Train/Dev/Test set

Data: | training set (train) | development set (dev) | test set |

Previous era → 60/20/20% ~ # data ~100,000 → 60,000/20,000/20,000

Big data → 98/1/1% ~ # data ~1,000,000 → 980,000 /10,000/10,000

Basic recipe ~ bias/variance tradeoff

High bias (training data problem) —YES→ bigger network < ↑# hidden layers, ↑# hidden units; train longer; NN architecture search*

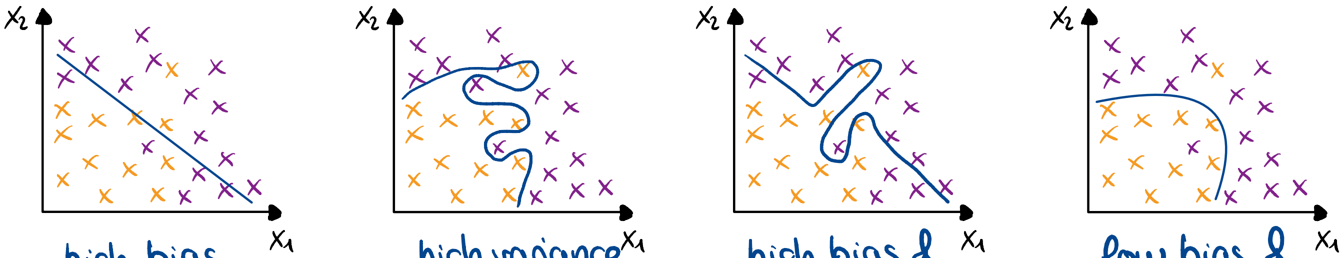
↓ NO

High variance (dev set problem) —YES→ more data; regularization; NN architecture search*

↓ NO

Done

*Not allways work



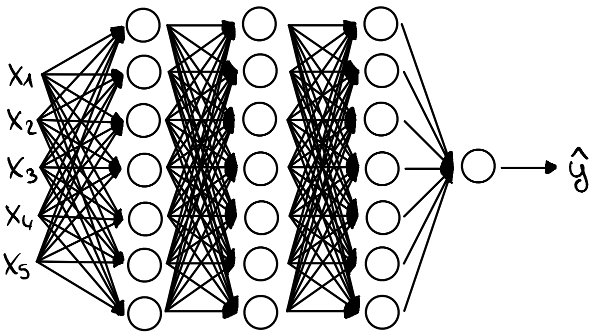
high bias | high variance | high bias & high variance | low bias & low variance

Train set error:	1%	15%	15%	0.5%
Dev set error:	11%	16%	30%	1%
	high bias	high variance	high bias & high variance	low bias & low variance

Deep L-layer Neural Network

4-layer Neural Network

layer: 0 1 2 3 4



elements: 5 7 7 7 1

layer 0 ~ input layer
layer 1-3 ~ hidden layer
layer 4 ~ output layer

shallow NN ~ 1 hidden layer
deep NN ~ > 1 hidden layer

$L = 4$ ~ # layers
$n^{[l]}$ ~ # nodes/elements in layer l $n^{[0]} = 5$ $n^{[1]} = n^{[2]} = n^{[3]} = 7$ $n^{[4]} = 1$
$a^{[l]}$ ~ activations in layer l $a^{[0]} = X$ $a^{[L]} = a^{[4]} = y$

Parameters

$W^{[l]} \in \mathbb{R}^{(n^{[l]}, n^{[l-1]})}$ → initialize to: random * 0.01 (break symmetry (no zeros!!)) (lower value → faster convergence)
$b^{[l]} \in \mathbb{R}^{(n^{[l]}, 1)}$ → initialize to: zeros

Hyperparameters

activation function
α ~ learning rate
β ~ optimization param. (0.9)
$\beta_1, \beta_2, \varepsilon$ ~ Adam optimization params (0.9, 0.999, 10^{-8})

\# iterations
\# layers
\# hidden units
learning rate decay
mini-batch size

Hyperparameter tuning - increase convergence

- Most important
- Second in importance
- Third in importance
- Never tuned

Neural Networks method

Repeat until convergence

1. Forward propagation

$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ ← broadcasting → $(m,n) \; \pm * / \; (m,1)$ or $(1,n)$ $= (m,n)$

$A^{[l]} = g^{[l]}(Z^{[l]})$

2. When $A^{[L]} \sim \hat{y}$ obtained:

$dZ^{[L]} = L(A^{[L]}, y)$ ~ RMSE, MAE ...

$dW^{[L]} = 1/m \; dZ^{[L]} A^{[L-1]T}$

$db^{[L]} = 1/m \sum dZ^{[L]}$ ← columns

3. Backward propagation ~ chain rule

$dZ^{[l]} = W^{[l+1]} dZ^{[l+1]} * g^{[l]'}(Z^{[l]})$ (* element-wise)

$dW^{[l]} = 1/m \; dZ^{[l]} A^{[l-1]T}$

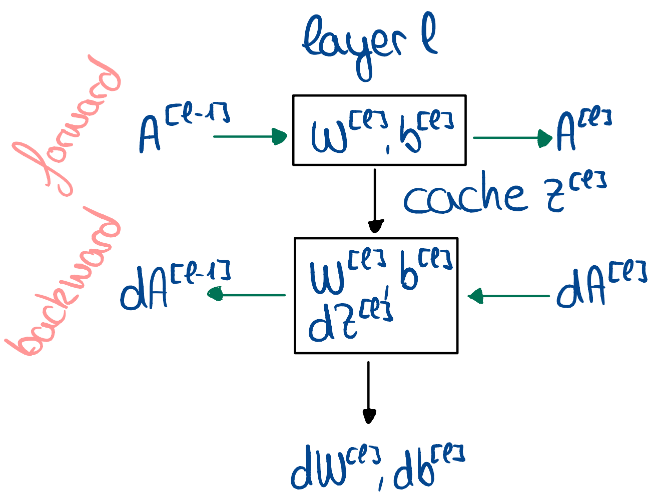
$db^{[l]} = 1/m \sum dZ^{[l]}$

4. Parameters update

$W^{[l]} = W^{[l]} - \alpha \, dW^{[l]}$

$b^{[l]} = b^{[l]} - \alpha \, db^{[l]}$

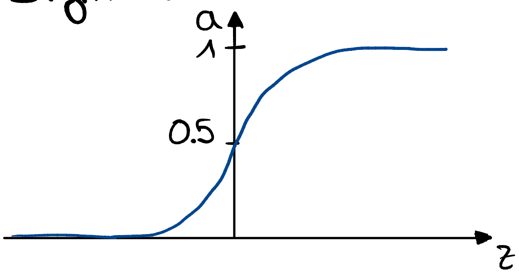
Deep Neural Networks in blocks



Most used activation functions

Can be different for different layers

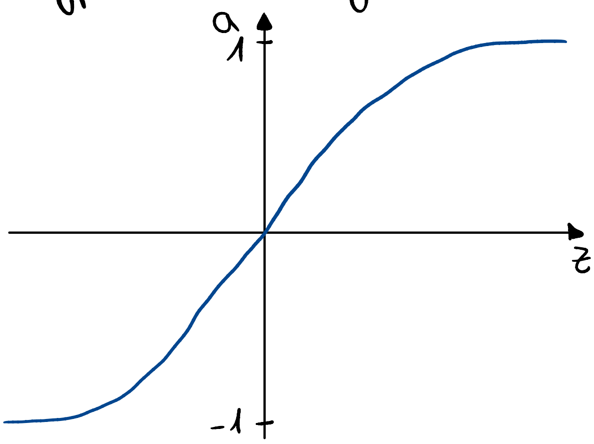
Sigmoid



$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(z) = a(1-a)$$

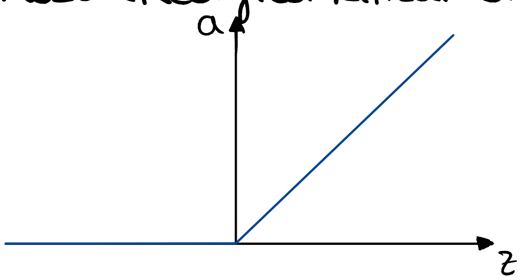
Hyperbolic tangent



$$g(z) = \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

$$g'(z) = 1-a^2$$

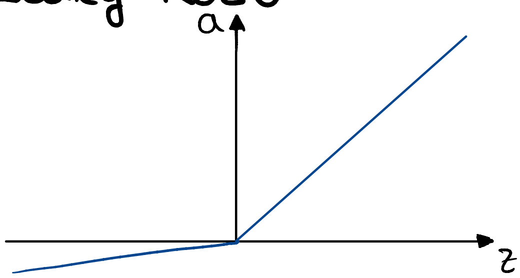
ReLU (Rectified Linear Unit)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 0 & \text{if } z<0 \\ 1 & \text{if } z\geq 0 \end{cases}$$

Leaky ReLU



$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z<0 \\ 1 & \text{if } z\geq 0 \end{cases}$$

Optimization algorithms

Mini-batch ~ divisions of the training set

$$X = [\underbrace{x^{(1)}, x^{(2)}, \dots, x^{(1000)}}_{X^{\{1\}}} \mid \underbrace{x^{(1001)}, \dots, x^{(2000)}}_{X^{\{2\}}} \mid \underbrace{\dots, x^{(m)}}_{X^{\{5000\}}}]$$

$X \in \mathbb{R}^{(n_x, m)}$ — $n^{[0]}$ ~ # variables; # training examples

$$Y = [\underbrace{y^{(1)}, y^{(2)}, \dots, y^{(1000)}}_{Y^{\{1\}}} \mid \underbrace{y^{(1001)}, \dots, y^{(2000)}}_{Y^{\{2\}}} \mid \underbrace{\dots, y^{(m)}}_{Y^{\{5000\}}}]$$

$Y \in \mathbb{R}^{(1, m)}$ — if 1 output

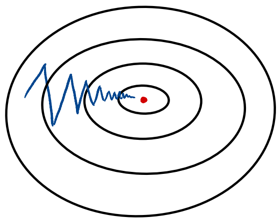
If m = 5,000,000 – mini-baches of 1,000 each

Typical mini-batch sizes: 64, 128, 256, 512 ≡ 2^n

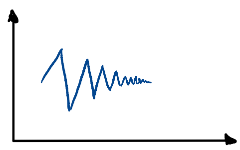
If mini-batch size = m : Batch gradient descent
If mini-batch size = 1 : Stochastic gradient descent
} In practice somewhere between both

Adam optimization

Convergence:



⇒



⇒ We want: ↕ slower, ↔ faster ⇒ Adam

Learning rate decay (most used method)

$$\alpha = \frac{1}{1 + decayrate * \#epochs} \alpha_0$$

1 epoch = 1 pass through data