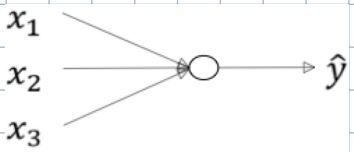
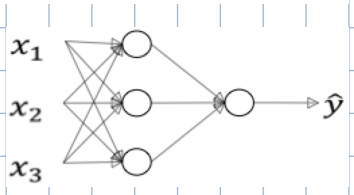
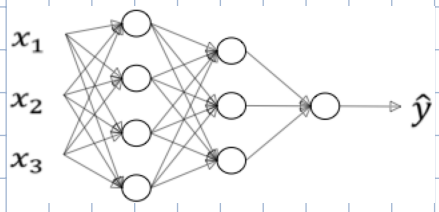
What is a deep neural network?



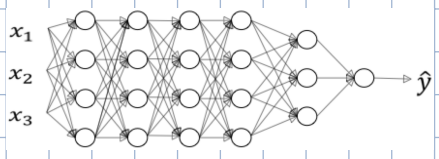
Logistic Regression



1 Hidden Layer



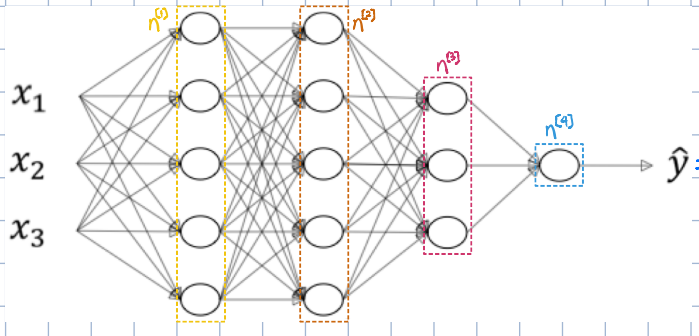
2 Hidden Layers



5 Hidden Layers

Deep Neural Networks Notation:

It's a 4-layer NN with 3 hidden layers.



$L = 4$, number of layers

$n^{[l]}$: number of nodes in layer l

$n^{[1]} = 5$, $n^{[2]} = 5$

$n^{[3]} = 3$, $n^{[4]} = n^{[L]} = 1$

$n^{[0]} = n_x = 3$, the dimension of the input layer

$w^{[l]}$, $b^{[l]}$: weights for $z^{[l]}$

$z^{[l]} = w^{[l]} X + b^{[l]}$

$g^{[l]}$: the activation function for the layer l

$a^{[l]} = g^{[l]}(z^{[l]})$, the activation for the layer l

The input feature $X = a^{[0]}$

Standard notations for Deep Learning

This document has the purpose of discussing a new standard for deep learning mathematical notations.

1 Neural Networks Notations.

General comments:

· superscript (i) will denote the i^{th} training example while superscript [l] will denote the l^{th} layer

Sizes:

·m : number of examples in the dataset

·n_x : input size

·n_y : output size (or number of classes)

·$n_h^{[l]}$: number of hidden units of the l^{th} layer

In a for loop, it is possible to denote $n_x = n_h^{[0]}$ and $n_y = n_h^{[\text{number of layers } +1]}$.

·L : number of layers in the network.

Objects:

·$X \in \mathbb{R}^{n_x \times m}$ is the input matrix

·$x^{(i)} \in \mathbb{R}^{n_x}$ is the i^{th}example represented as a column vector

·$Y \in \mathbb{R}^{n_y \times m}$ is the label matrix

·$y^{(i)} \in \mathbb{R}^{n_y}$ is the output label for the i^{th} example

·$W^{[l]} \in \mathbb{R}^{\text{number of units in next layer} \times \text{number of units in the previous layer}}$ is the weight matrix,superscript [l] indicates the layer

·$b^{[l]} \in \mathbb{R}^{\text{number of units in next layer}}$ is the bias vector in the l^{th} layer

·$\hat{y} \in \mathbb{R}^{n_y}$ is the predicted output vector. It can also be denoted $a^{[L]}$ where L is the number of layers in the network.

Common forward propagation equation examples:

$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1)$ where $g^{[l]}$ denotes the l^{th} layer activation function

$\hat{y}^{(i)} = softmax(W_h h + b_2)$

· General Activation Formula: $a_j^{[l]} = g^{[l]}(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}) = g^{[l]}(z_j^{[l]})$

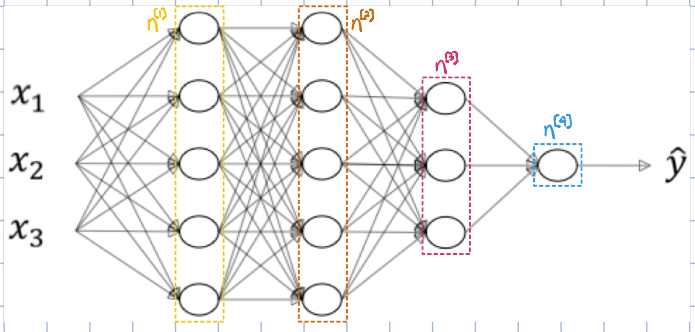
· $J(x, W, b, y)$ or $J(\hat{y}, y)$ denote the cost function.

Examples of cost function:

· $J_{CE}(\hat{y}, y) = -\sum_{i=0}^{m} y^{(i)} \log \hat{y}^{(i)}$

· $J_1(\hat{y}, y) = \sum_{i=0}^{m} \mid y^{(i)} - \hat{y}^{(i)} \mid$

Forward Propagation in Neural Networks



General rule for Forward Propagation:

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

For one training sample x:

$$z^{[1]} = w^{[1]} x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$
$$\vdots$$
$$z^{[4]} = w^{[4]} a^{[3]} + b^{[4]}$$
$$a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$$

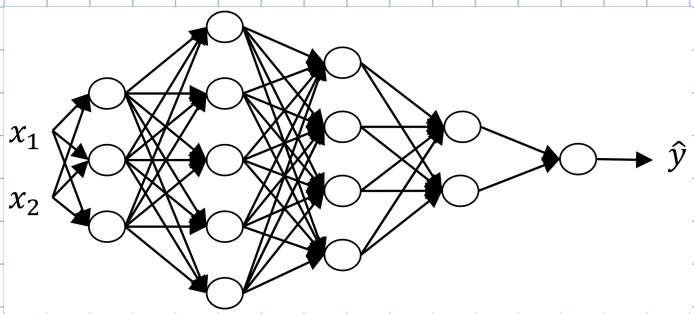
A for-loop is used here to calculate forward propagation. It seems it's impossible to vectorize the calculation here.

For vectorized all training sample X:

$$Z^{[1]} = W^{[1]} \underbrace{X}_{A^{[0]}} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
$$\vdots$$
$$Z^{[4]} = W^{[4]} A^{[3]} + b^{[4]}$$
$$A^{[4]} = g^{[4]}(Z^{[4]}) = \hat{y}$$

Getting the Matrix Dimension Right

When learning and reading materials of machine learning, especially neural networks, please write down the dimensions of matrices in every layer and every step. It will help me to form a deeper understanding of the algorithms and visualize the process from the input to the output.



$n^{[0]} = 2, n^{[1]} = 3, n^{[2]} = 5, n^{[3]} = 4, n^{[4]} = 2, n^{[5]} = 1$

$w^{[l]} : (n^{[l]}, n^{[l-1]})$, $dw^{[l]} : (n^{[l]}, n^{[l-1]})$
$b^{[l]} : (n^{[l]}, 1)$, $db^{[l]} : (n^{[l]}, 1)$

For 1 sample $x : (n^{[0]}, 1)$. $z^{[l]} = w^{[l]} x + b^{[l]}$

$z^{[1]} : (n^{[1]}, 1)$, $z^{[2]} : (n^{[2]}, 1)$ ... $z^{[5]} : (n^{[5]}, 1)$
$w^{[1]} : (n^{[1]}, n^{[0]})$, $w^{[2]} : (n^{[2]}, n^{[1]})$ ... $w^{[5]} : (n^{[5]}, n^{[4]})$
$b^{[1]} : (n^{[1]}, 1)$, $b^{[2]} : (n^{[2]}, 1)$ ... $b^{[5]} : (n^{[5]}, 1)$
$a^{[1]} : (n^{[1]}, 1)$, $a^{[2]} : (n^{[2]}, 1)$ ... $a^{[5]} : (n^{[5]}, 1)$

For vectorized m samples $X : (n^{[0]}, m)$

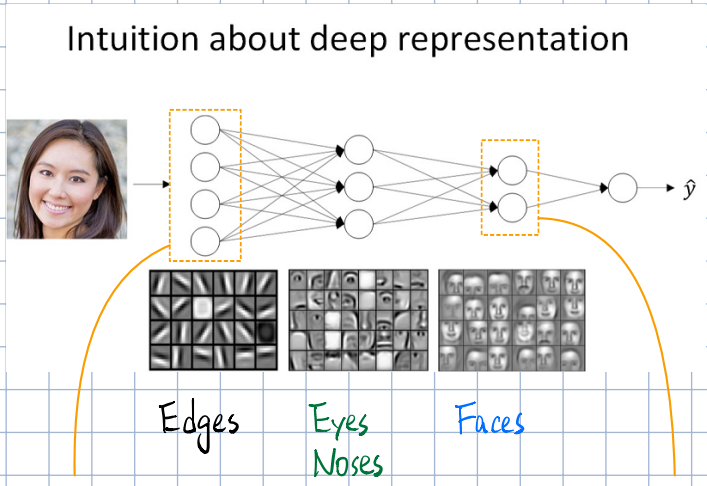
$Z^{[1]} : (n^{[1]}, m)$, $Z^{[2]} : (n^{[2]}, m)$ ... $Z^{[5]} : (n^{[5]}, m)$
$w^{[1]} : (n^{[1]}, n^{[0]})$, $w^{[2]} : (n^{[2]}, n^{[1]})$ ... $w^{[5]} : (n^{[5]}, n^{[4]})$
$b^{[1]} : (n^{[1]}, 1)$, $b^{[2]} : (n^{[2]}, 1)$ ... $b^{[5]} : (n^{[5]}, 1)$
$a^{[1]} : (n^{[1]}, m)$, $a^{[2]} : (n^{[2]}, m)$ ... $a^{[5]} : (n^{[5]}, m)$

$Z^{[l]}, A^{[l]} : (n^{[l]}, m)$. When $l = 0$, $A^{[0]} = X : (n^{[0]}, m)$

$dZ^{[l]}, dA^{[l]} : (n^{[l]}, m)$.

Intuition about Deep Neural Networks Representation

A big neural network needs to be Deep. Big is not enough.



A deep neural network first finds simple things like edges. And then it composes things together to detect more complex things like noses, eyes, etc. And it further composes those together to compute even more complex things like faces.

Another example:

Audio: Low Level Waveform Features → Phonemes → Words

Another intuition about why Deep Neural Networks work well:

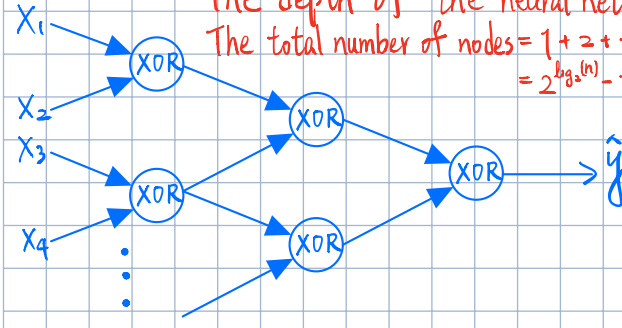
Circuit Theory:

Informally: There are functions you can compute with a "small" L-layer deep neural network that a shallower networks require exponentially more hidden units to compute.

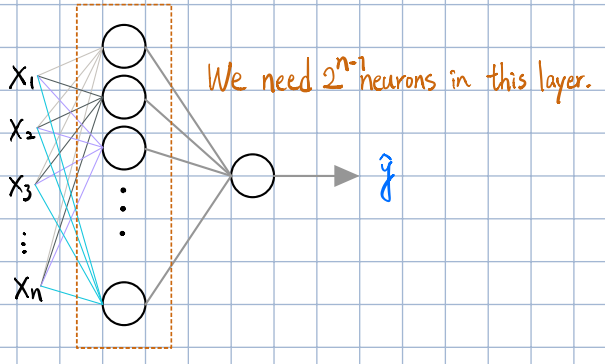
Example:

We have n-dim features, $x_1, \dots, x_n$.

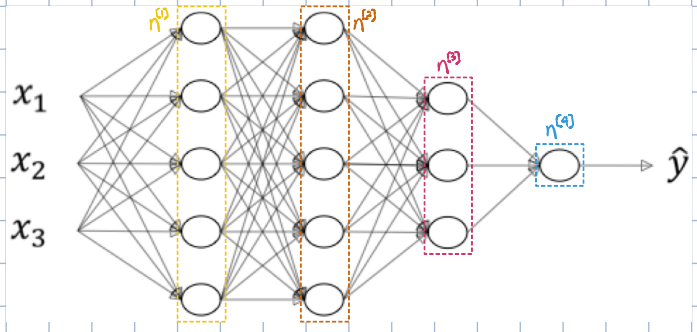
We want to find $Y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \cdots \text{ XOR } x_n$



Now, if only One hidden layer is used.



Building Blocks of a Deep Neural Network



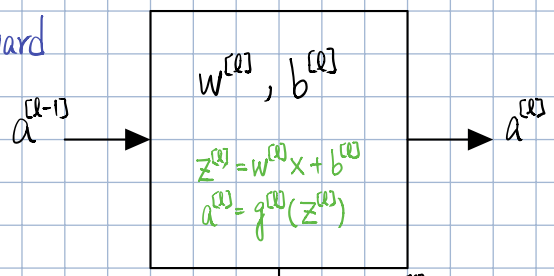
Parameters: $W^{[l]}$ and $b^{[l]}$

Hyperparameters: learning rate α, num. of iterations, num. of hidden layers, num. of hidden units, choice of activation functions, momentum, mini-batch size, regularization, etc.

Applied deep learning is a very empirical process. AutoML!

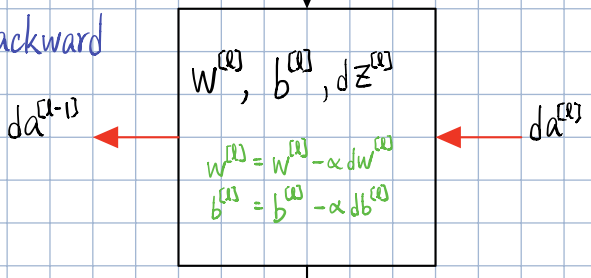
In the layer l:

Forward

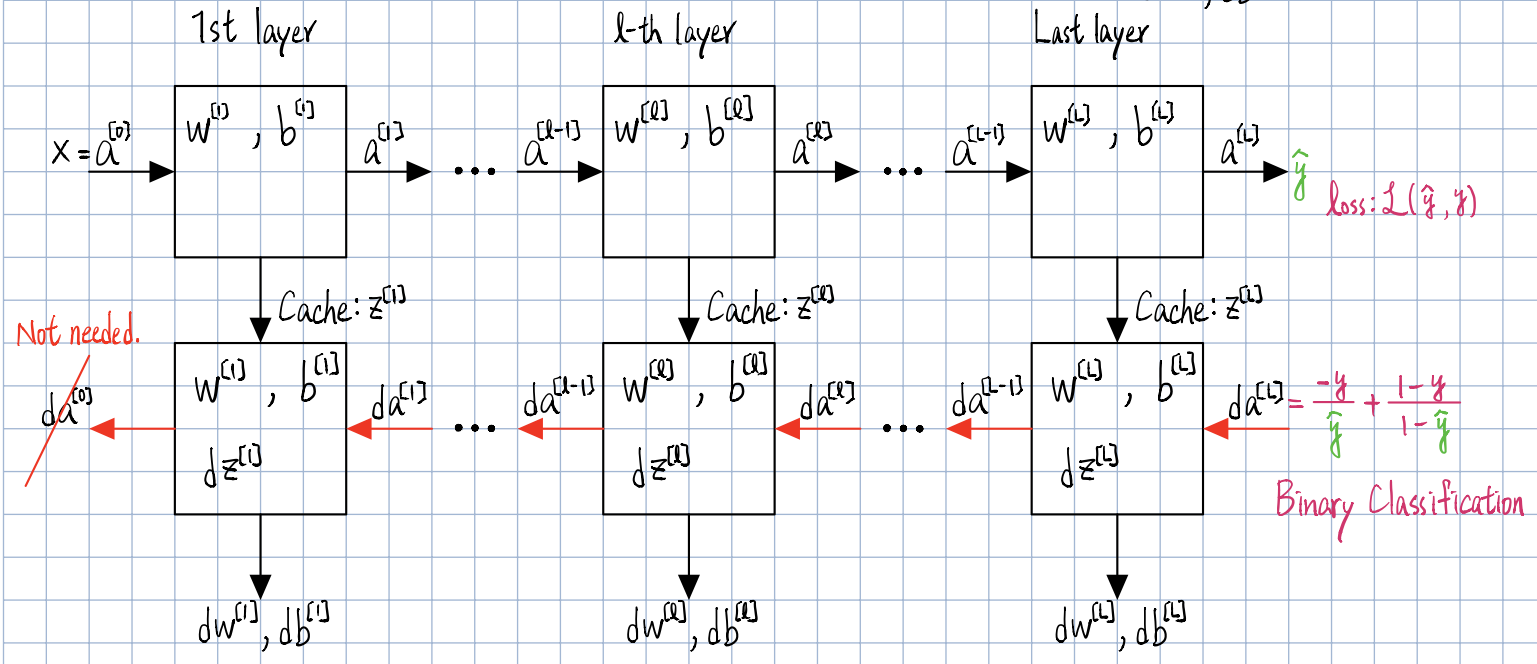


Cache: $z^{[l]}$

Backward



$dw^{[l]}, db^{[l]}$



Forward Propagation for m samples

$$X = A^{[0]}$$

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]})$$

$$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

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

$$\vdots$$

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

Backward Propagation for m samples

$*$: element-wise multi.

$$dZ^{[L]} = A^{[L]} - Y$$

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

$$db^{[L]} = \frac{1}{m} \text{np.sum}(dZ^{[L]}, \text{axis}=1, \text{keepdims}=\text{True})$$

$$dZ^{[L-1]} = W^{[L]T} dZ^{[L]} * g'^{[L-1]}(Z^{[L-1]})$$

$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]}) = W^{[l+1]T} dZ^{[l+1]} * g^{[l]\prime}(Z^{[l]})$$

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

$$db^{[l]} = \frac{1}{m} \text{sum}(dZ^{[l]}, \text{axis}=1)$$

$$dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$$

$$\vdots$$

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g'^{[1]}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]T}$$

$$db^{[1]} = \frac{1}{m} \text{np.sum}(dZ^{[1]}, \text{axis}=1, \text{keepdims}=\text{True})$$