A Soft Sensor Modelling of Biomass Concentration during Fermentation using Accurate Incremental Online ν-Support Vector Regression Learning Algorithm

Binjie Gu and Feng Pan

Key Laboratory of Advanced Process Control for Light Industry, Ministry of Education, Jiangnan University, Wuxi, China

Abstract: In order to model real fermentation process, a soft sensor modelling of biomass concentration during fermentation using accurate incremental online ν-Support Vector Regression (ν-SVR) learning algorithm was proposed. Firstly, an accurate incremental online ν-SVR learning algorithm was proposed. This algorithm solved the two complications introduced in the dual problem based on the equivalent formulation of ν-SVR. Moreover, it addressed the infeasible updating path problem during the adiabatic incremental process by relaxed adiabatic incremental adjustments and accurate incremental adjustments. Then, the proposed algorithm is used to predict the biomass concentration of glutamic acid fed-batch fermentation process online. The results of simulation experiment showed that the soft sensor modelling of biomass concentration during fermentation using the proposed algorithm was of better generalization ability and cost less training time than that of ν-SVR.

Keywords: Soft Sensor Modelling, ν-Support Vector Regression, Online Learning, Biomass Concentration, Fed-Batch Fermentation

Introduction

The optimal control of fermentation process plays an important role in microbial fermentation, which concerns the success or failure of industrial production (Wei and Yang, 2008). According to different process flow, fermentation process can be partitioned into three categories: Batch fermentation, continuous fermentation and fed-batch fermentation (Shi and Pan, 2010). Fed-batch fermentation has successfully solved problems such as substrate inhibition, strain degeneration and contamination existed in batch fermentation or continuous fermentation. Moreover, the optimal control of fermentation process can be realized easily. Therefore, fed-batch fermentation has been widely used in industrial fermentation in recent years.

It is well known that fermentation process is time-variant and nonlinear (Chen and Li, 2002; Yu, 2011). Therefore, in order to realize the optimal control of fermentation process, it is necessary to master the state information of fermentation process timely. Unfortunately, some important variables, including biomass concentration, product concentration or substrate concentration are rather difficult to be measured online in real fermentation process. This seriously influences the implementation of optimization strategy (Yoshida et al., 1973). To address this issue, the most widely used approach is soft sensor modelling based on training samples. The idea of soft sensor modelling is to infer or estimate some important variables which cannot or rather difficult to be measured by establishing mathematical relations based on other known or easily measurable variables. Therefore, to some extent, the soft sensor modelling can replace the function of hardware.

Nowadays, many offline soft sensor modelling algorithms had been proposed and achieved remarkable results. Petrova et al. (1998) realized the modelling of biomass growth based on Artificial Neural Network (ANN), the chief drawback of this method was that large amount of samples were required to train ANN, which was unsuitable for small sample learning scenarios. Feng et al. (2004) proposed a soft sensor model of the Box-Jenkins gas furnace and FCCU based on weighted Support Vector Machine (SVM). Gao et al. (2006) proposed a soft sensor model of the penicillin fermentation process based on SVM. However, once the offline model was established, if the model parameters need to be tuned, we must wait until the whole production process was over, which caused re-training...
the model from scratch. Therefore, they were unsuitable for real production process.

To overcome this problem, Wang et al. (2009) proposed a soft sensor modelling of biotechnical process based on online ϵ-Support Vector Regression (ε-SVR) and achieved better results than those of offline algorithms. Compared with offline algorithms, the model parameters of online algorithms can be tuned online, which is more suitable for real fermentation process. Unfortunately, it is rather difficult to select an appropriate C. To address this issue, Schölkopf et al. (2000) proposed the v-SVR, which uses a new parameter v to replace the parameter C. Moreover, it is easier to tune parameter v than C. However, compared with the dual problem of ϵ-SVR, two complications are introduced in v-SVR. The first one is that the box constraints are related to C and the length of the training samples and the second one is that one more inequality constraint is introduced (Gu et al., 2015). Moreover, as proved in Gu et al. (2012), it will not guarantee that a feasible updating path can always be generated. To sum up, it is rather difficult to design an online v-SVR learning algorithm.

In this study, an accurate incremental online v-SVR learning algorithm is proposed to address the problems mentioned above. Then, the proposed algorithm is applied in the soft sensor modelling of glutamic acid fed-batch fermentation process. Finally, compared with v-SVR, the results of modelling are analyzed to show the superiority of the proposed algorithm.

**Accurate Incremental Online v-SVR Learning**

To make the symbols easier to follow, we give a summary of the symbols at the end of this article.

**Equivalent Formulation of v-SVR**

Given a training sample set \( F = \{ (x_i, y_i) \} \) such that \( x_i \in \mathbb{R}^d \) is an input and \( y_i \in \mathbb{R} \), \( i = 1, \ldots, l \) is a target output, the primal problem of v-SVR is (Chang and Lin, 2002):

\[
\min_{w, b, \xi, \xi^*} \frac{1}{2} w^T w + C \left( \nu \epsilon + \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) \right)
\]

\[\text{s.t.} \quad (w^T \phi(x_i) + b) - y_i \leq \epsilon + \xi_i, \quad y_i - (w^T \phi(x_i) + b) \leq \epsilon + \xi_i^*, \quad \xi_i, \xi_i^* \geq 0, i = 1, \ldots, l, \epsilon \geq 0\]  

(1)

Where:

- \( w \) = The weight column vector
- \( b \) = Bias
- \( \xi_i \) and \( \xi_i^* \) = Nonnegative slack variables
- \( l \) = The length of training samples

The training vectors \( x_i \) are mapped into a high dimensional Reproducing Kernel Hilbert Space (RKHS) by the transformation function \( \phi \). The ϵ-insensitive loss function means that if \( w^T \phi(x_i) + b \) is in the range of \( y_i \in \mathbb{R} \), no loss is considered. \( v \) is the introduced new proportion parameter with \( 0 \leq v \leq 1 \), which lets one control the number of support vectors and errors. To be more precise, \( v \) is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors. In addition, with probability 1, asymptotically, \( v \) equals to both fractions (Gu et al., 2015). Therefore, it is easier to tune parameter \( v \) than \( \epsilon \)-SVR.

The dual problem of Equation 1 is (Chang and Lin, 2002):

\[
\min_{\alpha, \alpha^*} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i - \alpha_i^*) \alpha_j - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) y_i
\]

\[\text{s.t.} \quad \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0, \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) \leq C, \quad 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{l}, i = 1, \ldots, l\]  

(2)

where, \( H_y = K(x, x_i) = \phi^T(x) \phi(x_i) \); \( K \) is the kernel function.

Compared with the dual problem of stand \( \epsilon \)-SVR (Chang and Lin, 2002), it is clear that two complications are introduced in the dual problem of v-SVR. The first one is that the box constraints \( 0 \leq \alpha_i, \alpha_i^* \leq C/l \) are related to \( C \) and \( l \), the second one is that Equation 2 has an extra inequality constraint.

To solve the first complication, we multiply the objective function \( P \) in Equation 1 by the length of training samples and consider the following primal problem:

\[
\min_{\nu, \alpha, \alpha^*, \epsilon} \frac{l}{2} w^T w + C \left( \nu \epsilon + \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) \right)
\]

\[\text{s.t.} \quad \left( w^T \phi(x_i) + b \right) - y_i \leq \epsilon + \xi_i, \quad y_i - \left( w^T \phi(x_i) + b \right) \leq \epsilon + \xi_i^*, \quad \xi_i, \xi_i^* \geq 0, i = 1, \ldots, l, \epsilon \geq 0\]  

(3)

It is easy to verify that Equation 3 is equivalent to Equation 1. The corresponding dual problem of Equation 3 is:

\[
\min_{\alpha, \alpha^*, \epsilon} \frac{l}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i - \alpha_i^*) \alpha_j - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) y_i
\]

\[\text{s.t.} \quad \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0, \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) \leq C, \quad 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{l}, i = 1, \ldots, l\]  

(4)
where, \( P_{ij} = H_{ij}/l \) and \( y_i' = y_i/C \).

Furthermore, we can solve the second complication based on Lemma 1.

**Lemma 1** For any given \( \nu \) in Equation 4, if \( 0 \leq \nu \leq 1 \), there are always optimal solutions which happen at the equality \( (\alpha_i + \alpha_i^*) = \nu l \).

The detailed proof of Lemma 1 can be found in Chang and Lin (2002), it is omitted here.

Based on Lemma 1, we consider the following dual problem instead of Equation 4:

\[
\begin{align*}
\min_{\alpha_i, \alpha_i^*} D &= \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i - \alpha_i^*) p_i (\alpha_j - \alpha_j^*) + \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) y_i' \\
\text{s.t.} \quad \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) &= 0, \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) = \nu l \\
0 &\leq \alpha_i, \alpha_i^* \leq 1, i = 1, \ldots, l
\end{align*}
\]

In order to make Equation 5 more compact, let \( z_i \) denotes the label of the training sample \((x_i, y_i')\), we define the expanded training sample set \( T \), which is defined as:

\[ T = T^* \cup T^- = \{ (x_i, y_i', z_i = +1)^{(1)\ldots(l)} \} \cup \{ (x_i, y_i', z_i = -1)^{(1)\ldots(l)} \} \]

Then, Equation 5 can be rewritten as:

\[
\begin{align*}
\min_{\alpha_i} D &= \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} Q_i Q_j \alpha_i \alpha_j + \sum_{i=1}^{l} z_i \alpha_i y_i' \\
\text{s.t.} \quad \sum_{i=1}^{l} z_i \alpha_i &= 0, \sum_{i=1}^{l} \alpha_i = \nu l, \quad 0 \leq \alpha_i \leq 1, i = 1, \ldots, 2l
\end{align*}
\]

where, \( Q_{zi} = z_i \phi(\|x\|) \).

According to the convex optimization theory (Bertsekas, 2009), the solution of Equation 6 can be obtained by minimizing the following convex quadratic objective function under constraints:

\[
\begin{align*}
\min_{\alpha_i} W &= \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i Q_i \alpha_j + \sum_{i=1}^{l} z_i \alpha_i y_i' \\
&+ b \left( \sum_{i=1}^{l} \alpha_i \right) + \rho \left( \sum_{i=1}^{l} \alpha_i - \nu l \right)
\end{align*}
\]

where, \( b \) and \( \rho \) are Lagrange multipliers.

Optimizing Equation 7 leads to the following Karush_Kuhn_Tucker (KKT) conditions (Karush, 1939):

\[
\begin{align*}
\frac{\partial W}{\partial \alpha_i} &= Q_i y_i' + \nu \alpha_i - z_i b - \rho = 0 \quad \forall \alpha_i \in S \\
> 0 &\quad \alpha_i = 0 \quad \forall \alpha_i \in R \\
< 0 &\quad \alpha_i = 1 \quad \forall \alpha_i \in E
\end{align*}
\]

Based on the value of \( g_i \), the expanded training sample set \( T \) can be partitioned into three independent sets, which is shown in Fig. 1.

(a) The set \( S \) including margin support vectors strictly on the margin, for convenience, the numbers of training data points in the set \( S \) is donated as \( r \); (b) The set \( E \) including error support vectors exceeding the margin; (c) The set \( R \) including the remaining vectors covered by the margin.

![Fig. 1. The partition of the expanded training sample set T into three independent sets: (a) the set S; (b) the set E; (c) the set R](image-url)
Accurate Incremental Online ν-SVR Learning Algorithm

The first accurate incremental online ε-support vector learning algorithm was proposed by Cauwenberghs and Poggio (2001) (further referred to as a C&P algorithm). Unfortunately, as proved in Gu et al. (2012), when the C&P algorithm is directly applied to the equivalent formulation of ν-SVR, it will not guarantee that a feasible updating path can always be generated. To address this issue, an accurate incremental online ν-SVR learning algorithm (it is called AIOSVR learning algorithm) is proposed in this section.

AIOSVR Learning Algorithm:

The AIOSVR learning algorithm is presented in Algorithm 1.

Algorithm 1 The AIOSVR learning algorithm: High-level summary.

Inputs: The new candidate sample \((x_i, y_i)\) and \(α_i, i ∈ S\).

Outputs: Updated \(α_i, i ∈ S\).

1: Read the new candidate sample \((x_i, y_i)\) to obtain \((x_i, y_i')\), let \(T_{\text{new}} = \{(x_i, y_i', +1), (x_i, y_i', -1)\}\) and set the weights of the samples in \(T_{\text{new}}\) as \(α_i = 0\) and compute \(g_i, i ∈ T_{\text{new}}\) according to Equation 8.

2: Update \(T ← T ∪ T_{\text{new}}\), \(g ← \frac{1}{l+1}g_i, i ∈ T\), \(b ← \frac{1}{l+1}b\)

3: Compute \(N\) based on \(N\).

4: do // see RAIA

5: Compute \(δ_i, ζ_i, i ∈ R ∪ E\) and \(Δα_{\text{new}}\).

6: Update \(α_i, i ∈ S, g_i, i ∈ R ∪ E\), the sets \(S, R\) and \(E\).

7: Update the inverse matrix \(N\).

8: while \(g ≤ 0\) and \(α < 1\)

9: Compute \(N\) based on \(N\).

10: do // seeARA

11: Compute \(δ_i, ζ_i, i ∈ R ∪ E\) and \(Δη_i\).

12: Update \(α_i, i ∈ S, g_i, i ∈ R ∪ E\), the sets \(S, R\) and \(E\).

13: Update the inverse matrix \(N\)

14: while \(Σ_i α_i ≠ ν(l+1)\)

15: Ready for the next new candidate sample, go back to step 1.

RAIA:

During the adiabatic incremental adjustments for \(α_i\), in order to keep all the training samples satisfying the KKT conditions, based on Equation 8 to 10, we have:

\[Δg_i = \sum_{j \in S} Δα_j Q_{ij} + Δα_j Q_{ij} + z_i Δb + Δρ = 0, ∀ i ∈ S\]

\[\sum_{j \in S} z_i Δα_j + z_i Δα_j = 0\]

\[\sum_{j \in S} Δα_j + Δα_j = ν\]

Initially, \(α_i\) is set as \(α_i = 0\). If the samples in the set \(S\) has the same \(z_i\) and \(z_{i'}\) of the added new candidate sample is the same as \(z_{i'}\), then Equation 12 is changed into:

\[\sum_{j \in S} Δα_j + Δα_j = 0\]

If \(ν \neq 0\), it is obvious that Equation 13 and 14 cannot hold simultaneously. We call this the contradiction 1, which is shown in Table 1.

The existence of contradictions will cause the fact that \(α_i\) cannot be adjusted effectively. To address this issue, we first give up Equation 10 and then a strategy is utilized to restore Equation 10. Giving up Equation 10 means \(Δρ = 0\), therefore, referring to C&P algorithm, this step is called Relaxed Adiabatic Incremental Adjustments (RAIA).

Define \(z_i = [z_i, …, z_{i'}]^T\) and \(Δα_i = [Δα_i, …, Δα_i]^T\), then Equation 11 and 12 can be further rewritten as the following matrix form:

\[
\begin{bmatrix}
0 & z_i^T \\
\hat{Q}_{\hat{S}} & \hat{Q}_{\hat{S}}^T
\end{bmatrix}
\begin{bmatrix}
Δb \\
Δα_i
\end{bmatrix}
= -\begin{bmatrix}
z_i \\
\hat{Q}_{\hat{S}}^T
\end{bmatrix}
Δα_i
\]

Let \(N = M^{-1}\) and then we have:

\[
\begin{bmatrix}
Δb \\
Δα_i
\end{bmatrix}
= M^{-1}
\begin{bmatrix}
z_i \\
\hat{Q}_{\hat{S}}^T
\end{bmatrix}
Δα_i
\]

where, \(\gamma_i\) stands for the dimension corresponding to \(b\) in the column vector \(γ\) and it is similar for \(\gamma_{i'}\).

Finally, substituting Equation 16 into 11, we have:

\[
\Delta g_i = \left(\sum_{j \in S} γ_i' Q_{ij} + Q_{ij} + z_i γ_{i'}\right) Δα_i = γ_i' Δα_i, ∀ i ∈ T
\]
Let $\tilde{N} = \tilde{M}^{-1}$ and then we have:

$$
\begin{bmatrix}
\Delta b \\
\Delta \rho \\
\Delta \alpha_s \\
\end{bmatrix} = -\tilde{N} \begin{bmatrix}
0 \\
-1 \\
1 \\
\end{bmatrix} \cdot \Delta \eta = \begin{bmatrix}
\tilde{\gamma}_s \\
\tilde{\gamma}_s \\
\tilde{\gamma}_s \\
\end{bmatrix} = \begin{bmatrix}
\hat{\gamma}_s \\
\hat{\gamma}_s \\
\hat{\gamma}_s \\
\end{bmatrix} \cdot \Delta \eta
$$

(23)

where, $\hat{\gamma}_s$ stands for the dimension corresponding to $b$ in the column vector $\gamma$ and it is similar for $\hat{\gamma}_s$ and $\hat{\gamma}_s$.

Finally, substituting Equation 23 into 18, we have:

$$
\Delta g = \left( \sum_{j \in S} \hat{\gamma}_s Q_{ij} + z \hat{\gamma}_s + \hat{\gamma}_s \right) \cdot \Delta \eta = \hat{\zeta}_s, \Delta \eta, \forall i \in T
$$

(24)

It is obvious that $\hat{\zeta}_s = 0, \forall i \in S$. From Equation 21 and 23, we have $\sum_{j \in S} \Delta \alpha_s = -(\beta \hat{\gamma}_s + 1) \Delta \eta$, which means the adjustment of $\sum_{j \in S} \Delta \alpha_s$ can be achieved by $\Delta \eta$.

**Computing the Critical Adjustment Quantity $\Delta \eta^*$:**

The critical adjustment quantity $\Delta \eta^*$ is computed to ensure that only a training sample will migrate among the sets $S$, $E$ and $R$. This strategy can address the problem when directly apply the ARA to obtain the new optimal solution of Equation 5. If $\Sigma_{i \in S} \alpha_i > \nu(I+1)$, compute the maximal adjustments $\Delta \eta^*$ and let $\Delta \eta^* = \Delta \eta_{\text{max}}$, if $\Sigma_{i \in S} \alpha_i < \nu(I+1)$, compute the minimal adjustments $\Delta \eta^*$ and let $\Delta \eta^* = \Delta \eta_{\text{min}}$. Three cases should be considered to account for such structural changes.

**Case 1:** A certain training sample migrates from the set $S$ to the set $E$ or the set $R$.

When $\Sigma_{i \in S} \alpha_i > \nu(I+1)$, the possible weight updates are:

$$
\Delta \alpha_{i} = \begin{cases}
-1 - \alpha_i, & i \in I'_s \\
-1, & i \in I'_s
\end{cases}
$$

where, $I'_s = \{ \hat{\gamma}_s > 0, \forall i \in S \}$ and $I'_s = \{ \hat{\gamma}_s < 0, \forall i \in S \}$.

Thus the maximal possible $\Delta \eta_{\text{max}}$ is:

$$
\Delta \eta_{\text{max}} = \min(\Delta \alpha_{i} / \hat{\gamma}_s, \forall i \in I'_s \cup I'_s)
$$
When \( \Sigma_{i \in S} \alpha_i < \nu(l+1) \), the possible weight updates are:

\[
\Delta \alpha_i^\text{max} = \begin{cases} 
- \alpha_i, & i \in I^c_s \\
1 - \alpha_i, & i \in I_s 
\end{cases}
\]

Thus the minimal possible \( \Delta \eta^\text{con1} \) is:

\[
\Delta \eta^\text{con1} = \max(\Delta \alpha_i^\text{max}, \forall i \in I_s \cup I^c_s)
\]

**Case 2:** A certain training sample migrates from the set \( E \) or the set \( R \) to the set \( S \).

When \( \Sigma_{i \in S} \alpha_i > \nu(l+1) \), the maximal possible \( \Delta \eta^\text{con2} \) is:

\[
\Delta \eta^\text{con2} = \min(-g_i/\hat{\eta}_i, \forall i \in I_s \cup I^c_s)
\]

where, \( I^c_s = \{z_i, \alpha_i > 0, \forall i \in E\} \) and \( I_s = \{z_i, \alpha_i < 0, \forall i \in R\} \).

When \( \Sigma_{i \in S} \alpha_i > \nu(l+1) \), the minimal possible \( \Delta \eta^\text{con1} \) is:

\[
\Delta \eta^\text{con1} = \max(-g_i/\hat{\eta}_i, \forall i \in I_s \cup I^c_s)
\]

where, \( I_s = \{z_i, \alpha_i > 0, \forall i \in E\} \) and \( I^c_s = \{z_i, \alpha_i < 0, \forall i \in R\} \).

**Case 3:** When \( \Sigma_{i \in S} \alpha_i = \nu(l+1) \), which means the termination condition is met, then the critical adjustment quantity \( \Delta \eta^\text{con3} \) in case 3 is:

\[
\Delta \eta^\text{con3} = \sum_{i \in S} \alpha_i - \nu(l+1) = 0
\]

Finally, if \( \Sigma_{i \in S} \alpha_i > \nu(l+1) \), the smallest value:

\[
\Delta \eta^* = \min(\Delta \eta^\text{con1}, \Delta \eta^\text{con2}, \Delta \eta^\text{con3})
\]

will constitute the maximal incremental adjustments of \( \Delta \eta \); otherwise, the largest value:

\[
\Delta \eta^* = \max(\Delta \eta^\text{con1}, \Delta \eta^\text{con2}, \Delta \eta^\text{con3})
\]

will constitute the minimal incremental adjustments of \( \Delta \eta \).

Upon the critical adjustment quantity \( \Delta \eta^* \) is determined, we can update \( \alpha_i \), \( \forall i \in S \), \( g_i \), \( \forall i \in R \cup E \), the sets \( S \), \( R \), \( E \) similar to C&PP algorithm. 

**Efficiently Updating the Inverse Matrix \( \hat{N} \):**

Once a training sample is either removed from or added to the set \( S \), the inverse matrix \( \hat{N} \) should be changed accordingly. Fortunately, based on Lemma 2, we can update the inverse matrix \( \hat{N} \) efficiently without solving the inverse \( \hat{N} \) directly.

**Lemma 2** Suppose a \((s+1) \times (s+1)\) matrix \( B \) can be partitioned into a block form:

\[
B = \begin{bmatrix} A & \eta^* \\ \eta^\top & Q_s \end{bmatrix}
\]

where, \( A \) is \( s \times s \) matrix and \( A \) is invertible, \( \eta_i = [Q_i, \ldots, Q_s]^T \), \( Q_s \neq 0 \) is a constant.

Then, the inverse matrix of \( B \) can be expanded as follows:

\[
B^{-1} = A^{-1} 0 + \begin{bmatrix} \gamma_i \\ 0 \end{bmatrix} \begin{bmatrix} \gamma_i \top \\ 1 \end{bmatrix}
\]

where, \( \gamma_i = -A^{-1} \eta_i \) and \( k = \eta^\top i \gamma_i + Q_s \).

Furthermore, if \( B \) is invertible and \((B^{-1})_{st} \neq 0 \), \( t = s+1 \), then the inverse matrix of \( A \) can be contracted as follows:

\[
A^{-1} = (B^{-1})_{st} \left( (B^{-1})_{st} - (B^{-1})_{st} \right)^{-1} ((B^{-1})_{st}
\]

where, \( s \neq t \).

It can be easily verified that \( BB^{-1} = I_{s+1} \) and \( AA^{-1} = I_s \). The detailed proof of Lemma 2 can be found in Laskov et al. (2006), it is omitted here.

Based on Lemma 2, if a sample \((x_i, y_i, z_i)\) is removed from the set \( S \), then \( \hat{N} \) can be contracted as follows:

\[
\hat{N} \leftarrow \hat{N} - \left( \hat{N} \cdot \tilde{N}_s \right) \tilde{N}_s
\]

Similarly, if a sample \((x_i, y_i, z_i)\) is added to the set \( S \), then \( \hat{N} \) can be expanded as follows:

\[
\hat{N} \leftarrow \hat{N} + \left( \hat{N} \cdot \tilde{N}_s \right) \tilde{N}_s
\]

where, \( \tilde{N}_s = \sum_{j \in S} \tilde{f}_j Q_j + y_i \tilde{f}_s + \tilde{f}_s^\top + Q_s \) and

\[
\gamma_i = \begin{bmatrix} \tilde{f}_s \nonumber \\ \tilde{f}_s \nonumber \end{bmatrix}
\]

\[
\hat{N} \leftarrow \hat{N} - \begin{bmatrix} z_i \\ 1 \nonumber \end{bmatrix} \begin{bmatrix} Q_s \nonumber \end{bmatrix}
\]

**Preparations for the Next Round of Adjustments:**

From Algorithm 1, it is obvious that we should prepare the inverse matrix \( \hat{N} \) for the next round of ARA.
after RAIA. Similarly, we should also prepare the inverse matrix \( N \) for the next round of RAIA after ARA.

Fortunately, based on Lemma 2, the inverse matrix \( \hat{N} \) can be expanded as follows:

\[
\hat{N} \leftarrow \begin{bmatrix} N & 0 \\ \theta' & 0 \\ \end{bmatrix} + \frac{1}{k_i} \begin{bmatrix} A \\ \theta' \\ \end{bmatrix} \begin{bmatrix} \theta' & 1 \\ \end{bmatrix} 
\]

(29)

where, \( k_i = \theta - [0 \ e_{\theta}] \cdot N \begin{bmatrix} 0 \\ e_{\theta} \end{bmatrix} \) and \( A = -N \begin{bmatrix} 0 \\ e_{\theta} \end{bmatrix} \).

Similarly, we can obtain the inverse matrix \( N \) from the following three steps.

First, based on Lemma 2, compute the inverse matrix \( R = \hat{Q}_L \) by using the contracted rules as follows:

\[
R \leftarrow \hat{N}_{\alpha \rho} - \frac{1}{N_{\alpha \rho}} (\hat{N}_{\alpha \rho} \cdot \hat{N}_{\rho \alpha})_{\alpha \rho}, \quad R \leftarrow \hat{R}_{ss} - \frac{1}{\hat{R}_{ss}} (\hat{R}_{ss} \cdot \hat{R}_{ss})_{ss}
\]

Second, Update the inverse matrix of \( Q_{ss} \) by the rule

\[
R \leftarrow \frac{R + 1}{R}
\]

Finally, based on Lemma 2, the inverse matrix \( N \) can be expanded as follows:

\[
N \leftarrow \begin{bmatrix} R & 0 \\ \theta' & 0 \\ \end{bmatrix} + \frac{1}{k_i} \begin{bmatrix} B \\ \theta' \\ \end{bmatrix} \begin{bmatrix} \theta' & 1 \\ \end{bmatrix}
\]

(30)

where, \( k_i = z_{\theta}' \cdot B \) and \( B = -R \cdot z_{\theta} \).

Similar to Gu and Sheng (2013), we can also prove the feasibility and finite convergence of the AIOSVR learning algorithm. The detailed proof is omitted here. The feasibility of the AIOSVR learning algorithm ensures that there always exist the inverse matrices \( N \) and \( \hat{N} \) during the adiabatic incremental adjustments and the set \( S \) will always be nonempty during the RAIA and ARA. The finite convergence analysis ensures that the AIOSVR learning algorithm will converge to the optimal solution of the minimization problem within finite steps. Therefore, the AIOSVR learning algorithm is effective and reliable.

**Glutamic Acid Fed-batch Fermentation Process**

Glutamic acid fed-batch fermentation is a rather complicated nonlinear process. Its corresponding fermentation system is relatively complex (Desai et al., 2006). However, with the rapid development of modern technology, glutamic acid fed-batch fermentation is equipped with advanced devices, for example, off-gas analyzer and control cabinet. These devices have greatly enhanced the automation of fermentation process.

### Seeds Culture and Fermentation Condition

The seeds (Corynebacterium Glutamicum S9114) are offered by key laboratory of industrial biotechnology, ministry of education, Jiangnan University. The seeds are added into a shake flask filled with liquid nutrient medium and they are adequately cultivated for 8 to 10 h under the condition of 32°C and 200 r/min. Then, the seeds are added into the fermenter, which has approximately 3.4L liquid fermentation nutrient medium and the pressure of the fermenter is kept in 0.07MPa. The initial pH of the liquid fermentation nutrient medium is in the range of 7.1±0.1 during the whole fermentation process, 25% ammonia is automatically added, which also offers necessary ammonia for glutamic acid synthesis. Moreover, 50% glucose is added in the fermentation process to offer necessary glucose concentration for glutamic acid growth. Furthermore, in order to avoid the step change of substrate concentration, a modified feed-rate profile is utilized in this study, which is shown in Fig. 2.

### Variables in Glutamic Acid Fed-batch Fermentation Process

There are three different kinds of variable in glutamic acid fed-batch fermentation process: Physical variable, chemical variable and biological variable. The physical variable, such as temperature (°C), pressure (MPa) and air flow (m³/h), etc., can be measured online. Similarly, the chemical variable, such as pH and Dissolved Oxygen (DO), can also be measured online by certain electrode.
Fig. 2. Feed-rate profiles

However, the biological variable, such as biomass concentration, glutamic acid concentration and substrate concentration, cannot be measured online. Generally, these three biological variables can only be analyzed offline every two hours. Fortunately, with the help of soft sensor modelling, they can be measured and tuned online. In our experiment, such variables as temperature, pressure, air flow, pH and DO keep constant. The mixing speed of motor can control the DO concentration. Several batch experiments are carried out under the condition of keeping 10%, 20, 30 and 50% DO concentration, respectively.

Soft Sensor Modelling of Biomass Concentration during Fermentation using AIOSVR Learning Algorithm

Data Preprocessing

The data can be divided into offline data and online data. Generally, the temperature, pressure, air flow and pH are almost constant. Therefore, they cannot be treated as the input variable of the soft sensor modelling. The offline data is measured every two hours, so the data can be divided into offline data and online data. The data can be measured and tuned online.

Selection of Input Variables

The glutamic acid growth mainly depends on biomass concentration and substrate concentration by inner mechanism analysis (Zhang et al., 2005). However, different DO leads to different cell activity, which means that the oxygen uptake \( O_{2} \) should be treated as an input variable of the soft sensor modelling. Therefore, there are three different kinds of input variables in the soft sensor modelling of biomass concentration: Biomass concentration \( X_{v} \), substrate concentration \( S_{n} \), glucose feed-rate \( F_{i} \) and oxygen uptake \( O_{2} \).

In our simulation experiment, the following variables \( X_{v1}, X_{v2}, X_{v3}, S_{n1}, S_{n2}, S_{n3}, F_{i}, F_{i1}, F_{i2}, F_{i3} \) and \( O_{2}, O_{21} \) are selected as the input variables of the soft sensor modelling. The result of input variables using Principle Component Analysis (PCA) (Li et al., 2008) is shown in Table 3.

According to the characteristic of fermentation and the contribution of each principle component variable in Table 3, the input variables \( X_{v1}, X_{v2}, S_{n1}, F_{i1}, F_{i2}, F_{i3} \) and \( O_{2}, O_{21} \) are ignored. So \( X_{v1}, X_{v2}, S_{n1}, S_{n2}, F_{i}, F_{i1} \) and \( O_{2} \) are selected as the input vector \( x_{i} \) of the expanded training sample set \( T \).

Setting of Model Parameters

In order to ensure the matrix \( Q_{x}^{N} \) is always invertible, \( K(x_{i}, x_{j}) = \exp(\frac{-||x_{i}-x_{j}||^{2}}{2\sigma^{2}}) \) is used as the kernel function, where the kernel width parameter is set as \( \sigma = 0.7.71 \). Due to the main function of the parameter \( C \) is to transform the output \( y^{*} \) into \( y^{*} \), for convenience, \( C \) is set as 100. In addition, from Equation 23, it is easy to verify \( \hat{y}_{\hat{x}}, \hat{y}_{\hat{p}} \) and \( \hat{y}_{\hat{s}}, i \in S \) have the same denominator \( \text{det}(M) \) and \( \text{det}(M) \) only correlates with \( \text{det}(M) \). Therefore, \( \vartheta \) can determine \( \Delta \eta \), but is independent with the structural changes of the sets \( S, R \) and \( E \). So the parameter \( \vartheta \) is fixed at -1. The parameter \( \nu \) of \( \nu \)-SVR and AIOSVR is set as 0.3.

Training and Testing of Data

Six batches of glutamic acid fermentation data are selected from the above experiments and each batch of

Table 3. The result of input variables using PCA

<table>
<thead>
<tr>
<th>Principle component variable</th>
<th>Eigenvalue</th>
<th>Contribution (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_{v1} )</td>
<td>2.87</td>
<td>19.02</td>
</tr>
<tr>
<td>( X_{v2} )</td>
<td>2.23</td>
<td>14.78</td>
</tr>
<tr>
<td>( X_{v3} )</td>
<td>0.25</td>
<td>1.66</td>
</tr>
<tr>
<td>( S_{n1} )</td>
<td>1.92</td>
<td>12.72</td>
</tr>
<tr>
<td>( S_{n2} )</td>
<td>1.35</td>
<td>8.95</td>
</tr>
<tr>
<td>( S_{n3} )</td>
<td>0.35</td>
<td>2.32</td>
</tr>
<tr>
<td>( F_{i} )</td>
<td>1.98</td>
<td>13.12</td>
</tr>
<tr>
<td>( F_{i1} )</td>
<td>1.58</td>
<td>10.47</td>
</tr>
<tr>
<td>( F_{i2} )</td>
<td>0.14</td>
<td>0.93</td>
</tr>
<tr>
<td>( O_{2} )</td>
<td>2.36</td>
<td>15.64</td>
</tr>
<tr>
<td>( O_{21} )</td>
<td>0.06</td>
<td>0.40</td>
</tr>
</tbody>
</table>
data can represent the whole fermentation process. The data includes offline analyzed data and online measured data. Five batches are used to train the model and the remaining one batch is used to test the model. The detailed steps of training and testing of data using AIOSVR learning algorithm is listed as follows:

**Step 1:** Select the input and output of the expanded training sample set $T$. Here, the input is selected as $x_i = [X_{i-1}, X_{i-2}, S_{i-1}, S_{i-2}, F_i, F_{i-1}, O_i]$ and the output is $y_i' = y_i/100 = X_{i}$.  

**Step 2:** In order to enhance the speed of operation, the AIOSVR is used to train the model offline based on the five batches of glutamic acid fermentation data.

**Step 3:** The remaining one batch glutamic acid fermentation data is used to test the model which is already established. Furthermore, the minimum error allowed is set as $\text{Err}_{\text{min}} = 2$.  

**Step 4:** Calculate the prediction error: $e_i = y_i' - \hat{y}_i'$, where $y_i'$ stands for the real value and $\hat{y}_i'$ stands for the predictive value. If $|e_i| < \text{Err}_{\text{min}}$, the model will not be tuned and the data is saved for future use; otherwise, the model will be tuned by training the previously saved data one by one according to Step 2.

**Step 5:** The next step of prediction is carried out according to Step 4 until the whole fermentation process is over.

### Results

All simulation experiments are performed on a 3.1 GHz Intel® Core™ i5-2400 with 4GB RAM and MATLAB 2010a platform.

The prediction results and prediction errors of biomass concentration based on v-SVR and AIOSVR are shown in Fig. 3 and 4, respectively. Note that biomass concentration refers to OD620 value, which is measured by diluting extracted broth 100 times. The Mean Square Error (MSE) and training time of v-SVR and AIOSVR are shown in Table 4. Note that $T_{24}$ and $T_{32}$ refer to the training time at fermentation time of 24 and 32 h, respectively.

### Discussion

From Fig. 3 and 4, it is clear that a relatively large prediction error occurred at fermentation time of 24 and 32 h, which leads to a larger prediction error. The reason is that the biomass activity becomes stronger at fermentation time of 24 and 32 h. Fortunately, AIOSVR learning algorithm can tune the parameters quickly to cope with this change and obtain smaller prediction error than v-SVR by online learning. Therefore, the soft sensor modelling based on AIOSVR is of stronger adaptive ability than v-SVR.

Table 4 demonstrates that the MSE of AIOSVR is around half that of v-SVR, which means AIOSVR has stronger generalization ability. Furthermore, the training time of AIOSVR is also less than that of v-SVR.

In summary, the AIOSVR learning algorithm is of better generalization ability and online learning speed than v-SVR. So it is more suitable for the soft sensor modelling of real fermentation process.

<table>
<thead>
<tr>
<th></th>
<th>v-SVR</th>
<th>AIOSVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>2.616</td>
<td>1.303</td>
</tr>
<tr>
<td>$T_{24}(s)$</td>
<td>48.500</td>
<td>36.300</td>
</tr>
<tr>
<td>$T_{32}(s)$</td>
<td>26.500</td>
<td>18.400</td>
</tr>
</tbody>
</table>
Conclusion

In order to design an online soft sensor modelling for real fermentation process, we first proposed an AIOSVR learning algorithm and then presented its application in soft sensor modelling of biomass concentration for glutamic acid fermentation process.

In theory, the AIOSVR learning algorithm can also be applied in the soft sensor modelling of substrate concentration, product concentration and so on. Therefore, the AIOSVR learning algorithm can be widely used in the soft sensor modelling of microbial fermentation process.

Acknowledgement

The authors will give special thanks to Professor Yuanda Song of Jiangnan University for proofreading the manuscript and giving some constructive comments and suggestions.

Funding Information

This work was supported by the National Natural Science Foundation of China under Grant No.61273131.

Author’s Contributions

Binjie Gu: Conceived of the research, designed the research plan, drafted and revised the manuscript.
Feng Pan: Involved in acquisition and analysis of data, manuscript reviewing and provided funding support.

Ethics

This article is original containing unpublished materials. All authors have read and approved the manuscript and no ethical issues involved.

References

Karush, W., 1939. Minima of functions of several variables with inequalities as side conditions. MSc Thesis, University of Chicago, USA.


List of Symbols

- $R^d$ denotes the $d$-dimensional Euclidean space
- $T$ denotes matrix transposition
- $\Delta$ denotes the amount of the change of each variable
- $Q_{Sc}$ denotes the submatrix of $Q$ with the rows and columns indexed by the set $S$
- $Q_{S}$ denotes the subvector of the matrix $Q$ with the rows and columns indexed by the set $S$ and $c$, respectively
- $e$ denotes the all ones column vector indexed by the set $S$
- $0$ denotes the all zeros column vector with proper dimensions
- $M^{-1}$ denotes the inverse of the matrix $M$
- ‘def’ above = ‘ denotes the left side of the equal sign is defined as the right side
- $I_m$ denotes the identity matrix with $m$ dimensions
- $\hat{N}_{it}$ denotes the $i$ th row and the $t$ th column of the matrix $\hat{N}$, where $i$ stands for the corresponding index in $\hat{N}$
- $\hat{N}_{it}$ denotes the submatrix of $\hat{N}$ with deleting the $i$ th row and $t$ th column, where $i$ stands for the corresponding index in $\hat{N}$
- $X_i$ denotes the biomass concentration at time $i$ (g/L)
- $S_i$ denotes the substrate concentration at time $i$ (g/L)
- $F_i$ denotes the glucose feed-rate at time $i$ (mL/h)
- $O_{2i}$ denotes the oxygen uptake at time $i$ (mol)
- $CO_{2i}$ denotes the $CO_2$ production at time $i$ (mol)
- $V$ denotes the volume of broth (m$^3$)
- $OUR_i$ denotes the oxygen utilization rate at time $i$ (mol/m$^3$/h)
- $CER_i$ denotes the $CO_2$ evolution rate at time $i$ (mol/m$^3$/h)