TD4: K-nearest neighbors

- $\mathcal{Y}$ is the set of labels. Here we consider L classes, and we choose $\mathcal{Y} = \{1, \dots, L\}$ to represent the L possible labels. Binary classification corresponds to $L = 2$.
- $\mathbf{x} = (x_1, \dots, x_p)^\top \in \mathcal{X} \subset \mathbb{R}^p$ is one observation, one example, one point, one sample.
- $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \dots n\}$ is the full dataset with the samples and their labels.
- We consider a probabilistic model governing random variables X and Y: $\forall i \in \{1, \dots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$.
- We seek to construct from $\mathcal{D}_n$ a function, named classifier, $\hat{f} : \mathcal{X} \to \mathcal{Y}$ which to a new point $\mathbf{x}_{\text{new}}$ gives a label $\hat{f}(\mathbf{x}_{\text{new}})$.

Formal setting

k-nn is an intuitive algorithm. Its principle is as follows: for each new point, $\mathbf{x}$ we first find its k- nearest neighbors in the training dataset, denoted $V_k(\mathbf{x})$ The class given to the new point is then the class which is the most represented in $V_k(\mathbf{x})$. We first choose a distance $d : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$. For a new point $\mathbf{x}$, we define the k-nearest neighbors $V_k(\mathbf{x})$ in the sense of this distance. An illustration is given in Figure 1 for $L = 3$ classes.

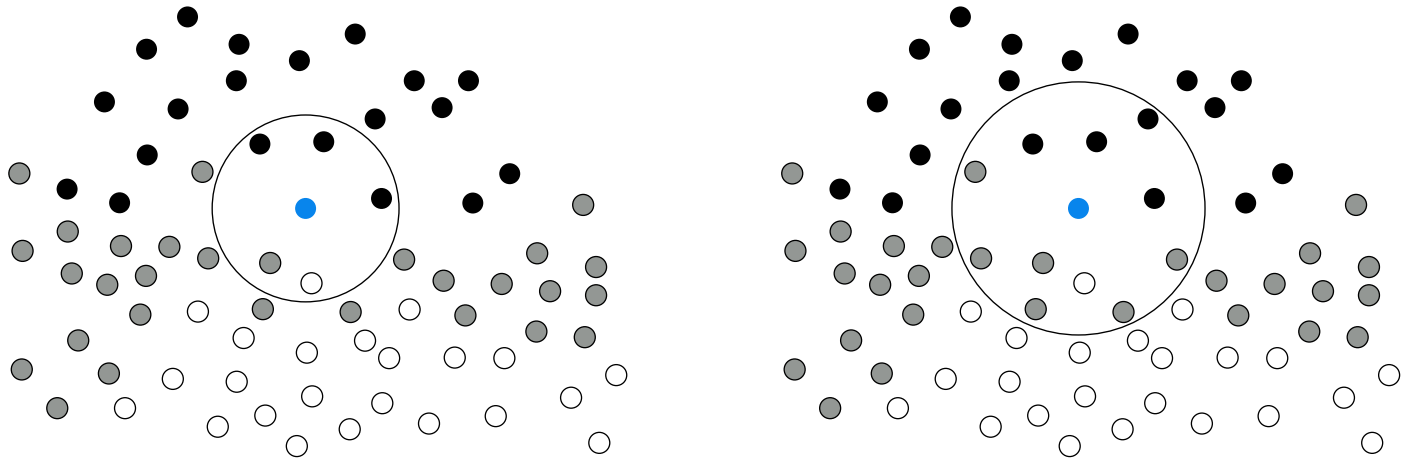

Figure 1: Exemple of k-nn for $k = 5$ and $k = 11$, with $L = 3$ classes represented in black ($y = 1$), grey ($y = 2$) and white ($y = 3$). For $k = 5$ (left) we predict black at the new point $\mathbf{x}$ (in blue), while for $k = 11$ (right) we predict grey.

1. Propose a formal definition of the neighbors of x using the distance function d.
2. Assume we perform classification with a finite set of labels $\mathcal{Y}$. Propose a formal definition of the K-NN prediction function $f : \mathbb{R}^d \to \mathcal{Y}$.
3. Propose an adaptation of this algorithm to regression, that is, the case where $\mathcal{Y} = \mathbb{R}$.
4. One of the drawbacks of this formulation is that all neighbors are equal regardless of how far they are. Propose a variant of your formal definition to mitigate this.

Implementation

This section should be done on the associated jupyter notebook KNN.ipynb.

1. Generate data samples from the 4 different samplers in the first cell of the notebook and visualize them using scatter plots in the same figure.
2. Fill the KNNClassifier to reimplement the decision function you defined above. Check your predictions by comparing them to the results of + of scikit-learn, on one, then all of the toy datasets introduced above. For quicker computation, you will now use the functions from scikit-learn instead of your implementation. From now on you should use the checkers dataset, unless specified otherwise.
3. Vary k. What happens when $k = 1$? $k = n$? Plot these cases on one dataset. When is the decision frontier simple? complicated?
4. What is the fraction of errors on your training data when $k = 1$? and on test data ?
5. Plot the error curves as a function of k on one of the datasets for n taking values 100, 500 and 1000. What is the best k? Is it the same for all datasets ? Be careful to evaluate the error on the testing data. You can use the class ErrorCurve.
6. What are the pros and cons of this classifier? What are the pros and cons of this way of tuning K?
7. Apply this method the the DIGITS dataset with different choices of $k \geq 1$. Refer to https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html to load the data.
8. Compute the confusion matrix $(\mathbb{P}\{Y = i, C_k(X) = j\})_{i,\,j}$ associated to your classifier C_k. Refer to https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html.
9. Propose a method to choose k and implement it. You can use LOOCurve.
10. How can you add the idea proposed in Q4 of the previous section within scikit-learn's implementation ?

To go further

Global details available at [HTF09, Chapitre 13]. For a theoretical understanding of the method, refer to [DGL96, Chapitre 11], and for the limits of the method when $k = 1$, http://certis.enpc.fr/%7Edalalyan/Download/DM1.pdf. Finally, for algorithmic considerations, one can read http://scikit-learn.org/stable/modules/neighbors.html#brute-force and following paragraphs.

References

[DGL96] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1996. 2

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 2