Report Lab01

Emanuele Falci, Simone Varriale

03/11/2023

1 Introduction

In this lab we "trained" a K-NN model to perform classification. This is an instance based model, so it is not a real training but the classification is determined by the points present in the training set, hence the training phase consists only on loading data. The label is attribuited by votation of the K nearest points to the one we are classifing, where the distance is defined as Minkowski distance with parameter p: $D(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$.

2 Validation set and choose of k parameter

Once the validation set has been imported, we calculate the distance matrix, wich contains the distance between each point of the validation set to each one of the training set. Then through a sorting phase we obtain the K nearest point of the training set to each validation point, finally the more the present class is attributed to each validation point.
In order to perform hyperparameters tuning we repeat the classification on the validation set with different models, basically this operation is done varying k and p. The performance of the different models is evaluated through the accuracy, although is not always a reliable metrics for example with unbalanced dataset, checking wheter or not the model assigned the validation point to the correct class. In our final choice we decided to take into account also the time needed to perform classification by the different models, because if a model perform slightly better than another, but it takes a lot of time to gives us the result we can also prefer to have a faster model.
It seems that the most accurate model has parameters $k = 23$ and $p = 5$ with an accuracy of 82.7%, so it is our choice as final model due the best result. However, if we seek for a faster model, we can use the parameters $k = 27$ and $p = 2$ which has a sligthly lower accuracy (82.5%). A part of the results of the test with different model are in the appendix *(figure 1)*.

3 Curse of dimensionality

The K-NN model works well only if we have few dimensions, when the dimensional space grows we encounter the problem of the course of dimensionaility. When the dimensionality grows, the volume of the space increases so fast that samples data become sparse. In fact, the amount of sample data that we need to obtain a reliable result grows exponentially with the dimensionality, precisely the density is proportional to $N^{1/p}$.
Another problem that occours is that most of the data points are closer to the boundary rather than to other data points. This means that there is not an even space around them to search for K neighbours and that could cause problems to determine correctly the class of the point.
Consider the nearest-neighbor procedure for inputs uniformly distributed in a D-dimensional unit hypercube, if we want to capture a fraction r of it through a hypercubical neighborhood we need an edge length of $e(r) = r^{1/D}$. In our case, the ratio is given by the number of neighbors (k) over the total number of data points (N), so $r = k/N$. Joining with the formula above we obtain that $e(r) = (k/N)^{1/D}$.
We have sampled our training data from a D-dimensional hypercube for which the entire input range is 1. Then, through the graph *(figure 2)* it is possible to check what percentage of the range of the input we need to cover if we want to capture 1% of our sample data (10/1000) to perform a local average and apply k-nn. It is possible to notice that if the dimension are few the data are dense and we can cover just around 20% of each dimension for 2 or 3 dimension. After 5 dimension there is a fast increase of the requested input range for each dimension. After the 20-th dimension we need at least 80% coverage of the input space thus a lot of sample datas. The function is exponenntially increasing as the number of dimensions grows.

4 Images

	A	B	C	D
1	k	p	accuracy	time
2	23	5	82.7083333333333	0.115674257278442
3	24	4	82.5	0.221186637878418
4	24	5	82.5	0.17973256111145
5	26	4	82.5	0.225013971328735
6	26	5	82.5	0.162299394607544
7	26	6	82.5	0.162532329559326
8	26	7	82.5	0.173702478408813
9	26	8	82.5	0.11090087890625
10	26	9	82.5	0.100701093673706
11	27	2	82.5	0.0400567054748535
12	27	3	82.5	0.10833477973938
13	21	7	82.2916666666667	0.126822233200073
14	21	8	82.2916666666667	0.124855041503906
15	22	9	82.2916666666667	0.126926183700562
16	22	10	82.2916666666667	0.116991758346558
17	23	4	82.2916666666667	0.11767053604126
18	23	6	82.2916666666667	0.123497009277344
19	24	6	82.2916666666667	0.214393854141235
20	26	10	82.2916666666667	0.096060037612915

Figure 1: K-NN performance results

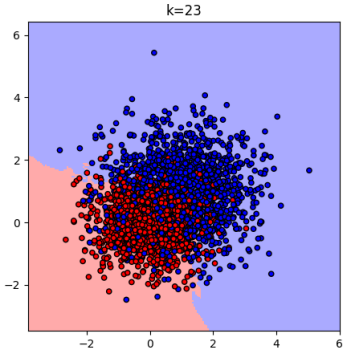

Figure 2: K-NN decision boundaries w/ k = 10

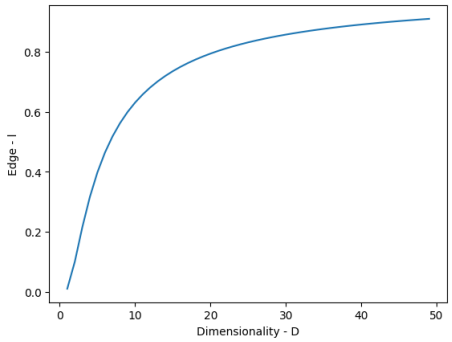

Figure 3: Edge length in function of Dimensionality