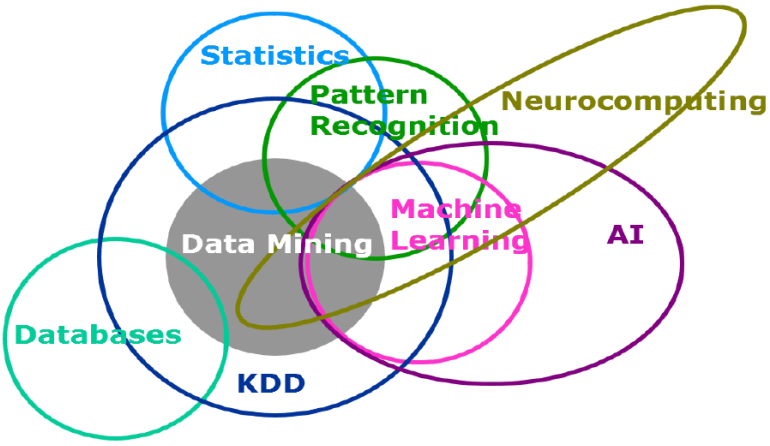
Pattern Recognition(ML)

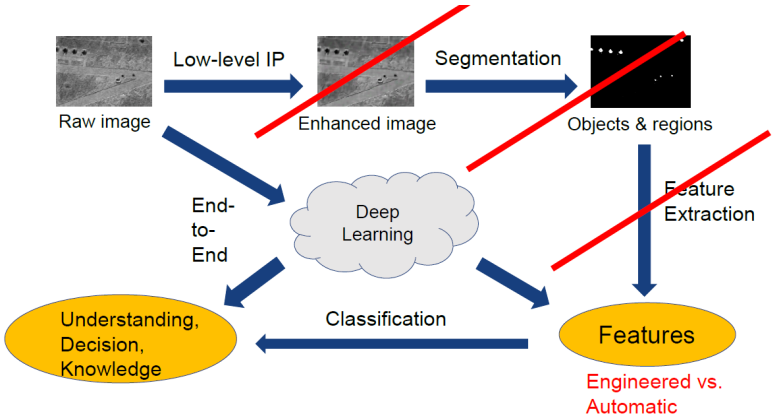
- Pattern



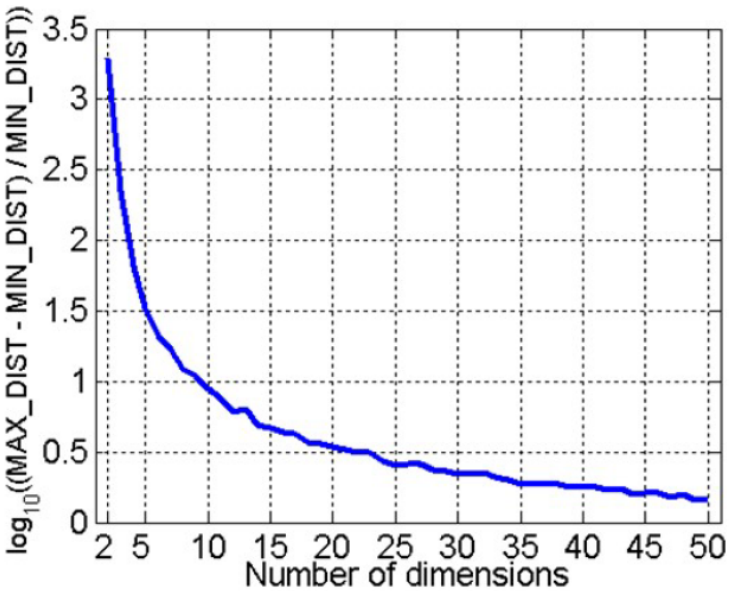
  - Pattern recognition is the automated recognition of patterns and regularities in data. Pattern recognition is closely related to artificial intelligence and machine learning, together with applications such as data mining and knowledge discovery in databases, and is often used interchangeably with these terms.
  - A pattern is represented by a set of d features, or attributes, viewed as a d-dimensional feature vector.

$$X = (x_1, x_2, \dots x_d)^\top$$

- Process
  - Data acquisition and sensing:
    - Measurements of physical variables.
    - Important issues: bandwidth, resolution , etc.
  - Pre-processing:
    - Removal of noise in data.
    - Isolation of patterns of interest from the background.
  - Feature extraction:
    - Finding a new representation in terms of features.
  - Classification/Clustering
    - Using features to learn models for different tasks.
  - Post-processing
    - Evaluation of confidence in decisions
- Deep learning paradigm

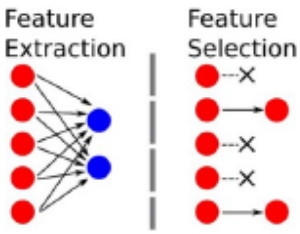


- Dimensionality
  - Problems with High Dimensionality
    - Takes up storage space
    - Hard to visualise
    - Computationally complex (long training time)
    - Result in complex models, harder to interpret
    - Cause model overfitting -poor predictions
    - Run into the Curse of Dimensionality

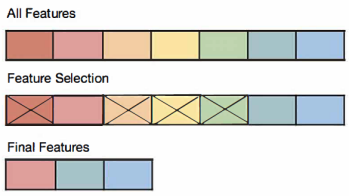


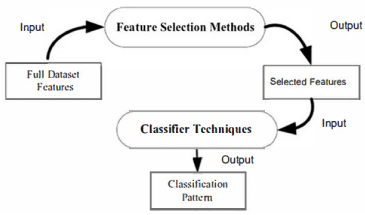
- The curse of dimensionality refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases.
- When dimensionality increases, data becomes **increasingly sparse** in the space that it occupies.
- Definitions of **density and distance** between points, which are critical for clustering and outlier detection, become less meaningful

- Dimensionality Reduction



  - Feature Selection Methods



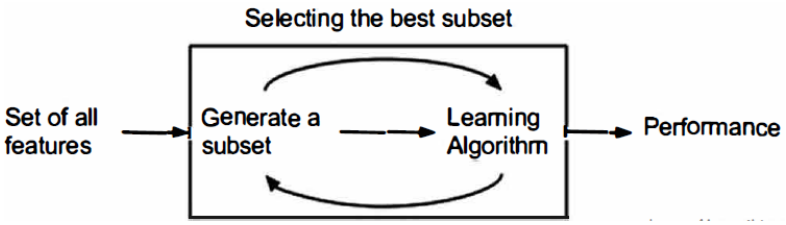


    - goal
      - remove all features that either do not bring much information or add no new information
      - The goal of feature selection in machine learning is to find the best set of features that allows one to build useful analytical models.
      - The best set of features is one that does not contain redundant and irrelevant features
    - Method
      - Filter Methods

        - Filter methods select features from a dataset based on the individual characteristics of the features.

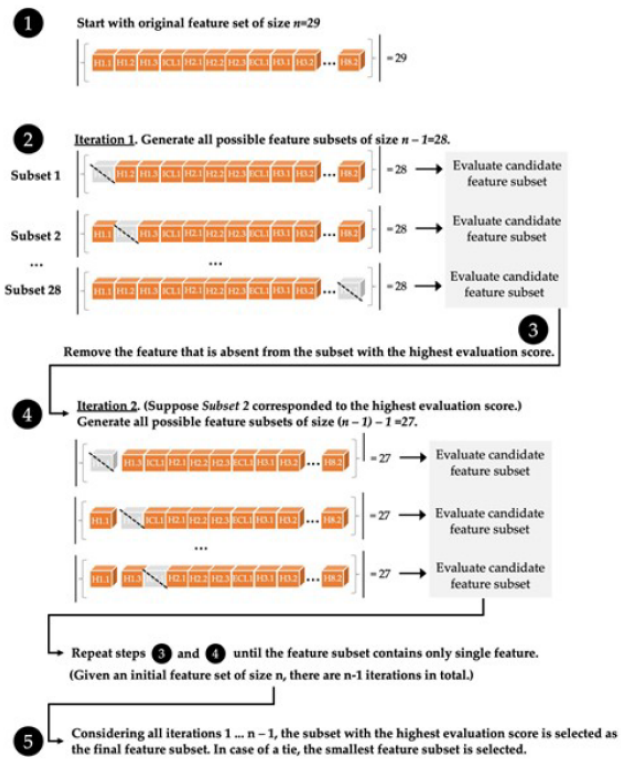
- Features are filtered out before applying machine learning and are therefore independent of any machine learning algorithm.
- These methods are simple yet powerful to quickly remove features and are generally the first method used for feature selection.
- Filtering is faster and less computationally expensive than wrapper methods.
- Key Techniques
  - Pearson's Correlation
    - Remove features that are highly corelated (r > 0.9)
  - Variance Threshold
    - Remove features that have low variability or 80% of the values are similar
  - Chi-Square test
    - The Chi-square test of independence is a statistical hypothesis test used to determine whether two categorical variables are likely to be related or not.
    - If the feature is highly related to the target variable, then we retain it.
  - f test (Anova)
    - The Analysis of Variance (ANOVA) F-statistic is used to determine whether the means from two or more data samples come from the same distribution.
    - The higher this ratio between a numerical input feature and a categorical target feature, the higher the dependence between the two and therefore the feature is more likely to be useful for model training.

- Wrapper Method



  - Wrapper methods work by systematically evaluating a subset of features using a machine learning algorithm and selects the best subset based on the algorithm's performance evaluation criteria.
  - This method employs a greedy search strategy aimed to find the best possible combination of features that results in the best machine learning performance.
  - It is computationally expensive, and can be impractical for very large datasets.

- Practically any combination of a search strategy and a machine learning algorithm can be used as a wrapper.
- The wrapper methods usually result in better predictive accuracy than filter methods
- Key technologies
  - Forward Feature Selection
    - A greedy search that iteratively finds the best new feature to add to the set of selected features.
    - Start initially with zero feature and find the one feature that maximizes a cross-validated score when an estimator is trained on this single feature. Once that first feature is selected, we repeat the procedure by adding a new feature to the set of selected features.
    - The procedure stops when the desired number of selected features is reached, as determined by the nJeatures_to_selectparameter.
  - Backward Feature Elimination

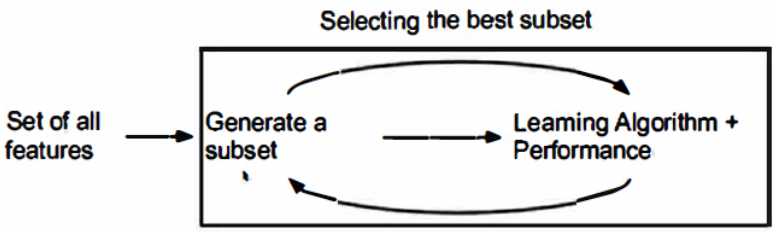


    - Follows the same idea but works in the opposite direction: instead of starting with no feature and greedily adding features, we start with all the features and greedily remove features from the set.
  - In general, forward and backward selection do not yield equivalent results. Also, one may be much faster than the other depending on the requested number of selected features: if we have 10 features

and ask for 7 selected features, forward selection would need to perform 7 iterations while backward selection would only need to perform 3.

- Recursive feature elimination (RFE)
  - A very popular technique that iteratively selects the best number of features through a given supervised learning model (estimator).
  - Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of RFE is to select features by recursively considering smaller and smaller sets of features.
  - First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef_ attribute or through a feature_importances_ attribute.
  - Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

- Embedded Method



  - Embedded methods performs the feature selection within the machine learning algorithm itself during the model training.
    - 1. Train a machine learning model
    - 2. Derive feature importance from the model when making prediction
    - 3. Remove non-important features using the derived feature importance
  - Embedded method solves issues with the filter and wrapper methods:
    - Take into consideration the interaction of features like wrapper methods do
    - Run fast like filter methods
    - More accurate than filter methods
    - Finds the feature subset for the algorithm being trained
    - Less prone to overfitting
  - Key Techniques

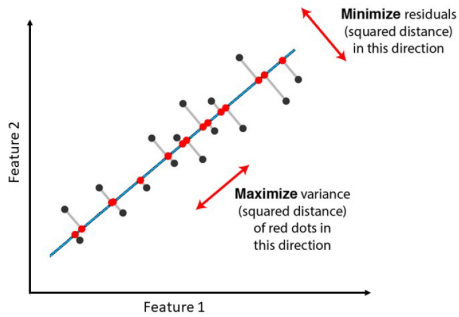
- Embedded techniques are implemented using algorithms that have their own built-in feature selection methods
- Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting
- Lasso regression (Ll regularization which adds penalty equivalent to absolute value of the magnitude of coefficients)
- Ridge regression (L2 regularization which adds penalty equivalent to square of the magnitude of coefficients)
- Random Forest
  - Random Forests or Decision tree ensembles, are useful for feature selection in addition to being effective classifiers.
  - Usage statistics about each feature are generated during the tree construction. Some features contain the most informative subset used to predict a target class.
  - Example: Build a large forest (2,000) of shallow trees (2 or 3 levels). If a feature is selected very often as the best split, then it is very likely to be an informative feature that we should keep.
  - Importance score is calculated as the number of times that the feature was selected for the split, divided by the number of times in which it was a candidate. Important features are those with the highest scores.

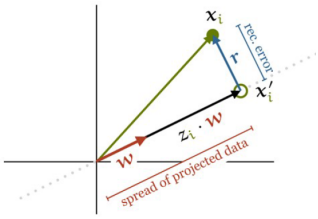
- Summary

	Filter	Wrapper	Embedded
How it works	Uses univariate, generic algorithm-independent criteria	Uses the machine learning algorithm's performance criteria to find optimal features	Feature selection is embedded during the model building process. Selection is done by observing each iteration of model training
Computation Speed	Fast	Very High Computation time with many features	Somewhere between Filter and Wrapper methods
Overfitting problem		Prone to overfitting because it involves training of models with different feature combinations	Generally reduces overfitting through some penalty mechanism
Prediction Performance	Not always the best performing	Best performance compared to other two methods	Somewhere between Filter and Wrapper methods.
Techniques	Correlation, Variance Threshold, ChiSquare, Anova	Forward Selection, Backward Elimination, Recursive Feature Elimination	Lasso, Ridge, Randomforests

  - Filter Method
    - Features are selected before the application of machine learning.
    - Features are selected independent of the analytical task using univariate analysis
  - Wrapper Method
    - Similar to a greedy search, it evaluates all the possible combinations of features using a learning algorithm's performance as evaluation criteria

        - Retains the feature subset that produces the best results
    - Embedded Method
        - Feature selection occurs intrinsically as part of the machine learning.
        - Specifically, the algorithm itself decides which attributes to use and which to ignore in order to maximize its performance
        - Can be sub-categorised into Supervised and Un-supervised feature selections methods
- Feature Extraction Methods
    - keep all original features by extracting them to form new independent variables
- PCA(Principal Component Analysis)



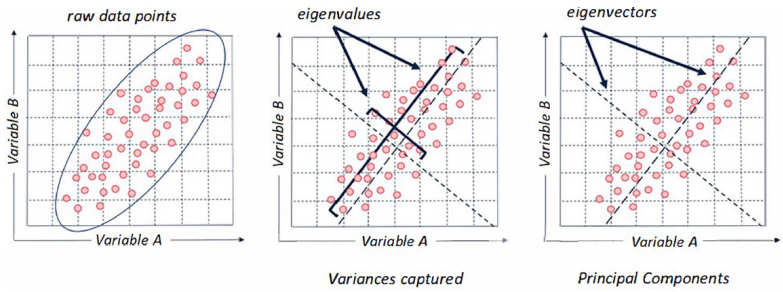


    - Brief Info
        - PCA uses a vector space transform to project the data from a high dimensional space into a lower-dimensional space.
        - Its main purpose is to reduce a correlated multidimensional data set (xi) to an uncorrelated lower dimensional space (yi) with maximum variance.
    - Principal components are linear combinations of the xi :

$$
\begin{aligned}
y_1 &= a_{11}x_1 + a_{21}x_2 + \cdots + a_{p1}x_p \\
y_2 &= a_{12}x_1 + a_{22}x_2 + \cdots + a_{p2}x_p \\
&\vdots \\
y_p &= a_{1p}x_1 + a_{2p}x_2 + \cdots + a_{pp}x_p.
\end{aligned}
$$

y_s principal component
x_s original Feature
a_{ij} coefficients

        - Each component is a weighted sum of the xi where the aij are the weights, or coefficients.
        - The properties of Principal Components:
            - are linear combinations of the original features,
            - are orthogonal (uncorrelated) to each other, and
            - capture the maximum amount of variance in the data
            - 1st component captures the most , followed by 2nd, then 3rd, etc.
    - PCA tries to find a new axis that captures the maximum variance within the data once it is projected onto the new axis:



- The new axes are called principal components (PCs)
- The 1st PC accounts for the maximum variance in the data
- The 2nd PC accounts for the maximum variance that has not been accounted by ;; 0 the 1st variable
- The pth PC accounts for the maximum variance that has not been accounted by the p-1 variables
- The Principal Components are considered as new variables, and they are all uncorrelated.
- If a substantial amount of the total variance in the data is accounted for by a few principal components then one can use these fewer no. of variables for further investigation instead of original features.
- This subset of variables can be identified and shortlisted by looking at the eigenvalues (i.e. variance accounted for by the PC)

- Pipeline(example)

Income	Education	Age	Residence	Employ	Savings	Debt	Credit cards
50000	16	28	2	2	5000	1200	2
72000	18	35	10	8	12000	5400	4
61000	18	36	6	5	15000	1000	2
88000	20	35	4	4	980	1100	4
91100	18	38	8	9	20000	0	1
45100	14	41	15	14	3900	22000	4
36200	14	29	6	5	100	7000	5
41000	12	34	9	8	5000	200	3
40000	16	32	8	7	19000	1760	2
32000	16	30	2	2	16000	550	1
29000	16	28	1	4	2100	4600	2
21240	12	26	2	2	100	10010	3
58700	12	38	9	9	4500	7800	5
41000	14	29	5	4	300	10000	6
38720	16	36	11	11	24500	540	2
88240	16	38	13	12	13600	8100	2
40000	18	39	7	6	16000	1300	2

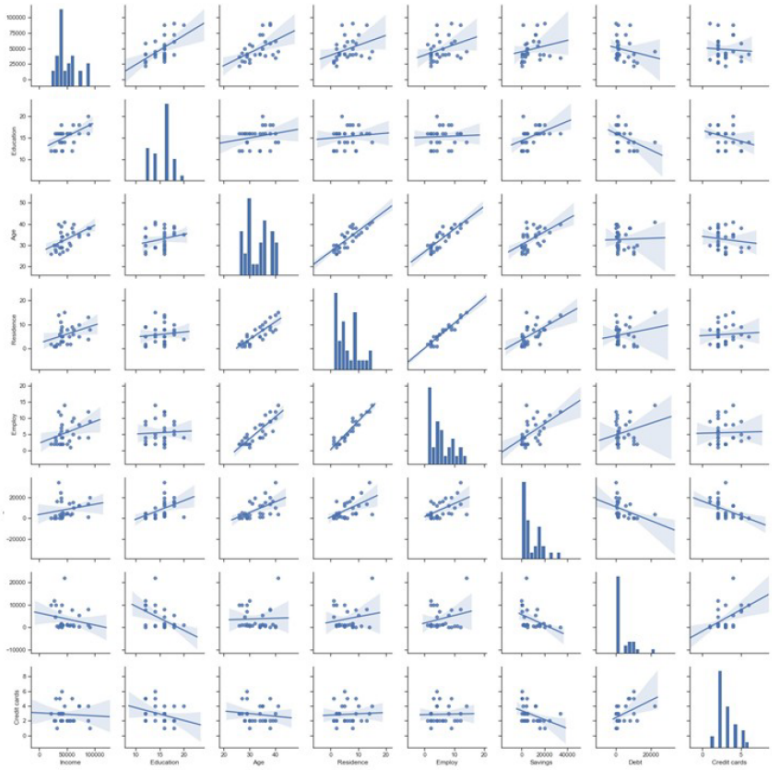
Income: Yearly Income
Education: no. of years
Age: years
Residence: years at current addr
Employ: years at current coy
Savings: current saving
Debt: current debts
Credit cards: #credit cards owned

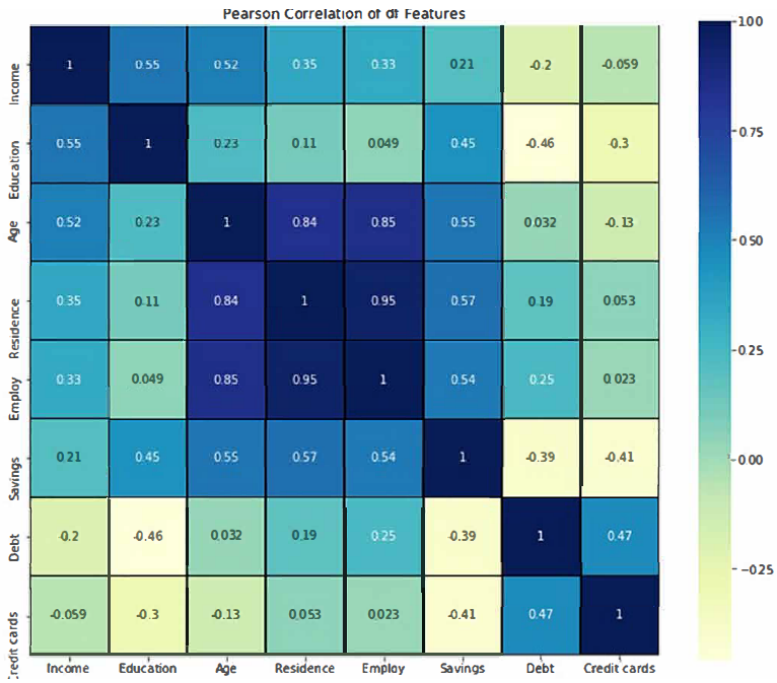
- Basic Data understanding
  - Summary statistics

```
Data Dimensions:   (30, 8)
Missing Values found: 0
Summary Statistics:
                miss  count      mean       std      min       25%      50%       75%      ma
Income             0   30.0  48646.00  18505.45  21240.0  37055.00  41100.0  58125.00  91100.0
Education          0   30.0     15.27      2.13     12.0     14.00     16.0     16.00     20.0
Age                0   30.0     32.80      4.63     26.0     29.00     33.0     36.00     41.0
Residence          0   30.0      5.93      3.98      1.0      2.25      5.0      8.00     15.0
Employ             0   30.0      5.50      3.66      1.0      2.00      4.0      8.00     14.0
Savings            0   30.0   8319.33   8550.50      0.0   1950.00   4750.0  14050.00  34000.0
Debt               0   30.0   3812.00   5042.51      0.0    800.00   1200.0   6600.00  22000.0
Credit cards       0   30.0      2.87      1.25      1.0      2.00      2.5      3.75      6.0
```

  - Correlation Matrix/Heatmap

	Income	Education	Age	Residence	Employ	Savings	Debt	Credit cards
Income	1.00	0.55	0.52	0.35	0.33	0.21	-0.20	-0.06
Education	0.55	1.00	0.23	0.11	0.05	0.45	-0.46	-0.30
Age	0.52	0.23	1.00	0.84	0.85	0.55	0.03	-0.13
Residence	0.35	0.11	0.84	1.00	0.95	0.57	0.19	0.05
Employ	0.33	0.05	0.85	0.95	1.00	0.54	0.25	0.02
Savings	0.21	0.45	0.55	0.57	0.54	1.00	-0.39	-0.41
Debt	-0.20	-0.46	0.03	0.19	0.25	-0.39	1.00	0.47
Credit cards	-0.06	-0.30	-0.13	0.05	0.02	-0.41	0.47	1.00





- Check for appropriateness of PCA Method
  - Kaiser-Meyer-Olkin Test
    - The Kaiser-Meyer-Olkin (KMO) test is a Measure of Sampling Adequacy.

- The KMO is a statistic that indicates the proportion of variance in your variables that might be caused by underlying factors.
- In essence, it answers "How suited is your data for PCA ?'
- KMO returns values between O and 1.
- A rule of thumb for interpreting the statistic:
  - KMO > 0.5 indicates the sampling is adequate
  - KMO < 0.5 indicate the sampling is NOT adequate

- Bartlett's Test

$$\mathbf{H_0}: \text{ Correlation Matrix = Identity Matrix}$$
$$\mathbf{H_1}: \text{ Correlation Matrix} \neq \text{Identity Matrix}$$

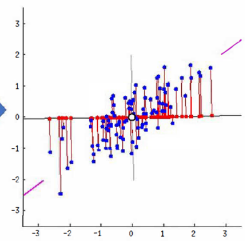
REJECT the NULL if the P-value is < 0.05

- Bartlett's Test of Sphericity tests the hypothesis that your correlation matrix is an identity matrix, which would indicate that your variables are unrelated and therefore unsuitable for structure detection.
- We want to reject the Null hypothesis - i.e. we want the correlation matrix to diverge significantly from the identity matrix. If true, it implies the existence of redundancies between the variables such that PCA can summarize this with fewer variables.

- Run PCA and examine the outputs

PCA Transformation

Income	Education	Age	Residence	Employ	Savings	Debt	Credit cards
50000	16	28	2	2	5000	1200	2
72000	18	35	10	8	12000	5400	4
61000	18	36	6	5	15000	1000	2
88000	20	35	4	4	980	1100	4
91100	18	38	8	9	20000	0	1
45100	14	41	15	14	3900	22000	4
36200	14	29	6	5	100	7000	5
41000	12	34	9	8	5000	200	3
40000	16	32	8	7	19000	1760	2
32000	16	30	2	2	16000	550	1
29000	16	28	1	4	2100	4600	2
21240	12	26	2	2	100	10010	3
58700	12	38	9	9	4500	7800	5
41000	14	29	5	4	300	10000	6
38720	16	36	11	11	24500	540	2
88240	16	38	13	12	13600	8100	2
40000	18	39	7	6	16000	1300	2



Principal Components

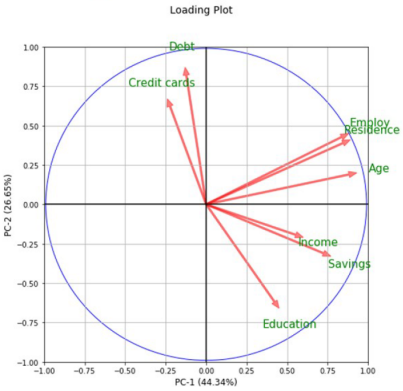
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Income	0.314	-0.145	0.676	-0.347	0.241	-0.494	-0.018	-0.030
Education	0.237	-0.444	0.401	0.240	-0.622	0.357	-0.103	0.057
Age	0.484	0.135	0.004	-0.212	0.175	0.487	0.657	-0.052
Residence	0.466	0.277	-0.091	0.116	0.035	0.085	-0.487	-0.662
Employ	0.459	0.304	-0.122	-0.017	0.014	0.023	-0.368	0.739
Savings	0.404	-0.219	-0.366	0.436	-0.143	-0.568	0.348	-0.017
Debt	-0.067	0.585	0.078	-0.281	-0.681	-0.245	0.196	-0.075
Credit cards	-0.123	0.452	0.468	0.703	0.195	0.022	0.158	0.058

- Principal Components
- Interpretation of PC {Loadings)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Income	0.6015	-0.2153	0.7028	-0.2573	0.1572	-0.2050	-0.0065	-0.0062
Education	0.4540	-0.6594	0.4169	0.1780	-0.4057	0.1482	-0.0371	0.0118
Age	0.9272	0.2005	0.0042	-0.1572	0.1141	0.2021	0.2366	-0.0107
Residence	0.8927	0.4114	-0.0946	0.0860	0.0228	0.0353	-0.1754	-0.1366
Employ	0.8793	0.4515	-0.1268	-0.0126	0.0091	0.0095	-0.1325	0.1525
Savings	0.7739	-0.3252	-0.3805	0.3233	-0.0933	-0.2357	0.1253	-0.0035
Debt	-0.1284	0.8688	0.0811	-0.2084	-0.4442	-0.1017	0.0706	-0.0155
Credit cards	-0.2356	0.6713	0.4865	0.5213	0.1272	0.0091	0.0569	0.0120

Component ***loadings*** are interpreted as ***correlation*** coefficients between the original variables and components. It is analogous to *Pearson's r*, and the squared *loading* is the percent of variance in that variable explained by the component.

- Loading Plot



- The Loading Plot graphs the **loadings** of Principal components.
- Loading Plots allow you to identify which variables have the **largest effect** on each component.
- Loadings close to -1 or 1 indicate that the variable **strongly influences** the component.
- Loadings close to 0 indicate that the variable has a **weak influence** on the component.
- Evaluating the loadings can also help you **characterize each component** in terms of the variables.

- How Many Components to retain?
  - Eigenvalues {Kaiser Rule)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Income	0.6015	-0.2153	0.7028	-0.2573	0.1572	-0.2050	-0.0065	-0.0062
Education	0.4540	-0.6594	0.4169	0.1780	-0.4057	0.1482	-0.0371	0.0118
Age	0.9272	0.2005	0.0042	-0.1572	0.1141	0.2021	0.2366	-0.0107
Residence	0.8927	0.4114	-0.0946	0.0860	0.0228	0.0353	-0.1754	-0.1366
Employ	0.8793	0.4515	-0.1268	-0.0126	0.0091	0.0095	-0.1325	0.1525
Savings	0.7739	-0.3252	-0.3805	0.3233	-0.0933	-0.2357	0.1253	-0.0035
Debt	-0.1284	0.8688	0.0811	-0.2084	-0.4442	-0.1017	0.0706	-0.0155
Credit cards	-0.2356	0.6713	0.4865	0.5213	0.1272	0.0091	0.0569	0.0120

```
Eigenvalues          :  3.67  2.206 1.081 0.55  0.425 0.172]
```

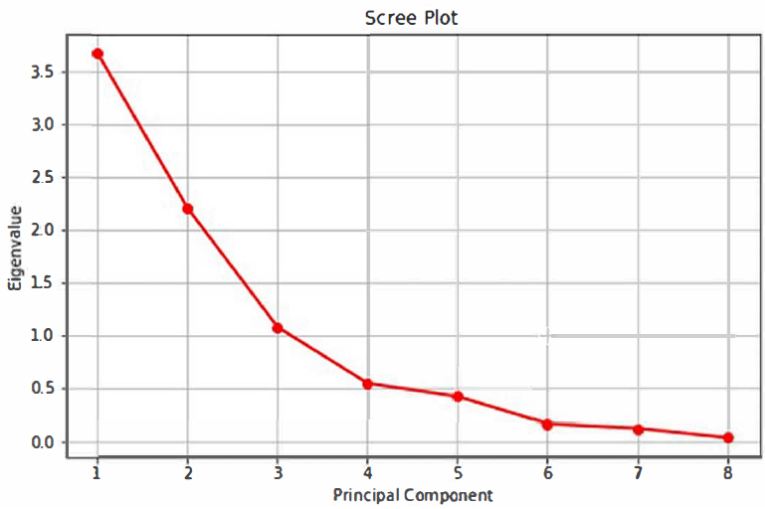
    - represent the total amount of variance that is explained by a given principal component. Eigenvalues are standardized and hence it carries one unit of information.Eigenvalues are the sum of squared component loadings across all items.
    - The Kaiser Rule is perhaps the most widely used criterion for selecting which components to keep.
    - It is based on the idea that a component should be considered insignificant if it does worse than a single field.
    - Each single field contains one unit of standardized variance, thus components with eigenvalues below 1 are not extracted.
  - Total variance explained

```
                        PC1    PC2    PC3    PC4   PC5   PC6   PC7   PC8
Eigenvalues          : [3.67  2.206 1.081 0.55  0.425 0.172 0.13  0.043]
% Variance Explained: [44.34 26.65 13.06  6.64  5.14  2.08  1.57  0.51]
% Cumulative)        : [44.34 70.99 84.05 90.69 95.83 97.91 99.48 99.99]
```

    - Eigenvalues can also be expressed in terms of a percentage of the total variance of the original fields.
    - The second row of the table denotes the proportion of the variance attributable to each component, and the next row denotes the proportion of the variance jointly explained by all components up to that point.

- The first 4 components explains 90% of the variability with 10% loss of information.

- Scree Plot



  - The Scree Plot shows the eigenvalues vs the components in decreasing order:

- Communalities Matrix

** Communalities :

	2	3	4
Income	0.408208	0.902090	0.968296
Education	0.640920	0.814708	0.846378
Age	0.899891	0.899908	0.924620
Residence	0.966167	0.975117	0.982515
Employ	0.977002	0.993088	0.993247
Savings	0.704765	0.849539	0.954061
Debt	0.771255	0.777831	0.821247
Credit cards	0.506117	0.742829	1.014565

  - Does the solution account for all the original variables?
    - Communality represents the total amount of variance of a specific field that is jointly accounted for by all the components. It is calculated as the sum of squared loadings of the field across all components.
    - High communality values indicate that the original field is sufficiently explained by the reduced PCA solution.
    - Low communality implies an insignificant contribution to the formation of the PCA Solution
    - communalities > 0.5 are desired

- Proceeding to the next step
  - Eigenvectors

```
PC1 = 0.314*Income     + 0.237*Education + 0.484*Age +
      0.466*Residence + 0.459*Employ    + 0.404*Savings +
      -0.67*Debt       + -0.123*Credit cards
```

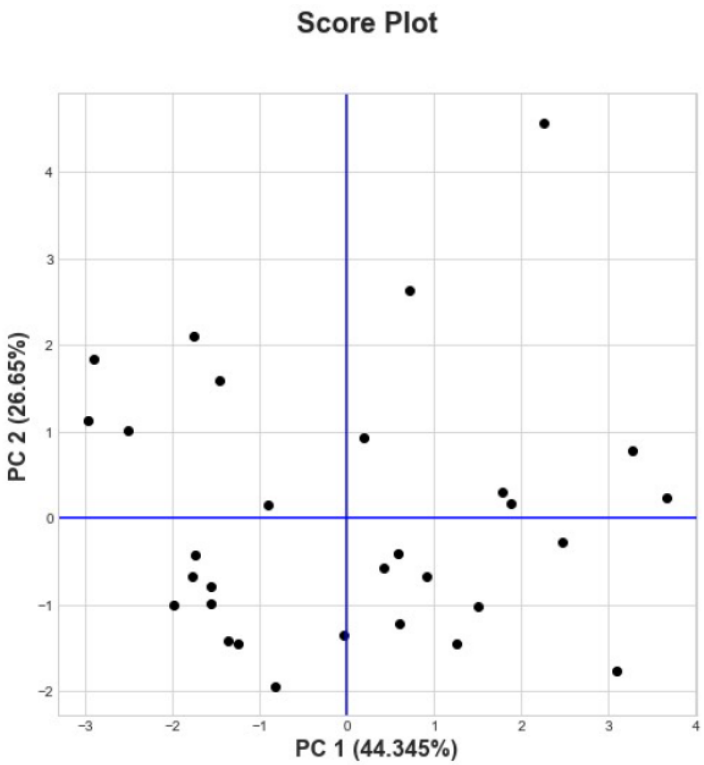
- Principal components are the linear combinations of the original variables that account for the variance in the data.
- The eigenvectors, which are comprised of coefficients corresponding to each variable, are used to calculate the principal component scores.
- The coefficients indicate the relative weight of each variable in the component.

- Component Scores

	PC1	PC2	PC3	PC4
0	-1.356145	-1.422152	0.169596	-0.337685
1	1.790714	0.306738	1.508913	0.621809
2	1.252730	-1.443638	0.334810	-0.056943
3	0.597244	-1.226393	3.172501	0.059572
4	3.086463	-1.759539	0.659380	-0.939320
5	2.258250	4.576150	0.046110	-0.843019
6	-1.469670	1.590170	0.526923	0.885788
7	0.186354	0.925404	-0.924450	-0.096049
8	0.913751	-0.676040	-1.107618	0.514616
9	-0.816324	-1.949886	-1.367173	-0.051855

  - These component scores are the values of the new "features".
  - They can be used like any other features in subsequent tasks.
  - Note that the scores are standardized values with mean=O and SD=l.
  - Hence the scores represent the number of standard deviations above or below the overall mean where each customer lies.

- Score Plot



- This score plot graphs the scores of the First principal component versus the scores of the Second principal component.
- If the first two components account for most of the variance in the data, you can use the score plot to assess the data structure and detect clusters, outliers, and trends.
- Groupings of data on the plot may indicate two or more separate distributions in the data .
- If the data follow a normal distribution and no outliers are present, the points are randomly distributed around zero .

- Assumptions for PCA
  - When you choose to analyse your data using PCA, part of the process involves checking to make sure that the data you want to analyse can actually be analysed using PCA. You need to do this because it is only appropriate to use PCA if your data "passes" four assumptions that are required for PCA to give you a valid result.
  - Sampling adequacy: ideally, there should be 150+ cases and there should be ratio of at least five cases for each variable (Pallant, 2010)
  - Suitability (Correlations exist): there should be some linear correlation among the factors to be considered for PCA
  - Continuous variables: variables should be interval or ratio.
  - Outliers: PCA is sensitive to outliers; disproportionate influence on your results. Remove data that are > 3 standard deviations
- Data Loss

```
[[ 19.   4.  90. 150.]
 [ 43.  12.  30.  35.]
 [ 13.   3.  10.  20.]
 [ 60.  14. 100.  80.]
 [  5.   1.  30.  55.]
 [ 56.  11.  25.  35.]
 [ 25.   7.  30.  28.]
 [  3.   1.  65.  82.]
 [ 40.   9.  15.  30.]
 [ 65.  15.  20.  40.]]
```

Original Data

```
[[ 23.   5. 106. 138.]
 [ 35.   8.  27.  36.]
 [ 37.   9.  14.  20.]
 [ 28.   6.  73.  97.]
 [ 33.   8.  39.  52.]
 [ 36.   8.  24.  33.]
 [ 35.   8.  25.  34.]
 [ 29.   7.  65.  86.]
 [ 36.   9.  19.  26.]
 [ 36.   8.  24.  33.]]
```

1 PC (55.098%)

```
[[ 14.   3. 103. 139.]
 [ 44.  10.  29.  36.]
 [ 14.   4.   7.  22.]
 [ 66.  15.  85.  92.]
 [  5.   1.  30.  55.]
 [ 54.  12.  30.  31.]
 [ 28.   7.  22.  34.]
 [  6.   1.  58.  88.]
 [ 38.   9.  20.  26.]
 [ 61.  14.  32.  30.]]
```

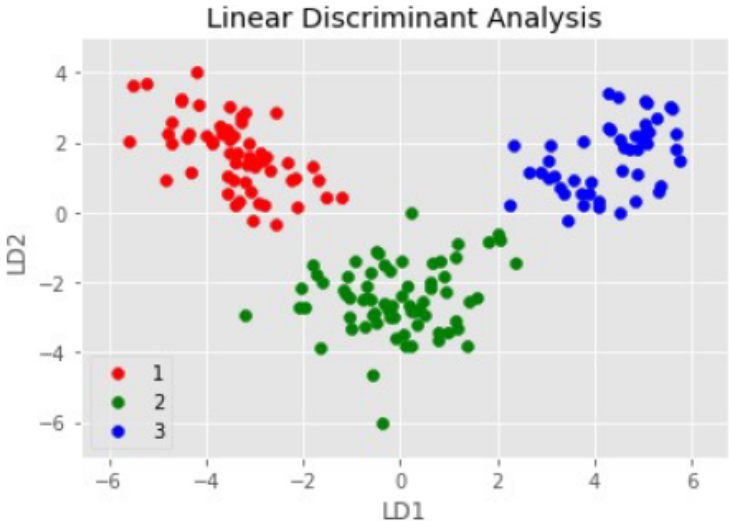
2 PCs (96.564%)

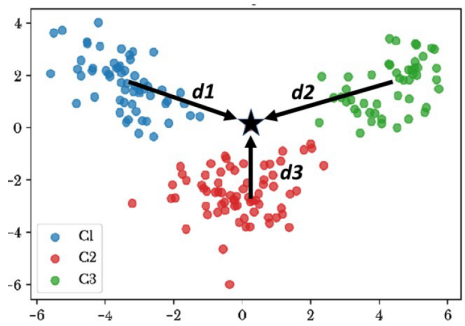
```
[[ 19.   3.  90. 149.]
 [ 43.  10.  30.  34.]
 [ 12.   3.   9.  20.]
 [ 59.  14.  99.  80.]
 [  4.   1.  29.  55.]
 [ 55.  12.  24.  35.]
 [ 25.   6.  30.  27.]
 [  2.   1.  64.  82.]
 [ 39.   9.  14.  30.]
 [ 65.  14.  20.  39.]]
```

3 PCs (96.985%)

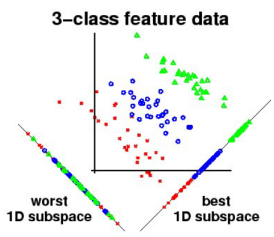
- The goal of PCA is to extract the smallest number of components which account for as much as possible of the information of the original features.
- Hence, there will be some amount of information that will be lost.
- To illustrate information lost, we will reconstruct the original data using only the saved principal components.

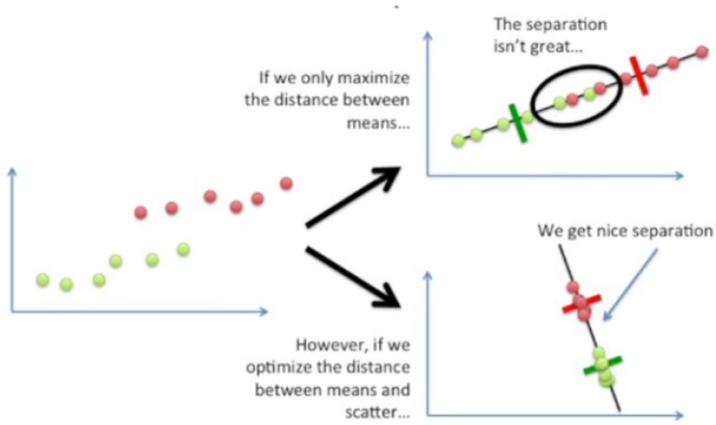
- LDA(LINEAR DISCRIMINANT ANALYSIS)



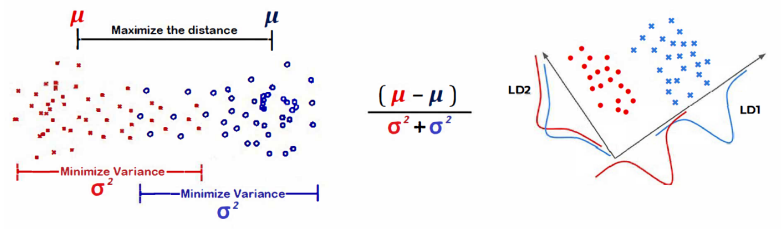


$$\frac{d1 + d2 + d3}{\sigma^2 + \sigma^2 + \sigma^2}$$



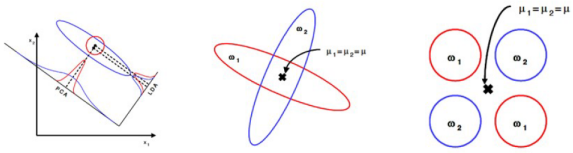


- We not only reduce the number of features BUT at the same time create class separability. This is most useful for multi-variate classification-type problems.
- LDA finds a new dimension that yields



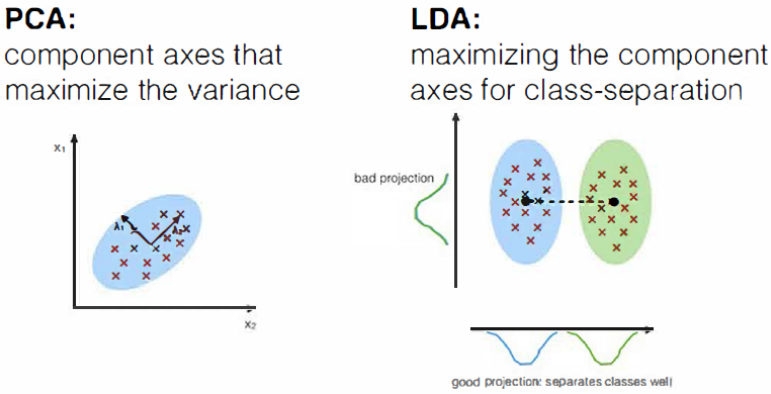
- Maximum separation between the class means
- Minimum variance within class

- Assumption
  - Assumes that the data is uni-modal gaussian (normally distributed)
  - LDA will fail if the discriminatory information is not in the mean but in the variance of the data



  - LDA is not stable for problems with small sample size
  - LDA produces c-1 discriminant vectors, where c = number of classes

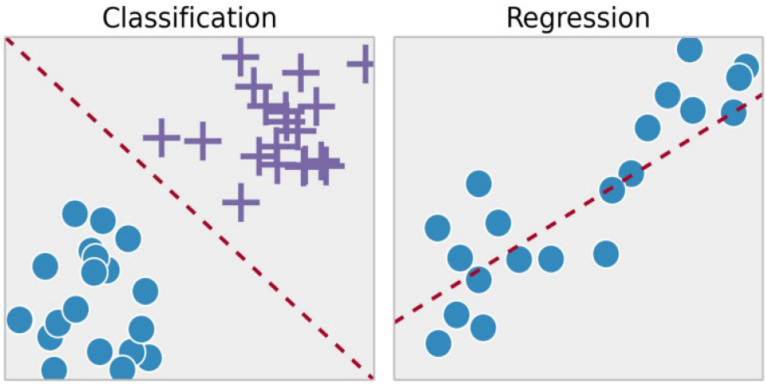
- PCA &LDA

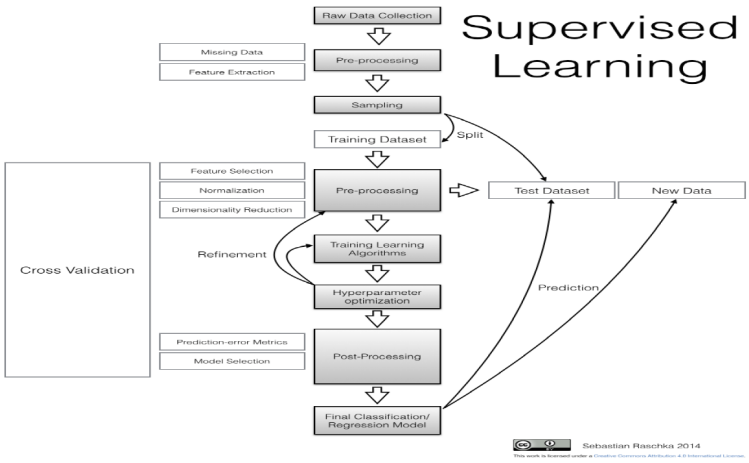


  - Whereas PCA yields the directions (principal components) that maximize the variance of the data, LOA also aims to find the directions that maximize the separation (or discrimination) between different classes, which can be useful in pattern classification problem.
  - LDA takes the class label into consideration whereas PCA ignores them.
  - Since it uses the class information, LOA is a "supervised learning" method while PCA is an unsupervised learning method.
  - PCA maximizes the variation explained in the data
  - LDA maximizes the separation of the classes in the data
  - Both rank the components according to their importance:
  - PCA ranks them according to the largest amount of variance explained in the data
  - LOA ranks them according to the largest amount of variance explained between

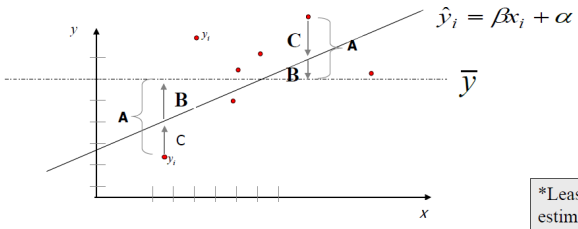
- ML for pattern recognition

- Supervised learning





  - A teacher provides a category label or cost for each pattern in the training set.
  - Linear Regression



*Least squares estimation gave us the line (β) that minimized C^2

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

A^2
SS_{total}
Total squared distance of observations from naïve mean of y
Total variation

B^2
SS_{reg}
Distance from regression line to naïve mean of y

Variability due to x (regression)

C^2
$SS_{residual}$
Variance around the regression line

Additional variability not explained by x—what least squares method aims to minimize

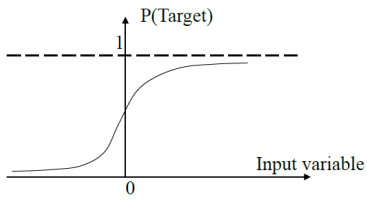
$R^2 = SS_{reg}/SS_{total}$

    - Use for numeric targets
    - Good if you know the target changes linearly
    - Assumes the model: t = ax +by +cz+ d etc.
  - Logistic Regression

$$\ln\left(\frac{P}{1-P}\right) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

P= Class probability
P/(1-P) = odds
ln(p/(1-p) = logit (log odds)

- Designed for classification problems
- Tries to estimate class probabilities directly
- one input variable



Which is a good match for many T/F prediction situations

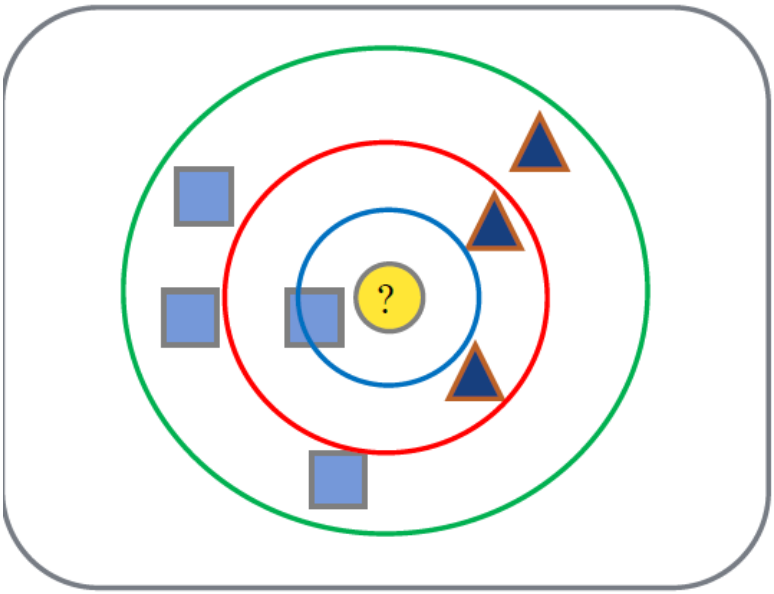
The transformation ln(p/1-p) turns this into a straight line (p = prob(target))

- multiple input

Predict the probabilities of the different possible outcomes

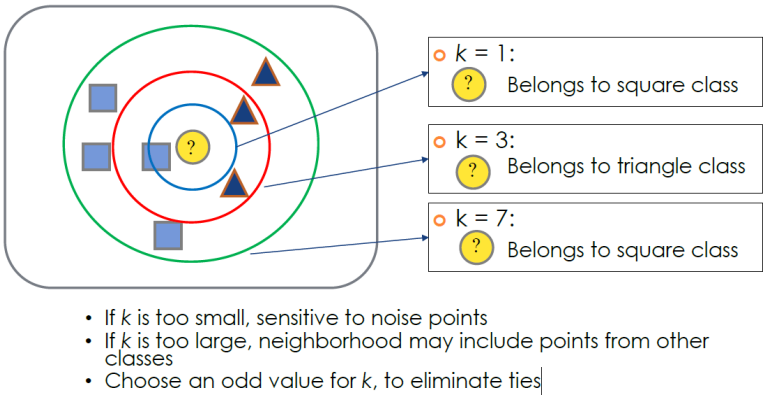
$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$
$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$
$$\cdots\cdots$$
$$\Pr(Y_i = K) = \frac{e^{\boldsymbol{\beta}_K \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

- Instance based Learning (K-NN)



- Non-linear decision surfaces
- Can be computationally intensive
- Distance measure is important
- Requires 3 things:
  - Feature Space(Training Data)
  - Distance metric

            - to compute distance between records
        - The value of k



- If *k* is too small, sensitive to noise points
- If *k* is too large, neighborhood may include points from other classes
- Choose an odd value for *k*, to eliminate ties

            - the number of nearest neighbors to retrieve from which to get majority class
        - To classify an unknown record:
            - Compute distance to other training records
            - Identify k nearest neighbors
            - Use class labels of nearest neighbors to determine the class label of unknown record
    - Common Distance Metrics:
        - Euclidean distance
        - Hamming distance
    - Determine the class from knearest neighbor list
        - Take the majority vote of class labels among the k-nearest neighbors
        - Weighted factor
    - Advantage
        - Simple technique that is easily implemented
        - Building model is inexpensive
        - Extremely flexible classification scheme
        - Nearest Neighbor classifiers are lazy learners
            - **lazy learning** is a learning method in which generalization of the training data is, in theory, delayed until a query is made to the system,
        - Scaling issues
            - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Naïve Bayes Classifiers

- Suppose X = (x_1, x_2, …, x_n) and *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$
- Using Bayes' theorem, maximize $P(C_i|X)$ is equivalent to maximize

$$\frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem.
- It assumes that attributes are conditionally independent
- example

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Let's say you are given a fruit that is: Long (1), Sweet (1) and Yellow(1), can you predict what fruit it is ?

P(C=Banana | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long\,|\,Banana) * P(Sweet\,|\,Banana) * P(Yellow\,|\,Banana) * P(Banana)}{P(Long) * P(Sweet) * P(Yellow)}$$

$$= \frac{0.8 * 0.7 * 0.9 * 0.5}{P(evidence)} = 0.252/p(evidence)$$

P(C=Orange | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long\,|\,Orange) * P(Sweet\,|Orange) * P(Yellow\,|Orange) * P(Orange)}{P(Long) * P(Sweet) * P(Yellow)}$$

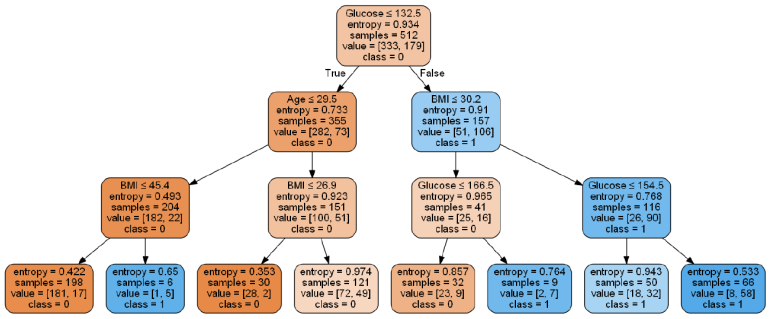
$$= \frac{\frac{0}{300} * \frac{150}{300} * \frac{300}{300} * \frac{300}{1000}}{P(evidence)} = 0/p(evidence)$$

P(C=Other | X1=Long, X2=Sweet and X3=Yellow)=

$$\frac{P(Long\,|\,Other) * P(Sweet\,|Other) * P(Yellow\,|Other) * P(Other)}{P(Long) * P(Sweet) * P(Yellow)} = \frac{\frac{100}{200} * \frac{150}{200} * \frac{50}{200} * \frac{200}{1000}}{P(evidence)} = 0.01875/p(evidence)$$

- Laplace Correction
  - increase the count of the variable with zero to a small value (usually 1) in the numerator, so that the overall probability doesn't become zero.
- Pros&Cons
  - Pros:
    - Easy and fast to predict. Perform well in multi class prediction
    - When assumption ofindependence holds, a Naive Bayes classifier performs better than othermodelslike logistic regression and needs less training data.

        - It perform well in case of categorical input variables compared to numerical variables. For numerical variable, normal distribution is assumed.
    - Cons:
        - The assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.
- Decision Trees



    - A decision tree is a flow-chart-like tree structure.
        - An internal node performs a test on an attribute
        - A branch represents a result of the test
        - A leaf node represents a class label
        - At each node, one feature is chosen to split training examples into distinct classes
        - A new sample is classified by following a matching path to a leaf node
    - Building a tree
        - Top-down tree construction
        - At start, all training data are at the root.
        - Partition the examples recursively by choosing one feature each time.
        - At each node, available attributes are evaluated on the basis of separating the classes of the training examples. A goodness function is used for this purpose.
        - Typical goodness measures:
            - Information gain (ID3/C4.5)
            - Information gain ratio (C4.5)
            - Gini index (CART)

- If a data set T contains examples from n classes, gini index -- gini(T) is defined as

$$gini\ (T) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in T

gini(T) is minimized if the classes in T are skewed

- After splitting T into two subsets T1 and T2 with sizes N1 and N2, the gini index of the split data is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute providing smallest $gini_{split}(T)$ is chosen to split the node

- Heuristic Search
  - choose the attribute that produces the “purest”nodes
  - Search bias: Search the space of decision trees from simplest to increasingly complex (greedy search, no backtracking, prefer small trees)
  - Search heuristics: At a node, select the attribute that is most useful for classifying examples, split the node accordingly
  - choose attribute that gives the greatest information gain
    - **Information gain** measure is aimed to minimize the number of tests needed for the classification of a new object
    - **Gain(S,A)** - expected reduction in entropy of **S** due to sorting on **A**

$$Gain(S,A) = E(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot E(S_v)$$

Information Gain by splitting on A

Expected information (entropy) needed to classify S

Information needed (after using A to split S into v partitions) to classify S

    - **Most informative** attribute: **max Gain(S,A)**
    - Computing Information-Gain for Continuous-Valued Attributes
      - Let attribute A be a continuous-valued attribute
      - Must determine the best split pointfor A
        - Sort the value A in increasing order
        - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
        - (ai+ai+1)/2 is the midpoint between the values of aiand ai+1
        - The point achieving the maximum information gainfor A is selected as the split-point for A
      - Split:
        - S1 is the set of tuples in S satisfying A ≤ split-point, and S2 is the set of tuples in S satisfying A > split-point
      - Stopping criteria
        - if all examples belong to same class Cj, label the leaf with Cj

- if all attributes were used, label the leaf with the most common value Ckof examples in the node
- min_samples_split-The minimum number of samples required to split an internal node.
- min_samples_leaf-The minimum number of samples required to be at a leaf node
- max_depth-The maximum depth of the tree.

- Highly-Branching Features
  - Problematic: attributes with a large number of values (extreme case: ID code)
  - Subsets are more likely to be pure if there is a large number of values
    - Information gain is biased towards choosing features with a large number of values
    - This may result in overfitting(selection of a feature that is non-optimal for prediction)
  - Gain ratio
    - Intrinsic information: entropy of distribution of instances into branches

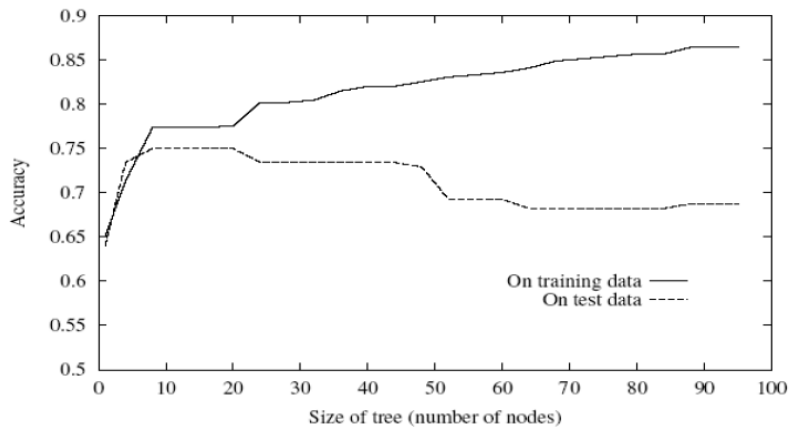
$$GainRatio(S,A)=\frac{Gain(S,A)}{IntrinsicInfo(S,A)}.$$

    - *Gain ratio* (Quinlan'86) normalizes info gain by:

$$IntrinsicInfo(S,A)=-\sum\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}.$$

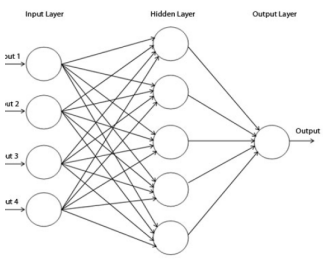
    - The attribute with the maximum gain ratio is selected as the splitting attribute
    - a modification of the information gain that reduces its bias on highly-branching attributes
    - Gain ratio takes number and size of branches into account when choosing an attribute

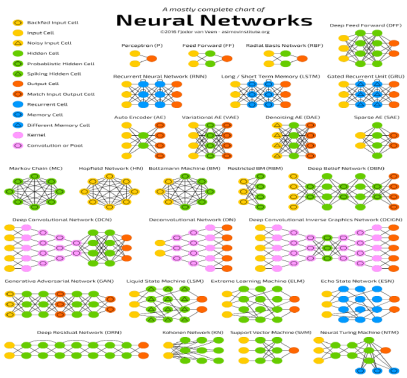
- Overfitting



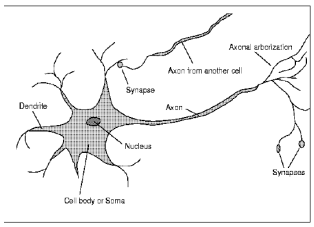
- Too many branches, some may reflect anomalies due to noise or outliers

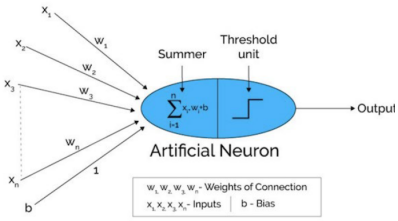
- Poor accuracy for unseen samples
- solution:
  - Pre-pruning (forward pruning): stop growing the tree e.g.
    - When data split not statistically significant
    - Too few examples are in a split
  - Postpruning: Remove branchesfrom a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of validation data to decide which is the “best pruned tree”
  - Fixed depth/ Early stopping
- Pros&Cons
  - Pros:
    - simple to understand and interpret
    - little data preparation and little computation
    - indicates which attributeare most important for classification
  - Cons:
    - not guaranteed to produce an optimal decision tree
    - perform poorly with many classes and small data
    - over-complex trees do not generalise well from the training data (overfitting)
- Neural Networks





  - Make no assumptions about the data
  - General Architecture of Neural Networks

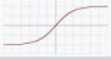
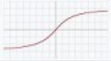
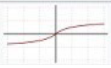




    - Input layer + Hidden Layer + Output Layer
    - Weights

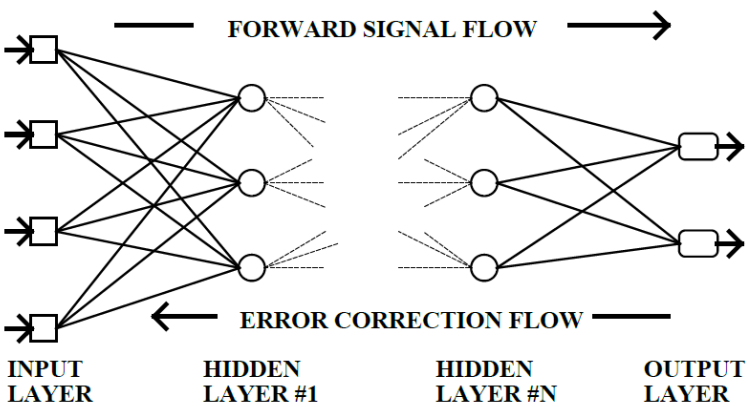
- Normally initial weights are randomised to small real numbers

- Activation functions f(net)

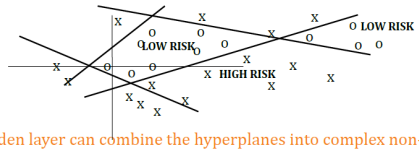
Name	Plot	Equation	Derivative (with respect to x)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$

- Learning rule
  - determine how to adapt connection weights in order to optimise the network performance Wi(t+1)=Wi(t)+ΔWi(t)
  - indicate how to calculate the weight adjustment during each training cycle
- Activation calculation & Weight adjustment
  - Compute the activation levels across the network
  - Weight adjustment based on the errors /distance
- Limits
  - Require lots of training data
  - Training can be slow!
  - Limit training by
    - the time taken
    - number or training iterations
    - the accuracy

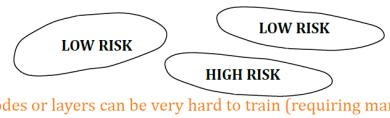
- MLP



- Nodes in the first hidden layer represent hyperplanes

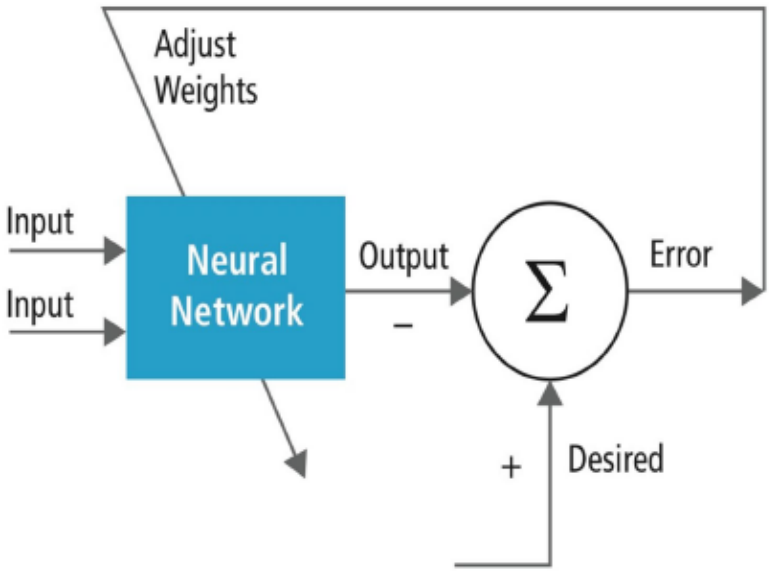


- Nodes in second hidden layer can combine the hyperplanes into complex non-linear surfaces



- Beware.... too many nodes or layers can be very hard to train (requiring many samples & long training time)

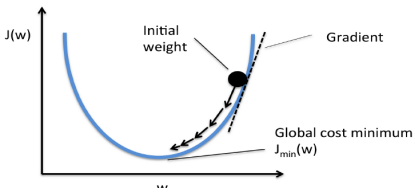
- Propagate signals forward and then errors backward
- Backpropogation(BP) ~ gradient descent learning

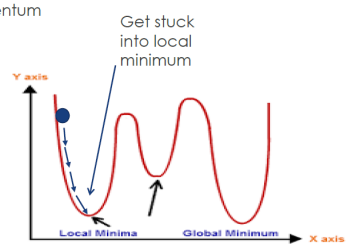


$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E}{\partial w_{ji}(t)} + \alpha \Delta w_{ji}(t)$$

Leaning rate

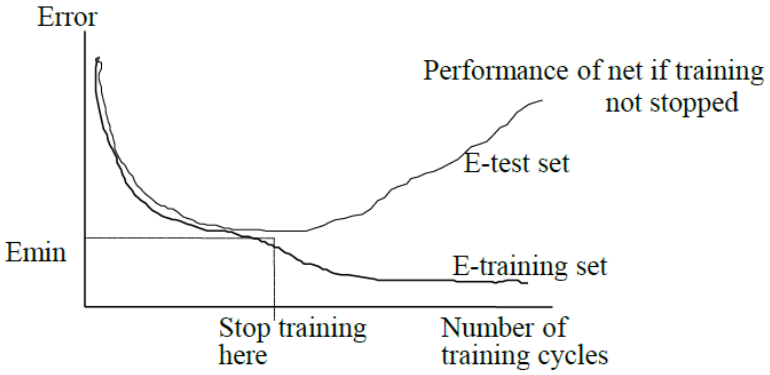
Momentum rate





- 1.Initialize the weights to small random numbers
- 2. Randomly select a training pattern pair (xp, tp) and present the input pattern xpto the network. Compute the corresponding network output pattern zp
- 3. Compute the error Epfor pattern (xp, tp)
- 4. Backpropagatethe errors according to the BP weight adjustment formulas
- 5. Test the Loss Function (mean square error (MSE), cross-entropy, etc):
- If it is below the required threshold, stop. Otherwise, repeat steps 2-5.
- 6. Test for generalization performance if appropriate

- Weights in hidden layers are adjusted to reduce aggregate errors in the output layer
- Generalization & Overtraining /Overfiting



  - Generalization is the ability of a network to correctly classify a pattern it has not seen(not been trained on).NNs generalize when they recognize patterns not previously trained on or when they predict new outcomes from past behaviors.
  - Networks can be overtrained.It means that they memorize the training set and are unable to generalize well

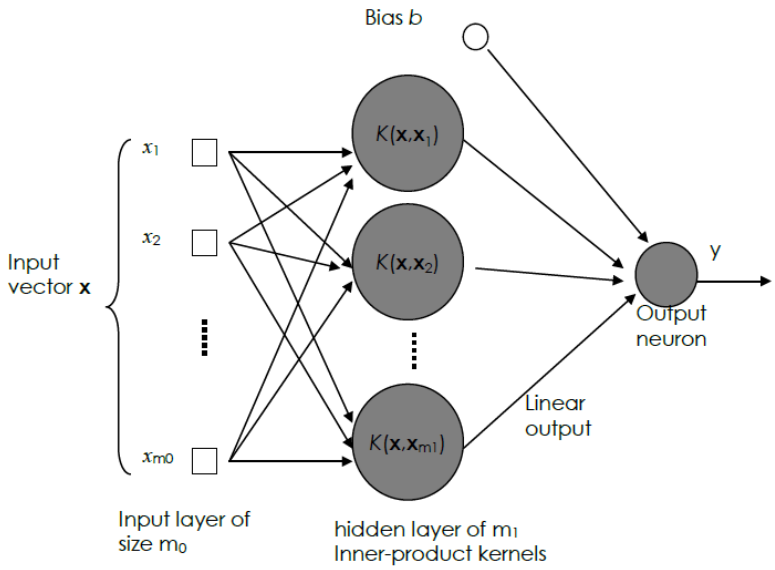
- Building NN & Pre-processing Data
  - Training/test data set
    - Perform statistical analyses to support data set choices
    - Select representative training set
    - Divide training set & testing set appropriately
  - Pre-processing the data
    - Data Coding
    - Data Smoothing
    - Data Transformation

Log	$y=\log(x)$
Delta	$\Delta x_i = x_i - x_{i-1}$
Normalization	$y = \frac{x - \min(x)}{\max(x) - \min(x)}$
Normalized Z score	$z = \frac{x - \mu}{\sigma}$

  - Testing / Evaluation
    - Testing the Generalization ability of a trained NN
      - Look for good performance on a validation set and test set
      - Changing the training algorithm
    - The performance varies with training/ solution procedures

- Network optimization should be performed after training/testing (eliminate redundant unneeded nodes and the corresponding weights –is called ‘pruning’)
- Periodic performance testing is essential to verify model’s accuracy - environmental changes can cause the data to change thereby afflicting the performance of the developed model

- SVM and Kernel Methods
  - SVM can be used for pattern classification and non-linear regression –but uses statistical learning theory
  - General architecture of a support vector machine



    - Input layer
    - Hidden layer of Inner-product kernels (fully connected with the input layer)
    - Output neuron
  - For nonlinear problem, it uses a nonlinear mapping to transform the original training data into a higher dimension
  - With the new dimension, it searches for the linear optimal separating hyperplane
  - SVM finds this hyperplane using support vectors (“essential” training tuples) and margins (defined by the support vectors)
  - Training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
  - Important concepts from the theoretical background

Consider a group of training patterns

- Training samples: $\{(\mathbf{x}_i, y_i)\}$ $i = 1, 2, \ldots, N$

  $\mathbf{x}_i$: the input pattern for the i-th example

  $y_i \in \{-1,1\}$: the corresponding desired output

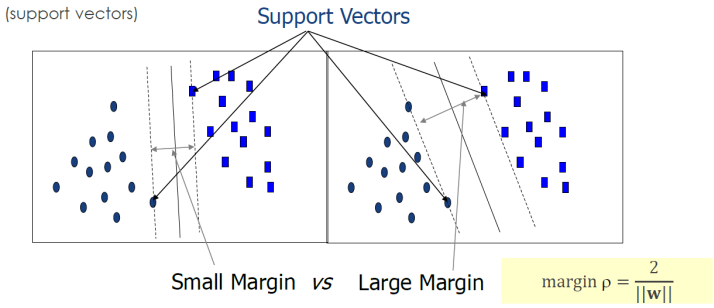
- The decision surface for the separation is a hyperplane

  $\mathbf{w}^T\mathbf{x} + b = 0$ (e.g. $w_1x_1 + w_2x_2 + \ldots + w_Nx_N + b = 0$)

  i.e. $\mathbf{w}^T\mathbf{x} + b \geq 0$ for $y_i = 1$

  $\mathbf{w}^T\mathbf{x} + b < 0$ for $y_i = -1$

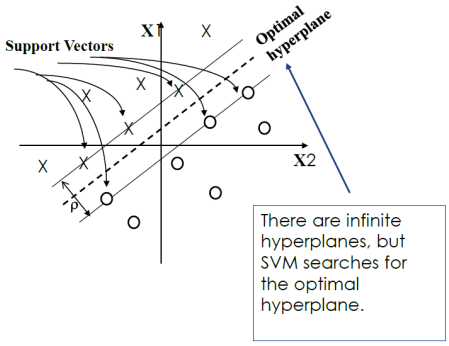
- *Margin of separation*
  - The separation between the decision surface hyperplane and the closest data points (support vectors)



- Optimal hyperplane for separable or non-separable patterns
- Support vector(A training pattern can be represented as a vector from the problem space)
- SVM structure
  - Hard Margin Linear SVM

The goal of a support vector machine for *linearly separable patterns* is to find the particular hyper-plane for which the margin of separation ρ is maximized.

Support vectors: those data points that lie closest to the decision surface and are therefore the most difficult to classify



$$\text{maximize } \frac{2}{||\mathbf{w}||} \Rightarrow \boxed{\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i \end{array}}$$

where $\mathbf{w}$ satisfy
$$\mathbf{w}^T\mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1$$
$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1$$

Consruct the lagrangian function (primal problem):

$$J(W, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

where $a_i, i = 1, \ldots, l,\ a_i \geq 0$ are Lagrange multipliers

- Dual problem:

Maximize $$Q(\boldsymbol{\alpha}) = \sum_{i}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Subject to: $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $\alpha_i \geq 0 \quad i = 1,2 \ldots l$

- This is a quadratic optimization problem.
- The solutions to the above dual optimization are a set of optimal α_i^* for $i = 1, \ldots, l$
  - for support vectors (SVs) $\mathbf{x}_i$, $\alpha_i^* > 0$, for non-SVs $\mathbf{x}_j$, $\alpha_j^* = 0$
  - $\boldsymbol{\alpha}^*$ determine the optimal parameters $\mathbf{w}^*$ and b^*

- Linear SVM

Linear SVM

- decision hypersurface is given by

l training samples $\{(\mathbf{x}_i, y_i)\}_l$

$$d(\mathbf{x}) = \boldsymbol{w}^T \mathbf{x} + b = \sum_{i=1}^{l} y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b$$

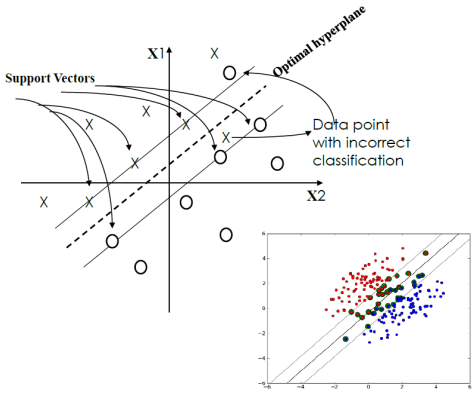
Inner product

for non-SVs, $\alpha_i = 0$

- Soft Margin Linear SVM

Given a set of *not linearly separable* training patterns, it is not possible to construct a separating hyperplane without encountering classification error.

The goal of a support vector machine for *not linearly separable patterns* is to find an optimal hyperplane that minimizes the misclassification error, averaged over the training set.



To classify data sets that are not linearly separable, the SVM within the linear framework is extended by introducing soft margin

- Replace the restriction

$$\text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

where ξ_i, called slack variables, are positive variables that indicate tolerance of misclassification.

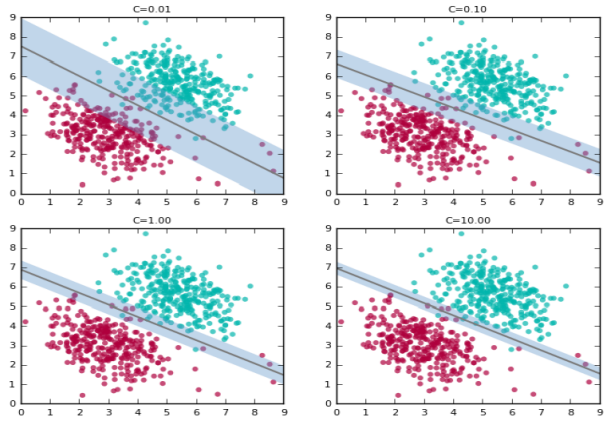
Note that $\xi_i = 0$ if there is no error for $\mathbf{x}_i$

There are optimization functions proposed for the case with soft margin, such as

$$\boxed{\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \end{array}}$$

- C is a penalty parameter
  - small $C \Rightarrow$ wide margin (more tolerance)
    - many support vectors will be on the margin
  - large $C \Rightarrow$ narrow margin
    - there will be few support vectors on the margin
  - $C \to \infty$ enforces all constraints $\Rightarrow$ hard margin

A higher value of C implies you want lesser errors on the training data.

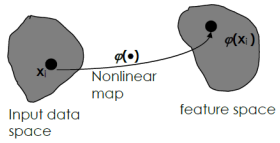


- ## SVM with Non–linear Kernels

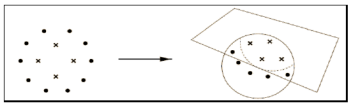
To construct a SVM for classification with an input space made up of non-linearly separable patterns

Form Inner-product kernels

- The multidimensional input space is transformed to a new feature space where the patterns are linearly separable with high probability, provided
  - **(a) The transformation is nonlinear**
  - **(b) The dimensionality of the feature is high enough**
- A subset of training samples $\{x_1, x_2, \ldots x_{m1}\}$ will be used as support vectors



Define the separating hyperplane as a linear function of vector drawn from the feature space rather than the original input space



Example

- Map the original 2-dimensional input space to a 3-dimensional feature space

$$\underset{\text{Input space } X}{x = (x_1, x_2)} \xrightarrow[\phi(x)]{} \underset{\text{Feature space } Z}{(x_1, x_2, x_1x_2)}$$

- The original non-linearly separable problem becomes linearly separable in the feature space

$$\underset{\text{Feature space } Z}{(x_1, x_2, x_1x_2)} \xrightarrow[\text{decision function } y = w \cdot \phi(x) + b]{} \underset{\text{Target (output) space } Y}{y \in Y}$$

The target of learning is to achieve a minimized error of classification with decision surface

Using dual representation we can rewrite

$$d(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

Provided in sample data

Parameters associated with input vectors in sample data, $\alpha_i \neq 0$ for support vectors

$$d(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle + b$$

All the information the learning algorithm needs is the inner products between data points in the feature space, where all $\mathbf{x}_i$ are in the input space.

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

A function that performs this direct computation of inner product is known as a kernel function, which is equivalent to the distance between **x** and **x'** measured in the higher dimensional feature space transformed by ϕ.

Apply a kernel function $K(X_i, X_j)$ to the original data, i.e.

$$K(X_i, X_j) = \Phi(X_i)\,\Phi(X_j)$$

Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(\kappa \boldsymbol{X}_i \cdot \boldsymbol{X}_j - \delta)$$

- Example

Objective: Classification for 1-D data

Suppose we have 5 training data points

- $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class A and 4, 5 as class B $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$

We use the polynomial kernel $K(x_i, x_j) = (x_i . x_j + 1)^2$ and C is set to 100. We need to find α_i (i=1, …, 5) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

After solving optimization problem, we get

- $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
- The support vectors are $\{x_2=2, x_4=5, x_5=6\}$

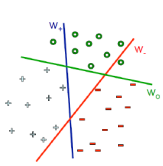
For a new point z, the discriminant function is

$$\begin{aligned} & f(z) \\ &= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b \\ &= 0.6667z^2 - 5.333z + b \end{aligned}$$

b is solved by solving f(2)=1 or by f(5)=-1 or by f(6)=1, all three give b=9

$$\boxed{f(z) = 0.6667z^2 - 5.333z + 9}$$

- multi-class
  - One vs. others
    - Training: Learn an SVM for each vs. the others
    - Testing: Apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value



- Learn 3 classifiers:
  - - vs. {o,+}, weights w_-
  - + vs. {o,-}, weights w_+
  - o vs. {+,-}, weights w_o
- Predict label using:

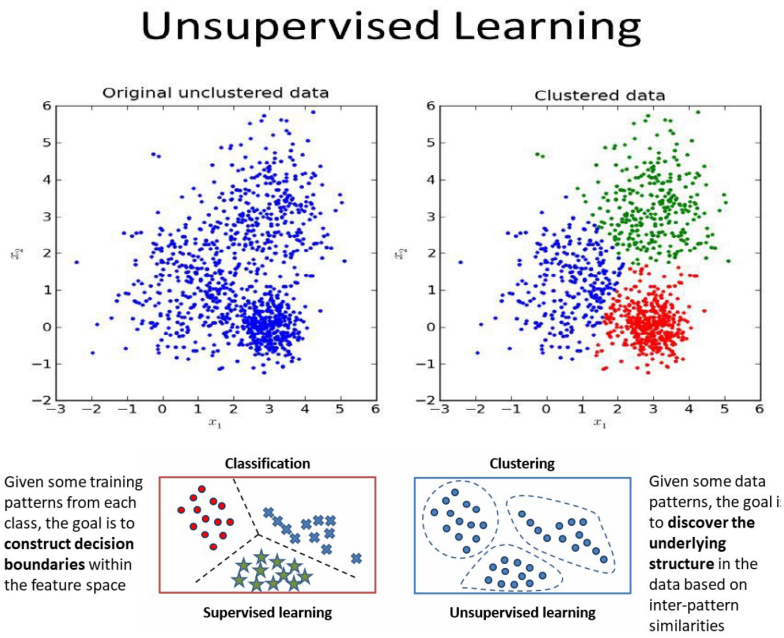
$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

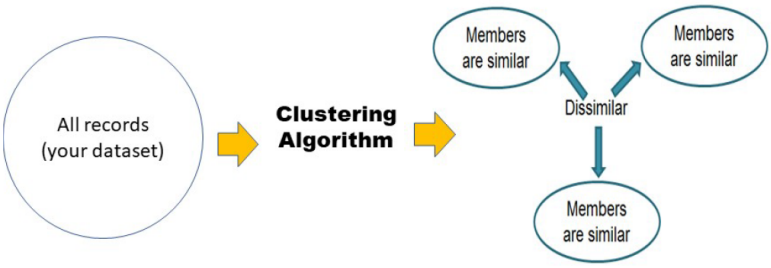
  - One vs. one
    - Training: Learn an SVM for each pair of classes
    - Testing: Major voting from each learned SVM

- Summary
  - The SVM is an elegant and highly principled learning method for the design of a feedforward network with a single hidden layer of nonlinear units
  - Design hinges on the extraction of a subset of the training data that serves as support vectors and therefore represents a stable characteristic of the data

- Learning in SVM
  - Learning algorithm operates only in a batch mode
  - The near-to-perfect classification performance is achieved at the cost of a significant demand on computational complexity
- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- An SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

- Unsupervised learning





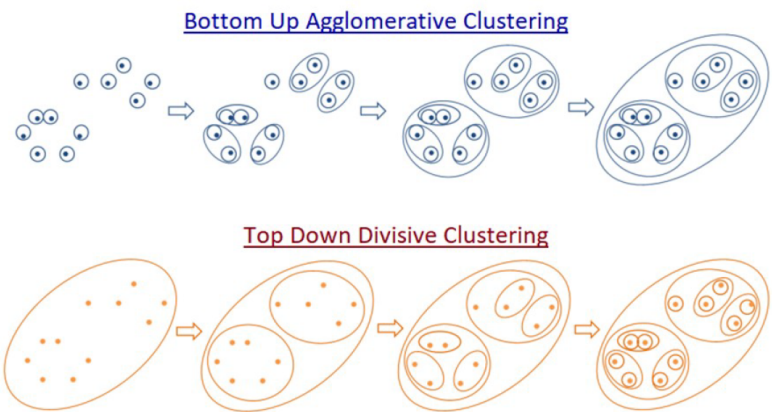
- The system forms clusters or natural groupings of the input patterns (based on some similarity criteria).
- Cluster Analysis (or simply "Clustering") is a multivariate data analysis technique
- It is an unsupervised learning method because there are no predefined classes or labels
- Clustering is useful for understanding complex business problems
  - Yields better understanding by uncovering natural patterns in data
  - Reduces data dimensions and make features more interpretable
  - Divide and conquer analysis of smaller, homogenous clusters may yield better results

- Members of a cluster share the same characteristics and hence are likely to react in the same manner. E.g. "typical" buyers of products. We can predict new buyers by measuring their similarity to these clusters
-

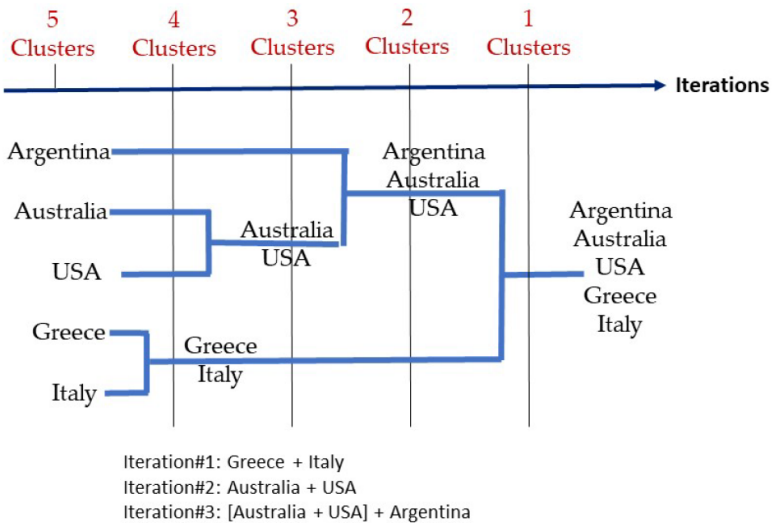
- Assumption
  - data is not uniformly distributed but rather they comprise of natural clumps.
- Measure similarity

Euclidean distance	Manhattan distance	Cosine Similarity	Chebyshev Distance	Jaccard Similarity
$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$	$\sum_{i=1}^{k}\lvert x_i - y_i \rvert$	$arccos\frac{A.B}{\lVert A \rVert \lVert B \rVert}$	$max_k \lvert X_{ik} - X_{jk} \rvert$	$\frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$

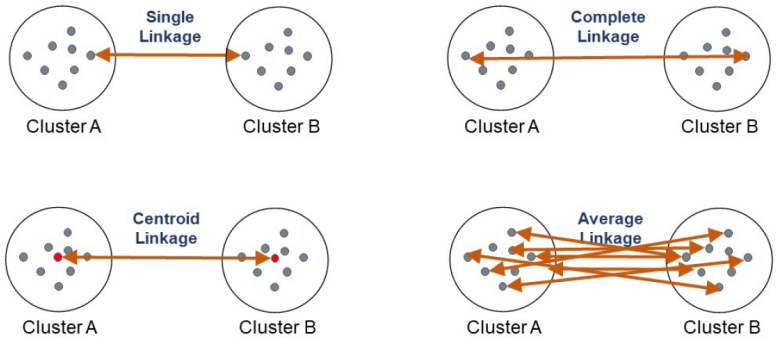
  - Similarity is an amount that reflects the strength of relationship between two data items
  - Lower/closer distance indicates that data or observation are similar and would get grouped in a single cluster.
  - Minimize the intra-cluster distance
  - Maximize the inter-cluster distance
  - Distance Measure for Non-numerical data
    - Ordinal Data
      - Customer Satisfaction:{Pitiful, Poor, Satisfactory, Good, Excellent}
      - The data value used for clustering is the index of the ordering- i.e. treated as if it were continuous data
    - Nominal Data
      - Job: {Lawyer, Doctor, Teacher}
      - The data value used for clustering is the exact category name.
- Method
  - Hierarchical Clustering



    - Agglomerative(most common)



- Start with the points as individual clusters and, at each step, merge the closest pair of clusters.
- This requires defining a notion of cluster proximity.
- Pipeline -> Start with N records in the dataset
  - Step 1: Assign each record as its own cluster
  - Step 2: Calculate the distances between each cluster
  - Step 3: Find the closest pair of clusters and merge them into a single cluster
  - Step 4: Repeat steps 2-4 until all records are clustered into a single cluster consisting of N members
- Inter-cluster Distance Measure



  - Single Linkage
    - Measure the distance between the closest points of two clusters.
  - Complete Linkage
    - Measure the distance between the farthest points of two clusters.
  - Centroid
    - Measure the distance between the centroids of two clusters.
  - Average

            - Measure the distance between all possible combination of points between the two clusters and take the mean.
    - Divisive
        - Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.
        - Pipeline -> Start with 1 Cluster consisting of N records
            - Step 1: Apply a flat clustering algorithm (e.g. K-means) to split the cluster into 2 clusters
            - Step 2: Add the 2 clusters to the pool of clusters
            - Step 3: Extract 1 cluster from the pool that consist of> 1 member (data observation)
            - Step 4: Repeat steps 1-4 until the pool consist only of clusters with a single member (data observation)
    - The results of hierarchical clustering is often displayed graphically using a tree-like diagram called a dendrogram, which displays both the cluster-subcluster relationships and the order in which the clusters were merged.

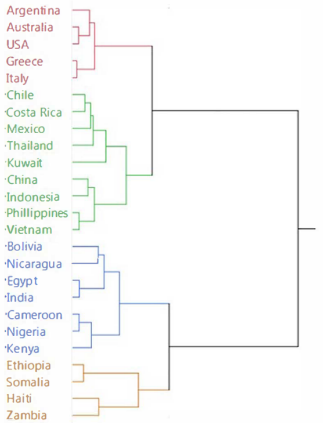
A **dendrogram** is a tree diagram used to illustrate the arrangement of the clusters

Read from left to right

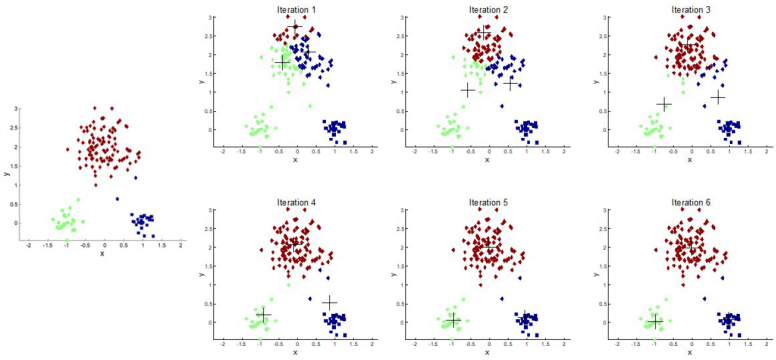
1. Greece + Italy
2. Australia + USA
3. Philippines + Vietnam
4. Cameroon + Nigeria ...

This color display is for 4 clusters

- C1: Argentina ... Italy
- C2: Chile ... Vietnam
- C3: Bolivia ... Kenya
- C4: Ethiopia ... Zambia



    - Limits
        - Once two clusters are merged, it cannot be undone.
        - It is hard to decide on the number of clusters to retain
        - Computationally heavy with a few thousand cases
        - Dendrogram can be too large to read for thousands of data observations
        - A single pass through the data may yield poorer results
- Partition Clustering
    - K-Means Clustering



- K-means clustering is a type of top-down divisive clustering
- Data observations are then clustered based on its proximity to a nearest cluster centroid.
- Cluster centroids are dynamically calculated as cluster members migrate during each iteration of the algorithm.
- It has the advantage of being more efficient since it does not generate a complete hierarchy all the way down to individual objects
- k-means is perhaps the most widely used clustering algorithm due to its simplicity and efficiency
- Limits
  - K-means requires the user to specify in advance the number of clusters to be formed (k clusters).
  - If you have no intuition about the number clusters, then K-means requires some trial and error, multiple runs, and evaluation of different solutions
  - It is distance based and unlike the hierarchical algorithms, it does not need to calculate the distance between all pairs of records
  - K-means is computationally efficient and fast clustering algorithm that can handle both long (many records) and wide datasets (many input fields)
- Pipeline -> Specify K number of clusters
  - Step 1: Arbitrarily designate k data point as seed points (initial centroids)
  - Step 2: Calculate the distances between every data point and the
  - centroid.
  - Step 3: Re-assign data points to the nearest centroid to form new clusters
  - Step 4: For each cluster that has lost or gained a data point, re-compute
  - the new cluster centroids
  - Step 5: Repeat Step 2-4, until (a) no more re-assignment is possible or (b)

- the newly computed centroids do not deviate significantly from the
- original or (c) some maximum number of iteration is reached

- Objective Function ->Sum of Squared Error (SSE)

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

where x is a data point in cluster C_i and m_i is the centroid (mean) for cluster C_i and *dist* is usually the Euclidean distance measure

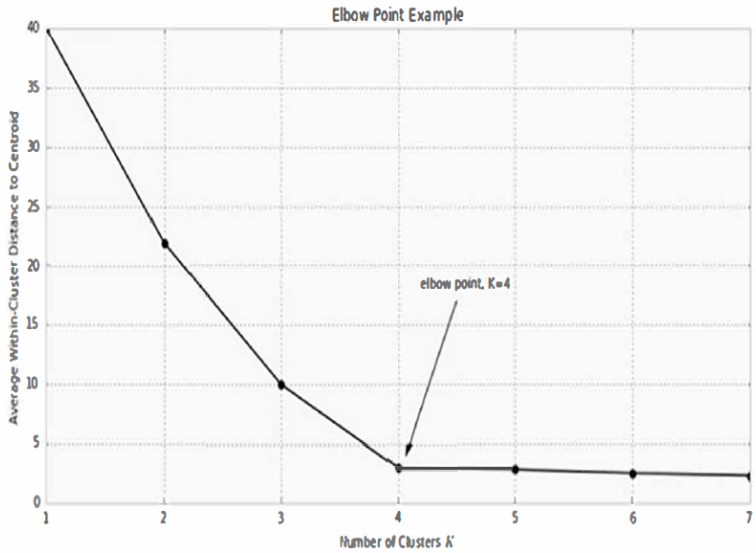
SSE improves in each iteration of K-means until it reaches a local or global minima

  - For each point, the error is the distance to the nearest cluster center
  - To get SSE, we square these errors and sum them.

- Solution to initial centroids problem
  - Multiple runs with different sets of initial random centroids
    - Choose the set with the least SSE
    - Helps, but probability is not on your side
  - Use hierarchical clustering to determine initial centroids
    - Take a small sample of points and cluster using hierarchical clustering
    - Extract K cluster centroids and use that as initial points
  - Select points that are farthest away from current centroids
    - Select the first point at random or take the centroid of all points
    - For each successive initial centroid, select the point that is farthest from any of the initial centroids already selected
    - Guaranteed to be random and well separated
    - Tendency to include outliers

- value of K
  - The K-Means algorithm is somewhat na"i"ve because it clusters the data into k clusters, even if k is not the right number of clusters to use.
  - In general, there is no accurate method to find the exact value of K, but a good estimate can be obtained using these methods.
    - Use domain knowledge to ascertain the right number of cluster by running Kmeans several times and compare the results
    - Use the within-cluster-sum-of-squares or inertia. (see next slide)
    - Others: cross-validation, silhouette method, information criteria, etc.

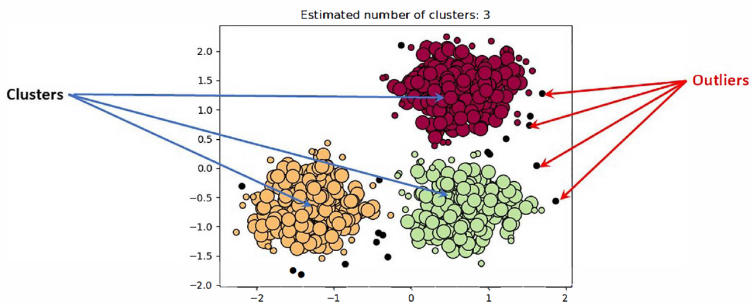
- Elbow Method



- 1. Run k-means for a range of values e.g. k = 2 ... 10
- 2. calculate the sum of squared errors (SSE) for each value of k
- 3. Plot a line chart for SSE-vs-K
- 4. Look for an elbow - that value of k is the best
- Note that the SSE will decrease towards Oas we increase K.
- Our goal is to choose a small value of k just before (elbow) it starts to show diminishing returns.

- Pros & Cons
  - The K-Means algorithm is guaranteed to converge to a result. However, the result may be a local optimum (i.e. not necessarily the best possible).
  - Clustering is a heuristic method and therefore much experimentation is needed to achieve the desired outcome.
  - Pro: Fast, easy, General clustering for all kinds of business problems.
  - Con: Sensitive to outlier/noise

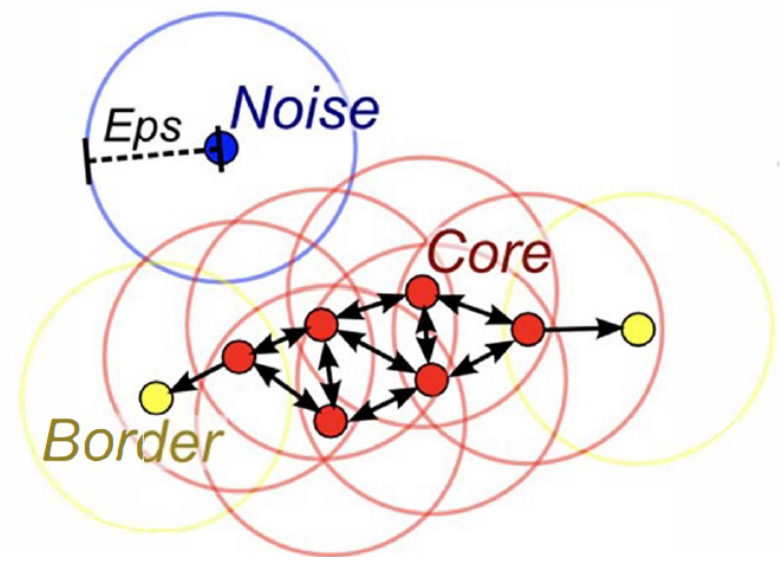
- Density-based Clustering
  - Density-based Spatial ,Clustering of Applications with Noise (DBSCAN)

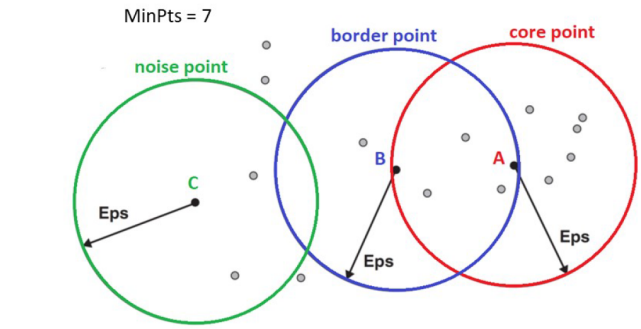


  - Brief
    - In DBSCAN, there are no centroids, and clusters are formed by linking nearby points to one another.

- There is the concept of noise points in the DBSCAN algorithm - points that do not belong to any clusters. These points are eliminated by the algorithm.
- In DBSCAN, there is no need to specify the number of clusters but does require specifying two parameters that influence the decision of whether two nearby points should be linked into the same cluster.
  - A distance threshold, Eps (e:-epsilon), and
  - A minimum number of points ("MinPts")
- In DBSCAN, only a single pass through the data is needed, and once a point has been assigned to a particular cluster, it never changes.
- DBSCAN is a simple, but effective algorithm for finding dense regions of objects that are surrounded by low-density regions.

- Basic structure

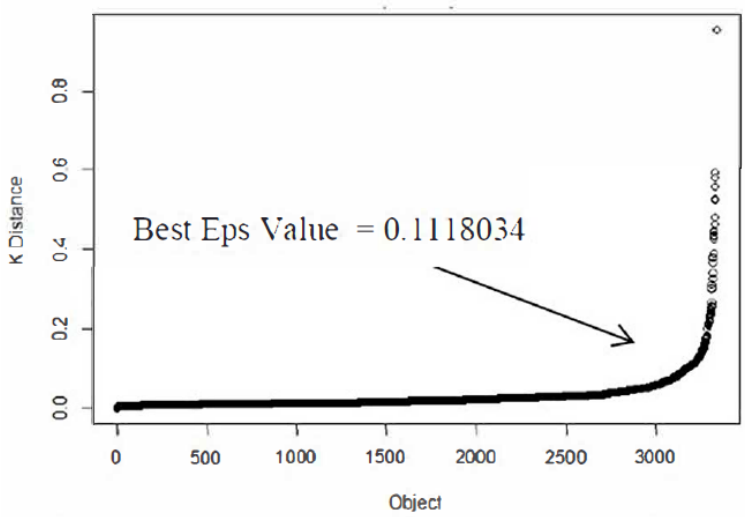




A is core because it has MinPts within its Eps neighborhood
B is a border point because it is within the Eps neighborhood of A but doesn't satisfy MinPts
C is a noise point because it is NOT within an Eps neighborhood and have no MinPts

- Density at point p = number of points within a specified radius (fps)
- A point is a core point if it has > a specified number of points (MinPts) within Eps

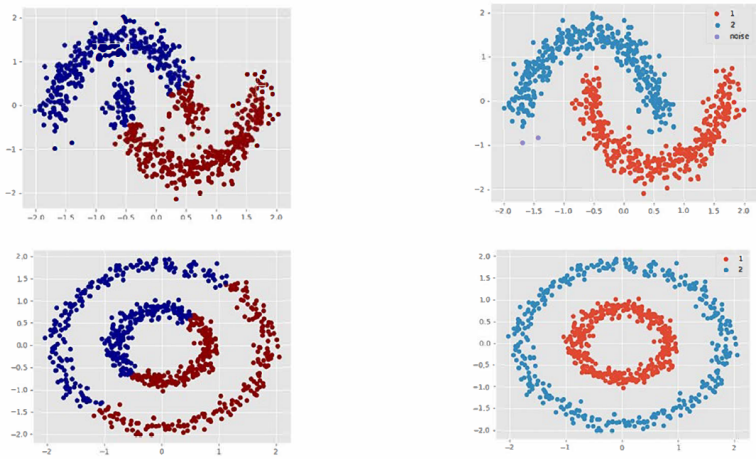
        - These are points that are at the interior of a cluster
    - A border point has fewer than MinPts within its Eps but is in the neighborhood of a core point
    - A noise point is any point that is not a core point or a border point
- Pipeline -> Start with N data observations
    - Step 1: Randomly select a point, P
    - Step 2: Retrieve all points that satisfy the Eps and MinPts criteria w.r.t. to P Step 3: If Pis a core point, a cluster is formed
    - Step 4: Check if P is a border point, attach to core point
    - Step 5: Make each group of connected core points into a separate cluster. Step 6: Assign each border point to one of the clusters of its associated core points
- Pros and Cons
    - Does not require you to specify the number of clusters
    - Can find arbitrarily shaped cluster even if the cluster is completely surrounded by a different cluster
    - Able to detect noise and is robust to outliers
    - Pro: Arbitrary shaped data, robust to noise
    - Con: require sufficient high-density areas, not popular in business
- Parameter Estimation
    - MinPts
        - As a rule of thumb, a minimum minPts can be derived from the number of dimensions D in the data set, as minPts > D + 1.
        - Note that if minPts < 2, the result will be similar to hierarchical clustering with the single linkage, with the dendrogram cut at height E.
        - Therefore, minPts must be chosen at least 3.
        - As a rule of thumb, minPts = 2·dim (Sander et al., 1998) can be used but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.
    - Epsilon value of E, clusters will merge and the majority of objects will be in the same cluster.



Points sorted by distance to the 3rd nearest neighbor

- The MinPts can be used to estimate the Epsilon
- We calculate the average distances between each point ana its k nearest neighbors, where k = the MinPts.
    - The average k-distances are then plotted in
    - ascending order on a graph.
    - If E is too small, a large part of the data will not be clustered; whereas for a too high
    - The optimal value for E can be found at the
    - In general, small values of E are preferable, and as a rule of thumb only a small fraction of points should be within this distance of each other.

- Density-Based vs Centroid Based
    - K-Means is a Centroid-based clustering algorithm because data are assigned to the nearest centroid to form new clusters.
    - One of the key strengths of K-Means is that it is easy to understand and efficiently used in practice.
    - However, K-Means is highly sensitive to outliers, and can sometimes cause undesirable effects.
    - It is possible that outliers are assigned to a nearest cluster where it is not supposed to.
    - Outliers can also impact the centroid as it gets pulled towards these extreme values
- Kmeans vs DBSCAN



- Evaluating the Validity of Clustering Models
  - Hopkins Test for Clustering Tendency

Calculate distances:

a. d_X: $p_1, \ldots, p_m$ to points in X

b. d_U: $q_1, \ldots, q_m$ to points in X

Hopkins statistic:

$$H = \frac{\sum_{i=1}^{m} d_{U,i}}{\sum_{i=1}^{m} d_{X,i} + \sum_{i=1}^{m} d_{U,i}}$$

$H \approx 1$: data highly clustered

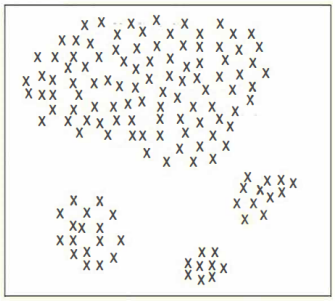
$H = 0.5$: data random

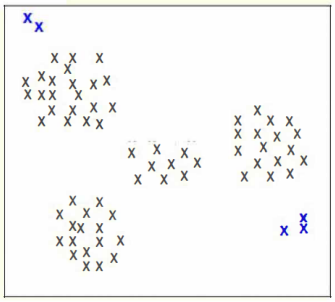
$H \approx 0$: data uniform

    - The Hopkins statistic (Lawson and Jurs 1990) is used to assess the clustering tendency of a data set by measuring the probability that a given data set is generated by a uniform data distribution.
    - Randomly select p points from your dataset, X
    - Randomly generate q points with same range as X
    - P and q have the same number of points
  - Cluster Validation are procedures used to evaluate the goodness of cluster model results. This is important because clustering is an unsupervised learning method for knowledge discovery.
  - Clusters are meant to uncover patterns and provide insights for decision making, and therefore clusters must be valid in the eyes of the stakeholders - sometimes even when cluster statistics does not provide "strong" support
  - Generally there are TWO approaches:

- Examining and Evaluating Cluster solutions (Technical)
- Understanding the Clusters through Profiling (Business)

- Valid Solution
  - Examine the number of clusters and their relative sizes





    - If a single cluster contains majority of the data (i.e. dominates), it could imply the need for further clustering. Bear in mind that it could also reflect the true profile of your customers
    - If there are very small clusters, it could mean that they represent outliers. This should warrant further investigation.
  - Examine the cluster cohesion.

$$cluster\ SSE_i = \sum_{x \in C_i} (x - c_i)^2 \qquad total\ SSE = \sum_i \sum_{x \in C_i} (x - c_i)^2$$

    - A good clustering solution is composed of dense concentration around the centroids. Large dispersion values indicate non-homogenous groupings and suggests further partitioning of the dataset.
    - The following statistics are commonly used to evaluate cluster cohesion:
      - Standard Deviation and pooled Standard Deviation
      - low variability and small values denote increased cohesion.
      - Maximum (Euclidean) Distance from cluster centroid
      - This radius represents the range of each cluster; the remotest member.
      - Sum of Squared distance between member and its cluster centroids
  - Examine the cluster Separation.
    - A good clustering solution is composed of well-separated cluster centroids. Large dispersion values are desired.
    - The following statistics are commonly used to evaluate cluster separation:

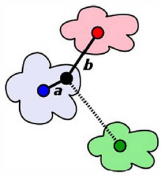
$$\text{Average SSB} = \frac{1}{N} \sum_{i \in C} N_i * dist(c_i, c)^2$$

- Sum of Squared distance between cluster centroids to the overall centroids
- where ci is the centroid of cluster i, c the overall centroid, N the total cases, and Ni the number of cases in cluster i.
- The SSB is directly related to the pairwise distances between the centroids: the higher the SSB, the more separated the derived clusters.

- Examine the cluster Silhouette Coefficient.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

a(i) = distance between i from its centroid (cohesion)
b(i) = distance of i to the nearest cluster (separation)



The overall silhouette coefficient is the average of the data silhouettes

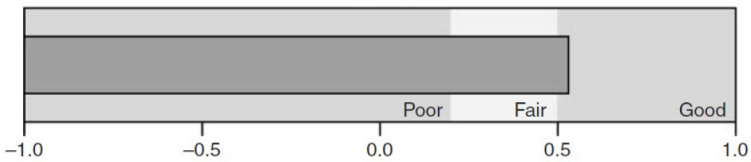
$$S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

- Silhouette coefficient is a combined measure that assesses both the internal cohesion and the external separation of a clustering solution.
- output

Model Summary

Algorithm	K-Means
Input Features	5
Clusters	6

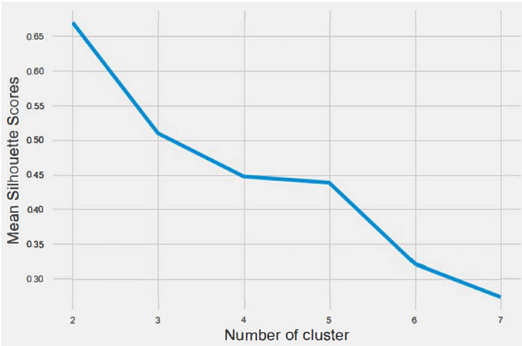
Cluster Quality



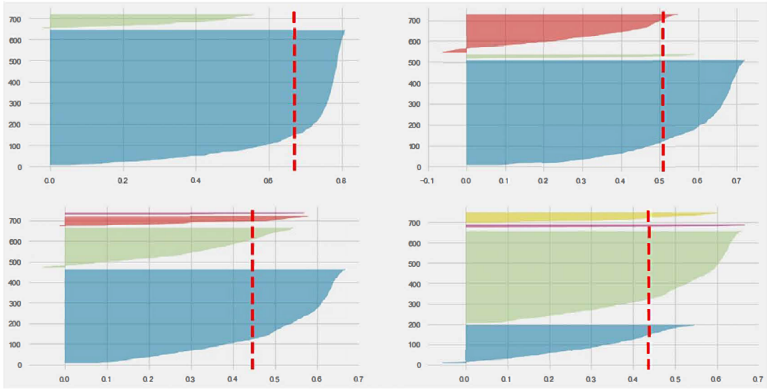
Silhouette measure of cohesion and separation

- Average Silhouette Scores used for deciding K

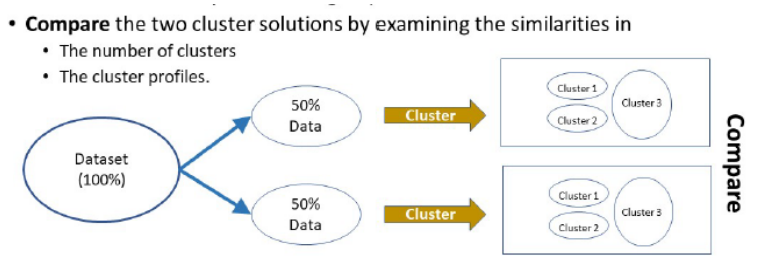
Clusters	Silhouette
2	0.6701
3	0.5098
4	0.4475
5	0.4386
6	0.3215
7	0.2739



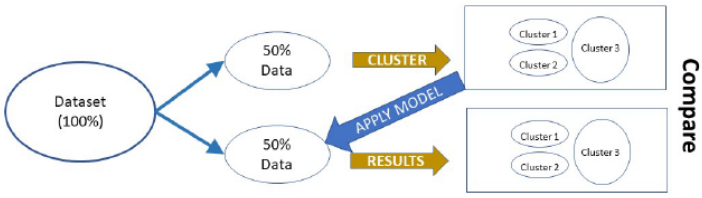
- Cluster Silhouette Plots



- Cross-validation

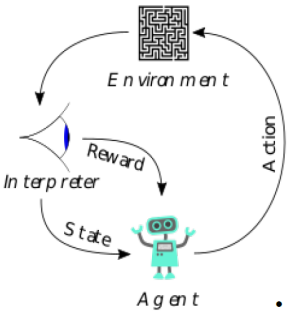


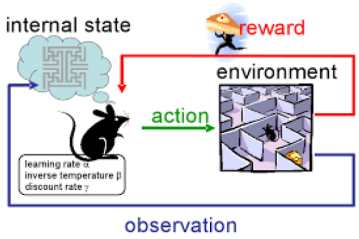
- Second method is to split randomly the clustering data set into two parts
- Development and Validation (50:50)
- Perform cluster analysis on development sample **only**
  - Create cluster profiles
  - Validate the results on the validation sample using development sample clustering algorithm (which becomes the implementation code)



- profile
  - In general, you want to give meaningful labels to each cluster to highlight their distinct characteristics.
  - Some points to consider:
    - Can the clustering be explained in practical terms?
    - Examine the value of each variable of the cluster centroid, (the cluster profile) of each cluster.
    - Look at the distinguishing characteristics of each cluster's profile and identify substantial differences between clusters
    - Cluster solutions failing to show substantial variation between clusters indicate that this may be a spurious clustering
  - A clustering solution is justified only if it satisfies the following:
    - Actionability
    - Potential Business Value
    - Interpretability
    - Ease of use

- Reinforcement learning

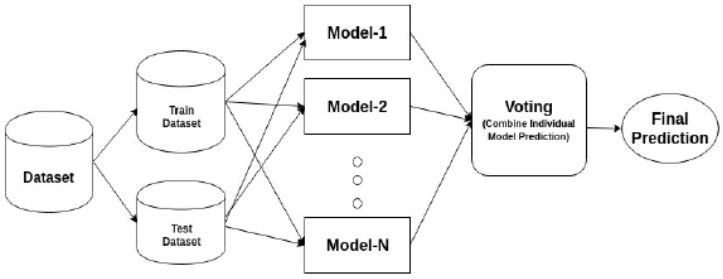




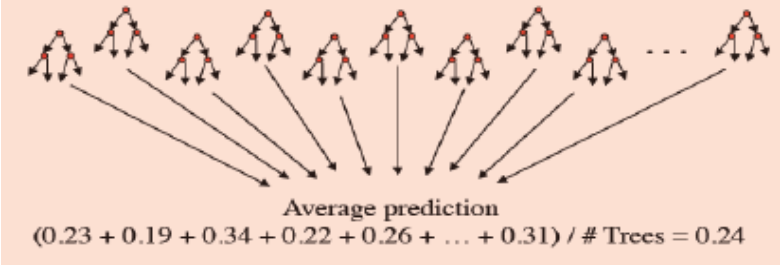
- A reinforcement learning agent interacts with its environment in discrete time steps.

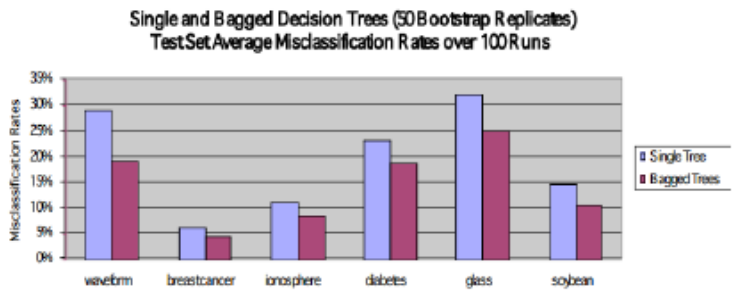
  - No desired category is given but the teacher provides feedback to the system such as the decision is right or wrong
  - The assumption is that your data is not uniformly distributed but rather they comprise of natural clumps.

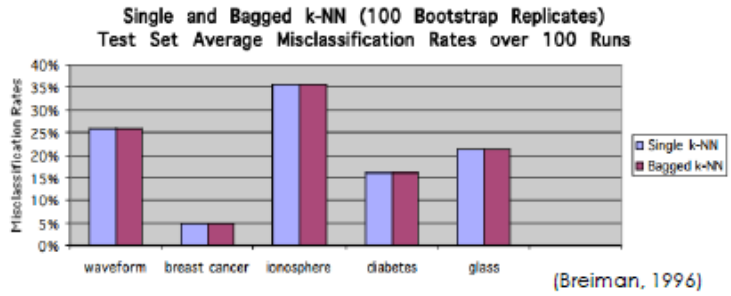
- Ensemble & Hybrid
  - Ensemble



    - The term ensemble is usually reserved for methods that generate multiple hypotheses using the same base learner
    - Ensembles tend to yield better results when there is a significant diversity among the models
    - create model diversity
      - Manipulating the training data (e.g. bagging)
      - Manipulating the input features
      - Varying the classifier type, architecture
    - Method
      - Bagging: Bootstrap Aggregating



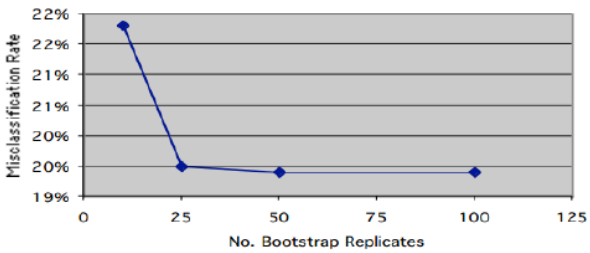




(Breiman, 1996)

- Draw N (say 100) bootstrap samples (sampling with replacement) from the training data, Train models (eg. NN, decision tress) on each sample
- Algorithm:
  - Randomly draw 67% (say, two thirds) of the training data
  - Train a model on this sample
  - Repeat this N times to get N models
- Take the un-weighted average prediction of all models
- How Many Bagged Models Are Required?

  Example from the soybean data set:



  Depends on data and problem, but generally, < 50 models should work well and often < 25 is adequate

- Error

  Assume we build a prediction model $\hat{f}(X)$ to estimate a function $f(X)$ where X is the set of input variables and Y is the variable to be predicted

  Then the expected squared prediction error at point x is:

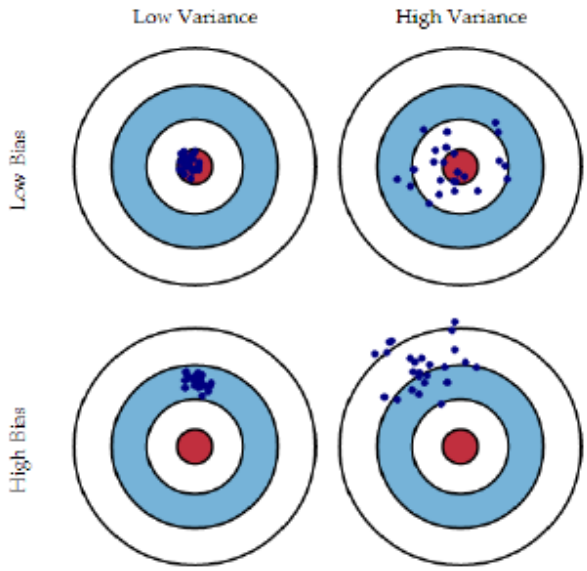
$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

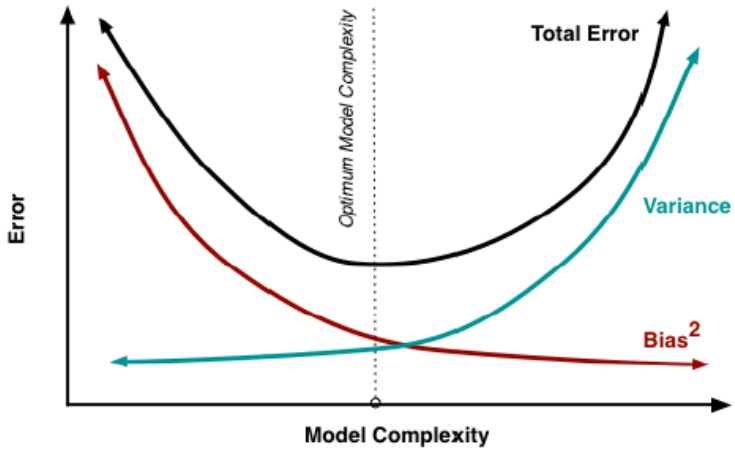
  We can decompose this into 3 components

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

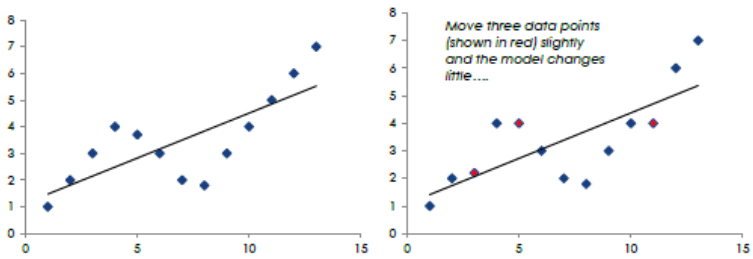
$$Err(x) = \text{Bias}^2 + \text{Variance} + \underbrace{\text{Irreducible Error}}_{\text{Noise, cannot be reduced by the model}}$$

- Bias versus Variance

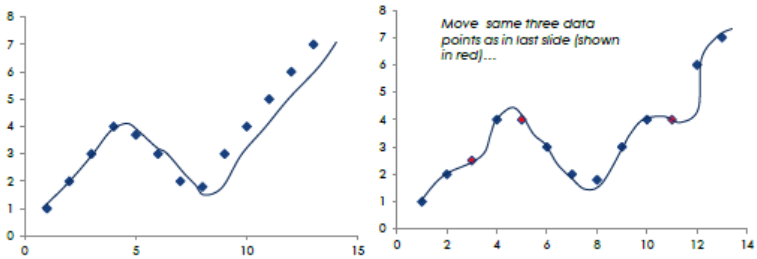




  - Error due to Bias:
    - The difference between the expected (or average) prediction of the model and the correct value
  - Error due to Variance:
    - The variability of the model prediction for a given data point
  - under-fit data



    - High bias ~ the model doesn't fit the training data very well
    - Low variance –removing/changing a few training data points won't change the model or predictions much
  - over-fit data



- Low bias ~ the model fits the training data very well
- High variance –removing/changing a few training data points can change the model and hence the predictions a lot

- Reduce Variance Without Increasing Bias
  - Averaging reduces variance:

$$Var(\overline{X}) = \frac{Var(X)}{N}$$

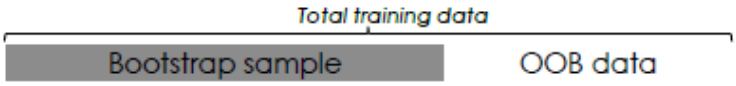
  - Average models to reduce model variance
  - For large N, residual model error mainly due to bias!
  - In practice:
    - models are correlated, so reduction is smaller than 1/N
    - variance of models trained on fewer training cases is usually larger
    - only works with some learning methods: very stable learning methods have such low variance to begin with, that bagging does not help much.

- Limits -> hurt
  - Each base classifier is trained on less data
    - E.g. only about 67% of the data points are in any one bootstrap sample
  - If data is poor, then losing this much data can hurt accuracy
  - Bagging usually helps, but sometimes not much

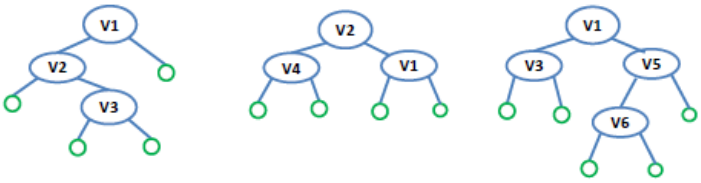
- Random Forests
  - Brief
    - Draw 1000+ bootstrap samples of data
    - Draw random sample of available features at each tree split
      - Randomisation(hence model diversity) now occurs in two places:Random sampling of training data + Random sampling of feature set
      - Training speed increases due to less computation at each tree split (less features to evaluate the splitting cost function for)

- Train trees on each sample/attribute set -> 1000+ trees
- Use un-weighted voting to get final prediction (as with bagging)

- Testing
  - No need for separate test set
  - Method:
    - Test each tree against the data left over after the bootstrap sample was taken; this is called the OOB (out-of-bag) data

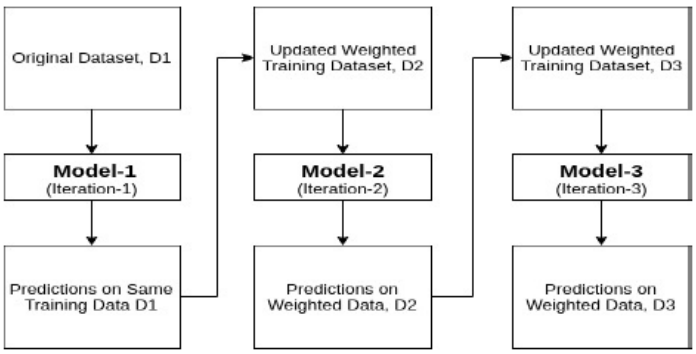
    - If each bootstrap sample takes 67% of the training data, then after building T trees every training example will have been OOB (and hence a valid test example) for about T/3 times.
    - For each training example, take the majority vote of all T/3 test predictions to get the forest's prediction* and compare with the actual class value. Output 1 if forest prediction != actual, else output 0
    - Sum over all training examples to get error estimate for the forest
    - * For regression problems, average all T/3 test predictions to get the forest prediction. Then compute MSE as Σtraining examples (prediction –actual)**2
- Measuring Variable Importance
  - For a single tree the order in which the variables occur in the tree is a measure of their relative importance to the prediction

  - For a forest?
    - Naïve method: Count the number of times the variable occurs in all of the trees, more occurrences => more important
    - Better method: sum the total reduction in impurity (eg. the decreases in the Gini index) for all nodes that test the variable
  - Permutation Method
    - Randomly shuffle the values of a given input variable to "break" the bond of the variable to the response. Then the difference of the model accuracy before and after the shuffling is a measure of how important the variable is for predicting the response
    - Detailed Steps

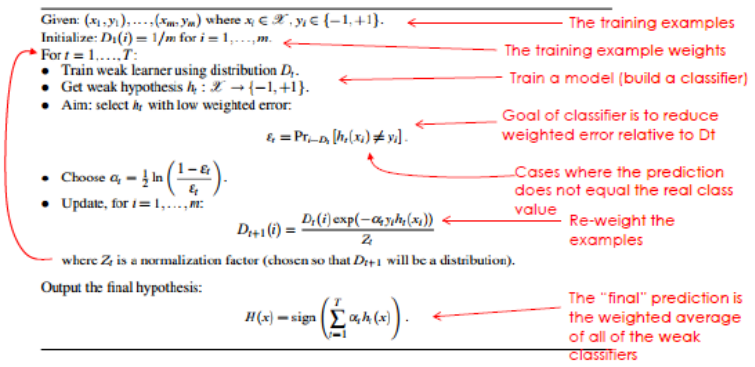
- 1.Test every tree on its own OOB examples. For each training example ecount the votes for the correct class (call this NormalCorrectVotese)
- 2.For each input variable v:
  - For each tree t:Randomly permute the values for vin the OOB examples and retest the tree
  - For each training example, count the votes for the correct class
  - $$\text{Importance}_v = \text{average}\left[\frac{\text{NormalCorrectVotes}_e - \text{ShuffledCorrectVotes}_e}{\text{TotalVotes}_e}\right]$$

- Pros and cons
  - Usually works better than bagging
    - robust to noise, easy to use, surprisingly high accuracy
    - but.. lots of trees means hard to interpret (becomes a black box)
  - Variance of RF trees is higher than Bagged Trees
    - typically needs 10X as many trees
    - trees should be (generally) unpruned (to encourage diversity)
    - RF needs 100's to 1000's
  - Extra parameter to tune: p(feat)
    - probability of getting to use feature at each split
    - fortunately, usually not too sensitive
    - Breiman suggests SQRT(N), N: total number of features
  - Unlike Bagging and Boosting, RF is for decision trees only
- Boosting

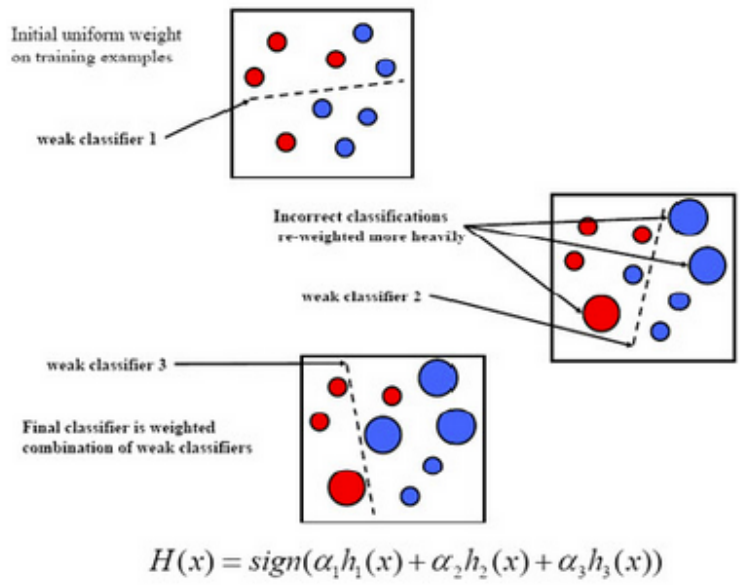


  - essentials
    - Build a model (but don't 100% over-fit the data!)
    - Increase weights ofthe training examples the model gets wrong
    - Retrain a new model using the weighted training set
    - Repeat many times…

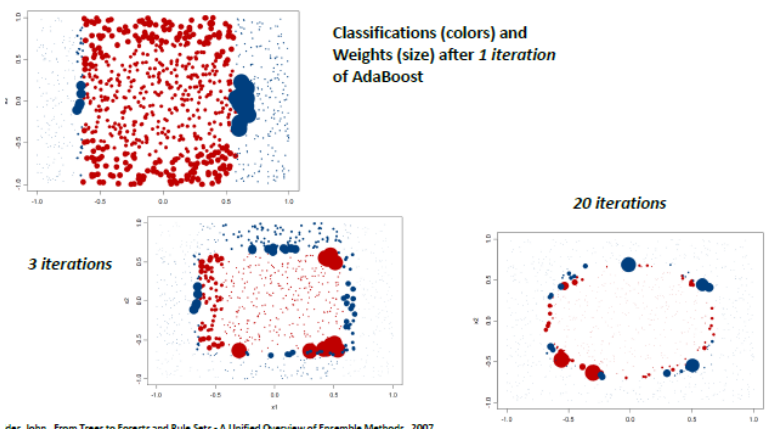
- Basic Boosting Algorithm
  - 1.Weight all training samples equally
  - 2.Train model on train set
  - 3.Compute error of model on train set
  - 4.Increase weights on train cases that the model gets wrong!
  - 5.Train new model on re-weighted train set
  - 6.Re-compute errors on weighted train set
  - 7.Increase weights more on cases it still gets wrong
  - 8.Repeat until tired (100+ iterations)
  - 9.Final model: weighted prediction of each model (aka base models)
- AdaBoost* (Adaptive Boosting)



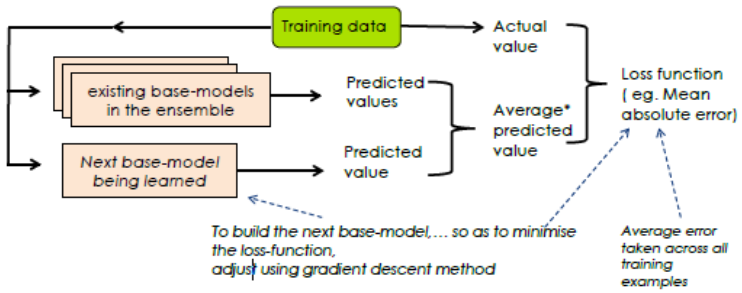
  - Conceptual



  - Results



- Application
  - Neural Networks –Scale learning rate by weight
  - Decision Trees –instance membership is scaled by weight
  - k-NN –node vote is scaled by weight
- Gradient Boosting



Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

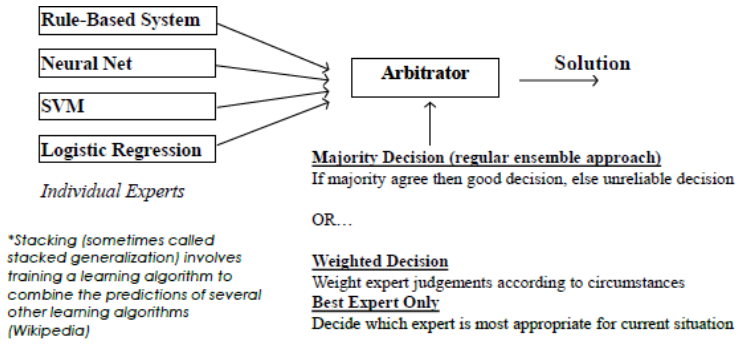
   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

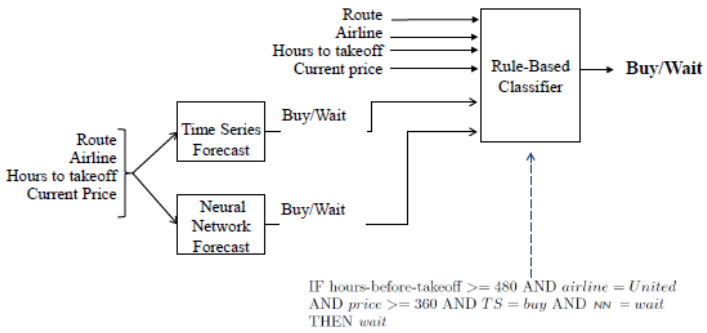
3. Output $F_M(x)$.

  - Treat learning as Gradient Descent optimization
  - Like Adaboost, models are learned and added to the ensemble sequentially
- Boosting versus Bagging/RF
  - In practice bagging/RF almost always helps
  - Bagging doesn't work as well with stable models
    - Boosting and RF might still help.
  - Often, boosting helps more than bagging
    - Boosting might hurt performance on noisy datasets
    - Bagging/RF don't have this problem.

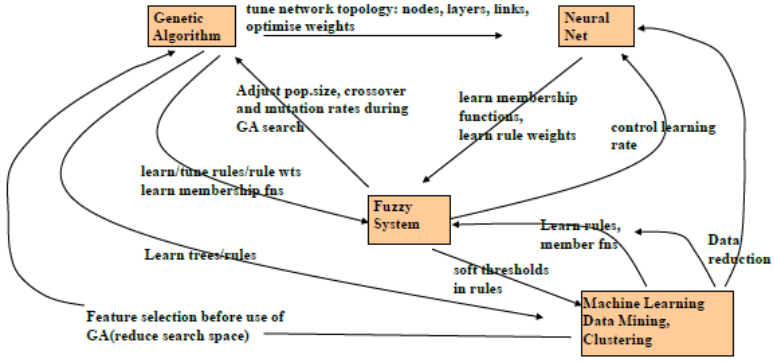
        - Bagging/RF is easier to parallelize
    - Summary
        - Using multiple models to reduce variance and increase accuracy
        - Works best if models don't agree with each other (need model variance)
        - Usually refers to multiple models of same type
        - Bagging & Boosting are the most popular generic methods
        - Random Forest increasingly popular
        - Ensemble Creation Approaches
            - A good goal is to get less correlated errors between models
            - Injecting randomness –initial weights, different learning parameters, etc.
            - Different Training sets –Bagging, different features, etc.
            - Forcing differences –different objective functions, auxiliary tasks
        - Ensemble Combining Approaches
            - Unweighted Voting (e.g. Bagging)
            - Weighted voting –based on accuracy, heuristics, etc. (e.g. Boosting)
            - Stacking -Learn the combination function
- Hybrid Machine Learning Models
    - Multiple Classifier Systems
        - In "Multiple Classifier Systems" approach, each classifier is an expert in certain situations
            - Each model type has different strengths and weaknesses
            - Usually have relatively small number of experts
            - Allows for more model combination methods apart from averaging
        - Different solution strategies (experts) offer alternative solutions. Another process decides which solution to accept or how to combine the solutions, e.g. majority vote algorithm.
        - This architecture is also known as stacking*



The Experts have same skills (Buy/Wait decision) but sometimes one is better than the other! The arbitrator helps decide which to use and when



- Self-Tuning Systems

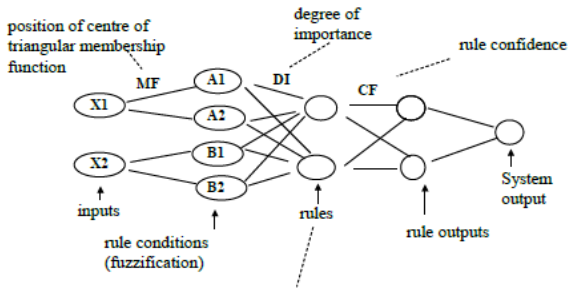


  - used to tune or learn the architecture for another
  - example
    - Neuro-Fuzzy Systems

      Neural Network is used to represent and "learn" a Fuzzy System

      Nodes represent rule inputs, conditions, actions etc

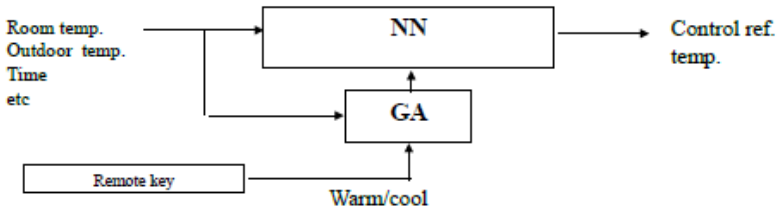
      Special training algorithm required



Taking only the strongest connection to each condition element yields rules like (other schemes exist):-

e.g. If X1 is A1 (DI1) And X2 is B1 (DI2) Then output = C (CF1)

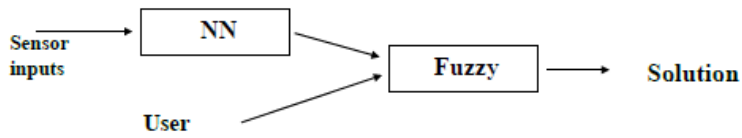
    - GA-Neural



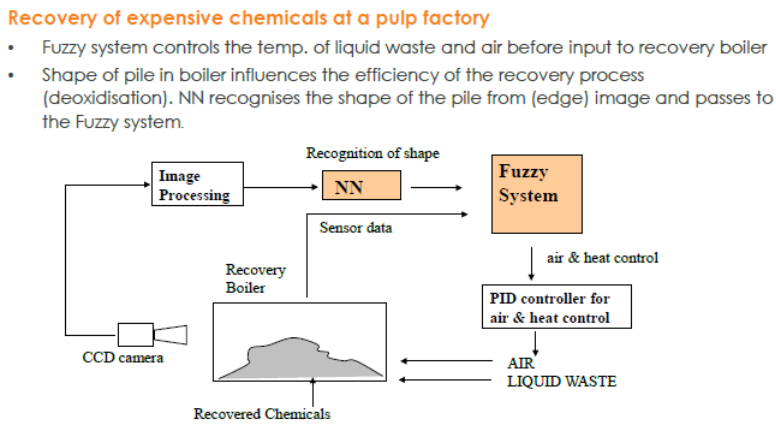
      - NNs are generated/tuned by GAs
      - GA chromosome represents NN topology
      - number of hidden layers, hidden nodes and number of links and/or weights
      - LG Electric developed an air-con controlled by an NN

- If the user wants the air-con to adapt to their preferences then a GA is used to change the number of neurons and weights
- Pros: GA can avoid local minima more than back-prop
- Cons: size of chromosome gets prohibitively large if topology is being learned

- Co-operating Experts



  - "Pooled Expertise"
    - Different techniques work together as a team to produce a single solution, no single technique/expert is sufficient alone
    - E.g, NN provides input to Fuzzy System
  - example



- Ensemble & Hybrid
  - Both ensemble models and hybrid models make use of the information fusion concept but in slightly different way.
  - Ensemble Models -multiple but homogeneous, weak models are combined
  - Hybrid Models combine different, heterogeneous machine learning approaches.
  - They both may considerably improve quality of reasoning and boost adaptivity of the entire solutions.