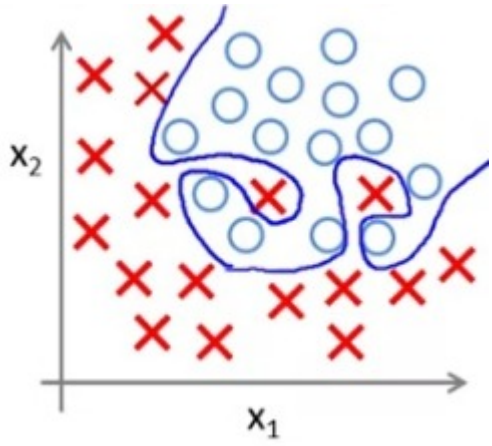
Regularized Logistic Regression

We saw earlier that Logistic Regression can also be prone to overfitting if we fit it with a very, sort of, high order polynomial features. Where g is the sigmoid function:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

and in particular we end up with a hypothesis, whose decision bound to be just sort of an overly complex and extremely contortive function that really isn't such a great hypothesis for the training set, and more generally if we have logistic regression with a lot of features. Not necessarily polynomial ones, but just with a lot of features we can end up with **overfitting**.



The following formula is our cost function for logistic regression.

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

And if we want to modify it to use regularization, all we need to do is add to it the following term $\sum_{j=1}^{n} \theta_j^2$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})] + \sum_{j=1}^{n} \theta_j^2$$

And this has to effect therefore, of penalizing the parameters $\theta_1, \theta_2, \dots, \theta_n$ from being too large. Then it will the have the effect that even though we're fitting a very high order polynomial with a lot of parameters. So long as we apply regularization and keep the parameters small we're more likely to get a more rasonable decision boundary, for separating the positive and the negative examples.

So when using regularization even when we have a lot of features, that regularization can help take care of the overfitting problem. How do we actually implement this? in order to implement the gradient descent algorithm and in order to modify this algorithm, to use a regularized cost function, all we need to do is pretty similar to what we did for linear regression is actually to just modify the second update rule as follows:

Gradient descent

Repeat $\{$

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

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$(j = 1, 2, 3, \dots, n)$

$\theta_1 \dots \theta_n$

$\}$

But this is not the same algorithm as we had, because now the hypothesis is defined using: $h_\theta(x) = \frac{1}{1+e^{-\theta^T X}}$.

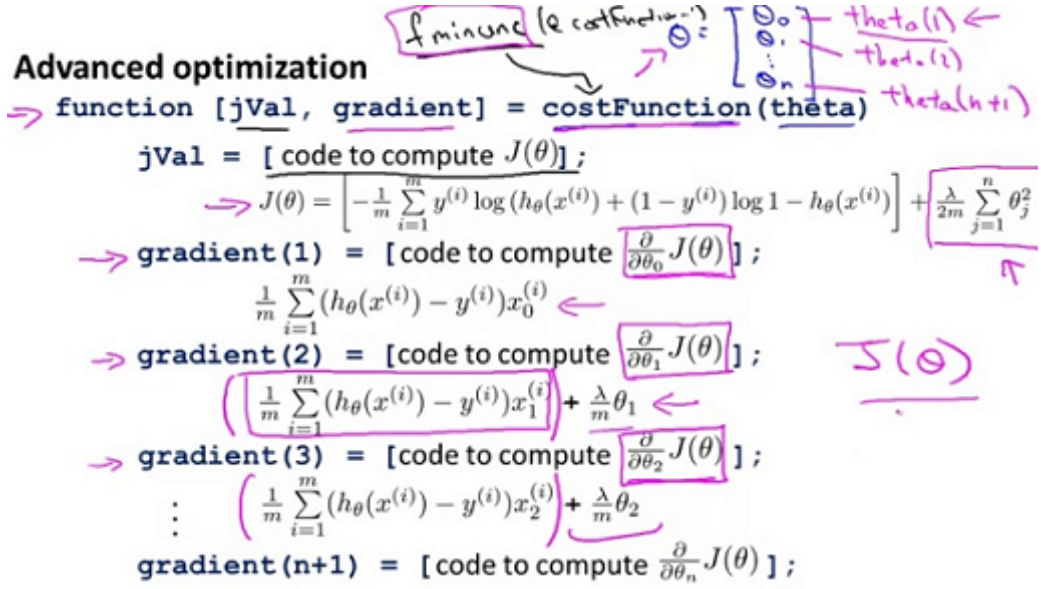
Video Question: When using regularized logistic regression, which of these is the best way to monitor whether gradient descent is working correctly?

- Plot $-[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})]$ as a function of the number of iterations and make sure it's decreasing.
- Plot $-[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}) - \sum_{j=1}^{n} \theta_j^2]$ as a function of the number of iterations and make sure it's decreasing.

> Plot $-[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}) + \sum_{j=1}^{n} \theta_j^2]$ as a function of the number of iterations and make sure it's decreasing.

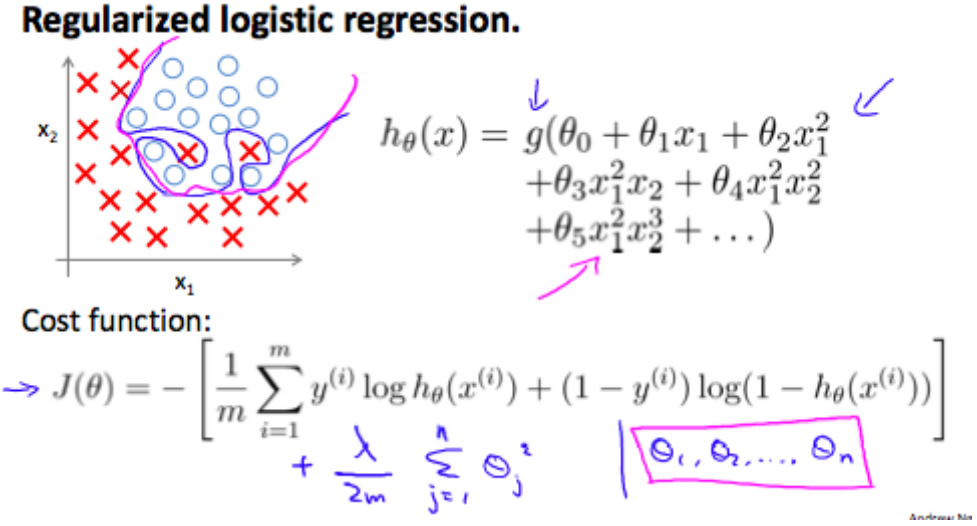
- Plot $\sum_{j=1}^{n} \theta_j^2$ as a function of the number of iterations and make sure it's decreasing.

How to implement regularized cost function:



Summary

We can regularize logistic regression in a similar way that we regularize linear regression. As a result, we can avoid overfitting. The following image shows how the regularized function, displayed by the pink line, is less likely to overfit than the non-regularized function represented by the blue line:



Cost Function

Recall that our cost function for logistic regression was:

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

We can regularize this equation by adding a term to the end:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})] + \sum_{j=1}^{n} \theta_j^2$$

The second sum, $\sum_{j=1}^{n} \theta_j^2$ **means to explicitly exclude** the bias term, θ_0. I.e. the θ vector is indexed from θ to n (holding $n + 1$ values, θ_0 through θ_n), and this sum explicitly skips θ_0, by running from 1 to n, skipping 0. Thus, when computing the equation, we should continuously update the two following equations: