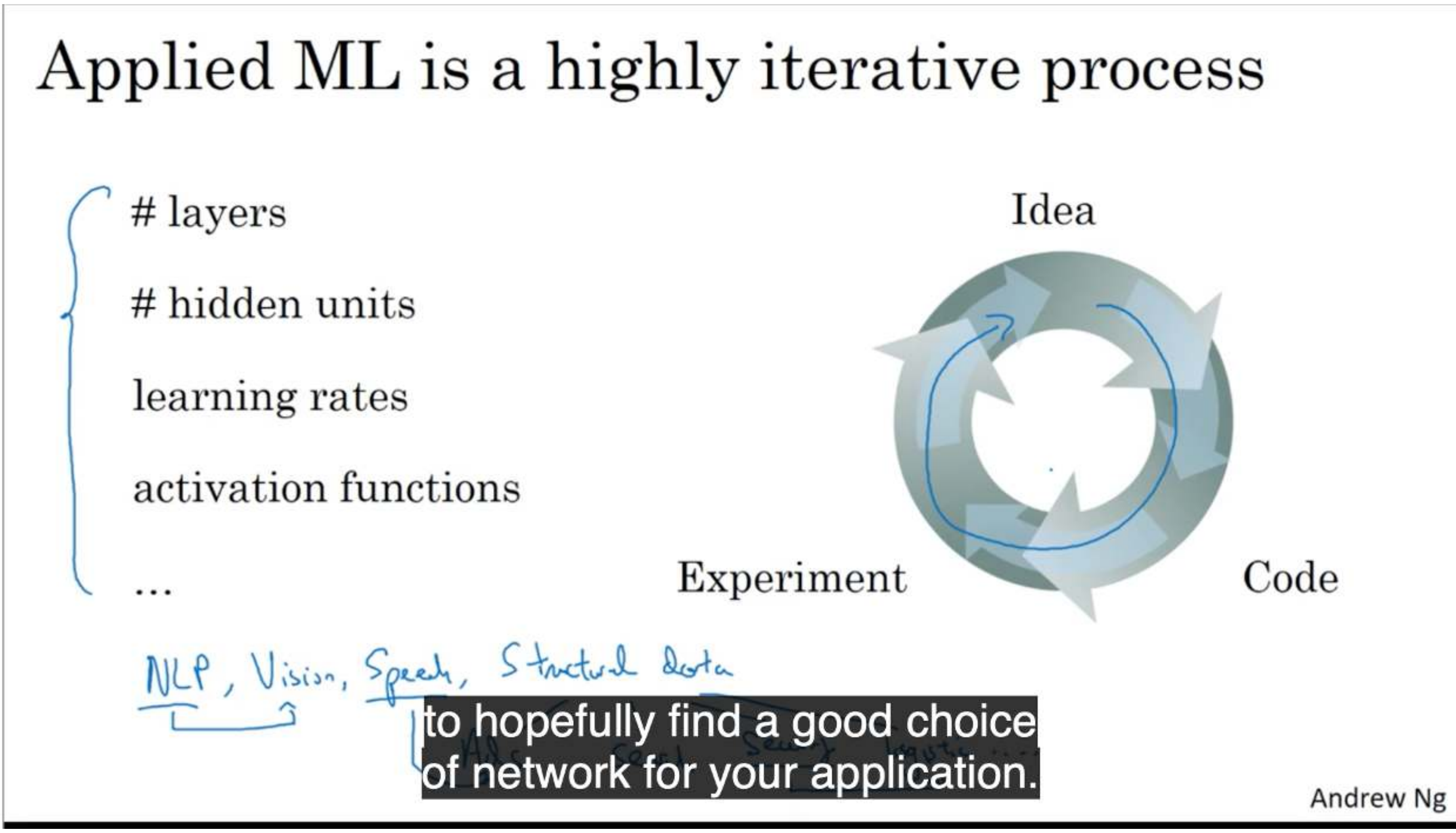
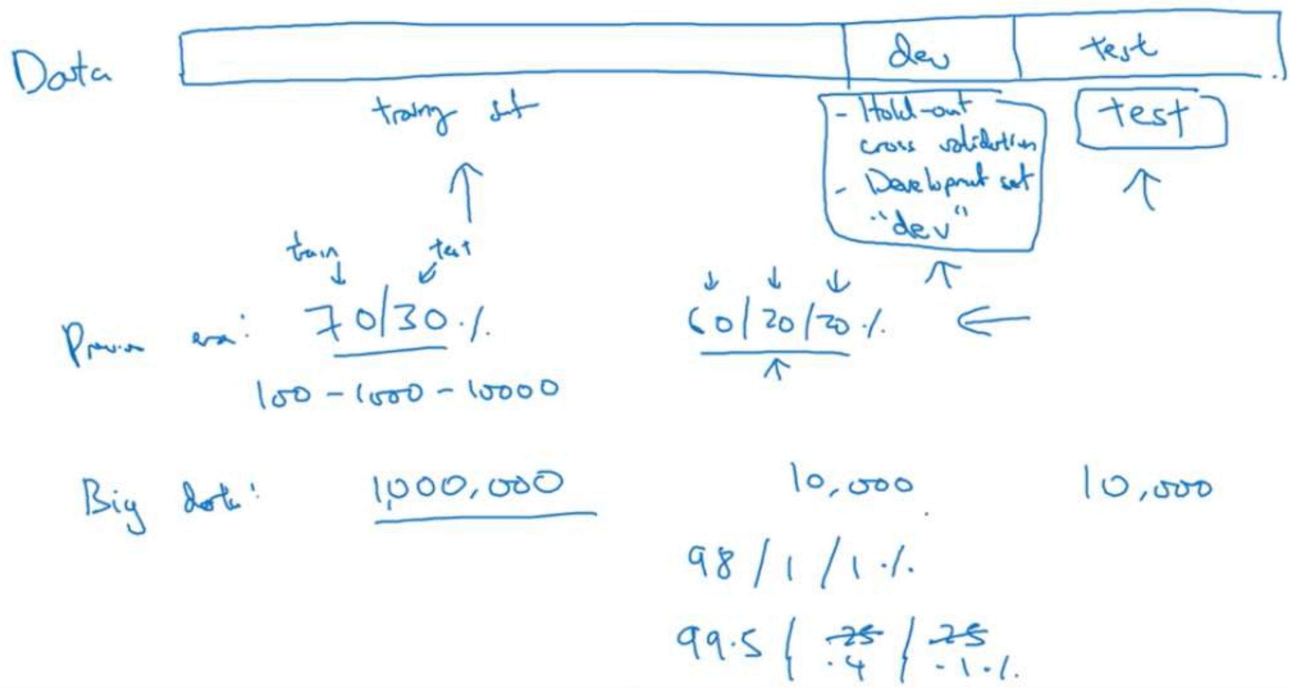

Applied ML is a highly iterative process
layers
hidden units
learning rates
activation functions
...
Idea
Code
Experiment
NLP, Vision, Speech, Structural Data
to hopefully find a good choice
of network for your application.
Andrew Ng

Train/dev/test sets



Andrew Ng

Mismatched train/test distribution

Cats

Training set:
Cat pictures from webpages

Dev/test sets:
Cat pictures from users using your app

→ Make sure dev and test come from same distribution.

"test"

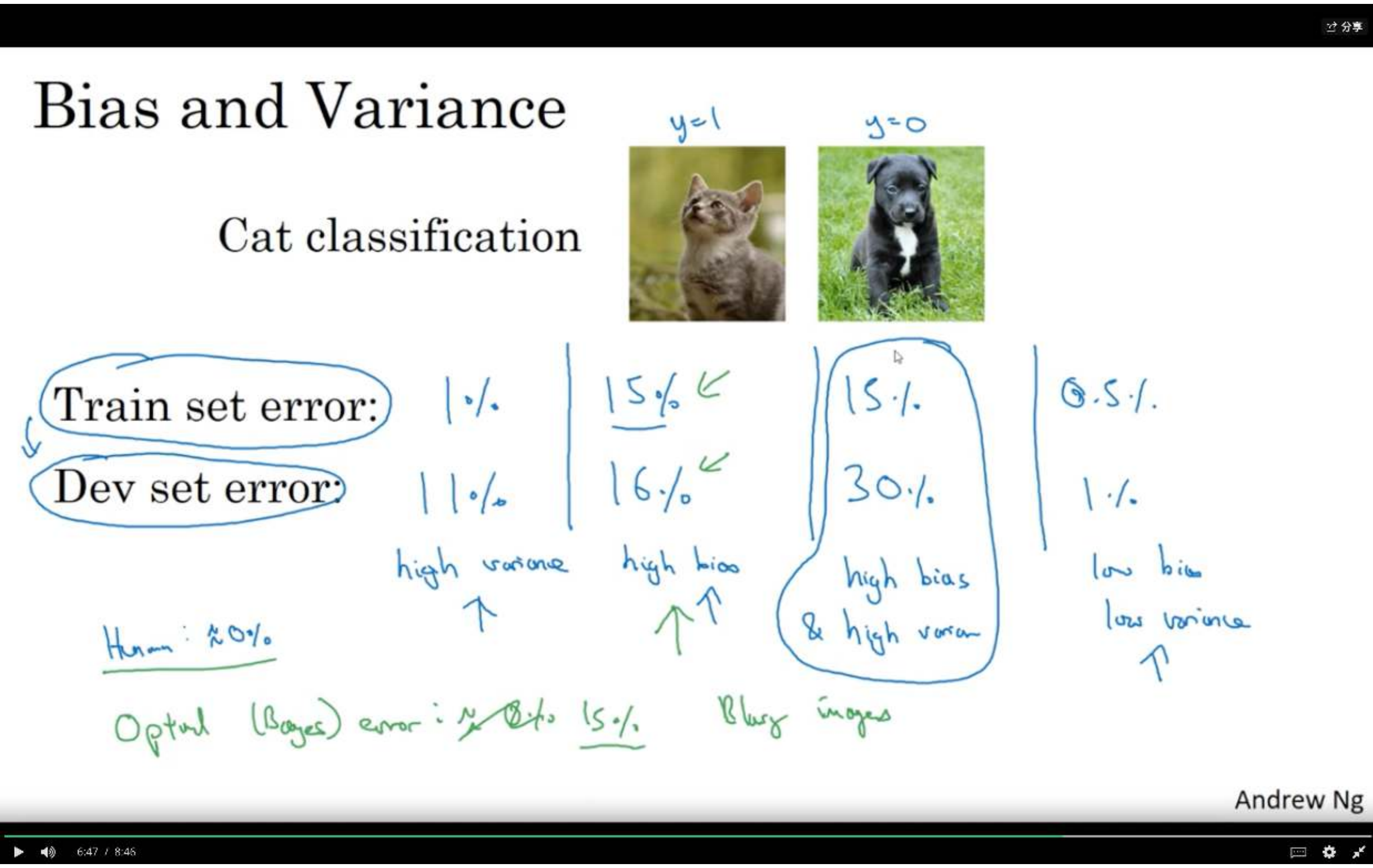
train / dev

train / test

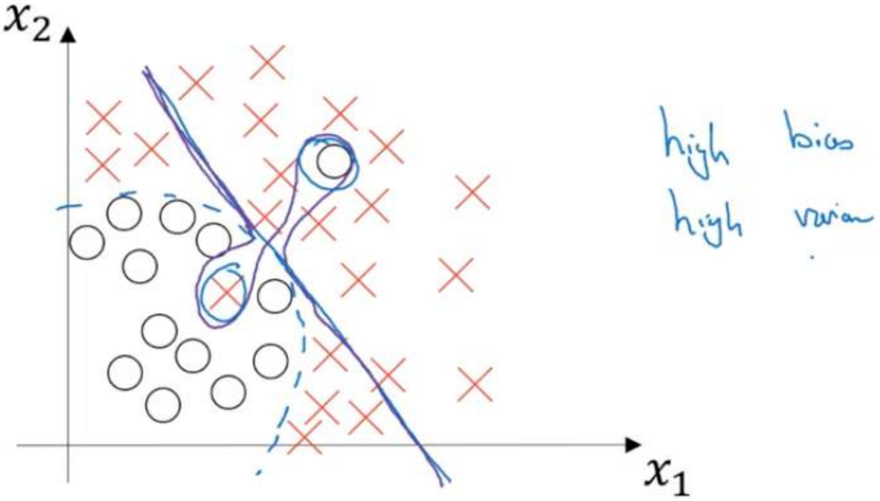
→ train / dev

Not having a test set might be okay. (Only dev set.)

Andrew Ng


Bias and Variance
Cat classification
y=1
y=0
Train set error:
Dev set error:
1%
11%
high variance
15%
16%
high bias
15%
30%
high bias
& high varian
0.5%
1%
low bias
low variance
Human: ≈0%
Optimal (Bayes) error: ~~≈0%~~ 15%
Blurry images
Andrew Ng
6:47 / 8:46

High bias and high variance



in high dimensional inputs that seem less contrived.

Andrew Ng

Basic recipe for machine learning

High bias?
(training data performance)

$\rightarrow$ Y $\rightarrow$ $\boxed{\text{Bigger network}}$

$\rightarrow$ Train longer.

(NN architecture search)

$\downarrow$ N

High variance?
(dev set performance)

$\rightarrow$ Y $\rightarrow$ $\boxed{\text{More data}}$

$\rightarrow$ Regularization

(NN architecture search)

$\downarrow$ N

Done

"Bias variance tradeoff"

Bias $\downarrow$ $\uparrow$ $\downarrow$ variance $\downarrow$ $\downarrow$ $\uparrow$

Logistic regression

$w \in \mathbb{R}^{n_x}, \quad b \in \mathbb{R}$

λ = regularization parameter

lambda lambd

$$\min_{w,b} J(w,b)$$

$$J(w,b) = \underbrace{\frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})}_{} + \frac{\lambda}{2m}\underbrace{\|w\|_2^2}_{} \quad \cancel{+ \frac{\lambda}{2m} b^2}$$

omit

L_2 regularization $\quad \|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w \longleftarrow$

L_1 regularization $\quad \frac{\lambda}{2m}\sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m}\|w\|_1$

w will be sparse

Andrew Ng

4:30 / 9:42

Neural network

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

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} (w_{ij}^{[l]})^2 \qquad w: (n^{[l]}, n^{[l-1]}).$$

"Frobenius norm" $\qquad \|\cdot\|_2^2 \qquad \|\cdot\|_F^2$

$$dw^{[l]} = \boxed{(\text{from backprop}) + \frac{\lambda}{m} w^{[l]}} \qquad \frac{\partial J}{\partial w^{[l]}} = dw^{[l]}$$

$$\rightarrow w^{[l]} := w^{[l]} - \alpha \, dw^{[l]}$$

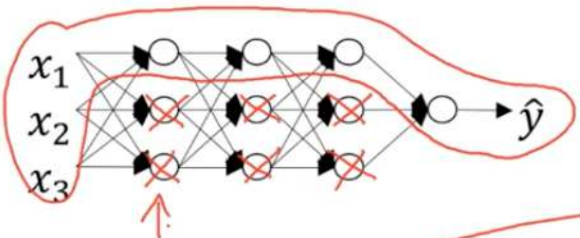
"Weight decay"

$$w^{[l]} := w^{[l]} - \alpha \left[(\text{from backprop}) + \frac{\lambda}{m} w^{[l]} \right]$$

$$\boxed{\left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]}} = \boxed{w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]}} - \alpha (\text{from backprop})$$

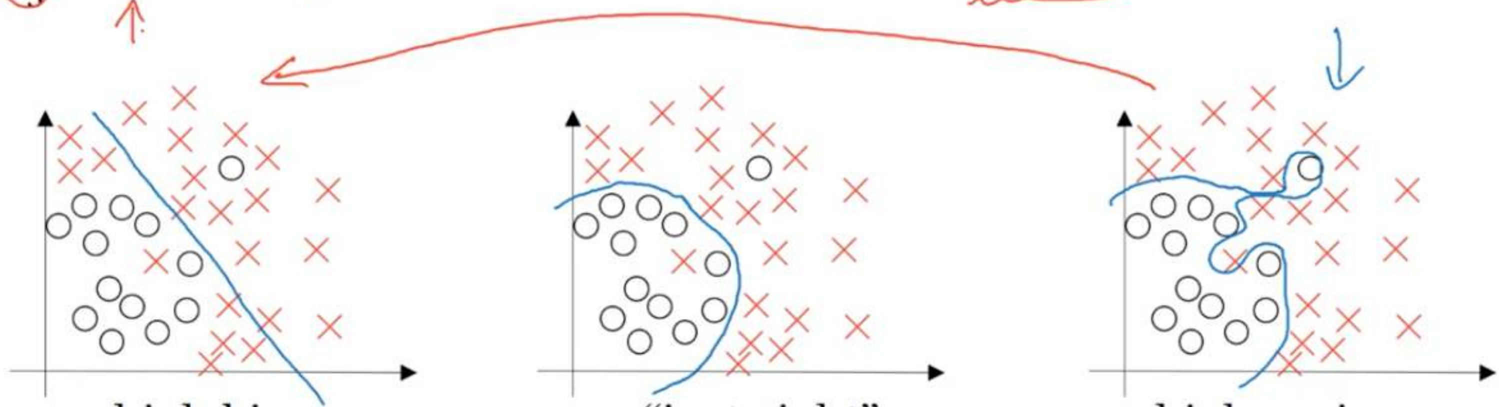
Andrew Ng

How does regularization prevent overfitting?



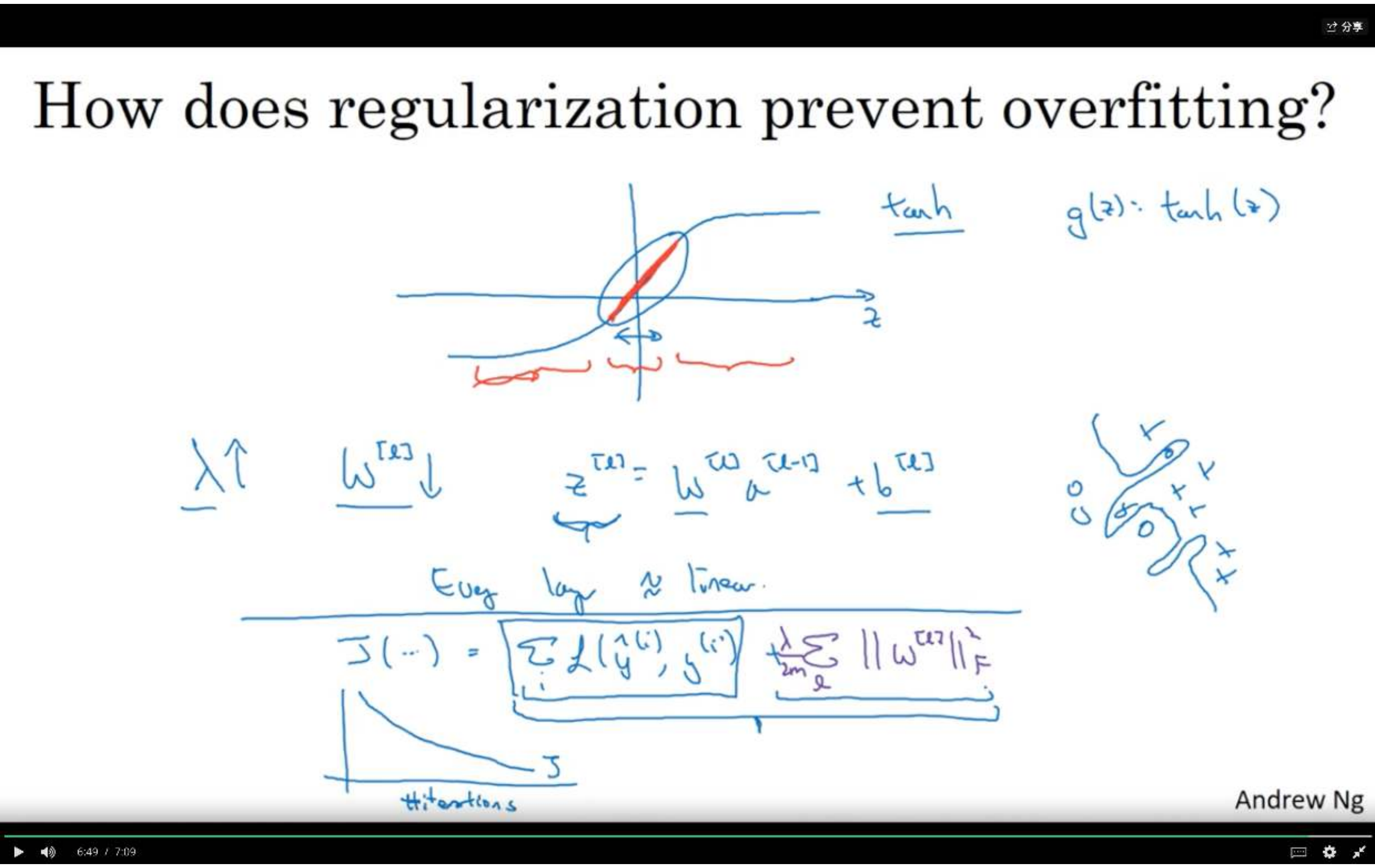
$$J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \underbrace{\|w^{[l]}\|_F^2}$$

$$\underbrace{w^{[l]} \approx 0}$$

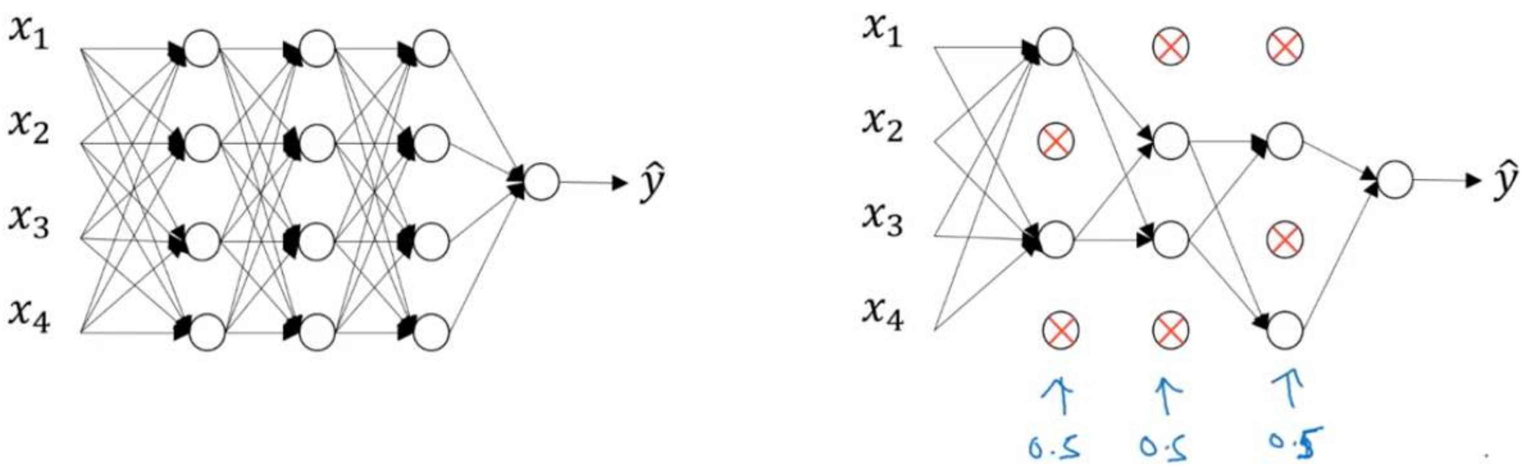


high bias | "just right" | high variance

Andrew Ng


分享
How does regularization prevent overfitting?
tanh
g(z) = tanh(z)
z
λ↑
W[l] ↓
z[l] = W[l] a[l-1] + b[l]
Every layer ≈ linear.
J(...) = Σ L(ŷ(i), y(i)) + λ/2m Σ ||W[l]||²F
J
#iterations
Andrew Ng
6:49 / 7:09

Dropout regularization



Andrew Ng

1:37 / 9:25

Implementing dropout ("Inverted dropout")

Illustrate with layer $l=3$. keep-prob = 0.8 0.2

→ $\boxed{d3}$ = np.random.rand(a3.shape[0], a3.shape[1]) < keep-prob

a3 = np.multiply(a3, d3) # a3 *= d3.

→ $\boxed{\text{a3 /= } \cancel{0.8} \text{ keep-prob}}$ ←

50 units. ⤳ 10 units shut off

$$z^{[4]} = W^{[4]} \cdot a^{[3]} + b^{[4]}$$

reduced by 20%. Test

/= 0.8

Andrew Ng

Making predictions at test time

$a^{[0]} = X$

No drop out.

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = \dots$

$\downarrow$

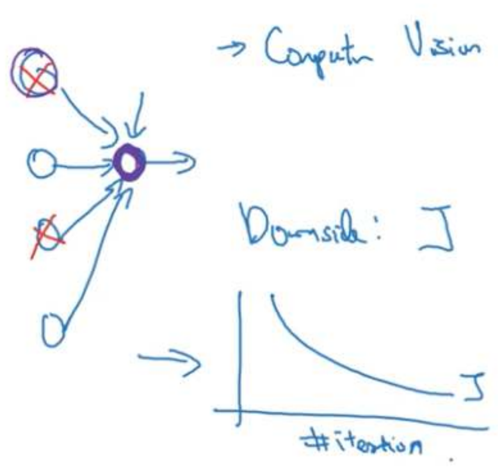
$\hat{y}$

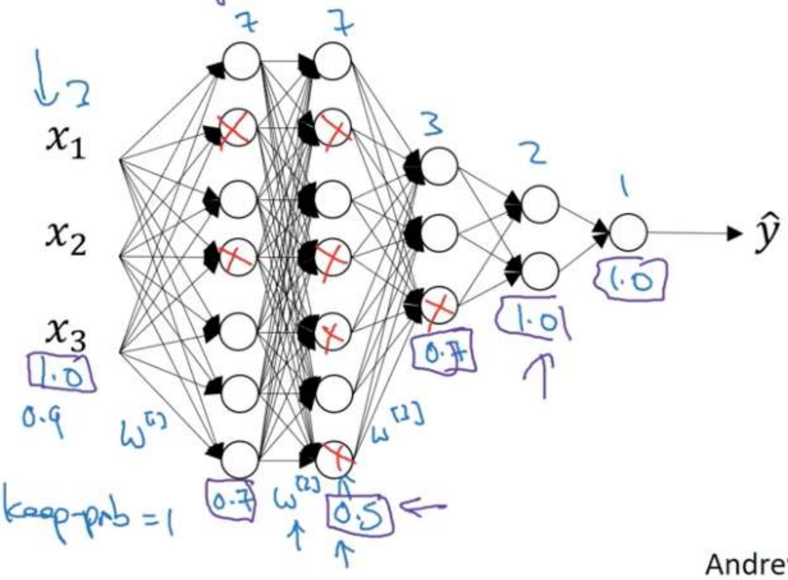
/= keep-prob

Andrew Ng

Why does drop-out work?

Intuition: Can't rely on any one feature, so have to spread out weights. ⇝ Shrink weights. L_2



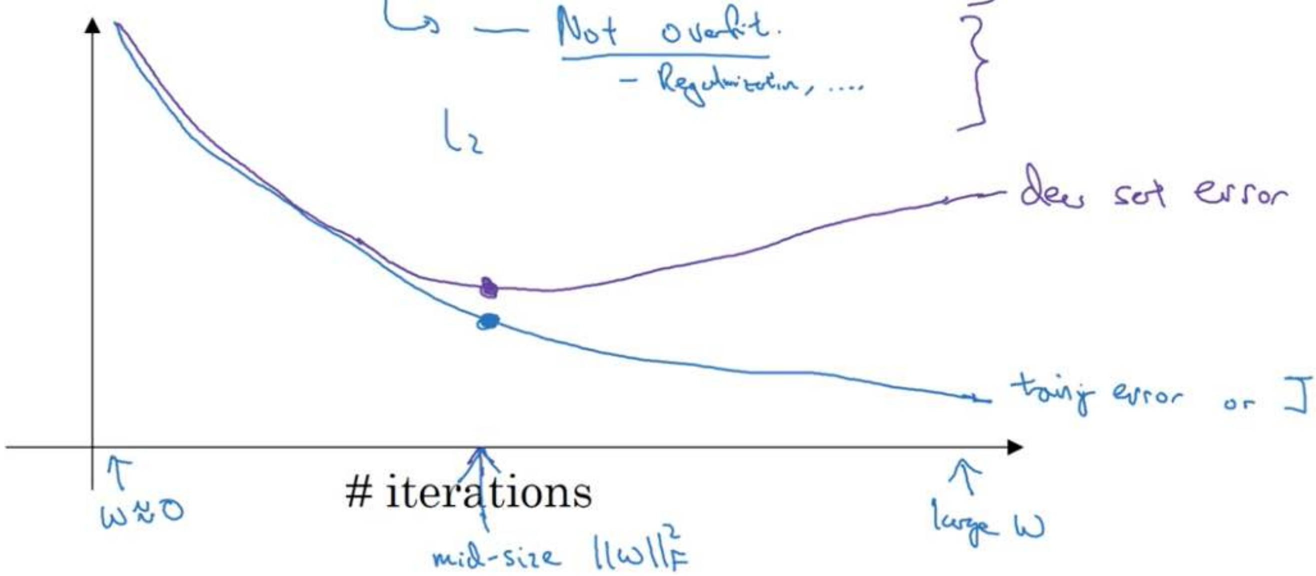


Early stopping

Orthogonalization.

- Optimize cost function J
  - Gradient, ...
- Not overfit.
  - Regularization, ...

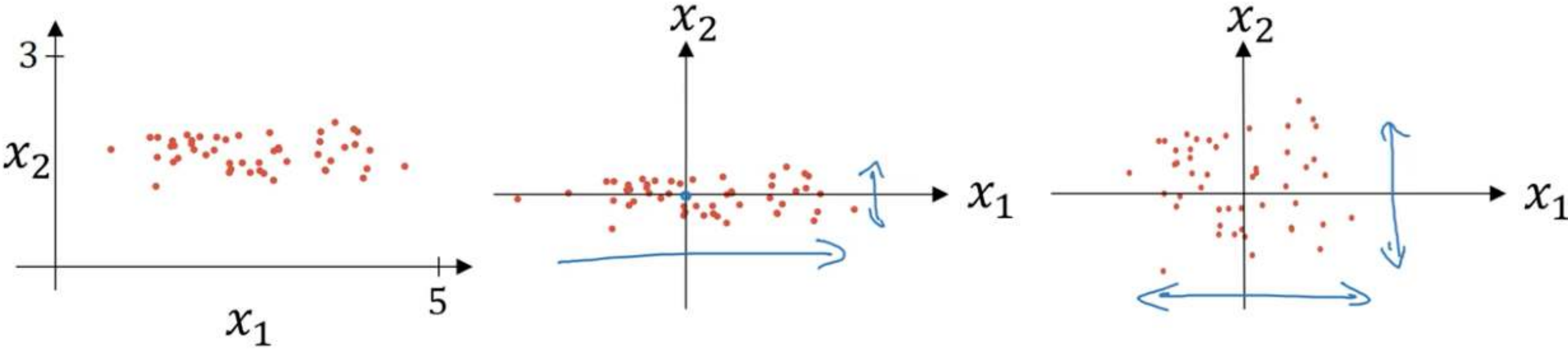
$J(w,b)$



Andrew Ng

Normalizing training sets

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$



Subtract mean:

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

$$\boxed{x := x - \mu}$$

Normalize variance

$$\boxed{\sigma^2} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} ** 2$$

↖ element-wise

$$\boxed{x \mathrel{/}= \sigma^2}$$

Use same μ σ^2 to normalize test set.

Andrew Ng

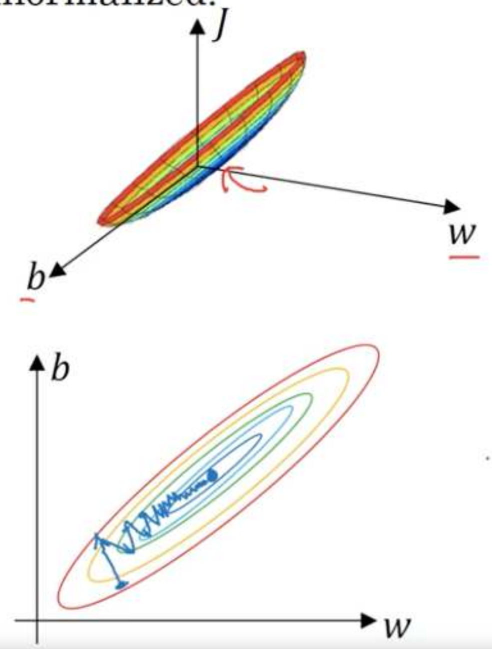
Why normalize inputs?

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

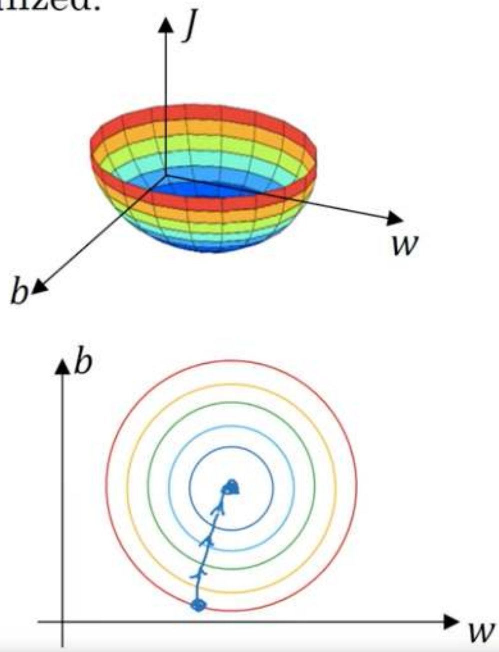
w_1 x_1: 1....1000

w_2 x_2: 0...1

Unnormalized:

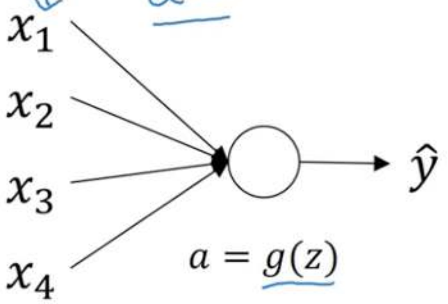


Normalized:



Andrew Ng

Single neuron example



$W^{[l]}$

$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \cancel{+b}$

Large n → Smaller w_i

$Var(w_i) = \cancel{\frac{1}{n}} \; \frac{2}{n}$

$W^{[l]} = $ np.random.randn(shape) $*$ $\boxed{\text{np.sqrt}\left(\frac{2}{n^{[l-1]}}\right)}$

ReLU $\qquad g^{[l]}(z) = \text{ReLU}(z)$

Other variants:

tanh $\boxed{\sqrt{\frac{1}{n^{[l-1]}}}}$

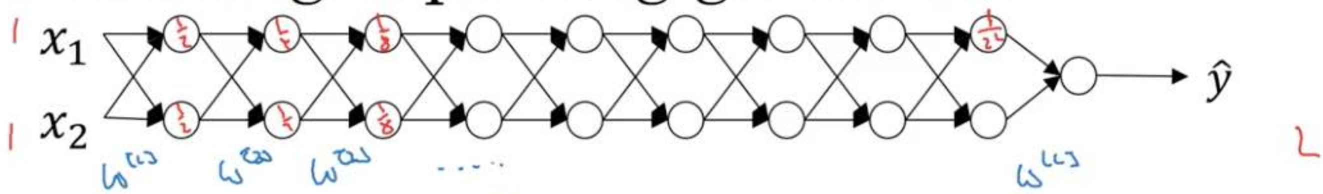
Xavier initialization

$\sqrt{\frac{2}{n^{[l-1]} + n^{[l]}}}$

Andrew Ng

分享

Vanishing/exploding gradients



$g(z) = z$. $\quad b^{[l]} = 0$.

$$\hat{y} = W^{[L]} W^{[L-1]} W^{[L-2]} \cdots \boxed{W^{[3]} \boxed{W^{[2]} W^{[1]} x}}$$

$a^{[3]}$

1.5^L

0.5^L

$z^{[1]} = W^{[1]} x$

$a^{[1]} = g(z^{[1]}) = z^{[1]}$

$a^{[2]} = g(z^{[2]}) = g(W^{[2]} a^{[1]})$

$W^{[l]} > I$

$W^{[l]} < I \quad \begin{bmatrix} 0.9 & \\ & 0.9 \end{bmatrix}$

$$W^{[l]} = \begin{bmatrix} \cancel{1.5}\ 0.5 & 0 \\ 0 & \cancel{1.5}\ 0.5 \end{bmatrix}$$

$$\hat{y} = W^{[L]} \begin{bmatrix} \cancel{1.5}\ 0.5 & 0 \\ 0 & \cancel{1.5}\ 0.5 \end{bmatrix}^{L-1} x$$

$1.5^{L-1} x$

$0.5^{L-1} x$

Andrew Ng

5:05 / 6:07

Gradient checking (Grad check)

$J(\theta) = J(\theta_1, \theta_2, \theta_3, \ldots)$

for each i:

$$\rightarrow \underline{d\theta_{approx}[i]} = \frac{J(\theta_1, \theta_2, \ldots, \theta_i + \varepsilon, \ldots) - J(\theta_1, \theta_2, \ldots, \theta_i - \varepsilon, \ldots)}{2\varepsilon}$$

$$\approx \underline{d\theta[i]} = \frac{\partial J}{\partial \theta_i}$$

$$d\theta_{approx} \overset{?}{\approx} d\theta$$

Check

$$\frac{\| d\theta_{approx} - d\theta \|_2}{\| d\theta_{approx} \|_2 + \| d\theta \|_2} \approx 10^{-7} \text{ - great!}$$

10^{-5}

$\rightarrow 10^{-3}$ - worry.

$\underline{\varepsilon = 10^{-7}}$

Andrew Ng

5:47 / 6:34