Applied Artificial Intelligence

Session 11: Linear Models II

Fall 2018

NC State University
Lecturer: Dr. Behnam Kia

Course Website: https://appliedai.wordpress.ncsu.edu/

Regression Problem and Regression Models

- In Regression problems the task is to approximate a mapping function (*h*) from input variables (*x*) to continuous output variables, (also called real-valued outputs) , (also called numeric outputs).

- In this session we work on linear regression models:

$$\hat{Y} = X.W$$

Training data: (x_1,y_1), (x_2,y_2), (x_3,y_3),…, (x_m,y_m)

House Prices

Goal: Come Up with a Linear Regression Model that Explains the Training Data, and Predicts the Price of Other Houses as Well.

(x ft²,	$ y K)
(3883 ft²,	$432K)
(1668 ft²,	$218K)
(3577 ft²,	$366K)
(1668 ft²,	$218K)
(765 ft²,	$123K)
(3822 ft²,	$493K)

.
.
.

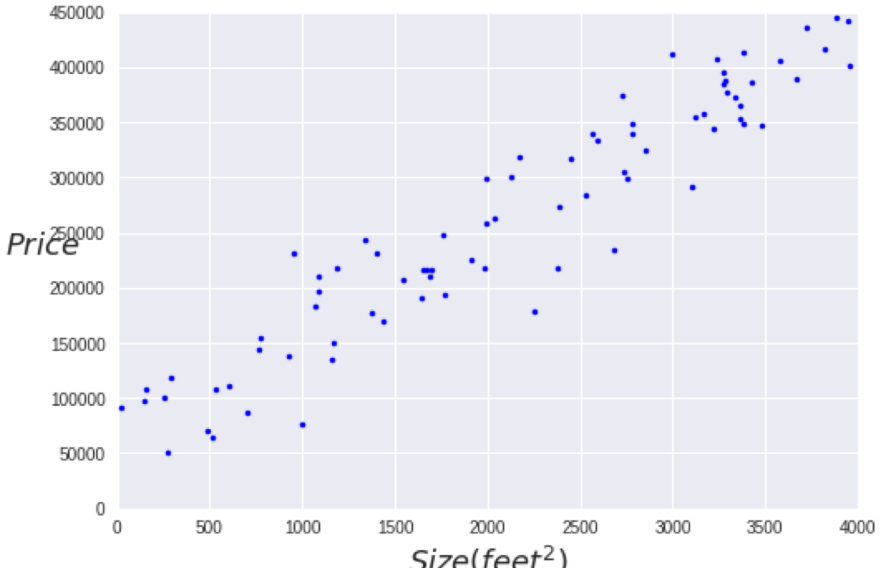
House Prices

Goal: Come Up with a Linear Regression Model that Explains the Training Data, and Predicts the Price of Other Houses as Well.

Input: x (the size of house)

Target: y (the price of the house), y$\in R$

Linear Model: $\hat{y} = w_0 + w_1 x_1$



House Prices
Goal: Obtaining a Linear Regression Model

Input: x (the size of house)

Target: y (the price of the house), y$\in R$

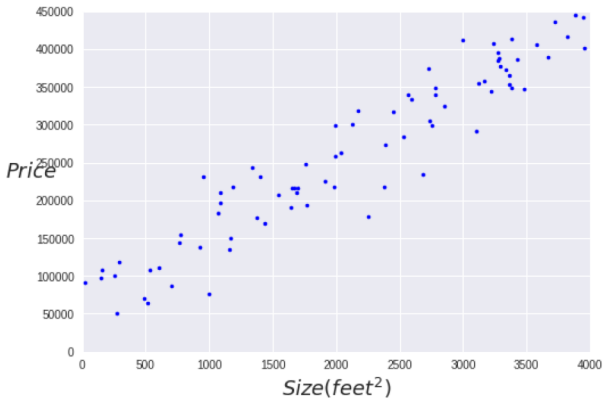
Linear Model: $\hat{y} = w_0 x_0 + w_1 x_1$

$$\hat{y} = X.W$$

where:

$$x_p = [1, x_1]$$

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$



House Prices
Goal: Obtaining a Linear Regression Model

Input: x (the size of house)

Target: y (the price of the house), y$\in R$

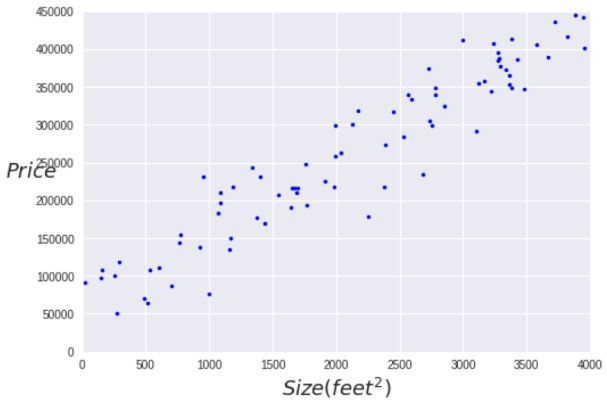
Linear Model: $\hat{y} = w_0 x_0 + w_1 x_1$

$$\hat{y} = X.W$$

where:

$$x_p = [1, x_1]$$

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$?



House Prices
Cost Function: How well or poorly a model (Hypothesis) explains the training data

(x ft²,	$ y K)
(3883 ft²,	$432K)
(1668 ft²,	$218K)
(3577 ft²,	$366K)
(765 ft²,	$123K)
(3822 ft²,	$493K)

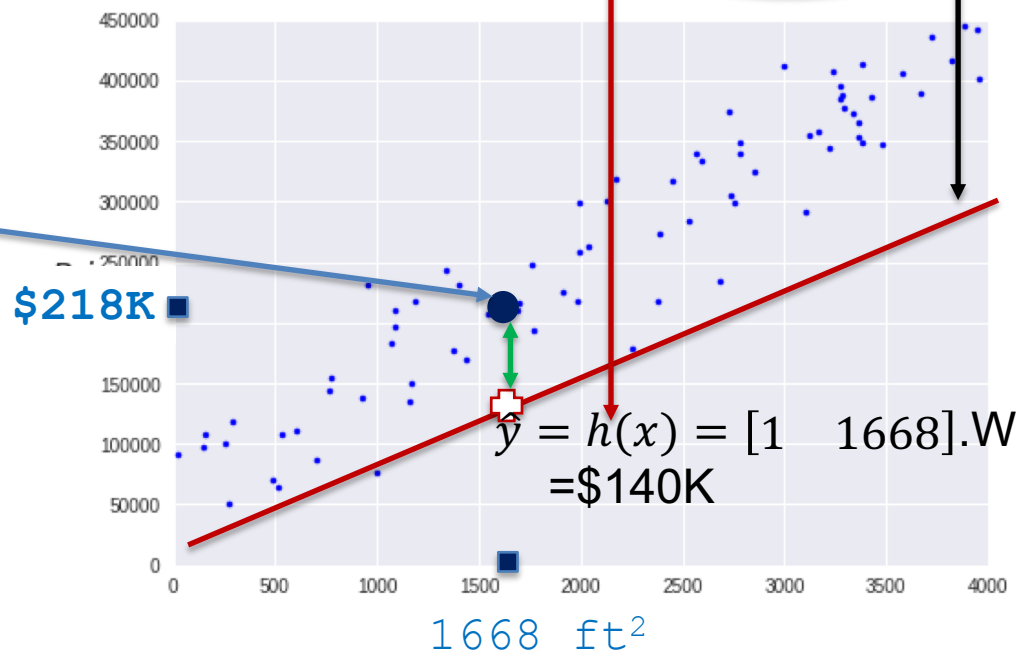
(1668 ft², $218K)

Cost Function for entire training set:

$$J(W)=\sum_i (h(\hat{x}^i) - y^i)^2$$

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

?



House Prices
Mean Squared Error
as Cost Function

(x ft², $ y K)
(3883 ft², $432K)
(1668 ft², $218K)
(3577 ft², $366K)
(765 ft², $123K)
(3822 ft², $493K)
(1668 ft², $218K)

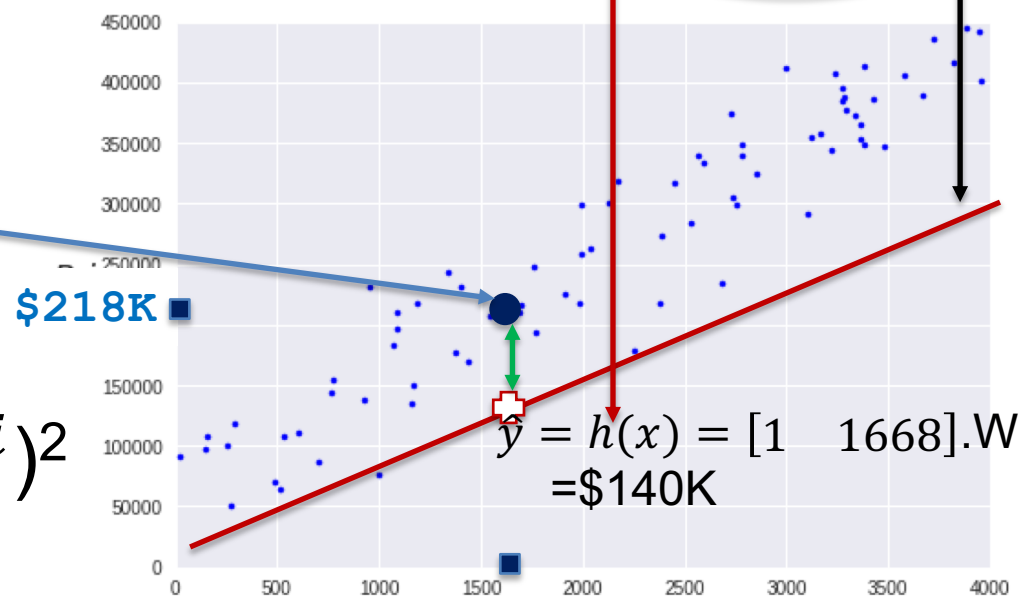
Cost Function for entire training set:

$$J(W)=\frac{1}{m}\sum_i (h(\hat{x}^i)-y^i)^2$$

Size of training data

$$W=\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

?



How to find the minimum point of cost function?

$$W_{optimal} = \underset{W}{\operatorname{argmin}}\ \mathrm{J(W)}$$

?

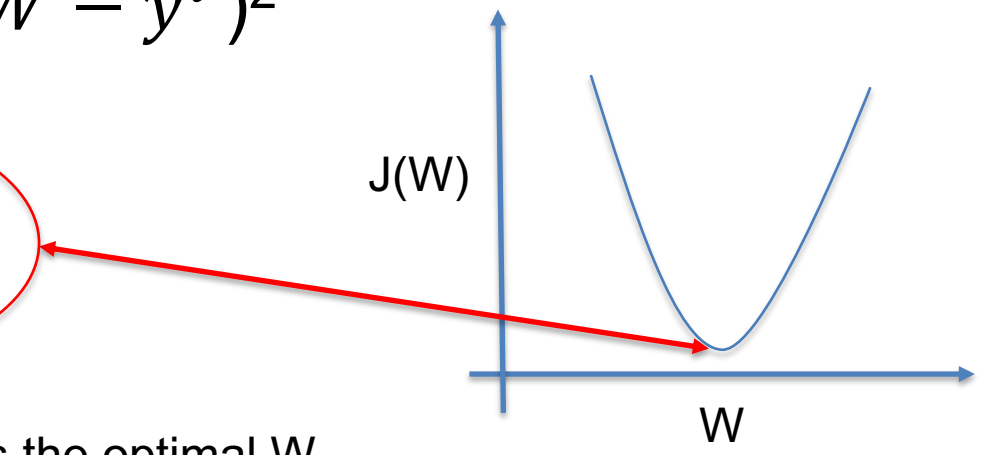
$$\mathrm{J(W)} = \frac{1}{2m}\sum_i (x_p^i . W - y^i)^2$$

How to find the minimum point of cost function? Analytical Method

$$W_{optimal} = \underset{W}{\operatorname{argmin}}\ \mathrm{J(W)}$$

$$\mathrm{J(W)} = \frac{1}{2m}\sum_i (x_p^i . W - y^i)^2$$

$$\frac{\partial J(W)}{\partial W} = 0$$



Solving this will give us the optimal W

How to find the minimum point of cost function? Analytical Method

$$W_{optimal} = \underset{W}{\operatorname{argmin}}\ \mathrm{J(W)}$$

$$\mathrm{J(W)} = \frac{1}{2m}\sum_i (x_p^i . W - y^i)^2$$

(n,1+1)

$$X_p = \begin{bmatrix} 1 & 3883 \\ 1 & 1668 \\ 1 & 3577 \\ \vdots & \vdots \\ 1 & 1668 \end{bmatrix}$$

x_p^1 x_p^n

$$\frac{\partial J(W)}{\partial W} = 0 \Rightarrow \mathrm{W_{optimal}} = (X_p^T . X_p)^{-1} . X_p^T . \mathrm{y}$$

(n,1)

$$y = \begin{bmatrix} 432K \\ 218K \\ 366K \\ \vdots \\ 218K \end{bmatrix}$$

y^1 y^n

See proof at: The Elements of Statistical Learning, T. Hastie, R. Tibshirani, J. Friedman, Page 12, and pages 44-45

The Problems of the Analytical Method

$$W_{optimal} = \underset{W}{\operatorname{argmin}}\ \mathrm{J(W)}$$

$$\mathrm{J(W)} = \frac{1}{2n}\sum_i (W^T . x_p^i - y^i)^2$$

(n,1+1)

$$X_p = \begin{bmatrix} 1 & 3883 \\ 1 & 1668 \\ 1 & 3577 \\ \vdots & \vdots \\ 1 & 1668 \end{bmatrix}$$

x_p^1 x_p^n

$$\frac{\partial J(W)}{\partial W} = 0 \Rightarrow \mathrm{W_{optimal}} = (X_p^T . X_p)^{-1} . X_p^T . \mathrm{y}$$

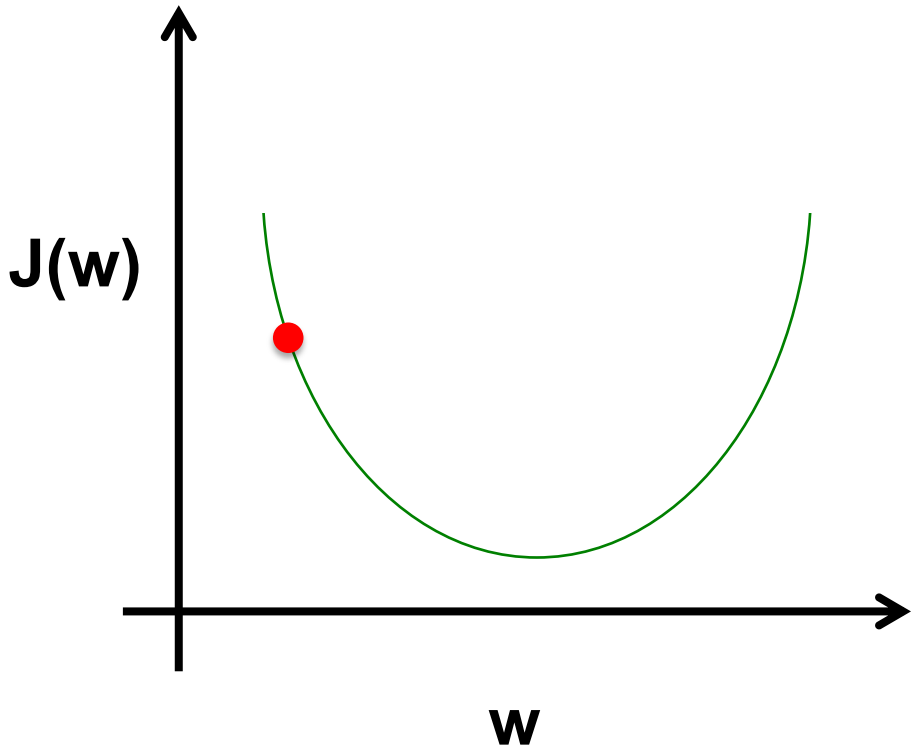
(n,1)

$$y = \begin{bmatrix} 432K \\ 218K \\ 366K \\ \vdots \\ 218K \end{bmatrix}$$

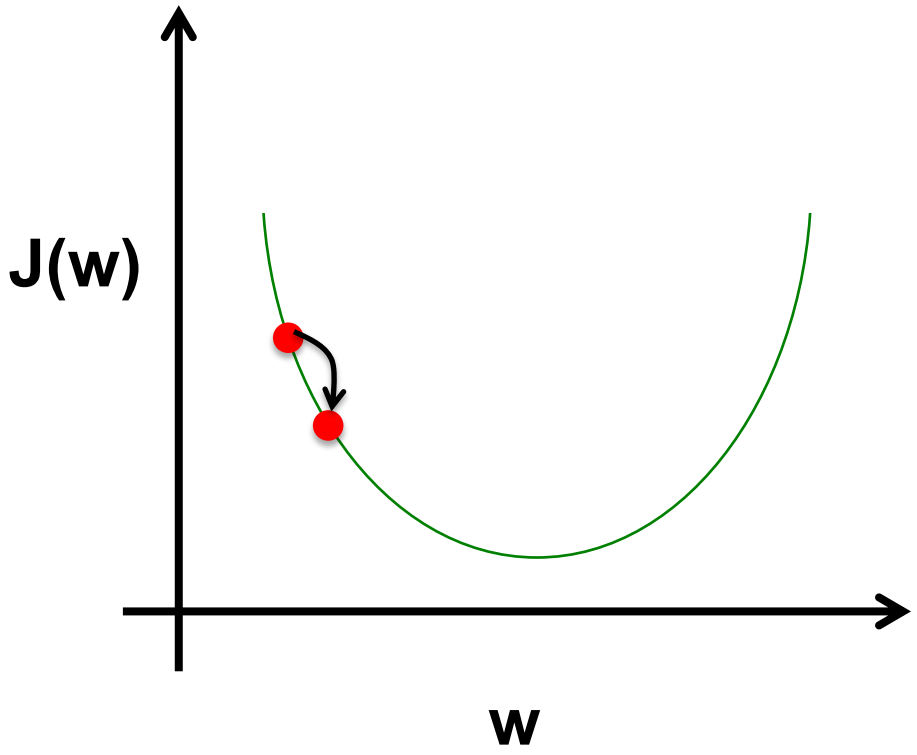
y^1 y^n

See proof at: The Elements of Statistical Learning, T. Hastie, R. Tibshirani, J. Friedman, Page 12, and pages 44-45

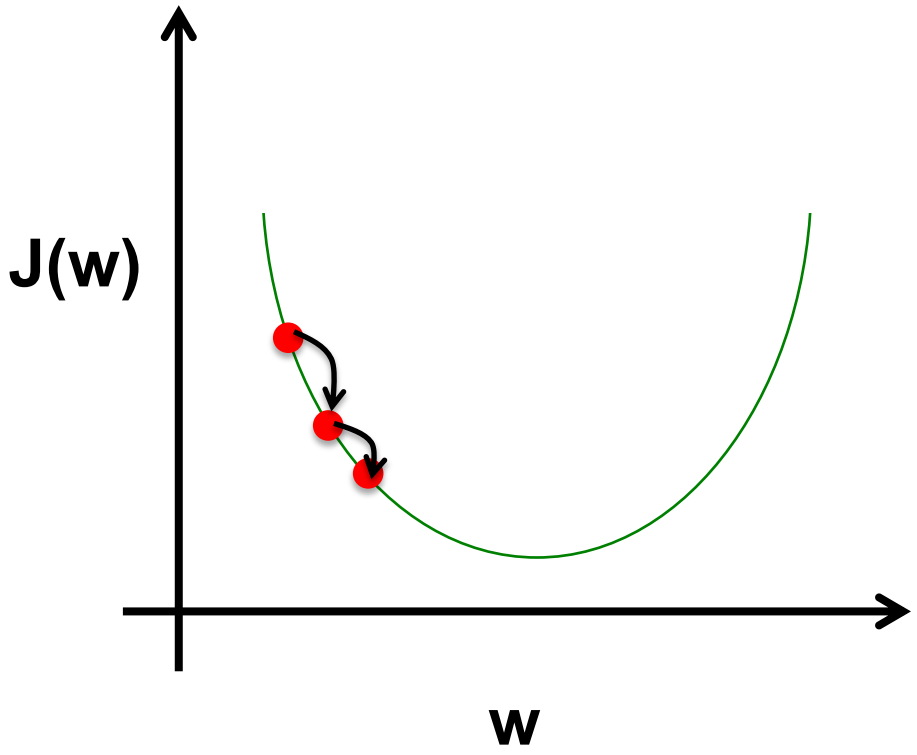
Gradient Descent



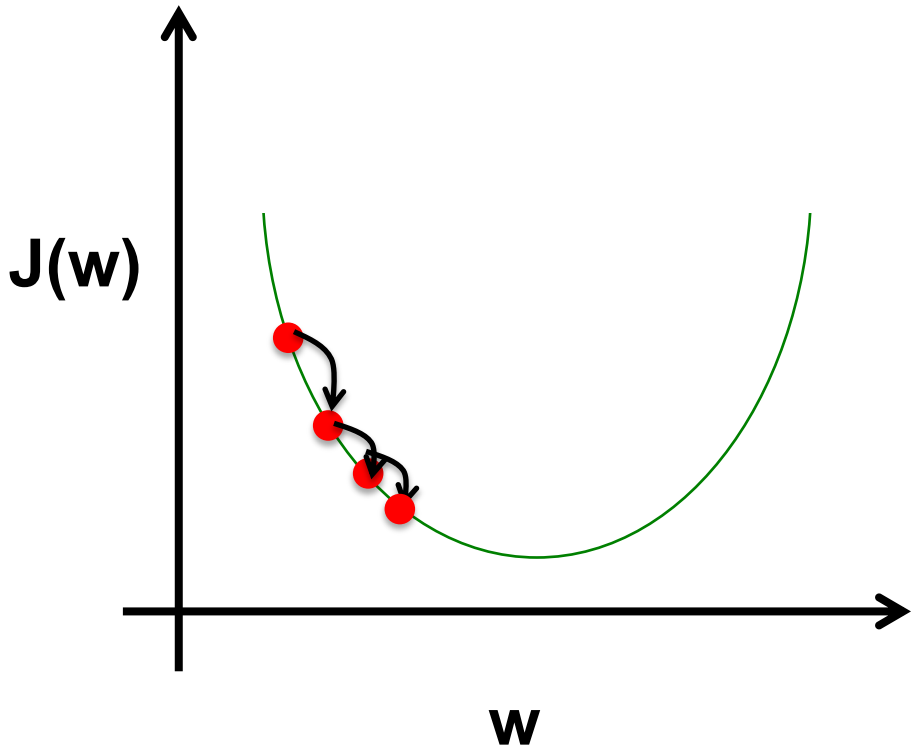
Gradient Descent



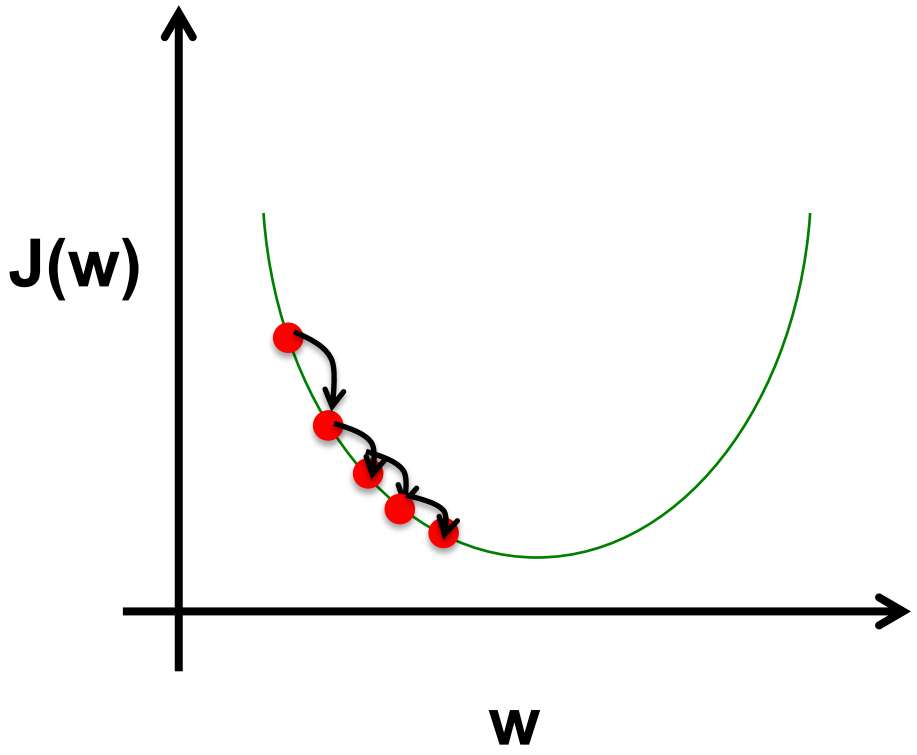
Gradient Descent



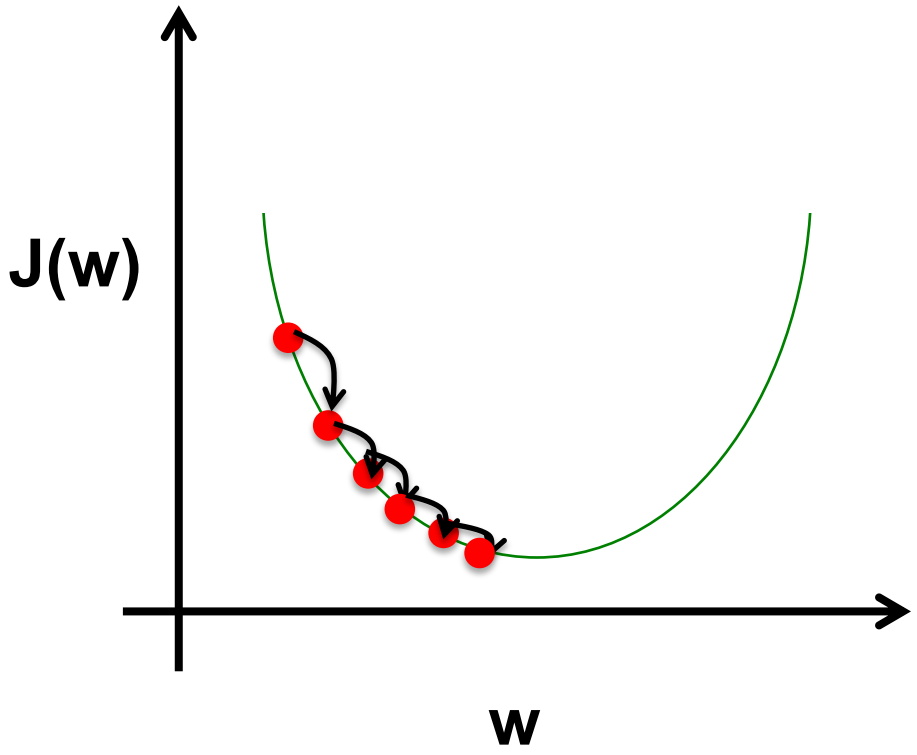
Gradient Descent



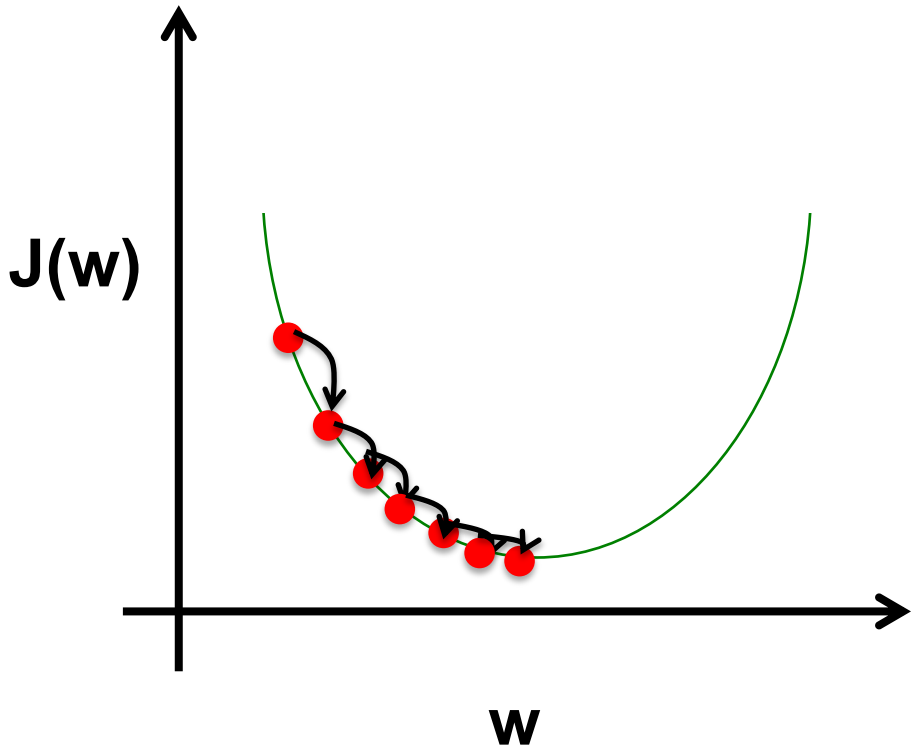
Gradient Descent



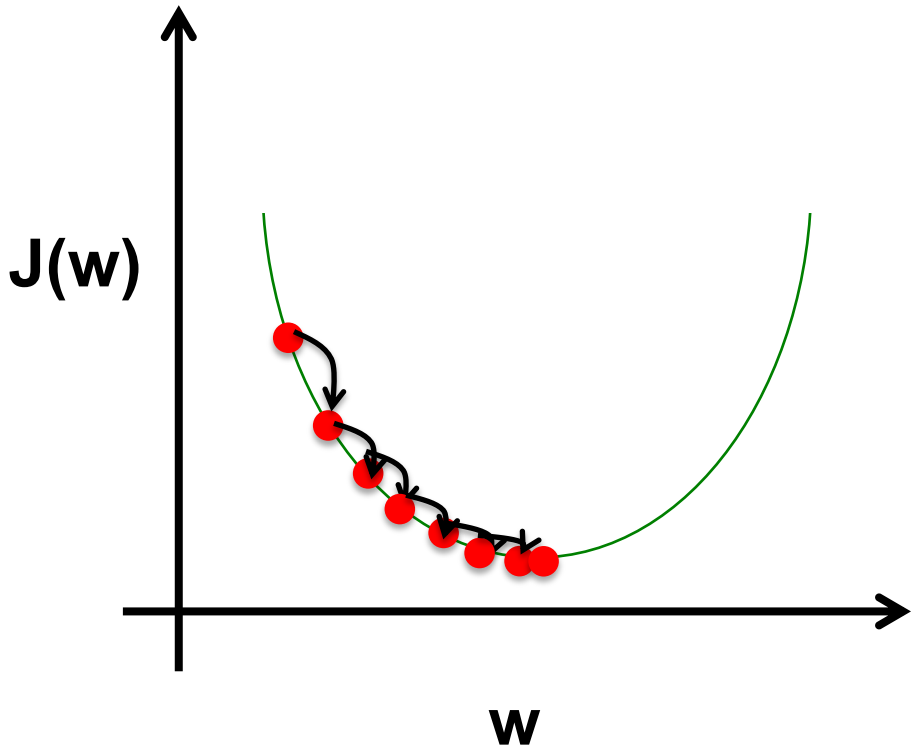
Gradient Descent



Gradient Descent



Gradient Descent

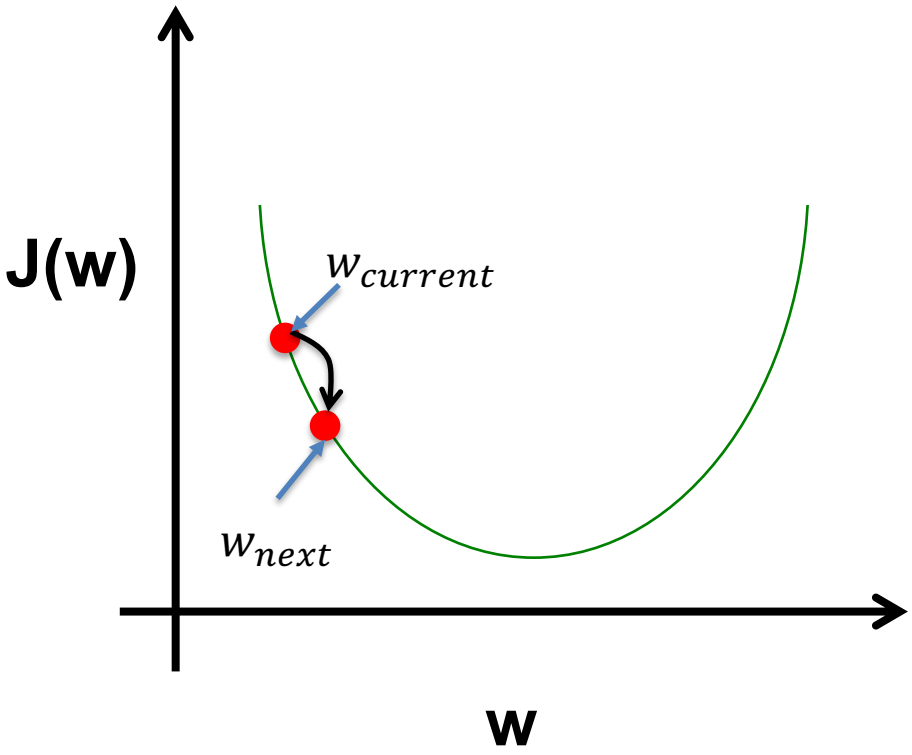


How to Formulate Gradient Descent?

For simplicity we start from 1-D *w*, then will extend the concepts to 2-D *w*'s.

$$w_{next} = w_{current} - \alpha \frac{\partial J(w)}{\partial w}$$

α is learning rate.



- Codes

How to Formulate Gradient Descent for Linear Regression?

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

α is learning rate.

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

$$J(W) = \frac{1}{2m} \sum_i (x_p^i . W - y^i)^2$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

How to Formulate Gradient Descent for Linear Regression?

$$w_{next} = w_{current} - \alpha \frac{\partial J(w)}{\partial w}$$

α is learning rate.

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

$$J(W) = \frac{1}{2m} \sum_i (x_p^i . W - y^i)^2$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

Example: The house price problem

$$\frac{\partial J(W)}{\partial w_0} = \frac{1}{n} \sum_i 1 \times ([1 \quad x^i] . \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - y^i)$$

$$\frac{\partial J(W)}{\partial w_0} = \frac{1}{n} \sum_i x^i \times ([1 \quad x^i] . \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - y^i)$$

$$w_{0_{next}} = w_{0_{current}} - \alpha \frac{\partial J(w)}{\partial w_0}$$

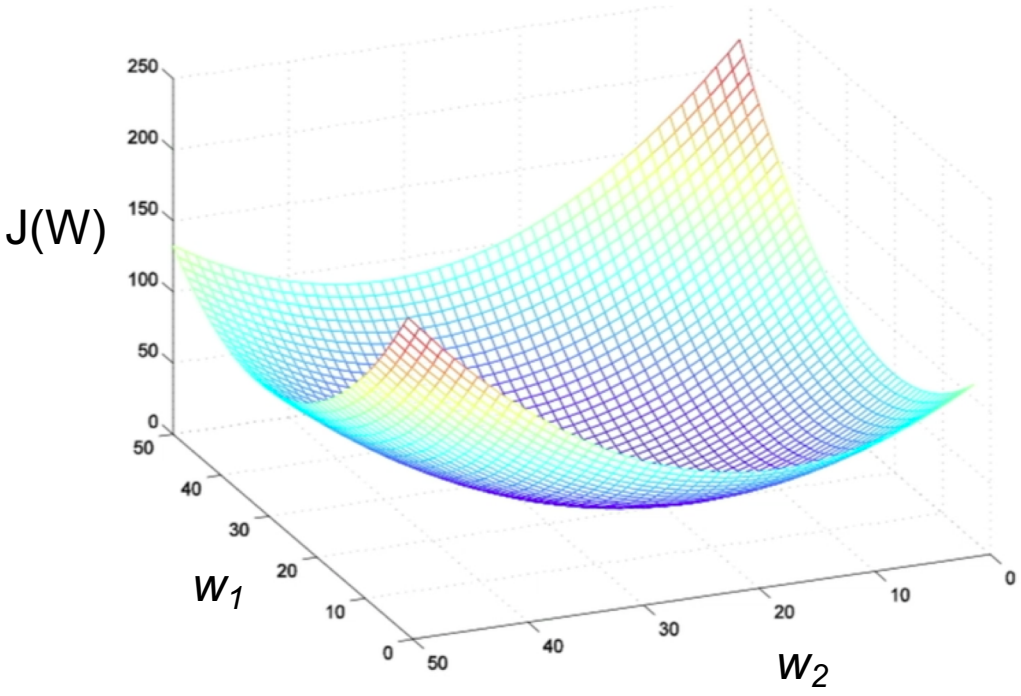
$$w_{1_{next}} = w_{1_{current}} - \alpha \frac{\partial J(w)}{\partial w_1}$$

Vectorized Gradient Descent in 2D

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

$$\nabla_W J(W) = \frac{1}{m} X_p^T . (X_p . W - y)$$



"Batch" Gradient Descent

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

Loops over the entire training set.

"Stochastic" Gradient Descent

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

"Batch" Gradient Descent

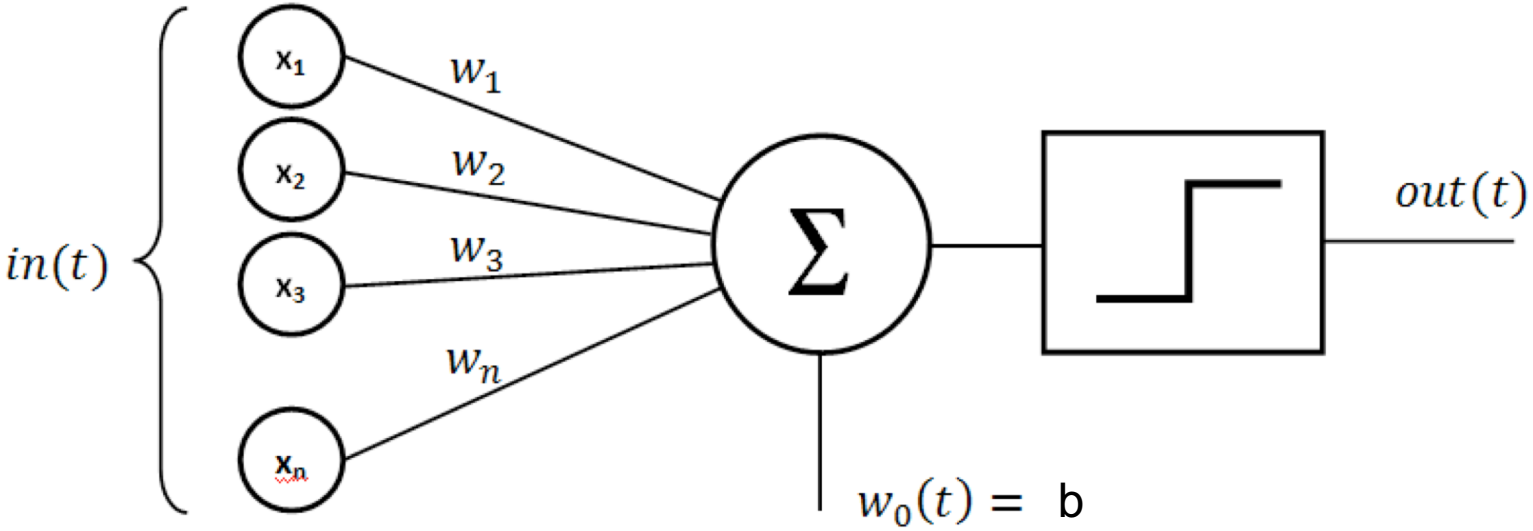
$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

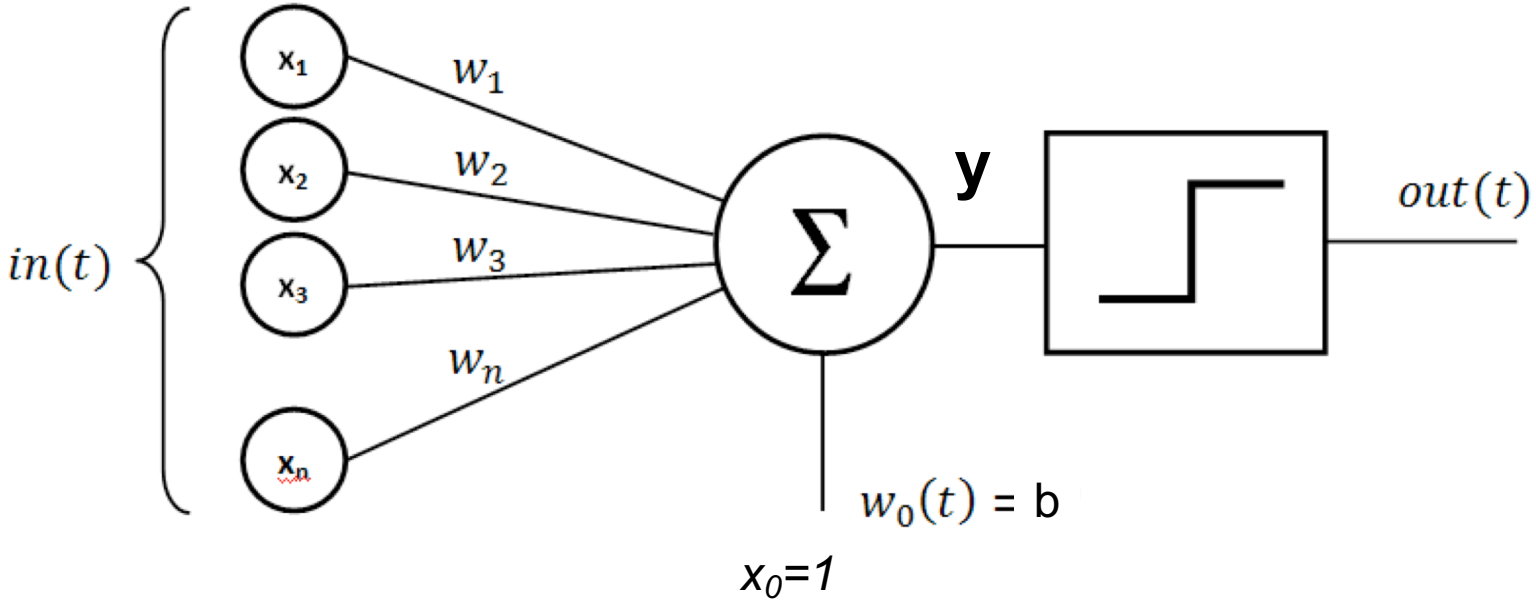
$$\frac{\partial J(W)}{\partial w_j} = x_{pj}^i (x_p^i . W - y^i)$$

i is randomly selected.

- codes

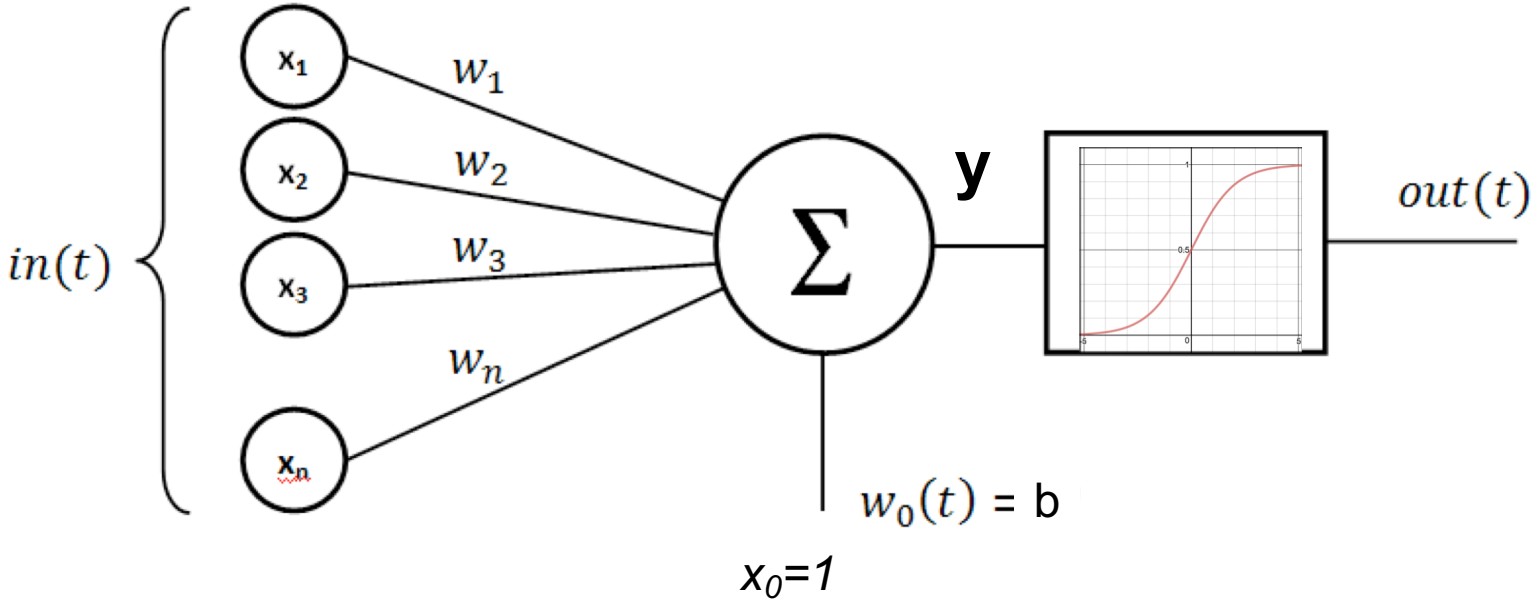
Perceptron: A Computational Neuron Model





$$y = b + w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n$$
$$= w_0\, x_0 + w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n$$
$$= X_p . W$$

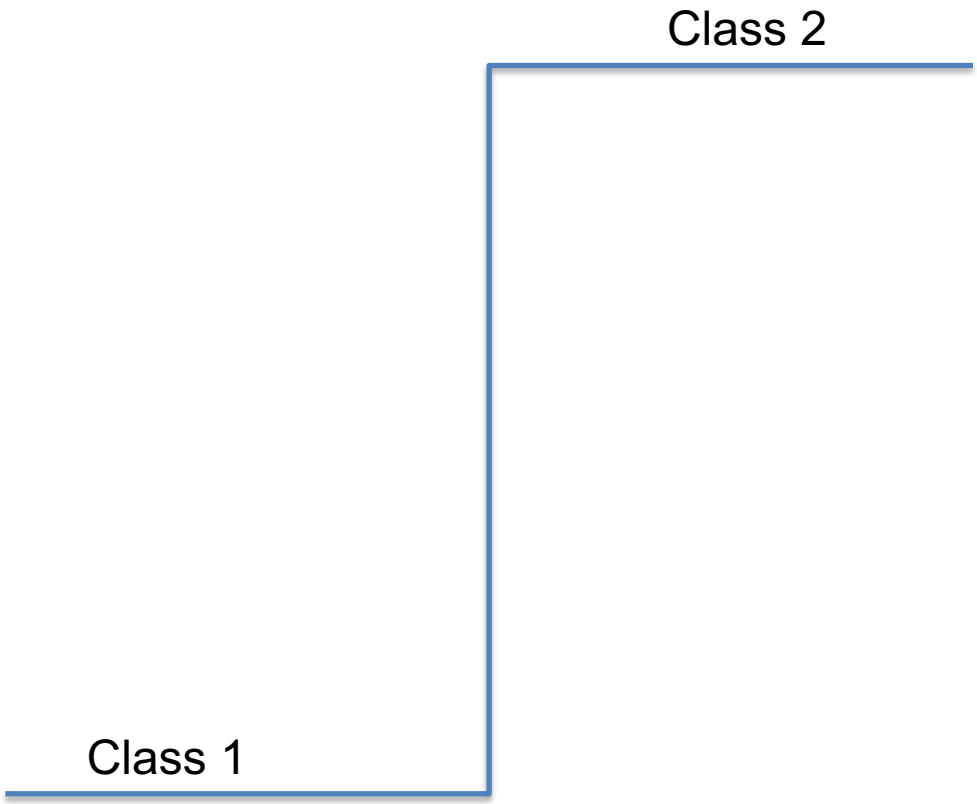
$$output = \begin{cases} 1 & if \quad X_p . W > 0 \\ 0 & otherwise \end{cases}$$

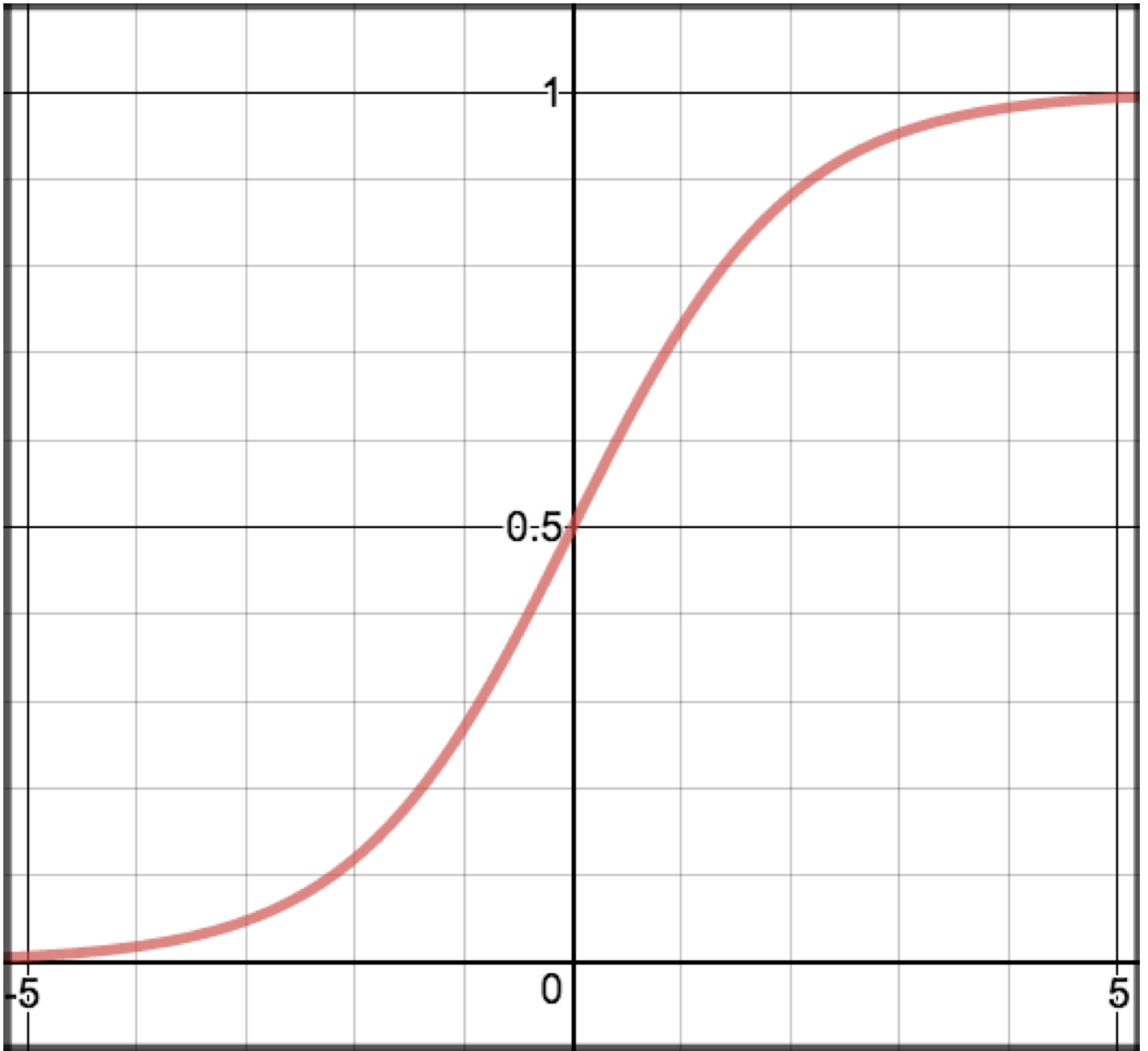


$$y = b + w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n$$
$$= w_0\, x_0 + w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n$$
$$= X_p . W$$

$$output = s(y)$$

$$S(y) = \frac{1}{1 + e^{-y}}$$


Class 2
Class 1


1
0.5
-5
0
5
Class 2
Class 1

Cost Function

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log(x_p^i . W) + (1 - y^i)\log(1 - x_p^i . W)$$

Cost Function

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log(x_p^i . W) + (1 - y^i)\log(1 - x_p^i . W)]$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m}\sum_i x_{pj}^i(\sigma(x_p^i . W) - y^i)$$

Homework

- As homework 2, we used Genetic Algorithm to Train a Classifier.
- Let's train another linear classifier, but this time using:
  - Batch Gradient Descent.
  - Stochastic Gradient Descent.
  - Mini-Batch Gradient Descent.
- Use the same dataset as homework 1, but now:
  - We should use sigmoid nonlinear function instead of the step function.
  - Use cross-entropy as a measure of error instead of counting the number of misclassification.