Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Felix Borutta

Munich, June 06, 2019

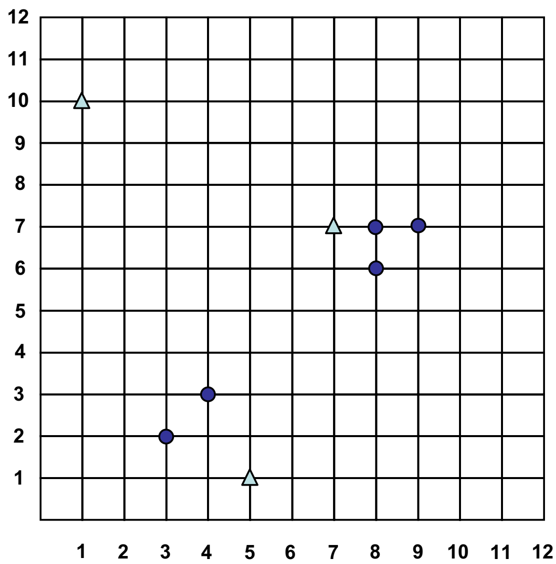
Managing Massive Multiplayer Online Games
SS 2019

Exercise Sheet 6: Knowledge Discovery and Data Mining

The assignments are due June 12, 2019

Assignment 6-1 *Instance-based learning: kNN-Classification*

Consider the following data set consisting of 8 points. The triangles are one class and the circles are another class.



Determine the classes of the given data points by using the k-nearest neighbors algorithm. If not stated differently, use the Manhattan distance (l_1 norm) as distance measure:

$$L_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

(a) Determine the class of point (2,7) for $k = 2$ using the majority class among the k-nearest neighbors, i.e. the point is assigned to the class which occurs most frequently among its k-nearest neighbors.

(b) Determine the class of point (2,7) for $k = 3$ using the majority class among the k-nearest neighbors.

(c) Determine the class of point (2,7) for $k = 5$ using the majority class among the k-nearest neighbors.

(d) Determine the class of point (6,1) for $k = 3$ using the majority class among the k-nearest neighbors.

(e) Determine the class of point (6,1) for $k = 3$ using the majority class among the k-nearest neighbors. This time, employ a weighted version for the class decision, i.e., weight the class occurrences with the inverse Manhattan distance.

$$L_1(x, y)^{-1} = \frac{1}{\sum_{i=1}^{d} |x_i - y_i|}$$

In general, the k-nearest neighbor query retrieves those k objects of a database D that are closest to a given query object with respect to some distance measure. Precisely the set of k-nearest neighbors can be defined in two ways, the so-called deterministic and non-deterministic definitions:

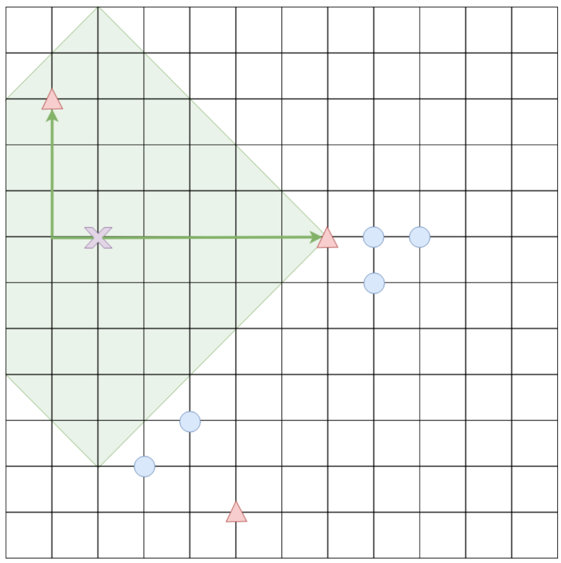
Deterministic: The set of k-nearest neighbors is the set $NN(q,k) \subseteq DB$ with *at least* k objects such that

$$\forall o \in NN(q,k), \forall o' \in DB \backslash NN(q,k) : dist(q,o) < dist(q,o').$$

Non-deterministic: The set of k-nearest neighbors is the set $NN(q,k) \subseteq DB$ with *exactly* k objects such that

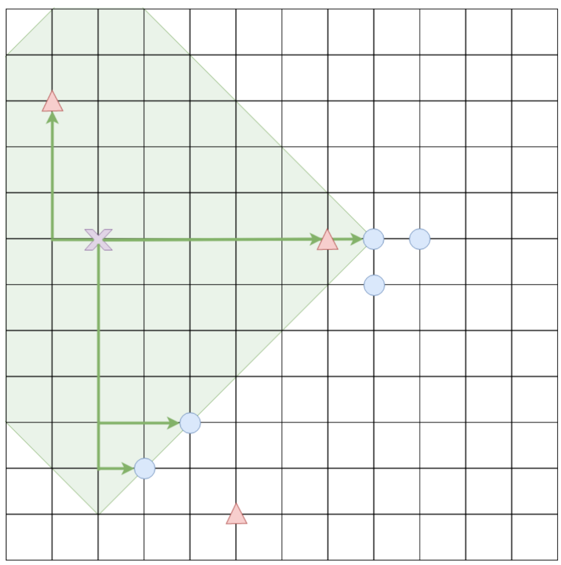
$$\forall o \in NN(q,k), \forall o' \in DB \backslash NN(q,k) : dist(q,o) \leq dist(q,o').$$

(a) Determine the class of point (2,7) for $k = 2$ using the majority class among the k-nearest neighbors.



The green area denotes the k-distance, which is 5. Two objects fall into this area, both of class "triangle" and hence the test object is classified as triangle.

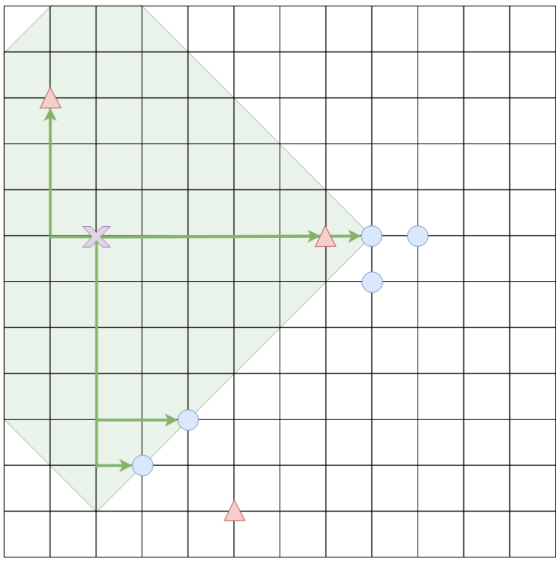
(b) Determine the class of point (2,7) for $k = 3$ using the majority class among the k-nearest neighbors.



Again, the green area denotes the k-distance, which is 6. Here we can distinguish between a deterministic and non-deterministic kNN classification.

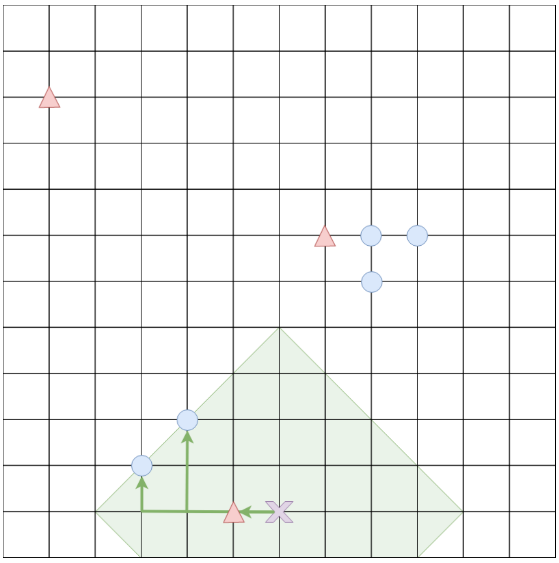
In case of non-deterministic (only 3 objects are considered): the target class is "triangle". In case of deterministic (all 6 objects that fall into the green area are considered): the target class is "circle".

(c) Determine the class of point (2,7) for $k = 5$ using the majority class among the k-nearest neighbors.



The target class is "circle".

(d) Determine the class of point (6,1) for $k = 3$ using the majority class among the k-nearest neighbors.



The target class is "circle".

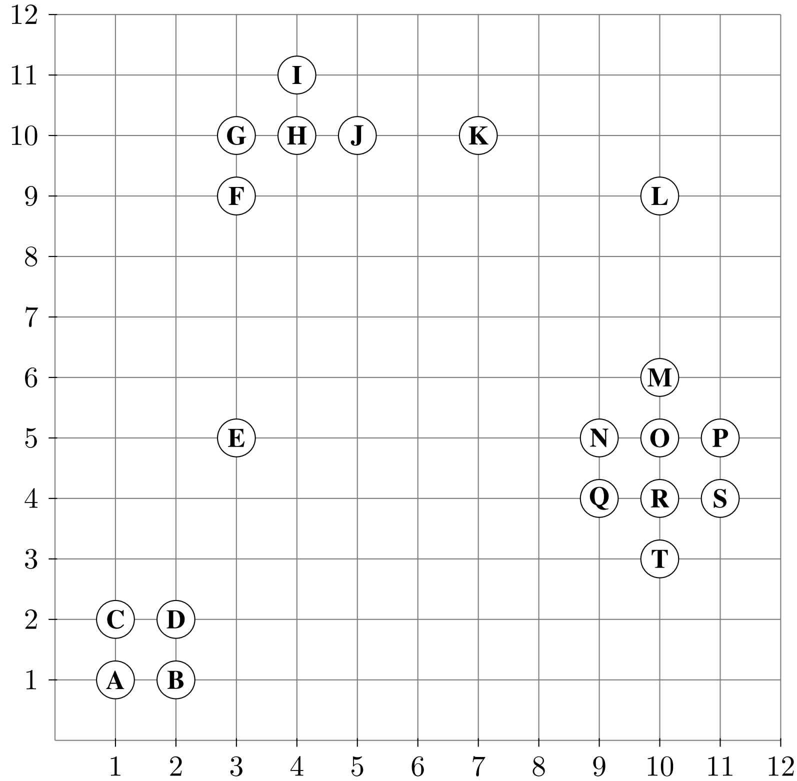
(e) Determine the class of point (6,1) for $k = 3$ using the majority class among the k-nearest neighbors. This time, employ a weighted version for the class decision, i.e., weight the class occurrences with the inverse Manhattan distance.

$$L_1(x, y)^{-1} = \frac{1}{\sum_{i=1}^{d} |x_i - y_i|}$$

The two circle objects both get a weight of 0.25, the triangle object gets a weight of 1, and therefore the classification decision is triangle since $1 > 0.25 + 0.25$.

Assignment 6-2 *Unsupervised Learning: Clustering with DBSCAN*

The following dataset is given:



Cluster this dataset using DBSCAN. Use the Manhattan distance as distance function and the parameters $\epsilon = 1.1$ and $minPts = 3$.

The solution can be found in the extra document provided on the website.

Assignment 6-3 *Supervised Learning: Naive Bayes Classifier*

Given the following table of observations describing under which weather conditions person *A* was playing computer games.

Outlook	Temperature	Humidity	Play Computer Games
Sunny	Moderate	High	No
Sunny	High	Low	Yes
Rainy	Moderate	High	Yes
Rainy	High	High	No
Sunny	Moderate	Low	No
Sunny	Low	Low	No
Rainy	Low	Low	Yes

Outlook, *Temperature* and *Humidity* denote the observed features and *Play Computer Games* is the target variable.

Given the observation *o = (Outlook = Sunny, Temperature = High, Humidity = High)*, decide whether *A* is going to play computer games or not. Calculate the class probabilities by using the naive Bayes classifier.

Bayes Theorem: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$

Mapped to our problem: We need to calculate the probabilities for $P(PlayComputerGames = yes|o)$ and $P(PlayComputerGames = no|o)$. Or somewhat more formally: $P(y_i|X)$, with $y_i \in Y = \{yes, no\}$ being the target variable and X denoting the feature vector o.

To finally make a class decision, we want to take the class y_i which maximizes the probability, i.e.,

$$\hat{y} = argmax_{y_i \in Y}\{P(y_i|X)\} = argmax_{y_i \in Y}\{\frac{P(X|y_i) \cdot P(y_i)}{P(X)}\}$$

As the denominator[1] is constant (and it does not depend on the class) we can simply ignore this term such that we get

$$\hat{y} = argmax_{y_i \in Y}\{P(X|y_i) \cdot P(y_i)\}.$$

As a consequence we'll get a relative probability instead of a true probability, since the denominator was the normalization term.

However, applying the Bayes Theorem for $y_i = yes$:

$$P(PlayComputerGames = yes|O = Sunny, T = High, H = High) =$$
$$P(O = Sunny, T = High, H = High|PlayComputerGames = yes) \cdot P(PlayComputerGames = yes)$$

The probability $P(PlayComputerGames = yes)$ can be calculated from our previously observed instances. We simply take the number of observations for which $PlayComputerGames = yes$ and divide by the total number of observations, i.e.,

$$P(PlayComputerGames = yes) = \frac{3}{7}.$$

The tricky part is the first term, i.e.,

$$P(O = Sunny, T = High, H = High|PlayComputerGames = yes),$$

as this is a new observation for which we have no historical observations. Also, this likelihood is particularly hard to define as our feature vector X, resp. o, is high-dimensional and regarding the chain rule for solving joint probabilities, it might become expensive.

Naive Bayes overcomes this problem by making a rather strong assumption, namely the assumption that all features are independent from each other. This way, we can bypass the problem of calculating the joint probability and end up with

$$P(\mathbf{x}|y_i) \propto \prod_{j=0}^{d-1} P(x_j|y_i),$$

or in our case

$$P(O = Sunny, T = High, H = High|PlayComputerGames = yes) =$$
$$P(O = Sunny|PlayComputerGames = yes) \cdot P(T = High|PlayComputerGames = yes) \cdot$$
$$P(H = High|PlayComputerGames = yes).$$

The single product terms, i.e., the conditional probabilities, can be determined easily from our training data:

$$\begin{aligned} P(O = Sunny|PlayComputerGames = yes) &= \frac{1}{3} \\ P(T = High|PlayComputerGames = yes) &= \frac{1}{3} \\ P(H = High|PlayComputerGames = yes) &= \frac{1}{3} \end{aligned}$$

[1] $P(X)$ is the evidence, i.e., the probability that observation X occurs in nature. This is often hard or even impossible to determine.

Putting all together for class *PlayComputerGames=yes* we get

$$P(PlayComputerGames = yes|O = Sunny, T = High, H = High) =$$
$$P(O = Sunny, T = High, H = High|PlayComputerGames = yes) \cdot P(PlayComputerGames = yes) =$$
$$P(O = Sunny|PlayComputerGames = yes) \cdot P(T = High|PlayComputerGames = yes) \cdot$$
$$P(H = High|PlayComputerGames = yes) \cdot P(PlayComputerGames = yes) =$$
$$\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{3}{7} = \frac{3}{189}$$

Doing the same for class *PlayComputerGames=no* we get

$$P(PlayComputerGames = no|O = Sunny, T = High, H = High) =$$
$$P(O = Sunny, T = High, H = High|PlayComputerGames = no) \cdot P(PlayComputerGames = no) =$$
$$P(O = Sunny|PlayComputerGames = no) \cdot P(T = High|PlayComputerGames = no) \cdot$$
$$P(H = High|PlayComputerGames = no) \cdot P(PlayComputerGames = no) =$$
$$\frac{3}{4} \cdot \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{4}{7} = \frac{24}{448}$$

Since $\frac{24}{448} > \frac{3}{189}$, the classifier would decide that person *A* does not play computer games if the weather conditions are sunny, high temperature and high humidity.