

# Machine Learning (机器学习)-Regression

---

**Prof.Shuyuan Yang, Zhixi Feng**

**E-mail: [syyang@xidian.edu.cn](mailto:syyang@xidian.edu.cn)**

**[zxfeng@xidian.edu.cn](mailto:zxfeng@xidian.edu.cn)**



# Contents

- What is Machine Learning
- Applications of Machine Learning
- Components of Machine Learning
- Machine Learning and Other Fields

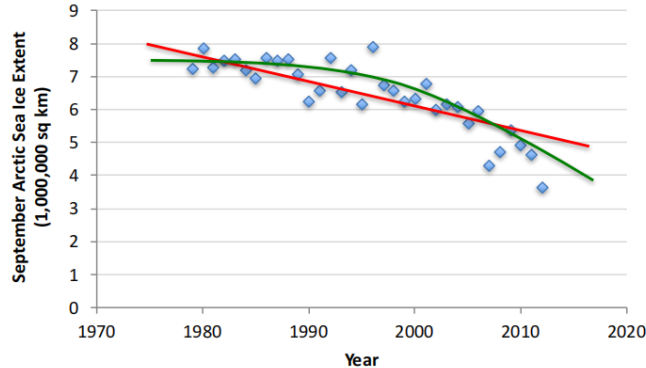
# Contents

- What is Regression
- Linear Regression
- Components of Machine Learning
- Machine Learning and Other Fields

# Supervised Learning:

## Regression

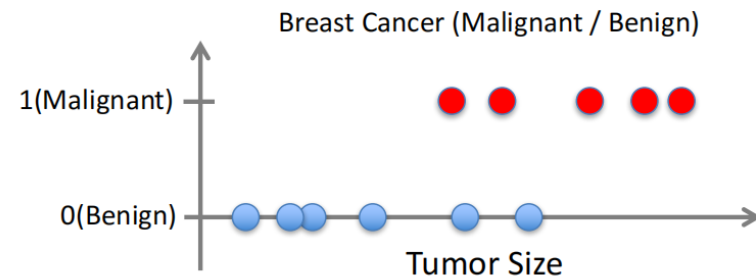
- Given  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function  $f(x)$  to predict  $y$  given  $x$ 
  - $y$  is real-valued == regression



Continuous variables

## Classification

- Given  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function  $f(x)$  to predict  $y$  given  $x$ 
  - $y$  is categorical == classification



Discrete variable

# Linear Regression

## Housing Dataset

Sales of individual residential property in Ames, Iowa from 2006 to 2010

- **Instances:** 1,022
- **Features:** 79 total (19 ordinal, 25 nominal, 35 numeric)
- **Target Variable:** Sales price

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

# Mixed Feature Types

- Most data sets contain **more than one type of feature**
- Types and possible values may not be obvious
  - Consulting a data dictionary is critical!

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	11400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

**Categorical features**

**Ordinal features**

**Numeric features**

**Looks numeric, but is actually categorical**

**Encoding Features**

**Understanding the Data ! ! !**

# Linear Regression

- As with all datasets, the first thing we do is examine it...

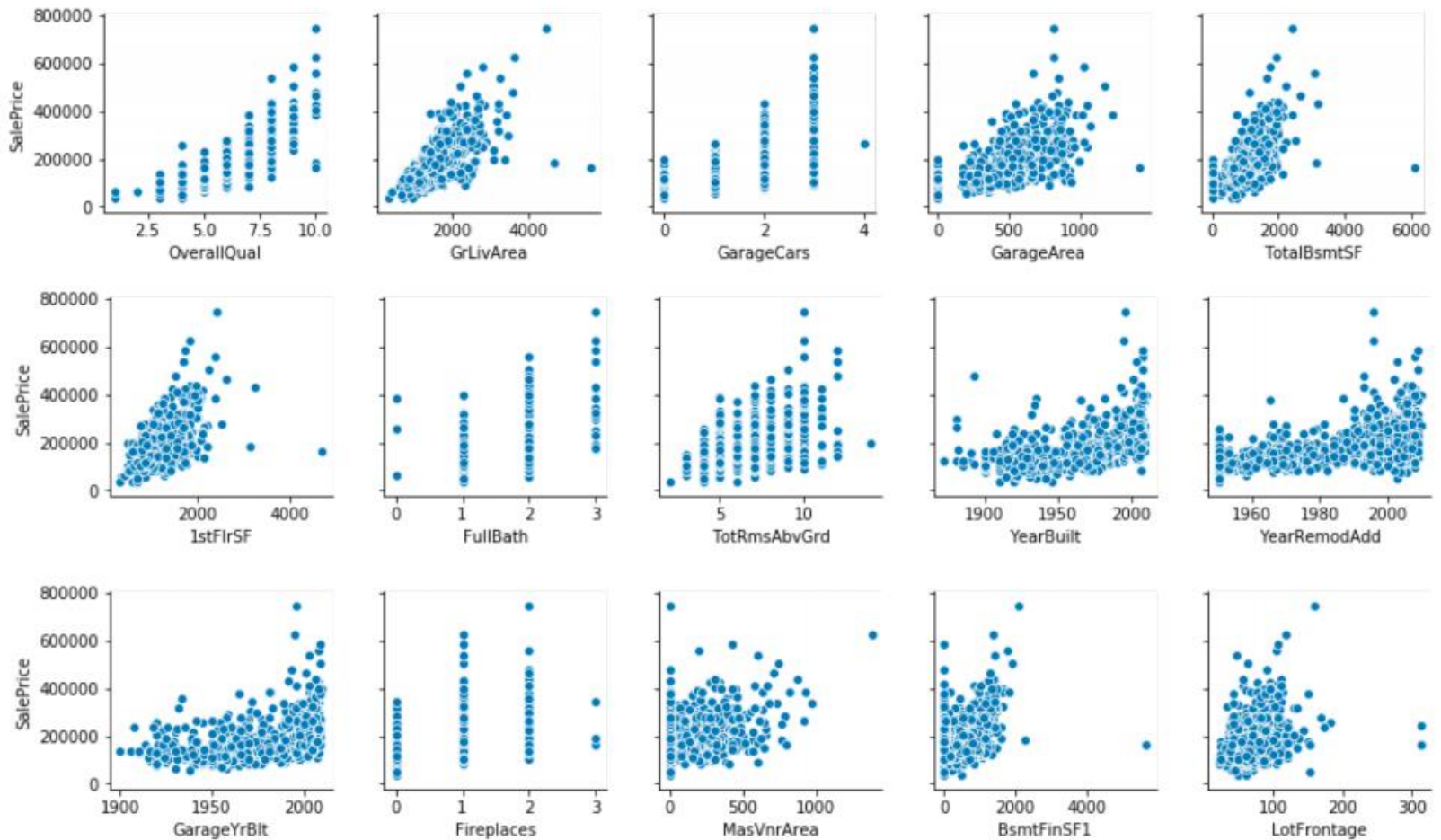
Note the  
missing values

No missing target  
values

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea
count	1022.000000	1022.000000	832.000000	1022.000000	1022.000000	1022.000000	1022.000000	1022.000000	1019.000000
mean	732.338552	57.059687	70.375000	10745.437378	6.128180	5.564579	1970.995108	1984.757339	105.261040
std	425.860402	42.669715	25.533607	11329.753423	1.371391	1.110557	30.748816	20.747109	172.707705
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000
25%	367.500000	20.000000	59.000000	7564.250000	5.000000	5.000000	1953.000000	1966.000000	0.000000
50%	735.500000	50.000000	70.000000	9600.000000	6.000000	5.000000	1972.000000	1994.000000	0.000000
75%	1100.500000	70.000000	80.000000	11692.500000	7.000000	6.000000	2001.000000	2004.000000	170.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1378.000000

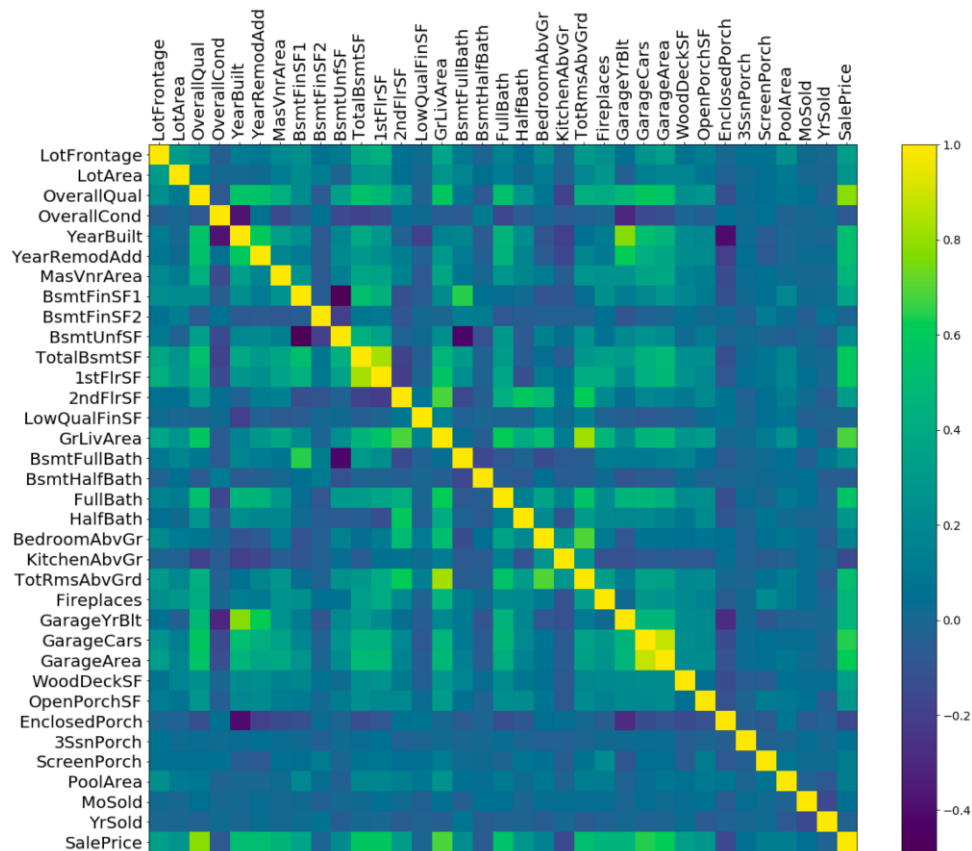
Potential outliers

# Linear Regression





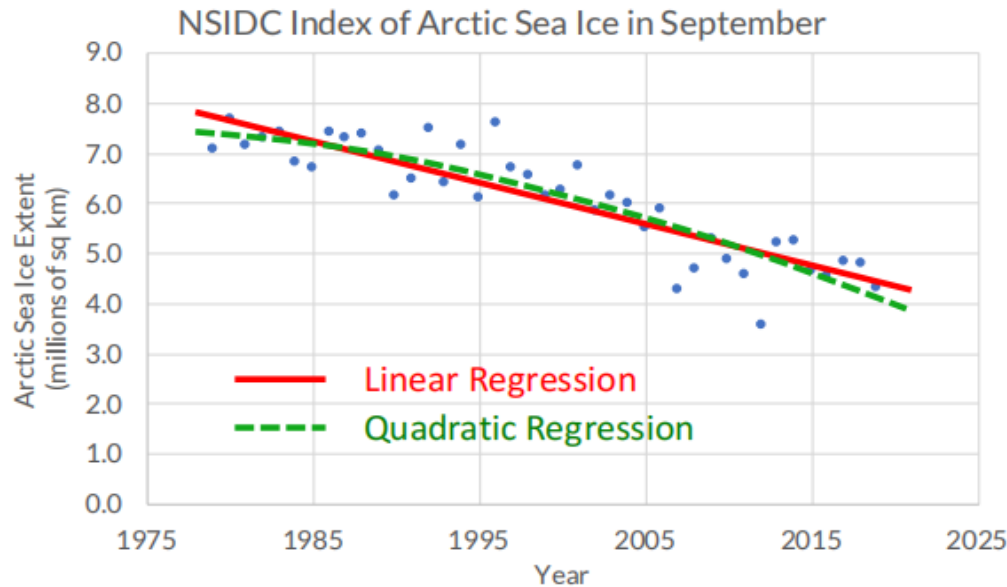
## Feature Correlation Matrix



# Regression

Given

- Data  $X = \{x_1, \dots, x_n\}$  where  $x_i \in \mathbb{R}^d$
- Corresponding labels  $y = \{y_1, \dots, y_n\}$  where  $y_i \in \mathbb{R}$



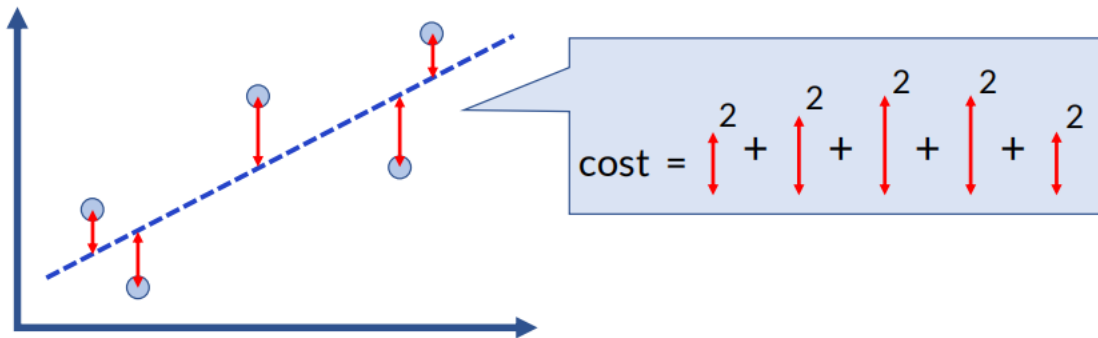
# Ordinary Least Squares

- Least Squares Linear Regression
- Hypothesis:

$$y = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id} = \sum_{j=0}^d \theta_j x_{ij}$$

Assume  $x_{i0} = 1$

- Fit model by minimizing **sum of squared errors**



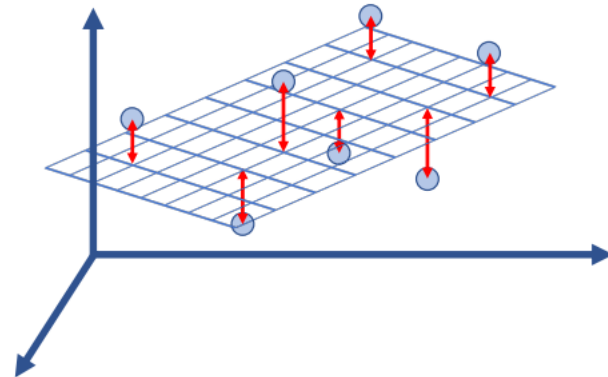
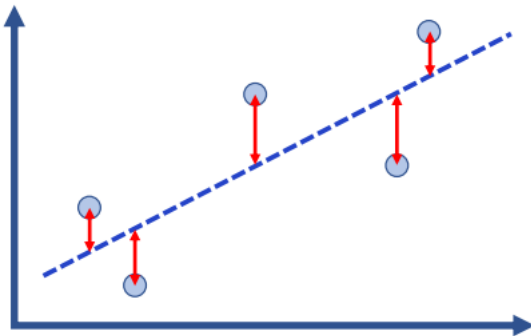
# Regression

- Cost Function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i)^2$$

- Fit by solving

$$\min_{\theta} \mathcal{L}(\theta)$$



# Intuition Behind the Cost Function

- Cost Function

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2$$

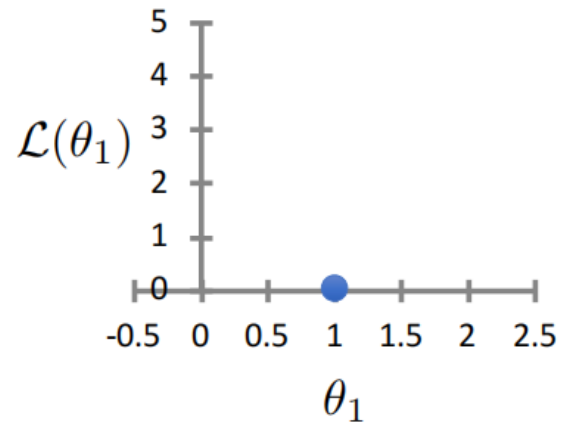
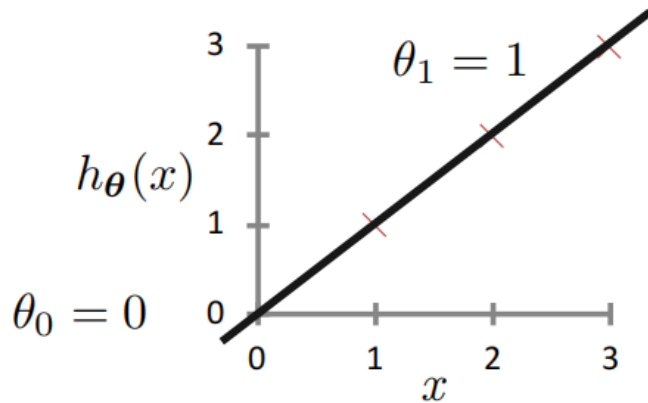
- For insight on  $\mathcal{L}(\boldsymbol{\theta})$ , let's assume  $x \in \mathbb{R}$  so  $\boldsymbol{\theta} = [\theta_0, \theta_1]$

# Intuition Behind the Cost Function

- Cost Function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2$$

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

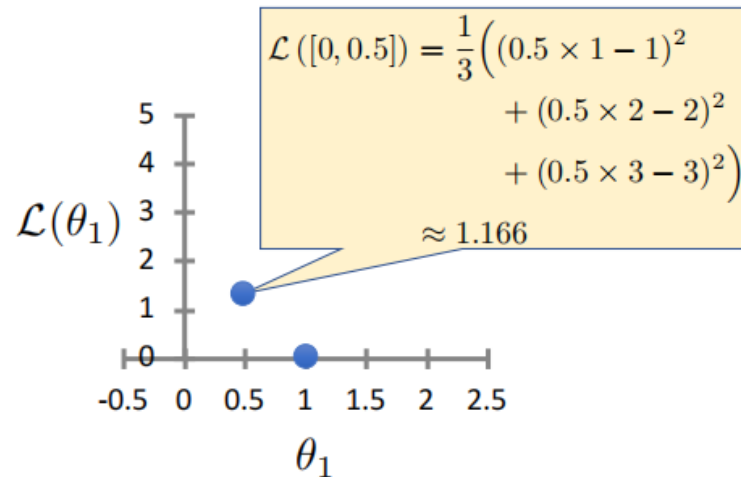
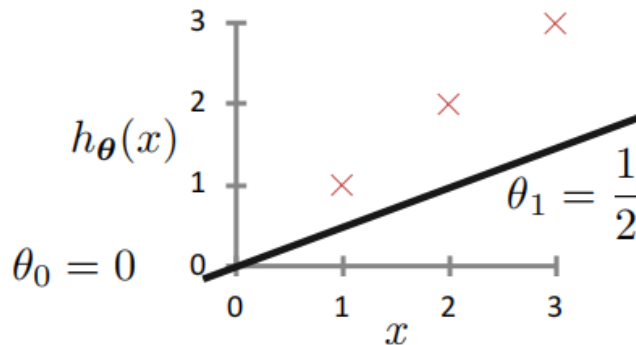


# Intuition Behind the Cost Function

- Cost Function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2$$

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

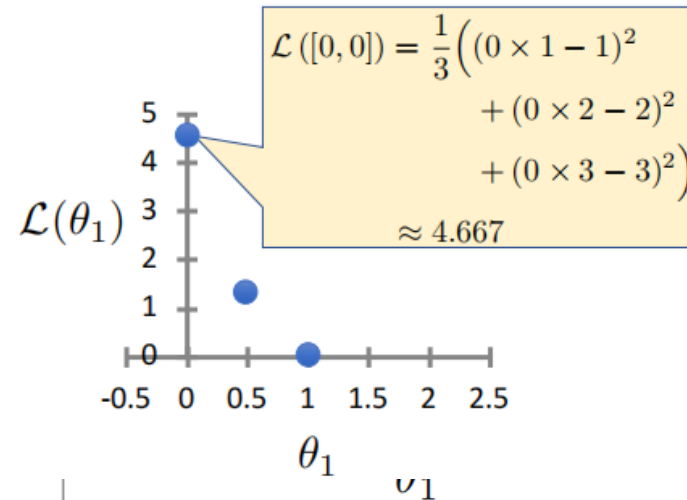
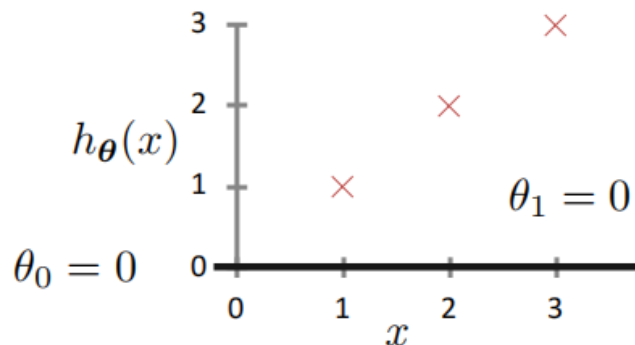


# Intuition Behind the Cost Function

- Cost Function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2$$

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



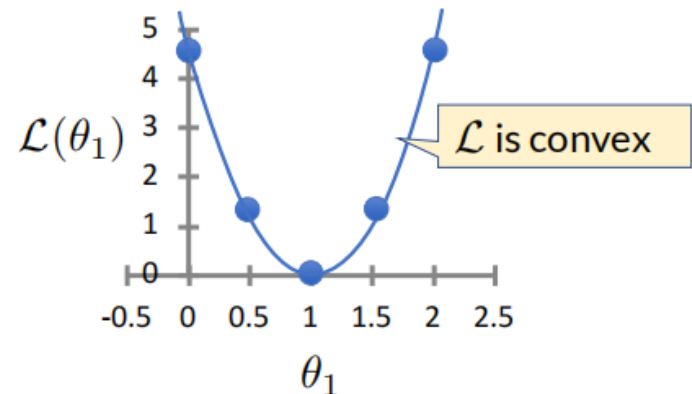
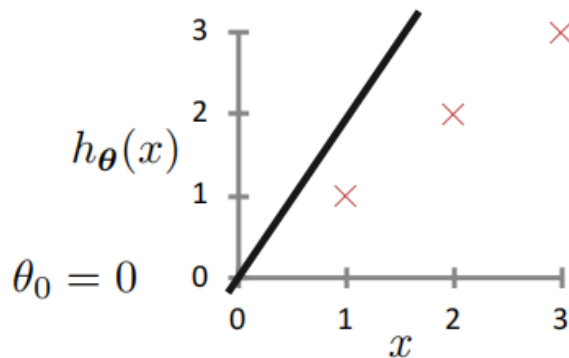


# Intuition Behind the Cost Function

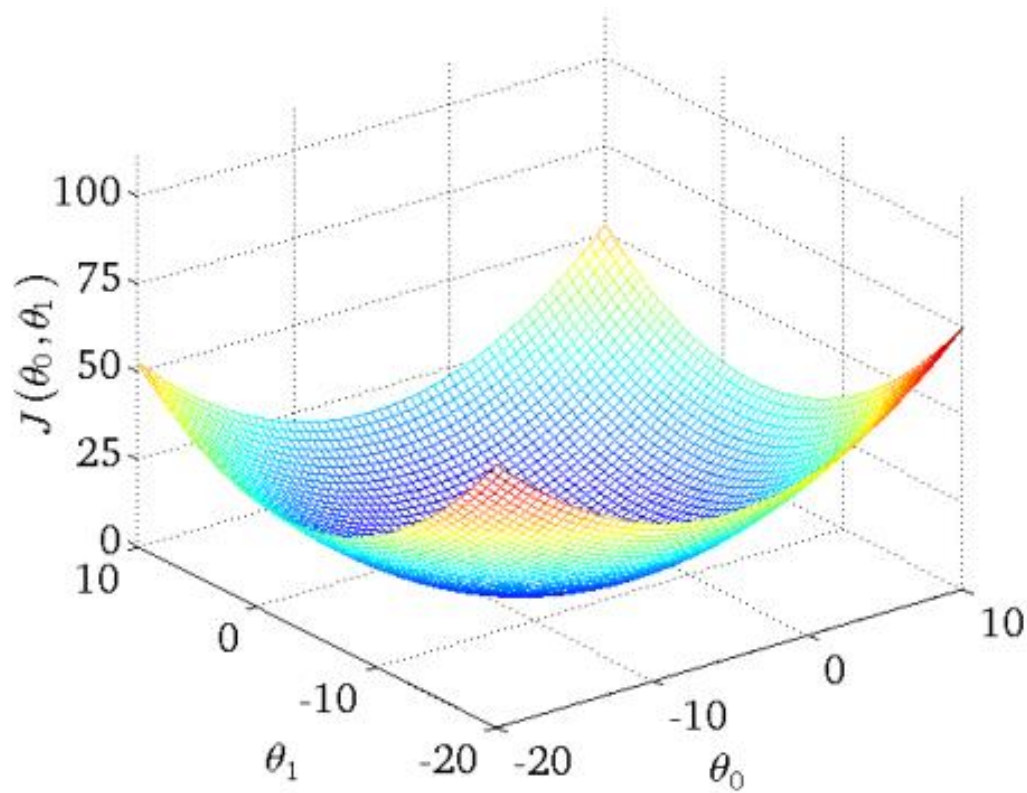
- Cost Function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2$$

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



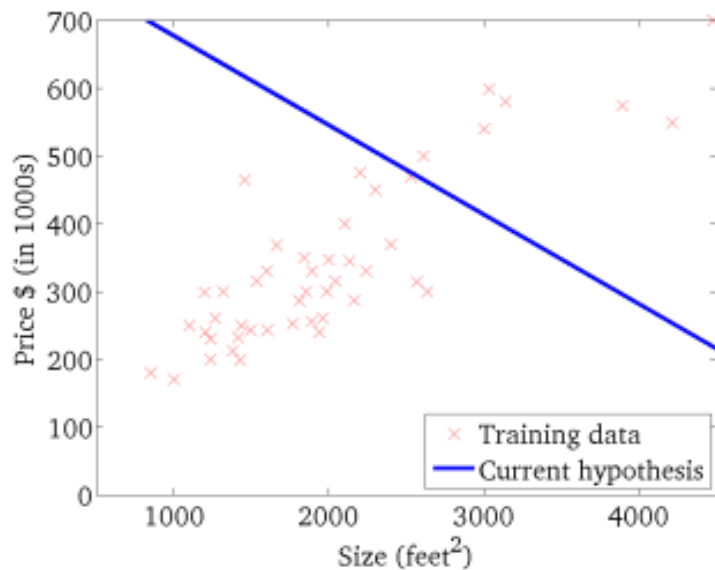
# Intuition Behind the Cost Function



# Intuition Behind the Cost Function

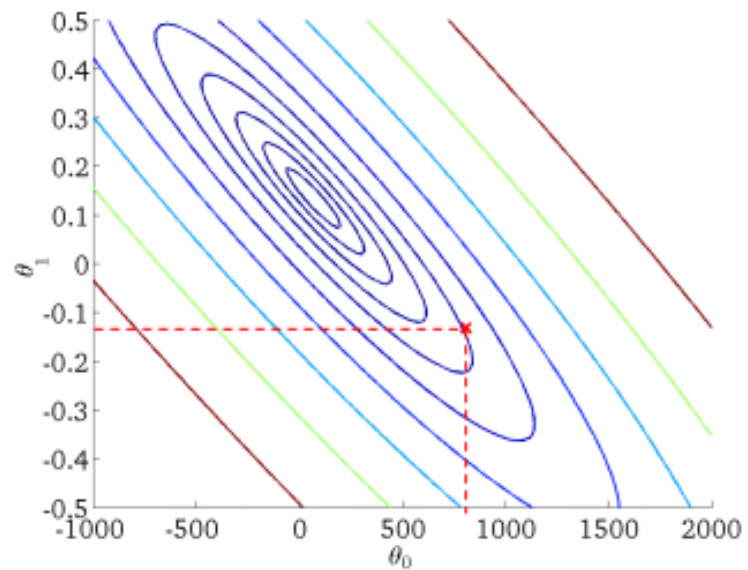
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$\mathcal{L}(\theta_0, \theta_1)$$

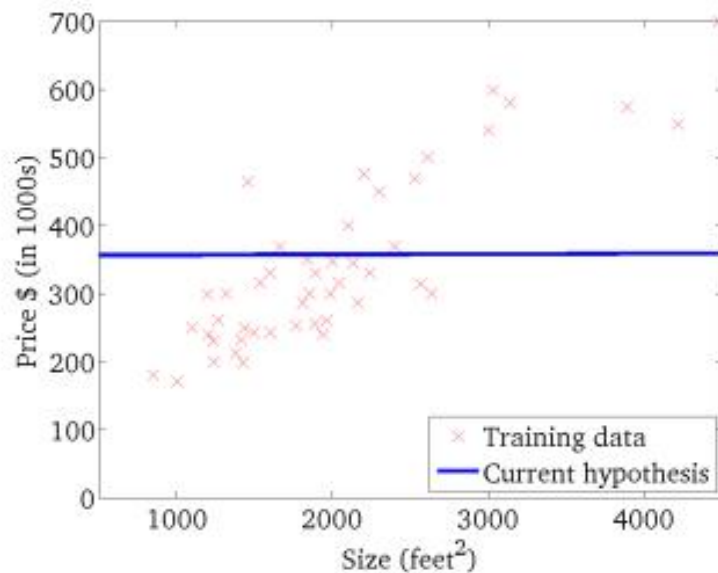
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind the Cost Function

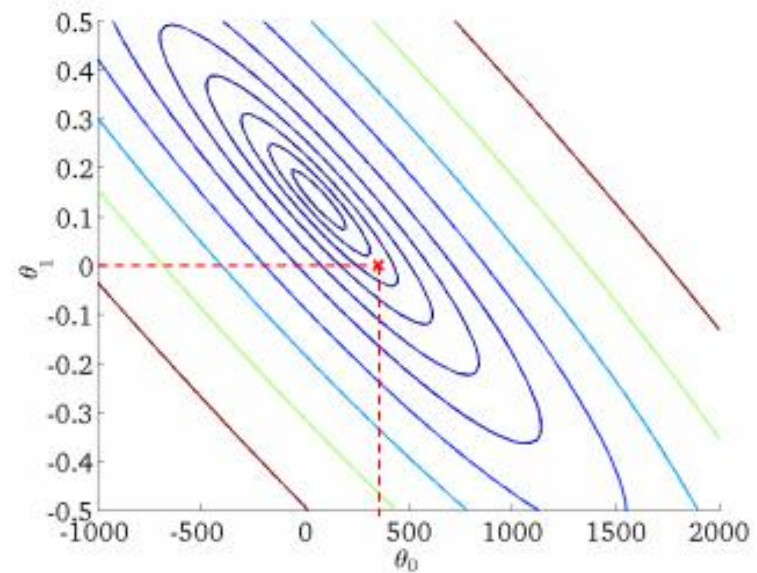
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$\mathcal{L}(\theta_0, \theta_1)$$

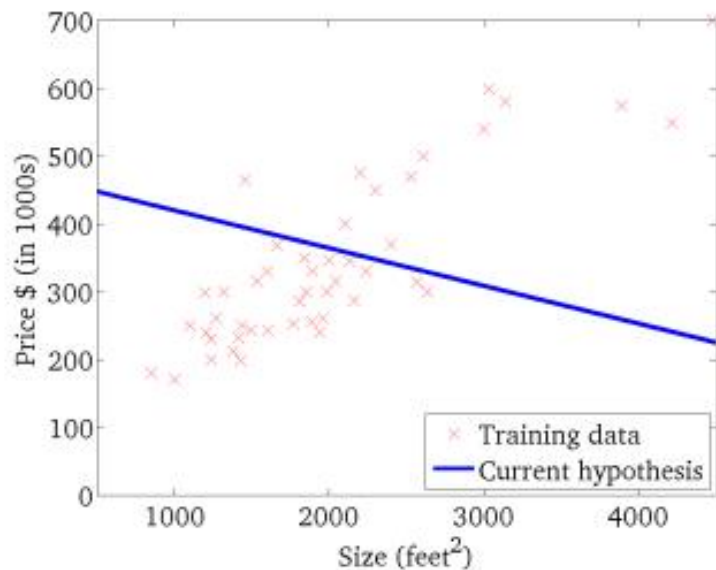
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind the Cost Function

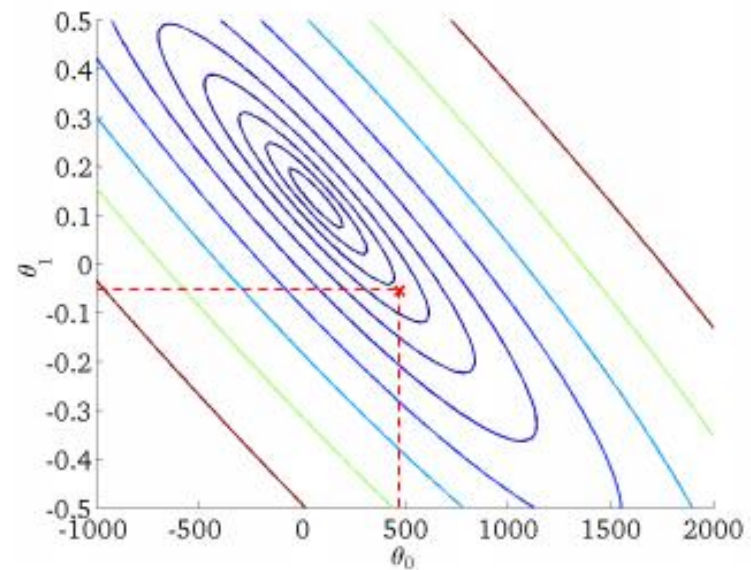
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$\mathcal{L}(\theta_0, \theta_1)$$

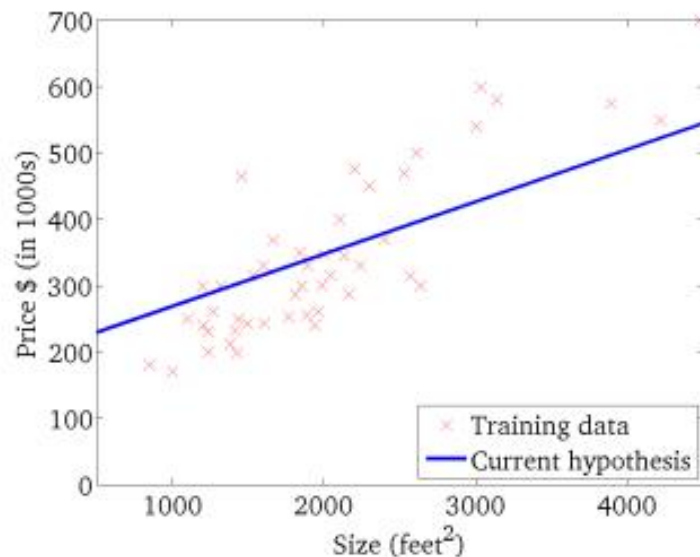
(function of the parameters  $\theta_0, \theta_1$ )



# Intuition Behind the Cost Function

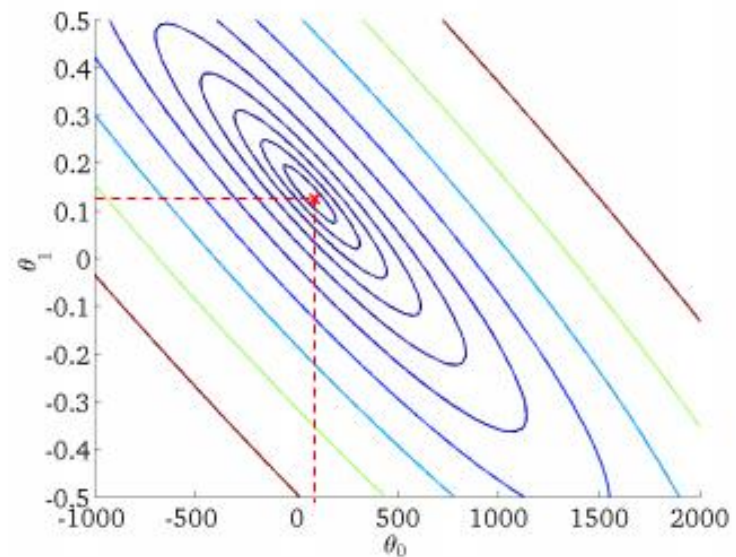
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$\mathcal{L}(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



# Closed Form Solution

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model

$$h(\mathbf{x}_i) = \sum_{j=0}^d \theta_j x_{ij}$$

- Let  $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$   $\mathbf{x}_i^\top = [1 \quad x_{i1} \quad \dots \quad x_{id}]$

- Can write the model in vectorized form as

$$h(\mathbf{x}_i) = \boldsymbol{\theta}^\top \mathbf{x}_i$$

# Closed Form Solution

- Consider our model  $h(\mathbf{x}_i) = \sum_{j=0}^d \theta_j x_{ij}$  for  $n$  instances:

- Let  $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$   $\mathbb{R}^{(d+1) \times 1}$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i,1} & \dots & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{bmatrix}$$

$\mathbb{R}^{n \times (d+1)}$

- Can write the model in vectorized form as  $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{X}\boldsymbol{\theta}$



# Closed Form Solution

- For the linear regression cost function:

Let:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\boldsymbol{\theta}^\top \mathbf{x}_i - y_i)^2 \\ &= \frac{1}{n} (\underbrace{\mathbf{X}\boldsymbol{\theta}}_{\mathbb{R}^{1 \times n}} - \underbrace{\mathbf{y}}_{\mathbb{R}^{n \times 1}})^\top (\underbrace{\mathbf{X}\boldsymbol{\theta}}_{\mathbb{R}^{n \times (d+1)}} - \underbrace{\mathbf{y}}_{\mathbb{R}^{(d+1) \times 1}})\end{aligned}$$

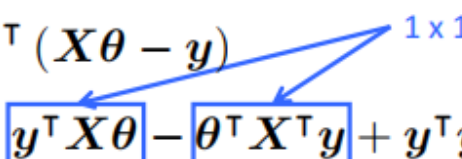
# Closed Form Solution

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\boldsymbol{\theta}^\top \mathbf{x}_i - y_i)^2 \\ &= \frac{1}{n} (\underbrace{\mathbf{X}\boldsymbol{\theta}}_{\mathbb{R}^{1 \times n}} - \underbrace{\mathbf{y}}_{\mathbb{R}^{n \times 1}})^\top (\underbrace{\mathbf{X}\boldsymbol{\theta}}_{\mathbb{R}^{1 \times n}} - \underbrace{\mathbf{y}}_{\mathbb{R}^{n \times 1}})\end{aligned}$$

$\mathbb{R}^{n \times (d+1)}$   
 $\mathbb{R}^{(d+1) \times 1}$

# Closed Form Solution

- Idea: Solve for optimal  $\theta$  analytically
- Notice that the solution is when  $\frac{\partial}{\partial \theta} \mathcal{L}(\theta) = 0$

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{n} (X\theta - y)^\top (X\theta - y) \\ &\propto \theta^\top X^\top X \theta - \boxed{y^\top X \theta} - \boxed{\theta^\top X^\top y} + y^\top y \\ &\propto \theta^\top X^\top X \theta - 2\theta^\top X^\top y + y^\top y\end{aligned}$$


Take derivative and set equal to 0, then solve for  $\theta$ :

$$\frac{\partial}{\partial \theta} (\theta^\top X^\top X \theta - 2\theta^\top X^\top y + \cancel{y^\top y}) = 0$$

$$(X^\top X)\theta - X^\top y = 0$$

$$(X^\top X)\theta = X^\top y$$

Closed Form Solution:

$$\theta = (X^\top X)^{-1} X^\top y$$

# Closed Form Solution

- Can obtain  $\theta$  by simply plugging  $X$  and  $y$  into  $\theta = (X^T X)^{-1} X^T y$

$$X = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i,1} & \dots & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- If  $X^T X$  is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that  $d \leq n$

# Closed Form Solution

- Can obtain  $\theta$  by simply plugging  $X$  and  $y$  into  $\theta = (X^T X)^{-1} X^T y$

$$X = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i,1} & \dots & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

This computation is intractable for even moderate size  $n$  and  $d$

- Computing  $(X^T X)^{-1}$  is roughly  $O(d^3)$
- $X$  and/or  $(X^T X)$  may be too large to fit in memory
- Numerical accuracy issues due to ill-conditioning

# Gradient Descent vs Closed Form

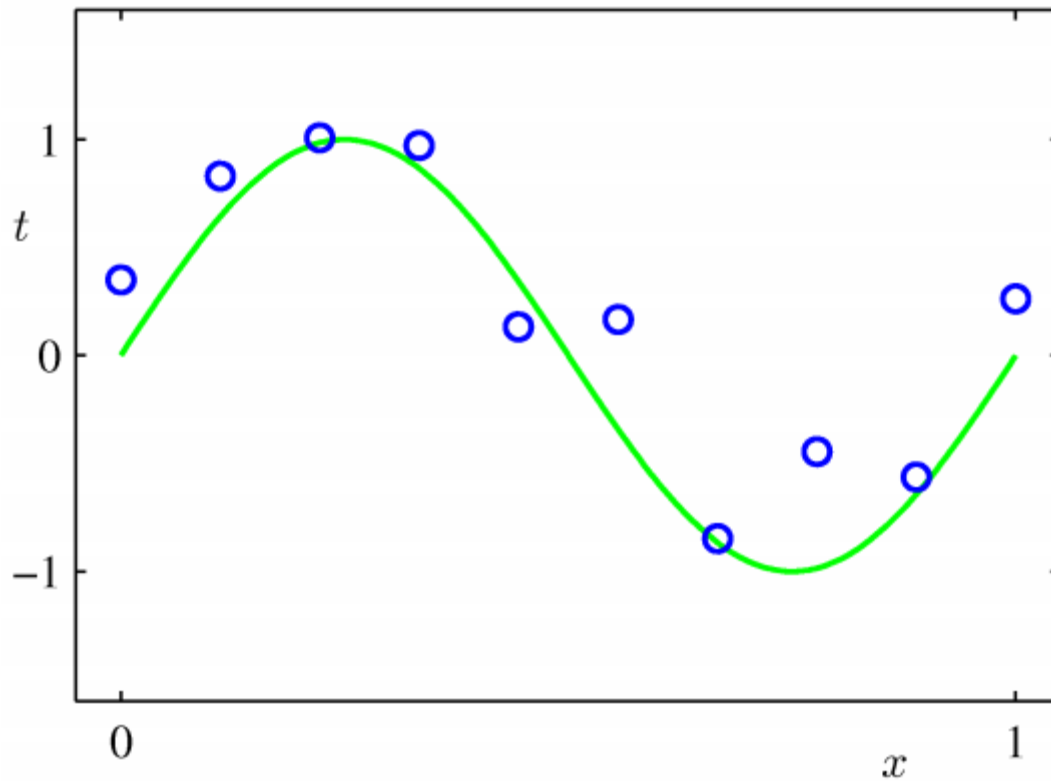
## Gradient Descent

- Requires multiple iterations
- Need to choose  $\alpha$
- Works well when  $n$  is large
- Can support incremental learning

## Closed Form Solution

- Non-iterative
- No need for  $\alpha$
- Slow if  $n$  and/or  $d$  is large  
Computing  $(X^T X)^{-1}$  is roughly  $O(d^3)$

# Example of Fitting a Polynomial Curve with a Linear Model



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j$$

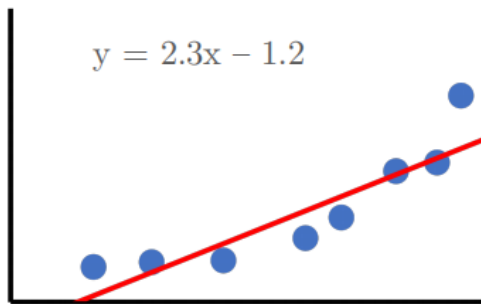
# Linear Basis Function Models

- Basic Linear Model: 
$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^d \theta_j x_j$$
- Generalized Linear Model: 
$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^d \theta_j \phi_j(\mathbf{x})$$
- Once we have replaced the data by the outputs of the basis functions,
- fitting the generalized model is exactly the same problem as fitting the
- basic model
  - Unless we use the [kernel trick](#) – more on that with support vector machines
  - Therefore, there is no point in cluttering the math with basis functions

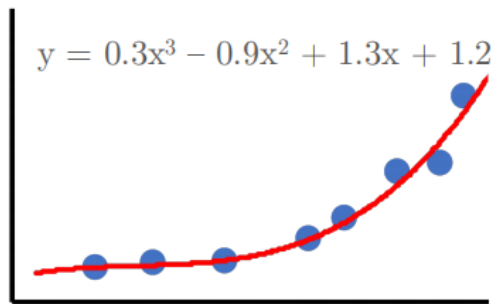


# Regularized Linear Regression

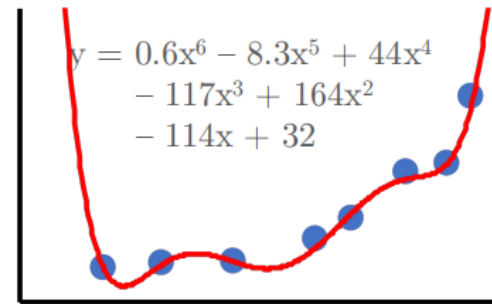
- Ridge Regression



**Underfitting**  
(high bias)



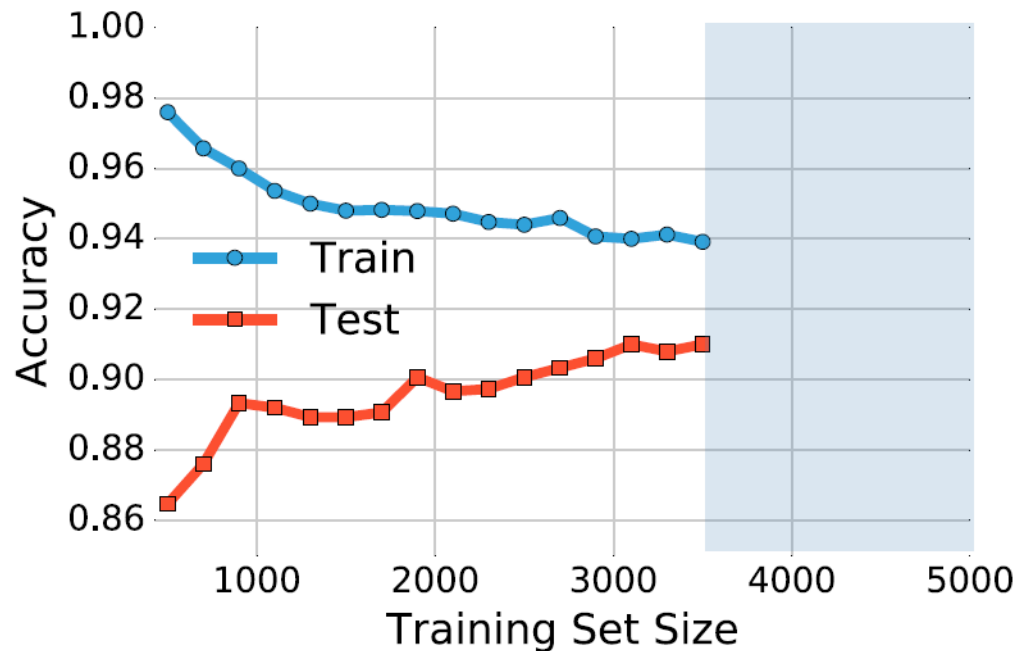
**Correct fit**



**Overfitting**  
(high variance)

- Overfitting occurs when:
  - The learned hypothesis fits the training set very well  $\mathcal{L}(\theta) \approx 0$
  - But fails to generalize to new examples

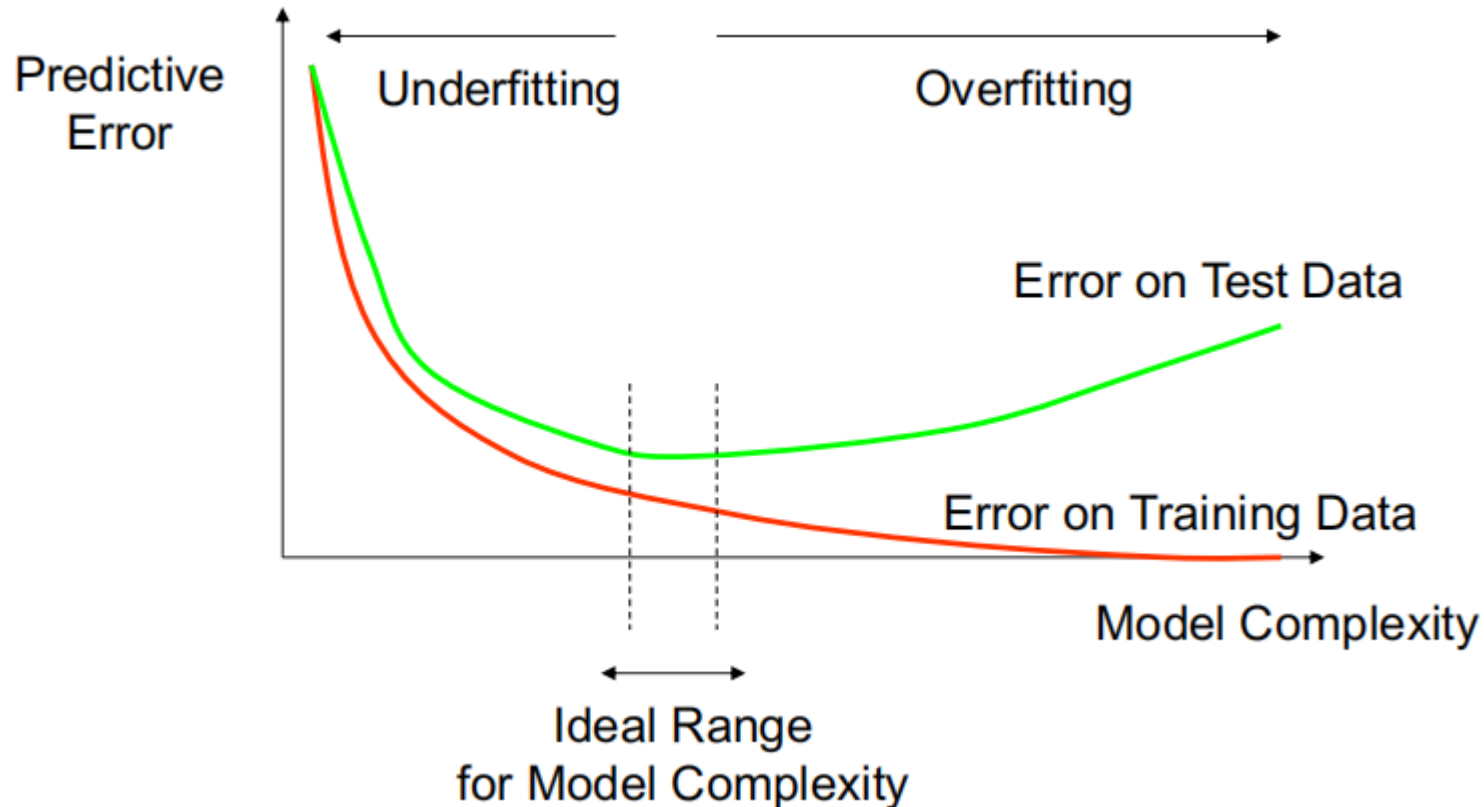
- Best Way to Reduce Overfitting is Collecting More Data



Softmax on MNIST subset (kept test set size constant)

- Collecting more data is always recommended
- If not possible, data augmentation is also helpful (e.g., for images: random rotation, crop, translation ...) -- actually, this is always recommended (and easy to do)
- Additionally, reducing the capacity (e.g., regularization) helps

# How Overfitting Affects Prediction



# Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- Idea: penalize for large values of  $\theta_j$ 
  - Incorporate penalty into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

# L2 Regularization

- Regularized linear regression objective function:

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2}_{\text{model fit to data}} + \underbrace{\lambda \sum_{j=1}^d \theta_j^2}_{\text{regularization}}$$

- is the regularization parameter  $\lambda \geq 0$
- No regularization on  $\theta_0$

# Understanding L2 Regularization

Cost Function: 
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- Note that 
$$\sum_{j=1}^d \theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$$

This is the magnitude of the feature coefficient vector squared!

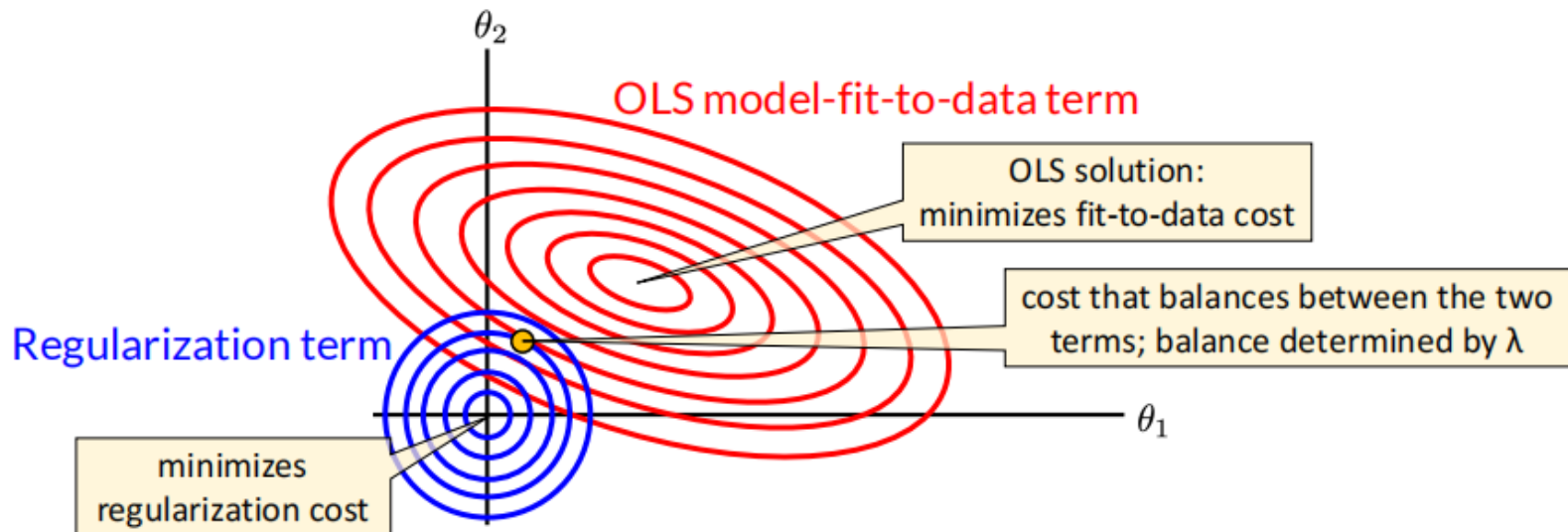
- We can also think of this as

$$\sum_{j=1}^d (\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{0}\|_2^2$$

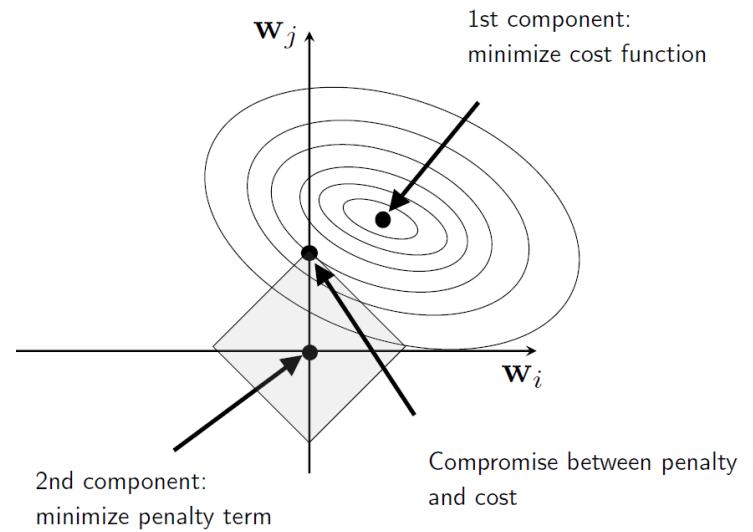
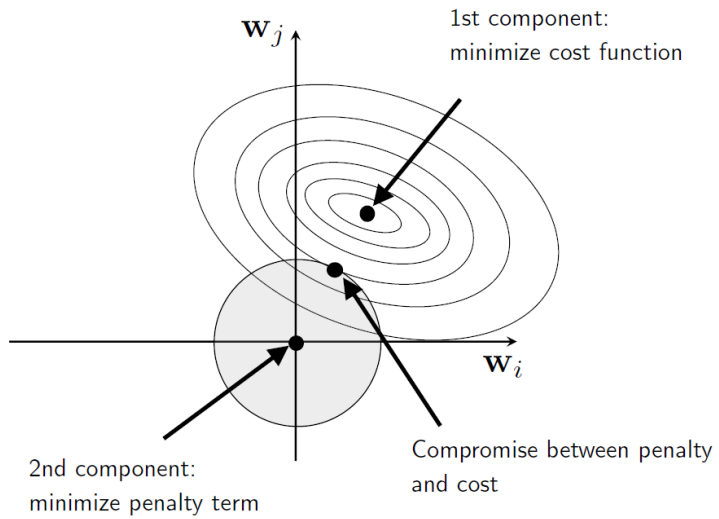
L2 regularization pulls coefficients toward 0

# Understanding L2 Regularization

Cost Function: 
$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^d \theta_j^2$$





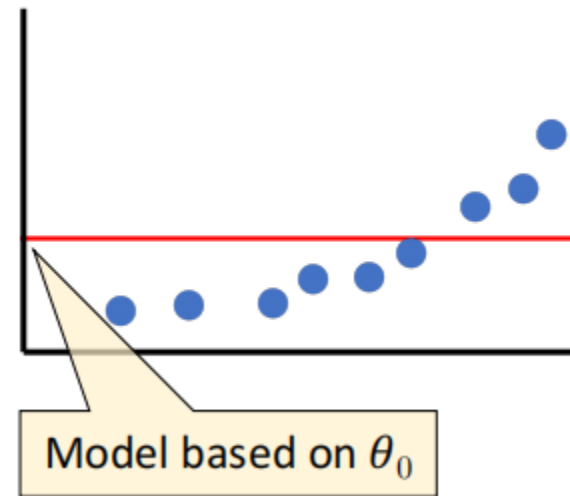


# Understanding L2 Regularization

Cost Function: 
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

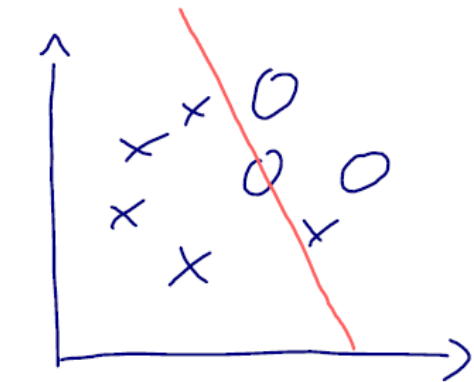
- What happens as  $\lambda \rightarrow \infty$ ?

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

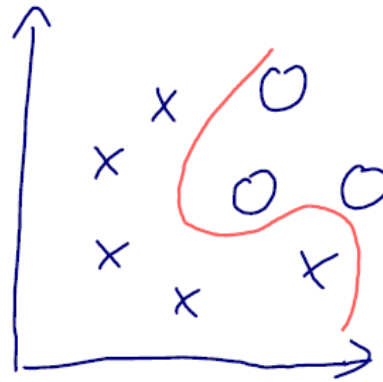


# Effect of Norm Penalties on the Decision Boundary

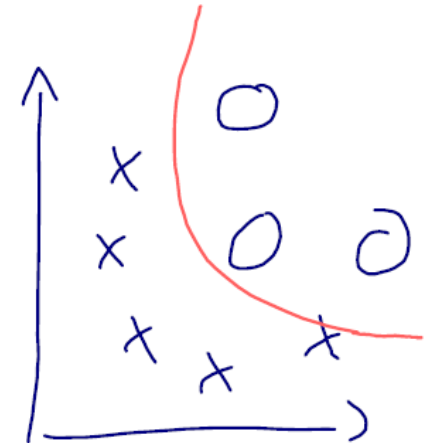
- Assume a nonlinear model



Large regularization penalty  
 $\Rightarrow$  high bias

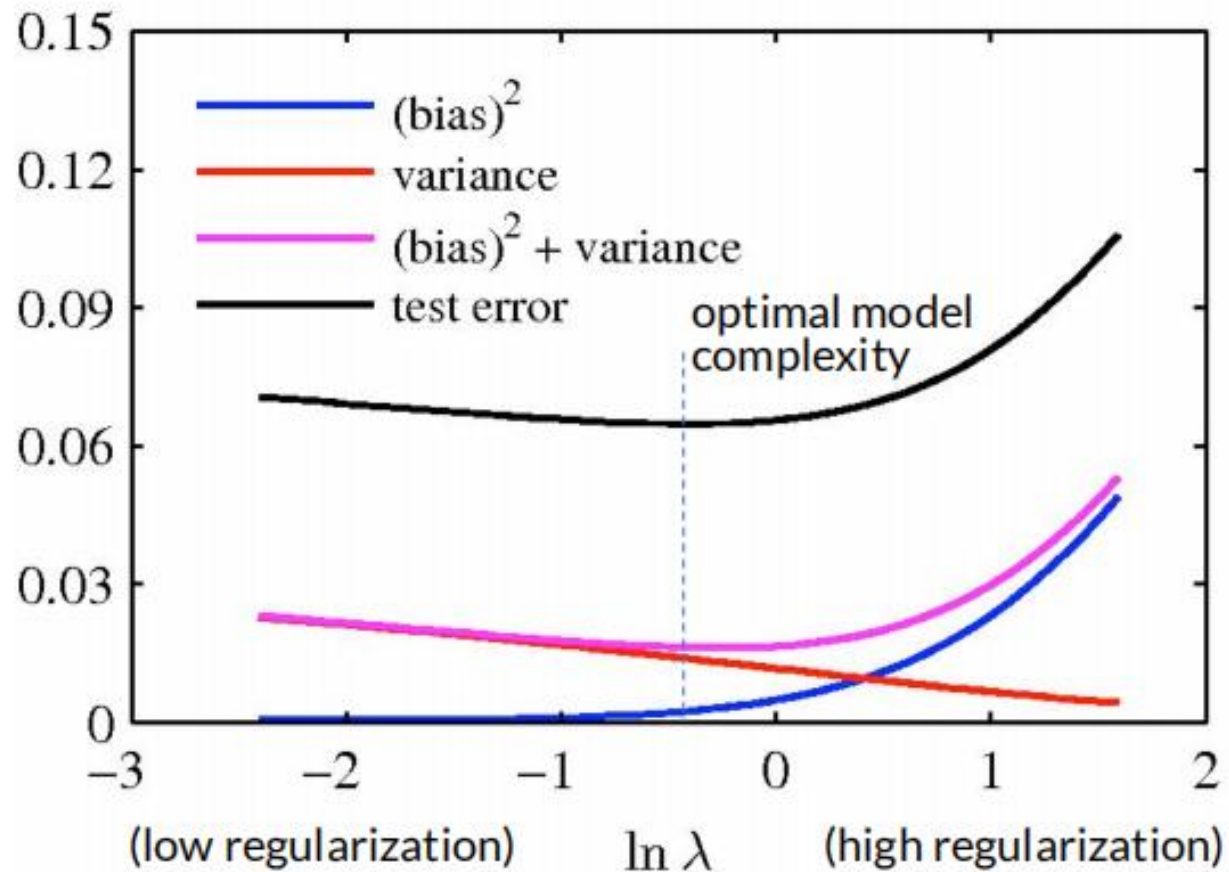


Low regularization  
 $\Rightarrow$  high variance



Good compromise

# Illustration of Bias-Variance



# Illustration of Bias-Variance

$$E(f; D) = \underbrace{bias^2(\mathbf{x})}_{\text{red}} + \underbrace{var(\mathbf{x})}_{\text{blue}} + \underbrace{\varepsilon^2}_{\text{green}}$$

期望输出与真实输出的差别

$$bias^2(\mathbf{x}) = (\bar{f}(\mathbf{x}) - y)^2$$

$$var(\mathbf{x}) = \mathbb{E}_D \left[ (f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2 \right]$$

同样大小的训练集的变动，所导致的性能变化

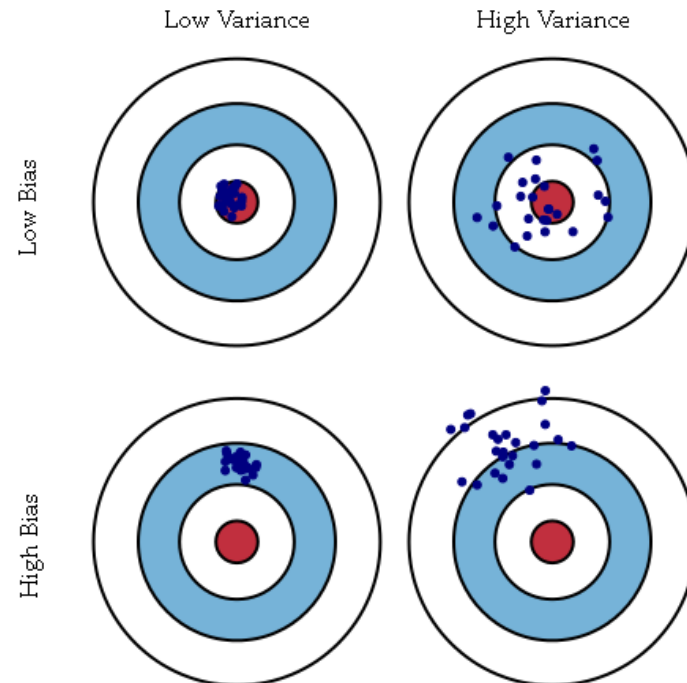
训练样本的标记与真实标记有区别

$$\varepsilon^2 = \mathbb{E}_D \left[ (y_D - y)^2 \right]$$

表达了当前任务上任何学习算法所能达到的期望泛化误差下界

# Illustration of Bias-Variance

- Bias: a learner's tendency to consistently learn the same wrong thing
- Variance: the tendency to learn random things irrespective of the real signal



# Regularized Linear Regression

Cost Function: 
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^d \theta_j^2$$

- Fit by solving  $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} \mathcal{L}(\boldsymbol{\theta}) \quad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)$$

No regularization on  $\theta_0$

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}) \quad \theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i) x_{ij} - \underbrace{\alpha \lambda \theta_j}_{\text{regularization}}$$

Extra factor of 2  
in derivatives  
absorbed into  $\alpha$

regularization

# Regularized Linear Regression

Cost Function:  $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^d \theta_j^2$

Gradient update:

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i) x_{ij} - \alpha \lambda \theta_j$$

$$\theta_j \leftarrow \theta_j (1 - \alpha \lambda) - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i) x_{ij}$$



# Regularized Closed Form Linear Regression

- To incorporate regularization into the closed form solution:

$$\theta = \left( X^T X + \lambda I \right)^{-1} X^T y$$

# Regularized Closed Form Linear Regression

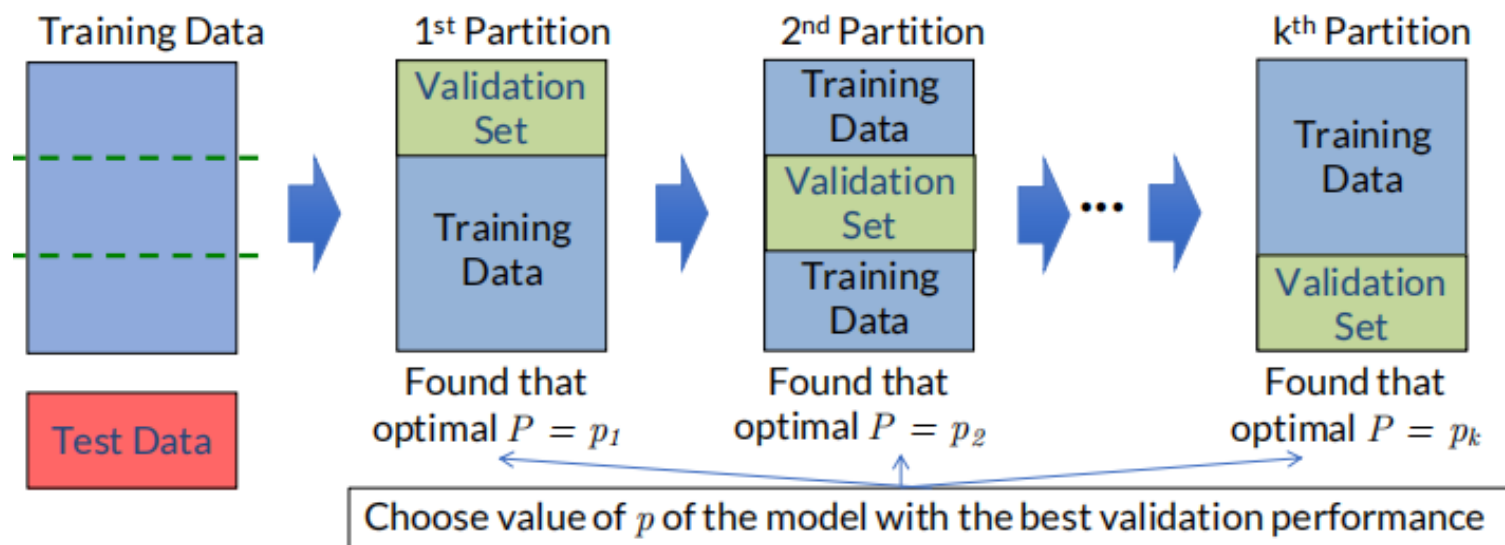
- To incorporate regularization into the closed form solution:

$$\boldsymbol{\theta} = \left( \mathbf{X}^\top \mathbf{X} + \lambda \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \right)^{-1} \mathbf{X}^\top \mathbf{y}$$

- Can derive this the same way, by solving  $\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = 0$
- Can prove that for  $\lambda > 0$ , inverse exists in the equation above

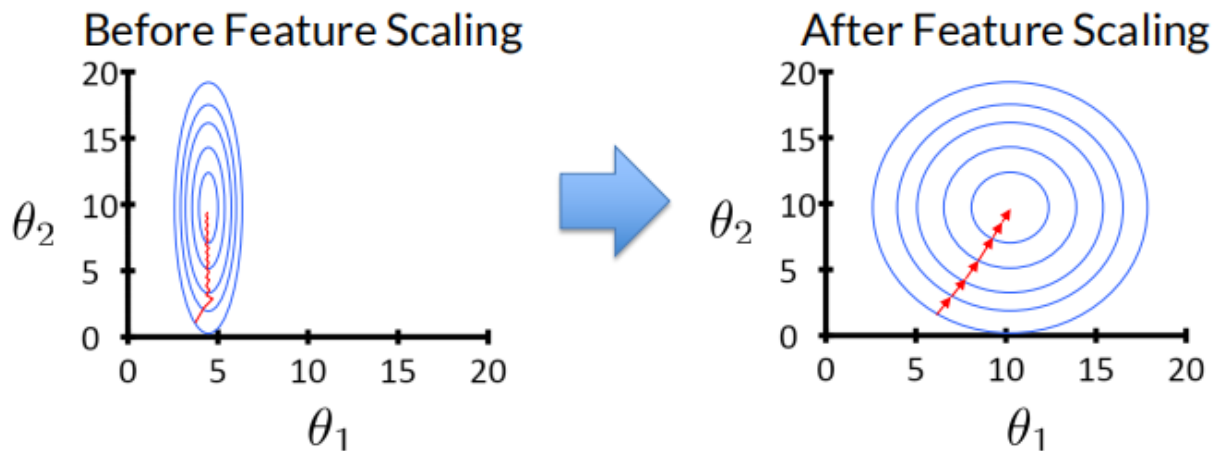
# Optimizing Model Parameters

- Using **Cross Validation** to choose value of model parameter  $P$  :
- Search over space of parameter values
  - Evaluate model with  $P = p$  on validation set
- Choose value  $p'$  with highest validation performance
- Learn model on full training set with  $P = p'$



# Improving Learning: Feature Scaling

- Idea: Ensure that feature have similar scales



- • Makes gradient descent converge much faster
- • Important for ridge regression, since it preferentially shrinks large coefficients

# Feature Standardization

- Rescales features to have zero mean and unit variance

- Let  $\mu_j$  be the mean of feature  $j$ : 
$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$$

- Replace each value with: 
$$x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j} \quad \text{for } j = 1 \dots d \text{ (not } x_0\text{!)}$$

- $s_j$  is the standard deviation of feature  $j$
- Could also use the range of feature  $j$  (max value – min value) for  $s_j$

- Outliers can cause problems

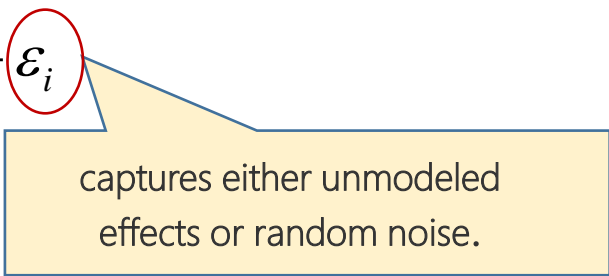
Must use the **same transformation** for both training and prediction

# Linear Regression

- **Why** might linear regression, and specifically why might the least-squares cost function  $J$ , be a reasonable choice?

$$\min \frac{1}{2} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2 = \min J(\theta)$$

- Assume that the target variables and the inputs are related via the equation

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$


captures either unmodeled effects or random noise.

The effect resulted from the model aren't modeled properly, such as if there are some features very pertinent to predicting housing price, but that we'd left out of the regression.

# Linear Regression

- Further assume that the  $\varepsilon_i$  are distributed **IID (independently and identically distributed)** according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance  $\sigma^2$

$$\varepsilon^{(i)} \sim N(0, \sigma^2)$$

$$X \sim N(\mu, \sigma^2)$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < +\infty,$$

- I.e., the density of  $\varepsilon^{(i)}$  is given by

$$P(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)$$

# Linear Regression

- This implies that

$$P(y_i | x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}\right)$$

- The notation  $P(y_i | x_i; \theta)$  indicates that this is the distribution of  $y_i$  given  $x_i$  and parameterized by  $\theta$
- Given  $\mathbf{X}$  (the design matrix, which contains all the  $\mathbf{x}_i$ ) and  $\boldsymbol{\theta}$ , what is the distribution of the  $\boldsymbol{\varepsilon}_i$ 's?
- The probability of the data is given by . This quantity is typically viewed a function of (and perhaps  $\mathbf{X}$ ), for a fixed value of  $w$ .



# Linear Regression

- When we wish to explicitly view this as a function of  $w$ , we will instead call it the **likelihood function**:

$$L(\theta) = L(\theta; \mathbf{X}, \mathbf{y}) = P(\mathbf{y} | \mathbf{X}; \theta)$$

- Note that by the independence assumption on the  $y_i$ 's (and hence also the  $\mathbf{x}_i$ 's given the  $\mathbf{x}_i$ 's), this can also be written

$$\begin{aligned} L(\theta) &= \prod_{i=1}^n P(y_i | \mathbf{x}_i; \mathbf{w}) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) \end{aligned}$$

# Linear Regression

- ? Given this probabilistic model relating the  $x$ 's and the  $y$ 's, what is a reasonable way of choosing our best guess of the parameters  $w$ ?
- The principle of maximum likelihood says that we should choose  $w$  so as to make the data as high probability as possible. I.e., we should choose  $w$  to maximize  $L(w)$ .

# Linear Regression

- Instead of maximizing  $L(\theta)$ , we can also maximize any strictly increasing function of  $L(\theta)$ . In particular, the derivations will be a bit simpler if we instead maximize the log likelihood  $\ell(\theta)$ :

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) \\ &= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^n (y_i - \theta^T \mathbf{x}_i)^2\end{aligned}$$

# Linear Regression

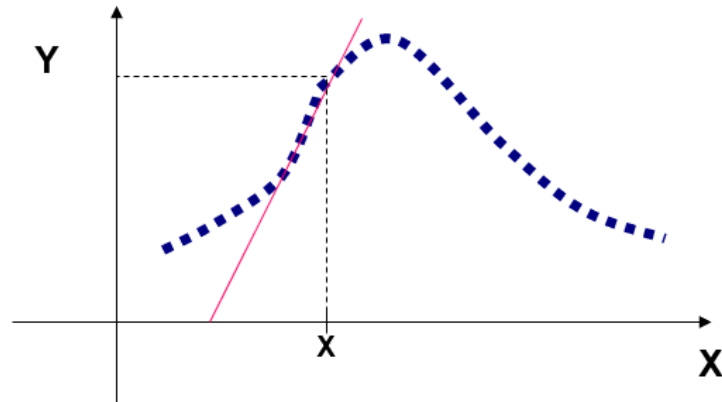
- Hence, maximizing  $J(\boldsymbol{\theta})$  gives the same answer as minimizing

$$\frac{1}{2} \sum_{i=1}^n (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2$$

- which we recognize to be  $J(\boldsymbol{\theta})$ , our original least-squares cost function.
- Under the previous probabilistic assumptions on the data, least-squares regression corresponds to finding the maximum likelihood estimate of  $\boldsymbol{\theta}$ .
- This is thus one set of assumptions under which least-squares regression can be justified as a very natural method that's just doing maximum likelihood estimation.

# Locally Weighted Regression

- The choice of features is important to ensuring good performance of a learning algorithm.
- **Locally weighted linear regression (LWR)** algorithm which, assuming there is sufficient training data, makes the choice of features less critical.



To evaluate  $h$  at a certain  $x$

$$\text{LR : Fit } \mathbf{w} \text{ to } \min \sum_i (y^{(i)} - \mathbf{w}^t \mathbf{x}^{(i)})^2$$

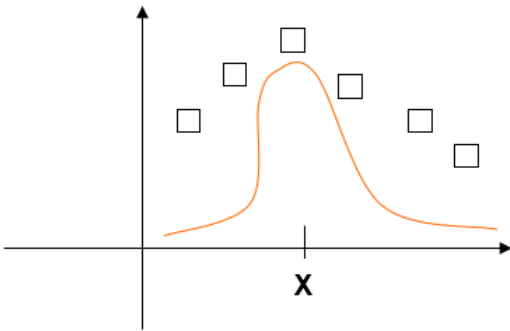
Return  $\mathbf{w}^t \mathbf{x}$

# Locally Weighted Regression

LWR : Fit  $\theta$  to  $\min \sum_i \omega^{(i)} (y^{(i)} - \theta^T \mathbf{x}^{(i)})^2$

Return  $\theta^T \mathbf{x}$

where  $\omega^{(i)}$ 's are non-negative valued weights



Intuitively, if  $\omega^{(i)}$  is large for a particular value of  $i$ , then in picking  $\mathbf{w}$ , we'll try hard to make  $(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$  small. If  $\omega^{(i)}$  is small, then the  $(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$  error term will be pretty much ignored in the fit.

# Locally Weighted Regression

Normally  $\omega^{(i)} = \exp\left(-\frac{(\mathbf{x}^{(i)} - \mathbf{x})^2}{2\tau^2}\right)$

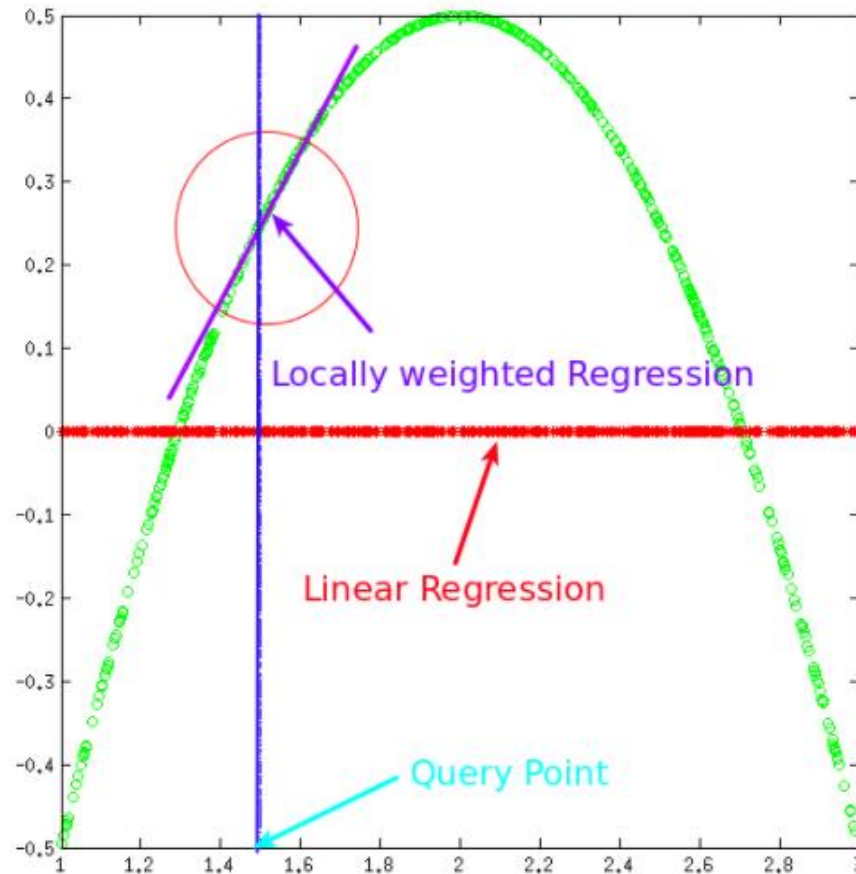
**bandwidth**

If  $|\mathbf{x}^{(i)} - \mathbf{x}|$  small, then  $\omega^{(i)} \approx 1$

If  $|\mathbf{x}^{(i)} - \mathbf{x}|$  large, then  $\omega^{(i)} \approx 0$

$$\min \sum_i \omega^{(i)} (y^{(i)} - \theta^t \mathbf{x}^{(i)})^2$$

# Locally Weighted Regression





# Locally Weighted Regression

- "Parametric" learning algorithm
  - 's fixed set of parameters
- "Non-parametric" learning algorithm
  - No. of parameters grows with  $m$
- **Locally Weighted Regression is non-parametric learning algorithm**

The (unweighted) linear regression algorithm is known as a parametric learning algorithm

# Logistic Regression

- For instance, if we are trying to build a spam classifier for email.
- Then  $x^{(i)}$  may be some features of a piece of email, and  $y$  may be 1 if it is a piece of spam mail, and 0 otherwise.
- 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+".
- Given  $x_i$ , the corresponding  $y_i$  is also called the label for the training example.

# Logistic Regression

- We could approach the classification problem ignoring the fact that  $y$  is discrete-valued, and use our old linear regression algorithm to try to predict  $y$  given  $x$ .

It is easy to construct examples where this method performs very poorly. Intuitively, it also doesn't make sense for  $h_{\theta}(x)$  to take values larger than 1 or smaller than 0 when we know that  $y \in \{0, 1\}$ .

To fix this, let's change the form for our hypotheses  $h_{\theta}(x)$ . We will choose

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

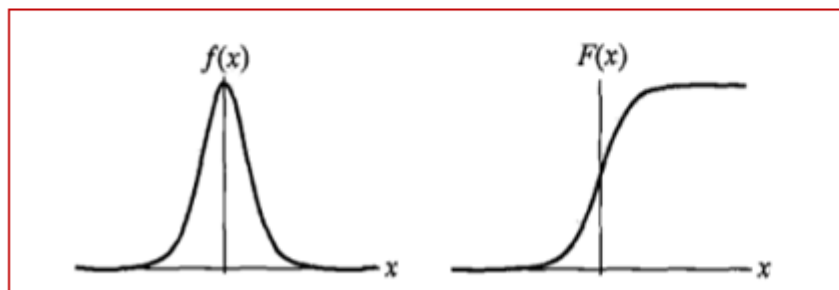
# Logistic Regression

## Logistic distribution

$$F(x) = P(X \leq x) = \frac{1}{1 + e^{-(x-\mu)/\gamma}}$$
$$f(x) = F'(x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma(1 + e^{-(x-\mu)/\gamma})^2}$$

位置参数:  $\mu$

形状参数:  $\gamma > 0$




密度函数

分布函数

## Sigmoid curve:

关于点  $(\mu, 1/2)$

$$F(-x + \mu) - \frac{1}{2} = -F(x - \mu) + \frac{1}{2}$$

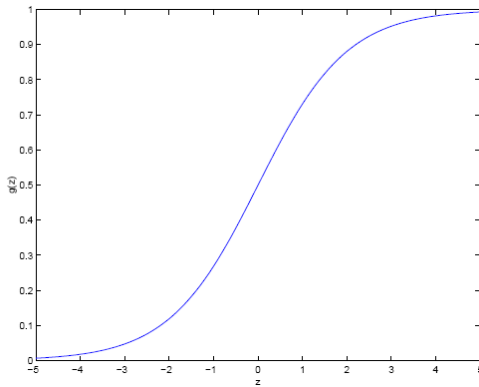

$$F(x) = P(X \leq x) = \frac{1}{1 + e^{-(x-u)/\gamma}}$$

# Logistic Regression

- Logistic Function

$$g(z) = \frac{1}{1 + e^{-z}}$$

- is called the **logistic function** or the **sigmoid function**. Here is a plot



$g(z) \rightarrow 1$  as  $z \rightarrow \infty$ ,  
and  $g(z) \rightarrow 0$  as  $z \rightarrow -\infty$ .  
 $g(z) \in [0, 1]$ ,  $h_{\theta}(\mathbf{x}) \in [0, 1]$

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\ &= \frac{1}{(1 + e^{-z})^2} (e^{-z}) \\ &= \frac{1}{(1 + e^{-z})} \cdot \left( 1 - \frac{1}{(1 + e^{-z})} \right) \\ &= g(z)(1 - g(z)) \end{aligned}$$

Here's a useful property of the derivative of the sigmoid function, which we write as :

# Logistic Regression

- ? Given the logistic regression model, how do we fit  $\theta$  for it?

We will endow our classification model with a set of probabilistic assumptions, and then fit the parameters via **maximum likelihood**.

# Logistic Regression

- Assuming that

$$\begin{aligned}P(y = 1|x; \theta) &= h_{\theta}(x) \\P(y = 0|x; \theta) &= 1 - h_{\theta}(x)\end{aligned}$$

- Note that this can be written more compactly as

$$P(y|x; \theta) = h_{\theta}(x)^y (1 - h_{\theta}(x))^{1-y}$$

- Assuming that the  $m$  training examples were generated independently, we can then write down the likelihood of the parameters as:

$$\begin{aligned}L(\theta) &= P(\mathbf{y}|x; \theta) \\&= \prod_i P(y^{(i)}|x^{(i)}; \theta) \\&= \prod_i h_{\theta}(x^{(i)})^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}}\end{aligned}$$



# Logistic Regression

- As before, it will be easier to maximize the log likelihood

$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \end{aligned}$$

对  $l(\theta)$  求极大值，得到对  $\theta$  的估计值

How do we maximize the likelihood?

Similar to our derivation in the case of linear regression, we can use gradient ascent. Written in vectorial notation, our updates will therefore be given by

$$\theta := \theta + \alpha \nabla_{\theta} l(\theta)$$

# Logistic Regression

- Let's start by working with just one training example  $(x, y)$ , and take derivatives to derive the stochastic gradient ascent rule:

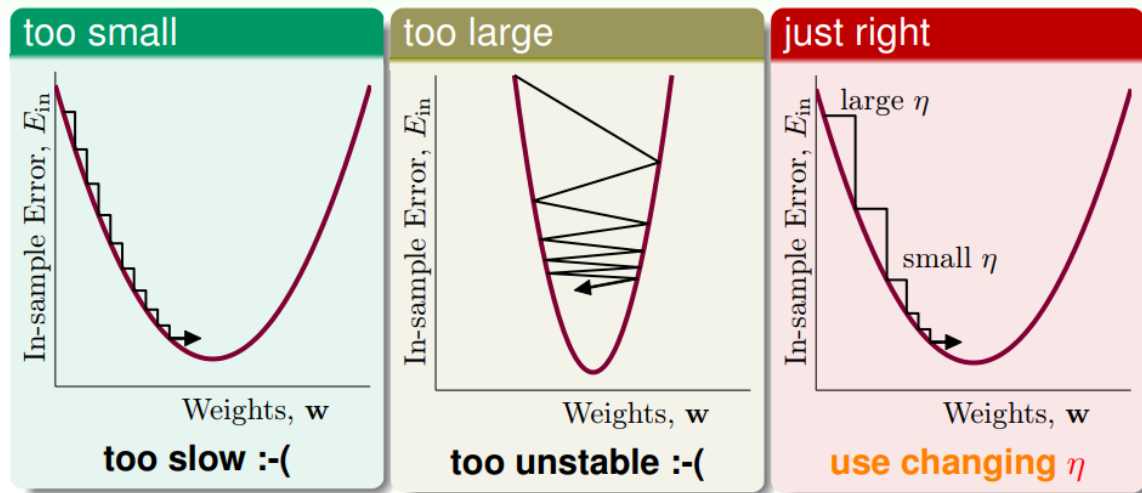
$$\begin{aligned}\frac{\partial}{\partial \theta_j} l(\theta) &= \left( y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= \left( y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)} \right) g(\theta^T x) (1-g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\ &= (y(1-g(\theta^T x)) - (1-y)g(\theta^T x)) x_j \\ &= (y - h_\theta(x)) x_j\end{aligned}$$

Above, we used the fact that  $g'(z) = g(z)(1-g(z))$

- This therefore gives us the stochastic gradient ascent rule

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

Comparing this to the LMS update rule, we see that it looks identical; but this is not the same algorithm, because  $h_{\theta}$  is now defined as a non-linear function of



Newton's method ? Is ok ?

数据搜集



数据清洗



特征工程



数据建模



# Softmax Regression

- Binary classification:  $y^{(i)} \in \{0,1\}$
- the hypothesis took the form

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

- the model parameters  $\theta$  were trained to minimize the cost function

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

# Softmax Regression

- Multinomial logistic regression  $y^{(i)} \in \{1, 2, \dots, K\}$
- (Note that our convention will be to index the classes starting from 1, rather than from 0.)
- hypothesis will output a K-dimensional vector (whose elements sum to 1)

$$h_{\theta}(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \dots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(-\theta^{(j)T} x)} \begin{bmatrix} \exp(-\theta^{(1)T} x) \\ \exp(-\theta^{(2)T} x) \\ \dots \\ \exp(-\theta^{(K)T} x) \end{bmatrix}$$

Where  $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)} \in R^n$  are the parameters of our model.

$$\theta = \begin{bmatrix} | & | & \dots & | \\ \theta^{(1)} & \theta^{(2)} & \dots & \theta^{(K)} \\ | & | & | & | \end{bmatrix}.$$

# Softmax Regression

- Cost Function

$$J(\theta) = - \left[ \sum_{i=1}^m \sum_{k=1}^K 1 \{y^{(i)} = k\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^K \exp(\theta^{(j)\top} x^{(i)})} \right]$$

$$\begin{aligned} J(\theta) &= - \left[ \sum_{i=1}^m (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) + y^{(i)} \log h_{\theta}(x^{(i)}) \right] \\ &= - \left[ \sum_{i=1}^m \sum_{k=0}^1 1 \{y^{(i)} = k\} \log P(y^{(i)} = k | x^{(i)}; \theta) \right] \end{aligned}$$

sum over the K different possible values of the class label

$$P(y^{(i)} = k | x^{(i)}; \theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^K \exp(\theta^{(j)\top} x^{(i)})}$$

# Softmax Regression

- Iterative optimization algorithm

$$\begin{aligned}
 \frac{\partial L(\theta)}{\partial \theta_j} &= -\frac{1}{m} \frac{\partial}{\partial \theta_j} \left[ \sum_{i=1}^m \sum_{j=1}^k 1\{y_i = j\} \log \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right] \\
 &= -\frac{1}{m} \frac{\partial}{\partial \theta_j} \left[ \sum_{i=1}^m \sum_{j=1}^k 1\{y_i = j\} \left( \theta_j^T x_i - \log \sum_{l=1}^k e^{\theta_l^T x_i} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m 1\{y_i = j\} \left( x_i - \sum_{j=1}^k \frac{e^{\theta_j^T x_i} \cdot x_i}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i 1\{y_i = j\} \left( 1 - \sum_{j=1}^k \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i \left( 1\{y_i = j\} - \sum_{j=1}^k 1\{y_i = j\} \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i \left( 1\{y_i = j\} - \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i (1\{y_i = j\} - p(y_i = j|x_i; \theta)) \right]
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial L(\theta)}{\partial \theta_j} &= -\frac{1}{m} \left[ \sum_{i=1}^m \frac{\partial}{\partial \theta_j} \left( 1\{y_i = j\} \log \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} + \sum_{c \neq j}^k 1\{y_i = c\} \log \frac{e^{\theta_c^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m \left( 1\{y_i = j\} \left( x_i - \frac{e^{\theta_j^T x_i} \cdot x_i}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) + \sum_{c \neq j}^k 1\{y_i = c\} \left( -\frac{e^{\theta_c^T x_i} \cdot x_i}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i \left( 1\{y_i = j\} \left( 1 - \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) - \sum_{c \neq j}^k 1\{y_i = c\} \frac{e^{\theta_c^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i \left( 1\{y_i = j\} - 1\{y_i = j\} p(y_i = j|x_i; \theta) - \sum_{c \neq j}^k 1\{y_i = c\} p(y_i = c|x_i; \theta) \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i \left( 1\{y_i = j\} - \sum_{j=1}^k 1\{y_i = j\} p(y_i = j|x_i; \theta) \right) \right] \\
 &= -\frac{1}{m} \left[ \sum_{i=1}^m x_i (1\{y_i = j\} - p(y_i = j|x_i; \theta)) \right]
 \end{aligned}$$



# Softmax Regression

Regularization:



overfitting

$$L(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{j=1}^k 1\{y_i = j\} \log \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}} \right] + \lambda \sum_{i=1}^k \sum_{j=1}^n \theta_{ij}^2$$

Optimization:

gradient:

$$\frac{\partial L(\theta)}{\partial \theta_j} = -\frac{1}{m} \left[ \sum_{i=1}^m x_i (1\{y_i = j\} - p(y_i = j|x_i; \theta)) \right] + \lambda \theta_j$$

# Softmax Regression

Over-parameterized

$$\begin{aligned}(y_i = j | x_i; \theta) &= \frac{e^{(\theta_j - \psi)^T x_i}}{\sum_{l=1}^k e^{(\theta_l - \psi)^T x_i}} \\&= \frac{e^{\theta_j^T x_i} e^{-\psi^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i} e^{-\psi^T x_i}} \\&= \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}}\end{aligned}$$

# Softmax Regression

## ◆ softmax VS. logistic

$$K=2 \quad h_{\theta}(x_i) = \begin{bmatrix} p(y_i = 1|x_i; \theta) \\ p(y_i = 2|x_i; \theta) \end{bmatrix} = \frac{1}{e^{\theta_1^T x_i} + e^{\theta_2^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \end{bmatrix}$$



$$\begin{aligned} h_{\theta}(x_i) &= \frac{1}{e^{\vec{0}^T x_i} + e^{(\theta_2 - \theta_1)^T x_i}} \begin{bmatrix} e^{\vec{0}^T x_i} \\ e^{(\theta_2 - \theta_1)^T x_i} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{1 + e^{(\theta_2 - \theta_1)^T x_i}} \\ \frac{e^{(\theta_2 - \theta_1)^T x_i}}{1 + e^{(\theta_2 - \theta_1)^T x_i}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{1 + e^{(\theta_2 - \theta_1)^T x_i}} \\ 1 - \frac{1}{1 + e^{(\theta_2 - \theta_1)^T x_i}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{1 + e^{(\theta'}^T x_i)} \\ 1 - \frac{1}{1 + e^{(\theta')^T x_i}} \end{bmatrix} \end{aligned}$$

# Multi-Class Classification

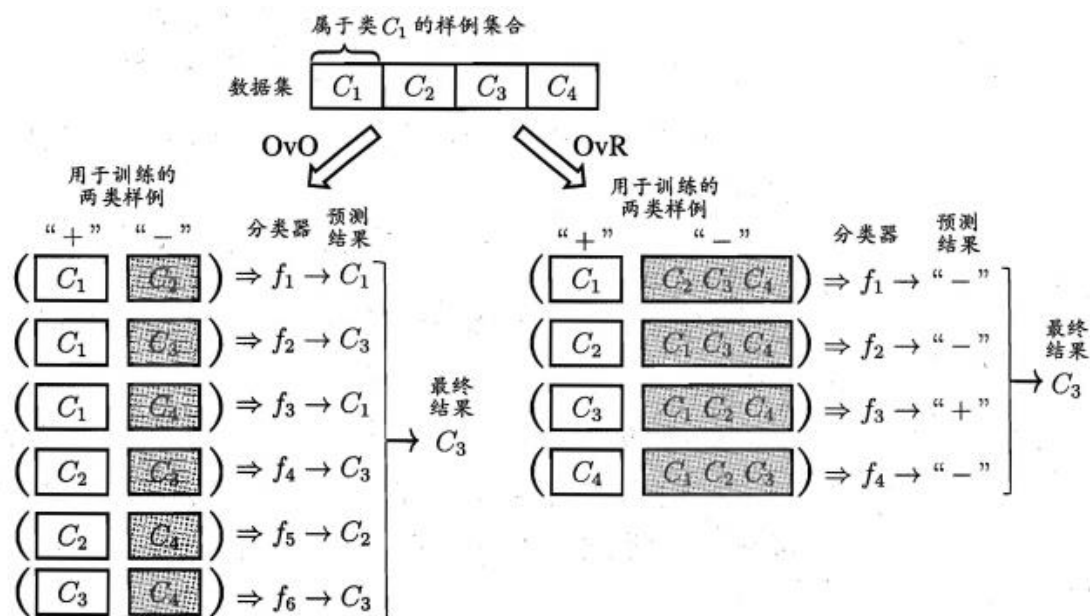
- 多分类（Multi-class Classification） ---》 [Multi-Label Classification](#)
- -- refers to those classification tasks that have more than two class labels.
- Examples include:
  - Face classification.
  - Plant species classification.
  - Optical character recognition.

Algorithms that are designed for binary classification can be adapted for use for multi-class problems.

- Logistic Regression.
- Support Vector Machine.

# Multi-Class Classification

- **One-vs-Rest:** Fit one binary classification model for each class vs. all other classes.
- **One-vs-One:** Fit one binary classification model for each pair of classes
- **Many-vs-Many:** --ECOC



# Multi-Class Classification

- One-vs-Rest: Fit one binary classification model for each class vs. all other classes.
- One-vs-One: Fit one binary classification model for each pair of classes
- Many-vs-Many: --ECOC

进行比较, 将距离最小的编码所对应的类别作为预测结果. 例如在图 3.5(a) 中, 若基于欧氏距离, 预测结果将是  $C_3$ .

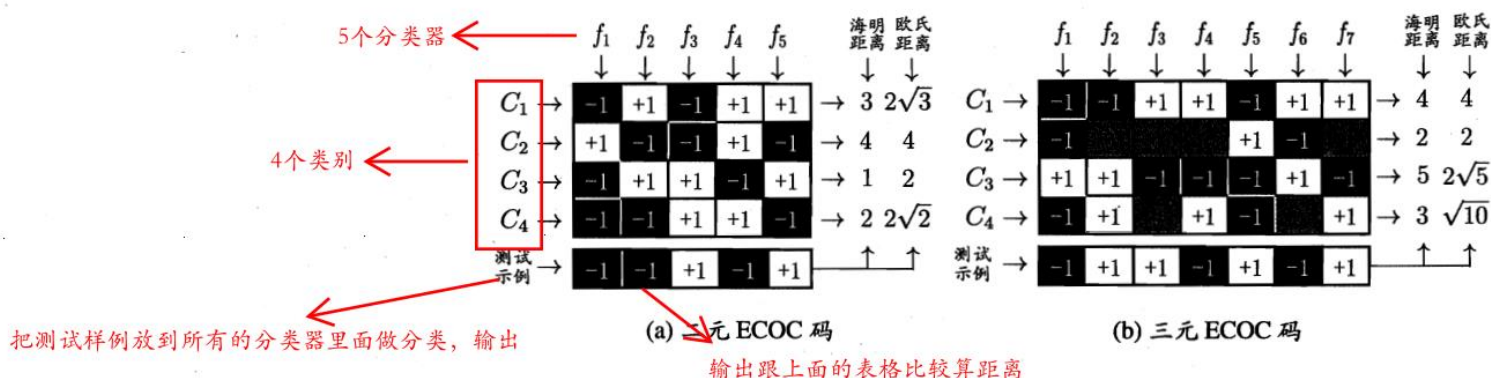
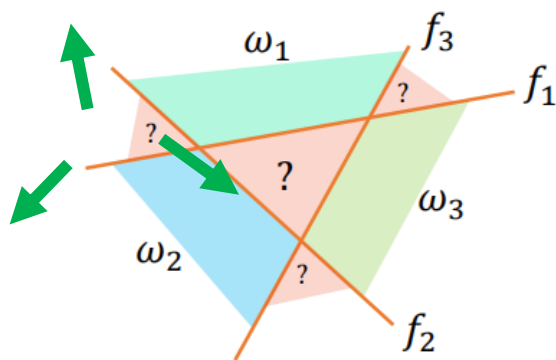


图 3.5 ECOC 编码示意图. “+1”、“-1” 分别表示学习器  $f_i$  将该类样本作为正、反例; 三元码中 “0” 表示  $f_i$  不使用该类样本

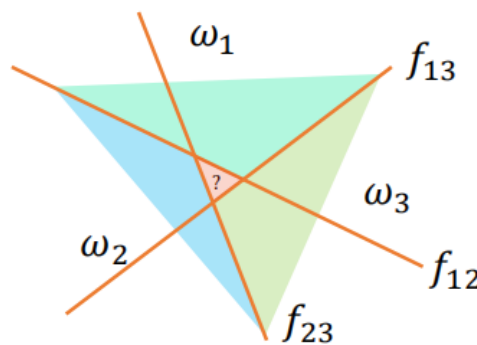
<http://blog.csdn.net/hit2015spring>

# Multi-Class Classification

- **One-vs-Rest:** Fit one binary classification model for each class vs. all other classes.
- **One-vs-One:** Fit one binary classification model for each pair of classes
- **Many-vs-Many:** --ECOC



(a) “一对其余”方式



(b) “一对一”方式

# Classification evaluation index

- 混淆矩阵（Confuse Matrix）

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$



# Classification evaluation index

- 准确率(Accuracy)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- 精准率(Precision)

$$Accuracy = \frac{TP}{TP + FP}$$

# Classification evaluation index

- 召回率(Recall)

$$Accuracy = \frac{TP}{TP + FN}$$

- F1 Score --- balance P & R

$$F1 = \frac{2 \times P \times R}{P + R}$$

$$Precision = \frac{1}{1 + 0} = 1$$

$$Recall = \frac{1}{1 + 9} = 0.1$$

	预测为好	预测为坏
标签为好	1	9
标签为坏	0	10

$$Precision = \frac{10}{10 + 10} = 0.5$$

$$Recall = \frac{10}{10 + 0} = 1$$

	预测为好	预测为坏
标签为好	10	0
标签为坏	10	0

不同的分类问题，对精确率和召回率的要求也不同！

# Regression evaluation index

- 平均绝对误差 MAE (Mean absolute error)

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$$

- 均方差 MSE (Mean squared error)

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|^2$$

$$\sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$

Root Mean Squard Error

- Goodness of Fit --R-Squared