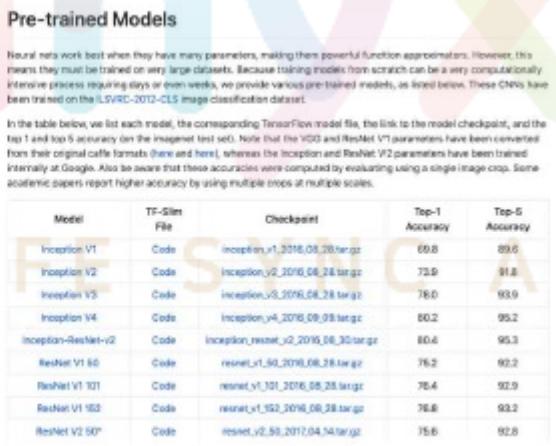
The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

You can construct a model with random weights by calling its constructor:



Pre-trained Models

Neural nets work best when they have many parameters, making them powerful function approximators. However, this means they must be trained on very large datasets. Because training models from scratch can be a very computationally intensive process requiring days or even weeks, we provide various pre-trained models, as listed below. These CNNs have been trained on the ILSVRC-2012-CLS image classification dataset.

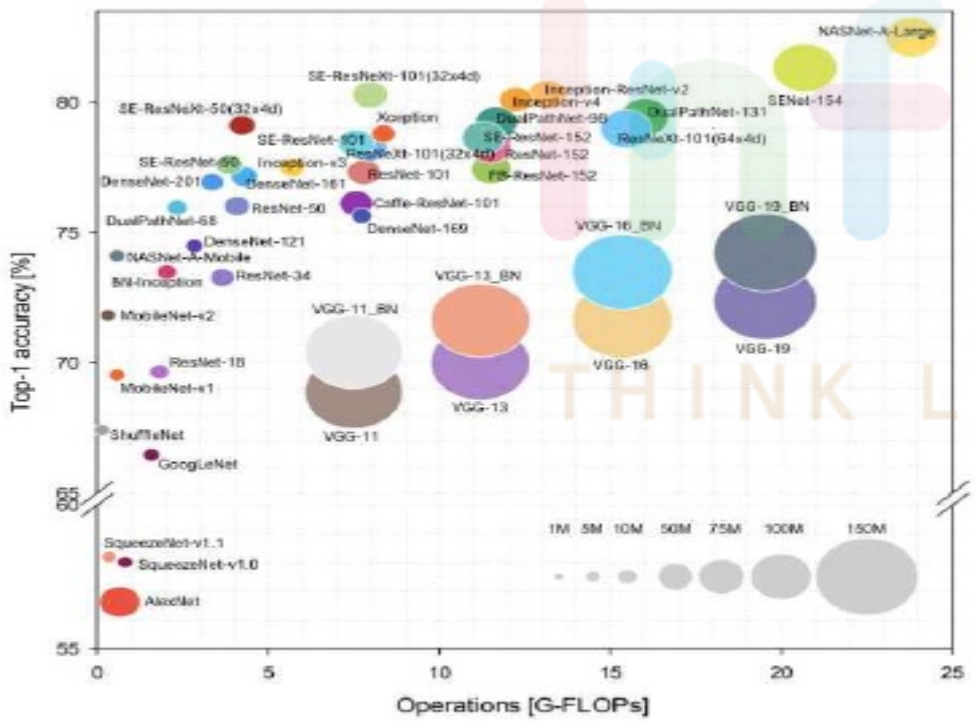
In the table below, we list each model, the corresponding TensorFlow model file, the link to the model checkpoint, and the top 1 and top 5 accuracy (on the imagenet test set). Note that the VGG and ResNet V1 parameters have been converted from their original caffe formats (here and here), whereas the Inception and ResNet V2 parameters have been trained internally at Google. Also be aware that these accuracies were computed by evaluating using a single image crop. Some academic papers report higher accuracy by using multiple crops at multiple scales.

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50*	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8

https://pytorch.org/serve/model_zoo.html

딥 러닝 학습 방법론

Model – image net zoo



[ImageNet 기반 모델 성능 비교 그래프]

※ "일반세팅" 4/8GPU 컴퓨팅 환경에서 ImageNet 크기의 데이터셋에서 학습 시 최소
3일~7일 시간 소요 (GPU 성능 의존)

딥 러닝 학습 방법론

(2) 모델 선택 & 학습: 기존 모델 사용 시
- 일반적으로 ImageNet 기준 상 성능이 좋았던 모델 사용
- 학습할 데이터셋의 크기 등을 고려하여 적절한 모델을 선택하며, 학습 환경에 맞춰 모델을 세부 조정하는 과정에서 새로운 모델을 개발하기도 함
- ResNet 50, VggNet 16 등 모델 사용

(3) 모델 선택 & 학습: 새로운 모델 개발 시
- 모델 구조 설계·개발 및 training scheme 최적화 진행

1) 성능·속도 측면에서 데이터셋에 적합한 모델을 설계·연구(반복 실험 필요)
2) 최적화된 학습 방법을 테스트하여 모델 정확도 향상

※ 학습 방법에 따라 모델 정확도(Accuracy) 기준 3~5% 이상 차이날 수 있음

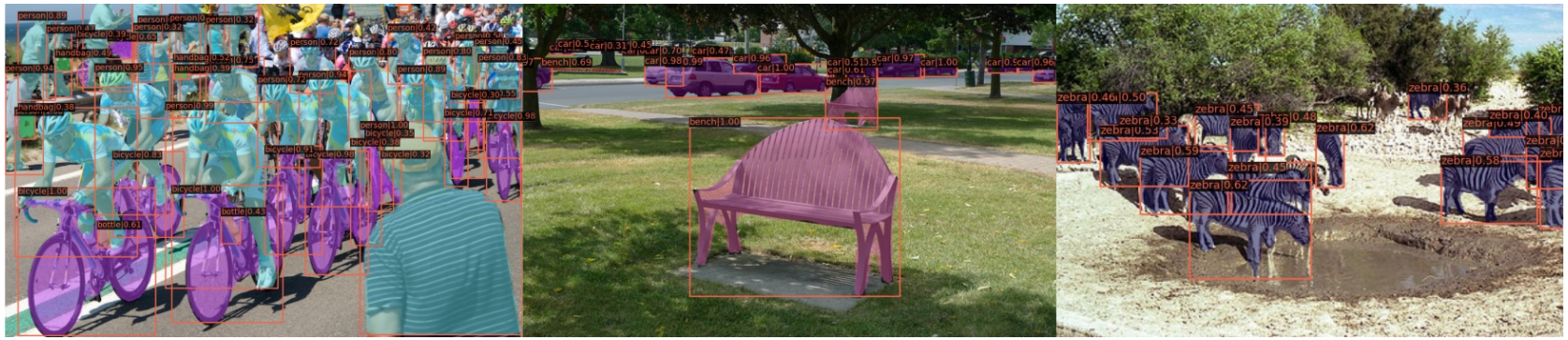
3) 기존 모델과 비교하여 더 좋은 성능을 보이는 지 확인(아닐 경우 1)에서 반복)

딥 러닝 학습 방법론

Model – Detection zoo

OpenMMLab에서는 매우 많은 최신 모델을 Open Source Projects로 구현하여 공개하고 있다.
2021.08.30 기준 11개의 Officially Endorsed Projects와 6개의 Experimental Projects를 공개하고 있다.

MMDetection



딥 러닝 학습 방법론

Model – Detection zoo

Detectron2이란?

Detectron2는 Facebook AI Research(FAIR)에서 개발한 Pytorch 기반의 Object Detection, Segmentation 라이브러리이다. MMDetection과 마찬가지로 모듈식으로 작동한다.

딥 러닝 학습 방법론

성능 측면 – 무거운 딥 러닝 모델

1) VGG-16 (파라미터는 많으나, 생각보다 빠르고 성능이 좋음)
2) ResNet50-SE (ResNet101이나 ResNet152 대비 efficient, GPU 서빙의 마지노선)
3) ResNeXT101-SE + FPN (FPN은 detection등에 적용할 때 성능에 큰 영향)
4) Xception 계통 (의외로 효율적이고 강력한 모델, group-conv 기반)
5) EfficientNet-B4 (효율적이고 성능이 좋지만, dw-conv 기반)

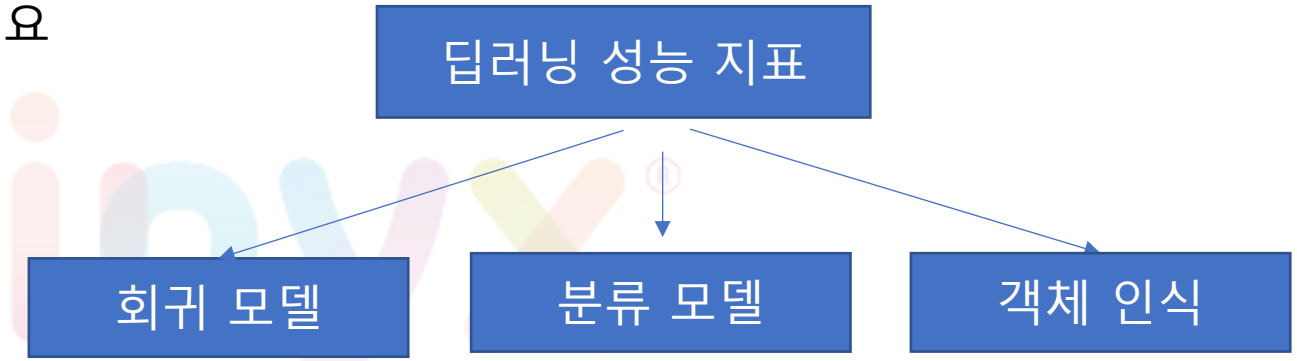
속도 측면 – 가벼운 딥 러닝 모델

1) ResNet-18 (ResNet-50보다 빠르지만, 여전히 크기가 큼)
2) Xception 계통 (CPU용, Xception을 작게 만들어 사용)
3) MobileNetV1 (CPU용, MobileNetV2보다 더 좋을 때가 많음)
4) YOLO 계열
5) SSD 계열

딥러닝 평가 지표

각 모델성능의 평가 지표에 대한 이해 필요

1. 회귀 모델 성능 지표
2. 분류 모델의 성능 지표
3. 객체 인식 모델의 성능 지표



딥러닝 평가 지표

회귀 모델의 성능 지표

1. 평균 절대 오차
2. 평균 제곱 오차
3. 평균 제곱근 오차
4. 평균 절대 비율 오차

- 회귀모델이 잘 만들어 졌는지 확인하는 성능평가지표는 실제값과 예측값을 오차들의 통계값을 활용합니다.

Infinyx

딥러닝 평가 지표

평균절대오차

실제 정답 값과 예측 값의 차이를 절댓값으로 변환한 뒤 합산하여 평균을 구한다.
특이값이 많은 경우에 주로 사용된다. 값이 낮을수록 좋다.

$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

장점
- 직관점임
- 정답 및 예측 값과 같은 단위를 가짐

단점
- 실제 정답보다 낮게 예측했는지, 높게 했는지를 파악하기 힘듦
- 스케일 의존적임(scal dependency): 모델마다 에러율 크기가 동일해도 에러율은 동일하지 않음

딥러닝 평가 지표

평균 제곱 오차

실제 정답 값과 예측 값의 차이를 제곱한 뒤 평균을 구한다. 값이 낮을수록 좋다.

$$MSE = \frac{\sum_{i=1}^{n}(y - \hat{y})^2}{n}$$

장점
- 직관적임

단점
- 제곱하기 때문에 1미만의 에러는 작아지고, 그 이상의 에러는 커짐
- 실제 정답보다 낮게 예측했는지, 높게 했는지를 파악하기 힘듦
- 스케일 의존적임(scal dependency): 모델마다 에러율 크기가 동일해도 에러율은 동일하지 않음

딥러닝 평가 지표

평균 제곱근 오차

MSE에 루트는 씌워서 에러를 제곱해서 생기는 값의 왜곡이 줄어든다. 값이 낮을수록 좋다.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y-\hat{y})^2}{n}}$$

장점

- 직관적임

단점

- 제곱하기 때문에 1미만의 에러는 작아지고, 그 이상의 에러는 커짐
- 실제 정답보다 낮게 예측했는지, 높게 했는지를 파악하기 힘듦
- 스케일 의존적임(scal dependency): 모델마다 에류 크기가 동일해도 에러율은 동일하지 않음

딥러닝 평가 지표

평균 절대 비율 오차

MAE를 비율, 퍼센트로 표현하여 스케일 의존적 에러의 문제점을 개선한다 값이 낮을수록 좋다.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y-\hat{y}}{y}\right|$$

장점

- 직관적임
- 다른 모델과 에러율 비교가 쉬움

단점

- 실제 정답보다 낮게 예측했는지, 높게 했는지를 파악하기 힘듦
- 실제 정답이 1보다작을 경우,무한대의 값으로 수렴할 수 있음

딥러닝 평가 지표

분류 모델의 성능 평가 지표

1. Accuracy
2. Precision
3. Recall
4. F1 score

분류 모델(classifier)을 평가할 때 주로 Confusion Matrix를 기반으로 Accuracy, Precision, Recall, F1 score를 측정한다.

딥러닝 평가 지표

Confusion Matrix(혼동 행렬, 오차 행렬)

분류 모델(classifier)의 성능을 측정하는 데 자주 사용되는 표로 모델이 두 개의 클래스를 얼마나 헷갈려하는지 알 수 있다.

		예측	
		Positive	Negative
정답	Positive	TP	FN
	Negative	FP	TN

- **T**(True): 예측한 것이 **정답**
- **F**(False): 예측한 것이 **오답**
- **P**(Positive): 모델이 **positive**라고 **예측**
- **N**(Negative): 모델이 **negative**라고 **예측**

- **TP**(True Positive): 모델이 **positive**라고 **예측**했는데 실제로 **정답**이 **positive** (정답)
- **TN**(True Negative): 모델이 **negative**라고 **예측**했는데 실제로 **정답**이 **negative** (정답)
- **FP**(False Positive): 모델이 **positive**라고 **예측**했는데 실제로 **정답**이 **negative** (오답)
- **FN**(False Negative): 모델이 **negative**라고 **예측**했는데 실제로 **정답**이 **positive** (오답)

딥러닝 평가 지표

Accuracy(정확도)

모델이 전체 문제 중에서 정답을 맞춘 비율이다.
하지만 데이터가 불균형할 때(ex) positive:negative=9:1)는 Accuracy만으로 제대로 분류했는지는 알 수 없기 때문에 Recall과 Precision을 사용한다.
0 ~ 1 사이의 값을 가지며, 1에 가까울수록 좋다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

딥러닝 평가 지표

Precision(정밀도) = PPV(Positive Predictive Value)

모델이 positive라고 예측한 것들 중에서 실제로 정답이 positive인 비율이다.
실제 정답이 negative인 데이터를 positive라고 잘못 예측하면 안 되는 경우에 중요한 지표가 될 수 있다.
Precision을 높이기 위해선 FP(모델이 positive라고 예측했는데 정답은 negative인 경우)를 낮추는 것이 중요하다.
0 ~ 1 사이의 값을 가지며, 1에 가까울수록 좋다.

$$Precision = \frac{TP}{TP + FP}$$

딥러닝 평가 지표

Recall(재현율) = Sensitivity(민감도) = TPR(True Positive Rate)

실제로 정답이 positive인 것들 중에서 모델이 positive라고 예측한 비율이다.
실제 정답이 positive인 데이터를 negative라고 잘못 예측하면 안 되는 경우에 중요한 지표가 될 수 있다.
Recall를 높이기 위해선 FN(모델이 negative라고 예측했는데 정답이 positive인 경우)을 낮추는 것이 중요하다.
0 ~ 1 사이의 값을 가지며, 1에 가까울수록 좋다.

$$Recall = \frac{TP}{TP + FN}$$

딥러닝 평가 지표

F1 score

Recall과 Precision의 조화평균이다.
Recall과 Precision은 상호 보완적인 평가 지표이기 때문에 F1 score를 사용한다.
Precision과 Recall이 한쪽으로 치우쳐지지 않고 모두 클 때 큰 값을 가진다.
0 ~ 1 사이의 값을 가지며, 1에 가까울수록 좋다.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

딥러닝 평가 지표

객채 인식 모델의 성능 지표

1. IOU
2. Precision Recall Curve
3. AP
4. mAP

딥러닝 평가 지표

객체 인식 모델의 성능 지표

1. IOU

객체 검출의 정확도를 평가하는 지표. 일반적으로 Object Detection에서 개별 객체(Object)에 대한 검출(Detection)이 성공하였는지를 결정하는 지표로 0~1 사이의 값을 가짐

실제 객체 위치 bounding box B_{gt}=gournd truth와 예측한 bounding box B_p=prediction 두 box가 중복되는 영역의 크기를 통해 평가하는 방식으로 겹치는 영역이 넓을 수록 잘 예측한 것으로 평가

딥러닝 평가 지표

객채 인식 모델의 성능 지표

1. IOU

IoU 값은 0~1 사이 값을 갖습니다. 1에 가까울수록 성능이 좋은 겁니다.

실제 경계 박스와 예측 경계 박스가 정확히 일치할수록 IoU가 1에 가까워집니다.

딥러닝 평가 지표

객채 인식 모델의 성능 지표

2. Precision Recall Curve

P-R 곡선(Precision-Recall Curve)는 confidence[1] score를 조정하면서 얻 은 Recall 값의 변화에 따른 Precision을 나타낸 곡선으로 모델(Object detector)의 성능을 평가하는 방법으로, 일반적으로 P-R 곡선의 면적(AOU, Area under curve) 값으로 계산됨

- Recall이 높아져도 Precision이 유지되는 경우, 특정 Class 검출을 위한 모델 성능이 좋을 것으로 평가됨. 즉 Confidence threshold를 변경하더라도, Precision과 Recall이 모두 높은 경우 모델 성능이 좋을 것으로 평가
- 관련된 객체 만 detection 할 수 있는 모델. 즉 오탐 낮은(0 FP = Precision 높음) 경우도 좋은 모델로 평가할 수 있음
- 실제 Object를 모두 찾아내기 위해 Object 수를 많이 Detect 하는 경우 (FP 높은 경우 = Precision이 낮음), 일반적으로 P-R 곡선이 높은 Precision으로 시작하지만, Recall이 증가함에 따라 감소함

딥러닝 평가 지표

객채 인식 모델의 성능 지표

3. Average Precision(AP, 평균 정밀도)

AP 곡선(Average Precision curve)은 Precision과 Recall을 고려한 종합적 평가 지표이며, 실제로 AP는 0~1 사이의 모든 Recall에 대응하는 평균 Precision

- 다른 검출방식에 비해 곡선의 업다운이 심하고, 곡선이 자주 교차하는 경향이 있기 때문에 같은 플롯으로 비교하는 것이 쉽지 않음. AP 평가 지표는 다양한 모델을 비교에 유용한 방식.
- 2010년부터, PASCAL VOC 챌린지에서 AP 계산 방식이 변경되어, 현재는 11점 보간법(11-point interpolation) 뿐만 아니라 모든 점 보간법(interpolating all data)을 사용함

$$AP@[.5:.05:.95] = \frac{(AP50 \times 0.5 + AP55 \times 0.55 + AP60 \times 0.6 + AP65 \times 0.65 + AP70 \times 0.7 + AP75 \times 0.75 + AP80 \times 0.8 + AP85 \times 0.85 + AP90 \times 0.9 + AP95 \times 0.95)}{0.5 + 0.55 + 0.6 + 0.65 + 0.7 + 0.75 + 0.8 + 0.85 + 0.9 + 0.95}$$

딥러닝 평가 지표

객채 인식 모델의 성능 지표

4. mAP(mean Average Performance)

mAP는 모든 점 보간법을 이용해서 AP를 구한 값의 평균

즉, 여러 Class에 대한 AP를 구해야 하므로, 각각의 Class에 대해 AP를 구하고 평균을 산출

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

AP_i : i 번째 class의 AP값

N: 총 Class수

mAP (mean Average Performamce) 계산


감사합니다.
THINK LIFE SYNC AI
Infinyx