AI Lab 2

Karl-Johan Djervbrant

January 3, 2020

Task 1

1.1 a)

	Algorithm	Accuracy (%)	Correct	Time (s)
Own KNN_Classifier_fast	euclidean	0.8525	341/400	4.76143
Own KNN_Classifier	euclidean	0.8525	341/400	54.7676
Sklearn	euclidean	0.8525	341/400	0.0145493

Table 1: Performance of own KNN algorithm vs sklearn implementation.

In this task we had to implement our own version of a KNN_Classifier. The Algorithm works by calculating the distance from one data point to another, and assigning the class to the new data point which the K-Nerest-Neighbors has.

The difference between `KNN_Classifier` and `KNN_Classifier_fast` is that the faster version only keep track of the K closest distances currently calculated, this means that you don't have to sort the whole list of calculated distances, (which increases every time you calculate a distance). With the faster algorithm, you only need to sort a list with K elements in it. This also saves on memory allocation.

As we can see in Table 1.1, both version of the own implementation perform equally good compared to the KNN-algorithm implemented by scikit-learn. The only difference is the execution time, where the scikit-learn implementation is much faster.

1.2 b)

	Score	Time (s)
SVC	0.9475	0.230703
Decision Tree	0.8800	0.005497
Random Forest	0.9100	0.013979

Table 2: Performance between three classification algorithms, higher score is better.

In this task we got to implement and compare three different algorithms from the scikit-learn library. I choose to test the performance on Scalar-Vector-Classifier (SVC), Decision Tree Classifier and Random Forest Classifier. From the data in Table 1.2 we can see that they perform good on different measures, the SVC algorithm does the best prediction but is also the slowest. The Decision Tree algorithm is the worst one, but also the fastest.

1.3 c)

To change the distance algorithm used, you initialize the constructor with the parameter p set to a specific value. $p = 1$ is for Manhattan-distance and $p = 2$ is for Euclidean-distance. If you set p to another value you can calculate other types of distances, the algorithm used is the Minkowski-distance algorithm [**?**]. In Table 1.3 we can se the difference in performance depending on distance algorithm and amount of neighbors. On this data set the Manhattan-distance perform slightly better than the Euclidean-distance, the best one is with $K = 7$.

Classifier	Neighbors	Distance Measure	Score
KNN	3	Manhattan	0.73 (+/-0.45)
KNN	5	Manhattan	0.73 (+/-0.45)
KNN	7	Manhattan	0.74 (+/-0.45)
KNN	9	Manhattan	0.71 (+/-0.47)
KNN	11	Manhattan	0.73 (+/-0.48)
KNN	3	Euclidean	0.70 (+/-0.45)
KNN	5	Euclidean	0.71 (+/-0.48)
KNN	7	Euclidean	0.70 (+/-0.50)
KNN	9	Euclidean	0.68 (+/-0.48)
KNN	11	Euclidean	0.67 (+/-0.48)
KNN	3	Minkowski p=$2^{1.5}$	0.68 (+/-0.47)
KNN	5	Minkowski p=$2^{1.5}$	0.70 (+/-0.49)
KNN	7	Minkowski p=$2^{1.5}$	0.70 (+/-0.47)
KNN	9	Minkowski p=$2^{1.5}$	0.67 (+/-0.48)
KNN	11	Minkowski p=$2^{1.5}$	0.67 (+/-0.47)

Table 3: KNN-Regression with different amount of neighbors and distance algorithms

1.4 e)

	Algorithm	Score	Time (s)
Own KNN_Regressor_fast	euclidean	0.395646	9.06785
Own KNN_Regressor	euclidean	0.395646	78.4536
Sklearn	euclidean	0.395646	0.00223327

Table 4: Performance of own KNN-Regression algorithm vs sklearn on estimating Fuel Consumption.

As we can see in the Table 1.4, KNN-Regression isn't working too well on estimating the fuel consumption in this data set. This is partly due to that KNN is sort of a improved look-up table and on data which is

1.5 f)

	Score	Time (s)
SVR	0.768634	3.850483
Decision Tree Regressor	0.754809	0.007603
Random Forest Regressor	0.608807	0.023281

1.6 g)

Classifier	Neighbors	Distance Measure	Score
KNN	3	Manhattan	-0.31 (+/-2.87)
KNN	5	Manhattan	-0.27 (+/-2.94)
KNN	7	Manhattan	-0.17 (+/-2.61)
KNN	9	Manhattan	-0.15 (+/-2.46)
KNN	11	Manhattan	-0.11 (+/-2.38)
KNN	3	Euclidean	-0.46 (+/-3.36)
KNN	5	Euclidean	-0.35 (+/-3.02)
KNN	7	Euclidean	-0.25 (+/-2.65)
KNN	9	Euclidean	-0.23 (+/-2.61)
KNN	11	Euclidean	-0.22 (+/-2.55)
KNN	3	Minkowski p=$2^{1.5}$	-0.50 (+/-3.33)
KNN	5	Minkowski p=$2^{1.5}$	-0.35 (+/-3.07)
KNN	7	Minkowski p=$2^{1.5}$	-0.26 (+/-2.77)
KNN	9	Minkowski p=$2^{1.5}$	-0.25 (+/-2.70)
KNN	11	Minkowski p=$2^{1.5}$	-0.26 (+/-2.74)

Task 2

2.1 a)

2.2 b)

Classifier	Score	Time (s)
KNN	0.75	0.000464
SVC	0.95	0.017687
Decision Tree	1.00	0.000918

2.3 c)

Classifier	Neighbors	Distance Measure	Score
KNN	2	Manhattan	0.70 (+/-0.30)
KNN	3	Manhattan	0.75 (+/-0.22)
KNN	4	Manhattan	0.72 (+/-0.29)
KNN	5	Manhattan	0.80 (+/-0.25)
KNN	6	Manhattan	0.70 (+/-0.20)
KNN	7	Manhattan	0.70 (+/-0.34)
KNN	8	Manhattan	0.62 (+/-0.27)
KNN	9	Manhattan	0.60 (+/-0.29)
KNN	10	Manhattan	0.50 (+/-0.27)
KNN	11	Manhattan	0.45 (+/-0.12)
KNN	12	Manhattan	0.47 (+/-0.19)
KNN	2	Euclidean	0.65 (+/-0.29)
KNN	3	Euclidean	0.75 (+/-0.22)
KNN	4	Euclidean	0.72 (+/-0.37)
KNN	5	Euclidean	0.70 (+/-0.30)
KNN	6	Euclidean	0.65 (+/-0.19)
KNN	7	Euclidean	0.57 (+/-0.25)
KNN	8	Euclidean	0.45 (+/-0.25)
KNN	9	Euclidean	0.40 (+/-0.19)
KNN	10	Euclidean	0.38 (+/-0.16)
KNN	11	Euclidean	0.42 (+/-0.12)
KNN	12	Euclidean	0.40 (+/-0.10)
KNN	2	Minkowski p=$2^{1.5}$	0.60 (+/-0.37)
KNN	3	Minkowski p=$2^{1.5}$	0.70 (+/-0.20)
KNN	4	Minkowski p=$2^{1.5}$	0.72 (+/-0.37)
KNN	5	Minkowski p=$2^{1.5}$	0.65 (+/-0.33)
KNN	6	Minkowski p=$2^{1.5}$	0.60 (+/-0.40)
KNN	7	Minkowski p=$2^{1.5}$	0.60 (+/-0.19)
KNN	8	Minkowski p=$2^{1.5}$	0.50 (+/-0.00)
KNN	9	Minkowski p=$2^{1.5}$	0.42 (+/-0.25)
KNN	10	Minkowski p=$2^{1.5}$	0.40 (+/-0.10)
KNN	11	Minkowski p=$2^{1.5}$	0.42 (+/-0.12)
KNN	12	Minkowski p=$2^{1.5}$	0.40 (+/-0.19)