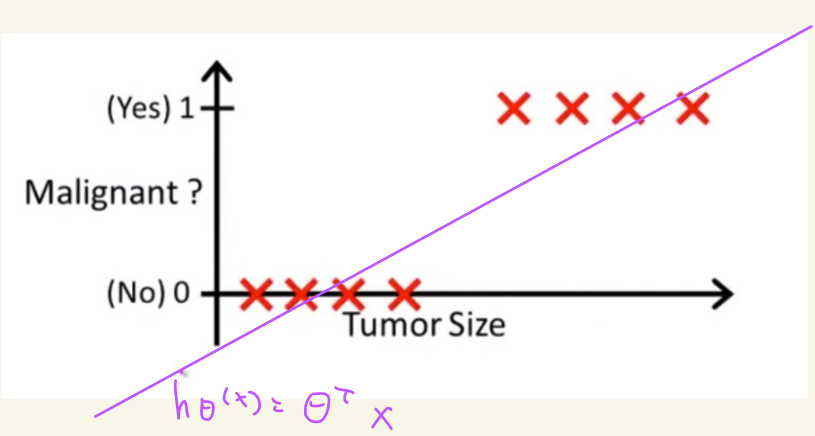
Classification

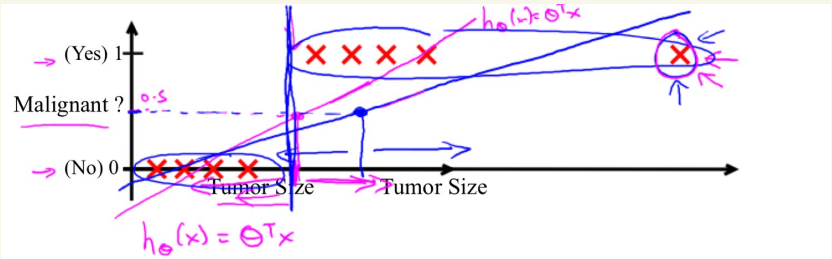
Classification

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

- $y \in \{0, 1\}$
  - 0: "Negative Class" (e.g., benign tumor)
  - 1: "Positive Class" (e.g., malignant tumor)

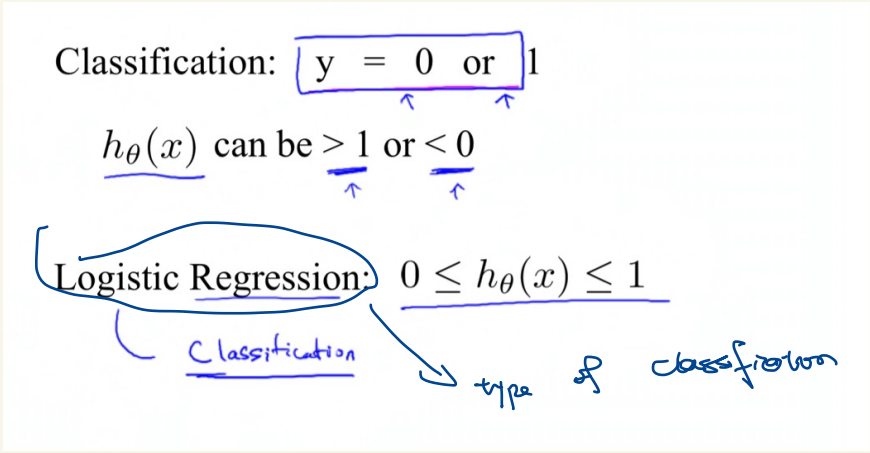
- $y \in \{0, 1, 2, 3\}$





- Threshold classifier output $h_\theta(x)$ at 0.5:
  - If $h_\theta(x) \geq 0.5$, predict "y = 1"

  If $h_\theta(x) < 0.5$, predict "y = 0"



Question

Which of the following statements is true?

- If linear regression doesn't work on a classification task as in the previous example shown in the video, applying feature scaling may help.
- If the training set satisfies $0 \leq y^{(i)} \leq 1$ for every training example $(x^{(i)}, y^{(i)})$, then linear regression's prediction will also satisfy $0 \leq h_\theta(x) \leq 1$ for all values of x.
- If there is a feature x that perfectly predicts y, i.e. if $y = 1$ when $x \geq c$ and $y = 0$ whenever $x < c$ (for some constant c), then linear regression will obtain zero classification error.
- None of the above statements are true.

✓ **Correct**

Classification

To attempt 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.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the **binary classification problem** in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then $x^{(i)}$ may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, $y \in \{0,1\}$. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+." Given $x^{(i)}$, the corresponding $y^{(i)}$ is also called the label for the training example.

Hypothesis Representation

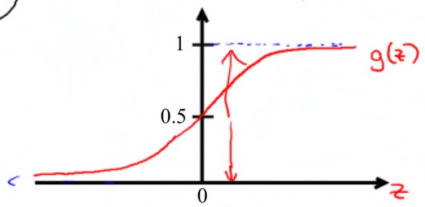
Logistic Regression Model

Want $0 \le h_\theta(x) \le 1$

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

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

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



Sigmoid function

Logistic function

(mean the same)

Parameters θ.

Interpretation of Hypothesis Output $h_\theta(x)$

$h_\theta(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$ → $y=1$ → Prob $y=1$

Tell patient that 70% chance of tumor being malignant

$h_\theta(x) = P(y=1 \mid x; \theta)$ — Prob that $y=1$, given x, parameterized by θ.

$y = 0$ or 1

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

Question

Suppose we want to predict, from data x about a tumor, whether it is malignant ($y = 1$) or benign ($y = 0$). Our logistic regression classifier outputs, for a specific tumor, $h_\theta(x) = P(y = 1|x; \theta) = 0.7$, so we estimate that there is a 70% chance of this tumor being malignant. What should be our estimate for $P(y = 0|x; \theta)$, the probability the tumor is benign?

- (●) $P(y = 0|x; \theta) = 0.3$
- ( ) $P(y = 0|x; \theta) = 0.7$
- ( ) $P(y = 0|x; \theta) = 0.7^2$
- ( ) $P(y = 0|x; \theta) = 0.3 \times 0.7$

✓ **Correct**

Hypothesis Representation

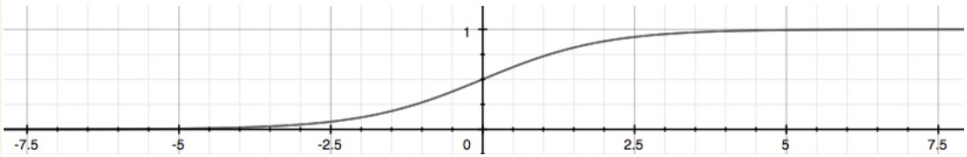
We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given x. However, it is easy to construct examples where this method performs very poorly. Intuitively, it also doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that y ∈ {0, 1}. To fix this, let's change the form for our hypotheses $h_\theta(x)$ to satisfy $0 \leq h_\theta(x) \leq 1$. This is accomplished by plugging $\theta^T x$ into the Logistic Function.

Our new form uses the "Sigmoid Function," also called the "Logistic Function":

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

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

The following image shows us what the sigmoid function looks like:



The function g(z), shown here, maps any real number to the (0, 1) interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.

$h_\theta(x)$ 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. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

$$h_\theta(x) = P(y = 1|x; \theta) = 1 - P(y = 0|x; \theta)$$
$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$

Decision Boundary

$x=5$. $5 \cdot x_1 \geq 0$

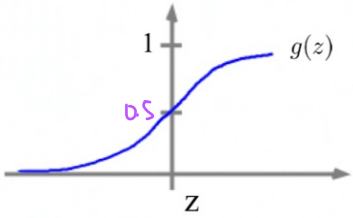
$-x_1 \geq -5$

$x_1 \leq 5$

Logistic regression

$$h_\theta(x) = g(\theta^T x) = P(y=1 \mid x; \theta)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



suppose predict "$y=1$" if $h_\theta(x) \geq 0.5$

$g(z) \geq 0.5$ when $z \geq 0$.

since $h_\theta(x) = g(\theta^T x) \geq 0$ whenever $\theta^T x \geq 0$.

z

- Predict "$y=0$" if $h_\theta(x) < 0.5$.

$g(z) < 0.5$

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

$\therefore y = 0 \Rightarrow \theta^T x < 0$

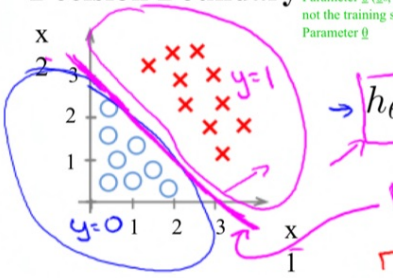
Parameter vector.

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$

Decision Boundary

How to fit (find) Parameter θ

Parameter θ (θ_0, θ_1, θ_2) defines the decision boundary not the training set. Training set may be used to find the Parameter θ



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

-3 1 1

Decision boundary

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

$x_1 + x_2 \geq 3$

x_1, x_2

$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

$x_1 + x_2 < 3$

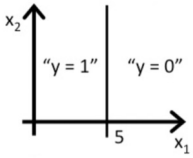
$y = 0$

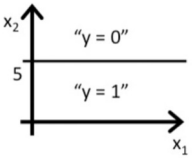
even tk away the data, the line still separates the region.

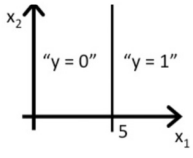
property of hypothesis including $\theta_0, \theta_1, \theta_2$

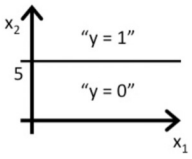
Question

Consider logistic regression with two features x_1 and x_2. Suppose $\theta_0 = 5, \theta_1 = -1, \theta_2 = 0$, so that $h_\theta(x) = g(5 - x_1)$. Which of these shows the decision boundary of $h_\theta(x)$?



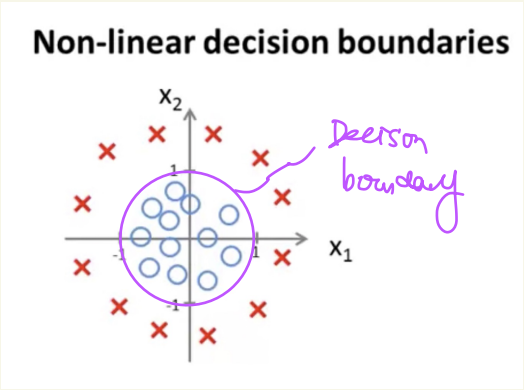






Correct

Predict Y = 0 if x_1 is greater than 5.



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

$\theta_0 = -1$, $\theta_1 = 0$, $\theta_2 = 0$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$.

Training set doesn't decide decision boundary. parameter decides.

fit data

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \cdots)$$

Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$$g(z) \geq 0.5$$
$$when\ z \geq 0$$

Remember.

$$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$$
$$z \rightarrow \infty, e^{-\infty} \rightarrow 0 \Rightarrow g(z) = 1$$
$$z \rightarrow -\infty, e^{\infty} \rightarrow \infty \Rightarrow g(z) = 0$$

So if our input to g is $\theta^T X$, then that means:

$$h_\theta(x) = g(\theta^T x) \geq 0.5$$
$$when\ \theta^T x \geq 0$$

From these statements we can now say:

$$\theta^T x \geq 0 \Rightarrow y = 1$$
$$\theta^T x < 0 \Rightarrow y = 0$$

The **decision boundary** is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.

Example:

$$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$
$$y = 1\ if\ 5 + (-1)x_1 + 0x_2 \geq 0$$
$$5 - x_1 \geq 0$$
$$-x_1 \geq -5$$
$$x_1 \leq 5$$

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes y = 1, while everything to the right denotes y = 0.

Again, the input to the sigmoid function g(z) (e.g. $\theta^T X$) doesn't need to be linear, and could be a function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data.

Cost Function.

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

m examples (training examples.)

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad x_0 = 1, y \in \{0, 1\}$$

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

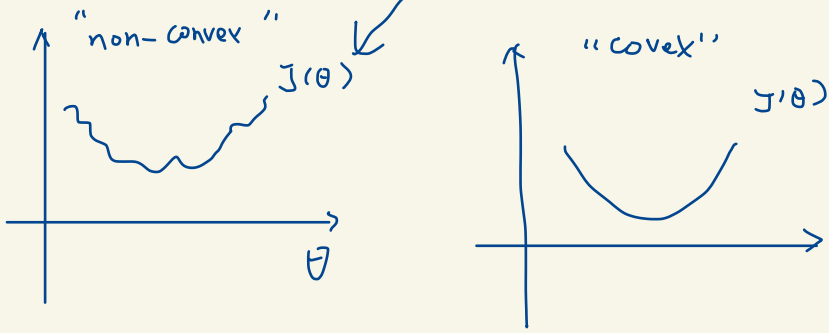
How to choose parameters θ ?

- Linear regression:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2}_{\text{Cost}(h_\theta(x^{(i)}), y)}.$$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

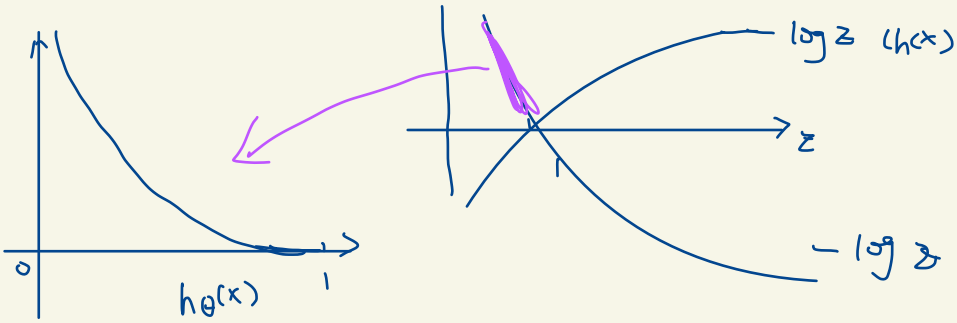
↳ if use as (logistic regression) may not converge. to the local minimum.



- Logistic regression cost func.

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

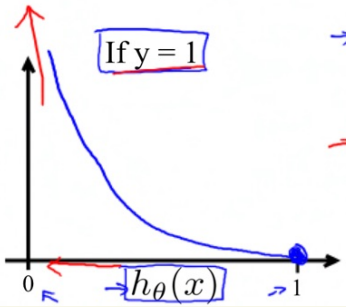
if $y = 1$



Logistic regression cost function

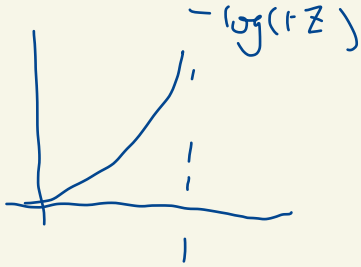
Different Cost Function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



→ Cost = 0 if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
$Cost \to \infty$

→ Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.



Question

In logistic regression, the cost function for our hypothesis outputting (predicting) $h_\theta(x)$ on a training example that has label $y \in \{0, 1\}$ is:

$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log h_\theta(x) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

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

- [x] If $h_\theta(x) = y$, then $\text{cost}(h_\theta(x), y) = 0$ (for $y = 0$ and $y = 1$).

✓ **Correct**

- [x] If $y = 0$, then $\text{cost}(h_\theta(x), y) \to \infty$ as $h_\theta(x) \to 1$.

✓ **Correct**

- [ ] If $y = 0$, then $\text{cost}(h_\theta(x), y) \to \infty$ as $h_\theta(x) \to 0$.
- [x] Regardless of whether $y = 0$ or $y = 1$, if $h_\theta(x) = 0.5$, then $\text{cost}(h_\theta(x), y) > 0$.

✓ **Correct**

Cost Function

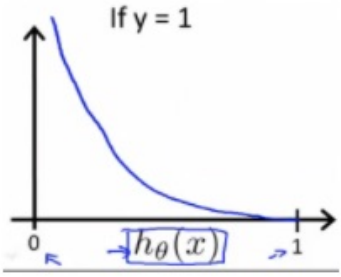
We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

Instead, our cost function for logistic regression looks like:

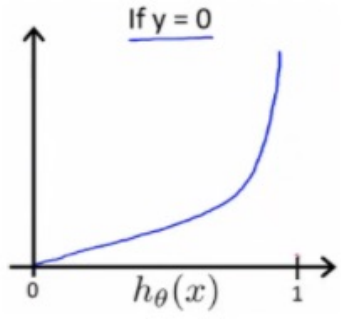
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

When y = 1, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



Similarly, when y = 0, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



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

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

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

Simplified Cost function and Gradient Descent

- Logistic regression cost function.

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

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

→ put into one eq

Note: $y=0$ or 1 always

Same.

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

- Suppose

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

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

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

To fit parameters θ:

$$\min_\theta J(\theta)$$

To make a prediction given new x:

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

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

Repeat {

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

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

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

after simplifying get the same eq as Non logistic.

Question

Suppose you are running gradient descent to fit a logistic regression model with parameter $\theta \in \mathbb{R}^{n+1}$. Which of the following is a reasonable way to make sure the learning rate α is set properly and that gradient descent is running correctly?

- ○ Plot $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ as a function of the number of iterations (i.e. the horizontal axis is the iteration number) and make sure $J(\theta)$ is decreasing on every iteration.
- ◉ Plot $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$ as a function of the number of iterations and make sure $J(\theta)$ is decreasing on every iteration.
- ○ Plot $J(\theta)$ as a function of θ and make sure it is decreasing on every iteration.
- ○ Plot $J(\theta)$ as a function of θ and make sure it is convex.

✓ **Correct**

Question

One iteration of gradient descent simultaneously performs these updates:

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

$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$

$\vdots$

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

We would like a vectorized implementation of the form $\theta := \theta - \alpha\delta$ (for some vector $\delta \in \mathbb{R}^{n+1}$).

What should the vectorized implementation be?

- ◉ $\theta := \theta - \alpha\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}]$
- ○ $\theta := \theta - \alpha\frac{1}{m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})] \cdot x^{(i)}$
- ○ $\theta := \theta - \alpha\frac{1}{m}x^{(i)}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})]$
- ○ All of the above are correct implementations.

✓ **Correct**

Simplified Cost Function and Gradient Descent

Note: [6:53 - the gradient descent equation should have a 1/m factor]

We can compress our cost function's two conditional cases into one case:

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

Notice that when y is equal to 1, then the second term $(1-y)\log(1-h_\theta(x))$ will be zero and will not affect the result. If y is equal to 0, then the first term $-y\log(h_\theta(x))$ will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

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

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:

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

Notice that this algorithm is identical to the one we used in linear regression. We still have to simultaneously update all values in theta.

A vectorized implementation is:

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

Advanced Optimization.

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

=> - advantages
- No need to manually pick α
- Often faster than gradient descent

- disadvantages:
- More complex.

Example:

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $\min_\theta J(\theta)$

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

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

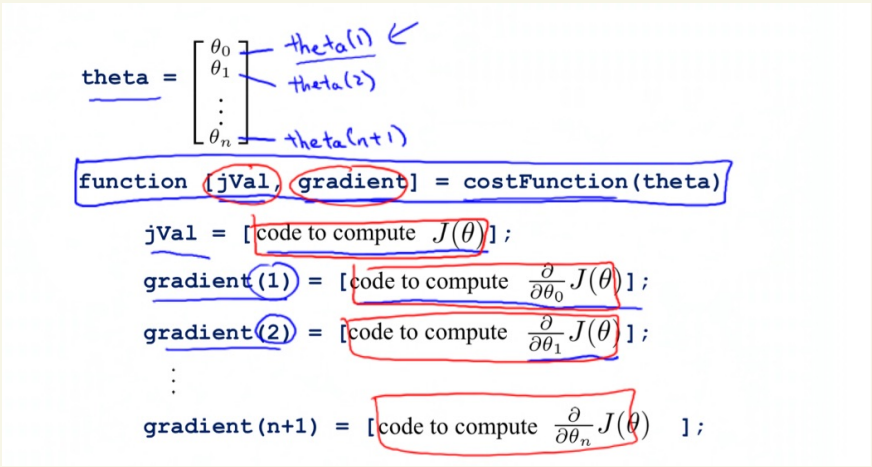
$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

```
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);
```

Set parameter for optimal alg.

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

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



Question

Suppose you want to use an advanced optimization algorithm to minimize the cost function for logistic regression with parameters θ_0 and θ_1. You write the following code:

```
function [jVal, gradient] = costFunction(theta)
  jVal = % code to compute J(theta)
  gradient(1) = CODE#1 % derivative for theta_0
  gradient(2) = CODE#2 % derivative for theta_1
```

What should CODE#1 and CODE#2 above compute?

- CODE#1 and CODE#2 should compute $J(\theta)$.
- CODE#1 should be theta(1) and CODE#2 should be theta(2).
- (selected) CODE#1 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)})-y^{(i)})\cdot x_0^{(i)}](=\frac{\partial}{\partial\theta_0}J(\theta))$, and CODE#2 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)})-y^{(i)})\cdot x_1^{(i)}](=\frac{\partial}{\partial\theta_1}J(\theta))$.
- None of the above.

✓ Correct

Advanced Optimization

Note: [7:35 - '100' should be 100 instead. The value provided should be an integer and not a character string.]

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent. We suggest that you should not write these more sophisticated algorithms yourself (unless you are an expert in numerical computing) but use the libraries instead, as they're already tested and highly optimized. Octave provides them.

We first need to provide a function that evaluates the following two functions for a given input value θ:

$$J(\theta)$$
$$\frac{\partial}{\partial\theta_j} J(\theta)$$

We can write a single function that returns both of these:

```
function [jVal, gradient] = costFunction(theta)
  jVal = [...code to compute J(theta)...];
  gradient = [...code to compute derivative of J(theta)...];
end
```

Then we can use octave's "fminunc()" optimization algorithm along with the "optimset()" function that creates an object containing the options we want to send to "fminunc()". (Note: the value for MaxIter should be an integer, not a character string - errata in the video at 7:30)

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

```

We give to the function "fminunc()" our cost function, our initial vector of theta values, and the "options" object that we created beforehand.

Multiclass Classification

- One-vs-all

Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

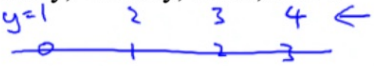
$y=1$ $y=2$ $y=3$ $y=4$

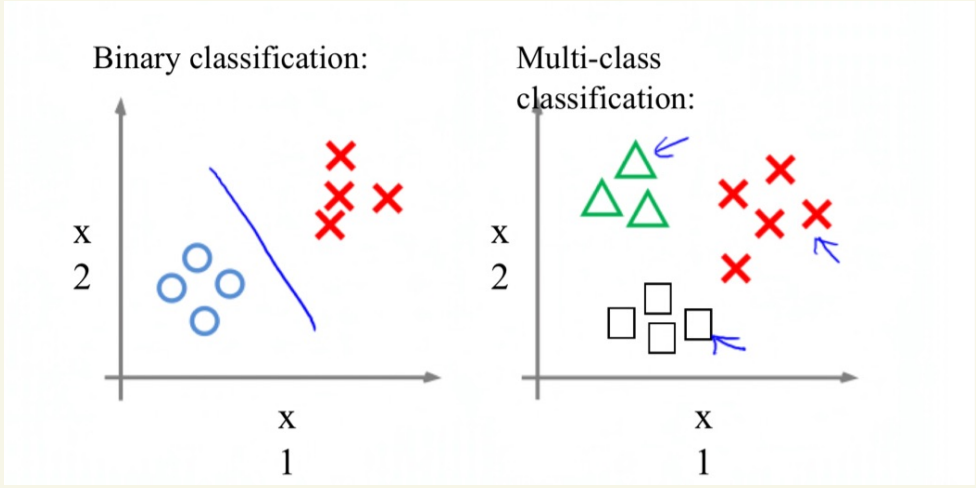
Medical diagrams: Not ill, Cold, Flu

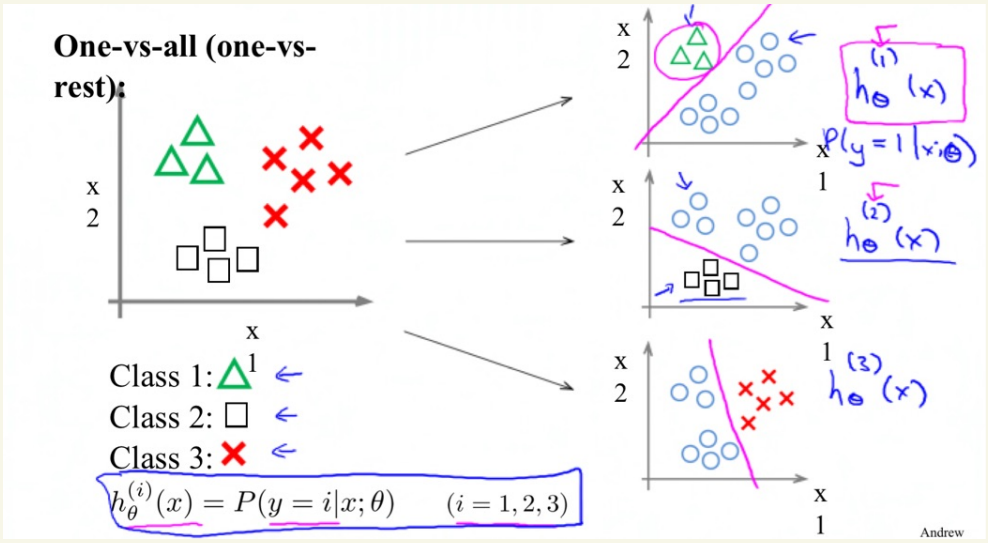
$y=1$ 2 3

Weather: Sunny, Cloudy, Rain, Snow

$y=1$ 2 3 4







One-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Question

Suppose you have a multi-class classification problem with k classes (so $y \in \{1, 2, \ldots, k\}$). Using the 1-vs.-all method, how many different logistic regression classifiers will you end up training?

- ○ $k - 1$
- ◉ k
- ○ $k + 1$
- ○ Approximately $\log_2(k)$

✓ Correct

Multiclass Classification: One-vs-all

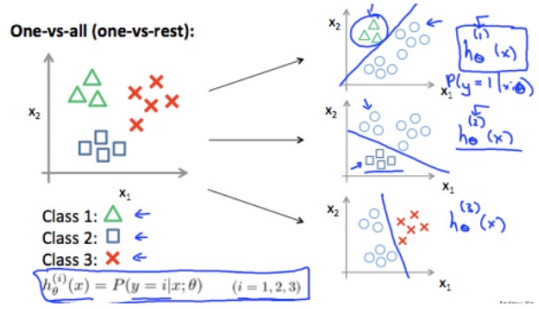
Now we will approach the classification of data when we have more than two categories. Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.

Since y = {0,1...n}, we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

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

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

The following image shows how one could classify 3 classes:



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 $h_\theta(x)$