Biochemical Engineering Journal 136 (2018) 28–39

Contents lists available at ScienceDirect

Biochemical Engineering Journal

journal homepage: www.elsevier.com/locate/bej

A deep learning based data driven soft sensor for bioprocesses

Vineet Gopakumar, Sarthak Tiwari, Imran Rahman*

Chemical Engineering and Process Development, National Chemical Laboratory, Dr. Homi Bhaba Road, Pune - 411008, Maharashtra, India

ARTICLE INFO

Article history:
Received 9 December 2017
Received in revised form 13 April 2018
Accepted 22 April 2018
Available online 27 April 2018

Keywords:
Deep learning
Deep neural networks
Soft-sensor
Streptokinase
Penicillin
Fermentation

ABSTRACT

Developing accurate and robust sensors for nonlinear and highly varying systems is a challenge. Deep learning, an advanced technique to learn deep architectures, has become a popular training strategy while dealing with complex problems. In this paper, deep learning has been introduced to develop data driven soft sensors for estimating crucial parameters in two fermentation processes, namely, Streptokinase and Penicillin. Additionally, the performance of the developed soft sensor is compared to an SVR based soft sensor. The results clearly indicate that deep learning is an attractive alternative to traditional techniques for soft sensor modelling as it represents nonlinear systems better, makes full advantage of process data by also incorporating unlabelled data and handles large datasets efficiently. Deep learning proves to be a promising technique for soft sensor modelling in highly data driven complex bioprocesses.

1. Introduction

In fermentation processes, some crucial variables that are basic indicators of the process behavior are often difficult to measure online. These variables are normally measured in laboratories by using an off line sample analyzer. Such analyzers turn out to be expensive as they incur high and frequent maintenance costs [1]. Furthermore, these measurements cannot be used as feedback signals for control systems because of the significant delay in laboratory testing and analysis. To solve this problem, soft-sensors are being widely utilized to make reliable online estimations of such important quality process variables. Soft sensors predict difficult to be measured variables by correlating them with easily available variables.

Soft sensors can be categorized into two broad classes, model driven and data driven. Model driven or First Principle Models (FPM) based sensors require in-depth knowledge of the process mechanism. However, they can't be employed when the process mechanism is unknown, especially in the case of complex processes. Furthermore, FPM based models are computationally expensive. Data driven models, also known as Black-Box models rely only on historic data obtained from the industrial process and can be developed quickly without requiring knowledge of the phenomenology involved. Advancements in collection, storage and management of data have made data driven models a preferable choice for soft sensor modeling of complex processes [2–5]. Currently, some of the popular learning techniques being employed for data driven soft sensor modeling are Principal Component Regression (PCR) [6], Fuzzy Logic [7], Support Vector Regression(SVR) [4,5] and Artificial Neural Networks [8]. Despite of their popularity, current data driven soft sensor modeling techniques face issues like, poor generalization for highly nonlinear systems, underutilized unlabelled data, inability to predict multiple outputs simultaneously and requiring high computational efforts to process large amount of data.

Nomenclature

S	Substrate Concentrations
s_0	Initial Substrate Concentrations
q	Feed rate of glucose (l/h)
s_t	Streptokinase concentration (g/l)
t	Time
v	Reactor Volume (l)
x_t	Total biomass concentration (g/l)
x_a	Active (plasmid containing) biomass concentration (g/l)
x_{aO}	Active biomass concentration at time t = 0 (g/l)
x_{t0}	Total biomass concentration at time t = 0 (g/l)
μ	Specific growth rate, h^{-1}

* Corresponding author.
E-mail addresses: ch15b073@smail.iitm.ac.in (V. Gopakumar), f2011836@pilani.bits-pilani.ac.in (S. Tiwari), r.imran@ncl.res.in (I. Rahman).

https://doi.org/10.1016/j.bej.2018.04.015
1369-703X/

In recent years, Deep learning based architectures have been applied extensively in Speech Recognition, Computer Vision, Natural Language Processing(NLP), Bioinformatics and various other domains [9].Deep learning, a subset of machine learning, is capable of learning deep, hierarchical artificial neural networks efficiently [10]. Since deep architectures are composed of multiple layers of parameterized nonlinear modules, they achieve better generalization in case of highly varying nonlinear functions [11,12].On the other hand, generalization of highly varying functions with shallow architectures, consisting of one or two non linearities, would require a large number of units [13]. This ability of deep networks to accurately represent highly varying functions is an attractive feature in the context of bioprocesses which involve highly varying and complex bio-chemical reactions. A practical difficulty encountered in soft sensors is gradual deterioration of the predictive accuracy due to sudden changes in the state of the chemical process, deterioration in catalyst performance and sensor drifts [14]. Neural networks overcome this difficulty due their adaptive nature of updating weights after each iteration of a batch wise training cycle. Another important issue faced in soft sensor modeling is effective utilization of unlabelled data. Due to the slow speed of complex bioprocesses, quality variables which are measured off-line have a low sampling rate, 1–2 days, and are more difficult to obtain than on-line measured variables. This leads to a mismatch in input – output pairs. Therefore the historic data from industrial process would always contain some unlabelled data, which only consists of input variables. This unlabelled data might contain useful process information which, if utilized effectively, could improve soft sensor performance. Traditional modeling techniques do not incorporate unlabelled data in soft sensor modeling and hence their prediction accuracy might get affected when the amount of labelled data is limited. However, semi-supervised learning techniques enable Deep Neural Networks to utilize unlabelled data containing rich process information [15,16].

Shang et al. [17] introduced deep learning to soft sensor modelling for online estimation of 95% cut point of Heavy Diesel in the Crude Distillation Unit (CDU). It was shown that a pre trained deep neural network extracts latent variables effectively and thereby achieves better generalization as compared to the traditional SVR. Despite being an effective technique for representing complex, nonlinear and highly varying functions, Deep learning has not yet been explored in the field of nonlinear bioprocesses. In this paper we introduce a Deep Neural Network based data driven soft sensor for online estimation of product and biomass concentrations in the Streptokinase and Penicillin processes.

The paper is organized as follows. Section 2 discusses the advantages of deep architectures and their suitability with respect to nonlinear bioprocesses. Soft sensor development is explained in detail in Section 3. Simulation case studies for the two fermentation processes are given in Section 4. The final section gives concluding remarks.

2. Developing and training the neural network

In the present study, a semi-supervised learning strategy is employed to train the network. The first step involves unsupervised learning, where the network is pre-trained using only unlabelled data. Section 2.1 explains the Self Organizing Maps (SOM) algorithm used for unsupervised learning. The second step is supervised learning with labelled data, in which the network is fine-tuned by using the error back propagation algorithm. The parameters generated in the unsupervised learning step are used for initializing the weights in the supervised learning step. Sections 2.2 and 2.3 explain forward and backward propagation steps respectively. Over fitting, a common problem that arises while training neural networks, is encountered when the number of unknown parameters such as weights and bias of the network is greater than the number of datasets used to train the network. This study employed various methods to overcome the problem of over fitting as listed in Section 2.4.

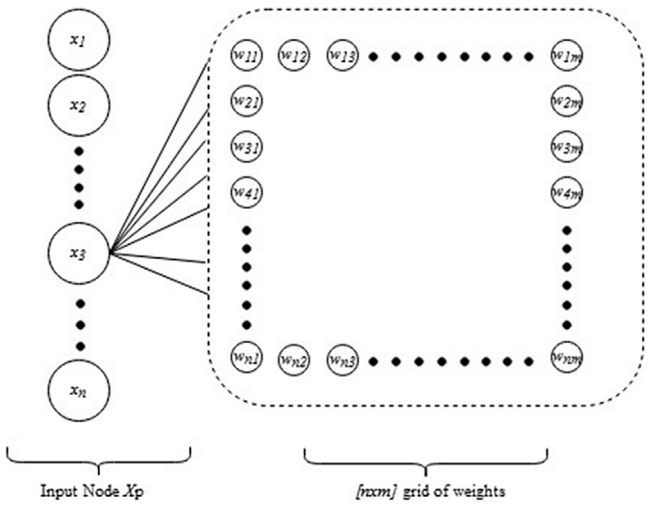

Fig. 1. Two dimensional grid depicting SOM.

2.1. Unsupervised learning using SOM (self-organizing maps)

Unsupervised training is employed when unlabelled data points are fed to the input layer of the neural network. This study used self-organizing maps (SOM), which possesses high clustering capability for high dimensional datasets. SOM is a single layered network consisting of a [n x m] two dimensional grid of weights w_{ij} *i=(1,2,3,. . ..n), j=(1,2,3,. . ..m)* as shown in Fig. 1. The algorithm used to train the grid is shown in [18].

The weights used by the SOM for clustering in the input layer are equated to the initial weights in the input layer of the neural network before it undergoes supervised training. The speed and probability of convergence in the back propagation step is influenced greatly by weight initialization. Pre-training the network extracts implicit process information from unlabelled data and initialising weights based on this process information, opposed to initializing weights randomly, would make the back propagation more effective.

2.2. Feed-forward neural network

Feed forward networks are typically used for MIMO systems, where input data are propagated through the network to the output layer. The weight associated with the k^{th} hidden node in the i^{th} hidden layer for the input variable (j) is denoted as $w_{kj}^{(i)}$ for k = (1,2,3. . ..n).

The input to the k^{th} hidden node in the i^{th} hidden layer is given as:

$$\sum_j w_{kj}^{(i)} a^{(i)} + b_k^{(i)} = net_k^{(i)} \quad (1)$$

The data fed to the input layer is passed to a hyperbolic tangent sigmoid transfer function:

$$\text{tansig(x} = \frac{2}{1+2e^{-2x}} - 1 = \text{f(x)} \quad (2)$$

so as to normalize all inputs that are fed to the subsequent hidden layers and the output layers to range between 1 and −1.

Normalization makes initializing and updating weights and bias less mathematically expensive and improves training time [19].

Post activation, the output from the k^{th} hidden node is:

$$f\left(net^{(i)}\right) = f\left(\sum_{j=1}^{N} w_{kj}^{(i)} a^{(i-1)} + b_k^{(i)}\right) = a_k^{(i)} \tag{3}$$

The input to the k^{th} node in the next $i+1^{th}$ layer is:

$$\sum_{j=1}^{N} w_{kj}^{(i)} f(net^{(i)}) + b_k^{(i+1)} \tag{4}$$

Post activation it is:

$$f\left(\sum_{j=1}^{N} w_{kj}^{(i+1)} f\left(net_k^{(i)}\right) + b_k^{(i+1)}\right) = a_k^{(i+1)} \tag{5}$$

2.3. Supervised learning using error back propagation

Error back propagation uses the Widrow-Hoff delta learning algorithm [20–21] where weights are updated by minimizing the mean square error of the output response in the backward direction from the output layer to the input layer as seen in Fig. 2. The mean square error of the output response is given as:

$$E = \frac{1}{2}\sum_{k}(c - a_k^N)^2 \tag{6}$$

where N is the number of hidden layers in the neural architecture. To minimise the network error, $\Delta W = \frac{\partial E}{\partial W}$ is used to update the weights. This simplifies to:

$$\Delta w_{kj} \propto \frac{\partial E}{\partial w_{kj}} = \sum_{k}\left[\frac{\partial E}{\partial a_k^N}\frac{\partial a_k^N}{\partial net_k^{(N)}}\frac{\partial net_k^N}{\partial a_k^{(N-1)}}\frac{\partial a_k^{N-1}}{\partial net_k^{(N-1)}}\cdots\frac{\partial net_k^{i+1}}{\partial a_k^{(i)}}\right]\frac{\partial a_k^i}{\partial net_k^{(i)}}\frac{\partial net_k^i}{\partial w_{kj}} \tag{7}$$

By chain rule for partial derivatives, for the ith hidden layer. On evaluating these partial derivatives we get:

$$\varepsilon\sum_{k}\left[\left(c^N - a_k^N\right)a_{k(1-}^N a_k^N)w_k^{i+1}a_{k(1-}^N a_k^N)a_k^{i-1}\right] \tag{8}$$

$$\Rightarrow \in \sum_{k}\left[\delta_k^N w_k^{(i+1)}\right]a_k^i \tag{9}$$

$$\Rightarrow \varepsilon\delta_k^{i+1}a_k(i-1) \tag{10}$$

Hence,

$$w_{kj} = w_{kj} - \eta\Delta w_{kj} \tag{11}$$

Although widely used, the error back propagation algorithm shows slow convergence because of its small step sizes so as to locate the global minima of the error function. Gauss Newton method was adopted instead to minimize the error function by using the second order derivative of the error function. However this method diverges for functions where quadratic approximation cannot be employed such as the Rosenbrock function. Thus, Levenberg-Marquardt algorithm is used to train neural networks which uses a combined training process for complex curvatures of the error function [22]. Gradient descent method is used until the curvature is smooth enough to employ quadratic approximation for Gauss Newton method.

For Gauss Newton method, we define the error vector as:

$$e = [e_1\, e_2 e_3 \ldots . e_n] \tag{12}$$

and the corresponding Jacobian $\boldsymbol{J}$ for $\boldsymbol{e}$ as:

$$J = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w1} & \frac{\partial e_1(w)}{\partial w2} & \cdots & \frac{\partial e_1(w)}{\partial wm} \\ \frac{\partial e_2(w)}{\partial w1} & \frac{\partial e_2(w)}{\partial w2} & \cdots & \frac{\partial e_2(w)}{\partial wm} \\ \cdots & \cdots & & \\ \cdots & \cdots & & \\ \frac{\partial e_n(w)}{\partial w1} & \frac{\partial e_n(w)}{\partial w2} & \cdots & \frac{\partial e_n(w)}{\partial wm} \end{bmatrix} \tag{13}$$

Where n is the number of test variables and m is the number of weights.

Weights are updated by:

$$w^{i+1} = w^i + \left[J^T J\right]^{-1} J^T e \tag{14}$$

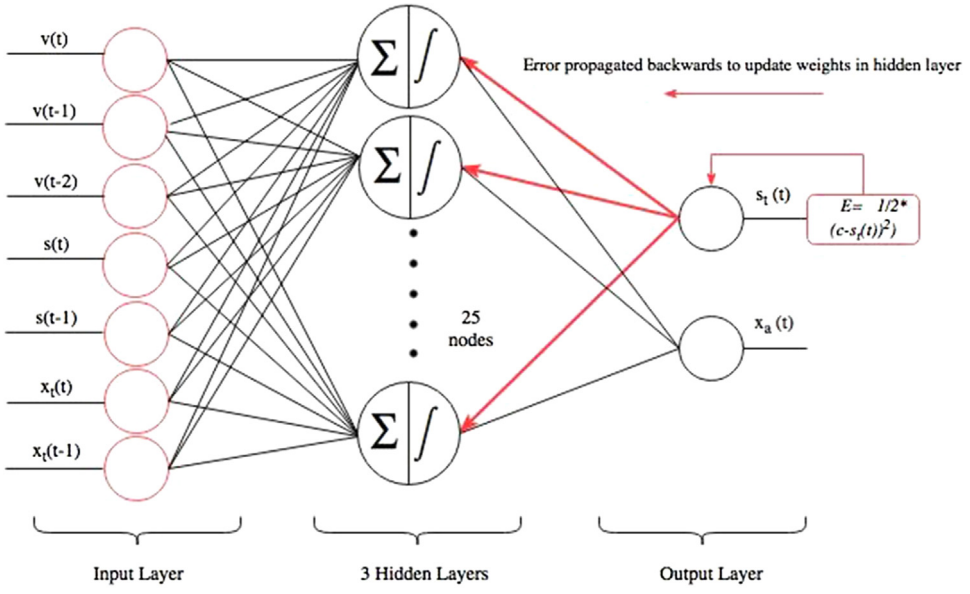

Fig. 2. Representation of error back propagation algorithm for Streptokinase process.

In the Levenberg- Marquardt algorithm, weights are updated by [23]:

$$w^{i+1} = w^i + \left[J^T J + \mu I\right]^{-1} J^T e \quad (15)$$

Where μ is the combination coefficient large values of μ transforms (15) to (11) and negligible values transforms it to (14). Thus, μ is chosen according to the curvature of the error function.

2.4. Overcoming the Over fitting problem

2.4.1. Early stopping

The efficacy of a deep neural network depends on the number of data sets that train the network. However, a common problem of over fitting a neural network persists when the network begins to memorize the training data and generalization is lost. Early stopping is a conventional method employed to overcome over fitting as explained in [24]. Apart from monitoring error on the training set, validation set error is also calculated. The validation error shoots up after a certain number of epochs as the network begins to lose generalization, observed in Figs. 3 and 7. The point where the validation error is observed to achieve a minimum after which it begins to increase is termed as the early stopping point and the network is deemed to have met the early stopping criteria. At this epoch, the network weights provide the best generalization.

2.4.2. Training using Bayesian regularisation

Regularization is employed to overcome over fitting in the network. Unlike the conventional training algorithms, the Bayesian framework for neural networks doesn't update randomly initialized weights but involves a probabilistic distribution of weights [25,26]. The network error was measured by $E_T = \frac{1}{2}\sum_k \left(c - a_k^N\right)^2$ in the Levenberg-Marquardt algorithm. However, in the Bayesian framework a weight decay term, $E_T = \frac{1}{m}\sum_{i=1}^{m} w_i^2$ is added to network error to obtain a smooth curvature for E and give a more generalized network. On adding this weight decay term, the Bayesian framework performs shrinkage and looks for a minimum in E with minimal weights. This approach increases the robustness of the model by estimating the number of effective parameters or weights required for the network thus eliminating the need of finding parameters that will not significantly increase the robustness of the model [26]. A detailed explanation to Bayesian Regularization can be found in [27].

3. Soft sensor development

The procedure adopted to develop the soft-sensor is briefly summarized below.

1) Identify relevant input and output variables based on process knowledge
2) Gather data and segregate it into training and testing data. Further partition training data into labelled and unlabelled data
3) Find optimal architecture with minimum over fitting by applying k-fold cross validation
4) Pre-train the network by carrying out SOM based unsupervised learning on unlabelled data
5) Use weights from Step 4 to initialize the network parameters
6) Carry out regularized LM based back propagation to fine-tune the network up to maximum number of epochs given by early stopping
7) Evaluate soft sensor performance on the testing dataset

4. Industrial case studies

4.1. Streptokinase process

A blood clot (thrombus) developed in the circulatory system can cause vascular blockage leading to serious conditions. A healthy haemostatic system prevents development of blood clots in normal circulation and minimizes vascular injuries, thus avoiding blood loss. Pathologies which involve formation of a blood clot due to failure in haemostasis, require clinical intervention of intravenous administration of thrombolytic agents such as Streptokinase.

Streptokinase (SK) is considered to be a very valuable biomolecule for thrombolytic therapy because of its lower cost and longer in-vivo half-life than other similar molecules such as urokinase(UK) and tissue plasminogen activator (TPA) [28,29]. It doesn't have any enzymatic activity of its own, but a 1:1 stoichiometric complex of Streptokinase and Plasminogen produces trypsin-like serine protease activity which helps in dissolution of fibrin blood clot [29]

In the present work, data were generated by simulating the phenomenological model [30–31] given in Appendix A. Parametric values and initial conditions are mentioned in Tables A1 and A2 respectively. A total of 70 batches of data were simulated with sampling interval of 0.25 h. Also, the batch durations were kept different with time spans between 9 h −15 h. To show batch to batch changes, initial conditions of s and x_a were varied abruptly in the ranges, $60 \le s(0) \le 80$ and $0.5 \le x_a \le 0.9$. To replicate real life process scenario, where there is always some instrumental and measurement noise present, Gaussian noise of strength 5–10% was added to all the process variables. An important factor which may influence the performance evaluation of the model is the method adopted to split the data into training and testing sets. An ideal training set should contain as many process conditions as possible so that it represents the system thoroughly. If this is not fulfilled, the model would not learn the experiences which were not present in the training set and the predictive performance for those experiences would suffer. Hence it was ensured that the training dataset consisted of batches which covered entire ranges of $s(0)$ and x_a. The test dataset on the other hand should not contain too many similar process conditions. Otherwise model performance would vary greatly for different test sets with different process conditions and a true picture of the soft sensor performance would not be achieved. To fulfil this condition the number of similar batches, in terms of $s(0)$ and x_a values, were limited. Keeping the above conditions in mind, out of the 70 batches of data generated, 56 were used for training and 14 for testing. Training data was further partitioned into labelled data and unlabelled data by randomly deleting the target vectors in 25% data points of the training dataset. The number of data points in unlabelled and labelled datasets were 840 and 2520 respectively.

Since changes in reactor volume, total biomass and substrate concentrations are major indicators of change in Streptokinase and biomass concentrations, both current and lagged values of these variables were included in the input matrix. The number of lagged values or the input-output function was heuristically optimized such that the cross validation error was minimized. Multiple input matrices, and hence datasets, with a varying number of lagged values, k = (1,4)], for the three variables were created. Based on these input matrices, DNN models were trained using 80% of the training set and validated on the remaining 20%. After cross validation, the k value, or the input – output function with the lowest RMSE on the validation set was selected. Accordingly, the following input-output function was used to predict streptokinase and biomass concentrations.

$$s_t(t) = f_1(v(t), v(t\text{-}1), v(t\text{-}2), s(t), s(t\text{-}1), x_t(t), x_t(t\text{-}1)) \quad (16)$$

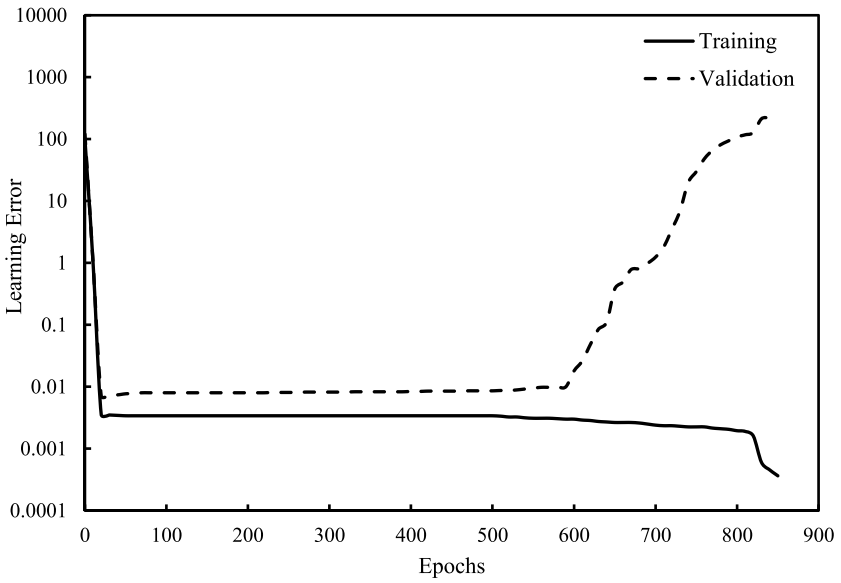

Fig. 3. Performance curve for Streptokinase process.

$$x_a(t) = f_2(v(t), v(t\text{-}1), v(t\text{-}2), s(t), s(t\text{-}1), x_t(t), x_t(t\text{-}1)) \quad (17)$$

An optimal architecture with best generalization and minimal over fitting was found using the five-fold cross validation technique. For carrying out the cross validation, the labelled dataset was segregated into five equal subsets. Of the five subsets, a single subset was used as validation subset and the rest as training subsets. The process was then carried out five times, with each subset exactly used once as validation subset. The five RMSE's were then averaged out to gauge the performance of that architecture. Considering the RMSE's for various architectures, 7-5-5-5-2 was deemed as the most robust architecture for the network.

The network was then pre-trained by carrying out unsupervised learning using the unlabeled data set. The weights generated by SOM were used as initial weights in the supervised training phase. The adaptive nature of the neural network is introduced by varying the value of μ, as mentioned in Section 2.3, between 0.5 and 0.005 with a step size of 10 for the streptokinase process. As explained in Section 2.4.1, the validation/testing error begins to increase after early stopping criteria is met. In Fig. 3. the root mean square error during testing is observed to shoot up after 500 epochs which was fixed as maximum number of epochs for back propagation.

On performing Bayesian regularization, shrinkage of the network was performed by obtaining 71.7 effective parameters out of 112 (maximum number of parameters) parameters of the 7-5-5-5-2 network. The model was finally tested for its generalization ability using the testing dataset. Root Mean Square Error given below in (18), was used to evaluate the prediction capability.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_p} \left(y_i - y_{\hat{i}}\right)^2}{N_p}} \quad (18)$$

Where $y_i, \hat{y}_i$ respectively, are the desired value of the i^{th} input–output pattern, and its estimate obtained from model, and N_p is the total number of input–output patterns.

In order to compare the performance of Deep Neural Networks with shallow architectures, with regard to nonlinear modelling in highly data driven systems, a soft sensor based on traditional SVR was also developed. Since SVR is a widely used non-linear soft sensor modelling technique with excellent generalization, it was chosen for the purpose of comparison. The main advantages of SVR, regarding nonlinear modeling are : excellent generalization ability, convergence to the unique global minimum and the ability to learn non linear relationships by efficiently using non linear kernel transformations. Exponential RBF was chosen as a kernel function since it has good performance in most cases. Parameters, C and ε were carefully chosen using the 5 fold cross validation technique. The optimal parameters of SVR are mentioned in Appendix C. A Deep Neural Network trained only using supervised learning strategy (backpropagation) was also developed for the purpose of comparison. This was done to evaluate the effectiveness of un-supervised learning on the performance of Deep Neural Networks. The above mentioned DNN had the same architecture as semi-supervised DNN, but used random initial weights.

Table 1
Comparison of RMSE values of different soft sensors for the Streptokinase process.

Variable	SVR	Supervised DNN	Semi supervised DNN	% Improvement (1&3)
Streptokinase	0.089	0.073	0.0565	36.51
Biomass	0.046	0.015	0.0091	80.2

It can be observed from Table 1 that semi-supervised DNN based soft sensor gives the lowest RMSE values for both the variables. On comparing the RMSE values between semi-supervised and supervised DNNs, it can be observed that the modelling performance gets improved as unlabelled data is taken into account.

Figs. 4 and 5 show the prediction curves by the developed models for two test batches. It can be seen that semi-supervised DNN accurately captures the varying process dynamics. Considering the RMSE values and prediction curves, it can be observed that although semi supervised DNN based soft sensor gives the best results, it performs only marginally better than the other soft sensors. Since in this case study, DNN based soft sensors utilized a relatively shallow architecture i.e. with only 3 hidden layers, their performance showed only a small improvement over that of SVR. Also, the amount of unlabelled data was not large enough to create a significant difference between the performances of semi-supervised and supervised DNN based soft sensors.

4.2. Penicillin process

In this section, the proposed strategy is applied to the Penicillin process, which is a known benchmark process for modelling and

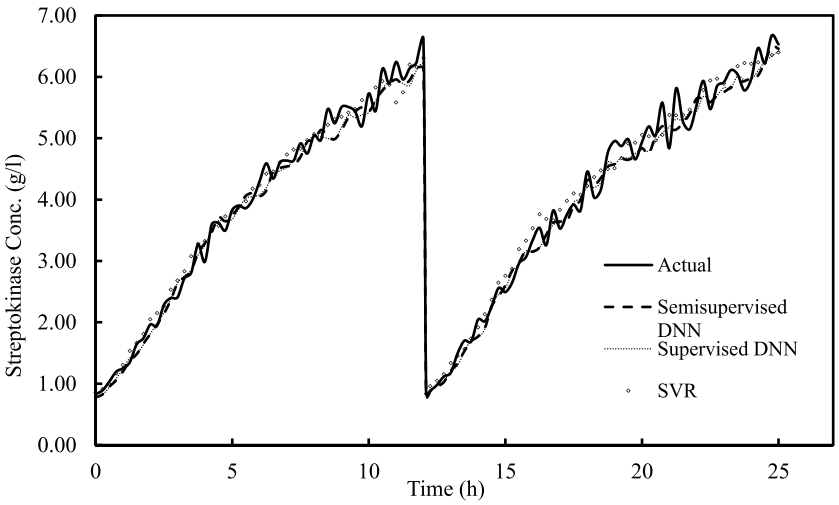

Fig. 4. Prediction curves of Streptokinase Concentration by DNN based soft-sensor.

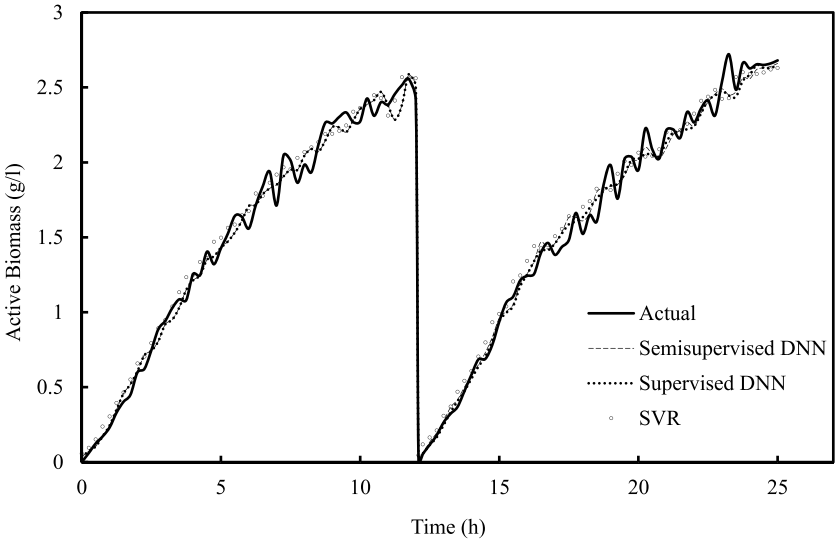

Fig. 5. Prediction curves for Active Biomass Concentration by DNN based soft-sensor.

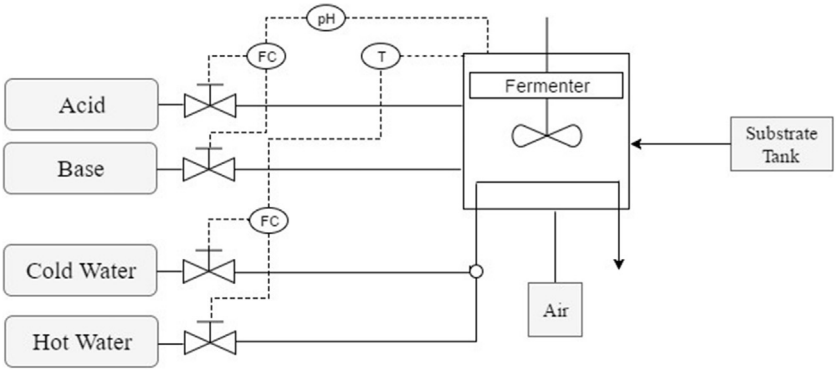

Fig. 6. Process flow for the Penicillin process.

control of fed batch reactors [31,32]. Penicillin production is an industrially important process with nonlinear dynamics and multistage characteristics. Since formation of antibiotic, Penicillin in present case, is not associated with cell growth, initially a short batch operation is carried out to grow the microorganism. When the microorganism has grown enough to deplete the substrate concentration, substrate feed is introduced to carry out fed-batch operation for Penicillin synthesis. The initial batch operation lasts around 40 h while the entire Penicillin process has the duration of 400 h. The process flow is depicted in Fig. 6.

In present work, a simulator called PenSim V2.0, developed by the Process Modelling, Monitoring, and Control Research Group

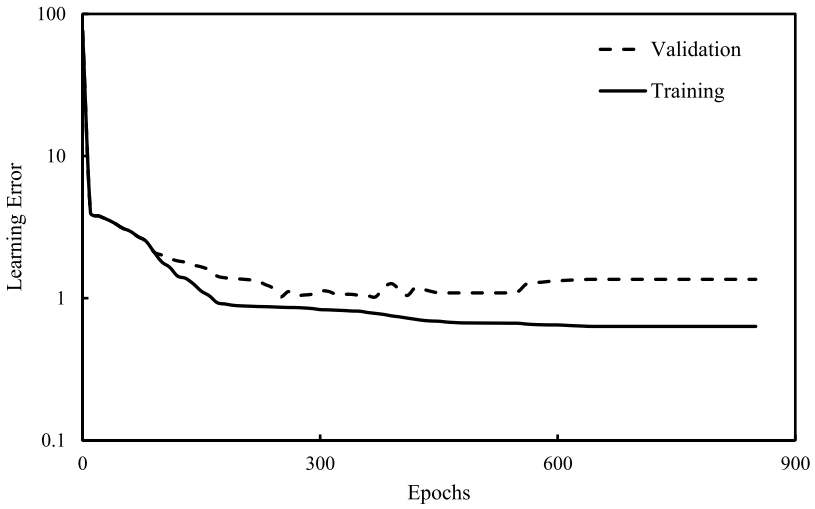

Fig. 7. Performance curve for Penicillin process.

at Illinois Institute of Technology has been used to generate data [33]. The simulator, which can perform under various operational modes, is based on a detailed unstructured model which considers additional variables such as pH, temperature, aeration rate, agitation power, and CO2 evolution and heat generation for completeness. A total of 60 batches of data were generated. To generate different batches of data, initial conditions and set point values of operational parameters were varied randomly within their respective ranges, given in Tables B1 and B2 in Appendix B. To create real life like conditions, feeding strategy was changed batch to batch. The simulations were run under closed loop control of pH and Temperature, while glucose and air were operated under open loop conditions. Each batch was run for a period of 400 h. The sampling interval was chosen to be 0.5 h. Finally, 5–10% Gaussian noise was added to all the variables to replicate actual conditions where noise and measurement errors are present. In this study, the crucial parameters considered for prediction are Penicillin Concentration, Substrate Concentration and Biomass Concentration. The inputs chosen are substrate feed rate, substrate feed temperature, fermenter temperature, aeration rate, agitation power, and pH, dissolved oxygen concentration, CO_2 concentration, culture volume and generated heat.

Out of the 60 batches of data, 40 batches (32,000 data points) were chosen for training and the remaining 20 batches (16,000 data points) were chosen for testing. Similar to the previous case, the training dataset was artificially partitioned into labelled and unlabelled datasets keeping the proportion of unlabelled data around 25% of the total training data points.

After applying the 5-fold cross validation technique, as mentioned in the previous study, the best architecture obtained was 10-25-25-25-25-25-3. Unlabelled data-set containing 8000 data points was used to train the SOM in the unsupervised learning phase. The input datasets were projected onto a $[5 \times 5]$ gird for the Penicillin process. The synaptic weights in the SOM were used as the initial weights in the supervised learning phase.

The neural network for the Penicillin process was made adaptive by varying the value of μ between 10^{-5} and 10^{-7} with a step size of 10. For the Penicillin process, the network is observed to have best generalization at 660 epochs, after which the testing error begins to increase as observed in Fig. 7. This was fixed as the maximum number of epochs for back propagation.

On performing Bayesian regularization, shrinkage of the network was performed by obtaining 2653 effective parameters out of 2950 (maximum number of parameters) parameters of the 10-25-25-25-25-25-3 network. As in the previous case Root Mean Square Error (RMSE) was used as a criterion to gauge soft sensor performance. The accuracy of the network can be visualized by comparing actual and predicted values as seen in Figs. 8–10.

Table 2
Comparison of RMSE values of different soft sensors for the Penicillin process.

Variable	RMSE (SVR)	Supervised DNN	Semi-supervised DNN	% Improvement (1&3)
Penicillin	0.26	0.031	0.0274	89.46
Biomass	0.74	0.017	0.0069	99.06
Substrate.	0.15	0.003	0.00063	99.58

The estimation ability of the semi-supervised DNN based soft sensor was compared to SVR and supervised DNN based soft sensors. Parametric values for SVR are mentioned in Appendix C and were chosen in the exact manner as in the previous case. Supervised DNN utilized the same architecture as semi supervised DNN, but initialized weights randomly. Table 2 shows the RMSE values for the three variables estimated by the soft-sensors. Similar to the previous case, here also, deep learning based soft sensors yield lower RMSE values compared to SVR. However, in the present case, there is a remarkable reduction in RMSE, which can be attributed to the higher number of input and output variables than the previous case. Also, in the present case, the size of unlabelled dataset is more and hence by incorporating this additional information the performance of semi-supervised DNN is further improved. It can be observed that in spite of using the same architecture, supervised DNN achieves an inferior performance compared to semi-supervised DNN because it totally ignored the profuse information contained in unlabelled data. Nevertheless, supervised DNN still outperforms SVR based soft sensor.

From Figs. 8–10, it can be seen that semi-supervised DNN based soft sensor accurately characterizes the shifting dynamics within various phases i.e. – the lag phase (0–50 h), exponential phase (50–300 h) and stationary phase (300–400 h). However, it's superior generalization ability is best identified in the exponential phase, which is considered the most difficult phase to model since it involves rapid Penicillin production, causing the variables to change sharply. To show the distribution of predictions given by the soft sensors, a 5% error band is created around the actual values. The percentage of total predictions that fall within the 5% error range are given in Table 3. It can be observed from Figs.8–10 and Table 3, that a higher number of predictions fall within the 5% error band for Semi-Supervised DNN as compared to SVR and Supervised

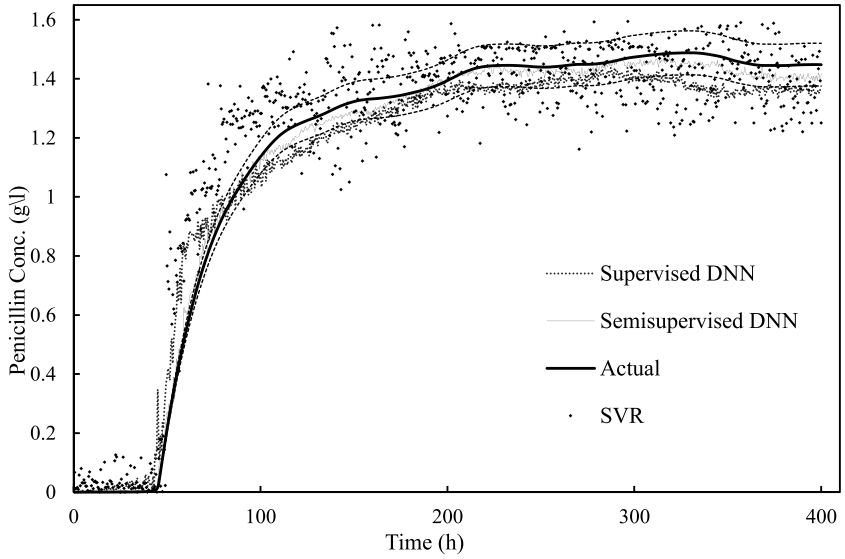

Fig. 8. Comparison of actual and predicted values by DNN based soft-sensor for Penicillin concentration.

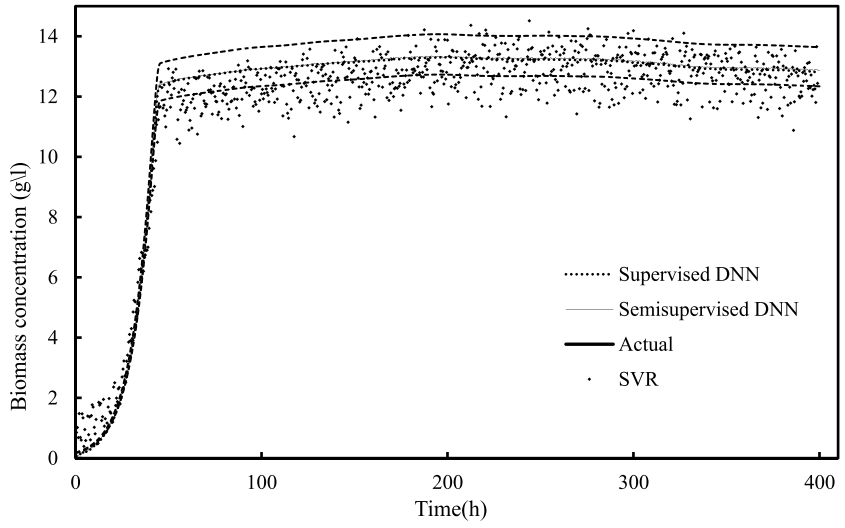

Fig. 9. Comparison of actual and predicted values by DNN based soft-sensor for Biomass concentration.

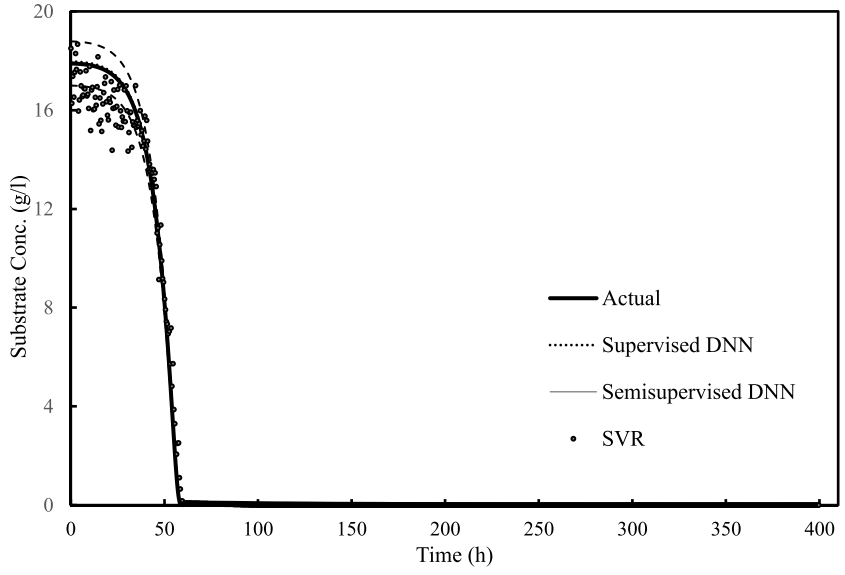

Fig. 10. Comparison of actual and predicted values by DNN based soft-sensor for Substrate concentration.

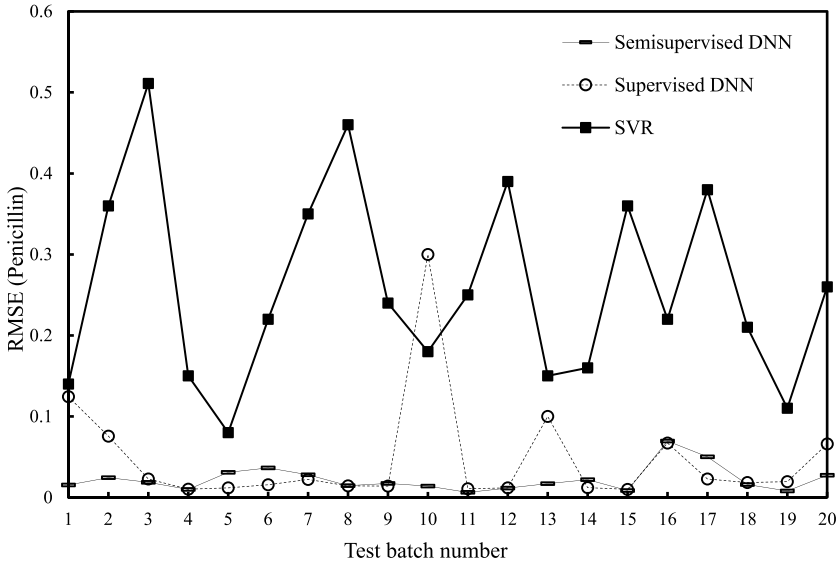

Fig. 11. Distribution of RMSE values across the test batches.

Table 3
Percentage of predictions within 5% of actual values.

Variable	Semi-supervised DNN	Supervised DNN	SVR
Penicillin	80.3%	60.5%	41.2%
Biomass	96.3%	82%	58.7%
Substrate	86.1%	83.5%	71.5%

DNN. In industries, the predictions made by soft sensors act as feedback signals to Advanced Process Control (APC) systems employed to optimize production yield and maintain product specifications. Faulty predictions with high errors seriously deteriorate the control action, and thereby have an adverse effect on both product quantity and quality. Hence the ability of the semi-supervised deep learning technique to generate lower no of estimations with high errors is an attractive feature from an industrial point of view.

The RMSE values of the developed soft sensors for Penicillin concentration across the 20 test batches are plotted in Fig. 11. It can be noticed that the RMSE values given by SVR and supervised DNN show higher variation across the test batches. On the contrary, the RMSE values given by semi-supervised DNN are consistently low and vary less across different test batches. Stable predictions across different test charges suggest that the semi-supervised DNN based soft sensor works reliably under various operational modes and adapts well to process transitions between different operating regimes.Therefore, it is a reliable modelling strategy for live bioprocesses which are inherently dynamic in nature.

4.3. Analysis of computational affordability

A soft sensor should be computationally efficient to make it feasible to implement for industrial purposes. Hence analysis of computational affordability is a matter of importance in soft sensor development. In this section the computational viability of the soft sensor is evaluated for the offline development phase and the online prediction phase.

During the offline development phase, majority of the time is spent on finding the optimal model hyper parameters. Although for both DNN and SVR, choosing optimal or near optimal hyper parameters requires expertise, it was observed that selecting hyper parameters for SVR was much more cumbersome due to its slower training. SVR obtains a separation hyper plane by solving a Quadratic Programming (QP) equation which makes it computationally expensive. This can also be observed from Table 4

Table 4
Average run-time values for semi-supervised DNN, supervised DNN and SVR based soft sensors.

Process	Model	Training (s)	Testing (s)	Background (s)
Streptokinase	Semi-Supervised DNN	198 Unsupervised – 8 Supervised – 190	~ 0.2	16
	Supervised DNN	200	~ 0.2	16
	SVR	215	3	40
Penicillin	Semi-Supervised DNN	485 Unsupervised – 15 Supervised – 480	1	25
	Supervised DNN	600	1	25
	SVR	865	8	60

which shows the average values of time taken to train and test the final model when the simulations were run on a PC with 1.6 GHz dual-core Intel Core i5 configuration. It can be seen that semi-supervised DNN takes the least amount of training time in both the case studies. However, it should be observed that in the Penicillin process, which involves higher number of data points, the difference in training time values between SVR and Semi-supervised DNN is much more, a reduction by 44%, compared to the difference in the Streptokinase process, a reduction by 8%. This can be explained by the fact that the training complexity for DNN depends linearly on n, the number of samples, whereas for SVR it depends on the order of $n^2 - n^3$ [34]. This suggests that DNN's would achieve a higher computational efficiency than SVR when dealing with large volumes of data. Therefore, advancements in data sampling and storage techniques coupled with increased dependency on data would make deep neural networks a preferred choice for soft sensor modelling in future.

The effect of unsupervised learning on the training time can also be observed from Table 4. It can be seen that for the supervised learning (back propagation) phase, semi supervised DNN takes 10 s less for Streptokinase and 120 s less for Penicillin when compared to supervised DNN. Learning error curves, during the back propagation step, of semi supervised and supervised DNNs for the Penicillin process are plotted in Fig.12. Error convergence curves for the Streptokinase process have not been plotted since there

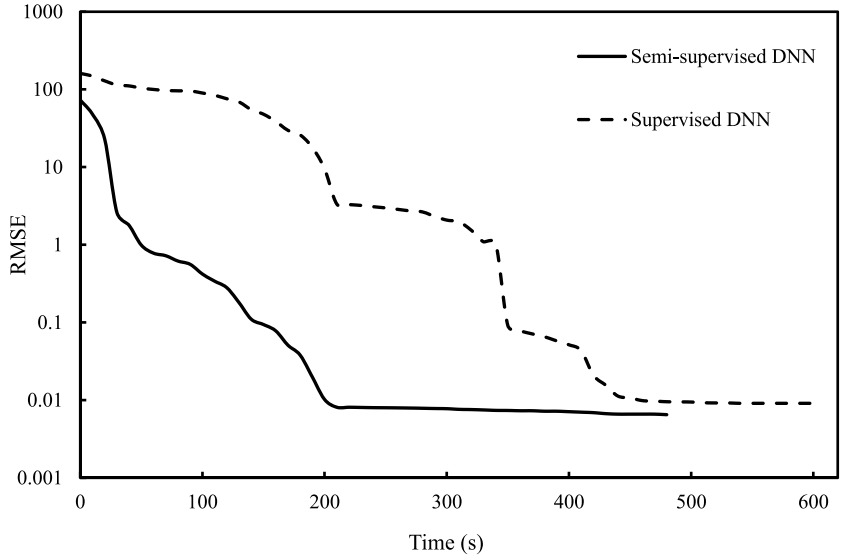

Fig. 12. Learning error curves during backpropagation for the Penicillin process.

is only a marginal improvement of 10 s in the backpropagation training time. Fig. 12 shows that during the backpropagation step, semi-supervised DNN converges faster than supervised DNN. Unsupervised learning results in better initialization of weights, bringing them closer to an optimum value, and thereby speeding up convergence.

As discussed earlier, soft sensor predictions act as feedback signals to Advanced Process Control (APC) systems installed in industries. In order to ensure real-time control, the online computation time i.e. the time taken by the soft sensor to predict a single set of outputs should be less than the sampling interval of the controller. Hence the computational complexity of a soft sensor is an important issue from an industrial perspective. The online computation time values for the different models are given in column 4 of Table 4. It can be seen that in both the case studies, DNN based soft-sensors make faster predictions than SVR based soft-sensors. For nonlinear models, computational complexity depends linearly on the number of nonlinear units. In this study, since the number of neurons utilized by DNNs is much lower than the number of support vectors utilized by SVR models, the computational complexity and hence online computation time is lower for DNN based soft sensors. However, online computation time values for supervised and semi supervised DNN's are equal since they possess the same architecture. This result indicates that semi supervised DNN based soft-sensors would be more preferred when faster control is required.

5. Conclusion

This paper introduces deep learning into data driven soft sensor modelling for nonlinear bioprocesses. The proposed biosensor was employed to estimate crucial parameters which are usually hard to measure on-line. Unlabelled data in the training set was used to carry out unsupervised learning using Self Organized Maps (SOM). The problem of over fitting was mitigated by applying Bayesian regularization and early stopping techniques.

Simulation results show that semi supervised DNN based soft sensor gave superior overall performances i.e. in terms of generalization ability, reliability and computational efficiency, in both the case studies. However, it performed remarkably better in the Penicillin process, which had more input – output variables, higher number of data points and a larger unlabelled dataset. This indicates that deep neural networks are more suitable for data intensive systems than shallow architectures. Furthermore, the effect of semi-supervised learning on overall soft sensor performance was observed. By taking into account pre-trained weights from unlabelled data, through semi-supervised learning, deep neural networks make better use of process data and hence achieve better representational ability. Also, un-supervised learning leads to effective initialization of weights, thereby making the backpropagation step faster. The semi supervised deep learning technique is therefore a promising alternative for soft sensor modelling particularly for highly data driven bioprocesses involving complex bio-chemical reactions. In future work, deep learning based neural networks will be explored in advanced control applications for nonlinear bioprocesses.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A

Phenomenological model for Streptokinase process [29] is given as:

$$\frac{dx_t}{dt} = \mu x_a - \left(\frac{q}{v}\right) x_t \tag{19}$$

$$\frac{dx_a}{dt} = (\mu - K_d) x_a - \left(\frac{q}{v}\right) x_a \tag{20}$$

$$\frac{ds}{dt} = \left(\frac{-\mu x_a}{Y_X}\right) + \left(\frac{q}{v}\right)(s_{in} - s) \tag{21}$$

$$\frac{dl_a}{dt} = Y_M \mu X_a - \left(\frac{q}{v}\right) l_a \tag{22}$$

$$\frac{ds_t}{dt} = Y_p (\mu - K_d) x_a - K_p s_t - \left(\frac{q}{v}\right) s_t \tag{23}$$

$$\frac{dv}{dt} = q(t) \tag{24}$$

$$\mu = \mu_m \left(\frac{s}{K_s + s}\right) \left(\frac{K_I^b}{K_I^b + l_a}\right) \tag{25}$$

Where, x_t, x_a are concentrations of total and active biomass, respectively?

Table A1
Parameter values for the Streptokinase process.

Parameter	Value
a_0	$0.9959\ (l/h)$
a_3	$0.6499\ (l/h)$
K_p	$0.0005\ (l/h)$
s_{in}	$70\ (g/l)$
Y_p	$0.441\ (g/g)$
a_1	$-0.3037\ (l/h)$
b	2.39
K_I	$12.66\ (g/l)$
T	$12\,(h)$
Y_x	$0.15\ (g/g)$
a_2	$-1.3418(l/h)$
K_d	$0.020\ (l/h)$
K_s	$13.14\ (g/l)$
Y_M	$4.80\ (g/g)$
μ_m	$0.74\ (l/h)$

Table A2
Initial conditions for the Streptokinase process.

Parameter	Value
$l_a\,(0)$	$0\ (g/l)$
$x_t\,(0)$	$0.7\ (g/l)$
$s_t\,(0)$	$0\ (g/l)$
$x_a\,(0)$	$0.7\ (g/l)$
$s\,(0)$	$70\ (g/l)$
$v\,(0)$	$5\,(l)$

V is the volume of reactor, s is concentrations of substrate, s_t is a concentration of streptokinase, l_a is concentration of lactic acid.

Feed rate for the system that maximizes the streptokinase production is given by following nonlinear equation and it's useful for data generation.

$$q(t) = a_0 + a_1\left(\frac{t}{T}\right) + a_2\left(\frac{t}{T}\right)^2 + a_3\left(\frac{t}{T}\right)^3 \tag{26}$$

The initial conditions and parameter values for simulation of the Streptokinase process are given below.

Appendix B

Initial conditions and set point ranges for parameters in Penicillin process:

Table B1
Initial conditions for operational parameters in fed batch Penicillin process.

Parameter	Value
Penicillin Concentration	0 (g/l)
Biomass Concentration	0.05–0.15 (g/l)
Substrate Concentration	13–17 (g/l)
Dissolved O2 Concentration	1.05–1.25 (g/l)
CO2 concentration	0.5–1 (g/l)
pH	4.5–5.5
Fermenter Temperature	293–303 K
Volume	99–104 (l)
Generated Heat	0 (kcal)

Table B2
Set points for operational parameters in fed batch Penicillin process.

Parameter	Value
Substrate feed rate	0.035–0.048 (l/h)
Substrate feed temperature	294–298 (K)
Fermenter temperature	296–300 (K)
Aeration rate	8–9 (l/h)
Agitation power	28.5–31.5 (W)
pH	4.8–5.2

Appendix C

Optimal values for SVR specific parameters:
See Table C1

Table C1
Optimal parametric values for SVR.

System	ε	C	Kernel type	Kernel parameters
Streptokinase process	0.01, 0.02	100, 10	Exponential RBF	8,20
Penicillin process	0.01,0.01,0.001	10, 8, 15	Exponential RBF	30, 25, 28

Appendix D. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.bej.2018.04.015.

References

[1] M.A. Henson, Biochemical reactor modeling and control, IEEE Control Syst. Mag. 26 (2006) 54–62.
[2] S.A. Russell, P. Kesavan, J.H. Lee, B.A. Ogunnaike, Recusrive data based prediction and control of batch product quality.
[3] K. Desai, Y. Badhe, S.S. Tambe, B.D. Kulkarni, Soft-sensor development for fed-batch bioreactors using support vector regression, Biochem. Eng. J. 27 (2006) 225–239.
[4] J.L. Wang, T. Yu, C.Y. Jin, On-line estimation of biomass in fermentation process using support vector machine, Chin. J. Chem. Eng. 14 (2006) 383–388.
[5] Y. Liu, Z. Gao, P. Li, H. Wang, Just-in-Time Kernel learning with adaptive parameter selection for soft sensor modeling of batch processes, Ind. Eng. Chem. Res. 51 (2012).
[6] Y. Yao, F. Gao, A survey on multistage/multiphase statistical modeling methods for batch processes, Annu. Rev. Control. 33 (2009) 172–183.
[7] S. Birle, M.A. Hussien, T. Becker, Fuzzy logic control and soft sensing applications in food and beverage processes, Food Control 29 (2013) 254–269.
[8] D.M. Himmelblau, Accounts of experiences in the application of artificial neural networks in chemical engineering, Ind. Eng. Chem. Res. 47 (2008) 5782–5796.
[9] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[10] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127.
[11] Y. Bengio, O. Delalleau, N.L. Roux, The curse of highly variable functions for local kernel machines, Adv. Neural Inf. Process. Syst. 18 (2006) 107.
[12] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153–160.
[13] Y. Bengio, On the challenge of learning complex functions, Prog. Brain Res. 165 (2007) 521–534.
[14] Kaneko Hiromasa, Kimito Funatsu, Adaptive soft sensor model using online support vector regression with time variable and discussion of appropriate hyperparameter settings and window size, Comput. Chem. Eng. 58 (2013) 288–297.
[15] Z. Ge, Z. Song, Semisupervised bayesian method for soft sensor modeling with unlabeled data samples, AIChe J. 57 (2011) 2109–2119.
[16] Z. Ge, B. Huang, Z. Song, Nonlinear semisupervised principal component regression for soft sensor modeling and its mixture form, J. Chemom. 11 (2014) 793–804.
[17] C. Shang, F. Yang, D. Huang, W. Lyu, Data-driven soft sensor development based on deep learningtechnique, J. Process Control 24 (2013) 223–233.
[18] T. Henson, Self-organized formation of topologically correct feature maps, Biol. Cyber. 1 (1982) 59–69.

[19] M.R. Zadeh, S. Amin, D. Khalili, V.P. Singh, Daily outflow prediction by multi-layer perceptron with logistic sigmoid and tangent sigmoid activation functions, Water Resour. Manag. 24 (2010) 2673–2688.
[20] R. Lippmann, An introduction to computing with neural nets, IEEE ASSP Mag. 4 (1987) 4–22.
[21] G. Velasquez, A Distributed Approach to a Neural Network Simulation Program, University of Texas, El Paso, 1998.
[22] J.J. Moré, The Levenberg-Marquardt algorithm: implementation and theory, in: G.A. Watson (Ed.), Numerical Analysis, Springer, Germany, 1978, pp. 105–116.
[23] D.W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, J. Soc. Ind. Appl. Math. 11 (1963) 431–441.
[24] D.C. Plaut, Experiments on Learning by Back Propagation, Carnegie-Mellon University, 1986.
[25] D.J.C. MacKay, A practical Bayesian framework for back propagation networks, Neural Comput. 4 (1992) 448–472.
[26] C.M. Bishop, Bayesian Methods for Neural Networks, Aston University-Birmingham, 1992.
[27] F. Burden, D. Winkler, Bayesian regularization of neural networks, Methods Mol. Biol. 458 (2008) 25–44.
[28] Anirban Banerjee, Yusuf Chisti, U.C. Banerjee, Streptokinase—a clinically useful thrombolytic agent, Biotechnol. Adv. 22 (4) (2004) 287–307.
[29] Adinarayana Kunamneni, Thaer Taleb Abed Abdelghani, Poluri Ellaiah, Streptokinase—the drug of choice for thrombolytic therapy, J. Thromb. Thrombolysis 23 (1) (2007) 9–23.
[30] P.R. Patnaik, Improvement of the microbial production of streptokinase by controlled filtering of process noise, Process Biochem. 35 (1999) 309–315.
[31] Y. Zhang, Y. Teng, Y. Zhang, Complex process quality prediction using modified kernel partial least squares, Chem. Eng. Sci. 65 (2010) 2153–2158.
[32] Y. Hu, H. Ma, H. Shi, Enhanced batch process monitoring using just-in-time-learning based kernel partial least squares, Chemom. Intell. Lab. Syst. 123 (2013) 15–27.
[33] G. Birol, C. Undey, A. Cinar, A modular simulation package for fed-batch fermentation: penicillin production, Comput. Chem. Eng. 26 (2002) 1553–1565.
[34] L. Bottou, C.J. Lynn, Support Vector Machine Solvers, Large Scale Kernel Machines, MIT Press, USA, 2007, pp. 301–320.