SELF-LEARNING PREDICTION SYSTEM FOR OPTIMISATION OF WORKLOAD MANAGEMENT IN A MAINFRAME OPERATING SYSTEM

Michael Bensch[1], Dominik Brugger[1], Wolfgang Rosenstiel[1], Martin Bogdan[1,2], Wilhelm Spruth[1,2]
[1]*Department of Computer Engineering, Tübingen University, Sand 13, Tübingen, Germany*
[2]*Department of Computer Engineering, Leipzig University, Johannisgasse 26, Leipzig, Germany*

Peter Baeuerle
IBM Germany Development Lab, Schönaicher Str. 220, 71032 Böblingen, Germany

Keywords: Workload management, time series prediction, neural networks, feature selection.

Abstract: We present a framework for extraction and prediction of online workload data from a workload manager of a mainframe operating system. To boost overall system performance, the prediction will be incorporated into the workload manager to take preventive action before a bottleneck develops. Model and feature selection automatically create a prediction model based on given training data, thereby keeping the system flexible. We tailor data extraction, preprocessing and training to this specific task, keeping in mind the non-stationarity of business processes. Using error measures suited to our task, we show that our approach is promising. To conclude, we discuss our first results and give an outlook on future work.

1 INTRODUCTION

In a large system environment in mainframe computing like the IBM z-Series many different workloads of different types compete for the available resources. The types of workload cover things like online transaction processing, database queries or batch jobs as well as online timesharing users. The resources that are needed by these workloads are hard resources like CPU capacities, main memory or I/O channels as well as soft resources like available server processes that serve transactions (Vaupel et al., 2004).

Every installation wants to make the best use of its resources and maintain the highest possible throughput. To make this possible the Workload Manager (WLM) was introduced into the z/OS operating system (Aman et al., 1997). With the Workload Manager, one defines performance goals and assigns a business importance to each goal. The user defines the goals for work in business terms, and the system decides how many resources, such as CPU and storage, should be given to it in order to meet the predefined goal. The Workload Manager will constantly monitor the system and adapt processing to meet the goals.

Due to the growth and rapid change of the workload requirements in today's information processing, the challenge for workload management is to assign the required resource to the correct workload exactly when it is needed – or even before it is needed – to avoid any delays. This is especially important as workload management nowadays not only assigns available resources, but increasingly provides additional resources when they are needed. Any over- or underprovisioning is a direct cost factor.

Thus, a load prediction system that adapts itself to each customer's specific behaviour is required. Based on such load predictions the z/OS operating system could provide and assign the right resources to the important workload right in time instead of waiting until they are needed or even longer (Bigus et al., 2000) to improve performance and throughput while optimising resources usage and minimising costs.

This paper describes the project and the results achieved regarding the prediction quality of the self-

Bensch M., Brugger D., Rosenstiel W., Bogdan M., Spruth W. and Baeuerle P. (2007).
SELF-LEARNING PREDICTION SYSTEM FOR OPTIMISATION OF WORKLOAD MANAGEMENT IN A MAINFRAME OPERATING SYSTEM.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - AIDSS*, pages 212-218
DOI: 10.5220/0002392102120218

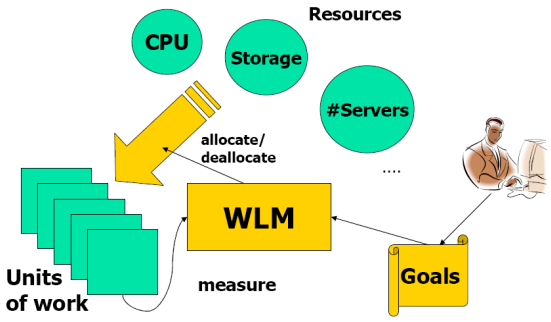

Figure 1: z/OS Workload Manager.

learning system. One has to take into account that the process being modelled is non-stationary. Future investigations will consider questions of re-learning when the customer's specific behaviour is changing, and how far explicit knowledge can be incorporated to improve the prediction quality.

The paper is organised as follows: Section 2 introduces the Workload Manager responsible for the resource allocation which will be influenced by the load prediction. Section 3 explains the data extraction and training process we use. Results are presented in Section 4, followed by a summary in Section 5.

2 WORKLOAD MANAGER

The z/OS Workload Manager (WLM) is a functionality of the base control program of the z/OS operating system for IBM Mainframes. Its basic functions are dynamic allocation and re-allocation of system resources to the different workloads running on clusters of z/OS driven systems, called Sysplex.

Resources could be dispatching priorities of CPU access, real storage, software servers of transactions and so forth. They are permanently allocated and re-allocated between the workloads based on a policy of business goals that was defined by the customer. Units of work could be things like batch jobs, online

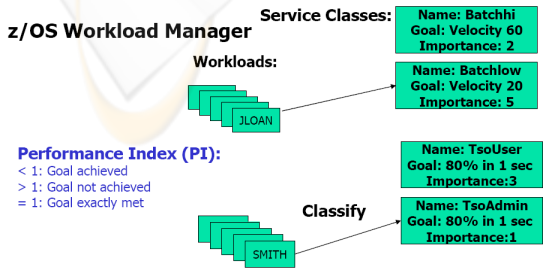

Figure 3: Classification of workloads into service classes.

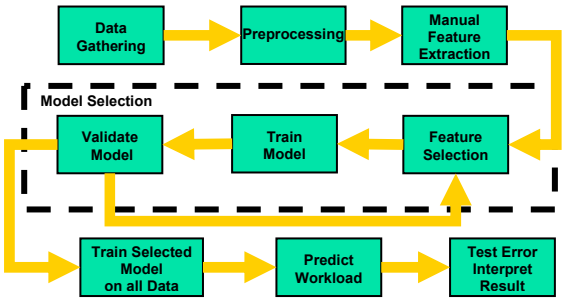

Figure 2: Overview of the training and prediction method.

transactions like CICS, DB2 or web-transactions, or interactive TSO work (see Figure 1).

The goals are defined in different service classes. Each incoming workload is classified into a service class, based on customer defined classification rules. An example is shown in Figure 2. Each service class has an importance between 1 (very important) and 5 (very unimportant) or it is discretionary.

The WLM tries to fulfil each service class's goal by constantly allocating/reallocating resources to the workloads. More important goals are preferred over less important ones, when they compete for resources. The central measurement parameter of goal achievement is the so-called Performance Index (PI) for each service class.

A PI of 1 means, that the goal of this service class is exactly met. If the goal is over achieved, the PI becomes less than 1. A PI > 1 indicates that the goal is not achieved.

3 MODUS OPERANDI

An overview of our data processing, feature selection, training, and prediction method is given in Figure 3. Detailed explanations of these methods can be found in the following subsections.

3.1 Data

The data gathering and feature extraction steps are specific to z/OS. The preprocessing methods are universal and applicable to other domains as well.

3.1.1 Gathering

The prediction of the PI of a selected service class must be based on those input data, which have the biggest influence on the achievement of the predefined goal of that service class. One important goal of this project was to find out, which features of

the data are the most relevant for the prediction quality.

Some of the candidates for those data features are:

- The historical time series of the PI that we want to predict itself.
- The time series of the PIs of other service classes.
- % CPU usage (% of total time where CPU was active)
- % workflow (% execution speed)
- % delay (% of time where processes waited for resources)
- Number of (active) users
- Unreferenced interval count (UIC, measure of memory contention)
- Transaction ended rate
- Number of CPUs

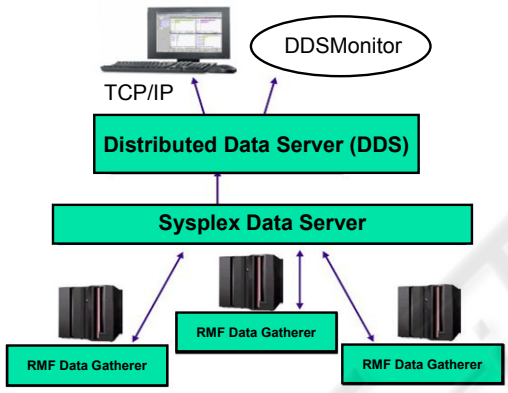

Figure 4: Resource Measurement Facility (RMF).

Those data, the so called SMF Data, were gathered on repeating intervals by using a standard add-on product for z/OS, RMF (Resource Measurement Facility). RMF is able to gather data from each system of a Sysplex and consolidate the data Sysplex-wide into one data pool. It can make those consolidated data available through a TCP/IP interface by a component called RMF DDS (Distributed Data Server).

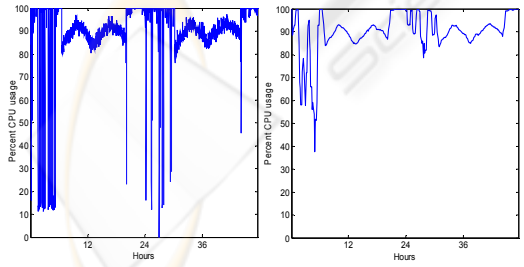

Figure 5: Original "% CPU usage" time series (left) and after downsampling and applying moving average (right).

In order to make the data available to the prediction system we have developed a JAVA based program, DDSMonitor, that takes the data from DDS and does some preprocessing and feature extraction. The architecture of the data gathering system can be seen in Figure 4.

3.1.2 Preprocessing

The data preprocessing has three major objectives:

- Data reduction
- Data smoothing
- Data scaling

Data reduction and smoothing were done by average downsampling and calculating a moving average.

Average downsampling means that the data points of n intervals are replaced by one data point, which represents the average of n intervals. This reduces the data to 1/n. As a trade-off between data reduction and maintaining enough data precision a value of n=5 was obtained empirically and used.

Calculating a moving average means that each data point is replaced by the average of itself plus the next m data samples. This smoothens the data series curves and reduces coincidental deviations from the main slope. A value of m=5 was used.

Figure 5 shows the effect of applying down-sampling and moving average to the original data for the "% CPU usage" time series.

To obtain a limited range of data values as input to the prediction algorithm, a hyperbolic tangent function is used to scale the data *d* (see Equation 1).

$$scaled = 2 \cdot \tanh(0.5 \cdot (d - 1.4)) \qquad (1)$$

3.1.3 Manual Feature Extraction

After the preprocessing step, we use our domain knowledge to manually discard some of the features which are known to be irrelevant for the PI prediction. This step speeds up the model selection phase, which includes an automated feature selection step (described in the following subsection).

3.2 Feature Selection

To reduce the number of features of the training data (which amounts to reducing the complexity of the problem), we employ Recursive Feature Elimination (Guyon et al., 2002), which is based on Support Vector methods. We have already employed feature

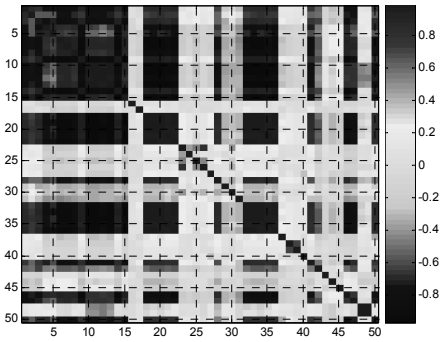

Figure 6: Cross-correlation of features numbered 1 to 50. A dark block indicates high correlation of the corresponding feature on the y-axis with another feature on the x-axis. Thus, the diagonal shows a feature's correlation with itself.

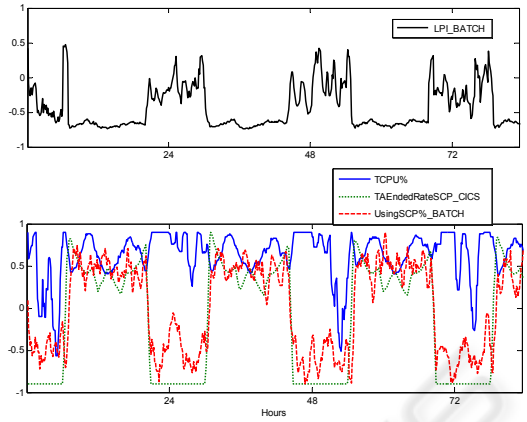

Figure 7: Performance index of one week (top) with 3 important features selected for the prediction (bottom).

selection successfully for quality control problems (Bensch et al., 2005). Since every load configuration is individual depending on the customer, this step should be repeated whenever load characteristics have changed significantly. The selected features (system parameters) are considered to be relevant for predicting the PI.

As can be seen in Figure 6, many feature types are highly correlated over all service classes (dark bands) and thus do not contribute any additional information to the prediction. Ideally, the feature selection algorithm will retain only one feature of such a highly correlated set.

Investigations showed that it suffices to merely take into account the features belonging to the same service class as the PI to be predicted (see Figure 11). An example of the three most important features to predict the PI (except the PI itself) which were selected by Recursive Feature Elimination is shown in Figure 7.

3.3 Self-learning Models

To predict the PI time series, various prediction models are compared. Momentarily two artificial neural network (ANN) types are used, a feedforward topology and the FlexNet. In addition, Support Vector Regression (SVR) is employed. These prediction methods are introduced in the following subsections.

3.3.1 Feedforward Network

We use a standard multi-layer feedforward network (number of layers and neurons optimised by model selection) with a backpropagation learning method (Rumelhart et al., 1986). A regularised error measure is used to prevent overfitting, displayed in Equation 2.

$$msereg = \gamma \cdot mse + (1-\gamma) \cdot msw \qquad (2)$$

mse refers to the typically used mean squared error of prediction to true value during training, *msw* is the mean of the sum of squares of the network weights and biases and γ is the regularisation parameter. Our model selection generally found values for γ in the range $0.8 \leq \gamma < 1$. To speed up convergence, we use conjugate gradient descent with Polak-Ribière updates of the weights (Polak and Ribière, 1969). Figure 8 shows a typical configuration of the feedforward network used.

3.3.2 FlexNet Network

FlexNet is a flexible, easy to use neural network construction and training algorithm. Starting with input and output neuron layers only, the network structure is incrementally defined during the training process. An example can be seen in Figure 9. FlexNet determines the best suited position/layers for competing groups of candidate neurons in the current network before adding them. As this allows

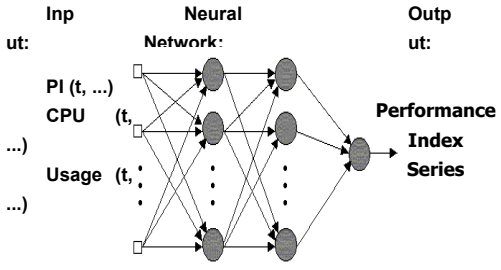

Figure 8: Feedforward network.

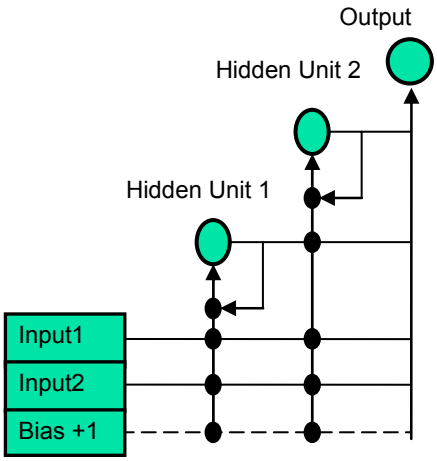

Figure 9: FlexNet topology.

for new neurons being added to both, new and existing layers, the created networks are not necessarily as deep and narrow as networks constructed by Cascade Correlation. FlexNet creates networks with as many layers and hidden units as needed to solve a given problem (Mohraz and Protzel, 1996). The FlexNet allows us to skip some of our model selection steps (see Subsection 3.4.1).

3.3.3 Support Vector Regression

A standard ν-SVR algorithm was compared to the ANN approaches. First results did not reach the prediction accuracy of the ANN methods, as the regression did not predict the periods of high fluctuation of the PI. The reason for this will be the object of further investigation.

3.4 Training Process

Our training process consists of two phases: A model selection phase, in which we determine the prediction model best suited to the data available, and a final training phase in which the prediction method is trained for application to a future PI time series. Note that due to the non-stationarity of the underlying business processes, model selection and final training have to be repeated at regular intervals. The following two subsections explain the training phases.

3.4.1 Model selection

Model selection is a vital step for the accuracy of the later prediction. A review is given by (Kearns et al., 1997). We estimate the prediction error of various prediction models for yet unknown data by a

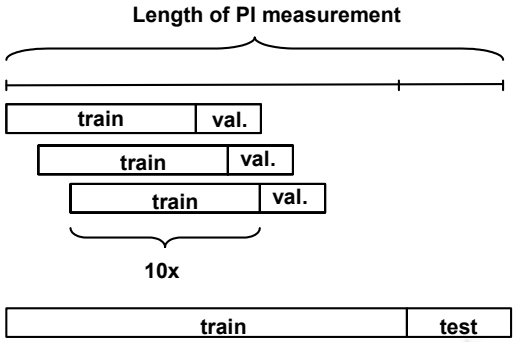

Figure 10: Sliding window validation.

modified cross-validation. This method keeps the training data in chronological order and is called sliding window validation (see Figure 10). The models we tested were combinations of hyper-parameters for the prediction methods (number of input/hidden neurons and γ for FF-net, ν and C for SVR), and the size of the input data window. The FlexNet does not require model selection for the number of input/hidden neurons since it develops its topology automatically.

The model with the lowest validation error is used for the final training.

3.4.2 Final Training

The final training phase consists of training the model on all available data that were collected from the system. Typical load characteristics are repeated on a daily basis. However, their details differ by some amount from day to day. The first four weekdays were used as a training set to predict the fifth weekday, which thus was the test set. For online operation we would use all five days as training set.

3.5 Prediction Method

Our aim was to predict system workload over a time of 7 hours, i.e. 50 samples (after preprocessing, one sample holds data of 500 seconds). The system was trained with data vectors of length $m = w \times f$, whereby w is the length of the input window, typically 20-40 samples, and f is the number of features (up to 5).

Results are evaluated with two error measures: The number of predicted samples outside a predefined error band (Prediction Error or PE, defined in accordance with WLM experts), and the mean average percentage error (MAPE) commonly used for evaluating electricity load predictions (Ortiz-Arroyo et al., 2005).

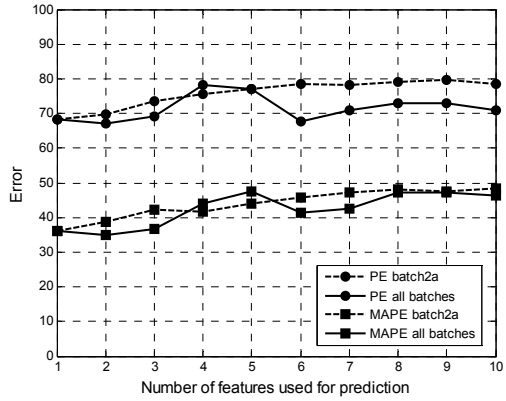

Figure 11: PE error (round markers) and MAPE error (square markers) in percent for various numbers of features shown on x-axis. The use of predicted batch2a features only (dotted lines) is compared to the use of features from all batches (solid lines).

4 RESULTS

We present results for two data sets. Dataset A is a synthetic simulation of a 3-week load scenario and dataset B is a real-time measurement of system parameters during a realistic load scenario. All results were obtained with smoothed data as explained in Subsection 3.1.2.

We first examined whether the prediction accuracy stays constant when only features from the batch to be predicted (batch2a in this example) are used. Apart from the fact that the use of more than 1-2 additional features actually decreased the prediction accuracy, we could ascertain that there is only a minimal loss when limiting the features to a particular batch, as can be seen in Figure 11.

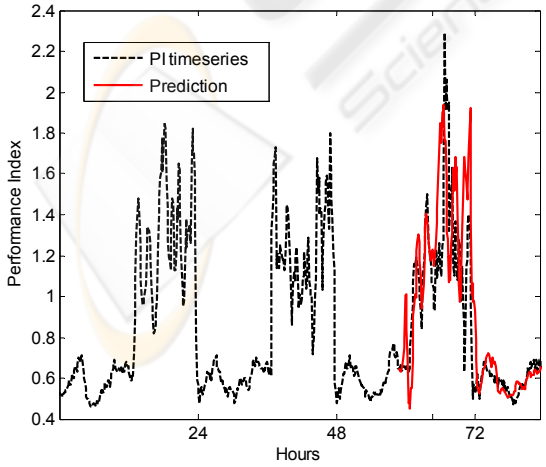

Figure 12: Performance Index time series over three days (dashed line) with prediction of last day (solid line).

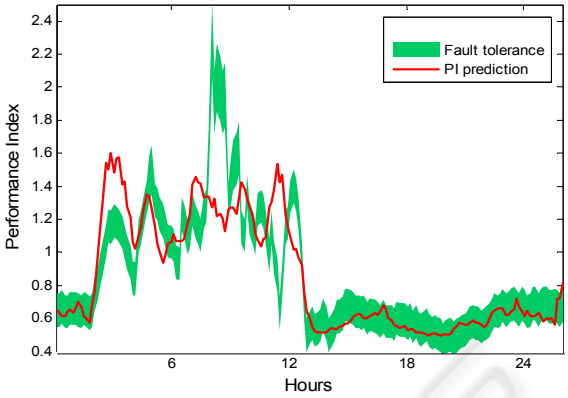

Figure 13: 50-sample prediction (corresponds to 7 hours) for dataset B shown over the course of one day, with a fault tolerance band.

The FlexNet returns better results on the synthetic data, whereas the FFN achieved improvement on the real data. Figure 12 shows the test result with a FFN trained on the first 4 days, using a 40-sample time lag, 20 hidden neurons, $\gamma = 0.7$ and a prediction over 50 samples into the future. Figure 13 zooms in on the last day to display the prediction and the fault tolerance band we use to evaluate the result.

Regarding the PE and MAPE, the best results achieved with a single PI as training input and with additional features for a 7-hour prediction (50 samples) are displayed in Table 1.

Table 1: Results for 50-sample prediction.

Features	Data	PE	MAPE	Method
PI only	A	28	170	FlexNet
	B	52	20.7	FFN
Many features	A	54	57.9	FlexNet
	B	34	12.6	FFN

Table 2 shows the same results for an 8-minute (single-sample) prediction. As expected, the results for single-sample prediction are better than for the 50-sample prediction. Note that prediction over 50 samples is generally considered to be a difficult problem. In general, employing feature selection to select the best three features and adding these to the PI, improved prediction results.

Table 2: Results for single-sample prediction.

Features	Data	PE	MAPE	Method
PI only	A	2.5	190	FlexNet
	B	13	6	FFN
Many features	A	5.6	28	FlexNet
	B	14.4	6.7	FlexNet

5 CONCLUSION AND FURTHER WORK

Early results seem promising, as the system could predict the tendencies of the workload behaviour as it changed over time. However, we will shorten the prediction horizon to gain higher prediction accuracy in future. A prediction horizon of 2-4 hours, rather than the 7 hours used so far, is sufficient for the future applications that are planned to exploit that prediction system. We also plan to investigate the inclusion of calendaring data into the prediction method, to include prior knowledge about load peaks on special days of the month.

The results of this project show that it is possible to predict the PI, as a relevant performance indicator, of a complex mainframe system cluster, like a z/OS Sysplex. This enables us to develop several functionalities that do resource assignment or resource provisioning to workloads right in time. This can avoid resource contention, like CPU or memory usage, significantly. Especially in big mainframe environments with very large numbers of competing workloads this can improve the throughput and optimal resource usage significantly and thus optimise data processing costs.

Further work in this area is to analyse and develop an automatic relearning environment for the prediction of non-stationary processes. This way, the prediction system will be able to permanently improve its prediction quality for each customer by learning more and more about its particular environment and workload behaviour. As an additional benefit, relearning adjusts the prediction to changes in typical customer workloads, e.g. when business related changes take effect.

We expect further improvements, when additional explicit knowledge is incorporated, e.g. changes or higher workload at special days of the month or upcoming events.

No other results in the field of operating system workload management prediction are known to us, hindering the comparison of our early results with others.

ACKNOWLEDGEMENTS

The authors thank Clemens Gebhard and Sarah Kleeberg for their participation in this project.

REFERENCES

Aman, J., Eilert, C. K., Emmes, D., Yocom, P., Dillenberger, D., 1997. Adaptive algorithms for managing a distributed data processing workload. In *IBM Systems Journal*, 36(2): 242-283.

Bensch, M., Schröder, M., Bogdan, M., Rosenstiel, W., 2005. Feature Selection for High-Dimensional Industrial Data. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN)*, Bruges.

Bigus, J. P., Hellerstein, J. L., Jayram, T. S., Squillante, M. S., 2000. AutoTune: A Generic Agent for Automated Performance Tuning. In *Practical Application of Intelligent Agents and Multi Agent Technology*.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 46(1): 389-422.

Kearns, M., Mansour, Y., Ng, A. Y., Ron, D., 1997. An Experimental and Theoretical Comparison of Model Selection Methods. *Machine Learning*, 27(1): 7–50.

Mohraz, K., Protzel, P., 1996. FlexNet: A Flexible Neural Network Construction Algorithm. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN)*, Bruges.

Ortiz-Arroyo, D., Skov, M. K., Huynh, Q., 2005. Accurate Electricity Load Forecasting with Artificial Neural Networks. In *Computational Intelligence for Modelling, Control and Automation*, 1: 94-99.

Polak, E., Ribiere, G., 1969. Note sur la Convergence de Methodes de Directions Conjugees. *Revue Francaise d'Informatique et de Recherche Operationnelle*, 3: 35-43.

Rumelhart, D., Hinton, G., Williams, R., 1986. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, pages 318-362. MIT Press.

Vaupel, R., Teuffel, M., 2004. Das Betriebssystem z/OS und die zSeries – Die Darstellung eines modernen Großrechnersystems. ISBN 3-486-27528-3, *Oldenbourg Wissenschaftsverlag,* Germany.