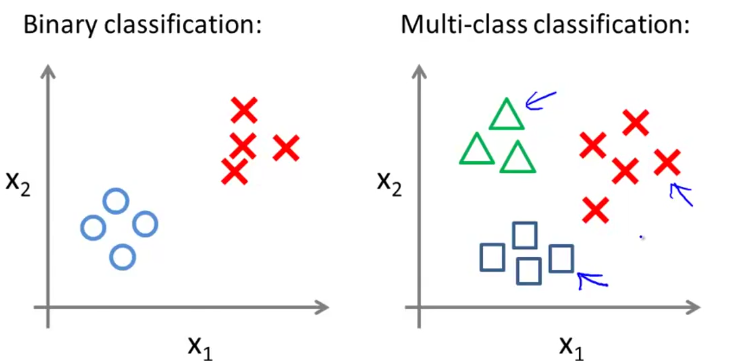
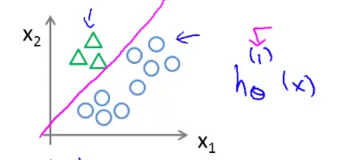
***Multiclass Classification***

**I. ONE VS. ALL CLASSIFICATION**

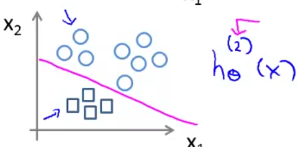
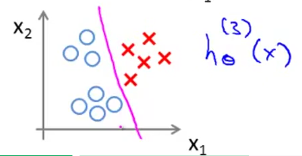
* Logistic regression can work for **multiclass classification** problems as well, such as w/ an algorithm called **one-versus-all classification.**
* Multiclass classification problems:
* Learning algorithm to automatically tag emails (work, friends, family, etc.)
* 4 classes to which we might assign emails 🡪 classes y = 1, y = 2, y = 3, y = 4
* For medical diagnosis, if a patient comes in w/ a stuffy nose, possible diagnosis could be they're not ill (y = 1), have a cold (y = 2), or have a flu (y = 3)
* Classify weather as sunny, cloudy, rainy, or snow,
* In all of these examples, y can take on > 2 values
* Whereas for a binary classification problem, our data sets looks like the left, for a multi-class classification problem our data sets may look like the right



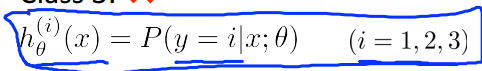
* The question is, given the data set w/ 3 classes, how do we get a learning algorithm to work here?
* We already know how to do binary classification using a regression, maybe fit a straight line to set for positive + negative classes
* Using **one-vs-all/one-vs-rest classification**, we can then make this work w/ multi-class classification as well
* For 3 classes, take the training set + turn it into 3 *separate BINARY* classification problems

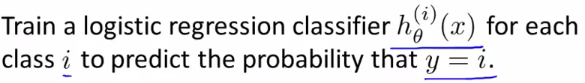
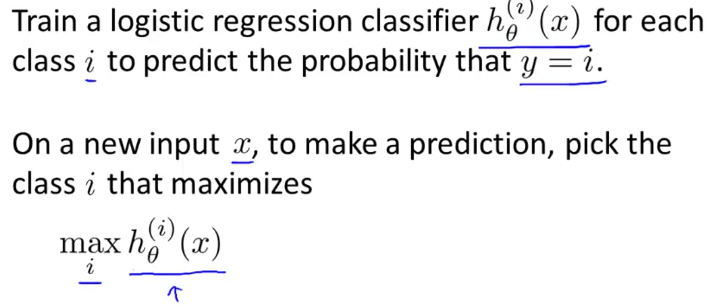


* For class 1 (triangle), essentially we create a new fake training set where classes 2 + 3 get assigned to the negative class + class 1 is assigned to the positive class + then we fit a **classifier**, hθ(x)1
* Think of triangles being assigned the value = 1 and circles (+ squares) assigned the value = 0.
* Then we’d train a standard logistic regression classifier to give us a position boundary
* Then we do the same thing for class 2 and 3 + fit new classifiers to them
* i.e. assign squares as class 1, then circles as class 1, and have the others be in class 2

* We've fit 3 classifiers in trying to estimate the probability y = class i, given x, + parametrized by θ



* In the 1st, the classifier was learning to recognize triangles/thinking of triangles as a positive class
* hθ(x)1 is trying to estimate the probability y = 1, given x is parametrized by θ.
* Then we treat the square class as a positive class + estimate probability y = 2 and so on.
* So we end up w/ 3 classifiers, each trained to recognize 1 of the 3 classes.
* To summarize, we want to:
* 
* Finally to make a prediction when given a new input x, we run all i classifiers on the input x + pick the class i that maximizes hθ(x)i
* Whichever value of i gives us the highest probability (classifier that is most confident), we predict y to be that value.
* 
* 
* **k classes**