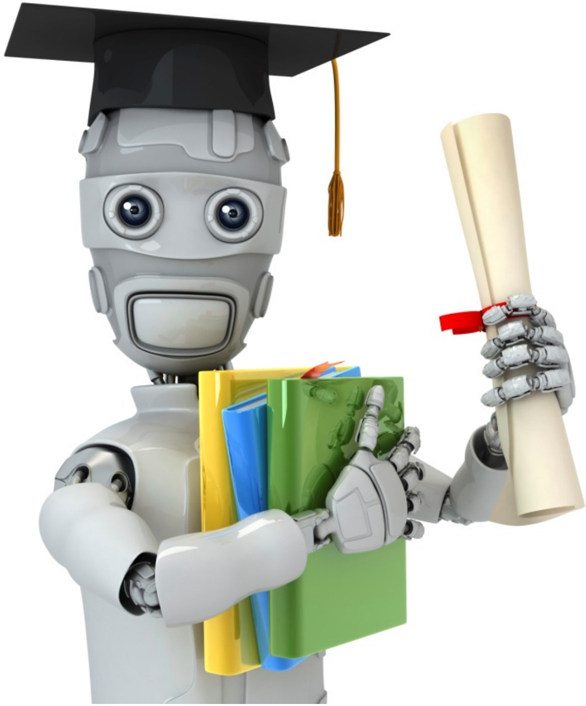


# Advice for applying machine learning

## Deciding what to try next

Machine Learning

What I would like to do is make sure that if you are developing machine learning systems, that you know how to choose one of the most promising avenues to spend your time pursuing. And on this and the next few videos I'm going to give a number of practical suggestions, advice, guidelines on how to do that.

**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?

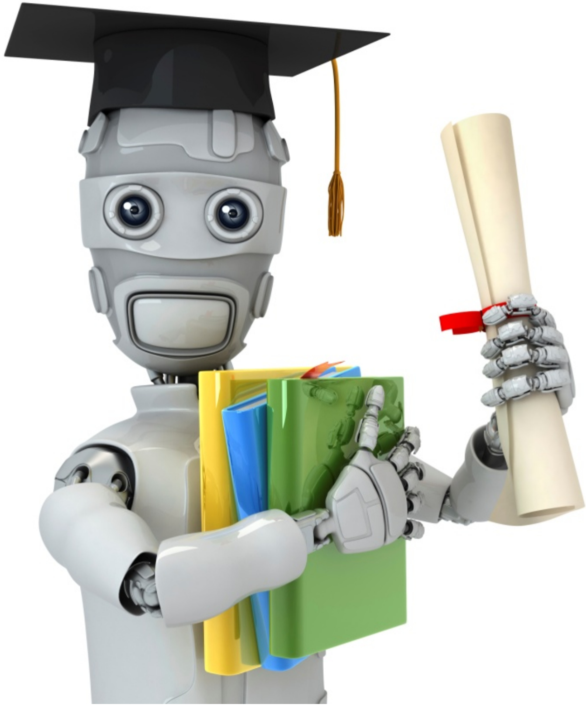
- Get more training examples — But sometimes getting more training data doesn't actually help
- Try smaller sets of features — to prevent overfitting — $x_2, \dots, x_{100}$
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc.})$
- Try decreasing $\lambda$
- Try increasing $\lambda$

Fortunately, there is a pretty simple technique that can let you very quickly rule out half of the things on this list as being potentially promising things to pursue. And there is a very simple technique, that if you run, can easily rule out many of these options,

**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.

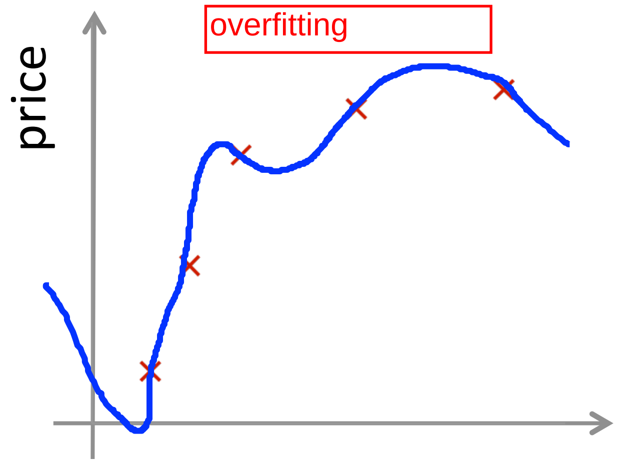


Machine Learning

# Advice for applying machine learning

## Evaluating a hypothesis

# Evaluating your hypothesis




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

just because a hypothesis has low training error, that doesn't mean it is necessarily a good hypothesis. 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

Fails to generalize to new examples not in training set.

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$
$\vdots$
$x_{100}$

So how do you tell if the hypothesis might be overfitting. In this simple example we could plot the hypothesis h of x and just see what was going on. But in general for problems with more features than just one feature, 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.

# Evaluating your hypothesis

Dataset:

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

random 70%

70%

Training set

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

random 30%

30%

Test set

$$(x^{(1)}_{test}, y^{(1)}_{test})$$
$$(x^{(2)}_{test}, y^{(2)}_{test})$$
$$\vdots$$
$$(x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$$

$m_{test}$ = no. of test example

$(x^{(1)}_{test}, y^{(1)}_{test})$

Andrew Ng

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**

Correct

- 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 $\theta$ from training data (minimizing training error $J(\theta)$)

70%

- Compute test set error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left( h_\theta(x^{(i)}_{test}) - y^{(i)}_{test} \right)^2$$

# Training/testing procedure for logistic regression

- Learn parameter $\theta$ from training data
- Compute test set error:

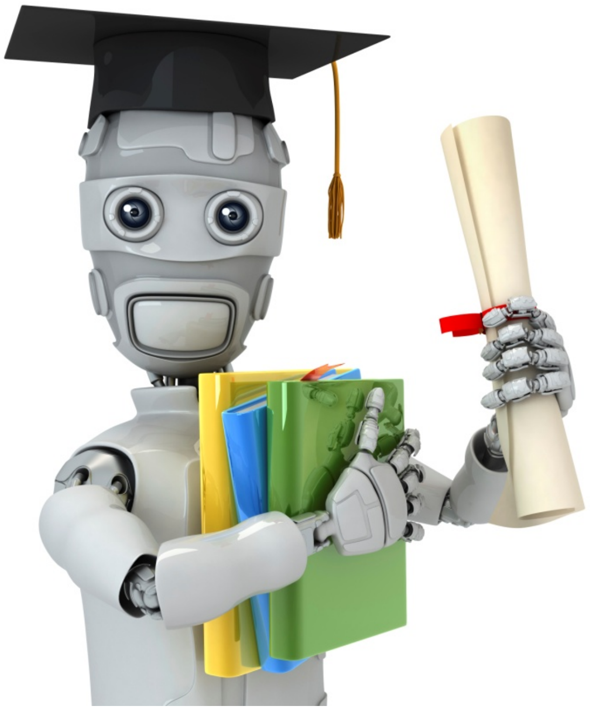
$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$$

- Misclassification error (0/1 misclassification error):

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

error

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



Machine Learning

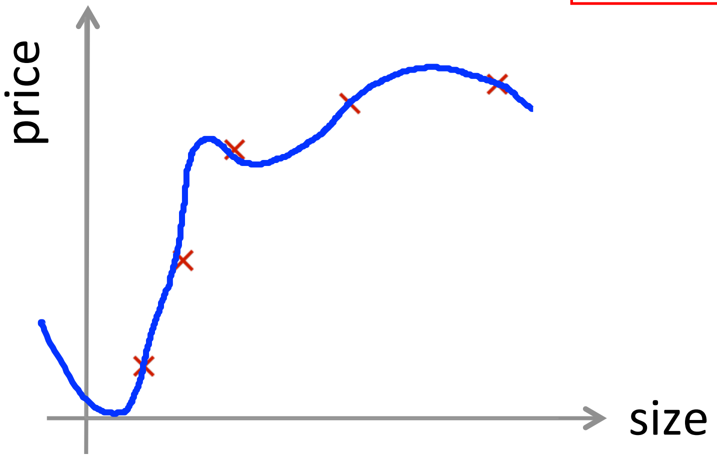
# Advice for applying machine learning

## Model selection and training/validation/test sets

Training Set: this data set is used to adjust the weights on the neural network.
Validation Set: this

https://en.wikipedia.org/wiki/Generalization_error
.....................
In supervised learning applications in machine learning and statistical learning theory, generalization error (also known as the out-of-sample error) is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

# Overfitting example




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

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.

# Model selection

$d$ = degree of polynomial

degree

$d=1$ 1. $h_\theta(x) = \theta_0 + \theta_1 x \rightarrow \Theta^{(1)} \rightarrow J_{test}(\Theta^{(1)})$

$d=2$ 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow \Theta^{(2)} \rightarrow J_{test}(\Theta^{(2)})$

$d=3$ 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3 \rightarrow \Theta^{(3)} \rightarrow J_{test}(\Theta^{(3)})$

$\vdots$

$d=10$ 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \rightarrow \Theta^{(10)} \rightarrow J_{test}(\Theta^{(10)})$

$\boxed{\Theta_0, \Theta_1, \ldots}$

Choose $\boxed{\theta_0 + \ldots \theta_5 x^5}$

How well does the model generalize? Report test set error $J_{test}(\theta^{(5)})$.

$\Theta^{(5)}$

Problem: $J_{test}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter ($d$ = degree of polynomial) is fit to test set.

And let's just say for this example that I ended up choosing the fifth order polynomial.
So, this seems reasonable so far.
But now let's say I want to take my fifth hypothesis, this, this, fifth order model,
and let's say I want to ask, how well does this model generalize?
One thing I could do is look at how well my fifth order polynomial hypothes is had done on my test set.
But the problem is this will not be a fair estimate of how well my hypothesis generalizes.
And the reason is what we've done is we've fit this extra parameter d,
that is this degree of polynomial. And what fits that parameter d,using the test set, namely,
we chose the value of d that gave us the best possible performance on the test set.
And so, the performance of my parameter vector theta5, on the test set,
that's likely to be an overly optimistic estimate of generalization error.
Right, so, that because I had fit this parameter d to my test set is no longer fair to
evaluate my hypothesis on this test set, because I fit my parameters to this test set,
I've chose the degreed of polynomial using the test set.

# Evaluating your hypothesis

Dataset:

| | Size | Price |
|---|---|---|
| 60% | 2104 | 400 |
| | 1600 | 330 |
| | 2400 | 369 |
| | 1416 | 232 |
| | 3000 | 540 |
| | 1985 | 300 |
| 20% | 1534 | 315 |
| | 1427 | 199 |
| 20% | 1380 | 212 |
| | 1494 | 243 |

Training set

training set

Cross validation set (CV)

cross validation

test set

test validation

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

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

$m_{cv}$ = no. of cv example

$(x_{cv}^{(i)}, y_{cv}^{(i)})$

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

$m_{test}$

https://stackoverflow.com/questions/2976452/whats-is-the-difference-between-train-validation-and-test-set-in-neural-networ

Once you're finished training, then you run against your testing set and verify that the accuracy is sufficient.

**Training Set**: this data set is used to adjust the weights on the neural network.

or for model selection ?

**Validation Set**: this data set is used to minimize overfitting. You're not adjusting the weights of the network with this data set, you're just verifying that any increase in accuracy over the training data set actually yields an increase in accuracy over a data set that has not been shown to the network before, or at least the network hasn't trained on it (i.e. validation data set). If the accuracy over the training data set increases, but the accuracy over then validation data set stays the same or decreases, then you're overfitting your neural network and you should stop training.

**Testing Set**: this data set is used only for testing the final solution in order to confirm the actual predictive power of the network.

# Train/validation/test error

Training error:

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

$J(\theta)$

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:

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

# Model selection

$d=1$ 1. $h_\theta(x) = \theta_0 + \theta_1 x \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$

$d=2$ 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$

$d=3$ 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3 \rightarrow \theta^{(3)}$

$\vdots \qquad \vdots \qquad \vdots \qquad J_{cv}(\theta^{(4)})$

$d=10$ 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \rightarrow \theta^{(10)} \rightarrow J_{cv}(\theta^{(10)})$

And then I'm going to pick the hypothesis with the lowest cross validation error

$d=4$

Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4 \leftarrow$

Estimate generalization error for test set $J_{test}(\theta^{(4)}) \leftarrow$

Consider the model selection procedure where we choose the degree of polynomial using a cross validation set. For the final model (with parameters $\theta$), we might generally expect $J_{CV}(\theta)$ To be lower than $J_{test}(\theta)$ because:

- ◉ An extra parameter ($d$, the degree of the polynomial) has been fit to the cross validation set.

  **Correct**

- ○ An extra parameter ($d$, the degree of the polynomial) has been fit to the test set.
- ○ The cross validation set is usually smaller than the test set.
- ○ The cross validation set is usually larger than the test set.

ref:https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

.................................................................

In statistics and machine learning, the bias–variance tradeoff (or dilemma) is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:

.................................................................

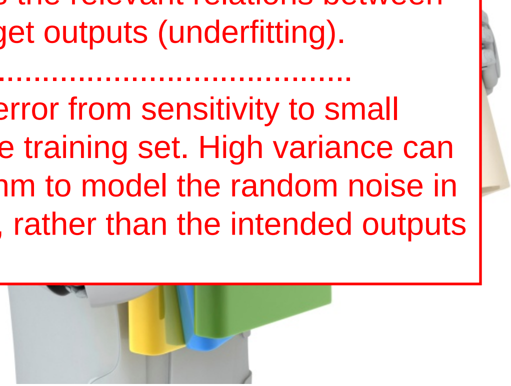
The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

.................................................................

The variance is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

# Advice for applying machine learning

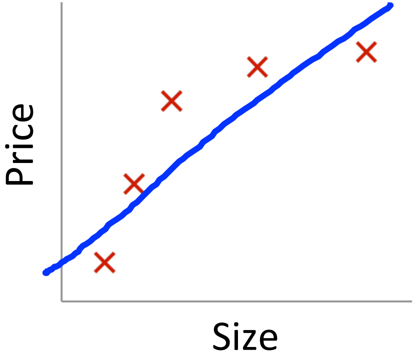
## Diagnosing bias vs. variance



Machine Learning

If you run the learning algorithm and it doesn't do as well as you are hoping, almost all the time it will be because you have either a high bias problem or a high variance problem. In other words they're either an underfitting problem or an overfitting problem.

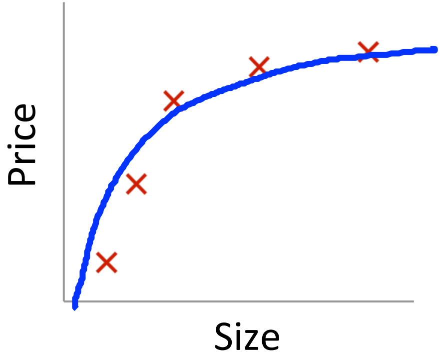
# Bias/variance




$\theta_0 + \theta_1 x$

High bias
(underfit)

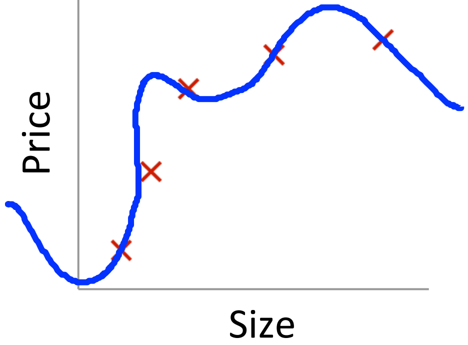
$d=1$




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

"Just right"

$d=2$




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

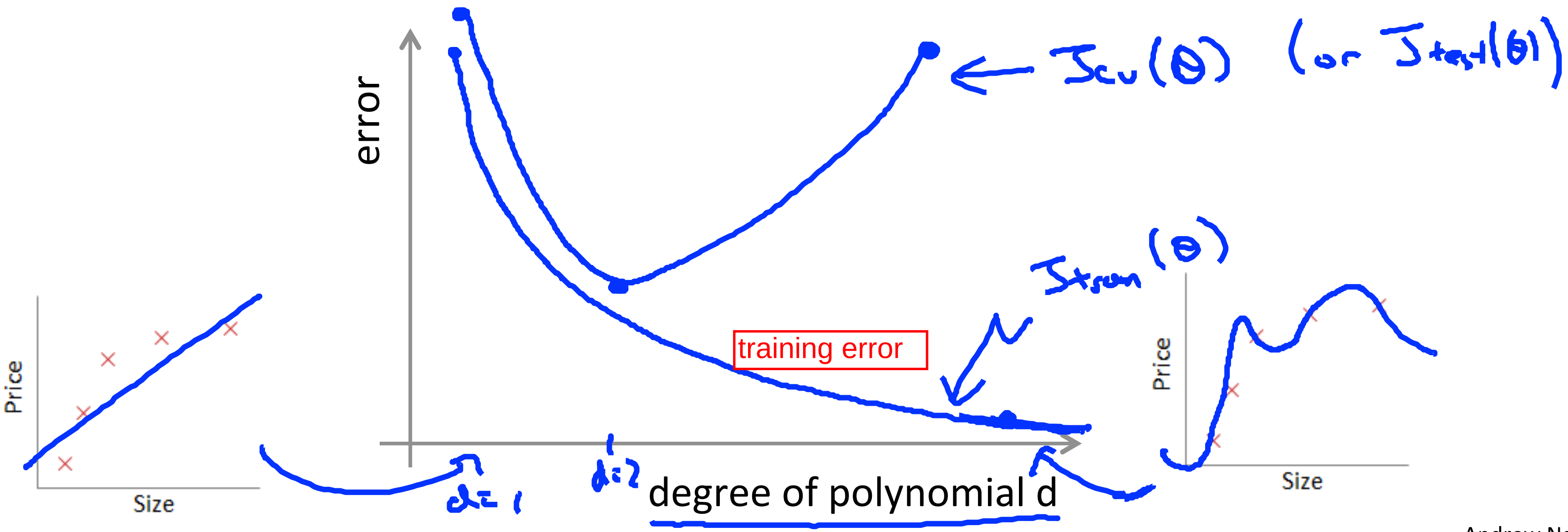
High variance
(overfit)

$d=4$

# Bias/variance

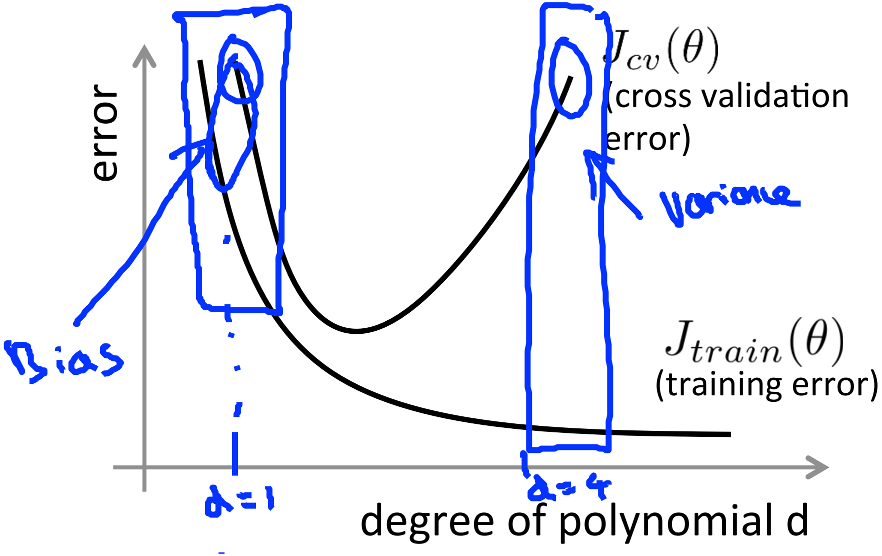
Training error: $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$ (or $J_{test}(\theta)$)



## Diagnosing bias vs. variance

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?




**Bias (underfit):** Jtran and Jcv are high

→ $J_{train}(\theta)$ will be high

$J_{cv}(\theta) \approx J_{train}(\theta)$

**Variance (overfit):**

→ $J_{train}(\theta)$ will be low

$J_{cv}(\theta) \gg J_{train}(\theta)$

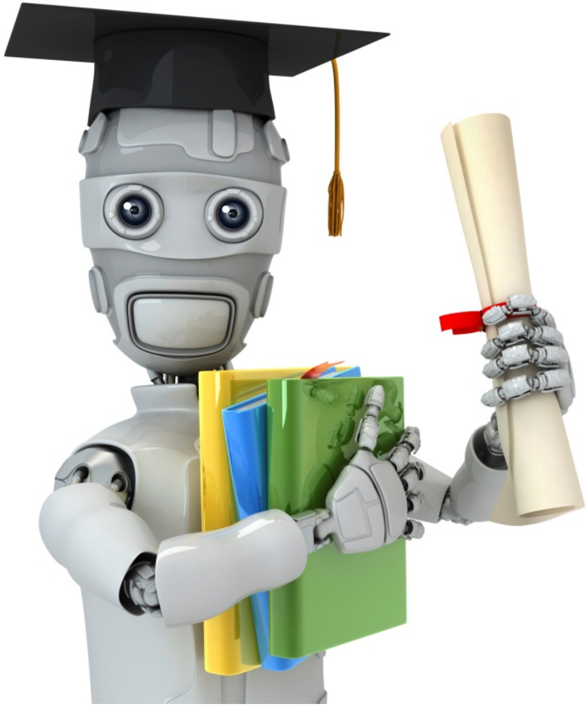
$\gg$

Suppose you have a classification problem. The (misclassification) error is defined as $\frac{1}{m}\sum_{i=1}^{m} \mathrm{err}(h_\theta(x^{(i)}), y^{(i)})$, and the cross validation (misclassification) error is similarly defined, using the cross validation examples $(x_{\text{cv}}^{(1)}, y_{\text{cv}}^{(1)}), \ldots, (x_{\text{cv}}^{(m_{\text{cv}})}, y_{\text{cv}}^{(m_{\text{cv}})})$. Suppose your training error is 0.10, and your cross validation error is 0.30. What problem is the algorithm most likely to be suffering from?

- ○ High bias (overfitting)
- ○ High bias (underfitting)
- ◉ High variance (overfitting)

**Correct**

- ○ High variance (underfitting)



# Advice for applying machine learning

## Regularization and bias/variance

Machine Learning

You've seen how regularization can help prevent over-fitting. But how does it affect the bias and variances of a learning algorithm? In this video I'd like to go deeper into the issue of bias and variances and talk about how it interacts with and is affected by the regularization of your learning algorithm.

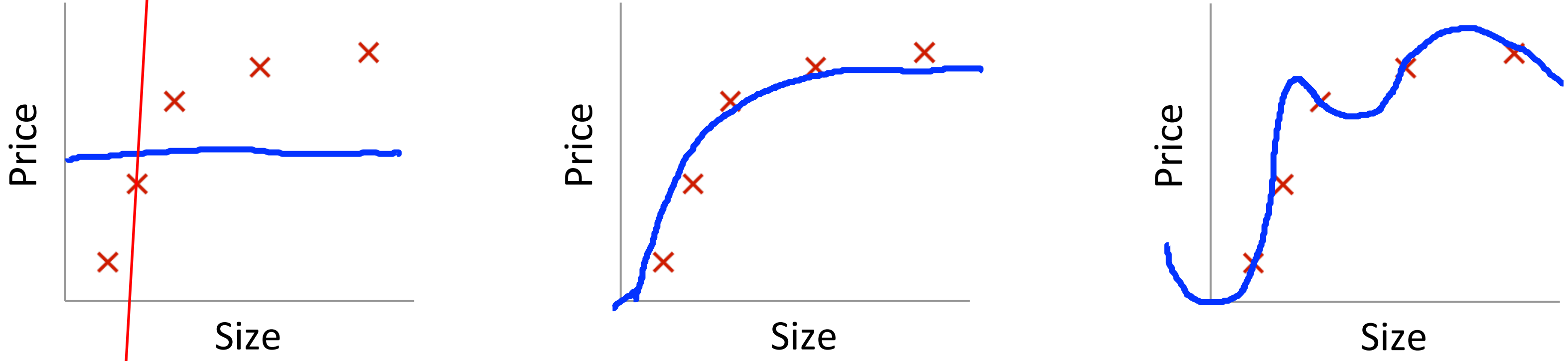
# Linear regression with regularization

So we have this regularization term to try to keep the values of the prem to small. And as usual, the regularizations comes from J = 1 to m, rather than j = 0 to m.

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

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

use regularization to prevent it from overfitting




| Large $\lambda$ | Intermediate $\lambda$ | Small $\lambda$ |
| --- | --- | --- |
| High bias (underfit) | "Just right" | High variance (overfit) |
| $\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0$ | | $\lambda = 0$ |
| $h_\theta(x) \approx \theta_0$ | | |

large lambda, smaller theta 1 to theta n, large theta 0 => bias

Andrew Ng

how can we automatically choose a good value for the regularization parameter?

# Choosing the regularization parameter $\lambda$

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

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

$J(\theta)$

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

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

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

$J_{train}$

$J_{cv}$

$J_{test}$

In the figure above, we see that as $\lambda$ increases, our fit becomes more rigid. On the other hand, as $\lambda$ approaches 0, we tend to over overfit the data. So how do we choose our parameter $\lambda$ 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. λ∈{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 $\lambda$s and for each $\lambda$ go through all the models to learn some $\Theta$.
4. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ **without** regularization or λ = 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.

# Choosing the regularization parameter $\lambda$

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

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

use regularization to train theta

use the trained theta to compute J_train and J_cv without regularization

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

$\vdots$

12. Try $\lambda = 10$ $\longrightarrow \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$

10.24

usually use 10.24

And I would then pick whichever one of these 12 models gives me the lowest error on the trans validation set

Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$

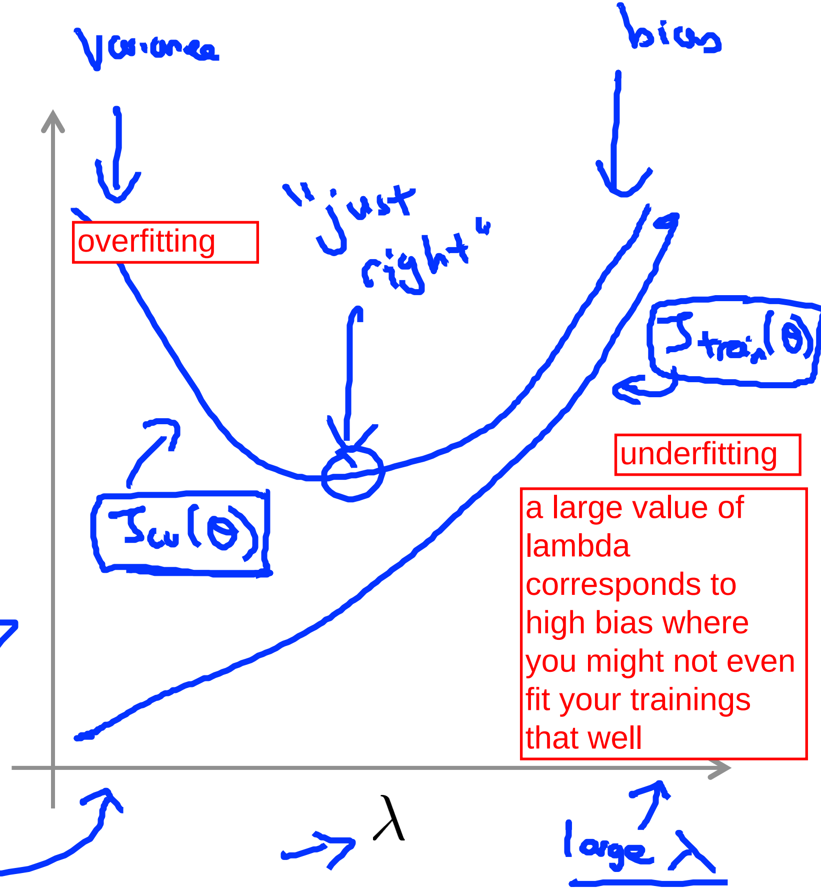
Andrew Ng

# Bias/variance as a function of the regularization parameter $\lambda$

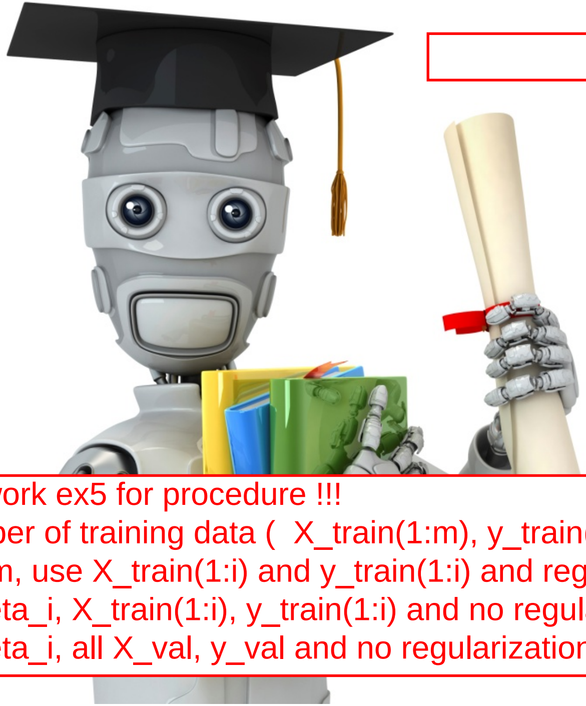
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2}$$

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

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




for small values of lambda, you're not regularizing,you can really fit a very high degree polynomial to your data, Jtrain can be small



# Advice for applying machine learning

## Learning curves

see homework ex5 for procedure !!!
let m=number of training data (  X_train(1:m), y_train(1:m)  )
let i = 1 to m, use X_train(1:i) and y_train(1:i) and regularization (lambda) to train a theta_i
use this theta_i, X_train(1:i), y_train(1:i) and no regularization(lambda=0) to evaluate J_train
use this theta_i, all X_val, y_val and no regularization(lambda=0) to evaluate J_val

Learning curves is often a very useful thing to plot. If either you wanted to sanity check that your algorithm is working correctly, or if you want to improve the performance of the algorithm.
And learning curves is a tool that I actually use very often to try to diagnose if a physical learning algorithm may be suffering from bias, sort of variance problem or a bit of both.

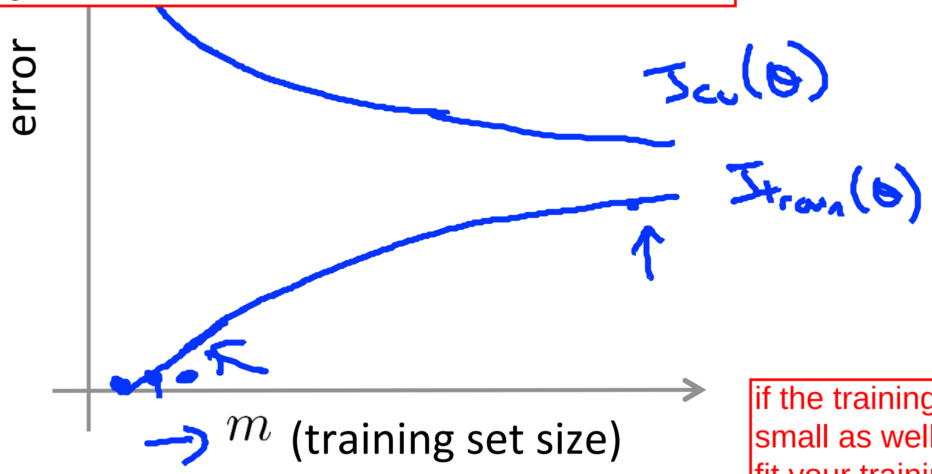
Here if I set M=3, say, and I train on only three examples,for this figure I am going to measure my training error only on the three examples that actually fit my data

# Learning curves

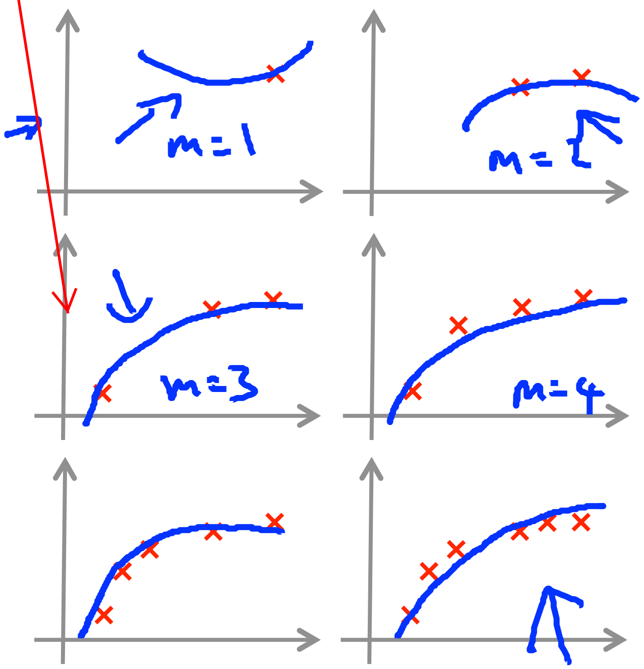
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

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

training and validation error does not include the regularization term!



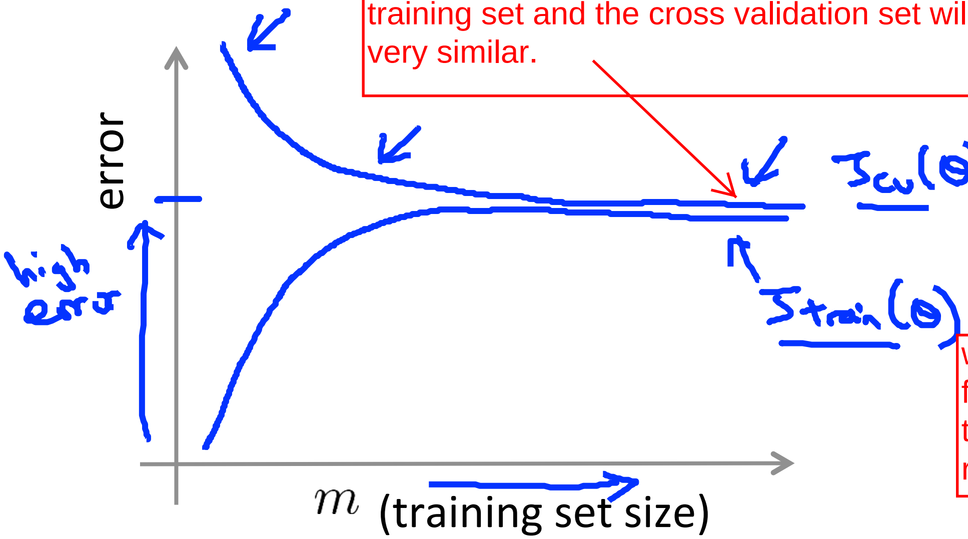

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

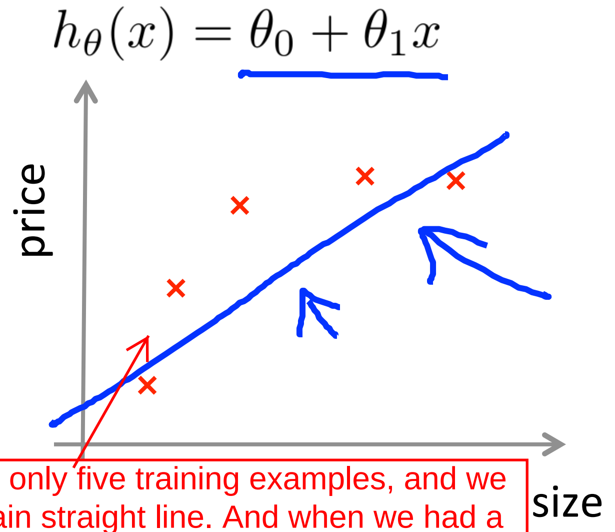



if the training set size is small then the training error is going to be small as well. because a small training set is going to be very easy to fit your training set very well

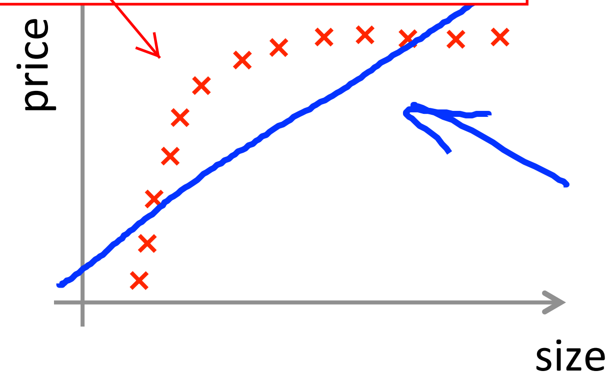
# High bias

the high bias case:training error will end up close to the cross validation error, because you have so few parameters and so much data, at least when m is large. The performance on the training set and the cross validation set will be very similar.




we had only five training examples, and we fill certain straight line. And when we had a ton more training data, we still end up with roughly the same straight line



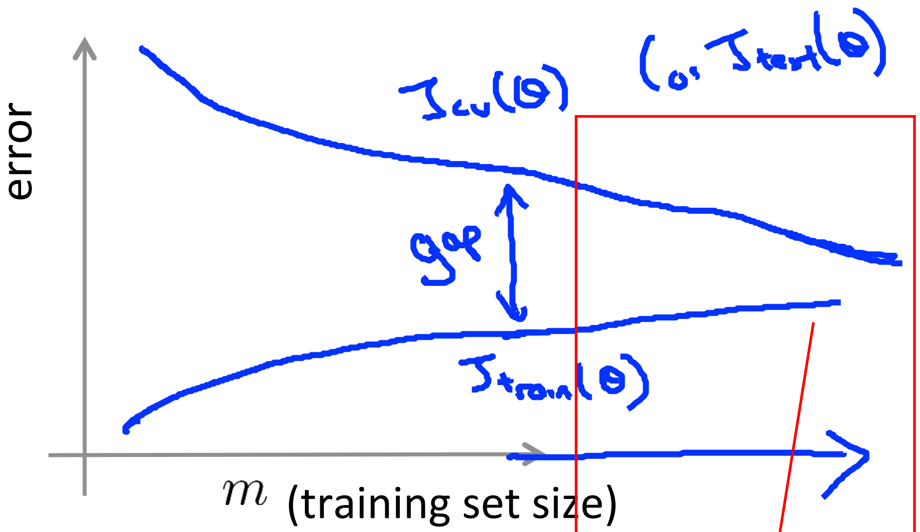

If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

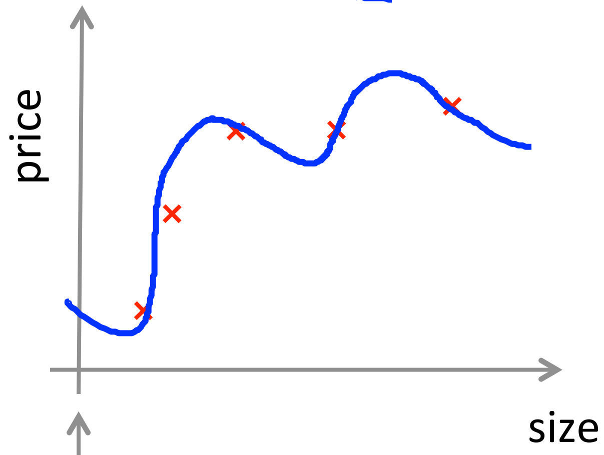
Andrew Ng

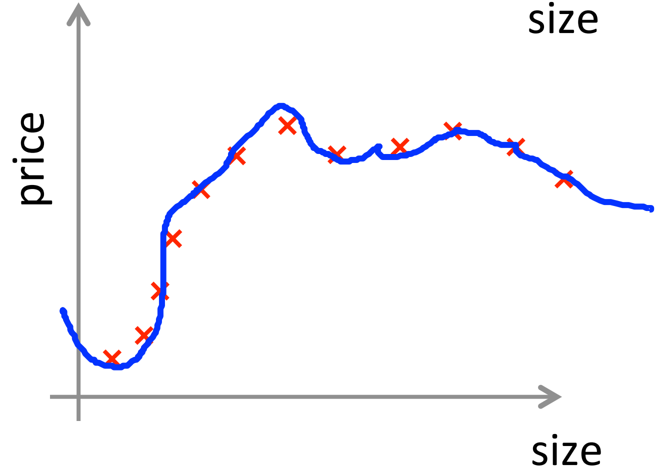
# High variance

$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$$

(and small $\lambda$)




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

Andrew Ng

In which of the following circumstances is getting more training data likely to significantly help a learning algorithm's performance?

- [ ] Algorithm is suffering from high bias.

**Un-selected is correct**

- [x] Algorithm is suffering from high variance.

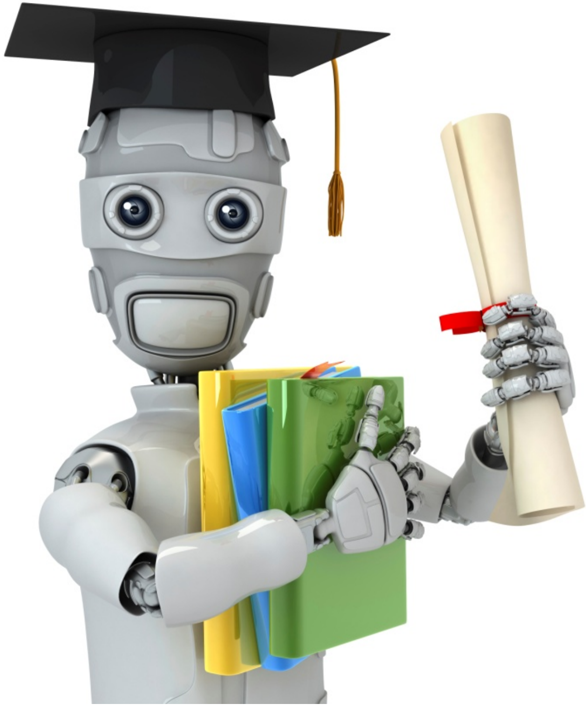
**Correct**

- [x] $J_{\text{CV}}(\theta)$ (cross validation error) is much larger than $J_{\text{train}}(\theta)$ (training error).

**Correct**

- [ ] $J_{\text{CV}}(\theta)$ (cross validation error) is about the same as $J_{\text{train}}(\theta)$ (training error).

**Un-selected is correct**



Machine Learning

# Advice for applying machine learning

## Deciding what to try next (revisited)

fixing high bias problems:adding extra features(polynomial feature) usually because your hypothesis is too simple, and so we want to get additional features to make our hypothesis better able to fit training set

**Debugging a learning algorithm:**

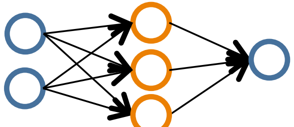
Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples → fixes high variance
  not helping high bias
- Try smaller sets of features → fixes high variance
  not helping high bias
- Try getting additional features → fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1 x_2, \text{etc})$ → fixes high bias.
- Try decreasing $\lambda$ → fixes high bias
- Try increasing $\lambda$ → fixes high variance

# Neural networks and overfitting

using a single hidden layer is a reasonable default, but if you want to choose the number of hidden layers, one other thing you can try is find yourself a training cross-validation, and test set split and try training neural networks with 1, 2, or 3 hidden layers and see which performs best on the cross-validation sets....

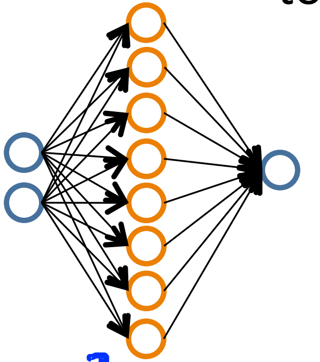
"Small" neural network (fewer parameters; more prone to underfitting)

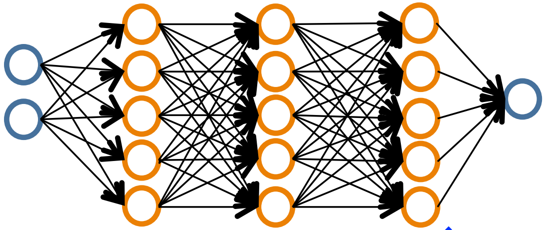


Computationally cheaper

a network like this might have relatively few parameters and be more prone to underfitting.The main advantage of these small neural networks is that the computation will be cheaper.

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





Computationally more expensive.

Use regularization ($\lambda$) to address overfitting.

$J_{cv}(\Theta)$

Andrew Ng

Suppose you fit a neural network with one hidden layer to a training set. You find that the cross validation error $J_{CV}(\theta)$ is much larger than the training error $J_{\text{train}}(\theta)$. Is increasing the number of hidden units likely to help?

- Yes, because this increases the number of parameters and lets the network represent more complex functions.
- Yes, because it is currently suffering from high bias.
- No, because it is currently suffering from high bias, so adding hidden units is unlikely to help.
- No, because it is currently suffering from high variance, so adding hidden units is unlikely to help.

Correct

because its overfitting and adding more layer/unit is not gonna help ?

?