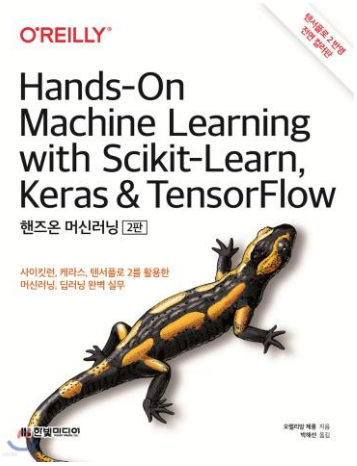
모두의연구소



핸즈온 머신러닝(chapter 3)

모두의연구소

Contents

이진 분류기 훈련

다음시간까지 문제 풀 것

모두의연구소

문제

1. Stochastic Gradient Descent이 무엇인가? 그리고 GD와 SGD의 수식적인 차이가 무엇인지 간단히 말하시오.

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} x^i \left(wx^i + b - y^i\right)$$

이진 분류기 훈련

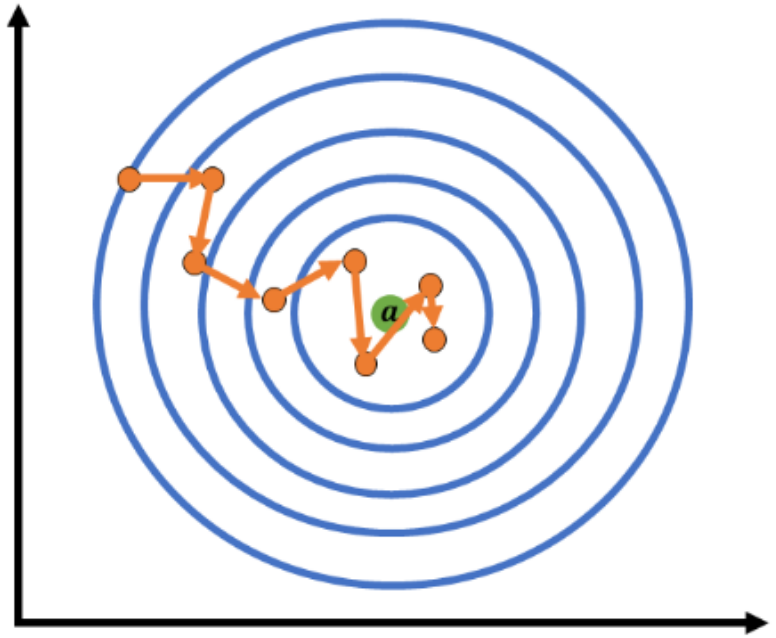
모두의연구소

SGDClassifier

- 확률적 경사하강법
- 한 번에 하나씩 훈련 샘플 처리 후 파라미터 조정
- 매우 큰 데이터셋 처리에 효율적이며 온라인 학습에도 적합함

$$\frac{\partial \mathcal{L}}{\partial w} = x\,(wx + b - y)$$

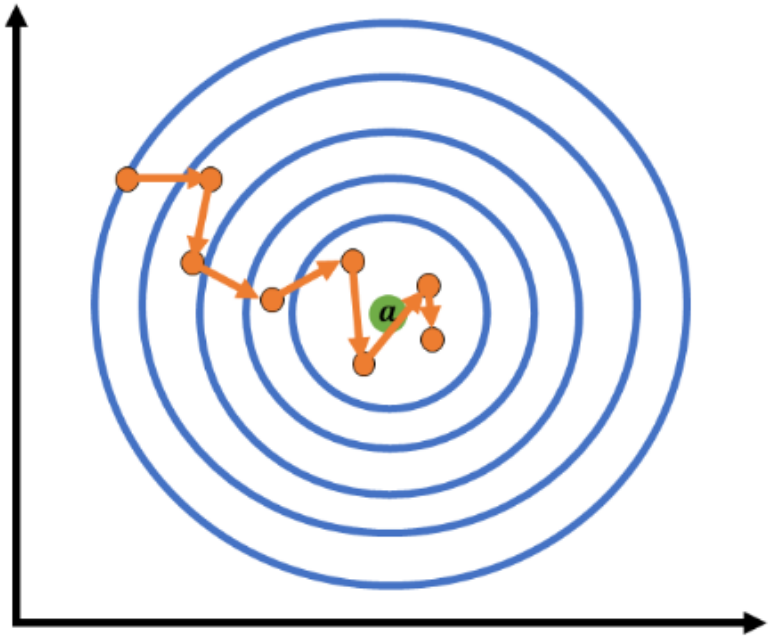
```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train_5)
```



이진 분류기 훈련

SGDClassifier

- 오로지 샘플 데이터셋에 대해서만 경사를 계산하므로 매 반복마다 다뤄야 할 데이터가 줄어들었고, 학습 속도가 빠르다는 장점이 있습니다.
- 같은 이유로 메모리 소모량이 매우 낮으며, 큰 데이터 셋이라 할지라도 학습이 가능합니다.
- 학습 중간 과정에서 진폭이 크고 배치 경사 하강법보다 불안정하게 움직입니다.
- 데이터를 하나씩 처리하기 때문에 오차율이 크고, GPU의 성능을 전부 활용할 수 없습니다.
- 손실 함수가 최솟값에 가는 과정이 불안정하다 보니 최적해(global minimum)에 정확히 도달하지 못할 가능성이 있습니다.
- 배치 경사 하강법과 반대로 local minimum에 빠지더라도 쉽게 빠져나올 수 있습니다. 또한, global minimum을 찾을 가능성이 SGD가 더 큽니다.



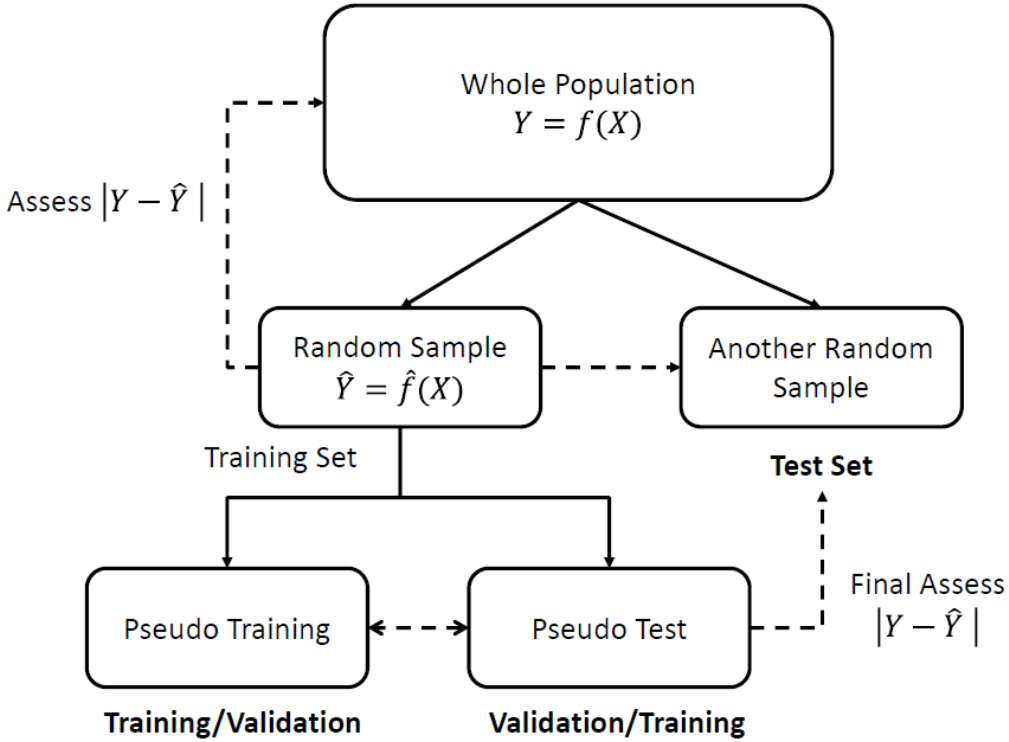
성능 측정

성능 측정

모두의연구소

교차 검증을 활용한 정확도 측정

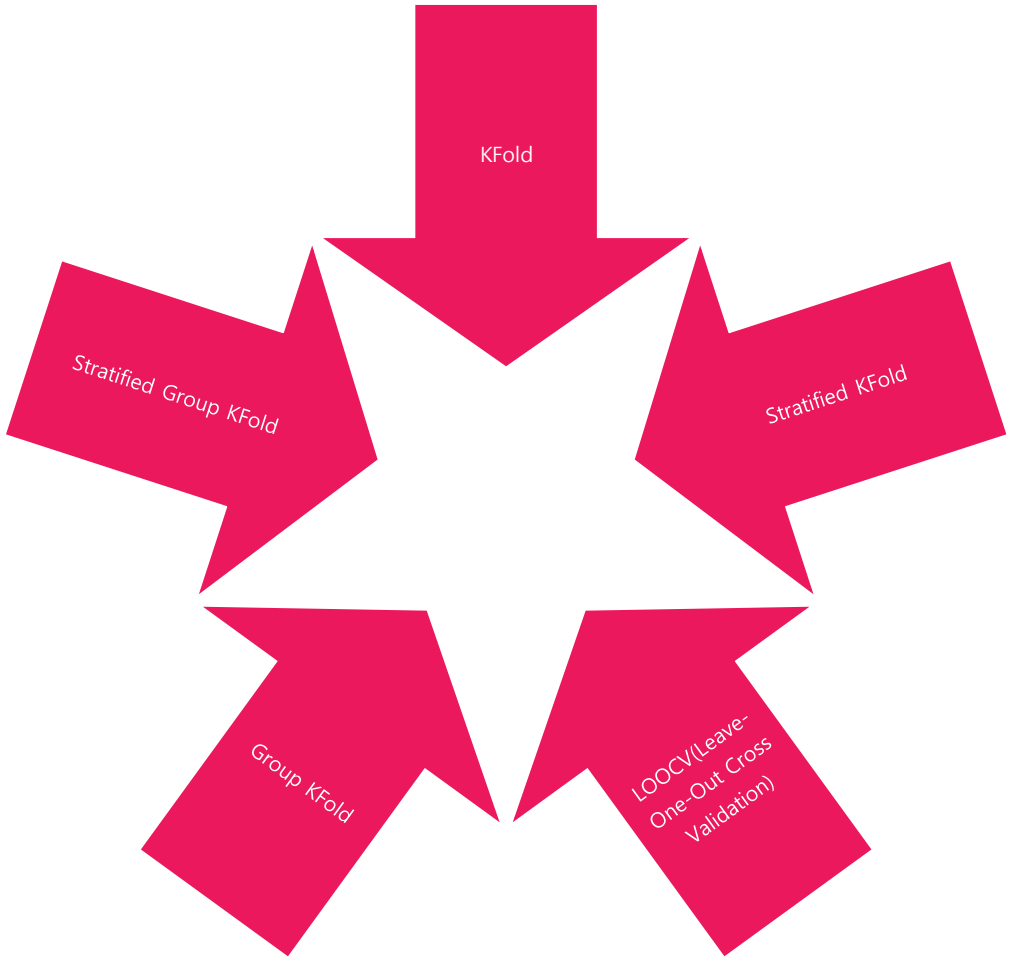
- 교차 검증은 필수적인 요소는 아니지만 필요하다고 생각함
- 특히 kaggle에서 많이 활용함
- 최근 추천시스템 대회에서 교차검증을 활용하여 높은 성적을 냈음
- Similar with the validation set approach, but use both sets for training and validation



성능 측정

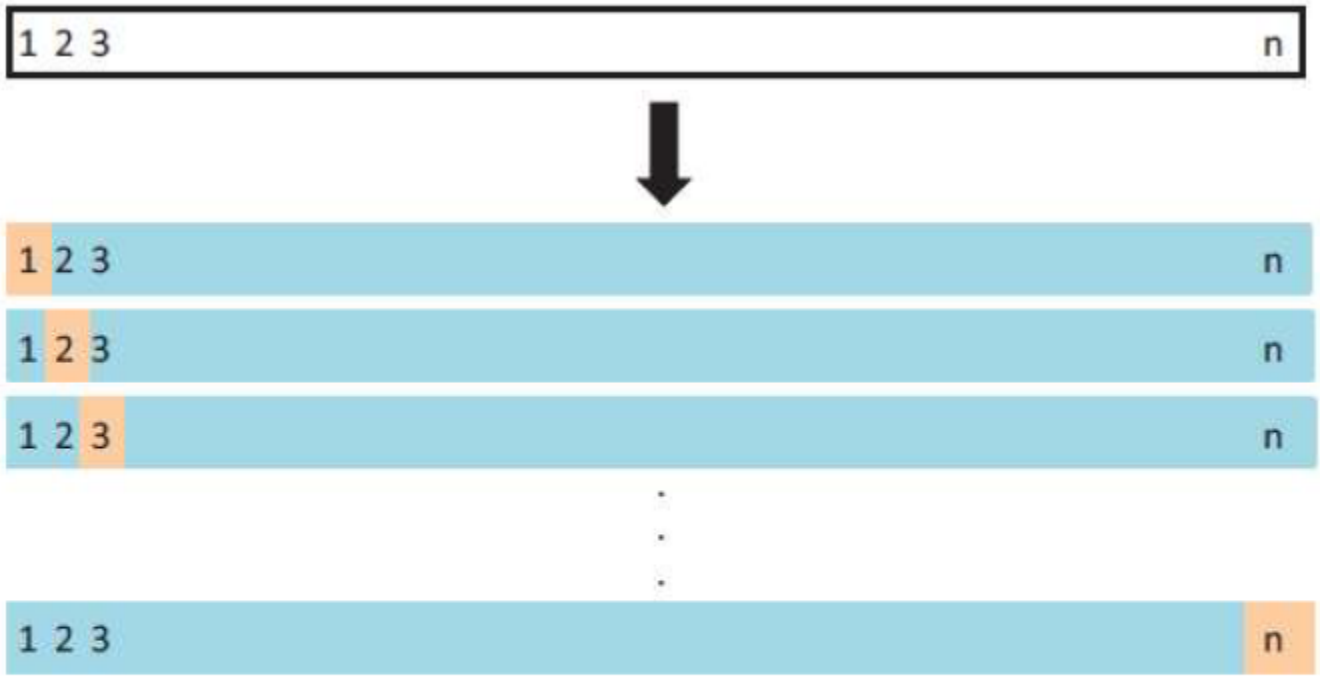
모두의연구소

교차 검증의 종류



성능 측정

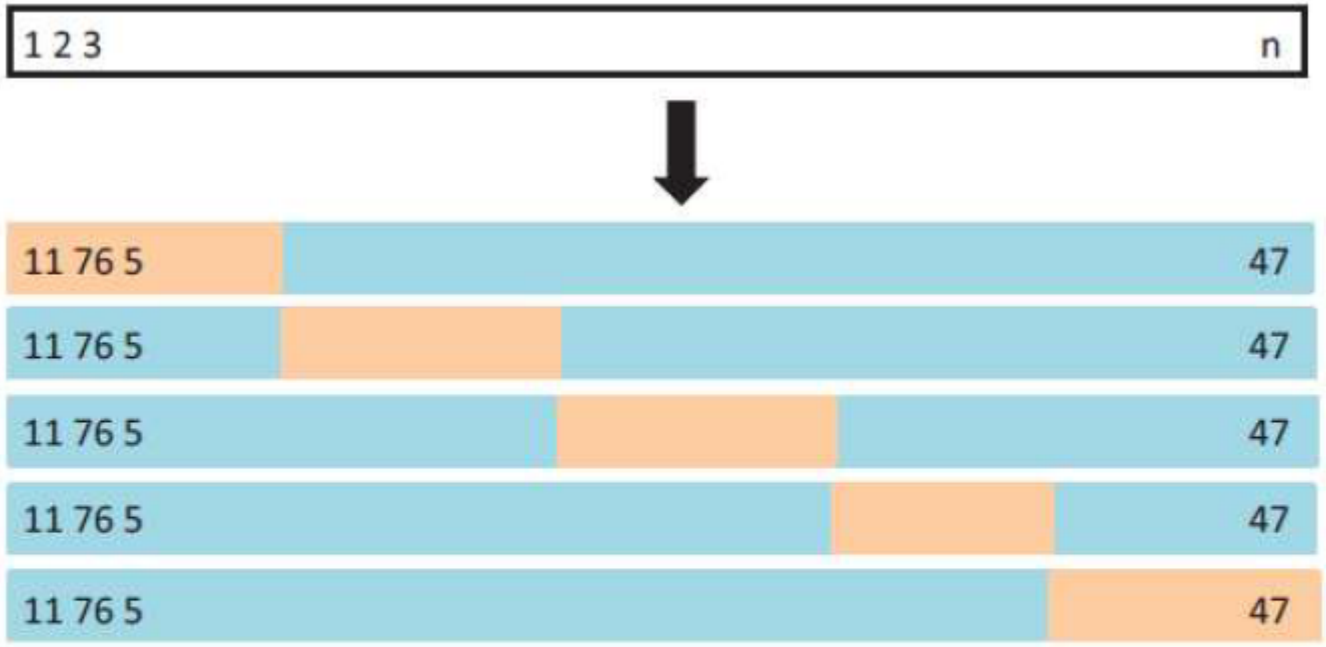
Leave-One-Out Cross Validation(LOOCV)



$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

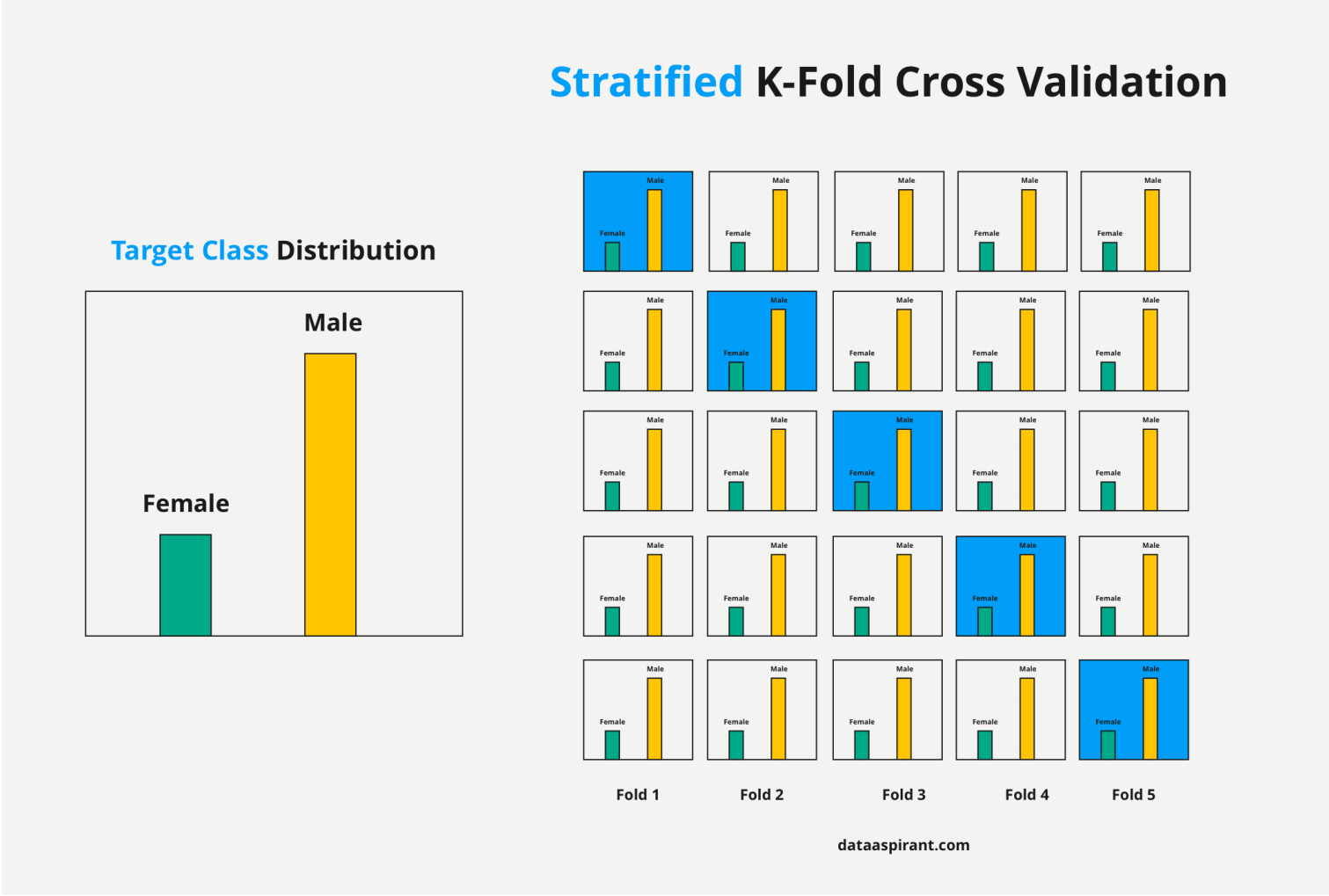
성능 측정

KFold Cross Validation



$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

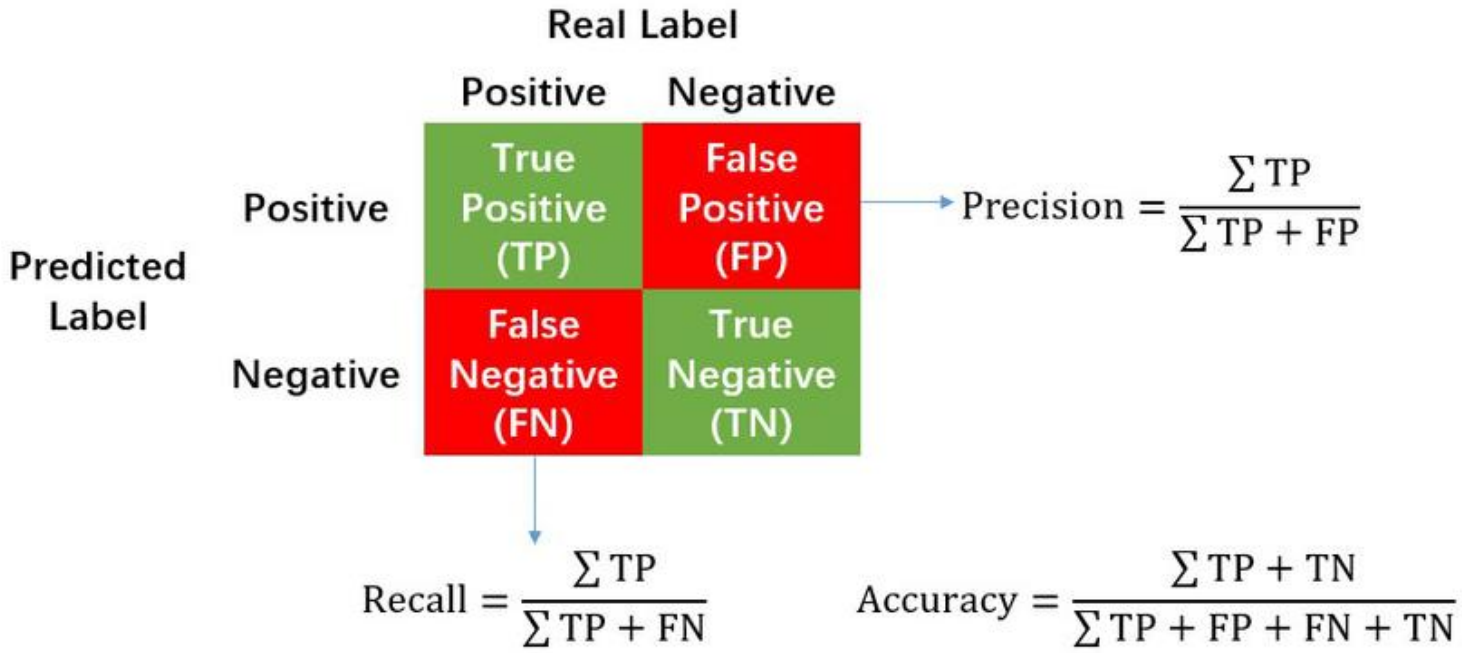
Stratified KFold Cross Validation



성능 측정

평가지표

- $Log\ loss = -\sum_t^C t_i \log(f(s)_i)$
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F\ score = \frac{1}{\alpha\frac{1}{Precision}+(1-\alpha)\frac{1}{Recall}}$
- $ROC\ AUC = \int_0^1 TPR\left(FPR^{-1}(x)\right)dx$



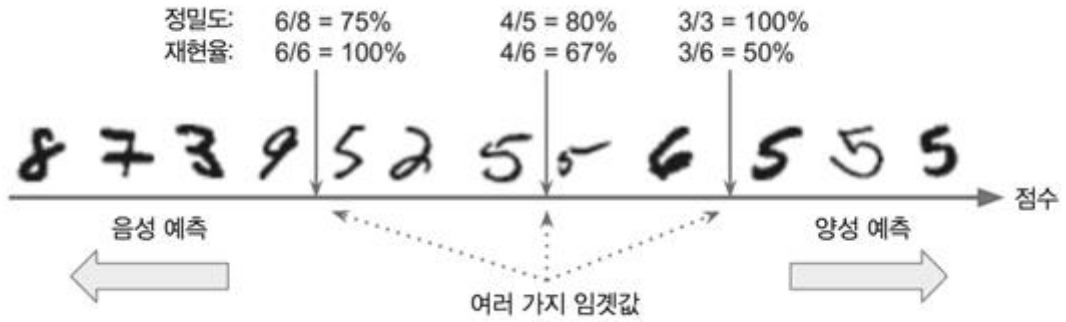
성능 측정

모두의연구소

정밀도/재현율 트레이드오프

- 결정 함수($decision\ function$): 분류기가각 샘플의 점수를 계산할 때 사용
- 결정 임계값($decision\ threshold$): 결정 함수의 값이 이 값보다 갖거나 크면

  양성 클래스로 분류, 아니면 음성 클래스

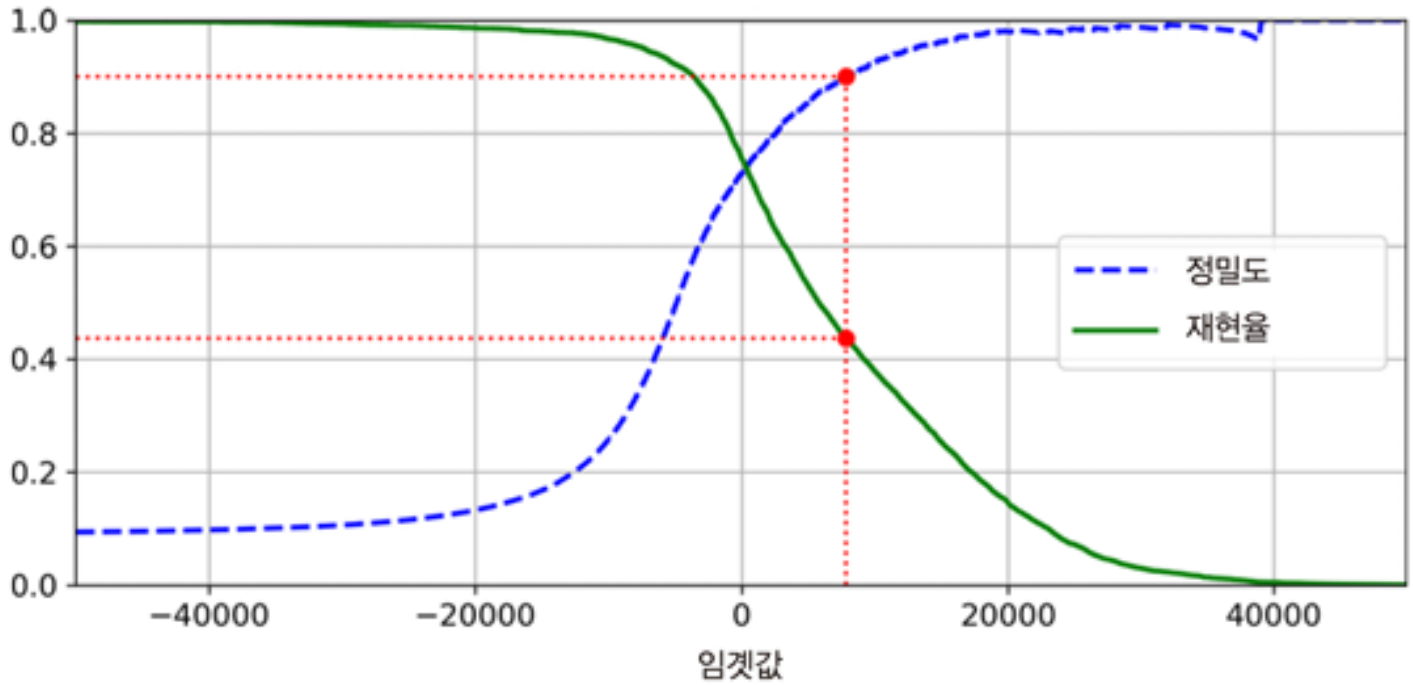


성능 측정

모두의연구소

정밀도/재현율 트레이드오프

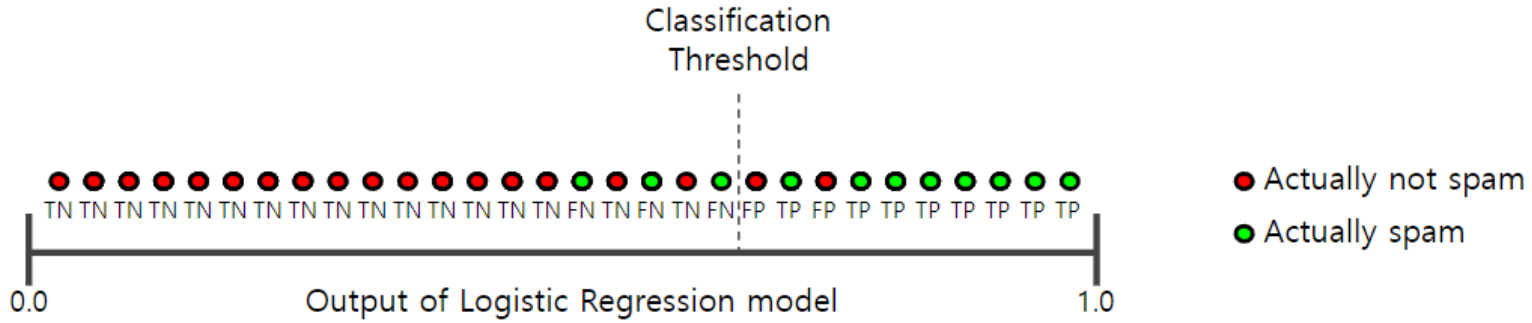
- 임계 값이 커질수록 정밀도 올라감, 재현율 떨어짐



정밀도/재현율 트레이드오프

주의사항

- 정밀도 곡선이 높은 결정 임계점에서 울퉁불퉁하다. 반면에 재현율은 무조건 줄어들며, 따라서 부드럽게 0에 수렴한다.
- 예제 1: 책 137쪽 예제 참조
  - 가운데 임계값을 오른쪽으로 숫자 하나만큼 이동하면 정밀도가 4/5에서 3/4로 줄어든다.
- 예제 2: 아래 스팸메일을 분류기의 결과를 보면 임계값(Classification Threshold)이 오른쪽으로 한 칸씩 이동하면 정밀도가 8/10=0.8, 8/9=0.889, 7/8=0.875, 7/7=1 등으로 오르락내리락 하면서 변한다. 물론 최종적으로는 1에 수렴한다. (녹색은 스팸 메일을, 빨간색은 정상 메일을 나타냄)



<그림 출처: 구글 ML 단기집중강좌: 정밀도와 재현율>

정밀도/재현율 조율: 좋은 결정 임계값 정하기

앞서 설명하였듯이 정밀도와 재현율 사이에 서로 줄다리기 게임을 한다. 따라서 어떻게 좋은 결정 임계값을 정해야 할지 기준이 요구된다. 예를 들어, F_1 점수를 높게 만드는 결정 임계값을 사용할 수도 있다. 다만, 이미 언급한대로 조심해야할 필요가 있다. 여기서는 정밀도와 재현율 사이의 직접적인 관계를 그래프로 나타낸 후 좋은 결정 임계값을 정하는 다른 방법을 살펴본다.

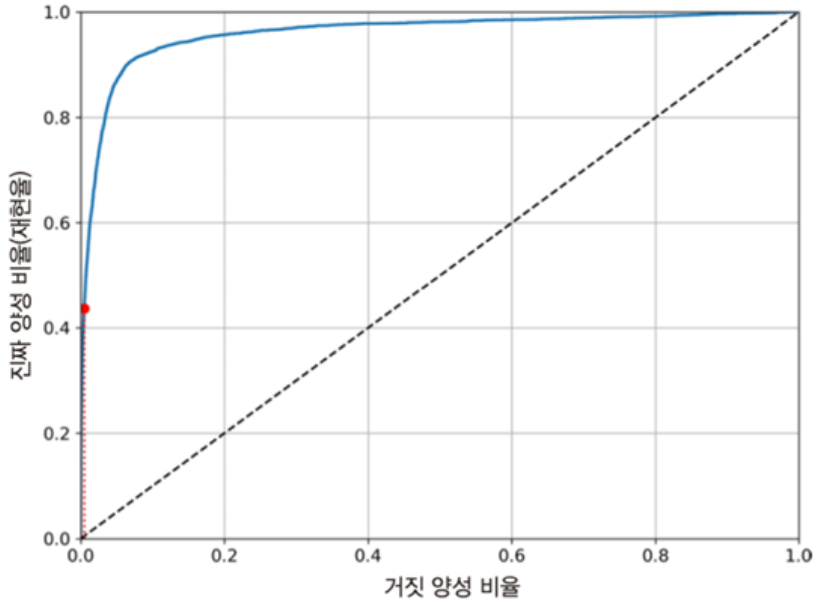
정밀도와 재현율 사이의 그래프는 앞서 확인한 값들을 그대로 이용한다. 앞서 설명한 대로 정밀도의 마지막 항목과 재현율의 마지막 항목을 그대로 사용하기 때문에 그래프가 y-축에서부터 시작한다는 사실에 주의하라.

성능 측정

ROC 곡선

- 결정 임곗값에 따른 거짓 양성 비율(false positive rate, FPR)에 대한 참 양성 비율(true positive rate, TPR)의 관계를 나타낸 곡선
- 참양성 비율: 재현율
- 거짓 양성 비율: 원래 음성인 샘플 중에서 양성이라고 잘못 분류된 샘플들의 비율

$$Recall = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

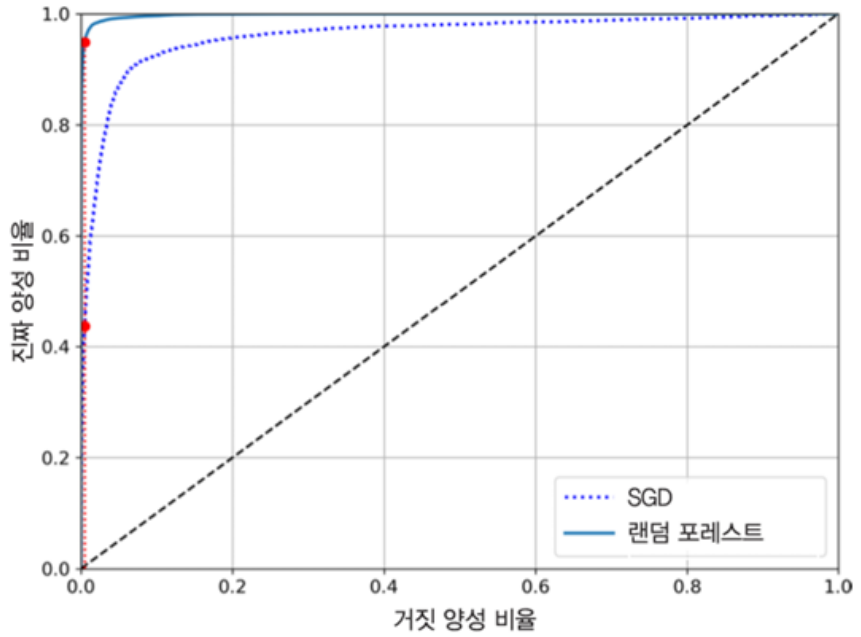


성능 측정

모두의연구소

AUC와 분류기 성능

- 재현율(TPR)과 거짓 양성 비율(FPR) 사이에도 서로 상쇄하는 기능이 있다는 것을 확인 가능
  - 재현율(TPR)을 높이고자 하면 거짓 양성 비율(FPR)도 함께 증가
- 따라서 좋은 분류기는 재현율은 높으면서 거짓 양성 비율은 최대한 낮게 유지해야함
- ROC 곡선이 y축에 최대한 근접하는 결과가 나오도록 해야함.
- AUC: ROC 곡선 아래의 면적
  - 이 면적이 1에 가까울 수록 성능이 좋은 분류기로 평가



다중 분류

One vs Rest

- 정의 -

One vs Rest 방법은 다중 클래스 문제를 여러 개의 이진 분류 문제로 바꾸는데 이때 하나의 이진 분류기는 한 클래스를 1, 나머지 클래스를 0으로 취급하여 학습된다.

a. 과정

먼저 K개의 클라스가 있다고 하자. 그리고 데이터 $(x_i, c_i), i = 1, \ldots, n$가 있다. 이때 x_i는 p차원 설명 변수 벡터이고 $c_i \in \{1, \ldots, K\}$이다.

이러한 상황에서 One vs Rest 방법은 총 K개의 이진(Binary) 분류기가 다음과 같이 만들어진다.

$k \in \{1, \ldots, K\}$에 대하여

(1) 만약 $c_i = k$라면 $z_i = c_i$, 아닌 경우 $z_i = 0$으로 설정한다.

(2) 새롭게 만들어진 데이터 $(x_i, z_i), i = 1, \ldots, n$에 대하여 이진 분류기를 학습한다.

One vs Rest

b. 예측

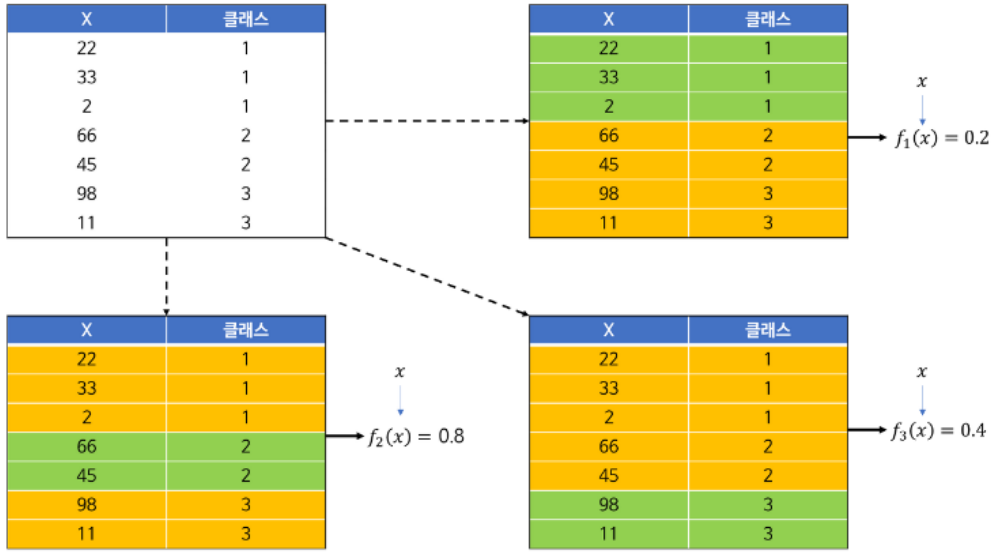
새로운 데이터 x가 들어왔을 때 $z_i = k, k = 1, \ldots, K$일 확률(또는 점수)을 나타내는 $f_k(x)$를 계산한다. 이런 식으로 K개의 확률(또는 점수)을 계산하고 이를 최대화하는 k를 최종 예측 라벨로 선정한다.

$$\hat{c} = \arg\max_k f_k(x)$$

만약 분류기가 확률을 계산할 수 없고(예: 서포트 벡터 머신) 라벨만을 예측하는 경우에는 가장 많이 나온 라벨을 예측 라벨로 한다.

One vs Rest

먼저 3개의 클래스가 있다고 하자. 각각 라벨은 1, 2, 3으로 하겠다. 그렇다면 아래와 같이 3개의 이진 분류기를 학습하고 각 클래스에 대한 확률(또는 점수)을 계산한다.



one vs rest 설명

위 그림에서 라벨 1과 나머지(2, 3)에 대해서 이진 분류기를 학습하고 새로운 데이터 x에 대하여 라벨 1일 확률이 $f_1(x) = 0.2$으로 계산된 것이다. 라벨 2와 나머지 (1, 3)에 대해서도 이진 분류기를 학습하고 새로운 데이터 x에 대하여 라벨 2일 확률이 $f_2(x) = 0.8$로 계산 된 것이다.

그렇다면 새로운 데이터 x의 라벨은 어떻게 예측될까? $f_2(x)$의 값이 제일 크므로 예측 라벨은 2가 된다.

$$\begin{aligned}\arg\max_{k\in\{1,2,3\}}\{f_1(x), f_2(x), f_3(x)\} &= \arg\max_{k\in\{1,2,3\}}\{0.2, 0.8, 0.4\} \\ &= 2\end{aligned}$$

One vs Rest

- 장단점 -

a. 장점

구현이 쉽다.

b. 단점

One vs Rest 방법은 K 클래스인 경우 K개의 이진 분류기를 학습한다. 이때 각 분류기가 확률이 아닌 다른 점수(Score)를 계산한다면 이러한 점수들은 이진 분류기 별로 스케일 차이가 날 수 있다. 이는 이진 분류기들의 각 클래스에 대한 점수 비교를 어렵게 한다.

또한 원(Raw) 데이터 상으로 각 클래스별로 균형 잡힌(Balanced) 데이터인데 이를 한 클래스대 나머지로 설정해버리면 나머지 클래스 데이터가 일방적으로 많게 되어 불균형(Unbalanced) 데이터가 된다. 이는 각 이진 분류기가 학습이 잘 안 되는 결과를 초래할 수 있다.

One vs One

- 정의 -

One vs One 방법은 다중 클래스(Multi-Class)를 가진 데이터를 서로 다른 두 개의 클래스를 갖는 데이터로 쪼개고 이를 이진 분류기로 학습하는 방법이다.

a. 과정

먼저 K개의 클래스가 있다고 하자. 그리고 데이터 $(x_i, c_i), i = 1, \ldots, n$가 있다. 이때 x_i는 p차원 설명 변수 벡터이고 $c_i \in \{1, \ldots, K\}$이다.

그렇다면 K개 중에서 서로 다른 2개를 뽑는 조합의 수 $J = K(K-1)/2$만큼의 이진 분류기를 학습하게 되는데 과정은 다음과 같다.

$j \in \{1, \ldots, J\}$에 대하여
j번째 클래스 쌍을 (m, n)이라 하자. 즉, $1 \leq n \leq K, 1 \leq m \leq K$이다. 이때 전체 데이터 중에서 $c_i \in \{m, n\}$인 데이터를 추출한 뒤 이를 이용하여 이진 분류기를 학습한다.

One vs One

b. 예측

새로운 데이터가 x 들어왔을 때 j번째 이진 분류기에 확률(또는 점수) 값을 $f_j(x)$라 하자. 이때 $f_j(x)$를 최대화하는 j를 J^* 다음과 같다고 해보자.

$$J^* = \underset{j \in \{1,\ldots,J\}}{\arg\max}\, f_j(x)$$

새로운 데이터가 x 들어왔을 때 예측 라벨은 다음과 같다.

$$f_{J^*}(x)\text{'s Label}$$

위 식은 예제를 보면 이해가 될 것이다.

만약 분류기가 확률을 계산할 수 없고(예: 서포트 벡터 머신) 라벨만을 예측하는 경우에는 가장 많이 나온 라벨을 예측 라벨로 한다.

One vs One

이번에도 라벨이 3개 있다고 하고 각 라벨은 1, 2, 3이다. 이 경우 아래와 같이 3(=(3x2)/2) 개의 이진 분류기를 학습하게 된다.

j	클래스 쌍
1	1, 2
2	1, 3
3	2, 3

새로운 x에 대하여 j번째 학습된 이진 분류기의 라벨이 k인 확률 $f_{jk}(x)$가 다음과 같이 계산되었다고 하자.

j	예측 확률
1	$f_{11}(x) = 0.2, f_{12}(x) = 0.8$
2	$f_{21}(x) = 0.4, f_{23}(x) = 0.6$
3	$f_{32}(x) = 0.5\ f_{33}(x) = 0.5$

이 경우 예측 확률의 최대값은 $f_{12}(x) = 0.8$이고 이에 대응하는 라벨인 2를 최종 예측 라벨로 선정하게 된다.

One vs One

- 장단점 -

a. 장점

구현이 쉽다.

b. 단점

라벨 개수가 많아지면 학습해야 할 이진 분류기가 많아지게 된다. 그리고 확률이 아닌 점수와 같은 형태로 예측하게 된다면 One vs Rest에서의 단점과 같이 스케일이 문제가 될 수 있다.

https://github.com/liganega/handson-ml2/blob/master/notebooks/03_classification.ipynb