A Novel Method to Choose Optimum *k* Value in *k*-NN Classification Algorithm

Masoud Maleki
Karadeniz Technical University
masoud.maleki1361@yahoo.com

Negin Manshouri
Karadeniz Technical University
n_manshoori90@yahoo.com

Temel Kayikcioglu
Karadeniz Technical University
tkayikci@ktu.edu.tr

Abstract

The *k*-nearest neighbor (*k*-NN) algorithm is one of the traditional methods that is used in classification. It assigns an unseen point to the dominant class between its *k* nearest neighbors within the data set. However, lack of a formal framework for selecting the number of the neighborhood *k* is problematic. This article investigates a new method for calculating the optimum value of *k* using cross-validation techniques. The proposed method is also fully automatic with no user-set parameters. The proposed method is tested on different benchmark data sets with high classification accuracy rate. The authors compared results with other methods and the proposed method showed 0.1% - 4% higher classification accuracy rate than other methods.

Keywords: K-fold cross-validation, optimum *k*, *k*-nearest neighbor, leave-one-out, pattern recognition

Introduction

This Classification is one of the most active research areas and also important measures in many applications. Classification is the allocation of unknown samples to a known class-based feature vector. Selection of a classifier depends on the kind of problem, used features, and other parameters of problem. The dissatisfactory classification occurs when feature vectors have overlapping areas. In this case, an optimum decision boundary should be made such that the probability of misclassification is minimized (Yooii and Joon, 1995). Classification Accuracy Rate (CAR) is one of the important parameters in performance of a classifier. CAR is the percentage of the number of trials classified correctly in the testing data over the total testing data trials.

k-nearest neighbor classification is one of the fastest, easy to implement, and common algorithms among the existing classification algorithms for statistical pattern recognition (Fix and Hodges, 1951; Cover and Hart, 1968). It forms a limited partition X1, X2, . . . , XJ of the sample space X such that an unknown observation x is classified into the jth class if x ∈ Xj. Performance of a nearest neighbor classifier depends on the distance function and value of the neighborhood parameter k. There are several ways for calculating the distance of two points, which include Minkowski distance, Euclidean distance, City block (Manhattan) distance, Canberra distance, Chebyshev distance, and Bray Curtis distance (Sorensen distance). It is worth mentioning that Euclidean distance method is commonly used in *k*-NN algorithm. If the observations are not of comparable units and scales, it is meaningful to standardize them before using the Euclidean distance.

The other parameter, which controls the volume of the neighborhood and consequently the smoothness of the density estimates, is *k* number of neighbors. It plays a very important role in the performance of a nearest neighbor classifier. If *k* is too small, then the result can be sensitive to noise points; on the other hand, if *k* is too large, then the neighbors may include too many points from other classes (Wu et al., 2008). In many classification studies, selection methods of *k* have not been stated and, in some studies, *k* has been selected using trial-and-error method. In the study by Duda et al. (2008), the best k was selected using (1) in any data set:

$$m=\sqrt{n} \quad (1)$$

n is the number of observations of training data set and the nearest integer value of m is determined as the best *k* value. In this algorithm, *k* is a function of training data set. Enas and Choi (1986) accomplished a simulation study and suggested *k* scaling as n^(2/8) or n^(3/8). n is also the number of observations of training data set. In this algorithm, value of *k* also depends on training data set. In brief, no method is dominating the literature and simply setting *k*=1 or choosing *k* via cross-validation appears the most popular methods (Ripley, 1996). The advantage of cross-validation is that *k*-NN classifies testing observations with awareness and acquaintance to training data set; as a result, it influences the misclassification rate.

In some papers, empirical algorithms have been used, like K-fold cross validation (K-FCV). The best *k* value is selected by maximum value of classification accuracy. In some studies *k*-NN algorithm is trained by K-FCV, in which the best *k* is selected according to maximum classification accuracy rate (Efron, 1983; Huang and Lee, 2009; Onder and Temel, 2011). In another paper, Onder and Temel (2011) used leave one out cross-validation (LOO-CV) method to determine optimum *k* value. They utilized LOO-CV method, since it makes the best use of the available data and avoids the problems of random selections. This algorithm has a high response time when the number of data set is high. In another *k* selection algorithm, Temel and Onder (2010) used sub-sampling method. They repeated this method 30 times and computed each CAR for the validation set for different *k* values. They then selected *k* of maximum CAR and used it in testing data set. As can be seen from literature, in many studies, the value of *k* is selected by many trials on the training and validation sets. But these selected methods are often based on chance and so they are not acceptable. In this work, an awareness algorithm to selecting optimum *k* using cross-validation methods is proposed. This algorithm identify the data set and then select optimum *k*. It does not select *k* by chance. The performance of the proposed method was tested using artificial data set and six different medical datasets, downloaded from UCI machine learning data repository.

The rest of the paper is organized as follows: the next section describes data set. Then, the third section introduces *k* selection algorithms. In Section 4, the authors explain the proposed algorithm. In Section 5, the authors present our experimental results and compare our algorithm with other classical algorithms. Finally, conclusion and discussions are presented in the last section.

Material and methods

Description of data sets

In the following subsections, the authors describe the used data sets.

Description of artificial data sets

The three different types of data sets with different variances and means as well as different distributions in two classes (class1 and class2) were made. These three types (A, B, and C) of distributions made different hypotheses and so changes in sample distributions affected the operation of classifiers in various ways. In Figure 1, these three types of distributions are presented. Observations were 2-dimensional and their number was the same in each class. Variance and mean of different distributions of the data sets are sorted in Table 1. In this table, Dim. denotes dimension. For example, in type ‘A’, mean of the first component of class1 is 3 and the second component is 6. Also, variance of the first component of class 1 is 1.5 and the second component has 15. Means of the first and second components of class2 are 3 and 6, respectively. Variance of the first and second components of class2 is 15 and 1.5, respectively.

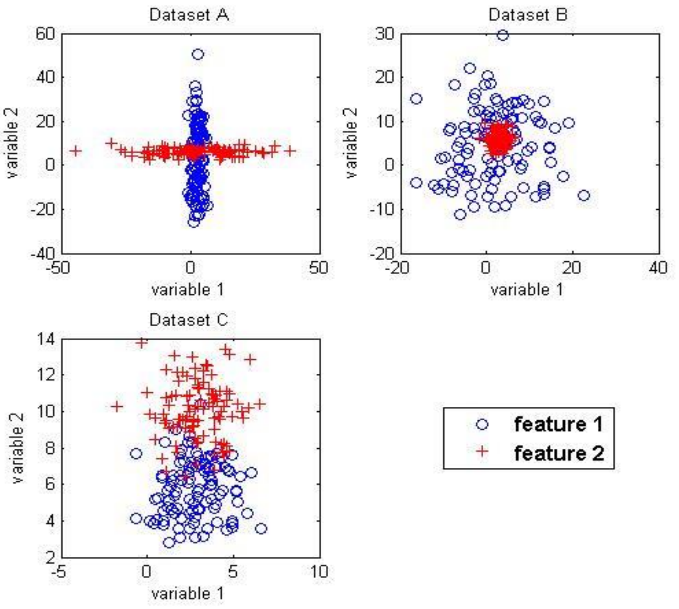

Figure1. Three types of distributions generated by rand function with seed 12 in Matlab environment

Table 1. Mean and variance of artificial data sets with different distributions

Class/distribution		Data Type		
		A	B	C
Mean (First Dim., Second Dim.)	Class1	(3,6)	(3,6)	(3,6)
	Class2	(3,6)	(3,6)	(3,10)
Variance (First Dim., Second Dim.)	Class1	(1.5,15)	(8,8)	(1.5,1.5)
	Class2	(15,1.5)	(1.5,1.5)	(1.5,1.5)

To access the effectiveness of the proposed algorithm, it was tested in the data set with 200 and 1000 observations separately. Table 2 shows the details about the artificial data sets.

Table 2. Artificial data sets characteristics

Dataset Name	Total No. of Observations	Total No. of Features	Total No. of class
Type A	200	2	2
	1000		
Type B	200	2	2
	1000		
Type C	200	2	2
	1000		

Description of UCI data sets

To evaluate the effectiveness of the proposed algorithm on real data, classification of data from University of California, Irvine (UCI) machine learning was performed. The six data sets used for this evaluation are described in detail at the UCI website (accessible at http://www.ics.uci.edu/~mlearn/MLRepository.html.). A summary of each data set is given in Table 3.

Table 3. UCI data sets characteristics

Data set Name	Total No. of Instances	Total No. of Features	Total No. of class
Breast Cancer Wisconsin	699	10	2
Pima Indians Diabetes	768	8	2
Bupa	345	6	2
Heart	270	13	2
Thyroid	7200	21	3
Iris	150	4	3

Proposed method for selecting optimum k

In the following subsections, the authors describe our methods for selecting optimum *k* value using K-fold cross-validation and LOO-CV, separately.

K-fold cross-validation

K-fold cross-validation (K-FCV) is one of the most widely adopted criteria for assessing the performance of a model and for selecting a hypothesis within a class. An advantage of this method, over the simple training-testing data splitting, is the repeated use of the whole available data for both building a learning machine and testing it. Hence, it reduces the risk of (un)lucky splitting (Anguita et al., 2005). In K-fold cross-validation method, data set is randomly split into K subsets with equal size and the method is repeated K times. Each time, one of the K subsets is used as the validation set and the other K-1 subsets are put together to form the pre-training.

The authors illustrate the use of the proposed method using an example. If K=10, the training data set is divided into 10 parts; in each iteration, 9 parts are for pre-training and the rest are related to the pre-testing. Then, the authors check *k* nearest neighbor for the samples. CAR value for all values of *k* is calculated 10 times (due to K=10). Average of CAR is calculated for each *k* in these 10 repetitions. Finally, *k* of the highest CAR is selected. This value of *k* is optimum *k* value. Our results are demonstrated in Table 4 for distribution type A (200 samples in total). As shown in this table, average of CAR (in 10 folds) for *k*=7 neighbor is 96.5; so, it is optimum *k* in 25 nearest neighbor. A common problem in cross-validation methods is the number of folds, into which the training set is divided. In this paper, the authors checked two kinds of folds.

Table 4. Selection of *k* value for distribution type 'A' by proposed method

k /folds	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Average.*
k=1	0,90	1	0,9	1	0,85	0,8	0,9	0,95	1	0,85	0,915
k=3	0,95	1	0,9	0,95	0,85	0,95	0,95	0,9	1	0,85	0,930
k=5	0,95	1	1	0,95	0,85	0,95	0,95	0,9	0,95	0,9	0,940
k=7	0,95	1	1	1	0,9	0,95	1	0,95	0,95	0,95	**0,965**
k=9	0,95	1	0,9	1	0,9	0,95	0,9	0,95	0,95	0,95	0,945
k=11	0,95	1	0,9	1	0,9	1	0,9	0,95	0,95	0,95	0,950
k=13	0,95	1	0,9	1	0,9	1	0,9	0,95	0,95	0,95	0,950
k=15	0,95	1	0,9	1	0,9	1	0,9	0,95	0,95	0,95	0,950
k=17	0,95	1	0,9	1	0,9	1	0,9	0,95	0,95	0,95	0,950
k=19	0,95	1	0,9	1	0,9	1	0,9	0,95	0,95	0,95	0,950
k=21	0,95	1	0,9	1	0,9	0,95	0,9	0,95	0,95	0,95	0,945
k=23	0,95	1	0,85	1	0,9	1	0,95	0,95	0,95	0,85	0,940
k=25	0,90	1	0,85	1	0,9	1	0,9	0,95	0,95	0,8	0,925

Leave-one-out cross-validation

LOO-CV is a particular case of K-FCV with K=N, where N if size of the training set. Hence, the validation sets are all of size one. Like other algorithms, the training data set is divided into two groups. The procedure of LOO-CV method is to take one out of N observations and use the remaining N-1 observations as the training set for deriving the parameters of the classifier (Alippi and Roveri 2010). This process is repeated for all N observations to obtain the estimation for the classification accuracy.

The proposed method was applied in LOO-CV, like K-FCV. If K=N, the training data set is divided into N parts; in each iteration, K-1 parts are used for pre-training and the rest for the pre-testing. Then, the authors check *k* nearest neighbor for the samples. CAR value is calculated for all the values of *k* for N times (due to K=N). Average of CAR is calculated for each *k* in these N times. Finally, *k* of the highest CAR is selected. This value of *k* is optimum *k* value for that data set.

Experimental Results

The authors begin with three types of artificial data set on two class classification problems. These data sets were described earlier in Section 2. MATLAB R2014a was the used environment for creating the data set. By using artificial data set, the number of the available observations was controlled and noise was added according to the experimental purpose. *rand* function in MATLAB R2014a was used to make artificial data set with "seed12". The proposed algorithm was tested and checked on three different distributions of the data set. For all the artificial datasets, the results were reported based on both Euclidean and Mahalanobis distances. The data sets were randomly divided to training and testing observations with the same number. For example, when there were 200 observations in total, in each partition, 100 observations (50 observations from each class) were used for training and 100 observations (50 observations from each class) were for the testing. To test effect of fold in cross-validation, K was set in 10, 20, and N (for LOO-CV). When there were 200 or 1000 observations in total, N was 100 and 500, respectively. In the training item, the proposed method tried to find optimum *k* value between 1 and 25 as well as 1 and 50 when total observations were 200 and 1000, respectively. To verify the proposed method, it was repeated 10 times in each data set. Table 5 shows CARs and standard deviations for three different distributions of artificial data sets when total observations were 1000. Also Table 6 shows results for data sets with 200 observations in total. These results were compared with those of classical training methods.

Table 5. Results for all distributions of artificial data sets with 1000 observations in total

Algorithms & Folds	Methods	Type A	Type B	Type C
City block, 10-Fold	**Proposed method**	**0.9221**±0.0075	**0.9225**±0.0030	0.8971±0.0056
	Onder' method	0.9118±0.0057	0.9095±0.0053	0.8949±0.0053
	Duda' method	0.9112±0.0058	0.9079±0.0071	**0.8978**±0.0066
City block, 20-Fold	**Proposed method**	**0.9236**±0.0060	**0.9243**±0.0075	0.8781±0.0082
	Onder' method	0.9170±0.0103	0.9036±0.0090	0.872 ±0.0119
	Duda' method	0.9133±0.0079	0.9086±0.0077	**0.8984**±0.0089
City block, 500-Fold	**Proposed method**	**0.9224**±0.0062	**0.9241**±0.0034	**0.9035**±0.0069
	Onder' method	0.9051±0.0057	0.8784±0.0067	0.8530±0.0095
	Duda' method	0.904±0.0055	0.9085±0.0067	**0.9035**±0.0062
Euclidean, 10-Fold	**Proposed method**	**0.9240** ±0.0055	**0.9245**±0.0081	0.8903±0.0056
	Onder' method	0.9023±0.0094	0.9008±0.0073	0.8974±0.0086
	Duda' method	0.9025± 0.0068	0.9110±0.0094	**0.9013**±0.0072
Euclidean, 20-Fold	**Proposed method**	0.9264±0.0046	**0.9249**±0.0049	**0.8990**±0.0060
	Onder' method	**0.9270**±0.0057	0.9042±0.0056	0.8907±0.0104
	Duda' method	0.9243± 0.0037	0.9087±0.0068	**0.8997**±0.0066
Euclidean, 500-Fold	**Proposed method**	0.9184±0.0053	**0.9235**±0.0044	0.9007±0.0070
	Onder' method	0.9049±0.0044	0.8798±0.0046	0.8480±0.0064
	Duda' method	**0.9190**±0.0054	0.9094±0.0050	**0.9027**±0.0068

Table 6. Results for all distributions of artificial data sets with 200 observations in total

Algorithms/Folds	Methods	Type A	Type B	Type C
City block, 10-Fold	**Proposed method**	**0.9190**±0.0160	**0.9130**±0.0071	0.9010±0.0204
	Onder' method	0.9020±0.0132	0.9025±0.0175	0.8950±0.0255
	Duda' method	0.9065±0.0194	0.8810±0.0122	**0.9020**±0.0254
City block, 20-Fold	**Proposed method**	**0.9250**±0.0131	**0.8860**±0.0209	**0.9030**±0.0153
	Onder' method	0.9175±0.0223	0.8685±0.0253	0.8830±0.0275
	Duda' method	0.9125±0.0136	0.8685±0.0176	0.9025±0.0174
City block, 500-Fold	**Proposed method**	**0.9145**±0.0172	**0.9080**±0.0130	**0.8995**±0.0174
	Onder' method	0.8990±0.0242	0.8750±0.0111	0.8610±0.0250
	Duda' method	0.9095±0.0109	0.8790±0.0250	0.8970±0.0189
Euclidean, 10-Fold	**Proposed method**	**0.9140**±0.0287	**0.9115**±0.0251	**0.9110**±0.0223
	Onder' method	0.9025±0.0241	0.9055±0.0318	0.9025±0.0190
	Duda' method	0.9075±0.0241	0.8910±0.0204	0.9000±0.0204
Euclidean, 20-Fold	**Proposed method**	**0.9180**±0.0125	**0.9095**±0.0205	0.9085±0.0257
	Onder' method	0.9080±0.0236	0.8915±0.0300	0.8935±0.0252
	Duda' method	0.9180±0.0226	0.8695±0.0206	**0.9095**±0.0251
Euclidean, 500-Fold	**Proposed method**	**0.9220**±0.0225	**0.9075**±0.0206	0.8960±0.0313
	Onder' method	0.9015±0.0267	0.8845±0.0254	0.8660±0.0273
	Duda' method	0.9005±0.0228	0.8660±0.0254	**0.8990**±0.0211

Here, along with the artificial data set, six other real data sets (Breast Cancer Wisconsin, Pima Indians Diabetes, Bupa, Heart, Thyroid and Iris) were used for illustration. For all the datasets, where the measurement variables were of the same unit and scale, the results were computed based on both Euclidean and Manhattan distances. Those sets were formed by randomly partitioning the data. Data sets were randomly divided into training and testing observations in almost the same number. To test the effect of fold on cross-validation, K was set in 10, 20, and N (for LOO-CV) as artificial data sets. To satisfy the value of K in 10 and 20 in some real data sets, observations of training and testing ones were not equal. For example, when there were 268 observations in Pima dataset, in the training partition, 140 observations (70 observations from each class) and 128 observations (64 observations from each class) were used for testing. In the training item, the proposed method tried to find optimum *k* value between 1 and 25 in all the data sets. To verify the proposed method, it was repeated 10 times in each data set. Table 7 and 8 shows these results for two class and three class problems. CARs and standard deviations for all data sets are shown in these tables.

Table 7. Results for all real data sets with two class

algorithms/types	Methods	Pima (140/128)	Wisconsin (120/119)	Bupa (80/65)	Heart (60/60)
City block, 10-Fold	**Proposed method**	0.7078±0.0211	**0.9647**±0.0147	0.6277±0.0342	**0.6967**±0.0276
	Onder' method	0.7016±0.0239	0.9597±0.0180	**0.6362**±0.0441	0.6833±0.0340
	Duda' method	**0.7094**±0.0281	0.9513±0.0174	0.6269±0.0329	0.6842±0.0307
City block, 20-Fold	**Proposed method**	**0.7399**±0.0133	**0.9739**±0.0118	**0.6408**±0.0301	**0.6992**±0.0268
	Onder' method	0.7063±0.0178	0.9567±0.0074	0.6162±0.0608	0.6883±0.0500
	Duda' method	0.7063±0.0220	0.9504±0.0071	0.6231±0.0274	0.6900±0.0378
City block, N-Fold	**Proposed method**	**0.7364**±0.0220	**0.9768**±0.0187	0.6377±0.0574	**0.7033**±0.0233
	Onder' method	0.6465±0.0232	0.9592±0.0179	0.5900±0.0514	0.6367±0.0193
	Duda' method	0.7145±0.0096	0.9555±0.0171	**0.6438**±0.0570	0.6958±0.0252
Euclidean, 10-Fold	**Proposed method**	**0.7061**±0.0210	**0.9772**±0.0099	0.6223±0.0268	0.6375±0.0255
	Onder' method	0.6926±0.0252	0.9667±0.0156	0.6285±0.0366	0.6417±0.0255
	Duda' method	0.6918±0.0303	0.9535±0.0098	0.6192±0.0374	**0.6475**±0.0319
Euclidean, 20-Fold	**Proposed method**	0.6887±0.0315	**0.9873**±0.0075	0.6208±0.0399	**0.6592**±0.0234
	Onder' method	0.6871±0.0268	0.9618±0.0121	0.6262±0.0359	0.6250±0.0412
	Duda' method	**0.6930**±0.0281	0.9605±0.0084	**0.6315**±0.0335	0.6317±0.0222
Euclidean, N-Fold	**Proposed method**	**0.7016**±0.0312	**0.9723**±0.0060	**0.6246**±0.0351	**0.6350**±0.0340
	Onder' method	0.6535±0.0334	0.9664±0.0114	0.5831±0.0427	0.6033±0.0261
	Duda' method	0.6875±0.0128	0.9588±0.0092	0.6192±0.0361	0.6267±0.0218

Table 8. Results for real data sets with three class

algorithms/types	Methods	Iris 25/25 10 times	Thyriod 90/76 10 times
City block, 10-Fold	**Proposed method**	0.9573±0.0216	**0.7030**±0.0405
	Onder' method	0.9480±0.0160	0.6825±0.0397
	Duda' method	**0.9587**±0.0203	0.6829±0.0582
City block, 20-Fold	**Proposed method**	**0.9573**±0.0216	**0.7086**±0.0283
	Onder' method	0.9427±0.0199	0.6833±0.0348
	Duda' method	0.9520±0.0157	0.6601±0.0342
City block, N-Fold	**Proposed method**	**0.9680**±0.0129	**0.6850**±0.0299
	Onder' method	0.9507±0.0167	0.6632±0.0248
	Duda' method	**0.9680**±0.0157	0.6658±0.0308
Euclidean, 10-Fold	**Proposed method**	**0.9780**±0.0143	**0.6533**±0.0315
	Onder' method	0.9613±0.0098	0.6298±0.0361
	Duda' method	0.9667±0.0169	0.6167±0.0436
Euclidean, 20-Fold	**Proposed method**	**0.9627**±0.0371	**0.6512**±0.0233
	Onder' method	**0.9627**±0.0138	0.6373±0.0339
	Duda' method	0.9500±0.0327	0.5982±0.0314
Euclidean, N-Fold	**Proposed method**	0.9560±0.0209	**0.6512**±0.0409
	Onder' method	0.9560±0.0090	0.6386±0.0326
	Duda' method	**0.9627**±0.0225	0.6184±0.0333

Discussion

When the number of observations in the data set is high, the learning task needs a long time and using LOO-CV is not suitable. In this case, K-FCV is a good way for the learning task. The results showed that the present algorithm by K-FCV was a very good way for finding the optimum value of *k* when the number of observations was high, because the response time of the proposed algorithm was very low. Also, these results illustrated that the size of the folds in K-FCV algorithm did not greatly affect the results. As mentioned, the results were computed based on both Euclidean and Manhattan distances and it was found that the kind of distance did not affect the proposed method. The present results were compared with those of classical training methods Finally, Correct understanding of dataset's distribution is very important in selecting optimum k. Duda' method (2008) like Enas and Choi method (1986) without consideration of this important factor, select a *k* as a optimum k. Onder' method like other some researches (Efron, 1983; Huang and Lee, 2009; Onder and Temel, 2011, Onder and Temel (2011), Temel and Onder (2010) try to understand of dataset's distribution. But this trying is not enough. Proposed method with consideration of correct understanding of dataset's distribution, select optimum k. The proposed method select optimum *k* because results show 0.1% - 4% higher CAR than other methods, as shown in Tables 5, 6, 7, and 8.

Conclusion

k-nearest neighbour (*k*-NN) algorithm is one of the most popular methods that supervises learning algorithm and exploits lazy learning between classification methods. *k*-NN's performance is highly competitive with other techniques. There are several key issues that affect the performance of *k*-NN, one of which is distribution of data set. Value of *k* plays an important role in the decision of unknown pattern in *k*-NN; so, it can be considered another issue. In this paper, a new method was presented for selecting optimum *k*-nearest neighbor using cross-validation methods in *k*-NN algorithm. The results were computed based on both Euclidean and Manhattan distances. It was found that the kind of distance and number of folds in cross validations methods did not affect the proposed method. The proposed method was also fully automatic with no user-set parameters. The experimental results by the proposed algorithm could decide the most optimum *k* value in compare with other algorithms according to achieved classification accuracy rates (CAR). This algorithm was applied to different distributions of artificial and real data sets.

References

Anguita, D., Ridella, S., Fabio R. (2005). K-Fold Generalization Capability Assessment for Support Vector Classifiers, Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4.

Alippi C., M. Roveri. (2010). Virtual k-fold cross validation: an effective method for accuracy assessment, in Proc. IEEE International Joint Conference on Neural Networks (IEEE IJCNN 2010), Barcelona, Spain, July 18-23.

Cover, T.M., Hart, P.E. (1968). Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13, 21–27.

Duda R. ET Hart P. E., and Stark D. G. (2000). Pattern classification, 2nd edition, John Wiley.

Efron B., (1983). Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation, Journal of the American Statistical Association, Vol. 78, No. 382, pp. 316-331.

Enas, G. G. and Choi, S. C. (1986). Choice of the smoothing parameter and efficiency of k-nearest neighbor classification, Comput. Math.Applic. A, 12, 235–244.

Fix, E., Hodges, J.L. (1951). Discriminatory analysis nonparametric discrimination: consistency properties, International Statistical Review, Project 21-49-004, Report 4, pp. 261–279. US.

Huang, P., Lee. C. H. (2009). Automatic Classification for Pathological Prostate Images Based on Fractal Analysis, IEEE Transaction on Medical Imaging, vol. 28, NO. 7, JULY.

Onder, A. and Temel. K. (2011). Wavelet Transform Based Classification of Invasive Brain Computer Interface Data, Radio engineering, 20(1), pp.: 31-38.

Onder, A. and Temel. K., (2011). Comparative Performance Assessment of Classifiers in Low-Dimensional Feature Space Which are Commonly Used in BCI Applications, Elektrorevue, 2(4), pp.: 58-63.

Ripley, B.D. (1996). Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge.

Temel, K. and Onder. A. (2010). A Polynomial Fitting and k-NN Based Approach for Improving Classification of Motor Imagery BCI Data, Pattern Recognition Letters, 31(11), pp. 1207-1215.

Wu X., Kumar V. and J. Ross Quinlan. (2008). Top 10 algorithms in data mining, Knowledge and Information Systems, Volume 14, Issue 1, pp 1-37.

Yooii K. Kim and Joon H. Han. (1995). Fuzzy K-NN Algorithm using Modified K- Selection, 0-7803-2461-7/95, International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., IEEE.