USING OF THE K NEAREST NEIGHBOURS ALGORITHM (k-NNs) IN THE DATA CLASSIFICATION

Gîlcă Natalia, *Roşia de Amaradia Technological High School*, Gorj, ROMANIA
Gîlcă Gheorghe, *"Constantin Brîncuşi" University from Tîrgu*-Jiu, ROMANIA

ABSTRACT: The article aims to implement a data classification algorithm based on the nearest k neighbors (k-NNs). To begin we propose a series of improvements to the k-NNs algorithm that can lead to the optimization of the algorithm. The simulation results in the R2012b Matlab environment highlights the accuracy in the data classification for the k-NNs algorithm. In the simulation we have considered three different classes having a different number of components: 22, 35, 45, and a value of k = 10 was taken. We have considered three classified points and the output of the system has very good results, assigning for each point one of the 3 classes which contains most neighbors from the total.

KEYWORDS: k-Nearest Neighbours, classification, facial expressions, training, prototype vector, weight

1. INTRODUCTION

The k nearest neighbours algorithm (kNNs) is the simplest method of classification based on instances.
The algorithm assumes that all instances are placed in a n-dimensional space. Each test instance is classified according to the nearest k examples from memory, without trying to determine the distribution function on the basis of which the instances are placed in n-dimensional space [1], [3] - [6].
To find and remove from memory these examples, it needs a way to compare how similar two instances are. If working with real values metrics like the Euclidean distance, the normalized distance, or the cosine function between instances can be applied . For the situations where the values are discrete, the number of distinct characteristics between the two instances is calculated.

After defining a metrics for determining similarity between two instances, the next step is to calculate the distance between the test example and all learning examples stored in memory. In the final step for each test instance is determined what are the closest k instances from the memory set and based on these similar instances, it is decided which is its most probable classification .
This is achieved through the voting between the most similar k instances.
For the example from figure 1, new case x_q is assigned to the class to which the majority of cases belong from the total cases k, here being five case, considered the closest from the training set. Thus the new case is assigned to the class to which the most similar cases belong, respectively the class labeled with -, because in the ellipse there are three elements belonging to it, and only two items belonging to the class labeled with +.

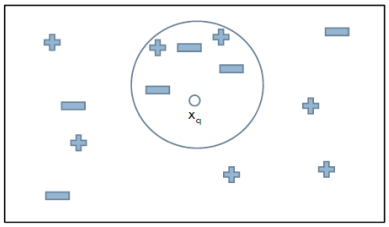

Figure 1. The 5-NN classification example

2. THE k-NNs CLASSIFICATION ALGORITHM

The k-NNs algorithm consists of the following steps [2]:

1) k prototype vectors w_i , i = 1...k is chosen with the same dimension as the input vector;
   a) k corresponds to the number of groups (clusters);
   b) there is a vector corresponding to each neuron prototype; these neurons are called competitive neurons;
   c) the vectors prototypes are also called "codebook" or simply weight vectors.
2) an input vector p is chosen and it calculates the distance between p and all prototype vectors (competitive neurons);
3) the index i^* of the nearest competitive neuron is found, then its activation output is set to 1 and the activation of all other competitive neurons is set to 0;
   a) the neuron i* won the competition;
4) the weight vector only of the winner neuron is adapted using Kohonen's rule:

$$w_{i^*}(q) = w_{i^*}(q-1) + \eta\left(p(q) - w_{i^*}(q-1)\right) \quad \textbf{(1)}$$

, where η is learning rate, $\eta \in (0,1)$

5) another vector input is selected and then goes to step 2;
6) At the end time, it checks if the stop criteria are satisfied:
   a) There aren't changes in the position of weight vectors (e.g. each weight vector becomes a prototype for a different cluster) or
   b) The maximum number of epochs is reached.

3. THE IMPROVING OF THE K-NNS CLASSIC ALGORITHM

This algorithm is resistant to the noise that may occur in the training data and is very efficient if it is provided with a set of large enough training data. The k-NNs algorithm is a very efficient inductive inference method for many practical problems. One of the biggest problems of KNN algorithm is failing to provide informations about what value should be chosen for k so as to obtain the best results.

Another problem of the k-NNs algorithm is the weight which the instances have in the calculating of the final result. The question is whether all instances of the training set should have the same weight.

The last problem considered in this paper was that of the average distance between the test instance and the instances from the training set.

The small values of k allows a more efficient identification of small structures in the problem space, but can lead to a very complex model (overfitting). In addition, if small values of k are considered, then the results are significantly influenced by the incorrect data (noisy data).

If high values for k are chosen , it avoids the problem of noise in the training data, and also the probabilities can be better estimated if working with the discrete values.

The problem of weighting consist of finding out if the training data should have different weights in expected results.

This means giving a greater importance (translated by a higher weight) to the instances which are closer to the test instance and a lesser importance

(translated by a lower weight) to the instances that are far from it.
The approach in this paper solves by default this problem by consideration of the multiple instances which are closer to the test instance. While the instance is closer to the test instance, its value will be considered in more applications of the kNNs algorithm with different values of k. This way, the instances wich are closest of the test instance take on greater weights than others in the end, when deciding which will be the value of the test instance.
A final problem is to identify to what extent the consideration of the distance in the weighting of results is useful. The idea that generated the possibility of considering it as a weighting factor was that if we obtain a small value of the average distance between the test instance and the training set, we are dealing with a more compact structure than if this value is high, and therefore the structure could be closer to reality.

4. THE RESULTS OF THE SIMULATION

We used in the Matlab R2012b simulation installed on a PC with the configuration: Intel Celeron Dual-Core T3500 and 4GB RAM memory. Figure 2 shows the three classes of labeled data: setosa, versicolor and virginica, each having a different number of elements, in order: 22, 35, 45.
Figure 3 shows data classes, and the points of classification, marked with x of black color.

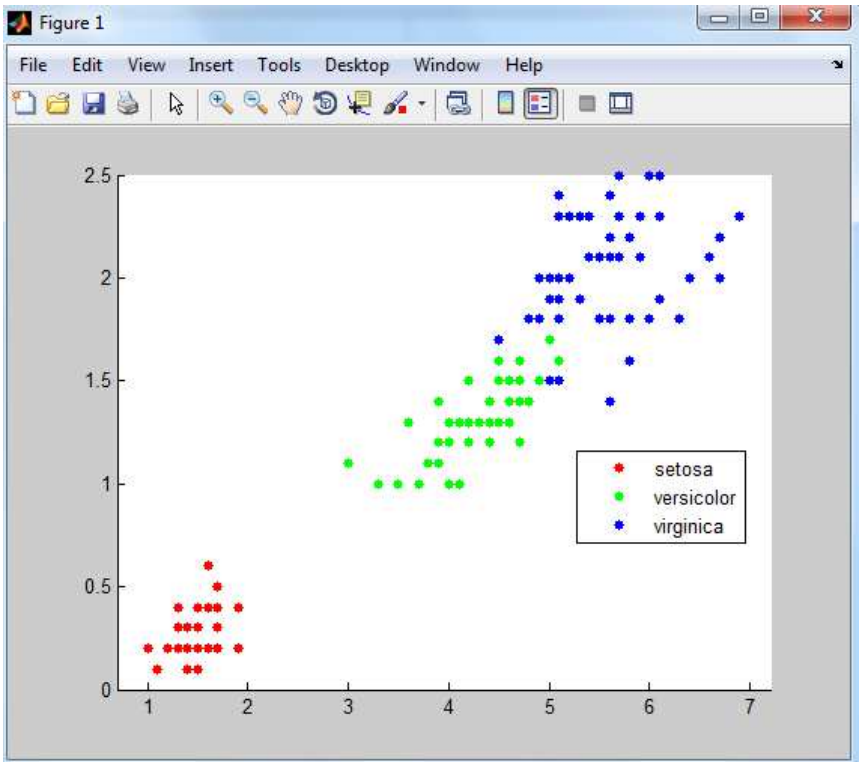

Figure 2. The display of the three data classes

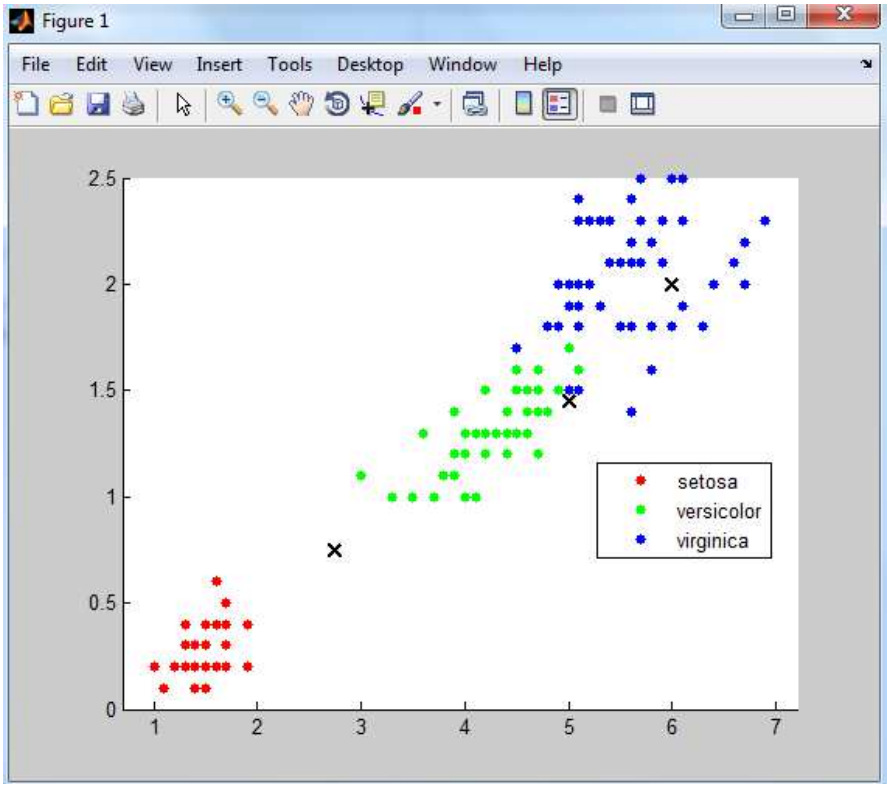

Figure 3. The display of the classes and of the three points to be classified

Figure 4 shows the data classes, the 3 points for classification, marked with circles the points that are closer to the element that needs to be classified. 10 neighbors are used in the k-NNs algorithm, the measure of the distance between new point and points of the classes being the Euclidean distance.

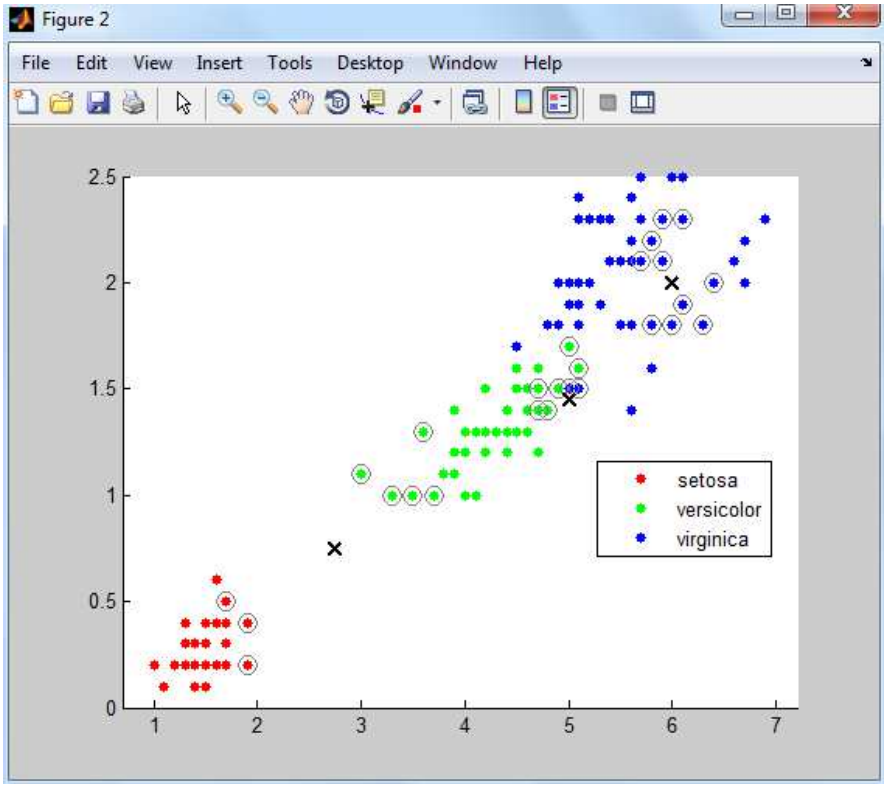

Figure 4. The display of the 10 nearest neighbours of the three points

Figure 5 shows the results of the classification using the k-NNs algorithm. For the first point (the point from bottom left) we have seven neighbors from the versicolor class and three from the setosa class, this being classified in the majority class, respectively versicolor. For the second point (the point from middle) we have eight neighbors from the versicolor class and two neighbors from the virginica class, thus being classified in the versicolor class. The third point (the point from top right) has all neighbors in the virginica class, so it is classified in the same class with them.

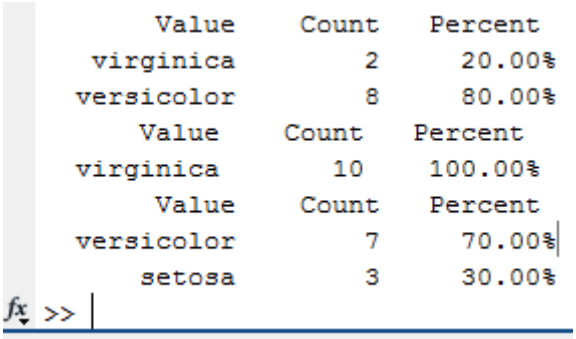

Figure 5. The results of classification for the three point

5. CONCLUSIONS

In this paper a data classification system based on the k-NNs algorithm is presented.

The choosing of the k value influences greatly the response of the algorithm: if k has small values, the algorithm is more effective in finding small structures from the problem space, but can lead to a model too complex (overfitting), and can be influenced significantly by incorrect data (noisy data); If high values for k are chosen,the noises problem in the training data are avoided and the probabilities for tworking with discrete values may be better estimated.

Also an important role in influencing the results is held by the the weights of each neighbor (the closest neighbour has the largest share) and the distances used to calculate metrics between the new element and its neighbors.

The results of the K-NNs algorithm are very good, because the three points are classified correctly, one of them even having maximum percentage.

BIBLIOGRAPHY

[1]T. Mitchell. Machine Learning. McGraw Hill, 1997.

[2]COMP4302/COMP5322, Lecture 6 NEURAL NETWORKS , Clustering. Self-Organizing Feature Maps (SOM), s2, 2003, Available on: http://sydney.edu.au/engineering/it/~comp4302/ann6-6s.pdf.

[3]Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. In: ACM Trans. Information Systems, vol. 22, no. 1, pp. 5-53 (2004).

[4]T. Hofmann. Latent Semantic Models for Collaborative Filtering. ACM Trans. Information Systems, vol. 22, no. 1, 2004, pp. 89–115.

[5]J. Konstan et al. GroupLens: Applying Collaborative Filtering to Usenet News. Comm.ACM, vol. 40, no. 3, 1997, pp. 77–87.

[6]Mitchell T.M., Machine learning and data mining. Communications of the ACM, 42(11):31-36, November 1999.