No more learning rate tuning in gradient descent

Introduction to Machine Learning CSC2515 Project Proposal

Mathieu Ravaut, Satya Gorti

October 31, 2017

Abstract

We propose a variation of the gradient descent algorithm in the which the learning rate η is not fixed. Instead, we learn η itself via Newton method. That way, gradient descent for any machine learning algorithm can be optimized.

1 Context

For a generic gradient descent problem with a cost function L that we want to minimize, let's introduce the following notations:

- $\omega(t)$ represents the weights at instant t
- $g(t) = \nabla L(\omega(t))$ the gradient of the cost function at instant t
- $\eta(t)$ is the learning rate at instant t

and the gradient descent problem with an adaptive learning rate can be written:

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

$$\eta(t+1) = \eta(t) - \frac{f'(\eta(t))}{f''(\eta(t))} \tag{2}$$

where f is a real function defined by:

$$f : \eta \to L(\omega(t) - \eta.g(t))$$

Getting the first derivative of f is straightforward:

$$f'(\eta) = -g(t)^T.\nabla L(\omega(t) - \eta g(t)) \text{ for any } \eta \tag{3}$$

Let's note that:

$$f'(\eta(t)) = -g(t)^T.g(t+1) \tag{4}$$

but its second derivative is a bit harder to get.

2 Analytical formula

The analytical formula for the second derivative is given by:

$$f''(\eta) = g(t)^T.H_{\omega(t)-\eta.g(t)}(-g(t)) \tag{5}$$

However, with n parameters in the model, the Hessian has dimension $n \times n$, so it is very expensive to compute. We would then need a cheap way to approximate this second derivative.

3 Finite differences

We propose to approximate the second derivative of f via the **finite differences** method, which has a very nice expression in this case:
For any ϵ:

$$f'(\eta(t)) \approx \frac{f(\eta(t)+\epsilon) - f(\eta(t)-\epsilon)}{2\epsilon} \tag{6}$$

and:

$$f''(\eta(t)) \approx \frac{f(\eta(t)+2\epsilon) + f(\eta(t)-2\epsilon) - 2f(\eta(t))}{4\epsilon^2} \tag{7}$$

so:

$$\eta(t+1) \approx \eta(t) - 2\epsilon \frac{f(\eta(t)+\epsilon) - f(\eta(t)-\epsilon)}{f(\eta(t)+2\epsilon) + f(\eta(t)-2\epsilon) - 2f(\eta(t))} \tag{8}$$

To avoid overflowing problems, we can introduce **Laplacian smoothing**:

$$\eta(t+1) \approx \eta(t) - 2\epsilon \frac{f(\eta(t)+\epsilon) - f(\eta(t)-\epsilon) + \alpha}{f(\eta(t)+2\epsilon) + f(\eta(t)-2\epsilon) - 2f(\eta(t)) + \alpha} \text{ for a small real } \alpha \text{ to set} \tag{9}$$

Finally, to avoid having to set both parameters ϵ and α, we can first set $\alpha = \epsilon$, which leads to the formula:

$$\eta(t+1) \approx \eta(t) - 2\epsilon \frac{f(\eta(t)+\epsilon) - f(\eta(t)-\epsilon) + \epsilon}{f(\eta(t)+2\epsilon) + f(\eta(t)-2\epsilon) - 2f(\eta(t)) + \epsilon} \tag{10}$$

4 Experiments

We plan on exploring the effects of this method on the quality of convergence. Quality means both **speed of convergence**, and avoidance of **local minima**. This general method for automatic learning rate setting can be applied to any machine learning task involving gradient descent. We will compare traditional gradient descent with a fixed learning rate with our method on the following examples:

- Linear Regression. The proposed dataset is the Boston House Prices dataset studied in Assignment 1.
- Logistic Regression.
- Image classification via neural networks. Application to simple datasets such as MNIST or CIFAR-10.