As a basis I am using the 33-item sheet given out by Boris listing the general topics covered by the course.

One-dimensional data can be displayed in different ways, for different effects. A histogram can show the "shape" of data distribution. A pie-chart can show the proportions of data as a part of the whole. *Box plots* and *stick plots* are other way to display such data. **Two-dimensional data** can be displayed using a **scatter-plot**.

Entity-to-feature tables
All information held about entities – even their names – can be expressed as features of the entity. Values of **features** can be **quantitative** (i.e. possess a numerical value and it is meaning to take the average of such values), or **categorical** (i.e. hold one of a number of non-quantitative values). Categorical values can be **binary** (i.e. hold one of two distinct values, 1-or-0, yes-or-no, true-or-false, male-or-female) or **nominal** (i.e. hold one of several distinct, unordered values, similar to a C *enum*). Furthermore, all binary and nominal features can be converted into 2 or more binary features with a 1 representing the presence of that feature and 0 the absence of that feature.

Pre-processing and standardising data prior to analysis
This process transforms a raw entity-to-feature table into a quantitative matrix. Firstly, categorical data must be transformed to a quantitative format as described above. Secondly, feature-columns of data must be standardised to make them comparable; this is done by *shifting* (the origin) and *scaling*. The scale factor can be one of a number of values, e.g. SD or range. Once **standardised**, features become comparable. Subtracting feature averages (grand means) centres the feature around 0.

2D Linear Regression (aka **'line of best fit')** attempts to model a relationship between two measures by finding the line of best fit uses the least squares criterion for minimising residual errors. The **correlation coefficient** ρ (rho) has a value between -1 (inverese correlation) and 1 (perfect correlation), with 0 indicating no correlation. For example, at $\rho = 0.9$ the unexplained variance of y constitutes $1 - \rho^2 = 19\%$ of its original variance. The square of the correlation coefficient, ρ^2, is called the **determination coefficient**. The slope a is proportional to ρ. When $\rho = 0$ the slope is zero too. With a good correlation, the y value of a new data pair can be predicted with confidence from the x value.

Bootstrapping: the method of re-sampling for the purpose of gaining confidence in measures already obtained (correlation or regression coefficients).

Distance and **Inner product:**
Each entity, being a row of features, can be called a *vector*. The set of vectors is called the *feature space*. The **distance** between any two vectors is the sum of the square of each of the components of the vectors, i.e. $d(x,y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_m - y_m)^2$
This is also known as the **Euclidean squared distance**.
The **inner product** is the sum of the product of each of the corresponding components of the vectors, i.e. $(x,y) = (x_1.y_1) + (x_2.y_2) + \ldots + (x_m.y_m)$
The *distance* and the *inner product* are closely related.

Centre points:
mean average (also called *grand mean*), midrange (midway between highest and lowest values), median (middle value), mode (most popular). The **median** is the most stable, but requires sorting the data, as does the midrange. Mode is most useful when distribution is far from uniform.

Spread:
Several characteristics have been defined to measure feature's dispersion or spread, the simplest of which is **range** (difference between minimum and maximum values). Others include **variance** (**s**^{2}) which is the average squared deviation of all value from the *grand mean*, and **standard deviation** (**s**), which is simply the square root of the variance. The standard deviation is zero if and only if all the values are equal to each other.
Remember, for a sample *n*, divide *s* and s^2 by $n - 1$, for data set, divide by *n*.
For any feature, its standard deviation is at least twice as small as its range. Proof: max. variance when data is 0 & 1, is $p(1 - p) = ¼$, $\therefore$ max. SD = $\sqrt{¼} = ½$. QED.

Co-variance measures similarity across **2 dimensions**, i.e. their variable with respect to **each other**. A *co-variance matrix* (*n* x *n*) can be generated to contain all the co-variances between *n*-dimensional data. This matrix is symmetric about its diagonal since $cov(x,y) = cov(y,x)$.

It is possible to generate **eigenvalues** and **eigenvectors** for the co-variance matrix. The eigen*vectors* with the highest eigen*values* are the **principal components**. From these principal components the new data set can be derived (with reduced dimensionality, i.e. compressed).

Principal component analysis (**PCA**) is one of the oldest multivariate methods of data reduction. It transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.
The **objectives of PCA** are (1) to reduce the dimensionality of the data set while retaining as much data as possible, and (2) to identify new meaningful underlying variables. Once patterns have been found, PCA can compress the data with little loss of information.
The first principal component is the combination of variables that explains the greatest amount of variation. The second principal component defines the next largest amount of variation and is independent to the first principal component. And so on.
For example, with 1000 entities and 50 features, we wish to derive 2 vectors of length 1000 and 50 which, when multiplied, get us as close to the original values as possible, i.e $x_{iv} \approx z_i.c_v$ The principal components (total length: 1,050 values) may provide 90% of the accuracy of the original data set (total length: 50,000 values). By adding the residuals, we get $x_{iv} = z_i.c_v + e_{iv}$ (and we can continue the process of reducing e_{iv} if necessary for greater accuracy/reduced error).
This technique can be used in image processing for compression and noise reduction, as well as for signal processing, in similar ways, analysis of gene expression data, classification of land use from satellite photographs, and in neural network-based facial recognition.

Similar to PCA is **singular value decomposition** (**SVD**). It is an amazing and useful fact that every $m \times n$ matrix has a singular value decomposition. The singular value decomposition for a matrix A rewrites A as the product of (hanger)(stretcher)(aligner) matrices. SVD is good for *data compression*, by reducing a large matrix (e.g. an image) to three smaller matrices. It also has applications in *signal processing* and *statistics* and language analysis.
Notationally, $A_{n \times p} = U_{n \times n} \cdot S_{n \times p} \cdot V^T_{p \times p}$ where $U^T U = I_{n \times n}$ and $V^T V = I_{p \times p}$ and S is a diagonal matrix whose values are called the *singular values*.

Clustering: Clustering is a discipline aimed at revealing groups or clusters of similar entities in data. Typically, a clustering application involves the following five stages: (a) developing a data set, (b) data pre-processing, (c) clustering data, (d) interpretation of clusters, (e) drawing conclusions.

***K*-means clustering** is a major clustering technique. It is computationally easy, fast and memory-efficient. Its main problems lie with initial selection of cluster numbers and centroid locations. It partitions the data set into *K* non-overlapping clusters; the means are aggregate representations of clusters, or centroids. The **minimum distance rule** assigns entities to their nearest centroids according to Euclidean squared distance. Centroid centres are updated. The process is reiterated until clusters centres are stable. Pre-processing includes transforming data into quantitative values and normalizing it to remove weighting (if desired).

Initial seed locations can affect not only the speed of convergence but also the final results. This is a big subject in itself, and there are several methods of selecting K and initialising centroids:
(i) **randomly** select *K* entities; (ii) randomly select *K* vectors within the feature space (i.e. they need not contain actual entity values); (iii) Starting with a larger value of *K*, gradually reduce *K* until a marked difference in the results is seen. Use *K+1*;
(iv) **MaxMin** approach: select the two entities farthest apart and call these the set of initial centroids. Repeatedly add to this pair those entities farthest away from the entities in this set until a stop condition is reached, which may be a pre-specified *K*, a distance threshold or a significant change in the distances calculated. This approach uses ad-hoc thresholds (i.e. guesses); (v) **reference point**-based clustering (**separate/conquer**). Choose an 'optimal' reference point (e.g. the grand mean). Shift the space origin to this point. Choose the most distant entity from the reference point and calculate K-means with these two, keeping the reference point centroid unchanged. The deviate centroid is the anomalous cluster. By removing these entities and repeating the process, clusters can be identified. Can select only the most populous clusters, for example.

Square Error Criterion is the function which sums the square distance of the error within a cluster and across all clusters. Minimizing this function is the aim, since the smaller the result of the function, the closer the clusters are to their centroids. Interpretation: (i) look for typical representatives in a cluster, i.e. those entities closest to the centroid; (ii) look for the most salient features of a cluster; (iii) describe clusters in terms of their most salient features.

There are two approaches to building a cluster hierarchy: (a) agglomerative, building from the bottom-up by merging smaller clusters to make larger clusters; (b) divisive, splitting greater clusters from the top-down, starting with the entire data set.

A spanning tree is a subgraph that is a tree and connects all the nodes together. A **minimum spanning tree** (**MST**) or minimum weight spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree. Uses include finding minimum cost (e.g. for laying pipes, telephone wires, transportation, etc).
Prim's algorithm (R. C. Prim) is one of three classical 'greedy' minimum-spanning tree algorithms (the others being Kruskal's and Boruvka's algorithms):

```
select an arbitrary vertex (node) to start
while (there are fringe vertices)
    select minimum-weight edge between tree and fringe (if a tie, choose either)
    add the selected edge and vertex to the tree
```

MSTs are useful because (a) they create a sparse subgraph that gives information about the original graph; (b) they provide a way to identify clusters – deleting the long edges leaves natural clusters.

One of the simplest **agglomerative** hierarchical clustering methods is **single linkage**, also known as the nearest neighbor technique. These clusters may be *chains*. The algorithm is thus: Start with each item assigned to its own cluster. Then join the closest two clusters. Repeat until all items added to a single cluster (of course there is no point in having all the N items grouped in a single cluster but, once you have got the complete hierarchical tree, if you want *k* clusters you just have to cut the *k-1* longest links.). Crucially, we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. This is *single-linkage* (*Full- or Complete-linkage* considers the distance between two clusters as the *greatest* distance from any member of one cluster to any member of the other cluster. *Average-linkage* not surprisingly considers the distance between two clusters as the *average* distance from any member of one cluster to any member of the other cluster).

A **minimum spanning tree** allows for finding the **single-linkage** hierarchy by a **divisive** method: sequentially cutting edges in the MST beginning from the largest, in decreasing order.

Hierarchical clustering. There are two approaches, agglomerative and divisive. The one-entity clusters (singletons) are terminal nodes, the universal cluster is the root of the tree; other clusters are nodes of the tree. Heighted trees convey information about the *tightness* of clusters through the height of each node.

Ward agglomerative clustering is uses a weighted group average calculation to merge related clusters. The *Ward distance* (*dw*) between two clusters (*S*) is defined as the product of the cardinalities (*N*) of two clusters divided by their sum, multiplied by the Euclidean distance between their centroids (*c*), i.e.

$$dw(S_1, S_2) = \frac{N_1 \cdot N_2}{N_1 + N_2} \cdot d(c_1, c_2)$$

Method: starting with single entity clusters, compute the Ward distance between all clusters. Merge the two with the smallest Ward distance. Recompute distances. Repeat until merged into a single cluster. Heights are defined as the square error of the clusters, which are added to those of their children, meaning they grow exponentially high. Cutting the tree at long edges provides suitable numbers of clusters for *K*-means.
Key point: Ward agglomeration can be used to identify the initial setting for *K*-means.

Ward agglomeration, *unlike K*-means, is computationally intensive and *does not scale well*. **Note:** Ward agglomeration and minimum spanning tree are similar and achieve similar aims.

Artifical **Neural Networks** are helpful when (a) we can't formulate an algorithmic solution, (b) we can get lots of examples of the behaviour we require, (c) we need to pick out structure from data.
A **perceptron** is a single-layer neural network (invented in 1957 by Frank Rosenblatt). Perceptron's can only classify linearly separable sets of vectors, i.e. a straight line or plane needs to separate the vectors into their correct categories (1969, Marvin Minsky & Seymour Papert). Thus AND and OR can be solved by perceptrons, but XOR cannot. These limitations can be overcome by **multilayered perceptrons** (**MLP**) with **back error propogation**, whereby errors can be fed back to hidden layers of neurons, enabling the weights to be adapted using any *gradient-based optimization* method, that is, look for the steepest ascent/descent. **Instance processing** refers to the useful feature that each neuron need only process its own data, either being fed-forward from input layers, or fed-back from output layers.
For **supervised learning**, networks adapt by changing their weights by an amount proportional to the difference between the actual output and the desired output. A *learning rate* defines the actual rate of change. This is called the **perceptron learning rule**. The learning rate must be well chosen. A dynamic learning rate would adapt as the weights converged to their optimum.

Neural networks have been applied to expert systems, speech recognition, financial market predictors and medical diagnoses.

Nature-inspired algorithms include Genetic Algorithms (**GA**s), evolutionary algorithms (**EA**s) and Swarm optimisation.

GA Terminology: The data set is called a **population**. Entities, or vectors within the data set are called **chromosomes**. Analogies of biological mechanisms are applied to the population, namely **selection** (pairing of chromosomes), **cross-over** (algorithmic changes) and **mutation** (random changes). Given a mechanism for determining the *fitness* of the population, the mechanisms are iteratively applied until a stop condition, or threshold, is reached. *Selection* can be made randomly or according to probability; *crossover* can be made with a fixed probability (e.g. 0.8) at a random point on the chromosome; *mutation* can be made using a fixed, small probability (e.g. <0.01) or a dynamic one. An additional step, **elitist survival**, suggests storing the best fitting chromosome to avoid it being crossed-over or mutated.

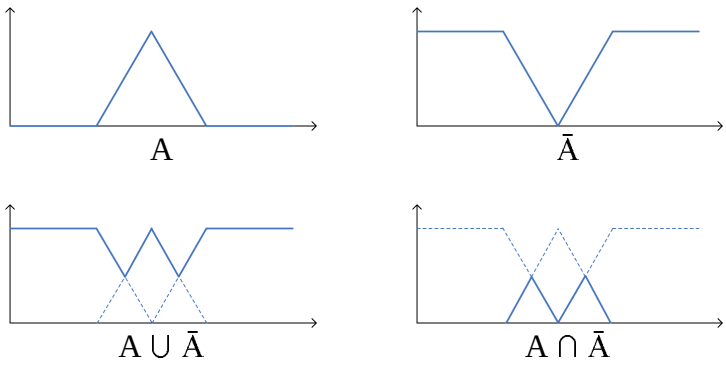
GA *K*-means: for clustering of N entities, randomly generate a population P of vectors of N integers, each integer i, $1 \leqslant i \leqslant K$. Apply GA to this population.

EA *K*-means: for clustering entities with v features, randomly generate chromosomes of $K \cdot v$ real numbers, which represents the centroids $c_1, c_2, \ldots, c_k$. The length of the chromosome is not dependent on the number of entities, an advantage over GA *K*-means above. Cross-over and mutation apply as with GA. **Differential evolution** is a variation on EA whereby cross-over and mutation are merged into a single step involving four chromosomes at a time.

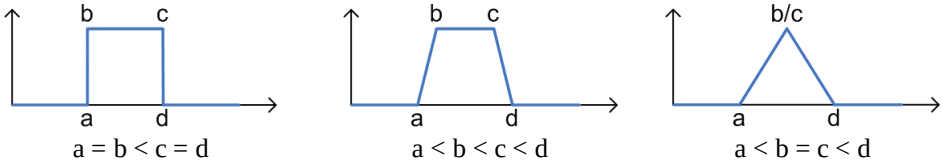
Particle swarm optimisation has no biological roots. Instead of a population and chromosome, the terms **swarm** and **particle** are used. Particles do not cross-over or mutate but do move in a seemingly random fashion, yet provide order within the overall swarm. Compare with the Brownian motion of smoke particles, the movement of shoals of fish, of swarms of wasps or herds of animals, or the results of elections or 'ask the audience' in the TV quiz show *Who wants to be a Millionaire*.

Fuzzy Sets are good for modelling *uncertainty*. Rather than using absolute values, a fuzzy set, valid to some degree over an **interval**, is used. A fuzzy set is expressed by a **membership function**, which can have any shape, although triangular and trapezoidal are common shapes. Fuzzy sets differ from probabilistic distributions whereby the latter indicate the chance of an entity being part of a set. In fuzzy logic, an entity *can be* just partly in a fuzzy set.

Set theory applies to fuzzy sets:
Union: $A \cup B \rightarrow \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$
Intersection: $A \cap B \rightarrow \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$
Complement: $\bar{A} \rightarrow \mu_{\bar{A}}(x) = 1 - \mu_A(x)$



Square (*crisp*, or *interval*), trapezoidal and triangular fuzzy sets can be defined by four (or fewer) axis points:



Applications for fuzzy sets include car navigation systems, information retrieval systems, automatic train controllers, robot arc-welders and graphics controllers for automated police sketchers.

A **central fuzzy set** is a single fuzzy set derived from a set of fuzzy sets. For example, with a set of triangular fuzzy sets, each defined by a triplet, the central set could be derived by calculating the grand mean for each element of the triplet, i.e.

$$L(\bar{a},\bar{b},\bar{c}) = \Sigma_i a_i/N,\ \Sigma_i b_i/N,\ \Sigma_i c_i/N.$$

This can also be expressed as the minimal value for the function

$$L(a,b,c)=(\Sigma_i(a_i-a)^2+\Sigma_i(b_i-b)^2+\Sigma_i(b_i-b)^2)/N$$

A central fuzzy set is distinct from the centre of a fuzzy set, which can be calculated in many ways, e.g. centre of gravity or centre of mass.