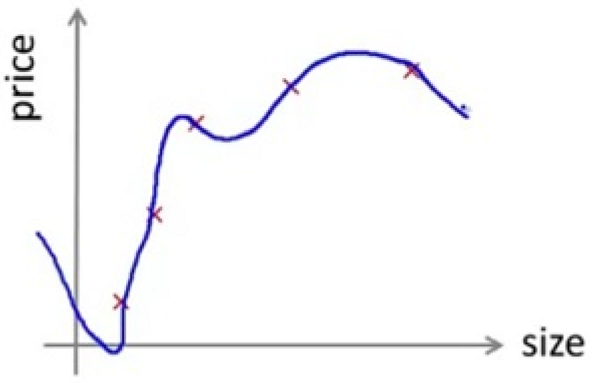
Evaluating a Hypothesis ¶

When we fit the parameters of our learning algorithm we think about choosing the parameters to minimize the training error. One might think that getting a really low value of training error might be a good thing, but we have already seen that just because a hypothesis has low training error, that doesn't mean it is necessarily a good hypothesis.

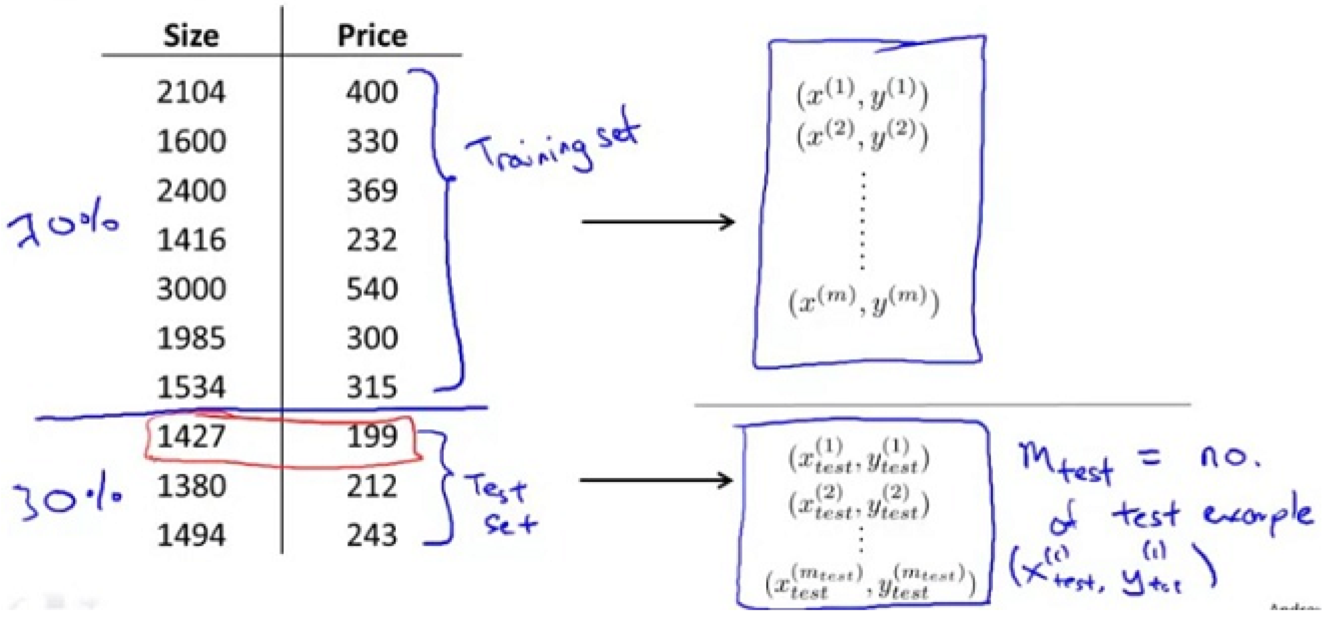


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

And we've already seen the example of how a hypothesis can overfit. And therefore **fail to generalize the new examples not in the training set**. So how do we tell if the hypothesis might be overfitting?

In this simple example we could plot the hypothesis $h_\theta(x)$ and just see what was going on. But in general for problems with more features than just one feature ($x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_{100}$), for problems with a large number of features like these it becomes **hard** or may be **impossible** to plot what the hypothesis looks like and so we need some other way to evaluate our hypothesis.

The standard way to evaluate a learned hypothesis is as follows. Suppose we have a data set, where we have 10 training examples.



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
$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
$(x^{(m)}, y^{(m)})$
$(x^{(1)}_{test}, y^{(1)}_{test})$
$(x^{(2)}_{test}, y^{(2)}_{test})$
$(x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$
m_{test} = no. of test example
$(x^{(i)}_{test}, y^{(i)}_{test})$

In order to make sure we can evaluate our hypothesis, what we are going to do is split the data we have into two portions. The first portion is going to be our usual **training set** and the second portion is going to be our **test set**, and a pretty typical split of this all the data we have into a training set and test set might be around say a 70% and 30% split (if there is some sort of order in the data, it would be better to send a random 70% of our data to the training set and a random 30% of our data to the test set).

Video Question: Suppose an implementation of linear regression (without regularization) is badly overfitting the training set. In this case, we would expect:

> The training error $J(\theta)$ to be low and the test error $J_{\text{test}}(\theta)$ to be high

- The training error $J(\theta)$ to be low and the test error $J_{\text{test}}(\theta)$ to be low
- The training error $J(\theta)$ to be high and the test error $J_{\text{test}}(\theta)$ to be low
- The training error $J(\theta)$ to be high and the test error $J_{\text{test}}(\theta)$ to be high

Training/testing procedure for linear regression

- Learn parameter θ from training data (minimizing training error $J(\theta)$, where $J(\theta)$ was defined using that 70% of all the data we have).
- Compute test set error (we're going to denote the test error like $J_{test}(x)$):

So what we do is take our parameter theta that we have learned from the training set, and plug it in the test error function $J_{test}(\theta)$ and compute our test set error.

- $J_{test}(\theta)$ = average square error as measured on the test set

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

This is the definition of the **test set error**, if we are using linear regression and using the squared error metric.

Training/testing procedure for logistic regression

- Learn parameter θ from training data (the same, learn using 70% of the data, test with the remaining 30%)
- Compute test set error:

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

It's the same objective function as we always use but we just logistic regression, except that now is define using our m_{test}, test examples. Sometimes there is an alternative test sets metric that might be easier to interpret, and that's the misclassification error.

- Misclassification error (0/1 misclassification error):

We define the error as follows:

$$err(h_\theta(x), y) = \begin{cases} 1 & \left.\begin{array}{l} \text{if } h_\theta(x) \geq 0.5, \quad y = 0 \\ \text{or if } h_\theta(x) < 0.5, \quad y = 1 \end{array}\right\} \text{error} \\ 0 & \text{otherwise} \end{cases}$$

We could then define the test error, using the misclassification error metric:

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

Summary

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

1. For linear regression: $J_{test}(\Theta) = \dfrac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\Theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$
2. For classification ~ Misclassification error (aka 0/1 misclassification error):

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

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

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

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

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