GRADIENT OMISSIVE DESCENT IS A MINIMIZATION ALGORITHM

Gustavo A. Lado and Enrique C. Segura

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

ABSTRACT

This article presents a promising new gradient-based backpropagation algorithm for multi-layer feedforward networks. The method requires no manual selection of global hyperparameters and is capable of dynamic local adaptations using only first-order information at a low computational cost. Its semi-stochastic nature makes it fit for mini-batch training and robust to different architecture choices and data distributions. Experimental evidence shows that the proposed algorithm improves training in terms of both convergence rate and speed as compared with other well known techniques.

KEYWORDS

Neural Networks, Backpropagation, Cost Function Minimization

1. INTRODUCTION

1.1. MOTIVATION

When minimizing a highly non-convex cost function by means of some gradient descent strategy, several problems may arise. If we consider a single direction in the search space, there could appear local minima [5], where it is necessary first to climb up in order to have the possibility to reach a better or global minimum, or plateaus [16], where partial derivatives have very small values even though being far from a minimum. Of course, in a multidimensional space these problems can combine to produce other ones, such as partial minima (ravines) [1], where for some direction (subspace) there exists a local minimum and for other direction there is a plateau, and the saddle points [3], where partial derivatives have small values since a partial minimum appears in a direction and a maximum appears in another direction. Even if considering a single search direction it can be easily seen that an algorithm using only gradient information might be in trouble when looking for a good minimum.

1.2. BACKGROUND

Approaches such as Stochastic Gradient Descent (SGD) [15] or Simulated Annealing (SA) [14, 7] introduce a random factor in the search which permits, to some extent, overcome the problem of local minima, whilst other methods as Gradient Descent with Momentum (GDM) [11] or adaptive parameters [2] allow for speeding up the search in the case of plateaus. When combined with incremental or mini-batch training, these techniques can be effective to some degree even with ravines and saddle points, but they use global hyperparameters that are generally problem-dependent.

DOI :10.5121/ijscai.2019.8103

Other techniques such as Conjugate Gradient Descent (CGD) [12], or others which incorporate second derivative information, as the Newton Method and the Levenberg-Marquardt algorithm [17], can speed up the search through a better analysis of the cost function, but are in general computationally expensive and highly complex, turning more difficult its implementation and use.

Finally, more recent algorithms such as Quick Back-Propagation (QBP) [17], Resilient Back-Propagation (RBP) [13] and Adaptive Momentum Estimation (AME) [19, 4] use adaptive parameters independent for each search direction, hence they do not require selection of global hyperparameters and can adapt its strategy as to cope with different kinds of problems. However, even in these situations, they can get stuck in bad solutions (poor minima). In addition, all these techniques require batch training, which in many cases may be not only inconvenient but impossible or highly impractical.

The main difficulty in the search of a good minimum (global or not) is that these algorithms try to take good decisions solely on the base of local information and values that, in many cases, are the result of estimations of estimations. In other words: it is difficult to take intelligent decisions with incomplete and inaccurate information.

Perhaps a better approach would be just to try that by the end of each epoch the probability of cost decrease be greater than that of increase and, in the case that it increased, it have a reasonably high probability of escaping to another basin of the cost surface. Thus, instead of trying to ensure convergence through analysis of the cost function, the matter is to warrant statistical tendency to a good minimum.

2. THE PROPOSED ALGORITHM

Our method borrows in part certain ideas from other techniques. For example, from SGD the introduction of a random element in the descent; from CGD the exploration of the cost function - though nonlinear in this case- and from RBP the use of the sign of the gradient as indicator of the search direction, omitting its magnitude [12,13, 14, 15, 17]. The main idea is that during training the search moves across the cost surface with random steps within a radius of search, so as to statistically converge to the minimum; and this radius decreases as the solution is approached.

2.1. DESCRIPTION

In each epoch the dataset is divided into mini-batches. A partial estimation of the cost function E is made for each one of those mini-batches, and the gradient is computed by standard error backpropagation. Then weights are modified in a random magnitude, bounded by a value R, but using the direction (sign) of the ΔW previously computed. At the end of each epoch, the cost is compared to that of earlier steps and, should it correspond, necessary adjustments are made on R.
If we look at each mini-batch as a sample of the dataset, we can consider each epoch as a Montecarlo estimation of the cost function. Similarly, the fact that the search radius gradually decreases suggests a resemblance to simulated annealing [10]. However, to impose a fixed reduction of R as a function of the number of epochs or of the cost is impractical due to the variability between different problems. For this reason a strategy was chosen where adjustments to R are dynamically applied, depending on the relative cost [9]. This turns it more robust, but the adjustments must be small, in the order of 1%, since by the very nature of the estimation of E, oscillations might occur.

Then the basic idea for modifying R can be expressed as follows: if the cost increased with regard to the last epoch, reduce the radius of search. This is relatively effective in a variety of scenarios, but these results can still be improved by using some local parameters instead of global ones [18]. For example, at the end of each epoch an estimation can be made of the individual contribution of each unit to the total cost, that is, the cost by unit (E_u). Similarly, instead of using a global radius of search R, we can have an individual radius (R_u) for each unit.

Under these conditions the rule for adjustment of search radii is modified as follows: if the total cost increased, reduce search radius for units with higher error and increase radius for the others (see Algorithm 1). Then, in this strategy the radius of search can be seen as the level of resolution of the cost function. If this function cannot decrease any more, the radius is reduced in order to look at the cost surface in more detail [2]. Increasing the radius for those units whose cost grew up helps to maintain an equilibrium and possibly to escape from a bad solution.

Algorithm 1. The Gradient Omissive Descent algorithm.

initialize W, R, E, t
while $(E > min\_error) \wedge (t < max\_epoch)$:
  $E^{[t]} \leftarrow 0$
  for every batch b:
    $W \leftarrow W + \mathrm{U}(0, R)^{|W|} \times sgn(\partial E(W, b) \partial W)$
    $E^{[t]} \leftarrow E^{[t]} + E(W, b)$
  $dE \leftarrow E^{[t]} - E^{[t-1]}$
  if $dE > 0$:
    for every unit u:
      if $dE_u > 0$:
        $R_u \leftarrow R_u \times 0.99$
      else:
        $R_u \leftarrow R_u \times 1.01$
  $t \leftarrow t + 1$

3. EXPERIMENTS

3.1. METHODS

In all experiments our technique, Gradient Omissive Descent (GOD), was compared with three other algorithms: SGD, GDM, RBP. These were chosen in order to cover the alternatives of incremental training, mini-batch and batch respectively. In future work we will also include other approaches such as CGD, though the results might not be easy to compare, and AME, for which we do not have yet a properly tested implementation.

The stop condition in all simulations was either the completion of a maximum number of epochs or having reached an error under a minimum value. This error was computed as the average of all the errors in the training dataset. Since data were artificially generated, it was possible to keep the same dataset size through all the tests.

As a measure of the efficacy of the algorithms, and since there exists no unique stop condition, the quantity *epoch/max_epoch* × *error/min_error* was used, where *epoch* and *error* correspond to final values, and *max_epoch* and *min_error* are the values for the stop condition. Thus, lower values indicate a higher efficacy and, in particular, quantities less than 1 imply that a solution has been obtained.
Experiments were realized on two kinds of problems, typical for evaluation of this class of algorithms: parity and auto-encoding. In both cases, additional constraints were introduced in order to have a better control and thus to adjust the level of difficulty [8]. Final results were computed as the average of several simulations under the same constraints but with different datasets.

3.1.1. PARITY

The parity problem, i.e. multivariate XOR, is defined as the result of verifying if the number of 1's in a list of binary variables is even. To generate data for these experiments we use variables $x_i \in \{-1, 1\}$ instead of binary, defining the function *even* as:

$$even(x_0 \cdots x_n) = \begin{cases} 1 & if\ (\sum_{i=0}^{n} u(x_i)) \bmod 2 = 1 \\ -1 & otherwise \end{cases}$$

Where $u(x) = 1$ if $x = 1$, and equals to zero otherwise.

In its simplest form, this problem consists of mapping n inputs into a single output which indicates the parity. What makes it particularly interesting is the fact that changing the value of just one input variable produces a change in the output. In our experiments we used 10 input variables $[x_0..x_9]$ and 10 output variables $[z_0..z_9]$, where $z_i = even(x_0..x_i)$. That is, each output corresponds to the parity on a portion of the input variables. This turns the problem more difficult, as the unique hidden layer must find a representation satisfying simultaneously all outputs.

3.1.2. AUTO-ENCODING WITH DEPENDENT VARIABLES

One of the critical factors in the efficacy of an auto-encoding is how independent are the data between each other. If the dataset is built up with completely independent variables and of the same variance, it is not possible to make an effective reduction of dimension [6].

For this reason, the datasets for the experiments were constructed with different proportions between independent and dependent variables. Independent variables were drawn from a random uniform distribution between -1 and 1, and the dependent ones were random linear combinations of the former. Final values result from applying the sign function in order to work also in this case with values in $\{-1, 1\}$.

However, in real situations, data are not often equally correlated between them. This fact was also taken into account, splitting the dataset into several classes. Each class not only uses different correlations for dependent variables, but also assigns different positions to the independent ones.

3.2. RESULTS

This section presents comparisons between the results obtained with the different algorithms under consideration for the problems described above. For each method, the parameters that

produced the better results were chosen. In all simulations the stop condition was 3000 epochs or an average error less than 10^{-3}. These settings had the purpose to ensure that all the algorithms had the possibility to converge.

The weights were independently initialized at random from an uniform distribution over the interval $\{-N^{-12}, N^{-12}\}$, being N the number of neurons in the previous layer. Initial values for R were computed as $1/N \times M$, where N is the number of units in the previous layer and M is the number of units in the same layer. Each mini-batch, both for GDM and for GOD, consisted of a 10% of the elements of the dataset, chosen at random without repetition.

3.2.1. PARITY

For this group of experiments the network had a single hidden layer of 40 units. The training set included all the 2^{10} possible combinations for the input. For SGD the learning rate η was 0.01, for GDM η = 0.001 and the momentum factor β = 0.5. For RBP, η^+ = 1.2 and η^- = 0.5 in all cases, as proposed in [13].

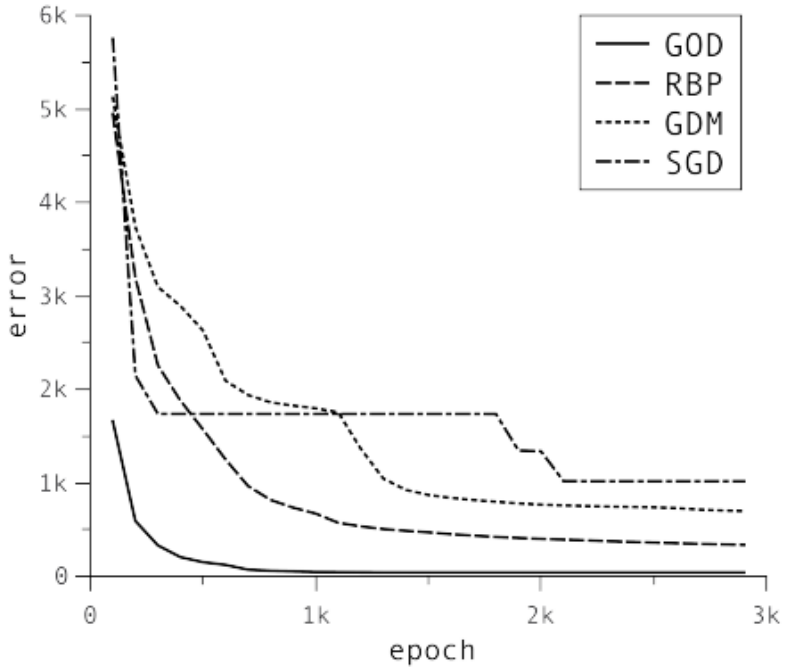

Figure 1. Training results of several algorithms for multiple parity.

Figure 1 shows the time evolution of the sum of square errors by epoch for different algorithms. These results correspond to a typical run of training, since the average of several simulations would not show the differences in the behaviour of different methods. Although the inner stochasticity of the training process introduces large variations in the way of descent for each algorithm, the order of convergence is markedly consistent. It can be noted that the cost decreases much more rapidly with the proposed method.

3.2.2. AUTO-ENCODING WITH DEPENDENT VARIABLES

A network with 100 input and 100 output neurons was used, while the size of the hidden layer was varied between 70, 80 and 90 units. The training set had 1000 instances, from which 25% were set apart for validation. In view of the number of simulations under the same conditions, the validation method used was holdout, since in these circumstances there is no need for more partitions of cross-validation. The parameters were: for SGD $\eta = 0.001$; for GDM $\eta = 0.001$, $\beta = 0.5$.

Figure 2 shows the variations in efficacy of different algorithms as changing the proportion of independent variables and the number of units in the hidden layer. The three parts of the figure correspond to the variation of number of units in the hidden layer, indicated on the left axis (a smaller value corresponds to a more difficult problem); percentages on the horizontal axis show the proportion of dependent variables (a smaller value also corresponds to a more difficult problem), and values on vertical axes indicate the efficacy of the algorithms computed as previously defined and averaged over several runs. In these simulations, 5 classes were included; results for other number of classes are not displayed here since they show a not quite pronounced variation, but consistent with the problem complexity.

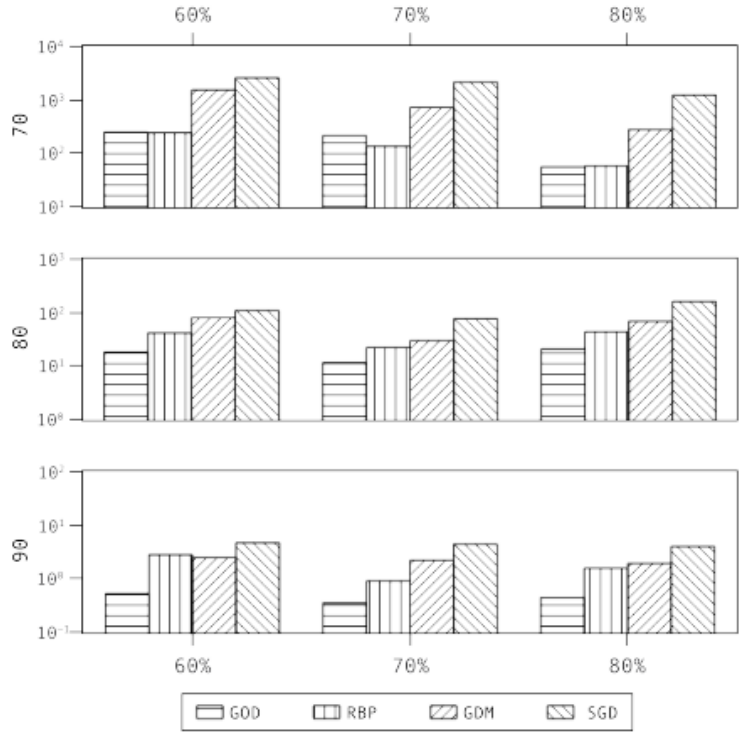

Figure 2. Comparison of the performance of different algorithms in problems of auto-encoding with correlated data.

It is worth noting not only how the efficacy of the proposed technique varies with problems of different complexity, but also how varies the relative behaviour of all compared methods. Depending on the complexity of the problem, the quality of our solution relatively changes with regard to other algorithms, but it can also be seen a clear trend to a faster convergence.

4. CONCLUSIONS

In the present work an optimization technique was introduced with the ability to find minima or quasi-minima of cost functions for non-convex, highly dimensional problems. The proposed algorithm manages successfully some classical difficulties reported in the literature, such as local minima, plateaus, ravines and saddle points. In our view, the method is an advance regarding the previous ones, from which it also takes inspiration, in the form of an exploration of the cost function, in a semi-random descent, and omitting the magnitude of the gradient.

The proposed algorithm works efficiently on a wide range of problems, featuring convergence rate and speed higher than other common methods. We also note that in certain cases, for example when the problem is very simple, less sophisticated algorithms as SGD can be more effective, but in its turn have the disadvantage of requiring a parameter setting. Similarly, in problems which can be intractable -due for example to architecture limitations- our solution is comparable to RPB but, unlike it, our method can work in mini-batches.

ACKNOWLEDGEMENTS

The idea for this algorithm was fundamentally conceived while L. G. was working with Professor Verónica Dahl in Vancouver, BC. Special thanks to her for the invitation and to the Emerging Leaders in the Americas Program (ELAP) of the Canada's government for making that visit possible.

REFERENCES

[1] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, Yann LeCun. The loss surfaces of multilayer networks. *Artificial Intelligence and Statistics*:192-204, 2015.

[2] Christian Darken, Joseph Chang, John Moody. Learning rate schedules for faster stochastic gradient search. *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*:3-12, 1992.

[3] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*:2933-2941, 2014.

[4] Timothy Dozat. Incorporating nesterov momentum into adam. , 2016.

[5] Marco Gori, Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76-86, 1992.

[6] Geoffrey E. Hinton, Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504-507, 2006.

[7] Scott Kirkpatrick, C. Daniel Gelatt, Mario P. Vecchi, others. Optimization by simulated annealing. *science*, 220(4598):671-680, 1983.

[8] Steve Lawrence, C. Lee Giles, Ah Chung Tsoi. What size neural network gives optimal generalization?: Convergence properties of backpropagation. 1998.

[9] George D. Magoulas, Michael N. Vrahatis, George S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769-1796, 1999.

[10] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087-1092, 1953.

[11] Miguel Moreira, Emile Fiesler. Neural networks with adaptive learning rate and momentum terms. Idiap, 1995.

[12] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525-533, 1993.

[13] Martin Riedmiller, Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *Neural Networks, 1993., IEEE International Conference on*:586-591, 1993.

[14] Herbert Robbins, Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*:400-407, 1951.

[15] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[16] Richard S. Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. *Proc. 8th annual conf. cognitive science society*:823-831, 1986.

[17] Bogdan M. Wilamowski. Neural network architectures and learning. *Industrial Technology, 2003 IEEE International Conference on*, 1:-1, 2003.

[18] Z. Zainuddin, N. Mahat, Y. Abu Hassan. Improving the Convergence of the Backpropagation Algorithm Using Local Adaptive Techniques. *International Conference on Computational Intelligence*:173-176, 2004.

[19] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

AUTHORS

Gustavo Lado is a graduated student from the University of Buenos Aires in Computing Science. His previous research interests include the simulation of the human visual system with artificial neural networks. Part of this work was recently published in the book "La Percepción del Hombre y sus Máquinas". He is currently working in neural networks applied to the natural language processing and cognitive semantics as part his Ph.D.

Enrique Carlos Segura was born in Buenos Aires, Argentina. He received the M.Sc. degrees in Mathematics and Computer Science in 1988 and the Ph.D. degree in Mathematics in 1999, all from University of Buenos Aires.He was Fellow at the CONICET (National Research Council, Argentina) and at the CONEA (Atomic Energy Agency, Argentina). He had also a Thalmann Fellowship to work at the Universitat Politécnica de Catalunya (Spain) and a Research Fellowship at South Bank University (London). From 2003 to 2005 he was Director of the Department of Computer Science of the University of Buenos Aires, where he is at present a Professor and Resarcher. His main areas of research are Artificial neural Networks - theory and applications- and, in general, Cognitive Models of Learning and Memory.