gradDescentR: An R Package Implementing Gradient Descent and Its Variants for Regression Tasks

Lala Septem Riza, Imam Fachmi Nasrulloh, and Enjun Junaeti
Department of Computer Science Education
Universitas Pendidikan Indonesia
Indonesia
lala.s.riza@upi.edu

Rena Zain and Asep Bayu Dani Nandiyanto
Department of Chemistry Education
Universitas Pendidikan Indonesia
Indonesia

***Abstract*—Gradient Descent (GD) is one of famous machine-learning methods used for prediction on regression tasks in many fields. However, only a few software library utilizing it can be found in the literature. Therefore, this research is aimed to implement the method and the following variants: Mini-Batch Gradient Descent (MBGD), Stochastic Gradient Descent (SGD), and Stochastic Average Gradient Descent (SAG). The package is written in the programming language *R*, which is a software environment offering many facilities for statistics and machine-learning tools. Moreover, we provide the use of these approaches in a case study of gas industries, which is to predict values of the compressibility factor (Z-factor) of CO_2. Basically, we consider the problem as a regression task including pressure and temperature as the input attributes. The first step is generating a learning model from available training data by using the methods. Then, the model is used to predict the Z-factor over new data. A performance comparison between the package and other methods from the literature is presented as well.**

Keywords – gradient descent; R package; compressibility factor; machine learning; experimental study

I. Introduction

Nowadays, machine learning, which is one field in Computer Science focusing on algorithms that are able to learn from data [1], has been discussed and used for dealing with many problems intensively. GD [2] as a basic method in machine learning is used for optimization of local minimal and prediction on regression problems. For example, in the research [3, 4] GD is embedded to optimize fuzzy rule bases.

Because of the popularity, researchers have been developing many algorithms based on GD, so that the performance can be improved. We can find some examples of GD's modifications, such as SGD [5] and SAG [6]. Implementations on parallel computing are also done by e.g., [7].

In this study we attempt to accomplish two objectives. First, it is aimed to implement GD and its variants for solving regression tasks in the programming language *R* [8]. We choose *R* since it has been showing as an open-source environment that provides complete software libraries for many areas, such as machine learning, statistics, graphics, high performance computing, etc. Moreover, according to a survey conducted by KDnuggets [9], *R* was the most popular programming language to be used for an analytics/data mining/data science in 2015. The second objective is to predict Z-factor of CO_2 by utilizing the proposed package. It is an important task in industry, e.g., chemical, oil, and gas engineering. For example, it is to determine CO_2 compression, design of pipeline, material balance calculations and surface facilities design [10].

The remainder of this paper is structured as follows. Section II briefly gives an introduction to the programming language *R* and its ecosystem. Section III presents the first objective, which is the implementation of GD and its variants in *R*. In Secton IV and Section V, we illustrate some experimentations of Z-factor calculations and their analysis. Finally, Section VI concludes the research and its future work.

II. *R* and its Ecosystem

R is an open-source programming language and analysis environment containing more than 8000 packages for statistics, bio-informatics, visualization, machine learning, economics, etc [8]. Moreover, *R* can be run in many operating system such as Linux, Mac OS X, Solaris, and MS Windows. As of this writing, the stable version of *R* is 3.2.5, which can be downloaded at http://www.r-project.org.

R is constantly improving and offering a lot of benefits for scientists, engineers, and practitioners. The following are some advantages to use *R*: (i) it is open source and available for multiple platforms; (ii) it provides comprehensive data structures for data analysis, such as *matrix*, *data.frame*, *list*, etc; (iii) it allows to write code in procedural, functional, and object-oriented programming ways; (iv) it can be embedded by other programming languages, such as *C++* and *FORTRAN*; (v) it provides a lot of primitive functions, especially for statistics and graphics; (vi) there are some solid communities to build useful packages and to help *R* users (e.g., the Rhelp mailing list (http://stat.ethz.ch/mailman/listinfo/r-help), stackoverflow (http://stackoverflow.com/questions/tagged/r), etc). Moreover, some books have been published to introduce R, such as [11, 12].

To make easy to use, to keep in good quality, and to maintain continuously, mostly R packages are saved in the following repositories: the Comprehensive R Archive Network (CRAN, http://cran.r-project.org/) and Bioconductor Project (http://www.bioconductor.org/). The first one contains more than 6000 packages included in 30 task views (e.g., Cluster, Environmetrics, High Performance Computing, Machine Learning, Natural Languages Processing, etc.) whereas Bioconductor focuses on R packages related to the bio-informatics field. For example, in CRAN we can find following packages grouped in Machine Learning: *frbs* [13, 14] and *RoughSets* [15].

III. Implementation: Gradient Descent and Its Variants

GD is a famous algorithm to find a local minimum of an objective function by searching along the steepest descent direction. Therefore, as long as the current iterate point is not a stationary point, the method certainly moves to a lower value of the objective function [16].

In machine learning, it is mostly used for dealing with supervised learning, which is regression tasks. By using GD, we construct a model represented in a linear equation that maps the relationship between input variables and the output one. In other words, GD determines suitable coefficients of each variable so that the equation can express the mapping correctly. Algorithm 1 illustrates steps to obtain the coefficients of linear equations.

```
input  : Data training with $dim(m, n+1)$
         containing input samples
         $\boldsymbol{X} : \boldsymbol{x}^1, \ldots, \boldsymbol{x}^m$ and output values
         $\boldsymbol{y} : y^1, \ldots, y^m$,
         maximum iteration $maxIter$,
         step size $\alpha$.
output : The best coefficients $\boldsymbol{\theta}$ for the
         hypothesis function $h_\theta$
Generate initial coefficients $\boldsymbol{\theta}$ randomly;
while $(t < maxIter) || ((\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}) < \epsilon)$ do
    $\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)$
    $\theta_1 \leftarrow \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x^1$
    ...
    $\theta_n \leftarrow \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x^n$
    Update the coefficients $\boldsymbol{\theta}_{new} : \theta_0, \theta_1, \ldots, \theta_n$
end
```

Algorithm 1: Pseudo code of GD [2].

From the computing perspective, obviously there is a main drawback in Algorithm 1, which is high computation cost corresponding to numbers of datasets. It is happened because we need to calculate the hypothesis function of each data sample for every iteration. Therefore, researchers have been introducing some variants of the method. For example, the study [17] proposes the MBGD algorithm. SGD [5] and SAG [6] are also used to speed up the algorithms. These algorithms have been implemented to deal with many problems, such as image processing [18, 19] and breast cancer classification [20].

In this research, we develop an R package implementing four algorithms based on GD: original GD, MBGD, SGD, and SAG. The package is called *gradDescentR*. These algorithms are written into a main R function, with the following signature:

```
trainData <- function(dataset,
+  nData = nrow(dataset), theta,
+  alpha = 0.1, maxIter = 1,
+  maxError = 0, isNormalize = TRUE,
+  typeMethod = "GD", seed = NULL)
```

It can be seen that to run the function, we need to supply several parameters. Some parameters have a default value while the others must be defined. The most important argument is *dataset* representing data training in *data.frame*. *theta* is used to initiate the coefficients while *alpha*, *maxIter*, *maxError*, *isNormalize*, *typeMethod*, and *seed* express the step size, maximum iteration, tolerated error, normalization of the dataset, the chosen method, and a seed value of random generation, respectively. Especially for *typeMethod*, we can select one of the following values: *"GD"*, *"MBGD"*, *"SGD"*, and *"SAG"*.

After obtaining a linear model, we need to perform the *prediction()* function. This function has the following signature:

```
prediction <- function(model, dataTest)
```

So, there are two arguments in the function: *model* and *dataTest*, representing an R object of the linear model and new data for testing, respectively.

IV. Experimental Study: The compressibility factor of CO_2

In this section, we illustrate the use of the package for calculating Z-factor of CO_2. However, firstly we explain the problem formulation more detail in the next section. Data gathering and experimental design are discussed as well.

A. Problem Definition

One of the problems in calculation in industry is analyzing Z-factor, which 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]. This factor is important for prediction the thermodynamic properties of the gas, relating to the phase change, the temperature, and the pressure of the gas [22]. In short, Z-factor is defined in terms of the gas constant R and the measurable gas variables (i.e., pressure P, volume V, absolute temperature T, and quantity). The quantity is expressed as the number of moles n and density ρ. The simple expression of Z factor can be written in the following relationship [23]:

$$Z = \frac{P}{\rho RT}$$

Moreover, Z-factor of CO_2 is an essential calculations in chemical, oil, and gas engineerings. For example, calculating Z factor is used to determine CO_2 compression, design of pipeline, material balance calculations and surface facilities design [10]. It is also important to do in enhanced oil recovery combined with carbon sequestration in mature oil fields [24, 25].

Mostly, there are four strategies to determine Z factor of CO_2. First, it can be done by experimental/laboratory measurements. Even though we can obtain the precise values, these procedures are usually expensive, time consuming, and cumbersome experiments [26]. Another method to predict the Z factor is by calculating equations of state (EOS) [27]. A main drawback of this approach is that we need to construct and solve a model containing complicated and implicit equations. The third way is much easier and faster, which is via correlations. Since correlations are built by multiple steps sequentially, the error will be propagated in the other calculations [28]. Because of these disadvantages of each approach, machine-learning approaches as the last strategy are proposed. The following are studies conducting machine learning for calculating Z factor: [29–32].

B. Data Gathering and Experimental Design

In order to do some experimentations, we are using some data collected experimentally by George C. Kennedy [33]. The data contain 2110 samples involving temperature (T) in *Celsius*, pressure (P) in *bars*, and density in *gr/cc*. After doing conversion, we obtain a relationship between input attributes (i.e., T in *Kelvin* and P in *atm*) and the output (i.e., Z factor). To get a good distribution of T, P, and Z, we need to shuffle the data.

After gathering and pre-processing the data, we make the following experimental design. We firstly split the data into two groups: training and testing. The training data, used for learning or constructing a linear model, contain 80% of data while the rest is for testing. Corresponding to four implemented methods based on GD, there are four scenarios of the experiments: using GD, MBGD, SGD, and SAG. For every scenario, we perform several maximum iterations (*maxIter*) and number of datasets on each batch (*nData*) with two different values of *alpha* as illustrated in Table I. The last step is to calculate Root Mean Squared Error (RMSE). Furthermore, we compare the results with other methods: Subtractive Clustering (*SBC*) and Dynamic Evolving Neural-Fuzzy Inference System (*DENFIS*) included in the *R* package *frbs* [13].

TABLE I
SCENARIOS AND THE INPUT PARAMETERS.

Scenario	Method	*alpha*	*nData*	*maxIter*
1	GD	0.1, 0.01	-	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 1000
2	MBGD	0.1, 0.01	10, 20	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
3	SGD	0.1, 0.01	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
4	SAG	0.1, 0.01	10, 20	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

TABLE II
RESULT COMPARISON WITH OTHER METHODS.

Method	Parameters	Best RMSE
GD	$maxIter = 1000$, $alpha = 0.1$	0.151
MBGD	$maxIter = 3$, $nData = 10$, $alpha = 0.01$	0.152
SGD	$maxIter = 2$, $alpha = 0.1$	0.153
SAG	$maxIter = 4$, $nData = 10$, $alpha = 0.1$	0.165
SBC	$r.a = 0.5$, $eps.high = 0.5$, $eps.low = 0.15$	0.229
DENFIS	$Dthr = 0.01$, $max.iter = 300$, $step.size = 0.01$, $d = 2$	0.563

V. RESULTS AND DISCUSSION

After running some simulations with several values of the parameters indicated in Table I, we obtain the best predicted Z factor of CO_2 of all methods as illustrated in Figure 1. It can be seen that even though at beginning all methods cannot predict precisely, they give small errors after the index 144. It might be happened because Z-factor values on the index from 1 to 144 are fluctuating and sensitive. In other words, these values should be approximated by a method using a set of hypothesis functions expressed by non-linear equations instead of a linear one as GD.

In more detail, we can see RMSE of all methods based on GD and its comparison with *SBC* and *DENFIS* in Table II. It can be seen that GD gives the smallest RMSE, which is 0.151 on the iteration 1000. However, regarding the number of iterations, SGD, MBGD, and SAG can obtain the best RMSE on second, third, and fourth iterations. It should be noted that even though on the second iteration SGD has the best RMSE, it does not mean on the next iterations, the method consistently has the same or better values of RMSE. It can be understood because they are included in a stochastic algorithm. Moreover, two other methods as the comparison, which are SBC and DENFIS, are difficult to obtain better results because the Z-factor datasets probably contain small intervals and sensitive values.

VI. CONCLUSION AND FUTURE WORK

There are two main contributions in this research. The first one is to create an *R* package for regression tasks, called *gradDescentR*. It implements four algorithms based on GD and its variants (i.e., MBGD,

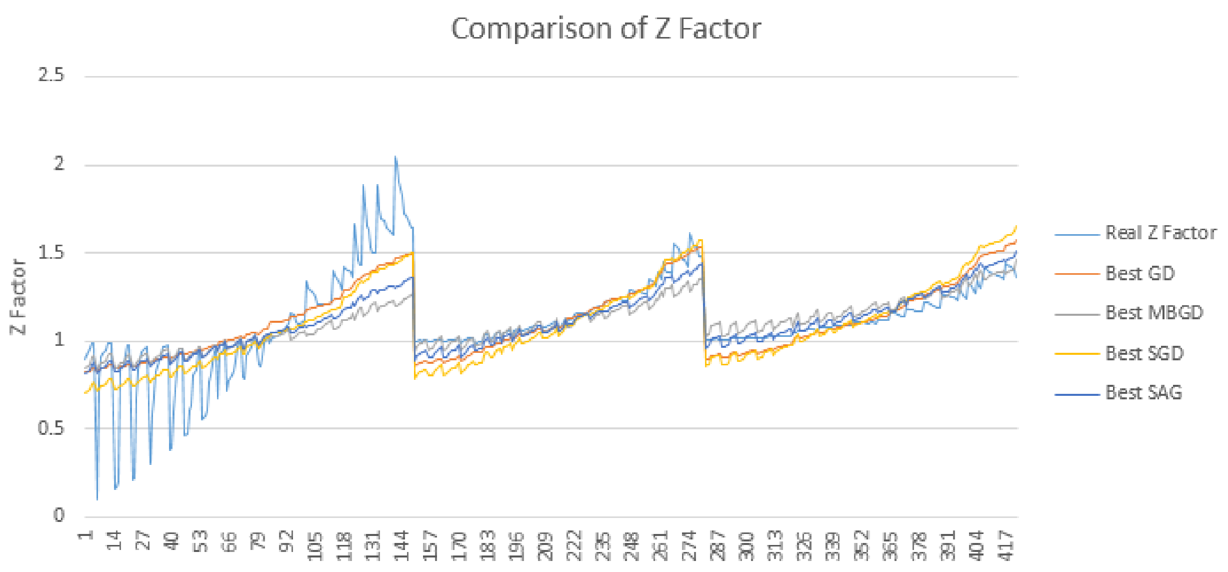

Figure 1. Comparison between real and predicted values of Z factor.

SGD, and SAG). Second, we perform some experiments to calculate Z-factor of CO_2 by using the package. Moreover, some comparisons are also presented.

As future work, we plan to implement other variants of GD, such as Accelerated Gradient Method (AGM), Incremental Aggregated Gradient (IAG), and GD with momentum, for the same task. Furthermore, we will be studying the implementations of the methods in parallel computing.

References

[1] R. Kohavi and F. Provost, "Glossary of terms," *Machine Learning*, vol. 30, no. 2-3, pp. 271–274, 1998.

[2] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[3] J.-S. R. Jang, "Anfis: adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, no. 3, pp. 665–685, 1993.

[4] F. Guély and P. Siarry, "Gradient descent method for optimizing various fuzzy rule bases," in *Fuzzy Systems, 1993., Second IEEE International Conference on.* IEEE, 1993, pp. 1241–1246.

[5] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010.* Springer, 2010, pp. 177–186.

[6] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Advances in Neural Information Processing Systems*, 2012, pp. 2663–2671.

[7] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[8] R. Ihaka and R. Gentleman, "R: A language for data analysis and graphics," *Journal of computational and graphical statistics*, vol. 5, no. 3, pp. 299–314, 1996.

[9] KDnuggets, "Languages for analytics/data mining," Tech. Rep., 2015, http://www.kdnuggets.com/2015/05/poll-r-rapidminer-python-big-data-spark.html.

[10] A. M. Elsharkawy, "Efficient methods for calculations of compressibility, density and viscosity of natural gases," *Fluid Phase Equilibria*, vol. 218, no. 1, pp. 1–13, 2004.

[11] R. D. C. Team, *An Introduction to R.* Vienna, Austria: R Foundation for Statistical Computing, 2008. [Online]. Available: http://www.R-project.org/

[12] J. Adler, *R in a nutshell: A desktop quick reference.* " O'Reilly Media, Inc.", 2010.

[13] L. S. Riza, C. Bergmeir, F. Herrera, and J. M. Benítez, "frbs: Fuzzy rule-based systems for classification and regression in R," *Journal of Statistical Software*, vol. 65, no. 1, pp. 1–30, 2015.

[14] L. S. Riza, C. Bergmeir, F. Herrera, and J. M. Benítez, "Learning from data using the R package frbs," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on.* IEEE, 2014, pp. 2149–2155.

[15] L. S. Riza, A. Janusz, C. Bergmeir, C. Cornelis, F. Herrera, D. Ślezak, J. M. Benítez *et al.*, "Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets"," *Information Sciences*, vol. 287, pp. 68–89, 2014.

[16] Y.-x. Yuan, "Step-sizes for the gradient method," *AMS IP Studies in Advanced Mathematics*,

vol. 42, no. 2, p. 785, 2008.

[17] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," in *Advances in neural information processing systems*, 2011, pp. 1647–1655.

[18] Z. Xie, H. Ma, B. Qi, G. Ren, Y. Tan, B. He, H. Zeng, and C. Jiang, "Restoration of degraded images using pupil-size diversity technology with stochastic parallel gradient descent algorithm," *IEEE Photonics Journal*, vol. 8, no. 2, pp. 1–10, 2016.

[19] Y. Qiao, B. Van Lew, B. Lelieveldt, and M. Staring, "Fast automatic step size estimation for gradient descent optimization of image registration," *IEEE Transactions on Medical Imaging*, vol. 35, no. 2, pp. 391–403, 2016.

[20] D. Mittal, D. Gaurav, and S. Roy, "An effective hybridized classifier for breast cancer diagnosis," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2015-August, 2015, pp. 1026–1031.

[21] F. De Monte, "Calculation of thermodynamic properties of R407C and R410A by the Martin–Hou equation of state—part i: theoretical development," *International journal of refrigeration*, vol. 25, no. 3, pp. 306–313, 2002.

[22] J. Rowlinson and I. Watson, "The prediction of the thermodynamic properties of fluids and fluid mixtures-i the principle of corresponding states and its extensions," *Chemical Engineering Science*, vol. 24, no. 10, pp. 1565–1574, 1969.

[23] M. Waxman and J. R. Hastings, "A burnett apparatus for the accurate determination of gas compressibility factors and second virial coefficients, and an evaluation of its capability based on some results for argon and carbon dioxide," *J. Res. Natl. Bur. Stand. Sect. C*, vol. 75, pp. 165–176, 1971.

[24] K. Srinivasan, "Identification of optimum interstage pressure for two-stage transcritical carbon dioxide refrigeration cycles," *The Journal of Supercritical Fluids*, vol. 58, no. 1, pp. 26–30, 2011.

[25] A. Hemmati-Sarapardeh, S. Ayatollahi, A. Zolghadr, M.-H. Ghazanfari, and M. Masihi, "Experimental determination of equilibrium interfacial tension for nitrogen-crude oil during the gas injection process: the role of temperature, pressure, and composition," *Journal of Chemical and Engineering Data*, vol. 59, no. 11, pp. 3461–3469, 2014.

[26] T. Ahmed *et al.*, *Reservoir engineering handbook.* Gulf Professional Publishing, 2006.

[27] T. H. Ahmed, *Hydrocarbon phase behavior.* Gulf Pub Co, 1989, vol. 7.

[28] M. Mahmoud, "Development of a new correlation of gas compressibility factor (Z-factor) for high pressure gas reservoirs," *Journal of Energy Resources Technology*, vol. 136, no. 1, p. 012903, 2014.

[29] E. Mohagheghian, A. Bahadori, and L. A. James, "Carbon dioxide compressibility factor determination using a robust intelligent method," *The Journal of Supercritical Fluids*, vol. 101, pp. 140–149, 2015.

[30] A. Kamari, A. Hemmati-Sarapardeh, S.-M. Mirabbasi, M. Nikookar, and A. H. Mohammadi, "Prediction of sour gas compressibility factor using an intelligent approach," *Fuel Processing Technology*, vol. 116, pp. 209–216, 2013.

[31] A. Shokrollahi, M. Arabloo, F. Gharagheizi, and A. H. Mohammadi, "Intelligent model for prediction of co_2–reservoir oil minimum miscibility pressure," *Fuel*, vol. 112, pp. 375–384, 2013.

[32] A. Fayazi, M. Arabloo, and A. H. Mohammadi, "Efficient estimation of natural gas compressibility factor using a rigorous method," *Journal of Natural Gas Science and Engineering*, vol. 16, pp. 8–17, 2014.

[33] G. C. Kennedy, "Pressure-volume-temperature relations in co_2 at elevated temperatures and pressures," *Am. J. Sci*, vol. 252, no. 4, pp. 225–241, 1954.