ML- CHAPTER

Chapter (1)

ML: is the field of study that gives the computer the ability to learn without explicitly programmed.

Taxonomy of ML

* Supervised learning : guided
* Unsupervised learning : unguided
* Reinforcement learning : learn and improved from previous knowledge. (feedback)

Regression Vs Classification (Supervised)

- Regression → Continuous Variable
- Classification → discrete Variable

Clustering (unsupervised learning)

- K-means clustering

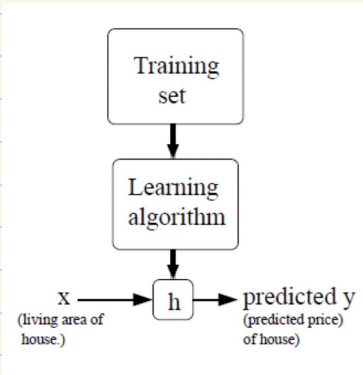
Chapter (2)

Linear Regression

* the aim is to predict the price of other houses, as a function of the size of their living area

∴ given a training set ; we aim to learn a function $h: X \rightarrow Y$

X : the space of input value
Y : the space of output value
h : hypothesis



$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_i : are the parameters (weights)
i : number of features.

for problem with two features : hypothesis will be:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

for multiple input features : (multivariate linear Regression):

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x$$

Cost Function : It takes an average difference of all the results of the hypothesis with input x's and the actual output y's.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta (x^{(i)}) - y^{(i)} \right)^2$$

→ difference b/w the predicted value and the actual value

its called :

* square error function * Mean square error (MSE) or least square error

Gradient Descent.

* take the derivative of the cost function
* it will give us the direction to move towards
* go the opposite direction
* the size of each step is determined by a parameter, which is called learning rate.

gradient descent algorithm

* start with some "initial guess" for θ
* repeat the convergence:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

* where $j = 0, 1$ represent feature index number
* α is the learning rate
* at each iteration j, one should simultaneously update the parameters $\theta_1, \theta_2 \cdots \theta_n$

* Batch gradient descent : the parameter are updated based on all example in the dataset.
* stochastic gradient descent: (incremental gradient descent) : the parameter are updated according to the gradient descent with respect to each training example individually.

* Learning Rate: α : the size of each step is determined by it.
  * if α is too small, gradient descent can be slow
  * if α is too large, gradient descent can over shoot the minimum.

Debugging gradient descent.

- Make a plot with *number of iterations* on the x-axis.
- Plot the cost function, J(θ) over the number of iterations of gradient descent.
- If J(θ) ever increases, then you probably need to decrease α.

Automatic convergence test.

- Declare convergence if J(θ) decreases by less than E in one iteration, where E is some small value such as 10^{-3}.
- However in practice it's difficult to choose this threshold value.

The Normal Equation

* is to minimise j explicitly by taking the derivative of j with respect to θ's and and setting them to zero.

$$\nabla_\theta j(\theta) = \nabla_\theta \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

Gradient descent vs. normal equations

Gradient Descent	Normal Equation *
Need to choose learning rate α	No need to choose learning rate α
Needs many iterations	No need to iterate
Works well when n is large	Slow if n is very large (in practice, n exceeds 10,000)
$O(kn^2)$	$O(n^3)$, need to calculate inverse of $X^T X$
Feature scaling is important	**no need** to do feature scaling
May find suboptimal solution	Finds unique optimal solution

Scikit learn uses normal equations in Linear regression implementation

(chapter 3)

Binary classification

↓ Binary means 0 or 1

* target variable can only take two values: 0 or 1
* Binary classification problem is called logistic function or sigmoid function.

$$g'(z) = \frac{d}{dz} \frac{1}{1+e^{-z}} = g(z)(1-g(z))$$

* probability:

$$p(y=1 \mid x;\theta) + p(y=0 \mid x;\theta) = 1$$

Likelihood of parameters:

$$L(\theta) = p(\vec{y} \mid x;\theta)$$

Objective function (log likelyhood)

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log (1-h(x^{(i)}))$$

Gradient ascent

$$\theta = \theta + \alpha \nabla_\theta l(\theta)$$

* to maximise the likelihood we can use gradient ascent.

stochastic gradient ascent rule:

$$\theta_j = \theta_j + \alpha \left(y^i - h_\theta(x^i)\right) x_j^i$$

Newton's Method

∴ is for finding the roots of non linear equations $f(\theta) = 0$.

$$\theta = \theta - \frac{f(\theta)}{f'(\theta)}$$

Newton-Raphson method.

* generalisation of newtons method to multidimensional setting

$$\theta = \theta - H^{-1} \nabla_\theta l(\theta)$$

H^{-1}: hessian

$\nabla_\theta l(\theta)$: vector of partial derivatives

fisher scoring method

* when Newton's method is applied to maximise the logistic regression log likelihood function $l(\theta)$, the resulting method is also called fisher scoring.

chapter (4)

Tree-Based method and Ensemble learning

* **Tree based:-** method for classification and regression. this involve stratifying or segmenting the predictor space into a number of simple regions (rectangles).
* simple * fast * useful
* works well with both categorical and quantitative features.

Terminology for decision Trees

* terminal Nodes
* leaf Nodes h(x)
* are typically upside down
* internal Nodes

Classification of Decision Tree

* Decision tree : are trained in greedy, recursive fashion, downward from the root.
* left subtree is only those point satisfying $x_j < v$ (smaller)
* Right subtree is only those point satisfying $x_j \geq v$ (bigger)

How to choose best split

step 1 : Try all splits. (All features and all splits within a feature).
step 2 : for a set S let J(S) be the cost of S.
step 3 : Choose split that minimise $J(S_L) + J(S_R)$; or
step 4 : choose split that minimise the weighted average:

$$\frac{|S_L| J(S_L) + |S_R| J(S_R)}{|S_L| + |S_R|}$$

How to choose cost J(S)
Misclassification rate

* label S with the class C that label the most example in S.
* J(S) : number of example in S not in class C

Entropy

- We can use a cost function based on the information theory by measuring the entropy.
- Let Y be random class variable, and suppose $P(Y=c) = P_c$.
- The surprise of Y being class C is: $-\log_2 P_c$. (nonnegative)
  - event with probability 1 gives us zero suprise.
  - event with probability 0 gives us infinite suprise.

$$H(S) = -\sum_c P_c \log_2 P_c$$

["Not (-) some ($\sum_c$) c $P_c \log_2 P_c$"]

Information Gain

- Choose split that maximise the information gain:

$$I(S) = H(S) - H_{after}$$

where:

$H(S)$: entropy

H_{after}: is the average entropy after split.

$$H_{after} = \frac{|S_L| H(S_L) + |S_r| H(S_r)}{|S_L| + |S_r|}$$

Gini impurity

- another way to asses the quality of a split is Gini Impurity (strictly concave)
- it measures how often a randomly chosen element from the set would be incorrectly labeled if it where randomly labeled according to the distribution of labels in subset.

$$G(S) = \sum_c P_c (1 - P_c)$$

(G of S = Sum c P_c of one minus P_c)

Stopping Criteria

* deep decision tree = overfitting.
* shallow decision tree = underfitting.
* heuristic to consider to decide when to stop splitting:
  * fixed depth:
  * Node purity
  * Information gain Criteria.

Regression decision tree

* the goal is to find the regions (boxes) $R_1, \dots, R_J$ that minimize the residual sum of squares (RSS), given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where:

$\hat{y}_{R_j}$ is the mean response for the training observation within the j^{th} box.

Advantage of Trees

* Trees are very easy to explain to people.
* trees can be displayed graphically.
* Can easily handle qualitative predictors

Dis advantage of trees

* trees generally do not have the same level of predictive accuracy.
* they can be non-robust.

Ensemble learning

weak learners → "building block"

* Ensemble method :- is an approach that combines many simple models in order to obtain a single and potentially very powerfull model.
* Bagging * Random forest * Boosting.

* **Bagging:** is a general-purpose procedure for reducing the variance of a statistical learning method.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

out-of-bag error estimation

* each bagged tree, make use of $\frac{2}{3}$ of the observation. So; the remaining $\frac{1}{3}$ of the observation not used to fit the given bagged tree are referred to as the out-of-bage (OOB) observation.

Random forest: provide an improvement over bagged trees by decorrelating the trees.

Boosting:

* Bagging involve creating multiple copies of original training data set, using the bootstrap, fitting a seperate decision tree to each copy and then combining all the trees in order to create a single predictive model. Boosting works in similar way, But each tree is grown using information from previous grown tree.

Adaboost: adaptive boosting; is a method for binary classification.

Boosting for Regression

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{b}(x)$$

Tuning parameters for boosting

* Number of trees (B)
* shrinkage parameter (lambda)
* number of splits (d)

Chapter (5)

Support vector machine (SVM)

* SVM :- is a supervised ML algorithm used for both classification and regression tasks
* Hyper plane : is the decision boundary used to seperate data points of diffrent class in a feature space.
* Maximal margin classifier :- is the biggest gap or margin b/w two classes.
* non-seperable data :- is when a data are not seperable by a linear boundary.
* Support vector :- are the closest data points to the hyperplane.
* Margin :- refers to the distance b/w support vector and the hyperplane.
* Kernel :- is a mathematical function used in SVM to map input data (low D) to a higher dimensional feature space.
* Hard margin :- refers to the maximum-margin hyperplane that perfectly seperate the data points of different classes without any misclassifications.
* Soft margin :- when a data contain outliners or is not perfectly separable, SVM uses the soft margin technique. which introduce slack variable.
* C parmeter :- is a regularisation term that balances margin maximisation and the penalty for mis classification.
* Hinge loss :- is a common loss function in SVM. it penalise misclassification point or margin violation.
* Dual problem :- it involves solving for the Lagrange multipliers associated with the support vectors.
* Slack Variable :- tell us where the observation is located, relative to hyperplane and relative to the margin.

Kernel function in SVM

1- linear : $K(w,b) = w^T x + b$

2- polynomial : $K(w,x) = (\gamma w^T x + b)^N$

3- Gaussian RBf : $K(w,x) = \exp(-\gamma ||x_i - x_j||^n)$

4- Sigmoid : $K(x_i, x_j) = \tanh(\alpha x_i^T x_j + b)$

Radial Kernel

* it controls Variance by squashing down most dimensions severely.

How Radial Kernels works

- If a given test observation $x^* = (x_1^*, \ldots, x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance,
- Then $\sum_{j=1}^{p} (x_j^* - x_{ij})^2$ will be large,
- and so $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$. will be tiny.
- This means that, x_i will play virtually no role in f(x).

Advantages of Kernels

* Computational

SVM in More than 2 Classes

* One versus all (OVA): each class versus the rest.
* One versus one (OVO): classify x^* to the class that wins the most pairwise competitions.

Different b/w SVM and logistic Regression

1. SVM tries to find the "best" margin that separates the classes and this reduces the risk of error in data. while RL does not.
2. SVM works well with structured and semi-structured data like text and images while RL works with already identified independent variables.
3. SVM is based on geometrical properties of the data while LR is based on statistical approaches.
4. the risk of overfitting is less in SVM, while LR is vulnerable to overfitting

Which to use: SVM or LR?

* when classes are (nearly) separable, use SVM.
* If you wish to estimate probabilities, use LR
* for Nonlinear boundaries, Kernel, use SVM.

How does SVM algorithm works

1. Problem Setup:

- SVM is given labeled training data: (x_i, y_i), where:
  - x_i represents the features of a data point.
  - $y_i \in \{-1, 1\}$ indicates the class label (for binary classification).
- The goal is to classify new, unseen data points correctly.

2. Linear Separability:

- **Hyperplane**: A linear decision boundary in the form $w \cdot x + b = 0$, where:
  - w is the weight vector (defining the orientation of the hyperplane).
  - b is the bias term (shifts the hyperplane).
  - x is the feature vector.
- Data points satisfy:
  - $w \cdot x + b > 0$: Class $+1$
  - $w \cdot x + b < 0$: Class -1

3. Margin Maximization:

- SVM aims to find the hyperplane that maximizes the **margin**, which is the distance between the hyperplane and the nearest data points (support vectors).
- Margin = $\frac{2}{\|w\|}$, where $\|w\|$ is the norm of w.

Optimization Problem:

- Minimize: $\frac{1}{2}\|w\|^2$ (to maximize the margin).
- Subject to:
  - $y_i(w \cdot x_i + b) \geq 1$ for all i.

This ensures all points are classified correctly and lie on the correct side of the margin.

4. Non-Linearly Separable Data:

- If data isn't linearly separable, SVM uses the **slack variable** ξ_i to allow some misclassification:
  - $y_i(w \cdot x_i + b) \geq 1 - \xi_i$
  - Add a penalty term $C\sum \xi_i$ to the objective function, where C is the regularization parameter.
- **Kernel Trick**:
  - For highly complex data, SVM uses kernels to transform the input space into a higher-dimensional space where a linear hyperplane can separate the classes.
  - Examples of kernels:
    - **Linear**: $K(x, z) = x \cdot z$
    - **Polynomial**: $K(x, z) = (x \cdot z + c)^d$
    - **RBF (Gaussian)**: $K(x, z) = \exp(-\|x - z\|^2/(2\sigma^2))$

Limitations:

- Computationally expensive for large datasets.
- Sensitive to the choice of kernel and hyperparameters (C, σ).

Advantages of SVM:

- Works well in high-dimensional spaces.
- Effective for small datasets with a clear margin of separation.
- Can handle non-linear data using kernels.

5. Predictions:

- Once trained, the SVM predicts the class of a new data point x using:
  $f(x) = \text{sign}(w \cdot x + b)$

For non-linear kernels, $f(x)$ depends on the kernel function and the support vectors.

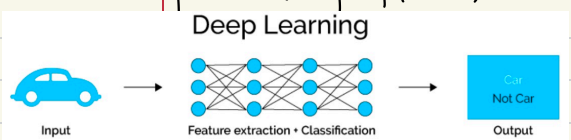
Chapter (6)

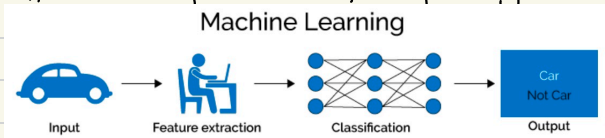
Deep Learning

DL :- is a branch of ML that is Based on ANN architecture.

Why DL is useful?

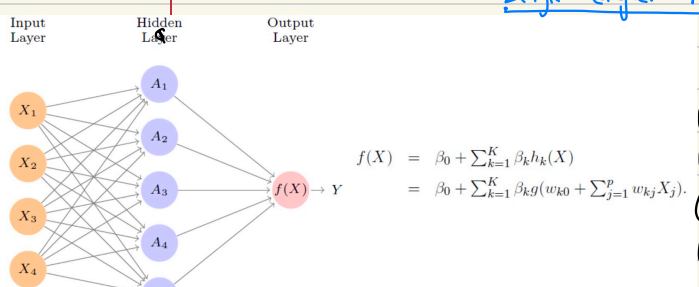
* Can learn both supervised and Un supervised.
* Effective end-to-end 'join System learning
* utilise large amounts of training data
* Learned features are easy to adapt and fast to learn.
* provide very flexible, Universal, learnable framework for representing world.





* **Representational power:** NN with atleast one hidden layer are Universal approximators.
* **perceptron:** is the basic processing element. it has input that may come from the environment or may be the output of other perceptrons.

Single Layer Neural Network



Steps:

1. initialised with random weight.
2. present a training instance
3. feed it through to get output
4. Compare with target output.
5. Adjust weight based on error
6. present a training instance

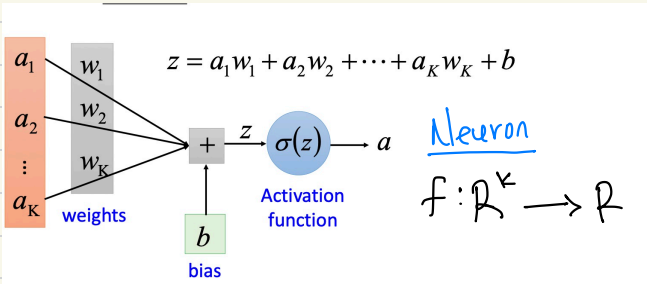
Repeat from this steps till Okay

Some points

* if f(x) is a non-linear, a network with 1 hidden layer can, in theory, learn perfectly any classification problem.
* if f(x) is linear, the NN can only draw straight decision boundary
* NN use nonlinear f(x) so they can draw complex boundries, but keep the data unchanged.
* SVM only draw straight lines, but they trasform the data first in a way that makes that OKay.

Example: Handwriting Digit Recognition.

Elements of Neural Network



* **Neuron**: the basic unit that recieve inputs each neuron is governed by a threshold and an activation function.
* **Connections**: link between neurons that carry information, regulated by weight and bais.
* **weight & biases**: these parameter determines the influence of a connection
* **propagation function**: Mechanism that help process and transfer data across layers of neurons.
* **learning Rule**: the method that adjust the weight and bias overtime to improve accuracy.

Soft max layer

* is an activation function commonly used in NN for multiclassification problems.
* **Soft max function**: is a mathematical function that convert a vector of raw prediction scores (often called logits) from the NN to probabilities.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

Activations functions

* **Sigmoid function**: takes a real-valued number and squashes it into the range b/w 0 and 1.
* **Tanh function**: takes a real-valued number and squashes it into the range b/w 1 and -1.
* **ReLU (Rectified Linear Unit)**: takes a real-valued number and thresholds it at zero
* **Linear function**: means that the out put signal is proportional to the input signal to the neuron.

Training NNs

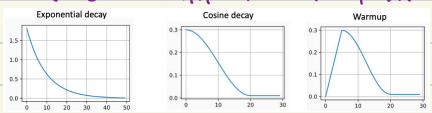
1- Data preprocessing: help convergence during training
   * mean subtraction: to obtain zero-centred data
   * Normalization:
      * divide each feature by its standard deviation.
      * or scale the data within the range $[0,1]$ or $[-1,1]$.
2- To train a NN: Set the parameters θ such that for a training subset of images, the corresponding elements in the predicted output have maximum value.
3- Define a loss function $L(\theta)$ that calculate the diffrence (error) b/w model prediction and the true label.
4- for training set of N images, Calculate the total loss overall all image
   * find the optimal parameter θ that minimise the total loss $L(\theta)$

Loss functions

classification task	Regression task
* Training example: grand truth class label y_i	* Training example: grand truth output value
* out put layer: Softmax activation	* output layer: Linear or sigmoid activation
* loss function: Cross entropy.	* loss function: MSE / MAE

* Gradient descent Algorithm: is an optimisation algorithm that minimise the cost function of a neural network modelling during training.
* Back propagation: is an algorithm that back-propagate the error from the output nodes to the input nodes. it compute the gradient descent of the loss function with respect to the network weight.
* Batch gradient descent: to update the model parameter values like weight and bias, the entire training data set is used to compute the gradient and update the parameter at each iteration.
* stochastic gradient descent (SGD): only one training example is use to compute the gradient and update the parameter at each iteration.
* Mini-batch gradient descent: a small batch of training example is used to compute the gradient and update the parameter at each iteration.
* Momentum-based gradient descent: it incorporates information from the previous weight updates to help the algorithm converge more quickly to the optimal solution.
* Adam: it combine the benefits of momentum, Adagrad, the learning rate is adaptively adjusted for each parameter based on the moving average of the gradient.

* Learning rate :- the gradient tell us the direction in which the loss has the steepest rate of increase, but it does not tell us how far along the opposite direction we should step.
* Learning rate scheduling :- is applied to change the value of the learning rate during training.
  * Annealing : is reducing the learning rate over time (also learning rate decay)
  * Warmup : is gradually increasing the learning rate initially, and afterward let it cool down until the end of the training.



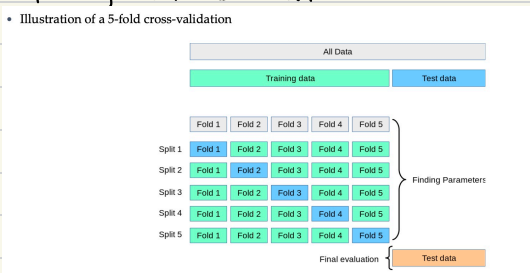
Generalization

* Underfitting :- the model is too simple to capture the data complexity.
  high bia, low variance.
* Overfitting :- when the model does not make accurate predictions on testing data.
  high variance low bias
* weight decay : is a term that penalises large weight is added to the loss function
* Drop out :- Randomly drop unit (along with their connections) during training.
* Early stopping :- stop when the validation accuracy (or loss) has not improved after n epochs.
* Batch normalisation layers : they calculate the mean μ and variance σ of a a batch data, and normalised the data x to a zero mean and unit variance.

Hyper parameter tuning

* number of layers, and number of neutrons per layers
* Initial learning rate
* learning rate decay schedule
* Optimiser type
* loss function
* Regularisation parameters
* Batch size
  Activation function
* Grid Search : check all values in range with a step value
* Random search : Randomly sample values for the parameter.
* Bayesian hyper parameter optimization : is an active area of research.

K-fold cross validation :- for hyper parameter tuning is common when the size of the training data is small.



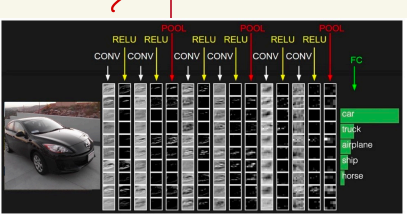
* Ensemble learning: is training multiple classifier seperatly and combining their prediction.

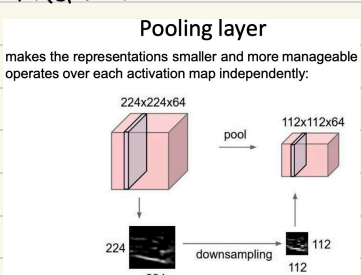
Convolutional Neural Network (CNNs)

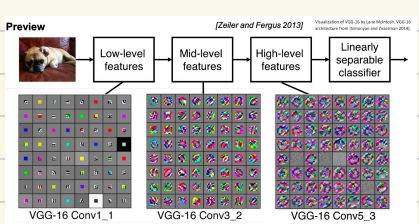
* They were primarily design for image data. CNN use a convolutional operators for extracting data features.

How CNN works

* the CNN build up an image in a hierarchical fashion.
* Edges and shapes (local features) are recognised and pieced together to form more complex shapes. (Compound features)
* the hierarchical construction is achieved using convolution and pooling layers.
* Convolution layer :- when convolutional filter are scanned over the image, they capture useful features. (eg edge detection).
* pooling layer: reduce the spatial size of the feature map.
  * max pooling :- report the maximum output within a rectangular neighbourhood.
  * Average pooling :- report average output of a rectangular neighbourhood.
* fully connected layers : Contain neurons that connect to the entire input volume, as in ordinary Neural Networks.







Chapter (11)

Unsupervised learning (Clustering) + PCA

* Clustering :- is a technique for finding similar group in data, called Clusters.

Aspect of clustering

* clustering algorithms :
  * partitional clustering
  * Hierarchical clustering
* a distance function
* Clustering quality
  * Inter-cluster distance => maximised / minimized

K-means clustering

* is a partitional clustering. for dividing data points into K clusters. based on their features, with each other cluster represented by the mean of its data points.

K-means algorithms

1- Randomly choose K data point to be the initial Centroids, cluster centres.
2- assign each data point to the closest Centroid.
3- Re-compute the Centroids using the current cluster memberships.
4 if a convergence criterion is not met, go to (2)

1- No (or minimum) re-assignments of data points to different clusters.
2- No (or minimum) change of Centroids
3- minimum decrease in the Sum of Squared error (SSE)

$$SSE = \sum_{j=1}^{K} \sum_{x \in C_j} dist(x, m_j)^2$$

C_j = the j^{th} cluster, M_j = is the centroid of cluster C_j and $dist(x,m)$ = is the distance b/w data point x and centroid m_j.

strengths of K-means

* Simple * Efficient * both K and t are small
* Kmeans is the most popular clustering algorithm

Note: it terminate at local optimum if SSE is used. the Global optimum is hard to find due to complexity.

Weaknesses of K-means

* The user needs to specify the K. * The algorithm is sensitive to outliers.
* The algorithms is only applicable if the mean is defined.

Common ways to represent clusters

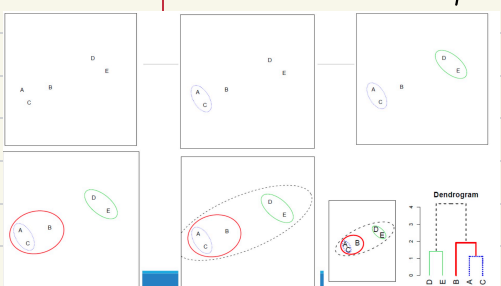
* Compute the radius and standard deviation of the cluster to determine its spread.
* Using classification model
* use frequent value to represent cluster

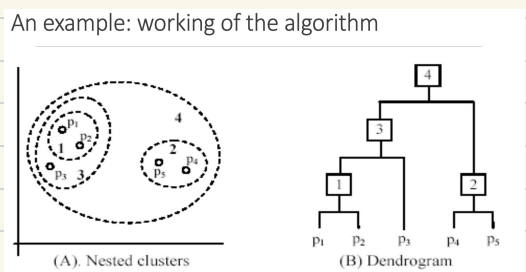
Hierarchical Clustering

* it build a hierachy of clusters either from bottom up (Agglomerative) or from the top down (divisive), where each data point starts in its cluster and pair of clusters are merge or split recursively.

Types of hierarchical clustering

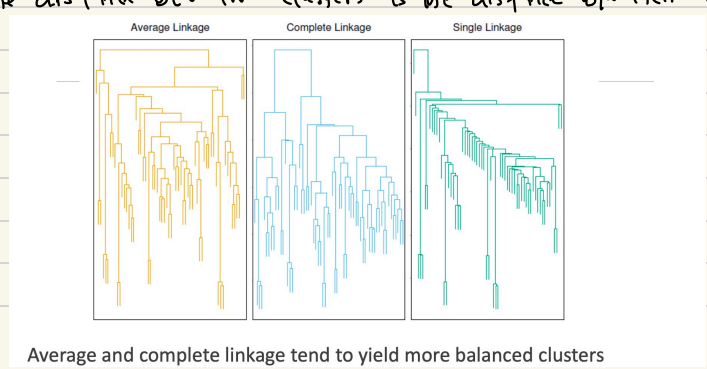
* Agglomerative (bottom up) clustering: it builds the dendrogram (tree) from bottom level and merge the most similar pair of clusters.
* Divisive (top down) clustering: it start with all data points in one cluster the root, split the root into a set of child cluster. each child cluster is recursively divided further.





Measuring the distance of two clusters

1. Single link: is distance b/w two closest data points, in the two clusters
2. Complete link: is the distance b/w to furthest data points in the two clusters.
3. Average link: the distance b/w two clusters is the average distance of all pair-wise distances b/w the data points in two clusters.
4. Centroid method: the distance b/w two clusters is the distance b/w their centroids.



Distance functions

- If $h = 1$, it is the **Manhattan distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ir} - x_{jr}|$$

- **Chebychev distance**: one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \ldots, |x_{ir} - x_{jr}|)$$

- **Weighted Euclidean distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \ldots + w_r(x_{ir} - x_{jr})^2}$$

- If $h = 2$, it is the **Euclidean distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2}$$

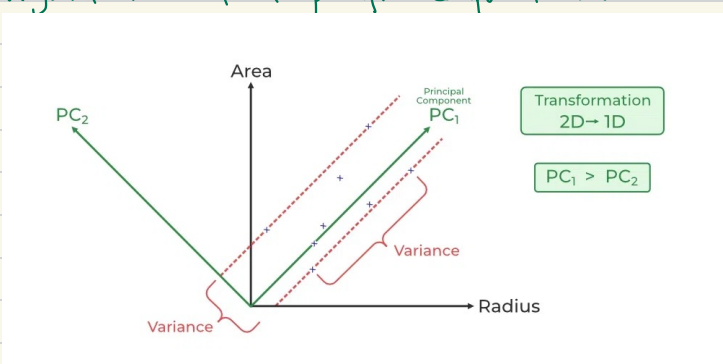
- **Squared Euclidean distance**: to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2$$

Principle Component analysis (PCA)

* PCA: is an unsupervised learning algorithm techniques used to examine the interrelation among a set of variables. The main goal is to reduce the dimensionality of data set while preserving the most important pattern or relationship b/w the variables without any prior knowledge of the target variable.

* The first principle component: of a set of features, is the normalised linear combination of the features. The first PC capture the most variation in the data, but the second PC capture the maximum variance that is orthogonal to the first principle component and so on.



Need to Read & watch more tutorials on PCA to fully understand what it means.