Lab 6

Multi-class Classification
&
Neural Networks

Muticlass Classification

The target variable that we want to predict can take multiple variables

$$y \in \{0, 1, 2, ..., N\}$$

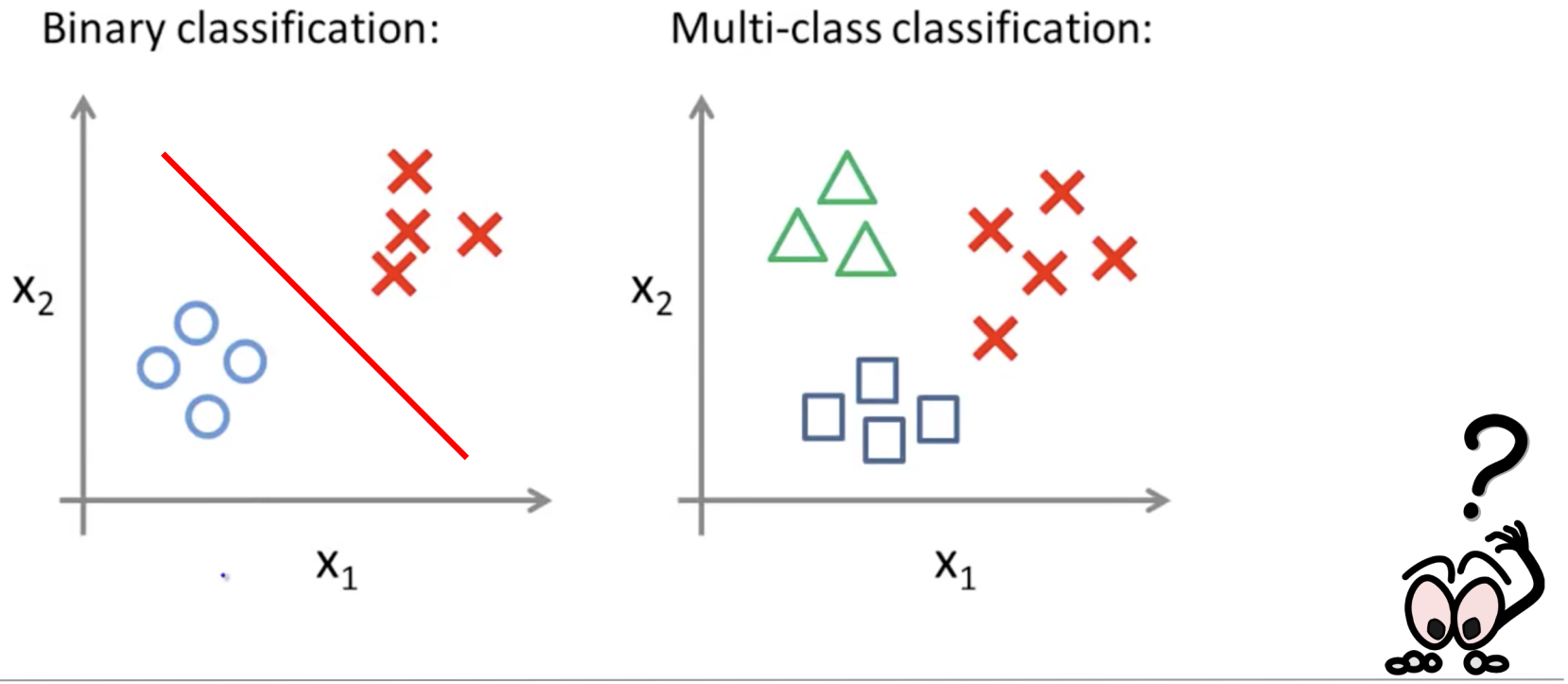
Email foldering:
Work, Friends, Family, Hobby

Weather:
Sunny, Windy, Cold, Rain, Snow

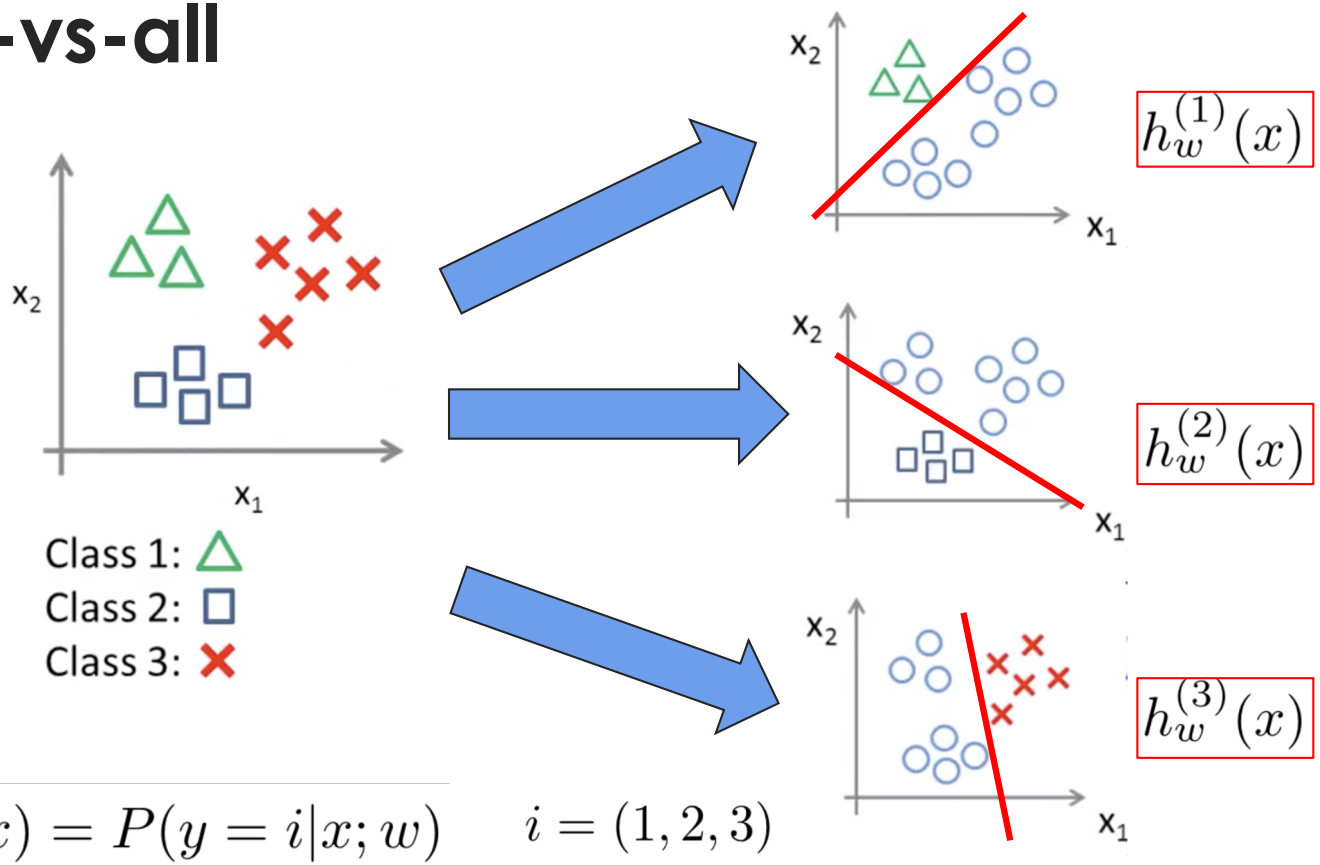
Recognising numbers
0,1,2,3,4,5,6,7,8,9

Example taken from Pang-Ning Tan et al. - Introduction to Data Mining

Multi-Class Classification



One-vs-all



$$h_w^{(i)}(x) = P(y = i|x; w) \qquad i = (1, 2, 3)$$

One-vs-all

- Train a logistic regression classifier $h_w^{(i)}(x)$ for each class *i*
  - Each classifier will predict the probability that $y = i$

- On a new input ***x*** , to make a prediction, we run all the classifiers and pick up the class *i* that maximises all the classifiers

$$\max_i h_w^{(i)}(x)$$

Example – recognize numbers

. . .

- We want to recognize numbers
- y = [0,1,2,3,4,5,6,7,8,9]
- We want to use logistic regression to classify the numbers
- ***How many logistic regressions we should run (how many classifiers we need)?***

The number of classifiers : 10 (for 0,1,2…9)

Assignment Lab 6 – numbers

1
2
.
.
.
5000

- Number of examples m=5000
- Number of features (20x20 pixel grade scale): n = 400 ($x_1 \ldots x_{400}$)
- Load data:
  - X(5000x400)
  - Y(5000)

$$X = \begin{bmatrix} x_1^{(1)} x_2^{(1)} \ldots x_{400}^{(1)} \\ x_1^{(2)} x_2^{(2)} \ldots x_{400}^{(2)} \\ x_1^{(3)} x_2^{(3)} \ldots x_{400}^{(3)} \\ \ldots. \\ x_1^{(m)} x_2^{(m)} \ldots x_{400}^{(m)} \end{bmatrix} \qquad Y = \begin{bmatrix} 1 \\ 9 \\ \ldots \\ 7 \\ 4 \\ 1 \\ 0 \\ \ldots \\ 7 \end{bmatrix}$$

Logistic regression for each class

. . .

- Logistic expression deals with Yes (1) and No (0)
- $h_w^{(i)}(x) = \text{sigmoid}(z) \in [0,1]$
- Our classes are $c \in [0,1..9]$
- **Value of the result in the regression 0 or 1**

- We do logistic regression for each $c \in [0,1..9]$
- The maximum values of $h_w^{(i)}(x)$ for specific c is set at prediction for c

Full vectorization – work with vectors as much as possible

1
2
.
.
.
5000

- Number of examples m=5000
- Number of features: n = 401 (x_0 + $x_1 \ldots x_{400}$)
- Number of classifiers: ck = 10
- ***For each classifier, what is the dimension of w?***

$size(w) = (n+1)$ $\quad w = [\, w_0\ w_1\ w_2\ \ldots\ w_n \,]$

For all classes $\quad all\_w = \begin{bmatrix} w_{0(1)}\ w_{1(1)}\ \ldots\ w_{n(1)} \\ w_{0(2)}\ w_{1(2)}\ \ldots\ w_{n(2)} \\ w_{0(3)}\ w_{1(3)}\ \ldots\ w_{n(3)} \\ \ldots \\ w_{0(ck)}\ w_{n(ck)}\ \ldots\ w_{n(ck)} \end{bmatrix}$

Full vectorization

1
2

.
.
.

5000

- Number of examples m=5000
- Number of features: n = 401 (x_0 + $x_1 \ldots x_{400}$)
- Number o classifiers: ck = 10
- ***How does h_w looks like***
  - ***a) for a specific class, b) as a matrix for all classes?***

$size(h_w) = m * 1$ $h_w = \begin{bmatrix} h_1 \\ h_2 \\ h_1 \\ \ldots \\ h_m \end{bmatrix}$ For all classes $h\_all = \begin{bmatrix} h_{0(1)}\, h_{0(2)} \ldots h_{0(ck)} \\ h_{1(1)}\, h_{1(2)} \ldots h_{1(ck)} \\ h_{21(1)}\, h_{0(2)} \ldots h_{2(ck)} \\ \ldots \\ h_{m(1)}\, h_{m(2)} \ldots h_{m(ck)} \end{bmatrix}$

What is the final result?

1
2

.
.
.

5000

$$h\_all = \begin{bmatrix} h_{0(1)}\, h_{0(2)} \ldots h_{0(ck)} \\ h_{1(1)}\, h_{1(2)} \ldots h_{1(ck)} \\ h_{21(1)}\, h_{0(2)} \ldots h_{2(ck)} \\ \ldots \\ h_{m(1)}\, h_{m(2)} \ldots h_{m(ck)} \end{bmatrix} = \begin{bmatrix} h_{0(1)} & h_{0(2)} \ldots & h_{0(10)} \\ h_{1(1)} & h_{1(2)} \ldots & h_{1(10)} \\ h_{21(1)} & h_{0(2)} \ldots & h_{2(10)} \\ \ldots & & \\ h_{5000(1)} & h_{5000(2)} \ldots & h_{5000(10)} \end{bmatrix}$$

What is the final hypothesis?

$$\begin{bmatrix} Index\,(max\,(h_{0(1)}\, h_{0(2)} \ldots h_{0(ck)})) \\ Index\,(\,max\,(\,h_{1(1)}\, h_{1(2)} \ldots h_{1(ck)})) \\ Index\,(\,max\,(\,h_{21(1)}\, h_{0(2} \ldots h_{2(ck)})) \\ \ldots \\ Index(\,max\,(\,h_{m(1)}\, h_{m(2)} \ldots h_{m(ck)})) \end{bmatrix}$$

The largest h per example
Matrix mx1

Part 2
Neural networks

Neuron model - Perceptron

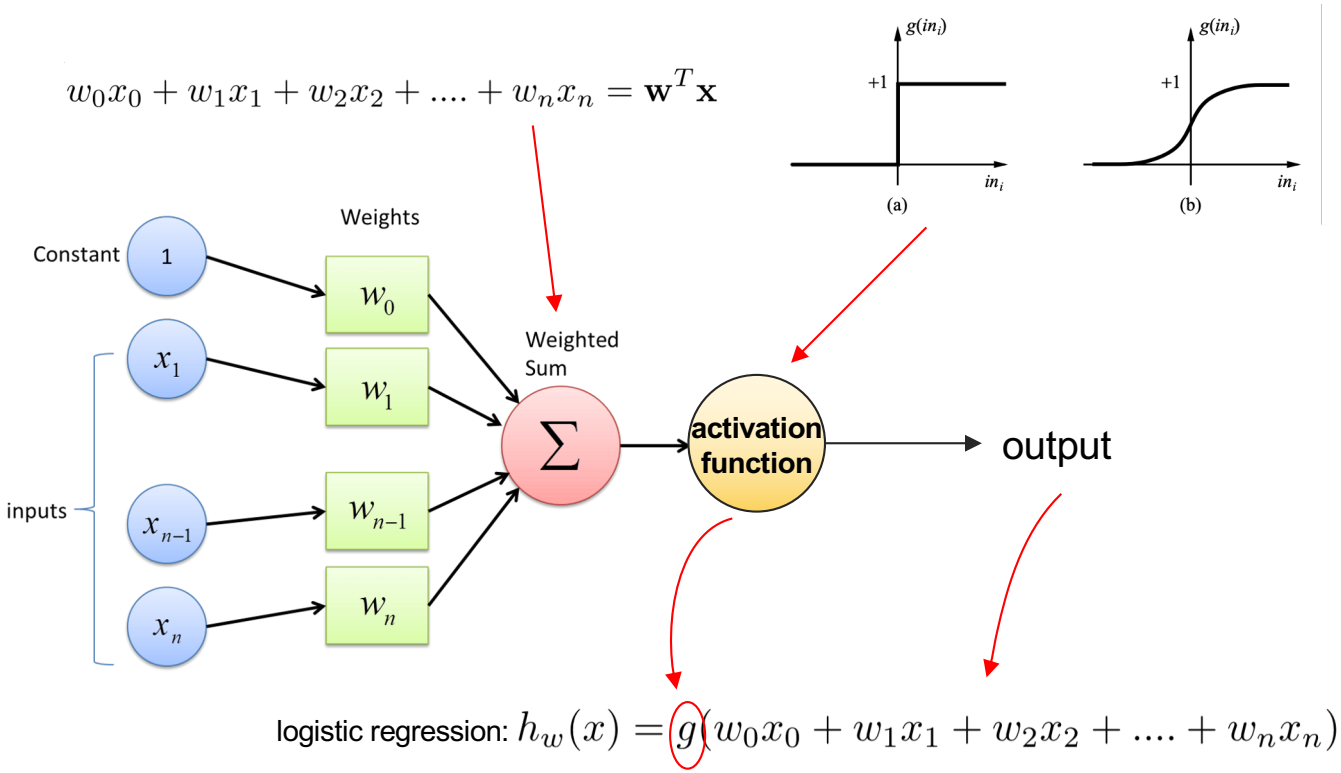
$x_0 = 1$

w_0 is the *bias* weight

Weights shows the strength of the particular node.

A **bias** value allows you to shift the activation function curve up or down.

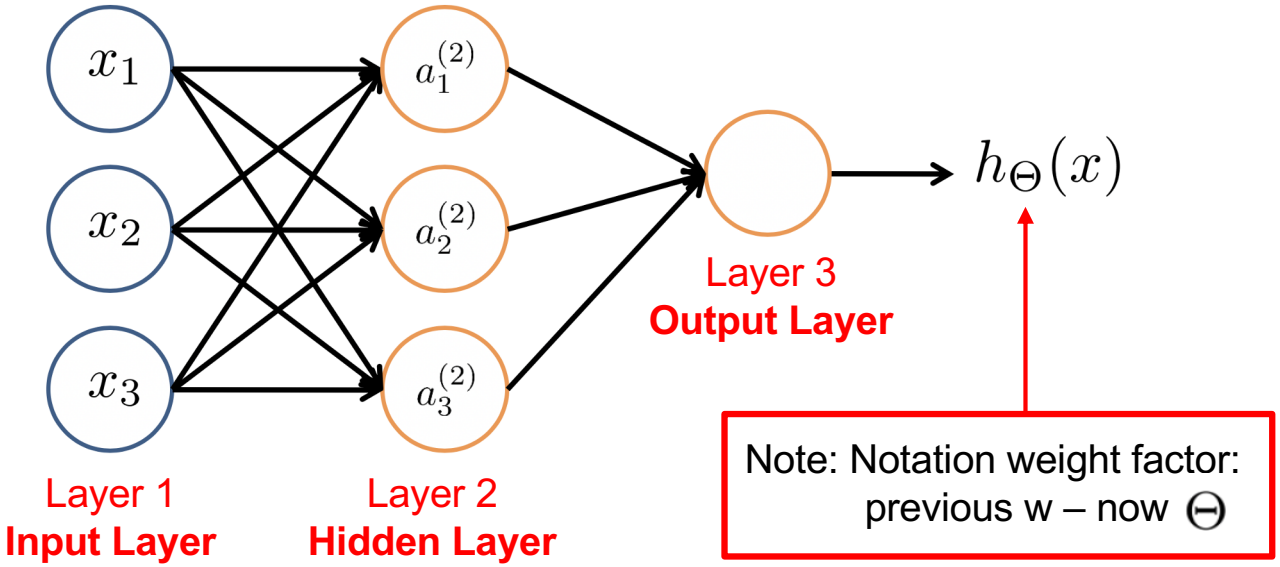
$$w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n = \mathbf{w}^T \mathbf{x}$$



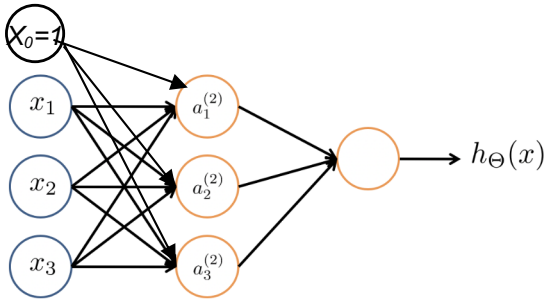
logistic regression: $h_w(x) = g(w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n)$

Artificial Neural Network

Multi-Layer Perceptrons architecture or Vanilla Neural Network



Artificial Neural Network



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

g is the **activation function** (e.g. Sigmoid)

vectors multiplication

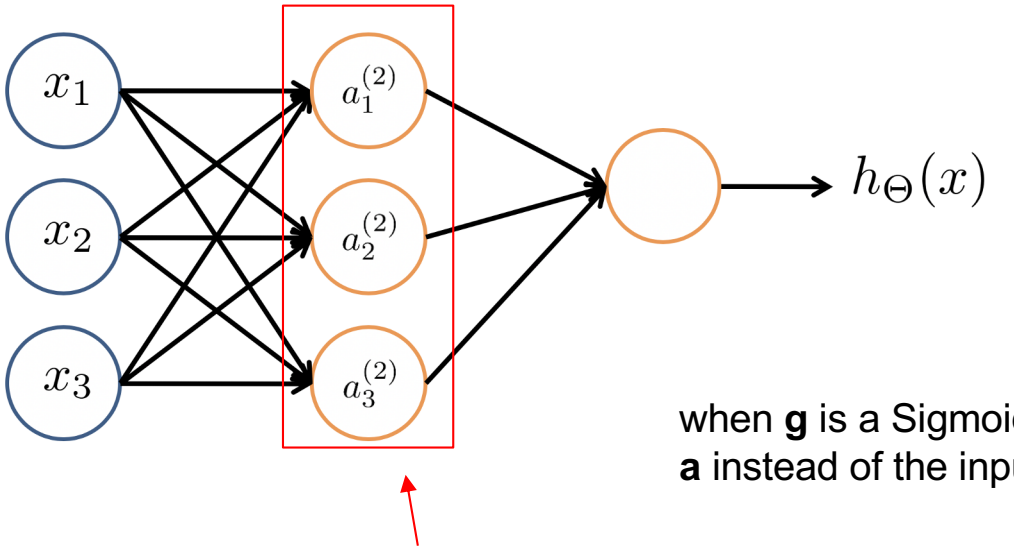
In the example:

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Artificial Neural Network

Forward Propagation



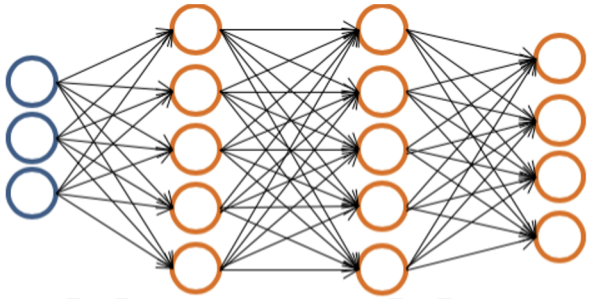
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

when **g** is a Sigmoid, it's doing Logistic regression on **a** instead of the input features **x**…

The features vector **a** is *learned* from the input features!

Neural Networks

Multiple output units



$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

Neural Networks - Cost Function

Logistic Regression:

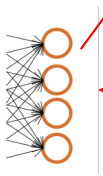
m samples in the training set

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

Neural Network: Instead of having 1 logistic regression output unit, we have **K** of them

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$(h_\Theta(x))_i = i^{th}$ output



$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

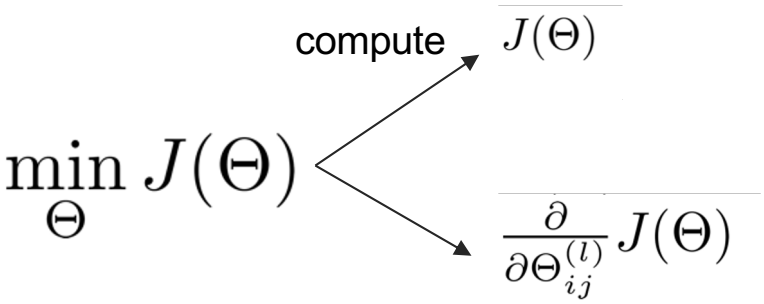
(Think of $\text{cost(i)} \approx (h_\Theta(x^{(i)}) - y^{(i)})^2$**)**

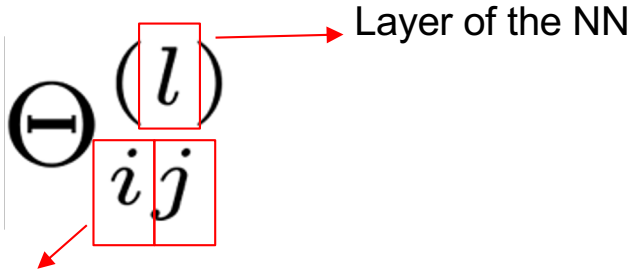
"how well is the network doing at predicting example *i*?"

Neural Networks - Backpropagation Algorithm

Minimizing the Cost Function

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$





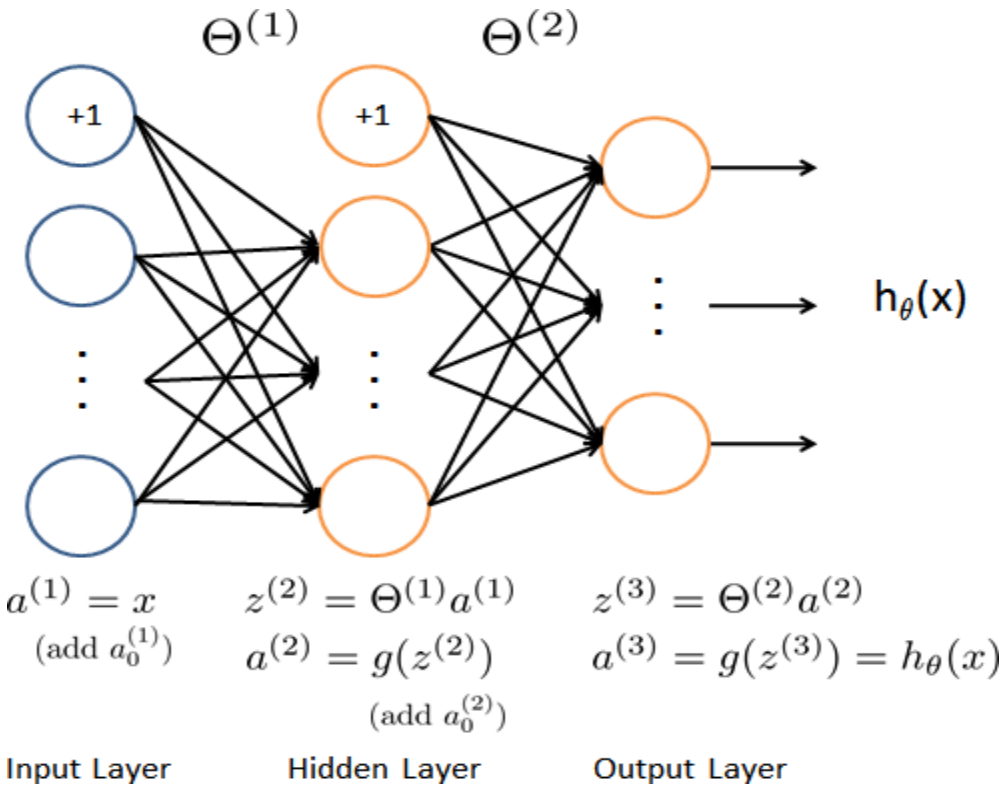
Lab 6. 2 – assignment Feedforward Propagation and Prediction

. . .

The 20 by 20 grid of pixels is “unrolled” into a 400-dimensional vector 5000 training examples.

$$X = \begin{bmatrix} — \ (x^{(1)})^T \ — \\ — \ (x^{(2)})^T \ — \\ \vdots \\ — \ (x^{(m)})^T \ — \end{bmatrix}$$

[5000 x 400]

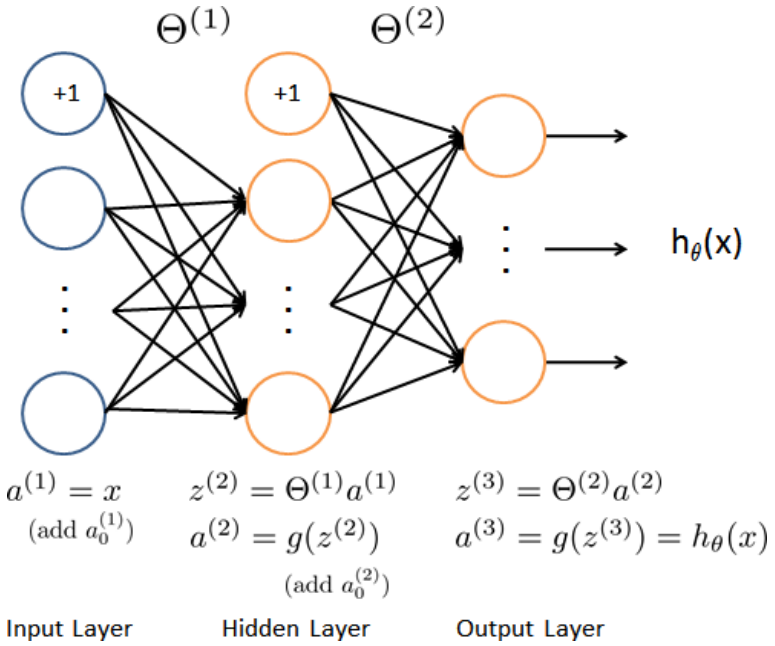


NN model

($\Theta^{(1)}$, $\Theta^{(2)}$) provided from the lab Theta1 and Theta2.

Lab 6. 2 – assignment Feedforward Propagation and Prediction

Implementation of Neural Network



- $(\Theta^{(1)}, \Theta^{(2)})$ provided from the lab
- Calculate
  1. Add $X_0 = 1$ to X matrix
  2. Compute $z^{(2)}$, $a^{(2)}$
  3. Add $a^{(2)}_0 = 1$ to $a^{(2)}$
  4. Compute $z^{(3)}$, $a^{(3)} = h_\theta(x)$
  5. Find max ($h_\theta(x)$)

Note: $h_\theta(x)$ = [5000 x 10]
Why?