HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Laboratory of Computer and Information Science

Strategies for the Long-Term Prediction of Time Series using Local Models

Master's Thesis

Antti Sorjamaa

Laboratory of Computer and Information Science
Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Laboratory of Computer and Information Science

ABSTRACT OF MASTER'S THESIS

Author: Antti Sorjamaa

Title of thesis:
Strategies for the Long-Term Prediction of Time Series using Local Models

Date:	October 14th 2005	**Pages:**	12 + 43
Professorship:	Computer and Information Science	**Code:**	T-115
Supervisor:	Professor Olli Simula		
Instructor:	Doctor Amaury Lendasse		

Long-Term Prediction is a difficult task in time series prediction field. One needs to overcome many different problems in order to predict further in time than only to the next one. Therefore, it is very crucial to select and develop good and accurate strategies to increase the accuracy and to decrease the prediction errors. In this thesis, 3 different long-term prediction strategies are presented and compared: Recursive, Direct and Dirrec.

Two local model families, the k-NN and the Lazy Learning, are presented and compared. Both families are presented as in their original forms as well as with the improvements developed. Both families need some structure selection methods and in this thesis 4 different methods are presented: k-fold Cross validation, Leave-one-out, Bootstrap and Bootstrap 632.

Another problem closely related to the time series prediction is the selection of appropriate inputs. This aspect is also dealt in this thesis by improving the original local methods by including the input selection in them.

All approximation methods, input selection methods and prediction strategies are compared using several different time series: Santa Fe, Darwin Sea Level Pressure and Poland Electricity Load. One competition task is also used, called the CATS Benchmark.

Keywords: Time Series, Long-Term Prediction, Prediction Strategies, Input Selection, Local Models, Lazy Learning, k-NN
Language: English

TEKNILLINEN KORKEAKOULU	DIPLOMITYÖN TIIVISTELMÄ
Tietotekniikan osasto	
Informaatiotekniikan laboratorio	

Tekijä:	Antti Sorjamaa		
Työn nimi:	Strategies for the Long-Term Prediction of Time Series using Local Models		
Päiväys:	14. lokakuuta 2005	**Sivumäärä:**	12 + 43
Professuuri:	Informaatiotekniikka	**Koodi:**	T-115
Työn valvoja:	professori Olli Simula		
Työn ohjaaja:	tohtori Amaury Lendasse		

Aikasarjan arvojen ennustaminen pitkän matkan päähän on erittäin vaikeaa ja useita ongelmakohtia on otettava huomioon. Siksipä onkin tärkeää kehittää hyviä ja tarkkoja keinoja päästäkseen ennustamisessa hyvään tarkkuuteen ja pitääkseen virheet mahdollisimman pieninä. Tässä diplomityössä esitellään kolme erilaista pitkän matkan ennustusstrategiaa: rekursiivinen (Recursive), suora (Direct) sekä näiden yhdistelmä (Dirrec).

Työssä käytetään kahta ennustusmenetelmää: k:n lahimmän naapurin menetelmää ja menetelmää nimeltä Lazy Learning ("laiska oppiminen"). Molempien menetelmien kohdalla esitellään ja vertaillaan alkuperäisiä menetelmiä sekä niihin tehtyjä parannuksia. Kaikki menetelmät tarvitsevat mallinvalintatyökaluja, joista 4 seuraavaa esitellään tarkemmin: k-kertainen ristiinvalidointi, Leave-one-out ristiinvalidointi, Bootstrap sekä Bootstrap 632.

Toinen tiiviisti aikasarjaennustamiseen liittyvä pulma on oikean syötteen valinta. Tässä työssä syötteen valinta on sisällytetty edellä mainittujen menetelmien parannuksiin.

Kaikkien mainittujen menetelmien ja strategioiden toimintaa on vertailtu kolmen eri aikasarjan avulla: Santa Fe, Darwin Sea Level Pressure sekä Poland Electricity Load. Lopuksi otetaan osaa CATS Benchmark -kilpailuun parhaaksi todetulla menetelmällä.

Avainsanat:	Aikasarjat, aiksarjaennustaminen, pitkän matkan ennustusstrategiat, syötteen valinta, paikalliset mallit, Lazy Learning, k:n lähimmän naapurin menetelmä
Kieli:	Englanti

Acknowledgements

This thesis has been made in the Laboratory of Computer and Information Science (CIS) in Helsinki University of Technology (HUT).

First of all, I want to thank my supervisor Professor Olli Simula, for his flexible supervision and helpfulness, and my instructor Doctor Amaury Lendasse, for his infinite wisdom and guidance throughout the process of writing this thesis. It has been an honor and a pleasure to work with both of them.

Many thanks goes to the CIS lab, which has been a very good place to work, mostly because of the relaxed and efficient atmosphere among the co-workers and the well-equipped and stimulating facilities. One HUGE thanks goes especially to the TSP Group. It has been fun to work with all of you guys; Jin Hao, Yongnan Ji, Nima Reyhani, Elia Liitiäinen and Tuomas Kärnä plus all the visitors and part-time group members. Special thanks also to Jan-Hendrik Schleimer for his endless help and patience in helping me with the painful world of LaTeX.

Last thanks go to all my crazy friends who have kept me sane during these long and busy school years and to my parents for their unending love, strong support, believing in me and cheering for me throughout my life.

Espoo October 14^{th} 2005

[illegible]

Antti Sorjamaa

Abbreviations and Acronyms

ANN	Artificial Neural Network
AR	AutoRegressive
Boot	Bootstrap
Boot 632	Bootstrap 632
GPLL	Globally Pruned Lazy Learning, LL with global input selection
LL	Lazy Learning
LOO	Leave-one-out
LPLL	Locally Pruned Lazy Learning, LL with local input selection
LS-SVM	Least Squares Support Vector Machines
LT	Long-Term
k-NN	k Nearest Neighhbors
MA	Moving Average
MSE	Mean Square Error
PRESS	PREdicted Sum of Squares
TSP	Time Series Prediction

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many fields of science use time series and time series prediction [1]. In finance, specialists predict stock exchange courses or stock market indices for the following days or weeks, data processing people predict the flow of information on their networks and electricity producers predict the electricity load of the following day. The common point to all fields is the following problem: "How to use the knowledge of the past to predict the future?"

In the field of time series prediction, the goal is to predict the future values of the time series using the previous values. In meteorology, it is very common to register the temperature every day and then predict the next days temperature from the gathered data. In the same way the meteorologists predict the cloud movement, the air pressure change and other weather phenomena and finally show the results in the nine o'clock news for everybody.

Previous example might sound quite straightforward and easy, but there is huge amount of calculation and modeling behind all that. In many cases, even if we know that it is possible to mathematically derive the dependencies between the measurements and the time spaces, formulation can be very hard task and sometimes the complexity makes it almost impossible. Maybe the underlying phenomenon is not only a function of the previous states and values of the single time series, but also of some unknown states or external variables. Finding the states and the variables makes the formulation task even harder and increases the complexity exponentially.

To start the prediction process the first task is to choose an appropriate model family. There are many different model families to choose from such as linear methods (autoregressive (AR) models, moving-average (MA) models) [2] and non-linear methods (artificial neural networks (ANN)) [1]. In order to select the most optimal family, one can use some prior knowledge of the problem or to do some preliminary tests to have some insight of the nature of the time series. If the group of models is too large or the models are too complex, it takes too much time to test all of them. On the other hand, simple models allow a larger group of models to be tested in a reasonable time.

In many cases, the methods try to build a model of the underlying process and then use the acquired model and the last values of the time series to predict the future values. In order to do this, the method must overcome a large number of obstacles, in example the selection of the relevant information to use in the prediction, the selection of method-specific structural parameters and finally optimizing the parameters of the selected model using a learning procedure.

Input selection is the most troubling obstacle. Usually the number of previous values needed for the prediction increases according to the complexity of the time series. Some values are more important (contain more information) than others and sometimes unnecessary inputs even decrease the accuracy of the prediction and increase the calculation time needlessly. Input selection is especially needed, when the number of data is too few compared to the number of possible inputs. On the other hand, too few inputs are not enough to build an accurate model of the underlying system.

In this thesis, two families of models are selected: local constant models (k-NN) and local linear models (Lazy Learning). The first one is very simple approximator family with only one structural parameter to select. The second one is more time taking, but more accurate approximator. Both families can use input selection methods as part of the routine itself.

For the k-NN, four ways to select the inputs are presented: Exhaustive Search, Backward Selection, Forward Selection and Forward-Backward Selection. The last one is a combination of the Forward and the Backward Selection methods.

The Lazy Learning model family uses Backward Selection method to select the inputs. The selection is done locally and globally and the performances are compared. Also continuously selected inputs are used and compared.

The prediction problem comes even more complex and demanding when the prediction is needed for several steps ahead, called a long-term prediction. Reaching for further to the unknown future increases the amount of uncertainty, which comes from many sources, in example from the accumulation of the prediction errors and the lack of relevant information. Also the calculation time and the total number of parameters, which need to be chosen, increase.

The selection of the prediction strategy becomes even more important the further in time the prediction is done. In this thesis, three different long-term prediction strategies are compared: Recursive, Direct and Dirrec. From these strategies, the Recursive is the fastest, but it suffers from the accumulation of the prediction error. The Direct strategy avoids the accumulation effect with the cost of increase in the calculation time. Finally the Dirrec strategy combines the Recursive and Direct strategies into one.

Section 2 defines the basic concepts related to the time series, some means of manipulation necessary for the time series prediction and the different strategies to manage long-term prediction. Section 3 describes in more detail the structure selection methods, the k-NN approximator and various Lazy Learning methods. All presented methods are applied to the long-term prediction problem using three data sets and the results are presented in Section 4.

1.1 Publications

Short descriptions of the publications [3, 4, 5, 6, 7] related to the thesis work in a chronological order:

Publication [3] describes how to use the LL methods in classification and function approximation. The task is to classify business plans into two categories: successful and unsuccessful ones. This is done by using the LPLL method, which combines locality with input selection. This publication has been selected for a special issue in International Journal of Neural Systems.

Publication [4] takes the LL methods further away from the original one. The input selection is done globally and the performance is compared to LL without input selection and to the k-NN method.

In publication [5] the k-NN is used as an input selection method with three different structure selection methods: Leave-one-out, Bootstrap and Bootstrap 632. After the input selection, the k-NN is also used as an approximator to compare the structure selection methods.

Publication [6] is a joint venture between the k-NN and the Mutual Information. Here the k-NN is used to tune the Mutual Information procedure and to evaluate its input selection performance in a long-term prediction problem.

Publication [7] compares three different input selection criteria: k-NN, Mutual Information and Nonparametric Noise Estimation. Each criteria is used to select the inputs based on three methods: Forward, Backward and Forward-Backward selection. Input selection performance is evaluated using LS-SVM.

Chapter 2

Time Series Prediction

2.1 Time Series

Even there exists many different time series with many different properties, we have to define some ground rules in order to use the methods described later. This first chapter defines the basic notations and assumptions about the series we are using in this thesis.

Single value of the time series is denoted as $x(t)$ and, given the definition of time t, we have a total of N values of the series x, that is from $x(1)$ to $x(N)$, from the oldest measurement to the newest one, which can be considered to be the present measurement of the series. Formal definition of a time series is given in Equation 2.1.

$$\{x(t)\}_{t=1}^{N}, \; x(t) \in \Re, \; 1 \leq t \leq N. \tag{2.1}$$

If the present or current time is denoted with t, the future is then denoted with $t+n$ and the past time as $t-n$, where n can be any positive integer with maximum of $N-1$ when denoting the past time. Time is always considered to be divided into steps, *time steps*, of equal length. The length can be almost anything from a few microseconds up to several years, but always a constant throughout the series.

2.2 Time Series Prediction

All the methods described in this thesis can be applied to a time series with *external inputs*. They can be described as some other related time series, than the one being predicted, like humidity in the case of predicting the temperature. External inputs can also be other kind of information, not only the type of time series, in example a working state of a machine.

However, in order to avoid complex notation we do not include the use of the external inputs in this thesis. Therefore, all the approximations and selections in the experiments are based only on the measured values of the time series itself. From this measured data, the methods try to approximate the future after some structural and parametrical finetuning called *learning*.

First phase is to select a *class of models*, in example Multi-Layer Perceptron networks [8] or Radial Basis Function networks [9]. The classes of interest in this thesis are local linear models and local constant models.

After the definition of the class, or classes, we can define all the models in the classes, as

$$f_q(\mathbf{x}, \theta(q)),\ 1 \leq q \leq Q, \tag{2.2}$$

where q is the index number of a model structure from all the classes with the total number of structures Q, $\mathbf{x}$ is the vector of time series values and $\theta(q)$ includes the parameters of the model f_q. It is essential to clarify, that structure becomes a model after all the parameters are defined or fixed one way or the other. If one of the parameters is changed, the model is definitely changed, but the structure can remain the same.

The ultimate goal is to find the best model f_q among the Q possible structures to best fit our prediction purposes. At the same time one has to consider the problem of finding the optimal parameters $\theta(q)$ for each structure to be able to rank the structures and find the best one. Then the best structure with corresponding parameters is used to predict the needed values of the time series.

In order to do the ranking of the structures and models, we need a definition of *prediction error*, which comes from the inaccuracy of the approximation, presented as

$$\begin{gathered} \hat{x}(t+1) = f_q(x(t), x(t-1), ..., x(t-d+1), \theta(q)), \\ \text{or} \\ x(t+1) = f_q(x(t), x(t-1), ..., x(t-d+1), \theta(q)) + \epsilon_{t+1}, \end{gathered} \tag{2.3}$$

where $\hat{x}(t+1)$ denotes the approximation at time $t+1$, ϵ_{t+1} denotes the approximation error and d is the number of previous time series values used in approximation. From these two representations of prediction we can derive a formula for the prediction error

$$\epsilon_{t+1} = x(t+1) - \hat{x}(t+1). \tag{2.4}$$

If this error is zero, there is no noise in the time series and the approximation is exactly the real value. In the presence of noise, there should always be a small error in the prediction, or we'll be dealing with *overfitting.* It means, that we have also modeled the error in the time series and that should be avoided or the accuracy of the prediction is degraded in the long run. This happens because time series are not completely deterministic, they are at least partly stochastic processes.

The approximation error can also come from other sources, in example from incorrectly selected parameters or structure or the insufficient number of learning data available.

Models from different classes can have totally different properties, such as computational load, simplicity, robustness and tolerance of noise. It is not obvious how to choose the best structure among many different classes. Some classes have certain limitations and may be harder to implement due to the number of free parameters to tune or the assumptions made about the underlying noise in the time series. Selection of the classes can be based on the prior knowledge about the type of the problem itself or many different classes should be tested and validated to find the most optimal one.

Structure selection methods used in this thesis are described more deeply in Section 3.1.

Although no external information or variables, besides the time series itself, are used, we can use some heuristics to decide how to preprocess the data [10]. Preprocessing methods include in example removing the mean of the data set

$$\left\{ x_{\text{zeromean}}(t-n) = x(t-n) - \frac{1}{N}\sum_{h=0}^{N-1} x(t-h) \right\}_{n=0}^{N-1}, \tag{2.5}$$

removing the trend

$$\{x_{\text{notrend}}(t-n) = x(t-n-1) - x(t-n)\}_{n=0}^{N-2} \tag{2.6}$$

and scaling the sample variance to one

$$\left\{ x_{\text{unitvariance}}(t-n) = \frac{x(t-n)}{\frac{1}{N-1}\sum_{h=0}^{N-1} x_{zeromean}(t-h)^2} \right\}_{n=0}^{N-1}. \tag{2.7}$$

In above equations, h is used as a temporary variable. Note that removing the first order trend makes $x_{notrend}$ one value shorter than the original time series. It is also possible to use higher order trend removal with trivial changes to Equation 2.6 [10].

Many approximation methods have hard time dealing with trends or non-zero means in the data and benefit vastly from the applied preprocessing.

After the approximation of the future values, it's many times necessary to reverse the preprocessing in the approximations to see the real prediction and to evaluate the prediction error.

2.3 Long-Term Prediction Strategies

In many occasions, it's fairly easy and straightforward to predict the next value of the time series. But the further we delve into the future, the more uncertain we are and the bigger prediction errors we get.

The concept of *Long-Term* is not precise. It somewhat describes the amount of timesteps to be predicted toward the unknown future. One can find or create many different interpretations and definitions to the concept long-term. In this thesis the term is used when the prediction is done further than one step ahead. Maximum horizon of prediction used is 20 steps ahead and based on any definition that is considered to be long-term.

In the next sections, three different prediction strategies are explained and their pros and cons discussed from a theoretical point of view. In Section 4 these strategies are compared and the performances evaluated from a practical point of view.

2.3.1 Recursive

In the Recursive prediction strategy the same model is used over and over again and the previous predictions are used with the original data set as inputs to evaluate the next prediction. In this way, we are actually applying one-step-ahead prediction many times, recursively.

If we use a continuous input set of size 4, we can write the Recursive strategy as

$$\begin{array}{rcl}
\hat{x}(t+1) & = & f_q(x(t), x(t-1), x(t-2), x(t-3)), \theta(q)), \\
\hat{x}(t+2) & = & f_q(\hat{x}(t+1), x(t), x(t-1), x(t-2)), \theta(q)), \\
\hat{x}(t+3) & = & f_q(\hat{x}(t+2), \hat{x}(t+1), x(t), x(t-1)), \theta(q)), \\
\hat{x}(t+4) & = & f_q(\hat{x}(t+3), \hat{x}(t+2), \hat{x}(t+1), x(t)), \theta(q)), \\
\hat{x}(t+5) & = & f_q(\hat{x}(t+4), \hat{x}(t+3), \hat{x}(t+2), \hat{x}(t+1)), \theta(q)), \\
 & \vdots &
\end{array} \qquad (2.8)$$

The further we go, the more approximations are introduced to the input set and after $t+4$ all the inputs are approximations. Here the model structure q stays the same in each timestep.

Because the approximation is not perfect, there's some prediction error in the approximations, so we can write the above equations as

$$
\begin{aligned}
\hat{x}(t+1) &= f_q(x(t), x(t-1), x(t-2), x(t-3)), \theta(q)), \\
\hat{x}(t+2) &= f_q(x(t+1)+\epsilon_{t+1}, x(t), x(t-1), x(t-2)), \theta(q)), \\
\hat{x}(t+3) &= f_q(x(t+2)+\epsilon_{t+2}, x(t+1)+\epsilon_{t+1}, x(t), x(t-1)), \theta(q)), \\
\hat{x}(t+4) &= f_q(x(t+3)+\epsilon_{t+3}, x(t+2)+\epsilon_{t+2}, x(t+1)+\epsilon_{t+1}, \\
&\quad x(t)), \theta(q)), \\
\hat{x}(t+5) &= f_q(x(t+4)+\epsilon_{t+4}, x(t+3)+\epsilon_{t+3}, x(t+2)+\epsilon_{t+2}, \\
&\quad x(t+1)+\epsilon_{t+1}), \theta(q)), \\
&\vdots
\end{aligned}
\tag{2.9}
$$

If the prediction errors are all zero, the Recursive strategy is very fast and accurate in the prediction of the future values, even in long-term. It needs only one model to be learned and after that any number of timesteps can be predicted.

But if the errors are non-zero and the approximations are used as inputs again and again, the more and more cumulative prediction error is included in the approximations. In practice, this is the normal case, there is some error in every approximation, because the time series are partly stochastic processes.

2.3.2 Direct

Comparing to the Recursive strategy, the Direct strategy uses different model for each time step but always the real measured data as inputs. No approximations are introduced to the input set.

Again, if we use a continuous input set of size 4, we can write the Direct strategy as

$$
\begin{aligned}
\hat{x}(t+1) &= f_{q_1}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_1)), \\
\hat{x}(t+2) &= f_{q_2}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_2)), \\
\hat{x}(t+3) &= f_{q_3}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_3)), \\
\hat{x}(t+4) &= f_{q_4}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_4)), \\
\hat{x}(t+5) &= f_{q_5}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_5)), \\
&\vdots
\end{aligned}
\tag{2.10}
$$

In this strategy, there's no cumulative error introduced through the inputs, because only original data set values are used in the approximation of future values. Each time step only the normal prediction error e_{t+n} is present.

Every time step incorporates its own model and may also have its own selection of inputs, if the input selection is used. These selections increase the calculation time considerably, but in practice give better results in the long-term prediction due to the lack of cumulative error [11].

2.3.3 Dirrec

The Dirrec strategy combines aspects from both, the DIRrect and the RECursive strategies. It uses a different model at every time step and introduces the approximations from previous steps into the input set.

If we use a continuous input set of size 4, we can write the Dirrec Strategy as

$$\begin{array}{rcl}
\hat{x}(t+1) & = & f_{q_1}(x(t), x(t-1), x(t-2), x(t-3)), \theta(q_1)), \\
\hat{x}(t+2) & = & f_{q_2}(\hat{x}(t+1), x(t), x(t-1), x(t-2), x(t-3)), \\
 & & \theta(q_2)), \\
\hat{x}(t+3) & = & f_{q_3}(\hat{x}(t+2), \hat{x}(t+1), x(t), x(t-1), x(t-2), \\
 & & x(t-3)), \theta(q_3)), \\
\hat{x}(t+4) & = & f_{q_4}(\hat{x}(t+3), \hat{x}(t+2), \hat{x}(t+1), x(t), x(t-1), \\
 & & x(t-2), x(t-3)), \theta(q_4)), \\
\hat{x}(t+5) & = & f_{q_5}(\hat{x}(t+4), \hat{x}(t+3), \hat{x}(t+2), \hat{x}(t+1), x(t), \\
 & & x(t-1), x(t-2), x(t-3)), \theta(q_5)), \\
 & \vdots &
\end{array} \tag{2.11}$$

Every time step the input set in increased with one more input, the approximation of the previous step. When we use the input selection, we can determine if the approximation is accurate enough to be included in the next step and so on. So in a way, the Dirrec strategy gives not only the prediction of each step, but also information about the validity of the approximations done in the previous steps.

If no input selection is used, the complexity of the model increases linearly and more and more inputs with prediction error are fed into the model. The cumulative prediction error increases also linearly, but is always less than in the Recursive strategy, because the real measurements remain as inputs.

Chapter 3

Approximation Methods

This section describes four model structure selection methods, k-fold Cross Validation, Leave-one-out, Bootstrap and Bootstrap 632, and two classes of models, k-Nearest Neighbors and Lazy Learning. The first model class includes local constant models and the latter one local linear models. Both classes include input selection methods inside the structure selection if needed. So there is no need for external input selection, but it can be used if available.

For all the methods, we need to transform the time series prediction problem to a function approximation problem and by that make it more usable for the models and the learning process. In order to do so, we construct an *input matrix*, or *regressor matrix* (also known as *regression matrix*), and an *output vector*, denoted $\mathbf{X}$ and $\mathbf{Y}$ respectively. They are illustrated as

$$\begin{gathered}
\mathbf{X} = \begin{bmatrix} x(1) & x(2) & \cdots & x(d) \\ x(2) & x(3) & \cdots & x(d+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-d) & x(N-d+1) & \cdots & x(N-1) \end{bmatrix}, \\
\mathbf{Y} = \begin{bmatrix} x(d+1) \\ x(d+2) \\ \vdots \\ x(N) \end{bmatrix},
\end{gathered} \tag{3.1}$$

where d is the maximum number of input variables (or inputs) and N is the number of values in the time series.

From this point on, each row of $\mathbf{X}$ and the corresponding value of $\mathbf{Y}$ is called a data sample or a sample. Each sample consists of d-dimensional input vector and an output scalar, denoted with $\mathbf{x}_i$ and y_i respectively, where i denotes the row of $\mathbf{X}$ and $\mathbf{Y}$. Approximation of one output scalar will be denoted with $\hat{y}_i$. The columns of the matrix $\mathbf{X}$ will be denoted with superscript $\mathbf{x}^n$ for the n^{th} column.

Another interesting aspect in the time series prediction is the input selection. It is neither computationally efficient nor necessary to use all the possible past values of the time series in the evaluation of the future values, but instead select only the most relevant ones and build the model using them. The selected inputs finally form the regressor matrix that is used in the prediction.

In this thesis, the input selection is included in the structure selection of the model, and few different input selection approaches are compared with each other.

3.1 Structure Selection Methods

Structure selection methods try to determine the structure, which is able to approximate as accurately as possible an unknown function. The structure is selected from a group of predefined structures, which can come from many different classes.

In many cases the selection of the structure is based on the *generalization error* [12], defined as

$$E_{\text{gen}}(q, \theta) = \lim_{M \to \infty} \frac{\sum_{i=1}^{M} (f_q(\mathbf{x}_i, \theta(q)) - y_i)^2}{M} \tag{3.2}$$

where $\mathbf{x}_i$ is d-dimensional input vector to the model f_q, $\theta(q)$ holds the model parameters and y_i is a scalar output corresponding to $\mathbf{x}_i$.

The data set of measurements is usually divided into two sets, to the learning set (or the training set) and to the test set. The structure of the model and the model parameters are selected and tuned or *learned* in the learning set. The test set is then used to estimate the overall model performance.

According to Equation 3.2, the generalization error is a mean square error of a model, computed on an infinite sized test set. Such a set is not available in practice, so we must approximate the generalization error. The best structure f_q is the one that minimizes the approximation of the generalization error.

The next four subsections present several generalization error approximation methods, which will be used in the structure selection process.

3.1.1 *k*-fold Cross-Validation

The k-fold Cross-Validation [13] (k-fold) is one of the resampling methods constructed to ease the problem of not having infinite learning data available. It is based on a simple Cross-Validation and can be used in the selection of structures, models and their parameters.

In the k-fold Cross-Validation, the training samples are divided into k approximately equal sized sets. One of the sets at a time is used as a learning set and all the rest as a validation set. The learning set is used to tune the parameters of the model and the validation set is used to approximate the validation error.

This procedure is repeated k times, for each structure until each of the k sets have been used once in the learning. The sets should be the same for all the structures included in the selection procedure, so no extra bias will be introduced to the approximations, if some folds are more difficult to approximate than others. For the same reason the folds should be as equally sized as possible in order to rank the structures correctly.

The final approximation of the generalization error is calculated as a mean of all k validation errors. The whole procedure is summarized as

$$\hat{E}_{\text{gen}}^{\text{kCV}}(q,\theta^*) = \frac{\sum_{s=1}^{k}\left[\sum_{i \notin s}^{N}(f_q(\mathbf{x}_i,\theta_s^*(q)) - y_i)^2\right]}{N}, \tag{3.3}$$

where s denotes the current fold and $\theta_s^*(q)$ holds the model parameters calculated using the samples in the s^{th} fold.

3.1.2 Leave-one-out

The Leave-one-out (LOO) [13, 14, 15, 16] is a particular case of the k-fold Cross-Validation resampling method. The LOO procedure is the same as the k-fold with k equal to the size of the training set N.

For each model to be tested, the LOO procedure is used to calculate the generalization error estimate by removing one sample at a time from the training set, building a model with the rest of the samples and calculating the validation error with the one taken out. This procedure is done for every sample in the training set and the estimate of the generalization error is calculated as a mean of all k, or N, validation errors. The process is summarized as

$$\hat{E}_{\text{gen}}^{\text{LOO}}(q,\theta^*) = \frac{\sum_{i=1}^{N}(f_q(\mathbf{x}_i,\theta_i^*(q)) - y_i)^2}{N} \tag{3.4}$$

where $\theta_i^*(q)$ holds the model parameters without using i^{th} sample in the learning process.

Empirically, the Leave-one-out is found out to be better approximation method than the k-fold Cross-Validation. That is because of the almost entire use of the given training samples in tuning the parameters and the large number of repetitions. On the other hand, the number of repetitions makes the LOO slower than the k-fold.

In the experiments, LOO is often used method in the selection of the parameters for the structures and the structures themselves. Despite the simplicity, it's one of the most reliable approximation methods and widely used in the field.

3.1.3 Bootstrap

The Bootstrap [17] is another resampling technique originally developed to estimate some statistical parameters, in example the mean of some population or the variance of it. In the case of a model structure selection, the parameter to be estimated is the generalization error.

When using the 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 [17]. This difference is called *optimism*. The final estimation of the generalization error will be the sum of the apparent error, Equation 3.5, and the estimated optimism, Equation 3.8.

The apparent error is computed using all the available samples on the training set as

$$E_{\text{Apparent}}^{I,I}(q,\theta^*) = \frac{\sum_{i=1}^{N}(f_q(\mathbf{x}_i^I,\theta^*(q)) - y_i^I)^2}{N}, \tag{3.5}$$

where I denotes the original training set samples.

The optimism is estimated using a resampling technique, based on a random drawing with replacement from the training set. This randomly created *Bootstrap set* is as large as the original training set with its participants drawn randomly from the training set. This random drawing is done many times in order to get an accurate approximation of the optimism.

A model is trained using one Bootstrap set at a time. the *Learning error*, Equation 3.6, is calculated in the Bootstrap set where the model is trained.

$$E_{\text{Learning},j}^{B_j,B_j}(q,\theta^*) = \frac{\sum_{i=1}^{N}(f_q(\mathbf{x}_i^{B_j},\theta_j^*(q)) - y_i^{B_j})^2}{N}, \tag{3.6}$$

where B_j is the j^{th} Bootstrap set, $\mathbf{x}_i^{B_j}$ is the i^{th} input vector from the Bootstrap set and $y_i^{B_j}$ is the corresponding output.

The *Validation error*, Equation 3.7, is calculated in the original training set using the same model as in the calculation of the learning error.

$$E^{B_j,I}_{\text{Validation},j}(q,\theta^*) = \frac{\sum_{i=1}^{N}(f_q(\mathbf{x}_i^I,\theta_j^*(q)) - y_i^I)^2}{N}, \tag{3.7}$$

The optimism is calculated as the difference between the learning error 3.6 and the validation error 3.7 and this calculation procedure is repeated as many times, or rounds, as possible considering linearly increasing computational time. The final approximation of the optimism for each structure is calculated as a mean of the difference of the two error functions, the learning error and the validation error as

$$optim\hat{i}sm(q) = \frac{\sum_{j=1}^{J}\left(E^{B_j,I}_{\text{Validation},j}(q,\theta^*) - E^{B_j,B_j}_{\text{Learning},j}(q,\theta^*)\right)}{J}, \tag{3.8}$$

where J is the number of bootstrap rounds done.

The final generalization error estimate is the sum of the apparent error, Equation (3.5), and the optimism estimate, Equation (3.8), as

$$\hat{E}^{\text{Boot}}_{\text{gen}}(q) = optim\hat{i}sm(q) + E^{I,I}_{\text{Training}}(q,\theta^*). \tag{3.9}$$

The Bootstrap does not produce comparable estimations of the generalization error, because it is biased [17]. Of course, it is usable as a structure selection method, where the errors are compared with other errors estimated by the Bootstrap using the same Bootstrap sets and so all the errors are biased in the same way. But with other generalization error approximation methods it cannot be compared directly.

The Bootstrap is also quite heavy method with a big computational load and thus, it is not good to use it in exhaustive structure selection, where all the possible structures are compared. The Bootstrap should be used in the validation of a smaller subgroup of structures or model parameters.

Every problem needs to have enough Bootstrap rounds and the same number of rounds may not be the best number for all the problems. Too many rounds increase calculation time needlessly and too few rounds give varying results with a large variance.

3.1.4 Bootstrap 632

Bootstrap 632 [18] is a modified version of the original Bootstrap. Where the original Bootstrap gives biased estimation of the generalization error, the Bootstrap 632 is not biased [18] and thus is more comparable with other methods estimating the generalization error.

The main difference in the calculation between the standard Bootstrap and the Bootstrap 632 is the estimation of the optimism. In the original Bootstrap the optimism is calculated as the difference between two errors, the learning error and the validation error, Equation 3.8. The Bootstrap 632 estimates the optimism using data points **not** drawn into the bootstrap set. The model is trained using a set of data points, which are not selected to a bootstrap set, and the error is evaluated on the same bootstrap set.

$$\widehat{optimism}^{632}(q) = \frac{\sum_{j=1}^{J} E_{\text{Boot632},j}^{\bar{B}_j, B_j}(q, \theta^*)}{J}, \tag{3.10}$$

where the error function is the same as the learning error, Equation 3.6, except the model is trained using $\bar{B}_j$, the complement of the bootstrap set B_j.

The estimation of the generalization error of the Bootstrap 632 is calculated as a weighted sum of the training error and the approximation of the optimism as

$$\hat{E}_{gen}^{Boot632}(q) = 0.368\, \widehat{optimism}^{632}(q) + 0.632\, E_{\text{Training}}^{I,I}(q, \theta^*). \tag{3.11}$$

From the above equation it is 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 a single sample to be drawn into the bootstrap set from the training set [17, 18]. The weight of the optimism is $1 - 0.632$, so that the total weight of the terms sum up to 1.

Because of the simpler optimism estimation formula, the Bootstrap 632 is almost twice as fast as the original Bootstrap. At the same time, the Bootstrap 632 provides unbiased estimation of the generalization error, so in theory it sounds much better and more reliable method than the Bootstrap.

3.2 *k*-Nearest Neighbors

The *k*-Nearest Neighbors (*k*-NN) approximation method is a very simple, but powerful method. It has been used in many different applications and particularly in classification tasks [8]. The key idea behind the *k*-NN is that similar training samples have similar output values. One has to look for a certain number of nearest neighbors, according to the Euclidean distance [8], and their corresponding output values to get the approximation of the desired output.

We calculate the estimation of the output simply by using the average of the outputs of the neighbors in the neighborhood as

$$\hat{y}_i = \frac{\sum_{j=1}^{k} y_{P(j)}}{k}, \tag{3.12}$$

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

It is possible to use some weighting of the neighbors in the neighborhood, different distance measure or more sophisticated neighbor selection methods, but these aspects are not considered here.

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

The k-NN has to somehow deal with the input selection problem, but in the case of k-NN, it is not so big problem, thanks to the simplicity of the k-NN. Only thing that is influenced by the input selection is the distance between samples, instead of influencing to the parameters of the model at the same time. Indeed, the k-NN is a method with no parameters whatsoever: only the structural aspects, the number of neighbors and the inputs, need to be determined. After that, the k-NN is ready to be applied to the problem at hand.

The following four sections describe the four different input selection methods: Exhaustive Search, Backward Selection (or pruning), Forward Selection and Forward-Backward Selection.

3.2.1 Exhaustive Search

In the Exhaustive Search, all the 2^d possible input variables are built and evaluated. d represents the maximum number of variables to be used in the evaluation. This kind of search is very time consuming, but it is guaranteed to give the global optimum in the defined search space.

In practice, the k-NN is almost only method simple enough to be able to use the Exhaustive Search. There are many methods, which are too complicated and time consuming to use this kind of search. Even with the k-NN, there might be a more optimal way to select the inputs in order to reduce the calculation time.

3.2.2 Backward Selection or Pruning

Each variable from x^1 to x^d is taken out one at a time. The k-NN is applied to each different pruned input set. Then the pruning that minimizes the error approximation is selected and the corresponding input is permanently taken away. This operation is repeated until there are no inputs left in the set. The optimal set of inputs is the one that gave the smallest error approximation.

In the Backward Selection, only $d(d-1)/2$ different input sets are evaluated with the k-NN. This is much less than the number of input sets evaluated with the Exhaustive Search.

On the other hand, optimality is not guaranteed. The selected set on inputs may not be the global optimal one, but instead the selection procedure can be stuck on some local minima. This disadvantage is common to all pruning methods [19].

3.2.3 Forward Selection

The Forward Selection is effectively the opposite of the Backward Selection. The initial input set is an empty set whereas with the Backward Selection the initial set was full.

Each variable from x^1 to x^d is put into the input set one at a time and the one that gives the smallest error after the k-NN approximation, is selected and permanently inserted into the set. This is continued until all the inputs are inserted into the set. The input set that gave the smallest error is selected to be used in the final evaluation.

The Forward Selection evaluates the same amount of input sets than the Backward selection, $d(d-1)/2$. Also the same restriction apply, optimality is not guaranteed.

3.2.4 Forward-Backward Selection

The Forward-Backward Selection combines both methods described in previous sections. It can start from any input set, even from randomly initialized set.

The state of each variable is changed one at a time: if a variable was already selected into input set, it is removed, and if not, it is put into the input set. This is done for all the inputs from x^1 to x^d and the error approximation using the k-NN is calculated with every selection. Then the action that gave the smallest error is done permanently to the input set.

In example, if we use non-continuous regressor of size 4 and describe the process of approximating one time step ahead as

$$\hat{x}(t+1) = f_q(x(t), x(t-1), x(t-3), x(t-5)), \theta(q)). \tag{3.13}$$

The Forward-Backward Selection tests all the following input sets and approximations presented in Table 3.1.

Change #	Input, t-{...}					
Try 1		1		3		5
Try 2	0			3		5
Try 3	0	1	2	3		5
Try 4	0	1				5
Try 5	0	1		3	4	5
Try 6	0	1		3		

Table 3.1: Forward-Backward Selection example.

The change that gives the smallest error is permanently done to the input set and the process continues by estimating the next set of inputs.

In this way it is continued until a stopping criteria is fulfilled and the input set giving the smallest error is selected. The criteria used in the experiments is to take the lowest error and try three steps further. If one of these steps gives smaller error than the previous smallest one, we try again three steps further from the new smallest one. If a smaller error is not achieved, then the input set giving the minimum error is selected and the selection procedure is terminated. The procedure is illustrated in Figure 3.1.

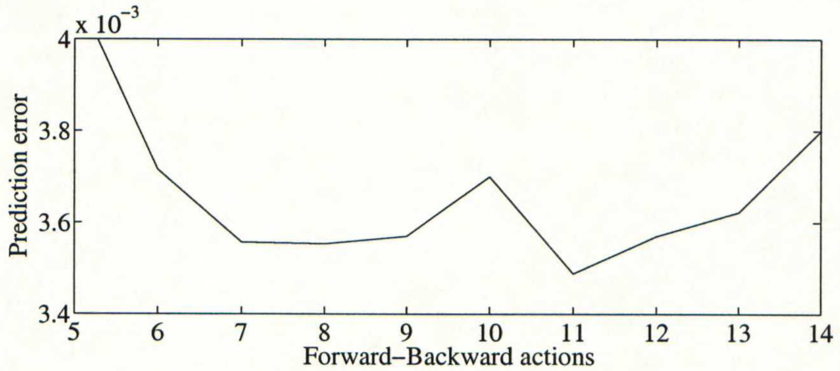

Figure 3.1: Example of Prediction error with the Forward-Backward Selection. Even the error increases after the action in step 8, according to the stopping criteria we try 3 steps further and find even smaller error in step 11, in this case the global minimum.

With this kind of stopping criteria, it is possible to 'leap over' some local minima and have 'more global' minimum, see Figure 3.1. Still, it is not guaranteed that in all cases this selection method will find the global optimal input set, even in theory it should be closer to the global optimum than the Backward or the Forward methods alone.

The number of different input sets evaluated varies. It's dependent from the initialization on the input set, the stopping criteria and the nature of the problem. Using some heuristics in the choices of initialization, it is possible to decrease the number of evaluated input combinations considerably.

3.3 Lazy Learning

The Lazy Learning (LL) models are linear piecewise approximation models based on a recursive least squares algorithm introduced by Aha [20]. Around each sample, the k-nearest neighbors are determined and used to build a local linear model. This approximation model is presented as

$$\hat{y} = LL(x^1, x^2, \cdots, x^d), \tag{3.14}$$

where $\hat{y}$ is the approximation of the real output y and x^1 to x^d represent the d selected inputs.

For the original Lazy Learning the inputs are selected in a continuous fashion. Therefore, only the number of inputs is needed to be determined beforehand. In general case, it is not necessary to use all the inputs from 1 to d, but instead select the most necessary ones in a non-continuous fashion. In both cases, the input selection must be done before the LL model can be used in the prediction.

The optimization of the number of neighbors is also crucial. When the number of neighbors is small, local linearity assumption is valid. On the contrary, if the number of neighbors is large, the local linearity assumption is not valid anymore and the linear model fails to provide good approximations. The choice of the number of neighbors is illustrated in Figure 3.2.

For an increasing size of the neighborhood, the Leave-one-out procedure, described in Section 3.1.2, is used to evaluate the generalization error of each different LL model structure [17]. Once the generalization error for each neighborhood size is approximated, the number of neighbors that minimizes the approximation is selected.

There is a recursive formula to speedup the calculation of the LOO estimations based on PRESS statistics. It will be presented in Section 3.3.1.

The main advantages of the LL models are the simplicity of the model itself and the low computational load. Furthermore, the local model can be built only around the sample for which the approximation is requested. This gives

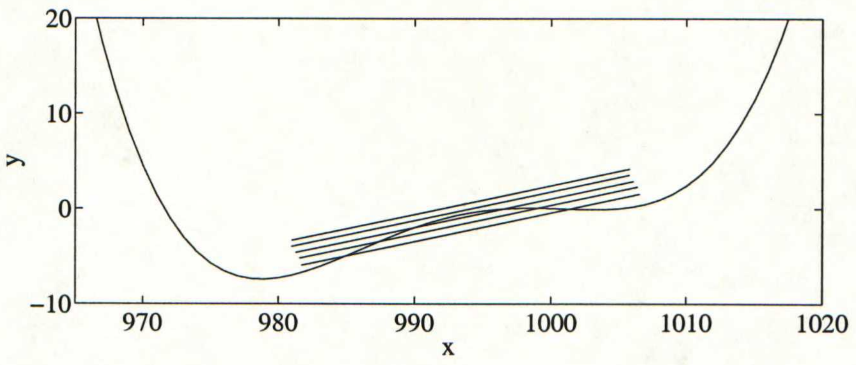

Figure 3.2: Toy example demonstrating the effect of different sizes of the neighborhood. The best approximation is achieved with the line in the middle. The ones above it have too few neighbors and the lines below it have too many.

the name to the Lazy Learning: there's no need to do anything before the approximation is requested.

It is also possible to use a *global neighborhood*, where the number of neighbors is the same for all the data points. In order to use the LL with the global neighborhood, the global size of the neighborhood must be determined beforehand. Each neighborhood size for each sample is evaluated and the size minimizing globally the estimation of the generalization error is selected.

If the density of the training samples is fairly constant, the use of global neighborhood really speeds up the calculation process and makes the approximations more accurate. If the density varies, it is wiser to use local neighborhood to get good approximations.

Unfortunately, as already mentioned, the inputs have to be known *a priori*. Several sets of inputs have to be evaluated to select the optimal one. The error has to be evaluated around each data point for each set of inputs. The optimal input set is the one that gives the smallest approximation of the generalization error. This procedure is quite long and reduces considerably the advantages of the Lazy Learning methodology.

The Lazy Learning and its improvements are briefly summarized in Figure 3.4 in the end of this section.

3.3.1 PRESS Statistics

The PRESS statistics [21] gives the tools to calculate the Leave-one-out Cross-Validation errors for linear models without excessive computation. When we add this remarkable mathematical derivation to the standard recursive least squares algorithm, introduced by Biermann [22], we have a nice and fast way to validate and select the LL structures.

This combination of methods and mathematical derivation is introduced in [23] and only briefly summarized here.

Everything starts by defining a prediction error for the linear model in the case of one sample left out in the same way than in the Leave-one-out. This prediction error is called the PRESS residual

$$\epsilon_{i,-i}^{\text{PRESS}} = y_i - \hat{y}_{i,-i} = y_i - \mathbf{x}_i\mathbf{b}_{-i}. \tag{3.15}$$

where i is the index number of the sample and $-i$ means the i^{th} sample is left out, $\mathbf{x}$ is the input vector, y is the output and $\mathbf{b}$ includes the linear model parameters. Then the PRESS, or PREdiction Sum of Squares, is defined as

$$\text{PRESS} = \sum_{i=1}^{N}(\epsilon_{i,-i})^2 = \sum_{i=1}^{N}(y_i - \mathbf{x}_i\mathbf{b}_{-i})^2, \tag{3.16}$$

which is effectively the same as the generalization error approximation (MSE) by the Leave-one-out in Section 3.1.2 multiplied by N, which is the overall number of the samples.

The power of the PRESS residuals is really understood when the possibility to calculate them without explicitly identifying the linear model parameters $\mathbf{b}_{-i}$ is introduced. For that, we need some mathematical derivation from [21] to get the final PRESS residual formula

$$\epsilon_{i,-i}^{\text{PRESS}} = \frac{y_i - \mathbf{x}_i\hat{\mathbf{b}}}{1 - \mathbf{x}_i'\mathbf{P}\mathbf{x}_i}, \tag{3.17}$$

where $\mathbf{P} = (\mathbf{X}'\mathbf{X})^{-1}$ and $\mathbf{X}$ contains all the neighbors in the neighborhood. The above formula can be understood as the linear model error in the point $\mathbf{x}_i$ divided with one minus the estimated prediction variance in the point $\mathbf{x}_i$. The point $\mathbf{x}_i$ is the one left out and the linear model is calculated from all the points in the neighborhood. The prediction variance cannot be greater than one or smaller than $1/k$, where k is the number of neighbors in the neighborhood. Greater prediction variance leads to greater PRESS residual, which in turn increases the LOO error of the neighborhood size in question.

Introducing the recursive formulas from the least squares algorithm

$$\begin{aligned}
\mathbf{P}(k+1) &= \mathbf{P}(k) - \tfrac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}'(k+1)\mathbf{P}(k)}{1+\mathbf{x}'(k+1)\mathbf{P}(k)\mathbf{x}(k+1)}, \\
\gamma(k+1) &= \mathbf{P}(k+1)\mathbf{x}(k+1), \\
\epsilon(k+1) &= y(k+1) - \mathbf{x}'(k+1)\hat{\mathbf{b}}(k), \\
\hat{\mathbf{b}}(k+1) &= \hat{\mathbf{b}}(k) + \gamma(k+1)\epsilon(k+1),
\end{aligned} \tag{3.18}$$

where $\mathbf{x}(k+1)$ is the next nearest neighbor when the neighborhood size is enlarged.

Finally, we have the recursive way to calculate the LOO errors for each neighborhood size of the LL structure.

$$\epsilon_{i,-i}^{\text{PRESS}} = \frac{y_i - \mathbf{x}_i'\hat{\mathbf{b}}(k+1)}{1 - \mathbf{x}_i'\mathbf{P}(k+1)\mathbf{x}_i}. \tag{3.19}$$

In each neighborhood size the PRESS residuals are evaluated. Then the LOO error of the neighborhood size is the mean of all k residuals. The parameters for the next neighborhood size are evaluated using recursive least squares formulas and again the PRESS residuals are evaluated. This is continued for all the necessary neighborhood sizes and the one giving the smallest PRESS is the chosen neighborhood size. The algorithm is summarized in Figure 3.3.

1. Initialization
   - Calculate initial linear model parameters using the smallest possible amount of neighbors, not smaller than the number of parameters in the linear model.
   - Calculate the initial $\mathbf{P}$.
   - Calculate the LOO error using the standard LOO approximation method.
2. Use Equation 3.18 formulas to calculate parameters for the next neighborhood size.
3. Calculate LOO error using Equation 3.19.
4. Repeat from step 2 until all neighborhood sizes are done.
5. Select the neighborhood size with the smallest LOO error.

Figure 3.3: Recursive PRESS statistics.

3.3.2 Globally Pruned

Globally Pruned Lazy Learning (GPLL) method [4] is a modified version of the LL that has been presented in Section 3.3. This method uses input selection to prune out the least important input variables in the same way as the Backward Selection or pruning method presented in Section 3.2.2.

The initial structure is built according to Equation 3.14 in Section 3.3 using some maximum number of inputs d. Selection of the initial number of inputs is highly problem dependent and varies from application to another.

Each input variable from x^1 to x^d is taken out one at a time. The Lazy Learning is applied to each pruned input set and the approximation of the generalization error is obtained. Then, the pruning that minimizes the error approximation is selected and the corresponding input is permanently taken away. This operation is repeated until there are no inputs left to prune. The selected set of inputs is the one that gave the smallest error approximation.

With this method $d(d-1)/2$ different input sets are built and evaluated. This is significantly less than the number of all possible input sets (2^d) that could be built. However, it is not guaranteed that the selected input set is globally the most optimal one. This disadvantage is common to all pruning methods [19].

3.3.3 Locally Pruned

Basically, Locally Pruned Lazy Learning (LPLL) method [3] is the same method as the GPLL presented in the previous section. The difference is that the LPLL prunes the inputs locally, around each sample one at a time. Thus each sample can have not only a different number of neighbors, but also different inputs, which are pruned when the approximation is needed. In this case, only the maximum number of inputs is needed to fix beforehand.

This method is more like-minded with the original Lazy Learning, it does not need so much work before the approximation is needed. All the necessary inputs, variables and parameters are tuned "on the fly".

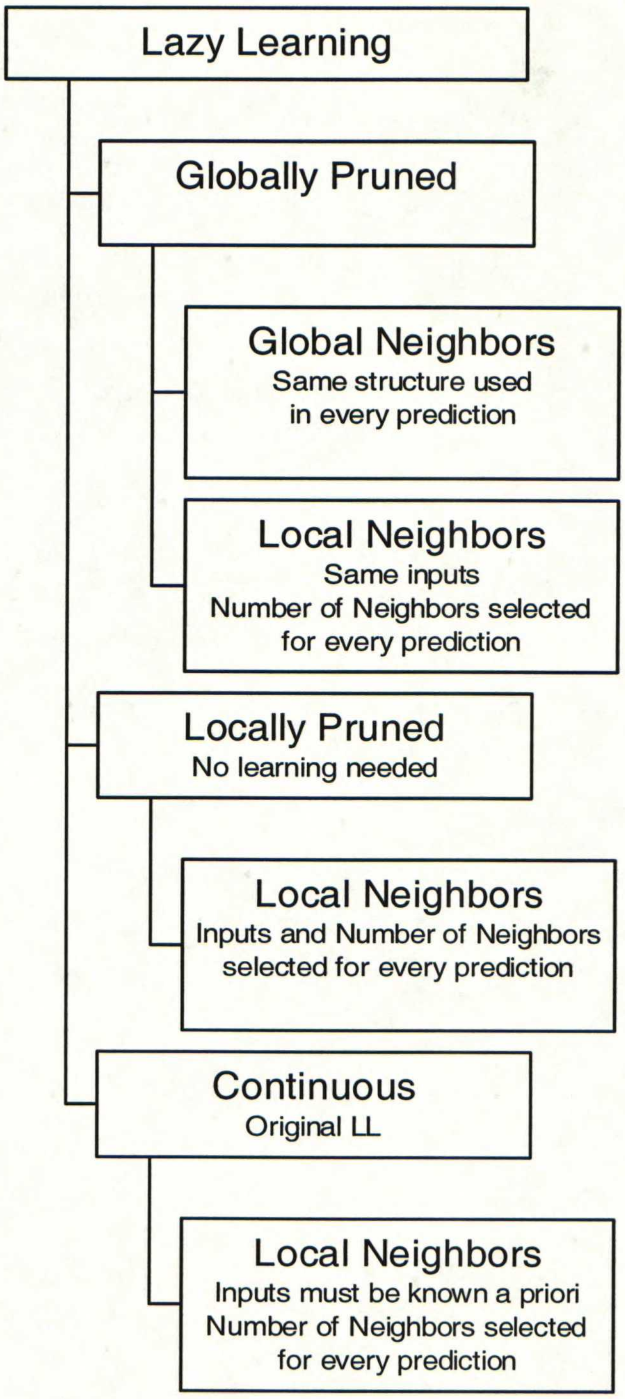
Lazy Learning
Globally Pruned
Global Neighbors
Same structure used in every prediction
Local Neighbors
Same inputs
Number of Neighbors selected for every prediction
Locally Pruned
No learning needed
Local Neighbors
Inputs and Number of Neighbors selected for every prediction
Continuous
Original LL
Local Neighbors
Inputs must be known a priori
Number of Neighbors selected for every prediction

Figure 3.4: Lazy Learning method summarized.

Chapter 4

Applications to Long-Term Prediction of Time Series

The very first thing to do is to select a structure selection method for the k-NN. The possible choices are the Leave-one-out, the Bootstrap and the Bootstrap 632, presented in Section 3.1. k-fold Cross Validation is not considered to be reliable enough [13] and therefore, it is not included in the comparison. Essentially, the Leave-one-out is the same as the k-fold with the number of folds equal to the number of samples.

After that, all approximation methods are used with 3 time series. For each benchmark two prediction strategies are used, the Recursive and the Direct strategy. The Dirrec strategy needs so much more computational time that it cannot be used with all the methods and therefore it is handled separately.

The last part of the chapter is an application to a competition task using the best method, selected using the performances in the three benchmarks.

The k-NN Selection inholds all the selection schemes except the Exhaustive search (Section 3.2). The input set with the smallest LOO error is selected and then used in the test of the k-NN Selection.

All time series have different properties and maximum bounds (maximum number of neighbors, maximum number of inputs, etc.) are also different for the tested structures. The bounds used in each case are given in the description of the time series in question.

For comparison, a continuous Linear model and the Linear model with input selection (the Backward Selection, 3.2.2) are presented. In the model structure selection of the Linear models, the Bootstrap 632, Section 3.1.4, is used.

To evaluate the accuracy of the predictions, Mean Square Error (MSE) is used in all test error tables as the measuring criteria.

4.1 k-NN Structure Selection Method

In order to choose the method for the k-NN structure selection, we use two time series. First one is the Poland Electricity Load data set, which is described and introduced more deeply in Section 4.4. The test errors, the selected inputs and the number of neighbors are presented in Table 4.1. The LOO and Boot 632 selection method is using the LOO as the input selection method and then the Bootstrap 632 in the selection of the number of neighbors.

Test Errors	
LOO	$1{,}8495 \cdot 10^{-3}$
Boot	$2{,}8482 \cdot 10^{-3}$
Boot 632	$2{,}8482 \cdot 10^{-3}$
LOO and Boot 632	$2{,}8482 \cdot 10^{-3}$

Inputs	Max: 8		t - {...}		
LOO	7	6	4	1	0
Boot	7	6	4	1	0
Boot 632	7	6	4	1	0

k	Max: 100
LOO	4
Boot	1
Boot 632	1
LOO and Boot 632	1

Table 4.1: k-NN structure selection using Poland Electricity Load. The LOO and Boot 632 means that LOO is used for the input selection and then the Bootstrap 632 for the selection of k.

According to the performance of the four selection schemes in the Poland Electricity Load data set, three methods are selected to the next round. In the next round, Santa Fe time series is used. The Santa Fe is described and introduced more deeply in Section 4.2. The test errors, the selected inputs and number of neighbors are presented in Table 4.2.

From the experiments above, the Leave-one-out structure selection method is clearly the best one according to the test errors in both time series. It is also the fastest method, which makes it the most preferable one to use in the input selection. Therefore the LOO is selected to be used with the k-NN in the following experiments as the input selection method as well as in the selection of k.

Test Errors	
LOO	53,64
Boot 632	58,92
LOO and Boot 632	58,92

Inputs	Max: 12	t - {...}	
LOO	11	1	0
Boot 632	11	1	0

k	Max: 100
LOO	3
Boot 632	1
LOO and Boot 632	1

Table 4.2: k-NN structure selection using Santa Fe. The LOO and Boot 632 means that the LOO is used for the input selection and then the Bootstrap 632 for the selection of k.

4.2 Benchmark 1: Santa Fe

The Santa Fe is a widely used benchmark and it was generated using an infrared laser in a chaotic state. It's originally introduced as one of the Santa Fe competition data sets [24]. It inholds over 10 000 values from which the 1000 first values, called Santa Fe A series in the literature, are used for learning and the rest, little bit over 9000 values, are used for the testing.

For the learning, maximum of 100 neighbors and maximum of 30 inputs are used for every method. Only exception is the k-NN with Exhaustive Search, which had to be limited to a maximum of 17 inputs, because of the calculation time limitations. Each input more doubles the calculation time needed in the case of Exhaustive Search.

The learning set is shown in Figure 4.1. The test errors are shown in Tables A.1 and A.2 using the Recursive and the Direct strategies respectively.

From the two tables it can be seen that the Direct strategy is many times better than the Recursive one. Overall best method is the GPLL with global k, but the k-NN with the Exhaustive Search is not far behind even the maximum number of inputs is only 17. Third best method is the GPLL with local k. The test errors of the best GPLL method are plotted in Figure 4.2.

Almost all methods except the Linear ones got better error using the Direct strategy than the Recursive. The prediction with the three best methods is presented in Figure 4.3 using the Direct prediction strategy. The inputs

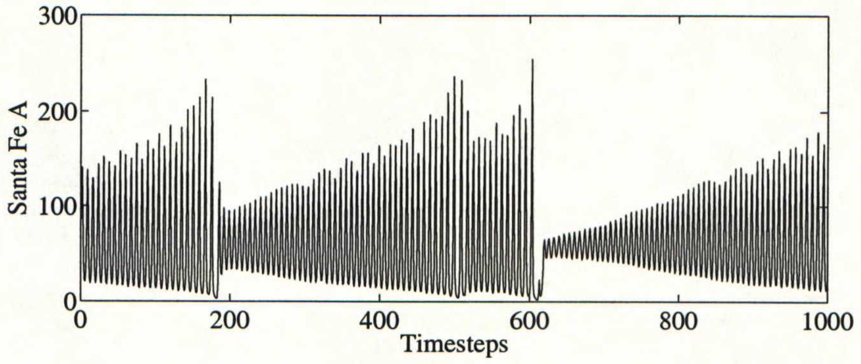

Figure 4.1: Santa Fe A, the learning set.

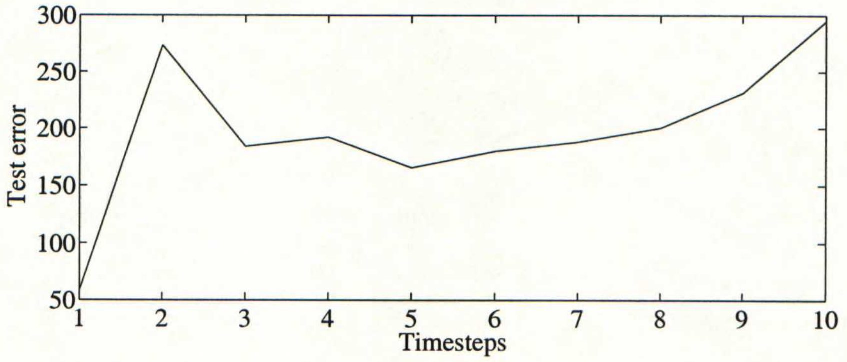

Figure 4.2: Santa Fe test errors using the GPLL with global k and the Direct prediction strategy.

used in the prediction are shown in Tables B.1 and B.2.

4.3 Benchmark 2: Darwin Sea Level Pressure

The Darwin Sea Level Pressure data set (called Darwin later on) consists of 1400 sea level air pressure values collected monthly from the Darwin Sea. The data can be found from [25]. This time series is the hardest benchmark from the three presented here. 1300 first values are used as the learning data and the 100 rest are used for testing. The learning set is shown in Figure 4.4.

For the learning, maximum of 500 neighbors and maximum of 30 inputs are used for every method. Only exception is the k-NN with the Exhaustive Search, which had to be limited to a maximum of 17 inputs, because of the calculation time limitations. Each input more doubles the calculation time

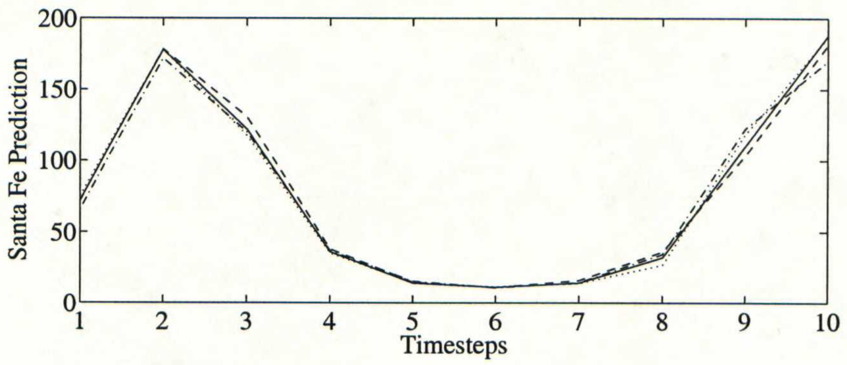

Figure 4.3: Santa Fe data 10 steps ahead prediction using the Direct prediction strategy. Solid line represents the real values, dashed is the GPLL with global k, dotted is the k-NN using the Exhaustive Search and dash-dotted is the GPLL with local k.

needed in the case of Exhaustive Search.

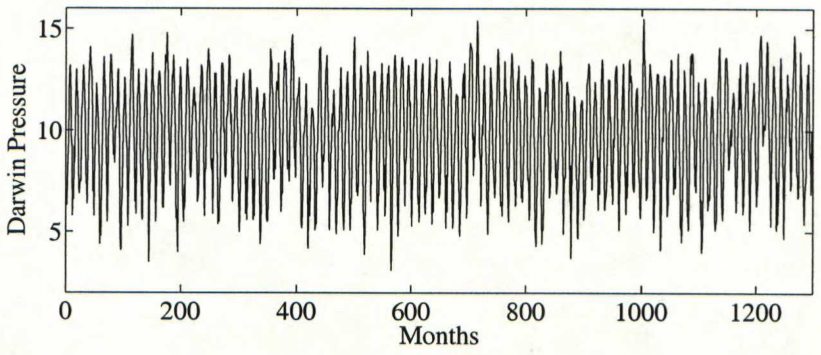

Figure 4.4: Darwin learning set.

The test error values are in Tables A.3 and A.4, the first table using the Recursive strategy and the latter using the Direct strategy. From the two tables it is quite evident and easy to see the trend, where the Direct strategy gives smaller errors more often than the Recursive one. It is also very surprising how well the Linear models perform against the local linear models and the k-NN.

If we consider timesteps from 1 to 5, the GPLL with global k is the best on average, but somehow the timesteps from 6 to 10 decrease the overall performance. On the other hand, the Darwin test set has only 100 values, which is too small to make very reliable conclusions. The lack of test data gives the test results more variance and makes the results more unstable and vague. At the same time, a huge number of neighbors needed suggest the

Darwin time series to be mostly linear, which would also explain the good success of the Linear models.

The test errors of the GPLL with global k are plotted in Figure 4.5. Figure 4.6 shows 3 best predictions, according to the mean MSE, over all 10 prediction steps. Tables B.3 and B.4 show the inputs used in the 3 best predictions.

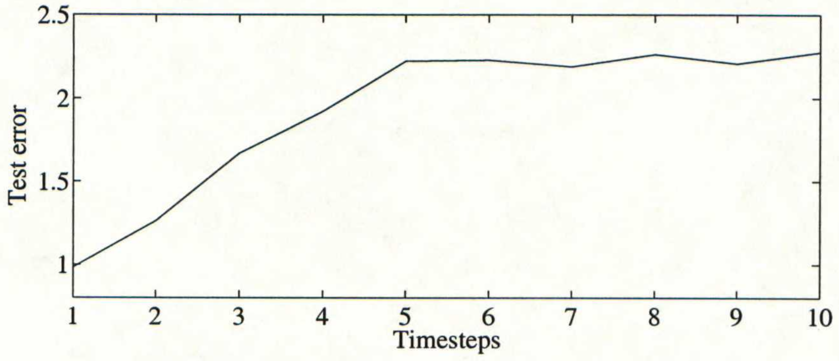

Figure 4.5: Darwin test errors using the GPLL with global k and the Direct prediction strategy.

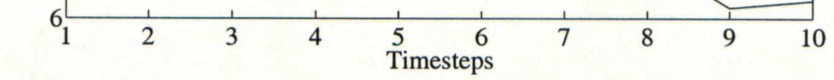

Figure 4.6: Darwin data 10 steps ahead prediction using the Direct prediction strategy. Solid line represents the real values, dashed is the Linear, dotted is the Linear with pruning and dash-dotted is the LL with global k. First two use the Direct strategy and the LL uses the Recursive strategy.

4.4 Benchmark 3: Poland Electricity Load

This data set describes the daily average of electricity load in Poland for over four years of time in the 1990's. The data set can be downloaded from

[26]. The data set is divided into a learning set, including 1400 values, and a separate test set, containing 201 values. The learning set is shown in Figure 4.7.

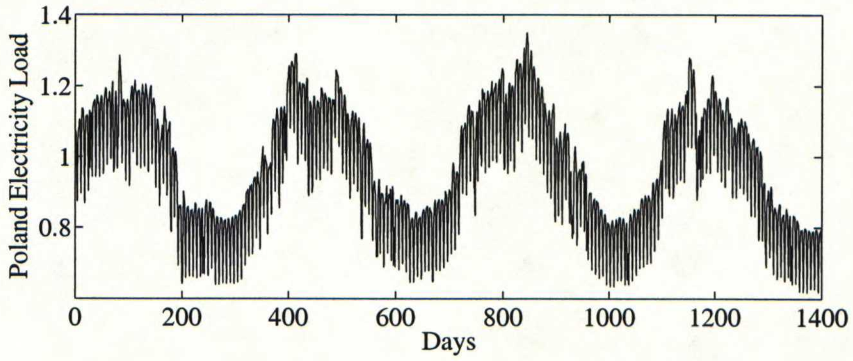

Figure 4.7: Poland Electricity Load learning set.

Tables A.5 and A.6 show the test errors using the Recursive and the Direct strategies respectively. In the tests the number of neighbors is limited to 200 and the number of inputs to 14.

It is quite surprising how well the Linear models perform with this time series and the fact that the smallest mean test error is achieved with the Recursive strategy and not with the Direct. It is true that the time series itself is fairly easy even for the basic Linear models to predict the future values and the Lazy Learning or the k-NN have very little left to improve in the prediction process.

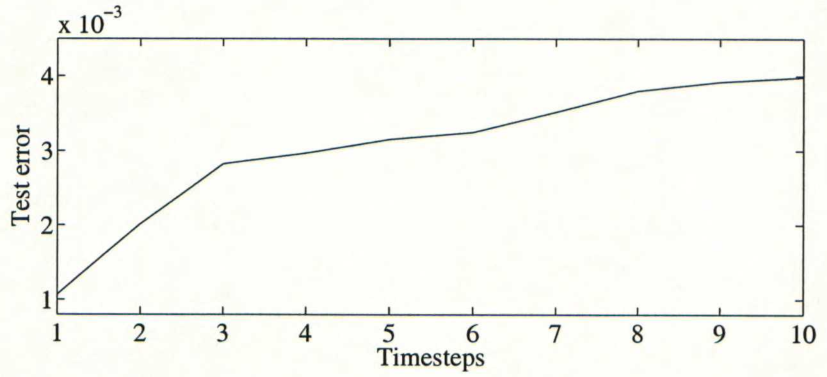

Figure 4.8: Poland Electricity Load test errors using the LL with global k and the Recursive prediction strategy.

However, the best method according to the mean test error is the LL with global k. The GPLL is not far from the LL, but the fact that the LL without the input selection gives better performance than with the input selection can be due to a small maximum number of inputs. The test errors of the best LL method are plotted in Figure 4.8.

From the test error tables, three best methods according to the mean test error are selected to predict 10 steps of the test set. The prediction results are shown in Figure 4.9 and the selected inputs in Table B.5

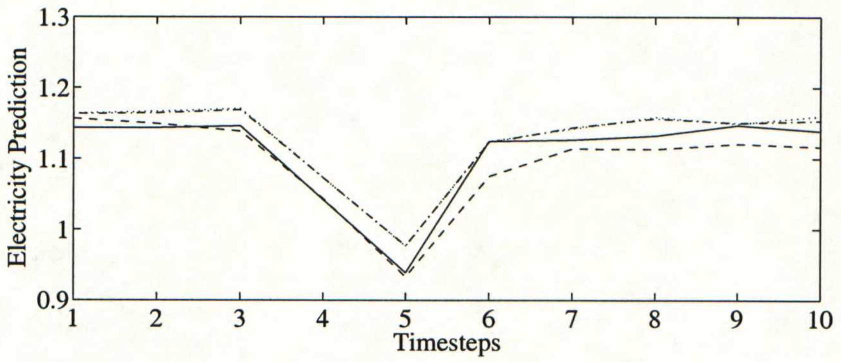

Figure 4.9: Poland Electricity Load 10 steps ahead prediction. Solid line is the real values, dashed is the LL with global k, dotted is the Linear with pruning and dash-dotted is the Linear continuous. The first method is done using the Recursive strategy and the two last ones with the Direct strategy.

4.5 Dirrec prediction strategy

In this section, the third prediction strategy, Dirrec strategy, is tested with two time series: the Santa Fe data set and the Poland Electricity Load data set. The Darwin series requires so many neighbors in the LL methods that it is too calculation intensive to use with the Dirrec strategy. Where the two other benchmarks use a maximum of 200 neighbors, the Darwin uses 500.

The criteria to select the methods to be used in the Dirrec experiments is based on the calculation time requirements. Methods selected must be able to produce all needed models for the 10 time steps in a reasonable time. Some methods take way too long time to be able to handle the linearly increasing regressor size of the Dirrec strategy.

With the Dirrec strategy the following methods are compared: the first one is the combination of the k-NN and the Leave-one-out, the second one is the joint venture of the k-NN and the Lazy Learning, third one is the continuous LL and finally the fourth one is the GPLL. In both combinations involving the k-NN, the used input selection method is the Forward-Backward Selection. The other participant in the group is used as the approximator. Both LL based methods use global k.

In Table A.7 the test errors of the 10 prediction steps are presented for both time series, the Santa Fe and the Poland Electricity Load. In Tables B.6 and B.7 are the selected inputs of Santa Fe and in Table B.8 are the selected inputs of Poland Electricity Load. In all the tables presenting the inputs, the worst method, according to the mean test error, has been left out.

From the results in Table A.7, the best overall results for both time series can be found. The mean test errors over all 10 prediction steps are smaller than with the Direct or with the Recursive strategy for both time series.

For the Santa Fe, the best method, according to the mean test error, is the k-NN with the LOO. The GPLL method does not perform as well as the k-NN or even as well as the GPLL with the Direct method. This can be due to the inefficient input selection which is in critical role, because of the increasing number of inputs. The prediction of the best method is shown in Figure 4.10.

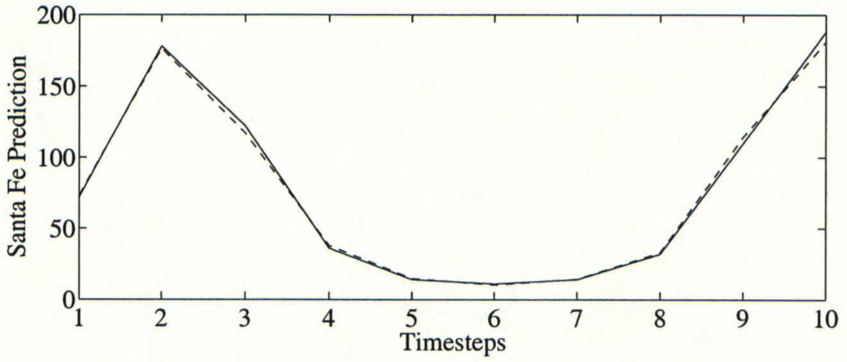

Figure 4.10: Santa Fe prediction with the Dirrec strategy. Solid line represents the real values and dashed one is the prediction using the k-NN with the LOO.

As a comparison of the overall performances of the Dirrec and the Direct strategies, the test errors of are shown in Figure 4.11. The strategies use different approximation methods, so the comparison is a bit tricky. But what can be said, is that the overall performance with the Dirrec strategy is clearly better than with the Direct strategy.

For the Poland Electricity Load the best method is the continuous LL with global k. With this time series, the same kind of effect exists than in the Santa Fe case; the GPLL performs well in the first timesteps, but then at timestep 8 the continuous LL begins to give better performance. The prediction of the best method is shown in Figure 4.12.

In Poland Electricity case, the same method gives the best overall performance with the Dirrec strategy as well as with the Recursive strategy. The test errors of both strategies are shown in Figure 4.13. In this case, the

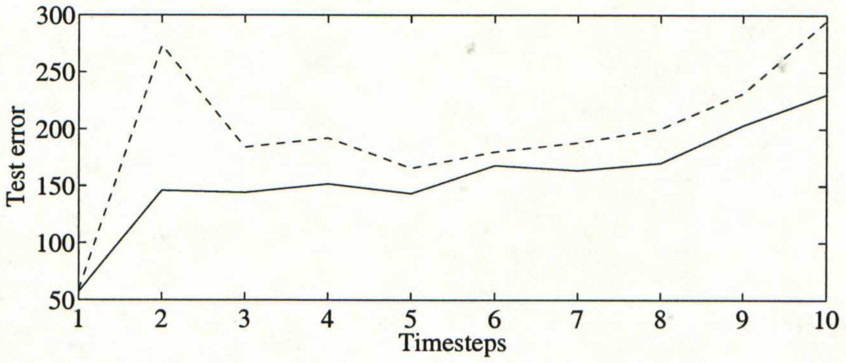

Figure 4.11: Santa Fe test errors of the Dirrec and the Direct strategy. The solid line represents the Dirrec and dashed line the Direct strategy. The Dirrec strategy uses the k-NN with the LOO and the Direct strategy uses the GPLL with global k.

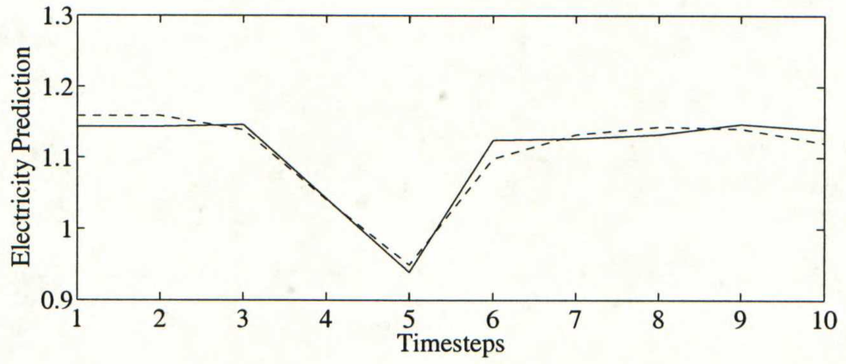

Figure 4.12: Poland Electricity Load prediction with the Dirrec strategy. Solid line represents the real values and dashed one is the prediction using the continuous LL with global k.

difference is not big, but still in overall performance the Dirrec strategy is better than the Recursive.

From these experiments it can be said that the Dirrec strategy outperforms both the Recursive and the Direct strategies in these time series. But the selection of the method must be done carefully or the performance is degraded.

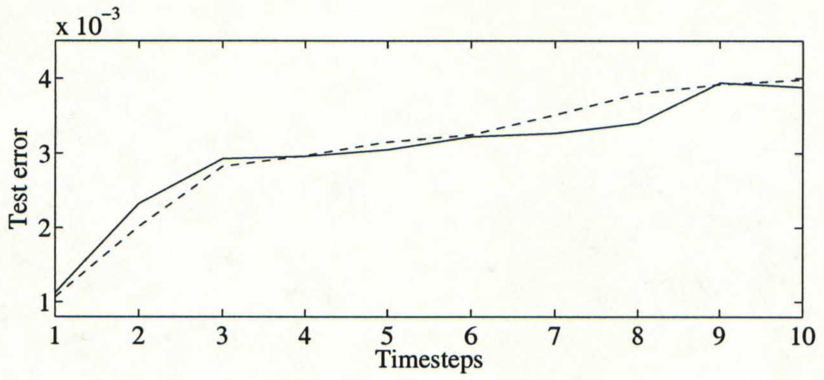

Figure 4.13: Poland Electricity Load test errors of the Dirrec and the Recursive strategy. The solid line is the errors of the Dirrec and dashed line the Recursive. Both strategies use the LL with global k.

4.6 Application: CATS Competition

The CATS Competition was arranged in 2004 as a special session of IJCNN, International Joint Conference on Neural Networks [27]. The data set consists of 5000 values from which 100 values are missing. The missing values are divided into 5 "holes", the first hole covering values from 981 to 999, second from 1981 to 1999 and so on until the data set ends with the last hole of 20 values. This means that we have 4 separate missing value subproblems and one ordinary 20-step ahead prediction problem.

As mentioned earlier, only one method was used in solving all the subproblems. Based on the experiments done with the 3 benchmarks, the GPLL method with global k is selected to be used for the CATS Competition data set.

All the available data, 4900 values are used as the learning set, shown in Figure 4.14 and a close-up to the second hole in Figure 4.15.

Because of the need for the 20 step ahead prediction and the calculation time limitations, the Direct prediction strategy is used. For the CATS data set, the maximum number of neighbors is set to 300 and the number of inputs to 15. The first order trend removal is used before the actual learning and it is reversed afterwards.

For the four missing value subproblems the following procedure is used: for each hole 20 step ahead prediction is done from both sides of the hole. Then the predictions are weighted according to the inverse of the prediction step and averaged.

The results of the predictions of each hole and the pure 20 step ahead prediction in the end of the data set are presented in Figures from 4.16 to 4.20.

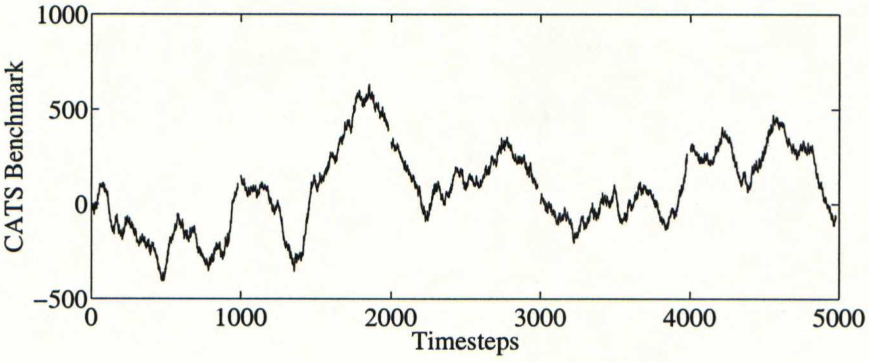

Figure 4.14: CATS Benchmark learning set.

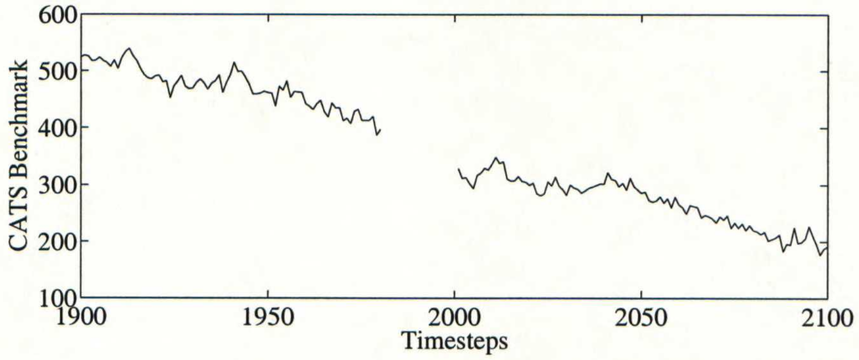

Figure 4.15: Close-up to the CATS Benchmark learning set, values from 1900 to 2100 with values missing from 1981 to 1999.

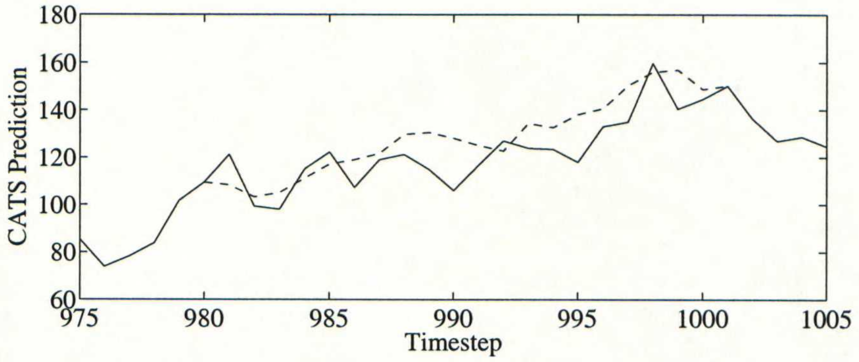

Figure 4.16: CATS Benchmark prediction, first hole. Solid line represents the real values and dashed one the prediction. The prediction MSE in this hole is 125.

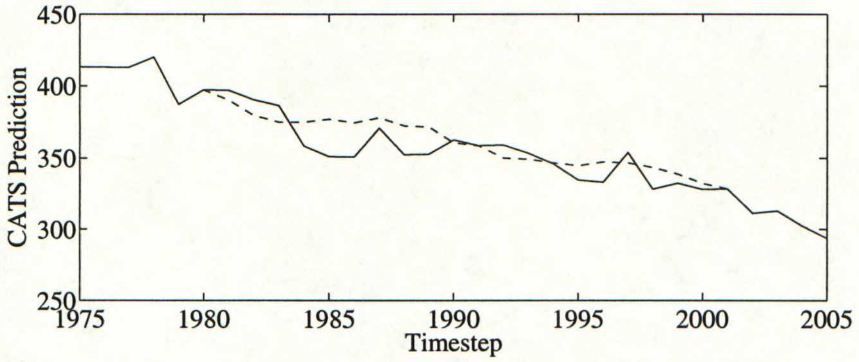

Figure 4.17: CATS Benchmark prediction, second hole. Solid line represents the real values and dashed one the prediction. The prediction MSE in this hole is 168.

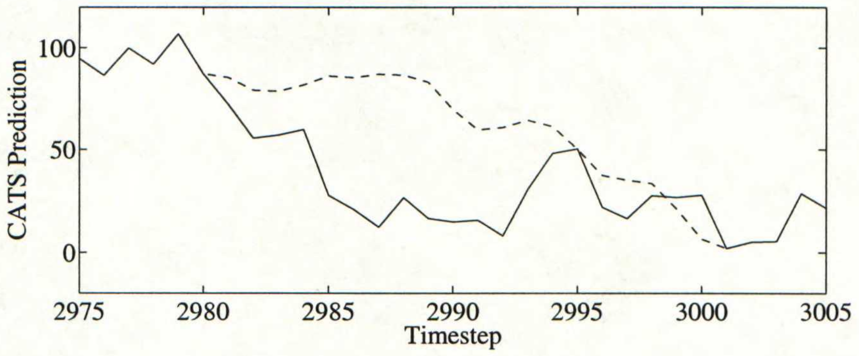

Figure 4.18: CATS Benchmark prediction, third hole. Solid line represents the real values and dashed one the prediction. The prediction MSE in this hole is 1650.

The first two holes are predicted very well by the GPLL, but the rest is not so good. Final result in the competition [27] would be 11^{th} place with overall MSE_1 value of 747. In order to enhance the performance, the Direct strategy could be replaced with the Dirrec and the initial number of inputs increased.

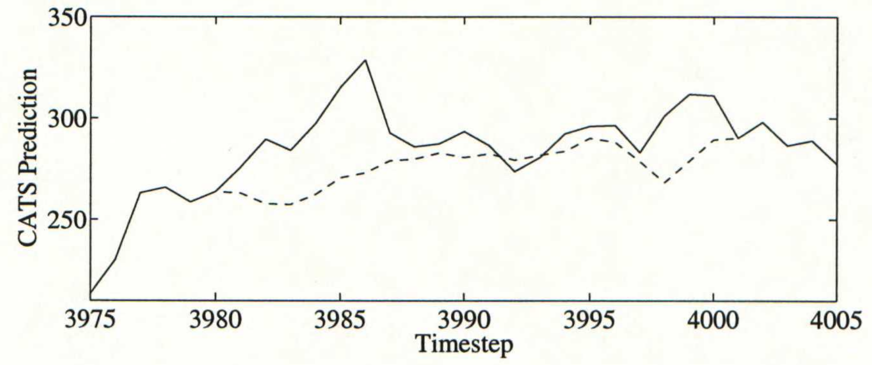

Figure 4.19: CATS Benchmark prediction, fourth hole. Solid line represents the real values and dashed one the prediction. The prediction MSE in this hole is 572.

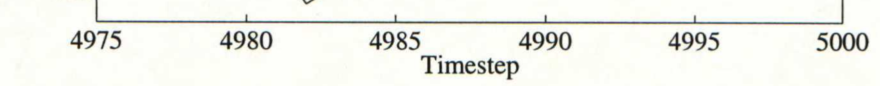

Figure 4.20: CATS Benchmark prediction, end part. Solid line represents the real values and dashed one the prediction. The prediction MSE in the end part is 1220.

Chapter 5

Conclusions

The Lazy Learning is a good method. It provides accurate approximations in a reasonable time. But with the original LL, the problem is the input selection. The GPLL fixes the problem and makes the approximations even better. In both methods, global k gives better performance than locally selected one. Maybe the number of local neighbors is too unstable and therefore gives not so accurate results.

Anyhow, with both improvements to the original LL, the input selection and the global k, it can be said that "it's not good to be Lazy". One must do some calculation and structure and input selection before the LL can be efficiently used as an approximator.

Despite the simplicity, the k-NN is quite accurate and reliable method. The simplicity allows wider search for model structures and different input sets. In many time series prediction problems it is very important to be able to use the past values very far from the current time point, and so k-NN can use the much needed information further than more complicated methods.

In general the Direct strategy is better than the Recursive. Even if each prediction step further to the future needs it's own model structure selection and parameter tuning phase, it is an acceptable cost in order to get better prediction performance.

But if the input selection method is good enough, the Dirrec strategy outperforms both, the Direct and the Recursive strategies. In a way, the Dirrec has captured the best aspects from both strategies. Hence, the Dirrec is a union of the Direct and the Recursive, DIR $\cup$ REC.

If the input selection used with the Dirrec is not powerful enough or the approximator cannot provide accurate enough approximations, it might be better to use the Direct strategy instead. Increase in the regressor size is of no use, if the increasing part is inaccurate. Therefore, it is quite surprising to see that the very simple k-NN model can give satisfying input selections and accurate enough approximations with the Dirrec strategy. It is also the fastest combination used with the Dirrec.

From the input tables in the appendix B it can be seen that the time space between some selected inputs and the timestep to be predicted stays constant. It means that there is some dependency at the same distance in time in the time series dealt here. That information could be used as prior knowledge of the time series, in example when initializing the input set for the Forward-Backward method.

Bibliography

[1] A. Weigend and N. Gershenfeld, *Times Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley Publishing Company, 1994.

[2] L. Ljung, *System identification theory for User.* Prentice-Hall, Englewood CliPs, NJ, 1987.

[3] A. Sorjamaa, A. Lendasse, D. Francois, and M. Verleysen, "Business plans classification with locally pruned lazy learning models," pp. 112–119, Connectionist Approaches in Economics and Management Sciences - ACSEQ, Lille, France, 18-19 November, 2004.

[4] A. Sorjamaa, A. Lendasse, and M. Verleysen, "Pruned lazy learning models for time series prediction," pp. 509–514, European Symposium on Artificial Neural Networks - ESANN, Bruges, Belgium, 27-29 April, 2005.

[5] A. Sorjamaa, N. Reyhani, and A. Lendasse, "Input and structure selection for k-nn approximator," in *Lecture Notes in Computer Science* (J. Cabestany, A. Prieto, and F. Sandoval, eds.), vol. 3512, (Berlin), pp. 985–991, Computational Intelligence and Bioinspired Systems: 8th International Workshop on Artificial Neural Networks - IWANN, Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005, Springer-Verlag GmbH, 2005.

[6] A. Sorjamaa, J. Hao, and A. Lendasse, "Mutual information and k-nearest neighbors approximator for time series predictions," in *Lecture Notes in Computer Science*, vol. 3697, pp. 553–558, International Conference on Artificial Neural Networks - ICANN, Warsaw, Poland, September 11-15, 2005.

[7] J. Hao, A. Sorjamaa, N. Reyhani, Y. Ji, and A. Lendasse, "Methodology for long-term prediction of time series." Submitted to Neurocomputing after the selection of the best papers of IWANN 2005.

[8] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[9] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[10] S. Haykin, *Neural Networks - A Comprehensive Foundation, 2nd edition.* Upper Saddle River, New Jersey 07458: Prentice Hall, 1999.

[11] Y. Ji, J. Hao, N. Reyhani, and A. Lendasse, "Direct and recursive prediction of time series using mutual information selections," in *Lecture Notes in Computer Science* (J. Cabestany, A. Prieto, and F. Sandoval, eds.), vol. 3512 / 2005, pp. 1010–1017, Computational Intelligence and Bioinspired Systems: 8th International Workshop on Artificial Neural Networks - IWANN, Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005, Springer-Verlag GmbH, July 2005. ISBN: 3-540-26208-3, ISSN: 0302-9743.

[12] A. Lendasse, V. Wertz, and M. Verleysen, *Model selection with cross-validations and bootstraps - Application to time series prediction with RBFN models*, pp. 573–580. No. 2714, Berlin: Springer-Verlag, 2003.

[13] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, 1995.

[14] G. C. Cawley and N. L. C. Talbot, "Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers.," *Pattern Recognition*, vol. 36, no. 11, pp. 2585–2592, 2003.

[15] T. Zhang, "A leave-one-out cross validation bound for kernel methods with applications in learning," in *COLT '01/EuroCOLT '01: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, (London, UK), pp. 427–443, Springer-Verlag, 2001.

[16] D. R. Wilson and T. R. Martinez, "Combining cross-validation and confidence to measure fitness," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, no. 163, 1999.

[17] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap.* Chapman et Hall, 1993.

[18] B. Efron and R. J. Tibshirani, "Improvements on cross-validation: The .632+ bootstrap method," *Journal of the American Statistical Association*, vol. 92, pp. 548–560, 1997.

[19] M. Cottrell, B. Girard, M. Mangeas, and C. Muller, "Neural modeling for time series: a statistical stepwise method for weight elimination," *IEEE Transaction on Neural Networks 6*, pp. 1355–1364, 1995.

[20] D. W. Aha, "Editorial of special issue on lazy learning," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 1–6, 1997.

[21] R. H. Myers, *Classical and Modern Regression with Applications, 2nd edition.* Pacific Grove, CA, USA: Duxbury, 1990.

[22] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation.* New York: Academic Press, 1977.

[23] M. Birattari, G. Bontempi, and H. Bersini, "Lazy learning meets the recursive least squares algorithm," in *Proceedings of the 1998 conference on Advances in neural information processing systems II*, (Cambridge, MA, USA), pp. 375–381, MIT Press, 1999.

[24] Santa Fe data website:
http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html.

[25] Darwin Sea Level Pressure data website:
http://ingrid.ldgo.columbia.edu/SOURCES/.Indices/.Darwin/.slp/.

[26] Time Series Prediction Group homepage:
http://www.cis.hut.fi/projects/tsp/?page=Timeseries.

[27] CATS Competition data website:
http://www.cis.hut.fi/~lendasse/competition/competition.html.

Appendix A

Test Errors

Method	Timestep 1	2	3	4	5	6	7	8	9	10	Mean
Linear	407,84	513,09	558,91	613,98	634,39	643,76	655,94	671,36	966,30	1035,8	670,14
Linear Pruning	410,22	520,89	560,72	**608,94**	**627,87**	**634,90**	**650,12**	**658,78**	956,17	1035,7	666,43
LL, Local k	56,04	241,74	5527,3	$3{,}3 \cdot 10^5$	$2{,}1 \cdot 10^7$	$1{,}4 \cdot 10^9$	$9 \cdot 10^{10}$	$6 \cdot 10^{12}$	$4 \cdot 10^{14}$	$2 \cdot 10^{16}$	$2 \cdot 10^{15}$
LL, Global k	**46,37**	**140,54**	**310,05**	781,95	4720,9	63048	$9{,}2 \cdot 10^5$	$1{,}3 \cdot 10^7$	$2{,}0 \cdot 10^8$	$2{,}9 \cdot 10^9$	$3{,}1 \cdot 10^8$
GPLL, Local k	46,57	153,95	514,55	12626	$7{,}0 \cdot 10^5$	$4{,}2 \cdot 10^7$	$2{,}5 \cdot 10^9$	$2 \cdot 10^{11}$	$9 \cdot 10^{12}$	$6 \cdot 10^{14}$	$6 \cdot 10^{13}$
GPLL, Global k	58,50	258,82	1626,3	19735	$1{,}3 \cdot 10^5$	$5{,}2 \cdot 10^6$	$3{,}2 \cdot 10^8$	$2 \cdot 10^{10}$	$1 \cdot 10^{12}$	$8 \cdot 10^{13}$	$8 \cdot 10^{12}$
LPLL, Local k	111,93	537,96	12936	$1{,}7 \cdot 10^5$	$1{,}0 \cdot 10^7$	$7{,}6 \cdot 10^8$	$6 \cdot 10^{10}$	$4 \cdot 10^{12}$	$3 \cdot 10^{14}$	$2 \cdot 10^{16}$	$2 \cdot 10^{15}$
k-NN Exhaustive 17	53,50	207,86	545,47	764,78	842,27	857,40	858,10	901,84	929,79	895,73	685,67
k-NN Selection 17	51,89	208,29	520,20	764,96	809,79	819,74	816,41	842,90	907,97	**886,76**	**662,89**
k-NN Selection	51,91	207,99	518,63	766,67	809,22	820,81	816,00	843,26	**906,89**	890,84	663,22

Table A.1: Test MSE values of the Santa Fe data set using the Recursive strategy. The smallest error of each timestep is marked with bold. All methods use maximum of 100 neighbors. The numbers after the method refer to the maximum number of inputs used, if it differs from the 30.

Method	Timestep 1	2	3	4	5	6	7	8	9	10	Mean
Linear	407,61	513,42	559,67	622,26	648,36	658,97	665,32	697,51	962,19	1033,0	676,83
Linear Pruning	409,96	544,58	550,87	605,17	621,64	643,31	652,91	666,53	951,55	1004,7	665,12
LL, Local k	58,18	189,87	223,37	232,66	236,28	260,46	266,11	309,46	403,43	339,27	251,91
LL, Global k	46,76	190,20	368,17	413,57	330,55	388,53	376,95	246,34	355,71	423,53	314,03
GPLL, Local k	**46,71**	234,06	256,75	236,41	225,04	234,17	222,51	223,16	276,37	320,45	227,56
GPLL, Global k	58,60	273,35	**183,88**	**192,02**	**165,38**	179,84	**187,86**	**200,35**	**231,22**	294,92	**196,74**
LPLL, Local k	111,93	206,10	280,76	236,15	263,70	**176,91**	223,65	326,93	393,00	422,45	264,16
k-NN Exhaustive 17	53,59	**99,50**	192,27	211,10	204,35	204,57	233,82	285,31	243,01	**268,78**	199,63
k-NN Selection 17	51,97	197,10	425,17	434,09	409,47	278,82	249,83	327,76	322,80	369,31	306,63
k-NN Selection	51,99	197,15	374,44	433,98	416,91	279,46	221,60	260,34	323,28	352,84	291,20

Table A.2: Test MSE values of the Santa Fe data set using the Direct strategy. The smallest error of each timestep is marked with bold. All methods use maximum of 100 neighbors. The numbers after the method refer to the maximum number of inputs used, if it differs from the 30.

Method	Timestep										
	1	2	3	4	5	6	7	8	9	10	Mean
Linear	1,083	1,387	1,746	2,141	2,317	2,567	2,628	2,608	2,562	2,486	2,153
Linear Pruning	1,085	1,389	1,751	2,152	2,325	2,582	2,643	2,628	2,578	2,502	2,163
LL, Local *k*	1,181	1,758	3,222	2,720	2,730	3,099	2,885	2,478	2,158	2,507	2,474
LL, Global *k*	0,996	**1,248**	**1,610**	**1,830**	**1,931**	**2,147**	2,243	2,281	2,348	2,367	**1,900**
GPLL, Local *k*	1,563	1,721	1,828	2,094	2,493	2,597	**2,029**	2,420	2,589	**1,979**	2,131
GPLL, Global *k*	**0,979**	1,311	1,679	1,942	2,048	2,298	2,331	2,354	2,405	2,294	1,964
LPLL, Local *k*	1,614	1,742	7,019	3,948	3,330	3,377	3,882	3,425	5,302	24,265	5,790
k-NN Exhaustive 17	1,375	1,706	2,041	2,350	2,453	2,332	2,302	2,072	1,993	2,014	2,064
k-NN Selection 17	1,330	1,641	1,910	2,236	2,313	2,342	2,241	**2,058**	**1,982**	1,995	2,005
k-NN Selection	1,175	1,444	1,661	2,006	2,239	2,258	2,322	2,343	2,328	2,307	2,008

Table A.3: Test MSE values of the Darwin data set using the Recursive strategy. The smallest error of each timestep is marked with bold. All LL methods use maximum of 500 neighbors except the LPLL, which uses only 100. Also the k-NN methods use maximum of 100 neighbors. The numbers after the method refer to the maximum number of inputs used, if it differs from the 30.

	Timestep										
Method	1	2	3	4	5	6	7	8	9	10	Mean
Linear	1,088	1,372	1,725	1,993	**2,083**	2,160	2,130	2,133	2,089	2,074	**1,885**
Linear Pruning	1,090	1,386	1,756	2,045	2,086	**2,157**	2,160	2,114	2,111	2,069	1,897
LL, Local k	1,186	4,171	1,944	2,399	3,567	3,440	3,825	3,040	2,317	3,915	2,980
LL, Global k	0,994	**1,217**	3,059	3,634	2,130	2,375	2,999	2,067	2,283	2,968	2,373
GPLL, Local k	1,413	2,557	1,807	2,877	2,824	4,220	3,321	2,798	2,564	3,028	2,741
GPLL, Global k	**0,983**	1,262	**1,669**	**1,922**	2,223	2,229	2,190	2,264	2,208	2,277	1,923
LPLL, Local k	1,614	5,879	5,574	3,925	3,138	4,524	4,392	9,400	3,341	3,947	4,573
k-NN Exhaustive 17	1,385	1,702	2,104	2,374	2,561	2,674	**2,070**	**1,891**	1,895	1,920	2,058
k-NN Selection 17	1,332	1,763	2,050	2,587	2,544	2,404	2,196	1,900	1,795	**1,682**	2,025
k-NN Selection	1,171	1,560	2,148	2,600	2,663	2,553	2,354	1,990	**1,741**	1,760	2,054

Table A.4: Test MSE values of the Darwin data set using the Direct strategy. The smallest error of each timestep is marked with bold. All LL methods use maximum of 500 neighbors except the LPLL, which uses only 100. Also the k-NN methods use maximum of 100 neighbors. The numbers after the method refer to the maximum number of inputs used, if it differs from the 30.

Method	Timestep 1	2	3	4	5	6	7	8	9	10	Mean
Linear	1,631	2,320	**2,725**	**2,966**	3,190	3,261	3,412	4,104	4,612	4,814	3,304
Linear Pruning	1,648	2,329	2,730	2,985	3,213	3,292	3,425	4,148	4,640	4,843	3,325
LL, Local k	1,396	2,785	4,117	5,068	5,395	6,424	7,027	7,830	9,508	8,184	5,773
LL, Global k	**1,067**	**2,015**	2,822	2,968	**3,152**	**3,246**	3,517	**3,799**	**3,919**	**3,984**	**3,049**
GPLL, Local k	1,744	3,306	4,653	5,425	5,443	4,989	19,976	21,970	10,976	31,713	11,019
GPLL, Global k	1,165	2,149	2,947	3,101	3,439	3,377	**3,342**	4,082	4,548	4,845	3,300
LPLL, Local k	2,418	4,856	7,127	7,570	7,571	7,165	9,812	12,374	9,842	18,175	8,691
k-NN Exhaustive	1,585	2,647	3,250	3,516	3,667	3,649	4,134	4,707	4,950	5,094	3,720
k-NN Selection	1,588	2,899	3,410	3,904	4,079	4,161	4,575	5,257	5,868	6,003	4,174

Table A.5: Test MSE values of the Poland Electricity Load data set using the Recursive strategy. All error values are multiplied by 1000 for convenience. The smallest error of each timestep is marked with bold. All methods use maximum of 200 neighbors and maximum of 14 inputs.

Method	Timestep										Mean
	1	2	3	4	5	6	7	8	9	10	
Linear	1,874	2,591	**2,578**	**2,831**	3,072	3,150	3,281	**3,750**	4,199	4,461	3,179
Linear Pruning	1,649	2,247	2,609	2,850	**3,068**	3,119	**3,267**	3,958	4,297	**4,447**	**3,151**
LL, Local k	1,396	2,821	5,126	3,943	5,381	4,045	4,048	5,853	4,931	6,096	4,364
LL, Global k	**1,083**	2,078	2,754	2,834	3,528	3,920	3,889	3,816	**4,008**	4,643	3,255
GPLL, Local k	1,744	2,460	5,276	6,750	4,152	4,160	3,831	6,186	6,957	6,497	4,801
GPLL, Global k	1,165	**1,863**	2,687	3,134	3,324	**2,996**	3,324	4,857	5,041	4,854	3,324
LPLL, Local k	2,418	5,534	5,648	4,958	7,261	6,830	9,021	10,162	12,297	13,912	7,804
k-NN Exhaustive	1,585	2,471	2,871	3,402	3,902	3,473	3,829	5,334	5,161	5,121	3,715
k-NN Selection	1,588	2,792	4,305	4,486	4,787	3,889	4,024	4,702	5,594	6,968	4,313

Table A.6: Test MSE values of the Poland Electricity Load data set using the Direct strategy. All error values are multiplied by 1000 for convenience. The smallest error of each timestep is marked with bold. All methods use maximum of 200 neighbors and maximum of 14 inputs.

Method	Timestep										Mean
	1	2	3	4	5	6	7	8	9	10	
Santa Fe											
LL, Global k	42,33	138,46	282,98	317,07	9303,1	1254,2	816,37	$1{,}4 \cdot 10^5$	6036,0	12679	17417
GPLL, Global k	30,53	**85,11**	189,68	223,30	291,52	270,30	238,19	358,18	240,50	685,27	261,26
k-NN with LOO	57,15	145,88	**144,06**	**151,65**	**143,28**	**167,78**	**163,45**	**170,02**	**203,23**	**230,24**	**157,67**
k-NN with LL	**30,53**	97,39	223,57	371,56	251,18	259,93	258,62	301,73	278,41	294,71	236,76
Electric Load											
LL, Global k	1,121	2,324	2,926	2,958	3,047	3,222	3,268	**3,404**	**3,942**	**3,885**	**3,010**
GPLL, Global k	**1,085**	**2,179**	2,915	**2,887**	**3,015**	**3,121**	**3,237**	4,137	4,198	4,701	3,147
k-NN with LOO	1,585	2,690	3,074	3,300	3,491	3,473	4,020	3,967	4,332	4,688	3,462
k-NN with LL	1,673	2,251	**2,803**	3,044	3,019	3,377	3,512	4,800	4,552	4,393	3,342

Table A.7: Test MSE values of the Santa Fe data set and the Poland Electricity Load data set using the Dirrec strategy. The smallest error of each timestep of each data set is marked with bold. Note that all error values of the Poland Electricity Load are multiplied by 1000 for convenience. In the case of Santa Fe, all methods use a maximum of 100 neighbors and a starting regressor length of 30, and in the case of Poland Electricity Load, all methods use a maximum of 200 neighbors and a starting regressor length of 14.

Appendix B

Selected inputs

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
0	A B C	A B C	A B C	B C	A	B	B	A C	A B C	A B C
1	A B C	B C	B		B	B	A C	A B C	B C	A B
2		A	B C	B	A B	A C	A B C	A C		
3		B	B	B C	A	A B C	A C		A B C	A B C
4			A B	A C	B C	A C		A	B	
5			A	A	A C		B	C	A C	A C
6		C		A C		B	A C			A C
7	A	A			B		B		A	
8				B	C	B	B		A C	
9					B	B				
10		B	A	B C	B C			B	B	B
11	B						B			
12				A		B		B		
13					B					
14				B						

Table B.1: Santa Fe prediction selected inputs from $t-0$ to $t-14$. A denotes the GPLL with global k, B denotes the k-NN using the Exhaustive Search and C denotes the GPLL with local k. All methods use the Direct strategy.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
15			B				C	B	C	
16								B	B	B
17										
18								C		
19										
20										
21										
22				C		C	C	A		C
23			A					A		
24								C	A	
25							C			
26					C		A	A		
27					A	A C	A C			C
28	A		A		C	A C	C			
29				A	A		C		C	A C

Table B.2: Santa Fe prediction selected inputs from $t - 15$ to $t - 29$. A denotes the GPLL with global k, B denotes the k-NN using the Exhaustive Search and C denotes the GPLL with local k. All methods use the Direct strategy.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
0	A B C	A B	A B	A B	A B	A B	A B	A B	A	A
1	A B C	A B	A B	A B	A B	A B	A	A	A	A B
2	A B C	A	A	A	A	A B	A	A	A B	A B
3	A B C	A	A	A B	A B	A	A	A B	A B	A B
4	A C	A	A	A B	A	A	A B	A B	A	A
5	A B C	A B	A B	A B	A B	A	A B	A B	A	A
6	A B C	A B	A	A	A	A B	A B	A	A	A B
7	A B C	A	A	A B	A B	A B	A	A	A B	A B
8	A B C	A B	A B	A B	A B	A	A B	A B	A	A
9	A C	A B	A B	A	A	A	A B	A	A	A B
10	A B C	A B	A B	A	A	A B	A	A B	A B	A B
11	A B C	A	A B	A B	A B	A B	A B	A B	A B	A
12	A C	A B	A B	A B	A B	A B	A B	A B	A	A
13	A B C	A B	A B	A B	A B	A B	A B	A	A	A B
14	A B C	A B	A B	A	A B	A B	A	A	A	A

Table B.3: Darwin prediction selected inputs from $t - 0$ to $t - 14$. A denotes the continuous Linear model, B denotes the Linear model with pruning and C denotes the LL with global k. First two methods use the Direct strategy and the LL uses the Recursive strategy. Therefore, the LL has only one set of selected inputs.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
15	A B C	A	A	A B	A B	A B	A	A B	A B	A
16	A C	A	A B	A B	A	A	A	A B	A B	A B
17	A C	A B	A B	A	A	A B	A B	A B	A	A
18	A B C	A B	A	A	A	A B	A B	A B	A	A B
19	A C	A	A B	A B	A B	A B	A B	A B	A	A B
20	A B C	A B	A B	A B	A B	A	A	A B	A B	A B
21	A B C	A B	A B	A B	A	A	A	A B	A	A
22	A B C	A B	A B	A	A	A	A B	A	A B	A B
23	A B C	A	A	A B	A	A B	A	A B	A B	A
24	A B C	A	A B	A	A B	A B	A B	A B	A	A
25	A B C	A B	A	A B	A B	A B	A B	A	A	A
26	A B C	A	A B	A	A B	A	A	A	A	A B
27	A C	A B		A B	A	A B	A	A B	A B	B
28	A B C			A	A B	A	A	A B		
29				A B	A B	A B	A B			B

Table B.4: Darwin prediction selected inputs from $t-15$ to $t-29$. A denotes the continuous Linear model, B denotes the Linear model with pruning and C denotes the LL with global k. First two methods use the Direct strategy and the LL uses the Recursive strategy. Therefore, the LL has only one set of selected inputs.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
0	A B C	B C	B C	B C	B C	B C	B C	B C	B C	B C
1	A B C	C	C	C	C	B C	C	C	C	C
2	A B C	C	C	B C	B C	C	C	C	B C	C
3	A C	C	B C	B C	C	C	C	B C	C	C
4	A C	B C	B C	C	C	C	B C	C	C	B C
5	A B C	B C	C	C	C	C	C	C	B C	C
6	A B C	C	B C	B C	B C	C	C	B C	B C	B C
7	A B C	B C	B C	B C	B C	B C	B C	B C	B C	B C
8	A C	C	B C	C	C	B C		C	C	C
9	C	C	C	B C	B C			C	B C	C
10	C	C	B C	B C				B C	C	C
11	B C	B C	B C				B	B C	C	B C
12	B C	B C				B		C	B C	B C
13	B C	C			B		B	B C	B C	B C

Table B.5: Poland Electricity Load prediction selected inputs. A denotes the LL with global k, B denotes the Linear model with pruning and C denotes the Linear model with continuous input. The LL uses the Recursive strategy and therefore, it has only one selected set of inputs. The Linear models use the Direct strategy.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
$-\hat{9}$										A B C
$-\hat{8}$									A B C	A C
$-\hat{7}$								A C	A C	A B
$-\hat{6}$							A C	A C	A B	C
$-\hat{5}$						A C	A C	A B C	B	A B
$-\hat{4}$					A C	A C	A B	A	B	B C
$-\hat{3}$				A B C	A	A B	A	A B	B C	A B C
$-\hat{2}$			A B C	A B	A B	A	B	B	A B C	A
$-\hat{1}$		A B C	A B C	A B C	A C		B	A B	A C	A C
0	A B C	A B C	A	A B	A B C	A B	A B C	A B C		A
1	A B C	A B	A B	A	A B	A B	A B C			A C
2		A	B	A	A B	A B C				A C
3			A	A	A B C				C	B
4			A C	A B C	C				B	B
5			A C		A		C	A B C	B	
6		A B C	A C	A C			A B		C	C
7			A			A B				A
8		A			A B					
9		B		A					C	
10		A	A		C					

Table B.6: Santa Fe selected inputs from $t+9$ to $t-10$ of the Dirrec strategy. A denotes the k-NN with the LOO combination, B denotes the k-NN with the LL combination and C denotes the GPLL with global k. The approximated inputs are denoted by the hat symbol.

t - {...}	Timestep									
	1	2	3	4	5	6	7	8	9	10
11	A B C	A	A							C
12							B			C
13						B				
14								B		
15						A	B			
16				A	A B C	B			A B	
17			A		B			A B		B C
18		A					A B C		A	B
19						A B		A		
20					A B		A			
21				A		A B	C			
22			A		A B					A
23			C	B					A	A C
24							C	A		
25							A C			B
26						A			C	
27					A					
28			C	A				B		C
29			A	C	C				C	

Table B.7: Santa Fe selected inputs from $t-11$ to $t-29$ of the Dirrec strategy. A denotes the k-NN with the LOO combination, B denotes the k-NN with the LL combination and C denotes the GPLL with global k.

	Timestep									
t - {...}	1	2	3	4	5	6	7	8	9	10
$-\hat{9}$										A B C
$-\hat{8}$									A B C	A
$-\hat{7}$								A B C	A B	A C
$-\hat{6}$							A B C	A C	A C	A
$-\hat{5}$						A C	A	A B	A	A C
$-\hat{4}$					A B C	A B	A	A B	A B C	A C
$-\hat{3}$				A B C	A B	A B	A B	A C	A B C	A B C
$-\hat{2}$			A C	A C	A B	A B	A C	A B C	A B C	A C
$-\hat{1}$		A B C	A B C	A B C	A B	A C	A C	A B C	A B C	A
0	A B C	A B C	A B C	A B	A B C	A B C	A B C	A B C	A C	A
1	A B	A B	A B	A C	A B C	A B C	A	A B C	A	A
2	A B	A B C	A B C	A B C	A B C	A	A	A B	A	A
3	A	A B	A B C	A B C	A	A	A	A	A B	A
4	A B	A C	A B C	A	A B		A	B		C
5	A C	A B C	A	A B	A		A	B	B C	
6	A B C	A B C	A	A B	A B	B	A B	B C		
7	A B C	A	A B	A	A		A B C		B	
8	A		A	A	A C	B C	A			
9			A	A C	A B C		A B	B		
10			A C	A B C	A	B	A			C
11		C	A B C	A	A B		A		C	B C
12		B C	A	A C	A		C	C	C	
13	B C	B	A B	A B	A B	B C	B C	B C		

Table B.8: Poland Electricity Load selected inputs of the Dirrec strategy. A denotes the LL with global k, B denotes the GPLL with global k and C denotes the k-NN with the LL combination. The approximated inputs are denoted by the hat symbol.