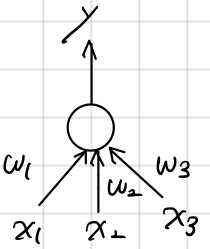
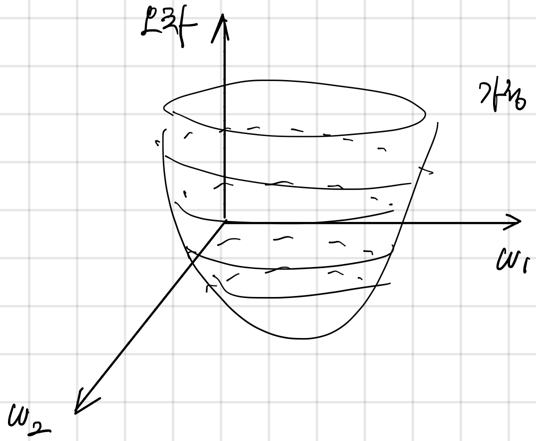
2.1 오차함수 (error function)

$$E = \frac{1}{2}\sum_i \left(t^{(i)} - y^{(i)}\right)^2$$



$$y^{(i)} = w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)}$$

i번째 뉴런의 출력



가정 ① $\vec{w} = (w_1, w_2)$, ② 선형뉴런

$$E = \frac{1}{2}\sum_{i=1}^{n}\left[t^{k} - (w_1 x_1 + w_2 x_2 + w_0)\right]^2$$

n : 해당 층의 뉴런개수

이게 원통 모양이던데 ...

$ax^2 + by^2 = z$

→ 이게 원통이지...

xy 항은 왜 사라지는 건가?

2.3 델타 규칙과 학습률 (delta rule, learning rate)

가중치 ← 경사로 따라 + 학습률 (ϵ, learning rate)

경사 $= \dfrac{\partial E(\vec{w})}{\partial \vec{w}}$: 가중치에 대한 오차함수의 편미분함수

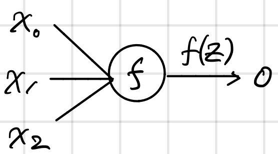
$$\Delta w_k = -\epsilon \frac{\partial E}{\partial w_k} = -\epsilon \frac{\partial}{\partial w_k}\left[\frac{1}{2}\sum_{i=1}\left(t^{(i)} - y^{(i)}\right)^2\right]$$

0 ~ 1

$$= \epsilon \sum_{i=1}\left\{t^{(i)} - y^{(i)}\right\}\cdot\frac{\partial y^{(i)}}{\partial w_k} = \sum_{i=1}\epsilon x_i \left(t^{(i)} - y^{(i)}\right)$$

비선형 vs 선형

2.4 시그모이드 뉴런의 경사 하강법



$z = \sum_{k=1}^{n} w_k \cdot x_k$ (n: 입력의 개수) : 로짓함수

$f(z) = \dfrac{1}{1+e^{-z}}$: 시그모이드 함수

$$\Delta w_k = -\epsilon\frac{\partial E}{\partial w_k} = -\epsilon\frac{\partial}{\partial w_k}\left[\frac{1}{2}\sum_{i=1}^{n}\left(t^{(i)} - f(x)^{(i)}\right)^2\right]$$

(n: 한 층에 존재하는 뉴런의 개수)

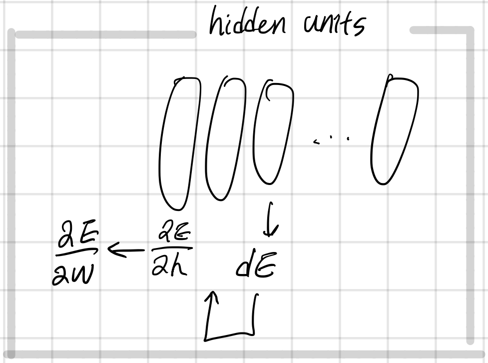
$$= \epsilon\sum_{i=1}\left(t^{(i)} - f(x)\right)\cdot\frac{\partial f(x)}{\partial w_k} = \sum_{i=1}^{n}\epsilon x_k^{(i)} y^{(i)}\left(1-y^{(i)}\right)\left(t^{(i)} - y^{(i)}\right)$$

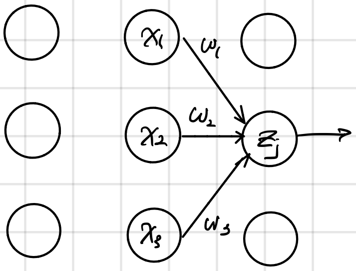
$$\frac{\partial f}{\partial w_k} = \sum_{k=1}^{n}\frac{\partial f}{\partial z}\frac{\partial z}{\partial w_k} = x_k\frac{e^{-z}}{(1+e^{-z})^2} = x_k y(1-y)$$

2.5 역전파 알고리즘 (backpropagation)

- hidden unit에서 활성함수를 바꿀 때 얼마나 빨리 오차가 변하는지 계산할 수 있다
➡️ 개별 가중치를 바꿀 때 오차가 얼마나 빠르게 변하는지 알 수 있다

Error Derivative





$\rightarrow y_j : h(z_j)$

where $z = \sum_{u=1}^{3} w_u x_u$: Logit

출력층에서 오차 도함수 $E = \frac{1}{2}\sum_{j \in output}(t_j - y_j)^2$

$$\frac{\partial E}{\partial y_j} = -(t_j - y_j)$$

귀납법 j번째 층에 대한 오차 도함수를 가정,
j-1 (i) 번째 층에 대한 오차 도함수를 계산해보자.

① $\frac{dE}{dx_j} = \sum_j \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_j} = \sum_j W_{ij}\frac{\partial E}{\partial z_j}$ ② $\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{dy_j}{dz_j} = y_j(1-y_j)\frac{\partial E}{\partial y_j}$

$\frac{\partial z_i}{\partial x_j} = W_{ij}$

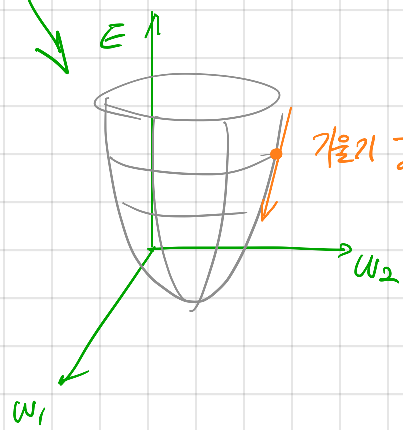
활성화함수를 시그모이드로 가정

$\frac{dy_j}{dz_j} = y_j(1-y_j)$

① + ② ⇒ $\frac{\partial E}{\partial y_i} = \sum_j W_{ij}\frac{\partial E}{\partial z_j} = \sum_j W_{ij}\, y_j(1-y_j)\frac{\partial E}{\partial y_j}$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{\partial z_j} = y_i y_j(1-y_j)\frac{\partial E}{\partial y_j}$$

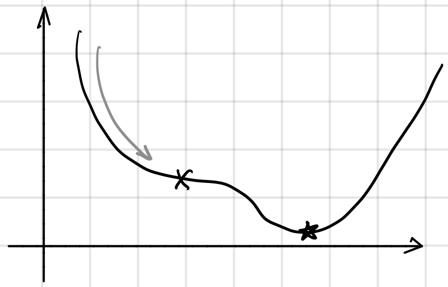
$$\Delta W_{ij} = -\sum_{k \in output} \epsilon\, y_i^{(k)} y_j^{(k)}(1-y_j^{(k)})\frac{\partial E^{(k)}}{\partial y_j^{(k)}}$$



기울기 $\frac{\partial E}{\partial w_{ij}}$: 가장 가파른 하강 경로를 따라 다음 가중치를 설정

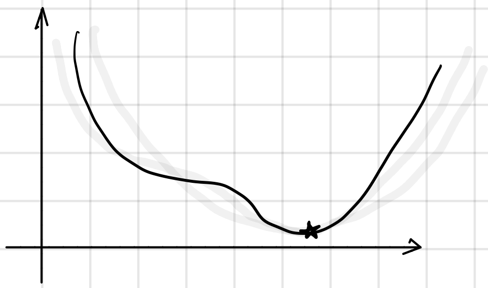
2.6 확률적 경사 하강법(stochastic gradient descent)과 미니배치 경사 하강법 (batch gradient descent)

pp.42 "배치경사 하강법이 안장점에 민감하다"
여기에만? local_minimum은?



➡️ 단일가중치 설정 후 경사하강법을 사용했을 때
발생 가능한 문제점 : local-minimum을 찾아서 고립될 수 있음

해결책 1. 확률적 경사 하강법



➡️ 단일 정적 오차 곡면X, 동적 오차 곡면O
단점 : 시간 오래걸림

2.7 테스트 데이터와 검증 데이터 그리고 과적합

답러닝의 딜레마적 문제 : 많은 뉴런 X 수십개의 층 = 과적합 문제 vs 과도하게 일반화

1. 딥러닝은 복잡한 문제로 아주 복잡한 문제를 풀고 과적합 방지를 위해 추가 대책을 적용
2. 학습 데이터와 테스트 데이터를 나눈다.
3. 과적합이 시작되자마자 학습을 멈추는 것이 좋다. ➡️ 학습 과정을 epoch으로 나눈다
   - 학습 데이터 크기가 d, 미니배치 경사하강법의 배치크기를 d라고할 때 d/b번의 epoch이 수행
   - 각 epoch 끝에 모델이 얼마나 잘 학습되었는지 검증 데이터 세트를 통해 평가한다
4. 하이퍼파라미터 튜닝 : gridsearch

2.8 신경망에서 과적합 막기

1. 정형화 Regulation (규제)
- 중요한 피처의 가중치 값을 늘리는 항을 추가

$$objective = error + \lambda f(\theta)$$

하이퍼 파라미터

θ : 규제X
시행착오 필요

1) L2 정형화

$$E_{L2} = E + \frac{1}{2}\lambda w^2$$

큰값은 조금 크게, 작은 값은 조금 작게…

- 경사하강법이 진행됨에 따라 결국 모든 가중치가 선형적으로 0으로 감소하기 때문에 일반적으로 가중치 감쇠(weight decay)라고 불림

2) L1 정형화

$$E_{L1} = E + \lambda|w|$$

- 최적화 하는 동안 가중치 벡터를 드문드문하게 만듬
- 입력 노이즈에 매우 잘 견딤
- 어떤 피처가 예측 결과에 기여하고 있는지 이해하고자 할 때 유용