***Classification and Representation***

**I. CLASSIFICATION**

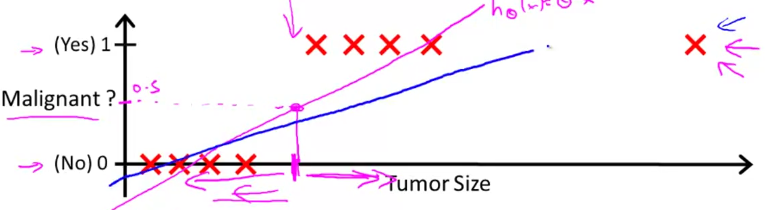
* **Binary Classification**  🡪 variable you want to predict, y, is valued via **logistic regression**
* Ex: spam or not, online transactions as fraudulent/not (someone using a stolen credit card or user password), tumors as malignant or benign.
* In all of these problems, the variable we're trying to predict takes on TWO values 🡪 0/1, spam/not spam, fraudulent/not fraudulent, malignant/benign.

 🡪 0 = the **negative class** (benign), 1 = **positive class** (malignant)

* Assignment of the 2 classes is somewhat arbitrary
* Intuition = negative class conveys “absence” of something, positive class conveys presence of
* Ex: Training set for classifying tumor as malignant or benign where malignancy takes on 2 values, 0/1
* 1 thing we could do, given this training set is apply the *linear* regression algorithm that we already know to this data set + try to fit a straight line to the data to get a hypothesis, hθ(x)



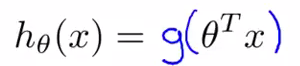
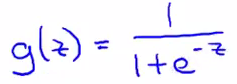
* To make predictions, we can try to **threshold** the classifier outputs at y = 0.5 🡪 if hθ(x) outputs a value >= 0.5, say y = 1, if < 0.5 say y = 0.



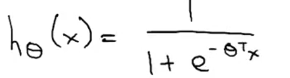
* For this example, it looks like linear regression is actually doing something reasonable at first, even though this is classification
* But w/ that training example very out to the right (outlier), linear regression actually gives a worse straight line fit to the data (blue line) + there a worse hypothesis.
* So, applying linear regression to a classification problem often isn't a great idea.
* Above, linear regression was just getting lucky + got us a hypothesis that worked well
* For classification we know that y = 0/1, but w/ linear regression, hθ(x) can output values much larger than 1 or less than 0, even if all training examples have labels y = 0 or 1.
* **LOGISTIC regression** has the property that the output/predictions are *always* between 0 and 1

**II. HYPOTHESIS REPRESENTATION**

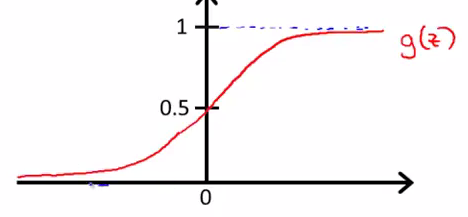
* We want our classifier to *output values* between 0 and 1, so we want to come up with a hypothesis that satisfies this property w/ its *predictions* being between 0 and 1.
* When using linear regression, we had **hθ(x) = θ(t)\*x.**
* For logistic regression, we modify this a little bit + make:

 where 

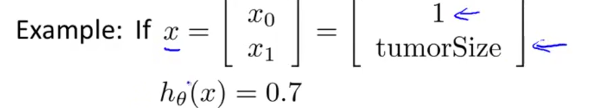
* g(z) = **the** **sigmoid function/the logistic function**
* Put these 2 together:



* Plotted, the sigmoid function, g(z), starts off near 0 + rises until it crosses y = 0.5 at the origin + then flattens out again near 1 = horizontal **asymptotes** at 1 + 0



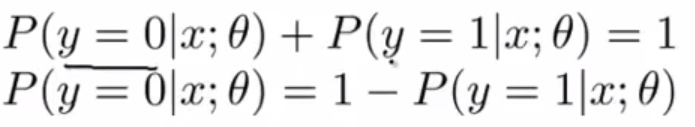
* *So b/c g(z) values are between 0 and 1, we also have that hθ(x) must be between 0 and 1.*
* Finally, given this hypothesis representation + a training set, we need to fit the parameters θ to our data
* To interpret the output of hθ(x) 🡪 when hθ(x) outputs some number, treat it as the **estimated probability** that y = 1 on a new input, x.



* A patient has a 70% chance, or a 0.7 chance of being malignant, due to their tumor size
* *More formally, h(x) outputs the probability* ***P*** *of y = 1 given x, which is parameterized by θ.*

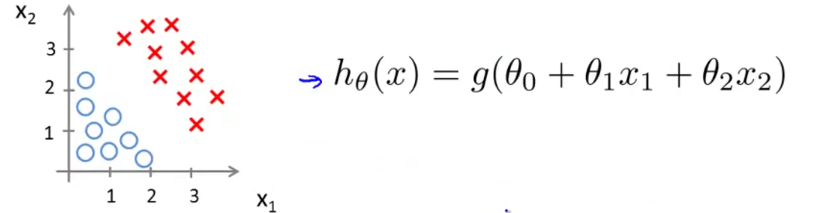


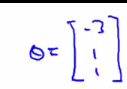
* Since this is a classification task, we know y must be 0 or 1, so given hθ(x), we can therefore compute the probability of y = 0
* As well y = 1 via [ 1 – hθ(x) ] b/c probability of y = 0 + probability of y = 1 must be = 1.

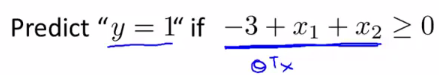


**III. DECISION BOUNDARY**

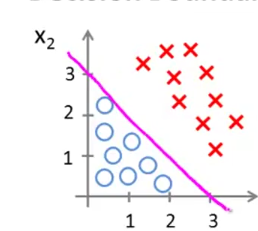
* We can assume that if hθ(x) >= 0.5, its more likely to be y = 1 than y equals 0, so we predict y = 1, and vice versa for hθ(x) < 0.5
* Looking at the sigmoid function, g(z) >= 0.5 whenever z >= 0, where z = θ(t)\*x
* Since hθ(x) for logistic regression is therefore going to be >= 0.5 whenever θ(t)\*x (i.e. z) >= 0, *a hypothesis* hθ(x) *predicts y = 1 whenever θ(t)\*x >= 0*
* By similar argument, hθ(x) < 0.5 whenever g(z) < 0.5 because θ(t)\*x being < 0 causes g(z) to take on values < 0.5
* So when g(z) < 0.5, a hypothesis will predict y = 0 (or will predict y = 0 whenever θ(t)\*x < 0)
* To summarize, if we decide to predict y = 1 or y = 0 depending on whether an estimated probability hθ(x) >= 0.5 or < 0.5, it’s the same as saying we predict y = 1 whenever θ(t)\*x >= 0 and predict y = 0 whenever θ(t)\*x < 0



* Suppose we have a training set + hθ(x) above + suppose that via a procedure-to-be-specified, we end up choose parameter value θ0 = -3, θ1 = 1, θ2 = 1.
* Parameter vector = 3\*1 = 
* Given this choice of hθ(x) parameters, let’s try to figure out where a hypothesis hθ(x) would end up predicting y = 1 or y = 0.
* We know the probability y = 1 is more likely (>= 0.5) when θ(t)\*x > 0
* W/ our θ values, this is: *y = 1 if [ -3 + x1 + x2 ] >= 0 🡺 x1 + x2 >= 3*



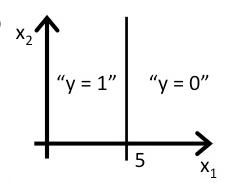
* x1 + x2 = 3 defines the equation of a straight line which passes through 3 on the x1 + the x2 axis.



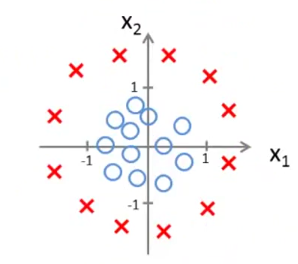
* So the part of the x1-x2 plane that corresponds to when x1 + x2 >= 3 is everything to the *upper right portion of the magenta line* = the region where our hypothesis hθ(x) predicts y = 1
* The region where x1 + x2 < 3 corresponds to the region below the line = where our hypothesis hθ(x) predicts y = 0.
* This magenta line = **the decision boundary.**
* x1 + x2 = 3 corresponds to the region where hθ(x) = 0.5 exactly
* The straight decision boundary line separates the region where hθ(x) predicts y = 1 from the region where it predicts y = 0.
* The decision boundary is a property of *the hypothesis* hθ(x), including the parameters θ0, θ1, θ2, *NOT of the data set.*
* Later we will fit the parameters + use the training set data to determine the value of the parameters



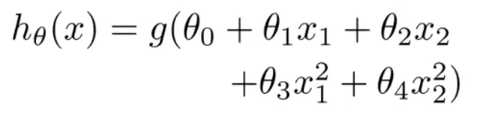
* So we have 5 – x1 >= 0 for y = 1 🡪 y = 1 is x1 > 5



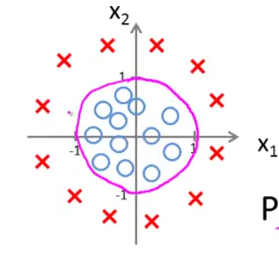
* More complex example 🡪 non-linear



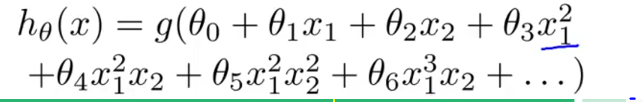
* How could we get logistic regression to fit this kind of data?
* Could add extra higher-order polynomial terms to our features, like in **polynomial linear** **regression**
* Add 2 extra features, x1^2 and x2^2, to the features = now have 5 parameters, θ0 through θ4.



* Assume via a future procedure to be specified, we choose θ0 = -1, θ1 = 0, θ2 = 0, θ3 = 1 + θ4 = one, so our parameter vector transposed = 5\*1 🡪 [-1, 0, 0, 1, 1] 🡪 ignore x1 + x2
* So hθ(x) predicts y = 1 when -1 + x1^2 + x2^2 >= 0 (whenever θ(t)\*the features >= 0)
* So hθ(x) predicts y = 1 when (x1^2 + x2^2) >= 1
* hθ(x) is the equation for circle of radius 1 centered around the origin = creates a decision boundary



* Everything outside the circle 🡪 predict y = 1 Everything inside the circle 🡪 y = 0
* So by adding more complex polynomial terms to our sets of features, we can get more complex decision boundaries that don't just try to separate positive and negative examples via a straight line
* \*\*\*\*\*Once again, the decision boundary is a *property*, NOT of the *training 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 θ.*
* But, once you have the parameters θ, ***hθ(x)*** is what defines the decisions boundary.
* Can we come up w/ even more complex decision boundaries w/ even higher-order polynomial terms?



* Now we can find decision boundaries that may be an ellipse or some funny shape

