지도학습 : 회귀, 추천

이 건 명
충북대학교 소프트웨어학과

인공지능 : 튜링 테스트에서 딥러닝까지

학습 내용

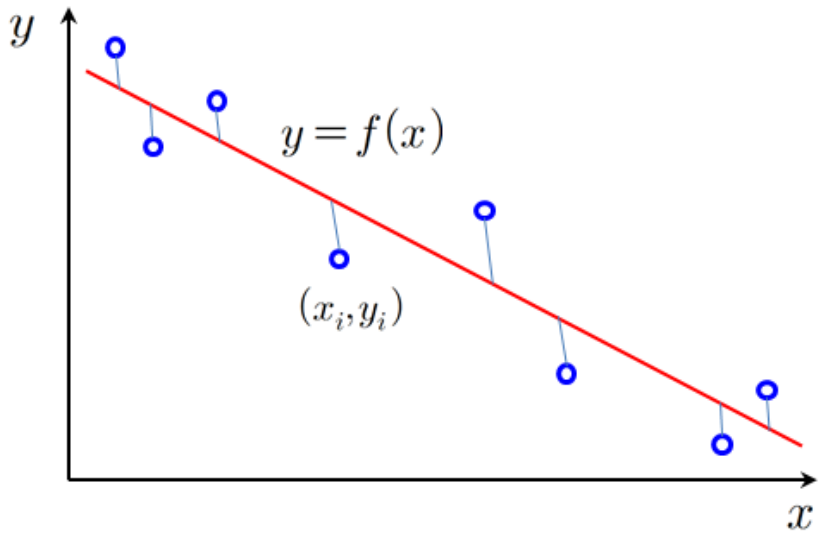
- 회귀 문제의 특성에 대해서 알아본다.
- 로지스틱 회귀 문제의 특성에 대해서 알아본다.
- 편향-분산 트레이드오프에 대해 알아본다.
- 추천 문제의 특성과 전략에 대해서 알아본다.

1. 회귀

❖ **회귀** (regression)

- **학습 데이터에 부합**되는 **출력**값이 **실수**인 함수를 찾는 문제

$$f^*(x) = \arg\min_f \sum_{i=1}^{n} (\boldsymbol{y_i - f(x_i)})^2$$



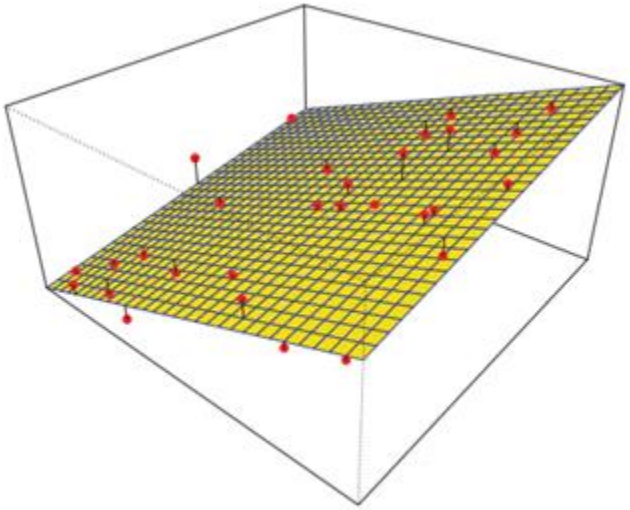
회귀

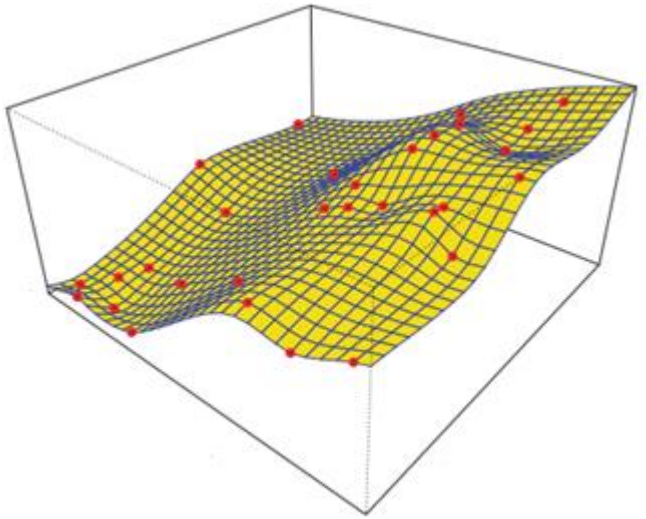
❖ **회귀** (regression) – cont.

- **성능**
  - **오차** : 예측값과 실제값의 차이
    - 테스트 데이터들에 대한 (예측값 – 실제값)^{2}의 평균 또는 평균의 제곱근

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

  - **모델**의 종류(**함수의 종류**)에 영향을 받음





회귀

❖ **경사 하강법**(gradient descent method)

- 오차 함수 E의 **그레디언트**(gradient) 반대 방향으로 조금씩 움직여 가며 최적의 파라미터를 찾으려는 방법
- **그레디언트(gradient)**
  - 각 파라미터에 대해 편미분한 벡터

$$E = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y_i} - \boldsymbol{f}(\boldsymbol{x_i}))^2$$

$$f(x) = ax + b$$

$$\nabla E = \left(\frac{\partial E}{\partial a}, \frac{\partial E}{\partial b}\right)$$

- 데이터의 입력과 출력을 이용하여 각 파라미터에 대한 그레디언트를 계산하여 **파라미터**를 **반복적**으로 **조금씩 조정**

$$a \leftarrow a - \square? \frac{\partial E}{\partial a}$$

$$b \leftarrow b - \frac{\partial E}{\partial b}$$

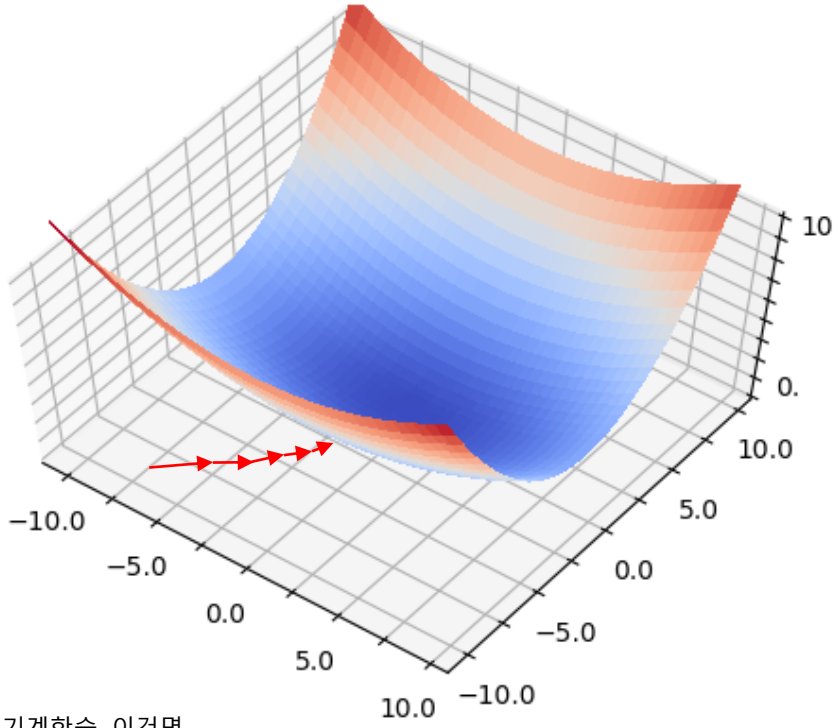
회귀

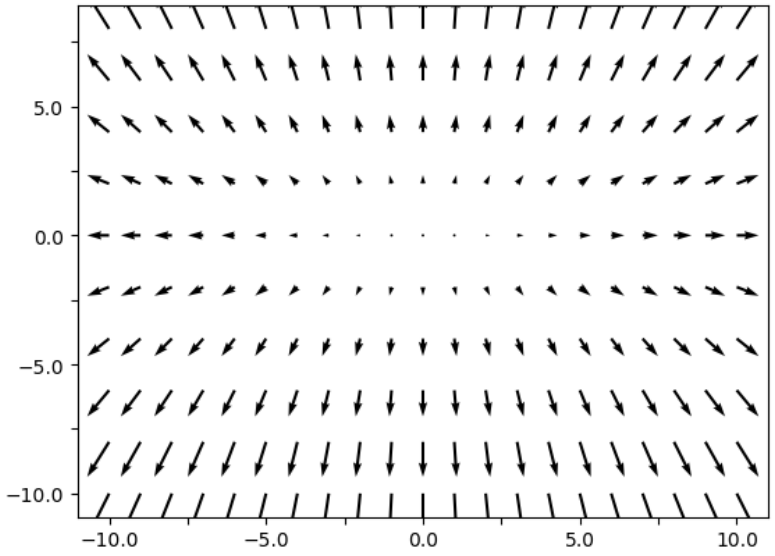
❖ **경사 하강법 – cont.**

- **학습 데이터에 부합**되는 **출력**값이 되도록 **파라미터 변경**하는 일

$$E = \sum_i (y_i - f(x_i))^2$$

$$\nabla E = \left(\frac{\partial E}{\partial a}, \frac{\partial E}{\partial b}\right)$$

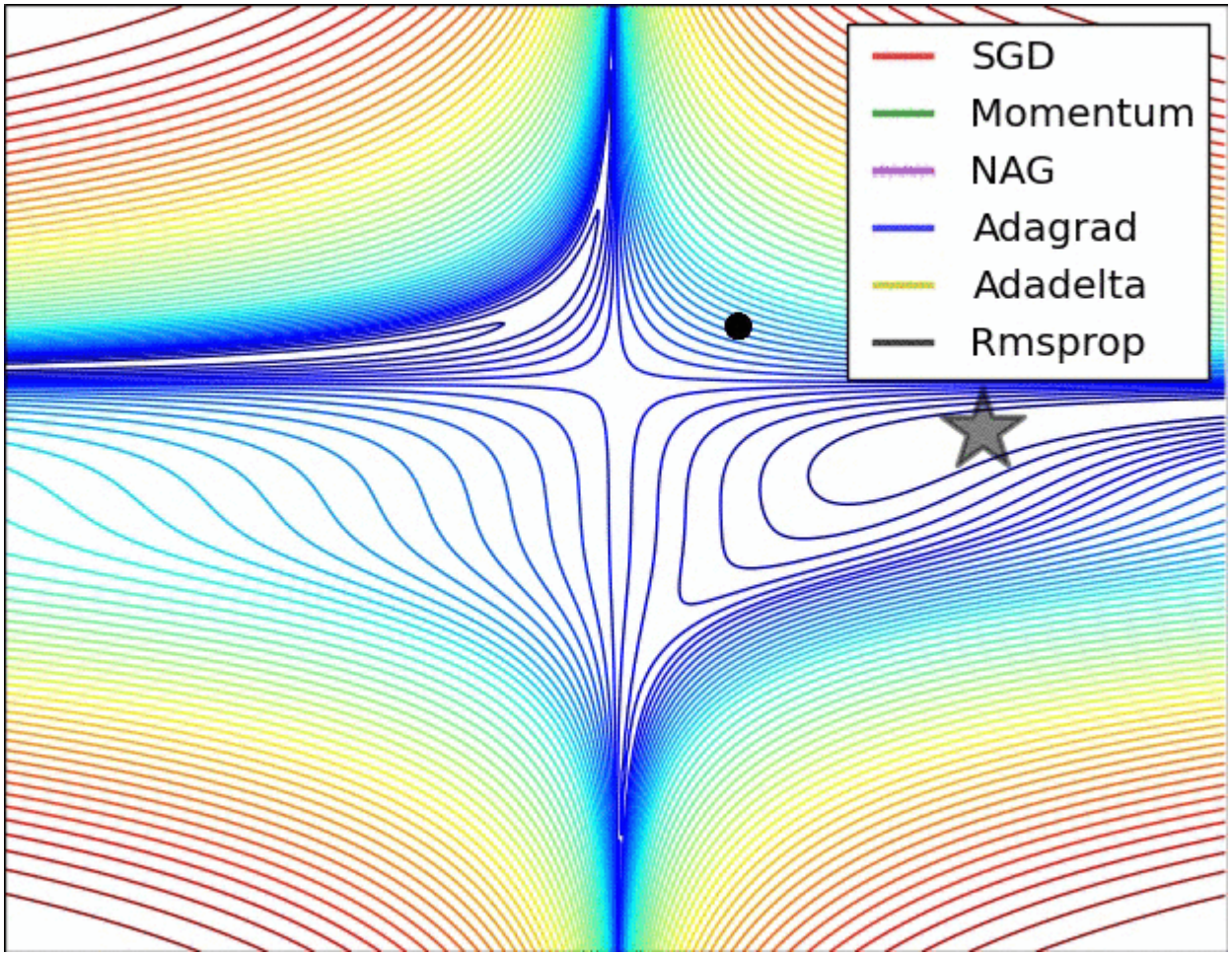




$$a^{(t+1)} \leftarrow a^{(t)} - \eta \nabla_a$$

회귀

- **경사하강법 기반의 학습 알고리즘**



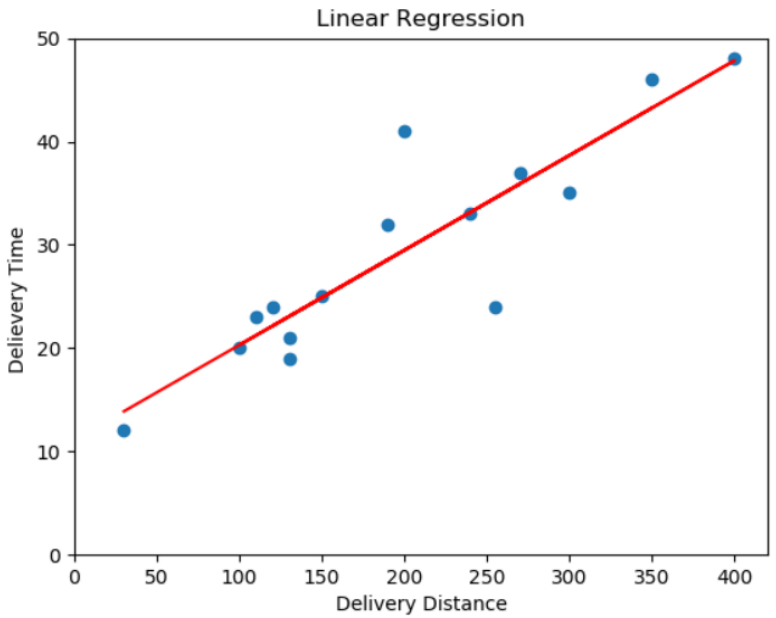
http://sebastianruder.com/content/images/2016/09/contours_evaluation_optimizers.gif

회귀

❖ [실습] 선형회귀

- 입력: 배달거리(delivery distance), 출력: 배달시간(delivery time)
- 파라미터에 대한 1차 방정식을 사용한 회귀

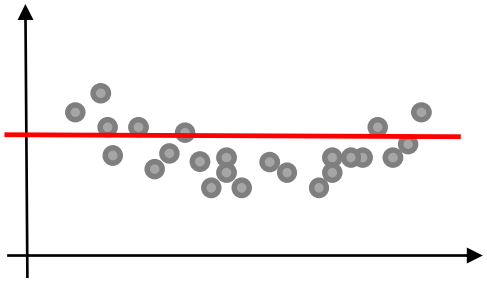
```
import numpy as np
from sklearn.linear_model import LinearRegression
from matplotlib import pyplot as plt

data = np.array([[30, 12], [150, 25], [300, 35], [400, 48], [130, 21],
                 [240, 33],[350, 46], [200, 41], [100, 20], [110, 23],
                 [190, 32], [120, 24], [130, 19], [270, 37], [255, 24]])

plt.scatter(data[:, 0], data[:, 1])  # 데이터 위치의 산포도 출력
plt.title("Linear Regression")
plt.xlabel("Delivery Distance")
plt.ylabel("Delievery Time ")
plt.axis([0, 420, 0, 50])

x = data[:, 0].reshape(-1, 1)   # 입력
y = data[:, 1].reshape(-1, 1)   # 출력

model = LinearRegression( )
model.fit(x, y)                 # 모델 학습

y_pred = model.predict(x)   # 예측값 계산
plt.plot(x, y_pred, color='r')
plt.show( )
```



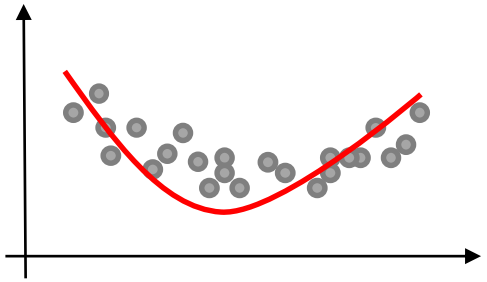
회귀

❖ **회귀**의 **과적합**(overfitting)과 **부적합**(underfitting)

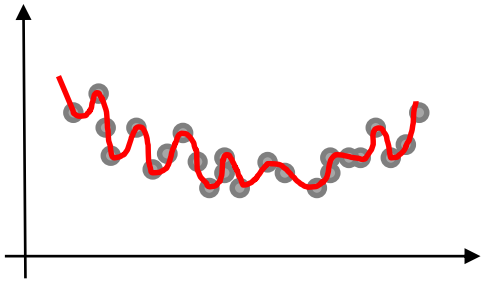
- **과적합**
  - 지나치게 복잡한 모델(함수) 사용
- **부적합**
  - 지나치게 단순한 모델(함수) 사용



부적합(underfitting)



정적합(good fitting)



과적합(overfitting)

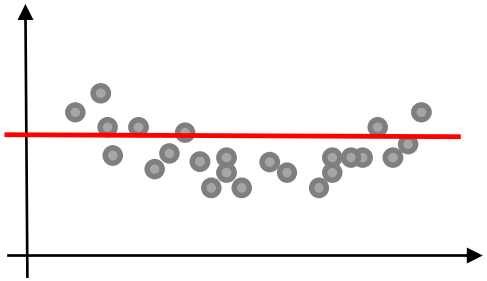
회귀

❖ **회귀**의 **과적합**(overfitting) 대응 방법

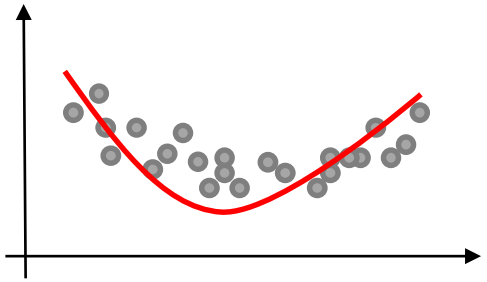
- **모델의 복잡도(model complexity)를 성능 평가에 반영**

목적함수 = 오차의 합 + (가중치)*(모델 복잡도)

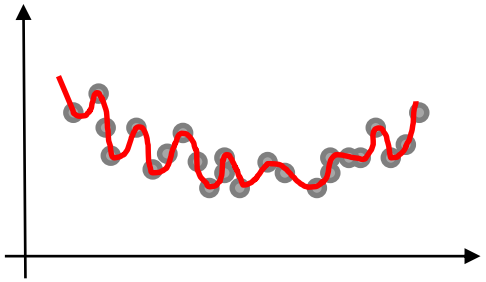
벌점(penalty) 항



부적합(underfitting)



정적합(good fitting)

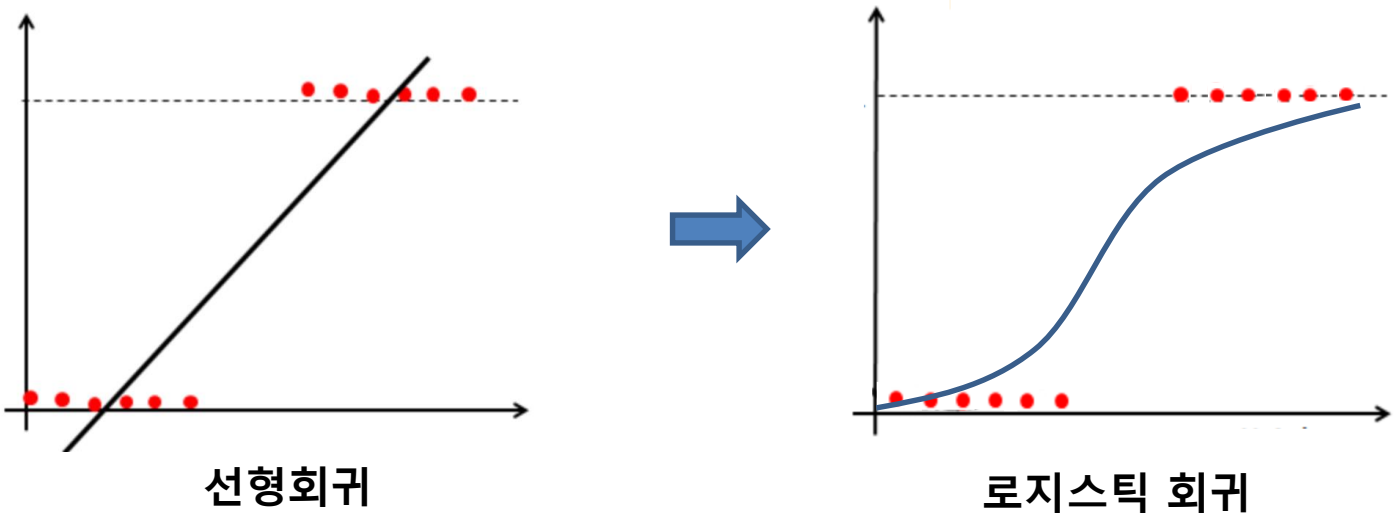


과적합(overfitting)

회귀

❖ 로지스틱 회귀 (logistic regression)

- 학습 데이터 : $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$, $y_i \in \{0, 1\}$: 이진 출력



선형회귀 로지스틱 회귀

- **이진 분류**(binary classification) 문제에 적용

회귀

❖ 로지스틱 회귀 (logistic regression)

- 학습 데이터 : $\mathrm{X} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$, $y_i \in \{0, 1\}$: 이진 출력
- 로지스틱 함수를 이용하여 함수 근사

$$f(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}$$

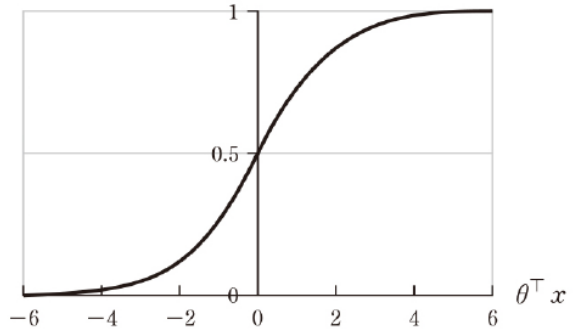


그림 4.13 로지스틱 함수.

- **가능도**(likelihood)
  - 모델이 학습 데이터를 생성할 가능성
  - $P(X) = \prod_{i=1}^{N} f(\boldsymbol{x}_i)^{y_i}(1 - f(\boldsymbol{x}_i))^{1-y_i}$
  - $LogP = -\frac{1}{N}\log P(X) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log f(\boldsymbol{x}_i) + \left(1 - f(\boldsymbol{x}_i)\right)\log(1 - y_i)\right)$

- **경사하강법 사용하여 학습**

[실습] 로지스틱 회귀

https://drive.google.com/file/d/1Upqoz2gIAYq6LByD7YjHU-9_9K4EOhOh/view

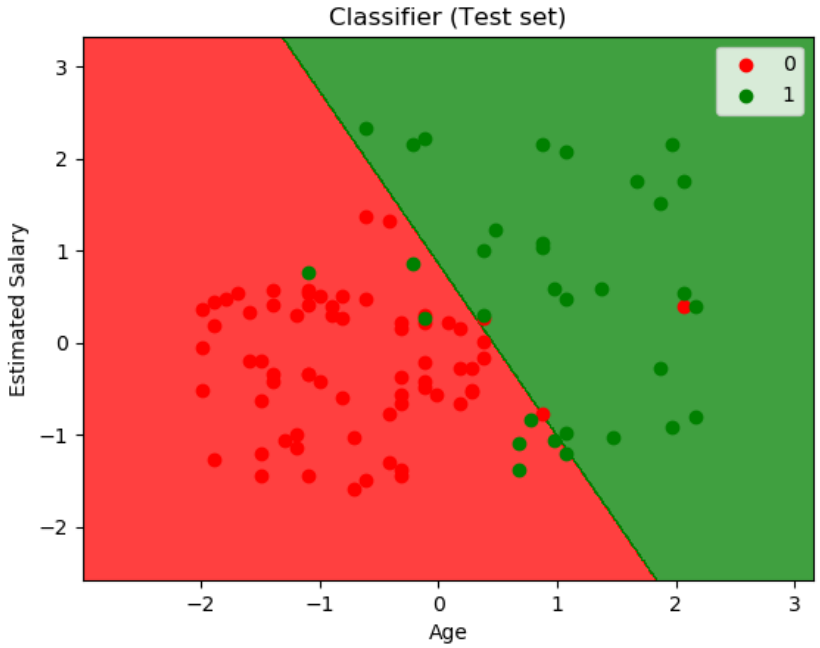
User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv('User_Data.csv')

x = dataset.iloc[:, [2, 3]].values   # 입력
y = dataset.iloc[:, 4].values   # 출력

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)
print (xtrain[0:10, :])

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)
y_pred = classifier.predict(xtest)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)
print ("혼동행렬 : ₩n", cm)

from sklearn.metrics import accuracy_score
print ("정확도 : ", accuracy_score(ytest, y_pred))
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]
혼동행렬 :
 [[65  3]
 [ 8 24]]
정확도 :  0.89
```

https://www.geeksforgeeks.org/ml-logistic-regression-using-python/

```
from matplotlib.colors import ListedColormap
X_set, y_set = xtest, ytest
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                               stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                               stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(
             np.array([X1.ravel(), X2.ravel()]).T).reshape(
             X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



지도학습

❖ 회귀에서 오차의 편향과 분산 분해

- **오차의 기대값(E(E))**

  = 편향^2 + 분산

$E = (f(x) - y)^2$: 오차

$f(x)$: 회귀 함수로 예측한 값

y : 실제 측정값

$$\begin{aligned}
\mathrm{E}(E) &= \mathrm{E}[(f(x) - y)^2] \\
&= \mathrm{E}[(f(x) - E[f(x)] + \mathrm{E}[f(x)] - y)^2] \\
&= \mathrm{E}[(\mathrm{E}[f(x)] - y)^2 + 2(f(x) - \mathrm{E}[f(x)])(\mathrm{E}[f(x)] - y) + (f(x) - \mathrm{E}[f(x)])^2] \\
&= \mathrm{E}[(\mathrm{E}[f(x)] - y)^2] + \mathrm{E}[2(f(x) - \mathrm{E}[f(x)])(\mathrm{E}[f(x)] - y)] + \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2] \\
&\quad (\because \mathrm{E}[f(x)] - y = \text{상수}) \\
&= (\mathrm{E}[f(x)] - y)^2 + 2(\mathrm{E}[f(x)] - y)\mathrm{E}[(f(x) - \mathrm{E}[f(x)])] + \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2] \\
&\quad (\because \mathrm{E}[(f(x) - \mathrm{E}[f(x)])] = \mathrm{E}[f(x)] - \mathrm{E}[f(x)] = 0) \\
&= (\mathrm{E}[f(x)] - y)^2 + \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2] \\
&= bias^2 + variance
\end{aligned}$$

$bias = |\mathrm{E}[f(x)] - y|$ **편향** : 측정값(y)와 예측값들의 평균(E[$f(x)$])과의 차이

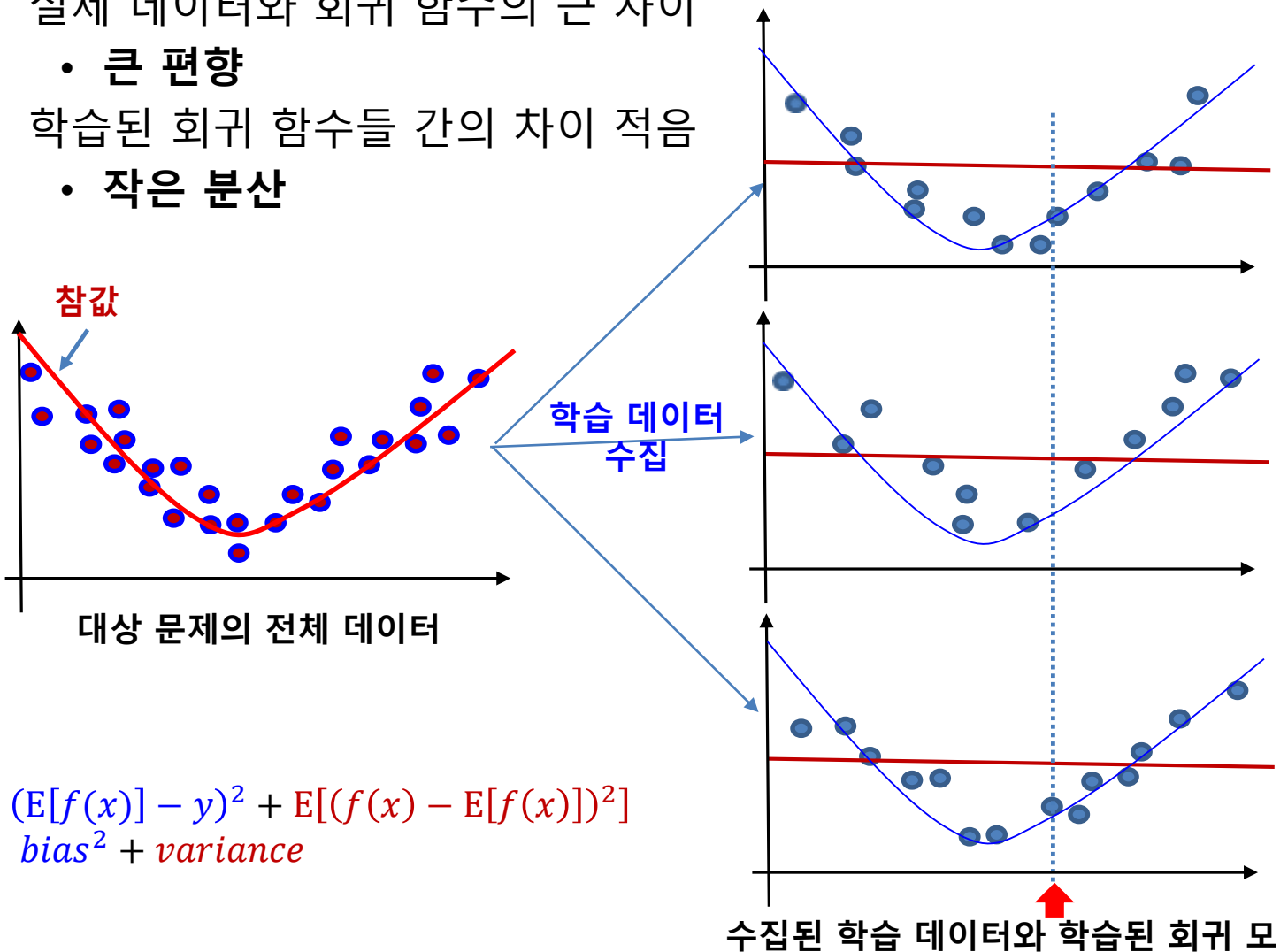
$variance = \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2]$ **분산** : 예측값들의 분산

- **편향과 분산은 회귀 함수의 형태에 따라 영향을 받음**

지도학습

❖ **복잡도가 낮은 단순한 모델을 이용한 회귀의 반복**

- 실제 데이터와 회귀 함수의 큰 차이
  - **큰 편향**
- 학습된 회귀 함수들 간의 차이 적음
  - **작은 분산**

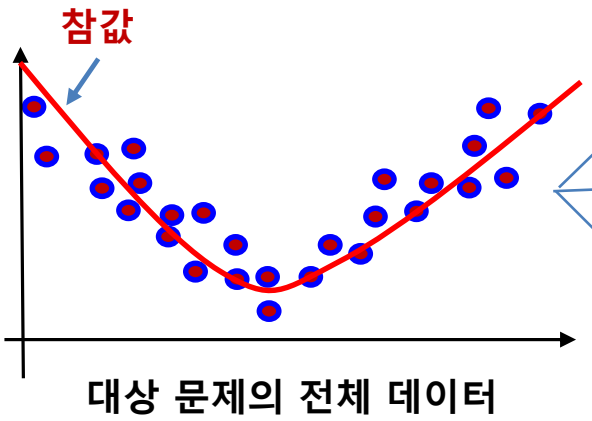


$$\mathrm{E}(E) = (\mathrm{E}[f(x)] - y)^2 + \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2]$$
$$= bias^2 + variance$$

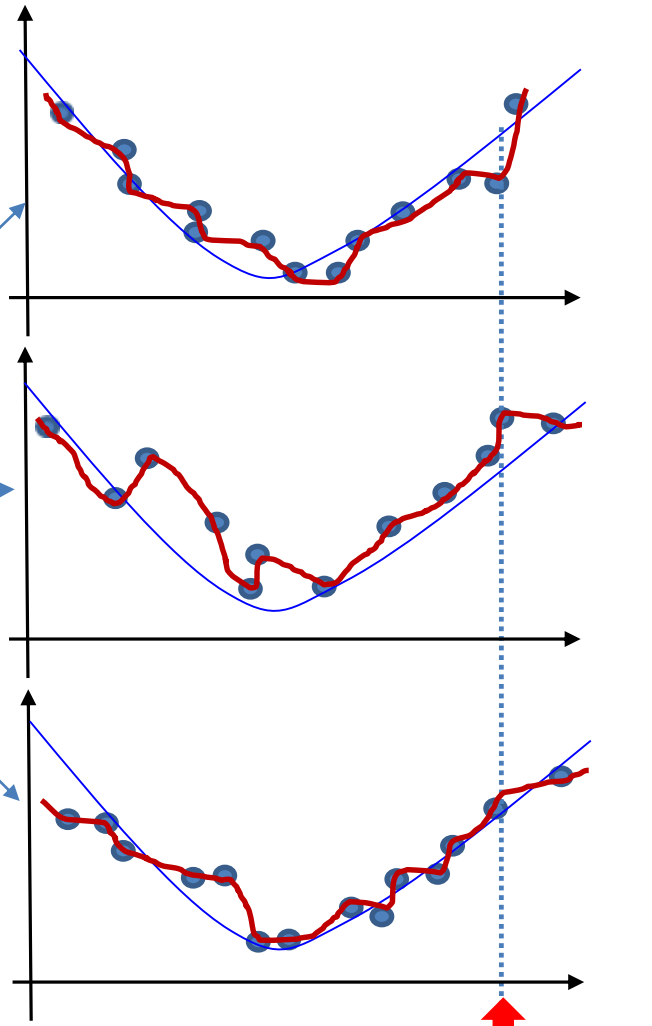
지도학습

❖ **복잡도가 높은 복잡한 모델을 이용한 회귀의 반복**

- 실제 데이터와 회귀 함수의 작은 차이
  - **작은 편향**
- 학습된 회귀 함수들 간의 큰 차이
  - **큰 분산**



학습 데이터 수집

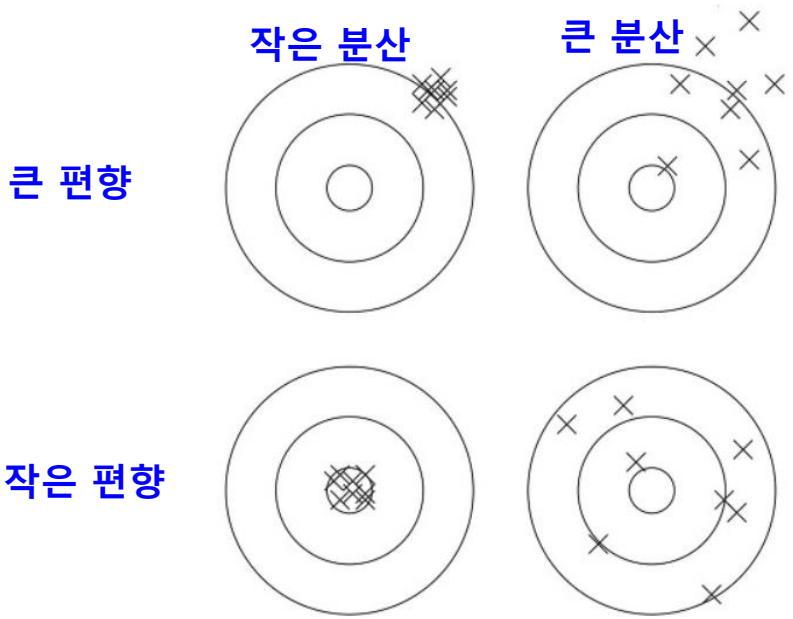


$$\mathrm{E}(E) = (\mathrm{E}[f(x)] - y)^2 + \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2]$$
$$= bias^2 + variance$$

지도학습

❖ **편향-분산 트레이드오프**(bias-variance tradeoff)

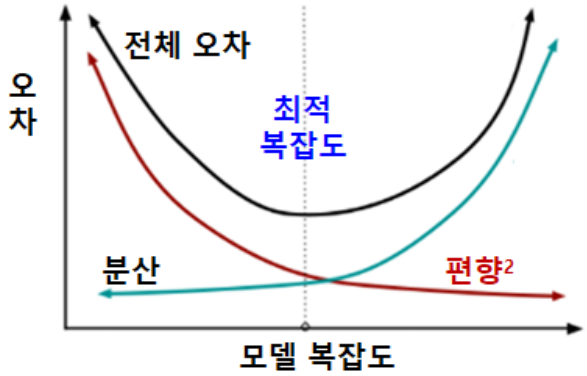
- **편향 (bias)**
  - **단순한 모델로 잘못 가정**을 할 때 발생하는 크게 발생
    - **부적합**(underfitting) 문제 초래
- **분산 (variance)**
  - **복잡한 모델 사용에 따라 학습 데이터에 내재된 작은 변동** (fluctuation) 때문에 발생
    - 큰 잡음(noise)까지 학습하는 **과적합**(overfitting) 문제 초래

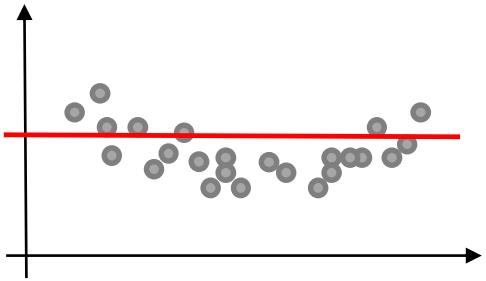


지도학습

❖ **편향-분산 트레이드오프**(bias-variance tradeoff) – cont.

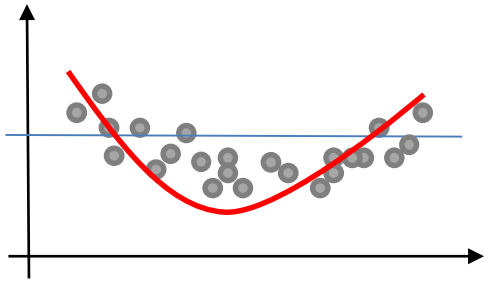
- **모델 복잡도 증가**
  - 편향 감소, 분산 증가
- **모델 복잡도 감소**
  - 편향 증가, 분산 감소
- 편향과 분산 **동시 축소 어려움**
  - 적합한 복잡도의 모델 선택





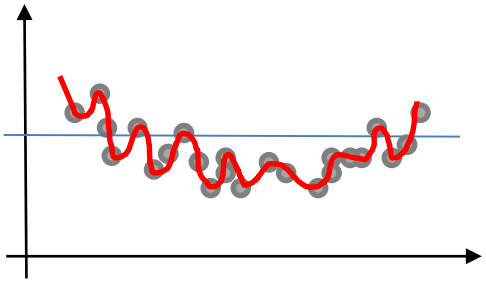
부적합(underfitting)

큰 편향
작은 분산



정적합(good fitting)

중간 편향
중간 분산



과적합(overfitting)

작은 편향
큰 분산

2. 추천

❖ **추천** (recommendation)

- 개인별로 맞춤형 정보를 제공하려는 기술
- 사용자에게 맞춤형 정보를 제공하여 정보 검색의 부하를 줄여주는 역할

- **추천 데이터**
  - **희소 행렬(sparse matrix) 형태**
    - 많은 원소가 비어 있음
    - 비어 있는 부분을 채우는 것이 추천에 해당

추천

❖ 추천 기법

- **내용 기반 추천**(content-based recommendation)
  - 고객이 이전에 높게 평가했던 것과 유사한 내용을 갖는 대상을 추천
- **협력 필터링**(collaborative filtering)
  - **사용자간 협력 필터링**(user-user collaborative filtering)
    - 추천 대상 사용자와 비슷한 평가를 한 사용자 집합 이용
  - **항목간 협력 필터링**(item-item collaborative filtering)
    - 항목간의 유사도를 구하여 유사 항목을 선택
- **은닉 요소 모델**(latent factor model)
  - 행렬 분해에 기반한 방법

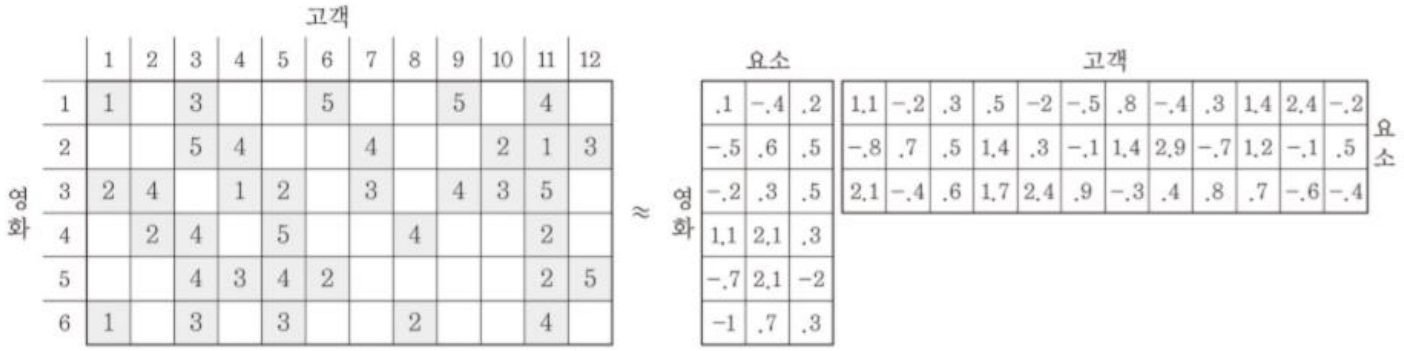


그림 7.18 추천 데이터 행렬의 분해

Quiz

❖ **회귀에 대한 설명으로 옳지 않는 것을 선택하시오.**

① 회귀에서는 출력값이 실수인 함수를 찾는다.
② 회귀에서는 오차의 제곱합을 목적함수로 사용할 수 있다.
③ 복잡도가 높은 함수를 사용하면 부적합 문제가 발생할 수 있다.
④ 목적함수에 모델 복잡도 항을 포함시키면 부적합 학습을 피할 수 있다.

❖ **로지스틱 회귀에 대한 설명으로 옳지 않은 것을 선택하시오.**

① 출력이 1 또는 0인 이진 분류 문제에 적용되는 기법이다.
② 로지스틱 회귀의 결과로 학습되는 모델의 출력은 구간 (0,1) 사이의 값이다.
③ 로지스틱 회귀 모델의 목적함수로는 교차 엔트로피가 사용될 수 있다.
④ 교차 엔트로피의 값은 클 수록 바람직하기 때문에 경사 상승법을 사용하여 학습한다.

Quiz

❖ **편향-분산 트레이드오프에 대한 설명으로 옳지 않는 것을 선택하시오.**

① 편향은 학습 알고리즘에서 선택한 모델이 너무 단순할 때 크게 발생한다.
② 분산은 학습 모델이 복잡해질 때 커지는 경향이 있다.
③ 학습을 할 때 편향이 크더라도 분산이 작으면 좋은 모델이다.
④ 과적합이 된 모델인 경우 분산이 커지는 경향을 보인다.

❖ **추천에 트레이드오프에 대한 설명으로 옳지 않는 것을 선택하시오.**

① 추천 데이터는 희소 행렬 행태를 가지며, 이런 희소 행렬의 비어있는 곳을 채우는 것이 추천에서 해야 할 일이다.
② 고객이 이전에 높게 평가했던 것과 유사한 내용을 갖는 대상을 추천하는 것을 내용기반 추천이라고 한다.
③ 추천 대상 사용자와 비슷한 평가를 한 사용자 집합을 찾아 이들의 추천 정보를 이용하는 것을 사용자가 협력 필터링이라고 한다.
④ 희소행렬을 행렬의 곱으로 근사하는 방법으로 항목 간의 유사도를 결정하여 추천하는 기법을 은닉 요소 기반 기법이라고 한다.