Holy Bible

Indhold

Knowledge discovery in databases

Frequent pattern mining

Set theory

Powerset complexity: 2^n where n is the length of the set

Frequent itemsets

k-itemset: The length of an itemset

An itemset is a subset of a collection of items. They are ordered by a total ordering.

Support: The support is how often an itemset occurs in the database

Frequency: This is the support divided by the size of the database.

Frequent itemset: On itemset is frequent if its support or frequency is above a threshold.

Association rules

A rules which states, that if one thing then something else. So if we buy milk, we also buy coffee:

items bread, butter, eggs, milk etc.

transaction a set of items

rule $L \Rightarrow R, L, R \subseteq$ items, $L \cap R = \emptyset$

L *left-hand-side* or *antecedent*

R *right-hand-side* or *consequent*

- {butter, flour} $\Rightarrow$ {milk}
- {sugar} $\Rightarrow$ {butter}

IT describes co-occurences NOT causality. It does not have to hold for all cases.

Support of the association rule: The support is the union of how often the left hand side occurs and the right hand side occurs.

The frequency: is the support of the rule divided by the size of the database.

Confidence: How certain is the rule. This is the support of the full association rule divided by the support of the left hand side. (antecedent)

Association rule mining

We try to find the associations rules, where the support is above a threshold and the confidence is above a threshold.

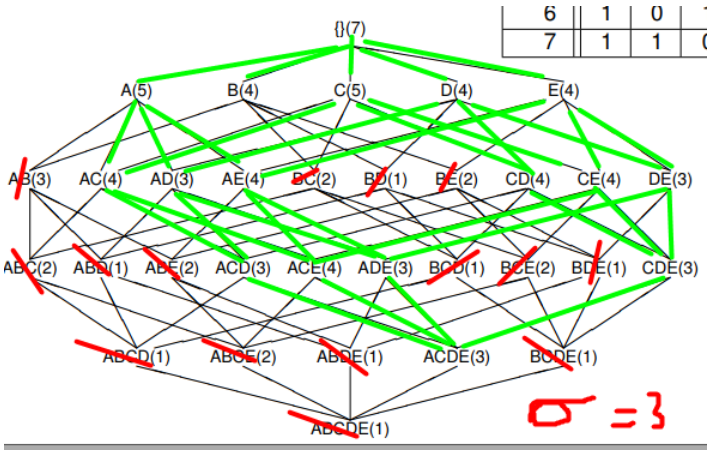
$$\{X \Rightarrow Y | s(X \Rightarrow Y) \geq \sigma \wedge \text{conf}(X \Rightarrow Y) \geq c\}$$

Brute force: Take each possible itemset from the list of items and check the database if the count is frequent: The complexity is 2^n * *length(database)* So the complexity is the 2 to the power of the number of items times the length (number of transactions) of the database.

Monotonicity and anti-monotonicity

Monotonicity: If a set is frequent is subsets will be frequent as well

Anti-monotinicity: if a set is infrequent its subsets will be infrequent as well.



The web above is called a lattice.

This is a positive border if the set cde is frequent.

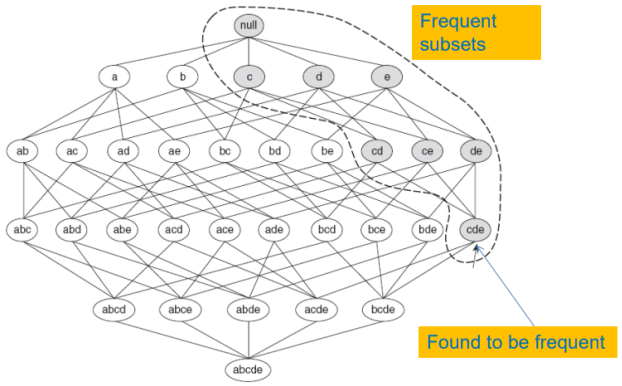
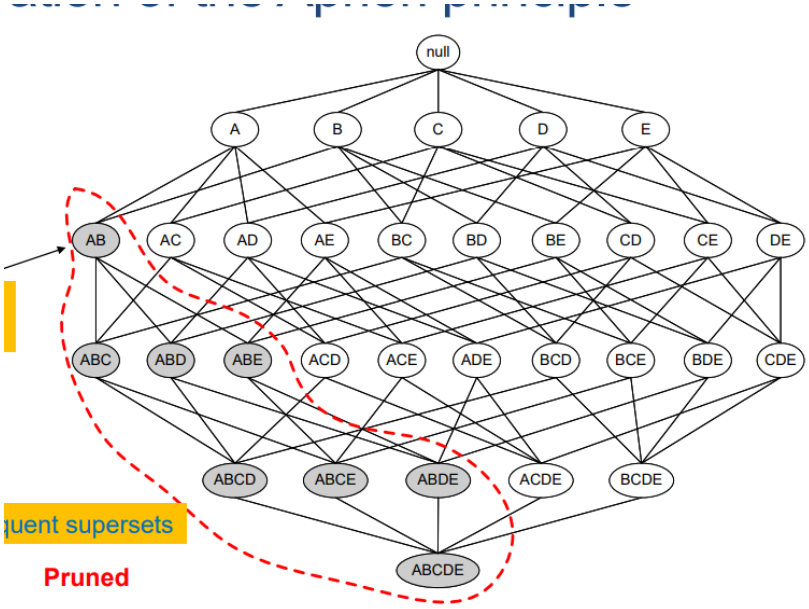


Figure 6.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

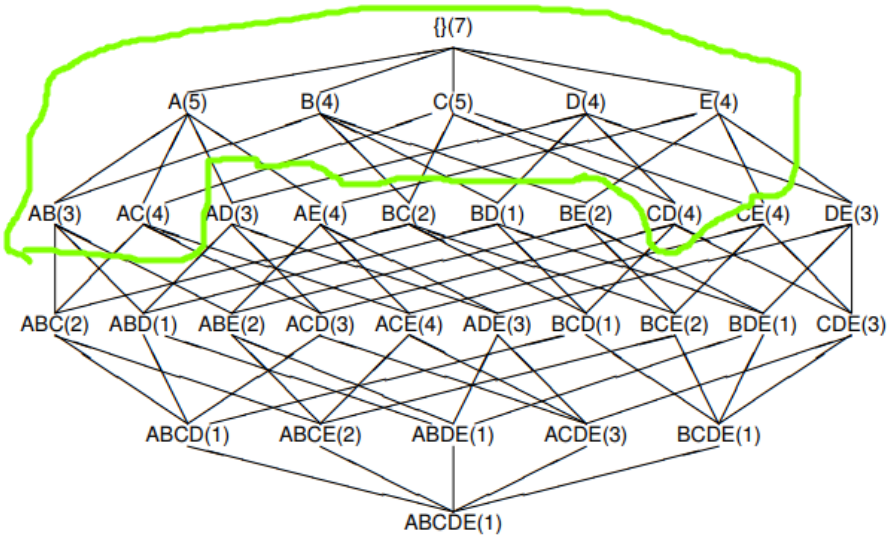
This is a negative border:



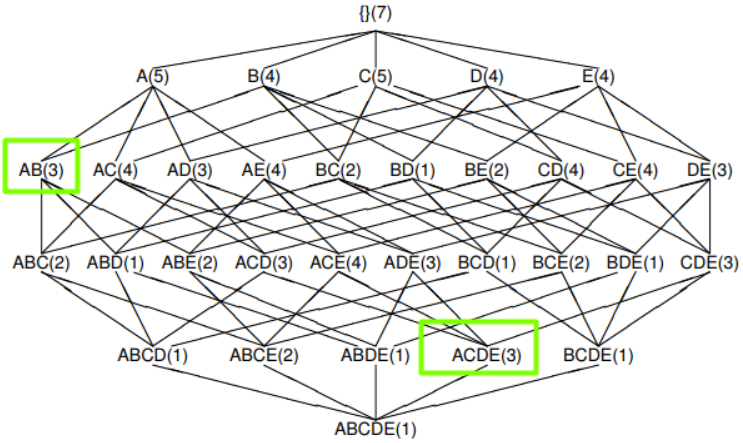
Max and closed frequent itemset

Representation	Description	Identification	Advantages
Maximal Frequent Itemset	Is a frequent itemset for which none of its immediate supersets are frequent.	1. Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets. 2. Identify all of its immediate supersets. 3. If there exist at least one superset that frequent, the itemset is not maximal frequent, otherwise if none of the immediate supersets are frequent, the itemset is maximal frequent.	They are valuable because they provide a compact representation of frequent itemsets. In actuality they form the smallest representation of frequent itemsets and so they are the most practical to use when space is an issue **Limitation:** They do not contain the support of their subsets and so an additional pass needs to be made of the data in order to find this information.
Closed Frequent Itemset	Is a frequent itemset that is a superset of the maximal frequent itemset. A closed frequent itemset is a frequent itemset that is both closed and its support is greater than or equal to minsup. An itemset X is closed in a data set if there exists no superset that has the same support as X in the data set.	1. First identify all frequent itemsets. 2. Then from this group find those that are closed by checking to see if there exists a superset that has the same support as the frequent itemset, if there is, the itemset is disqualified, but if none can be found, the itemset is closed. 3. Finally, of these itemsets that you have found to be closed, crosscheck to make sure that their support is greater than minsup (if you did step 1 right, all of your itemsets should pass this test)	They are useful in removing some redundant association rules. They are a subset of frequent itemsets and so they provide a compact representation. They include the necessary information that helps us determine the support of their subsets.

Closed frequent itemset: An itemset is closed if none of its immediate superset has the same support and if it is frequent. So here is an example of a closed itemset with support = 3:



Max frequent itemsets: Is a set where, none if its immediate supersets are frequent. Here is an example with support = 3:



These frequent itemsets do not have a frequent superset.

Apriori algorithm

This algorithm utilizes the power of monotonicity and anti-monotinicity to only search the itemsets, that will be frequent by pruning itemsets, where its subsets are infrequent.

The algorithm is pretty simple:

1. Create all singleton itemsets
2. Count how often each singleton itemset occurs
3. C_1 = Prune / remove those, that have a support lower than the threshold.
4. Create all 2-itemsets combinations of the remaining itemset
5. Return Set of items (set_1)

K = 1

While set_k is not empty

6. Join the itemsets where the first k-1 element is the same in set_k.
7. C_k = prune remove a set, if it contains a subset that is not part of set_k
8. Count how often the c_k occur in the database
9. Set_k = Prune / remove those that have a lower support in the Db than the threshold
10. K = k + 1

Complexity O(2^n) where n is the number of items

Association rules in apriori

For each frequent itemset, that is left, we build association rules and delete rules with confidence below a given threshold.

for frequent itemset X:

- for each $Y \subset X$, $Y \neq \emptyset$, build the rule $Y \Rightarrow (X \setminus Y)$
- $\text{conf}(Y \Rightarrow (X \setminus Y)) = \frac{s(X)}{s(Y)}$
- delete rules with confidence below a given threshold c

Support is useful for business, confidence measures how reliable a rule is. But there are other measrues called interestingness that can be used instead, since you can get a high confidence, let's say 25% buy coffee and tea together, this might sound interesting, but if you then consider that 80% of all people buy coffee, then it is pretty uninteresting.

	coffee	no coffee	
tea	150	50	200
no tea	650	150	800
	800	200	1000

$\{tea\} \Rightarrow \{coffee\}$ $\frac{150}{800} = 0,25$

support?

confidence?

$\{\} \Rightarrow \{coffee\}$ $\frac{800}{1000} = 0,8$

support?

confidence?

Conclusion?

(Discussed by Tan et al. [2006], page 372f., example 6.3.)

Feature spaces and distances

We can transform an object into a lower representation feature space. The goal is to only keep the relevant information.

3 types of features:

nominal (categories):
- It is possible to tell whether two values are equal or not – but no direction (better, more,...) or meaningful distance.
- Examples: sex, eye color, healthy/sick, amino acids

ordinal
- there is an order relation (e.g., better/worse), but no uniform distance
- Examples: grade, quality label, age class (e.g., 20-29, 30-39,...), color (?)

metric
- differences and relations between values are meaningful, values can be discrete or continuous
- Examples: weight, selling counts, age

We can aggregate features, such as counting their frequency, relative frequency, their median or their average (slide 112)

Centroids and medoids

A centroid is the mean of all our samples:

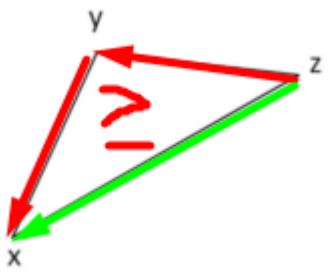
$$\mu_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \cdot \sum_{o \in \mathcal{D}} o$$

A medoid is the point which have the smallest distance to all other datapoints:

Distance

A feature space have a distance function, it must be strict (that is one is higher than the other), it must be symmetric (if x > y then y < x) and reflexive (if distance from x to x must equal 0)

A metric space is a feature space, where the triangle inequality holds, most common is the euclidian space:



Z --> y --> x is greater than y →x

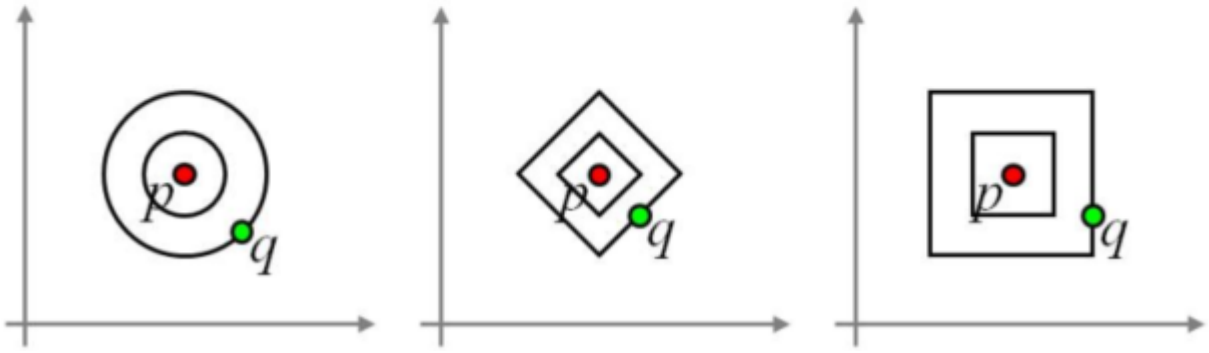
We have this formula for calculating distances:

$$\text{dist}_{\mathcal{P}}(p,q) = \left(|p_1 - q_1|^{\mathcal{P}} + |p_2 - q_2|^{\mathcal{P}} + \ldots + |p_n - q_n|^{\mathcal{P}}\right)^{\frac{1}{\mathcal{P}}}$$

Euclidean norm (L_2):

Manhattan norm (L_1):

Maximum norm (L_∞, also: $L_{\max}$, supremum dist., Chebyshev dist.)



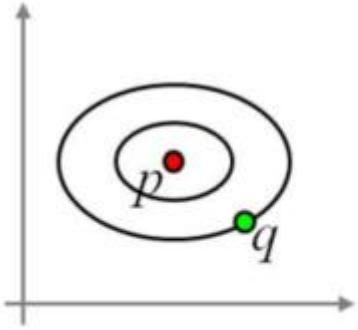
We can also have euclidian weighted distance and quadratic matrix distance:

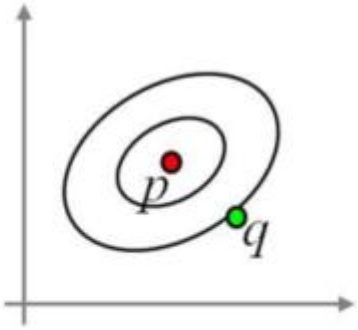
weighted Euclidean norm:

$$\text{dist}(p,q) = \left(w_1|p_1 - q_1|^2 + w_2|p_2 - q_2|^2 + \ldots + w_n|p_n - q_n|^2\right)^{\frac{1}{2}}$$

quadratic form*:

$$\text{dist}(p,q) = \left((p-q)M(p-q)^{\mathsf{T}}\right)^{\frac{1}{2}}$$





Images

With a color histogram we can create a histogram where we count the frequencies of each color. Moreover, we can reduce the number of colors, which can be useful for reducing features.

Clustering

Clustering is an unsupervised approach to categorize data points, since in the works case scenario (if we used a naïve method) we would end up having $O(k^n)$ partitions.

Partitional clustering

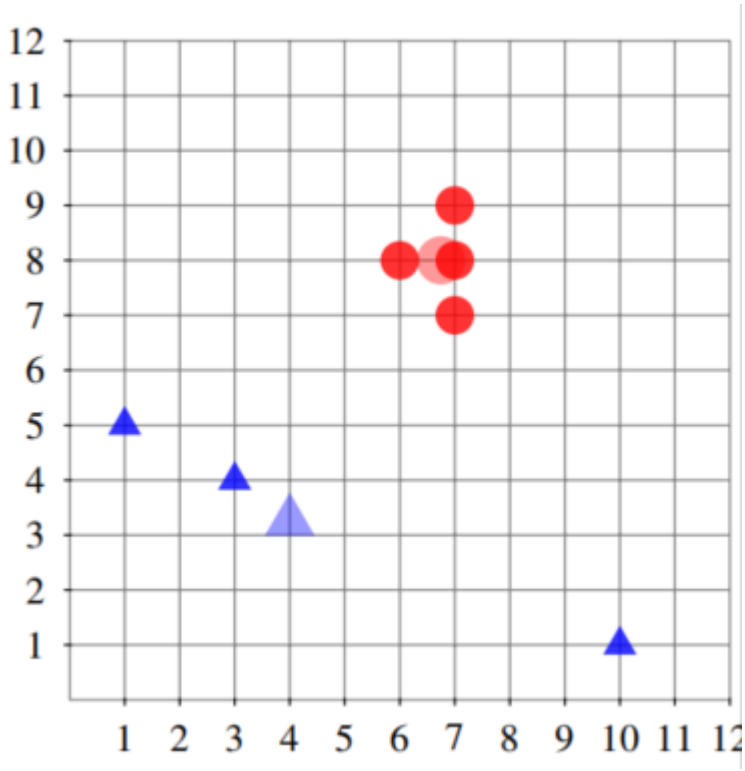
A data set is portioned into k cluster with the goal being minimizing a cost function.

Central assumptions for approaches in this family are typically:

- number k of clusters known
- clusters are characterized by their compactness
- compactness measured by some distance function (e.g., distance of all objects in a cluster from some cluster representative is minimal)
- criterion of compactness typically leads to convex or even spherically shaped clusters

This strategy requires us to chose how many k we want. We then optimize iteratively, this is done by assigning the points to the same cluster as its closests representative. We can use a centroid, medoid or a gauissian distribution model.

The centroid of a cluster is the mean of all points in the cluster



$$\mu_{C_i} = \frac{1}{|C_i|} \cdot \sum_{o \in C_i} o$$

The measurement of its compactness is done by summing up the distance from each point to its centroid in the cluster. This is called the sum or squared error, and is a loss function we should minimize

$$TD^2(C) = \sum_{p \in C} \text{dist}(p, \mu_C)^2$$

We can see how the total clustering is by summing up the sum of squared error for all clusters:

$$TD^2(C_1, C_2, \ldots, C_k) = \sum_{i=1}^{k} TD^2(C_i)$$

Clustering by minimization of Variance (Forgy, Lloyd)

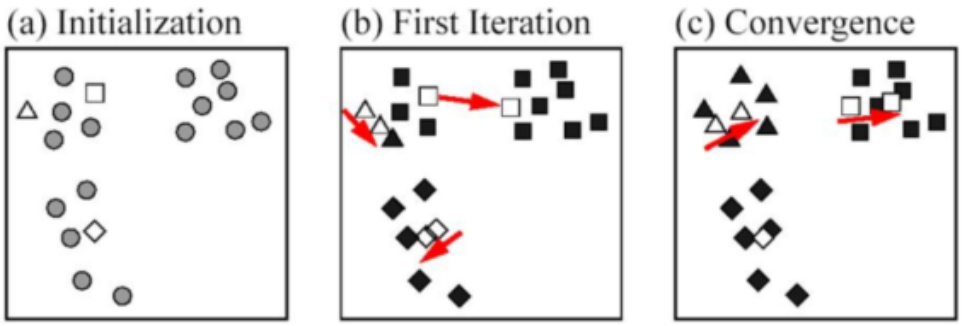
We chose k random points, and assigns all points to its closest representative. Then we compute the centroid and assign each point to its closest centroid. We repeat this process until there are no new assignments

---

Algorithm 4.1 (Clustering by Minimization of Variance)

1. *start with k (e.g., randomly selected) points as cluster representatives (or a random partition in k "clusters")*
2. *repeat:*
   - 2.1 *assign each point to the closest representative*
   - 2.2 *compute new representatives based on the given partitions (centroid of the assigned points)*

   until there is no change in assignment



K-means clustering

This is ALMOST like the clustering by minimization of variance, but the centroid is recalculated as soon as a point is reassigned, rather than waiting for all points to be assigned.

Since the centroid is calculated immediately after a point is reassigned, the cluster will depend on the order the data is processed.

Runtime: O(k * n) pr. Iteration (usually 10 iterations)

pros

- efficient: $\mathcal{O}(k \cdot n)$ per iteration, number of iterations is usually in the order of 10.
- easy to implement, thus very popular

cons

- k-means converges towards a *local* minimum
- k-means (MacQueen-variant) is order-dependent
- deteriorates with noise and outliers (all points are used to compute centroids)
- clusters need to be convex and of (more or less) equal extension
- number k of clusters is hard to determine
- strong dependency on initial partition (in result quality as

k-medoids

K-medoids is basically the same however it does not compute the centroid, but the medoid, this means that it can be used in more types of spaces, since it can work in non-euclidian spaces. It is also less sensitive to outliers.

k-modes

This works with categorical data. This likewise minimizes distance. However, if two points are not the same category, their distance will just be one otherwise 0. If the categories are numerical, we can use a traditional centroid.

Choosing k and evalutaing

There can max be n – 1 clusters

Chose the one with the best score. How to evaluate this is hard since TD^2 becomes smaller with higher k, therefore we use silhouttes since it is independent of k

We are evaluating to things with silhouttes the cohesion (how tighly are members connection in a cluster) and separation, which evaluates how well clusters are separated.

There are the standard and the simplified version, the standard looks at the average distance between all members of a cluster and all members of the cluster that is closest. While The simplified only looks at the centroids.

Silhouette [Rousseeuw, 1987]

- let $a(o)$ be the average distance between o and all oth members of the same cluster
- let $b(o)$ be the average distance between o and all members of another cluster that minimizes this averag
- the silhouette of o is given by

$$s(o) = \frac{b(o) - a(o)}{\max(a(o), b(o))}$$

- it holds that $-1 \leq s(o) \leq 1$
- $s(o) \approx -1, 0, 1$: bad, indifferent, good assignment of o

Simplified Silhouette Coefficient [Rousseeuw 1987]: Points

- let $a(o)$ be the distance between o and its "own" cluste representative
- let $b(o)$ be the distance between o and the closest "foreign" cluster representative
- the silhouette of o is given by

$$s(o) = \frac{b(o) - a(o)}{\max(a(o), b(o))}$$

- it holds that $-1 \leq s(o) \leq 1$
- $s(o) \approx -1, 0, 1$: bad, indifferent, good assignment of o

- given the individual silhouette values $s(o)$
- the silhouette s_C of a clustering C is the average silhouette of all n points:

$$s_C = \frac{1}{n} \sum_{o \in \mathcal{D}} s(o)$$

- interpretation: $s_C > 0.7$: strong cluster structure, $s_C > 0.5$: reasonable cluster structure

Empty clusters

These cause problems since we cannot do the mean of an empty cluster.

We can remove one cluster (which I do), use a point that is farthest away from any centroid, or chose a random point as a new centroid.

Classification

A supervised process where we have examples and information about the classes. The goal is to map an object o to a class c with a function, the goal of classification is to learn this function also known as the target function.

The function we learn will only be an approximation of the target function.

We train the model on training data.

A classifier has some assumptions on how the separation between classes can be achieved.

These assumptions define the hypothesis space or the set of hypotheses that are possible to create (example a decision tree will only separate verticaly and horizaontally). This hypothesis space is the bias, which means that certain algorithms tend to prefer certain hypotheses. However this is needed in order to avoid overfitting or learning the training data by heart.

Evalutaion

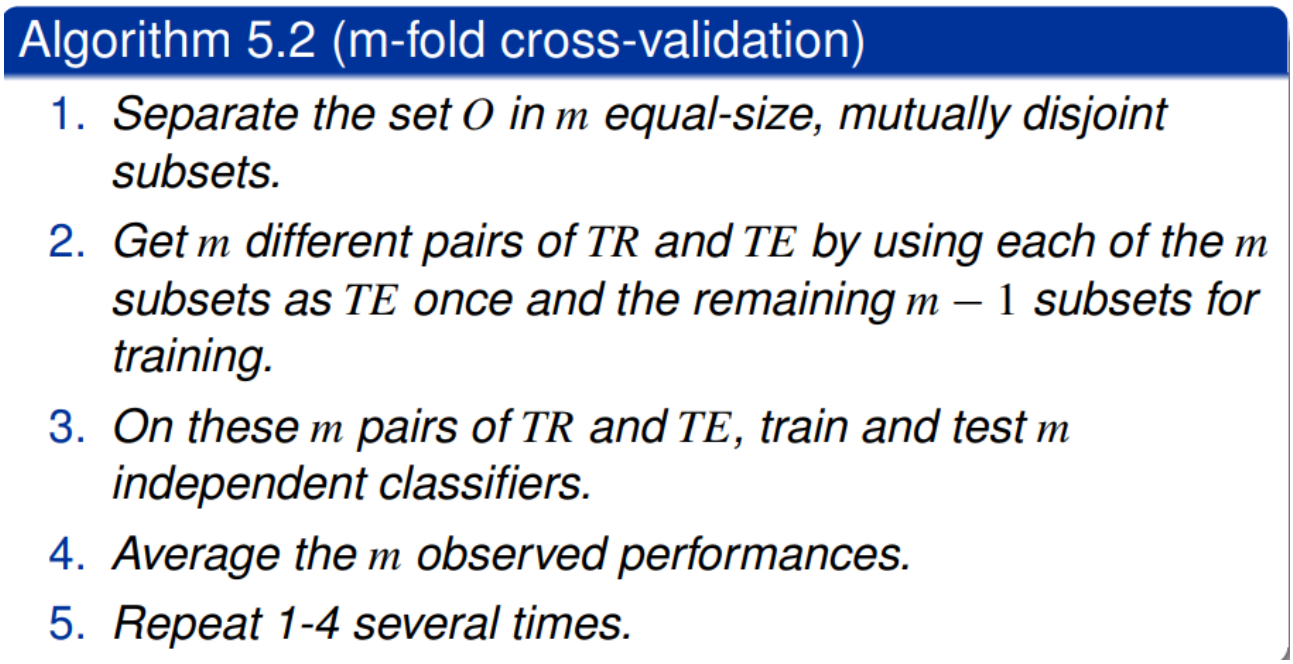
We tend to get higher evaluation scores on training data than on test data, this is called overfitting. Moreover, it is hard to evaluate a classifier, since we do not know anything about the predictions on unseen data, we cannot verify them.

To avoid this problem we train the data on training data and evaluate on test data. The problem is that we are then reducing our training data.

m-fold validation

This is a method to evaluate our performance. We basically split our data in m subsets. For each subset we leave one out for testing and use the rest for training. We then average the observed performances and get

a more robust performance result.



Stratificaiton

With stratification we try to represent the class proportions in each fold. Each class must be present in the training set, and the distribution of each should reflect the distribution of classes in the whole training set.

Bootstrap

With bootstrap we randomly draw objects with replacement (that is they can be reused) from the whole set and train on this.

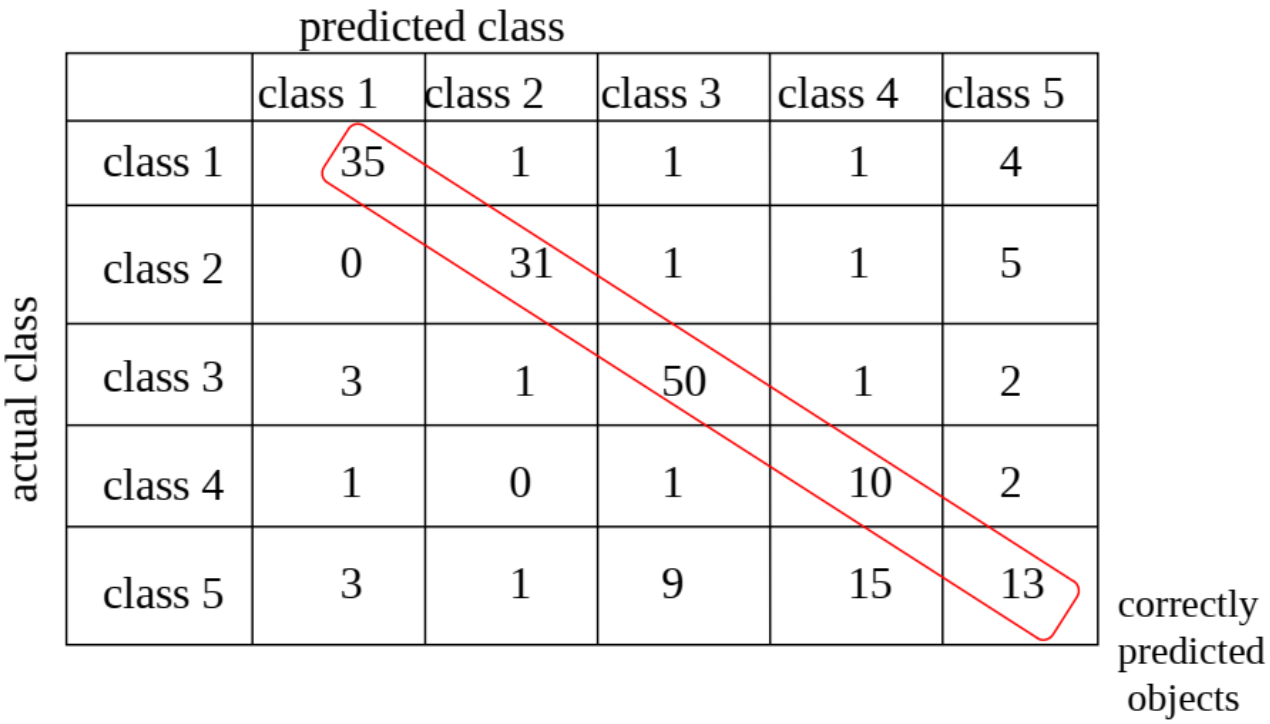
Leave-one-out or jack-knife

This is n-fold cross validation. For each sample we leave one out for testing and train on the rest of the dataset. We then average the results of all the test. This gives a pretty pessimistic performance.

Confusion matrix

A confusion matrix shows the number of correctly predicted classes on the diagonal and how the classifier misclassified on the classes:

The confusion matrix represents the number of correctly and incorrectly predicted classes per actual and per predicted class:



	class 1	class 2	class 3	class 4	class 5
class 1	35	1	1	1	4
class 2	0	31	1	1	5
class 3	3	1	50	1	2
class 4	1	0	1	10	2
class 5	3	1	9	15	13

Accuracy, true error and apparent error

Accuracy is the number of correctly classified predictions divided by the total number of predictions in the test set

True error is the number of misclassified predictions divided by the total number of predictions in the test set

The apparent error is the number of misclassified predictions in the training set divided by the length of the training set

accuracy of h on TE:

$$\text{acc}_{TE}(h) = \frac{|\{o \in TE | h(o) = f(o)\}|}{|TE|}$$

true classification error:

$$\text{err}_{TE}(h) = \frac{|\{o \in TE | h(o) \neq f(o)\}|}{|TE|}$$

apparent classification error:

$$\text{err}_{TR}(h) = \frac{|\{o \in TR | h(o) \neq f(o)\}|}{|TR|}$$

True positives and false positives

We a binary classification problem we can split our observations into true positives and false positives in a matrix

	predicted positive	predicted negative
given positive	TP (true positive)	FN (false negative)
given negative	FP (false positive)	TN (true negative)

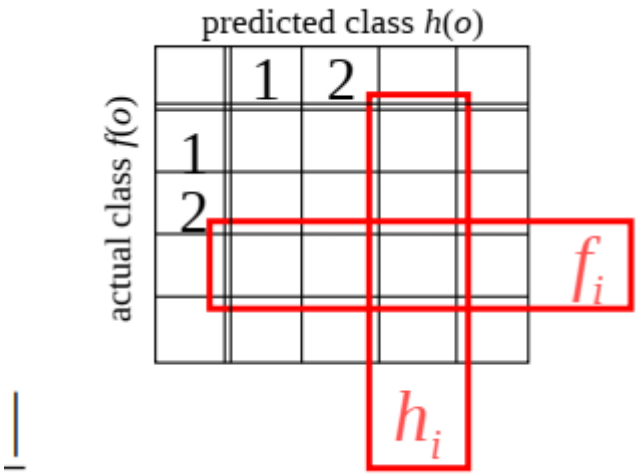
This is useful if there for example is an intereset in minimizing false negatives (for example with medical tests, where we want to minize the number of sick people being classified as healthy).

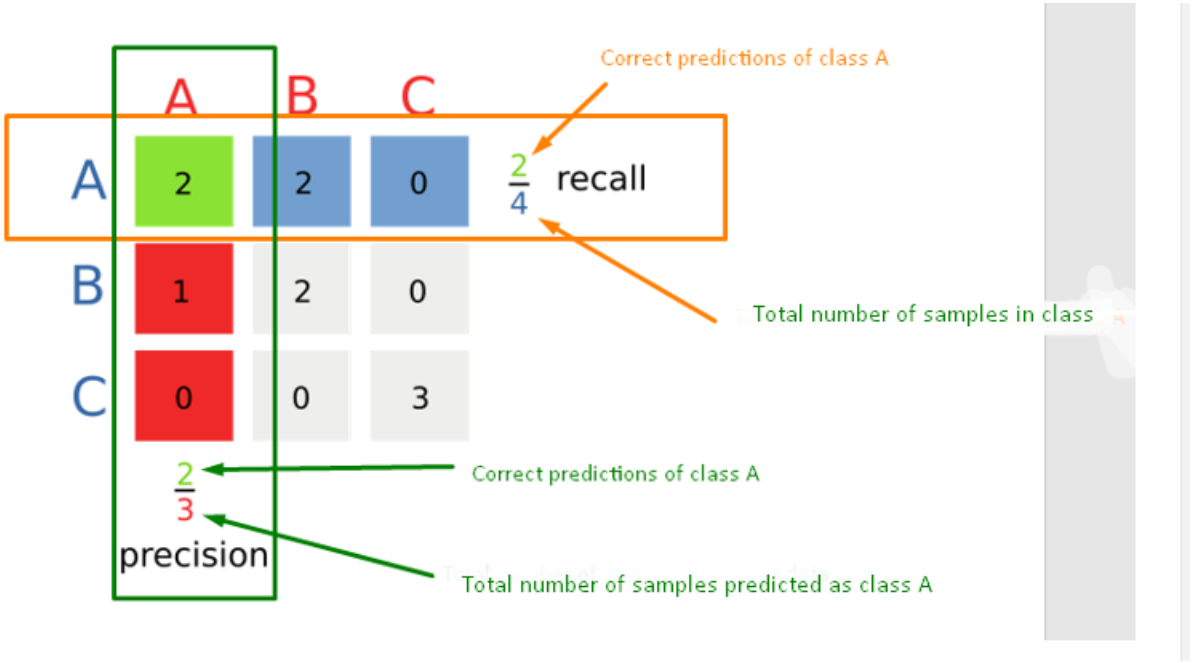
Precision, recall

Recall and precision are very similar, but they fundamentally try to answer the following questions for a class:

Precision: What proportion of a predicted class was actually correct? It is calculated by taking the number of correct predictions in a class and dividing it by the number of samples in that class

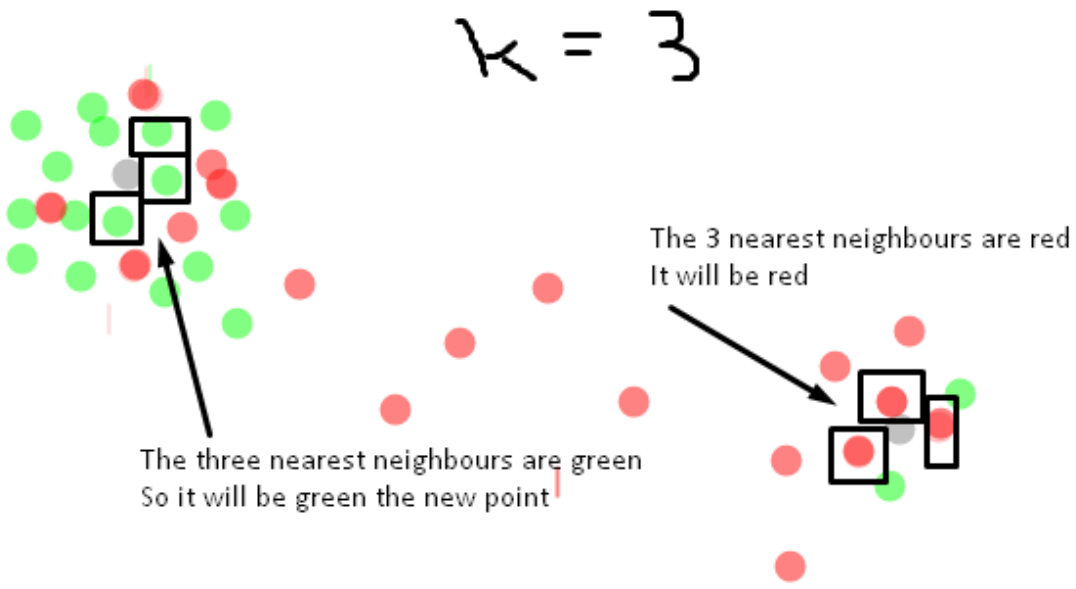
Recall: What proportion of a the class was correctly classified? It is calculated by taking the number correct predictions in a class and dividing it by the total number of samples that was predicted as belonging to this class.





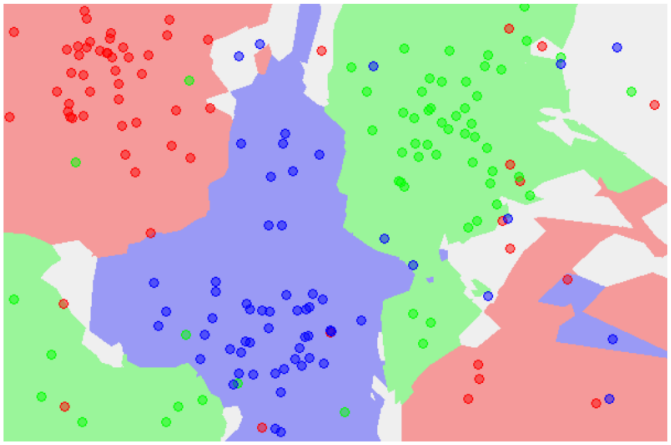
KNN (K nearest neighbour)

K-nearest neighbour is a very simple algorithm. It takes a point and then depending on its k-nearest neighbours assigns the object to the majority class. If k = 1 it will assign it to the same class as the 1 closest neighbour, it is 3 it will look at the three nearest neighbours and assign it to the class, which contains the majority. It is a lazy learner, since it does not save a model, it only contains the data, and the learning process happens, when new data points are added to the training data.



This will create a bunch of decision boundaries, which can be compared to a voronochi diagram

$k = 5$



K should not be too small, otherwiser it might be too sensitive to outliers. If it is to large, it might overreach and take unrelated classes into account. 1 < k < 10

It is possible to add a weight on classes. This can either be on distance or on on the class proportions, so smaller classes have a higher voting power.

Ties: If there is a tie, we can do one of the following:

- Chose a different k
- Randomly chose between one of the values

- Look at the distances between points and chose those, which have the shortest distance

pros

- easy to apply: requires only training data and distance function
- often good classification accuracy
- incremental: easy adaptation to new training data
- no training required ("lazy learner")

cons

- inefficient prediction: each decision requires k nearest neighbor query
- does not deliver explicit knowledge about classes
- difficult to choose k, esp. if classes are of very different size

Baysian learning

It is a way to calculate conditional probabilities, what is the probability of y given x. The best example is this, If I think I saw a donkey, but know it is most likely a horse, I will conclude that it is a mule.

Sample space: A set of all possible outcomes in a random process

Event: A subset of the sample space, which contains individual outcomes of this process is an event.

Allowable events: This is a set of events that are possible, where each subset each also a subset of the sample space.

Probability function: A function between 0 and 1, which returns the probability of an event, the probability of the whole sample space is always 1, and the sum of probabilities for k events is the same as the probability for the union of these events.

Definition 6.2 (Probability Space)

A probability space is given by three components:

1. a sample space Ω;
2. the allowable events $\mathcal{F} = \wp(\Omega)$; and
3. a probability function $\Pr : \mathcal{F} \to \mathbb{R}$.

.

- $E_1 \cap E_2$ denotes the occurrence of both, E_1 and E_2 (i.e., their co-occurrence).
- $E_1 \cup E_2$ denotes the occurrence of either E_1 or E_2 (or both).
- $E_1 \setminus E_2$ denotes the occurrence of event E_1 without E_2 occurring as well.
- $\overline{E} = \Omega \setminus E$ denotes the complementary event of E.

Lemma 6.3 (Combined Probability)

For any two events E_1 and E_2:

$$\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1 \cap E_2)$$

Lemma 6.4 (Union Bound)

For any finite or countably infinite sequence of events $E_1, E_2, E_3, \ldots$:

$$\Pr\left(\bigcup_{i\geq 1} E_i\right) \leq \sum_{i\geq 1} \Pr(E_i).$$

Two events are independt if the probability of their co-occurrence is the same as the the events multiplied by each other. If events are independent, then knowledge about one event does not change the probability of the other events. So if I flip a coin and get heads, it wont influence the probability of my second coinflip.

Definition 6.5 (Independent Events)

Two events E and F are *independent* if and only if

$$\Pr(E \cap F) = \Pr(E) \cdot \Pr(F).$$

More generally, events $E_1, E_2, \ldots, E_k$ are mutually independent if and only if

$$\forall I \subseteq [1,k] : \Pr\left(\bigcap_{i\in I} E_i\right) = \prod_{i\in I} \Pr(E_i).$$

To get the conditional probability of (e and f) we take the probability of e and f together and divide it by the probability of f happening at all:

Definition 6.6 (Conditional Probability)

The *conditional probability* that event E occurs given that event F occurs is

$$\Pr(E|F) = \frac{\Pr(E \cap F)}{\Pr(F)}$$

Quality measures of association rules can be seen as probabilities.

Support is the probability of a sample being in the database

the database:

$$\Pr(X) = \frac{s(X)}{|\mathcal{D}|}$$

Confidence is the conditional probability of a association rule:

$$\text{conf}(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)} = \frac{\Pr(X \cap Y)}{\Pr(X)}$$

The probability of an event is the same as the sum of all conditional probabilities for that event times the probability of the condition:

Theorem 6.7 (The Law of Total Probability)

Let $E_1, E_2, \ldots, E_n$ be mutually disjoint events in the sample space Ω, and let $\bigcup_{i=1}^{n} E_i = \Omega$. Then

$$\Pr(B) = \sum_{i=1}^{n} \Pr(B \cap E_i) = \sum_{i=1}^{n} \Pr(B|E_i) \Pr(E_i).$$

Bayes theorem

I sin simpleste form er bayes theorem for P(A | B) at vi tager den conditional probability for P(A | B) og ganger den med P(A) og dividerer den med P(B)

$$\Rightarrow \Pr(A|B) = \frac{\Pr(B|A) \Pr(A)}{\Pr(B)}$$

Dette kan generaliseres

Theorem 6.8 (Bayes' Rule)

Let $E_1, \ldots, E_n$ be mutually disjoint events, and let $\bigcup_{i=1}^{n} E_i = \Omega$. Then for any other event B, $\Pr(B) > 0$, $j = 1, \ldots, n$:

$$\Pr(E_j|B) = \frac{\Pr(E_j \cap B)}{\Pr(B)} \tag{6.1}$$

$$= \frac{\Pr(B|E_j)\Pr(E_j)}{\sum_{i=1}^{n} \Pr(B|E_i)\Pr(E_i)} \tag{6.2}$$

- A doctor sees a patient with fever and rash.
- 80% of patient with flu, 45% of allergy patients, and 90% of infection patients have these symptoms.
- The doctor knows that 50% of the patients she sees have flu, 40% have allergy, and 10% have an infection.
- Should the doctor treat the patient for infection?

This gives the following:

```
(90/100 * 10/100) / ((90/100 * 10/100) + (80/100 * 50/100) + (45/100 * 40/100))
✓ 0.1s
0.13432835820895522
```

With a baysian there is always different hypotheses, here we are hypothesizing wether the patient has flu, so there is a 1 – 0.13 = 87% percent chance that is it something else. To test the others we have to perform the same test but with other values in the top and bottom, they have to be swapped.

With a model we set a prior which in the formula below is Pr(B). If we don't know it we can set it to ½, but if we then get some more knowledge about the prior, we can change it.

$$\Pr(B|A) = \frac{\Pr(B \cap A)}{\Pr(A)} = \frac{\Pr(A|B)\Pr(B)}{\Pr(A|B)\Pr(B) + \Pr\left(A|\overline{B}\right)\Pr\left(\overline{B}\right)}$$

Baysian learning and probabilities

- P(h) *prior probability of* h, reflects any background knowledge about the chance that h is correct
- P(D) *prior probability of* D, probability that D will be observed
- P(D|h) probability of observing D given a world in which h holds
- P(h|D) *posterior probability of* h, reflects confidence that h holds after D has been observed

With this we can observe the likelyhoof of an hypothesis given the observed data:

Bayes Theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Which can be used to get the most likely hypothesis

Brute-Force MAP LEARNING algorithm

1. For each hypothesis h in H calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis $\boldsymbol{h_{MAP}}$ with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{argmax}\ P(h|D)$$

- ***P(D|h)*** is the probability of observing the target values D = ***(d_1 . . .d_m)*** for the fixed set of instances ***(x_1 . . . x_m),*** given a world in which hypothesis ***h*** holds
- Since we assume noise-free training data, the probability of observing classification ***d_i*** given ***h*** is just ***1*** if ***d_i = h(x_i)*** and 0 if ***d_i # h(x_i).*** Therefore,

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \in D \\ 0 & \text{otherwise} \end{cases}$$

This gives that the likelihood of a hypothesis is just:

$$\frac{|\mathcal{H}_D|}{|\mathcal{H}|}$$

So the amount of hypothesis d divided by the amount of hypotheses or said another way, the probability of each hypothesis is just the proportion of each hypothesis, so if P(h_1 | D) = 0.7 and P(h_2 | D) = 0.3 then the MAP is h_1

By letting P(D | h) take other values than 0 or 1 we can model noisy data.

Bayes optimal classifier

- We obtain the most probable classification by combining the predictions of all hypotheses weighted by the posterior probabilities.
- For the set of classes C, for any $c_j \in C$, we have

$$\Pr(c_j|\mathcal{D}) = \sum_{h_i \in \mathcal{H}} \Pr(c_j|h_i) \Pr(h_i|\mathcal{D})$$

- The optimal classification is therefore:

$$\arg\max_{c_j \in C} \sum_{h_i \in \mathcal{H}} \Pr(c_j|h_i) \Pr(h_i|\mathcal{D})$$

- Any system classifying new instances according to this rule is called a “Bayes optimal classifier”.

The bayes optimal classifier has a different hypothesis space since it can give a probability score

Naïve bayes

The naïve bayes algorithm test the likelihood by combining the likelihood of each feature / attribute and which class is most likely is, it then choses the one based on this. Due to this is assumes that each feature is independent of each other.

ID	forecast	temperature	humidity	wind	play tennis?
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

$$h_{\text{naïve Bayes}} = \arg\max_{c_i \in C} \prod_{j=1}^{n} \Pr(a_j|c_i) \Pr(c_i)$$

$$\begin{aligned} h_{\text{naïve Bayes}} &= \arg\max_{c_i \in \{\text{yes,no}\}} \prod_{j=1}^{n} \Pr(a_j|c_i) \Pr(c_i) \\ &= \arg\max_{c_i \in \{\text{yes,no}\}} \Pr(\text{sunny}|c_i) \Pr(\text{cool}|c_i) \Pr(\text{high}|c_i) \\ &\qquad \Pr(\text{strong}|c_i) \Pr(c_i) \end{aligned}$$

$$\Pr(\text{yes}) = \frac{9}{14} = 0.64 \qquad \Pr(\text{wind=strong}|\text{yes}) = \frac{3}{9} = 0.33$$
$$\Pr(\text{no}) = \frac{5}{14} = 0.36 \qquad \Pr(\text{wind=strong}|\text{no}) = \frac{3}{5} = 0.60$$

The Naïve Bayes classifier decides by finding the class maximizing the product of probabilities:

$$\Pr(\text{sunny}|\text{yes}) \Pr(\text{cool}|\text{yes}) \Pr(\text{high}|\text{yes}) \Pr(\text{strong}|\text{yes}) \Pr(\text{yes}) = 0.0053$$
$$\Pr(\text{sunny}|\text{no}) \Pr(\text{cool}|\text{no}) \Pr(\text{high}|\text{no}) \Pr(\text{strong}|\text{no}) \Pr(\text{no}) = 0.0206$$

$$h_{\text{naïve Bayes}}(\langle \text{sunny}, \text{cool}, \text{high}, \text{strong} \rangle) = \text{no}$$

If we are interested in the conditional probability for "no", we could normalize these quantities to sum up to one:

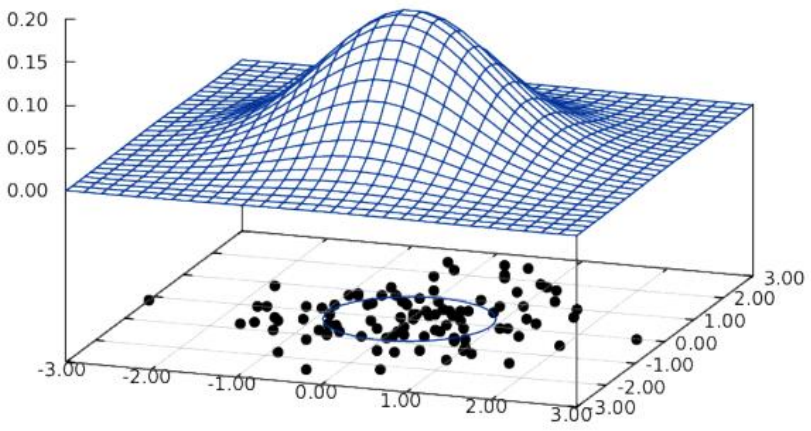
$$\frac{0.0206}{0.0206 + 0.0053} = 0.795$$

Learning with distributions

We are often interested in some complex event, such as the sum of two dice rolls, here we are mapping the result of our sample space to a real number, this is a random variable.

The mean or average is the expected value of a random variable:

When we look at data we can either interpret it as a distribution, where the data set is a random sample, that can be used to make inferences about the population or we can view it as points in a space, where we can calculate distances.



EM clustering (parametric learning)

Each cluster follows a probability density function (normal distribution typically) points are assigned to a cluster depending on the probability of belonging to a cluster. For each point we check it probability with conditional probability:

Hence we have the probability of some point to belong to a given cluster C_i:

$$\Pr(C_i|x) = W_i \cdot \frac{\Pr(x|C_i)}{\Pr(x)}$$

$$W_i = \frac{1}{n}\sum_{j=1}^{n} \Pr(C_i|x_j)$$

Each cluster is a hypothesis that tries to explain the data as k collections of distrivutions. We try to get the hypothesis that maximizes this expectation:

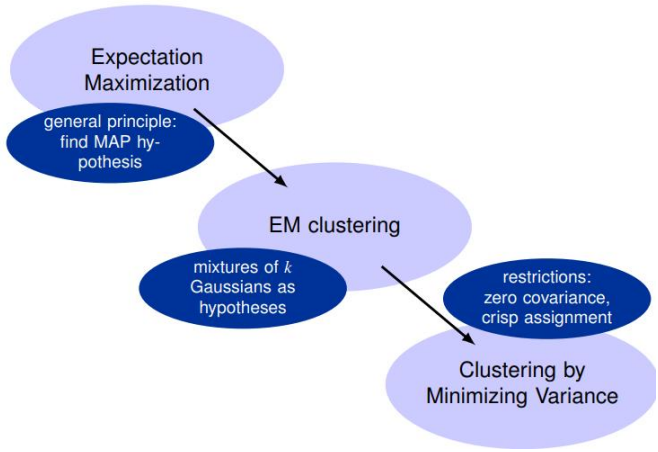
$$E(M) = \sum_{x \in \mathcal{D}} \log(\Pr(x))$$

- The "expectation" characterizes how well the clustering M explains the data.
- The higher $E(M)$, the more likely is it for the dataset to look like it does, given the model M.
- We therefore have to maximize $E(M)$ (cf. slide 309).

Runtime:

$$\mathcal{O}(n \cdot k \cdot \#\text{iterations})$$

The full model looks like this:



Non-parametric learning

With parametric learning we assume that the data follows a specific distribution such as gaussian. We learn the mean vector and the covariance matrix of this distribution.

Non-parametric learning tries to infer probabilities directly from the sample without knowing the specific distribution or density function. K-nearest neighbours and naïve bayes are non-parametric, since they do not make any assumptions about the underlying probability density function for the whole data set.

Density based clustering

With density based clustering we assume that a cluster if a area of high density separated from other clusters by a low density area

For am object in a cluster the local density must exceed a threshold:

For each object in a cluster, the local density exceeds a given threshold:

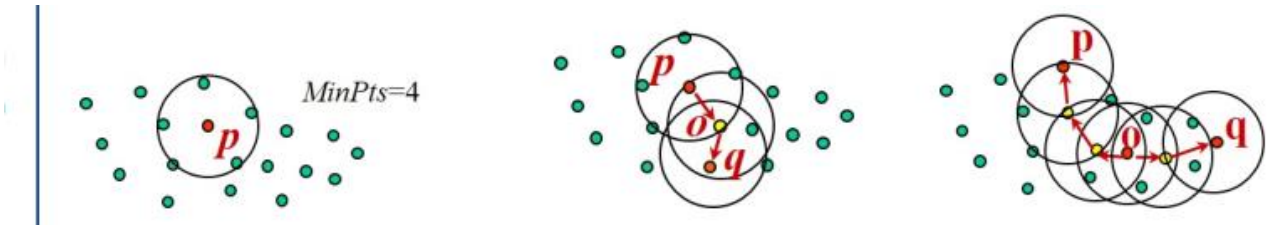
$$\forall o \in C : \phi(o) \geq \theta$$

Cf. density-contour by Hartigan [1975] and discussion by Kriegel et al. [2011], Campello et al. [2015].

Slide 438 gives a nice demonstration of how this threshold affect how many clusters we end up with. Moreover density based clusters often discard points outside clusters as noise instead of classifying them.

Density based structures makes an intuitive sense in natural data

The basic idea is that we have a core point, where the density is defined as having a threshold of points that must be reachable from this points, we then follow each reachable points until we hit a point without any reachable points, and then we have our cluster.



It has two properties, which allow for efficient cluster searching:

A *(density-based) cluster* C w.r.t. ε and MinPts is a non-empty subset of $\mathcal{D}$ satisfying the following conditions:

1. *maximality*: $\forall p, q$: if $p \in C$ and q is density-reachable from p w.r.t. ε and MinPts, then $q \in C$.
2. *connectivity*: $\forall p, q \in C$: p is density-connected to q w.r.t. ε and MinPts.

Algorithm 7.2 (DBSCAN [Ester et al., 1996])

```
DBSCAN(DB,Eps,MinPts)
  // DB is UNCLASSIFIED
  C_Id := nextId(NOISE);
  FOR i FROM 1 TO DB.size DO
    Point := DB.get(i);
    IF Point.C_Id = UNCLASSIFIED THEN
      IF ExpandCluster(DB, Point, C_Id, Eps, MinPts) THEN
        C_Id := nextId(C_Id)
```

Algorithm 7.3 (DBSCAN – expandCluster)

```
ExpandCluster(DB, Point, C_Id, Eps, MinPts):  Boolean
  seeds := DB.rq(Point,Eps)
  IF seeds.size < MinPts THEN // no core point
    DB.changeC_Id(Point,NOISE);
    RETURN FALSE;
  ELSE // all points in seeds are dens-reach from Point
    DB.changeC_Ids(seeds,C_Id);
    seeds.delete(Point);
  WHILE seeds <> Empty DO
    currentP := seeds.first();
    result := DB.rq(currentP,Eps);
    IF result.size >= MinPts THEN
      FOR i FROM 1 TO result.size DO
        resultP := result.get(i);
        IF resultP.C_Id IN {UNCLASSIFIED,NOISE} THEN
          IF resultP.CiId = UNCLASSIFIED THEN
            seeds.append(resultP);
          DB.changeC_Id(resultP,C_Id);
    seeds.delete(currentP);
  RETURN TRUE;
```

One problem is that sometimes different clusters have different density thresholds, this means that some clusters are more dense than others. A way to try and account for this is the shared neighbour similarity

---

Algorithm 7.5 (SNN clustering [Ertöz et al., 2003])

given k, ε, MinPts:

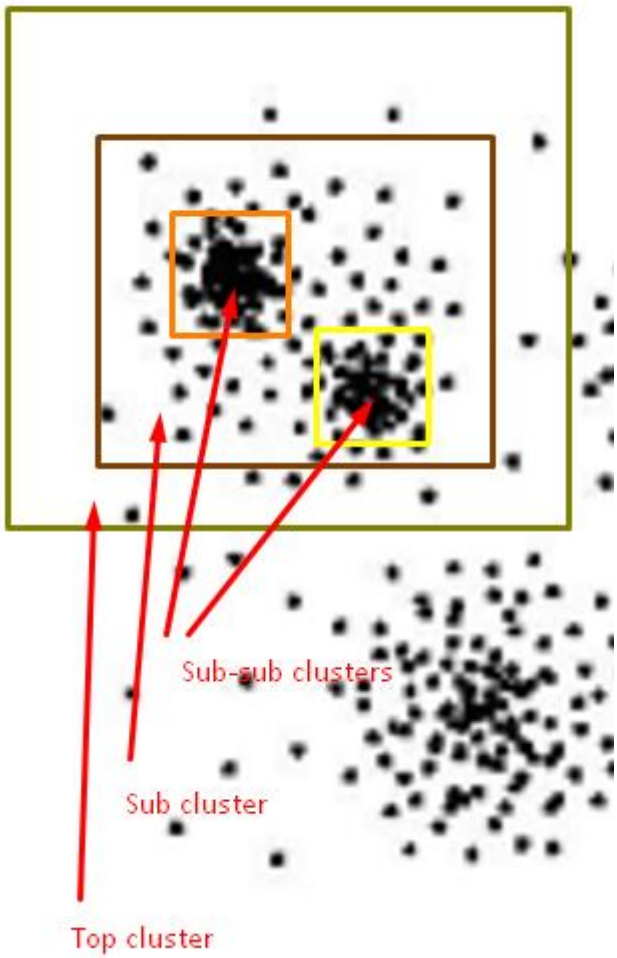
1. *compute similarity matrix and -graph (as for Alg. 7.4)*
2. *compute SNN density for each point w.r.t. ε*
3. *derive core points w.r.t. MinPts*
4. *cluster core points p, q if $\mathrm{SNN}_k(p, q) \geq \varepsilon$*
5. *assign a non-core point p to cluster C if there is a core point $q \in C$ with $\mathrm{SNN}_k(p, q) \geq \varepsilon$*
6. *all remaining non-core points are noise*

Note that:

Steps 2 to 6 are nothing but DBSCAN when using a different notion of distance!

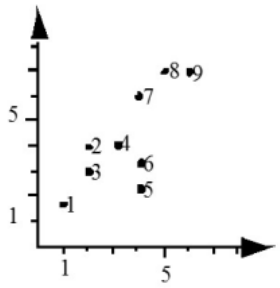
Hierarchical clustering

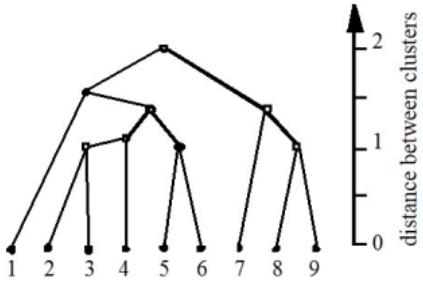
The idea is that a cluster is made up of subclusters, which form hierarchies



This is done using trees

- root: represents the complete database
- leaf: represents a single object
- inner node: represents a cluster containing all the objects of the subtree rooted at this node





categories of hierarchical clustering algorithms (typically best-first heuristic):

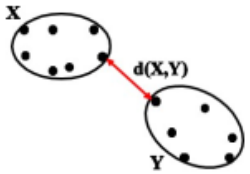
- bottom-up construction of the dendrogram (agglomerative), at each node merging two clusters
- top-down construction of the dendrogram (divisive) at each node splitting a cluster into two parts

Algorithm 7.6 (Agglomerative Hierarchical Clustering)

1. *Initial clusters consist of one object each.*
2. *Compute distances between all pairs of clusters.*
3. *Merge those two clusters with smallest distance.*
4. *Compute distance between the new cluster and all remaining clusters.*
5. *If only one cluster remains: terminate; otherwise: repeat from step 3.*

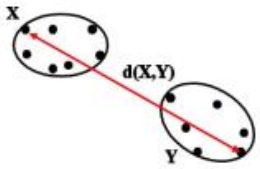
- Some common distance measures are:
  - single link
  - complete link
  - average link (a.k.a.: UPGMA – unweighted pair group method with arithmetic mean)

$$\text{dist}_{\text{single link}}(X, Y) = \min_{x \in X, y \in Y} \text{dist}(x, y)$$



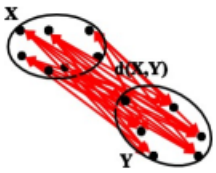
Single-Link

$$\text{dist}_{\text{complete link}}(X, Y) = \max_{x \in X, y \in Y} \text{dist}(x, y)$$



Complete-Link

$$\text{dist}_{\text{average link}}(X, Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} \text{dist}(x, y)$$



Average-Link

pros

- Does not require knowledge of number of clusters.
- Not only flat partition but a hierarchy of clusters.
- A single partition can be retrieved by some horizontal cut.

cons

- If you want to have a flat partition, where is a good place to cut?
- Greedy heuristic, cannot correct bad decisions.
- single-link-effect, complete-link-effect
- in general: inefficient

OPTICS

Optics is almost like dbscan, but computes clusters for different density thresholds simultaneously. This gives a type of hierarchical clustering which is based on densityies.

data structures:

- SeedList (keeps visited points sorted w.r.t. current reachability distance)
- ClusterOrder (result, is built successively)

Algorithm 7.7 (OPTICS [Ankerst et al., 1999])

```
SeedList := ∅;
WHILE there are unlabeled objects in 𝒟 DO
  IF SeedList = ∅
  THEN
    insert arbitrary object into SeedList with reachdist = ∞;
  ELSE
    insert first object ("o") from SeedList
      with current reachdist into ClusterOrder;
    label o as processed;
  FOR ALL neighbor ∈ RQ(o, ε) DO
    SeedList.update(neighbor,o);
```

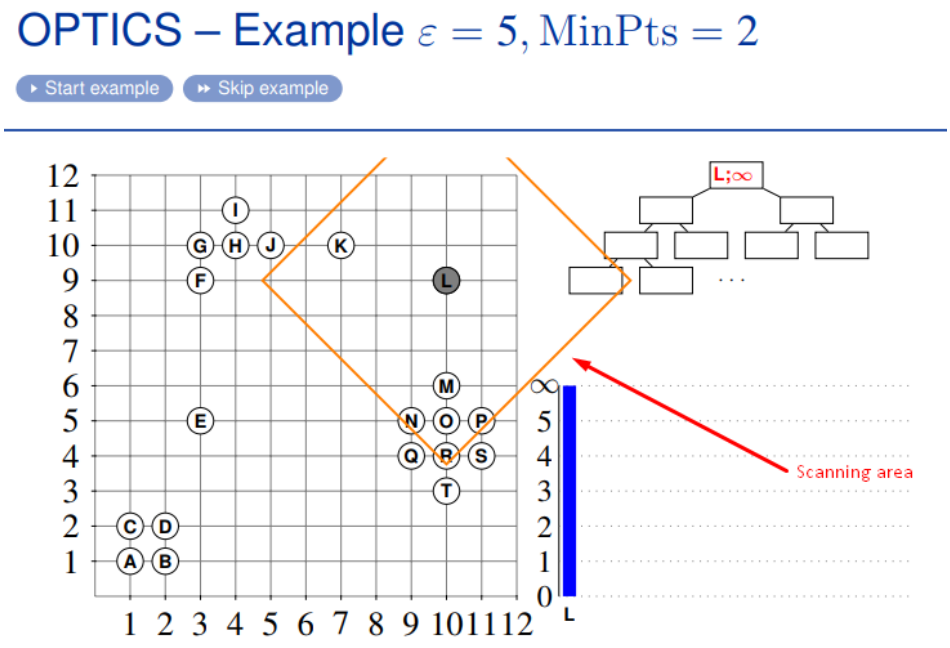
- for each object p in SeedList we keep the current reachdist: p.rdist
- SeedList is organized as a heap (sorted with increasing rdist)

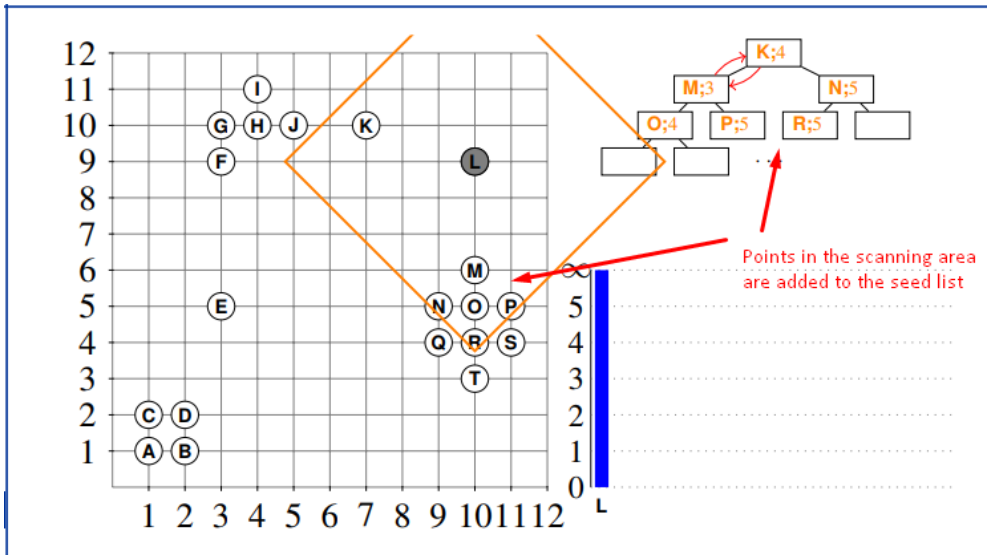
Algorithm 7.8 (OPTICS: update seed list)

```
SeedList :: update(o,obj)
  compute reachdist(o,obj) =: current_rdist_o;
  IF o is already in SeedList THEN
    IF current_rdist_o < o.rdist THEN
      o.rdist := current_rdist_o;
      reorganize heap;
  ELSE
    insert o with o.rdist := current_rdist_o into SeedList;
```

Here we have two parameters,

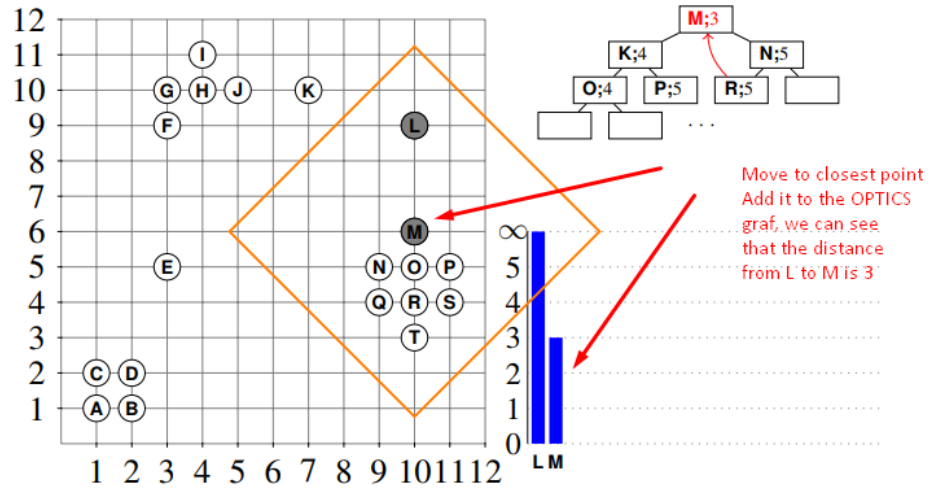
We have minpts or minimum number of points that must be reachable, and episolon, which is the area we are scanning for points in the cluster.

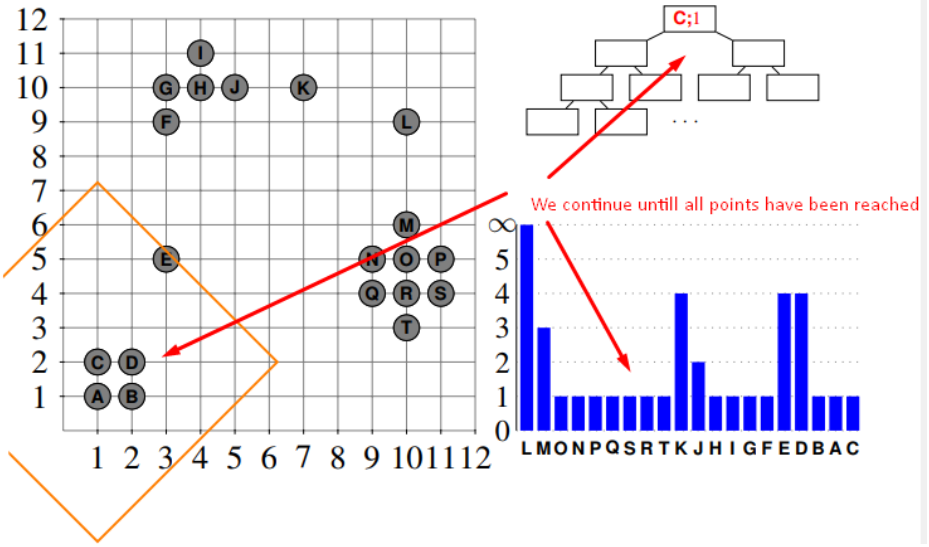


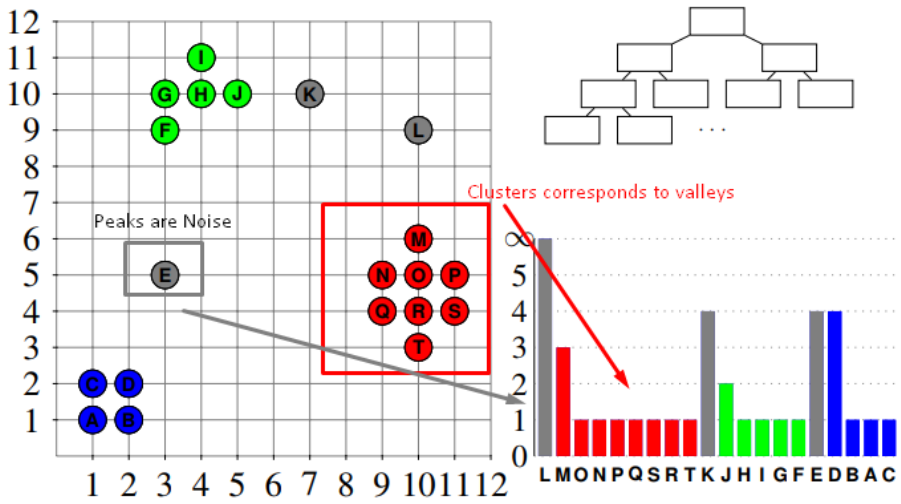


OPTICS – Example $\varepsilon = 5, \mathrm{MinPts} = 2$

Start example Skip example







The OPTICS plots shows the reachability distances from one point to another and how the algorithm traverses the data points most effectively.

Outlier detection

- outliers are points that have a low probability to be generated by the overall distribution (e.g., deviate more than 3 times the standard deviation from the mean)

basic assumptions

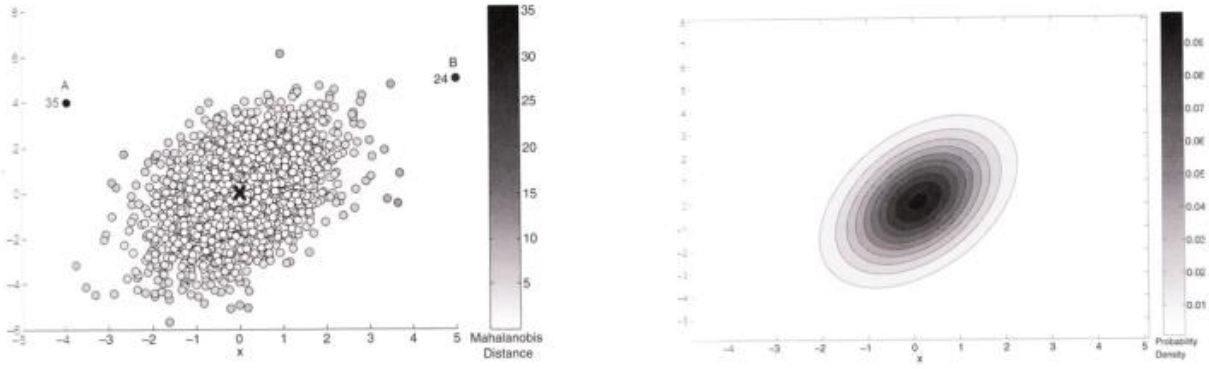
- normal data objects follow a (known) distribution and occur in a high probability region of this model
- outliers deviate strongly from this distribution

Statistical outlier detection

We calculate the mahalanobis distance for each data point to the center which is the mean vector

These distances should follow a chi-sqared distribution with d degrees of freedom (d is the number of dimensions).

- all points x, with $\text{MDist}(x, \mu) > \chi^2(0.975)$ $[\approx 3 \cdot \sigma]$ are considered outliers



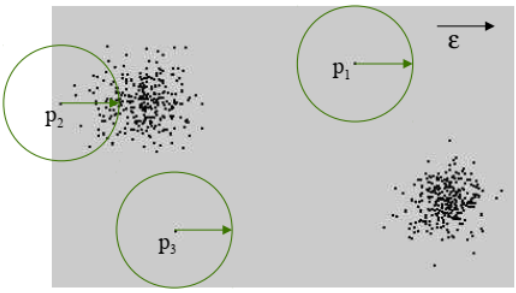
(Figure from Tan et al. [2006].)

Non-parametric outlier detection

Distance-based Outliers

DB(ε, π)-outlier [Knorr and Ng, 1997]

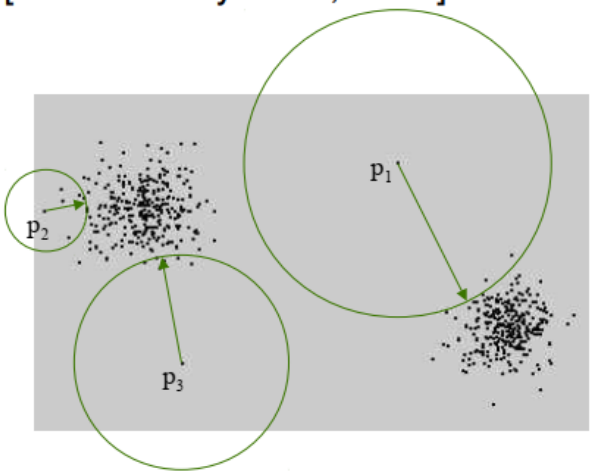
- given ε, π
- A point p is considered an outlier if at most π percent of all other points have a distance to p less than ε



$$OutlierSet(\varepsilon, \pi) = \left\{ p \left| \frac{Cardinality(q \in \mathcal{D} \,|\, \mathrm{dist}(q,p) < \varepsilon)}{Cardinality(\mathcal{D})} \leq \pi \right. \right\}$$

Outlier scoring based on kNN distances:

- Take the kNN distance of a point as its outlier score [Ramaswamy et al., 2000]



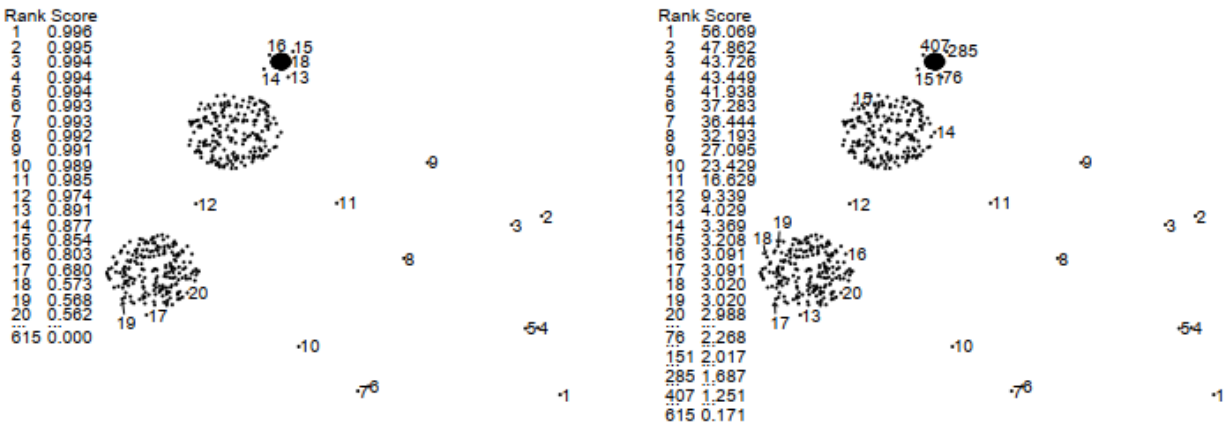
Density based outliers looks at the density and gets the outliers based on that.

Outliers can be local and global there can be outliers in a cluster, and there can be outliers globally. The ide is that that outliers have have lower local density than their neighbours.

Outlier evaluation

Outlier detection is not a classification problem of outlier vs. inlier, since there are too few outliers to make any meaningful evaluations (if there is only three outliers it is hard to do any proper training and evaluation). Therefore they deliver a outlier score, and the detection of outliers if based on the scores ranking for each data point.

It can be done manually:



examples taken from Campello et al. [2015]

And then create a two class confusion matrix:

- two-class confusion matrix ($C(o)$: class, $P(o)$: prediction):

$C(o)\backslash P(o)$	*outlier*	*inlier*
outlier	*TP*	*FN*
inlier	*FP*	*TN*

We then create a ROC curve, where the ROC AUC is between 0 and 1, the higher it is the better, if is 1 then it is perfect and there are no False positives.

Entropy, purity, Separation

Entropy is a measurement of how much randomness there is in the data. A random variable with n outcomes is:

Definition 8.1 (Entropy of a Discrete Random Variable)

The *entropy* in bits of a discrete random variable X is given by

$$H(X) = -\sum_{x} \Pr(X = x) \log_2 \Pr(X = x),$$

where the summation is over all values x in the range of X.

For a binary variable the function is:

$$H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p).$$

If the variable has n outcomes of equal like probability the entropy is:

$$= \log_2 n$$

The entropy of class distributions is given by:

$$H(\mathcal{D}) = -\sum_{i=1}^{k} \Pr(c_i|\,\mathcal{D}) \log_2 \Pr(c_i|\,\mathcal{D})$$

- Considering labeled data with $k = 2$, a very pure set (almost all labels belong to class A, only some belong to class B) has very low entropy (i.e., low disorder, low uncertainty):

Gini index

The gini-index measures the entropy / impurity of a dataset with k classes:

Definition 8.4 (Gini Index)

The Gini index as a measure for the (im-)purity of a data set $\mathcal{D}$ w.r.t. k classes $c_1, \ldots, c_k$ is given by:

$$G(\mathcal{D}) = 1 - \sum_{i=1}^{k} \Pr(c_i|\,\mathcal{D})^2$$

- If a dataset contains only one class, the probability of that class is 1, the dataset has minimal impurity, the Gini index is 0.
- When each class is equally represented, we have $\Pr(c_i|\,\mathcal{D}) = \frac{1}{k}$, the dataset is maximally impure, and the Gini index is $\frac{k-1}{k}$ (i.e., approaching 1 as $k \to \infty$).
- Probabilistic interpretation of the square: if we randomly draw two objects from $\mathcal{D}$, how likely are they belonging to the same class?

Decision trees

A decision tree works by partionioning the data until it eventually only contains one single classes. It does this by taking an attribute and measuring the information gain:

- Information gain is a measure based on the entropy.

$$H(T) = -\sum_{i=1}^{k} \Pr(c_i|T) \log_2 \Pr(c_i|T)$$

- Information gain measures the reduction of entropy (i.e., gain of information) by a split of set T into partitions $T_1, \ldots, T_m$:

$$\textit{information gain}(T, T_1, \ldots, T_m) = H(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} H(T_i)$$

- Higher information gain means larger reduction of entropy.
- We choose the attribute and split point that maximize the information gain.

We can use the weighted gini index to compare partionionings:

$$G(\mathcal{D}) = 1 - \sum_{i=1}^{k} \Pr(c_i|\,\mathcal{D})^2$$

n an analogous way, we can use the weighted Gini index of nduced partitions to compare partitionings:

$$G(T_1, \ldots, T_m) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} G(T_i)$$

- Smaller value of the Gini index means lower impurity.
- We choose the attribute and the split that minimizes the Gini index.