Overfitting problem in a virtual sensor obtained with W–M method

Przemysław Andrzej Wałeґa*

Institute of Philosophy, University of Warsaw, 3 Krakowskie Przedmieście St., 00-927 Warszawa, Poland

Abstract

In the paper we will analyze how a virtual sensor may be obtained, by means of Wang and Mendel method of generating fuzzy Rule Base. In particular, we will analyze how the number of fuzzy sets influences on the method’s performance. We will state that increasing number of fuzzy sets leads to the overfitting effect, which will be compared to the overlearning effect known from Artificial Neural Networks. Afterwards, we will introduce an algorithm for overcoming overfitting problem in Wang–Mendel method. Finally, we will present a virtual sensor based on real industrial data and discuss its quality.

© 2014 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/).

Peer-review under responsibility of KES International.

Keywords: Wang–Mendel Method; Overfitting; Machine Learning; Virtual Sensor; Robustness

1. Introduction

Fuzzy logic is currently commonly used in a field of automatic control in numerous applications, e.g., to obtain fuzzy controllers or to create process models. Therefore, fuzzy systems are used not only in process controlling but also in fault diagnosing. Fuzzy logic became popular among theorists and is commonly applied in industrial engineering.

A main concept of fuzzy logic is a fuzzy set (introduced by Lotfi Zadeh in1), which is a set of pairs consisting of an element and its degree of being a member of a certain fuzzy set (i.e., a value of a membership function). Therefore, a fuzzy set A in a space of points X with a generic element x ∈ X is characterized by a membership function fa(x) which maps x into the interval [0, 1] and may be formally captured in a following way:

\[ A = \{(f_a(x), x) : x \in X\} \]  \hspace{1cm} (1)

Notice, that a membership function may have various shapes, but the most popular are triangular, trapezoidal and gaussian shapes. Fuzzy sets may be used to model linguistic concepts with meaning that seems not to have sharp
boundaries, e.g., "a tall person" or "a short person". Additionally, fuzzy sets may be used to model concepts known from automatic control, such as a positive error of a measured variable in a controller. Furthermore, fuzzy sets may be used to divide a space of interest into a number of intervals. A reader may find a more precise description of fuzzy logic, e.g., in 2.

In numerous automatic control systems it is important to create a model of a process, i.e., find a relation between input and output values. Since theoretical equations are hard (end expensive) to obtain, models are usually created, by means of previously registered data, i.e., using process history. Models obtained in such a way are automatically generated and may use big sets of data. One of the simplest fuzzy methods for generating such models is the Wang–Mendel method (W–M in short) which was originally presented in 3. W–M is a very popular method, therefore there are numerous modifications of W–M (e.g., see 4,5,6). It is interesting, that W–M is able to model any process with any accuracy (for details see 7). Nevertheless, our engineering experience convinces us, that data from process history is never ideal (measuring devices are imprecise and there is always some kind of noise in the data), therefore models obtained by means of registered data always contain error. What is interesting, is that model’s error may occur also due to a large model’s complexity. This effect is called overfitting and will be investigated in details in what follows.

We will present a W–M application for obtaining virtual sensor for industrial purposes. Our program is using real industrial data from 8 and contains an algorithm for overcoming overfitting problem. Our tests show a good performance of the obtained virtual sensor and convinces us, that it is appropriate for industrial applications.

Our paper is organized as follows. In Section 2 we will describe in details W–M method and indicate its main features. Then, in Section 3 we will discuss overfitting problem in W–M and compare it to a well–known overtraining problem in Artificial Neural Networks. Afterwards, in Section 4 we will present a virtual sensor that has been obtained with W–M together with an algorithm for overcoming overfitting problem. At the end, we will include concluding remarks in Section 5.

2. Wang–Mendel method

W–M is a simple method for generating a Rule Base (RB in short) from a combination of a set of input–output data and a set of expert linguistic rules. RB stores an aggregated knowledge from both of abovementioned data types and is in a form of a set of "IF–THEN" rules. W–M was originally presented in 3 and due to its good performance, quickly became a well-known tool which is currently widely used in fuzzy logic systems. Although, authors of W–M stress in 3 that their method is able to combine input–output data and expert linguistic rules, our engineering experience convinces us that expert linguistic rules are hardly available in most of the real industrial applications. Therefore, in what follows, we will consider only input–output data for RB generating. Notice, that using only input–output data enables us to make the procedure of generating RB automatic (it does not require expert intervention).

The algorithm for obtaining RB from input and output data, by means of W–M consists of 4 steps, namely, 1. divide the input and output spaces into fuzzy regions, 2. generate fuzzy rules from given data pairs, 3. assign a degree to each rule, 4. create a combined fuzzy RB. The procedure is as follows:

1. Divide each input space and each output space into an odd number of intervals. Select a type of a membership function (e.g., triangular or trapezoidal) and assign fuzzy sets to obtained intervals. The favorable division contains overlapping intervals.

2. Generate fuzzy rules, i.e., "IF–THEN" rules for input–output pairs. The antecedent of the rule states to which fuzzy sets belong input values and the consequent states to which fuzzy sets belong output values, e.g., "IF \( x_1 \) is in \( A_1 \) and \( x_2 \) is in \( A_2 \), then \( y_1 \) is in \( A_3 \) and \( y_2 \) is in \( A_4 \)”. Notice, that if fuzzy sets overlap, then one value may belong to few fuzzy set at the same time. Therefore, one input–output data pair may generate more than one "IF–THEN" rule (usually one input–output data pair generates a number of rules).

3. Assign an importance degree to each "IF–THEN" rule in order to find out which rules are most appropriate and should be contained in RB. The importance degree of a rule is a product of membership functions values of all inputs and outputs. Let us consider a model with \( m \) inputs, i.e., \( x_1, \ldots, x_m \), \( n \) outputs i.e., \( y_1, \ldots, y_n \), and fuzzy sets
denoted by $A_i$. Each rule may be represented in a following manner: "IF $x_1$ is $A_1$ and $x_2$ is $A_2$ and $x_3$ is $A_3$ then $y_1$ is $B_1$ and $y_2$ is $B_2$ and $y_3$ is $B_3$. " The importance degree $D$ of the abovementioned rule $R_k$ is as follows:

$$ D(R_k) = \mu_{A_1}(x_1) \cdot \ldots \cdot \mu_{A_n}(x_n) \cdot \mu_{B_1}(y_1) \cdot \ldots \cdot \mu_{B_m}(y_m) $$

where $\mu_{A_i}(x_i)$ is a value of a membership function associated with $A_i$ for $x_i$ argument.

4. Create RB that contains only rules with the highest importance degree, i.e., whenever there are two or more rules with the same antecedents, choose the one with a highest importance degree. After generating RB, importance degrees of rules are not used any more.

Input–output data used for generating RB is called a training data set. After obtaining RB, another set of input values — called a test data set — may be introduced to the system in order to calculate the output value. Let us consider a simple 2–input 1–output model. After generating RB, $(x_1, x_2)$ input from a test data set is presented to the system. The procedure of calculating output is as follows. At first, find out to which fuzzy sets $x_1$ and $x_2$ belong to. Then, for each rule $R_i$ from RB with antecedents corresponding to $x_1$ and $x_2$, determine the rule consequence $A_i$ (i.e., the fuzzy set to which the output should belong to). Calculate the degree $\mu'_{A_i}$ of the output membership to $A_i$ according to the $i$th rule, with the antecedents of a form: "IF $x_1$ is in $A_i$ and $x_2$ is in $A_i$". The degree value may be obtained, e.g., by means of Mamdani inference rule, i.e.:

$$ \mu'_{A_i} = \mu_{A_i}(x_1) \cdot \mu_{A_i}(x_2) $$

After calculating all $\mu'_{A_i}$, a defuzzification method needs to be chosen in order to calculate the output value. One of the simplest (and also very popular) defuzzification methods is a height method (HM in short). According to HM, the output value $y$ is as follows:

$$ y = \sum_{i=1}^{K} \frac{\mu'_{A_i} \cdot c_{A_i}}{\sum_{i=1}^{K} \mu'_{A_i}} $$

where $K$ is a number of fuzzy rules and $c_{A_i}$ is a center value of a region $A_i$.

It is proved\cite{3,7} that W–M (mapping function from input space into output space) is an universal approximator, i.e., it can approximate any real continuous function to any accuracy. The universal approximator feature will be discussed in Section 3 while investigating the overfitting effect (an universal approximator is sensitive to noise data, therefore overfitting problem may occur in W–M).

It is worth noticing, that W–M may be performed using various: number of fuzzy sets, their type (shape) and defuzzification method. Our experience convinces us that a number of introduced fuzzy sets has a major influence on the W–M performance and computational complexity. The commonly used engineering practice is to establish a same odd number of fuzzy sets from the interval $[3, 7]$, i.e., 3, 5 or 7 for all inputs and outputs. Abovementioned practice gives good results in numerous applications but it is not known what is the best number of fuzzy sets in general and how may it be calculated. Therefore, in what follows, we will investigate how exactly does a number of introduced fuzzy sets influence on the W–M method performance.

There are various modifications of W–M (see\cite{4,6,5}). They try to improve W–M performance, e.g., by means of changing a method of RB construction or introducing a method for fuzzyfication optimization. Abovementioned W–M modifications usually improve the system’s performance but, on the other hand, increase a number of calculations. Therefore, it is always a choice between better performance and simpler method.

As we have already stated, W–M enables automatic generation of RB from input–output data. We have implemented a program that analyzes input–output training data, generates RB and then, is able to calculate output values for testing data. Our program enables to generate models for any number of inputs and outputs, to define different number of fuzzy sets for various input and output spaces, to choose a shape of membership functions and to chose a defuzzification method. Additionally, training and testing data are plotted, model’s error is calculated and rules from RB may be viewed. Therefore, our program gives a great opportunity to test various modifications of W–M using real industrial data.
3. Overfitting

The overfitting effect is a well-known problem in machine learning. It occurs when a model too precisely fits to training data and as a result, there is a large model’s error (calculated with respect to testing data). The overfitting problem occurs commonly, while building a model, by means of noisy input–output data. If a model is too complex, it fits to noises included in training data. Therefore, we can prevent overfitting problem in two ways, namely, by removing noises from training data or by reducing model’s complexity.

If a training set contains real data (e.g., data from real industrial process) it always contain noise. Therefore, before creating RB, data need to be prepared. Although, there are methods for reducing noise it can never be completely removed. Therefore, although noise reduction is important, it cannot completely prevent from overfitting problem.

Another method for preventing overfitting problem is to select an appropriate structure of a model. Unfortunately, it is hard to find out what is the best structure in general. On the one hand, a model needs to be complex enough to model a given process, but on the other hand, it cannot be too complex in order to prevent from overfitting problem.

The overfitting is a well-known problem in a field of Artificial Neural Networks (ANN in short) and often called overtraining problem. Too long ANN training with noisy training data, results in an increase of a model’s error. It is usually treated by reducing training time of ANN (or reducing complexity of ANN, i.e., reducing a number of neurons in ANN structure). We will claim that W–M models behave similarly to ANN while considering overfitting problem. Too complex W–M models, i.e., models with too many fuzzy sets for input and output data spaces, may lead to overfitting. Analogously to ANN, reducing W–M model complexity, i.e., reducing number of fuzzy sets, prevents from overfitting.

In what follows, we will compare ANN and W–M training with artificially generated sinusoidal training data. We will show, how training performs when ideal (without noise) sinusoidal training data is used and what happens when training data contains noise (white noise). Training data consists of 126 input–output pairs of values, where input values are from the interval \([0, 2\pi]\) and output values are sinus values (with input values as arguments). Consequently, the model is 1–input, 1–output. Testing data is another 126 input–output pairs of ideal (without noise) sinusoidal function with inputs from the interval \([0, 2\pi]\). It is worth mentioning, that input values from testing data are from the same interval as input values from training data but mentioned sets are disjoint. Training and testing data sets are presented in Fig.1.

![Fig. 1: (a) ideal training data; (b) noisy training data; (c) testing data.](image)

W–M model was obtained by means of our program. We have used triangular membership functions, the same number of fuzzy sets for input and output values spaces and HM defuzzification method. While training ANN we have used Java Neural Network Simulator (JNNS in short)\(^9\) developed at the Wilhelm-Schickard-Institute for Computer Science in Tübingen. We have generated a simple feedforward artificial neural network with 2 layers; 1 input neuron, 10 hidden neurons and 1 output neuron. Detail information about training method in JNNS are as follows: Resilient Propagation method, with parameters: \(\delta_0 = 0.1, \delta_{\text{max}} = 50.0, \alpha = 4.0\). ANN structure used in further experiments is presented in Fig.2.

In order to evaluate model’s error we have used the following error measure \(E\):

\[
E = \sum_{i \in N} (y_i - \hat{y}_i)^2
\] (5)
where $\hat{y}_i$ is an output value obtained from the model after introducing $i$th input from testing data, $y_i$ is a known desired output value for $i$th input from testing data and $N$ is a number of input–output values in testing data (in our case $N = 126$).

In what follows, we will present experiments’ results for ANN and W–M trained with ideal sinusoidal training data, followed by ANN and W–M trained with noisy data. ANN was trained with 10 to 20000 epochs (where 1 epoch means presenting each member of training data set once to ANN). Experiments with W–M consist of lunching W–M algorithm with 3 to 200 fuzzy sets for input and output spaces.

At first, we will show how ANN and W–M perform while training data without noise is used. The increasing number of epochs in ANN leads to a better model obtaining. Since training data has no noise, ANN learns how to fit to ideal sinusoidal signal and even if the number of epochs is large, no overfitting problem occurs. Experiments’ results are presented in Fig.3. Notice, that in various parts of signal ANN models it with different accuracy. The first part of signal – for arguments from the interval $[0, \pi]$ is learned much faster than the second part. W–M training with training data without noise also gives better results while increasing the number of fuzzy sets. Experiments’ results are presented in Fig.4. Notice, that since fuzzy sets are evenly located in input and output spaces, W–M model has the same accuracy in all parts of the training data. Furthermore, while triangular membership functions are used, an achieved model of a signal is sharp (whereas model achieved with ANN is very smooth).

As a result, we can compare a model error as function of a number of epochs (in case of ANN) and as a function of a number of fuzzy sets (in case of W–M). In both cases we have noticed (nearly monotonic) decrease of model’s error. The comparison is presented in Fig.5.

In what follows, we will present experiments’ results for noisy training data. At first, we will show how ANN learns noisy signal. Experiments’ results are presented in Fig.6. While number of epochs is small, i.e., up to 3000, model becomes better while increasing a learning time (number of epochs). In this case ANN models the main sinusoidal signal. However, when a number of epochs is larger then 30000, we can notice the overfitting problem, i.e., ANN models not only the main sinusoidal signal but also the noise. W–M training with noisy training data also leads to overfitting – as presented in Fig.7. It is interesting to notice differences between overfitting that occurs in ANN and
W–M. When the number of epochs is well–chosen (30000 in our experiment) ANN models the main sinusoidal signal very precisely and hardly any influence of the noise in training data is noticed. On the other hand, overfitting in W–M is noticed almost immediately (9 fuzzy sets in our experiment). Consequently, there is no well–chosen number of fuzzy sets that results in obtaining as precise model as in case of ANN. Furthermore, as we have already stated, W–M method is an universal approximator. Therefore, it can model noisy training data with any accuracy, what can be seen in Fig.7 when number of fuzzy regions is big, e.g., 200.

Once more, we can compare a model’s error in case of ANN and W–M. In both cases we have noticed overfitting effect – the increase of model’s error. The comparison is presented in Fig.8.

4. Robust virtual sensor

A virtual sensor (also called a soft sensor) is a software program that calculates values of parameters that are not measured. Virtual sensors are commonly used in automatic control in order to decrease a cost of measurement devices or to obtain redundant measurements (duplication of measurement of critical parameters). In order to program
a virtual sensor, one needs to know equations that describe relations between parameters of interest. Unfortunately, in real industrial applications, theoretical equations usually are hard to obtain. Moreover, even if such equations are obtained, after few months of installation working, equations become outdated. Therefore, virtual sensors need to be often actualized. Hence, it is obvious that the best idea is to develop virtual sensors that automatically actualize themselves once in a while. W–M seems to be a great tool for establishing such a virtual sensor. Unfortunately, as we have showed – W–M is vulnerable to overfitting effect. In what follows, we will present a method for overcoming this problem. Our method of reducing overfitting effect will lead to obtaining a robust virtual sensor, i.e., a sensor which works correctly in different conditions.

The idea for overcoming overfitting effect is to generate a model with a minimal reasonable number of fuzzy sets (i.e., 3 fuzzy sets) and then systematically check the model’s error while increasing a number of fuzzy sets. While the number of fuzzy sets is small, model’s error may change drastically – it means that the model in not properly established. When drastic changes stop, we need to find a number of fuzzy sets that results in the smallest model’s error. Since the error not always changes monotonically, there may occur some local minimums. Therefore, the algorithm does not stop working if the model’s error starts increasing but when the model’s error is significantly greater than the hypothetical minimum. The algorithm is presented in Algorithm 1.

Since, W–M is very vulnerable to overfitting effect – it occurs even if a number of fuzzy sets is small – our algorithm usually does not need to perform more then 10 steps. On the other hand, while there are many inputs and outputs in the process, the time needed to generate RB is may become very long (it increases exponentially). Nevertheless, the actualization of RB is done once for a while and does not need to be done very fast.

In what follows, we will present a virtual sensor done, by means of our program and based on real industrial data. The virtual sensor is responsible for calculating juice flow in a pipe just after a valve with a pneumatic servo motor. Flow $F$ is calculated, by means of a measured value of pressure before the valve $P_1$, a measured value of pressure just after the valve $P_2$ and a servomotor rod displacement $X$. $F$ is in [m$^3$/h], $P_1$ and $P_2$ are in [kPa] whereas $X$ is in [%]. The installation scheme is presented in Fig. 9.

The installation comes from a real industry. Data, i.e., $F$, $P_1$, $P_2$ and $X$ were registered since 29th of October 2001 until 22th of November 2001. There are 86400 registered values (they have been registered every second) which are available in a benchmark published in 8.
Algorithm 1 Calculate optimal number of fuzzy sets $N_{best}$

\begin{align*}
N_{best} &= 0 \\
N &= 3 \\
& \text{calculate model's error } E_1 \text{ for } N \text{ fuzzy sets} \\
E_{min} &= E_1 \\
\text{while } N_{best} == 0 \text{ do} \\
& \quad N = N + 2 \\
& \quad \text{calculate model's error } E_2 \text{ for } N \text{ fuzzy sets} \\
& \quad \text{if } |E_2 - E_1| < 10\% \cdot \frac{|E_2 - E_1|}{2} \text{ then} \\
& \quad \quad \text{if } E_2 < E_1 \text{ then} \\
& \quad \quad \quad E_{min} = E_2 \\
& \quad \quad \quad N_{min} = N \\
& \quad \quad \text{else } \{E_2 \geq E_1\} \\
& \quad \quad \quad \text{if } E_2 - E_{min} > 10\% \cdot E_{min} \text{ then} \\
& \quad \quad \quad \quad N_{best} = N_{min} \\
& \quad \quad \text{end if} \\
& \quad \text{end if} \\
& \quad \text{end if} \\
& \quad E_1 = E_2 \\
\end{align*}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig9}
\caption{Installation scheme.}
\end{figure}

Theoretical relation between measured values is as follows:

$$F \sim X \cdot \sqrt{P_1 - P_2} \quad (6)$$

Therefore, our model is 2–input 1–output, where inputs are: $X$ and $\sqrt{P_1 - P_2}$, whereas $F$ is an output. We have generated and tested a virtual sensor, by means of, W–M method together with our algorithm for overcoming overfitting effect. Each of training and testing data consists of 1800 input–output disjoint pairs of values. An example of a model’s performance is presented in Fig. 10. The graph shows signal obtained from a model – plotted in red, and a desired signal from a testing data – plotted in blue.

An average error value $E$ obtained from 20 tests is about 3.5 which gives 0.8\% percentage error. We conclude, that our virtual sensor is precise enough to be used in industrial applications, e.g., for fault detection or as a measured variable in a controller.

5. Conclusion

Our experiments confirm that too large number of fuzzy sets in a W–M system results in the overfitting problem. In order to overcome the abovementioned effect we have proposed an algorithm, which may be used together with automatic generation of RB. Moreover, we have developed a program for automatic generating virtual sensors that may be useful in industrial applications or in a further investigation of W–M modifications. Our tests have confirmed that the obtained program has a good performance and a small modeling error. Additionally, it is simple and universal, i.e., it may be used for modeling processes with any number of inputs and outputs.
Our future work consists of developing a faster method for RB generation, which would be especially important for real–time applications working with big training data sets or for models with a large number of inputs and outputs. We are convinced that further improvements of W–M will lead to even wider range of industrial applications of this method.

Acknowledgements

The work presented in this paper is supported by the Polish National Science Centre grant 2011/02/A/HS1/00395.

References