Week 3

Classification and Representation

Classification

- one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

Hypothesis Representation

- Sigmoid Function/Logistic Function
- $$h_\theta(x) = g(\theta^T x)$$
  $$z = \theta^T x$$
  $$g(z) = \frac{1}{1 + e^{-z}}$$
- **$h_\theta(x)$ will give us the probability that our output is 1**. For example, $h_\theta(x)=0.7$ gives us a probability of 70% that our output is 1.

Decision Boundary

- $h_\theta(x)=g(\theta^T x)\geq 0.5 \rightarrow \theta^T x \geq 0 \Rightarrow y=1;$
- $h_\theta(x))=g(\theta^T x)<0.5 \rightarrow \theta^T x<0 \Rightarrow y=0$
- Decision boundary: $\theta^T x = 0$

- The decision boundary is a property, not of the trading set, but of the hypothesis under the parameters. The training set is not what we use to define the decision boundary. The training set may be used to fit the parameters theta.

Logistic Regression Model

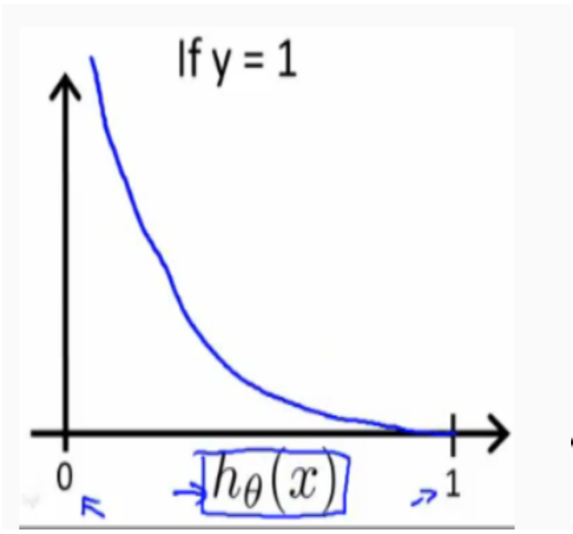
Cost Function

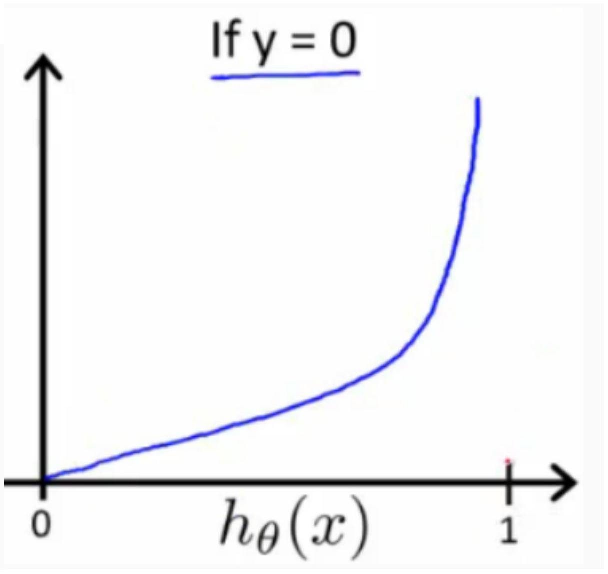
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y} = 1$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y} = 0$$

Note writing the cost function in this way guarantees that J(θ) is convex for logistic regression.





$$\mathrm{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$

$$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$

$$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

Simplified Cost Function and Gradient Descent

Compress our cost function's two conditional cases into one case (since y only could be 0 or 1):

$$\mathrm{Cost}(h_\theta(x), y) = -y\ \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

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

A vectorized implementation is:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m}\cdot\left(-y^T\log(h) - (1-y)^T\log(1-h)\right)$$

Gradient Descent

Remember that the general form of gradient descent is:

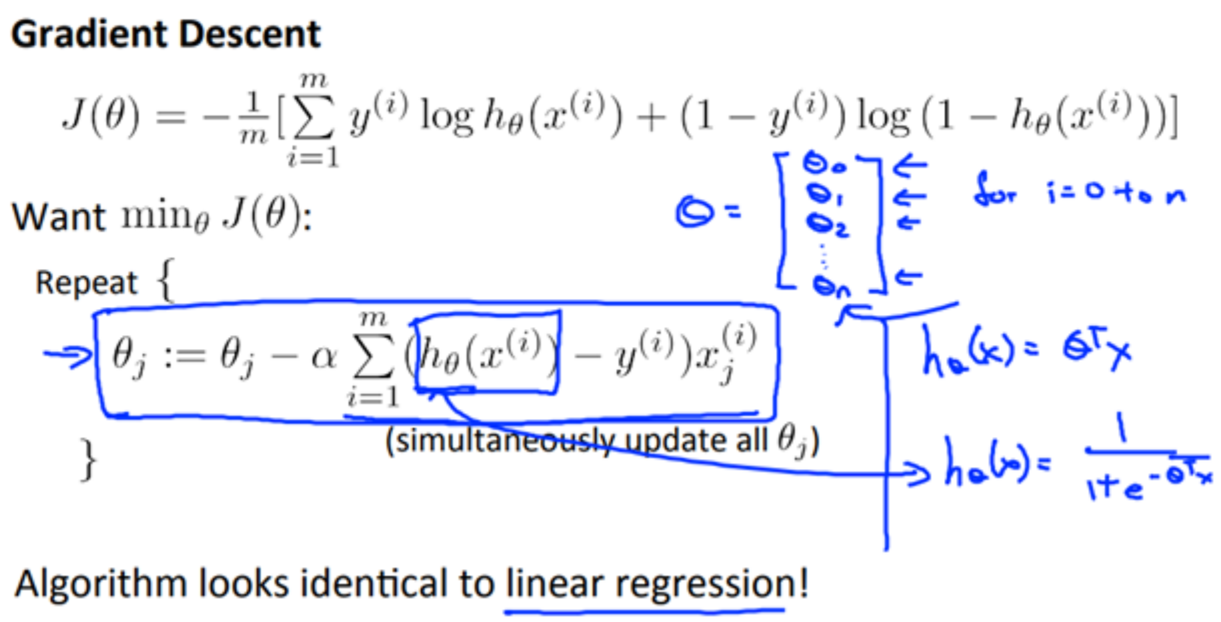
$$\begin{aligned}&Repeat\ \{\\ &\quad \theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)\\ &\}\end{aligned}$$

We can work out the derivative part using calculus to get:

$$\begin{aligned}&Repeat\ \{\\ &\quad \theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\\ &\}\end{aligned}$$

A vectorized implementation is:

$$\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \vec{y})$$



Gradient Descent

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

Want $\min_\theta J(\theta)$:

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ for i=0 to n

Repeat {

$$\boxed{\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (\boxed{h_\theta(x^{(i)})} - y^{(i)}) x_j^{(i)}}$$

(simultaneously update all θ_j)

}

$h_\theta(x) = \theta^T x$

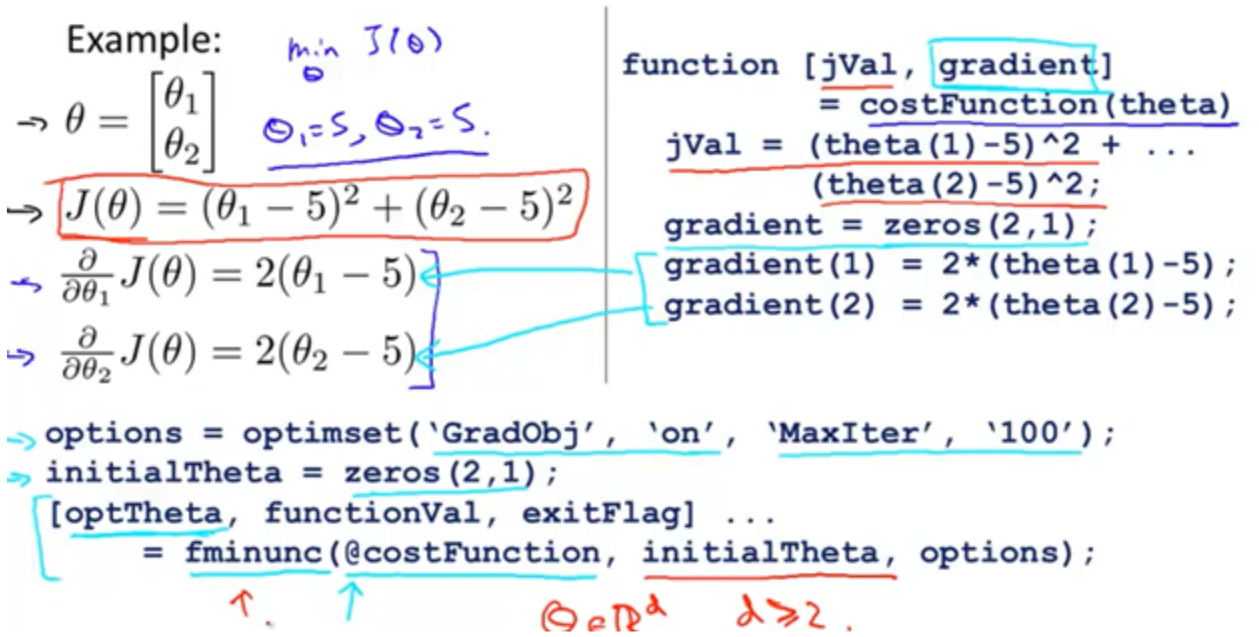
$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Algorithm looks identical to linear regression!

Note: $J(\theta)$ for logistic regression is similar to linear regression, but $h_\theta(x)$ are different.

Advanced Optimization

"Conjugate gradient", "BFGS", and "L-BFGS" are faster than Gradient Descent but complex.



Example: $\min_\theta J(\theta)$

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $\theta_1 = 5, \theta_2 = 5.$

$$\boxed{J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2}$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
           = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1)  = 2*(theta(1)-5);
  gradient(2)  = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
      = fminunc(@costFunction, initialTheta, options);
```

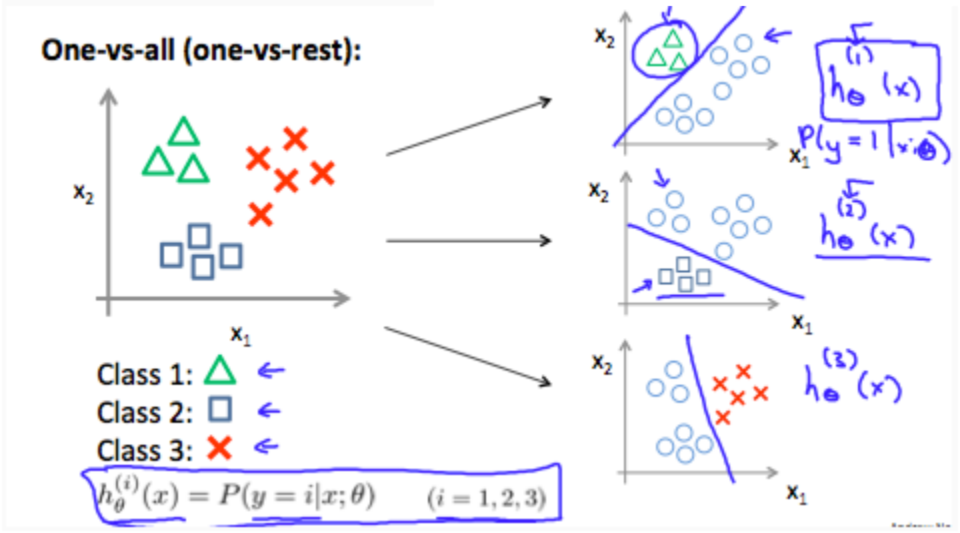
$\theta \in \mathbb{R}^d$ $d \geq 2.$

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{array}{l} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{array}$$

```
function [jVal, gradient] = costFunction(theta)

    jVal = [ code to compute $J(\theta)$ ];

    gradient(1) = [code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$];

    gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$];

      ⋮

    gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$ ];
```

Multiclass Classification: one-vs-all

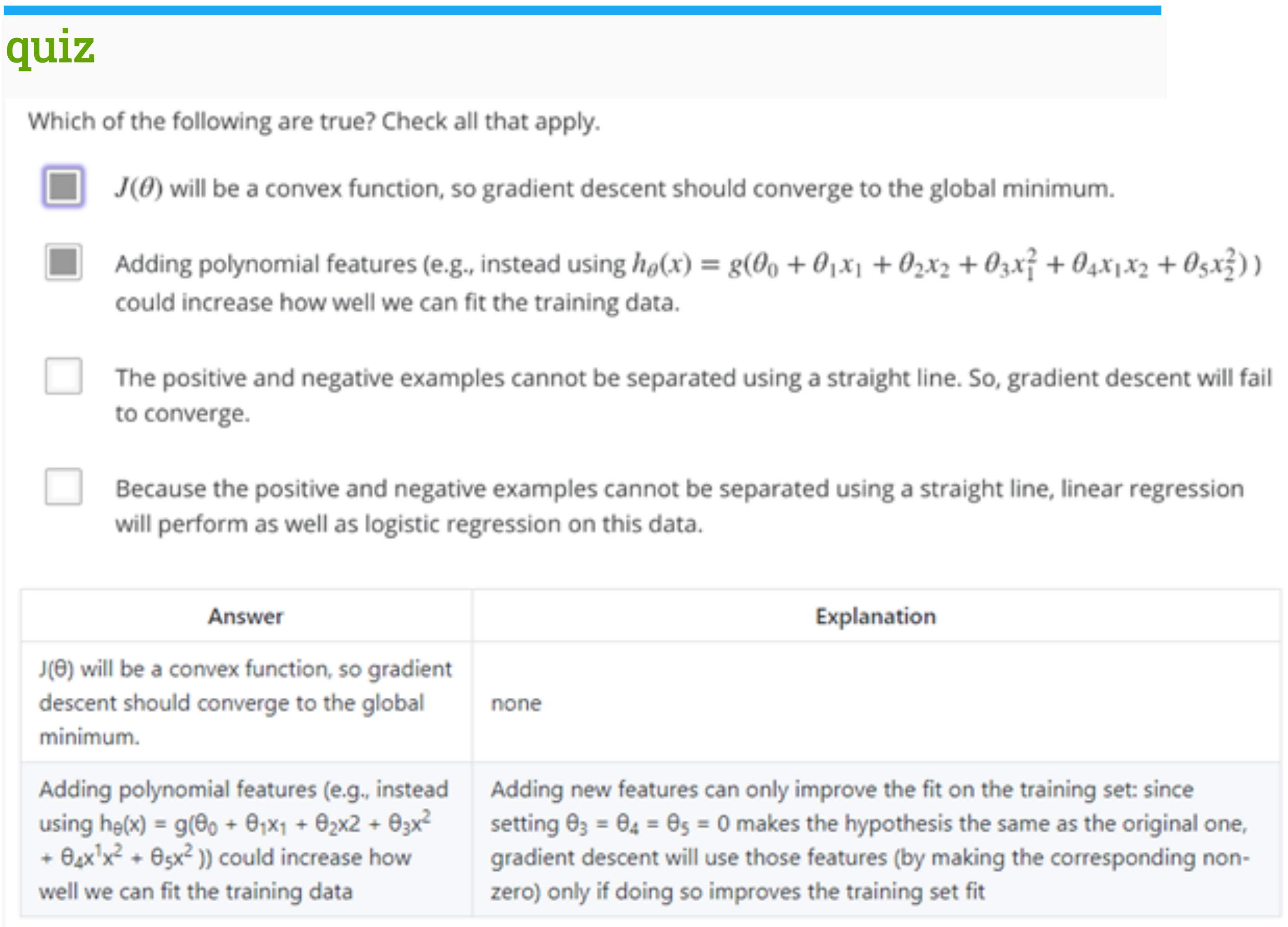
$$\begin{aligned}
&y \in \{0, 1 \ldots n\} \\
&h_\theta^{(0)}(x) = P(y = 0|x;\theta) \\
&h_\theta^{(1)}(x) = P(y = 1|x;\theta) \\
&\cdots \\
&h_\theta^{(n)}(x) = P(y = n|x;\theta) \\
&\text{prediction} = \max_i(h_\theta^{(i)}(x))
\end{aligned}$$



To summarize:

Train a logistic regression classifier $h\theta(x)$ for each class￼ to predict the probability that ￼ ￼y = i￼ ￼.

To make a prediction on a new x, pick the class ￼that maximizes $h\theta(x)$

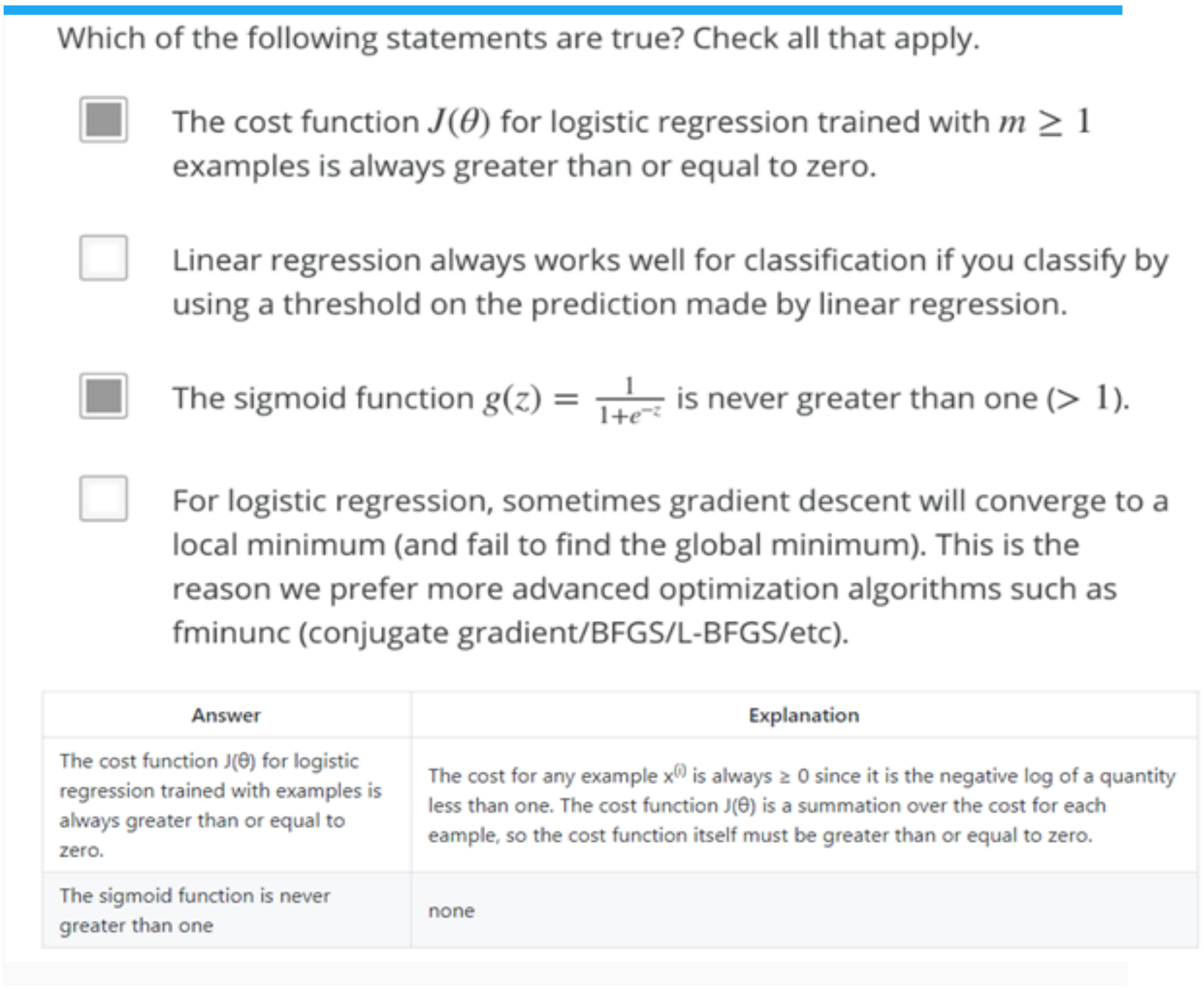


quiz

Which of the following are true? Check all that apply.

- ☑ $J(\theta)$ will be a convex function, so gradient descent should converge to the global minimum.
- ☑ Adding polynomial features (e.g., instead using $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2)$) could increase how well we can fit the training data.
- ☐ The positive and negative examples cannot be separated using a straight line. So, gradient descent will fail to converge.
- ☐ Because the positive and negative examples cannot be separated using a straight line, linear regression will perform as well as logistic regression on this data.

Answer	Explanation
J(θ) will be a convex function, so gradient descent should converge to the global minimum.	none
Adding polynomial features (e.g., instead using $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2$x2 $+ \theta_3 x^2 + \theta_4 x^1 x^2 + \theta_5 x^2$)) could increase how well we can fit the training data	Adding new features can only improve the fit on the training set: since setting $\theta_3 = \theta_4 = \theta_5 = 0$ makes the hypothesis the same as the original one, gradient descent will use those features (by making the corresponding non-zero) only if doing so improves the training set fit



Which of the following statements are true? Check all that apply.

[✓] The cost function $J(\theta)$ for logistic regression trained with $m \geq 1$ examples is always greater than or equal to zero.

[ ] Linear regression always works well for classification if you classify by using a threshold on the prediction made by linear regression.

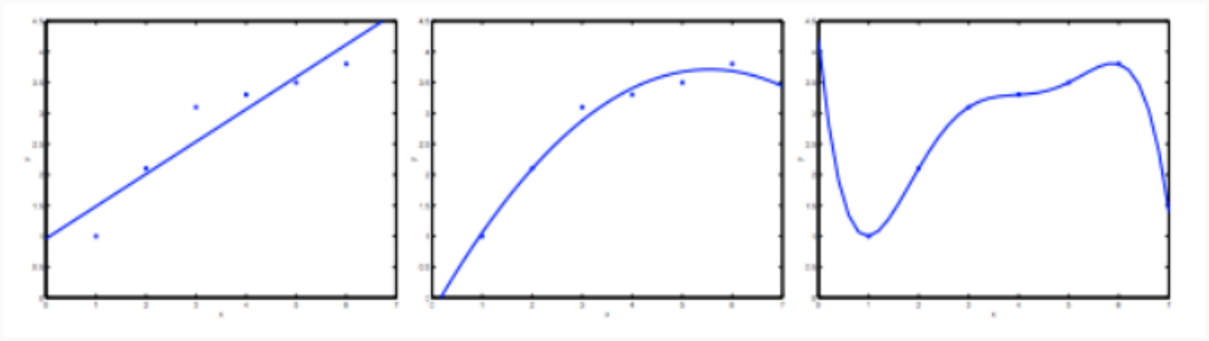
[✓] The sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ is never greater than one (> 1).

[ ] For logistic regression, sometimes gradient descent will converge to a local minimum (and fail to find the global minimum). This is the reason we prefer more advanced optimization algorithms such as fminunc (conjugate gradient/BFGS/L-BFGS/etc).

Answer	Explanation
The cost function J(θ) for logistic regression trained with examples is always greater than or equal to zero.	The cost for any example $x^{(i)}$ is always ≥ 0 since it is the negative log of a quantity less than one. The cost function J(θ) is a summation over the cost for each eample, so the cost function itself must be greater than or equal to zero.
The sigmoid function is never greater than one	none

Solving the Problem of Overfitting

The Problem of Overfitting



underfit/ high bias just right overfit/high variance

There are two main options to address the issue of overfitting:

1) Reduce the number of features:

- Manually select which features to keep.
- Use a model selection algorithm (studied later in the course).

2) Regularization

- Keep all the features, but reduce the magnitude of parameters θj.
- Regularization works well when we have a lot of slightly useful features.

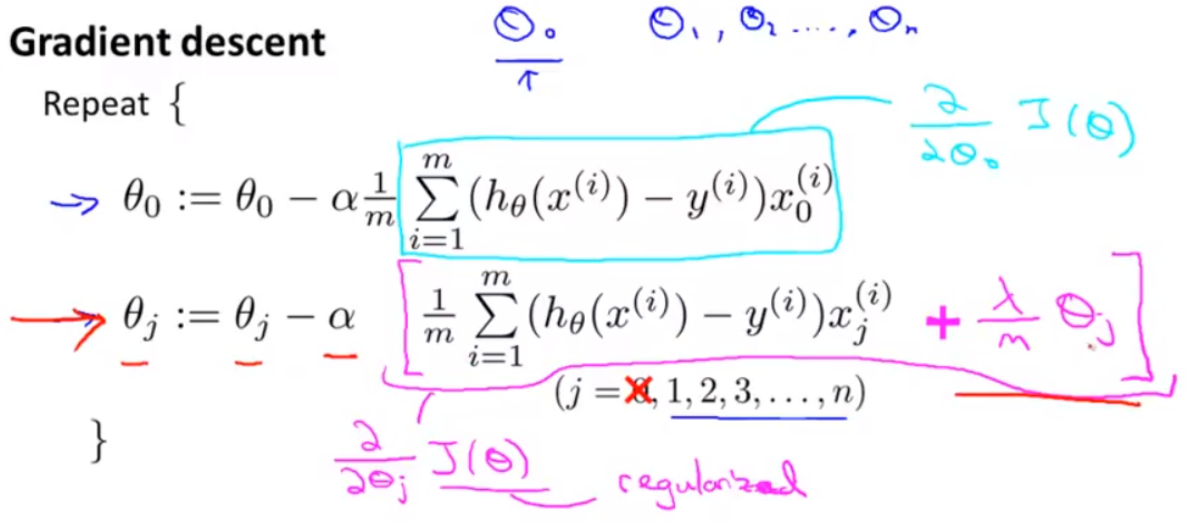
Cost Function

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

The λ, or lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are inflated. If lambda is too large, it may cause underfit problem.

Regularized Linear Regression

Gradient Descent



Gradient descent

$\Theta_0 \qquad \Theta_1, \Theta_2, \dots, \Theta_n$

Repeat {

$$\theta_0 := \theta_0 - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}} \qquad \frac{\partial}{\partial\Theta_0} J(\Theta)$$

$$\underline{\theta_j} := \underline{\theta_j} - \underline{\alpha} \underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\Theta_j\right]}_{\frac{\partial}{\partial\Theta_j} J(\Theta) \text{ regularized}}$$

$$(j = \cancel{0}, \underline{1, 2, 3, \dots, n})$$

}

separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha \ \frac{1}{m} \ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \ \left[\left(\frac{1}{m} \ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j\right] \qquad j \in \{1, 2...n\}$$

}

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$1 - \alpha\frac{\lambda}{m}$ will always be less than 1, since α is a small number

Normal Equation

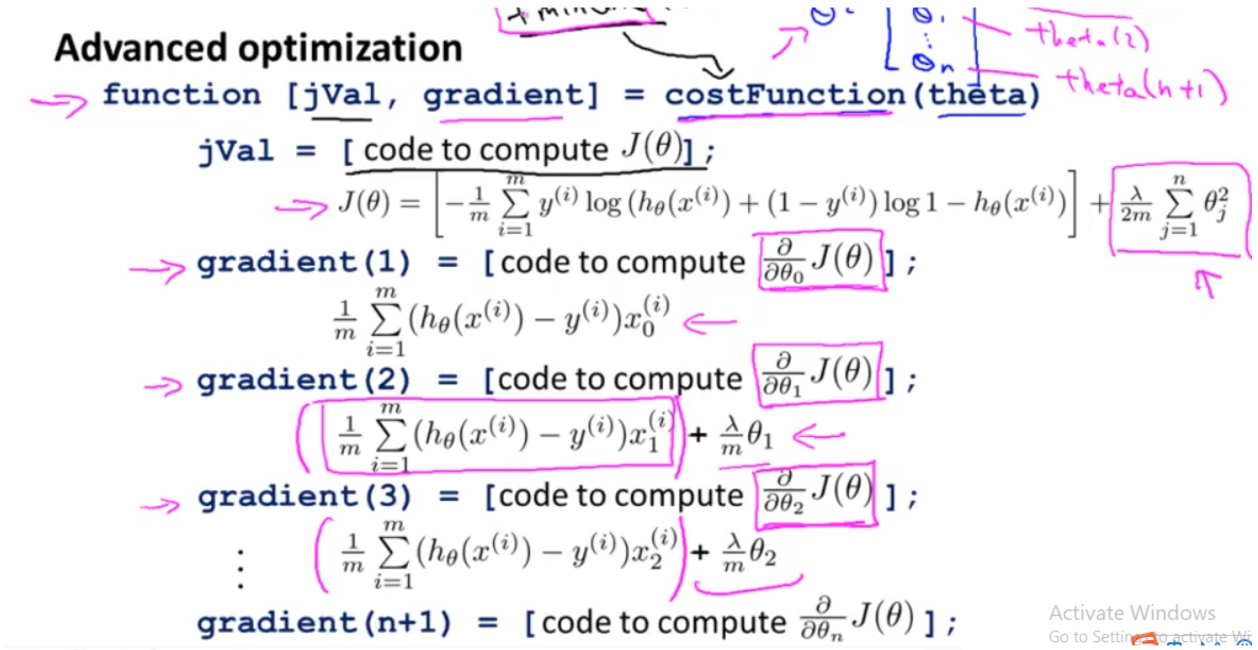
$$\theta = \left(X^T X + \lambda \cdot L\right)^{-1} X^T y$$

$$\text{where } L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

L: (n+1)×(n+1).

if m < n, then $X^T X$ is non-invertible. However, when we add the term λ⋅L, then $X^T X$ + λ⋅L becomes invertible.

Regularized Logistics Regression



Advanced optimization

`function [jVal, gradient] = costFunction(theta)`

$\theta_0 \quad \theta_1, \ldots, \theta_n$ theta(1) theta(2) theta(n+1)

`jVal = [`code to compute $J(\theta)$`];`

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

`gradient(1) = [`code to compute $\boxed{\frac{\partial}{\partial \theta_0} J(\theta)}$`];`

$$\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

`gradient(2) = [`code to compute $\boxed{\frac{\partial}{\partial \theta_1} J(\theta)}$`];`

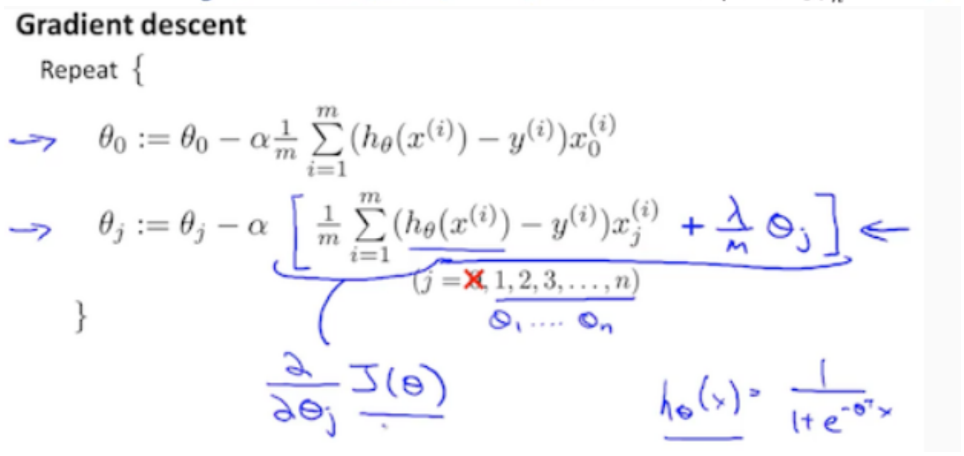
$$\left(\boxed{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}} \right) + \frac{\lambda}{m} \theta_1$$

`gradient(3) = [`code to compute $\boxed{\frac{\partial}{\partial \theta_2} J(\theta)}$`];`

$$\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) + \frac{\lambda}{m} \theta_2$$

⋮

`gradient(n+1) = [`code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$`];`



Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]}_{\frac{\partial}{\partial \theta_j} J(\theta)} \qquad (j = \cancel{0}, 1, 2, 3, \ldots, n)$$

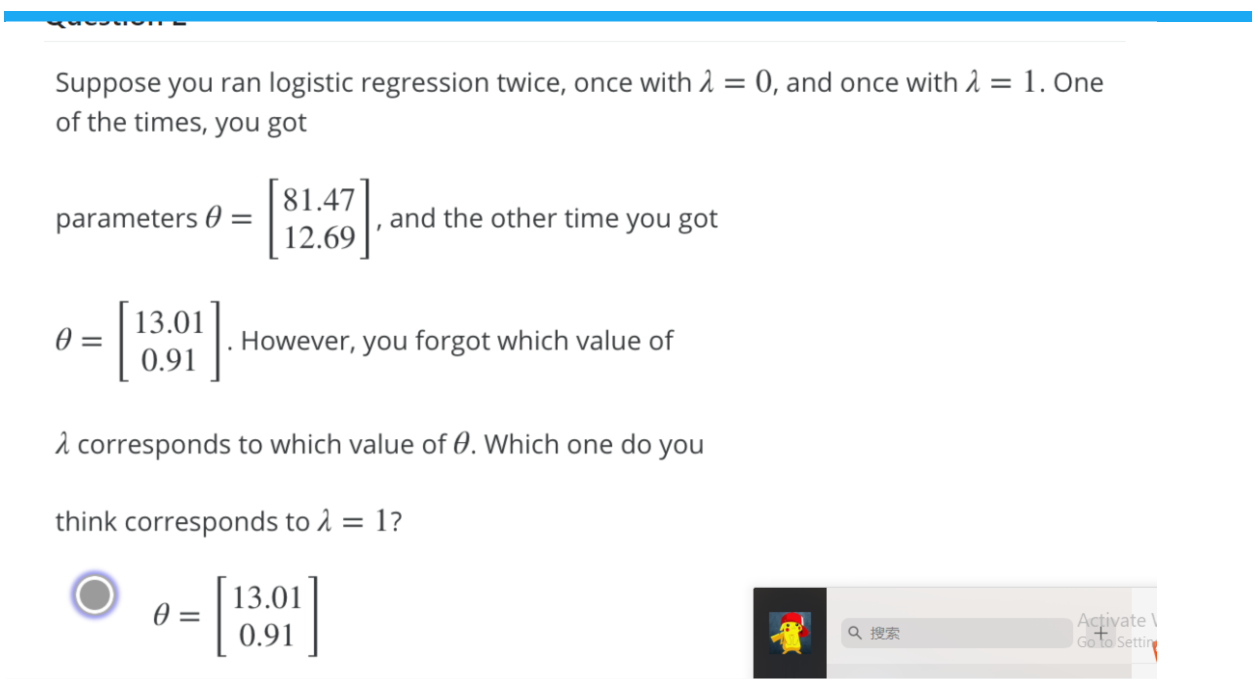
$\theta_1, \ldots, \theta_n$

}

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Quiz

Regularized logistic regression and regularized linear regression are both convex, and thus gradient descent will still converge to the global minimum.



Question 2

Suppose you ran logistic regression twice, once with $\lambda = 0$, and once with $\lambda = 1$. One of the times, you got

parameters $\theta = \begin{bmatrix} 81.47 \\ 12.69 \end{bmatrix}$, and the other time you got

$\theta = \begin{bmatrix} 13.01 \\ 0.91 \end{bmatrix}$. However, you forgot which value of

λ corresponds to which value of θ. Which one do you

think corresponds to $\lambda = 1$?

● $\theta = \begin{bmatrix} 13.01 \\ 0.91 \end{bmatrix}$

Week 4

Motivations

Non-linear Hypotheses

Too many features for the linear regression such as for label training set.

Neurons and the Brain

Neural Networks Origins: Algorithms that try to mimic the brain.

Neural Networks

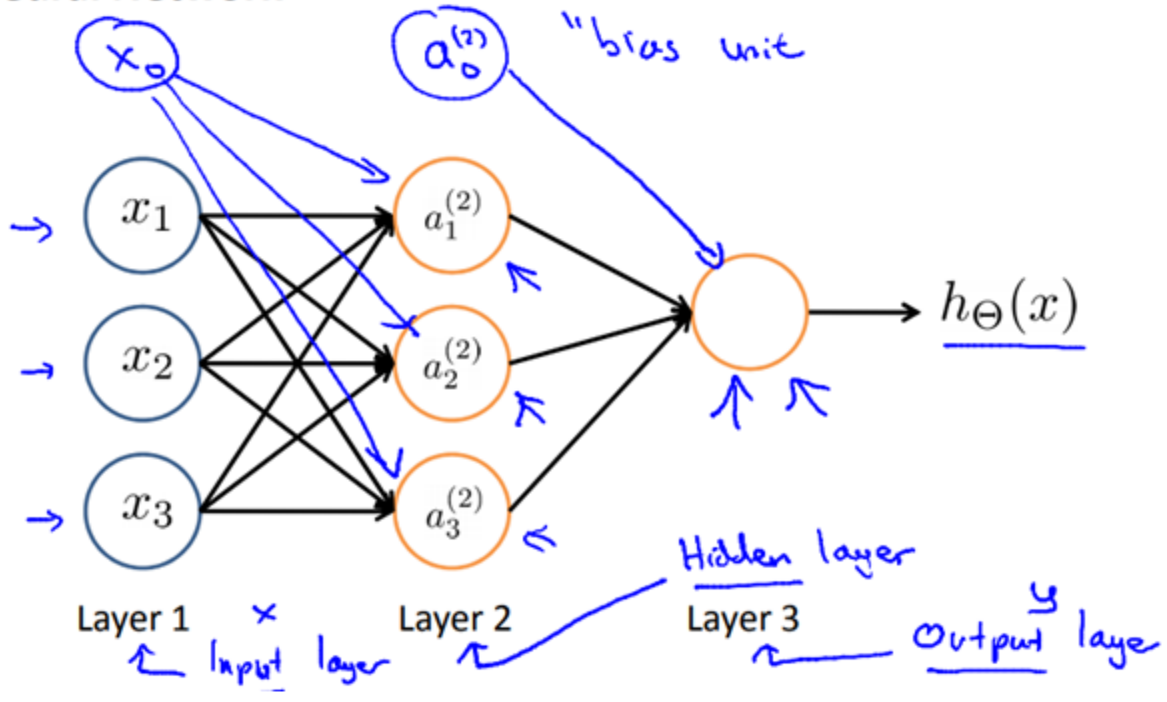
Model Representation I

In this model our x_0 input node is sometimes called the "bias unit." It is always equal to 1. In neural networks, we use the same logistic function as in classification, $\frac{1}{1+e^{-\theta^T x}}$, yet we sometimes call it a sigmoid (logistic) **activation** function. In this situation, our "theta" parameters are sometimes called "weights"

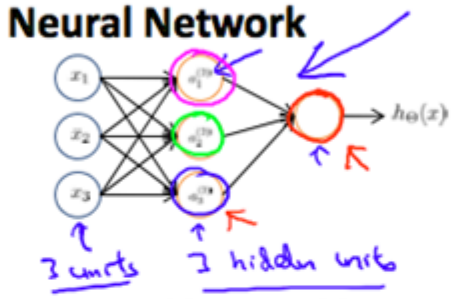
$a_i^{(j)} =$ "activation" of unit i in layer j

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

Neural Network



Neural Network



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

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

$\Theta^{(1)} \in \mathbb{R}^{3\times4}$

$h_\Theta(x)$

$\Theta^{(2)}$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

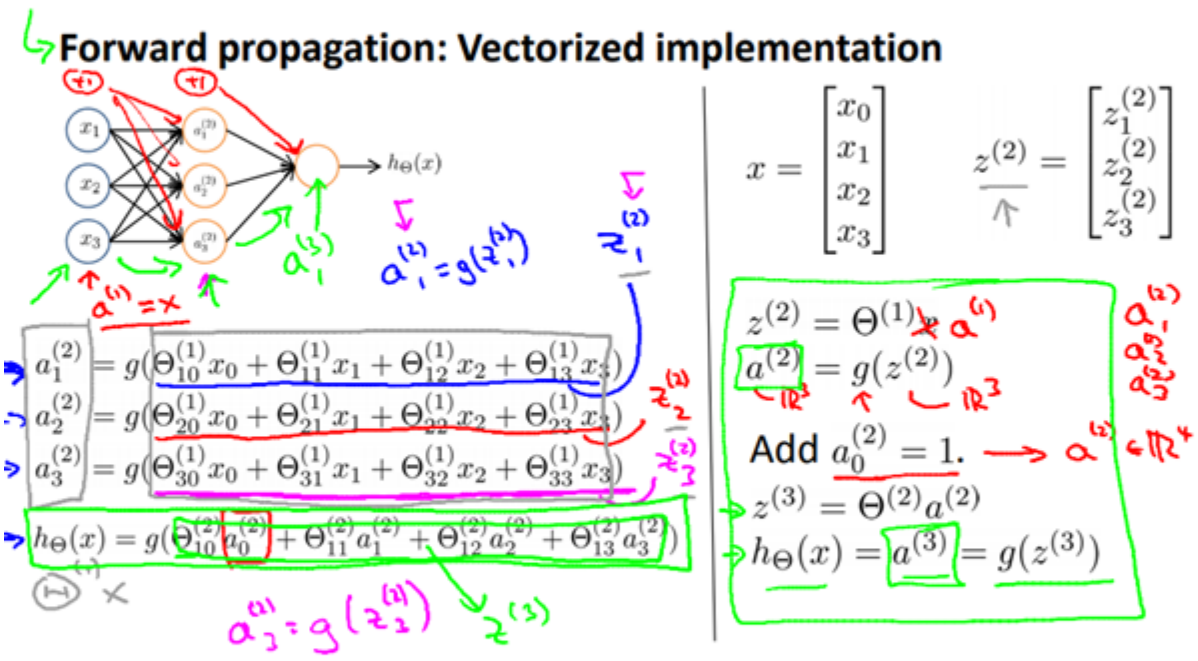
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

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

$S_{j+1} \times (S_j + 1)$

Model Representation II

Forward propagation: vectorized implementation



Forward propagation: Vectorized implementation

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

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)}\cancel{x}\, a^{(1)}$$
$$\boxed{a^{(2)}} = g(z^{(2)})$$
Add $a_0^{(2)} = 1$. $\longrightarrow a^{(2)} \in \mathbb{R}^4$
$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = \boxed{a^{(3)}} = g(z^{(3)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

etting $x = a^{(1)}$, we can rewrite the equation as:

$$z^{(j)} = \Theta^{(j-1)}a^{(j-1)}$$

eg: $z^{(2)} = \Theta^{(1)}a^{(1)};\ a^{(2)} = g(z^{(2)})$

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$

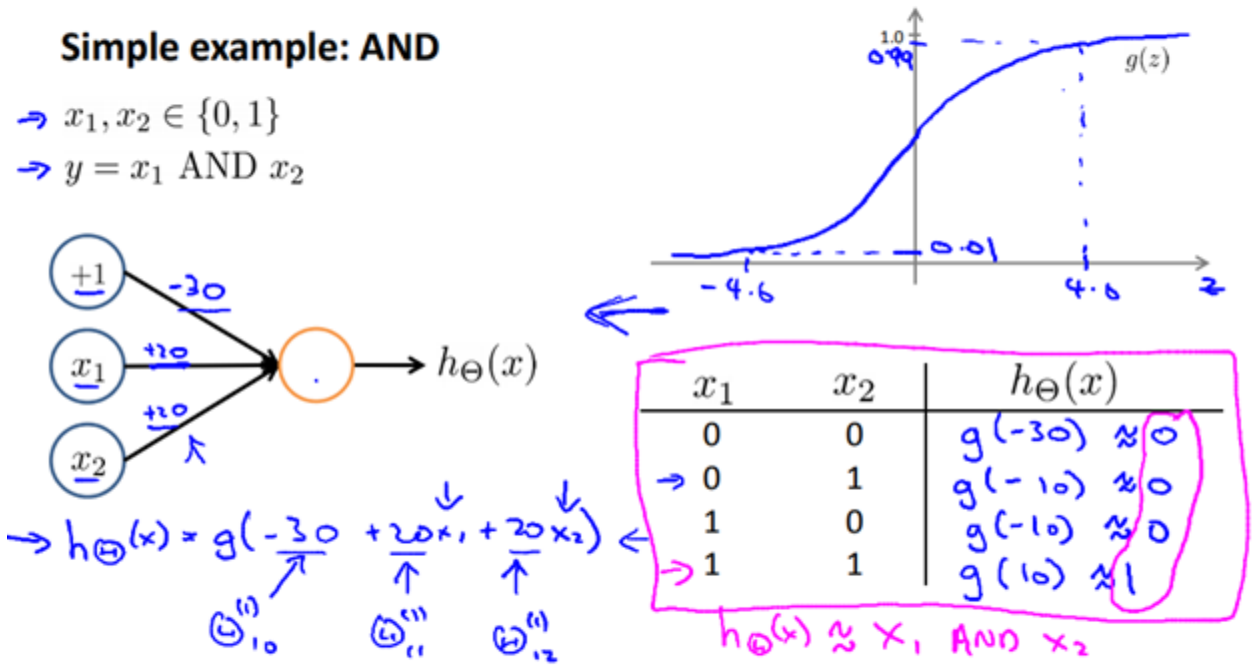
$$a^{(j)} = g(z^{(j)})$$

We can then add a bias unit (equal to 1) to layer j after we have computed $a(j)$. This will be element $a_0^{(j)}$ and will be equal to 1. To compute our final hypothesis, let's first compute another z vector:

$$z^{(j+1)} = \Theta^{(j)}a^{(j)} \qquad h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)})$$

Applications

Examples and Intuitions I



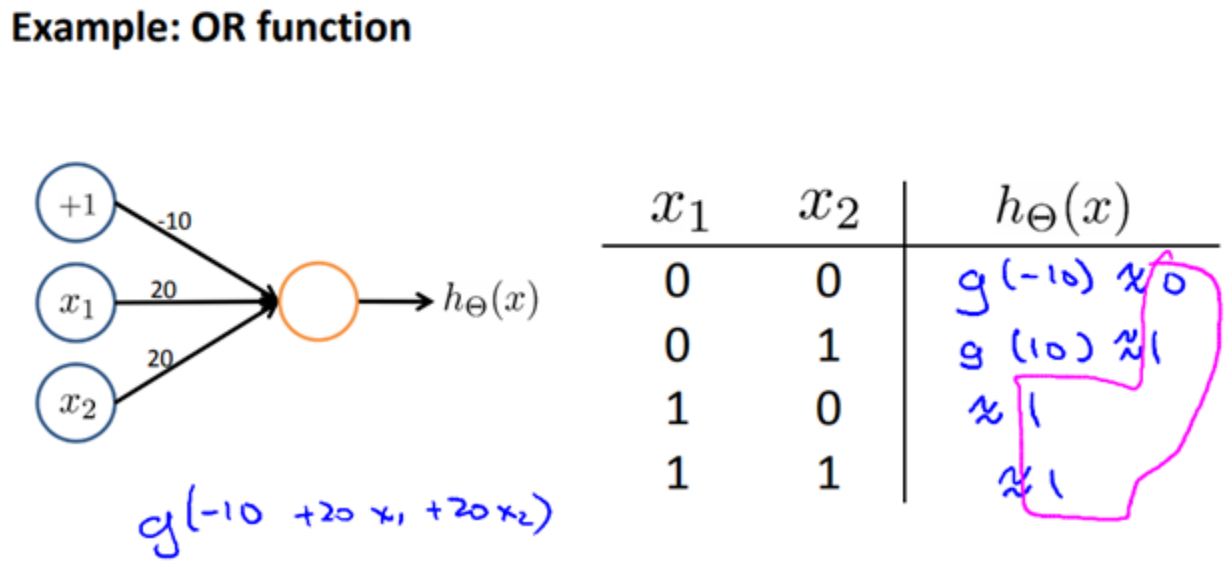
Simple example: AND

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \text{ AND } x_2$

$$h_\Theta(x) = g(\underset{\Theta^{(1)}_{10}}{-30} + \underset{\Theta^{(1)}_{11}}{20}x_1 + \underset{\Theta^{(1)}_{12}}{20}x_2)$$

x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

$h_\Theta(x) \approx x_1 \text{ AND } x_2$



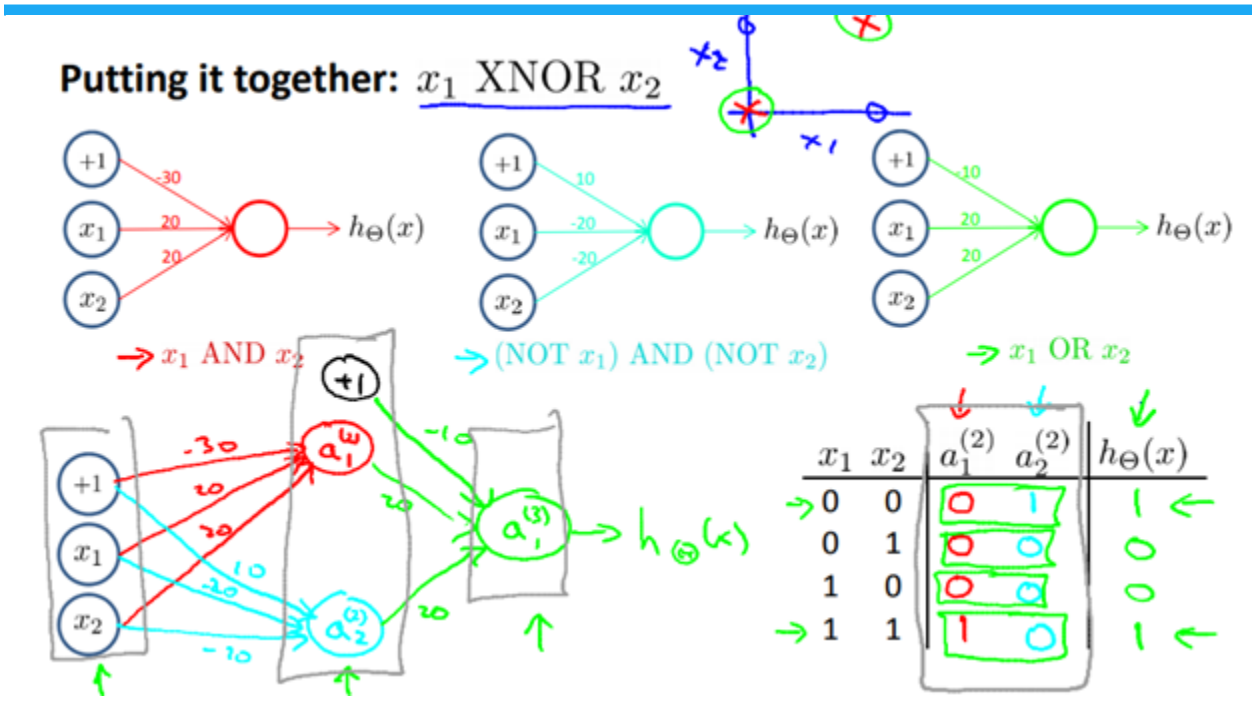
Example: OR function

$$g(-10 + 20x_1 + 20x_2)$$

x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	≈ 1
1	1	≈ 1

Examples and Intuitions II

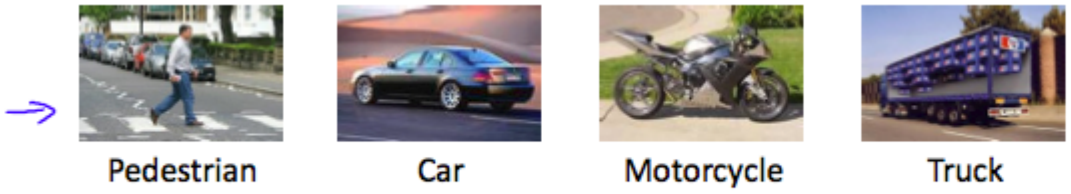
Example: (NOT x_1) AND (NOT x_2)1 if only if $x_1 = x_2 = 0$

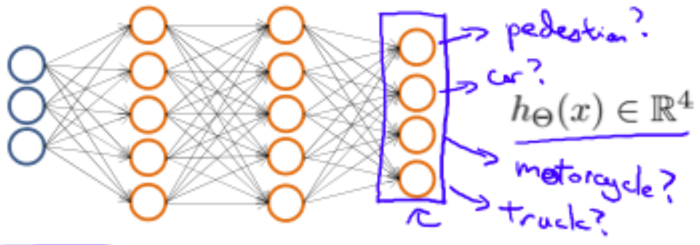


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Multiclass Classification

Multiple output units: One-vs-all.





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

when pedestrian when car when motorcycle

Andrew Ng

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

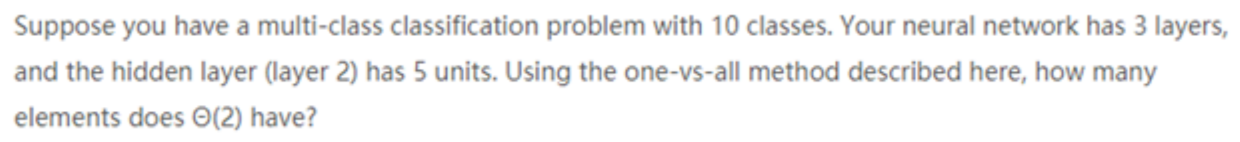
Each $y^{(i)}$ represents a different image correspond
provide us with some new information which lead

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

Our resulting hypothesis for one set of inputs may

$$h_\Theta(x) = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

Quiz



Suppose you have a multi-class classification problem with 10 classes. Your neural network has 3 layers, and the hidden layer (layer 2) has 5 units. Using the one-vs-all method described here, how many elements does Θ(2) have?

Hidden layer has one bias. 60=10*6;