머신러닝2 과제 2025

(ver 1.3)

25. 5.30

학번 : 1958015

이름 : 박시형

Preview

>이전의 작업 요약

(ver 1.2)

1. **데이터 전처리**
   - process1_result.csv 사용, 피처: Sex, SibSp, Parch, Embarked, TicketNumeric
   - 결측치 처리: Age → Pclass×Sex별 중앙값; Embarked → 최빈값; Cabin 제외

---

2. **Logistic Regression**
   - instance SGD(learning_rate=0.001) + Early Stopping → 최적 에폭스 = 9
   - 최적 k-fold = 18,
   - test1 결과: Acc 78.35%, F1 70.07%

---

3. **Decision Tree**
   - Gini 불순도 기반 분할, max_depth 교차검증 (k=2→30 탐색)
   - 최적 k-fold = 16, max_depth = 15
   - test1 결과: Acc 78.69%, F1 70.64%

---

4. **KNN**
   - 교차검증을 통한 k 탐색 (n_split(k-fold) 2→50 → 35–45 구간에서 수렴)
   - 최적 k-fold = 37, k = 20
   - test1 결과: Acc 79.79%, F1 69.70%

[PCA]

>고차원 데이터(차원이 많은 데이터)가 있을 때,

차원을 줄이기 위해 사용하는 방법론

분산이 가장 큰 방향(축)을 찾고,

새 축 위로 데이터들을 투사시켜 상위 k개의 주성분만 남긴다.

(분산만을 사용, 공분산 행렬의 고유벡터들이 새 축이 되는 것이고, 이 고유값이 분산 크기를 나타낸다.)

보편적 관점

> 분산을 구하고, 누적 분산이 최대가 되는 지점을 선택하여 고유값을 이용해 차원을 줄인다.

pdf (MSE 관점)

> 보편적으로 고유치가 큰 방향을 선택했다는 것 = 원본 분산이 크고, 재구성 오차가 큰 비중을 차지하는 정보축을 최대한 살렸다는 뜻이다.

따라서 mean square error를 최소화하는 방식으로 사용한다.

두 방식의 수학적 차이는 없고,

보편적 분산 관점에서의 Xi를 k차원으로 투사*복원할 때 생기는 재구성 오차를 최대한 줄이기 위해

1/n sigma i || Xi - x^i || ^2 를 직접 최소화 하기 위해 MSE(mean square error)의 관점을 사용한다.

mse의 관점에서 차원을 축소하면 "복원 오차가 최소" 라는 최적성을 명시적으로 보장하기 때문

- 연산 효율(covariance 생략, direct SVD)

- 수치 안정성

- Compression 최적성 보장

축을 바꾸는 이유

- 원본 feature 축이 서로 직교하지 않고,

  의미적으로도 중복된 정보가 많을 수 있으므로

서로 직교하고 분산 순으로 정렬되어 있기 때문에 상위 몇개 만으로도 원본 정보의 대부분을 담을 수 있다.

따라서,

PCA로 차원을 축소하고,

이 결과에 대해

logistic, knn, dicision tree를 진행하는 것.

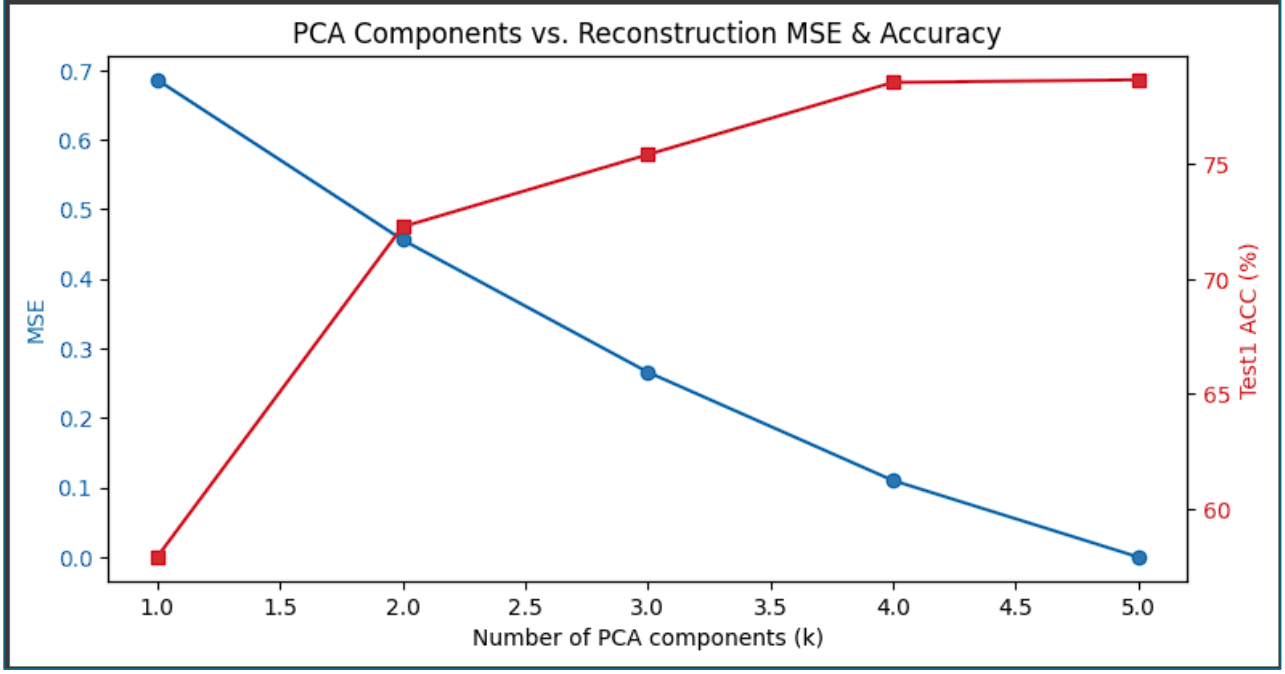
1. 데이터 zero centering
2. 공분산 행렬 계산
3. 고유분해
4. 고유값 내림차순 정렬
5. 상위 k주성분(고유벡터) 선택
6. 저차원 투사
7. 데이터 복원
8. 재구성 오차(MSE) 계산

이 값을 최소화하는 Vk = 분산이 큰 축 순 으로 고른 주성분이라는 결론이 나온다.

이 순서대로 진행해보겠다.

(log)

```
k= 1 → MSE=0.6858, Test1 ACC=57.91%
k= 2 → MSE=0.4559, Test1 ACC=72.28%
k= 3 → MSE=0.2658, Test1 ACC=75.42%
k= 4 → MSE=0.1100, Test1 ACC=78.56%
k= 5 → MSE=0.0000, Test1 ACC=78.68%
```



K = 피쳐의 차원

Pca를 통해 차원을 축소함에 따라, test1에 대한 acc가 점점 내려가는 것을 보이고 있고, Mse 또한 점점 커지고 있다.

따라서 차원축소를 최소한으로 하는 **k의 값을 4**로 지정한다.

(k=4임에 대해 log, knn, dt에서 동일한 조건이어야 함)

K=4일 때, log에서

Test1 : 78.56

Test2 : 76.794

F1 score(test1) : 70.57

(knn)

K=4일 때, knn에서

Test1 : 80.58

Test2 : 77.272

F1 score(test1) : 71.59

(dt)

K=4일 때, dt에서

Test1 : 94.16

Test2 : 72.009

F1 score(test1) : 92.02

[LDA]

>pca와 마찬가지로 차원을 축소하는 방법론.

그 방식에 있어서 차이가 있다.

축을 어떻게 골라야 하는가에 대해 고민한 방법.

클래스A와 B가 서로 최대한 멀리 떨어져 있으면서, 같은 클래스끼리 뭉쳐보일 수 있는지에 대해서 축을 선택한다.

이때, 단순한 거리가 가장 큰 것으로 축을 잡는다면, 구분되지 않는 경우가 있기 때문에

두 분산의 비율

J(w) = 클래스 간 분산 / 클래스 내 분산

을 최대화하는 방향을 찾아 그것을 축으로 한다.

(따라서 FLD는 binary한 discriminant라고 할 수 있다.)

즉, 하나의 축만을 가진다는 것.

이 축이 가장 분리도가 큰 축인 것.

고유치를 통해 가장 분리가 잘 되는 축을 찾은 뒤에,

그 축에 대해서 고유벡터를 구하고

고유벡터에 대해 모든 데이터들을 투사시킨다.

1. 데이터 중앙화*스케일링
2. 클래스별 평균 벡터 계산
3. 클래스 내 산포 행렬 Sw
4. 클래스 간 산포 행렬 Sb
5. 판별 벡터 w 계산 최대화
6. 판별 벡터 w 정규화
7. 새로운 축에 투사

(log)

Test1 : 78.90

Test2 : 77.033

F1 score(test1) : 71.08

(dt)

Test1 : 92.14

Test2 : 67.224

F1 score(test1) : 89.14

(knn)

Test1 : 79.46

Test2 : 77.511

F1 score(test1) : 70.72

(conclusion)

PCA 사용 : knn의 test1, test2 개선 (acc상승, f1score 하락)

FLD 사용 : DT의 f1score 개선, knn의 test2 개선

Next step

>edit k-fold, apply to pca, FLD

Have to learn MLP