Journal of Manufacturing Engineering, March 2015, Vol. 10, Issue. 1, pp 35-38

A PARTICLE SWAM OPTIMISATION WITH STACKED NEURAL NETWORK APPROACH FOR BATCH PROCESSES MANUFACTURING

[*]Manoj Kumar[1], Jyoti Raman[2] and Priya[3]

[1-3]International Engineering Services, H.No.- 87A, RZI – Block, West Sagarpur, New Delhi – 110046

ABSTRACT

An optimal control strategy for batch processes manufacturing using particle swam optimisation (PSO) and stacked neural networks is presented in this paper. Stacked neural networks are used to improve model generalisation capability, as well as provide model prediction confidence bounds. In order to improve the reliability of the calculated optimal control policy for batch processes manufacturing, an additional term is introduced in the optimisation objective function to penalise wide model prediction confidence bounds. PSO can cope with multiple local minima and could generally find the global minimum. Application to a simulated fed-batch process demonstrates that the proposed technique is very effective.

Keywords*: Batch processes, Neural networks, Particle swam optimisation, Reliability.*

1. Introduction

Batch or semi-batch processes are suitable for the responsive manufacturing of high value added products [1]. To maximise the profit from batch process manufacturing, optimal control should be applied to batch processes. The performance of optimal control depends on the accuracy of the process model. Developing detailed mechanistic models is usually very time consuming and may not be feasible for agile responsive manufacturing. Data based empirical models, such as neural network models [2] and nonlinear partial least square models [3], and hybrid models [4] have to be utilised. Stacked neural networks have been shown to possess better generalisation capability than single neural networks [5,6] and are used in this paper to model batch processes. An additional feature of stacked neural networks is that they can also provide prediction confidence bounds indicating the reliability of the corresponding model predictions [7]. Due to model-plant mismatches, the "optimal" control policy calculated from a neural network model may not be optimal when applied to the actual process [8]. Thus it is important that the calculated optimal control policy should be reliable.

Conventional gradient base optimisation techniques are not effective to deal with objective functions with multiple local minima and can be trapped in local minima. Particle swam optimisation (PSO) is a recently developed optimisation technique that can cope with multiple local minima. This paper proposes using PSO and stacked neural networks to find the optimal control policy for batch processes. A standard PSO algorithm and three new PSO algorithms with local search were developed. In order to enhance the reliability of the obtained optimal control policy, an additional term is added to the optimisation objective function to penalise wide model prediction confidence bounds.

2. Particle Swarm Optimisation

PSO was first proposed by Kennedy and Eberhart [9]. The main principle behind this optimisation method is communication. In PSO there is a group of particles that look for the best solution within the search area. If a particle finds a better value for the objective function, the particle will communicate this result to the rest of the particles.

All the particles in the PSO have "memory" and they modify these memorized values as the optimisation routine advances. The recorded values are: velocity (*V*), position (*p*), best previous performance (*pbest*) and best group performance (*gbest*). The velocity describes how fast a particle should move from its current position which contains the coordinates of the particle. The last two parameters are the recorded best values that have been found during the iterations. A simple PSO algorithm is expressed as [9]:

$$V(k+1)=wV(k)+C_1r(pbest(k)-p(k))+C_2r(gbest(k)-p(k)) \quad (1)$$

$$p(k+1) = p(k) + V(k+1) \quad (2)$$

where w is the halt parameter, C_1 is the personal parameter, C_2 is the group parameter and r is a random number between 0 and 1.

**Corresponding Author - E- mail: kumarm1968@rediffmail.com*

The parameters w, C_1 and C_2 play important roles in PSO. The halt parameter (w) helps the particles to move around the search area. If it is too large the particles may miss the solution and if it is too small they may not reach it. Good values are usually slightly less than 1 [9]. The coefficients C_1 and C_2 indicate the preference of the particles for personal or communal results. If the value of C_1 is larger than C_2 then the particles will search for the best value within the best results obtained during its own search; they will not try to reach a communal best point. If vice versa, the particles will not perform individual searches, this will diminish the ability of the particles to perform "adventurous" searches. Kennedy and Eberhart [9] recommended that these values should be 2. This keeps a balance between the personal and communal search.

Four PSO algorithms were developed here and they perform different ways to communicate search results within the community. The first one is the simplest code presented in [9], where the particles have the ability to communicate its result to all the members of the community. The other three are based on local searches performed within small groups formed in the community. In the second algorithm, the group is based on a circular community [10]. These small groups will only communicate with members of their own community. The expected result with this formation is that the particles will search more intensively the solution area. In the third algorithm, local search is presented as a cluster community. The difference with the circular community is the fact that only one particle will communicate and compare the results with members of other groups. The fourth algorithm performs a geographical search in that the particles will communicate with the particles that are close to them in the solution area. The expected results are that the local search algorithms explore more intensively the search area.

The algorithms were then tested on the following two optimisation problems with multiple local minima or maxima:

$$\max F = \frac{1}{0.1+\left(\sum_{i=1}^{2}\frac{x_i^2}{4000}-\prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1\right)} \quad (3)$$

$$\min F = 20+x_1^2+x_2^2-50\left(\cos 2\pi x_1+\cos 2\pi x_2\right) \quad (4)$$

All the four PSO algorithms can find the global optimal solutions whereas the gradient based optimisation algorithm from the MATLAB Optimisation Toolbox, *fminunc*, fails to find the global optimal solutions when the initial values are not close to the global optimal solutions.

3. Modelling of A Fed-Batch Process Using Neural Networks

3.1 A Fed-Batch Process

The fed-batch reactor used in this work was taken from [11]. The batch reactor is based on the following reaction system:

$$A+B \xrightarrow{k_1} C$$

$$B+B \xrightarrow{k_2} D$$

This reaction is conducted in an isothermal semi-batch reactor. The desired product in this system is C. The objective is to convert as much as possible of reactant A by the controlled addition of reactant B, in a specified time t_f = 120 min. It is not appropriate to add all B initially because the second reaction will take place, increasing the concentration of the undesired by-product D. Therefore, to keep a low concentration of product D and at the same time increase the concentration of product C, the reactant B has to be fed in a stream with concentration b_{feed} = 0.2. A mechanistic model for this process can be found in [11].

3.2 Modelling the Fed-Batch Process Using Stacked Neural Networks

Neural network models for the prediction of the amount of desired product $C_C(t_f)V(t_f)$ and the amount of undesired by-product $C_D(t_f)V(t_f)$ at the final batch time are of the form:

$$y_1 = f_1(U) \quad (5)$$

$$y_2 = f_2(U) \quad (6)$$

where $y_1 = C_C(t_f)V(t_f)$, $y_2 = C_D(t_f)V(t_f)$, $U = [u_1\ u_2\ \ldots\ u_{10}]^T$ is a vector of the reactant feed rates during a batch, f_1 and f_2 are nonlinear functions represented by neural networks.

For the development of neural network models simulated process operation data from 50 batches with different feeding profiles were generated using the mechanistic model of the process. In each batch, the batch duration is divided into 10 equal stages. Within each stage, the feed rate is kept constant. The control policy for a batch consists of the feed rates at these 10 stages.

In the stacked neural network models several individual networks are trained using bootstrap re-sampling of the original data. The individual network outputs are combined to give the final model output. For each of the stacked neural network models, a group of thirty individual neural networks were developed. Each neural network contains in the hidden layer three nodes. The number of hidden nodes was selected based on the performance on the testing data. The nodes in the hidden layer use a hyperbolic tangent activation function while

that in the output layer uses a linear activation function. The stacked neural network output is taken as the average of the individual network outputs.

Fig. 1 and Fig. 2 show, respectively, the performance of individual networks and stacked networks for predicting the amount of desired product $C_C(t_f)V(t_f)$ on the training and unseen validation data sets. Fig. 1 indicates that in some networks the SSE on the training data is small but this is not the case on the unseen validation data. These results show that individual networks are not reliable. It can be seen from Fig. 2 that stacked networks give consistent performance on the training data and on the unseen validation data. The performance gradually improves when more networks are combined and approaches a stable level. This is observed on both the training and unseen validation data. This result indicates that the stacked model for predicting the amount of desired product $C_C(t_f)V(t_f)$ is more reliable as the number of individual networks is increased. It does not matter if some networks do not have a good performance, what matters is the communal performance of the group.

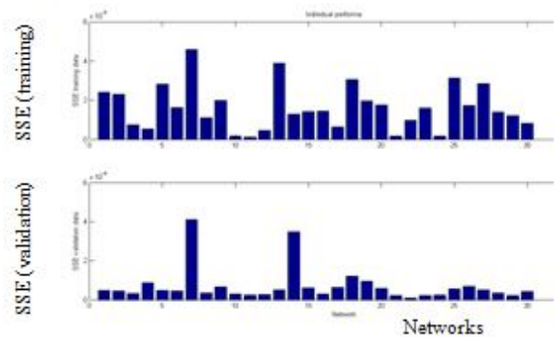

Fig. 1 Performance of individual networks for predicting $C_C(t_f)V(t_f)$

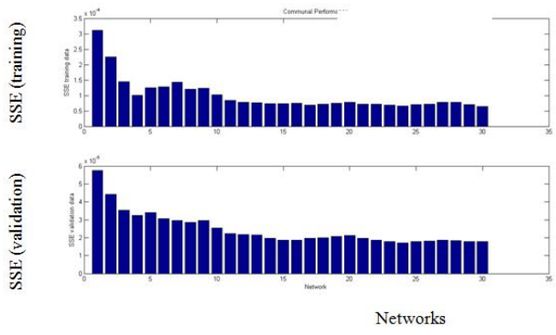

Fig. 2 Performance of stacked networks for predicting $C_C(t_f)V(t_f)$

4. Optimising Control Using PSO

The objective of the optimisation is to maximise the amount of the final product while reducing the amount of the by-product. The optimisation problem solved in this work is:

$$\min_U J = \{\alpha_1[D](t_f) - \alpha_2[C](t_f)\}V(t_f)$$

s.t.

$$0 \le u_i \le 0.01, \qquad (i = 1, 2, \cdots, m)$$

$$V(t_f) = 1$$

where α_1 and α_2 are weighting parameters which were both set to 0.5 in this study, U is a vector of control actions (reactant feed rates), and V is the reaction volume.

Table 1 lists the parameters used in global PSO (PSO_{G1} to PSO_{G4}) and local PSO (PSO_{L1} to PSO_{L4}) algorithms. For the local PSO algorithms, the sizes of the internal communities were kept the same in all the cases: 17 particles.

Table 1. Parameters used in PSO algorithms

	PSO_{G1}	PSO_{G2}	PSO_{G3}	PSO_{G4}	PSO_{L1}	PSO_{L2}	PSO_{L3}	PSO_{L4}
Particles	50	70	50	70	20	40	20	40
Halt	0.01	0.01	0.005	0.005	0.01	0.01	0.005	0.005

For the purpose of comparison, optimisation using a single neural network was first carried out. Table 2 shows the obtained results. As can be seen from the table, the values for the difference between the final amounts of product and by-product using the PSO codes were similar to the ones obtained using the MATLAB Optimisation Toolbox function, *fmincon*, in this fed-batch reactor. However, PSO can cope with multiple local minima in general as shown in Section 2.

It can also be appreciated that an increment in the number of particles in the global version of the PSO code does not help the code to find a better solution for the optimization problem. This could indicate that the PSO code only needed a minimum number of particles and the inclusion of more particles will not be helpful. A different behaviour was encountered in the local version of the PSO. When more particles were used for the solution of the problem, then the code required less number of iterations to solve the problem.

Changing the value of the halt did not show any improvement in the performance. As can be seen from the table, the results obtained using different halt values are similar. Another difference that could be seen between the two PSO codes is the fact that the local

version can find a similar answer to the problem using fewer particles than the global version of the PSO code.

Once the optimal feed rates were obtained, they were applied to the actual process (i.e. simulation by the mechanistic model of the process). Table 2 shows the difference between the amounts of the final product and by-product on neural network model and the actual process. It can be seen from Table 2 that the actual amounts of product and by-product under these "optimal" control policies are quite different from the neural network model predictions. This indicates that the single neural network based optimal control policies are only optimal on the neural network model and are not optimal on the real process. Hence, they are not reliable. This is mainly due to the model plant mismatches, which is unavoidable in data based modelling.

A method to overcome the impact of model plant mismatch on optimisation performance was previously investigated by Zhang [8] where model prediction confidence bounds are incorporated as a penalty in the objective function. Therefore, the objective function can be modified as

$$\min_U J = \{\alpha_1[D](t_f) - \alpha_2[C](t_f)\}V(t_f) + \alpha_3(stderr[C] + stderr[D])$$

s.t.

$$0 \le u_i \le 0.01, \qquad (i = 1,2,\cdots,m)$$

$$V(t_f) = 1$$

where *stderr*[*C*] and *stderr*[*D*] are the standard errors of the stacked models, α_3 is a weighting factor for model prediction confidence and was selected as 0.5 in this work.

Table 2. Values of ([*C*](t_f) - [*D*](t_f))*V*(t_f) on neural network models and actual process

	Single neural network		Stacked neural network	
	Neural network	Process	Neural network	Process
fmincon	0.0411	0.0314	0.0304	0.0363
PSO_{G1}	0.0400	0.0344	0.0296	0.0359
PSO_{G2}	0.0405	0.0319	0.0297	0.0370
PSO_{G3}	0.0399	0.0325	0.0302	0.0358
PSO_{G4}	0.0396	0.0347	0.0300	0.0368
PSO_{L1}	0.0377	0.0341	0.0298	0.0338
PSO_{L2}	0.0407	0.0307	0.0298	0.0364
PSO_{L3}	0.0394	0.0364	0.0297	0.0348
PSO_{L4}	0.0397	0.0301	0.0297	0.0363

Table 2 shows the results obtained using the new objective function with stacked neural network models. It can be seen from Table 2 that the modified objective function with stacked neural network models leads to better performance on the actual process. It can be appreciated that the actual performance is very close to the ones calculated using the stacked neural network. This demonstrates that control policies obtained using stacked neural networks considering model prediction confidence bounds is much more reliable than those obtained using a single neural network model

5. Conclusions

The study demonstrates that particle swam optimisation is a powerful optimisation technique, especially when the objective function has several local minima. Conventional optimisation techniques could be trapped in local minima but PSO could in general find the global minimum. Stacked neural networks can not only given better prediction performance but also provide model prediction confidence bounds. In order to improve the reliability of neural network model based optimisation, an additional term is introduced in the optimisation objective to penalize wide model prediction confidence bound. The proposed technique is successfully demonstrated on a simulated fed-batch reactor.

References

1. *Bonvin D (1998) , J. Process Control,Vol. 8, 355-368.*
2. *Zhang J (2005) , "Transactions Instrumental Measurement Control",Vol. 27, 391-410.*
3. *Zhao S J Zhang J Xu Y M (2006) "Industrial Engineering Chemistry Research",Vol. 45, 3843-3852.*
4. *TianY Zhang J Morris A J (2001), "Industrial Engineering Chemistry Research", Vol. 40, 4525-4535.*
5. *Sridhar D V Seagrave R C Bartlett E B (1996), "American Institute of Chemical Engineers Journal" (AIChE J)., Vol.42, 2529-2539.*
6. *Zhang J Martin E B Morris A J Kiparissides C (1997) "Computer Chemical Engineering",Vol. 21, s1025-s1030.*
7. *Zhang J (1999), Neurocomputing, Vol.25, 93-113.*
8. *Zhang J (2004), Ind. Eng. Chem. Res., Vol.43, 1030-1038.*
9. *Kennedy J Eberhart R (1995), "Particle Swarm Optimization", In Proceedings of the 1995 IEEE international conference on neural networks, Perth, Australia. Vol VI, 1942 – 1948.*
10. *Kennedy J Mendes R (2006), "IEEE Transacations System Man Cybernetics. Part C-Appl. Rev., Vol.36, 515-519.*
11. *Terwiesch P Ravemark D Schenker B Rippin D W T (1998), Computer Chemical Engineering,Vol. 22, 201-213.*

