Machine Learning for Physics and Astronomy: Exercises

David Sergio

July 1, 2025

Problem 2.6.2

Code up kNN from scratch, using the Euclidean distance and uniform weights. [1]

Background

A k-nearest neighbors (kNN) algorithm is a lazy learning algorithm. [2]

Euclidean distance for an n-dimensional vector is defined as:

$$\texttt{distance} = \sum_{i=0}^{n} \sqrt{(a_i - b_i)^2} \tag{1}$$

Dataset

The datasets used in this exercise include a training dataset of N samples, each with d features, and a test dataset of M samples. Each sample is represented as a vector in an n-dimensional space.

Let's define a simple training dataset with $N = 5$ samples and $d = 2$ features (x and y):

i	x	y
0	2.0	2.0
1	1.0	5.0
2	3.0	4.0
3	7.0	2.0
4	1.0	6.0

Table 1: Training dataset

And let's define a simple test dataset with $M = 1$ sample:

x	y
6.0	8.0

Table 2: Test dataset

The distance of the test sample to each training sample can be calculated using the Euclidean distance formula, then sorted according to the distance to the test data(ascending).

i	x	y	distance	indices	indices (sorted)
0	2.0	2.0	7.2	0	2
1	1.0	5.0	5.83	1	4
2	3.0	4.0	5.0	2	1
3	7.0	2.0	6.08	3	3
4	1.0	6.0	5.38	4	0

Table 3: Training dataset

Then, if we take the $k = 3$ nearest neighbors, we can see that the indices of the nearest neighbors of (6.0, 8.0) are $2, 1, 4$. The corresponding samples are:

- Sample 2: (3.0, 4.0)
- Sample 4: (1.0, 6.0)
- Sample 1: (1.0, 5.0)

Algorithm

```java

package ml;

public class KNN {

        public int k;

        public KNN(int k) {
                this.k = k;
        }

        public static double distance(double[] a, double[] b) {
                double sum = 0.0;
                for (int i = 0; i < a.length; i++) {
                        sum += Math.pow(a[i] - b[i], 2);
                }
                return Math.sqrt(sum);
        }

        public int[] predict(double[][] trainData, double[] testData) {
                int n = trainData.length;
                double[] distances = new double[n];
                for (int i = 0; i < n; i++) {
                        distances[i] = distance(trainData[i], testData);
                }

                int[] indices = new int[n];
                for (int i = 0; i < n; i++) {
                        indices[i] = i;
                }

                sort(indices, distances);

                int[] neighbors = new int[k];
                for (int i = 0; i < k; i++) {
                        neighbors[i] = indices[i];
                }

                return neighbors;
        }

        public static void sort(int[] indices, double[] distances) {
                for (int i = 0; i < distances.length - 1; i++) {
                        for (int j = i + 1; j < distances.length; j++) {
                                if (distances[indices[i]] > distances[indices[j]]) {
                                        int temp = indices[i];
                                        indices[i] = indices[j];
                                        indices[j] = temp;
                                }
                        }
                }
        }

}

```

Testing code:

```

    package ml;

public class Tester {

        public static void main(String[] args) {

                int d = 2;
                int k = 3;
                int n = 5;

                KNN knn = new KNN(k);

                double[][] trainData = {
                                {2.0, 2.0},
                                {1.0, 5.0},
                {3.0, 4.0},
                {7.0, 2.0},
                {1.0, 6.0}
                };

                double[] testData = {6.0, 8.0};

                int[] predictions = knn.predict(trainData, testData);

                System.out.println("Predictions: ");

                for (int i = 0; i < predictions.length; i++) {
                        System.out.print("trainData[" + predictions[i] + "] = ");
                        for (int j = 0; j < d; j++) {
                                System.out.print(trainData[predictions[i]][j]);
                                if (j < d - 1) {
                        System.out.print(", ");
                    }
                        }
                        System.out.println();


                }


        }

}
```

References

[1] V. Acquaviva. *Machine Learning for Physics and Astronomy*. Cambridge University Press, 2025.

[2] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2012.