

# Supervised Learning : Linear Regression and Logistic Regression

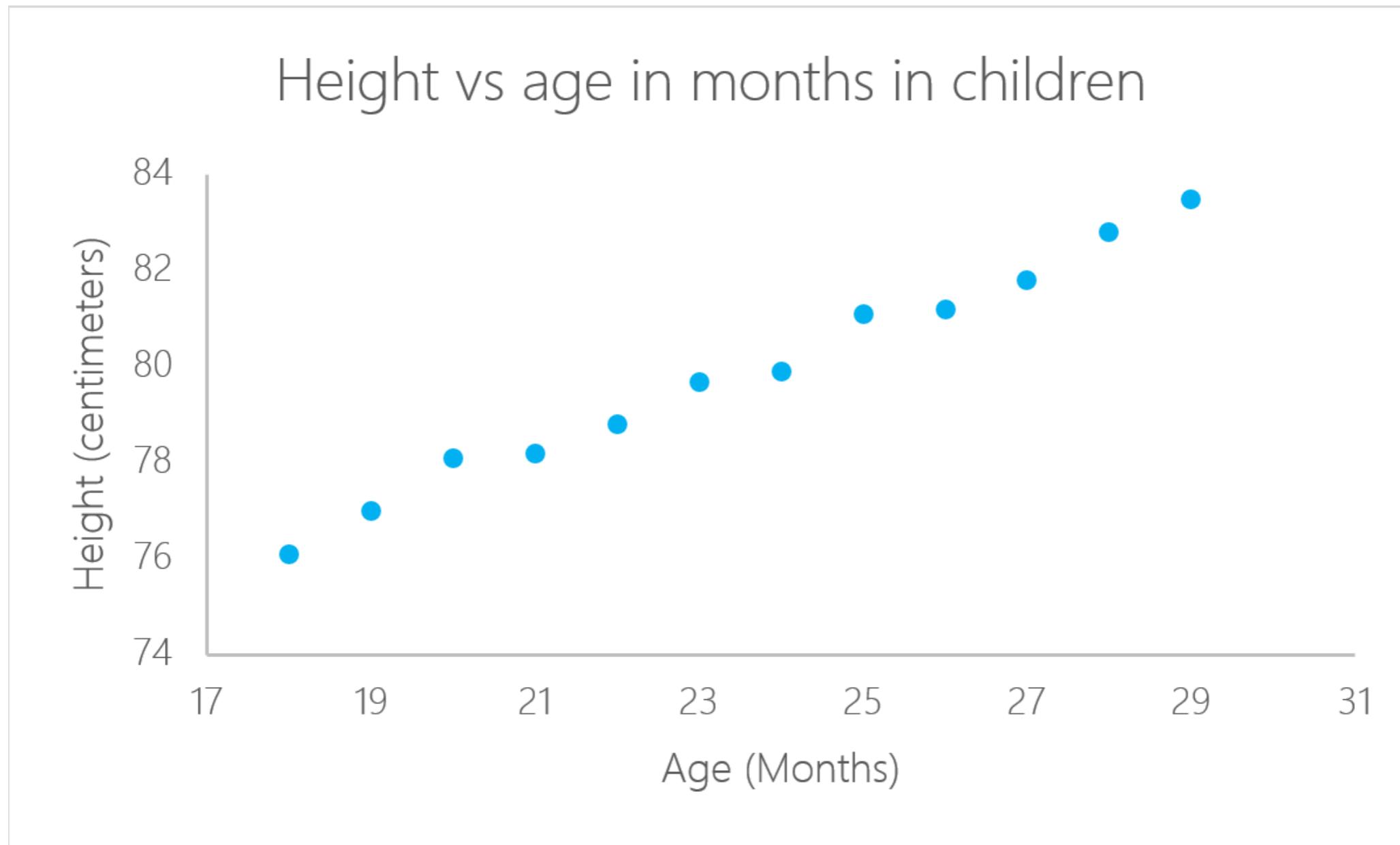
Dr. Srijith P K

Computer Science and Engineering

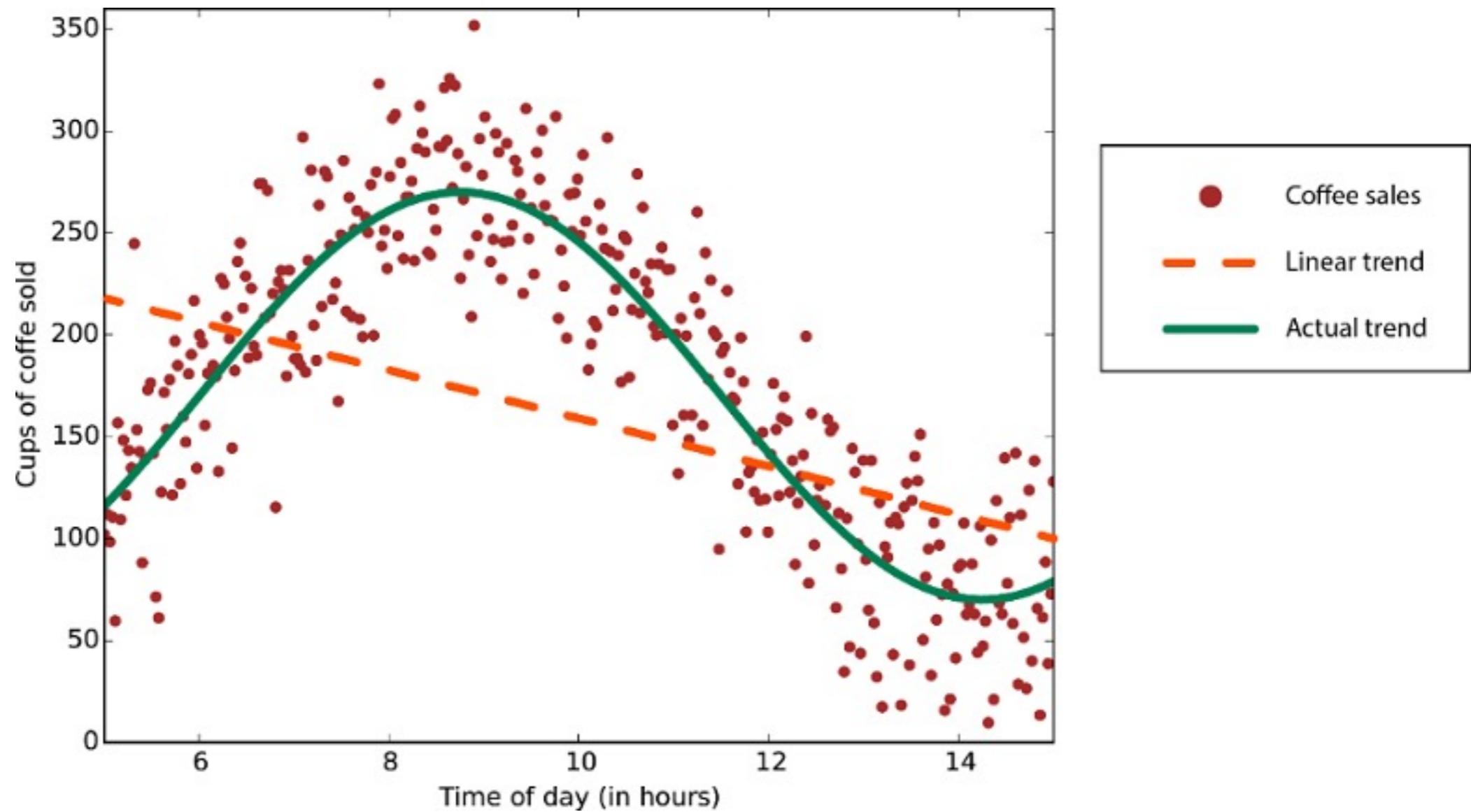
IIT Hyderabad



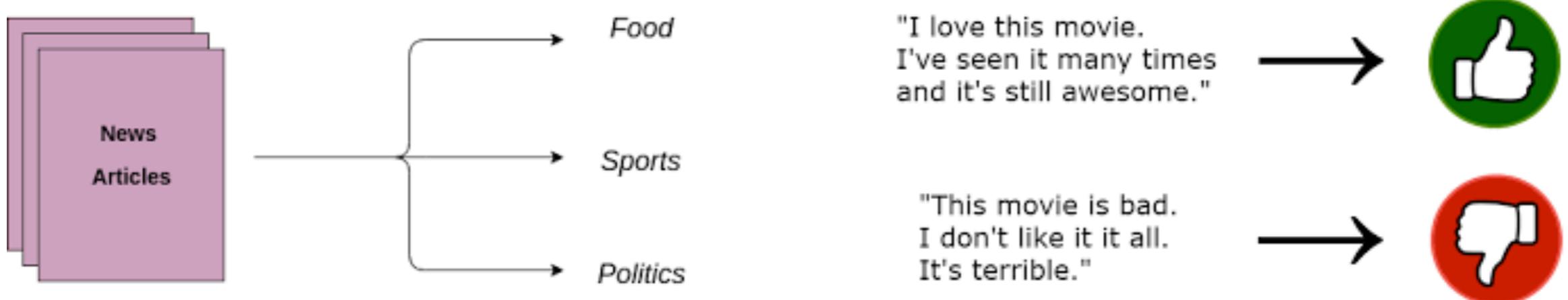
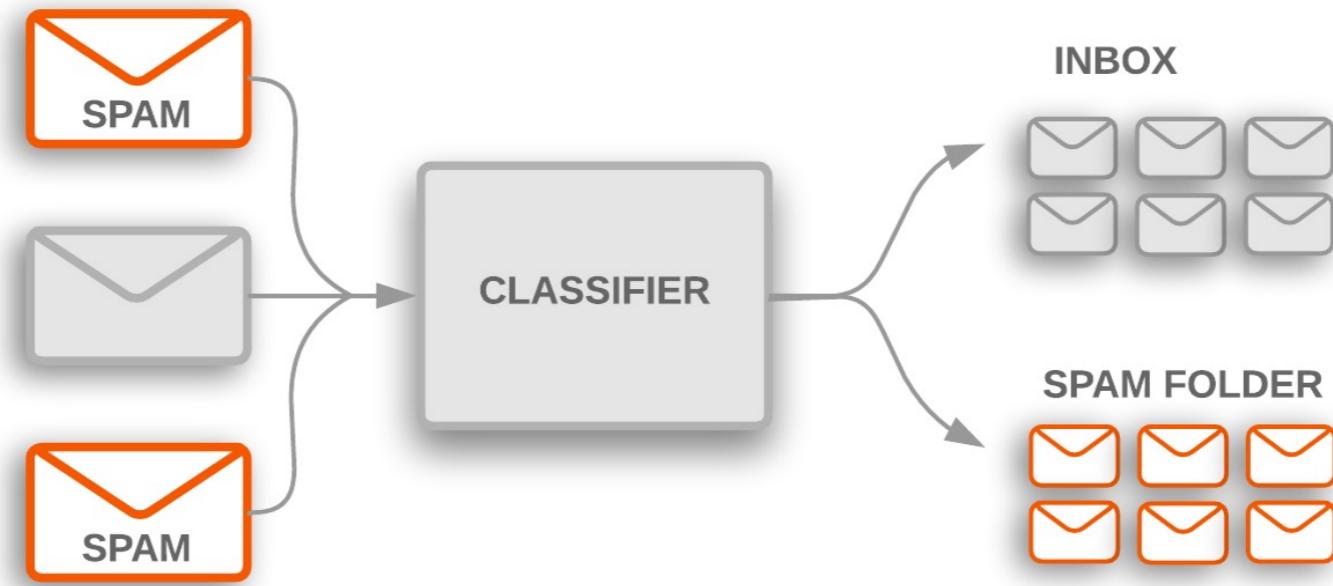
# Supervised learning



# Supervised learning



# Supervised Learning

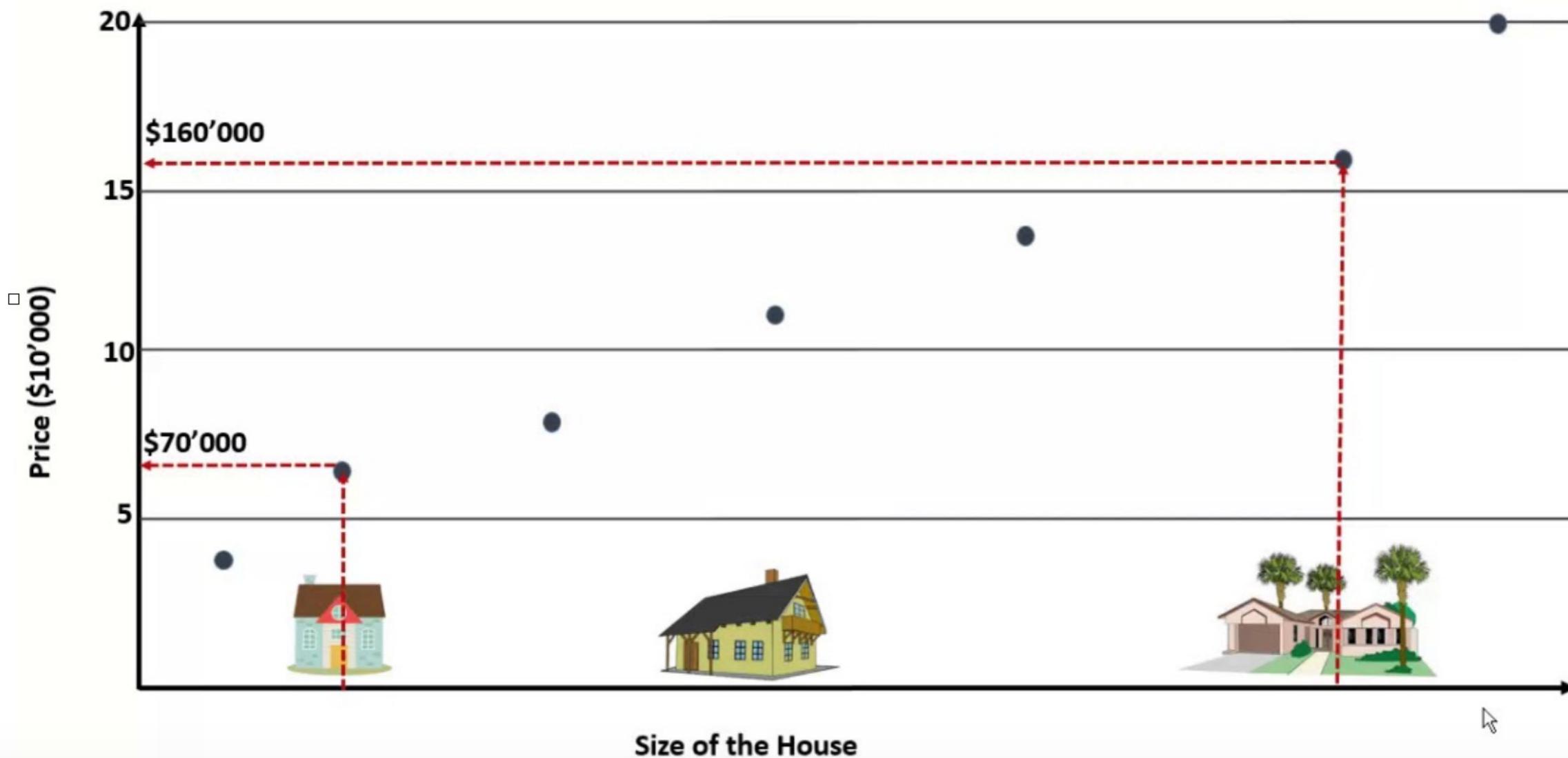


# Outline

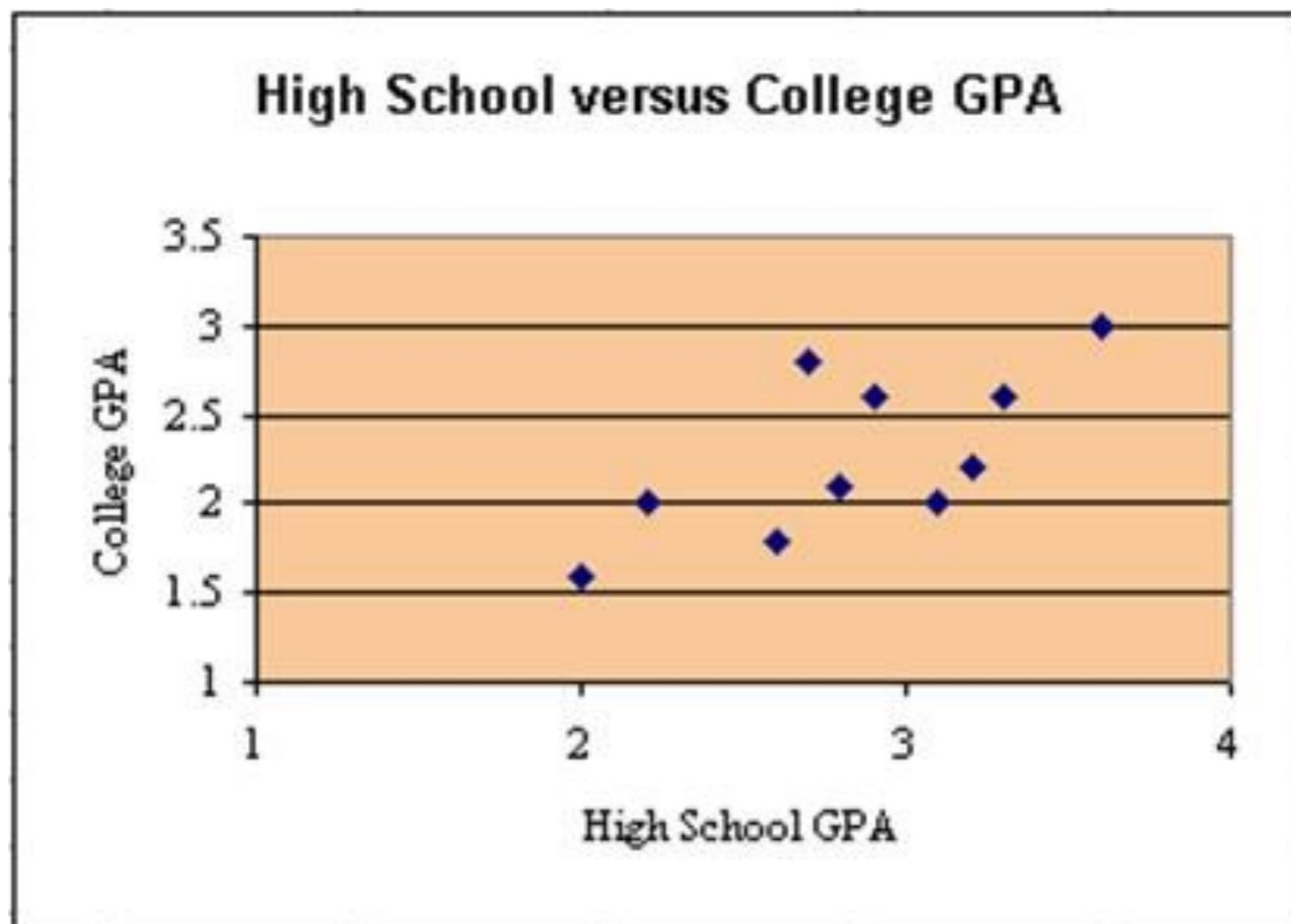
- Supervised Learning
- Regression
  - Linear Regression
  - Logistic Regression
  - Poisson Regression
- Classification

# Supervised learning : Regression

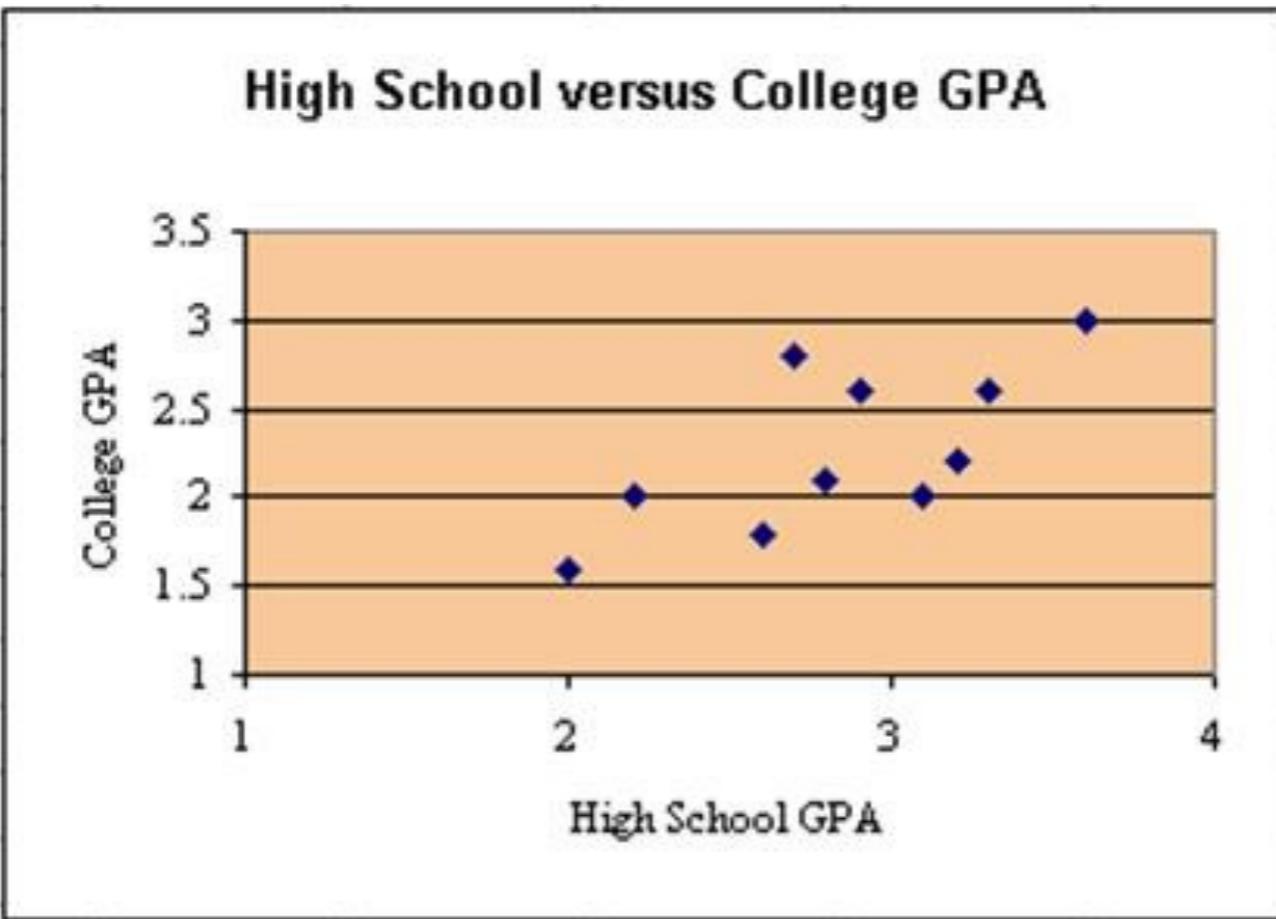
## Estimating Price of a house



# Supervised learning : Regression



# Supervised learning : Regression



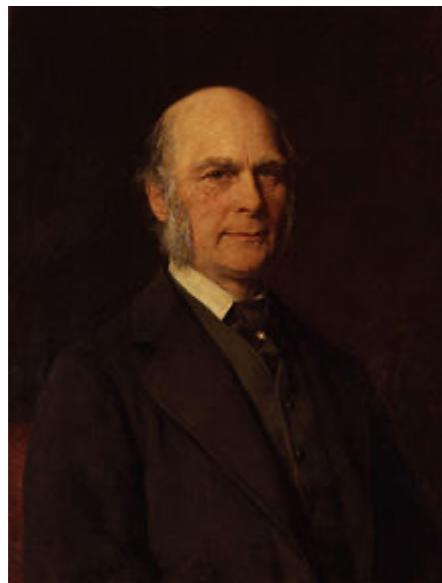
Real valued targets (outputs)

Generalization performance

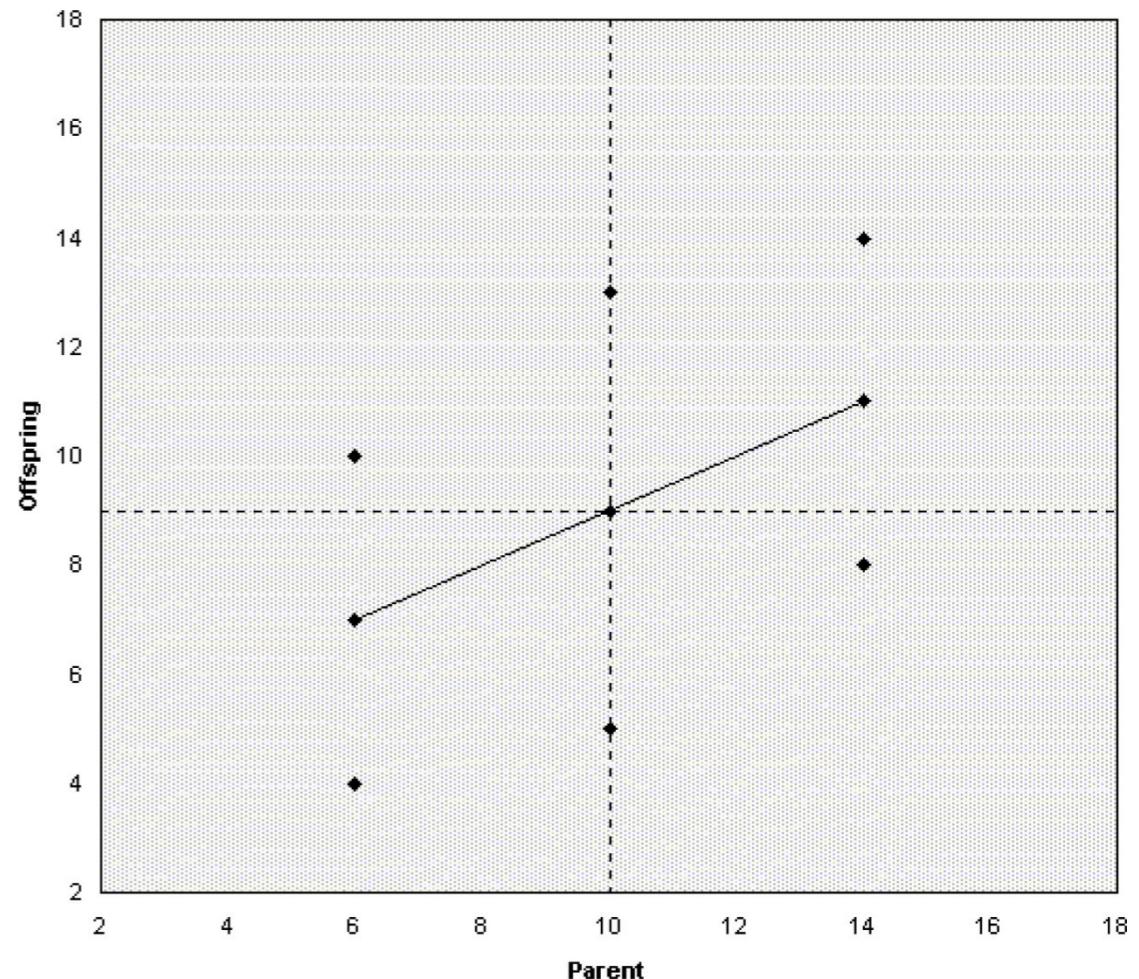
Goal is to learn a function which maps inputs to outputs so that it will predict well on future data points

# A historical look at regression

Genetics : parent sweet pea size on the  $X$ -axis and the offspring sweet pea size on the  $Y$ -axis (1900s)



Sir Francis Galton



$$\{(6, 4), (6, 7), (6, 10), (10, 5), (10, 9), (10, 13), (14, 8), (14, 11), (14, 14)\}$$

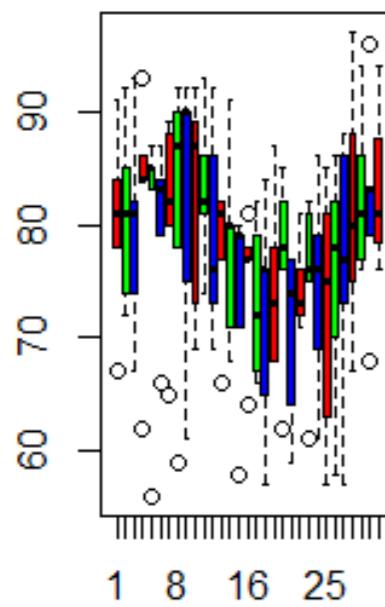
# Airquality data.

- Data set has various air quality parameters in New York city.
- These are the parameters in the data set:
  - Daily temperature from May to August
  - Solar radiation data
  - Ozone data
  - Wind data
- Goal : predict the temperature for a particular month in New York using solar radiation, ozone and wind data.

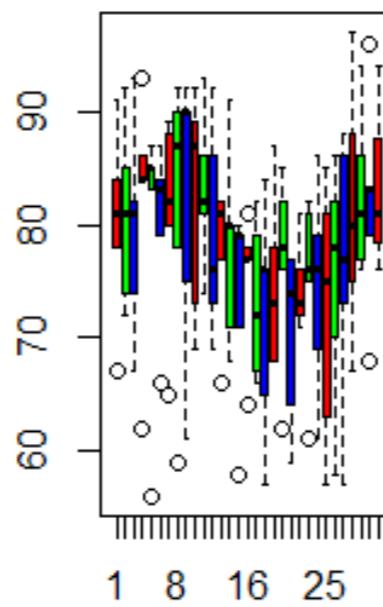
# Airquality data

```
##   Ozone Solar.R Wind Temp Month Day
## 1   41    190  7.4   67     5    1
## 2   36    118  8.0   72     5    2
## 3   12    149 12.6   74     5    3
## 4   18    313 11.5   62     5    4
## 5   NA     NA 14.3   56     5    5
## 6   28     NA 14.9   66     5    6
## 7   23    299  8.6   65     5    7
## 8   19     99 13.8   59     5    8
## 9    8     19 20.1   61     5    9
## 10  NA    194  8.6   69     5   10
```

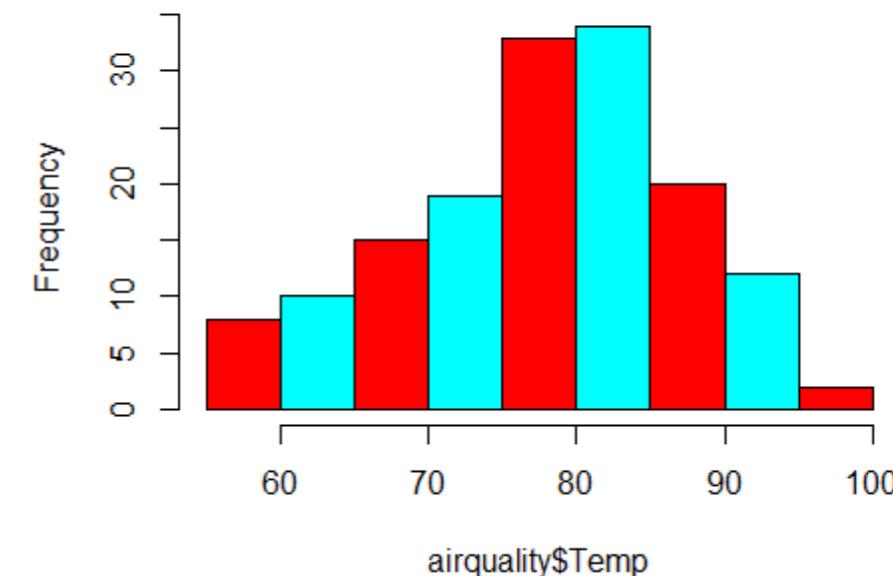
Month 5



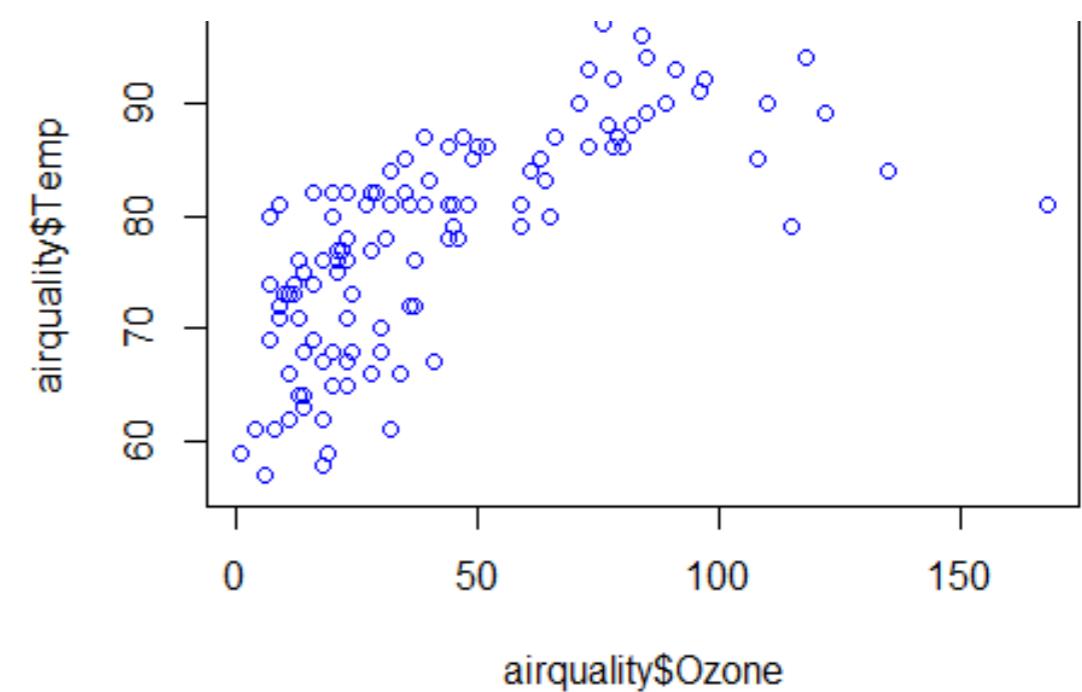
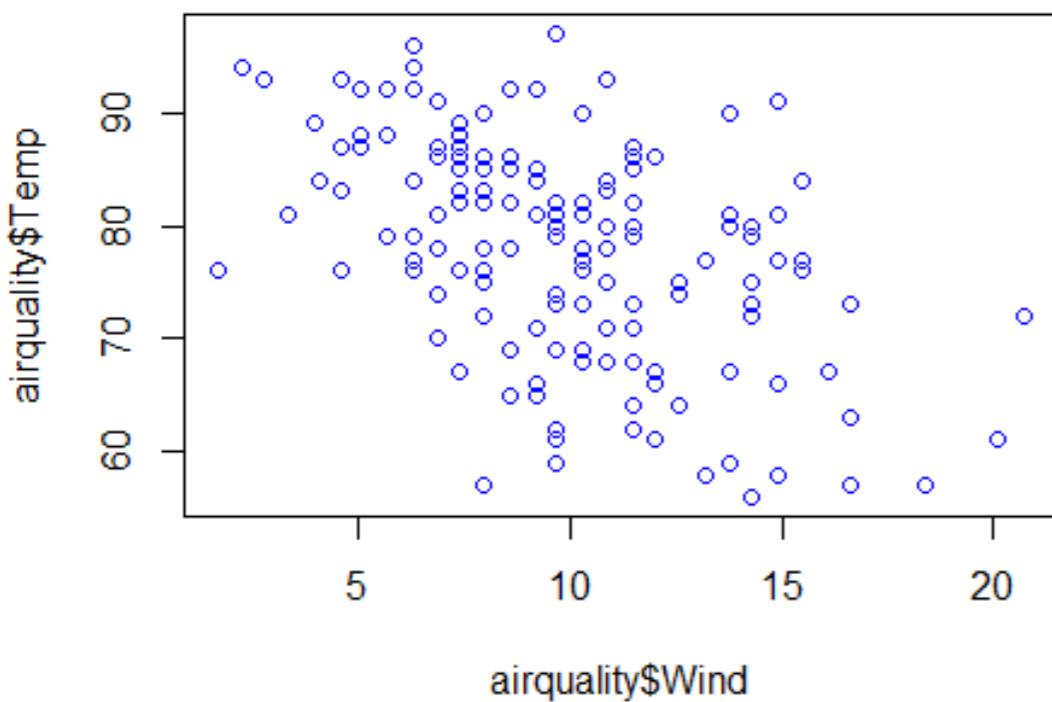
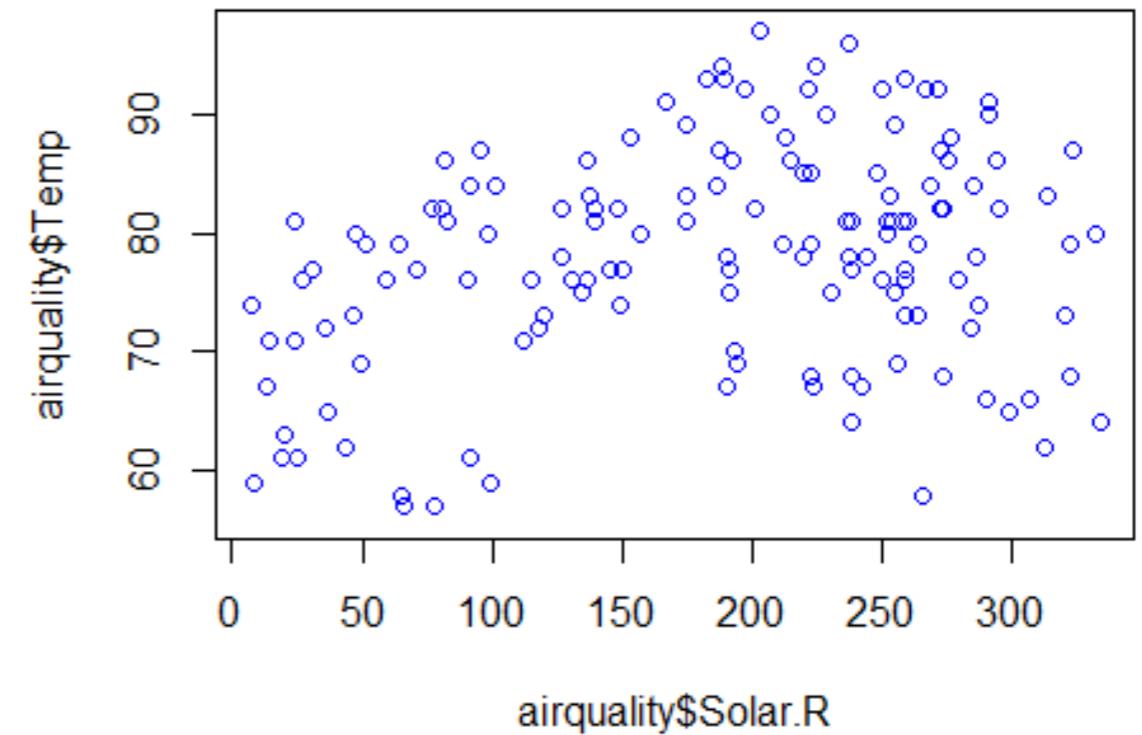
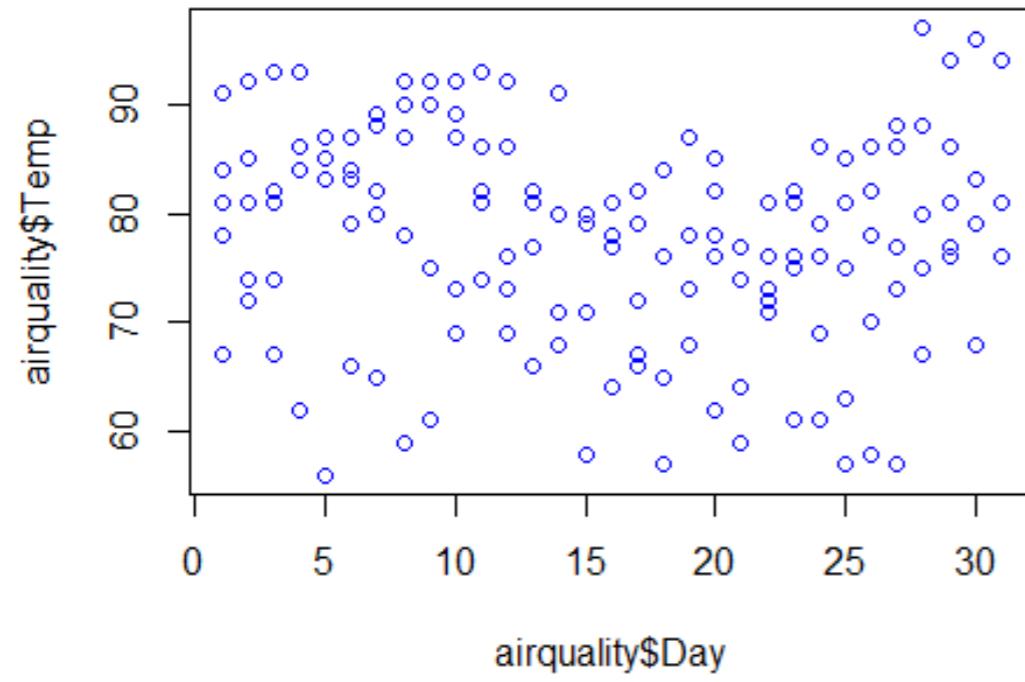
Month 6



Histogram of airquality\$Temp



# Airquality data

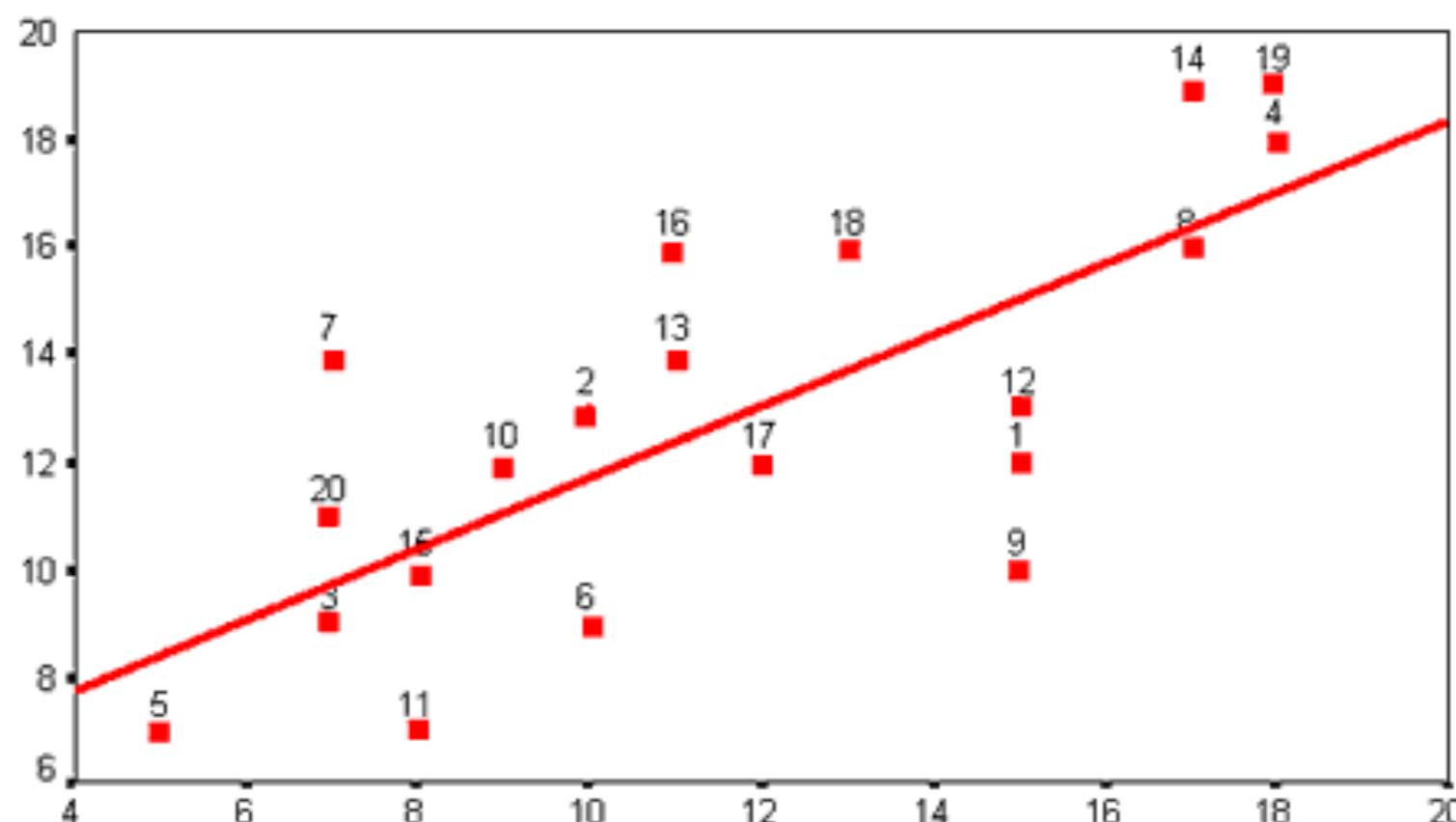


# Linear regression

- $\text{Temp} = w_1 \cdot \text{Solar.R} + w_2 \cdot \text{Ozone} + w_3 \cdot \text{Wind} + \text{error}$ .
- Temperature of house depends on ozone, wind and solar radiations
- linear regression helps to discover relation between dependent and independent variables

# Linear Regression

- Observations need not lie on a line
  - Observations are not generated by a linear line
  - Observations are noisy, due to measurement errors



# Linear regression

- Learn a function which maps input to output  $f : X \rightarrow Y$
- Consider a Linear function

$$\hat{Y}_i = w_0 + w_1 X_i$$

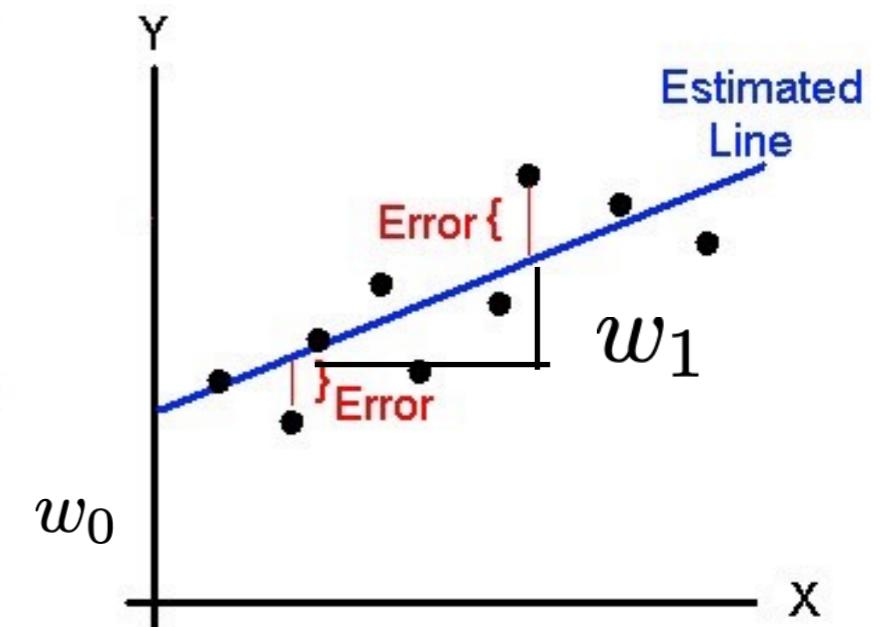
Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

Regression Output is real and scalar,  $y \in \mathbb{R}$



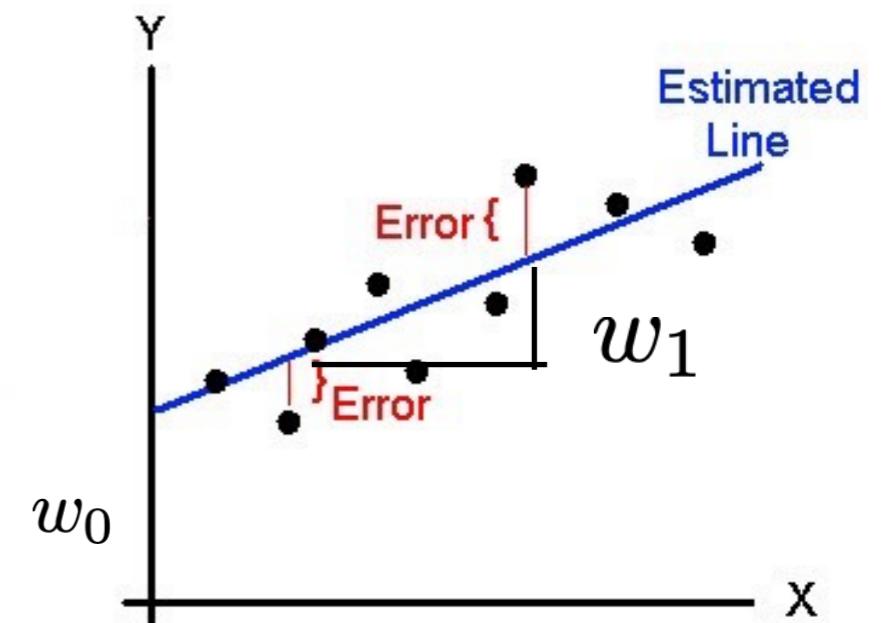
# Linear regression

- Learn a function which maps input to output  $f : X \rightarrow Y$
- Consider a Linear function

Regression Output is real and scalar,  $y \in \mathbb{R}$

$$\hat{Y}_i = w_0 + w_1 X_i$$
$$= X_i^T w$$

Estimated (or predicted) Y value for observation i  
Estimate of the regression intercept  
Estimate of the regression slope  
Value of X for observation i



$$\begin{array}{lll} \text{1 dim input } X_i = [1, X_i]^T & w = [w_0, w_1]^T \\ \text{D dim input } X_i = [1, X_{i1}, \dots, X_{iD}]^T & w = [w_0, w_1, \dots, w_D]^T \end{array}$$

# Linear Regression - Learning Parameters

- Learn the function which passes through as many points as possible :  
Minimize the **Least Squares Error**

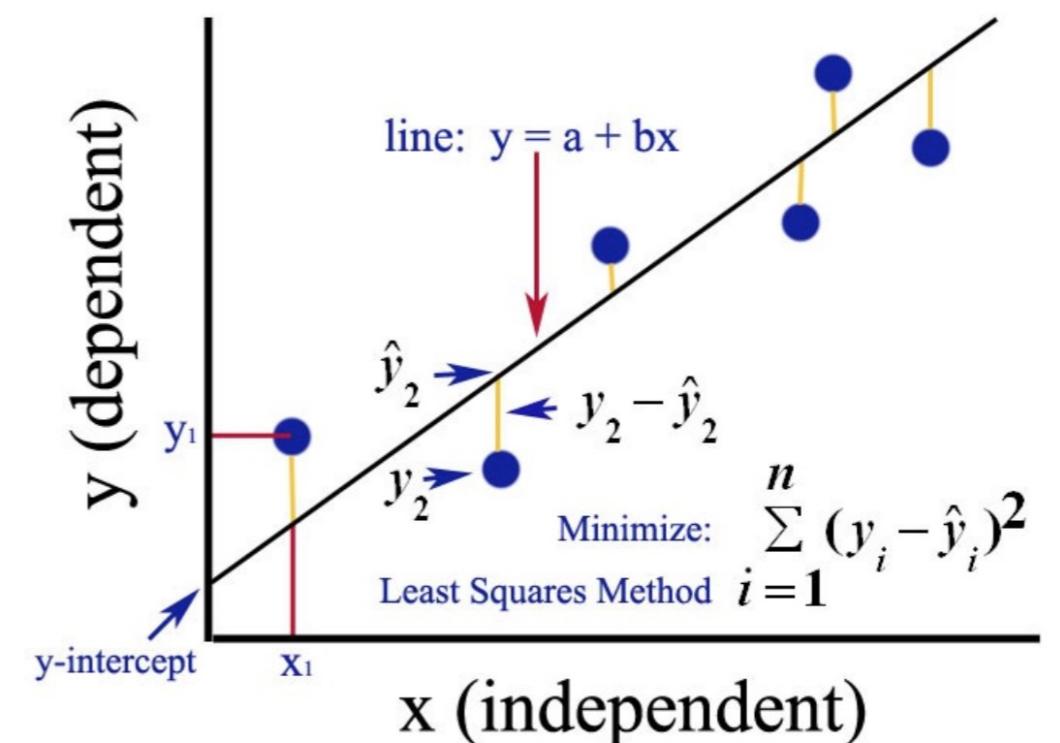
$$E(w) = \frac{1}{2} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^N (y_i - X_i^\top w)^2$$

$$\frac{1}{2} \| (y - X^\top w) \|^2$$

$$X_i = [1, X_{i1}, \dots, X_{iD}]^\top$$

Design matrix  $X = \begin{pmatrix} X_1, X_2, \dots, X_N \end{pmatrix}^{(D+1) \times N}$



# Linear Regression - Learning Parameters

- Learn the function which passes through as many points as possible :  
Minimize the **Least Squares Error**

$$E(w) = \frac{1}{2} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^N (y_i - X_i^\top w)^2$$

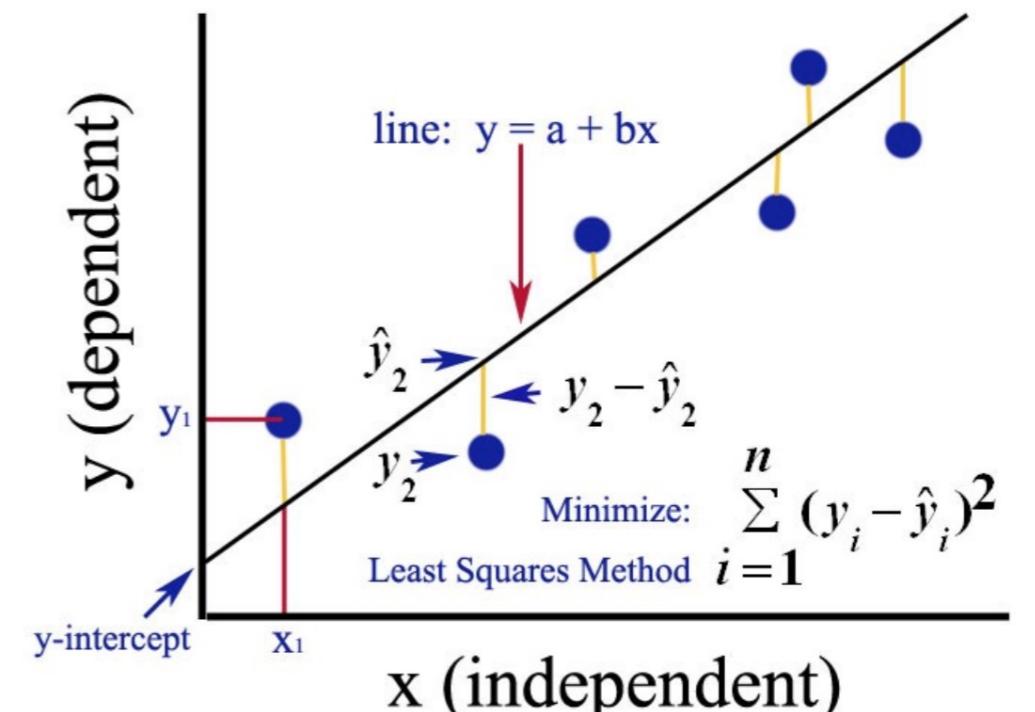
$$\frac{1}{2} \| (y - X^\top w) \|^2$$

$$\nabla E(w) = Xy - XX^\top w = 0$$

$$w_{ML} = (XX^\top)^{-1}Xy$$

**Design matrix**

$$X = \begin{pmatrix} X_1, X_2, \dots, X_N \end{pmatrix}^{(D+1) \times N}$$



$$\frac{\partial}{\partial s} (x - As)^\top W (x - As) = -2(x - As)^\top WA$$

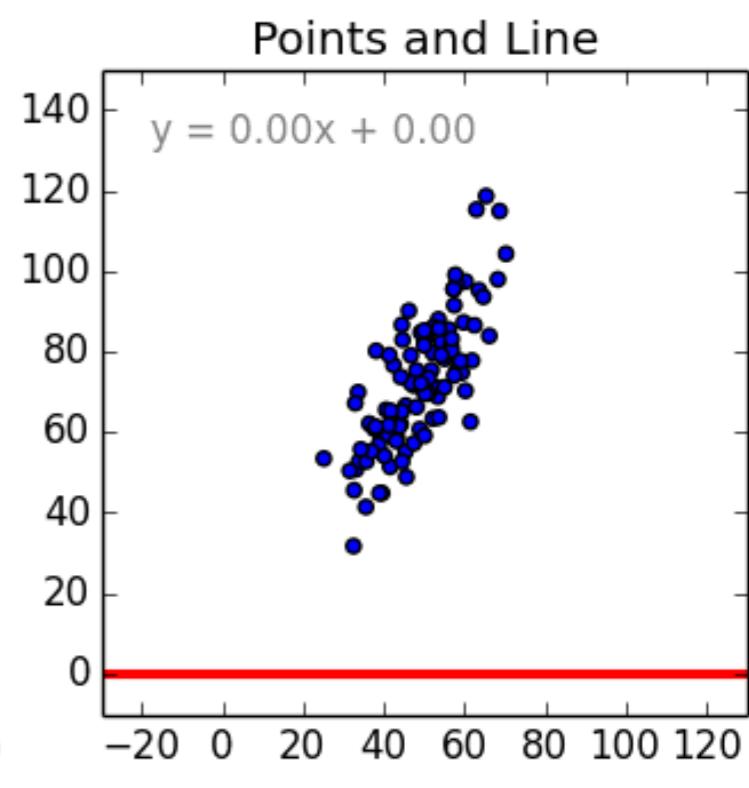
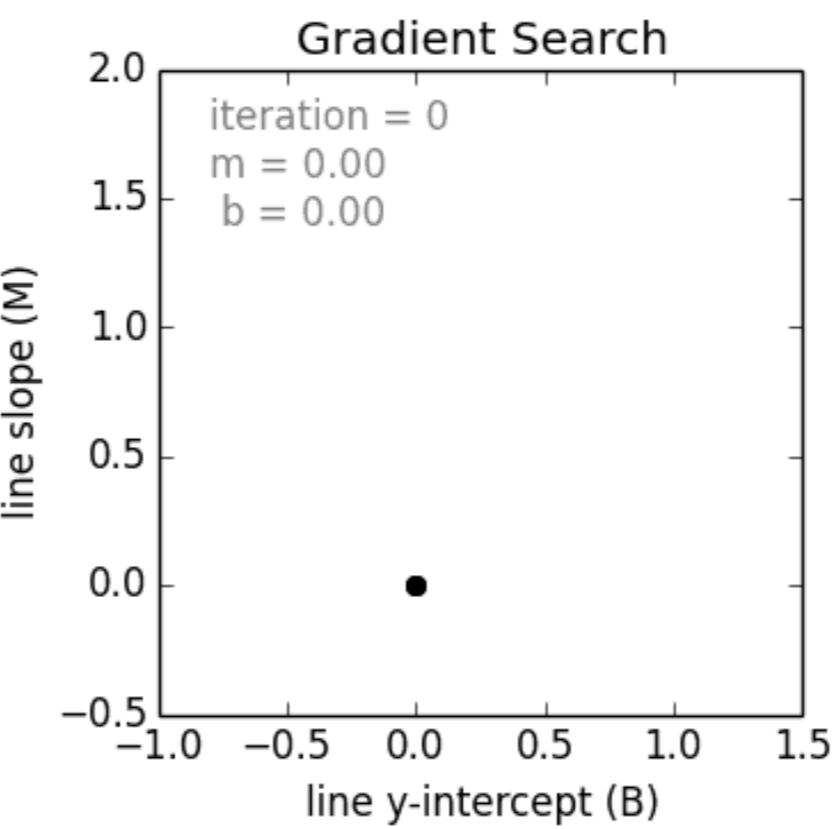
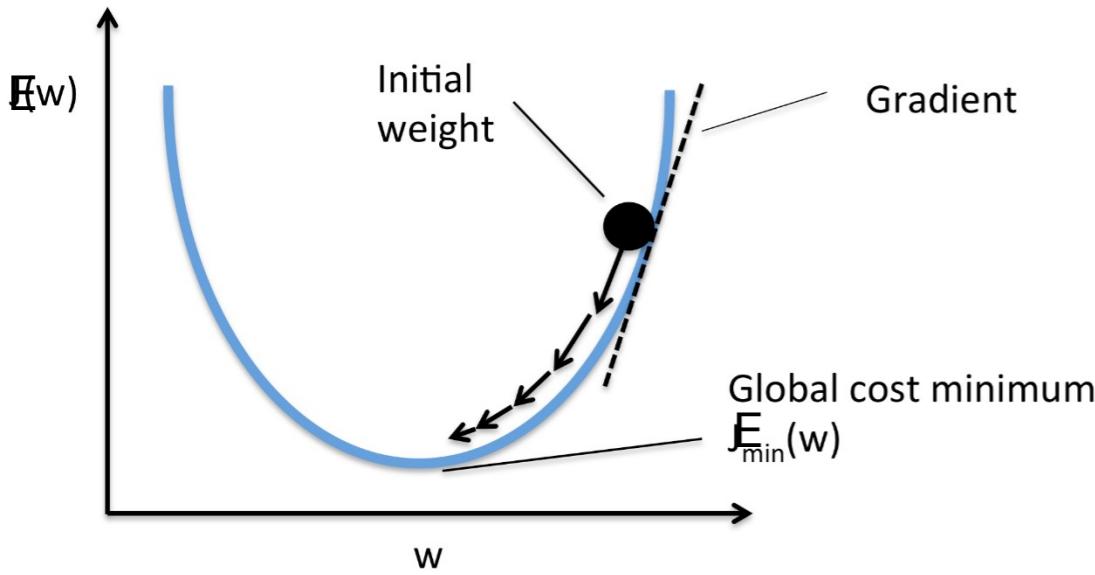
# Linear Regression - Learning Parameters

- Learn the function which passes through as many points as possible  
: Minimize the **least squares error** using **gradient descent**

$$\nabla E(w) = Xy - XX^\top w$$

$$Error_{(m,c)} = \frac{1}{N} \sum_{i=1}^N (y_i - (mx_i + c))^2$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$



# Question

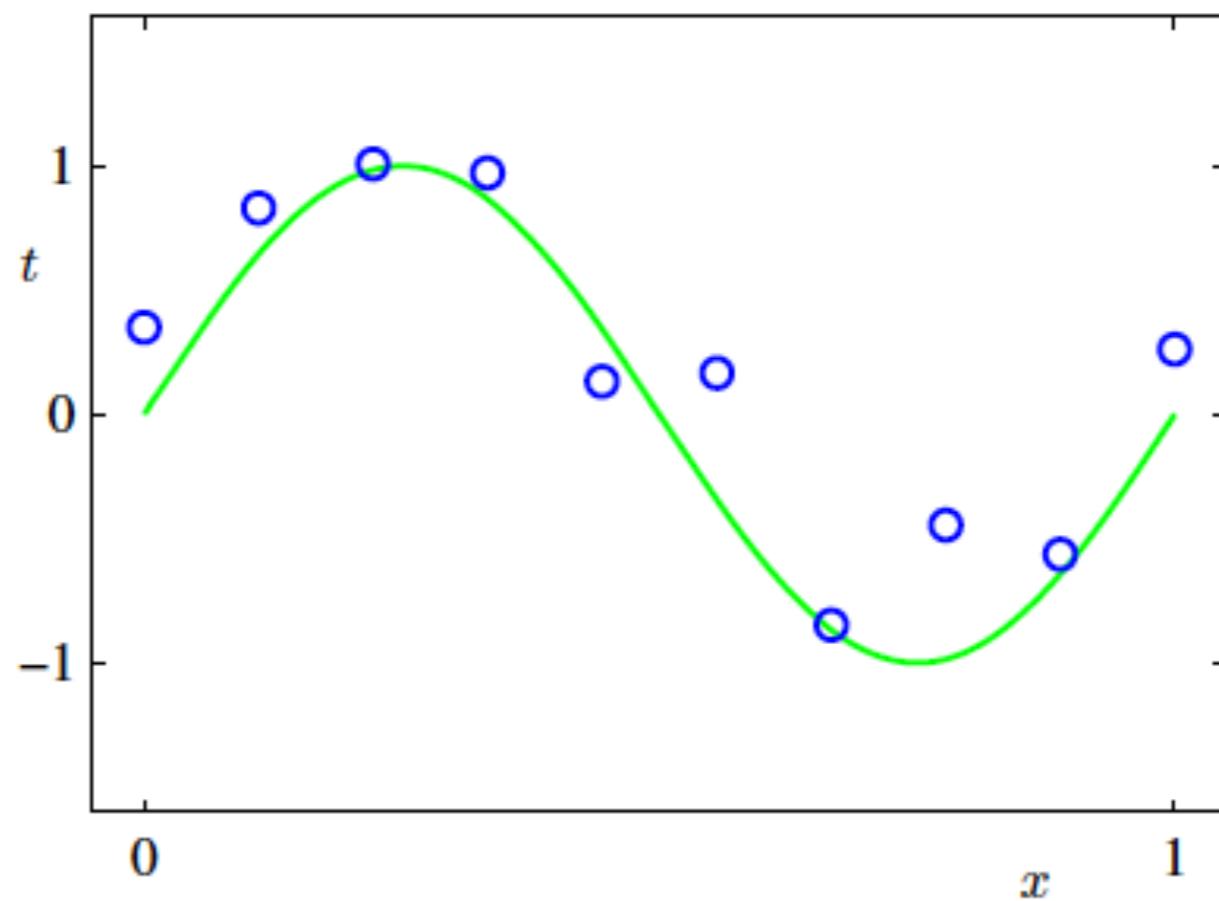
- **write down** three equations for the line  $y = mx+c$  to go through  $y = 7$  at  $x = -1$ ,  $y = 7$  at  $x = 1$  and  $y = 21$  at  $x = 2$ . Find the least squares solution  $(c,m)$ ?
- Implement In python least squares solution to linear regression
  - Analytical approach
  - Gradient descent approach

## Question

- write down three equations for the line  $y = mx+c$  to go through  $y = 7$  at  $x = -1$ ,  $y = 7$  at  $x = 1$  and  $y = 21$  at  $x = 2$ . Find the least squares solution  $(c,m)$ ?
- Answer : (9,4)

# Regression - curve fitting

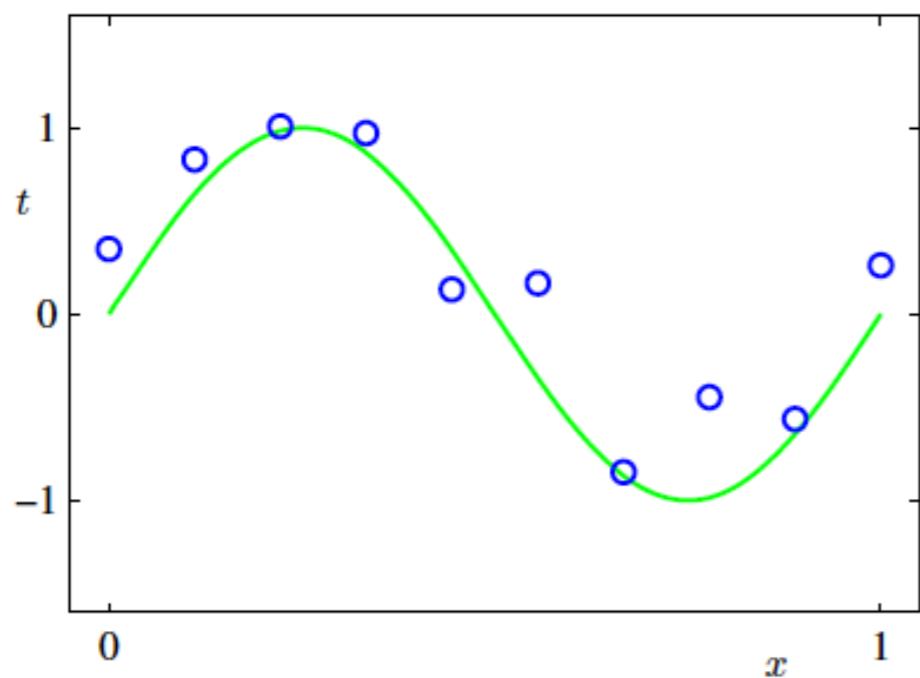
- Remember high school maths !
- Real-valued target variable  $t$ .
- Training set comprising  $N$  observations



# Polynomial Regression - curve fitting

- M is the order of the polynomial,  $y(x,w)$  is a nonlinear function of x, it is a linear function of the coefficients w.
- Functions, such as the polynomial, which are linear in the unknown parameters have important properties and are called **linear models**

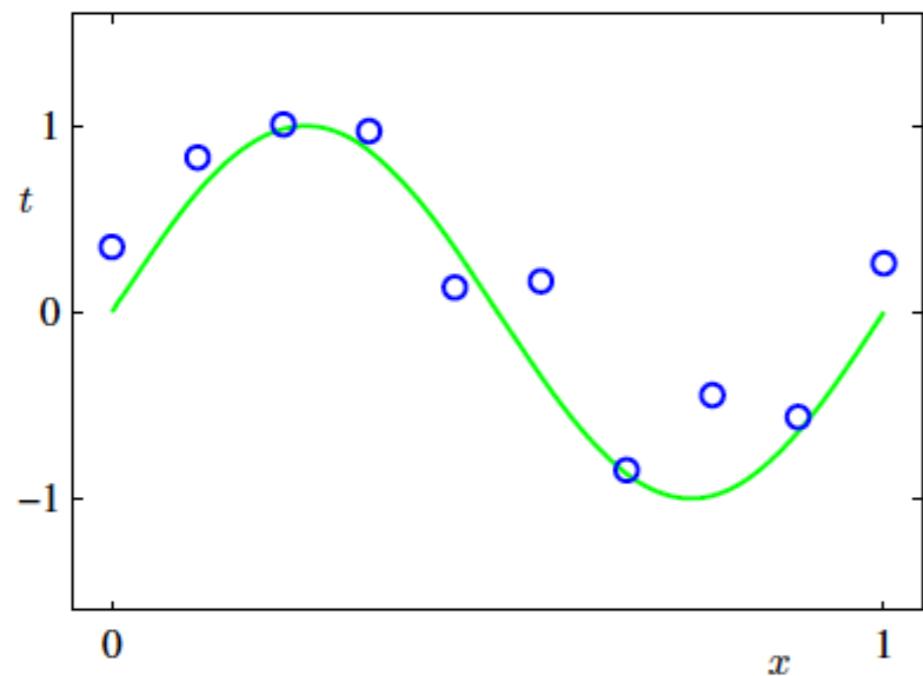
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$



# Polynomial Regression - curve fitting

- M is the order of the polynomial,  $y(x,w)$  is a nonlinear function of x, it is a linear function of the coefficients w.
- Functions, such as the polynomial, which are linear in the unknown parameters have important properties and are called **linear models**

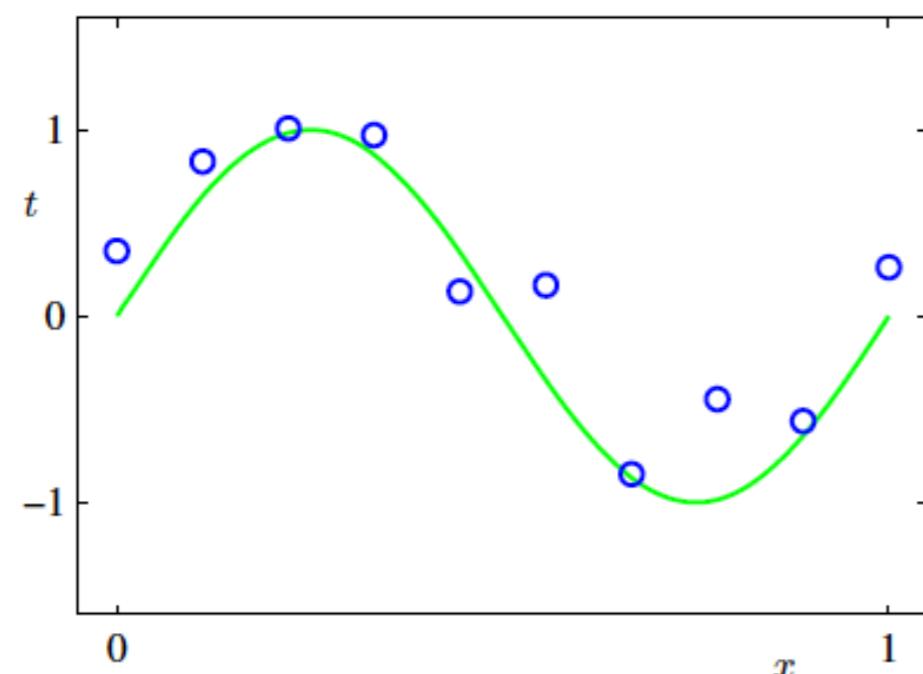
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$



Linear  
regression

# Polynomial Regression - curve fitting

- M is the order of the polynomial,  $y(x,w)$  is a nonlinear function of x, it is a linear function of the coefficients w.
- Functions, such as the polynomial, which are linear in the unknown parameters have important properties and are called **linear models**



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$

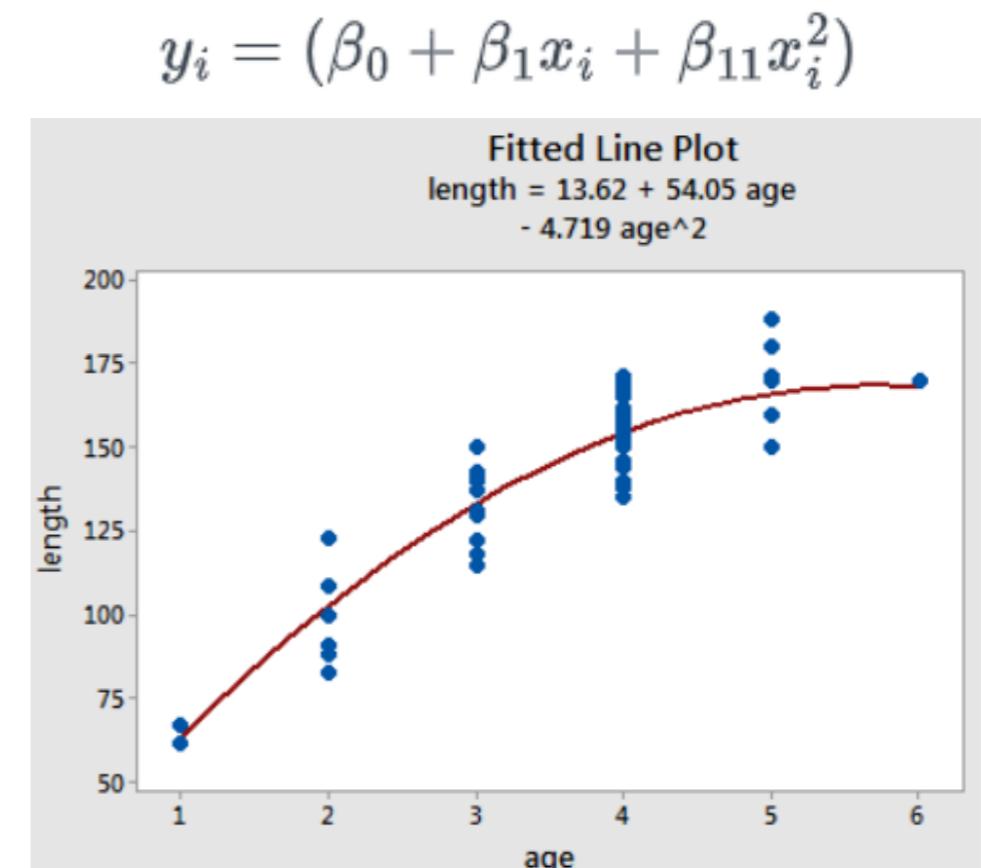
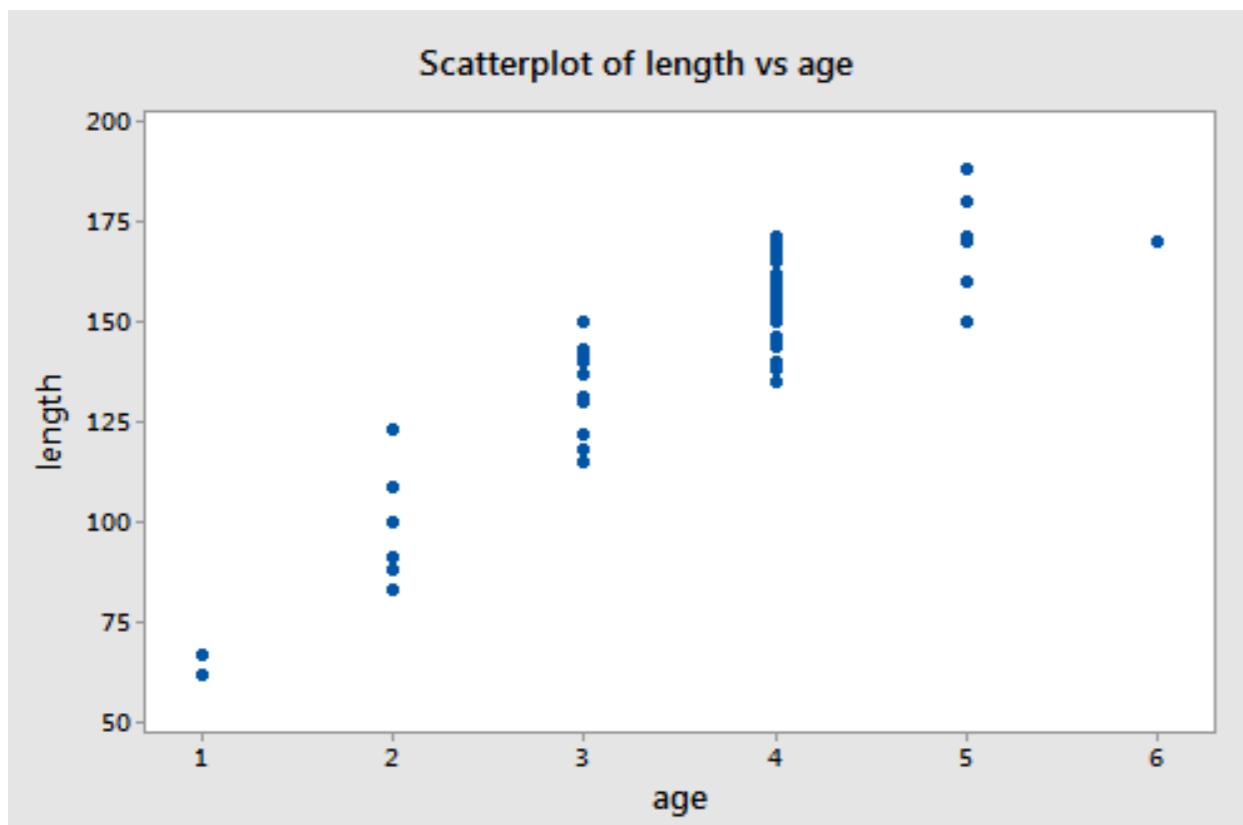
Non-Linear  
regression

$$y_i = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$
$$y_i = \frac{\beta_0 + \beta_1 x_i}{1 + \beta_2 e^{\beta_3 x_i}}$$

$$Y = \frac{\beta_0 X}{\beta_1 + X}$$

# Polynomial Regression - Example

- Some researchers are interested in learning how the length of a bluegill fish is related to its age. To study this, some bluegill fishes were randomly sampled from a Lake and recorded the following data
  - Response (y): length (in mm) of the fish
  - Input (x): age (in years) of the fish
- Positive trend but not linear : **Second-order polynomial model**

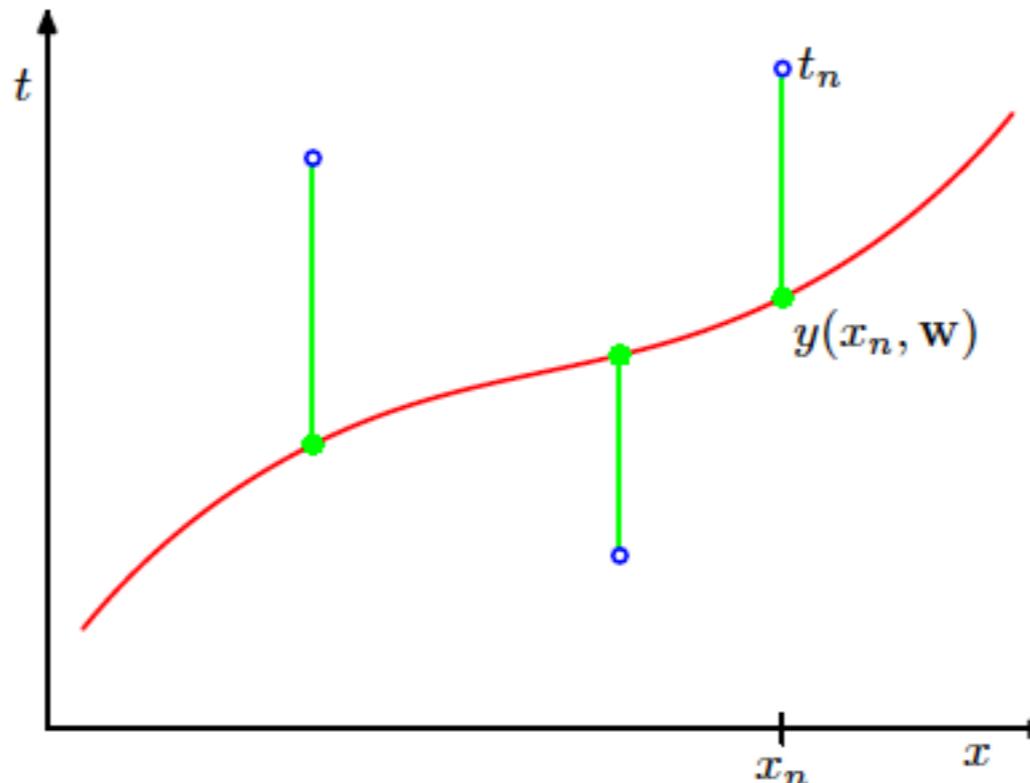


# Polynomial Regression - Parameter estimation

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$

- Coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

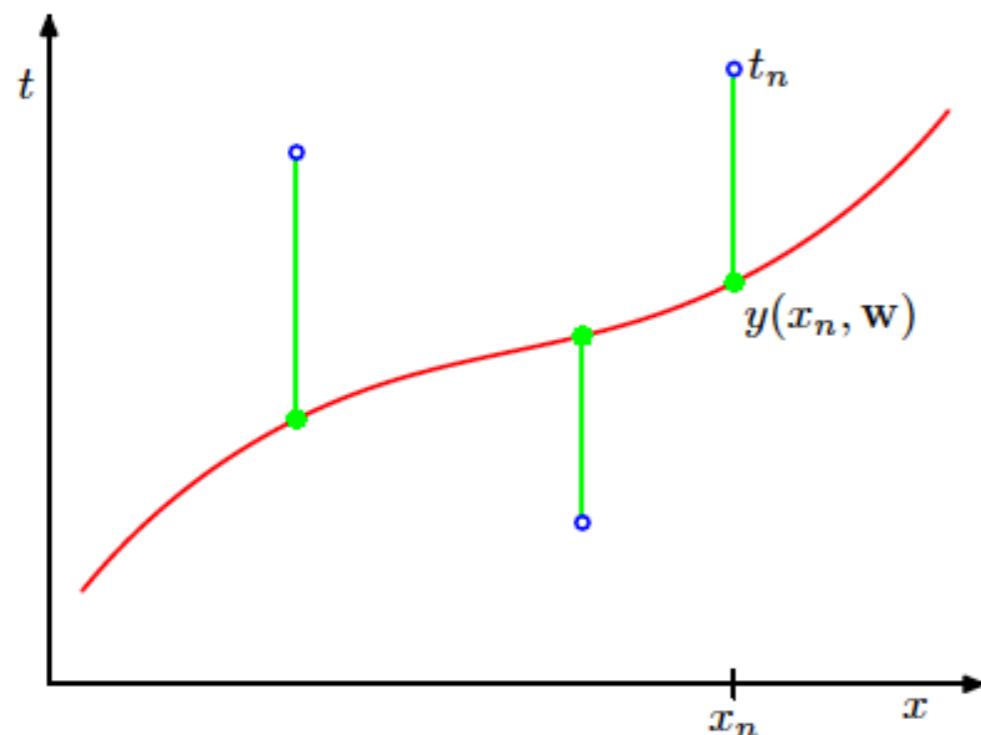


# Polynomial Regression - Parameter estimation

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_{M-1} x^{M-1} = \sum_{j=0}^{M-1} w_j x^j$$

- Coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

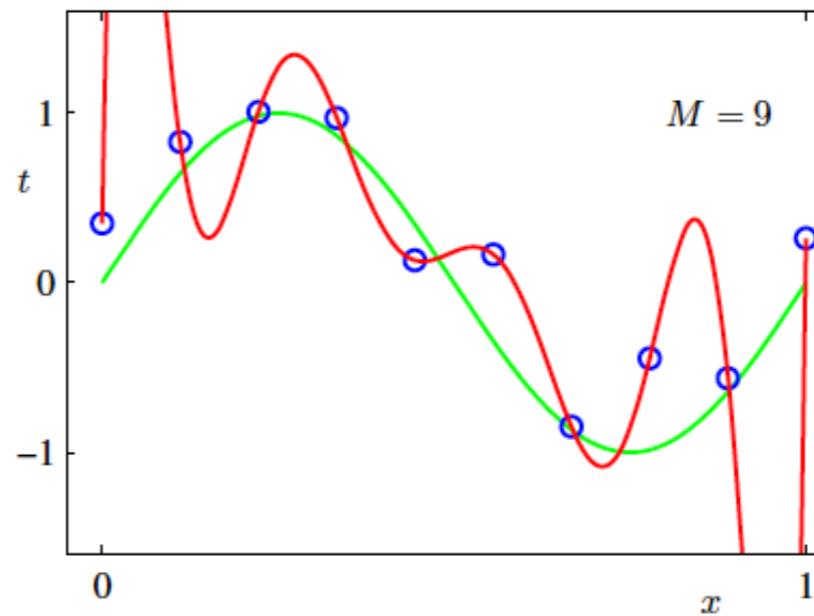
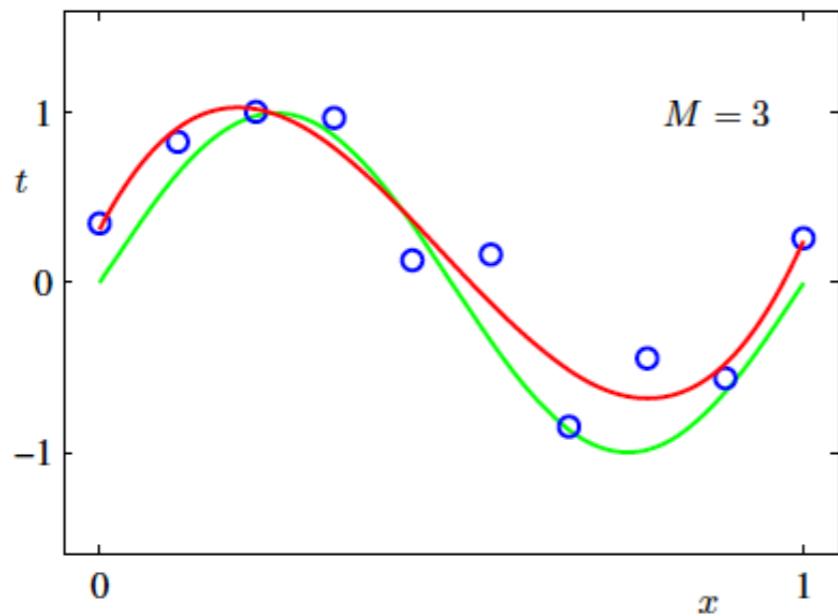
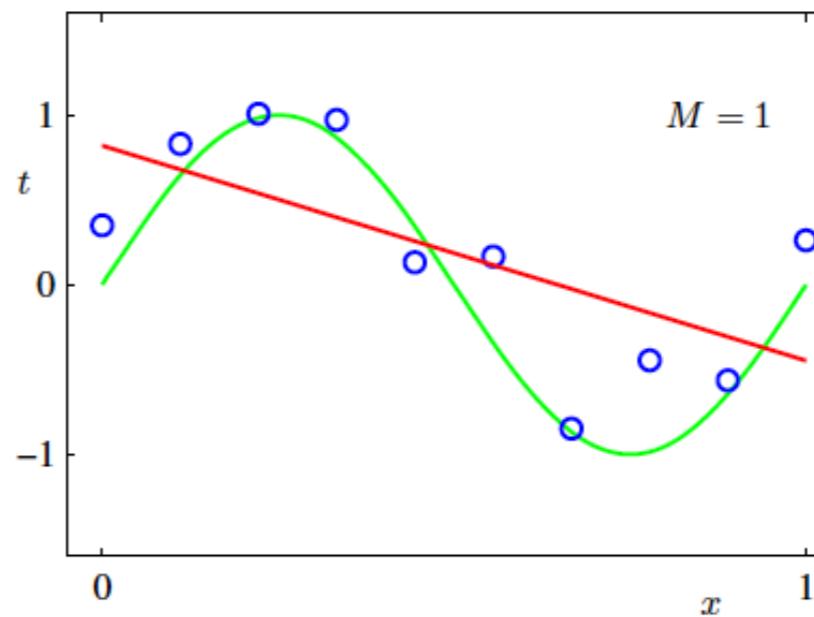
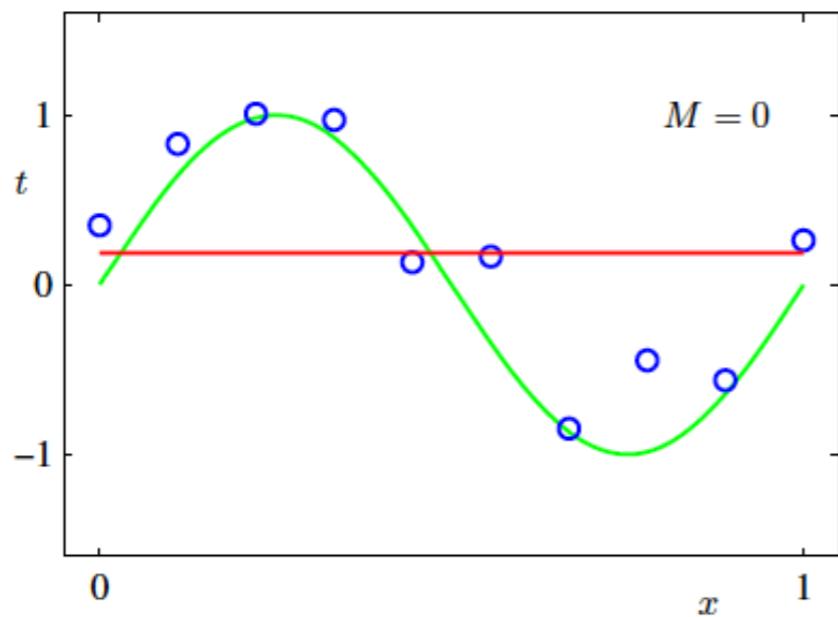


$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}_{N \times M}$$

$$\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

# Polynomial regression – Model Selection

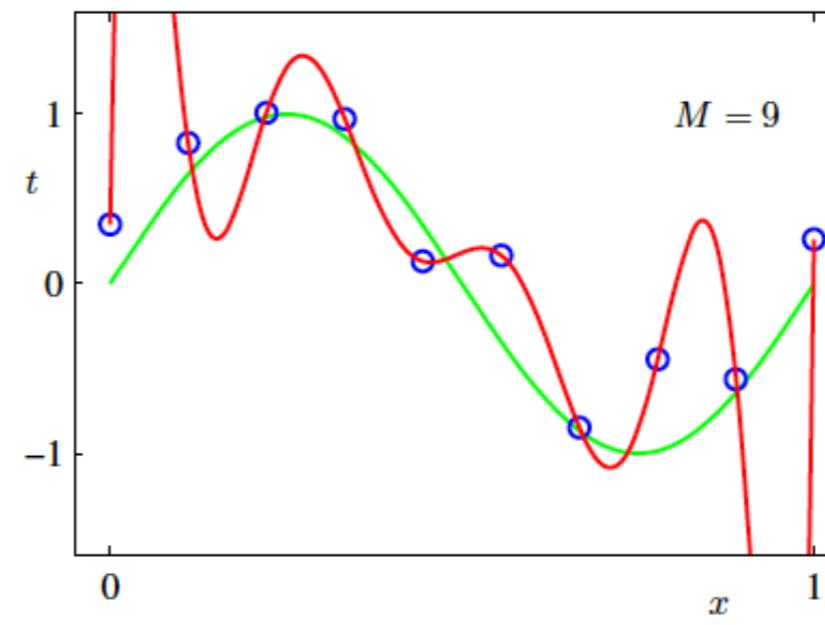
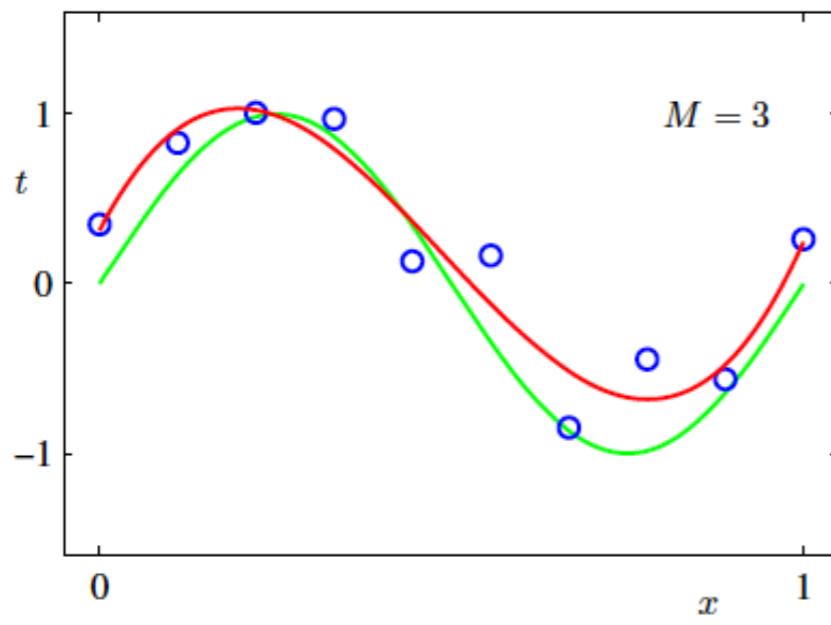
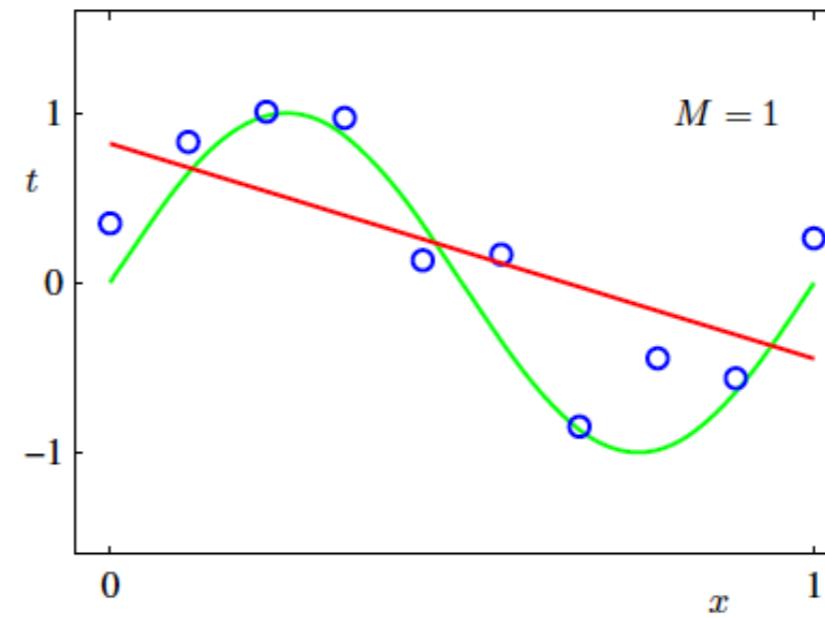
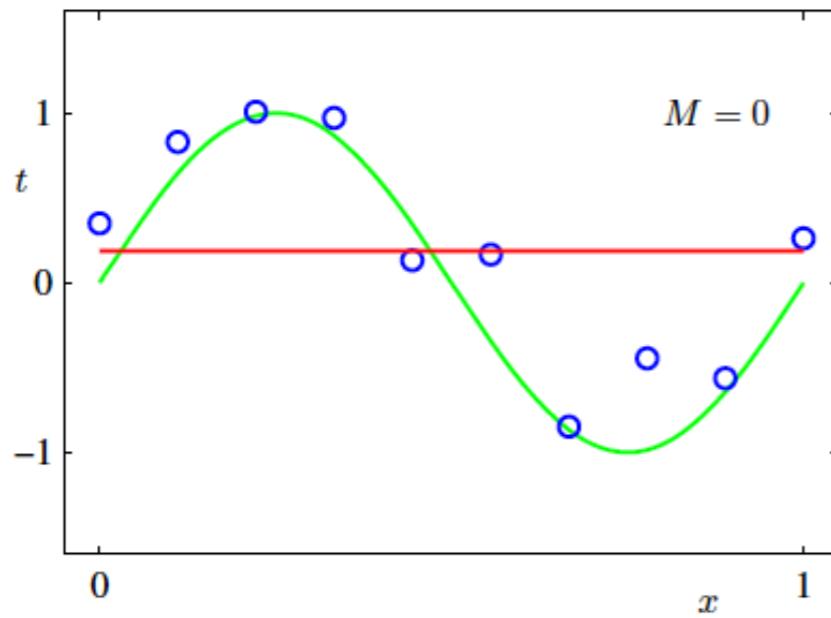
- Model selection (choosing  $M$ ) : higher order polynomial ( $M = 9$ ), provide excellent fit to the training data but gives a very poor representation of the function



# Polynomial regression

## Overfitting

- Model selection (choosing  $M$ ) : high-degree polynomial ( $M = 9$ ), provide excellent fit to the training data, gives a very poor representation of the function

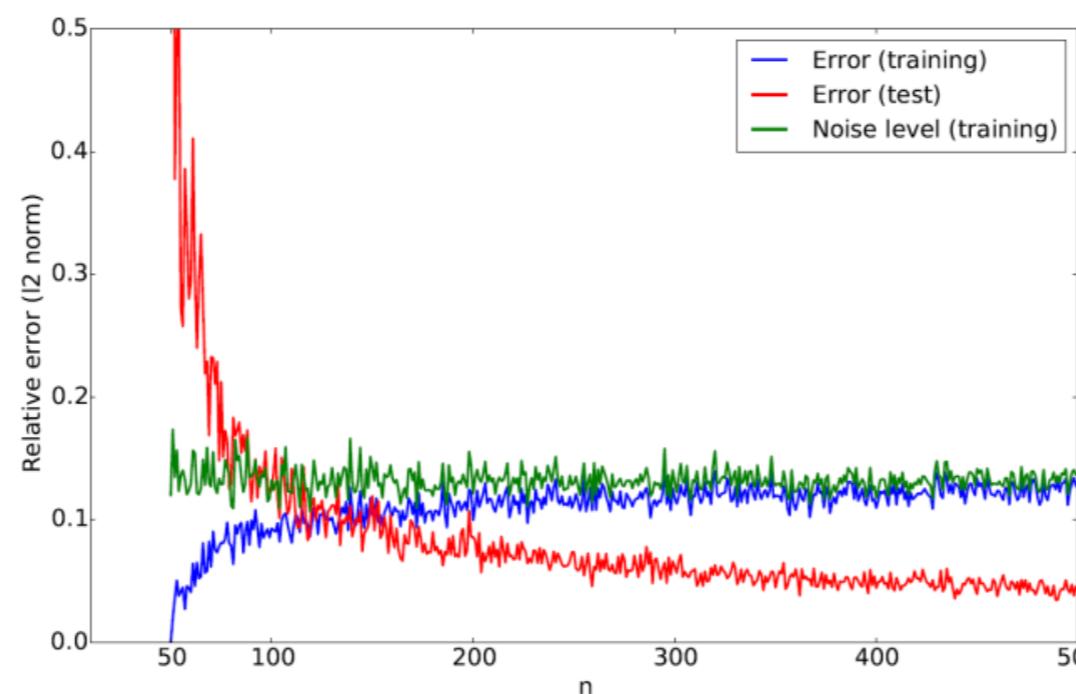
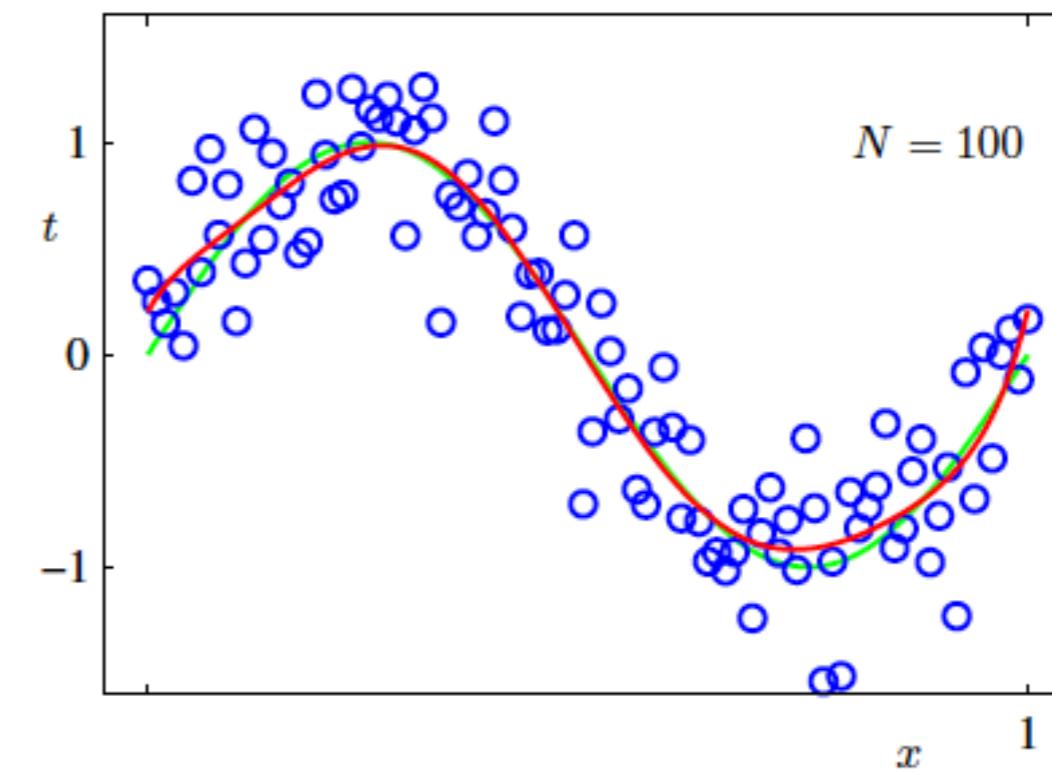
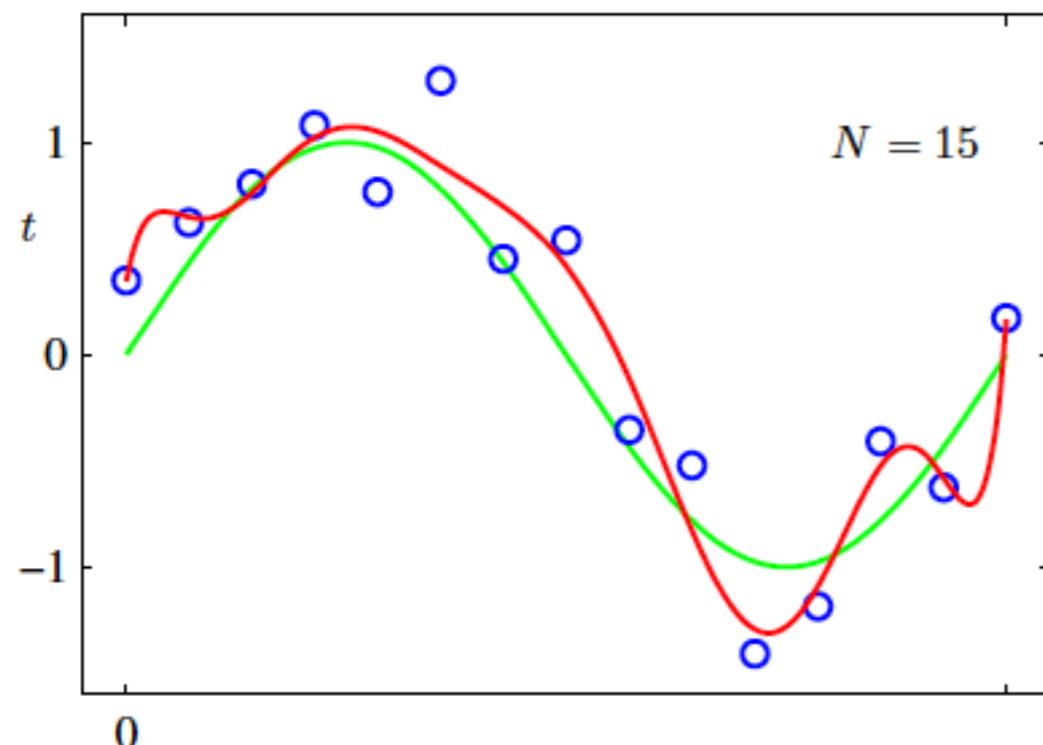


model that is too flexible with respect to the number of data



# Polynomial regression - Model Selection

- Given model complexity, the over-fitting problem become less severe as the size of the data set increases.



# Polynomial regression - regularization

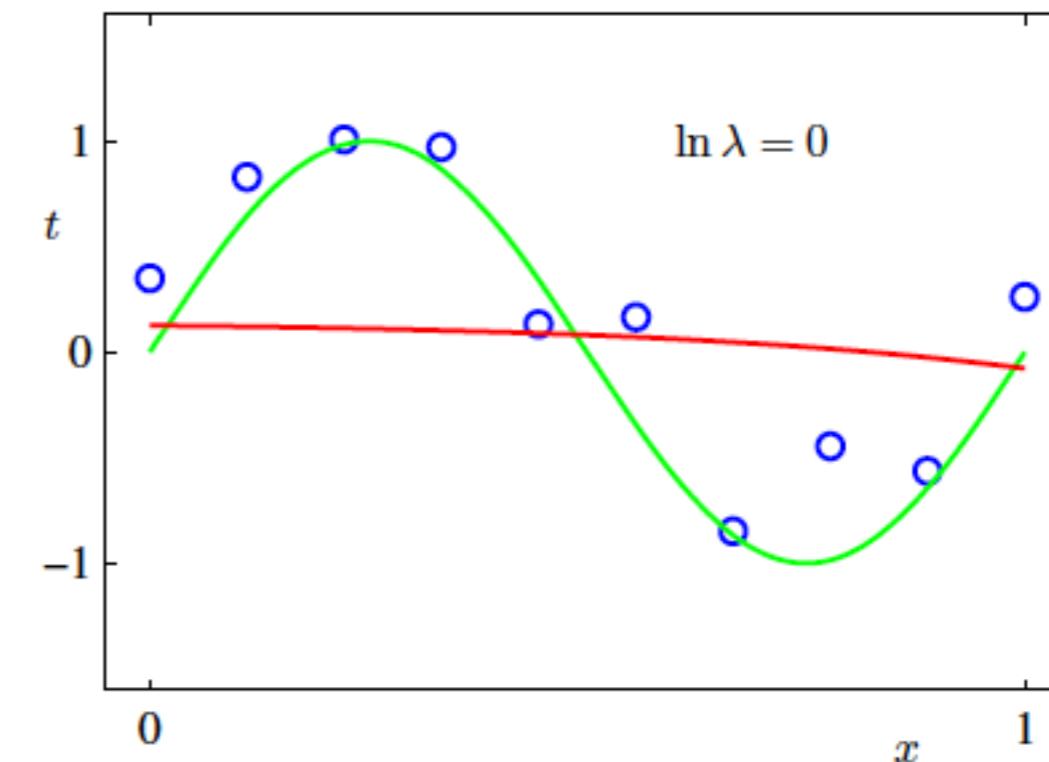
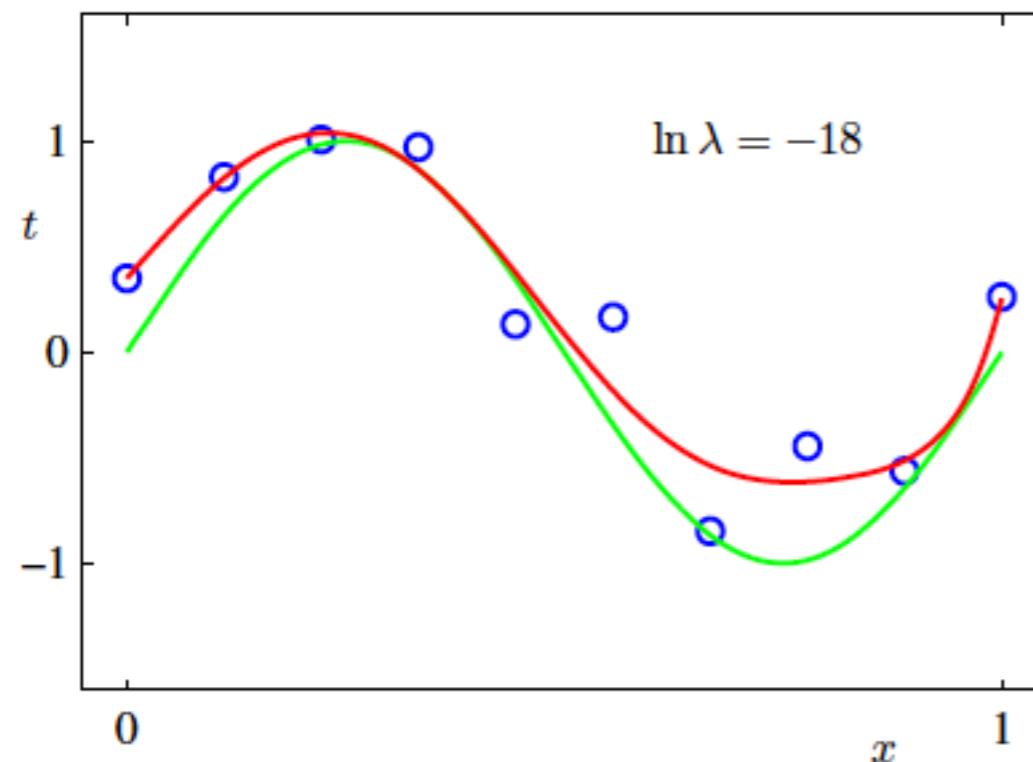
- Add a penalty term to the error function (1.2) in order to discourage the coefficients from reaching large values

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

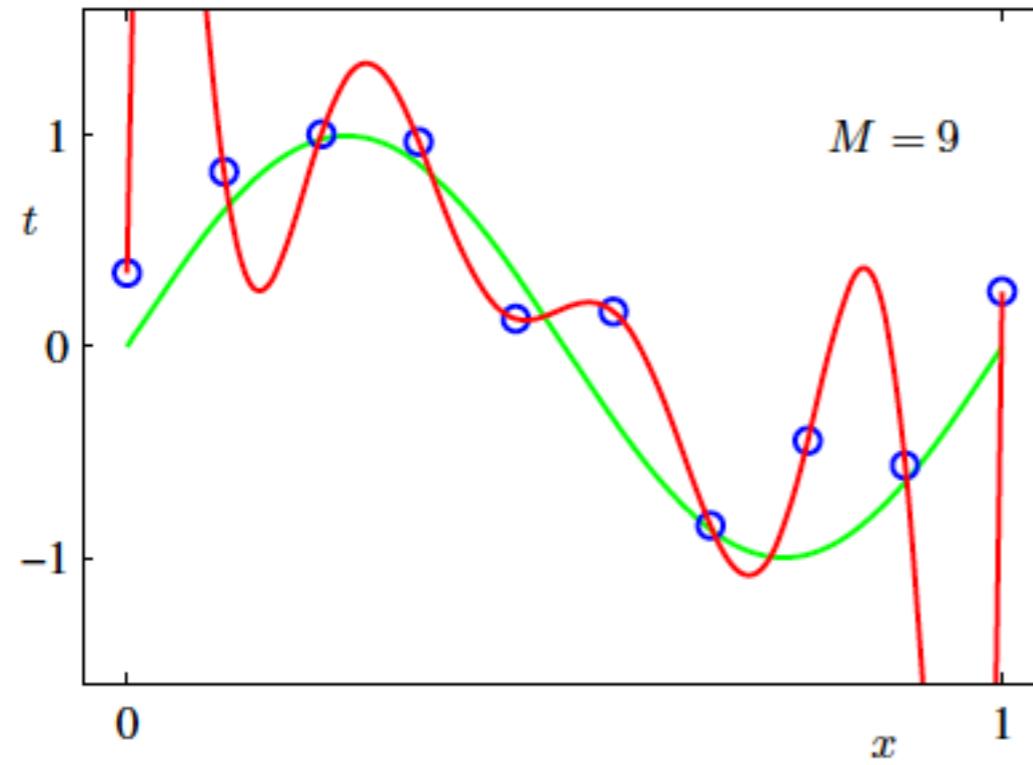
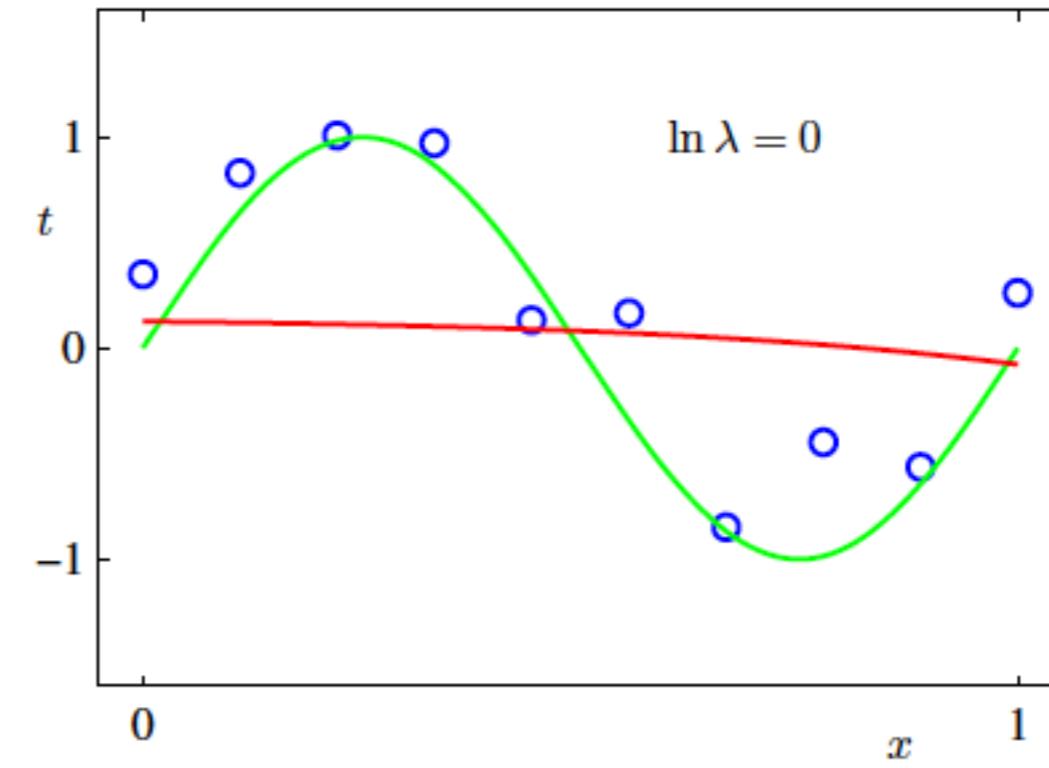
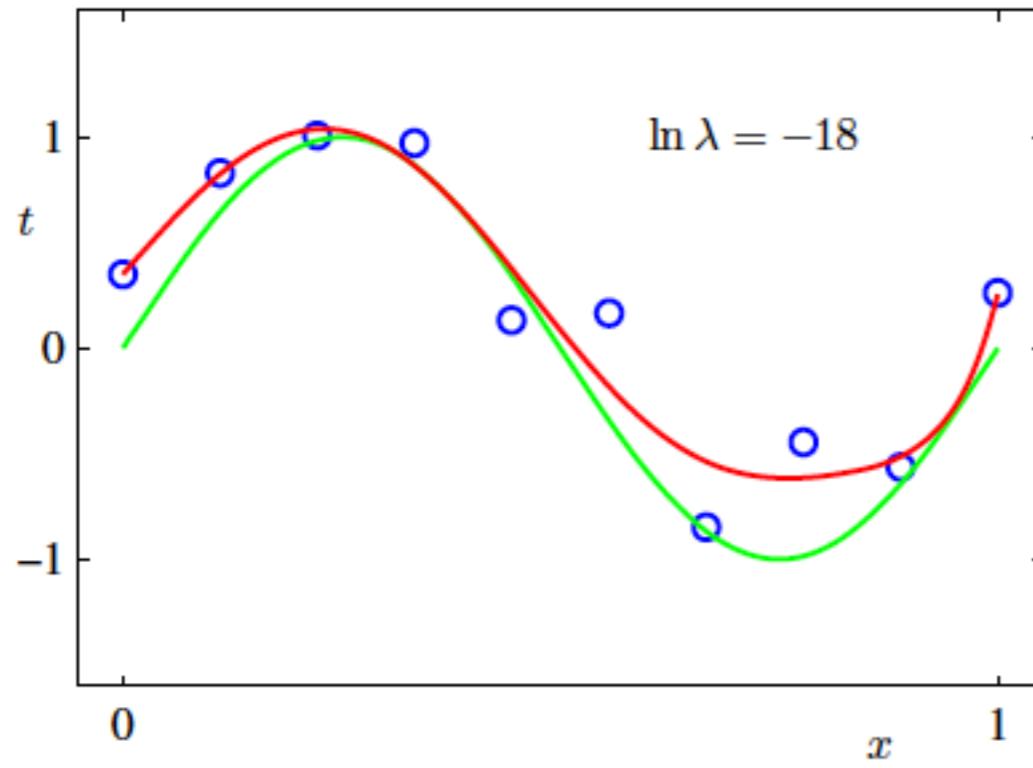
- Ridge regression : L2 norm

$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \dots + w_M^2$$

Regularization constant



# Polynomial regression - regularization



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

# Regularized Least Squares

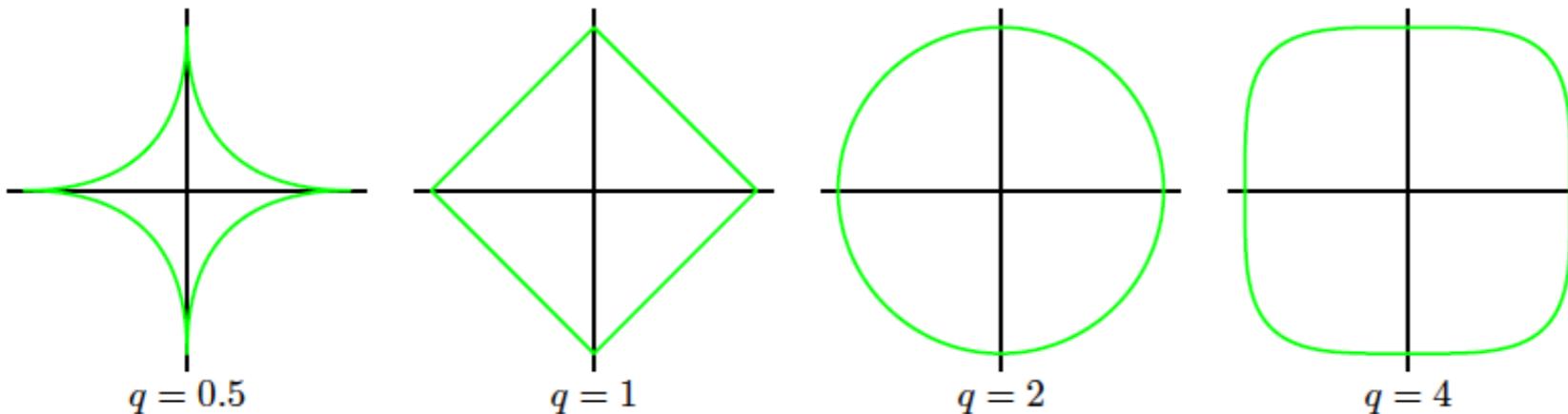
- Parameter shrinkage, weight decay
- **Ridge regression**  $q=2$
- **Lasso regression**  $q=1$ , if  $\lambda$  is sufficiently large, some of the coefficients are driven to zero
- Elastic net regularization

$$\frac{\lambda}{2} \sum_{j=1}^M w_j^2$$

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|$$

$$\frac{1}{n} \|Y - Xw\|_2^2 + \lambda_1 \sum_{j=1}^d |w_j| + \lambda_2 \sum_{j=1}^d |w_j|^2$$



# Regularized Least Squares

- Parameter shrinkage, weight decay

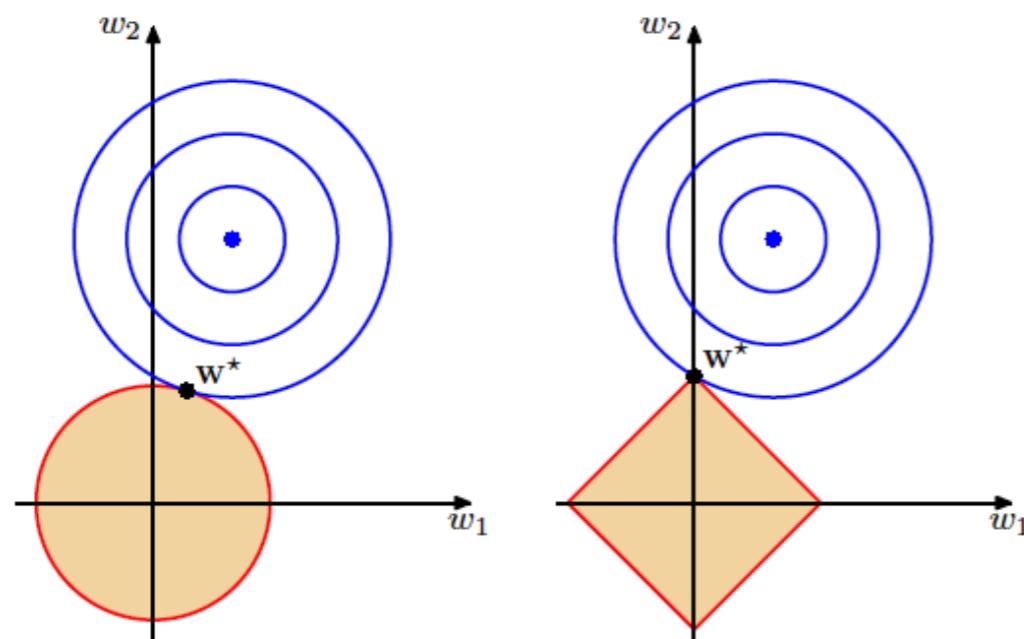
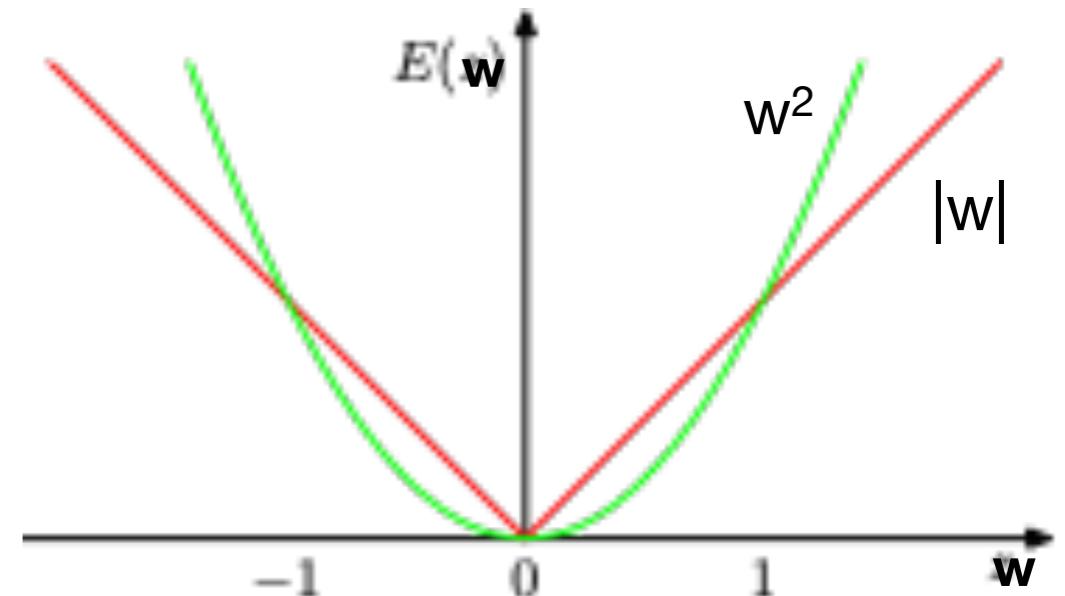
- **Ridge regression**  $q=2$

$$\frac{\lambda}{2} \sum_{j=1}^M w_j^2$$

- **Lasso regression**  $q=1$ , if  $\lambda$  is sufficiently large, some of the coefficients are driven to zero

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|$$

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$



# Ridge Regression

- Regularized Least Squares

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}.$$

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}_{N \times M}$$

- Show that the regularized least squares solution is

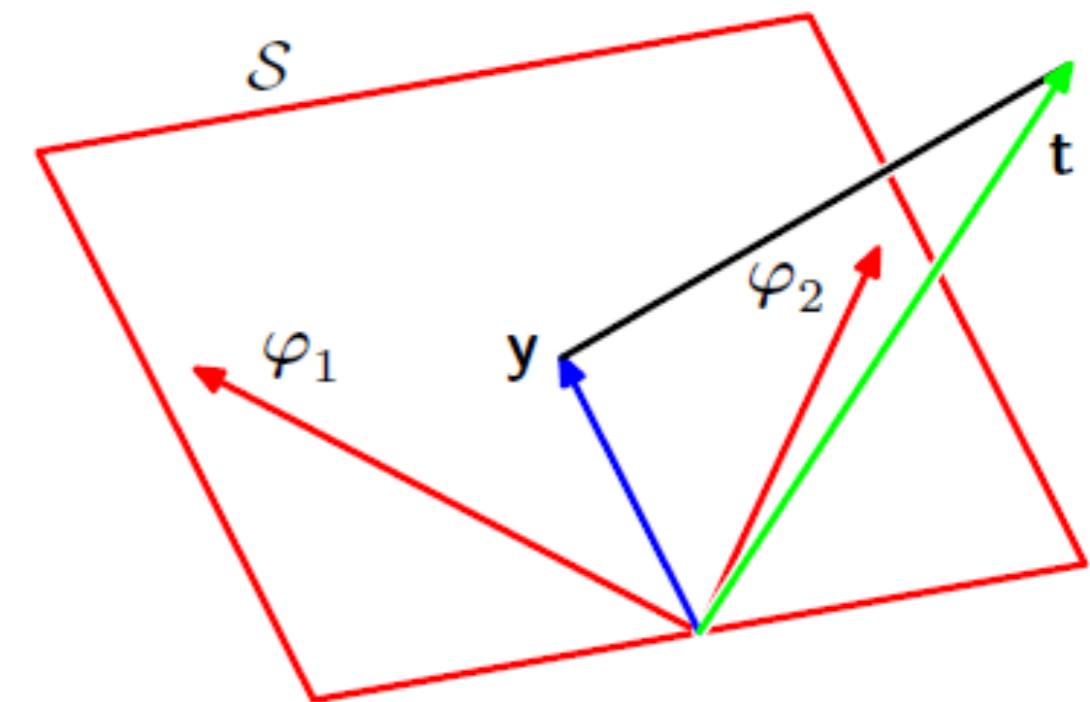
$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}.$$

Stable and unique solution

# Least Squares Geometric Interpretation

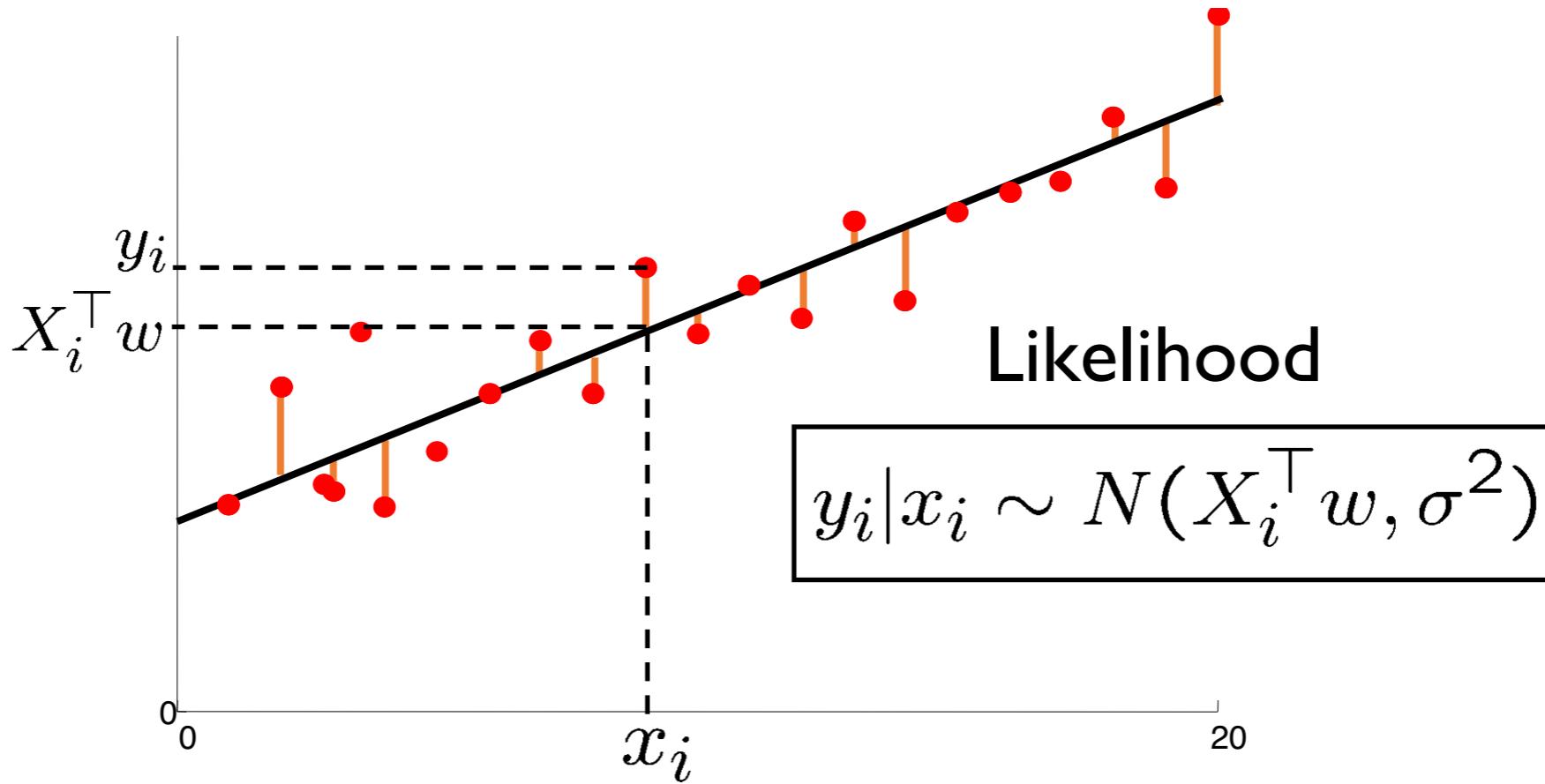
$$\Phi = \begin{pmatrix} & & & \varphi_j \\ \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}_{N \times M}$$
$$\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$
$$\Phi \mathbf{w} = \underbrace{\Phi (\Phi^\top \Phi)^{-1} \Phi^\top}_{\text{Orthogonal Projection}} \mathbf{t}$$

Geometrical interpretation of the least-squares solution, in an  $N$ -dimensional space whose axes are the values of  $t_1, \dots, t_N$ . The least-squares regression function is obtained by finding the orthogonal projection of the data vector  $\mathbf{t}$  onto the subspace spanned by the basis functions  $\phi_j(x)$  in which each basis function is viewed as a vector  $\varphi_j$  of length  $N$  with elements  $\phi_j(x_n)$ .



# Probabilistic Interpretation : Least Squares = Maximum likelihood estimation

$$y_i \sim w \cdot x_i + N(0, \sigma^2) = N(w \cdot x_i, \sigma^2)$$



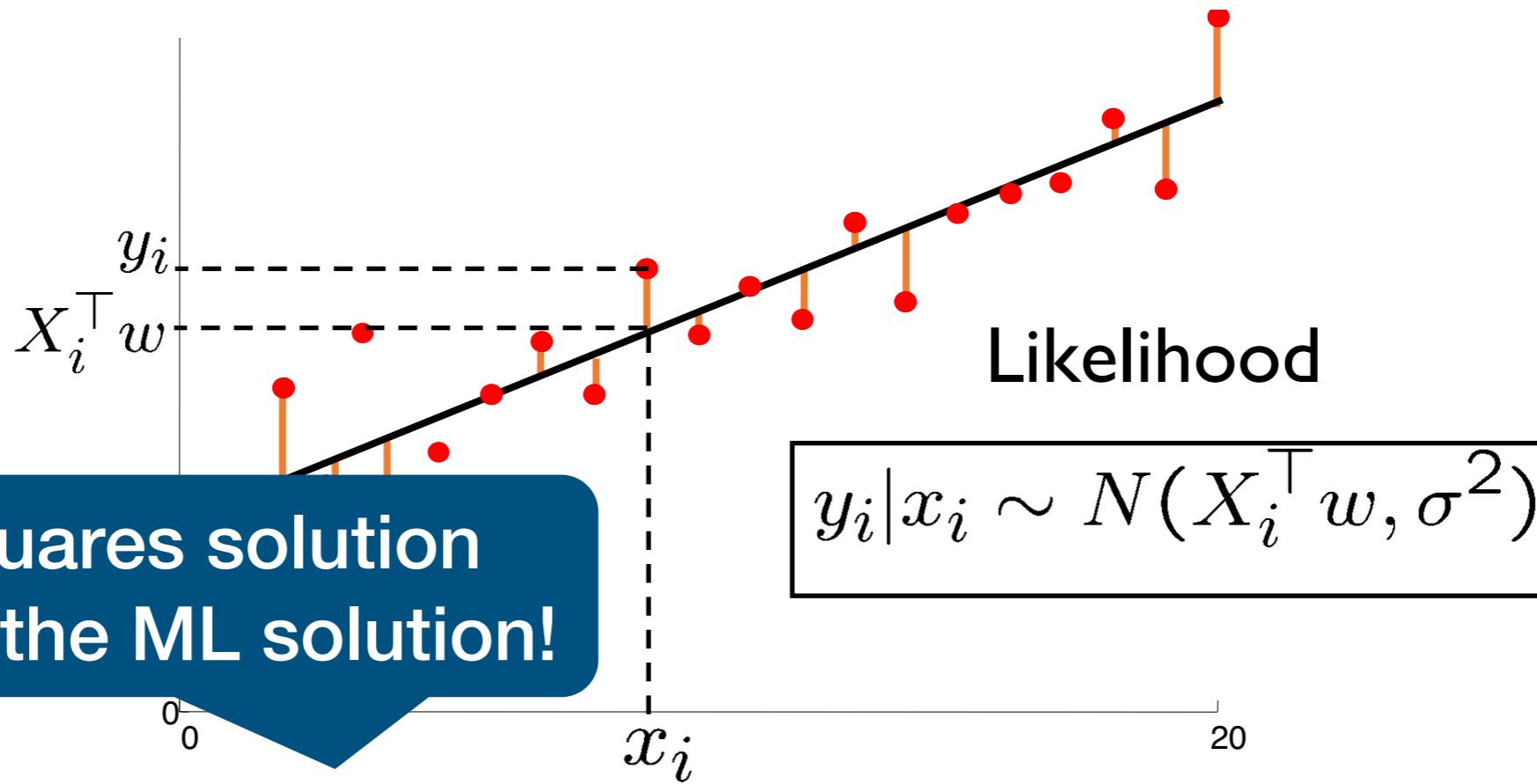
$$L = \prod_i \exp -\frac{1}{2\sigma^2} (X_i^\top w - y_i)^2 = \exp -\frac{1}{2\sigma^2} \sum_i (X_i^\top w - y_i)^2$$

$$\underset{w}{\text{argmax}} L = \underset{w}{\text{argmin}} E$$

- Similarly Regularized least squares is same as maximum a posteriori estimate assuming  $p(w)$  to be a Gaussian :  $\underset{w}{\text{argmax}} p(y_i | w, x_i) p(w)$

# Probabilistic Interpretation : Least Squares = Maximum likelihood estimation

$$y_i \sim w \cdot x_i + N(0, \sigma^2) = N(w \cdot x_i, \sigma^2)$$



$$L = \prod_i \exp -\frac{1}{2\sigma^2} (X_i^T w - y_i)^2 = \exp -\frac{1}{2\sigma^2} \sum_i (X_i^T w - y_i)^2$$

$$\underset{w}{\operatorname{argmax}} L = \underset{w}{\operatorname{argmin}} E$$

- Similarly Regularized least squares is same as maximum a posteriori estimate assuming  $p(w)$  to be a Gaussian :  $\underset{w}{\operatorname{argmax}} p(y_i | w, x_i) p(w)$

# Regularized least squares regression = Maximum Aposteriori Estimate

- Ridge Regression

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Compute Maximum aposteriori (MAP) estimate

- Prior over parameters  $p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2} \mathbf{w}^T \mathbf{w}\right\}$

- Posterior

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) p(\mathbf{w}|\alpha).$$

- MAP estimate

$$\frac{\beta}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w}.$$

Unique Solution

$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}. \quad \lambda = \alpha/\beta.$$

# Model Selection

- Limiting the number of basis functions in order to avoid over-fitting has the side effect of limiting the flexibility of the model to capture interesting and important trends in the data.
- Regularization allows complex models to be trained on data sets of limited size without severe over-fitting, essentially by limiting the effective model complexity.
- Model Selection : determining a suitable value of the hyper-parameter which is the regularization coefficient  $\lambda$ .
- What happens when you minimizes the regularized error function with respect to weight vector  $w$  and regularization coefficient  $\lambda$  ?

# Regularized Least Squares : Cross Validation

How to choose  $\lambda$ ?  
Use validation data!

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

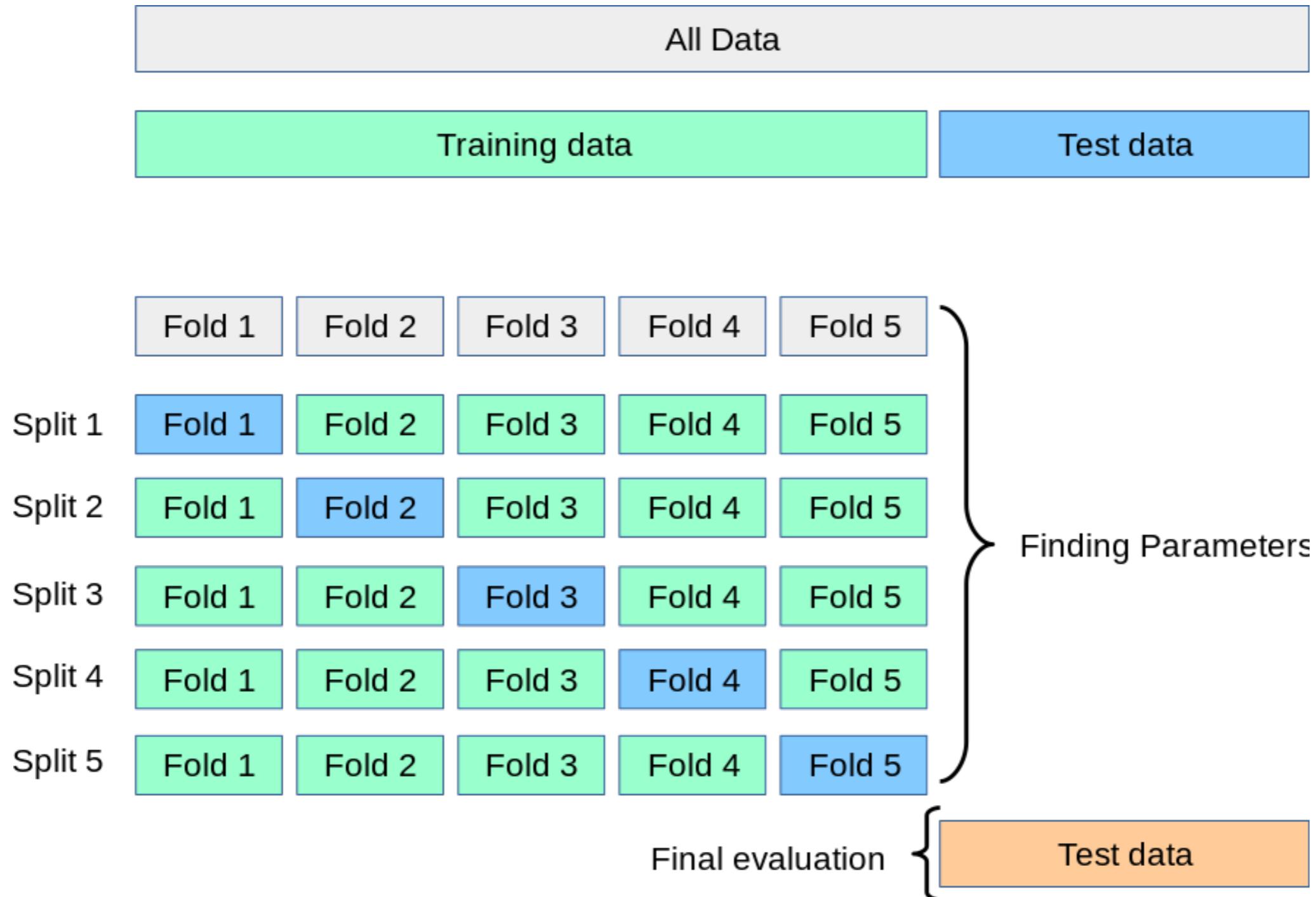


1. **Training set** is a set of examples used for learning a model parameters (e.g., weight vector  $w$  in linear regression)
2. **Validation set** is a set of examples that cannot be used for learning the model parameter but can help tune model hyper-parameters e.g Regularization constant in LR. Validation helps control overfitting.
3. **Test set** is used to assess the performance of the final model and provide an estimation of the test error.

Example: Split the data randomly into 60% for training, 20% for validation and 20% for testing.

**Note: Dont use the test set to further tune the parameters or revise the model.**

# Cross Validation



# Linear Regression

- Parameter Estimation
  - Maximum Likelihood
  - Maximum Aposteriori
- Hyper-parameter Estimation
  - Cross-validation
- Evaluation : Root mean square error, absolute error

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

# Training vs Generalization Error

- Training Error
  - Not very useful
  - Relatively easy to obtain low error

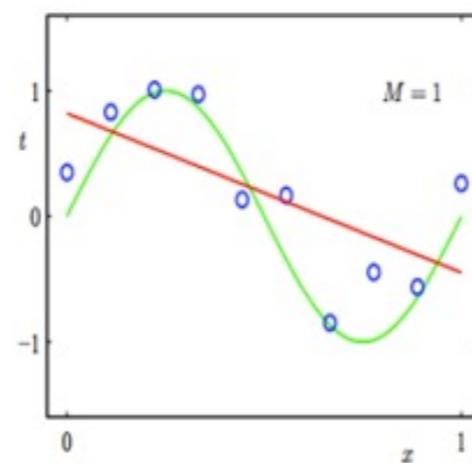
$$E_{train} = \frac{1}{n} \sum_{i=1}^n \text{error}(f_D(\mathbf{x}_i), y_i)$$

same? different by how much?  
training examples      value we predicted      true value

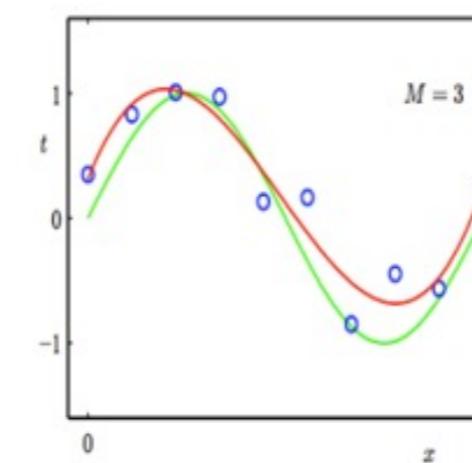
- Generalization Error
  - How well we do on future data

How to compute generalization error?

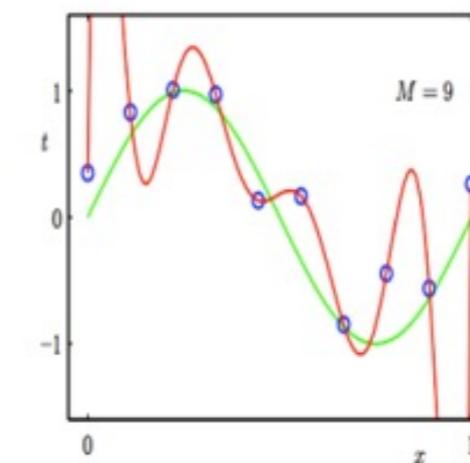
Regression



predictor too inflexible:  
cannot capture pattern



Not responding for data  
properly i.e inflexible



predictor too flexible:  
fits noise in the data

# Estimating Generalization Error

- Testing Error

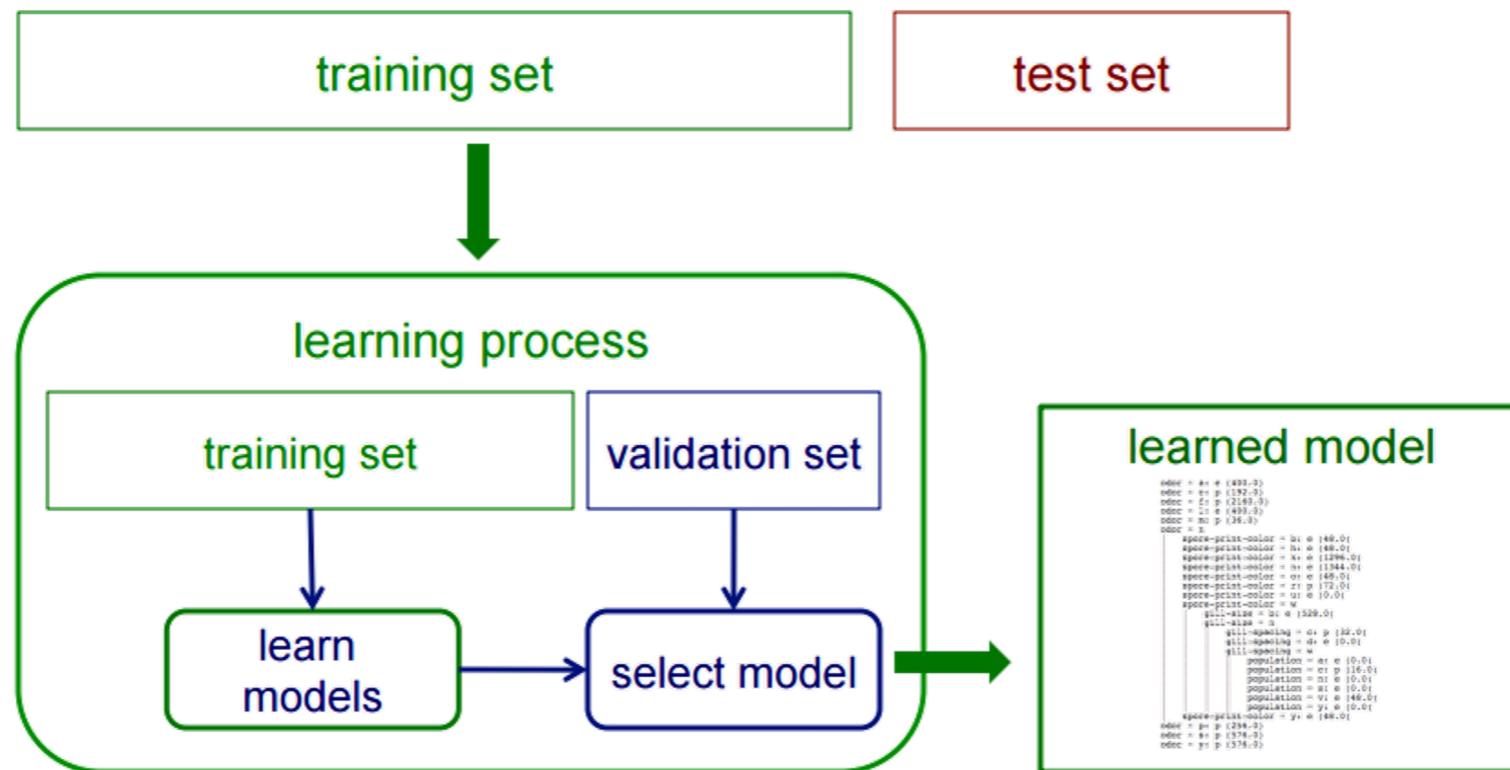
- Set aside part of training data (testing set)
- Learn a predictor without using any of this test data
- Predict values for testing set, compute error
- This is an estimate of generalization error

$$E_{test} = \frac{1}{n} \sum_{i=1}^n \text{error}(f_D(\mathbf{x}_i), y_i)$$

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

# Use of Validation and Test Sets

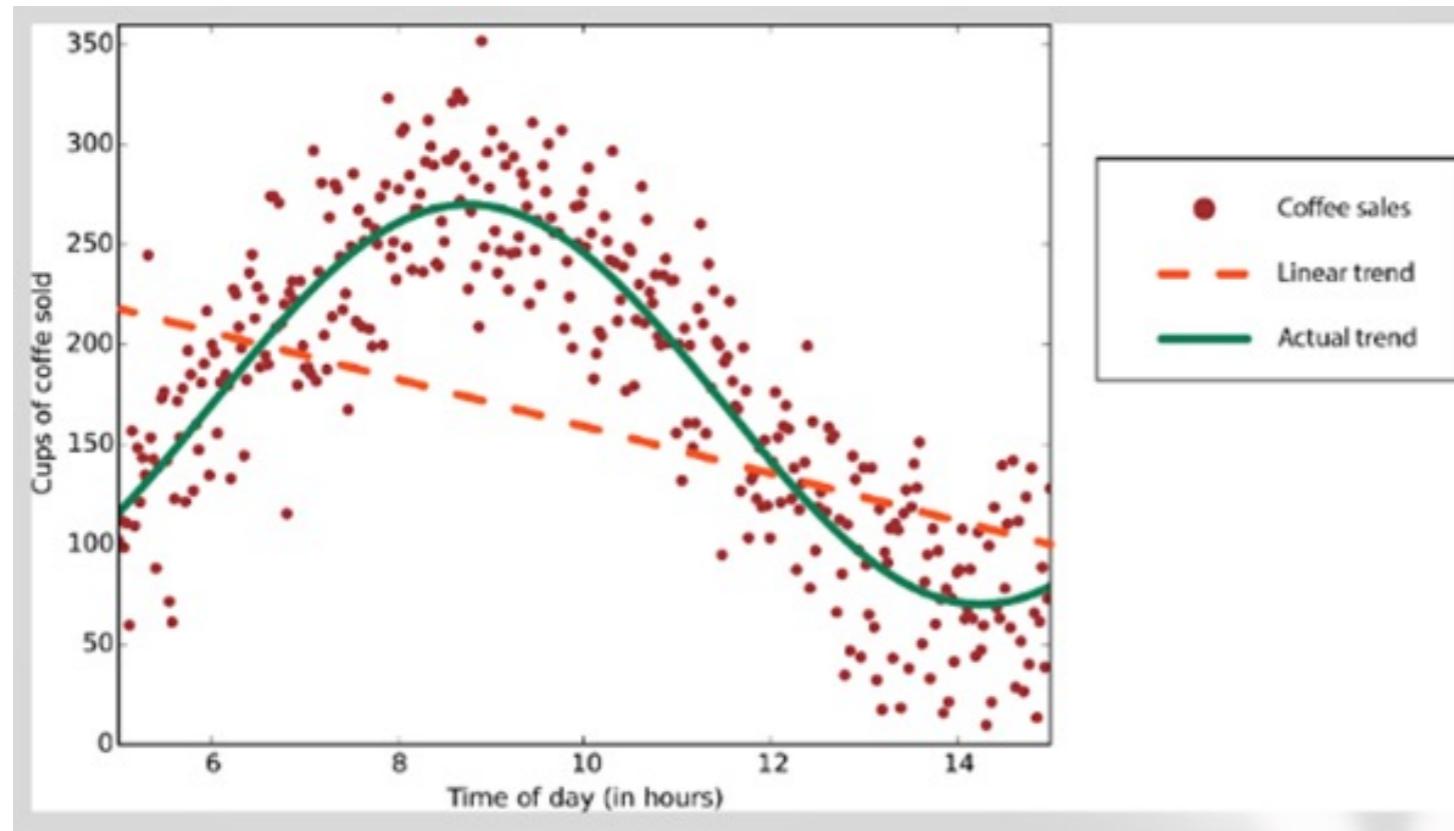
- If we want unbiased estimates of accuracy during the learning process:



# Linear Regression

- Parameter Estimation
    - Maximum Likelihood
    - Maximum Aposteriori
  - Hyper-parameter Estimation
    - Cross-validation
  - Prediction
- $$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

# Supervised learning : Regression



Number of vehicles passing a junction

$p(y|x)$ ?

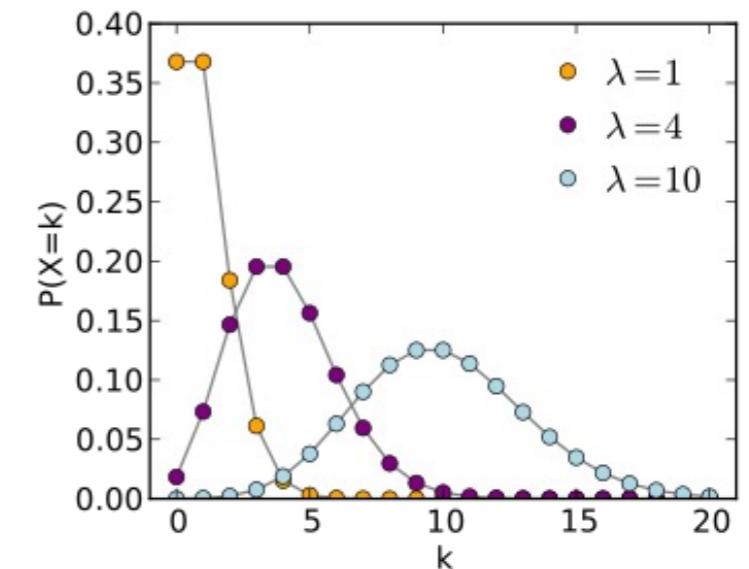
Y take values 0,1,2,3,..... but not 2.1, 3.4, 5.55.....

# Poisson Regression

- Poisson distribution : Model number of events occurring in a fixed interval of time/space

$$P(k \text{ events in interval}) = e^{-\lambda} \frac{\lambda^k}{k!}$$

- $\lambda$  is the average (mean) number of events
- $Y$  has a Poisson distribution, and assumes the logarithm of its expected value can be modeled by a linear combination of unknown parameters.



$$\lambda := E(Y | x) = e^{\theta' x},$$

$$p(y | x; \theta) = \frac{\lambda^y}{y!} e^{-\lambda} = \frac{e^{y\theta' x} e^{-e^{\theta' x}}}{y!}$$

# Poisson Regression : Learning parameters

- Likelihood

$$p(y_1, \dots, y_m \mid x_1, \dots, x_m; \theta) = \prod_{i=1}^m \frac{e^{y_i \theta' x_i} e^{-e^{\theta' x_i}}}{y_i!}.$$

- Estimate parameters by maximum likelihood estimation

$$\ell(\theta \mid X, Y) = \log L(\theta \mid X, Y) = \sum_{i=1}^m \left( y_i \theta' x_i - e^{\theta' x_i} - \log(y_i!) \right).$$

- Use gradient descent to find the optimal value of  $\theta$ .

# Bias Variance Decomposition

Given the actual conditional distribution  $p(t|x)$ , and  $p(x,t) = p(t|x)p(x)$ , let the optimal function one can learn is given by  $h(x)$

For any given data set  $D$ , we can run our learning algorithm and obtain a prediction function  $y(x;D)$ .

Expected squared difference between  $y(x;D)$  and the regression function  $h(x)$ , averaged across multiple data sets

$$\begin{aligned} & \mathbb{E}_D [ \{y(x; D) - h(x)\}^2 ] \\ &= \underbrace{\{\mathbb{E}_D[y(x; D)] - h(x)\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_D [ \{y(x; D) - \mathbb{E}_D[y(x; D)]\}^2 ] }_{\text{variance}}. \end{aligned}$$

$$\begin{aligned} & \{y(x; D) - \mathbb{E}_D[y(x; D)] + \mathbb{E}_D[y(x; D)] - h(x)\}^2 \\ &= \{y(x; D) - \mathbb{E}_D[y(x; D)]\}^2 + \{\mathbb{E}_D[y(x; D)] - h(x)\}^2 \\ &\quad + 2\{y(x; D) - \mathbb{E}_D[y(x; D)]\}\{\mathbb{E}_D[y(x; D)] - h(x)\}. \end{aligned}$$

# Bias- Variance Decomposition

- Bias Error
  - Bias are the simplifying assumptions made by a model to make the target function easier to learn.
  - **Low Bias:** Suggests less assumptions about the form of the target function.
  - **High-Bias:** Suggests more assumptions about the form of the target function.

$$\begin{aligned} & \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \end{aligned}$$

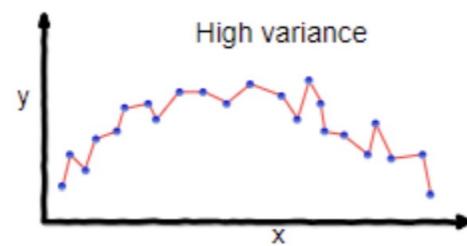
There is a trade-off between bias and variance, with very flexible models having low bias and high variance, and relatively rigid models having high bias and low variance.

# Bias- Variance Decomposition

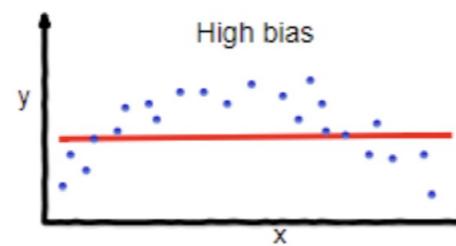
- **Variance Error**

- Variance is the amount that the estimate of the target function will change if different training data was used.
- **Low Variance:** Suggests small changes to the estimate of the target function with changes to the training dataset.
- **High Variance:** Suggests large changes to the estimate of the target function with changes to the training dataset.

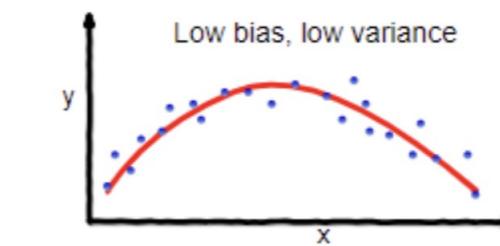
$$\begin{aligned} \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \end{aligned}$$



overfitting



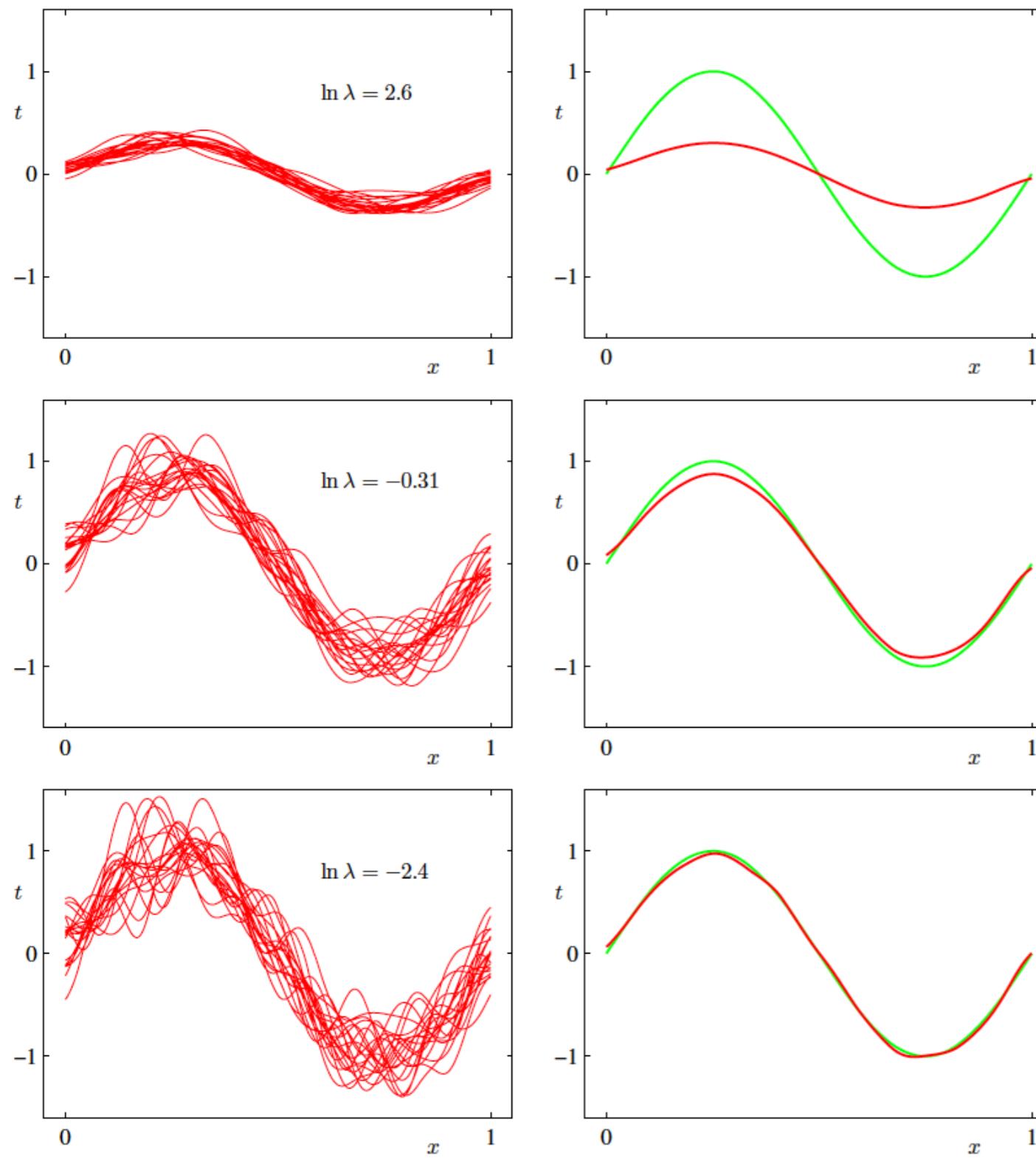
underfitting



Good balance

- Linear machine learning algorithms often have a high bias but a low variance.
- Nonlinear machine learning algorithms often have a low bias but a high variance.

# Bias Variance Decomposition

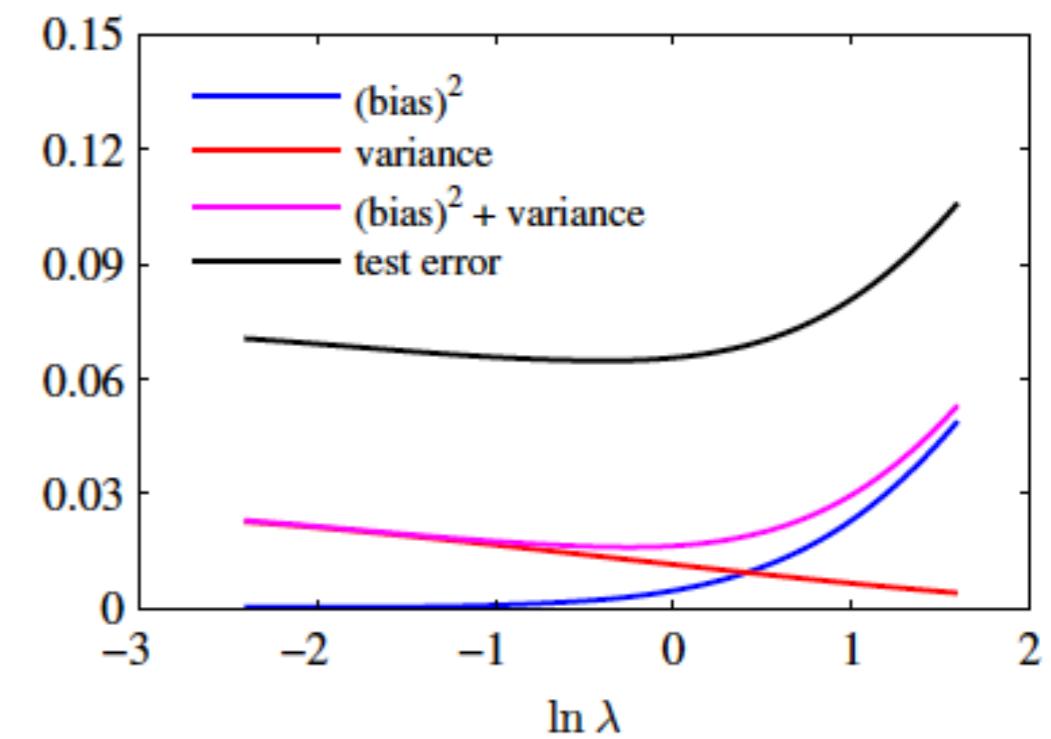


$L = 100$  data sets, each having  $N = 25$  data points,

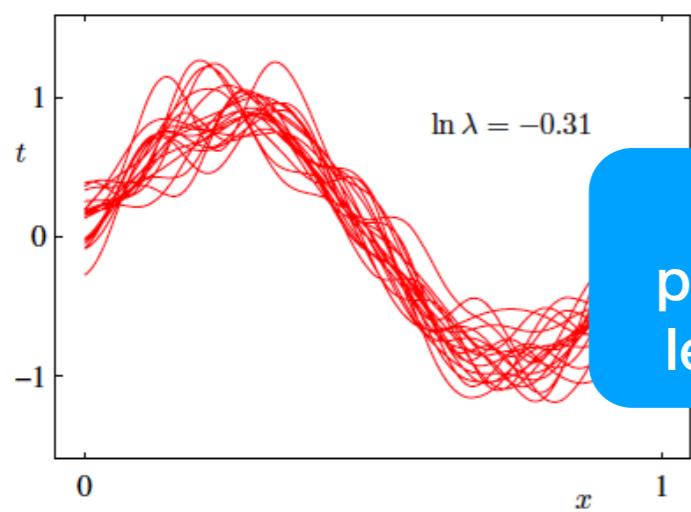
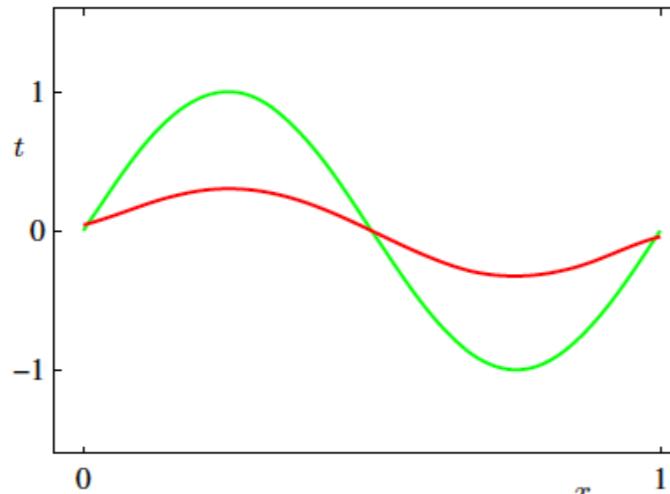
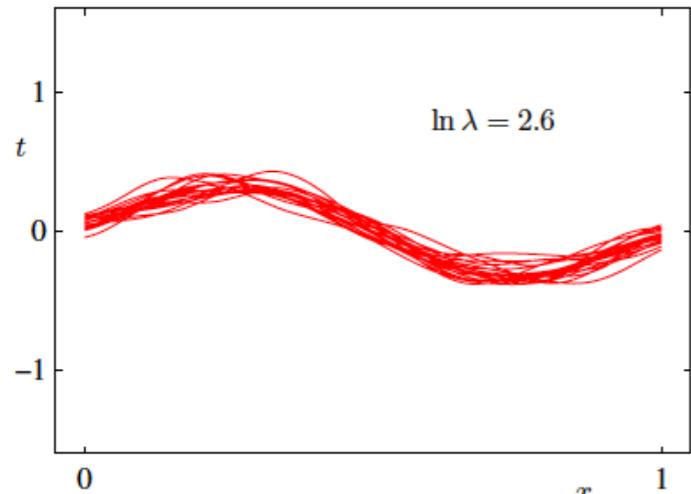
$$h(x) = \sin(2\pi x). \quad \bar{y}(x) = \frac{1}{L} \sum_{l=1}^L y^{(l)}(x)$$

$$\text{(bias)}^2 = \frac{1}{N} \sum_{n=1}^N \{\bar{y}(x_n) - h(x_n)\}^2$$

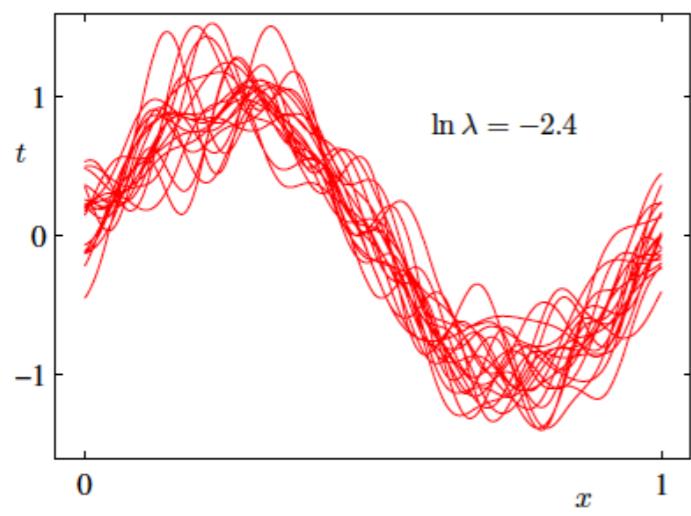
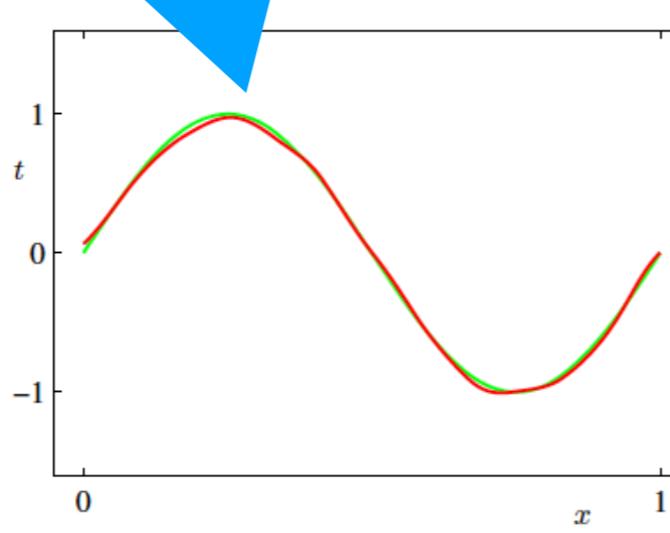
$$\text{variance} = \frac{1}{N} \sum_{n=1}^N \frac{1}{L} \sum_{l=1}^L \{y^{(l)}(x_n) - \bar{y}(x_n)\}^2$$



# Bias Variance Decomposition

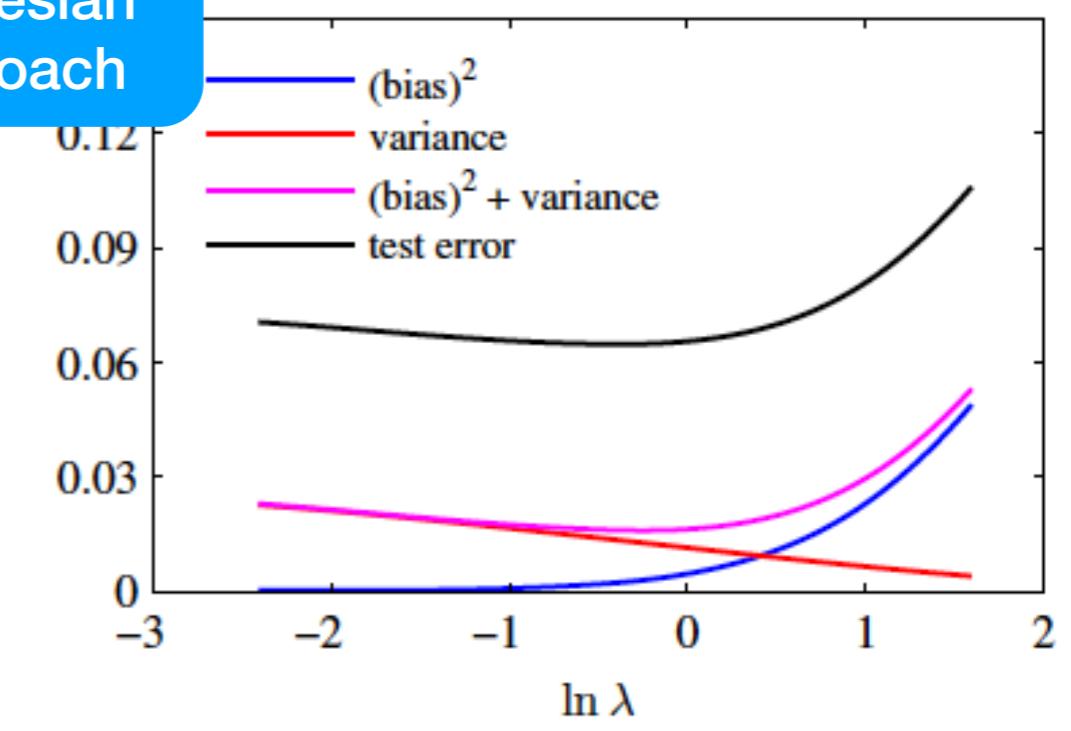


averaging may be a beneficial procedure : Idea behind Bayesian learning and Ensemble approach



$$h(x) = \sin(2\pi x).$$

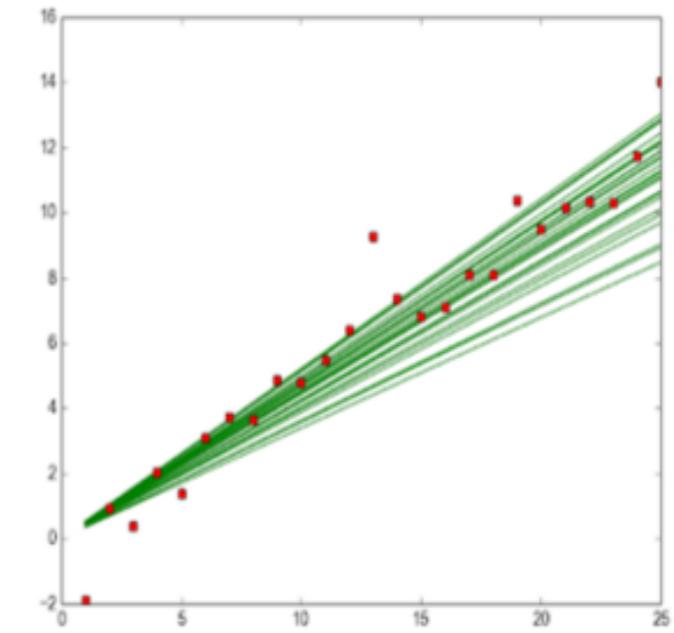
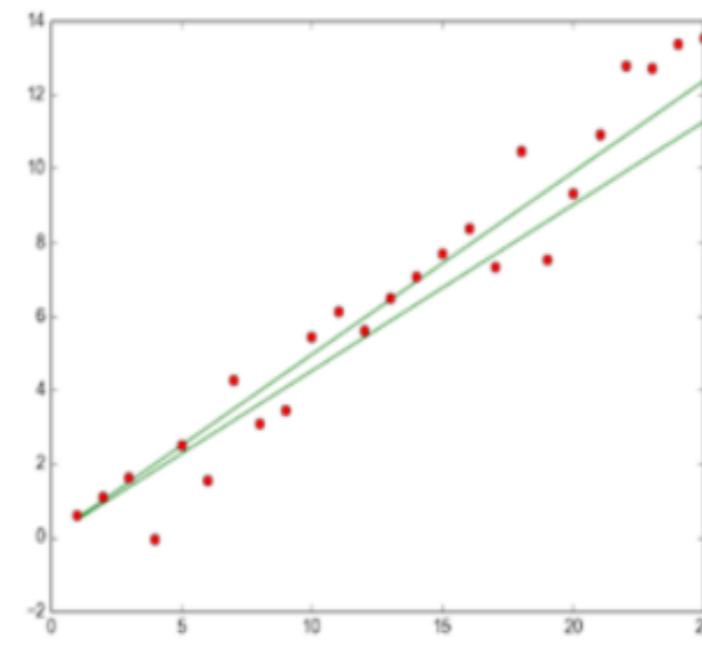
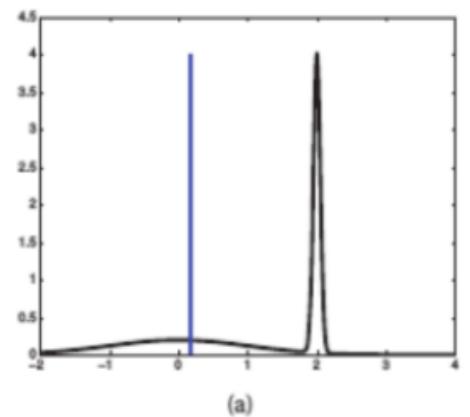
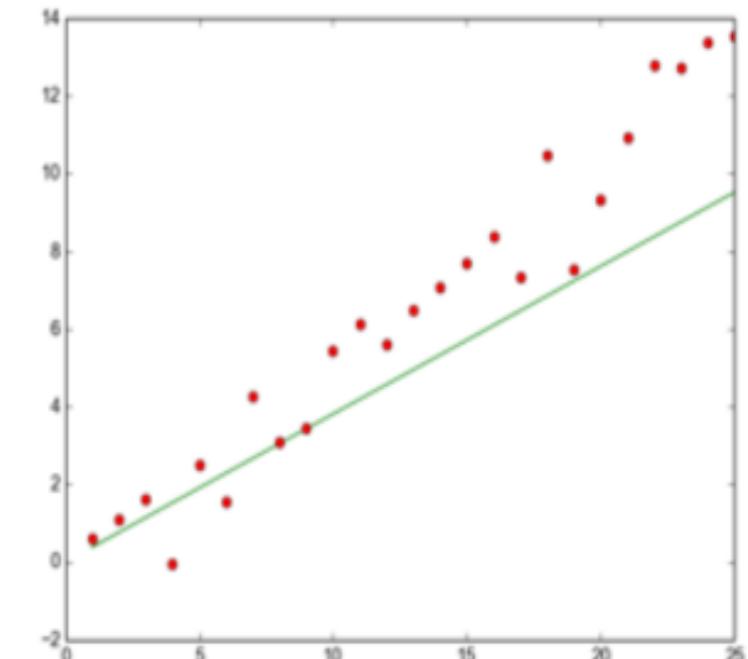
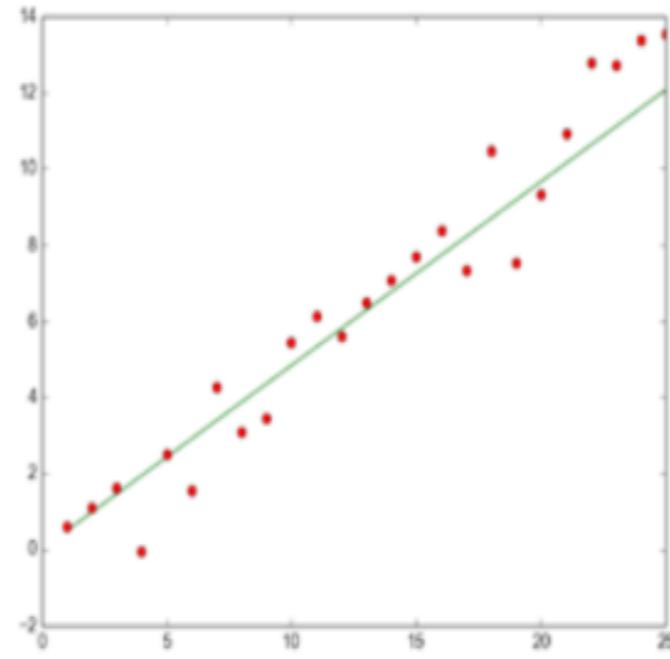
$L = 100$  data sets,  
each having  $N = 25$   
data points,



# Bayesian Linear Regression

$$Prediction : p(y_*|x_*) = \int p(y_*|x_*, w)p(w|D)dw$$

$$\begin{array}{c} p(w|D) \\ \text{Posterior} \end{array} \propto \begin{array}{c} p(D|w) \\ \text{Likelihood} \end{array} \propto \begin{array}{c} p(w) \\ \text{Prior} \end{array}$$

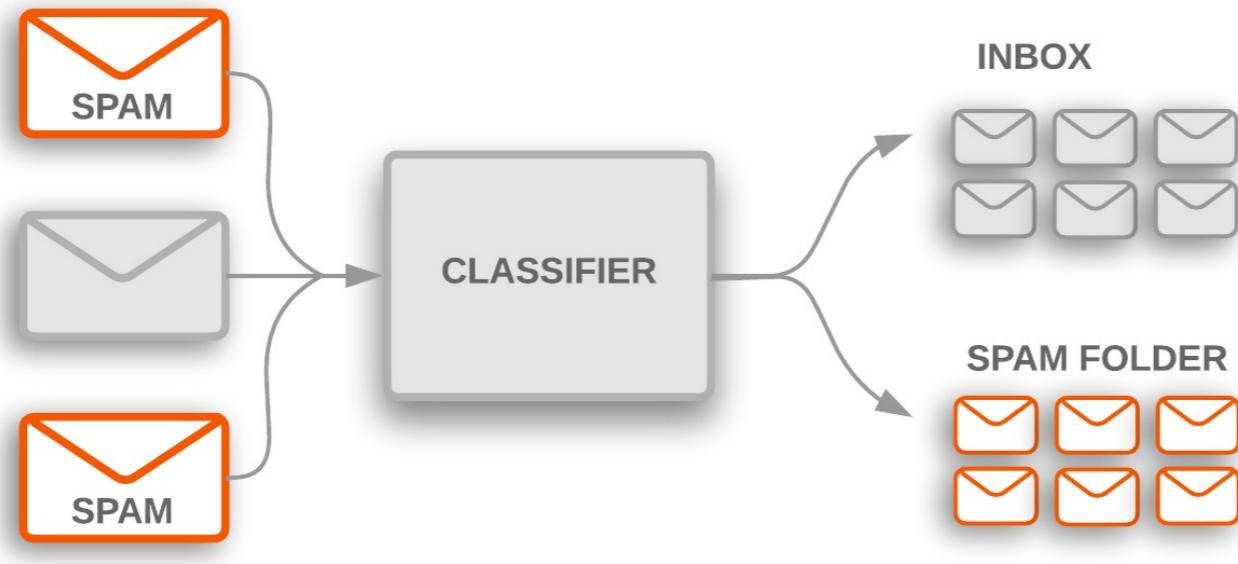


# Bias Variance Tradeoff

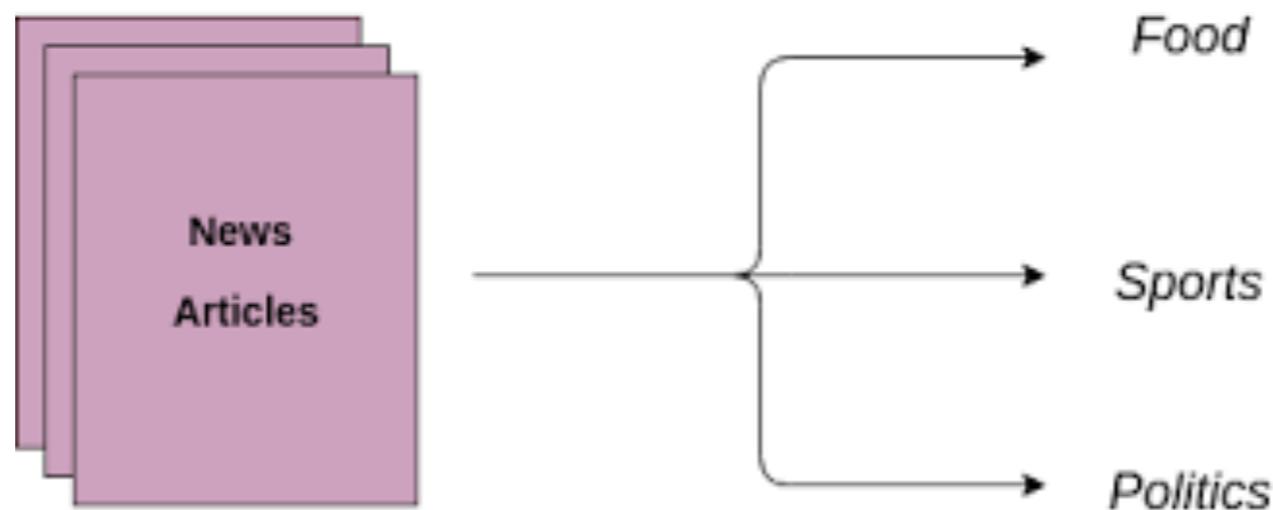
- Dimensionality reduction and feature selection can decrease variance by simplifying models.
- Similarly, a larger training set tends to decrease variance.
- Adding features (predictors) tends to decrease bias, at the expense of introducing additional variance.
- linear and Generalized linear models can be regularized to decrease their variance at the cost of increasing their bias

# Supervised Learning : Classification

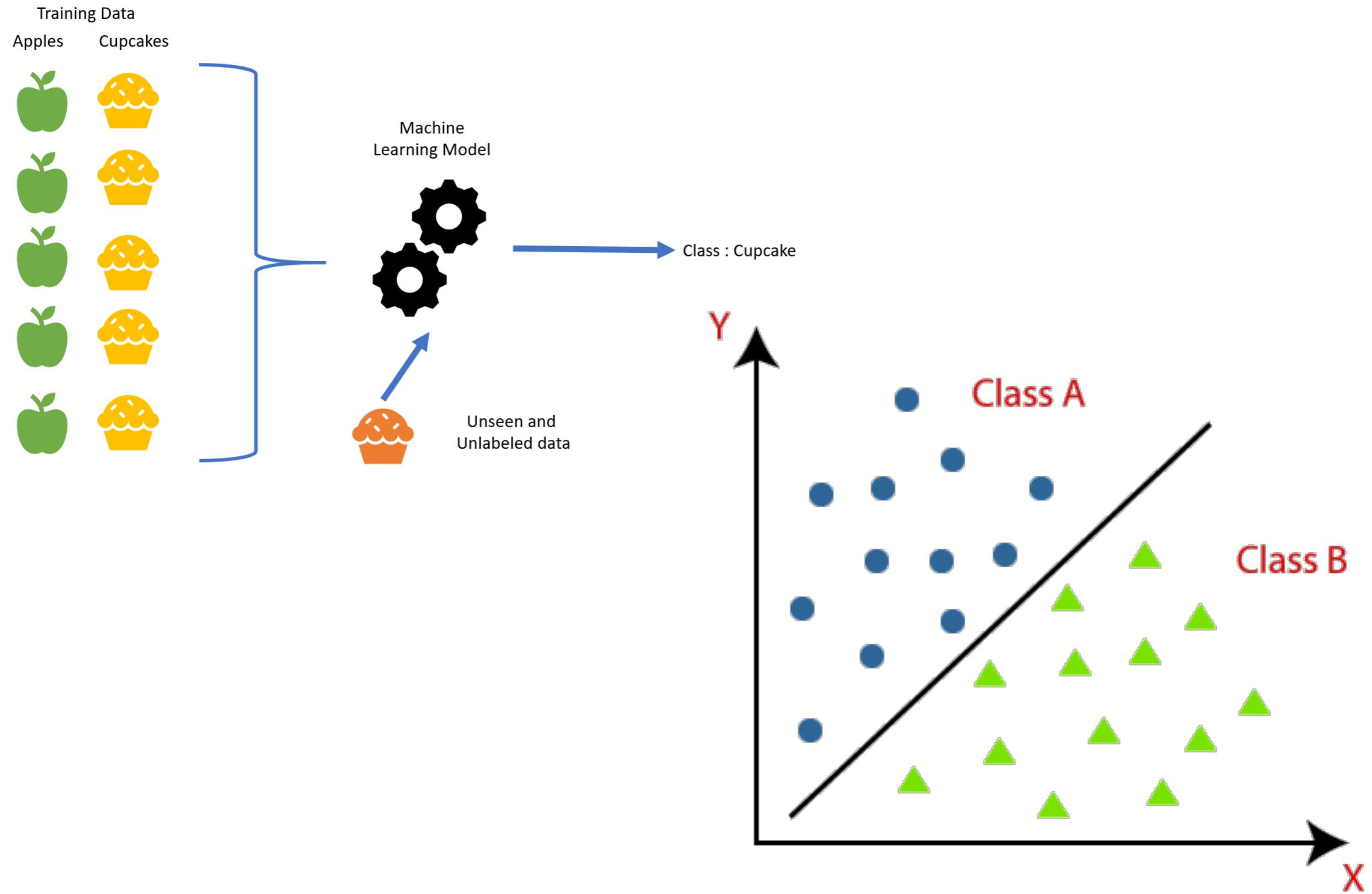
- Binary classification :  $y = \{0,1\}$
- Multiclass classification :  $y = \{1,2,\dots,K\}$



$p(y|x)?$

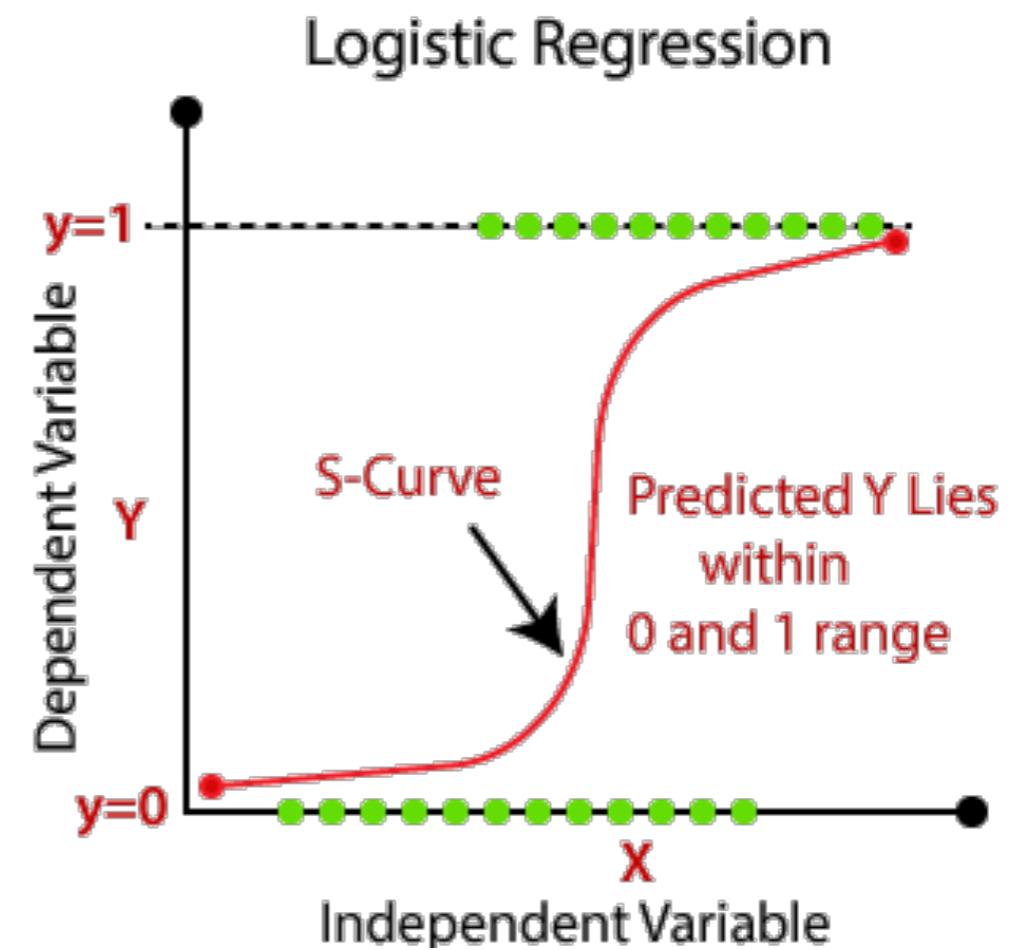
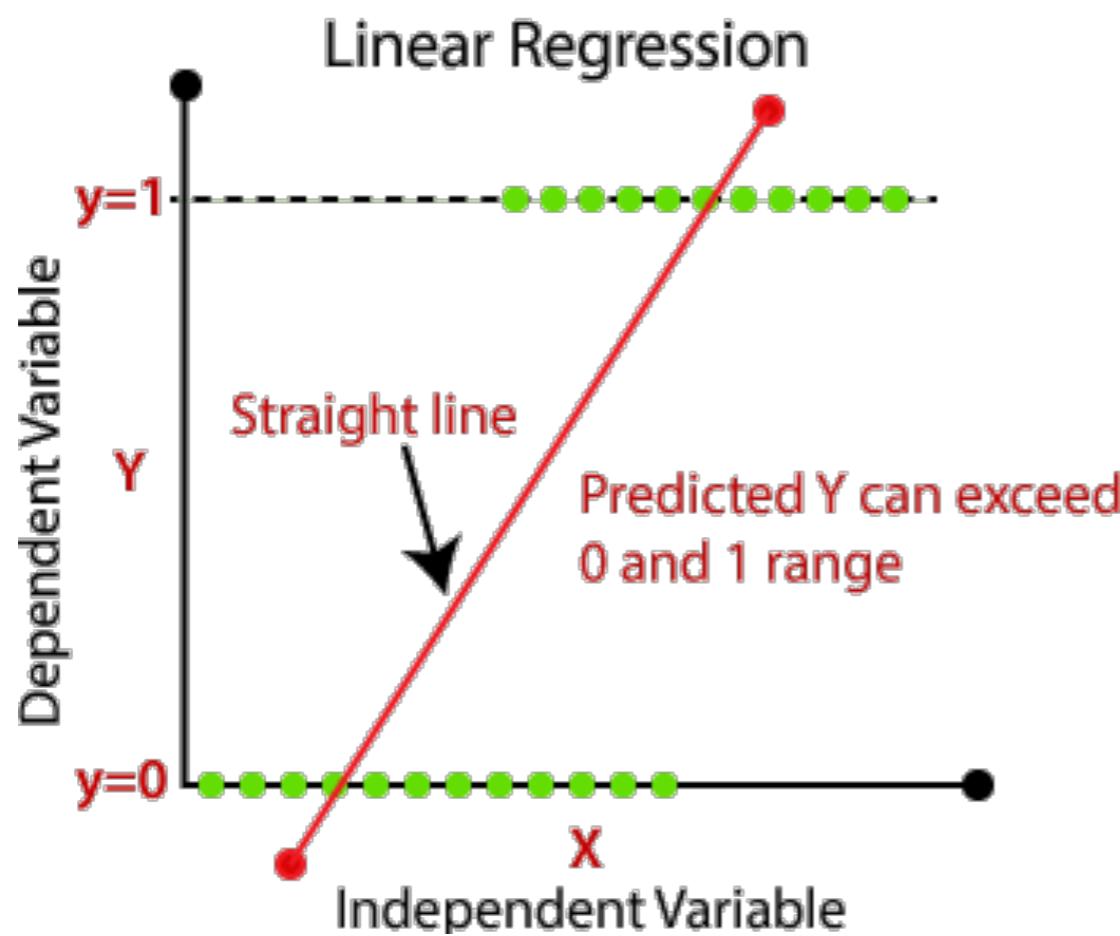


# Classification : Linear Models



# Linear regression to Logistic regression

- Y takes value 0 or 1

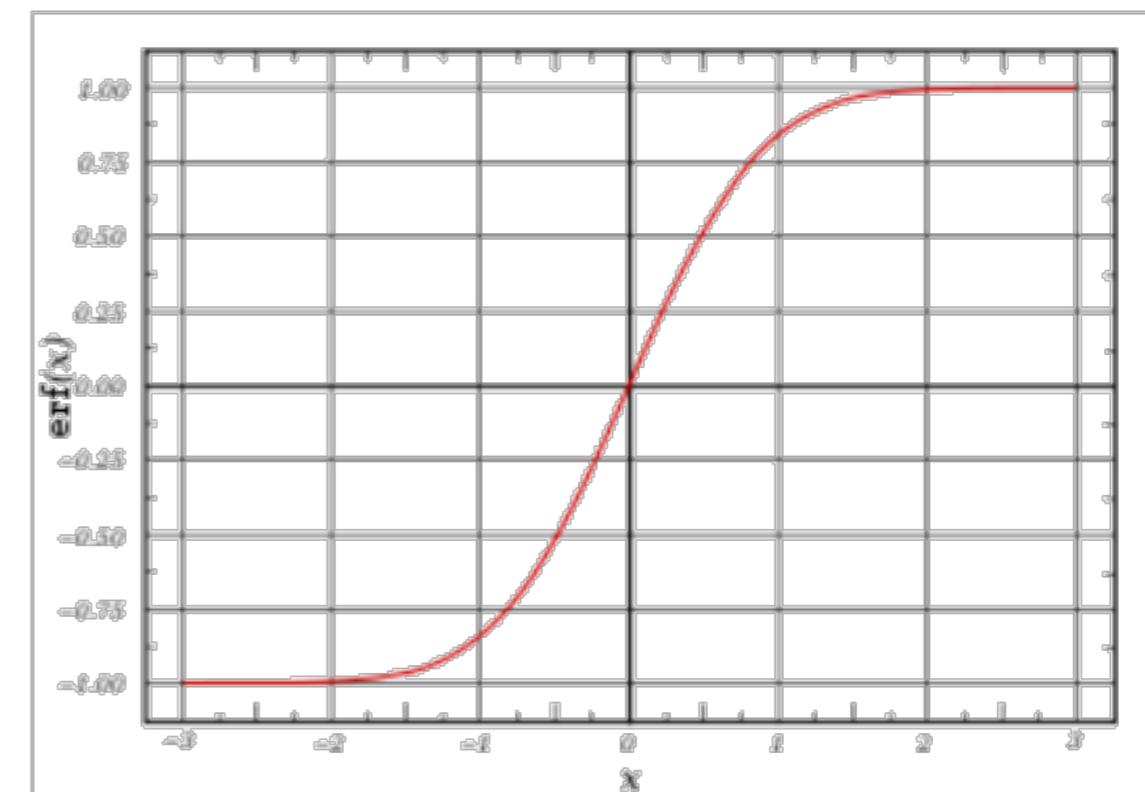
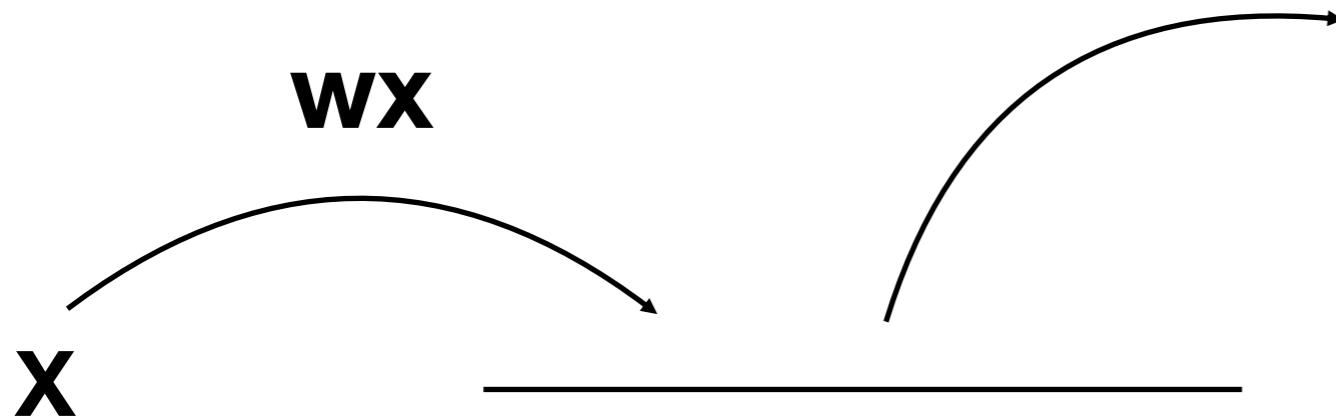


# Logistic Regression

- A discriminative approach which directly models  $p(y|x)$
- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.
- Let  $X$  be the data instance, and  $Y$  be the class label  $\{0, 1\}$  :  
Model  $P(Y|X)$  directly using a **Sigmoid function**:

**Logistic Sigmoid :**  $P(Y = 1 | X) = \frac{1}{1 + e^{-wx}}$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



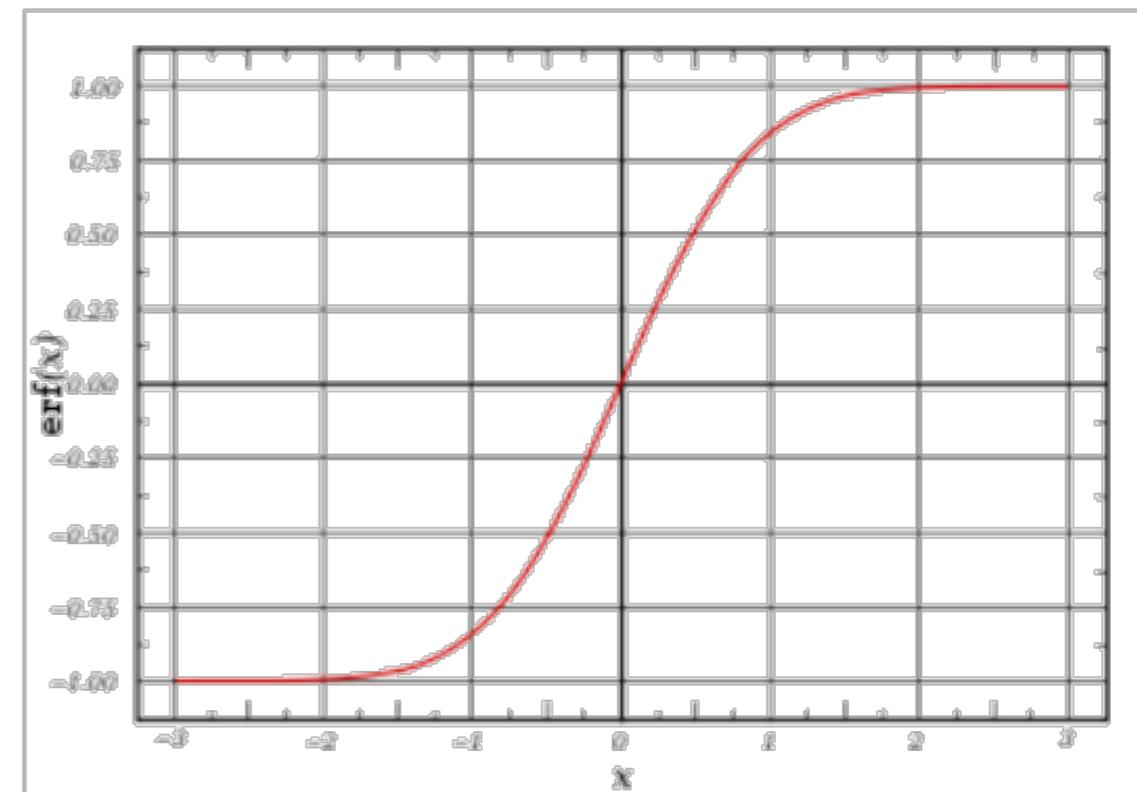
# Logistic Regression

- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.
- Let  $\mathbf{X}$  be the data instance, and  $\mathbf{Y}$  be the class label:  
Model  $P(\mathbf{Y}|\mathbf{X})$  directly using a **Sigmoid function**: Logistic function

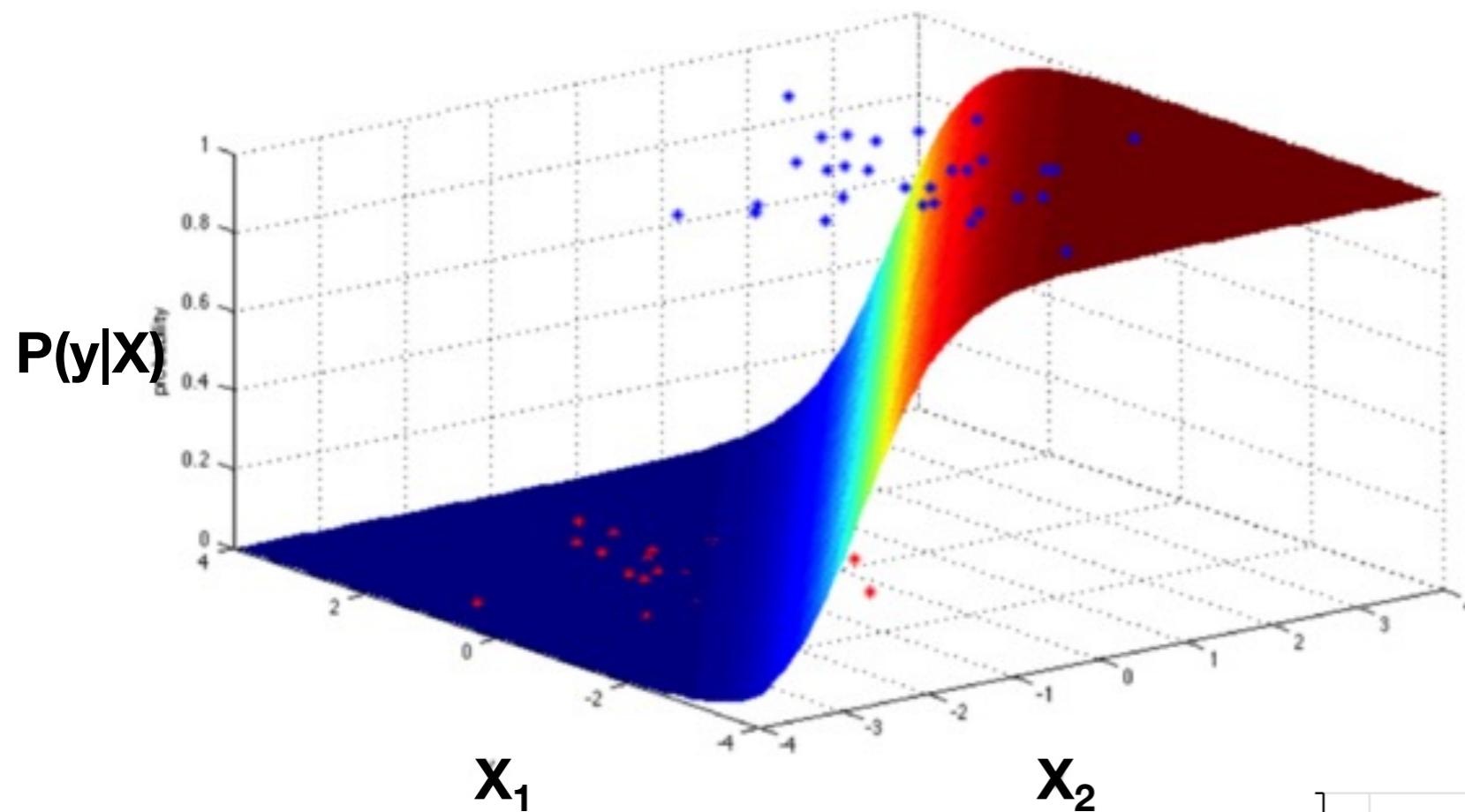
$$P(Y = 1 | \mathbf{X}) = \frac{1}{1 + e^{-\mathbf{w}\mathbf{x}}}$$

[HW] Find derivative of  $s(w) = p(y=1|\mathbf{X})$ !

$$\Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta|0, 1) d\theta$$



# Logistic Regression

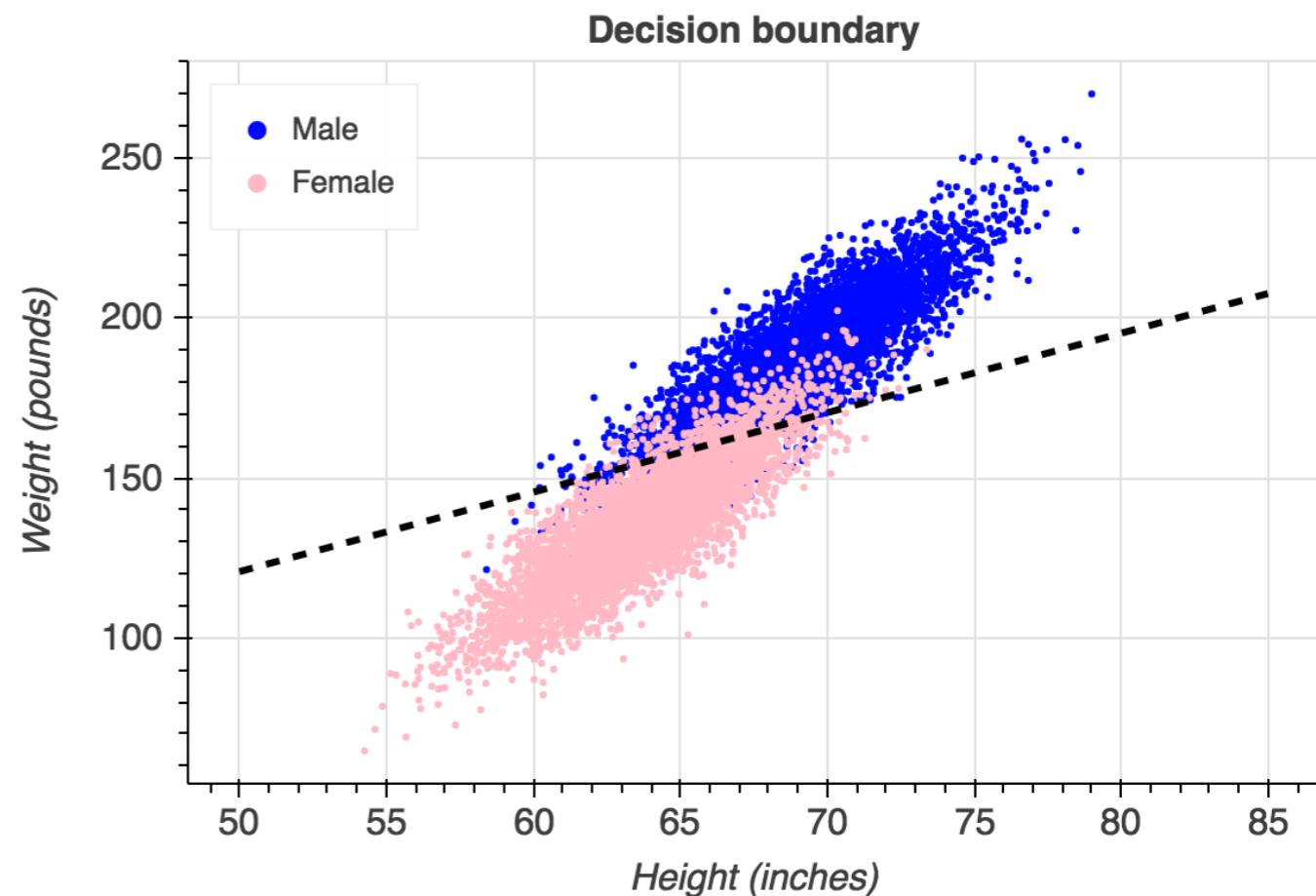


Decision surfaces correspond to  $y(\mathbf{x}) = \text{constant}$ , so that  $\mathbf{w}^T \mathbf{x} + w_0 = 0$

## Discriminant Functions

$$P(Y = 1 | \mathbf{X}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

Decision surfaces are linear functions of  $\mathbf{x}$



# Discriminant function

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

input vector  $\mathbf{x}$  is assigned to class  $C1$  if  
 $y(\mathbf{x}) > 0$  and to class  $C2$  otherwise

decision boundary is therefore  
defined by the relation  $y(\mathbf{x}) = 0$

$\mathbf{w}$  is orthogonal to every vector lying within  
the decision surface, and so  $\mathbf{w}$  determines  
the orientation of the decision surface.

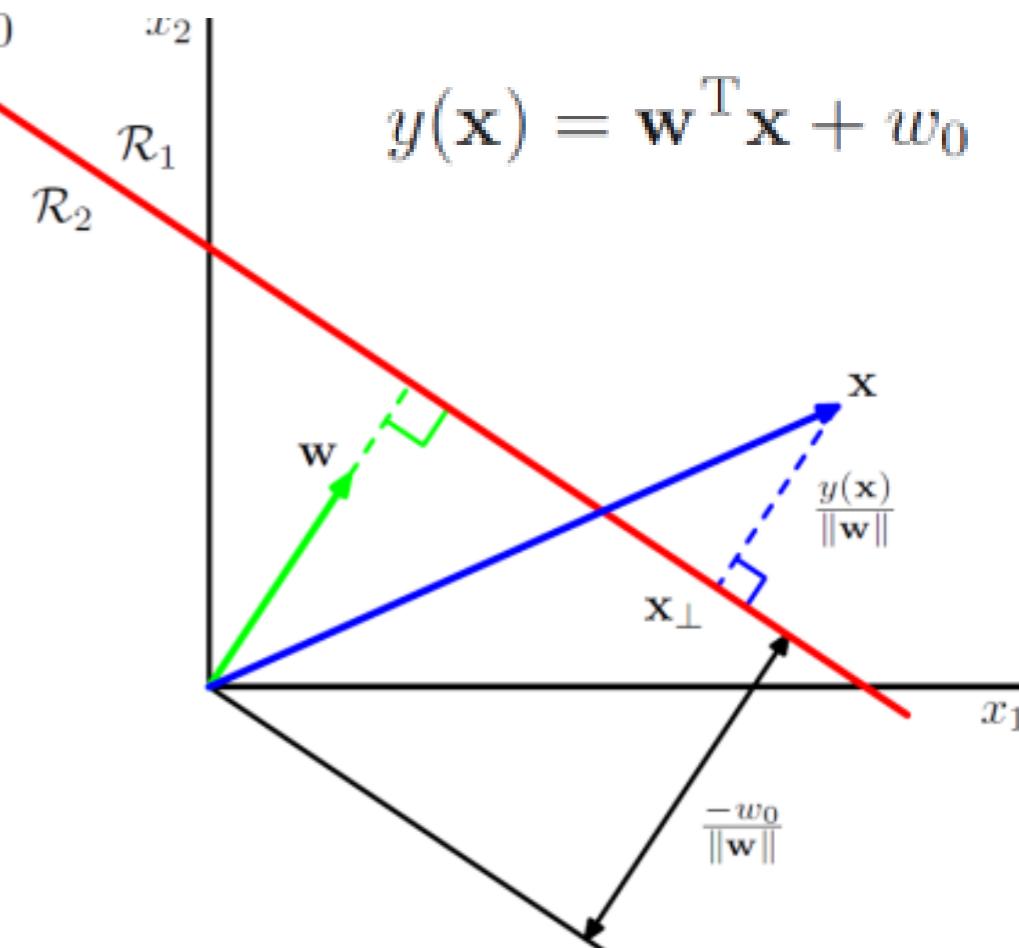
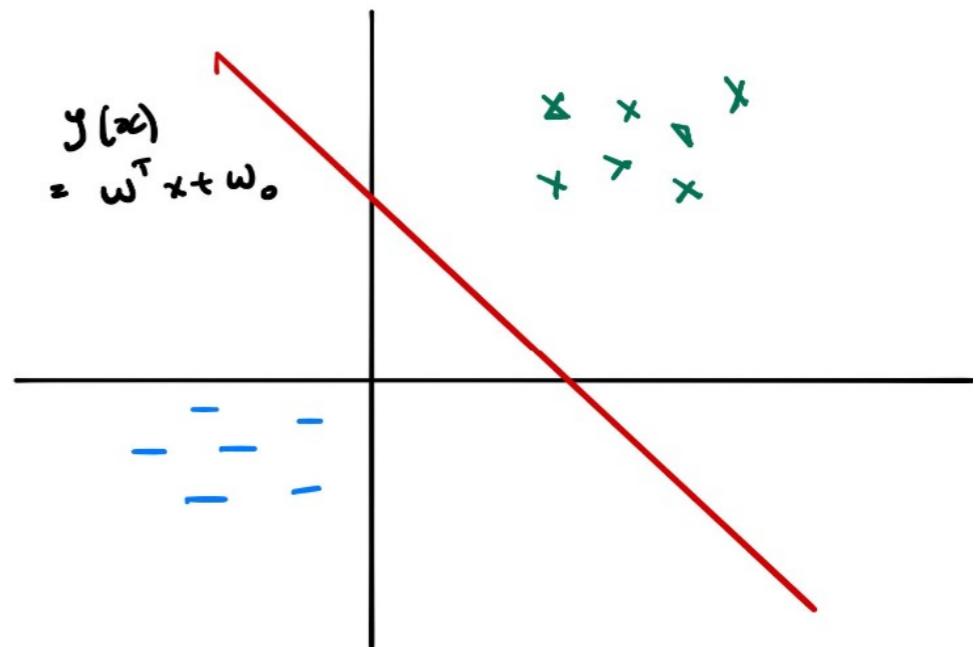
two points  $\mathbf{XA}$  and  $\mathbf{XB}$  both of which lie on  
decision surface.

$y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$ , we have  $\mathbf{w}^T (\mathbf{x}_A - \mathbf{x}_B) = 0$ :

$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

$$y(\mathbf{x}_\perp) = \mathbf{w}^T \mathbf{x}_\perp + w_0 = 0,$$

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}.$$



# Least squares classification

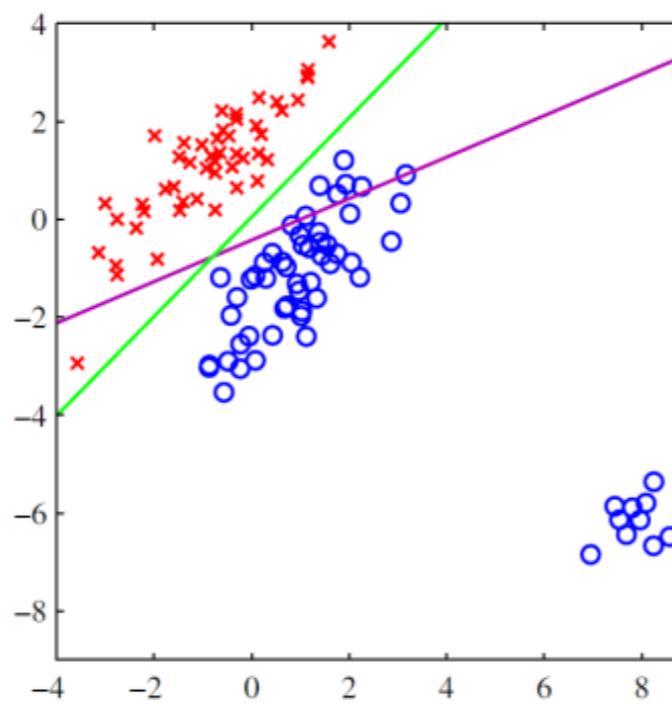
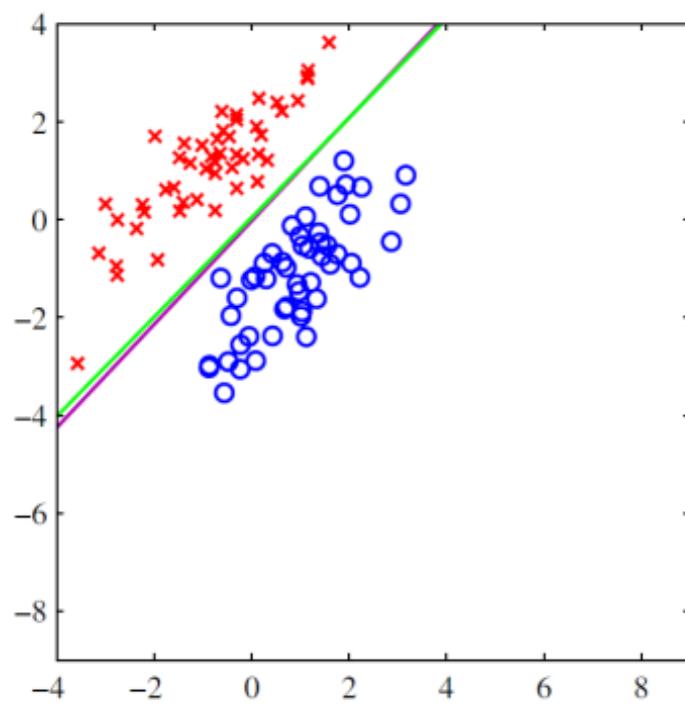
- Loss function

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \right\}.$$

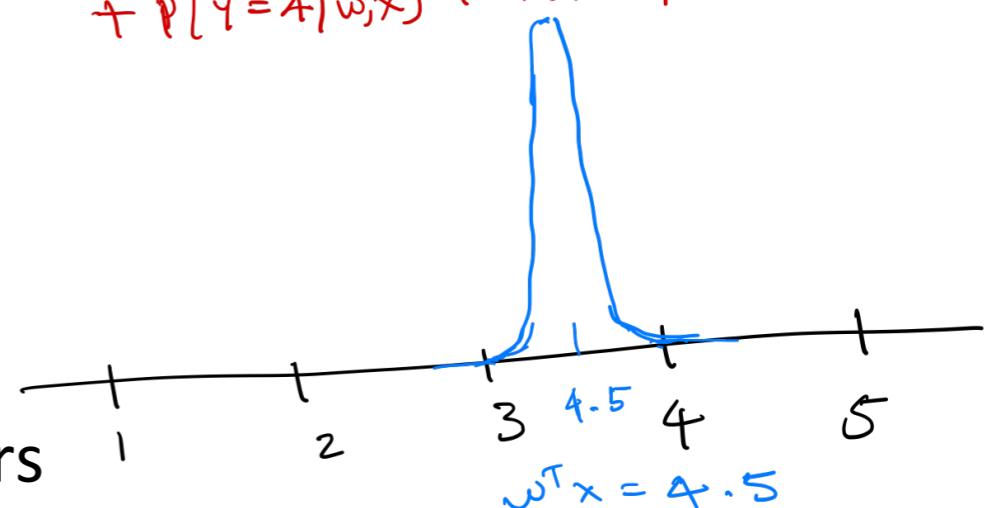
- Solution

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T} = \tilde{\mathbf{X}}^\dagger \mathbf{T}$$

- least-squares solutions lack robustness to outliers



$$\begin{aligned} p(y=1|\omega, x) + p(y=2|\omega, x) + p(y=3|\omega, x) \\ + p(y=4|\omega, x) + p(y=5|\omega, x) \neq 1 \end{aligned}$$



LS :  $p(y|\omega, x) \sim N(y; \omega^T x, \sigma^2)$

**Least squares** corresponds to maximum likelihood under the assumption of a **Gaussian conditional** distribution, whereas binary target vectors clearly have a distribution that is far from Gaussian.

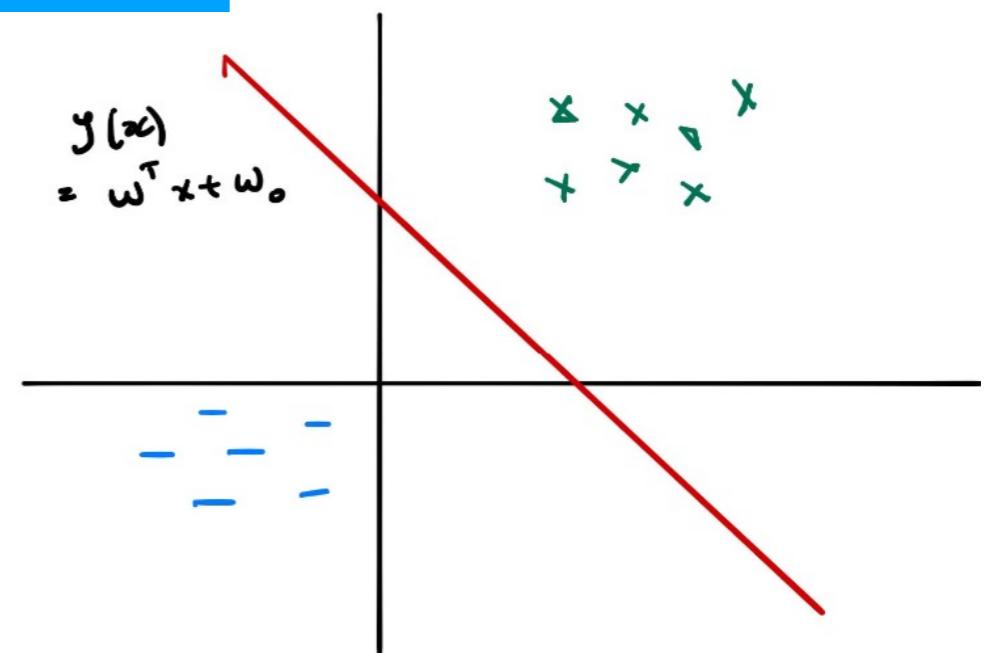
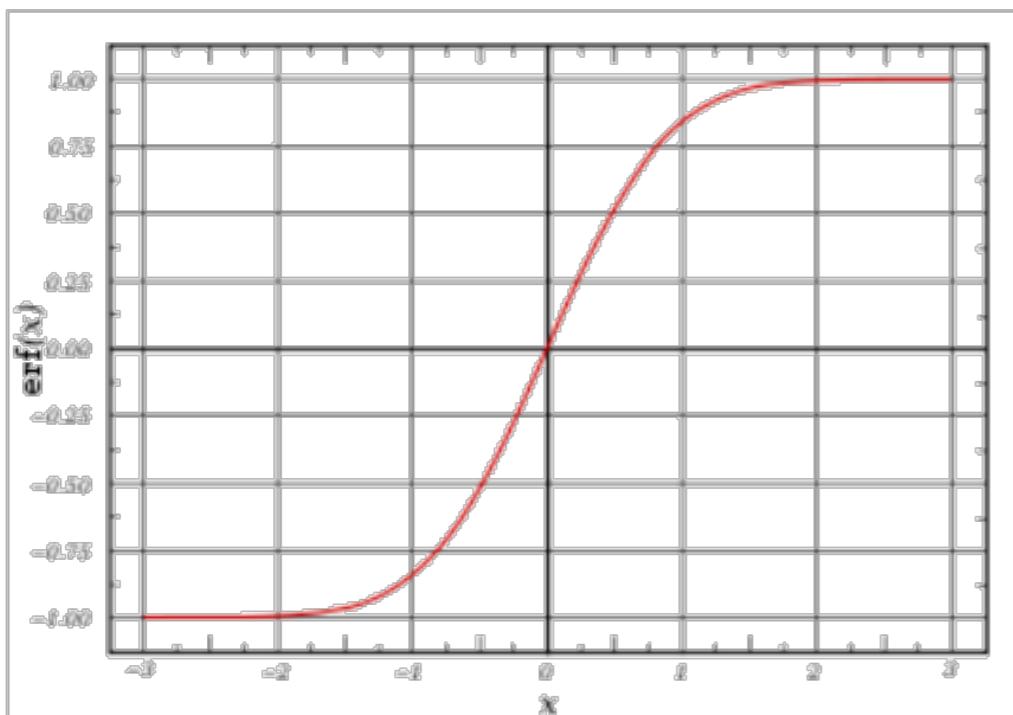
# Classification : Probabilistic vs Geometric views

- Logistic Regression
- Model probability predicting  $y$  given  $x$  and  $w$
- Discriminant Analysis
- Models a discriminant function separating classes

$$P(Y = 1 | \mathbf{X}) = \frac{1}{1 + e^{-\mathbf{w}\mathbf{x}}}$$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

How to learn the parameters  $w$  ?



# Logistic Regression

- In logistic regression, we learn the conditional distribution  $P(y|x)$
- Let  $p_y(x;w)$  be our estimate of  $P(y|x)$ , where  $w$  is a vector of adjustable parameters.
- Assume there are two classes,  $y = 0$  and  $y = 1$  and

$$p_1(x; w) = \frac{1}{1 + e^{-wx}} \quad p_0(x; w) = 1 - \frac{1}{1 + e^{-wx}} = \frac{1}{1 + e^{wx}}$$

- This is equivalent to

$$\log \frac{p_1(x; w)}{p_0(x; w)} = wx$$

- That is, the log odds of class 1 is a linear function of  $x$
- Q: How to find  $W$ ?

- Alternate representation of  $p(y|x)$  :  $p_y(x; w) = \frac{1}{1 + e^{-yw\cdot x}}$ ;  $y = \{-1, 1\}$

# Logistic Regression

- Conditional data likelihood - Probability of observed  $Y$  values in the training data, conditioned on corresponding  $X$  values.
- We choose parameters  $w$  that satisfy

$$\mathbf{w} = \arg \max_{\mathbf{w}} \prod_l P(y^l | \mathbf{x}^l, \mathbf{w})$$

- where
  - $\mathbf{w} = \langle w_0, w_1, \dots, w_n \rangle$  is the vector of parameters to be estimated,
  - $y^l$  denotes the observed value of  $Y$  in the  $l$  th training example, and
  - $\mathbf{x}^l$  denotes the observed value of  $\mathbf{X}$  in the  $l$  th training example

# Logistic Regression

- Equivalently, we can work with log of conditional likelihood:

$$\mathbf{w} = \arg \max_{\mathbf{w}} \sum_l \ln P(y^l | \mathbf{x}^l, \mathbf{w})$$

- Conditional data log likelihood,  $l(\mathbf{W})$ , can be written as

$$l(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 | \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 | \mathbf{x}^l, \mathbf{w})$$

- Note here that  $Y$  can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given  $y^l$

# Logistic Regression

- We need to estimate:

$$\mathbf{w} = \arg \max_{\mathbf{w}} \sum_l \ln P(y^l | \mathbf{x}^l, \mathbf{w})$$

$$l(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 | \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 | \mathbf{x}^l, \mathbf{w})$$

- Equivalently, we can minimize negative log likelihood using gradient descent technique
- No closed-form solution though. Iterative method required.
- [HW] Find the derivative of  $l(\mathbf{w})$  !

# Logistic Regression

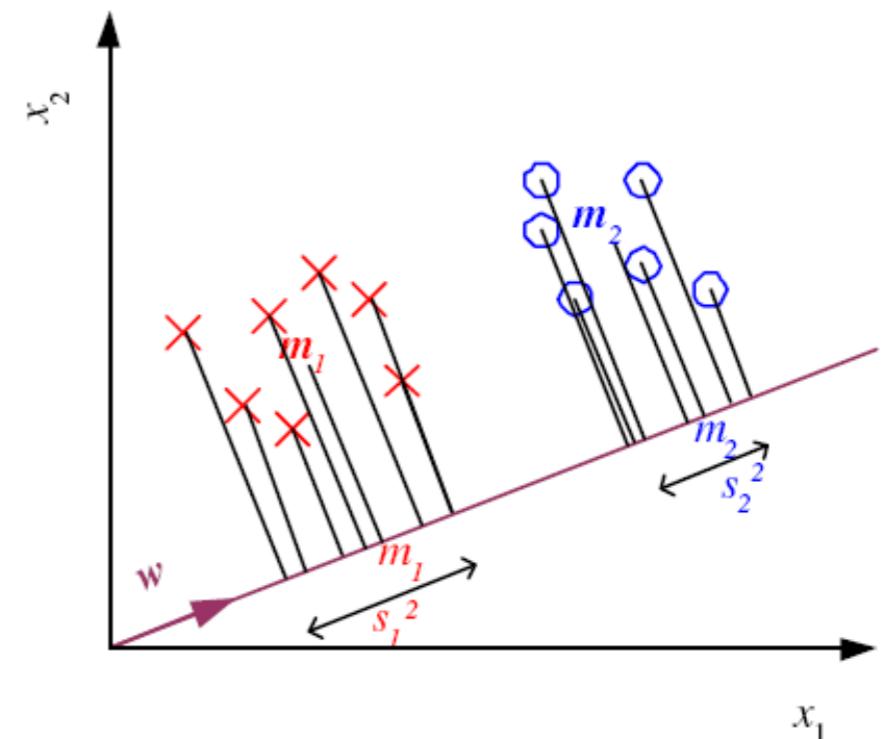
- Overfitting can arise especially when data has very high dimensions and is sparse.
- One approach -> modified “penalized log likelihood function,” which penalizes large values of  $\mathbf{w}$ , as before.

$$\mathbf{w} = \arg \max_{\mathbf{w}} \sum_l \ln P(y^l | \mathbf{x}^l, \mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- [HW] Find the Derivative !

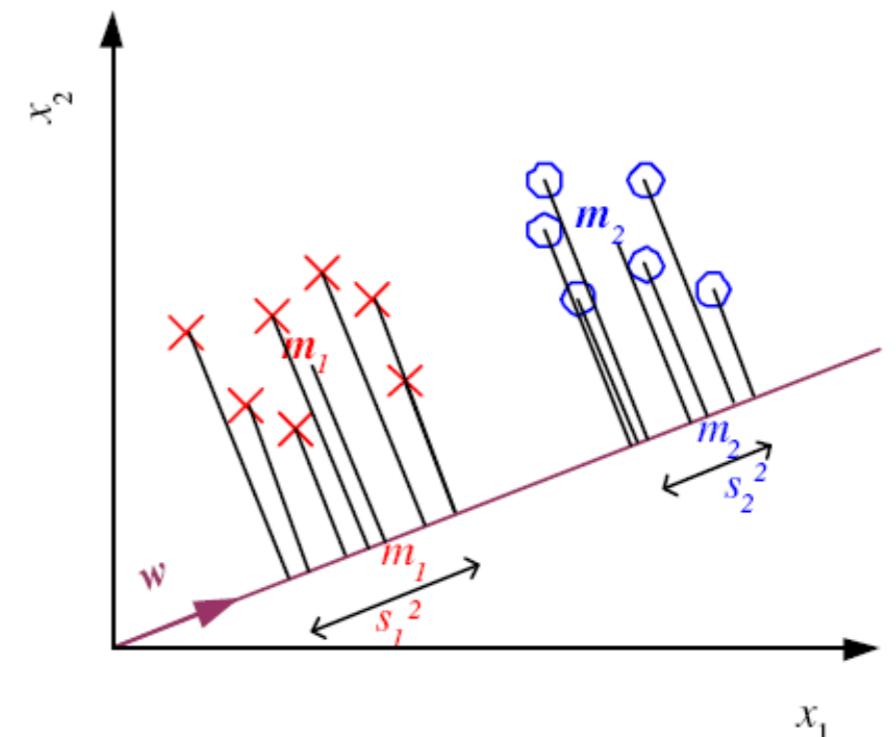
# Discriminant Analysis

- Supervised linear discriminant analysis (LDA)
- $w$  as the direction to project  $x$
- Find  $w$  such that when  $x$  is projected, classes are well-separated.



# Linear Discriminant Analysis

- Supervised linear discriminant analysis (LDA)
- $w$  as the direction to project  $x$
- Find  $w$  such that when  $x$  is projected, classes are well-separated.
- Simplest measure of the separation of the classes, when projected onto  $w$ , is the separation of the projected class means.



$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n. \text{ Maximize } \mathbf{m}_2 - \mathbf{m}_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$

# Linear Discriminant Analysis

$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$

## Pseudocode

This expression can be made arbitrarily large by increasing the magnitude of  $\mathbf{w}$

- ① Constrain  $\mathbf{w}$  to unit-length,  $\sum_i w_i^2 = 1$
- ② Use Lagrange multipliers for the constrained maximisation
- ③ Find the solution,  $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$  ( $\star^1$ )

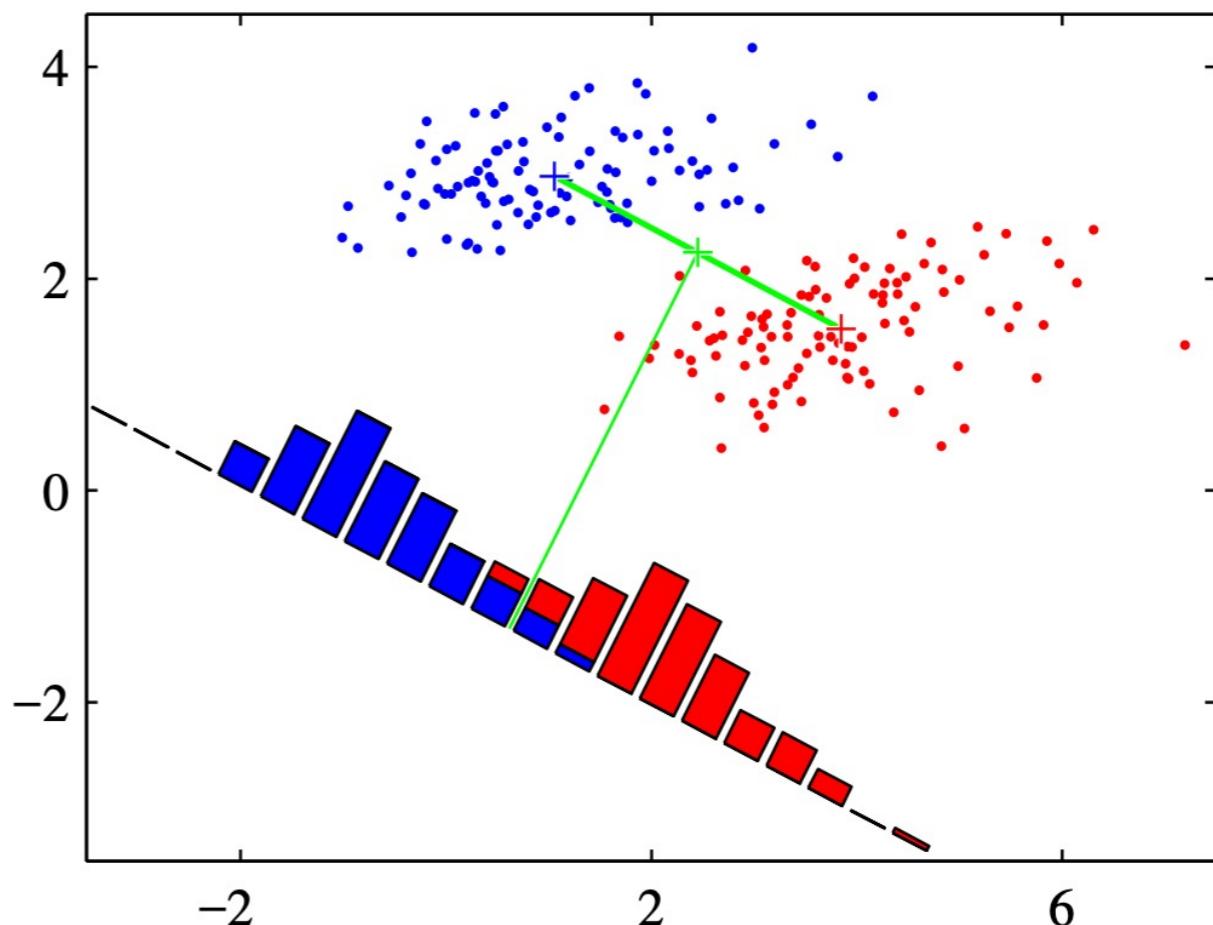
The optimal projection is along the line joining the original class means

$$\begin{aligned} {}^1L &= \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1) + \lambda(\mathbf{w}^T \mathbf{w} - 1), \text{ then } \nabla L = \mathbf{m}_2 - \mathbf{m}_1 + 2\lambda\mathbf{w} = 0 \text{ to get} \\ \mathbf{w} &= 1/(2\lambda)(\mathbf{m}_2 - \mathbf{m}_1) \end{aligned}$$

# Linear Discriminant Analysis

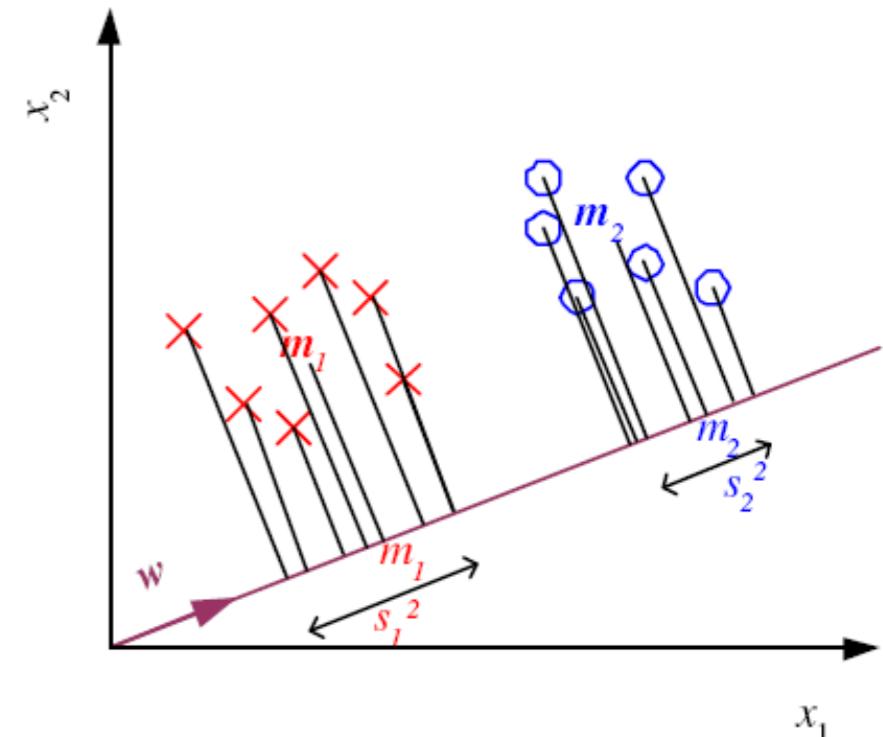
Projection onto the line joining the class means

- Good separation in the original  $2D$  space
- Considerable class overlap in the projection  $1D$  space



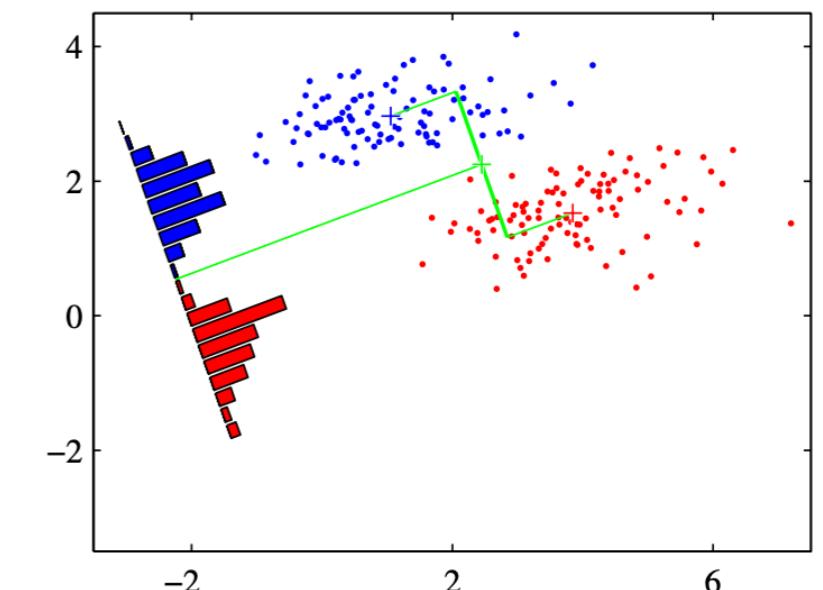
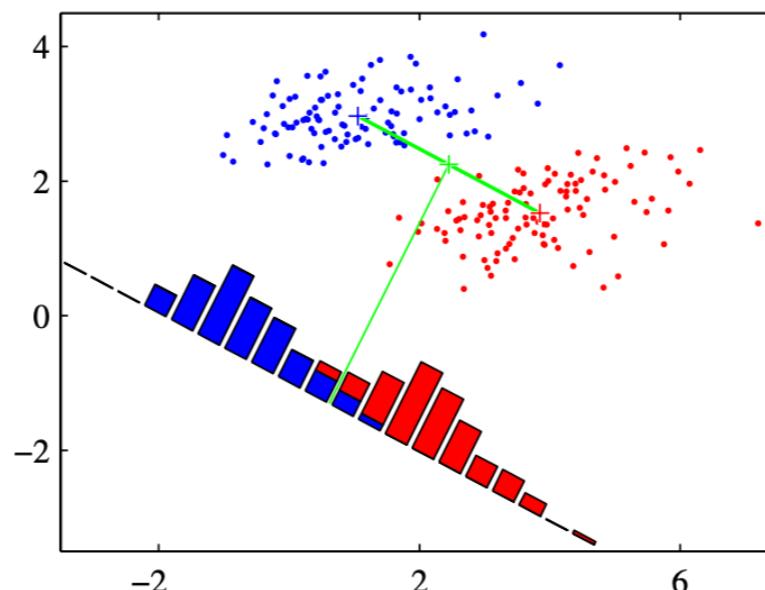
# Fishers Linear Discriminant

- Idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while giving a small variance within each class, thereby minimizing the class overlap.
- Find  $\mathbf{w}$  that maximizes



$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} \quad s_1^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t$$



# Fishers Linear Discriminant

- Between-class scatter:

$$\begin{aligned}(m_1 - m_2)^2 &= (\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2)^2 \\&= \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w} \\&= \mathbf{w}^T \mathbf{S}_B \mathbf{w} \text{ where } \mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\end{aligned}$$

- Within-class scatter:

$$\begin{aligned}s_1^2 &= \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t \\&= \sum_t \mathbf{w}^T (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T \mathbf{w} r^t = \mathbf{w}^T \mathbf{S}_1 \mathbf{w}\end{aligned}$$

$$\text{where } \mathbf{S}_1 = \sum_t (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T r^t$$

$$s_1^2 + s_2^2 = \mathbf{w}^T \mathbf{S}_W \mathbf{w} \text{ where } \mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

# Fisher's Linear Discriminant

- Find  $\mathbf{w}$  that max

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{\left| \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w} \right|}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- FLD soln:

$$\mathbf{w} = c \cdot \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

# Fisher's Linear Discriminant

- Find  $\mathbf{w}$  that max

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{|\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w}|}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- FLD soln:  $\mathbf{w} = c \cdot \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$

After differentiating with respect to  $\mathbf{w}^3$ , we get that  $J(\mathbf{w})$  is maximised

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$$

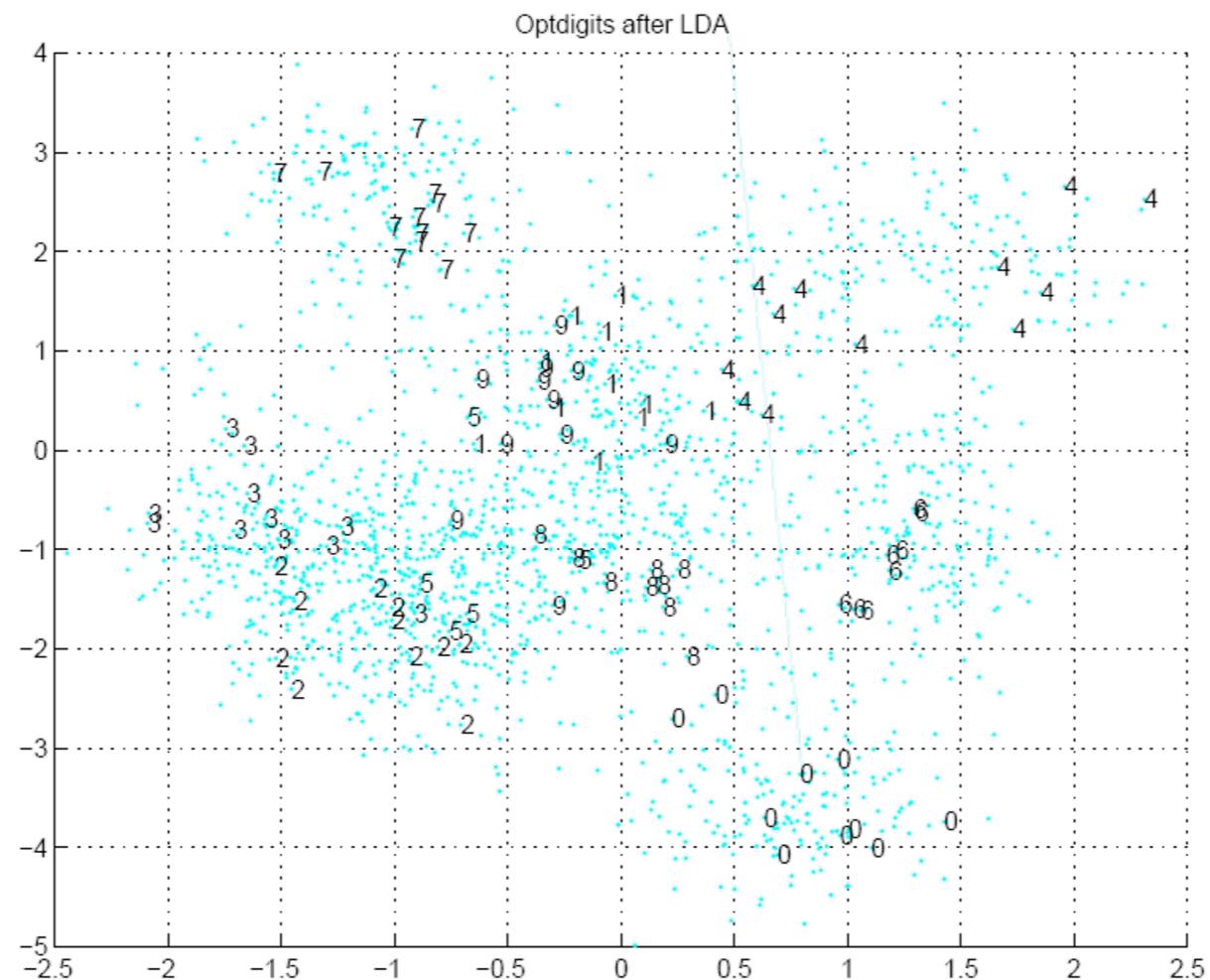
Multiplying both sides of  $(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$  by  $\mathbf{S}_W^{-1}$

$$\mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

$$\frac{\partial}{\partial \mathbf{w}} \left( \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \right) = \frac{2}{(\mathbf{w}^T \mathbf{S}_W \mathbf{w})^2} [(\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w} - (\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w}]$$

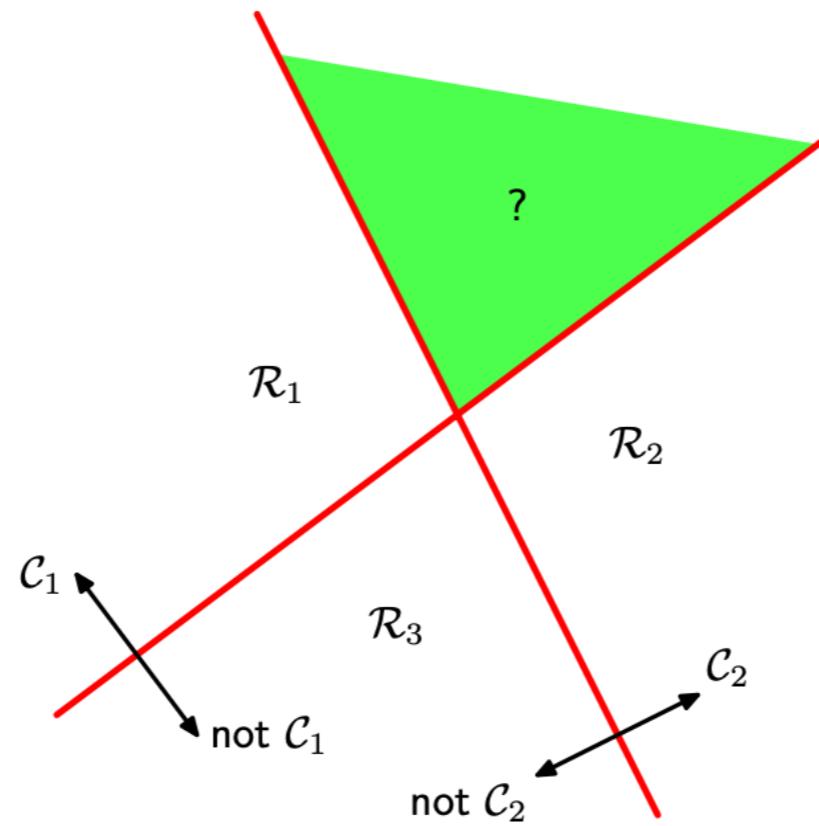
$(\mathbf{m}_1 - \mathbf{m}_2) \mathbf{w}$  is a scalar.

# FLD: Illustration

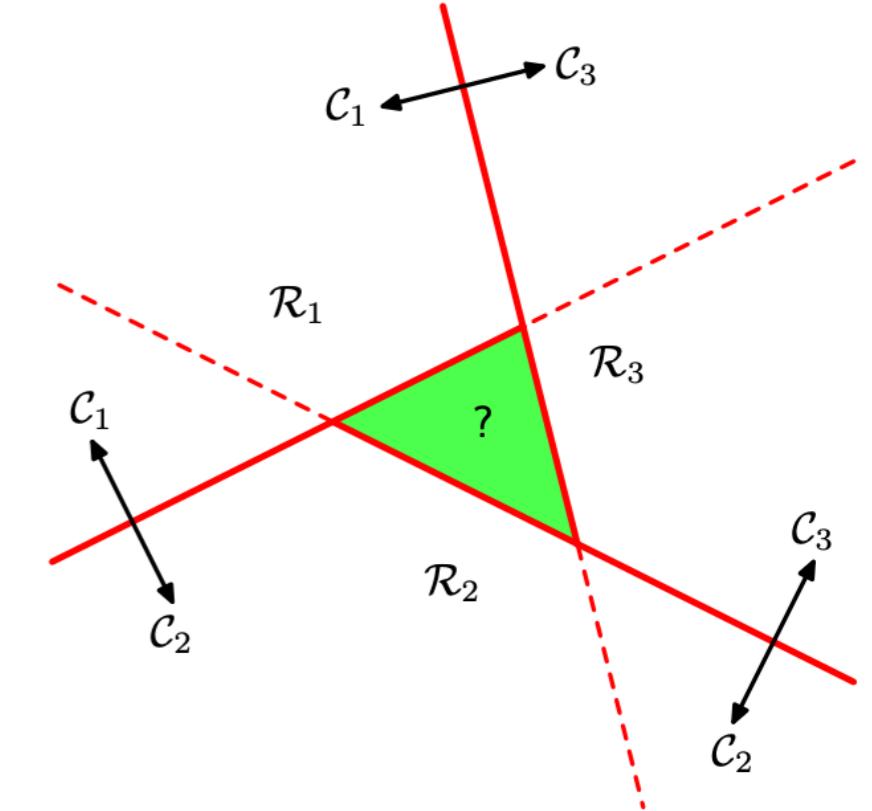


# Multi Class Classification

- One vs Rest



- One vs One



- Discriminant function

$\mathbf{x}$  to class  $\mathcal{C}_k$  if  $y_k(\mathbf{x}) > y_j(\mathbf{x})$  for all  $j \neq k$ .

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

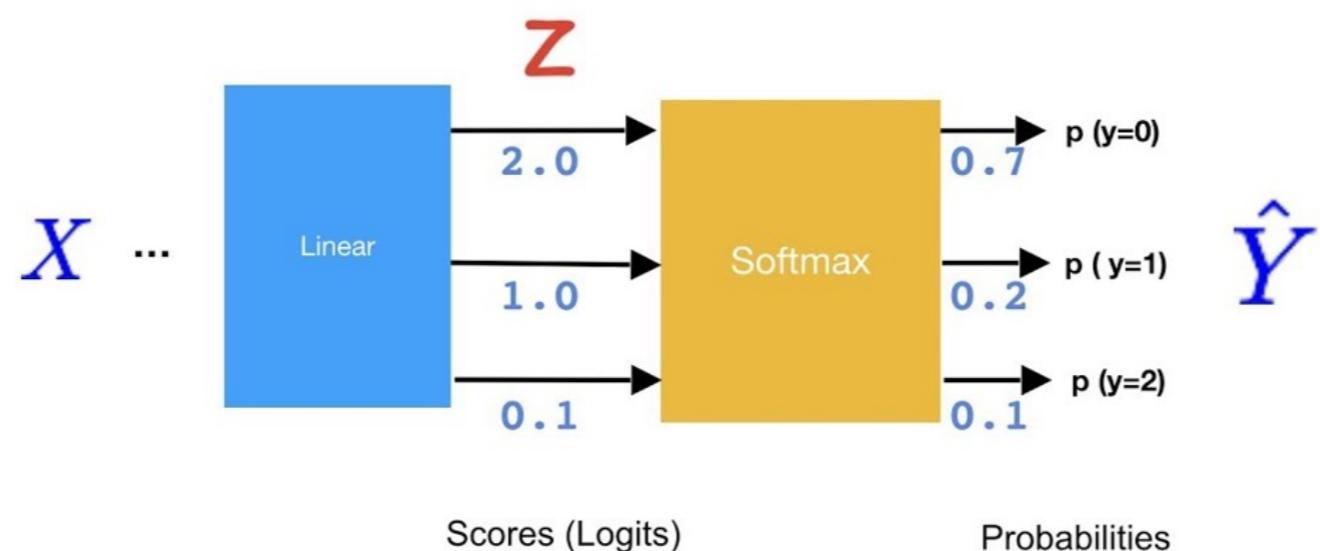
# Multi class classification

- LR: Functional form of  $P(Y|X)$ , no assumption on  $P(X|Y)$
- LR is a linear classifier
- LR optimized by conditional likelihood
- Extending logistic regression to multiple classes
  - Use softmax for each class  $k$ !

$$p(y = k|x) = \frac{\exp(w_k^\top x)}{\sum_{i=1}^K \exp(w_i^\top x)}$$

Meet Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K.$$



# **Thank you !**

## **Reference**

- [1] Christopher Bishop, Pattern Recognition and Machine Learning**
- [2] Kevin Murphy, Machine Learning : A Probabilistic Perspective**