**Chapter 3. Classification**

1 장에서 우리는 가장 일반적인 감독 학습 과제가 회귀(예측 값)와 분류(예측 클래스)라는 것을 언급

2 장에서는 선형 회귀, 결정 트리, 랜덤 포레스트 등 다양한 알고리즘을 사용하여 주택 값을 예측하는 회귀 작업에 대해 살펴 보았다

이제 우리는 분류 시스템에 관심을 돌릴 것입니다.

1. MNIST
2. Training a binary classifier
3. Performance measures Measuring accuracy using cross-validation Confusion matrix Precision and recall Precision/recall tradeoff The ROC curve
4. Multiclass classification
5. Error analysis
6. Multilabel classification
7. Multioutput classification
8. **MNIST**

이 장에서는 MNIST 데이터 세트를 사용할 것입니다. MNIST 데이터 세트는 고등학생 및 미국 인구 조사국 직원이 수작업으로 작성한 70,000 개의 작은 숫자 이미지 세트입니다. 각 이미지는 그 이미지가 나타내는 숫자로 표시됩니다. 이 세트는 너무 많이 연구되어 종종 기계 학습의 "Hello World"라고 불리며, 사람들이 새로운 분류 알고리즘을 생각할 때마다 MNIST에서 어떻게 작동하는지보고 싶어합니다. 조만간 기계 학습을 배울 때마다 MNIST를 다루게됩니다.

Scikit-Learn에 의해로드 된 데이터 세트는 일반적으로 다음과 같은 사전 구조를가집니다.

* DESCR key : 데이터 세트를 설명
* data key : 인스턴스 당 하나의 행과 feature당 하나의 열이있는 배열을 포함
* target key: 라벨이있는 배열이 포함



70,000 개의 이미지가 있으며 각 이미지가 28 × 28 픽셀이기 때문에 784개의 feature가 있습니다. 각각의 feature는 0 (흰색)에서 255 (검정)까지 한 픽셀의 강도를 단순히 나타냅니다.



예로 36000번째 이미지는 5라는 것을 알 수 있다.



다음은 분류 작업의 복잡성에 대한 느낌을 주기 위해 MNIST 데이터 세트의 몇 가지 이미지



MNIST 데이터 세트는 실제로 훈련 세트 (첫 60,000 이미지)와 테스트 세트 (마지막 10,000 이미지)로 이미 분리되어 있습니다.

훈련 세트를 섞어 봅시다. 이렇게 하면 cross-validation folds가 모두 유사하게 됩니다.(일부 학습 알고리즘은 학습 인스턴스의 순서에 민감하며 행에 유사한 인스턴스가 많이 생성되면 제대로 수행되지 못합니다)



1. **Training a Binary Classifier**

지금 당장 문제를 단순화하고 예를 들어 숫자 5를 식별해보자

이 "5- 탐지기"는 이진 분류 자의 한 예일 뿐이며 5와 5가 아닌 두 클래스를 구별 가능

Scikit-Learn의 SGDClassifier 클래스를 사용

이 분류 기준은 매우 큰 데이터 세트를 효율적으로 처리 할 수 ​​있다는 이점이 있습니다. 이는 부분적으로 SGD가 한 번에 하나씩 교육 인스턴스를 개별적으로 처리(SGD가 On-line 학습에 적합 함을 나타냄)



5 대 전체 대상 클래스 (y\_train\_5)을 사용하여 학습

In 148에서 some\_digit을 36000으로 위에서 고정 부울 대수로 5가 참임을 나타냄

1. **Performance measures**

Classifier를 평가하는 것은 종종 회귀 변수를 평가하는 것보다 훨씬 까다로울 수 있으므로이 장의 많은 부분을이 주제에 사용합니다. 사용할 수 있는 성능 측정 방법은 많다.

**3.1) Measuring accuracy using cross-validation**

모델을 평가하는 좋은 방법은 2장에서 했던 것처럼 cross-validation를 사용하는 것입니다.

때때로 Scikit-Learn이 제공하는 것보다 교차 유효성 검사 프로세스에 대한 통제가 필요합니다. 이러한 경우 직접 상호 유효성 검사를 구현할 수 있습니다.

StratifiedKFold 클래스는 cross\_val\_score () 함수와 거의 동일한 기능을 수행하며 동일한 결과를 인쇄

5를 찾는 분류기



기

다음은 5가 아닌 것을 찾는 분류기



세 분류기 다 정확한 결과가 나오지만 단지 10퍼센트만 5인 것을 찾는 분류기와 90퍼센트의 5가 아닌 것을 찾는 분류기의 성능 차이.....(90퍼센트의 5가 아닌 것을 찾는데 시간을 더 쓴다)

이는 왜 정확도가 일반적으로 분류 기준에 대해 선호되는 실적 척도가 아닌지,

특히 비대칭 데이터 세트를 다루는 경우( 즉, 일부 분류가 다른 분류보다 훨씬 빈번한 경우)의 이유를 보여줍니다.

**3.2) Confusion matrix**

분류 기준의 성능을 평가하는 훨씬 더 좋은 방법은 Confusion matrix을 보는 것입니다

분류A의 인스턴스가 분류B로 분류되는 횟수를 세는 것입니다.

예를 들어, 분류기가 5의 이미지를 3과 혼동 한 횟수를 알고 싶다면, Confusion matrix의 5 번째 행과 3 번째 열을 보는 것

혼돈 행렬을 계산하려면 먼저 예상 집합을 가져야하므로 cross\_val\_predict () 함수를 사용

cross\_val\_score () 함수와 마찬가지로 cross\_val\_predict ()는 K- fold cross-validation 검사를 수행하지만 평가 점수를 반환하는 대신 각 테스트 폴드에 대한 예측을 반환합니다. 즉, 트레이닝 셋의 각 인스턴스에 대해 깨끗한 예측 ( "깨끗한"은 트레이닝 중에 데이터를 본 적이 없는 모델에 의해 예측됨을 의미)을 의미합니다

이제 confusion\_matrix () 함수를 사용하여 혼란 행렬을 얻을 준비가 되었습니다.



* 첫번재 행 : 5가 아닌 분류 53272개의 5가 아닌 것 1307개의 5이지만 아닌 것으로 분류된 것
* 두번째 행 : 5인 분류 1077개의 5가 아니지만 5로 분류된 것 4344개의 5인 것



완벽한 분류기는 True Postives와 True Negatives만을 가집니다.



이것을 분류 자의 정밀도라고합니다(True 의 인스턴스 비율). 분류기 ​(classifier)가 하나의 긍정적 인 예를 제외한 모든 것을 무시할 것이므로 매우 유용하지는 않을 것이다.

따라서 정밀도는 일반적으로 민감도 또는 참 긍정 비율 리콜(TPR)이라는 또 다른 식과 함께 사용됩니다. 이것은 분류기분류기에 의해 올바르게 감지 된 Positives인 인스턴스의 비율입니다



 3.3)Precision and recall Scikit-Learn은 분류 기준식을 계산하는 몇 가지 기능을 제공합니다.



특히 정밀도와 회수율을 F1 점수라는 단일 측정 항목에 결합하는 것이 편리합니다.

F1 점수는 정밀도와 리콜의 조화평균 입니다.

결과적으로 분류기는 리콜과 정확도가 모두 높으면 높은 F1 점수를 얻습니다





F1 점수는 항상 우리가 원하는 값만 출력하는 것은 아니다. 상황에 맞게 precision과 recall 중 중점적으로 생각할 수 있을 것이다.

안타깝게도 두 가지 방법을 동시에 사용할 수는 없습니다. 정밀도를 높이면 리콜이 줄어들고 그 반대의 경우도 마찬가지입니다. 이를 precision / recall tradeoff라고 합니다.

**3.4) Precision/recall tradeoff**

이 단점을 이해하기 위해 SGDClassifier가 분류 결정을 내리는 방법을 살펴 보겠습니다.

각 인스턴스에 대해 결정 함수를 기반으로 점수를 계산하고 점수가 임계 값보다 큰 경우 인스턴스를 양수 클래스에 할당하거나 그렇지 않으면 음수 클래스에 할당합니다.



임계치가 가운데에서 오른쪽으로

Scikit-Learn은 임계 값을 직접 설정할 수는 없지만 예측을 위해 사용하는 의사 결정 점수에 대한 액세스를 제공합니다.

기의 predict () 메소드를 호출하는 대신 각 인스턴스에 대해 점수를 반환하는 decision\_function () 메소드를 호출 한 다음 원하는 모든 임계 값을 사용하여 해당 점수를 기반으로 예측을 수행 할 수 있습니다.



Scikit-learn 0.19.0에서 cross\_val\_score ()함수의 method="decision\_function"의 반환 shape는 (60000,2)이고 y\_train은(60000,1)로 형식이 맞지 않아서 오류값이 나온다. 오류가 생겨 hack을 사용하여 y\_scores[:1]로 변환

이제 이 점수를 사용하여 precision\_recall\_curve () 함수를 사용하여 가능한 모든 임계 값에 대해 정밀도와 호출을 계산할 수 있습니다.

마침내 Matplotlib을 사용하여 임계 값의 함수로 정밀도를 재현하고 호출 할 수 있습니다.



Precision과 Recall 의 상관관계를 그래프로 한눈에 볼 수 있음



리콜에 대한 정밀도를 직접 플롯하는 것입니다.

이제 작업에 대한 최상의 정밀도 / 리콜 트레이드 오프를 제공하는 임계 값을 간단하게 선택할 수 있습니다.

3.5)The ROC curve

수신기 동작 특성 (ROC) 곡선은 이진 분류 자와 함께 사용되는 또 다른 일반적인 도구입니다. Precision / Recall 곡선과 매우 유사하지만 정밀도 대 회귀선을 플로팅하는 대신 ROC 곡선은 True positive rate(TPR) 대 False positive rate(FPR). (탐지율에 대한 진정한 양의 속도(리콜의 다른 이름)를 그립니다.)

ROC 커브를 그리려면 먼저 roc\_curve () 함수를 사용하여 다양한 임계 값에 대한 TPR 및 FPR을 계산해야합니다. 그런 다음 Matplotlib를 사용하여 TPR에 대해 FPR을 플로팅 할 수 있습니다.



다시 말하면, 리콜(TPR)이 높을수록 분류기가 생성하는 오탐지(FPR)이 많아집니다. 점선은 순전히 무작위 분류 자의 ROC 곡선을 나타낸다. 좋은 구분 기호는 가능하면 해당 선에서 멀리 떨어져 있습니다 (왼쪽 상단 모서리 방향).

분류 기준을 비교하는 한 가지 방법은 곡선 아래의 면적 (AUC)을 측정하는 것입니다. 완벽한 분류기는 ROC AUC가 1 인 반면, 순수한 분류기는 ROC AUC가 0.5가됩니다.

Scikit-Learn은 ROC AUC를 계산하는 함수를 제공합니다.



어떤 것을 사용할 지 결정하는 방법을 궁금해 할 수 있습니다. 일반적으로 Positive class가 드물거나 False negatives보다 False positives에 더 관심이있을 때, PR 곡선을 선호해야하며 그렇지 않으면 ROC 곡선을 선호해야합니다.

RandomForestClassifier를 학습하고 ROC 곡선과 ROC AUC 점수를 SGDClassifier와 비교

ROC 곡선을 그리려면 확률이 아니라 점수가 필요합니다.







RandomForestClassifier의 ROC 커브는 SGDClassifier의 ROC 커브보다 훨씬 뛰어납니다. 왼쪽 위 모서리에 훨씬 가깝습니다. 결과적으로 ROC AUC 점수도 상당히 향상되었습니다.

정밀도와 리콜 점수를 측정 해보십시오 98.5 %의 정밀도와 82.8 %의 리콜

지금까지 5와 5가 아닌것에 대한 바이너리 분류기 훈련을 하였다

작업에 적합한 행렬을 선택하고, cross-validation을 사용하여 분류기를 평가하고, 필요에 맞는 정밀도 / 리콜 곡선과, ROC 곡선 및 ROC AUC 점수를 사용하여 다양한 모델을 비교하는 방법을 알았다

이제는 멀티 분류방법에 대해 알아보자

1. **Multilclass classification**

바이너리 분류기가 두 클래스를 구별하는 반면, 멀티 클래스 분류기 (다항 분류기라고도 함)는 두 개 이상의 클래스를 구별 할 수 있습니다.

Random Forest classifiers나 naive Bayes classifiers와 같은 일부 알고리즘은 여러 클래스를 직접 처리 할 수 ​​있습니다.

Support Vector Machine classifiers나 Linear classifiers와 같은 것은 엄격하게 이진 분류기입니다.

그러나 여러 이진 분류기를 사용하여 여러 클래스 분류를 수행하는 데 사용할 수있는 다양한 전략이 있습니다.

* One-versus-All (OvA)(one-versus-the-rest) 10 개의 이진 분류기를 각 숫자 (0- 탐지기, 1- 탐지기, 2- 탐지기 등)에 하나씩 훈련시키는 것입니다. 그런 다음 이미지를 분류하려면 해당 이미지의 각 분류 기준에서 결정 점수를 얻고 분류 기준이 가장 높은 점수를 출력하는 클래스를 선택합니다 대부분의 2 진 분류 알고리즘의 경우 OvA가 선호됩니다.
* One-versus-One (OvO)  모든 쌍의 자릿수에 대해 이진 분류기를 훈련시키는 것입니다. 하나는 0과 1을 구별하고, 다른 하나는 0과 2를 구별하며, 다른 하나는 1과 2를 구별하는 식입니다. 1대1로 구별하는거라 45개의 분류기를 학습해야한다.  OvO의 가장 큰 장점은 각 분류 자만 구별해야하는 두 클래스에 대한 교육 집합 부분에 대해서만 학습하면된다는 것입니다. Support Vector Machine 분류 자와 같은 일부 알고리즘은 훈련 세트의 크기에 따라 가늠할 수 없으므로 OvO 선호

Scikit-Learn은 멀티 클래스 분류 작업에 바이너리 분류 알고리즘을 사용하려고 할 때 이를 감지하고 자동으로 OvA를 실행.

SGDClassifier를 사용



5 대 전체 대상 클래스 (y\_train\_5) 대신 0-9 (y\_train)의 원래 대상 클래스를 사용하여 학습 집합의 SGDClassifier를 교육한 다음 예측

실제로 이것이 사실인지 확인하려면 decision\_function () 함수를 호출하면됩니다. 인스턴스 당 단 하나의 점수를 반환하는 대신 클래스 당 10 개의 점수를 반환합니다.

가장 높은 점수는 실제로 클래스 5에 해당하는 점수입니다.



교차검증 cross\_val\_score () 함수를 사용하여 SGDClassifier의 정확성을 평가



모든 test fold에서 85%이상의 값이 나옴

ScikitLearn이 one-versus-one 또는 one-versus-all을 사용하게 하려면 OneVsOneClassifier 또는 OneVsRestClassifier 클래스를 사용할 수 있습니다

OneVsOneClassifier를 이용한 멀티 클래스 분류기(SGDClassifier를 기반으로)



45개의 분류기가 나온다

RandomForestClassifier를 교육하는 것도 쉽습니다.

분류기가 인스턴스를 여러 클래스로 직접 분류 할 수 있기 때문에 Scikit-Learn은 OvA 또는 OvO를 실행할 필요가 없었습니다.

predict\_proba ()를 호출하여 분류기가 각 클래스에 대해 각 인스턴스에 할당 할 확률 목록을 얻을 수 있습니다.



2장에서 feature scaling에서 입력 값을 단순히 스케일링하면 정확도가 90 % 이상으로 향상됩니다.



1. **Error analysis**

여기서 우리는 당신이 유망한 모델을 발견했다고 생각할 것이고 그것을 향상시킬 방법을 찾고 싶을 것입니다. 이를 수행하는 한 가지 방법은 오류 유형을 분석하는 것입니다.

먼저, 혼란 행렬을 볼 수 있습니다. 이전처럼 cross\_val\_predict () 함수를 사용하여 예측을 수행 한 다음 confusion\_matrix () 함수를 호출해야합니다.



이것은 숫자로 되어있기 떄문에 Matplotlib의 matshow () 함수를 사용하여 혼동 행렬의 이미지 표현을 보는 것이 더 편리합니다.



대각선에 밝은 이미지가 위치하고 있기 때문에 올바르게 분류 되었다는 것을 의미한다.

오류에 관한 이미지를 집중적으로 살펴보자. confusion matrix의 각 값을 해당 클래스의 이미지 수로 나누어 절대 오류수 대신 오류율을 비교할 수 있어야합니다.



오류의 종류를 명확하게 볼 수 있습니다. 행은 실제 클래스를 나타내고 열은 예측 클래스를 나타냅니다.

8과 9의 열이 밝은 이유는 다른 숫자를 8과 9로 분류하지 한다는 것

8과 9의 행이 밝은 이유는 실제 5를 다른 숫자들과 혼동 하는 것

1의 경우 어두운 이유는 대부분의 1이 정확하게 분류된다는 것을 의미

오류는 완벽하게 대칭적은 아니다

개별 오류를 분석하는 것은 분류 자의 역할과 실패 원인에 대한 통찰력을 얻는 좋은 방법이 될 수 있지만 더 어렵고 시간이 많이 걸립니다.



3인데 3으로 예측, 3인데 5로 예측

5인데 3으로 예측, 5인데 5로 예측

분류 기준은 이미지 이동 및 회전에 매우 민감합니다. 혼란을 줄이는 한 가지 방법은 이미지를 전처리하여 이미지가 잘 중심에 있고 너무 회전하지 않도록 하는 것입니다. 이렇게 하면 다른 오류도 줄일 수 있습니다.

1. Multilabel classification

지금까지 각 인스턴스는 항상 하나의 클래스에만 할당되었습니다. 경우에 따라 분류기가 각 인스턴스에 대해 여러 클래스를 출력하도록 할 수 있습니다.

동일한 그림에서 여러 사람을 인식하려면, 그것이 인식하는 사람마다 하나의 라벨을 부착해야 합니다.

분류기가 앨리스, 밥, 찰리의 세 얼굴을 인식하도록 훈련 받았다고 가정

앨리스와 찰리의 사진이 표시되면 [1, 0, 1]

이진 라벨을 여러 개 출력하는 분류 시스템을 다중 라벨 분류 시스템이라고 합니다.

각 숫자 이미지에 대해 두 개의 대상 레이블을 포함하는 y\_multilabel 배열을 만듭니다.

첫 번째 숫자는 숫자가 큰지 (7), 두 번째 숫자는 홀수인지 여부를 나타냅니다.

KNeighborsClassifier 인스턴스 (멀티 라벨 분류를 지원하지만 모든 분류기를 지원하지는 않습니다)를 만들고 여러 타겟 배열을 사용하여 트레이닝 합니다. 이제 예측을 할 수 있으며 두 개의 레이블을 출력한다는 것을 알 수 있습니다.



7보다 큰지 2의 나머지가 1인지(홀수)

다중 라벨 분류 기준을 평가하는 방법은 여러 가지가 있으며

를 들어, 한 가지 방법은 각 개별 레이블 (또는 앞에서 설명한 다른 이진 분류 기준)에 대한 F1 점수를 측정 한 다음 단순히 평균 점수를 계산하는 것입니다.



이것은 모든 레이블이 똑같이 중요하다고 가정. 밥이나 찰리보다 앨리스 사진이 더 많은 경우 앨리스 사진에서 분류 기준 점수에 더 많은 비중을 두는 것이 좋습니다.(각 라벨에 지원과 동일한 가중치 (즉, 그 대상 라벨을 갖는 인스턴스의 수)를 부여하는 것이다. 이렇게 하려면 앞의 코드에서 average = "weighted"를 설정)

1. **Multioutput classification**

마지막 유형의 분류 작업을 Multioutput - multiclass classification (또는 단순히 Multioutput classification)라고 합니다.

이는 각 레이블이 multiclass일 수 있는 Multilabel classification의 일반화 (즉, 두 개 이상의 가능한 값을 가질 수 있음)입니다.

이를 설명하기 위해 이미지에서 노이즈를 제거하는 시스템을 구축해 보겠습니다

입력으로 잡음이 많은 숫자 이미지를 선택, MNIST 이미지처럼 픽셀 강도의 배열로 표현되는 깨끗한 숫자 이미지를 출력

분류 자의 출력은 multilabel(픽셀 당 하나의 라벨)이며 각 라벨은 여러 값 (픽셀 강도 범위는 0에서 255까지)을 가질 수 있습니다.

먼저 NumPy의 randint () 함수를 사용하여 MNIST 이미지를 가져 와서 픽셀 강도에 노이즈를 추가하여 교육 및 테스트 세트를 작성



왼쪽에는 noise 입력 이미지가 있고 오른쪽에는 깨끗한 대상 이미지가 있습니다. 이제 분류기를 훈련시키고 이 이미지를 깨끗하게 만듭니다.



이것으로 분류에 대한 결론을 마칩니다.

분류 작업에 적합한 행렬을 선택하고, 적절한 precision / recall tradeoff를 선택하고, 분류기를 비교하는 방법을 알고, 보다 일반적으로 다양한 작업에 적합한 분류 체계를 구축 할 수 있어야한다.