International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:10, 2016

Performance Comparison of Different Regression Methods for a Polymerization Process with Adaptive Sampling

Florin Leon, Silvia Curteanu

***Abstract*—**Developing complete mechanistic models for polymerization reactors is not easy, because complex reactions occur simultaneously; there is a large number of kinetic parameters involved and sometimes the chemical and physical phenomena for mixtures involving polymers are poorly understood. To overcome these difficulties, empirical models based on sampled data can be used instead, namely regression methods typical of machine learning field. They have the ability to learn the trends of a process without any knowledge about its particular physical and chemical laws. Therefore, they are useful for modeling complex processes, such as the free radical polymerization of methyl methacrylate achieved in a batch bulk process. The goal is to generate accurate predictions of monomer conversion, numerical average molecular weight and gravimetrical average molecular weight. This process is associated with non-linear gel and glass effects. For this purpose, an adaptive sampling technique is presented, which can select more samples around the regions where the values have a higher variation. Several machine learning methods are used for the modeling and their performance is compared: support vector machines, k-nearest neighbor, k-nearest neighbor and random forest, as well as an original algorithm, large margin nearest neighbor regression. The suggested method provides very good results compared to the other well-known regression algorithms.

***Keywords*—**Adaptive sampling, batch bulk methyl methacrylate polymerization, large margin nearest neighbor regression, machine learning.

I. Introduction

GENERALLY, polymerization processes have series of difficulties in modeling and optimization actions, because of their specific features, as well as the general characteristics of the chemical processes. From the last point of view, reactions are complex and, often, their phenomenology is not fully elucidated. Or, elaborating credible phenomenological models involves precise knowledge of physical and chemical laws that govern the process. Often, some approximations are required, affecting the accuracy of the model results. In addition, the complexity of mathematical models causes extra difficulties on how to solve them and, also, on the necessary time, which means the inability to use the models in on-line optimal control procedures. In these circumstances, empirical modeling based on input-output data becomes a preferable alternative to the phenomenological modeling, in terms of both methodology and result accuracy. Several examples are given.

The first part of article [1] is a review concerning the use of artificial neural networks (ANNs) in polymerization reaction engineering, focusing on different types of methodologies and applications.

In the chemical engineering area, and especially for the polymerization processes, ANNs were applied to a diversity of processes, included in various methodologies. Some examples are the following: direct and inverse modeling of free radical polymerization of methyl methacrylate [1], [2], development of a virtual soft sensor in the polyethylene terephthalate production process [3], modeling the styrene living radical polymerization mediated by 2,2,6,6-tetramethyl-1-piperidinoxyl [4], selection of mixture initiators for batch polymerization [5], modeling the free radical polymerization of styrene [6], reaction temperature prediction during the styrene polymerization [7], fluorescence modeling of the polydimethylsiloxane/silica composites containing lanthanum [8] and the list remains opened.

Free radical polymerization of methyl methacrylate is considered the case study of this approach. The gel, glass and cage effects are exhibited in the bulk polymerization of MMA. The gel effect arises because of the decrease in termination rate constant at high monomer conversion, associated with increased diffusional resistance to the growing radicals. It is manifested as a sudden increase in conversion, as well as in the gravimetrical average molecular weight with time, after some polymerization has occurred. Similarly, the glass effect is associated with the decrease of propagation rate constant, due to increased diffusional resistance to the movement of the monomer toward a growing radical. This leads to the polymerization stopping short of complete monomer conversion, even though the reactions are irreversible. A special adaptive technique is applied in modeling to take into account the high variations of some parameters in a short time.

Monomer conversion (x), numerical average molecular weight (Mn), gravimetrical average molecular weight (Mw) are determined as function of reaction conditions (initiator concentration, I_0, temperature, T, and time, t).

II. Adaptive Sampling

The free radical polymerization of methyl methacrylate achieved in a batch bulk process is associated with non-linear gel and glass effects, i.e. it has regions where the values of

F. Leon is with Department of Computer Science and Engineering, "Gheorghe Asachi" Technical University of Iaşi, Bd. Mangeron 27, 700050 Iaşi, Romania (e-mail: fleon@cs.tuiasi.ro)

S. Curteanu is with the Department of Chemical Engineering, "Gheorghe Asachi" Technical University of Iaşi, Bd. Mangeron 73, 700050 Iaşi, Romania (e-mail: scurtean@ch.tuiasi.ro)

some parameters (conversion and molecular weight) have a higher variation. The use of a constant sampling method may miss these regions. In order to address this problem, an adaptive sampling technique is proposed, which can select more samples around the critical regions.

The adaptive sampling algorithm is presented in Pseudocode I. It first computes the local differences between successive points in the *y* function, which can stand for either of the three considered quantities: monomer conversion (*x*), numerical average molecular weight (*Mn*) and gravimetrical average molecular weight (*Mw*). This difference has the same meaning as a derivative. Then, it uses a running sum to determine whether the next point will be sampled. In regions with approximately constant values, the derivative is small, therefore the space between the sampled points will be larger. In regions with high variation, the derivative will be higher, and thus more points will be sampled.

PSEUDOCODE I
THE ADAPTIVE SAMPLING ALGORITHM

```
procedure AdaptiveSampling
input: y: the array of all points representing functions x, Mn or Mw
output: samples: the array of sampled points
begin
  for i in 1 .. length(y) - 1 do
    diff(i) = abs( y(i+1) - y(i) )
  end for
  avg = Select(diff, typeOfFunction)
  samples(1,1) = 1
  samples(1,2) = y(1)
  sum = 0
  count = 1
  for i in 1 .. length(diff) do
    sum = sum + diff(i) ^ alpha
    if sum > avg then
      samples(count, 1) = i
      samples(count, 2) = y(i)
      count = count + 1
      sum = 0
    end if
  end for
  return samples
end procedure
```

In the pseudocode, the *alpha* constant is used, which balances the importance of the local gradient. In our case, its value was set to 0.8. Also, the *avg* variable is initialized by the SELECT function, which returns different values by taking into account the target quantity and the values of the *diff* array. This variable controls the number of the sampled points. The lower it is; the more points will be sampled. Its value was empirically determined, thus: in case of monomer conversion, $avg(x) = 1000 \cdot \overline{diff}$, in case of numerical average molecular weight, $avg(Mn) = 500$, and in case of gravimetrical average molecular weight, $avg(Mw) = 36 \cdot \overline{diff}$, where $\overline{diff}$ stands for the mean value of the *diff* array.

Figs. 1-3 present the adaptive sampling results for the three considered functions. The sampled points are marked with circles. In all cases, a number of points between 300 and 400 were obtained.

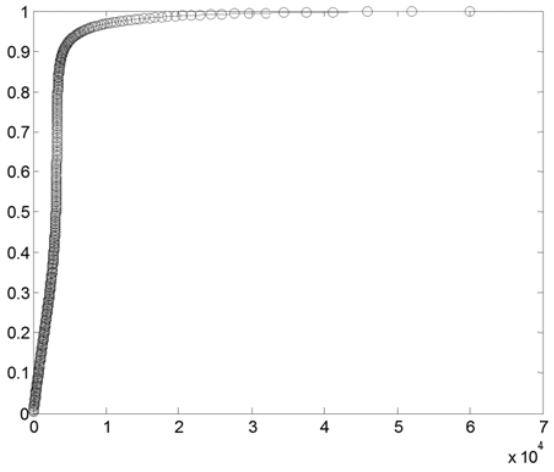

Fig. 1 Adaptive sampling results for monomer conversion (*x*) vs. time: 326 points

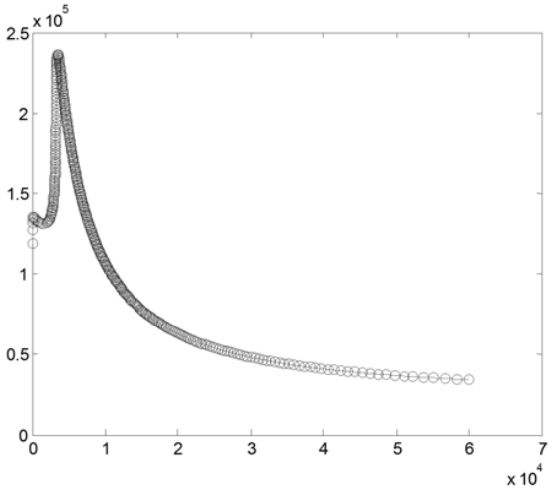

Fig. 2 Adaptive sampling results for numerical average molecular weight (*Mn*) vs. time: 348 points

Before applying the machine learning algorithms, a preprocessing stage was also used: the values of the *samples* matrix were normalized by columns, such as all values should lie in the [0,1] domain. These values will constitute the dataset used for regression.

For each problem, the order of the vectors was randomized and then, two thirds were selected for the training set and the remaining third was selected for the testing set. In order to have a meaningful comparison, all the regression algorithms applied use the same training and testing sets. The coefficient of determination (R^2), the squared coefficient of correlation, is used as a metric to compare the performance of different models.

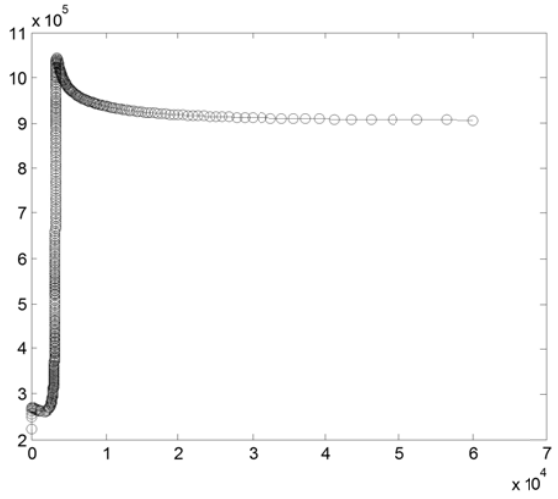

Fig. 3 Adaptive sampling results for gravimetrical average molecular weight (*Mw*) vs. time: 392 points

III. Regression by Established Algorithms

First, various algorithms implemented in the popular collection of machine learning algorithms Weka [9] were evaluated on the three problems. The corresponding results will constitute a basis for comparison with our original algorithm, *Large Margin Nearest Neighbor for Regression* (LMNNR) [10], [11], whose results will be presented in the following section.

Tables I-III show the performance of the algorithms implemented in Weka for the training set alone and for the testing set. The results on the latter are more important, because they evaluate the generalization capabilities of the models. The results are sorted in decreasing order of the testing set results. The abbreviations in the tables are as follows: *ν-SVR* and *ε-SVR* stand for ν-Support Vector Regression and ε-Support Vector Regression, respectively, *RBF* stands for Radial Basis Function kernel, *P2* stands for polynomial kernel of the second degree, and *kNN* stands for k-Nearest Neighbors.

TABLE I
Performance of Different Algorithms Implemented in Weka for Monomer Conversion (*X*)

Algorithm	Training set	Testing set
kNN, k=10, 1/d	0.999800	0.999800
Random forest, N=100	0.999800	0.999800
REPTree	0.999200	0.998401
M5 rules	0.990423	0.986844
Additive regression	0.967666	0.968256
ν-SVR, RBF, C=10000	0.886611	0.824827
ε-SVR, RBF, C=10000	0.886987	0.822105
ν-SVR, RBF, C=100	0.801025	0.758815
ε-SVR, RBF, C=100	0.802816	0.754466
ν-SVR, P2, C=100	0.589363	0.329591
ν-SVR, P2, C=10000	0.586909	0.318773
ε -SVR, P2, C=100	0.585684	0.314160
ε -SVR, P2, C=10000	0.547156	0.219399

TABLE II
Performance of Different Algorithms Implemented in Weka for Numerical Average Molecular Weight (*Mn*)

Algorithm	Training set	Testing set
Random forest, N=100	0.999800	0.999200
kNN, k=10, 1/d	0.999400	0.998600
REPTree	0.997202	0.995405
M5 rules	0.970225	0.958637
Additive regression	0.946145	0.954138
ε-SVR, RBF, C=10000	0.812702	0.795307
ν-SVR, RBF, C=10000	0.802816	0.753250
ε-SVR, RBF, C=100	0.781633	0.744769
ε -SVR, P2, C=100	0.774576	0.732051
ε -SVR, P2, C=10000	0.774928	0.731196
ν-SVR, RBF, C=100	0.777571	0.729658
ν-SVR, P2, C=10000	0.776161	0.715039
ν-SVR, P2, C=100	0.775985	0.712505

TABLE III
Performance of Different Algorithms Implemented in Weka for Gravimetrical Average Molecular Weight (*Mw*)

Algorithm	Training set	Testing set
kNN, k=10, 1/d	0.999800	0.999600
Random forest, N=100	0.999800	0.999600
REPTree	0.999000	0.998001
M5 rules	0.987241	0.984064
Additive regression	0.967076	0.959420
ε-SVR, RBF, C=10000	0.441427	0.441294
ν-SVR, RBF, C=10000	0.428501	0.436392
ε-SVR, RBF, C=100	0.364937	0.362163
ν-SVR, RBF, C=100	0.340589	0.341991
ε -SVR, P2, C=10000	0.216970	0.214276
ν-SVR, P2, C=100	0.216970	0.213999
ν-SVR, P2, C=10000	0.216970	0.213906
ε -SVR, P2, C=100	0.216970	0.213444

For kNN, 10 neighbors were selected and the inverse distance weighting of instances were used. For Random Forest, 100 trees were used, and for SVM, two different values for the cost parameter were used: 100 and 10000.

It can be seen that the kNN and Random Forest are clearly better for the considered problems. It is somehow surprising that the Support Vector Machine models have such a poor performance in these cases.

IV. Regression by the Large Margin Nearest Neighbor Algorithm

In all neighbor-based methods, the distance metric is crucially important. That is why researchers have tried to not only use a single metric for all the problems, but to adapt the distance metric to the problem at hand, in order to have better performance. One way to adapt is was to use the idea of a large margin, one of the fundamental ideas of support vector machines. It was transferred to the kNN method for classification tasks [12]–[14], resulting in the large-margin nearest neighbor method (LMNN). In this case, learning involves the optimization of a convex problem, defined as:

$$\begin{aligned} \min \quad & \sum_{ij} \eta_{ij} d_{ij} + \lambda_h \sum_{ijl} \nu_{ijl} \xi_{ijl} \\ \text{such that} \quad & (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) = d_{ij} \\ & (\mathbf{x}_i - \mathbf{x}_l)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_l) - d_{ij} \geq 1 - \xi_{ijl} \\ & \mathbf{M} \succeq 0,\ \xi_{ijl} \geq 0, \forall i, j, l \end{aligned} \quad (1)$$

where $\eta_{ij} \in \{0,1\}$ is 1 only when $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$, $\nu_{ijl} = \eta_{ij}$ if and only if $y_i \neq y_l$ and 0 otherwise, d_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, ξ_{ijl} is the loss and $\lambda_h \geq 0$ is a constant. A "target" neighbor is an instance with the same class label, and an "impostor" is a neighbor with a different class label.

The LMNN method was adapted for regression, resulting in the *Large Margin Nearest Neighbor for Regression* (LMNNR) algorithm [10], [11]. It uses an objective function F, which is to be minimized, which takes into account 3 criteria:

$$F = w_1^F \cdot F_1 + w_2^F \cdot F_2 + w_3^F \cdot F_3, \quad (2)$$

where the weights of the criteria are normalized: $w_1^F + w_2^F + w_3^F = 1$.

In order to simplify the expressions of the F_i functions, let us make the following notations, where d_M means the weighted square distance function using the weights we search for: $d_{ij} = d_M(\mathbf{x}_i, \mathbf{x}_j)$, $d_{ik} = d_M(\mathbf{x}_i, \mathbf{x}_k)$, $g_{ij} = |f(\mathbf{x}_i) - f(\mathbf{x}_j)|$ and $g_{ik} = |f(\mathbf{x}_i) - f(\mathbf{x}_k)|$.

The first criterion is:

$$F_1 = \sum_{i=1}^{n} \sum_{j \in N(i)} d_{ij} \cdot (1 - g_{ij}), \quad (3)$$

where $N(i)$ is the set of the nearest k neighbors of instance i. This criterion says that the nearest neighbors of i should have similar values to the one of i, and more distant ones should have different values. It tries to minimize the distance between an instance i and its neighbors with similar values. If a neighbor j has a dissimilar value, the second factor, $1 - g_{ij}$, becomes small and the distance is no longer necessary to be minimized.

The second criterion is expressed as:

$$F_2 = \sum_{i=1}^{n} \sum_{j \in N(i)} \sum_{l \in N(i)} \max(1 + d_{ij} \cdot (1 - g_{ij}) - d_{ik} \cdot (1 - g_{il}), 0). \quad (4)$$

It takes into account a pair of neighbors, j and l, by analogy to a target and an impostor. We try to minimize the distance to the neighbors with close values (the positive term), while simultaneously trying to maximize the distance to the neighbors with distant values (the negative term). The value of this function criterion cannot be negative. Moreover, the distance to an instance with a dissimilar value should be larger than the distance to an instance with a close value. The large margin concept is applied by forcing these two distances to differ by at least 1. This is an arbitrary value and can be changed without affecting the optimization problem; it would only result in the scaling of the model weights.

The third criterion is used in general to prevent the values of the weights from becoming too large (however, it is not used for our particular case studies, because the weights of the model do not become very large anyway):

$$F_3 = \sum_{j=1}^{n_p} \sum_{i=1}^{n_i} m_{ii}(j). \quad (5)$$

The optimization of the objective function is performed by gradient descent, using an approximate differential method with the central difference definition of the derivative [15]. That is, for a small ε, the following relation holds, where the truncation error is $O(\varepsilon^2)$:

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}. \quad (6)$$

TABLE IV
THE BEST PERFORMANCE OF THE LMNNR ALGORITHM FOR THE THREE CONSIDERED PROBLEMS

Dataset	No. proto-types	No. optimization neighbors	No. regression neighbors	Training set	Testing set
Monomer conversion (x)	1	3	3	1	0.999952
	1	5	5	1	0.999953
	2	3	3	1	0.999952
	2	5	5	1	0.999953
Numerical average molecular weight (Mn)	1	3	3	1	0.999638
	1	5	5	1	0.999623
	2	3	3	1	0.999638
	2	5	5	1	0.999623
Gravimetrical average molecular weight (Mw)	1	3	3	1	0.999812
	1	5	5	1	0.999816
	2	3	3	1	0.999812
	2	5	5	1	0.999816

For all the following configurations, the algorithm was applied 10 times and the best result was recorded. Table IV presents the best performance of the LMNNR algorithm. Similar as for the algorithms implemented in Weka, the coefficient of determination R^2 was used as a performance measure.

V. CONCLUSIONS

Even if the optimization method of LMNNR requires repeated experiments for the same configuration, the results are very good and actually better than the results provided by well-established machine learning algorithms implemented in Weka. This shows that the model based on the combination of the k-nearest neighbor paradigm with the idea of the large margin has a great potential for regression problems.

The good quality of the results is also due to the adaptive sampling method, which gives the most relevant information

to the algorithms by selecting more sample points in the regions where the process has a higher variation and a smaller number of points in the more uniform regions.

ACKNOWLEDGMENTS

This work was supported by the "Partnership in priority areas – PN-II" programme, financed by ANCS, CNDI - UEFISCDI, project PN-II-PT-PCCA-2011-3.2-0732, no. 23/2012.

REFERENCES

[1] S. Curteanu, "Direct and Inverse Neural Network Modeling in Free Radical Polymerization", *Central European Journal of Chemistry*, vol. 2, no. 1, 2004, pp. 113–140.

[2] S. Curteanu, F. Leon and D. Gâlea, "Neural Network Models for Free Radical Polymerization of Methyl Methacrylate", *Eurasian Chemico-Technological Journal*, vol. 5, no. 3, 2003, pp. 225–231.

[3] J. C. B. Gonzaga, L. A. C. Meleiro, C. Kiang and R. Maciel Filho, "ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process", *Computers & Chemical Engineering*, vol. 33, no. 1, 2009**,** pp. 43–49.

[4] S. Contant, P. V. R. Mesa and L. M. F. Lona, "Modeling of Styrene Living Radical Polymerization Mediated by Tempo Using Neural Networks", Proceedings of the *2nd Mercosur Congress on Chemical Engineering and 4th Mercosur Congress on Process System Engineering*, ENPROMER, 2005, Rio de Janeiro, Brazil, 2005, pp. 1–10.

[5] F. A. N. Fernandes, "Selection of a mixture of initiators for batch polymerization using neural networks", *Journal of Applied Polymer Science***,** vol. 98, issue 5, 2005, pp. 2088–2093.

[6] S. Curteanu, F. Leon, R. Furtună, E. N. Drăgoi and N. Curteanu, "Comparison between Different Methods for Developing Neural Network Topology Applied to a Complex Polymerization Process", Proceedings of the *International Joint Conference on Neural Networks*, IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010, pp. 1293–1300.

[7] M. S. Leite, B. F. Dos Santos, L. M. F. Lona, F. V. Da Silva and A. M. Frattini Fileti, "Application of Artificial Intelligence Techniques for Temperature Prediction in a Polymerization Process", *Chemical Engineering Transactions*, vol. 24, 2011, pp. 385–390.

[8] A. Salman, A. P. Engelbrecht and M. G. H. Omran, "Empirical analysis of self-adaptive differential evolution", *European Journal of Operational Research*, vol. 183, issue 2, 2007, pp. 785–804.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update", *ACM SIGKDD Explorations*, vol. 11, no. 1, 2009, pp. 10–18.

[10] F. Leon and S. Curteanu, "Evolutionary Algorithm for Large Margin Nearest Neighbour Regression", Proceedings of the *7th International Conference on Computational Collective Intelligence Technologies and Applications*, ICCCI 2015, Madrid, Spain, Part I, *Lecture Notes in Artificial Intelligence, LNAI 9329*, Springer International Publishing Switzerland, doi: 10.1007/978-3-319-24069-5_29, 2015, pp. 286–296.

[11] F. Leon and S. Curteanu, "Large Margin Nearest Neighbour Regression Using Different Optimization Techniques", *Journal of Intelligent and Fuzzy Systems*, IOS Press, in press

[12] K. Q. Weinberger, J. Blitzer and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification", *Advances in Neural Information Processing Systems*, vol. 18, MIT Press, Cambridge, MA, USA, 2006, pp. 1473–1480.

[13] K. Q. Weinberger and L. K. Saul, "Fast solvers and efficient implementations for distance metric learning", Proceedings of the *25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 1160–1167.

[14] K. Q. Weinberger and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification", *Journal of Machine Learning Research*, vol. 10, 2009, pp. 207–244.

[15] H. Jeffreys and B. S. Jeffreys, "Central Differences Formula", in *Methods of Mathematical Physics*, 3rd edition, Cambridge University Press, 1988, pp. 284–286.