# Preprocessing

## Missing Data

If data is missing:

* Delete/Ignore the instance that is missing
  + Problem is that if large portions of the dataset are missing values then the dataset becomes much smaller
* Impute all missing instances with a constant value
  + Problem if not done correctly you may create a new class
* Attribute mean
  + Problem may change relationships between attributes adding bias to the data
* Attribute means of class
  + Problem may change relationships between attributes adding bias to the data
* Most probable value: Impute values using inference-based methods such as decision trees or KNN to predict the class labels of missing instances
  + Problems are if you do not have enough data then the inference methods will not correctly predict the missing classes

## Noisy Data

Noise: random error in a variable

If the data is noisy:

* Binning – smooth noisy attributes
  + First sort data and then partition into bins
  + Then smooth bins by calculating mean/median or each bin – creates the new value
  + Problem:
    - Lose a bit of variance
    - Could pull you in the wrong direction if you have too much noisy data in a bin
* Regression – smooth noisy attributes
  + Smooth by fitting data into regressed lines
    - Use the expected value (predicted value) to substitute for extreme values
    - Use the expected value (predicted value) to substitute for extreme values
  + Problem:
    - Can overfit towards the data (as using the fit line)
    - Any extremes may pull the regression line causing bias
* Clustering – Find groups of data that are similar to one another. Noise will be outliers.
  + Detect and remove outliers

## Data Reduction

Data Reduction: Obtain a compact representation of the data that produces similar/same results when modelling. Reduction will remove attributes that do not provide any additional information and transform the attributes to a reduced representation.

Dimensionality reduction (remove unimportant features):

* PCA (Principal Component Analysis)
* Feature selection/creation strategies

**Curse of Dimensionality**As dimensionality increases, the data becomes sparser. This is a huge issue as the density/distance between points becomes less meaningful and so classifiers/regressor (like clustering algorithms) will not perform as well.

Dimensionality reduction:

* Avoid curse of dimensionality
* Help eliminate irrelevant features and reduce noise
* Reduce time and space requirements of algorithm
* *Principal Component Analysis (PCA)*

PCA is a dimensionality reduction technique. It only works on numeric data.  
New feature space each attribute is a linear combination of the original dataset features. This is how the feature space is reduced in dimensions.

* *Feature Selection*

Chi-squared test (correlation test) can be used to determine if two attributes are related to one another. If two attributes are related, then you only need to keep one of them. Only for categorical/factor data. For numerical data use Pearson’s correlation coefficient.

If two attributes that are highly correlated, then both attributes are not needed as they contain the same information. Note keep the attribute (out of the correlated attributes) that is highly correlated to the target.  
Keep attributes that are uncorrelated  
Additionally, calculate correlation to class and throw away attributes that are uncorrelated to the class

*Search for features: Heuristic search and Relief*

## Data Transformations

Map the entire set of values of a given attribute to a new set of values.

* Normalization: Scale attribute values to a smaller range
  + Min max normalization
    - Scale the values within some pre-set min max values
  + Z-score normalization
    - Zero mean the attribute values and scale to unit variance
  + Normalization by decimal scaling
    - Remove decimal points

## Imbalanced Data

Imbalanced data refers to an imbalanced class distribution. For example if there are many more positive examples than negative examples in the dataset.

There are a few problems with this:

1. Accuracy as a metric no longer works correctly
   1. To fix this
      1. Precision and recall metrics
      2. ROC curve and AUC-ROC
2. Classifiers tend to over-predict one class (the majority class)
   1. Under and oversample the data to improve prediction of the minority class
      1. Under-sample: randomly remove instances from the majority class
         1. Adv: balances out the dataset
         2. Dis: discarded observations could contain important information
         3. Dis: Can introduce bias
      2. Over-sample: random oversample the minority class
         1. Adv: balances the dataset
         2. Adv: no information loss
         3. Dis: Risk of overfitting
   2. Cluster-based oversampling
      1. Cluster positive and negative instances independently and then apply over/under sampling to each cluster
      2. Adv: we do not morph the distribution of the training set too badly which may happen with the under/oversampling techniques above.
   3. SMOTE
      1. Create new artificial instances
      2. SMOTE finds neighbours of the minority class that are close to one another and randomly generates data points between them
      3. Can apply until you have a good distribution of the minority class

# Instance-based Learning and SVM

Instance-based learning algorithms are a class of algorithms that do not learn a function but rather directly compare to known examples.   
The key assumption of instance-based learning algorithms – similar instances will have the same classification

Note: Instance based learning is very sensitive to the curse of dimensionality – very sensitive on too many attributes.

## K-Nearest Neighbour (kNN)

When kNN gets a new instance, we calculate the distances between the new instances and the latest example. Then we find the closest neighbours (smallest distances) and use their classes to classify the new instance.

kNN very sensitive to curse of dimensionality as distance measure becomes more meaningless as the data becomes sparse.

* Dimensions get too large and classifier decreases in performance
* Greater sparsity means the number of attributes increases and thus the distances tend towards zero. As a result, distances will all have the same effect.

Redundant/Irrelevant attributes

* Pull the learner in the wrong direction
* Learner will overfit towards redundant attributes

kNN really needs reliable data, as noise (misclassification of labels) will throw off the result. To resolve this using a *high K* will reduce the effect of noise.

## Instance Based Learning

In instance-based learning prediction is based on looking at examples in the training set.

Advantages:

* Training time is very fast

Disadvantages:

* Very sensitive to noise, thus preprocessing is required
* Computationally space expensive: have to save all training instances
* Sensitive to irrelevant attributes (curse of dimensionality)
* Performance is highly dependent on the distance measure

## Support Vector Machine (SVM)

Note SVM: only considers support vectors and not all examples. Classify based on position relative to margin between positive and negative examples.

*Binary Classification SVM:*Basic idea – find a line that splits the space into two sections. Each section contains each class. A new instance would be classified by determining what section, determining what class its in.

1. Find the support vectors
   1. The points closest to the line when it is shifted. We only examine these points not all the points.
2. Find the margin
   1. Distance between the positive and negative support vectors
3. SVM will then create a maximum margin hyperplane
   1. Find the maximum margin possible
   2. Maximising the margin will give the best line to split the data into classes
4. Once the maximum margin hyperplane is created you can easily classify positive and negative instances

*Complexity Parameter*Noise (misclassifications) and complexities in the data will mean we cannot find a clean line to separate to the two classes. To resolve this issue, we would add the complexity parameter (c)

This parameter allows for noise and fuzziness. The performance of the SVM depends heavily on this parameter.

*Non-linear class boundaries - Kernels*The Kernel trick: Change the shape of the data to make it separable. This is done by transforming the data to be higher dimension. The support vectors can fit, and maximum margin hyperplane can be found now as the data is separable.

Note that we don't change the dataset itself. We change the dot product and thus shape of hyperplane. This way we virtually change the shape of the data to something else. Using kernels in SVM’s we transform the problem into a higher dimension, but the data stays in the same dimension.

Kernels can contain expert domain knowledge to help with a problem. Three common kernels; Linear kernel, Polynomial kernel and RBF kernel.

Note that: parameters in SVM’s require a lot of tuning. Once they are tuned, they typically work better than most of other learning algorithms.

*Support Vector Regression*

Ignore the errors in the margin tube and around the hyperplane.   
Note the kernel trick will still work if the data is non-linear.

*Multi-class SVM*

* Naturally SVM does not support multi-class classification as it is not easy to change the definition of the hyperplane to support this.
* Create a binary class problem from the multi-class and iterate through the classes
* However, there are two tricks to support multi-class
  + 1 vs rest
    - For example, Tennis, Squash and Badminton
      * Tennis vs not tennis
      * Squash vs not squash
      * Badminton vs not badminton
    - Have multiple predictions and combine
  + 1 vs 1
    - For example, Tennis, Squash and Badminton
      * Tennis vs Squash
      * Tennis vs Badminton
      * Badminton vs Squash
    - Have multiple predictions and combine

# Bayesian Learning

Bayesian Learning applies the Bayes Theorem (conditional probability) to search for the most likely/most probable hypothesis.

Bayesian Learning is relevant:

1. Explicit manipulation of probabilities to solve learning problems
2. Framework for understanding learning methods that do not explicitly manipulate probabilities

Disadvantages to Bayesian Learning:

1. Need initial knowledge of probabilities
   * We can proceed without this and specify that any candidate hypothesis is equally likely (choose uniform distribution)
2. Significant computational costs
   * When we have a lot of examples
   * However not a major problem as we can parallelize the algorithms

Machine learning finds the best hypothesis *h* from some space of hypotheses *H* given some training data *D.* The best hypothesis can be defined as the most probable.

## Naïve Bayes Classifier

Inductive Bias of Naïve Bayes: Attributes are conditionally independent. This makes it Naïve.

For example: Using Bayes Theorem in Naïve Bayes:

Where c = class and d = new datapoint   
 = Posterior probability: What is the probability of this class contains this datapoint

= likelihood: probability that this datapoint is observed in this class

= prior probabilities: P(c) – probability of this class in the dataset, P(d) – probability of this datapoint being observed in the data

Then to **classify** the datapoint into a class pick the class with the highest posterior probability (MAP – maximum A Posterori hypothesis)

where d = x1,x2,x3….xn where x1 to xn are the features

*Laplace Smoothing* – used in the likelihood

For text classification problems the likelihoods can be represented by counts. If the denominator of the equation is 0 the equation is undefined. Laplace smoothing adds a constant to denominator and numerator which doesn’t make much of a different in the long run of trying to maximise the posterior probability.

## Bayesian Networks

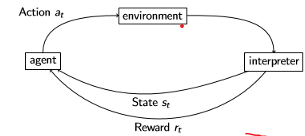
Naïve Bayes assumption of conditional independence is very strict and restrictive. Bayesian Belief networks relax this assumption: they describe conditional independence among subsets of variables.

A Bayesian network is a directed acyclic graph with nodes (one per attribute) and directed edges (one attribute influences another)

# Reinforcement Learning

Reinforcement learning can be used in problems involving an agent interacting with an environment which provides rewards.

The goal of reinforcement learning is for an agent to learn a policy (how to take actions) that maximises its rewards



* Agent interacts with the environment
* Interpreter reads from the environment
* Interpret describes the environment with a certain state
* Based on the state the agent chooses an action
* Agent gets a reward for the action

## Markov Decision Process

MDP is the mathematical basis of RL:

* Uses the Markov property where the future state is solely based on the current state and action
* MDP is defined by:
  + S = set of possible state
  + A = set of possible actions
  + R = distribution of reward given based on an action state pair
  + P = transition probability
  + = discount factor – weighing impact of future rewards

The goal of RL mathematically in MDP: find optimal policy π\* that maximises the expected reward from this action and subsequent future actions.   
Gamma reduces the impact of future rewards. It reduces future rewards even stronger than more recent ones (more recent rewards are weighed higher).

## Value Function

Overcomes the problem (like in chess, go and connect 4) where we do not know the value of intermediate moves during play as we only really know at the end if the intermediate moves you made were successful.

So, we define a value function for each state in a policy:

* Expected reward of all the future steps that will be performed
* Sum of the expected rewards times discount factor

## Q-Learning

*Value Iteration Algorithm*Value Iteration algorithm: Algorithm that learns the optimal value function if we know

probability of going into the future state given we are in a particular state and an action is applied.

Note that: This assumes the transitional state probability is known which we never know! This is why we do **Q-Learning**.

*Q-Learning: an algorithm to optimise π\* by iteratively simulating the Markov decision process and updating the Q matrix.*

*Q-Learning for Deterministic Worlds*

* Define a new function Q that is closely related to the value function. Q is defined over some state and action while V is only defined on a state.
* Choose some action (make some decision) and compute its expected value of reward
  + Then we apply the optimal policy for the next future states
* Agent then can work out Q values for each state action pair

We can choose optimal policy and actions without knowing the transition probability.

Note Q and V are closely related: We can get optimal value function if we maximise Q function for a given state across all actions.

Furthermore, the learning must approximate Q (Q-hat).   
Estimated Q value for a given action state pair = immediate reward + estimate of future rewards

Q-hat will converge to Q under a set of conditions:  
1. System is a deterministic MDP  
2. Immediate reward values are bounded  
3. Agents select actions such that every action state pair is visited infinitely often - Agent has to select an action such that it will be able to visit it again in the future

*Q-Learning for Non-deterministic Worlds*Q-learning training rule can be extended to cover non-deterministic worlds.

## Generalizing from examples

To make Q-learning more practical (as we do not know all possible states because new states could be added on the fly) for unseen state-action pairs we can use supervised learners.

Replace the Q matrix with a supervised learner that predicts the Q-estimates.

However, there are some **issues** with using a supervised learner (ANN for example):

If the learner (ANN in this case) updates the network to better fit the training q-values for a particular transition, then the network weights are updated for all other transitions too

* 1. This weight change may increase error in q-estimate predictions in other transitions
  2. Convergence no longer holds
  3. This is the difficulty developing supervised learning models (using NN) for RL

# ARM: Association Rule Mining

ARM is a problem where given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Note that implication means correlation not causality.

## Apriori Algorithm

Apriori principle: If an itemset if frequent, all of its subsets must also be frequent.

The Apriori algorithm uses a generate-and-test approach. The algorithm generates candidate itemsets and tests if they are frequent

Apriori generates a large amount of output so we must filter that down by looking at closed and maximal itemsets.

Apriori disadvantages:

* Generation of candidate itemsets is expensive
* Support counting is expensive
* Subset checking is expensive
* Multiple database scans

Note that the output of Apriori are frequent itemsets.

## Rule generation

We use interestingness measures to prune/rank the patterns. Prune redundant rules and rules that are not interesting.

Confidence has a drawback: misleading rules can be produced. Confidence is not a good measure to evaluate a rule.

Therefore, use **lift**. Lift will tell you if rules are positive or negatively correlated.   
Lift values typically go from 0 to Inf. Greater the lift, more highly correlated the variables are.

However, lift also has its own issues. The smaller the support the higher the lift. Realistically when we talk about lift:

* It depends on the distribution of data
* Smaller the value of P(x) and P(y) the higher the lift

## FP-Tree and FP-Growth

*FP-Tree*

* Compressed/compact representation of the dataset without any information loss
* Easy to transverse and thus quickly find patterns associated with a certain item
* Tree is ordered by item frequency

After we build the tree, we pass it to *FP-Growth algorithm*FP-Growth: divide and conquer mining technique

* Construct a conditional FP tree
* Concatenate patterns from the conditional subtree

FP-Tree and FP-Growth does not follow a generate and test approach - it generates the itemsets by itself

# Clustering

Cluster Analysis: find groups of datapoints such that observations in a group will be similar to one another and different from the datapoints in other groups.

* Inter-cluster distances are maximised
* Intra-cluster distances are minimised

*Partitional Clustering*

Separates datapoints into non-overlapping subsets (clusters) such that each datapoint is in exactly one cluster

*Hierarchical Clustering*

A set of nested clusters organised into a hierarchical tree.

## K-Means Clustering – Basic Algorithm

Partitional Clustering approach

* Select K points as the initial centroids. Initial centroids are chosen randomly and thus K-means is not-deterministic.
* Form K clusters by assigning points to the closest centroid (calculate distance using distance measure)
* Recompute the centroid of each cluster
* Repeat the steps above until the centroids do not shift OR only a very few amounts of points change clusters

The centroid is typically the mean points of the cluster.

*Evaluate K-means clustering*

We want well-defined and well-separated clusters. To assess these clusters, we use sum of squared error (SSE). We want to minimize SSE over all points in all clusters.

To reduce SEE increase K (increase the number of clusters) – however it’s a tradeoff: a good clustering with low K can have lower SSE than higher K.

## Select good initial centroids

Selecting good initial centroids is key (it is hard to select one centroid from each cluster during initialization) if you don’t select good initial centroids the clustering can turn out to be horrible.

Ways to select good initial centroids:

1. Multiple runs
2. Sample and use hierarchical clustering to determine initial centroids
3. Select more than k initial centroids and then select among these centroids
   1. Select the most widely separated
4. Bisecting K-means
   1. Uses BIC as an optimization function to separate out clusters and stops when you reach BIC score threshold
   2. Not as suspectable to initialization issues

## Limitations to K-means

K-means have problems when the clusters are:

* Differing sizes
* Differing densities
* Non-globular shapes

K-means also has problems with noisy data and data with outliers.

## Hierarchical Clustering

Strengths of hierarchical clustering:

1. Do not have to assume any particular number of clusters like in K-means
2. May produce meaningful taxonomies – schemes of classification

## Agglomerative Clustering

* Start with the points as individual clusters
* At each step, merge the closest pair of clusters until there is only one/k clusters left

Distance/Similarity operations (inter cluster distances)

* MIN
  + Adv: Non-globular shapes
  + Dis: Sensitive to noise and outliers
* MAX
  + Adv: Less sensitive to noise and outliers
  + Dis: Tends to break up large clusters
  + Dis: Biased towards globular clusters (will break up non-circular and irregular shaped clusters)
* Group Average – compromise between MAX and MIN
  + Adv: Less sensitive to noise and outliers
  + Dis: Biased towards globular clusters (will break up non-circular and irregular shaped clusters)
* Ward’s Method
  + Adv: Less sensitive to noise and outliers
  + Dis: Biased towards globular clusters (will break up non-circular and irregular shaped clusters)

*Hierarchical Clustering*

* Issue: Once a decision to combine two clusters has been made, it cannot be undone
* Different merging schemes have problems (listed above)
* No objective function is directly minimized

## DBSCAN - Density Based Spatial Clustering for Applications with Noise

DBSCAN is a density-based algorithm.

Defines 3 types of points – core point, border point and noise point.

DBSCAN strengths:

* Resistant to noise
* Can handle clusters of different shapes and sizes

DBSCAN issues:

* Varying densities
* High-dimensional data

# Data Streams

A data stream is a sequence of information that is in the process of being transmitted. Data stream is a real-time, continuous, ordered sequence of items.

* Continuous and sequential information
* Unpredictable input rate
* Large amounts of data
* Not error free

*Problems when dealing with data streams:*

* Data in a stream is large and fast therefore getting information from it in real time is an issue. We thus accept approximate solutions.
* Stream may be evolving and thus model have to adapt when there is a change in the distribution of the incoming data.

Requirements for stream mining algorithms:

1. Be able to process an instance at a time
2. Be able to process each instance quickly - in a short amount of time
   1. Preferably linear or log time
3. Need to use limited amount of memory to process the instances
4. Provide real time analytics/answers at any time
5. Adapt to temporal changes (concept drift). Adapt to differences in distribution of the data or content of the data.
   1. Traditional ML models cannot do this

Sampling – pick and choose instances to approximate the prediction

## Reservoir Sampling

Simple random sample without replacement of k items from a population in a single pass

Goal: maintains a fixed-size uniform random sample

1. Add k elements to the reservoir.
2. When a the t’th element arrives where t>k then adds it to the reservoir with the prob k/t
3. If you add it to the reservoir then randomly remove an element from the reservoir.

The size of the sample stays the same the entire time.

## Lossy Counting

Approximate heavy hitters and counting the frequent items in the data streams. Guarantees that heavy hitters will be kept within your data stream.

Tracks items and counts. For each block of items, merge with stored item counts. Decrement all counts by 1 and remove empty counts.

By removing the empty counts, you get rid of obsolete information. Only recent/current is kept in the stream and you the ability to keep track of what is current in the data stream.

## Merging Sketches – Exponential Histogram

Store the error rates in buckets.   
Merging buckets allows you to maintain the exponential histogram. It is a very compact representation of the stream and is good because what is happening is more weight is given to new/recent information and old information is summarized (bucket merging).

## Change/Concept drift

Change occurs when the distribution of the input data in the stream has changed. This means the learning algorithm has to adapt too.

When change occurs (monitor the error rate) the prediction error will decrease and stay consistently decreased. Monitoring the error rate allows you to make judgements on if the model is deteriorating or not.

If error rate is monitored, you can see within the bands the accuracy dropped all of a sudden. This indicates a change in the distribution of incoming data. The classifier trained on the original data no longer holds/no longer predicts well for predicting the new instances.

Types of change:

1. Shift (sudden change)
2. Gradual change
3. Incremental change
4. Reoccurring concepts
5. Outlier

*Virtual drift vs Concept drift*Virtual drift is when the features change but prob predictions remain unchanged.  
Concept drift is when the prediction prob changes without or without changes in the inputs

* Concept drift: distribution of the data itself has shifted.
  + As new information has come in the original decision boundary is no longer accurate
  + If you use the original boundary error rate of the prediction will increase
  + So instead if you learn the model again you can get a better decision boundary and lower error
* Virtual concept drift:
  + As new information comes in the distribution of features space changes but boundary does not change. No degradation of accuracy in this case.
  + No shift in the decision boundary but distribution of feature space has changed.

Deal with concept drift: Drift detectors

1. CUSUM – sequential analysis
2. DDM – statistical process control
3. ADWIN – monitor distributions
4. Contextual approaches

CUSUM  
Mean of input data significantly deviates from the previous values.

DDM  
Monitors error rate and if prediction error rate increases, this is evidence that change has occurred.   
Note that DDM first issues a warning and starts to store examples. If a change is confirmed the model is thrown out and retrained on the saved examples.

ADWIN  
Change detector and estimation algorithm based on exponential histograms.  
ADWIN we are compressing the error rate into an exponential histogram.

To figure out a shift:

* Monitor the changes between the new value and the old batches
* Look at if there is a difference between the mean of the new batch and the current batches
  + Given content and capacity you can calculate mean
* We then shift the window and look at the mean between the last 2 batches and the first two
* We then shift the window again and look at the mean between the last 3 batches and the first one
* This way we look for multiple cut/change points in the data stream

Contextual approach – ensemble of detectors   
Main strength of this technique: Capture the diversity of the different types of concept drift by using multiple drift detectors

## Error Estimation

* Holdout: Good method when data is abundant, carry out tests periodically. Learner is trained continuously and then a small batched dataset is used to calculate the accuracy.
* Interleaved test then train: Each example is used to test the model and then trained into the model. Remember to test then train to ensure you do not bias the model (keep unobserved data principle).
* Prequential: Same as interleaved test then train but more recent examples are more important (enforced using a sliding window or decaying factor)
* Interleaved Chunks: Similar to interleaved test then train but you do it chunk by chunk.

## Performance Measures

Note that prequential accuracy is only good if the classes are balanced. Use Kappa statistic and Kappa M statistic.

* Kappa: Gives the probability of a chance classifier and quantifies how much better are we than a random classifier
* Kappa M: measure that compares against a majority class classifier instead of chance classifier. Quantifies how much better are you then a majority class classifier
  + Works well with an imbalanced dataset

## Hoeffding Tree

Hoeffding Tree is an online decision tree – very fast tree algorithm for streaming data.   
Wait for data to come in and then wait to split on attributes that are deemed the best.  
Hoeffding Tree uses Hoeffding Bound (non-parametric bound - ideal as we are not inferring any distribution in the stream).

When instance comes in we are have check whether we should grow the tree or not.

1. For each instance that comes in - calculate a gain on the attributes
2. Then check whether the gain on the best attribute minus gain on second best is greater than this formula. The formula is the Hoeffding bound.
   1. Estimating the entropy at specific attributes to see whether we should be splitting on it
3. If the difference in gain > Hoeffding bound, then split

## VFDT – Very Fast Decision Tree

Extension to Hoeffding Tree where:

1. Speed improvement: instead of computing best attributes at every instance only compute it at n\_min instances
2. For ties in gain, if two attributes have the same gain split if the condition is met.

NOTE: Hoeffding Tree/VFDT do not account for concept drift.

## Hoeffding Adaptive Tree (HAT)

Hoeffding Tree that uses ADWIN (as change detector and error estimator).

HAT replaces old tree with alternate tree as soon as there is evidence of a change at a node. If there is a change detected by ADWIN then HAT will grow a subtree.

# Anomaly Detection

Key challenges in outlier detection:

* Modelling normal behaviour and outliers correctly
* Outlier detection must be tailored to the domain
* Handling noise in outlier detection
* Degrees of outliers – understandability

3 core methods for outlier detection:

1. Supervised Learning (classification problem)

* Dis: Imbalanced classes (outliers are rare)
  + Boost the outlier class and make up some artificial outliers (SMOTE)
* Dis: catch as many outliers as possible
  + Recall (ROC curves, AUC and precision) are more useful metrics than accuracy

2. Unsupervised Learning (assume normal objects can be clustered into groups having distinct features). An outlier is assumed to be far away from normal objects.

* Dis: cannot detect collective outliers effectively. Normal objects may not share any strong patterns, but the collective outliers may have high similarity in a small area.

3.Semi-supervised learning – number of labelled datapoints is small

## Statistical approaches

Model-based methods (statistical methods) assume that the normal data follow some statistical distribution. The data that does not follow the distribution are outliers.

The effectiveness of this method is reliant on whether the assumption that the normal data follow the statistical distribution holds.

Idea: Learn a generative model (statistical distribution) fitting a given dataset and identify observations in low probability regions as outliers.

## Proximity-based approaches

Idea: Objects that are far away from others are outliers.

Can be partitioned into two core areas: distance-based outlier detection and density-based outlier detection.

* Distance-based: point p is an outlier if its neighbourhood does not have enough points (i.e. if it is distant from other points)
* Density-based: point p is an outlier if its density is relatively much lower than its neighbours (i.e. points in low density regions)

Both these methods can be categorized as nearest-neighbour approaches.

*Distance-based Outlier Detection*Observation p is an outlier if most of the other datapoints are far away from p.   
Use formulae to assess the fraction of objects close to the observation and if fraction greater than threshold then its not an outlier.

*Density-based Outlier Detection*Idea: density around an outlier object is significantly different from the density around its neighbours.

*Density-based Approach - LOF Approach – Local Outlier Factor*

* For each datapoint compute density of its local neighbourhood
* Compute LOF of datapoint
* Outliers are points with the highest LOF values

LOF: lower LOF means it is less likely to be an outlier, higher LOF means it is more likely to be an outlier

## Clustering-based approaches

Can solve for 3 types of outliers:

* Points that do not belong to any clusters
  + Use DBSCAN to identify these points
* Points that are far from clusters
  + Use K-means and a distance scoring method to find these points
* Points that belong to small or sparse cluster
  + Use FindCBLOF algorithm to detect outliers in small clusters

Adv’s and Dis’s of clustering methods:

Adv:

* Detect outliers without any labelled data
* Work for many types of data
* Fast prediction (once clusters are computed figuring out if a new point is an outlier is an issue)

Dis:

* Effectiveness depends on clustering method used
* High initial computation cost to find clusters
  + Can reduce this by using fixed width clustering

## Classification Approaches

Idea: train a classification model that can distinguish normal data from outliers

One class model: build a classifier to describe the normal data.

* Train SVM to learn the decision boundary of normal class
* Any samples that do not belong to the normal class (not within the decision boundary) are outliers
* Adv: can detect new unseen outliers

Adv: outlier detection is fast

Dis: Quality of detection depends on quality and quantity of training data which is difficult to find

## Unsupervised Approaches

Isolation forest: collection of trees that partition the dataset

Each iteration a random feature is selected, and the data is split on a randomly chosen value. The process is repeated until the entire dataset is partitioned to form an individual tree in the forest.

Idea: Anomalies are easier to detect as they form much shorter paths from the root than normal data points. Thus, they are easier to isolate.

Use paths as a measure to detect/score anomalies.

## Evaluation methods – Score Anomalies

Accuracy is not a good method for evaluation of anomaly detection. This is because of the highly imbalanced classes within the data (anomalies are rare versus normal events occur so much).

Therefore, we want to be using **Precision and Recall** instead of accuracy.

Furthermore, we could be using **ROC and AUC** to see the predictive power of our anomaly detection methods.