A New Soft Sensor Modeling Method Based on Modified AdaBoost with Incremental Learning

Huixin Tian, Anna Wang, Zhizhong Mao

Abstract—Aiming at the characteristics of soft sensors, an ensemble learning algorithm AdaBoost.RT is used to establish the soft sensor models. According to the shortcomings of AdaBoost.RT and the difficulties of on-line updating for soft sensor models, a self-adaptive modifying threshold $\phi$ and an incremental learning method are proposed for improving the performance of original AdaBoost.RT. The new modified AdaBoost.RT can overcome the disadvantages of original AdaBoost.RT and update the soft sensor model in real time. The new method is used to establish the soft sensor model of molten steel temperature in 300t LF. Practical production data are used to test the model. The results demonstrate that the new soft sensor model based on modified AdaBoost.RT can improve the prediction accuracy and has good ability of update.

I. INTRODUCTION

In practical industrial process, there are a large number of data need be instrumented for process monitoring and control. But some of above data can not be measured accurately for complex production environment. Approximately two decades ago researchers started to predict these kind of data indirectly by building predictive models based on related data which can be measured accurately. In the context, these predictive models are called Soft Sensors [1]. This term is a combination of the words “software”, because the models are usually computer programs, and “sensors”, because the models are delivering similar information as their hardware counterparts.

At a very general level one can distinguish two different classes of soft sensors, namely model-driven and data-driven. Model driven models are also called white-box models because they have full mechanistic knowledge about the process background. In contrast to this purely, data-driven models are called black box techniques because the model itself has no knowledge about the process and is based on empirical observations of the process.

With the fast development of artificial intelligent technology, the data-driven soft sensor approaches based on intelligent algorithm have been widely applied in practical production process [2]. The most popular modeling techniques applied to data-driven soft sensors are the Principle Component Analysis (PCA) in a combination with a regression model, Partial Least Squares, Artificial Neural Networks, Neuro-Fuzzy Systems and Support Vector Machines (SVMs). However, these single algorithms have not satisfied the needs of complex industrial process. In addition, some limitations still exist in above algorithms, and they will affect the performance of soft sensors. Ensemble technique that combined the predictors is an efficient strategy for achieving high performance of prediction, especially in fields where the development of a powerful single predictor system requires considerable efforts, and has received much attention by researchers [3]. The combined predictor researches have shown higher correctness of predictability than any individual methods. The ensemble of predictors is often called ensemble or committee machine. In a committee machine, an ensemble of base predictors is firstly generated by means of applying a base learning algorithm to different distributions of the training data, and then the predictions from each ensemble member are combined suitably by ensemble method to predict a new example. In this paper, an ensemble algorithm for regression called AdaBoost.RT is used to establish soft sensor model for better performance.

Furthermore, during practical application process of soft sensors, it is hard to obtain a complete training data set which is a necessary factor to ensure the predictive accuracy. It attributes to the limitation of problem comprehension ability at the beginning of training, and the complexity of practical problems. Therefore, on-line update abilities are very important for soft sensor models. When new data comes, they are added into the data set for retraining. It is one of the most traditional update methods. This method has to waste lots of space and time, and limits the performance of soft sensor. There are some other traditional update methods: When new data comes, a part of old data are replaced by this new data. It is also called sliding windows. But the old information will disappear together with the old data. Some update methods for special problems such as PLS proposed by Helland, etc. [4]. These kinds of method are only satisfied the needs of special problems. So they are not generalized to the other problems.

According to the limitation of above update method, an improved method with incremental learning is proposed in this paper. Moreover, a self-adaptive modifying the value of threshold method is presented for overcoming the disadvantage of original AdaBoost.RT. Then a temperature soft sensor model of molten steel in LF is established using the modified AdaBoost.RT by production data. The experiment is done by different soft sensor using original and modified AdaBoost.RT. The results demonstrate that the soft sensor established by modified AdaBoost.RT has good performance and good ability of update.

H.X. Tian. Author is with the Electrical and Automation Engineering Department, Tianjin Polytechnic University, Tianjin, China (corresponding author to provide e-mail: iicedwl@163.com).

A.N. Wang and Z.Z. Mao is with the Information Science and Engineering Department, Northeastern University, Shenyang, China.
II. MODIFIED ADABOOST.RT

Boosting was used in classification problems at the earlier stage. Many researchers have proved that Boosting can solve the classification problems effectively. The boosting algorithms are used widely in the field of classification. However the researches about boosting algorithm in regression problems are much less than the one in classification problems. Schapire and Freund (1997) proposed the AdaBoost.R algorithm to generate regression models [6]. Breiman researched how to use the boosting in regression problems, and proposed the arc-gv algorithm in the same year [7]. Drucker (1997) developed the AdaBoost.R2 algorithm, which is an ad hoc modification of AdaBoost.R [8].

He conducted some experiments firstly for regression problems and obtained good results. Solomatine and Shrestha (2004) proposed the AdaBoost.RT algorithm to solve the regression problems more efficiently [9].

There are some characters in the AdaBoost.RT algorithm: Firstly, in AdaBoost.RT, the training examples are classified in two classes by comparing the accuracy of prediction with the pre-set relative error threshold. Secondly, the weight updating parameter not only ensures that the value of $\log(1/\beta)$ is non-negative, but also gives relatively more emphasis to the harder examples. Thirdly, the process is repeated until a preset number of machines are constructed or $\varepsilon_{t} < 0.5$ for classification or $\bar{\varepsilon}_{t} < 0.5$ for AdaBoost.R2. It should be noted that for AdaBoost.RT iterations do not stop when $\varepsilon_{t}$ is higher than 0.5. Owing to above characteristic, AdaBoost.RT gives the better performance in presence of noises. But there still exist some disadvantages, for instance, how to select threshold $\phi$ optimally.

Aiming at the limitation of AdaBoost.RT and the necessity of updating during the process of soft sensor, in this paper a modified AdaBoost.RT algorithm for regression is used to improve the performance of single ELM as ensembles method and establish the soft sensor model. The new method of self-adaptive modifying $\phi$ is proposed for improving the performance of AdaBoost.RT. And the incremental learning method are presented as an improved method for updating. The new modified AdaBoost.RT can overcome the disadvantages of original AdaBoost.RT and update the soft sensor model in real time.

A. AdaBoost.RT Algorithm

A new boosting algorithm for regression problems is described, AdaBoost.RT [9].

1) Input:
   Sequence of $m$ examples $(x_i, y_i), \ldots, (x_m, y_m)$, where $y \in R$
   Weak learning algorithm (Weak Learner)
   Integer $T$ specifying number of iterations (machines)
   Threshold $\phi$ (0< $\phi$ <1) for demarcating correct and incorrect predictions

2) Initialize:

Machine number or iteration $t = 1$
Distribution $D(i) = 1/m$ for all $i$
Error rate $\varepsilon_{t} = 0$
3) Iterate while $t \leq T$
   Call Weak Learner, providing it with distribution $D_{t}$
   Build the regression model: $f_{t}(x) \rightarrow y$
   Calculate absolute relative error for each training example as
   \[
   ARE_{t}(i) = \left| \frac{f_{t}(x_{i}) - y_{i}}{y_{i}} \right|
   \]
   Calculate the error rate of $f_{t}(x)$: $\varepsilon_{t} = \sum_{i \in \text{ARE}(i) > \phi} D_{t}(i)$
   Set $\beta_{t} = \varepsilon_{t}^{-\phi}$, where $n = 1, 2$ or 3 (linear, square, or cubic)
   Update distribution $D_{t}$
   \[
   D_{t+1}(i) = D_{t}(i) \times \begin{cases} \beta_{t} & \text{if } ARE_{t}(i) \leq \phi \\ 1 & \text{otherwise} \end{cases}
   \]
   Where $Z_{t}$ is a normalization factor chosen such that $D_{t+1}$ will be a distribution
   Set $t = t+1$
4) Output the final hypothesis:
   \[
   f_{m}(x) = \frac{\sum_{t} \left( \log \frac{1}{\beta} \right) f_{t}(x)}{\sum_{t} \left( \log \frac{1}{\beta} \right)}
   \]

The regression problem in AdaBoost.RT is projected into the binary classification problem while demarcating well-predicted and poorly predicted examples. However, unlike the methods with absolute big error margin, absolute relative error (ARE) is used to demarcate examples as either well or poorly predicted. If the ARE for any particular example is greater than $\phi$, the predicted value for this example is considered to be incorrect, otherwise it is correct. The numbers of correct and incorrect predictions are counted to calculate $\varepsilon_{t}$. Indeed, the machine attempts to find the $f_{t}(x)$ with small $\varepsilon_{t}$, which is possible only by reducing the ARE of the each example.

In contrast to the AdaBoost.R2 algorithm, AdaBoost.RT needs to optimally select threshold $\phi$. The experiments with the AdaBoost.RT have shown that the performance of the committee machine is sensitive to $\phi$ [9]. If $\phi$ is too low, then it is generally very difficult to get a sufficient number of correctly predicted example that are only few “hard” examples, which are often outliers, and intuitively these examples will get boosted. This will affect the performance of the committee machine seriously and may make it unstable. Consequently value of threshold has to be calibrated by more experiments. In order to estimate the values of $\phi$ efficiently, a new method to improve AdaBoost.RT is proposed in this paper.
B. Self-adaptive Modifying threshold \( \phi \)

A self-adaptive modifying threshold \( \phi \) method is used instead of the invariable \( \phi \). The value of \( \phi \) will decrease while the error \( e_i \) in current iteration is bigger than the error \( e_{i,1} \) in last iteration. Conversely, if the error \( e_i \) is smaller than the error \( e_{i,1} \), \( \phi \) will increase. Threshold \( \phi \) is adjusted during the whole procedure of iterations of AdaBoost.RT algorithm. This method can overcome the limitation of original AdaBoost.RT algorithm which is attribute to estimating threshold \( \phi \). In Solomatine’s research [9], conclusions have been drawn: the committee machine is stable while \( \phi \) is between 0 and 0.4, and at the value of 0.4, the AdaBoost.RT becomes unstable due to overfitting and the boosting of noise. Therefore \( 0 < \phi < 0.4 \) is defined at the beginning of AdaBoost.RT algorithm. Here we choose 0.2 as the default initial \( \phi \). The improved method of self-adaptive modified \( \phi \) is described: Calculate the root mean square error (RMSE) of output \( e \) in every iteration:

\[
e = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{4}
\]

Where, \( \hat{y}_i \) is the predicted temperature of molten steel by soft sensor. \( y_i \) is the real measured temperature of molten steel by thermocouple.

The value of \( \phi \) will decrease while \( e_i < e_{i,1} \), reversely the value of \( \phi \) will increase while \( e_i > e_{i,1} \). Moreover \( \phi \) will be invariant while \( |e_i - e_{i,1}| \geq d \), where \( d \) is a constant. The detail of change is shown as the following equations:

\[
\begin{align*}
\phi_{i+1} &= \phi \cdot \lambda & \text{if } e_i < e_{i,1} \\
\phi_{i+1} &= \phi \cdot (1 + \lambda) & \text{if } e_i > e_{i,1} \\
\phi_{i+1} &= \phi & \text{if } |e_i - e_{i,1}| \geq d
\end{align*}
\tag{5}
\]

Where \( \lambda \) is relative to the change rate of root mean square error:

\[
\lambda = r \frac{e_i - e_{i,1}}{e_i} \tag{6}
\]

The default value of \( r \) is 0.5, it can also be determined by users for different problems.

By using the above approach, users need not select the value of threshold by experiments anymore, and their time and energy will be saved. In section IV the experiments demonstrate the self-adaptive modifying threshold \( \phi \) method can ensure the good performance of AdaBoost.RT.

C. Incremental Learning Character

Aiming at the limitation of updating methods in soft sensor model, incremental learning is used in AdaBoost.RT for updating the soft sensor. After the last iteration, the weights \( D(i) \) for all \( i \) are calculated and saved. The old data with the least weight is replaced by new data. Then a new training data set is obtained. The weights of data are re-distributed. The \( e_i \) and predictors are saved. They are used to obtain the final hypothesis together with the old \( e_i \) and predictors. The new updating method of modified AdaBoost.RT is presented as follows:

1. Initialize the number of new data \( n \). When the number of new data cumulates up to \( n \), go to the following step:
2. The new training data set is obtained:
3. Go to step 3) in AdaBoost.RT. Iterate \( t = T + 1 \), \( t = T + 2 \), and compute the new \( e_i \) and weights.
4. Output the final hypothesis:

\[
f_{\text{fin}}(x) = \frac{\sum_{i=1}^{2n} \left[ \log \left( \frac{1}{\beta} \right) \cdot f_i(x) \right]}{\sum_{i=1}^{2n} \left[ \log \left( \frac{1}{\beta} \right) \right]} \tag{7}
\]

Figure 1 shows the updating process based on incremental learning for modified AdaBoost.RT.

During practical production process, the number of new production data will continuously increase with the continuous producing. Therefore, the ability of update is very important to soft sensor. The new modified AdaBoost.RT with updating can save the information of old predictors that have been trained, and need not save the original data. That is to say, only a few new production data being saved is enough. It is efficient to save a mass of space. Furthermore, the new improved method also may save a lot of time as compared with traditional updating method. It is attributed to the conservation of old \( e_i \) and predictors.

D. Weak Learning Machine

It is also important to choose a suitable weak learning
machine for ensemble algorithms. In this study, a new neural network algorithm extreme learning machine (ELM) for single-hidden layer feedforward neural networks (SLFNs) that was proposed recently by Guang-bin Huang is selected as the weak learning machine to construct an AdaBoost.RT soft sensor [10],[11]. In ELM, the input weights (linking the input layer to the hidden layer) and hidden biases are randomly chosen, and the output weights (linking the hidden layer to the output layer) are analytically determined by using Moore–Penrose (MP) generalized inverse. The main reason of selecting ELM reflects the following aspects. Firstly, a neural network is often viewed as a “universal approximator” [10]. That is, neural networks have the ability to provide flexible mapping between inputs and outputs. Secondly, the ELM algorithm looks much simpler than most learning algorithms for feedforward neural networks. Unlike traditional algorithms of neural networks, it does not need to calibrate the parameters (learning rate, learning epochs, etc) anymore. Therefore, the ELM algorithm can overcome the difficulties that the traditional classic gradient-based learning algorithms have to face, such as local minima, learning rate, learning epoch, stopping criteria and overfitting, etc. Thirdly, the learning speed of ELM is extremely fast. It is necessary as a weak learning machine of AdaBoost.RT to establish a soft sensor model [11].

III. TEMPERATURE OF MOLTEN STEEL SOFT SENSOR MODEL IN LF

In this section, a temperature of molten steel soft sensor model is established using above modified AdaBoost.RT for 300t LF (Ladle Furnace) in Baoshan Iron & Steel Co. Ltd. Firstly, the main factors that affect the temperature are used as inputs of soft sensor model by analyzing the energy change during whole LF refining process. ELM is selected as kernel intelligent algorithm. The modified AdaBoost.RT is used to ensemble the ELM, and then the molten steel temperature soft sensor is established.

A. Obtain the inputs of soft sensor model

For establishing the intelligent temperature soft sensor model of molten steel, the whole LF refining process is considered as an energy conservation system. Traditionally, the metallurgical process of LF is from the ladle entry to the ladle exit [12],[13]. However, in practical refining process, the temperature is not measured every time at the beginning of the process. The measurement may be done after the power turned on. The energy change from ladle entry to the power turned on is neglected. Therefore, in order to ensure the balance of energy in whole system, we choose the time of the last temperature being measured before power turned on as the start time of the conservation system, and this temperature is the initial temperature. Similarly, the ending time is the time of end temperature being measured. That is to say, the LF refining process is from the beginning time to the ending time mentioned above in the energy conservation system (Fig. 2). The energy change during above refining process is considered to establish the intelligent model. By analyzing the thermodynamics and conservation of energy during the metallurgical process, the energy gain of LF is mainly due to the electric arc, and the energy loss mainly includes the following three sections: the first section is the heat exchanges between ladle furnace and surroundings, which include the energy loss of the ladle refractory wall and energy loss from the top surface. The energy loss in this section is relatively stable. It will increase with time. Therefore this energy loss may be reflected by refining time. The second section is the energy change for additions, which includes the sum of heat exchanges and chemical reaction heats. They can be calculated by the parameters of temperature change for various metal alloy and slag addition in 300t LF. The third section is the energy loss by argon purging. To sum up, the main factors that affect the temperature can be obtained, which are the refining power consumption, the initial temperature, the ladle states, the heat effects of additions, the volume of argon purging, the weight of molten steel and the refining time.

B. Establishing the temperature soft sensor model of molten steel

On the basis of above analysis, the inputs of basic ELM prediction model can be determined. They are the 6 factors that mainly affect the temperature of molten steel in LF. The output of basic ELM prediction model is the temperature of molten steel. There are 9 hidden nodes assigned for ELM model according to the empirical formulation. The iteration (the number of weak machines) of AdaBoost.RT is 8. Therefore, a temperature soft sensor model of molten steel is established.

IV. EXPERIMENT

Three hundred and eighty data of production from 300t LF in Baoshan Iron & Steel Co. Ltd. from June to November in 2006 are used to test the new temperature soft sensor model of molten steel in LF based on new modified AdaBoost.RT modeling method. 50 data are randomly selected from this data set as testing data. According to the refining order, the first 300 data of rest data are selected as training data, and the other 30 data are used as updating data. The initial iteration \( T \) is 8. The number of new data \( n \) is 15. The experiments are performed by original AdaBoost.RT and proposed modified AdaBoost.RT with self-adaptive modifying \( \phi \) and...
incremental learning using above production data. The experiment results are analyzed by different performance of different modeling method.

Figure 3~figure 5 show the results of experiments. Figure 3 shows the performance of temperature soft sensor using original AdaBoost.RT. Figure 4 shows the predicted temperature by the soft sensor using modified AdaBoost.RT with self-adaptive modifying $\phi$ before updating. Figure 5 shows the comparison of real measured temperature and predicted temperature by soft sensor after updating using modified AdaBoost.RT. The results of experiment demonstrate the improved method of self-adaptive modifying threshold $\phi$ can enhance the performance of soft sensor with AdaBoost.RT. The temperature soft sensor model with incremental learning has the good ability of update. The predicted temperature by soft sensor after updating is better than the one before updating.

The comparison of performance of the temperature soft sensor model using original AdaBoost.RT and the model using modified AdaBoost.RT can be obtained by calculating the prediction accuracy and RMSE as table I shown. The accuracy is calculated as follows:

$$\text{accuracy} = \frac{N_u}{N_w}$$  \hspace{1cm} (8)

Where, $N_u$ is the number of heats with error $< 5$ °C, $N_w$ is the number of whole testing heats. The RMSE is calculated as equation (4). According to the comparison, we can draw a conclusion that the modified methods can improved the performance of AdaBoost.RT effectively. Moreover, the new temperature soft sensor of molten steel based on modified AdaBoost.RT has good ability of updating. The above mentioned problems of update in soft sensor methods are solved successfully by this new method. The new soft sensor method can be used widely in practical predicting process for its good ability of updating.

<table>
<thead>
<tr>
<th>COMPARISON OF PREDICTION PERFORMANCES OF DIFFERENT SOFT SENSOR MODELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature soft sensor of molten steel based on AdaBoost.RT</td>
</tr>
<tr>
<td>Traditional AdaBoost.RT</td>
</tr>
<tr>
<td>Modified AdaBoost.RT before updating</td>
</tr>
<tr>
<td>Modified AdaBoost.RT after updating</td>
</tr>
</tbody>
</table>

V. CONCLUSION

In this study, an ensemble algorithm AdaBoost.RT is selected to establish soft sensor model, according to the characters of soft sensor and the problems in practical application. Aiming at the disadvantage of traditional AdaBoost.RT, a self-adaptive modifying threshold $\phi$ method is proposed to improve the performance of AdaBoost.RT. Furthermore, in order to solving the on-line update problems better during practical application of production process, an improved method with incremental learning is presented. A temperature soft sensor model of molten steel in LF is established by using the modified AdaBoost.RT in Baoshan Iron & Steel Co. Ltd. According to analyzing the energy change during whole LF refining process, the main actors that affect the temperature are obtained and used as the inputs of soft sensor model. ELM is selected as kernel intelligent algorithm for its especial characters. The experiments are done using production data for different soft sensor model by original AdaBoost.RT and modified AdaBoost.RT. The comparison demonstrates that the soft sensor model with modified AdaBoost.RT has the better performance than the one with original AdaBoost.RT,
and good ability of on-line updating. The new method might be used in more extensive field of soft sensor.

Appendix: Review of ELM

A. Single Hidden Layer Feedforward Networks (SLFNs) with Random Hidden Nodes

For \( N \) arbitrary distinct samples \( \{(x_j,t_j)\}_{j=1}^{N} \), where \( x_j = [x_{j1}, x_{j2}, \ldots, x_{jm}]^T \in \mathbb{R}^m \) and \( t_j = [t_{j1}, t_{j2}, \ldots, t_{jm}]^T \in \mathbb{R}^m \), a standard SLFNs with \( \tilde{N} \) hidden nodes and activation function \( g(x) \) are mathematically modeled as

\[
\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = o_j, \quad j = 1, \ldots, N \quad (1)
\]

where \( w_i = [w_{i1}, w_{i2}, \ldots, w_{im}]^T \) is the weight vector connecting the ith hidden node and the input nodes, \( \beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T \) is the weight vector connecting the ith hidden node and the output nodes, \( o_j = [o_{j1}, o_{j2}, \ldots, o_{jm}]^T \) is the output vector of the SLFN, and \( b_i \) is the threshold of the ith hidden node. \( w_i \cdot x_j \) denotes the inner product of \( w_i \) and \( x_j \). The output nodes are chosen linear. The standard SLFNs with \( \tilde{N} \) hidden nodes with activation function \( g(x) \) can approximate these \( N \) samples with zero error means that \( \sum_{j=1}^{N} \| o_j - t_j \| = 0 \). And these \( N \) equations can be written compactly as:

\[
H \beta = T \quad (2)
\]

where

\[
H = \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}})
\end{bmatrix}_{N \times \tilde{N}}
\]

\[
\beta = \begin{bmatrix}
\beta_1 \\
\vdots \\
\beta_{\tilde{N}}
\end{bmatrix}_{\tilde{N} \times 1}
\]  

and

\[
T = \begin{bmatrix}
t_1 \\
\vdots \\
t_N
\end{bmatrix}_{N \times 1}
\]

Here \( H \) is called the hidden layer output matrix.

B. ELM algorithm

In most case, the number of hidden nodes is much less than the number of training samples ( \( \tilde{N} << N \) ). This means that \( H \) is a nonsquare matrix and there may not exist \( w_i, b_i, \beta (i = 1, \ldots, \tilde{N}) \) such that \( H \beta = T \). The parameters of hidden nodes need not be tuned and can be randomly generated permanently according to any continuous probability distribution. The unique smallest norm least squares solution of the above linear system is

\[
\hat{\beta} = H^+ T
\]

(6)

Where \( H^+ \) is the Noore-Penrose generalized inverse of matrix \( H \).

Thus, a simple learning method for SLFNs called extreme learning machine (ELM) can be summarized as follows:

Step 1: Randomly assign input weight \( w_i \) and bias \( b_i \), \( i = 1, \ldots, \tilde{N} \).

Step 2: Calculate the hidden layer output matrix \( H \).

Step 3: Calculate the output weight \( \beta = H^+ T \).

Acknowledgment

This research is supported by National Natural Science Foundation of China (Grant No. 60843007).

References


