OptML: Optimization in non-stationary environments with Adam

Intro

Adam uses first and second order momentum estimations from encountered gradient to adjust the effective learning rate of each dimension in order to accelerate convergence. What happens if the environment producing the gradients (data, loss function, ...) is not stationary ?

The process producing the data is usually assumed to be stationary in Machine Learning. However, it may not be the case. For instance, in Reinforcement Learning, the environment the agent interacts with may change over time. Therefore we propose two simple experiments to study the behavior of a wildly used optimization algorithm, Adam, in a synthetic non-stationary stochastic process.

Theory

[Adam algorithm]

[Show that Adam estimates an effective learning for each dimension]

[Show convergence rate of Adam in comparison to SGD]

Experiment 1: Minimization of a quadrature-modulated parabola

(continuity changes)

Modeling

let

$$f(x,t) = \alpha(t)(x-u)^2, \text{ with } \mathrm{x} \in \mathbb{R}$$

be a function to optimize. The analytic solution is

$$x^\star = \arg\min_{x \in \mathbb{R}} f(x,t) = u$$

Idea

Simulate a non-stationary environment by modulating the quadrature $\alpha(t)$ of f and compare the convergence of Adam versus SGD. The idea is that as Adam estimates an effective learning rate based on the previous gradients, it may fail (diverge) depending on the dynamics of $\alpha(t)$. On the other hand, SGD simply uses the gradient locally and therefore should not be affected badly by $\alpha(t)$.

method

The dynamics model for the quadrature is

$$\alpha(t) = \begin{cases} c & \text{if } t \leq T \\ \frac{A}{2}[1 + sin(\omega t)] + \epsilon & \text{if } t > T \end{cases}$$

where:

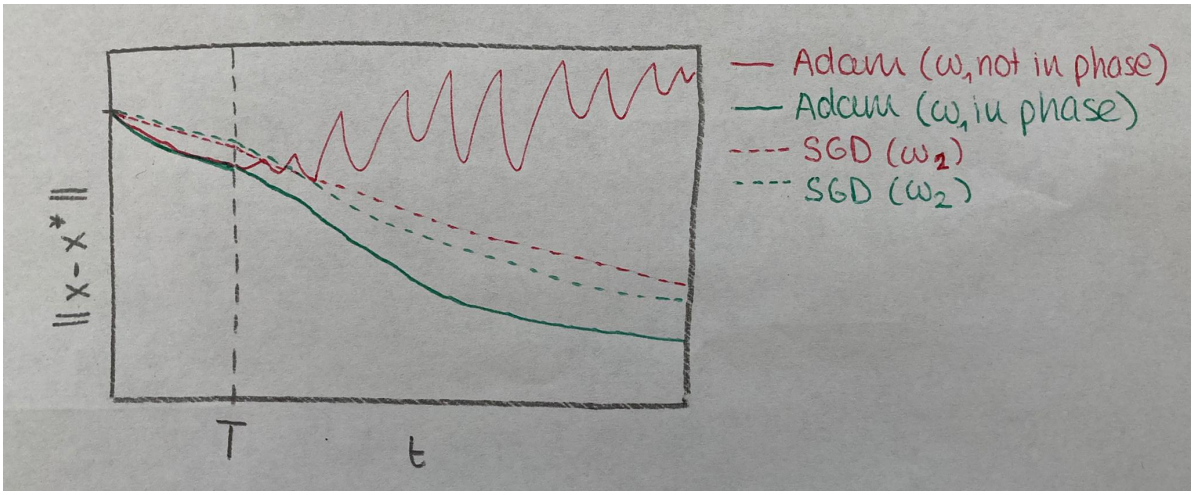
- c and A are constants
- ω is a pulsation
- ϵ is a small positive constant avoiding the curve to become completely flat to ensure that $x^\star$ is the global minimum
- T is a constant number of time steps allowing Adam to build an initial representation before entering the non-stationary regime

Using a sinusoidal representation may allow us to analyze the convergence or divergence of the algorithms as being **in phase** or **not in phase** with the environment.

Then, we propose the following **algorithm**

- Let Δ_{adam} be a list
- Let Δ_{sgd} be a list
- For various ω
    - Let δ_{adam} be a list
    - Let δ_{sgd} be a list
    - let T be a list
    - compute $x^\star$
    - Initialize x_{adam} randomly
    - Initialize $x_{sgd} = x_{adam}$
    - initialize $t = 0$
    - For as many time steps as wanted
        - update x_{adam} by performing Adam step on $f(x, t)$
        - update x_{sgd} by performing SGD step on $f(x, t)$
        - append $||x_{adam} - x^\star||^2$ to δ_{adam}
        - append $||x_{sgd} - x^\star||^2$ to δ_{sgd}
        - append t to T
        - increment t by 1
    - append δ_{adam} to Δ_{adam}
    - append δ_{sgd} to Δ_{sgd}
- plot Δ_{adam} and Δ_{sgd} as functions at each time steps in T

The expected result should look qualitatively like the following schematic



Discuss findings

Experiment 2: Classification of non-stationary Gaussian processes

(discontinuity changes)

Modeling

In that first experiment we can consider a simple non stationary data model. Let

$$Y \sim Bernouilli(0.5)$$

be the binary random variable from which the labels are drawn and

$$X_{|y}(t) \sim \mathcal{N}(\mu_{|y}(t), \sigma^2_{|y}(t)I), \text{ with } \mu_{|y}(t) \in \mathbb{R}^d$$

be the distributions of the data points given a label y.

Let us consider a simple least square model

$$\mathcal{L}(\theta) = \frac{1}{N}(Y - X\theta)^\top (Y - X\theta) = \frac{1}{N}\sum_{n=1}^{N}\left[\theta^\top \mathbf{x}_n - y_n\right]^2 = \frac{1}{N}\sum_{n=1}^{N}\left[f(\mathbf{x}_n, \theta) - y_n\right]^2$$

where N is the number of data points available. The analytical solution is given by

$$\nabla\mathcal{L}(\theta) = X^\top(Y - X\theta) = 0 \Rightarrow \theta^\star = (X^\top X)^{-1}X^T Y$$

and can also be numerically approximated using gradient descent type algorithms.

Idea

Simulate a non-stationary Gaussian process by reseting $\mu_{|y}(t)$ and $\sigma^2_{|y}(t)$ at random timestamps. Observe the adaptation of Adam against a SGD baseline.

Method

First, we define an additional random variable

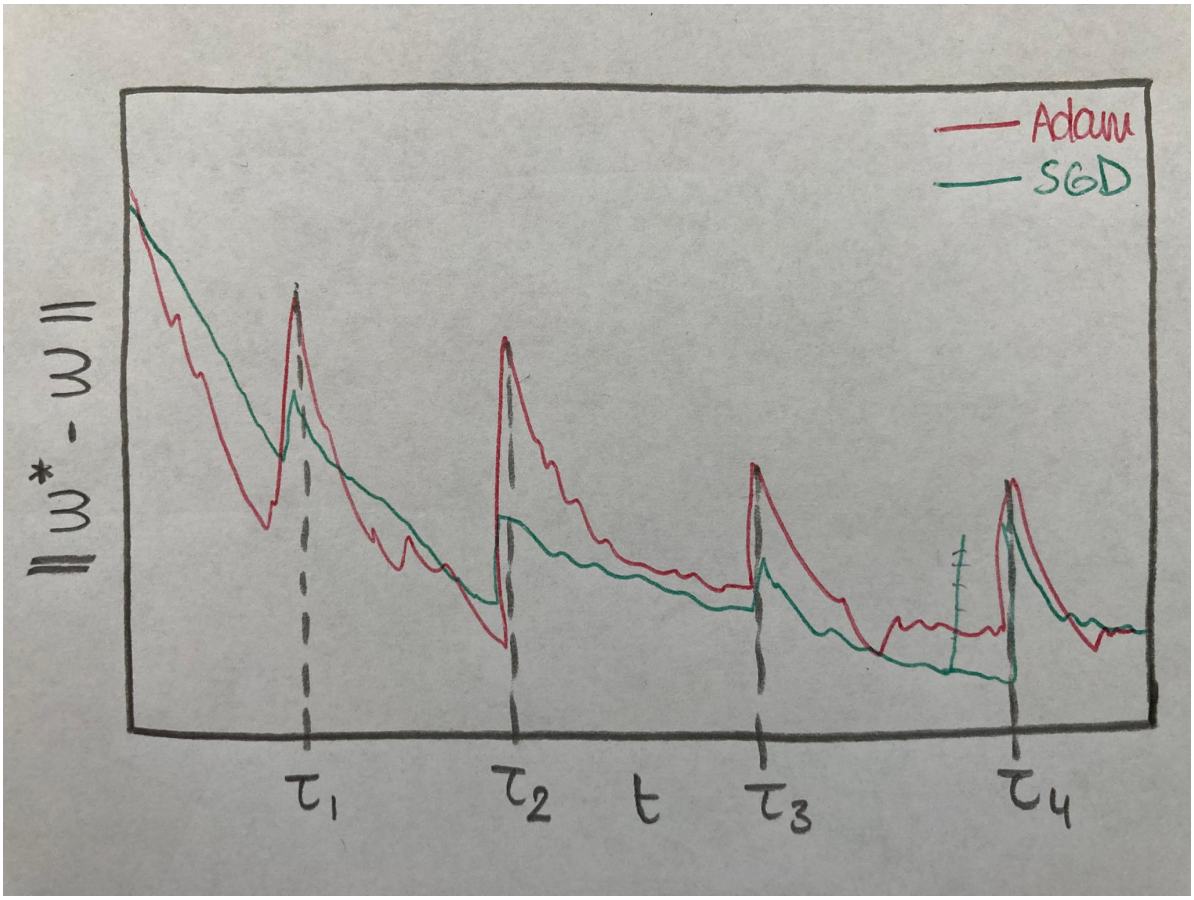
$$\tau \sim expo(\lambda)$$

which will characterize the lifetime of the $X_{|y}(t)$.

Then, we propose the following **algorithm**

- Let δ_{adam} be a list
- Let δ_{sgd} be a list
- let $\mathcal{T}$ be a list
- let T be a list
- Initialize $\tau \sim exp(\lambda)$
- append τ to $\mathcal{T}$
- Initialize the data distributions $X_{|y}(t)$ randomly
- Initialize $\theta^\star$
- Initialize θ_{adam} randomly
- Initialize $\theta_{sgd} = \theta_{adam}$
- initialize $t = 0$
- For as many time steps as wanted
  - sample $\{(x_n, y_n)\}_{n=1}^N$ where $y_i \sim Y$ and $x_i \sim X_{|y_i}$
  - perform Adam step on θ_{adam}
  - perform SGD step on θ_{sgd}
  - append $||\theta_{adam} - \theta^\star||^2$ to δ_{adam}
  - append $||\theta_{sgd} - \theta^\star||^2$ to δ_{sgd}
  - append t to T
  - increment t by 1
  - if $t = \tau$
    - sample $\tau \sim exp(\lambda)$
    - append τ to $\mathcal{T}$
    - Reinitialize the data distributions $X_{|y}(t)$ randomly
- plot δ_{adam} and δ_{sgd} as functions at each time steps in T

The expected result should look qualitatively like the following schematic



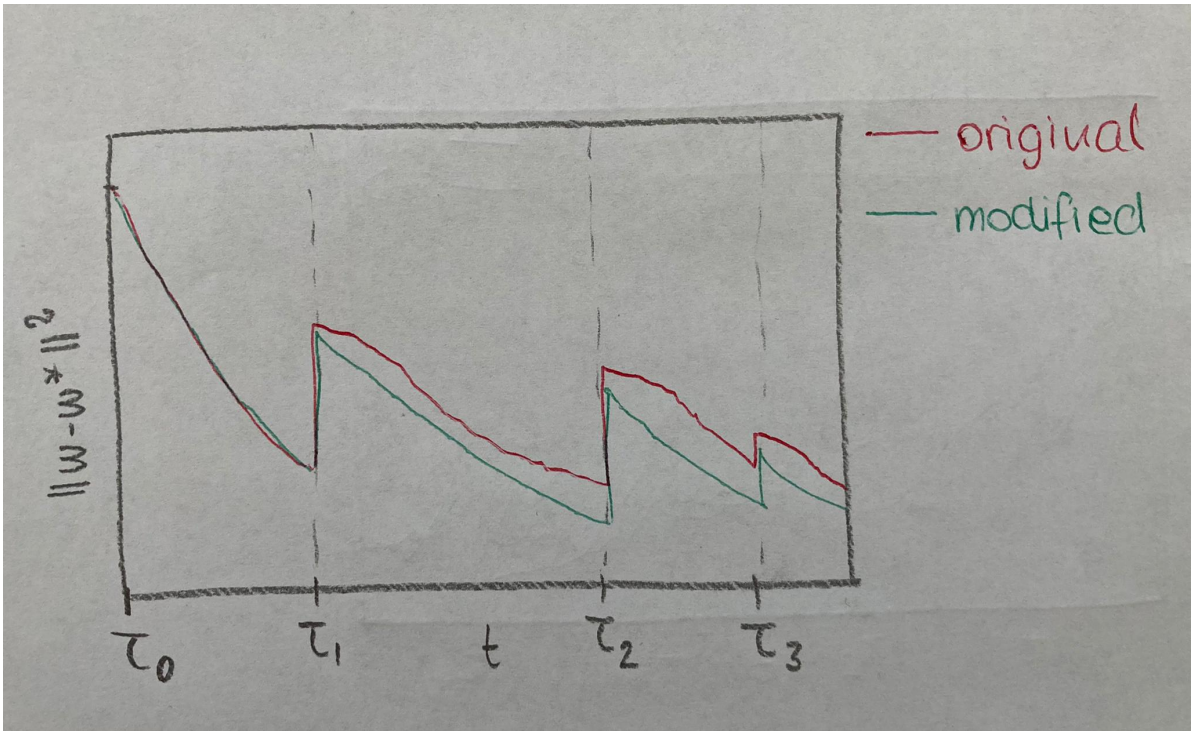
Then discuss the results.

Solution proposition

One may want to add an extra threshold parameter to the Adam algorithm so that if the variation in between the estimates and the value for the batch exceed the threshold, the estimates are reset to zero.

We could do a little experiment similar to experiment 1 in which at each change of distribution, we reset the internal state of Adam. Compare with true Adam.

expected:


original
modified
$\|w - w^*\|^2$
τ_0
τ_1
t
τ_2
τ_3