1 Deciding What to Try Next

1.1 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 - sometimes it doesn't help
- Try smaller sets of features
- Try geting additional features
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc)
- Try decreasing λ
- Try increasing λ

Many people will randomly pick one of those options and for example spend 6 months collecting data. A technique to rule out many of those options called machine **learning diagnostics**.

1.1.1 Machine learning diagnostic:

Diagnostic: A test that you can run to gain insight what is/isn't working with a learning algorithm, and gain guidance as to how best to improve its performance.

Diagnostics can take time to implement, but doing so can be a very good use of your time.

1.2 Evaluating a Hypothesis

When we fit the parameters of our learning algorithm, we think about choosing the parameters to minimize the training error. But just because a hypothesis has a low error, doesn't mean it's a good hypothesis and we saw how it can overfit and fail to generalize to new examples not in the training set.

How do we tell if a hypothesis is overfitting. For a small number of features, we can plot but for a large number of features that may be hard or impossible what the hypothesis function looks like.

The standard way to evaluate a hypothesis is as follows. We can split the data in two portions:

- Training set

$$(x^{(1)}, y^{(1)})$$
$$(x^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x^{(m)}, y^{(m)})$$

- Test set

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$(x_{test}^{(2)}, y_{test}^{(2)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

where $m_{test} = number\,of\,test\,examples$

A typical split is 70/30 %.

If an implementation of linear regression without regularization is badly overfitting the training set, we canexpect the training error $J(\Theta)$ to be low and the test set error $J_{test}(\Theta)$ to be high.

If there is any sort of ordering to the data, take a random 70/30 % of the training set.

1.2.1 Training/testing procedure for linear regression

- Learn parameter Θfrom training data (minimizing training error $J(\Theta)$), the 70%.
- Compute test set error:

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

Average squared error as measured on the test set. This is for linear regression and squared error metric.

1.2.2 Training/testing procedure for logistic regression

- Learn parameter Θfrom training data
- Compute test set error:

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

- Misclassification error (or 0/1 misclassification error) - alternative test set metric, may be easier to interpret. 0 - right, 1 - wrong

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

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

Test error is the fraction of examples in test set that the hypothesis has mislabeled.

1.3 Model selection and training/validation/test sets

Suppose we'd like to decide what degree of a polynomial to fit to the data set or choose a regularization parameter λ for learning algo. These are called model selection problems.

1.3.1 Overfitting example

Once parameters Θ_0, Θ_1, ..., $\Theta_4$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.

A hypothesis that fits well for the training set doesn't necessarily fit well for examples not in the training set.

1.3.2 Model selection

Let's say you try to choose what degree polynomial to fit to data:

1. $h_\Theta(x) = \Theta_0 + \Theta_1 x \quad \longrightarrow d = 1,\ \Theta^{(1)} \rightarrow J_{test}(\Theta^{(1)})$
2. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 \quad \longrightarrow d = 2,\ \Theta^{(2)} \rightarrow J_{test}(\Theta^{(2)})$
3. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \ldots + \Theta_3 x^3 \quad \longrightarrow d = 3,\ \Theta^{(3)} \rightarrow J_{test}(\Theta^{(3)})$

$\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots$

10. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \ldots + \Theta_{10} x^{10} \quad \longrightarrow d = 10,\ \Theta^{(10)} \rightarrow J_{test}(\Theta^{(10)})$

$d = degree\, of\, polynomial$

Let's say you want to choose one of those models and also get some estimate your fitted hypothesis would generalize to new examples. You could take the first model and minimize the training error, that would give you some parameter vector $\Theta^{(d)}$. One thing we could do then is take those parameters and look at the test set errors $J_{test}(\Theta^{(d)})$. So take each of the hypotheses with the corresponding parameters and measure the performance on the test set. Then in order to select one of the models, we can select a model with the **lowest test set error**. Let's say for this example it was the fifth order model $J_{test}(\Theta^{(5)}) = \Theta_0 + \ldots + \Theta_5 x^5$.

1.3.3 How well does the model generalize?

We could look at how well the hypothesis had done on the test set. The problem is this will not be a fair estimate of how well it generalizes. $J_{test}(\Theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d=degreee of a polynomial) is fit to test set.

1.3.4 Evaluating your hypothesis

To address the problem in the model selection setting if we want to evaluate a hypothesis, this is what we do. Given a data set, instead of splitting it into 2, split it 3 pieces:

- training set ~ 60%

$$(x^{(1)}, y^{(1)})$$
$$(x^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x^{(m)}, y^{(m)})$$

- cross validation set or validation set (CV) ~ 20%

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$
$$(x_{cv}^{(2)}, y_{cv}^{(2)})$$
$$\vdots$$
$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

- test set ~ 20%

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$(x_{test}^{(2)}, y_{test}^{(2)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

m_{cv}= number of CV examples $(x_{cv}^{(i)}, y_{cv}^{(i)})$

m_{test}= number of test examples

1.3.5 Training/validation/test error

Training error:

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

Cross Validation error:

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

Test error:

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

So when faced with a model selection problem, we're going to use the validation set. We'll take the first model and minimize the cost function and that will give some parameter vector $\Theta$for the linear model. We do the same for quadratic up to 10^{th} degree polynomial (as per ex.). Then instead of testing these on a test set, we'll test them on the cross validation set. Next, we pick the hypothesis with the lowest cross validation error. Let's say it's 4^{th}. We now fit the parameter d using the validation set. The degree of polynomial is no longer fit to the test set. We can use the test set and we can use it to estimate the generalization error of the model that was selected

1. $h_\Theta(x) = \Theta_0 + \Theta_1 x \longrightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(1)} \rightarrow J_{cv}(\Theta^{(1)})$
2. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 \longrightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(2)} \rightarrow J_{cv}(\Theta^{(2)})$
3. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \ldots + \Theta_3 x^3 \longrightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(3)} \rightarrow J_{cv}(\Theta^{(3)})$

$\vdots \qquad \vdots$

10. $h_\Theta(x) = \Theta_0 + \Theta_1 x + \ldots + \Theta_{10} x^{10} \longrightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(10)} \rightarrow J_{cv}(\Theta^{(10)})$

Pick $\Theta_0 + \Theta_1 x_1 + \ldots + \Theta_4 x^4$

Estimate generalization error for test set $J_{test}(\Theta^{(4)})$.

1.4 Diagnosing bias vs. variance

If you run a learning algorithm and it doesn't do as well as you were hoping, almost all the time it will be either an under or over fitting problem - high bias or high variance. It's important to figure out which one or little of both you have.

Bias/variance

High bias - underfit

High variance - overfit

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

Cross Validation error: $J_{cv}(\Theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\Theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$

We can plot training and cross validation errors:

horizontal = degree of polynomial

vertical = error

As we increase the degree of the polynomial, we'll be able to fit our training set better and better. So if $d = 1$ we're going to have a relatively high training error, if a very high degree polynomial, we can have an error close to 0. For cross validation, we may have a very high validation error for $d = 1$, for $d = 2$, we can have a much lower error, because we're finding a much better fit to the data. Conversely if we have a high degree, we're again going to have a high error, because we're overfitting.

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\Theta)$ or $J_{test}(\Theta)$ is high). Is it a bias problem or a variance problem? [7:40]

- **Bias** (uderfit):
  - Cross validation error: high
  - Training errors: high
- **Variance** (overfit):
  - Training error is going to be low
  - Cross validation error will be much higher than training error

1.5 Regularization and bias/variance

1.5.1 Linear regression with regularization

Suppose we're fitting a high order polynomial and we're going to use regularization to prevent overfitting.
Model: $h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 + \Theta_3 x^3 + \Theta_4 x^4$

$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{i=1}^{m}\Theta_j^2$

For a house price to size example, let's consider 3 cases:

- **Large** λ, High bias (**underfit**) - All the Θs will be heavily penalized and we'll end up with most of those parameters ≈ 0 and $h_\Theta(x) \approx \Theta_0$. We'll probably end up with a horizontal straight line that doesn't fit the data.
- **Small** λ, high variance (**overfit**).

Only for an intermediate value of λthat is neither too large nor too small that we end up with Θs that give us a reasonable fit.

Let's define:

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

Previously when we were not using regularization, we defined $J(\Theta)$ to be the same as $J_{train}(\Theta)$ as the cost function but when we're using regularization it'll be defined as average square error on training set without the regularization term. We also define:

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

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

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

Average sum of squares on the training, validation and test sets.

1.5.2 Choosing the regularization parameter λ

Model:

$h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 + \Theta_3 x^3 + \Theta_4 x^4$

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

What we can do is have a range of values for λ. We can go larger than 10

1. Try $\lambda = 0$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(1)} \rightarrow J_{cv}(\Theta^{(1)})$
2. Try $\lambda = 0.01$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(2)} \rightarrow J_{cv}(\Theta^{(2)})$
3. Try $\lambda = 0.02$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(3)} \rightarrow J_{cv}(\Theta^{(3)})$
4. Try $\lambda = 0.04$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(4)} \rightarrow J_{cv}(\Theta^{(4)})$
5. Try $\lambda = 0.08$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(5)} \rightarrow J_{cv}(\Theta^{(5)})$

$\vdots$ $\vdots$

12. Try $\lambda = 10$ $\rightarrow \min_\Theta J(\Theta) \rightarrow \Theta^{(12)} \rightarrow J_{cv}(\Theta^{(12)})$

Pick, say $\Theta^{(5)}$, the one with lowest cross validation error. Test error: $J_{test}(\Theta^{(5)})$

We can take each of these and minimize the cost function $J(\Theta)$. That would give some parameter vector $\Theta^{(1)}$. Then we can use the cross validation set to evaluate the $\Theta^{(i)}$parameters and pick whichever of those models has the lowest cross validation error. Finally we take the parameter and look how well it does on the test set.

1.5.3 Bias/variance as a function of the regularization parameter λ.

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{i=1}^{m}\Theta_j^2$$

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

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

How well do $J_{train}(\Theta)$ and $J_{cv}(\Theta)$ do as we vary λ?

- If λ is small, we're not running regularization and run a risk of overfitting, whereas if $\lambda$is large, then we run a risk of having a bias problem.
- For small values of λ, the regularization term goes away and we're just minimizing the square error. $J_{train}(\Theta)$ will tend to increase as λ increases, because large value of λ corresponds to high bias, where as small corresponds to fitting a high degree polynomial to the data set. For cross validation error, when we have a large λ we may end up with high bias, where as with low λ then we may be overfitting - high variance. An intermediate lambda will work fine.

1.6 Learning curves

Used for sanity check or improve performance of an algorithm, or determine bias or variance errors.

We plot:

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

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

with:

- horizontal axis: m (training set size)
- vertical: error

We'll artificially reduce the number of examples, from 100 to, say, 10, and plot training and cross validation errors.

Let's say we're fitting a quadratic function: $h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2$

For 1, 2, 3 training example, quadratic function will fit perfectly, so $J_{train}(\Theta) \approx 0$

So if the training set is small, the error will be small as well and easy to fit the set.

When the trainig set becomes larger, it becomes harder to fit the hypothesis and the average training error increases, for larger m.

For a small training set, we're not going to be able to generalize well, so the cross validation error will be large and will decrease the more training examples we test on.

1.6.1 High bias - underfit

The $J_{cv(}(\Theta)$ will start with a high error for small m and decrease slightly as m becomes larger, however it'll flatten out pretty fast.

The training error will be small for small m and will increase finally flattening out for large ms, close to the cross validation error. The performance of J_{train} and J_{cv} will be similar. However, both J_{train} and J_{cv} are ultimately high.

If a learning algorithm is suffering from high bias, gettig more training data will not (by itself) help much. e.g. you'll get a straight line for quadratic function fitting examples regardless of whether you have 5 training examples or 100.

1.6.2 High variance - overfit

If you have a very small trainig set and fitting high order polynomial and we're using a small λ, we'll end up with an overfit, $J_{train}(\Theta)$ will be small. As the training size increases, the error will still be pretty low. The cross validation error will remain high even if we get a high training set because the hypothesis is overfitting. The indicative diagnostic that we have a high variance problem is that we have a large gap between $J_{cv}(\Theta)$ and $J_{train}(\Theta)$. However the training and cross validation errors are converging. If we were to extrapolate a graph to the right, cross validation would keep going down and training error up.

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

1.7 Deciding What to Do Next Revisited

1.7.1 Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unaccpetably large errors in its prediction. What should you try next?

- Get **more training examples**: **fixes high variance**
- Try **smaller sets of features**: **fixes high variance**
- Try **getting additional features**: may **fix high bias**
- Try **adding polynomial features** (x_1^2, x_2^2, $x_1 x_2$, *etc*): **fixes high bias**
- Try **decreasing** λ: **fixes high bias**
- Try **increasing** λ: **fixes high variance**

1.7.2 Neural networks and overfitting

"Small" neural network (fewer parameters; more prone to underfitting), computationally cheaper.

Alternatively, "large" neural network (more parameters; more prone to overfitting). Computationally more expensive. Use regularization, λ to address overfitting. Often using a large network is more effective than using a smaller one. The number of hidden layers - try training several and see which performs best for the cross validation set.

Suppose you fit a neural network with one hidden layer to a training set. You find that the cross validation error $J_{cv}(\Theta) \gg J_{train}(\Theta)$. Increasing the number of hidden units is unlikely to help, because it is currently suffering from high variance.