SRS document for Stochastic Optimization & Visualization

08-04-2021

Background:

Definitions

Optimization Mathematical optimization (alternatively spelled optimisation) or mathematical programming is the selection of a best element, with regard to some criterion, from some set of available alternatives.

In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function.

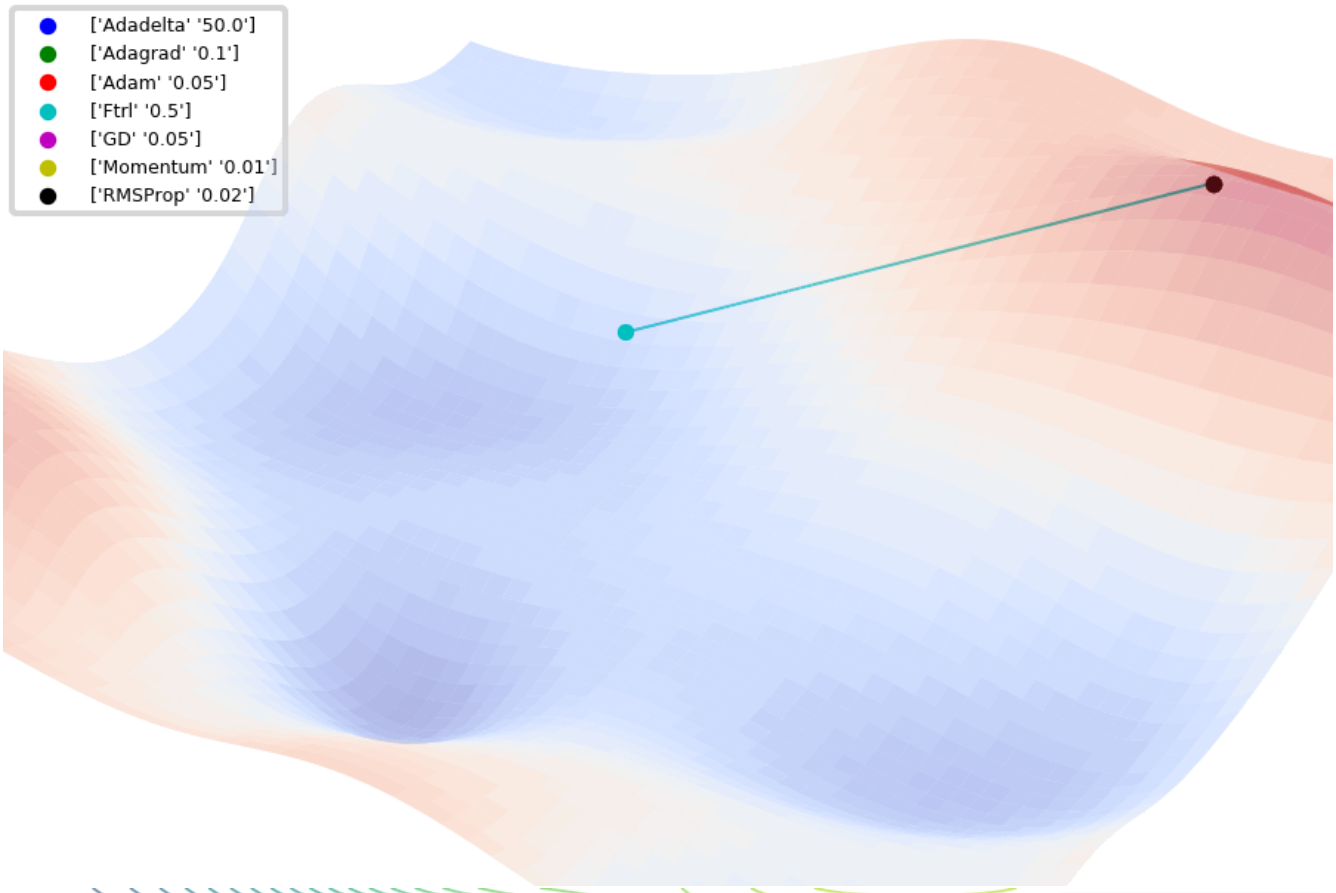
In this project we will be focusing on finding the minima, given the function. More specifically, we will focus on using Stochastic Gradient Descent Methods for finding minima of functions.

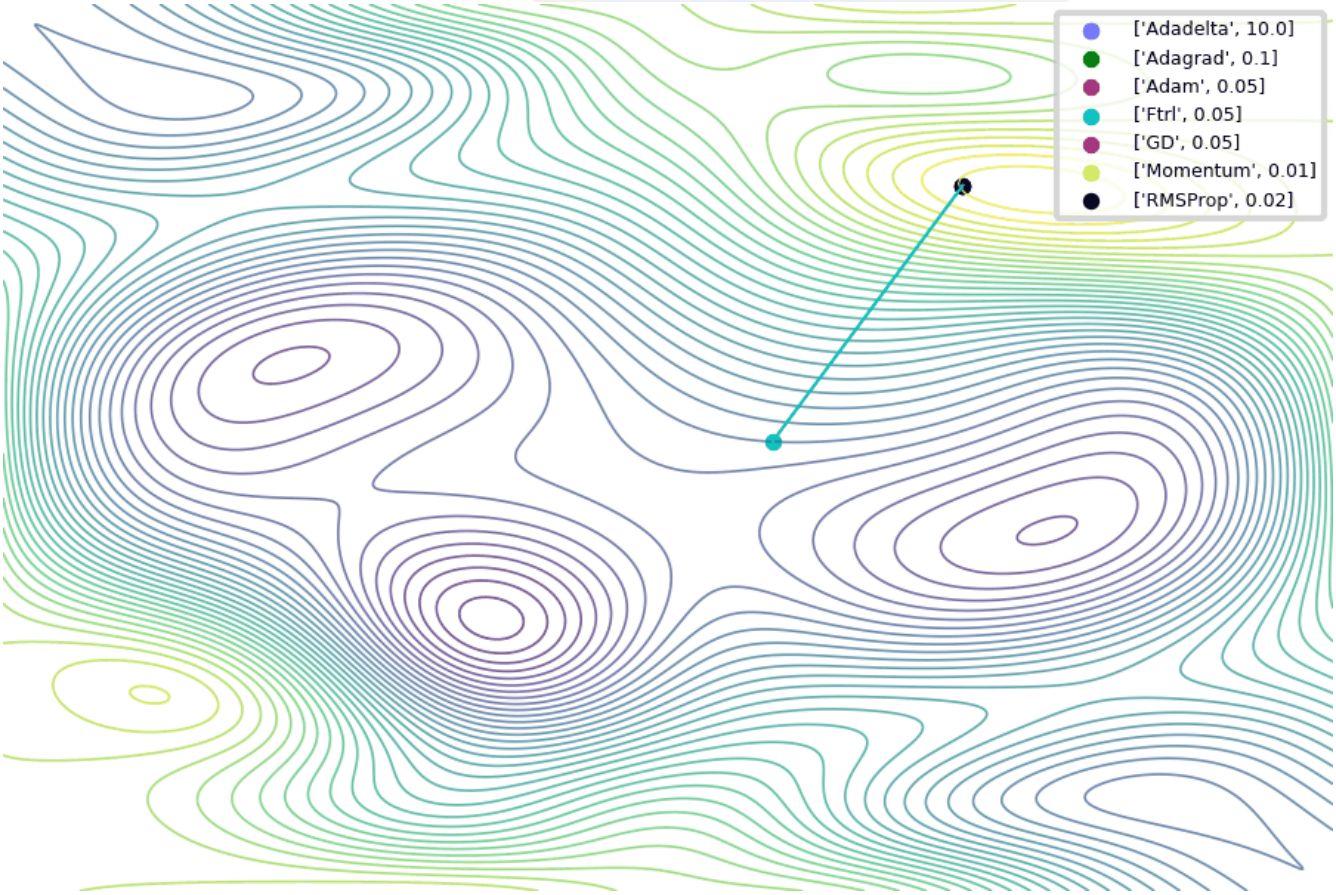
Stochastic Gradient Methods Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (Differentiable etc.). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). This is extremely useful in topics like Machine Learning, where we need to find minima of a loss function without having to do too much computation.

The software that we aim to produce makes the process of visualizing the directions taken by the optimizer easier.

How do you visualize optimizers?? With any iterative optimization method, we start at a starting point x_0 which is then changed in succesive iterations to generate points that give lower value of function that we try and optimize. Let the list of points generated be $x_0, x_1 \dots x_n$ and the function we want to optimize is $f : \mathbb{R}^n \to \mathbb{R}^n$

We first plot the loss function over a range; then for each x_i, we plot $f(x_i)$ to get a point, and interpolate in between them. We might get something like this:


['Adadelta' '50.0']
['Adagrad' '0.1']
['Adam' '0.05']
['Ftrl' '0.5']
['GD' '0.05']
['Momentum' '0.01']
['RMSProp' '0.02']


['Adadelta', 10.0]
['Adagrad', 0.1]
['Adam', 0.05]
['Ftrl', 0.05]
['GD', 0.05]
['Momentum', 0.01]
['RMSProp', 0.02]

Mathematical aspects of optimization

We focus on the simple case of standard gradient descent.

We take a initial point x_0 and a function f and and a learning rate. We succesively update x_0 as follows

$$x_{i+1} = x_i - \eta * \nabla_{x=x_i} f(x)$$

here η is called the learning rate, which is used to regulate the step size that we take. Even though this is a very elementary example, many new algorithms have come up which are way mmore efficient than this.

This software shows the visualization of SGD methods on a vareity of loss functions and algortihms.

Overall Description

Product Perspective

System Interface This is primarily a software which displays visualizations on a webpage. Hence we would like this software to work on MacOS, Windows and Linux machines on google chrome, firefox and microsoft egde browsers.

User Interface

- The user must be able to interact with the application through GUIs and also the website should be fast (without delays).
- Since the application is to be used by students and the researchers , we want that much of the work can be done by the help of keyboard only (i.e. less use of mouse) , so we want the following shortcuts to be implemented :
- Help - to open the help tab (which will show them an example of how to use the site with a step by step example and will also have the shortcuts which we provide)
- Plot - as soon as user enters all the data and press 'Shift + Enter' the graph plotting hould start.
- Choose optimizer - a shortcut to open the drop down menu to choose the optimiser that is to be used.
- choose loss function - a shortcut to open the drop down menu to choose the loss function that is to be use
- Save shortcut - Will save a gif of the plot(that is how with different iterations the function reaches which point) and also the data that was used to make the plots.
- Toggling between different numerical fields - learning rate , range to plot, starting point, function that is to be optimised.

Hardware Interface

- supported device: MACOS, LINUX, WINDOWS
- nature of data is mostly Floating point arrays and some strings
- Most of the backend runs on the CPU on the programming language of choice.

Software interface This portion explains the possible software libraries and frameworks that one could make use of while creating this software:

- Visualization: PlotlyJS, Bokeh, Plots.jl, RPlots (To make the plots)
- Languages: Julia, Python, R
- Web interface: Javascript
- Frontend Libraries: Any javascript frontend library of designer's choice ,like ReactJS , Angular,etc.
- Backend(calculations to be run on the user's machine): Python preffered and using python modules like numpy which are pretty fast.

NOTE: If the user does not have the dependencies and the programming languages to install them, there must be a installer script for all the dependencies. Please note that this might vary depending on the operating system

Communication interface

- In the frontend we take the following data -

1) Function (choose from some pre-chosen functions)
2) The optimizer and the hyperparameters of the optimizer.
3) range to plot (should depend on the function chosen)
4) starting point (should be between the range to plot points)
5) Number of iteration (we put a limit of 200)

Then we sent this data to our backend and then backend should return the follwing values:

- the list of values taken by the variable over iterations as a list/array.
- the value of loss function at each of these points

All the computations in the backend is done on the user's hardware sestem; which is then shown on the web-browser of the user.

Then the frontend module should use the relevant libraries to show how the function is optimised. These points should be connected by splines . When the user hovers over these points the plot should show the relevant data correspoing to that point, Eg. the loss value at that point, number of iterations done at that point, the value of the function at that point.

Memory Constraints

- The software must be lighweight, not more than 500 MB.

Product function

This is an application which helps in the visualization of the effectiveness of different optimizers in different settings(changing hyperparameters, loss functions etc.). We ask users to choose various

parameters like the function to be optmized, the starting point, the range in which we have to optimise the function, the optimiser to use and the number of iterations to for which we need to run the optimizer.

There are various checks for the input data, some might be in frontend and some in the backend but the application should work optimally.

User Characteristics

The majority of of the users of this software will be students(both college and school going) and teachers(schools and colleges). While students may use it to explore gradient based optimizations on their own. Teachers may use this to teach students about these optimization methods

Constraints, assumptions, and dependencies

Since python is used to run calculations, the user needs to have python and other libraries that may be used in the building of the programs. An assumption that we make is that the number of iterations is limited to 200 for better resource usage.

Specific Requirements

Performance requirements

Since we are making visualising software, it is really important that we have performant code. Using optimized libraries like `numpy` in python, or using precompiled binaries in languages made for fast computation might be useful.

Logical Database requirements

The only data we store is the list of points that is taken while optimzation to help plotting. New optimizers and tester functions can be directly embedded in the code base instead of having to save them in a database

Software system attributes

Reliability The software must prove to be stable if it is to be used widely. Since security is an important aspect of reliability, please refer to sections on Security Requirements.

Availability We aim to make this software to be free for everyone. While the development of this software would be closed-source, we want to release an open source version after we have a stable product. This can enable us to find bugs more easily, and enable contributions

Security Requirements Proper input sanitisations with the contraints as mentioned in the Communication Interface part. Also the software should do proper error handling on explosion of gradients or loss values(like in case of Inf and NaN values)

Maintainability The code must be optimally modular to help finding bugs and writing tests. Optimal modularity of code will also enable easy testing and addition of new features. The documentation on the software must be of high quality, to ensure easy learning curve for new developers and contributors. The code in the software must confer to a style guide, for example Black for python and prettier for javascript etc.

Portability The users need to be able to use this application on OSes like Windows, MAC and linux: on web browsers like google Chrome, Firefox and Microsoft edge. Since this is a standalone project , so there is no interoperatability with other projects.

Hence, we need not provide any means to embed this software in other programs.