Hands On Machine Learning Ch.3
Classification(분류)

발표자: 한형서

Date: 2021.02.07(日)

3.0 Classification

- 가장 일반적인 지도 학습 작업으로 Class 예측
- 즉, 기존에 존재하는 데이터의 유형관계를 파악해서, 새로운, 또는 새롭게 관측된 데이터의 유형을 판별하는 과정

[예시: 옷 사진들이 많은 Data의 유형관계 파악 -> 새로운 옷 사진을 제시했을 때 무슨 옷인지 구별하여 분류하는 작업]

3.1 MNIST (Modified National Institute of Standard and Technology database)

- 미국 인구조사국 지권들과 미국 고등학생이 손으로 쓴 70,000개의 작은 숫자 이미지 데이터 셋.
- Machine Learning 분야의 'Hello World!!'
- tensorflow, keras, 사이킷런 등 다양한 루트를 통해서 데이터 셋을 가져올 수 있음

3.2 Binary Classification 훈련

- 이진 분류기(Binary Classification): **두 개의 클래스**를 구분할 수 있는 분류기

(책의 예시: 5-감지기 [사진이 5인지 아닌지 구별하는 감지기])

[배치 경사 하강법(Batch Gradient Descent: BGD)]

- 모든 트레이닝 데이터를 하나의 배치로 묶어서 학습시키는 경사 하강법.
- 정확하게 최적값을 찾을 수 있음.
- 트레이닝 데이터의 용량이 커질수록, 기울기를 구하는 계산하는 시간이 오래 걸림.

[확률적 경사 하강법(Stochastic Gradient Descent)]

- 트레이닝 데이터 중에서 한 개의 데이터를 하나의 배치로 묶어서 학습시키는 경사 하강법.

(즉, 배치 크기 = 1)

- 매우 큰 데이터셋을 효율적으로 처리하는 장점.

[쉽게 생각하면]

- 배치 경사 하강법: 모든 사진을 집어넣어서 각각의 기울기를 구하는 방식

(사진의 모양이 5일 확률에 대한 각각의 기울기)

- 확률적 경사 하강법: 모든 사진 중에서 임의의 사진들을 뽑아서 하나씩 기울기를 구하는 방식

3.3 성능 측정

1) 교차 검증(Cross Validation)

[교차 검증의 개념]

- 흔히 데이터 셋을 분리할 때 Train - Validation - Test 셋으로 나눔
- 여기서 문제점은 고정된 Test 셋이 고정되어 있다면, 모델을 수정할 때 Test 셋에만 잘 동작하는 모델로 수정할 가능성이 매우 높음 = 즉, test 셋에 과적합(overfitting)될 가능성이 농후하기 때문에, 실제 다른 데이터로 예측을 수행하면 결과가 좋지 않을 수밖에 없다.
- 교차 검증은, 기존 test 셋을 고정해야 하는 관념을 깨고, 데이터 셋의 모든 부분을 사용하여 모델 검증을 하며, test 셋을 고정하지 않는다는 것이다.

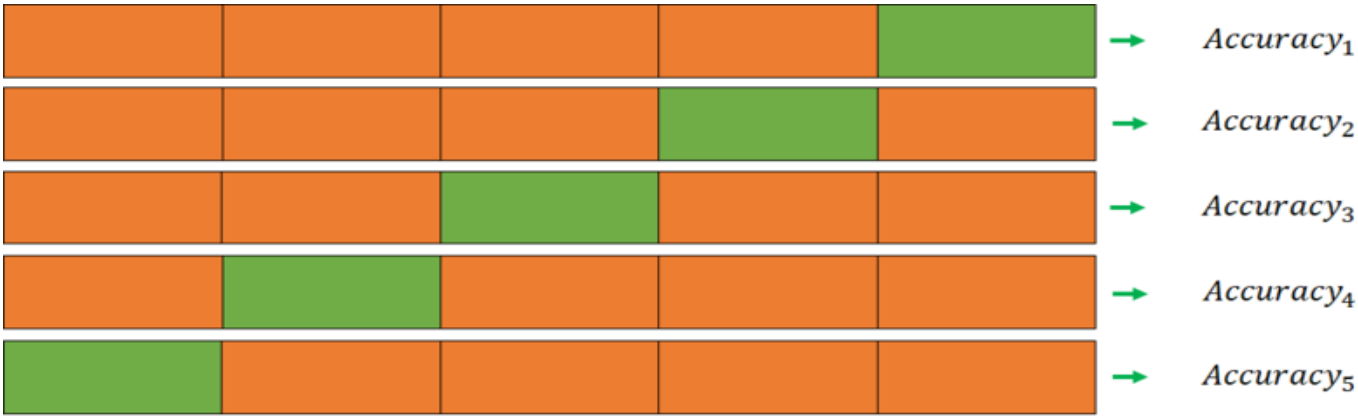
장점: 1. 모든 데이터 셋을 평가에 활용가능(특정 평가 데이터 셋에 overfit 되는 것을 방지 가능)
2. 모든 데이터 셋을 훈련에 활용할 수 있음
   - 정확도 향상 가능
   - 데이터 부족으로 인한 underfitting 방지 가능

단점: Iteration 횟수가 많음 -> 모델의 훈련 및 평가 시간이 오래 걸림.

[교차 검증 종류]

1. K-fold cross validation

- 가장 일반적으로 사용되는 교차 검증 방법
- 데이터를 k개의 '데이터 Fold set'으로 만들고, 총 k번의 학습 및 훈련을 실시함.



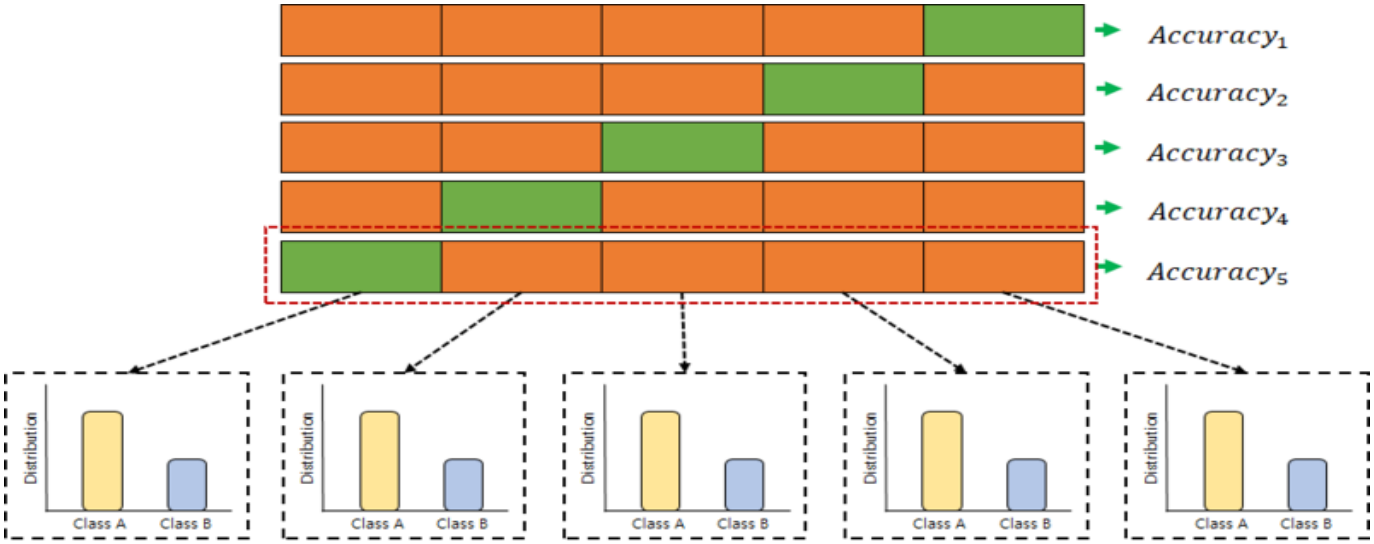
$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_k)$$

(위 사진을 예시를 들면, k = 5로 데이터셋을 train fold 4개, test fold 1개로 나누고
각각의 test마다 test fold가 바뀐다는 것이다.)

- k번의 Iteration후, 각각의 fold set에 대한 검증 결과들을 평균 내어 최종적인 검증 결과로 도출.

- 데이터가 편향되어 있을 경우(한쪽에 몰려 있을 경우) 단순한 K-fold 교차 검증을 사용하면 성능 평가가 잘 안될 가능성이 농후함. -> 이때 쓰는 게 계층별 k-겹 교차검증

- 매개변수로 n_splits, shuffle, random_state를 가짐.
    1. n_splits: 몇 개의 데이터 fold도 분할할지 정하는 매개변수
    2. shuffle: (default값: False) True시, fold로 분할하기전 데이터를 무작위로 섞음
    3. random_state: 랜덤 순서를 재현하기 위한 일종의 세이브 포인트
- 보통 회귀에는 k-fold 교차 검증을 사용, 분류에는 계층별 k-겹 교차 검증을 사용함.



$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_k)$$

2) 오차 행렬(Confusion Matrix)

[오차 행렬의 개념]

- 클래스 C1의 샘플이 클래스 C2로 분류된 횟수를 세는 것.

[예시]

- 모델: 5분류기
- 클래스 C1: 5 이외의 숫자
- 클래스 C2: 숫자 5 이미지

1. 5이외의 숫자 이미지를 정확히 예측: True Negative(TN)
2. 이미지가 숫자 5는 아닌데, 모델에서 숫자 5라고 예측함: False Positive(FP)
3. 이미지는 숫자 5가 맞지만, 모델에서는 5가 아니라고 예측함: False Negative(FN)
4. 숫자 5의 이미지를 정확히 예측: True Positive(TP)

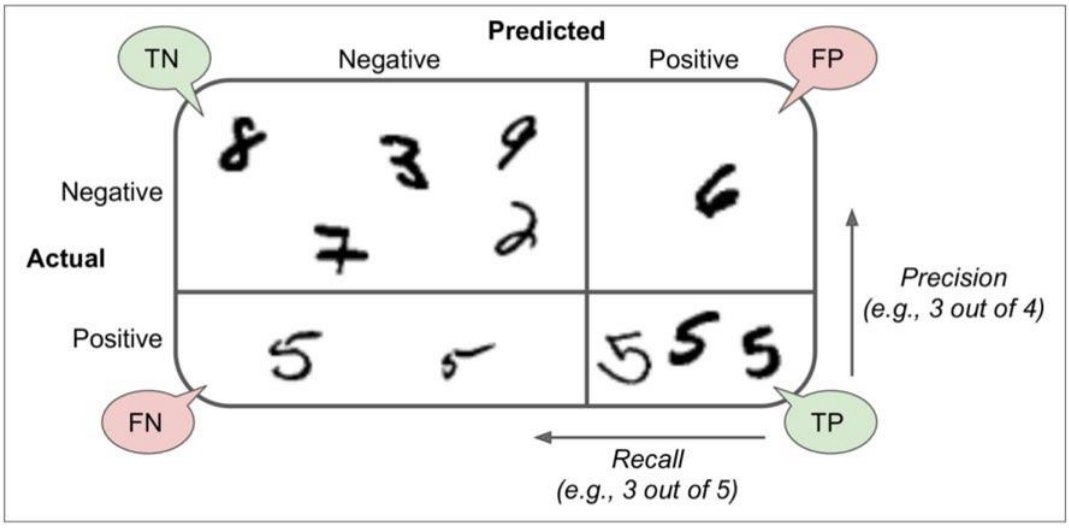


Figure 3-2. An illustrated confusion matrix

[오차 행렬을 통해 얻을 수 있는 정보?]

정밀도(Precision)/재현율(recall) = 민감도(sensitivity) = 진실 양성 비율(True Positive Rate/TPR)

- 정밀도: TP(True Positive)/TP+FP(False Positive)
- 재현율: TP/TP+FN(False Negative)

3) 정밀도와 재현율 및 트레이드오프

- 정밀도: 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율
- 재현율: 실제 True인 것 중에서 모델이 True라고 예측한 것의 비율
- 정밀도와 재현율은 trade-off 관계, 즉 정밀도가 상승 -> 재현율은 하락 | 재현율이 상승 -> 정밀도 하락
- 따라서 어느 것을 우선 순위에 두는지에 따라서 하나의 수치를 낮춰야 한다.

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

- 따라서 높은 정밀도를 요구 시, 최소 재현율 기준을 물어보는 것이 좋음. 또는 두 수치의 균형점을 찾아줘야 한다. (둘 다 고려하는 평가 수치: F1 Score)

[정밀도 및 재현율 트레이드오프]

- 기본 개념: 정밀도가 상승 -> 재현율 하락 | 재현율 상숭 -> 정밀도 하락
- 임계점을 설정(default = 50% 지점)해서 정밀도와 재현율들을 구함
- 임계점에 따라서 구해지는 정밀도와 재현율에 차이가 존재함
- 임곗값 증가 -> 정밀도(Precision) 상승, 재현율(Recall) 감소
- 임곗값 감소 -> 재현율(Rcall) 상승, 정밀도(Precision) 감소

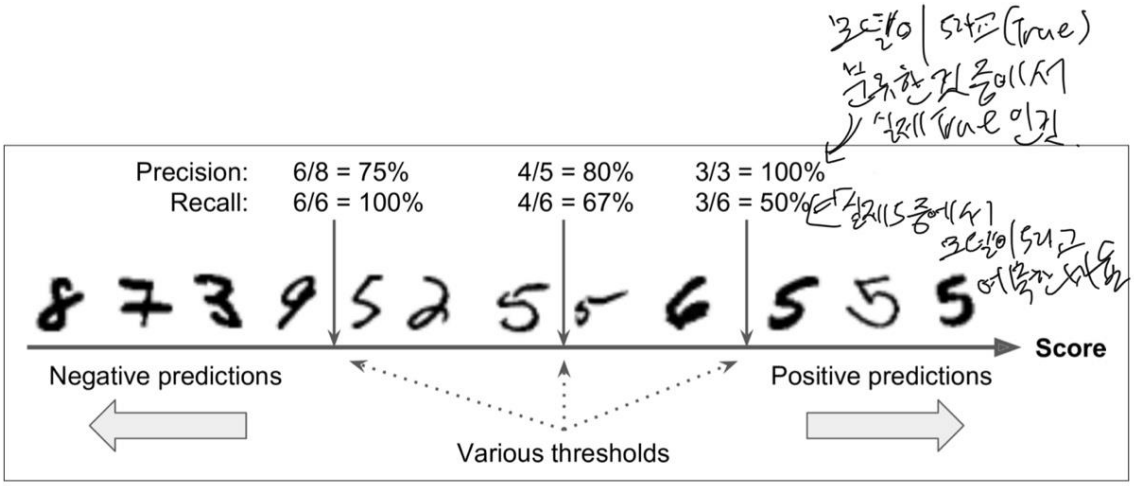


Figure 3-3. Decision threshold and precision/recall tradeoff

- decision_function 리턴값 -> True에 (즉, 클래스C2에) 속한다고 믿는 정도를 실수 값으로 리턴. 양수 값은 양성 클래스 의미, 음수 값을 음성(즉 다른) 클래스 의미

4) ROC 곡선: 모든 분류 임계값에서 분류 모델의 성능을 보여줌

참양성비율(True Positive Rate)와 허위 양성 비율(False Positive Rate)

TPR = TP/TP+FN

FPR = FP/FP+TN

-TRP과 FPR은 서로 반비례적 관계임.

(예: 의사가 암환자를 진단할 때, 성급한 의사는 아주 조금의 징후만 보여도 암이라고 판단함.
-> 이 경우 TPR은 1에 가까워지지만 FPR은 반대로 매우 낮아짐. [정상인 사람도 다 암이니깐])
(예: 돌팔이 의사가 암환자를 알아내지 못한다면, 모든 환자에 대해 암이 아니라고 할거임.
-> 이 경우 TPR은 매우 낮아지고, FPR은 급격히 높아져 1에 가까워짐.
[암환자가 와도 진단을 안 해버리니, 암환자라고 잘못 진단하는 경우가 없음.])

[예시]

확률적 경사 하강법(SGD)와 랜덤 포레스트(Random Forest) 분류의 ROC 및 AUC를 통한 비교

3.4 다중 분류 : 둘 이상의 클래스를 구별할 수 있는 경우.

[종류]

1) 알고리즘으로 여러 개의 클래스를 직접 처리하는 방식
(ex. 랜덤 포레스트 분류기, 나이브 베이즈 분류기)
2) 이진 분류기를 여러 개 사용하여 다중 클래스를 분류

[2번의 방식으로 다중 클래스를 분류할 때 방식]

1) OvA(one-versus-all) : 각 분류기의 결정 점수 중에서 가장 높은 것을 클래스로 선택
(예 0~9까지의 분류기를 각각 만들고 데이터를 모든 분류기에 통과시킨 후, 점수가 제일 높은 숫자로 인식을 하게됨)

2) OvO(one-versus-one): 0과 1 구별, 1과 2 구별, 0과 2 구별 등 각 클래스의 조합마다 이진분류기를 훈련 시키는 방법
(분류기는 총 N(N-1(2)개가 필요함)
-OvO 전략의 장점 : 각 분류기의 훈련에 전체 훈련 세트 중 구별할 두 클래스의 해당되는 샘플만 필요하다는 것.
(예 : 1과 2를 구별하는 구별기를 생성할 때, 1 그리고 2의 데이터셋만 있으면 된다는 말.)

3.5 에러 분석

[우리가 실제 프로젝트를 진행할 때]

- 머신러닝 프로젝트 체크리스트를 통해 만들어짐(챕터 2 참고). 즉,
    1. 데이터 준비 단계에서 가능한 선택사항을 탐색
    2. 여러 모델을 시도
    3. 가장 좋은 몇 개를 골라 GridSearchCV를 사용해 하이퍼파라미터를 세밀하게 튜닝
    4. 파이프라인을 통해 가능한 자동화 해야함.

[이번 섹터에선 가능성이 높은 모델을 하나 찾았다고 가정하고 모델의 성능을 향상시킬 방법 모색]
-> 한가지 방법으로 에러의 종류를 분석하는 것.

3.6 다중 레이블 분류: 한 샘플이 여러 클래스에 속할 때

예) 여러 사람이 나오는 한 사진 -> 누가 있고 누가 없는지 구분할 때
[영희, 철수, 짱구를 분류하는 분류기 -> 영희와 짱구가 나온 사진을 보고 해당 인물들이 있다는 것을 분류기가 알아야한다]

- 다중 레이블 분류기 중에서 사이킷런에서는 KNeighborsClassifier 분류기가 있음.

- 평가 지표: 각 레이블의 f1 점수를 구하고 평균을 내기
    ->f1_score함수의 average옵션을 macro로 주면 f1들의 평균값을 계산해줌.

3.7 다중 출력 분류 = 다중 출력 다중 클래스 분류
-> 하나의 레이블이 다중 클래스가 될 수 있도록 일반화한 것.