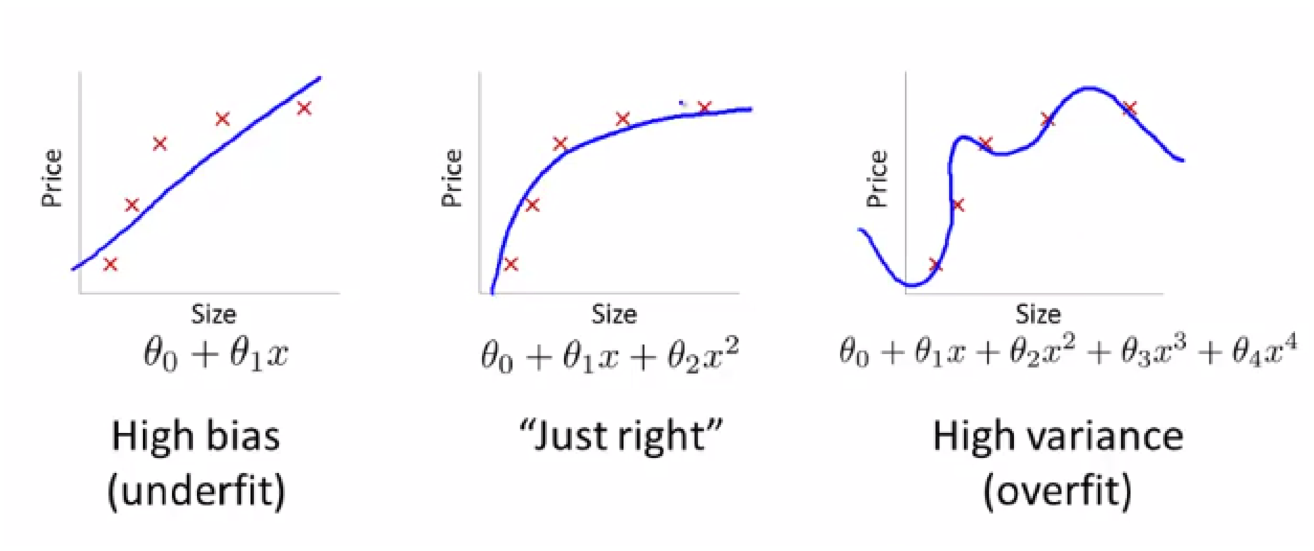
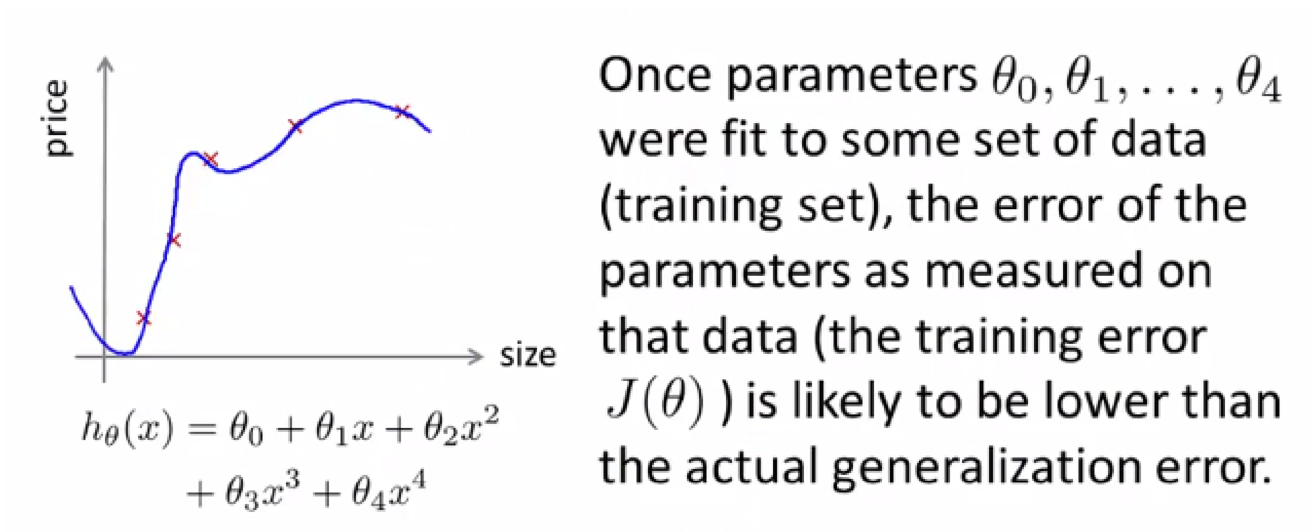
ML:Advice for Applying Machine Learning

Review

Over and under fitting



Overfitting



Once parameters $\theta_0, \theta_1, \ldots, \theta_4$ were fit to some set of data (training set), the error of the parameters as measured on that data (the training error $J(\theta)$) is likely to be lower than the actual generalization error.

Deciding What to Try Next

Errors in your predictions can be troubleshooted by:

- Getting more training examples
  - A lot of time can be wasted here
- Trying smaller sets of features
  - Prevent over fitting
- Trying additional features
  - A lot of time can be wasted here
  - Determine if this will even help
- Trying polynomial features
- Increasing or decreasing λ

Don't just pick one of these avenues at random. We'll explore diagnostic techniques for choosing one of the above solutions in the following sections.

Evaluating a Hypothesis

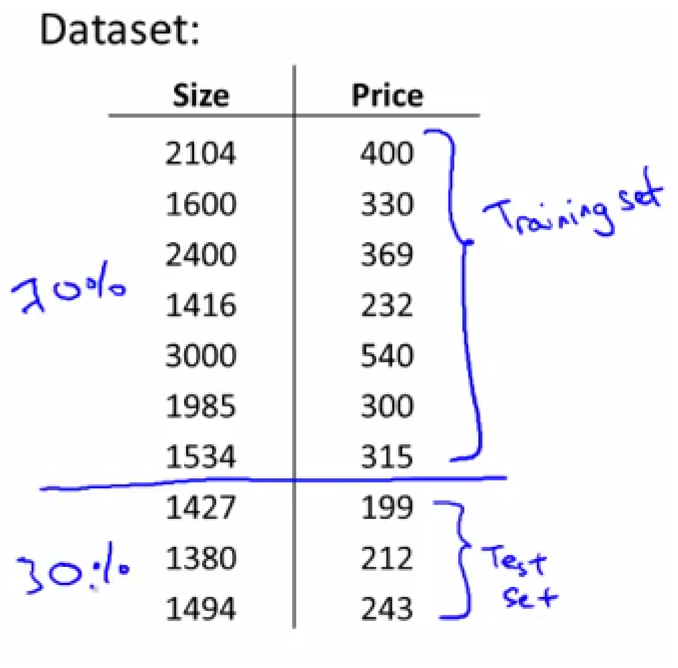
A hypothesis may have low error for the training examples but still be inaccurate (because of overfitting).

With a given dataset of training examples, we can split up the data into two sets:

- a **training set**
  - +- 70%
- a **test set**
  - +- 30%
- NOTE: Randomly select the data if ordering in the data is important

The new procedure using these two sets is then:

1. Learn Θ and minimize Jtrain(Θ) using the training set
2. Compute the test set error Jtest(Θ)


Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

70%
Training set
30%
Test set

The test set error

1. For linear regression:

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

2. For classification ~ Misclassification error (aka 0/1 misclassification error):

$$err(h_\Theta(x), y) = \begin{matrix} 1 & \text{if } h_\Theta(x) \geq 0.5 \; and \; y = 0 \; or \; h_\Theta(x) < 0.5 \; and \; y = 1 \text{ (incorrect prediction)} \\ 0 & otherwise \end{matrix}$$

This gives us a binary 0 or 1 error result based on a misclassification.

The average test error for the test set is

$$\text{Test Error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\Theta(x_{test}^{(i)}), y_{test}^{(i)})$$

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

Model Selection and Train/Validation/Test Sets

Example: How do we select the polynomial degree?

In order to choose the model of your hypothesis, you can test each degree of polynomial and look at the error result.

NOTE:

- Just because a learning algorithm fits a training set well, that does not mean it is a good hypothesis.
- The error of your hypothesis as measured on the data set with which you trained the parameters will be lower than any other data set.

Without the Validation Set (note: this is a bad method - do not use it)

1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the test set.
3. Estimate the generalization error also using the test set with $J_{\text{test}}(\Theta^{(d)})$, ($d = \theta$ from polynomial with lower error);
    1. report generalized error = error of $J_{\text{test}}(\theta^{(d)})$

In this case, we have trained one variable, d, or the degree of the polynomial, using the test set. This will cause our error value to be greater for any other set of data as the stated error is the most optimistic error for the model.

NOTE The problem with this method is that the selection process does not take into account the possibility of over fitting of the variable d. i.e our extra parameter d is fit to the test set and we do in fact not know how well it will do on data that the model has not seen. *Solutions:* Use another (unseen) data set to test the generalized error

Use of the *Cross validation* set

To solve this, we can introduce a third set, the **Cross Validation Set**, to serve as an intermediate set that we can train d with. Then our test set will give us an accurate, non-optimistic error.

One example 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.

With the Validation Set (note: this method presumes we do not also use the CV set for regularization)

1. Optimize the parameters in Θ using the ***training set*** for each polynomial degree.

2. Find the polynomial degree d with the least error using the ***CV set***
3. Estimate the generalization error using the ***test set*** with $J_{\text{test}}(\Theta^{(d)})$, ($d = \theta$ from polynomial with lower error);

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

(Mentor note: be aware that using the CV set to select 'd' means that we cannot also use it for the validation curve process of setting the lambda value).

Take away

When calculating errors be sure to consider how the different layers of trained variables are fitted, and how the data sets might as a result report incorrect or optimistic error values.

We need to have n independent data sets for every n layer of training variables (think of them as training variables that are dependent on each other) .

i.e. if we want to include the selection of lambda in the method stated above we will need a forth independent data set.

Diagnosing 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.

High bias (underfitting):

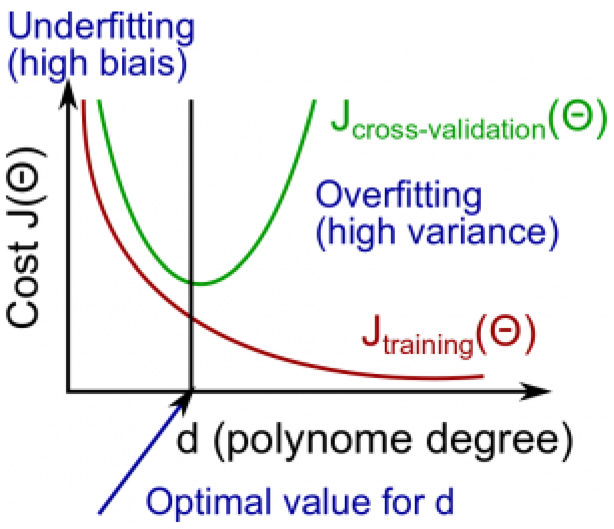
- $J_{\text{train}}(\Theta)$ and $J_{\text{CV}}(\Theta)$ will be high.
- $J_{CV}(\Theta) \approx J_{\text{train}}(\Theta)$.

High variance (overfitting):

- $J_{\text{train}}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be much greater than $J_{\text{train}}(\Theta)$.
- $J_{CV}(\Theta) >> J_{\text{train}}(\Theta)$

Our goal when dealing with these two terms is to find the golden mean between these two.

Example: Selecting polynomial degree d



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.

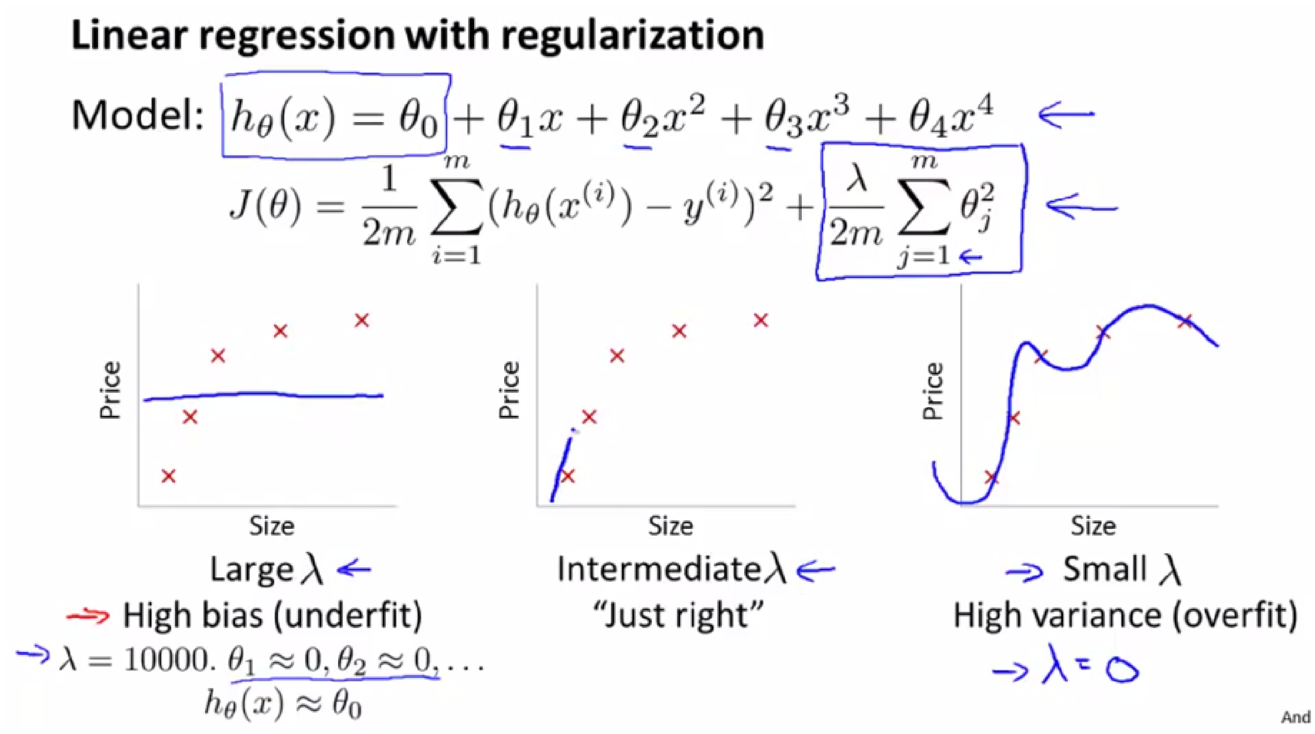
Regularization and Bias/Variance

Context question: How does regularization affect Bias/Variance?

Instead of looking at the degree d contributing to bias/variance, now we will look at the regularization parameter λ.

- Small λ: High variance (overfitting)
- Intermediate λ: just right
- Large λ: High bias (underfitting)

A large lambda heavily penalizes all the Θ parameters, which greatly simplifies the line of our resulting function, so causes underfitting.



Linear regression with regularization

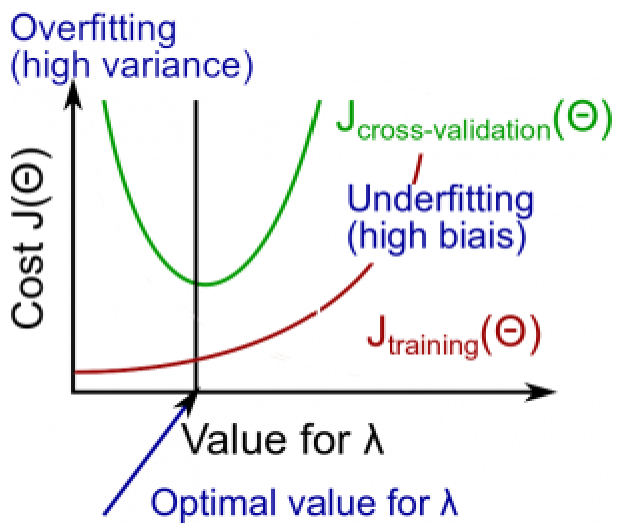
Model: $\boxed{h_\theta(x) = \theta_0} + \underline{\theta_1} x + \underline{\theta_2} x^2 + \underline{\theta_3} x^3 + \theta_4 x^4 \leftarrow$

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

The relationship of λ to the training set and the variance set is as follows:

- **Low λ**: $J_{\text{train}}(\Theta)$ is low and $J_{CV}(\Theta)$ is high (high variance/overfitting).
- **Intermediate λ**: $J_{\text{train}}(\Theta)$ and $J_{CV}(\Theta)$ are somewhat low and $J_{\text{train}}(\Theta) \approx J_{CV}(\Theta)$.
- **Large λ**: both $J_{\text{train}}(\Theta)$ and $J_{CV}(\Theta)$ will be high (underfitting /high bias)

The figure below illustrates the relationship between lambda and the hypothesis:



In order to choose the *model* and the *regularization* λ, we need:

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 $J_{\text{test}}(\Theta)$ to see if it has a good generalization of the problem.

NOTE: Calculating the cost functions for regression

While the cost function used to find the optimal θ values is:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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

When we calculate the cost error values:

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

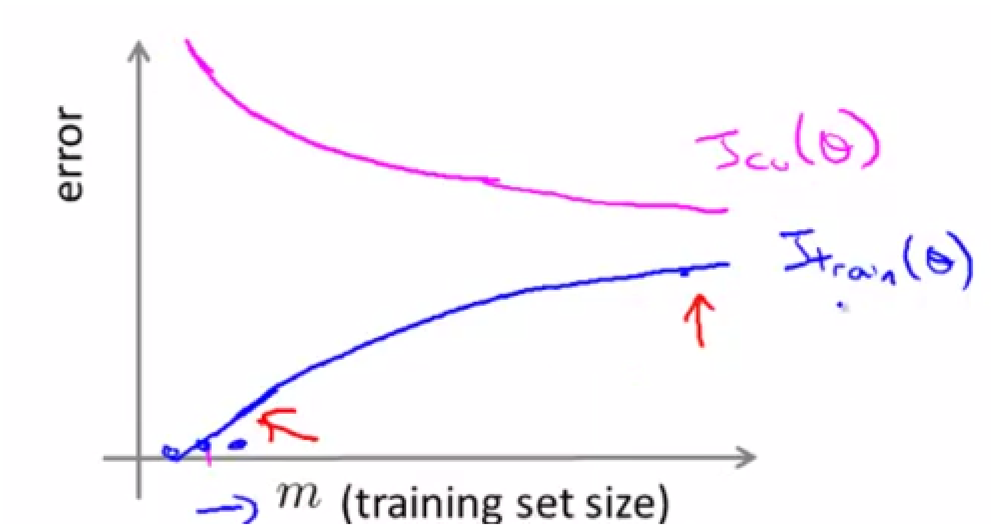
Learning Curves

Context:

- If you wanted to sanity check that your algorithm is working correctly, or if you want to improve the performance of the algorithm.
- It is a tool to diagnose if a learning algorithm may be suffering from bias or variance problem or a bit of both.

Core Idea:

When training set n is small, its very easy to fit those points, the consequence of this is that your training error will also be small. When n is big it becomes harder for all the data to be fitted perfectly, the consequence of this is that your training error is likely to be larger. The converse will be true for when the cross validation error is looked at.

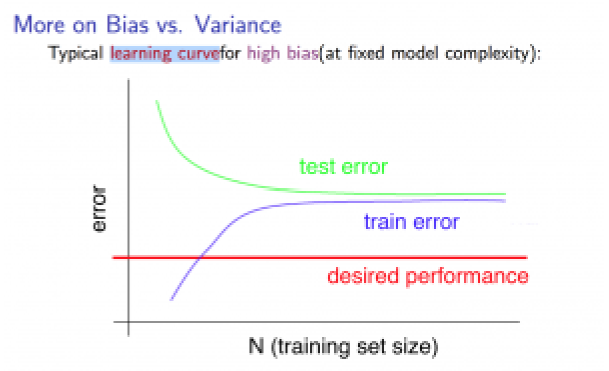


Example:

Training 3 examples will easily have 0 errors because we can always find a quadratic curve that exactly touches 3 points.

- 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.

With high bias

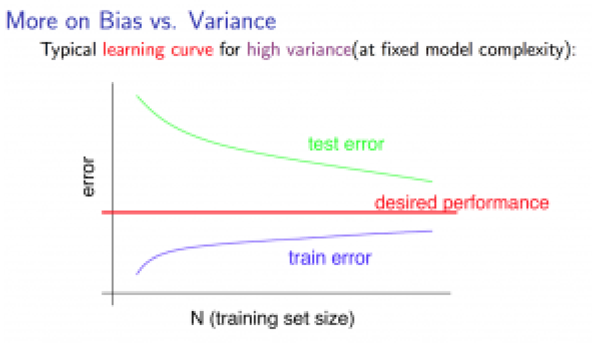


- **Low training set size**: causes $J_{\text{train}}(\Theta)$ to be low and $J_{CV}(\Theta)$ to be high.
- **Large training set size**: causes both $J_{\text{train}}(\Theta)$ and $J_{CV}(\Theta)$ to be high with $J_{\text{train}}(\Theta) \approx J_{CV}(\Theta)$.

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

For high variance, we have the following relationships in terms of the training set size:

With high variance



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

NOTE 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
- Add regularization to model: Fixes high variance / high bias
  - Decreasing λ: Fixes high bias
  - Increasing λ: Fixes high variance

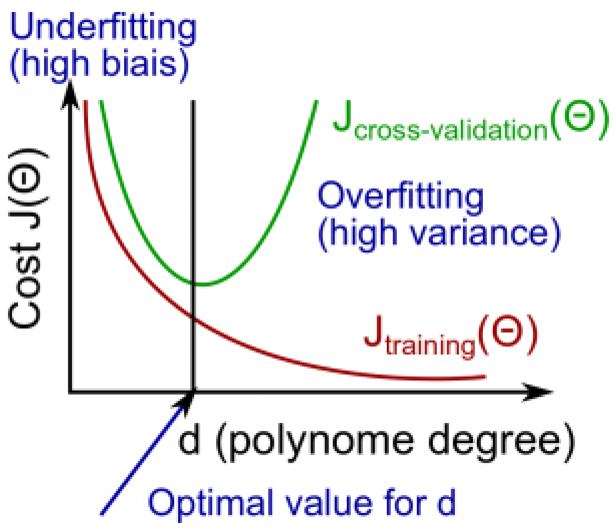
Diagnosing Neural Networks

- 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.*

Some extra notes:

The behavior of Neural Networks is very similar to the regression example when looking at the polynomial degree d



If $J_{\text{CV}} >> J_{\text{test}}$, the NN is overfitting (high variance), the consequence of this is that adding more hidden layers will not improve performance.

If $J_{\text{CV}} \approx J_{\text{test}}$, the NN is underfitting (high bias), the consequence of this is that adding more hidden layers will likely improve performance.

Model Selection:

Choosing M the order of polynomials.

How can we tell which parameters Θ to leave in the model (known as "model selection")?

There are several ways to solve this problem:

- Get more data (very difficult).
- Choose the model which best fits the data without overfitting (very difficult).
- Reduce the opportunity for overfitting through regularization.

Bias: approximation error (Difference between expected value and optimal value)

- High Bias = UnderFitting (BU)
- $J_{\text{train}}(\Theta)$ and $J_{CV}(\Theta)$ both will be high and $J_{\text{train}(\Theta)} \approx J_{CV}(\Theta)$

Variance: estimation error due to finite data

- High Variance = OverFitting (VO)
- $J_{\text{train}}(\Theta)$ is low and $J_{CV}(\Theta) \gg J_{\text{train}}(\Theta)$

Intuition for the bias-variance trade-off:

- Complex model => sensitive to data => much affected by changes in X => high variance, low bias.
- Simple model => more rigid => does not change as much with changes in X => low variance, high bias.

One of the most important goals in learning: finding a model that is just right in the bias-variance trade-off.

Regularization Effects:

- Small values of λ allow model to become finely tuned to noise leading to large variance => overfitting.
- Large values of λ pull weight parameters to zero leading to large bias => underfitting.

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.

A typical rule of thumb when running diagnostics is:

- More training examples fixes high variance but not high bias.
- Fewer features fixes high variance but not high bias.
- Additional features fixes high bias but not high variance.
- The addition of polynomial and interaction features fixes high bias but not high variance.
- When using gradient descent, decreasing lambda can fix high bias and increasing lambda can fix high variance (lambda is the regularization parameter).
- When using neural networks, small neural networks are more prone to under-fitting and big neural networks are prone to over-fitting. Cross-validation of network size is a way to choose alternatives.

ML:Machine Learning System Design

Prioritizing What to Work On

Different ways we can approach a machine learning problem:

- Collect lots of data (for example "honeypot" project but doesn't always work)
- Develop sophisticated features (for example: using email header data in spam emails)
- Develop algorithms to process your input in different ways (recognizing misspellings in spam).

It is difficult to tell which of the options will be helpful.

Error Analysis

The recommended approach to solving machine learning problems is:

- Start with a simple algorithm, implement it quickly, and test it early.
- Plot learning curves to decide if more data, more features, etc. will help
- **Error analysis:** manually examine the errors on examples in the cross validation set and try to spot a trend.

Error analysis example: Assume that we have 500 emails and our algorithm misclassifies a 100 of them. We could manually analyze the 100 emails and categorize them based on what type of emails they are. We could then try to come up with new cues and features that would help us classify these 100 emails correctly. Hence, if most of our misclassified emails are those which try to steal passwords, then we could find some features that are particular to those emails and add them to our model.

Quick Evaluation

It's important to get error results as a single, numerical value. Otherwise it is difficult to assess your algorithm's performance.

You may need to process your input before it is useful. For example, if your input is a set of words, you may want to treat the same word with different forms (fail/failing/failed) as one word, so must use "stemming software" to recognize them all as one, but inorder to quickly do this a quick to understand error result should be used to determine if our prosed solution makes the ML model better.

Take away:

For classification problems, perform data analysis on the incorrect ML model outputs. Look at the aggregated features across the classification categories to determine what features are important and not important - look at what is being missed. This helps guide where you should focus to create more sophisticated features.

This sort of error analysis, which is the process of manually examining the mistakes that the algorithm makes, can often help guide you to the most fruitful avenues to pursue

Error Metrics for Skewed Classes

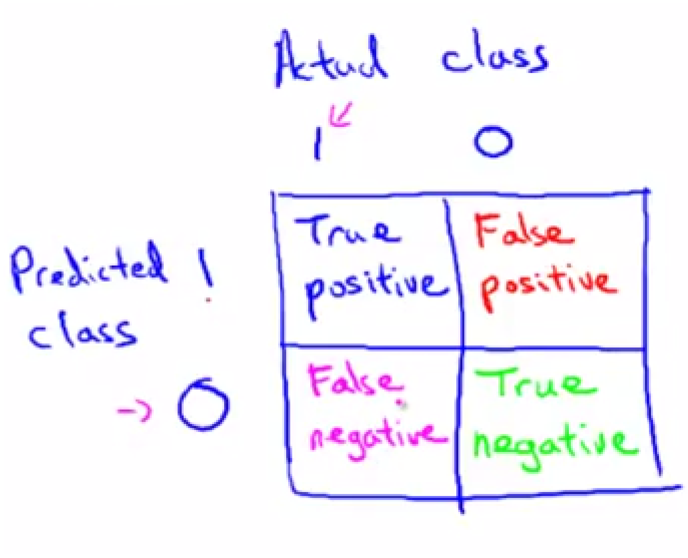
It is sometimes difficult to tell whether a reduction in error is actually an improvement of the algorithm.

- For example: In predicting a cancer diagnoses where 0.5% of the examples have cancer, we find our learning algorithm has a 1% error. However, if we were to simply classify every single example as a 0, then our error would reduce to 0.5% even though we did not improve the algorithm.

This usually happens with **skewed classes**; that is, when our class is very rare in the entire data set.

Or to say it another way, when we have lot more examples from one class than from the other class.

For this we can use **Precision/Recall**.



Precision: of all patients we predicted have cancer, what fraction actually has cancer (in that set)?

$$\frac{\text{True Positives}}{\text{Total number of predicted positives}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False POSITIVES}}$$

$$\frac{\text{True positives}}{\text{\#predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$$

Recall: Of all the patients that actually have cancer, what fraction did we correctly detect as having cancer?

$$\frac{\text{True Positives}}{\text{Total number of actual positives}} = \frac{\text{True Positives}}{\text{True Positives} + \text{False NEGATIVES}}$$

$$\frac{\text{True positives}}{\text{\#actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$$

These two metrics give us a better sense of how our classifier is doing. We want both precision and recall to be high.

If we predict all the patients to be cancer free, $y = 0$, then our **recall** will be $\frac{0}{0+f} = 0$, so despite having a lower error percentage, we can quickly see it has worse recall.

$\text{Accuracy} = \frac{\text{true positive+true negative}}{\text{total population}}$

Note:

- if an algorithm predicts only negatives like it does in one of exercises, the precision is not defined, it is impossible to divide by 0. F1 score will not be defined too.
- By convention, $y = 1$ for the class that is rarer

Trading Off Precision and Recall

Context: Controlling the precision and recall using a threshold

We might want a **confident** (higher threshold) prediction of two classes using logistic regression, predict $y = 1$ only if very confident.

One way is to increase our threshold:

- Predict 1 if: $h_\theta(x) \geq 0.7$
- Predict 0 if: $h_\theta(x) < 0.7$

This way, we only predict cancer if the patient has a 70% chance.

- Doing this, we will have **higher precision** but **lower recall** (refer to the definitions in the previous section).

In the opposite example, we want to avoid missing to many cases of cancer $y = 1$, we can lower our threshold:

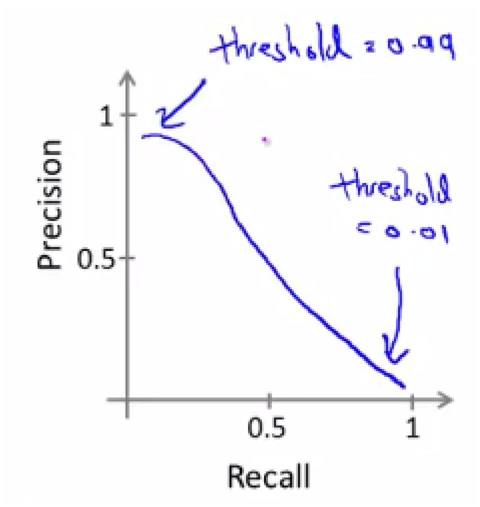
- Predict 1 if: $h_\theta(x) \geq 0.3$
- Predict 0 if: $h_\theta(x) < 0.3$

That way, we get a very **safe** prediction. This will cause **higher recall** but **lower precision**.

Property

- The greater the threshold, the greater the precision and the lower the recall.
- The lower the threshold, the greater the recall and the lower the precision.

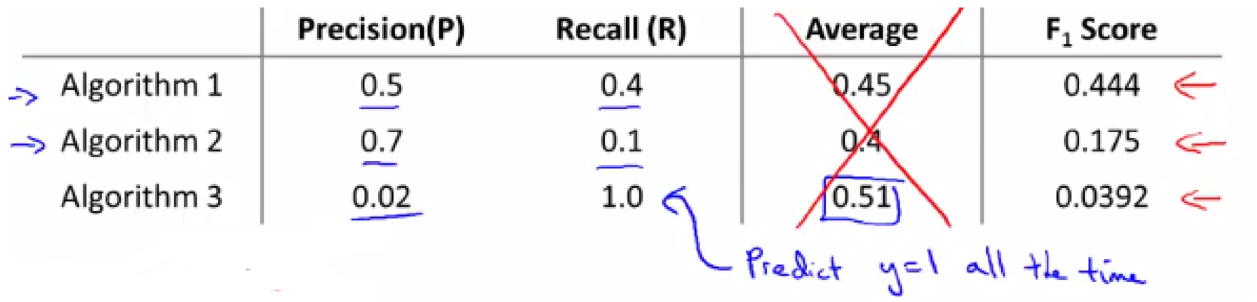
Example diagram below. The precision recall curve can have different forms.



F Score

Context: Single real number evaluation metric.

How do we easily evaluate the following table? *Using the F1 score*



	Precision(P)	Recall (R)	~~Average~~	F_1 Score
Algorithm 1	0.5	0.4	~~0.45~~	0.444
Algorithm 2	0.7	0.1	~~0.4~~	0.175
Algorithm 3	0.02	1.0	~~$\boxed{0.51}$~~	0.0392

Predict y=1 all the time

1) *One way is to take the **average**:*

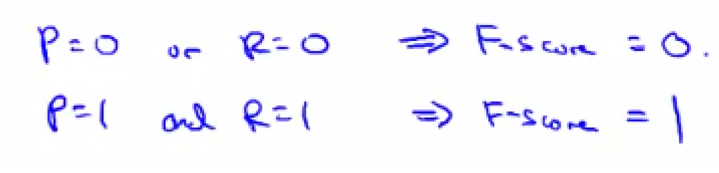
$$\frac{P+R}{2}$$

This does not work well. If we predict all $y = 0$ then that will bring the average up despite having 0 recall. If we predict all examples as $y = 1$, then the very high recall will bring up the average despite having 0 precision.

2) *A better way is to compute the **F Score** (or F1 score):*

$$FScore = 2\frac{PR}{P+R}$$

- Intuition: Its like the average but gives the lower value.
- In order for the F Score to be large, both precision and recall must be large.
- For worst case and best case:



$P=0$ or $R=0 \Rightarrow$ F-score $=0.$

$P=1$ and $R=1 \Rightarrow$ F-score $=1$

NOTE: We want to train precision and recall on the **cross validation set** so as not to bias our test set.

Data for Machine Learning

How much data should we train on?

In certain cases, an "inferior algorithm," if given enough data, can outperform a superior algorithm with less data.

We must choose our features to have **enough** information. A useful test is: Given input x, would a human expert be able to confidently predict y?

Rationale for large data: if we have a **low bias** algorithm (many features or hidden units making a very complex function), then the larger the training set we use, the less we will have overfitting (and the more accurate the algorithm will be on the test set).

Quiz instructions

When the quiz instructions tell you to enter a value to "two decimal digits", what it really means is "two significant digits". So, just for example, the value 0.0123 should be entered as "0.012", not "0.01".

References:

- https://class.coursera.org/ml/lecture/index
- http://www.cedar.buffalo.edu/~srihari/CSE555/Chap9.Part2.pdf
- http://blog.stephenpurpura.com/post/13052575854/managing-bias-variance-tradeoff-in-machine-learning
- http://www.cedar.buffalo.edu/~srihari/CSE574/Chap3/Bias-Variance.pdf