Introduction to Machine Learning

Dr Pradipta Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
http://cpdm.iisc.ernet.in/PBiswas.htm
Contents

- Supervised and Unsupervised Learning
  - Supervised Learning
    - Decision Tree
    - Linear Regression – Gradient Descent
    - Neural Network – Backpropagation Algorithm
  - Clustering
    - K-means
    - K-medoid
    - Hierarchical
    - Cluster Validation Index
  - IUI Case Studies
Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set

- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
What is classification? What is regression?

Issues regarding classification and prediction

Classification by decision tree induction
Classification vs. Prediction

- **Classification:**
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values *(class labels)* in a classifying attribute and uses it in classifying new data
- **Regression:**
  - models continuous-valued functions, i.e., predicts unknown or missing values
- **Typical Applications**
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis
Credit approval

- A bank wants to classify its customers based on whether they are expected to pay back their approved loans
- The history of past customers is used to train the classifier
- The classifier provides rules, which identify potentially reliable future customers

Classification rule:
- If age = “31...40” and income = high then credit_rating = excellent

Future customers
- Paul: age = 35, income = high ⇒ excellent credit rating
- John: age = 20, income = medium ⇒ fair credit rating
Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test samples is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
**Classification Process (1): Model Construction**

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

**IF** rank = ‘professor’ **OR** years > 6  
**THEN** tenured = ‘yes’
Classification Process (2): Use the Model in Prediction

Accuracy=?

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Mellisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

Tenured?

Yes
Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers

- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree
## Training Dataset

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>
Output: A Decision Tree for “buys_computer”

1. **age?**
   - **<=30**
     - **student?**
       - no
         - no
       - yes
         - yes
   - **30..40**
     - yes
   - **>40**
     - credit rating?
       - excellent
         - no
       - fair
         - yes
Linear regression technique
- Basic but powerful machine learning algorithm
- Fitting a straight line through a set of points

https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e
Cost Function/Loss Function evaluates the performance of Machine Learning Algorithm.

- Loss function computes the error for a single training example
- Cost function is the average of the loss functions for all the training examples

\[
\text{Cost} = \frac{1}{N} \sum_{i=1}^{N} (Y' - Y)^2
\]
Parameters with small changes:

\[
\begin{align*}
m &= m - \delta m \\
b &= b - \delta b
\end{align*}
\]

Given Cost Function for 'N' no of samples

\[
Cost = \frac{1}{N} \sum_{i=1}^{N} (Y'_i - Y_i)^2
\]

Cost function is denoted by J where J is a function of m and b

\[
J_{m,b} = \frac{1}{N} \sum_{i=1}^{N} (Y'_i - Y_i)^2
\]

Substituting the term \(Y'_i - Y_i\) with error for simplicity

\[
J_{m,b} = \frac{1}{N} \sum_{i=1}^{N} (Error_i)^2
\]
Derivatives

\[ J_{m,b} = \frac{1}{N} \sum_{i=1}^{N} (Error_i)^2 \]

\[ \frac{\partial}{\partial m} \text{Error} = \frac{\partial}{\partial m} (Y' - Y) \]

\[ \frac{\partial}{\partial b} \text{Error} = \frac{\partial}{\partial b} (mX + b - Y) \]

\[ \frac{\partial}{\partial m} \text{Error} = \mathbf{x} \]

\[ \frac{\partial}{\partial b} \text{Error} = 1 \]
\[ \frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial m} \text{Error} \]

\[ \frac{\partial J}{\partial b} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial b} \text{Error} \]

\[ \frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot X \cdot \text{Learning Rate} \]

\[ \frac{\partial J}{\partial b} = \text{Error} \cdot \text{Learning Rate} \]

\[
\begin{align*}
\text{Determines the} & \quad \text{Determines how large a} \\
\text{direction to} & \quad \text{step to take} \\
\text{minimize the} & \quad \text{Error} \\
\text{Error} & 
\end{align*}
\]

\[
\begin{align*}
\text{Since } m &= m - \delta m \\
\text{Since } b &= b - \delta b \\
\text{Since } m^1 &= m^0 - \text{Error} \cdot X \cdot \text{Learning Rate} \\
b^1 &= b^0 - \text{Error} \cdot \text{Learning Rate}
\end{align*}
\]
Here $x_1$ and $x_2$ are normalized attribute values of data.

- $y$ is the output of the neuron, i.e., the class label.
- $x_1$ and $x_2$ values multiplied by weight values $w_1$ and $w_2$ are input to the neuron $x$.
- Value of $x_1$ is multiplied by a weight $w_1$ and values of $x_2$ is multiplied by a weight $w_2$.
- Given that
  - $w_1 = 0.5$ and $w_2 = 0.5$
  - Say value of $x_1$ is 0.3 and value of $x_2$ is 0.8,

So, weighted sum is:

$$\text{sum} = w_1 \times x_1 + w_2 \times x_2 = 0.5 \times 0.3 + 0.5 \times 0.8 = 0.55$$
The neuron receives the weighted sum as input and calculates the output as a function of input as follows:

\[ y = f(x), \text{ where } f(x) \text{ is defined as} \]

\[ f(x) = 0 \text{ when } x < 0.5 \]
\[ f(x) = 1 \text{ when } x \geq 0.5 \]

For our example, \( x \) (weighted sum) is 0.55, so \( y = 1 \).

That means corresponding input attribute values are classified in class 1.

If for another input values, \( x = 0.45 \), then \( f(x) = 0 \),
so we could conclude that input values are classified to class 0.
The neuron is the basic information processing unit of a NN. It consists of:

1. A set of **links**, describing the neuron inputs, with **weights** $W_1, W_2, \ldots, W_m$

2. An **adder** function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$u = \sum_{j=1}^{m} w_j x_j$$

3. **Activation function**: for limiting the amplitude of the neuron output.

$$y = \varphi(u + b)$$
Why We Need Multi Layer?

- Linear Separable:
  - $x \lor y$

- Linear inseparable:
  - $x \land y$

- Solution?
A Multilayer Feed-Forward Neural Network

Output nodes

Hidden nodes

Input nodes

Output Class

$O_k$

$O_j$

Input Record: $x_i$

Network is fully connected

$w_{ij}$ - weights

$w_{jk}$ - weights

Network is fully connected
The inputs are fed simultaneously into the input layer.

The weighted outputs of these units are fed into hidden layer.

The weighted outputs of the last hidden layer are inputs to units making up the output layer.
The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.

A network containing two hidden layers is called a three-layer neural network, and so on.

The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer.
- Back Propagation learns by iteratively processing a set of training data (samples).

- For each sample, weights are modified to minimize the error between network’s classification and actual classification.
Backpropagation Example

\[ net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1 \]

\[ net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775 \]

We then squash it using the logistic function to get the output of \( h_1 \):

\[ out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992 \]

Carrying out the same process for \( h_2 \) we get:

\[ out_{h2} = 0.596884378 \]

Backpropagation Example

Error Calculation

\[ E_{total} = \sum \frac{1}{2} (target - output)^2 \]

For example, the target output for \( o_1 \) is 0.01 but the neural network output 0.75136507, therefore its error is:

\[ E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083 \]

Repeating this process for \( o_2 \) (remembering that the target is 0.99) we get:

\[ E_{o2} = 0.023560026 \]

The total error for the neural network is the sum of these errors:

\[ E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109 \]
Backward Pass

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial \text{out}_o} \ast \frac{\partial \text{out}_o}{\partial \text{net}_o} \ast \frac{\partial \text{net}_o}{\partial w_5}
\]

\[
E_{o1} = \frac{1}{2}(\text{target}_o - \text{out}_o)^2
\]

\[
E_{\text{total}} = E_{o1} + E_{o2}
\]

\[
E_{\text{total}} = \frac{1}{2}(\text{target}_o - \text{out}_o)^2 + \frac{1}{2}(\text{target}_o - \text{out}_o)^2
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_o} = 2 \ast \frac{1}{2}(\text{target}_o - \text{out}_o)^{2-1} \ast -1 + 0
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_o} = -(\text{target}_o - \text{out}_o) = -(0.01 - 0.75136507) = 0.74136507
\]
Backward Pass

\[ out_{o1} = \frac{1}{1 + e^{-net_{o1}}} \]

\[ \frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1} (1 - out_{o1}) = 0.75136507 (1 - 0.75136507) = 0.186815602 \]

Finally, how much does the total net input of \( o1 \) change with respect to \( w_5 \)?

\[ net_{o1} = w_5 \ast out_{h1} + w_6 \ast out_{h2} + b_2 \ast 1 \]

\[ \frac{\partial net_{o1}}{\partial w_5} = 1 \ast out_{h1} \ast w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992 \]

Putting it all together:

\[ \frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial net_{o1}} \ast \frac{\partial net_{o1}}{\partial w_5} \]

\[ \frac{\partial E_{total}}{\partial w_5} = 0.74136507 \ast 0.186815602 \ast 0.593269992 = 0.082167041 \]

\[ w_5^+ = w_5 - \eta \ast \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 \ast 0.082167041 = 0.35891648 \]
Backward Pass – Hidden Layer

\[
\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} \times \frac{\partial \text{out}_{h1}}{\partial \text{net}_{h1}} \times \frac{\partial \text{net}_{h1}}{\partial w_1}
\]

\[
\frac{\partial E_{\text{total}}}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}}
\]

\[
E_{\text{total}} = E_{o1} + E_{o2}
\]
Demonstration

Approximation using Multi-Layer Neural Network

<table>
<thead>
<tr>
<th>Data</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

Settings:
- Learning rate: 0.1
- Momentum: 0
- Sigmoid's alpha value: 2
- Neurons in first layer: 20
- Iterations: 1000
- Current iteration
  - Iteration: 
  - Error:
Validation

- Cross Validation (10-fold)
  - Randomly divide training data set in 10 segments
  - Train with 9 and test on remaining 1
  - Repeat the procedure 10 times
  - Training sample should be balanced
    - Nearly equal number of all possible classes

- Leave-1-out Validation: same as above, we take one sample as test set and train with the rest
Case Study – Classification for Target Prediction
Comparing F1 scores for different Classifiers

- Discriminant Analysis
- Support Vector Machine
- Naive Bayes
- Neural Network

EyeGaze
HeadTracking
HandTracking
Neural Network

Back Propagation Neural Network
Algorithm – Neural Network

- For every change in position of pointer in screen
  - Calculate angle of movement
  - Calculate velocity of movement
  - Calculate acceleration of movement
- Run Neural Network with Angle, Velocity and Acceleration
- Check output
- If output predicts homing phase
  - Find direction of movement
  - Find nearest target from current location towards direction of movement
Evaluation Criteria

- **Availability**: In how many pointing tasks the algorithm makes a successful prediction.

- **Accuracy**: Percentage of correct prediction among all predictions

- **Sensitivity**: How quickly an algorithm can detect intended target
Results

Target Prediction Availability for Mouse

Accuracy for Mouse

Sensitivity for Mouse

Target Prediction Availability for Head Tracker

Accuracy for Head Tracker

Sensitivity for Head Tracker

Target Prediction Availability for Eye-gaze Tracker

Accuracy for Eye-gaze Tracker

Sensitivity for Eye-gaze Tracker
Demonstration
Case Study 2 – Gaze Controlled HUD / HMD
Existing eye trackers are developed for desktop computing environment where
- Tracker is attached below display
- Display is a flat screen
- We used eye tracker to track eyes on windshield
- Display was away from eye tracker
- Display surface was not flat like a computer screen
Compared ML systems to convert eye gaze coordinates to screen coordinates on windshield

Set up Linear Regression and Backpropagation Neural Network Models for
- Predicting x-coordinate in screen from x coordinate recorded by gaze tracker
- Predicting x-coordinate in screen from x and y coordinates recorded by gaze tracker
- Predicting y-coordinate in screen from y coordinate recorded by gaze tracker
- Predicting y-coordinate in screen from x and y coordinates recorded by gaze tracker

Compared $R^2$ and RMS error

Neural Network model worked better than Linear Regression
Implementation

- Transform raw gaze coordinates geometrically for inverted image
- Run calibration program to train neural net
- Filter predicted gaze coordinates
- Correct offset based on initial calibration
- Activate target nearest to predicted gaze location
User Study

- Set up HUD in a Toyota Etios Car
- Collected data from 9 users
- Undertook standard pointing and selection task following ISO 9241 standard
- Collected 81 pointing tasks
Results

- Median pointing and selection time 2.1 secs
- Average selection time was 1.8 secs and standard deviation was 1.1 secs
HMD

Activation Function for hidden layer = \frac{1}{1 + \exp(-x)}

Normalized x, y coordinates

Screen_X/Screen_Y
Comparison

R² values for predicting x

R² values for predicting y

RMS error values for predicting x

RMS error values for predicting y
Video

AERO INDIA 2019
The Runway to a Billion Opportunities
Results

Response Times

Flight Altitude Deviation

Deviation from Designated Path
Cluster Analysis
Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.

- Intra-cluster distances are minimized.
- Inter-cluster distances are maximized.
What is Cluster Analysis?

- **Cluster**: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- **Cluster analysis**
  - Grouping a set of data objects into clusters
- **Clustering is unsupervised classification**: no predefined classes
- **Clustering is used**:
  - As a stand-alone tool to get insight into data distribution
    - Visualization of clusters may unveil important information
  - As a preprocessing step for other algorithms
    - Efficient indexing or compression often relies on clustering
Applications of Clustering

- Pattern Recognition
- Image Processing
  - cluster images based on their visual content
- Bio-informatics
- WWW and IR
  - document classification
  - cluster Weblog data to discover groups of similar access patterns
Distance metrics are normally used to measure the similarity or dissimilarity between two data objects. The most popular conform to *Minkowski* distance:

\[
L_p(i,j) = \left( |x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \ldots + |x_{in} - x_{jn}|^p \right)^{1/p}
\]

where \( i = (x_{i1}, x_{i2}, \ldots, x_{in}) \) and \( j = (x_{j1}, x_{j2}, \ldots, x_{jn}) \) are two \( n \)-dimensional data objects, and \( p \) is a positive integer.

If \( p = 1 \), \( L_1 \) is the **Manhattan (or city block)** distance:

\[
L_1(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}|
\]
If $p = 2$, $L_2$ is the Euclidean distance:

$$d(i, j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_n} - x_{j_n}|^2)}$$

**Properties**

- $d(i, j) \geq 0$
- $d(i, i) = 0$
- $d(i, j) = d(j, i)$
- $d(i, j) \leq d(i, k) + d(k, j)$

Also one can use weighted distance:

$$d(i, j) = \sqrt{(w_1 |x_{i_1} - x_{j_1}|^2 + w_2 |x_{i_2} - x_{j_2}|^2 + \ldots + w_n |x_{i_n} - x_{j_n}|^2)}$$
Major Clustering Approaches

- **Partitioning algorithms**: Construct random partitions and then iteratively refine them by some criterion.
- **Hierarchical algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion.
- **Density-based**: based on connectivity and density functions.
- **Grid-based**: based on a multiple-level granularity structure.
- **Model-based**: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.
Partitioning method: Construct a partition of a database $D$ of $n$ objects into a set of $k$ clusters

- **$k$-means** (MacQueen’67): Each cluster is represented by the center of the cluster
- **$k$-medoids** or PAM (Partition around medoids) (Kaufman & Rousseeuw’87): Each cluster is represented by one of the objects in the cluster
K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, $K$, must be specified
- The basic algorithm is very simple

1: Select $K$ points as the initial centroids.
2: repeat
3: Form $K$ clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: until The centroids don’t change
Limitations of K-means

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-spherical shapes

- K-means has problems when the data contains outliers. Why?
The \textit{K-Medoids} Clustering Method

- Find \textit{representative} objects, called \textit{medoids}, in clusters
- \textit{PAM} (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - \textit{PAM} works effectively for small data sets, but does not scale well for large data sets
- \textit{CLARA} (Kaufmann & Rousseeuw, 1990)
- \textit{CLARANS} (Ng & Han, 1994): Randomized sampling
- Use distance matrix as clustering criteria. This method does not require the number of clusters $k$ as an input, but needs a termination condition.

![Hierarchical Clustering Diagram]

**Hierarchical Clustering**

- **Agglomerative (AGNES)**
- **Divisive (DIANA)**

![Diagram of Clustering Steps]

Step 0 | Step 1 | Step 2 | Step 3 | Step 4
---|---|---|---|---
$0$ | $1$ | $2$ | $3$ | $4$

(a) b
b
((c d e)
(c d e)
d e)

(a b)
(a b c d e)

63
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

E.g., level 1 gives 4 clusters: \{a,b\},\{c\},\{d\},\{e\},
level 2 gives 3 clusters: \{a,b\},\{c\},\{d,e\}
level 3 gives 2 clusters: \{a,b\},\{c,d,e\}, etc.
What happens when we can not specify the optimum number of clusters beforehand

Can we find the optimum number of clusters?

Two methods can return overlapping clusters
  - Fuzzy c-means
  - EM Clustering algorithm
Fuzzy c-means

- Place a set of cluster centres
- Assign a fuzzy membership to each data point depending on distance
- Compute the new centre of each class
- Termination is based on an objective function
- Returns cluster centres and membership values of each data point to each cluster
Assume data came from a set of Gaussian Distribution

Assign data points to distributions and find Expected probability

Update mean and std dev of distributions to Maximize probabilities
1) E Step : Evaluate responsibilities using the current parameters values

2) M Step : Re-estimate the parameters using the current responsibilities

3) Evaluate the log likelihood

\[
\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}.
\]

\[
\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n
\]

\[
\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (x_n - \mu_k^{new}) (x_n - \mu_k^{new})^T
\]

\[
\pi_k^{new} = \frac{N_k}{N}
\]

\[
\ln p(X | \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}
\]
Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - **External Index:** Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - **Internal Index:** Used to measure the goodness of a clustering structure without respect to external information.
    - Sum of Squared Error (SSE)
  - **Relative Index:** Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
  - **Sometimes these are referred to as criteria instead of indices**
    - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.
Cluster validity indexes are used to evaluate the fitness of partitions produced by clustering algorithms.

- Entropy values are also used to evaluate the fitness of partitions.
- XB indexing is one type of validity function proposed by Xie and Beni.
- Ratio between compactness measure and separation measure.
Classification and Clustering
  - Decision tree and neural network for classification
  - Linear Regression
  - Cross validation
  - Hierarchical & K-means clustering
  - Soft Clustering
  - Cluster Validation Index
  - Case studies on IUI