Supervised Machine Learning: Pattern Classification

Bayes Classifier

Bayes Classifier

- Let C_1, C_2, …, C_i, …, C_M be the M classes
  - Each class has N_i number of training examples
- Given: a test example $\mathbf{x}$
- To Compute:
  - Probability of class, $P(C_i \mid \mathbf{x})$
- Bayes decision rule:

Posterior Probability of a class | Likelihood | Prior

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})}$$

Total probability

  - Prior: Prior probability of a class C_i is obtained from the training data of class C_i

$$P(C_i) = \frac{N_i}{N}$$

where, N is total number of training examples
N_i is total number of training examples in class C_i

  - Likelihood of a class (Class conditional density), $p(\mathbf{x} \mid C_i)$:

    Given the training data of a class (C_i), what is the likelihood that $\mathbf{x}$ is coming that class
    - It follows the density (distribution of the data) of a class C_i
    - Computation of class conditional density depends on the parameters of that density (probability distribution)
    - Parameters of the density is estimated from the training data of class C_i

2

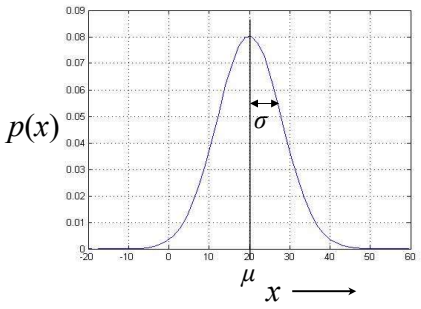
Bayes Classifier

- Training data of a class is can come from any probability distribution (density)
- For the simplicity,
  - Training data of a class is considered to be forming a single cluster, and
  - Assume that training data of a class is coming from a normal (Gaussian) distribution

3

Bayes Classifier with Unimodal Gaussian Density

- Gaussian distribution is a unimodal distribution
  - Single mode or single peak at the center of the data
  - If the dimension of the data is 1, then it is called as univariate unimodal Gaussian distribution



$$p(x) = \mathcal{N}(x \mid \mu, \sigma)$$

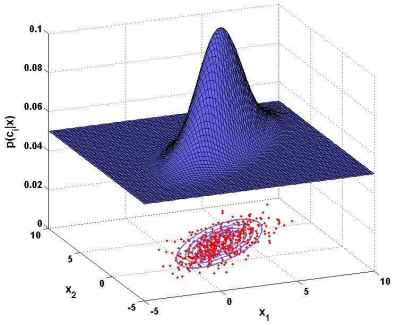
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- μ is the mean
- σ^2 is the variance

4

Bayes Classifier with Unimodal Gaussian Density

- Gaussian distribution is a unimodal distribution
  - Single mode or single peak at the center of the data
  - If the dimension of the data is 1, then it is called as univariate unimodal Gaussian distribution
  - If the dimension of the data is larger the 1, then it is called as multivariate unimodal Gaussian distribution
    - *Bivariate when the dimension is 2*



$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

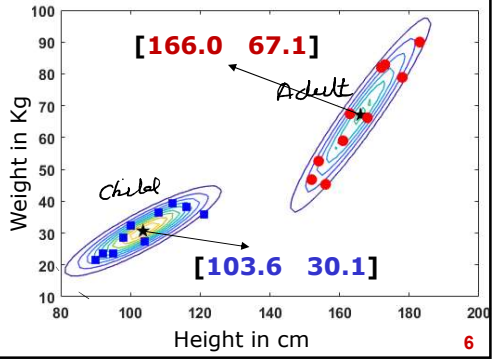
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

- $\boldsymbol{\mu}$ is the mean vector
- $\boldsymbol{\Sigma}$ is the covariance matrix

5

Bayes Classifier with Unimodal Gaussian Density

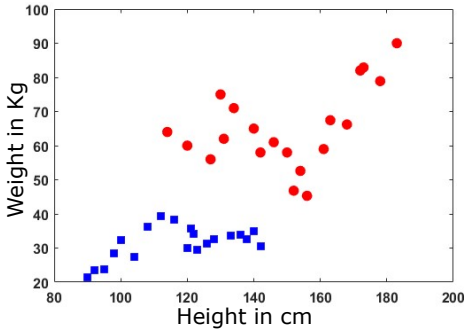
- Limitation:
  - We assume that the data is following the Gaussian distribution
- However, this assumption works for most of the real world data
- If the underlaying distribution (density) of data is known, then Bayes classifier is the minimum error classifier
- The real world data need not be unimodal
  - The shape of the density can be arbitrary
  - Bayes classifier?
- Multimodal density function

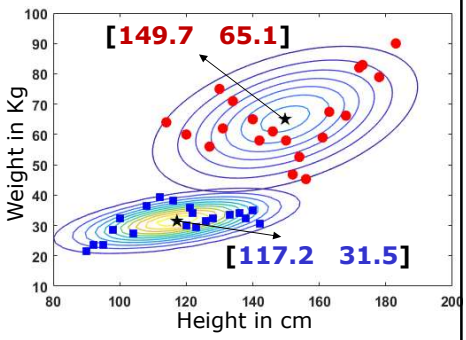


6

Multimodal Distribution: Adult-Child Data

Unimodal Gaussian Density

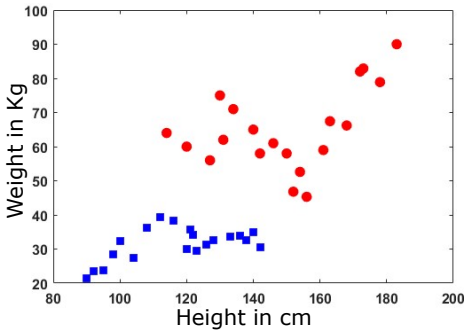


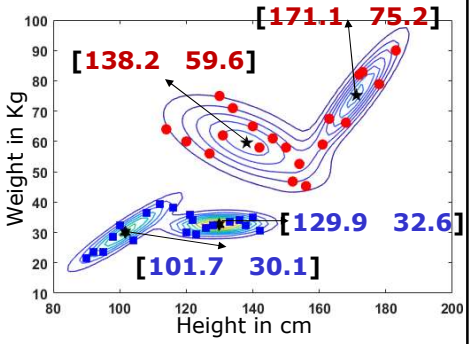


7

Multimodal Distribution: Adult-Child Data

- For a class whose data is considered to have multiple clusters, the probability distribution is multimodal



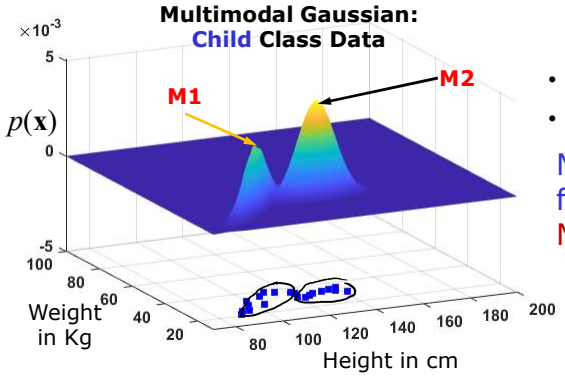


Multimodal Gaussian Density

8

Multimodal Distribution: Adult-Child Data

- For a class whose data is considered to have multiple clusters,
- Each cluster is represented as Gaussian distribution
- Then the probability distribution is multimodal Gaussian



- **M1**: Cluster 1 (mode 1)
- **M2**: Cluster 2 (mode 2)

Multimodal density function – Gaussian Mixture Model (GMM)

9

Bayes Classifier: Multimodal Density

- Let $C_1, C_2, \ldots, C_i, \ldots, C_M$ be the M classes
  - Each class has N_i number of training examples
- Given: a test example $\mathbf{x}$
- Bayes decision rule:

Posterior Probability of a class | Likelihood | Prior

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})}$$

Evidence

  - Likelihood of a class (Class conditional density) follows the distribution of the data of a class – multimodal distribution
- Bayes decision rule can be given as $P(\boldsymbol{\theta}_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \boldsymbol{\theta}_i)P(C_i)}{P(\mathbf{x})}$
  - $\boldsymbol{\theta}_i$ is the parameters of the multimodal distribution of class C_i *estimated from training data* of that class

Class label for $\mathbf{x}$ = $\arg\max_i P(\boldsymbol{\theta}_i \mid \mathbf{x})$ $\quad i = 1, 2, \ldots, M$

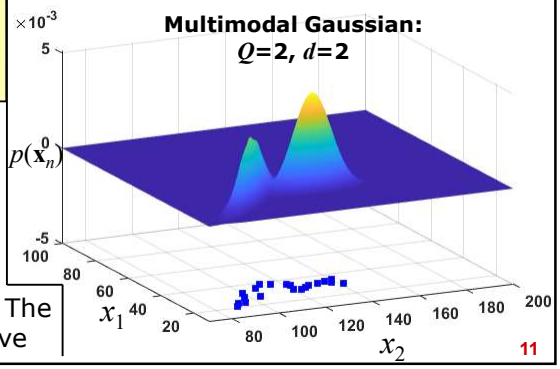
10

Multimodal Gaussian Distribution: Gaussian Mixture Model

- Given: Data having N samples - $\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_n \in \mathbb{R}^d$
- Assume that data is forming a Q number of clusters
  - Hence, data is coming from multimodal density
- Gaussian mixture model (GMM): to represent a multimodal distribution
- GMM is a linear superposition of multiple (Q) *Gaussian components*:

$$p(\mathbf{x}_n) = \sum_{q=1}^{Q} w_q \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$$

  - $\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ is a Gaussian density of q^{th} cluster
  - w_q is the weight (strength) of q^{th} cluster
- Shape of the distribution: The overall envelope of the curve



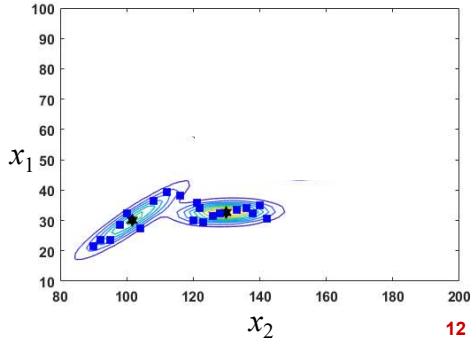
11

Multimodal Gaussian Distribution: Gaussian Mixture Model

- Given: Data having N samples - $\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_n \in \mathbb{R}^d$
- Assume that data is forming a Q number of clusters
  - Hence, data is coming from multimodal density
- Gaussian mixture model (GMM): to represent a multimodal distribution
- GMM is a linear superposition of multiple (Q) *Gaussian components*:

$$p(\mathbf{x}_n) = \sum_{q=1}^{Q} w_q \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$$

  - $\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ is a Gaussian density of q^{th} cluster
  - w_q is the weight (strength) of q^{th} cluster
- Shape of the distribution: The overall envelope of the curve



12

Gaussian Mixture Model (GMM)

- GMM is a linear superposition of multiple Gaussians:

$$p(\mathbf{x}_n) = \sum_{q=1}^{Q} w_q \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$$

- For a d-dimensional feature vector representation of data, the parameters of GMM with Q Gaussian components are
  - d-dimensional mean vector, $\boldsymbol{\mu}_q$, for all $q = 1,2,\ldots, Q$
  - dxd size covariance matrices, $\boldsymbol{\Sigma}_q$, for all $q = 1,2,\ldots, Q$
  - Mixture coefficients, w_q, for all $q = 1,2,\ldots, Q$
    - *Mixture weight or Strength of each clusters (or mixtures or modes)*
    - Property: $\sum_{q=1}^{Q} w_q = 1$
- Objective of training process: To estimate the parameters of the GMM from the training data

13

Training Process: Parameter Estimation of GMM: Incomplete Data Problem

- Given: Training data for a class C_i having N_i samples

$$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_{Ni}\}, \; \mathbf{x}_n \in \mathbb{R}^d$$

- Known: Training data is multimodal in nature
  - Assume: The number of clusters
- Unknown: identity of the cluster (or mixture) of these training data points
- Incomplete data problem:
  - To estimate the parameters of GMM, the information of which data points belong to which cluster is required
  - Given is only data points but not their identity (i.e. to which cluster it belongs)
  - Hidden (latent) information: Identity of data points to the cluster

14

Training Process: Parameter Estimation of GMM: Incomplete Data Problem

- *If identity (latent information) is given, how to estimate parameters of GMM?*
  - *Which sample belonging to which cluster will be known*
  - *We know that each example in a cluster is coming from a Gaussian density*
- Apply maximum likelihood method to estimate the parameters of each of the q mixtures (clusters) ($\boldsymbol{\mu}_q$ and $\boldsymbol{\Sigma}_q$)
- Mixture coefficients, w_q is computed as

$$w_q = \frac{N_{iq}}{N_i}$$

  - N_{iq}: Number of samples in cluster q
  - N_i: Number of samples in class C_i
- In practice, we do not have this information
- Goal of parameter estimation (training process): To find the best possible values of parameters of GMM such that the total likelihood of data is maximized
  - Maximum likelihood method for training a GMM: ***Expectation-Maximization (EM) method***

15

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters
- Expectation-Maximization (EM) method is an iterative process
- Given: Training data for a class C_i having N_i samples

$$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_{Ni}\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

1. Initialize the mean vectors $\boldsymbol{\mu}_q$, covariance matrices $\boldsymbol{\Sigma}_q$ and mixing coefficients w_q, and evaluate the initial value of the total data log likelihood

$$p(\mathcal{D}_i \mid \boldsymbol{\theta}_i) = \prod_{n=1}^{N_i} p(\mathbf{x}_n \mid \boldsymbol{\theta}_i) \quad \text{where } \boldsymbol{\theta}_i = [w_1 \ldots w_q \ldots w_Q, \boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_q \ldots \boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_1 \ldots \boldsymbol{\Sigma}_q \ldots \boldsymbol{\Sigma}_Q]^T$$

$$\mathcal{L}(\boldsymbol{\theta}_i) = \ln p(\mathcal{D}_i \mid \boldsymbol{\theta}_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n \mid \boldsymbol{\theta}_i)$$

16

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters
- Expectation-Maximization (EM) method is an iterative process
- Given: Training data for a class C_i having N_i samples

$$\mathcal{D}_i=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_{Ni}\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

1. Initialize the mean vectors $\boldsymbol{\mu}_q$, covariance matrices $\boldsymbol{\Sigma}_q$ and mixing coefficients w_q, and evaluate the initial value of the total data log likelihood

$$p(\mathcal{D}_i \mid \boldsymbol{\theta}_i) = \prod_{n=1}^{N_i} p(\mathbf{x}_n \mid \boldsymbol{\theta}_i) \quad \text{where } \boldsymbol{\theta}_i = [w_1 \ldots w_q \ldots w_Q, \boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_q \ldots \boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_1 \ldots \boldsymbol{\Sigma}_q \ldots \boldsymbol{\Sigma}_Q]^T$$

$$\mathcal{L}(\boldsymbol{\theta}_i) = \ln p(\mathcal{D}_i \mid \boldsymbol{\theta}_i) = \sum_{n=1}^{N_i} \ln \left(\sum_{q=1}^{Q} w_q \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \right)$$

expectation step

2. **E-step**: Evaluate the responsibilities $\gamma_k(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster

17

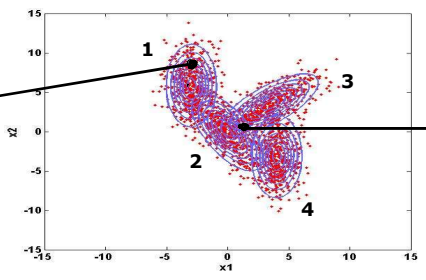
Training Process: EM Method – Responsibility Term

- A quantity that plays an important role is the responsibility term, $\gamma_q(\mathbf{x})$
- It is given by

$$\gamma_q(\mathbf{x}) = \frac{w_q \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)}{\sum_{q=1}^{Q} w_q \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)}$$

- w_q : mixture coefficient or prior probability of cluster q,
- $\gamma_q(\mathbf{x})$ gives the posterior probability of the cluster q for the observation $\mathbf{x}$

$\gamma_1(\mathbf{x}_n) = 0.99$
$\gamma_2(\mathbf{x}_n) = 0.01$
$\gamma_3(\mathbf{x}_n) = 0.00$
$\gamma_4(\mathbf{x}_n) = 0.00$



$\gamma_1(\mathbf{x}_m) = 0.08$
$\gamma_2(\mathbf{x}_m) = 0.42$
$\gamma_3(\mathbf{x}_m) = 0.34$
$\gamma_4(\mathbf{x}_m) = 0.16$
1.00

18

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters

1. Initialize the mean vectors $\boldsymbol{\mu}_q$, covariance matrices $\boldsymbol{\Sigma}_q$ and mixing coefficients w_q, and evaluate the initial value of the total data log likelihood

2. **E-step**: Evaluate the responsibilities $\gamma_q(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster

3. **M-step**: Re-estimate the parameters $\boldsymbol{\mu}_q^{new}$, $\boldsymbol{\Sigma}_q^{new}$ and w_q^{new} using the current responsibilities

$$\boldsymbol{\mu}_q^{new} = \frac{1}{N_q}\sum_{n=1}^{N_i}\gamma_q(\mathbf{x}_n)\,\mathbf{x}_n \qquad \boldsymbol{\Sigma}_q^{new} = \frac{1}{N_q}\sum_{n=1}^{N_i}\gamma_q(\mathbf{x}_n)\,(\mathbf{x}_n-\boldsymbol{\mu}_q)(\mathbf{x}_n-\boldsymbol{\mu}_q)^{\mathsf{T}}$$

$$w_q^{new} = \frac{N_q}{N} \qquad N_q = \sum_{n=1}^{N_i}\gamma_q(\mathbf{x}_n)$$

- N_q: Effective number of points assigned to the cluster q

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters

1. Initialize the mean vectors $\boldsymbol{\mu}_q$, covariance matrices $\boldsymbol{\Sigma}_q$ and mixing coefficients w_q, and evaluate the initial value of the total data log likelihood

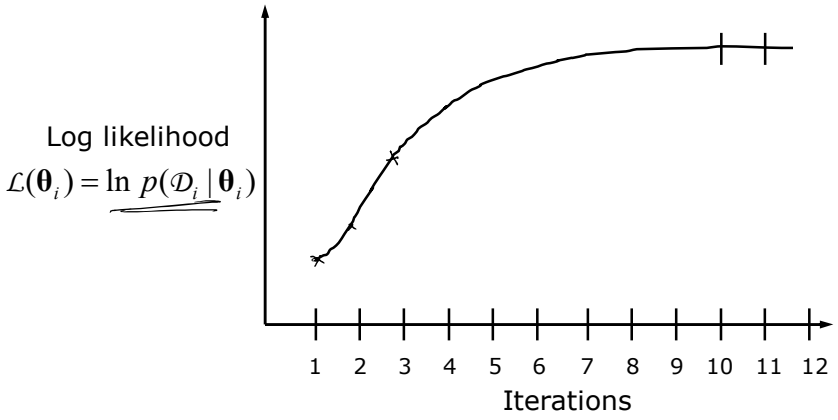
2. **E-step**: Evaluate the responsibilities $\gamma_q(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster

3. **M-step**: Re-estimate the parameters $\boldsymbol{\mu}_q^{new}$, $\boldsymbol{\Sigma}_q^{new}$ and w_q^{new} using the current responsibilities

4. Evaluate the total data log likelihood using the re-estimated parameters and check for convergence of the total data log likelihood
   - If the convergence criterion is not satisfied return to step 2

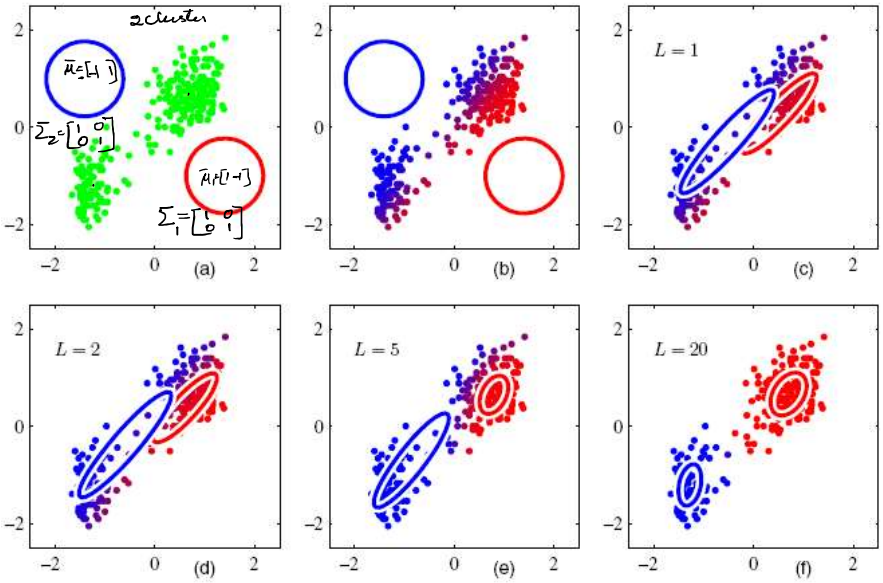
Training Process:
Expectation-Maximization (EM) for GMMs

- Convergence criterion: Difference between total data log likelihoods of successive iterations fall below a threshold (E.g. 10^{-3})



21

Training Process:
Illustration of Parameter Estimation



C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

22

Testing Phase: Bayes Classifier: Multimodal Data

- Let $C_1, C_2, \ldots, C_i, \ldots, C_M$ be the M classes
- Given: a test example $\mathbf{x}$
- Bayes decision rule:

Posterior Probability of a class | Likelihood | Prior | Evidence

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i) P(C_i)}{P(\mathbf{x})}$$

$$p(\mathbf{x} \mid C_i) = \sum_{q=1}^{Q} w_{iq}\, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{iq}, \boldsymbol{\Sigma}_{iq}) \quad \longrightarrow \text{GMM}$$

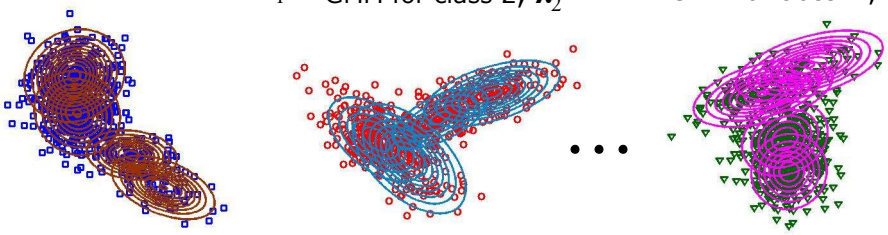
$$\text{Class label for } \mathbf{x} = \arg\max_i P(C_i \mid \mathbf{x}) \qquad i = 1, 2, \ldots, M$$

23

Bayes Classifier with Multimodal Gaussian Density (GMM) – Training Process

- Let $C_1, C_2, \ldots, C_i, \ldots, C_M$ be the M classes
- Let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i, \ldots, \mathcal{D}_M$ be the training data for M classes
- Build GMM ($\boldsymbol{\lambda}$) for each of the classes

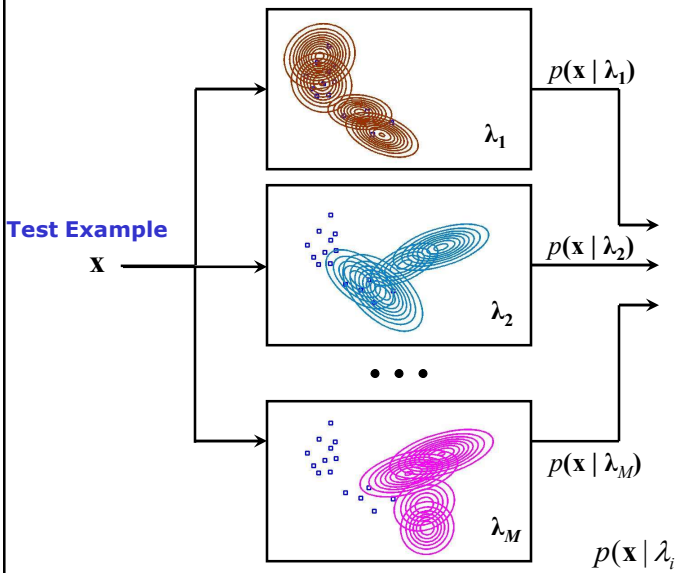
GMM for class 1, $\boldsymbol{\lambda}_1$ | GMM for class 2, $\boldsymbol{\lambda}_2$ | GMM for class M, $\boldsymbol{\lambda}_M$



$$\text{GMM for Class } i, \boldsymbol{\lambda}_i = [w_{iq}, \boldsymbol{\mu}_{iq}, \boldsymbol{\Sigma}_{iq}]_{q=1}^{Q}$$

24

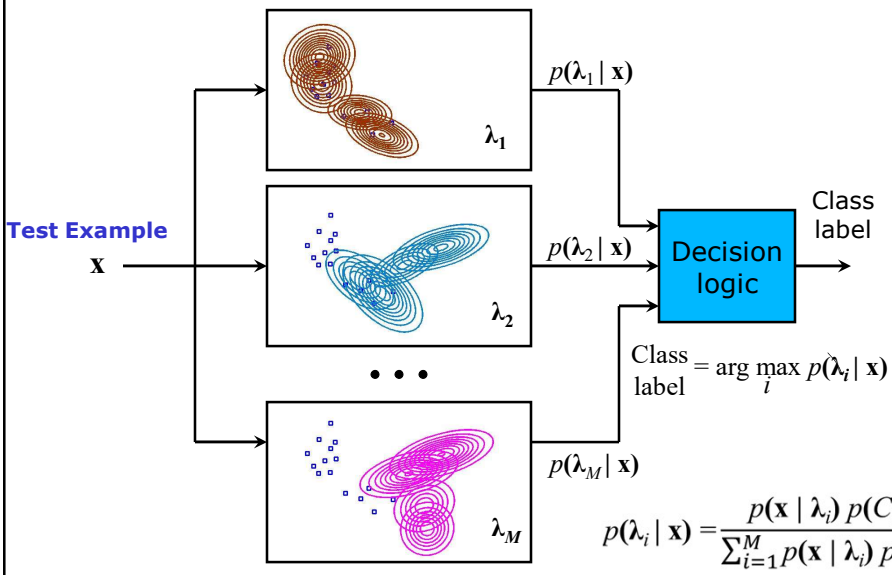
Bayes Classifier with Multimodal Gaussian Density (GMM) – Classification (Test)



$$p(\mathbf{x} \mid \lambda_i) = \sum_{q=1}^{Q_i} w_{iq} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{iq}, \boldsymbol{\Sigma}_{iq})$$

25

Bayes Classifier with Multimodal Gaussian Density (GMM) – Classification (Test)



$$p(\lambda_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \lambda_i)\, p(C_i)}{\sum_{i=1}^{M} p(\mathbf{x} \mid \lambda_i)\, p(C_i)}$$

26

Determining Q, Number of Gaussian Components

- This is determined experimentally
- Starting with Q=1, test set is used to estimate the accuracy of the Bayes classifier
  - **Note**: Q=1 means, GMM with one Gaussian component
  - GMM with one Gaussian component is equivalent to unimodal Gaussian
- This process is repeated each time by incrementing Q to allow for more Gaussian components
- The Bayes classifier using GMM with Q components that gives the maximum accuracy may be selected

27

Bayes Classifier with GMM – Summary

- Multimodal probability distribution for each class is represented by a Gaussian mixture model
- GMM is a powerful way of modeling data
- Using GMM, a data of any arbitrary shaped distribution can be modeled
- In GMM, number of parameters to be estimated for each class is dependent on:
  - Dimensionality of the data space d
  - Number of Gaussian mixtures Q
  - Number of parameters to be estimated:

$$Q \times d + Q \times (d(d+1))/2 + Q \rightarrow$$

- For large values of d and Q, the number of examples required to estimate the parameters properly will be large

28

Bayes Classifier with GMM – Summary (Contd.)

- Bayes classifier using GMM performs better than Bayes classifier using unimodal Gaussian
  - GMM approximates the true class-conditional density
- *When the estimated class-conditional densities are the same as the true densities, Bayes classifier gives minimum classification error*

29

Bayes Classifier - Summery

- Let C_1, C_2, …, C_i, …, C_M be the M classes
  - Each class has N_i number of training examples
- Given: a test example $\mathbf{x}$
- Bayes decision rule:

Posterior Probability of a class

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})}$$

Likelihood, Prior, Evidence

  - Likelihood of a class (Class conditional density) follows the distribution of the data of a class –
    - Unimodal Gaussian distribution – Training data of a class as single cluster
    - Multimodal Gaussian distribution (Gaussian mixture model - GMM) - Training data of a class as multiple cluster
- Bayes decision rule can be given as $P(\boldsymbol{\theta}_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \boldsymbol{\theta}_i)P(C_i)}{P(\mathbf{x})}$
  - $\boldsymbol{\theta}_i$ is the parameters of the distribution of class C_i *estimated from training data* of that class

Class label for $\mathbf{x}$ = $\arg\max_i P(\boldsymbol{\theta}_i \mid \mathbf{x})$ $\quad i = 1, 2, ..., M$

30

Supervised Machine Learning: Pattern Classification

Naïve Bayes Classifier

Naïve Bayes Classifier

- Special case of Bayes classifier using unimodal density function
- Naïve Bayes assumes that features (attributes) are independent or uncorrelated
- When the class conditional density is considered as unimodal Gaussian density, naïve Bayes classifier is a Bayes classifier using unimodal Gaussian density with diagonal covariance matrix

32

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

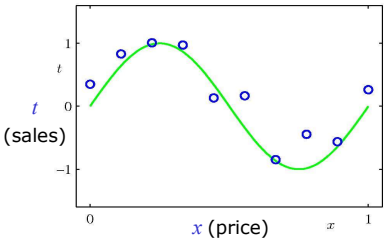
3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

33

Supervised Machine Learning: Regression

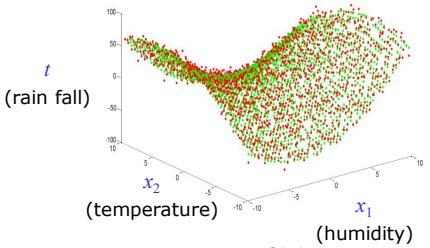
Numeric Prediction (Regression)

- Numeric prediction: Task of predicting continuous (or ordered) values for given input
- Example:
  - Predicting potential sales of a new product given its price



$t = f(x)$

  - Predicting amount of rain fall given the temperature and humidity in the atmosphere



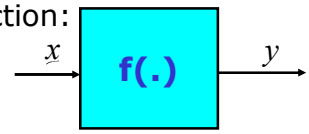
$t = f(\mathbf{x})$

$\mathbf{x} = [x_1, x_2]^T$

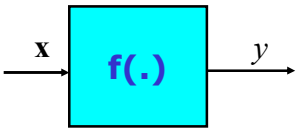
2

Numeric Prediction (Regression)

- Regression analysis is used to model the relationship between one or more independent (input) variable and a dependent (output) variable
  - Dependent variable is always continuous valued or ordered valued
  - Example: Dependent variable: Rain fall
    Independent variable(s): temperature, humidity
- The values of independent variables are known
- The dependent variable is what we want to predict
- Regression analysis can be viewed as mapping function:



- Single independent variable (x)
- Single dependent variable (y)



- Multiple independent variable ($\mathbf{x} \in \mathbb{R}^d$)
- Single dependent variable (y)

3

Numeric Prediction (Regression)

- Regression is a two step process
  - Step1: Building a regression model
    - Learning from data (training phase)
    - Supervised learning: In supervised learning, each example is a *pair* consisting of an input example (independent variables) and a desired output value (dependent variable)
    - Regression model is build by analysing or learning from a training data set made up of one or more independent variables and their dependent labels

$$y_n = f(\mathbf{x}_n)$$

      - $\mathbf{x}_n$ is the n^{th} input example and y_n is the corresponding output variable
  - Step2: Using regression model for prediction
    - Testing phase
    - Predicting dependent variable
- Accuracy of a predictor:
  - How well a given predictor can predict for new values
- Target of learning techniques: Good generalization ability

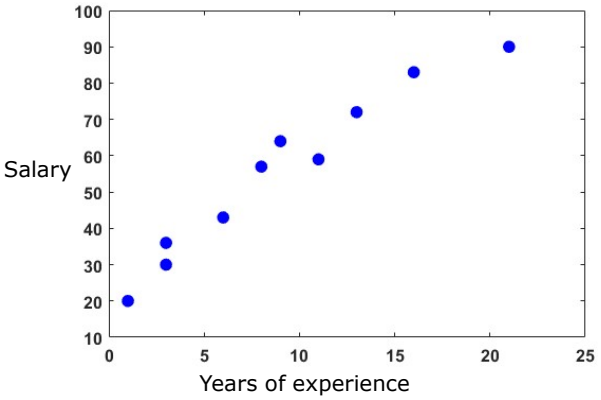
4

Illustration of Training Set: Salary Prediction

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Independent variable: Years of experience

Dependent variable: Salary

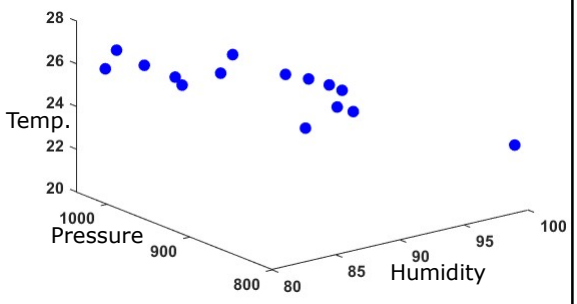


5

Illustration of Training Set: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Independent variable: Humidity, Pressure
- Dependent variable: Temperature (Temp)



6

Illustration of Training Set: Wine Quality Prediction [1]

Fixed Acidity (x_1)	Volatile Acidity (x_2)	Citric acid (x_3)	Residual Sugar (x_4)	Chlorides (x_5)	Free SO_2 (x_6)	Total SO_2 (x_7)	Density (x_8)	pH (x_8)	Sulphates (x_9)	Alcohol (x_{10})	Quality (y)
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.42
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5.57
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5.17
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6.65
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.68
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5.63
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5.32
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7.16
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7.2
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5.18

- Number of independent variable: 10
- Dependent variable: Quality

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. "*Modeling wine preferences by data mining from physicochemical properties*," In Decision Support Systems, Elsevier, vol. 47, issue 4, pp. 547-553, 2009.

7

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

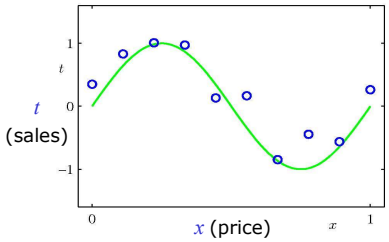
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

8

Supervised Machine Learning: Regression

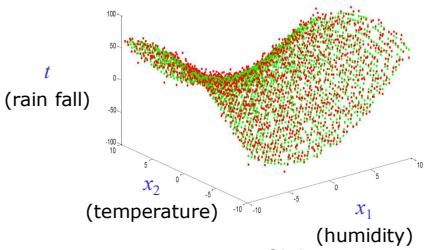
Numeric Prediction (Regression)

- Numeric prediction: Task of predicting continuous (or ordered) values for given input
- Example:
  - Predicting potential sales of a new product given its price



$t = f(x)$

  - Predicting amount of rain fall given the temperature and humidity in the atmosphere



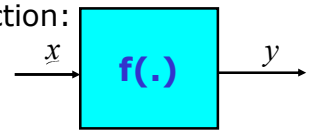
$t = f(\mathbf{x})$

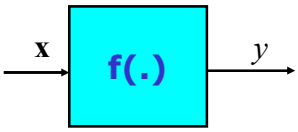
$\mathbf{x} = [x_1, x_2]^T$

2

Numeric Prediction (Regression)

- Regression analysis is used to model the relationship between one or more independent (input) variable and a dependent (output) variable
  - Dependent variable is always continuous valued or ordered valued
  - Example: Dependent variable: Rain fall
    Independent variable(s): temperature, humidity
- The values of independent variables are known
- The dependent variable is what we want to predict
- Regression analysis can be viewed as mapping function:





- Single independent variable (x)
- Single dependent variable (y)
- Multiple independent variable ($\mathbf{x} \in \mathbb{R}^d$)
- Single dependent variable (y)

3

Numeric Prediction (Regression)

- Regression is a two step process
  - Step1: Building a regression model
    - Learning from data (training phase)
    - Supervised learning: In supervised learning, each example is a *pair* consisting of an input example (independent variables) and a desired output value (dependent variable)
    - Regression model is build by analysing or learning from a training data set made up of one or more independent variables and their dependent labels

$$y_n = f(\mathbf{x}_n)$$

      - $\mathbf{x}_n$ is the n^{th} input example and y_n is the corresponding output variable
  - Step2: Using regression model for prediction
    - Testing phase
    - Predicting dependent variable
- Accuracy of a predictor:
  - How well a given predictor can predict for new values
- Target of learning techniques: Good generalization ability

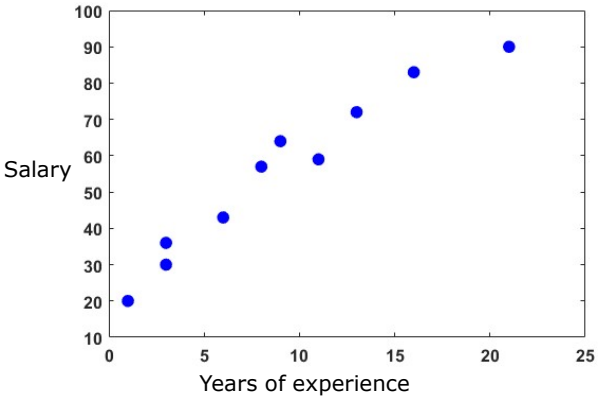
4

Illustration of Training Set: Salary Prediction

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Independent variable: Years of experience

Dependent variable: Salary

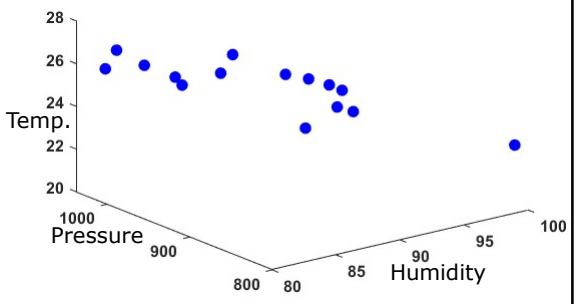


5

Illustration of Training Set: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Independent variable: Humidity, Pressure
- Dependent variable: Temperature (Temp)



6

Illustration of Training Set: Wine Quality Prediction [1]

Fixed Acidity (x_1)	Volatile Acidity (x_2)	Citric acid (x_3)	Residual Sugar (x_4)	Chlorides (x_5)	Free SO_2 (x_6)	Total SO_2 (x_7)	Density (x_8)	pH (x_8)	Sulphates (x_9)	Alcohol (x_{10})	Quality (y)
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.42
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5.57
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5.17
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6.65
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.68
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5.63
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5.32
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7.16
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7.2
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5.18

- Number of independent variable: 10
- Dependent variable: Quality

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. "*Modeling wine preferences by data mining from physicochemical properties*," In Decision Support Systems, Elsevier, vol. 47, issue 4, pp. 547-553, 2009.

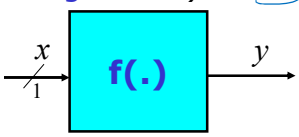
7

Supervised Machine Learning: Regression

Linear Regression

Linear Regression

- Linear approach to model the relationship between a scalar response, (y) (or dependent variable) and one or more predictor variables, (x *or* $\mathbf{x}$) (or independent variables)
- The output is going to be the linear function of input (one or more independent variables)
- Simple linear regression (straight-line regression):
  - Single independent variable (x)
  - Single dependent variable (y)
  - *Fitting a straight-line*



- Multiple linear regression:
  - two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)
  - *Fitting a hyperplane (linear surface)*

9

Straight-Line (Simple Linear) Regression

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
  - x_n: n^{th} input example (independent variable)
  - y_n: Dependent variable (output) corresponding to n^{th} independent variable
- Example: Predicting the salary given the year of experience

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

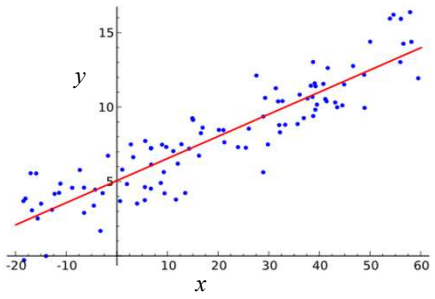
- Independent variable:
  - Years of experience
- Dependent variable:
  - Salary

$y = f(x)$

10

Straight-Line (Simple Linear) Regression

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
  - x_n: n^{th} input example (independent variable)
  - y_n: Dependent variable (output) corresponding to n^{th} independent variable
- Function governing the relationship between input and output: $y_n = f(x_n, w, w_0) = w\,x_n + w_0$
  - The coefficients w_0 and w are parameters of straight-line (regression coefficients) ***- Unknown***



- Function $f(x_n,w,w_0)$ is a linear function of x_n and it is a linear function of coefficients w and w_0
  - ***Linear model for regression***
- The values for the coefficients will be determined by fitting the linear function (straight-line) to the training data

11

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n,w,w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w\,x_n + w_0$$

$$(\hat{y}_n - y_n)^2 \qquad \forall n = 1,2,\ldots, N$$

12

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n,w,w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w\, x_n + w_0$$

$$E(w, w_0) = \frac{1}{2}\sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

13

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n,w,w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w\, x_n + w_0$$

$$E(w, w_0) = \frac{1}{2}\sum_{n=1}^{N} (f(x_n, w, w_0) - y_n)^2$$

14

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n,w,w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w\, x_n + w_0$$

$$\underset{w,w_0}{\text{minimize}} \;\; E(w, w_0) = \frac{1}{2}\sum_{n=1}^{N} \left(f(x_n, w, w_0) - y_n\right)^2$$

- Minimize the error such that the coefficients w_0 and w represent the parameter of line that best fit the training data

15

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n,w,w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w\, x_n + w_0$$

$$\underset{w,w_0}{\text{minimize}} \;\; E(w, w_0) = \frac{1}{2}\sum_{n=1}^{N} \left(\hat{y}_n - y_n\right)^2$$

- The derivatives of error function with respect to the coefficients will be linear in the elements of w and w_0
- Hence the minimization of the error function has unique solution and found in closed form

16

Straight-Line (Simple Linear) Regression: Training Phase

- Cost function for optimization:

$$E(w, w_0) = \frac{1}{2}\sum_{n=1}^{N}\left(f(x_n, w, w_0) - y_n\right)^2$$

- Conditions for optimality: $\frac{\partial E(w, w_0)}{\partial w} = 0$ $\frac{\partial E(w, w_0)}{\partial w_0} = 0$

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(w\,x_n + w_0 - y_n\right)^2}{\partial w} = 0 \qquad \frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(w\,x_n + w_0 - y_n\right)^2}{\partial w_0} = 0$$

- Solving this give optimal $\hat{w}$ and $\hat{w}_0$ as

Homework: Derive the expression for $\hat{w}$ and $\hat{w}_0$

$$\hat{w} = \frac{\sum_{n=1}^{N}(x_n - \mu_x)(y_n - \mu_y)}{\sum_{n=1}^{N}(x_n - \mu_x)^2}$$

$\text{cov}(x, y)$

$\text{var}(x)$

$$\hat{w}_0 = \mu_y - \hat{w}\mu_x$$

- μ_x: sample mean of independent variable x
- μ_y: sample mean of dependent variable y

17

Straight-Line (Simple Linear) Regression: Testing Phase

- For any test example x, the predicted value is given by:

$$\hat{y} = f(x, \hat{w}, \hat{w}_0) = \hat{w}\,x + \hat{w}_0$$

  - For any $\hat{w}$ and $\hat{w}_0$ are the optimal parameters of the line learnt during training

18

Evaluation Metrics for Regression: Squared Error and Mean Squared Error

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
  - y : actual value
  - $\hat{y}$: predicted value
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error (RMSE):

$$E_{\text{RMSE}} = \sqrt{\frac{1}{N_t}\sum_{n=1}^{N_t}(\hat{y}_n - y_n)^2}$$

- RMSE expressed in % as:

$$E_{\text{RMSE}}(\%) = \frac{\sqrt{\frac{1}{N_t}\sum_{n=1}^{N_t}(\hat{y}_n - y_n)^2}}{\frac{1}{N_t}\sum_{n=1}^{N_t} y_n} * 100$$

19

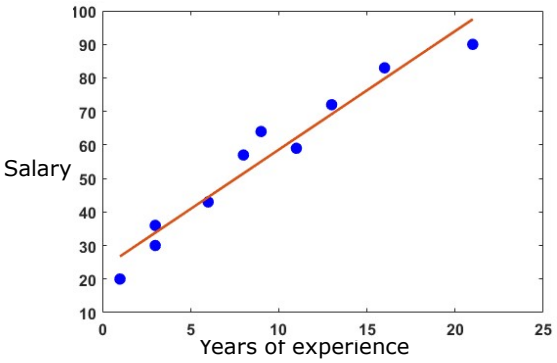
Illustration of Simple Linear Regression: Salary Prediction - Training

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

$$\hat{w} = \frac{\sum_{n=1}^{N}(x_n - \mu_x)(y_n - \mu_y)}{\sum_{n=1}^{N}(x_n - \mu_x)^2} \qquad \hat{w}_0 = \mu_y - \hat{w}\mu_x$$

N = 10

- μ_x: 9.1
- μ_y: 55.4
- $\hat{w}$: 3.54
- $\hat{w}_0$: 23.21



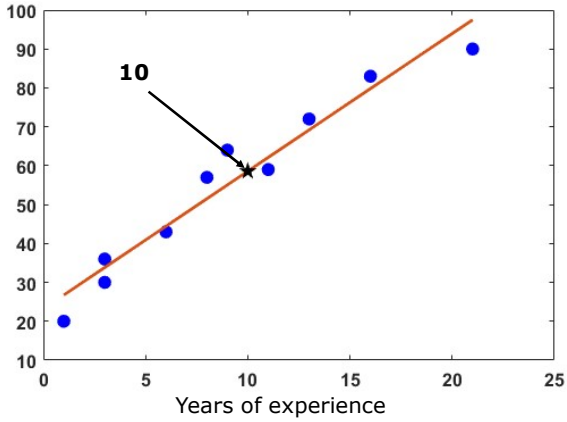
20

Illustration of Simple Linear Regression: Salary Prediction - Test

- $\hat{w}$: 3.54
- $\hat{w}_0$: 23.21

Years of experience (x)	Salary (in Rs 1000) (y)
10	-

$\hat{y} = \hat{w}_0 + \hat{w} x$ at $x = 10$

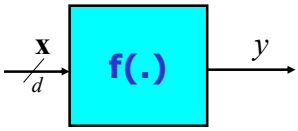


- Predicted salary: 58.584
- Actual salary: 58.000
- Squared error: **0.34**

21

Multiple Linear Regression

- Multiple linear regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
  - d: dimension of input example (number of independent variables)
  - $\mathbf{x}_n$: n^{th} input example (d independent variables)
  - y_n: Dependent variable (output) corresponding to n^{th} input example
- Function governing the relationship between input and output: $y_n = f(\mathbf{x}_n, \mathbf{w}) = w_d x_{nd} + ... + w_2 x_{n2} + w_1 x_{n1} + w_0 = \sum_{i=0}^{d} w_i x_{ni} = \mathbf{w}^{\mathsf{T}} \mathbf{x}_n$
  - The coefficients $w_0, w_1, \ldots, w_d$ are collectively denoted by the vector $\mathbf{w}$ ***- Unknown***
- Function $f(\mathbf{x}_n, \mathbf{w})$ is a linear function of $\mathbf{x}_n$ and it is a linear function of coefficients $\mathbf{w}$
  - ***Linear model for regression***

22

Linear Regression: Linear Function Approximation

- Linear function:
  - 2 input variable case (3-dimensional space): The mapping function is a **plane** specified by

$$y = f(\mathbf{x}, \mathbf{w}) = w_2 x_2 + w_1 x_1 + w_0 = 0$$

where $\mathbf{w} = [w_0, w_1, w_2]^\mathsf{T}$ and $\mathbf{x} = [1, x_1, x_2]^\mathsf{T}$

  - d input variable case (d+1–dimensional space): The mapping function is a **hyperplane** specified by

$$y = f(\mathbf{x}, \mathbf{w}) = w_d x_d + ... + w_2 x_2 + w_1 x_1 + w_0 = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^\mathsf{T}\mathbf{x} = 0$$

where $\mathbf{w} = [w_0, w_1, ..., w_d]^\mathsf{T}$ and $\mathbf{x} = [1, x_1, ..., x_d]^\mathsf{T}$

23

Multiple Linear Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n, \mathbf{w})$, in the training set for any given value of $\mathbf{w}$

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x}_n + w_0 = \sum_{i=0}^{d} w_i x_i$$

- The error function is a

$$\underset{\mathbf{w}}{\text{minimize}} \; E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

  - quadratic function of the coefficients $\mathbf{w}$ and
  - The derivatives of error function with respect to the coefficients will be linear in the elements of $\mathbf{w}$
- Hence the minimization of the error function has unique solution and found in closed form

24

Multiple Linear Regression: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(f(\mathbf{x}_n, \mathbf{w}) - y_n\right)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

$$\begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\sum_{i=0}^{d} w_i x_{ni} - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \mathbf{x}_n - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

25

Multiple Linear Regression: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(f(\mathbf{x}_n, \mathbf{w}) - y_n\right)^2$$

$d \to 100$
$N = 50$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \mathbf{x}_n - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1}\mathbf{X}^\top \mathbf{y}$$

$(d+1)\times(d+1)$ $(d+1)\times 1$ $(d+1)\times 1$ $(d+1)\times N$ $N\times(d+1)$ $N\times 1$

– Assumption: $d < N$

curse of dimensions

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nd} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \\ \dots \\ y_N \end{bmatrix}$$

$N \times (d+1)$

$\mathbf{X}$ is data matrix

26

Multiple Linear Regression: Testing Phase

- Optimal coefficient vector $\mathbf{w}$ is given by

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{\mathbf{w}} = \mathbf{X}^{+} \mathbf{y}$$

where $\mathbf{X}^{+} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top$ is the pseudo inverse of matrix $\mathbf{X}$

- For any test example $\mathbf{x}$, the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\top \mathbf{x} = \sum_{i=0}^{d} \hat{w}_i x_i$$

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}$$

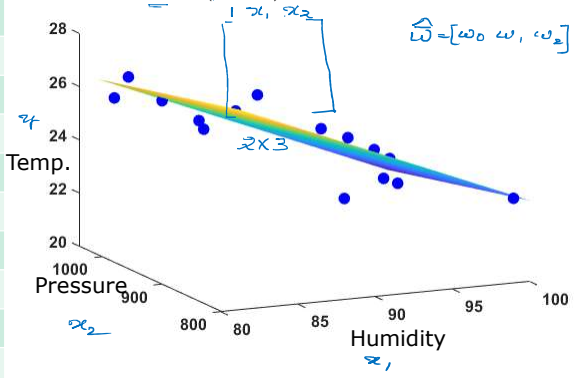
27

Illustration of Multiple Linear Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Training:

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$



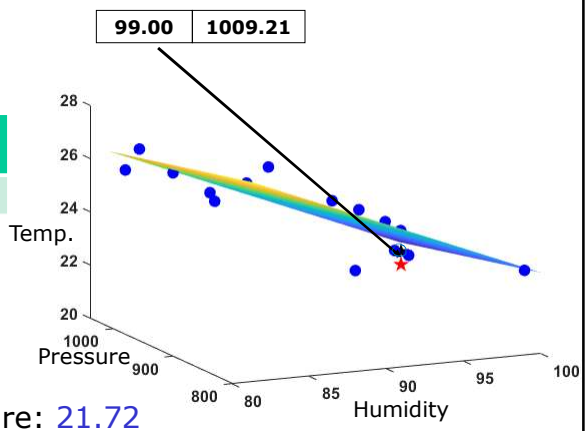
28

Illustration of Multiple Linear Regression: Temperature Prediction - Test

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$$

Humidity (x_1)	Pressure (x_2)	Temp (y)
99.00	1009.21	-

$$y = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\mathsf{T}}\mathbf{x}$$



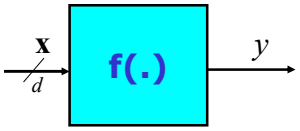
- Predicted temperature: 21.72
- Actual temperature: 21.24
- Squared error: **0.2347**

29

Application of Regression: A Method to Handle Missing Values

- Use most probable value to fill the missing value:
  - Use regression techniques to predict the missing value (regression imputation)
    - Let x_1, x_2, …, x_d be a set of d attributes
    - Regression (multivariate): The n^{th} value is predicted as

$$y_n = f(x_{n1}, x_{n2}, \ldots, x_{nd})$$



    - Simple or Multiple Linear regression:

$$y_n = w_1 x_{n1} + w_2 x_{n2} + \ldots + w_d x_{nd}$$

    - Popular strategy
    - It uses the most information from the present data to predict the missing values
    - It preserves the relationship with other variables

Application of Regression: A Method to Handle Missing Values

- Training process:
  - Let y be the attribute, whose missing values to be predicted
  - Training examples: All $\mathbf{x}=[x_1, x_2, \ldots, x_d]^T$, a set of d dependent attributes for which the independent variable y is available
  - The values for the coefficients will be determined by fitting the linear function to the training data

	Dates	Temperature	Humidity	Rain
2	08-07-2018	25.46875	82.1875	6.75
3	09-07-2018	26.19298	83.1491	1761.75
4	10-07-2018	25.17021	85.3404	652.5
5	11-07-2018	NaN	87.6866	963
6	12-07-2018	24.06923	87.6462	254.25
7	13-07-2018	21.20779	95.9481	339.75
8	15-07-2018	23.48571	96.1714	38.25
9	18-07-2018	NaN	98.5897	29.25
10	19-07-2018	25.09346	88.3271	4.5
11	20-07-2018	25.39423	90.4327	112.5
12	21-07-2018	NaN	94.5378	735.75
13	22-07-2018	22.5098	99	607.5
14	23-07-2018	22.904	98	717.75
15	24-07-2018	NaN	99	513
16	25-07-2018	23.18182	98.9697	195.75

- Dependent variable: Temperature
- Independent variables: Humidity and Rainfall

Application of Regression: A Method to Handle Missing Values

- Testing process (Prediction):
  - Optimal coefficient vector $\mathbf{w}$ is given by

$$\hat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

  - For any test example $\mathbf{x}$, the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^T\mathbf{x} = \sum_{i=0}^{d} \hat{w}_i x_i$$

	Dates	Temperature	Humidity	Rain
2	08-07-2018	25.46875	82.1875	6.75
3	09-07-2018	26.19298	83.1491	1761.75
4	10-07-2018	25.17021	85.3404	652.5
5	11-07-2018	NaN	87.6866	963
6	12-07-2018	24.06923	87.6462	254.25
7	13-07-2018	21.20779	95.9481	339.75
8	15-07-2018	23.48571	96.1714	38.25
9	18-07-2018	NaN	98.5897	29.25
10	19-07-2018	25.09346	88.3271	4.5
11	20-07-2018	25.39423	90.4327	112.5
12	21-07-2018	NaN	94.5378	735.75
13	22-07-2018	22.5098	99	607.5
14	23-07-2018	22.904	98	717.75
15	24-07-2018	NaN	99	513
16	25-07-2018	23.18182	98.9697	195.75

	Dates	Temperature	Humidity	Rain
2	08-07-2018	25.46875	82.1875	6.75
3	09-07-2018	26.19298	83.1491	1761.75
4	10-07-2018	25.17021	85.3404	652.5
5	11-07-2018	24.2	87.6866	963
6	12-07-2018	24.06923	87.6462	254.25
7	13-07-2018	21.20779	95.9481	339.75
8	15-07-2018	23.48571	96.1714	38.25
9	18-07-2018	21.5	98.5897	29.25
10	19-07-2018	25.09346	88.3271	4.5
11	20-07-2018	25.39423	90.4327	112.5
12	21-07-2018	23.7	94.5378	735.75
13	22-07-2018	22.5098	99	607.5
14	23-07-2018	22.904	98	717.75
15	24-07-2018	21.6	99	513
16	25-07-2018	23.18182	98.9697	195.75

Summary: Regression

- Regression analysis is used to model the relationship between one or more independent (predictor) variable and a dependent (response) variable
- Response is some function of one or more input variables
- Linear regression: Response is linear function of one or more input variables
  - If the response is linear function of one input variable, then it is simple linear regression (straight-line fitting)
  - If the response is linear function of two or more input variable, then it is multiple linear regression (linear surface fitting or hyperplane fitting)

33

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

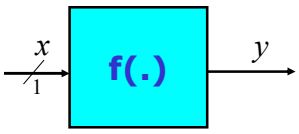
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

34

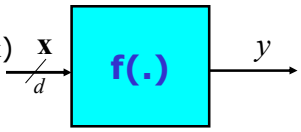
Supervised Machine Learning: Regression
Nonlinear Regression

Nonlinear Regression

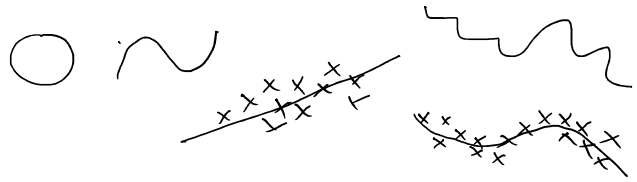
- Nonlinear approach to model the relationship between a scalar response, (y) (or dependent variable) and one or more predictor variables, (x *or* $\mathbf{x}$) (or independent variables)
- The response is going to be the nonlinear function of input (one or more independent variables)
- Simple nonlinear regression (Polynomial curve fitting, Neural Network):
  - Single independent variable (x)
  - Single dependent variable (y)
  - *Fitting a curve*



- Multiple nonlinear regression (Polynomial regression, Neural Network):
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)
  - *Fitting a surface*



2



Supervised Machine Learning: Regression

Polynomial Curve Fitting

approximate

Polynomial Curve Fitting

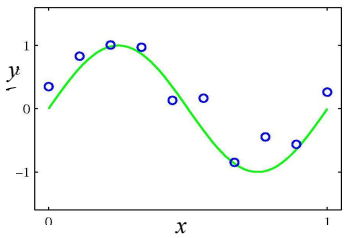
- Given:-Training data:

$$\mathcal{D} = \{x_n, y_n\}_{n=1}^{N},\ x_n \in \mathbb{R}^1 \text{ and } y_n \in \mathbb{R}^1$$

- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_p x_n^p = \sum_{j=0}^{p} w_j x_n^j$$

- Here, $1, x_n, x_n^2, x_n^3, \ldots, x_n^p$ are the monomials of polynomial up to degree p



$y = f(x, \mathbf{w})$

- The coefficients $\mathbf{w}=[w_0, w_1, \ldots, w_p]$ are parameters of polynomial curve (regression coefficients) ***- Unknown***
- Polynomial function $f(x_n, \mathbf{w})$ is a nonlinear function of x_n and
- Function $f(x_n, \mathbf{w})$ is a linear function of coefficients $\mathbf{w}$
  - ***Linear model for regression***

4

Polynomial Curve Fitting: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of $\mathbf{w}$

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_p x_n^p$$

$$\underset{\mathbf{w}}{\text{minimize}} \quad E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

- Minimize the error such that the coefficients $\mathbf{w}$ represent the parameter of polynomial curve that best fit the training data

5

Polynomial Curve Fitting: Training Phase

- Given:- Training data: $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of $\mathbf{w}$

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_p x_n^p$$

$$\underset{\mathbf{w}}{\text{minimize}} \quad E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

- The error function is a
  - quadratic function of the coefficients $\mathbf{w}$ and
  - The derivatives of error function with respect to the coefficients will be linear in the elements of $\mathbf{w}$
- Hence the minimization of the error function has unique solution and found in closed form

6

Polynomial Curve Fitting: Training Phase

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_p x_n^p = \sum_{j=0}^{p} w_j x_n^j$$

- Lets consider: $x_n \;\; x_n^2 \;\; x_n^3 \;\; \ldots \;\; x_n^p$ p is degree of polynomial

$$\downarrow \quad \downarrow \quad \downarrow \quad \ldots \quad \downarrow$$

$$z_{n1} \;\; z_{n2} \;\; z_{n3} \;\; \ldots \;\; z_{np}$$

$$\hat{y}_n = f(\mathbf{z}_n, \mathbf{w}) = w_0 + w_1 z_{n1} + w_2 z_{n2} + \ldots + w_p z_{np}$$

$$\hat{y}_n = f(\mathbf{z}_n, \mathbf{w}) = \sum_{j=0}^{p} w_j z_{nj} = \mathbf{w}^\top \mathbf{z}_n$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_p]^\top$ and $\mathbf{z}_n = [1, z_{n1}, \ldots, z_{np}]^\top$

$p+1$ $p+1$

$z_{np} = x_{n1}^p$

7

Polynomial Curve Fitting: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(f(\mathbf{z}_n, \mathbf{w}) - y_n \right)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \tilde{0}$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^{N} \left(\sum_{j=0}^{p} w_j z_{nj} - y_n \right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^{N} \left(\mathbf{w}^\top \mathbf{z}_n - y_n \right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

8

Polynomial Curve Fitting: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(f(\mathbf{z}_n, \mathbf{w}) - y_n\right)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$
- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathsf{T}}\mathbf{z}_n - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\hat{\mathbf{w}} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{y}$$

– Assumption: $p < N$

$\mathbf{Z}$ is Vandermonde matrix

$$\mathbf{Z} = \begin{bmatrix} 1 & z_{11} & z_{12} & \cdots & z_{1p} \\ 1 & z_{21} & z_{22} & \cdots & z_{2p} \\ - & - & - & - & - \\ 1 & z_{n1} & z_{n2} & \cdots & z_{np} \\ - & - & - & - & - \\ 1 & z_{N1} & z_{N2} & \cdots & z_{Np} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ - \\ y_n \\ - \\ y_N \end{bmatrix}$$

where, $z_{nj} = x_n^j$

9

Polynomial Curve Fitting: Testing

- Optimal coefficient vector $\mathbf{w}$ is given by

$$\hat{\mathbf{w}} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{y}$$

$$\hat{\mathbf{w}} = \mathbf{Z}^{+}\mathbf{y}$$

where $\mathbf{Z}^{+} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}$ is the pseudo inverse of matrix $\mathbf{Z}$

- For any test example x, the predicted value is given by:

$$\hat{y} = f(x, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\mathsf{T}}\mathbf{z} = \sum_{j=0}^{p}\hat{w}_i x^j$$

- The prediction accuracy is measured in terms of squared error: $E = \left(\hat{y} - y\right)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t}\sum_{n=1}^{N_t}\left(\hat{y}_n - y_n\right)^2}$$

10

Determining p, Degree of Polynomial

- This is determined experimentally
- Starting with p=1, test set is used to estimate the accuracy, in terms of error, of the regression model
  - Note: The polynomial degree p=1 is equivalent to simple linear regression (straight-line regression)
- This process is repeated each time by incrementing p
- The regression model with p that gives the minimum error on test set may be selected

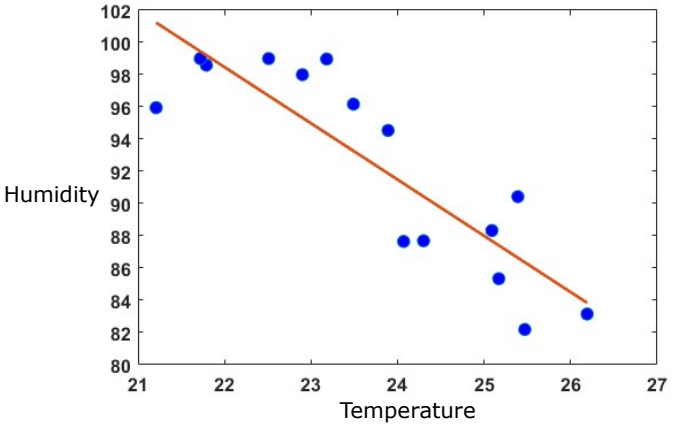
11

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 1

$$\hat{\mathbf{w}} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{y}$$ $\mathbf{Z}$ is 15 x 2 matrix

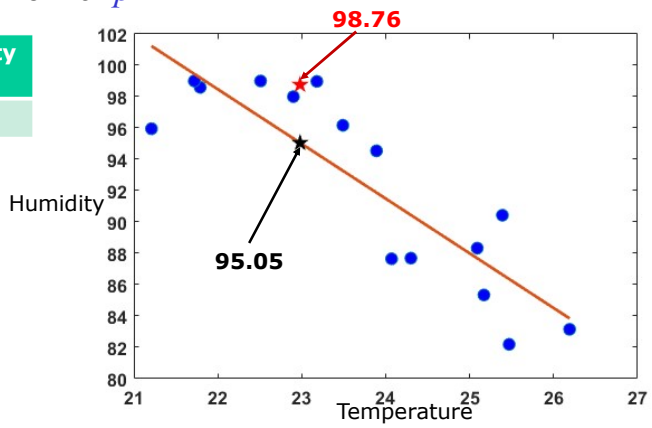


12

Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 1

Temp (x)	Humidity (y)
22.98	--



- Predicted humidity: 95.05
- Actual humidity: 98.76
- Squared error: **13.77**

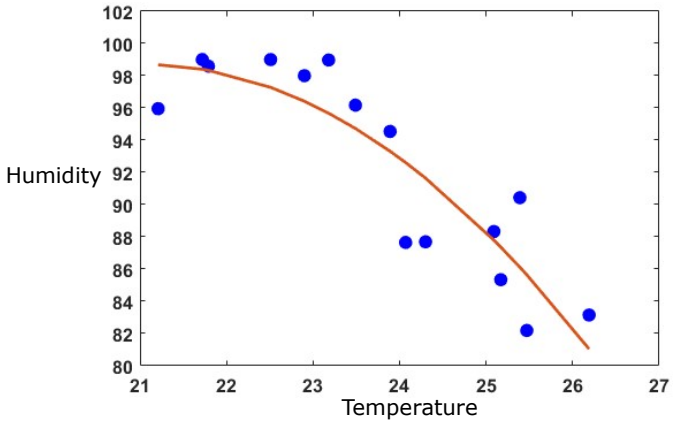
13

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 2

$$\hat{\mathbf{w}} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{y}$$ $\mathbf{Z}$ is 15 x 3 matrix

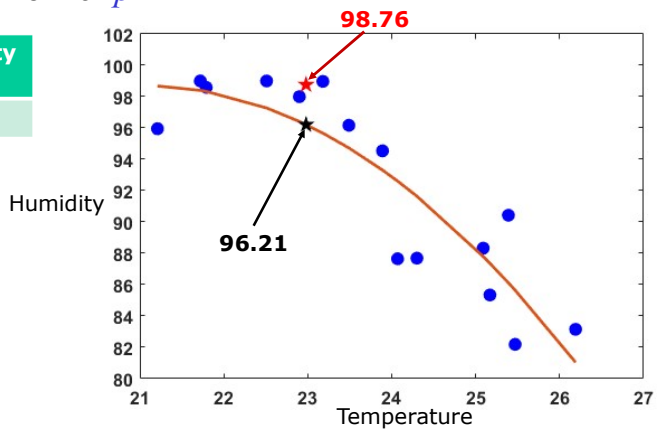


14

Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 2

Temp (x)	Humidity (y)
22.98	--



- Predicted humidity: 96.21
- Actual humidity: 98.76
- Squared error: **06.49**

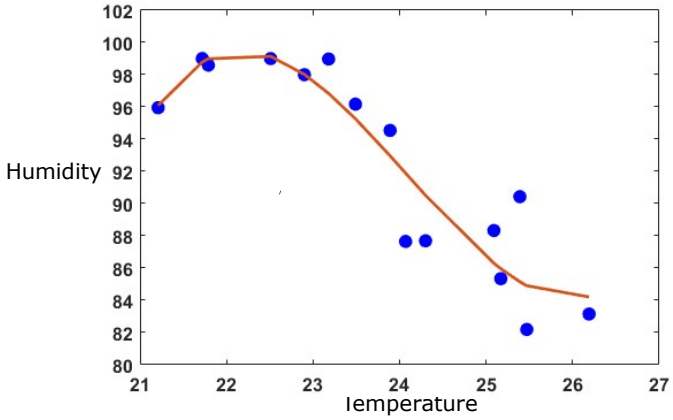
15

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 3

$$\hat{\mathbf{w}} = \left(\mathbf{Z}^{\mathsf{T}}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{y}$$ $\mathbf{Z}$ is 15 x 4 matrix

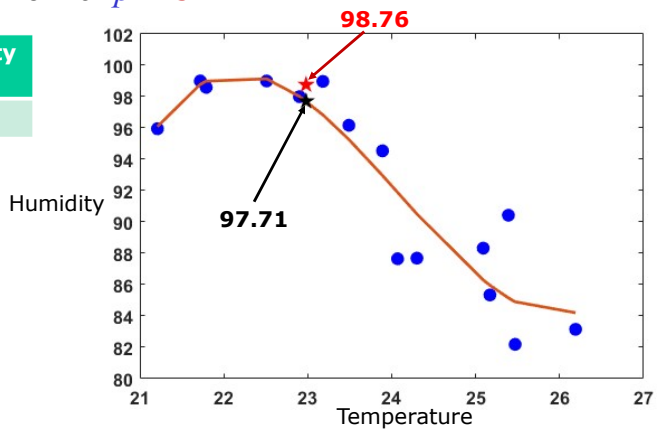


16

Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 3

Temp (x)	Humidity (y)
22.98	--



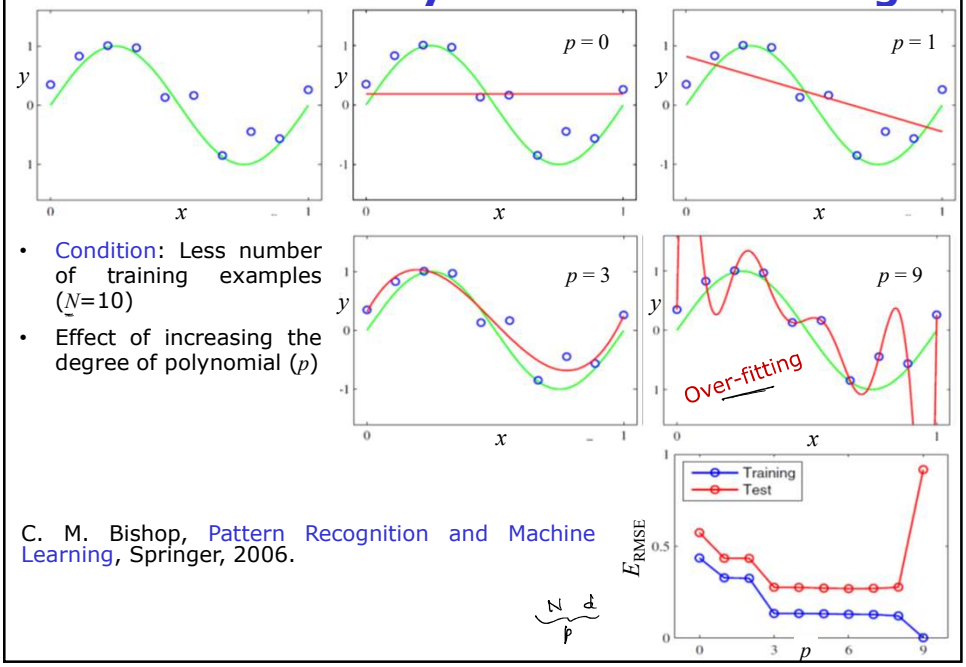
- Predicted humidity: 97.71
- Actual humidity: 98.76
- Squared error: **01.11**

p < N

p > N

17

Illustration: Polynomial Curve Fitting

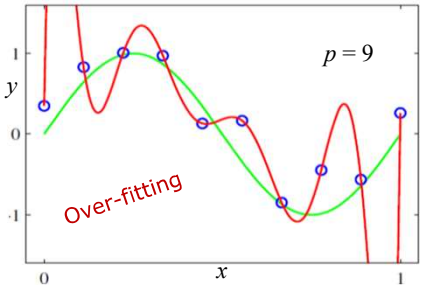


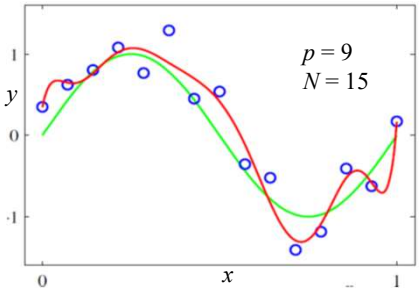
- Condition: Less number of training examples (N=10)
- Effect of increasing the degree of polynomial (p)

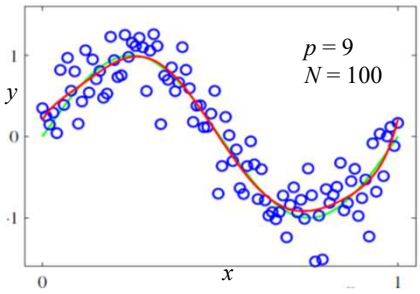
C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

N ≤ p

Illustration: Polynomial Curve Fitting







- Increasing the size of the data set reduces the over-fitting problem

C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

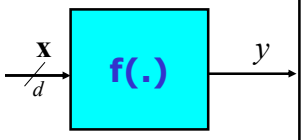
19

Supervised Machine Learning: Regression

Polynomial Regression

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$
- For 2-dimensional input, $\mathbf{x}_n=[x_{n1}, x_{n2}]^\intercal$ and degree, $p=2$

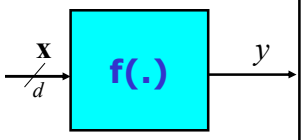
$$\boldsymbol{\varphi}(\mathbf{x}_n) = [\varphi_0(\mathbf{x}_n),\ \varphi_1(\mathbf{x}_n),\ \varphi_2(\mathbf{x}_n),\ \varphi_3(\mathbf{x}_n),\ \varphi_4(\mathbf{x}_n),\ \varphi_5(\mathbf{x}_n)]^\intercal$$

$$\boldsymbol{\varphi}(\mathbf{x}_n) = [\ 1,\quad x_{n1},\quad x_{n2},\quad x_{n1}^2,\quad x_{n2}^2,\quad x_{n1}x_{n2}\ ]^\intercal \qquad m = 6$$

21

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$
- For 2-dimensional input, $\mathbf{x}_n=[x_{n1}, x_{n2}]^\intercal$ and degree, $p=2$

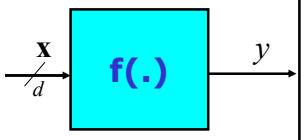
$$y_n = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = w_0 + w_1 x_{n1} + w_2 x_{n2} + w_3 x_{n1}^2 + w_4 x_{n2}^2 + w_5 x_{n1}x_{n2}$$

$$y_n = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = w_0\varphi_0(\mathbf{x}_n) + w_1\varphi_1(\mathbf{x}_n) + w_2\varphi_2(\mathbf{x}_n) + w_3\varphi_3(\mathbf{x}_n) + w_4\varphi_4(\mathbf{x}_n) + w_5\varphi_5(\mathbf{x}_n)$$

22

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

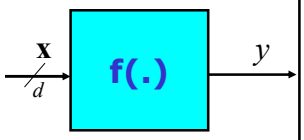
  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$
- For 2-dimensional input, $\mathbf{x}=[x_1, x_2]^T$ and degree, $p=3$

$\phi(\vec{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1x_2, x_1x_2^2, x_2x_1^2]^T$ $m = 10$

23

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

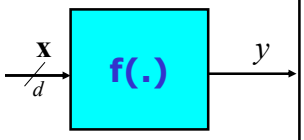
  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$
- For 3-dimensional input, $\mathbf{x}=[x_1, x_2, x_3]^T$ and degree, $p=2$

$\phi(\vec{x}) = [1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3]^T$ $m = 10$

24

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - One or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

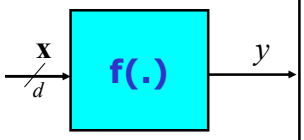
  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$

The number of monomials m for the polynomial of degree p and the dimension of d is given by $m = \frac{(d+p)!}{d!\,p!}$

25

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
  - Two or more independent variable ($\mathbf{x}$)
  - Single dependent variable (y)



- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

  - m is the number of monomials of polynomial up to degree p
  - $\varphi_j(\mathbf{x}_n)$ is the jth monomial of degree p for $\mathbf{x}_n$

$m = \frac{(d+p)!}{d!\,p!}$

Example: Let the dimension of input variable is d=6 and the polynomial of degree p=3

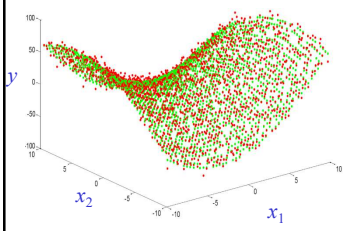
- The number of monomials m = **84**

26

Nonlinear Regression: Polynomial Regression

- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p:

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$



$y = f(\mathbf{x}_n, \mathbf{w})$

$\mathbf{x} = [x_1, x_2]^T$

Fitting a surface

- The coefficients $\mathbf{w}=[w_0, w_1, \ldots, w_{m-1}]$ are parameters of surface (polynomial function) (regression coefficients) ***- Unknown***
- Polynomial function $f(\mathbf{x}_n, \mathbf{w})$ is a nonlinear function of $\mathbf{x}_n$ and
- Function $f(\mathbf{x}_n, \mathbf{w})$ is a linear function of coefficients $\mathbf{w}$
  - ***Linear model for regression***

27

Polynomial Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n, \mathbf{w})$, in the training set for any given value of $\mathbf{w}$

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

$$\underset{\mathbf{w}}{\text{minimize}} \quad E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

- Minimize the error such that the coefficients $\mathbf{w}$ represent the parameter of polynomial curve that best fit the training data

28

Polynomial Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- **Method of least squares**: Minimizes the sum of the squared error between
  - all the actual data (y_n) i.e. actual dependent variable and
  - the estimate of line (predicted dependent variable ($\hat{y}_n$)) i.e. the function $f(x_n, \mathbf{w})$, in the training set for any given value of $\mathbf{w}$

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

$$\text{minimize}_{\mathbf{w}} \quad E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

- The error function is a
  - quadratic function of the coefficients $\mathbf{w}$ and
  - The derivatives of error function with respect to the coefficients will be linear in the elements of $\mathbf{w}$
- Hence the minimization of the error function has unique solution and found in closed form

29

Polynomial Regression : Training Phase

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w})$$
$$\hat{y}_n = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w})$$
$$\hat{y}_n = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$
$$\hat{y}_n = \mathbf{w}^{\mathsf{T}} \boldsymbol{\varphi}(\mathbf{x}_n)$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_{m-1}]^{\mathsf{T}}$ and

$$\boldsymbol{\varphi}(\mathbf{x}_n) = [\varphi_0(\mathbf{x}_n), \varphi_1(\mathbf{x}_n), \varphi_2(\mathbf{x}_n), \ldots, \varphi_{m-1}(\mathbf{x}_n)]^{\mathsf{T}}$$

30

Polynomial Regression : Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) - y_n\right)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n) - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_n) - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

31

Polynomial Regression : Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) - y_n\right)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_n) - y_n\right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\hat{\mathbf{w}} = \left(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{y}$$

– Assumption: $m < N$

$$\boldsymbol{\Phi} = \begin{bmatrix} \varphi_0(\mathbf{x}_1) & \varphi_1(\mathbf{x}_1) & \dots & \varphi_{m-1}(\mathbf{x}_1) \\ \varphi_0(\mathbf{x}_2) & \varphi_1(\mathbf{x}_2) & \dots & \varphi_{m-1}(\mathbf{x}_2) \\ \hline \varphi_0(\mathbf{x}_n) & \varphi_1(\mathbf{x}_n) & \dots & \varphi_{m-1}(\mathbf{x}_n) \\ \hline \varphi_0(\mathbf{x}_N) & \varphi_1(\mathbf{x}_N) & \dots & \varphi_{m-1}(\mathbf{x}_N) \end{bmatrix}$$

32

Polynomial Regression: Testing

- Optimal coefficient vector $\mathbf{w}$ is given by

  $\hat{\mathbf{w}} = \left(\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{y}$

  $\hat{\mathbf{w}} = \mathbf{\Phi}^{+}\mathbf{y}$

  where $\mathbf{\Phi}^{+} = \left(\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathsf{T}}$ is the pseudo inverse of matrix $\mathbf{\Phi}$
- For any test example $\mathbf{x}$, the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x})$$

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t}\sum_{n=1}^{N_t}(\hat{y}_n - y_n)^2}$$

33

Determining p, Degree of Polynomial

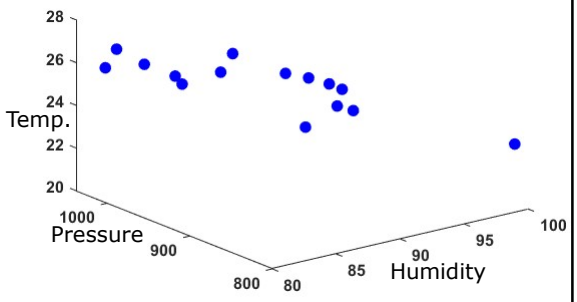
- This is determined experimentally
- Starting with p=1, test set is used to estimate the accuracy, in terms of error, of the regression model
  - Note: The polynomial degree p=1 is equivalent to multiple linear regression
- This process is repeated each time by incrementing p
- The regression model with p that gives the minimum error on test set may be selected

34

Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

$d = 2$



35

Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- **Training**:
- Polynomial Degree p = 3

$$\hat{\mathbf{w}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$\mathbf{\Phi}$ is 15 x 10 matrix

$$\text{Number of monomials, } m = \frac{(d+p)!}{d!\,p!} = \frac{(2+3)!}{2!*3!} = 10$$

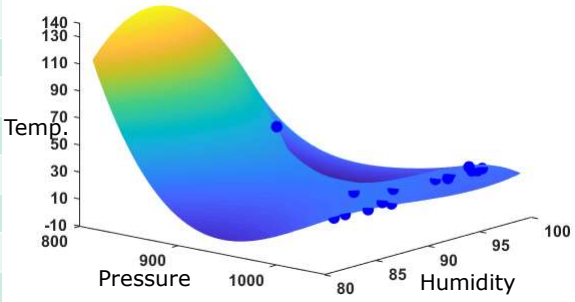
36

Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- **Training**:
- Polynomial Degree p = 3

$$\hat{\mathbf{w}} = \left(\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{y}$$



37

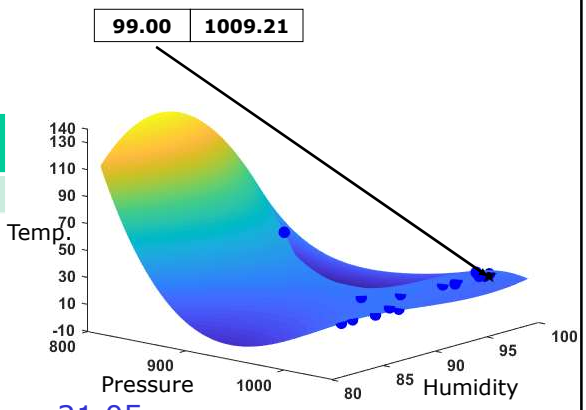
Illustration of Polynomial Regression: Temperature Prediction - Test

- Degree of polynomial p = 3

$$\hat{\mathbf{w}} = \left(\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{y}$$

Humidity (x_1)	Pressure (x_2)	Temp (y)
99.00	1009.21	-

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x})$$

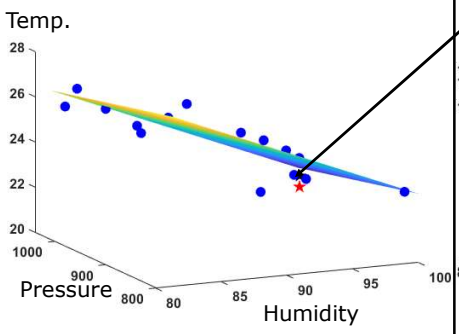


- Predicted Temperature: 21.05
- Actual Temperature: 21.24
- Squared error: **0.035**

38

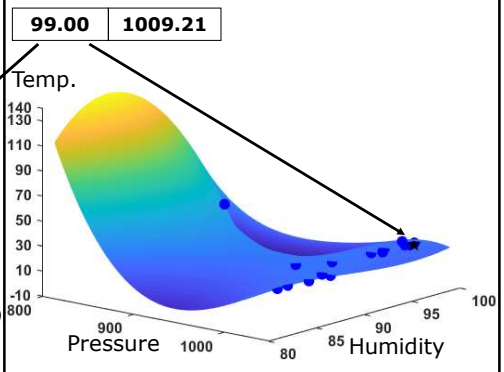
Multiple Linear Regression vs Polynomial Regression Temperature Prediction

- Multiple Linear Regression



- Predicted Temperature: 21.72
- Actual Temperature: 21.24
- Squared error: **0.2347**

- Polynomial Regression
  - Degree of polynomial $p = 3$



- Predicted Temperature: 21.05
- Actual Temperature: 21.24
- Squared error: **0.035**

39

Summary: Regression

- Regression analysis is used to model the relationship between one or more independent (predictor) variable and a dependent (response) variable
- Response is some function of one or more input variables
- Linear regression: Response is linear function of one or more input variables
  - If the response is linear function of one input variable, then it is simple linear regression (straight-line fitting)
  - If the response is linear function of two or more input variable, then it is multiple linear regression (linear surface fitting or hyperplane fitting)
- Nonlinear regression: Response is nonlinear function of one or more input variables
  - Polynomial regression: Response is nonlinear function approximated using polynomial function up to degree p of one or more input variables
  - When the degree of polynomial (p) is 1, then it is linear regression

40

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

41

Supervised Machine Learning: Regression

Time Series Prediction

Time Series Data

- Time series is a sequential set of data points, measured typically over successive times
- Time series data are simply a collection of observations gathered over time
- Time series is a time oriented sequence of observations on a variable of interest
- It is clearly structured and numeric in nature
- Time series data is collected at some intervals
  - These intervals can be as large as years or as small as seconds
- Example:
  - Weekly sales – time interval is week
  - Daily temperature in Kamand – time interval is day

2

Time Series Data

- Time series is a sequential set of data points, measured typically over successive times
- Time series data are simply a collection of

Date/ Time	Temperature (C)/ Humidity (%)	Pressure (Pa)	Rain (inches)	Light Intensity (lux)	Accelerations (g)	Force (N)	Moisture (%)
2017-09-06 18:44:32	23.00,56.00	617.64	0.01	3	0.52,0.31,-0.80,0.00,0.00,0.00,31.36,-159.01	0.02	81.00
2017-09-06 18:33:32	24.00,58.00	619.47	0.01	12	0.52,0.30,-0.79,0.00,0.00,0.00,31.45,-159.12	0.02	82.00
2017-09-06 18:22:39	24.00,58.00	623.37	0.00	71	0.52,0.31,-0.80,0.00,0.00,0.00,31.35,-158.88	0.02	83.00
2017-09-06 18:11:31	25.00,60.00	627.02	0.05	194	0.51,0.31,-0.80,0.00,0.00,0.00,30.80,-159.00	0.02	81.00

Daily temperature in Kamand time interval is day

- Time series data is collected at some intervals
  - These intervals can be as large as years or as small as seconds

3

Time Series Data

- Time series data is given as:

$$\mathbf{X} = (x_1, x_2, \ldots, x_t, \ldots, x_T)$$

  - x_t is the observation at time t
  - T be the number of observations

- Scope: We consider single variable x_t

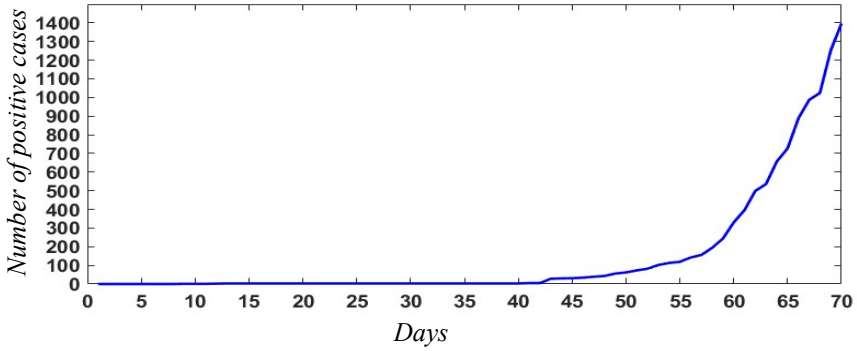
Eg: Daily Temp. recorded.

$X = \text{Temp} = (\text{Temp}_1, \text{Temp}_2, \ldots, \text{Temp}_T)$

4

Time Series Data

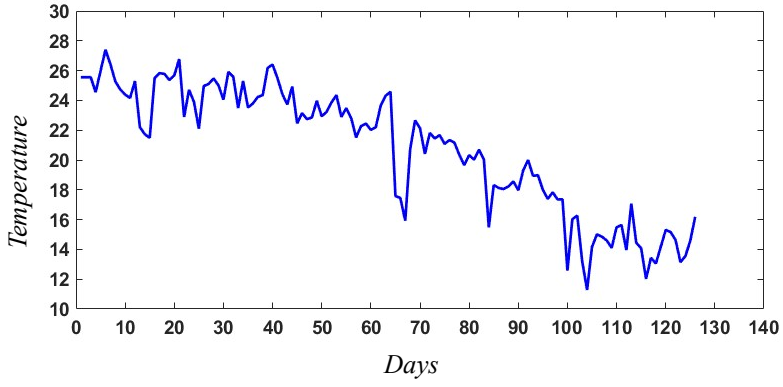
- Trend: Shows how data moves over a period of time
  - COVID positive cases in India between 22 Jan 2020 to 31 March 2020



5

Time Series Data

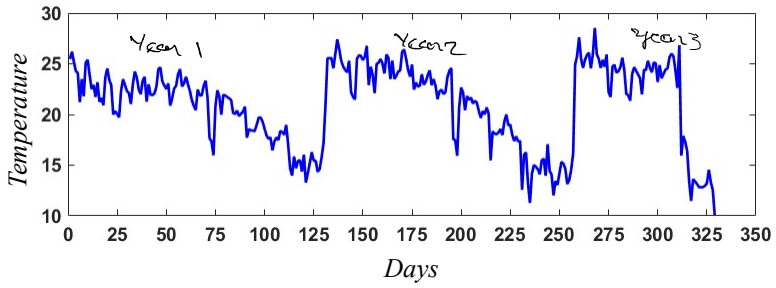
- Trend: Shows how data moves over a period of time
  - Daily temperature at IIT Mandi from June-Nov 2018



6

Time Series Data

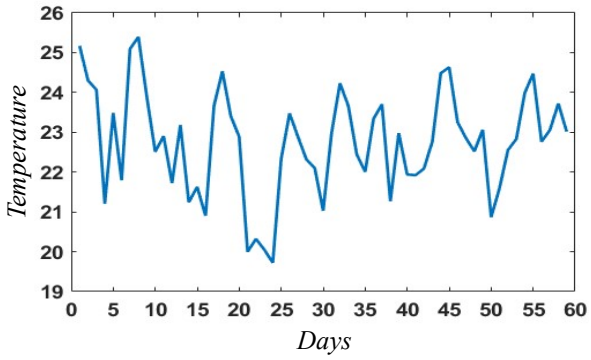
- Seasonality: A type of pattern which repeats over a specific period of time
  - Daily temperature recorded in IIT Mandi for 3 years
    - Duration of recorded: July-Nov (2017-2019)



7

Time Series Data

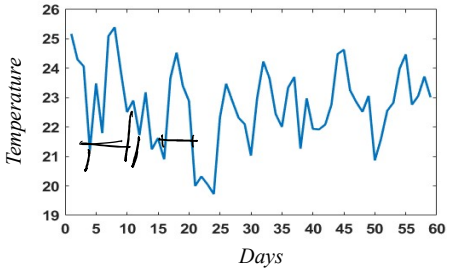
- Random or error: Series does not have any trend, seasonality or cyclic component
  - Daily temperature recorded in IIT Mandi (1 July - 30 August 2019)



8

Stationary Time Series

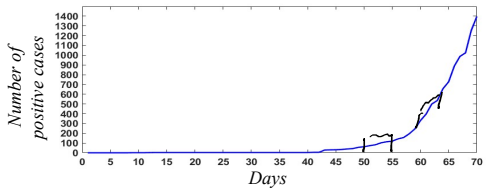
- Stationary time series:
  - Statistical properties remain same at any given interval of time
  - Time independent kind of series
  - Mean and variance should be time independent
    - Mean and variance computed at any one part of the series should be similar to that of the mean and variance computed at another part
  - Stationary time series are easier to predict
- Example:
  - Daily temperature recorded in IIT Mandi (1 July - 30 August 2019)



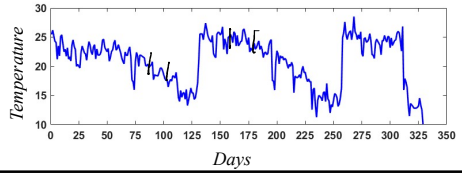
9

Non-stationary Time Series

- Non-stationary time series: Time series having trends or seasonality
  - Mean and variance are not time independent
- Example:
  - COVID positive cases in India between 22 Jan 2020 to 31 March 2020



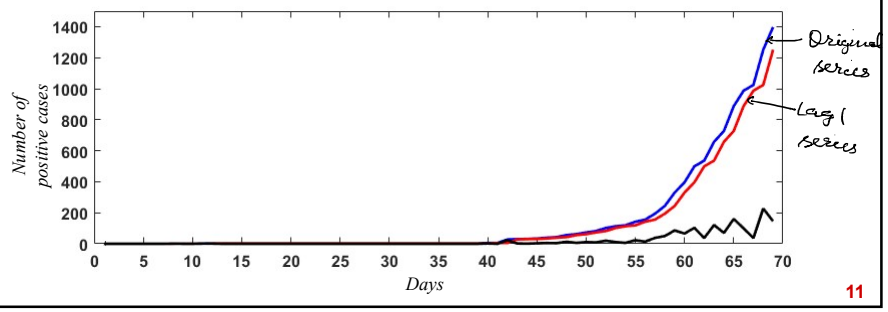
  - Daily temperature recorded in IIT Mandi for 3 years



10

Differencing

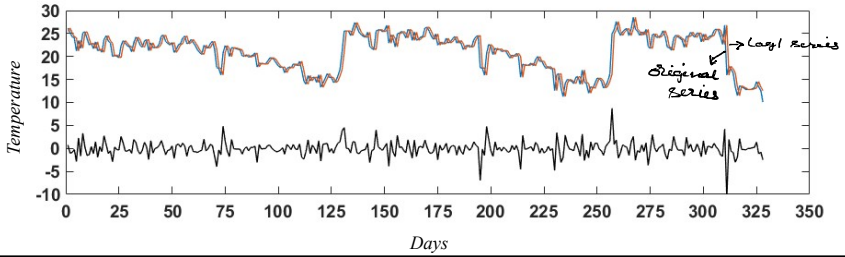
- Non-stationary time series are made stationary by differencing
  - Difference between the original series and the lag series
  - Lag is the shift in the time series by a given number of observations
    - Lag 1: Shift by one time step
    - Lag 2: Shift by two time step
  - By differencing, non-stationary time series become more stationary



11

Differencing

- Non-stationary time series are made stationary by differencing
  - Difference between the original series and the lag series
  - Lag is the shift in the time series by a given number of observations
    - Lag 1: Shift by one unit
    - Lag 2: Shift by two unit
  - By differencing, non-stationary time series become more stationary
    - Mean and variance become almost same in the different parts
  - Example: Daily temperature recorded in IIT Mandi for 3 years



12

Time Series Data and Dependence

- Time series data is given as:

$$\mathbf{X} = (x_1, x_2, \ldots, x_t, \ldots, x_T)$$

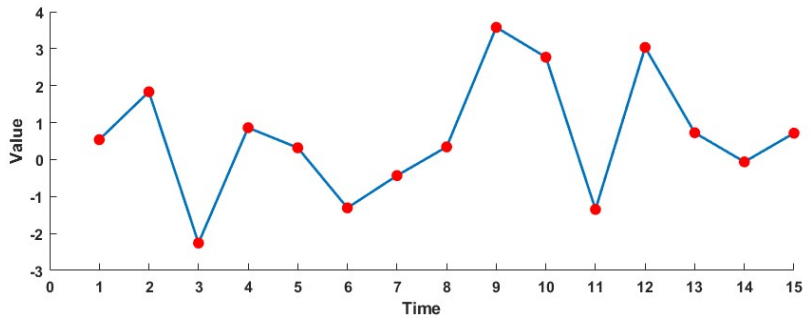
  - x_t is the observation at time t
  - T be the number of observations
- In time series data, value of each element at time t (x_t) is dependent on the values elements at previous p time steps (x_{t-1} , x_{t-2}, …, x_{t-p}) – p time lag
  - Lag is the shift in the time series by a given number of observations

13

Time Series Data and Dependence

- Example: Data series in i.i.d
  - x_t is a random number drawn from $\mathcal{N}(0,1)$
- Each element at time t (x_t) is not dependent on the values elements at previous p time steps (x_{t-1} , x_{t-2}, …, x_{t-p}) – p time lag

0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06	0.71

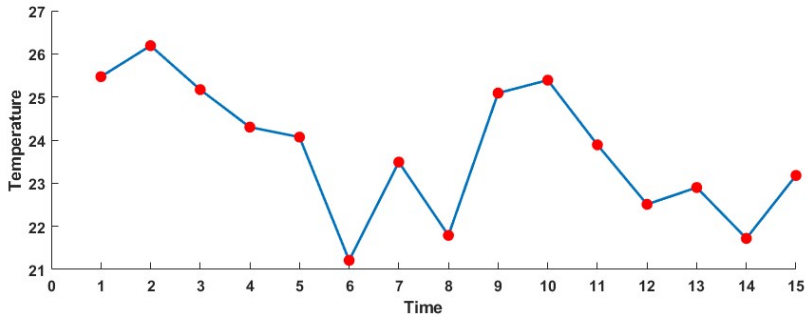


14

Time Series Data and Dependence

- Example: Daily temperature at IIT Mandi
- Each element at time t (x_t) is dependent on the values elements at previous p time steps (x_{t-1} , x_{t-2}, …, x_{t-p}) – p time lag
  - *Temperature recorded for 15 days (1 Sept. 2019 – 15 Sept. 2019)*

25.47	26.19	25.17	24.3	24.07	21.21	23.49	21.79	25.09	25.39	23.89	22.51	22.9	21.72	23.18



15

Checking Dependency

- It's not always easy to just look at a time-series plot and say whether or not the series is independent
- x_t in a series is independent means that knowing previous values doesn't help you to predict the next value
  - Knowing x_{t-1} doesn't help to predict x_t
  - More generally, knowing x_{t-1}, x_{t-2}, …, x_{t-p} doesn't help to predict x_t
    - p is the number of previous time step (time lag)
- Dependency of each element at time t (x_t) with the values of elements at previous p time steps (x_{t-1} , x_{t-2}, …, x_{t-p}) is observed using autocorrelation

16

Checking Dependency - Autocorrelation

- The relationship between variables is called correlation
- Autocorrelation: The correlation calculated between the variable and itself at previous time steps
- Example: Data series in i.i.d
  - Autocorrelation between x_t and x_{t-p} – Pearson correlation coefficient between original series and lag-p series

Original Series

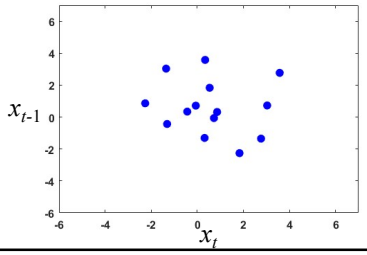
x_t	0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06	0.71

Lag-1 Series

x_{t-1}	0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06

  - Autocorrelation:

	x_t	x_{t-1}
x_t	1	**-0.1242**
x_{t-1}	**-0.1242**	1



17

Checking Dependency - Autocorrelation

- The relationship between variables is called correlation
- Autocorrelation: The correlation calculated between the variable and itself at previous time steps
- Example: Daily temperature at IIT Mandi
  - Autocorrelation between x_t (original series) and x_{t-1} (Lag-1 series)

Original Series

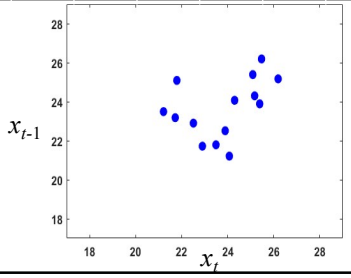
x_t	25.47	26.19	25.17	24.3	24.07	21.21	23.49	21.79	25.09	25.39	23.89	22.51	22.9	21.72	23.18

Lag-1 Series

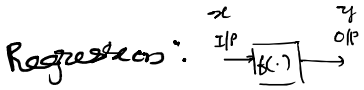
x_{t-1}	25.47	26.19	25.17	24.3	24.07	21.21	23.49	21.79	25.09	25.39	23.89	22.51	22.9	21.72

  - Autocorrelation:

	x_t	x_{t-1}
x_t	1	**0.4054**
x_{t-1}	**0.4054**	1



18



Autoregression (AR)

Autoregression (AR)

- Regression on the values of same attribute
- Autoregression is a time series model that
  - uses observations from previous time steps as input to a linear regression equation to predict the value at the next time step
    - Output variable: value at next time step
    - Input variable: observations from previous time step
  - Output variable is a linear function of input variables

$(x_1\ x_2\ x_3\ x_4)$

O/P

$x_4 \leftarrow h(x_3)$

20

Autoregression (AR)

- Autoregression (AR): Regression on the values of same attribute
  - It is a time series model
  - Linear regression model that uses observations from previous p time steps as input to predict the value at the next time step
  - It makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
  - The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
  - Dependency of each element at time t (x_t) with the values of elements at previous p time steps (x_{t-1} , x_{t-2}, …, x_{t-p}) is observed using autocorrelation
  - Autocorrelation: The correlation calculated between the variable and itself at previous time steps

21

Autoregression (AR) Model

- Autoregression (AR) is a linear regression model that uses observations from previous time steps as input to predict the value at the next time step
- An autoregression (AR) model makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
- The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
- Interestingly, if all lag variables (x_{t-1} , x_{t-2}, …, x_{t-p}) show low or no correlation with the output variable (x_t), then it suggests that the time series problem may not be predictable
- This can be very useful when getting started on a new dataset

22

Autoregression (AR) Model

- Building an AR model depends on how many time lag (p) is considered
- AR(1) model: AR model using one time lag (p=1)
  - uses x_{t-1} i.e. value of previous time step to predict x_t

23

Illustration AR(1) Model – Prediction of Temperature

Date	Temp (x_{t-1})	Temp (x_t)	Date
	I/P	~~25.47~~	~~Sept 1~~
Sept 1	25.47	26.19	Sept 2
Sept 2	26.19	25.17	Sept 3
Sept 3	25.17	24.30	Sept 4
Sept 4	24.30	24.07	Sept 5
Sept 5	24.07	21.21	Sept 6
Sept 6	21.21	23.49	Sept 7
Sept 7	23.49	21.79	Sept 8
Sept 8	21.79	25.09	Sept 9
Sept 9	25.09	25.39	Sept 10
---	---	---	---
Oct 28	22.76	23.06	Oct 29
Oct 29	23.06	23.72	Oct 30
Oct 30	23.72	23.02	Oct 31

- T, the number of observations = 61
- Independent variable:
  - Temperature at the time t-1
- Dependent variable:
  - Temperature at the time t

24

AR(1) Model

- AR(1) model: AR model using one time lag (p=1)
  - uses x_{t-1} i.e. value of previous time step to predict x_t
- Given: Time series data: $\mathbf{X} = (x_1, x_2, \ldots, x_t, \ldots, x_T)$
  - x_t is the observation at time t
  - T be the number of observations
- AR(1) model is given as: $x_t = f(x_{t-1}, w_0, w_1) = w_0 + w_1 x_{t-1}$
  - The coefficients w_0 and w_1 are parameters of straight-line (regression coefficients) ***- Unknown***
- The regression coefficients are obtained as seen in simple linear regression (straight-line regression) using least square method

25

p=1

AR(1) Model - Training

- The regression coefficients are obtained as seen in simple linear regression (straight-line regression) using least square method
- Minimize the squared error between the actual data (x_t) at time t and the estimate of linear function (predicted variable ($\hat{x}_t$)) i.e. the function $f(x_{t-1}, w_0, w_1)$

$$\hat{x}_t = f(x_{t-1}, w_0, w_1) = w_0 + w_1 x_{t-1}$$

$$\underset{w, w_0}{\text{minimize}} \; E(w_0, w_1) = \frac{1}{2} \sum_{t=2}^{T} (\hat{x}_t - x_t)^2$$

- The optimal $\hat{w}_0$ and $\hat{w}_1$ is given as

$$\hat{w}_1 = \frac{\sum_{t=1}^{T} (x_{t-1} - \mu_{t-1})(x_t - \mu_t)}{\sum_{t=1}^{T} (x_{t-1} - \mu_{t-1})^2}$$

$$\hat{w}_0 = \mu_t - w_1 \mu_{t-1}$$

Berapa

- μ_{t-1}: sample mean of variables at time t-1, x_{t-1} (sample mean of Lag-1)
- μ_t: sample mean of variables at time t, x_t (sample mean of original series)

26

AR(1) Model: Testing

- For any test example at time t-1, x_{t-1}, the predicted value at time t, $\hat{x}_t$ is given by:

$$\hat{x}_t = f(x_{t-1}, w_0, w_1) = \hat{w}_0 + \hat{w}_1 x_{t-1}$$

27

Evaluation Metrics for Time Series Prediction: Squared Error and Root Mean Squared Error

- The prediction accuracy is measured in terms of squared error: $E = (\hat{x}_t - x_t)^2$
  - x_t: actual value
  - $\hat{x}_t$: predicted value
- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error (RMSE):

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2}$$

- RMSE expressed in % as:

$$E_{\text{RMSE}}(\%) = \frac{\sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2}}{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} x_t} * 100$$

28

Evaluation Metrics for Time Series Prediction: Absolute Error and Mean Absolute Percentage Error (MAPE)

- Absolute error: $E_a = \frac{|x_t - \hat{x}_t|}{x_t}$
  - x_t: actual value
  - $\hat{x}_t$: predicted value
- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of mean absolute percentage error:

$$E_{\text{MAP}} = \left(\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} \frac{|x_t - \hat{x}_t|}{x_t} \right) * 100$$

29

Illustration AR(1) Model – Prediction of Temperature: Training

Temp (x_{t-1})	Temp (x_t)	Date
	25.47	Sept 1
25.47	26.19	Sept 2
26.19	25.17	Sept 3
25.17	24.30	Sept 4
24.30	24.07	Sept 5
24.07	21.21	Sept 6
21.21	23.49	Sept 7
23.49	21.79	Sept 8
21.79	25.09	Sept 9
25.09	25.39	Sept 10
---	---	---
22.76	23.06	Oct 29
23.06	23.72	Oct 30
23.72	23.02	Oct 31

- T, the number of observations = 61

$$\hat{w}_1 = \frac{\sum_{t=1}^{60} (x_{t-1} - \mu_{t-1})(x_t - \mu_t)}{\sum_{t=1}^{60} (x_{t-1} - \mu_{t-1})^2}$$

$$\hat{w}_0 = \mu_t - w_1 \mu_{t-1}$$

- μ_{t-1}: 22.81
- $\hat{w}_1$: 0.523
- μ_t : 22.85
- $\hat{w}_0$: 10.861

30

Illustration AR(1) Model – Prediction of Temperature: Test

- Predict Temperature for Nov 2

- $\hat{w}_1$: 0.523
- $\hat{w}_0$: 10.861

Nov 1

Temp (x_{t-1})	Temp (x_t)
22.30	-

$\hat{x}_t = \hat{w}_0 + \hat{w}_1 x_{t-1}$

- Predicted Temperature for Nov 2 : 22.52
- Actual Temperature on Nov 2 : 21.43
- Squared error : **1.19**
- Absolute error : **0.0509**

31

Autoregression Model

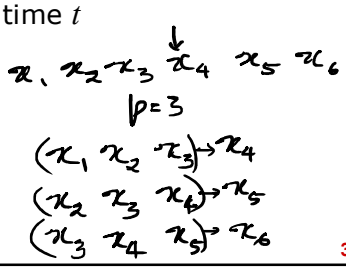
- AR(p) model: AR model using p time lags ($p < T$)
  - uses x_{t-1} , x_{t-2}, …, x_{t-p} i.e. value of previous p time step to predict x_t

32

Illustration AR(p) Model – Prediction of Temperature

Temp (x_{t-3})	Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
			25.47	Sept 1
		25.47	26.19	Sept 2
	25.47	26.19	25.17	Sept 3
25.47	26.19	25.17	24.30	Sept 4
26.19	25.17	24.30	24.07	Sept 5
25.17	24.30	24.07	21.21	Sept 6
24.30	24.07	21.21	23.49	Sept 7
24.07	21.21	23.49	21.79	Sept 8
21.21	23.49	21.79	25.09	Sept 9
---	---	---	---	---
22.83	23.98	24.47	22.76	Oct 28
23.98	24.47	22.76	23.06	Oct 29
24.47	22.76	23.06	23.72	Oct 30
22.76	23.06	23.72	23.02	Oct 31

- T, the number of observations = 61
- p = 3
- Independent variable:
  - Temperature at the time t-1 , t-2 and t-3
- Dependent variable:
  - Temperature at the time t



33

Autoregression Model

- AR(p) model: AR model using p time lags ($p < T$)
  - uses x_{t-1} , x_{t-2}, …, x_{t-p} i.e. value of previous p time step to predict x_t
- Given: Time series data: $\mathbf{X} = (x_1, x_2, \ldots, x_t, \ldots, x_T)$
  - x_t is the observation at time t
  - T be the number of observations
- AR(p) model is given as:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, w_0, w_1, \ldots, w_p) = w_0 + w_1 x_{t-1} + \ldots + w_p x_{t-p}$$

$$x_t = f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{p} w_j x_{t-j} = \mathbf{w}^{\mathsf{T}} \mathbf{x}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_p]^{\mathsf{T}}$ and $\mathbf{x} = [1, x_{t-1}, x_{t-2}, \ldots, x_{t-p}]^{\mathsf{T}}$

  - The coefficients w_0, w_1, …, w_p are parameters of hyperplane (regression coefficients) ***- Unknown***

34

AR (p) Model - Training

- The regression coefficients are obtained as seen in multiple linear regression with p input variables using least square method
- Minimize the squared error between the actual data (x_t) at time t and the estimate of linear function (predicted variable ($\hat{x}_t$)) i.e. the function $f(\mathbf{x},\mathbf{w})$

$$\hat{x}_t = f(\mathbf{x},\mathbf{w}) = w_0 + \sum_{j=1}^{p} w_j x_{t-j} = w_0 + \mathbf{w}^{\mathsf{T}}\mathbf{x}$$

$$\underset{\mathbf{w}}{\text{minimize}} \quad E(\mathbf{w}) = \frac{1}{2}\sum_{t=p+1}^{T}\left(\hat{x}_t - x_t\right)^2$$

- The autocorrelation statistics help to choose which lag variables (p) will be useful in a model

35

AR (p) Model - Training

- The optimal $\hat{\mathbf{w}}$ is given as $\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{x}^{(t)}$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{t-p} & \dots & x_{t-3} & x_{t-2} & x_{t-1} \\ 1 & x_{(t+1)-p} & \dots & x_{t-2} & x_{t-1} & x_t \\ - & - & - & - & - & - \\ 1 & x_{(t+n)-p} & \dots & x_{t+n-3} & x_{t+n-2} & x_{t+n-1} \\ - & - & - & - & - & - \\ 1 & x_{T-p} & \dots & x_{T-3} & x_{T-2} & x_{T-1} \end{bmatrix} \quad \mathbf{x}^{(t)} = \begin{bmatrix} x_t \\ x_{t+1} \\ - \\ x_{t+n} \\ - \\ x_T \end{bmatrix}$$

$\mathbf{X}$ is data matrix with time lag

- The autocorrelation statistics help to choose which lag variables (p) will be useful in a model

36

AR (p) Model: Testing

- The value at time t, $\hat{x}_t$ is predicted by taking values from past p time steps (x_{t-1}, x_{t-2}, …, x_{t-p}) as input:

$$\hat{x}_t = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{w}_0 + \sum_{j=1}^{p} \hat{w}_j x_{t-j} = \hat{\mathbf{w}}^{\mathsf{T}} \mathbf{x}$$

- The prediction accuracy is measured in terms of squared error:

$$E = (\hat{x}_t - x_t)^2$$

- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2}$$

- Mean absolute percentage error (MAPE) is also used as a measure

37

Illustration AR(p) Model – Prediction of Temperature: Checking Dependency

Temp (x_{t-3})	Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
			25.47	Sept 1
		25.47	26.19	Sept 2
	25.47	26.19	25.17	Sept 3
25.47	26.19	25.17	24.30	Sept 4
26.19	25.17	24.30	24.07	Sept 5
25.17	24.30	24.07	21.21	Sept 6
24.30	24.07	21.21	23.49	Sept 7
24.07	21.21	23.49	21.79	Sept 8
21.21	23.49	21.79	25.09	Sept 9
---	---	---	---	---
22.83	23.98	24.47	22.76	Oct 28
23.98	24.47	22.76	23.06	Oct 29
24.47	22.76	23.06	23.72	Oct 30
22.76	23.06	23.72	23.02	Oct 31

- p = 3
- T, the number of observations = 61
- Autocorrelation between x_t and x_{t-1} : 0.54
- Autocorrelation between x_t and x_{t-2} : 0.25
- Autocorrelation between x_t and x_{t-3} : -0.08
- An autocorrelation is deemed significant if

$$|\text{autocorrelation}| > \frac{2}{\sqrt{T}} = 0.25$$

- Time lag p=2 is sufficient as x_t is significant with x_{t-1} and x_{t-2}

38

Illustration AR(p) Model – Prediction of Temperature: Training

Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
		25.47	Sept 1
	25.47	26.19	Sept 2
25.47	26.19	25.17	Sept 3
26.19	25.17	24.30	Sept 4
25.17	24.30	24.07	Sept 5
24.30	24.07	21.21	Sept 6
24.07	21.21	23.49	Sept 7
21.21	23.49	21.79	Sept 8
23.49	21.79	25.09	Sept 9
---	---	---	---
23.98	24.47	22.76	Oct 28
24.47	22.76	23.06	Oct 29
22.76	23.06	23.72	Oct 30
23.06	23.72	23.02	Oct 31

- $p = 2$
- T, the number of observations = 59
- Multiple linear regression with number of input variables = 2

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{x}^{(t)}; \quad \hat{\mathbf{w}} \in \mathbf{R}^3$$

39

Illustration AR(p) Model – Prediction of Temperature: Test

- Predict Temperature for Nov 2

$$\hat{\mathbf{w}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{x}^{(t)}; \quad \hat{\mathbf{w}} \in \mathbf{R}^3$$

Oct 31 Nov 1

Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)
23.02	22.30	--

- **AR(2) model:**
- Predicted Temperature for Nov 2 : 22.49
- Actual Temperature on Nov 2 : 21.43
- Squared error : **1.13**
- Absolute error : **0.0495**
- **AR(1) model:**
- Predicted Temperature for Nov 2 : 22.52
- Actual Temperature on Nov 2 : 21.43
- Squared error : **1.19**
- Absolute error : **0.0509**

40

Summary: Autoregression

- Autoregression (AR): Regression on the values of same attribute
  - It is a time series model
  - Linear regression model that uses observations from previous p time steps as input to predict the value at the next time step
  - It makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
  - The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
- AR model can be performed on time series data with single variable or with multiple variables
- In this course we are limited only on the time series data with single variable

41

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

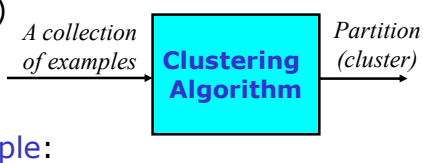
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

42

Unsupervised Machine Learning: Clustering

Clustering

- Process of grouping a set of examples (samples)
- Clustering generates a partition consisting of cohesive groups or clusters from given collection of examples (samples)



- For example:
  - Grouping students of this class based on gender
  - Grouping students of this class based the month of birth
  - Grouping students of this class based on the states
- The examples to be clustered are unlabelled
  - Do not rely on predefined classes
  - Learning by observation, rather than learning by examples.

2

Clustering

- Clustering is a two step process
  - Step1: Partition the collection of examples (clustering)
    - Learning by observation (training phase)
    - Unsupervised learning: Do not rely on predefined classes and class-labelled training examples
    - Group the collection of examples into finite number of clusters such that the examples that are similar to one another within the same cluster and are dissimilar to examples in other clusters
    - Obtaining cluster identity or optimal cluster representative
  - Step2: Assign cluster labels to new examples
    - Testing phase
- Classification also groups the data into classes, i.e., each class is associated with cluster. We know what the cluster is
  - However, the label of the classes are known during training phase – Learning by example
- In clustering, labels of clusters are unknown. We do not know what the cluster is
  - Clusters are learnt by observing the samples (parameters of clusters are learnt) – Learning by observation

3

Categorization of Clustering Methods

- Partitioning methods
- Hierarchical methods
- Density-based methods

4

Categorization of Clustering Methods

- Partitioning methods:
  - These methods construct K partitions of the data, where each partition represents a cluster
  - Idea: Cluster the collection of examples based on the distance between examples
  - Results in spherical shaped clusters
  1. K-means algorithm
  2. K-medoids algorithm
  3. Gaussian mixture model
- Hierarchical methods:
  - These methods create a hierarchical decomposition of the collection of examples
  - Results in spherical shaped clusters
  1. Agglomerative approach (bottom-up approach)
  2. Divisive approach (top-down approach)

5

Categorization of Clustering Methods

- Density-based methods:
  - These methods cluster collection of examples based on the notion of density
  - General idea: To continue growing the given cluster as long as density (number of examples) in the neighbourhood exceeds some threshold
  - Results in arbitrary shaped clusters
  - Example:
    - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

6

Unsupervised Machine Learning: Clustering

Summary: Introduction to Clustering

- Clustering is an unsupervised machine learning
  - Data used for learning (Train data) is unlabeled
  - Given these unlabeled data machine tries to identify the pattern based on similarity and similar patterns are placed in same group
  - Learning by observation
- Clustering methods are categorized into
  - Partitioning methods
  - Hierarchical methods
  - Density-based methods

8

Classical Partitioning Methods

- Partition the collection of examples into K clusters based on the distance between examples
  - The number of clusters K is decided empirically
- Centroid-based technique:
  - Cluster similarity is measured in regard to the sample mean of the examples within a cluster
  - Cluster centroid or center of gravity: Sample mean value of the examples within a cluster
  - Cluster center is used to represent the cluster
  - Example(s): K-means algorithm

    Gaussian mixture model (GMM)
- Representative object-based technique:
  - Actual example is considered to represent the cluster
  - One representative example per cluster
  - Example: K-medoids algorithm

9

Unsupervised Machine Learning: Clustering

K-Means Clustering Algorithm

K-Means Clustering Algorithm

- Dividing the data into K groups or partitions
- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$ and K
- Target: Partition the set $\mathcal{D}$ into K clusters (disjoint subsets), $\{\mathcal{D}_k\}_{k=1}^{K}$
  - Each of the clusters is associated with centers, $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$
  - Training phase: Come up with the centers of clusters
    - Cluster center acts as a cluster representative
- The examples closer to the cluster center forms the group (cluster)
- Euclidean distance with center of a cluster can be used as a measure of dissimilarity

11

K-Means Clustering Algorithm: Training Phase

- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$ and K

1. Initialize the cluster center, $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$ using randomly selected K data points in $\mathcal{D}$
2. Assign each data point $\mathbf{x}_n$ to nearest cluster center k^*

$$k^* = \arg\min_k \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \quad \textit{Squared Euclidian distance}$$

3. Update $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$: Re-compute $\boldsymbol{\mu}_k$ after assigning all the data points
   - Compute the mean vector for each cluster using the data assigned to each cluster

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_{\mathcal{D}_k} \mathbf{x}_n}{N_k}$$

$\mathcal{D}_k$: Data for cluster k

N_k: Number of examples in cluster k

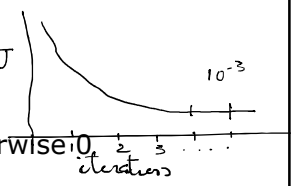
4. Repeat the steps 2 and 3 until the convergence

12

K-Means Clustering Algorithm: Training Phase

- Convergence criteria:
  - No change in the cluster assignment **OR**
  - The difference between the distortion measure (J) in the successive iteration falls below the threshold
    - Distortion measure (J) : Sum of the squares of the distance of each example to its assigned cluster center

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$$

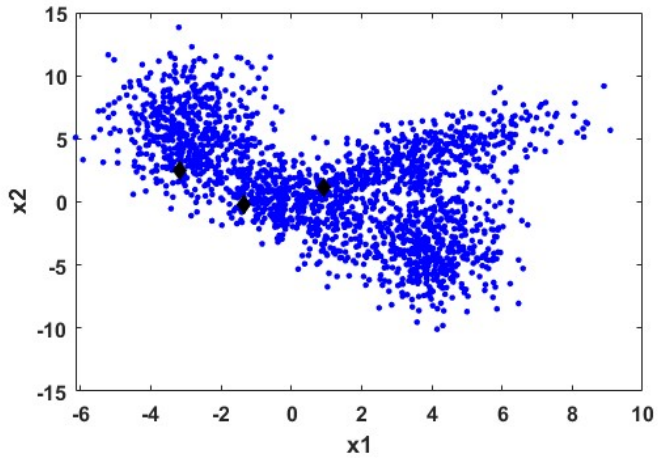


z_{nk} is 1 if $\mathbf{x}_n$ belongs to cluster k, otherwise 0

- Overall, K-means clustering algorithm involve in minimizing the distortion measure to obtain the optimal cluster center

13

Illustration of K-Means Clustering – Training Phase

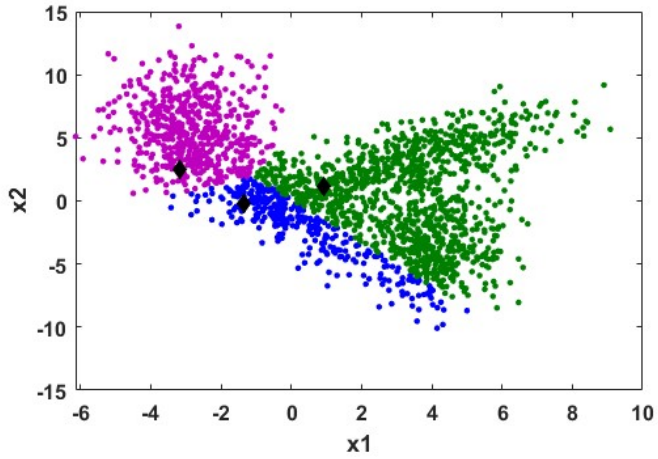


$K = 3$

Iteration 1

14

Illustration of K-Means Clustering – Training Phase

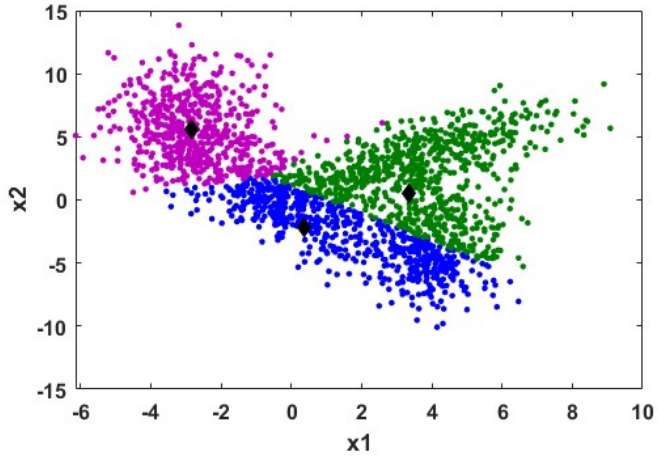


$K = 3$

Iteration 1

15

Illustration of K-Means Clustering – Training Phase

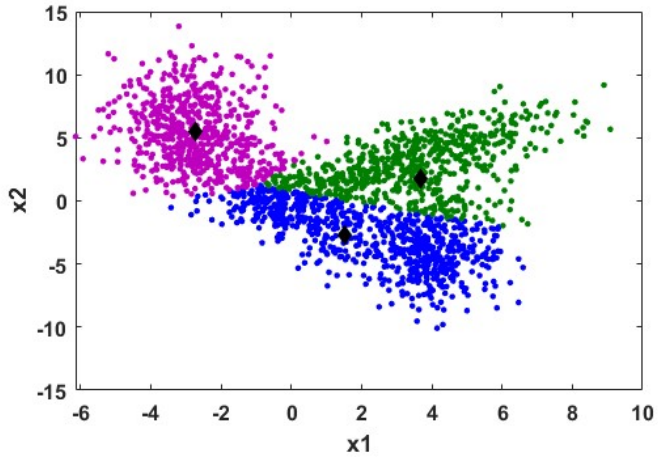


$K = 3$

Iteration 2

16

Illustration of K-Means Clustering – Training Phase

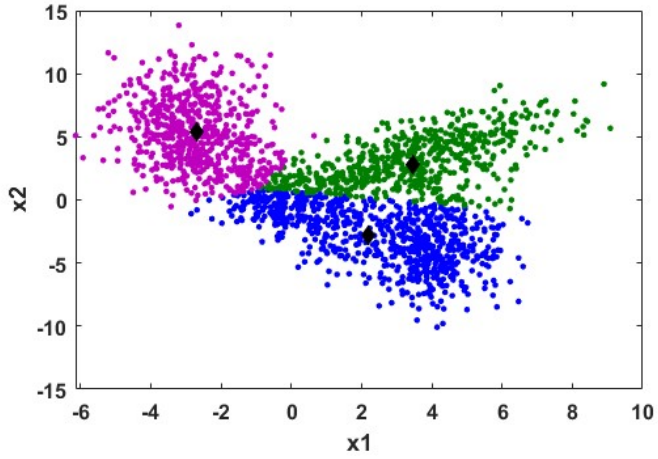


$K = 3$

Iteration 3

17

Illustration of K-Means Clustering – Training Phase

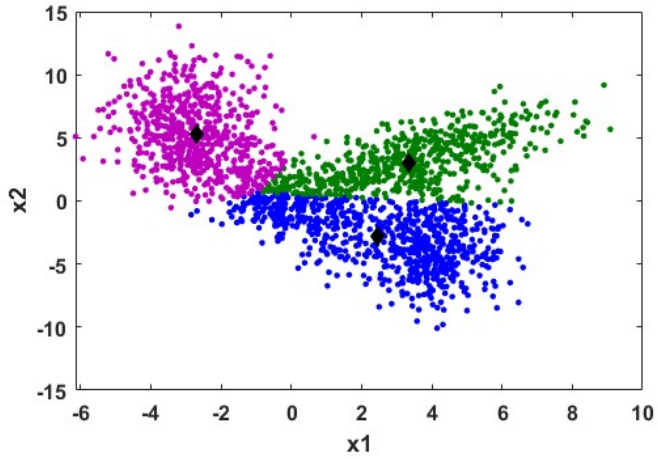


$K = 3$

Iteration 4

18

Illustration of K-Means Clustering – Training Phase

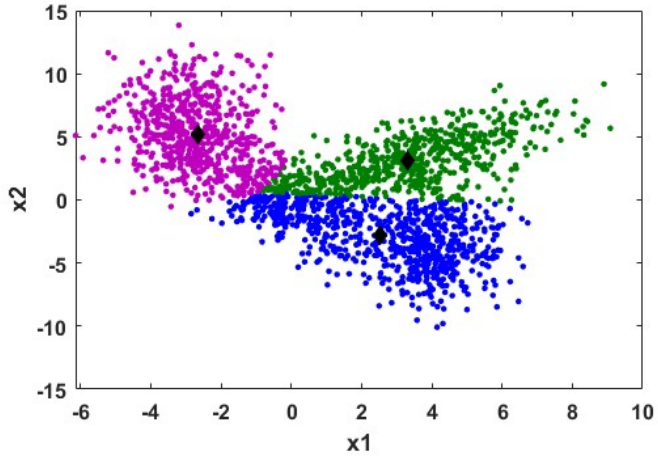


$K = 3$

Iteration 5

19

Illustration of K-Means Clustering – Training Phase

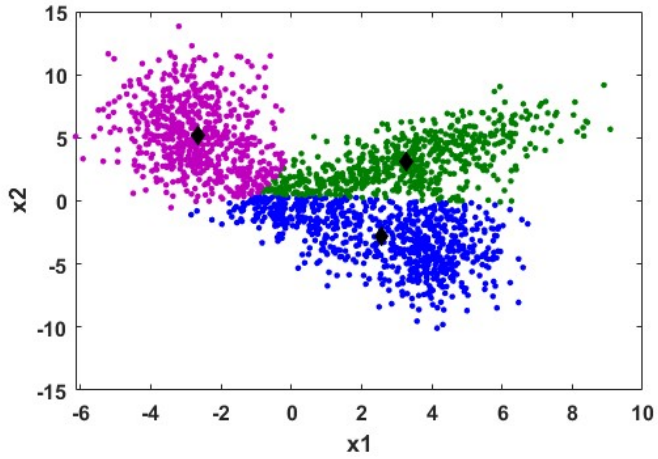


$K = 3$

Iteration 6

20

Illustration of K-Means Clustering – Training Phase

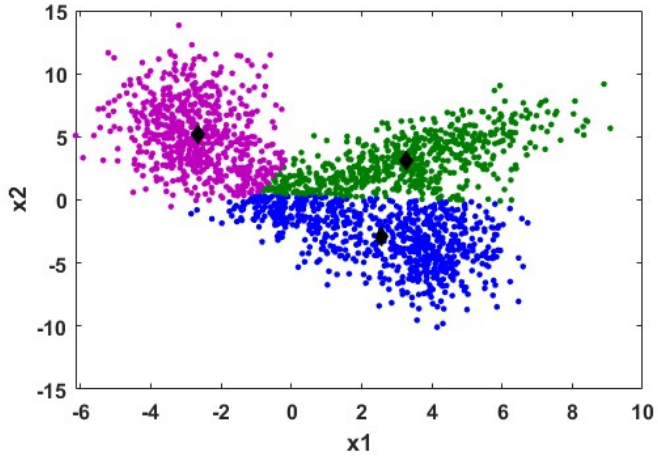


$K = 3$

Iteration 7

21

Illustration of K-Means Clustering – Training Phase

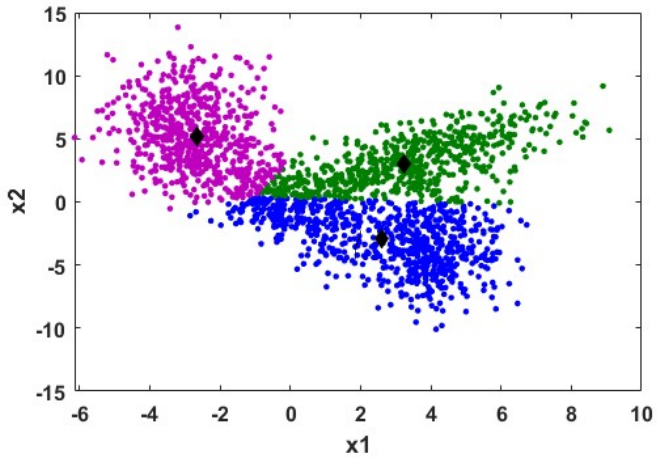


$K = 3$

Iteration 8

22

Illustration of K-Means Clustering – Training Phase



$K = 3$

Iteration 9

- Boundary between the cluster is linear
- Hard clustering: Each example must belong to exactly one group

23

K-Means Clustering Algorithm: Test Phase

- Given a test example $\mathbf{x}$,
  - Assign $\mathbf{x}$ to nearest cluster center k^*

$$k^* = \arg\min_k \left\| \mathbf{x} - \boldsymbol{\mu}_k \right\|^2 \quad \textit{Squared Euclidian distance}$$

  - $\boldsymbol{\mu}_k$ is the center of the k^{th} cluster obtained from training phase

24

Modified K-Means Clustering Algorithm

- Dividing the data into K groups or partitions
- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$ and K
- Target: Partition the set $\mathcal{D}$ into K clusters (disjoint subsets), $\{\mathcal{D}_k\}_{k=1}^{K}$
  - Each of the clusters is associated with centers, $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$
  - ***Better representative for a cluster***
    - ***Cluster center*** *and* ***Spread***
  - Come up with the centers of clusters and variance & covariance (covariance matrix) of clusters
  - Cluster center and covariance matrix of that cluster act as cluster representatives

- Mahalanobis distance with cluster representatives can be used as a measure of dissimilarity to group the similar example

25

Modified K-Means Clustering Algorithm: Training Phase

- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$ and K

1. Initialize the cluster center, $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$ using randomly selected K data points in $\mathcal{D}$
2. Initialize the covariance matrix of each cluster, $\boldsymbol{\Sigma}_k$, $k=1, 2, \ldots, K$ using unit matrix $\quad \boldsymbol{\Sigma}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
3. Assign each data point $\mathbf{x}_n$ to a nearest cluster k^*

$$k^* = \arg\min_k (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)$$ *Squared Mahalanobis distance*

4. Update $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $k=1, 2, \ldots, K$: Re-compute $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ of each cluster after assigning all the data points.

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{\mathcal{D}_k} \mathbf{x}_n}{N_k} \qquad \hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{\mathcal{D}_k} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^\top}{N_k}$$

$\mathcal{D}_k$: Data for cluster k

N_k: Number of examples in cluster k

5. Repeat the steps 3 and 4 until the convergence

26

Modified K-Means Clustering Algorithm: Training Phase

- Convergence criteria:
  - No change in the cluster assignment **OR**
  - The difference between the distortion measure (J) in the successive iteration falls below the threshold
    - Distortion measure (J) : Sum of the squares of the distance of each example to its assigned cluster center

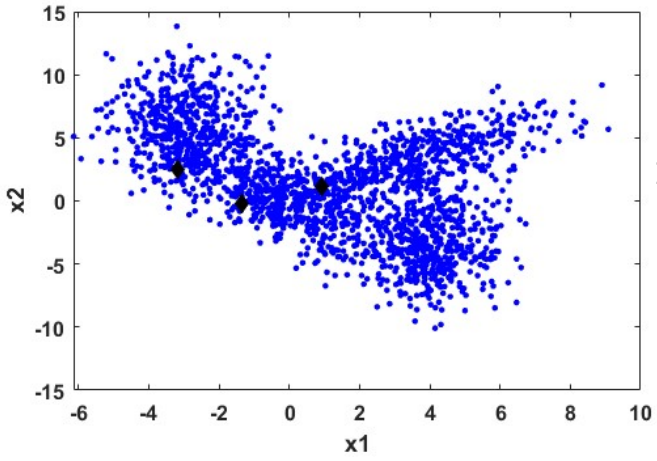
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left[(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \right]$$

z_{nk} is 1 if $\mathbf{x}_n$ belongs to cluster k, otherwise 0

- Hard clustering: Each example must belong to exactly one group

27

Illustration of Modified K-Means Clustering – Training Phase

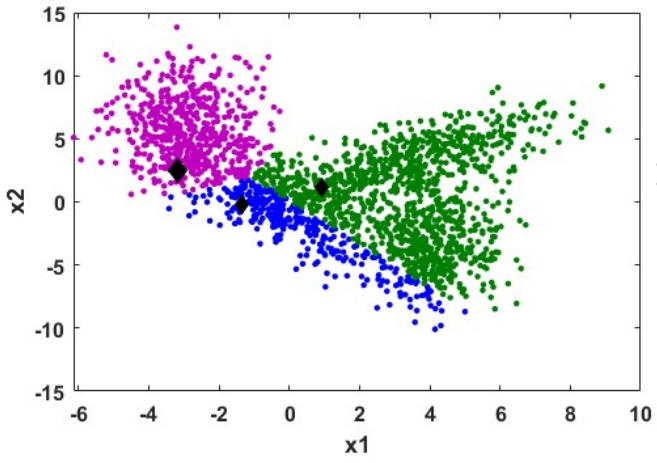


$K = 3$

Iteration 1

28

Illustration of Modified K-Means Clustering – Training Phase

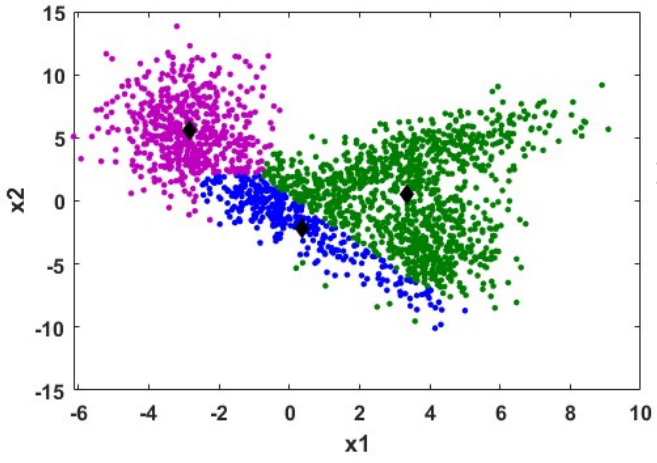


$K = 3$

Iteration 1

29

Illustration of Modified K-Means Clustering – Training Phase

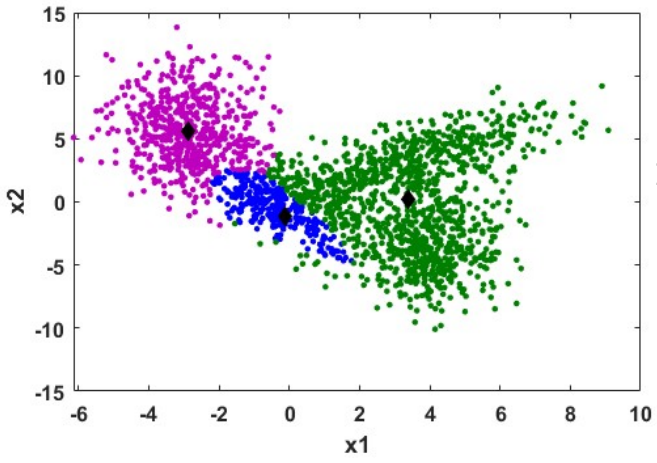


$K = 3$

Iteration 2

30

Illustration of Modified K-Means Clustering – Training Phase

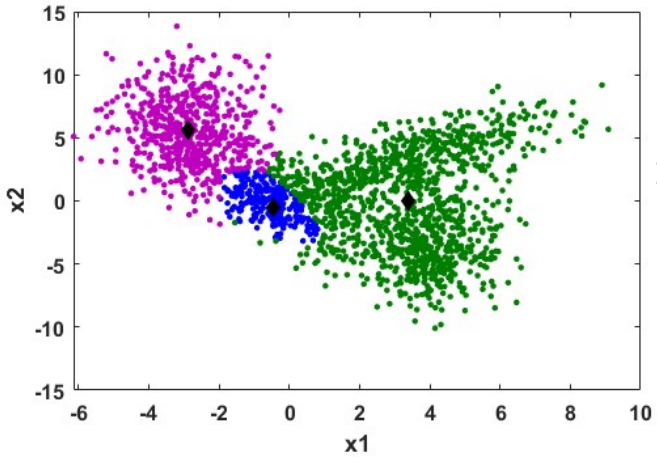


$K = 3$

Iteration 3

31

Illustration of Modified K-Means Clustering – Training Phase

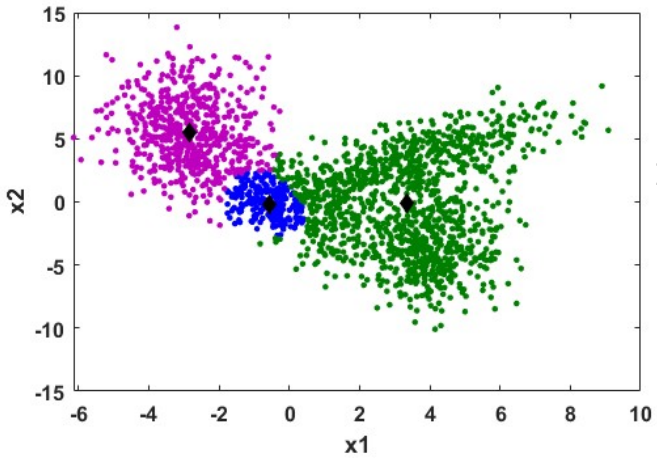


$K = 3$

Iteration 4

32

Illustration of Modified K-Means Clustering – Training Phase

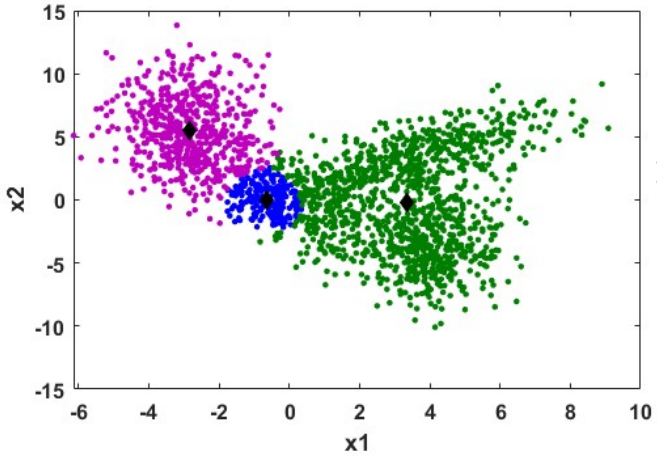


$K = 3$

Iteration 5

33

Illustration of Modified K-Means Clustering – Training Phase

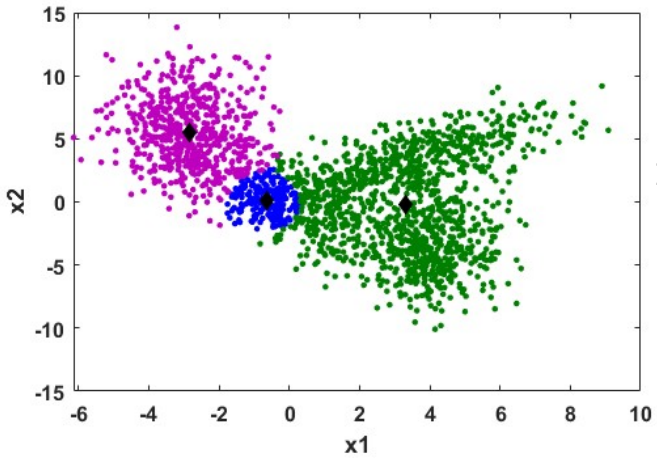


$K = 3$

Iteration 6

34

Illustration of Modified K-Means Clustering – Training Phase

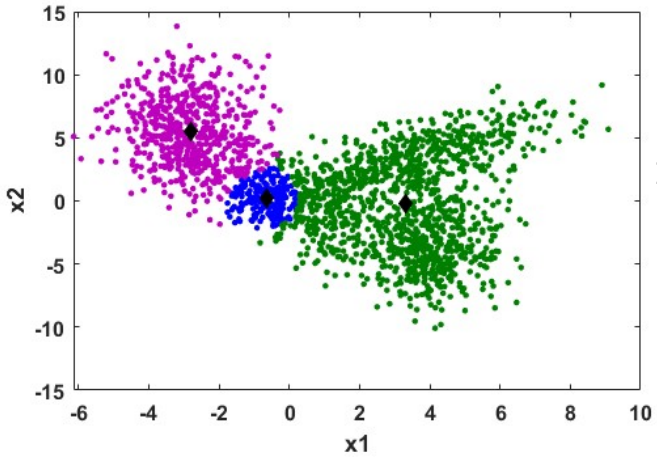


$K = 3$

Iteration 7

35

Illustration of Modified K-Means Clustering – Training Phase

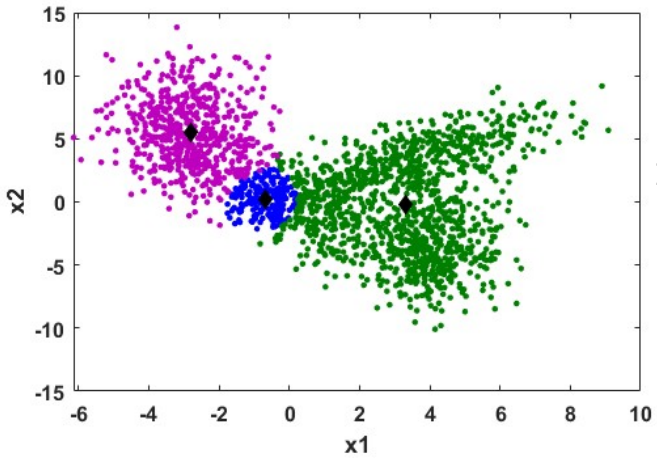


$K = 3$

Iteration 8

36

Illustration of Modified K-Means Clustering – Training Phase



$K = 3$

Iteration 9

- Boundary between the cluster is quadratic
- Hard clustering: Each example must belong to exactly one group

37

Modified K-Means Clustering Algorithm: Test Phase

- Given a test example $\mathbf{x}$,
  - Assign $\mathbf{x}$ to nearest cluster center k^*

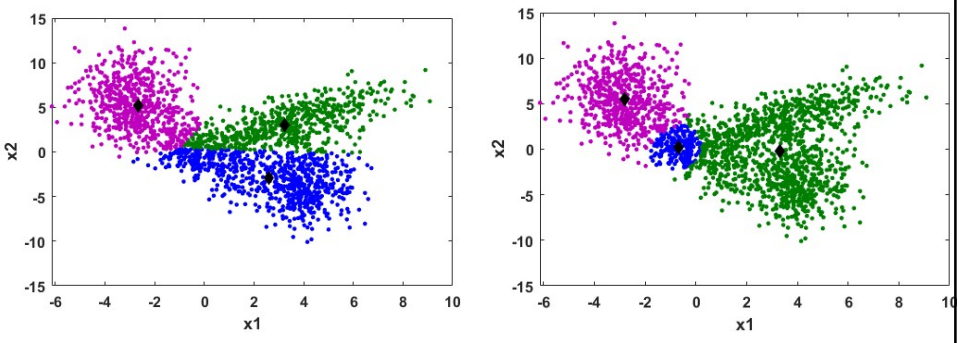
$$k^* = \arg\min_k (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)$$ *Squared Mahalanobis distance*

  - $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the center and covariance matrix of the k^{th} cluster obtained from training phase

38

Illustration of K-Means Clustering

$K = 3$

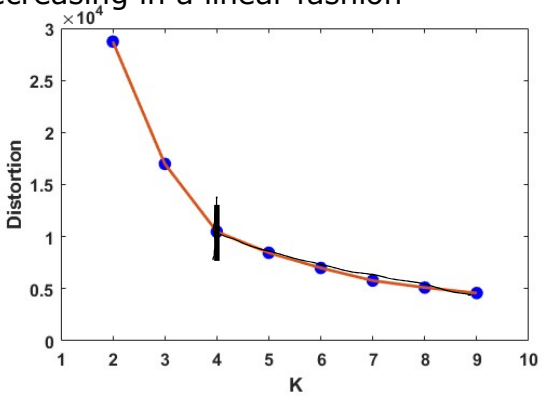


Measure of Dissimilarity: ***Euclidian Distance*** ***Mahalanobis Distance***

39

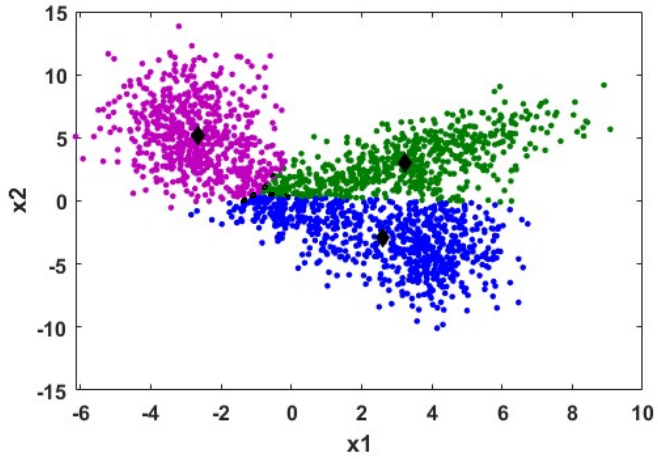
Elbow Method to Choose K

- Determine the distortion measure for different values of K
- Plot the K vs Distortion
- Optimal number of clusters: Select the value of K at the "elbow" i.e. the point after which the distortion start decreasing in a linear fashion



40

K-Means Clustering as Hard Clustering Algorithm

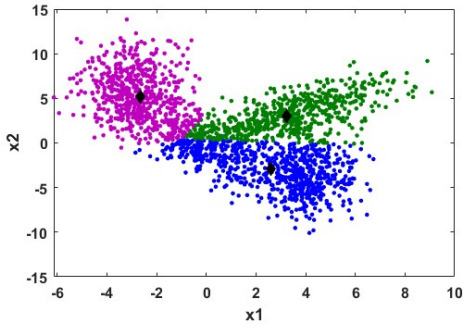


$K = 3$

- Hard clustering: Each example must belong to exactly one group

41

Soft Clustering



- Soft clustering: Each example belong to each group with some probability
  - Fuzzyness at the boundary of the clusters
- Gaussian mixture model (GMM) is one of the soft clustering techniques
- GMM can be seen as similar to K-means clustering
- Each cluster is represented as Gaussian density

42

Unsupervised Machine Learning: Clustering

Gaussian Mixture Model (GMM)

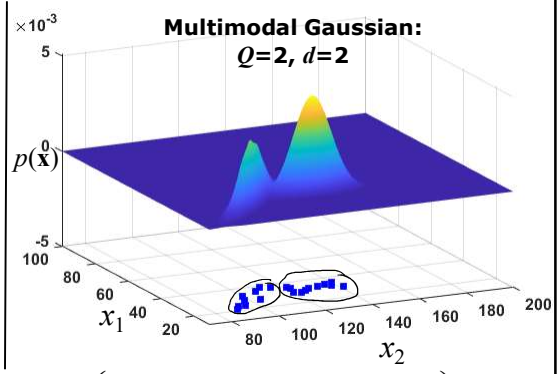
Gaussian Mixture Model (GMM)

- Data is considered to have multiple clusters and each cluster is an Gaussian distribution
- Given: Training data having N samples

$$\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

- GMM is a linear superposition of multiple (K) *Gaussian components*:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathsf{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

44

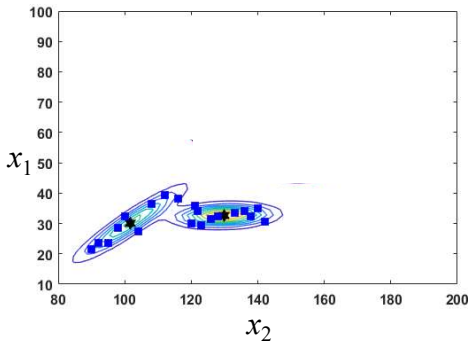
Gaussian Mixture Model (GMM)

- Data is considered to have multiple clusters and each cluster is an Gaussian distribution
- Given: Training data having N samples

$$\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

- GMM is a linear superposition of multiple (K) *Gaussian components*:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^{\mathsf{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right)$$

45

Gaussian Mixture Model (GMM)

- GMM is a linear superposition of multiple Gaussians:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- For a d-dimensional feature vector representation of data, the parameters of GMM are
  - Mixture coefficients, w_k, $k = 1,2,\ldots, K$
    - *Mixture weight or Strength of each clusters (or mixtures or modes)*
    - Property: $\sum_{k=1}^{K} w_k = 1$
  - d-dimensional mean vector, $\boldsymbol{\mu}_k$, $k = 1,2,\ldots, K$
  - dxd size covariance matrices, $\boldsymbol{\Sigma}_k$, $k = 1,2,\ldots, K$
- Training process objective:
  - Partition the data into K groups
  - To estimate the parameters of the each cluster in GMM

46

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters
- Given: Training data having N samples

$$\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

1. Initialize the mean vectors $\boldsymbol{\mu}_k$, covariance matrices $\boldsymbol{\Sigma}_k$ and mixing coefficients w_k, and evaluate the initial value of the log likelihood
   - Initialize the cluster center, $\boldsymbol{\mu}_k$, $k=1, 2, \ldots, K$ using randomly selected K data points in $\mathcal{D}$
   - Initialize the covariance matrix, $\boldsymbol{\Sigma}_k$, $k=1, 2, \ldots, K$ using unit matrix
   - Initialize the mixing coefficient $w_k = \frac{1}{K}$, $k=1, 2, \ldots, K$

$$\sum_{k=1}^{K} w_k = 1$$

Training Process: Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters
- Given: Training data having N samples

$$\mathcal{D}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

1. Initialize the mean vectors $\boldsymbol{\mu}_k$, covariance matrices $\boldsymbol{\Sigma}_k$ and mixing coefficients w_k, and evaluate the initial value of the log likelihood

$$p(\mathcal{D}\,|\,\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}_n\,|\,\boldsymbol{\theta}) \text{ where } \boldsymbol{\theta} = [w_1 \ldots w_k \ldots w_K, \boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_k \ldots \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1 \ldots \boldsymbol{\Sigma}_k \ldots \boldsymbol{\Sigma}_K]^{\mathsf{T}}$$

$$\mathcal{L}(\boldsymbol{\theta}) = \ln p(\mathcal{D}\,|\,\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n\,|\,\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}_n\,|\,\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

2. **E-step**: Evaluate the responsibilities $\gamma_k(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster

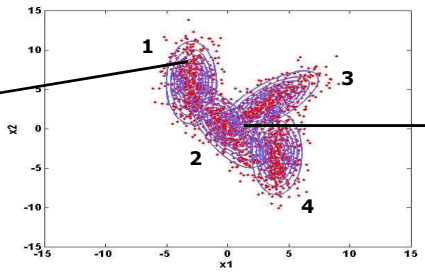
Training Process:
EM Method – Responsibility Term

- A quantity that plays an important role is the responsibility term, $\gamma_k(\mathbf{x})$
- It is given by

$$\gamma_k(\mathbf{x}) = \frac{w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- w_k : mixture coefficient or prior probability of cluster k,
- $\gamma_k(\mathbf{x})$ gives the posterior probability of the cluster k for the observation $\mathbf{x}$

$\gamma_1(\mathbf{x}_n) = 0.99$
$\gamma_2(\mathbf{x}_n) = 0.01$
$\gamma_3(\mathbf{x}_n) = 0.00$
$\gamma_4(\mathbf{x}_n) = 0.00$



$\gamma_1(\mathbf{x}_m) = 0.08$
$\gamma_2(\mathbf{x}_m) = 0.42$
$\gamma_3(\mathbf{x}_m) = 0.34$
$\gamma_4(\mathbf{x}_m) = 0.16$

49

Training Process:
Expectation-Maximization (EM) for GMMs

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters

1. Initialize the mean vectors $\boldsymbol{\mu}_k$, covariance matrices $\boldsymbol{\Sigma}_k$ and mixing coefficients w_k, and evaluate the initial value of the log likelihood
2. **E-step**: Evaluate the responsibilities $\gamma_k(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster
3. **M-step**: Re-estimate the parameters $\boldsymbol{\mu}_k^{new}$, $\boldsymbol{\Sigma}_k^{new}$ and w_k^{new} using the current responsibilities

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)\, \mathbf{x}_n \qquad \boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)\, (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}}$$

$$w_k^{new} = \frac{N_k}{N} \qquad N_k = \sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)$$

- N_k: Effective number of points assigned to the cluster k

50

Training Process: Expectation-Maximization (EM) for GMMs

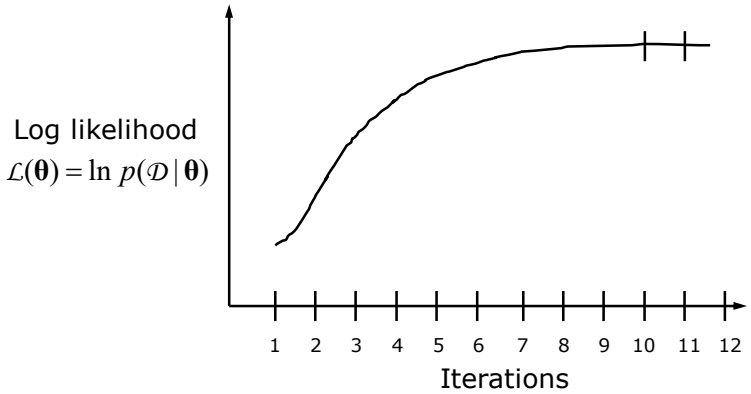
- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters

1. Initialize the mean vectors $\boldsymbol{\mu}_k$, covariance matrices $\boldsymbol{\Sigma}_k$ and mixing coefficients w_k, and evaluate the initial value of the log likelihood
2. **E-step**: Evaluate the responsibilities $\gamma_k(\mathbf{x})$ using the current parameter values – Assign the data points to each cluster
3. **M-step**: Re-estimate the parameters $\boldsymbol{\mu}_k^{new}$, $\boldsymbol{\Sigma}_k^{new}$ and w_k^{new} using the current responsibilities
4. Evaluate the log likelihood and check for convergence of the log likelihood
   - If the convergence criterion is not satisfied return to step 2

51

Training Process: Expectation-Maximization (EM) for GMMs

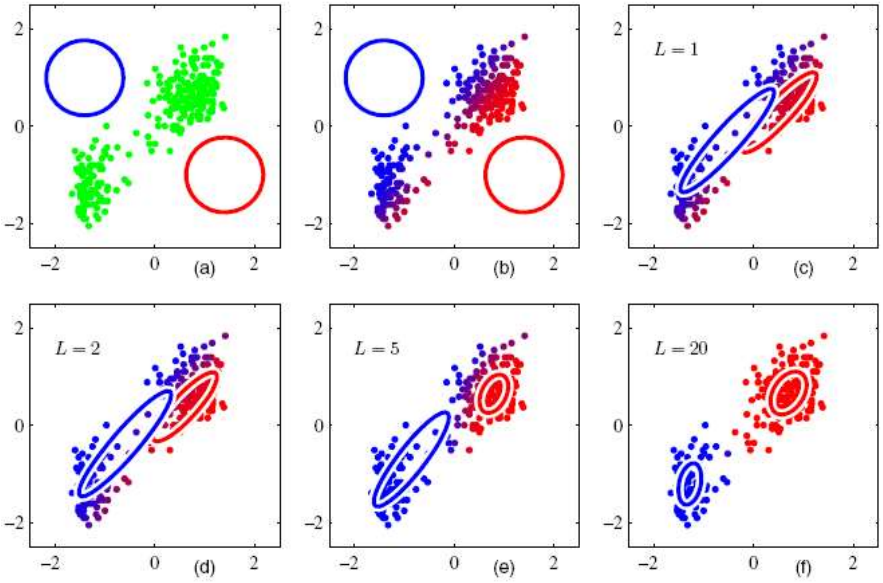
- Convergence criterion: Difference between log likelihoods of successive iterations fall below a threshold (E.g. 10^{-3})



$$\mathcal{L}(\boldsymbol{\theta}) = \ln p(\mathcal{D}\,|\,\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}\,|\,\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)$$

52

Training Process:
Illustration of Parameter Estimation



C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

53

GMM as a Soft Clustering Algorithm:
Test Phase

- Given a test example $\mathbf{x}$,
  - Compute $\gamma_k(\mathbf{x})$, posterior probability of the cluster k for the observation $\mathbf{x}$

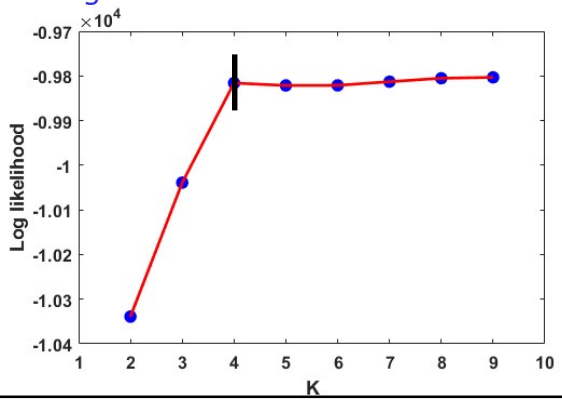
$$\gamma_k(\mathbf{x}) = \frac{w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

    - $\gamma_k(\mathbf{x})$ gives the probability that $\mathbf{x}$ belong to k^{th} cluster
  - Since GMM is a soft clustering technique, $\mathbf{x}$ is assigned to each cluster with probability $\gamma_k(\mathbf{x})$
  - If at all it is to assign to one cluster, then assign $\mathbf{x}$ to cluster with highest probability $\gamma_k(\mathbf{x})$

54

Elbow Method to Choose K in GMM

- Determine the total data log likelihood for different values of K
- Plot the K vs total data log likelihood
- Optimal number of clusters: Select the value of K at the "elbow" i.e. the point after which the log likelihood start increasing in a linear fashion



55

Unsupervised Machine Learning: Clustering

Partitioning Method

Summary

- Partitioning methods: Partition the collection of examples into K clusters based on the distance between examples
- Centroid-based technique:
  - Cluster similarity is measured in regard to the sample mean of the examples within a cluster
  - K-means algorithm: Hard clustering method
  - Gaussian mixture model (GMM): Soft clustering method

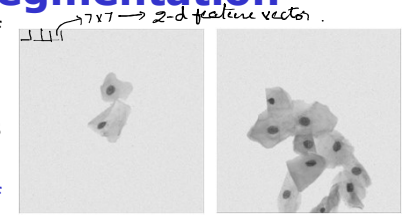
57

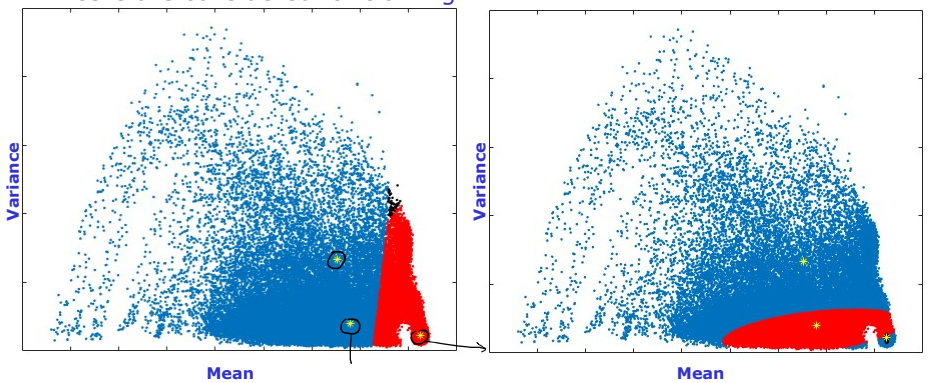
Application of Clustering: Segmentation using Clustering

58

Application of Clustering: Cell and Nucleus Segmentation

- An image is divided into patches of size 7 x 7
- From each patch, mean and variance of pixel values is considered as features
- 100 images with different numbers of cells are considered for training



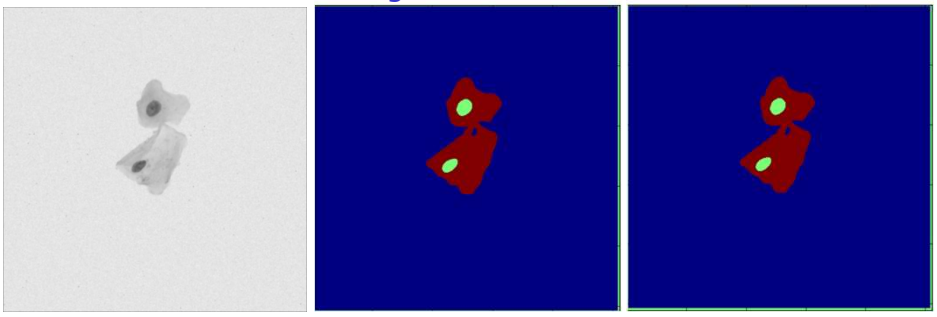


K-Means **GMM**

59

Application of Clustering: Cell and Nucleus Segmentation

- An image is divided into patches of size 7 x 7
- From each patch, mean and variance of pixel values is considered as features
- 100 images with different numbers of cells are considered for training

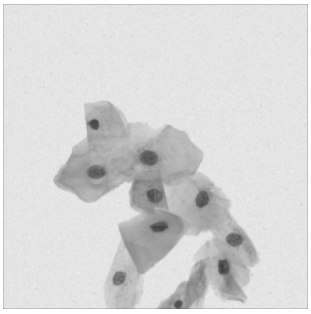


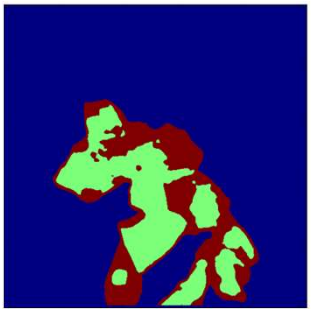
K-Means **GMM**

60

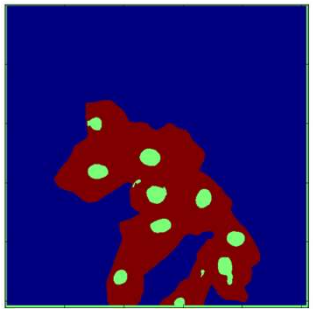
Application of Clustering: Cell and Nucleus Segmentation

- An image is divided into patches of size 7 x 7
- From each patch, mean and variance of pixel values is considered as features
- 100 images with different numbers of cells are considered for training





K-Means



GMM

61

Summary

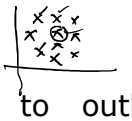
- Partitioning methods: Partition the collection of examples into K clusters based on the distance between examples
- Centroid-based technique:
  - Cluster similarity is measured in regard to the sample mean of the examples within a cluster
  - K-means algorithm: Hard clustering method
  - Gaussian mixture model (GMM): Soft clustering method
- Representative object-based technique:

62

Unsupervised Machine Learning: Clustering

K-Medoid Clustering Algorithms

K-Medoid Clustering Algorithms



- Related to K-means clustering
- The K-means algorithm is sensitive to outliers because an example with extremely large value may substantially distort the distribution of data
- Solution: One of the data points is chosen as representative of cluster, instead of mean value of the cluster
- Its replaces the mean vector of cluster with medoid
  - (Medoid is in d-dimension and median in 1-dimension)
- Partitioning around medoids
- A medoid of a finite dataset: The data point from the set, whose average dissimilarity (distance) to all the points is minimal
  - The most centrally located point in the set

64

K-Medoid Clustering Algorithm

- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$ and K

1. Initialize the medoid, $\hat{\mathbf{x}}_k$, k=1, 2, …, K using randomly selected K data points in $\mathcal{D}$
2. Assign each data point $\mathbf{x}_n$ to the closest medoid

$$k^* = \arg\min_k \left\|\mathbf{x}_n - \hat{\mathbf{x}}_k\right\|^2$$ *Squared Euclidian distance*

3. Update medoids $\hat{\mathbf{x}}_k$, k=1, 2, …, K
   - For each data point $\mathbf{x}_n$ assigned to a cluster k compute the average dissimilarity (distance) of $\mathbf{x}_n$ to all the data points assigned to cluster k

$$\text{Average dissimilarity for } \mathbf{x}_n = \frac{\sum_{\mathbf{x}_m \in \mathcal{D}_k} \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2}{N_k}$$ N_k: Number of examples in cluster k

   - Select the example with minimum average dissimilarity as *medoid*
4. Repeat the steps 2 and 3 until the convergence

65

K-Medoid Clustering Algorithm

- Convergence criteria:
  - No change in the cluster assignment **OR**
  - The difference between the distortion measure (absolute-error) (J) in the successive iteration falls below the threshold
    - Distortion measure (J) : Sum of the squares of the distance of each example to its corresponding reference point (medoid)

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left\|\mathbf{x}_n - \hat{\mathbf{x}}_k\right\|^2$$

z_{nk} is 1 if $\mathbf{x}_n$ belongs to cluster k, otherwise 0

- Optimal number of clusters (k) can obtained using elbow method

66

Unsupervised Machine Learning: Clustering

Evaluation of Clustering: Purity Score

Evaluation of Clustering: Purity Score

- Assumption:
  - The class index for each example is given
  - One cluster is responsible for one class i.e. one of the class is most common in a cluster
- Purity score: Purity is a measure of the extent to which clusters contain a single class
  - For each cluster, count the number of data points from the most common class
  - Take the sum of the number of data points from the most common class over all clusters and divide by the total number of data points
- Let M be the number of classes, $C_1, C_2, \ldots, C_m, \ldots, C_M$
- Let K be the number of clusters, $k = 1,2,\ldots, K$
- Let N be the number of data points

68

Evaluation of Clustering: Purity Score

- For each cluster k,
  - Count the number of data points from each class
  - Consider the number of data points of most common class

$$\max_m |N_k \cap C_m|$$

$|N_k \cap C_m|$ is the number of data points in k^{th} cluster belonging to class m

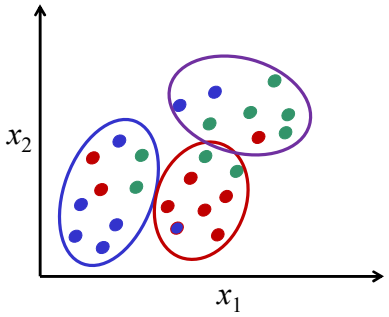
- Take the sum over all clusters, k
- Divide by the total number of data points (N)

$$\text{Purity Score} = \frac{1}{N}\sum_{k=1}^{K}\max_m |N_k \cap C_m|$$

69

Illustration of Computing Purity Score

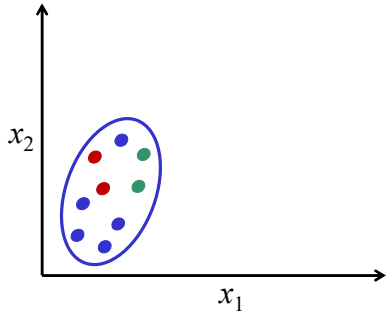
- Number of data points, N = 25
- number of classes, M = 3
- number of clusters, K = 3



70

Illustration of Computing Purity Score

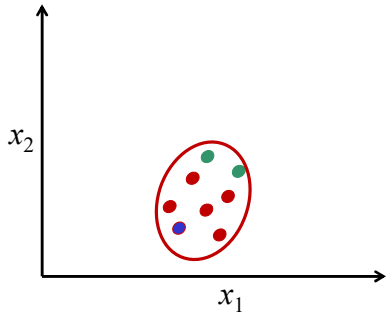
- Number of data points, N = 25
- number of classes, M = 3
- number of clusters, K = 3
- *Cluster 1*:
  - Number of examples of Blue Class are more, i.e. **5**



71

Illustration of Computing Purity Score

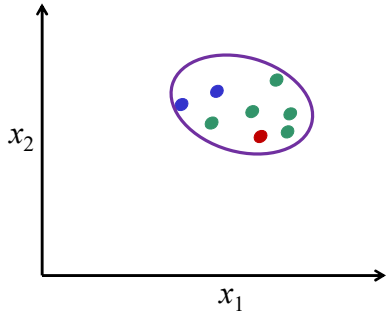
- Number of data points, N = 25
- number of classes, M = 3
- number of clusters, K = 3
- *Cluster 1*:
  - Number of examples of Blue Class are more, i.e. **5**
- *Cluster 2*:
  - Number of examples of Red Class are more, i.e. **5**



72

Illustration of Computing Purity Score

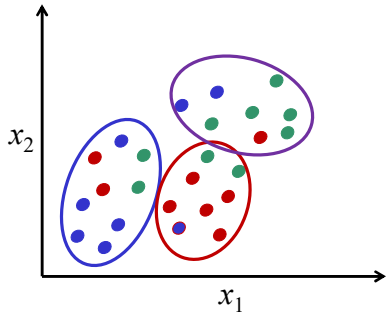
- Number of data points, N = 25
- number of classes, M = 3
- number of clusters, K = 3
- *Cluster 1*:
  - Number of examples of Blue Class are more, i.e. **5**
- *Cluster 2*:
  - Number of examples of Red Class are more, i.e. **5**
- *Cluster 3*:
  - Number of examples of Green Class are more, i.e. **5**



73

Illustration of Computing Purity Score

- Number of data points, N = 25
- number of classes, M = 3
- number of clusters, K = 3
- *Cluster 1*:
  - Number of examples of Blue Class are more, i.e. **5**
- *Cluster 2*:
  - Number of examples of Red Class are more, i.e. **5**
- *Cluster 3*:
  - Number of examples of Green Class are more, i.e. **5**
- **Purity score: (5+5+5)/25 = 0.60**



74

Summary

- Partitioning methods: Partition the collection of examples into K clusters based on the distance between examples
- Centroid-based technique:
  - Cluster similarity is measured in regard to the sample mean of the examples within a cluster
  - K-means algorithm: Hard clustering method
  - Gaussian mixture model (GMM): Soft clustering method
- Representative object-based technique:
  - Actual example is considered to represent the cluster
  - One representative example per cluster
  - K-medoid algorithm
- Cluster purity score as evaluation metric for clustering

75

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

76

Unsupervised Machine Learning: Clustering

Hierarchical Clustering

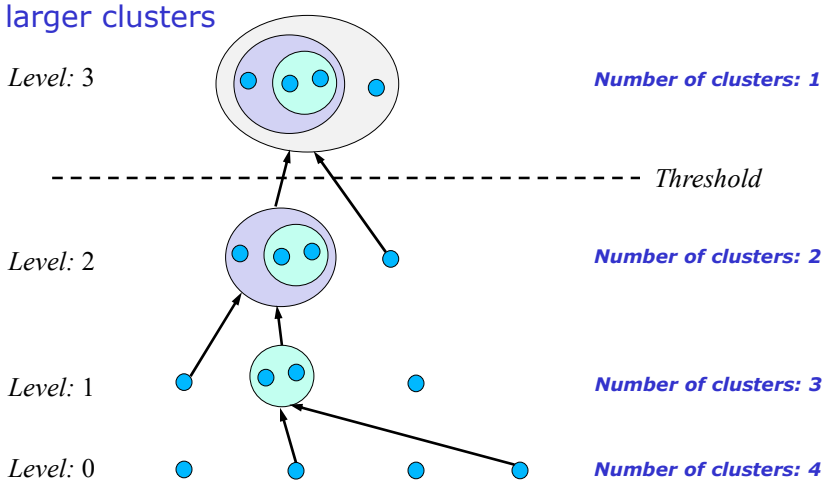
Hierarchical Clustering Algorithms

- These methods create a hierarchical decomposition of the collection of examples
- Produce nested sequence of data partitions
- These sequence can be depicted using a tree structure
- Hierarchical clustering method works by grouping data points into a tree of clusters
- Hierarchical algorithms are either agglomerative or divisive
  - This classification of hierarchical clustering is depending on whether the hierarchical decomposition is formed in a
    - Bottom-up (merging) OR
    - Top-down (splitting) fashion
- Need not have to specify the number of clusters

2

Agglomerative Hierarchical Clustering

- Bottom-up approach
- This strategy starts by placing each example in its own cluster (atomic clusters or singleton clusters) and then merges these atomic clusters into larger and larger clusters



3

Agglomerative Hierarchical Clustering

- Bottom-up approach
- This strategy starts by placing each example in its own cluster (atomic clusters) and then merges these atomic clusters into larger and larger clusters
- Starts with N clusters where each example is a cluster
- At each successive step (level), the most similar pair of clusters are merged
  - The measure of closeness (intercluster similarity) is considered to decide which two clusters are merged
  - At each level, number of clusters is reduces by one
- The process continues till all the examples are in a single cluster or until certain termination conditions are satisfied
  - Termination condition could be
    - Number of clusters
    - Intercluster similarity between each pair of cluster is within a certain threshold

4

Agglomerative Hierarchical Clustering

- ***Once two examples are placed in the same cluster at a level, they remain in same cluster at all subsequent levels***
- Example: AGglomerative NESting (AGNES)
- Most hierarchical clustering methods belong to this category
- They differ only in their definition of intercluster similarity
- Intercluster similarity is to identifying two closest cluster for merging
- When there is one example in a cluster, two closest clusters are found by computing minimum Euclidian distance between two clusters
- However, there is no unique way to find the two closest clusters when there are more than one data points in each clusters

5

Agglomerative Hierarchical Clustering

- Different intercluster similarities to find similarity between the clusters having more than one examples:
  1. Minimum distance between any two examples from two clusters C_i and C_j

$$d_{\min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

  - Select a pair of clusters for merging whose minimum distance between any two examples is minimum of all the pair of clusters

  2. Distance between the centers of two clusters C_i and C_j

$$d_{mean}(C_i, C_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$$

  - Where $\boldsymbol{\mu}_i$ is the center of C_i and $\boldsymbol{\mu}_j$ is the center of C_j
  - Select a pair of clusters for merging whose distance between the centers is minimum of all the pair of clusters

6

Agglomerative Hierarchical Clustering

- Different intercluster similarities to find similarity between the clusters having more than one examples:
  3. Average distance of all the points in one cluster (C_i) to all the points in another cluster (C_j)

$$d_{avg}(C_i, C_j) = \frac{1}{N_i N_j} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

  - Where N_i and N_j are the number of examples in clusters C_i and C_j respectively
  - Select a pair of clusters for merging whose average distance is minimum than that of all the pair of clusters

7

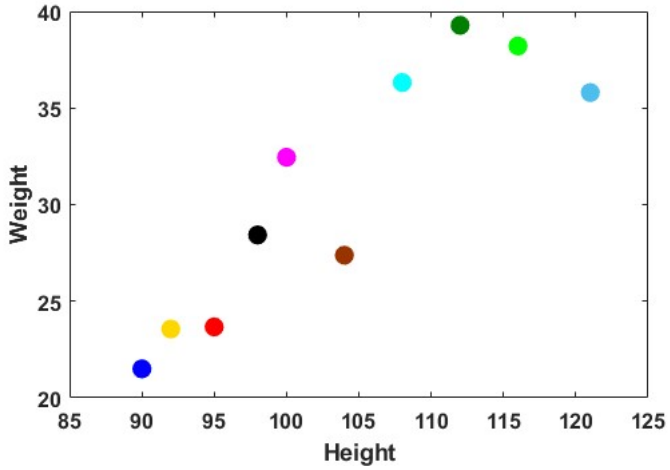
Agglomerative Hierarchical Clustering

- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$
- Target: Partition the data
- Step1: N clusters where each example is a cluster
- Step2: Compute intercluster similarity between each pair of clusters
  - Euclidian distance is considered as intercluster distance if the clusters are singleton clusters
  - The intercluster similarity defined in the previous 2 slides will be considered when the clusters are not singleton clusters
- Step3: Choose a pair of clusters that are most similar (minimum intercluster distance) and merge them
- Step4: Repeat Step2 and Step3 until all the examples are in a single cluster or until certain termination conditions are satisfied

8

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

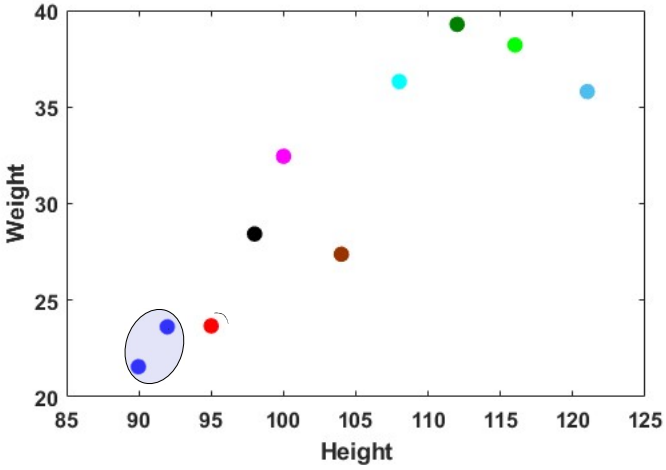


Level: 0 *Number of clusters:* 10

9

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

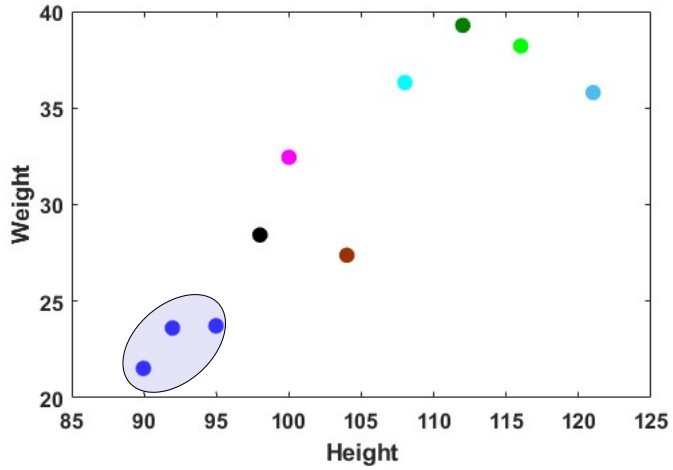


Level: 1 *Number of clusters:* 9

10

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

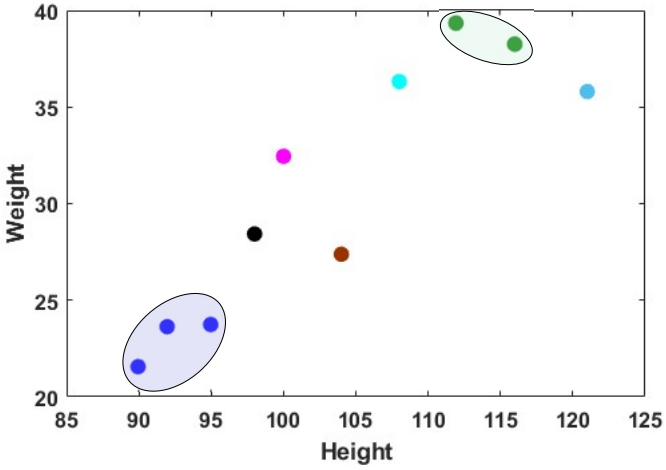


Level: 2 *Number of clusters:* 8

11

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

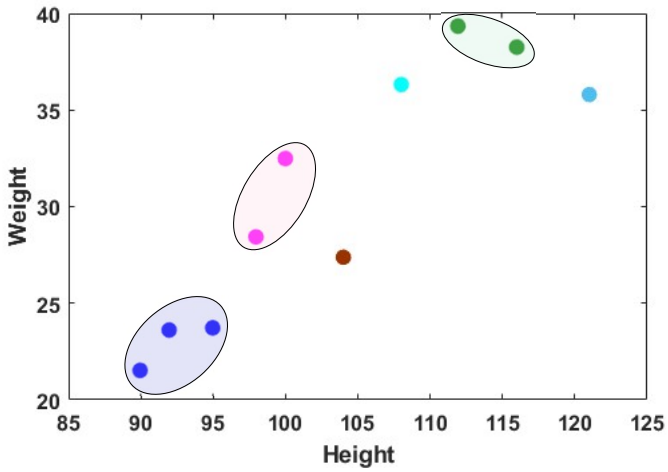


Level: 3 *Number of clusters:* 7

12

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

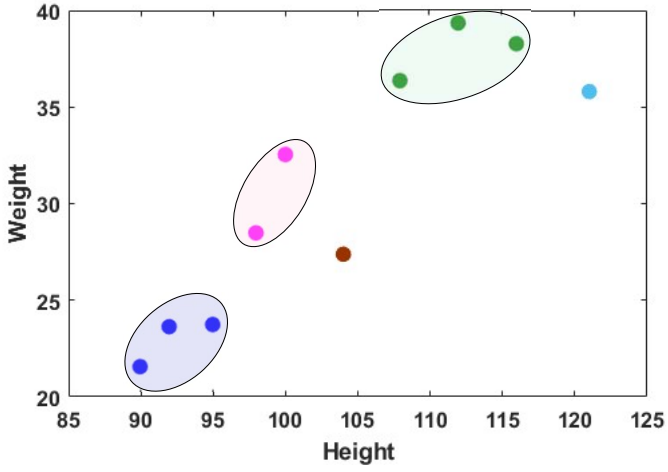


Level: 4 *Number of clusters:* 6

13

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

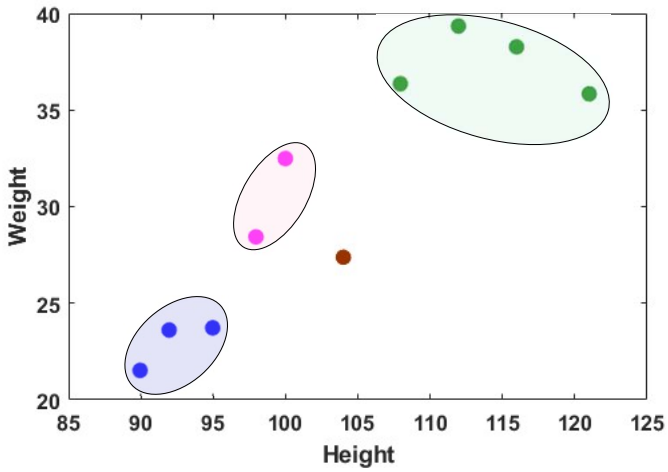


Level: 5 *Number of clusters:* 5

14

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

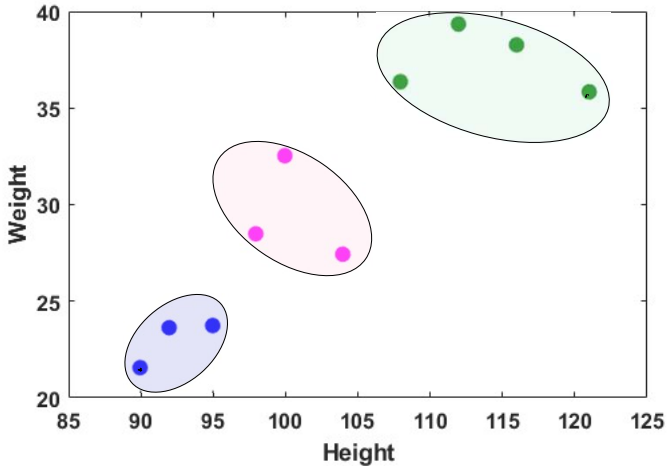


Level: 6 *Number of clusters:* 4

15

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
  - Single example in clusters: Euclidian distance
  - More than one examples in cluster: Distance between the centres of two clusters

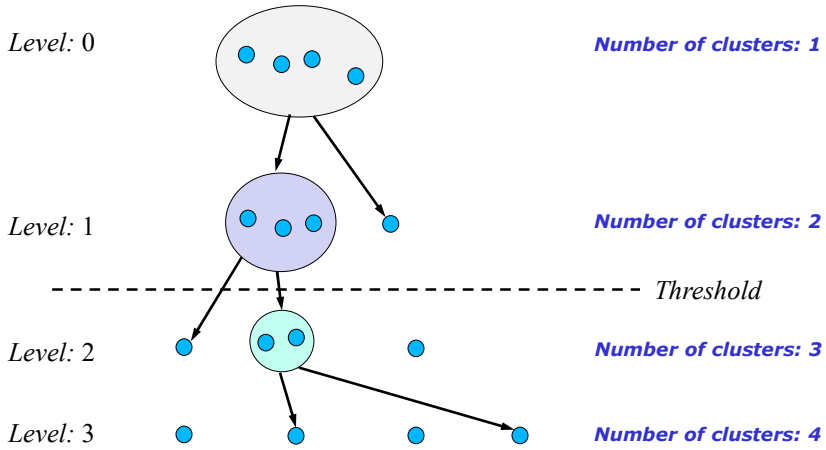


Level: 7 *Number of clusters:* 3

16

Divisive Hierarchical Clustering

- Top-down approach
- Starts with single cluster having all the examples
- It subdivides the cluster into smaller and smaller clusters in the successive step



17

Divisive Hierarchical Clustering

- Top-down approach
- Starts with single cluster having all the examples
- It subdivides the cluster into smaller and smaller clusters in the successive step
- At each successive step, a compactness measure is used to choose which cluster to split
  - Compactness measure: Average value of distance between the data points of a cluster
  - Compactness measure (CM_i) of a cluster C_i :

$$CM_i = \frac{1}{N_i^2} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_i} \|\mathbf{x} - \mathbf{x}'\|$$

    - Where N_i is the number of examples in cluster C_i
  - Choose the cluster with larger value of compactness measure to split

18

Divisive Hierarchical Clustering

- Split the cluster around the farthest two examples on that cluster
- To split a cluster, find a pair of examples having maximum Euclidian distance and split around these two examples (keeping them as centroids)
- At each level, number of clusters is increases by one
- The process continues until each example forms a cluster (atomic or singleton cluster) or until it satisfies certain termination condition
  - Termination condition could be
    - Number of clusters
    - Compactness measure of each cluster is within a certain threshold
- ***Once two examples are placed in two different clusters at a level, they remain in different clusters at all subsequent levels***
- Example: DIvisive ANAlysis (DIANA)

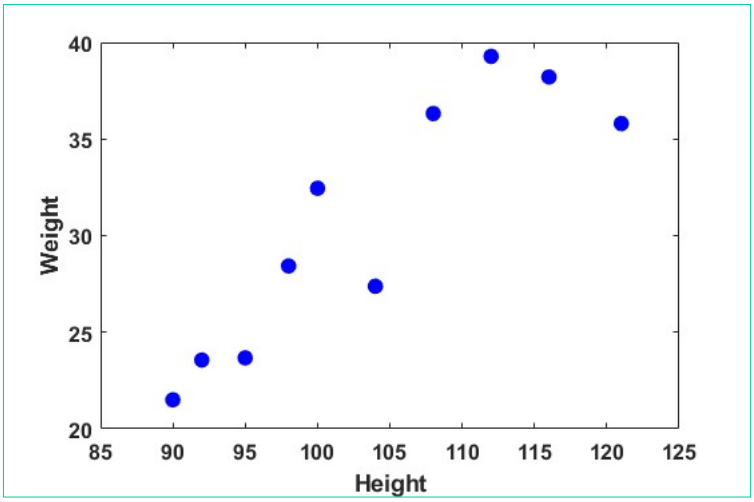
19

Divisive Hierarchical Clustering

- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$
- Target: Partition the data
- Step1: Single cluster having all the examples
- Step2: Find a pair of examples with in a cluster having maximum Euclidian distance
  - These examples act as centroid
- Step3: Split into two clusters by assigning each data point to one of these two examples (centroids) using Euclidian distance
- Step4: Compute compactness measure for each cluster
- Step5: Choose the cluster with larger value of compactness measure to split
- Step6: Repeat Step2 to Step5 until each example forms a cluster (atomic or singleton cluster) or until it satisfies certain termination condition

20

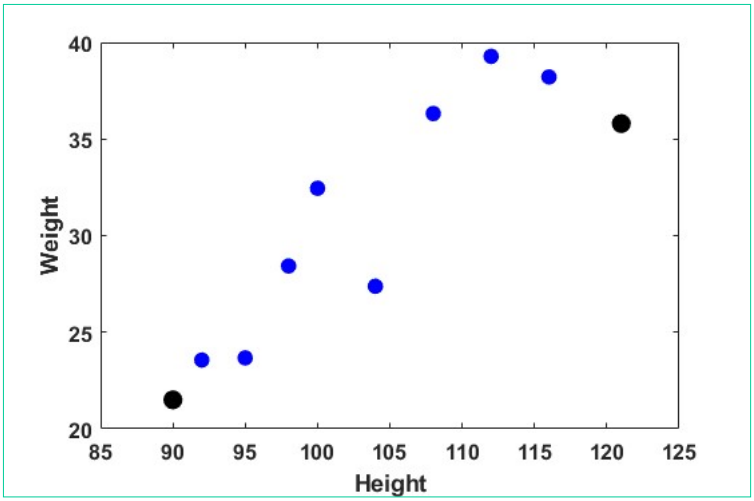
Divisive Hierarchical Clustering



Level: 0 *Number of clusters:* 1

21

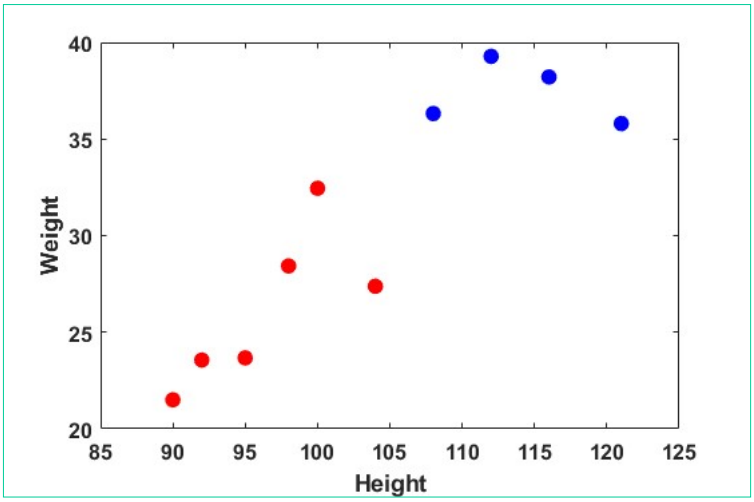
Divisive Hierarchical Clustering



Level: 0 *Number of clusters:* 1

22

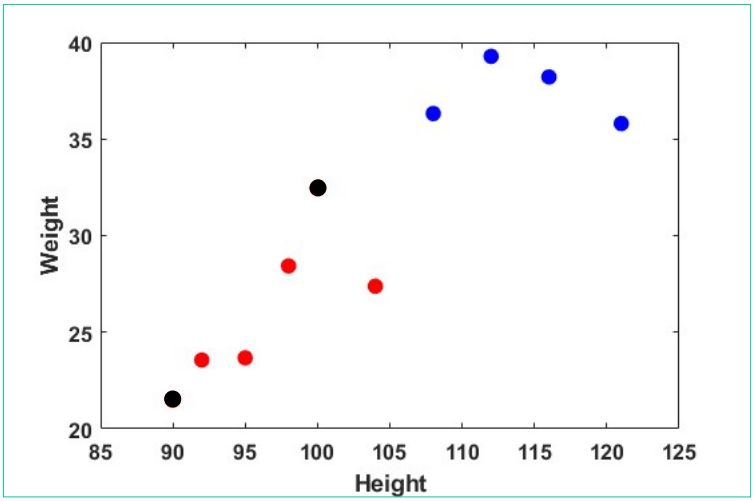
Divisive Hierarchical Clustering



Level: 1 *Number of clusters:* 2

23

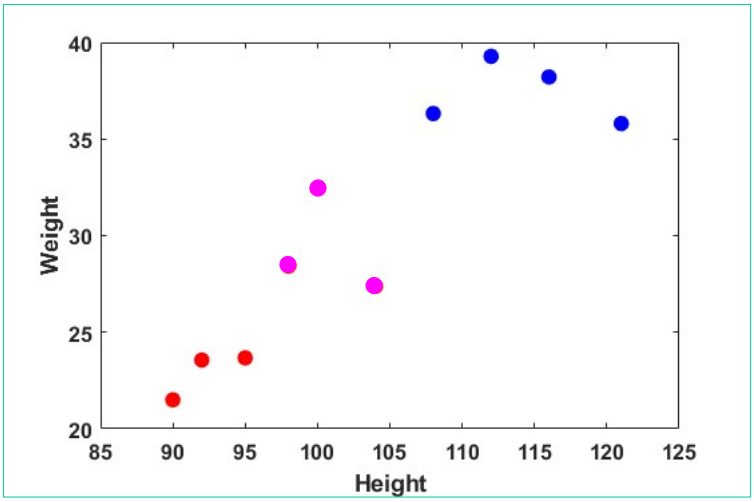
Divisive Hierarchical Clustering



Level: 1 *Number of clusters:* 2

24

Divisive Hierarchical Clustering



Level: 2 *Number of clusters:* 3

25

Summary: Hierarchical Clustering

- Hierarchical clustering methods create a hierarchical decomposition of the collection of examples
- Hierarchical clustering method works by grouping data points into a tree of clusters
- Hierarchical algorithms are either agglomerative or divisive depending on whether the hierarchical decomposition is formed in a
  - Bottom-up (merging) OR
  - Top-down (splitting) fashion
- Agglomerative clustering
  - Example: AGglomerative NESting (AGNES)
- Divisive clustering
  - Example: DIvisive ANAlysis (DIANA)

26

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

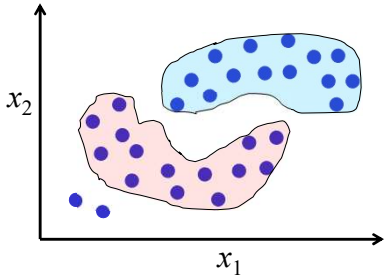
27

Unsupervised Machine Learning: Clustering

Density-Based Clustering

Density-Based Clustering

- These methods cluster collection of examples based on the notion of density (concentration of data points)
- These methods regard clusters as dense regions of examples in the data space that are separated by regions of low density (i.e. noise)
- They discover clusters with arbitrary shape



2

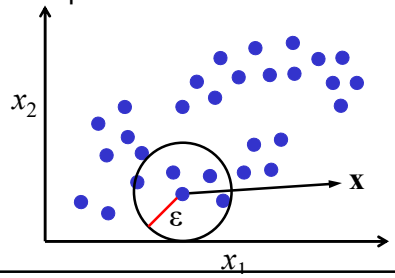
Density-Based Clustering

- These methods cluster collection of examples based on the notion of density
- These methods regard clusters as dense regions of examples in the data space that are separated by regions of low density (i.e. noise)
- They discover clusters with arbitrary shape
- They automatically identifies the number of clusters
- General idea: To continue growing the given cluster as long as density (concertation or number of examples) in the neighbourhood exceeds some threshold
- Example: Density-based Spatial Clustering of Applications with Noise (DBSCAN)
  - It grows the clusters according to a density-based connectivity analysis

3

Density-based Spatial Clustering of Applications with Noise (DBSCAN)

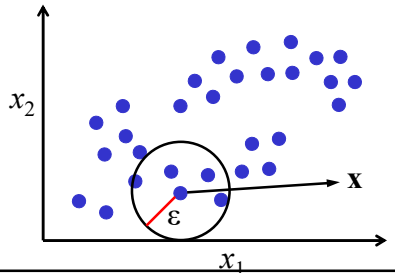
- DBSCAN is a density-based clustering included with noise
- It grows the regions with sufficiently high density (neighbors) into clusters with arbitrary shape
- It defines a cluster as a maximal set of density-connected points
- DBSCAN has 5 important definitions (components):
  1. Epsilon (ε): It is a value of radius of boundary from every example



4

Density-based Special Clustering of Applications with Noise (DBSCAN)

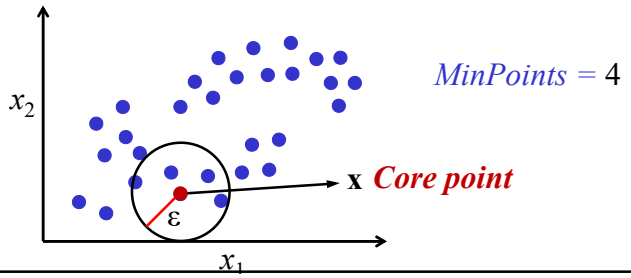
- DBSCAN has 5 important components:
  1. Epsilon (ε): It is a value of radius of boundary from every example
  2. *MinPoints*: Minimum number of examples present inside the boundary with radius of ε from an example **x**
     - These examples with in a boundary are neighbors to **x** and called as ε-neighborhood of an example, **x**



5

Density-based Special Clustering of Applications with Noise (DBSCAN)

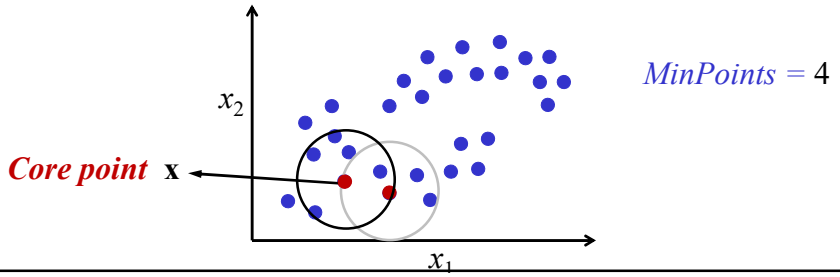
- DBSCAN has 5 important definitions (components):
  1. Epsilon (ε): It is a value of radius of boundary from every example
  2. *MinPoints*: Minimum number of examples present inside the boundary with radius of ε from an example **x**
  3. Core point: If there are atleast *MinPoints* number of examples are with in ε–radius from **x**, then **x** is called as core point



6

Density-based Special Clustering of Applications with Noise (DBSCAN)

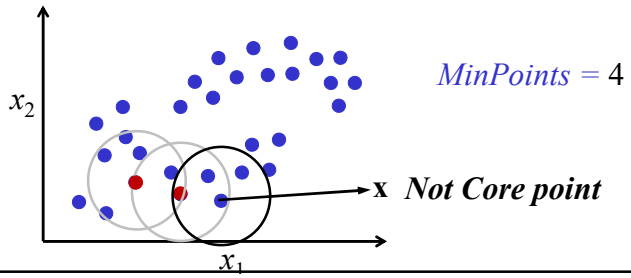
- DBSCAN has 5 important definitions (components):
  1. Epsilon (ε): It is a value of radius of boundary from every example
  2. *MinPoints*: Minimum number of examples present inside the boundary with radius of ε from an example **x**
  3. Core point: If there are atleast *MinPoints* number of examples are with in ε–radius from **x**, then **x** is called as core point



7

Density-based Special Clustering of Applications with Noise (DBSCAN)

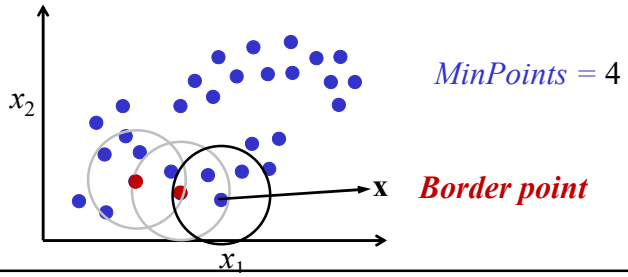
- DBSCAN has 5 important definitions (components):
  1. Epsilon (ε): It is a value of radius of boundary from every example
  2. *MinPts*: Minimum number of examples present inside the boundary with radius of ε from an example **x**
  3. Core point: If there are atleast *MinPoints* number of examples are with in ε–radius from **x**, then **x** is called as core point



8

Density-based Special Clustering of Applications with Noise (DBSCAN)

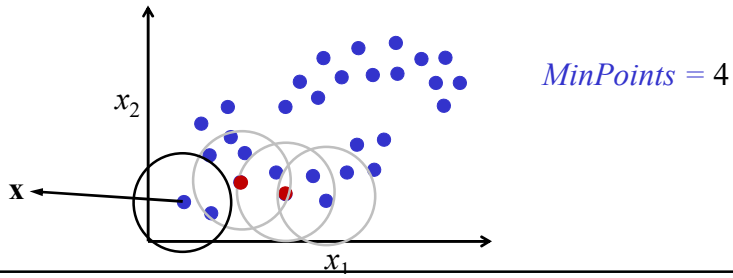
- DBSCAN has 5 important definitions (components):
  3. Core point: If there are atleast *MinPoints* number of examples are with in ε–radius from **x**, then **x** is called as core point
  4. Border point:
     - The number of examples within ε–radius from **x** is less than *MinPoints* **AND** atleast one of the example in neighborhood is core point, then **x** is called as border point



9

Density-based Special Clustering of Applications with Noise (DBSCAN)

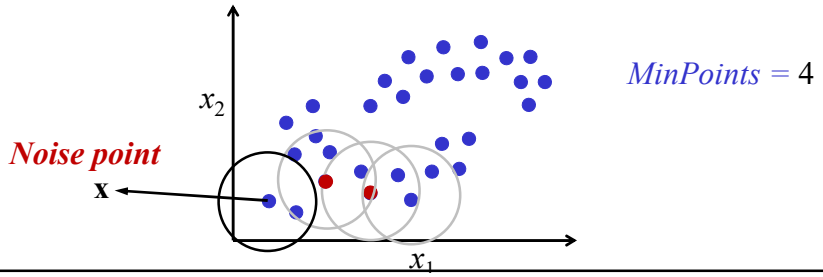
- DBSCAN has 5 important definitions (components):
  3. Core point: If there are atleast *MinPoints* number of examples are with in ε–radius from **x**, then **x** is called as core point
  4. Border point:
     - The number of examples within ε–radius from **x** is less than *MinPoints* **AND** atleast one of the example in neighborhood is core point, then **x** is called as border point



10

Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important definitions (components):
  5. Noise point:
     - The number of examples within ε–radius from $\mathbf{x}$ is less than *MinPoints* **AND** no example in neighborhood is core point
     - The noise point is similar to outlier



11

Clustering using DBSCAN

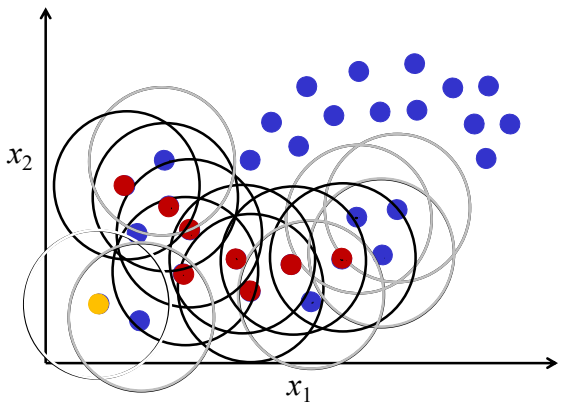
- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$
- First step is to identify core points, border points and noise points for the fixed ε and *MinPoints*
  - Note: ε and *MinPoints* are hyper-parameters (user defined parameters) of DBSCAN
  - Only core points and border points are considered inside the cluster
  - Noise points are not taken into the cluster
  - Thus DBSCAN is robust to outliers
- Next step is to find the connected components of core points
  - Connected component of core points: Connecting the core points that are reachable from any point
    - Each connected components is a graph i.e. tree
  - All the connected (reachable) core points form a cluster

Clustering using DBSCAN

- The connected component of core points is obtained by understanding following *two* definitions.
- Directly density-reachable: A core point $\mathbf{x}_i$ is directly density-reachable to a core point $\mathbf{x}_j$ if the core point $\mathbf{x}_j$ is within ε–distance from core point $\mathbf{x}_i$
- Density-reachable: A core point $\mathbf{x}_i$ is indirectly reachable to another core point $\mathbf{x}_j$ through other core points, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_K$ such that
  - $\mathbf{x}_i$ is directly density-reachable to $\mathbf{x}_1$
  - $\mathbf{x}_1$ is directly density-reachable to $\mathbf{x}_2$
  - ….
  - $\mathbf{x}_k$ is directly density-reachable to $\mathbf{x}_{k+1}$
  - ….
  - $\mathbf{x}_K$ is directly density-reachable to $\mathbf{x}_j$

Clustering using DBSCAN

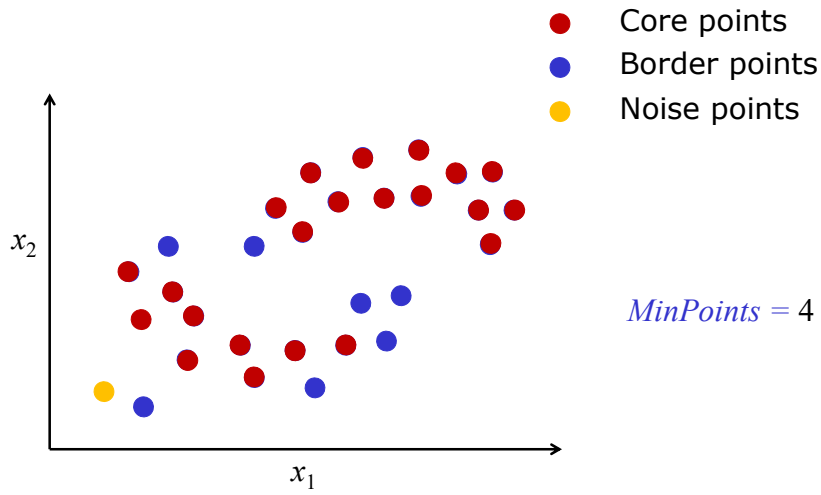
- Identify core points, border points and noise points



MinPoints = 4

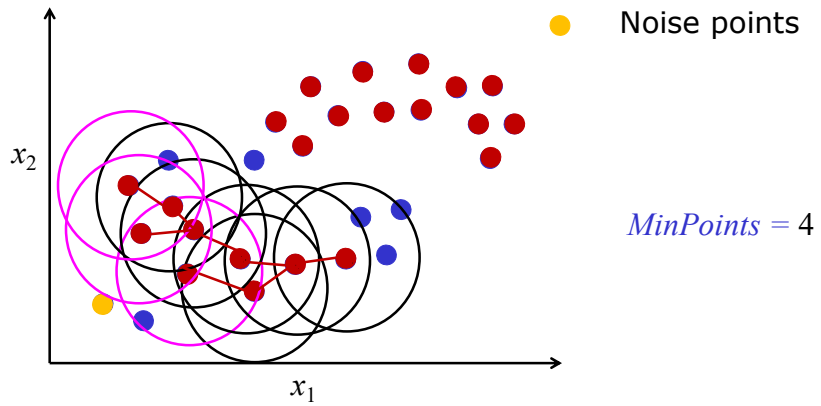
Clustering using DBSCAN

- Identify core points, border points and noise points



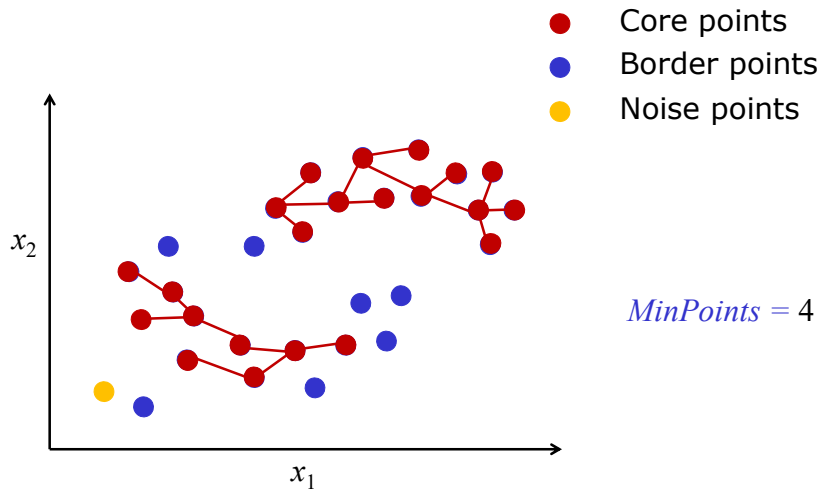
Clustering using DBSCAN

- Obtain the connected component of core points
  - Directly density-reachable:
  - Density-reachable:
  - Connected components does not include any loop/cycle



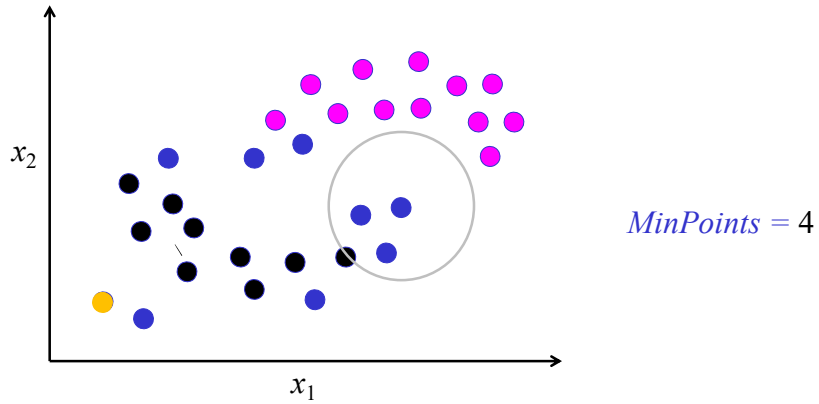
Clustering using DBSCAN

- All the core points with connected component forms a cluster



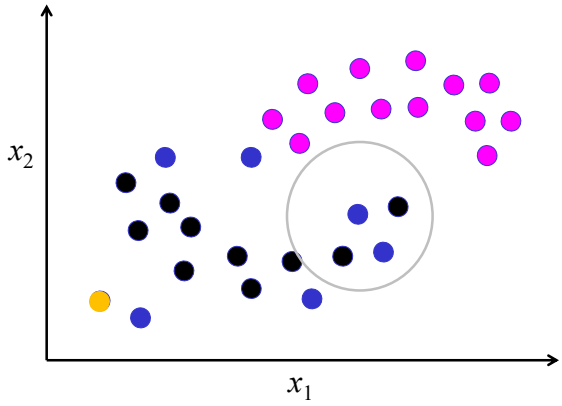
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



Clustering using DBSCAN

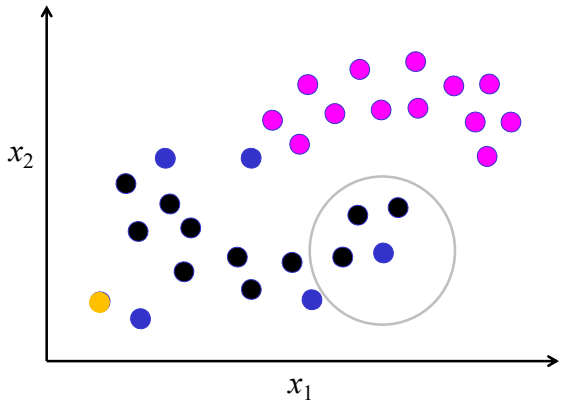
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

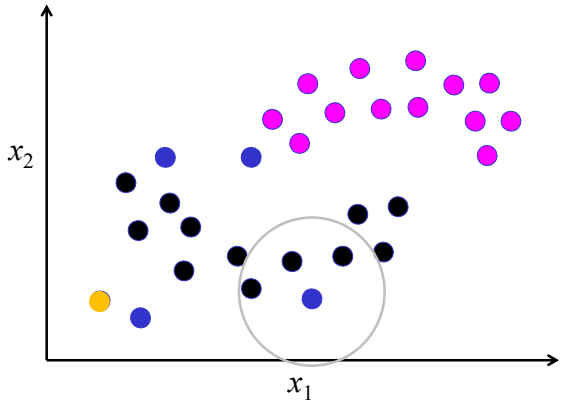
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

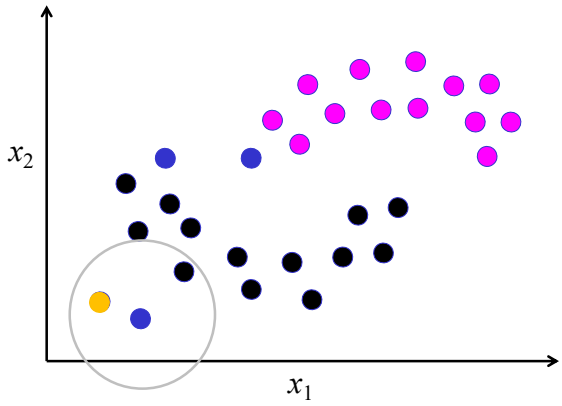
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

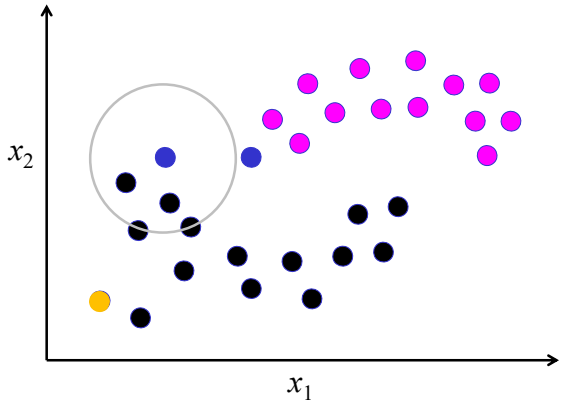
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

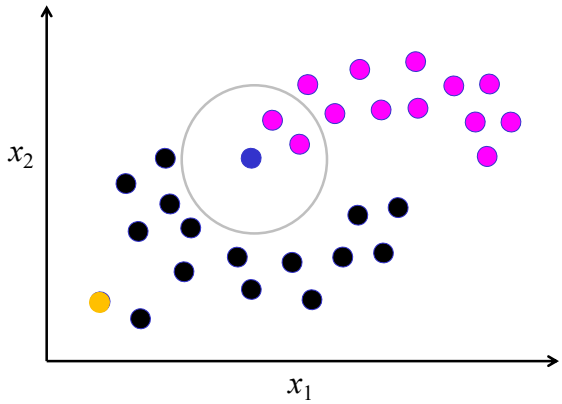
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

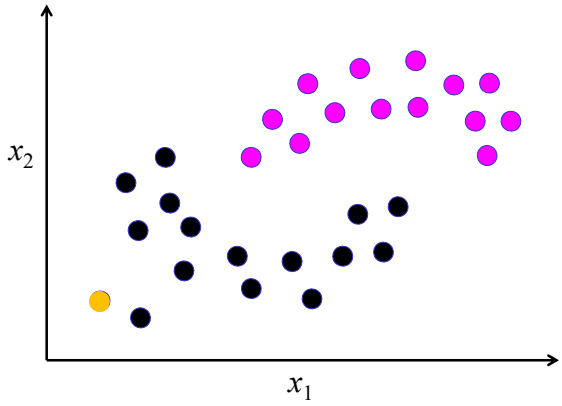
- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the border points to nearby cluster which is at ε–radius from that border point



$MinPoints = 4$

Clustering using DBSCAN
Training Phase

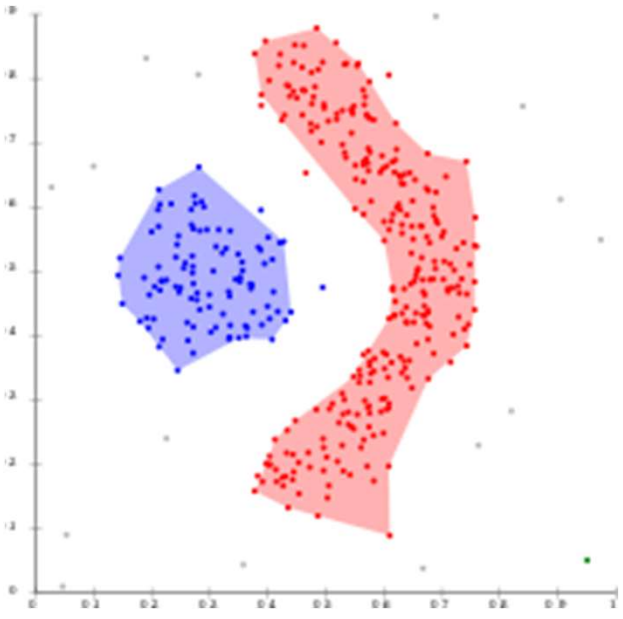
- Given: Training data, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^d$
- Identify the core points, border points and noise points
- Find the connected components of core points
- Each connected component forms a cluster
- Assign each of the border points to a nearby cluster which is at ε–radius from that border point
- Noise points are not assigned to any clusters
- Note: The parameters ε and *MinPoints* are experimentally set by the users

- Training process stores the core points as model

Clustering using DBSCAN
Test Phase

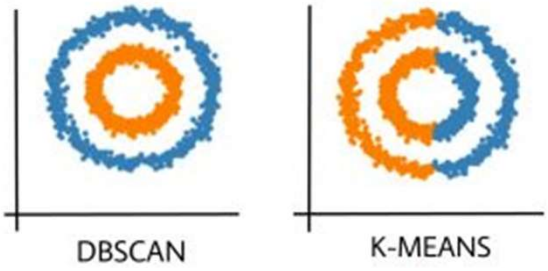
- For a test example, identify it as core point or border point or noise point
- If it is a core point, assign it to a cluster to which it is directly density-reachable or density-reachable
- If it is a border point, assign it to a nearby cluster which is at ε–radius from that border point
- If it is a noise point, do not assign to any cluster

Clustering using DBSCAN



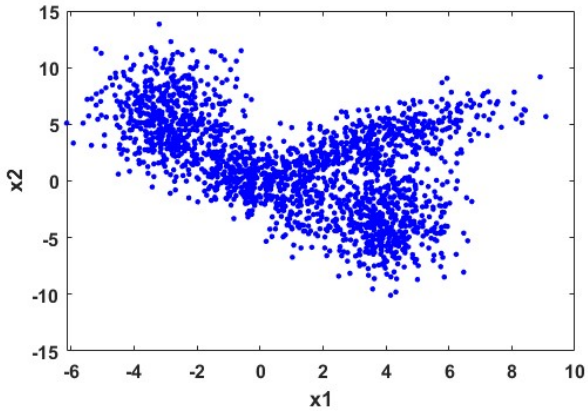
Advantages of DBSCAN

- DBSCAN does not require to specify the number of clusters in the data a priori
- DBSCAN can find arbitrarily shaped clusters
- DBSCAN has a notion of noise, and is robust to outliers



Limitation of DBSCAN

- DBSCAN is not suitable when the data is completely dense and there is no low dense area to separate



- The parameters ε and *MinPoints* should be chosen carefully

Summary: Density-Based Clustering

- These methods cluster collection of examples based on the notion of density
- These methods regard clusters as dense regions of examples in the data space that are separated by regions of low density (i.e. noise)
- They discover clusters with arbitrary shape
- Example: Density-based Spatial Clustering of Applications with Noise (DBSCAN)
  - It defines a cluster as a maximal set of density-connected points
  - No need to specify the number of clusters

31

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

32

Supervised Machine Learning: Pattern Classification
Logistic Regression

Classification

- Problem of identifying to which of a set of categories a new observation belongs
- Predicts categorical labels
- Example:
  - Classifying a person into the "adult" or "child" class
  - Classifying an Iris flower into one of the three categories: Iris Setosa, Iris Versicolour, Iris Virginica

2

Classification

- Classification is a two step process
  - Step1: Building a classifier (data modeling)
    - Learning from data (training phase)
    - Supervised learning: In supervised learning, each example is a *pair* consisting of an input example and a desired output value (class label)
    - Training phase or learning phase is viewed as the learning of a mapping or function that can predict the associated class label of a given training example

$$y_n = f(\mathbf{x}_n)$$

      - $\mathbf{x}_n$ is the n^{th} training example and y_n is the associated class label
  - Step2: Using classification model for prediction
    - Testing phase - Predicting class label for the unseen data
- Accuracy of a classifier: Percentage of test examples that are correctly classified by the classifier
- Target of learning techniques: Good generalization ability

3

Classification

- Training phase or learning phase is viewed as the learning of a mapping or function that can predict the associated class label of a given training example

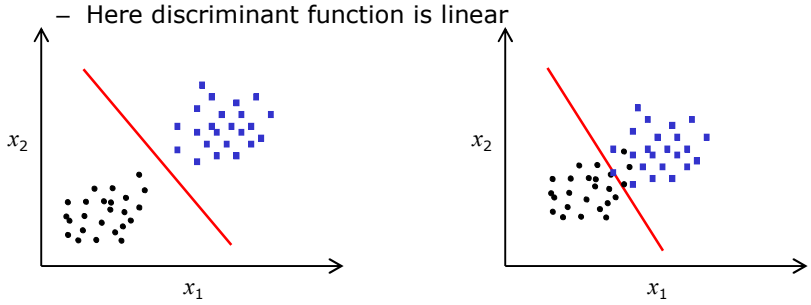
$$y_n = f(\mathbf{x}_n)$$

  - $\mathbf{x}_n$ is the n^{th} training example and y_n is the associated class label
- The mapping function $f(.)$ that is learnt during training phase is called discriminant function
- Discriminant function:
  - Function that discriminates the region of separation between the classes
  - Function that indicate the shape of the boundary between the classes
  - Shape of the boundary (discriminant function) is either linear or nonlinear

4

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear



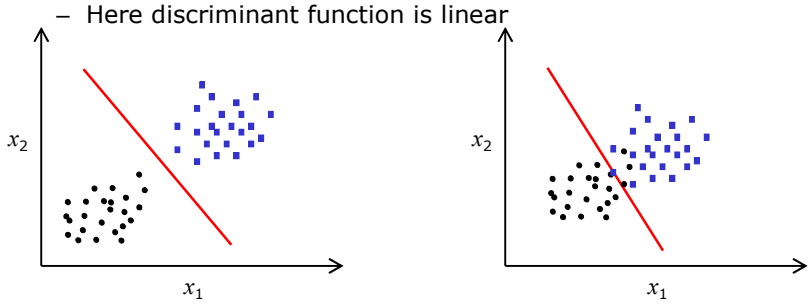
- Linear discriminant function in 2-dimensional space :

$\mathbf{x}_n=[x_{n1}, x_{n2}]^T$ $\quad f(\mathbf{x}_n, w_1, w_2, w_0) = w_1\, x_{n1} + w_2\, x_{n2} + w_0$

5

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear



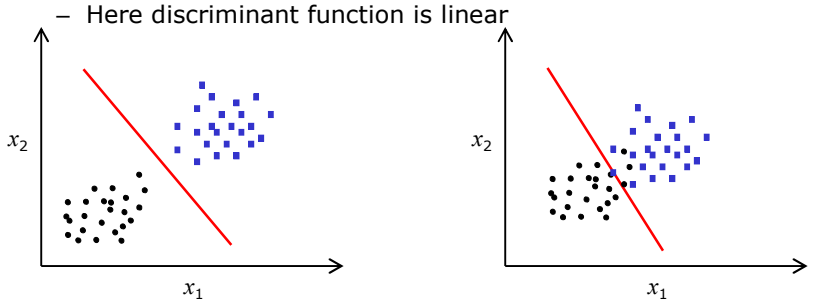
- Linear discriminant function in 2-dimensional space :

$\mathbf{x}_n=[x_{n1}, x_{n2}]^T$ $\quad f(\mathbf{x}_n, w_1, w_2, w_0) = w_1\, x_{n1} + w_2\, x_{n2} + w_0 = 0$

6

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear



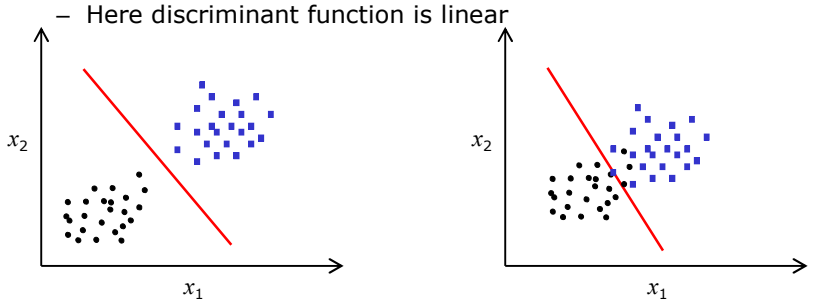
- Linear discriminant function in 2-dimensional space :

$\mathbf{x}_n=[x_{n1}, x_{n2}]^\top$ $\quad x_{n2} = \underbrace{-\frac{w_1}{w_2}}_{m} x_{n1} \underbrace{-\frac{w_0}{w_2}}_{c} = mx_{n1} + c$

7

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear



- Linear discriminant function in d-dimensional space :

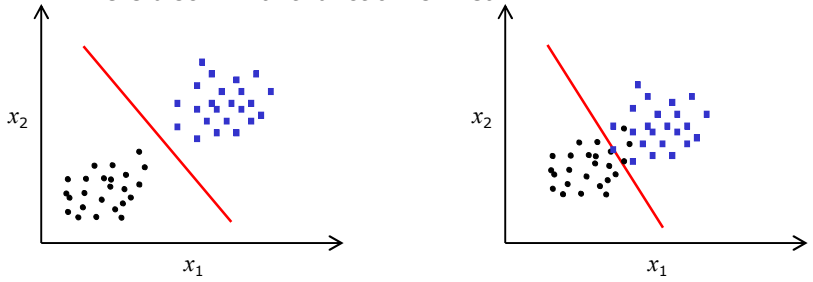
$\mathbf{x}_n=[x_{n1}, x_{n2}, \ldots, x_{nd}]^\top$

$f(\mathbf{x}_n, w_d, \ldots, w_1, w_2, w_0) = w_1 x_{n1} + w_2 x_{n2} + \ldots + w_d x_{nd} + w_0$

8

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear



- Discriminant function in d-dimensional space :

$\mathbf{x}_n=[x_{n1}, x_{n2}, \ldots, x_{nd}]^\mathsf{T}$ $\quad f(\mathbf{x}_n, \mathbf{w}, w_0) = \mathbf{w}^\mathsf{T}\mathbf{x}_n + w_0 = \sum_{i=0}^{d} w_i x_{ni}$

9

Linear Method for Classification: Logistic Regression

- *2-class classification*:
  - Class label: 0 or 1
- Requirement: Predict the probability of class

$$P(y = C \mid \mathbf{x})$$

  - The probability of class C should lead to a linear discriminant function $f(\mathbf{x},\mathbf{w})$
- *Let us define*
  - $P(\mathbf{x})$ is $P(C=1|\mathbf{x})$ i.e. probability that output is 1 given input (probability of success)
  - $1\text{-}P(\mathbf{x})$ is $P(C=0|\mathbf{x})$ i.e. probability that output is 0 given input (probability of failure)

10

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function:

$$\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = w_0 + w_1 x_1 + \ldots + w_d x_d = \mathbf{w}^T \hat{\mathbf{x}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^T$

  - For 1-dimensional (d=1) space, x

$$\log\left(\frac{P(x)}{1-P(x)}\right) = w_0 + w_1 x \qquad \frac{P(x)}{1-P(x)} = e^{(w_0 + w_1 x)}$$

11

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function:

$$\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^T \hat{\mathbf{x}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^T$

  - For 1-dimensional (d=1) space, x

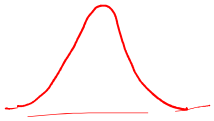
$$\frac{P(x)}{1-P(x)} = e^{(w_0 + w_1 x)}$$ Homework.

$$P(x) = \frac{e^{(w_0 + w_1 x)}}{1 + e^{(w_0 + w_1 x)}} = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

12

Logistic Regression

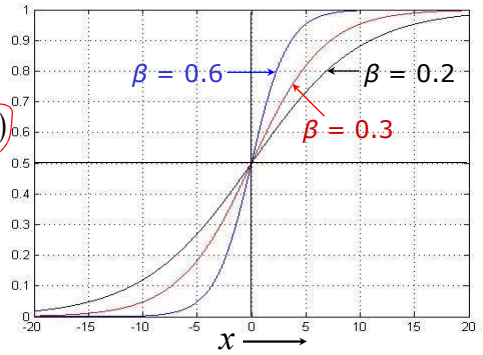
$$P(x) = \frac{1}{1+e^{-(w_0+w_1x)}}$$



- This function is a sigmoidal function, specifically called as logistic function
- Logistic function:

$$P(x) = \frac{1}{1+e^{-(w_0+w_1x)}}$$

$$P(x) = \frac{1}{1+e^{-(\beta x)}}$$



- For any test example x:
  - If $P(x) \geq 0.5$ then x is assigned with label 1
  - If $P(x) < 0.5$ then x is assigned with label 0

13

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^T\hat{\mathbf{x}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^T$

$$\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = w_0 + w_1x_1 + \ldots + w_dx_d = \mathbf{w}^T\hat{\mathbf{x}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^T$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^T\hat{\mathbf{x}})}$$

14

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^{\mathsf{T}}$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})} \quad \text{where } \mathbf{w}=[w_0, w_1, \ldots, w_d]^{\mathsf{T}} \text{ and } \hat{\mathbf{x}}=[1, x_1, \ldots, x_d]^{\mathsf{T}}$$

$$P(\mathbf{x}) = \frac{e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}{1+e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}$$

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}$$

What about the classifier learning here?

It is still a linear classifier – Boundary is linear surface i.e. hyperplane

15

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^{\mathsf{T}}$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})} \quad \text{where } \mathbf{w}=[w_0, w_1, \ldots, w_d]^{\mathsf{T}} \text{ and } \hat{\mathbf{x}}=[1, x_1, \ldots, x_d]^{\mathsf{T}}$$

$$P(\mathbf{x}) = \frac{e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}{1+e^{(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}$$

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}$$

For any test example $\mathbf{x}$:

If $P(\mathbf{x}) \geq 0.5$ then $\mathbf{x}$ is assigned with label 1

If $P(\mathbf{x}) < 0.5$ then $\mathbf{x}$ is assigned with label 0

16

Logistic Regression - Training Phase: Estimation of Parameter

- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^\top$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}$

$x_n \in$ class 1 ($y_n = 1$) $\quad x_n \in$ class 0 ($y_n = 0$)

Probability that $\mathbf{x}$ has label 1 — Probability that $\mathbf{x}$ has label 0

17

Logistic Regression - Training Phase: Estimation of Parameter

- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^\top$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}$ — Binomial distribution (Bernoulli Distribution)
- Total data likelihood: $P(\mathcal{D} \mid \mathbf{w}) = \prod_{n=1}^{N} P(\mathbf{x}_n \mid \mathbf{w})$

18

Logistic Regression - Training Phase: Estimation of Parameter

- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^T$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}$ — Binomial distribution (Bernoulli Distribution)
- Total data likelihood: $P(\mathcal{D} \mid \mathbf{w}) = \prod_{n=1}^{N} P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}$

19

Logistic Regression - Training Phase: Estimation of Parameter

- Total data log likelihood:

$$l(\mathbf{w}) = \ln\left(P(\mathcal{D} \mid \mathbf{w})\right)$$

$$l(\mathbf{w}) = \sum_{n=1}^{N} y_n \ln(P(\mathbf{x}_n)) + (1-y_n)\ln(1-P(\mathbf{x}_n))$$

- Choose the parameters for which the total data log likelihood is maximum:

$$\mathbf{w}_{\text{ML}} = \arg\max_{\mathbf{w}} l(\mathbf{w})$$

- Cost function for optimization:

$$l(\mathbf{w}) = \sum_{n=1}^{N} y_n \ln(P(\mathbf{x}_n)) + (1-y_n)\ln(1-P(\mathbf{x}_n))$$

- Conditions for optimality: $\frac{\partial\, l(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

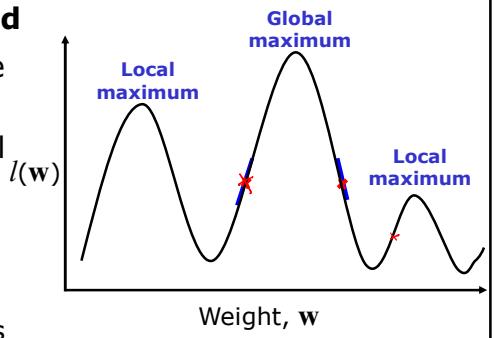
$$\begin{bmatrix} \frac{\partial\, l(\mathbf{w})}{\partial w_0} \\ \frac{\partial\, l(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial\, l(\mathbf{w})}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \\ 0 \end{bmatrix}$$

- Unfortunately, solving this, no closed form expression for $\mathbf{w}$ is obtained
- Solution: Gradient accent method

20

Estimation of Parameter in Logistic Regression

- **Gradient accent method**
- It is an iterative procedure
- We start with an initial value for $\mathbf{w}$
- At each iteration:
  - Estimate change in $\mathbf{w}$
  - The change in $\mathbf{w}$ ($\Delta\mathbf{w}$) is proportional to the slope (gradient) of the likelihood surface
  - Then, the $\mathbf{w}$ is updated using $\Delta\mathbf{w}$
- This indicate, we move in the positive slope of the likelihood surface, likelihood is maximum in each iteration



$$\Delta\mathbf{w} \propto -\frac{\partial\, l(\mathbf{w})}{\partial \mathbf{w}} \qquad \Delta\mathbf{w} = -\eta \frac{\partial\, l(\mathbf{w})}{\partial \mathbf{w}}$$

where $0 \le \eta \le 1$ is proportionality constant

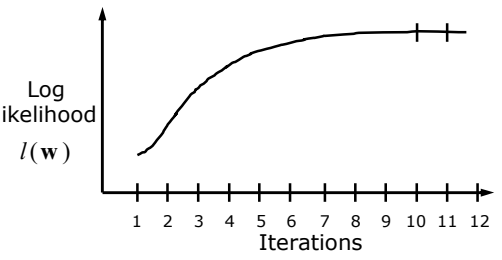
21

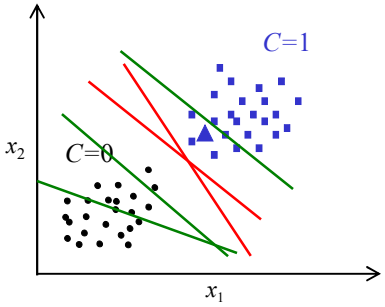
Estimation of Parameter in Logistic Regression – Gradient Accent Method

- Given a training dataset, the goal is to maximize the likelihood function with respect to the parameters of linear function
  1. Initialize the $\mathbf{w}$
     - Evaluate the initial value of the log likelihood, $l(\mathbf{w})$
  2. Determine the change in $\mathbf{w}$ ($\Delta\mathbf{w}$): $\Delta\mathbf{w} = -\eta \frac{\partial\, l(\mathbf{w})}{\partial \mathbf{w}}$
  3. Update the $\mathbf{w}$: $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
  4. Evaluate the log likelihood and check for convergence of the log likelihood
     - If the convergence criterion is not satisfied repeat from steps 2 to 4
- Convergence criterion: Difference between log likelihoods of successive iterations fall below a threshold (E.g. threshold=10^{-3})

22

Estimation of Parameter in Logistic Regression – Gradient Accent Method





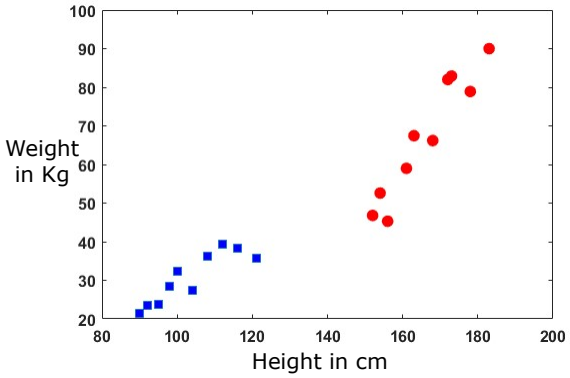
- **Test phase:**
- Classification of a test pattern $\mathbf{x}$ using the weights $\mathbf{w}$ obtained by training the model:
  - If $P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x})}} \geq 0.5$ then $\mathbf{x}$ is assigned to class with label 1
  - If $P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x})}} < 0.5$ then $\mathbf{x}$ is assigned to class with label 0

23

Illustration of Classification using Logistic Regression: Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Number of training examples (N) = 20
- Dimension of a training example = 2
- Class label attribute:
  - Child (0)
  - Adult (1)



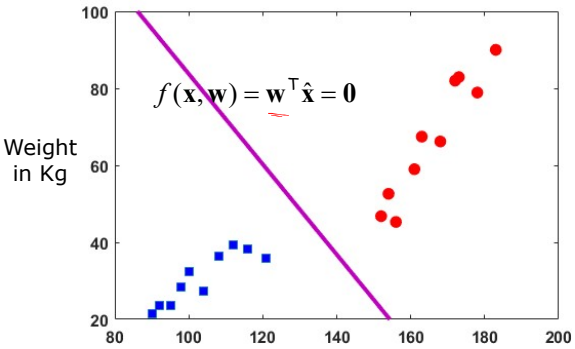
24

Illustration of Classification using Logistic Regression: Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Training:

$$\mathbf{w} = \begin{bmatrix} -378.2085 \\ 2.2065 \\ 1.8818 \end{bmatrix} \begin{matrix} w_0 \\ w_1 \\ w_2 \end{matrix}$$

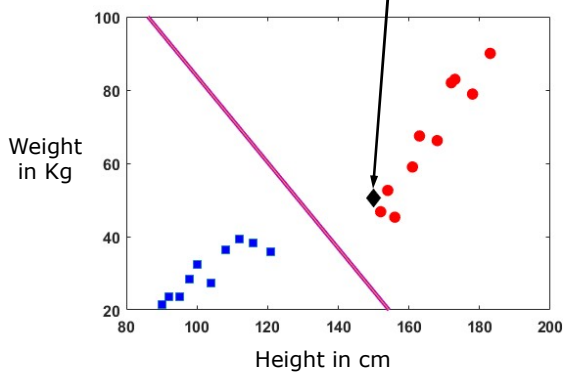


25

Illustration of Classification using Logistic Regression: Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

Test Example: 150 | 50.6



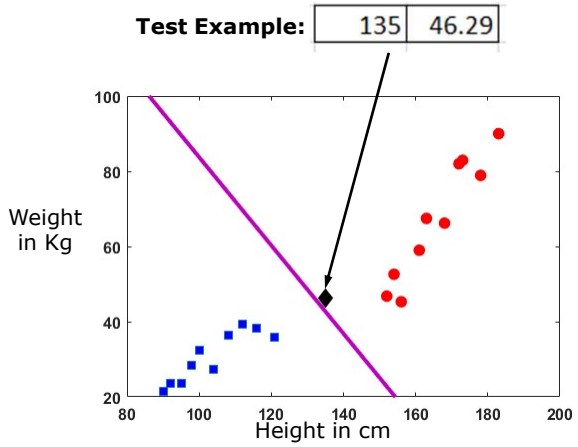
$$\mathbf{w}^T\hat{\mathbf{x}} = 47.9851 \qquad P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\hat{\mathbf{x}})}} = 1$$

- Class: Adult (C2)

26

Illustration of Classification using Logistic Regression: Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1



$$\mathbf{w}^T\hat{\mathbf{x}} = 6.7771 \qquad P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\hat{\mathbf{x}})}} = 0.9$$

- Class: Adult (C2)

27

Summery: Logistic Regression

- Logistic regression is a linear classifier
- Logistic regression looks simple, but yields a very powerful classifier
- It is used not just building classifier, but also used in sensitivity analysis
  - Logistic regression is used to identify how each attribute contribute to output
    - How much each attribute is important for predicting class label
- Perform logistic regression and observe $\mathbf{w}$
- The value of each element of $\mathbf{w}$ indicate how much each attribute is contributing to the output

28

Text Books

1. R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, John Wiley, 2001.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

4. Satish Kumar, Neural Networks - A Class Room Approach, Second Edition, Tata McGraw-Hill, 2013.

5. S. Haykin, Neural Networks and Learning Machines, Prentice Hall of India, 2010.

29

Supervised Machine Learning: Pattern Classification
Logistic Regression

Linear Method for Classification: Logistic Regression

- *2-class classification*:
  - Class label: 0 or 1
- Requirement: Predict the probability of class

$$P(y = C \mid \mathbf{x})$$

  - The probability of class C should lead to a linear discriminant function $f(\mathbf{x},\mathbf{w})$
- *Let us define*
  - $P(\mathbf{x})$ is $P(C{=}1|\mathbf{x})$ i.e. probability that output is 1 given input (probability of success)
  - $1\text{-}P(\mathbf{x})$ is $P(C{=}0|\mathbf{x})$ i.e. probability that output is 0 given input (probability of failure)

2

Logistic Regression

- Logit function: Log of odds function $\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)$
- Odds function: $\frac{P(\mathbf{x})}{1-P(\mathbf{x})}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function:

$$\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = w_0 + w_1 x_1 + \ldots + w_d x_d = \mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^{\mathsf{T}}$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^{\mathsf{T}}$

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}}$$

- $P(\mathbf{x})$ is now defined by logistic function
- For any test example $\mathbf{x}$:
  - If $P(\mathbf{x}) \geq 0.5$ then $\mathbf{x}$ is assigned with label 1
  - If $P(\mathbf{x}) < 0.5$ then $\mathbf{x}$ is assigned with label 0

3

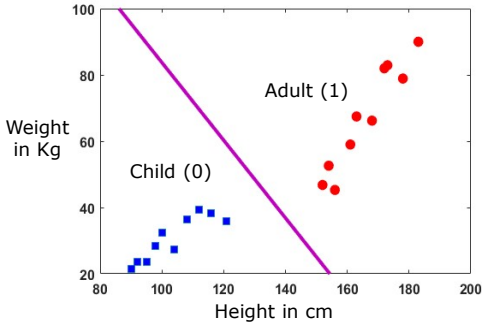
Logistic Regression - Training Phase: Estimation of Parameter

- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1, \ldots, w_d]^{\mathsf{T}}$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- As the goal is to model the probability of class, we are maximizing the total log-likelihood of data
  - It is shown to follow Bernoulli distribution (binomial distribution)
- A gradient ascent method is used to solve the maximum likelihood method and optimal parameters, $\mathbf{w}$, are obtained

4

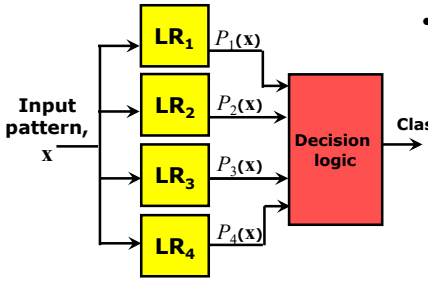
Logistic Regression - Test Phase

- Classification of a test pattern $\mathbf{x}$ using the weights $\mathbf{w}$ obtained by training the model:
  - If $P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\hat{\mathbf{x}})}} \geq 0.5$ then $\mathbf{x}$ is assigned to class with label 1
  - If $P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\hat{\mathbf{x}})}} < 0.5$ then $\mathbf{x}$ is assigned to class with label 0
- Adult(1)-Child(0) Classification:



5

Multi-class Pattern Classification using Logistic Regression (LR)



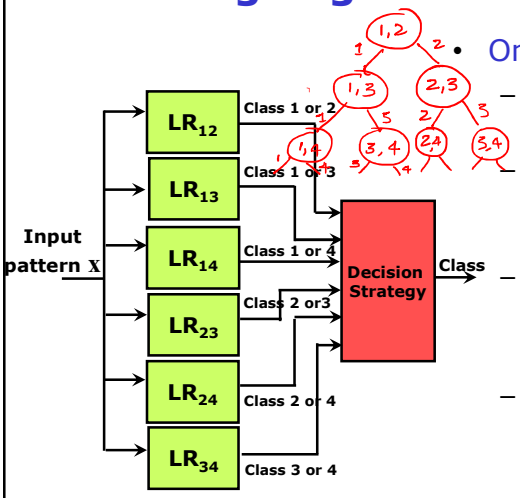
- One-against-rest approach:

Class label for $\mathbf{x}$ = $\arg\max_i P_i(\mathbf{x}) \quad i = 1, 2, ..., M$

- A logistic regression model is built for each class where one of the class is 1 rest of the class is 0
  - **LR**i : Class i is considered as class with label 1, then rest of the classes are considered as
  - This is repeated for each of the M classes.
- Let C=4 be the total number of classes. Then, number of logistic regression models is L = 4
- A test pattern $\mathbf{x}$ is classified by using winner-takes-all strategy

6

Multi-class Pattern Classification using Logistic Regression (LR)



- One-against-one approach:
  - A logistic regression model is built for every pair of classes
  - Let number of classes, M=4 be the total number of classes
  - Number of pairwise logistic regression models is

$$L = M(M-1)/2 = 6$$

  - The maxwins strategy is used to determine the class of a test pattern $\mathbf{x}$
    - In this strategy, a majority voting scheme is used

- **LR$_{ij}$**: Logistic regression model built between the classes i and j

7

Classification and Discriminant Function

- Training phase or learning phase is viewed as the learning of a mapping or function that can predict the associated class label of a given training example

$$y_n = f(\mathbf{x}_n)$$

  - $\mathbf{x}_n$ is the n^{th} training example and y_n is the associated class label
- The mapping function $f(.)$ that is learnt during training phase is called discriminant function
- Discriminant function:
  - Function that discriminates the region of separation between the classes
  - Function that indicate the shape of the boundary between the classes
  - Shape of the boundary (discriminant function) is either linear or nonlinear

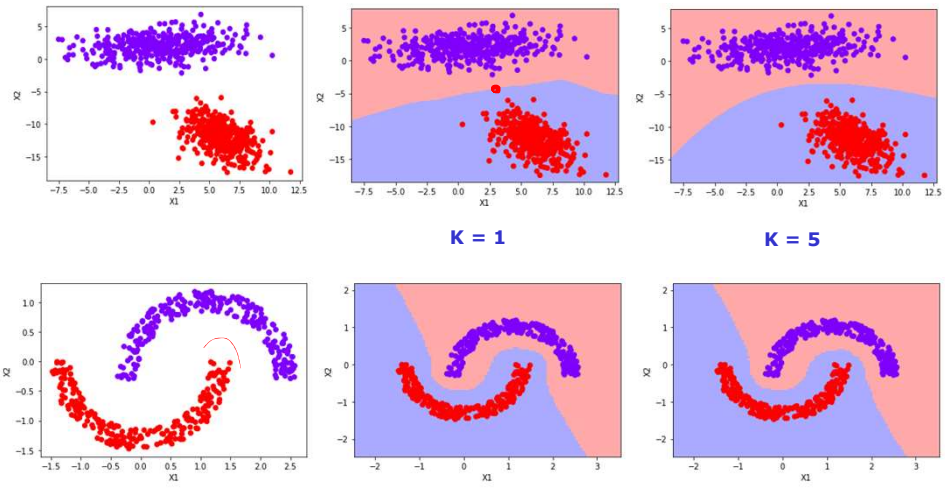
8

Classification and Discriminant Function

- The K-NN classifier and Bayes classifier does not learn the discriminant function $y_n = f(\mathbf{x}_n)$ directly
- K-NN simply stores the training examples
- Bayes classifier estimate the parameter of the distribution from the training data of each classes independently
- However, they implicitly learn the discriminant function
- The shape of the decision boundary (discriminant function) will be seen during the testing (validation) process
- K-NN classifier indirectly learn the non-linear discriminant function
- Bayes classifier with unimodal Gaussian density implicitly learns nonlinear (hyperquadratic – up to second order polynomial function) discriminant function
- Bayes classifier with multimodal Gaussian density (Gaussian mixture model) implicitly learns nonlinear discriminant function

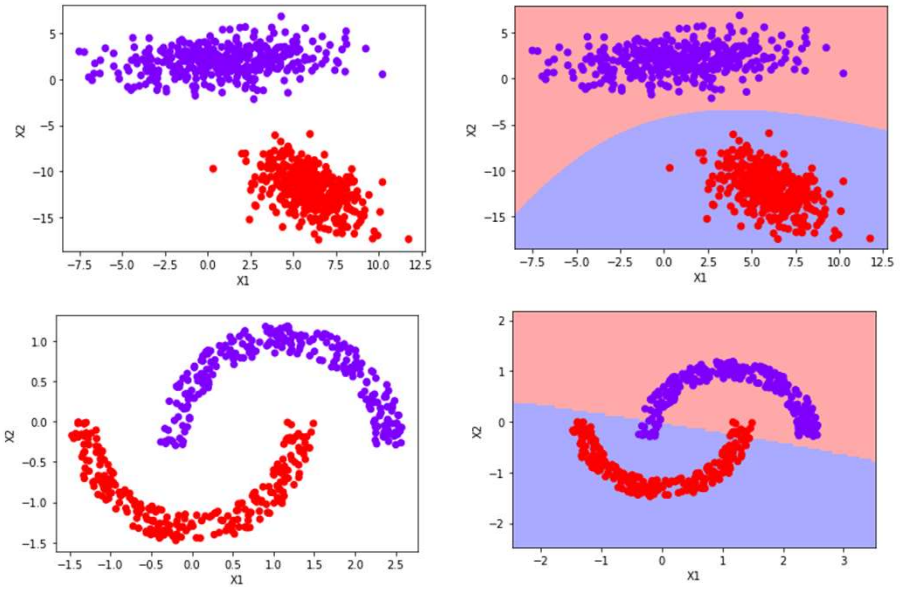
9

Illustration: Discriminant Function Learnt in K-NN Classifier



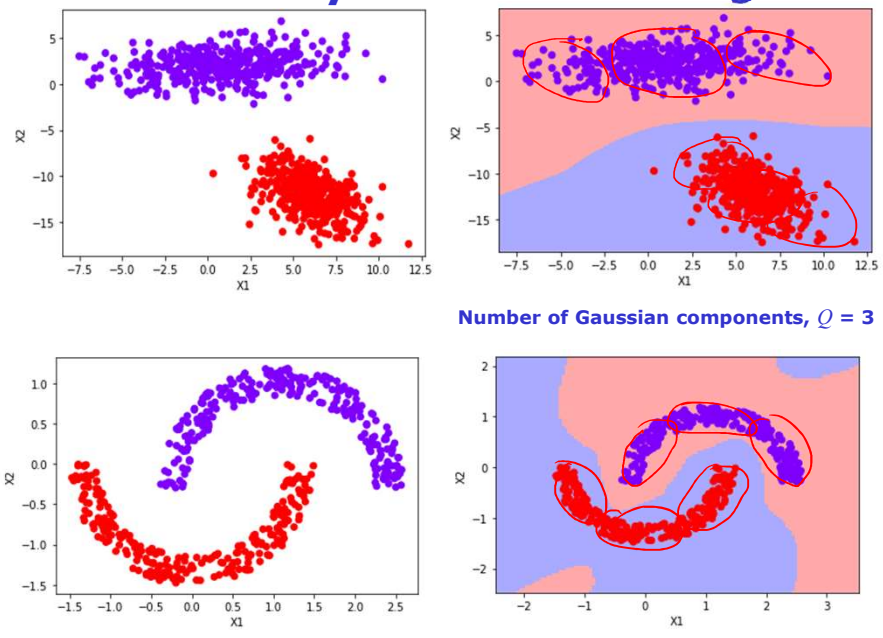
10

Illustration: Discriminant Function Learnt in Bayes Classifier with Unimodal Gaussian Density



11

Illustration: Discriminant Function Learnt in Bayes Classifier using GMM



Number of Gaussian components, Q = 3

12

Two Classes of Approaches for Classification

1. Modeling a discriminating function:

2. Directly learn a discriminant function (hyperplane/hypersurface):

13

Two Classes of Approaches for Classification

1. Modeling a discriminating function:
   - Let C_1, C_2, …, C_i, …, C_M be the M classes
   - For each class, a discriminant function $g_i(\mathbf{x}, \mathbf{w}_i)$ is defined
     - $\mathbf{w}_i$ is the parameter of the model for a class C_i
   - Let $g_i(\mathbf{x}, \mathbf{w}_i)$ be the discriminant function for i^{th} class

     Class label for $\mathbf{x}$ = $\arg\max_i g_i(\mathbf{x}, \mathbf{w}_i)$ $\quad i = 1, 2, \ldots, M$

   - Discriminant function is defined independent of the classes
   - Non-discriminative learning

14

Two Classes of Approaches for Classification

1. Modeling a discriminating function:
   - Let $C_1, C_2, \ldots, C_i, \ldots, C_M$ be the M classes
   - Let $g_i(\mathbf{x}, \mathbf{w}_i)$ be the discriminant function for i^{th} class
     - $\mathbf{w}_i$ is the parameter of the model for a class C_i
   - Discriminant function is defined independent of the classes
   - Non-discriminative learning
   - Examples:
     - K-nearest neighbor classifier:
       - $g_i(\mathbf{x}, \mathbf{w}_i)$ = Euclidian distance between $\mathbf{x}$ and training data of class C_i
     - Bayes classifier using unimodal (Gaussian) and multimodal distribution (GMM):
       - $g_i(\mathbf{x}, \mathbf{w}_i) = P(\mathbf{x} \mid \mathbf{w}_i)$: Posterior probability of a class
     - Logistic regression:

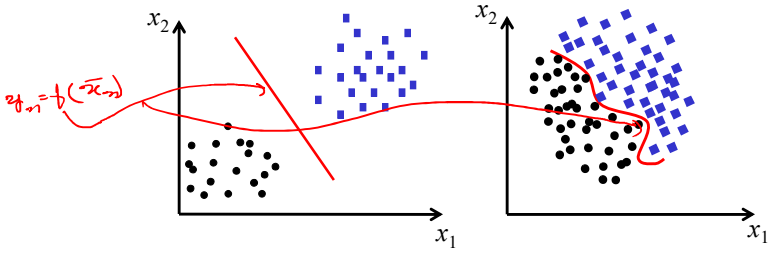
$$g_i(\mathbf{x}, \mathbf{w}_i) = P(\mathbf{x} \mid \mathbf{w}_i) = \frac{1}{1+e^{-(\mathbf{w}_i^T \hat{\mathbf{x}})}}$$

Probability of success – Probability of class with label 1

15

Two Classes of Approaches for Classification

2. Directly learn a discriminant function (hyperplane/hypersurface):
   - *Classic method*: Discriminant function between the classes is learnt



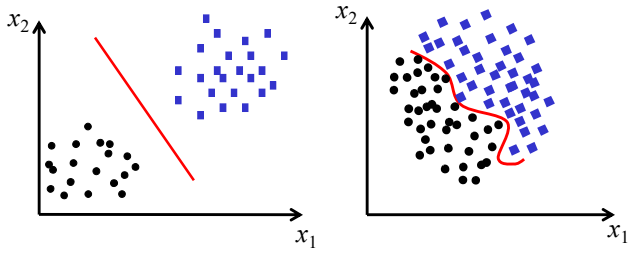
   - Perceptron (linear discriminant function is learnt)
   - Support vector machine (SVM) (linear discriminant function is learnt in input or transformed feature space)
   - Neural networks (learning to approximate nonlinear discriminant function)

16

Supervised Machine Learning: Discriminative Learning Methods for Pattern Classification
Perceptron

Discriminative Learning Methods for Classification

- Directly learn the surface (discriminant function) that better separates the region of classes, from the training data
- Learns a function that maps input data to output



- Perceptron method: Learns linear discriminant function
  - Linear discriminant function: Function that indicate the boundary between the classes which is linear

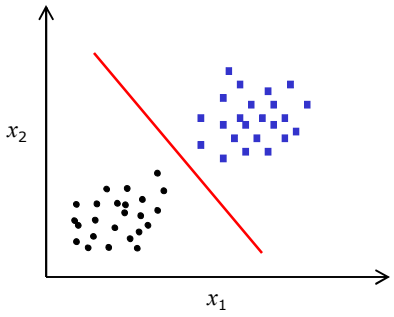
18

Linear Discriminant Function

- Regions of two classes are separable by a linear surface (line, plane or hyperplane)
- 2-dimensional space: The decision boundary is a **line** specified by

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



- d-dimensional space: The decision surface is a **hyperplane** specified by

$$w_d x_d + ... + w_2 x_2 + w_1 x_1 + w_0 = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^T \hat{\mathbf{x}} = 0$$

where $\mathbf{w} = [w_0, w_1, ..., w_d]^T$ and $\hat{\mathbf{x}} = [1, x_1, ..., x_d]^T$

19

Discriminant Function of a Hyperplane

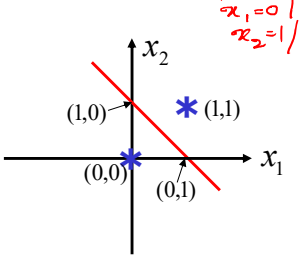
- The discriminant function of a hyperplane:

$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

- For any point the lies on the hyperplane

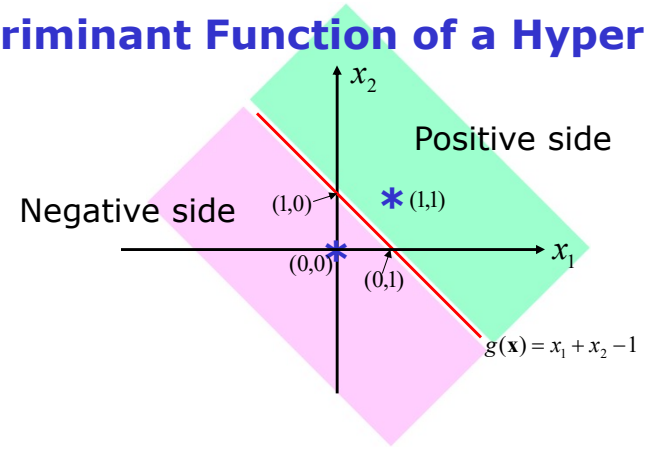
$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0 = 0$$

- Example:
  - Consider a straight line with its equation as $x_2 + x_1 - 1 = 0$
  - Discriminant function of the straight line is $g(\mathbf{x}) = x_2 + x_1 - 1$
  - For points (1,0) and (0,1) that lie on this straight line $g(\mathbf{x}) = 0$
  - For the point (0,0), $g(\mathbf{x}) = -1$ i.e. the value of $g(\mathbf{x})$ is negative
  - For the point (1,1), $g(\mathbf{x}) = +1$ i.e. the value of $g(\mathbf{x})$ is positive



20

Discriminant Function of a Hyperplane

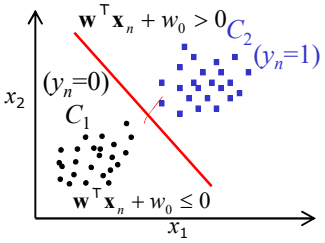


- A hyperplane has a positive side and a negative side
  - For any point on the positive side, the value of discriminant function, $g(\mathbf{x})$, is positive
  - For any point on the negative side, the value of discriminant function, $g(\mathbf{x})$, is negative

21

Perceptron Model

- Perceptron model is a 2-class classification model that uses perceptron learning algorithm [1]
- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{0,1\}$
  - N: number of training examples
  - $\mathbf{x}_n$: n^{th} training example
  - y_n: Class label of n^{th} training example
- Goal: To estimate parameter vector $\mathbf{w}=[w_0, w_1, \ldots, w_d]^{\mathsf{T}}$
  - such that linear function (hyperplane) is placed between the training data of two classes so that training error (classification error) is zero
  - Learning by adaptation



[1] A.G. Ivakhnenko and V.G. Lapa. Cybernetic predicting devices. 1965.

22

Perceptron Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{0,1\}$
  1. Initialize the $\mathbf{w}$ with random values
  2. Choose a training example $\mathbf{x}_n$
  3. Update the $\mathbf{w}$, if $\mathbf{x}_n$ is misclassified

     $\mathbf{w} = \mathbf{w} + \eta\, \mathbf{x}_n$, for $\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + w_0 \le 0$ and $\mathbf{x}_n \in$ class with label 1 (y_n=1)

     $\mathbf{w} = \mathbf{w} - \eta\, \mathbf{x}_n$, for $\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + w_0 > 0$ and $\mathbf{x}_n \in$ class with label 0 (y_n=0)
     - Here $0 < \eta < 1$ is a positive, learning rate parameter
     - Increment the misclassification count by 1

Perceptron Learning

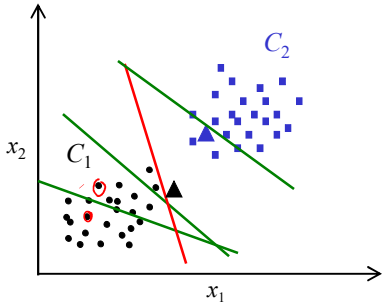
- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{0,1\}$
  1. Initialize the $\mathbf{w}$ with random values
  2. Choose a training example $\mathbf{x}_n$
  3. Update the $\mathbf{w}$, if $\mathbf{x}_n$ is misclassified

     $$\mathbf{w} = \mathbf{w} + \eta\,(y_n - \hat{y}_n)\mathbf{x}_n$$

     If $\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 > 0$ then $\hat{y}_n$ is 1
     If $\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 \le 0$ then $\hat{y}_n$ is 0
     - Here, y_n is actual label and $\hat{y}_n$ is the predicted label
     - Here $0 < \eta < 1$ is a positive, learning rate parameter
     - Increment the misclassification count by 1
  4. Repeat steps 2 and 3 till all the training examples are presented
  5. Repeat steps 2 to 4 by setting misclassification count to 0, till the convergence criterion is satisfied
- Convergence criterion:
  - Total misclassification count is 0

Perceptron Learning

- Training:
  - *Error correction law*
  - Learning rate parameter:
    - Should be wisely chosen
    - Larger value:
      - Error goes down faster
      - Oscillation before converging
    - Smaller value:
      - Slow convergence



- **Test phase:**
- Classification of a test pattern $\mathbf{x}$ using the weights $\mathbf{w}$ obtained by training the model:
  - If $\mathbf{w}^T\mathbf{x} + w_0 > 0$ then $\mathbf{x}$ is assigned to class with label 1 (C_2)
  - If $\mathbf{w}^T\mathbf{x} + w_0 \leq 0$ then $\mathbf{x}$ is assigned to class with label 0 (C_1)

25

Summary: Perceptron Learning

- Perceptron model is a 2-class classification model that uses perceptron learning algorithm
- Perceptron learning algorithm: Learns linear discriminant function
- Perceptron convergence theorem:
  - Perceptron learning law converges to a *finite set of weight values* in *finite number of steps* if the classes are linearly separable
- Limitation:
  - Not suitable when the data not linearly separable
    - *This limitation is addressed in support vector machine (SVM) and neural network (single neuron model)*
  - The learnt linear discriminant function is not optimal
    - *This limitation is addressed in SVM*

26

Text Books

1. R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, John Wiley, 2001.

2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.

3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

4. Satish Kumar, Neural Networks - A Class Room Approach, Second Edition, Tata McGraw-Hill, 2013.

5. S. Haykin, Neural Networks and Learning Machines, Prentice Hall of India, 2010.

27