Ensembles of Nearest Neighbor Forecasts

Dragomir Yankov[1], Dennis DeCoste[2], and Eamonn Keogh[1]

[1] University of California, Riverside CA 92507, USA,
{dyankov,eamonn}@cs.ucr.edu,
[2] Yahoo! Research, 3333 Empire Blvd., Burbank CA, USA,
decosted@yahoo-inc.com

Abstract. Nearest neighbor forecasting models are attractive with their simplicity and the ability to predict complex nonlinear behavior. They rely on the assumption that observations similar to the target one are also likely to have similar outcomes. A common practice in nearest neighbor model selection is to compute the globally optimal number of neighbors on a validation set, which is later applied for all incoming queries. For certain queries, however, this number may be suboptimal and forecasts that deviate a lot from the true realization could be produced.
To address the problem we propose an alternative approach of training ensembles of nearest neighbor predictors that determine the best number of neighbors for individual queries. We demonstrate that the forecasts of the ensembles improve significantly on the globally optimal single predictors.

1 Introduction

K-nearest neighbor (k-NN) methods for forecasting work by first identifying the k most similar time series to a given query and then, by combining their historical continuations, evaluate the expected outcome for the query. The methods are linear with respect to the model parameters, and yet they turn out to be suitable for predicting highly nonlinear fluctuations too. This is due to the fact that the identified neighbors themselves could comprise complex nonlinear patterns.

One significant drawback of the k-NN forecasts is their sensitivity to changes in the input parameters, e.g. the number of nearest neighbors, the weighting function, the prediction horizon or the length of the query vector. The impact of the number of neighbors is especially interesting as the resulting models may have intrinsically different characteristics. Namely, a forecast combining too many neighbors, quite often turns out to be biased and one that uses just a few of them, to have large variance. The effect is known as the *bias-variance dilemma* [5] and has been observed before in the context of k-NN forecasting [3, 10]. Yet, no consistent approach has been suggested that could improve the forecasts of the method.

Here we propose a procedure, which rather than searching for the globally optimal k-NN predictor, constructs an ensemble of two predictors {k_1-NN, k_2-NN}, using a different number of neighbors k_1 and k_2 respectively. For each

individual query the procedure selects one of the predictors from the ensemble to perform the forecast. Suppose that we have an oracle that looks at the actual continuation of every query and lets the method correctly pick the better of the two predictors. Studying the performance of such 'perfect' ensembles we observed the following effect. The ensembles which perform best, tend to have distant values for k_1 and k_2, with k_1 usually being very small (one or two nearest neighbors). What is also interesting, is that most of the ensembles, even those that are composed of suboptimal predictors, have better accuracy and stability than the globally optimal single predictor.

The observation suggests that it is often the case when we can split the queries into two distinct classes, one that requires a predictor using a small number of neighbors and another one that is better predicted with large number of neighbors. We can look at the globally optimal k-NN predictor as a safe compromise for the two classes, such that does not perform too poorly on each of them, but in general is not optimal for them either. Learning to separate those two classes could give us a powerful tool for improving on the k-NN models and in this work we demonstrate how this can be achieved. The potential of the approach to reduce both the bias and the variance of the NN forecasts is also illustrated.

The rest of the paper is organized as follows. In Section 2 we make an overview of the NN forecasting literature. Section 3 defines the k-NN forecasting framework. Section 4 describes the proposed method. An evaluation of the performance of our approach as compared to the optimal single k-NN predictor is presented in Section 5.

2 Related Work

Nearest neighbor forecasting methods have become popular with the advancement in dynamic systems. The relevance of the methods for time series, arising from such systems, is established by Taken's 'delay coordinate embedding theorem' [12]. It states that if there are enough observations, one can reconstruct the manifold representing the state space of the system. If a query is produced by the same system, then it should be part of the manifold too and its outcome will lie close to the outcome of its nearest neighbors.

The effectiveness of k-NN methods becomes apparent during the Santa Fe forecasting competition [2], when they are demonstrated to be competitive to other more complex methods as feedforward networks. Two entries from the competition that use a k-NN model, submitted by Sauer [10] and Casdagli et al. [3], show very good performance with Sauer taking second place on the Laser generated data set (see Section 5.1). Both Sauer and Casdagli et al. discuss the bias-variance problem of the forecasts but no principled approach for its solution has been suggested.

3 Formalization

Let a time series $\mathbf{y}(t) = (x_1, x_2, ..., x_t)$ be defined as a sequence of scalar observations measured at equal intervals in time. In its general form the forecasting problem targets the estimation of h consecutive future values, i.e. $\mathbf{y}_t(h) = (x_{t+1}, x_{t+2}, ..., x_{t+h})$, using any of the currently available observations from $\mathbf{y}$ (and possibly other time series). Here h is the user specified *prediction horizon*.

The available time series are organized in a training set by running a sliding window of size l along each of them. I.e. the set contains elements of the form $\mathbf{y}_t(l) = (x_{t+1}, x_{t+2}, ..., x_{t+l})$, called *lag* vectors. Note, that if the initial observations are long enough, for most elements $\mathbf{y}_t(l)$ the continuation $\mathbf{y}_{t+l}(h)$ will also be available in the training set. An estimate for the continuation of a query vector $\mathbf{q}(l)$ is then computed by the k-NN predictor as the linear combination:

$$\hat{\mathbf{q}}_c(h) = w_1 \mathbf{y}^1_{c_1}(h) + w_2 \mathbf{y}^2_{c_2}(h) + ... + w_k \mathbf{y}^k_{c_k}(h) \tag{1}$$

where $\mathbf{y}^i_{c_i}(h)$ is the continuation of the i-th nearest neighbor starting at time point c_i, and $\mathbf{w}(q) = (w_1, w_2, ..., w_k)$ is a preselected weighting function (see Section 3.3).

Equation (1) gives the *direct* forecast of the k-NN algorithm for h steps ahead. A different type of prediction is the iterative one, where a single point is predicted at a time and is afterwards appended to the query vector for subsequent predictions. Iterative predictions are more accurate for short horizons, but as the prediction error accumulates faster, for long horizons, direct predictions tend to outperform them [8]. All results presented here are for direct forecasts, yet the proposed ensemble scheme could as well be applied for the iterative predictions.

3.1 Similarity Metric

The majority of the works on NN forecasting utilize the Euclidean distance [4, 10], or some of its modifications. Two popular such modifications are the standardized Euclidean distance in which the series are transformed to have a mean zero and variance one [3], or the weighted Euclidean distance [9] which assigns lower weights to coordinates in the lag vector that are further in time from the target value. The weighted Euclidean distance complicates the model by adding another parameter to it, and it also makes the forecast more sensitive to the sizes of the lag vectors. The standardized Euclidean distance, on the other hand, is not very robust to noise and to non-stationary first and second moments.

In the current implementation the data are also standardized, but then the resulting vectors are compared with the scale-shift invariant distance metric, discussed by Goldin et al. [6]. If $\mathbf{q}$ is the query, $\mathbf{y}$ is the lag vector and $\mathbf{q}_s$ and $\mathbf{y}_s$ are their standardized representations, then the scale-shift transformation further changes $\mathbf{y}_s$ as: $\tilde{\mathbf{y}} = a\mathbf{y}_s + b$. The distance $d(\mathbf{q}, \mathbf{y})$ is now measured as the Euclidean distance (L_2) between the standardized query and the transformed neighbor, i.e. $d(\mathbf{q}, \mathbf{y}) = L_2(\mathbf{q}_s, \tilde{\mathbf{y}})$. The coefficients a and b are estimated using least square linear fit between $\mathbf{q}_s$ and $\mathbf{y}_s$.

3.2 Estimating the Prediction Accuracy

To evaluate the proposed procedure we measure the root mean square error (RMSE) between the actual outcome and the prediction, normalized respectively by the standard deviations of the query and the linear combination of its nearest neighbors: $RMSE = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(\hat{q}_{t+i} - q_{t+i})^2}$. The residual $(\hat{q}_{t+i} - q_{t+i})$ stands for the difference between the scalar prediction and the true outcome for time point $(t + i)$. The RMSE has been the preferred error function for comparing forecasting accuracy in a number of time series prediction competitions. It is symmetric, i.e. both over or underestimating the true value are penalized equally, and measures the loss in the same units as the recorded variable. As the outcome and the forecast are normalized, for stationary time series the RMSE of the simple mean value predictor is equal to one. This further provides a baseline for comparing the goodness of a forecasting algorithm when evaluated on data sets as the Laser oscillations in Section 5.1.

3.3 Weighting Functions

There are two conceptually different weighting schemes, that a NN model can utilize, *kernel regression* and *locally weighted regression* (LWR) [1].

The *kernel* is a function of the distance between the query and its neighbors. One popular kernel is the *uniform* one, assigning equal weights to all neighbors. Atkeson et al. [1] argue that there is no clear evidence for the benefit of using a particular kernel function with NN-learning in general. While our experiments confirmed this observation, we also found out that the uniform kernel behaves more coherently across different data sets and when the input parameters are varied.

Rather than computing a distance function, LWR finds the linear combination of the neighbors that approximates most closely, in a least square sense, the query. The heuristic assumption is then made that the same linear combination of the neighboring continuations will also be the one that approximates best the query outcome. This heuristic holds for short horizons, but as the horizon increases, the assumption gets more unrealistic and the prediction performance becomes considerably poor.

For better consistency across different data sets or input parameters, the presented model utilizes the uniform kernel function.

4 Ensembles of NN Predictors

A general NN model selection procedure computes the globally best number of neighbors on a validation set, and then uses this number for forecasting all subsequent queries. There could be individual queries though, for which the forecasts are way off the true outcome and a different number of neighbors might be more suitable for them. Here we describe a procedure that improves

on the error accuracy of the single NN predictors, by adapting to the individual queries.

Suppose that, rather than using a single k-NN predictor for the forecasts, we form the ensemble of two such predictors $Ens = \{k_1\text{-NN}, k_2\text{-NN}\}$. The ensemble works as follows. For every query $\mathbf{q}$, it selects this one of the *subpredictors* k_1-NN or k_2-NN that has a better forecasting accuracy on $\mathbf{q}$. For the time being let us neglect the issue of how exactly that selection is made and assume that Ens always makes the perfect choice. Table 1 lists some results for such ensemble predictors on the Impressions data set (see Section 5.2). It also compares them with the values for several single k-NN predictors[3].

Table 1: Validation error of several ensembles and several simple predictors.

k	RMSE(k-NN)	(k_1, k_2)	RMSE(Ens)
1	2.0447	(1,20)	1.5829
2	1.9504	(2,40)	1.5996
6	1.8321	(6,1)	1.6305
100	2.9608	(100,1)	1.6095

The single predictor with the smallest validation error for this data set is k^*-NN = 6-NN. The pairs in column 3 are formed by fixing k_1 and selecting k_2 which minimizes the error of the resulting predictor. Among all such pairs the globally optimal one for the validation set is $Ens^* = \{1\text{-NN}, 20\text{-NN}\}$.

There are two important aspects in the above result. Firstly, note that the subpredictors using a relatively small number of neighbors form optimal ensembles with subpredictors that use a large number of neighbors and vice versa. And secondly, the ensembles, though composed of suboptimal predictors, perform with 10% to 15% better than the globally optimal single predictor 6-NN. To provide some insight on those two effects consider the histograms on Figure 1.

On the left histogram, the difference of the error RMSE(6-NN) - RMSE(1-NN) is spread evenly with much volume on both sides of the median. This means that for almost half of the queries 6-NN is not optimal and their forecasts could be improved. The right histogram shows the difference RMSE(6-NN) - RMSE(10-NN). As the forecasts of 6-NN are similar to these of 10-NN, it cannot clearly identify all those queries for which 6-NN is not optimal. In general, the flatter the histogram, and the more volume it has on both sides of the median, the more improvement will be introduced by the ensemble.

It turns out that it is also easier to find a classifier that might separate well the types of queries inferred by distant predictors, rather than by similar ones. Furthermore, as k-NN has in general smaller bias when k is small and smaller variance when k is large, this suggests that grouping distant predictors might result in ensembles that have the potential for improving both the bias and

[3] We report validation errors here to avoid test selection biases arising during our experimentation. All later experimental work reports true test error performance.

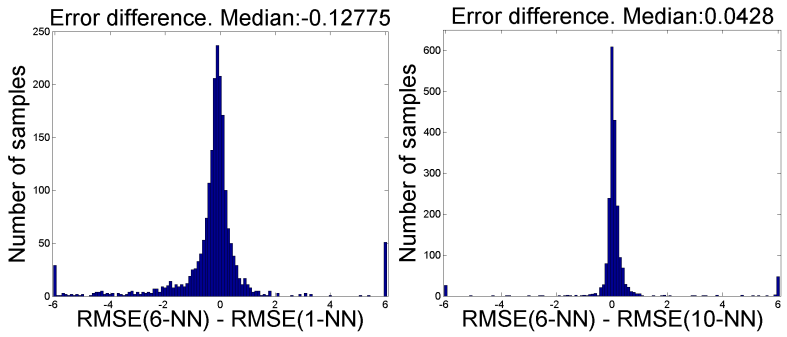

Fig. 1: Distant values have flatter histograms suggesting possible improvement.

the variance of the single predictors. In Section 5.3 we provide some empirical evidence to support this conjecture.

4.1 Learning Classes of Queries

So far we have demonstrated that an ensemble of a biased and unbiased NN predictors outperforms the globally optimal single predictor, provided that for every query we correctly select the more accurate of the two subpredictors. Now we demonstrate that the two classes, inferred by those subpredictors, are often largely separable and could be successfully learned by a classifier that will assign them to the right predictor.

Feature Selection Two type of features have been defined and used in the classifiers that we train in this work: statistical and performance related. The former include some statistical properties of the query and the identified neighbors. The second type deal with the performance of the subpredictors on part of the query or on its nearest neighbors.

Variance of the query, its nearest neighbors and the two forecasts. We measure the variance of all input vectors and that of the two forecasts. When the time series are non stationary (or contain a lot of noise), as $k_1 < k_2$, the forecast of k_2-NN due to the effect of aggregation usually varies less and yields smaller prediction error. On the contrary, when the time series are stationary or have very close neighbors in the training set, then increasing the number of neighbors also increases the chance of identifying an outlier. In those cases the forecasts of k_1-NN have smaller variance and are often closer to the true realization. The feature turns out to be a very strong indicator for the better subpredictor in some of the tested data sets.

Distances between the individual forecasts. When the individual forecasts of the first k_2 neighbors are very similar, then it is reasonable to have higher confidence in their combined forecast. Using k_2-NN will give us smoother and supposedly better forecast.

Performance on the nearest neighbors. We measure the accuracy of k_1-NN and k_2-NN on some of the nearest neighbors to the query. If the space is dense, then the better predictor for the neighbors will likely be the better predictor for the query too.

Step-back forecasts. Looking only at the beginning of the query we test which subpredictor forecasts its ending more accurately. The feature is a strong indicator for short horizon forecasts and for *self-similar* time series.

Classification We look for a classifier to differentiate between the queries that are better predicted by any of the two subpredictors in the ensemble. Ideally, it should allow to be flexibly tuned between underfitting, when all samples are classified with the safe majority label, and overfitting the data. For the purpose, we use a Support Vector Machine (SVM) with a Gaussian kernel [11].

If $\mathbf{u}_i$ are the vectors of features (see Section 4.1) corresponding to the queries, and $\mathbf{v}_i$ are the respective labels (+1 for the dominant class, and -1 for the other one), then SVM classifies a test sample $\mathbf{u}$ according to the rule:

$$sgn(\mathbf{u}) = sgn \sum_{i=1}^{n} (\alpha_i \mathbf{v}_i K(\mathbf{u}_i, \mathbf{u}) + b) \tag{2}$$

where $0 \leq \alpha_i \leq C, i = 1..n$. In equation (2) α_i are the solution of the dual SVM optimization problem, b is a threshold also learned in the optimization, C is a parameter which determines the trade-off between the complexity and the training error of the classifier and needs to be specified prior to the optimization, and $K(\mathbf{u}_i, \mathbf{u})$ is a kernel function computing the distance between the test sample and a training sample in a highly dimensional feature space. The Gaussian kernel with width σ is defined as $K(\mathbf{u}_i, \mathbf{u}) = \mathbf{exp}[-\|\mathbf{u}_i - \mathbf{u}\|^2/(2\sigma^2)]$, where σ also has to be specified in advance.

Using SVM with this type of kernel we can easily obtain the safe asymptotic classifier that underfits the data, for example by letting $C \to 0$ (see [7]). Then, by tuning the two parameters C and σ using cross-validation, a more optimal classification can be found. However, the procedure might become very computationally intensive as a quadratic search has to be performed within a very large set of values.

To find the best (C, σ) pair, we apply the heuristic described by Keerthi et al. [7]. They show that the solution of the equation $\log \sigma^2 = \log C - \log \tilde{C}$, where $\tilde{C}$ is the trade-off parameter for a linear SVM, provides a good approximation of the optimal C and σ. As both checking $\tilde{C}$ and the corresponding solutions of the equation require linear number of steps, the overall time for finding the approximation is linear.

An important aspect concerning the classification of the queries is that we do not need a classifier with perfect accuracy. It can make a lot of errors around the median of the histogram (Figure 1) but as long as the more distant queries are classified correctly the ensemble will still outperform the optimal simple predictor k^*-NN.

5 Empirical Results

The performance of the ensemble predictors is studied on two data sets - the laser oscillation data from the Santa Fe competition and real world data collected from web logs. Three horizons are considered: a relatively short one (30 steps ahead), a medium range one (60) and a long range horizon (100). The query size used is 30 time points.

Unless otherwise specified the best single predictor is compared with the ensemble {1-NN, 10-NN}. It tends to be a good choice in most cases as its sub-predictors have distant values for the number of neighbors used, but in general is not globally optimal. When the improvement that we obtain with this ensemble is not significant enough, we also look at the performance of the optimal ensemble as inferred from a validation set. The result of the better of the two is displayed.

5.1 Laser Oscillation Data

The data represents the oscillation of a laser that can be modeled with a Lorenz system. It appears as *DataSet A* in the Santa Fe forecasting competition. The oscillations are comparatively easy to predict, but the transitions between the larger segments occur randomly. The available data are randomly split into a training (6000 elements), validation (2000) and test(2000) sets.

The best single predictor for all three horizons is the 3-NN predictor and the optimal ensemble, again for the three horizons, is {1-NN, 4-NN}. The hypothetically possible improvement in prediction error, by using the best ensemble with an oracle, over the best single predictor is: 16% (horizon 30), 17% (60), 16% (100).

For the single predictors, when increasing k beyond 3 the forecasts steadily begin to worsen. The good performance of the predictors with small number of neighbors is a result of the stationarity and the cleanness of the data. Two

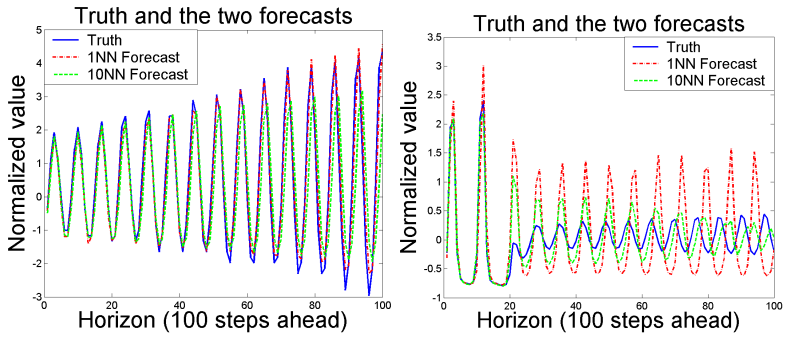

Fig. 2: *Left:* Example where 1-NN works better. *Right:* Example where more nearest neighbors perform better. *(Laser Oscillation Data)*

examples to illustrate the strengths and weaknesses of predictors with small and large number of neighbors are given on Figure 2.

When none of the neighbors predict any transition in the oscillation, using just one neighbor is usually preferable. Adding more neighbors increases the chance of selecting an outlier (Figure 2, *Left*). On the other hand, if many neighbors predict a transition, then increasing k makes it more likely to detect the actual amplitude of the oscillation after the transition (Figure 2, *Right*).

The test set accuracy of the SVM classifier separating the samples into groups, better predicted by k_1-NN or k_2-NN is summarized in Table 2 (column 2). For this data set the samples are equally distributed among the two groups. We found out that this is another premise for a better performance of the ensemble method. The table also lists the prediction test error and its standard deviation for the three horizons. The ensemble improves on both of the components over the optimal 3-NN predictor.

Table 2: Test error and classification accuracy. *(Laser Oscillation Data)*

Horizon	Class. Accuracy	Predictor	Test RMSE	Std
$h = 30$	0.75	3-NN (optimal k)	0.124	0.132
		Ens ={1-NN,10-NN}	**0.120**	**0.130**
$h = 60$	0.74	3-NN (optimal k)	0.207	0.170
		Ens^* ={1-NN,4-NN}	**0.189**	**0.162**
$h = 100$	0.81	3-NN (optimal k)	0.355	0.226
		Ens ={1-NN,10-NN}	**0.329**	**0.213**

For horizon 60 the improvement that we obtained with {1-NN, 10-NN} was not significant. We also check the performance of the globally optimal ensemble according to the validation set, in this case Ens^* ={1-NN, 4-NN}.

Finally, Figure 4 *Left* summarizes the percentage improvement of the ensemble forecast compared to the best single predictor. The improvement in test error is between 25% and 50% of the hypothetically possible improvement measured earlier on the validation set.

5.2 Web Site Impressions Data

The time series represent the number of *impressions* (users that have seen the banner ads), for a set of web sites, recorded over a period of three years.

The series have weekly recurrences, often with seasonal trends, a lot of noise and are highly nonstationary. The training set contains approximately 50 000 vectors, the validation and the test set have 2000 samples each. As a very small portion of the samples increase the error with orders of magnitude, to be fair to the representative majority of the samples we look at the 95%-quantile of the error and its deviation.

The optimal single predictors computed on the validation set are: 10-NN (horizon 30), 8-NN (60), 6-NN (100). The optimal ensembles are: {3-NN, 100-NN} (30), {1-NN, 30-NN} (60) and {1-NN, 20-NN} (100). The hypothetical margin for improvement if using an oracle is: 14% (30), 13% (60), 14% (100).

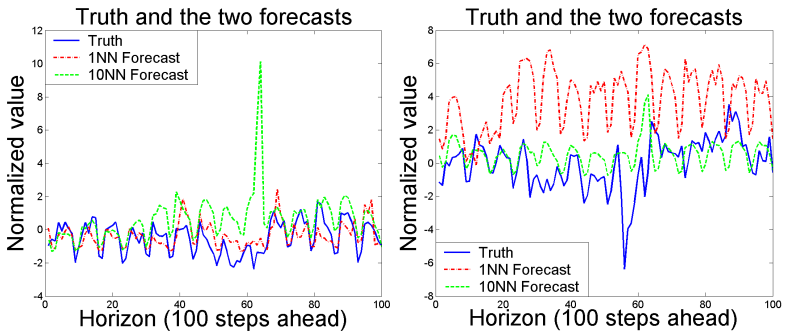

Fig. 3: *Left:* Example where 1-NN works better. *Right:* Example where more nearest neighbors perform better. The smoother forecast is usually the more accurate one. *(Web Impressions Data)*

Due to the large amount of noise, the more conservative, i.e. the smoother forecasts, are usually the more accurate ones (Figure 3). Therefore, the statistical features discriminate quite well between the two classes for the extreme examples, i.e. the examples for which applying the wrong subpredictor increases the error significantly. On the other hand the random spikes and drops in the data are hard to forecast and for these samples the assigned subpredictor is often incorrect. Because of the above two effects, the accuracy of the SVM classifier is comparatively low, but the overall improvement introduced by the ensemble method is quite good (see Table 3). The improvement in the test error and its de-

Table 3: Test error and classification accuracy. *(Web Impressions Data)*

Horizon	Class. Accuracy	Predictor	Test RMSE	Std
$h = 30$	0.58	10-NN (optimal k)	1.1235	0.644
		Ens ={1-NN,10-NN}	**1.021**	**0.452**
$h = 60$	0.77	8-NN (optimal k)	1.549	0.862
		Ens ={1-NN,10-NN}	**1.412**	**0.685**
$h = 100$	0.58	6-NN (optimal k)	1.8676	1.183
		Ens ={1-NN,10-NN}	**1.6881**	**0.961**

viation (Figure 4 *Right*) is between 60% and 70% of the hypothetically possible improvement computed on the validation set.

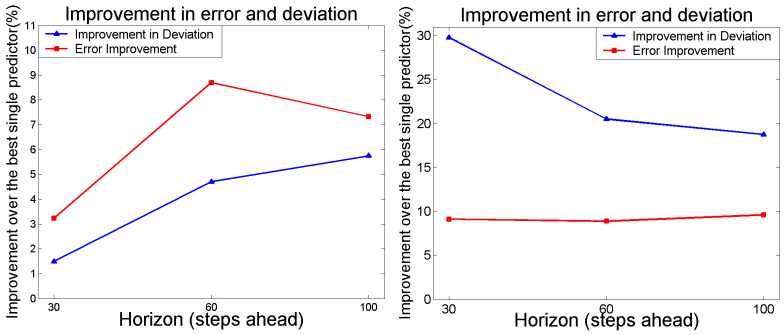

Fig. 4: Test error improvement of the ensemble approach over the best single predictor. *Left*: Laser Oscillation Data. *Right*: Web Impressions Data

5.3 Bias-Variance Improvement

The squared loss of a predictor decomposes into the following two components [5]:

$$\mathcal{E}_{\mathcal{D}}[\{\hat{\mathbf{q}} - \mathbf{q}\}^2] = \underbrace{\{\mathcal{E}_{\mathcal{D}}[\hat{\mathbf{q}}] - \mathbf{q}\}^2}_{bias^2} + \underbrace{\mathcal{E}_{\mathcal{D}}[\{\hat{\mathbf{q}} - \mathcal{E}_{\mathcal{D}}[\hat{\mathbf{q}}]\}^2]}_{variance} \tag{3}$$

where the expectations are computed over a number of different training sets $\mathcal{D}$.

In the previous experiments it was demonstrated that the ensembles can decrease the test error, and hence the overall loss of the single predictors. It is essential to understand whether that improvement originates from one or both of the components in equation 3.

From the larger of the data sets, the Impressions data, we draw 50 random replicas, of size 90% of the original training set size. For every query in the test set, the bias and the variance over the replicas $\mathcal{D}$ are computed. The average bias and variance over all queries, for horizon 100, is presented in Table 4.

Table 4: Bias and variance for horizon 100 on the Impressions data set. The ensemble improves on both of the components

Predictor	Bias^2	Variance
1-NN	5.468	1.174
6-NN (optimal k)	5.042	0.638
10-NN	5.690	1.96
Ens ={1-NN,10-NN}	**3.721**	**0.204**

As seen from the table, the ensembles can decrease both terms in the squared loss decomposition, which suggests that they are a potentially powerful approach towards the bias-variance problem of the k-NN forecasts.

6 Conclusion

We have presented a method for learning how to separate time series queries into two classes, that are better predicted with one of two possible NN predictors. The experimental evaluation shows that such ensembles have better prediction error, compared to the single globally optimal k-NN predictor.

The work raises some interesting questions. For example, what kind of improvement would one expect, if the ensembles include more than two subpredictors. The results indicate that essential for the performance is the identification of bad cases for the individual predictors. Including more subpredictors adds more alternatives to select from, when forecasting these bad samples. On the other hand, the multiway classification might have lower accuracy. Another interesting research direction is how to combine models that differ with respect to other input parameters, such as weighting function, query lengths, or prediction horizon. In this case a criterion for what models should be combined and different features, characteristic of the new models, need to be derived.

References

1. C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 1996.
2. A.Weigend and N. Gershenfeld. *Time Series Prediction. Forecasting the Future and Understanding the Past*. Addison-Wesley Publishing Company, 1994.
3. M. Casdagli and A. Weigend. Exploring the continuum between deterministic and stochastic modeling. *Time Series Prediction. Forecasting the Future and Understanding the Past*, 59(8):347–366, August 1994.
4. J. Farmer and J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59(8):845–848, August 1987.
5. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1 – 58, August 1992.
6. D. Goldin and P. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. *Lecture Notes in Computer Science*, 976(7):137–153, January 1995.
7. S. Keerthi and C. Lin. Asymptotic behaviors of Support Vector Machines with Gaussian kernel. *Neural Computation*, (15):1667–1689, 2003.
8. J. McNames, J. Suykens, and J. Vandewalle. Winning entry of the K.U.Leuven time series prediction competition. *Internation Journal of Bifurcation and Chaos,*, 9(8):1485–1500, August 1999.
9. D. Murray. Forecasting a chaotic time series using an improved metric for embedding space. *Physica D*, 68(8):318–325, August 1993.
10. T. Sauer. Time series prediction by using delay coordinate embedding. *Time Series Prediction. Forecasting the Future and Understanding the Past*, 59(8):175–193, August 1994.
11. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
12. F. Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics, Dynamical Systems and Turbulence*, 898(7):366–381, January 1981.