K-최근접 이웃 알고리즘

- K는 가장 가까운 이웃 '하나'가 아니라 훈련 데이터에서 새로운 데이터 포인트에 가장 가까운 'k개의' 이웃을 찾는다는 뜻

지도 학습

1. K-최근접 이웃 알고리즘
   K는 가장 가까운 이웃 '하나'가 아니라 훈련 데이터에서 새로운 데이터 포인트에 가장 가까운 'k개의' 이웃을 찾는다는 뜻

   KNeighborsClassifier

   주어진 데이터 포인트의 최근접 이웃을 찾아 다수결 투표를 통해 해당 데이터 포인트 클래스를 할당합니다. 즉, 주어진 데이터 포인트와 가장 가까운 이웃들의 클래스 중 가장 많은 클래스로 예측한다

   KNeighborsRegressor

   주어진 데이터 포인트의 최근접 이웃을 찾아 해당 이웃들의 평균 또는 가중 평균을 통해 해당 데이터 포인트의 출력 값을 예측

   선형 모델 : 입력 특성에 대한 선형함수를 만들어 예측을 수행

   ➔ 선형 회귀(최소제곱법)
   관찰된 데이터와 모델로부터 예측된 데이터 간의 오차를 최소화하여 모델을 적합시키는 방법이다
   기울기 파라미터는 가중치 또는 계수(coef), 편향 또는 절편 파라미터(intercept_)

   ➔ 리지 회귀

   모델을 단순하게 (계수를 0에 가깝게) 해주고 훈련 세트에 대한 성능 사이를 절충할 수 있는 방법

   Alpha 값을 높이면 계수를 0에 더 가깝게 만들어서 훈련세트의 성능은 나빠지지만 일반화에는 도움을 줄 수 있다

   ➔ 라소
   선형 회귀에 규제를 적용하는데 ridge의 대안으로 Lasso
   일부 계수를 정말 0으로 만들어서 모델을 이해하기 쉽게 하고 이 모델의 가장 중요한 특성이 무엇인지 드러낸다
   Alpha 값을 낮추면 모델의 복잡도는 증가하여 훈련 세트와 테스트 세트에서의 성능이 좋아진다. 그러나 너무 낮추면 규제의 효과가 없어져 과대적합이 된다

   ➔ 분위수 회귀 분석 (백분위에 따른 회귀분석)

➔ 분류용 선형 모델

LinearSCV and LgisticRegression은 회귀 알고리즘이 아니라 분류 알고리즘이다

(SCV = 서포트 벡터 머신

규제의 강도를 결정하는 매개변수 C의 값이 낮아지면 데이터 포인트 중 다수에 맞추려고 하는 반면 높아지면 규제가 감소하여 정확히 분류하려고 한다

L1 규제 (라쏘), L2 규제 (릿지)

➔ 다중 클래스 분류용 선형 모델

똑같이 LinearSCV 나 LogisticRegression 사용

- SGD(확률적 경사 하강법)

  확률적으로 데이터를 뽑아 한 번의 반복당 한 개의 데이터를 사용하여 가중치들을 업데이트 하는 방법

  장점 : 대용량 데이터 사용할 수 있음

  단점 : 튜닝할 하이파라미터가 많고 특성 스케일에 민감하다

2. 나이브 베이즈 분류기

LogisticRegression이나 LinearSVC 같은 선형 분류기보다 훈련속도가 빠르지만, 일반화 성능이 조금 뒤짐

각 특성을 개별로 취급해 파라미터를 학습하고 각 특성에서 클래스별 통계를 단순하게 취합하기 때문

훈련과 예측속도가 빠르며 훈련 과정을 이해하기 쉽다

희소한 고차원 데이터에서 잘 작동하며 비교적 매개변수에 민감하지 않다

3. 결정 트리

결정에 다다르기 위해 예/아니오 질문을 이어 나가면서 학습 (스무고개 느낌)

- 결정 트리의 복잡도 제어하기
  1. 사전 가지치기

     트리 생성을 일찍 중단하는 전략

     Max_depth 로 설정해준다
  2. 사후 가지치기

     트리를 만든 후 데이터 포인트가 적은 노드를 삭제하거나 병합하는 전략
- 결정 트리 분석

  Export_graphviz() 를 통해서 시각화 할 수 있음

  Plot_tree() 함수 사용하면 .dot 파일 만들지 않고 바로 트리를 그릴 수 있음
- 트리의 특성 중요도

  트리를 만드는 결정에 각 특성이 얼마나 중요한지를 평가하는 특성 중요도

4. 결정 트리의 앙상블'

앙상블은 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법

1. 랜덤 포레스트

   과대적합 문제를 회피하는 방법

   기본적으로 조금씩 다른 여러 결정 트리의 묶음

   n_estimators : 생성할 트리의 개수 정함

   몇개의 특성을 고를지는 max_features를 통해서 정할 수 있음

   값을 크게 하면 트리들은 매우 비슷해지고 가장 두드러진 특성을 이용해 데이터에 잘 맞춰짐

   낮추면 트리들은 많이 달라지고 각 트리는 데이터에 맞추기 위해 깊이가 깊어지게 된다

   많은 트리를 사용할수록 더 부드러운 결정경계가 만들어짐

   단일 트리보다 더 넓은 시각으로 데이터를 바라볼 수 있음

2. 그레이디언트 부스팅 회귀 트리

   여러 개의 결정 트리를 묶어 강력한 모델을 만드는 또 다른 앙상블

   이전 트리의 오차를 보완하는 방식으로 순차적으로 트리를 만듬

   무작위성이 없다

   대신 강력한 사전 가지치기 사용

   # Learning rate : 이전 트리의 오차를 얼마나 강하게 보정할 것인지

   # n_estimators : 이 값을 키우면 앙상블에 트리가 더 많이 추가되어 모델의 복잡도가 커지고 훈련 세트에서의 실수를 바로잡을 기회가 더 많아진다

   과대적합을 막기 위해서 트리의 최대 깊이를 줄여 사전 가지치기를 강하게 하거나 학습률을 낮춘다

3. 엑스트라 트리

   랜덤 포레스트와 비슷하지만 후보 특성을 무작위로 분할한 다음 최적의 분할을 찾음

4. 에이다 부스트

   이전의 모델이 잘못 분류한 샘플에 가중치를 높여서 다음 모델을 훈련시킨다

   훈련된 각 모델은 성능에 따라 가중치가 부여된다

5. 히스토그램 기반 부스팅

   데이터의 특성을 이애하기 위해 먼저 히스토그램으로 변환

➔ 이를 위해 입력 데이터를 여러 개의 구간으로 나누고, 각 구간에 속하는 데이터 포인트의 개수를 기록

5. 커널 서포트 벡터 머신(SVM)
   - ■ 선형 모델과 비선형 특성

     직선과 초평면은 유연하지 못하여 저차원 데이터셋에서는 선형모델이 매우 제한적

     선형 모델을 유연하게 만드는 한 가지 방법은 특성끼리 곱하거나 특성을 거듭제곱하는 식
   - ■ 커널 기법

     실제로 데이터를 확장하지 않고 확장된 특성에 대한 데이터 포인터들의 거리 계산
   - ■ SVM

     두 클래스 사이의 경계에 위치한 데이터 포인트들 = 서포트 벡터

     새로운 데이터 포인트에 대해 예측하려면 각 서포트 벡터와의 거리를 측정

     데이터 포인트 사이의 거리 : 가우시안 커널에 의해 계산
     - ◆ SVM 매개 변수 튜닝

       gamma : 하나의 훈련 샘플이 미치는 영향의 범위, 모델의 복잡도에 영향을 줌

       Gamma가 크면 decision boundary는 더 굴곡지고, Gamma가 작으면 decision boundary는 직선에 가깝다

       C : 각 포인트의 중요도를 제한 , 값이 커지면 포인터들이 모델에 큰 영향을 줘 결정 경계를 휘어서 정확하게 분류
     - ◆ SVM을 위한 데이터 전처리

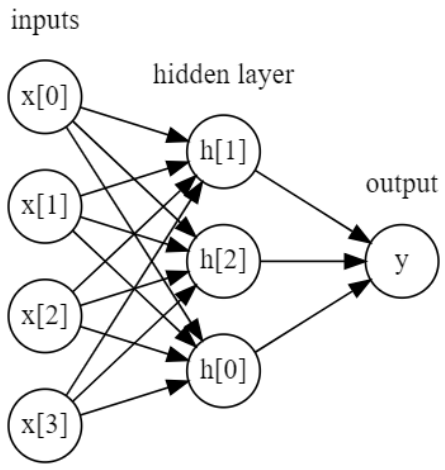
       SVM은 매개변수 설정과 데이터 스케일에 매우 민감

       입력 특성의 범위와 비슷해야 함

       Sol) 특성 값의 범위가 비슷해지도록 조정한다

       ⇨ 모든 특성 값을 평균이 0이되고 단위 분산이 되도록 하거나, 0과 1사이로 맞추는 방법을 많이 사용한다
6. 신경망(딥러닝)

   다층 퍼셉트론 (MLP)



각 은닉 유닛의 가중치 합을 계산한 후 그 결과에 비선형 함수인 렐루나 하이퍼볼릭 탄젠트 적용

Relu : 0이하를 잘라버림

Tanh : 낮은 입력값에 대해서는 -1로 수렴하고 큰 입력값에 대해서는 +1로 수렴

- 신경망 튜닝

  더 매끄러운 결정 경계를 원한다면

  Sol) 은닉 유닛 추가

  은닉층 추가

  Tanh 함수를 사용

  Ex) hidden_layer_size = [10] -> [10, 10] -> [100, 100]

  신경망의 복잡도를 제어하는 방법

  1. 은닉층의 수
  2. 은닉층의 개수
  3. 규제 (alpha)
- 모든 입력 특성을 평균은 0, 분산은 1이 되도록 변형하는 것이 좋다

  데이터 평균을 빼고 표준편차로 나누면

  평균은 0, 표준 편차 1인 데이터로 변환

  Ex) X_train_scaled = (X_train - mean_on_train) / std_on_train
- 훈련 세트와 테스트 세트 사이에는 성능차이가 있을 때

  일반화 성능을 올린다 : 복잡도를 낮춘다

  Ex) alpha : 0.0001 -> 1

7. 분류 예측의 불확실성 추정
   - 결정 함수 = decision_function
     - 반환값의 크기 : (n_samples,)

       다중 분류에서는 (n_samples, n_classes)

       각 열은 각 클래스에 대한 확신 점수
     - 임의의 범위를 갖고 있다
   - 예측 확률 = predict_prob
     - 각 클래스에 대한 확률

       각 행끼리 더하면 1이 나온다 (확률 전체 값)
     - 반환값의 크기 : (n_samples, n_classes)

       각 열은 각 클래스에 대한 확신 확률