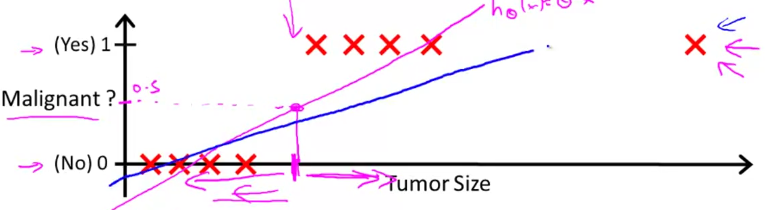
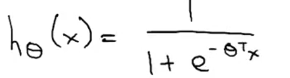
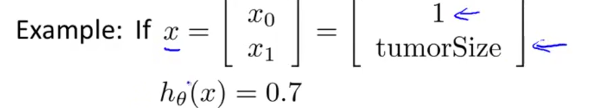
* **Binary Classification**  🡪 the variable that you want to predict, y, is valued via **logistic regression**
* Ex: email spam classification, classifying online transactions as fraudulent/not (someone is using a stolen credit card or stolen user password), classifying tumors as malignant or benign.
* 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
* **Multiclass classification problem** **🡪** values of 1-4, etc.
* 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 = 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, h(x)
* 
* To make predictions, try to **threshold** the classifier outputs at 0.5 🡪 if hypothesis 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, even though this is classification
* But now change the problem a bit w/ 1 training example very out to the right (outlier + if you run linear regression now, we instead get a worse straight line fit to the data + a worse hypothesis.
* 
* So, applying linear regression to a classification problem often isn't a great idea.
* In the 1st example, linear regression was just getting lucky + got us a hypothesis that worked well
* For classification we know that y = 0/1, but if using linear regression, h(x) can output values much larger than 1 or less than 0, even if all training examples have labels y = 0 or 1.
* Even though we know the labels should be 0/1, the algorithm can output values much larger than 1 or much smaller than0
* **Logistic regression** has the property that the output/predictions are always between zero and one
* Want our classifier to output values between 0 and 1, so we want a hypothesis that satisfies this property w/ predictions between 0 and 1.
* When using linear regression, h(x) = theta(transpose)\*x.
* For logistic regression, we modify this a little bit + make h(x) = g(theta(t)\*x), + define g as:
* G(z) = 1 / (1 + E^-z) = the **sigmoid function/the logistic function**
* Put these 2 together 🡪 h(x) = 1 / (1 – E^-(theta(t)\*x))



* Plotted, the sigmoid function, g(z), starts off near 0 + rises until it crosses 0.5 at the origin + then flattens out again near 1 = **asymptotes** at one + zero
* So b/c g(z) values are between zero and one, we also have that h(x) must be between zero and one.
* Finally, given this hypothesis representation + given a training set, we need to fit the parameters theta to our data.
* To interpret, when h(x) outputs some number, treat it as the **estimated probability** that y = one on a new input, x.



* Patient has a 70% chance, or a 0.7 chance of being malignant.
* *More formally, h(x) outputs the probability* ***P*** *of y = 1 given x, parameterized by theta.*



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

