5M project: stochastic optimization

Big data analytics

Project outline

The purpose of this project is to learn the basics of contemporary stochastic optimization methods. You will learn how such methods operate by implementing them and by tuning their hyperparemeters. The stochastic optimization algorithms considered in the project are stochastic gradient descent (SGD), stochastic gradient descent with momentum (MSGD), stochastic gradient descent with Nesterov accelerated gradient (NAGSGD), AdaGrad, RMSProp and Adam.

Several of these stochastic optimization approaches were developed to train convolutional neural networks. For the purposes of the project, linear regression will be used instead of deep learning models. By using a familiar model such as linear regression, your focus will be placed on understanding the stochastic optimization algorithms.

Inference for linear regression coefficients via stochastic optimization will be conducted for three different datasets. The first dataset contains $1,000$ simulated data points and is associated with two regression coefficients, namely the intercept and slope of a simple linear regression model. Starting with simple linear regression and simulated data, you will develop the involved stochastic optimization algorithms. Once you have implemented the stochastic optimization methods, you will use them on two real datasets. The second dataset is available on Kaggle, pertains to weather in Szegen between 2006-2016 and it contains $96,453$ observations and six parameters. The third dataset is the million song dataset (MSD), which you have encountered in the first lab and is available on the UCI machine learning repository. Recall that MSD is a big dataset, containing $515,345$ data points and 91 parameters, so you will have the chance to fit multiple linear regression on a big dataset using stochastic optimization.

R code template

A `template` folder is available on Moodle, which will help you get started with coding. It contains three R scripts, namely `cost.r`, `optimizers.r` and `question1.r`.

The script `cost.r` provides the definition of the cost function and its gradient for linear regression. This is the only cost function needed for the project, since linear regression is the only model applied to the three datasets.

The script `optimizers.r` provides the R function `gd()` for gradient descent (GD). You can use the `gd()` function for answering the relevant questions of the project and as a coding basis for developing the stochastic optimization algorithms. Moreover, `optimizers.r` contains R comments with function signatures for the required stochastic algorithms. You will need to complete the function body of these functions to tackle the project.

The script `question1.r` sources `cost.r` and `optimizers.r` initially, and provides a template of R comments subsequently to help structure your answer to question 1. Towards the end of `question1.r`, commented lines give you directions towards storing the numerical and visual output of your simulations for question 1.

The `template` folder contains also the data files `simulated_data.csv`, `weather_data.csv` and `uci_data.csv` associated with questions 1, 2 and 3, respectively.

An introduction to stochastic optimization

This section outlines the stochastic optimization methods appearing in the project. Gradient descent (GD), which has been taught in lectures 4-5 and has been coded in lab 1, is the starting point for stochastic optimization.

Gradient descent (GD)

Let $J(\boldsymbol{\theta}, D)$ be a cost function, where $\boldsymbol{\theta}$ is an n_θ-length parameter vector and D a dataset of n_d samples. Moreover, let $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D)$ be the gradient of cost function $J(\boldsymbol{\theta}) := J(\boldsymbol{\theta}, D)$ with respect to $\boldsymbol{\theta}$. The approximation $\boldsymbol{\theta}^{(k)}$ of $\boldsymbol{\theta}$ at step k of GD given the approximation $\boldsymbol{\theta}^{(k-1)}$ of $\boldsymbol{\theta}$ at step $k-1$ is defined as

$$\boldsymbol{\theta}^{(k)} := \boldsymbol{\theta}^{(k-1)} - a \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k-1)}, D), \tag{1}$$

where the learning rate a is a positive real number.

In the case of a linear regression model, the data are set to $D := \{X, y\}$, where X is the $n_d \cdot n_\theta$ design matrix and y is the n_d-length output vector associated with linear regression.

Stochastic gradient descent (SGD)

For a big data set D consisting of a large number n_d of data points, the evaluation of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D)$ can be computationally expensive. To reduce the computational complexity of GD, stochastic gradient descent (SGD) samples an s-size subset $D_s^{(k)}$ of the data D at step k and evaluates the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D_s^{(k)})$ using the subset $D_s^{(k)}$. Thus, the approximation $\boldsymbol{\theta}^{(k)}$ of $\boldsymbol{\theta}$ at step k of SGD given the approximation $\boldsymbol{\theta}^{(k-1)}$ of $\boldsymbol{\theta}$ at step $k-1$ is given by

$$\boldsymbol{\theta}^{(k)} := \boldsymbol{\theta}^{(k-1)} - a \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k-1)}, D_s^{(k)}). \tag{2}$$

SGD is more computationally efficient than GD in large-scale machine learning problems entailing large n_d or large n_θ.

As an implementation hint, use the `sample()` function in R to sample s out of n_d samples at step k without replacement. Subsequently, make sure that you index the design matrix X and output variable y appropriately.

The rest of stochastic optimization methods of this project also use a random subset $D_s^{(k)}$ of the data D at step k.

SGD with momentum (MSGD)

In SGD with momentum (MSGD), a 'momentum' term $\mathbf{m}$ is defined, which is a moving average of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D_s^{(k)})$, and the parameters are then updated using the momentum $\mathbf{m}$ instead of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, D_s^{(k)})$.

The momentum $\mathbf{m}^{(k)}$ and parameter approximation $\boldsymbol{\theta}^{(k)}$ at the k-th step of MSGD are defined recursively according to

$$\mathbf{m}^{(k)} := b\mathbf{m}^{(k-1)} + (1-b)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k-1)}, D_s^{(k)}), \tag{3}$$

$$\boldsymbol{\theta}^{(k)} := \boldsymbol{\theta}^{(k-1)} - a\mathbf{m}^{(k)}. \tag{4}$$

MSGD has two hyper-parameters, a momentum memory factor b with values in $(0, 1)$ and a positive learning rate a. Furthermore, an initial momentum value $\mathbf{m}^{(0)}$ is required by equation (3) at step $k = 1$.

For values of b closer to one, the moving average equation (3) retains stronger memory of momentum history. For values of b closer to zero, the moving average equation (3) remembers less any past momentum history and places more emphasis on the current gradient of the cost function.

Variant representations of MSGD are available in the literature. Equations (3)-(4) have been preferred due to providing a clearer understanding of MSGD.

SGD with Nesterov accelerated gradient (NAGSGD)

SGD with Nesterov accelerated gradient (NAGSGD) differs from MSGD only in the way it calculates the gradient of the cost function in the moving average equation. More specifically, NAGSGD calculates the gradient of the cost function with respect to approximate future position $\boldsymbol{\theta}^{(k-1)} - ab\mathbf{m}^{(k-1)}$ rather than with respect to the current parameter approximation $\boldsymbol{\theta}^{(k-1)}$.

Thus, step k of NAGSGD is defined as

$$\mathbf{m}^{(k)} := b\mathbf{m}^{(k-1)} + (1-b)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k-1)} - ab\mathbf{m}^{(k-1)}, D_s^{(k)}), \tag{5}$$

$$\boldsymbol{\theta}^{(k)} := \boldsymbol{\theta}^{(k-1)} - a\mathbf{m}^{(k)}. \tag{6}$$

Alternative expressions for NAGSGD can be found in the literature. Equations (5)-(6) have been chosen for the purpose of communicating the moving average model for momentum and the resulting parameter update clearly.

AdaGrad

The learning rate a of SGD can be too small for one parameter and too large for another one. This problem tends to manifest itself for large number n_θ of parameters. AdaGrad mitigates such a problem by scaling a different learning rate for each parameter adaptively and automatically.

Using the shorthand $\mathbf{g}^{(k)} := \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k)}, D_s^{(k)})$, consider the outer product $\mathbf{g}^{(k)} \times \mathbf{g}^{(k)}$ at step k, which is an $n_\theta \cdot n_\theta$ matrix. Summing over all steps up to k yields the $n_\theta \cdot n_\theta$ matrix

$$G^{(k)} := \sum_{\ell=0}^{k} \mathbf{g}^{(\ell)} \times \mathbf{g}^{(\ell)}. \tag{7}$$

At step k, AdaGrad performs the parameter update

$$\boldsymbol{\theta}^{(k)} := \boldsymbol{\theta}^{(k-1)} - a(\mathrm{diag}(G^{(k-1)}) + \epsilon)^{\odot(-0.5)} \odot \mathbf{g}^{(k-1)}, \tag{8}$$

where a is a positive learning rate, $\mathrm{diag}(G^{(k-1)})$ is the diagonal of matrix $G^{(k-1)}$, ϵ is a positive smoothing term that avoids division by zero (usually on the order of $1e-8$) and $\odot$ denotes Hadamard (element-wise) operations. Step $k=1$ requires an initial value $G^{(0)}$.

To further clarify AdaGrad, equation (8) will be written below in an alternative form by using element-wise notation instead of matrix notation. Let $g_j^{(k)} := \partial J(\boldsymbol{\theta}^{(k)}, D_s^{(k)})/\theta_j$ be the j-th partial derivative of the cost function at step k, i.e. the j-th coordinate of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k)}, D_s^{(k)})$. The AdaGrad update of the j-th parameter $\theta_j^{(k)}$ at step k is

$$\theta_j^{(k)} := \theta_j^{(k-1)} - \frac{a}{\sqrt{G_{jj}^{(k-1)} + \epsilon}} g_j^{(k-1)}, \tag{9}$$

where $G_{jj}^{(k-1)}$ is the (j,j)-th diagonal element of $G^{(k-1)}$.

Equation (9) makes it clear that AdaGrad uses a differently scaled learning rate $a/\sqrt{G_{jj}^{(k-1)} + \epsilon}$ for each parameter θ_j.

As a hint to assist implementation, you may focus on equation (9). You do not need to compute the outer products appearing in equation (8). It suffices to calculate the diagonal of matrix $G^{(k)}$, which means that it suffices to calculate the sum of squared gradients

$$G_{jj}^{(k)} = \sum_{\ell=0}^{k} (g_j^{(\ell)})^2. \tag{10}$$

AdaGrad eliminates the need to manually tune the learning rate, since it tunes adaptively the learning rate for each parameter according to equation (9). However, the adaptive tuning of learning rates in AdaGrad comes with a weakness. Squared gradients accumulate in the denominator of learning rate $a/\sqrt{G_{jj}^{(k-1)} + \epsilon}$, hence learning rates shrink during training and AdaGrad is no longer able to perform parameter updates via equation (9).

RMSProp

RMSProp attempts to alleviate the aforementioned weakness of AdaGrad by applying a moving average model to squared gradients and by then using the output of the moving average model in the AdaGrad parameter update mechanism.

More concretely, the j-th coordinate $v_j^{(k)}$ of the moving average of squared gradients and j-th parameter $\theta_j^{(k)}$ at step k of RMSprop are updated according to

$$v_j^{(k)} := cv_j^{(k-1)} + (1-c)(g_j^{(k-1)})^2, \tag{11}$$

$$\theta_j^{(k)} := \theta_j^{(k-1)} - \frac{a}{\sqrt{v_j^{(k)} + \epsilon}} g_j^{(k-1)}. \tag{12}$$

The hyperparameter c is known as the memory factor for squared gradients, taking values in $c \in (0,1)$. Equation (11) requires an initial value $v_j^{(0)}$ at step $k = 1$. Taking into account all coordinates $j = 1, 2, \ldots, n_\theta$, the initial value $\mathbf{v}^{(0)}$ for the moving average model of RMSProp is an n_θ-length vector.

Adam

Adaptive moment estimation (Adam) is a stochastic optimization method that computes a different learning rate for each parameter adaptively. Apart from retaining a moving average of squared gradients similarly to RMSProp, Adam stores a moving average of gradients additionally.

The Adam update of the j-th parameter $\theta_j^{(k)}$ at step k is given by

$$m_j^{(k)} := bm_j^{(k-1)} + (1-b)g_j^{(k-1)}, \tag{13}$$

$$v_j^{(k)} := cv_j^{(k-1)} + (1-c)(g_j^{(k-1)})^2, \tag{14}$$

$$\hat{m}_j^{(k)} := \frac{m_j^{(k)}}{1-b^k}, \tag{15}$$

$$\hat{v}_j^{(k)} := \frac{v_j^{(k)}}{1-c^k}, \tag{16}$$

$$\theta_j^{(k)} := \theta_j^{(k-1)} - \frac{a}{\sqrt{\hat{v}_j^{(k)} + \epsilon}} \hat{m}_j^{(k)}. \tag{17}$$

Adam has three hyperparameters, namely a positive learning rate a, a memory factor b of gradients with values in $(0,1)$ and a memory factor c of squared gradients with values in $(0,1)$. Morevover, Adam requires the initial values $\mathbf{m}^{(0)}$ and $\mathbf{v}^{(0)}$ at step $k = 1$.

Equations (13) and (14) provide the moving average of gradients and of squared gradients, respectively. Adam takes its name after these first and second order estimators of gradients. Due to $\mathbf{m}^{(0)}$ and $\mathbf{v}^{(0)}$ being initialized typically as vectors of zeros, the moving averages $\mathbf{m}^{(k)}$ and $\mathbf{v}^{(k)}$ are asymptotically biased towards zero, especially when b and c are set to a value close to one. To contouract such biases while estimating the moving averages of gradients and of squared gradients, equations (15) and (16) are introduced.

Question 1

The goal of question 1 is to develop the stochastic optimizers of this project using a simulated dataset and a cost function for simple linear regression.

To help you get started, the provided `template` folder contains an `optimizers.r` file with R code for gradient descent, a `cost.r` file with R code for the cost function for linear regression and its gradient, a `question1.r` file with some preliminary R code towards answering question 1 and the `simulated_data.csv` file on which simple linear regression will be fitted.

As part of your answer, you will edit the `optimizers.r` file to include in it your R code for the optimizers of this project and you will also edit the file `question1.r` to include in it your R

code for running the optimizers and simulations of question 1. You will not provide any output files or a report as part of your project submission.

Make sure that every optimizer in `optimizers.r` is a standalone R function with a return statement

```
return(list(theta=theta, cost=cost))
```

as in the provided `gd()` function, to store the iterations of the parameter estimates and of the cost function.

You need to make sure before submission that the R command `source("question1.r")` runs successfully and stores the required results in output files as explained further below.

`question1.r` starts with the lines

```
source("optimizers.r")
source("cost.r")
```

to load the R code for the cost function for linear regression and its gradient as well as the R code for the optimizers. The subsequent line in `question1.r` loads the data for question 1:

```
simulated_data <- read.csv("simulated_data.csv", header=TRUE)
```

`simulated_data.csv` consists of two columns; the first column is labelled as `y` and represents the output variable, while the second column is labelled as `covariates` and plays the role of covariate. The remaining lines in `question1.r` are R comments meant to guide you towards answering question 1.

Firstly, standardize the `covariate` and create the design matrix `X` including a first column of ones for intercept.

Fit a simple linear regression to the data using the standardized `covariate` and the `lm()` function in R. The model will have two parameters, namely a regression coefficient corresponding to `covariate` and an intercept.

Use the same initial value $\boldsymbol{\theta}^{(0)} = (7, -8)^T$ for all optimizers.

Run 100 iterations of GD tuning the learning rate a of equation (1) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 100 iterations of SGD by sampling a subset of $s = 100$ data points per iteration. Tune the learning rate a of equation (2) to converge towards the regression coefficient estimates obtained via `lm()`. Notice that even for such a simple example, SGD obtains an approximation to the regression coefficient estimates known from `lm()`. Although SGD does not yield an exact solution to the optimization problem, it provides a reasonable approximation.

Run 100 iterations of MSGD by sampling a subset of $s = 100$ data points per iteration. Set the initial value for momentum to be a vector of zeros, that is set $\mathbf{m}^{(0)} = (0, 0)^T$. Tune the learning rate a of equation (4) and memory factor b of equation (3) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 100 iterations of NAGSGD by sampling a subset of $s = 100$ data points per iteration. Set the initial value for momentum to be a vector of zeros, that is set $\mathbf{m}^{(0)} = (0, 0)^T$. Tune the learning rate a of equation (6) and memory factor b of equation (5) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 100 iterations of AdaGrad by sampling a subset of $s = 100$ data points per iteration. Set $\text{diag}(G^{(0)}) = (0, 0)^T$ and $\epsilon = 1e - 8$. Tune the learning rate a of equation (9) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 100 iterations of RMSProp by sampling a subset of $s = 100$ data points per iteration. Set $\mathbf{v}^{(0)} = (0, 0)^T$ and $\epsilon = 1e - 8$. Tune the learning rate a of equation (12) and memory factor c of equation (11) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 100 iterations of Adam by sampling a subset of $s = 100$ data points per iteration. Set $\mathbf{m}^{(0)} = (0, 0)^T$, $\mathbf{v}^{(0)} = (0, 0)^T$ and $\epsilon = 1e-8$. Tune the learning rate a of equation (17), memory factor b of equation (13) and memory factor c of equation (14) to converge towards the regression coefficient estimates obtained via `lm()`.

Your code in `question1.r` will store numerical summaries in CSV files and visual summaries in PDF files as follows:

(a) Save the design matrix `X` in the comma-separated value file `answer1a.csv`.

(b) Save the parameter estimates obtained via `lm()` and the final parameter estimates at the last iteration of each optimizer in the comma-separated value file `answer1b.csv`. The parameter estimates saved in `answer1b.csv` correspond to a $2 \cdot 8$ matrix, in which the first and second row represent the respective regression coefficient estimates θ_0 and θ_1, while the eight columns represent the parameter estimates obtained via lm(), GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam. R code in the form of comments has been provided in `question1.r` as guidance to facilitate the process of saving all parameter estimates.

(c) Save the value of the cost function at the last itration of each optimizer in the comma-separated value file `answer1c.csv`. The cost function values saved in `answer1c.csv` correspond to a $1 \cdot 7$ matrix, in which the the seven columns represent the parameter estimates obtained via GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam. R code in the form of comments has been provided in `question1.r` as guidance to facilitate the process of saving all cost function values.

(d) Plot the cost function of NAGSGD and of AdaGrad against the number of iterations in a single plot. Ensure that it is possible to distinguish between the overlaid cost function values of NAGSGD and of AdaGrad. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer1d.pdf`. R code in the form of comments has been provided in `question1.r` as guidance to facilitate the process of saving the plot of cost function values.

(e) Plot the successive iterations of parameter θ_0 against the successive iterations of parameter θ_1 for GD and RMSProp. Moreover, add a point in the plot indicating the parameter estimate obtained via `lm()`. All optimizers started from the same initial parameter value $\boldsymbol{\theta}^{(0)} = (7, -8)^T$, so the trajectories of GD and RMSProp must have the same starting point in the plot. Moreover, the end points of the trajectories of GD and RMSProp must be in near proximity to the single point representing the parameter estimate obtained via `lm()` if the optimizers were tuned appropriately. Ensure that it is possible to distinguish between the overlaid trajectories of GD and RMSProp. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer1e.pdf`. R code in the form of comments has been provided in `question1.r` as guidance to facilitate the process of saving the plot of parameter trajectories.

Upload only the R scripts `cost.r`, `optimizers.r` and `question1.r`. Make sure that the R command `source("question1.r")` works free from bugs and that it generates the output files `answer1a.csv`, `answer1b.csv`, `answer1c.csv`, `answer1d.pdf` and `answer1e.pdf`. However, do not upload the output files themselves. I will assess your work by running the R command `source("question1.r")` and by looking into the generated output files.

Question 2

The goal of question 2 is to run the stochastic optimizers of this project to fit multiple linear regression on a real dataset of relatively small size. Given that you have already developed the R code for stochastic optimization to tackle question 1, the only additional work required for tackling question 2 is to tune the hyperparameters of the optimizers.

The files `cost.r` and `optimizers.r` should be available after having answered question 1. Create an R script with name `question2.r` to include in it your R code for running the optimizers and simulations of question 2. You will not provide any output files or a report as part of your project submission.

You need to make sure before submission that the R command `source("question2.r")` runs successfully and stores the required results in output files as explained further below.

Similarly to `question1.r`, the file `question2.r` will start with the lines

```
source("optimizers.r")
source("cost.r")
```

to load the R code for the cost function for linear regression and its gradient as well as the R code for the optimizers. All subsequent R code for answering question 2 should be saved in `question2.r`.

Start by loading the data file `weather_data.csv`, whose first line is the header containing column labels. For the purposes of this project, it suffices to know that the column labelled `apparent_temperature` is the output variable, while the columns labelled `temperature`, `humidity`, `wind_speed`, `wind_bearing` and `pressure` will be used as covariates in the multiple linear regression model.

The dataset `weather_data.csv` can be found on Kaggle at

`https://www.kaggle.com/budincsevity/szeged-weather`

There is no need to use Kaggle in order to address question 2. If you decide to look up the dataset on Kaggle, keep in mind that the original column labels have been altered to make R coding more user-friendly.

Firstly, standardize all five covariates and create the design matrix `X` including a first column of ones for intercept.

Fit a multiple linear regression to the data using the standardized covariates and the `lm()` function in R. The model will have six parameters, namely five regression coefficient corresponding to the covariates and an intercept.

Use the same initial value $\boldsymbol{\theta}^{(0)} = (-5, -3, 4, 1, 10, -9)^T$ for all optimizers.

Run 70 iterations of GD tuning the learning rate a of equation (1) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 200 iterations of SGD by sampling a subset of $s = 1000$ data points per iteration. Tune the learning rate a of equation (2) to converge towards the regression coefficient estimates obtained via `lm()`. Notice that even for such a simple example, SGD obtains an approximation to the regression coefficient estimates known from `lm()`. Although SGD does not yield an exact solution to the optimization problem, it provides a reasonable approximation.

Run 200 iterations of MSGD by sampling a subset of $s = 1000$ data points per iteration. Set the initial value for momentum to be a vector of zeros, that is set $\mathbf{m}^{(0)} = (0, 0, 0, 0, 0, 0)^T$. Tune the learning rate a of equation (4) and memory factor b of equation (3) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 200 iterations of NAGSGD by sampling a subset of $s = 1000$ data points per iteration. Set the initial value for momentum to be a vector of zeros, that is set $\mathbf{m}^{(0)} = (0, 0, 0, 0, 0, 0)^T$. Tune the learning rate a of equation (6) and memory factor b of equation (5) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 200 iterations of AdaGrad by sampling a subset of $s = 1000$ data points per iteration. Set $\text{diag}(G^{(0)}) = (0, 0, 0, 0, 0, 0)^T$ and $\epsilon = 1e - 8$. Tune the learning rate a of equation (9) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 200 iterations of RMSProp by sampling a subset of $s = 1000$ data points per iteration. Set $\mathbf{v}^{(0)} = (0, 0, 0, 0, 0, 0)^T$ and $\epsilon = 1e-8$. Tune the learning rate a of equation (12) and memory factor c of equation (11) to converge towards the regression coefficient estimates obtained via `lm()`.

Run 200 iterations of Adam by sampling a subset of $s = 1000$ data points per iteration. Set $\mathbf{m}^{(0)} = (0, 0, 0, 0, 0, 0)^T$, $\mathbf{v}^{(0)} = (0, 0, 0, 0, 0, 0)^T$ and $\epsilon = 1e - 8$. Tune the learning rate a of equation (17), memory factor b of equation (13) and memory factor c of equation (14) to converge towards the regression coefficient estimates obtained via `lm()`.

Your code in `question2.r` will store numerical summaries in CSV files and visual summaries in PDF files as follows:

(a) Save the design matrix `X` in the comma-separated value file `answer2a.csv`.

(b) Save the parameter estimates obtained via `lm()` and the final parameter estimates at the last iteration of each optimizer in the comma-separated value file `answer2b.csv`. The parameter estimates saved in `answer2b.csv` correspond to a $6 \cdot 8$ matrix, in which each row represents one of the six regression coefficient estimates, while the eight columns represent the parameter estimates obtained via lm(), GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam.

(c) Save the value of the cost function at the last itration of each optimizer in the comma-separated value file `answer2c.csv`. The cost function values saved in `answer2c.csv` correspond to a $1 \cdot 7$ matrix, in which the the seven columns represent the parameter estimates obtained via GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam.

(d) Plot the cost function of NAGSGD and of AdaGrad against the number of iterations in a single plot. Ensure that it is possible to distinguish between the overlaid cost function values of NAGSGD and of AdaGrad. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer2d.pdf`.

(e) Plot the successive iterations of parameter θ_2 against the successive iterations of parameter θ_3 for GD, MSGD and AdaGrad, recalling that indexing notation for parameters has been set to start from zero for the intercept θ_0. Moreover, add a point in the plot indicating the parameter estimate (θ_2, θ_3) obtained via `lm()`. All optimizers started from the same initial parameter value $\boldsymbol{\theta}^{(0)} = (-5, -3, 4, 1, 10, -9)^T$, so the trajectories of GD, MSGD and AdaGrad must have the same starting point in the plot. Moreover, the end points of the trajectories of GD, MSGD and AdaGrad must be in near proximity to the single point representing the parameter estimate obtained via `lm()` if the optimizers were tuned appropriately. Ensure that it is possible to distinguish between the overlaid trajectories of GD, MSGD and AdaGrad. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer2e.pdf`.

Upload only the R scripts `cost.r`, `optimizers.r` and `question2.r`. Make sure that the R command `source("question2.r")` works free from bugs and that it generates the output files `answer2a.csv`, `answer2b.csv`, `answer2c.csv`, `answer2d.pdf` and `answer2e.pdf`. However, do not upload the output files themselves. I will assess your work by running the R command `source("question2.r")` and by looking into the generated output files.

Question 3 (not assessed)

Question 3 will not be assessed, it is provided for educational purposes. If you do not tackle question 3, you will not be penalized. However, it is advised that you try to answer question 3 in order to experience the computational advantage of stochastic optimization over optimization in the case of big datasets.

The goal of question 3 is to run the stochastic optimizers of this project to fit multiple linear regression on a relativey big dataset. Given that you have already developed the R code for stochastic optimization to tackle question 1, the only additional work required for tackling question 3 is to handle data processing efficiently and to tune the hyperparameters of the optimizers.

The dataset `uci_data.csv` used in question 3 is the same subset of the million song data set used in lab 1. It contains $515,345$ songs (file rows) and 91 columns. The first column is the year of release, ranging from 1922 to 2011, which plays the role of output variable. The remaining 90 columns are continuous predictors of the year of release. `uci_data.csv` is included in the `template` folder. More information about this subset of the million song data set is available at

`http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD`

You will observe the computational cost of fine-tuning optimization given a relatively big dataset. At the same time, you will see how stochastic optimization reduces runtime dramatically in comparison to gradient descent. Moreover, you will see that stochastic optimization attains reasonable parameter estimates in a problem somewhat less trivial in terms of data space and parameter space dimensionality ($n_d = 515,345$ and $n_\theta = 91$).

The files `cost.r` and `optimizers.r` should be available after having answered question 1. Create an R script with name `question3.r` to include in it your R code for running the optimizers and simulations of question 3.

The R command `source("question3.r")` will store the results in output files as explained further below. The anticipated runtime of the R command `source("question3.r")` is not more than fifteen minutes on a computer at one of the labs of the Mathematics and Statistics building.

Similarly to `question1.r`, the file `question3.r` will start with the lines

```
source("optimizers.r")
source("cost.r")
```

to load the R code for the cost function for linear regression and its gradient as well as the R code for the optimizers. All subsequent R code for answering question 3 can be saved in `question3.r`.

Start by loading the data file `uci_data.csv` using the `fread()` function of the `data.table` package in R. Do not use the `read.csv()` function, which is prohibitively slow for the needs of question 3. The dataset `uci_data.csv` does not contain a header.

Firstly, standardize all 90 covariates and create the design matrix `X` including a first column of ones for intercept.

Fit a multiple linear regression to the data using the standardized covariates and the `lm()` function in R. The model will have 91 parameters, namely 90 regression coefficient corresponding to the covariates and an intercept.

Randomly sample the initial value $\boldsymbol{\theta}^{(0)}$ of the parameters by running the R command

```
theta0 <- rnorm(npars, mean=0, sd=10)
```

Use the same initial value $\boldsymbol{\theta}^{(0)}$ for all optimizers.

Run $1,000$ iterations of GD tuning the learning rate a of equation (1) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $50,000$ iterations of SGD by sampling a subset of $s = 500$ data points per iteration. Tune the learning rate a of equation (2) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $20,000$ iterations of MSGD by sampling a subset of $s = 500$ data points per iteration. Set the initial value $\mathbf{m}^{(0)}$ for momentum to be a vector of zeros. Tune the learning rate a of equation (4) and memory factor b of equation (3) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $20,000$ iterations of NAGSGD by sampling a subset of $s = 500$ data points per iteration. Set the initial value $\mathbf{m}^{(0)}$ for momentum to be a vector of zeros. Tune the learning rate a of equation (6) and memory factor b of equation (5) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $20,000$ iterations of AdaGrad by sampling a subset of $s = 500$ data points per iteration. Set $\text{diag}(G^{(0)})$ to be a vector of zeros and $\epsilon = 1e-8$. Tune the learning rate a of equation (9) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $20,000$ iterations of RMSProp by sampling a subset of $s = 500$ data points per iteration. Set $\mathbf{v}^{(0)}$ to be a vector of zeros and $\epsilon = 1e-8$. Tune the learning rate a of equation (12) and memory factor c of equation (11) to converge towards the regression coefficient estimates obtained via `lm()`.

Run $20,000$ iterations of Adam by sampling a subset of $s = 500$ data points per iteration. Set each of $\mathbf{m}^{(0)}$ and $\mathbf{v}^{(0)}$ to be a vector of zeros and $\epsilon = 1e-8$. Tune the learning rate a of equation (17), memory factor b of equation (13) and memory factor c of equation (14) to converge towards the regression coefficient estimates obtained via `lm()`.

Your code in `question3.r` will store numerical summaries in CSV files and visual summaries in PDF files as follows:

(a) Save the design matrix `X` in the comma-separated value file `answer3a.csv`.

(b) Save the parameter estimates obtained via `lm()` and the final parameter estimates at the last iteration of each optimizer in the comma-separated value file `answer3b.csv`. The parameter estimates saved in `answer3b.csv` correspond to a $91 \cdot 8$ matrix, in which each row represents one of the regression coefficient estimates, while the eight columns represent the parameter estimates obtained via lm(), GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam.

(c) Save the value of the cost function at the last itration of each optimizer in the comma-separated value file `answer3c.csv`. The cost function values saved in `answer3c.csv` correspond to a $1 \cdot 7$ matrix, in which the the seven columns represent the parameter estimates obtained via GD, SGD, MSGD, NAGSGD, AdaGrad, RMSProp and Adam.

(d) Plot the cost function of RMSProp against the number of iterations. Save the plot in the PDF file `answer3d.pdf`.

(e) Plot the cost function of RMSProp for iterations between $15,000$ and $20,000$ against the number of iterations. Save the plot in the PDF file `answer3e.pdf`. Confine the vertical axis of the plot in `answer3e.pdf` in a narrower range than the vertical axis of the plot in `answer3d.pdf` so that it the former plot is visibly less smooth and more volatile than the latter plot.

(f) Plot the cumulative mean of the cost function of RMSProp against the number of iterations. Save the plot in the PDF file `answer3f.pdf`.

(g) Plot the successive iterations of parameter θ_0 against the successive iterations of parameter θ_1 for GD, MSGD and RMSProp, recalling that indexing notation for parameters has been set to start from zero for the intercept θ_0. Moreover, add a point in the plot indicating the parameter estimate (θ_0, θ_1) obtained via `lm()`. All optimizers started from the same initial parameter value $\boldsymbol{\theta}^{(0)}$, so the trajectories of GD, MSGD and RMSProp must have the same starting point in the plot. Moreover, the end points of the trajectories of GD, MSGD and RMSProp must be in near proximity to the single point representing the parameter estimate obtained via `lm()` if the optimizers were tuned appropriately. Ensure that it is possible to distinguish between the overlaid trajectories of GD, MSGD and RMSProp. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer3g.pdf`.

(h) Plot the successive iterations of parameter θ_0 against the successive iterations of parameter θ_9 for GD, NAGSGD and AdaGrad, recalling that indexing notation for parameters has been set to start from zero for the intercept θ_0. Moreover, add a point in the plot indicating the parameter estimate (θ_0, θ_9) obtained via `lm()`. All optimizers started from the same initial parameter value $\boldsymbol{\theta}^{(0)}$, so the trajectories of GD, NAGSGD and AdaGrad must have the same starting point in the plot. Moreover, the end points of the trajectories of GD, NAGSGD and AdaGrad must be in near proximity to the single point representing the parameter estimate obtained via `lm()` if the optimizers were tuned appropriately. Ensure that it is possible to distinguish between the overlaid trajectories of GD, NAGSGD and AdaGrad. Add a legend to the plot to make such a distinction clearer. Save the plot in the PDF file `answer3h.pdf`.

(i) Record the runtime of each simulation for each optimizer using the `system.time()` function in R. `system.time()` returns three different times labelled as `user`, `system` and `elapsed`. Make use of times labelled as `user` to answer this sub-question. Create a $7 \cdot 2$ table, in which every row corresponds to one of the seven optimizers, while the first column provides the runtime of each optimizer. The second column will provide the relative speed-up of each optimizer, defined as the ratio of the runtime of the optimizer divided by the runtime of GD. Save this $7 \cdot 2$ table in the comma-separated value file `answer3i.csv`.

The R command `source("question3.r")` will generate the output files `answer3a.csv`, `answer3b.csv`, `answer3c.csv`, `answer3d.pdf`, `answer3e.pdf`, `answer3f.pdf`, `answer3g.pdf` and `answer3i.csv`. Do not upload the output files themselves. Moreover, you do not need to upload `source("question3.r")` as part of your submission, since question 3 will not be assessed.

Project submission

Your answers must be uploaded on the relevant section of the Moodle page for Big Data Analytics by 4:00pm on Wednesday 6 February 2019. You will upload a single folder using your matriculation number as the name of the folder. The folder will contain the files `question1.r`, `question2.r`, `optimizers.r`, `cost.r` and `declaration.pdf`.

Do not upload any output files or any files other than the ones required.

I will assess your work by running the following sequence of commands:

```
source("question1.r)
source("question2.r)
```

Make sure that these commands work before submitting your answers. If any of these commands do not run or if they do not generate the required output files as explained in the project description, marks will be deducted.

The declaration of originality form `declaration.pdf` is available in the `template` folder. Make sure that you sign `declaration.pdf` before uploading it.

Make sure that you submit your work on Moodle before the deadline. Late submissions will receive a penalty of 10%.