Journal of Engineering Science and Technology
Vol. 13, No. 8 (2018) 2342 - 2351

DEVELOPMENT OF *R* PACKAGE AND EXPERIMENTAL ANALYSIS ON PREDICTION OF THE CO_2 COMPRESSIBILITY FACTOR USING GRADIENT DESCENT

LALA SEPTEM RIZA[1,*], DENDI HANDIAN[1],
RANI MEGASARI[1], ADE GAFAR ABDULLAH[2],
ASEP BAYU DANI NANDIYANTO[3], SHAH NAZIR[4]

[1]Department of Computer Science, Universitas Pendidikan Indonesia, Jl. Dr. Setiabudi 229, Bandung 40154, Indonesia
[2]Departmen Pendidikan Teknik Elektro, Universitas Pendidikan Indonesia, Jl. Dr. Setiabudi 229, Bandung 40154, Indonesia
[3]Departmen Kimia, Universitas Pendidikan Indonesia, Jl. Dr. Setiabudi 229, Bandung 40154, Indonesia
[4]Department of Computer Science, University of Swabi, Swabi, Pakistan

*Corresponding Author: lala.s.riza@upi.edu

Abstract

Nowadays, many variants of gradient descent (i.e., the methods included in machine learning for regression) have been proposed. Moreover, these algorithms have been widely used to deal with real-world problems. However, the implementations of these algorithms into a software library are few. Therefore, we focused on building a package written in R that includes eleven algorithms based on gradient descent, as follows: Mini-Batch Gradient Descent (MBGD), Stochastic Gradient Descent (SGD), Stochastic Average Gradient Descent (SAGD), Momentum Gradient Descent (MGD), Accelerated Gradient Descent (AGD), Adagrad, Adadelta, RMSprop and Adam. Additionally, experimental analysis on prediction of the CO_2 compressibility factor were also conducted. The results show that the accuracy and computational cost are reasonable, which are 0.0085 and 0.142 second for the average of root mean square root and simulation time.

Keywords: Gas compressibility factor, Machine learning, Regression, R programming language.

1. Introduction

There are many problems solved by utilizing gradient descent and its variants. For example, research conducted by Klein et al. [1] used a variant of gradient descent, which is adaptive stochastic gradient descent, for image registration. In literature [2], gradient descent was used for maximizing sharpness on moving objects in scanning electron microscope. The research focused on segmentation and restoration of incomplete characters, such as 1000 ancient Hebrew characters in 8^{th} – 7^{th} century BCE, which has been done by calculating with gradient descent [3]. Furthermore, gradient methods are mostly embedded and used for optimization in other methods. Elastic averaging stochastic gradient descent is used for optimization in deep learning in the parallel computing [4]. Estimation of support vector machine parameters can be done by using gradient descent based algorithms [5]. The algorithm gradient descent was implemented to construct fuzzy rule-based systems in the software library "*frbs*" [6, 7].

It can be seen that gradient descent and its variants have been used and improved for dealing with many fields. Basically, gradient descent is a method to search a local minimum of an objective function, which is represented by a stationary point with gradient of zero [8]. Implementations of methods based on gradient descent in machine learning are included in supervised learning, which is regression. A model of gradient descent is represented in a linear one expressing a map between input and output variables. To improve the accuracy and computational cost, many researchers have been proposing new variants of the algorithm, such as stochastic gradient descent (SGD) [9], Adadelta [10], and Adaptive Subgradient Method (Adagrad) [11]. Even though there are more than 10 algorithms based on gradient descent, a few software libraries have been built. As a book library that collects a lot of books so that people can be easy to find them, a software library needs to implement so that people who do not understand how to create a program can be easy to use the algorithms for tackling their problems.

Therefore, this research is aimed to develop a software library that contains many algorithms based gradient descent to deal with regression tasks. Basically, we develop the previous package, which is "gradDescentR" [12] that implements four algorithms based on gradient descent for handling regression tasks. In this version, we consider the following algorithms: gradient descent [8, 13], mini-batch gradient descent (MBGD) [14], stochastic gradient descent (SGD) [9], stochastic average gradient descent (SAGD) [15], momentum gradient descent (MGD) [16], accelerated gradient descent (AGD) [17], Adadelta [10], Adagrad [11], RMSprop [18], and Adam [19]. The software library has been developed in R ecosystem, which is an open-source programming language that provides more than 8000 packages. We choose to implement the algorithms in R because it has two repositories (i.e., Comprehensive R Archive Network (CRAN) at http://cran.r-project.org/) and Bioconductor Project at http://www.bioconductor.org/). So, users can download, install, and use the package easily.

The remainder of this paper is structured as follows. Section II briefly gives an introduction to gradient descent. Section III presents the development of the software library in R. In Section IV and Section V, we demonstrated experimental analysis that predict the gas compressibility factor (i.e., *Z*-factor). Results and discussion are presented in Section VI. Finally, Section VII concludes the research and its future work.

2. The Gradient Descent Method and Its Variants

In machine-learning concepts, gradient descent is included as supervised learning for dealing with regression tasks. It is also named as steepest descent, which is a method to find an optimal value of an objective function by minimizing cost function [8].

In regression tasks, data training, that are mostly arranged in a table where rows represents instances/samples while columns is involved parameters and output variable, should be provided. The learning step is conducted to construct a model. In this case, the model basically contains coefficients of each variable in the hypothesis function (i.e., linear equation). So, it can be seen that gradient descent is used to determine coefficients in the model by minimizing cost functions. After obtaining the model, prediction over data testing can be done by calculating the linear model.

As we mentioned previously that there are many variants of gradient descent that have been proposed. In this section, we provided several algorithms based on gradient descent. Firstly, we provided the pseudo code of gradient descent, as illustrated in Fig. 1. Obviously, updating coefficients (θ) is done to obtain a suitable model so that mapping between input parameters and output parameter in data training is correct. Additionally, according to the algorithm we have to calculate the hypothesis function of each data sample for every iteration. Because of that, the computational cost can be so high.

Input: Data training with *dim*(m, n + 1) containing input samples $\boldsymbol{X}: x^1, \ldots, x^m$ and output values $\boldsymbol{y}: y^1, \ldots, y^m$, maximum iteration *maxIter*, step size α.
Output: The best coefficients $\boldsymbol{\theta}$ for the hypothesis function h_θ
Algorithm:
Generate initial coefficients $\boldsymbol{\theta}$ ($\theta_0, \theta_1, \ldots, \theta_n$) randomly
While $(t < maxIter) \parallel ((\boldsymbol{\theta_{new}} - \boldsymbol{\theta_{old}}) < \varepsilon)$ **do**

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 \leftarrow \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) . x_1^{(i)}$$

…

$$\theta_n \leftarrow \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) . x_n^{(i)}$$

Update the coefficients $\boldsymbol{\theta_{new}}$: $\theta_0, \theta_1, \ldots, \theta_n$
End

Fig. 1. Pseudo code of gradient descent [13].

One modification proposed to improve the performance is stochastic average gradient descent [15]. Figure 2 shows that the computation cost is reduced by performing the stochastic process. Additionally, in this case we do not need to calculate all data samples. Other variants that we consider in this research are mini-

batch gradient descent (MBGD) [14], stochastic gradient descent (SGD) [9], momentum gradient descent (MGD) [16], accelerated gradient descent (AGD) [17], Adadelta [10], Adagrad [11], RMSprop [18], and Adam [19]. Detailed algorithms can be found in respective literatures.

Input: Data training with *dim*(*m*, *n* + 1) containing input samples $\boldsymbol{X}$: $x^1, \ldots, x^m$ and output values $\boldsymbol{y}$: $y^1, \ldots, y^m$, maximum iteration *maxIter*, step size α.
Output: The best coefficients $\boldsymbol{\theta}$ for the hypothesis function h_θ
Algorithm:
Generate initial coefficients $\boldsymbol{\theta}$ ($\theta_0, \theta_1, \ldots, \theta_n$) randomly
While ($t < maxIter$) || (($\boldsymbol{\theta_{new}} - \boldsymbol{\theta_{old}}$) < ε) **do**
Generate random number (*rd*): 0 or 1
Shuffle data training
If (*rd* == 1) **then**
Select randomly *r* numbers of data samples

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 \leftarrow \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}).\ x_1^{(i)}$$

…

$$\theta_n \leftarrow \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}).x_n^{(i)}$$

Update the coefficients $\boldsymbol{\theta_{new}}$: $\theta_0, \theta_1, \ldots, \theta_n$
End
End

Fig. 2. Pseudo code of stochastic average gradient descent (SAGD) [15].

3. The Development of Software Library in R: "gradDescent"

As we mentioned previously that in the package "gradDescent" there are currently 10 methods based on gradient descent. Figure 3 illustrates classification of these methods.

As shown in Fig. 3, we can see that generally speaking, there are three different strategies of gradient descent modifications that are implemented. Firstly, four variants, which are batch gradient descent, MBGD, SGD, and SAG, are included in algorithms that have different ways on choosing number of data samples. Two techniques on optimization of learning speed are MGD and AGD. The last group is based on determining learning rate with adaptive mechanism. In this group, we consider 4 algorithms: Adagrad, Adadelta, RMSProp, and Adam.

Figure 4 explains data flow diagram (DFD) applied in the package "gradDescent". It can be seen that six modules have been implemented, as follows: feature scaling, splitting dataset, learning, prediction, reverse feature scaling, and error calculation. Feature scaling, well known as normalization, is used to change scales of datasets by using variance scaling and min-max scaling so that the new dataset is generated with range between [-1, 1] and [0, 1]. It is important to be executed when we have widely different scales between features/variables. In the fitting step, sometimes we need to divide the datasets into two parts: data training

(*DataTrain*) and data testing (*DataTest*). To accomplish this objective, we provided splitting dataset. The main function included in this package is to learn data training in order to obtain a model. In the learning step, we provide some functions related to their algorithms. After that, with the obtained model, we can predict data testing. In case we do normalization, de-normalization/ reverse feature scaling must be executed to obtain real numbers of predicted values. Error can be calculated by performing the last function in DFD.

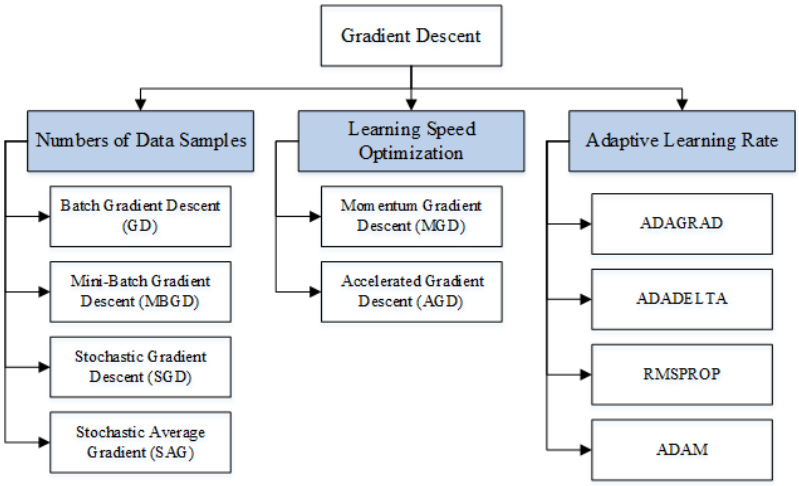

Fig. 3. Gradient descent and its variants implemented in "gradDescent".

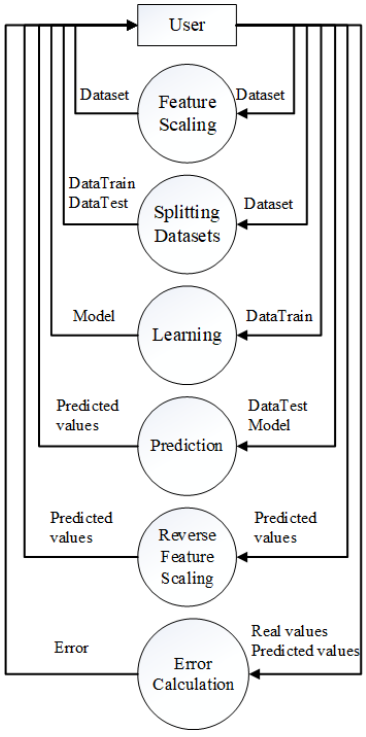

Fig. 4. Data flow diagram (DFD) in "*gradDescent*".

All considered algorithms have been implemented in R programming language. The published package, currently on version 2.0.1, can be downloaded from https://cran.r-project.org/package=gradDescent. Furthermore, the manual article to show how to use all functions is included as well. The following example is the function signature of the Adadelta algorithm:

```
ADADELTA(dataTrain, maxIter = 10, momentum = 0.9, seed = NULL)
```

It means that the Adadelta algorithm can be executed by calling `ADADELTA()` with the following parameters:

- dataTrain: a data.frame that represents training data ($m \times n$), where m is the number of instances and n is the number of variables where the last column is the output variable. dataTrain must have at least two columns and ten rows of data that contain only numbers (integer or float).
- maxIter: the maximal number of iterations. The default value is 10.
- momentum: a float value representing momentum give a constant speed to learning process. The default value is 0.9.
- seed: an integer value for static random. Default value is NULL, which means the function will not do static random.

The other methods can be seen at the website.

4. Experimental Study on Prediction of the Gas Compressibility Factor

This section is aimed to the use of the "*gradDescent*" package on prediction of the gas compressibility factor (i.e., *Z*-factor). This factor is required to be predicted because it represents the thermodynamic properties of the gas, relating to the phase change, the temperature, and the pressure of the gas [20]. Basically, it is defined as the ratio of the molar volume of a gas to the molar volume of an ideal gas at the same temperature and pressure [21]. Information regarding the problem statement of *Z*-factor can be found in detail in literature [12].

Previous research in literature [12] have shown the datasets that are used in the experiments. The datasets were experimentally collected by Kennedy in 1954 [22]. The data contain 2110 samples with three columns/ variables: temperature (T) in *Celsius*, pressure (*P*) in *bars*, and density in *gr/cc*. After doing conversion, we obtain the temperature (*T*) in Kelvin and pressure (*P*) in atm, and the compressibility of gas (*Z* factor). Then, we shuffled the data.

Experimentations are conducted on the four steps as follows:

1) Datasets are normalized by executing the function `varianceScaling()`.
2) We split the data into two parts by calling `splitData()`: data training and data testing. In this case, we defined 80% for data training and the rest for data testing.
3) In the learning step, models are generated by performing 10 algorithms. The following are functions executed: `GD()`, `MBGD()`, `SGD()`, `SAGD()`, `MGD()`, `AGD()`, `ADAGRAD()`, `ADADELTA()`, `RMSPROP()`, and `ADAM()`. For each function, we simulate the following different maximum

iterations: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 1000, and 10000. It can be seen that we performed 120 simulations in total. We need to do all simulations in order to obtain trend of convergences.

4) Prediction over data testing is perform by calling `prediction()`.
5) Since on the previous step we do normalization, we need to obtain real predicted values by conducting de-normalization with the function `varianceDescaling()`.
6) Finally, we calculate the root mean squared error (RMSE) by executing `RMSE()`. Additionally, some comparisons with other methods are conducted as well.

5. Results and Discussion

After running 120 simulations that have been explained in the previous section, we obtained RMSE of each simulation. Then, these errors are plotted in order to obtain their trend. Comparisons among algorithms can be also presented as illustrated in Fig. 5. In horizontal axis, we have 12 values of maximum iterations (i.e., 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 1000, and 10000), while the vertical axis represents RMSE values. It can be seen that each algorithm has different paths or trends. For example, Adagrad, AGD, and Adadelta have relatively stable along with all maximum iteration, but the others are very fluctuated.

In details, the best average of RMSE of all algorithms can be seen in Table 1. It can be seen that AGD has outperformed the rest of the algorithms with 0.123 s for the computation cost. Moreover, the fastest method is RMSPROP that took only 0.044 s.

Predicted values of all data testing can be seen in Fig. 6. Even though gradient descent and its variants can follow the Z-Factor trend, the extreme values are difficult to be predicted correctly. Thus, it seems that the other algorithms based on non-linear models or soft computing should be considered to be used, such as fuzzy rule-based system [6, 7], and rough sets [18].

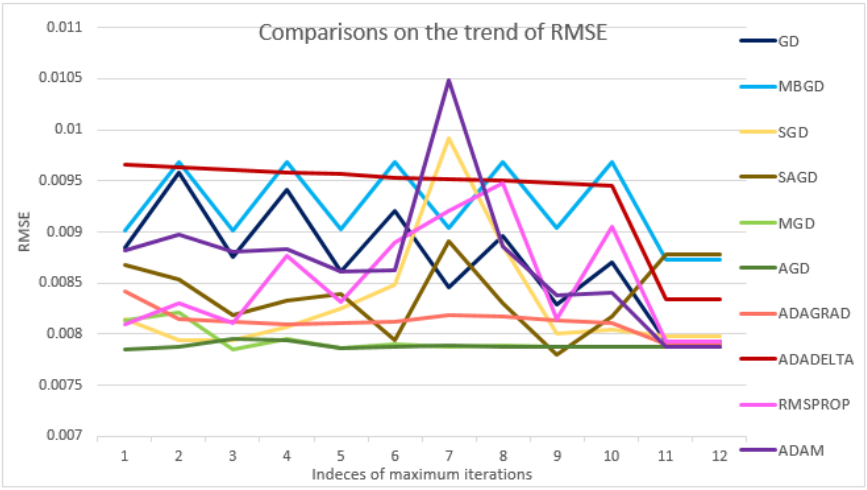

Fig. 5. Comparisons on RMSE trend of all simulations.

Table 1. Best average of RMSE of each algorithm.

Methods	Average of RMSE	Computation Cost (second)
GD	0.008723882	0.115833333
MBGD	0.009253105	0.0783333
SGD	0.0083202638	0.0325
SAGD	0.008401404	0.24166667
MGD	0.007930742	0.115
AGD	0.007887178	0.123333
ADAGRAD	0.008116677	0.565
ADADELTA	0.00935268	0.059166667
RMSPROP	0.008518413	0.044166667
ADAM	0.008713801	0.051666667

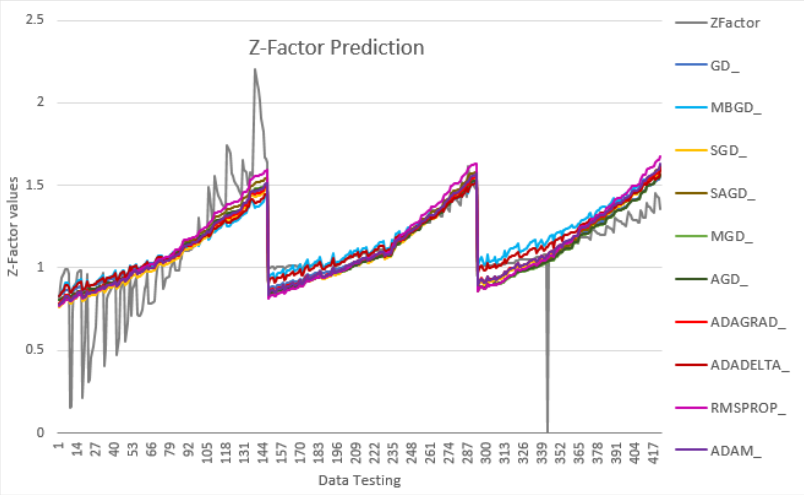

Fig. 6. Predicted values of each algorithm.

6. Conclusion

In this research, we have developed the *R* package, namely "gradDescent". It implements a linear model based on gradient descent for dealing with regression tasks. In total, eleven algorithms have been embedded as follows: Mini-Batch Gradient Descent, Stochastic Gradient Descent, Stochastic Average Gradient Descent, Momentum Gradient Descent, Accelerated Gradient Descent, Adagrad, Adadelta, RMSprop and Adam. Other features are included in the package as well, such as normalization, de-normalization, and error calculation. Furthermore, to validate the implementations, we perform experimental study on prediction of *Z*-Factor, which is a task required to know the thermodynamic properties of the gas. The results show that the algorithms included in the package provide reasonable predicted values of *Z*-Factor. So, the package "*gradDescent*" can be used as an alternative software library for dealing with various regression tasks in the realistic world problems.

Acknowledgements

A.B.D.N acknowledged RISTEK DIKTI for grant-in-aid in Penelitian Terapan Unggulan Perguruan Tinggi Negeri (PTUPT) and Penelitian Unggulan Strategi Nasional (PUSN).

References

1. Klein, S.; Pluim, J.P.W.; Staring, M.; and Viergever, M.A. (2009). Adaptive stochastic gradient descent optimisation for image registration. *International Journal of Computer Vision*, 81(3), 227.
2. Kudryavtsev, A.V.; Dembélé, S.; and Piat, N. (2017). Autofocus on moving object in scanning electron microscope. *Ultramicroscopy*, 182, 216-225.
3. Sober, B.; and Levin, D. (2017). Computer aided restoration of handwritten character strokes. *Computer-aided Design*, 89, 12-24.
4. Zhang, S.; Choromanska, A.E.; and LeCun, Y. (2015). Deep learning with elastic averaging SGD. *Proceedings of the Advances in Neural Information Processing Systems*. Montreal, Canada, 685-693.
5. Villa, A.; Fauvel, M.; Chanussot, J.; Gamba, P.; and Benediktsson, J.A. (2008). Gradient optimization for multiple kernel's parameters in support vector machines classification. *Proceedings of the Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*.Boston, US, 1-2.
6. Riza, L.S.; Bergmeir, C.; Herrera, F.; and Benitez, J.M. (2014, July). Learning from data using the R package" FRBS". *Proceedings of the Fuzzy Systems (FUZZ-IEEE)*, 2149-2155.
7. Riza, L.S.; Bergmeir, C.; Herrera, F.; and Benitez, J.M. (2014). FRBS: Fuzzy Rule-Based Systems for classification and regression in R. *Journal of Statistical Software*, 65(6), 1-30.
8. Yuan, Y.X. (1999). Step-sizes for the gradient method. *AMS IP Studies in Advanced Mathematics*, 42(2), 785.
9. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*. Paris, France, 177-186.
10. Zeiler, M.D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701. Retrieved on December 2017, from https://arxiv.org/abs/1212.5701.
11. Duchi, J.; Hazan, E.; and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121-2159.
12. Riza, L.S.; Nasrulloh, I.F.; Junaeti, E.; Zain, R.; and Nandiyanto, A.B.D. (2016). GradDescentR: An R package implementing gradient descent and its variants for regression tasks. *Proceedings of the Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Bandung, Indonesia, 125-129.
13. Cauchy, A.-L. (1847). Methode generale pour la resolution des systemes d'equations. *Comptes rendus de l'Académie des Sciences de Paris*, 25, 536-538.
14. Cotter, A.; Shamir, O.; Srebro, N.; and Sridharan, K. (2011). Better mini-batch algorithms via accelerated gradient methods. *Proceedings of the Advances in neural information processing systems*. Granada, Spain, 1647-1655.

15. Schmidt, M.; Le Roux, N.; and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2), 83-112.
16. Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1), 145-151.
17. Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2). *Doklady AN USSR*, 269(3), 543-547.
18. Riza, L.S.; Janusz, A.; Bergmeir, C.; Cornelis, C.; Herrera, F.; Slezak, D.; and Benítez, J.M. (2014). Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets". *Information Sciences*, 287, 68-89.
19. Kingma, D.; and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. Retrieved on December 2017, from https://arxiv.org/abs/1412.6980.
20. Rowlinson, J.S.; and Watson, I.D. (1969). The prediction of the thermodynamic properties of fluids and fluid mixtures-I The principle of corresponding states and its extensions. *Chemical Engineering Science*, 24(10), 1565-1574.
21. De Monte, F. (2002). Calculation of thermodynamic properties of R407C and R410A by the Martin–Hou equation of state—part I: Theoretical development. *International Journal of Refrigeration*, 25(3), 306-313.
22. Kennedy, G.C. (1954). Pressure-volume-temperature relations in CO2 at elevated temperatures and pressures. *American Journal of Science*, 252(4), 225-241.