INDUSTRIAL AI RESEARCH CENTER

어프렌티스 프로젝트

11 주차 – 분류(Classification)

충북대학교 산업인공지능연구센터 김 재영

Dept. of Industrial AI

차례

1. MNIST

INDUSTRIAL AI RESEARCH CENTER

MNIST란?

- MNIST는 손으로 쓴 숫자의 이미지로 이루어진 대표적인 데이터셋 중 하나

MNIST 데이터셋의 주요 특징

- **이미지 구성:** MNIST는 28x28 크기의 흑백 이미지. 각 이미지는 0부터 9까지의 숫자 중 하나를 나타내며, 손으로 쓴 숫자의 이미지.
- **클래스:** 총 10개의 클래스가 있다. 각 클래스는 0부터 9까지의 숫자에 해당.
- **용도:** 주로 숫자 인식 알고리즘을 테스트하고 실험하는 데 사용. 또한, 기계 학습 및 딥러닝 모델을 학습하는 데에도 널리 사용.
- **훈련 집합과 테스트 집합:** 전체 데이터셋은 훈련 집합과 테스트 집합구성. 훈련 집합은 모델을 학습하는 데 사용되고, 테스트 집합은 학습된 모델의 성능을 평가하는 데 사용.

MNIST 데이터셋 다운로드

```
>> from sklearn.datasets import fetch_openml
>> mnist = fetch_openml('mnist_784', as_frame=False)
>> X, y = mnist.data, mnist.target
>> X
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
>> X.shape
(70000, 784)
>> y
array(['5', '0', '4', ..., '4', '5', '6'], dtype=object)
>> y.shape
(70000,)
```

2. 이진 분류기(Binary Classifier) 훈련

INDUSTRIAL AI RESEARCH CENTER

이진분류기란?

- 이진 분류기는 두 개의 클래스 중 하나로 예측하는 모델.
- 어진 입력에 대해 모델이 "Positive" 또는 "Negative" 중 하나를 선택하도록 설계.
- 다양한 분야에서 사용되며, 예를 들어 이메일이 스팸인지 아닌지 예측하거나, 환자가 특정 질병을 가지고 있는지 아닌지를 예측하는 의료 응용 프로그램 등에서 사용

- 이진 분류기의 주요 개념:
  1. True Positive (TP): 모델이 실제 Positive인 샘플을 Positive로 정확하게 예측한 경우.
  2. True Negative (TN): 모델이 실제 Negative인 샘플을 Negative로 정확하게 예측한 경우.
  3. False Positive (FP): 모델이 실제로는 Negative인데 Positive로 잘못 예측한 경우. (오류)
  4. False Negative (FN): 모델이 실제로는 Positive인데 Negative로 잘못 예측한 경우. (오류)

2. 이진 분류기(Binary Classifier) 훈련

이진분류기 사용에 유의할 점

- **클래스 불균형(Class Imbalance):** 클래스의 샘플 수에 큰 차이가 있는 경우, 정확도만을 고려하면 모델의 성능을 잘못 판단할 수 있다. 클래스 불균형 상황에서는 정밀도, 재현율, F1 스코어 등의 성능 지표를 함께 고려하는 것이 중요.
- **성능 지표 선택:** 어떤 성능 지표를 사용할지는 문제의 특성에 따라 다름. 예를 들어, 스팸 메일 감지에서는 정밀도가 더 중요할 수 있고, 의료 진단에서는 재현율이 더 중요할 수 있다. 따라서 문제의 도메인과 목적에 따라 적절한 성능 지표를 선택해야 한다.
- **오차 종류의 중요성:** False Positive와 False Negative의 영향이 문제에 따라 다르다. 어떤 오차가 더 큰 영향을 미치는지를 고려하여 모델을 평가하고 조정해야 한다.
- **데이터 전처리:** 입력 데이터의 특성에 따라 적절한 전처리가 필요. 이상치나 결측치 처리, 스케일링 등이 필요할 수 있다.
- **고려해야 할 다른 모델:** 문제에 따라 선형 분류 모델, 트리 기반 모델, 혹은 신경망 등 다양한 모델 중 어떤 것이 더 적합한지를 고려해야 한다.
- **하이퍼파라미터 튜닝:** 모델의 성능을 향상시키기 위해 하이퍼파라미터를 튜닝해야 한다.
- **모델 해석성:** 모델의 결정이 해석 가능해야 하는 경우, 해석이 쉬운 모델을 선택해야 한다.

2. 이진 분류기(Binary Classifier) 훈련

모델 훈련 최적화 알고리즘

- 훈련 데이터로부터 모델을 학습시키는 것은 모델의 파라미터를 조정하여 주어진 작업에서의 성능을 최대화하거나 손실을 최소화하는 것을 의미. 이 과정에서 사용되는 최적화 알고리즘은 모델의 파라미터를 어떻게 업데이트할지 결정하는 데 사용.
- 기본적으로 최적화 알고리즘의 목표는 손실 함수(Loss Function)를 최소화하는 것. 손실 함수는 모델의 예측과 실제 값 사이의 차이를 나타내며, 이를 최소화하면 모델의 성능이 향상.
- 다양한 최적화 알고리즘이 있지만, 그 중에서도 **경사 하강법(Gradient Descent)**이 가장 널리 사용. 경사 하강법은 다음과 같은 단계로 동작:
  1. **초기화:** 모델의 파라미터를 임의의 값으로 초기화.
  2. **예측:** 초기화된 모델을 사용하여 입력 데이터에 대한 예측을 수행.
  3. **손실 계산:** 예측된 결과와 실제 결과 간의 차이를 나타내는 손실 함수의 값을 계산.
  4. **기울기(Gradient) 계산:** 손실 함수의 기울기를 계산하여 각 파라미터에 대한 손실의 기여도를 알아낸다.
  5. **파라미터 업데이트:** 기울기의 반대 방향으로 파라미터를 업데이트하여 손실을 줄인다. 이 때, 학습률(learning rate)이라는 하이퍼파라미터를 사용하여 업데이트의 크기를 조절.
  6. **반복:** 손실이 충분히 감소할 때까지 2~5 단계를 반복.

2. 이진 분류기(Binary Classifier) 훈련

확률적 경사 하강법(Stochastic Gradient Descent, SGD)

- 경사 하강법의 변형 중 하나로, 매 반복에서 딱 하나의 훈련 샘플을 무작위로 선택하여 그 하나의 샘플에 대한 기울기(gradient)를 계산하고 모델의 파라미터를 업데이트하는 방식.
- 일반적인 경사 하강법은 매 반복에서 전체 훈련 데이터셋을 사용하여 기울기를 계산하는 데 반해, SGD는 각 반복에서 하나의 랜덤한 샘플을 사용하여 기울기를 계산. 이렇게 하면 전체 데이터셋을 사용하는 것보다 계산이 훨씬 빠르고, 특히 대규모 데이터셋에서 효율적으로 학습.
- SGDClassifier는 Scikit-Learn 라이브러리에서 제공하는 확률적 경사 하강법(Stochastic Gradient Descent, SGD) 기반의 선형 분류 모델:

SGDClassifier 이진 분류기 훈련 실행

```
>> from sklearn.linear_model import SGDClassifier
>> sgd_clf = SGDClassifier(random_state=42)
>> sgd_clf.fit(X_train, y_train_5)
>> sgd_clf.predict([some_digit])
array([ True])
```

3. 성능 측정

Cross-Validation (교차 검증)을 사용한 정확도 측정

```
>> from sklearn.model_selection import cross_val_score
>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.95035, 0.96035, 0.9604 ])
```

- cross_val_score 함수에서 scoring 매개변수는 모델의 성능을 평가하는 데 사용할 지표를 지정하는 데 사용
- "accuracy"는 이 매개변수에 사용될 수 있는 평가 지표 중 하나로 정확한 예측의 비율을 나타낸다.

  Accuracy = (정확하게 분류된 샘플 수) / (전체 샘플 수)

3. 성능 측정

Dummy Classifier을 사용한 정확도 측정

```
>> from sklearn.dummy import DummyClassifier
>> dummy_clf = DummyClassifier()
>> dummy_clf.fit(X_train, y_train_5)
>> cross_val_score(dummy_clf, X_train, y_train_5,
   cv=3, scoring="accuracy")
array([0.90965, 0.90965, 0.90965])
```

- "더미 분류기(dummy classifier)"는 매우 간단한 분류 모델로 모든 이미지를 단 하나의 클래스, 즉 가장 빈번한 클래스로 분류하는 분류기
- 이 경우에서는 가장 빈번한 클래스가 음성 클래스로, 이는 숫자 5가 아닌 모든 숫자를 의미

3. 성능 측정

정확도(Accuracy)를 사용한 정확도 측정

- "accuracy"를 평가 지표로 사용하면 모델이 전체 데이터셋에서 얼마나 정확한 예측을 하는지를 측정할 수 있다. 그러나 "accuracy"는 클래스 불균형 문제(두 클래스 간의 샘플 수가 현저히 분포가 다르다는 문제)를 다룰 때는 적합하지 않을 수 있다.
- "accuracy"를 평가 지표로 사용할 때는 데이터의 클래스 분포를 고려하여 모델의 성능을 해석하는 것이 중요.
- 클래스 불균형 문제에 대처할 때는 다른 평가 지표, 예를 들면 정밀도(precision), 재현율(recall), F1 점수(F1 score), ROC 곡선 아래 면적(AUC-ROC) 등을 함께 고려하는 것이 좋다.

3. 성능 측정

오차 행렬(Confusion Matrices)

- 오차 행렬(Confusion Matrix)은 분류 모델의 성능을 평가하는 데 사용되는 중요한 도구로, 모델의 예측 결과와 실제 관측값을 비교하는 표로, 주로 이진 분류(binary classification)에서 사용되며, 네 가지 주요 항목으로 구성:
  1. True Positives (TP): 실제로 양성이며 모델도 양성으로 예측한 경우.
  2. True Negatives (TN): 실제로 음성이며 모델도 음성으로 예측한 경우.
  3. False Positives (FP): 실제로는 음성이지만 모델이 양성으로 예측한 경우(거짓 양성).
  4. False Negatives (FN): 실제로는 양성이지만 모델이 음성으로 예측한 경우(거짓 음성).

- 오차 행렬을 기반으로 계산할 수 있는 주요 요약 평가 지표:
  - 정확도 (Accuracy): (TP + TN) / (TP + TN + FP + FN)
  - 정밀도 (Precision): TP / (TP + FP)
  - 재현율 (Recall) 또는 민감도 (Sensitivity): TP / (TP + FN)
  - 특이도 (Specificity): TN / (TN + FP)
  - F1 점수 (F1 Score): 2 * (정밀도 * 재현율) / (정밀도 + 재현율)

- 평가 지표들을 통해 모델이 어떤 클래스를 얼마나 잘 예측하고 있는지 평가할 수 있으며, 특히 어떤 오류가 더 중요한지에 따라 모델의 조정이 가능.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

3. 성능 측정

confusion_matrix() 함수를 이용한 오차 행렬 계산

```
>> from sklearn.model_selection import cross_val_predict
>> y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)

>> from sklearn.metrics import confusion_matrix
>> cm = confusion_matrix(y_train_5, y_train_pred)
>> cm
array([[53892, 687],
       [ 1891, 3530]])
```

	Predicted 0	Predicted 1
Actual 0	TN	FP type I errors
Actual 1	FN type II errors	TP

완벽한 분류기 경우의 오차 행렬

```
array([[54579, 0],
       [ 0, 5421]])
```

3. 성능 측정

정밀도(Precision)

- 모델이 양성(Positive)으로 예측한 데이터 중 실제로 양성인 비율:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

  - TP: 모델이 실제로 양성인 샘플을 정확하게 예측한 횟수
  - FP: 모델이 양성으로 예측했지만 실제로는 음성인 샘플

- 양성 클래스의 전체 개수 중에서 얼마나 많은 것을 모델이 올바르게 감지하는지는 알 수 없다. 이를 평가하기 위해 정밀도와 함께 재현율 (Recall) 또는 민감도가 사용된다.

3. 성능 측정

재현율 (Recall)

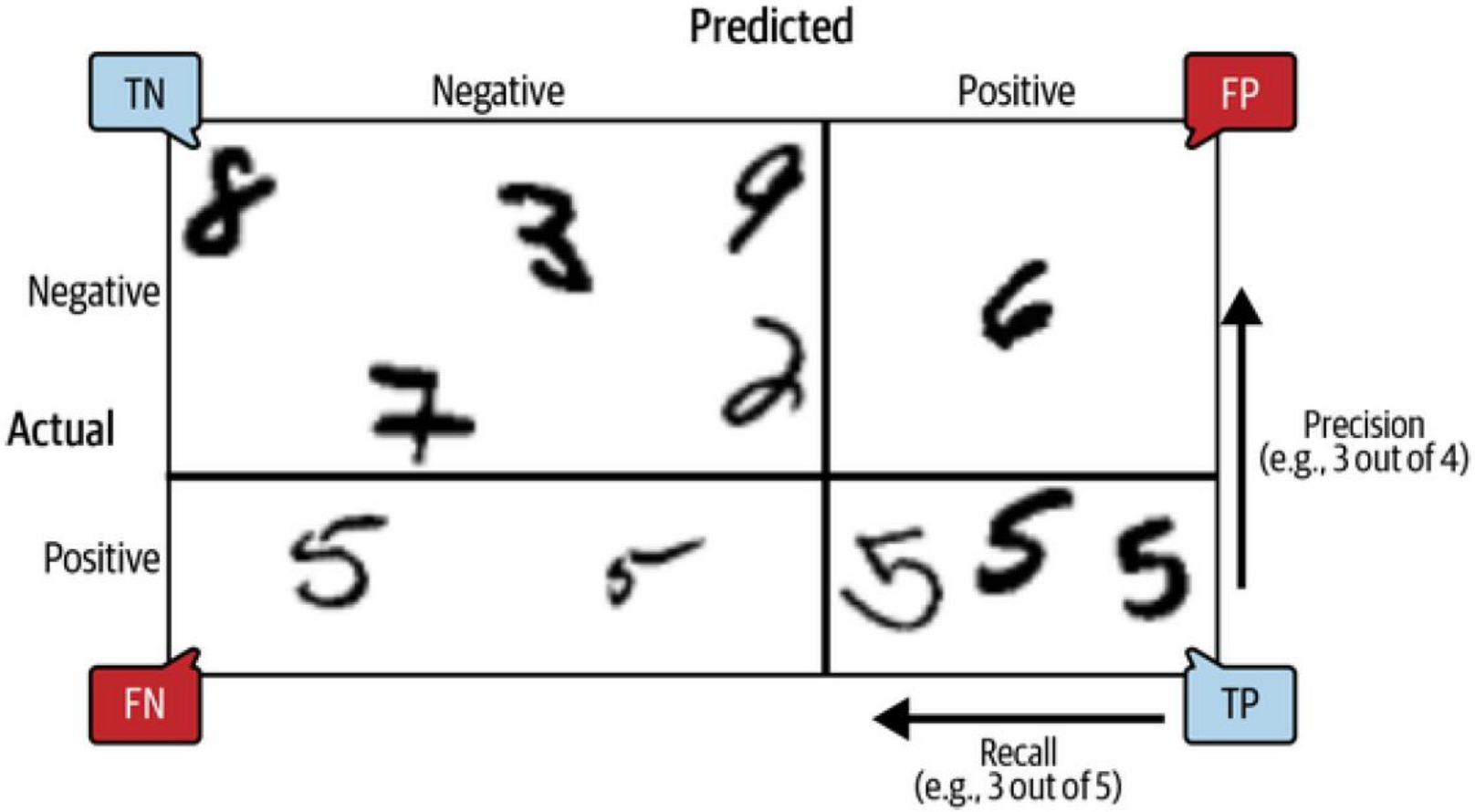
- 모델이 실제 양성 클래스에 속하는 모든 샘플을 얼마나 잘 감지하는지 측정. 다른 용어로는 민감도 (Sensitivity) 또는 진짜 양성 비율 (True Positive Rate, TPR)이라고도 한다:

$$\text{Recall (재현율)} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

  - TP: 모델이 실제로 양성인 샘플을 정확하게 예측한 횟수
  - FN: 모델이 음성으로 예측했지만 실제로는 양성인 샘플

- 재현율은 모델이 실제 양성 중에서 얼마나 많이 찾아냈는지를 나타내므로 높은 재현율은 모델이 실제 양성을 놓치는 경우를 줄여준다. 그러나 높은 재현율을 달성하기 위해 정밀도 (Precision)가 희생될 수 있다. 이것은 정밀도와 재현율 사이의 trade-off 관계를 나타낸다.

정밀도 vs. 재현율


Predicted
TN
Negative
Positive
FP
Negative
Actual
Positive
FN
TP
Precision
(e.g., 3 out of 4)
Recall
(e.g., 3 out of 5)

3. 성능 측정

F1 Score

- 정밀도 (Precision)와 재현율 (Recall)의 조화 평균을 나타내는 분류 모델의 평가 지표 중 하나
- 정밀도와 재현율 간의 trade-off 관계를 고려하여 정밀도와 재현율을 하나의 지표로 결합
- F1 Score는 다음과 같이 정의:

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

- 일반적으로 F1 Score는 불균형한 클래스 분포를 가진 데이터셋에서 모델의 성능을 평가하는 데 유용하며, 이진 분류 모델의 평가에 자주 사용.

3. 성능 측정

F1 Score의 모델 성능 평가 활용

- F1 Score는 정밀도와 재현율, 두 지표 간의 균형을 유지하는 모델이 더 높은 F1 점수를 얻는다. 즉, 모델의 정밀도와 재현율 사이의 균형을 나타낸다.
- 이 균형이 바람직하지는 않은 경우:
  - **예시 1: 아이들을 위한 안전한 비디오 탐지**

    정밀도가 더 중요할 수 있다. 왜냐하면 모델이 안전하지 않은 비디오를 실수로 허용하면 안전하지 않은 콘텐츠가 어린이에게 노출될 수 있어, 모델은 높은 정밀도를 유지하여 안전한 비디오만을 허용하도록 해야 한다.
  - **예시 2: 상점 감시 이미지에서 절도범 탐지**

    재현율이 더 중요할 수 있다. 왜냐하면 모델이 절도범을 실수로 놓칠 경우, 절도사건이 발생할 수 있어 모델은 높은 재현율을 유지하여 가능한 많은 절도범을 잡아내야 한다.
- 각 상황에 따라서 모델의 성능 평가 지표를 선택하는 것이 중요하며, 어떤 지표가 더 중요한지는 문제의 본질과 관련이 있다.

3. 성능 측정

Precision, Recall, F1 score 계산 함수

```
>> from sklearn.metrics import precision_score, recall_score, f1_score

>> precision_score(y_train_5, y_train_pred) # == 3530 / (687 + 3530)
0.8370879772350012

>> recall_score(y_train_5, y_train_pred) # == 3530 / (1891 + 3530)
0.6511713705958311

>> f1_score(y_train_5, y_train_pred)
0.7325171197343846
```

f1_score(y_true, y_pred, average='binary')

- y_true: 실제 타깃 레이블을 포함하는 배열 또는 데이터프레임.
- y_pred: 모델의 예측 레이블을 포함하는 배열 또는 데이터프레임으로, 모델이 얼마나 잘 예측했는지를 측정.
- average (선택적): 다중 클래스 분류 모델의 평균 지표를 계산하는 방법을 지정. 주요 옵션은 'binary' (이진 분류), 'micro' (전체 데이터셋에서 평균 계산), 'macro' (클래스별 평균 계산) 등이 있다. 'binary'는 이진 분류의 경우 주로 사용.

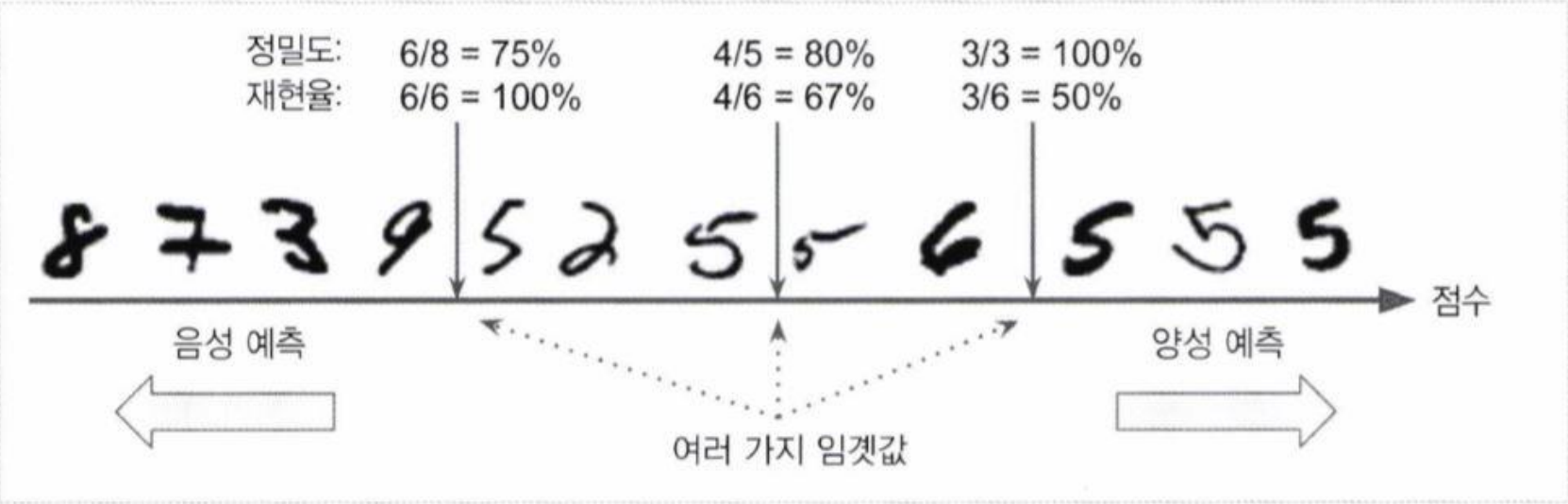
3. 성능 측정

INDUSTRIAL AI RESEARCH CENTER

정밀도/재현율 사이의 Trade-off

SGD(Stochastic Gradient Descent) 분류기의 분류 결정

- SGD 분류기는 각 인스턴스(데이터 포인트)에 대해 결정 함수를 기반으로 스코어(점수)를 계산.
- 이 점수가 어떤 임계값(threshold)보다 크면(또는 임계값 이상이면), 분류기는 해당 인스턴스를 양성 클래스로 할당하고, 그렇지 않으면 음성 클래스(negative class)로 할당
- 임계값을 조정하면 모델의 정밀도 및 재현율을 조절할 수 있다. 높은 임계값은 정밀도를 높이지만 재현율을 감소시킬 수 있으며, 낮은 임계값은 재현율을 높이지만 정밀도를 감소시킬 수 있다.



3. 성능 측정

정밀도/재현율 사이의 Trade-off

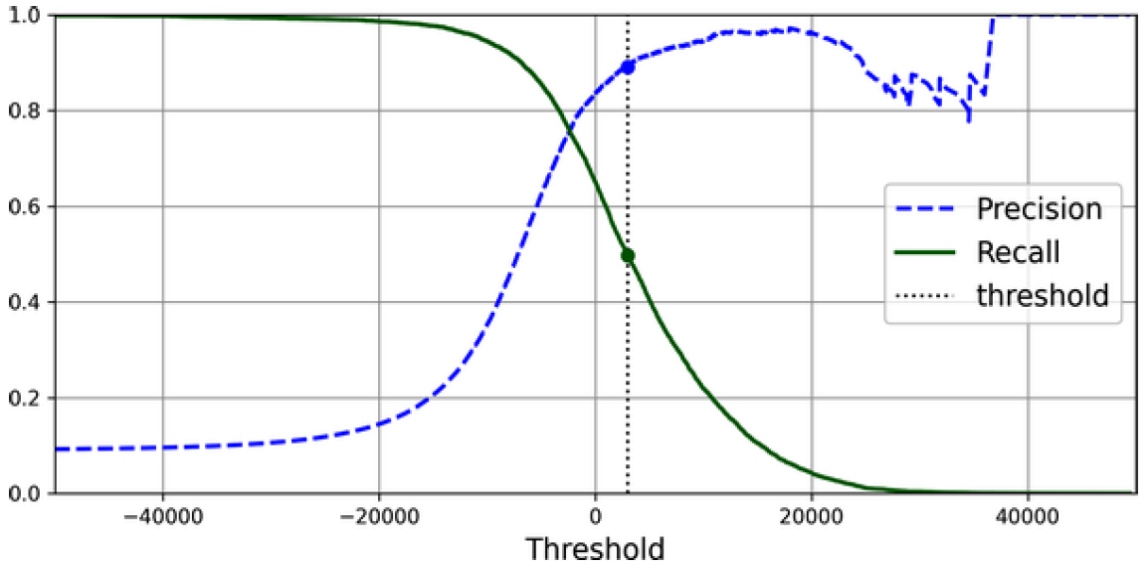
decision_function() 메서드

```
>> y_scores = sgd_clf.decision_function([some_digit])
>> y_scores
array([2164.22030239])
>> threshold = 0
>> y_some_digit_pred = (y_scores > threshold)
array([ True])
```

```
>> threshold = 3000
>> y_some_digit_pred = (y_scores > threshold)
>> y_some_digit_pred
array([False])
```

precision_recall_curve() 함수

```
>> y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
   method="decision_function")
>> from sklearn.metrics import precision_recall_curve
>> precisions, recalls, thresholds =
   precision_recall_curve(y_train_5, y_scores)
>> plt.plot(thresholds, precisions[:-1], "b--", label="Precision",
   linewidth=2)
>> plt.plot(thresholds, recalls[:-1], "g-", label="Recall",
   linewidth=2)
>> plt.vlines(threshold, 0, 1.0, "k", "dotted", label="threshold")
>> plt.show()
```

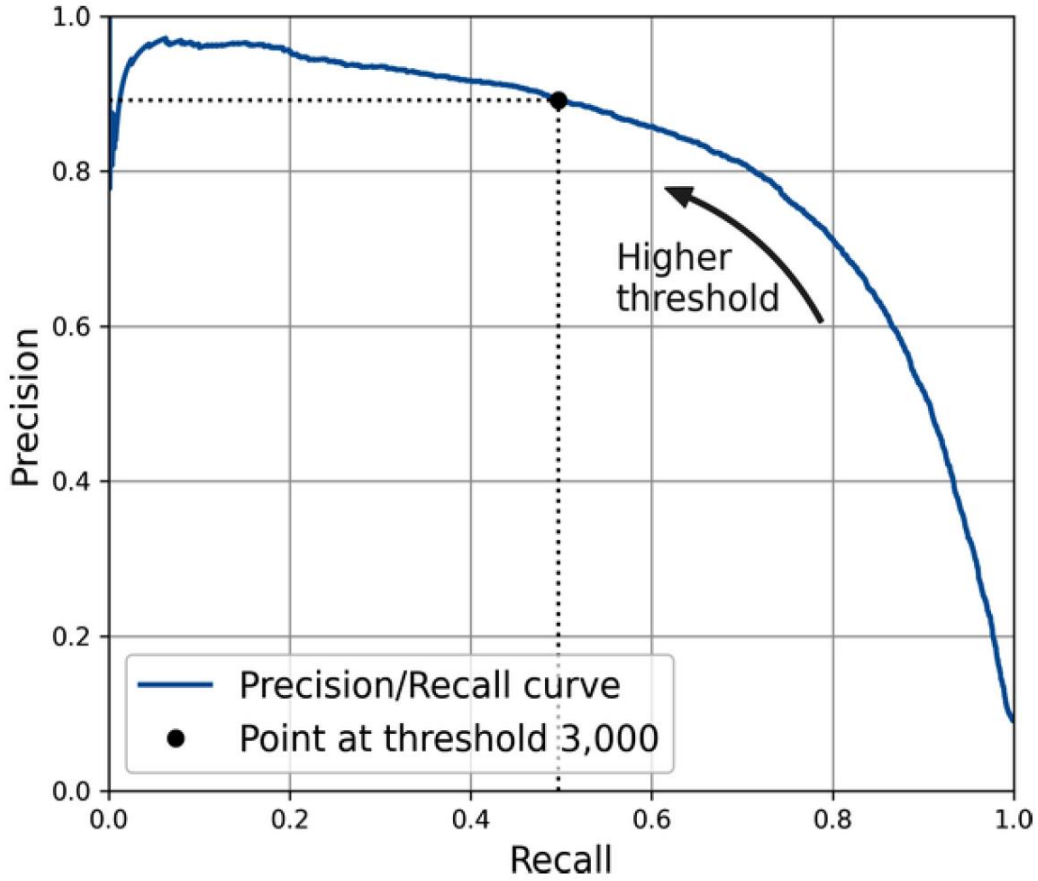


3. 성능 측정

정밀도/재현율 사이의 Trade-off

재현율에 대한 정밀도 곡선

```
>> plt.plot(recalls, precisions, linewidth=2,
   label="Precision/Recall curve")
>> plt.show()
```



3. 성능 측정

수신자 운영 특성 곡선(Receiver Operating Characteristic Curve), ROC 곡선

- ROC 곡선은 이진 분류 모델의 성능을 평가하기 위한 그래프로 정밀도와 재현율 사이의 관계를 시각화
- ROC 곡선은 다음과 같은 주요 개념을 기반으로 작동
  - 진짜 양성 비율 (True Positive Rate - TPR): 실제 양성 샘플을 정확하게 양성으로 분류한 모델의 비율(재현율).

    TPR = TP / (TP + FN)
  - 거짓 양성 비율 (False Positive Rate - FPR): 실제 음성 샘플을 잘못하여 양성으로 분류한 모델의 비율.

    FPR = FP / (FP + TN)
- ROC 곡선은 다양한 임계값(Threshold)에서 모델의 TPR 및 FPR을 그래프로 표시.
- ROC 곡선 아래 영역 (ROC AUC)이 클수록 모델의 분류 성능이 더 좋다고 판단할 수 있다.
- ROC 곡선은 민감도(TPR 또는 재현율)와 특이성(1 - FPR) 사이의 트레이드오프를 나타내므로, 모델의 분류 성능을 평가하는 데 유용.

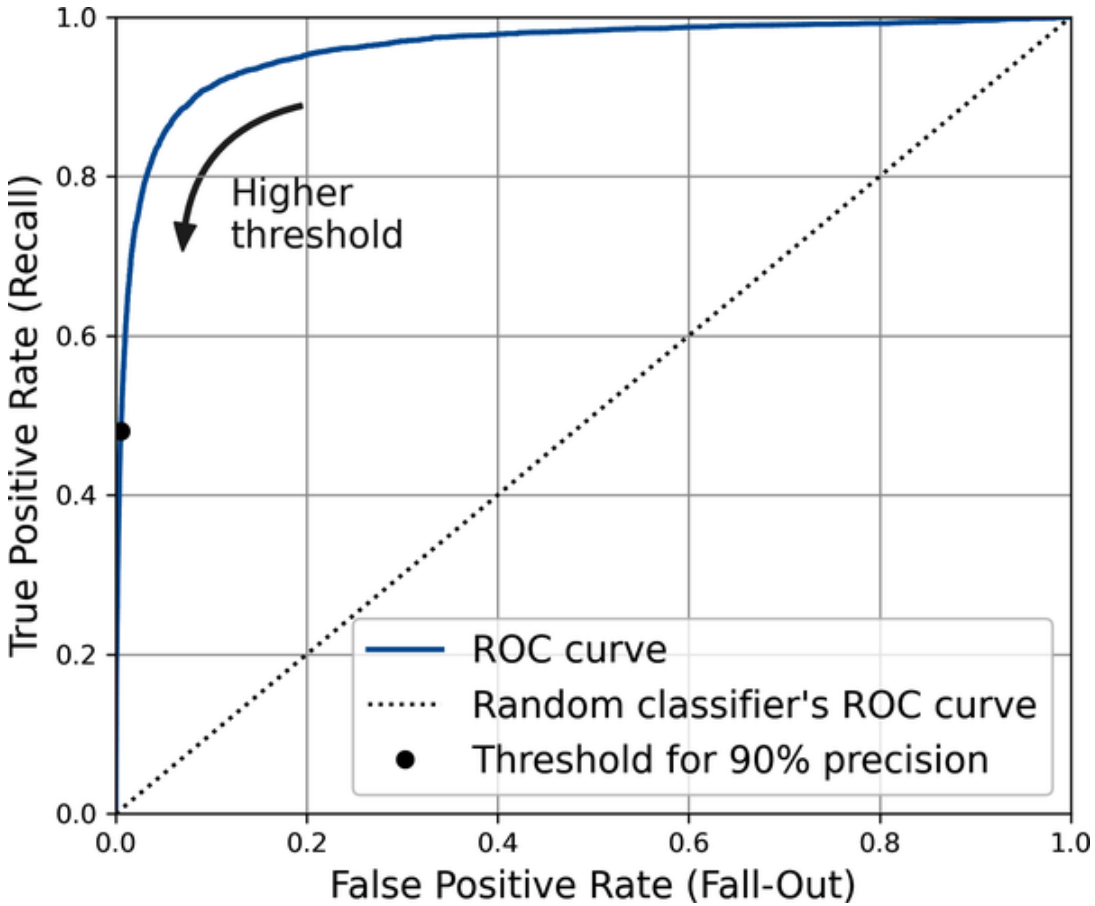
3. 성능 측정

수신자 운영 특성 곡선(Receiver Operating Characteristic Curve), ROC 곡선

roc_curve() 메서드

```
>> from sklearn.metrics import roc_curve
>> fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)

>> idx_for_threshold_at_90 = (thresholds <= threshold_for_90_precision).argmax()
>> tpr_90, fpr_90 = tpr[idx_for_threshold_at_90], fpr[idx_for_threshold_at_90]
>> plt.plot(fpr, tpr, linewidth=2, label="ROC curve")
>> plt.plot([0, 1], [0, 1], 'k:', label="Random classifier's ROC curve")
>> plt.plot([fpr_90], [tpr_90], "ko", label="Threshold for 90% precision")
>> [...] # beautify the figure: add labels, grid, legend, arrow, and text
>> plt.show()
```



4. 다중 분류(Multiclass Classification)

INDUSTRIAL AI RESEARCH CENTER

다중 분류란?

- 세 개 이상의 클래스 중 하나를 예측하는 문제로, 이진 분류 문제와 달리, 둘 이상의 클래스가 존재하는 상황에서 모델이 각 샘플을 올바른 클래스로 분류하는 것이 목표

One-versus-the-rest (OvR) 전략

- OvR 동작 방식:
  1. 클래스마다 이진 분류기 생성
  2. 이진 분류기 훈련
  3. 결정: 모든 이진 분류기를 실행하고, 그 중에서 가장 높은 점수를 갖는 클래스를 선택
- OvR 전략은 각 클래스에 대한 분류 문제를 독립적으로 해결하므로, 이진 분류 문제에 대한 표준 알고리즘(예를 들어, 로지스틱 회귀)을 사용할 수 있다.

One-versus-One (OvO) 전략

- OvR 동작 방식:
  1. 클래스 조합 결정: 클래스가 N개일 때, 가능한 모든 클래스 조합에서 두 개씩 묶어 이진 분류기를 생성. $N \times (N - 1) / 2$ 개의 이진 분류기가 만들어진다.
  2. 이진 분류기 훈련
  3. 결정: 분류할 때, 모든 이진 분류기를 실행하고, 그 중에서 가장 높은 점수를 갖는 클래스를 선택
- OvR 전략은 각 클래스에 대한 분류 문제를 독립적으로 해결하므로, 이진 분류 문제에 대한 표준 알고리즘(예를 들어, 로지스틱 회귀)을 사용할 수 있다.

Thank You!