A Review of Knowledge Technologies

COMP90049
Knowledge Technologies

Sarah Erfani and Karin Verspoor, CIS

Semester 2, 2018

THE UNIVERSITY OF MELBOURNE

Exam Instructions (subject to change!)

A Review of Knowledge Technologies

COMP90049 Knowledge Technologies

Exam structure

Basics

Data representation

Information Retrieval

Data Mining vs. Machine Learning

Supervised Learning

Unsupervised Learning

- Date: 9 Nov 2018
- Time: 8:30 am
- Reading Time allowed: 15 minutes
- Writing Time allowed: 2 hours
- Instructions to candidates:
  - This paper counts for 50% of your final grade.
  - Answer all questions.
  - Note that questions are not of equal value.
  - Closed book exam.
  - No external materials may be used for this exam.
  - Please use your script book for the long answer question, clearly marking where the response starts. Any pages which are not labelled as forming part of the response to that question number will not be considered during marking.
  - Write your name and student ID on top of the exam paper as well as the script book.

Questions and values

There are 85 marks in total, or 1 mark per 1.4 minutes.
The value of the question reflects approximately how long it should take you to answer it.
There will be groups of questions corresponding to one major topic/concept from the subject.

Scope:
All lecture material is examinable, except the guest lecture.
There will be questions that address topics from before the mid-semester test.
However, the emphasis will be on the second part of the semester.

Types of questions

There are a mix of question types on the exam.

- Conceptual: A question which tests or requires you to define or explain a concept, term, or algorithm introduced in the subject.
- Problem solving: A question which asks you to use a specific algorithm or formula to solve a problem on some data.
- Application: A question which asks you to demonstrate that you have gained a high-level understanding of the methods and algorithms covered in this subject, and can apply that understanding.

Conceptual Question: Example

1. Classify the following “attributes” as binary, discrete or continuous:
   1. number of patients in a hospital
   2. human gender
   3. ability to pass light, in terms of the following values: opaque, translucent and transparent

Conceptual Question: Example [Answer]

1. Classify the following "attributes" as binary, discrete or continuous:
   1. number of patients in a hospital **[CONT]**
   2. human gender **[BIN]**
   3. ability to pass light, in terms of the following values: opaque, translucent and transparent **[DISC]**

Conceptual Question: Example 2

2. Outline the difference between “deterministic” and “probabilistic” clustering.

Conceptual Question: Example 2 [Answer]

2. Outline the difference between “deterministic” and “probabilistic” clustering.

   Deterministic clustering assigns each instance to a unique cluster; probabilistic clustering generates a probability distribution for each instance across all clusters.

Problem Solving: Example

Given the following set of training and test document instances:

	Features				Class
	hand	*finger*	*palm*	*pilot*	
Training Instances					
A:	0	3	0	0	anatomy
B:	2	2	1	0	anatomy
C:	0	1	2	2	pda
D:	1	0	2	2	pda
Test Instance					
X:	2	1	1	1	

1. Calculate the "similarity" of test instance X with each training instance (A, B, C and D) using "cosine similarity" (without weighting).
2. Classify test instance X using the document similarities from (a) and the "3-NN method".

Problem Solving: Example [Answer]

	Features				
	hand	*finger*	*palm*	*pilot*	CLASS
TRAINING INSTANCES					
A:	0	3	0	0	`anatomy`
B:	2	2	1	0	`anatomy`
C:	0	1	2	2	`pda`
D:	1	0	2	2	`pda`
TEST INSTANCE					
X:	2	1	1	1	

Cosine similarity: $sim(A, B) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$

1. $sim(X, A) = \frac{2\times0+1\times3+1\times0+1\times0}{\sqrt{7}\times\sqrt{9}} = \frac{3}{3\sqrt{7}}$
   $sim(X, B) = \frac{2\times2+1\times2+1\times1+1\times0}{\sqrt{7}\times\sqrt{9}} = \frac{7}{3\sqrt{7}}$
   $sim(X, C) = \frac{2\times0+1\times1+1\times2+1\times2}{\sqrt{7}\times\sqrt{9}} = \frac{5}{3\sqrt{7}}$
   $sim(X, D) = \frac{2\times1+1\times0+1\times2+1\times2}{\sqrt{7}\times\sqrt{9}} = \frac{8}{3\sqrt{7}}$
2. Closest-matching docs = D, B, C
   Assuming each document has an equal vote, this means there are two votes for `pda` and one for `anatomy`, meaning that the final classification is the majority class of `pda`

A comment on Mathematical Concepts

We expect you to be able to do:

- addition, subtraction, multiplication, division
- reducing and ordering of fractions
- logarithms and square roots (where they have integer solutions)
- determine probabilities
- recognise and apply probabilistic formulas (e.g., for Naïve Bayes, Entropy, Language models in IR)
- calculate similarities (e.g., Cosine, Dice, Pearson)

A comment on Mathematical Concepts

We do expect that you can:

- Remember simple, key formulas (certainly, TF*IDF, Precision, Recall, F-score).
- Read and understand more complex formulas that have been presented for core concepts, provided “bare”.

We do not expect you to be able to do:

- matrix multiplication and linear algebra (e.g., for SVMs)
- mult-variable calculus (e.g., for Neural Networks)

Review of Knowledge Technologies

Fundamental Concepts

- Structured vs. Semi-structured vs. Unstructured data
- Supervised vs. Unsupervised learning
  (What are some examples of algorithms for each?)
- String processing: Regular Expressions

Approximate String Matching

String distance metrics:

- Edit distance
  The simplest edit distance is the number of insertions, deletions, and substitutions needed to turn one string into another.
- N-gram distance
  Let $G_n(s)$ be the substrings of length n of s.
  The n-gram distance of t and s is

$$|G_n(s)| + |G_n(t)| - 2 \times |G_n(s) \cap G_n(t)|$$

Edit distances

```
Global:
    lq = strlen(q); lt = strlen(t);
    for( i=0 ; i<=lq ; i++ ) F[0][i] = i;
    for( j=0 ; j<=lt ; j++ ) F[j][0] = j;

    for( i=1 ; i<=lq ; i++ )
        for( j=1 ; j<=lt ; j++ )
            F[i][j] = min3(
                F[i-1][j] + 1,
                F[i][j-1] + 1,
                F[i-1][j-1] + equal(q[i-1], t[j-1]));


Local:
    lq = strlen(q); lt = strlen(t);
    for( i=0 ; i<=lq ; i++ ) F[0][i] = 0;
    for( j=0 ; j<=lt ; j++ ) F[j][0] = 0;

    for( i=1 ; i<=lq ; i++ )
        for( j=1 ; j<=lt ; j++ )
            F[i][j] = max4(
                0,
                F[i-1][j] - 1,
                F[i][j-1] - 1,
                F[i-1][j-1] + equal(q[i-1], t[j-1]));
```

Vector representation

A *feature vector* is an n-dimensional vector of *features* that represent some object.

A *feature* is any distinct aspect, quality, or characteristic of that object.

- Features may be symbolic/categorical/discrete (e.g. colour, gender)
- Features may be numeric/continuous (e.g., height, age)

A vector locates an object (document, person, . . .) as a point in *n*-space. The angle of the vector in that space is determined by the relative weight of each term.

Suppose there are n distinct indexed terms in a collection. Then each document d can be thought of as a vector

$$\langle w_{d,1}, w_{d,2}, \ldots, w_{d,t}, \ldots, w_{d,n} \rangle$$

where $w_{d,t}$ is a weight describing the importance of term t in d.

Term weighting

- The basic elements used in term weighting are:
  - $f_{d,t}$, the frequency of term t in document d (TF)
  - f_d, the number of terms contained in document d
  - f_{ave}, the average number of terms contained in a document
  - f_t, the number of documents containing term t (DF)
  - F_t, the total number of occurrences of t across all documents
  - N, the number of documents in the collection
  - n, the number of indexed terms in the collection

These statistics are sufficient for computation of the similarity functions underlying highly effective search engines.

Information Retrieval

Definition:

Information retrieval (IR) is “the subfield of computer science that deals with storage and retrieval of documents” (Frakes & Baeza-Yates, 1992).

- There is an emphasis on the *user* with an *information need*. IR systems can be characterized as mechanisms for finding documents that are of value to an individual.
- The meaning or *content* of a document is of more interest than the specific words used to express the meaning.
- Searching: To resolve an information need using a search engine, a user chooses words and phrases that are intended to match appropriate documents, then use these words and phrases to construct a query.
- What makes a good *answer* to a query? *Relevance*
  A document is relevant (that is, on the right topic) if it contains knowledge that helps the user to resolve the information need.
- *Ranked Retrieval:* The more similar or likely a page is, relative to the other documents in the collection, the higher its *rank*.

Elements of a web search engine

Web search involves four main technological components.

- **Crawling**: the data to be searched needs to be gathered from the web.
- **Parsing**: the data then needs to be translated into a canonical form.
- **Querying**: the data structures must be processed in response to queries.
- **Indexing**: data structures must be built to allow search to take place efficiently.

Inverted File Index

Search structure

For each distinct word t, the search structure contains:

- A pointer to the start of the corresponding inverted list.
- A count f_t of the documents containing t.

That is, the search structure contains the *vocabulary*.

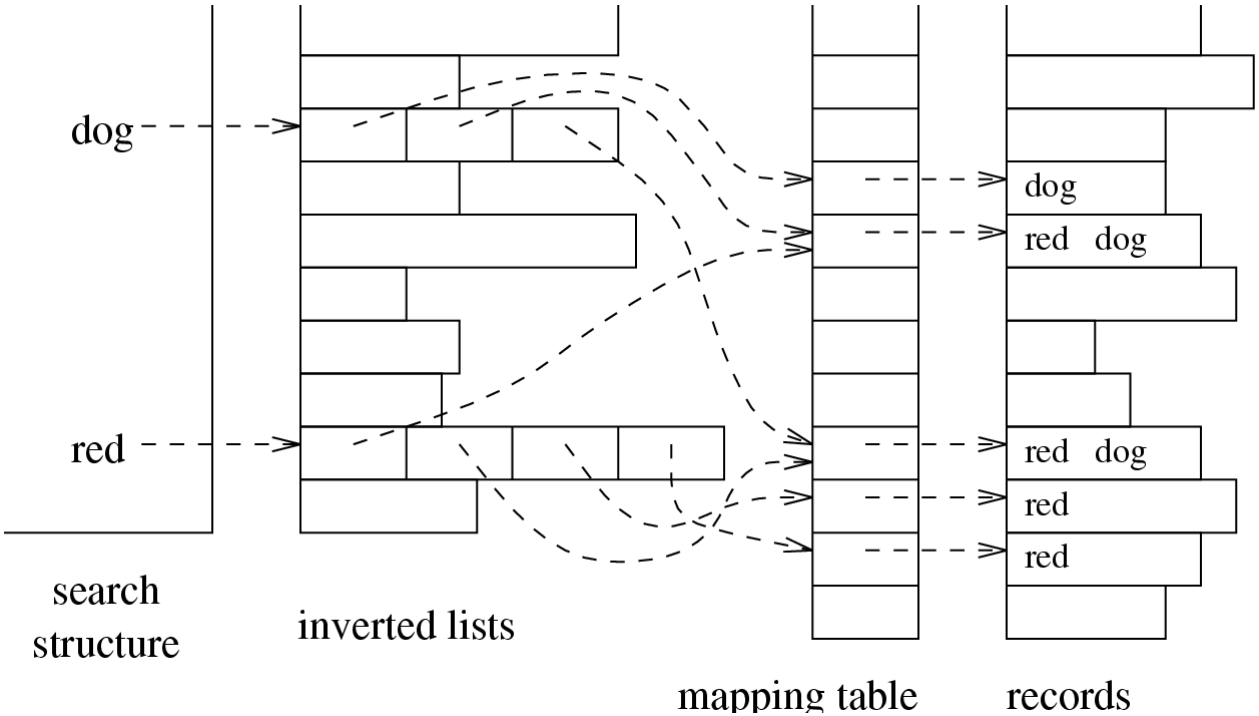
Inverted lists

For each distinct word t, the inverted list contains:

- The identifiers d of documents containing t, as ordinal numbers.
- The associated frequency $f_{d,t}$ of t in d. (We could instead store $w_{d,t}$ or $w_{d,t}/W_d$.)

Inverted File Index

Together with an array of W_d values (stored separately), the search structure and inverted file provide all the information required for Boolean and ranked query evaluation.



Querying, using an accumulator.

What is Data Mining?

Extracting

- implicit,
- *previously unknown*,
- potentially useful

information from data

- Needed: programs that detect patterns and regularities in the data
- Strong patterns $\rightarrow$ good predictions
  - Problem 1: most patterns are not interesting
  - Problem 2: patterns may be inexact (or spurious)
  - Problem 3: data may be garbled or missing

What is Machine Learning?

Algorithms for acquiring structural descriptions from examples

- Structural descriptions represent patterns explicitly
- Can be used to predict outcome in new situation
- Can be used to understand and *explain* how prediction is derived (may be even more important)

Methods originate from artificial intelligence, statistics, and research on databases

Supervised Learning

Learning in the context where we have training data labelled with a class value for each instance, for a given classification task.

- *Teach* the computer how to do something *(by example)*, then let it use its new-found knowledge to do it
- Labeled data: for given inputs, provide the expected output value ("the answer")
- Infer a function mapping from inputs to outputs

How to do (Supervised) Machine Learning

0. Get hired!
1. Pick a feature representation (Feature Engineering)
2. Compile data
3. Pick a (suitable) model
4. Train the model
5. Classify development data, evaluate results
6. Probably: *go to (1)*

Feature selection: why?

Better models!

- Better performance according to some evaluation metric

Side-goal:

- Tell us interesting things about the problem

Side-goal:

- Fewer features $\rightarrow$ smaller models $\rightarrow$ faster answer
  - More accurate answer $>>$ faster answer

Feature selection: how?

- *"Wrapper" methods:* Choose subset of attributes that give best performance on the development data
- *"Ablation" approach:* Start with all attributes. Remove one attribute, train and evaluate model.
- *"Embedded" methods:* Performed as part of the algorithm, e.g. linear classifiers, decision trees.
- *"Filtering" approach:* Evaluate goodness of feature. Look for features (reverse) correlated with class.

- *Mutual information*:

$$MI(T;C) = \sum_{t \in \{0,1\}} \sum_{c} P(t,c) \log_2 \frac{P(t,c)}{P(t)P(c)}$$

- χ^2 ("kai-square"):

$$\chi^2 = \frac{N(WZ - XY)^2}{(W+X)(W+Y)(X+Z)(Y+Z)}$$

Supervised Learning Algorithms

Algorithms

- *k*-Nearest Neighbour classification
- Naive Bayes classification
- Decision Trees & Random Forests
- Support Vector Machines
- Neural Networks

Nearest Neighbour classification

Given class assignments for existing data points, classify a new point, according to the class membership of the K closest data points.

[1-NN]: Classify the test input according to the class of the closest training instance.

[K-NN]: Classify the test input according to the majority class of the K nearest training instances.

[weighted K-NN]: Classify the test input according to the weighted accumulative class of the K nearest training instances, where weights are based on similarity of the input to each of the K neighbours.

[offset-weighted K-NN]: Classify the test input according to the weighted accumulative class of the K nearest training instances, where weights are based on similarity of the input to each of the K neighbours, factoring in an offset to indicate the prior expectation of a test input being classified as being a member of that class.

Naive Bayes

- Task: classify an instance $D = \langle x_1, x_2, ..., x_n \rangle$ according to one of the classes $c_j \in C$

$$\begin{aligned} c &= \text{argmax}_{c_j \in C} P(c_j | x_1, x_2, ..., x_n) \\ &= \text{argmax}_{c_j \in C} \frac{P(x_1, x_2, ..., x_n | c_j) P(c_j)}{P(x_1, x_2, ..., x_n)} \\ &= \text{argmax}_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \end{aligned}$$

posteriori $P(c_j | x_1, x_2, ..., x_n) = \frac{likelihood * prior}{evidence}$

- Predicts D belongs to c_i iff the probability $P(c_i | D)$ is the highest among all the $P(c_k | D)$ for all the K classes
- Since $P(x_1, x_2, ..., x_n)$ is constant for all classes, only $P(x_1, x_2, ..., x_n | c_j) P(c_j)$ needs to be maximised.

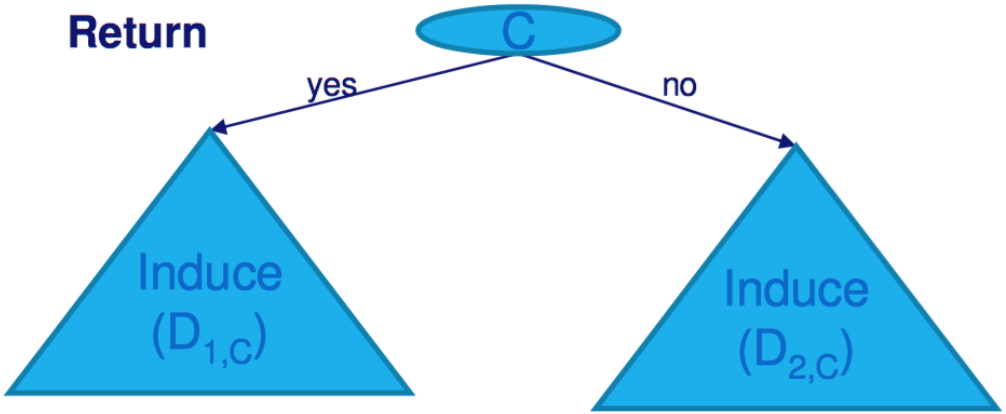
Applying the conditional independence assumption:

$$\begin{aligned} c &= \text{argmax}_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \\ &= \text{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \end{aligned}$$

Decision Trees: Hunt's algorithm

Input: Dataset D
Output: Decision tree t

Induce(D):
  If all tuples t in D have label + then
    return +
  If all tuples t in D have label - then
    return -
  For all split criteria C:
    $D_{1,C}$ = { t in D | t satisfies C}
    $D_{2,C}$ = D - D_1
    Measure Quality(D_1 ,D_2)
  Let C be the best split
  Return



Node impurity

- Classification Error at a node t

$$\text{Error}(t) = 1 - \max_i P(i|t)$$

($P(i|t)$: proportion of points in the i-th class)

- Entropy (homogeneity) at node t

$$H(t) = -\sum_{j=1}^{n} P(j|t) \log_2 P(j|t)$$

where $P(j|t)$ is the relative frequency of class j at node t

- Gini index

$$\text{GINI}(t) = 1 - \sum_{j=1} [P(j|t)]^2$$

where $P(j|t)$ is the relative frequency of class j at node t

Attribute selection: Information Gain

We determine which attribute R_A (with values $v_1, \ldots v_k$) best partitions the instances at a given root node R according to *information gain* based on the Impurity function I

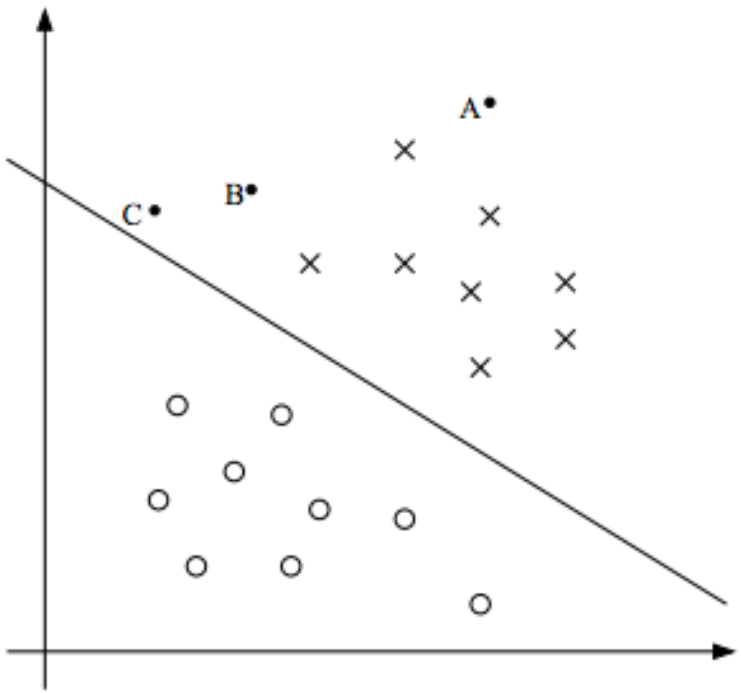
$$IG(A|R) = I(R) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

or *gain ratio*, which reduces the bias for information gain towards highly-branching attributes by normalising relative to the *split info*
Split info = entropy of a given split (evenness of split)

$$\begin{aligned} GR(A|R) &= \frac{IG(A|R)}{\text{Split_Info}(A|R)} = \frac{IG(A|R)}{H(A)} \\ &= \frac{I(R) - \sum_{j=1}^{k} P(v_j) I(v_j)}{-\sum_{j=1}^{k} P(v_j) \log_2 P(v_j)} \end{aligned}$$

Support Vector Machines

- A SVM model aims to find a hyperplane that separates two classes. The data points on the hyperplane are called *support vectors*.
- For a given training set, we would like to find a decision boundary that allows us to make all correct and confident (large margin) predictions on the training examples.
- SVMs apply to linearly separable data.



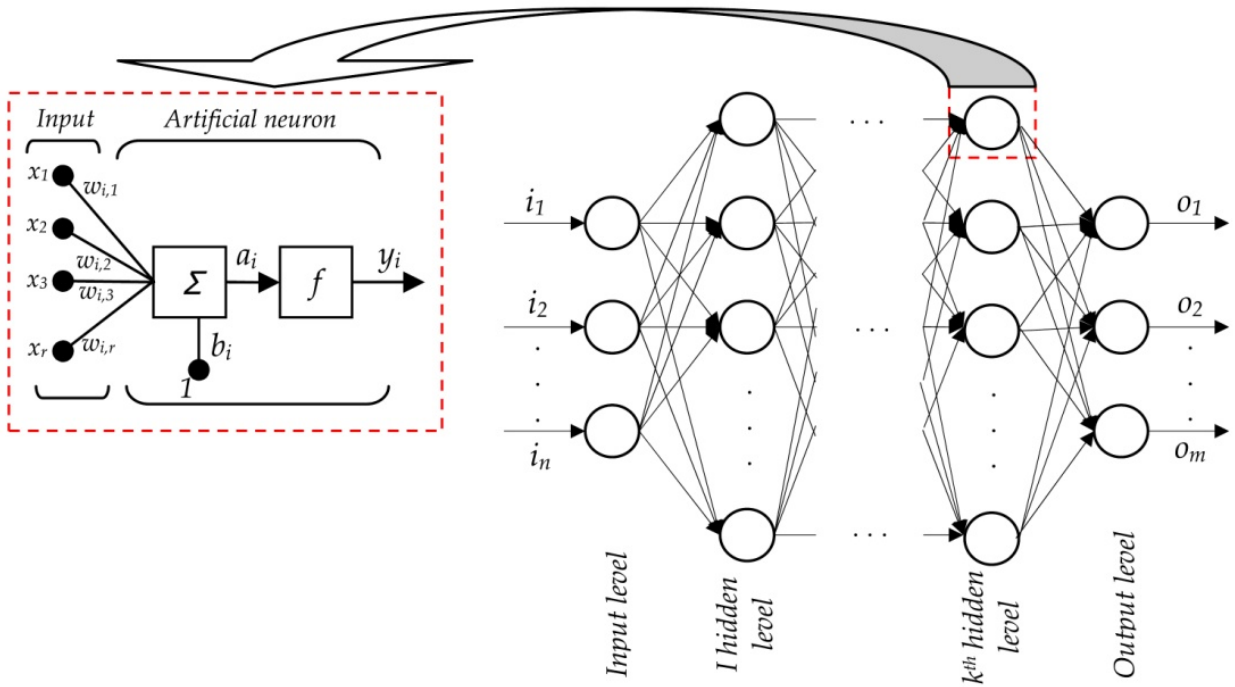
Concepts:

How do we handle non-linearly separable data?
What is the significance of a “soft margin”?
How do we extend a SVMs to a multi-class classification task?

Neural Networks

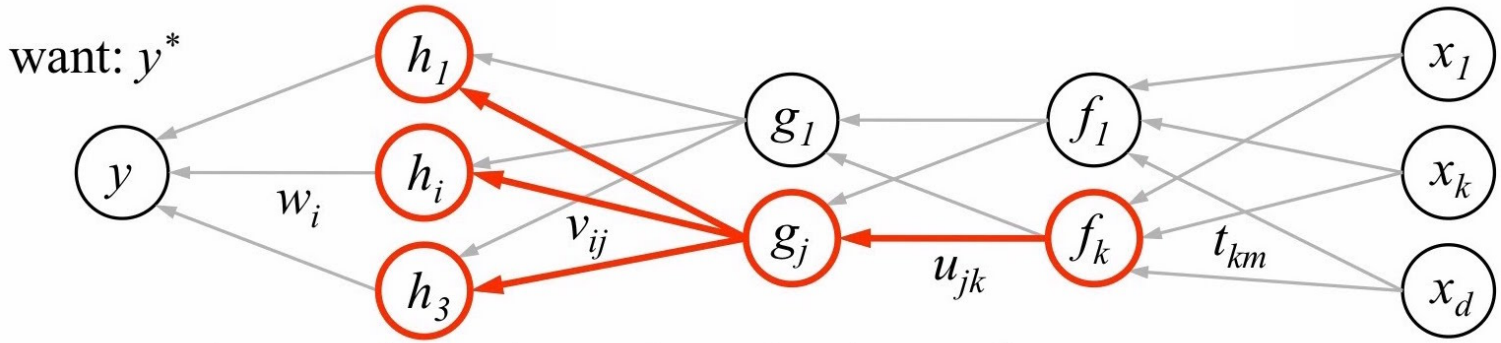
A neural network is a supervised machine learning method, usually represented graphically (with nodes and branches).

- input nodes and output nodes
- (usually) intermediate nodes
- edge weights determine how the incoming values contribute to node weight
- perform a weighted sum of the inputs, and then apply an activation function, to determine the activation that feeds forward



Backpropagation

Weights in a NN are learned through the *back-propagation algorithm* which propagates error back through the network, and adjust weights.



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Unsupervised Learning

Learning in the context where we *don't* have (or don't use) training data labelled with a class value for each instance.

- Let the computer *learn how to do something*
- Determine structure and patterns in data
- Unlabeled data: Don't give the computer "the answer"

Algorithms

- Clustering
- Association Rule Mining

Clustering algorithms

Finding groups of items that are *similar*.

- *k*-means clustering
- hierarchical clustering
  - agglomerative clustering
  - divisive clustering

k-means Clustering

Given k, the *k*-means algorithm is implemented in four steps:

1. Select *k* points to act as seed cluster centroids
2. **repeat**
3. Assign each instance to the cluster with the nearest centroid
4. Recompute the centroid of each cluster
5. **until** the centroids don't change

- Exclusive, deterministic, partitioning, batch clustering method

Agglomerative Clustering

1. Compute the proximity matrix, if necessary.
2. **repeat**
3. Merge the closest two clusters
4. Update the proximity matrix to reflect the proximity between the new cluster and the original clusters
5. **until** Only one cluster remains

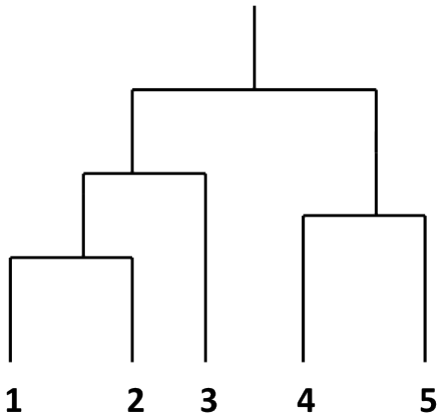
Updating the proximity matrix:

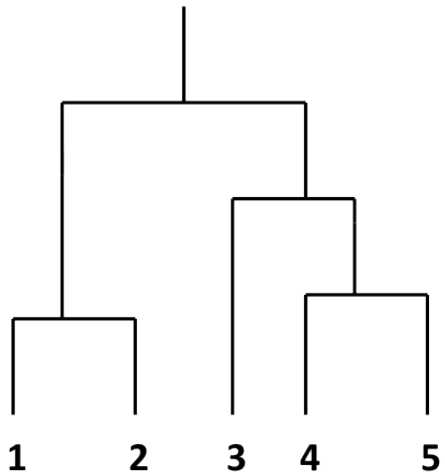
- Single Link: *Minimum* distance between any two points in the two clusters. (most similar members)
- Complete Link: *Maximum* distance between any two points in the two clusters. (most dissimilar members)
- Group Average: *Average* distance between all points (pairwise).

- Exclusive, deterministic, hierarchical, batch clustering method

Agglomerative Clustering Example

	1	2	3	4	5
1	1.00	0.90	0.10	0.65	0.20
2	0.90	1.00	0.70	0.60	0.50
3	0.10	0.70	1.00	0.40	0.30
4	0.65	0.60	0.40	1.00	0.80
5	0.20	0.50	0.30	0.80	1.00





Single link (min pairwise dist)

Complete link (max pairwise dist)

Association rule mining via the Apriori algorithm

An association rule is an implication $A \rightarrow B$, where A and B are disjoint *itemsets*.

- The *support count* $\sigma(X)$ of an itemset X is defined as the number of transactions that contain X, i.e.

$$\sigma(X) = \left|\{t_i | X \subseteq t_i, t_i \in T\}\right|$$

- We conventionally evaluate the "interestingness" of a given association rule via:
  - **support**$(A \rightarrow B) = \frac{\sigma(A \cup B)}{\sigma(*)}$ $(\sim P(A, B))$
    the proportion of transactions in the data set which contain the itemset
  - **confidence**$(A \rightarrow B) = \frac{\sigma(A \cup B)}{\sigma(A)}$ $(\sim P(B|A))$
    the proportion of the transactions for which items in B appear in transactions containing A

Mine association rules *using the Apriori Algorithm* which satisfy the following constraints:

- $support \geq M$
- $confidence \geq N$
- $|A| \geq 1$ and $|B| \geq 1$

Content-based recommendation

- Recommendations are based on information on the **content** of items
- Uses a machine learning algorithm to induce a profile of the users preferences from examples based on a featural description of content
  - Learn user preferences based on content interacted with/purchase history/etc. $\rightarrow$ *User Profile*
  - Locate/recommend items that are "similar" to the user preferences $\rightarrow$ Similarity of *User Profile* to *Item Profile*

Collaborative Filtering

Predict user preferences (*filtering*) by collecting preference information from many users (*collaborative*)

- User-based models: Similar users have similar ratings on the same item. (Calculate similarity of users based on similarity of item ratings.)
- Item-based models: Similar items are rated in a similar way by the same user. (Calculate similarity of items based on similarity of user ratings.)

Pearson correlation: $r(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\Sigma(x_i-\bar{x})^2\Sigma(y_i-\bar{y})^2}}$

Adjusted cosine similarity: adjusts for user differences in rating scale

$sim(i,j) = \frac{\Sigma(r_{u,i}-\bar{r_u})(r_{u,j}-\bar{r_u})}{\sqrt{\Sigma(r_{u,i}-\bar{r_u})^2\Sigma(r_{u,j}-\bar{r_u})^2}}$

Would you now call yourself a data scientist?

You know how to:

- Massage, pre-process, transform, and efficiently search data
- Represent complex data in straightforward data structures and with probabilities
- Address problems where the user's *information need* is primary
- Build models from data
- Set up experiments to test models
- Interpret and analyse the results of those experiments
- . . . and more! . . .

Thank you for your attention!

Good luck on the exam!!