In Prceddings of the 1998 IEEE Int. Conf. on Fuzzy Systems, Anchorage, Alaska

Constructing Fuzzy Controllers for Multivariate Problems by Using Statistical Indices

Jianwei Zhang and Alois Knoll
Technical Computer Science, Faculty of Technology,
University of Bielefeld, 33501 Bielefeld, Germany

Abstract

This paper proposes an approach for solving multivariate control problems with a fuzzy controller. Instead of using selected input variables, statistical indices are extracted to feed the fuzzy controller. The original input space is transformed into an eigenspace. If a sequence of training data are sampled in a local context, a small number of eigenvectors which possess larger eigenvalues provide a good summary of all the original variables. Fuzzy controllers can be trained for mapping the input projection in the eigenspace to the outputs. Implementations with the prediction of time series and vision-based robot control validate the concept.

1. Multivariate Problems in Modelling and Control

It is well-known that general fuzzy rule descriptions of systems with a large number of input variables suffer from the problem of the "curse of dimensionality". In many real applications, it is difficult to identify the correct influential factors and reduce their number to the minimum. A general solution to build fuzzy models is not only interesting from a theoretical point of view, but also practical meaningful since the application of fuzzy control could then be extended to a wider range of complex intelligent control problems.

In the recent literature on neuro-fuzzy modelling and machine learning, some standard benchmark problems with "middle"-dimensional input spaces, like "Box-Jenkins Gas Furnace Problem", "Mile-Per-Gallon" etc, are frequently discussed and simulation results are presented. In our industry consulting and research work on sensor-based robot control, we are faced with some high-dimensional problems concerning a large number of input variables whose importance and inter-dependence are not clearly known. These typical problems are briefly described as follows:

System Identification. As training data set for nonlinear system identification, the Box-Jenkins gas furnace data [1] is often studied and compared. The furnace input is the gas flow rate $x(t)$, the output $y(t)$ is the CO_2 concentration. At least 10 candidate inputs are considered: $x(t-6), x(t-5), \ldots, x(t-1), y(t-1), \ldots, y(t-4)$. If all of them are used, building a fuzzy controller means to solve a 10-input-1-output problem. If each input is defined by 5 linguistic terms, this would result in a fuzzy rule system of about 10 million rules.

The modelling and prediction of financial markets are also complex problems. Experts estimate more than hundred influential factors, which are hard to differentiate since they are inter-related.

Complex Machine Adjustment. One of our joint R/D projects aims at automatic adjustment of a thresher machine. To achieve the minimum grain loss and maximum harvest speed, a set of parameters of the thresher system, such as opening width, rotating and blowing speed, should be optimally set. Until now no models for the optimal machine settings are known. Even an expert cannot always achieve the best setting of the thresher parameters. The main assumed influence factors are: grain type, thousand-grain-weight, grain moisture, straw composition, straw moisture, air humidity, temperature, general weather condition, season, wind velocity, vehicle speed and so on. Clearly, fuzzy sets can find ideal applications in modelling many of vague, only approximately measurable variables. Since a fuzzy model can normally contain only a very limited number of inputs, the problem remains: which influence factors should be taken into account so that most information can be utilised?

Vision-guided robot motion. The classical approaches of robot vision face two big problems: a). The image processing procedures, such as segmentation, feature extraction and classification are not robust in real environments; b). These processing algorithms are computationally expensive, thus the real-time requirement cannot be fulfilled in most cases. It is a long-term research goal to find a general model which transforms raw image data directly to action values. For instance, an image has 192×144 pixels, each pixel is measured by brightness level. If no extra image processing is performed, then a control system with about 26,000 input variables needs to be modelled. The outputs of the system are the motion values for a robot (mobile robot as well as robot arm). Is it possible to build a fuzzy model to map the inputs to the outputs?

2. Existing Solutions

Two main methods examining the multivariate problem are "input selection" and "hierarchy".

Input Selection One implemented approach is "input selection", [4] and [2], which is in principle an experimental method to find the most important input variables among a large number of them. With this approach, all the combinatorial possibilities of the low-dimensional fuzzy model are considered and approximately tested. The selected inputs which enable the best result are viewed as the most influential ones to build an exact neuro-fuzzy model. There are two obvious problems with such a procedure of projection from a high dimension into a low one. First, all the other less influential inputs are discarded, which means an information loss for the controller. Secondly, the combinational number of lower-dimensional fuzzy controllers for a system with thousands or even millions of inputs is still too large to enumerate and to evaluate.

Hierarchy The solution with hierarchical structure assumes that the input information can be classified into groups, see [5] for an example. Within each group the inputs determine an intermediate variable, they can be decoupled from inputs of other groups. To realise such a grouping, there exists no general automatic approach but heuristics based on fusion of physical sensors.

If a fuzzy controller is only observed at each moment in time, the only input information to the controller comes from the whole set or a selected subset of the input variables. In this paper, this kind of information is called *horizontal index*. Both "input selection" and hierarchical structure consider only the horizontal indices. They presume that inputs are independent and give no priority or importance of the selected input variables.

3. Concept of Using Statistical Indices

3.1. Sensor Pattern and Situation

If the dimension of the input space is small, the input variables can be directly covered by fuzzy sets, thus fuzzy rules are formed in a tabular form. Each item of the rule table can be interpreted as a general situation. Such a "situation" is still intuitively understandable for a small number of inputs. This is certainly no longer the case if the number of the input variables is no more 2, 3, 4, but 10, 10.000, or even millions.

If all inputs are normalised and the value of each one is viewed as the brightness level of an image pixel, an input vector can be regarded as a general sensor pattern, like a CCD camera image. In this point of view, a multivariate modelling problem is in principle equivalent to a supervised vision-based control problem.

There will be no doubt that the complete "situations" of the high-dimensional sensor patterns cannot be represented as a number of computer internal memorable and computable tabular items. However, if we observe the sensor patterns in a certain local context, it can often be found that they distribute within certain parts of the whole input space.

3.2. Vertical versus Horizontal Indices

If the input vectors are considered as a pile of training data originating from a continuously varying process, instead of being viewed only as a separate input vector, *vertical indices* can be extracted by statistical analysis. These vertical indices describe data distribution, variances, and the most interesting features for control - projections on the eigenvectors (Fig. 1).

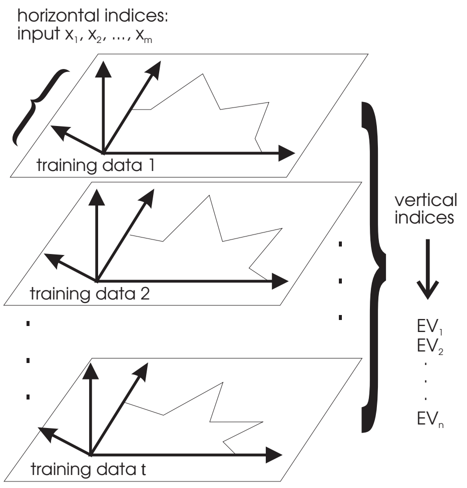


Figure 1. Vertical information about the distribution of the data in the input space can be extracted by statistics on a sequence of the training data set.

For an input space $X_1 \times X_2 \times \cdots \times X_m$, if all the variables x_1 to x_m could vary in all their universes in the observed sampling procedure, the input data would scatter in the whole input space. Nevertheless, in a high-dimensional input space, if the observed process runs continuously, the input vector varies gradually. Under an appropriately separated observation, e.g. within a local scenario of the robot environment, the input vectors possess a large grade of similarity in most cases. That means in the high-dimensional input space, the observed input data are correlated to a large degree.

If certain correctly selected original input variables happen to be the axes, along which the sampling data aggregate, they can be directly used as fuzzy controller inputs. Otherwise, the information from the unselected variables will be lost.

3.3. Projection in Eigenspace

A well-known technique for dealing with multivariate problems in statistics is the *principal component analysis*. Until now, it is mainly applied in data compression and pattern recognition, [7]. In our opinion, this technique is also suitable for reducing the dimension of the input space of a general control problem.

An eigenvector, noted as EV_i, is computed as $[a_{1,i}, a_{2,i}, \cdots, a_{m,i}]^T$. The eigenvectors form an orthogonal basis for representing the original individual sensor patterns. Assume that the eigenvectors $EV_1, EV_2, \cdots$ are sorted according to their eigenvalues in a descending order. An eigenspace with a reduced dimension n can be formed with the first n eigenvectors. EV_i accounts for the ith dimension in the eigenspace. The projection of an input vector $[x_1, x_2, \cdots, x_m]$ on eigenvector EV_i, called the ith principal component, is $a_{1,i}x_1 + a_{2,i}x_2 + \cdots + a_{m,i}x_m$. All projections of the sample data sequence form a manifold in the eigenspace.

Depending how "local" the measuring data are and therefore how similar the observed sensor patterns look like, a small number of eigenvectors can provide a good summary of all input variables. It can be possible that two or three eigenvectors supply the most information indices of the original input space. In a very high-dimensional case, an efficient dimension reduction can be achieved by projecting the original input space into the eigenspace.

The eigenvectors of a covariance matrix can be efficiently computed by the perceptron approach [8]. This approach has three advantages over Jacobian method [3] if the number of the eigenvectors are pre-selected. First, the Jacobian method calculates all eigenvectors of the covariance or explicit covariance matrix of the input data; the perceptron method only calculates the first n eigenvectors, thus saves computation time. Second, Jacobian requires the construction of the covariance matrix of the input data. With large input data sets this is often critical regarding the amount of memory needed. Thirdly, the found eigenvectors with the perceptron method are already in descending order.

3.4. Fuzzy Partition into Eigenspace

Partition of eigenvectors can be done with a descending resolution: if the EV_1, ..., EV_n is ordered in a descending order, we just partition the EV_1 with the finest resolution, since the data projection on this eigenvector contains the largest variance. The second eigenvector EV_2 is covered with some less linguistic terms, so on and so forth. The last selected eigenvector EV_n is described with the least number of linguistic terms, see the middle part of Fig. 2.

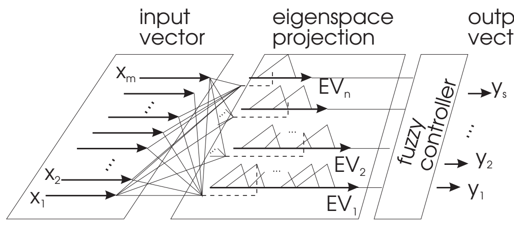


Figure 2. The structure of a fuzzy controller based on eigenspace projection.

In the online application, the input data are first projected into the eigenspace, then mapped to output based on the fuzzy control model (Fig. 2).

4. The Approach

By summarising the above ideas and formulating them in the chronological order, it turns out that constructing a fuzzy controller for a multivariate problem consists of two phases. The first phase is extra with this approach and should be done before the real construction procedure of the fuzzy controller. The second phase is in principle a normal supervised learning algorithm for a fuzzy controller, if the desired output data as well as the input data are available. After a fuzzy controller is constructed, given an online measured input $(x_1, x_2, \cdots, x_m)$, the outputs can be calculated through the eigenspace projection and fuzzy rule synthesis afterwards.

4.1. Phase 1: Sampling Training Data and Analysis

This special step aims at evaluating the statistical indices and performing the dimension reduction for the preparation of the direct controller inputs. Hereby it is desirable that all representative input data are generated.

1. Sample input data, record the desired output values if available.
2. Pre-process the input data: normalise and subtract the average value.
3. Transform the input variables into vectors.
4. Calculate the eigenvectors and eigenvalues.

4.2. Phase 2: Training a Fuzzy Controller

By using the supervised learning algorithm presented in our early paper [9], the training procedure looks as follows:

1. Select the n eigenvectors with the largest n eigenvalues, noted as $EV_1, \cdots, EV_n$.
2. Select the order of the B-spline basis functions for each eigenvector.
3. Determine the knots for partitioning each eigenvector.
4. Initialise the control vertices for the output.
5. Learn the control vertices using gradient descent method.
6. If the results are satisfied, terminate.
7. Modify the knots for eigenvectors, go to 4.

4.3. Online Application

In the case of supervised learning, each learning datum corresponds to a supporting point in the control space (Fig. 3). If a sensor pattern is taken online and its eigenvalues are calculated, the determination of the controller outputs is then the blending of all the firing rules.

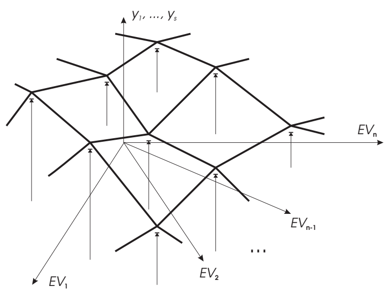


Figure 3. The application phase is an interpolation process if the control vertices have been learned.

The following steps are necessary:

1. Update the input data $(x_1, x_2, \cdots, x_m)$;
2. Pre-process the data - normalise and substrate average;
3. Project the input into the eigenspace $(EV_1, \cdots, EV_n)$;
4. Compute the output by feeding the projection vector into the fuzzy controller trained in section 4.2.

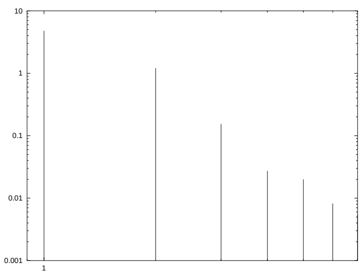
5. Implementations

The above approach can be applied to all type of fuzzy controllers, like Mamdani type and TSK type. In the following implementations, fuzzy controllers constructed by B-spline model are used (see our early work in [9]). We pointed out that definition of linguistic terms with B-spline basis functions requires less parameters than the other set functions such as trapezoid, Gaussian function, etc. The output computation becomes very simple and the interpolation process is transparent. In our later work, we demonstrated the good approximation capability and the rapid convergence of B-spline fuzzy controllers as well as the suitability for unsupervised learning [10].

In the following implementations, linguistic terms of the eigenvectors are defined by B-spline basis function of order three. The output value of each rule is represented as a fuzzy singleton which is called control vertex in the B-spline fuzzy controller. The control vertices are adaptively determined by minimising the "Root Mean Squared Error" (as in [4]) for the supervised learning.

5.1. Box-Jenkins Data

296 data in form of $(x(t), y(t))$ are first transformed into the form $((x(t-6), x(t-5), \ldots, x(t), y(t-1), \ldots, y(t-4)), y(t))$. The computed eigenvectors are shown in Table 4(a), the eigenvalues of each eigenvectors are depicted in Fig. 4(b). The projection of the data into the eigenspaces constructed by the first two and first three eigenvectors can be found in Fig. 5(a) and (b).



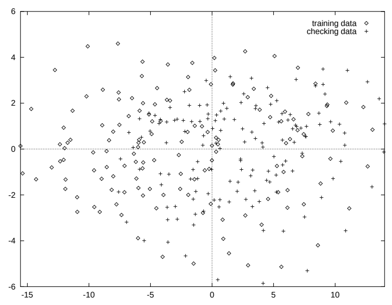
(a) Eigenvalues

	EV 1	EV 2	EV 3	EV 4	EV 5
$x(t-6)$	-0.158	0.074	0.239	0.354	0.602
$x(t-5)$	-0.149	0.204	0.233	-0.015	0.291
$x(t-4)$	-0.134	0.310	0.081	-0.382	0.115
$x(t-3)$	-0.117	0.372	-0.164	-0.419	0.288
$x(t-2)$	-0.099	0.386	-0.413	-0.044	0.228
$x(t-1)$	-0.084	0.361	-0.582	0.531	-0.222
$y(t-1)$	0.477	-0.334	-0.295	0.136	0.523
$y(t-2)$	0.487	-0.015	-0.236	-0.263	0.141
$y(t-3)$	0.480	0.278	0.053	-0.225	-0.248
$y(t-4)$	0.457	0.501	0.448	0.362	-0.017

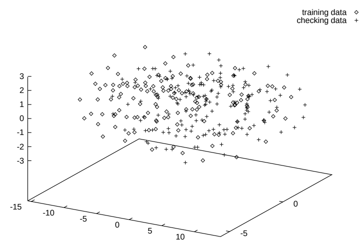
(b) Eigenvectors

Figure 4. The important eigenvectors and eigenvalues of the Box-Jenkins data.

The first, second, third and fourth eigenvector are defined by 10, 7, 7 and 5 linguistic terms respectively. The RMS (also MS) training errors achieved with the first three eigenvectors are listed in Tab. 1.



(a) 2D projection



(b) 3D projection

Figure 5. Distribution of the data in the 2D and 3D eigenspaces.

In [6], Table 2 was summarised as the comparison results for solving the Box-Jenkins gas furnace data. By comparing Tab. 1 and Tab. 2, it can be seen that as expected, the training error achieved with the first two eigenvectors is less than that achieved with all the above models, and with the first three or four eigenvectors, the error can be still reduced significantly. The RMS error is also less than that achieved by the ANFIS model with input selection in [4]. Fig. 6 shows the result achieved by using four eigenvectors.

5.2. Vision-Guided Grasping

This approach is applied to find exact grasp positions by a robot parallel gripper equipped with hand-camera system, Fig. 7. The novelty of this approach is that no prior hand-eye calibration is necessary.

	RMS (MS) Error		
epochs	1	2	3
100	0.73 (0.533)	0.22 (0.048)	0.25 (0.063)
1000	0.71 (0.504)	0.19 (0.036)	0.20 (0.04)
10000	0.71 (0.504)	0.19 (0.036)	0.17 (0.029)

Table 1. RMS (MS) training error by using 1, 2, and 3 first eigenvectors if all 296 data are used for training.

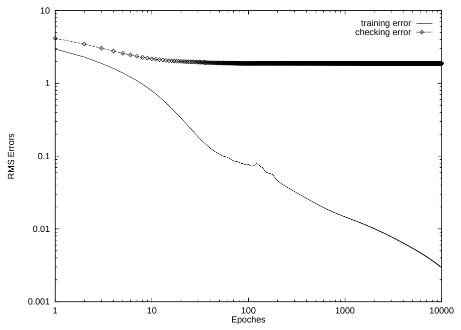


Figure 6. RMS errors of modelling the Box-Jenkins data with four eigenvectors. 148 data are used for training and 148 data for checking.

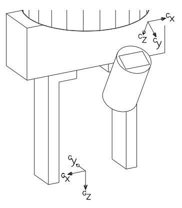
A off-line learning approach begins with positioning the robot gripper over the desired position. The robot hand is then incrementally perturbed, e.g. in the freedom of displacement x, y and/or the orientation θ. At each new location, an image is produced while the motion parameters are recorded. After all consequences of the possibly imprecise positioning of the robot hand were considered, a sequence of images is obtained. Each pair of two adjacent images is similar to a large degree, see Fig. 8.

The large amount of brightness data of the image pixels can be significantly reduced by finding the eigenvectors of their covariance matrix. If the eigenvectors are ordered according to the magnitude of their eigenvalues, it can be easily found that only a limited number of eigenvectors need to be considered while the others can be ignored.

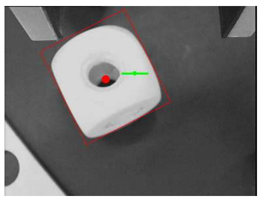
In this way, the dimension of the local perceptual space is reduced to a manipulable size of a subspace. If the eigenvalue of each selected eigenvector, noted as EV_j ($j = 1, \ldots, n$), is covered with B-spline basis functions, noted as $X^j_{i_j,k_j}$, the rule for determining the relative location of the robot gripper to the object can be written in the form:

Model	Input	Rule No.	MS Error
Tong's	$x(t-1), y(t-4)$	19	0.469
Pedrycz's	$x(t-1), y(t-4)$	81	0.320
Xu/Lu's	$x(t-1), y(t-3)$	25	0.328
Chiu's TSK 2	$x(t-1), y(t-3)$	3	0.146
Chiu's TSK 3	$x(t-1), x(t-3),$ $y(t-3)$	3	0.072
[6] GA-fuzzy	$x(t-1), y(t-4)$	25	0.257

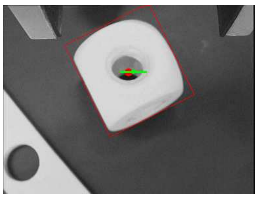
Table 2. Comparison of different models derived using the Box and Jenkins gas furnace data, excerpted from [6].



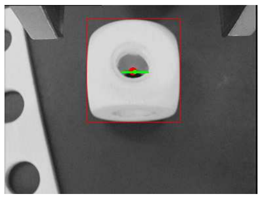
(a) “Self-viewing” hand-camera



(b) The start position



(c) Improved position



(d) The optimal position for grasping

Figure 7. Fine-Positioning using a “self-viewing” hand-camera.

IF $(EV_1$ IS $X^1_{i_1,k_1})$ and ... and $(EV_n$ IS $X^n_{i_n,k_n})$
THEN $(x$ is $X_{i_1 i_2 \dots i_n})$ and $(y$ is $Y_{i_1 i_2 \dots i_n})$
and $(\theta$ is $\Theta_{i_1 i_2 \dots i_n})$

Each rule corresponds to a supporting point for the interpolation in the eigenspace. Our experiment showed that with a few eigenvectors, a correction of the robot hand can be attained.

6. Discussion

The main advantage of the proposed approach to “input selection” is that less information is lost after the dimension reduction for problems with correlated input training data.

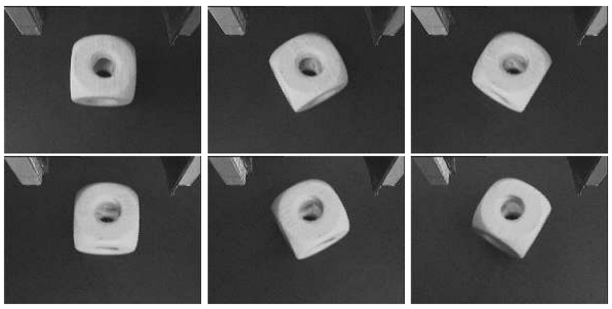


Figure 8. Six training images (192×144 pixels, with orientation variation $0°$, $30°$, $60°$, $90°$, $120°$, $150°$) for automatic grasping, taken from about 300 training images.

Even if the training data are not correlated at all, the projection in the eigenspace provides also information which input variables have larger variance. These variables can be good candidates for the inputs to be selected. Therefore, no “trial-comparison-select” procedure is necessary.

To generally deal with the high-dimensional input space, the solution based on the low-dimensional fuzzy controllers would need the partition of the complete high-dimensional input data set into clusters, within which the data are correlated to a large degree. Such a partition would be intrinsically fuzzy, since there are no crisp boundary between two continuous “situations”. A “behaviour arbiter” coordinates multiple simultaneously active local controllers to achieve a high-level task and can be realised with a set of meta-rules like: “IF $Situation\_Evaluation$ IS for_C_i THEN Apply Controller C_i.”

This idea was realised with mobile robots in blending the behaviours “approach a subgoal” and “collision-avoidance” [11]. Our future work is on further test of this approach in more global situations on vision-guided robot control as well as the fully automatic parameter adjustment of the complex thresher machine system.

References

[1] G. E. P. Box and G. M. Jenkins. *Time series analysis*. Holden Day, San Francisco, 1970.

[2] S. L. Chiu. Selecting input variables for fuzzy models. *Journal of Intelligent and Fuzzy Systems*, 4:243–256, 1996.

[3] J. Greenstadt. The determination of the chararcteristic roots of a matrix by the Jacobi method. *IBM Report*, 1961.

[4] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice Hall, 1997.

[5] V. Lacrose and A. Tilti. Fusion and hierarchy can help fuzzy logic controller designers. In *IEEE International Conference on Fuzzy Systems, Barcelona*, 1997.

[6] S. Lotvonen, S. Kivikunnas, and E. Juuso. Tuning of a fuzzy system with genetic algorithms and linguistic equations. In *Proceedings of Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen*, 1997.

[7] E. Oja. *Subspace methods of pattern recognition*. Research Studies Press, Hertfordshire, 1983.

[8] T. Sanger. *An optimality principle for unsupervised learning*. Advances in neural information processing systems 1. D. S. Touretzky (ed.), Morgan Kaufmann, San Mateo, CA, 1989.

[9] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems (forthcoming)*, 13(2/3):257–286, February/March 1998.

[10] J. Zhang, K. V. Le, and A. Knoll. Unsupervised learning of control spaces based on B-spline models, 1997.

[11] J. Zhang, F. Wille, and A. Knoll. Modular design of fuzzy controller integrating deliberative and reactive strategies. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, 1996.