IQ Estimation for Accurate Time-Series Classification

Krisztian Buza, Alexandros Nanopoulos, Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
{buza,nanopoulos,schmidt-thieme}@ismll.de

***Abstract*—Due to its various applications, time-series classification is a prominent research topic in data mining and computational intelligence. The simple k-NN classifier using *dynamic time warping* (DTW) distance had been shown to be competitive to other state-of-the art time-series classifiers. In our research, however, we observed that a single fixed choice for the number of nearest neighbors k may lead to suboptimal performance. This is due to the complexity of time-series data, especially because the characteristic of the data may vary from region to region. Therefore, local adaptations of the classification algorithm is required. In order to address this problem in a principled way by, in this paper we introduce individual quality (IQ) estimation. This refers to estimating the expected classification accuracy for each time series and each k individually. Based on the IQ estimations we combine the classification results of several k-NN classifiers as final prediction. In our framework of IQ, we develop two time-series classification algorithms, IQ-MAX and IQ-WV. In our experiments on 35 commonly used benchmark data sets, we show that both IQ-MAX and IQ-WV outperform two baselines.**

***Index Terms*—time series; classification; individual quality (IQ)**

I. Introduction

Classification of time series is a prominent research topic in data mining and computational intelligence, because of its numerous applications in various domains such as speech recognition [1], signature verification [2], brainwave analysis [3], handwriting recognition, finance, medicine, biometrics, chemistry, astronomy, robotics, networking and industry [4].

The simple 1-nearest neighbor (1-NN) method using dynamic time warping (DTW) distance [1] has been shown to be competitive or superior to many state-of-the art time-series classification methods [5], [6], [7]. However, the choice of parameter k in the k-NN classifier is known to affect the bias-variance trade-off [8]: smaller values of k may lead to overfitting, whereas larger values of k increase the bias and in this case the model may capture only global tendencies. Recent studies [9] have indicated that significant improvement in the accuracy of the k-NN time-series classification can be attained with k being larger than 1. This is due to intrinsic characteristics in time-series data sets, such as the mixture between the different classes, the dimensionality, and the skewness in the distribution of error (i.e., the existence of "bad hubs" [9] that account for a surprisingly large fraction of the total error). Parameter k can be chosen using a hold-out subset of the training data.

In complex time-series data sets, the intrinsic characteristics, such as the ones mentioned above, may vary from region to region. As a consequence, setting a single, global choice for k (≥ 1) can become suboptimal, since each individual region of a data set may require a different value of k.

In order to address the above problem in a principled way, and allow for accurate k-NN classification of complex time-series data, we propose individual quality (IQ) estimation. IQ estimation is a mechanism that considers a range of values for k and estimates for each time-series, t, that has to be classified, the local quality of each k-NN classifier. With local quality we mean the likelihood of correct classification of t by the k-NN classifier. This way, a quality score $q(t,k)$ is assigned to each pair $(t, k\text{-NN})$ of a time series t and a k-NN classifier.

This information is then used by a meta level decision method that combines the predictions of k-NN classifiers. In our first approach, IQ-MAX, for each time series t to be classified, the meta level decision method selects those k that maximizes the estimated quality. As the quality estimation is done for each time series t individually, for different time series, different k values can be selected.

In our second approach, IQ-WV, the outputs of the different k-NN classifiers are weighted according to the estimated quality of the k-NN classifiers.

As we propose the classification of time-series based on a quality score estimated individually for each of them, the proposed approach is called time-series classification based on *individual quality (IQ) estimation*. IQ estimation, in particular the calculation of the quality score, is performed by regression models that are trained in order to make accurate estimations for the quality of the k-NN classifier.

In summary, our contribution can be described as follows: (a) We introduce IQ estimation. This is a general technique, that can be applied in context of many different classification problems and algorithms (i.e., not just for the k-NN classification of time-series). (b) We propose a novel method for IQ estimation. We apply it to estimate k-NN classifiers' quality for the task of classifying time series. (c) In the IQ-estimation framework, we propose two approaches, IQ-MAX and IQ-WV that use the output of IQ estimation in two different ways in order to make time-series classification more accurate. (d) We perform a thorough experimental evaluation with 35 commonly used benchmark data sets. The results

indicate significant improvement in accuracy attained by the proposed approaches when compared with the widely used 1-NN classifier and with the k-NN classifier that determines a single optimal k ($k \geq 1$).

The rest of this paper is organised as follows: in Section II we overview the related work, in Section III we outline IQ estimation, whereas in Section IV we describe both proposed approaches IQ-MAX and IQ-WV. In Section V we present our experimental evaluation. We provide our conclusions in Section VI.

II. Related Work

By reason of the increasing interest in time-series classification, various approaches have been introduced ranging from neural [10] and Bayesian networks [11] to genetic algorithms, support vector machines [12] and frequent pattern mining [13], [14]. However, the k-nearest neighbor (k-NN) classifier (especially for $k = 1$), has been shown to be competitive to many other, more complex models [5], [6], [7]. Nearest-neighbor classification of time series uses Dynamic Time Warping (DTW) [1], because it is an elastic distance measure, i.e., it is robust w.r.t. shiftings and elongation in the time series. Recent works aimed at making DTW more accurate and scalable [15], [16]. DTW has been examined in depth (a thorough summary of results can be found at [17]), whereas Ding et al. found no other distance measure that significantly outperforms DTW [6].

Our proposed approach, i.e., using IQ estimation for k-NN time-series classification, could be related to works that perform local adaptation of k-NN classifier. A locally adaptive distance measure was proposed by Hastie and Tibshirani [18], while Domeniconi and Gunopulos [19] used SVMs to define a local measure of feature relevance, i.e., feature weights depending on the location of a data point to be classified. In [20] adaptive nearest neighbor classification in high-dimensional spaces was studied. In contrast to these works, our IQ estimation based approaches adapt by selecting the proper value of k (IQ-MAX) and by combining several k-NN classifiers (IQ-WV), but not by determining a localized distance function.

Ougiaroglou at al. [21] presented 3 early-break heuristics for k-NN which can be interpreted as adapting the number of nearest neighbors. Their heuristics, however, aimed at speeding-up k-NN, while we focus on making nearest neighbor classification more accurate using the principled framework of IQ estimation.

Methods for quality estimation (a.k.a. error estimation) are usually applied globally in order to estimate the overall performance of a classification model [22], [23]. In our approach, we focus on individualized quality estimation. This is similar to learning the residuals, i.e., the difference between predicted and actual labels. Duffy and Helmbold followed this direction and incorporated residuals into boosting of regression models [24]. In contrast to this work, we do not focus on boosting. Similarly to our work, Tsuda et al. [25] proposed an individualized approach for estimating the leave-one-out error of vector classification with support vector machines (SVM) and linear programming machines (LPM). Compared to this work, our proposed approach performs general IQ estimation (not just for leave one out). More importantly our approach *exploits* IQ estimation to improve accuracy of classification and not as a *per se* task, as done in [25].

A set of earlier approaches to localized quality estimation for the k-NN classifier was proposed by Wettschereck and Dietterich [26]. However, these approaches were based solely on heuristics such as using different k values per class or per cluster (after clustering the training set). Our proposed framework is more principled and more generic than these simple approaches: we distinguish between the quality estimation step and classification step, our framework supports systematic usage of the estimated quality, and our framework allows various classification and regression models. Furthermore, while predicting a class, our IQ-WV method can involve arbitrary number of models and arbitrary number of meta models, whereas Wettschereck and Dietterich [26] use fixed number models (mostly just one selected model) at the elementary level, while they use heuristics instead of meta models.

Finally, the aforementioned works concerned with classification of vectors (point data), while we focus on time-series classification.

III. Individual Quality Estimation

In this section, we introduce the concept of IQ estimation, which is the basis of the proposed algorithms that will be detailed in Section IV. We first provide a motivating example and then outline the approach we take for IQ estimation.

A. Motivating Example

As mentioned in Section I, the selection of a single value of k for the k-NN time-series classification, can lead to sub-optimal accuracy, because of varying characteristics among different regions of the data. We investigate this phenomenon in more detail by first presenting a motivating example for the simple setting of binary classification of a 2-dimensional data set.[1]

Figure 1 depicts a set of labeled instances from two classes that are denoted by triangles and circles. The density in the class of triangles (upper region) is larger than in the class of circles (lower region). We consider two test instances, denoted as '1' and '2', that have to be classified. We also assume that the ground-truth considers test instance '1' as a triangle, whereas '2' as a circle. For '1', its 1-NN is a circle. Thus, the 1-NN method classifies '1' incorrectly. Using the k-NN classifier with $k > 1$ (e.g., in the range between 3 and 6), we can overcome this problem. However, the selection of a single k from the above range results in incorrect classification of test instance '2'. Due to the lower density in the circles' class, by setting k so that $3 \leq k \leq 6$, we detect neighbors of '2' whose

[1] In this example, we use a 2-dimensional data set, thus we depart for the moment from the examination of time-series data that are in general high-dimensional, in order to ease the presentation with an illustrative figure.

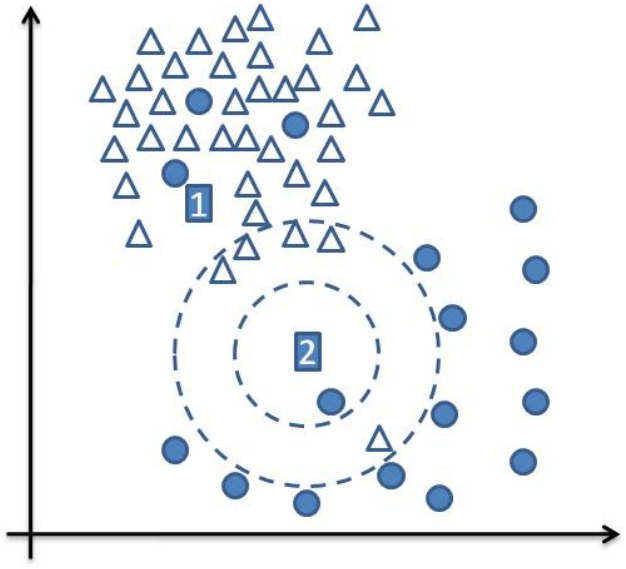

Fig. 1. The optimal choice of the number of nearest neighbors is not unique for the entire data, but it may be different from region to region: in case of the classification of the unlabeled instance denoted by '1', $k > 1$ (e.g., $k = 3$) is required; whereas for '2' we should choose $k = 1$. (We assume that the ground-truth considers test instance '1' as a triangle, whereas '2' as a circle.)

majority belongs to the triangles' class (we assumed '2' is a circle). This can be observed in Figure 1, where the large dashed cycle around '2' shows that among all its 6-NN, only 1 belongs to the circles' class. Thus, unlike for '1', $k = 1$ is a good choice for '2', because its 1-NN (shown inside the smaller dashed cycle) has the correct class.

The exemplified problem is amplified with time-series data due to their higher dimensionality and complexity. We propose to estimate the likelihood of correct classification (the quality) of the k-NN classifiers on an individualized basis, i.e., separately for each test instance to be classified we aim at estimating the performance of each classifier. Based on this information, we want to choose the classifiers having the best estimated quality and combine their outputs. Following this approach in the example of Figure 1, besides the k-NN classifier, we need an additional model, which will allow for predicting that $k_1 = 3$, $k_1 = 4$, $k_1 = 5$ and $k_1 = 6$ are good choices (i.e. the likelihood of correct classification is high), when we classify instance '1'; whereas $k_2 = 1$ is an appropriate choice for the classification of instance '2'. In the following we outline how the proposed approach can be developed.

B. IQ Estimation for Classification: IQ-MAX and IQ-WV

We propose a mechanism for individualized quality (IQ) estimation for k-NN classifiers, its schema is depicted in Fig. 2. This mechanism for IQ estimation considers a range of values for k.[2] This examined range of n values for k is denoted as $\{k_i\}_{i=1}^{n}$. For each k_i-NN classifier and for each time-series t, that has to be classified, we estimate the local quality: the quality score $q(t, k_i)$ denotes the likelihood that the k_i-NN classifier ($1 \leq i \leq n$) will correctly classify t.

For IQ estimation, i.e. for the calculation of the quality scores $q(t, k_i)$, we introduce a second layer of models. We refer to the models in this second layer as meta models, denoted as $M^*_{i,j}$ in Fig. 2. These meta models are regression models that are trained to predict the likelihood of correct classification of each considered k_i-NN classifier. For each k_i-NN classifier, we train several meta models, and we use the median of their outputs as quality score. Instead of the median we could also use the average, however, we decided to use the median because it is generally known to be more stable than the average.

In our first approach, IQ-MAX, for each time series t to be classified, we select $k^* \in \{k_i\}_{i=1}^{n}$ that maximizes estimated quality: $k^* = \text{argmax}_{k_i, 1 \leq i \leq n}\{q(t, k_i)\}$. Finally, the k^*-NN classifier is used to classify t. This is shown in Fig. 3.

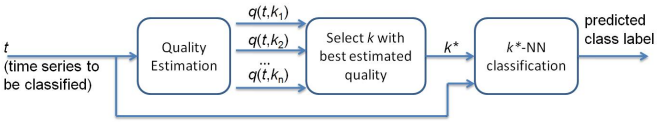

Fig. 3. Summary of IQ-MAX approach for k-NN classification.

In our second approach, IQ-WV, the labels predicted by the k_i-NN classifiers are combined according to a weighted voting schema, we use the quality scores as weights. Formally, we can describe this approach as follows. Let M_l^t denote the set of k_i-NN classifiers (models) that output label l as predicted class label for a time series t. Denoting the class label predicted by the k_i-NN classifier for the time series t as $y_{k_i}(t)$, we can write:

$$M_l^t = \{k_i | y_{k_i}(t) = l\}$$

We can calculate w_l^t, the weight of label l when classifying time series t, as the sum of the quality scores associated to those k_i-NN classifiers that predict l as class label:

$$w_l^t = \sum_{k_i \in M_l^t} q(t, k_i)$$

As final result of the classification of time series t we select the class label having maximal weight: $y(t) = \text{argmax}_l\{w_l^t\}$.

Note that in terms of Fig. 2, IQ-MAX and IQ-WV differ only in the meta-level decision method. In case of IQ-MAX this meta-level decision method consists of selecting that k_i-NN classifier which is expected to be the best. In case of IQ-WV, respectively, the meta-level decision method is realized as weighted voting using $q(t, k_i)$ as weights of the respective predicted class labels $y_{k_i}(t)$.

[2]Although this range is user-defined, its determination is much simpler and intuitive compared to selecting a single k. This will be asserted by our experimental results, which indicate that the range $1 - 10$ was appropriate for all examined benchmark data sets.

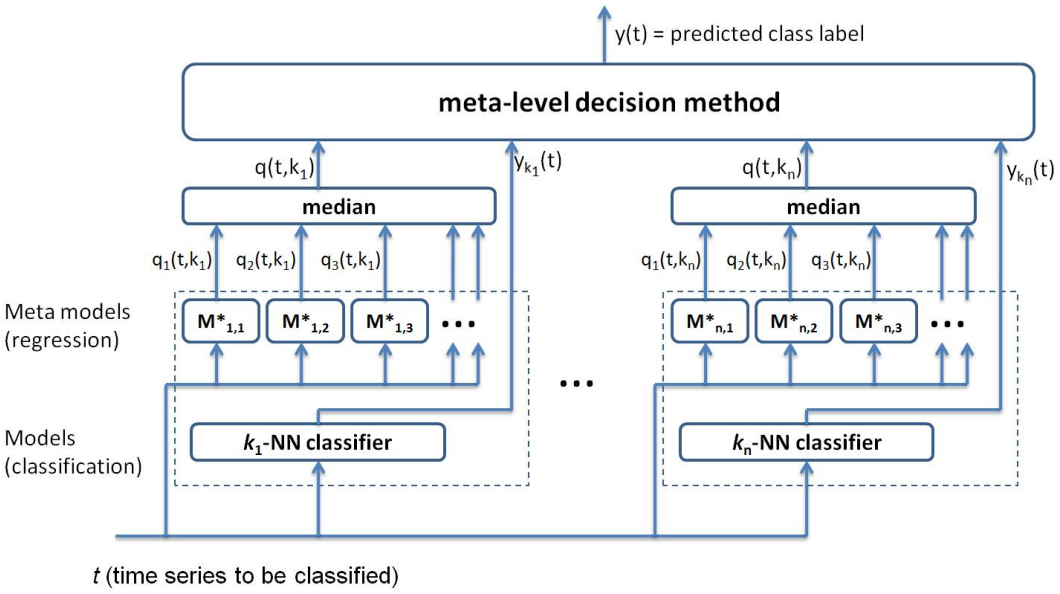

Fig. 2. Classification based on IQ estimation

A concrete algorithm for times-series classification is developed in the following section, by specifying the secondary models that perform IQ estimation.

IV. Time-Series Classification Based on IQ Estimation

The proposed mechanism for classification based on IQ estimation involves two types of models:

- *Primary models* (for simplicity we refer to them as *models*, wherever it is not confusing) – they classify time series using the k-NN approach (with the DTW distance).
- *Meta models* – they estimate the quality of the primary models, see $M^*_{i,j}$ in Fig. 2.

To train the meta models, we partition the original training data set, D, in two disjoint subsets D_1 and D_2 (i.e., $D_1 \cup D_2 = D, D_1 \cap D_2 = \emptyset$). D_2 is called hold-out set. For each time series $t \in D_2$, and for each examined value of k_i in a range $\{k_i\}_{i=1}^n$, we use D_1 to classify t with the k_i-NN classifier. Based on the class label of t that is given in D_2, we determine if the k_i-NN classifier (for each $1 \leq i \leq n$) has correctly classified t. In case of correct classification, we associate with t a quality score $q(t, k_i) = 1$, otherwise $q(t, k_i) = 0$. Thus, from the hold-out set D_2 we can generate n new data sets D'_i, $1 \leq i \leq n$. Each D'_i contains all time-series of the hold-out set D_2 along with their associated quality scores (1 or 0) for the corresponding k_i-NN classifier:

$$D'_i = \{\forall t \in D_2 : (t, q(t, k_i))\}$$

Next, each generated D'_i acts as the training set for the corresponding meta models. Thus, based on the associated quality scores in each D'_i, the corresponding meta models are trained as regression models in order to be able to predict the quality score of the k_i-NN classifier (i.e., the corresponding primary model) for new time series. This is summarized in Fig. 2. The quality of each primary k_i-NN classifier is estimated by n' meta models, denoted as $M^*_{i,j}$, $1 \leq j \leq n'$. The estimation outputted by the meta model $M^*_{i,j}$ is denoted as $q_j(t, k_i)$. As already mentioned, we aggregate these quality estimations by taking their median: $q(t, k_i) = \text{median}\{\forall j : q_j(t, k_i)\}$.

We implement each meta model $M^*_{i,j}$ as a k'_j-NN *regression model* based on the DTW distance. (We denote k'_j in order to distinguish from k_i that is used in the primary k_i-NN classification models.) The secondary level prediction for a new time series $t^* \notin D$ is calculated the following way:

$$q_j(t^*, k_i) = \frac{\sum_{t_N \in \mathcal{N}(t^*)} q(t_N, k_i)}{k'_j}$$

where $\mathcal{N}(t) \subset D_2$ is the set of k'_j nearest neighbors of t^* and $q(t_N, k_i)$ is the associated quality score of each $t_N \in \mathcal{N}(t)$ and the k_i-NN classifier.

A. Efficiency Considerations

We have to clarify that the training of meta models is being performed in an off-line fashion, i.e., the process of partitioning the train data into D_1 and D_2 and generating meta level training sets D'_i is performed off-line, independently of the (online) classification of unlabeled (or test) time series.

Regarding the (online) time needed to classify a time series, first note, that the DTW can be calculated fast and nearest neighbors can be found efficiently using recent indexing techniques [16], [27], [28]. Furthermore, we would like to point out that the schema presented in Fig. 2 is a conceptual description of our approach; in order to implement it efficiently, one can exploit an interesting property of nearest neighbor classification and regression which we describe below. Suppose we want to classify a time series t using its $k_1 < k_2 < ... < k_n$ nearest

neighbors.[3] For this task, most of the computational costs are spend for finding the nearest neighbors. However:

1) While we classify t with k_n nearest neighbors, with minimal additional overhead we can produce the classification results for the other cases too, because the sets of $k_1, k_2, ..., k_{n-1}$ nearest neighbors of t, denoted as $\mathcal{N}_{k_1}(t), \mathcal{N}_{k_2}(t), ..., \mathcal{N}_{k_{n-1}}(t)$, are subsets of the k_n nearest neighbors: $\mathcal{N}_{k_1}(t) \subset \mathcal{N}_{k_n}(t)$, $\mathcal{N}_{k_2}(t) \subset \mathcal{N}_{k_n}(t)$, ..., $\mathcal{N}_{k_{n-1}}(t) \subset \mathcal{N}_{k_n}(t)$. Therefore, the nearest neighbors required for k_1-NN, k_2-NN, ... , k_{n-1}-NN classifications can be found fast among the k_n nearest neighbors.
2) Furthermore: if the time series in two data sets are identical, only their class labels differ, and we want to classify a new time series $t*$, we need to determine the nearest neighbors of $t*$ only *once*. This can be exploited in our case, because the train sets of all the meta models consist of same time series (only the class labels differ).

Taking both of the above observations into account, the whole meta level, containing in total $n \times n'$ nearest neighbor regression models, can be implemented at approximately the same computational costs as one single nearest neighbor regression model with $k' = \max\{k'_1, k'_2, ...k'_{n'}\}$. Similarly, all the primary level models together can be implemented at approximately the same computational costs as one single nearest neighbor classification model with $k = \max\{k_1, k_2, ...k_n\}$.

Regarding the training procedure and the respective offline (training) time of our approach, the computationally expensive part of the calculations consist of the classification of the time series of the hold-out data set D_2. The same is done in case if we search for a globally optimal k for the k-NN classifier. Therefore, the execution time of the (offline) training procedure of our approach is the same as the time required for finding a globally optimal k for the k-NN classifier using the hold-out data set D_2.

We summarize this discussion by pointing out that, despite its complex schema, our approach can be implemented efficiently. Assuming such an implementation, the execution times do not drastically differ from that of one single a k-NN classifier. The online time necessary to classify new time series, only increase by a small factor, while the offline (training) time is approximately the same as the time required to find a globally optimal k for one single k-NN classifier.

V. Experimental Evaluation

A. Experimental Configuration

To assist reproducibility, we provide a detailed description of the configuration of our experiments.

Methods. We compare the proposed methods, denoted as IQ-MAX and IQ-WV, against two baselines: the 1-NN classifier and the k-NN classifier that selects k using a hold-out set from the training data. The latter baseline uses the same hold-out set as the proposed method, examines the same range of values for k, and selects the one that produces the smallest average error for all time series in the hold-out set. All examined methods are based on the same DTW distance that constrains the warping window size at 5% around the matrix diagonal [17].

Data sets. Out of all the 38 data sets used in [6], we examined 35 data sets: we excluded 3 of them (Coffee, Beef and OliveOil) due to their tiny size (less than 100 time series). The names of the remaining data sets and their size (number of time series they contain) are listed in the first and second columns of Table I.

Parameters. At the primary level of our both proposed methods, we use k-NN classifiers with all k values in the range $1-10$. We experimented with larger k values as well, but we observed that they increase the bias and deteriorate the resulting accuracy. For both of our proposed approaches, we implement the meta models as k'_j-NN regressors as described in Section IV. For IQ-MAX, in oder to keep the approach simple, we use a single value of $k' = 5$ at the meta level. Our experimental results show that this was appropriate for all the examined benchmark data sets.[4] In case of IQ-WV, we used a range of 1-10 as k' values.

Comparison protocol. We measure the misclassification error using 10-fold cross validation, with the exception of three data sets (FaceFour, Lighting2, and Lighting7) for which we used the leave-one-out protocol due to their small size. In each round of the 10-fold cross validation, out of the 9 training splits, we used 5 to train the primary models (D_1), the rest 4 splits served as hold-out data (D_2).[5] For classifying test data, i.e., after selecting for training IQ-MAX, IQ-VW and selecting the best k for k-NN, we can again use all training splits.

After using the above evaluation procedure, we made a striking observation about the performance of all examined methods (proposed and baselines): in the majority of data sets, the misclassification error was rather low (less than 5%). To have a challenging comparison with non trivial classification, we choose to affect intrinsic characteristics of the data sets. According to the findings in [9], time-series data sets usually have high intrinsic dimensionality and, thus, some of their instances tend to misclassify a surprisingly large number of other instances when using the k-NN classifier ($k \geq 1$). These instances are called "bad hubs" and are responsible for a very large fraction of the total error. For this reason, for each time series, t, in a data set, we measured two quantities: the badness $B(t)$ of t and the goodness $G(t)$ of t. $B(t)$ ($G(t)$, respectively) is the total number of time series in the data set, which have t as their first nearest neighbor while having different (same, respectively) class label from t. For each data set, we sort all time series according to the $G(t)-B(t)$ quantity in descending order. Then we change the label of first p percent time series

[3] In this discussion we assume that each k_i is much smaller than the number of *all* time series in the train data: $\forall k_i : k_i << |D|$.

[4] Note that we also experimented with other single k' values for IQ-MAX. For $k' \geq 5$ we observed similar results, whereas for small values of k', such as 1 or 2, we observed worse performance.

[5] Ratios other than the examined 5-4, gave similar results. In case of leave-one-out, the training data was split according to 5 to 4 proportion into D_1 and D_2.

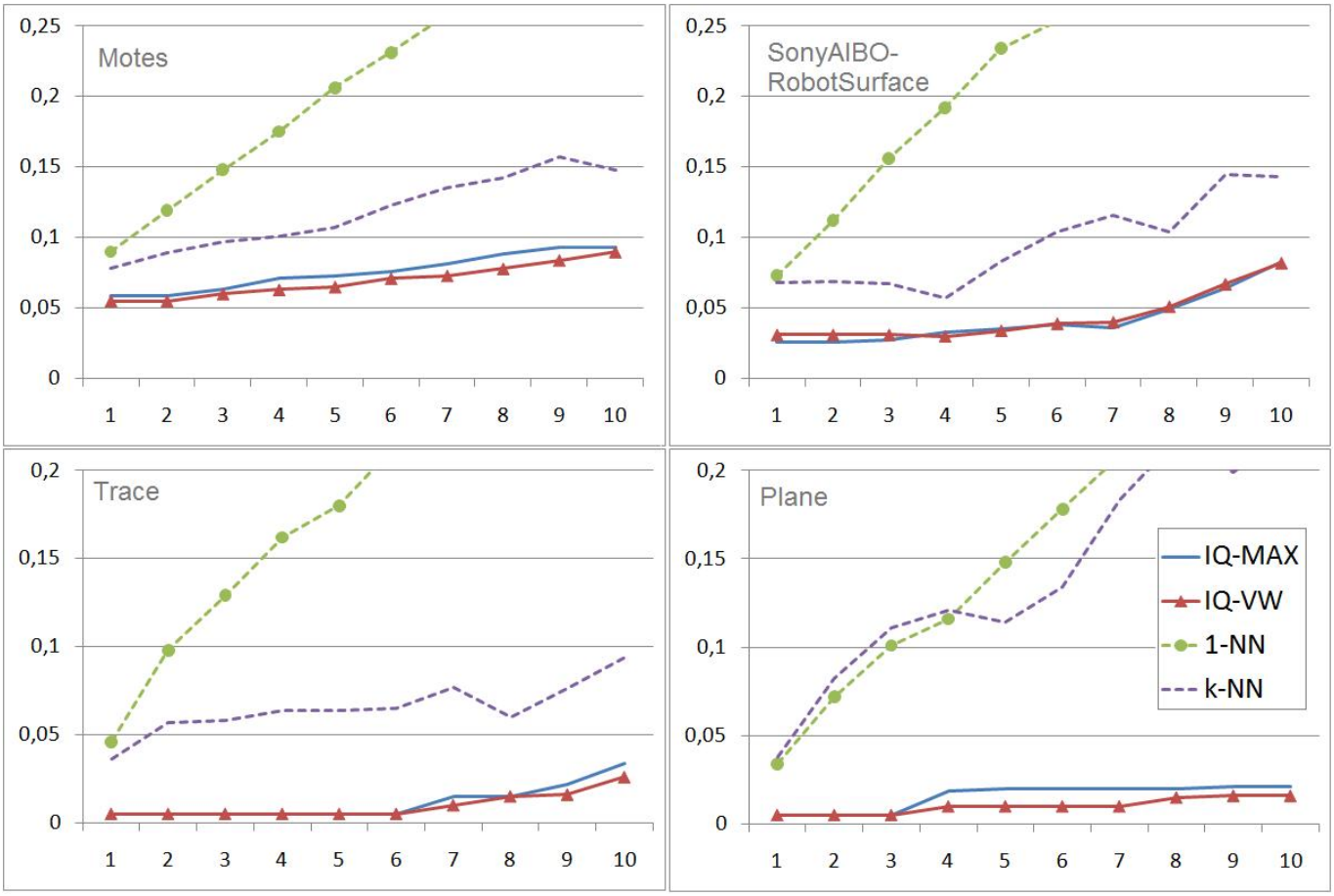

Fig. 4. Classification error (vertical axis) depending on the noise level (horizontal axis) for some data sets.

in this ranking (p varies in range 0-10%).[6] Since the above procedure results in data sets that have stronger "bad hubs" and a less clear separation between classes, the comparison among the examined methods becomes more challenging and can characterize better the robustness of the methods.

B. Experimental Results

The results on classification error are summarized in Table I. For brevity, in Table I we only report results at $p = 1\%$ and 5% noise, however we observed similar tendencies at all other noise ratios in the examined range of p: for four data sets, Motes, SonyAIBORobotSurface, Trace and Plane Fig. 4 shows the classification error for all the examined values of p.

In Table I bold font denotes the best method(s) for each data set, if both of our methods (IQ-MAX and IQ-WV) outperform the baselines, both are marked bold. In case where IQ-MAX and/or IQ-WV are/is superior to the baseline, we also provide two symbols in the form: $\pm/\pm$ to denote the result of statistical-significance test (t-test at 0.05 level) against 1-NN and k-NN, respectively, where a $+$ denotes significance and $-$ its absence. In case where the winner is neither IQ-MAX nor IQ-WV, we provide the result (again in form of $\pm/\pm$) of statistical-significance test of the winner against IQ-MAX and IQ-WV.

[6]The time series whose labels were changed by this procedure, are assigned to an additional class (not included in the original data set). To keep our experimental evaluation meaningful, the time series with changed labels were excluded from the test set.

TABLE II
NUMBER IQ-MAX'S AND IQ-WV'S WINS/LOOSES AGAINST 1-NN AND k-NN.

	$p = 1$ %	$p = 5$ %	total
IQ-MAX wins against 1-NN	29 (20)	34 (29)	63 (49)
IQ-MAX looses against 1-NN	5 (1)	1 (0)	6 (1)
IQ-MAX wins against k-NN	30 (15)	29 (9)	59 (24)
IQ-MAX looses against k-NN	5 (1)	5 (1)	10 (2)
IQ-WV wins against 1-NN	31 (21)	35 (31)	66 (52)
IQ-WV looses against 1-NN	3 (1)	0	3 (1)
IQ-WV wins against k-NN	31 (15)	34 (22)	65 (37)
IQ-WV looses against k-NN	4 (1)	0	4 (1)

Table II summarizes these results by reporting the number of cases, per noise level and in total, when IQ-MAX and IQ-VW wins/looses against 1-NN and k-NN (in parenthesis we report in how many cases wins/looses are statistically significant).

As shown, in the vast majority of the cases both IQ-MAX and IQ-WV outperform the competitors, often significantly; whereas when they loose, the difference is usually non-significant. Note that in several cases, the errors of our IQ

TABLE I
CLASSIFICATION ERROR.

Dataset	size	p =1%				p =5%			
		IQ-MAX	IQ-WV	1-NN	k-NN	IQ-MAX	IQ-WV	1-NN	k-NN
50 Words	905	**0.239** -/-	**0.241** -/-	0.249	0.242	0.270	**0.254** +/-	0.388	0.260
Adiac	781	**0.373** -/-	**0.377** -/-	0.381	0.384	**0.415** +/-	**0.411** +/+	0.508	0.451
Car	120	0.279	**0.270** -/-	0.278	0.303	**0.310** +/-	**0.283** +/+	0.416	0.353
CBF	930	**0.004** +/+	**0.001** +/+	0.106	0.047	**0.043** +/-	**0.034** +/+	0.328	0.057
ChlorineConcentration	4307	0.053	0.058	**0.021** +/+	**0.021** +/+	0.077	**0.073** -/-	0.075	0.075
CinC	1420	**0.003** +/+	**0.001** +/+	0.033	0.011	**0.008** +/-	**0.004** +/+	0.143	0.021
DiatomSizeReduction	322	**0.006** +/+	**0.006** +/+	0.031	0.038	**0.010** +/+	**0.010** +/+	0.141	0.049
ECG200	200	**0.136** -/-	**0.126** -/-	0.171	0.156	0.150	**0.124** +/-	0.313	0.134
ECGFiveDays	884	**0.013** +/+	**0.010** +/+	0.041	0.045	**0.020** +/+	**0.017** +/+	0.164	0.136
FaceFour	112	**0.063**	**0.072**	0.108	0.072	**0.075**	**0.075**	0.234	0.112
FacesUCR	2250	**0.029** +/+	**0.026** +/+	0.059	0.039	**0.044** +/-	**0.033** +/+	0.193	0.046
Fish	350	**0.228** -/-	**0.219** +/-	0.254	0.239	**0.244** +/-	**0.244** +/-	0.386	0.280
GunPoint	200	**0.010** -/+	**0.010** -/+	0.036	0.061	**0.016** +/+	**0.011** +/+	0.162	0.176
Haptics	463	**0.490** +/-	**0.482** +/-	0.582	0.532	**0.540** +/-	**0.533** +/-	0.681	0.553
InlineSkate	650	0.469	**0.442** -/+	0.461	0.483	**0.523** -/-	**0.504** +/+	0.562	0.570
ItalyPowerDemand	1096	**0.038** +/+	**0.034** +/+	0.087	0.081	**0.059** +/-	**0.047** +/-	0.237	0.060
Lighting2	121	0.192	0.142	0.133	**0.125**	0.209	**0.157**	0.270	0.209
Lighting7	143	0.254	**0.211**	0.254	0.289	**0.279**	**0.243**	0.426	0.338
Mallat	2400	**0.014** +/-	**0.012** +/-	0.055	0.018	**0.019** +/+	**0.017** +/+	0.178	0.034
MedicalImages	1141	**0.212** -/-	**0.203** -/-	0.228	0.234	**0.228** +/-	**0.211** +/+	0.339	0.256
Motes	1272	**0.059** +/+	**0.055** +/+	0.090	0.078	**0.073** +/+	**0.065** +/+	0.206	0.107
OSULeaf	442	0.320	0.301	**0.287** -/-	0.292	0.363	**0.308** +/+	0.402	0.345
Plane	210	**0.005** +/+	**0.005** +/+	0.034	0.038	**0.020** +/+	**0.010** +/+	0.148	0.114
SonyAIBORobotSurface	621	**0.026** +/+	**0.031** +/-	0.073	0.068	**0.035** +/+	**0.034** +/+	0.234	0.083
SonyAIBORobotSurfaceII	980	**0.034** +/+	**0.032** +/+	0.063	0.067	**0.037** +/-	**0.034** +/+	0.212	0.119
StarLightCurves	9236	0.076	**0.071** +/-	0.119	0.073	**0.096** +/-	**0.089** +/+	0.253	0.098
Symbols	1020	**0.023** +/-	**0.024** +/-	0.061	0.031	**0.029** +/-	**0.031** +/-	0.196	0.036
SyntheticControl	600	0.020	0.018	0.076	**0.017** -/-	**0.028** +/-	**0.035** +/-	0.227	0.058
SwedishLeaf	1125	**0.170** +/+	**0.169** +/+	0.206	0.197	**0.189** +/+	**0.181** +/+	0.328	0.216
Trace	200	**0.005** -/-	**0.005** -/-	0.046	0.036	**0.005** +/-	**0.005** +/-	0.180	0.064
TwoLeadECG	1162	**0.001** +/+	**0.001** +/+	0.041	0.052	**0.005** +/+	**0.002** +/+	0.175	0.025
TwoPatterns	5000	**0.001** +/+	**0.001** +/+	0.065	0.007	**0.014** +/-	**0.012** +/+	0.236	0.019
Wafer	7164	**0.003** +/-	**0.003** +/-	0.042	0.004	0.006	**0.005** +/-	0.160	**0.005** +/-
WordSynonyms	905	**0.224** -/-	**0.220** -/-	0.238	0.241	**0.270** +/-	**0.257** +/+	0.379	0.287
Yoga	3300	**0.071** +/+	**0.072** +/+	0.099	0.114	**0.085** +/-	**0.080** +/+	0.223	0.115

estimation based methods are an order of magnitude lower than the error of 1-NN and k-NN: see e.g. TwoLeadECG at $p = 1\%$ (for both IQ-MAX and IQ-VW) and at $p = 5\%$ (for IQ-WV), furthermore GunPoint and Trace at $p = 5\%$ (for both IQ-MAX and IQ-VW) in Tab. I. As expected, IQ-WV, which is more sophisticated than IQ-MAX, generally performs better than IQ-MAX, e.g. at $p = 5\%$ noise IQ-VW is superior to IQ-MAX on 29 datasets, whereas IQ-MAX is better than IQ-VW in only 2 cases.

C. Execution Time

Even for the large data sets, we observed the execution times of our methods to be reasonable, e.g. for IQ-MAX we measured 12.9, 19.8 and 6.8 minutes off-line (training) times (on a Xeon 2.3 GHz processor) for the data sets Wafer, TwoPatterns and ChlorineConcentration respectively. Note that this off-line time refers to the time required for training the regression models at the meta level, which has to be performed only *once*. The same off-line time was necessary for k-NN to find the globally optimal k. This is because training is dominated by the classification of the hold-out set D_2 in both cases. For IQ-MAX the on-line time required to classify a new time series was 0.22, 0.51 and 0.23 seconds (for the above mentioned data sets). For IQ-WV we measured similar execution times which justify our expectations based on the discussion in Section IV-A. Therefore it is evident that our approaches are able to maintain fast classification of new time series.

VI. CONCLUSION

We examined the problem of time-series classification based on the k-NN classifier and the DTW distance. Although the 1-NN classifier had been shown to be competitive, if not superior, to many state-of-the art time-series classification methods, we argued that in several cases we may not only consider $k > 1$ for the k-NN classifier, but also estimate quality of various k-NN classifiers in an individual base for each time series that has to be classified.

We proposed an IQ estimation mechanism that considers a range of k-NN classifiers (for different k values) and uses meta-level regression models that estimate the quality of each such classifier. In the framework of IQ estimation, we proposed two approaches.

Our first proposed approach, IQ-MAX, selects separately for each time series the classifier with the maximum estimated quality. Our second approach, IQ-WV combines the results of the primary-level classifiers according to the weighted voting schema, for which we used the estimated qualities as weights.

Both approaches allow for adapting to characteristics that are varying among the different regions in a data set and overcoming the problem of selecting a single k value.

Our experimental evaluation used a large collection of real data sets. Our results indicate that the proposed methods are more robust and compare favorably against two examined baselines by resulting in significant reduction in the classification error. Other advantageous properties of the proposed methods are their small sensitivity against the parameters it uses and the small overhead it adds in execution time.

It is important to state that the proposed IQ estimation mechanism has several generic features. For the k-NN classifier, IQ estimation can be employed for learning other parameters than k, such as the distance measure or the importance of nearest neighbors. More importantly, IQ estimation is not limited for the problem of k-NN classification of time-series data, since it can be used in combination with other classification algorithms and data types, whenever the complexity of the data requires such an individualized approach. Therefore, our future work involves the examination of IQ estimation in a more general context of classification problems.

REFERENCES

[1] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[2] C. Gruber, M. Coduro, and B. Sick, "Signature verification with dynamic RBF networks and time series motifs," 2006.

[3] S. Marcel and J. Millan, "Person authentication using brainwaves (EEG) and maximum a posteriori model adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 743–752, 2007.

[4] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[5] T. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2003.

[6] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," in *Proceedings of the VLDB Endowment*, vol. 1, no. 2, 2008, pp. 1542–1552.

[7] E. Keogh, C. Shelton, and F. Moerchen, "Workshop and challenge on time series classification," 2007. [Online]. Available: http://www.cs.ucr.edu/~eamonn/SIGKDD2007TimeSeries.html

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Verlag, 2009.

[9] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Time-Series Classification in Many Intrinsic Dimensions," in *SIAM International Conference on Data Mining*, 2010, pp. 677–688.

[10] A. Kehagias and V. Petridis, "Predictive modular neural networks for time series classification," *Neural Networks*, vol. 10, no. 1, 1997.

[11] P. Sykacek and S. Roberts, "Bayesian time series classification," *Advances in Neural Information Processing Systems*, vol. 2, pp. 937–944, 2002.

[12] D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, R. Porter, and J. Theiler, "Genetic algorithms and support vector machines for time series classification," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 4787, 2002, pp. 74–85.

[13] K. Buza and L. Schmidt-Thieme, "Motif-based classification of time series with Bayesian networks and SVMs," *Advances in Data Analysis, Data Handling and Business Intelligence*, pp. 105–114, 2010.

[14] P. Geurts, "Pattern extraction for time series classification," in *Principles of Data Mining and Knowledge Discovery (PKDD)*. Springer, 2001, pp. 115–127.

[15] C. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *SIAM International Conference on Data Mining*, 2004, pp. 11–22.

[16] E. Keogh and M. Pazzani, "Scaling up dynamic time warping for datamining applications," in *SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2000, pp. 285–289.

[17] C. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *SIGKDD International Workshop on Mining Temporal and Sequential Data*, 2004.

[18] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, 1996.

[19] C. Domeniconi and D. Gunopulos, "Adaptive nearest neighbor classification using support vector machines," *Neural Information Processing Systems (NIPS)*, 2001.

[20] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally adaptive metric nearest-neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[21] S. Ougiaroglou, A. Nanopoulos, A. Papadopoulos, Y. Manolopoulos, and T. Welzer-Druzovec, "Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors," in *Advances in Databases and Information Systems*. Springer, 2007, pp. 66–82.

[22] A. Molinaro, R. Simon, and R. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, no. 15, p. 3301, 2005.

[23] A. Jain, R. Dubes, and C. Chen, "Bootstrap techniques for error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 628–633, 2009.

[24] N. Duffy and D. Helmbold, "Boosting methods for regression," *Machine Learning*, vol. 47, no. 2, pp. 153–200, 2002.

[25] K. Tuda, G. Rätsch, S. Mika, and K. Müller, "Learning to predict the leave-one-out error of kernel based classifiers." LNCS 2130, Springer Verlag, 2001, pp. 331–338.

[26] D. Wettschereck and T. Dietterich, "Locally adaptive nearest neighbor algorithms," *Advances in Neural Information Processing Systems*, pp. 184–184, 1994.

[27] D. Gunopulos and G. Das, "Time series similarity measures and time series indexing," in *Proc. ACM SIGMOD International Conference on Management of Data*. ACM, 2001, p. 624.

[28] E. Keogh and C. Ratanamahatana, "Exact indexing of Dynamic Time Warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.