Analysis of the Application of Stochastic Gradient Descent to Detect Network Violations

Nataliya Boyko, Iryna Khomyshyn, Natalia Ortynska, Viktoria Terletska, Oksana Bilyk and Oleksandra Hasko

Lviv Polytechnic National University, Profesorska Street 1, Lviv, 79013, Ukraine

Abstract

Over the past decade, data sizes have grown faster than processor speeds. In this context, the capabilities of statistical machine learning methods are limited by computational time, not sample size. A more accurate analysis reveals qualitatively different trade-offs in the case of small and large learning problems. This work considers the occurrence of a large-scale case involving the basic algorithm's computational complexity and the implementation of optimization in non-trivial ways. Optimization algorithms such as stochastic gradient descent, which demonstrate exceptional performance for large-scale problems, are considered. The work compares the algorithms of gradient descent, stochastic gradient, and standard deterministic gradient descent. Here are given ways to apply them in machine learning and considered a formal description of the operation of gradient descent, stochastic gradient, and standard deterministic gradient descent algorithms. Here is analyzed the difference in the process of algorithms. The advantages and disadvantages of each of them are given. A comparison of both approaches is considered: deterministic and Stochastic algorithms. This work has analyzed the effectiveness of these two approaches. Convolutional (convolutional) neural networks are considered, which, unlike conventional deep neural networks, use convolutions to find patterns in images, thereby improving prediction in image recognition problems. A model of a convolutional neural network is presented. Here are analyzed the parameters of training the input sample and its testing. As a result of using a simulation of the usual gradient descent, when the weight is updated only after viewing all the elements, the value of the loss function is considered. The analysis of the selected problem and model also justifies the results of its use.

Keywords [1]

Machine Learning, Artificial Intelligence, stochastic gradient descent, stochastic algorithms, negative log likelihood, convolution neural network.

1. Introduction

As a sub-branch of artificial intelligence, machine learning is actively developing in various areas of technology and human life. It is used in several computational problems in which the development and programming of explicit algorithms with good performance are difficult or impossible. Examples of its applications include email filtering, detecting network criminals or malicious insiders, Optical Character Recognition, ranking training, computer vision, etc.

For so-called" training", machine learning often uses statistical techniques and algorithms, one of which is the gradient descent method.

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland
EMAIL: nataliya.i.boyko@lpnu.ua (N. Boyko); Khomyshyni@gmail.com (I. Khomyshyn); natalius2008@ukr.net (N. Ortynska); Viktoriia.O.Terletska@lpnu.ua (V. Terletska); Lubik.anelia@gmail.com (O. Bilyk); Oleksandra.I.Hasko@lpnu.ua (O. Hasko)
ORCID: 0000-0002-6962-9363 (N. Boyko); 0000-0002-6180-3478 (I. Khomyshyn); 0000-0002-5061-5340(N. Ortynska); 0000-0002-9334-2557 (V. Terletska); 0000-0001-6042-1147 (O. Bilyk); 0000-0003-4519-610X (O. Hasko)

However, when training the model, a massive sample of data is most often used, as a result of which the standard deterministic gradient descent does not show the best time results. A modification of deterministic gradient descent is stochastic gradient descent, which provides accelerated learning without significant loss of efficiency.

This work aims to analyze and reveal the operation of stochastic gradient descent, explore the application of this algorithm in machine learning training, and compare it with a deterministic algorithm.

2. Review of literature sources

Almost all machine learning problems are optimization problems, which is to find the extremum of the objective function [1]. Finding a model and the correct objective function is the first step in solving the machine learning problem. Then you need to define analytical and numerical methods that will help solve the optimization problem [2].

The paper discusses first-order methods, namely methods that use the gradient of a function. In the author's work [3], the general optimization method is the usual gradient descent. The author of [5] describes a formal description and description of the algorithm for solving the gradient descent method. In his work [5], the parameters are updated iteratively in the direction of the gradients of the objective function. The mechanism of the gradient descent method is described in the paper [2]. The gradient descent method is easy to implement. The solution is a global minimum or maximum when the objective function is convex [6]. It often converges more slowly if the variable is closer to the optimal solution, which requires more thorough iterations [1,4,6]. Due to the high computational complexity in each iteration, stochastic gradient descent (SGD) [3, 19] is proposed for a large amount of data. Therefore, the authors [8, 10] present an analysis of the application of this algorithm in the process of machine learning training and its comparison with a deterministic algorithm.

The manually adjusted learning rate strongly affects the SGD effect. This is a complex problem of setting the appropriate learning rate value. It is described in the authors' works [1-8]. The study uses popular machine learning libraries for the Python language (such as PyTorch, TensorFlow, and SkLearn). The standard algorithm is not even implemented, but only its stochastic version exists [12-15].

Using a simulation of the usual gradient descent described in [1-4, 16], the model's accuracy becomes very low when the weight is updated only after viewing all the elements. It is described that with this approach, gradients are updated more accurately. In this paper, the authors conduct a study with many epochs, where gradient descent shows the best accuracy of training.

3. Materials and methods

Gradient descent is one of the most popular optimization algorithms and the most common way to optimize neural networks. At the same time, each modern deep learning library contains implementations of various algorithms for optimizing gradient descent. However, these algorithms are often used as black-box optimizers because it is difficult to find practical explanations for their strengths and weaknesses.

The study aims to provide an analysis of the behaviour of various algorithms for optimizing Gradient Descent, which will help to apply them. First, you should consider different options for gradient descent. The next step is to summarize the problems during training. Therefore, this work presents the most common optimization algorithms that demonstrate their motivation to solve these problems and lead to the conclusion of update rules.

Gradient descent is a way to minimize an objective function $J(\theta)$ that is parameterized through model parameters $\theta \in \mathrm{R}^{\mathrm{d}}$ by updating parameters in the opposite direction of the objective function $\nabla_{\theta} J(\theta)$ gradient to parameters. The learning rate η determines the size of the steps that need to be taken to reach the (local) minimum. When analyzing, you need to monitor the direction of inclination of the surface down, which is created by the objective function.

3.1. Gradient Descent

Gradient descent is an iterative first-order optimization algorithm in which steps are taken to find the local minimum of a function proportional to the opposite value of the process's gradient (or approximate gradient) at the current point.

In short, since the gradient shows us the direction and speed of movement of the function relative to all arguments, to find the local minimum, we must move opposite to the action of the process (if it increases, we move it to decrease, if it declines, we move even more to the minimum).

For an intuitive understanding of gradient descent, we can imagine the movement of the ball (arguments) along with a parabolic bucket (function) in the direction of the bottom (minimum). Still, our algorithm will perform its role (Figure 1). When the ball is on the left side (i.e., the function decreases), the algorithm must "push" the ball forward to reach the bottom. When it is, on the right side (the function increases), as shown in Figure 1, then the algorithm, on the contrary, "pulls" it down. This movement continues step by step until the ball reaches the bottom.

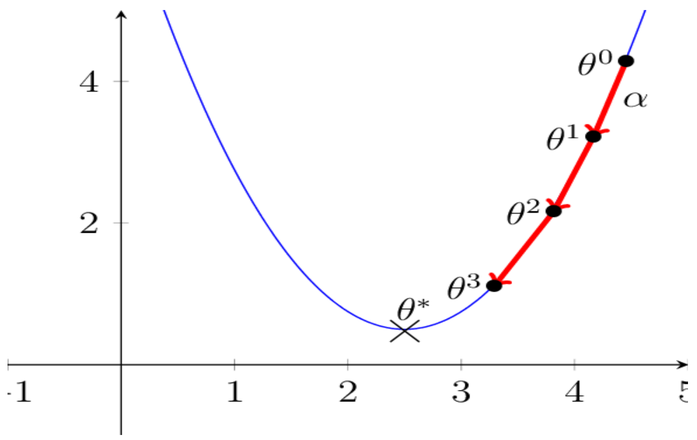

Figure 1: Intuitive image of the gradient descent algorithm

3.2. Standard deterministic gradient descent

Deterministic refers to an algorithm that returns the same result each time, regardless of the number of its runs.

Normal gradient descent can return the same values because it is performed on the total amount of data that comes to its input (that is, it works with the same input data and also serves the same steps on it without any accidents, which gives the same result at the end).

3.3. Stochastic gradient descent

Stochastic refers to an algorithm that returns different values each time, i.e. the algorithm contains a certain stochastic (random) relationship.

Different output values occur because stochastic gradient descent is not performed on the entire volume of input data but stochastically selects only a certain predefined number of records for each epoch. Such a set of individual records is called a batch.

3.4. Application in machine learning

Gradient descent is widely used in machine learning because it allows the model to learn from its mistakes and inaccuracies.

When training the model, a loss or error function is introduced, which provides information about the difference between the model's expected results. Since we cannot change the input data from the dataset, we must optimize the coefficients of our model.

The gradient of this function shows us how each coefficient affects the final error function. Applying a gradient descent to the loss function after this allows us to achieve the minimum of this function by adjusting the coefficients (i.e., make the error minimal). In this way, the configured model will enable us to use it for our tasks.

Thus, it becomes clear that gradient descent itself (stochastic and standard) is not used in machine learning. Its implementation is only an auxiliary tool for minimizing loss or error functions in machine learning algorithms.

4. Formal description of gradient descent algorithms

4.1. Mathematics in gradient descent

The gradient descent algorithm looks like this:

1. The calculation of new values of variables is presented in equation 1:

$$w = w - \gamma * grad(f(w)) = w - \frac{\gamma}{n}\sum_{i=1}^{n} grad(f_i(w)), \tag{1}$$

where $i > 0; \gamma > 0$ – a fairly small value, the pace of learning

2. Repeat the calculation of Step 1 until $grad(f(w_i)) > \varepsilon$, where $\varepsilon > 0$ – a fairly small constant.

4.2. Mathematics in Stochastic Gradient Descent

The stochastic gradient descent algorithm looks like this:

1. Randomly shuffle the dataset.
2. Select the k-required samples (in pure stochastic descent $k = 1$, in mini-batch $k > 1$).
3. The calculation of new values of variables is presented in equation 2

$$w = w - \gamma * grad(f(w)) = w - \frac{\gamma}{k}\sum_{i=1}^{k} grad(f_i(w)), \tag{2}$$

where $i > 0; \gamma > 0$ – a fairly small value, the pace of learning

4. Repeat Steps 1-3 until $grad(f(w_i)) > \varepsilon$, where $\varepsilon > 0$ – a fairly small constant.

4.3. Difference of algorithms

As can be seen from points 4.1 and 4.2, there is no difference in the formula for calculating new values for both algorithms. Differences appear in the approach to updating variables. While a standard gradient descent updates variables only after all n training samples have been passed, the stochastic algorithm updates these variables after passing only randomly selected models, which increases the number of updates per epoch.

4.4. Comparison of both approaches

On the one hand, a deterministic algorithm behaves more precisely since it processes a complete dataset to update data, but such training takes a very long time. On the other hand, the stochastic algorithm is not as accurate after each update, but its speed allows you to achieve the required accuracy in much less time.

Since machine learning works with massive data sets, the stochastic algorithm is much more efficient for its tasks. The standard algorithm is not even implemented in most popular machine learning libraries for the Python language (such as PyTorch, TensorFlow, and SkLearn), but only its stochastic version exists.

5. Recognition by convolutional neural networks trained using stochastic gradient descent

5.1. Analysis and justification of the selected task and model

Digit recognition is still a relevant topic today, which can be applied in various areas of our lives, starting from recognising bank card numbers in bank applications and ending with recognising mathematical expressions.

As you know, convolutional neural networks are best suited for image recognition, which, unlike conventional deep neural networks, use convolutions to find patterns in images that allow for improved prediction. Convolutions also compress images, which significantly increases the speed of analysis.

5.2. Description and analysis of the selected dataset

To recognize numbers, a well-known dataset is called MNIST. MNIST (short for Mixed National Institute of Standards and Technology) – a dataset of samples of handwritten writing of numbers. It is a standard proposed by the US National Institute of standards and technology to calibrate and compare image recognition methods using machine learning, primarily based on artificial neural networks. The dataset contains 60,000 images for training and 10,000 images for testing (Figure 2).

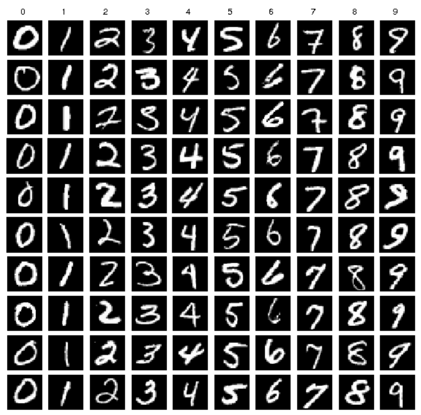

Figure 2: Example of the appearance of images from the MNIST dataset

5.3. Convolutional neural network model

To implement a convolutional neural network, I chose the following architecture, a diagram of which can be seen in Figure 3:

- input 28x28x1
- convolutional layer 24x24x10
- Max-puling 12x12x10
- ReLU activation function
- convolution 8x8x20
- Max puling 4x4x20
- complete connection of 320 neurons
- complete connection of 50 neurons
- output for 10 classes (10 digits)

Note: there is a ReLU activation function between all full connection layers $(\mathrm{Re}\,LU(x) = \max(0, x))$.

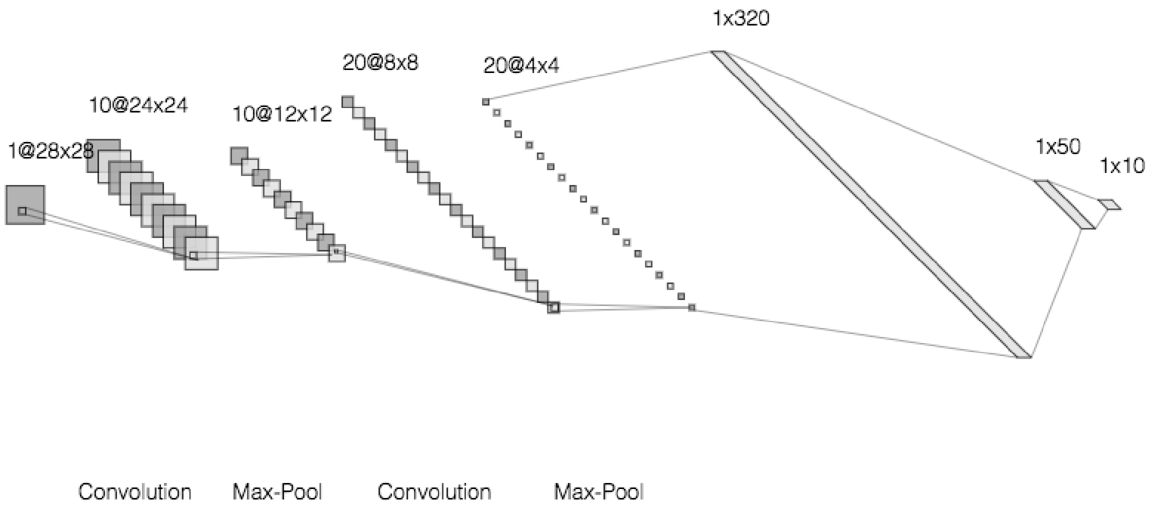

Figure 3: CNN diagram

5.4. Training and testing parameters

The data is randomly divided into 60,000 training images and 10,000 test images to prevent the possibility of retraining, when the model shows good results in training, but then cannot distinguish samples that were not seen during training.

As an optimizer for training, a stochastic gradient descent was chosen with the experimentally selected best learning rate *learningrate = 0.01*.

The training will last for 3 epochs with a different number of images in each betcha for the step of the stochastic gradient algorithm.

The negative logarithmic likelihood function $L(y) = -\log(y)$ is chosen as the loss function, which is simple, but at the same time powerful in classification problems.

Accuracy testing will reflect the ratio of the number of correctly predicted images to the total number of images in the dataset $\left(Accuracy = \frac{N_{true}}{N} * 100\% \right)$.

6. Work results

The graph shown in Figure 4 shows the change in the value of the loss function relative to the number of elements seen during a pure stochastic gradient descent (1 element per batch). At the same time, the accuracy of the neural network for 3 epochs was only 8%, which means that only about 800 out of 10,000 images were correctly identified by the network. The final error was about 4.2315.

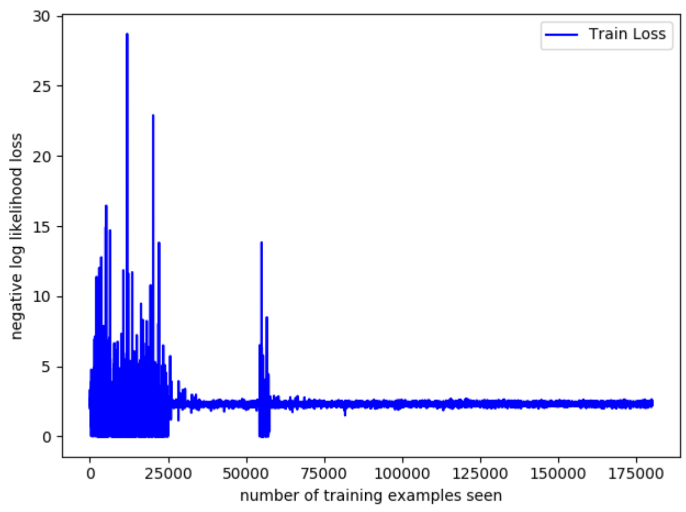

Figure 4: Graph of the ratio of the amount of losses relative to the number of elements passed by the network with a batch size of 1 element CNN diagram

The graph shown in Figure 5 shows the change in the value of the loss function relative to the number of elements seen with a batch size of 64 elements. At the same time, the accuracy of the neural network for 3 epochs was 97%. The final error was about 0.327.

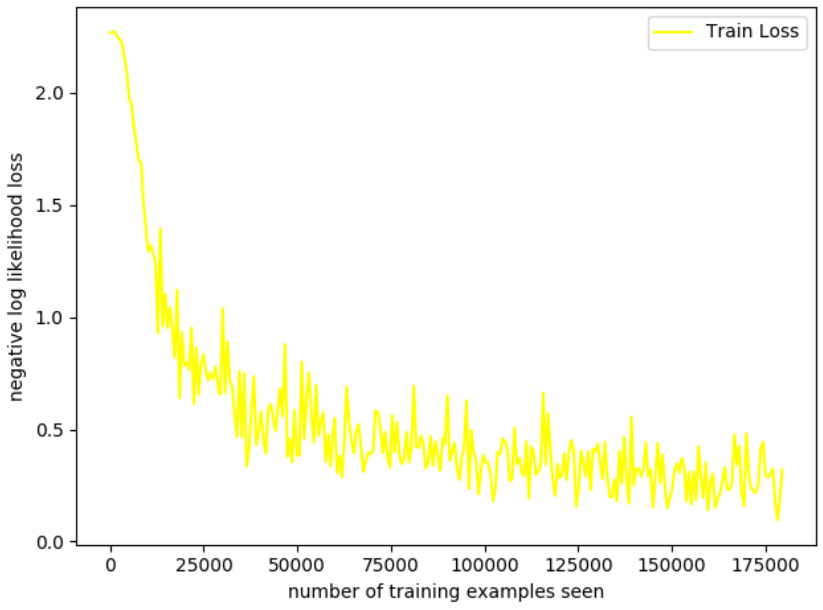

Figure 5: Graph of the ratio of the amount of losses relative to the number of elements passed by the network with a batch size of 64 elements

The graph shown in Figure 6 shows the change in the value of the loss function relative to the number of elements seen with a batch size of 128 elements. At the same time, the accuracy of the neural network for 3 epochs was 96%. The final error was about 0.421.

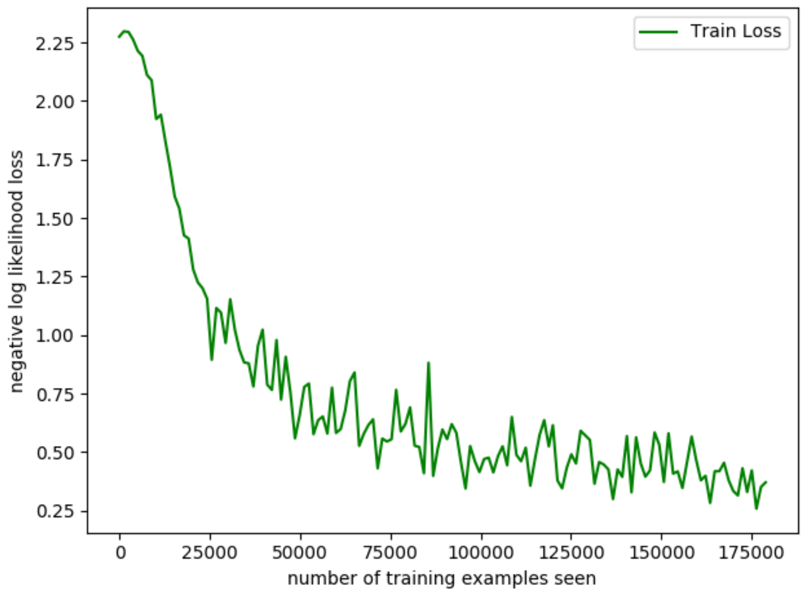

Figure 6: Graph of the ratio of the amount of losses relative to the number of elements passed by the network with a batch size of 128 elements

The graph shown in Figure 7 shows the change in the value of the loss function relative to the number of elements seen with a batch size of 256 elements. At the same time, the accuracy of the neural network for 3 epochs was 93%. The final error was about 0.686.

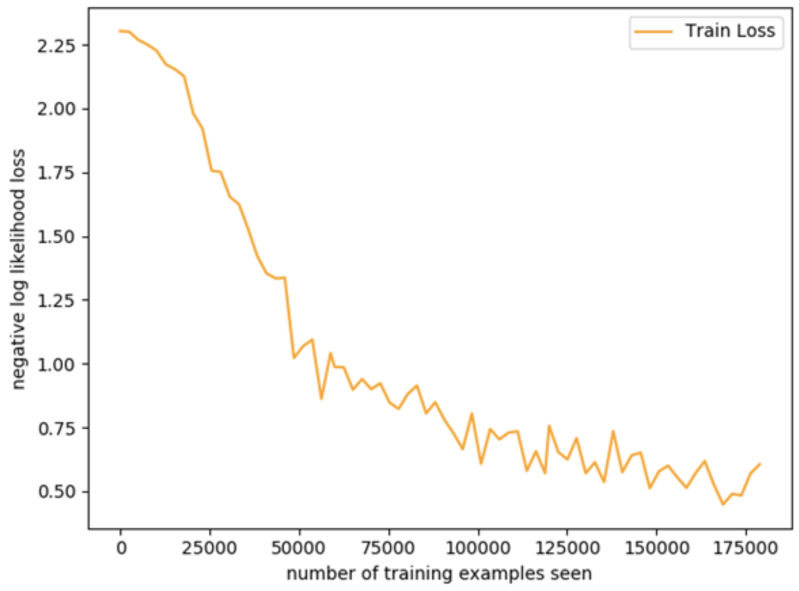

Figure 7: Graph of the ratio of the amount of losses relative to the number of elements passed by the network with a batch size of 256 elements

Since there is no ready-made implementation of normal gradient descent among known libraries, we can only stimulate it with stochastic gradient descent by setting the number of elements in the batch to 60,000 (i.e., to update the algorithm weights, we will need to view 60,000 images. The graph shown in Figure 8 shows the change in the value of the loss function relative to the number of elements seen during the simulated normal gradient descent. At the same time, the accuracy of the neural network for 3 epochs was only 9%. The final margin of error was about 2.284.

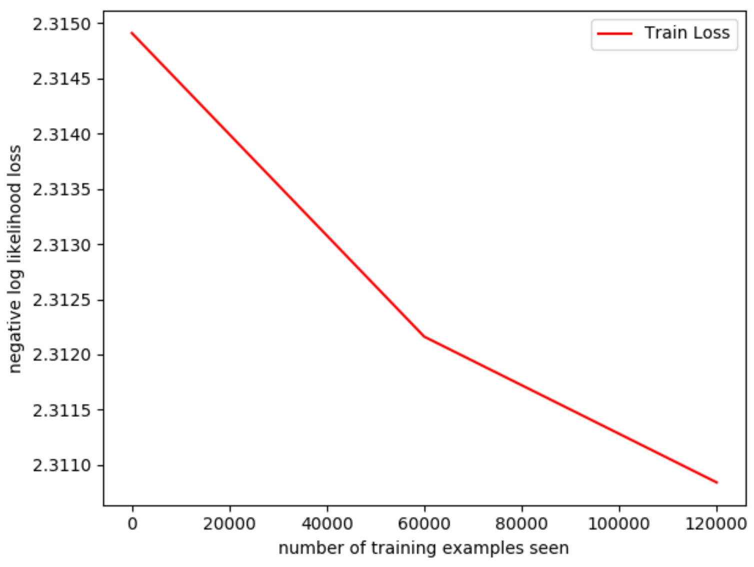

Figure 8: Graph of the ratio of the amount of losses relative to the number of elements passed by the network with a batch size of 60,000 elements (simulation of a normal gradient descent)

7. Discussion of results

When we consider only one random element to change the weights in pure stochastic gradient descent, the loss function has the highest values. One random component is not enough to describe the trend of movement of the loss function and the corresponding gradients for all other weights. It can be seen that the value of the loss function is constantly moving up and down. Obviously, with this movement of the process, the accuracy of neural network predictions was very low (only 8%).

When we use mini-batch stochastic gradient descent for a different number of elements in a batch (64/128/256), the value of the loss function increases and decreases alternately. However, in all cases, it still moves to a minimum. The only difference with a different number of batch elements is the frequency of these movements since the weights are updated with a different number of viewed features. At the same time, mini-batching showed the highest accuracy of predictions, which was 97% for a 64-element batch.

As a result of using the standard gradient descent simulation, when the weight update takes place only after viewing all the elements, we see that the value of the loss function also falls. Still, since the weight update is so rare, this movement is prolonged, so the model's accuracy also becomes very low. However, the gradients with this approach are updated more accurately. Presumably, with more epochs, gradient descent would show better training accuracy.

8. Conclusions

Looking at the results of stochastic gradient descent and comparing them with the simulated average gradient, we can see that SGD performs training much faster because only three epochs are enough to achieve 97% accuracy with the correct number of elements in the batch. At the same three epochs, gradient descent reduced the value of the loss function by several tenths.

At the same time, pure GHS also shows poor accuracy results since it cannot describe the change in weights using a single random element.

Considering the previous thesis, we can conclude that using a mini-party stochastic gradient is more effective and more in demand since it performs the fastest and best training of the model.

9. References

[1] J. Friedman, Another approach to polychotomous classification, Machine Learning, 2004.

[2] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. SpringerVerlag, New York, 2001.

[3] N. Boyko, A look trough methods of intellectual data analysis and their applying in informational systems, in: XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT), 2016, pp. 183-185.

[4] L. Breiman, Prediction Games and Arcing Algorithms. Neural Computation 11(7), 1999, pp. 1493-517. DOI:10.1162/089976699300016106

[5] P. Zhang, G. Zhang, Q. Pan, An ensemble learning method for classification of multiple-label data. Journal of Computational Information Systems 11(12), 2015, pp. 4539-4546. DOI: 10.12733/jcis14783.

[6] A.O. Pererodov, Characteristics of the quality of the regression model, 2020. URL:https://www.ereading.club/chapter.php/1002275/18/

[7] A. Bordes, L. Bottou, P. Gallinari, SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent, Journal of Machine Learning Research 10, 2009, pp. 1737-1754.

[8] L. Bottou, O. Bousquet, The Tradeoffs of Large Scale Learning, in: Advances in Neural Information Processing Systems, vol.20, 2008, pp. 161-168.

[9] L. Bottou, Y. Lecun, On-line Learning for Very Large Datasets. Applied Stochastic Models in Business and Industry, 21(2), 2004, pp. 137-151.

[10] N. Boyko, K. Kmetyk-Podubinska, and I. Andrusiak, Application of Ensemble Methods of Strengthening in Search of Legal Information, in: Lecture Notes on Data Engineering and Communications Technologies, Vol. 77, 2021, pp. 188-200. https://doi.org/10.1007/978-3-030-82014-5_13

[11] O. Bousquet, Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms. Thèse de doctorat, Ecole Polytechnique, Palaiseau, France, 2002.

[12] T. Joachims, Training Linear SVMs in Linear Time, in: Proceedings of the 12th ACM SIGKDD, 2006.

[13] M. Nazarkevych, S. Dmytruk, V. Hrytsyk, O. Vozna, A. Kuza, O. Shevchuk, V. Sheketa, Evaluation of the effectiveness of different image skeletonization methods in biometric security systems. International Journal of Sensors Wireless Communications and Control, 11(5), pp. 542-552.

[14] J. D. Lafferty, A. Mccallum, F. Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proceedings of ICML, 2001, pp. 282-289.

[15] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research 5, 2004, pp. 361-397.

[16] C. J. Lin, R. C. Weng, S. S. Keerthi, Trust region Newton methods for large-scale logistic regression, in: Proceedings of ICML, 2007, pp. 561-568.

[17] P. Massart, Some applications of concentration inequalities to Statistics. Annales de la Faculté des Sciences de Toulouse series 6,9,(2), 2000, pp. 245-303.

[18] S. Shalev-Shwartz, N. Srebro, SVM optimization: inverse dependence on training set size, in: Proceedings of the ICML, 2008, pp. 928-935.

[19] W. Xu, Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent. Journal of Machine Learning Research, 2010.