CHAPTER7. 앙상블 학습과 랜덤 포레스트

- 앙상블
- 투표 기반 분류기
- 배깅과 페이스팅
- 랜덤 패치와 랜덤 서브스페이스
- 랜덤 포레스트
- 부스팅
- 스태킹

7.1 앙상블 (Ensemble)

- 일련의 예측기로부터 예측을 수집하면 가장 좋은 모델 하나보다 더 좋은 예측을 얻을 수 있을 겁니다.
- 일련의 예측기를 앙상블이라고 부르기 때문에 이를 앙상블 학습이라고 합니다.
- 앙상블 방법은 예측기가 가능한 한 서로 독립적일 때 최고의 성능을 발휘합니다.
- 이렇게 하면 매우 다른 종류의 오차를 만들 가능성이 높아 앙상블 모델의 정확도를 향상시킵니다.
- 일반적으로 앙상블 결과는 하나의 예측기를 훈련시킬때와 비교해 편향은 비슷하지만 분산은 줄어듭니다.
- 랜덤 포레스트의 경우는 결정 트리의 앙상블이라 할 수 있습니다.

7.2 투표 기반 분류기

7.2.1 직접 투표 (Hard voting)

- 더 좋은 분류기를 만드는 간단한 방법은 각 분류기의 예측을 모아서 가장 많이 선택된 클래스를 예측하는 것입니다.
- 이렇게 다수결 투표로 정해지는 분류기를 직접 투표 분류기라고 합니다.
- 다수결 투표 분류기가 앙상블에 포함된 개별 분류기 중 가장 뛰어난 것보다도 정확도가 높을 경우가 많습니다. (큰수의 법칙)

7.2.2 간접 투표 (Soft voting)

- 모든 분류기가 클래스의 확률을 예측할 수 있을 때 가정합니다. (predict_proba( ) 메서드가 있을 때)
- 개별 분류기의 예측을 평균 내어 확률이 가장 높은 클래스를 예측하는 방법입니다.
- 확률이 높은 투표에 비중을 더 두기 때문에 직접 투표 방식보다 성능이 높습니다.

7.3 배깅과 페이스팅

- **배깅 (Bagging)** : Train Data에서 중복을 허용(=부트스트랩)하여 샘플링하는 방식을 의미합니다.
- **페이스팅 (Pasting)** : Train Data에서 중복을 허용하지 않고 샘플링하는 방식을 의미합니다.
- 훈련을 마치면 앙상블은 샘플에 대한 예측을 만드는데 분류일 경우는 최빈값, 회귀일 경우는 평균을 계산합니다.
- 예측기는 모두 동시에 다른 CPU 코어나 서버에서 병렬로 학습이 가능하고 예측도 병렬로 수행할 수 있습니다.
- 이런 확장성 때문에 배깅과 페이스팅의 인기가 높습니다.

7.4 랜덤 패치와 랜덤 서브스페이스

- **랜덤 패치 (Random Patch)** : 훈련 특성과 샘플을 모두 샘플링하는 것을 의미합니다.
- **랜덤 서브스페이스 (Random Subspace)** : 훈련 샘플을 모두 사용하여 특성은 샘플링하는 것을 의미합니다.
- 특성 샘플링은 더 다양한 예측기를 만들며 편향을 늘리는 대신 분산을 낮춥니다.

7.5 랜덤 포레스트 (Random Forest)

- 랜덤 포렛트는 일반적으로 배깅 방법 (또는 페이스팅)을 적용한 결정 트리의 앙상블입니다.
- 트리의 노드를 분할할 때 전체 특성 중에서 최선의 특성을 찾는 대신 무작위로 선택한 특성 후보 중에서 최적의 특성을 찾는 식으로 무작위성을 더 주입합니다.
- 이는 결국 트리를 더욱 다양하게 만들고 편향을 손해보는 대신 분산을 낮춰 전체적으로 훌륭한 모델을 만들어냅니다.

7.5.1 엑스트라 트리 (Extra Trees)

- 트리를 무작위하게 만들기 위해 최적의 임곗값을 찾습니다.
- 대신 후보의 특성을 사용해 무작위로 분할한 다음 그 중에서 최상의 분할을 선택합니다.
- 이렇게 극단적으로 무작위한 트리의 랜덤 포레스트를 의미합니다.
- 여기서도 역시 편향이 늘어나지만 대신 분산을 낮추게 됩니다.
- 일반적인 랜덤 포레스트보다 엑스트라 트리가 훨씬 빠릅니다.

7.5.2 특성 중요도

- 특성별 상대적 중요도를 측정할 수 있습니다.
- 특성을 선택해야 할 때 어떤 특성이 중요한지 빠르게 확인할 수 있어 편리합니다.

7.6 부스팅 (Boosting)

- 부스팅은 약한 학습기를 여러 개 연결하여 강한 학습기를 만드는 앙상블 방법을 말합니다.
- 아이디어는 앞의 모델을 보완해 나가면서 일련의 예측기를 학습시키는 것입니다.

7.6.1 에이다 부스트 (Ada Boost, Adaptive Boosting)

- 앞의 모델을 보완할 때 이전 모델이 과소적합했던 훈련 샘플의 가중치를 더 높입니다.
- 그러면 새로운 예측기는 학습하기 어려운 샘플에 점점 맞춰지게 됩니다.
- 예를 들어 Ada Boost 분류기를 만들려면 기반이 되는 첫 번째 분류기를 훈련 세트에서 훈련시키고 예측을 만듭니다.
- 그리고 잘못 분류된 훈련 샘플의 가중치를 상대적으로 높입니다.
- 두 번째 분류기는 업데이트된 가중치를 사용해 훈련 세트에서 훈련하고 다시 예측을 만듭니다.
- 이러한 과정을 반족해 계속 가중치를 업데이트하고 모델의 성능을 향상시킵니다.
- 연속된 학습기법이라 이전 예측기가 훈련되고 평가된 후에 학습할 수 있어 병렬화를 할 수 없습니다. (속도가 낮음)

[예측기 가중치]

$$\alpha_j = \eta log \frac{1-r}{r_j}$$

- η는 학습률 하이퍼파라미터이며, 예측기가 정확할수록 가중치가 더 높아지게 됩니다.
- 만약 무작위로 예측하는 정도라면 가중치가 0에 가까울 것이고 그보다 낮으면 기중치는 음수가 됩니다.

[가중치 업데이트 규칙]

$$\hat{y_j} = y^{(i)} \text{일 때 } W^{(i)} = W^{(i)}$$

$$\hat{y_j} \neq y^{(i)} \text{일 때 } W^{(i)} = W^{(i)} \cdot exp(\alpha_j)$$

- 업데이트 후에 모든 샘플의 가중치를 정규화 합니다.
- 그리고 업데이트된 가중치를 사용해 훈련되고 전체 과정이 반복됩니다.
- 예측을 할 때는 가중치 합이 가장 큰 클라스가 예측의 결과가 됩니다.

[AdaBoost 예측]

$$\hat{y}(x) = argmax \sum_{i=1}^{N} \alpha_j$$

- 여기서 N은 예측기 수를 의미합니다.

7.6.2 그래디언트 부스트 (Gradient Boost)

- Ada Boost처럼 반복마다 샘플의 가중치를 수정하는 대신 이전 예측기가 만든 잔여 오차에 새로 예측기를 학습시킵니다.
- Scikit-learn의 GradientBoostingRegressor를 사용해 앙상블을 훈련시킬 수 있습니다.
- learning_rate 매개변수가 각 트리의 기여 정도를 조절합니다.
- 0.1처럼 낮으면 낮으면 앙상블 Train Data에 학습시키기 위해 많은 트리가 필요하지만 성능은 좋습니다. (축소 규제방법)

7.7 스태킹 (Stacking, Stacked Generalization)

- 앙상블에 속한 모든 예측기 예측을 취합하는 간단한 함수를 사용하는 대신 취합하는 모델을 훈련시킬수 없을까라는 생각에서 출발합니다.
- Train Data를 서브셋으로 나눕니다. 서브셋1은 첫 번째 레이어의 예측을 훈련시키기 위해 사용됩니다.
- 첫 번째 레이어의 예측기를 사용해 서브셋2에 대한 예측을 만듭니다.
- 예측기들이 훈련하는동안 이 샘플들을 전혀 못봤기 때문에 이 때 만들어진 예측은 완전히 새로운 것입니다.
- 이제 홀드아웃 세트의 각 샘플에 대한 예측값들이 만들어 지고 타깃값은 그대로 쓰고 앞에서 예측한 값을 입력 특성으로 사용하는 새로운 Train Data를 만듭니다.
- 블렌더가 새로운 Train Data로 훈련됩니다.