# CMPE 258, Deep Learning

## Softmax, Multiclass classification

Feb 15, 2018

DMH 149A

**Taehee Jeong**

**Ph.D., Data Scientist**

SJSU SAN JOSÉ STATE UNIVERSITY

# Assignment_1

## It was graded.

## Please turn in a hard copy of your submission for Assignment_1 today.

1 (10pts). Linear regression with one variable

2 (30pts). Linear regression with two variables

3 (60pts). Linear regression with multiple variables

- from scratch (for loop, gradient descent)
- using matrix (gradient descent)
- using normal equation
- using scikit-learn linear regression model
- using TensorFlow gradient descent method

# Assignment_2: Any question?

Due is extended to 2/18. Please bring a hard copy until 2/20.

1 (30pts). Polynomial regression / overfitting / regularization
2 (30pts). Polynomial regression with train/validation/test
3 (40pts). Regularization with Tensorflow

- using L2 penalty (Ridge)
- using L1 penalty (Lasso)
- using matrix (gradient descent)
- using scikit-learn linear regression model
- using TensorFlow gradient descent method

# Logistic Regression

$$\hat{y} = \sigma(W^T x + b)$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log\left(\widehat{y^i}\right) + \left(1 - y^i\right)\log(1 - \widehat{y^i})]$$

<Machine Learning, Emily Fox & Carlos Guestrin>

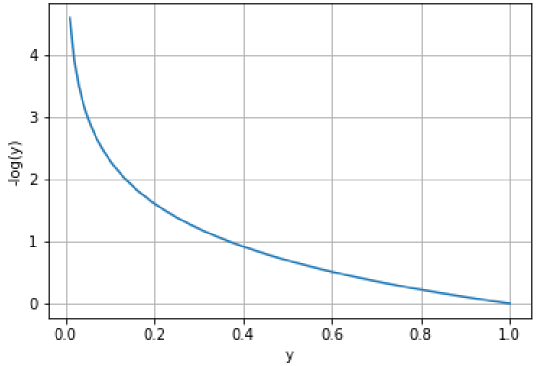
# Logistic Regression

## Loss function

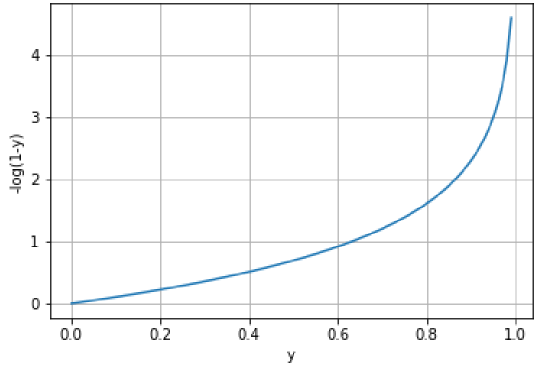
Given x, want $\widehat{y^i} \approx y^i$

$$L = -[y^i \log(\widehat{y^i}) + (1 - y^i)\log(1 - \widehat{y^i})]$$

If y = 1, $L = -\log(\widehat{y^i})$




If y = 0, $L = -\log(1 - \widehat{y^i})]$



# Logistic Regression

## Regularization

Cost function

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log(\widehat{y^i}) + (1 - y^i)\log(1 - \widehat{y^i})] + \frac{\lambda}{m}\sum_{j=1}^{n} W_j^2$$

gradient

$$\frac{\partial J}{\partial W_0} = -\frac{1}{m}\sum_{i=1}^{m}[(\widehat{y^i} - y^i)x_j^i] \quad \text{for j=0}$$

$$\frac{\partial J}{\partial W_j} = -\frac{1}{m}\sum_{i=1}^{m}[(\widehat{y^i} - y^i)x_j^i] + \frac{2\lambda}{m}W_j \quad \text{for j} \geq 1$$

# Mini-batch Gradient Descent

- Batch Gradient Descent

Computes the gradients based on full training set

Ex) Offline learning

- Stochastic Gradient Descent

Computes just one instance

Ex) Online learning

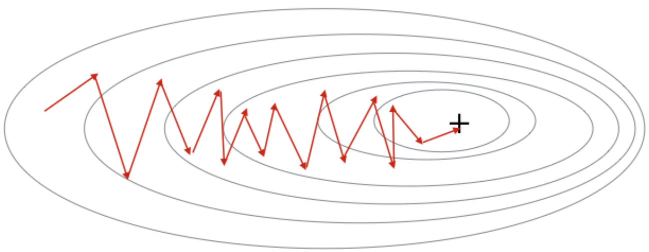
- Mini-batch Gradient Descent

Computes the gradients on small random sets of instances

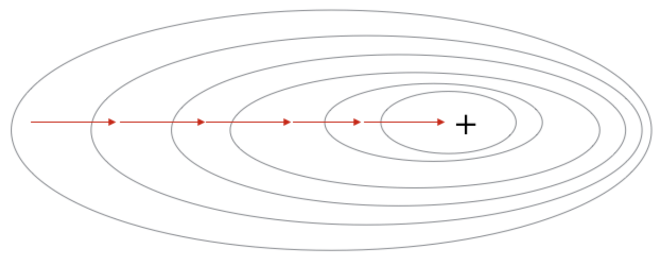
<Hands-On ML, Aurelien Geron>

# Converge

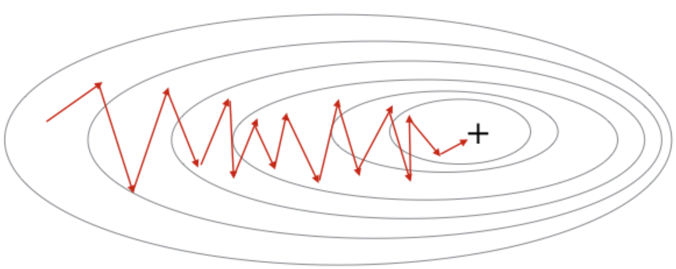
Stochastic Gradient Descent



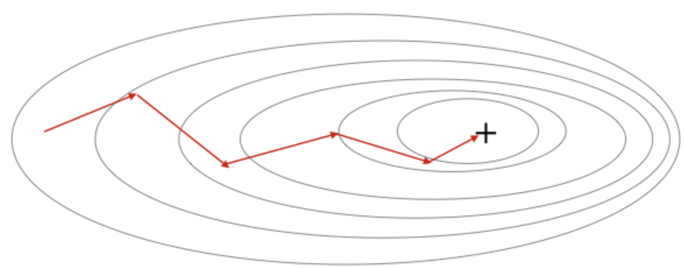
Gradient Descent



Stochastic Gradient Descent



Mini-Batch Gradient Descent



# Stochastic gradient descent

**Batch Gradient descent**

$$\frac{\partial J}{\partial W} = -\frac{1}{m}\sum_{i=1}^{m}[\,(\widehat{y^i} - y^i)x_j^i]$$

(Sum over data points)

**Stochastic Gradient descent**

$$\frac{\partial J}{\partial W} = -(\widehat{y^i} - y^i)x_j^i$$

(Each time, pick different data point)

# Multiclass Classification

Whereas binary classifiers distinguish between two classes, *multiclass classifiers* (also called *multinomial classifiers*) can distinguish between more than two classes.

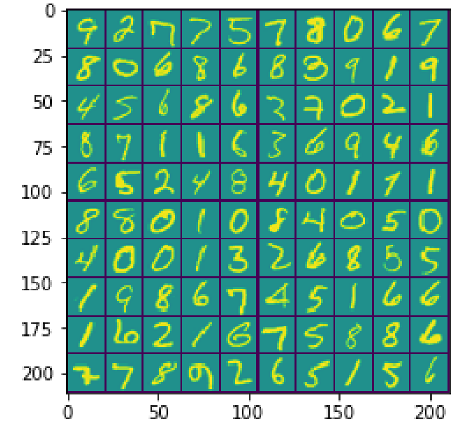
Random Forest Classifiers, naïve Bayes Classifiers handles multiple classes directly.

Support Vector Machine, Logistic classifiers are binary classifiers.

One-versus-all (OvA) strategy

One-versus-one (OvO) strategy

# One-versus-all (OvA)

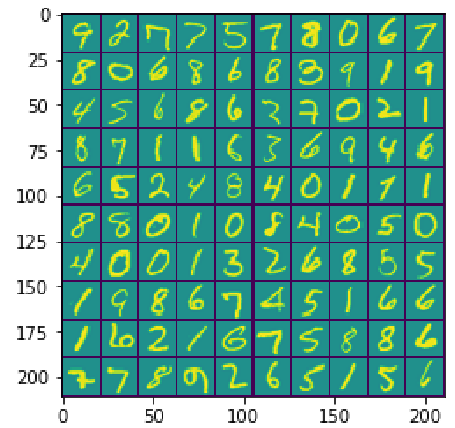



## One-versus-the-rest

For example, one way to create a system that can classify the digit images into 10 classes (from 0 to 9) is to train 10 binary classifiers, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).

Then when you want to classify an image, you get the decision score from each classifier for that image and you select the class whose classifier outputs the highest score.

# One-versus-one (OvO)




Another strategy is to train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on.

If there are $N$ classes, you need to train $N \times (N - 1) / 2$ classifiers.

For the MNIST problem, this means training 45 binary classifiers!

When you want to classify an image, you have to run the image through all 45 classifiers and see which class wins the most duels.

# Softmax classifier

## Multinomial Logistic Regression

The Logistic Regression model can be generalized to support multiple classes directly, without having to train and combine multiple binary classifiers.

When given an instance $\mathbf{x}$, the Softmax classifier first computes a score $s_k(\mathbf{x})$ for each class $k$, then estimates the probability of each class by applying the *softmax function* to the scores.

# Softmax function

Softmax score for class k

$$s_k(x) = W^k \cdot x$$

Softmax function

$$\boxed{p_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K} \exp(s_j(x))}}$$

- *K* is the number of classes.
- **s**(**x**) is a vector containing the scores of each class for the instance **x**.
- $p_k$ is the estimated probability that the instance **x** belongs to class *k* given the scores of each class for that instance.

# Softmax classifier

- Multiclass classifier
- Provides normalized class probabilities

$$p_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K} \exp(s_j(x))}$$

$$\boxed{\hat{y} = \underset{k}{\operatorname{argmax}}\, p_k}$$

It predicts the class with the highest estimated probability.

# Softmax in matrix

- for $x \in \mathbb{R}^{1\times n}$, $softmax(x) = softmax(\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}) = \begin{bmatrix} \frac{e^{x_1}}{\sum_j e^{x_j}} & \frac{e^{x_2}}{\sum_j e^{x_j}} & \dots & \frac{e^{x_n}}{\sum_j e^{x_j}} \end{bmatrix}$
- for a matrix $x \in \mathbb{R}^{m\times n}$, $x_{ij}$ maps to the element in the $i^{th}$ row and $j^{th}$ column of $x$, thus we have:

$$softmax(x) = softmax\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} = \begin{bmatrix} \frac{e^{x_{11}}}{\sum_j e^{x_{1j}}} & \frac{e^{x_{12}}}{\sum_j e^{x_{1j}}} & \frac{e^{x_{13}}}{\sum_j e^{x_{1j}}} & \dots & \frac{e^{x_{1n}}}{\sum_j e^{x_{1j}}} \\ \frac{e^{x_{21}}}{\sum_j e^{x_{2j}}} & \frac{e^{x_{22}}}{\sum_j e^{x_{2j}}} & \frac{e^{x_{23}}}{\sum_j e^{x_{2j}}} & \dots & \frac{e^{x_{2n}}}{\sum_j e^{x_{2j}}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{e^{x_{m1}}}{\sum_j e^{x_{mj}}} & \frac{e^{x_{m2}}}{\sum_j e^{x_{mj}}} & \frac{e^{x_{m3}}}{\sum_j e^{x_{mj}}} & \dots & \frac{e^{x_{mn}}}{\sum_j e^{x_{mj}}} \end{bmatrix} = \begin{pmatrix} softmax(\text{first row of x}) \\ softmax(\text{second row of x}) \\ \dots \\ softmax(\text{last row of x}) \end{pmatrix}$$

<Deep Learning, Andrew Ng>

# Biological Neurons

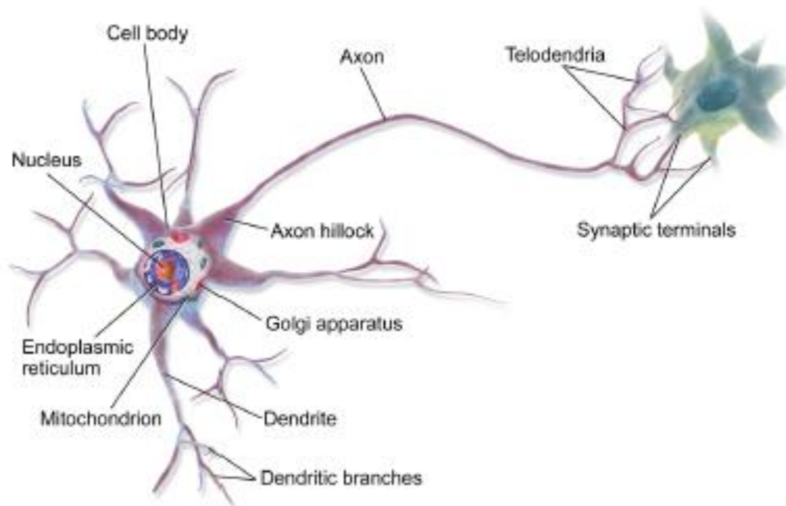
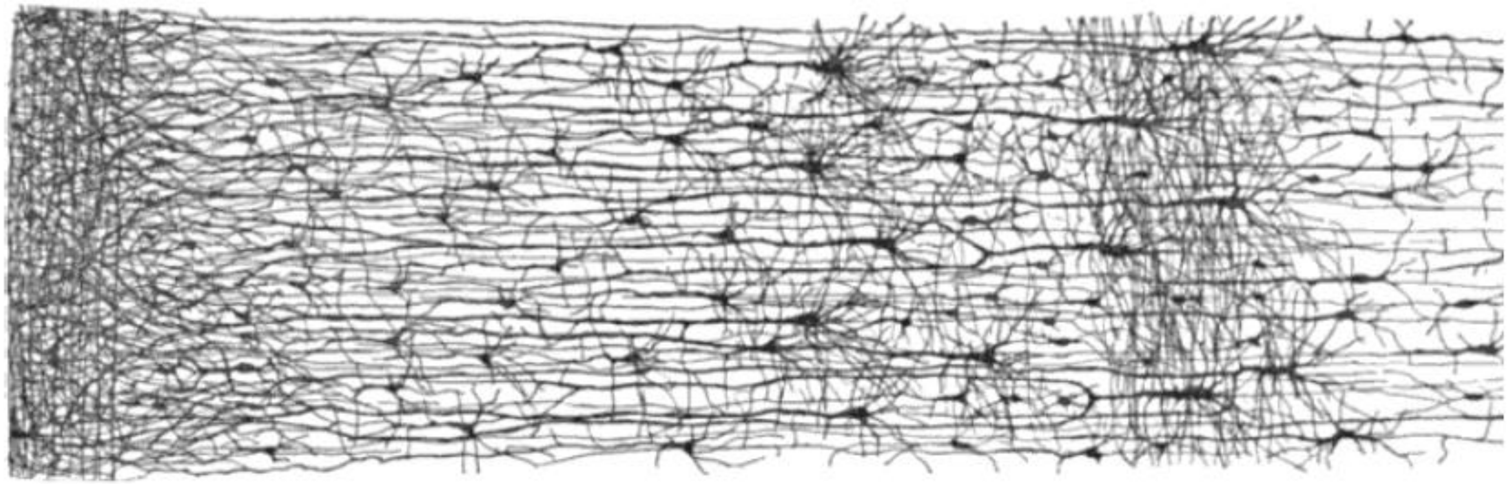



Image by Bruce Blaus (Creative Commons 3.0). Reproduced from *https://en.wikipedia.org/wiki/Neuron*.

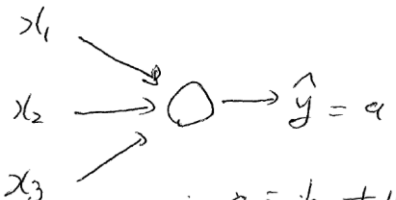
# Multiple layers in a biological neural network (human cortex)



Drawing of a cortical lamination by S. Ramon y Cajal (public domain). Reproduced from *https://en.wikipedia.org/wiki/Cerebral_cortex*.

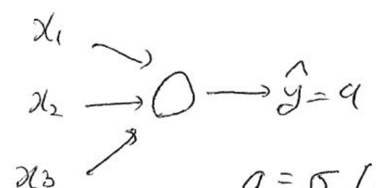
# Regression

## Linear regression




$$a = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$
$$= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

## Logistic regression




$$a = \sigma(w^T x + b)$$
$$= \frac{1}{1 + \exp(-w^T x - b)}$$

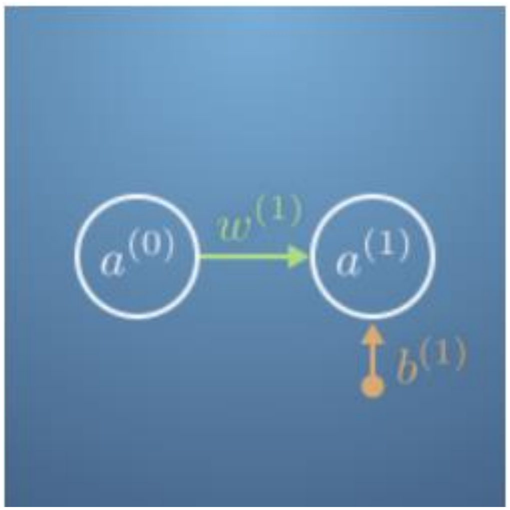
$\sigma(x)$ ; sigmoid (logistic) activation function



SJSU SAN JOSÉ STATE UNIVERSITY

# Artificial neurons

## Two neurons




<Mathematics for Machine Learning>

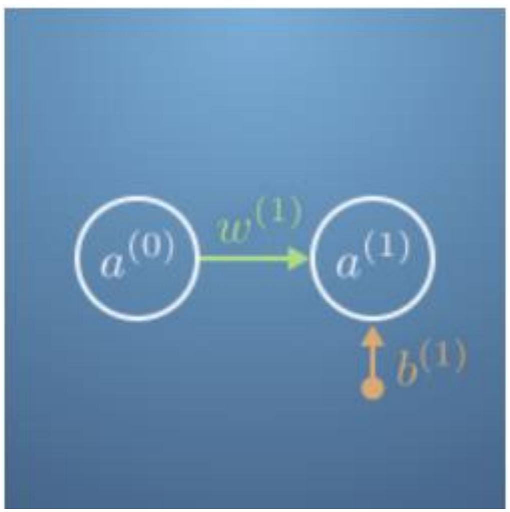
Univariate logistic regression

$$a^{[0]} = x^{[0]}$$

$$a^{[1]} = \sigma\left(w^{[1]} \cdot x^{[0]} + b^{[1]}\right)$$

# Artificial neurons

## Two neurons




<Mathematics for Machine Learning>

### Univariate logistic regression

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

$$z^{[1]} = w^{[1]} \cdot x^{[0]} + b^{[1]}$$

$$\sigma = \frac{1}{1+\exp(-z)}$$ : activation function

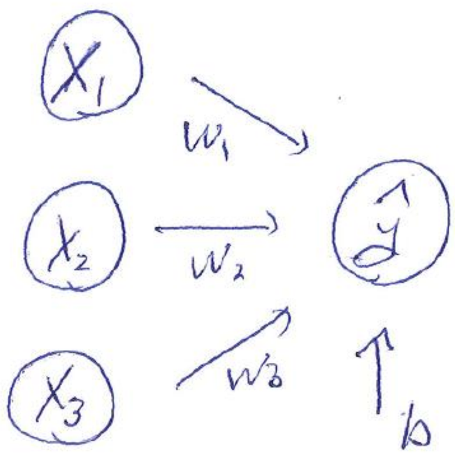
$a^{[0]}$ : input node

$a^{[1]}$ : activation node

or output node

# Neural Network

## Multivariate Logistic Regression

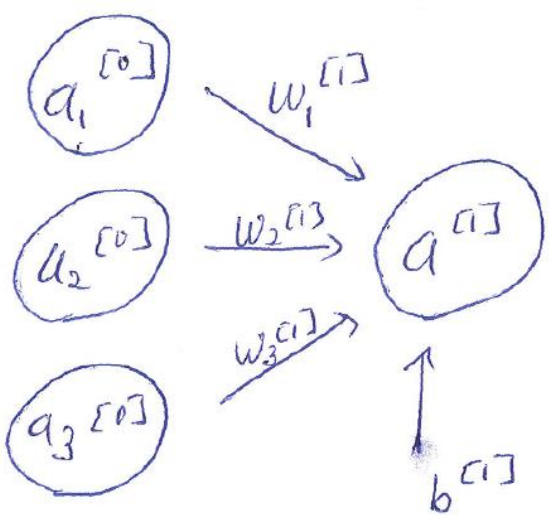



$$z = b + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$$

$$\hat{y} = \sigma(z)$$

# Neural Network

## Multivariate Logistic Regression

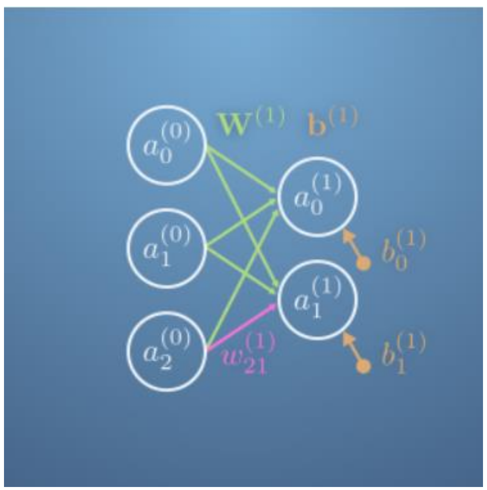



$$z^{[1]} = b^{[1]} + w_1^{[1]} \cdot a_1^{[0]} + w_2^{[1]} \cdot a_2^{[0]} + w_3^{[1]} \cdot a_3^{[0]}$$

$$a^{[1]} = \hat{y} = \sigma(z^{[1]})$$

# Neural network

## Multiclass classifier




<Mathematics for Machine Learning>

$$z_1^{[1]} = b_1^{[1]} + w_{11}^{[1]} \cdot a_1^{[0]} + w_{21}^{[1]} \cdot a_2^{[0]} + w_{31}^{[1]} \cdot a_3^{[0]}$$

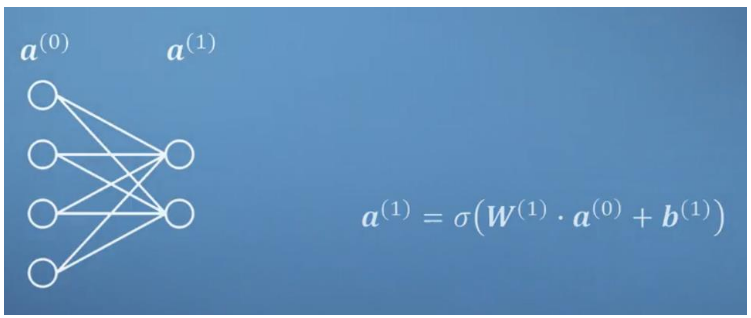
$$z_2^{[1]} = b_2^{[1]} + w_{12}^{[1]} \cdot a_1^{[0]} + w_{22}^{[1]} \cdot a_2^{[0]} + w_{32}^{[1]} \cdot a_3^{[0]}$$

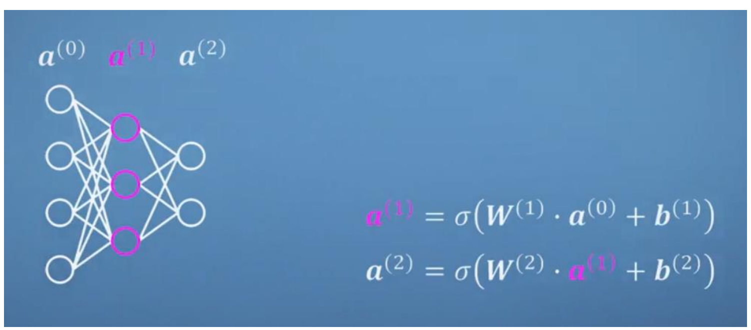
$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

# Neural network

## Hidden Layer

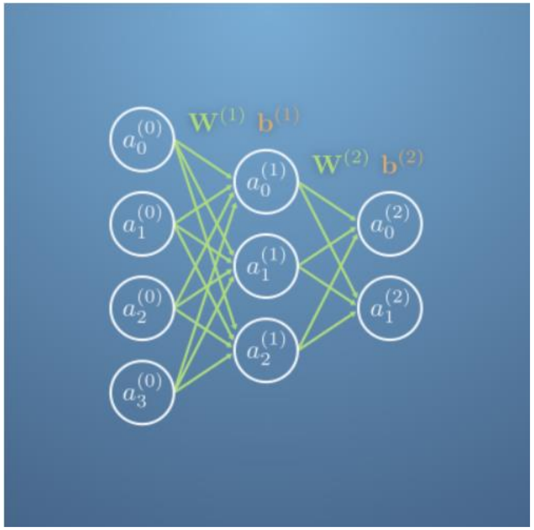



- Another layer (hidden layer) can be inserted between input layer and output layer.
- It works well if there is no direct (or strong) correlation between input layer and output layer.

# Neural Network

- Input layer : 4 neurons
- Hidden layer : 3 neurons




<Mathematics for Machine Learning>

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

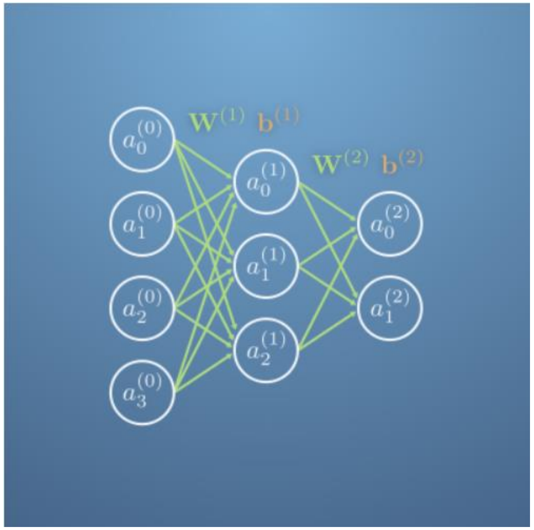
$$a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_1^{[1]} = b_1^{[1]} + w_{11}^{[1]} \cdot a_1^{[0]} + w_{21}^{[1]} \cdot a_2^{[0]} + w_{31}^{[1]} \cdot a_3^{[0]} + w_{41}^{[1]} \cdot a_4^{[0]}$$

$$z_2^{[1]} = b_2^{[1]} + w_{12}^{[1]} \cdot a_1^{[0]} + w_{22}^{[1]} \cdot a_2^{[0]} + w_{32}^{[1]} \cdot a_3^{[0]} + w_{42}^{[1]} \cdot a_4^{[0]}$$

$$z_3^{[1]} = b_3^{[1]} + w_{13}^{[1]} \cdot a_1^{[0]} + w_{23}^{[1]} \cdot a_2^{[0]} + w_{33}^{[1]} \cdot a_3^{[0]} + w_{43}^{[1]} \cdot a_4^{[0]}$$

# Neural Network




- Hidden layer : 3 neurons
- output layer : 2 neurons

$$a_1^{[2]} = \sigma(z_1^{[2]})$$

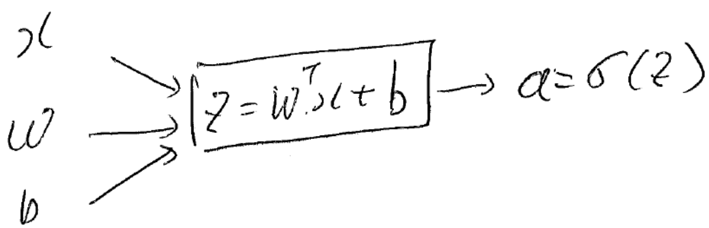
$$a_2^{[2]} = \sigma(z_2^{[2]})$$

$$z_1^{[2]} = b_1^{[2]} + w_{11}^{[2]} \cdot a_1^{[1]} + w_{21}^{[2]} \cdot a_2^{[1]} + w_{31}^{[2]} \cdot a_3^{[1]}$$

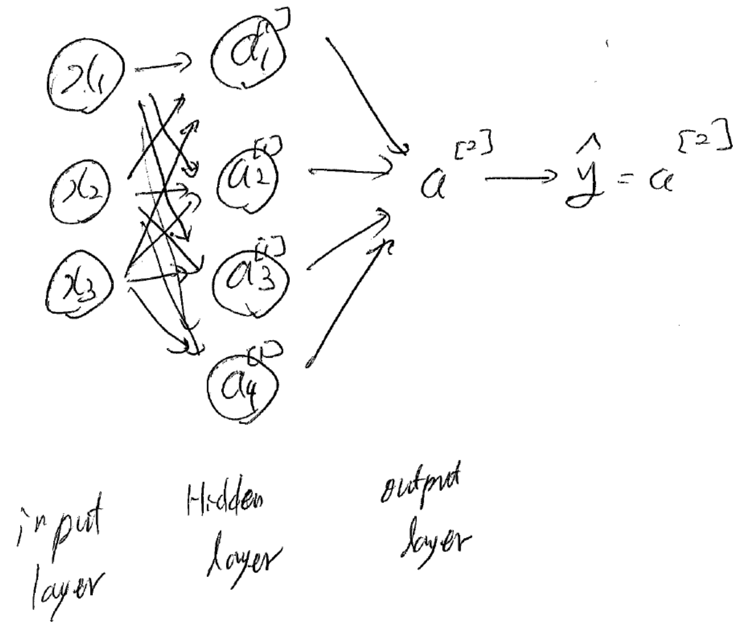
$$z_2^{[2]} = b_2^{[2]} + w_{12}^{[2]} \cdot a_1^{[1]} + w_{22}^{[2]} \cdot a_2^{[1]} + w_{32}^{[2]} \cdot a_3^{[1]}$$

# Neural Network

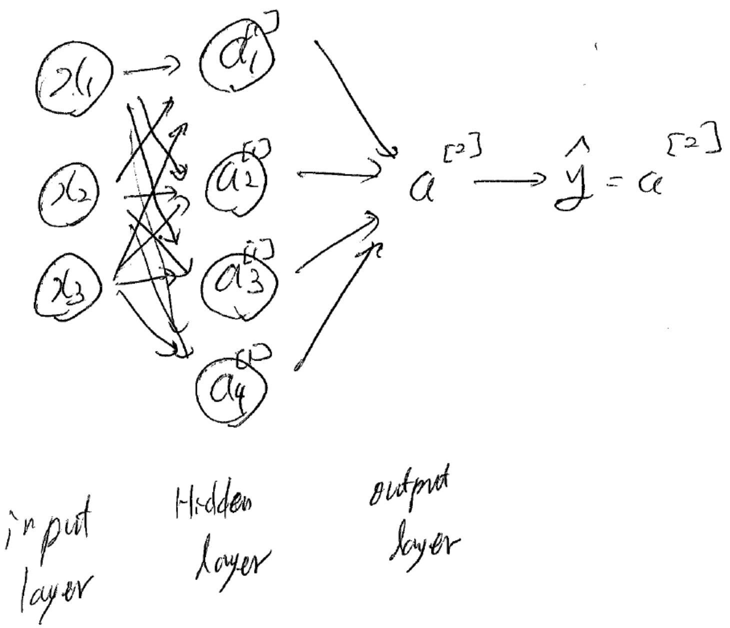
Logistic regression




Neural Network



# Neural Network



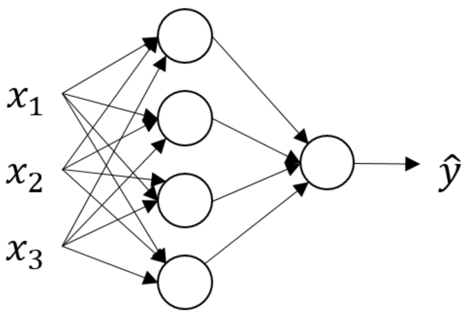

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

# Neural network



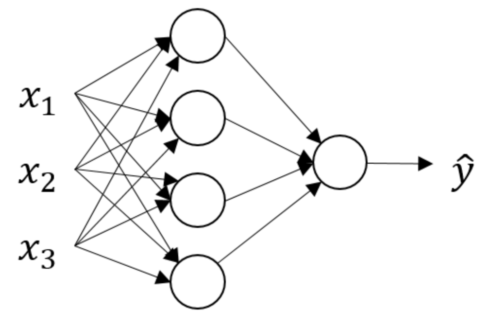

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

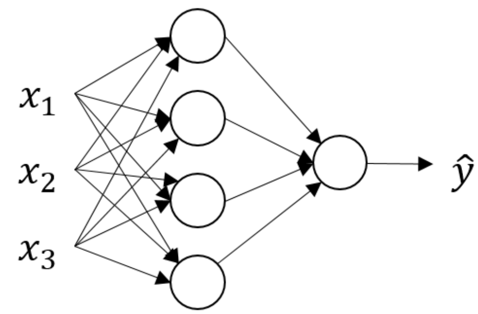
# Neural Network




$$z^{[1]} = \begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

$$= \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

# Neural Network




$$z^{[1]} = W^{[1]} x + b^{[1]}$$
(4,1) (4,3) (3,1) (4,1)

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

$$a^{[1]} = \sigma(z^{[1]})$$
(4,1) (4,1)

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$
(1,1) (1,4) (4,1) (1,1)

$$a^{[2]} = \sigma(z^{[2]})$$
(1,1) (1,1)

# Summary

- Multiclass classification
- One-versus-all (OvA)
- One-versus-one (OvO)
- Softmax
- Neural Network