Adam

1 Introduction

Adam is a method for efficient stochastic optimization that only requires firstorder gradients with little memory. The method computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients; the name *Adam* is derived from "adaptive moment estimation". Some of *Adam* advantages are that the magnitudes of parameter updates are invariant to the rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

2 Algorithm

Let $f(\theta)$ be a noisy objective function: a stochastic scalar function that is differentiable w.r.t. parameters θ. Consider, for example, the following function:

$$f(\beta_0,\beta_1,x,y) = (\beta_0 + \beta_1 x - y)^2 = 0 \tag{1}$$

where $\theta = [\beta_0,\beta_1]$. *Adam* algorithm is used for regression over the parameters θ, given the observations $(\mathbf{x},\mathbf{y})$. For this reason, we are interested in minimizing the expected value of f w.r.t. its parameters θ, i.e.

$$\min_{\theta} \mathbb{E}\left[f(\theta)\right] = \min_{\beta_\mathbf{0},\beta_\mathbf{1}} \frac{1}{N} \sum_{i=1}^{N} f(\beta_0, \beta_1, x_i, y_i)$$

where (x_i,y_i), $i = 1,\ldots,N$ are the observations. *Adam* is an iterative method, which updates parameters $\theta_\mathbf{t}$ according to a batch of observations at each time step. With $f(\theta_\mathbf{1}),\ldots,f(\theta_\mathbf{T})$ we denote the realisations of the stochastic function at subsequent timesteps $1,\ldots,T$. The stochasticity might come from the evaluation at random subsamples (minibatches) of datapoints or from inherent function noise. With $\mathbf{g_t} = \nabla_{\theta_{t-1}} f(\theta_\mathbf{t-1})$ we denote the gradient of f w.r.t. $\theta_\mathbf{t-1}$, evaluated at timestep $t - 1$.

Adam algorithm updates exponential moving averages of the gradient ($\mathbf{m_t}$) and the squared gradient ($\mathbf{v_t}$), where the hyper-parameters $\gamma_1,\gamma_2 \in [0,1)$ control the exponential decay rates of these moving averages. The moving averages themselves are the estimates of the 1st moment (mean) and the 2nd raw moment

(the uncentered variance) of the gradient. However, these moving averages are initialized as **0**'s, leading to moment estimates biased towards zero, especially during the initial timesteps, and especially when the decay rates are small. This initialization bias can be easily counteracted, resulting in bias-corrected estimates $\mathbf{m}\hat{}_t$ and $\mathbf{v}\hat{}_t$. *Adam* method is shown in Algorithm 1.

Algorithm 1 Adam method. $\alpha = 0.001$, $\gamma_1 = 0.9$, $\gamma_2 = 0.999$, $\epsilon = 10^{-8}$

1: **Input:** $f(\theta)$, θ_0, *observations*, *batch size*⁻
2: **Initialize:** $\mathbf{m_0} \leftarrow \mathbf{0}$, $\mathbf{v_0} \leftarrow \mathbf{0}$, $t \leftarrow 0$, *converged* ← *False*
3: **while** not *converged* **do**
4: $t \leftarrow t + 1$
5: *batch* ← extract a batch of *batch _size* points from *observations*
6: $\mathbf{g_t} \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$ (evaluate the gradient of f with the actual parameters in the *batch* of points)
7: $\mathbf{m_t} \leftarrow \gamma_1 \mathbf{m_{t-1}} + (1 - \gamma_1)\mathbf{g_t}$ (First moment estimate)
8: $\mathbf{v_t} \leftarrow \gamma_2 \mathbf{v_{t-1}} + (1 - \gamma_2)\mathbf{g_t}^2$ (Second moment estimate)
9: $\mathbf{m}\hat{}_t \leftarrow \mathbf{m_t}/(1 - \gamma_1)$
10: $\mathbf{v}\hat{}_t \leftarrow \mathbf{v}/(1 - \gamma^t)$
11: $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{\mathbf{m}}_t / (\sqrt{\mathbf{v}\hat{}_t} + \epsilon)$
12: Check if convergency reached and update *converged* accordingly
13: **end while**
14: **return** θ_t

The algorithm takes as input the function $f(\theta)$ to be minimized w.r.t. parameters θ, the initial parameters θ_0, the *observations* of the function, i.e., the set of points (x_i, y_i), $i = 1,...,N$, and the *batch _size* to extract, at each iteration, a batch of points in *observations*. The first and second moments, the step number t, and the flag *converged* are initialized (line 2). *Adam* iterations are executed until one of the algorithm convergence (or stop) criteria is met (lines 3-13). In particular, at each iteration, the step number and the batch are updated (lines 4-5), the evaluation of $\nabla_{\theta_{t-1}} f(\theta_{t-1})$ with respect to the *batch* and the parameters θ_{t-1} is carried out (line 6), and the first and second moments are, accordingly, updated (lines 7-8). Subsequently, new θ_t parameters are calculated from the previous ones, correcting the estimate of the first and second moments (lines 9-11). Lastly, the algorithm checks whether the convergence criteria are met at the end of the current step (line 12).

The method works assuming that *observations* are noisy, i.e., the following relation holds $y_i = h(x_i) + \epsilon$, with $\epsilon \sim N(0,\sigma^2)$, where h is an arbitrary function with parameters θ. Considering Equation (2), for example, data points are the realizations of the following function

$$y_i = h(x_i) + \epsilon = \beta_0 + \beta_1 x_i + \epsilon,\ i = 1,...,N,\ \epsilon \sim N(0,\sigma^2) \quad (2)$$

and the *Adam* method can be used to make regression over parameters β_0 and β_1 by minimising the mean squared error with respect to the observations, i.e.

$$\theta = \arg\min_{\theta} \mathbb{E}[f(\theta)] = \arg\min_{\beta_0,\beta_1} \frac{1}{N}\sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)^2 \quad (3)$$

Notice that all the operations with vectors are element-wise; for example, $\mathbf{g_t}^2$ denotes the vector whose elements are the squares of the elements of the vector $\mathbf{g_t}$. Note also that γ_1^t is the power elevation of parameter γ_1 at t, just as for γ_2^t. In addition, for simplicity, we have set the parameters α, γ_1, γ_2 and ϵ.

Regarding the convergence of θ_t, we consider three stopping criteria:

- maximum number of iterations ($t < max\ iterations$);
- Euclidean distance between θ_t and θ_{t-1} ($||\theta_t - \theta_{t-1}||_2 < tolerance$);
- absolute value of the update of the evaluation $f(\theta_t)$, with respect to $f(\theta_{t-1})$, computed on the actual batch of points ($|f(\theta_t) - f(\theta_{t-1})| < tolerance$).

3 Code implementation

The provided code, contains:

- the Point class implementation;
- the Monomial class implementation;
- the FunctionRn class implementation;
- the Adam class implementation;
- the file Utility.h with some helper function to generate the datasets;
- the files Test1.h, Test2.h and Test3.h to test your implementation;

- the main.cpp file, with which you can test your implementation.

Notice that the Monomial is slightly different from the one seen during class, since it is defined as:

$$c\,\beta_0\beta_1 \ldots \beta_M\, x_1 x_2 \ldots x_K$$

where c is the coefficient of the monomial, $\beta_0,\beta_1,\ldots,\beta_M$ are the parameters which we want to tune to minimize the function $f(\theta)$ over the observations, while $x_1,x_2,\ldots,x_K$ are the coordinates of the observations. Since the parameters can also vary, note that all functions that evaluate $f(\theta,\mathbf{x})$ require not only the points (i.e., the observations) on which to evaluate the function, but also the parameters to be considered. For examples

```
double Monomial::evaluate(const Point & observation,
   const Point & parameters) const;

double FunctionRn::evaluate_partial_derivative(std::size_t j, const Point &
   observation, const Point & parameters) const;
```

You have to implement the function for applying the *Adam* method

```
Point Adam::solve(const Point & initial_parameters);
```

which takes as input the initial parameters, defined as a Point, and returns a Point corresponding to the best values of the parameters found, i.e., the ones that minimize $f(\theta)$.