Input and Structure Selection for *k*-NN Approximator

Antti Sorjamaa, Nima Reyhani, and Amaury Lendasse

Neural Network Research Centre,
Helsinki University of Technology, P.O. Box 5400,
02015 Espoo, Finland
{asorjama, nreyhani, lendasse}@cis.hut.fi

Abstract. This paper presents *k*-NN as an approximator for time series prediction problems. The main advantage of this approximator is its simplicity. Despite the simplicity, *k*-NN can be used to perform input selection for nonlinear models and it also provides accurate approximations. Three model structure selection methods are presented: Leave-one-out, Bootstrap and Bootstrap 632. We will show that both Bootstraps provide a good estimate of the number of neighbors, *k*, where Leave-one-out fails. Results of the methods are presented with the Electric load from Poland data set.

Keywords: *k*-NN, Time Series Prediction, Bootstrap, Leave-one-out and Model Structure Selection.

1 Introduction

In any function approximation, system identification, classification or prediction task one usually wants to find the best possible model and the best possible parameters to have good performance. Selected model must be generalizing enough still preserving accuracy and reliability without unnecessary complexity, which increase computational load and thus calculation time. Optimal parameters must be determined for every model to be able to rank the models according to their performances.

Furthermore, in order to select the best model, or model class, one also needs to determine the best input set, to use in the determination of the best parameters for each structure. When too many inputs are selected, it is possible to get even worse results than with fewer inputs containing more accurate and valid information. Vice versa, with only few inputs the accuracy of the model might not be enough and results are poor and unreliable.

The problems mentioned above occur simultaneously, so it is difficult to find the right combination of correct attributes. It can be a very tedious and time-consuming procedure to go through every possible structure to select the best one.

In this paper, we focus in finding the optimal structure for *k-Nearest Neighbors* (*k*-NN) approximator. At the same time we consider the problem of selecting the most necessary and optimal inputs for the *k*-NN as well as selecting the structure of the approximator. The *k*-NN method is presented in Section 2; Section 3 describes the selection of the inputs with an exhaustive search and Section 4 the selection of the

J. Cabestany, A. Prieto, and D.F. Sandoval (Eds.): IWANN 2005, LNCS 3512, pp. 985 – 992, 2005.

structure using Leave-one-out (LOO) and Bootstraps. In Section 5 we show some experimental results with an electric load time series and then derived conclusions from the results in Section 6.

2 *k*-Nearest Neighbors

k-Nearest Neighbors approximation method is a very simple, but powerful method. It has been used in many different applications and particularly in classification tasks [1].

The key idea behind the *k*-NN is that similar input data vectors have similar output values. One has to look for a certain number of nearest neighbors, according to Euclidean distance [1], and their corresponding output values to get the output approximation. We can calculate the estimation of the outputs by using the average of the outputs of the neighbors in the neighborhood.

If the pairs (x_i, y_i) represent the data with x_i as an n-dimensional input and y_i as a scalar output value, *k*-NN approximation is

$$\hat{y}_i = \frac{\sum_{j=1}^{k} y_{P(j)}}{k}, \tag{1}$$

where $\hat{y}_i$ represents the output estimation, $P(j)$ is the index number of the j^{th} nearest neighbor of the input x_i and k is the number of neighbors that are used.

We use the same neighborhood size for every data point, so we use a global k, which must be determined.

3 Input Selection

In order to select the best set of inputs, all possible 2^n (n is the maximum number of inputs) input sets are built and evaluated. For each input set the global optimum number of neighbors is determined and the generalization error estimate (defined in Section 4) of the set is calculated as a mean of errors of all data points. In this way it is possible to compare all different input sets and take the best one to be used in the final *k*-NN approximation.

In this case, adding one input doubles the needed calculation time. We have to make a compromise between the maximum input size to use and the calculation time available. This kind of exhaustive search for best inputs is usually not preferred, because of the huge computational load. However, with *k*-NN the computations can be performed in a reasonable time, thanks to the simplicity of the *k*-NN.

4 Model Structure Selection

We consider the problem of determining a model which approximates as accurately as possible an unknown function $g(.)$. This approximation is chosen among a set of several possible models. Models in a set are denoted here by

$$h^q(x, \theta(q)) \ , \tag{2}$$

where q represents the q^{th} model in the set, $\theta(q)$ are the parameters of the q^{th} model and x is a n-dimensional input vector. The parameters that define a set of possible models are called hyper-parameters; they are not estimated by the learning algorithm, but by some external procedure [4].

In a typical learning procedure, the $\theta(q)$ parameters are optimized to minimize the approximation error on the learning set; the structure is determined as the minimization of *the generalization error* defined as

$$E_{gen}(q, \theta) = \lim_{M \to \infty} \frac{\sum_{i=1}^{M} \left(h^q(x_i, \theta(q)) - y_i \right)^2}{M} \ , \tag{3}$$

where x_i are n-dimensional input vectors to the model and y_i the corresponding scalar expected outputs.

According to the definition (3), the generalization error is the mean square error of the model, computed on an infinite sized test set. Such set is not available in practice, so we must approximate the generalization error. The best model structure q is the structure that minimizes the approximation of the generalization error.

In our case the model parameters consist in selecting the number of neighbors to use in the *k*-NN approximation. We have used two different methods to select the global optimal number of neighbors, Leave-one-out (LOO) and Bootstrap 632.

Leave-one-out is a common method used in many statistical evaluation purposes and we wanted to show that Bootstrap 632 is better than LOO in this case, even [2] claims that Bootstrap 632 doesn't work in selecting the number of neighbors.

4.1 Leave-One-Out

Leave-one-out [3] is a special case of *k*-fold cross-validation resampling method. In *k*-fold cross-validation the training data is divided into *k* approximately equal sized sets. Then model is trained by using all but one set and the leftover set is used in validation. The generalization error estimation of *k*-fold cross-validation is a mean of all *k* different validation results.

LOO procedure is the same as *k*-fold cross-validation with *k* equal to the size of the training set *N*. So, for each different neighborhood size, LOO procedure is used to calculate its generalization error estimate by removing each neighbor at a time from the training set, building a model with the rest of the training data and calculating the validation error with the one taken out. This procedure is done for every data point in the training set and the estimate of the generalization error is calculated as a mean of all *k*, or *N*, validation errors (4).

$$\hat{E}_{gen}(q) = \frac{\sum_{i=1}^{N} \left(h^q(x_i, \theta_i^*(q)) - y_i \right)^2}{N} \ , \tag{4}$$

where x_i is the i^{th} input vector from the training set, y_i is the corresponding output and $\theta_i^*(q)$ includes the model parameters without using (x_i, y_i) in the training.

Because we want to use the global optimum size of the neighborhood, we have to calculate the LOO error for each data point and each size of the neighborhood. After that we can take the mean over all data points to find out, which is globally the optimum number of neighbors. We select the number of neighbors that gives us the smallest generalization error.

4.2 Bootstrap and Bootstrap 632

Bootstrap [4] is a resampling technique developed to estimate some statistical parameters (like the mean of a population, its variance, etc). In the case of a model structure selection, the parameter to be estimated is the generalization error.

When using bootstrap, the generalization error is not estimated directly. Rather the bootstrap estimates the difference between the generalization error and the training error, or apparent error according to Efron [2]. This difference is called *the optimism*. The estimation of the generalization error will be the sum of the training error and the estimated optimism.

The training error is computed using all available data on the training set.

$$E^{I,I}(q,\theta^*(q)) = \frac{\sum_{i=1}^{N}\left(h^q(x_i^I,\theta^*(q)) - y_i^I\right)^2}{N}, \tag{5}$$

where h^q is the q^{th} model that is used, I denotes the training set, $\theta^*(q)$ includes the model parameters after learning, x_i^I is the i^{th} input vector from the training set, y_i is the corresponding output and N is the number of elements in the training set.

The optimism is estimated using a resampling technique, based on drawing with replacement within the training set. This *bootstrap set* is as large as the training set with its participants drawn randomly from the training set. Each model is trained using the bootstrap set and optimism is calculated as the difference between the learning error (6) and the validation error (7).

Learning error is calculated in the bootstrap set with model trained in the same bootstrap set.

$$E_j^{A_j,A_j}(q,\theta^*_j(q)) = \frac{\sum_{i=1}^{N}\left(h^q(x_i^{A_j},\theta^*_j(q)) - y_i^{A_j}\right)^2}{N}, \tag{6}$$

where A_j is the j^{th} bootstrap set, x_i^{Aj} is the i^{th} input vector from the bootstrap set and y_i^{Aj} is the corresponding output.

Validation error is calculated in the initial training set with model trained on the same bootstrap set than the learning error.

$$E_j^{A_j,I}(q,\theta^*_j(q)) = \frac{\sum_{i=1}^{N}\left(h^q(x_i^I,\theta^*_j(q)) - y_i^I\right)^2}{N}. \tag{7}$$

Above described optimism calculation procedure is repeated as many times, or rounds, as possible considering linearly increasing computation time. The optimism

of a model is then calculated, as a mean of the difference of the two error functions described above

$$\hat{optimism}(q) = \frac{\sum_{j=1}^{J} E_j^{A_j,I}(q,\theta_j^*(q)) - E_j^{A_j,A_j}(q,\theta_j^*(q))}{J}, \tag{8}$$

where J is the number of bootstrap rounds done.

The final generalization error estimate is the sum of the training error and the optimism

$$\hat{E}_{gen}(q) = \hat{optimism}(q) + E^{I,I}(q,\theta^*(q)) . \tag{9}$$

Bootstrap 632 [5] is a modified version of the original Bootstrap. Where the original Bootstrap gives biased estimation of the generalization error (3), Bootstrap 632 is not biased [4] and thus is more comparable with other methods estimating the generalization error. Bootstrap 632 converges towards the correct generalization error in a reasonable amount of calculation time.

The main difference between standard Bootstrap and Bootstrap 632 is the estimation of optimism. In original Bootstrap the optimism is calculated as difference of the errors in two sets, in the initial training data set and in the randomly drawn bootstrap set (8). Bootstrap 632 estimates optimism using the data points not drawn into the bootstrap set (10). The model is trained using this set of unselected data points and its error is evaluated on the bootstrap set.

$$\hat{optimism}^{632}(q) = \frac{\sum_{j=1}^{J} E_j^{\overline{A_j},A_j}(q,\theta_j^*(q))}{J}, \tag{10}$$

where the error function is the same as the learning error (6), except the model is trained using $\bar{A}_j$, the complement of the bootstrap set A_j.

The estimation of the generalization error of the Bootstrap 632 is calculated as weighted sum of the training error and the new optimism (10).

$$\hat{E}_{gen}(q) = .368\,\hat{optimism}^{632}(q) + .632\,E^{I,I}(q,\theta^*(q)) . \tag{11}$$

From equation (11) it gets quite clear to see, that the name of the Bootstrap 632 comes from the weighting coefficient of the training error term. The value 0.632 is the probability of one sample to be drawn to the bootstrap set from the training set [2, 5].

5 Experimental Results

5.1 Time Series Prediction

Time series forecasting [6] is a challenge in many fields. In finance, one forecasts stock exchange courses or stock market indices; data processing specialists forecast the flow of information on their networks; producers of electricity forecast the load of

the following day. The common point to their problems is the following: how can one analyse and use the past to predict the future?

Next section will demonstrate how *k*-NN can be used to predict future values of a time series.

5.2 Results of Data Set 1: Electric Load

The dataset used in experiments is a benchmark in the field of time series prediction: The Poland Electricity Dataset. It represents the electric load of Poland during 2500 days in the 90's.

In our experiments we used the first half of the data set as a training set and the other half as a test set in order to evaluate the selected model's performance between different methods.

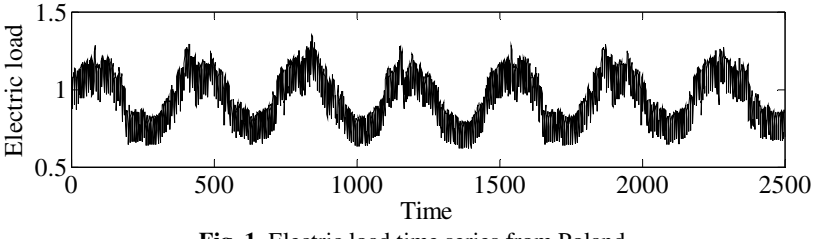

Fig. 1. Electric load time series from Poland

Fig. 2 and 3 show the generalization error estimates of all different methods according to the number of neighbors, when using the best selected input set.

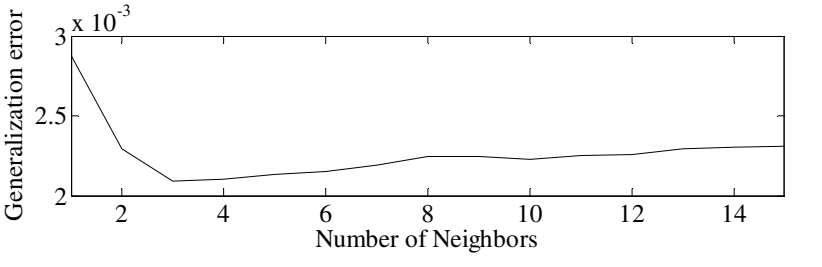

Fig. 2. The generalization error estimate using Leave-one-out according to the number of neighbors

The Table 1 shows the results of the experiments with three methods in selecting the inputs and number of neighbors. We have used $n = 8$ as a maximum number of inputs and $J = 100$ bootstrap rounds in calculations in both, Bootstrap and Bootstrap 632.

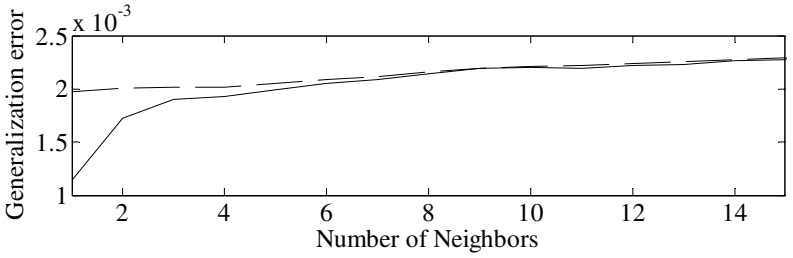

Fig. 3. The generalization error estimate using Bootstrap (solid line) and Bootstrap 632 (dashed line) according to the number of neighbors

Table 1. The results using Electric load data set and methods described in this paper

	Selected Inputs	*k*	$\hat{E}_{gen}$	Test error
LOO	t - {1, 2, 5, 7, 8}	3	0.0021	0.0011
Bootstrap	t - {1, 2, 5, 7, 8}	1	0.0011	0.0007
Bootstrap 632	t - {1, 2, 5, 7, 8}	1	0.0020	0.0007

All three methods select the same inputs but different number of neighbors. According to the test error both, Bootstrap and Bootstrap 632, select the best *k*. On the other hand, Bootstrap and Bootstrap 632 use *J* times more time than LOO.

In Fig. 4 we have used the model selected by bootstrap to predict 300 first test set values.

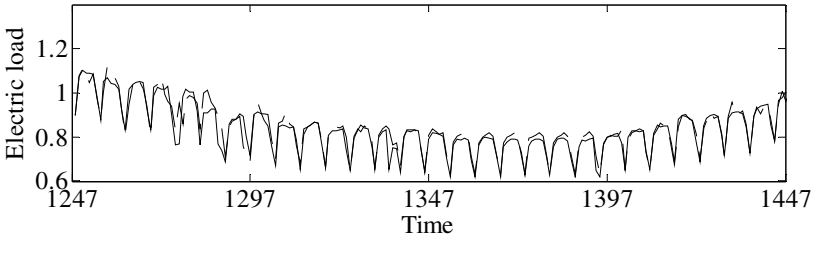

Fig. 4. Test set from Electric load data. 200 first values predicted and plotted. Solid line represents the real values and dashed represents the prediction

6 Conclusion

We have shown, that all methods, Leave-one-out and Bootstraps, select the same inputs. But number of neighbors is selected more efficiently by Bootstraps, according to the test error; even some studies have proven otherwise [2].

It has also been shown, that *k*-NN is a good approximator for time series. We have also tested *k*-NN with a couple of other time series and acquired the same results. As a conclusion we suggest Leave-one-out to be used in input selection and Bootstrap or Bootstrap 632 in selection of *k*.

Acknowledgements

Part the work of A. Sorjamaa, N. Reyhani and A. Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland.

References

1. Bishop C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995).
2. Efron B., Tibshirani R. J.: An introduction to the bootstrap. Chapman & Hall (1993).
3. Kohavi R.: A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Proc. of the 14th Int. Joint Conf. on A.I., Montréal (1995) 2:1137-1143.
4. Lendasse A., Wertz V., Verleysen M.: Model selection with cross-validations and bootstraps – Application to time series prediction with RBFN models. In: Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP (2003), Kaynak O., Alpaydin E., Oja E., Xu L. (eds): Springer-Verlag Lecture Notes in Computer Science 2714, Berlin (2003) 573-580.
5. Efron B., Tibshirani R. J.: Improvements on cross-validation: The .632+ bootstrap method. J. Amer. Statist. Assoc. (1997) 92:548–560.
6. Weigend A.S., Gershenfeld N.A.: Times Series Prediction: Forecasting the future and Understanding the Past. Addison-Wesley, Reading MA (1994).