Machine Learning (SS 23) Assignment 02: Classification

Team Members:

- Likhit Jain, 3678905, M.Sc. Computer Science
- Tareq Abu El Komboz, 3405686, M.Sc. Informatik
- Serge Kotchourko, 3309449, M.Sc. Informatik

Question 1: kNN for Data Classification

Assume that the training data is also used as the test data/set:

(1) kNN with k=1

The number of miss-classified points here is 0, as for each point in the test data, the point is also in the training data. Therefore, the euclidean distance will be zero between the points and the correct class/target/label will be assigned.

(2) kNN with k=3

First, we calculate the euclidean distance between each pair (Table 1):

points	-3	-1	1	7.2
-3	0	2	4	10.2
-1	2	0	2	8.2
1	4	2	0	6.2
7.2	10.2	8.2	6.2	0

Table 1: Euclidean distance metric between each point-pair

Now, for each point in the test set the three closest points are used to determined the classification of the point by majority voting (i.e. the point is classified as the most occurring label in its vicinity). The following Table 2 shows the resulting classification

attribute	x	-3	-1	1	7.2
target	$\hat{y}$	1	1	1	1

Table 2: Classification of test data by kNN with k=3

We can see, that the outer two points are miss-classified (i.e. 2 points are miss-classified, $x = -3$ and $x = 7.2$). This is due to the corresponding point with the same class being the most distant point and hence the resulting euclidean distance metric between those is high, leading to the miss-classification, as the nearest next points are of a different class.

Question 2: kNN and cross-validation

(1) leave-one-out cross-validation (LOOCV) error for kNN with k=1

We use LOOCV with accuracy as the performance metric and repeat the LOOCV 10 times (i.e. for each point in the set). Finally, we will calculate the average accuracy over all performances. As we only use one point in the validation/test set, the accuracy metric for each LOOCV run is $acc_i \in \{0, 1\}$, where $i \in \{1, ..., n\}$ is the run number and hence the average accuracy metric is calculated by:

$$acc_{avg} = \frac{1}{n} \sum_{i=1}^{n} acc_i$$

Note, for kNN we again use the euclidean distance metric for classification of points (we do leave out the calculation for classification here).

run i	1	2	3	4	5	6	7	8	9	10
left out point	$\begin{pmatrix}-1\\1\end{pmatrix}$	$\begin{pmatrix}-1\\2\end{pmatrix}$	$\begin{pmatrix}0\\2\end{pmatrix}$	$\begin{pmatrix}0\\3\end{pmatrix}$	$\begin{pmatrix}1\\-1\end{pmatrix}$	$\begin{pmatrix}1\\3\end{pmatrix}$	$\begin{pmatrix}2\\-1\end{pmatrix}$	$\begin{pmatrix}2\\0\end{pmatrix}$	$\begin{pmatrix}3\\0\end{pmatrix}$	$\begin{pmatrix}3\\1\end{pmatrix}$
acc_i	0	0	0	0	0	0	0	0	0	0

Table 3: LOOCV for kNN k=1 and accuracy error metric

The average performance here is $acc_{avg} = 0$.

(1) leave-one-out cross-validation (LOOCV) error for kNN with $k \in \{3, 5, 9\}$

Consider the above, we calculate the average performance for $k \in \{3, 5, 9\}$, where $acc_{i,k}$ describes the accuraccy og run i for kNN with k:

run i	1	2	3	4	5	6	7	8	9	10
left out point	$\begin{pmatrix}-1\\1\end{pmatrix}$	$\begin{pmatrix}-1\\2\end{pmatrix}$	$\begin{pmatrix}0\\2\end{pmatrix}$	$\begin{pmatrix}0\\3\end{pmatrix}$	$\begin{pmatrix}1\\-1\end{pmatrix}$	$\begin{pmatrix}1\\3\end{pmatrix}$	$\begin{pmatrix}2\\-1\end{pmatrix}$	$\begin{pmatrix}2\\0\end{pmatrix}$	$\begin{pmatrix}3\\0\end{pmatrix}$	$\begin{pmatrix}3\\1\end{pmatrix}$
$acc_{i,3}$	0	0	0	0	0	0	0	0	0	0
$acc_{i,5}$	0	0	0	0	0	0	0	0	0	0
$acc_{i,9}$	0	0	0	0	0	0	0	0	0	0

Table 4: LOOCV for kNN k=1 and accuracy error metric

The resulting average performance for the different $k \in \{3, 5, 9\}$ is $acc_{avg,3} = acc_{avg,5} = acc_{avg,9} = 0$. The main problem is, that for any point the "inverse" class holds the majority in its vicinity.

Note, Figure 1 is included for small showcase of the euclidean distance calculation and is mainly here for later study purposes.

Question 3: kNN for Image Classification

First, lets assume the most simplest case for images, a gray-scaled image, where each pixels color is described by one single value $c \in [0, 255]$. This allows for a nice representation as the following vector $\vec{c} = (c_0, c_1, \ldots, c_n)^T$, where c_i is the color for pixel i, if pixels were stored in a row. This allows to use the same "pipeline" of kNN classification as used in the two previous tasks (and next task) and

lecture, as an image is now just a high dimensional vector $[0, 255]^{h \cdot w}$, where h and w are the height and the width of the picture respectively. Meaning we are able to use euclidean distance to calculate the distance between pictures and any of the discussed decision rules for image classification with kNN, the simplest being a majority voting. For example, imagine a simple 2 by 2 image, where the top row consists of black pixels (i.e. value 0) and the bottom row of white pixels (i.e. value 255). This image can then be represented by the vector $\big(c\big) = (0, 0, 255, 255)^T$. Now, the euclidean distances between this picture and other 2 by 2 pictures can be calculated in the same manner as before (cf. Question 1), i.e. by $d(\big(p\big), \big(c\big)) = \sqrt{\sum_i^n (p_i - c_i)^2}$. Finally, the same classification approach can be applied as before, i.e. majority voting. This can be extended to pictures in color (e.g. RGB), by either calculating some (gray)value over each pixel i and using this value as c_i, or each *layer* (i.e. channel) can be seen as its own "picture" and a classification for each layer is made, over which then a final majority voting is held. There are a lot of problems with this approach, mainly, assume the same image to be classified, but the image is darkened in each pixel by some value y (i.e. new value $\tilde{c}_i = max(c_i - y, 0)$). If value y is high enough, the image might be miss-classified, even though the information of interest (e.g. animal in picture, face, ...) did not change!

Task 1: kNN and Classification

See Jupyter notebook

Question 3: Additional

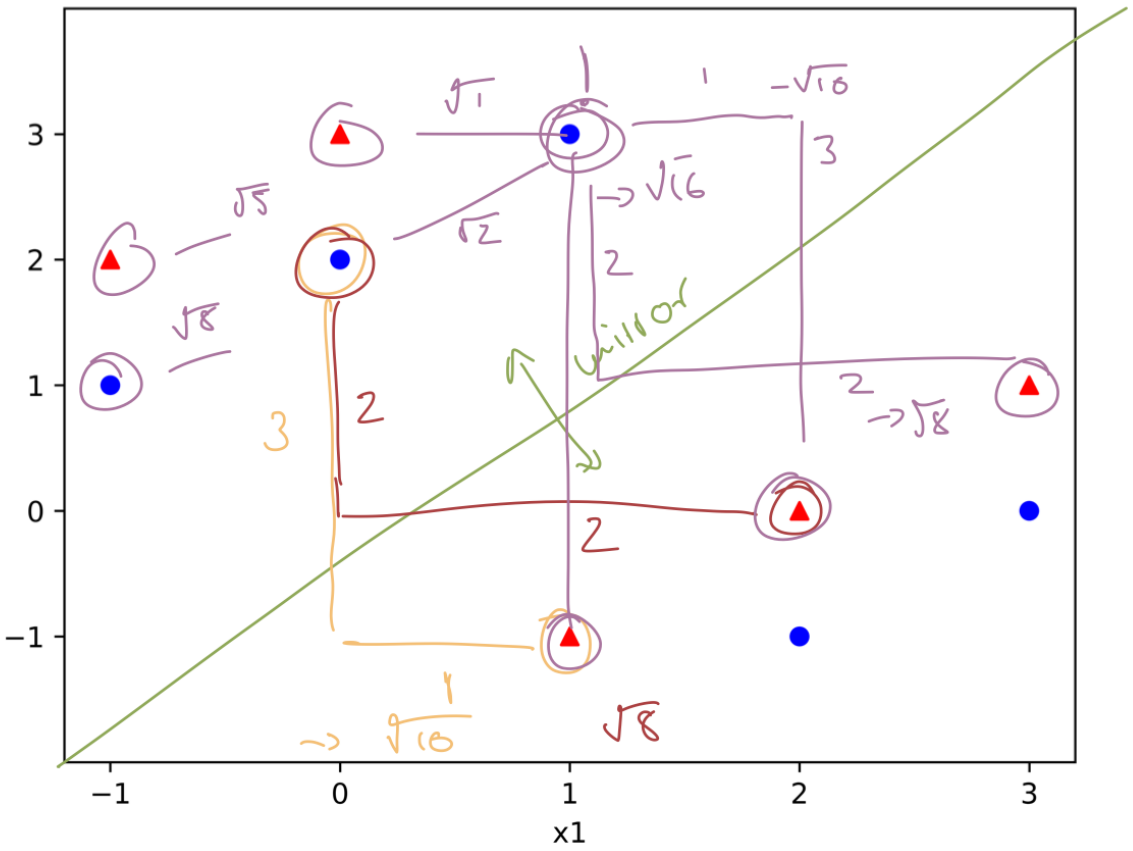

Figure 1: Example calculations of classification of points