20200106

Part.1

머신러닝 개요, 경사하강법
퍼셉트론, 로지스틱 회귀

1.2. 머신러닝 개념

지도 학습

- 레이블 된 훈련데이터로 모델 학습
- 직접 피드백

>> **분류 , 회귀**

비지도 학습

- 레이블 되지 않은 훈련데이터를 입력
- 피드백 X
- 데이터를 보고 스스로 학습

>> **군집화 , 차원축소**

강화 학습

- 시스템의 성능을 향상하는 것이 목적
- 지도학습과 달리 잘못된 것에 대한 수정X
- 보상시스템 (얼마나 좋은 행동인지)

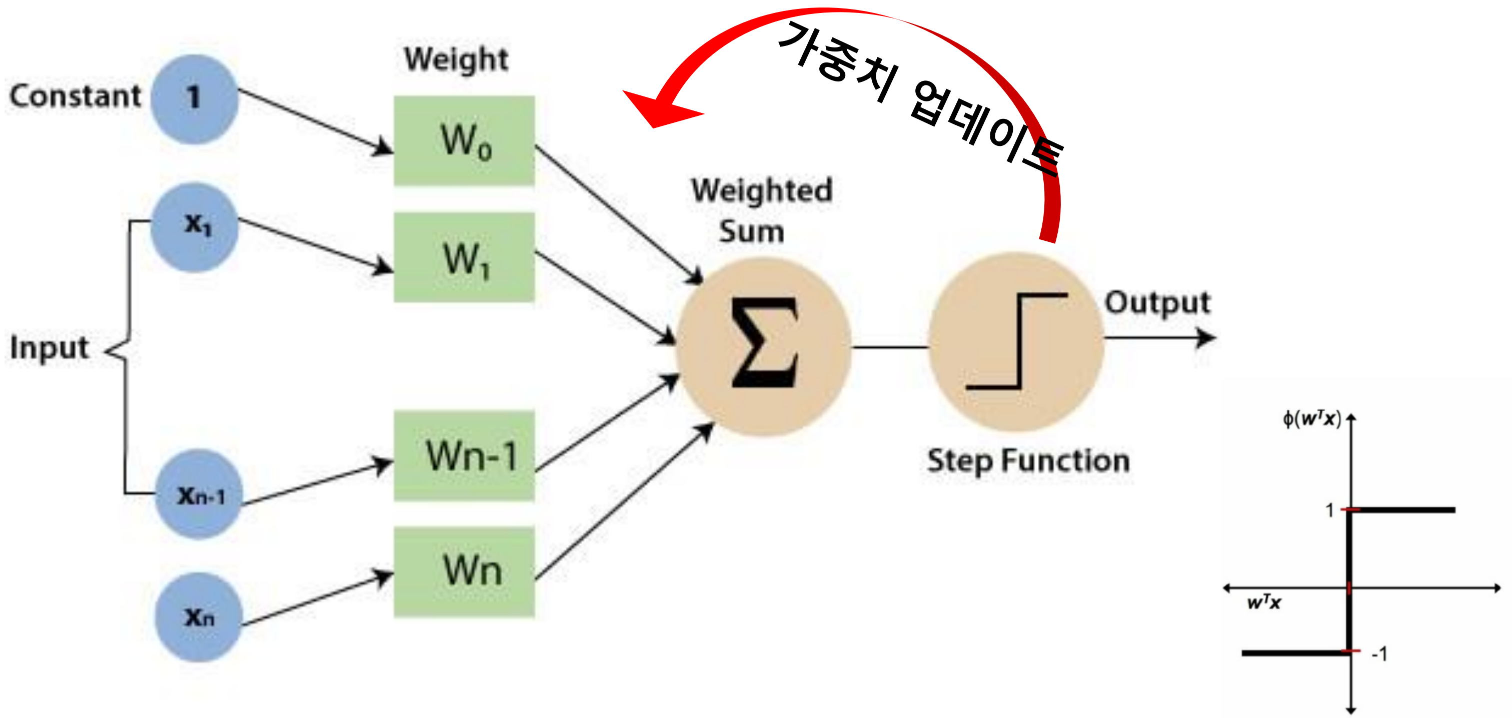
Ex) 체스게임

01
02
03

2.1. 퍼셉트론

퍼셉트론 이란?
: 인공신경망의 한 종류

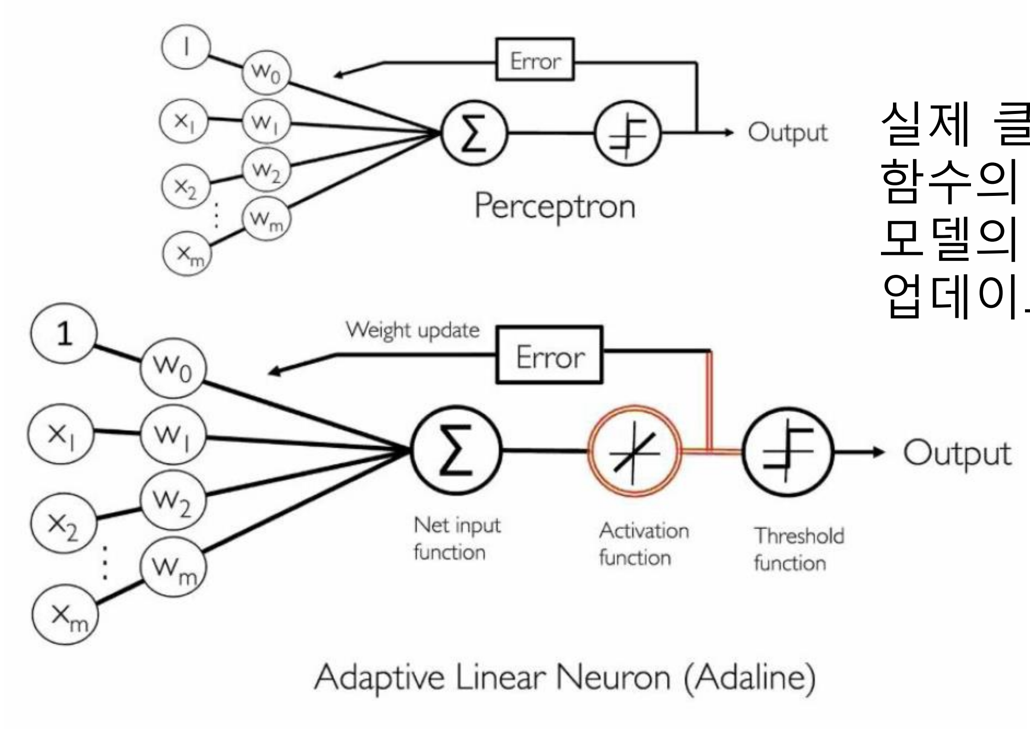
각 노드의 가중치와 입력치를 곱한 것을 모두 합한 값이 활성함수에 의해 판단되는데,
그 값이 임계치(보통 0)보다 크면 뉴런이 활성화되고 결과값으로 1을 출력한다. 뉴런이 활성화되지 않으면 결과값으로 -1을 출력한다.
이 과정을 통해서 최적의 가중치를 찾아가는 알고리즘 학습방법



2.3. 적응형 선형 뉴런과 학습의 수렴

교재 64p

적응형 선형 뉴런(Adaline)이 퍼셉트론과 다른점
: 가중치 업데이트 시 단위 계단함수 대신 <u>선형 활성화 함수</u>를 사용



실제 클래스 레이블과 선형 활성화 함수의 실수 출력값을 비교하여 모델의 오차를 계산하고 가중치를 업데이트함

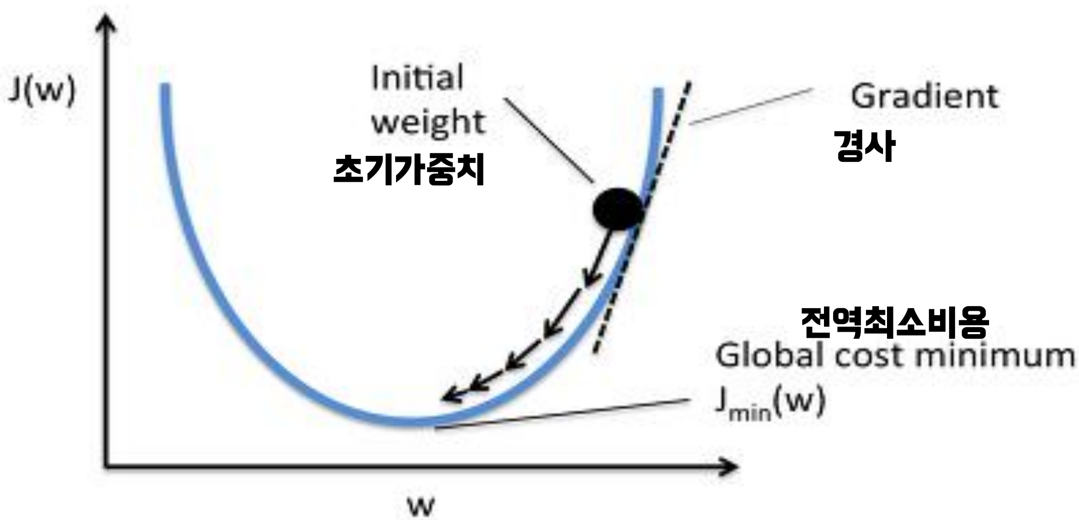
2.3.1 경사하강법으로 비용 함수 최소화

교재 66p

지도 학습의 핵심은 학습 동안 최소화시킬 또는 최대화 시킬 목적 함수를 잘 설정하는 것

아달린에서의 목적함수는 비용함수
: 계산한 출력값과 실제 class label의 SSE(sum of squared error)
즉, 지금 현재의 가중치에서 "틀린정도"를 알려주는 함수

$$J(w) = \frac{1}{2}\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

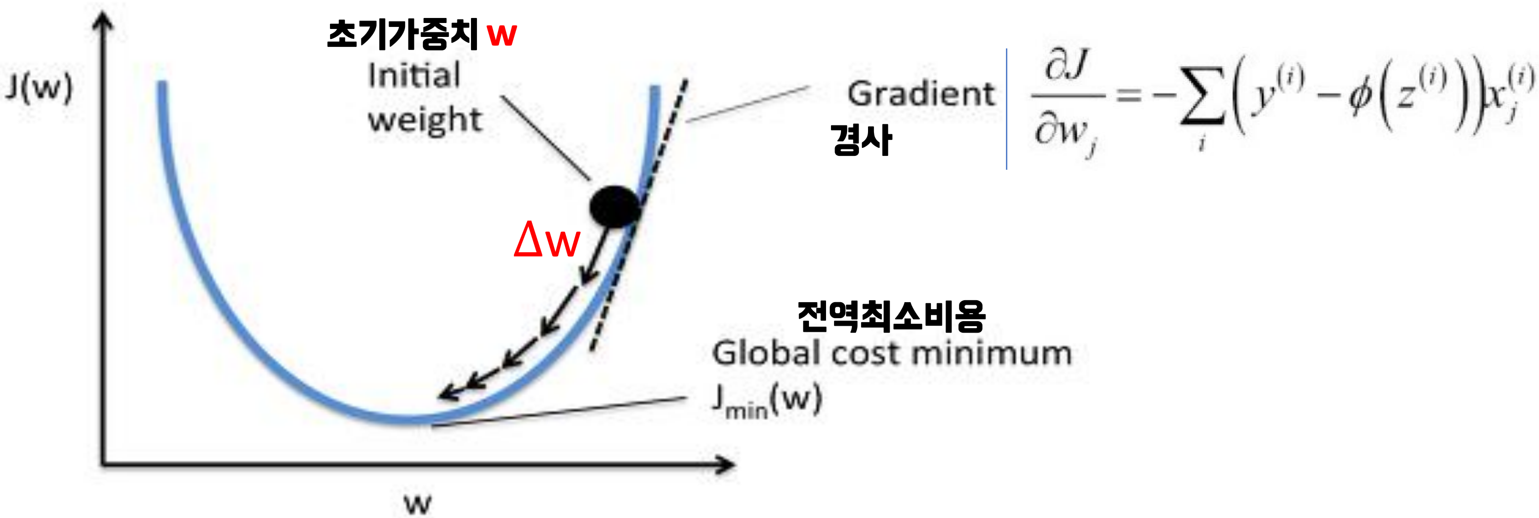


경사 하강법을 통해 비용함수의 경사의 반대편으로 조금씩 움직여 가중치를 업데이트한다.
진행 크기는 경사의 기울기와 학습률로 결정.

2.3.1 경사하강법으로 비용 함수 최소화

가중치 업데이트 $\boldsymbol{w} := \boldsymbol{w} + \Delta \boldsymbol{w}$

$$\Delta \boldsymbol{w} = -\eta \nabla J(\boldsymbol{w})$$

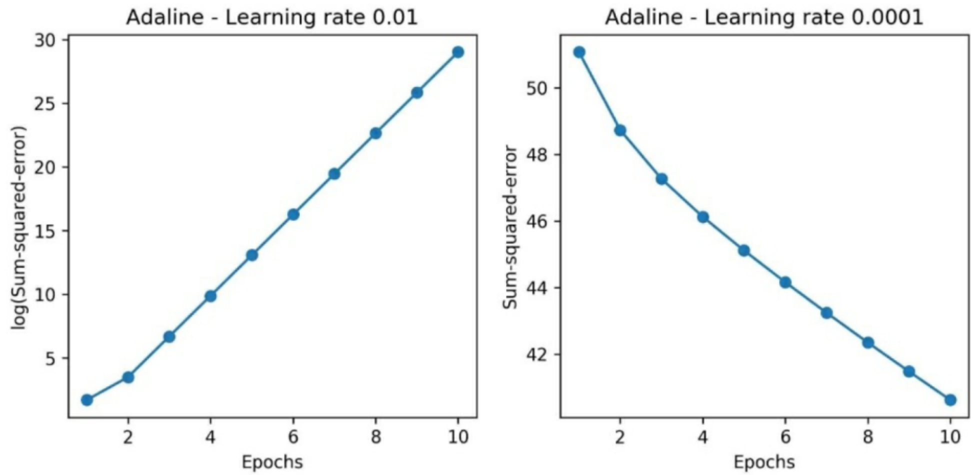


$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)x_j^{(i)}$$

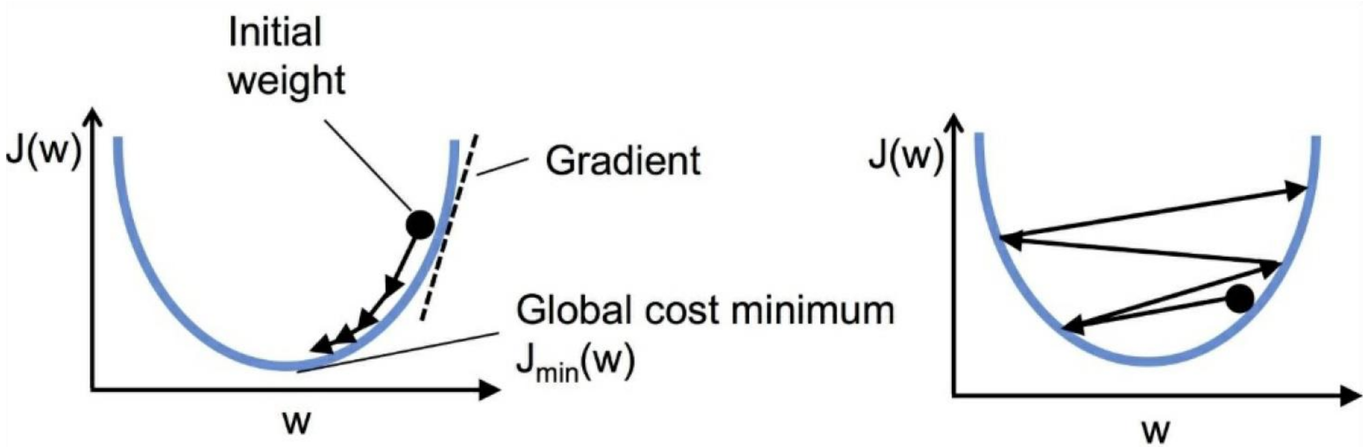
2.3.1 경사하강법으로 비용 함수 최소화

교재 71p

학습률에 따른 아달린 알고리즘의 수렴 $\Delta w = -\eta \nabla J(w)$



학습률이 너무 클 때 : 비용함수를 최소화하지 못하고 오차는 에포크마다 커짐
학습률이 너무 작을때 : 최소값에 수렴하려면 아주 많은 에포크 필요

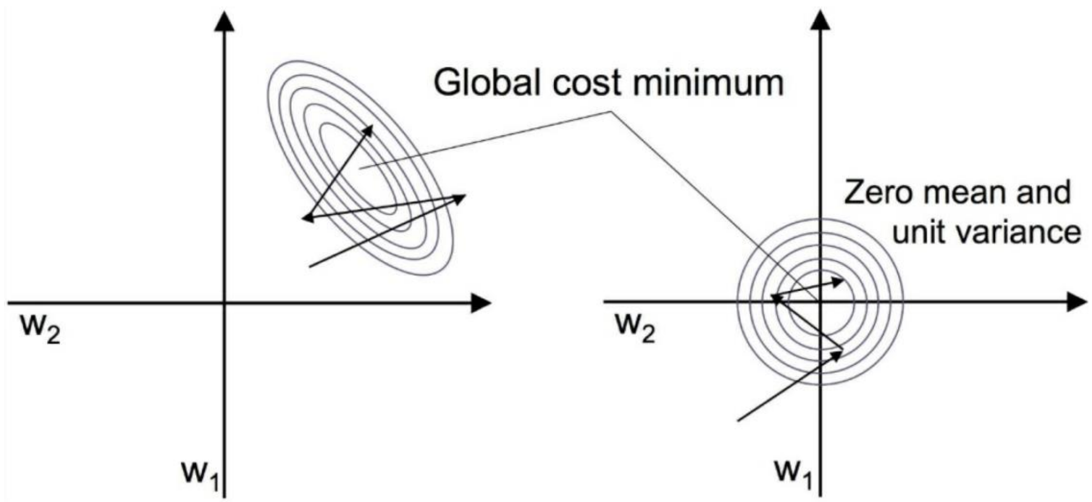


적절한 학습률 너무 큰 학습률

2.3.3 특성 스케일 조정하여 경사하강법 결과향상

표준화 $$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

: 데이터에 표준 정규분포의 성질을 부여하여 경사 하강법 학습이 좀 더 빠르게 수렴되도록 도움

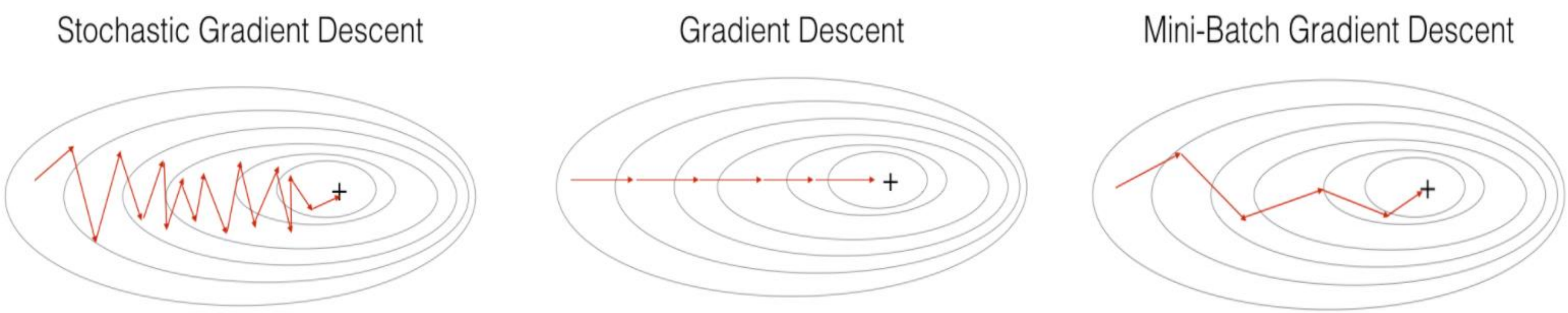


▲표준화 후

2.3.4 대규모 머신러닝과 확률적 경사 하강법

확률적 경사 하강법 (Stochastic Gradient Descent)
: (배치)경사하강법의 다른 대안으로, 모든 샘플 x에 대해 누적된 오차의 합을 기반으로 가중치를 업데이트 하지 않고, 각 훈련 샘플에 대해서 조금씩 가중치를 업데이트 함.

>가중치가 더 자주 업데이트되므로 수렴속도가 빠름
>하나의 훈련 샘플을 기반으로 계산되어 오차의 궤적이 어지러움(노이즈심함)



미니 배치 학습
: 확률적 경사하강법과 배치 경사 하강법의 절충점
하나의 훈련 샘플이 아닌 여러 개의 훈련 샘플로 경사 하강법을 적용.

>수렴속도가 빠름.
>확률적 경사 하강법에 비해서 계산 효율성이 향상되고, 노이즈가 적음

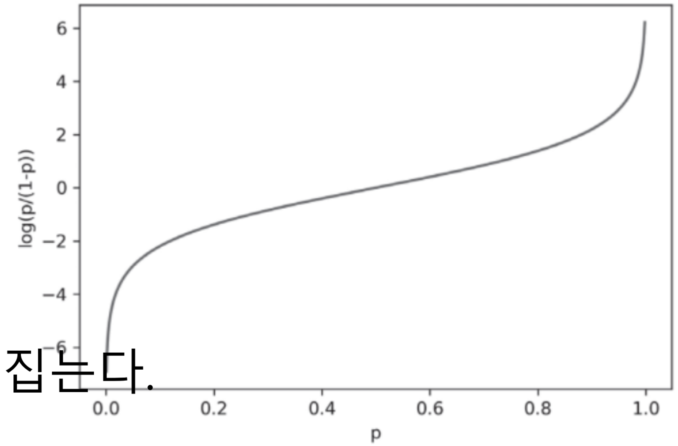
3.3. 로지스틱 회귀

오즈비(odds ratio) $\dfrac{p}{1-p}$

Ex) 질병이 발생할 확률(p)이 질병이 발생하지 않을 확률의 몇 배인지

오즈비에 로그를 취해 로짓함수를 만든

$$logit(p) = log\frac{p}{(1-p)}$$



어떤 샘플이 특정 클래스에 속할 확률을
예측하는 것이 목적이므로 로짓함수를 뒤집는다.

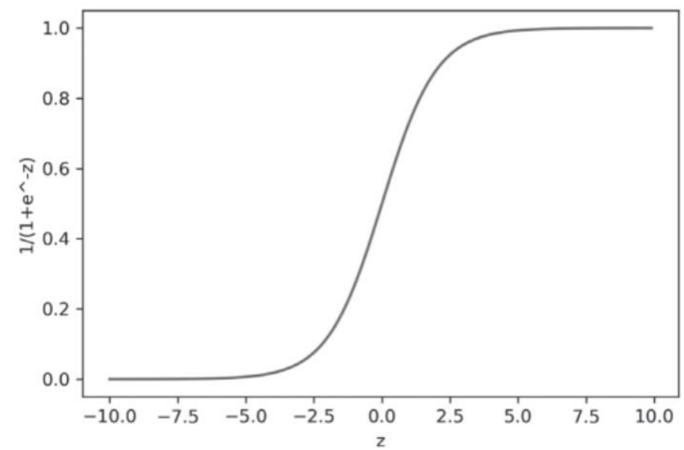
$$log(\frac{p}{1-p}) = z$$

$$\frac{p}{1-p} = e^z$$

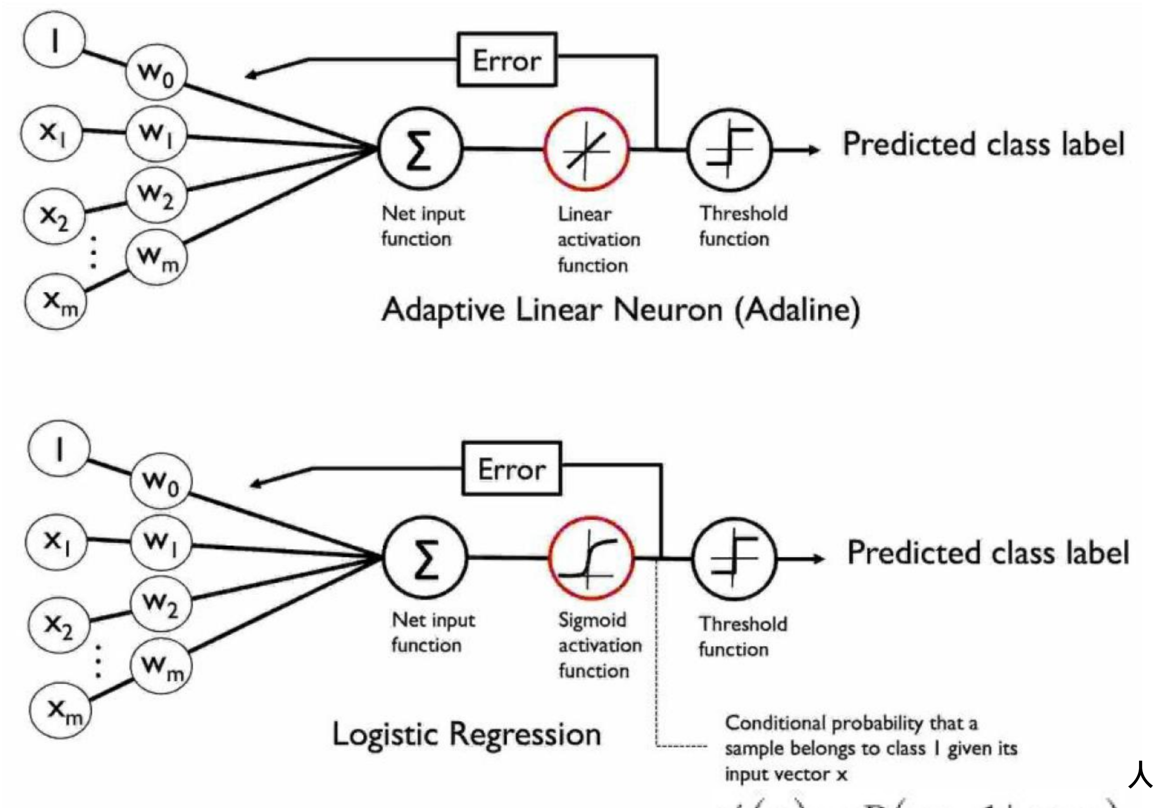
$$p(1+e^z) = e^z$$

$$p = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$ 시그모이드 함수

$$z = w^T x = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m$$



3.3. 로지스틱 회귀



시그모이드 함수의 출력

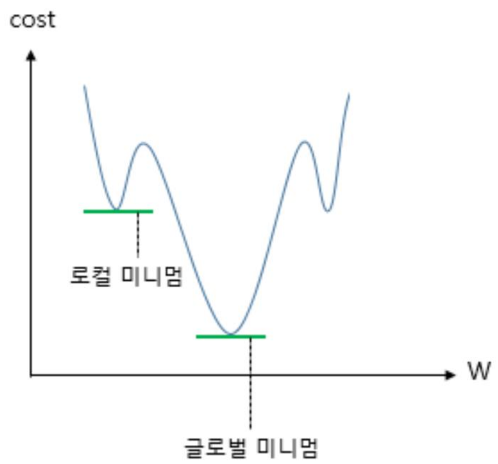
$\phi(z) = P(y=1 \mid \boldsymbol{x}; \boldsymbol{w})$: 특정 샘플이 클래스1에 속할 확

$$\hat{y} = \begin{cases} 1 & if\, \phi(z) \geq 0.5 \\ 0 & otherwise \end{cases}$$

시그모이드 함수에서 나온 확률값을 임계함수를 사용하여
간단하게 이진 출력으로 바꿈

3.3.2 로지스틱 비용함수의 가중치 학습

로지스틱 회귀에서 경사하강법



로지스틱 회귀에서는 잘못된 최소값에 빠질 수 있기 때문에 SSE를 쓰지 않는다.

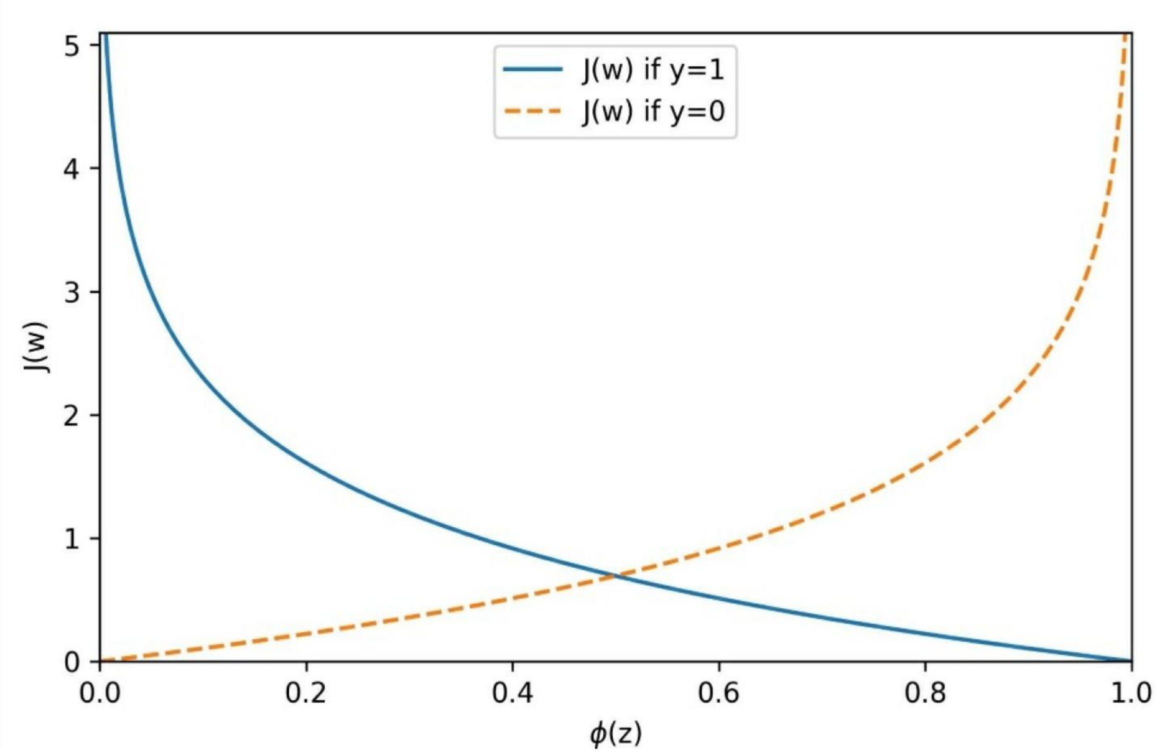
가능도 함수에 로그를 취한 로그 가능도 함수를 이용한다.

$$L(\boldsymbol{w}) = P(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w}) = \prod_{i=1}^{n} P\left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{w}\right) = \prod_{i=1}^{n} \left(\phi\left(z^{(i)}\right)\right)^{y^{(i)}} \left(1-\phi\left(z^{(i)}\right)\right)^{1-y^{(i)}}$$

$$\boxed{J(\mathbf{w}) = -\sum [y^{(i)} log(\phi(z^{(i)})) + (1-y^{(i)}) log(1-\phi(z^{(i)}))]}$$

3.3. 로지스틱 회귀

$$J(\phi(z), y; \boldsymbol{w}) = \begin{cases} -\log(\phi(z)) & if\ y=1 \\ -\log(1-\phi(z)) & if\ y=0 \end{cases}$$



실제값이 1일때 1로 예측을 하면 비용이 0 에 가깝고
실제값이 0 일때 0으로 예측할경우 비용이 0 에 가깝다.
> 잘못된 예측에 점점 더 큰 비용을 부여하는 비용함수!

3.3.3. 아달린 구현을 로지스틱 회귀 알고리즘으로 변경 **교재** 95p

비용함수를 바꾸면 로지스틱 회귀로 구현할 수 있음.

$$J(\mathbf{w}) = -\sum [y^{(i)} log(\phi(z^{(i)})) + (1 - y^{(i)}) log(1 - \phi(z^{(i)}))]$$

가중치 업데이트는 아달린에서와 동일한 규칙으로 적용하면 됨.

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

$$\boldsymbol{w} := \boldsymbol{w} + \Delta \boldsymbol{w}$$

3.3.5 규제를 사용해 과대적합 피하기

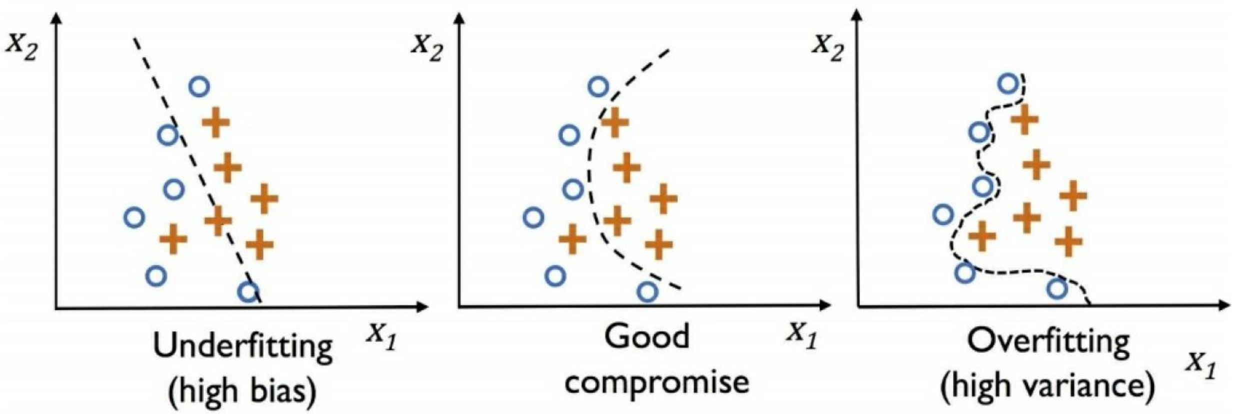
교재 102p

과대적합

훈련 데이터에만 너무 적합해서 일반화되지 않는 현상
너무 복잡한 모델을 만들기 때문에 분산이 크다

과소적합

모델이 너무 단순해서 훈련 데이터의 패턴을 감지하지 못



3.3.5 규제를 사용해 과대적합 피하기

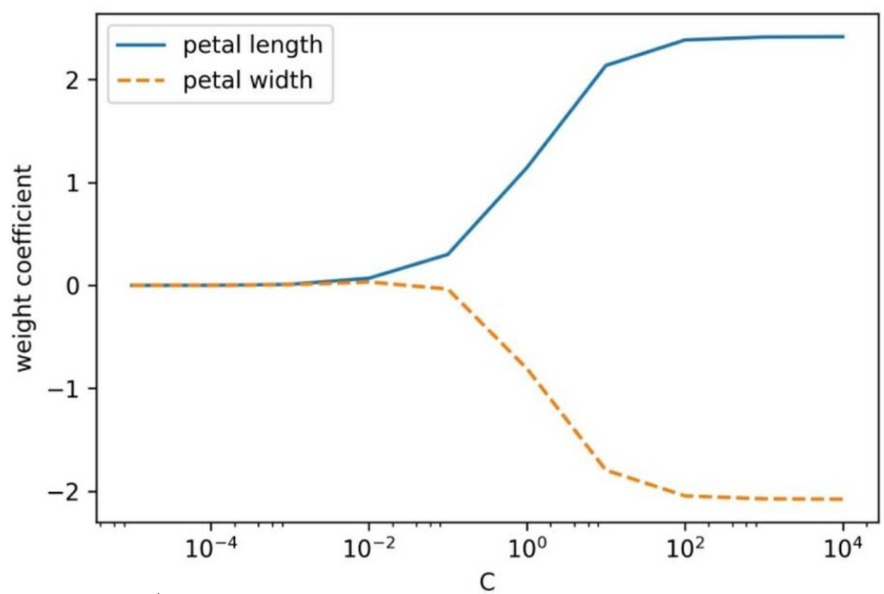
규제 : 모델을 단순하게 하고 과대적합의 위험을 감수시키기 위해 모델에 제약을 가하는 것

$$\frac{\lambda}{2}\|\boldsymbol{w}\|^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

λ : 규제 하이퍼파라미터

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left[-y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)\right] + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

λ 값을 증가하면 규제 강도가



람다값이 커질수록 가중치의 절댓값이 작아진다
= 규제 강도가 증가