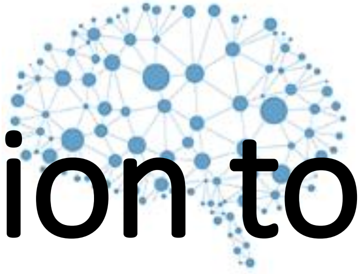


Introduction to Machine Learning

ENEE 6583 Neural Nets

Dr. Alsamman

Slide Credits: A Ng,

Related Fields

- Statistical Estimation
  - old; math; small; foundation
- Pattern Recognition
  - 60s; images; engineering
- Machine Learning
  - 80s; CS
- Artificial Intelligence
  - ML; CS
- Data Mining
  - 90s; data; info theory

Learning

- Supervised:
  - Labeled training
- Unsupervised
  - Unlabeled
- Semisupervised
  - Label deficient
- Reinforcement
  - Reward actions

Supervised Learning

- Goal: Given a labeled training data set, use a learning function to generate a good predictor of the output for any input.
- Notation:

$\boldsymbol{X}$	ALL input
$\Re^n$	n = dimension of input: $\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}$
$\boldsymbol{Y}$	ALL output
$\left(x^{(i)}, y^{(i)}\right)$	Training sample i
m	Size of training set: $\left\{\left(x^{(i)}, y^{(i)}\right);\ i = 1, \dots, m\right\}$
$h\left(\boldsymbol{x}^{(\boldsymbol{i})}\right)$	Learning function, aka, hypothesis
$h: \boldsymbol{X} \rightarrow \boldsymbol{Y}$	Learning is the mapping of input to output

Linear Regression

- Linear: line fitting
- Regression: reduction
- Learning function:

$$\widehat{\boldsymbol{Y}} = h\left(\boldsymbol{X}^{(i)}\right) = \boldsymbol{W}^T\boldsymbol{X} + b = w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b$$

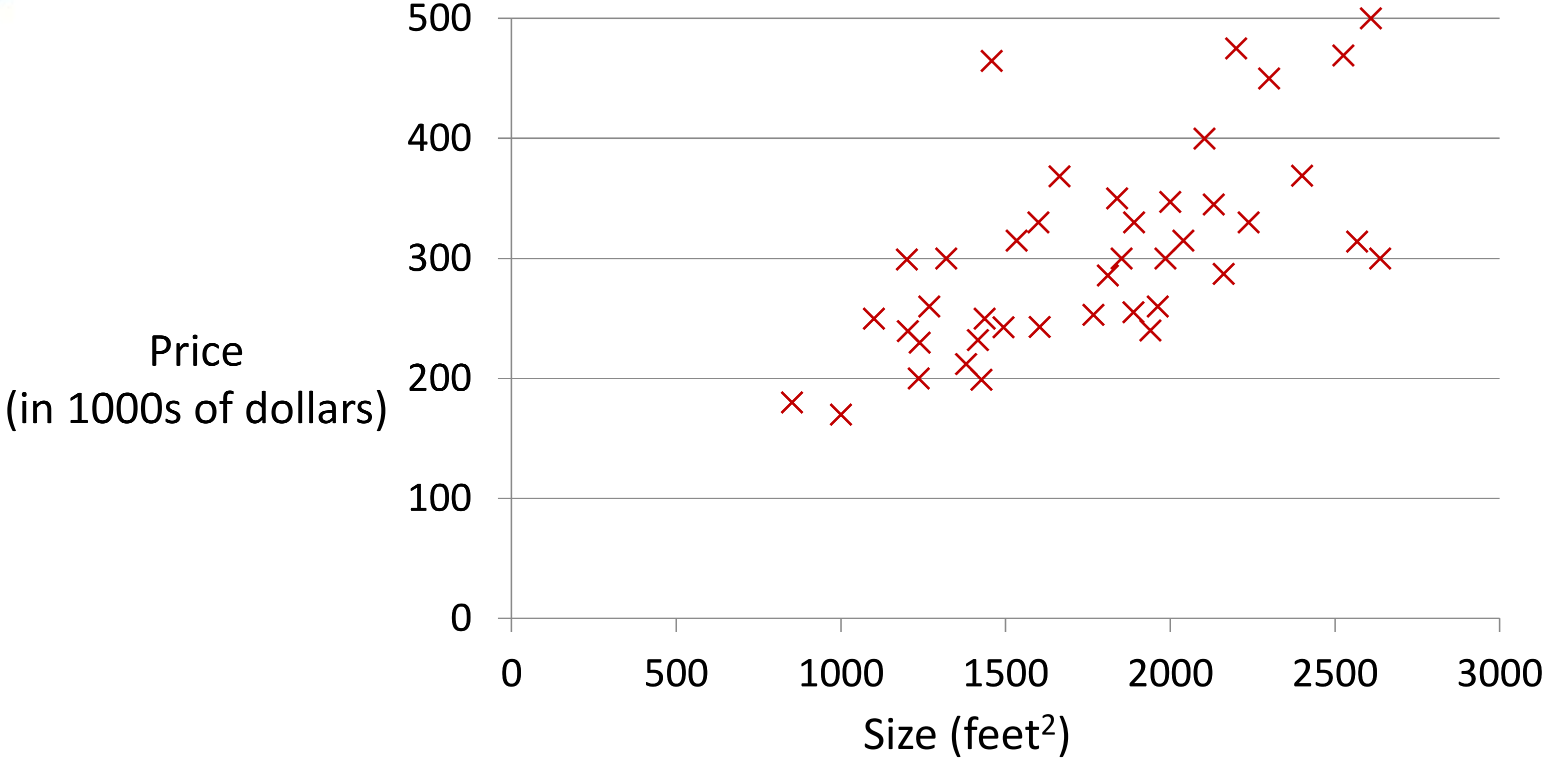
  - $\boldsymbol{W}, b$: learning parameters
  - $\boldsymbol{X}$: input data (aka features)
  - $\widehat{\boldsymbol{Y}}$: prediction
- Goal: $h(\boldsymbol{X})$ must be as close to $\boldsymbol{Y}$ (ground truth, aka labels) as possible

Example

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m=47

- How many inputs/outputs?
- Dimension of input/output?
- Hypothesis function?

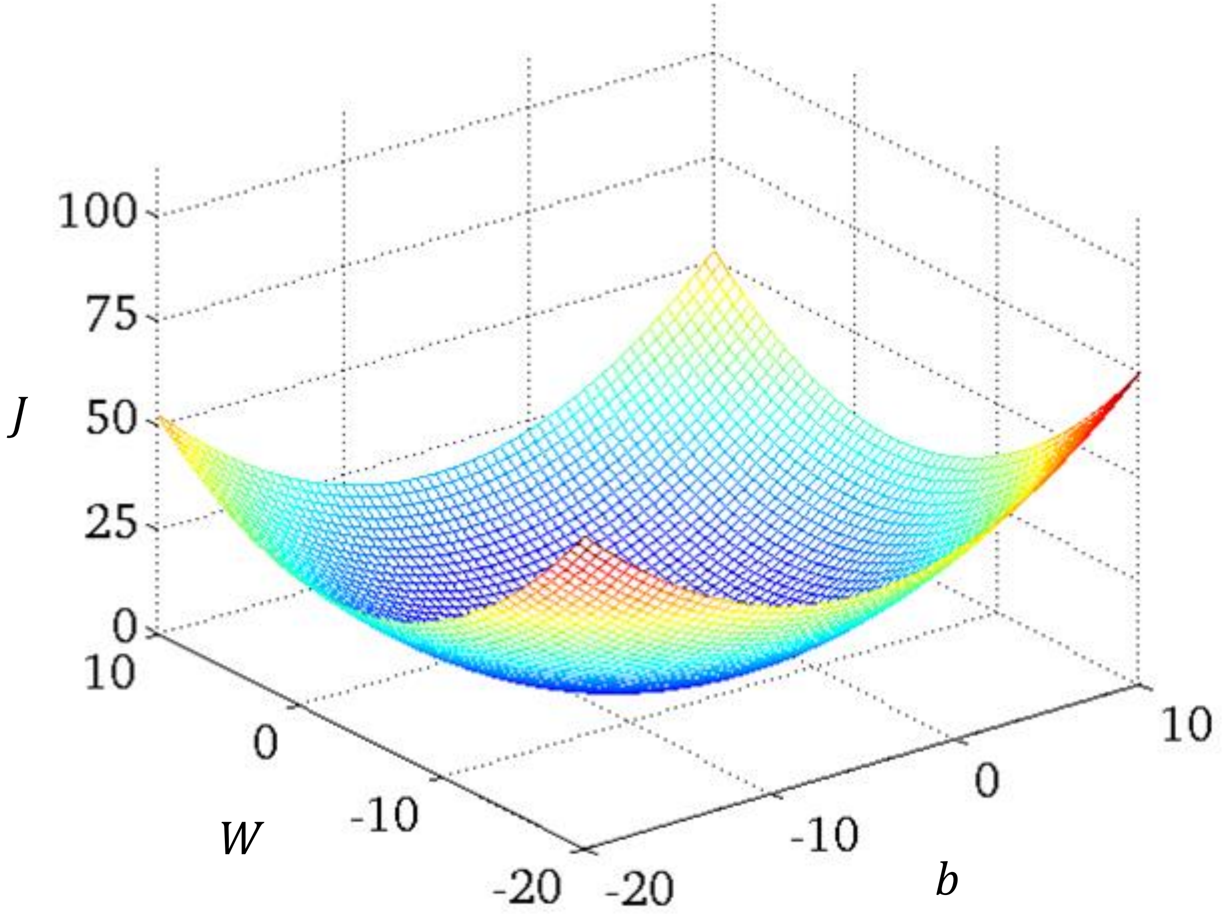

Price
(in 1000s of dollars)
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
Size (feet^2)

Cost Function

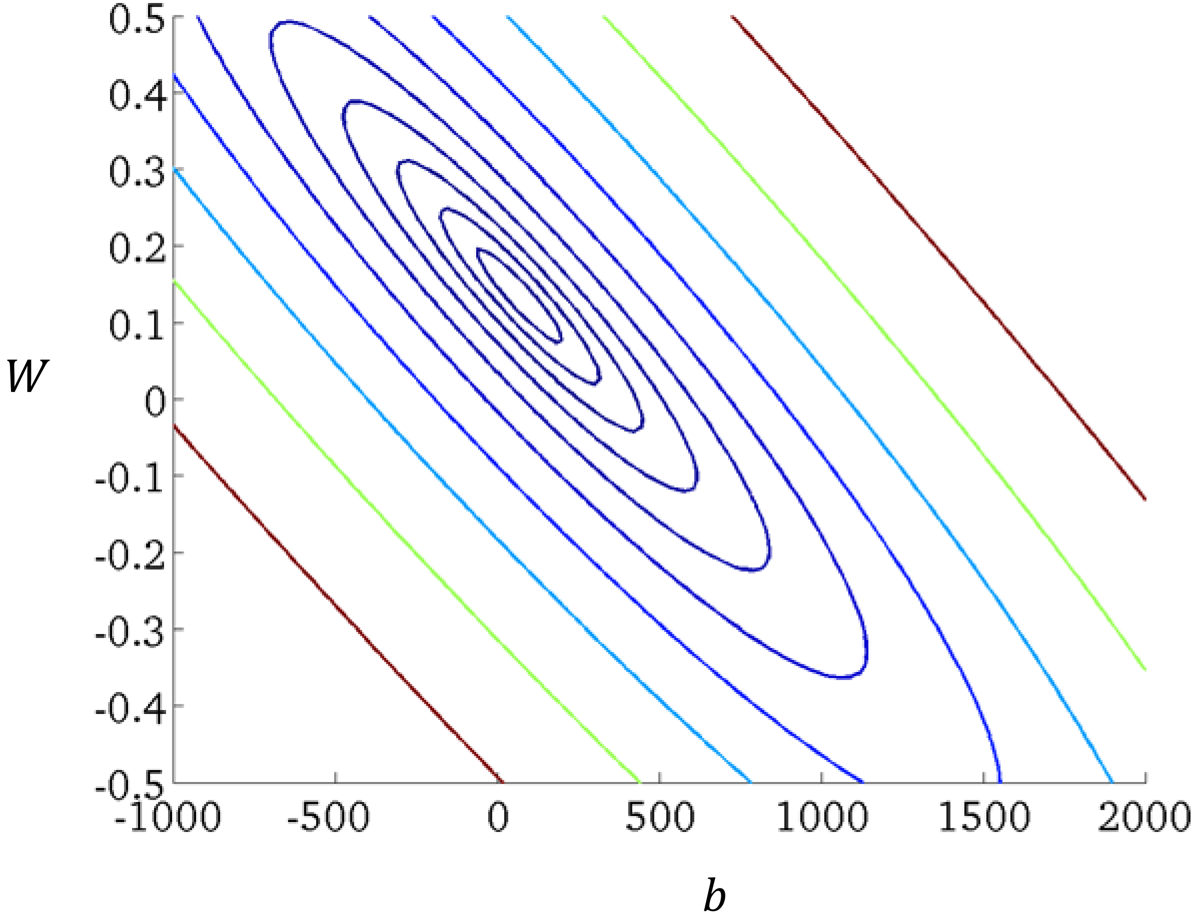
- Measure of how close $h(x^i)$ is to y
- Least squares cost function:

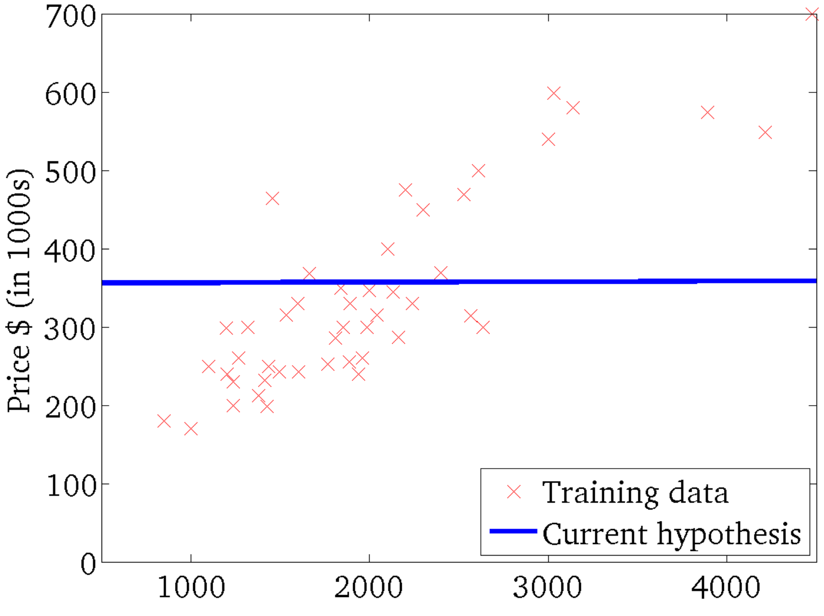
$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

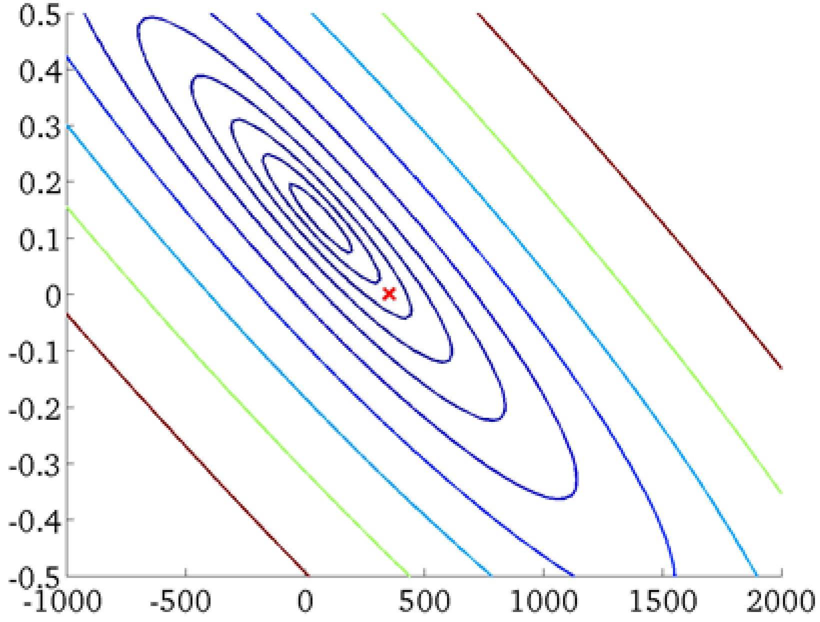
- Goal: minimize cost

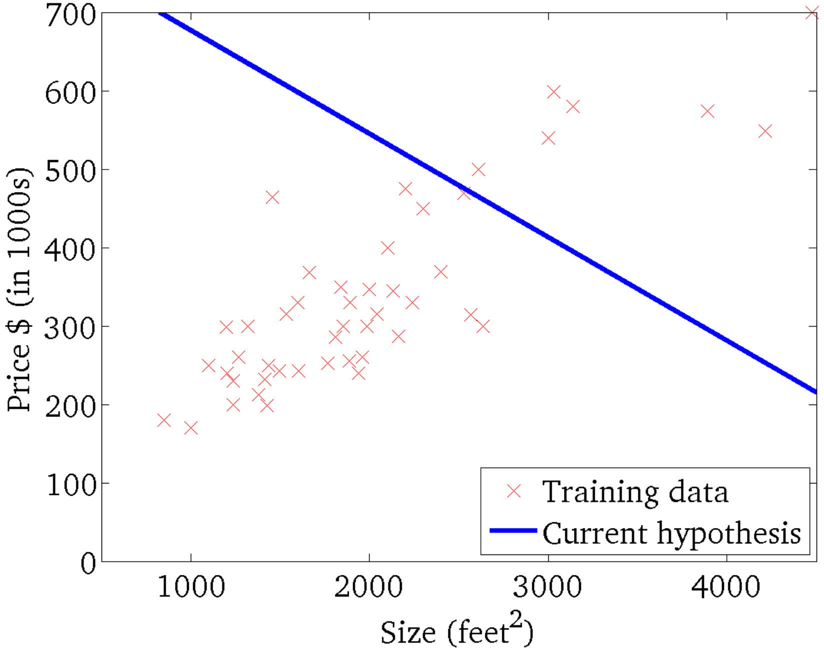


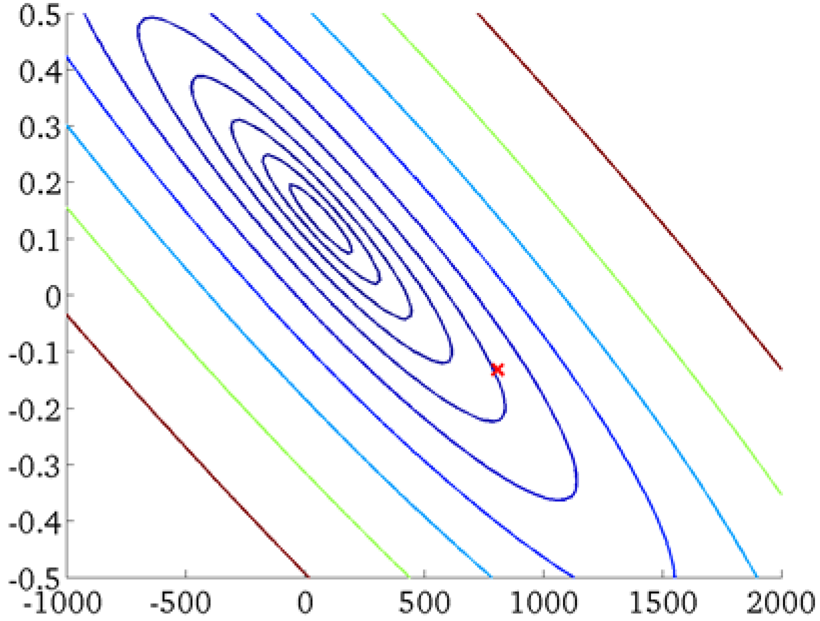
J
100
75
50
25
0
10
0
-10
-20
W
-20
-10
0
10
b

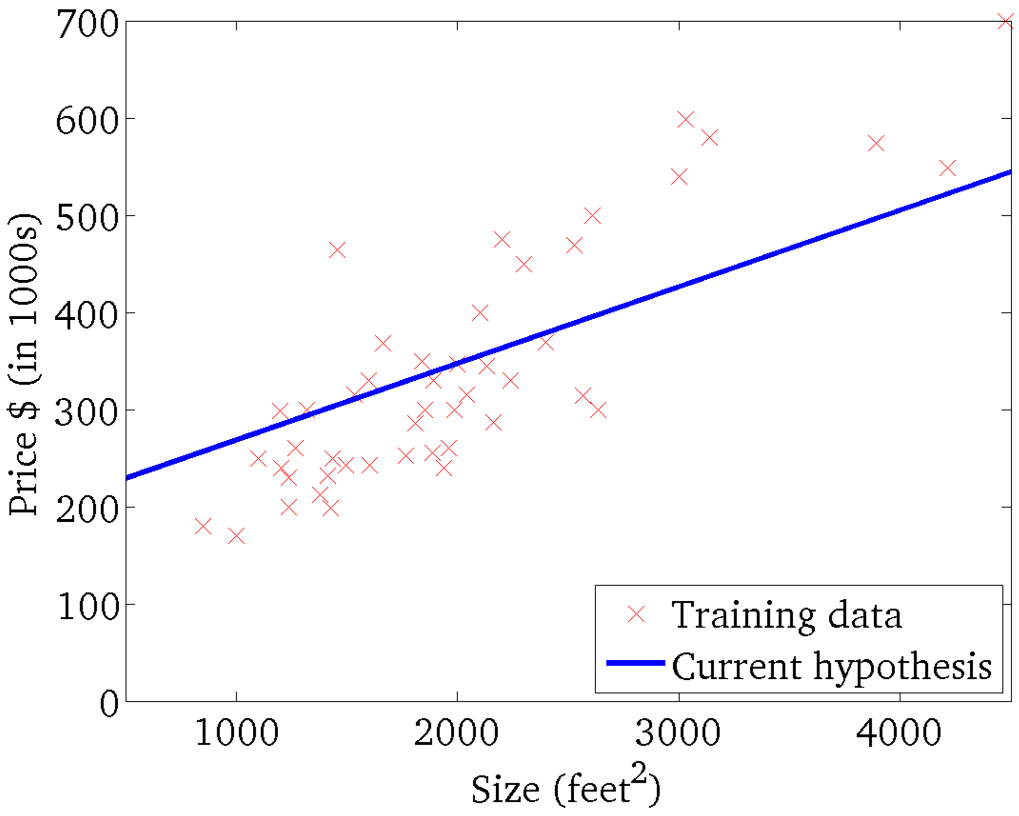

0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
W
-1000
-500
0
500
1000
1500
2000
b

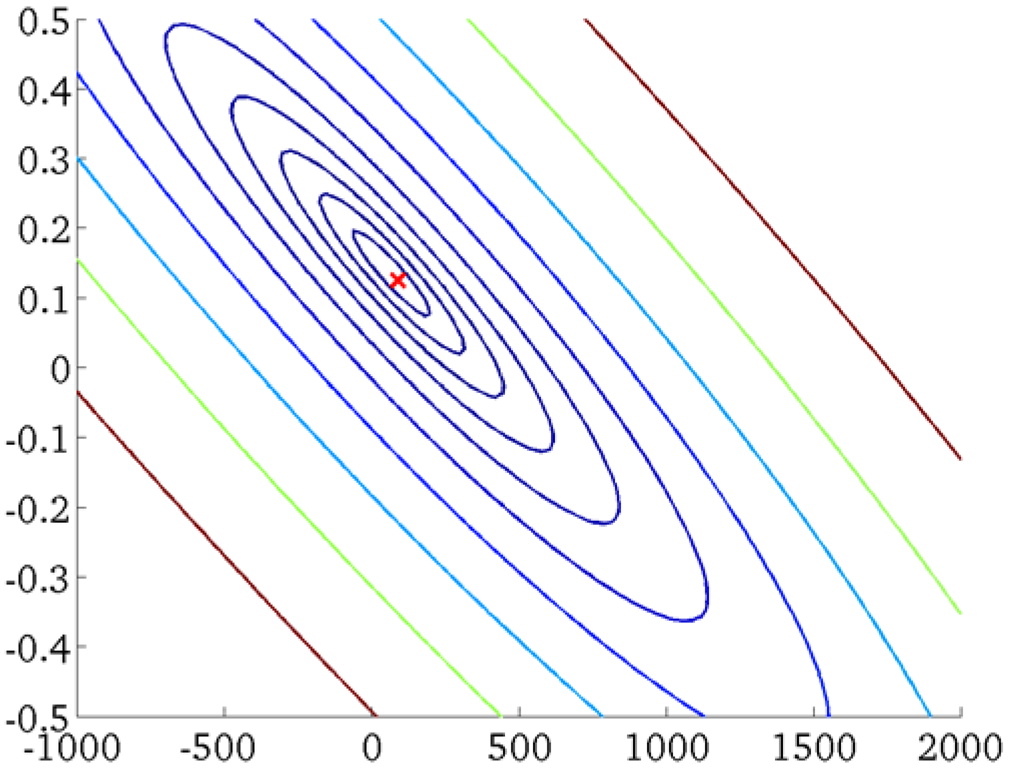

Price $ (in 1000s)
Training data
Current hypothesis




Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis




700
600
500
400
300
200
100
0
1000
2000
3000
4000
Size (feet^2)
Price $ (in 1000s)
Training data
Current hypothesis


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1000
-500
0
500
1000
1500
2000

Gradient Descent

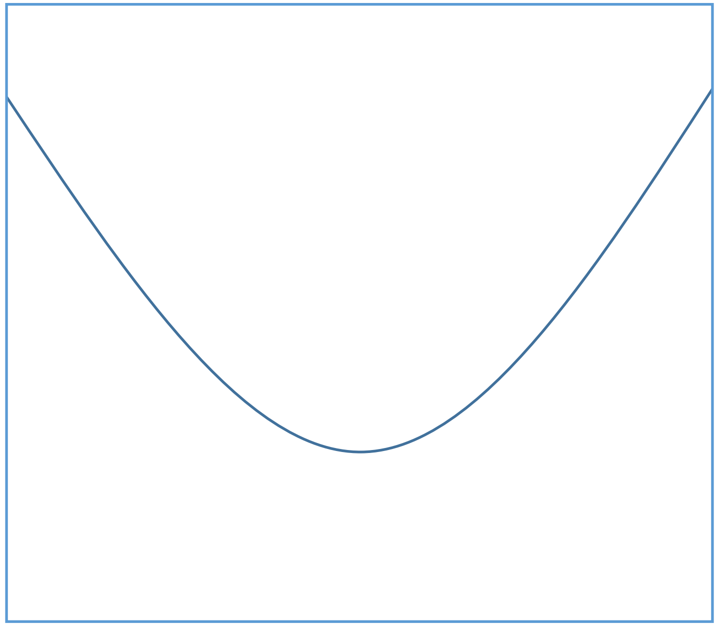
$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

- Gradient: use derivate/slope
- Descent: make sure that with each iteration cost is decreasing
- Gradient descent algorithm:
  1. Start with initial values for W, b
  2. Compute J
  3. Update all varaiables **simultaneously**:

     $$w = w - \alpha \frac{\partial}{\partial w} J(w, b) \qquad b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$
  4. Goto 2 until w, b converge.

Gradient Descent

- Understanding the effect of gradient descent
  - W,b: learning parameters
  - Learning rate α: hyperparameter



Gradient Descent

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$w = w - \alpha \frac{\partial}{\partial w_j} J(w,b)$$

- Batch gradient

$$w = w - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(\left(h\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}\right), \qquad b = b - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)$$

- Stochastic Gradient Descent

$$w = w - \alpha\left(h\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}, \qquad b = b - \alpha\left(h\left(x^{(i)}\right) - y^{(i)}\right)$$

Multiple Features: $\boldsymbol{X}$

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

- $m = 47$
- $\boldsymbol{X} = \left[\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})}\right]$

Multivariate Linear Regression

- Multivariate:

$$h\left(\boldsymbol{X}^{(i)}\right) = W_1 x_1^{(i)} + W_2 x_2^{(i)} + \cdots + W_n x_n^{(i)} + b$$

$$J(\boldsymbol{W}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(\boldsymbol{X}^{(i)}\right) - \boldsymbol{Y}^{(i)}\right)^2$$

- Gradient Descent:

$$W_j = W_j - \alpha \frac{\partial}{\partial W_j} J(\boldsymbol{W}, b)$$

$$W_j = W_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\left(h\left(\boldsymbol{X}^{(i)}\right) - \boldsymbol{Y}^{(i)}\right) x_j{}^{(i)}\right)$$

- Stochastic Descent:

$$W_j = W_j - \alpha\left(h\left(\boldsymbol{X}^{(i)}\right) - \boldsymbol{Y}^{(i)}\right) x_j{}^{(i)}$$

Matrix Based Solution

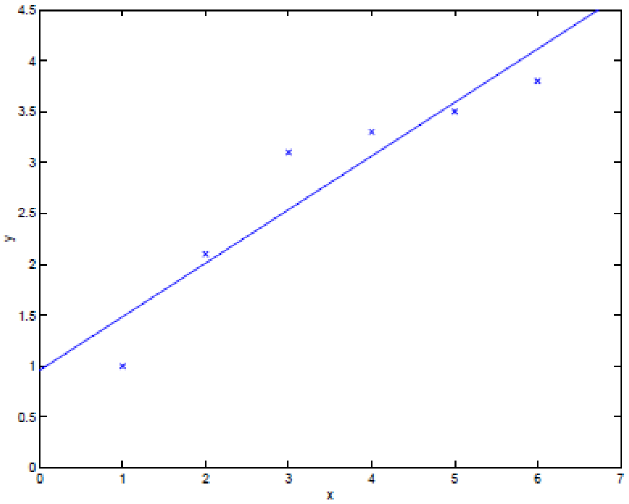
	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}, \qquad \boldsymbol{Y} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}, \qquad \boldsymbol{W} = \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix}$$

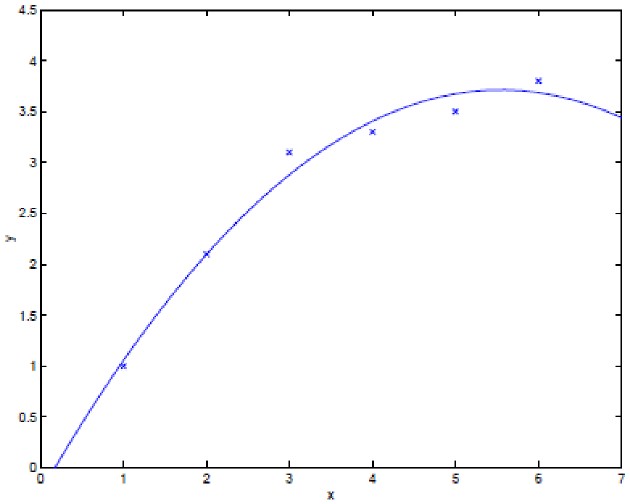
$$\boldsymbol{Xw} = \boldsymbol{y} \quad \Rightarrow \quad \boldsymbol{w} = \left(\boldsymbol{X^T X}\right)^{-\mathbf{1}} \boldsymbol{X^T y}$$

Univariate Polynomial Regression

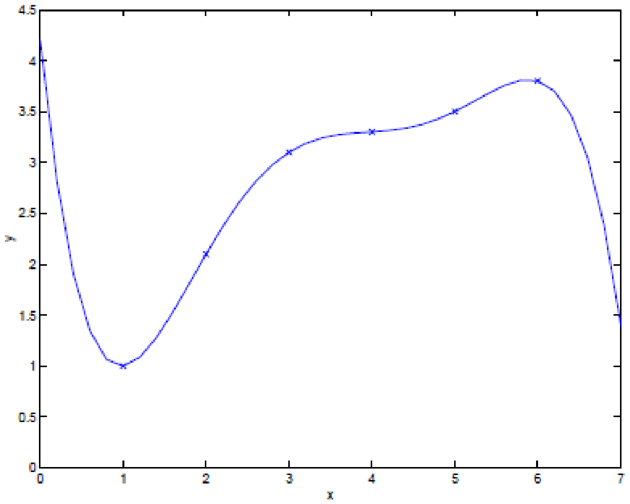
$$h\left(x^{(i)}\right) = W_1 x + W_2 x^2 + \cdots + W_n x^n + b$$



linear



Order 2



Oder 5

Logistic Regression

- Binary classification

$$0 \leq y \leq 1$$

- Linear Regression

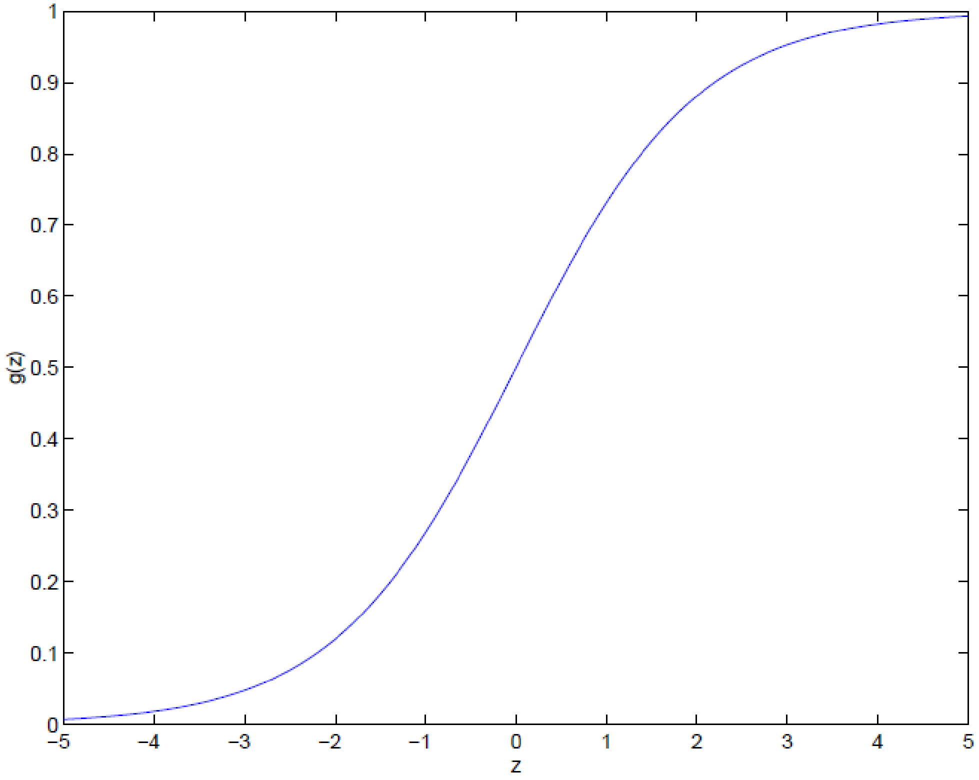
$$\boldsymbol{x} = [1 \; x_1 \; x_2]$$
$$\boldsymbol{w} = [b \; w_1 \; w_2]$$

- Logistic Hypothesis

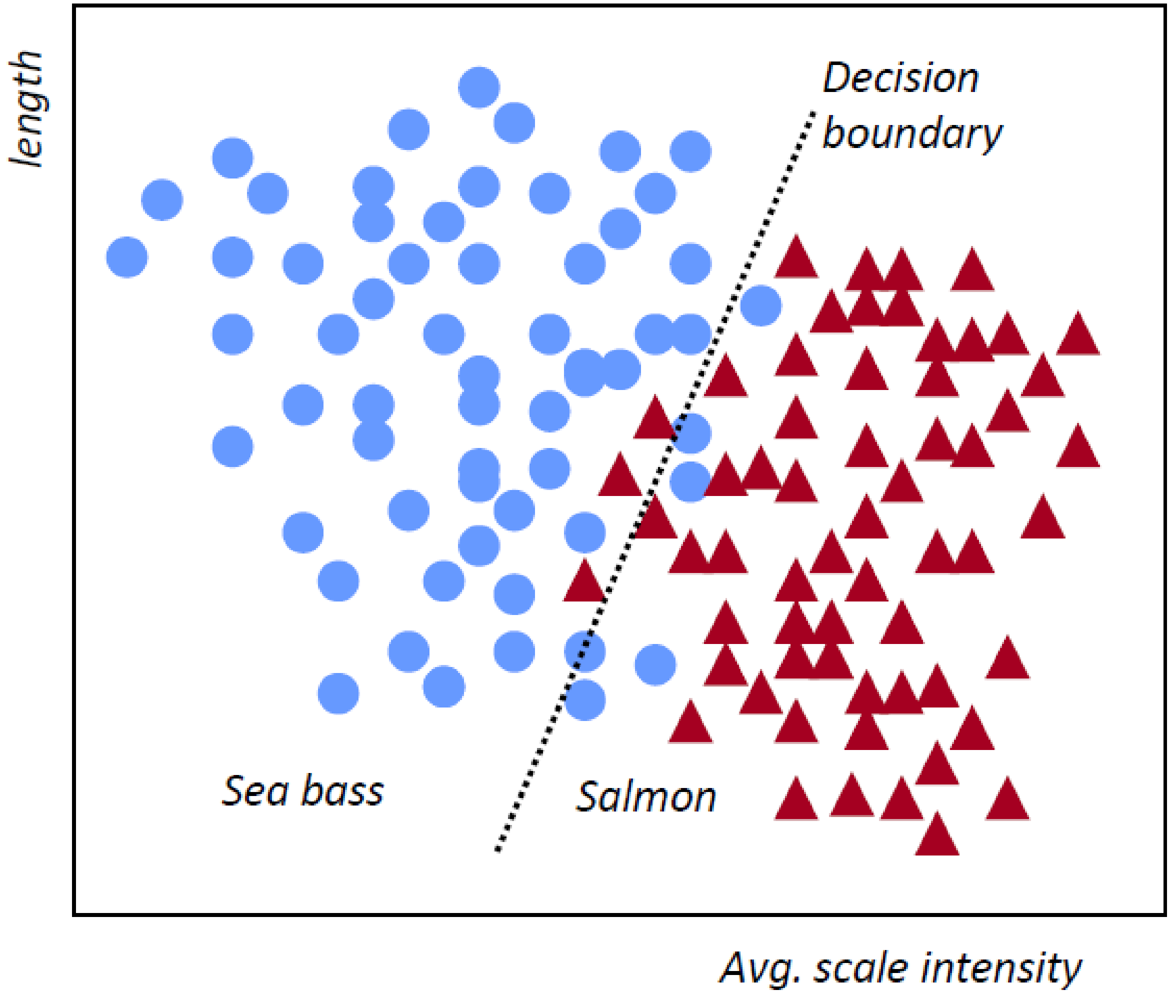
$$h(x) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$
$$h(x) = P(y = 1 | \boldsymbol{x}, \boldsymbol{w})$$
$$1 - h(x) = P(y = 0 | \boldsymbol{x}, \boldsymbol{w})$$
$$P(y|x; w) = \left(h(x)\right)^y \left(1 - h(x)\right)^{1-y}$$

Sigmoid

$$h(z) = \frac{1}{1 + \exp(-z)}$$



Example: Classify Salmon and Seabass



Likelihood

$$J(\boldsymbol{W}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(\boldsymbol{X}^{(i)}\right) - y^{(i)}\right)^2$$

$$L(\boldsymbol{W}, b) = \prod_{i=1}^{m} p\left(y^{(i)}|\boldsymbol{X}^{(i)}, \boldsymbol{W}, b\right) = \prod_{i=1}^{m}\left(h\left(\boldsymbol{X}^{(i)}\right)\right)^{y^{(i)}}\left(1 - h\left(\boldsymbol{X}^{(i)}\right)\right)^{1-y^{(i)}}$$

- Log likelihood:

$$\begin{aligned}\mathcal{L}(\boldsymbol{W}, b) &= \log\left(L(\boldsymbol{W}, b)\right) \\ &= \sum_{i=1}^{m} y^{(i)}\log\left(h\left(\boldsymbol{X}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h\left(\boldsymbol{X}^{(i)}\right)\right)\end{aligned}$$

Gradient Decent

$$w_j = w_j - \frac{\alpha}{m}\frac{\partial}{\partial w_j} l(w)$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial w}$$

Partial Gradient

$$\frac{\partial l}{\partial h} = \frac{\partial}{\partial h}\left(\sum y \log h + (1-y)\log(1-h)\right) = \sum y \frac{\partial}{\partial h}\log h + (1-y)\frac{\partial}{\partial h}\log(1-h)$$

$$\frac{d}{dh}\log_{\mathrm{a}} h = \frac{1}{h \ln a} = \frac{1}{h}\Big|_{a=e}$$

$$\frac{d}{dh}\ln 1 - h = \frac{d}{du}\ln u \frac{du}{dh} = -\frac{1}{1-h}$$

$$\frac{\partial l}{\partial h} = \sum \frac{y}{h} - \frac{(1-y)}{1-h}$$

Partial Gradient

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

$$h(z) = \frac{1}{u}$$

$$\frac{\partial h}{\partial z} = \frac{\partial h}{\partial u}\frac{\partial u}{\partial z} = \frac{1}{u^2}\frac{du}{dz} = \frac{-e^{-z}}{(1+e^{-x})^2} = \frac{1}{1+e^{-z}}\,\frac{-e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right)$$

$$\frac{\partial h}{\partial z} = h\,(1-h)$$

Gradient

$$z = w^T x$$

$$\frac{\partial z}{\partial w} = x$$

$$\frac{\partial l}{\partial w} = \sum \left(\frac{y}{h} + \frac{(1-y)}{1-h} \right) h(1-h)x = \sum y(1-h)x - (1-y)hx = (y-h)x$$

Gradient Descent

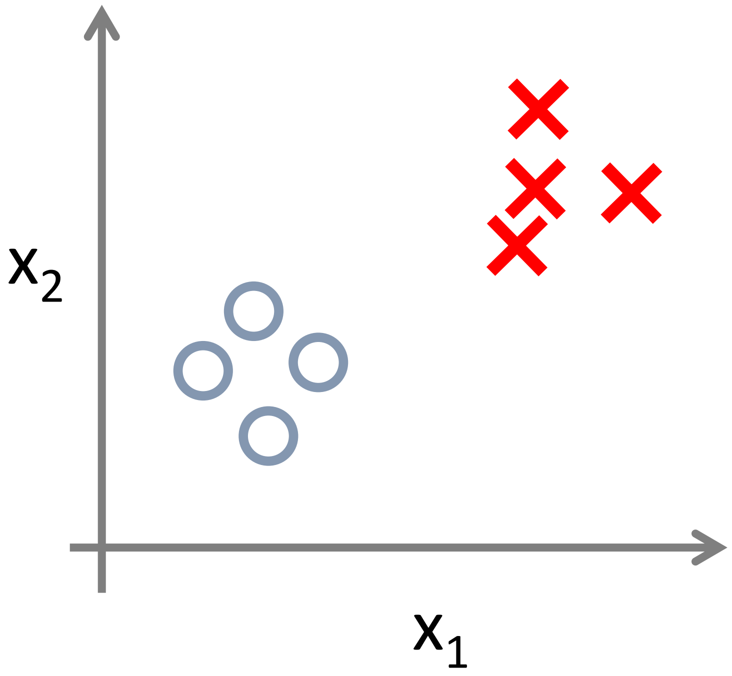
$$w_j = w_j - \frac{\alpha}{m}\frac{\partial}{\partial w_j} l(\boldsymbol{w})$$

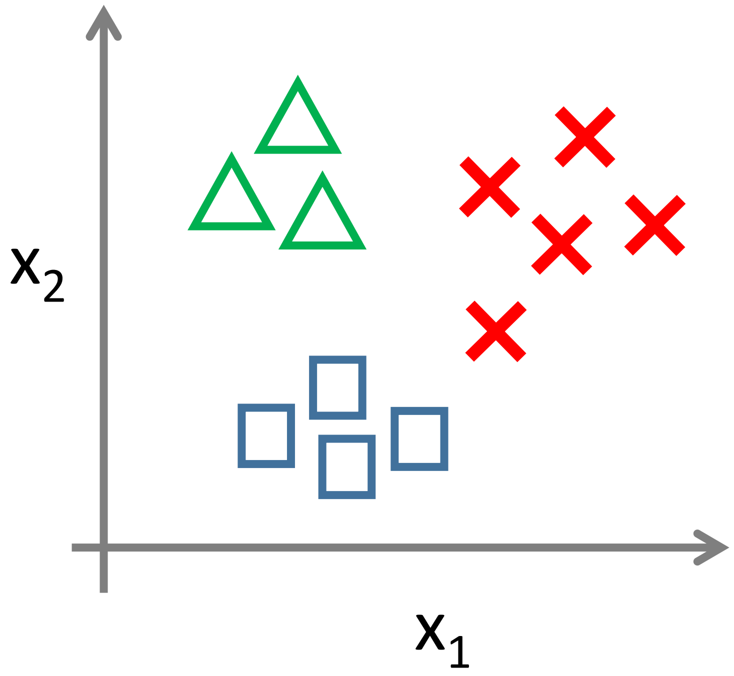
$$w_j = w_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left(y^{(i)} - h\left(w_j x_j^{(i)}\right)\right)x_j^{(i)}$$

- Vectorized:

$$\boldsymbol{W} = \boldsymbol{W} - \frac{\alpha}{m}\left(\boldsymbol{X}^{\boldsymbol{T}}\left(\boldsymbol{Y} - h(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{X})\right)\right)$$

Multiclass

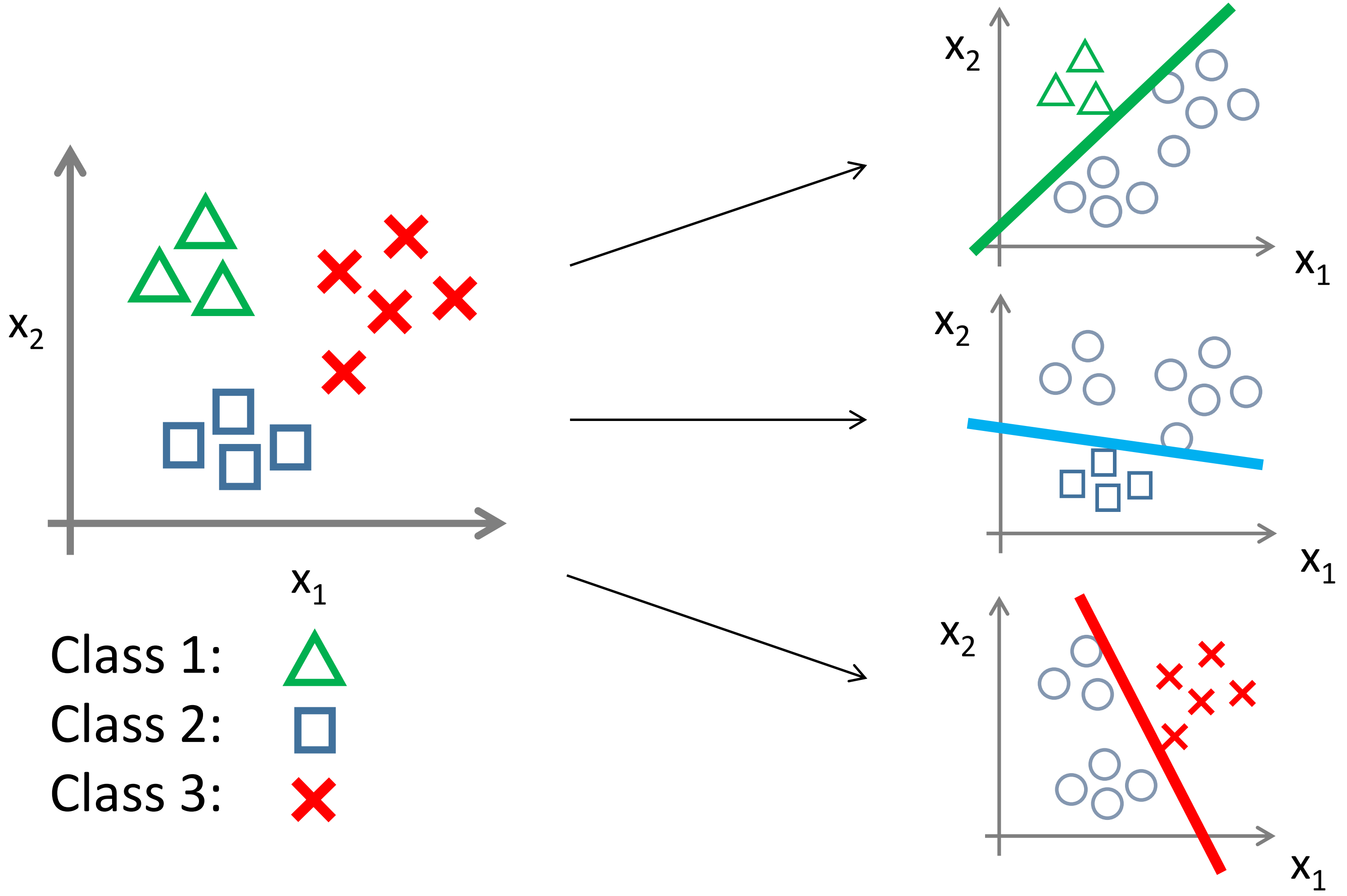




- Can treat as a k-binary
  - Good for classes that are non mutually exclusive
- 1-hot encoding

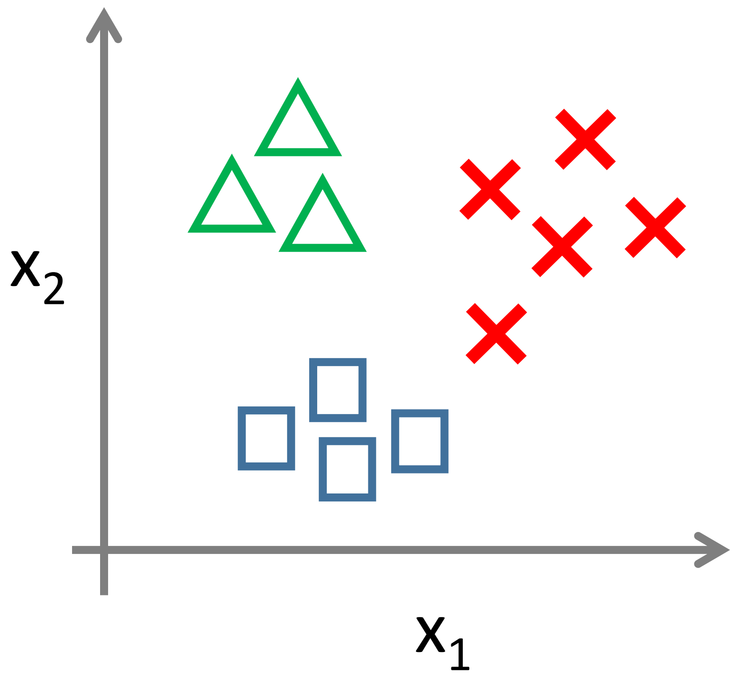
K-binary

- Treat 1 class as class ‘1’ and all others as class ‘0’
- Repeat for all classes
- Must train k times
- Works well when classes are independent



1-hot encoded Approach for Sigmoid

- Trained only once
- More effective
  - Balanced training set
- Reformat the GT into a vector
  - K-binary has only 1 output: Y = 0 or 1
  - 1-hot encoded output = a vector of k binary values
- E.g. k = 3 classes
  - Y is a vector of 3 values
  - Class 1: Y = [1 0 0]
  - Class 2: Y = [0 1 0]
  - Class 3: Y = [0 0 1]
  - 3 sigmoids sharing inputs



Softmax Regression

- 1-hot encoded approach
- New activation function

$$y^{(i)} \in \{1,2,3,\ldots,M\}$$

$$h\left(x^{(i)}\right) = p\left(y^{(i)} = n | x^{(i)}; \theta\right) = \frac{\exp\left({w_n}^T x^{(i)}\right)}{\sum_{j=1}^{M} \exp\left({w_j}^T x^{(i)}\right)} = \frac{class_n\ output}{Sum\ of\ All\ class\ outputs}$$

- For the class output to be 1
  - Must increase output of class, while suppressing other outputs

Likelihood

$$J = \prod_{i=1}^{m} \prod_{j=1}^{M} 1\{y_j^i = n\}\, h_j$$

$$1\{y_j^i = n\} = \begin{cases} 1 & y_j^i = n \\ 0 & y_j^i \neq n \end{cases}$$

$$L = \log J = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{M} 1\{y_j^i = n\} \log(h_j) \; = \frac{1}{m} \sum_{i=1}^{m} L_j$$

$$L_j = -1\{y_j^i = n\} \log h_j$$

Gradient

$$\frac{\partial L_j}{\partial w_j} = -1\{y_j^i = n\}\frac{\partial}{\partial w_j}\log h_i = -1\{y_j^i = n\}\frac{1}{h_j}\frac{\partial h_j}{\partial w_j}$$

$$\frac{\partial h_j}{\partial w_j} = \begin{cases} h_n\left(1 - h_j\right) & j = n \\ -h_j h_n & j \neq n \end{cases}$$

$$\frac{\partial L}{\partial w_j} = \left(h_j - \{y_j^i = n\}\right)x$$

Stable Softmax

- Exponential can be numerically unstable
- use a stabilizing coefficient

$$h(z_i) = \frac{ce^{z_i}}{\sum_j ce^{z_j}} = \frac{e^{z_i+D}}{\sum_j e^{z_j+D}}$$

  - $D = -\max\{z_j \ \forall j\}$

Gradient Descent