K nearest neighbors

Fabrice Rossi

The goal of the following exercises it to implement the k nearest neighbors algorithm and experiment with it.

Data sets

Data sets used in the exercises are available on the course page.

Artificial 2 dimensional data set

The `2d` data set zip file contains:

- `2d-learn.csv`: a learning set with 10000 examples described by 2 numerical variables and a target variable `Y`;
- `2d-eval.csv`: a test set with 50000 examples described by 2 numerical variables and without the target variable;
- `2d-eval-labels.csv`: the labels for the test set, with 2 variables. `Y` is the target variable while `Yopt` is the decision for an optimal model obtained using the true data distribution.

Exercise 1 (*Naive k-nn implementation*)

In this exercise, the goal is to implement completely the k-NN algorithm even if the implementation is somewhat naive (in terms of computational efficiency).

Question 1 Write a function `distance` with two parameters `a` and `b` and which returns the squared Euclidean distance between the parameters, assuming they are `numpy` vectors.

Question 2 Write a function `m_distance` with two parameters `A` and `b` and which returns the squared Euclidean distance between the rows of the `A` matrix and the vector `b`. The use of numpy broadcasting is recommended (no loop is required).

Question 3 Write a function `k_closest` with three parameters `A`, `b` and `k` which returns the indices (row numbers) of the `k` rows of the matrix `A` that are the closest to the vector `b`. It is recommended to use the function `numpy.argsort` which returns the indices that would sort its argument.

Question 4 Write a function `mode` with a parameter `a` which finds and returns the most frequent value in `a`.

Question 5 Write a function `knn` with four parameters `A`, `y`, `b` and `k` with returns the output of the k-NN model constructed on `A` (input vectors of the learning set) and `y` (output labels of the learning set), evaluated at `b`.

Question 6 Write a function `mknn` with four parameters `A`, `y`, `B` and `k` with returns the outputs of the k-NN model constructed on `A` (input vectors of the learning set) and `y` (output labels of the learning set), evaluated on all the rows of the `B` matrix.

Question 7 Write a function `confusion` with two parameters `a` and `b` which outputs the confusion matrix associated to predicting `a` while the true output is `b`. The `pandas.crosstab` function is recommended for this calculation.

Question 8 Load the `2d-learn.csv` data set in a pandas DataFrame and compute the confusion matrix of the model built by the k-NN model constructed on the data set and evaluated on the same data set, for several values of k (including $k = 1$).

Exercise 2 (*Improving the implementation*)

The naive implementation constructed in the previous exercise is too slow to be usable in practice on realistic data sets. We study in this exercise a faster solution.

> **Lecture notes**
>
> The SciPy collection of tools contains the NumPy library but also the SciPy library. The later contains numerous useful functions to complement NumPy for scientific calculations. For the k-NN implementation, several functions can be used.
>
> The module `scipy.spatial.distance` provides a function `cdist` which takes as arguments two matrices and computes all the pairwise distances between the rows of the matrices (interpreted as vectors). A third argument can be used to select other distances than the Euclidean distance. A typical use is
>
> ```
> import scipy.spatial.distance as ssd
> A2B = ssd.cdist(A, B, 'euclidean')
> ```
>
> After the execution of those lines, `A2B[i,j]` contains the Euclidean distance between `A[i,]` and `B[j,]`.
>
> In addition the module `scipy.stats` contains numerous statistical functions, among which the `mode` function that compute the most frequent value in a vector. It can be applied to any array provided the axis over which the calculation is done is specified.

Question 1 Write a function `mk_closest` with three parameters `A`, `B` and `k` which returns a matrix `C` such that `C[i,]` contains the indices of the rows of `A` that are the `k` closest to `B[i,]` (according to the Euclidean distance).

Notice that the `numpy.argsort` function can be applied to an array, provided the axis over which the calculation is done is specified.

Question 2 Write a function `mknn` with four parameters `A`, `y`, `B` and `k` with returns the outputs of the k-NN model constructed on `A` (input vectors of the learning set) and `y` (output labels of the learning set), evaluated on all the rows of the `B` matrix.

Question 3 Load the `2d-learn.csv` data set in a pandas DataFrame and compute the confusion matrix of the model built by the k-NN model constructed on the data set and evaluated on the same data set, for several values of k (including $k = 1$).

Lecture notes

The `time` module contains functions for date and duration manipulation. In particular the `time.time` function return the number of seconds since a reference time (in general January the 1st, 1970). While this is only for crude time measurement, this function can be used to evaluate the run time of a long running code. For instance

```
import time
before = time.time()
print(min(range(1000000)))
after = time.time()
print(after - before)
```

can print something like

```
0
0.02694082260131836
```

The second value depends on the computer but it corresponds to the time in seconds between the two calls to `time.time`.

Question 4 Compare the running time between the two versions of `mknn`.