MACHINE LEARNING (COURSERA)

SUPERVISED LEARNING

x := EXPLANATORY FEATURES

y := TARGET FEATURE

h := hypothesis FUNCTION

J := COST FUNCTION

θ := PARAMETERS (BETAS)

$$\left\{ \underset{\theta_0, \theta_1}{\text{MINIMIZE}} \frac{1}{2m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right)^2 \right.$$

$$\left\{ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right)^2 \right.$$

"MEAN SQUARED ERROR"

$$\left\{ h(x^{(i)}) = \theta_0 + \theta_1 . x^{(i)} \right.$$

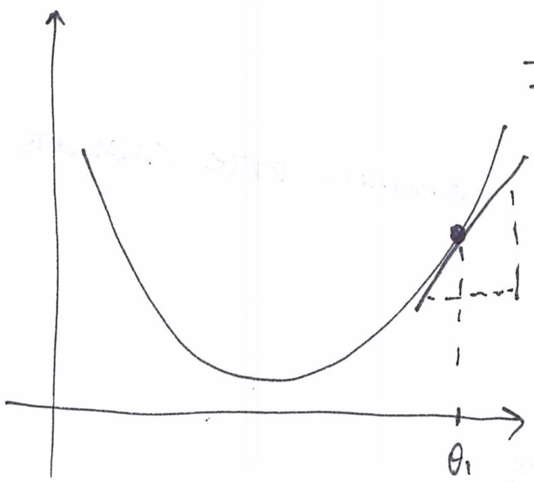
* LINEAR REGRESSION UNIVARIATE

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

"GRADIENT DESCENT"

$$\left[\begin{aligned} temp\_0 &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ temp\_1 &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_0 &:= temp\_0 \\ \theta_1 &:= temp\_1 \end{aligned} \right\}$$

"BATCH": EACH STEP OF GRADIENT DESCENT USES ALL TRAINING EXAMPLES.



$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\text{SLOPE POSITIVE } (\geq 0)}$$

$$\therefore \theta_1 := \theta_1 - \alpha . (\text{POSITIVE NUMBER})$$

So θ_1 IS GOING TO THE LEFT.

* α SMALL => CONVERGE IS SLOW
* α LARGE => PASS OVER MINIMUM OR DIVERGE!

FOR MULTIPLE FEATURES :

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \dots + \theta_n x_n \quad , x_0 = 1$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad , \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \therefore \boxed{h_\theta(x) = \theta^T . x}$$

FEATURE SCALING

FOR FASTER PROCESS OF GRADIENT DESCENT, IT MAY BE A GOOD IDEA TO SCALE EVERY FEATURE INTO THE FOLLOWING RANGE: $-1 \leq x_i \leq 1$

MEAN NORMALIZATION

$$x_1 \leftarrow \frac{x_1 - \mu_1}{S_1}$$

μ_1 := AVG VALUE OF x_1 IN TRAINING SET

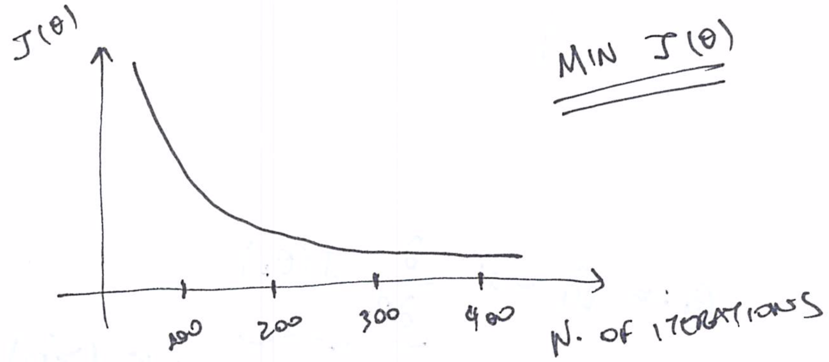
S_1 := RANGE (MAX − MIN)

MIN NORMALIZATION

$$x_1 \leftarrow \frac{x_1 - (\text{MIN } x_1)}{S_1}$$

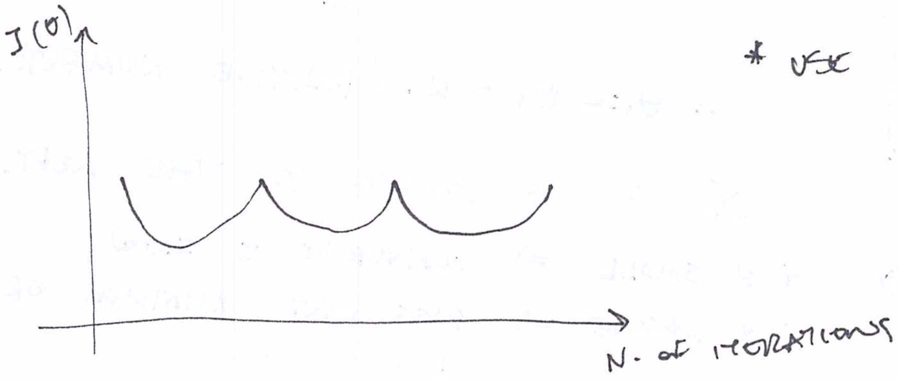
11

GRADIENT DESCENT & LEARNING RATE



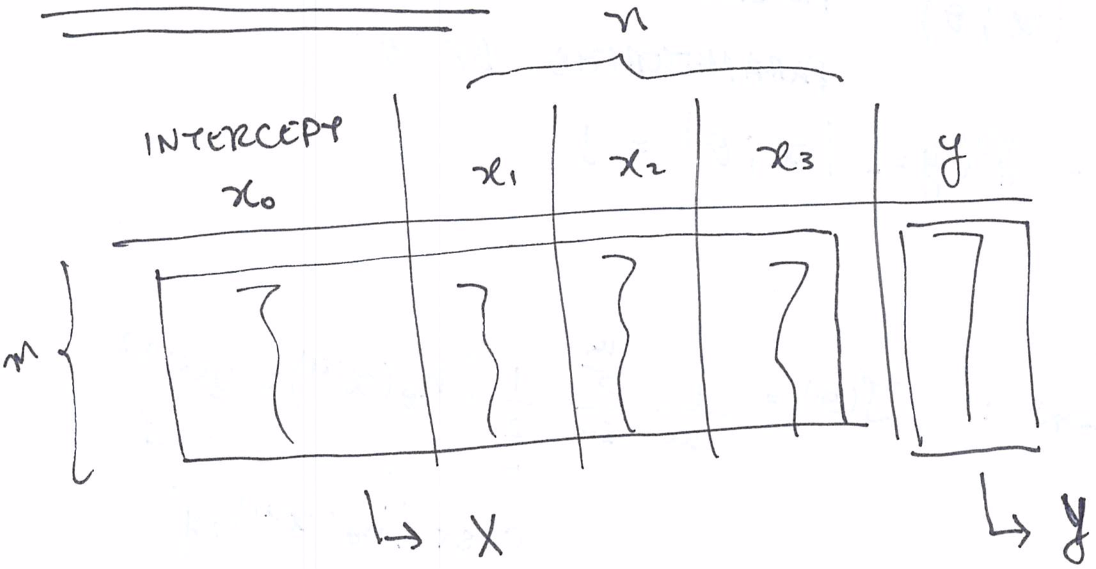
MIN J(θ)

* GRADIENT DESCENT IS WORKING IF VALUE OF J(θ) IS GETTING SMALLER ALONG THE ITERATIONS.



* USE LEARNING RATE SMALLER (α)

NORMAL EQUATION



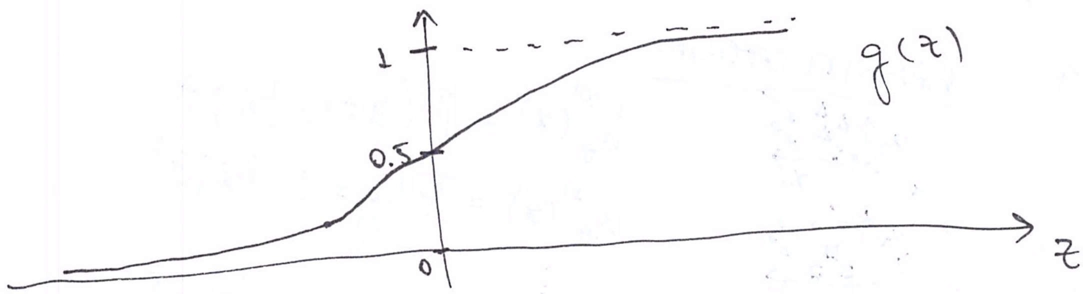
$$X \in \mathbb{R}^{m \times (n+1)}$$
$$y \in \mathbb{R}^{m}$$

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

GRADIENT DESCENT	NORMAL EQUATION
• NEED TO CHOOSE α	• NO NEED TO CHOOSE α
• NEEDS MANY ITERATIONS	• DON'T NEED TO ITERATE
• WORKS WELL EVEN WHEN n IS LARGE	• NEED TO COMPUTE $(X^T X)^{-1}$
	• SLOW IF n IS LARGE ($n > 10\,000$)

LOGISTIC REGRESSION MODEL

$0 \leq h_\theta(x) \leq 1$, $h_\theta(x) = g(\theta^T x)$, $g(z) = \dfrac{1}{1+e^{-z}}$ $\quad (z = \theta^T x)$



$$\therefore \; h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} =$$ ESTIMATED PROBABILITY THAT $y=1$ ON INPUT x

SIGMOID FUNCTION = LOGISTIC FUNCTION

$h_\theta(x) = \mathbb{P}(y=1 \mid x;\theta)$ "PROBABILITY THAT $y=1$, GIVEN x, PARAMETERIZED BY θ"

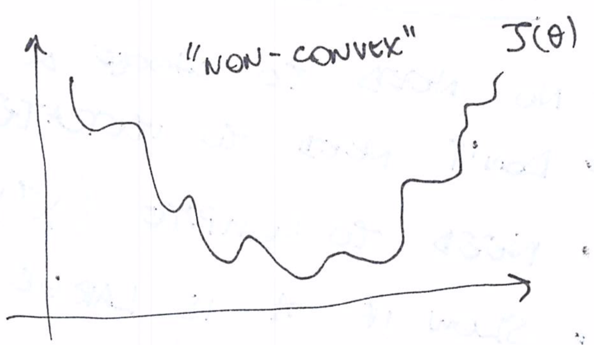
$$\mathbb{P}(y=0 \mid x;\theta) + \mathbb{P}(y=1 \mid x;\theta) = 1$$

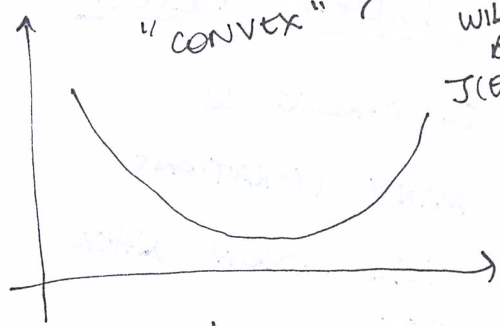
COST FUNCTION

LINEAR REGRESSION : $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{\text{COST}(h_\theta(x^{(i)}),\,y)}$

FOR LOGISTIC REGRESSION : $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ NON LINEAR

SO $J(\theta)$ WILL BE NON-CONVEX (WITH MULTIPLE LOCAL MIN.)



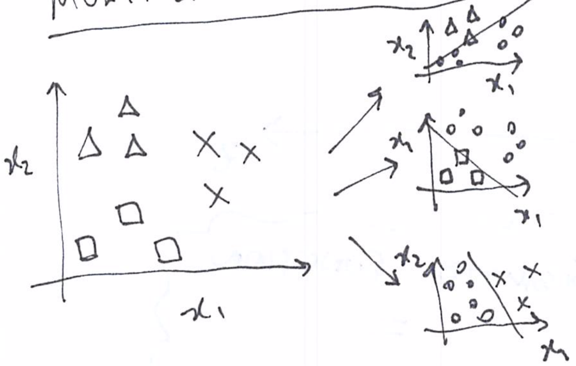


→ GRADIENT DESCENT WILL FIND A GLOBAL MINIMUM

LOGISTIC REGRESSION : $\text{COST}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & y=1 \\ -\log(1-h_\theta(x)), & y=0 \end{cases}$

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

MULTICLASS CLASSIFICATION



$h_\theta^{(0)}(x) = \mathbb{P}(y=0 \mid x;\theta)$

$h_\theta^{(1)}(x) = \mathbb{P}(y=1 \mid x;\theta)$

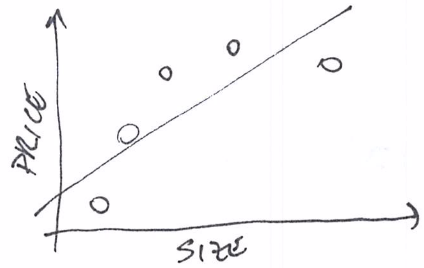
$\vdots$

$h_\theta^{(n)}(x) = \mathbb{P}(y=n \mid x;\theta)$

$$\boxed{\text{PREDICTION} = \max_i \left(h_\theta^{(i)}(x)\right)}$$

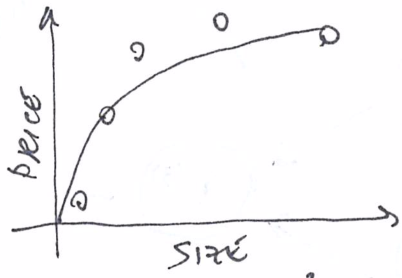
OVERFITTING

"LINEAR REGRESSION EXAMPLE"



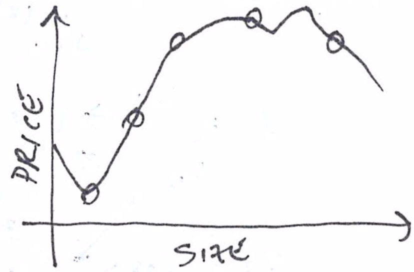
$\theta + \theta_1 x$

"UNDERFIT" / "HIGH BIAS"



$\theta_0 + \theta_1 x + \theta_2 x^2$

"JUST RIGHT"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"OVERFIT" / "HIGH VARIANCE"

DEF.: IF TOO MANY FEATURES, THE MODEL MAY FIT THE TRAINING SET TOO WELL ($J(\theta) \approx 0$), BUT FAIL TO GENERALIZE TO NEW EXAMPLE

MEANING:

- BIAS: MODEL HAS A BIAS, IN A WAY THAT DOESN'T FIT TO THE DATA (OR LOOK).
- VARIANCE: MODEL ~~TOO~~ IS VARIABLE, IN A WAY THAT IT CAN FIT PERFECTLY AT ANY **DATA**.

SOLVING OVERFITTING: SOME OPTIONS

1. REDUCE NUMBER OF FEATURES (MANUALLY / MODEL SELECTION)
2. REGULARIZATION (REDUCES MAGNITUDE / VALUES OF PARAMETERS θ)

FORMULA

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

REDUCES INFLUENCE OF PARAMETERS θ (RISE COST BASED ON SUMMARIZING θ & λ)

$\lambda :=$ REGULARIZATION PARAMETER

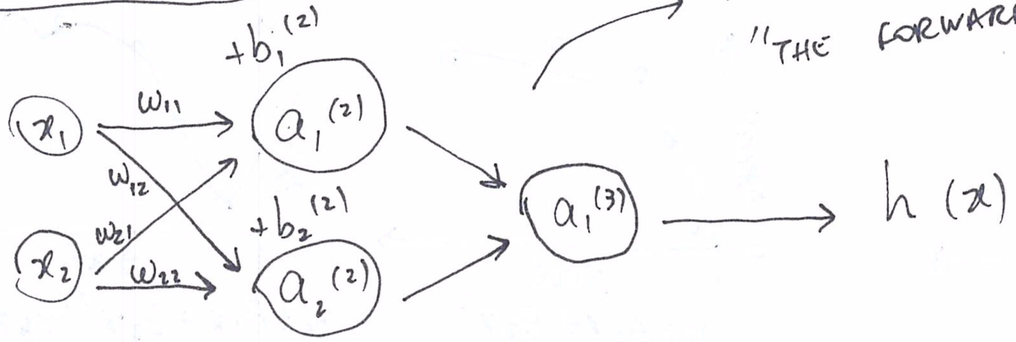
NEURAL NETWORKS

MOTIVATION: NON-LINEAR HYPOTHESES AND NUMBER OF FEATURES ARE TOO LARGE. EX.: IMAGE RECOGNITION.

MOREOVER: LINEAR METHODS ARE "WEAK" (MAKE STRONG ASSUMPTIONS) AND CAN ONLY EXPRESS RELATIVELY SIMPLE FUNCTIONS OF INPUTS.

MODEL REPRESENTATION

THIS PROCESS IS CALLED "THE FORWARD PROPAGATION"



LAYER 1 "INPUT LAYER"

LAYER 2 "HIDDEN LAYER"

LAYER 3 "OUTPUT LAYER"

$$a_1^{(2)} = g\left(w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 + b_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(w_{12}^{(1)} \cdot x_1 + w_{22}^{(1)} \cdot x_2 + b_2^{(2)}\right)$$

$$^{*} z_1^{(2)} = w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 + b_1^{(2)}$$

$a_i^{(j)} :=$ "ACTIVATION" OF UNIT i IN LAYER j

$w_{ik}^{(j)} :=$ "WEIGHTS" OF LAYER j FROM UNIT $\underline{i}$ TO UNIT $\underline{k}$

$g(\cdot) :=$ ACTIVATION FUNCTION

MULTICLASS CLASSIFICATION

$y^{(i)}$ RANGES, FOR EXAMPLE, AS: $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

* KIND OF AN INTRODUCTION TO SOFTMAX

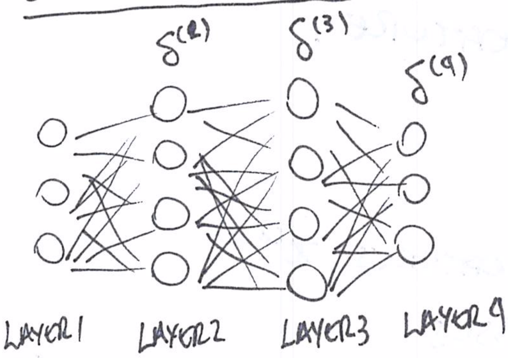
COST FUNCTION

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

$$+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(\Theta_{j,i}^{(l)}\right)^2$$

$$\begin{cases} L = \text{\# TOTAL LAYERS} \\ s_l = \text{\# UNITS IN LAYER } l \\ K = \text{\# OUTPUT UNITS/CLASSES} \\ m = \text{\# OF OBSERVATIONS} \end{cases}$$

BACK PROPAGATION



$$\delta_j^{(4)} = a_j^{(4)} - y_j$$ (PREDICT MINUS OBSERV.)

$$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* g'(z^{(3)})$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g'(z^{(2)})$$

"ALGORITHM"

TRAINING SET $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

SET $\Delta_{ij}^{(l)} = 0$

FOR $i = 1$ TO m:

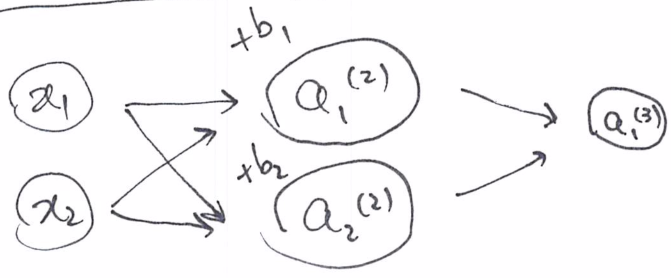
- SET $a^{(1)} = x^{(i)}$
- PERFORM FORWARD PROPAGATION TO COMPUTE $a^{(l)}$ $(l = 2, 3, \ldots, L)$
- USING $y^{(i)}$, COMPUTE $\delta^{(L)} = a^{(L)} - y^{(i)}$
- COMPUTE $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$
- $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \cdot \delta_i^{(l+1)}$

$$\begin{cases} D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}, & j \neq 0 \\ D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}, & j = 0 \end{cases}$$

(DO NOT USE REGULARIZATION ON BIAS)

$$\therefore \boxed{\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}}$$

WEIGHTS RANDOM INITIALIZATION



$\boxed{\Theta_{ij}^{(l)} = 0}$, $\forall i, j, l \Rightarrow$

$a_1^{(2)} = a_2^{(2)}$, $\delta_1^{(2)} = \delta_2^{(2)}$,

$\frac{\partial}{\partial \Theta_{01}^{(1)}} J(\Theta) = \frac{\partial}{\partial \Theta_{02}^{(1)}} J(\Theta)$

IF ALL HIDDEN UNITS ARE IDENTICAL THEN IT WOULD BE LIKE A MODEL OF ONLY ONE FEATURE.

SUMMARY

* NUMBER OF INPUT UNITS = DIMENSION OF FEATURES $x^{(i)}$
* NUMBER OF OUTPUT UNITS = NUMBER OF CLASSES
* NUMBER OF HIDDEN UNITS PER LAYER = USUALLY THE MORE THE BETTER
* NUMBER OF HIDDEN LAYERS = DEFAULT IS 1, BUT USUALLY THE MORE THE BETTER (# OF HIDDEN UNITS EQUAL BETWEEN LAYERS)

TRAINING A NN

1. RANDOMLY INITIALIZE THE WEIGHTS
2. IMPLEMENT FP TO GET $h_\theta(x^{(i)})$ FOR ANY $x^{(i)}$
3. IMPLEMENT COST FUNCTION
4. IMPLEMENT BP TO COMPUTE PARTIAL DERIVATIVES
5. USE GRADIENT CHECKING TO CONFIRM THAT BP WORKS. THEN DISABLE GRADIENT CHECKING.
6. USE GRADIENT DESCENT OR A BUILT-IN OPTIMIZATION FUNCTION TO MIN $J(\Theta)$ WITH WEIGHTS.