EE 510: Deep Learning Theory and Practice **Winter 2020**

Course Project: Project Topic
Due: March 15, 11:59 PM PT

Student Names: Matthew Bui

1 Paper Descriptions

1.1 Paper 3

Paper Title: Incorporating nesterov momentum into adam[1]
Student Name: Matthew Bui

1.1.1 Problem Description & Formulation

Gradient descent optimization is the most important topic in deep learning. It is the foundation for every model. In the deep learning recipe[2], gradient descent optimization always takes the first place. Other efforts such as collecting more data and improving the design of the network would be useless if the model does not get converged. Converge speed is also crucial for boosting the performance of a deep learning model, especially in large models, such as RNN and CNN. As a result, we need the models not only converged but also efficient.

There are many variants of stochastic gradient descent (SGD) such as classic momentum[3], Nesterov's accelerated gradient (NAG) [4] and Adam [5]. Each of them has pros and cons, so we will need to understand and decide when to use them. This paper will discuss the problems of each variant and propose a new optimization method. Algorithm 1 presents classic momentum with the notation used in this paper.

Algorithm 1: Classic momentum[1]

Result: new θ_t
Require: $\alpha_0 ... \alpha_T$: learning rate each timestep (presumably annealed);
Require: $\varphi_0 ... \varphi_T$: decay factor;
Require: $f(\theta)$: Stochastic objective function parameterized by θ and indexed by timestep i;
Require: θ_0: The initial parameters;
while θ_t *not converged* **do**
  $t \leftarrow t + 1$;
  $g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$;
  $m_t \leftarrow \varphi m_{t-1} + \alpha_t g_t$;
  $\theta_t \leftarrow \theta_{t-1} - m_t$;
end

Classic momentum memorizes previous update m_{t-1} and calculates the new update m_t based on that. This help the learning process more synchronized in the directions. However, there are some problems of classic momentum about the learning rate and the decay factor. In algorithm 1, the learning rate and the decay factor are fixed during the training and this makes the training unstable in the end of process. Toward to the end, we will need more adaptive learning rates and decay factors because the model needs less intensity update. These problems can be fixed in Adam. Algorithm 2 shows Adam Algorithm with the adaptive learning rate $\frac{\alpha}{\sqrt{\hat{n}_t + \epsilon}}$.

Algorithm 2: Adam[5]

Result: new θ_t
Require: α: learning rate;
Require: $\varphi_0 ... \varphi_T$: decay factor;
Require:$\nu_0 ... \nu_T$;
Require:$\epsilon \in$ Hyperparametter;
Require:$f(\theta)$: Stochastic objective function parameterized by θ;
Require:θ_0: The initial parameters;
$m_0, n_0, t \leftarrow 0$ (first/second moment vectors);
while θ_t *not converged* **do**
  $t \leftarrow t + 1$;
  $g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$;
  $m_t \leftarrow \varphi_t m_{t-1} + (1 - \varphi_t) g_t$;
  $n_t \leftarrow \nu_t n_{t-1} + (1 - \nu_t) g_t^2$;
  $\hat{m}_t = \frac{m_t}{1-\varphi_t}$;
  $\hat{n}_t = \frac{n_t}{1-\nu_t}$;
  $\theta_t \leftarrow \theta_{t-1} + \frac{\alpha \hat{m}_t}{\sqrt{\hat{n}_t + \epsilon}}$;
end

In Adam, the learning algorithm upgrades the classic momentum with the learning rate flexibility . In Algorithm 2, the learning rate are now more adaptive during timestep t. However, take a look the expanded θ_t updating using Adam to see its problem:

$$\theta_t \leftarrow \theta_{t-1} + \frac{\alpha \hat{m}_t}{\sqrt{\hat{n}_t + \epsilon}} \tag{1}$$

$$\theta_t \leftarrow \theta_{t-1} + \frac{\alpha \frac{m_t}{1-\varphi_t}}{\sqrt{\hat{n}_t + \epsilon}} \qquad \text{Sub } \hat{m}_t \tag{2}$$

$$\theta_t \leftarrow \theta_{t-1} + \alpha \frac{\varphi_t m_{t-1} + (1 - \varphi_t) g_t}{(1 - \varphi_t)(\sqrt{\hat{n}_t + \epsilon})} \qquad \text{Sub } m_t \tag{3}$$

In (3), m_{t-1} is independent from the current gradient and this also happens in classic momentum. Unfortunately, the independence of m_{t-1} creates a problem for the models. In figure 1, the gradient descent of classic momentum is less effective than Nesterov's accelerated gradient (NAG)[4] where m_{t-1} is not considered as a parameter. Adam is built on top of classical momentum, so it does not avoid this problem. As a result, we need a new optimizer fixing Adam as well as classic momentum gaps.

1.1.2 Application/Model/Algorithm/Theory Description

Adam is one of the latest effective gradient descent optimizer, so it has a lot of upgrades fixing problems of old ones. The obvious goal of deep learning and machine learning researchers is to go beyond the Adam and invent the state of the art optimizer. This is also the motivation of Dozat in his 2016 paper. Dozet has suggested to upgrade Adam optimizer based on the fact that basic gradient descent is provably accelerated using NAG[7]. Dozat's paper suggests a new way modifying Adam optimizer to fix Adam's independent m_{t-1} problems. Dozat idea is a significantly contribution because the Adam optimizer now is accelerated and we have a new powerful optimizer called Nadam[1].

Dozat's model is mixed Adam and NAG optimizers. This new model is like a racing car having extremely good breaks. NAG gives the model the of accelerating, in meanwhile, Adam gives it the abilities of adjusting the speed.

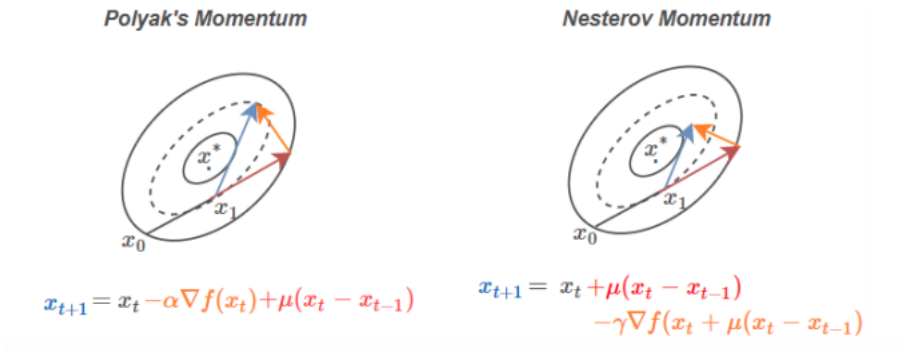

Figure 1: Independent m_{t-1} problem [6]

In figure 1, the gradient decent $\nabla f()$ of classic momentum does not depend on $m_{t-1} = \mu(x_t - x_{t-1})$, but the NAG does. NAG offers a provably better bound than gradient descent[7]. So, parameterizing m_{t-1} is , $\nabla f()$ gets close to the target x* faster. Let us take a look how NAG updates θ_t in a learning model:

$$g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1} - \varphi m_{t-1}) \tag{4}$$
$$m_t \leftarrow \varphi m_{t-1} + \alpha_t g_t \tag{5}$$
$$\theta_t \leftarrow \theta_{t-1} - m_t \tag{6}$$

It basically is same as classic momentum but we now parameterize m_{t-1}. We substitute m_t in (6) with (5)'s:

$$\theta_t \leftarrow \theta_{t-1} - (\varphi m_{t-1} + \alpha_t g_t) \tag{7}$$

Dozat suggests that in NAG, instead of parameterizing m_{t-1} to the ∇f, we can can simply apply the momentum step of timestep $t+1$ only once, during the update of the previous timestep t instead of $t+1$:

$$g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1}) \tag{8}$$
$$m_t \leftarrow \varphi m_{t-1} + \alpha_t g_t \tag{9}$$
$$\theta_t \leftarrow \theta_{t-1} - (\varphi_{t+1} m_t + \alpha_t g_t) \tag{10}$$

The reason for Dozat's suggestion is that Dozat wants to synchronize updating θ_t formulas of NAG and Adam, so we modify Adam's easily. Since NAG is a variant of classic momentum, NAG has the problem with fixed φ and α . However, we can fix it with Adam. Before we modify Adam, let take a look at how Adam basically update θ_t:

$$m_t \leftarrow \varphi_t m_{t-1} + (1 - \varphi_t) g_t \tag{11}$$
$$\theta_t \leftarrow \theta_{t-1} + \alpha \frac{m_t}{1 - \varphi_t} \tag{12}$$

Now we use the same technique substituting t with $t-1$ used in (10). We can substitute m_{t-1} with m_t and φ_t with φ_{t+1} in (12).

$$\theta_t \leftarrow \theta_{t-1} + \alpha \frac{\varphi_{t+1} m_t + (1 - \varphi_t) g_t}{1 - \varphi_t} \tag{13}$$

Let us apply the same idea to modify full Adam and create Nadam. Note that when we update $\hat{m}_t$, we only substitute t parameter for first part:

Algorithm 3: Nesterov-accelerated Adaptive Moment Estimation(Nadam)[1]

Result: new θ_t
Require: α: learning rate;
Require: $\varphi_0 ... \varphi_T$: decay factor;
Require: $\nu_0 ... \nu_T$;
Require: $\epsilon \in$ Hyperparametter;
Require: $f(\theta)$: Stochastic objective function parameterized by θ;
Require: θ_0: The initial parameters;
$m_0, n_0, t \leftarrow 0$ (first/second moment vectors);
while θ_t *not converged* **do**
  $t \leftarrow t + 1$;
  $g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$;
  $m_t \leftarrow \varphi_t m_{t-1} + (1 - \varphi_t) g_t$;
  $n_t \leftarrow \nu_t n_{t-1} + (1 - \nu_t) g_t^2$;
  $\hat{m}_t = \frac{\varphi_{t+1} m_t}{1 - \prod_{i=1}^{t+1} \varphi_i} + \frac{(1-\varphi_t) g_t}{1 - \prod_{i=1}^{t} \varphi_i}$;
  $\hat{n}_t = \frac{n_t}{1 - \nu_t}$;
  $\theta_t \leftarrow \theta_{t-1} + \frac{\alpha \hat{m}_t}{\sqrt{\hat{n}_t} + \epsilon}$;
end

1.1.3 Practical and/or Theoretical Results

"In order to test the performance of this Nesterov-accelerated Adam (Nadam), we train a convolutional autoencoder (adapted from Jones (2015)) with three conv layers and two dense layers in each the encoder and the decoder to compress images from the MNIST dataset (LeCun et al., 1998) into a 16-dimensional vector space and then reconstruct the original image (known to be a difficult task)."[1]

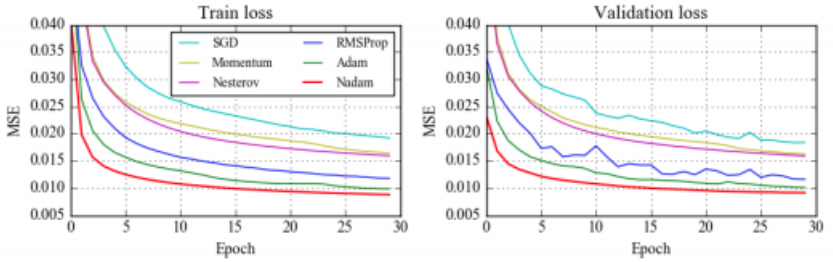

Figure 2: Training and validation loss of different optimizers on the MNIST dataset

In the experiment, SGD, classic momentum, Nesterov, RMSProp, and Adam work expectedly as the theories have shown. Among the old optimizers, Adam is the best and in meanwhile, SGD is the worst, and NAG has better performance than classic momentum. Dozat assumes that if the other optimizers are working correctly, the result would be realistic. This experiment confirms that Nadam is the best optimizer and this expected in Dozat's paper[1]. As we can see, Adam optimizers now can be advanced by combining NAG. Although Nadam has the best empirical performance, Dozat is still unclear about the modifying t to $t - 1$ for m and φt. It is not mathematically proved. Besides, the Nadam experiment is only done in non-sequential data so it needs to be evaluated in different architectures such as RNN as well as online learning.

1.1.4 Relation to Course Material

In this paper, we have walked through more different optimizer configurations and their weakness as well as strengths. In the first part of the paper, I have a chance of reviewing the SGD and classic momentum which we have gone through in class. Later, I have learned more about NAG, Adam, and Nadam which I have not to understand before this class. Different optimizers obviously have different usage cases. SGD is straight forward and easy to implement. Nadam, on the other hand, needs fewer epochs to train but it is harder to build. Depend on datasets and network architectures, we should pick the right optimizer. In this paper, even though I like the idea of Dozat, I am still not sure why he could just parameterize m_{t-1} to ∇f and keep others the same. Does it really make any differences?

References

[1] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[2] A. Ng, "Basic recipe for machine learning (c2w1l03)," 2017. [Online]. Available: https://www.youtube.com/watch?v=C1N_PDHuJ6Q

[3] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *1964*.

[4] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$." 1983.

[5] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[6] I. Mitliagkas, "Ift 6085 - lecture 6 nesterov's accelerated gradient, stochastic gradient descent," 2018.

[7] G. D. Ilya Sutskever, James Martens and G. Hinton, "On the importance of initialization and momentum in deep learning," *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.