Practices when Applying Machine Learning

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Introduction

Machine learning is one of areas where practical rules of thumb tend to emerge before the theoretical principles that might lead to those rules of thumb.

In this lecture, we cover some of the most important theoretical principles that underly or seek to explain our practical heuristics:

- The bias-variance trade-off
- How are training / cross-validation / test sets related?
- When we use use a high-variance model that is prone to overfitting, how can we control it?

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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?

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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
- Try getting additional features

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, etc.)$

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, etc.)$
- Try decreasing λ

Debugging a learning algorithm

Suppose you have implemented regularized linear regression to predict housing prices as follows:

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

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
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, etc.)$
- Try decreasing λ
- Try increasing λ **!**

Machine Learning Diagnostic

Diagnostic: A test that you can run to gain insight what is / is NOT 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.

Question

Which of the following statements about diagnostics are true?
Circle all that apply.

(i) It's hard to tell what will work to improve a learning algorithm, so the best approach is to go with gut feeling and just see what works.

(ii) Diagnostics can give guidance as to what might be more fruitful things to try to improve a learning algorithm.

(iii) Diagnostics can be time-consuming to implement and try, but they can still be a very good use of your time.

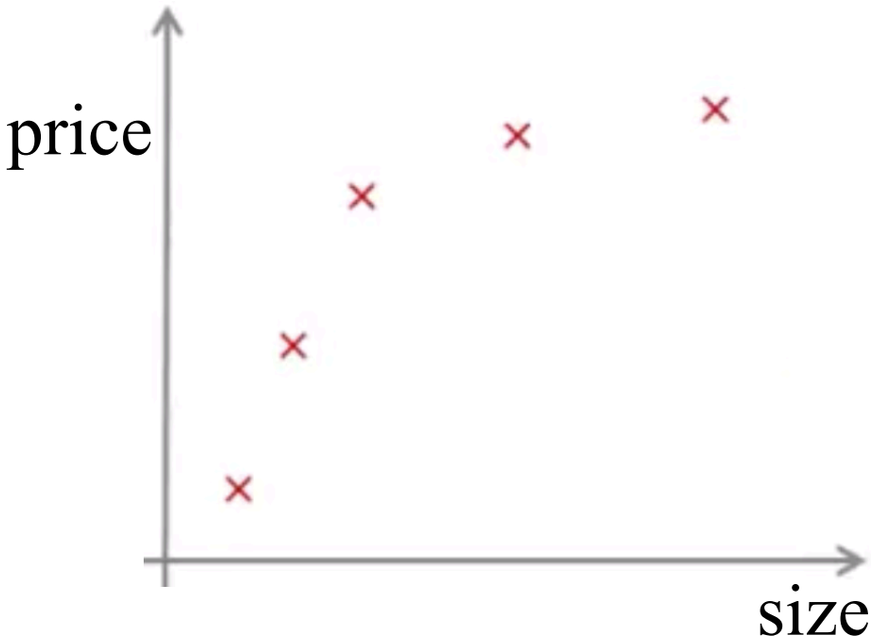
(iv) A diagnostic can sometimes rule out certain courses of action (changes to your learning algorithm) as being unlikely to improve its performance significantly.

Evaluating a Hypothesis

Motivation

Evaluating your hypothesis:

When we fit the parameters of our learning algorithm, we think about choosing the parameters to minimize the training error.

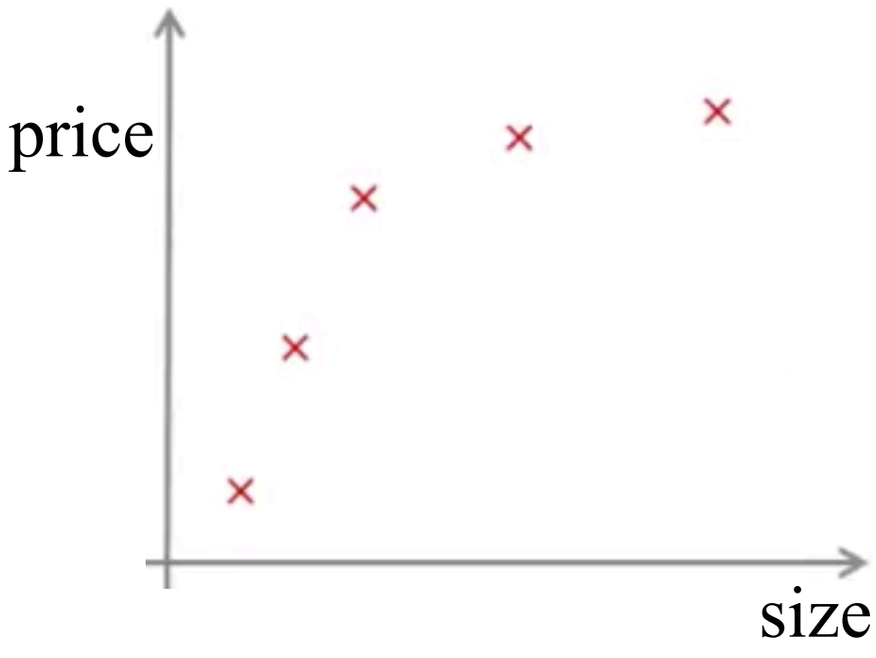


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Motivation

Evaluating your hypothesis:

When we fit the parameters of our learning algorithm, we think about choosing the parameters to minimize the training error.



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

Hence, fails to generalize to new examples not in the training set

Usually, there are a lot of features *e.g.*

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 = kitchen size
⋮

Evaluating Hypothesis

Our Dataset

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

Evaluating Hypothesis

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

30% Testing set $\longrightarrow$ $(x^{(1)}_{\text{test}}, y^{(1)}_{\text{test}})$, $(x^{(2)}_{\text{test}}, y^{(2)}_{\text{test}})$, $\vdots$, $(x^{(m)}_{\text{test}}, y^{(m)}_{\text{test}})$

Question

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

(i) The training error $J(\theta)$ to be **low** and the test error $J_{\textbf{test}}(\theta)$ to be **high**

(ii) The training error $J(\theta)$ to be **low** and the test error $J_{\textbf{test}}(\theta)$ to be **high**

(iii) The training error $J(\theta)$ to be **high** and the test error $J_{\textbf{test}}(\theta)$ to be **low**

(iv) The training error $J(\theta)$ to be **high** and the test error $J_{\textbf{test}}(\theta)$ to be **high**

Guidance for Linear Regression

Training / Testing procedure:

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

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

Guidance for Logistic Regression

Training / Testing procedure:

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

$$J_{\textbf{test}}(\theta) = -\frac{1}{m_{\textbf{test}}}\Sigma_{i=1}^{m\textbf{test}} y_{\textbf{test}}^{(i)} \log h_\theta\left(x_{\textbf{test}}^{(i)}\right) + \left(1 - y_{\textbf{test}}^{(i)}\right)\log h_\theta\left(x_{\textbf{test}}^{(i)}\right)$$

Guidance for Logistic Regression

Training / Testing procedure:

- Learn parameter θ from training data or the 70% portion (minimizing training error $J(\theta)$)
- Compute test set error
- Or, apply the 0/1 misclassification error technique:

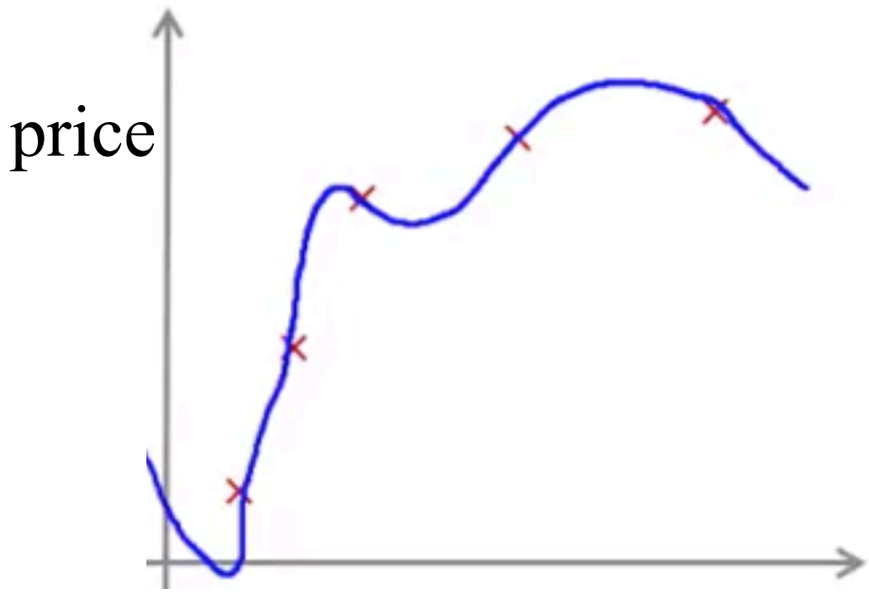
$$\textbf{err}(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5, y = 0 \\ & \text{or } h_\theta(x) < 0.5, y = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Test error} = \frac{1}{m_{\textbf{test}}} \sum_{i=1}^{m_{\textbf{test}}} \textbf{err}\left(h_\theta(x^{(i)}_{\textbf{test}}), y^{(i)}_{\textbf{test}}\right)$$

Model Selection and Training / Validation / Test Sets

Model Selection (Intuition)

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 Problem

Suppose you want to choose what degree polynomial to fit to data.

1. $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \Theta^{(1)} \longrightarrow J_{\textbf{test}}(\Theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{\textbf{test}}(\Theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{\textbf{test}}(\Theta^{(3)})$

$\vdots$

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

Model Selection Problem

Suppose you want to choose what degree polynomial to fit to data.

1. $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \Theta^{(1)} \longrightarrow J_{\textbf{test}}(\Theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{\textbf{test}}(\Theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{\textbf{test}}(\Theta^{(3)})$

$\vdots$

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

Suppose we choose $\theta_0 + \ldots + \theta_5 x^5$

This may seem reasonable; but,
it is likely to be an optimistic estimate of generalization error
i.e. the chosen degree of polynomial is fit to the test set.

Model Selection Problem

Our Dataset

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

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

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

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

Train / Validation / Test Error

Training error:

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

Cross Validation error:

$$J_{\textbf{cv}}(\theta) = \frac{1}{2m_{\textbf{cv}}}\Sigma_{i=1}^{m\textbf{cv}}\left(h_\theta(x^{(i)}_{\textbf{cv}}) - y^{(i)}_{\textbf{cv}}\right)^2$$

Test error:

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

Model Selection Problem

Suppose you want to choose what degree polynomial to fit to data.

1. $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \min_\theta J(\theta) \longrightarrow \Theta^{(1)} \longrightarrow J_{\mathbf{CV}}(\Theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{\mathbf{CV}}(\Theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{\mathbf{CV}}(\Theta^{(3)})$

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

10. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_{10} x^{10} \longrightarrow \Theta^{(10)} \longrightarrow J_{\mathbf{CV}}(\Theta^{(10)})$

Model Selection Problem

Suppose you want to choose what degree polynomial to fit to data.

$$
\begin{array}{lllll}
1. & h_\theta(x) = \theta_0 + \theta_1 x & \longrightarrow \min\limits_{\theta} J(\theta) \longrightarrow & \Theta^{(1)} & \longrightarrow J_{\textbf{cv}}(\Theta^{(1)}) \\
2. & h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 & \longrightarrow & \Theta^{(2)} & \longrightarrow J_{\textbf{cv}}(\Theta^{(2)}) \\
3. & h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 & \longrightarrow & \Theta^{(3)} & \longrightarrow J_{\textbf{cv}}(\Theta^{(3)}) \\
 & \vdots & & & \vdots \\
10. & h_\theta(x) = \theta_0 + \theta_1 x + \theta_{10} x^{10} & \longrightarrow & \Theta^{(10)} & \longrightarrow J_{\textbf{cv}}(\Theta^{(10)})
\end{array}
$$

Suppose that we select the 4th order degree polynomial *i.e.* $J_{\textbf{cv}}(\Theta^{(4)})$

Then, estimate generalization error for test set *i.e.* $J_{\textbf{test}}(\theta^{(4)})$

Question

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

(i) the selected degree of polynomial has been fit to the cross validation set.

(ii) the selected degree of polynomial has been fit to the test set.

(iii) the cross validation set is usually smaller than the test set.

(iv) the cross validation set is usually larger than the test set.

k-fold Cross Validation

When data is scarce, we'd prefer to select the model that is best on all the data, not just 20% or 30% of the data !

Then, k-fold cross validation makes more sense:

1. Split $\mathcal{S}$ into k disjoint subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ (Typical values of k are 5 and 10)
2. For each model M_i
    1. For $j = 1, \ldots, k$
        1. Train M_i on $\mathcal{S}_{\textbf{train}} = \mathcal{S}_1, \ldots, \mathcal{S}_{j-1}, \mathcal{S}_{j+1}, \ldots, \mathcal{S}_k$
        2. Test the trained M_i on $\mathcal{S}_{\textbf{CV}} = \mathcal{S}_j$
3. Select the M_i with the lowest average error on $\mathcal{S}_{\textbf{CV}}$ over the k folds.
4. Retrain M_i on all of $\mathcal{S}$.

(This is sometimes called 'leave-one-out' CV)

k-fold Cross Validation

Be careful about how you split !

- Usually, the partitioning of examples into *k* folds is random uniform.
- However, sometimes training items are related to each other (*e.g.* multiple crops of the same object in the same image).
- Randomized partitioning will put some related examples in different folds, leading to overestimated performance.
- In this case, it's necessary to ensure that the related examples are in the same fold.

Feature Selection

A special case of model selection is 'feature selection'.

Suppose you have many features *e.g.* $n \gg m$

(This is possible in some domains such as biological sequence analysis, where we might want to find segments of DNA sequence in a particular genome that code for a particular biological function, or to a lesser extent in text classification)

So, we treat the problem as model selection *i.e.* among the 2^n possible subsets of features, find the subset that gives the best cross validation performance.

Trying all 2^n subsets is impossible unless n is very small. So, we need a more efficient approach.

Feature Selection

The forward search method of feature selection:

1. Initialize $\mathcal{F} := \emptyset$
2. For $j \in \{1,\ldots,n\}$
    1. For $i \in \{1,\ldots,n\} \backslash \mathcal{F}$
        1. Let $\mathcal{F}_i := \mathcal{F} \cup \{i\}$
        2. Train on $\mathcal{F}_i$ with cross validation and obtain the cross validation error
    2. Let $\mathcal{F} := \mathcal{F} \cup \{i\}$, where i is the feature that gave the lowest cross validation error.
3. Select the feature set that gave the lowest cross validation error over all tests.

Feature Selection

When we have a method that wraps our learning algorithm, supplying a different feature set on each iteration, this is called 'wrapper model selection'.

Another wrapper method is backward search, in which we start with $\mathcal{F} := \{1,2,\ldots,n\}$ and iteratively remove the least informative feature.

Wrapper methods work well but takes a lot of time. For example, $\mathcal{O}(kn^2)$ calls to the optimization algorithm for forward selection with k-fold cross validation.

Filter Feature Selection

Filter feature selection methods use a heuristic to decide which subsets of features to try.

Example: we may rank features according to some measure of relatedness to the desired output; then, incrementally add features in that order if they improve generalization.

One of the most common measure of relatedness for discrete features is called 'mutual information', denoted by $\mathbf{MI}(X_i, Y)$, between feature X_i and target Y :

$$\mathbf{MI}(X_i, Y) = \Sigma_{x_i \in X} \Sigma_{y_i \in Y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i) p(y)}$$

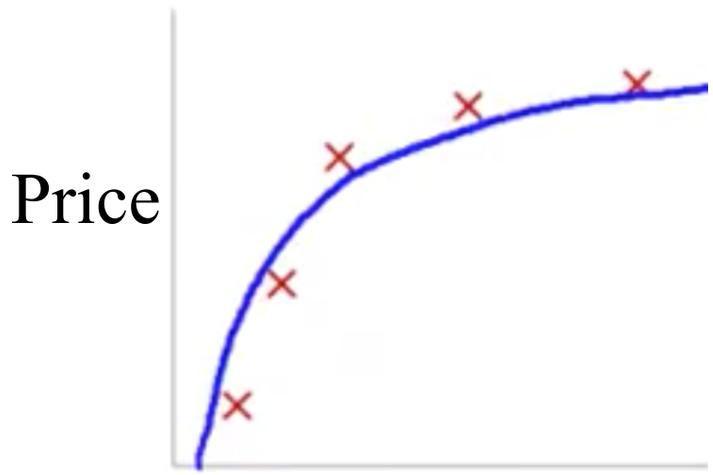
Diagnostic Bias *vs.* Variance

Bias / Variance (Recap)

$\theta_0 + \theta_1 x$

High bias (underfit)

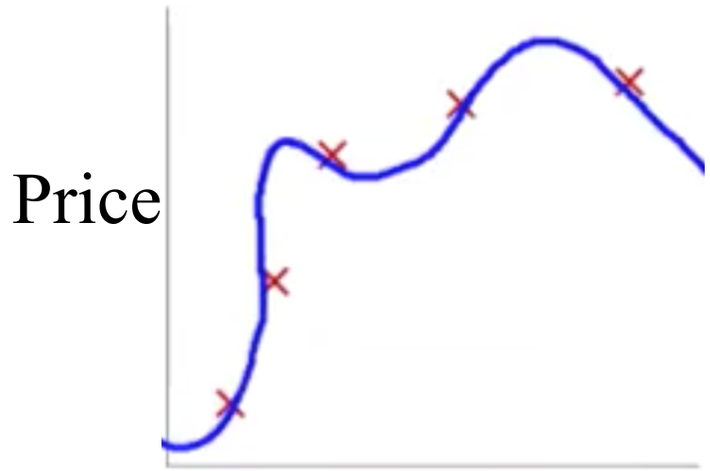
degree = 1



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

‘Just Right’

degree = 2



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

High variance (overfit)

degree = 4

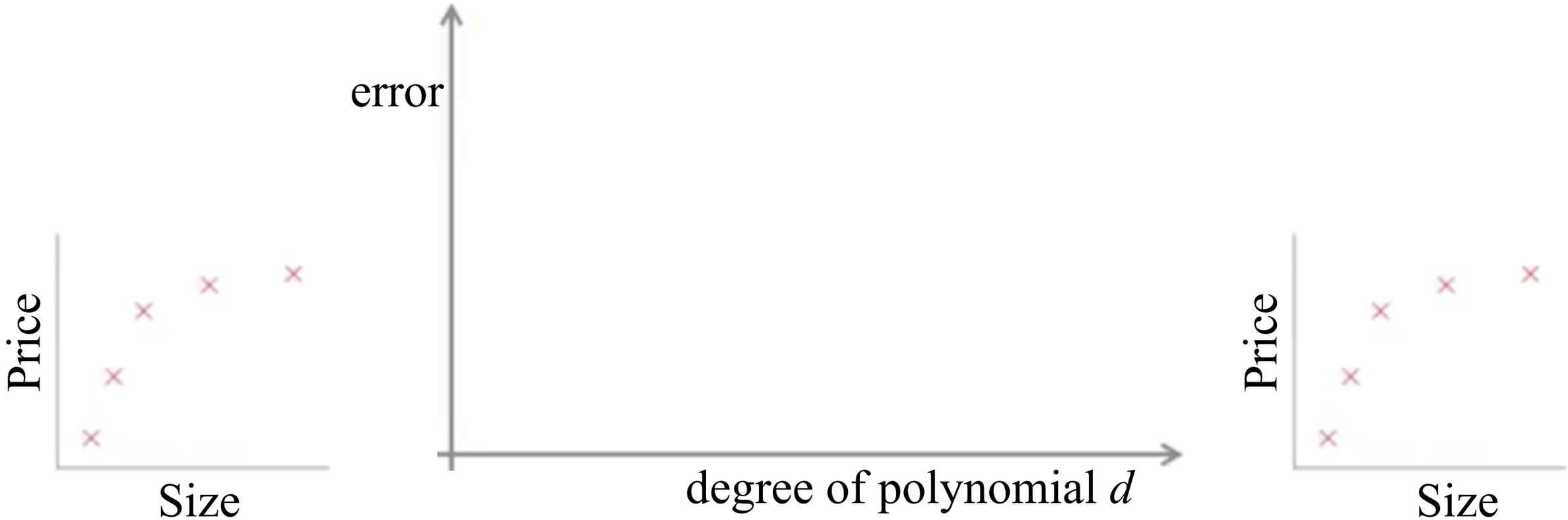
Bias / Variance

Training error:

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

Cross Validation error:

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



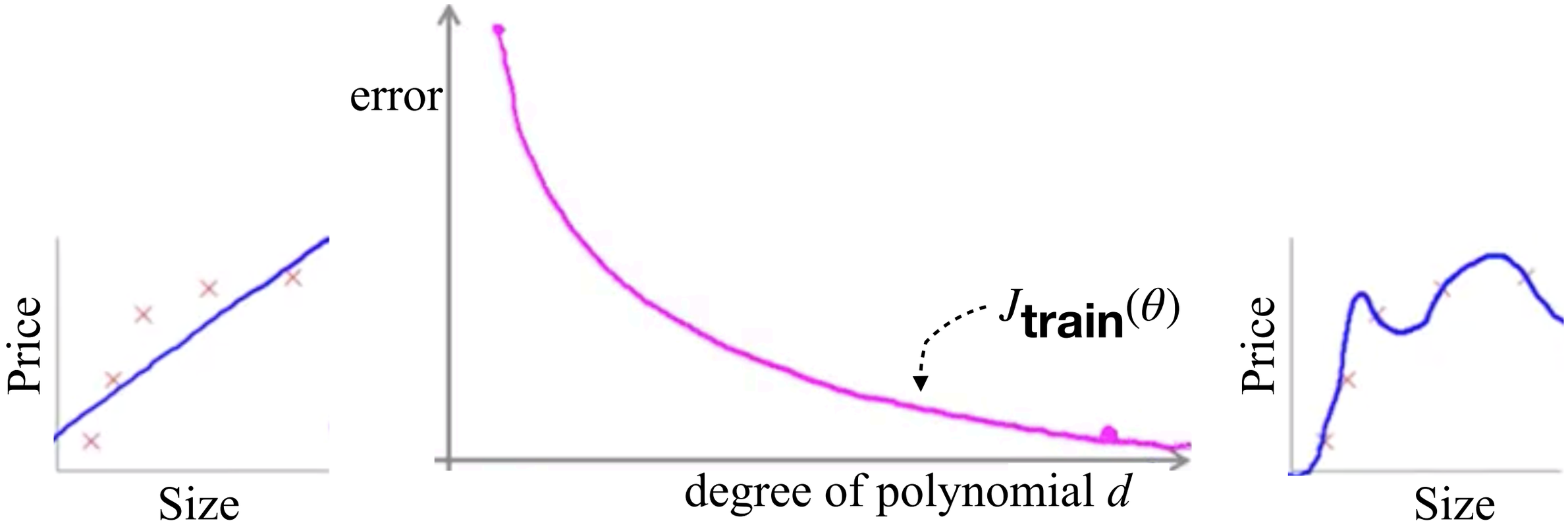
Bias / Variance

Training error:

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

Cross Validation error:

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



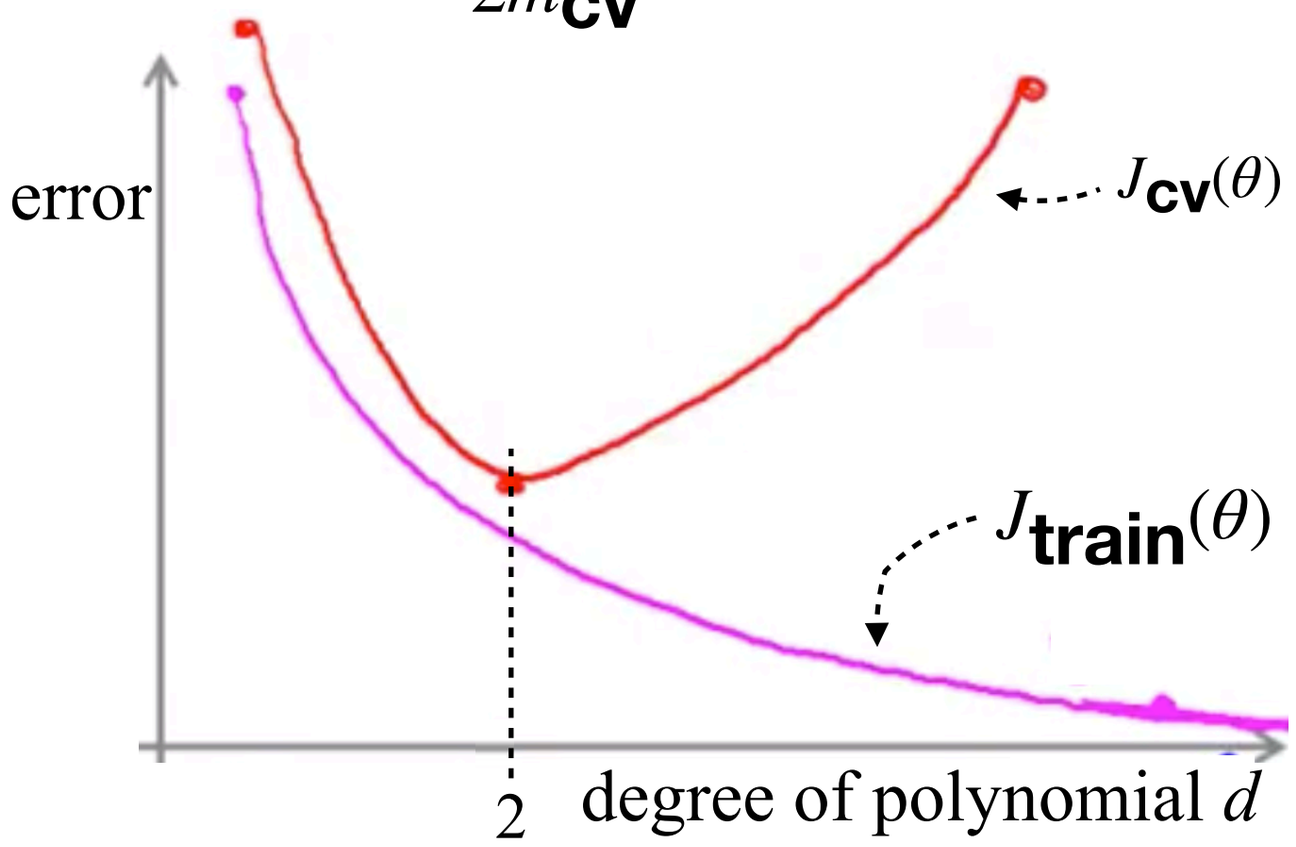
Bias / Variance

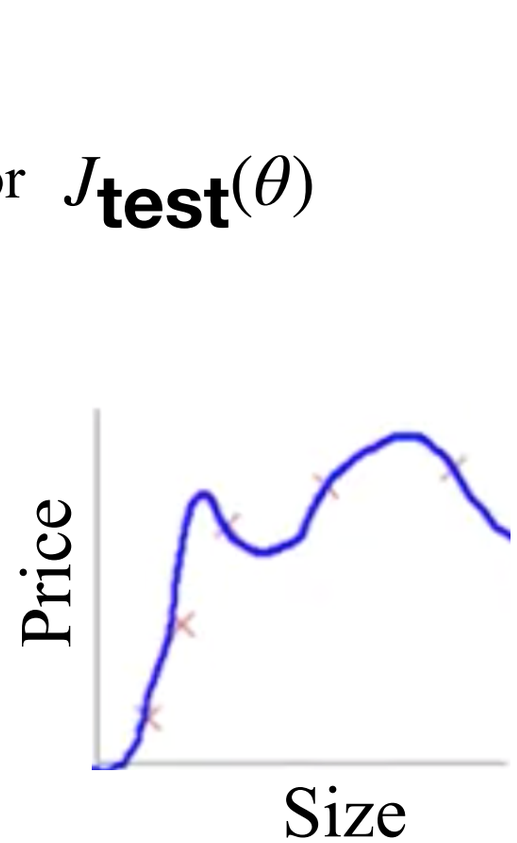
Training error:

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

Cross Validation error:

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

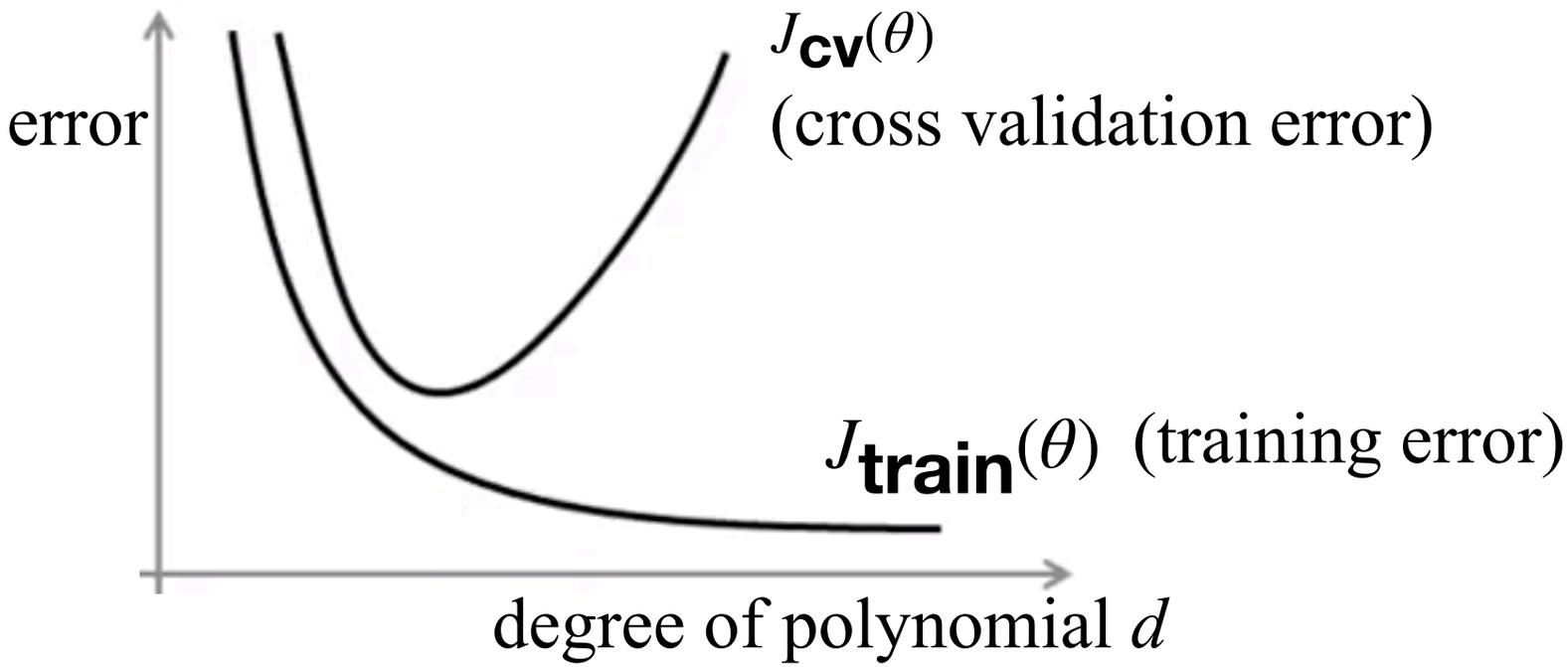




Diagnosing bias *vs.* variance

Suppose your learning algorithm is performing less well than you were hoping *i.e.* $J_{\mathbf{cv}}(\theta)$ or $J_{\mathbf{test}}(\theta)$ is high.

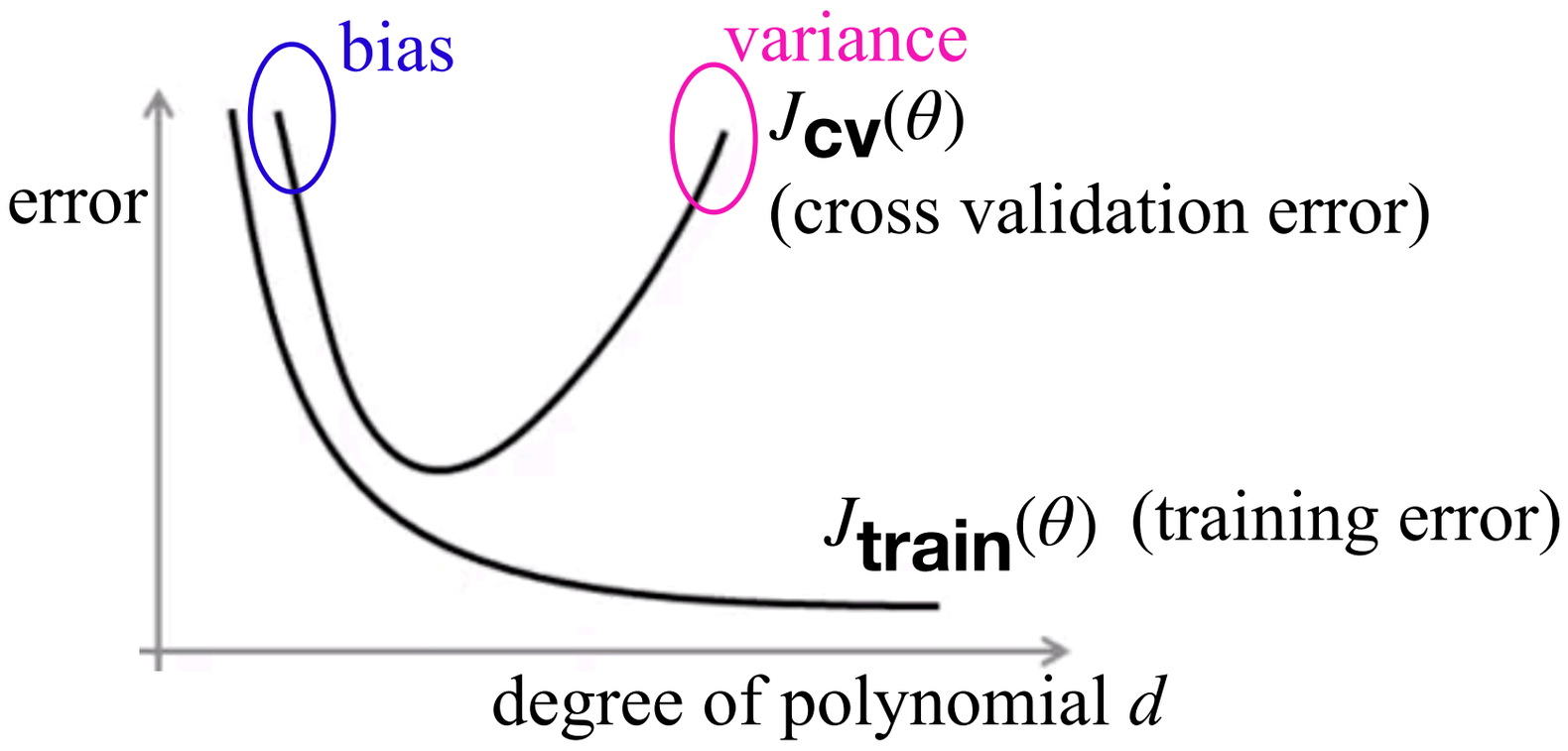
Is it a bias problem or a variance problem?



Diagnosing bias *vs.* variance

Suppose your learning algorithm is performing less well than you were hoping *i.e.* $J_{\mathbf{cv}}(\theta)$ or $J_{\mathbf{test}}(\theta)$ is high.

Is it a bias problem or a variance problem?



Bias (underfit)

- $J_{\mathbf{train}}(\theta)$ will be high
- $J_{\mathbf{cv}}(\theta) \approx J_{\mathbf{train}}(\theta)$

Variance (overfit)

- $J_{\mathbf{train}}(\theta)$ will be low
- $J_{\mathbf{cv}}(\theta) \gg J_{\mathbf{train}}(\theta)$

Question

Suppose you have a classification problem. The (misclassification) error is defined as $(1/m)\Sigma_1^m \mathbf{err}\left(h_\theta(x^{(i)}), y^{(i)}\right)$,

and the cross validation (misclassification) error is similarly defined, using the cross validation examples $(x^{(1)}_{\mathbf{cv}}, y^{(1)}_{\mathbf{cv}}), \ldots, (x^{(m_{\mathbf{cv}})}_{\mathbf{cv}}, y^{(m_{\mathbf{cv}})}_{\mathbf{cv}})$.

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?

(i) High bias (overfitting)
(ii) High bias (underfitting)
(iii) High variance (overfitting)
(iv) High variance (underfitting)

Regularization and Bias / Variance

Regularization (Recap)

Linear regression with regularization

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

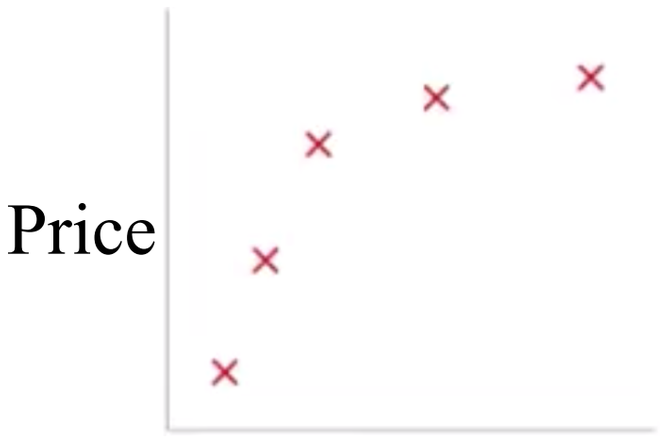
$$J(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\Sigma_{j=1}^{n}\theta_j^2$$

Regularization (Recap)

Linear regression with regularization

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

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



Large λ

High bias (underfit)

$\lambda = 10000. \theta_1 \approx 0, \theta_2 \approx 0, \ldots$

$h_\theta(x) \approx \theta_0$

Immediate λ

'Just Right'

Small λ

High variance (overfit)

$\lambda \approx 0$

Regularization (Recap)

Linear regression with regularization

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

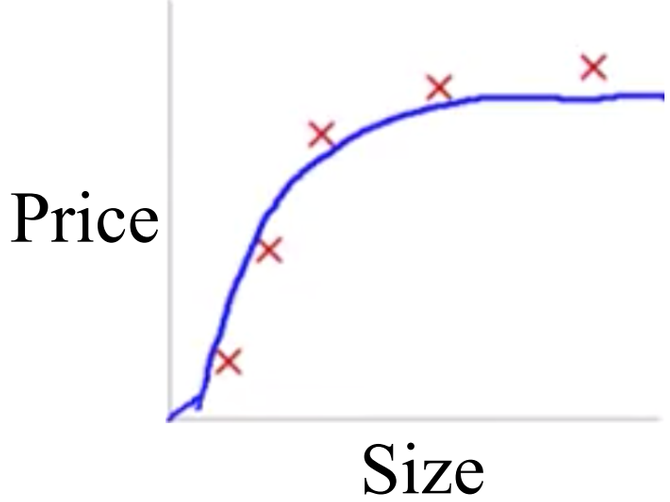
$$J(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\Sigma_{j=1}^{n}\theta_j^2$$

Large λ

High bias (underfit)

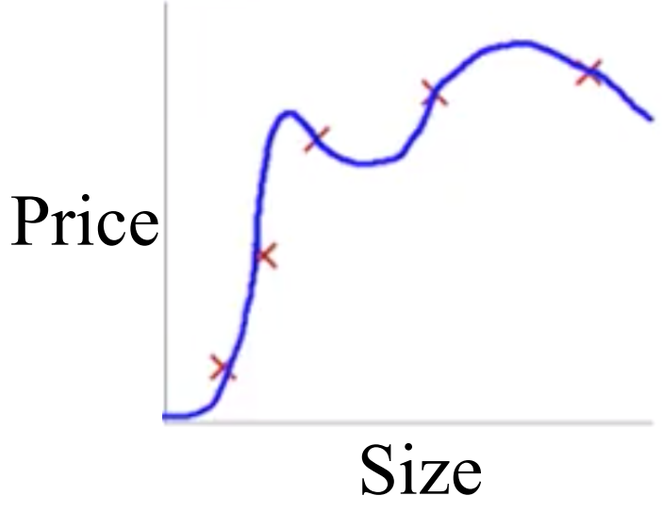
$\lambda = 10000.\theta_1 \approx 0, \theta_2 \approx 0, \ldots$

$h_\theta(x) \approx \theta_0$



Immediate λ

'Just Right'



Small λ

High variance (overfit)

$\lambda \approx 0$

Choosing Regularization Parameters

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

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

Choosing Regularization Parameters

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

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

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

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

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

Choosing Regularization Parameters

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

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

How to choose the regularization parameters λ automatically?

1. Try $\lambda = 0$
2. Try $\lambda = 0.01$
3. Try $\lambda = 0.02$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$

$\vdots$

12. Try $\lambda = 10$

Choosing Regularization Parameters

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

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

How to choose the regularization parameters λ automatically?

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

$\vdots$

12. Try $\lambda = 10 \longrightarrow \min_\theta J(\theta) \longrightarrow \theta^{(12)}$

Choosing Regularization Parameters

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

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

How to choose the regularization parameters λ automatically?

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

$\vdots$

12. Try $\lambda = 10$ $\longrightarrow$ $\min_\theta J(\theta)$ $\longrightarrow$ $\theta^{(12)}$ $\longrightarrow$ $J_{\mathbf{CV}}(\theta^{(12)})$

Choosing Regularization Parameters

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

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

How to choose the regularization parameters λ automatically?

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

$\vdots$

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

Pick up the lowest error !

e.g. $\theta^{(5)}$

Choosing Regularization Parameters

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

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

Next, compute test error *i.e.*

$J_{\textbf{test}}(\theta^{(5)})$

How to choose the regularization parameters λ automatically?

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

$\vdots$

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

Pick up the lowest error !

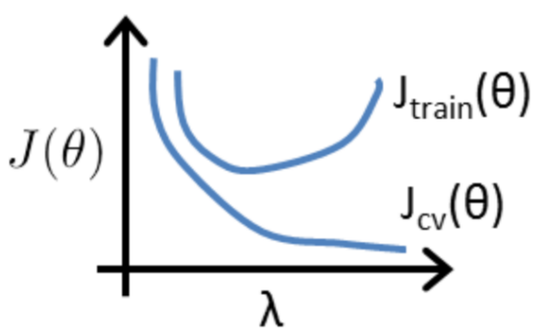
e.g. $\theta^{(5)}$

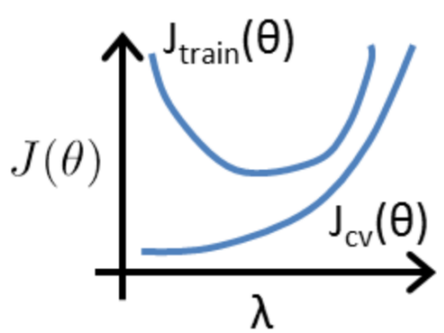
Question

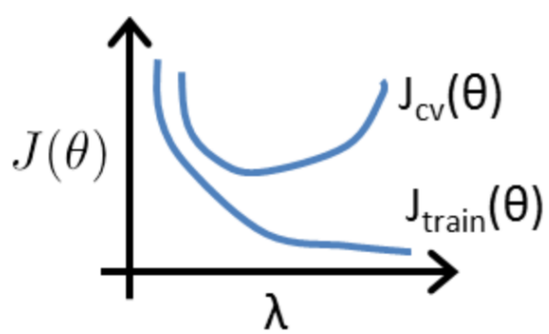
Consider regularized logistic regression. Let

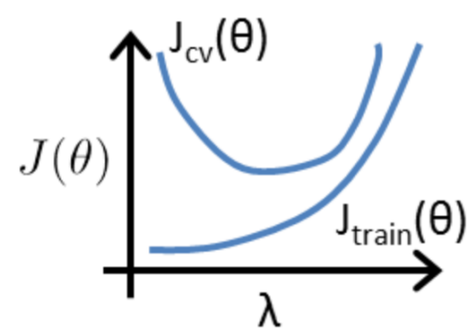
- $J(\theta) = \frac{1}{2m}\left[\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\Sigma_{j=2}^{n}\theta_j^2\right]$
- $J_{\mathbf{train}}(\theta) = \frac{1}{2m_{\mathbf{train}}}\left[\Sigma_{i=1}^{m\mathbf{train}}\left(h_\theta(x_{\mathbf{train}}^{(i)}) - y_{\mathbf{train}}^{(i)}\right)^2\right]$
- $J_{\mathbf{cv}}(\theta) = \frac{1}{2m_{\mathbf{cv}}}\left[\Sigma_{i=1}^{m\mathbf{cv}}\left(h_\theta(x_{\mathbf{cv}}^{(i)}) - y_{\mathbf{cv}}^{(i)}\right)^2\right]$

Suppose you plot $J_{\mathbf{train}}$ and $J_{\mathbf{cv}}$ as a function of the regularization parameter λ. Which of the following plots do you expect to get?









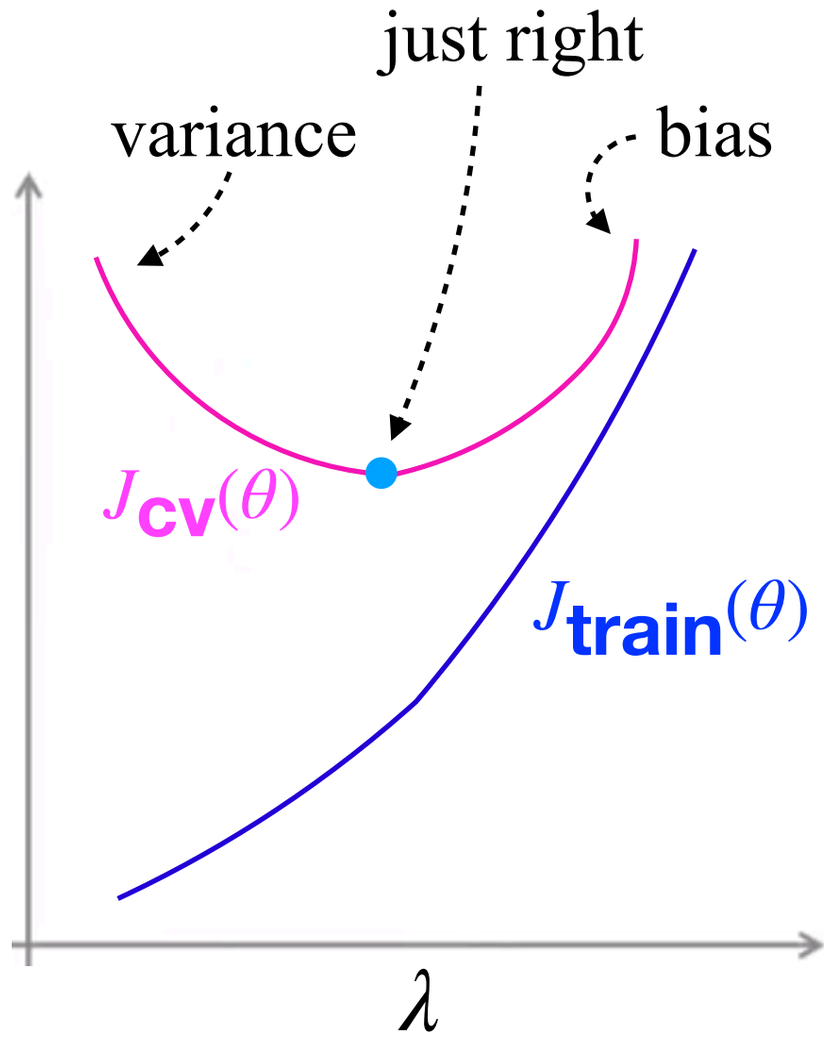
Choosing Regularization Parameters

Bias / variance as a function of the regularization parameter λ

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

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

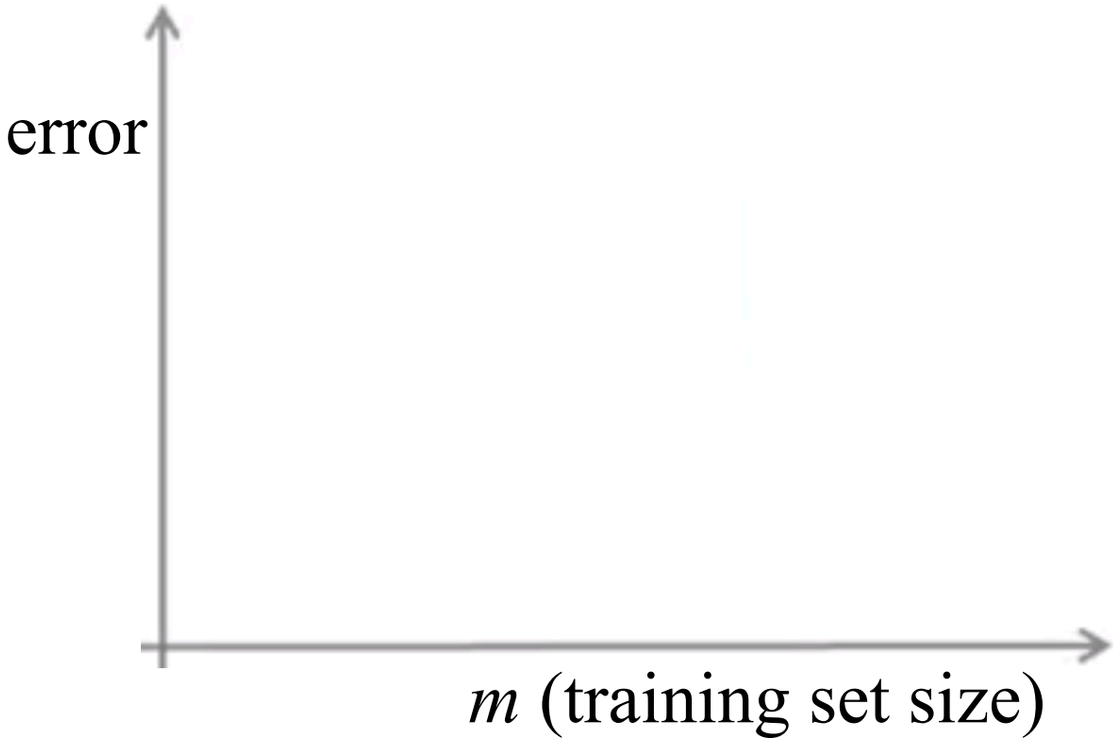
$$J_{\textbf{cv}}(\theta) = \frac{1}{2m_{\textbf{cv}}}\Sigma_{i=1}^{m_{\textbf{cv}}}\left(h_\theta(x_{\textbf{cv}}^{(i)}) - y_{\textbf{cv}}^{(i)}\right)^2$$



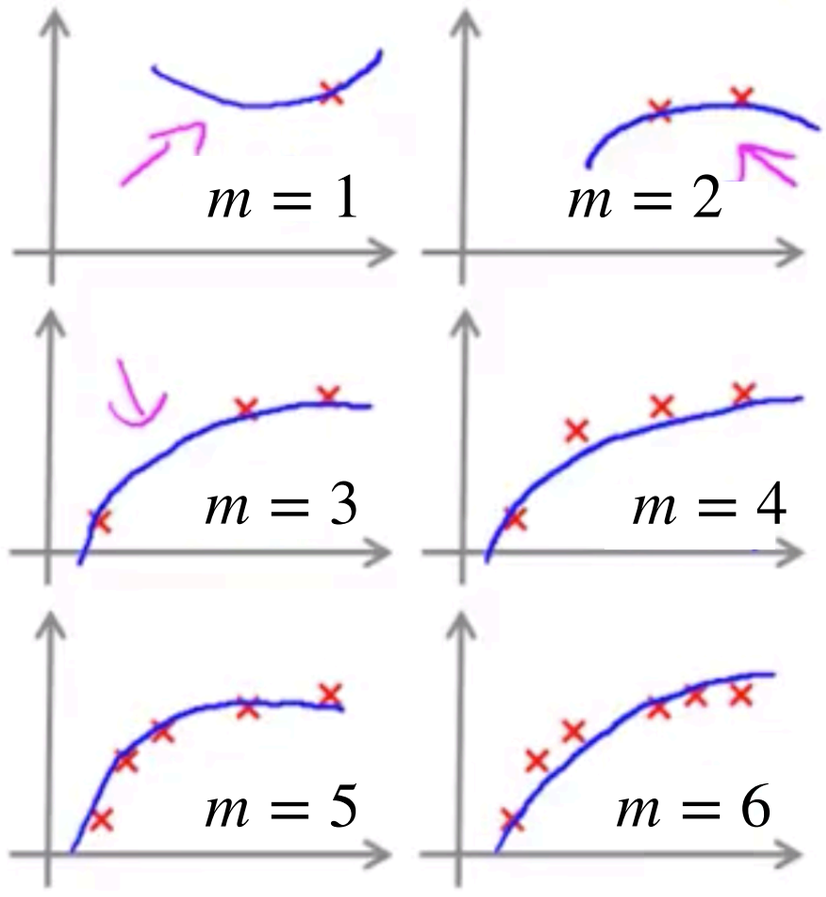
Learning Curves

Learning Curves (Idea)

- Plot $J_{\textbf{train}}(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- Plot $J_{\textbf{cv}}(\theta) = \frac{1}{2m_{\textbf{cv}}}\Sigma_{i=1}^{m_{\textbf{cv}}}\left(h_\theta(x_{\textbf{cv}}^{(i)}) - y_{\textbf{cv}}^{(i)}\right)^2$

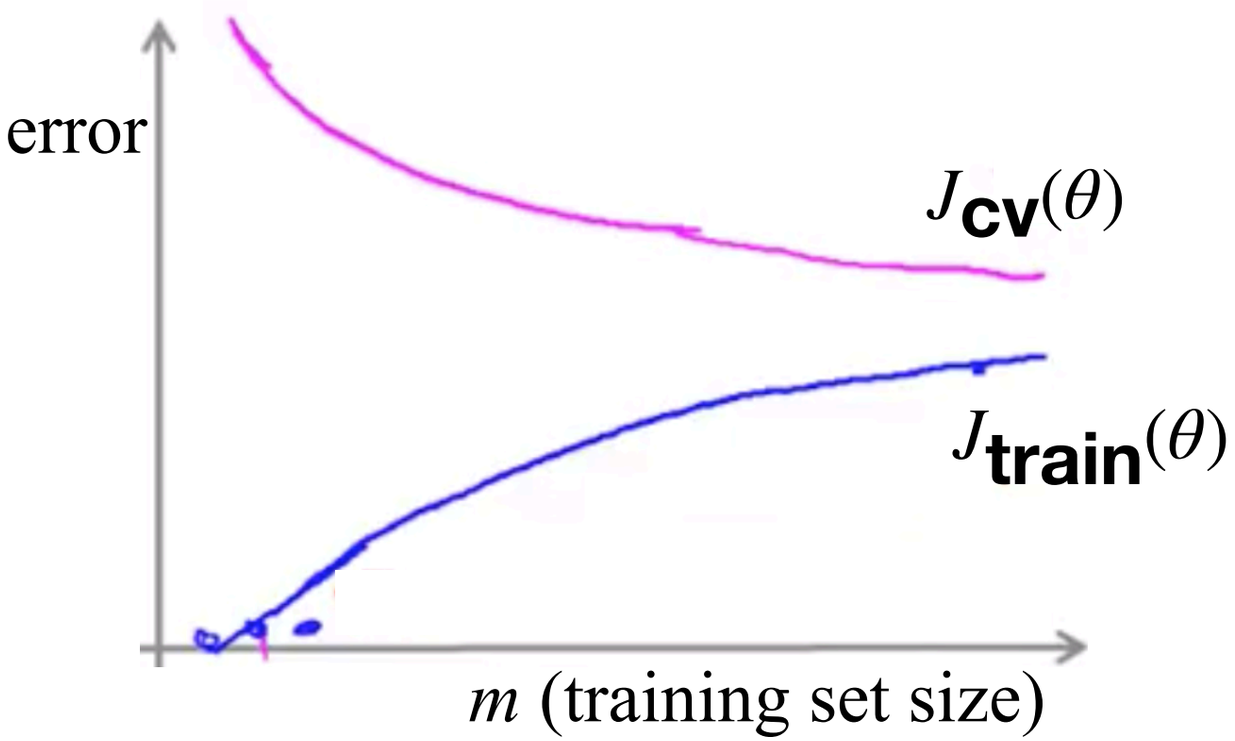


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

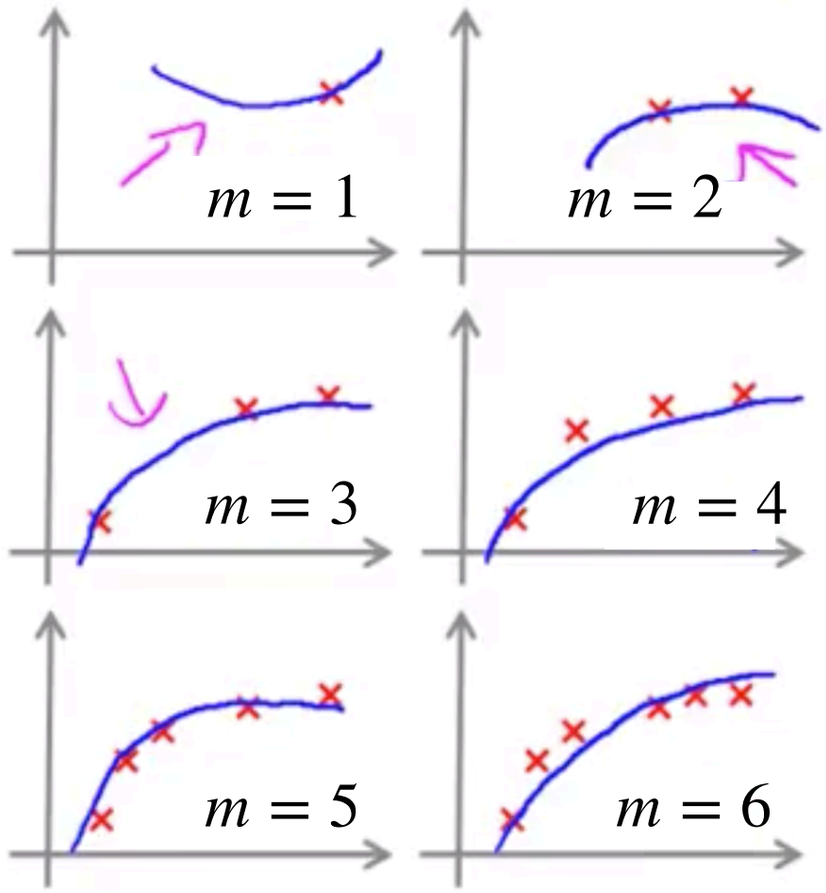


Learning Curves (Idea)

- Plot $J_{\textbf{train}}(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- Plot $J_{\textbf{cv}}(\theta) = \frac{1}{2m_{\textbf{cv}}}\Sigma_{i=1}^{m_{\textbf{cv}}}\left(h_\theta(x_{\textbf{cv}}^{(i)}) - y_{\textbf{cv}}^{(i)}\right)^2$



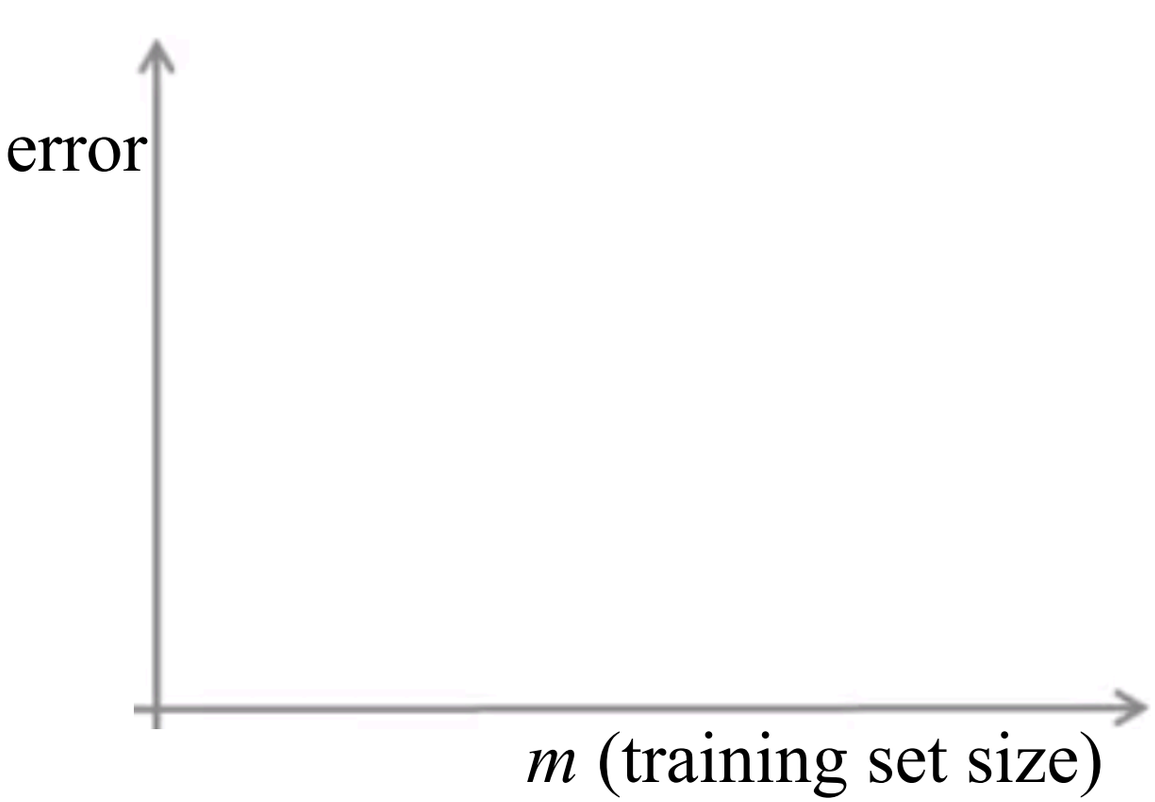
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

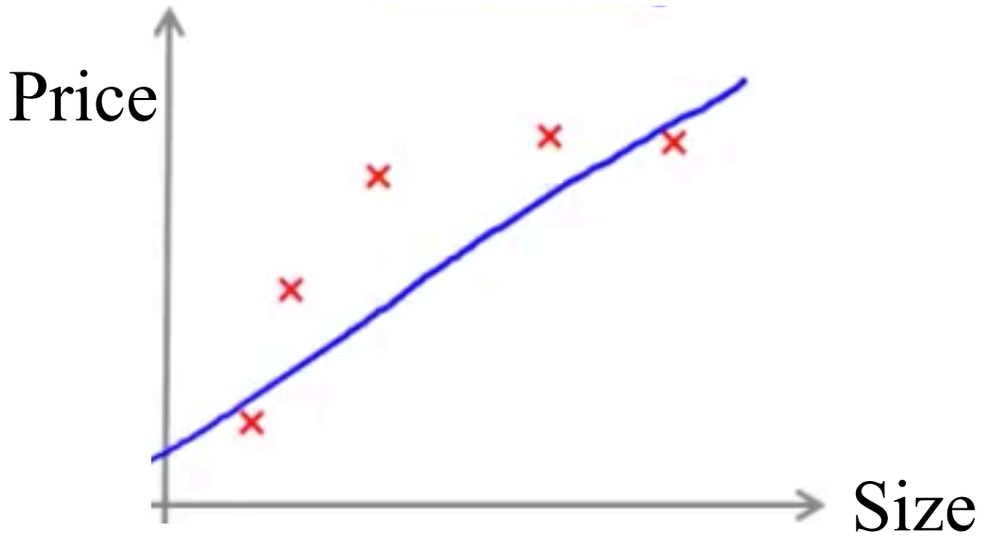


Learning Curves

High bias

$h_\theta(x) = \theta_0 + \theta_1 x$

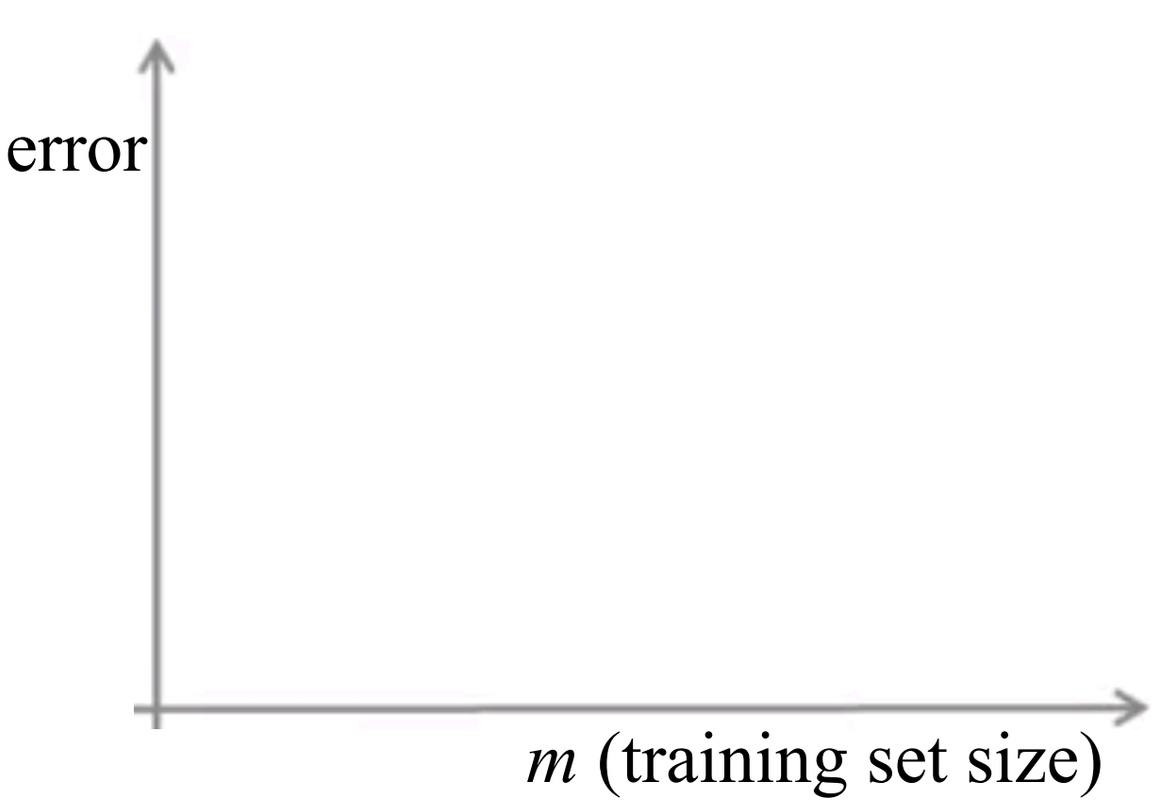


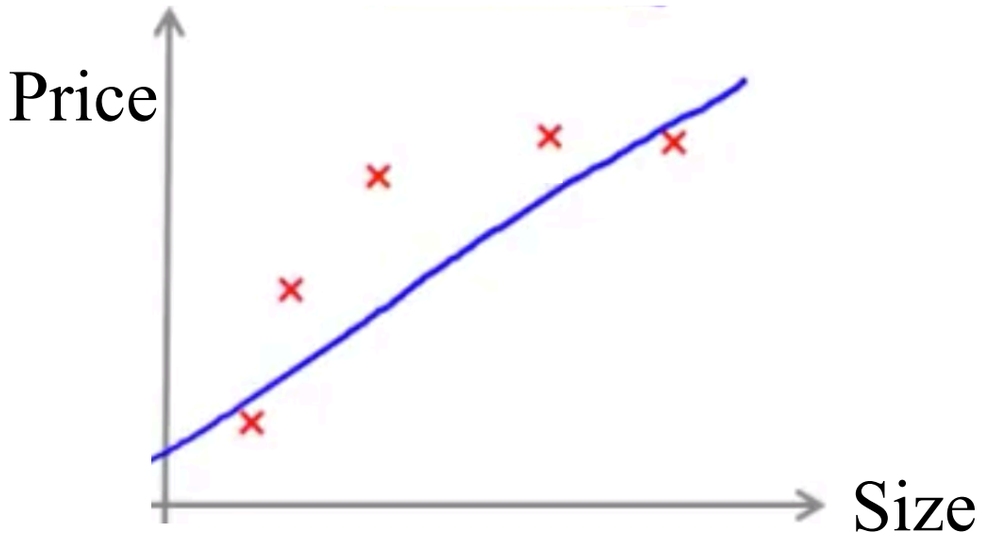


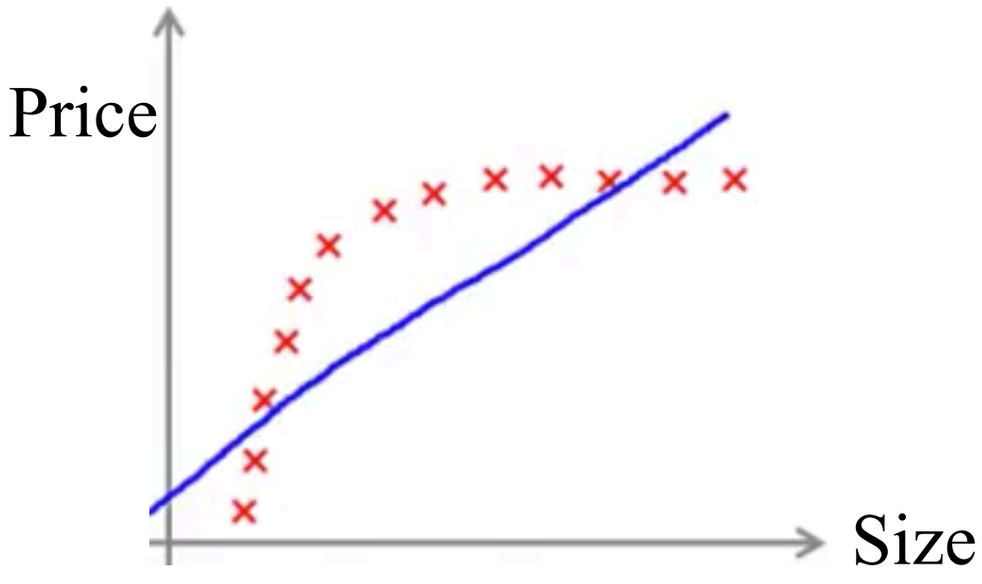
Learning Curves

High bias

$h_\theta(x) = \theta_0 + \theta_1 x$



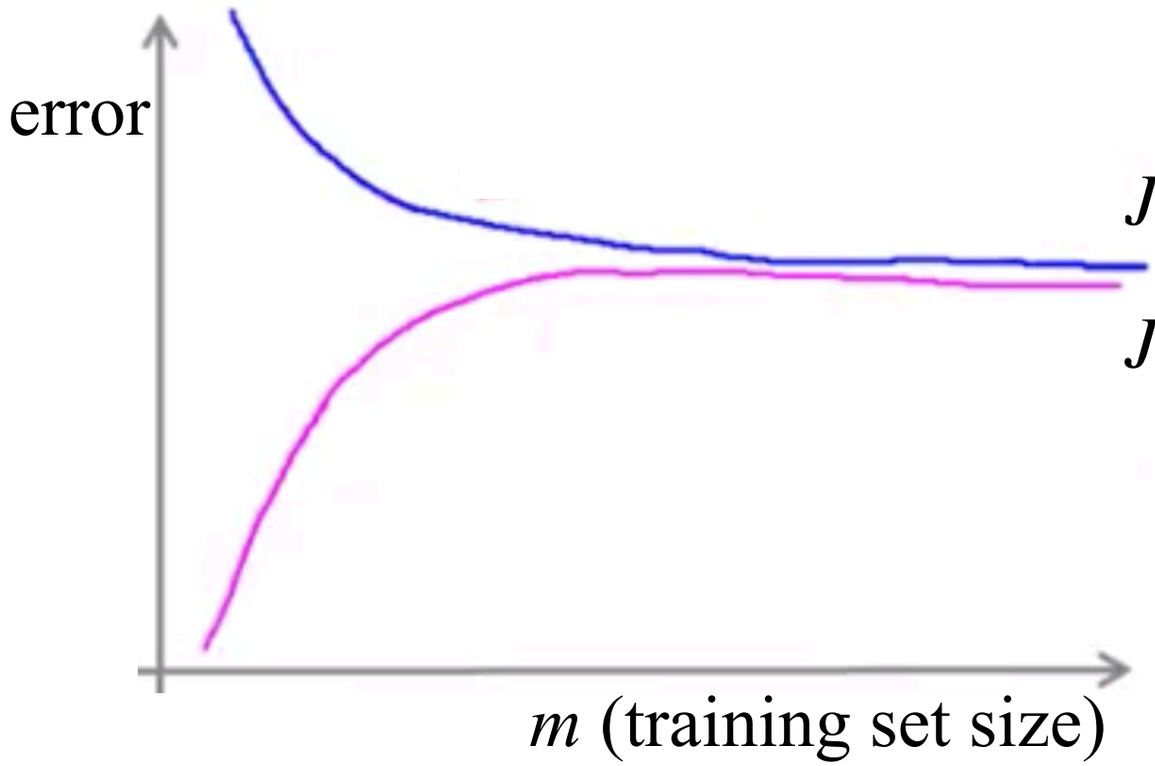


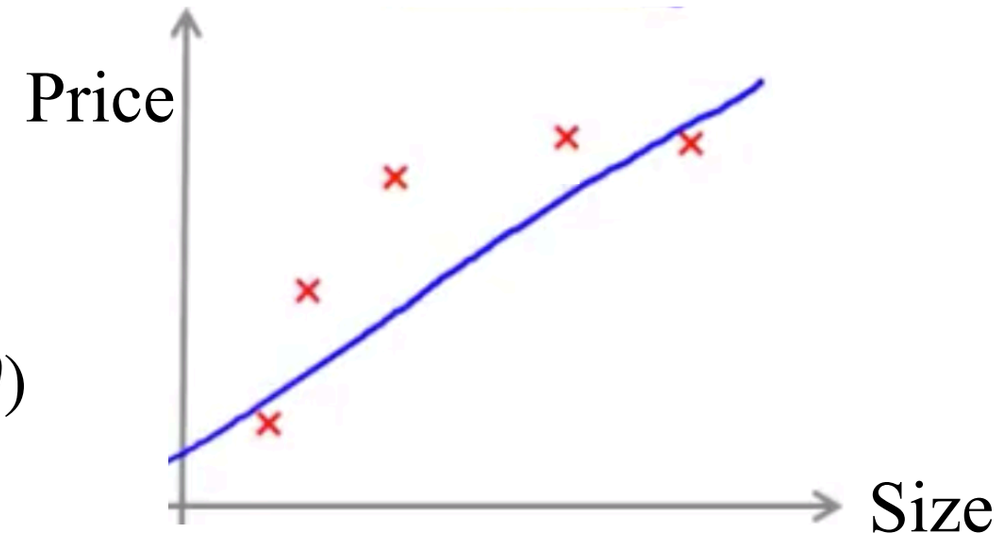


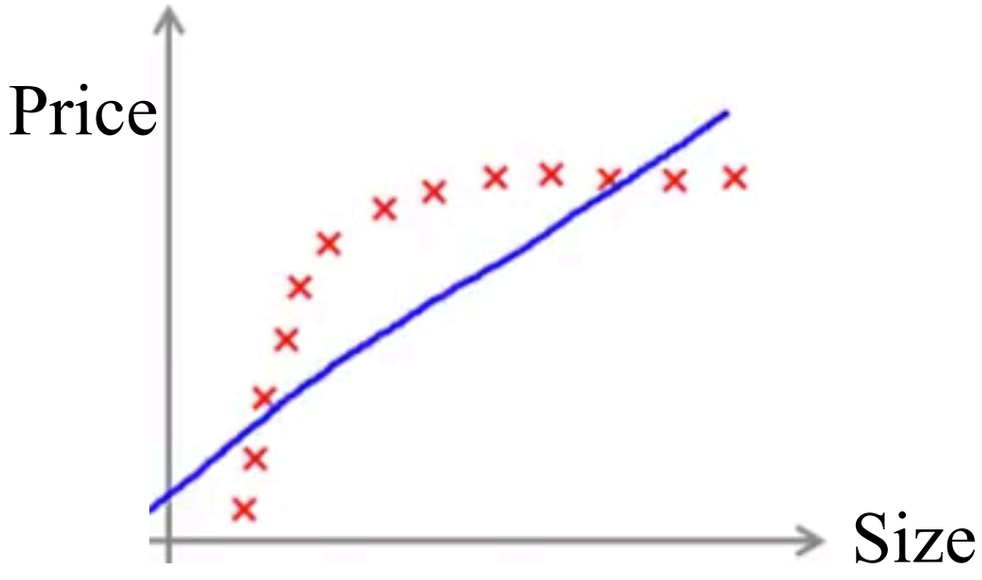
Learning Curves

High bias

$h_\theta(x) = \theta_0 + \theta_1 x$







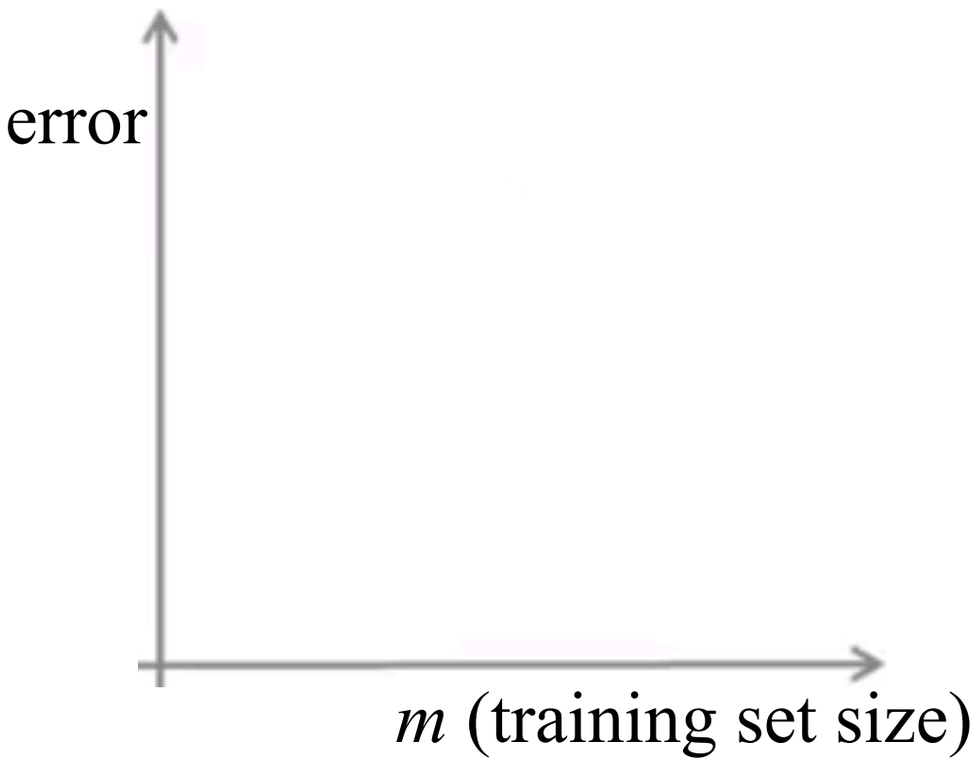
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much !

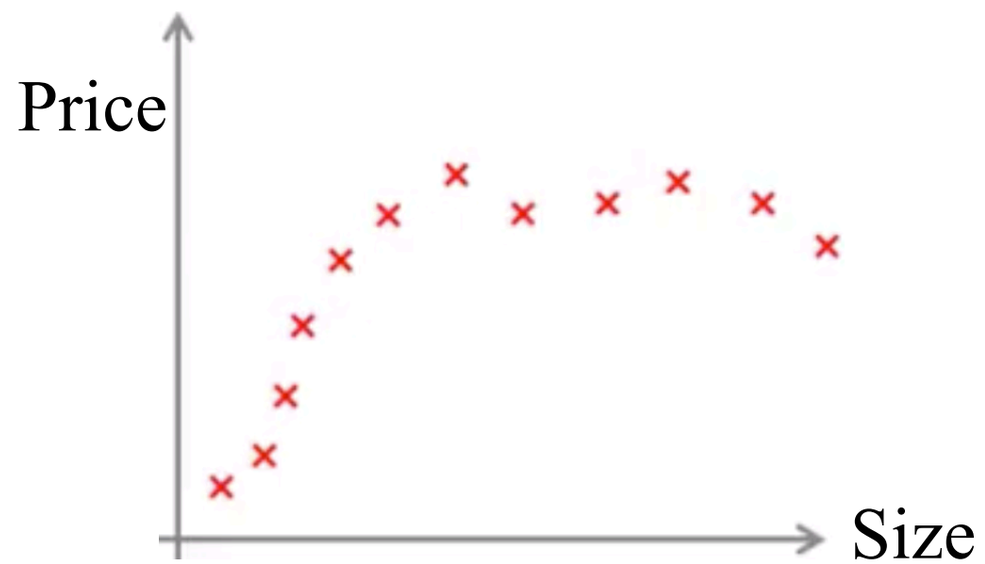
Learning Curves

High variance

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

(and small λ)



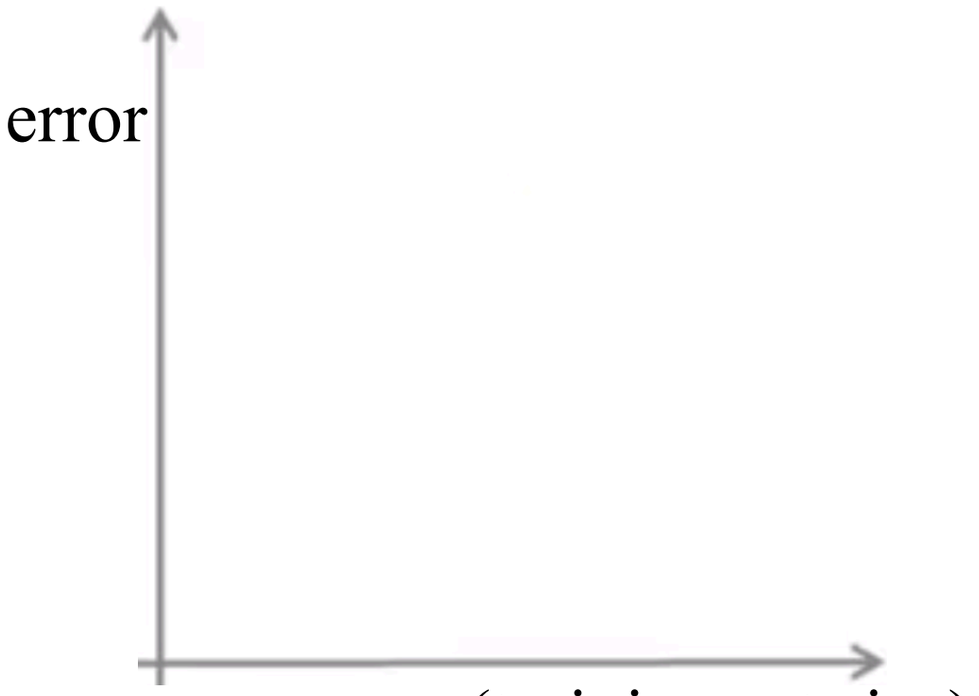


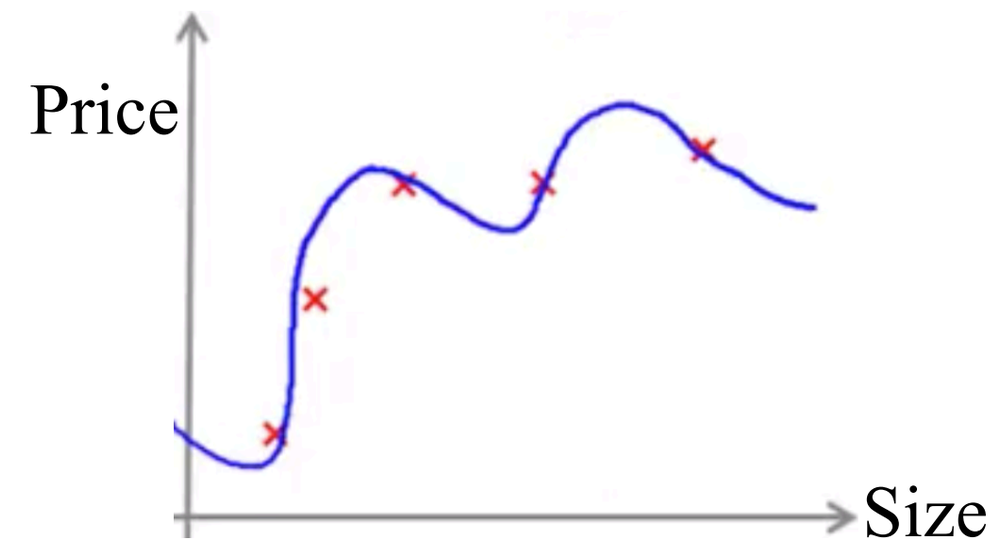
Learning Curves

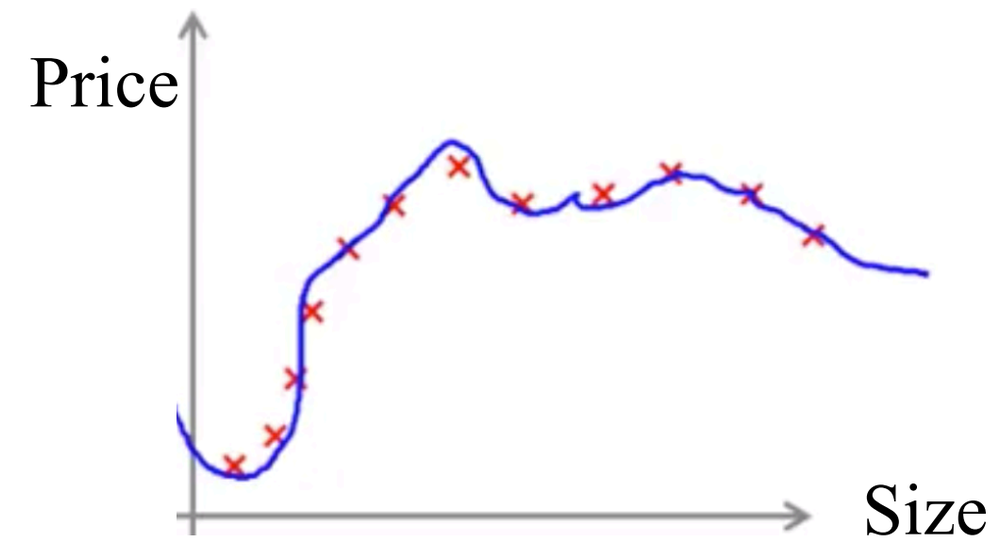
High variance

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

(and small λ)

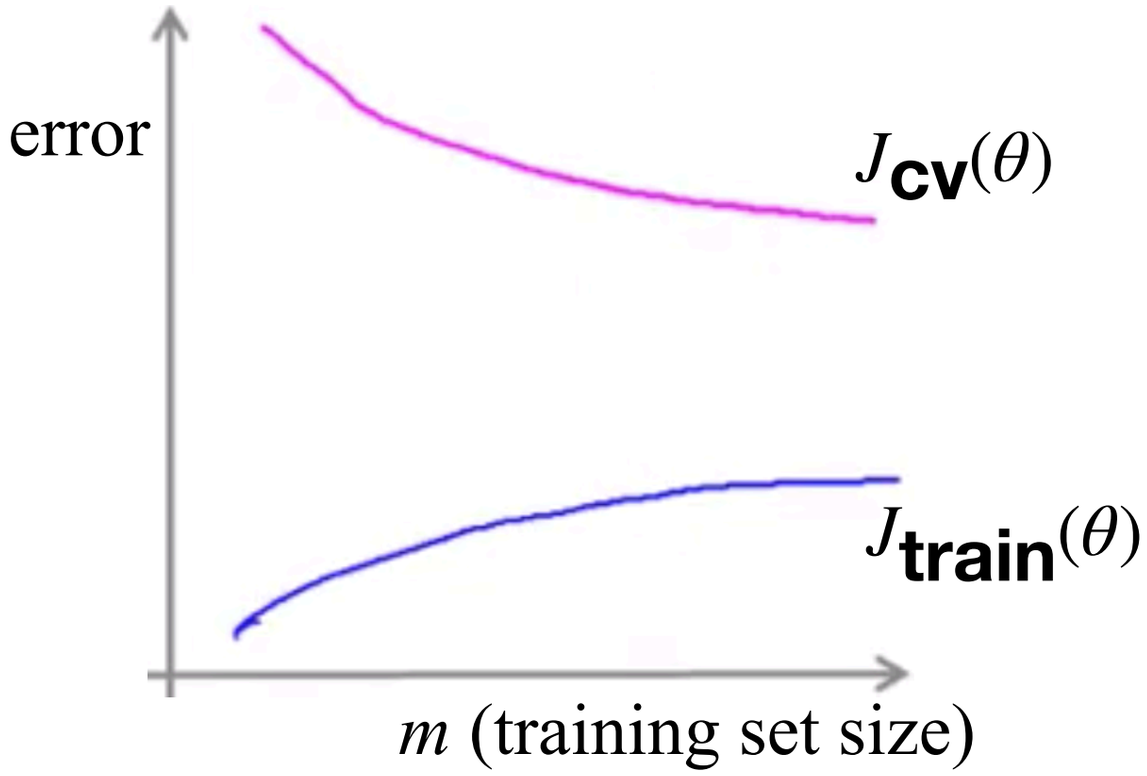






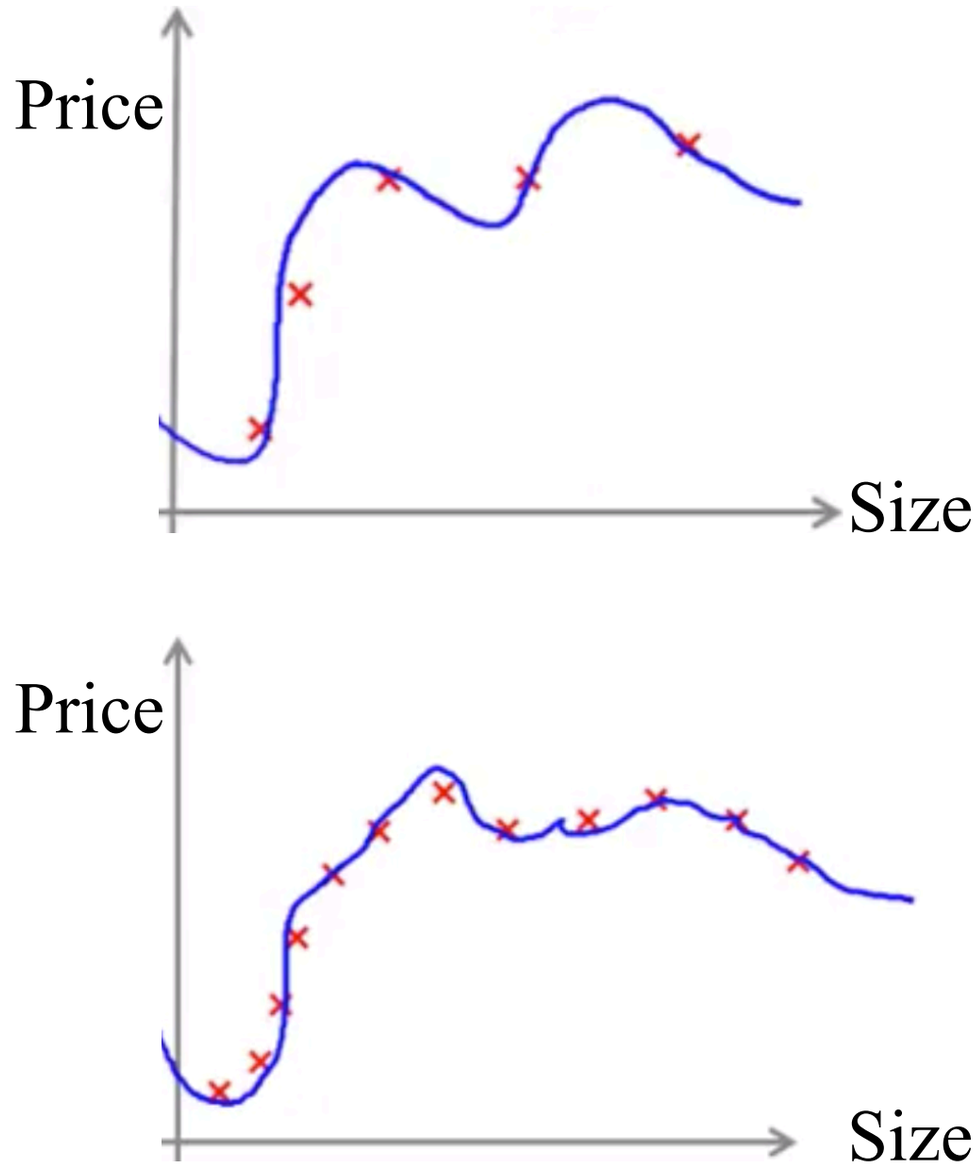
Learning Curves

High variance



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

(and small λ)

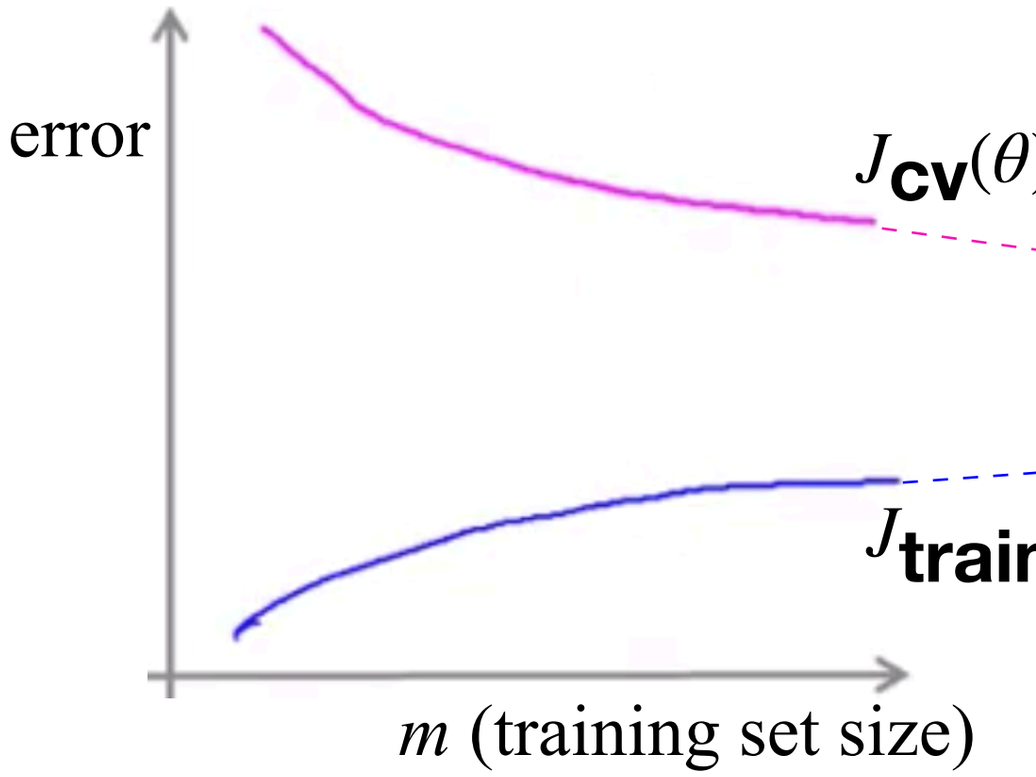


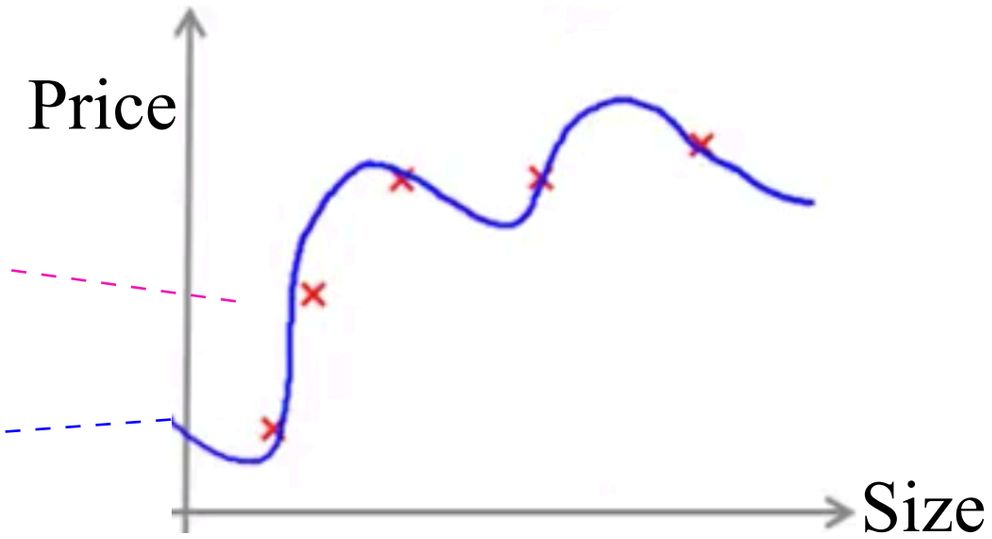
Learning Curves

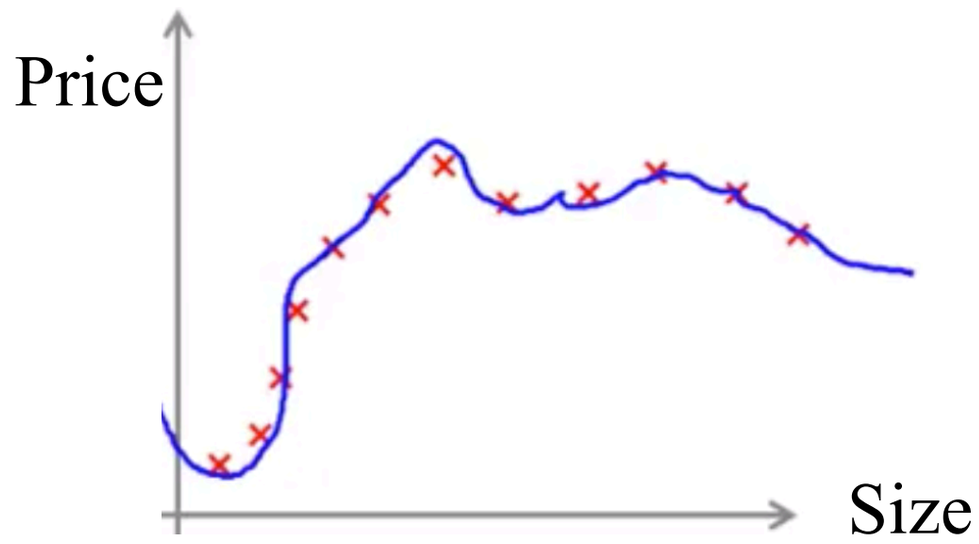
High variance

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

(and small λ)







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

Question

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

(i) Algorithm is suffering from high bias.

(ii) Algorithm is suffering from large variance.

(iii) $J_{\mathbf{cv}}(\theta)$ (cross validation error) is much larger than $J_{\mathbf{train}}(\theta)$ (training error).

(iv) $J_{\mathbf{cv}}(\theta)$ (cross validation error) is about the same as $J_{\mathbf{train}}(\theta)$ (training error).

Deciding what to try next

Motivating Example

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
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features ($x_1^2, x_2^2, x_1 x_2$, *etc.*)
- Try decreasing λ
- Try increasing λ

Motivating Example

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?

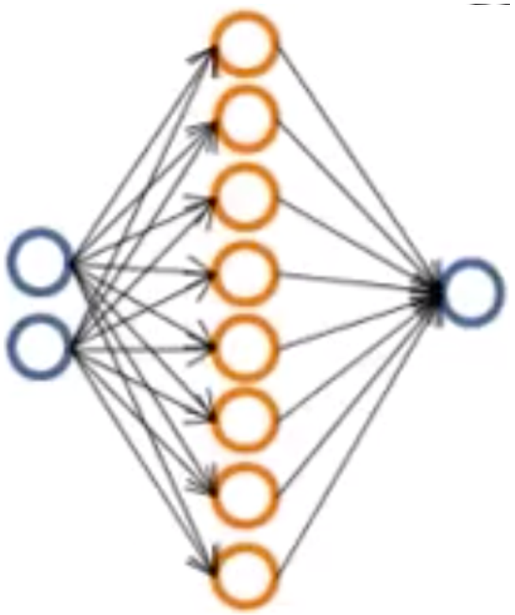
- Get more training examples $\longrightarrow$ fixes high variance
- Try smaller sets of features $\longrightarrow$ fixes high variance
- Try getting additional features $\longrightarrow$ fixes high bias
- Try adding polynomial features ($x_1^2, x_2^2, x_1 x_2$, *etc.*) $\longrightarrow$ fixes high bias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

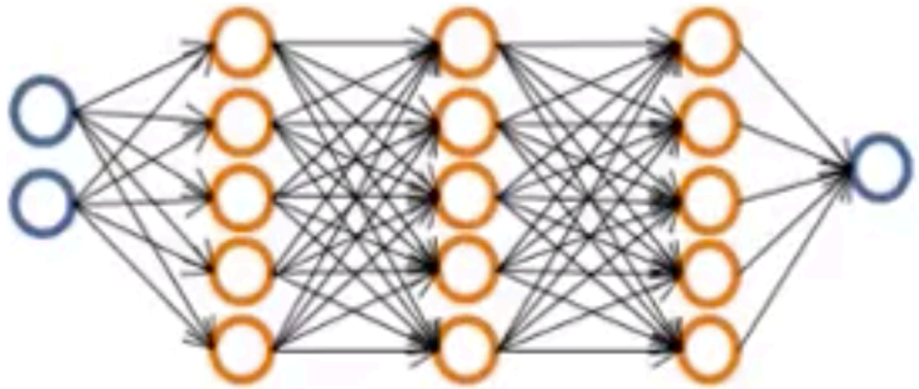
Neural Networks

‘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 (λ) to address overfitting

To decide #hidden layers, compute $J_{\mathbf{cv}}(\theta)$ or $J_{\mathbf{test}}(\theta)$

Question

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

(i) Yes, because this increases the number of parameters and lets the network represent more complex function.

(ii) Yes, because it is currently suffering from high bias.

(iii) No, because it is currently suffering from high bias, so adding hidden units is unlikely to help.

(iv) No, because it is currently suffering from high variance, so adding hidden units is unlikely to help.