Incorporating Nesterov Momentum into Adam

Timothy Dozat
tdozat@stanford.edu

Abstract

This work aims to improve upon the recently proposed and rapidly popularized optimization algorithm *Adam* (Kingma & Ba, 2014). Adam has two main components—a *momentum* component and an *adaptive learning rate* component. However, regular momentum can be shown conceptually and empirically to be inferior to a similar algorithm known as *Nesterov's accelerated gradient* (NAG). We show how to modify Adam's momentum component to take advantage of insights from NAG, and then we present preliminary evidence suggesting that making this substitution improves the speed of convergence and the quality of the learned models.

1 Introduction

When attempting to improve the performance of a deep learning system, there are a handful of approaches one can take–by improving the structure of the model, maybe by making it deeper; by improving the initialization of the model, so that the error signal is evenly distributed throughout the model parameters; by collecting more data or trying a different regularization technique to prevent overfitting; and by using a more powerful optimization algorithm, so that better solutions can be reached in a reasonable amount of time. This work aims to improve the quality of the learned models by providing a more powerful learning algorithm.

Many popular learning algorithms for optimizing non-convex objectives use some variant of stochastic gradient descent (SGD); this work will consider a subset of such algorithms in its examination. Algorithm 1 presents SGD with the notation used in this paper–all following algorithms will add to or modify this basic template:

Algorithm 1 Stochastic Gradient Descent

```
Require: $\alpha_0, \dots, \alpha_T$: The learning rates for each timestep (presumably annealed)
Require: $f_i(\theta)$: Stochastic objective function parameterized by $\theta$ and indexed by timestep $i$
Require: $\theta_0$: The initial parameters
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      $\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$
      $\theta_t \leftarrow \theta_{t-1} - \alpha_t \mathbf{g}_t$
  end while
  return $\theta_t$
```

2 Related Work

Classical momentum (Polyak, 1964) accumulates a decaying sum (with decay factor μ) of the previous updates into a momentum vector $\mathbf{m}$ and replaces the original gradient step in Algorithm 1 with that vector. That is, we modify the algorithm to include the following at each timestep:

$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + \alpha_t \mathbf{g}_t \tag{1}$$

$$\theta_t \leftarrow \theta_{t-1} - \mathbf{m}_t \tag{2}$$

Intuitively, this allows the algorithm to move faster along dimensions of low curvature where the update is consistently small but in the same direction, and slower along turbulent dimensions where the direction of the update is significantly oscillating (Sutskever et al., 2013).

Sutskever et al. (2013) show that *Nesterov's accelerated gradient* (NAG) (Nesterov, 1983)–which has a provably better bound than gradient descent for convex, non-stochastic objectives–can be rewritten as a kind of improved momentum. If we expand the term $\mathbf{m}_t$ in the original formulation of momentum in line 2, we see that the update is equivalent to taking a step in the direction of the previous momentum vector and a step in the direction of the current gradient:

$$\theta_t = \theta_{t-1} - (\mu \mathbf{m}_{t-1} + \alpha_t \mathbf{g}_t) \tag{3}$$

However, the momentum step $\mu\mathbf{m}_{t-1}$ doesn't depend on the current gradient $\mathbf{g}_t$, so we can get a higher-quality gradient step direction by updating the parameters with the momentum step *before* computing the gradient. Sutskever et al. (2013) propose modifying the gradient computation line in the loop of Algorithm 1 as in line 4 below to accomplish this:

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1} - \mu \mathbf{m}_{t-1}) \tag{4}$$

$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + \alpha_t \mathbf{g}_t \tag{5}$$

$$\theta_t \leftarrow \theta_{t-1} - \mathbf{m}_t \tag{6}$$

The authors also provide empirical evidence that this algorithm is superior to SGD, classical momentum, and Hessian-Free (Martens, 2010) for conventionally difficult optimization objectives.

Both classical momentum and NAG define $\mathbf{m}$ as a decaying *sum* over the previous *updates*; however, *Adaptive moment estimation* (Adam) (Kingma & Ba, 2014) defines it instead as a decaying *mean* over the previous *gradients* (this algorithm also includes an adaptive learning rate component not discussed here; cf. Duchi et al. (2011) and Tieleman & Hinton (2012)).

$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t \tag{7}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \frac{\mathbf{m}_t}{1-\mu^t} \tag{8}$$

Using the previous gradients instead of the previous updates allows the algorithm to continue changing direction even when the learning rate has annealed significantly toward the end of training, resulting in more precise fine-grained convergence. It also allows the algorithm to straightforwardly correct for the "initialization bias" that arises from initializing the momentum vector to zero (which is what the $(1-\mu^t)$ term in the denominator of line 8 is for). (Kingma & Ba, 2014) show that their algorithm outperforms a number of others, including NAG, on a small handful of benchmarks.

3 Modifying Adam's Momentum

It often helps to gradually increase or decrease μ over time, so for the rest of this section we will assume a list of values for μ indexed by timestep $\mu_1, \dots, \mu_T$ in order to aid clarity. Before modifying Adam's update rule, we show how to rewrite NAG to be more straighforward to implement (at the cost of some intuitivity). Rather than updating the parameters with just the momentum step in order to compute the gradient, then undoing that step to return to the original paramater state, then taking the momentum step again during the actual update, we can apply the momentum step of timestep $t+1$ only once, during the update of the previous timestep t instead of $t+1$:

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1}) \tag{9}$$

$$\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + \alpha_t \mathbf{g}_t \tag{10}$$

$$\theta_t \leftarrow \theta_{t-1} - (\mu_{t+1}\mathbf{m}_t + \alpha_t \mathbf{g}_t) \tag{11}$$

Notice that line 11 is nearly identical to line 3–the only difference is that the update uses $\mu_{t+1}\mathbf{m}_t$ rather than $\mu_t\mathbf{m}_{t-1}$. It is also easy to see that both the momentum step and the gradient step depend on the current gradient here, unlike with classical momentum.

Now we can use the same trick with Adam's momentum: first we rewrite Adam's update step in terms of $\mathbf{m}_{t-1}$ and $\mathbf{g}_t$, as in line 12, then we substitute the next momentum step for the current one, as in line 13 (taking care of the initialization bias accordingly).

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \left(\frac{\mu_t \mathbf{m}_{t-1}}{1-\prod_{i=1}^{t}\mu_i} + \frac{(1-\mu_t)\mathbf{g}_t}{1-\prod_{i=1}^{t}\mu_i} \right) \tag{12}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \left(\frac{\mu_{t+1} \mathbf{m}_{t}}{1-\prod_{i=1}^{t+1}\mu_i} + \frac{(1-\mu_t)\mathbf{g}_t}{1-\prod_{i=1}^{t}\mu_i} \right) \tag{13}$$

When we change Adam in this way, we get Algorithm 2. However, this form of momentum can in principle be combined with other algorithms that use adaptive learning rates as well, such as *Adamax* (Kingma & Ba, 2014) or *Equilibrated gradient descent* (EGD) (Dauphin et al., 2015).

Algorithm 2 Nesterov-accelerated Adaptive Moment Estimation (Nadam)

```
Require: $\alpha_0, \dots, \alpha_T$; $\mu_0, \dots, \mu_T$; $\nu$; $\epsilon$: Hyperparameters
  $\mathbf{m}_0; \mathbf{n}_0 \leftarrow 0$ (first/second moment vectors)
  while $\theta_t$ not converged do
    $\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$
    $\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + (1 - \mu_t)\mathbf{g}_t$
    $\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1 - \nu)\mathbf{g}_t^2$
    $\hat{\mathbf{m}} \leftarrow (\mu_{t+1}\mathbf{m}_t/(1 - \prod_{i=1}^{t+1} \mu_i)) + ((1 - \mu_t)\mathbf{g}_t/(1 - \prod_{i=1}^{t} \mu_i))$
    $\hat{\mathbf{n}} \leftarrow \nu \mathbf{n}_t/(1 - \nu^t)$
    $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{\mathbf{n}}_t + \epsilon}} \hat{\mathbf{m}}_t$
  end while
  return $\theta_t$
```

4 Experiment

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). We tested six optimization algorithms: SGD, momentum, NAG, RMSProp, Adam, and Nadam, all of which used initialization bias correction and decaying means (rather than decaying sums) where relevant. The best learning rate found for SGD was .2, for momentum/NAG was .5, for RMSProp was .001, and for Adam/Nadam was .002. μ, ν, and ϵ were set to .975, .999, and $1e^{-8}$ respectively (to varying degrees of arbitrarity) and left untuned. The results in Figure 1 show that even though Nadam and Adam have the most hyperparameters, they achieve the best results even with no tuning beyond the learning rate (which is generally unavoidable). Critically, Nadam clearly outperforms the other algorithms–including its parent algorithm Adam–in reducing training and validation loss.

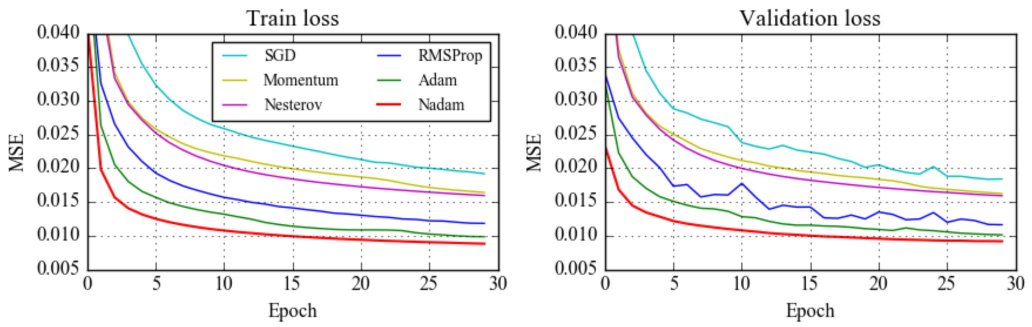

Figure 1: Training and validation loss of different optimizers on the MNIST dataset

5 Conclusion

Kingma & Ba (2014) essentially show how to combine classical momentum with adaptive learning rates, such as RMSProp or EGD, in a clean and elegant way. This work builds on that research by taking their approach one step further, and improving one of the principle components of their algorithm without noticeably increasing complexity.

References

Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems*, pp. 1504–1512, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Mike Swarbrick Jones. Convolutional autoencoders in python/theano/lasagne. Blog post (retrieved February 17, 2016), April 2015. URL `https://swarbrickjones.wordpress.com/2015/04/29/convolutional-autoencoders-in-pythontheanolasagne/`.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.

Boris Teodorovich Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1139–1147, 2013.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.