# 10. Advice for applying Machine Learning:

**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?

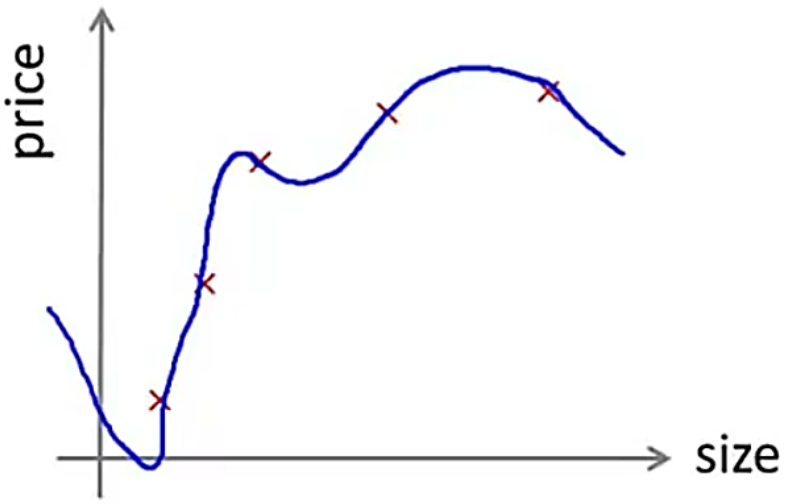
- Get more training examples
- Try smaller sets of features $x_1, x_2, x_3, \ldots, x_{100}$
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc.})$
- Try decreasing $\lambda$
- Try increasing $\lambda$

## 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.

# EVALUATING A HYPOTHESIS:




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

Due to overfitting, the learned hypothesis may fail to generalize to new examples not in training set

## One way to evaluate the hypothesis:

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.**

**Evaluating your hypothesis**

Dataset:

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

70%

30%

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

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

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

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

Andrew

- Train the model only using the 70% of input dataset, and use the remaining 30% as test set: and check if the predictions match the given values
- When dividing the dataset into training set and test set, make sure the data is not ordered in any way. Randomize it. Random is better.

**Procedure:**

## Training/testing procedure for linear regression

- Learn parameter $\theta$ from training data (minimizing training error $J(\theta)$) — 70%
- Compute test set error:

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

## Training/testing procedure for logistic regression

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

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

- Misclassification error (0/1 misclassification error):

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

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

**Err is 1→** if o/p was actually 1 but hypothesis gave 0 OR
o/p was 0 but hypothesis gave 1

Otherwise (when hypothesis and o/p match), its 0.

**Test error** – Gives us the proportion of the test data that was misclassified

---

## HOW TO SELECT MODEL:

### Evaluating your hypothesis

Dataset:

| | Size | Price | |
|---|---|---|---|
| 60% | 2104 | 400 | Training set |
| | 1600 | 330 | |
| | 2400 | 369 | |
| | 1416 | 232 | |
| | 3000 | 540 | |
| | 1985 | 300 | |
| 20% | 1534 | 315 | Cross validation set (CV) |
| | 1427 | 199 | |
| 20% | 1380 | 212 | test set |
| | 1494 | 243 | |

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

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

$m_{cv}$ = no. of CV examples $(x_{cv}^{(i)}, y_{cv}^{(i)})$

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

## Train/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:

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

We might generally expect JCV(θ) To be lower than $J_{test}$ (θ) because:

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

## Model selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

⋮

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

- Here, superscript is the no of model chosen (degree of polynomial)

➔ Pick the model with lowest $J_{CV}$(Θ): let's say d=4 is the one

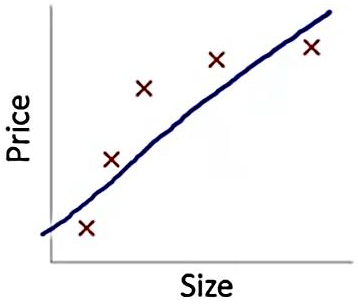
Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4$

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

# BIAS AND VARIANCE:

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

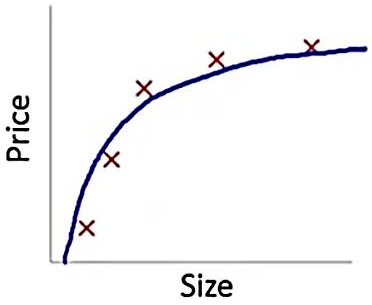
We need to distinguish whether bias or variance is the problem contributing to bad predictions.




$\theta_0 + \theta_1 x$

High bias (underfit)

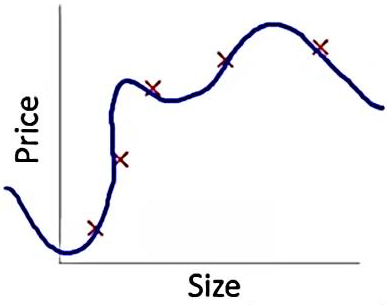
$d=1$




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

"Just right"

$d=2$



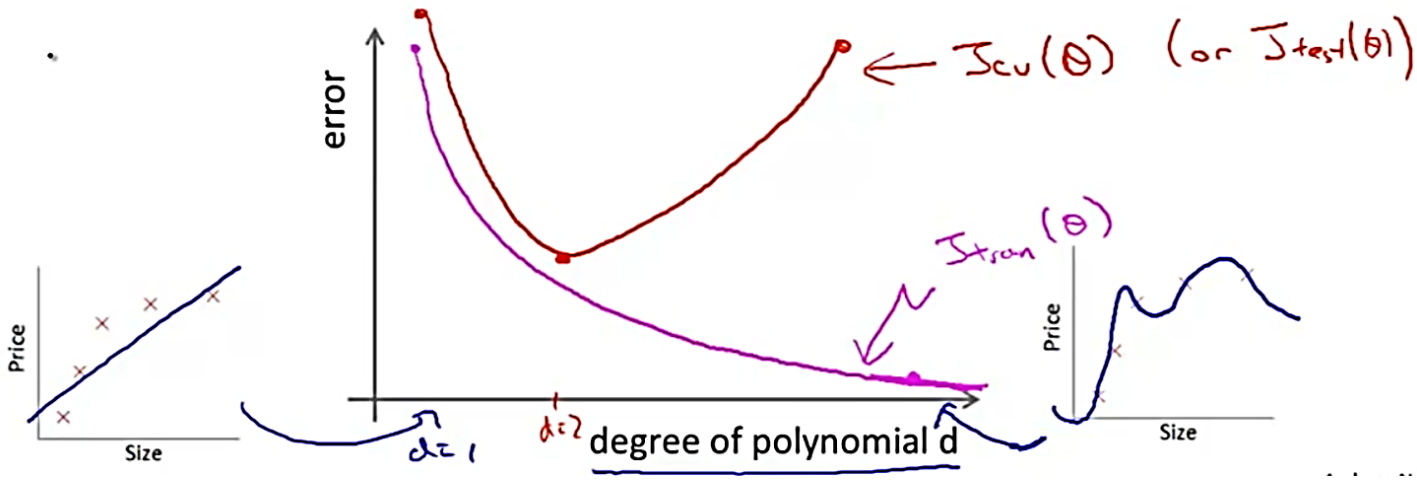

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance (overfit)

$d=4$

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 $J_{test}(\theta)$)




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.

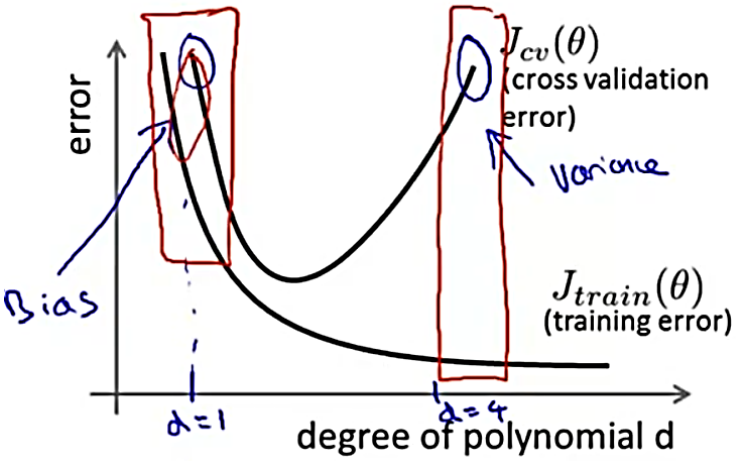
---

## DETERMINING WHETHER ITS BIAS OR VARIANCE:

**Diagnosing:** means determining/localizing the problem

### 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):
→ $J_{train}(\theta)$ will be high
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):
→ $J_{train}(\theta)$ will be low
$J_{cv}(\theta) >> J_{train}(\theta)$

**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 $J_{train}(\Theta)$.

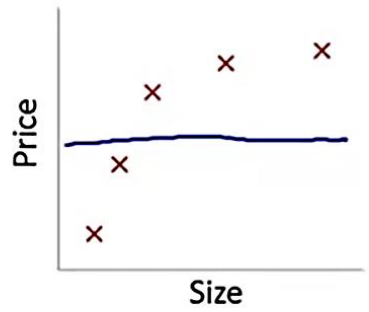
## Bias and Variance vs Regularization:

### Linear regression with regularization

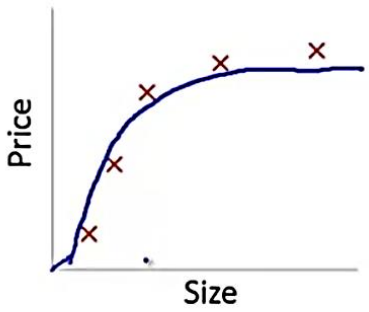
Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

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

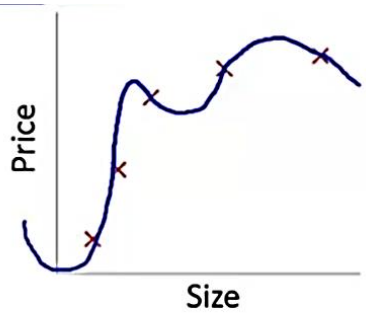
As λ approaches 0, hypothesis tends to overfit




Large $\lambda$ ←
→ High bias (underfit)
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \ldots$
$h_\theta(x) \approx \theta_0$




Intermediate $\lambda$ ←
"Just right"




→ Small $\lambda$
High variance (overfit)
→ $\lambda = 0$

And

---

## Choosing the regularization parameter λ:

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

⋮

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

↑ 10.24

Pick the one which gives lowest $J_{cv}$ (say 5th one)

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

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 $\lambda$s and for each $\lambda$ go through all the models to learn some $\Theta$.

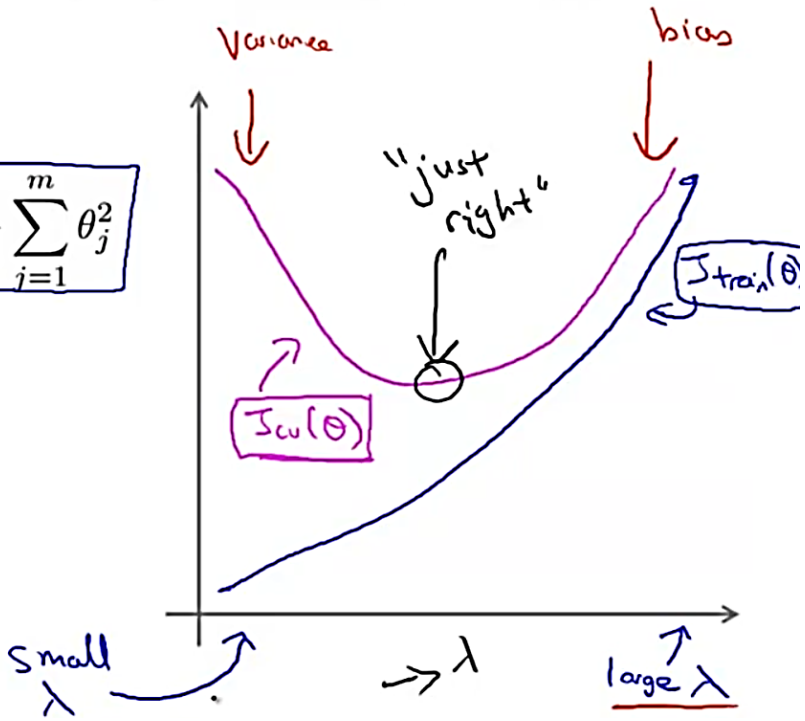
4. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ **without** regularization or λ = 0.
5. Select the best combo that produces the lowest error on the cross validation set.
6. Using the best combo Θ and λ, apply it on $J_{test}(\Theta)$ to see if it has a good generalization of the problem.

## Bias/variance as a function of the regularization parameter λ

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

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

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



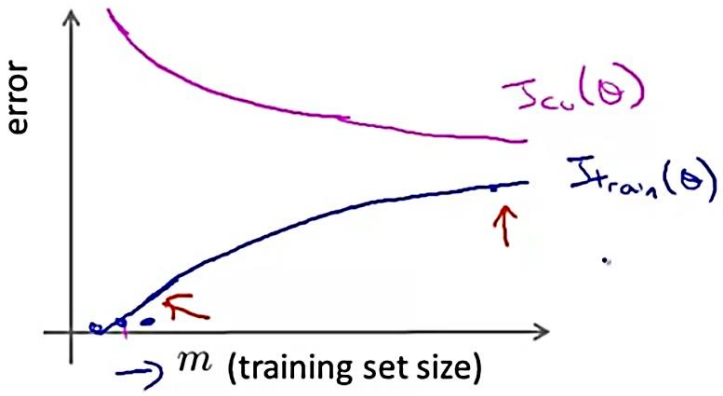

---

## BIAS/VARIANECE vs m (NO OF TRAINING EXAMPLES):

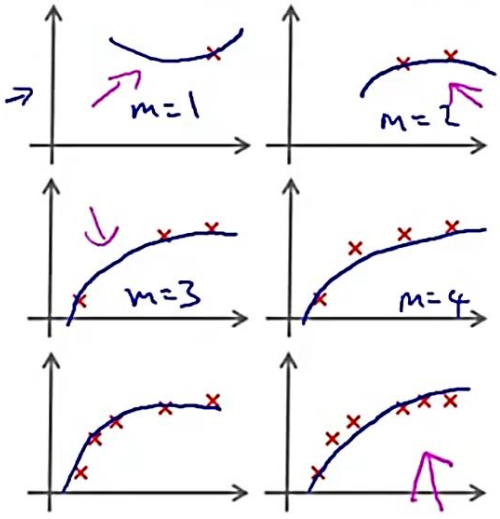
### Learning curves

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

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

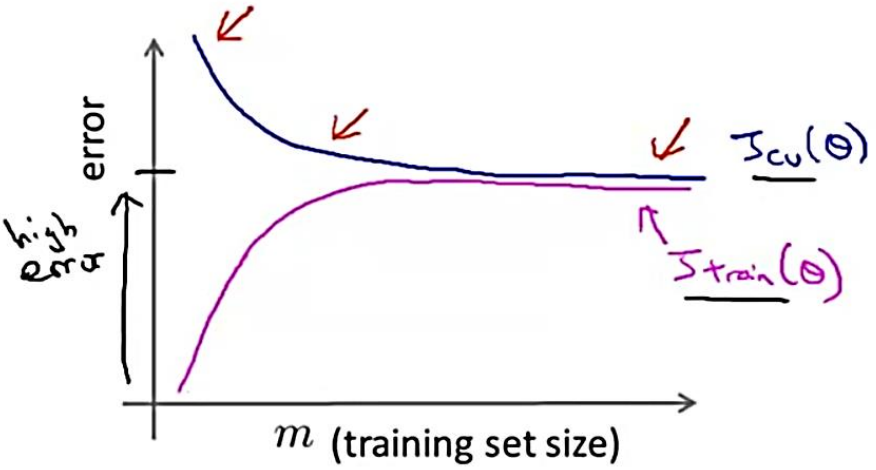


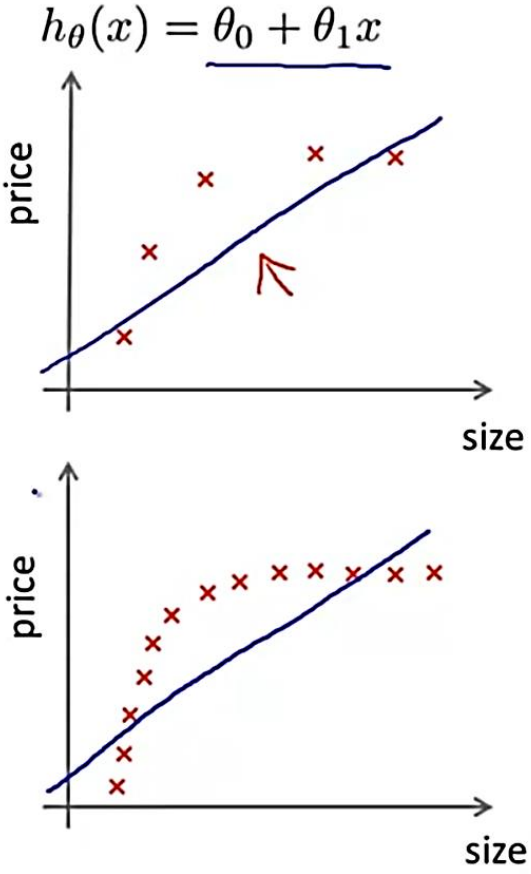

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



$J_{cv}$ tend to decrease as → more the no of training examples, better is the generalization to new data

**High bias**

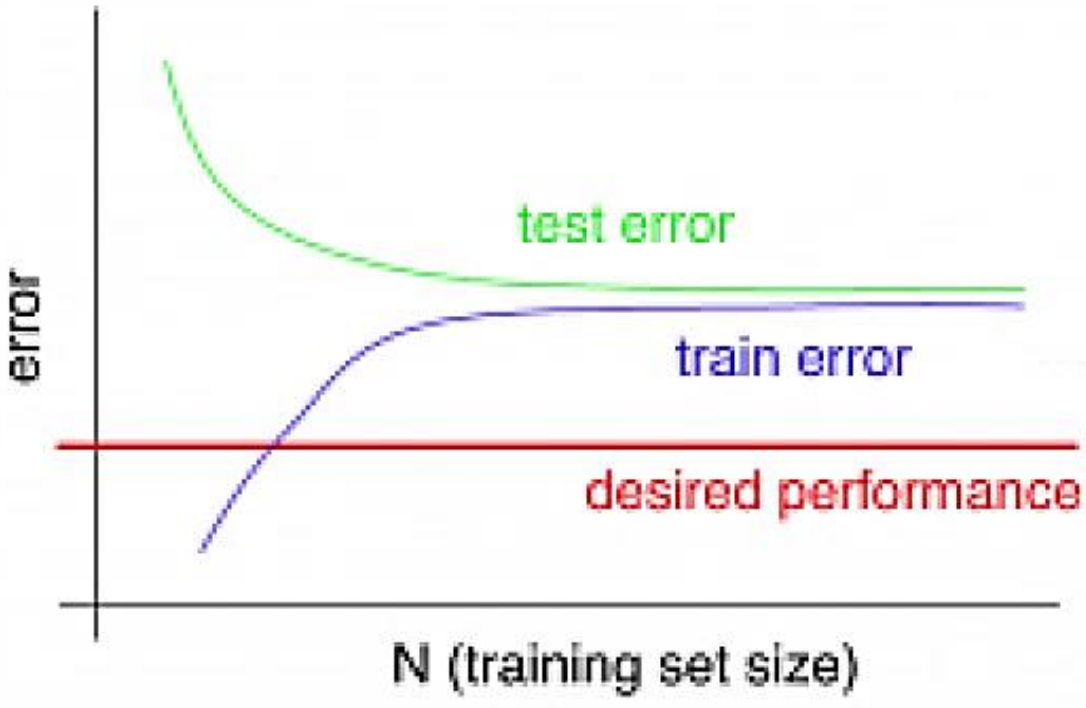



If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

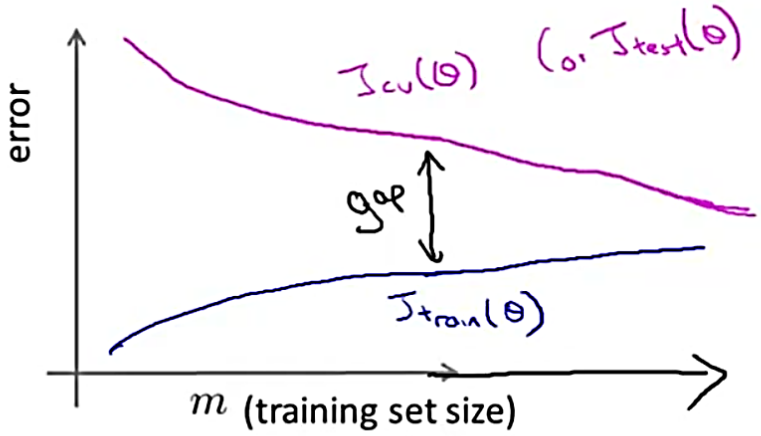
For high bias, as the training examples increases → $J_{cv}$ and $J_{train}$ come close to each other and both have a pretty high value

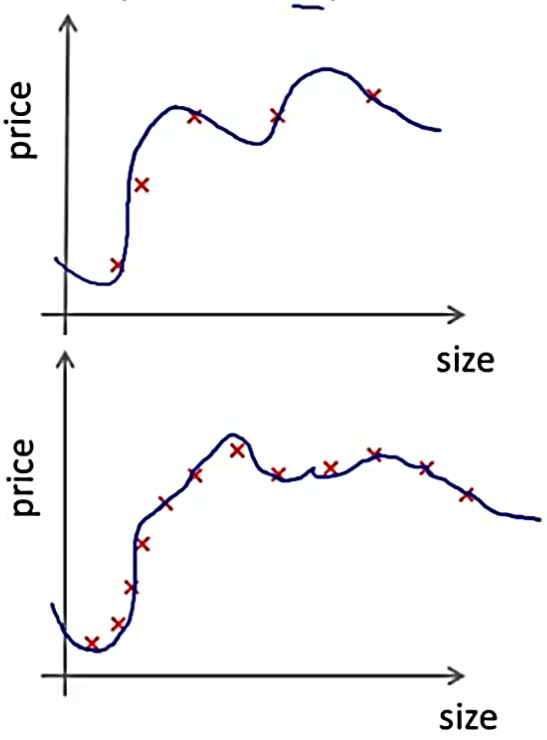
Typical learning curve for high bias (at fixed model complexity):



## High variance

$h_\theta(x) = \theta_0 + \theta_1 x$ (and small $\lambda$)



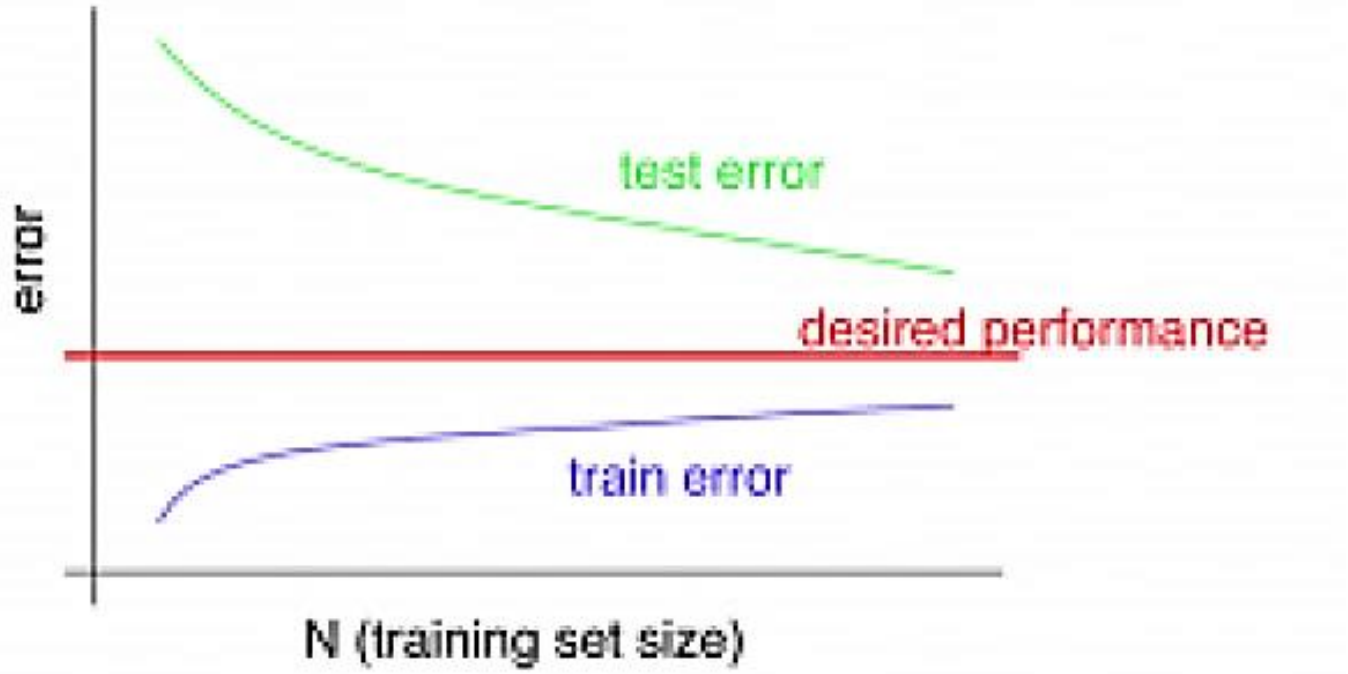




If a learning algorithm is suffering from high variance, getting more training data is likely to help.

The $J_{train}$ remains small as the data is perfectly fitted to trained model(overfitted). But, the $J_{cv}$ is quite high due to lack of generalization to new data

For high variance, the gap is quite large, but as we further increase the no of training examples, it is actually helpful.

Typical learning curve for high variance(at fixed model complexity):



# Debugging a learning algo:

**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
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc})$ → fixes high bias.
- Try decreasing $\lambda$ → fixes high bias
- Try increasing $\lambda$ → fixes high variance

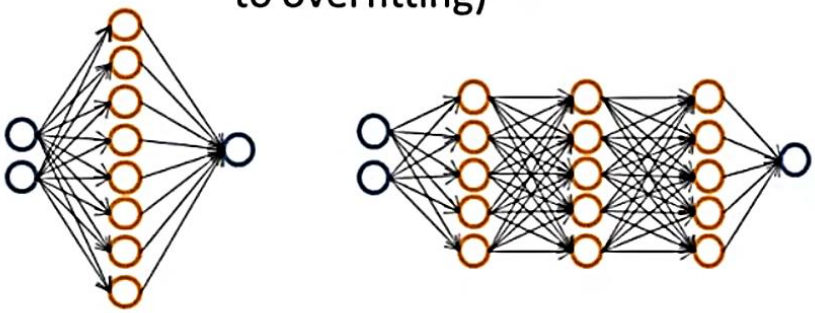
---

## 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 ($\lambda$) to address overfitting.

We can choose any type of NN, but a good choice is to chose a large NN with many hidden units or with many hidden layers, and use regularization in it to prevent overfitting.

**One good approach is to:**

- Try various types of NN
- Select the one with lowest $J_{CV}$

**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.

➔ **Why is the recommended approach to perform error analysis using the cross-validation data used to compute $J_{CV}(\theta)$ rather than the test data used to compute $J_{test}(\theta)$?**

If we develop new features by examining the test set, then we may end up choosing features that work well specifically for the test set, so Jtest(θ) is no longer a good estimate of how well we generalize to new examples.