Bikers 데이터셋

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mnth(월) | Hr(몇시) | workingday | temp | Weathersit(날씨) | Bikers(얼마나 탔냐) |
| January | 9 | 1 | 0.3 | Clear | 150 |
| February | 12 | 1 | 0.5 | Cloudy | 200 |
| March | 15 | 0 | 0.7 | Light Rain | 50 |
| April | 18 | 1 | 0.9 | Clear | 20 |

노랑색: qualitative(정성적) 🡺 hr은 continuous 형태가 아니라서.

4.6.1

라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

겨울 되면 될수록 자전거 사용량 막 내려감.

봄 가을은 자전거 사용량 증가. 5월 9월 가장 높음.

출퇴근 시간대 자전거 사용량 급증.

심야시간대 음수값. 자전거 사용량 감소.

낮시간대도 꽤 많이 타고.

근데 이 선형 회귀에서 음수값 예측(자전거 사용자가 음수)이 말이 안됨.

그래서 이 방식이 안 맞음.

평균값 작으면 분산도 작아지고, 평균값 크면 표준편차(표준편차 제곱은 분산)도 커짐.

이는 heteroscedasticity(분산 일정하지 않은) 현상.

선형회귀 식.

여기서 선형회귀는 continuous 한 형태로 나와야 하는데, 위의 데이터 bikers 봐보면 단순 count 값. 그래서 이 선형 회귀 모델 자체가 적합하지 않음.

이 문제를 해결하기 위해 폰트, 화이트, 텍스트, 도표이(가) 표시된 사진

자동 생성된 설명 로그 변환 방식 말함.

이렇게 해주면 출력이 음수가 안나오고, heteroscedasticity 문제 해결 가능.

근데 또 애초에 어떤 평균 방식이 아닌 log 씌어져 있어서 직관적이지도 않고, 로그에 0값 들어가면 정의 자체가 안되어서 이것도 막 좋은 방법은 아님.

그래서 포아송 회귀 방식 나옴.

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명포아송 PMF

Λ(평균) = E(Y) = Var(Y) , k !은 팩토리얼.

Y값 커질수록 분산도 같이 커짐.

평균값 5라고 가정.



이용자 0명



이용자 1명



이용자 2명

항상 0 이상 값 나오고, 분산도 이용자에 다라 달라져서 이 방식에 적합.

* Bikers 칼럼처럼 단순 정수 데이터(not continuous)에 매우 적합.

근데 위에는 λ=5로 가정해놓았고, 휘뚜루빠뚜루 바뀔 수 있게끔 하면,



이런식으로 정의 가능.



지수 형태로 변환하면 이렇게 되고,

람다(X\_1, …, X\_p)는 변수 따라 평균값 달라진다는 의미.

각 베타값들은 파라미터 값. 추정 해야 할 계수 값들.

결국 log 씌움으로서 exp 성질 자체가 양수값 나오므로 항상 0 이상 값 나오게끔.

이제 파라미터(베타\_1, 베타\_2, 베타\_3) 값들 추정해보자.

폰트, 텍스트, 라인, 친필이(가) 표시된 사진

자동 생성된 설명



여기서 위의 식은 MLE인데, 이 값 최대 되는거로 파라미터 찾음.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이 표 봐보면

Cloudy -> misty 로 갔을 때 보면, exp(-0.08) = 0.923,

이 말은 날씨 맑을 때의 92.3% 사람이 날씨 흐릴 때 자전거 탐.

선형회귀는 분산이 일정하다는 가정이 들어가있고,

 여기서 입실론 최종 평균값이 0이 되는데, 이게 모든 데이터 포인트에서 입실론 분산 값 일정하다 가정.

포아송은 음수 값이 안나오고 그래서 이런 데이터에서 적합.

선형회귀

텍스트, 폰트, 친필, 라인이(가) 표시된 사진

자동 생성된 설명로지스틱

 포아송 회귀

서로 Y값을 예측한다는 공통점이 있지만 위의 식처럼 차이가 존재.

 선형회귀

 로지스틱

 포아송

각각 결국 선형함수 적으로 예측하려고 이런 함수 씀. 좀 더 각 데이터에 적합하게끔 만들어 놓은 것.

이걸 링크 함수 라고 부름.

5.1

Validation

무작위로 훈련 셋이랑 검증 셋으로 나눔.

훈련 셋으로 학습, 검증 셋으로 예측 성능 평가

보통 MSE로 오류 추정.

스크린샷, 직사각형, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명

파란색이 train 셋, 주황색이 vali 셋.

이런식으로 무작위로 두 부분 나눠서 한다는걸 설명하기 위한 그림.

도표, 그래프, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

각각 MSE 구성 보여주는 그래프인데, 여러 회귀(선형, 이차, 삼차 등등) 훈련셋 학습 후 MSE 평가.

오른쪽 보면 전반적으로 2차 모델이 선형 보다 MSE 낮음.

3차 쪽 보면 살짝 MSE 올라간게 보임.

오른쪽이 총 10번 데이터 무작위로 나눠서 한건데, 각각 다 다른 MSE가 보임. 🡺 변동성 의미.

훈련 셋 검증셋 데이터 분할 어떻게 하느냐에 따라 결과 달라질 수 있다는 의미.

이런식으로 나누면 데이터 적을수록 모델 성능 낮아지고, 그렇기 때문에 교차 검증 사용.

5.1.2

LOOCV: n개 관측값 나누어서 그 중 하나만 검증셋으로, 나머지 n-1개는 학습 데이터로.

 🡺 검증 셋

🡺 train 셋스크린샷, 텍스트, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

이런식으로 쭉 나눠줌.

결국 최종적으로 MSE값 구해주는데, 모든 데이터 반복해서 MSE\_i값 구함.

폰트, 화이트, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명

이걸 평균내서 최종 오류값 냄.

이런 방식이 데이터 하나하나 마다 MSE를 다 구하는거여서 실제값과 일치가 잘 되고,

훈련 셋 부족해서 생기는 편향 문제도 적어짐.

데이터 하나하나 마다 다 구하는 거여서 데이터 분할 때마다 무작위로 썪이는 문제 없고,

LOOCV 여러 번 해도 동일한 결과 나옴.

근데 이렇다 보니 데이터셋 커지면 비용 엄청남.

폰트, 화이트, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

기존 LOOCV 보면 계산 비용 너무 높아서 위의 식으로 비용 대폭 줄이기 가능.

밑의 h\_i는 레버리지. 데이터 i가 예측값에 미치는 영향 측정한 값.

N번 학습하지 않는 방식.

다만, 선형적 방식에만 적용 가능하고, 일반적인 경우에는 기존 LOOCV 써야 함.

5.1.3

폰트, 텍스트, 화이트, 디자인이(가) 표시된 사진

자동 생성된 설명

K -fold cross validation.

K 개 그룹으로 나눠놓고 각 그룹 MSE 계산.

결국 각 데이터 셋 마다의 MSE를 구하는 LOOCV랑, 그룹마다의 MSE를 구하는 k-fold랑 비슷.

똑 같은 데이터셋이라고 하면, k = 10 이라고 했을 때, LOOCV 보다 1/10 만큼의 계산 비용.

도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

파란색 선: 실제 MSE

검은선: LOOCV MSE

주황선: 10-fold CV

엑스표시: 각각의 최적점(MSE 최소값)

왼쪽 그래프는 각 패턴은 비슷한데 실제 MSE랑 비교해보면 좀 과소평가 된게 보임.

가운데는 차수 낮으면 다 좋아 보이는데 점점 차수 높아질수록 과대평가.

* Overfitting 에서는 CV가 더 평가에 유리

오른쪽은 다 좋아보임.

5.1.4

Trade off 얘기

LOOCV:

N을 검증셋으로 두고, n-1이 훈련 셋이므로 편향 값은 상대적으로 작음

But, 모델들 correlation은 높아서 분산값은 높아짐.

훈련셋 거의 동일하므로 각 반복에서 학습된 모델 결과도 거의 동일.

그러므로 상관관계값 높음.

k-fold cv:

훈련셋 크기: (k-1)n / k

K fold 에서 전체 데이터셋 n, 그걸 k번 나눈거니까 이렇게 나옴.

훈련셋이 상대적으로 LOOCV보다는 작아서 대체적으로 편향값은 LOOCV보다는 높음.

# 훈련셋 작으면 학습할 데이터들이 줄어드는 거라서 패턴이 단순화 됨. 그래서 편향 높음.

훈련셋 간 겹침이 상대적으로 적어서 분산은 작게 나옴.

k-fold 에서 k값 높으면 편향 감소, 분산 증가

k값 낮으면 편향 증가, 분산 감소.

5.1.5

분류에서의 교차검증

폰트, 화이트, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명

회귀랑 동일한데, 여기서는 Err\_i로만 바뀜.

Err\_i: 실제 클래스랑 예측 클래스랑 다른지(0값), 같은지(1값) 이런식으로.

텍스트, 도표, 지도이(가) 표시된 사진

자동 생성된 설명

위 그림은 로지스틱 회귀 적용한 결정경계 그림.

Degree는 차수 얼만큼 했는지.

폰트, 텍스트, 화이트, 친필이(가) 표시된 사진

자동 생성된 설명

이건 이차항 로지스틱 회귀인데,

결정경계 전보다 좀 더 curved 해진게 보임.

테스트 에러값은 0.197로 나타났고, 1차항으로 했을 때는 0.201이었는데 이것보다는 좀 더 나아짐.

3차까지 차수 높이면 높일수록 0.197 => 0.16 이런식으로 오차 줄어듦

차수 높이면 높일수록 결정경계 좀 더 유연해져서 성능은 좋아짐.

4차부터는 0.162로 오류 올라갔는데 이건 과적합이 생김.

도표, 라인, 그래프, 경사이(가) 표시된 사진

자동 생성된 설명

갈색: test 에러

검은색: 10-fold CV

파랑: 훈련 에러

왼쪽은 로지스틱에 다항식 차수 늘려가면서, 오른쪽은 KNN의 K값

왼쪽 갈색(test):

왼쪽 보면 차수 높이면 훈련 에러는 줄어듦.

테스트 에러는 u자 형태. 2차쪽 보면 과소적합으로 오류값 다시 늘어남.

8차 다항식 넘어갈수록 overfitting으로 오류 다시 늘어남.

3차 다항식에서 가장 좋아보임(최적점)

왼쪽 검은색(10-fold):

테스트 오류랑 매우 비슷한 형태.

최적점(엑스 표시)이 테스트 에러랑 비슷한 지점(4차 다항식 쪽)에서 나타나는게 보임.

* 여기서는 3~6차 다항식 모델 선택하는게 적합.

KNN쪽:

X축 지점 1쪽 보면 훈련 오류 아예 없음. 근데

테스트 그래프 보면 u자 형태 보임.

# 1/k x축 형태. 1/k커질수록 k 값 작아짐(유연성 증가) 주의하면서 볼 것.

근데 점점 1에 가까워 지면 질수록(k값 작을수록) 과적합 때문에 테스트 오류 증가 하는 형태 보임.

10-fold 보면 test 에러는 완전히 못 따라가서 언더피팅. 그래도 x축 0.1 지점 봤을 때 가장 좋아보임.

* 결국 테스트 오류랑 k-fold가 가장 유사한 꼴이기 때문에 최적 k값 선택에 이게 더 좋다는 말.

5.2

부트스트랩:

Standard error, coefficient(분산) 추정하는데 사용.

X랑 Y 두개의 수익이 있다고 하면,

투자 X랑 Y 두개에 분산 투자.



여기서 알파가 X 투자 비율

1-알파가 Y 투자 비율.

결국 이 위의 식이 최소화 되는 분산 값 찾는게 목적.

폰트, 라인, 친필, 도표이(가) 표시된 사진

자동 생성된 설명

위의 분산값 최소화 위한 알파값 찾는 식.

시그마\_x^2: x 분산값

시그마\_y^2: y 분산값.



각각의 분산값들, 공분산 값들은 알수가 없어서

폰트, 화이트, 번호, 텍스트이(가) 표시된 사진

자동 생성된 설명

이런식으로 과거 데이터로 추정치 계산.

도표이(가) 표시된 사진

자동 생성된 설명

X, Y 데이터가 이런식으로 구성되어 있다고 하면,

이걸로 알파값을 추정할 수 있다.

폰트, 텍스트, 화이트, 도표이(가) 표시된 사진

자동 생성된 설명

총 1000번의 시물레이션 돌려서 알파값 계산했는데

각각 0.576, 0.532, 0.657, 0.651 로 나옴.

그걸 평균내보니 위의 값으로 나왔고, 실제 0.6과 매우 근사하게 나옴.

폰트, 텍스트, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

Standard error값은 이렇게 나옴.

도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

여기서 z라는 데이터가 있으면, 각각 z\*1~3은 부트스트랩 데이터셋 들.

봐보면 3 두번 나오고 2 두번 나오고 막 겹쳐지는게 보임.

그래서 최종적으로 알파 추정값들 생성하고,

뭐 암튼 부트스르랩 데이터셋 여러 개 뽑아낼 수 있음.

폰트, 텍스트, 화이트, 도표이(가) 표시된 사진

자동 생성된 설명

이 식으로 실제 알파값이랑 추정 알파값의 차이 알 수 있음.

B는 부트스트랩 데이터 개수.

a^-\*(시그마 붙어서 나누는 그거)은 모든 부트스트랩 데이터셋 평균 추정값.

A^\*\_r은 각 부트스트랩 데이터셋 계산된 알파 추정값.

**a^\*r – a^-\* 결국 MSE랑 매우 유사한 형태.**

도표, 라인, 평면도, 그래프이(가) 표시된 사진

자동 생성된 설명

왼쪽 그림이 1000개 랜덤해서 뽑은거 샘플링해서 알파값 추정한거.

그걸 히스토그램으로 나타낸거고

가운데가 그걸 부트스트랩 샘플링 해서 1000개 데이터 새로 뽑아낸거.

분홍색 세로 선이 참값인데

왼쪽 그림 가운데 그림 거의 일치하는게 보임.

오른쪽 박스플랏도 거의 비슷하고.

부트스트랩 SE(standard error) 계산해봤는데 0.087 나옴.

아까껀 0.083 나왔는데 거의 일치하는게 보임.

* 결국 부트스트랩 쓰는 이유가 데이터 부족할 때도 쓸 수 있고, 추정 데이터의 변동성(얼마나 안 맞는지) 평가(표준오차) 하기 위해서 쓰임.