

# Foundations of Machine Learning

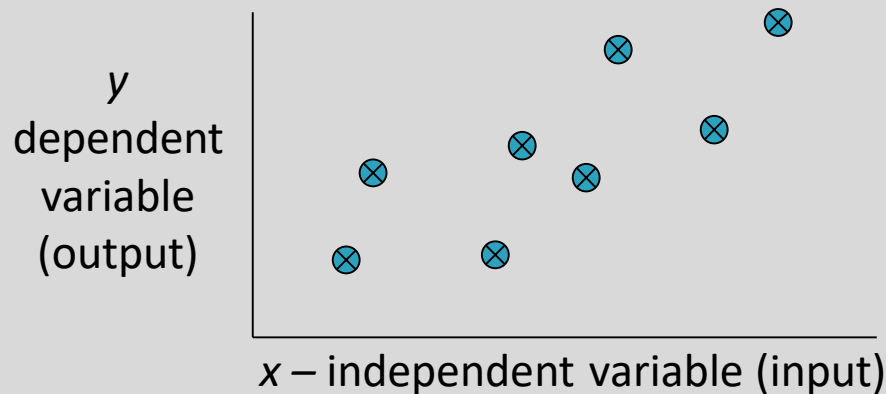
## Module 2: Linear Regression and Decision Tree

### Part A: Linear Regression

Sudeshna Sarkar  
IIT Kharagpur

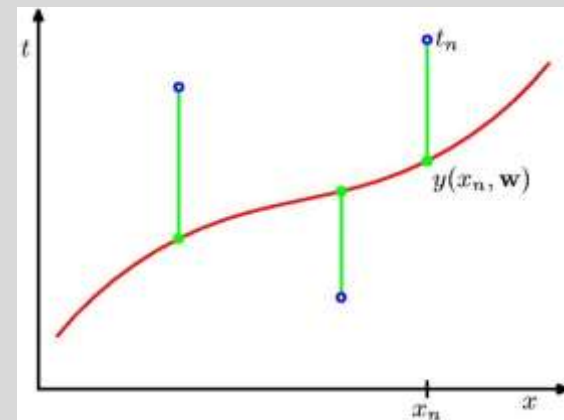
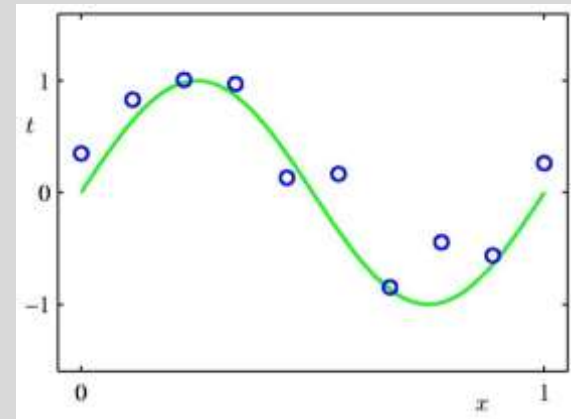
# Regression

- In regression the output is continuous
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points



# A Simple Example: Fitting a Polynomial

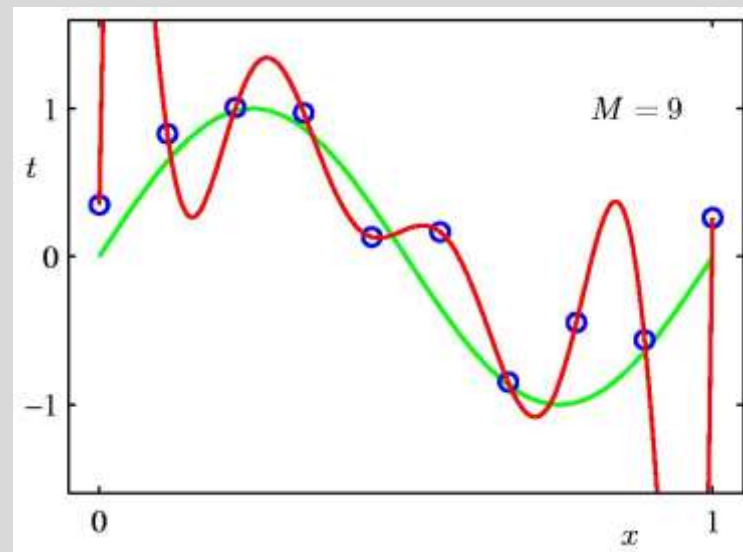
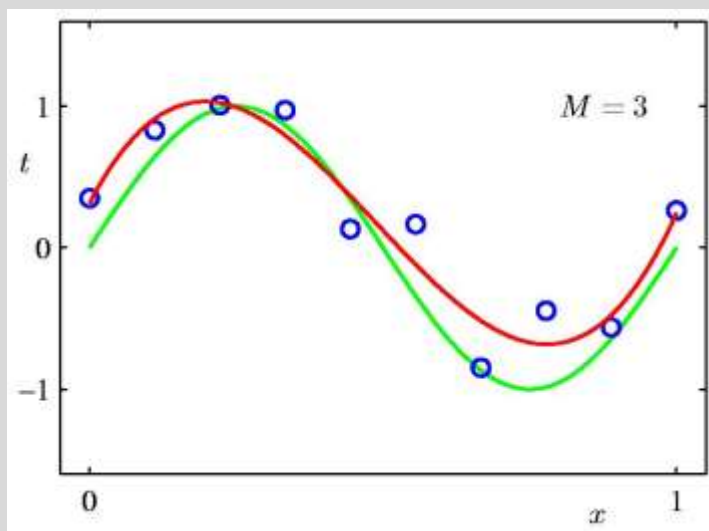
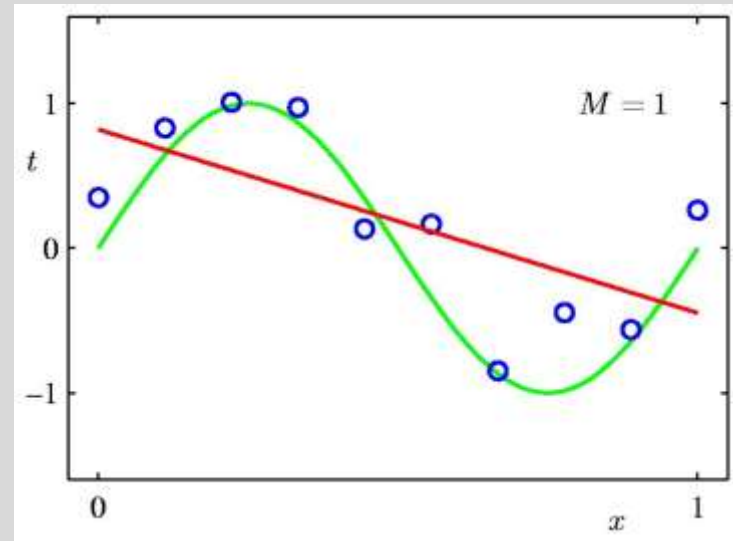
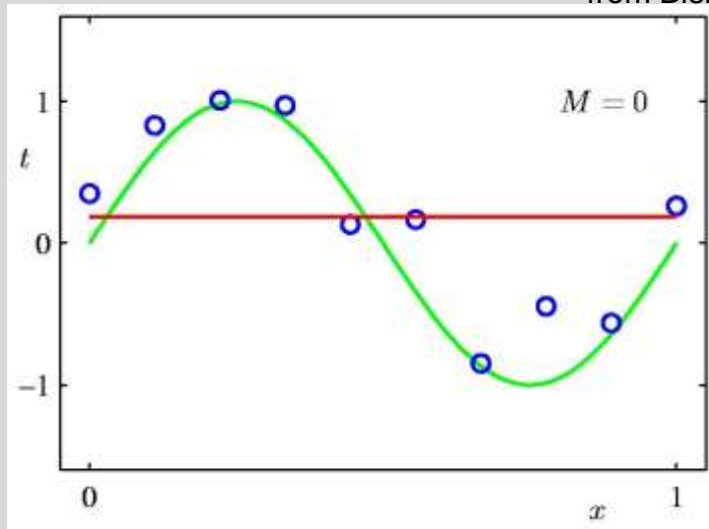
- The green curve is the true function (which is not a polynomial)
- We may use a loss function that measures the squared error in the prediction of  $y(x)$  from  $x$ .



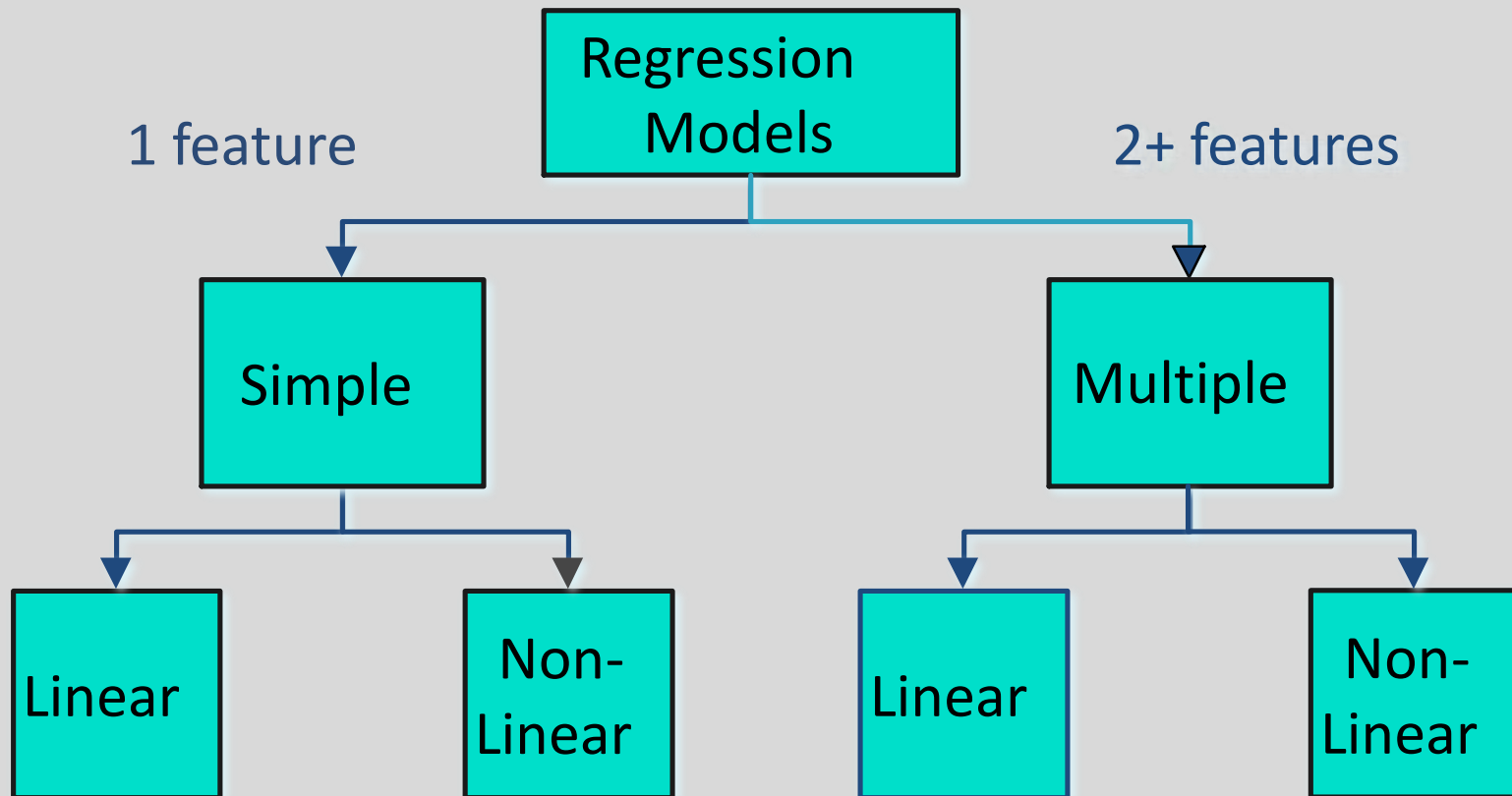
from Bishop's book on Machine Learning

# Some fits to the data: which is best?

from Bishop

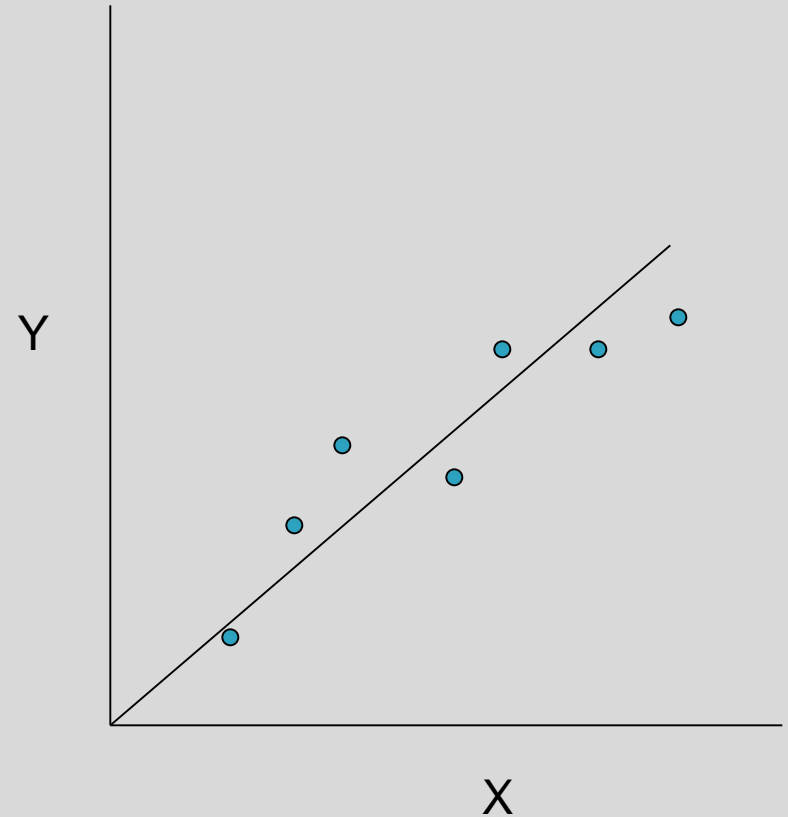


# Types of Regression Models

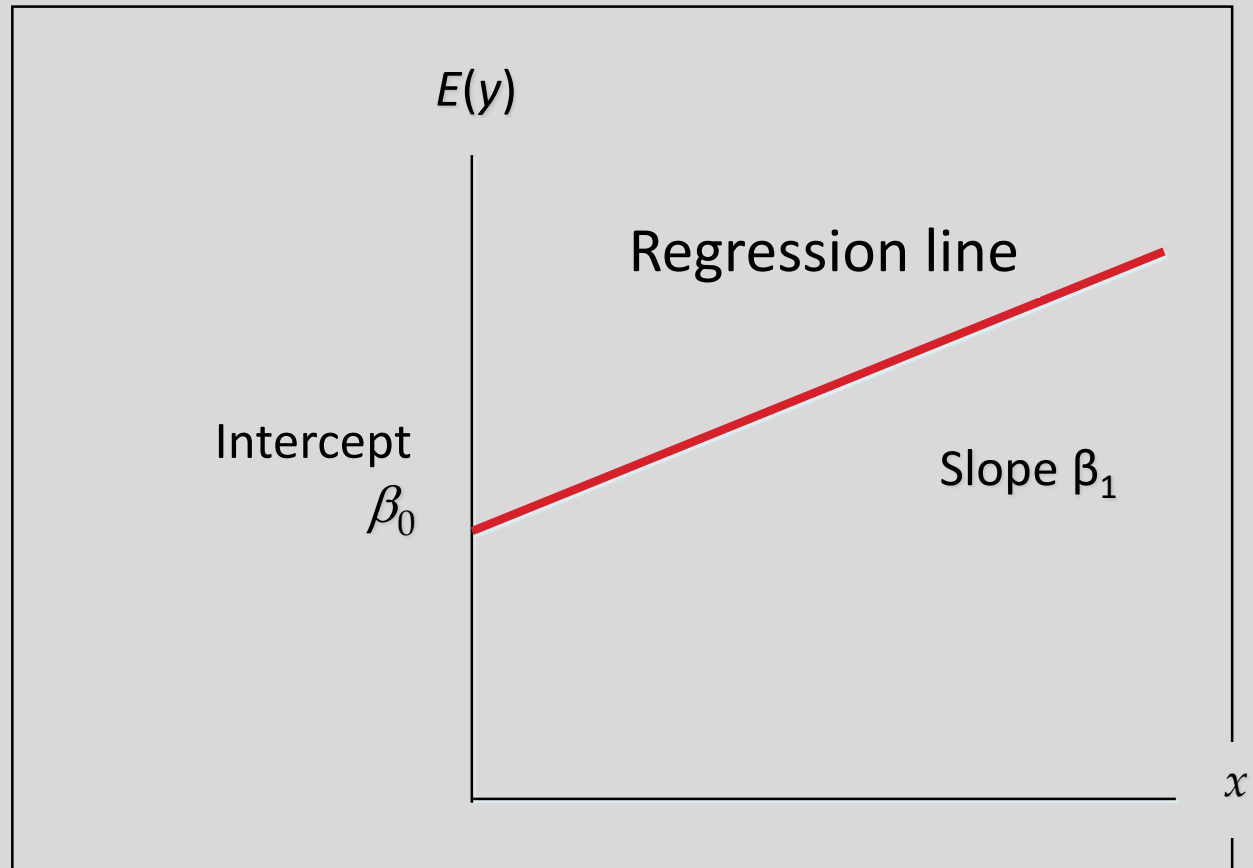


# Linear regression

- Given an input  $x$  compute an output  $y$
- For example:
  - Predict height from age
  - Predict house price from house area
  - Predict distance from wall from sensors



# Simple Linear Regression Equation



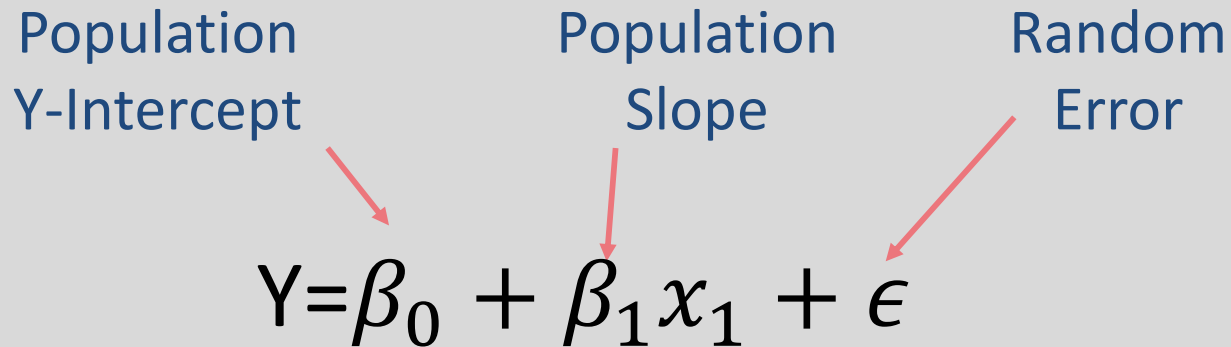
# Linear Regression Model

- Relationship Between Variables Is a Linear Function

Population  
Y-Intercept

Population  
Slope

Random  
Error

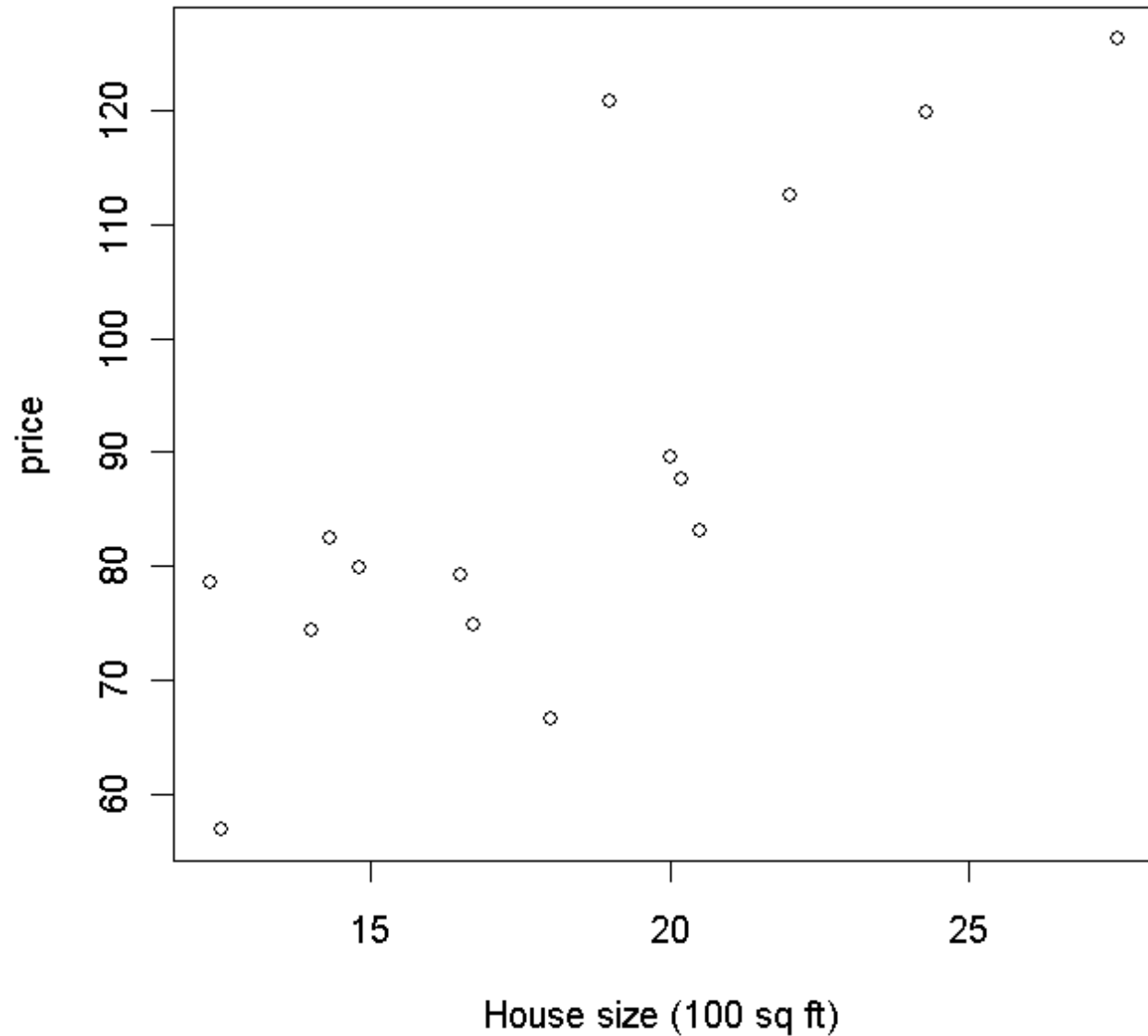

$$Y = \beta_0 + \beta_1 x_1 + \epsilon$$



House Number	Y: Actual Selling Price	X: House Size (100s ft <sup>2</sup> )
1	89.5	20.0
2	79.9	14.8
3	83.1	20.5
4	56.9	12.5
5	66.6	18.0
6	82.5	14.3
7	126.3	27.5
8	79.3	16.5
9	119.9	24.3
10	87.6	20.2
11	112.6	22.0
12	120.8	.019
13	78.5	12.3
14	74.3	14.0
15	74.8	16.7
Averages	88.84	18.17

Sample 15  
houses  
from the  
region.

# House price vs size



# Linear Regression – Multiple Variables

$$Y_i = b_0 + b_1X_1 + b_2X_2 + \cdots + b_pX_p + e$$

- $\beta_0$  is the intercept (i.e. the average value for Y if all the X's are zero),  $\beta_j$  is the slope for the  $j$ th variable  $X_j$

# Regression Model

- Our model assumes that

$$E(Y \mid X = x) = \beta_0 + \beta_1 x \quad (\text{the “population line”})$$

Population  
line

$$Y_i = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_p X_p + e$$

Least Squares  
line

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_1 + \hat{b}_2 X_2 + \cdots + \hat{b}_p X_p$$

We use  $\widehat{\beta}_0$  through  $\widehat{\beta}_p$  as guesses for  $\beta_0$  through  $\beta_p$  and  $\hat{Y}_i$  as a guess for  $Y_i$ . The guesses will not be perfect.

# Assumption

- The data may not form a perfect line.
- When we actually take a measurement (i.e., observe the data), we observe:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i,$$

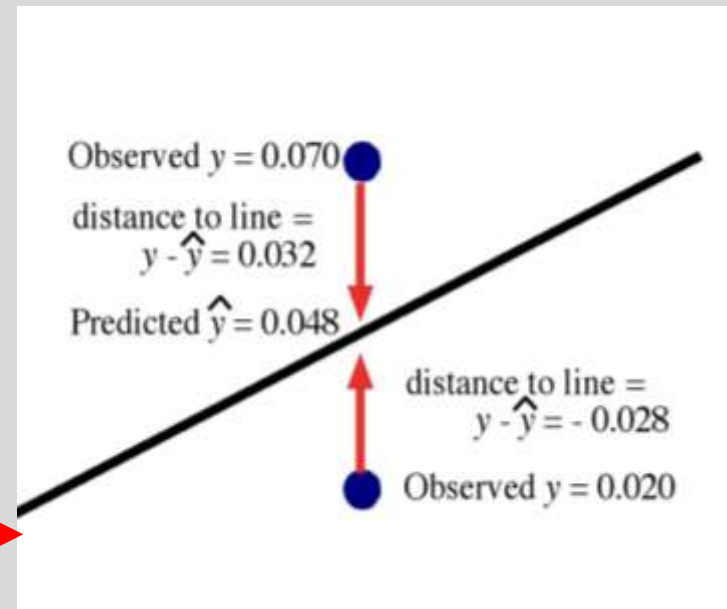
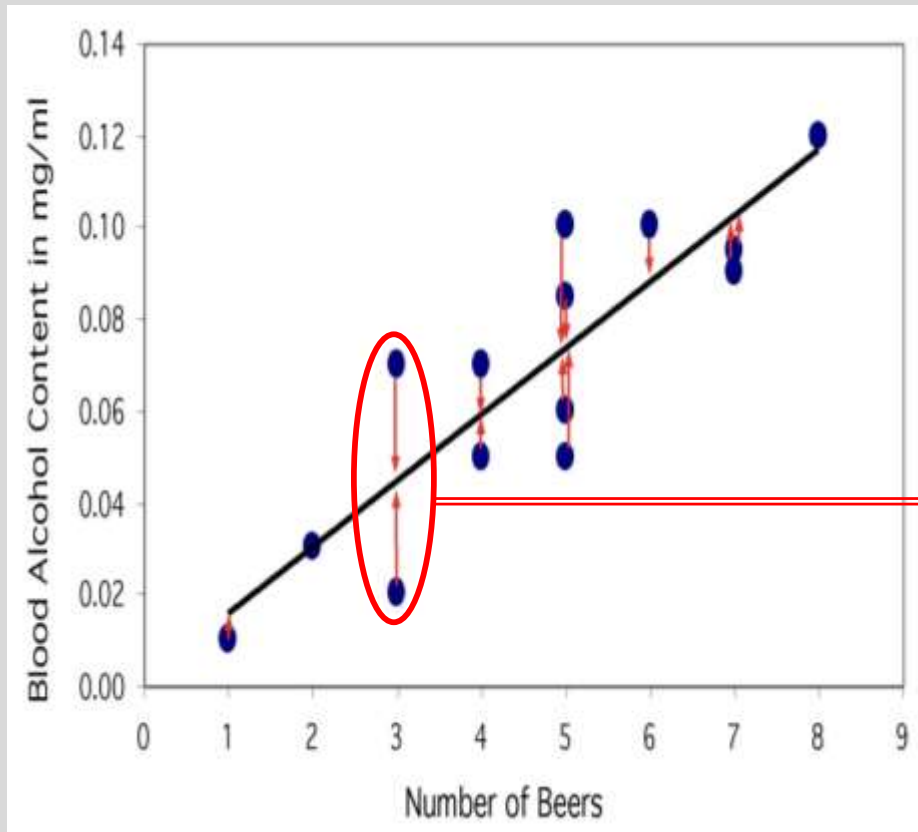
where  $\varepsilon_i$  is the random error associated with the  $i$ th observation.

# Assumptions about the Error

- $E(\varepsilon_i) = 0$  for  $i = 1, 2, \dots, n$ .
- $\sigma(\varepsilon_i) = \sigma_\varepsilon$  where  $\sigma_\varepsilon$  is unknown.
- The errors are independent.
- The  $\varepsilon_i$  are normally distributed (with mean 0 and standard deviation  $\sigma_\varepsilon$ ).

# The regression line

The least-squares regression line is the unique line such that the sum of the squared vertical (y) distances between the data points and the line is the smallest possible.



# Criterion for choosing what line to draw: method of least squares

- The method of least squares chooses the line (  $\widehat{\beta}_0$  and  $\widehat{\beta}_1$  ) that makes the sum of squares of the residuals  $\sum \varepsilon_i^2$  as small as possible
- Minimizes

$$\sum_{i=1}^n [y_i - (b_0 + b_1 x_i)]^2$$

for the given observations  $(x_i, y_i)$



# How do we "learn" parameters

- For the 2- $d$  problem

$$Y = b_0 + b_1 X$$

- To find the values for the coefficients which minimize the objective function we take the partial derivatives of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$b_1 = \frac{n \bar{a}_{xy} - \bar{a}_x \bar{a}_y}{n \bar{a}_{x^2} - (\bar{a}_x)^2}$$

$$b_0 = \frac{\bar{a}_y - b_1 \bar{a}_x}{n}$$

# Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

$$h(x) = \sum_{i=0}^n \beta_i x_i$$

- There is a closed form which requires matrix inversion, etc.
- There are iterative techniques to find weights
  - delta rule (also called LMS method) which will update towards the objective of minimizing the SSE.

# Linear Regression

$$h(x) = \sum_{i=0}^n \beta_i x_i$$

To learn the parameters  $\theta$  ( $\beta_i$ ) ?

- Make  $h(\mathbf{x})$  close to  $y$ , for the available *training examples*.
- Define a cost function  $J(\theta)$

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

- Find  $\theta$  that minimizes  $J(\theta)$ .

# LMS Algorithm

- Start a search algorithm (e.g. gradient descent algorithm,) with initial guess of  $\theta$ .
- Repeatedly update  $\theta$  to make  $J(\theta)$  smaller, until it converges to minima.

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

- $J$  is a convex quadratic function, so has a single global minima. gradient descent eventually converges at the global minima.
- At each iteration this algorithm takes a step in the direction of steepest descent(-ve direction of gradient).

# LMS Update Rule

- If you have only one training example  $(x, y)$

$$\begin{aligned}\frac{\partial}{\partial \theta} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h(x) - y) \frac{\partial}{\partial \theta_j} (h(x) - y) \\ &= (h(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left( \sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h(x) - y) x_j\end{aligned}$$

- For a single training example, this gives the update rule:

$$\beta_j = \beta_j + \alpha (y^{(i)} - h(x^i)) x_j^{(i)}$$

# m training examples

Repeat until convergence {

$$\theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h(x^{(i)})) x_j^{(i)}$$

}

**Batch Gradient Descent:** looks at every example on each step.

# Stochastic gradient descent

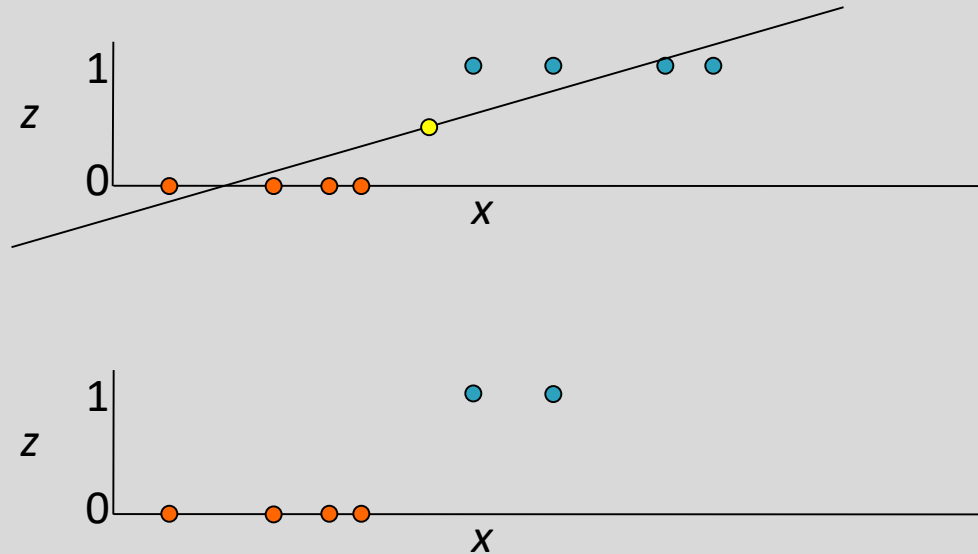
- Repeatedly run through the training set.
- Whenever a training point is encountered, update the parameters according to the gradient of the error with respect to that training example only.

Repeat {  
  for  $l = 1$  to  $m$  do  
     $\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x_j^{(i)}$  (for every  $j$ )  
  end for  
} until convergence

Thank You

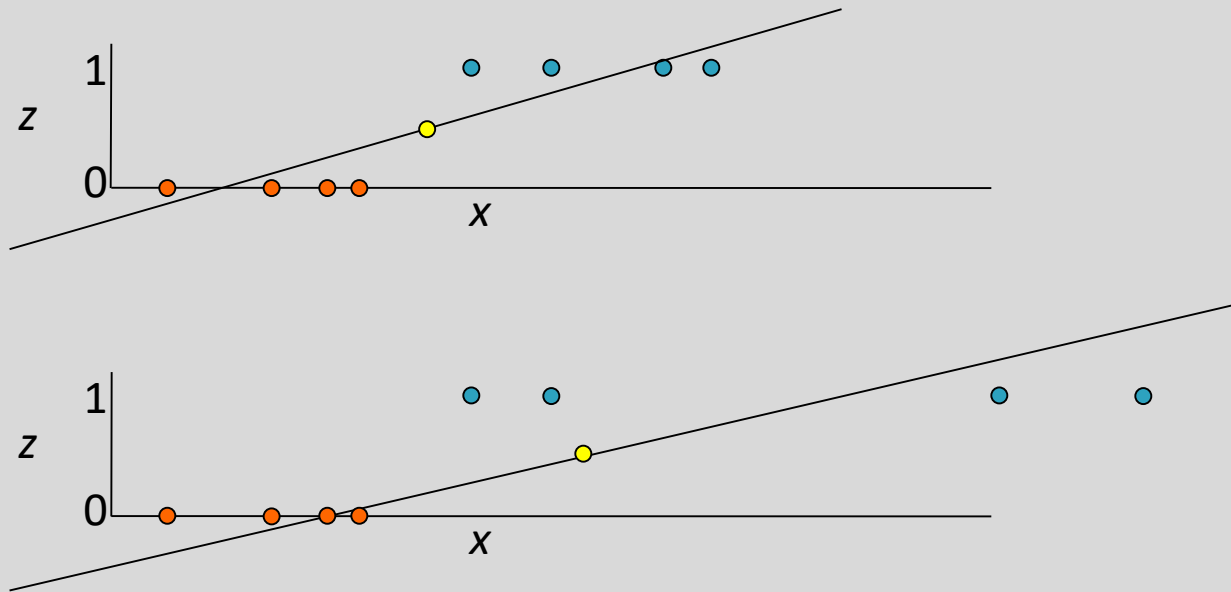


# Delta Rule for Classification



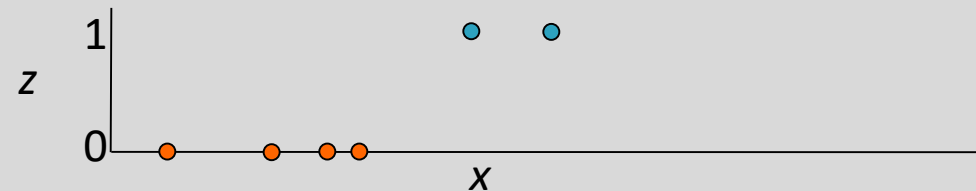
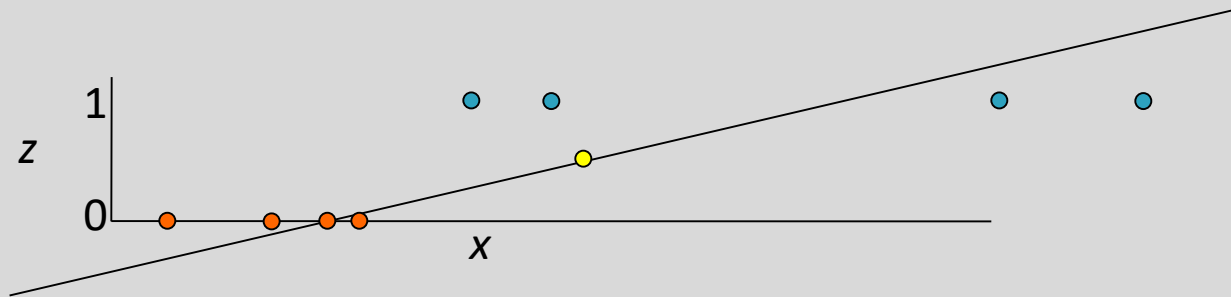
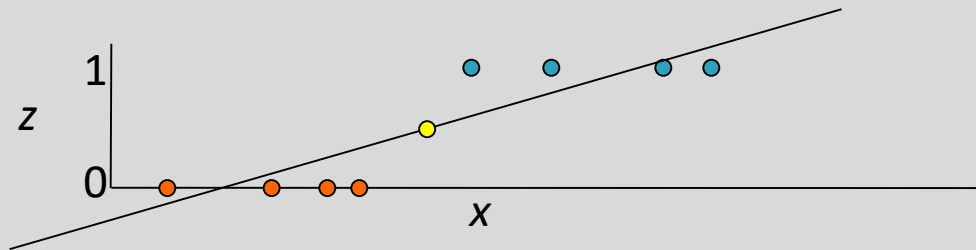
- What would happen in this adjusted case for perceptron and delta rule and where would the decision point (i.e. .5 crossing) be?

# Delta Rule for Classification



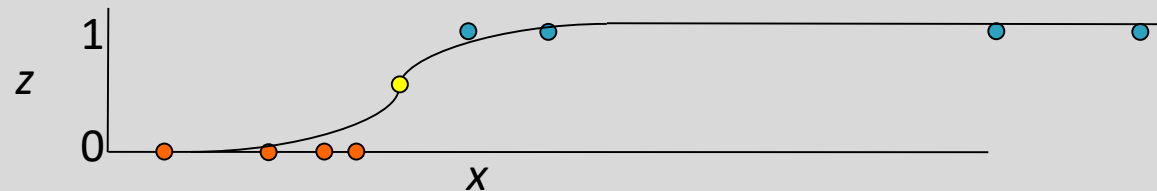
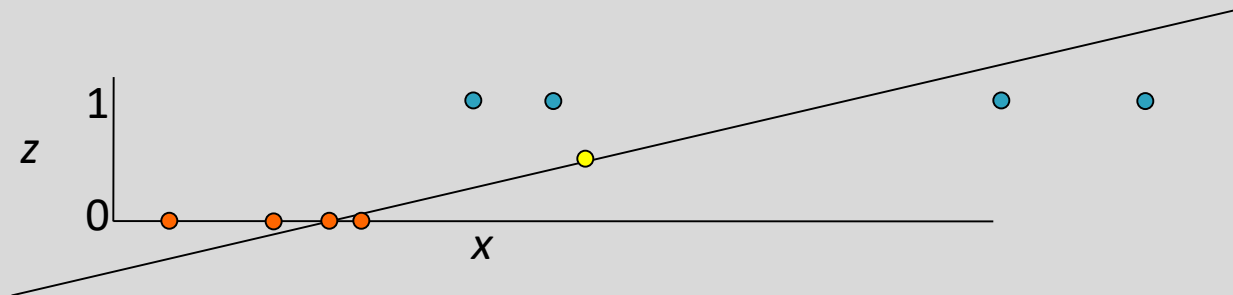
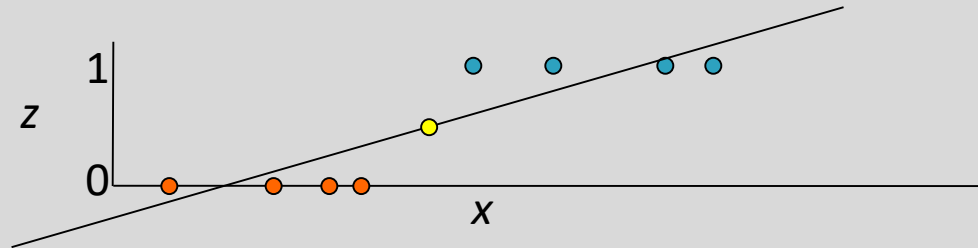
- Leads to misclassifications even though the data is linearly separable
- For Delta rule the objective function is to minimize the regression line SSE, not maximize classification

# Delta Rule for Classification



- What would happen if we were doing a regression fit with a sigmoid/logistic curve rather than a line?

# Delta Rule for Classification



- Sigmoid fits many decision cases quite well! This is basically what logistic regression does.