IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 1, January 2015.
www.ijiset.com
ISSN 2348 – 7968

Designing Dynamic Neural Network for Non-Linear System Identification

Chandradeo Prasad

Assistant Professor, Department of CSE, RGIT,Koderma

Abstract : System identification deals with many subtleties coming up when designing, conducting and interpreting results from such an experiment. System identification usually means identification of dynamic systems, so when dealing with neural architectures the emphasis will be on dynamic neuralnetworks.The purpose of this paper is to analyse and make calculations on the collection of ideas, insights, theorems, intuitions, language and practical experiences which constitute the art of system identification.

Abbreviations : AIC = Akaike Information Criteria, NN = Artificial Neural Networks, BPN = Back Propagation Network, BPTT = Back Propagation Through Time, CMAC = Cerebellar Model Arithmetic Computer , EIV = Errors In Variables, FIR = Finite Impulse Response, MLP = Multilayer Perceptron, MOE = Mixture Of Experts, NFIR = Non-linear Finite Impulse Response, NIC = Network Information Criterion, PCA = Principal Component Analysis, RBF = Radial Basis Function, RTRL = Real Time Recurrent Learning, SVMs = Support Vector Machines.

1.1 INDRODUCTION

In System Identification we are generally dealing withmathematicalmodel as opposed tophysicalmodels. With System Identification we use dedicated experiments to find a compact and accurate mathematical model of a dynamic system. It is frequently applied to optimise a controller for that system with knowledge from a sufficiently accurate description of the process. In many practical cases we have only a limited model of the dynamic behaviour of a complex process. In other cases the important parameters are only approximately known. In those cases System Identification can assist to obtain the required dynamic model.An important step in the identification procedure is the estimation of the parameters in the model. As will be discussed, System Identification will focus on a special class of systems and accompanying models. In these lecture notes the term Parameter Estimation will be used for more general cases, including non-linear systems. Furthermore, the models will have a clear physical background and hence the parameters involved have a physical meaning.Parameter Estimation will also be used to find models of dynamic systems from experimental data, but it will typically be used in combination with a physical model. Hence the obtained parameters should represent certain physical properties of the system being examined. The models may be linear as well as non-linear. It will be shown that the linearity of the model (or the system being examined) is not the main reason for the complexity of the parameter estimation. More precisely, it may be possible to derive a set of equations for a non-linear system that is linear-in-the-parameters . In that case, a straightforward procedure can be applied to uniquely determine the “best” parameter set. In reverse, even a linear system can easily give rise to equations that are non-linear-in-the-parameters and non-linear optimisation techniques have to be applied. As outlined above, we will deal mostly with dynamic systems, which means that the time dependency of the signals

ISSN 2348 – 7968

plays an important role. It also means that the models describing the systems will include differential equations in which time derivatives (or their discrete time equivalent differences) are present. Nevertheless are the applied techniques closely related to curve fitting techniques that are commonly applied to match experimental data and non-dynamic models, i.e. models in which the time does not play a role. Examples will include such curve fitting as well.

1.2 DYNAMIC NEURAL NETWORKS

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model or function. Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted which means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums up all weighted inputs and bias (Figure1). Artificial neural network is a method of information processing, which is developed by the biological neural systems inspired. Based on the learning sample process, the artificial neural network analyzes the data mode, builds the model and then finds some new knowledge. Neural network can automatically adjust the neurons input and output in accordance with the rules through learning, to change the internal state.

In black box system identification, however, the really important task is to build models for dynamic systems. In dynamic systems the output at a given time instant depends not only on its current inputs, but on the previous behaviour of the system. Dynamic systems are systems with memory.

There are several ways to form dynamic neural networks using static neurons. Storage elements can be used in different parts of a static network. For example, some storage modules can be associated with each neuron, with the inputs or with any intermediate nodes of a static network. As an example a feed-forward dynamic network can be constructed from a static multi-input − single-output network (e.g., from an MLP or RBF) if a tapped delay line is added as shown in Figure 1.1. This means that the static network is extended by an embedded memory, which stores the past values of the inputs.

Multi-input single-output

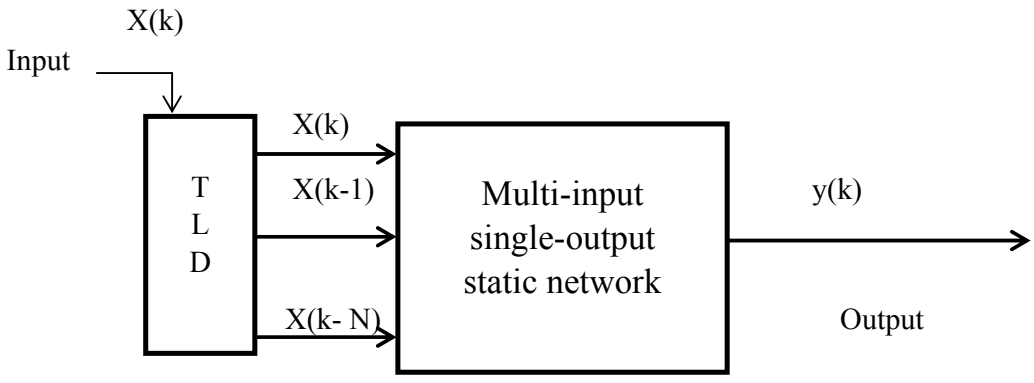


Figure 1.1 Feed-forward dynamic neural network architecture.

ISSN 2348 – 7968

If the tapped delay line is used in the output signal path, a feedback architecture can be constructed, where the inputs or some of the inputs of a feed-forward network consist of delayed outputs of the network. The resulted network is a recurrent one. A possible architecture where tapped delay lines are used both in the input and in the output signal paths is shown in Figure 1.2.

Multi-input single output

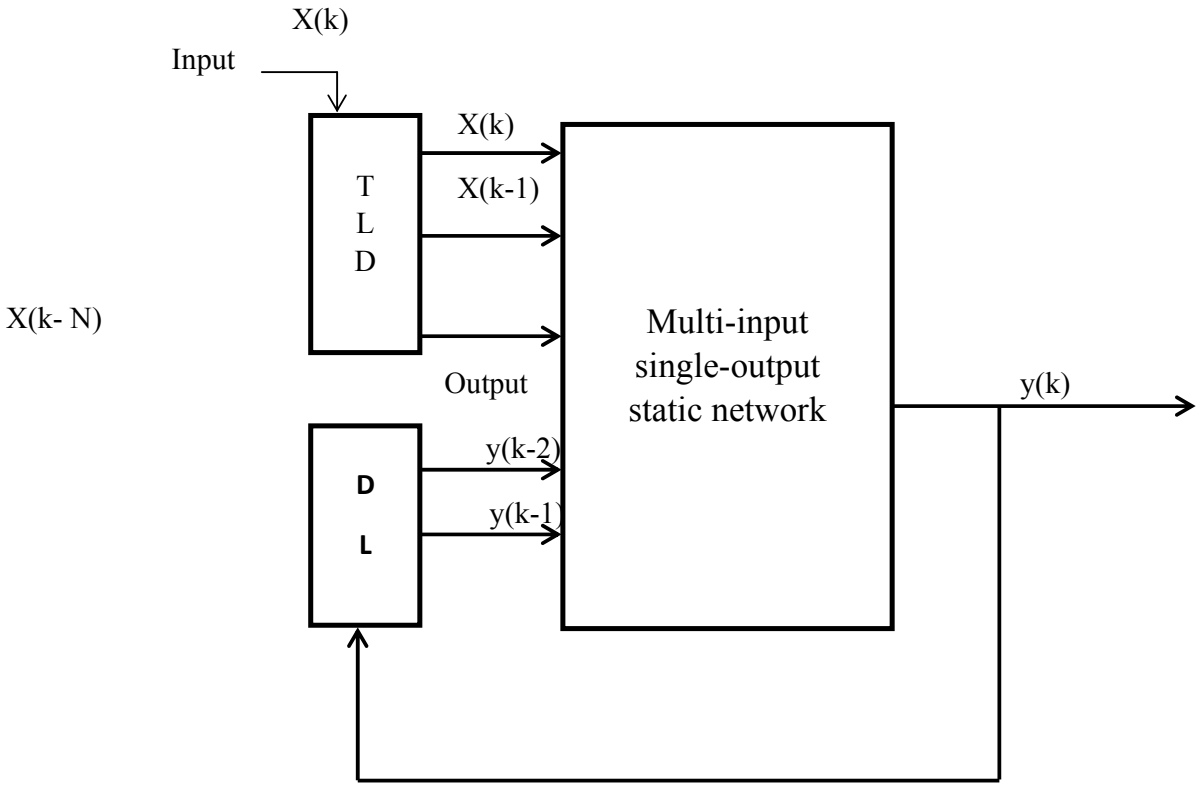
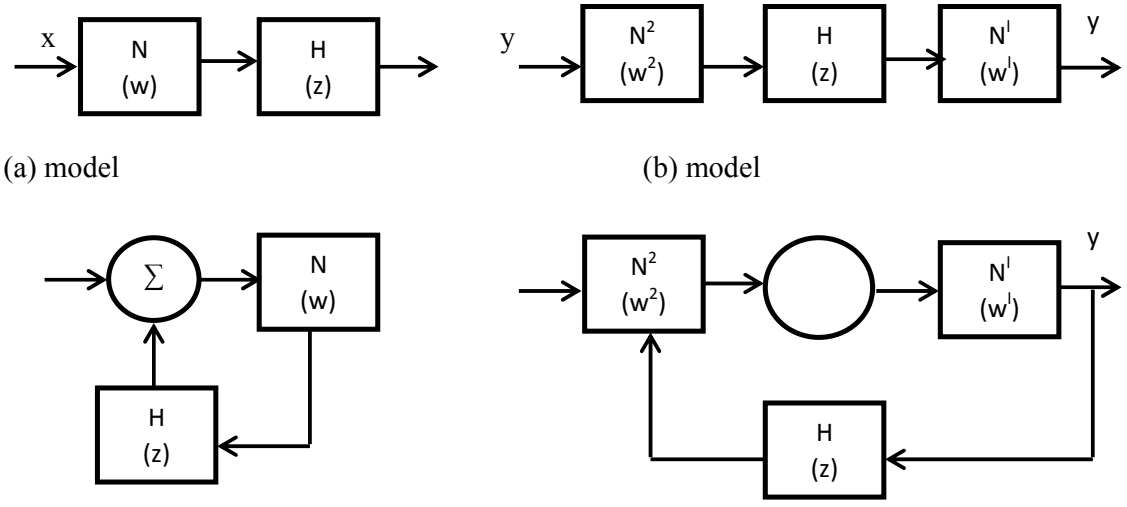


Figure 1.2: A dynamic neural architecture with feedback.

A further possibility to construct dynamic neural network is to combine static neural networks and dynamic linear networks. Within this approach both feed-forward and feedback architectures can be defined as proposed by Narendra . In Figure 1.3 some combined architectures are shown. There is a model structure called Wiener – Hammerstein model, which is similar to model (b) except that a static nonlinear system is placed between two linear dynamic ones.

ISSN 2348 – 7968



(a) model (b) model

(c) model (d) model

Figure 1.3: Combined dynamic neural architectures.

1.4 Designing dynamic model structures

In nonlinear system identification, a much more general approach can be followed. In this approach – similarly to the building of linear dynamic black box models – general nonlinear model structures can be designed. In these dynamic model structures a regressor vector is used, and the output of the model is described as a parameterized function of this regressor vector :

$$y_M(k)=f(\Theta,\phi(k)) \quad (1)$$

where Θ is the parameter vector and $\phi(k)$ denotes the regressor vector. The regressor can be formed from past inputs, past system outputs, past model outputs etc. according to the model structure selected. The following regressors are defined: When only the past inputs are used the regressor is formed as:

$$\phi(k)=[x(k-1),x(k-2),\ldots..,x(k-N)] \quad (2)$$

Based on this regressor, a feed-forward nonlinear model structure can be constructed. This model - similarly to its linear counterpart - is called an NFIR model. An NFIR model does not contain feedback so it cannot be unstable using any parameter vector. This is the simplest case of regressor-based architectures.

$$\phi(k)=[x(k-1),x(k-2),\ldots..,x(k-N),y(k-1),y(k-2),\ldots..,y(k-P)] \quad (3)$$

If both past inputs and system outputs are used in the regressor, the NARX model can be constructed. This model is often called series-parallel model, as it uses a feedback. The regressor can be formed from the past inputs and past model outputs

$$\phi(k)=[x(k-1),x(k-2),\ldots.. ,x(k-N),y_M(k-1),y_M(k-2),\ldots.. , y_M (k - P)] \quad (4)$$

ISSN 2348 – 7968

The corresponding structure is the NOE model. In a NOE model there is a feedback from model output to its input, so this is a recurrent network. Sometimes NOE model is called as parallel model. Because of its recurrent architecture serious instability problem may arise. In the NARMAX model the past inputs, the past system outputs and the past model outputs are all used. Usually the past model outputs are used to compute the past values of the difference between the outputs of the system and the model,

$$\varepsilon(k-i)=y(k-i)-y_M(k-i),\ i=1,2,\ldots..,L \quad (5)$$

so the regressor is as follows:

$$\phi(k)=[x(k-1), x(k-2),\ldots..,x(k- N), y(k-1), y(k-2), \ldots.. , y(k-P), \varepsilon(k-1), \ldots.. , \varepsilon(k–L) \quad (6)$$

The regressor for the NBJ models is formed from past inputs, past model outputs and the past values of two different errors, ε and ε x. Here ε is defined as before, while ε_x is

$$\varepsilon_x(k-i)=y(k-i)-y_{Mx}(k-i),\ i=1,2,\ldots..,K \quad (7)$$

In this equation $y_{Mx(}k-i)$ is the model output when only the past inputs are used. The corresponding regressor is

$$\phi(k)=[x(k-1),\ldots..,x(k-N),y_M(k-1),\ldots.,y_M(k-P),\varepsilon_x(k-1),\ldots.,\varepsilon_x(k–l),\ldots.., \varepsilon_x(k – K) \quad (8)$$

Although the definitions of these general model classes are different from the definition of the classical dynamic neural architectures, those structures can be classified according to these general classes. For example, an FIR-MLP is an NFIR network, but the combined models (a) and (b) in Figure 1.3 also belong to the NFIR model class.

1.5 Neural network training

In neural networks the estimation of parameters, the determination of the numerical values of the weights is called learning. Learning is an iterative process. When the weight values of the network are adjusted step by step, we can achieve the best fit between observed data and the model.

1.5.1 Training of dynamic networks

Dynamic networks are sequential networks, which means that they implement nonlinear mapping between input- and output data sequences. So the training samples of input–output data pairs of static networks are replaced by input–output data sequences and the goal of the training is to reduce a squared error derived from the elements of the corresponding error sequences. If ε(k) is the output error of a dynamic network at discrete time step k, the squared total error can be defined as:

$$\varepsilon_{total}=\sum_{i=1}^{k} \varepsilon^2(k)^2 \quad (9)$$

where k denotes the length of the sequence. Dynamic networks have memory, and this needs significant modification of the training algorithms. The basic training rules for dynamic

ISSN 2348 – 7968

systems are gradient-based algorithms. Here the weights are modified only after a whole training sequence were applied to the network. This will let the network be unchanged during a whole training data sequence is applied. The most important family of learning rules appropriate for dynamic networks is called dynamic backpropagation. For feed-forward networks a possible approach is to unfold the network in time. This strategy first removes all time delays in the network by expanding it into an equivalent static network. A more efficient learning for such network as an FIR-MLP is the temporal backpropagation. For recurrent networks two different approaches are applied . The first one uses also unfolding in time, which means that a recurrent dynamic network is transformed into a corresponding feed-forward static one. This transformation maps the neurons with their states at every time step into a new layer, where the number of resulting layers is equal to the length of the unfolding time interval. In the unfolded network all weights of the original recurrent network are repeated in every layer. The resulted static network are trained by standard backpropagation rule, except that these weights are physically identical and should be modified by the same value in one training step. The unfolding-in-time approach is called backpropagation through time (BPTT). BPTT can be explained most easily in an example. Figure 1.4 (a) shows a simple recurrent network with only two neurons.

Suppose that a four-length input sequence is used, the corresponding unfolded feed-forward static network is shown in Figure 1.4 (b). The two networks are equivalent for these four steps.

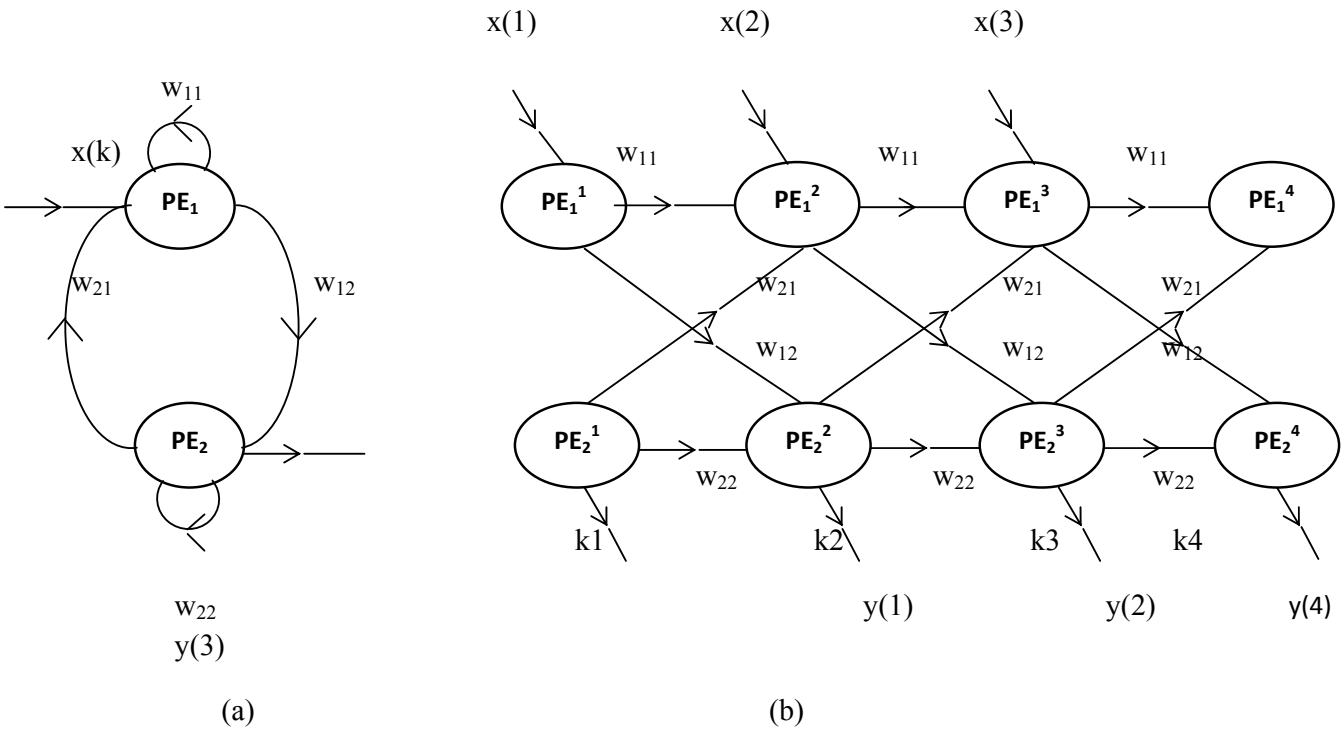


Figure 1.4 : Unfolding-in-time for a simple recurrent network

a) original recurrent network, b) unfolded static feed-forward network

ISSN 2348 – 7968

1.6. Model validation

We need to build an accurate model of the system in the whole operating range of interest. An important feature of a model is that it can approximate well the behaviour of a system not only at the training points, but in the whole operating range. This feature is called generalization.

A neural network without any generalization can only memorize the training points, so it works as a simple lookup table. Validation is used to estimate the performance of the model, to check its generalization capability. Validation serves several sub-goals. There are validation methods to check if model complexity was selected properly, and there are validation methods what can be used in the learning phase. The adequacy of the selected model class and model size can be determined only after model parameters are also determined. A model of proper complexity is used if both the model class and the model size (model order, the number of free parameters) are chosen appropriately. A proper model class can be selected either using prior knowledge about the system, or - according to the principle of parsimony - we have to select as simple model class as possible. For model size selection there are general validation methods used in linear or nonlinear system identification and there are special ones developed for neural networks.

Several different validation methods are used. Among them there are methods, which are used for both purposes: to check model complexity and check model parameters. It is well known, that the more complex model is used the better approximation can be reached at the training points. The reason is that increasing the number of the parameters the degree of freedom will be increased, which means that we can adjust the model parameters to fit the training data more. However, reducing the training error does not reduce necessarily the error at different points obtained from the same problem, but not used in training, so reducing the training error does not mean to get better generalization. For checking the generalization capability of the model we need a set of test data from the same problem, a test set, which is not used in training. Using different data sets for constructing a model and for validating it is an important principle. The validation method based on this principle is called cross-validation and it has a distinguished role in neural modelling. The effect of model complexity on the performance of the model can be followed in Figure 1.5. It shows the training and test errors versus model complexity. The performance is measured as usual, e.g., they are the sum of the squared errors at all training-points and at all test-points, respectively. It can be seen that as model complexity increases first both the training and the test errors decrease. This behaviour can be found until we reach a given complexity. From this point the lowering of the training error goes on, while test error is getting larger. A model of optimal complexity, a model with the best generalization property is obtained at the minimum point of the test error.

ISSN 2348 – 7968

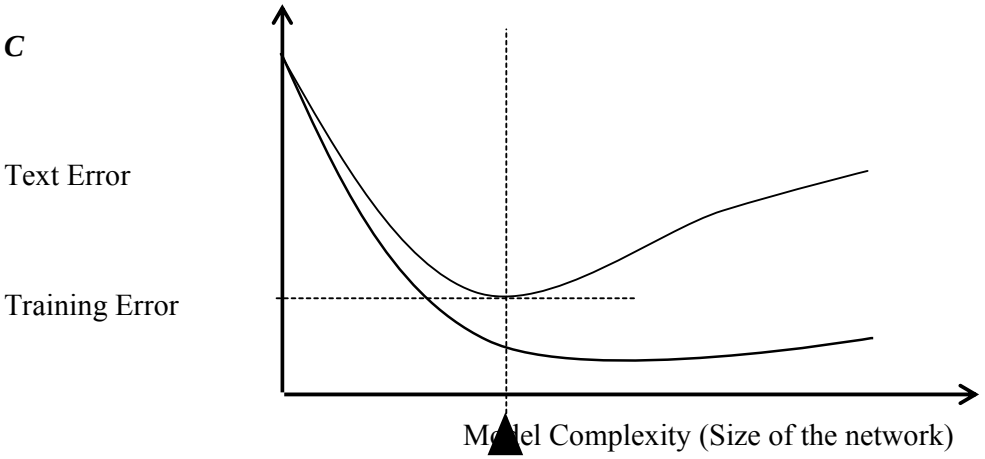


Figure 1.5 Training and test error versus model complexity

Optimal model complexity is the bias-variance trade-off. The modelling error is the sum of the squared errors or the average of the squared error

$$MSE_{emp}(\Theta)=\frac{1}{P}\sum_{k=1}^{P}(\varepsilon(k))^2=\frac{1}{P}\sum_{k=1}^{P}(y(k)-y_M(k))^2 \qquad (10)$$

where $\varepsilon(k)$ can be written in a more general form

$$\varepsilon(k)=y_k-y_M(\phi(k),\Theta) \qquad (11)$$

This error definition is valid for all model structures: if $\phi(k) = x(k)$. Now, consider the limit in which the number of training data samples goes to infinity, the average of the squared error approximates the mean square error, the expected value of the squared error, where expectation is taken over the whole data set.

$$MSE(\Theta)=E\{y-y_M(\Theta))^2\} \qquad (12)$$

This expression can be decomposed as:

$$MSE(\Theta)=E\{y-y_M(\Theta))^2\}= E\{(y_M(\Theta)-E\{y_M(\Theta)\}^2\}+(y-E\{y_M(\Theta)\})^2 \qquad (13)$$

Here the first term is the variance and the second one is the squared bias.

$$MSE(y_M(\Theta))=var(y_M(\Theta))bias^2(y_M(\Theta)) \qquad (14)$$

The size of the model, the model order will have an effect on the bias-variance trade- off. A small model with fewer than enough free parameters will not have enough complexity to represent the variability of the system's mapping, the bias will generally be high, while the variance is small. A model with too many parameters can fit all training data perfectly, even

ISSN 2348 – 7968

if they are noisy. In this case the bias term vanishes or at least decreases, but the variance will be significant.

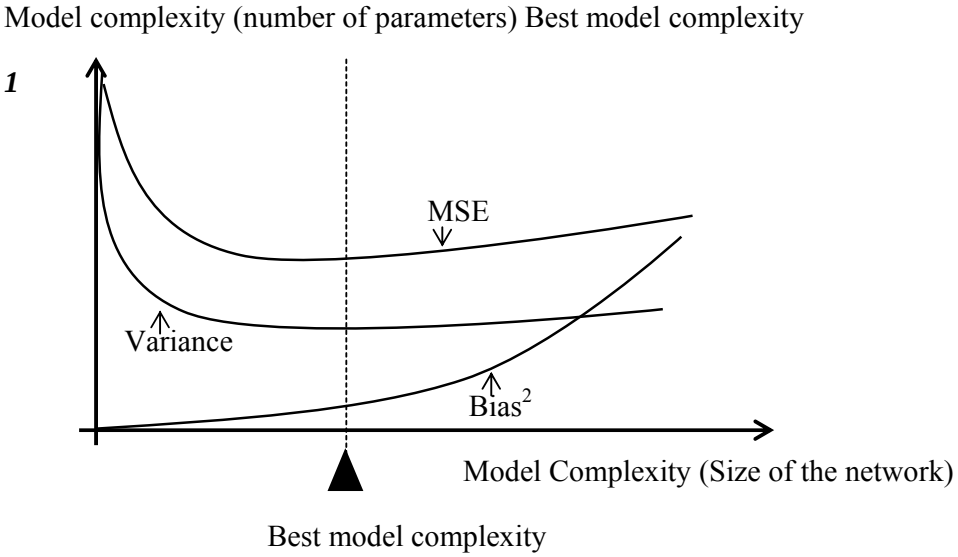


Figure 1.6 : Illustration to the bias-variance trade-off.

In dynamic models, a proper size of the selected model class must be determined at first. Moreover, it can be shown that the selection of the proper model complexity cannot be done independently from the number of available training data samples. There must be some balance between model complexity and the number of training data. The less training points are used, the less knowledge is available from the system and the less free parameters can be used to get a model of good generalization. Of course model complexity must reflect the complexity of the system, more complex systems need more data, which allows building more complex models: models with more parameters.

1.6.1 Model order selection for dynamic networks

A new heuristic method is proposed for identifying the orders of input-output models for unknown nonlinear dynamic systems . This approach is based on the continuity property of the nonlinear functions, which represent input-output mappings of continuous dynamic systems. The interesting and attractive feature of this approach is that it solely depends on the training data. The model orders can be determined using the following index:

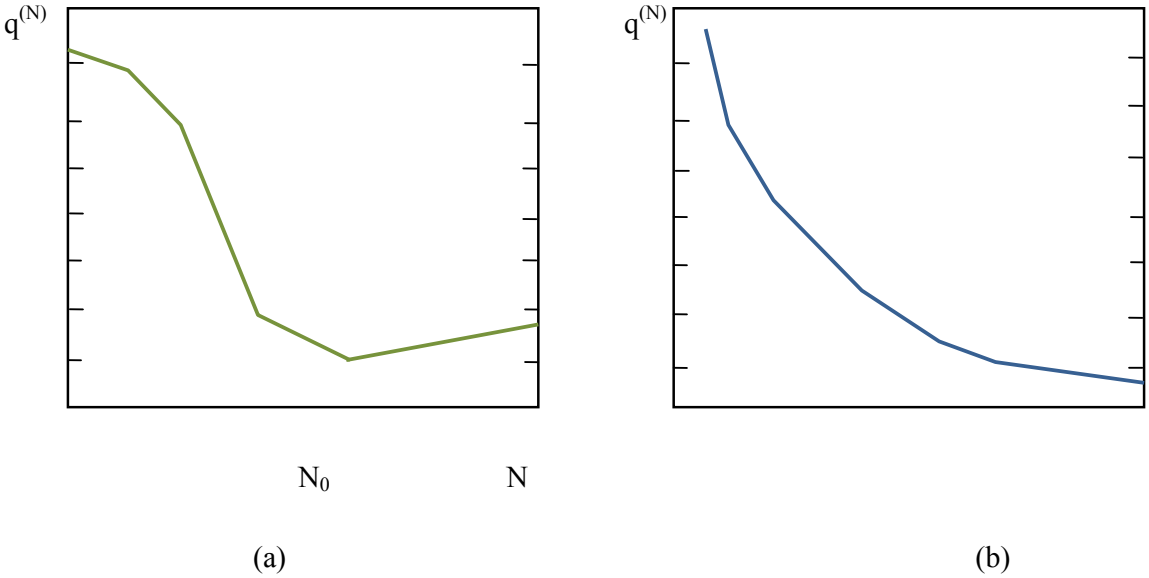
$$Q^{(N)}=\prod_{k-1}^{P}\sqrt{n}q^{(N)}(k)1/P \qquad (15)$$

where $q^{(N)}(k)$ is the k-th largest Lipschitz quotient among all q_{ij} ($i \neq j$; i, j = 1, 2, … , P) . N is the number of input variables and p is a positive number: usually 0.01P - 0.02P. Here the q_{ij} Lipschitz quotient is defined as:

$$q_{ij}=\frac{|y(i)-y(j)|}{|x(i)-x(j)|} \qquad (16)$$

ISSN 2348 – 7968

where the {x(i), y(i)} i=1, 2, … , P pairs are the measured input-output data samples from which the nonlinear function f(.) have to be reconstructed. This index has the property that $1+q^{(N)}$ is very close to $q^{(N)}$, while $1-q^{(N)}$ is much larger than $q^{(N)}$ if N is the optimal number of the input variables, so a typical curve of $q^{(N)}$ versus N has a definite point (N_0) where the decreasing tendency stops and $q^{(N)}$ enters a saturated range. For an NFIR model N_0 is the optimal number of input order. Figure 1.7 (a) shows a typical curve for $q^{(N)}$.



(a) (b)

Figure 1.7 : Typical curves of Lipschitz indexes (a) for noiseless data or data with low noise level, (b) for data with high noise level.

The Lipschitz index can be applied not only for NFIR structures but also for NARX model classes, where two, the order of the feed-forward and the feedback paths must be determined. For NARX model class

$$y_M(k)=f(\phi(k))=f[x(k-1),x(k-2),\ldots,x(k-M),y(k-1),y(k-2),\ldots,y(k-L)] \quad (17)$$

the following strategy is used. The Lipschitz index $q^{(N)} = q^{(L-M)}$ are computed for different model orders, where L denotes the feedback and M the feed-forward order values. Starting with N=1, where only y(k-1) is used as input $q^{(1+0)}$ can be computed. Then let N = 2, where the both x(k-1) and y(k-1) are used as inputs and $q^{(1+1)}$ can be computed. For N=3 the third input of the dynamic networks will be y(k-2) and $q^{(2+1)}$ will be computed. This strategy can be followed increasing step by step the feedback and the feed-forward orders. If at a given L and M one can observe that $q^{(L+M)}$ is much smaller than $q^{(L-1+M)}$ or $q^{(L+M-1)}$, but is very close to $q^{(L+1+M)}$ or $q^{(L+M+1)}$, we reached the appropriate order values. The most important advantage of this method is that it can give an estimate of the model order without building and validating different complexity models, so it is a much more efficient way of order estimation then the criteria based approaches. However, there is a significant weakness of the Lipschitz method: it is highly sensitive to observation noise. Using noisy data for model construction - depending on the noise level - we can often get a typical curve for the

ISSN 2348 – 7968

Lipschitz index as shown in Figure 1.7 (b). The most important feature of this Figure is that there is no definite break point.

1.6.2 Cross-validation

Modelling error can be used in another way for model validation. This technique is called cross-validation. In cross-validation – as it was mentioned before - the available data set is separated into two parts, a training set and a test set. The basic idea of cross-validation is that one part of the available data set is used for model construction and another part for validation. Cross-validation is a standard tool in statistics and can be used both at the model structure selection and at parameter estimation. Here its role in the training process will be presented. The previous validation techniques for selecting the proper model structure and size are rather complex, computation intensive methods. This is the most important reason why they are applied only rarely in practical neural model construction. The most common practical way of selecting the size of a neural network is the trial and error approach. First a network structure is selected, and then the parameters are trained. Cross-validation is used to decide whether or not the performance of the trained network is good enough. Cross-validation, however, is used for another purpose too. As it was mentioned in the previous section to determine the stopping time of training is rather difficult as a network with quite large number of free parameters can learn the training data almost perfectly. The more training cycles are applied the smaller error can be achieved on the training set. However, small training error does not guarantee good generalization. Generalization capability can be measured using a set of test data consists of samples never seen during training. Figure 1.8 shows two learning curves, the learning curves of the training and the test data. It shows, that usually the training error is smaller than the test error, and both curves decrease monotonically with the number of training iterations till a point, from where the learning curve for the test set starts to increase. The phenomenon when the decrease of the training error is going on, while the test error starts to increase is called overlearning or over fitting. In this case the network will memorize the training points more and more while at the test points the network's response is getting worse, we get a network with poor generalization. Overlearning can be avoided if training is stopped at the minimum point of the test learning curve. This is called early stopping and it is an effective way to improve the generalization of the network even if its size is larger than required. For cross-validation we need a training set and a test set of known examples. However there is a question which must be answered: in what ratio, the data points should be divided into training and testing sets in order to obtain the optimum performance. Using statistical theory a definite answer can be given to this question . When the number of network parameters M is large, the best strategy is to use almost all available known examples in the training set and only $\frac{1}{\sqrt{2M}}$ examples in the testing set, e.g., when M = 100, this means that only 7% of the training data points are to be used in the test set to determine the point for early stopping. These results were confirmed by large-

ISSN 2348 – 7968

scale simulations. The results show that when P > 30M cross-validation is not necessary, because the generalization error becomes worse by using test data to obtain adequate stopping time. However, for P < 30M, i.e. the number of the known examples is relatively small compared to the number of network parameters, overtraining occurs and using cross-validation and early stopping improves generalization. Cross-validation can be used not only for finding the optimal stopping point, but to estimate the generalization error of the network too. In network validation several versions of cross-validation are used. A version called one-leave-out cross-validation is used, especially if the number of known data is small. The one-leave-out cross-validation is an efficient way of using the examples available. Here we divide the set of examples into two sets as it was proposed before, but only one example will be omitted from the training set and this point will be used for testing. The process will be repeated P times, every time a different example is omitted for testing. Such a procedure allows us to use a high proportion of the available data (all but one) to train the network, while also making use of all data points in evaluating the cross-validation error. The disadvantage of this method is that it requires the training process to be repeated P times.

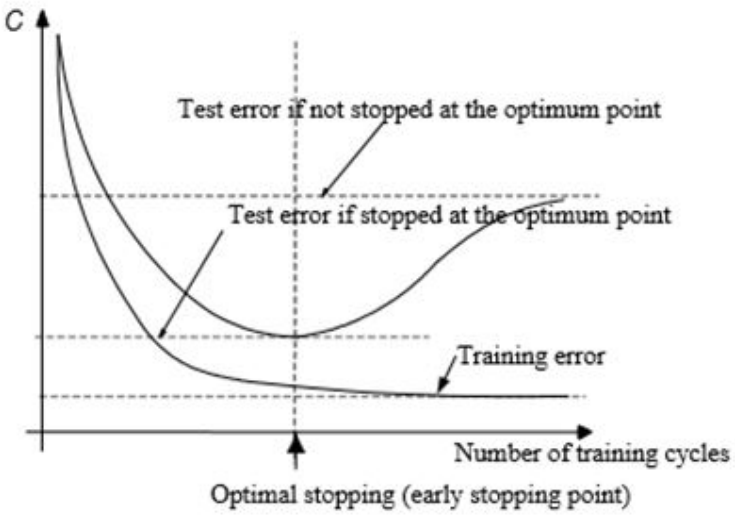


Figure 1.8: Learning curves for the training and the test data.

7. Conclusions

This paper shows that neural networks are general black box modelling devices, which have many attractive features: they are universal approximates, they have the capability of adaptation, fault tolerance, robustness, etc. For system modelling several different dynamic neural architectures can be constructed, so neural architectures are flexible enough for a rather large class of identification tasks. The construction of neural models - as they are black box architectures - is mainly based on measurement data observed about the system. This is

ISSN 2348 – 7968

why one of the most important parts of black box modelling is the collection of as much relevant data as possible, which cover the whole operating range of interest. All these problems need proper pre-processing, the importance of which cannot be overemphasized. Moreover, according to the experiences obtained from real-world modelling tasks, prior information and any additional knowledge to the observation has great importance. Prior information helps us to select proper model structure, to design excitation signal if it is possible to use excitation signals at all, to determine the operating range where valid model should be obtained, etc.

REFERENCES

[1] L. Ljung, System Identification - Theory for the User. Prentice-Hall, N.J. 2nd edition, 1999.

[2] J. Schoukens and R. Pintelon, System Identification. A Frequency Domain Approach, IEEE Press, New York, 2001.

[3] T. Söderström and P. Stoica, System Indentification, Prentice Hall, Enhlewood Cliffs, NJ. 1989.

[4] P. Eykhoff, System Identification, Parameter and State Estimation, Wiley, New York, 1974.

[5] A. P. Sage and J. L. Melsa, Estimation Theory with Application to Communications and Control, McGraw-Hill, New York, 1971.

[6] H. L. Van Trees, Detection Estimation and Modulation Theory, Part I. Wiley, New York, 1968.

[7] G. C. Goodwin and R. L. Payne, Dynamic System Identification, Academic Press, New York, 1977.

[8] K. Hornik, M. Stinchcombe and H. White, Multilayer Feed-forward Networks are Universal Approximators", Neural Networks Vol. 2. 1989. pp. 359-366.

[9] G. Cybenko, Approximation by Superposition of Sigmoidal Functions, Mathematical Control Signals Systems, Vol. 2. pp. 303-314, 1989.

[10] K. I. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks", Neural Networks, Vol. 2. No. 3. pp. 1989. 183-192.

[11] M. Leshno, V. Y. Lin, A. Pinkus and S. Schocken, Multilayer Feed-forward Networks With a Nonpolynomial Activation Function Can Approximate Any Function, Neural Networks, Vol. 6. 1993. pp. 861-67

[12] J. S. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), Transaction of the ASME, Sep. 1975. pp. 220-227.

[13] Y. H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading, Mass., 1989, pp. 197-222.

ISSN 2348 – 7968

[14] D. F. Specht, Polynomial Neural Networks, Neural Networks, Vol.3. No. 1 pp. 1990. pp. 109-118,

[15] J. Park and I. W. Sandberg, Approximation and Radial-Basis-Function Networks, Neural Computation, Vol 5. No. 2.

[16]David E Goldberg, „Genetic algorithms in search, optimization and machine learning", Addision- Wesley Pub.Co.1989.

[17]C.E.Shannon, "Communication Theory of Security System",Bell,System Technical Journal, 28, 1949

[18]A.J.Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", School of Information Systems, University Of East Anglia, 1996.

[19]N.Koblitz , „A Course in Number Theory and Cryptography", Springer-Verlag, New York, Inc., 1994.

[20]Menzes A. J., Paul, C., Van Dorschot, V., Vanstone, S. A., "Handbook of Applied Cryptography", CRS Press 5th Printing; 2001.

[21]National Bureau Standards, "Data Encryption Standard (DES)," FIPS Publication 46; 1977

[22]Tragha A., Omary F., Mouloudi A.,"ICIGA: Improved Cryptography Inspired by Genetic Algorithms", Proceedings of the International Conference on Hybrid Information Technology (ICHIT'06), pp. 335-341, 2006.

[23]Spillman R,Janssen M, Nelson B and Kepner N, "Use of Genetic Algorithm in Cryptanalysis of Simple Substitution Cipher" Cryptologia, Vol.17, No.4, pp. 367- 377, 1993.

[24]Spillman R,"Cryptanalysis of Knapsack Ciphers using Genetic Algorithms", Cryptologia, Vol.17, No.4, pp. 367-377, 1993.

[25]Garg Poonam, Genetic algorithm Attack on Simplified Data Encryption Standard Algorithm, International journal Research in Computing Science, ISSN1870-4069, 2006.

[26]Nalini, Cryptanalysis of Simplified data encryption standard via Optimization heuristics, International Journal of Computer Sciences and network security, vol 6, No 1B,Jan 2006. 2nd National Conference in Intelligent Computing & Communication Organized by Dept. of IT, GCET, Greater Noida, INDIA. ISBN: 9788175157538

[27] L. Ljung, System Identification - Theory for the User. Prentice-Hall, N.J. 2nd edition, 1999.

[28] J. Schoukens and R. Pintelon, System Identification. A Frequency Domain Approach, IEEE Press, New York, 2001.

[29] T. Söderström and P. Stoica, System Indentification, Prentice Hall, Enhlewood Cliffs, NJ. 1989.

[30] P. Eykhoff, System Identification, Parameter and State Estimation, Wiley, New York, 1974.

ISSN 2348 – 7968

[31] A. P. Sage and J. L. Melsa, Estimation Theory with Application to Communications and Control, McGraw-Hill, New York, 1971.

[32] H. L. Van Trees, Detection Estimation and Modulation Theory, Part I. Wiley, New York, 1968.

[33] G. C. Goodwin and R. L. Payne, Dynamic System Identification, Academic Press, New York, 1977.