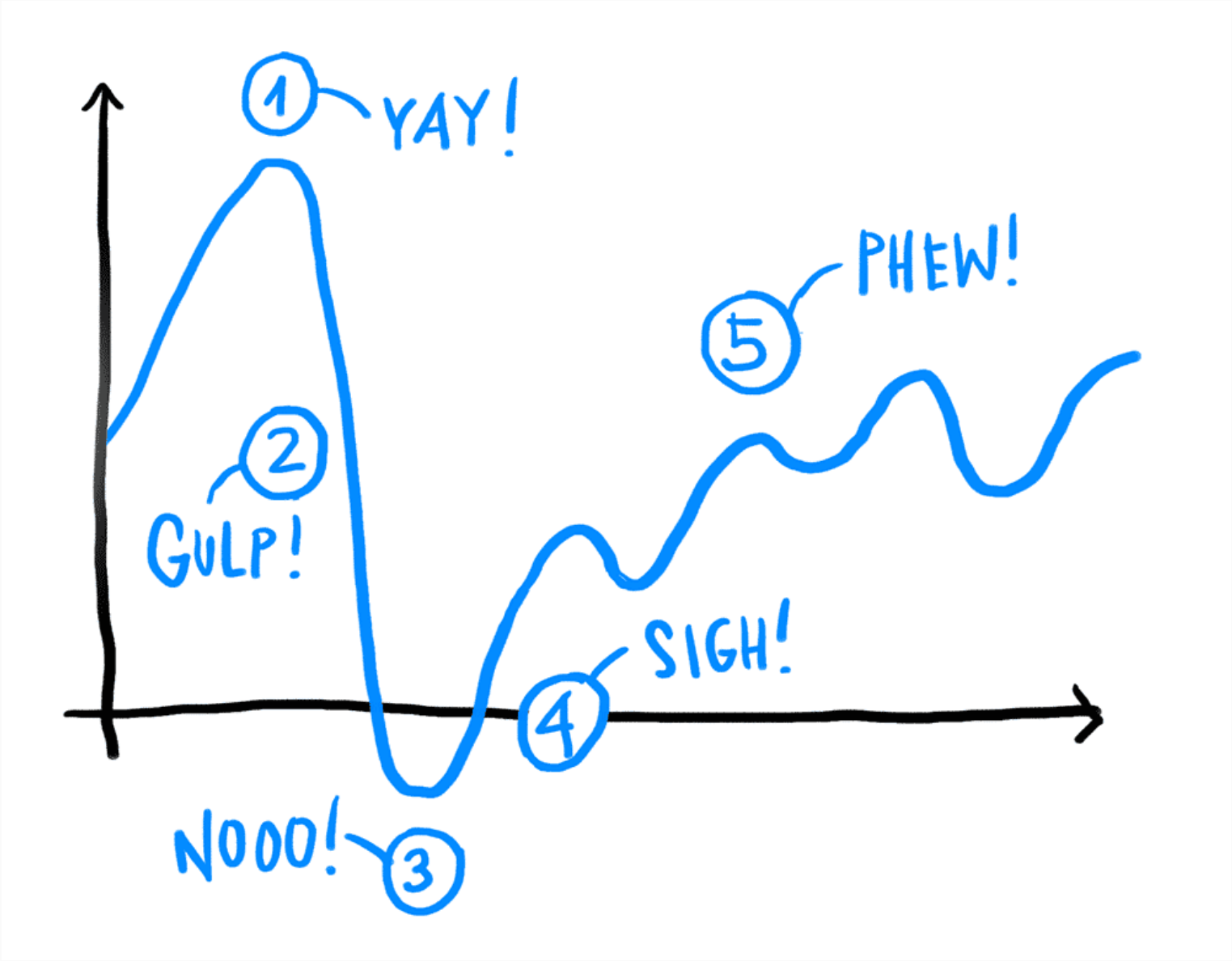
Lecture 22: Recap – Part 2

COMP90049
Introduction to Machine Learning
Semester 2, 2020

Lida Rashidi, CIS

THE UNIVERSITY OF MELBOURNE

This semester (?) – You have learnt a lot!



Source: https://www.evalotta.net/blog/2016/4/19/on-practice

Roadmap

This lecture

- Details on the exam
- Recap of part 2 of this course

Exam Details

Date, time, format...

- The exam will be on **November 17th at 3pm**
- The exam will be **2 hours (strict time limit)**, with an additional **15 minutes of reading time** and a **30 minute slack period for technical overhead** (including uploading solutions).
- The exam will be a **Canvas Quiz**
- The exam will not be invigilated, and it will be an **open book** exam.

Exam Content Details

- Worth **40% of your grade**
- A number of questions of three different categories
  - Multiple-choice Questions
  - Method Questions
  - Long Answer Questions
- You should attempt all questions (no pick-and-choose)
- Questions have different weight (!)
- **The exam is worth 120 marks**, i.e., $\approx$ 1 mark per minute. The marks associated with a question will give you an idea about how much time you should spend on it.

THE UNIVERSITY OF MELBOURNE

Decision Trees

Decision Trees: ID3 algorithm

Construct decision trees in recursive divide-and-conquer fashion

FUNCTION ID3 (Root)

IF all instances at root have same class

stop

ELSE

1. Select a new attribute to use in partitioning root node instances
2. Create a branch for each attribute value and partition up root node instances according to each value
3. Call ID3($LEAF_i$) for each leaf node $LEAF_i$

Entropy - Node Impurity

- A measure of **unpredictability**
- Given a probability distribution, the information (in bits) required to predict an event is the distribution's **entropy** or **information value**
- (The average information required to specify the outcome x when the receiver knows the distribution p)
- The entropy of a discrete random event x with possible states $1, ..n$ is:

$$H(x) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

$$\textit{where } 0 \log_2 0 =^{def} 0$$

NB: When $p(i)$ gets close to zero, entropy is close to zero.
When $p(i)$ gets close to 1, $log_2 p(i)$ gets close to zero, and entropy is close to zero.

Attribute Selection: Information Gain

- We determine which attribute R_A (with values $x_1, \ldots x_m$) best partitions the instances at a given root node R according to **information gain**:

$$IG(R_A|R) = H(R) - \sum_{i=1}^{m} w(x_i)H(x_i)$$

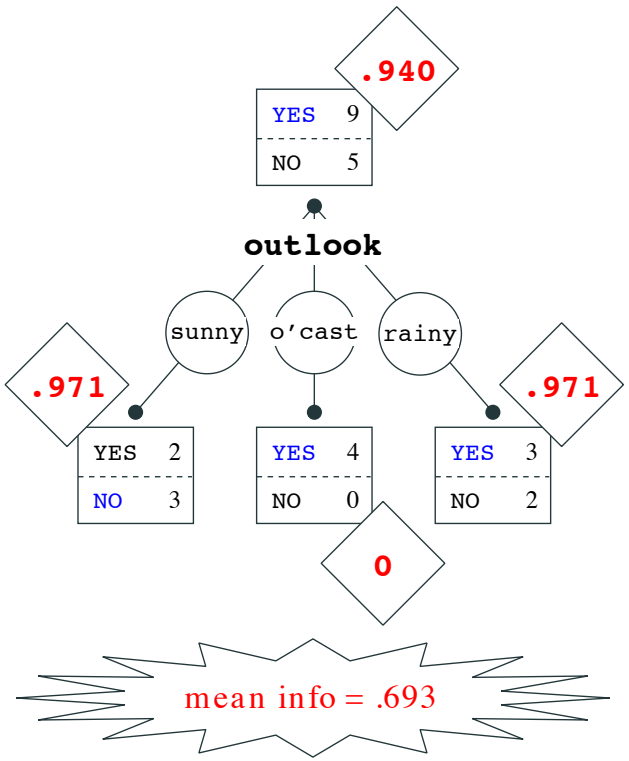
$H(rainy) =$
$-(3/5)\log_2(3/5) + (2/5\log_2(2/5))) = 0.971$

$H(o'cast) =$
$-((4/4)\log_2(4/4) + (0/4\log_2(0/4))) = 0$

$H(sunny) =$
$-((2/5)\log_2(2/5) + (3/5\log_2(3/5))) = 0.971$

$H(R) = -((9/14)\log_2(9/14) +$
$(5/14)\log_2(5/14)) = 0.940$

$Mean\_info(outlook) = P(rainy)H(rainy) +$
$P(o'cast)H(o'cast) + P(sunny)H(sunny) =$
$5/14 * 0.971 + 0 + 5/14 * 0.971$



Gain ratio

Shortcomings of Information Gain

- Information gain tends to prefer highly-branching attributes
- Subsets are more likely to be pure (above a purity threshold) if there is a large number of values
- This may result in overfitting/fragmentation

Solution: *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)

$$GR(R_A|R) = \frac{IG(R_A|R)}{\text{Split\_Info}(R_A|R)}$$

$$\text{Split\_Info}(R_A|R) = -\sum_{i=1}^{m} w(x_i) \log_2 w(x_i)$$

Feature Selection and Analysis

Feature selection: why?

Better models!

- Better performance according to some evaluation metric

Side-goal:

- Seeing important features can suggest other important features
- 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"):

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

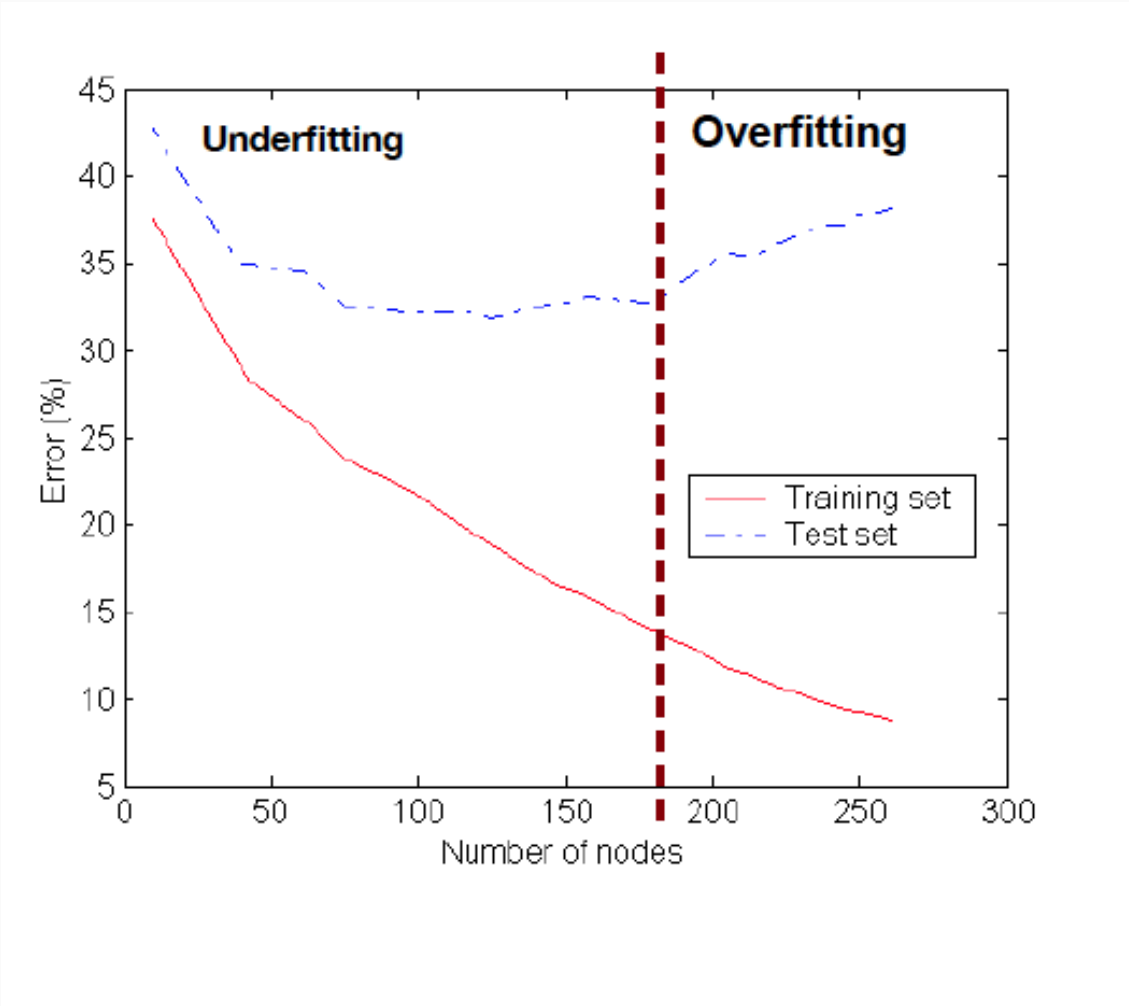
Evaluation II

Learning Curve I

- Learning curve is a plot of learning performance over experience or time
  - y-axis: performance measured by an evaluation metric such as F-score, precision, etc.
  - x-axis: different conditions, e.g. sizes of training dataset, model complexity, number of iterations etc.
- More training instances $\rightarrow$ (usually) better model
- More evaluation instances $\rightarrow$ more reliable estimate of effectiveness

Learning Curve II

- **Underfitting**: when model is too simple $\rightarrow$ both training and test errors are large
- **Overtting**: when model is too complex $\rightarrow$ training error is small and test error is large



Generalization Error

- The generalization error can be decomposed to:

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

- Or simply written as:

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- **Variance**: Captures how much your model changes if you train on a different training set. How ”over-specialized” is your classifier to a particular training set?
- **Bias**: What is the inherent error that you obtain from your model even with infinite training data? This is due to your model being ”biased” to a particular kind of solution. In other words, bias is inherent to your model.
- **Noise**: This error measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.

High Bias and High Variance Remedy

- High Bias Remedies
  - Use more complex model (e.g. use nonlinear models)
  - Add features
  - Boosting
- High Variance Remedy
  - Add more training data
  - Reduce features
  - Reduce model complexity – complex models are prone to high variance
  - Bagging

Evaluation Bias and Variance

How do we control bias and variance in evaluation?

- Holdout partition size
  - More training data: less model variance, more evaluation variance
  - Less training (more test) data: more model variance, less evaluation variance
- Repeated random subsampling and K-fold Cross-Validation
  - Less variance than Holdout
- Stratification: less model and evaluation bias
- Leave-one-out Cross-Validation
  - No sampling bias, lowest bias/variance in general

Ensemble Learning

Ensembles

- **Ensemble learning (aka. Classifier combination)**: constructs a set of base classifiers from a given set of training data and aggregates the outputs into a single meta-classifier
- **Intuition 1**: the combination of lots of weak classifiers can be at least as good as one strong classifier
- **Intuition 2**: the combination of a selection of strong classifiers is (usually) at least as good as the best of the base classifiers
- When does ensemble learning work?
  - **Diversity**: the base classifiers should not make the same mistakes
  - **Meaningful**: the base classifiers are reasonably accurate

Stacking I

- **Intuition**: smooth errors over a range of algorithms with different biases
- **Simple Voting**: generate multiple training datasets through different feature subsets, and train a base classifier over each dataset
  - presupposes the classifiers have equal performance
- **Meta Classification**: train a classifier over the outputs of the base classifiers
  - train using nested cross validation to reduce bias

Stacking II

- Given training dataset (X, y):
  - Train Neural Network
  - Train Naive Bayes
  - Train Decision Tree
- Discard (or keep) X, add new attributes for each instance:
  - predictions (labels) of the classifiers above
  - other data as available (NB scores etc.)
- Train meta-classifier (usually Logistic Regression)

Bagging I

- **Intuition**: the more data, the better the performance (lower the variance), so how can we get ever more data out of a fixed training dataset?
- **Method**: construct novel datasets through a combination of random sampling and replacement
  - Randomly sample the original dataset N times, with replacement (bootstrap)
  - Thus, we get a new dataset of the same size, where any individual instance is absent with probability $(1 - \frac{1}{N})^N$
  - construct k random datasets for k base classifiers, and arrive at prediction via voting

Random Forest

A Random Forest is:

- An ensemble of Random Trees (many trees = forest)
- Each tree is built using a different Bagged training dataset
- As with Bagging the combined classification is via voting
- The idea behind them is to minimise overall model variance, without introducing (combined) model bias

Boosting I

- Intuition: tune base classiers to focus on the hard to classify instances
- Approach: iteratively change the distribution and weights of training instances to reflect the performance of the classier on the previous iteration
  - start with each training instance having a probability of $\frac{1}{N}$ being included in the sample
  - over T iterations, train a classier and update the weight of each instance according to whether it is correctly classied
  - combine the base classiers via weighted voting

Semi-supervised and Active Learning

Semi-supervised learning I

- Semi-supervised learning is learning from both labelled and unlabelled data
- **Semi-supervised classification**:
  - L is the set of labelled training instances $\{x_i, y_i\}_{i=1}^{l}$
  - U is the set of unlabelled training instances $\{x_i\}_{i=l+1}^{l+u}$
  - Often $u \gg l$
  - Goal: learn a better classifier from $L \cup U$ than is possible from L alone

Self-Training

- Assume you have $L = \{x_i, y_i\}_{i=1}^{l}$ labelled and $U = \{x_i\}_{i=l+1}^{l+u}$ unlabelled training instances
- Repeat
  - Train a model f on L using any supervised learning method
  - Apply f to predict the labels on each instance in U
  - Identify a subset U' of U with "high confidence" labels
  - Remove U' from U and add it to L with the classifier predictions as the "ground-truth" labels ($U \leftarrow U \backslash U'$ and $L \leftarrow L \cup U$)
  - Until L does not change

Self-Training Assumptions

- Propagating labels requires some assumptions about the distribution of labels over instances:
- Classification errors are propagated

Active Learning I

- Active learning builds off the hypothesis that a classifier can achieve higher accuracy with fewer training instances if it is allowed to have some say in the selection of the training instances
- The underlying assumption is that labelling is a finite resource, which should be expended in a way which optimises machine learning effectiveness
- Active learners pose **queries** (unlabelled instances) for labelling by an **oracle** (e.g. a human annotator)

Query Strategies I

- One simple strategy: query instances where the classifier is **least confident** of the classification

$$x = \underset{x}{argmax}(1 - P_\theta(\hat{y}|x))$$

$$\textit{where} \qquad \hat{y} = \underset{y}{argmax}(P_\theta(y|x))$$

- Alternatively, **margin sampling** selects queries where the classifier is least able to distinguish between two categories, e.g.:

$$x = \underset{x}{argmin}(P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x))$$

- where $\hat{y}_1$ and $\hat{y}_2$ are the first- and second-most-probable labels for x
- Use **entropy** as an uncertainty measure to utilize all the possible class probabilities:

$$x = \underset{x}{argmax} - \sum_i P_\theta(\hat{y}_i|x) log_2 P_\theta(\hat{y}_i|x)$$

Query Strategies II

- A more complex strategy, if you have multiple classifiers: **query by committee (QBC)**
- Train multiple classifiers on a labelled dataset, use each to predict on unlabelled data, and select instances with the highest disagreement between classifiers
- Assumes that all the classifiers learn something different, so can provide different information
- Disagreement can be measured by entropy

Data Augmentation

- There are various ways to expand a labelled training dataset
- General: re-sampling methods
- Dataset-specific: add artificial variation to each instance, without changing ground truth label

Anomaly Detection

Different types of anomalies

- **Global Anomaly**: significantly deviates from the rest of the data
- **Contextual Anomaly**: significantly deviates based on a selected context
- **Collective Anomalies**: A subset of data objects that collectively deviate significantly from the whole data set, even if the individual data objects may not be anomalies
- **Unsupervised Anomaly Detection**
  - **Statistical**: assume that the normal data follow some statistical model (a stochastic model)
  - **Proximity-based**: An object is an outlier if the nearest neighbors of the object are far away
  - **Density-based**: Outliers are objects in regions of low density
  - **Clustering-based**: Normal data belong to large and dense clusters

Statistical Approaches

Univariate Data

- Assumption: normal distribution
- **Boxplot test**: Outlier is an object that lies outside $\mu \pm 3\sigma$
- **Grubb's test**: Outlier is an object that:

$$z = \frac{|x - \hat{x}|}{s} \qquad z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\frac{\alpha}{(2N)},N-2}}{N-2+t^2_{\frac{\alpha}{(2N)},N-2}}}$$

Multi-variate Data

- Assumption: Multivariate Gaussian distribution
- Outlier defined by Grubbs test on Mahalanobis distances

Statistical Approaches

Likelihood Approach

- Assumption: dataset D contains samples from a mixture of two probability distributions:

$$D = (1 - \lambda)M + \lambda A$$

Shortcomings of Statistical Approaches:

- Data may be hard to model parametrically.
  - multiple modes
  - variable density
- In high dimensions, data may be insufficient to estimate true distribution.

Proximity-based Approaches

- Data points for which there are fewer than p neighboring points within a distance d
- The top n data points whose distance to the kth nearest neighbor is greatest
- The top n data points whose average distance to the k nearest neighbors is greatest

Shortcomings:

- $O(n^2)$ complexity.
- Score sensitive to choice of k.
- Does not work well if data has widely variable density.

Density-based Approaches

- Outliers are objects in regions of low density
- Outlier score is inverse of density around object.

$$Density(x,k) = (\frac{1}{k} \sum_{y \in N(x,k)} distance(x,y))^{-1}$$

- Dealing with Variable Densities using relative density outlier score (Local Outlier Factor, LOF):

$$relativedensity(x,k) = \frac{denisty(x,k)}{\frac{1}{k}\sum_{y \in N(x,k)} denisty(y,k)}$$

Shortcomings of Density-based approaches:

- $O(n^2)$ complexity
- Must choose parameters k for nearest neighbour and d for distance threshold

Clustering-based Approaches

- Outliers are objects that do not belong strongly to any cluster.
- Assess degree to which object belongs to any cluster.

$$\frac{distance(x, centroid_c)}{median(\{\forall_{x' \in c} distnace(x', centroid_c)\})}$$

Shortcomings of Clustering-based approaches:

- Requires thresholds for minimum size and distance.
- Sensitive to number of clusters chosen.
- Hard to associate outlier score with objects.
- Outliers may affect initial formation of clusters.

Unsupervised Learning

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

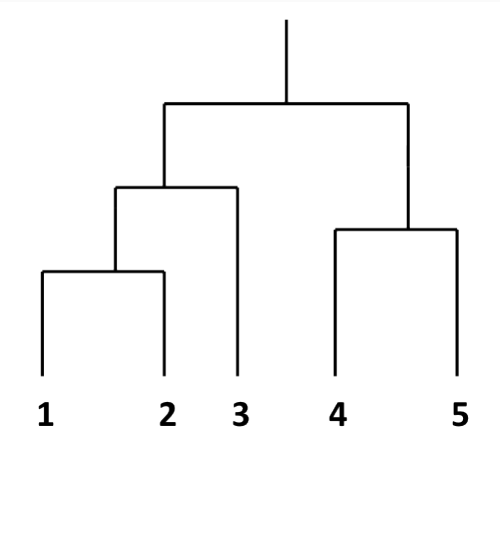
Updating the proximity matrix:

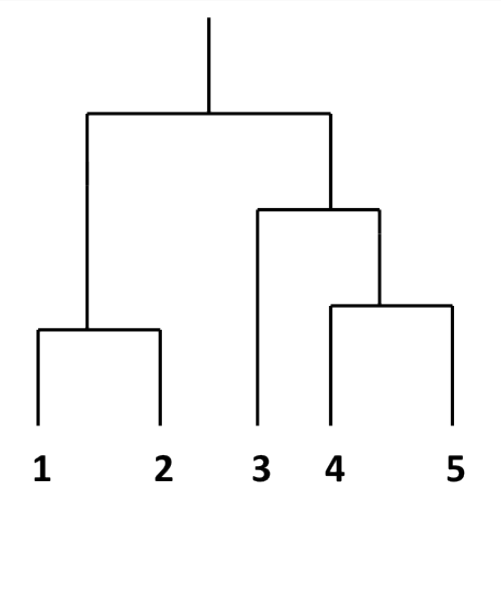
- Single Link: *Minimum* distance between any two points in the two clusters. (most similar members)
- Complete Link: *Maximum* distance between any two points in the two clusters. (most dissimilar members)
- Group Average: *Average* distance between all points (pairwise).

- 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

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

[1]

[1] A **Frequent Itemset** has a support greater than a given minsup support threshold.

Generating candidate item-sets and rules

Mine association rules which satisfy the following constraints:

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

Due to the *High Computational Cost* of brute-force strategy:

- **Apriori Principle**: Reduce he number of candidate item-sets and rules
- **Hash Tree**: Reduce the number of comparisons
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets
  - Hash Function
  - Maximum leaf size

Apriori Principle

Apriori Principle: If an itemset is frequent, then all of its subsets must also be frequent

- Candidate item-set pruning: anti-monotone property of support
  $\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$
- Rule pruning: confidence of rules generated from the same itemset has an anti-monotone property
  $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

Recommendation Systems

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

Summary

Source https://www.aitrends.com/machine-learning/here-are-six-machine-learning-success-stories/

And finally...

Please participate in the university feedback survey!

- What worked well?
- Suggestions for improvements?

Capstone / PhDs

I am looking for motivated master (capstone) and PhD students, interested in working on Graph-theoretical as well as Information Retrieval Problems. Feel free to get in touch if you're interested!