Department of Computer Sciences
University of Salzburg

VU Pattern Recognition II
WS 2013 / 2014

Classication of Different Datasets using k Nearest Neighbour Classifier

17. Februar 2014

Author: Christof Kauba [1], Stefan Mayer [2]
Academic supervisor: Ao. Univ.-Prof. Dr. Helmut A. Mayer [3]

Universität Salzburg
Fachbereich Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

[1] ckauba@cosy.sbg.ac.at - Matr.Nr.: 0825754
[2] smayer@cosy.sbg.ac.at - Matr.Nr.: 0930426
[3] helmut@cosy.sbg.ac.at

Inhaltsverzeichnis

Abstract

Classification is a very important task in the field of pattern recognition. One widely used method is the kNN (k nearest neighbour) classifier because of its speed and simplicity. A simple kNN classifier is implemented using Java, including a GUI where different distance metrics and k values can be chosen. The results for 3 of the standard UC Irvine Machine Learning Repository Dataset, which are Semeion Handwritten Digits, Red Wine Quality and Ionosphere are evaluated and discussed.

1 Introduction

A kNN (k nearest neighbour) classifier is a widely used method in the field of pattern recognition, not only because its simplicity but also because of its speed and efficiency. Unlike other methods such as ANNs a kNN classifier does not have an explicit training phase. Its training only consists of storing the input patterns. But it also has some disadvantages like the implicit assumption that the training data is evenly distributed. The simple kNN classifier tends to classify all unknown data with the class of the most appearing pattern for unevenly distributed training data.

In this work a simple kNN classifier without improvements like Voronoi Tesselation or optimized data structures but with the choice of several different distance metrics was implemented and applied in classifying 3 of the standard UC Irvine Machine Learning Repository Datasets, which are Semeion Handwritten Digits, Red Wine Quality and Ionosphere are presented.

The rest of this paper is structured as follows: Section 2 describes the kNN classifier, including its advantages, disadvantages and some improvements. Section 3 deals with the different distance metrics used with the kNN classifier. Section 4 gives a description of the datasets. Section 5 describes some implementation details. Section 6 discusses the classification results and Section 7 concludes this paper.

2 kNN Classifier

kNN stands for k- Nearest Neighbour and is a method in the field of pattern recognition, to solve classification and regression problems. The algorithm is one of the simplest classifier mechanism. The inputs are feature vectors for a n-dimensional feature space. The dimension of this vector space is based on the number of features (attributes). A kNN classifier determines the k closest vectors to an unknown pattern by computing the distance to all other vectors. The result is a set with k members. The simplest case is k=1. Here we look at the closest vector to comparison element. In practice, the k value is a small and positive integer value. For $k > 1$, we count the number of occurrence of each class in the set. The winning class is the most occurring class among all patterns in the set which is assigned to the pattern to be classified.

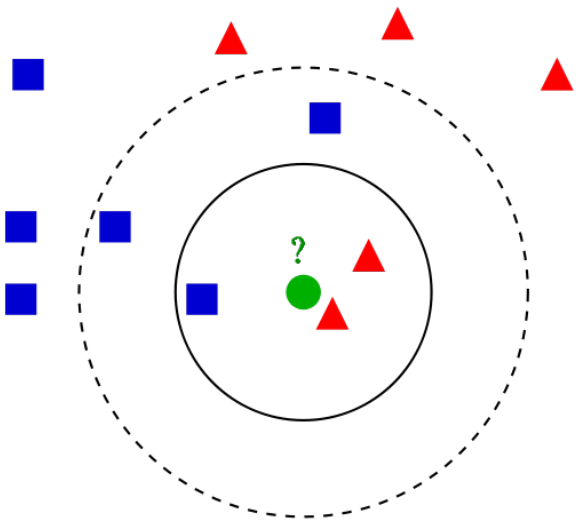

Figure 1: Example of k-NN classification

Figure 1 shows an example of a k-NN classifier. In this example the green circle is the unknown pattern. This scenario is a two class problem. The two classes are the blue squares and the red triangles. If we set $k = 3$ the test sample is assigned to the red class, because there are two triangles and only one square. If we consider $k = 5$ (the dashed circle) the test sample is assigned to the blue class, because the occurrence of squares is higher than the occurrence of red triangles.

2.1 Algorithm

The kNN algorithm uses n-dimensional feature vectors as input trainings samples in a multidimensional feature space, each with a class identifier. In principle, there is no training, but we can denote the storing of the feature vectors and other relevant informations as our training phase. The next step is the classification phase. At first the user has to define the k value. The unknown vector (our test vector) is classified by assigning the class identifier of the pattern with the most occurrences in the k nearest region to the test point. To calculate the distance from the unknown to all other feature vectors there can be used several distance metrics. One of the simplest metric is the Euclidean distance. To improve the classification accuracy, we implemented several distance metrics which we successively tried. The algorithm calculates the distance between all used feature vectors to the test vector. If all distances were computed, the k closest region is analysed.

2.2 kNN Advantages

KNN is one of the simplest classifier and can be implemented almost straightforward. Because of the simplicity of kNN, this method can achieve very fast runtimes, even without any optimizations (mentioned in section 2.4). In comparison to other classification methods, no explicit training is required here before, and new patterns can be added on the fly at very little performance costs. In kNN we have only two parameter which we can tune. On the one hand we can choose different distance metrics, and on the other hand we can change the k-value.

2.3 kNN Disadvantages

In our implementation we are comparing a test instance to all other data instances, and this is relatively expensive for large data sets, because all patterns have to be in the memory at the same time. Another disadvantage is the sensitivity to noisy or irrelevant attributes. This can be avoided by applying a PCA (principal component analysis) or other feature selection methods, before using kNN. In addition, the sensitivity of unbalanced datasets poses problems, because the neighbourhoods are dominated by the classes where most of the entities belong to.

2.4 kNN Improvements

In our implementation we used the simplest kNN mechanism described above. To improve runtime- and memory requirements, there are extended partitioning mechanisms (e.g. Voronoi Tessellation) and optimized data structures (e.g. R-trees).

2.4.1 Voronoi Tessellation

A Voronoi diagram is a method to divide the feature space into a number of partitions. For kNN the Voronoi Tessellation (Figure 2) is used to split the whole feature space in many smaller sub spaces.

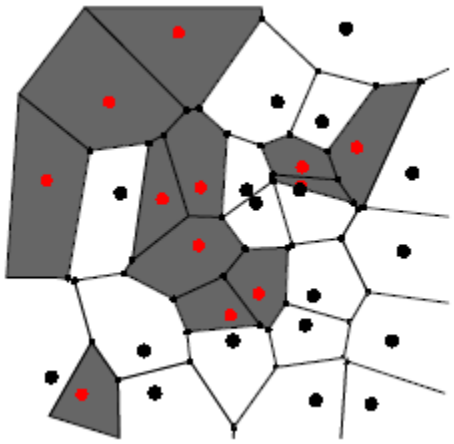

Figure 2: Voronoi tesselation of the space

In the simplest case we are given a finite set of points $P = \{p_1, p_2, ..., p_n\}$ in the Euclidean plane. Each site p_k is simply a point and R_k is the corresponding Voronoi cell consisting of every point whose distance to p_k is less than or equal to its distance to any other site. Let X be a nonempty space with a distance function d. P_k is the set of all points in X where the distance to P_k is less or equal than their distance to the other sites P_j, so that

$$R_k = \{x \in X \,|\, d(x, P_k) \leq d(x, P_j)\ \forall j \neq k\}$$

2.4.2 Optimized Data Structures

Optimized data structures allow fast but exact neighbour identification. Such structures are for example R-trees, which can be used for indexing multi-dimensional information.

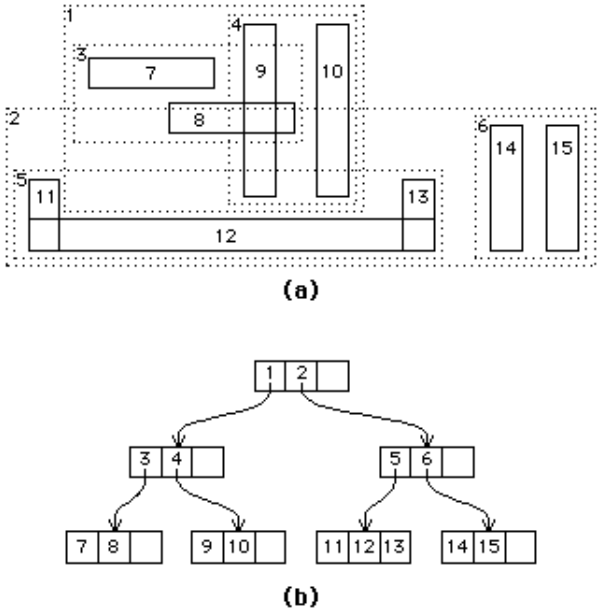

Figure 3: R-tree organization: (a) Spatial ordering (b) Tree representation

This structure is a height-balanced tree in which all nodes are at the same depth and contain spatial objects. Parent nodes store the boundary informations that tightly encloses the leaf objects below.

To search for a point (or in kNN a neighbour), a simple recursive walk through the tree to collect appropriate leaf nodes, is required.[4]

3 Distance Metrics

A distance metric is a mathematic function which defines a distance between two elements, for example two vectors. We can say a metric is a function X with $d : X \times X \to \mathbb{R}$, and $\forall x, y, z \in X$ we satisfy the following axioms:

- Non-negativity: $d(x, y) \geq 0$
- Symmetry: $d(x, y) = d(y, x)$
- Identity of indiscernibles: $d(x, y) = 0 \iff x = y$
- Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

3.1 Metrics

In our kNN implementation we used 12 different popular distance metrics. Let A, B be feature vectors and $a_i \in A, b_i \in B$, $\bar{A}, \bar{B}$ mean of A, B

- Manhattan Distance (L1-Family)

$$d(A, B) = \sum_{i=1}^{n} |a_i - b_i| \tag{1}$$

- Canberra Distance (L1-Family)

$$d(A, B) = \sum_{i=1}^{n} \frac{|a_i - b_i|}{|a_i| + |b_i|} \tag{2}$$

- Euclidian Distance (L1-Family)

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{3}$$

- Squared Euclidian (L2-Family)

$$d(A, B) = \sum_{i=1}^{n} (a_i - b_i)^2 \tag{4}$$

- Cosine Distance

$$d(A, B) = 1 - \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} + \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{5}$$

- Squared Chord Distance

$$d(A, B) = \sum_{i=1}^{n} (\sqrt{a_i} - \sqrt{b_i})^2 \tag{6}$$

- Kullbach Leiber Divergence

$$d(A, B) = \sum_{i=1}^{n} a_i \cdot log(\frac{a_i}{b_i}) \tag{7}$$

[4] http://www.rulabinsky.com/cavd/text/chap03-6.html

- Jeffrey Distance

$$d(A,B) = \sum_{i=1}^{n}(a_i \cdot log(\frac{a_i}{m_i}) + b_i \cdot log(\frac{b_i}{m_i})),\ m_i = \frac{a_i + b_i}{2} \tag{8}$$

- Historgram Intersection Distance

$$d(A,B) = \frac{\sum_{i=1}^{n} min\{a_i, b_i\}}{\sum_{i=1}^{n} b_i} \tag{9}$$

- Correlation Distance

$$d(A,B) = \frac{\sum_{i=1}^{n}(a_i - \bar{A}) \cdot (b_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{A})^2 \cdot \sum_{i=1}^{n}(b_i - \bar{B})^2}} \tag{10}$$

- Chebychev Distance

$$d(A,B) = max_i\{|a_i - b_i|\} \tag{11}$$

- Bhattacharyya Distance

$$d(A,B) = \sqrt{1 - \frac{1}{\sqrt{\bar{A} \cdot \bar{B} \cdot n^2}} \cdot \sum_{i=1}^{n} \sqrt{A(i) \cdot B(i)}} \tag{12}$$

- χ^2Distance

$$d(A,B) = \sum_{i=1}^{n} \frac{(A(i) - B(i))^2}{A(i) + B(i)} \tag{13}$$

4 Datasets

For our classifier we used three datasets from the UC Irvine Machine Learning Repository:

- Semeion Handwritten Digits
- Ionosphere
- Red Wine Quality

The UC Irvine Machine Learning website[5] is a collection of many free datasets which can be used freely for academic purposes.

4.1 Semeion Handwritten Digits

The Semeion Handwritten Digits are binarized images of handwritten digits with 16×16 pixels. Each line in the *.data* file contains 256 binary values (0/1) which represents one handwritten digit image. This dataset is splitted in 10 classes from digit 0 to digit 9. Altogether we have 1593 records. In Figure 4 we can see an example of some handwritten digits.

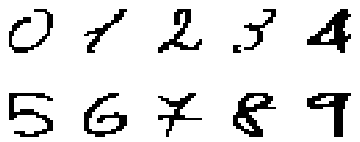

Figure 4: Semeion Handwritten Digits Example

[5] http://archive.ics.uci.edu/ml/index.html

4.2 Ionosphere

The Ionosphere dataset contains radar data collected by a phased array of 16 high-frequency antennas. Considered are free electrons in the ionosphere. Each instance consists of 34 predictor attributes and a class attribute which is ether "good" or "bad" according to the signal pass through the ionosphere. So we have a 2 class problem, with about 351 records. Figure 5 shows one sample data instance from the ionosphere data set.

1	0	0.99539	-0.05889	0.85243	0.02306
0.83398	-0.37708	1	0.03760	0.85243	-0.17755
0.59755	-0.44945	0.60536	-0.38223	0.84356	-0.38542
0.58212	-0.32192	0.56971	-0.29674	0.36946	-0.47357
0.56811	-0.51171	0.41078	-0.46168	0.21266	-0.34090
0.42267	-0.54487	0.18641	-0.45300		g

Figure 5: Ionosphere data instance

4.3 Wine Quality

The Wine Qualities contains several measurement data for different types of wine. Each wine is rated by experts who assigned a quality score between 0 and 10. The wine review is splitted into red- and white wines. We considered only the red wines. The data set is divided in 11 classes (quality score between 0 and 10). Overall we have 1599 records, which have a structure as the data instance sample shown in Figure 6.

fixed acidity	volatile acidity	citric acid	res. sugar	chlorides	free sul ox
7.4	0.7	0	1.9	0.076	11
total sul ox	**density**	**pH**	**sulphates**	**alcohol**	**quality**
34	0.9978	3.51	0.56	9.4	5

Figure 6: Red wine quality data instance

5 Implementation

5.1 Input Normalization

Some input patterns have very distinct ranges of parameter values. So we decided to implement an additional normalization. Particularly for distance metrics which depends on mean, variance or other statistical parameters, the requirement for the same range of values leads to improvements. The normalization maps all given values to a range of $[0,1] \in \mathbb{R}$. Let $A_1 ... A_n$ be feature vectors and $a_{ij} \in A_i$the j-th attribute of A_i. The normalized data value is calculated by the following formular:

$$\tilde{a_{ij}} = \frac{a_{ij} - min_i(a_{ij})}{max_i(a_{ij}) - min_i(a_{ij})} \tag{14}$$

5.2 kNN Implementation

We decided to implement two different variants of kNN. On the one hand we implemented a leave one pattern out cross validation and on the other hand a partitions cross validation (leave one partition out). In addition, the simplest kNN mechanism was implemented, i.e. without partitioning of feature space or special data structures. Our implementation was programmed in Java, and so we use simple Java vectors as data structure. The nearest neighbour algorithm is determined using a fixed length

(= k) priority queue. On start-up the queue is initialized with the first k neighbours. Afterwards, the distance of the current pattern and the unknown pattern is calculated. If the calculated distance is less than the maximal distance in the priority queue, the current pattern is inserted in the queue, and is automatically queued in ascending order. The neighbour element with the highest distance is removed from the queue. This process is done for all given input patterns. Finally we iterate over the whole queue to determine the resulting classification.

5.3 Program Features

The following list shows some program features of our implementation:

- As mentioned above, we used 12 different distance metrics
- To improve the usability we created a graphical user interface (GUI)
- To automate the classification runs, we build a batch testing mechanism, which runs the classification with different parameters. So user can define the range of k, the use of normalization, different distance metrics, and different classification methods (leave one pattern out, leave on partition out). The batch testing returns the best parameter settings, which can optional saved as *.csv* file
- To improve the running time, a multi-threading option was implemented

6 Results

6.1 Confusion Matrix

The confusion matix visualizes the performance of an algorithm. Each represents the predicted (output of the classifier) class and each row represents the actual class. The value n_{ij} in the cell located at the i-th column and j-th row means that n patterns have the actual class i and are classified as belonging to class j.

		predicted class		
		class 1	**class 2**	**class 3**
actual class	**class 1**	5	0	2
	class 2	1	7	1
	class 3	0	3	9

Table 1: Example of a Confusion Matrix

6.2 Runtimes and Classification Rates

As one advantage of the kNN classifier the runtimes are comparably low. The wine quality dataset contains five times as much as data instances as the ionospere one. Therefor the runtime is about six times higher. Although the semeion digits dataset contains nearly as many records as the wine quality one, the runtime is considerably higher because the number of attributes of each instance in the semeion dataset is 256 compared to 11 attributes for the wine quality dataset. Therefore the distance calculation takes much longer for each data instance.

Dataset	**Distance**	**Runtime**	**Best Classification Rate**
Semeion	Bhattacharyya	43.51 s	92.279%
Ionosphere	Jeffrey	0.61 s	92.877%
Wine Quality	Jeffrey	3.86 s	67.04%

Table 2: Runtimes and Classification Rates in comparison

6.3 Semeion

	0	1	2	3	4	5	6	7	8	9
0	98,76%	0,00%	0,00%	0,00%	0,62%	0,00%	0,00%	0,00%	0,62%	0,00%
1	0,00%	95,68%	1,23%	1,23%	0,62%	0,62%	0,00%	0,00%	0,00%	0,62%
2	0,00%	0,63%	94,97%	1,26%	0,00%	0,00%	0,00%	0,00%	2,52%	0,63%
3	0,00%	0,00%	0,63%	91,19%	0,00%	0,63%	0,63%	0,63%	5,03%	1,26%
4	0,00%	4,97%	0,62%	0,00%	91,93%	0,00%	1,86%	0,00%	0,00%	0,62%
5	0,00%	0,00%	0,00%	0,63%	0,00%	88,68%	5,66%	0,00%	1,89%	3,14%
6	1,24%	0,00%	0,00%	0,00%	0,00%	1,24%	97,52%	0,00%	0,00%	0,00%
7	0,00%	3,80%	0,00%	0,00%	0,63%	0,00%	0,63%	90,51%	0,00%	4,43%
8	0,65%	0,65%	3,23%	1,94%	0,00%	0,00%	1,29%	0,00%	89,03%	3,23%
9	0,63%	2,53%	0,00%	6,33%	0,00%	3,16%	0,63%	0,00%	5,06%	81,65%

Table 3: Semeion Confusion matrix

In Figure 7 we can see the classification results for different k values and for the best distance metric (Bhattacharyya Distance). As we can see the best result 92.279% is at $k = 4$. The normalization has no effect on the results. Similar digits (e.g. 8 and 9) are sometimes confused (5.06% of all 9 digits are classified as 8). Changes of the k-value only have a little impact on the classification results.

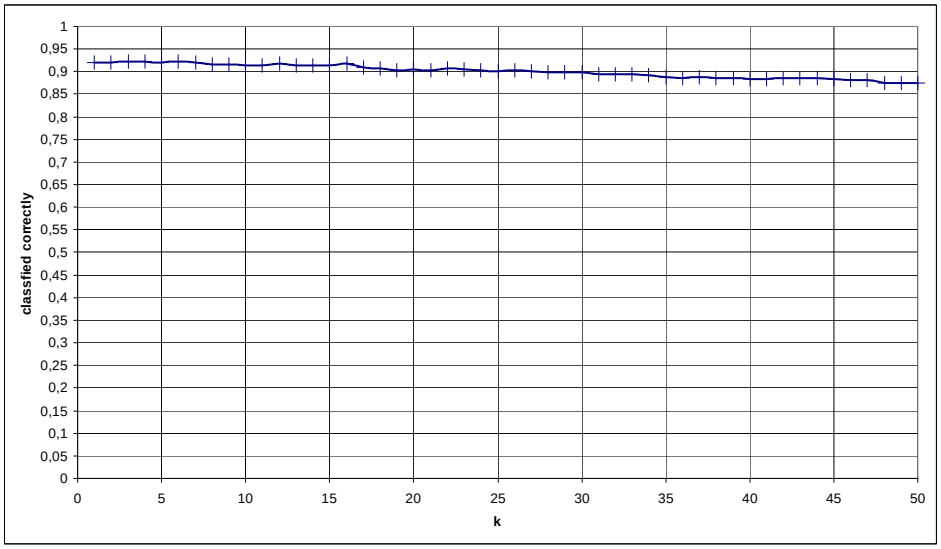

Figure 7: Different k Values for Bhattacharyya Distance on Semeion Dataset

6.4 Ionosphere

	good	bad
good	97,33%	2,67%
bad	15,08%	84,92%

Table 4: Ionospere Confusion matrix

The best distance metric for the ionosphere data set is the Jeffrey distance, shown in Figure 8. At $k = 11$, 92.877% can be correctly classified. The normalization improves the results for the ionosphere data. This is mainly due to the fact, that the Jeffrey distance calculates the mean for each $a_i, b_i \, i = 1, ..., n$ pair.

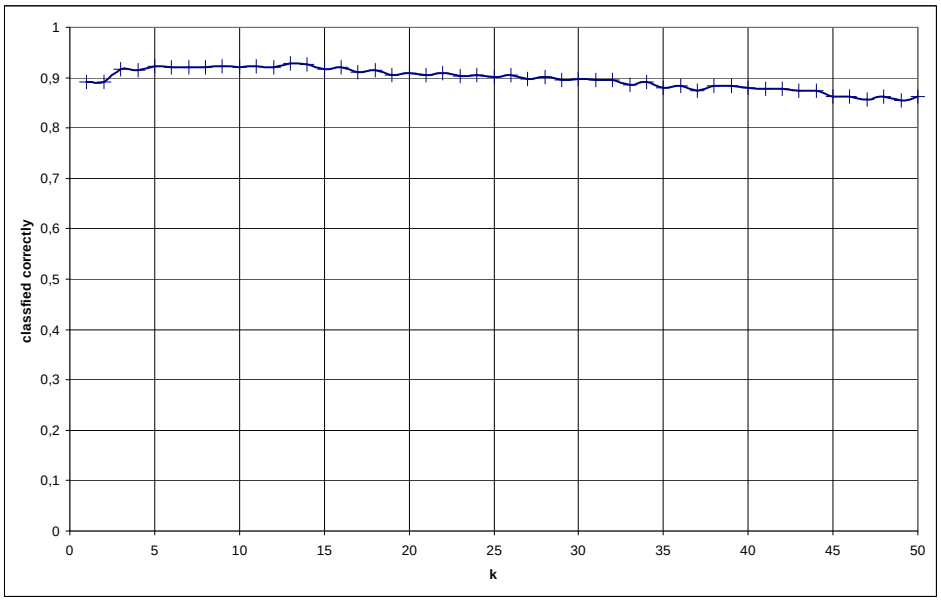

Figure 8: Different k Values for Jeffrey Distance on Ionosphere Dataset

6.5 Wine Quality

	0	1	2	3	4	5	6	7	8	9	10
0	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
1	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
2	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
3	0,00%	0,00%	0,00%	10,00%	30,00%	30,00%	30,00%	0,00%	0,00%	0,00%	0,00%
4	0,00%	0,00%	0,00%	7,55%	11,32%	37,74%	41,51%	1,89%	0,00%	0,00%	0,00%
5	0,00%	0,00%	0,00%	0,44%	3,08%	74,01%	20,56%	1,91%	0,00%	0,00%	0,00%
6	0,00%	0,00%	0,00%	0,16%	2,66%	19,28%	67,55%	9,72%	0,63%	0,00%	0,00%
7	0,00%	0,00%	0,00%	0,00%	0,00%	6,03%	27,14%	64,32%	2,51%	0,00%	0,00%
8	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	44,44%	44,44%	11,11%	0,00%	0,00%
9	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
10	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Table 5: Wine Quality Confusion matrix

As we can see in Figure 9, the wine quality dataset have the most significant differences for the k-values. The best result was at $k = 2$ with the Jeffrey distance and enabled normalization, where 67.04% was correctly classified. The relatively poor results can be attributed to the distribution of quality ratings. The dataset contains no records with a quality value lower than 5 and also no record with a value quality higher than 8. Moreover using a principle component analysis it becomes apparent that most of the alcohol property contains most of the information for classification and all the other properties contain only very little information. Therefore it is very difficult for a simple kNN classifier

to classify this dataset with a high accuracy because on the one hand the data instances are not distributed evenly and most of the properties do only contain noise and no relevant information for classification.

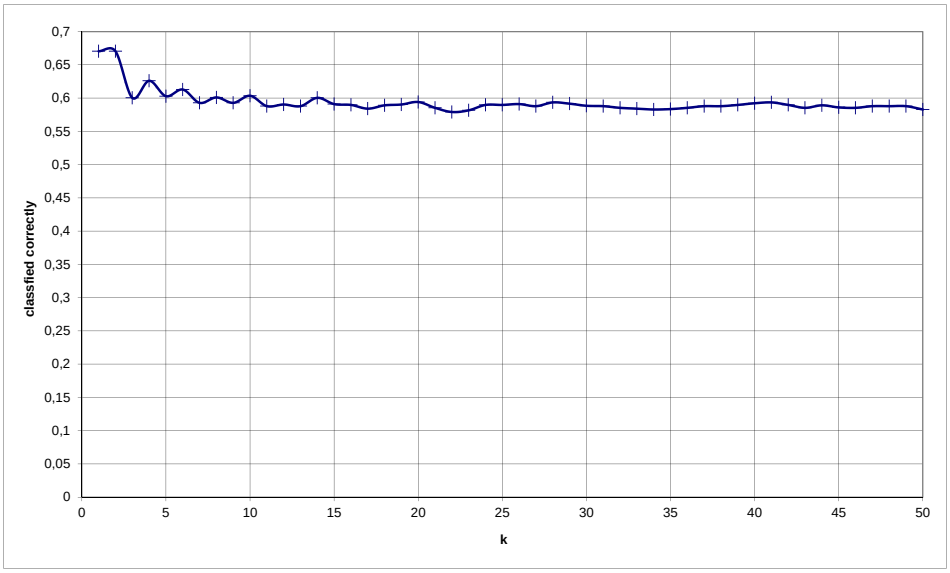

Figure 9: Different k Values for Jeffrey Distance on Wine Quality Dataset

6.6 Comparison of different distance metrics

In the following charts (Figure 10, Figure 11 and Figure 12) we can see the comparison of the different 12 distance metrics, for each dataset.

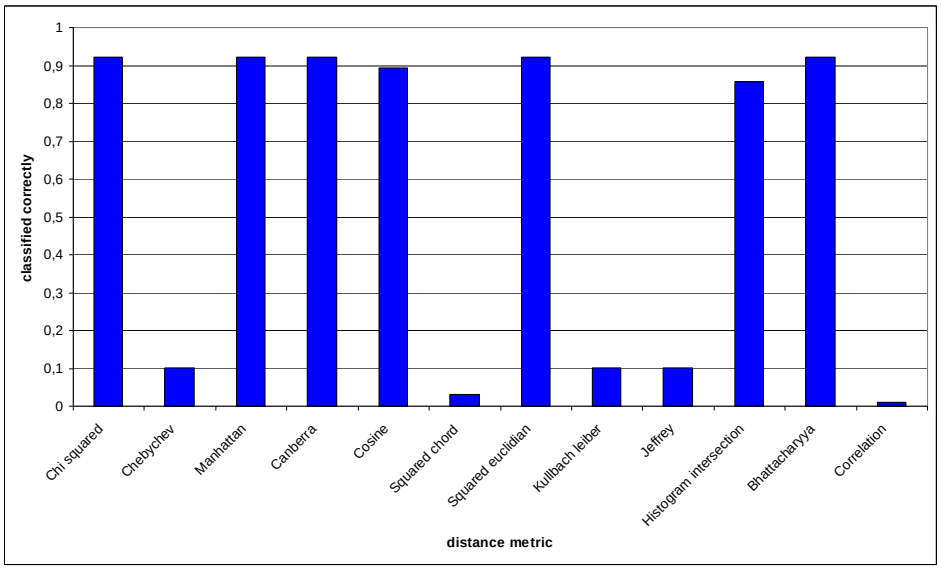

Figure 10: Different Distance Metrics for k = 7 on Semeion Dataset

Many attributes of the semeion dataset have 0-values, therefore distance metrics like the Jeffrey and the Kullbach Leiber Divergence cannot deal with mostly 0-values because they are based on *log* calculation. Correlation simply calculates the difference between corresponding attribute values, which are single pixel values that are not necessarily equal at the same pixel position for equal digits.

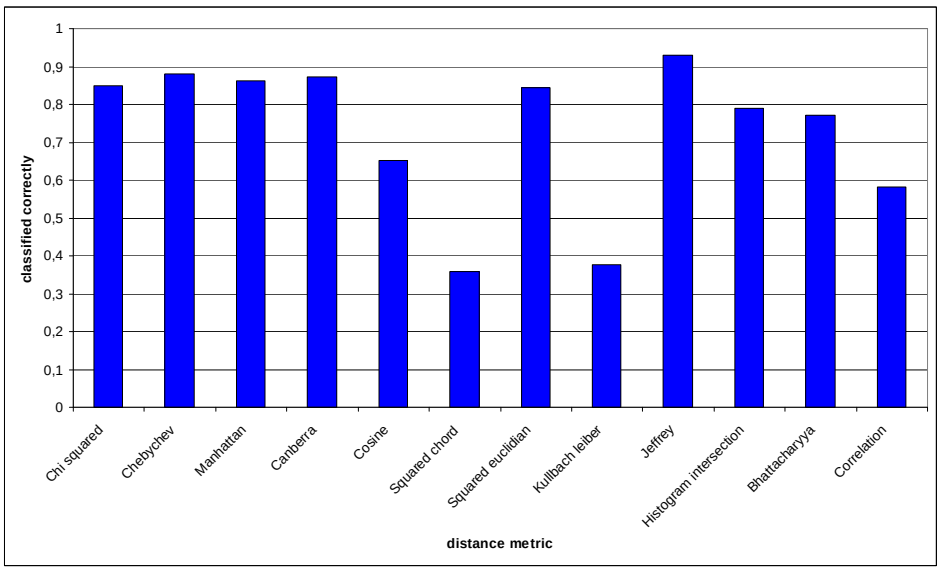

Figure 11: Different Distance Metrics for k = 13 on Ionosphere Dataset

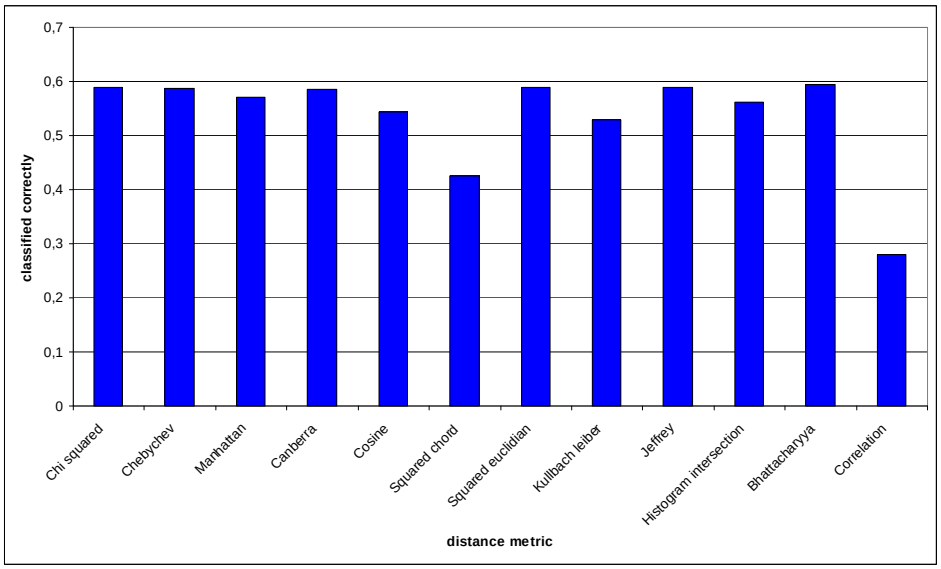

Figure 12: Different Distance Metrics for k = 42 on Wine Quality Dataset

Most of the distance metrics achieve a classification accuracy of 0.55 to 0.6 so the choice of the distance metric is not critical.

7 Conclusion

The native implementation of kNN is quite sufficient because the classification is fast enough. The maximum runtime was about 40 s on a contemporary computer which is quite acceptable. Our classification tests have also shown that the results for Semeion Digits and Ionosphere are comparatively good. Contrary to this, we have found that the Wine Quality dataset has to be pre-processed to achieve better results, for example by using a PCA. The main problem of this dataset is that most data instances have a score of 4-7 and also some attributes are irrelevant.