1. Pattern Classification/Clustering

Fuzzy clustering involves entities being able to be a member of more than one cluster, to varying (fuzzy) degrees.

Fuzzy c-means clustering place *c* cluster centres and assign membership function depending on distance from cluster centre. The cluster centres are calculated, and the membership values recalculated. Repeat until stop condition is met.

Consensus clustering is the process of utilising several different clustering mechanisms and pooling the results to obtain a consensus.

Adaptive systems & personalised recommenders can be classified as collaborative (user) filtering (e.g. Amazon "customers who bought this also bought…"), or content-base (item) filtering (e.g. itemsets).

Be aware of cold-start problems (e.g. no customers!).

2. Fuzzy Logic

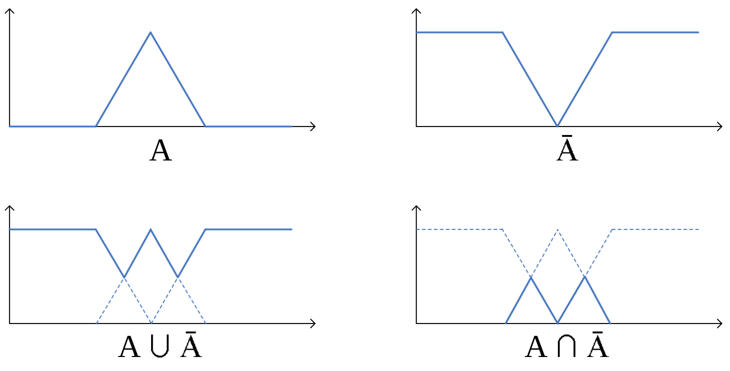
Fuzzy Sets are good for modelling *uncertainty*, and are a superset of Boolean logic. Rather than using absolute values, a fuzzy set, valid to some degree over an **interval**, is used. A fuzzy set is expressed by a **membership function**, which can have any shape, although triangular and trapezoidal are common shapes. Fuzzy sets differ from probabilistic distributions whereby the latter indicate the chance of an entity being part of a set. Probability is about the degree of certainty (i.e. the "chance") that something is 100% true or 100% false. With fuzzy logic, an entity can be partly true *and* partly false at the same time. The sum of probability adds up to 1 – not so in fuzzy logic where the sum of membership functions can be more than 1. For example, a car parked in a car park may be mostly in one place and partly in another.

There is plenty of **ambiguity** and imprecise language in real life, and in expert systems too, e.g. it "might" be…, it is "likely" that…, and terms such as very, quite, seldom, frequently are used too. Experts seldom reach the same conclusions. We can use aggregations of conclusions to reach a single consensus, or **fuzzy output** can often be acceptable, depending on the application.

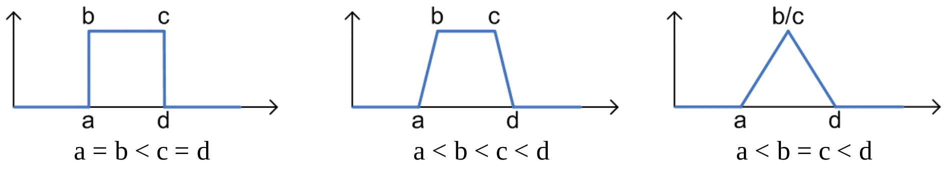
The "universe of discourse" is the range of all possible values applicable to a variable.

A fuzzy variable becomes a **linguistic variable** when we modify it with descriptive words, such as somewhat fast, very high, real slow, etc. Speed is a fuzzy variable. Accelerator setting is a fuzzy variable. Examples of linguistic variables are: *somewhat fast* speed, *very high* speed, *dead slow* speed, *excessively high* accelerator setting, accelerator setting *about right*, etc.

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.

Bayesian rules concern revising probabilities based on new data.

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.

3. Neural networks and Learning algorithms

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.

4. Neuro-fuzzy systems

A neuro-fuzzy system is one in which fuzzy logic is applied to the trigger algorithm which dictates when a neuron triggers.

5. Genetic and Evolutionary computing

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, including themselves), **cross-over** (algorithmic changes) and **mutation** (random changes). Given a mechanism for determining the *fitness* of the population (this is key), 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 fixed probability (e.g. 0.8) at (possibly) random point(s) 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. An effect of elitism is that the number of offspring is reduced in subsequent generations.

There are many ways to apply **crossovers**, one-point, multi-point, cut-and-splice. Depending on the nature of the chromosome, e.g. ordered lists (travelling salesman), crossovers may not always be possible.

Mutations are used to maintain diversity within a population. It helps to allow the algorithm to avoid local minima by preventing the population from becoming too similar to each other.

Tournament selection is one of many methods of selection, and involves a tournament or competition between a small section of the population with the winner (the with the best fitness) being chosen for crossover.

K-means using GA: 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.

K-means using EA: for clustering entities with v features, randomly generate chromosomes of $K \cdot v$ real numbers, which represents the centroids c_1, c_2, …, 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.

6. Integration of Biological Data Sources

Bioinformatic data exhibit several characteristics: it is diverse (gene expressions, disease characteristics, 3D protein structures), heterogenic (data represented in different ways with varying or overlapping structures, naming conventions and semantics, leading to inconsistencies duplication and redundancy) and autonomous (originators are free to modify data and structure without any "public" notification, which creates instability and unpredictability), and is accessed via various querying interfaces (requiring biologists to become technologists, and restricting access to potentially useful information).

There are three main architectural approaches to integration,
(i) **warehouse** – data is materialised into a single warehouse. ✓ good performance since data not distributed, ✓ high availability since data held locally, ✓ good scope for cleansing and optimising data; ✗ difficult to maintain & keep updated.
(ii) **mediator** – maps are produced linking the global schema to local data sources, with data remaining in their original sources. ✓ no need to store all the data, ✓ no need to 'ETL' data when updated; ✗ poor query optimisation & performance opportunities.
(iii) **navigational** – data also remains in original sources, but links are used to enable queries to navigate to the data. ✓ natural to use, ✓ good performance; ✗ users not isolated from data sources, ✗ links may break.

GAV *vs.* LAV, applicable to (i) & (ii) above.
With GAV, tables are implemented as views over local schema; with LAV local tables are expressed as views over global constructs & tables.
GAV is simpler to implement but changes to data sources necessitate changes to global schema. LAV makes it simpler to add or remove data sources, but more complex to implement queries.

Standardisation is needed for integrating biological data, including (i) identification labels, (ii) ontologies (so we are all talking the same language), (iii) data models and (iv) data interchange formats, e.g. MAGE-ML (when GE = gene expression).

Ontology-based mechanisms. These may differ in the constructs supported, but typically include
(i) concepts, both primitive (e.g. A is B, but that's all) and defined (e.g. A is B and B is – or is not – A).
(ii) relations, both taxonomic (e.g. organise concepts into hierarchical concepts, maybe a bit vague) and associative (more specific),
(iii) instances of concepts, and
(iv) axioms or constraints.
Ontologies help by providing a common language and common terms to allow for natural language queries to be expressed against the data.

Schema and data transformation mechanisms:
Birkbeck uses the **AutoMed** toolkit to allow for a partially-materialised warehouse by supporting mappings between local data sources and global warehouse schemas. AutoMed does not require a wholly GAV or LAV approach, but permits a **BAV** (both-as-view) approach by supporting bi-directional mappings.

7. Grid Computing

The term first arose in the mid-1990s and is also known as *utility computing* or *"in demand" computing.*
The world is full of inter-connected computing resources, but their distribution, heterogeneity and autonomy makes it hard for such resources to be shared. Grid computing attempts to right this wrong.

Grid can be seen as the latest and most complete evolution of distributed computing, peer-to-peer computing the Web and virtualisation technologies, creating *virtual organisations*.

Pros (weak):
Become an early adopter, build an image as an innovator, potential for public funding, form partnerships with academic organisations, influence the development of grid standards, take advantage of increasing globalisation (none of these are very convincing).
Cons:
Difficult & costly to refactor legacy systems, staff do not possess 'grid skills', little literature available, standards still immature, difficult to estimate effort required, concerns about security, legal and accounting aspects.

3 problems to be overcome: distributed, heterogeneous and autonomous.
2 solutions: networks, common standards.
Issues include security, co-ordination and accountability.
Motivations include better use of resources, cost benefit, efficiency benefits.

The dynamic collections of individuals, institutions and resources collectively form a *virtual organisation (VO)*.

There are four types of grid: departmental, enterprise (share across departments), partner (share across agreed companies, beyond firewalls) & open (anyone can provide and/or use resources).

The layered approach, analogous to the OSI 7-layer model, has been disbanded as being too cumbersome and is now replaced by a looser architecture involving just applications, a service pool (middleware) and resources.

Certain standards are required such as service discovery and quality of service (QoS).
OGSA refers to *resource handle* (RH) and *resource address* (RA) instead of using existing URNs and URIs.
Comparing grid architecture with web services, grid resources tend to be less static and particular resources cannot be guaranteed to be as available as a web service. A grid can be composed of web services; the two are not incompatible.

Acronyms:
OGSA (Open Grid Services Architecture),
OGSA-DAI (OGSA Data Access and Integration),
OGSA-DQP (OGSA Distributed Query Processing).