Exam Revision - 3152

Week 12 – Exam revision

LECTURE 1

- Given data, draw/describe a graphic to represent the data
    - Multivariate graphic – visualisation zoo
        - Networks
        - Size, shape, colour

QUESTION 1 – SAMPLE EXAM

- Rbind, cbind, sorting

LECTURE 2 – VISUALISATION OF DATA

- Ggplot2
- Visualisation zoo
- Sample exam – question 9

LECTURE 3 – DATA MANIPULATION in R

- Making tables and summaries
- Working with factors
    - Apply functions by groups
- Transforming data – analysis + reformatting output
    - Aggregate, cor, by, as.table, as.data.frame, colnames, merge, cbind, rbind, max, which.max, do.call
- Tidying output
    - Max.type = by(iris, iris[5[, function(df) df[which.max(df[,3]),])
    - Do.call(rbind, max.type)
- QUESTION 2 – SAMPLE EXAM
    - Correlation, don't need to put comma when defining col
        - Df[1] instead of df[,1]
        - Only for correlation as correlation works on columns, no rows

LECTURE 4 – GUEST LECTURE

- Methodologies
    - KDD, SEMMA, CRISP-DM
    - Business understanding, data understanding, data prep, modelling, evaluation, deployment
- Data preparation and pre-processing
- Know problems that may exist in data – dirty data
    - Values incorrect, inconsistent i.e. age is 20, when born in 2015
    - Break business rules i.e. Age can be -3
- See taxonomy of dirty data
- Dimension reduction
    - When two sets of variables are highly correlated, can most likely get rid of one of them

LECTURE 5 – LINEAR REGRESSION

- Understand the output
    - Median – residuals should be normally distributed, median should be close to 0, if not skewing could be going on
    - Estimate std (coefficient) – i.e. answer
    - PR(>|t|) – significance of the variables, look for the stars
    - T value – looking for big t-values
    - Multiple r-squared – the % of the model of the variability in y; explains by the data
    - P-value – overall significance of the regression, want the value to be close to 0
        - Overall, if it's close to 0, then some variables in the regression mean something

- SAMPLE QUESTION 3
    - What's the coefficient of colour 1
        - One of the factors has to be 0; colour 1 is 0
    - 3d)
        - highest values are the ones where the smallest amounts are subtracted
            - colour 2, 3 and 4
            - reliability 95%
        - smallest
            - the others
            - reliability = 99.99%
- Two main types of data for regression
    - Numerical
    - Factors
        - Use of levels

LECTURE 6 – NETWORKS

- Can't calculate iganvector by hand
    - Nodes (vertices) and edges (arcs), directed – undirected, weighted, unweighted
    - Walk, path, cycle, geodesic (shortest path), length, connected
    - Loop, complete, subgraph, clique, simple
- Network statistics
    - Diameter, average path length, degree distribution
- Vertex charactistics
    - Degree – how many lines go in or out (connections)
    - Centrality: betweeness (how many shortest paths pass through node), closeness
- SAMPLE QUESTION 4
    - Degree vertex 4 = 3
    - Closeness, distance to everyone else = 1/9
    - Betweenness,

LECTURE 7 – DECISION TREES

- Nodes, branches, tree nodes, entropy, model accuracy
- Know the entropy calculation
    - Log to base 2
    - Information gain
- Shortcuts for building decision trees
    - Check attributes to see if one group has either all yes or all no
        - This is because it reduces entropy/increase information gain
    - Worst class is 50/50 split; just a guess
- Metric for performance evaluation = accuracy
- Actual classes – from the actual world, what are the actual classes
- Why use decision tree
    - Classify unclassified data

LECTURE 8 – IMPROVING THE CLASSIFICATION MODEL

- NB, ROC curves, LIFT, Cross-validation
- Bayesian classifier
- ROC curves
    - TPR = TP/TP+FN; Y-axix
    - FPR = ; X-axis
    - If confidence levels aren't given, use the ones in the data
- AUC
    - Just estimate in the exam
- Lift
    - If you can target model to look at best bit, top x% most confident of, whats the improvement in the predicitabliluty
    - Lift factor = success rate with model/success rate without model
- No ensemble methods in exams

LECTURE 9 – CLUSTERING

- K-means
  - K is whatever we want; limitation
  - K is the number of centres
  - steps
- Hierarchical
  - Dendrogram
    - Can cut at any level to get clusters we want, to get different groups

LECTURE 10 – TEXT PROCESSING

- Bag of words
- Term-document matrix
- Need to know
  - How to process text for analysis
    - Bag of words
      - Pull words out, and put into a bag
      - Order
  - Steps of extracting structure
    - Tokenize
    - Convert case
    - Removing stop words
    - Stem
    - Lemmatize
    - Create n-grams

Week 1 – Data Analytics Overview

DATA STRUCTURES

- Array: data of the same type
  - Vector: 1D, Matrix: 2d, Array: 3+ Dimensions
    - Vector is considered as a smaller array
    - X <- c(1,2): X would be considered as a vector
- Data frame
  - Row x Column data format – each column is a vector
- List
  - An ordered collection of (possibly different) types

R-CODE

- Data() # lists the built in data sets

DESCRIPTIVE STATISTICS

- To compute column means – colMeans(DF, na.rm=TRUE)
- To compute means by group/factor – tapply(DF, factor, mean)

BIVARIATE DATA

- Analyse a relationship between two variables – cor(x, y)

Class

- X <- c (1,3)
  - Class (x) – x would be a numeric class
- X <- c (1,3 + “Monkey”)
  - Class (x) – x would be character. A class can only have one class (cant be numeric and character)

Week 2 – Data Visualisation

T-TEST

- T.test(WorkerA)
    - Perform a t.test() to generate confidence level
    - Performs one and two sample t-tests on vectors

VISUALISATION ZOO

- The goal of visualisation is to aid our understanding of data by leveraging the human visual system's highly tuned ability to see patterns, spot trends, and identify outliers
- The challenge is to create effective and engaging visualisations that are appropriate to the data
- Numerical data
    - Scatter plot or bar chart
    - Leads to most accurate decoding numerical data
- All visualisations will essentially use the same DNA (visual attributes)
    - Position
    - Size
    - Shape
    - Colour

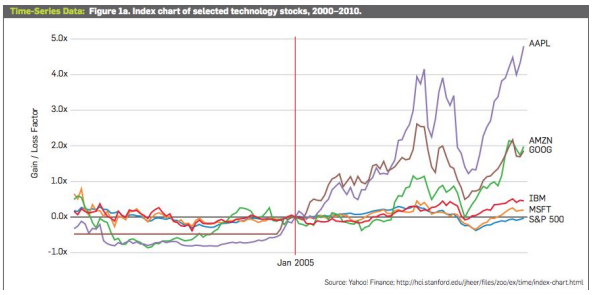
<u>GRAPHICS</u>

TIME SERIES DATA

- Comparing data over time
- Decomposition
    - Breaks down the time series analysis into specific factors

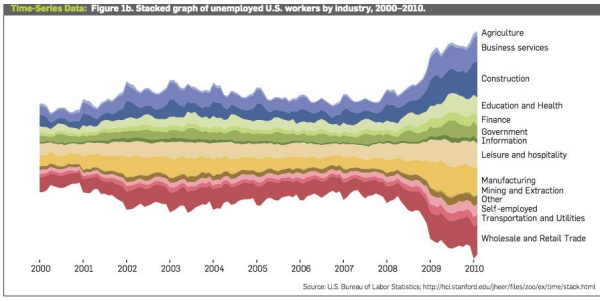
INDEX CHARTS

- An interactive line chart showing percentage changes for a collection of time-series data based on a selected index point



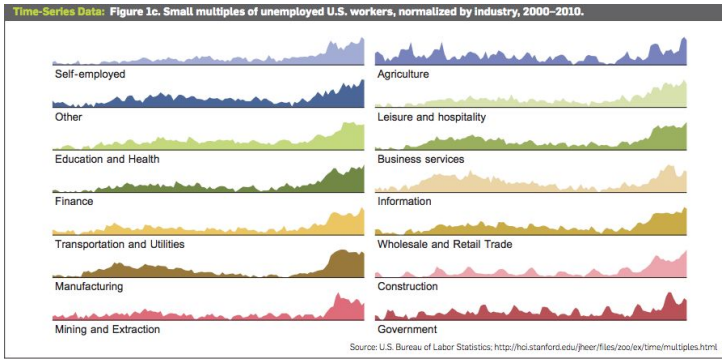
STACKED GRAPHS

- Other time-series graphs may be better visualised in aggregate
- Seen by stacking area charts on top of each other
- Does not support negative numbers, and meaningless for numbers that should not be summed (I.e temp)



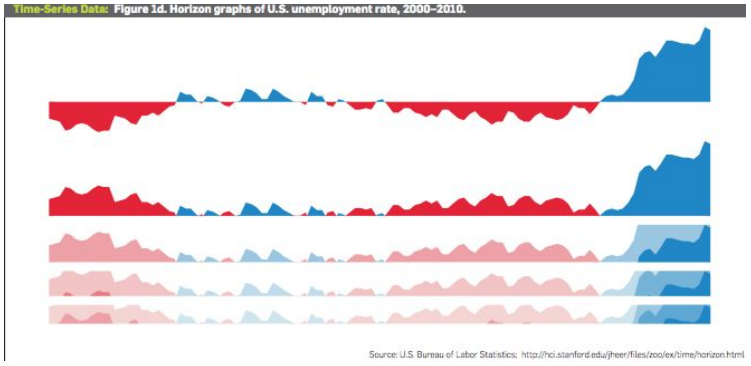
SMALL MULTIPLES

- Chart that contains multiple time series plots separated and not in the same graph
- Easier to compare
- Can be constructed with just about any type of plot (bar plot, pie chart etc.)



HORIZON GRAPHS

- Like small multiples, however the graphs are stacked on top of each other, without losing resolution
- The first graph is full size, and the following charts are zoomed in little by little, keeping the same resolution



STATISTICAL DISTRIBUTIONS

- Plots that are designed to reveal how a set of numbers is distributed to help better understand the statistical properties of the data
- Histograms are the most popular

STEM-AND-LEAF PLOTS

- Typically binds numbers according to the first significant digit, and then stacks the values within each bin by the second significant digit
- Uses the data itself to paint a frequency distribution, allowing one to assess both the overall distribution and the contents of each bin

Statistical Distributions: **Figure 2a. Stem-and-leaf plot of Mechanical Turk participation rates.**

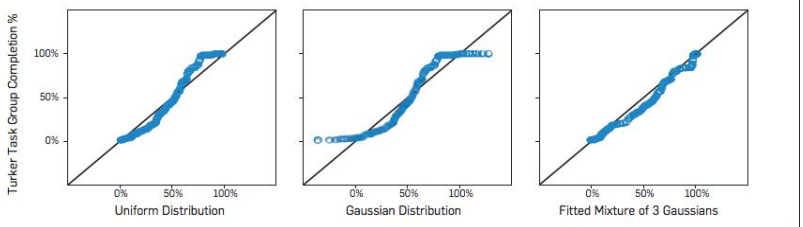
0	1	1	1	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	6	7	8	8	8	8	8	9
1	0	0	0	0	1	1	1	1	2	2	3	3	3	3	4	4	4	4	5	5	6	7	7	8	9	9	9	9	9	
2	0	0	1	1	1	5	7	8	9																					
3	0	0	1	2	3	3	3	4	6	6	8	8																		
4	0	0	1	1	1	1	3	3	4	5	5	5	6	7	8	9														
5	0	2	3	5	6	7	7	7	9																					
6	1	2	6	7	8	9	9	9																						
7	0	0	0	1	6	7	9																							
8	0	0	1	2	3	4	4	4	4	4	4	4	5	6	7	7	7	9												
9	1	3	3	5	7	8	8	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9				
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										

Source: Stanford Visualization Group; http://hci.stanford.edu/jheer/files/zoo/ex/stats/stem-and-leaf.html

Q-Q PLOTS

- Quantile-Quantile plot
- Compares two probability distributions by graphing their quantiles against each other
- If the two are similar, the plotted values will lie roughly along the central diagonal
- If the two are linear related, values will again lie along a line
- Limitation: assumes reader posses some statistical knowledge

Statistical Distributions: **Figure 2b. Q-Q plots of Mechanical Turk participation rates.**

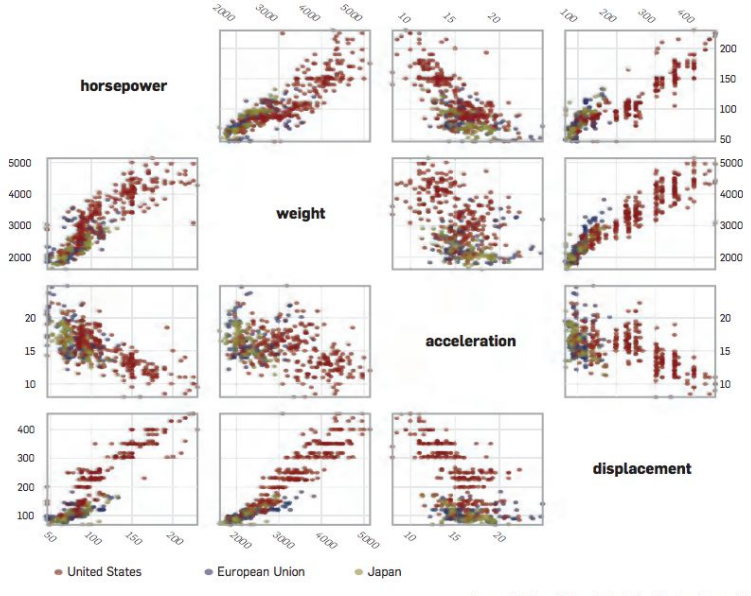


Source: Stanford Visualization Group; http://hci.stanford.edu/jheer/files/zoo/ex/stats/qqplot.html

SPLOM (SCATTER PLOT MATRIX)

- Represents the relationships among multiple variables
- Enables visual inspection of correlation between any pair variables

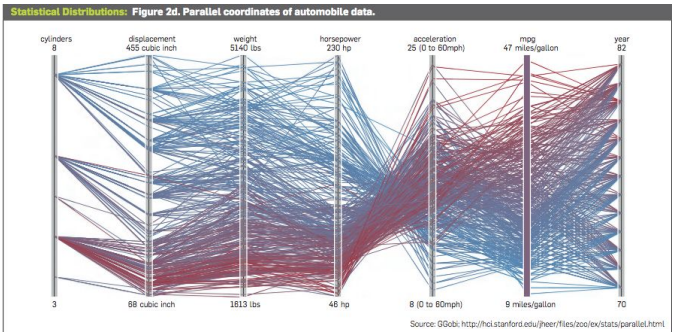
Statistical Distributions: **Figure 2c. Scatter plot matrix of automobile data.**



Source: GGobi; http://hci.stanford.edu/jheer/files/zoo/ex/stats/splom.html

PARRALLEL COORDINATES

- Chart that represents variables as vertical lines
- Each value on each row is plotted on the vertical lines, and then connected via a line
- Line crossing usually represents inverse correlation

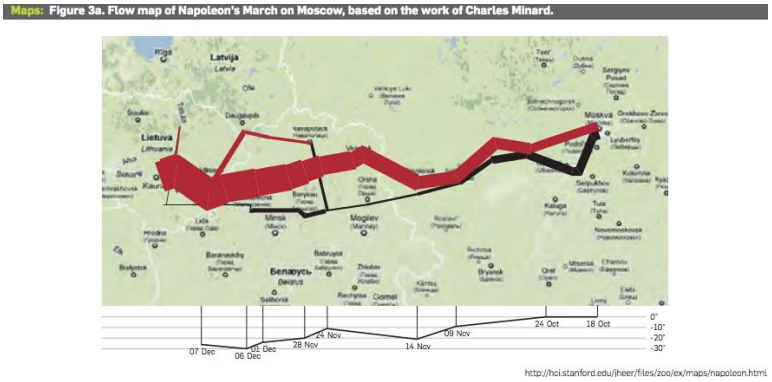


MAPS

- Usually maps geographical data

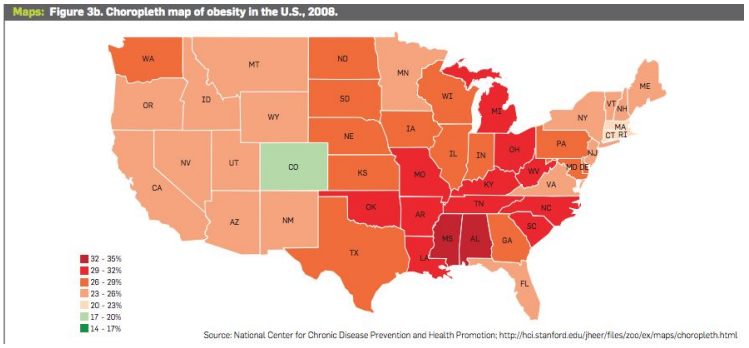
FLOW MAPS

- By placing lines on a map, it can depict the movement of a quantity in space and (implicitly) in time
- Flow lines usually encode a large amount of multivariate information
    - Path points
    - Direction
    - Line thickness
    - Colour
- All these can represent dimensions of information to the viewer



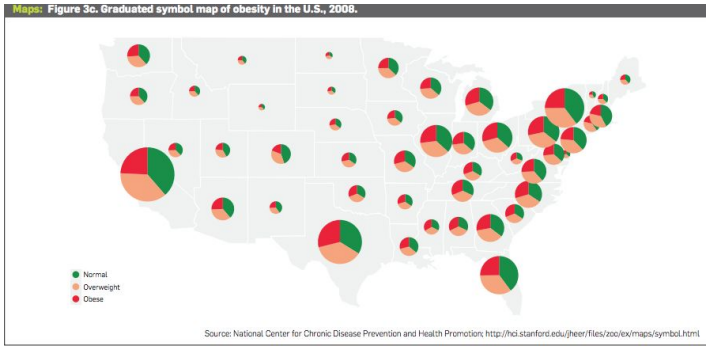
CHLOROPLETH MAPS

- Data usually collected and aggregated by geographical areas such as states
- Standard approach to using this data is to colour code geographical areas



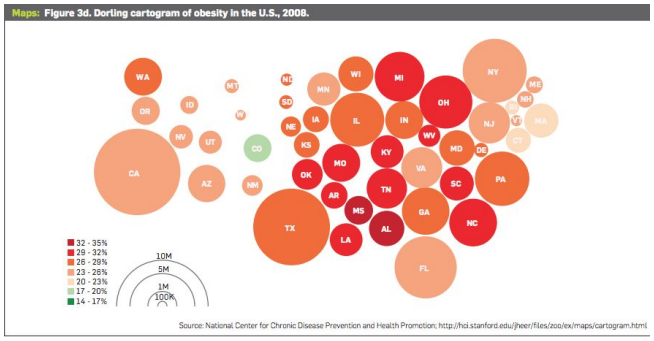
GRADUATED SYMBOL MAPS

- Like the chloropleth map, however instead of colour coding, it places a symbol on top of the geographical area – i.e. pie chart, to allow for more dimensions of information to be represented



CARTOGRAMS

- Distorts the shape of geographic regions so that the area directly encodes a data variable
- Can represent geographic data by size and colour

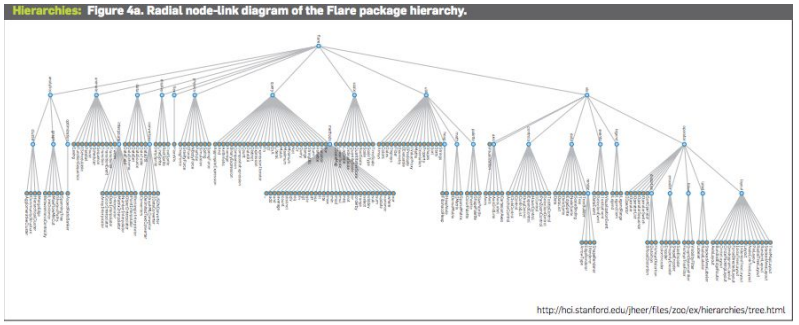


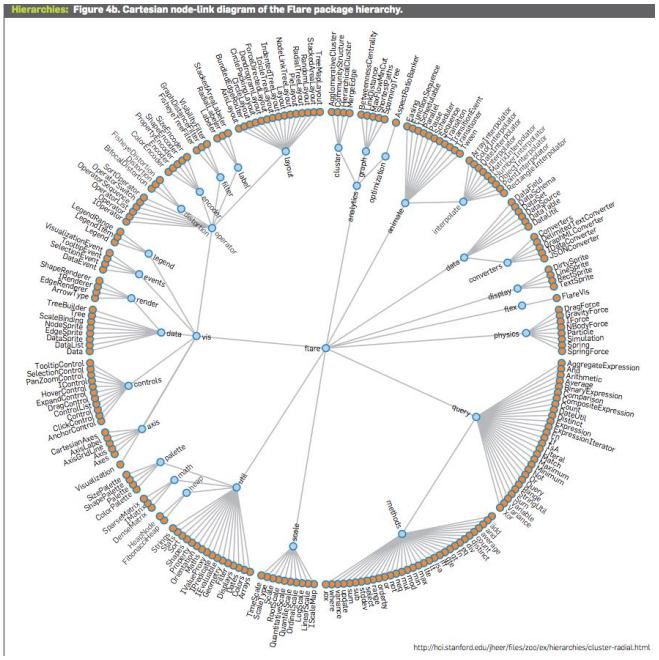
HIERARCHIES

- Charts representing data that can be organised into natural hierarchies

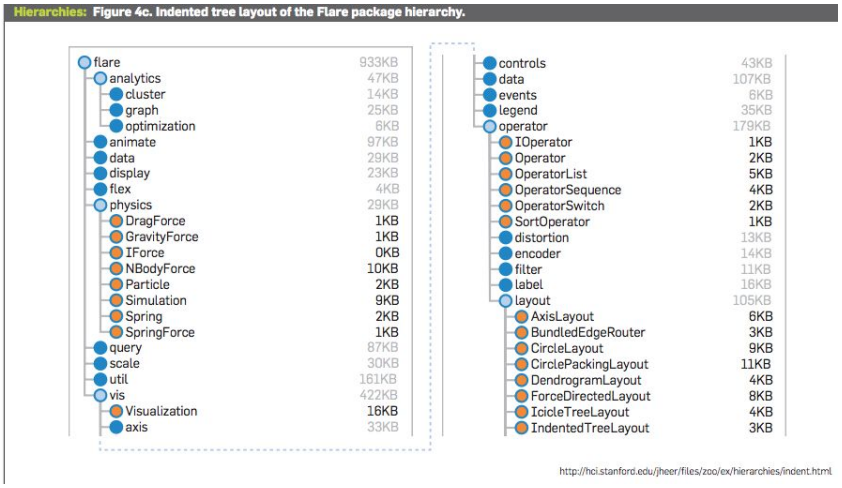
NODE-LINK DIAGRAMS

- Similar to a decision tree; a 2D representation of data and their hierarchies
- Dendrogram (or cluster) is an alternative
- Indented tree
    - A chart that represents hierarchies in a list through the use of bullet points
    - Multivariate data such as size can be represented adjacent to the data



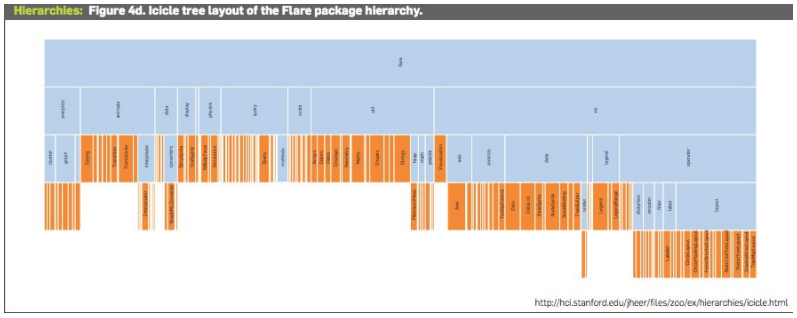


Indented tree

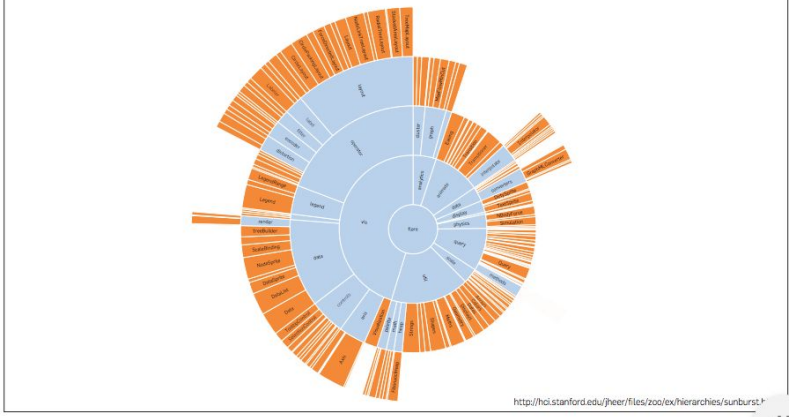


ADJACENCY DIAGRAMS

- A space filling variant of the node-link diagram
- Rather than drawing a line between parent and child in the hierarchy, nodes are drawn as areas, and their placement relative to adjacent nodes reveals their position in the hierarchy
- The colour portion represents the leaf node, or the end child in the hierarchy



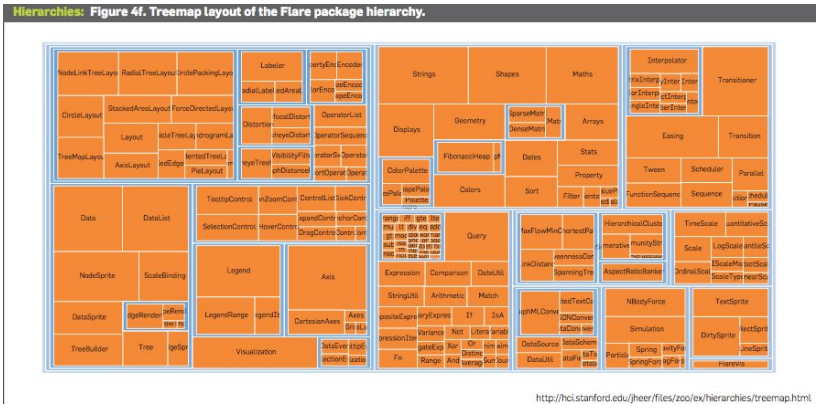
Hierarchies: **Figure 4e. Sunburst (radial space-filling) layout of the Flare package hierarchy.**

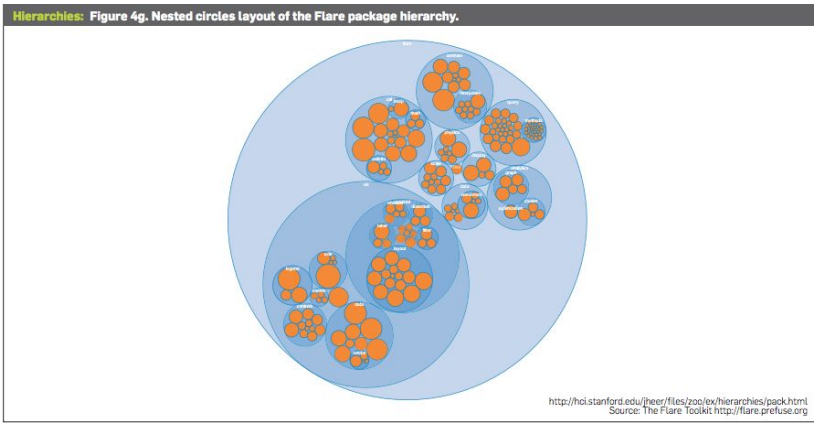


http://hci.stanford.edu/jheer/files/zoo/ex/hierarchies/sunburst.h

ENCLOSURE DIAGRAMS

- Also space-filling, however using containment instead of adjacency
- Subdivides the area into rectangles, the size of any node in the tree is quickly revealed
- Instead of using rectangles, using circles can produce a different sort of enclosure diagram



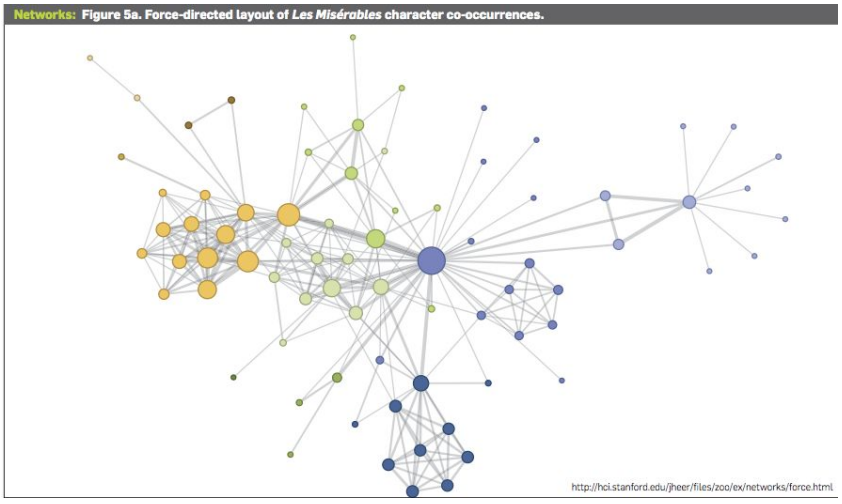


NETWORKS

- Allows data to be explored by relationship
- I.e. in a social network, who is friends with who? Who are the central players?

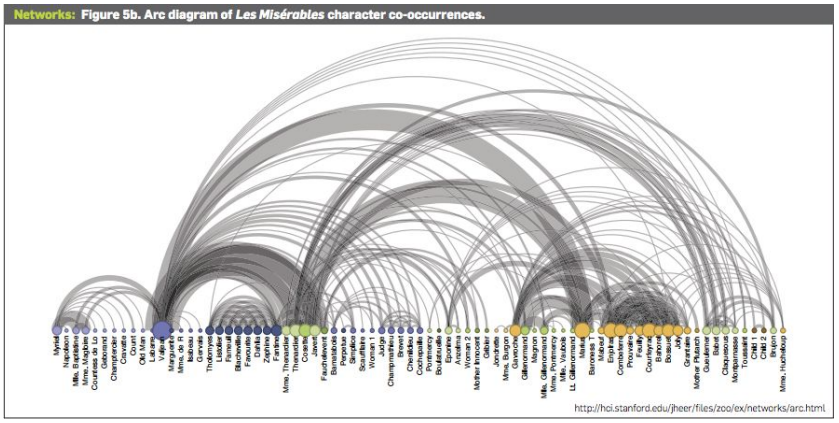
FORCED-DIRECTED LAYOUTS

- Model graphs as a physical system, nodes are charged particles that repel each other, and links are dampened springs that pull related nodes together
- Good starting point for understanding the structure of a general undirected graph
- Food network



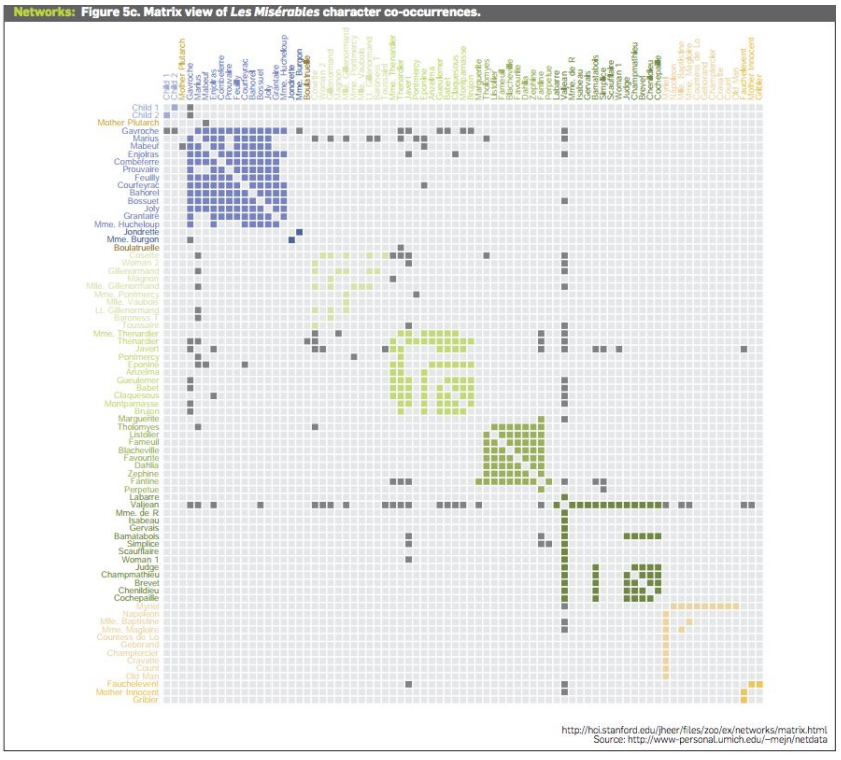
ARC DIAGRAMS

- One dimensional layout of nodes on a horizontal line
- With good ordering of nodes, can identify cliques and bridges
- Join by ‘arcs’



MATRIX VIEWS

- All variables are represented on a matrix, and if they are related the intersection is colour coded
- Cliques or related variables are colour coded



R

- Pch=as.numeric(Species) – changes symbol of each group

Week 3 – Data Manipulation

SUMMARISING DATA BY GROUPS (FACTORS)

- Applying a function to a single column
- Applying a function to a group of columns

FUNCTIONS – **output of functions always a list**

- Aggregate – compute summary statistic:
  - Creates a **table** by applying a function to data in **individual** columns, grouped by a factor (or factors)
  - Splits the data into subsets (factors), and computes the summary statistic for each
  - aggregate(iris[1:4], iris[5], mean)
- By – apply a function:
  - Enables a **function** to be applied across **multiple** columns of a data frame, grouped by a factor (or factors)
  - Calculating correlation of sepal length and width
    - By(iris, iris[5], function(df) cor(df$Sepal.Length, df$Sepal.Width))
    - Outputs into a list format
  - Function(df) – declaring a new function, df parameter is a **temporary data frame created for each factor**
  - Output = list:

```
Species: setosa
[1] 0.743
----------------------
Species: versicolor
[1] 0.526
----------------------
Species: virginica
[1] 0.457
```

  -
  - by(iris[1:4], iris[5], cor)
    - creates a correlation matrix between all columns by factor
- as.table:
  - Converts the output format of a function from a list to a table
  - as.table(by(iris, iris[5], function(df) cor(df[1], df[2])))
  - Output = table:

```
Species
    setosa versicolor  virginica
     0.743      0.526      0.457
```

  -
- as.data.frame:
  - Coerces the output of a table into a dataframe
  - Sepal.cor <- as.data.frame(as.table(by(iris, iris[5], function(df) cor(df[1], df[2]))))
  - Output = data frame

Sepal.cor

```
     Species  Freq
1     setosa 0.743
2 versicolor 0.526
3  virginica 0.457
```

  -
- colnames()
  - colnames(Sepal.cor) <- c("Species", "Sepeal.cor")
- Merging data frames and saving
  - Iris.cor <- merge(Sepal.cor, Petal.cor, by= "Species")
- Cbind()
  - Appends a data frame or vector by columns
  - Only works if the row names are the same

    - Niris <- cbind(iris, Sepal.ar, Petal.ar) # this binds columns together from the three data frames
    - Making new columns easier way
        - iris$Sepal.ar <- jakldjasd
- Removing columns
    - Iris$Sepal.Length <- NULL
    - For multiple columns
        - Niris <- niris[,c(5:7)]
- Boxplot by factor
    - boxplot(Petal.ar~Species, data=iris)
- with()
    - a generic function that evaluates an expression
- find the longest petal in each species:
    - use the aggregate function to Petal.Length column of the dataframe:
        - aggregate(iris[3], iris[5], max)
        - outputs the value of each petal by species
- find all measurements for the flower that has the longest petal
    - which.max()/ same applies to which.min()
    - determines the location i.e. index of the (first) maximum of a numeric vector
    - returns the row or col index/number
        - which.max(iris[,3])
            - returns the row number of the maximum value of col 3
    - to view all the other measurements
        - iris[which.max(iris[,3]),]
- to view all the measurements of the row with the longest petal for each species
    - by(iris, iris[5], function(df) df[which.max(df[,3]),])
    - output = list
- to tidy the output:
    - **do.call()**
        - constructs and executes a function call from a name or a function and a list of arguments to be passed to it
    - first assign the by function a name:
        - max.type = by(iris, iris[5], function(df) df[which.max(df[,3]),])
        - do.call(rbind, max.type)
- working with dates and times
    - to apply ‘which.min()’ or ‘which.max()’ to a date, you first need to convert character representation of date into a ‘date’ object using the ‘**as.Date()**’ function
    - to find print the values of the customer data by the min date of visit for each customer
    - min.visit <- by(dumpty, dumpty[1], function(df) df[which.min(as.Date(df[,2], “%d-%m-$y”)),])
    - do.call(rbind, min.visit)

Week 5 – Linear Regression

REGRESSION – LM()

- fitting the regression
  - fit <- lm(Function~Price)
    - response variable = function
    - how does the function change, when the price changes
- fitting line of best fit
  - abline(fit)
- to view **residuals**
  - fit$residuals

DIAGNOSTICS

- Residuals
  - Should be normally distributed
  - Normal distribution is the bell curve, so if plotting on a histogram, the centre should be highest, then reduce as it moves away from the centre
  - Hist(fit$residuals)

READING THE DIAGNOSTICS (SUMMARY)

- **Median**
  - In a good model, median should be close to 0**;** if the residuals are normally distributed, the median will be 0; otherwise skewing may be evident
  - The coefficients is the answer, the key values of the model
  - Pr(>|t|) – the p-value
    - Should be close to 0, it is a measure of the confidence level that the coefficient is correct, and has significance to the model
    - The *** next to the model will also indicate significance
    - Overall, if its close to 0, then some variables in the regression mean something
  - T value – looking for big t-values
  - Multiple r-squared – the % of the model of the variability in y; explains by the data
    - How much do the x values explain in the variability of y
  - Adjusted R-squared
    - The overall significance of regression: that all coefficients do not = 0
  - Median – residuals should be normally distributed, median should be close to 0, if not skewing could be going on
  - Estimate std (coefficient) – i.e. answer
  - PR(>|t|) – significants of the variables, look for the stars
  - T value – looking for big t-values
  - Multiple r-squared – the % of the model of the variability in y; explains by the data
  - P-value – overall significance of the regression, want the value to be close to 0

MULTIPLE LINEAR REGRESSION

NUMERICAL DATA

- Creating a linear regression model with multiple independent variables
- i.e.
  - test <- lm(Strength ~ Cement + Water, data=concrete))
  - summary(test)
    - to print out summary and evaluate the model
- to create a linear model with all variables in a dataframe
  - test1 <- lm(Strength~., data=concrete)

QUALITATITVE DATA

- Qualitative or categorical predictors are ones that are not numerical

- I.e. hair, gender, season, job type

- For this type of data, need to use an indicator (0,1) matrix to indicate category:

Person	Eye.colour		Person	Eye.Blue	Eye.Brown	Eye.Green
A	Blue		A	1	0	0
B	Brown		B	0	1	0
C	Green	--->	C	0	0	1
D	Blue		D	1	0	0
E	Blue		E	1	0	0

-

- EG. From slide 61
  - Lm(formula = log(price) ~ log(carat) + clarity)
  - What is the linear model?
    - Log(price) = log(carat) * carat + clarity + intercept
    - Log(price) = log(carat) * 1.8324 + clarity + 7.7884

for clarity 7 we are effectively multiplying the base price by $e^{1.029}$.

```
> d.fit

Call:
lm(formula = log(price) ~ log(carat) + clarity)

Coefficients:
(Intercept)    log(carat)      clarity2      clarity3
     7.7884        1.8324        0.4506        0.6052
   clarity4      clarity5      clarity6      clarity7
     0.7852        0.8264        0.9675        1.0290
   clarity8
     1.1138
```

-

What should a 1.5 carat, VVS1 diamond sell for?

```
log(price) = log(carat) (+ intercept) + clarity
log(price) = 1.8324 * ln(1.5) + 7.7884 + 1.0290
log(price) = 1.8324 * 0.4055  + 7.7884 + 1.0290
log(price) = 9.5603
    price  = $14,191
```

-

Week 6 – Networks

NETWORK GRAPH

- **Vertex/Vertices (nodes)**
  - Indicate each person/object
  - Represent the entities in the network
- **Edges (lines/arcs)**
  - Show that there is a connection between two vertices
  - Represents connections between these entities
  - Edges may be directed -> or undirected <->
  - Edges may be weighted to indicate the strength of the relationship or bond etc.
- Assume the relationships between two nodes are two-way

NETWORK STRUCTURES – TERMINOLOGY

- **Walk:** a sequence of links – could go backwards and forwards
- **Path:** a walk with no repeated vertices – from one point to another
- **Cycle:** a walk that begins and ends at the same vertex (loop)
- **Geodesic:** the shortest path between two vertices
- **Length:** the number of links in a walk or path – what is the length from M -> J?
- **Connected:** every pair of vertices is connected, no vertex is not connected to the network
- **Directed graphs:** all definitions above apply but travel on each edge is permitted in one direction only
- **Loop:** an edge from a vertex to itself; not connected to any other vertices
- **Complete:** a graph where every vertex is joined to every other vertex (clique)
- **Subgraph:** a subset of a graph
- **Clique:** a subgraph that is complete
- **Simple:** a graph with no loops or multi-edges (more than one edge between same pair of vertices) can be connected or disconnected

DEGREE OF VERTEX

- The most important measure of a vertex's importance in a network
- *Degree*: the number of edges connected to a vertex
- For directed graphs this is adapted to in-degree and out-degree
- How many lines going out of a vertex, or how many connections someone has

NETWORK STATISTICS

- *Diameter:* the longest geodesic (shortest path) between any two vertices
- *Average path length:* the average geodesic (shortest path) between two vertices
- *Degree distribution:* the probability distribution describing the magnitude of vertices in the network
- Analysis of these stats determine how connected, robust or fragile a network is

VERTEX CHARACTERISTICS

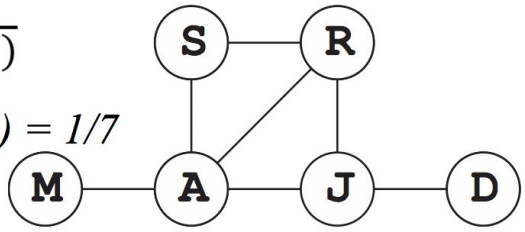
- The importance of a vertex is based on two factors:
  - Number of connections with other vertices
    - Degree
  - Centrality of vertex within the network (strategic power to control information)
    - Betweenness
    - Closeness
    - Eigenvector NOT EXAMINABLE – CANT CALCULATE BY HAND
      - Weights nodes according to the quality of its connections; nodes connected to important nodes are ranked higher

CLOSENESS CENTRALITY

- A vertex is 'close' if there is a short total distance between it and all the other vertices in the network
- Closeness centrality is the inverse of total distance between a vertex and the others

$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} dist(u, v)}$$

$c_{Cl}(J) = 1/(1+1+2+1+2) = 1/7$



-
- Measures how well node is connected locally; high CC nodes are strong local influencers

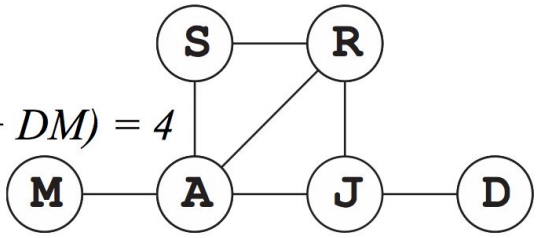
BETWEENNESS CENTRALITY

- This measure indicates the degree to which the vertex is 'between' other vertices
- Betweenness centrality sums the number of shortest paths between s and t, through vertex v (proportionally if more than one shortest path)

$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

$c_B(J) = (DR+DA+DS^*+ DM) = 4$

$c_B(M) = 0$



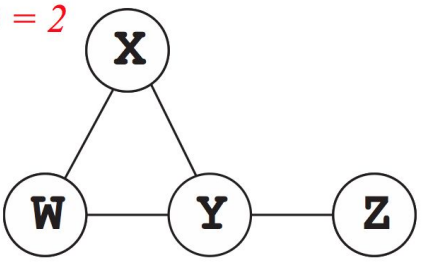
-
- High BC nodes act as hubs/relays/bridges

EXAMPLE

Class example – answers

For the graph below calculate the following:

- *Average path length ave(1, 1, 2, 1, 2, 1) = 1.333.*
- *Diameter for example d(W,Z) = 2*
- $d_Y = 3$
- *Degree distribution 2, 2, 2, 1*
- $c_B(Y)$ *W-Z, X-Z = 2*
- $c_{Cl}(Y)$ *1/(1+1+1) = 0.333.*
- *Cliques 2:W-X, W-Y, X-Y, Y-Z, 3:W-X-Y*



Week 7 – Decision Trees

MACHINE LEARNING

- Decision trees one of the most widely used and practical methods in machine learning
    - Model uses existing data attributes and values
    - Used to classify new instances
    - Robust to noise and missing values

DECISION TREES

- Decision trees consist of:
    - Leaf nodes (class/ the classifier) and non-leaf nodes, corresponding to the decision attributes
    - Branches – corresponding to the values of the decision attributes having either binary or multi-way splits
    - A path from the root to leaf node gives the class of the object

BUILDING A DECISION TREE

- Which attribute should be tested at a node?
- When should a node be declared a leaf?
- Many DT learning algorithms are variations on a core algorithm that employs *top-down, greedy search* through the space of possible decision trees
- The algorithm is aiming for *homogeneous* leaf nodes

TOP-DOWN INDUCTION: ID3 (ITERATIVE DICHOTOMISER 3)

- At each step, determine the "best" decision attribute, A, for next node
- Assign A as decision attribute for node
- For each attribute of A, create a new descendant

WHICH ATTRIBUTE TO SPLIT ON?

- At each stage of the process, we try to find the 'best' attribute and split to partition the data
- That decision may not be the best overall – but once it is made, we stay with it for the rest of the tree
- **This is called the greedy approach** and my not result in the best overall decision tree
- At each split, the goal is to increase the homogeneity of the resulting datasets with respect to the class or target variable (which we are trying to classify)

HOMOGENEITY

- When all values in the attribute are the same

INFORMATION GAIN

- Which attribute to choose for splitting?
- Measures how well a given attribute separates the training examples into homogeneous groups according to target classifications
- Used as the splitting criteria for building a tree; choose the attribute that provides the greatest information gain
- Determined by information theory call entropy

ENTROPY

- Measures the uncertainty in a random variable; the degree of randomness
- The more uncertain or random the event is, the more information it will contain

CALCULATING ENTROPY

For a two class problem: c_1 and c_2:

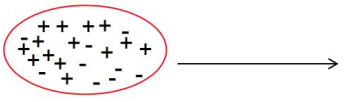
- P indicates the probability of belonging to each class, the number in each class is $N_{c1} + N_{c2} = N$.

$$\begin{aligned} \text{Entropy(S)} &= -P_{c1}\log_2(P_{c1}) - P_{c2}\log_2(P_{c2}) \\ &= -\frac{N_{c1}}{N}\log_2(\frac{N_{c1}}{N}) - \frac{N_{c2}}{N}\log_2(\frac{N_{c2}}{N}) \end{aligned}$$

For a multi-class problem

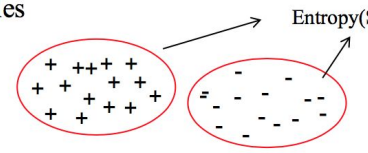
$$\begin{aligned} \text{Entropy(S)} &= -\sum_{i=1}^{C} P_i \log_2(P_i) \\ &= -\sum_{i=1}^{C} \frac{N_i}{N}\log_2(\frac{N_i}{N}) \end{aligned}$$

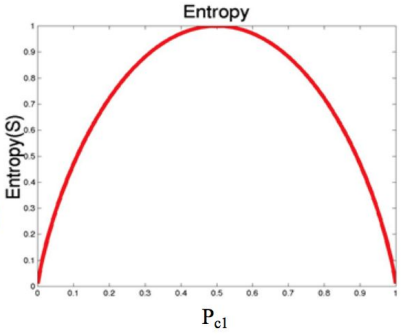
- E.g. Suppose S is a collection of 14 examples, 9 positive and 5 negative → [9+,5-]



$$\begin{aligned} \text{Entropy(S)} &= -P_{c1}\log_2(P_{c1}) - P_{c2}\log_2(P_{c2}) \\ &= -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) \\ &= 0.940 \end{aligned}$$

E.g. suppose S has all positive or all negative examples





- Entropy is 0 (minimum) if all members belong to the same class. Entropy is 1 (maximum) if the collection consists of equal number of positive and negative examples. Note: $0\text{Log}_2 0 = 0$

The previous example as a spreadsheet:

- If your calculator can't work out logs to base 2 then use the following:

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)} \approx \frac{\log_{10}(x)}{0.3010}$$

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

- Note: $\log_2$ yields entropy in units called "shannons"
- **LOG2*Pc1 = log(pc1)/log(2)**

INFORMATION GAIN

- The expected reduction in entropy caused by partitioning the examples according to an attribute A
- The higher the information gain, the better
- Choose the attribute to branch off next that provides the largest information gain, according to the 'greedy' principle

Gain(S,A) of an attribute A, relative to a collection of examples S (with v groups having | | elements) is:

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

Entropy before split — Expected entropy after split

-

HOW ID3 USES INFORMATION GAIN

- At each step, determine the "best" decision attribute, A, for the next node
- Assign A as decision attribute for node
- For each value of A, create a new descendant
- Sort training examples to that node according to the attribute value of the branch
- If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes

TERMINOLOGY

- Instance: single row in a data set; also called an example or object
- Attribute: an aspect of an instance; also called feature, variable. Can be categorical or numerical
- Value: category that an attribute can take
- Concept: the thing to be learned; also called **class** or **target**
- Usual to have several decision attribute and one class attribute

EXAMPLE

Training set (*S*): Initial entropy before splitting based on 9 Yes/5 No:

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

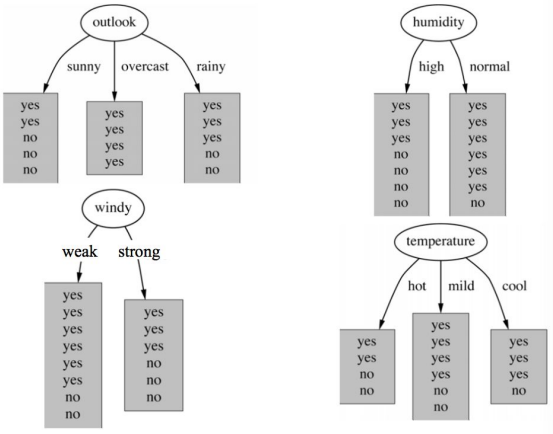
-

Which attribute to select?

Remember - ID3 chooses the attribute which provides the greatest information gain.

i.e. the attribute that gives the greatest reduction in entropy, or the 'purest' result.

We next calculate the information gain for each attribute in turn.



FIT3152 Data analytics– Lecture 7

-

Information gain: Temperature

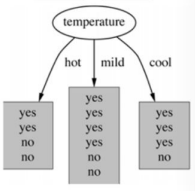
Gain(S,Temperature):

$$\text{Gain}(S,\text{Temperature}) = \text{Entropy}(S) - \sum_{v\ \{\text{hot, mild, cool}\}} \frac{|S_v|}{|S|}\text{Entropy}(S_v)$$

$$= 0.940 - \frac{4}{14}\text{Entropy}(S_{\text{hot}}) + \frac{6}{14}\text{Entropy}(S_{\text{mild}}) + \frac{4}{14}\text{Entropy}(S_{\text{cool}})$$

$$= 0.940 - \frac{4}{14}\cdot 1 + \frac{6}{14}\cdot 0.918 + \frac{4}{14}\cdot 0.811$$

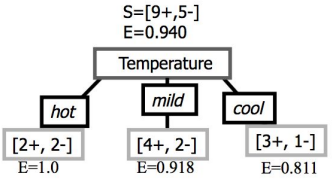
$$= 0.029$$



S_{hot} :[2+,2-] $\text{Entropy}(S_{\text{hot}}) = 1$

S_{mild} :[4+,2-] $\text{Entropy}(S_{\text{mild}}) = 0.918$

S_{cool} :[3+,1-] $\text{Entropy}(S_{\text{cool}}) = 0.811$



-

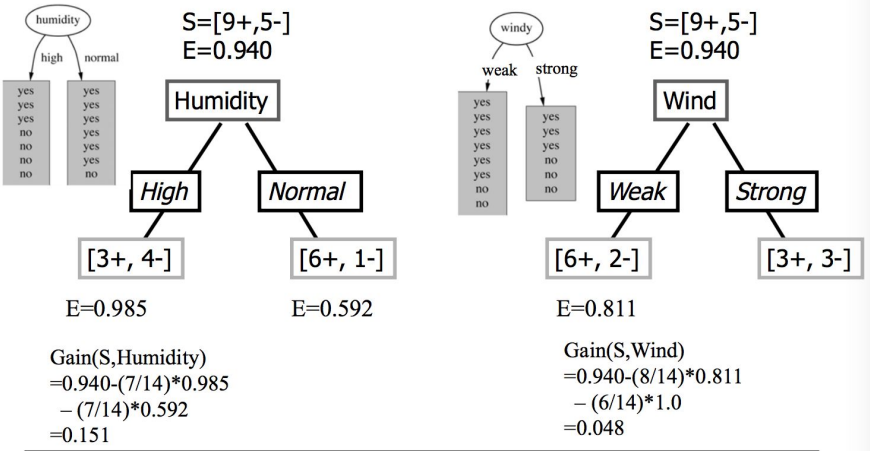
Information gain: Temperature

As a spreadsheet showing initial entropy and subsequent information gain:

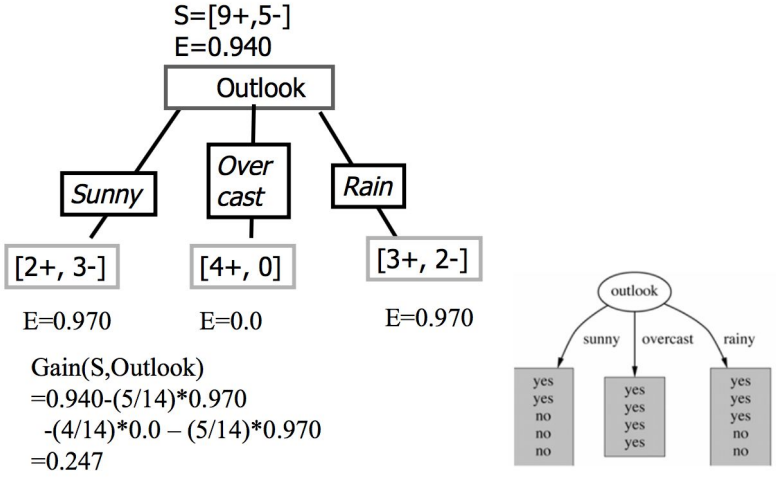
Initial State	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Entropy(S)	9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Temperature	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	2	2	0.5000	-1.0000	0.5000	-1.0000	1.0000
Mild	4	2	0.6667	-0.5850	0.3333	-1.5850	0.9183
Cool	3	1	0.7500	-0.4150	0.2500	-2.0000	0.8113
EEntropy(Temp)							0.9111
Gain(S, Temp)							**0.0292**

Information gain: Wind, Humidity



Information gain: Outlook



Calcs: Humidity, Wind, Outlook

Humidity	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	3	4	0.4286	-1.2224	0.5714	-0.8074	0.9852
Normal	6	1	0.8571	-0.2224	0.1429	-2.8074	0.5917
EEntropy(Humidity)							0.7885
Gain(S, Humidity)							**0.1518**

Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	6	2	0.7500	-0.4150	0.2500	-2.0000	0.8113
Strong	3	3	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.8922
Gain(S, Wind)							**0.0481**

Outlook	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Sunny	2	3	0.4000	-1.3219	0.6000	-0.7370	0.9710
Overcast	4	0	1.0000	0.0000	0.0000	0.0000	0.0000
Rain	3	2	0.6000	-0.7370	0.4000	-1.3219	0.9710
EEntropy(Outlook)							0.6935
Gain(S, Outlook)							**0.2467**

Attribute giving greatest information gain

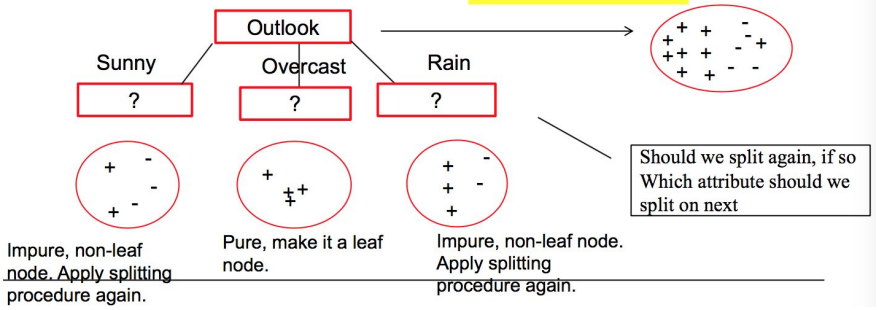
Which attribute to choose? Outlook, Temperature, Humidity or Wind?

Gain(S, Temperature) = 0.029

Gain(S, Humidity) = 0.151

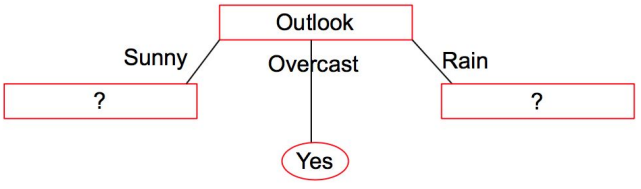
Gain(S, Wind) = 0.048

Gain(S, Outlook) = 0.247 ← Choose this one!



Which attribute to split on next?

Now, starting with Sunny, which attribute should be split on next? Temperature, Humidity or Wind?

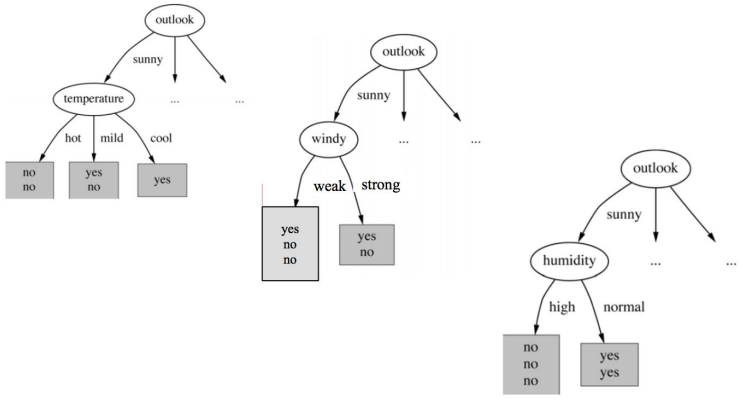


ID3 Step 2: gain(S_sunny, ???)

Now consider subset corresponding to "Sunny":

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
~~D3~~	~~Overcast~~	~~Hot~~	~~High~~	~~Weak~~	~~Yes~~
~~D4~~	~~Rain~~	~~Mild~~	~~High~~	~~Weak~~	~~Yes~~
~~D5~~	~~Rain~~	~~Cool~~	~~Normal~~	~~Weak~~	~~Yes~~
~~D6~~	~~Rain~~	~~Cool~~	~~Normal~~	~~Strong~~	~~No~~
~~D7~~	~~Overcast~~	~~Cool~~	~~Normal~~	~~Strong~~	~~Yes~~
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
~~D10~~	~~Rain~~	~~Mild~~	~~Normal~~	~~Weak~~	~~Yes~~
D11	Sunny	Mild	Normal	Strong	Yes
~~D12~~	~~Overcast~~	~~Mild~~	~~High~~	~~Strong~~	~~Yes~~
~~D13~~	~~Overcast~~	~~Hot~~	~~Normal~~	~~Weak~~	~~Yes~~
~~D14~~	~~Rain~~	~~Mild~~	~~High~~	~~Strong~~	~~No~~

Which attribute to split on next?



Entropy after "Sunny" Outlook

The entropy of each branch of the decision tree after split on outlook is shown below.

- Information gain in descendant trees is now measured as change in the entropy of each branch
- For example Entropy(Sunny) = 0.971

Outlook	Yes	No	P(Yes)	og2(Yes)	P(No)	og2(No)	Entropy
Sunny	2	3	0.400	-1.322	0.600	-0.737	0.971
Overcast	4	0	1.000	0.000	0.000	0.000	0.000
Rain	3	2	0.600	-0.737	0.400	-1.322	0.971
EEntropy(Outlook)							0.694

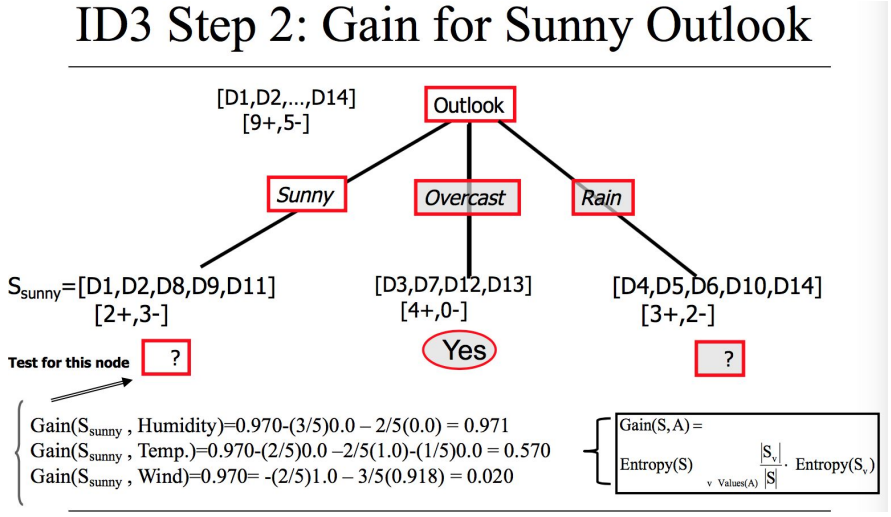
ID3 Step 2: Gain for Sunny Outlook

Sunny, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	0	2	0.0000	0.0000	1.0000	0.0000	0.0000
Mild	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
Cool	1	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.4000
Gain(Sunny, Temp)							**0.5710**

Sunny, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	0	3	0.0000	0.0000	1.0000	0.0000	0.0000
Normal	2	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.0000
Gain(Sunny, Humid)							**0.9710**

Sunny, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	1	2	0.3333	-1.5850	0.6667	-0.5850	0.9183
Strong	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.9510
Gain(Sunny, Wind)							**0.0200**

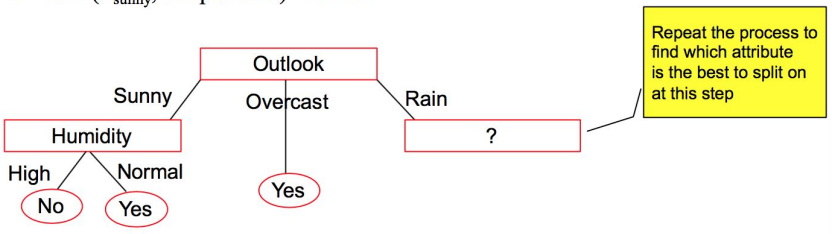
-



-

Which attribute to choose? Temperature, Humidity or Wind?

- $Gain(S_{sunny},Wind) = 0.020$
- $Gain(S_{sunny},Humidity) = 0.971$ ← Choose this one!
- $Gain(S_{sunny},Temperature) = 0.570$



-

Not all leaves need to be 'pure'.
Splitting stops when the data can't be split any further.

UNKNOWN OR MISSING ATTRIBUTE VALUES

- Possible ways of dealing with missing attribute values
  - Remove data objects with missing values – drastic
  - Impute the missing value by assigning common value of A among other examples with same target value
  - Some cases, better to leave value as "null", "zero" or "missing"

MERICS FOR PERFORMANCE EVALUATION

- Data divided into training and testing data; usually 70/30 split
- The training data is used for building the model, and the test data will test the model on unseen data
- We can calculate the accuracy based on a confusion matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Most widely-used metric:

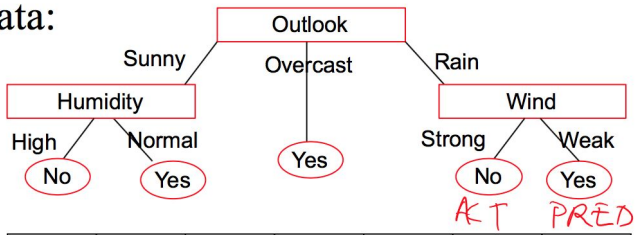
$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Also:

Precision = tp/(tp + fp)

Sensitivity = tp/(tp+fn)

- Let's see what our model would predict using the test data:



Day	Outlook	Temperature	Humidity	Wind	Play	Predict
D15	Sunny	Mild	Normal	Strong	No	Y
D16	Sunny	Hot	High	Weak	Yes	N
D17	Rain	Hot	High	Weak	No	Y
D18	Overcast	Cool	High	Strong	No	Y
D19	Overcast	Mild	Normal	Weak	Yes	Y
D20	Rain	Mild	Normal	Weak	Yes	Y

-

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

Most widely-used metric:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2+0}{6} = 33.3\%$$

-
-

Week 8 – improving the basic decision tree

DECISION TREES

PROS AND CONS OF DECISION TREE

Pros:

- Easy to interpret
- Able to find most important/discriminating attributes
- Overall accuracy good

Cons:

- Unstable – small changes may lead to completely different decision tree
- Can become overly complex

CLASSIFICATION ALGORITHMS

- Objective is to produce a model from labeled dataset (past data)
- Model is then used to label new, unlabeled classes (predict new data)

PERFORMANCE METRICS

- Accuracy = TP + TN/TP + FN + FP + TN
- Precision = TP/TP+FP (when test class is = YES)
    - How many selected items are relevant?
- Recall = TP/TP + FN (when actual class is = YES)
    - How many relevant items are selected?

OVER/UNDERFITTING

- Over fitting – a model that does not generalize well
    - Model is excessively complex
    - Performs well on training data, not unseen data
    - Low recall
- Under fitting – model that is too simple
    - Model too simple to give accurate labels
    - Performs poorly on both training and unseen data
    - Low precision

CROSS VALIDATION

- K-fold cross validation steps:
    - Data is partitioned into k disjoint subsets
    - K-fold: train on k-1 partitions, test on remaining one
        - I-e k=10, train on 1-9 partitions, test on 10^{th} partition
        - Train on 2-10 partitions, test on 1^{st} partition
    - Repeat until all partitions have been trained and tested on
    - Take the average performance across all partitions
    - 'Stratification' ensures classes are properly represented across partitions

TREE SIZE – WHAT DOES THAT MEAN?

- Leaf node count – the number of rules encoded in the decision tree
- Tree height – corresponds to the maximum rule length; how long it takes to get to the node

DECISION TREE – POST PRUNING

- Tree is first grown out as usual; depth of the tree is reduced by replacing sub-trees with leaf nodes
- The general concept is to remove rules on the basis that it has little or negligible effect on error rate
- Steps
    - Keep a portion of data as a validation set
    - Grow tree out to usual size as per no pruning
    - Prune by replacing sub-trees with leaf nodes, by measuring error against validation set

- Class label of replacement leaf node is determined by the **majority of labels** of removed sub-tree i.e. either YES or NO, which is higher

<u>BAYESIAN CLASSIFIERS</u>

BAYE'S THEOREM

- Calculate the conditional probability that x has attribute A given it is of class C:
  - $P(A|C) = P(A \cap C)/P(C)$ or rewritten as:
  - $P(C) * P(A|C) = P(A \cap C)$

NAÏVE BAYES CLASSIFIER

- A family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naïve) independence assumptions between the features
- A popular/ baseline method for text categorisation, the problem of judging documents as belonging to one category or another (such as spam or legitimate, sports, politics etc)
- Used to classify data into an unknown class
  - Work out for each from past data:
    - Probability of (N) & Probability of (Y)
    - Then multiply by the probability of each descriptor of probability (N) or (Y)
    - Then take the maximum value

<u>CLASSIFIER MODEL EVALUATION</u>

Measuring the effectiveness of each model, and determining which is the 'best' to use with the data

- Confusion matrix & performance measures (accuracy, precision, recall)
- ROC (Receiver Operating Characteristic) for binary classifiers;
- AUC
- Lift charts

RECEIVER OPERATING CHARACTERISTIC

- Characterise the trade-off between positive hits and false alarms between the train and test data
- ROC plots True Positive Rate, TPR on y-axis; against False Positive Rate, FPR, (on x-axis)
- Performance of a single classifier presented as a single point of ROC curve
- Calculating TPR and FPR:
  - TPR also called sensitivity, indicates how good a test is for correctly predicting "yes" when it should predict "yes": TPR = TP/TP+FN
  - FPR also known as false alarm, is: FPR = FP/FP+TN (misclassified)
- The values of these points are then plotted on the ROC chart as coordinates
  - If point sits below the diagonal line, it is a bad classifier; conversely, if the point sits above the line, than it is a better classifier; if it sits on the line, it is a guess 50:50; the further from the line the points sits, the better/worse

COMPARIGN CLASSIFIERS USING ROC CURVES

- ROC curves can be graphed for varying confidence 'threshold' values of a single classifier
- Calculate the TPR & FPR's for each confidence level (i.e. 0.3, 0.4, 0.6, 0.7, 0.8, 1) for the classifier model, the plot on the ROC chart

ROC CURVE – INTERPRETATION

- (TPR, FPR):
  - (0,0): If declare everything to be negative class
  - (1,1): If declare everything to be positive class
  - (1,0): ideal
- Diagonal line:
  - If it sits on the line, random guess
  - If it sits below diagonal line:
    - Prediction is opposite of the true class
    - Prediction is wrong

AREA UNDER CURVE

- For binary data
- Overall single measure of test performance
- Comparisons between two tests based on differences between estimated AUC

INTERPRETATION OF AUC

- Guidelines for interpreting AUC values:
    - AUC=0.5; no discrimination i.e. might as well flip a coin
    - 0.7 <= AUC < 0.8; Acceptable discrimination
    - 0.8 <= AUC < 0.9; Excellent discrimination
    - AUC => 0.9; Outstanding discrimination (but extremely rare)

LIFT

- For binary classification and prediction models
    - A measure of the effectiveness of a predictive model, calculated as the ratio between the results obtained with and without the predictive model
    - Lift charts are visual aids for measuring performance
    - Lift factor = success rate with model/ success rate without model

Look further..

~~ENSEMBLE METHODS~~ NOT EXAMINABLE

- ~~Used to improve classification accuracy~~
- ~~Advantage: often improves predictive performance~~
- ~~Disadvantage: usually produces output that is hard to analyse~~

~~BAGGING (BOOTSTRAP AGGREGATING)~~

- ~~Builds several data sets from original; each data set can be used to build a classifier~~
- ~~Repeatedly samples from a data set using uniform distribution~~
    - ~~Some original data objects can appear more than once~~
    - ~~Some may not appear at all~~
    - ~~Each bootstrap contains approx. 63% of the original data set~~
- ~~Each bootstrap is the same as the original data set~~
- ~~Build a classifier on each of the new data sets – e.g. decision tree~~
- ~~Use the classifier to 'vote' on unseen data – i.e. the test data~~

~~WHEN TO USE BAGGING~~

- ~~Useful if there is noise in the data~~
- ~~Useful for unstable classifiers – i.e. if small changes in the training data cause large changes in the classifier; unstable classifiers include decision trees, neural networks, linear regression~~
- ~~Not recommended for stable classifiers – K Nearest neighbour, Naïve Bayes~~

~~BOOSTING~~

- ~~To enable improved classification of **imbalanced** data sets~~
    - ~~i.e. A credit data set where most customers are a 'good' credit risk and minority are a 'bad' credit risk 70:30~~
- ~~New training data sets are generated:~~
    - ~~Modifies the distribution of training data to enable the classifier to focus on objects that are difficult to classify~~
    - ~~Assigns weight to each training example, the higher the weight for harder to classify models~~
    - ~~Uses a weighted average of results obtained from applying a prediction method to various samples~~
    - ~~Uses incremental improvement~~
- ~~Records that are wrongly classified will have their weights increased~~
- ~~Records that are classified correctly will have their weights decreased~~
- ~~Pros: tends to achieve better accuracy than bagging; can lead to over fitting~~

RANDOM FORESTS

- A refinement of bagged decision trees; specifically used for decision trees
- Combines prediction made by multiple decision trees
- Procedure
    - Create multiple data sets from the original training set using subsets of the objects and subsets of attributes
    - Build a decision tree classifier for each new data set – vary the parameter values
    - Combine the classifiers so that they 'vote' on the prediction for unseen data (i.e. the test data)
- Advantages:
    - More accurate than individual trees on large data sets
    - No need to prune
    - Not sensitive to outliers
    - Over fitting not a problem
- Differences between Bagging and Random Forests
    - Bagging varies data sample and the number of trees
    - Random Forests varies the **attributes** as well as the data sample and the number of trees

Week 9 – Cluster Analysis

SUPERVISED AND UNSUPERVISED LEARNING

SUPERVISED LEARNING ALGORITHMS
- Algorithms given labelled examples (target class) for the various types of data that need to be learned
- Classification algorithms such as decision trees, neural networks, Bayesian classifiers

UNSUPERVISED LEARNING ALGORITHMS
- Data is unlabelled (no predefined class); learning algorithms attempt to find patterns within the data or to cluster the data into groups or sets
- Clustering algorithms

CLUSTER ANALYSIS
- Finding groups of objects such that the objects in the group are similar or related to each other, and different/unrelated from objects in the other group
- Clusters are separated and divided into homogeneous groups (clusters)
- Clustering find structure in data i.e. group together documents/websites with similar content

CLUSTERING DEFINITION
- Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that
    - Data points in one cluster are more similar to one another
    - Data points in separate clusters are less similar to one another
- Similarity measures:
    - Euclidean Distance
    - Other distance-based measures (i.e. Manhattan)
    - (Other measures if the attribute values are not continuous)
- Why cluster?
    - Clustering identifies natural groups in the data
    - The set of data to be analysed can be reduced, based on typical examples from clusters

TYPES OF CLUSTERINGS
- Partitional Clustering
    - Dividing data objects into non-overlapping (unique) subsets, such that each data object is in exactly one subset
- Hierarchical Clustering
    - A set of nested clusters organised as a hierarchical tree

CLUSTERING TECHNIQUES

K-MEANS CLUSTERING
- Partitional clustering approach
- Each cluster is associated with a centroid (centre point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *k*, must be specified
- The basic algorithm:
    - Select *k* points (at random) as the initial centroids
    - **Repeat**
        - Form *k* clusters by assigning all points to the closest centroid
        - Re-compute the centroid of each cluster
    - **Until** centroids don't change
- Finding the centroids
    - Need to calculate the 'distance' between each point and all the centroids
- What k-Means is aiming to do
    - Pre-processing

        - Need to normalise the data, to ensure large valued attributes wont exert greater influence on the clustering
        - Eliminate outliers
    - Post-processing
        - Eliminate small clusters that may represent outliers
        - Split loose clusters – clusters with relatively high SSE
        - Merge clusters that are 'close' and that have relatively low SSE
- K-Means; how do we know which *K* to use?
    - Trial and error
- Initial centroids influence final clusters
    - Run multiple times
    - Select more than *k* initial centroids and then select among these initial centroids
- The basic *k*-Means algorithm can yield empty clusters
    - Must choose another seed
    - Choose a point from the cluster with highest SSE – this tends to have the best effect of splitting the cluster
    - If there are multiple empty clusters, the above can be repeated several times
- Limitations of *k*-Means clustering
    - Problems when clusters are of differing:
        - Sizes
        - Densities
        - Non-globular shapes
        - Contain outliers

HIERARCHICAL CLUSTERING

- Produces a set of nested clusters organised as a **hierarchical tree**
    - Records the sequences of mergers or splits
- Visualised as a dendrogram
- Strengths:
    - Don't need to assume any particular number of clusters
        - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- A 'dendrogram' shows how the clusters are merged hierarchically (looks like tree diagram)
    - It decomposes data objects into several levels of nested partitions (tree of clusters)
    - A clustering of the data objects is obtained by cutting the dendrogram at the desired level, and then each connected component forms a cluster
    - The height represents the distance between clusters
- Two main types of hierarchical clustering
    - Agglomerative:
        - Start with the points as individual clusters
        - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
    - Divisive:
        - Start with one, all-inclusive cluster
        - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Agglomerative clustering algorithm
    - Most popular; distance matrix stores the distances between each cluster
    - Basic algorithm is straightforward
        - Compute distance matrix; let each data point be a cluster
        - **Repeat**
        - Merge the two closest clusters
        - Update the distance matrix, **until** only a single cluster remains
    - Use either MIN, MAX, Group Average, Distance Between Centroids
    - Strength of MIN
        - Can handle non-elliptical shapes
    - Limitations of MIN
        - Sensitive to noise and outliers

    - Strength of MAX
        - Less susceptible to noise and outliers
    - Limitations of MAX
        - Tends to break large clusters
        - Biased towards globular clusters
    - Strength of Group Average
        - Less susceptible to noise and outliers
    - Limitations Group Average
        - Biased towards globular clusters
- Hierarchical clustering: Problems and Limitations
    - Once a decision is made to combine two clusters, it cannot be undone
    - Different methods have problems with one or more of the following:
        - Sensitivity to noise and outliers
        - Difficulty handling different sized clusters and convex shapes
        - Breaking large clusters

Week 10 – Text Analytics

TEXT ANALYTICS

- Aims to extract useful knowledge from text data:
  - Data is unstructured or semi-structured text
  - Ultimate goal is to get meaning in an automated way
  - Text data converted to counts, frequency distribution, correlations.. (categories)
  - Text data associated with sentiments via a library (e.g. LIWC)

APPLICATIONS OF TEXT ANALYSIS

- Text classification applications
  - Classifying emails to: filter out spam, sort into different folders
  - Classifying document streams: identify news feeds of interest
  - Sentiment analysis: determine public opinion
- Text clustering
  - Exploring text to find common words or features
  - Discovering group of documents with similar content
- Terms
  - Document:
    - A piece of text (from a book to a single sentence), tweet, job application, customer feedback, emails, blog posts, etc
  - Term (or token)
    - Documents consist of individual token or terms – usually words
  - Corpus
    - A collection of documents to be analysed
  - Feature set – Dictionary
    - All features in the corpus (may be all words or terms, but often a reduced set of words or terms)
- Obtaining structure from text by conversion to categories and counts:
  - Bag of words
  - Vector space model

VECTOR SPACE MODEL (VSM)

- Uses the bag of words approach
  - Each document assumed to be just a collection of words
  - Implicit assumption that the order of the words in a document does not matter
  - Syntactically similar documents are semantically similar – which is often the case
- A Term-Document Matrix (TDM) represents each document by a vector of words:
  - Key words are extracted from all documents (by how many times they are represented)
  - **A document is represented as a vector** is high dimensional space corresponding to all the key words
  - Proximity of documents is measured using a similarity measure defined over the vector space
  - Issues
    - How to select keywords to capture 'basic concepts'
    - How to assign weights to each term?
    - How to measure the similarity?
  - A Term-Document Matrix (TDM) is created from all documents in the corpus:
    - Doc1: {My dog ate my homework}
    - Doc2: {My cat ate the sandwich}
    - Doc3: {A dolphin ate the homework}
  - **Each column of the TDM represents a word (or term) and each row represents a document (as a vector)**
  - Without further processing of the above corpus (3 documents)
    - Tokens: a(1), ate(3), cat(1), dog(1), dolphin(1), homework(2), my(3), sandwich(1), the(2)

REFINING THE ‘BAG OF WORDS’

- The bag of words is improved by addressing the following:
    - Upper and lower case have the same meaning
    - Some frequently occurring words are not useful
    - Tense may make similar words appear different
    - Groups of words may be important for meaning

EXTRACTING STRUCTURE FROM TEXT

- Steps
    - Tokenise
    - Convert case
    - Remove stop words
    - Stem
    - Lemmatize
    - Create n-grams

TOKENISATION

- Tokens are discrete blocks of text, usually words
- Tokenising breaks up the text into tokens
    - Text document is split into a stream of words
    - Remove all punctuation (“, ., etc)
    - Replace tabs and other non-text characters by single white spaces
    - Merge all remaining words from all documents – this forms the dictionary of the documents collection (corpus) so if there is two ‘my’, then merge into one

FILTERING

- The process of removing ‘unnecessary’ words from the dictionary, for example:
    - Remove stop words – articles (a, an, the), conjunctions (and, but), prepositions (it, my, in, under)
    - Remove commonly occurring words that may not assist with clustering
    - Remove infrequently occurring words

STOP WORDS – ALSO CALLED NOISE WORDS

- Commonly occurring words such as the, an
- Articles: a, am, the, of..
- Auxiliary verbs: is, are, was, were..
- The process of filtering out the above is called stopping.
- Above example:
    - Remove my, the

STEMMING

- A natural group of words with the same (or very similar) meaning
    - Stemming reduces words to their stem or root form
    - Reduces the size of the dictionary
    - The Porter algorithm (1979) – most commonly used
- Examples
    - Computational, computing.. reduced to *comput*
    - Argue, argued, argues, and arguing reduced to *argu*
    - Ate => at

LEMMATIZATION

- Advanced form of stemming
    - Takes context and ‘part of speech’ into account
    - Eg. ‘meeting’
- Stemming
    - Stems to meet
- Lemmatization:
    - Reduces to ‘meet’ when it is a verb

    - Reduces to ‘meeting’ when it is a noun
- Time consuming – stemming used more often

CASE NORMALISATION
- Converts the entire corpus to lower (or upper) case
- Reduces size of dictionary, Both cases treated the same

N-GRAMS
- Enhances the bag of words approach
- Frequent sequences of words are identified and included in the bag of words (in addition to the individual words)

TERM-DOCUMENT MATRICES
- Most entries = 0
- Measures frequency of a word for a specific document
    - Terms should not be too common – not helpful for clustering
    - Arbitrary upper bound often placed on frequently occurring words
    - Terms should not be too infrequent, those occurring very rarely often removed – not helpful for clustering
- Helpful to take into account overall frequency of a word in the entire corpus

INVERSE DOCUMENT FREQUENCY
- Takes into account the relative number of documents in which a word occurs
    - Assumes the few documents a word appears in, the more important it is likely to be in the ones it does appear in
    - Gives a ‘boost’ to a term/word for being rare

TF-IDF WEIGHTING
- TF(t,d) = the number of times term *t* appears in document *d*
- TF = term frequency specific to one document
- IDF = inverse document frequency of a term in the entire corpus
- Terms are weighted using a combination of TF and IDF
- Rationale – give a higher rating to less common terms in the documents that contain them multiple times

COMBINING TF AND IDF
- TF and IDF are frequently multiplied to form TFIDF
- Takes into account the frequency in a given document and the relative frequency of the term in the corpus

TECHNIQUES FOR TEXT ANALYSIS
- The term document matrix for a corpus of documents can be used for:
    - Classification – classifying documents into predefined classes (DT, NB etc)
    - Clustering – finding documents with ‘similar’ content
        - K-means
        - Agglomerative hierarchical clustering