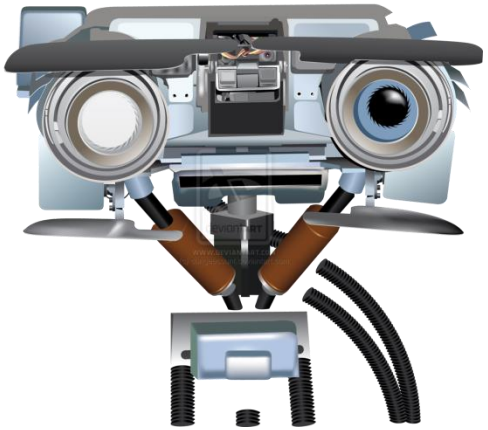


Supervised Learning III

Classification, Regularization

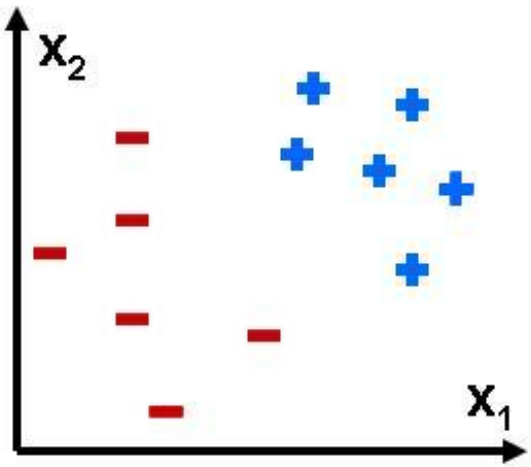
Recall: Logistic Regression

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

θ: parameters

$D = \left(x^{(i)}, y^{(i)}\right)$: data



Cost Function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Goal: minimize cost $\min_\theta J(\theta)$

Gradient descent for Logistic Regression

Cost

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$

(simultaneously update all θ_j)

Gradient descent for Logistic Regression

Cost

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

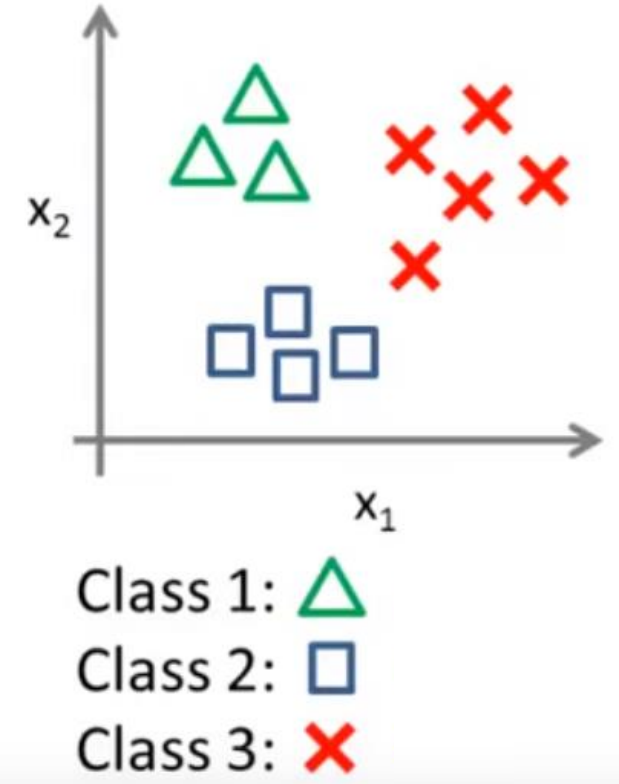
Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$

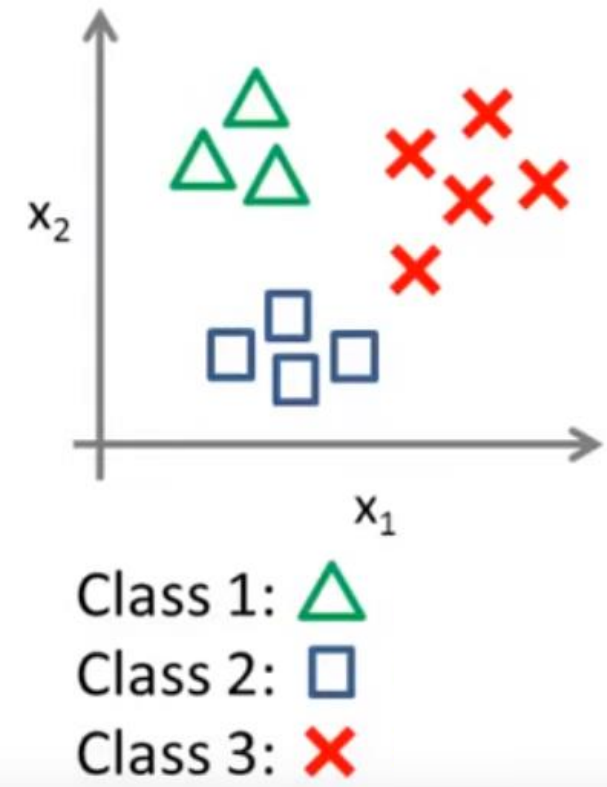
(simultaneously update all θ_j)

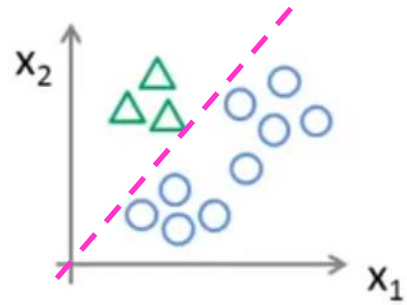
Logistic Regression for Multi-class Classification



Logistic Regression for Multi-class Classification

One-vs-all (One-vs-rest)



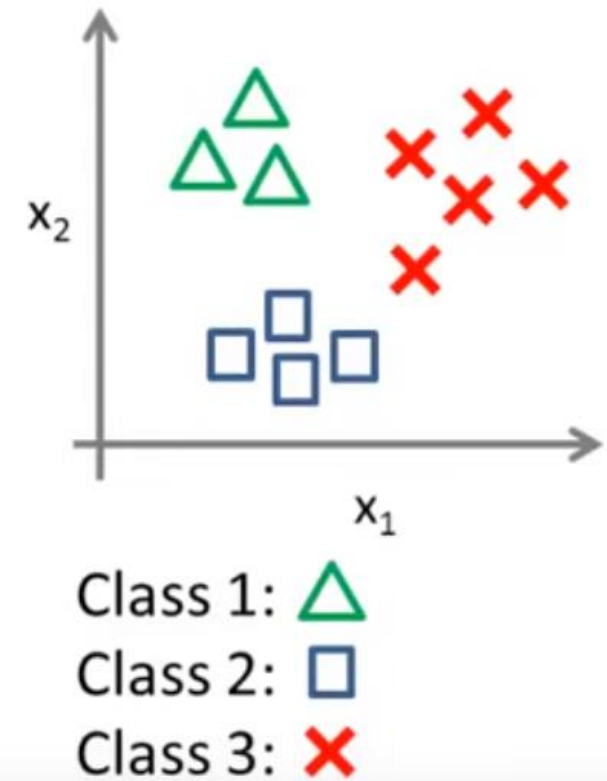


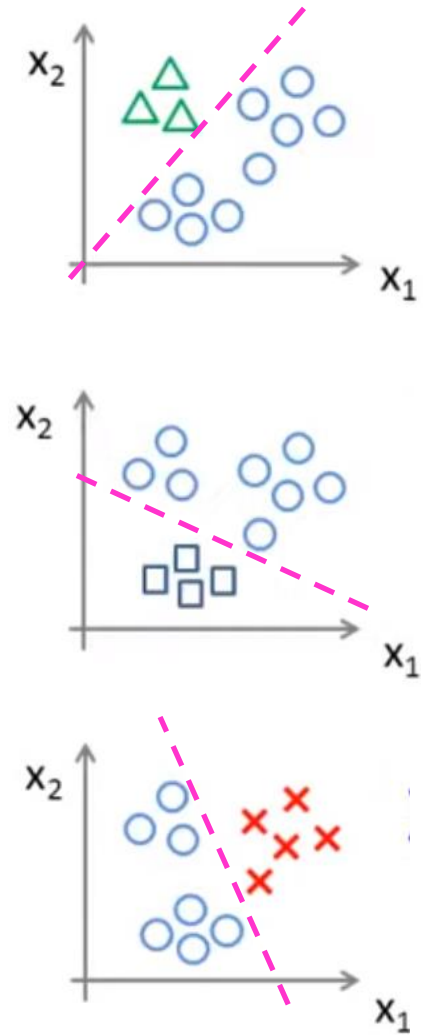
$h_\theta^1(x)$

$P(y = 1|x)$

Logistic Regression for Multi-class Classification

One-vs-all (One-vs-rest)





$h_\theta^1(x)$

$P(y = 1|x)$

$h_\theta^2(x)$

$P(y = 2|x)$

$h_\theta^3(x)$

$P(y = 3|x)$

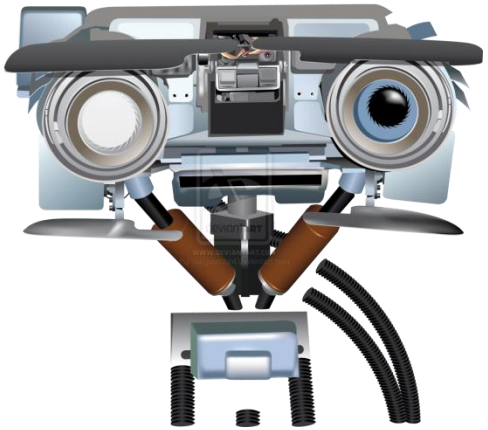
Andrew Ng

Logistic Regression for Multi-class Classification

- Trained a logistic regression classifier $h_\theta^i(x)$ for each class i to predict the probability that $y = i$.

- On a new input x, to make a prediction, pick the class i that maximizes:

$$\max_i h_\theta^i(x)$$

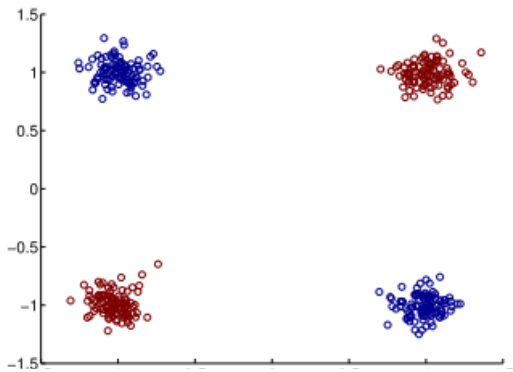


Supervised Learning III

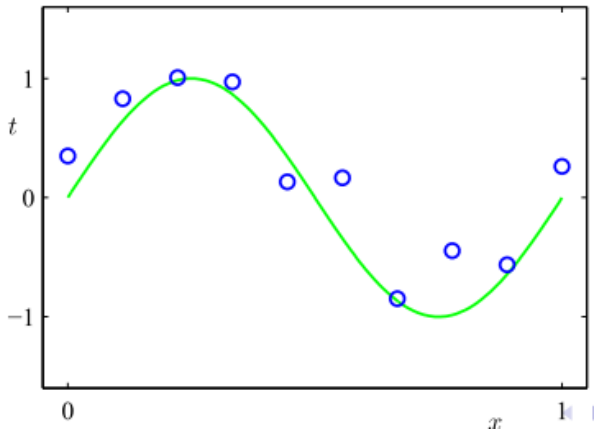
Non-linear features

What to do if data is nonlinear?

Example of nonlinear classification



Example of nonlinear regression

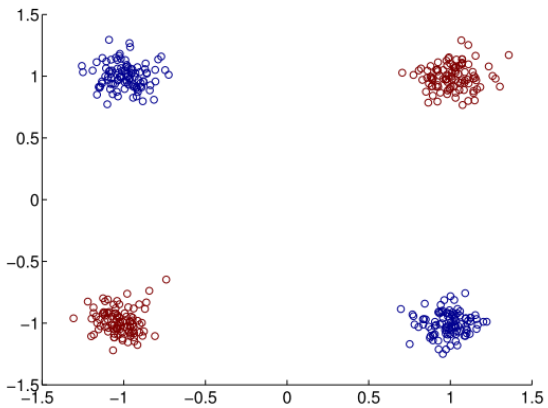


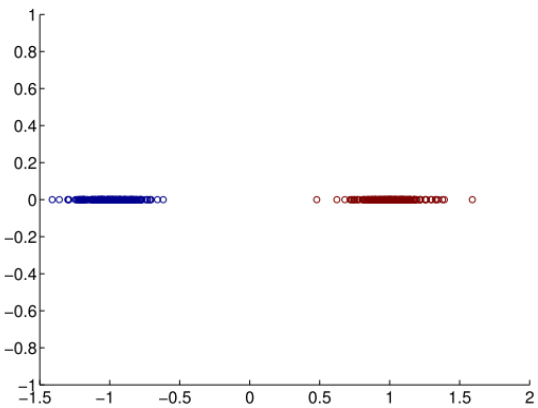
Nonlinear basis functions

Transform the input/feature

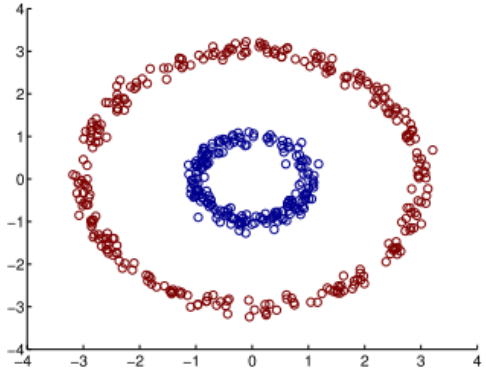
$$\phi(x) : x \in R^2 \rightarrow z = x_1 \cdot x_2$$

Transformed training data: linearly separable!



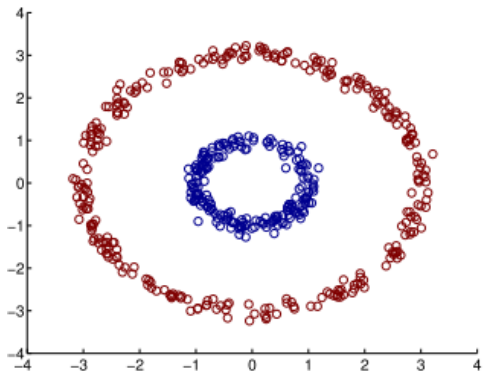


Another example



How to transform the input/feature?

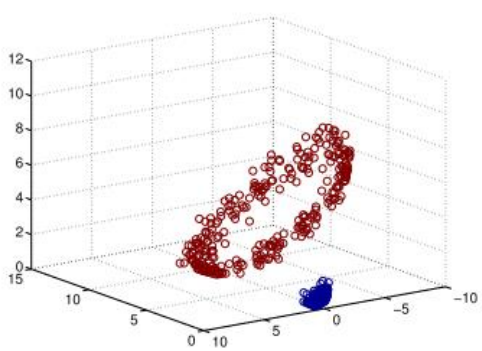
Another example



How to transform the input/feature?

$$\phi(x): x \in R^2 \rightarrow z = \begin{bmatrix} {x_1}^2 \\ x_1 \cdot x_2 \\ {x_2}^2 \end{bmatrix}$$

Transformed training data: linearly separable



Intuition: suppose $\theta = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Then $\theta^T z = {x_1}^2 + {x_2}^2$

i.e., the sq. distance to the origin!

Non-linear basis functions

- We can use a nonlinear mapping, or basis function

$$\phi(x) : x \in R^N \rightarrow z \in R^M$$

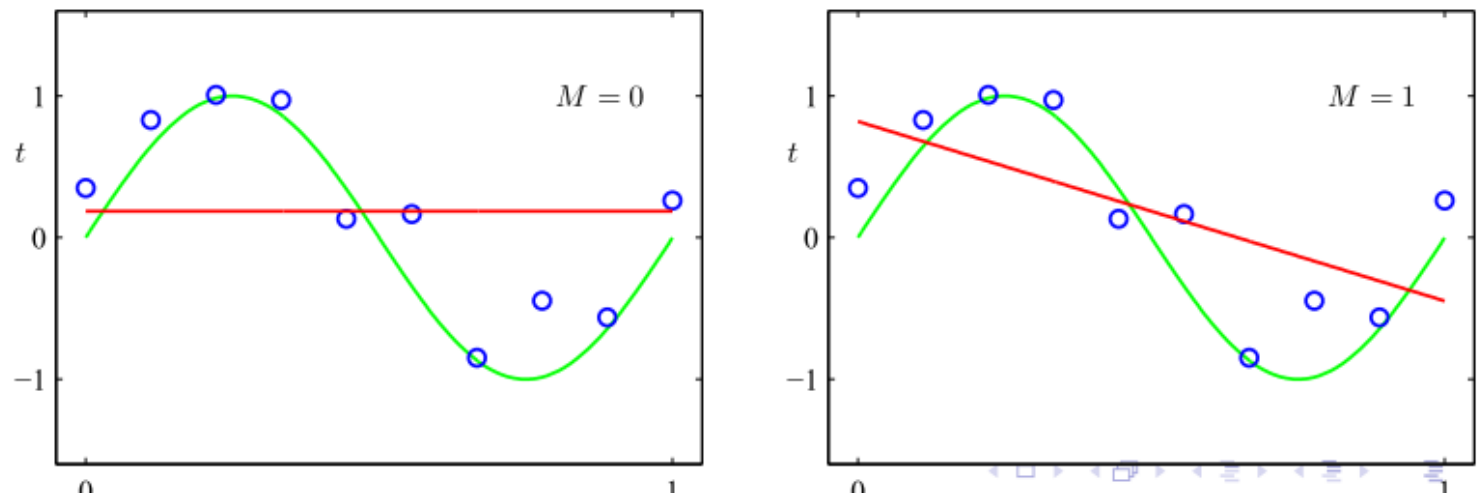
- where M is the dimensionality of the new feature/input z (or $\phi(x)$)
- Note that M could be either greater than D or less than, or the same

Example with regression

Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

Fitting samples from a sine function: *underrfitting* as $f(x)$ is too simple



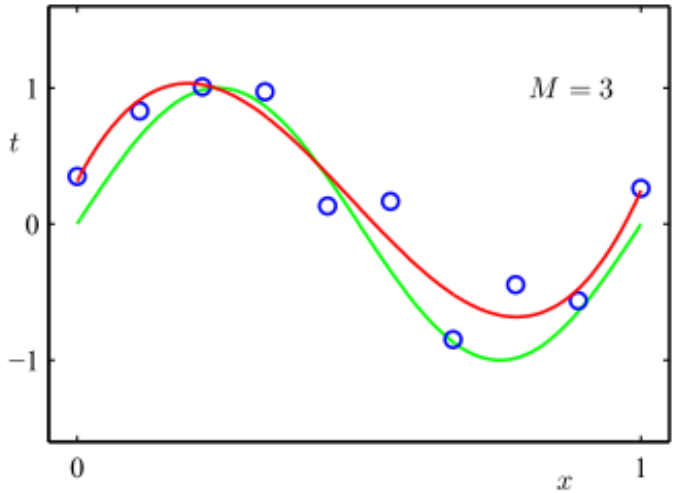
Add more polynomial basis functions

Polynomial basis functions

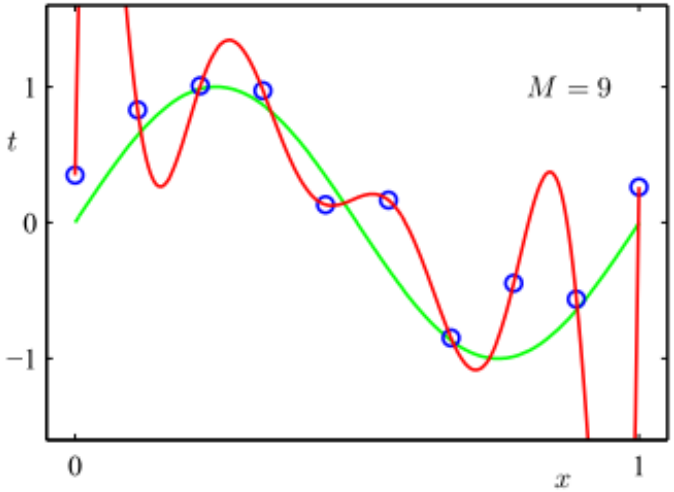
$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

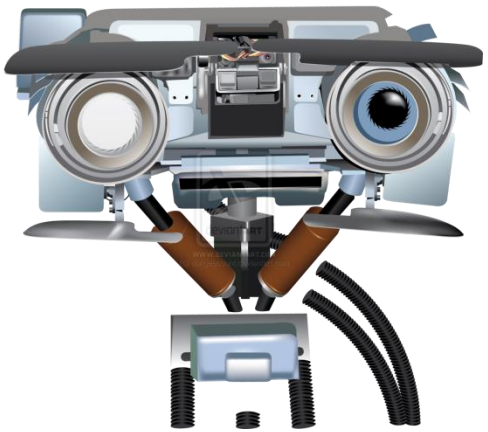
Being too adaptive leads to better results on the training data, but not so great on data that has not been seen!

M=3 *good fit*



M=9: *overfitting*





Supervised Learning III

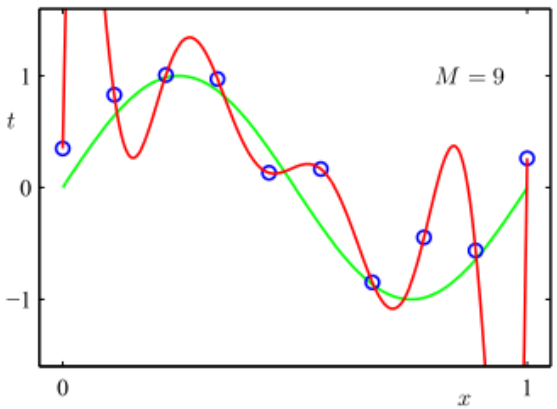
Overfitting

Overfitting

Parameters for higher-order polynomials are very large

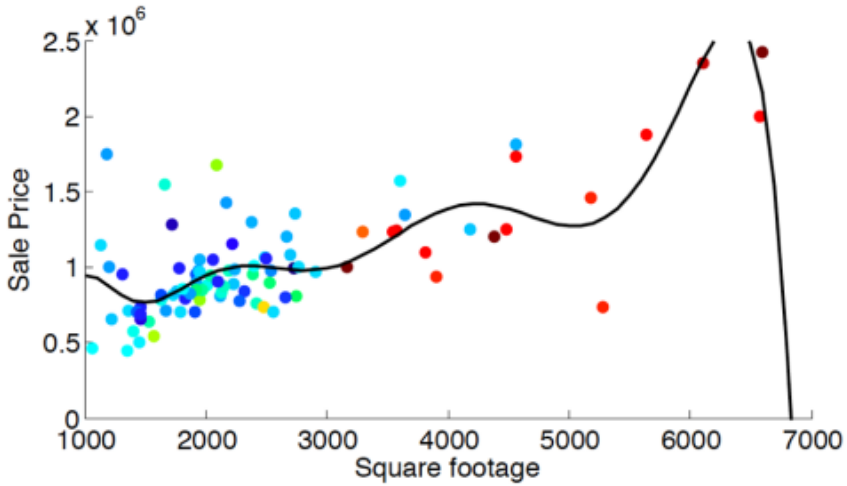
	M =0	M =1	M =3	M =9
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

M=9: *overfitting*



Overfitting disaster

Fitting the housing price data with M = 3

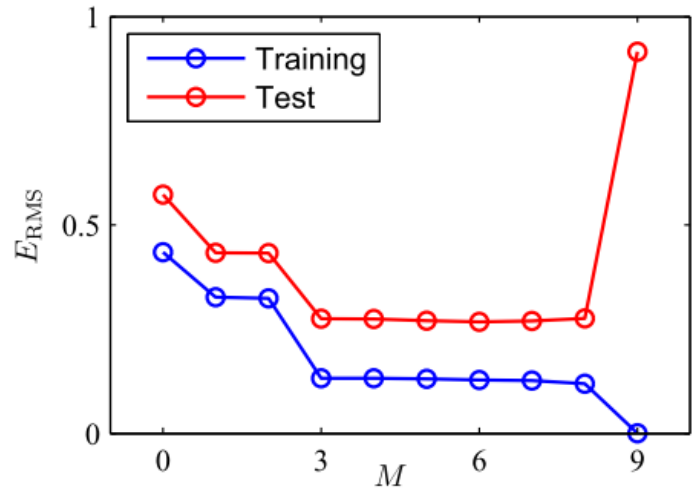


Note that the price would goes to zero (or negative) if you buy bigger houses!
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function on test/train data

As model becomes more complex, performance on training keeps improving while on test data it increases



Horizontal axis: measure of model complexity
In this example, we use the maximum order of the polynomial basis functions.

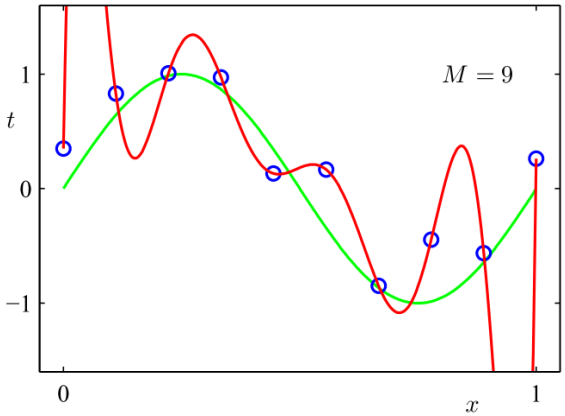
Vertical axis: For regression, it would be SSE or mean SE (MSE)
For classification, the vertical axis would be classification error rate or cross-entropy error function

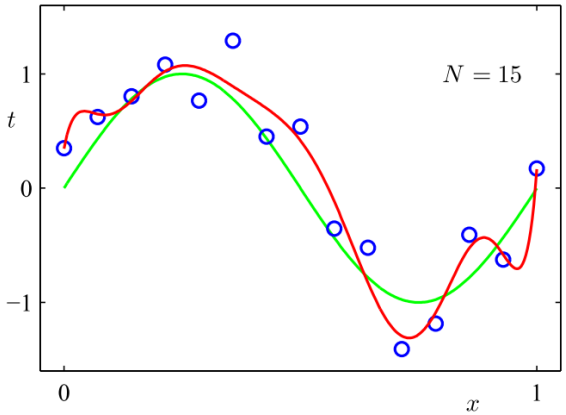
Overcoming overfitting

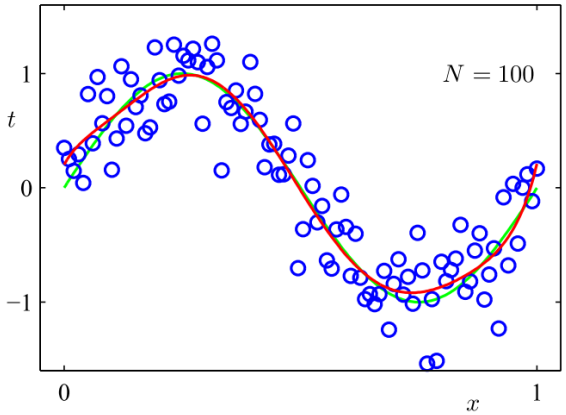
- Basic ideas
  - Use more training data
  - Regularization methods
  - Cross-validation

Solution: use more data

M=9, increase N



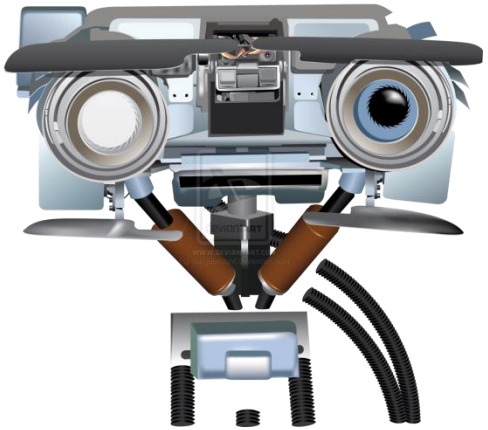




What if we do not have a lot of data?

Overcoming overfitting

- Basic ideas
  - Use more training data
  - Regularization methods
  - Cross-validation

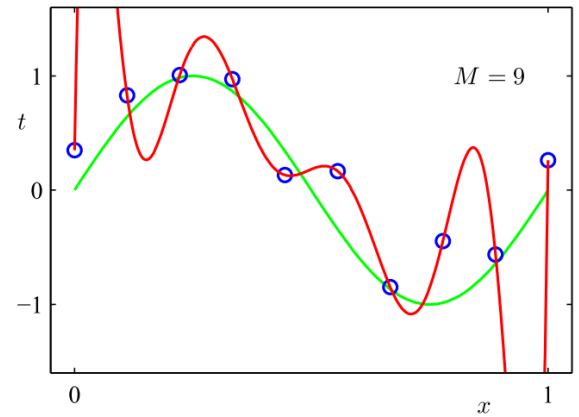


Supervised Learning III

Regularization

Solution: Regularization

- Use regularization:
  - Add $\lambda \|\theta\|_2{}^2$ term to SSE cost function
  - "L-2" norm squared, ie sum of sq. elements $\sum \theta_j^2$
  - Penalizes large θ
  - λ controls amount of regularization

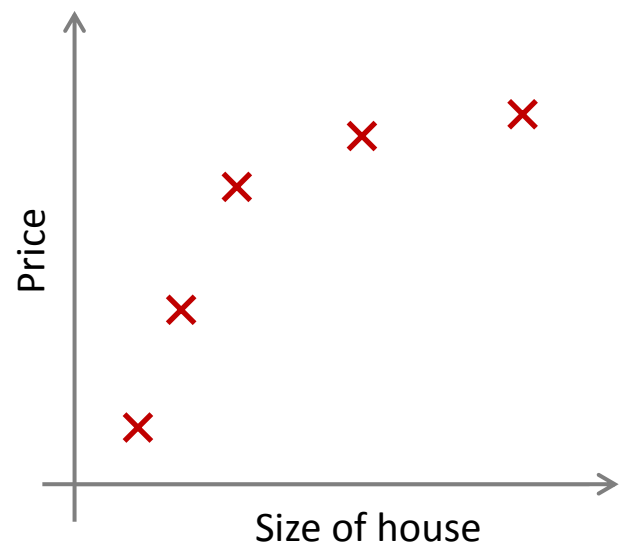


	M =9
θ_0	0.35
θ_1	232.37
θ_2	-5321.83
θ_3	48568.31
θ_4	-231639.30
θ_5	640042.26
θ_6	-1061800.52
θ_7	1042400.18
θ_8	-557682.99
θ_9	125201.43

Regularized Linear Regression

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

$$\min_\theta J(\theta)$$



Gradient descent for Linear Regression

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\boxed{\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\}$

replace with

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

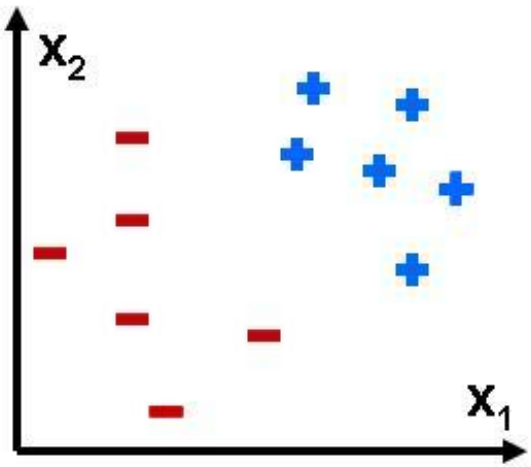
Regularized Logistic Regression

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

θ: parameters

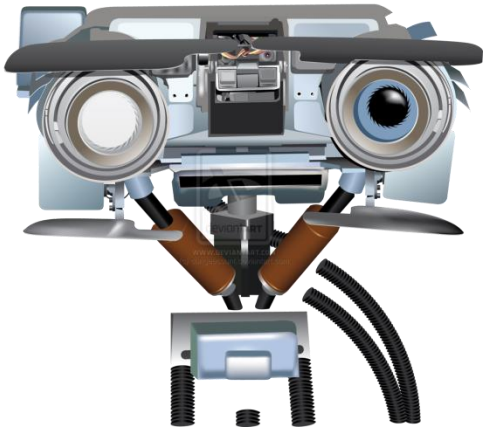
$D = \left(x^{(i)}, y^{(i)}\right)$: data



Cost Function:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right] + \lambda \|\theta\|_2{}^2$$

Goal: minimize cost $\min_\theta J(\theta)$

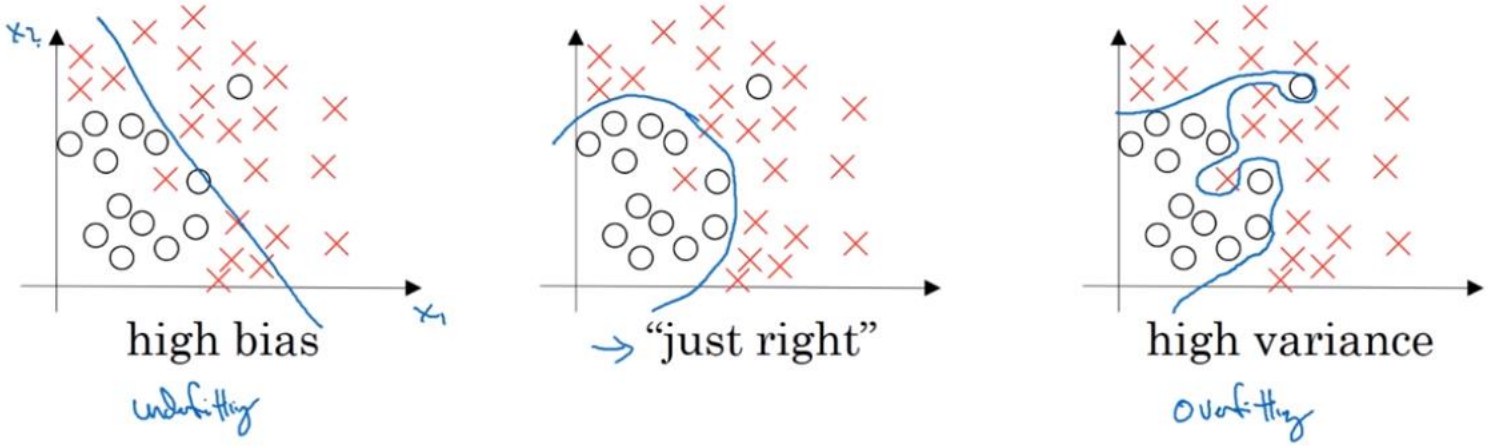


Supervised Learning III

Bias-Variance

Approximation-Generalization Tradeoff

- 2D case:



Not performing well on training data

Not generalizing well from training to unseen data

- For higher dimensional data, the following is used to "visualize" approximation-generalization:
  - Training Set Error
  - Validation Set Error

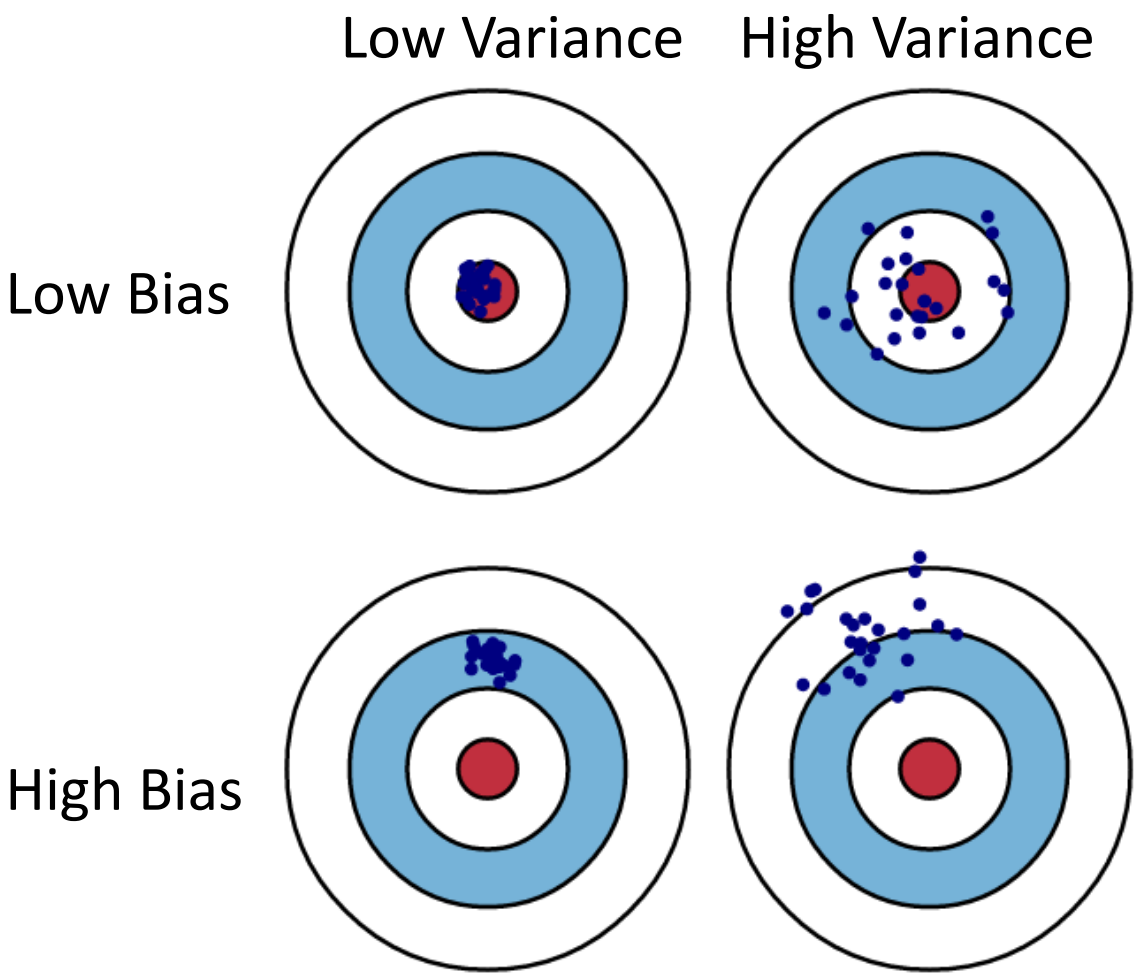
Bias *vs.* Variance

- Understanding how different sources of error lead to bias and variance helps us improve model fitting. ***Imagine you could repeat the whole model fitting process on many datasets.***

- **Error due to Bias**: The error due to bias is taken as the difference between the expected (or average) prediction of our model and the correct value which we are trying to predict.

- **Error due to Variance**: The variance is how much the predictions for a given point vary between different realizations of the model.

Graphical Illustration



http://scott.fortmann-roe.com/docs/BiasVariance.html

The Bias-Variance Trade-off

Hence, there is a trade-off between bias and variance:

- Less complex models (fewer parameters) have high bias and hence low variance
- More complex models (more parameters) have low bias and hence high variance
- Optimal model will have a balance

Which is worse?

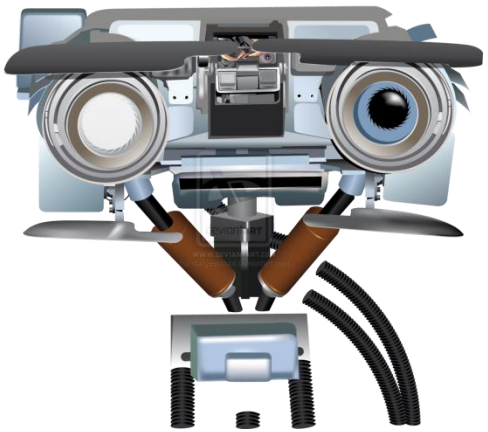
- A gut feeling many people have is that they should minimize bias even at the expense of variance.

- This is mistaken logic. It is true that a high variance and low bias model can preform well in some sort of long-run average sense. However, in practice modelers are always dealing with a single realization of the data set.

- In these cases, long run averages are irrelevant, bias and variance are equally important, and one should not be improved at an excessive expense to the other.

How to deal with bias/variance

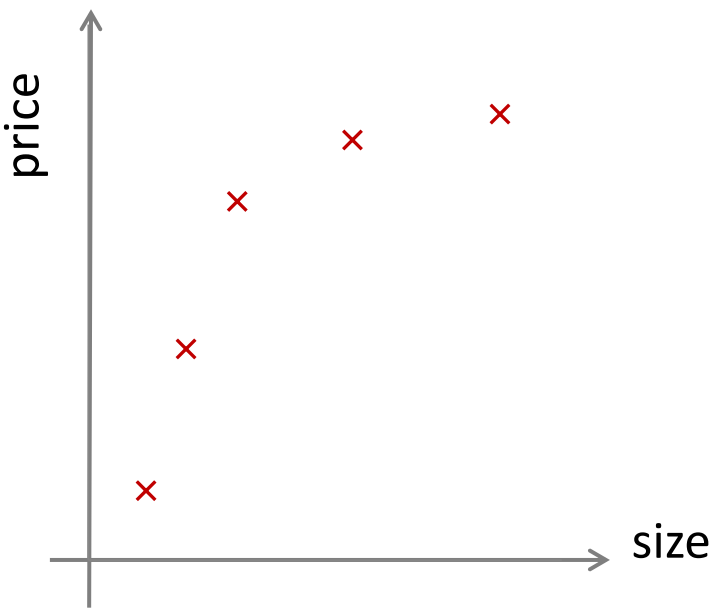
- Can deal with variance by
  - Bagging, e.g. Random Forest
  - Bagging trains multiple models on random subsamples of the data, averages their prediction
- Can deal with high bias by
  - Decreasing regularization/increasing complexity of model
  - Also known as *model selection*



Supervised Learning III

Model selection and training/validation/test sets

Overfitting example



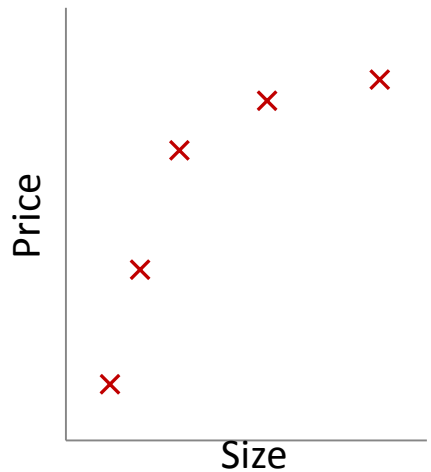
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Once parameters $(\theta_0, \theta_1, \dots, \theta_4)$ 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.

One solution is to regularize, but how can we choose the regularization weight λ?

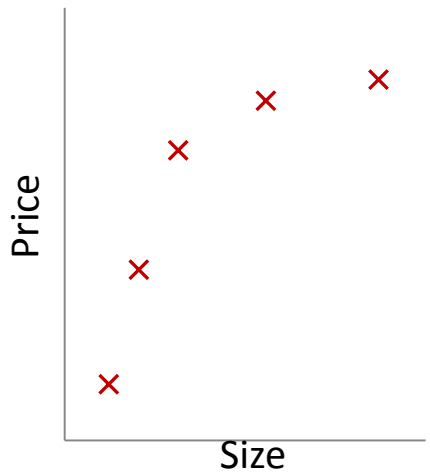
Choosing weight λ

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



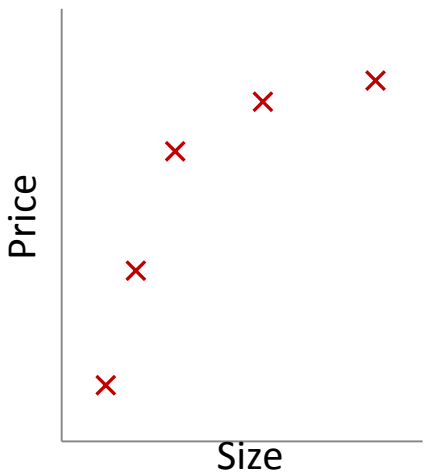
$\lambda = 100$

High bias
(underfit)



$\lambda = 1$

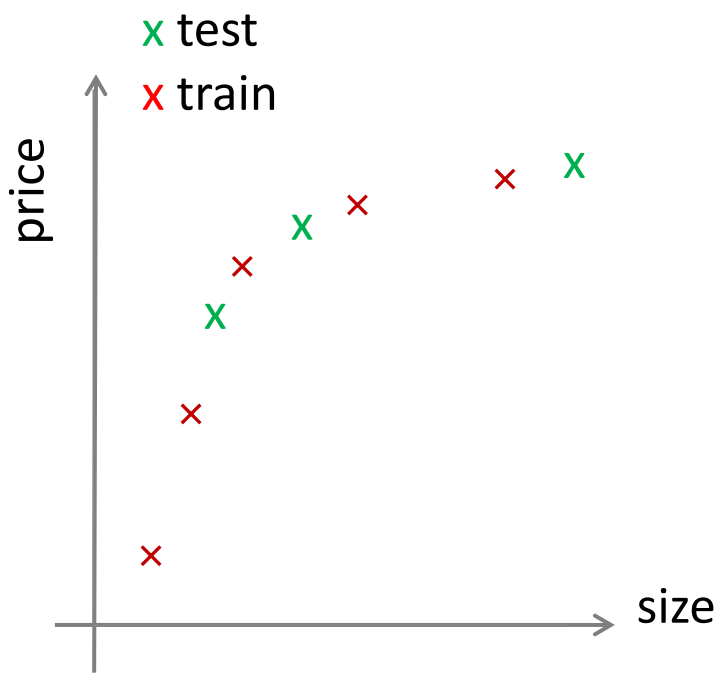
"Just right"



$\lambda = 0.01$

High variance
(overfit)

Model selection



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

Hyperparameters (e.g., degree of polynomial, regularization weight, learning rate) must be selected prior to training.

How to choose them?

Try several values, choose one with the lowest test error?

Problem: test error is likely an overly optimistic estimate of generalization error because we “cheat” by fitting the hyperparameter to the actual test examples.

Train/Validation/Test Sets

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

Solution: split data into three sets.

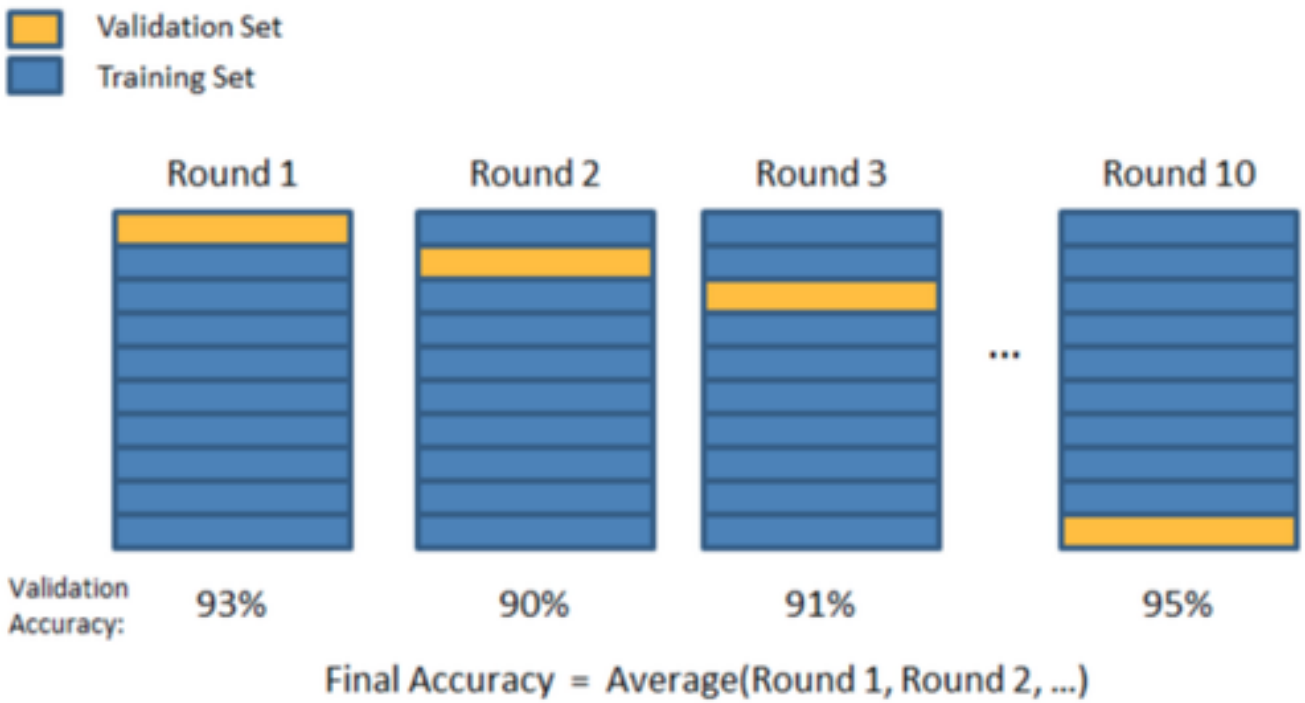
For each value of a hyperparameter, train on the train set, evaluate learned parameters on the validation set.

Pick the model with the hyper parameter that achieved the lowest validation error.

Report this model's test set error.

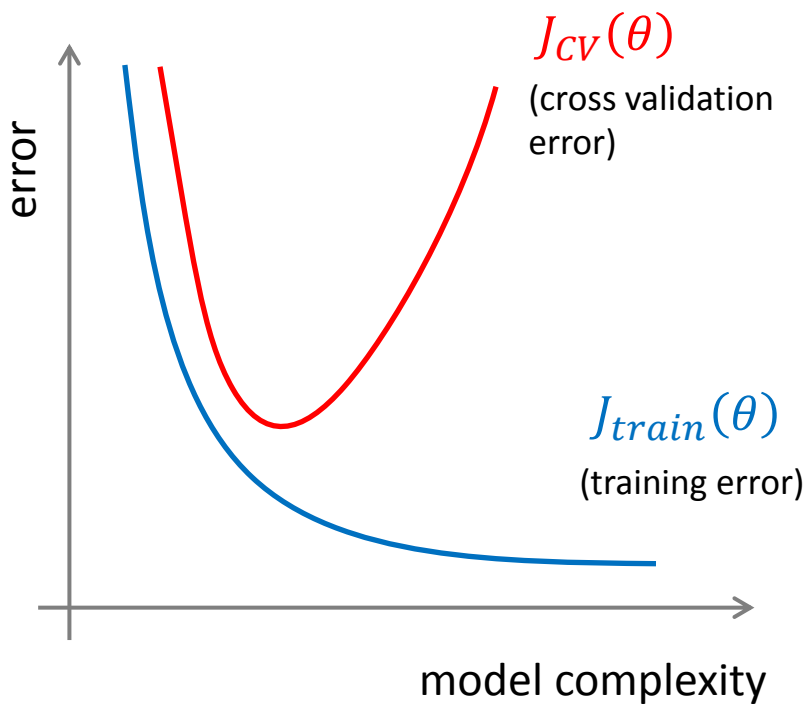
N-Fold Cross Validation

- What if we don't have enough data for train/test/validation sets?
- Solution: use N-fold cross validation.
- Split training set into train/validation sets N times.
- Report average predictions over N val sets, e.g. N=10:



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,

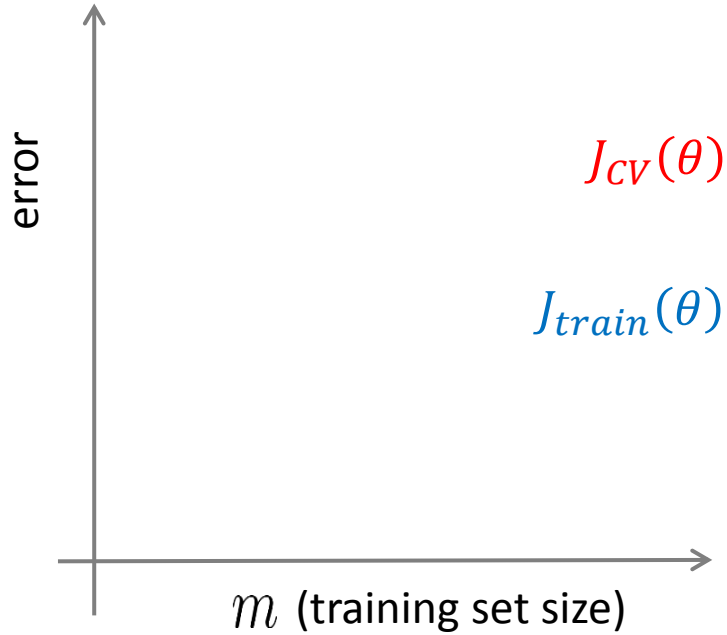
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):

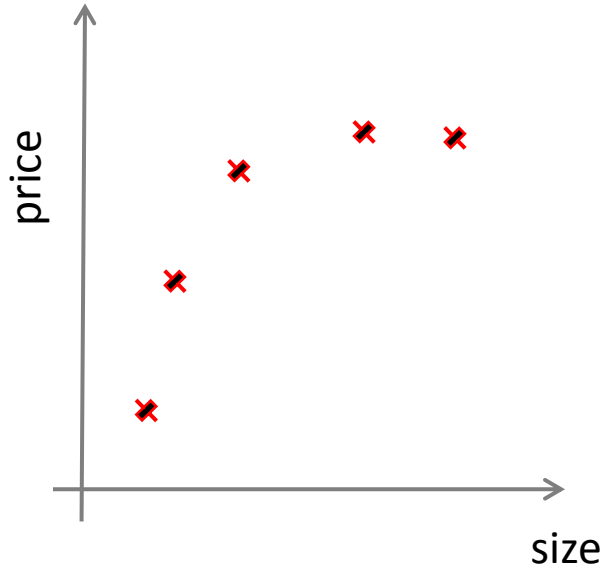
$J_{train}(\theta)$ will be low,

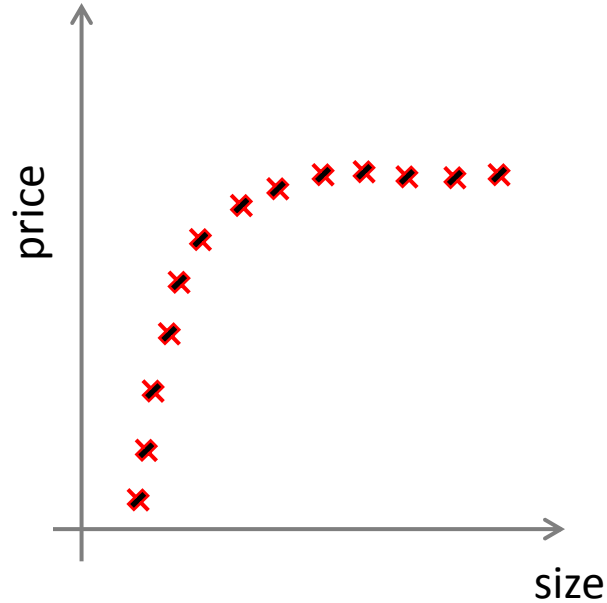
$J_{cv}(\theta) >> J_{train}(\theta)$

Learning Curves: High bias



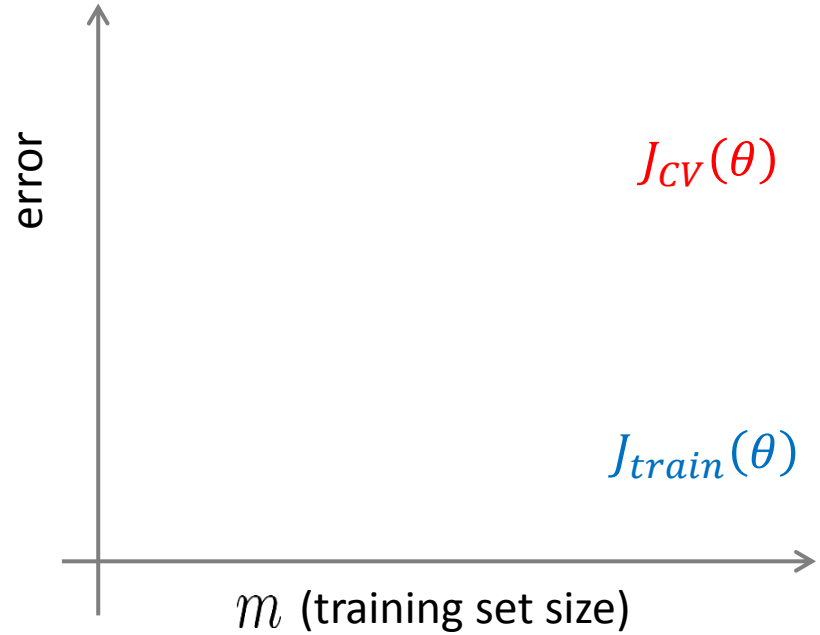
$$h_\theta(x) = \theta_0 + \theta_1 x$$





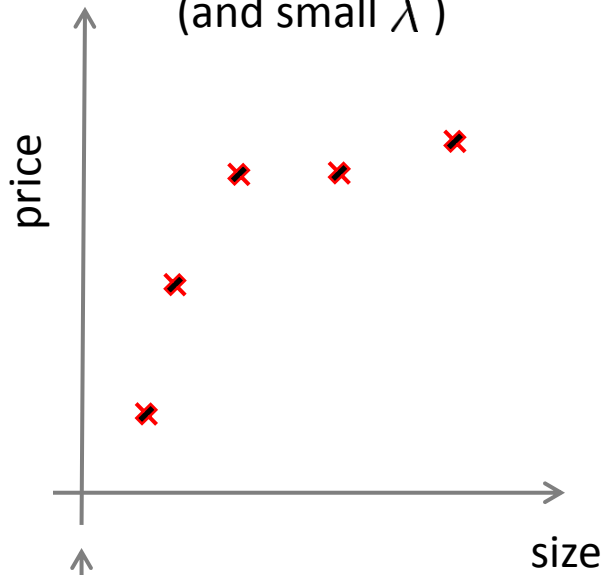
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

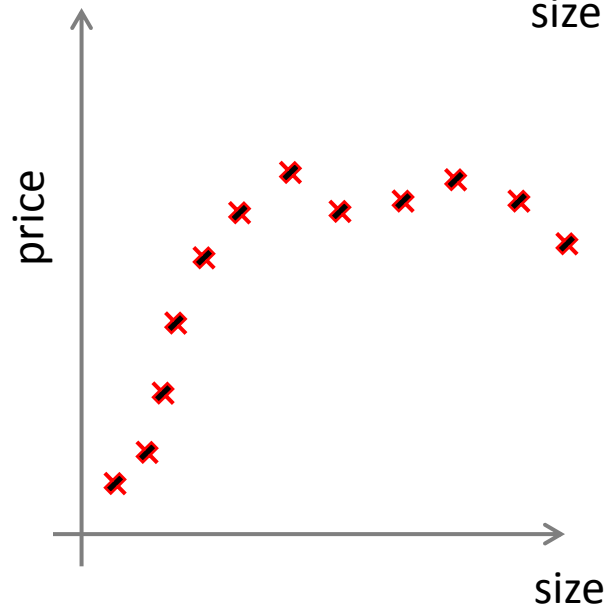
Learning Curves: High variance



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

(and small λ)





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

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?

To fix high variance

- Get more training examples
- Try smaller sets of features
- Try increasing λ

To fix high bias

- Try getting additional features
- Try adding polynomial features
- Try decreasing λ