Water Resour Manage (2011) 25:1341–1357
DOI 10.1007/s11269-010-9748-z

Clustered K Nearest Neighbor Algorithm for Daily Inflow Forecasting

Mahmood Akbari · Peter Jules van Overloop · Abbas Afshar

Received: 23 February 2010 / Accepted: 18 November 2010 /
Published online: 7 December 2010

Abstract Instance based learning (IBL) algorithms are a common choice among data driven algorithms for inflow forecasting. They are based on the similarity principle and prediction is made by the finite number of similar neighbors. In this sense, the similarity of a query instance is estimated according to the closeness of its feature vector with those of data available in calibration data. As the selected attributes in the feature vector are determined overall on calibration data, there may be some data points whose outputs do not follow the considered attributes. In fact, output values of these inconsistent data points may be a function of some other attributes which were not considered. Therefore, for some query instances, the inconsistent points may be appeared as the neighbors while they may not really be neighbor to the query instance. They can deteriorate forecasting results especially if they are very close to the query instance with the current similarity definition. In this study a clustered K nearest neighbor (CKNN) algorithm is introduced which can capture these inconsistent data points. Similar to the inconsistent data points, CKNN can be also robust against noisy data. The proposed algorithm was shown to be effective for a synthetic linear data set corrupted by noise. In addition, the utility of the algorithm was demonstrated for daily inflow forecasting of the Karoon1 reservoir located in Iran.

M. Akbari (✉) · A. Afshar
School of Civil Engineering and Enviro-Hydroinformatic Center of Excellence,
Iran University of Science & Technology, Tehran, P. O. Box 16765-163, Iran
e-mail: makbari@iust.ac.ir

A. Afshar
e-mail: a_afshar@iust.ac.ir

M. Akbari
Department of Civil Engineering, University of Kashan, Kashan, Iran

P. J. van Overloop
Water Management Section, Delft University of Technology, Delft, The Netherlands
e-mail: p.j.vanoverloop@citg.tudelft.nl

Keywords Inconsistent data · Inflow forecasting · K nearest neighbor · Subtractive clustering · Noisy data

1 Introduction

In the most commonly used learning algorithms, such as neural networks and decision trees, a single model is used to construct an explicit global representation of the target function over all training data. In contrast, the instance based learning (IBL) algorithms simply store some or all of the training examples and postpone any generalization effort until a new query instance must be predicted. In fact they apply specific cases or experiences to new situations by matching known cases and experiences with new cases (Atkeson et al. 1997). They can thus build query-specific local models, which attempt to fit the training examples only in a region around the query instance. This type of learning method is also referred to as lazy learning method. Because it just finds a set of the nearest neighbors and constructs a local model based on them. Examples of the types of local models include Nearest Neighbor (NN), K Nearest Neighbor (KNN) and Locally Weighted Regression (LWR). NN local models simply choose the closest point and use its output value. KNN local models average the outputs of the nearby points. The KNN method can further be improved by weighting each of K Neighbors inversely according to their distance to the query instance. LWR fits a linear or nonlinear local model to the nearby points using a distance weighted regression. Thus, locally weighted regression is very similar to least-squares regression, except that the error terms used to derive the best linear or nonlinear approximation are weighted by the inverse of their distance to the query instance.

Instance based learning algorithms in their different forms have been successfully applied in the field of inflow forecasting as in many other areas. Karlsson and Yakowitz were probably the first to explore the use of these methods for inflow forecasting (Karlsson and Yakowitz 1987; Yakowitz 1987). Galeati (1990) applied the KNN method for daily discharge forecasting and compared it to the ARX (Auto Regressive Exogenous input) model. Shamseldin and O'Connor (1996) extended the KNN method to NNLPM (Nearest Neighbor Linear Perturbation Model) for river flow forecasting. In this model, perturbations of the most recent rainfall segment of test pattern with respect to the mean of its nearest neighbors are linearly related to the resulting perturbation in the runoff from the mean runoff of the nearest neighbors.

Lall and Sharma (1996) presented a nearest neighbor bootstrap for resampling monthly streamflow and showed its utility in comparison to linear and nonlinear autoregressive models. Coulibaly et al. (2005) used the NN method in the framework of multi models approach for daily inflow forecasting. Solomatine et al. (2008) demonstrated the priority of the KNN and LWR approaches compared to Artificial Neural Networks (ANN) and M5 model trees in the short term inflow forecasting.

IBL uses the similarity (neighborhood) between the feature vector of the query instance and *K* similar sets of historical feature vectors to obtain the best estimate for the dependent variable of the query instance. IBL assumes that the prediction of a test instance will be close to the output values of the nearby points. As a result, in the ideal case, the output values of the nearest neighbors are not to be very far from each other. This is because the neighbors of the query instance are neighbors to each other too. In other words, when the output values of the neighbors are not very close, forecasting may be risky (Karlsson and Yakowitz 1987). This may arise mainly from three reasons: 1) calibration data set is not sufficiently large 2) relevant attributes of the query instance are not suitable 3) noisy data may be included in the neighbors. In the first case, because of unprecedented events, neighbors may be *inconsistent*. In such situations, good forecasts with IBL are not expected as any other data driven model. The second and third cases, however, are considered in this study.

Feature selection, as used in the IBL method, is one of the most important aspects of pattern recognition. In the field of inflow forecasting, the feature vector is commonly composed of the lagged rainfall and runoff as relevant attributes (Karlsson and Yakowitz 1987; Yakowitz 1987; Coulibaly et al. 2005; Shamseldin and O'Connor 1996; Solomatine et al. 2008). The feature vector can be also dependant on other relevant information such as soil moisture, snow pack, temperature, etc. (Karlsson and Yakowitz 1987). But such series are neither available nor considered to reduce complexity. Furthermore, relevant independent variables are usually determined overall by correlation analysis, mutual information analysis or other methods over the calibration data (Bowden et al. 2005). Unsuitable attributes can result in false estimation of the similarity and in choosing wrong neighbors and, consequently, in inaccurate forecasts. When attributes are selected overall, there may be some data points whose outputs do not follow the considered attributes. In fact, the output values of these inconsistent data points may be a function of some other attributes which were not considered. Therefore, for some query instances, the inconsistent points may appear as neighbors while they are not really neighbors to the query instance. They can deteriorate forecasting results, especially if they are very close to the query instance with the current similarity definition.

Real data always poses the challenge of managing noise. As forecasting in the IBL is done by the finite K nearby points, some noisy data points included in the data sets may affect the output in a similar way as inconsistent data points. In order to improve the prediction accuracy, it is often necessary to reduce the original training set by removing some noisy instances before the learning phase (Czarnowski and Jędrzejowicz 2006).

In this study, subtractive clustering algorithm is employed to capture the inconsistent data as well as noisy data points. The proposed clustered KNN algorithm (CKNN) prunes the inconsistent and noisy data points in the neighbors which results in improved forecasting as compared to the traditional KNN. The performance of the proposed flow forecasting scheme is tested on inflow to the Karoon1 reservoir located in Iran.

The rest of this paper is constructed as follows: first, the traditional KNN model is described. The proposed algorithm (CKNN) is introduced and its applications to synthetic and streamflow data are then presented. Finally the paper ends up with the conclusions.

2 K Nearest Neighbor Model

K Nearest Neighbor learning, one of the most popular realizations of IBL, combines the target values of K selected neighbors to predict the target value of a given test pattern.

A weighted Euclidean norm is usually used to measure the closeness (similarity) of the feature vector of query instance ($\boldsymbol{X}_q$) and any feature vector of calibration data set ($\boldsymbol{X}_i$):

$$d\left(\mathbf{X}_q, \mathbf{X}_i\right) = \sqrt{\sum_{j=1}^{m} w_j^a \left(x_{ij} - x_{qj}\right)^2} \tag{1}$$

where i, j are indices for data and attributes respectively, w_j^a is the weight of each attribute, m is the number of attributes, and x_{ij} is the normalized value of jth attribute of ith data point. It should be noted that each attribute is normalized to minimize the scale difference. The weights of attributes are determined such that they produce the lowest mean square error of forecasting over the calibration data set (Karlsson and Yakowitz 1987).

Given the output values of neighbors (Y_i), the predicted output for the test pattern (Y_q) is calculated as follows:

$$\hat{Y}_q = \sum_{i=1}^{K} w_i^N Y_i \Bigg/ \sum_{i=1}^{K} w_i^N \tag{2}$$

where w_i^N is the weight of each neighbor and K is the number of neighbors. In the simple form of KNN, w_i^N is employed to be $1/K$ and the estimate is the mean value of K nearest neighbors. In the modified form of KNN, however, each neighbor is usually given a weight based on the distance between the neighbor and the test pattern. A farther neighbor receives a smaller weight, which reduces its effect on the prediction results compared to other closer neighbors. In this way, some kernel functions, which decrease monotonically as distance increases, have been used. There are a number of well-known kernel functions such as linear kernel (Solomatine et al. 2008), inversion kernel (Ruprecht and Müller 1994; Solomatine et al. 2008), exponential kernel (Aha and Goldstone 1992), and Gaussian kernel (Wand and Schucany 1990) which are defined as follows:

$$\begin{array}{lll} \text{a) Linear} & w_i^N = 1 - d\left(\mathbf{X}_q, \mathbf{X}_i\right) \\ \text{b) Inverse} & w_i^N = \left(d\left(\mathbf{X}_q, \mathbf{X}_i\right)\right)^{-1} \\ \text{c) Square inverse} & w_i^N = \left(\left(d\left(\mathbf{X}_q, \mathbf{X}_i\right)\right)^2\right)^{-1} \\ \text{d) Exponential} & w_i^N = \exp\left(-d\left(\mathbf{X}_q, \mathbf{X}_i\right)\right) \\ \text{e) Gaussian} & w_i^N = \exp\left(-\left(d\left(\mathbf{X}_q, \mathbf{X}_i\right)\right)^2\right) \end{array} \tag{3}$$

Atkeson et al. (1997) claimed that there is no clear evidence that a specific kernel function is always superior to others, however, some outperformed others on some data sets (Solomatine et al. 2008).

The number of nearest neighbors which should be considered in estimation is itself a challenging issue. Indeed, no well-established method exists for selecting an optimal K for using the KNN. The number of neighbors is often chosen empirically by cross-validation or domain experts in practice (Kang and Cho 2008).

In this study, KNN is traditionally calibrated off-line using a calibration data set and then the optimal number of nearest neighbors (K) and the weights of the attributes $\left(w_j^a\right)$ are determined.

3 Clustered K Nearest Neighbor Algorithm

IBL is based on the assumption that the prediction of a test instance should be close to the output values of the nearby points. In this sense, neighborhood refers to the similarity between the feature vector of the query instance and the feature vectors of historical data. The CKNN presented in this study assumes that if the input vectors of the neighbors are similar to the query instance and consequently similar to each other, their targets are also similar. Therefore, the output values of the neighbors with near feature vectors are to be relatively close to each other. In this situation, the similar neighbors in the feature vector and the output values may form some local groups of the neighbors. The number of neighbors in each group can be different, and some neighbors may belong to none of the clusters. In such cases, the prediction based on these neighbors may be risky. In fact, if the output values of some neighbors are very far from the rest, the prediction for the query instance may be unreliable. This is because the prediction in KNN is based on similarity, but all neighbors are not similar. In the traditional KNN, each neighbor contributes to the output of the query instance proportional to its closeness to the query instance. In reality, however, each neighbor may not be as reliable as its closeness. If some neighbors stay alone, they may be the inconsistent points in which their output values probably do not follow the current attributes. In fact, they may not really be neighbor to the query instance. In this sense, noisy data can behave similar to the inconsistent data points.

In the CKNN, data clustering is applied for identifying the inconsistent and noisy data points among the neighbors for each query instance. Data clustering is a process of putting similar data into groups. A clustering algorithm partitions a data set into several groups such that the similarity within each group is larger than among groups (Jang et al. 1997). For the purpose of this study, each neighbor as a data point is identified with a two dimensional vector as follows:

$$\mathbf{V}_i = \left(d\left(X_q, X_i\right), Y_i\right) \tag{4}$$

In the CKNN model, after clustering the neighbors, each neighbor belonging to none of the clusters is removed from the neighbors and then estimation is done based on the remaining neighbors.

Some clustering algorithms need to know the number of clusters explicitly while others automatically specify the number of clusters during the process of clustering. This study employs the subtractive clustering algorithm proposed by Chiu (1994) which does not require any prior knowledge about the number of clusters. For this method, data points have to be rescaled to [0, 1] in each dimension. In the subtractive clustering algorithm, each data point is a potential cluster center. This potentiality is calculated based on the density of surrounding data points according to:

$$P^l = \sum_{i=1}^{K} \exp\left(-\frac{||\mathbf{V}_i^{norm} - \mathbf{V}_l^{norm}||}{(r_a/2)^2}\right); \quad l = 1, 2, \ldots, K \tag{5}$$

Where $\mathbf{V}_l^{norm}$ and P^l are the normalized vector and the potential of the lth data point respectively; $\|.\|$, the Euclidian distance; and r_a is a positive constant called cluster radius. Hence, a data point will have a high density value if it has many closer data points. In that case, the algorithm selects the first cluster center $\mathbf{V}_{C1}^{norm}$ as the point having the highest potential value P^{C1}. Next, the density measure of each data point $\mathbf{V}_l^{norm}$ is revised as follows:

$$P_{new}^l = P_{old}^l - P^{C1}.\exp\left(-\frac{||\mathbf{V}_l^{norm} - \mathbf{V}_{C1}^{norm}||}{(r_b/2)^2}\right); \quad l = 1, 2, \ldots, K, l \neq C1 \tag{6}$$

Thus, we subtract an amount of potential from each data point as a function of its distance from the first cluster center. The data points near the first cluster center will have a greatly reduced potential, and therefore will unlikely be selected as the next cluster center. To avoid obtaining closely spaced cluster centers, we set r_b to be somewhat greater than r_a. After revising the density function, the next cluster center is selected as the point having the greatest density or potential value. The process of acquiring new cluster centers proceeds based on potential value in relation to an acceptance threshold $\left(\bar{\varepsilon}.P^{C1}\right)$, rejection threshold $\left(\underline{\varepsilon}.P^{C1}\right)$ and the shortest distance between the candidate cluster center and all previously found cluster centers ($DIS_{\min}$). A data point with a potential greater than the acceptance threshold is directly accepted as a cluster center. A data point with a potential between the upper and the lower thresholds is accepted if the following inequality holds:

$$\frac{DIS_{\min}}{r_a} + \frac{P^l}{P^{C1}} \geq 1 \tag{7}$$

In such a case, $\mathbf{V}_l^{norm}$ is accepted as the next cluster center, otherwise if the above inequality does not hold or $P^l \prec \underline{\varepsilon}.P^{C1}$ the algorithm terminates. Finally, the data points within the radius distance of a cluster center are assigned to the related cluster and any data point whose minimum distance from the cluster centers is greater than the radius distance, is left out. The recommended values for the parameters of subtractive clustering algorithm are $0.15 \leq r_a \leq 0.3$, $1.25r_a \leq r_b \leq 1.5r_a$, $\bar{\varepsilon} = 0.5$ and $0.15 \leq \underline{\varepsilon} \leq 0.5$ (Chiu 1994).

After the neighbors are grouped into different clusters, each neighbor sitting in none of the clusters is recognized as an inconsistent point, thereby it is ignored and then prediction for the query instance is made by other neighbors.

4 Applications

This section illustrates how noisy and inconsistent points in training data set affect the predictive ability with respect to the original KNN through two experiments. The first experiment uses the data from known synthetic linear data corrupted by noise. The second one presents an application to the real world for daily flow forecasting where the inconsistent data points may deteriorate the results of the traditional KNN model. In the latter case, preprocessing of the data was done before applying the models to remove probable noisy data included into the calibration data set, but the data set may still be infected by some noisy data.

As performance measures, the root mean square error (RMSE) and mean absolute error (MAE) are employed for comparing the results of KNN and CKNN models:

$$\begin{aligned} RMSE &= \sqrt{\frac{\sum_{q=1}^{n}\left(Y_q - \widehat{Y}_q\right)^2}{n}} \\ MAE &= \frac{1}{n}\sum_{q=1}^{n}\left|Y_q - \hat{Y}_q\right| \end{aligned} \tag{8}$$

Comparing the performance of KNN to different distance kernel functions, it was observed that KNN with square inverse distance (3-c) resulted in the best performance for the real data experiment. The same kernel function was also used for the case of synthetic data. An off-line optimization process on the size of neighborhood (i.e. K) was employed using calibration data set and the optimal values of K were found to be 8 and 10 for the synthetic and real data applications, respectively.

In order to find the optimal CKNN model for each experiment, the parameters of the subtractive clustering algorithm were varied within the proposed ranges. For each combination of these parameters, a model was built and trained. Through optimization of the performance indices RMSE and MAE obtained on the calibration data, the optimal parameters combination was sought. Finally, the following values were retained: $r_a = 0.2$, $r_b = 0.3$, $\bar{\varepsilon} = 0.5$ and $\underline{\varepsilon} = 0.3$ for the first experiment and $r_a = 0.15$, $r_b = 0.2$, $\bar{\varepsilon} = 0.5$ and $\underline{\varepsilon} = 0.15$ for the second experiment.

4.1 Synthetic Linear Data Set

The aim of this experiment is to evaluate the performance of the proposed algorithm in the presence of noise. For this purpose, we performed experiments on a synthetic linear data set with simulated noise using Gaussian noise. The samples of 100 and 20 observations were generated as the training and test data sets, respectively, using the linear model of Eq. 9 as follows:

$$Y_i = 0.5X_i \tag{9}$$

Where i varies from 1 to the size of data set and X_i is generated from a standard normal distribution $N(0,1)$.

Then noise was added at different levels to the output values of the instances in the training set as described by Eq. 10 as follows:

$$Y_i \leftarrow Y_i + \varepsilon_i \tag{10}$$

where ε indicates a noise term from a normal distribution with zero mean and variance of 0.5; $N(0,0.5)$. We define the noise level to be the probability that the output values of the instances in the training set were corrupted by the noise. It should be noted that the output values of the instances in the test set are not noisy. The simulations were conducted with the levels of the noise ranging from $p = 0$, 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 in order to predict the output values of the test data set using both the KNN and CKNN models.

The RMSE and MAE results of each model for different levels of the noise are reported in Table 1. In the noiseless case, the CKNN model gives similar results to

Table 1 Comparison of the performance of the models in terms of RMSE and MAE for the different levels of noise for the linear data set

Noise level	RMSE		MAE	
	KNN	CKNN	KNN	CKNN
0	0.045701	0.045701	0.034833	0.034833
0.05	0.073589	0.045716	0.050421	0.034943
0.10	0.096044	0.045721	0.07634	0.034976
0.15	0.100488	0.045743	0.080357	0.034992
0.20	0.134972	0.065319	0.105489	0.044457
0.25	0.133257	0.094911	0.117494	0.057933
0.30	0.205223	0.187613	0.160952	0.149625

the KNN which are the best among all cases listed in the table. The RMSE and MAE of the results generally increase for the KNN model as the noise increases while they remain approximately constant for the CKNN model when the noise in the training data is increased from 0 to 0.15. In these cases, the CKNN can recognize the noise data and prune them among the neighbors before prediction. However, the CKNN model gradually approaches the KNN model as the noise level in the training set is increased to 0.3 and more. This is due to the fact that CKNN can not distinguish noisy instances from non-noisy instances when the number of noisy instances approaches to the number of non-noisy instances. This may be attributed to the fact that the CKNN assumes that noisy instances in training data are so little in which they are not grouped into any cluster at the clustering stage. This assumption is usually consistent with real systems where the number of correct data is much more than noisy data.

As a graphical illustration, Fig. 1a demonstrates the data points in the training set with the noise level of 0.2. As shown in this figure, the noisy data which do not follow the exact linear function can be simply distinguished from the correct data. Scatter plots between the real and computed outputs by the KNN and CKNN models over the test data are shown in Fig. 1b, c, respectively. From the figures, it is observed that the CKNN shows a closer match between the real and the predicted output values of the test data.

The results reveal how well the proposed model is able to predict the correct output when the training data set contains noise. Moreover, the traditional KNN is more sensitive to the noise level in the training set than the CKNN model and the accuracy of KNN's prediction decreases more quickly than does CKNN's as the level of the noise increases.

4.2 Daily Flow Forecasting

In order to demonstrate the effect of the proposed algorithm to capture the inconsistent data points, daily flow forecasting for the Karoon catchment is considered. The North Karoon River basin, with an area of 25,000 km^2, is located in the Southwest of Iran. There exist multiple reservoirs on the Karoon River in series designed mainly for power generation. Daily reservoir inflow predictions are essential to the operational planning and scheduling of these hydroelectric power systems. In this study, the Karoon1 reservoir was taken as an example and daily inflow forecasting for lead time of one day at Soosan gauging station located upstream of the dam was considered (Fig. 2).

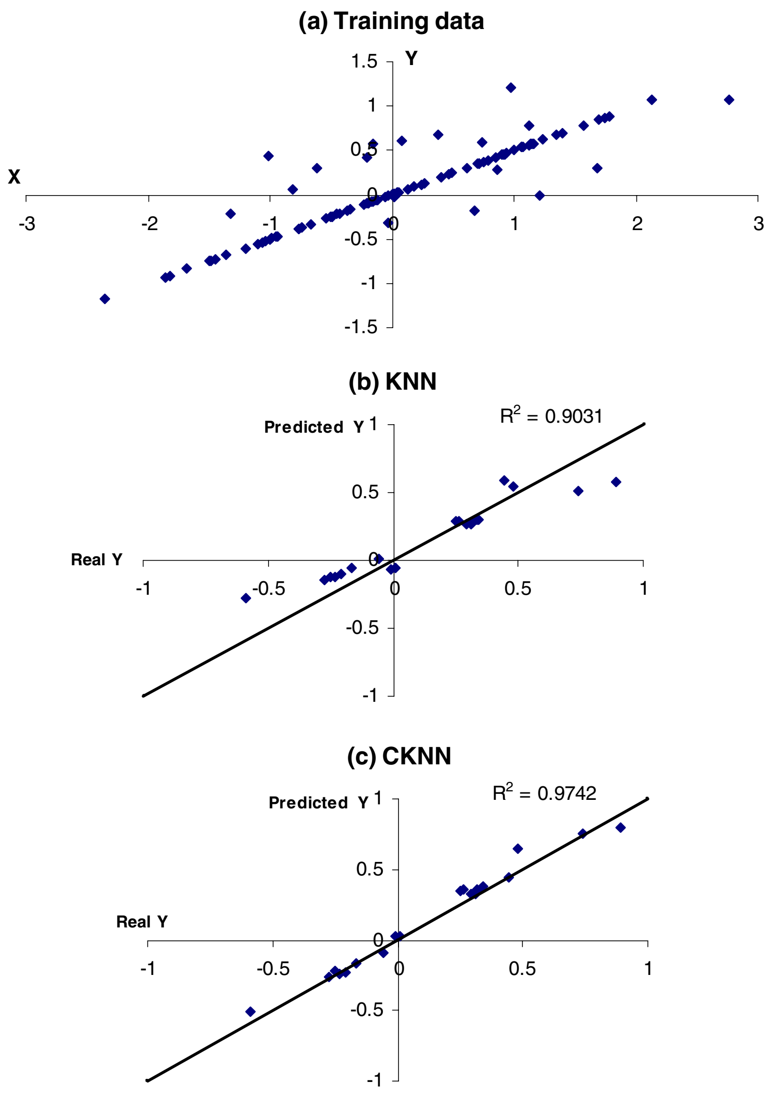

Fig. 1 Scatter plots for **a** the training data set corrupted by noise at the level of 0.2, **b** KNN and **c** CKNN models over the test data of the synthetic linear data experiment

Rainfall and flow data from some gauging stations located in the catchments are available and may be used to forecast the inflow to the Karoon1 reservoir.

As pointed out by Karlsson and Yakowitz (1987), a remarkable deficiency of KNN is its disability in predicting values higher than the historical discharges. However, for daily management purpose in which the interest is not centered on extreme values, it is viable (Wu et al. 2008). In a data driven model, an important step is the choice of input variables which will efficiently represent the modeled system. Bowden et al. (2005) reviewed different methods for choosing input variables of data driven, particularly ANN models. Cross-correlation (CC) and average mutual

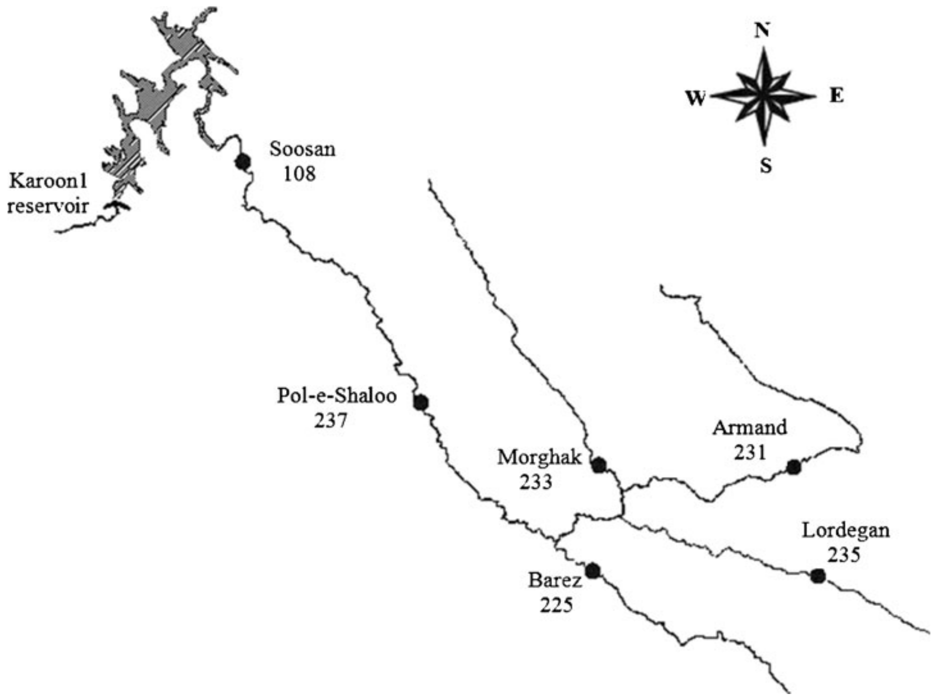

Fig. 2 The Karoon1 reservoir and the upstream gauging stations

information (AMI) analyses are the common techniques which are often employed for selecting appropriate inputs (Imrie et al. 2000; Sharma 2000; Abebe and Price 2003; Bhattacharya and Solomantine 2005; El-Shaıe et al. 2007, 2009; Mishra 2009; Solomatine et al. 2008). The major disadvantage of CC is its failure in detecting nonlinear dependences between variables. So in the cases of real physical systems in which nonlinear dependence may exist between inputs and outputs, CC may result in omission of important inputs carrying valuable knowledge (Bowden et al. 2005). Unlike CC, AMI is not restricted with linear functions; hence it may efficiently account for both linear and nonlinear dependencies between variables. AMI is based on Shannon's entropy (Shannon 1948) and is a measure of information that can be learned about one data set from another known data set. The AMI between two measurements x_i and y_j drawn from sets X and Y is defined by:

$$AMI_{XY} = \sum_{x_i y_j} p_{XY}\left(x_i, y_j\right) \log_2 \left(\frac{p_{XY}\left(x_i, y_j\right)}{p_X\left(x_i\right) p_Y\left(y_j\right)} \right) \tag{11}$$

where $p_{XY}(x_i, y_j)$ is the joint probability density for the measurements X and Y resulting in values x_i and y_j and $p_X(x_i)$ and $p_Y(y_j)$ are the individual probability density for the measurements of X and Y. If the measurement X resulting in x_i is completely independent of the measurement Y resulting in y_j, then the average mutual information AMI_{XY} is zero.

To continue the application process, one has to find out the appropriate attributes for flow forecasting model at Soosan gauging station. Therefore, the AMI between

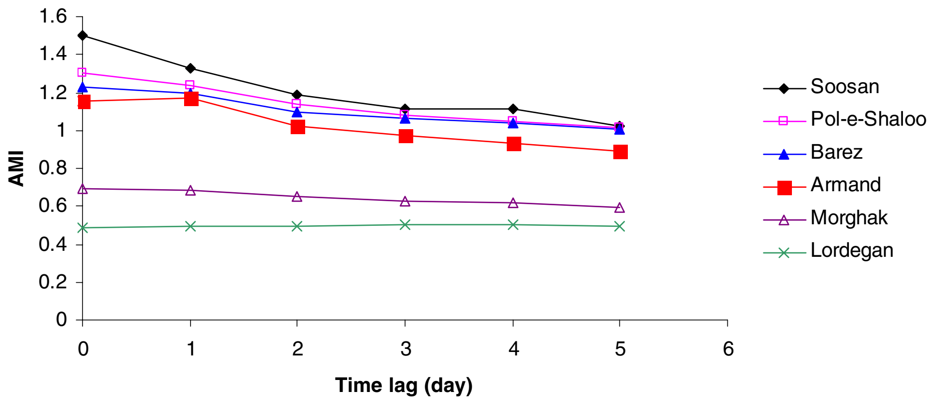

Fig. 3 Average mutual information between the flow at Soosan gauging station and the flows at the different gauging stations

the flow at Soosan gauging station and the flow and rainfall data from different gauging stations were computed up to a lag time of 5 days, as shown in Figs. 3 and 4, respectively. From Fig. 3, it is observed that in addition to the lagged flow at Soosan gauging station, previous flows at Pol-e-Shaloo, Armand and Barez gauging stations contain substantial information about the current flow at Soosan gauging station. However, the information content from two other gauging stations, namely Morghak and Lordegan is relatively little. Figure 4 shows the AMI values for the rainfall data from different gauging stations at varying lag times. The same figure shows that the rainfall data with a lag of 1 day at Barez and Lordegan gauging stations have the highest AMI values.

Although this approach does not directly determine the input parameters for the forecasting model, it does provide a useful guidance for recognizing the main input parameters to the model. Basically, the number and the type of input parameters may change depending on the kind of model being used (Sharma 2000). In this study,

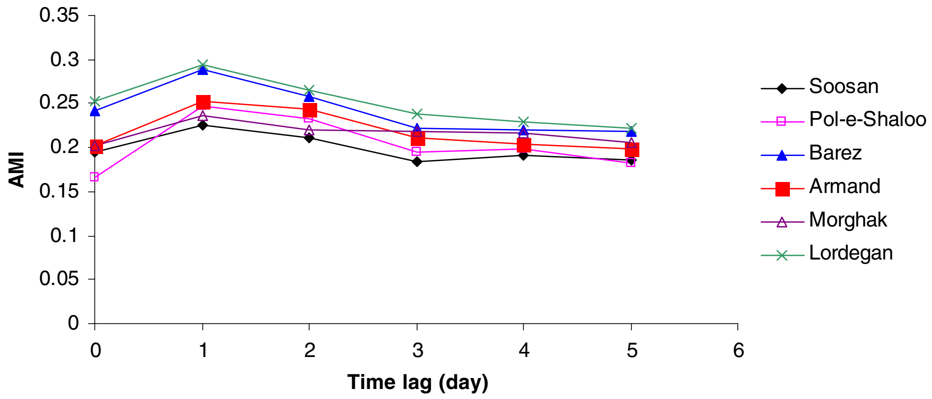

Fig. 4 Average mutual information between the flow at Soosan gauging station and the rainfalls at the different gauging stations

the different combinations of the main input parameters (i.e., rainfall and flow at different gauging stations) were tested using an efficient search procedure. The most suitable input subset with the smallest RMSE over the calibration data was identified and the best model was structured as follows:

$$\begin{aligned} Q_{108}(t) = f\{&Q_{108}(t-1), Q_{108}(t-2), Q_{237}(t-1), \\ &Q_{231}(t-1), Q_{225}(t-1), R_{235}(t-1)\} \end{aligned} \tag{12}$$

In which, Q and R are flow and rainfall respectively and the subscripts represent the national code number of gauging stations in Iran's telemetry system (see Fig. 2). Although the lagged rainfall at Lordegan station revealed relatively smaller AMI, compared to the lagged runoffs at Morghak and Lordegan stations which are not appeared in Eq. 12, it finally appeared as the model's input because of the highest fitness over the calibration data. Further investigation revealed the significant relation between the lagged rainfall and flow at Soosan gauging station especially for the peak flows.

Note that the input determination process, both at the AMI and fitness evaluations, is justified over all calibration data. The global evaluations provide the input parameters showing the highest impact on the overall performance of the models, however, some of the available data points in the calibration data set may result in the outputs which do not follow the selected inputs (attributes). In fact, the output values of these inconsistent data points may be a function of some other attributes which are not considered. In the KNN model for some query instances, the inconsistent points may appear as the neighbors with the similarity measure based on the selected attributes. The inconsistent points may not be the real neighbors to the query instance deteriorating the output estimation. The proposed clustered KNN is developed to identify the inconsistent points and removes them from the neighbors before estimation, thereby locally improving the prediction accuracy.

To develop the KNN and CKNN models, the available data are organized into pairs of feature vector X and output value of Y in which X is a six dimensional vector consisting of lagged rainfall and inflow variables from different gauging stations and Y represents the value of the predicted flow at Soosan gauging station for the next day. Table 2 shows the statistical parameters (i.e., minimum value: x_{min}, maximum value: x_{max}, mean: x_{mean}, and standard deviation: s_x) for the daily flow data set of Soosan gauging station. As is observed from the table, the data are splitted into the calibration and verification data sets with relatively similar statistics. In order to verify that both data sets (calibration and verification) have the similar distributions of low, medium and high flows, a chi-square test was used based on the relative frequencies of both data sets. The results showed that there is not a significant difference in the frequencies of low, medium and high flows of the calibration data as compared with the verification data at the 5% level.

Table 2 The statistical parameters for the data set of Soosan gauging station

Calibration					Verification				
Period	X_{min}	X_{mean}	X_{max}	S_x	Period	X_{min}	X_{mean}	X_{max}	S_x
1994–2000	79.8	285.5	2479	223.9	2001–2003	69.6	272.8	2233	217.9

X: flow (m^3/s)

Table 3 Comparison of the performance of the models in terms of RMSE and MAE for the different ranges of flow during the verification period

Range of flow	MAE			RMSE		
	ANN	KNN	CKNN	ANN	KNN	CKNN
Full range	29.79	36.90	27.14	58.93	72.78	52.17
Low range	18.43	12.29	9.22	24.45	24.40	18.28
Medium range	36.70	70.32	51.77	61.42	107.30	74.26
High range	167.51	151.55	116.46	227.51	176.08	138.37

Furthermore, an ANN model was used in the present experiment as a convenient benchmark against which the performances of the KNN and CKNN can be compared. Besides the lagged rainfall at Barez station, as seventh attribute, the input vector used in this model is the same as that used for the KNN and CKNN models.

Apart from the performance evaluation of the models over the whole test data set, their performances on the different ranges of flow were also considered. For this purpose, the magnitude of flow at the verification period was divided into low ($X <= X_{mean}$), medium ($X_{mean} < X <= X_{mean} + 2S_x$) and high magnitude ($X > X_{mean} + 2S_x$) flows. The number of data points in low, medium and high flows is 693, 363 and 39 respectively.

Table 3 compares the results of the models in terms of RMSE and MAE over the verification data set on the different ranges of flow. The results show that although the overall performance of the ANN seems better than the traditional KNN, the KNN is more effective than the ANN especially for high flows which might be of more interest in reservoir operation. The main superiority of the ANN is at the forecasting of the medium range flows as compared to the KNN model, while the ANN is not as successful at the forecasting of the high flows.

From the results listed, it is also concluded that the CKNN can efficiently improve the efficiency of the KNN in all ranges of flow so that the CKNN is found to be more efficient than the ANN on the whole test data set. The MAE value of the CKNN model on the full range of flow is 27.14 m^3/s which is 26.5% lower than that of the KNN model. In addition, the RMSE value of the CKNN model shows 28.3% improvement over the traditional KNN model. The results demonstrate that the proposed clustered KNN model can improve the original KNN model by efficiently identifying and removing the inconsistent data points of the training data set.

As a graphical demonstration, the scatter plots of the observed and computed flow values for the whole test data set for different models are presented in Fig. 5. These plots give a clear indication of relative capability of each of the models over the full range of flows.

In order to illustrate how an inconsistent point can deteriorate the prediction, an example from the verification data set is selected together with the nearest neighbors and a prediction is made by the KNN and CKNN. As presented in Table 4, the prediction for day 30 in March 2002 is based on the hydrological situation on the days listed in the table as the neighbors when the rainfall and flow records were similar.

Figure 6 is the output of the subtractive clustering algorithm which puts the similar neighbors in the feature vector and the output value into a cluster. It is observed that the neighbor 2 is an inconsistent neighbor which does not sit inside of the existing clusters. The CKNN ignores this neighbor which has resulted in the improvement of the forecasting as compared to the KNN.

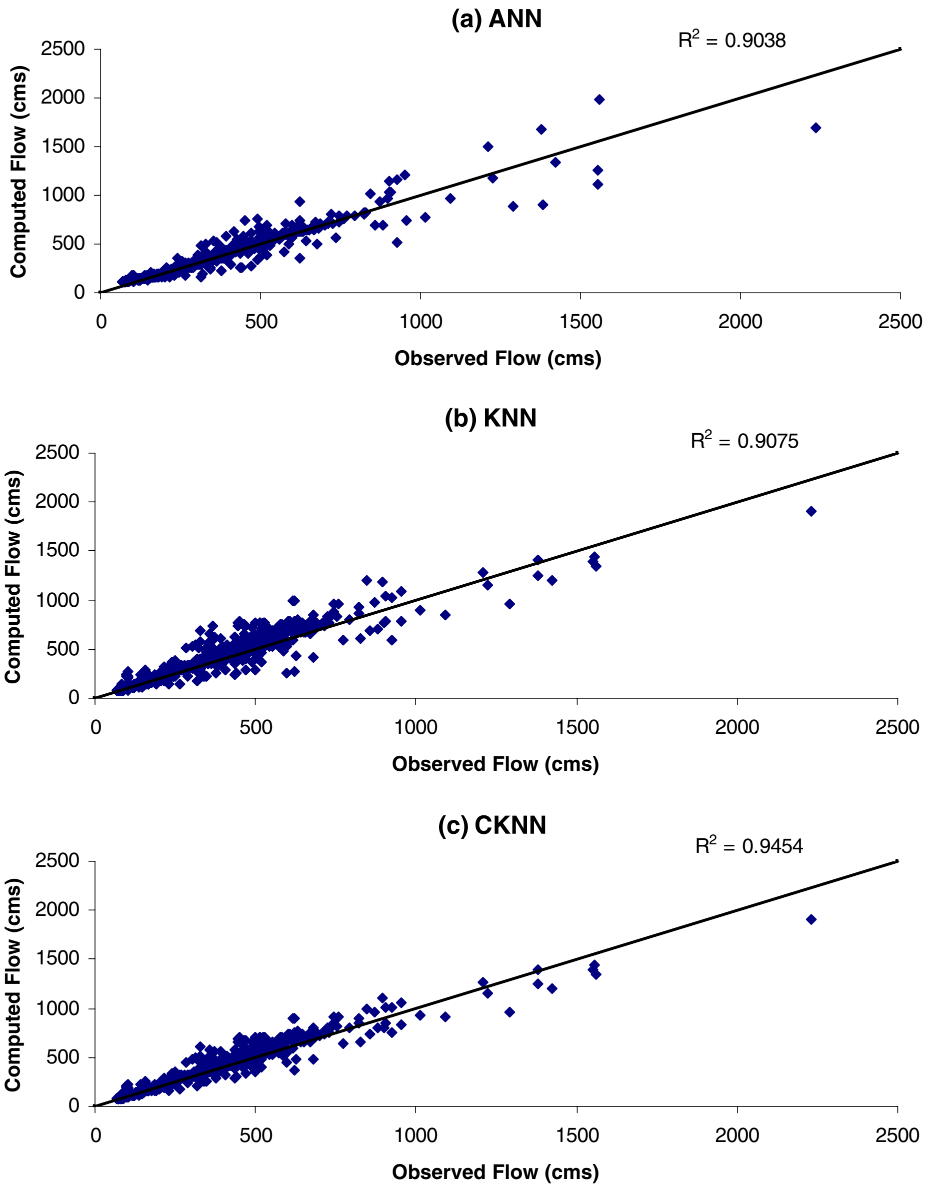

Fig. 5 Scatter plots for **a** ANN, **b** KNN and **c** CKNN models over the verification data

It should be noted that the proposed method can be more efficient where the size of the calibration data set is large enough. In the case of relatively small data set, the output values of neighbors may be naturally far from each other. So, applying the CKNN can result in the false elimination of some neighbors and consequently ignoring the valuable information of these data points.

As a final point, it should be emphasized that the inclusion of a clustering algorithm does not significantly increase the computing time in the CKNN, even

Table 4 Example of the nearest neighbors of a query instance for the daily flow data set

Point	Date	Attributes						Output	Prediction	
		Q_{108} (t–1)	Q_{108} (t–2)	Q_{237} (t–1)	Q_{231} (t–1)	Q_{225} (t–1)	R_{235} (t–1)	Q_{108} (t)	KNN	CKNN
Query	30.03.2002	733	438	860	274	439	39	596	853	668
Ne. 1	13.03.1994	679	364	767	153	172	58	814		
Ne. 2	06.04.1997	880	575	810	103	304	23	1870		
Ne. 3	06.05.1997	598	586	605	292	154	17.5	729		
Ne. 4	20.03.1996	763	654	789	214	290	0	816		
Ne. 5	19.03.1996	654	721	638	169	262	4	763		
Ne. 6	21.04.1995	486	460	504	177	192	0	563		
Ne. 7	26.03.2000	716	440	843	130	116	22	712		
Ne. 8	07.05.1997	729	598	854	286	247	0.5	682		
Ne. 9	19.03.1995	513	636	818	186	251	2	438		
Ne. 10	03.04.1994	543	340	717	228	128	13	496		

Ne. : neighbor

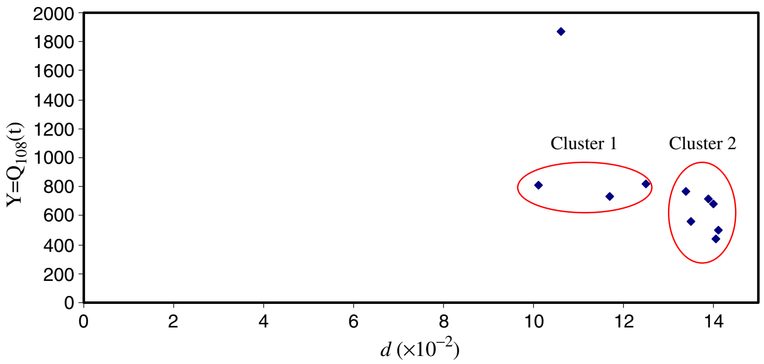

Fig. 6 Clustering of neighbors of a query instance included an inconsistent point as a neighbor

when using a large data set as a training set. As an example, for the case study considered, the computational time for each year of the test data set was 15 seconds for the KNN model while it was 35 seconds for the CKNN model which indicates the CKNN is still efficient from the computational time's point of view.

5 Conclusions

This paper presents a clustered K nearest neighbor algorithm (CKNN) which can capture the inconsistent data as well as noisy data points. These points may appear as the neighbors of some query instances in the KNN model deteriorating the results of forecasting while they are not really neighbor to the query instance. The proposed model recognizes these points, prunes them among the neighbors, and then forecasting is done based on the remainder of the neighbors. Experiments conducted over the different ranges of flow data of the Karoon1 reservoir, concluded that the forecasting accuracy in the CKNN is much higher than that of the KNN model. Furthermore, a synthetic linear data set was used to study how noise in the training data set affects the predictive ability with respect to the original KNN. The simulations showed that the CKNN model is more robust and maintains the good predictive ability in the presence of noise in the training data set.

The proposed idea about recognizing the inconsistent and noisy data by a clustering algorithm can be also applied in the LWR model as another type of instance based learning algorithm, but it is clear it can not be used in the NN where the forecasting is based on only the output value of the nearest neighbor.

Acknowledgements The authors would like to thank Professor Nick van de Giesen, the head of Water Management section, Faculty CEG, Delft University of Technology. The authors also acknowledge the helpful comments of the two anonymous reviewers and Mr. M. Fazel.

References

Abebe AJ, Price RK (2003) Managing uncertainty in hydrological models using complementary models. Hydrol Sci J 48(5):679–692

Aha DW, Goldstone RL (1992) Concept learning and flexible weighting. Proceedings of 14th Annual Conference of the Cognitive Science Society. Mahwah, USA, 534–539

Atkeson CG, Moore AW, Schaal S (1997) Locally weighted learning. Artif Intell Rev 11(5):11–73

Bhattacharya B, Solomantine DP (2005) Neural networks and M5 model trees in modeling water level–discharge relationship. Neurocomputing 63:381–396

Bowden GJ, Dandy GC, Maier HR (2005) Input determination for neural network models in water resources applications, Part 1, Background and methodology. J Hydrol 301:75–92

Chiu S (1994) Fuzzy model identification based on cluster estimation. J Intell Fuzzy Syst 2:267–278

Coulibaly P, Haché M, Fortin V, Bobée B (2005) Improving daily reservoir inflow forecasts with model combination. J Hydrol Eng, ASCE 10(2):91–99

Czarnowski I, Jędrzejowicz P (2006) Instance reduction approach to machine learning and multi-database mining. Annales UMCS Informatica AI 4:60–71

El-Shaıe A, Taha MR, Noureldin A (2007) A neuro-fuzzy model for in?ow forecasting of the Nile river at Aswan high dam. Water Resour Manag 21:533–556. doi:10.1007/s11269-006-9027-1

El-Shaıe A, Abdin AE, Noureldin A, Taha MR (2009) Enhancing in?ow forecasting model at Aswan high dam utilizing radial basis neural network and upstream monitoring stations measurements. Water Resour Manag 23:2289–2315. doi:10.1007/s11269-008-9382-1

Galeati G (1990) A comparison of parametric and non-parametric methods for runoff forecasting. Hydrol Sci J 35(1):79–94

Imrie CE, Durucan S, Korre A (2000) River flow prediction using artificial neural networks: generalization beyond the calibration range. J Hydrol 233:138–153

Jang JSR, Sun CT, Mizutani E (1997) Neuro-fuzzy and soft computing, a computational approach to learning and machine intelligence. Prentice Hall

Kang P, Cho S (2008) Locally linear reconstruction for instance-based learning. Pattern Recogn 41:3507–3518

Karlsson M, Yakowitz S (1987) Nearest neighbor methods for non parametric rainfall runoff forecasting. J Wat Resour Res 23(7):1308–1330

Lall U, Sharma A (1996) A nearest neighbor bootstrap for time series resampling. J Wat Resour Res 32:679–693

Mishra S (2009) Uncertainty and sensitivity analysis techniques for hydrologic modeling. J Hydroinform 11(3–4):282–296

Ruprecht D, Müller H (1994) A framework for generalized scattered data interpolation. Technical Report no. 539, Universit¨at Dortmund, Fachbereich Informatik, D-44221 Dortmund, Germany

Shamseldin AY, O'Connor KM (1996) A nearest neighbor linear perturbation model for river flow forecasting. J Hydrol 179:353–375

Shannon CL (1948) A mathematical theory of communication. Bell System Tech J 27:379–423 and 623–656

Sharma A (2000) Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: Part 1—a strategy for system predictor identification. J Hydrol 239:232–239

Solomatine DP, Maskey M, Shrestha DL (2008) Instance-based learning compared to other data-driven methods in hydrological forecasting. Hydrol Process 22:275–287

Wand MP, Schucany WR (1990) Gaussian-based kernels. Can J Stat 18(3):197–204

Wu CL, Chau KW, Li YS (2008) River stage prediction based on a distributed support vector regression. J Hydrology 358:96–111

Yakowitz S (1987) Nearest-neighbor methods for time series analysis. J Time Ser Anal 8(2):235–247