가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

기계학습

-지도학습 1-

미디어기술콘텐츠학과 / 의료인공지능학과
강호철

분류와 회귀

- 분류
  - 미리 정의된, 가능성 있는 여러 클래스 레이블 중 하나를 예측하는 것
  - 딱 두 개의 클래스로 분류하는 이진 분류(binary classification)와 셋 이상의 클래스로 분류하는 다중 분류(multiclass)로 나뉨
    - 예) 스팸 메일 분류?
    - 예) 붓꽃 분류?
- 회귀
  - 연속적인 숫자, 또는 프로그래밍 용어로 말하면 부동소수점수(수학 용어로는 실수)를 예측하는 것
    - 예) 어떤 사람의 교육 수준, 나이, 주거지를 바탕으로 연간 소득을 예측
    - 예) 옥수수 농장의 수확량 예측

일반화, 과대적합, 과소적합

- 예) 요트 구매 고객 예측
  - 요트를 구매한 고객과 구매하지 않은 고객 데이터를 이용하여 예측
  - 구매 할 확률이 높은 고객에게 홍보 메일을 보내는 것이 목표

Table 2-1. Example data about customers

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

일반화, 과대적합, 과소적합

- 예) 요트 구매 고객 예측
  - 가정1: 45세 이상 and (자녀가 셋 미만 or 이혼 하지 않은 고객)이 요트를 구매한다.
  - 정확도: 100%
  - 우리의 목적에 잘 맞을까?
    - 평가를 하려면?
  - 가정2: 50세 이상이 요트를 구매한다
  - 일반화 관점에서 가정1 vs. 가정2 중 더 좋은 모델은?
  - 가정 3: 이혼한 사람이 요트를 구매한다

Table 2-1. Example data about customers

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

일반화, 과대적합, 과소적합

- 일반화 성능이 최대가 되는 모델이 최적임
  - 일반화
    - 모델이 처음 보는 데이터에 대해 정확하게 예측할 수 있으면 이를 훈련 세트에서 테스트 세트로 "일반화(generalization)"되었다고 함
  - 과대적합
    - 가진 정보를 모두 사용해서 너무 복잡한 모델을 만드는 것을 "과대적합 (overfitting)"이라 함
  - 과소적합
    - 모델이 너무 간단하여 데이터의 면면과 다양성을 잡아내지 못하고 훈련 세트에도 잘 맞지 않는 경우처럼, 너무 간단한 모델이 선택되는 것을 "과소적합(underfitting)"이라 함

일반화, 과대적합, 과소적합

- 모델 복잡도에 따른 훈련과 테스트 정확도

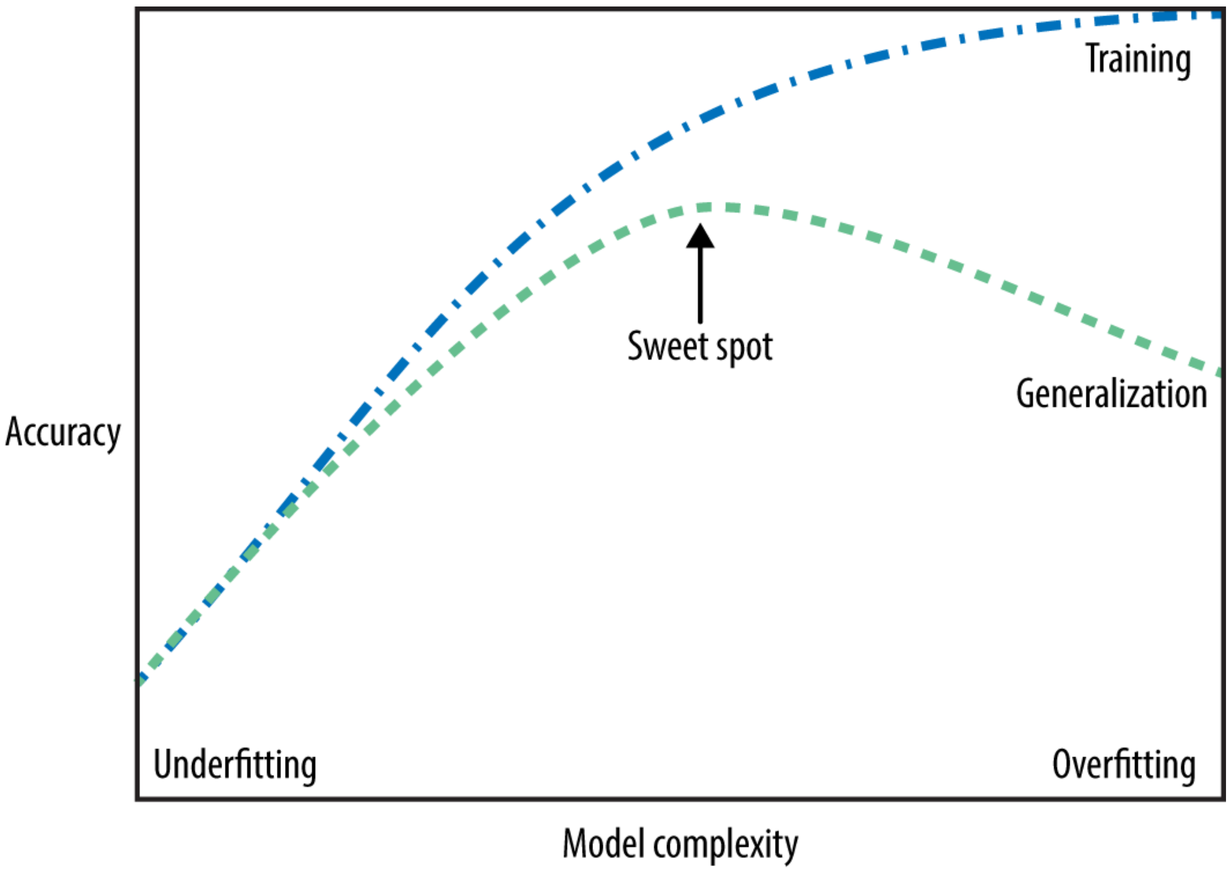


Figure 2-1. Trade-off of model complexity against training and test accuracy

일반화, 과대적합, 과소적합

- 모델 복잡도와 데이터 셋 크기의 관계
  - 모델의 복잡도는 훈련 데이터 셋에 담긴 입력 데이터의 다양성과 관련이 깊음
    - 데이터 포인트(샘플)이 많을수록 과대 적합 없이 복잡한 모델 생성
    - 요트 판매에서 고객 데이터 10,000개를 모았을 때 가정 1을 만족했다면 12개를 사용 할 때 보다 좋은 규칙이라고 생각 할 수 있음

지도 학습 알고리즘

- 머신러닝 알고리즘의 작동 방식 학습
  - 데이터로부터 어떻게 학습하고 예측하는가?
  - 모델의 복잡도가 어떤 역할을 하는가?
  - 알고리즘이 모델을 어떻게 만드는가?
  - 모델들의 장단점을 평가하고 어떤 데이터가 잘 들어맞을지 살펴보기
  - 매개변수와 옵션의 의미 학습

- 알고리즘
  - K-최근접 이웃
  - 선형모델
  - 나이브 베이즈 분류기
  - 결정 트리 및 앙상블
  - 커널 서포트 벡터 머신
  - 신경망 (딥러닝)

지도 학습 알고리즘

- 데이터 셋 생성

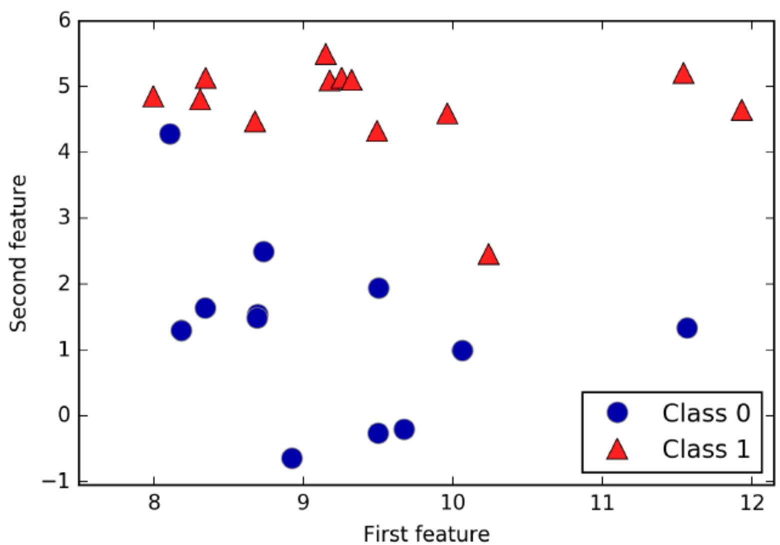


Figure 2-2. Scatter plot of the forge dataset

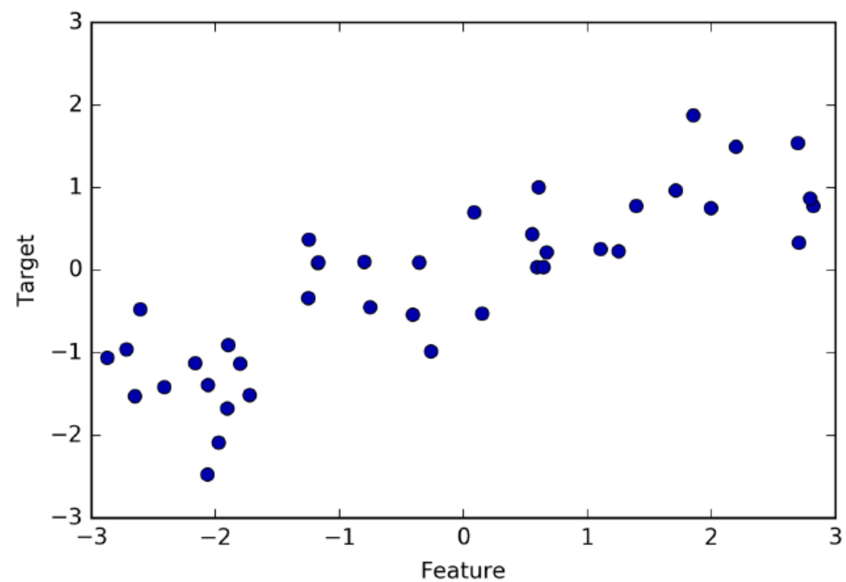


Figure 2-3. Plot of the wave dataset, with the x-axis showing the feature and the y-axis showing the regression target

지도 학습 알고리즘

- K-NN 알고리즘
  - 중요 매개변수
    - Metric
    - # of neighbor
  - 이해하기 매우 쉬운 알고리즘
  - 조정을 많이 하지 않아도 성능이 괜찮음
  - 복잡한 알고리즘을 사용하기 전에 시도해보기 좋음
  - 훈련 세트가 매우 크면 예측이 느려짐
  - 전처리 중요 (3장 참고)
  - 특성이 많은 경우 잘 작동하지 않음
  - 특성 값 대부분이 0인 경우 잘 작동하지 않음
  - 요약: 이해하기 쉬우나 예측이 느리고 많은 특성을 처리하는 능력 부족으로 현업에서는 잘 사용하지 않음

지도 학습 알고리즘

- 1차원 선형 모델
  - 학습데이터
    - 총 N개의 (나이, 키) 데이터
    - 입력변수 x = {x0, x1, x2, …. xN-1}
    - 목표변수 t = {t0, t1, t2, …. tN-1}
    - 데이터 랜덤 생성

지도 학습 알고리즘

- 1차원 선형 모델
  - Hypothesis (모델링)
    - 분포하는 데이터를 보고 모델링
    - 가장 간단한 모델인 직선 방정식 사용
      - $y(x) = w_0x + w_1$
  - Loss function (비용함수)
    - 학습 데이터 분포와 가장 잘 맞는 직선의 식을 찾는 것이 목적
    - 직선 방정식을 이루는 적당한 파라미터 (w_0, w_1)을 찾아야 함
      - 학습 데이터와 직선의 방정식이 가장 잘 맞는다의 의미?
        - 맞고 안 맞음의 정도를 측정할 수 있는 측정값 필요
    - 가장 많이 사용되는 평균제곱오차(MSE)
      - $J(w0, w1) = \frac{1}{N}\sum_{n=0}^{N-1}(yn - tn)^2$
      - $y_n = y(x_n) = w_0x_n + w_1$

지도 학습 알고리즘

- 1차원 선형 모델
  - Optimization (최적화)
    - Loss function을 최소화 하는 작업
      - J 함수가 최소가 되는 w0, w1 값 계산
      - Gradient Descent (경사하강법)
      - w(k+1) = w(k) - α∇J
      - ∇J = ?
      - w(k+1) = ?

```
def calcGrad(x, t, w0, w1):
    y = w0 * x + w1
    d_w0 = 2 * np.mean((y - t) * x)
    d_w1 = 2 * np.mean(y - t)
    return d_w0, d_w1

temp = calcGrad(X, T, 10, 165)
print(np.round(temp, 1))
```

지도 학습 알고리즘

- 1차원 선형 모델
  - Optimization (최적화)
    - Loss function을 최소화 하는 작업
      - grad_loss 함수 이용 경사하강법

```
def GradDsctOpt(x, t):
    w_init = [10.0, 165.0] # 초기 매개 변수
    alpha = 0.001 # 학습률
    i_max = 100000 # 반복의 최대 수
    eps = 0.1 # 반복을 종료 기울기의 절대 값의 한계
    w0 = w_init[0]
    w1 = w_init[1]
    for i in range(1, i_max):
        dmse = calcGrad(x, t, w0, w1)
        w0 = w0 - alpha * dmse[0]
        w1 = w1 - alpha * dmse[1]
        if max(np.absolute(dmse)) < eps: # 종료판정, np.absolute는 절대치
            break
    return w0, w1
```

```
# 구배법 호출
_W0, _W1 = GradDsctOpt(X, T)
# 결과보기
print('W=[{0:.6f}, {1:.6f}]'.format(_W0, _W1))
```

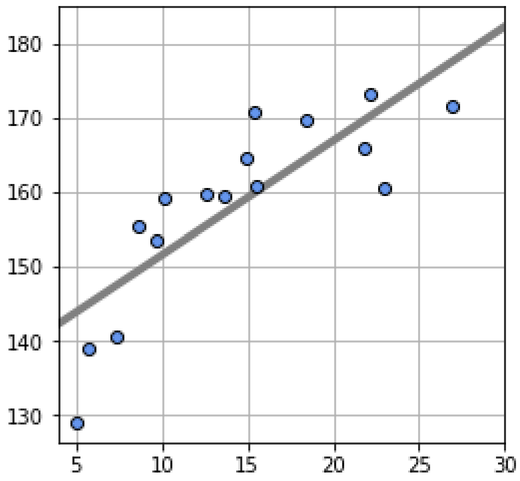
지도 학습 알고리즘

- 1차원 선형 모델
  - 최종 결과
    - 그래프 가시화

```
# 리스트 5-1-(10)
# 선 표시 ----------------------------------
def show_line(w):
    xb = np.linspace(X_min, X_max, 100)
    y = w[0] * xb + w[1]
    plt.plot(xb, y, color=(.5, .5, .5), linewidth=4)


# 메인 ------------------------------------
plt.figure(figsize=(4, 4))
W=np.array([W0, W1])
mse = mse_line(X, T, W)
print("w0={0:.3f}, w1={1:.3f}".format(W0, W1))
# mse = mse_line(X, T, W)
print("SD={0:.3f} cm".format(np.sqrt(mse)))
show_line(W)
plt.plot(X, T, marker='o', linestyle='None',
         color='cornflowerblue', markeredgecolor='black')
plt.xlim(X_min, X_max)
plt.grid(True)
plt.show()
```

```
w0=1.540, w1=136.176
SD=7.002 cm
```

```
def mse_line(x, t, w):
    y = w[0] * x + w[1]
    mse = np.mean((y - t)**2)
    return mse
```



지도 학습 알고리즘

- 선형 모델 실습
  - 1차원 직선 방정식을 정하고 이 식을 이용하여 데이터를 50개 생성
    - w0 = 3,   w1 = 5
    - x의 범위는 -20~50까지 50개 생성
    - 랜덤 값을 이용하여 생성 (scale = 50)
  - loss function, gradient function, gradient_descent function 구현
  - 최적화 및 w0, w1 값 구하기
    - 초기값 w0 = ?,   w1 = ?
    - learning rate = ?
    - iteration count = ?

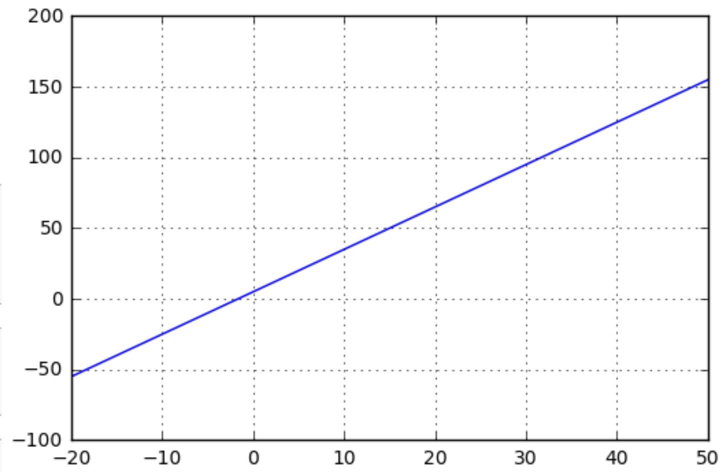
지도 학습 알고리즘

- 선형 모델 실습
  - 데이터 생성

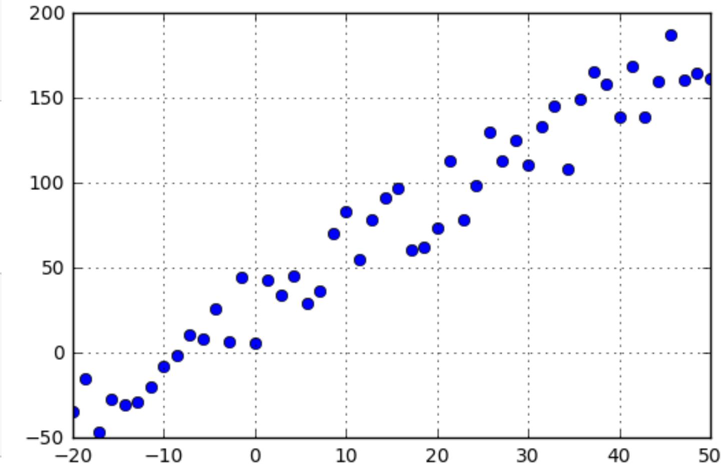
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

```
def targetfunc(x):
    return 3*x + 5
```

```
x = np.linspace(-20, 50, 50)
fx = targetfunc(x)
plt.plot(x,fx)
plt.grid()
plt.show()
```




```
np.random.seed(1)
t = fx + 50 * np.random.rand(len(x))
plt.plot(x, t, 'o')
plt.grid()
plt.show()
```



지도 학습 알고리즘

- 선형 모델 실습
  - gradient loss, gradient descent, show line function 정의

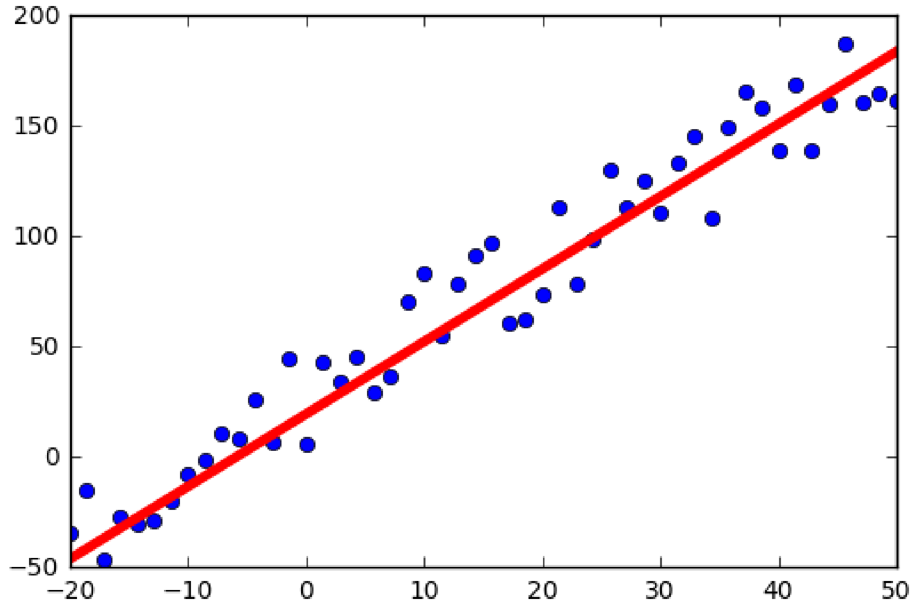
```
def grad_loss(x, t, w0, w1):
    y = w0 * x + w1
    grad_w0 = 2*np.mean((y-t) * x)
    grad_w1 = 2*np.mean(y - t)
    return grad_w0, grad_w1

def grad_descent(x, t, w0, w1, lr, itr):
    _w0 = w0
    _w1 = w1
    eps = 0.1
    for i in range(1, itr):
        grad_w = grad_loss(x, t, _w0, _w1)
        _w0 = _w0 - lr*grad_w[0]
        _w1 = _w1 - lr*grad_w[1]
        if (max(np.absolute(grad_w)) < eps):
            break
    return _w0, _w1

def show_line(x, t, w0, w1):
    # true - dot
    plt.plot(x, t, 'o')
    # model - line
    y = w0*x + w1
    plt.plot(x, y, color='red', linewidth = 4)
```

지도 학습 알고리즘

- 선형 모델 실습
  - 함수 실행

```
w0 = 1.0
w1 = 1.0
lr = 0.001
itr = 1000
w0_opt, w1_opt = grad_descent(x, t, w0, w1, lr, itr)
print('W = {0:.3f}, {1:.3f}'.format(w0_opt, w1_opt))
show_line(x, t, w0_opt, w1_opt)
```

```
W = 3.285, 19.581
```



지도 학습 알고리즘

- 선형 모델 – 회귀
  - 입력 특성에 대한 선형 함수를 만들어 예측 수행
  - 회귀의 선형 모델

$$\hat{y} = w[0] * x[0] + b$$

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b$$

    - 특성이 한 개인 경우: 직선
    - 특성이 두 개인 경우: 평면
    - 특성이 N 개인 경우: 초평면
  - 최소 제곱법

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2$$

Cost function for simple linear model

지도 학습 알고리즘

- 선형 모델 – 회귀
  - 릿지 회귀
    - 회귀를 위한 선형 모델의 일종
    - 가중치의 절대값을 가능한 한 작게 만듦
      - 특성이 출력에 주는 영향 최소화
      - L^2norm 규제 (regularization)

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}w_j^2$$

Cost function for ridge regression

지도 학습 알고리즘

- 선형 모델 – 회귀
  - 라쏘
    - 릿지와 마찬가지로 가중치의 절대값을 가능한 한 작게 만듦
    - L^1norm 규제 사용

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$

Cost function for Lasso regression

지도 학습 알고리즘

- 회귀 vs. 분류
  - 회귀문제
    - 목표 데이터: 연속된 수치
  - 분류문제
    - 목표 데이터: 클래스 (카테고리, 라벨)
    - 확률 개념 도입

지도 학습 알고리즘

- 2차원 2클래스 분류
  - 데이터 만들기
    - 선형 식을 정의하고 클래스 설정
      - 식 = $w_0*x_0 + w_1*x_1 + w_2$
      - 직선을 기준으로 아래에 있는 데이터는 클래스 0
      - 직선을 기준으로 위에 있는 데이터는 클래스 1
    - 데이터 분포 가시화 (N = 50)
    - $3/4x_0 + 1.0x_1 - 4/5 = 0$ 식을 이용하여 클래스 지정
  - 비용 함수 설정
    - Cross-Entropy
    - $E(W) = -\frac{1}{N}\sum_{n=0}^{N-1}\{tnlogyn + (1 - tn)\log(1 - yn)\}$
  - 최적화
    - 경사 하강법 사용

지도 학습 알고리즘

- 2차원 2클래스 분류
  - 데이터 만들기

```
np.random.seed(seed=1)  # 난수를 고정
W = np.array([3./4., 1.0, -4./5.])
N = 50
dim = 2
K = 2
scale = 1;
T = np.zeros((N, K), dtype=np.uint8)
X = scale*np.random.rand(N, dim)
print(X.shape)
```

```
(50, 2)
```

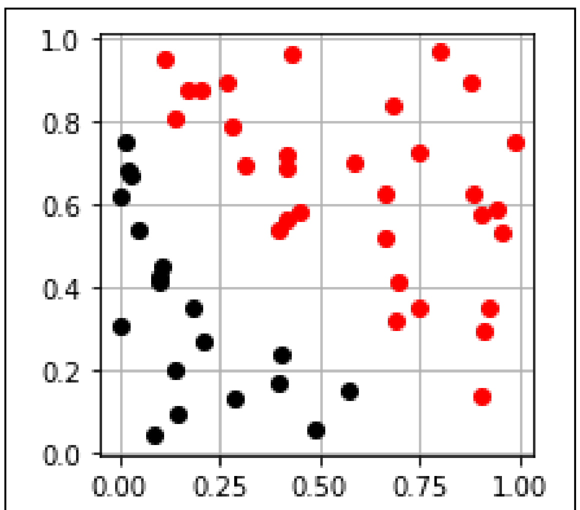
```
for n in range(N):
    for k in range(K):
        if W[0]*X[n, 0]+W[1]*X[n, 1]+W[2] > 0:
            T[n, 1] = 1
        else:
            T[n, 0] = 1

print(X[:5, :])
print(T[:5, :])
```

```
[[4.17022005e-01 7.20324493e-01]
 [1.14374817e-04 3.02332573e-01]
 [1.46755891e-01 9.23385948e-02]
 [1.86260211e-01 3.45560727e-01]
 [3.96767474e-01 5.38816734e-01]]
[[0 1]
 [1 0]
 [1 0]
 [1 0]
 [0 1]]
```

지도 학습 알고리즘

- 2차원 2클래스 분류
  - 데이터 만들기

```
def show_data(x, t):
    c = [[0, 0, 0], [1, 0, 0]]
    for k in range(K):
        plt.plot(x[t[:, k] == 1, 0], x[t[:, k] == 1, 1], linestyle='none', marker='o', color=c[k])
        plt.grid(True)

plt.figure(figsize=(3, 3))
show_data(X, T)
```



지도 학습 알고리즘

- 2차원 2클래스 분류
  - 비용 함수 및 최적화 함수

```
def logistic2(x0, x1, w):
    y = 1 / (1 + np.exp(-(w[0] * x0 + w[1] * x1 + w[2])))
    return y
```

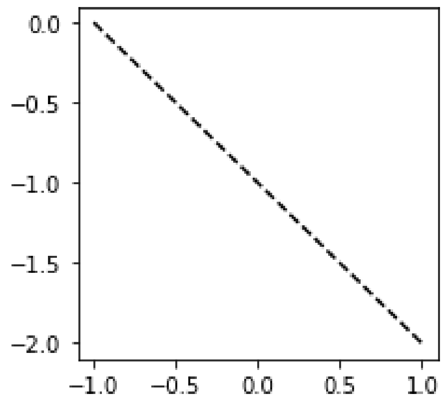
```
def cee_logistic2(w, x, t):
    X_n = x.shape[0]
    y = logistic2(x[:, 0], x[:, 1], w)
    cee = 0
    for n in range(len(y)):
        cee = cee - (t[n, 0] * np.log(y[n]) + (1 - t[n, 0]) * np.log(1 - y[n]))
    cee = cee / X_n
    return cee

# test ---
_W=[-1., -1., -1.]
cee_logistic2(_W, X, T)
```

```
0.7170005111218646
```

지도 학습 알고리즘

- 2차원 2클래스 분류
  - 직선 그리기 함수

```
def show_line(W):
    xn = 50 # 파라미터의 분할 수
    X_range0 = [-1, 1] # X0 범위 표시 용
    x0 = np.linspace(X_range0[0], X_range0[1], xn)
    x1 = -(W[0]/W[1])*x0 - W[2]/W[1]
    plt.plot(x0, x1, '--k')

# test ---
plt.figure(figsize=(3,3))
_W=[-1, -1, -1]
show_line(_W)
```

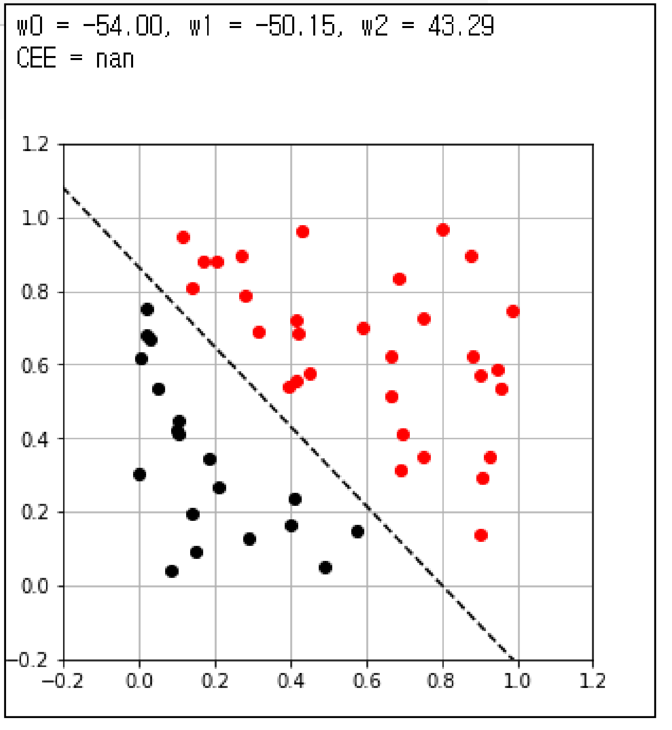


지도 학습 알고리즘

- 2차원 2클래스 분류
  - 최적화 - 경사하강법

```
def fit_logistic(w_init, x, t):
    #res = minimize(cee_logistic2, w_init, args=(x, t), jac=dcee_logistic2, method="CG")
    res = minimize(cee_logistic2, w_init, args=(x, t), method="CG")
    return res.x
```

```
#_W = grad_descent(W_init, X, T, lr, itr)
_W = fit_logistic(W_init, X, T)
```



지도 학습 알고리즘

- 선형 모델 – 분류
  - 이진 분류
    - 두 개의 클래스를 구분하여 예측

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b > 0$$

    - 결정 경계
  - 대표적인 선형 분류 알고리즘
    - 로지스틱 회귀
    - 선형 서포트 벡터 머신

지도 학습 알고리즘

- 선형 모델 – 분류
  - 다중 분류
    - 세 개 이상의 클래스를 구분하여 예측
    - 클래스 별로 이진 분류기에 사용되는 계수 벡터와 절편을 가짐

$$w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b$$

화이트 보드

참고자료

- Introduction to Machine Leanring with Python

  (파이썬 라이브러리를 활용한 머신러닝)
  - 안드레아스 뮐러, 세라 가이도 지음 / 박해선 옮김
  - 한빛미디어, 2019
- 파이썬으로 배우는 머신러닝의 교과서
  - 이토마코토 지음, 박광수 옮김
  - 한빛미디어, 2018