Model (2)

Supervisor

Naive Bayes (나이브 베이즈 분류)

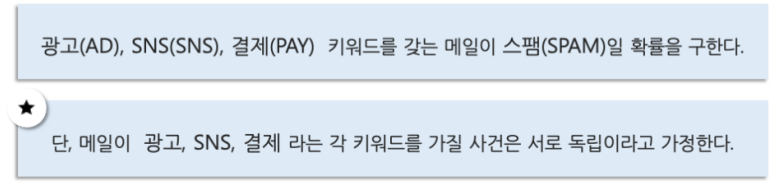
: 데이터의 특징을 가지고 각 레이블에 속할 확률 (조건부 확률)을 계산해 분류하는 방법

- “Naïve” = 데이터의 특징이 모두 상호 독립적이라는 가정 하에 확률 계산을 단순화함
- “Bayes”: 베이즈 정리

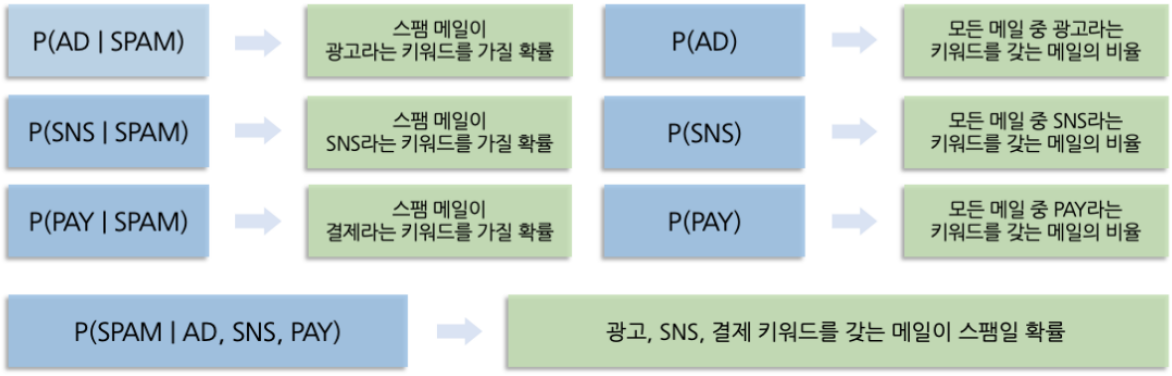
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

적용 예시: 받은 메일이 스팸 메일인지 아닌지 판단하는 분류기

> 수신한 어떤 한 메일의 내용을 키워드로 추출했을 때, 그 결과가 “광고”, “SNS”, “결제” 라고 하자. 이 키워드를 가지고 해당 메일이 스팸 메일인지 아닌지를 판단하는 분류기를 만들 때 나이브 베이즈 분류를 사용할 수 있다.



$$P(SPAM \mid AD, SNS, PAY) = \frac{P(AD \mid SPAM) \cdot P(SNS \mid SPAM) \cdot P(PAY \mid SPAM) \cdot P(SPAM)}{P(AD) \cdot P(SNS) \cdot P(PAY)}$$



스팸으로 분류될 확률이 임계치 이상이면 해당 메일을 스팸으로 분류한다!

코드예시

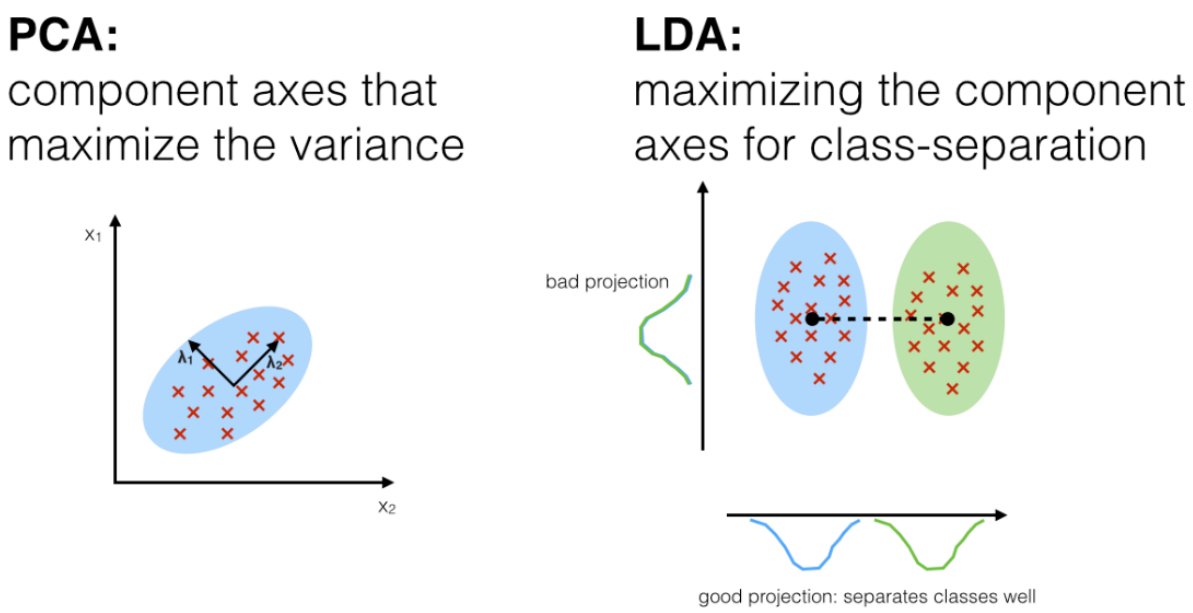
```
from sklearn.feature_extraction.text import CountVectorizer #BoW로 만들어줌
from sklearn.feature_extraction.text import TfidfTransformer #TF-IDF 계산
from sklearn.naive_bayes import MultinomialNB # 다항분포 나이브 베이즈 모델
from sklearn.metrics import accuracy_score #정확도 계산

dtmvector = CountVectorizer() #문서단어행렬
X_train_dtm = dtmvector.fit_transform(newsdata.data)
```

```
#성능 개선 위해 TF-IDF 가중치 적용한 TF-IDF 행렬로 변환
tfidf_transformer = TfidfTransformer()
tfidfv = tfidf_transformer.fit_transform(X_train_dtm)

mod = MultinomialNB()
mod.fit(tfidfv, newsdata.target)

newsdata_test = fetch_20newsgroups(subset='test', shuffle=True) #테스트 데이터 갖고오기
X_test_dtm = dtmvector.transform(newsdata_test.data) #테스트 데이터를 DTM으로 변환
tfidfv_test = tfidf_transformer.transform(X_test_dtm) #DTM을 TF-IDF 행렬로 변환

predicted = mod.predict(tfidfv_test) #테스트 데이터에 대한 예측
print("정확도:", accuracy_score(newsdata_test.target, predicted)) #예측값과 실제값 비교
```

Linear Discriminant Analysis (선형판별분석)

: 데이터 분포를 학습해 **결정경계 (decision boundary)** 를 만들어 데이터를 분류하는 방법

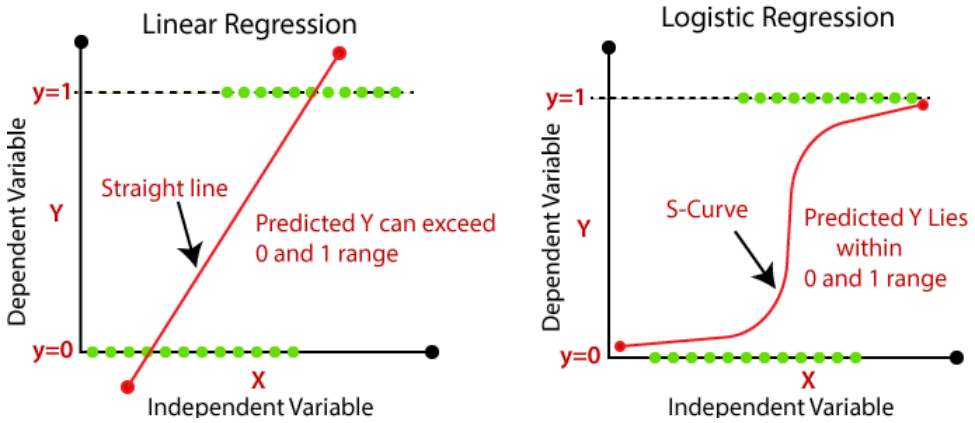
차원축소



- PCA (unsupervised): 축소하고자 하는 데이터의 변동성이 가장 큰 방향으로 축을 설정해 데이터 표현
- LDA: 분류할 수 있게 분별 기준을 최대한 유지하며 축 설정
- PCA 후 LDA 수행하기도 함
- 클래스 당 샘플 수가 적으면 오히려 PCA가 LDA보다 성능 좋기도 함

Logistic Regression (로지스틱회귀)

: 회귀를 사용하여 어떤 데이터가 어떤 범주에 속할 확률을 0~1의 값을 예측 & 해당 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류하는 방법



- 선형회귀는 음과 양의 방향으로 무한대까지 뻗어 가기 때문에 로지스틱 회귀 사용

> 💡 로지스틱 회귀 단계
>
> 1. 모든 속성(feature)들의 계수(coefficient)와 절편(intercept)을 `0`으로 초기화한다.
> 2. 각 속성들의 값(value)에 계수(coefficient)를 곱해서 **log-odds**를 구한다.
> 3. **log-odds**를 **sigmoid 함수**에 넣어서 `[0,1]` 범위의 확률을 구한다.

- odds = (사건이 발생할 확률)/(사건이 발생하지 않을 확률)
- dot product 방식으로 log odds 구함 (연산은 numpy의 `np.dot()` 으로 처리)

Information Theory (정보 이론)

- Information = '놀람의 정도'; 예상치 못한 데이터에 더 높은 가치를 매김

> "정보 이론의 핵심은 잘 일어나지 않는 사건 (unlikely event)의 정보는 자주 발생할만한 사건보다 정보량이 많다고 (informative) 하는 것"

KL Divergence

KL divergence = "놀라움"

- 두 확률분포가 가까웠다는 가정 하에, 가깝지 않다면 "놀라운 일" 이고 KL divergence는 높은 값을 갖게 됨.
- 분류문제에서 cross-entropy를 최소화, 즉 KL divergence를 최소화하면서 NN 학습

$$D_{KL}(p||q) = \sum_{i=0}^{n} p(x_i)\log(p(x_i)) - \sum_{i=0}^{n} q(x_i)\log(q(x_i))$$

Cross Entropy

: 분류 모델이 얼마나 잘 수행되는지 측정하기 위해 사용되는 지표

- 실제 분포 q에 대하여 알지 못하는 상태에서, 모델링을 통해 구한 분포인 p를 통하여 q를 예측하는 것. p와 q 사이의 차이를 측정함

Cross Entropy = Entropy + KL Divergence

$$H(p,q) = H(p) + D_{KL}(p||q) = -\sum_{i=0}^{n} p(x_i)\log(q(x_i))$$

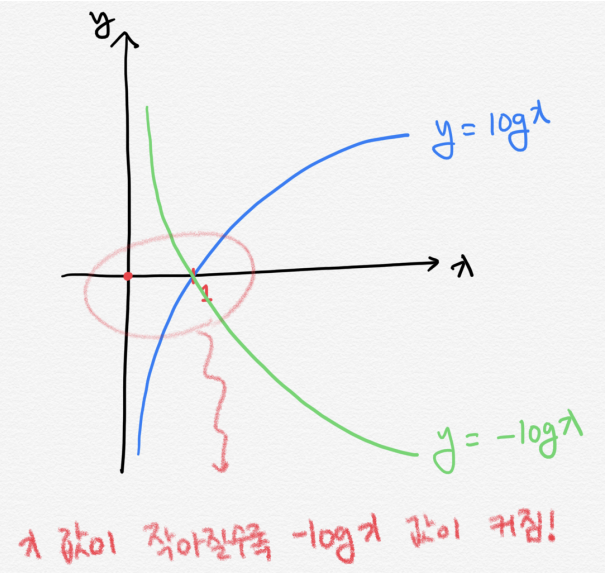
Shannon Entropy (=정보 엔트로피)

: 모든 사건 정보량의 기대값

- 엔트로피 = 불확실성 (**uncertainty**)의 측정
  - 불확실성 = 어떤 데이터가 나올지 예측하기 어려운 경우

$$H(x) = -\sum_{i=1}^{n} p(x_i) logp(x_i) = \sum_{i=1}^{n} p(x_i)\,(-logp(x_i))$$

- 이때 $p(x_i)$의 총합은 1



비교 예시

1) 동전을 던졌을 때 (앞/뒷면이 나올 확률은 모두 1/2)

$$
\begin{aligned}
&-[p(X=0)logp(X=0)+p(X=1)logp(X=1)] \\
&=-[0.5log0.5+0.5log0.5] \\
&=-[0.5*-0.6931471805599453*2] \\
&=0.693147180559945
\end{aligned}
$$

2) 주사위를 던졌을 때 (각 6면이 나올 확률은 모두 1/6)

= 1.79

⇒ 무엇이 나올지 알기 어려운 주사위의 엔트로피가 더 높음

확률이 낮을수록 어떤 정보일지는 불확실하게 되고, 우리는 이것을 “정보가 많다”, “엔트로피가 높다”고 표현한다.