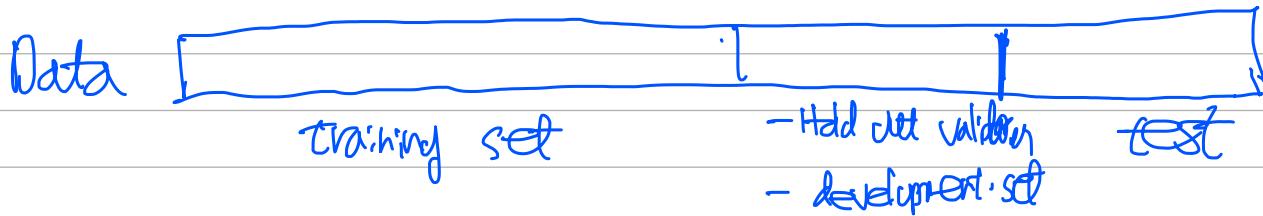


Setting up your ML application

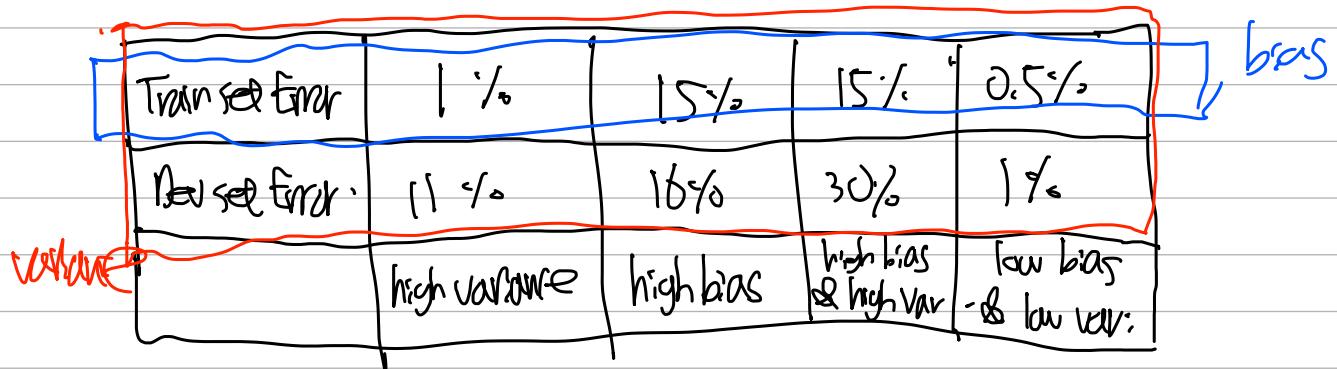
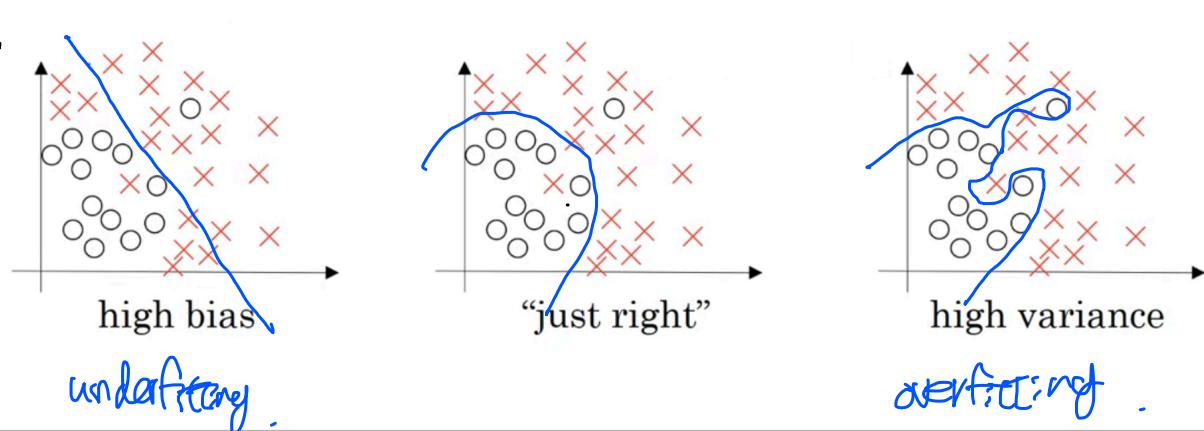
Train / dev / test sets



- Dev set의 목표는 다양한 알고리즘을 테스트하고 어떤 알고리즘이 잘 작동하는지 확인하는 것.
- Test set은 Dev set으로 만든 최종 모델이 얼마나 잘 수행되는지 확인하는 것.
- 데이터 set의 수가 적다면: 6:2:2
데이터 set의 수가 많으면: 98:1:1 or 95:0.25:0.25
- dev set과 test set은 동일한 distribution을 가져야 한다.
 - 예를 들어, train/dev의 60 이미지는 인터넷에서 수집한 이미지라면, test는 사용자가 휴대폰으로 양 goede 걸 사진한 경우, distribution은 일치하지 않는다.

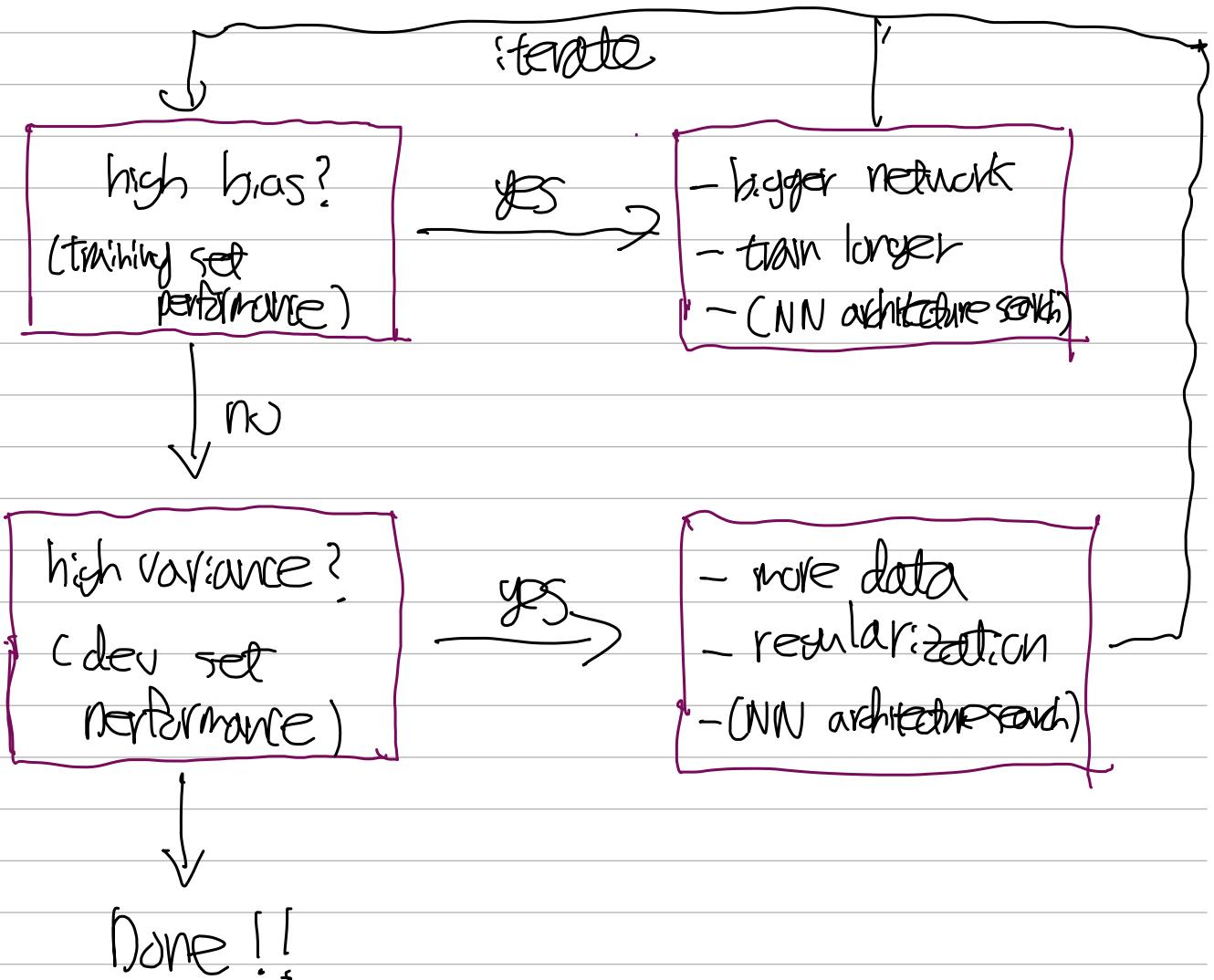
Bias and variance

- Bias / Variance techniques are easy to learn, but difficult to master.



- High Variance : Training set은 잘 풀었는데, dev set과 error의 gap이 너무 큼.
- High bias : Train error가 꽤 큰 것은 train 문제를 잘 풀지 못한다는 뜻.
- High bias & variance : 최악의 경우, 두 error 사이의 gap도 크고, error 자체도 크다.
- Low bias & variance : Ideal한 경우.

Basic recipe for machine learning



- Neural Network가 high bias 상태면 Network가 너무 simple한 것 같으면 parameter를 늘려 bigger network로 만들어야 함.
- 위 과정을 반복하면 overfitting 가능성이 높아지면서 variance 가 높아진다.
- Variance가 높아져 high variance 상태가 되면 이 때, dataset을 늘린다. Dataset을 늘리는 방식은 simple하고 효과가 좋지만, 데이터를 구하는 데 시간과 돈이 많이 드는 단점.

Regularizing your neural network

Regularization \Rightarrow Overfitting을 막고 Variance 2 줄인다.

Logistic Regression

λ : regularization parameter

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \|w\|_2^2 + \frac{\lambda}{2m} b^2.$$

~~b~~
OK!

$$L_2 \text{ Regularization} : \|w\|_2^2 = \sum_{j=1}^n w_j^2 = w^T w$$

$$L_1 \text{ Regularization} : \|w\|_1 = \sum_{j=1}^n |w_j| \rightarrow \text{spike } w$$

• Regularization for NN.

$$J(W^{[0]}, b^{[0]}, \dots, W^{[L]}, b^{[L]}) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^L \|w^{[l]}\|^2$$

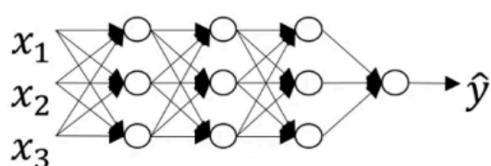
$$\|w^{[l]}\|^2 = \sum_{i=1}^{n^{[l+1]}} \sum_{j=1}^{n^{[l]}} (w_{ij}^{[l]})^2$$

$$dw^{[l]} = \underbrace{(\text{from backpropagation})}_{\rightarrow W^{[l]}} + \frac{\lambda}{m} w^{[l]}$$

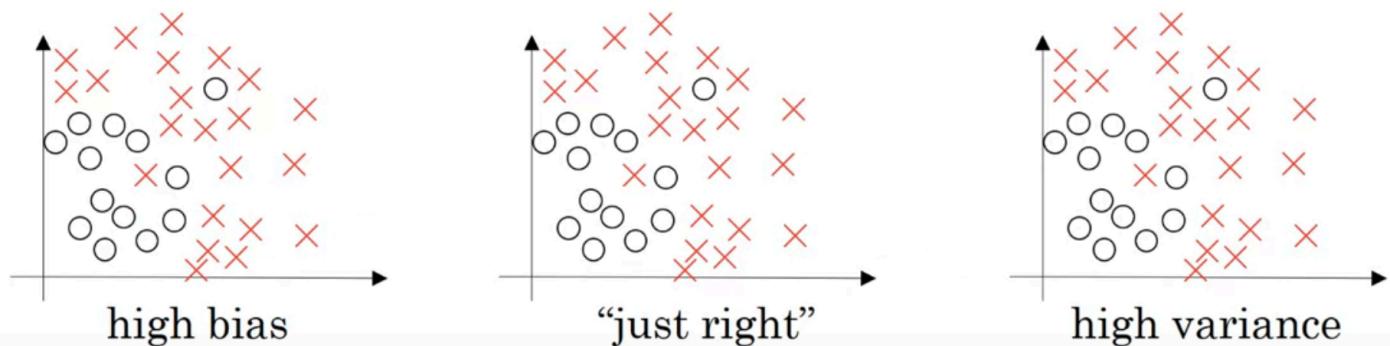
$$\rightarrow W^{[l]} = W^{[l]} - \alpha dw^{[l]}$$

$$\begin{aligned} \text{Weight Decay} : W^{[l]} &= W^{[l]} - \alpha \left((\text{from backprop}) + \frac{\lambda}{m} w^{[l]} \right) \\ &= \left(1 - \frac{\alpha \lambda}{m}\right) w^{[l]} - \alpha (\text{from backprop}) \end{aligned}$$

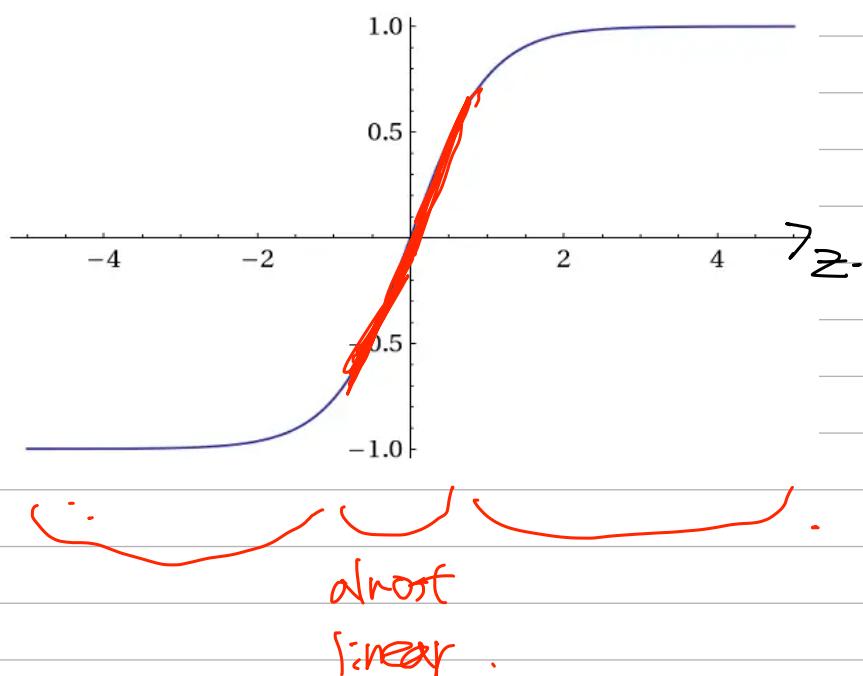
How does regularization prevent overfitting?



$$J(w^{(0)}, b^{(0)}, \dots, w^{(L)}, b^{(L)}) = \frac{1}{m} \sum L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m} \sum_{f=2}^L \|W_f\|_F^2$$



- λ 를 매우 큰 값으로 적용시키면 W 가 0에 가까워짐.
 - W 가 0에 가까워지면서 빠른 네트워크를 줄일 수 있다.

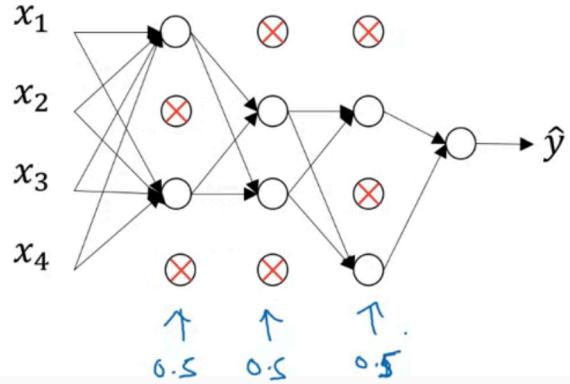
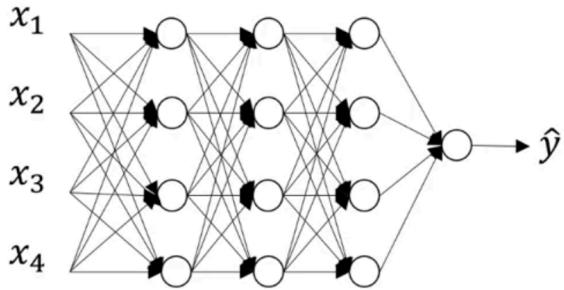


$$\lambda \uparrow \rightarrow W^{[l]} \downarrow \rightarrow z^{[l]} \downarrow \quad (z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]})$$

\Rightarrow every layers \cong linear.

- λ 가 0이거나 양수면 그라프가 직선에 가까워지기 때문에 non-linearity 줄 수 있다. 의 의미가 없어짐.

Dropout regularization



- Dropout regularization은 확률을 기반으로 각 training에서 neurons/weigths를 제거한다.
- 50개의 unit이 있고 그 중 10개가 - dropout rate 제가 됐다면 activation 결과도 20% 줄어든다.
이를 위해 $\alpha_3 = \text{keep prob}$, $\text{keep prob} = 0.8$ 로 단순 scaling 해준다.
- 위 과정을 훈련 test 때는 dropout 없이 전부 사용할 수 있음.
- Dropout은 모든 unit이 랜덤하게 자워질 수 있기 때문에 W값이 작아진다. → 값이 분산, → 특정 unit이 중요한 일을 맡을 수가 없다.
→ 따라서 한 곳에 집중하지 않고 여러 곳에 분산시켜 문제를 풀수 있다.
→ 특정 unit의 W값이 엄청 작거나, 커지는 것을 방지.
- Output layer에는 dropout을 적용하지 않는다.

Other regularization methods

Data augmentation



4



4

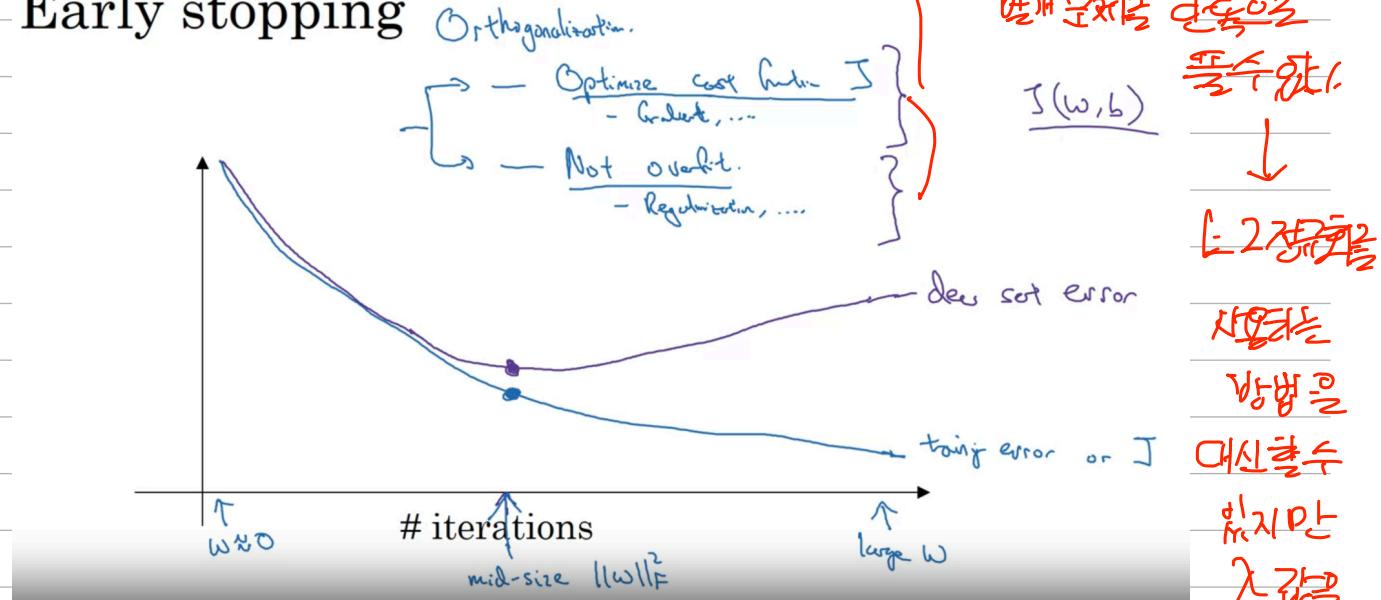
4

4

- Augmentation을 통해 Data set을 늘리는 방식 .
- Image를 변형해서 쉽게 Data set을 늘릴 수 있다 .
- 새로운 Image를 추가하는게 더 좋기 하지만, 쉽게 사용될 수 있는 방법들 .

Early stopping

Early stopping

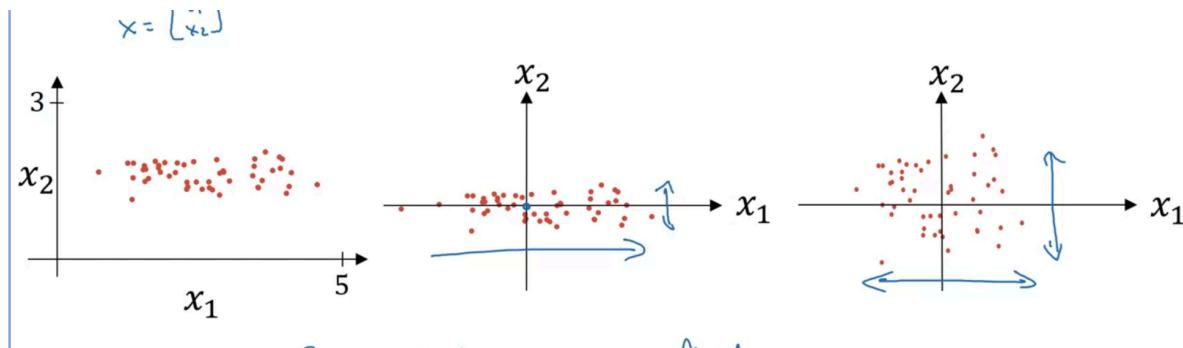


- training은 loss function의 값이 무조건 감소한다.
- 하지만, dev set의 경우는 줄어들다가 어느 순간부터 늘어날 수 있다. \rightarrow Overfitting.
- training의 경우에도 overfitting.
- training error와 dev set error 간의 격차가 커지는데 이를 high variance problem.

Setting up your optimization

Normalizing inputs

- 신경망 훈련 속도를 높이기 위한 technique 중 하나는 항목을 정규화하는 것.

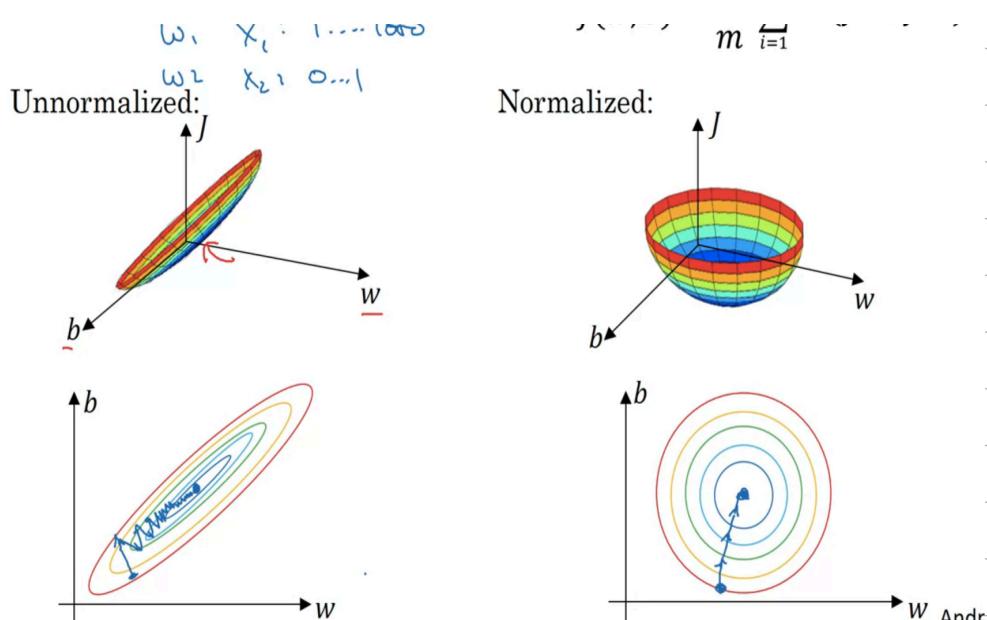


$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m \{x_j^{(i)}\}^2$$

$$x^{(i)} = x^{(i)} - \mu$$

$$x_j^{(i)} = \frac{x_j^{(i)}}{\sigma_j}$$

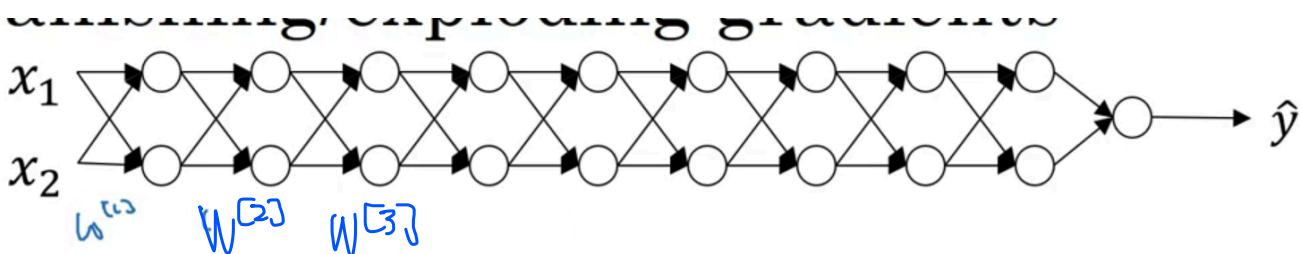


normalized을 하지
않으면 data가 평행으로
증재.
normalize를 하면
수렴이 빠르게 가장 잘은
지점이 도달할 수 있다.

normalize를 하지 않으면, 큰 x_1 과 작은 x_2 가 종재할 때,
 w_1 은 조금만 커져도 x_1 에 의해 그 차이가 매우 커지고, w_2 는 x_2 로
인해 값이 크게 바뀌어야 전파 경로에 영향을 미칠 수 있다 \rightarrow 훈련 문제를
해결하기 위해 정규화

Vanishing / exploding Gradients

- Vanishing gradient : w 가 0에 가까울 때 gradient를 거의 update 할 수 없거나 output이 지나치게 작은 값이 될 수 있다.
- exploding gradient : w 가 너무 크면 output이 너무 커질 수 있다. update는 많이 되지만, 너무 과하게 update



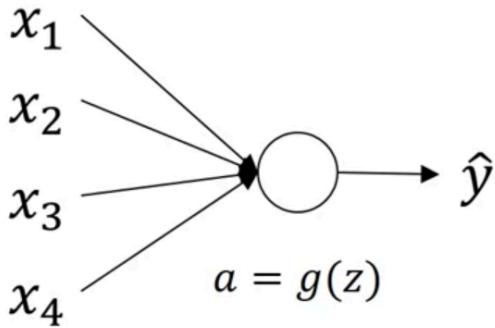
- Backpropagation은 뒤에서부터 계산하기 때문에 뒤쪽에서는 문제가 잘 발생하지 않다가도 update하면서 앞쪽으로 도달하면, w 값이 증정되면서 vanishing/explode gradient 문제 발생 가능.

$$W^{[L]} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}^L \quad x = \begin{bmatrix} 1.5^L & 0 \\ 0 & 1.5^L \end{bmatrix} \quad \Rightarrow \text{activations increase exponentially}$$

$$W^{[L]} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}^L \quad x = \begin{bmatrix} 0.5^L & 0 \\ 0 & 0.5^L \end{bmatrix} \quad \Rightarrow \text{activations decrease exponentially}$$

Weight initialization for deep network

Single neuron example



$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n + b \quad \text{학습 목표}$$

if large $n \rightarrow$ smaller w_i

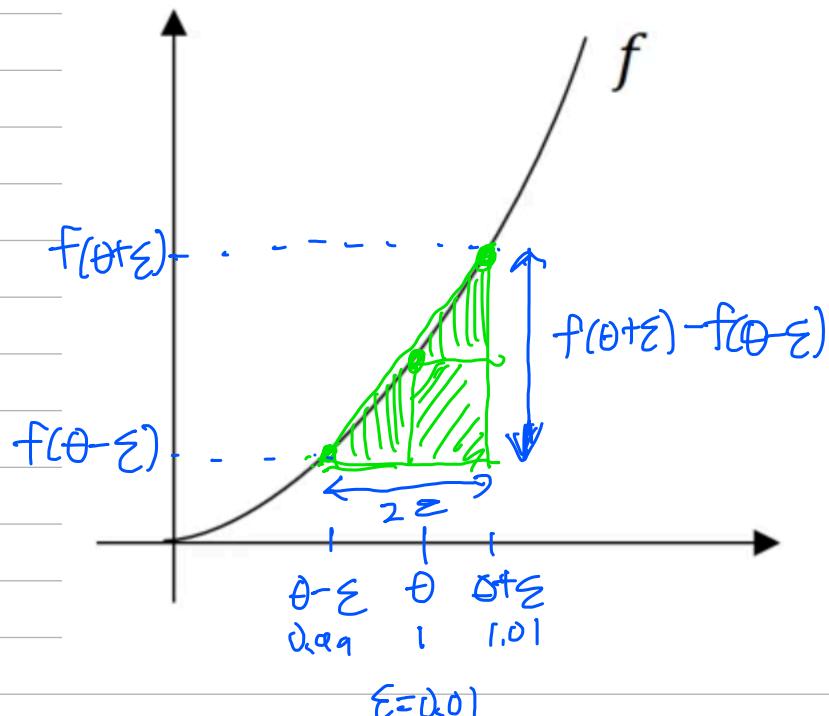
$$\text{Var}(w_i) = \frac{1}{n}$$

$$w[i] = np.random.randn(\text{shape}) * np.sqrt(\frac{1}{n^{[i-1]}}).$$

- w 를 initialize 할 때, input이 많으면 weighted sum z 가 양정 축소도 있고 input이 적으면 적어질 수도 있어서 input 개수에 따라 w 초기값을 다르게 한다.

Numerical Approximation of gradient

$$f(\theta) = \theta^3$$



$$\frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon}$$

$$= \frac{(1.01)^3 - (0.99)^3}{2(0.01)} = 3.001 \approx 3.$$

$$g(\theta) = 3\theta^2 = 3$$

$$\text{approx error} = 0.0001$$

- Gradient checking은 gradient를 approximate하고, 허용 범위를 찾는데 유용하지만, gradient descent 를 찾기 어렵다.

- Gradient Checking.

$$- J(w^{[0]}, b^{[0]}, \dots, w^{[L]}, b^{[L]}) = J(\theta)$$

$$- dw^{[0]}, db^{[0]}, \dots, dw^{[L]}, db^{[L]}$$

- Algorithm.

```
eps = 10^-7 # small number
for i in len(theta):
    d_theta_approx[i] = (J(theta1, ..., theta[i] + eps) - J(theta1, ..., theta[i] - eps)) / 2*eps
```

$$\frac{\|\theta_{\text{approx}} - \theta\|_2}{\|\theta_{\text{approx}}\|_2 + \|\theta\|_2}$$

$\leq 10^{-1}$: great
 $\approx 10^{-5}$: ok but need to inspect.
 $\geq 10^{-3}$
 If there are no particular big values
 → Bad.
 Probably.
 There is a bug. in d θ_{approx}
 or d θ .
 in backpropagation implementation

Gradient Checking Implementation notes.

- training set 사용 $X \rightarrow$ 편집
- Debugging 시도만 사용
- 알고리즘이 gradient check를 실패하면 그 이유를 살펴보기
- Don't forget add $\frac{\lambda}{2m} * \sum W^2$ to J if you are using L1 and L2 regularization
- Gradient check doesn't work with dropout because J is not consistent.