Machine Learning

CS 539

Worcester Polytechnic Institute

Department of Computer Science

Instructor: Prof. Kyumin Lee

HW1

- https://canvas.wpi.edu/courses/58900/assignments/355140

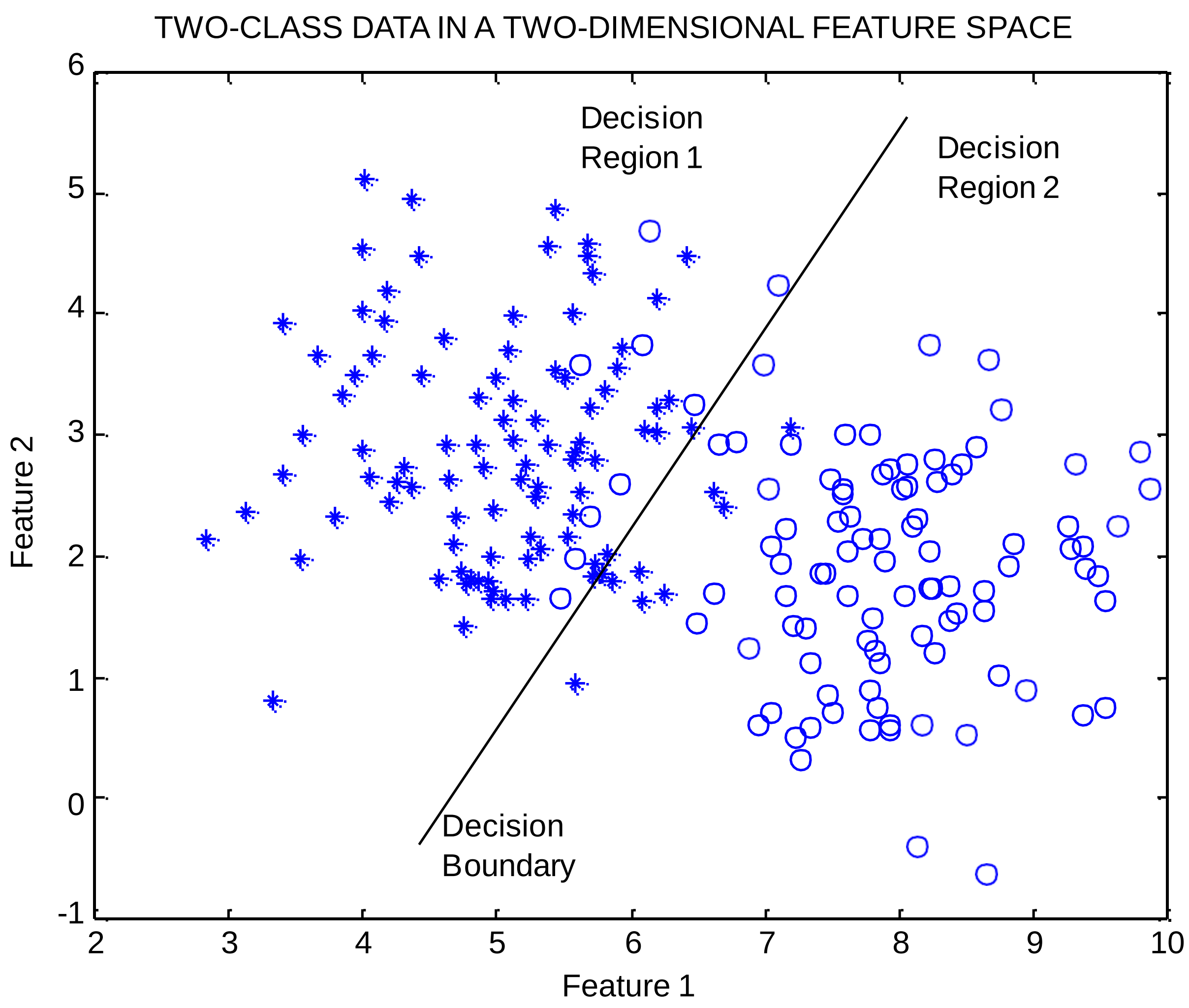
- Due date is June 6th 11:59pm.

Training Data and Test Data

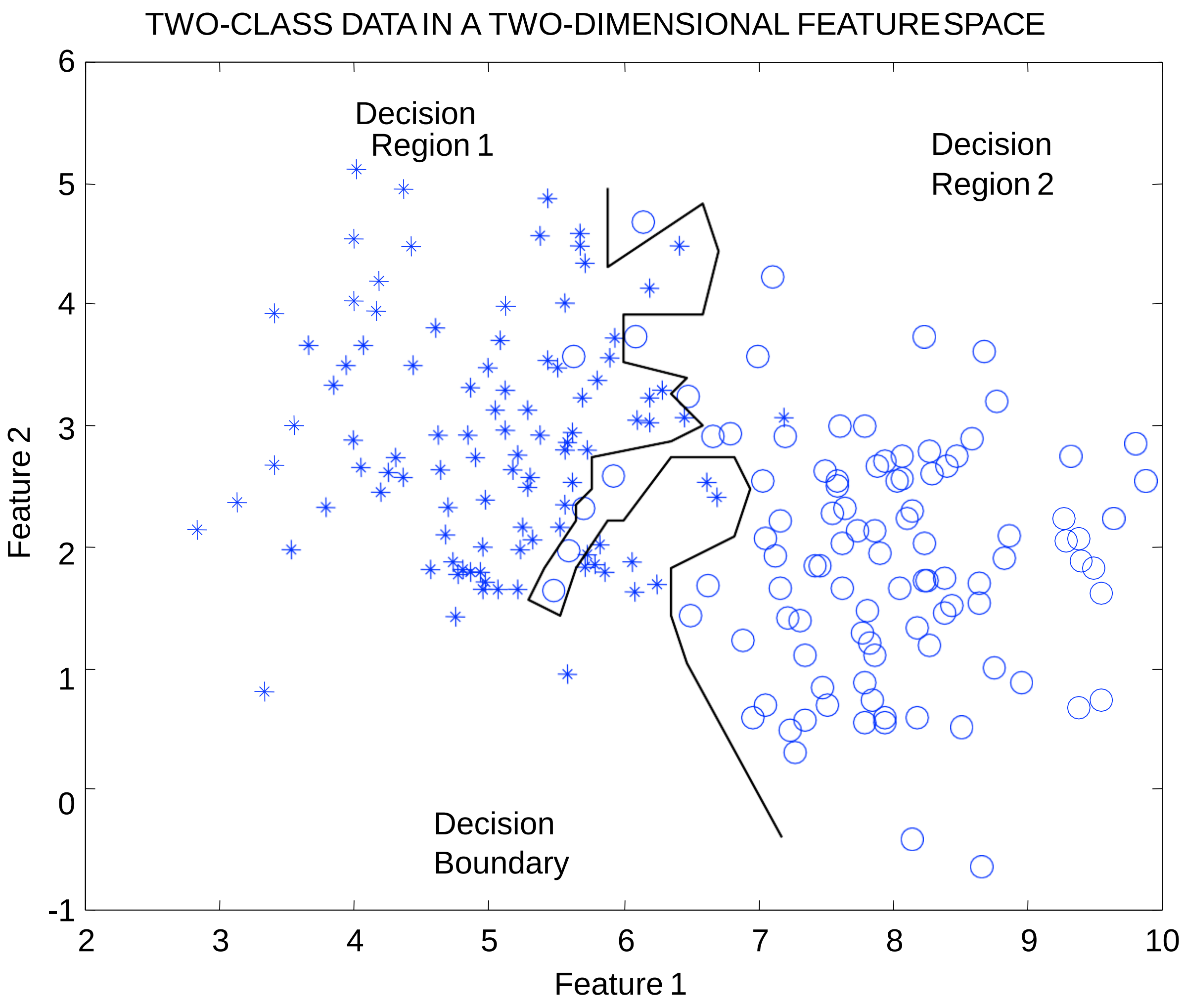
- Training data: data used to build the model
- Test data: new data, not used in the training process

- Training performance is often a poor indicator of generalization performance
  - Generalization is what we really care about in ML
  - Easy to overfit the training data
  - Performance on test data is a good indicator of generalization performance
  - i.e., test accuracy is more important than training accuracy

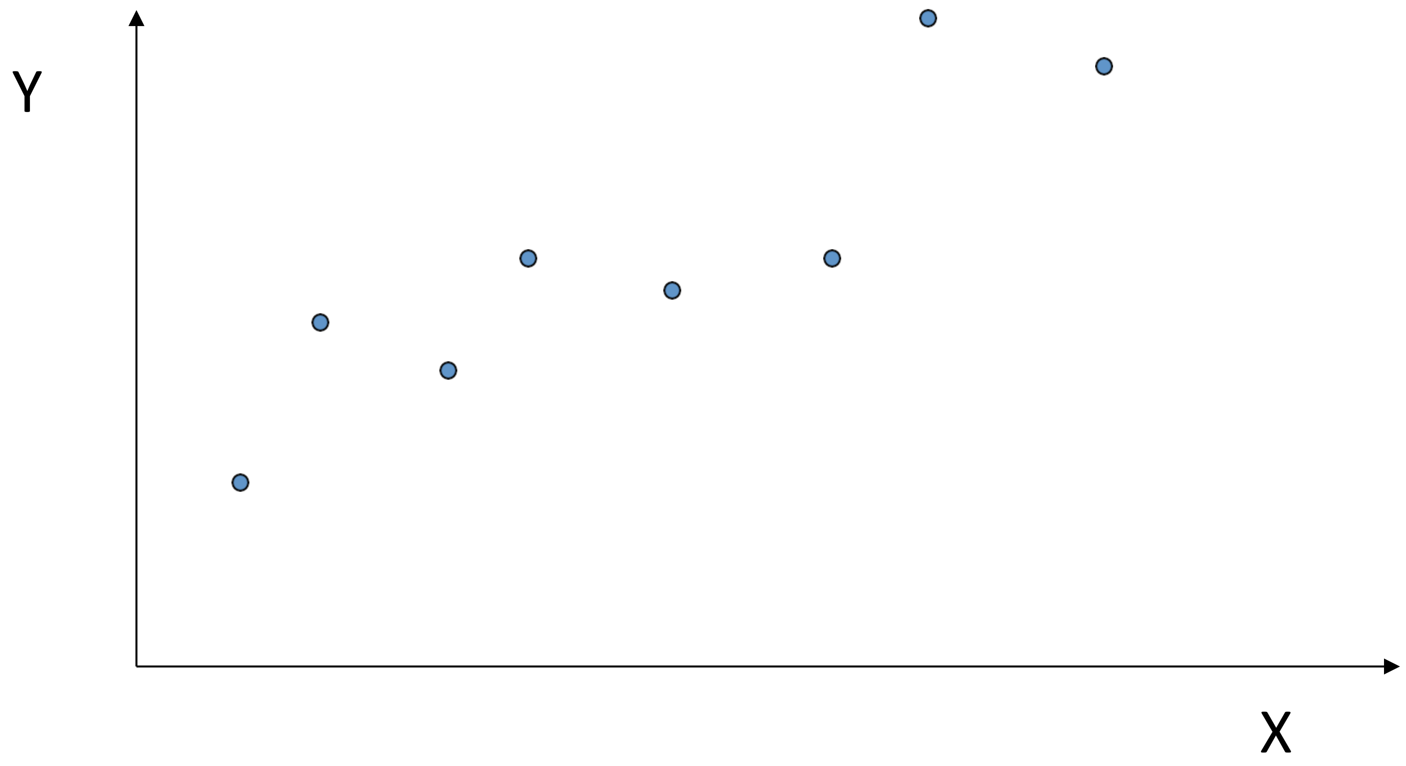
Simple Decision Boundary



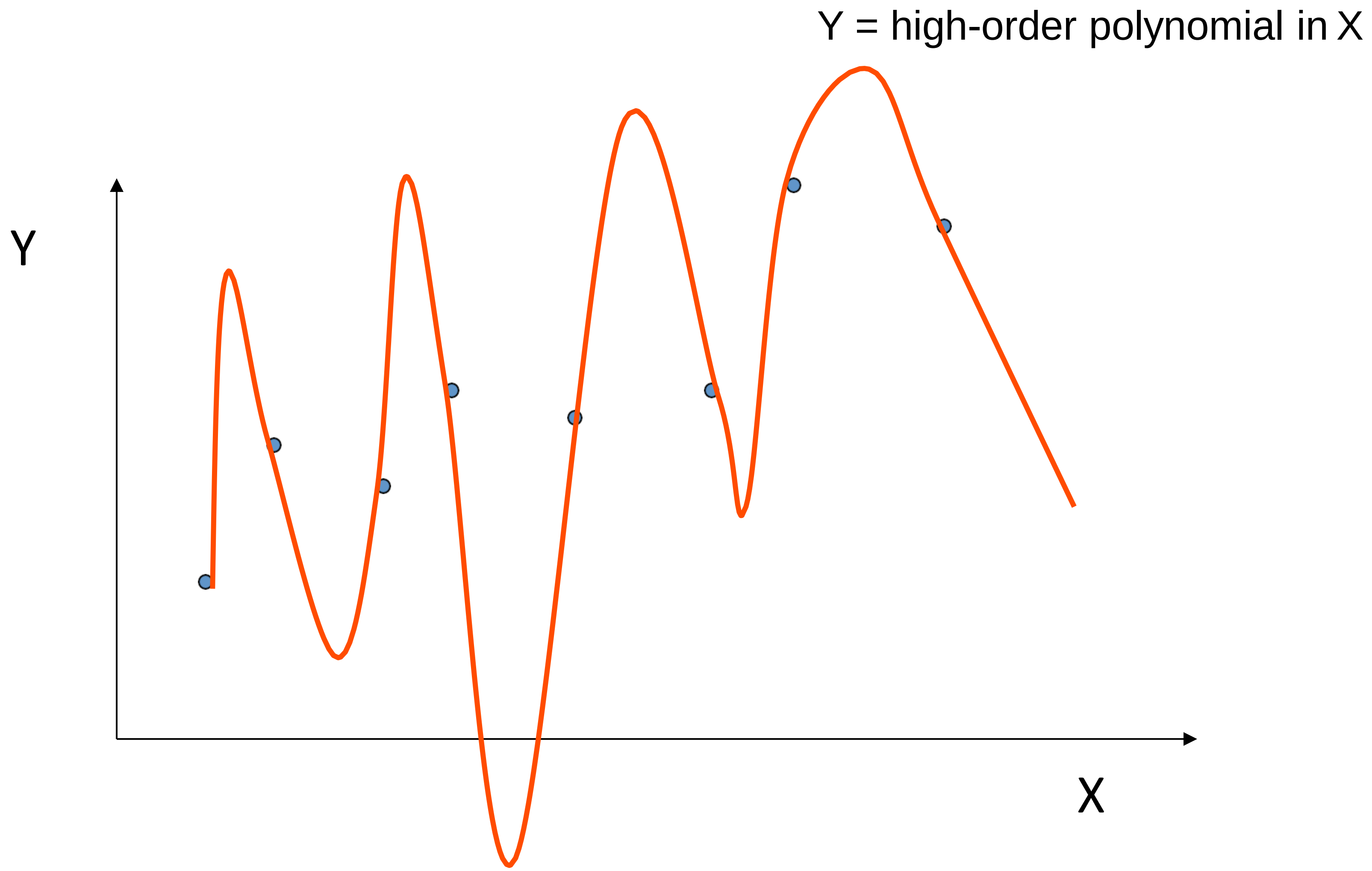
More Complex Decision Boundary



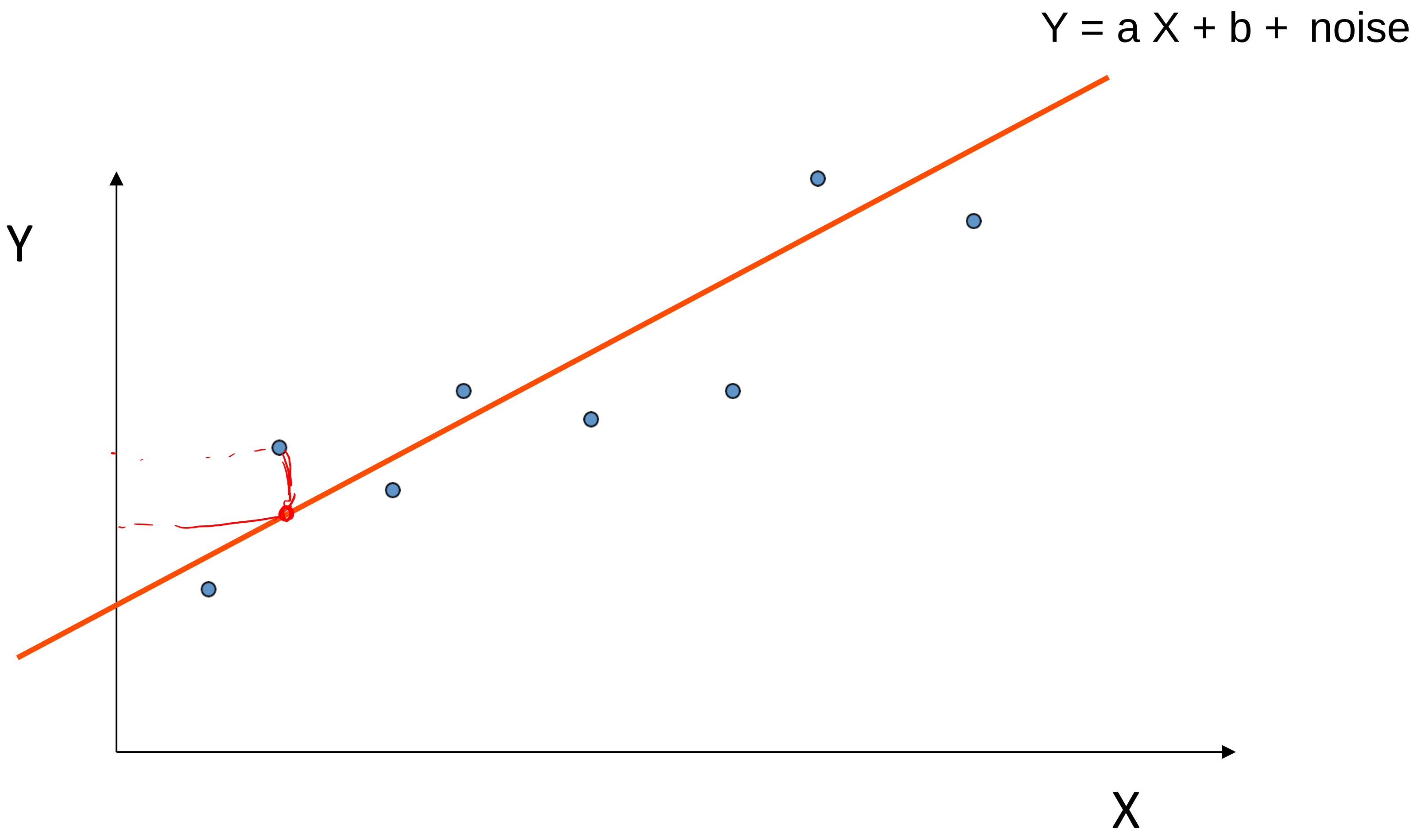
Example: The Overfitting Phenomenon



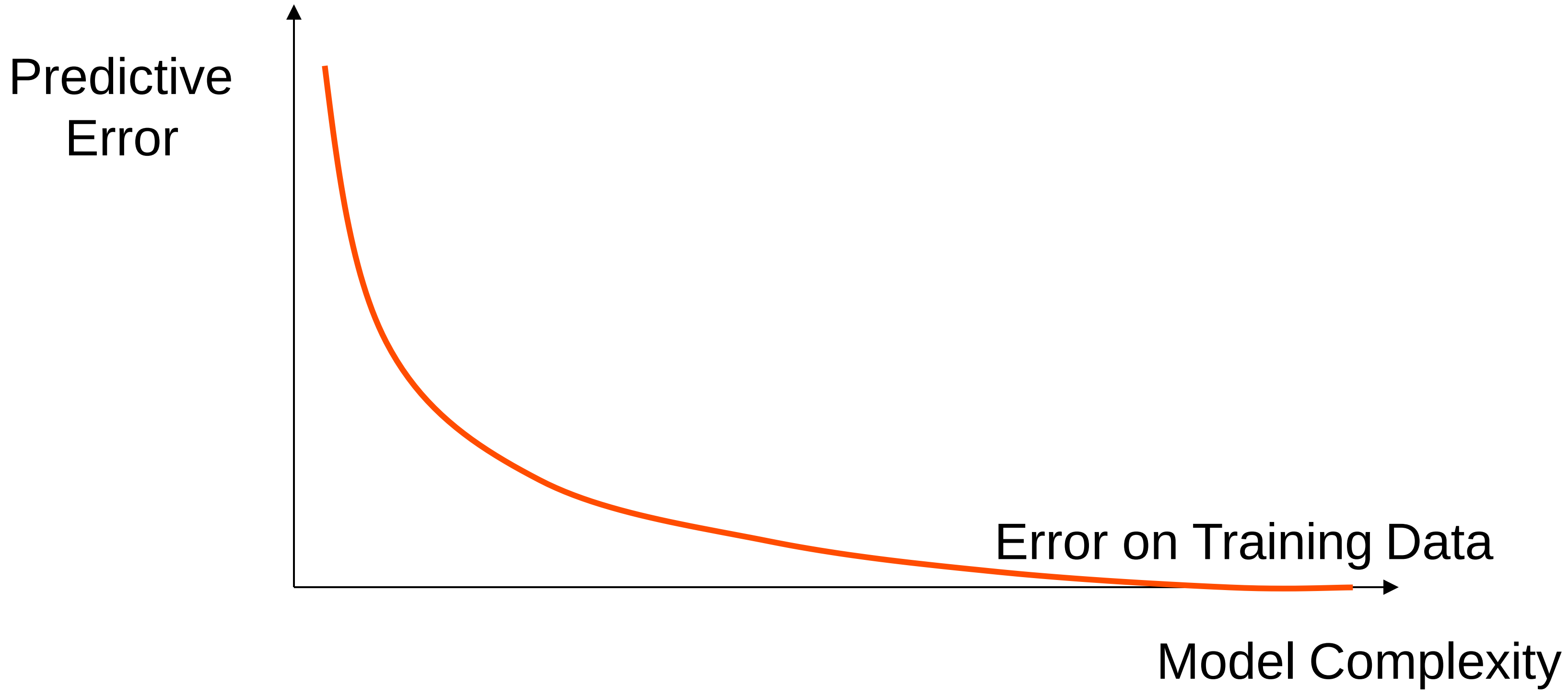
A Complex Model



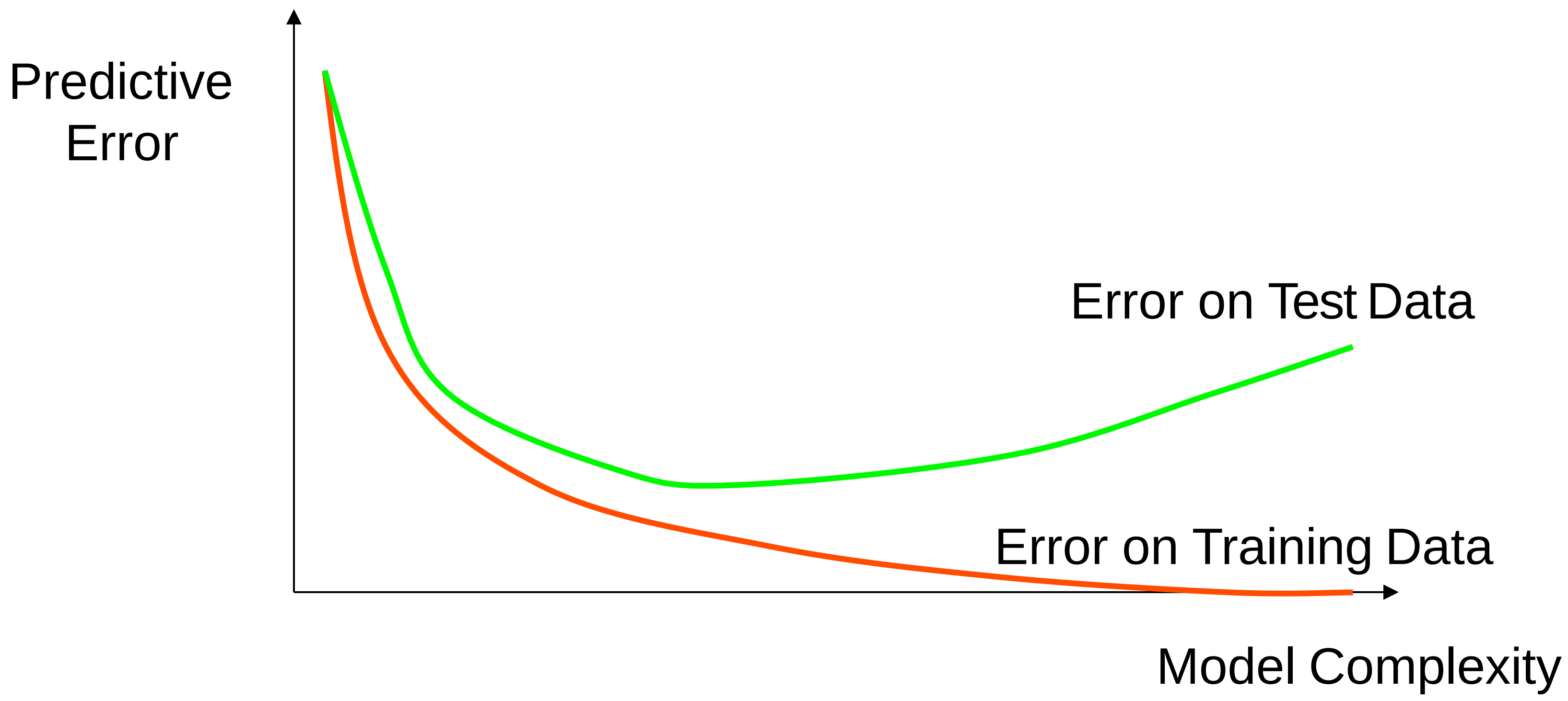
The True (simpler) Model



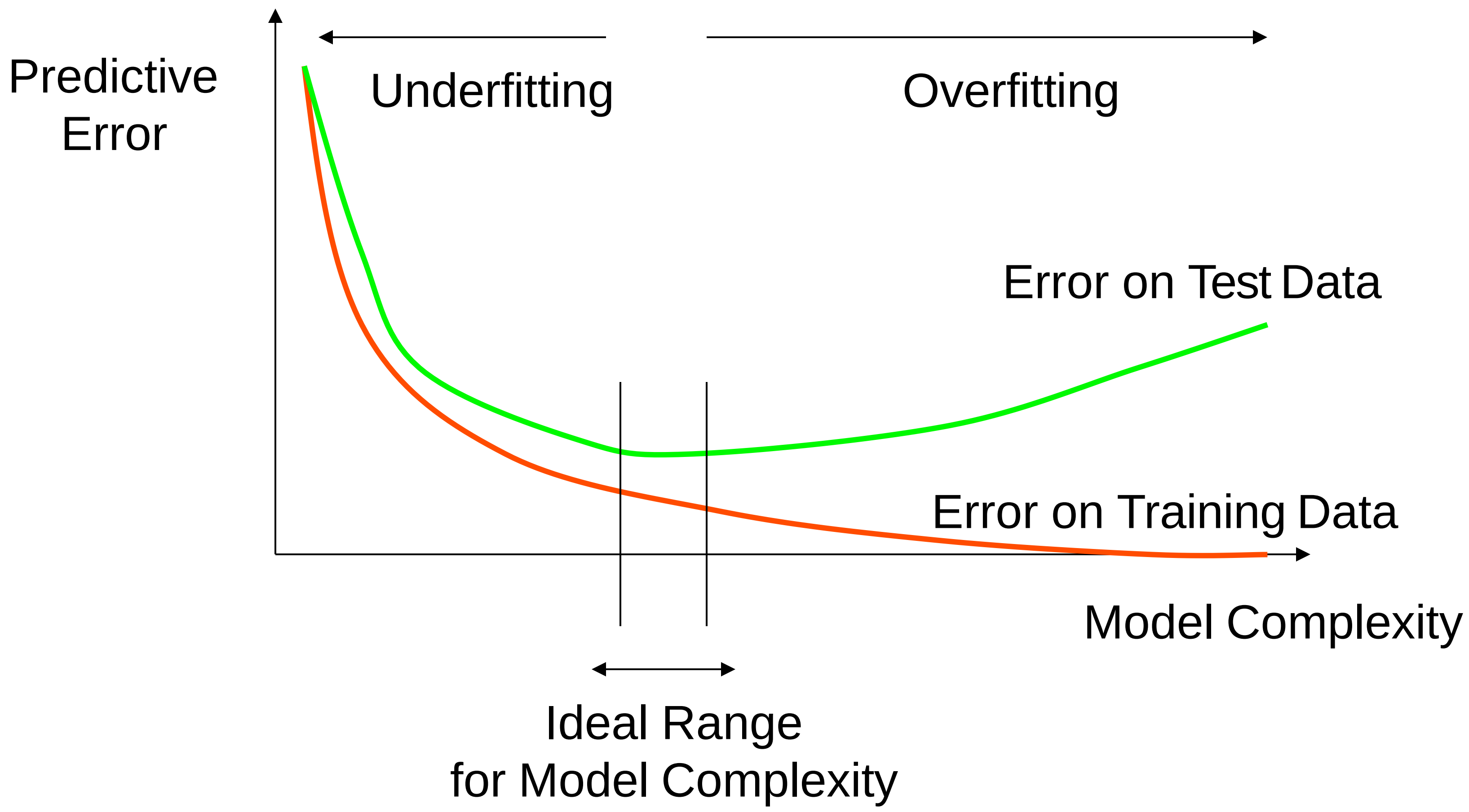
How Overfitting Affects Prediction



How Overfitting Affects Prediction



How Overfitting Affects Prediction



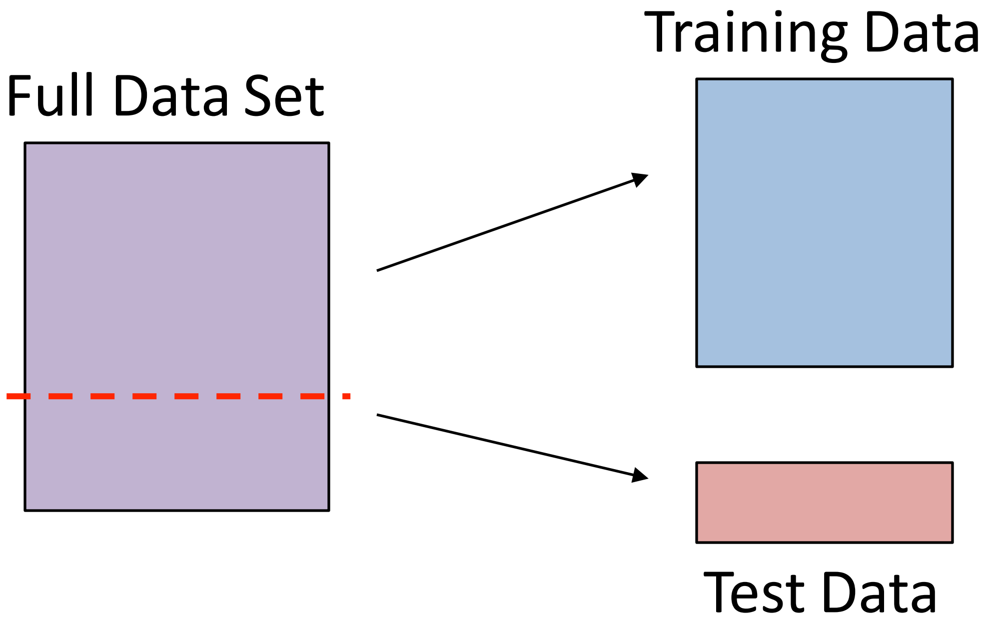
Comparing Classifiers

- Say we have two classifiers, *C1* and *C2*, and want to choose the best one to use for future predictions

- Can we use training accuracy to choose between them?
- No!

- Instead, choose based on test accuracy...

Training and Test Data



Idea:

Train each model on the "training data"...

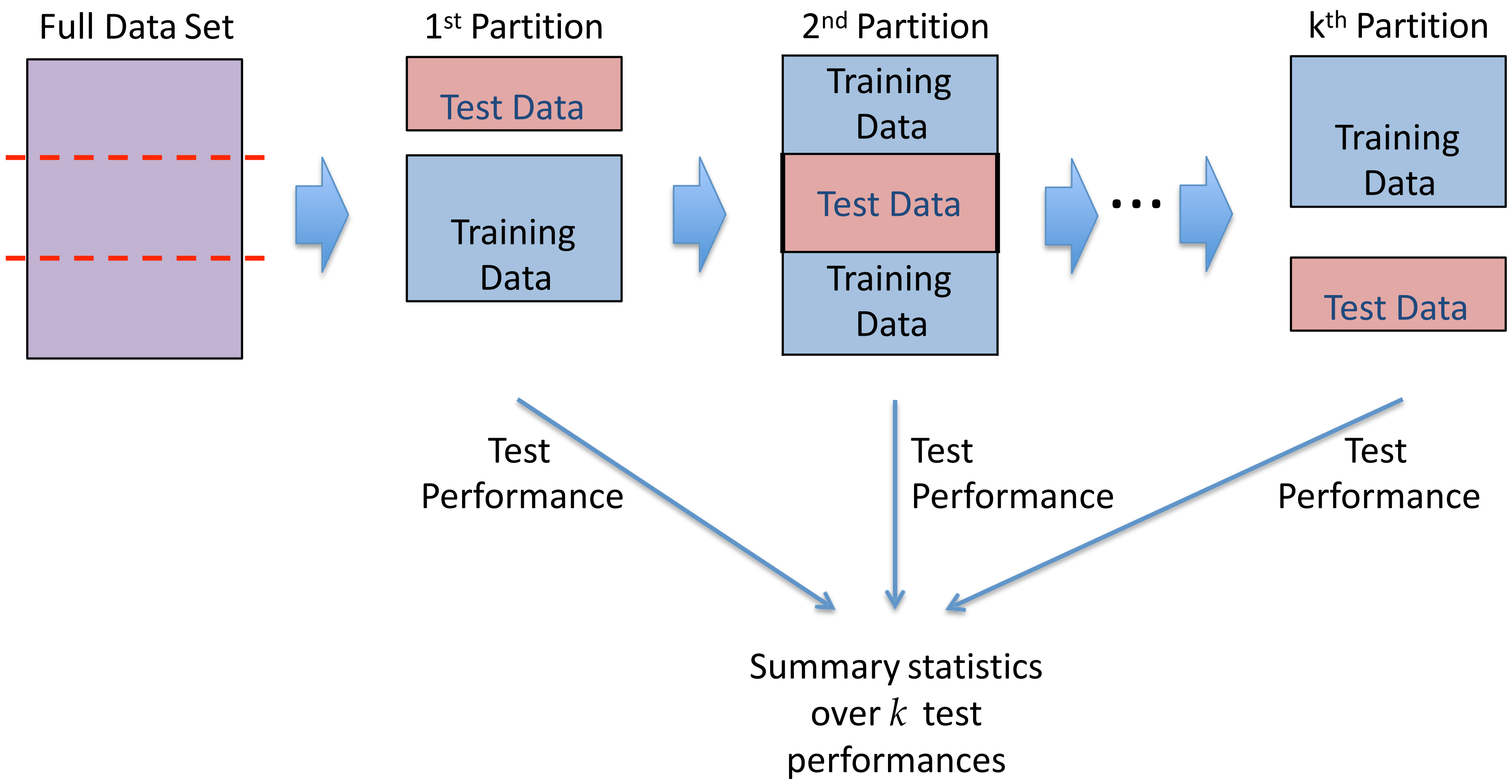
...and then test each model's accuracy on the test data

k-Fold Cross-Validation

- Why just choose one particular "split" of the data?
  - In principle, we should do this multiple times since performance may be different for each split

- *k*-Fold Cross-Validation (e.g., k=10)
  - randomly partition full data set of n instances into <u>*k* disjoint subsets</u> (each roughly of size n/k)
  - Choose each fold in turn as the test set; train model on the other folds and evaluate
  - Compute statistics over k test performances, or choose best of the k models

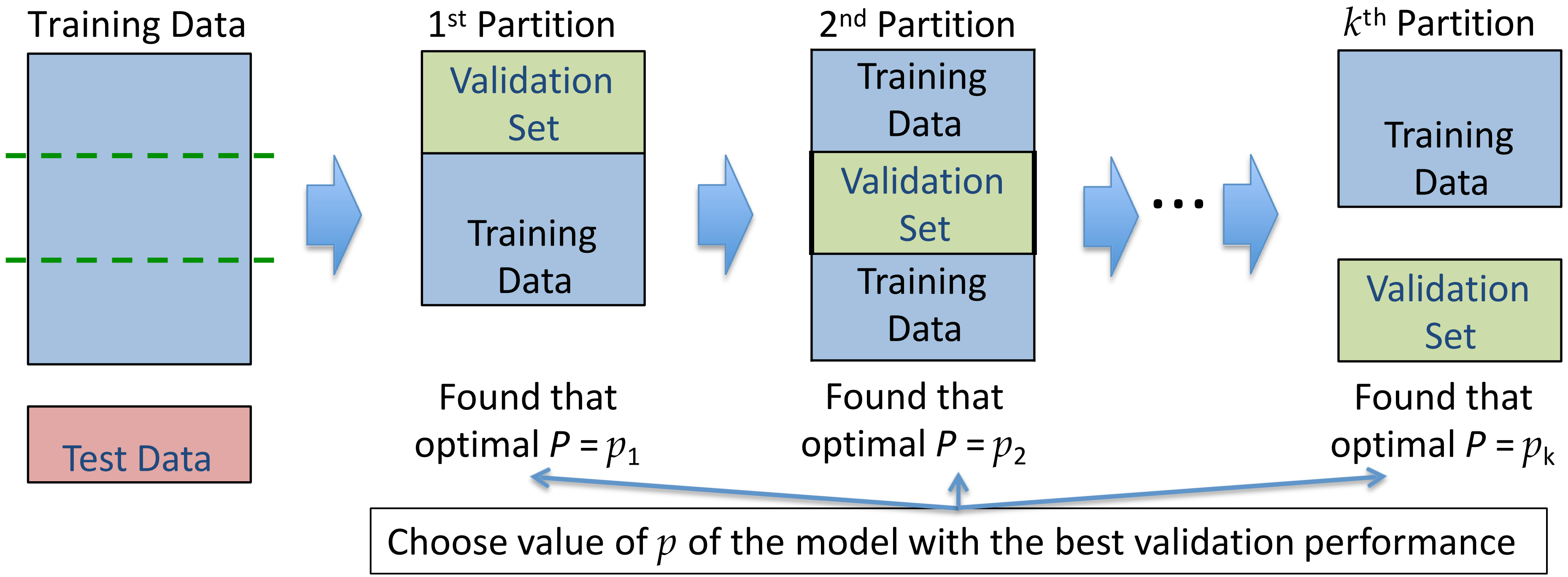
Example 3-Fold CV



Optimizing Model Parameters

Can also use CV to choose value of model parameter P

- Search over space of parameter values $p \in values(P)$
  - Evaluate model with $P = p$ on validation set
- Choose value p' with highest validation performance
- Learn model on full training set with $P = p'$

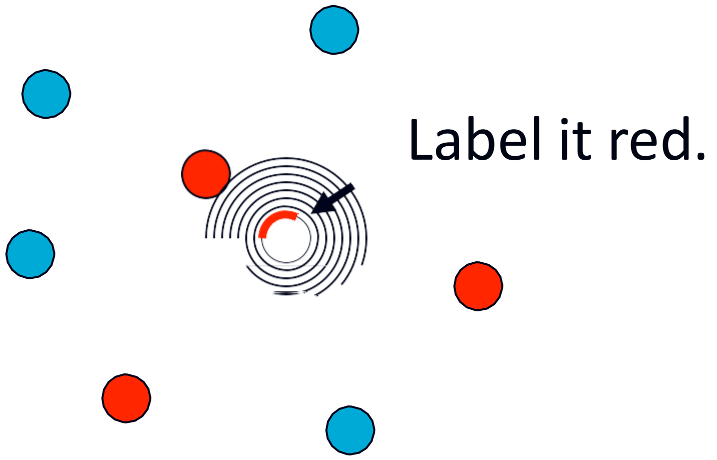


k-Nearest Neighbor

& Instance-based Learning

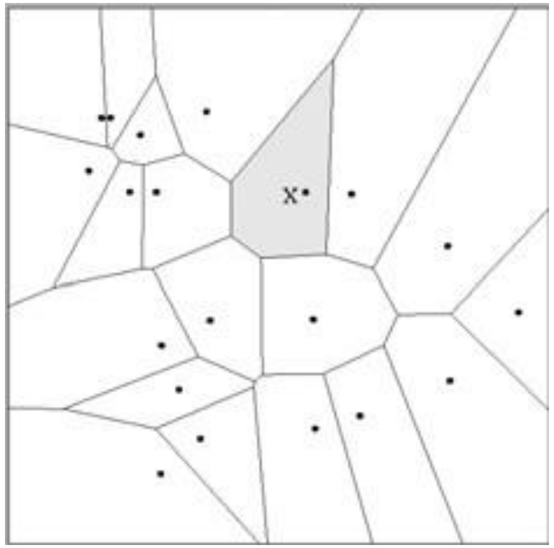
1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point



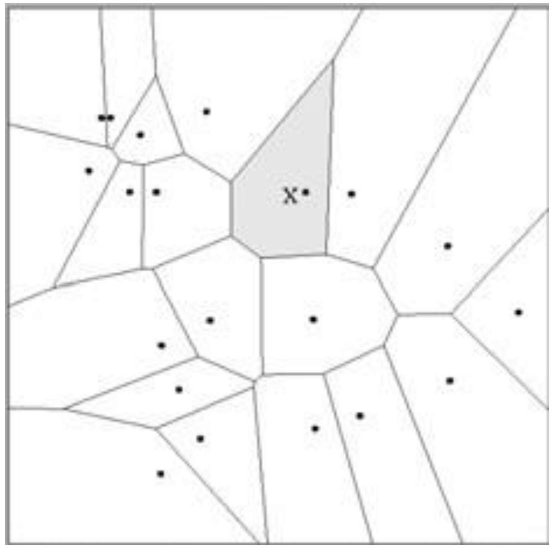
1-Nearest Neighbor

- A type of instance-based learning
  - Also known as “memory-based” learning
- Forms a Voronoi tessellation of the instance space

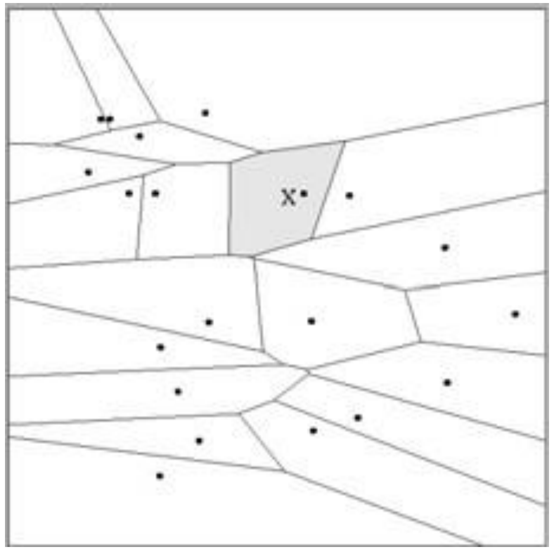


Distance Metrics

- Different metrics can change the decision surface



Dist(**a**,**b**) $=(a_1 - b_1)^2 + (a_2 - b_2)^2$



Dist(**a**,**b**) $=(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Standard Euclidean distance metric:
  - Two-dimensional: Dist(a,b) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$)
  - Multivariate: Dist(a,b) = sqrt($\sum (a_i - b_i)^2$)

Four Aspects of an Instance-Based Learner:

1. A distance metric

2. How many nearby neighbors to look at?

3. A weighting function (optional)

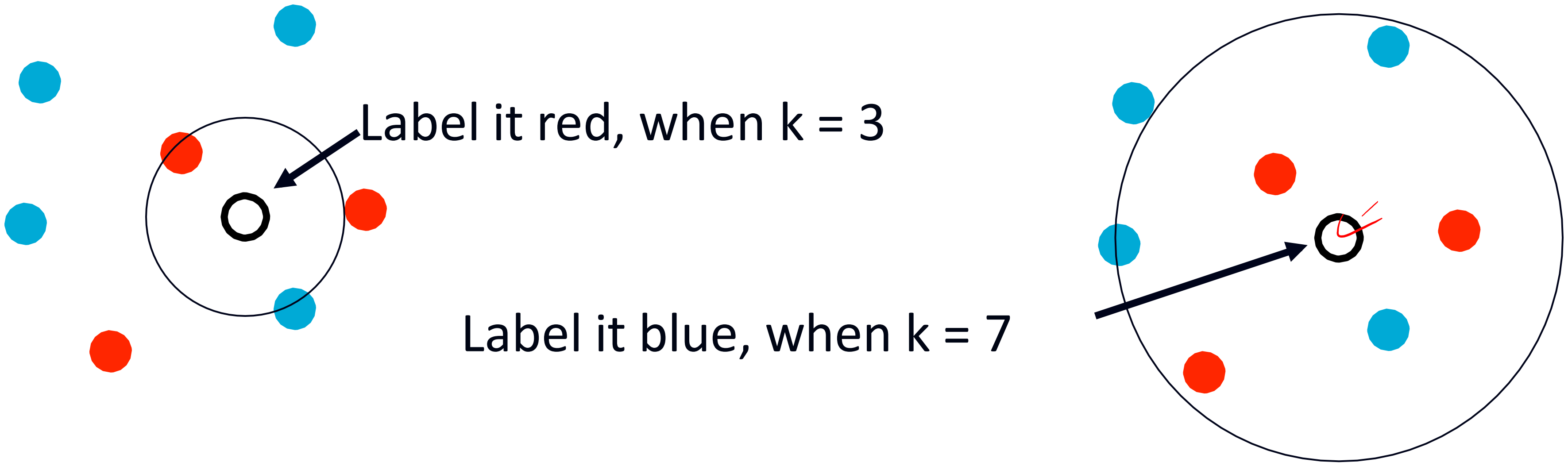
4. How to fit with the local points?

1-NN's Four Aspects as an Instance-Based Learner:

1. A distance metric
   - *Euclidean*
2. How many nearby neighbors to look at?
   - *One*
3. A weighting function (optional)
   - *Unused*
4. How to fit with the local points?
   - *Just predict the same output as the nearest neighbor.*

k – Nearest Neighbor

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its *k* nearest neighbors



k-NN

- Instance-based learning & lazy learning
- Memorize training data, and measure all each pair of instance in the training set and new instance in the test set
- However, computationally expensive
  - Require N comparison for the prediction
  - Refer to https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
- To reduce the search time (i.e., reduce O(n))
  - We may use some data structure
  - e.g., k-d tree (k- dimensional tree)
    - https://en.wikipedia.org/wiki/K-d_tree
- k-NN regression
  - k nearest neighbors' average target class value
  - http://www.saedsayad.com/k_nearest_neighbors_reg.htm
  - http://scikit-learn.org/stable/modules/neighbors.html

Quiz1

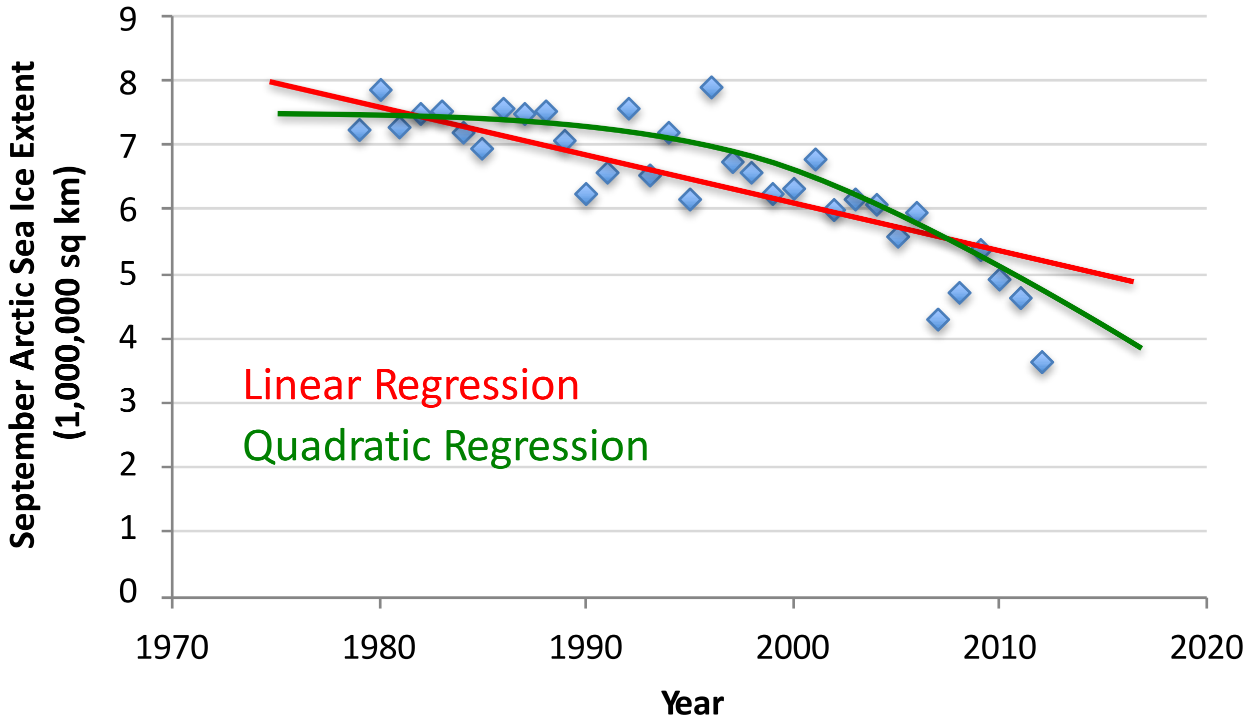
- Quiz1 will be taken on June 4 – it will be available only during the day

- The coverage will be from content of the first lecture to K-nearest neighbors

Linear Regression

Regression

- Given:
  - Data $X = \{x^{(1)}, \ldots, x^{(n)}\}$ where $x^{(i)} \in \mathbb{R}^d$
  - Corresponding labels $\mathrm{y} = \{y^{(1)}, \ldots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



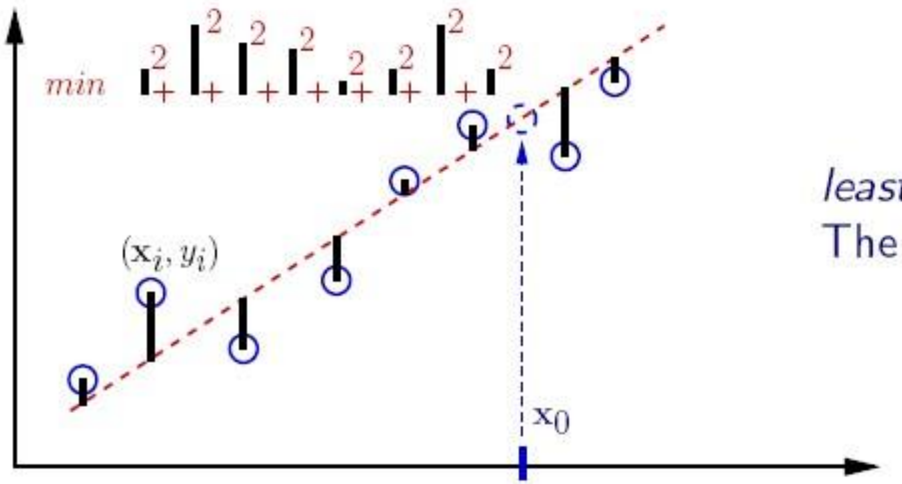
Linear Regression

- Hypothesis:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors



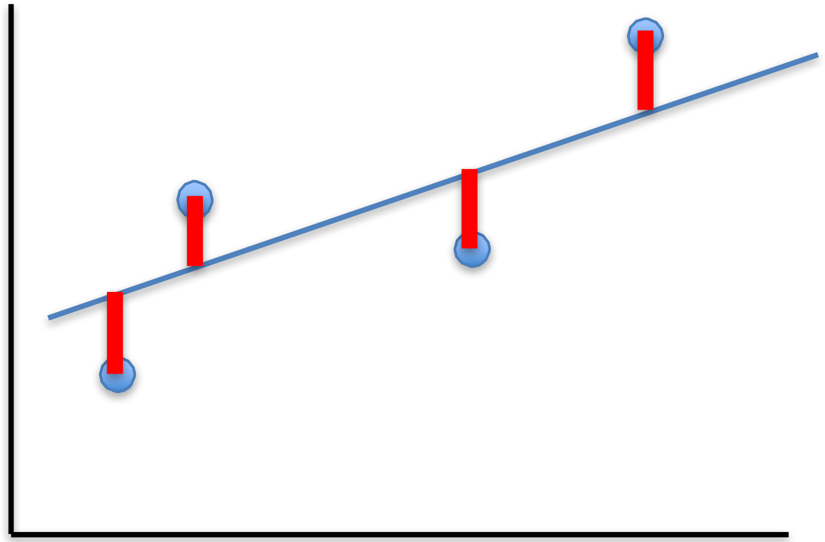
least squares (LSQ)
The fitted line is used as a predictor

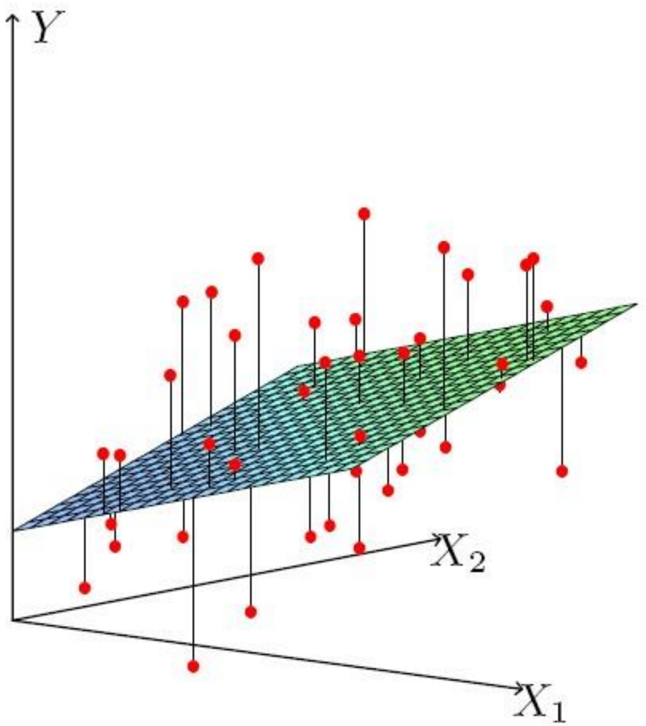
Least Squares Linear Regression

- Cost Function

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

- Fit by solving $\min_\theta J(\theta)$





Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$

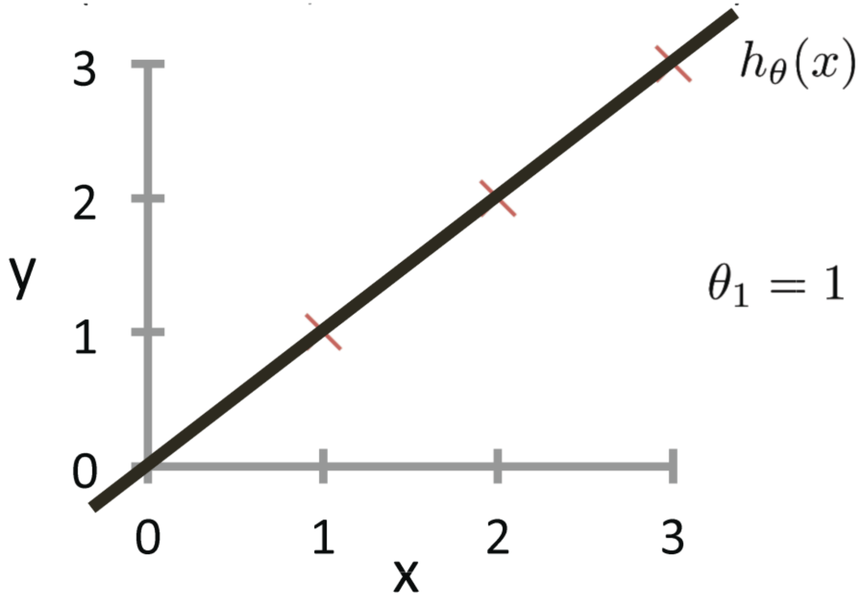
Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$ ➔ $\theta_0 = 0$

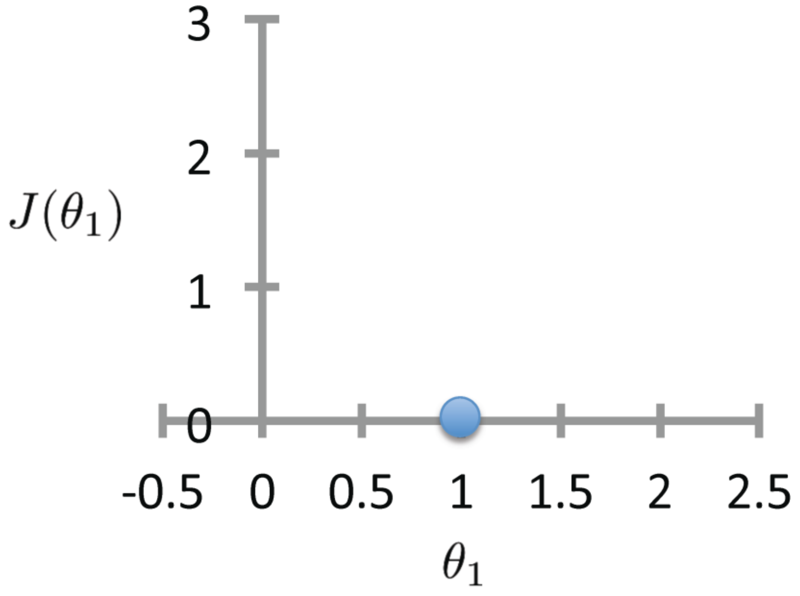
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta)$

(function of the parameter θ_1)



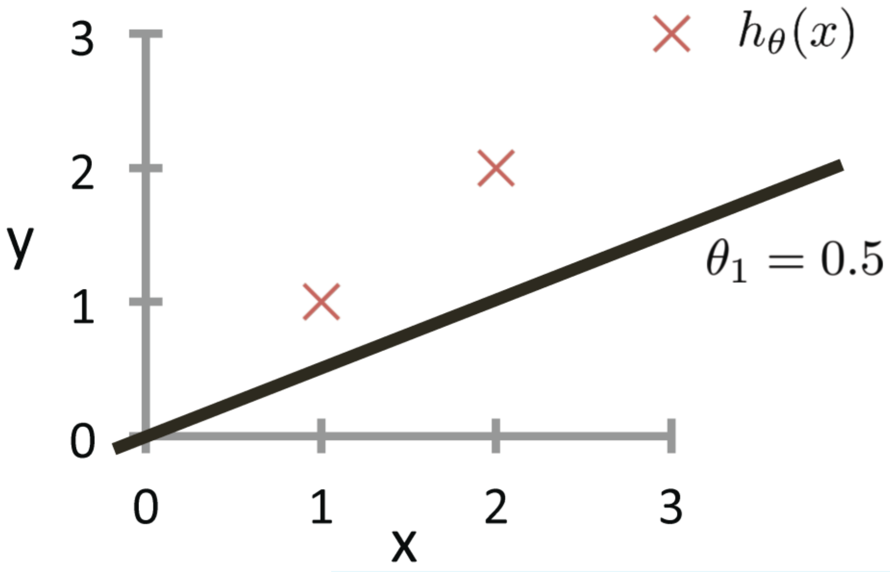
Intuition Behind Cost Function

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$ ➔ $\theta_0 = 0$

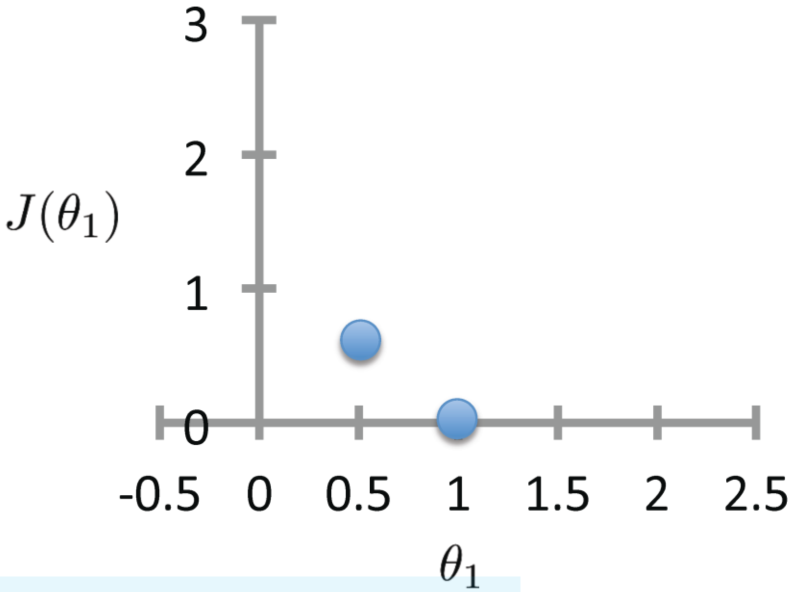
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta)$

(function of the parameter θ_1)



$$J([0,0.5]) = \frac{1}{2 \times 3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2 \approx 0.58$$

Intuition Behind Cost Function

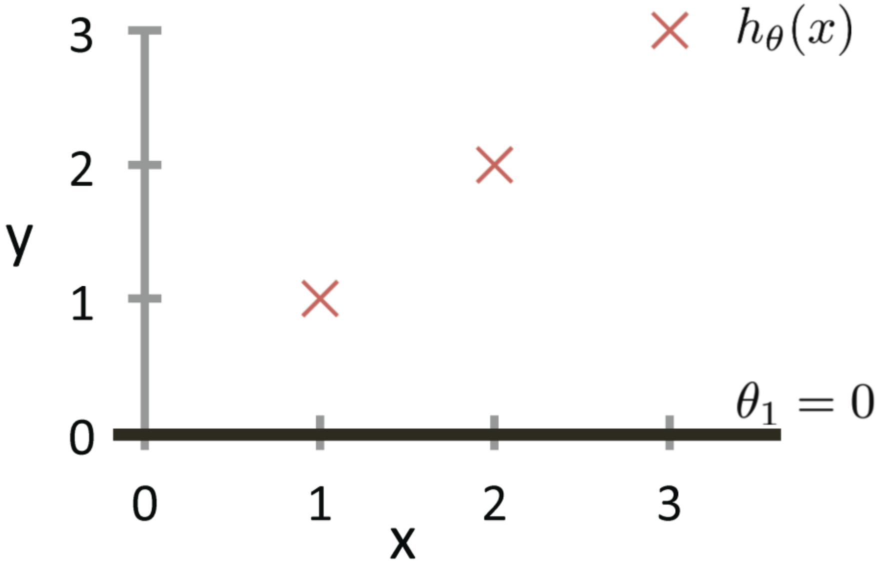
$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$

$\rightarrow \theta_0 = 0$

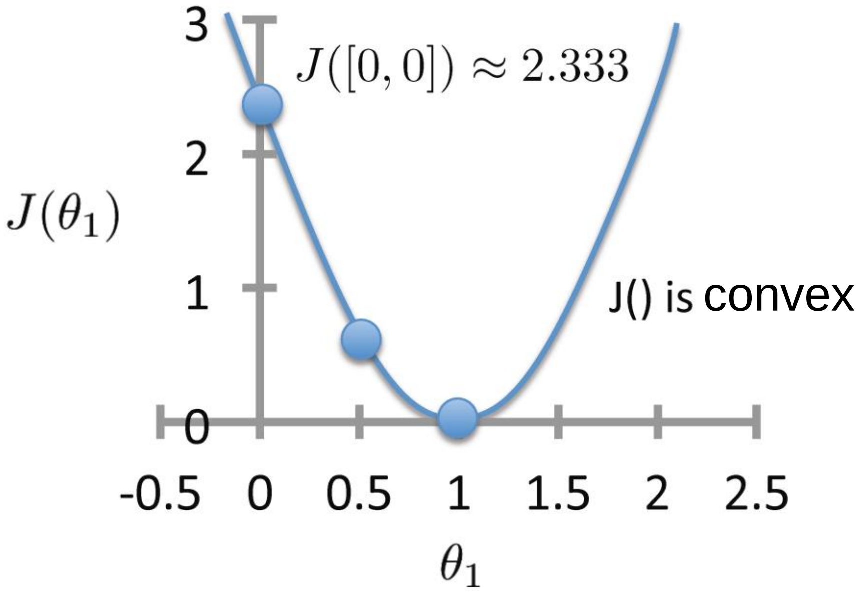
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta)$

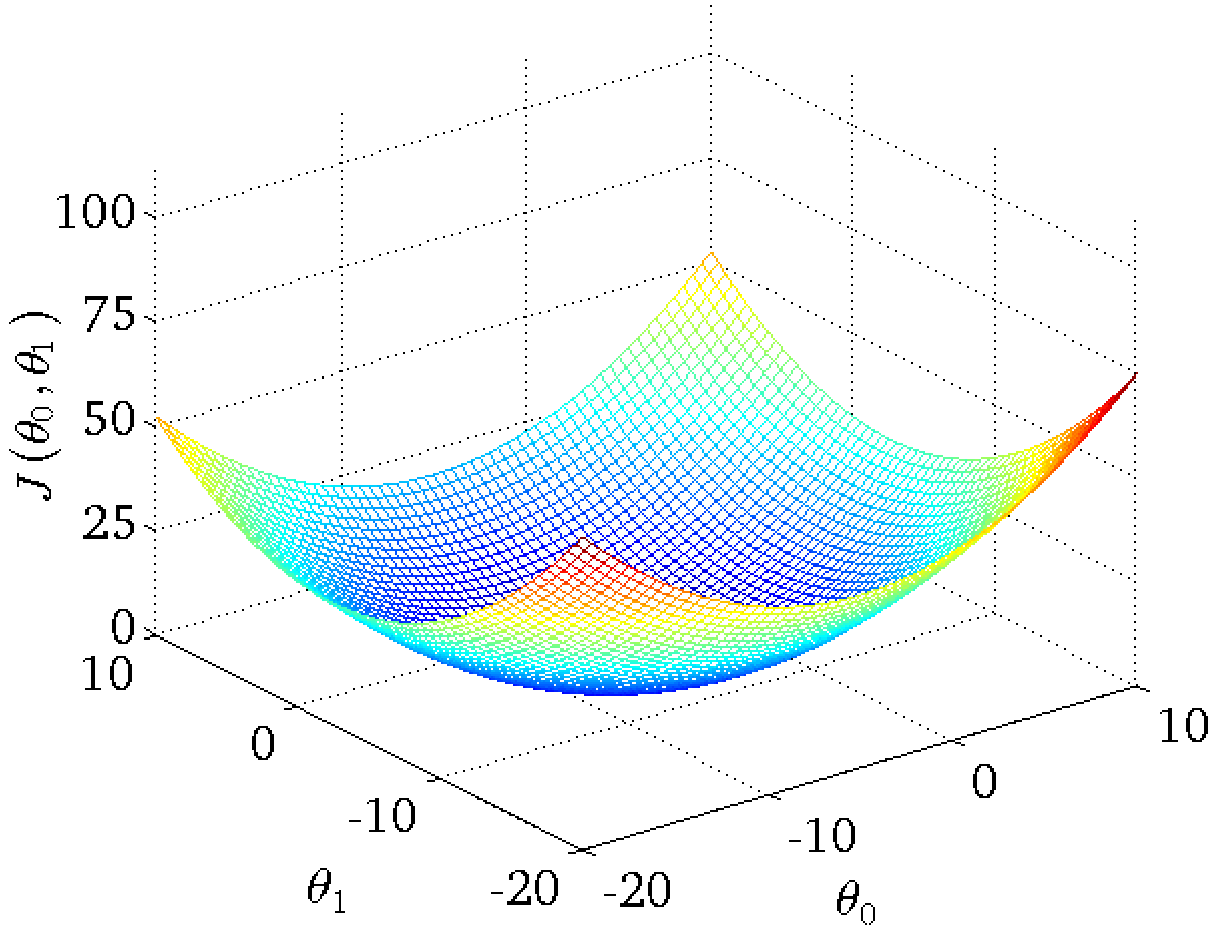
(function of the parameter θ_1)



http://mathworld.wolfram.com/ConvexFunction.html

https://www.desmos.com/calculator/kreo2ssqj8

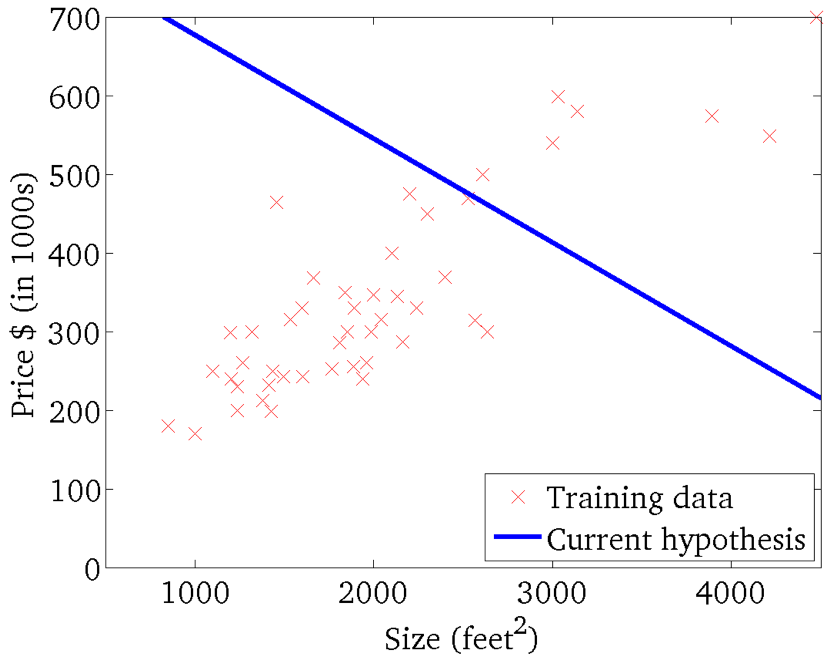
Intuition Behind Cost Function (3-D surface plot)



Intuition Behind Cost Function

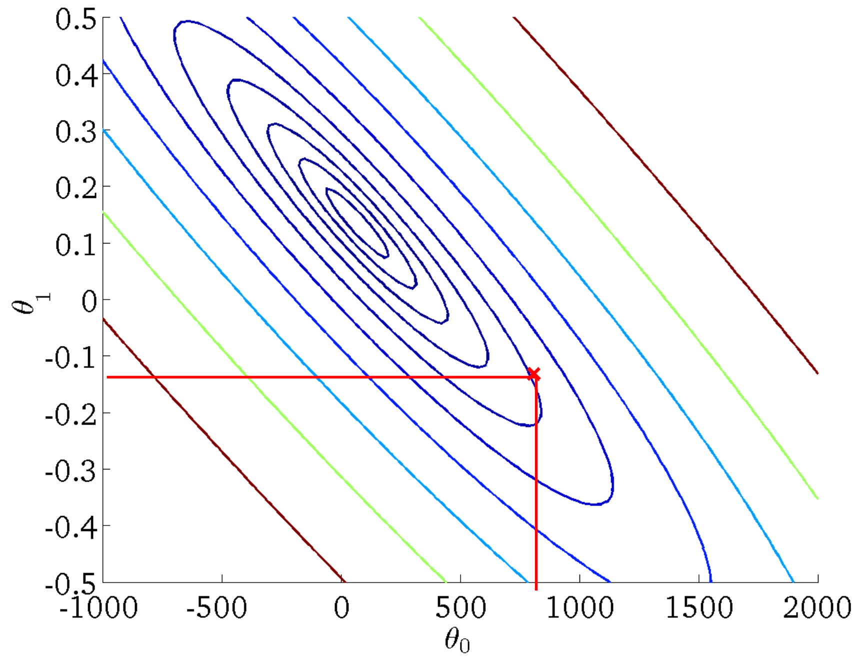
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameter θ_0, θ_1)

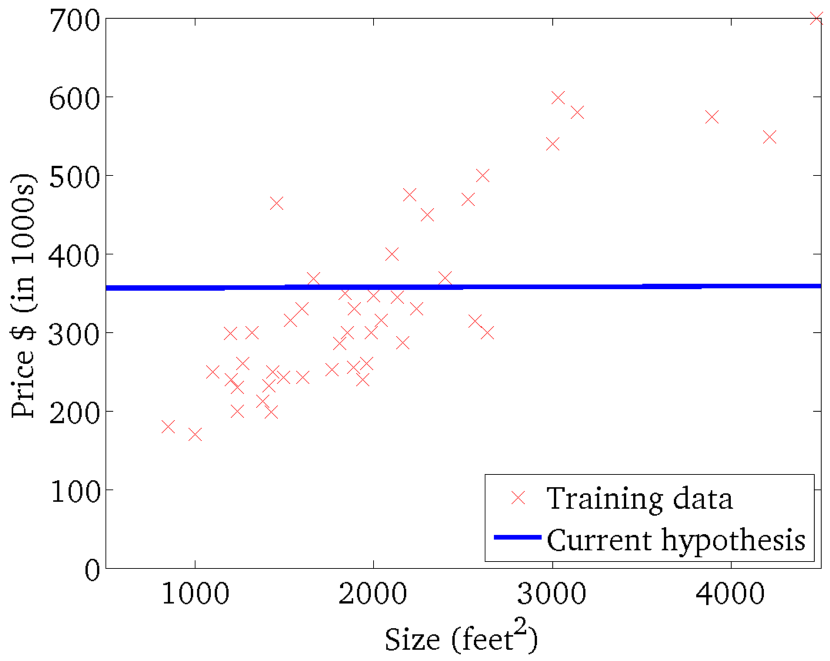


Contour Figure

Intuition Behind Cost Function

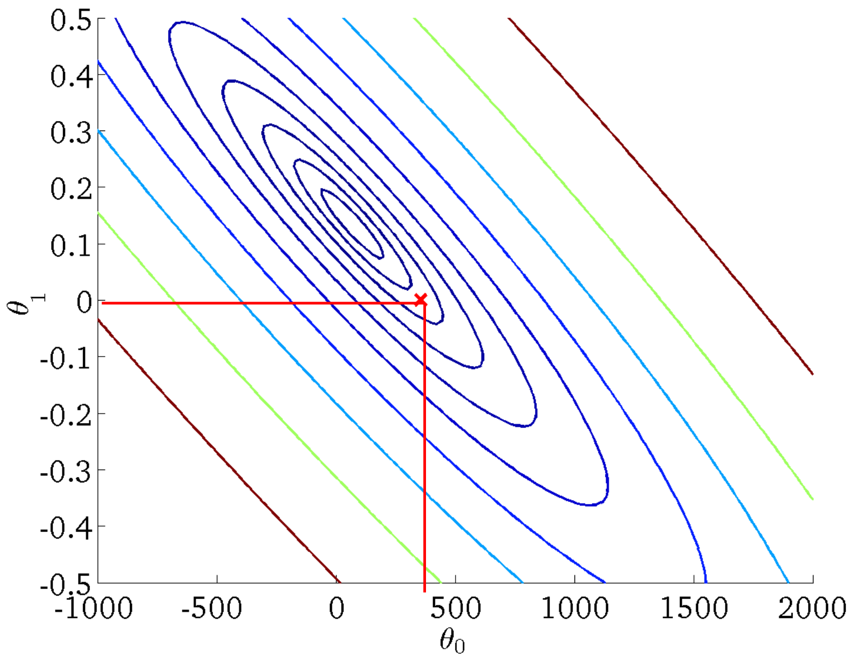
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

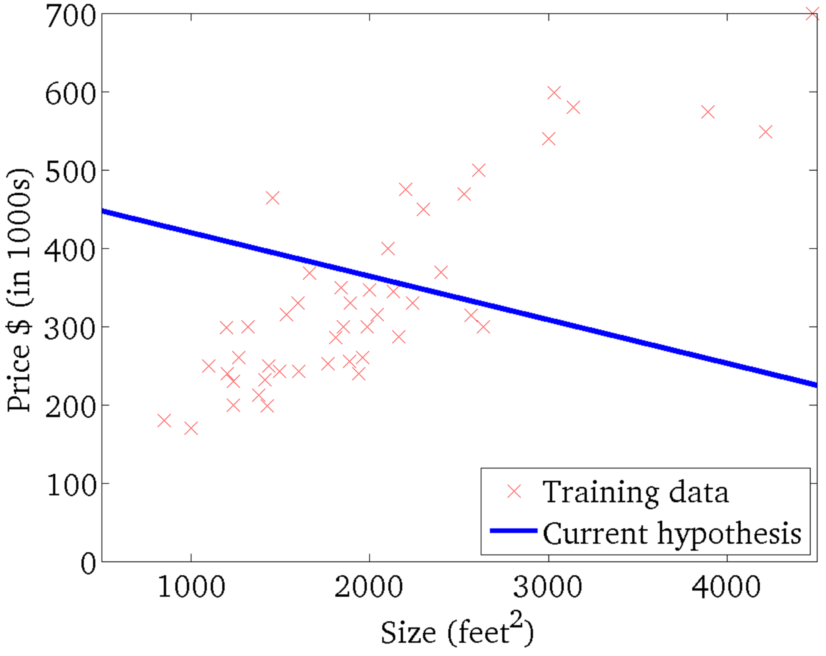
(function of the parameter θ_0, θ_1)



Intuition Behind Cost Function

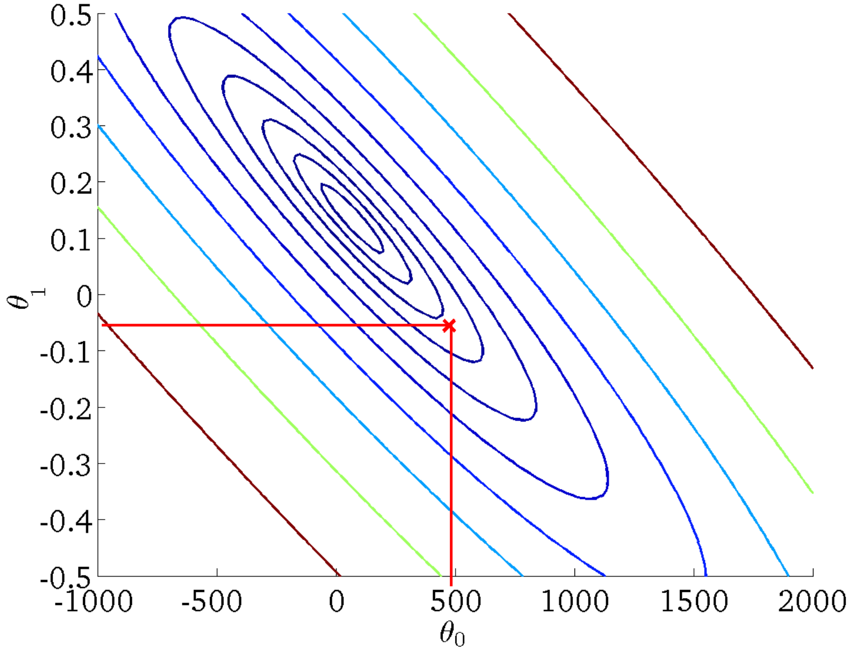
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



$$J(\theta_0, \theta_1)$$

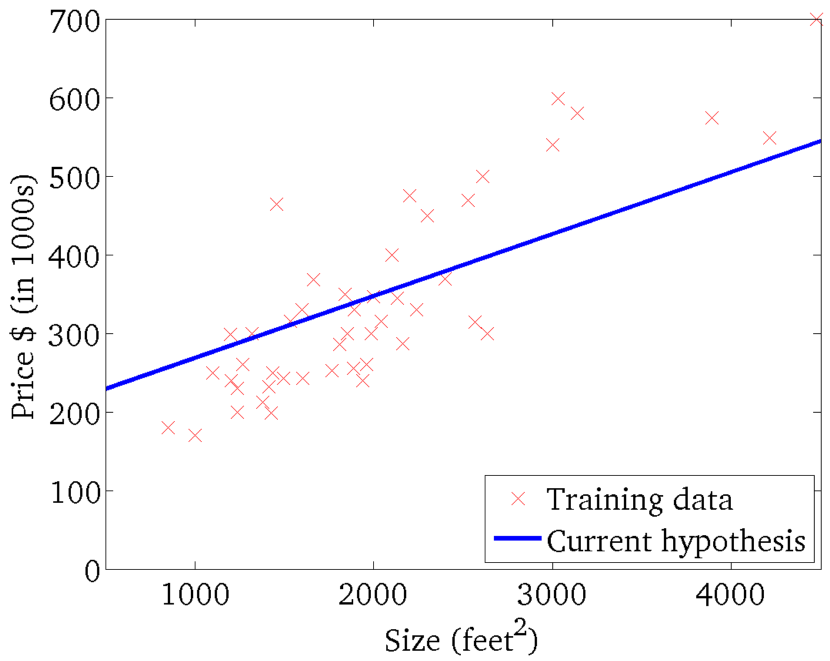
(function of the parameter θ_0, θ_1)



Intuition Behind Cost Function

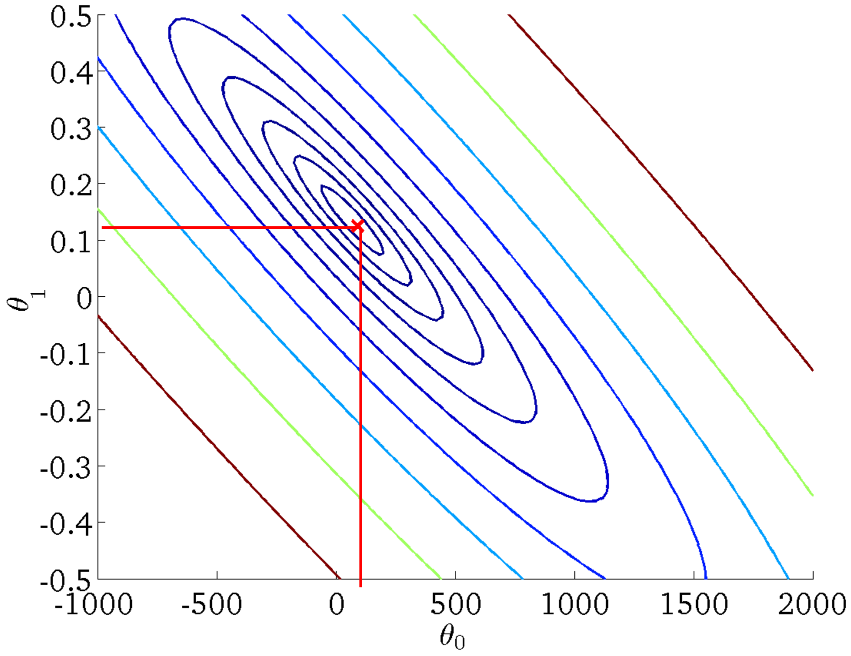
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



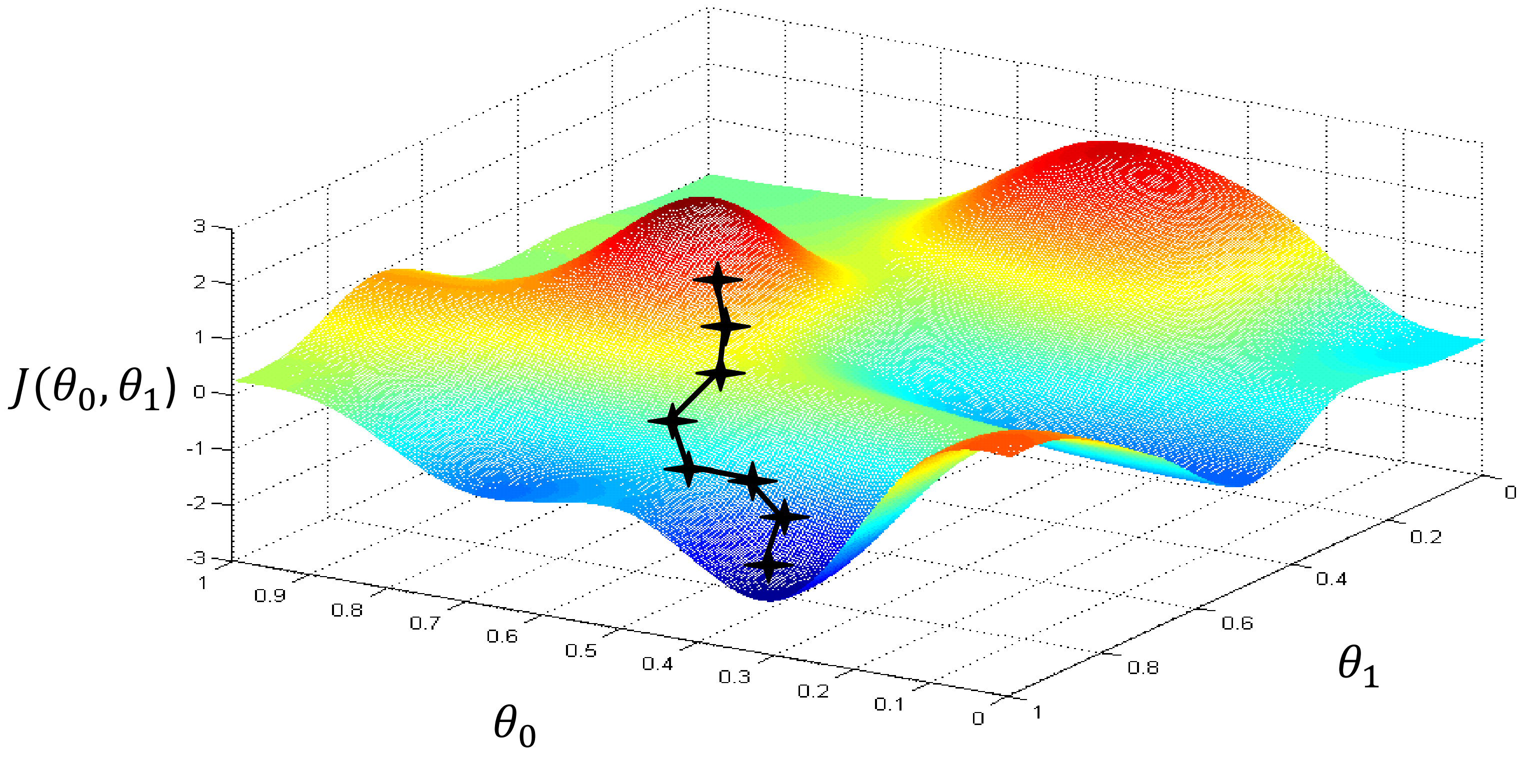
$J(\theta_0, \theta_1)$

(function of the parameter θ_0, θ_1)



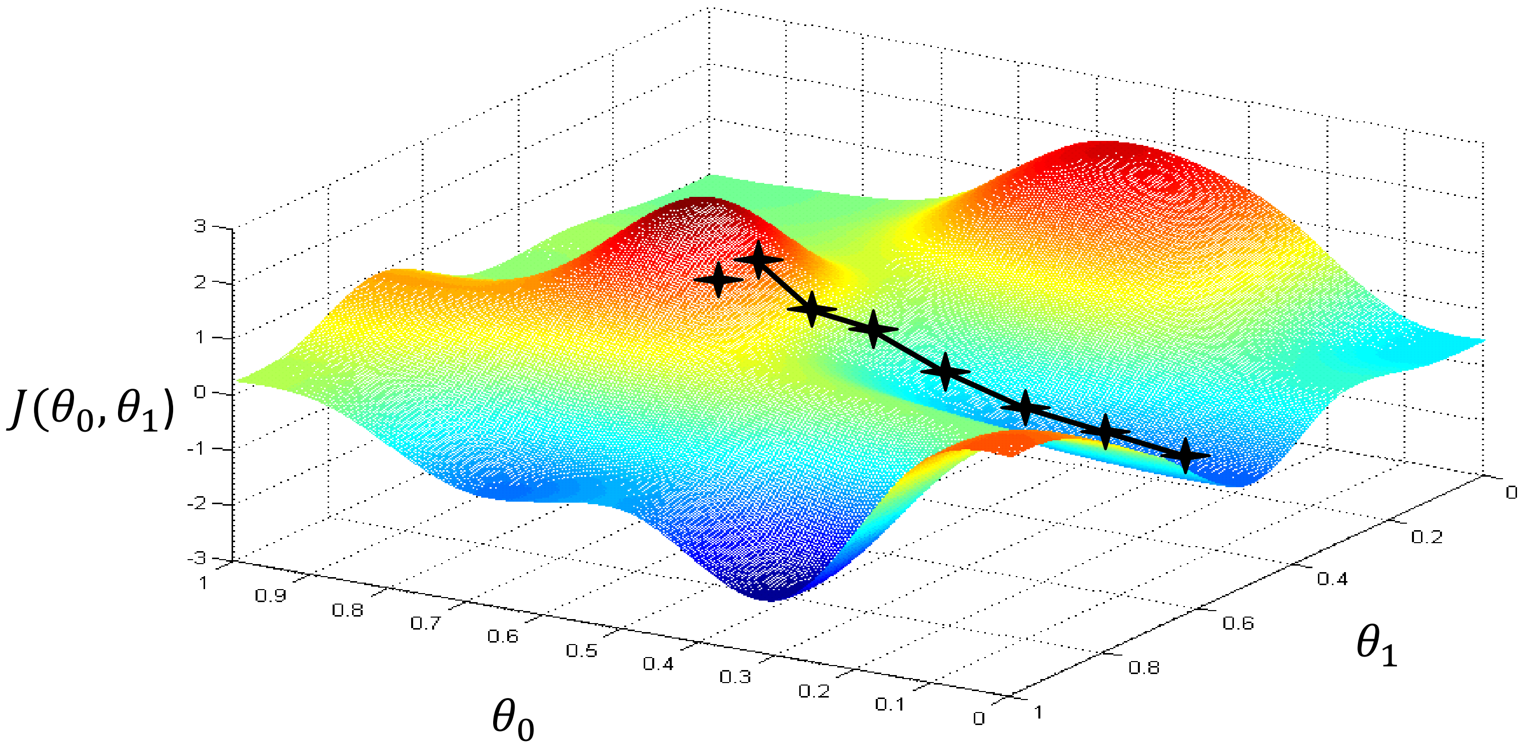
Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



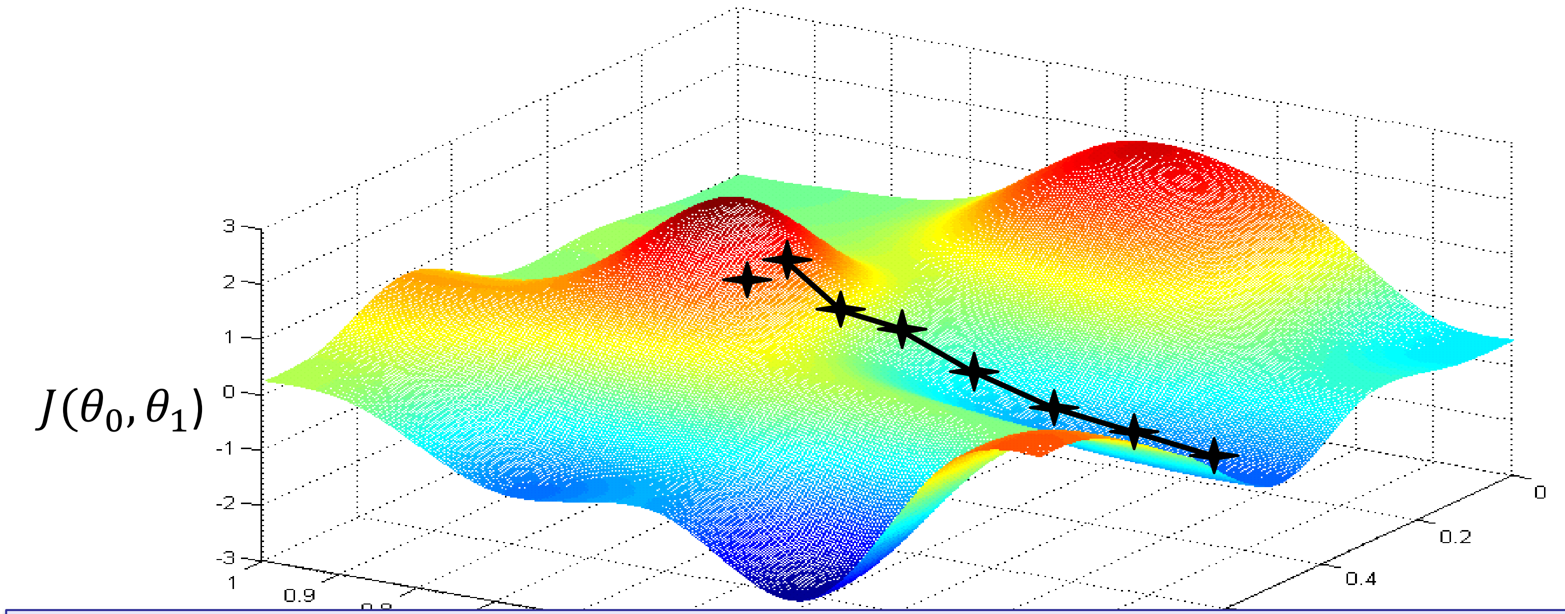
Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex, we don't need to worry about local minima