Faster gradient descent convergence via an adaptive learning rate schedule

Satya Krishna Gorti*
University of Toronto
27 King's College Circle
Toronto, ON M5S
satyag@cs.toronto.edu

Mathieu Ravaut*
University of Toronto
27 King's College Circle
Toronto, ON M5S
mravox@cs.toronto.edu

Abstract

Any gradient descent requires to choose a learning rate. With deeper and deeper models, tuning that learning rate can easily become tedious and does not necesarily lead to an ideal convergence. We propose a variation of the gradient descent algorithm in the which the learning rate η is not fixed. Instead, we learn η itself, either by another gradient descent (first-order method), or by Newton's method (second-order). That way, gradient descent for any machine learning algorithm can be optimized.

1 Introduction

In the past decades, gradient descent has been widely adopted to optimize the loss function in machine learning algorithms *ref*. Lately, the machine learning community has also used the stochastic gradient descent alternative *ref*. Gradient descent can be used in any dimension, and presents the advantage of being easy to understand and inexpensive to compute. Under certain assumptions on the loss function (such as convexity), gradient descent is guaranteed to converge to the minimum of the function. However, stochastic gradient descent has proven to be very efficient even in situations where the loss functions is not convex, as is mostly the case with modern deep neural networks *ref*. Other methods such as Newton's method guarantee a much faster convergence, but are typicaaly very expensive. Newton's method for instance requires to compute the inverse of the Hessian matrix of the loss functions with regards to all parameters, which is impossible with today's hardware and today's deep networks with millions of parameters *ref*.

The quality of a gradient descent heavily depends on the choice of the learning rate η. A too high learning rate will see the loss function jumping around the direction of steepest descent, and eventually diverge. While a very low learning rate guarantees non-divergence, convergence will be very slow, and the loss function might get stuck in a local minimum. Choosing an ideal learning rate requires an intuition of the problem. Typically, researchers would start by performing a line-search over a set of different orders of magnitude of learning rates, but this is long and costly. Besides, a line-search usually assumes a fixed learning rate over time, as doing one line-search per iteration would require exponential computation cost. Usually, we see researchers setting the learning rate to an initial value then decrasing it one of a few items after training has progressed *ref*.

In this paper, we propose to automatically find the learning rate. We still need to input an initial value, but at each iteration, our model will find a learning that optimizes best the loss function at this point of learning. We explore a first-order method and a second-order method to do so, with a strong emphasis on the latter. Our method could be applied to any machine learning algorithm using gradient descent. We show faster convergence on a variety of tasks and models.

2 Related work

3 Adaptive learning rate

An adaptive learning gradient descent has the following schedule:

$$w(t+1) = w(t) - \eta(t)g(t) \tag{1}$$

where $g(t) = \nabla L(w(t))$, L is the loss function, and w(t) represents the state of the model's weights at time t. In the following we present several ways to update $\eta(t)$.

3.1 First-order method

The first-order method consists in doing gradient descent on the learning rate. Let's introduce a function that will be useful in the following:

$$f : R^n \to R \tag{2}$$

$$\eta \to L(w(t) - \eta g(t)) \tag{3}$$

n corresponds to the number of learnable parameters in the model. f represents how the loss would be if we were to perform a gradient descent update with the given η.
The first-order method is written:

$$w(t+1) = w(t) - \eta(t)g(t) \tag{4}$$

$$\eta(t+1) = \eta(t) - \alpha f'(t) \tag{5}$$

This method introduces a new "meta" learning rate α. It has the advantage of begin light in cost as we only need gradients of f. Indeed:

$$\forall \eta, f'(\eta) = -g(t)^T . \nabla L(w(t) - \eta g(t)) \text{ where . is the dot product in dimension n} \tag{6}$$

In particular, at the value of η used to get w(t+1), we get:

$$f'(\eta(t)) = -g(t)^T . g(t+1) \tag{7}$$

3.2 Second-order method

The previous method transfers the problem of choosing an ideal learning rate for weights to choosing an ideal learning rate for the learning rate itself. To avoid that problem, the second-order method uses a Newton-Raphson algorithm:

$$w(t+1) = w(t) - \eta(t)g(t) \tag{8}$$

$$\eta(t+1) = \eta(t) - \frac{f'(\eta(t))}{f''(\eta(t))} \tag{9}$$

Now all the learning only depends on the loss. However, the second derivative of requires building its Hessian matrix:

$$\forall \eta, f''(\eta) = g(t)^T . H_L(w(t) - \eta g(t)) \tag{10}$$

We propose to approximate this Hessian using finite differences. By using on f' then f", we get an update formula of the learning rate depending only on several values of the loss function:

$$f'(\eta + \epsilon) \approx \frac{f(\eta + 2\epsilon) - f(\eta)}{2\epsilon} \tag{11}$$

$$\text{and } f'(\eta - \epsilon) \approx \frac{f(\eta) - f(\eta - 2\epsilon)}{2\epsilon} \tag{12}$$

$$\text{so } f''(\eta) \approx \frac{f(\eta + 2\epsilon) + f(\eta - 2\epsilon) - 2f(\eta)}{4\epsilon^2} \tag{13}$$

Given the finite differences on the first derivative:

$$f'(\eta) \approx \frac{f(\eta + \epsilon) - f(\eta - \epsilon)}{2\epsilon} \tag{14}$$

We get the final simple formula for η:

$$\eta(t+1) = \eta(t) - 2\epsilon \frac{(f(\eta + \epsilon) - f(\eta - \epsilon))}{f(\eta + 2\epsilon) + f(\eta - 2\epsilon) - 2f(\eta)} \tag{15}$$

3.3 BFGS

4 Experiments

4.1 Linear regression

4.2 Image classification with neural networks

4.3 Logistic regression

5 Further exploration

5.1 Learning the learning rate

5.2 Momentum

5.3 Getting loss values via a validation set

6 Limitations

6.1 Choice of step ϵ

6.2 Choice of the initial learning rate

6.3 Cost of loss computations

7 Conclusion

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.