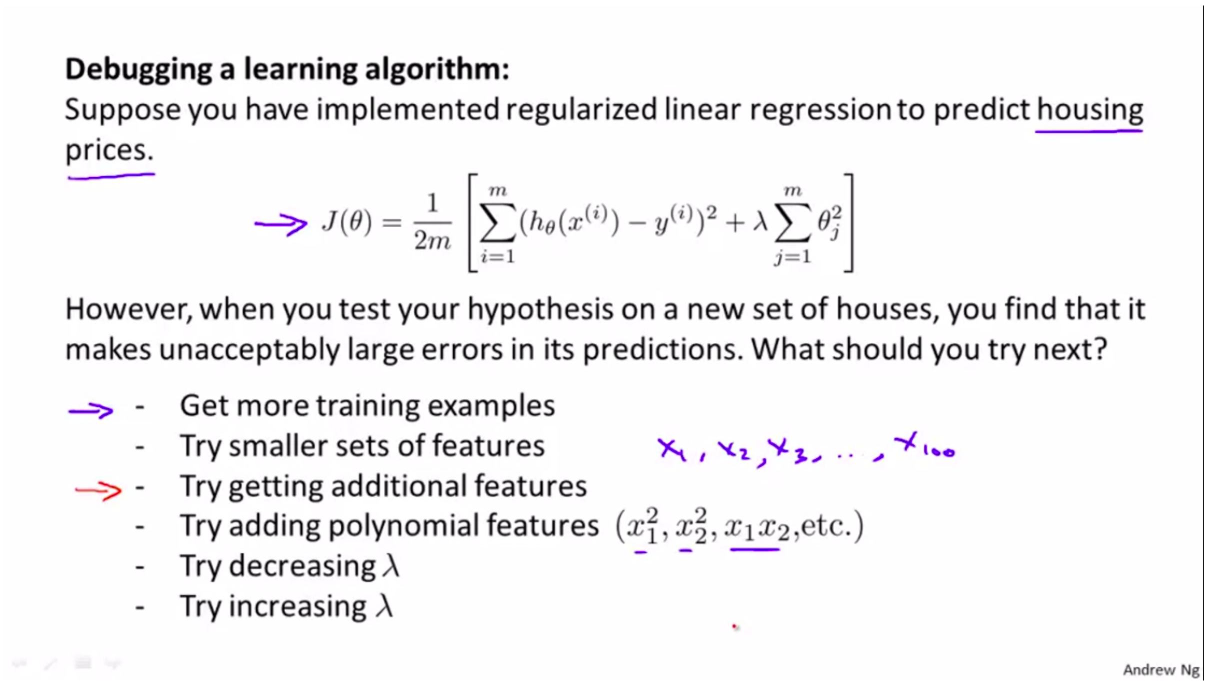
Week 6 - Lecture 1

Deciding what to try next



Debugging a learning algorithm:
Suppose you have implemented regularized linear regression to predict housing prices.

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

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features $x_1, x_2, x_3, \ldots, x_{100}$
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc.})$
- Try decreasing λ
- Try increasing λ

Andrew Ng

Evaluating Hypothesis

Once we have done some trouble shooting for errors in our predictions by:

- Getting more training examples
- Trying smaller sets of features
- Trying additional features
- Trying polynomial features
- Increasing or decreasing λ

We can move on to evaluate our new hypothesis.

A hypothesis may have a low error for the training examples but still be inaccurate (because of **overfitting**). Thus, to evaluate a hypothesis, given a dataset of training examples, we can split up the data into two sets: a **training set** and a **test set**. Typically, the training set consists of 70% of your data and the test set is the remaining 30%.

The new procedure using these two sets is then:

1. Learn Θ and minimize $J_{Train}(\Theta)$ using the training set
2. Compute the test set error $J_{Test}(\Theta)$.

The test set error

For linear regression:

$$J_{Test}(\Theta) = \frac{1}{2m_{Test}} \sum_{i=1}^{m_{Test}} \left(h_\Theta\left(x_{Test}^{(i)}\right) - y_{Test}^{(i)}\right)^2 \tag{1}$$

For classification (misclassification error, aka 0/1 misclassification error):

$$err(h_\Theta(x), y) = \begin{cases} \text{ if } h_\Theta(x) \geq 0.5 \text{ and } y = 0 \text{ or } h_\Theta(x) < 0.5 \text{ and } y = 1 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

This gives us a binary 0 or 1 error result based on a misclassification. The average test error for the test set is:

$$TestError = \frac{1}{m_{Test}} \sum_{i=1}^{m_{Test}} err\left(h_\Theta\left(x_{Test}^{(i)}\right), y_{Test}^{(i)}\right) \tag{3}$$

This gives us the proportion of the test data that was misclassified.

Model Selection and Train/Validation/Test Sets

Just because a learning algorithm fits a training set well, that does not mean it is a good hypothesis. It could over fit and as a result your predictions on the test set would be poor. The error of your hypothesis as measured on the data set with which you trained the parameters will be lower than the error on any other data set.

Given many models with different polynomial degrees, we can use a systematic approach to identify the 'best' function. In order to choose the model of your hypothesis, you can test each degree of polynomial and look at the error result.

One way to break down our dataset into the three sets is:

- Training set: 60%
- Cross validation set: 20%
- Test set: 20%

We can now calculate three separate error values for the three different sets using the following method:

- Optimize the parameters in Θ using the training set for each polynomial degree.
- Find the polynomial degree d with the least error using the cross validation set.
- Estimate the generalization error using the test set with $J_{test}(\Theta^{(d)})$, ($d = theta$ from polynomial with lower error);

This way, the degree of the polynomial d has not been trained using the test set.

Bias vs. Variance

In this section we examine the relationship between the degree of the polynomial d and the underfitting or overfitting of our hypothesis.

- We need to distinguish whether bias or variance is the problem contributing to bad predictions.

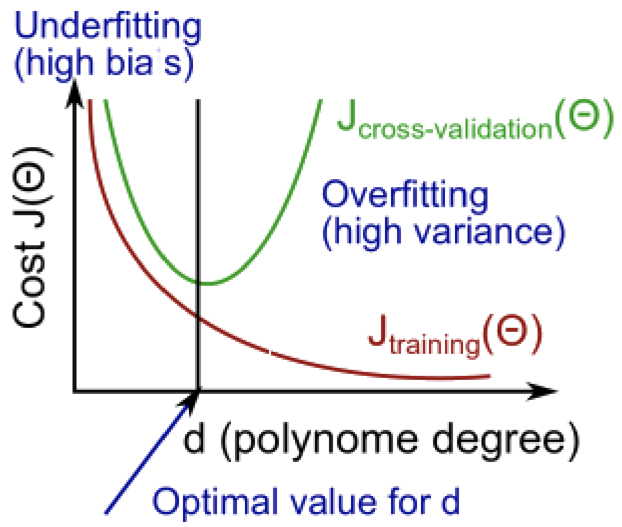
- **High bias is underfitting and high variance is overfitting**. Ideally, we need to find a golden mean between these two.

The training error will tend to decrease as we increase the degree d of the polynomial.

At the same time, the cross validation error will tend to decrease as we increase d up to a point, and then it will increase as d is increased, forming a convex curve.

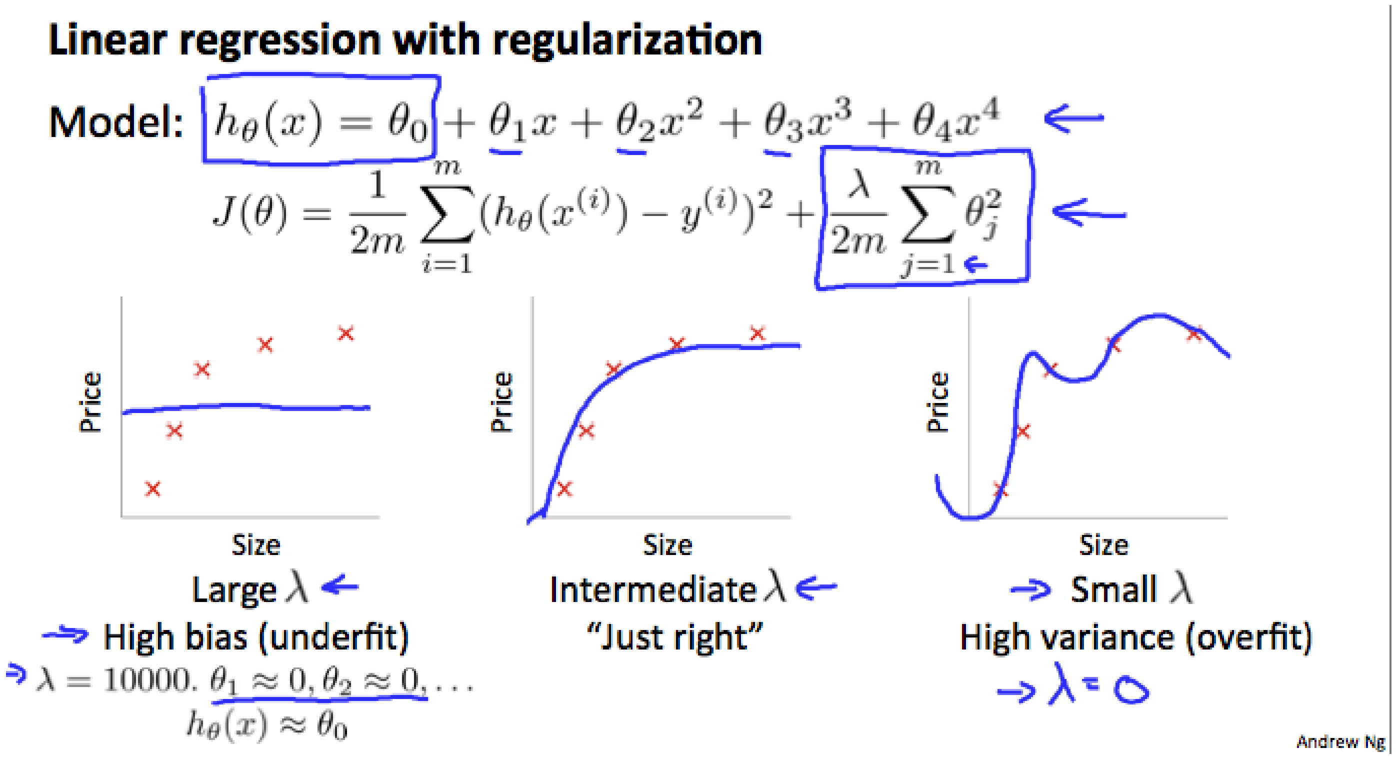
- **High bias (underfitting)**: both $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ will be high. Also, $J_{CV}(\Theta) \approx J_{train}(\Theta)$.
- **High variance (overfitting)**: $J_{train}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be much greater than $J_{train}(\Theta)$.

The is summarized in the figure below:



Regularization and Bias/Variance

Note: The regularization term below and through out the video should be $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ and NOT $\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$



Linear regression with regularization

Model: $\boxed{h_\theta(x) = \theta_0} + \underline{\theta_1} x + \underline{\theta_2} x^2 + \underline{\theta_3} x^3 + \theta_4 x^4 \leftarrow$

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

Price
Size
Price
Size
Price
Size

Large $\lambda \leftarrow$
$\rightarrow$ High bias (underfit)
$\rightarrow \lambda = 10000.\ \underline{\theta_1 \approx 0, \theta_2 \approx 0}, \ldots$
$h_\theta(x) \approx \theta_0$

Intermediate $\lambda \leftarrow$
"Just right"

$\rightarrow$ Small λ
High variance (overfit)
$\rightarrow \lambda = 0$

Andrew Ng

In the figure above, we see that as λ increases, our fit becomes more rigid. On the other hand, as λ approaches 0, we tend to overfit the data. So how do we choose our parameter λ to get it 'just right' ? In order to choose the model and the regularization term λ, we need to:

1. Create a list of lambdas (i.e. $\lambda \in \{0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24\}$);
2. Create a set of models with different degrees or any other variants.
3. Iterate through the λs and for each λ go through all the models to learn some Θ.
4. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ without regularization or $\lambda = 0$.
5. Select the best combo that produces the lowest error on the cross validation set.
6. Using the best combo Θ and λ, apply it on $J_{test}(\Theta)$ to see if it has a good generalization of the problem.

Learning Curves

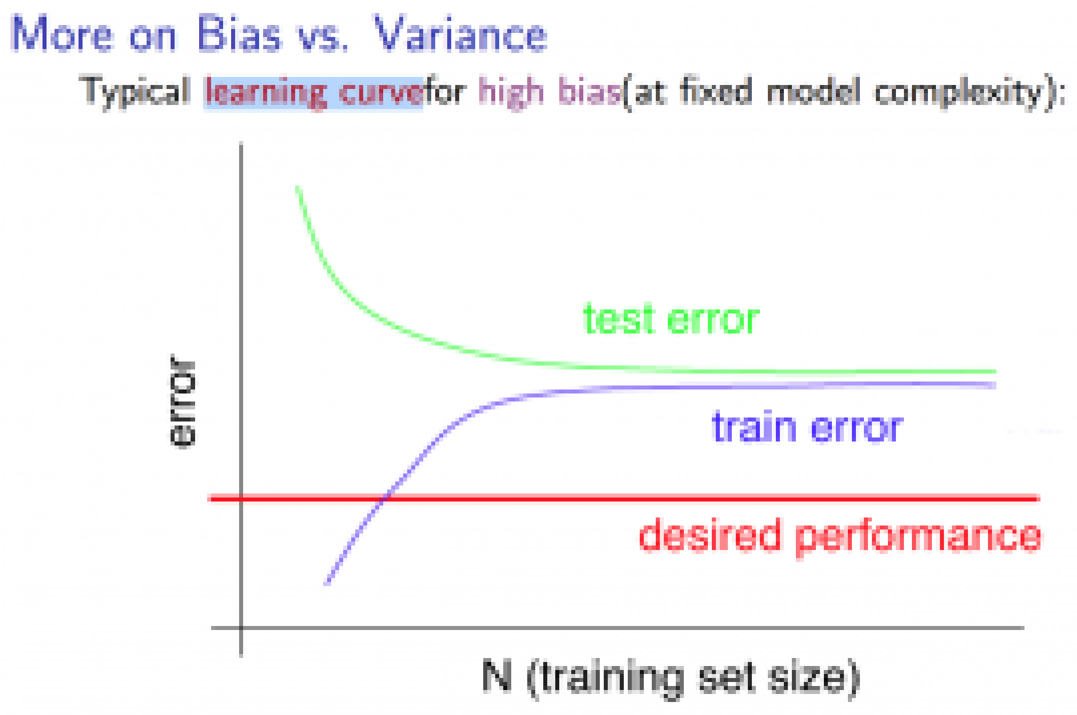
Training an algorithm on a very few number of data points (such as 1, 2 or 3) will easily have 0 errors because we can always find a quadratic curve that touches exactly those number of points. Hence:

- As the training set gets larger, the error for a quadratic function increases.
- The error value will plateau out after a certain m, or training set size.

Experiencing high bias:

- **Low training set size:** causes $J_{train}(\Theta)$ to be low and $J_{CV}(\Theta)$ to be high.
- **Large training set size:** causes both $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ to be high with $J_{train}(\Theta) \approx J_{CV}(\Theta)$.

If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

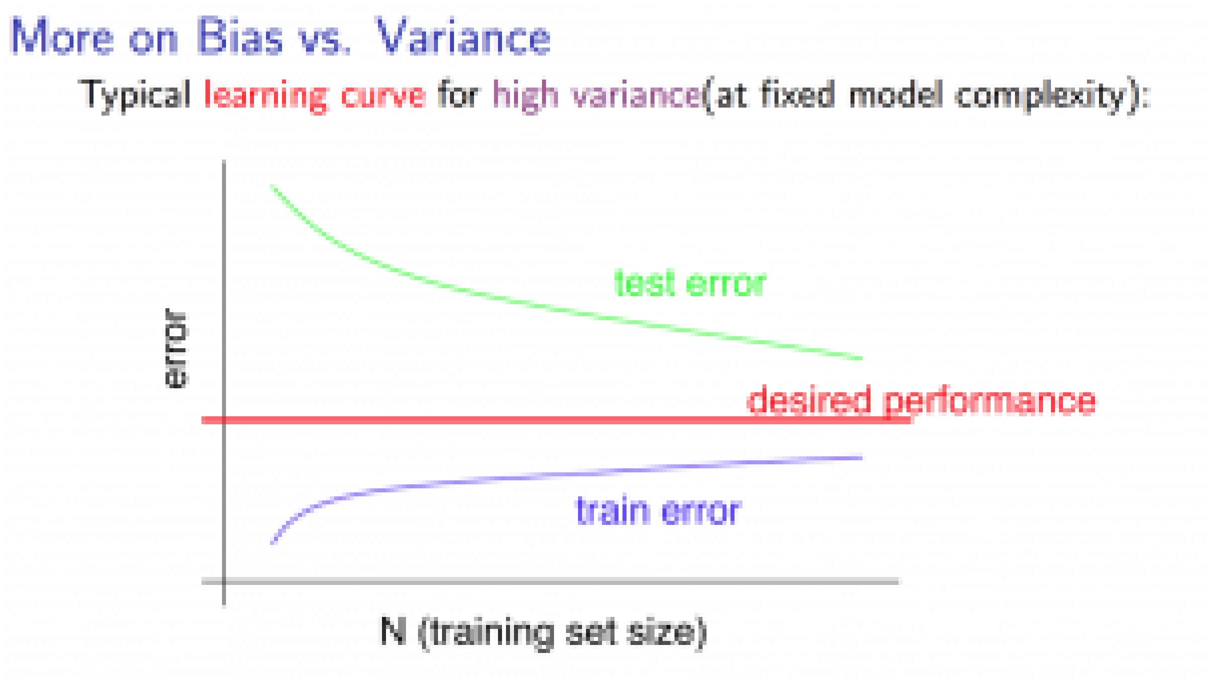


Experiencing high variance:

- **Low training set size:** $J_{train}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be high.

- **Large training set size:** $J_{train}(\Theta)$ increases with training set size and $J_{CV}(\Theta)$ continues to decrease without leveling off. Also, $J_{train}(\Theta) < J_{CV}(\Theta)$ but the difference between them remains significant.

If a learning algorithm is suffering from high variance, getting more training data is likely to help.



Deciding what to do next (revisited)

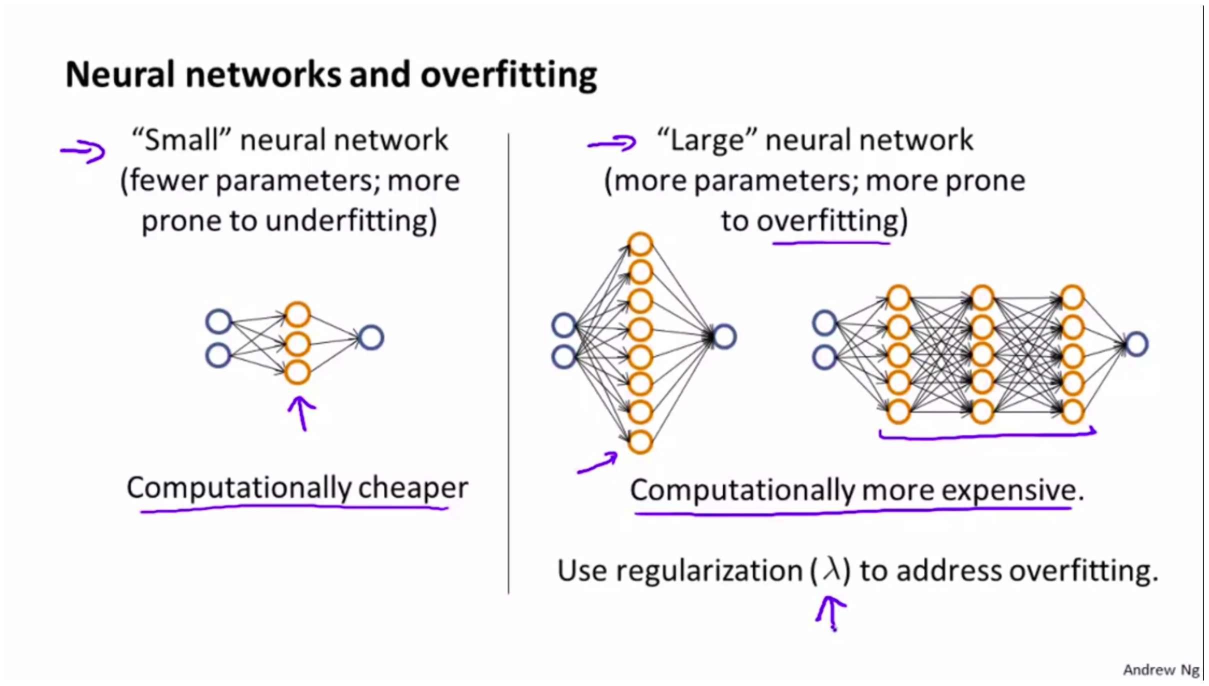
Our decision process can be broken down as follows:

- **Getting more training examples:** Fixes high variance
- **Trying smaller sets of features:** Fixes high variance
- **Adding features:** Fixes high bias
- **Adding polynomial features:** Fixes high bias
- **Decreasing λ:** Fixes high bias
- **Increasing λ:** Fixes high variance.

Diagnosing Neural Networks

- A neural network with fewer parameters is **prone to underfitting**. It is also computationally cheaper.
- A large neural network with more parameters is **prone to overfitting**. It is also computationally expensive. In this case you can use regularization (increase λ) to address the overfitting.

Using a single hidden layer is a good starting default. You can train your neural network on a number of hidden layers using your cross validation set. You can then select the one that performs best.



Neural networks and overfitting

"Small" neural network (fewer parameters; more prone to underfitting)

Computationally cheaper

"Large" neural network (more parameters; more prone to overfitting)

Computationally more expensive.

Use regularization (λ) to address overfitting.

Andrew Ng

Model Complexity Effects:

- Lower-order polynomials (low model complexity) have high bias and low variance. In this case, the model fits poorly consistently.
- Higher-order polynomials (high model complexity) fit the training data extremely well and the test data extremely poorly. These have low bias on the training data, but very high variance.
- In reality, we would want to choose a model somewhere in between, that can generalize well but also fits the data reasonably well.

References

[1] Machine Learning - Stanford University (https://www.coursera.org/learn/machine-learning)