Mini-project - CS-439 Optimization for machine learning
Saddle-Free Newton Method for Neural Networks

Amos Gerber, Alessandro Fornaroli, Sergei Kliavinek
EPFL - Lausanne, Switzerland

***Abstract*—Saddle points may often slow down optimization algorithms, as many standard methods (such as gradient-descent methods) are attracted by such points. This issue is particularly important in the case of artificial neural networks, as the high dimensionality of the parameter space leads to a much higher proportion of saddle points in the optimization landscape. The Saddle Free Newton Method is a proposed second-order optimization algorithm that aims at solving this problem.**

In this paper, we investigate the application of a variant of this algorithm, specifically the Low Rank Saddle Free Newton Method, to two classification problems involving artificial neural network. We compare the results with baselines based on first-order methods to understand whether there are benefits in using this method.

I. Introduction

Minimizing a non-convex function is often challenging, especially in high dimensions. In particular, one of the complications is the existence of saddle points: in fact, the number of saddle points in an optimization landscape grows exponentially with the dimension [1]. This is problematic as standard optimization algorithms such as SGD may get stuck in these saddle point, as the gradient around this point tends to zero. are attracted by saddle points and take time to go out of the saddle points. This slows down the algorithm, which might ultimately not converge.

A possible solution to this problem that has been proposed by Dauphin et al., specifically the Saddle Free Newton method (SFN) [2]. This is a second order method inspired by the Newton method, but instead of tacking the Hessian, we take the absolute value of the Hessian. Empirical results seems to show that SFN can quickly escape from saddle points. Given that in high dimensions the Hessian is excessively expensive to compute, this algorithm uses an approximation of the Hessian. In particular, it uses the Lanczos vectors to approximate the Hessian.

However, for the purpose of this paper, we will investigate a close variant to the SFN method, i.e. the Low-Rank Saddle Free Newton Method, proposed by O'Leary-Roseberry [3]. This method uses a low rank approximation of the Hessian.

II. Theory

Suppose that the function to minimize is $f(x)$, which is sum structured, that means :

$$f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

One well known second order method to minimize this function is Newton method that, given x_0, computes

$$x_{k+1} = x_k H(x_k)^{-1}\nabla f(x_k)$$

where H^{-1} is the inverse of the Hessian ($\nabla^2 f$) and ∇f is the gradient of f.

A problem with this algorithm, as many standard other methods, is that it is attracted by saddle points, and therefore it may take a lot of time to escape such points. The SFN method aims at solving this issue by taking the absolute value of the Hessian $|H|$ instead of H, where $|H|$ is the same as H except that we flip all negative value to become positive. So we get, given x_0, compute: $x_{k+1} = x_k|H|^{-1}\nabla f(x)$

In practice, N is very big, so it's take to much time to compute $|H|^{-1}$. So we need, an approximation of $|H|^{-1}$, we will use the Low rank method to make the approximation of $|H|^{-1}$

Definition 1. *The spectral decomposition of H is given by* $H = U\Lambda U^T = \sum_{i=1}^{d} \lambda_i u_i u_i^T$*, such that* $|\lambda_i| \geq |\lambda_j|$ *if* $i > j$*. Then the* ***low rank decomposition*** *of degree* r *of* H *is define as :*

$$H_r = U_r \Lambda_r U_r^T = \sum_{i=1}^{r} \lambda_i u_i u_i^T.$$

For solving the problem of the rank deficient of H_r we use $\tilde{H}_r = H_r + \gamma I$, where γ is called the regularization or damping parameter.

Furthermore, we can easily see that the spectral decomposition of $|H|$ is $\sum_{i=1}^{d} |\lambda_i| u_i u_i^T$.

So the low rank approximation of $|H|$ is $|\tilde{H}|_r = |H|_r + \gamma I = U|\Lambda|U^T + \gamma I$

Finally we get that the low rank approximation for $|H|^{-1}$ is $\frac{1}{\gamma} I - \frac{1}{\gamma^2} U_r \left(|\Lambda|^{-1} + \frac{1}{\gamma} I_r\right)^{-1} U_r^T$.

With this theory we are able to create the low rank saddle free Newton (LRSFN) method. Algorithm 1 illustrates the pseudocode for the LRSFN method.

III. Experimental Setup

A. Experiments

For the purpose of this paper, we want to analyse the performance of the low-rank SFN algorithm when applied to artificial neural networks. In particular, we compare this algorithm to other baselines - that we will later describe - on two different tasks, both involving neural networks.

First, we carry a classification using the MNIST dataset for digit classification [4]. This is a classical classification problem

Algorithm 1: LRSFN

Given x_0;
while *not converge* **do**
  Compute the low rank approximation for the Hessian ;
  $U_r^{(k)} \Lambda_r U_r^{(k)T} \approx H_r$
  Compute A_k the low rank approximation of $|H_r|^{-1}$
  $A_k = \frac{1}{\gamma} I - \frac{1}{\gamma^2} U_r \left(|\Lambda|^{-1} + \frac{1}{\gamma} I_r\right)^{-1} U_r^T$
  Compute the gradient g_k
  $g_k = \nabla f(x_k)$
  Define a step size α_k and compute
  $x_{k+1} = x_k - \alpha_k A_k g_k$
end

in which it is necessary to determine from information about a black and white image what number is in this image.

The second experiment is based on the XOR problem, and it is based on an artificial dataset that we construct. Here we are solving an XOR problem that is widely known in science. It consists in the fact that it is impossible to train one single perceptron in such a way that it works as an XOR operator. In this case, if a layer with more than one perceptron is used, the landscape of the resulting loss function has a large number of saddle points, to combat which the presented algorithm was developed. We make two different formulations of this experiment, a simple one which only considers the pairs $\{(0,1),(1,0),(0,0),(1,1)\}$ and a continuous one, which considers two numbers with value from -1 to 1. For the latter, we randomly generated pairs of random number $(x_1, x_2) \in [-1,1]^2$, and assign to each pair a target value of 1 if $x_1 x_2 \geq 0$, or 0 otherwise. Figure 1 shows the dataset that we used, with values corresponding to 1 in blue and 0 in red. These two variations of the XOR problems are particularly hard for a neural network as the classification boundary is highly non-linear [5].

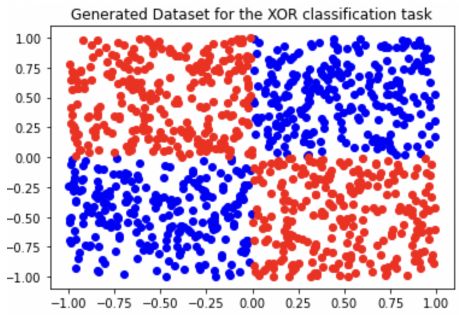

Fig. 1: Artificial dataset for the XOR experiment

B. Baselines

The following 3 algorithms were chosen for comparison: SGD, Adam [6], and RMSPror (proposed by Hinton) [7]. In particular, SGD was chosen because it is the most classical optimization method used in neural networks, being easy to understand and having many implementation; similarly Adam and RMSPror are popular and ubiquitous nowadays, being two of the gradient descent methods which generalize well giving good results, and having working implementation in multiple deep learning frameworks.

These algorithm clearly present major differences from the LRSFN algorithm, first and foremost because they are all first order methods, whereas the algorithm that we are investigating includes second order information. Nonetheless, they provide a good baseline because of their adaptability and popularity, as well as the ease of the implementation itself.

C. Implementation

The experiments were carried in the Keras framework for deep learning [8]. For the purpose of our experiments, we used the keras implementation of the low-rank Saddle Free Newton optimizer by O'Leary-Roseberry et al. [1] [3]. On the other hand, for the SGD, Adam and RMSPror algorithms, we made use of the standard Keras optimizers. In all of our experiments, we maintained the same neural network models and hyperparameters, changing only the type of optimizer used.

For the first experiment, on classification of digist with the MNIST dataset, we implemented a simple LeNet [9] type architecture consisting of 2 convolution layers and 2 fully connected layers. ReLU was chosen as the activation function, while Cross Entropy was chosen as the loss function, as it is excellent for the classification task. Training was performed over 25 epochs.

For the XOR classification problem, the same three optimization methods as in the first experiment were chosen for comparison. For this task, a neural network with an input full-connected layer of 2 neurons, a hidden full-connected layer and an output full-connected layer of 2 neurons was used. This architecture has been suggested by Mehta et al. [10]. Binary Cross Entropy was chosen as the loss function, which is ideal for the binary classification task. As a dataset, for the first varian of the XOR classification, 4 possible combinations of 0 and 1 were selected and fed into XOR. For the second one, 1000 datapoints in the space $[-1,1]^2$ were generated randomly, and assigned to two groups 0 and 1 as shown in figure 1. For the simpler method, the models were trained for 500 epochs, whereas for the second one they were trained for 25.

IV. Results

A. Digit Classification

For the experiment on digit classification on the MNIST dataset, the results are shown in the tab. I (after averaging over 5 calculations).

As we can see, all the classical algorithms did a great job, only SGD was slightly worse. LRSFN showed the weakest result compared to them. More importantly, the method did not converge from iteration to iteration. From this we can conclude that LRSFN is not suitable for solving standard problems.

[1] https://github.com/tomoleary/hessianlearn

TABLE I: Classification Accuracy for the XOR problem for different algorithms

Method	Accuracy (%)
SGD	98.8
Adam	99.9
RMSProp	99.9
LRSFN	35

B. Simple XOR Problem

The results were as follows. The only method that did not solve the problem in 500 epochs was SGD. It regularly got "stuck" at saddle points and at accuracy values of 50 or 75%. RMSPror and Adam solved the problem in ≈ 300 and ≈ 100 epochs, respectively. For the LRSFN method, only 5 epochs were enough. That is, indeed, this method does an excellent job when the loss function landscape is saturated with saddle points. However, it should be noted that the example we have presented is quite artificial, and in general, the proposed method has not gained much fame at the moment (see discussion).

C. Continuous XOR Problem

In this case, the data is heavily non-linearly separable, and this poses a great challenge to the neural network. With the neural network that we adopted, none of the optimization method managed to successfully separate the data, resulting in an accuracy score of approximately 50% (corresponding to random) in all cases. However, interestingly, as it can be seen from figure 2, the different optimization algorithm have lead to very different classification boundaries. In fact, the SGD, Adam and RMSProp have induced a linear boundary that approximates the diagonal of the square. On the other hand, the Saddle Free Newton Method generated a uniform classifier that always predicted the same value. Clearly, both these methods are highly unsuccessful in this case as they both lead to a very bad predictive power, indicating that the neural network itself was not good for this specific task. Nonetheless, this experiments highlights a difference between the outcomes of the two algorithms.

V. Discussion

Our experiments have shown that that the proposed method does not handle the standard MNIST dataset training task very well, but performs better in the case of the simple XOR problem. The experiment with the continuous XOR problem did not yield meaningful results, due to issues with the underlying classification model. Nonetheless, it is important to note that this paper is not focusing on the efficiency of the neural networks used to train the models, but rather to the optimization algorithm used to find the parameters of the model.

Moreover, it is also necessary to consider possible implementation issues that may hamper the final quality of the results: there might be some issues with the parameters chosen for the experiments, and possibly these two experiments, while in principle being meaningful and coherent, might not have been the best application of the SFN method. Further research and experiments would be needed to better assess the results of this method to different and more complex scenarios.

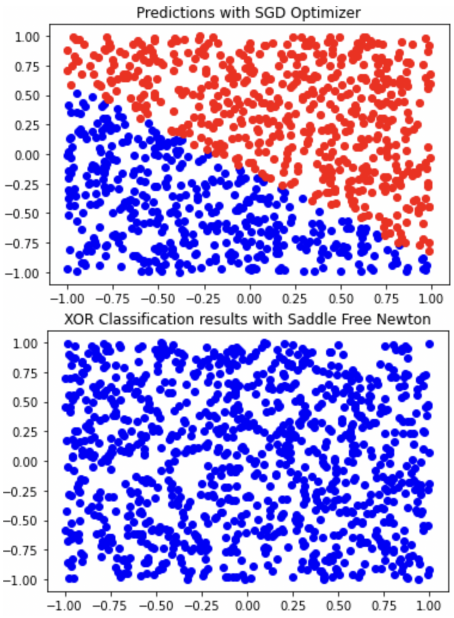

Fig. 2: Classification results on the test set, with SGD (above) and SFN (below). The Adam and RMSProp method yielded similar results to SGD.

Finally, it is also important to note the fact that the Saddle-Free Newton Method was slower than the other optimization algorithms used. This is due to the fact that each step of the LRSFN is more computationally expensive than the Adam and RMSProp optimizers, as even the low-rank approximation of the Hessian is also quite slow to compute. In fact, this is one of the major drawbacks associated to second-order methods for deep learning, that may make them less attractive than faster gradient-descent based methods [11].

VI. Conclusion

In conclusion, while having several theoretical advantages, the Saddle Free Newton Method also has drawbacks, and our experiments have not highlighted the presence of great benefits from the application of this algorithm. In fact, the first-order baselines have performed better than the SFN in the MNIST classification experiment, whereas in the XOR experiments the differences were minor.

While we did not experimentally find any strong evidence that would indicate to prefer the SFN method to first order methods, we have also observed its theoretical benefits, and we have seen that it does perform well in certain circumstances (such as the simple XOR problem).

References

[1] A. Auffinger and G. Ben Arous, "Complexity of random smooth functions on the high-dimensional sphere," *The Annals of Probability*, vol. 41, Nov 2013.

[2] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *arXiv preprint arXiv:1406.2572*, 2014.
[3] T. O'Leary-Roseberry, N. Alger, and O. Ghattas, "Low rank saddle free newton: Algorithm and analysis," *arXiv preprint arXiv:2002.02881*, 2020.
[4] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.
[5] T. Nitta, "Solving the xor problem and the detection of symmetry using a single complex-valued neuron," *Neural Networks*, vol. 16, no. 8, pp. 1101–1105, 2003.
[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[7] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.
[8] F. Chollet *et al.*, "Keras." https://keras.io, 2015.
[9] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, pp. 143–155, 1989.
[10] D. Mehta, X. Zhao, E. A. Bernal, and D. J. Wales, "Loss surface of xor artificial neural networks," *Physical Review E*, vol. 97, no. 5, p. 052307, 2018.
[11] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," 2017.