13.08.2024

Statistical Methods in AI (CS7.403)

Lecture-5: k-Nearest Neighbours, Linear Regression

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

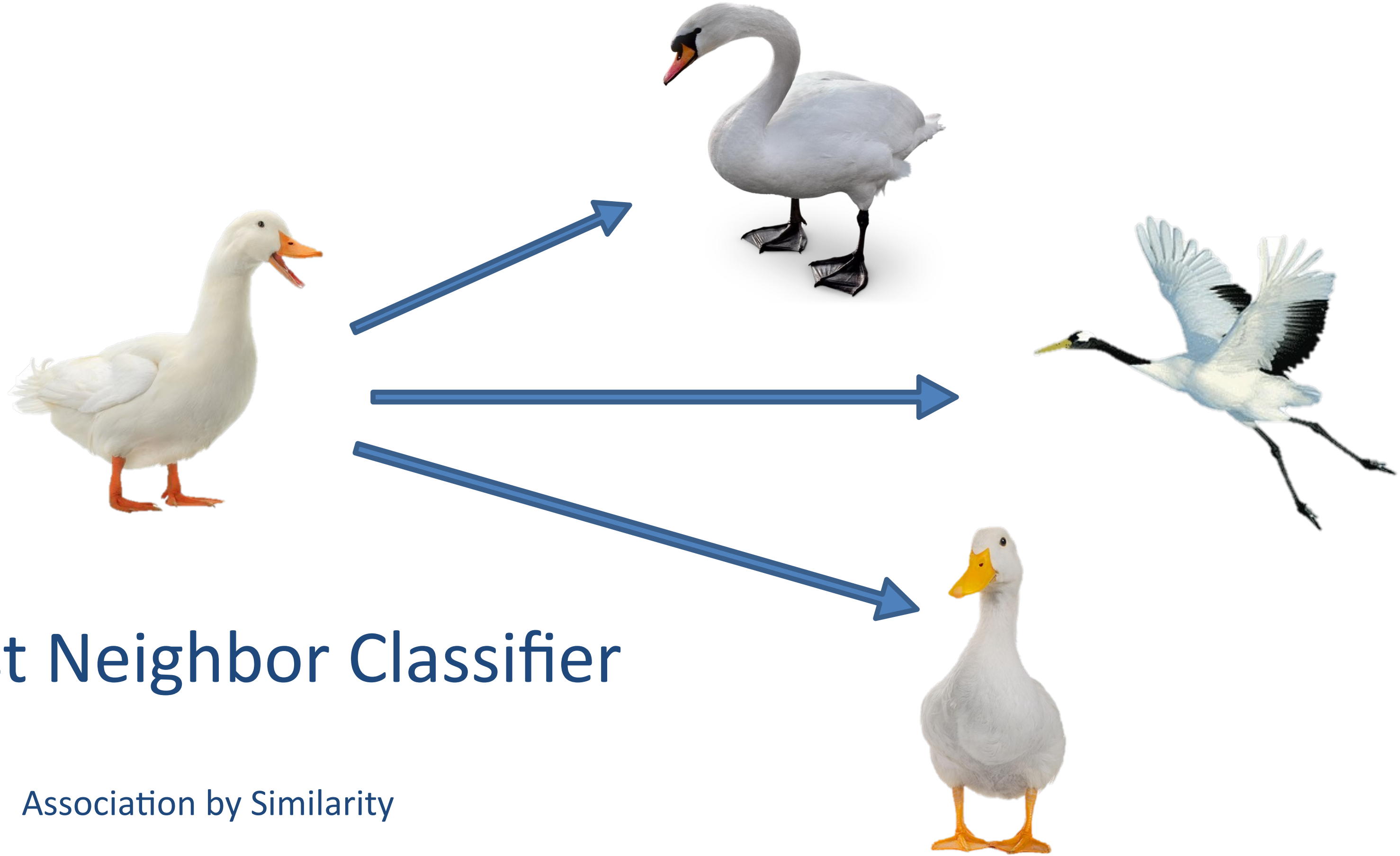
@vikataravi

Center for Visual Information Technology (CVIT)

IIIT Hyderabad

- F1-macro (HM) v/s F1-macro (scikit-learn)

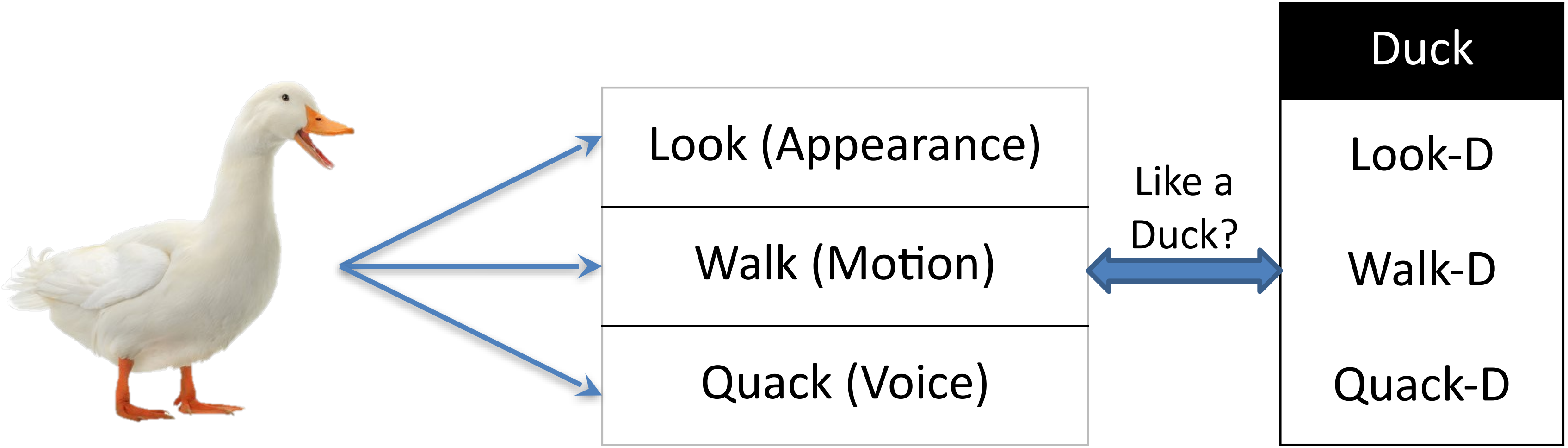
Nearest Neighbor Classifier



Association by Similarity

How do we compare?

If it looks like a duck, walks like a duck and quacks like a duck,



- Find distance to feature vectors of known classes

Nearest Neighbor Classifier

- Assign label of that sample which is nearest to the test sample

X_{test} :

[30.9, 15.1, 1.32]

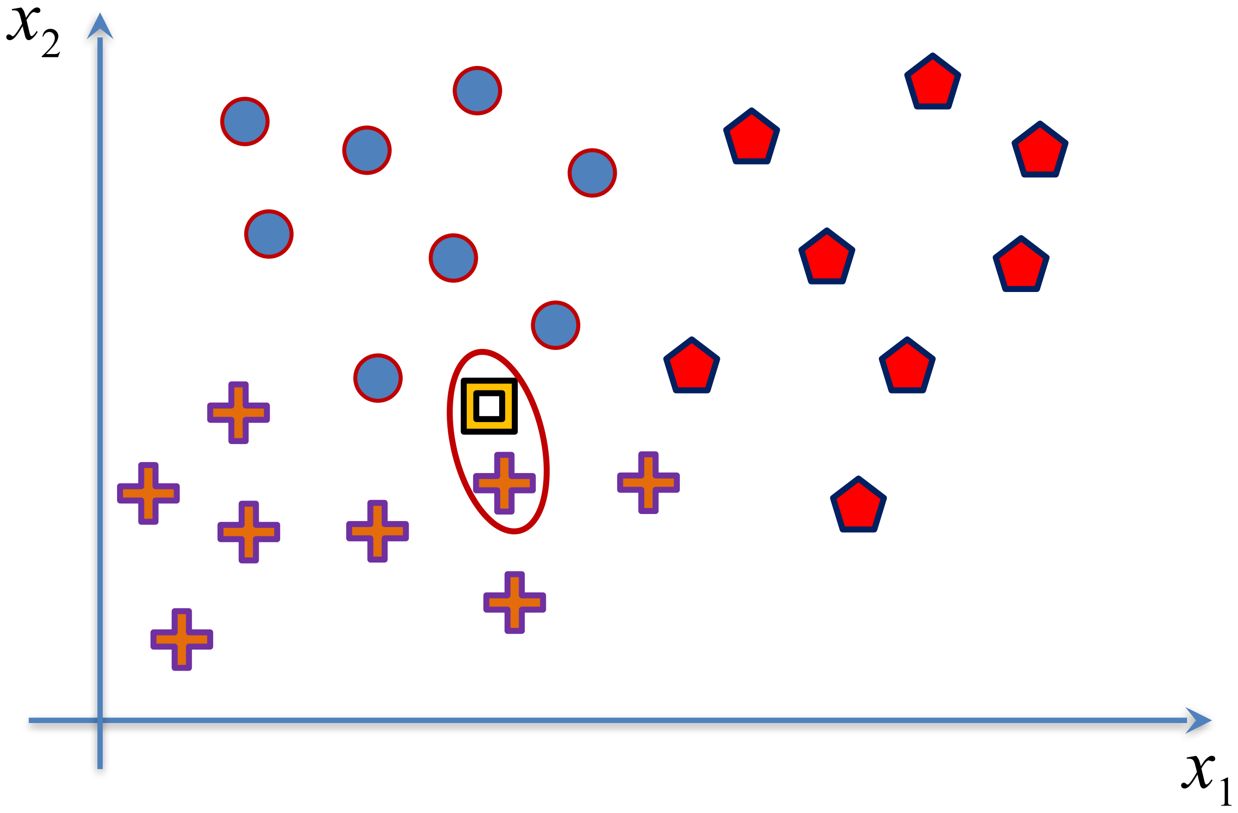
The test sample is a Duck

Training Samples

Distance	Feature Vector	Label
$\boxed{1.30}$	X_1 [32.1, 14.6, 1.42]	$\boxed{\text{Duck}}$
5.78	X_2 [25.3, 16.3, 2.11]	Swan
13.54	X_3 [42.2, 7.7, 0.38]	Crane
4.71	X_4 [26.7, 17.1, 2.04]	Swan
15.21	X_5 [44.1, 7.6, 0.32]	Crane
3.04	X_6 [31.4, 12.1, 1.29]	Duck
13.58	X_7 [41.9, 7.2, 0.35]	Crane

Visualization in Feature Space

- The nearest train sample is a +.
- We assign the test sample to the + class.



The 1-NN Classification Algorithm

Problem

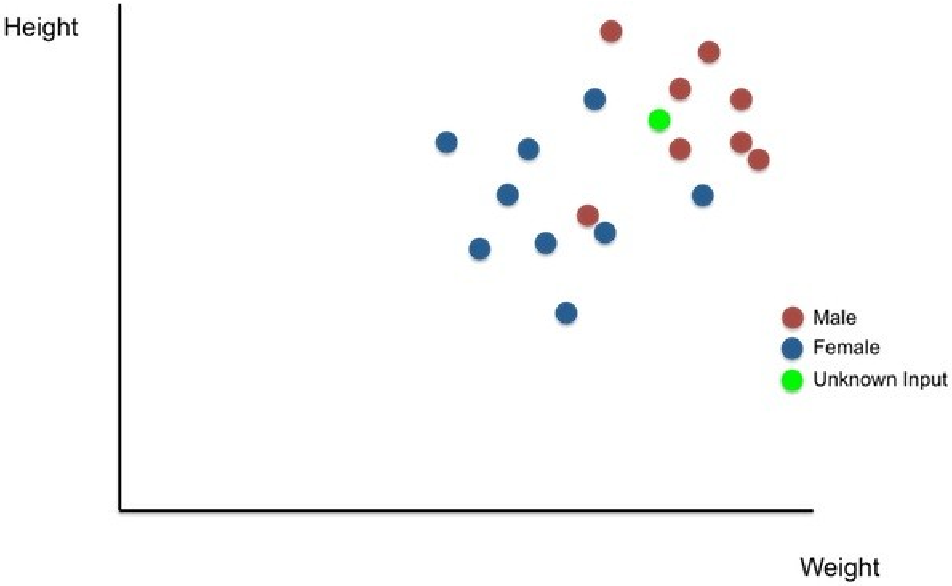
- Given:
  - A set of training ***n*** training samples: $(\mathbf{x}_i, y_i)$
  - A test sample: $\mathbf{x}_t$
- Find:
  - label($\mathbf{x}_t$)

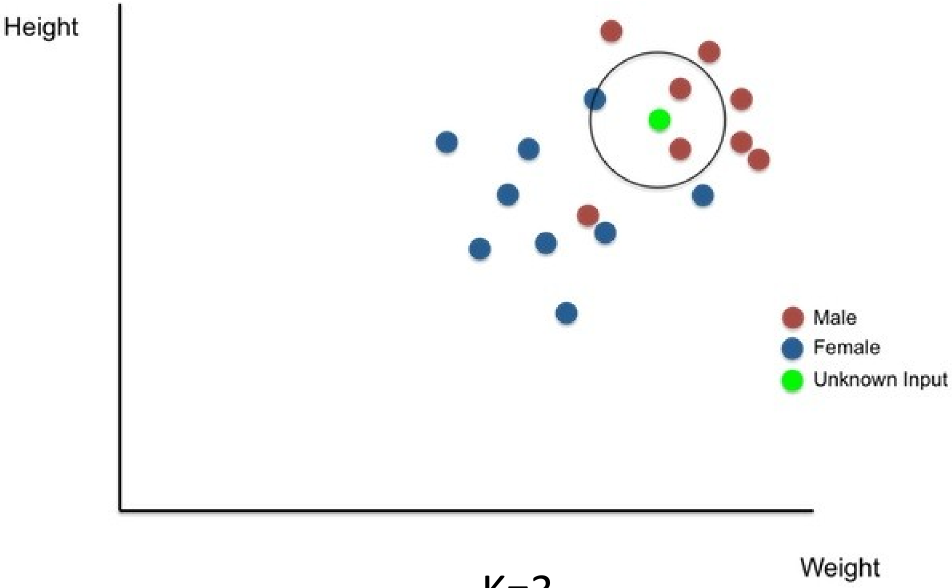
Algorithm

```
def 1NN_Classifier(X_train,x_t,dist_metric):

 minDist = sys.float_info.max
 for (x_i,y_i) in {X_train}:
    dist = Dist(x_i, x_t,dist_metric)
    if (dist < minDist):
            minDist = dist
            nearest = y_i
 return nearest
```

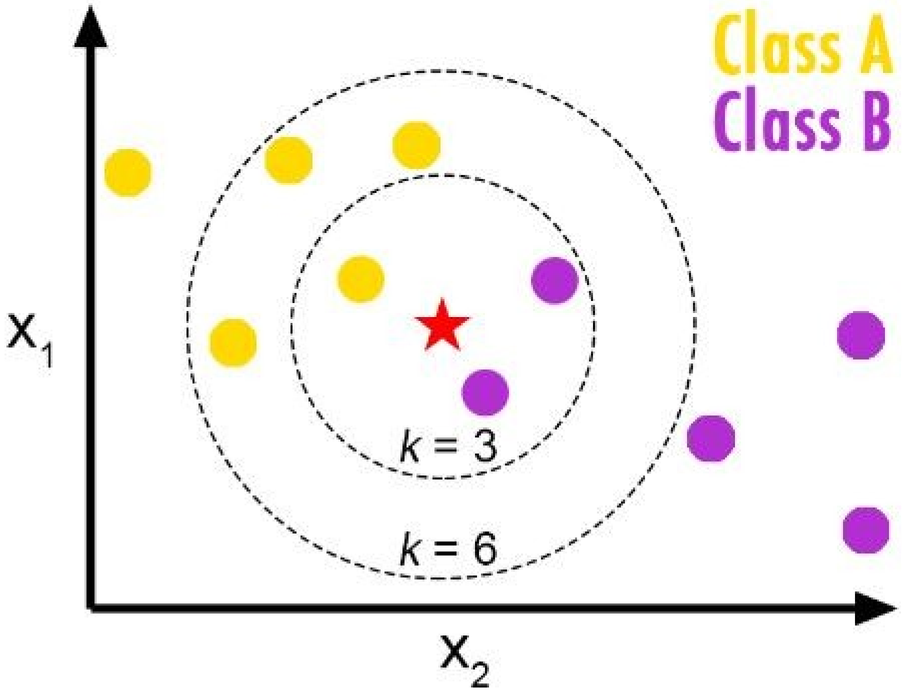
k-nearest neighbor classifier





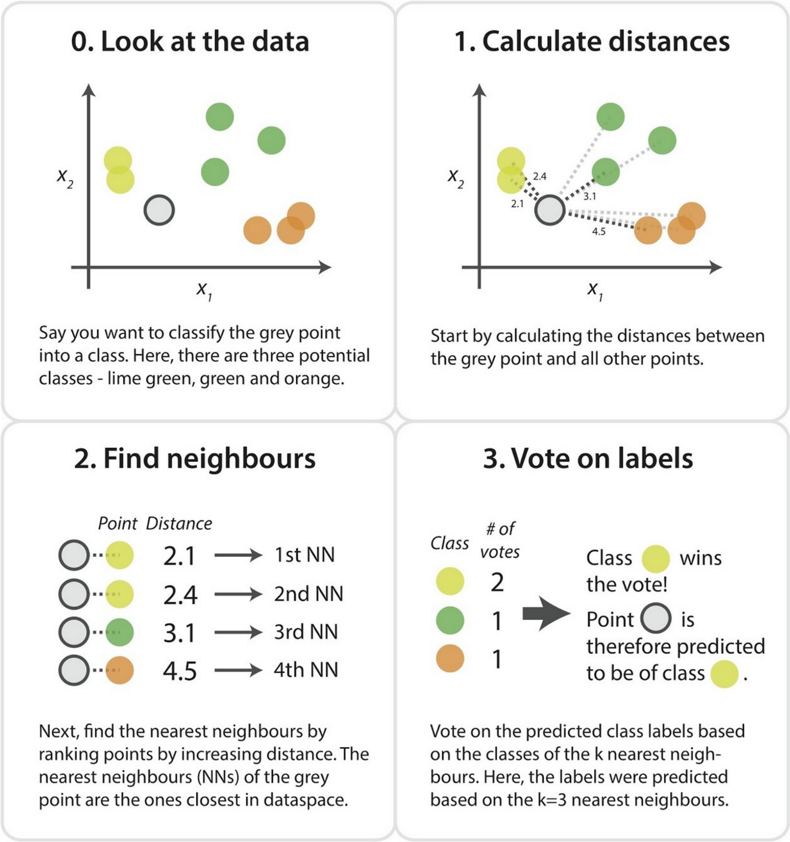
K=3

k-nearest neighbor classifier



K is usually an odd number

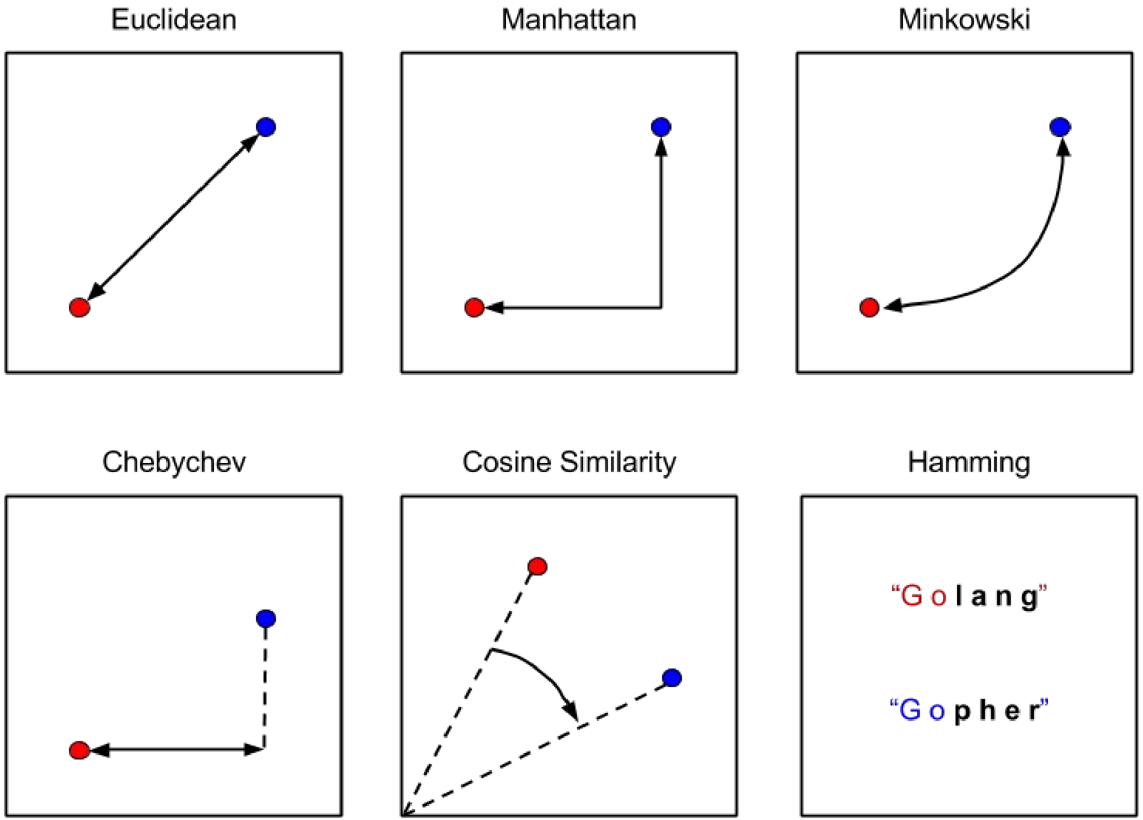
k-NN algorithm in pictures



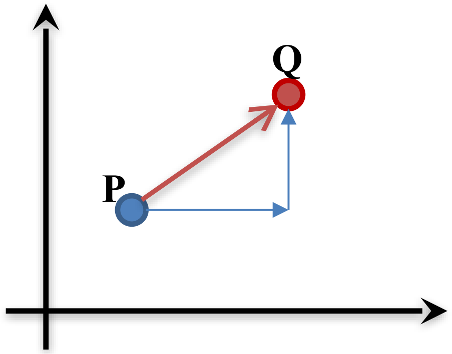
Complexity of k-NN

- Training
  - Time:
  - Space:
- Testing
  - Time:
  - Space:

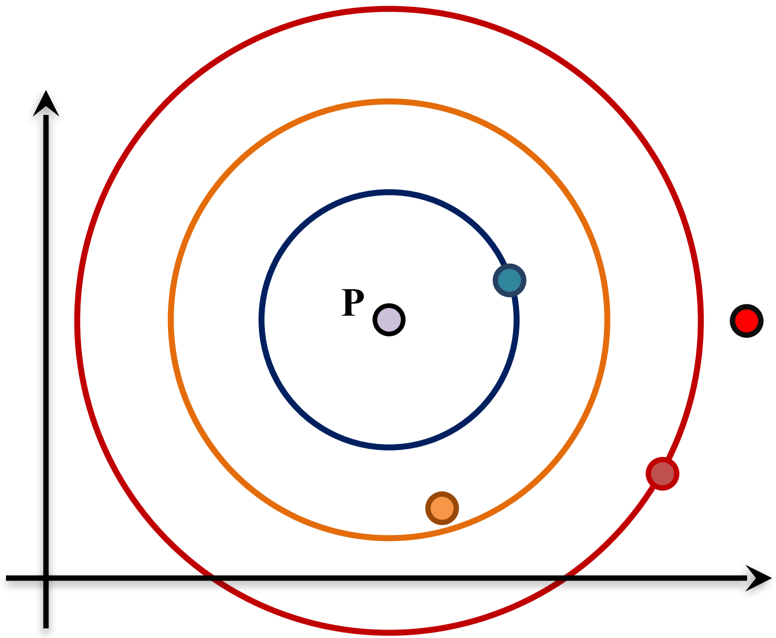
Distance measures



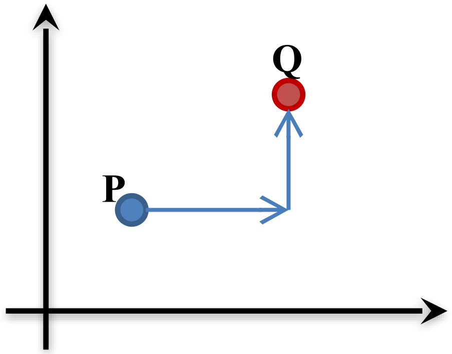
Euclidean Distance [L2]

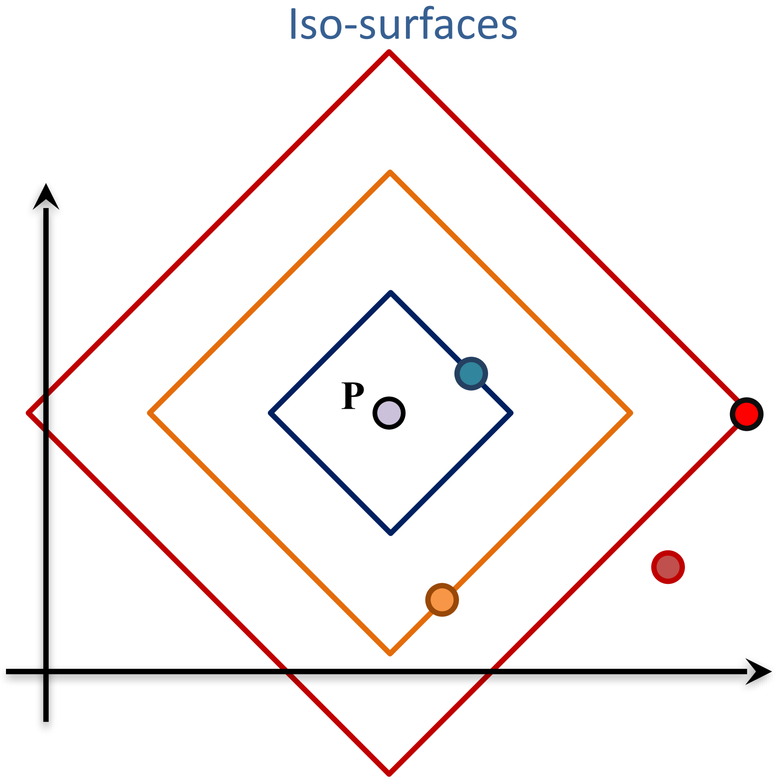


Iso-surfaces

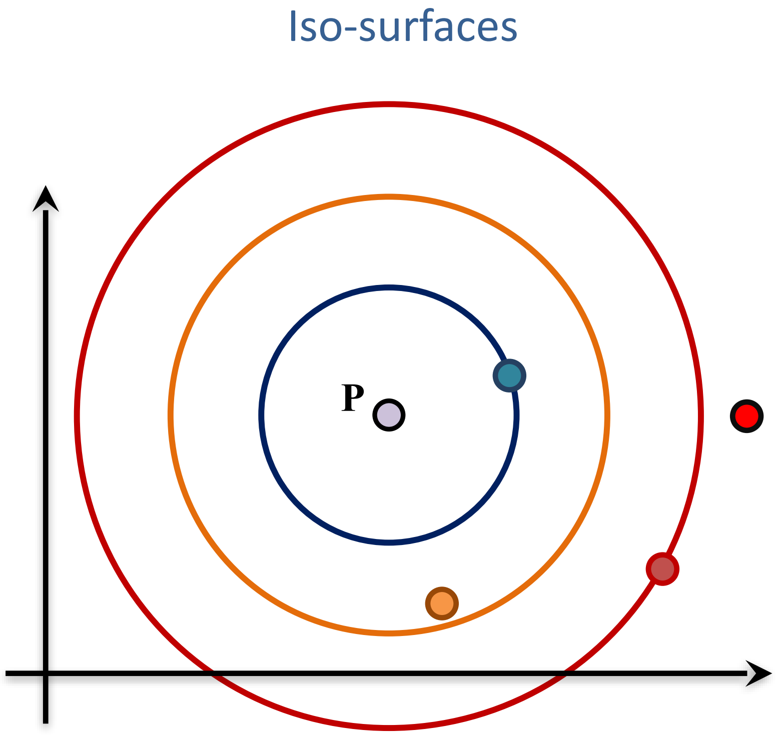


Manhattan Distance [L1]

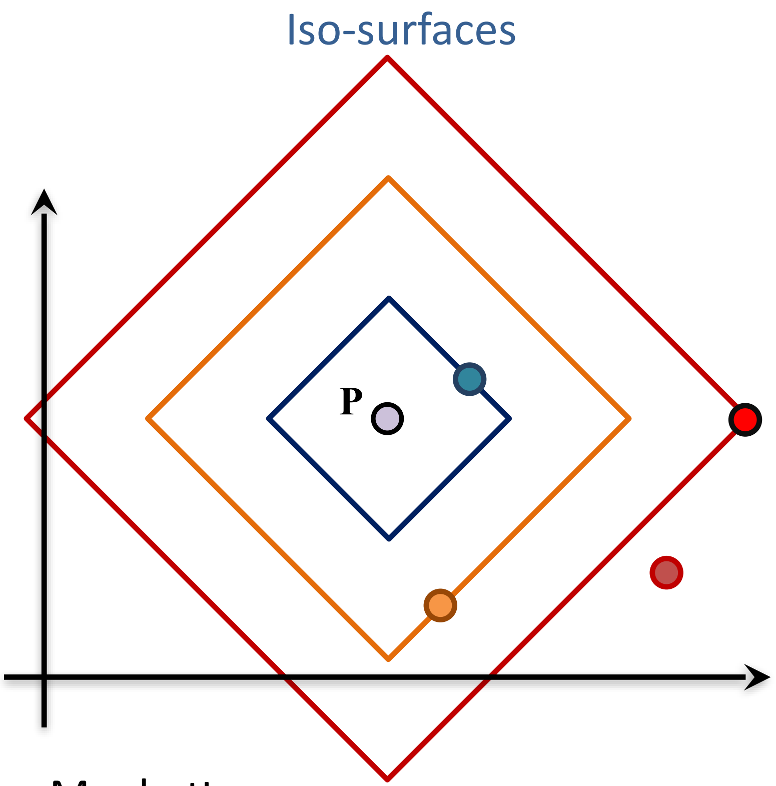




Distance measure can affect k-NN classification

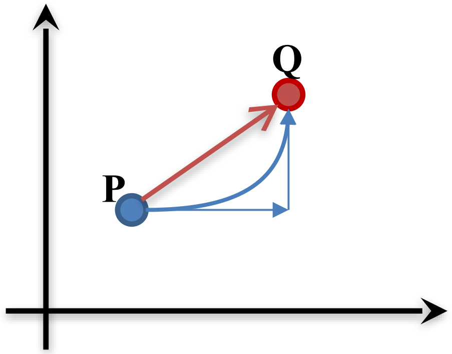


Euclidean



Manhattan

Minkowski Dist.



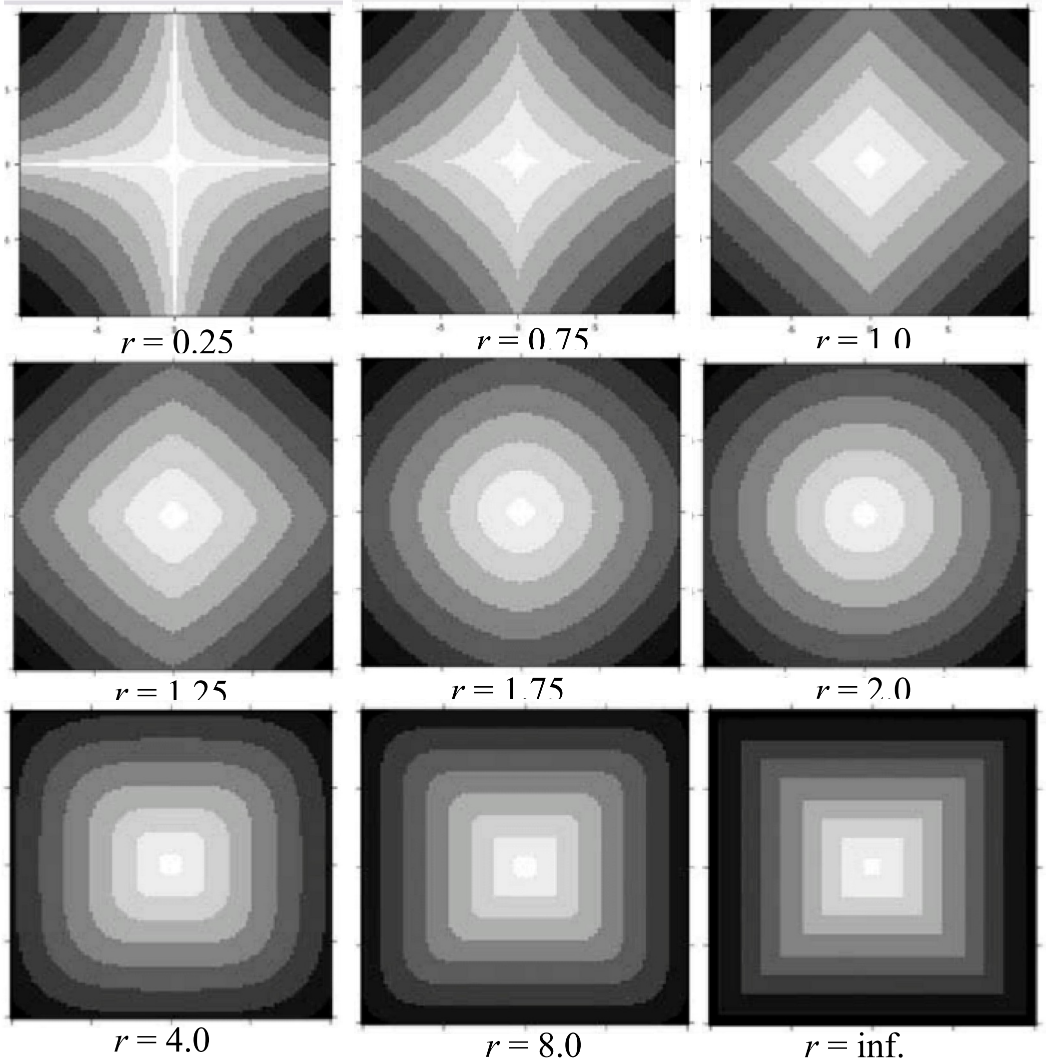
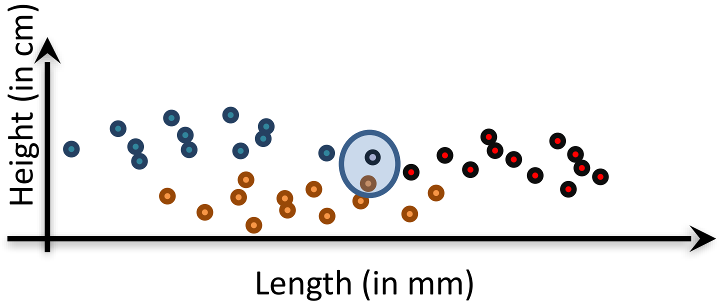
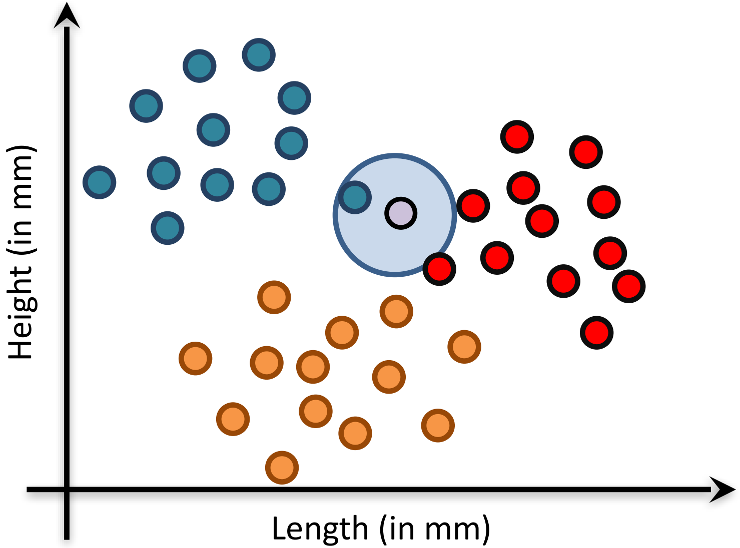


Fig. by Lu et al., "The Minkowski Approach for Choosing the Distance Metric in Geographically Weighted Regression"

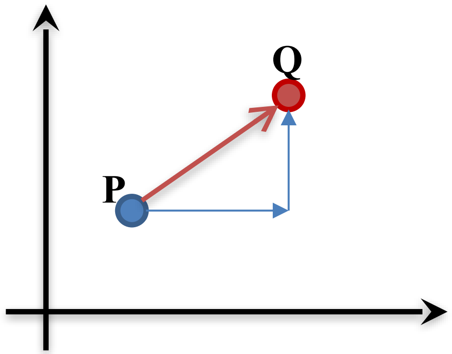
Note: Feature Normalization

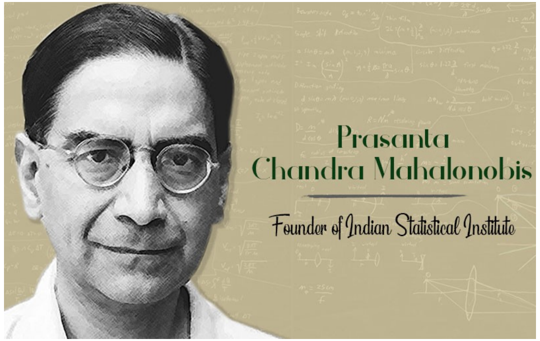
- If different features have different variances
  - Some features will dominate distance computation
  - Normalization can reduce this feature bias



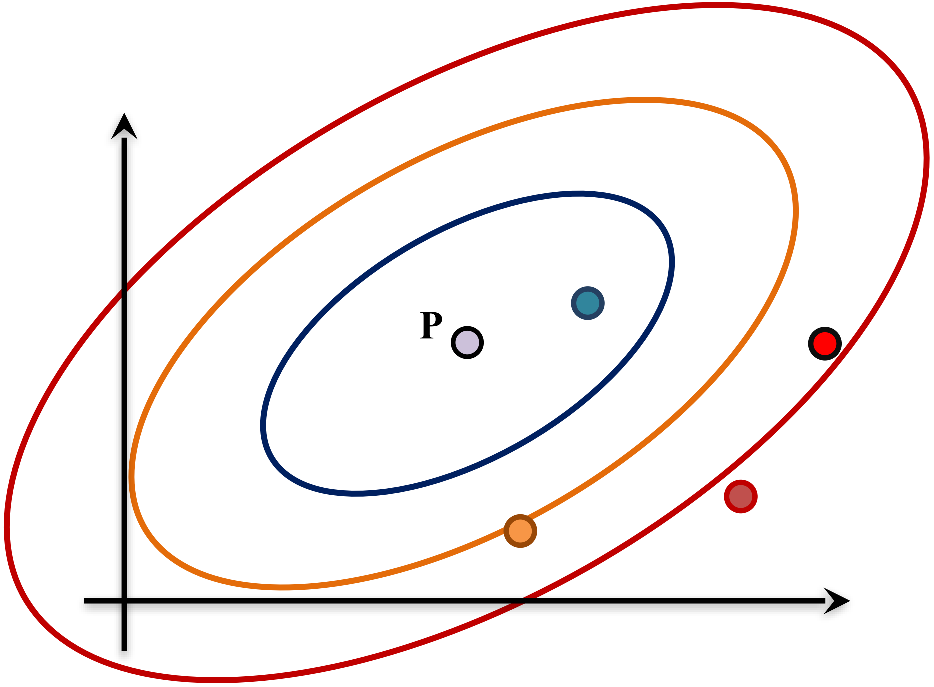


Mahalanobis Distance



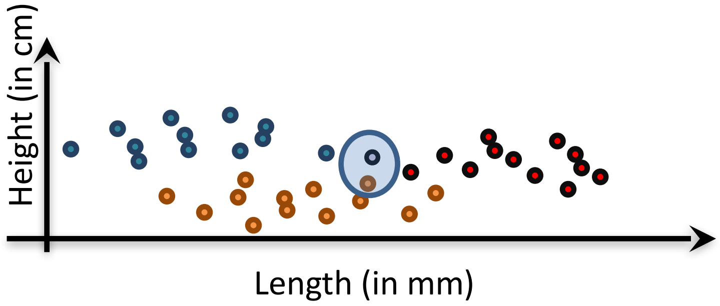


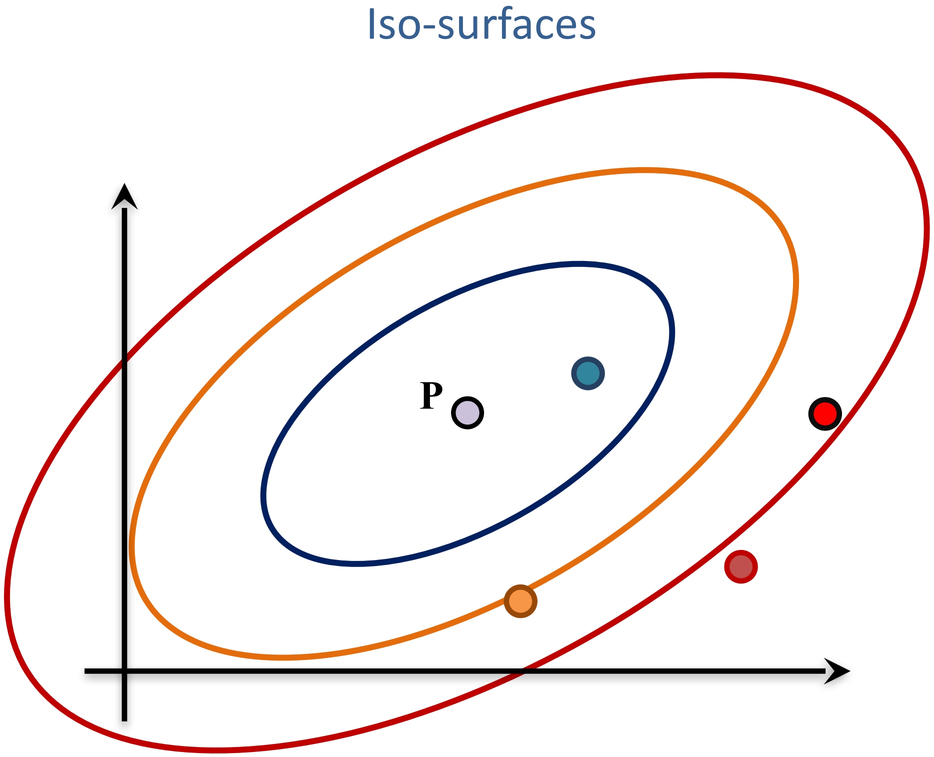
Iso-surfaces



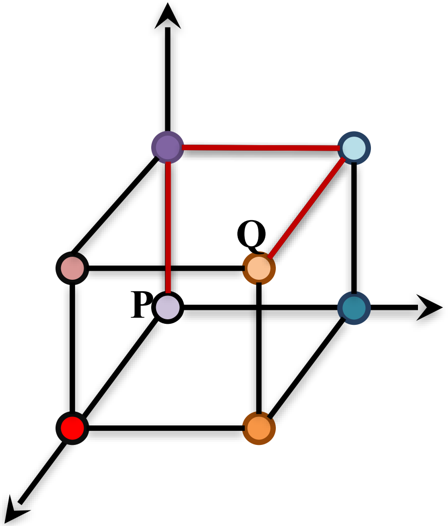
https://www.getbengal.com/details/the-indian-statistical-institute-in-kolkata-was-founded-by-the-father-of-indian-statistics-pc--mahalanobis-getbengal-story

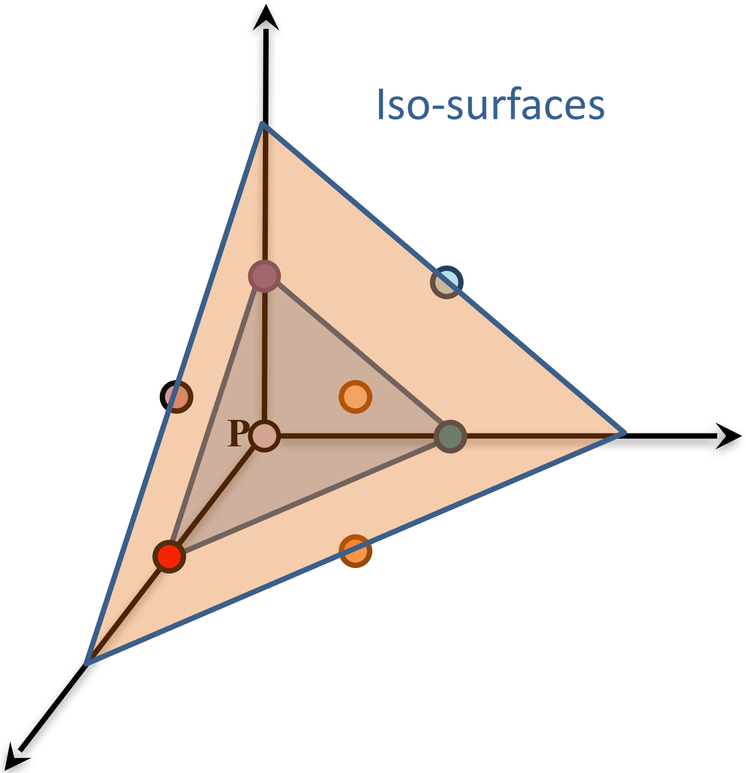
Mahalanobis Distance





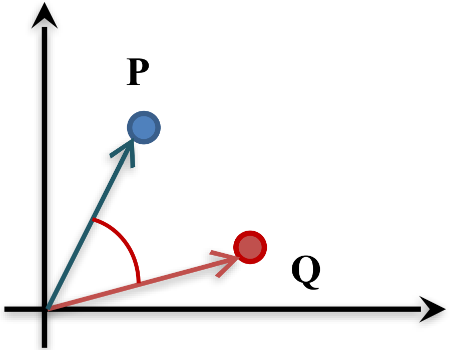
Hamming Distance



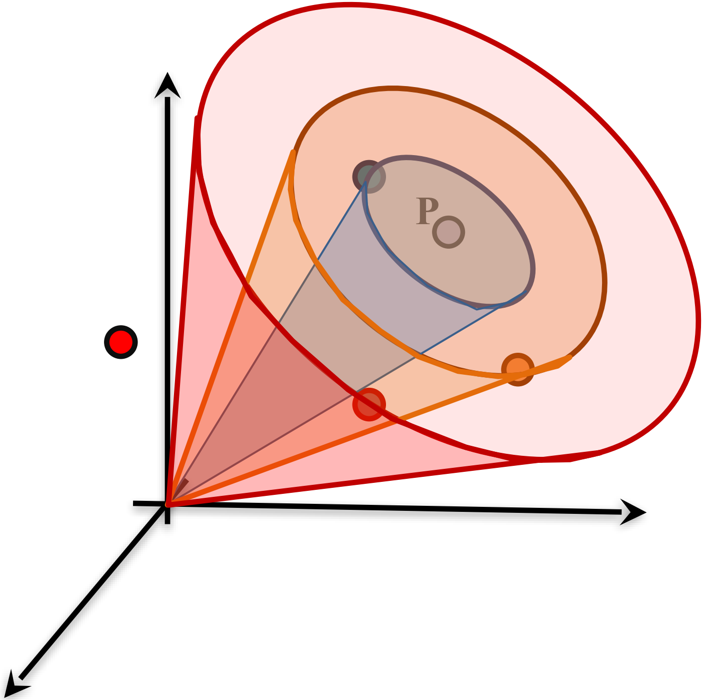


0	1	1	0	1	1	0
0	0	1	0	1	0	0

Cosine Distance



Iso-surfaces

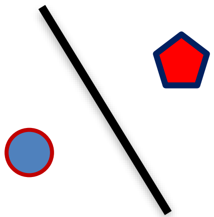


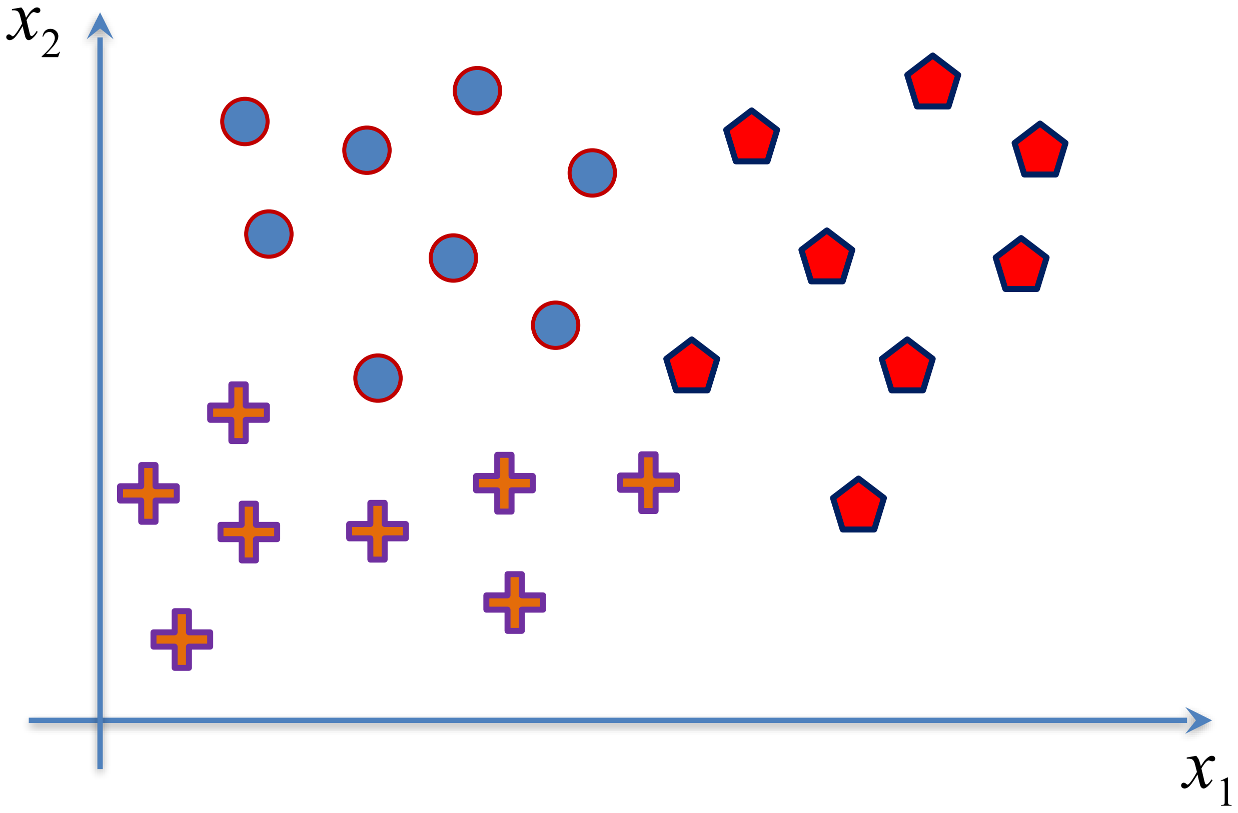
Summary of Distances

- Distance Metrics decide Neighborhoods
- Need not be a ”metric”
  - Jaccard Distance
  - Edit Distance
- Feature vectors need not be of same length
- Selection of metric depends on the nature of feature vector

Class Boundaries

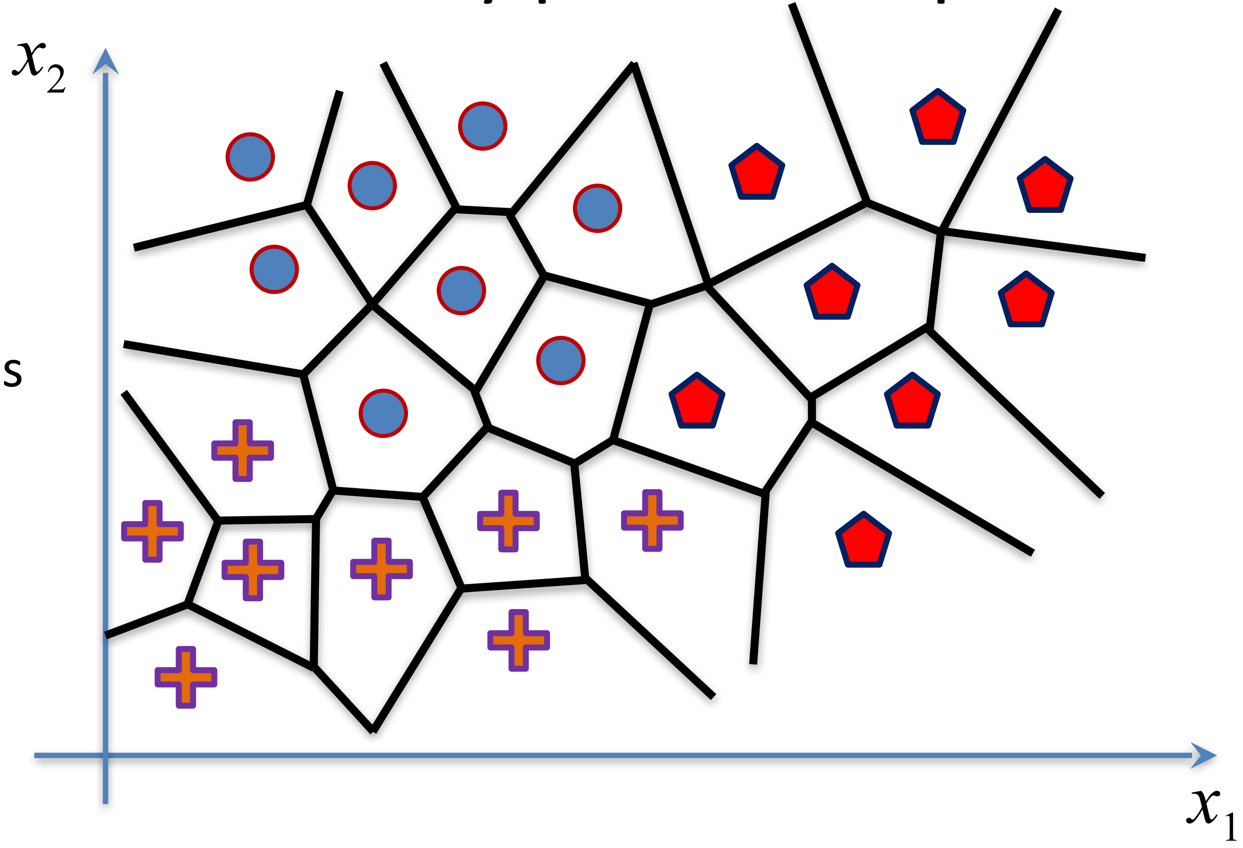
- Partition the feature space
- Consider 2 samples





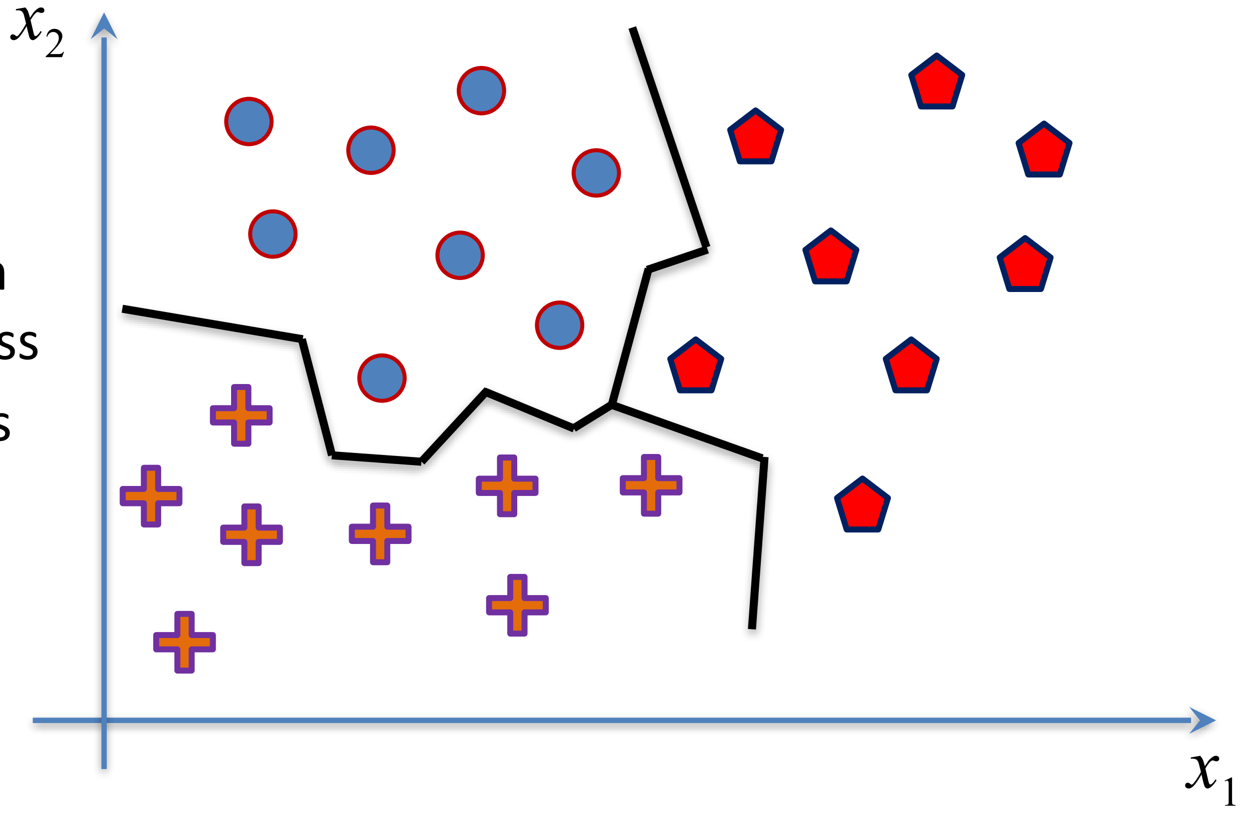
Boundaries between every pair of samples

- Voronoi Tessellation
- We can ignore boundaries between samples of same class

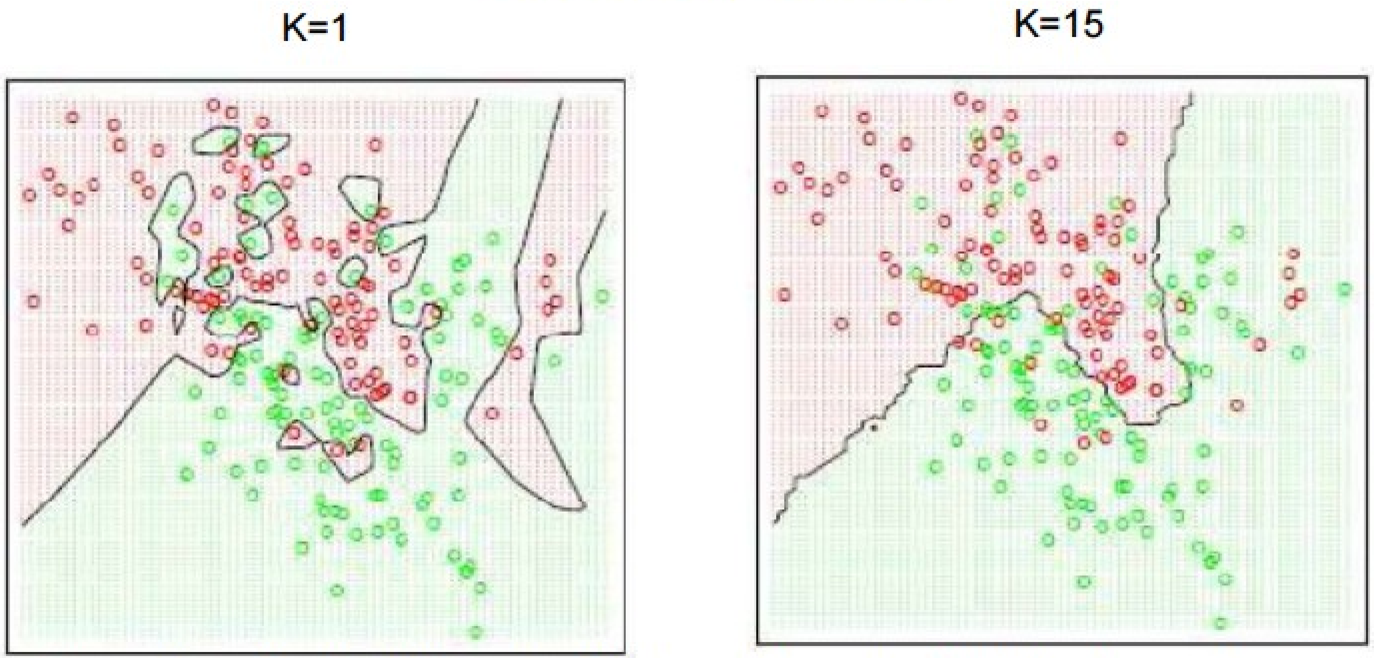


Boundaries between every pair of samples

- Voronoi Tessellation
- We can ignore boundaries between samples of same class
- Decision Boundary is piece-wise Linear

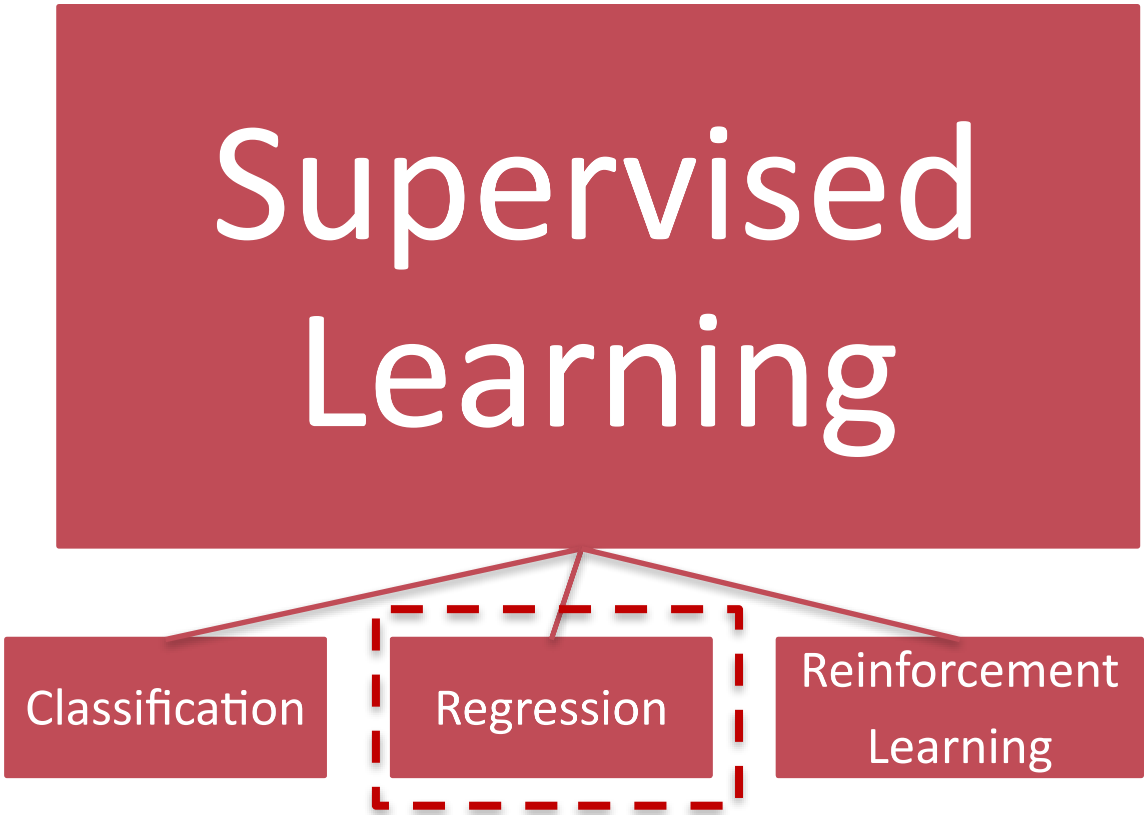


Effect of K



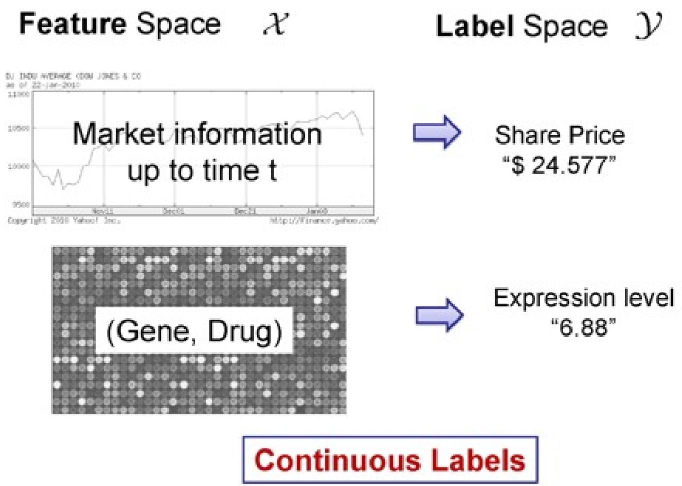
Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Larger k produces smoother boundary effect and can reduce the impact of class label noise.


Supervised Learning
Classification
Regression
Reinforcement Learning

Regression model

- Regression model
  - Explanatory variables: **independent** variables
  - Variables to be explained : **dependent** variables



Examples

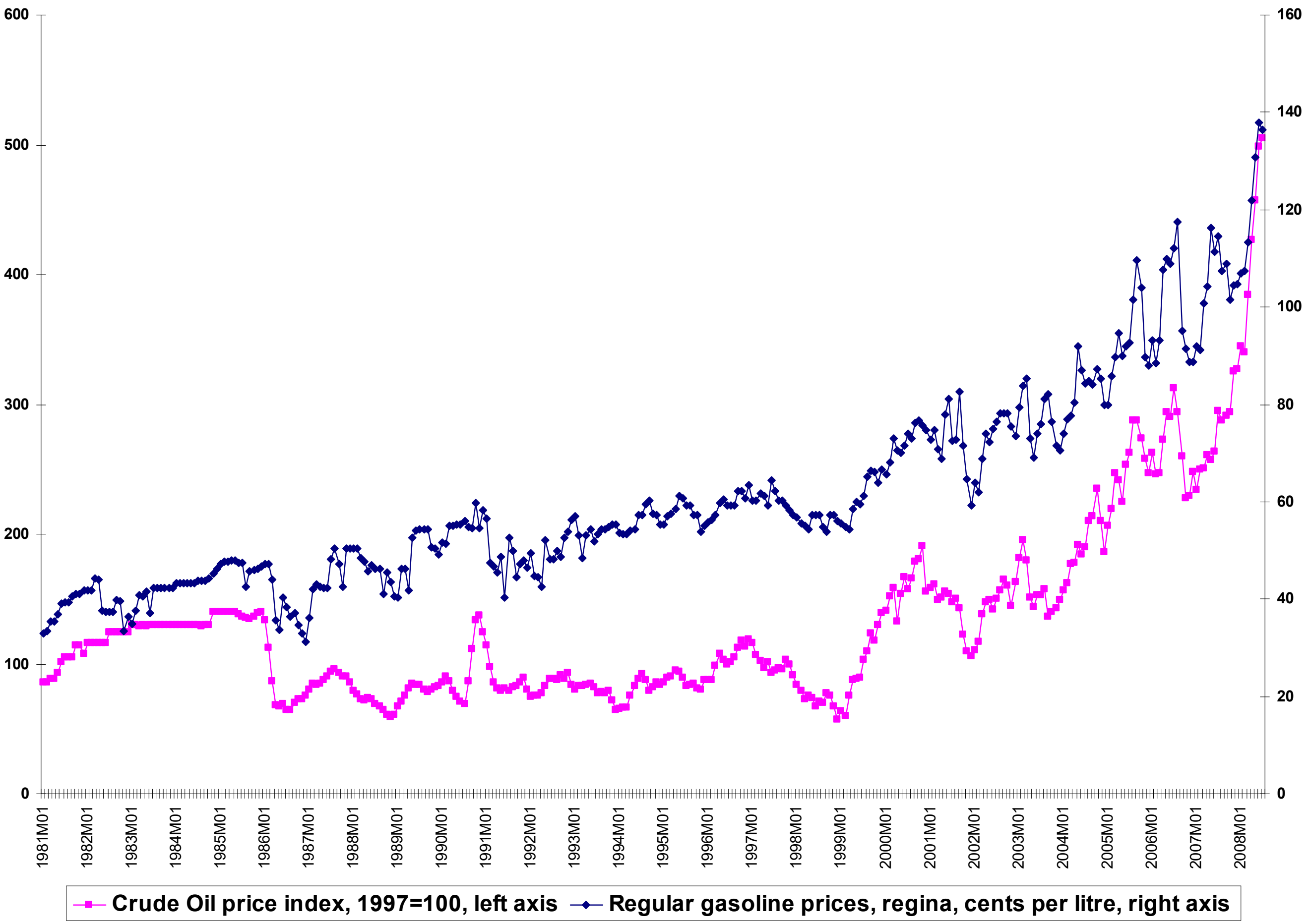
- Independent variable: Price of crude oil
- Dependent variable: Retail price of petrol

- Independent variables: hours of work, education, occupation, sex, age, years of experience etc.
- Dependent variable: Employment income

- Independent variables: Area of house, Population Density
- Dependent variable: Rent or Price of house

- Price of a product and quantity produced or sold:
  - Quantity sold affected by price. Dependent variable is quantity of product sold – independent variable is price.

  - Price affected by quantity offered for sale. Dependent variable is price – independent variable is quantity sold.



Source: CANSIM II Database (Vector v1576530 and v735048 respectively)

Linear Regression Model

- 1. Relationship Between Variables Is a Linear Function

Y-Intercept

Slope

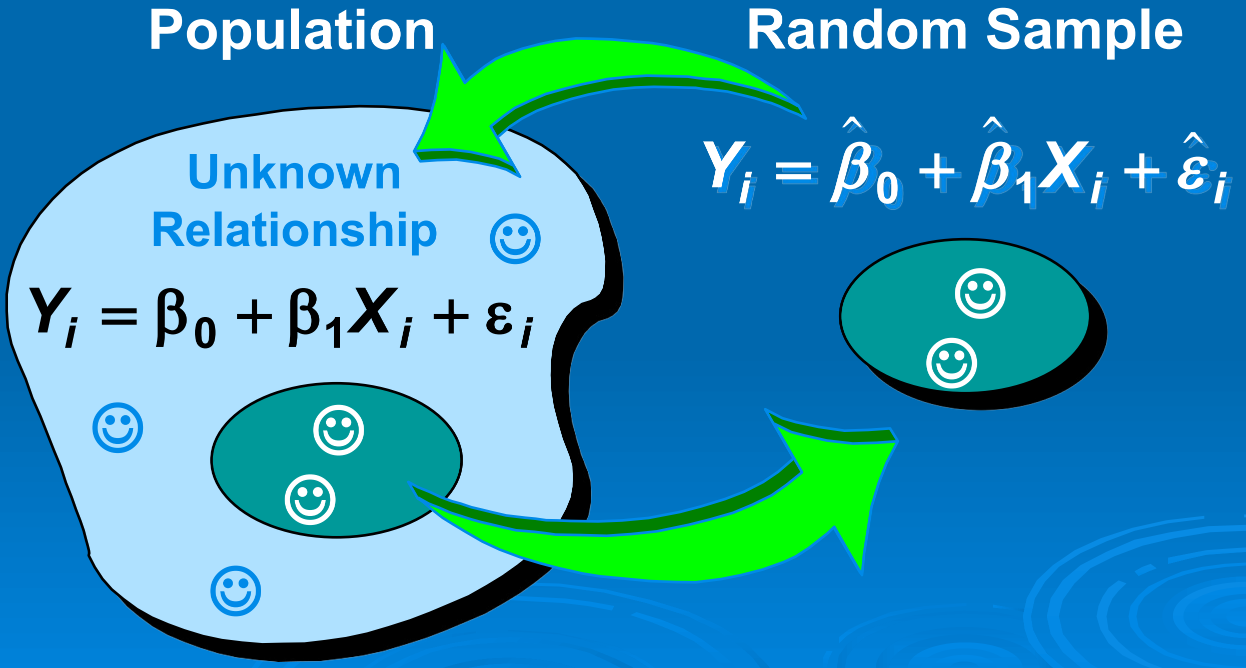
Random Error

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

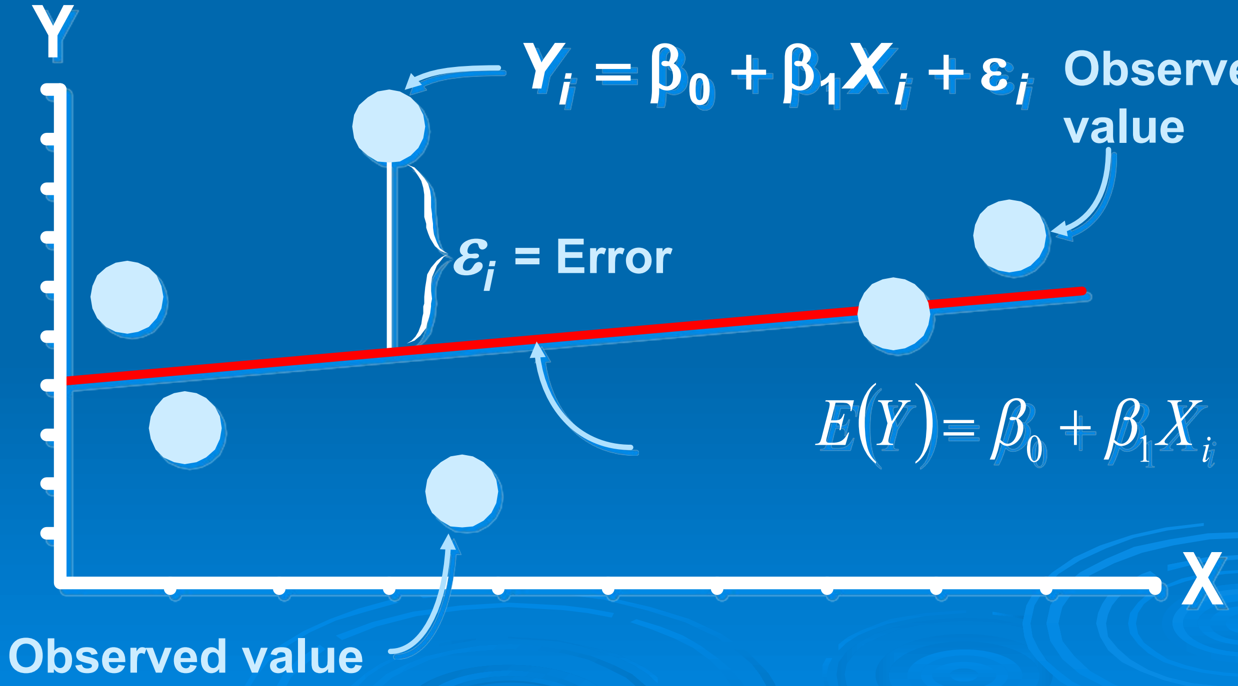
Dependent (Response) Variable (e.g. Salary)

Independent (Explanatory) Variable (e.g. Yrs of experience)

Population & Sample Regression Models



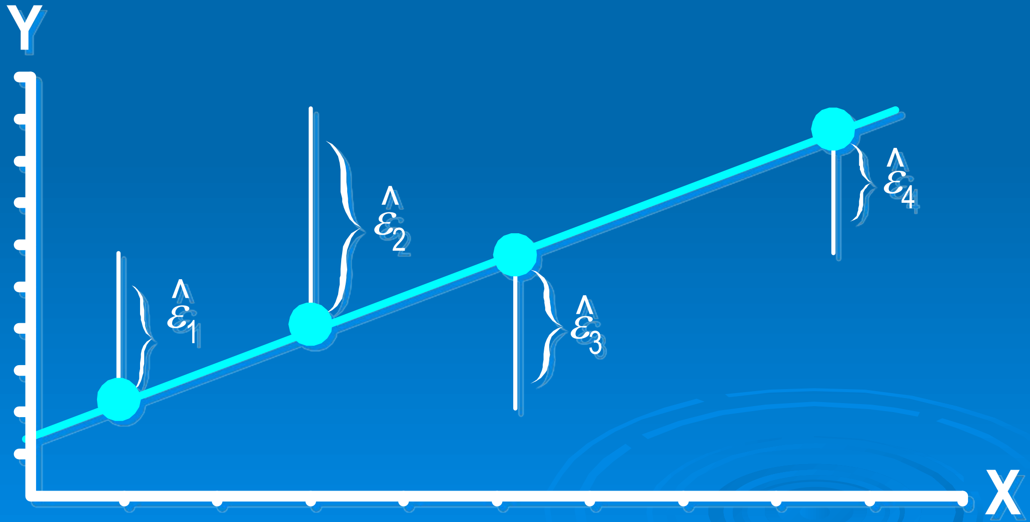
Linear Regression Model



Estimating Parameters: Least Squares Method

Least Squares

- 1. ‘Best Fit’ Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum. *But* Positive Differences Offset Negative ones

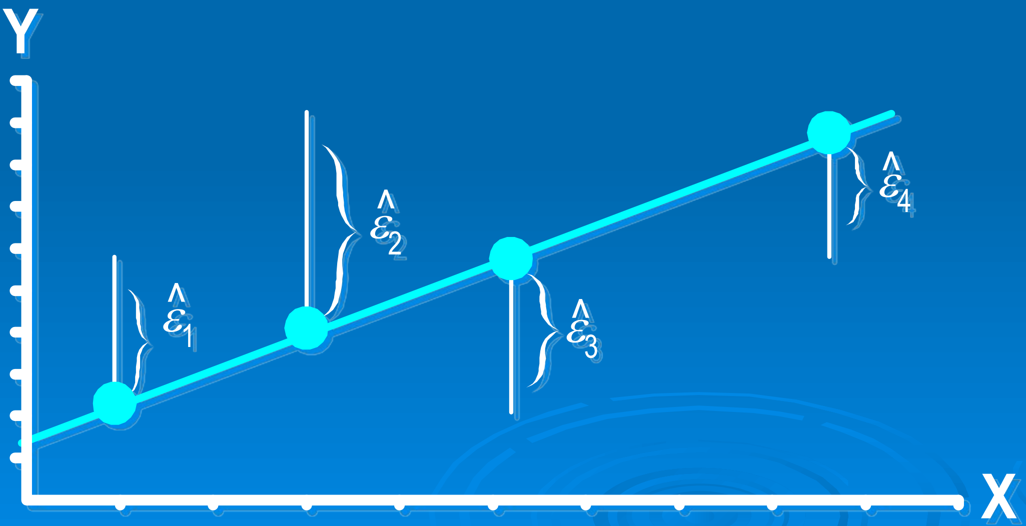


$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

$$E(Y) = \beta_0 + \beta_1 X_i$$

Least Squares

- 1. 'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values is a Minimum. *But* Positive Differences Offset Negative ones. So square errors!



$$\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

$$\mathbf{Y}_i = \beta_0 + \beta_1 \mathbf{X}_i + \varepsilon_i$$

$$E(Y) = \beta_0 + \beta_1 X_i$$

Least Squares

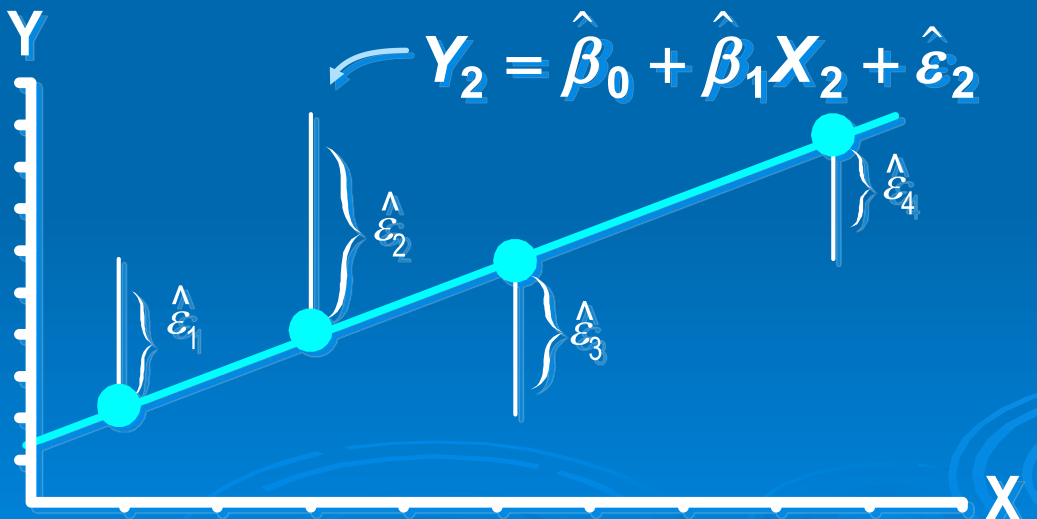
- 1. 'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum. *But* Positive Differences Off-Set Negative. So square errors!

$$\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

- 2. LS Minimizes the Sum of the Squared Differences (errors) (SSE)

Least Squares Graphically

LS minimizes $\sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \hat{\varepsilon}_1^2 + \hat{\varepsilon}_2^2 + \hat{\varepsilon}_3^2 + \hat{\varepsilon}_4^2$



Derivation of Parameters (1)

Least Squares (L-S):

Minimize squared error

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i \right)^2$$

Derivation of Parameters (1)

Least Squares (L-S):

Minimize squared error

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i \right)^2$$

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_0} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i \right)^2}{\partial \beta_0}$$

$$= -2\left(n\bar{y} - n\beta_0 - n\beta_1 \bar{x} \right)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Derivation of Parameters (1)

Least Squares (L-S):

Minimize squared error

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_1} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i\right)^2}{\partial \beta_1}$$

$$= -2 \sum x_i \left(y_i - \beta_0 - \beta_1 x_i\right)$$

$$= -2 \sum x_i \left(y_i - \overline{y} + \beta_1 \overline{x} - \beta_1 x_i\right)$$

$$\beta_1 \sum x_i \left(x_i - \overline{x}\right) = \sum x_i \left(y_i - \overline{y}\right)$$

$$\beta_1 \sum \left(x_i - \overline{x}\right)\left(x_i - \overline{x}\right) = \sum \left(x_i - \overline{x}\right)\left(y_i - \overline{y}\right)$$

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}}$$

Coefficient Equations

Prediction equation

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

Sample slope

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sum (x_i - \overline{x})^2}$$

Sample Y - intercept

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

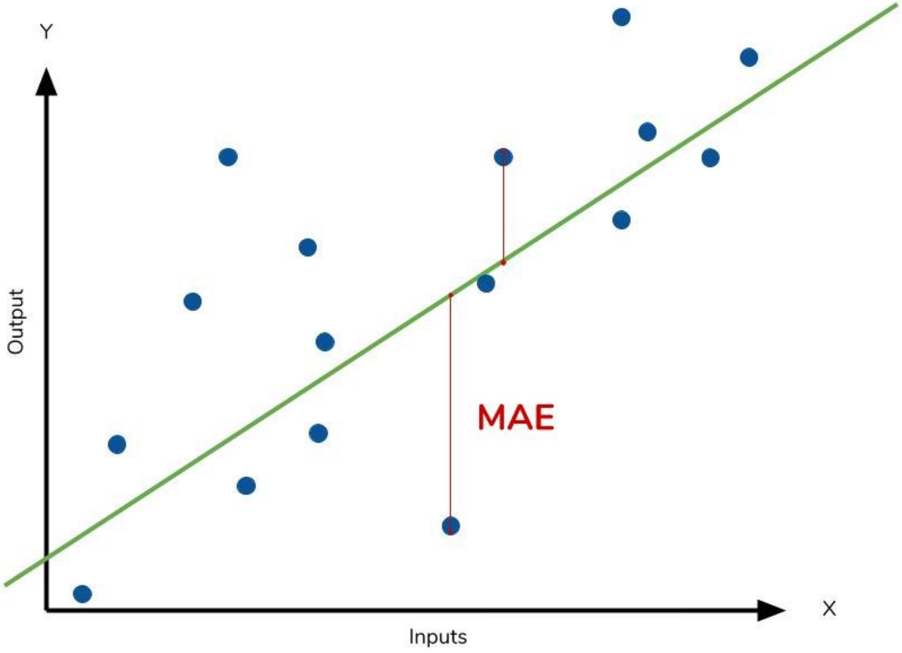
Regression – Error measures

Divide by the total number of data points

Predicted output value

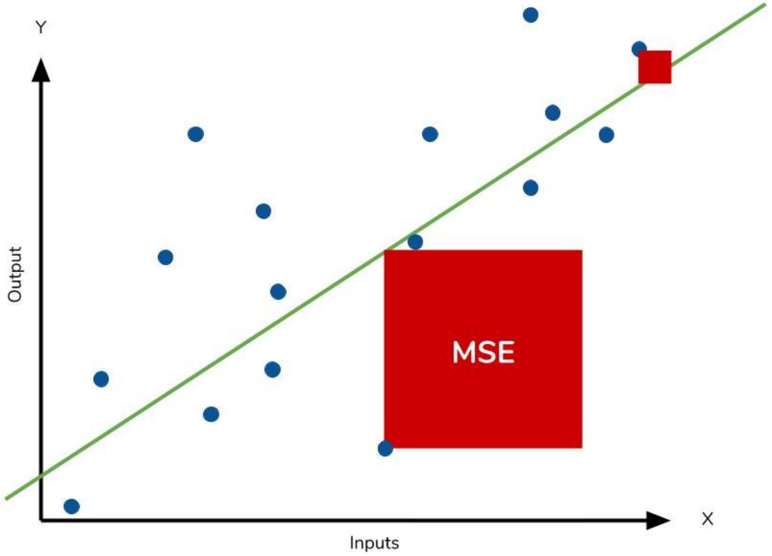
Actual output value

$$MAE = \boxed{\frac{1}{n}} \underset{\text{Sum of}}{\sum} \underbrace{\left| \boxed{y} - \boxed{\hat{y}} \right|}_{\text{The absolute value of the residual}}$$

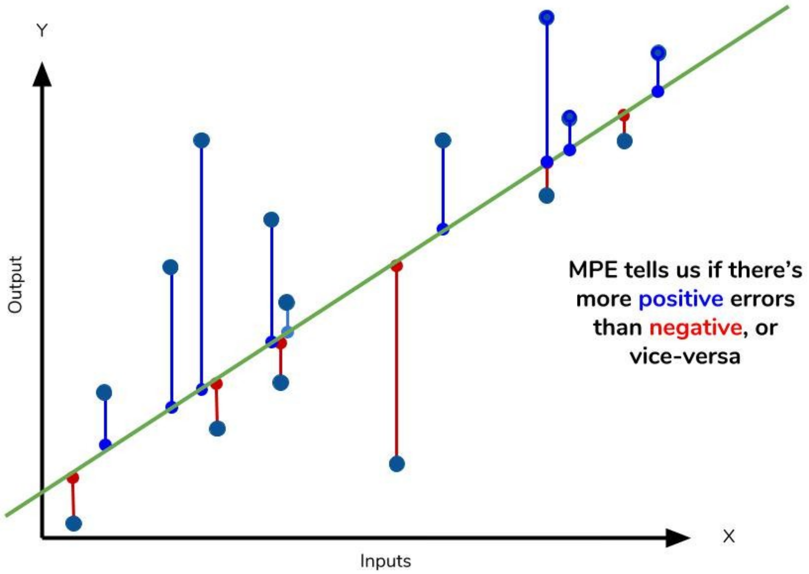


Regression – Error measures

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)}_{\text{The square of the difference between actual and predicted}}^{2}$$



$$MPE = \frac{100\%}{n} \Sigma \left(\frac{y - \widehat{y}}{y} \right)$$



Linear Regression – Matrix Form

Consider the model

$$Y = X\beta + \epsilon$$

where $Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}$ $\quad X = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix}$ $\quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$ $\quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$

Linear Regression – Matrix Form

Consider the model

$$Y = X\beta + \epsilon$$

where $Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}$ $X = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix}$ $\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$ $\epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$

Then using matrix calculus we find that the least squares estimate for β is given by

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Hence, the least squares regression line is $\hat{Y} = X\hat{\beta}$.

Influence Matrix

Linear Regression – Matrix Form - Issues

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Hence, the least squares regression line is $\hat{Y} = X\hat{\beta}$.

- N samples, p-dimensional (what if p > N ?)
- Complexity of matrix inversion (what if N very large ?)
- Collinearity

Gradient Descent

$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\begin{aligned} \boxed{\mathcal{L}(\mathbf{w})} &= \mathcal{L}(\beta_0, \beta_1) \\ &= \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2 \end{aligned}$$

1. Initialize the parametersto some random values.
2. Update the parameters using gradient descent rule

3. Repeat 2 until is close to 0

Gradient Descent

1. Initialize the parameters to some random values.
2. Update the parameters using gradient descent rule

3. Repeat 2 until is close to 0

$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\mathcal{L}(\beta_0, \beta_1)$$

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

$$= \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 \mathbf{x} - y)^2$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n} 2(\beta_0 + \beta_1 \mathbf{x} - y)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}(\beta_0 + \beta_1 \mathbf{x} - y)$$

Gradient Descent

1. Initialize the parameters to some random values.
2. Update the parameters using gradient descent rule

3. Repeat 2 until is close to 0

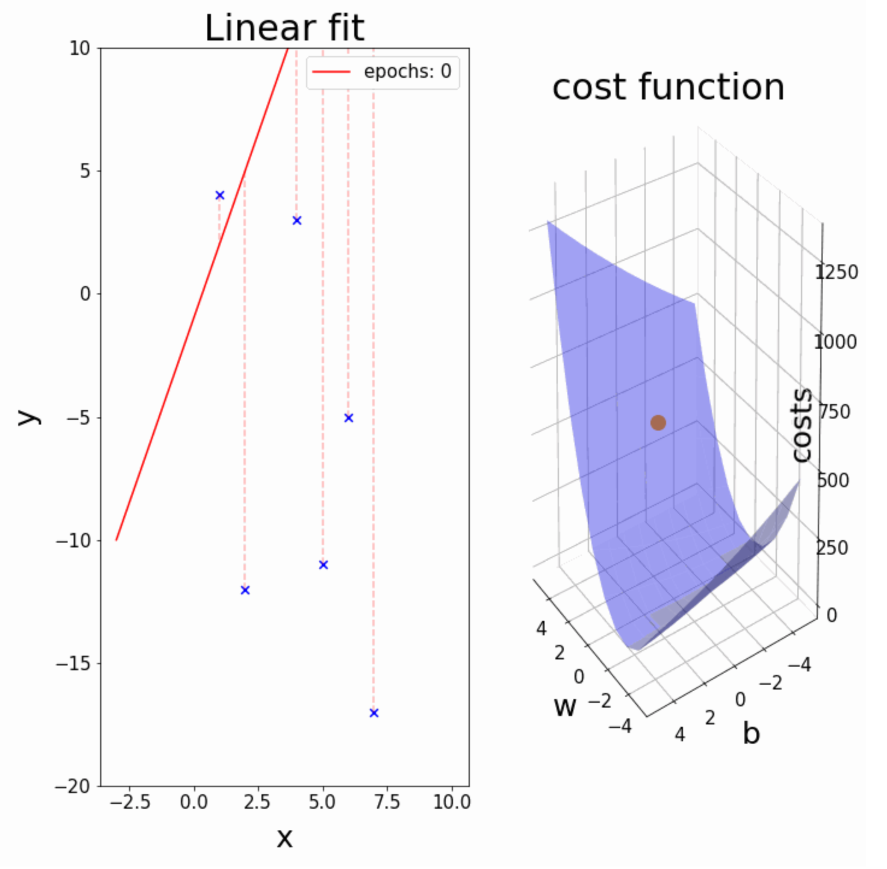
$$\mathbf{w} = [\beta_0 \;\; \beta_1]$$

$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \mathcal{L}(\beta_0, \beta_1) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 \mathbf{x} - y)^2
\end{aligned}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n} 2(\beta_0 + \beta_1 \mathbf{x} - y)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}(\beta_0 + \beta_1 \mathbf{x} - y)$$



Gradient Descent

1. Initialize the parameters to some random values.
2. Update the parameters using gradient descent rule

3. Repeat 2 until is close to 0

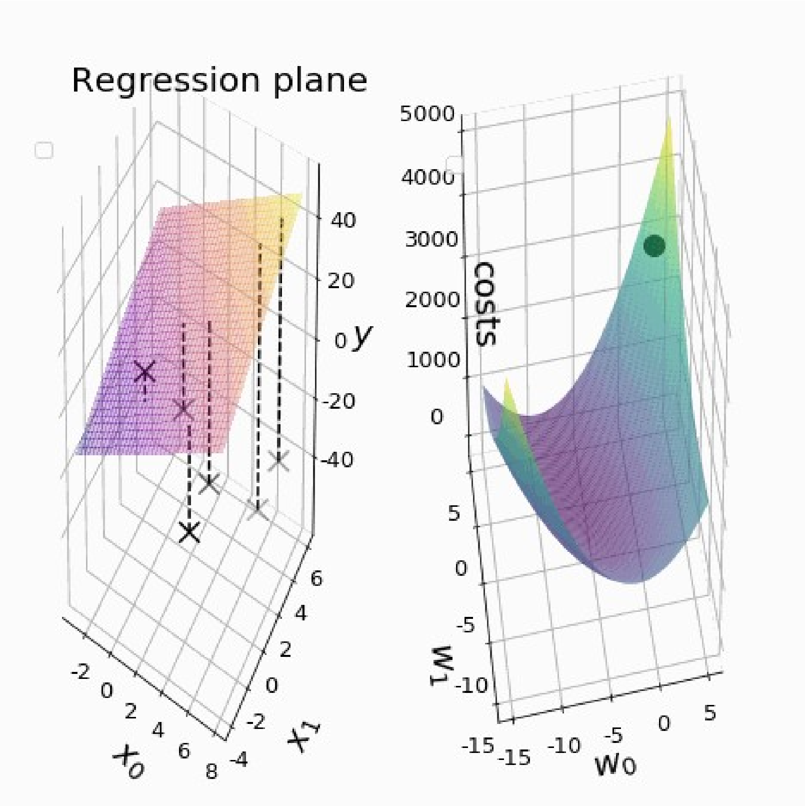
$$\mathbf{w} = [\beta_0 \;\; \beta_1]$$

$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \mathcal{L}(\beta_0, \beta_1) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 \mathbf{x} - y)^2
\end{aligned}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n} 2(\beta_0 + \beta_1 \mathbf{x} - y)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}(\beta_0 + \beta_1 \mathbf{x} - y)$$



Linear Regression

- Linear Regression → Linear in coefficients and NOT variables

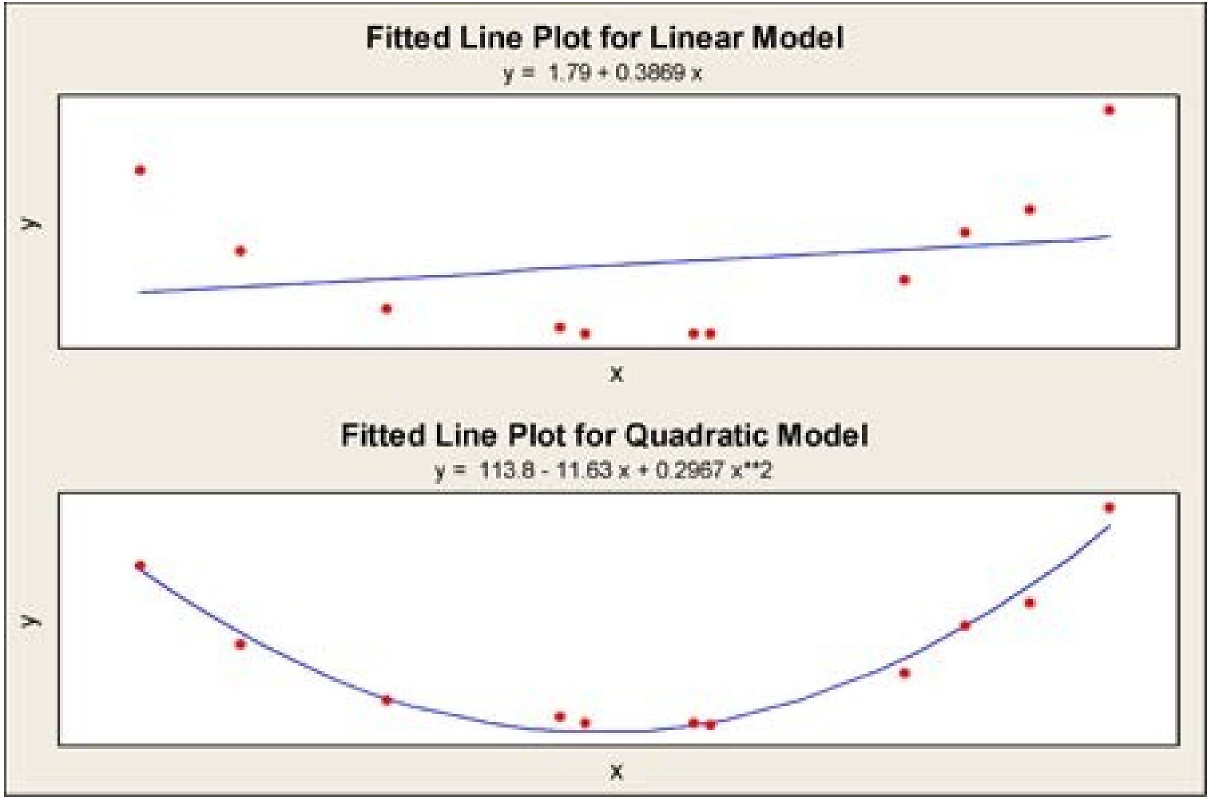
- A second-order model (quadratic model):

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

- β_1: Linear effect parameter.
- β_2: Quadratic effect parameter.

kth order polynomial model in one variable

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \epsilon$$



Fitted Line Plot for Linear Model
y = 1.79 + 0.3869 x
y
x
Fitted Line Plot for Quadratic Model
y = 113.8 - 11.63 x + 0.2967 x**2
y
x

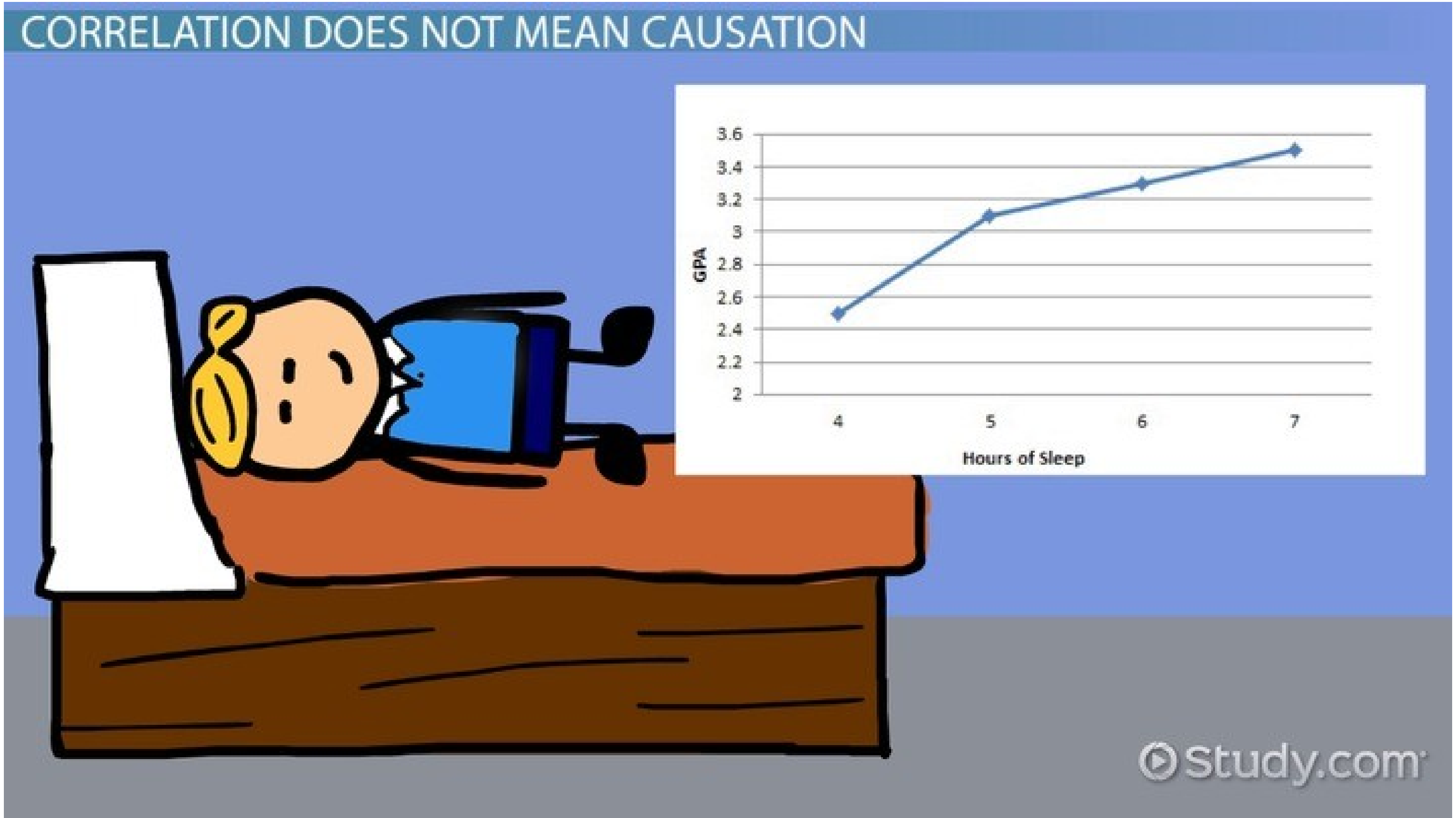
A quadratic polynomial regression function

$$Y_i = \beta_0 + \beta_1 X_i + \beta_{11} X_i^2 + \varepsilon_i$$

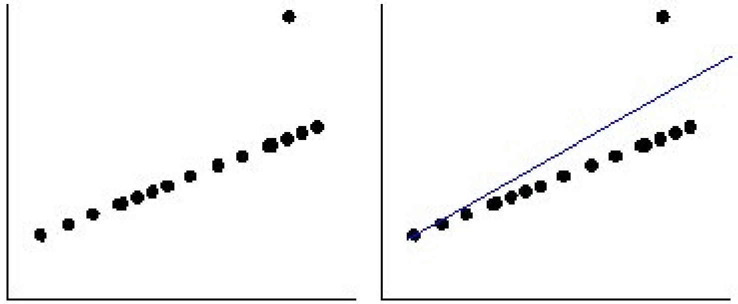
where:

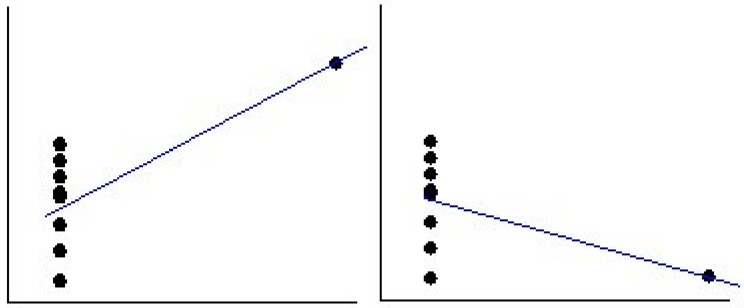
- Y_i = amount of immunoglobin in blood (mg)
- X_i = maximal oxygen uptake (ml/kg)
- typical assumptions about error terms ("INE")

Careful: X may not be causing y !

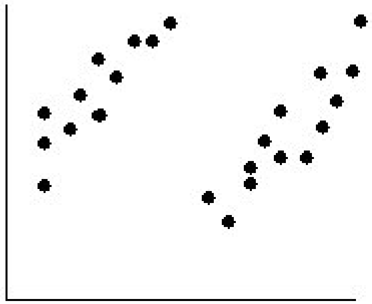


Linear Regression – Outliers

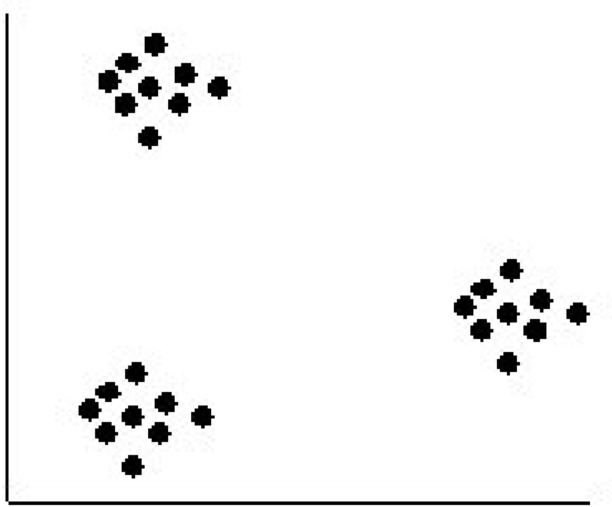


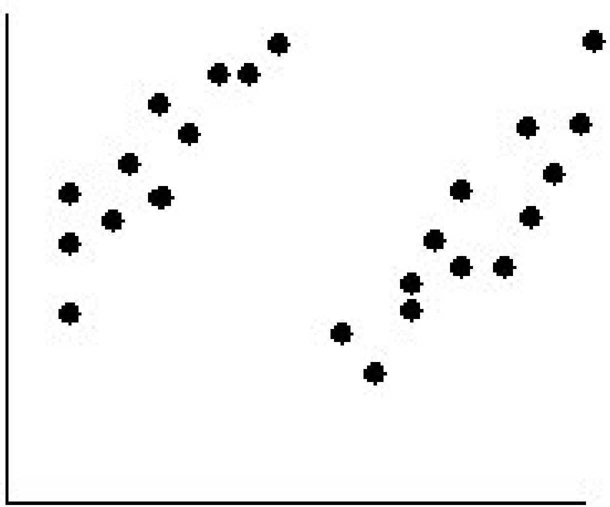


Linear Regression is problematic in many other cases

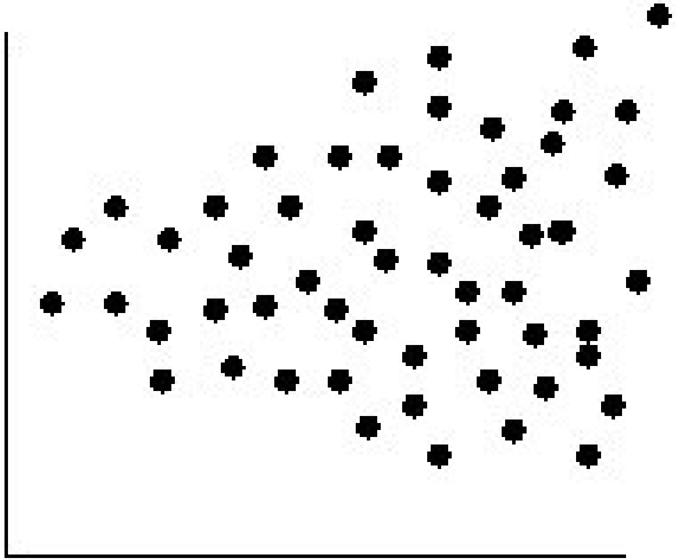


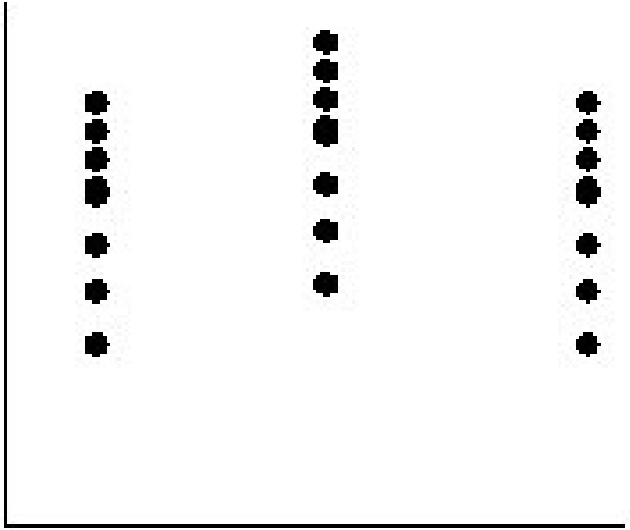
Linear Regression is problematic in many other cases



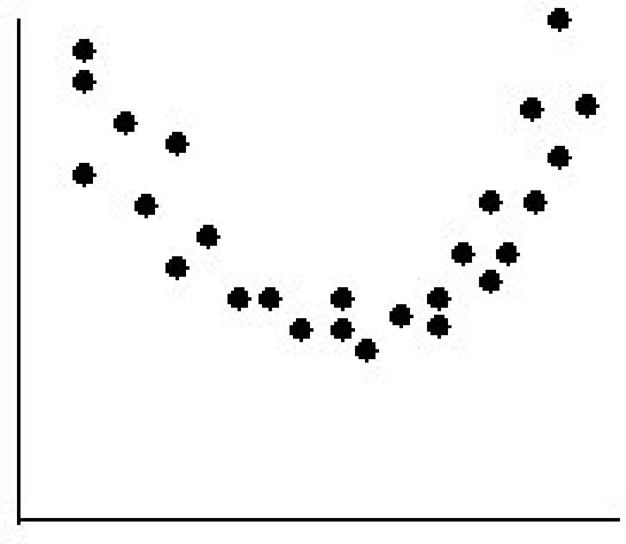


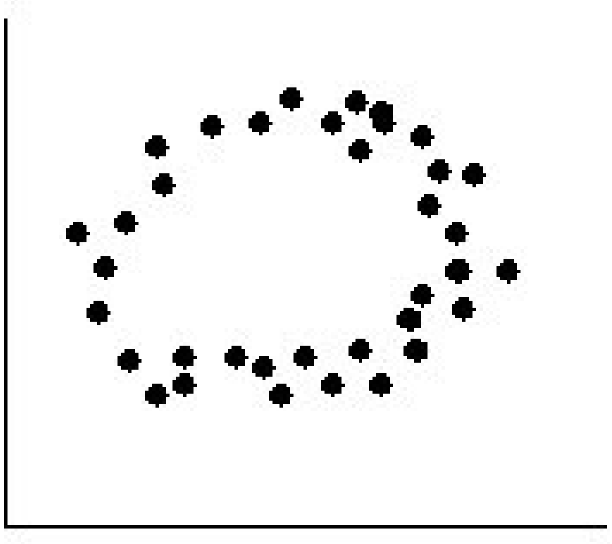
Linear Regression is problematic in many other cases



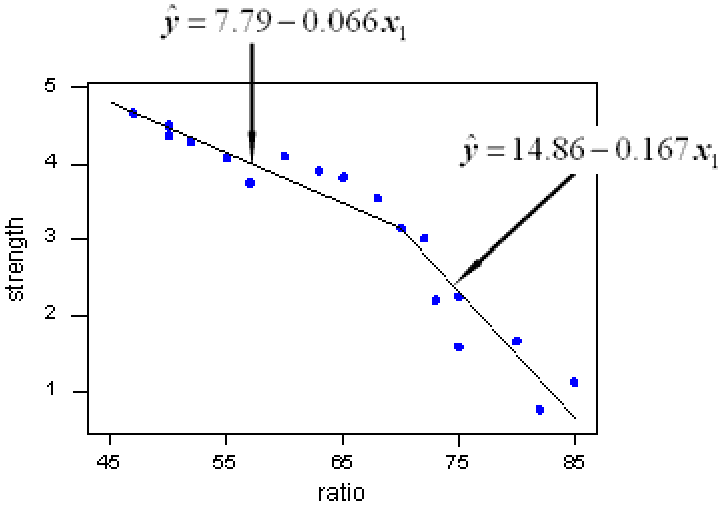


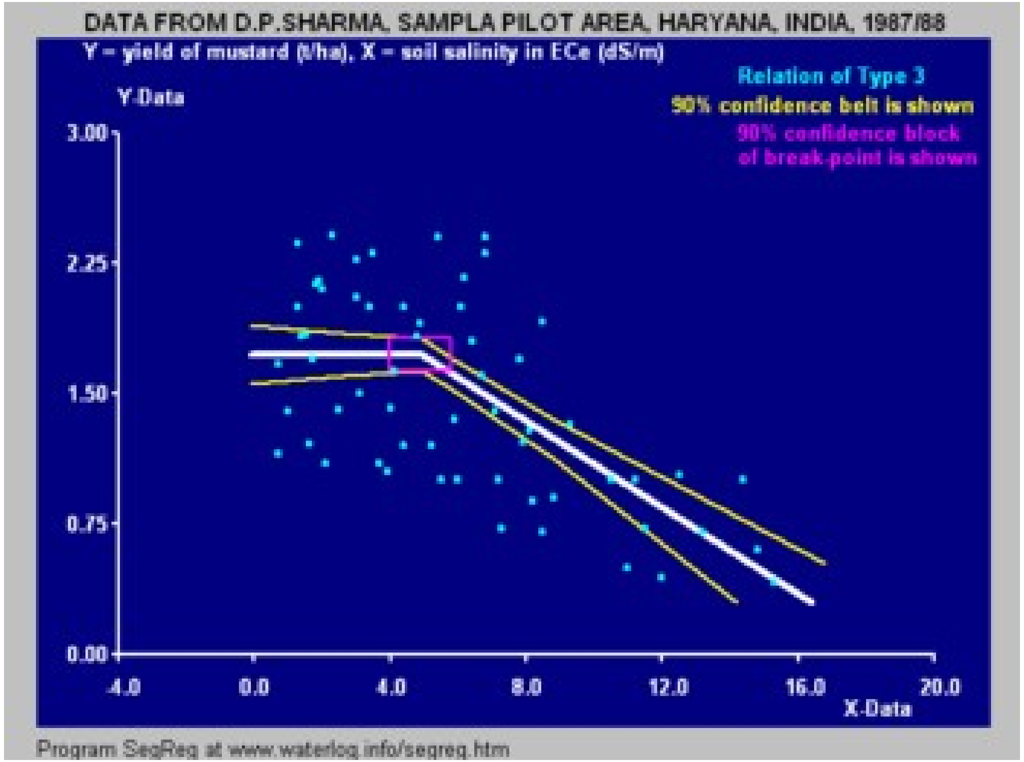
Linear Regression is problematic in many other cases





Piecewise Linear Regression





Also ref: Multivariate Adaptive Regression Splines (MARS)

Bivariate and multivariate models

Bivariate or simple regression model

(Education) $x \longrightarrow y$ (Income)

Multivariate or multiple regression model

(Education) x_1

(Gender) x_2

(Experience) x_3

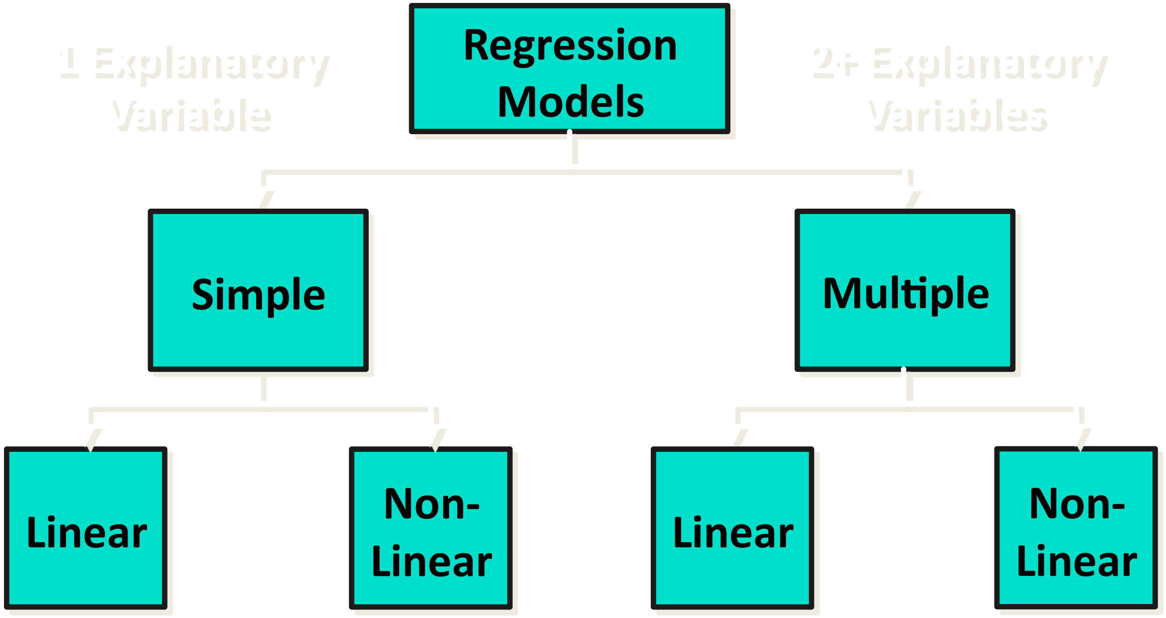
(Age) x_4

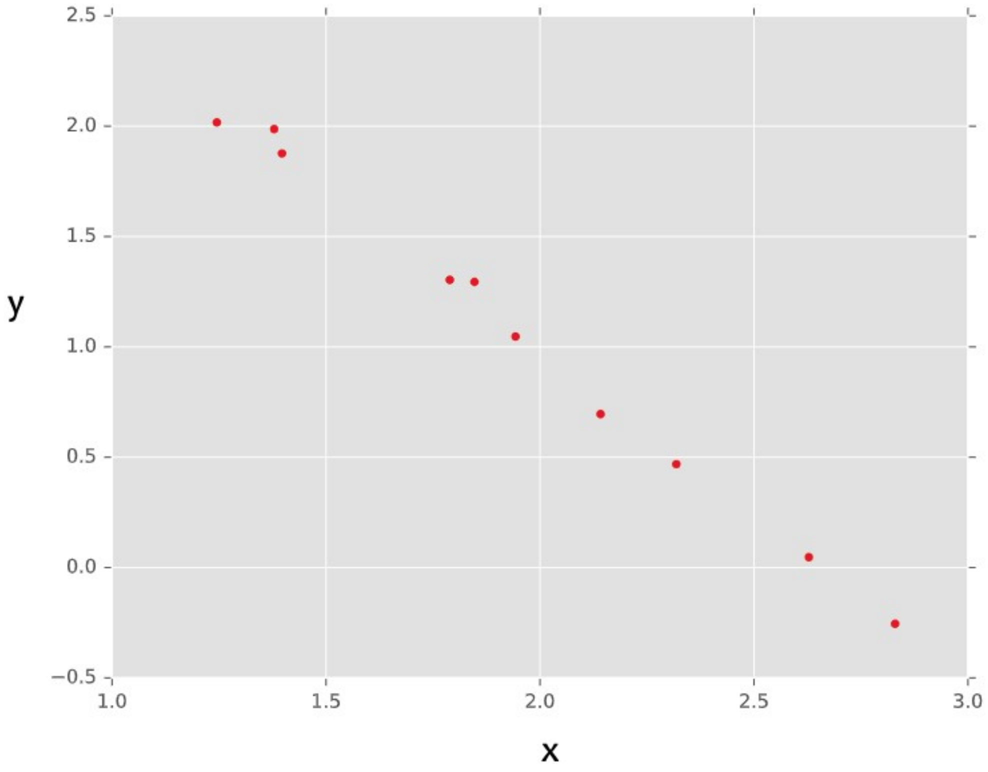
y (Income)

Model with simultaneous relationship

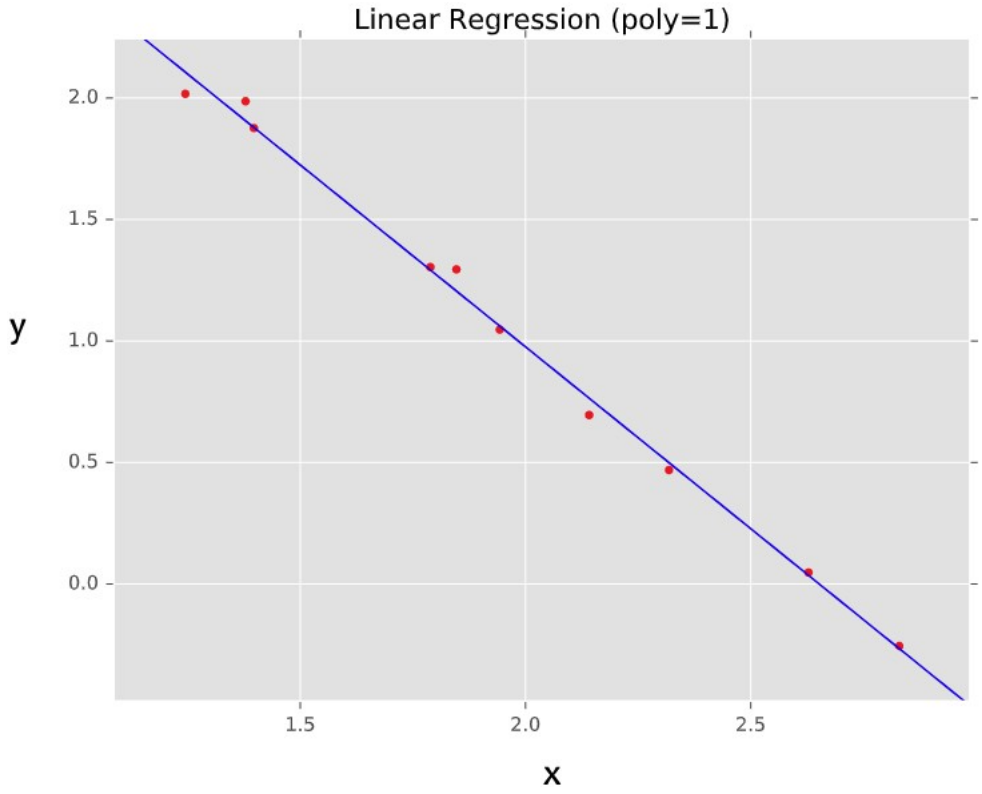
Price of wheat $\longleftrightarrow$ Quantity of wheat produced

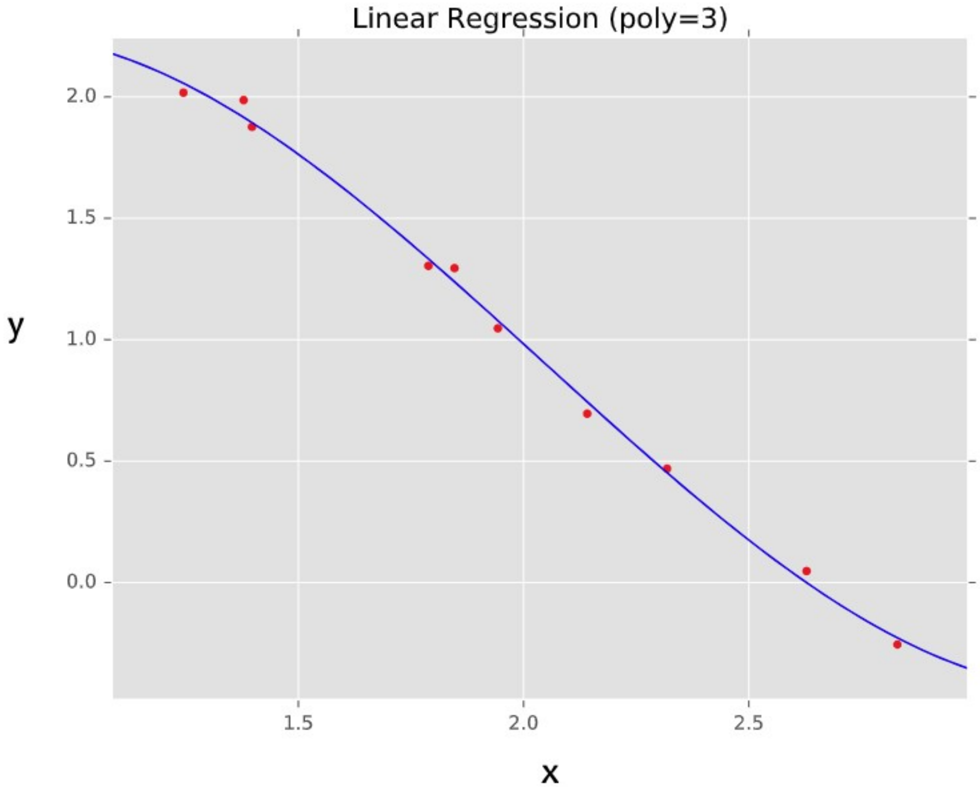
Types of
Regression Models

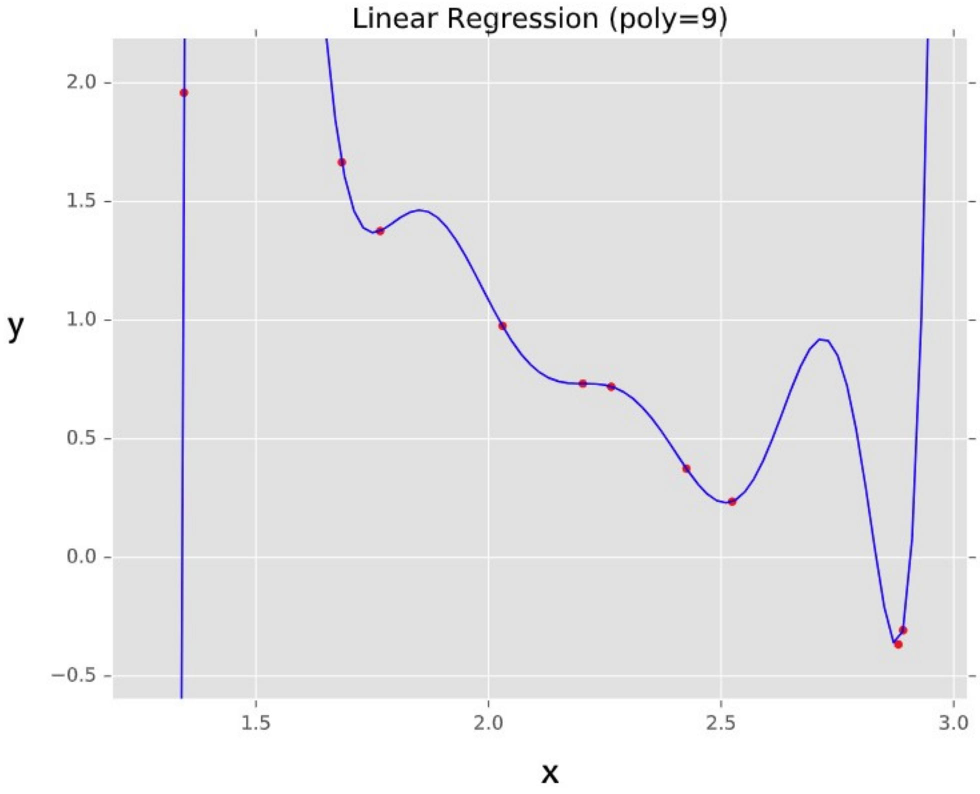




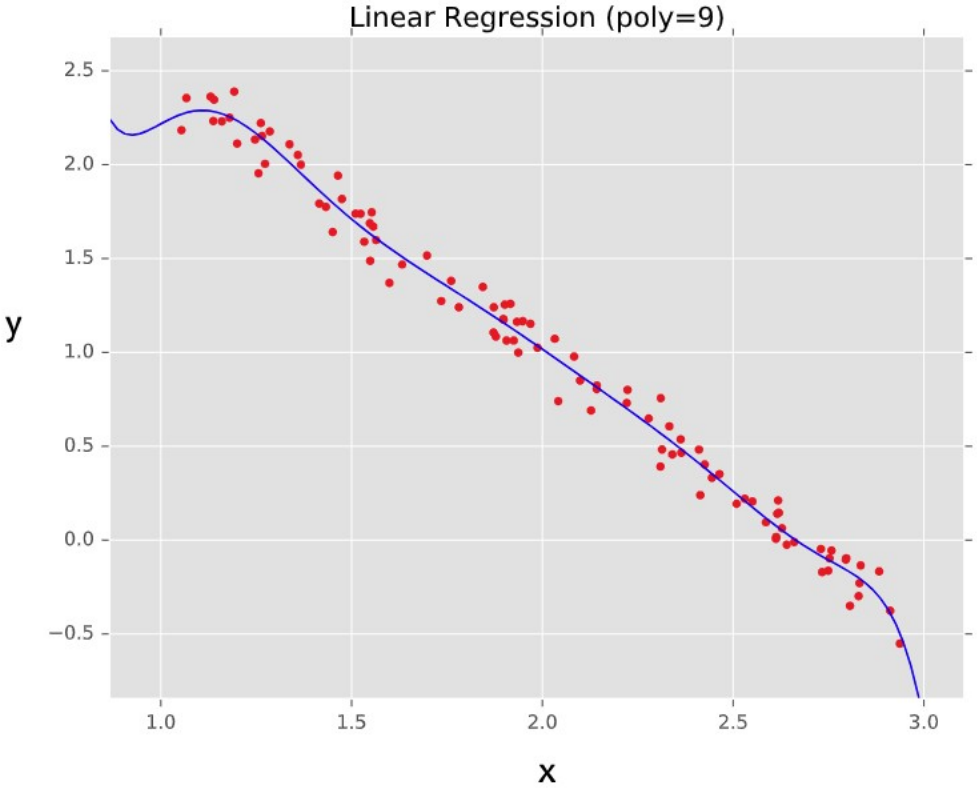
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
1.0
1.5
2.0
2.5
3.0
y
x


Linear Regression (poly=1)
y
x
2.0
1.5
1.0
0.5
0.0
1.5
2.0
2.5


Linear Regression (poly=3)
2.0
1.5
1.0
0.5
0.0
y
1.5
2.0
2.5
x


Linear Regression (poly=9)
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
1.5
2.0
2.5
3.0

Test Time



Overfitting

The example shows:

- perfect fit on **in sample (training)** data

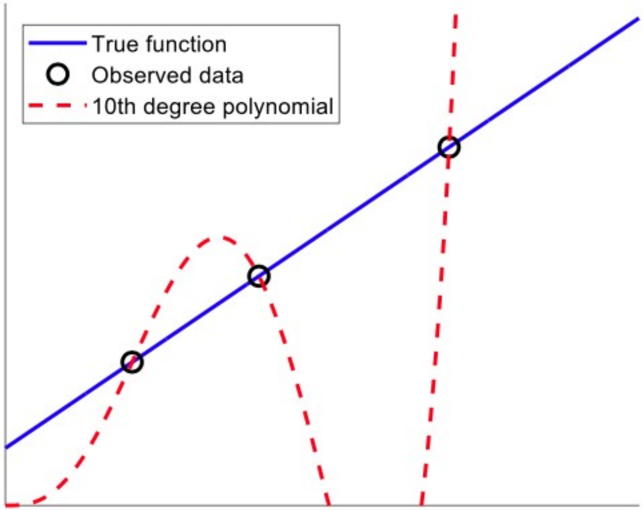
$$\downarrow$$

$$E_{in} = 0$$

- low fit on **out of sample (test)** data

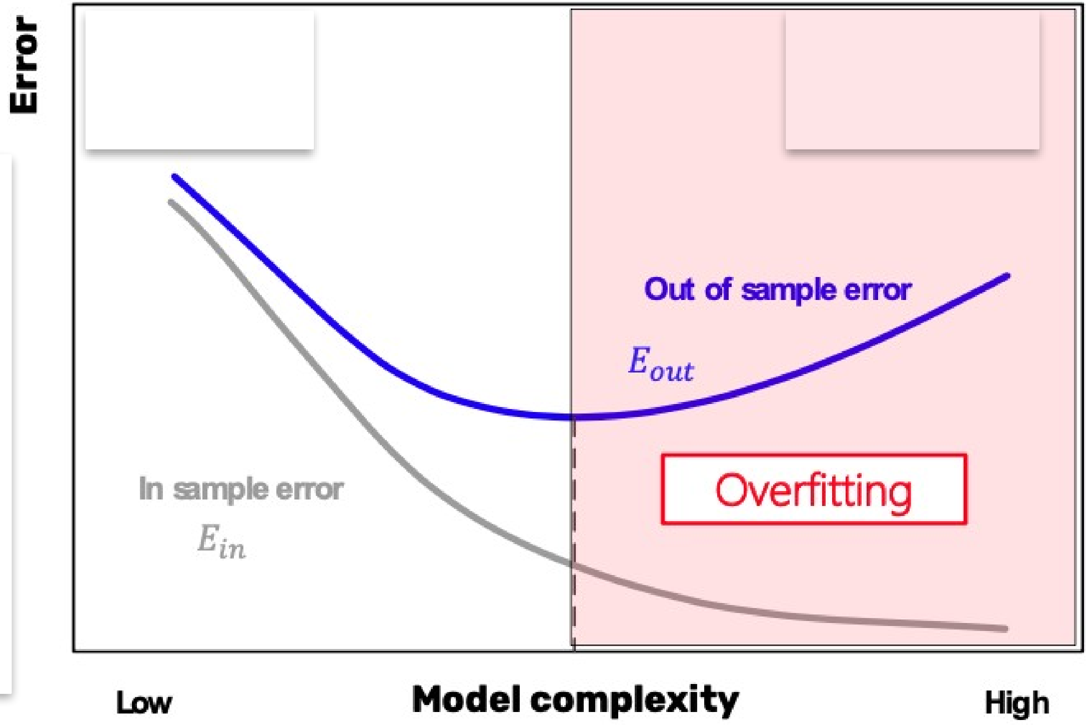
$$\downarrow$$

E_{out} **huge**



Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

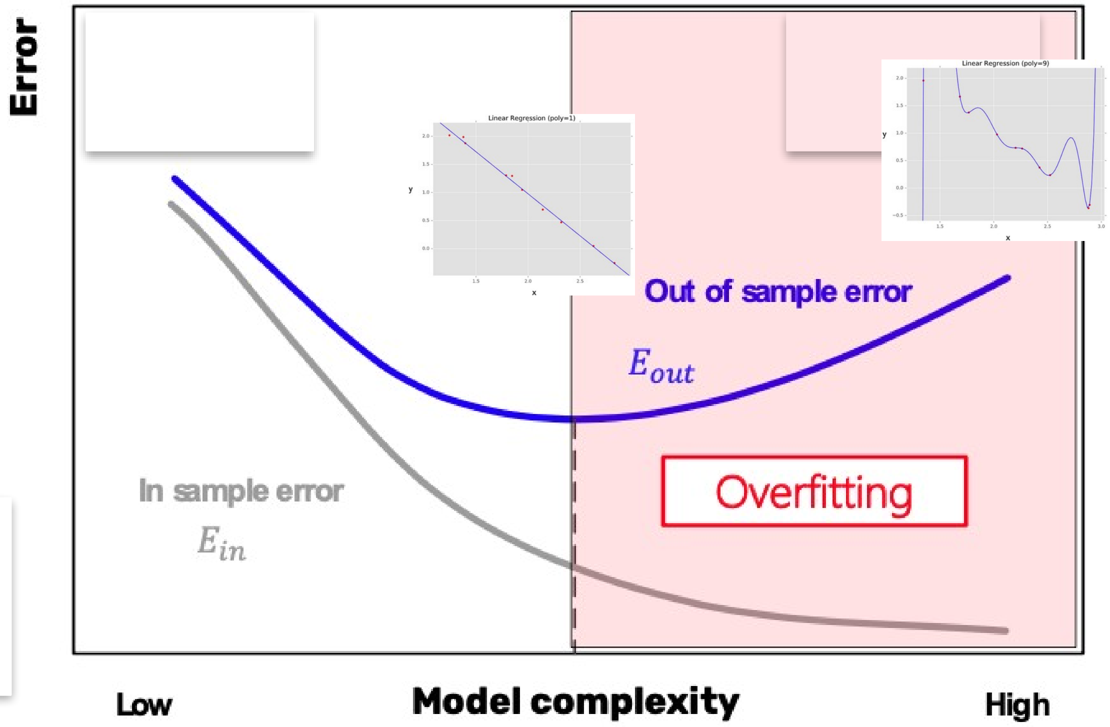


Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

Major source of failure for machine learning systems

Overfitting leads to bad generalization



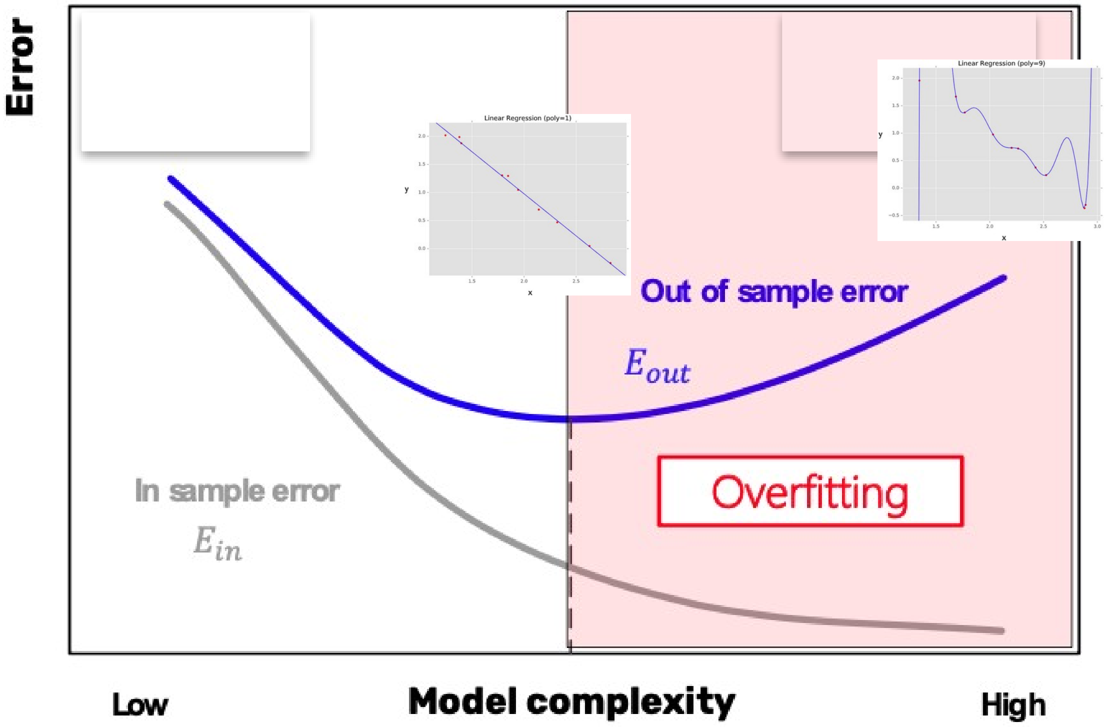
Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

Major source of failure for machine learning systems

Overfitting leads to bad generalization

A model can exhibit bad generalization even if it does not overfit



A cure for overfitting

Regularization is the first line of defense against overfitting

We have seen that **complex models** are more prone to overfitting

- This is because they are more powerful, and thus they can fit the noise

A cure for overfitting

Regularization is the first line of defense against overfitting

We have seen that **complex models** are more prone to overfitting

- This is because they are more powerful, and thus they can fit the noise

Simple models have limited expressivity and thus are less prone to fit the noise.

- However, if we stick only to simple models, we may not end up with a satisfying approximation of the target function $f(\cdot)$

A cure for overfitting

Regularization is the first line of defense against overfitting

We have seen that **complex models** are more prone to overfitting

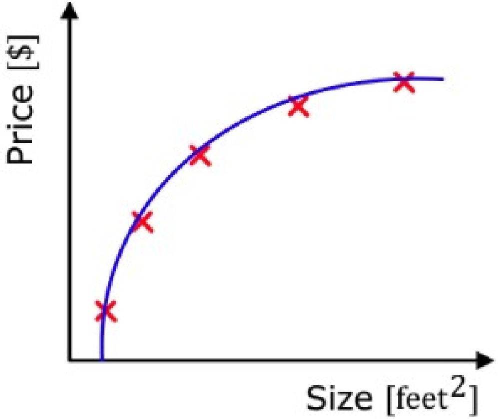
- This is because they are more powerful, and thus they can fit the noise

Simple models have limited expressivity and thus are less prone to fit the noise.

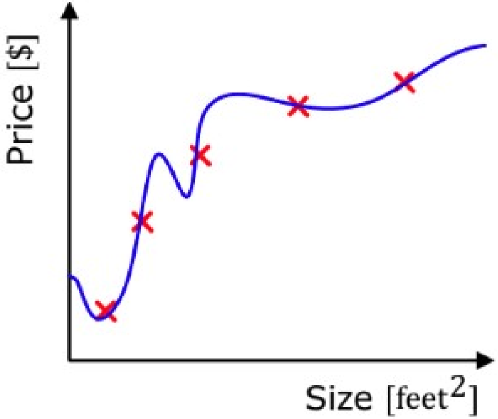
- However, if we stick only to simple models, we may not end up with a satisfying approximation of the target function $f(\cdot)$

*How can we retain the benefits of **both** worlds?*

A cure for overfitting

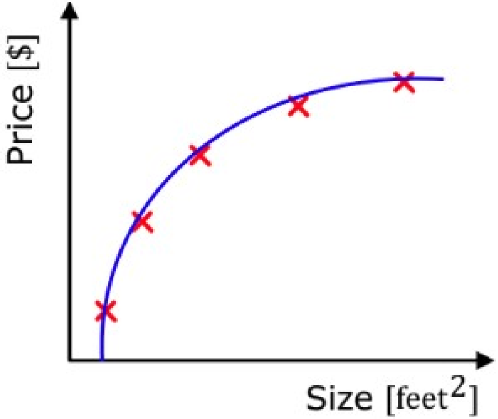


$\mathcal{H}_1\colon y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2$

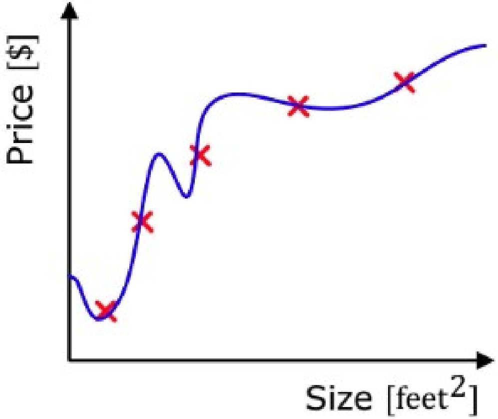


$\mathcal{H}_2\colon y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2 + \theta_3\varphi^3 + \theta_4\varphi^4$

A cure for overfitting



$\mathcal{H}_1$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2$



$\mathcal{H}_2$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2 + \theta_3\varphi^3 + \theta_4\varphi^4$

We can «recover» the model $\mathcal{H}_1$ from the model $\mathcal{H}_2$ by **imposing** $\theta_3 = \theta_4 = 0$

This can be done by minimizing, along with $J(\boldsymbol{\theta})$, also the value of the parameters θ_3, θ_4

Regularization

$h(\cdot)$ is some function that represents our model

More generally, instead of minimizing the in-sample error E_{in} (i.e. the cost function $J(\boldsymbol{\theta})$), minimize the **augmented error:**

For simplicity, suppose $J(\boldsymbol{\theta})$ as squared error function

$$\boxed{E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\big(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\big)^2 + \lambda \cdot \sum_{j=0}^{d-1}(\theta_j)^2}$$

Regularization

$h(\cdot)$ is some function that represents our model

More generally, instead of minimizing the in-sample error E_{in} (i.e. the cost function $J(\boldsymbol{\theta})$), minimize the **augmented error:**

For simplicity, suppose $J(\boldsymbol{\theta})$ as squared error function

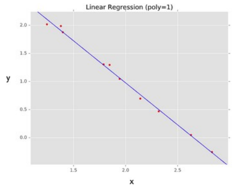
$$\boxed{E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2}$$

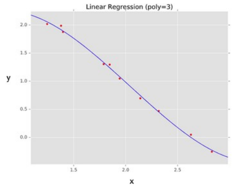
- The term $\Omega = \sum_{j=0}^{d-1}\left(\theta_j\right)^2$ is called **regularizer**

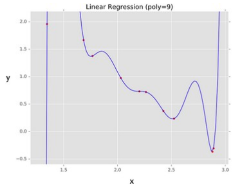
- The term λ (**regularization hyper-parameter**) weights the importance of minimizing $J(\boldsymbol{\theta})$, with respect to minimizing Ω

Polynomial Coefficients

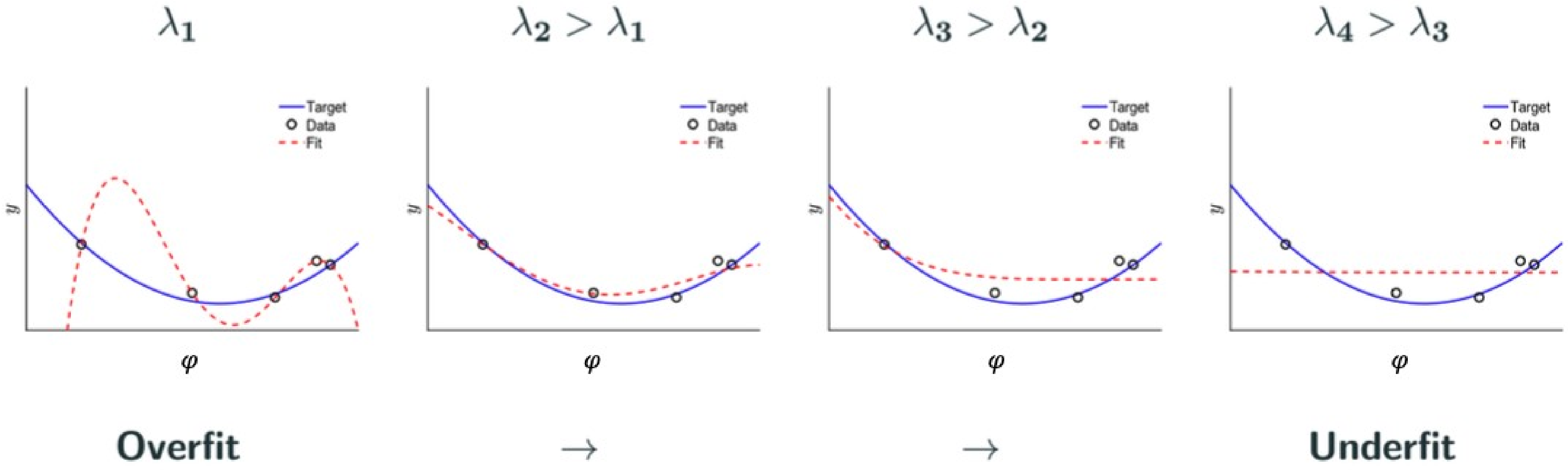
	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43







Effect of λ



$$E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

Choice of the regularizer

There are many choices of possible regularizers. The most used ones are:

- L_2 **regularizer:** also called **Ridge** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \left(\theta_j\right)^2 = \boldsymbol{\theta}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$
- L_1 **regularizer:** also called **Lasso** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \left|\theta_j\right| = \|\boldsymbol{\theta}\|_1$
- **Elastic-net regularizer:** $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \beta \left(\theta_j\right)^2 + (1-\beta) \sum_{j=0}^{d-1} \left|\theta_j\right|$

Choice of the regularizer

There are many choices of possible regularizers. The most used ones are:

- **L_2 regularizer:** also called **Ridge** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \left(\theta_j\right)^2 = \boldsymbol{\theta}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$
- **L_1 regularizer:** also called **Lasso** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \left|\theta_j\right| = \|\boldsymbol{\theta}\|_1$
- **Elastic-net regularizer:** $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} \beta\left(\theta_j\right)^2 + (1-\beta) \sum_{j=0}^{d-1} \left|\theta_j\right|$

The different regularizers behaves differently:

- The ridge penalty tends to shrink all coefficients to a **lower value**
- The lasso penalty tends to set more coefficients **exactly to zero**
- The elastic-net penalty is a compromise between ridge and lasso, with the β value controlling the two contributions