

Multiclass Classification: One-vs-all

What's a multiclass classification problem?

Let's say we want a learning algorithm to automatically put our email into different folders or to automatically tag our emails so we might have different folders or different tags for work email, email from our friends, email from our family, and emails about our hobby. And so we have a classification problem with four classes which we might assign to the classes $y = 1$, $y = 2$, $y = 3$, and $y = 4$.

Multiclass classification

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

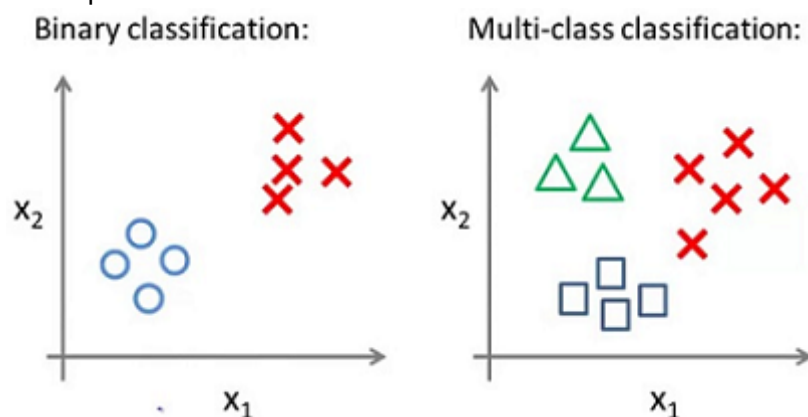
And another example, for medical diagnosis, if a patient comes into our office with maybe a stuffy nose, the possible diagnosis could be that they're not ill. Maybe that's $y = 1$. Or they have a cold, $y = 2$. Or they have a flu $y = 3$.

Medical diagrams: Not ill, Cold, Flu

And a third and final example if we are using machine learning to classify the weather, and we want to decide that the weather is sunny, cloudy, rainy, or snow, or if it's gonna be snow, y can take on a small number of values, and these are multiclass classification problems.

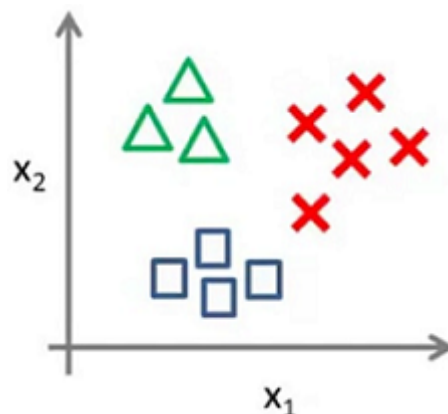
Weather: Sunny, Cloudy, Rain, Snow

For a multi-class classification problem our data sets may look like the following picture where we're using three different symbols to represent our three classes.






So the question is given the data set with three classes, how do we get a learning algorithm to work for the setting? We know how to fit a straight line to set for the positive and negative classes. We see an idea called **one-vs-all classification**. We can then take this and make it work for multi-class classification as well.

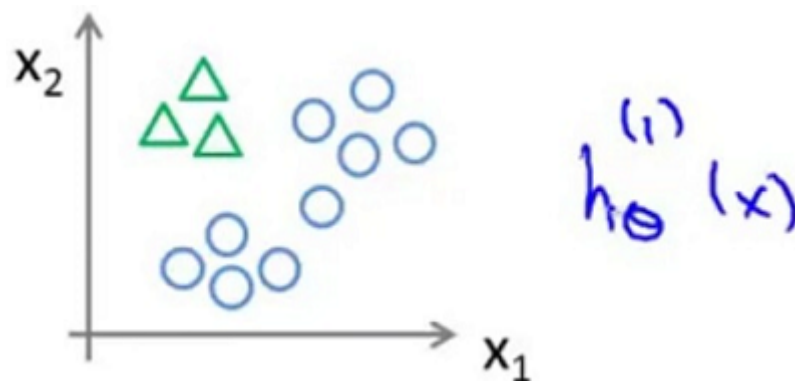
Let's say we have a training set like that:



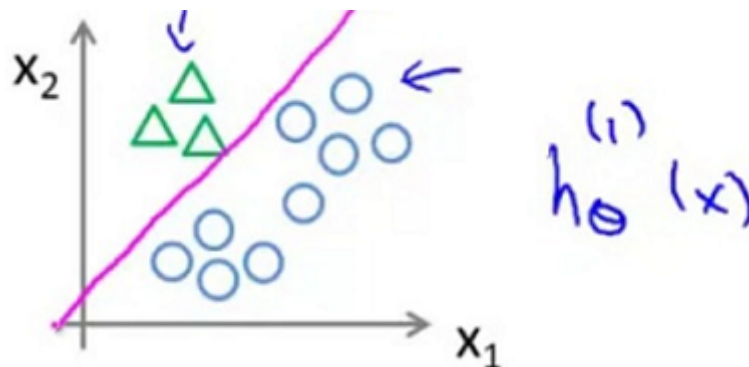
Where we have three classes of $y = 1$, we denote that with a triangle, if $y = 2$, the square, and if $y = 3$, the cross. What we're going to do is take our training set and turn this into three separate binary classification problems.

Class 1: 
Class 2: 
Class 3: 

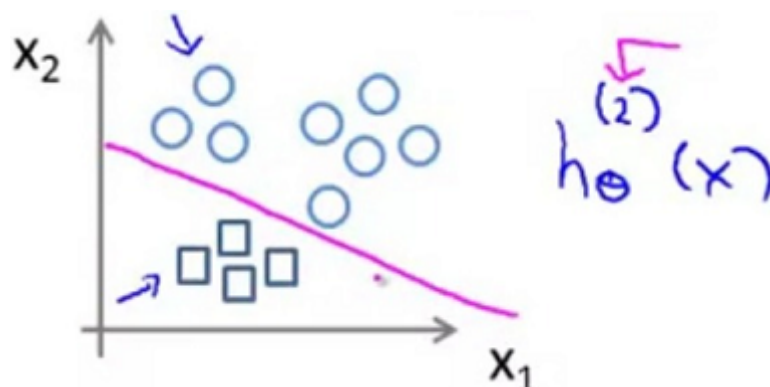
So let's start with class one which is the triangle. We're gonna essentially create a new sort of fake training set where classes two and three get assigned to the negative class. And class one gets assigned to the positive class. And we're going to fit a classifier which we're going to call $h_{\theta}^{(1)}(x)$ where the triangles are the positive examples and the circles are the negative examples.



So the triangles being assigned the value of one and the circles assigned the value of zero. And we're just going to train a standard logistic regression classifier and maybe that will give us a decision boundary.

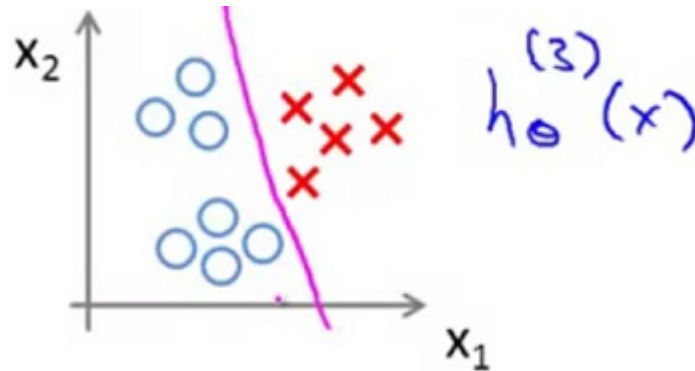


This superscript one on $h_{\theta}^{(1)}(x)$ stands for class one, so we're doing the decision boundary for the triangles of class one. Next we do the same thing for class two. We take the squares and assign the squares as the positive class, and assign everything else, the triangles and the crosses, as a negative class.



And then we fit a second logistic regression classifier $h_{\theta}^{(2)}(x)$. Where the superscript two denotes that we're now doing this, treating the square class as the positive class.

And finally, we do the same thing for the third class and fit a third classifier $h_{\theta}^{(3)}(x)$, and this will give us a decision boundary of the visible cross. This separates the positive and negative examples.



So to summarize, what we've done is, we've fit three classifiers.

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$

Thus trying to estimate what is the probability that y is equal to class i , given x and parameterized by θ .

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

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

Finally to make a prediction, when we're given a new input x , and we want to make a prediction. What we do is we just run all three of our classifiers on the input x and we then pick the class i that maximizes the three.

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

- $k - 1$

k

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

Summary

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 $y = \{0, 1 \dots n\}$.

Since $y = \{0, 1 \dots n\}$, 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.

$$y \in \{0, 1 \dots n\}$$

$$h_{\theta}^{(0)}(x) = P(y = 0|x; \theta)$$

$$h_{\theta}^{(1)}(x) = P(y = 1|x; \theta)$$

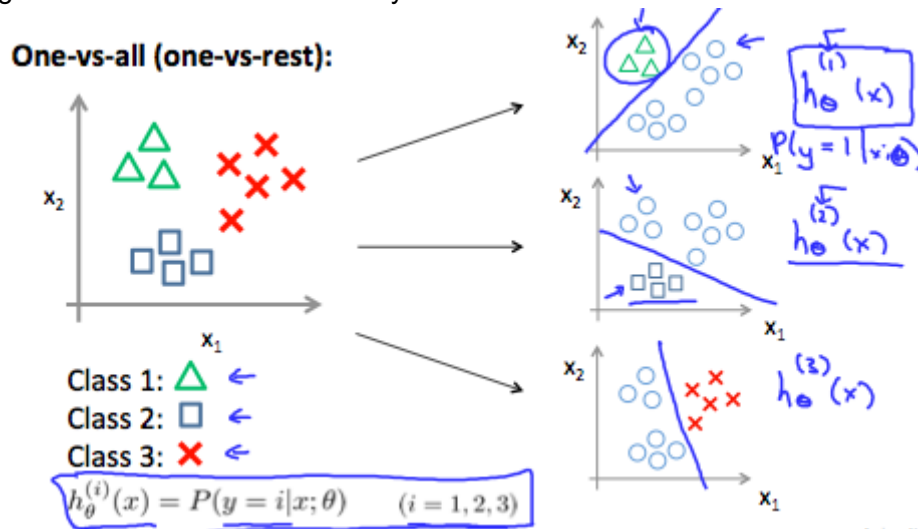
...

$$h_{\theta}^{(n)}(x) = P(y = n|x; \theta)$$

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

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 i to predict the probability that $P(y = i|x; \theta)$.

To make a prediction on a new x , pick the class i that maximizes $h_{\theta}(x)$