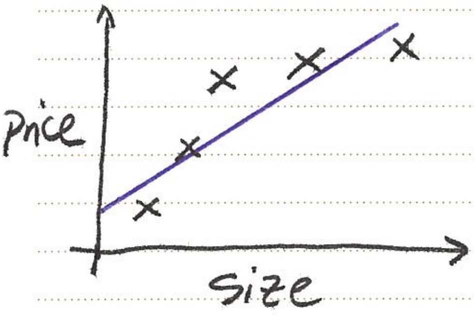
REGULARIZATION

Lecture 40: The Problem of Overfitting

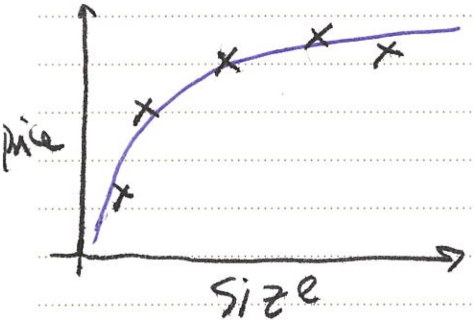
Consider fitting the same dataset with different linear regression models:



$$y = \theta_0 + \theta_1 x$$

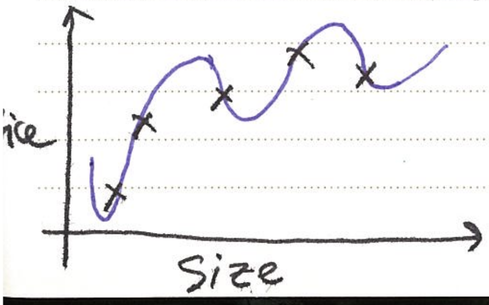
This model is said to be underfitting or that it has "high bias".

Our model has a strong preconception that house prices scale linearly with size, while the data clearly shows that's not the case.



$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

This seems like a reasonable model to use.

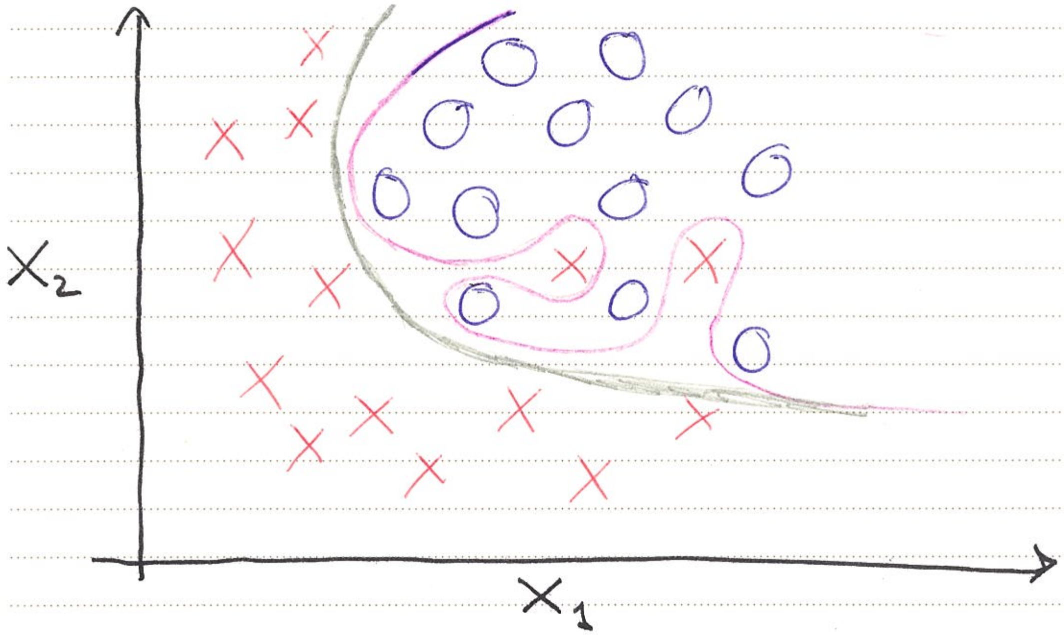


$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x$$

This model is fitting the dataset well, but it's clearly not generalizing well to other points $\rightarrow$ Overfitting

Overfitting can happen when there are too many features and the learning algorithm tries very hard to minimize the cost function as much as possible ($J(\theta) \approx 0$), but as a result the algorithm fails to generalize to new examples.

Our example was for linear regression, but it can of course also happen in logistic regression:



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

$\rightarrow$ reasonable fit

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

$\rightarrow$ overfit

Later in the course we'll talk about how we can detect when overfitting has occured. For now, we'll talk about ~~how~~ how to address the problem of overfitting once it's happened.

In the examples we looked at, it was relatively easy to see that overfitting is caused by higher order terms, just visually. Of course, where there are hundreds of features, you can't really plot the fit ~~&~~ to see what can be done about overfitting. We need something more systematic:

1. Reduce number of features.

   → Manually select which features to keep.

   → Model selection algorithm (later in course) → automatically determine which features to keep.

2. Regularization

   → Keep all features, but reduce impact of higher order terms causing overfitting.

   → Tends to work well when we have a lot of features, each of which contributes a bit to predicting y.

Note that reducing the number of features could mean loss of actual information, which may not be desirable.

Lecture 41: Cost function (for regularization)

Consider the following model in linear regression:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \underbrace{\theta_3 x^3 + \theta_4 x^4}$$

Assume these terms are causing overfitting.

If we have done feature scaling so that $|x| \lesssim 1$, it follows that θ_3 & θ_4 should be large compared to, say, θ_1:

$$\theta_1 x \sim \theta_3 x^3$$

$$\rightarrow \theta_3 \sim \frac{\theta_1}{x^2} \gg \theta_1 \quad \text{because } \frac{1}{x^2} \gg 1$$

So if we do feature scaling, we see that there's a correspondence between higher order terms and their θ multipliers being larger.

* Therefore, to supress higher order terms, we can penalize large values of θ in our cost functions. *

(Andrew Ng makes this point but he doesn't mention anything about feature scaling. I don't see how this argument holds generically without feature scaling.)

Here's one way of accomplishing this for linear regression.

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

Regularization Parameter

By definition, θ_0 is not included.

Note the importance of λ:

→ Too large: will result in underfitting because it penalizes too much.

→ Too low: won't solve the overfitting problem.

This particular regularization technique is called Tikhonov regularization. It tries to decrease the L^2 norm of Θ. Apparently this has a nice Bayesian interpretation where a prior probability distribution on the space of solution Θ is assumed.

Lecture 42: Regularized Linear Regression

Let's work out Gradient Descent and the Normal Equation with regularization

Gradient Descent

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}] x_0^{(i)} \rightarrow = 1 \ \forall i \text{ by def.}$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \left[\sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}] x_j^{(i)} + \lambda \theta_j \right] \quad j \in \{1, \dots, n\}$$

We can write this as:

Repeat until converge {

$$\theta_j := \theta_j \left[1 - (1 - \delta_{j,0}) \frac{\alpha \lambda}{m} \right] - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

} $\quad j \in \{0, 1, \dots, n\}$

Noting that $h_\theta(x) = \theta^T x$ & letting $\mathbb{1}^{(0)} \equiv \begin{bmatrix} 0 & & & \\ & 1 & & 0 \\ & & 1 & \\ & 0 & & \ddots \\ & & & & 1 \end{bmatrix}$

this can be rewritten as (see to the left of page 18)

Repeat until converge {

$$\theta := \left[\mathbb{1} - \frac{\alpha \lambda}{m} \mathbb{1}^{(0)} \right] \theta - \frac{\alpha}{m} X^T (X\theta - Y)$$

}

Identity matrix

Normal Equation (see to the left of page 18)

$$J(\theta) = \frac{1}{2m}\left[(X\theta - Y)^T(X\theta - Y) + \lambda(\mathbb{1}^{(0)}\theta)^T(\mathbb{1}^{(0)}\theta)\right]$$

$$\frac{\partial J}{\partial \theta} = \frac{1}{2m}\left[2X^T(X\theta - Y) + 2\lambda\,\mathbb{1}^{(0)}\theta\right]$$

$$\frac{\partial J}{\partial \theta} = 0 \rightarrow \left[X^TX + \lambda\,\mathbb{1}^{(0)}\right]\theta = X^TY$$

$$\rightarrow \boxed{\theta = \left[X^TX + \lambda\,\mathbb{1}^{(0)}\right]^{-1} X^TY}$$

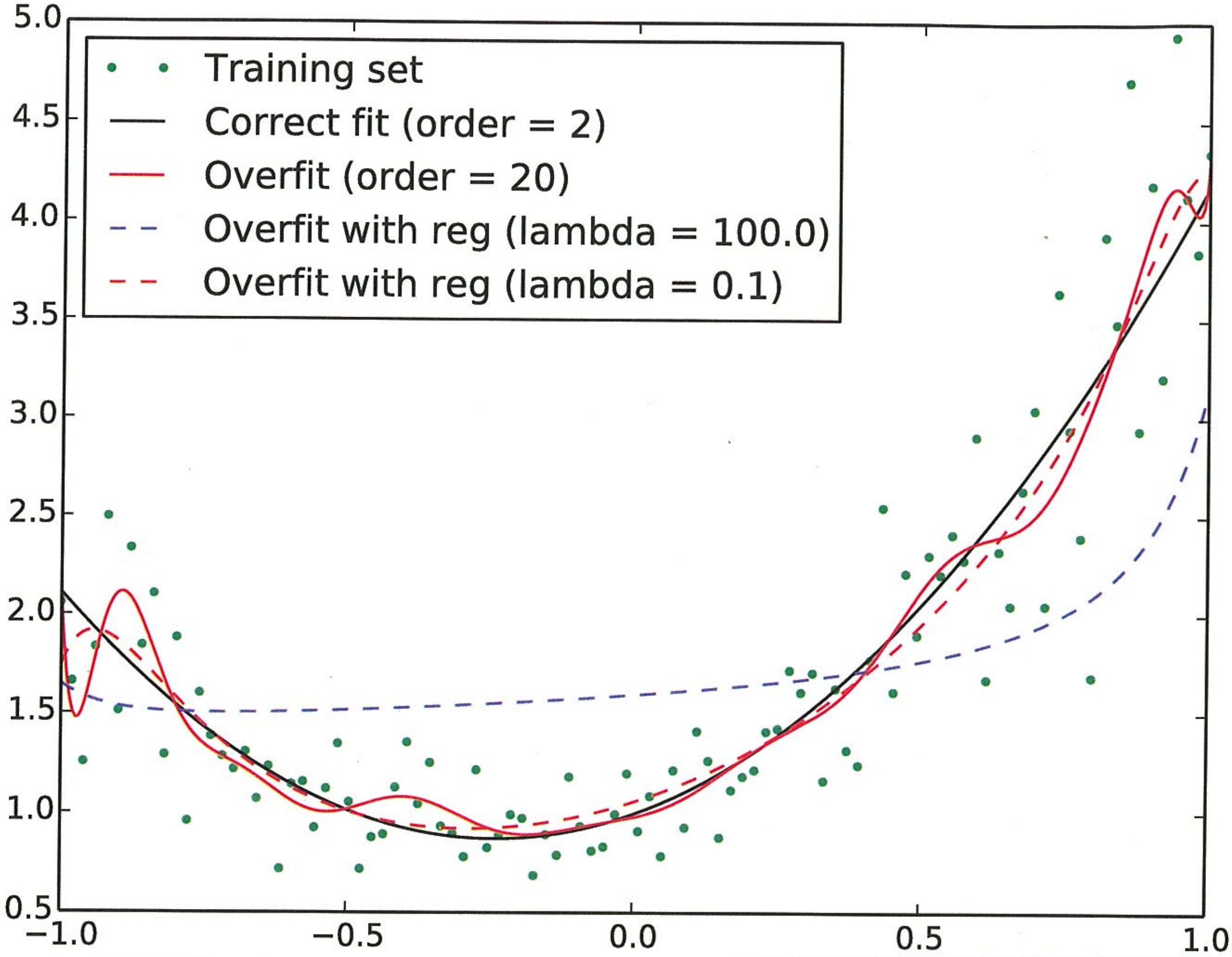
Always invertible!

where again $\mathbb{1}^{(0)} = \begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & \ddots\ 1 \end{bmatrix}$

In Gradient Descent, we see that the change due to regularization amounts to $\theta_j := \theta_j\underbrace{\left(1 - \alpha\frac{\lambda}{m}\right)} + \cdots$

$j \in \{1, \ldots,$

Note that $1 - \alpha\frac{\lambda}{m} < 1 \rightarrow$ making θ smaller!



Here's a concrete example. I generated the training set as follows: $y(x) = (1 + 2x + x^2)(1 + \varepsilon)$

Random number drawn from a normal distribution with mean $= 0$ & $\sigma = 0.2$

What are the various fits above?

* Correct fit: fit $y = \theta_0 + \theta_1 x + \theta_2 x^2$ using linear regression.

* overfit (order = 20): fit $y = \sum_{n=0}^{20} \theta_n x^n$.

  As it can be seen, it tries too hard to fit the training set, by capturing the noise.

* overfit with reg ($\lambda = 100$): fit $y = \sum_{n=0}^{20} \theta_n x^n$ using regulization with $\lambda = 100$. It's easy to see that this value of λ is too high & is causing underfitting.

* overfit with reg ($\lambda = 0.1$): fit $y = \sum_{n=0}^{20} \theta_n x^n$ using regularization with $\lambda = 0.1$. This seems like a good value of λ, since the fit is quite close to the correct fit.

Lecture 43: Regularized Logistic Regression

In logistic regression we have $h_\theta(x) = g(\theta^T x)$

Sigmoid function

Since it's a function of $\theta^T x$, the same argument we used for linear regression should work to suppress higher order terms which cause overfitting.

We will similarly modify the cost function for logistic regression (see page 33)

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

Of course: $h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}}$

$$\boxed{\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \lambda(1-\delta_{j,0})\theta_j\right]}$$

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

Gradient Descent:

Repeat until converge {

$$\theta_j := \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \lambda(1-\delta_{j,0})\theta_j\right]$$

}