PTML 8: 09/06/2022

TABLE DES MATIÈRES

1 CURSE OF DIMENSIONALITY AND ADAPTIVITY TO A LOW DIMENSIONAL SUPPORT

1.1 General case : curse of dimensionality

1.1.1 *Problem statement*

We consider the problem of a nearest neighbors estimation. We will use $\mathcal{X} = [0,1]^d$ as input space. Hence, here the dimensionality of the problem is d.

In the general case, we have seen in the class that if the target function f^* is Lipshitz-continuous, and if the n samples in the dataset are on a lattice that divides $[0,1]^d$ in cubes of equal size, then it is possible to show that the error made by a k-NN esimator is $\mathcal{O}(n^{-1/d})$.

Note that this is a rough bound. It is possible to have more subtle results, and the constant in the $\mathcal{O}$ depends on d and on the Lipshitz constant of the target function However, it is not possible to have a bound that is "way better" (i.e. "way smaller") than this bound.

This means that if d is large, then $1/d$ is small, and the decrease of the error is very slow. To be more precise, in order to reach an error ϵ, it is possible to show that the number of sampled needed n_ϵ verifies

$$n_\epsilon \geqslant \frac{\epsilon^{-d} d^{d/2}}{\alpha^{d/2}} \tag{1}$$

where $\alpha >$ is a constant. n_ϵ hence grows exponentially with $1/\epsilon$, and this is an example of curse of dimensionality.

1.1.2 *Simulation*

To illustrate this phenomenon, we will learn a estimator of the euclidean norm. This means that our target function f^* verifies $f^*(x) = \|x\|_2$, for all x. Of course, as we already know the target function, the goal of these exercise is to illustrate important aspects and limitations of the k-NN estimation.

Exercice 1 : Run a simulation that illustrates the curse of dimensionality by computing the error of a nearest neighbor estimator of f^*, for several numbers of samples availables and several dimensions d. You can use several values of k, the number of nearest neighbors used, e.g. $k = 2$.

Note that as we know f^*, it is possible to perfeclty evaluate the error made by the k-NN algorithm for each sample.

For instance, you can observe a picture like the one in figure 1. In the simulation that generated the figure, the samples (both the dataset and the test samples) were drawn randomly in $\mathcal{X}$, with a uniform distribution (the dataset is not on a regular lattice, although you can do use a regular lattice in your simulation).

1.1.3 *Python files*

You can use **knn_1.py**, and edit the function **predict()**, that implements the k-NN algorithm.

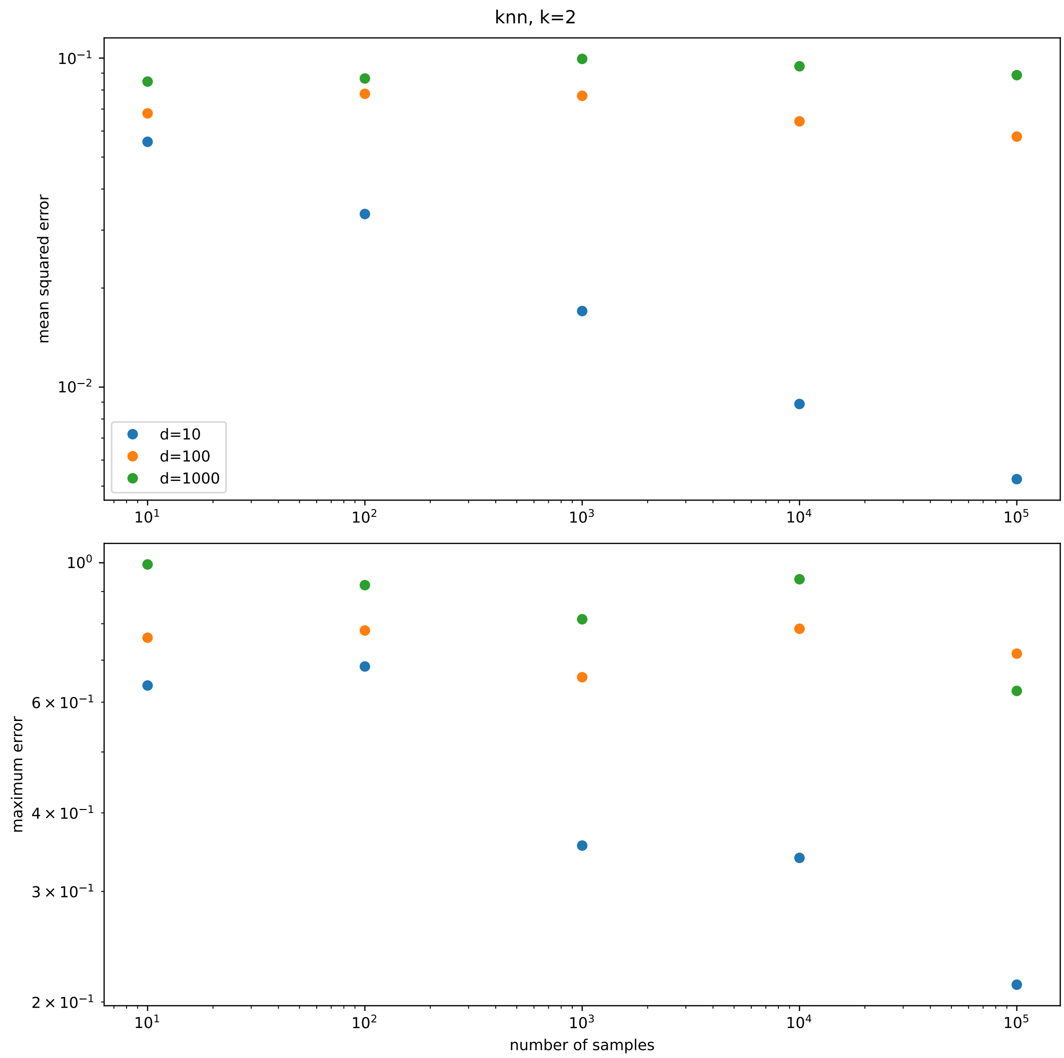

FIGURE 1 – Curse of dimensionality. We observe that the logarithm of the error is linear in the logarithm of the number of samples, with a slope that depends on the dimension d.

1.2 Adaptivity

1.2.1 *Setting*

Now, we would like to do a k-NN estimation on the same target function (f^* is still the squared norm), but on a different dataset. We now use $d = 30$ and $\mathcal{X} = [0, 1]^{30}$.

Exercice 2 : Perform a k-NN estimation using the data stored in **data_knn/** and use a varying number of samples n that you use from the dataset. Plot the error as a function of n.

1.2.2 *Files*

- You can start from **knn_2.py** and import the relevant functions from the previous file.
- **x_data_.npy** and **y_data_.npy** : dataset that is used by the k-NN algorithm.
- **x_data_test.npy** : data on which we estimate using the k-NN algorithm.

You should observe something like figure 2.

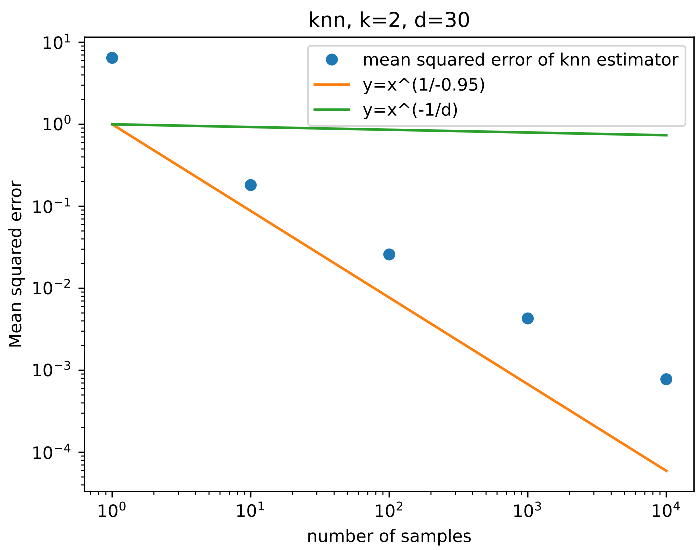

FIGURE 2 – Nearest neighbors estimation on a different dataset.

Exercice 3 : The learning rate is way better than $\mathcal{O}(n^{-1/d})$.

- What could be an explanation of this phenomenon ?
- How could we test this hypothesis ? (we have seen a way during the class)