COMP9417 - Machine Learning
Homew[illegible]arized Regression & Numerical [illegible]ptimization

Introduction In this homework we will explore some algorithms for *gradient* based optimization. These algorithms have been crucial to the development of machine learning in the last few decades. The most famous example is the backpropagation algorithm used in deep learning, which is in fact just an application of a simple algorithm known as (stochastic) gradient descent. We will first implement gradient descent from scratch on a deterministic problem (no data), and then extend our implementation to solve a real world regression problem.

Points Allocation There are a total of 28 marks.

- Question 1 a): 2 marks
- Question 1 b): 1 mark
- Question 1 c): 1 mark
- Question 1 d): 2 marks
- Question 1 e): 2 marks
- Question 1 f): 4 marks
- Question 1 g): 3 marks
- Question 1 h): 1 mark
- Question 1 i): 3 marks
- Question 1 j): 4 marks
- Question 2 a): 2 marks
- Question 2 b): 1 mark
- Question 2 c): 2 marks

What to Submit

- A **single PDF** file which contains solutions to each question. For each question, provide your solution in the form of text and requested plots. For some questions you will be requested to provide screen shots of code used to generate your answer — only include these when they are explicitly asked for.

- **.py file(s) containing all code you used for the project, which should be provided in a separate .zip file.** This code must match the code provided in the report.

- You may be deducted points for not following these instructions.

- You may be deducted points for poorly presented/formatted work. Please be neat and make your solutions clear. S[illegible] new page if necessary.

- You **cannot** sub[illegible] this will receive a mark of zero. This does not stop you from developing you[illegible]d then copying it into a .py file though, or using a tool such as **nbconvert** or si[illegible]

- We will set up a [illegible]tions about this homework. Please read the existing questions before posting r[illegible]o some basic research online before posting questions. Please only post clarifi[illegible] questions deemed to be *fishing* for answers will be ignored and/or deleted.

- Please check Moodle announcements for updates to this spec. It is your responsibility to check for announcements about the spec.

- Please complete your homework on your own, do not discuss your solution with other people in the course. General discussion of the problems is fine, but you must write out your own solution and acknowledge if you discussed any of the problems in your submission (including their name(s) and zID).

- As usual, we monitor all online forums such as Chegg, StackExchange, etc. Posting homework questions on these site is equivalent to plagiarism and will result in a case of academic misconduct.

- You may **not** use SymPy or any other symbolic programming toolkits to answer the derivation questions. This will result in an automatic grade of zero for the relevant question. You must do the derivations manually.

When and Where to Submit

- Due date: Week 4, Monday **June 19th**, 2023 by **5pm**. Please note that the forum will not be actively monitored on weekends.

- Late submissions will incur a penalty of 5% per day **from the maximum achievable grade**. For example, if you achieve a grade of 80/100 but you submitted 3 days late, then your final grade will be $80 - 3 \times 5 = 65$. Submissions that are more than 5 days late will receive a mark of zero.

- Submission must be done through Moodle, no exceptions.

Question 1. Gradient Based Optimization

The general framework for a gradient method for finding a minimizer of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x_k), \qquad k = 0, 1, 2, \ldots, \tag{1}$$

where $\alpha_k > 0$ is [illegible], or learning rate. Consider the following simple example of minimizing the fu[illegible]1. We first note that $g'(x) = 3x^2(x^3 + 1)^{-1/2}$. We then need to choose a starting [illegible] Let's also take the step size to be constant, $\alpha_k = \alpha = 0.1$. Then we have the follo[illegible]

$$\ldots^{(0)})^2((x^{(0)})^3 + 1)^{-1/2} = 0.7878679656440357$$

$$\ldots^{(1)})^2((x^{(1)})^3 + 1)^{-1/2} = 0.6352617090300827$$

$$x^{(3)} = 0.527[illegible]7477$$

$$\vdots$$

and this continues until we terminate the algorithm (as a quick exercise for your own benefit, code this up and compare it to the true minimum of the function which is $x_* = -1$). This idea works for functions that have vector valued inputs, which is often the case in machine learning. For example, when we minimize a loss function we do so with respect to a weight vector, β. When we take the step-size to be constant at each iteration, this algorithm is known as gradient descent. For the entirety of this question, **do not use any existing implementations of gradient methods, doing so will result in an automatic mark of zero for the entire question.**

(a) Consider the following optimisation problem:

$$\min_{x \in \mathbb{R}^n} f(x),$$

where

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{\gamma}{2}\|x\|_2^2,$$

and where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ are defined as

$$A = \begin{bmatrix} 1 & 2 & 1 & -1 \\ -1 & 1 & 0 & 2 \\ 0 & -1 & -2 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 3 \\ 2 \\ -2 \end{bmatrix},$$

and γ is a positive constant. Run gradient descent on f using a step size of $\alpha = 0.1$ and $\gamma = 0.2$ and starting point of $x^{(0)} = (1, 1, 1, 1)$. You will need to terminate the algorithm when the following condition is met: $\|\nabla f(x^{(k)})\|_2 < 0.001$. In your answer, clearly write down the version of the gradient steps (1) for this problem. Also, print out the first 5 and last 5 values of $x^{(k)}$, clearly indicating the value of k, in the form:

$$\begin{aligned} k &= 0, \qquad x^{(k)} = [1, 1, 1, 1] \\ k &= 1, \qquad x^{(k)} = \cdots \\ k &= 2, \qquad x^{(k)} = \cdots \\ &\vdots \end{aligned}$$

What to submit: an equation outlining the explicit gradient update, a print out of the first 5 ($k = 5$ inclusive) and last 5 rows of your iterations. Use the round function to round your numbers to [illegible] decimal places. Include a screen shot of any code used for this section and a copy of your python code in solutions.py.

(b) In the previous part, we used the termination condition $\|\nabla f(x^{(k)})\|_2 < 0.001$. What do you think this condition means in terms of convergence of the algorithm to a minimizer of f ? How would making the r[illegible] (say 0.0001) instead, change the output of the algorithm? Explain.

What to subm[illegible]

In the next fe[illegible]dient methods explored above to solve a real machine learning problem. Co[illegible]ta provided in `CarSeats.csv`. It contains 400 observations with each ob[illegible]ild car seats for sale at one of 400 stores. The features in the data set are o[illegible]

- Sales: Unit sales (in thousands) at each location
- CompPrice: Price charged by competitor at each location
- Income: Local income level (in thousands of dollars)
- Advertising: advertising budget (in thousands of dollars)
- Population: local population size (in thousands)
- Price: price charged by store at each site
- ShelveLoc: A categorical variable with Bad, Good and Medium describing the quality of the shelf location of the car seat
- Age: Average age of the local population
- Education: Education level at each location
- Urban A categorical variable with levels No and Yes to describe whether the store is in an urban location or in a rural one
- US: A categorical variable with levels No and Yes to describe whether the store is in the US or not.

The target variable is Sales. The goal is to learn to predict the amount of Sales as a function of a subset of the above features. We will do so by running Ridge Regression (Ridge) which is defined as follows

$$\hat{\beta}_{\text{Ridge}} = \arg\min_{\beta} \frac{1}{n}\|y - X\beta\|_2^2 + \phi\|\beta\|_2^2,$$

where $\beta \in \mathbb{R}^p$, $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$ and $\phi > 0$.

(c) We first need to preprocess the data. Remove all categorical features. Then use `sklearn.preprocessing.StandardScaler` to standardize the remaining features. Print out the mean and variance of each of the standardized features. Next, center the target variable (subtract its mean). Finally, create a training set from the first half of the resulting dataset, and a test set from the remaining half and call these objects X_train, X_test, Y_train and Y_test. Print out the first and last rows of each of these.

What to submit: a print out of the means and variances of features, a print out of the first and last rows of the 4 requested objects, and some commentary. Include a screen shot of any code used for this section and a copy of your python code in solutions.py.

(d) It should be obvious that a closed form expression for $\hat{\beta}_{\text{Ridge}}$ exists. Write down the closed form expression, and compute the exact numerical value on the training dataset with $\phi = 0.5$.

What to submit: Your working, and a print out of the value of the ridge solution based on (X_train, Y_train). Include a screenshot of any code used for this section and a copy of your python code in solutions.py.

We will now solve the ridge problem but using numerical techniques. As noted in the lectures, there are a few variants of gradient descent that we will briefly outline here. Recall that in gradient descent our update rule is

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla L(\beta^{(k)}), \qquad k = 0, 1, 2, \dots,$$

where $L(\beta)$ i[illegible] we are trying to minimize. In machine learning, it is often the case that the [illegible] form

$$L(\beta) = \frac{1}{n} \sum_{i=1}^{n} L_i(\beta),$$

i.e. the loss is an average of n functions that we have lablled L_i. It then follows that the gradient is also an average of the form

$$\nabla L(\beta) = \frac{1}{n} \sum_{i=1}^{n} \nabla L_i(\beta).$$

We can now define some popular variants of gradient descent .

(i) Gradient Descent (GD) (also referred to as batch gradient descent): here we use the full gradient, as in we take the average over all n terms, so our update rule is:

$$\beta^{(k+1)} = \beta^{(k)} - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla L_i(\beta^{(k)}), \qquad k = 0, 1, 2, \dots.$$

(ii) Stochastic Gradient Descent (SGD): instead of considering all n terms, at the k-th step we choose an index i_k randomly from $\{1, \dots, n\}$, and update

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla L_{i_k}(\beta^{(k)}), \qquad k = 0, 1, 2, \dots.$$

Here, we are approximating the full gradient $\nabla L(\beta)$ using $\nabla L_{i_k}(\beta)$.

(iii) Mini-Batch Gradient Descent: GD (using all terms) and SGD (using a single term) represents the two possible extremes. In mini-batch GD we choose batches of size $1 < B < n$ randomly at each step, call their indices $\{i_{k_1}, i_{k_2}, \dots, i_{k_B}\}$, and then we update

$$\beta^{(k+1)} = \beta^{(k)} - \frac{\alpha_k}{B} \sum_{j=1}^{B} \nabla L_{i_j}(\beta^{(k)}), \qquad k = 0, 1, 2, \dots,$$

so we are still approximating the full gradient but using more than a single element as is done in SGD.

(e) The ridge regression loss is

$$L(\beta) = \frac{1}{n} \|y - X\beta\|_2^2 + \phi \|\beta\|_2^2.$$

Show that we can write

$$L(\beta) = \frac{1}{n}\sum_{i=1}^{n} L_i(\beta),$$

and identify [illegible] $, L_n(\beta)$. Further, compute the gradients $\nabla L_1(\beta), \ldots, \nabla L_n(\beta)$
What to subm [illegible]

(f) In this questi [illegible] (batch) GD from scratch to solve the ridge regression problem. Use an initia [illegible] e vector of ones), and $\phi = 0.5$ and run the algorithm for 1000 epochs (an ep [illegible] e entire data, so a single GD step). Repeat this for the following step sizes:

$$\ldots 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$$

To monitor the performance of the algorithm, we will plot the value

$$\Delta^{(k)} = L(\beta^{(k)}) - L(\hat{\beta}),$$

where $\hat{\beta}$ is the true (closed form) ridge solution derived earlier. Present your results in a 3×3 grid plot, with each subplot showing the progression of $\Delta^{(k)}$ when running GD with a specific step-size. State which step-size you think is best and let $\beta^{(K)}$ denote the estimator achieved when running GD with that choice of step size. Report the following:

(i) The train MSE: $\frac{1}{n}\|y_{\text{train}} - X_{\text{train}}\beta^{(K)}\|_2^2$

(ii) The test MSE: $\frac{1}{n}\|y_{\text{test}} - X_{\text{test}}\beta^{(K)}\|_2^2$

What to submit: a single plot, the train and test MSE requested. Include a screen shot of any code used for this section and a copy of your python code in solutions.py.

(g) We will now implement SGD from scratch to solve the ridge regression problem. Use an initial estimate $\beta^{(0)} = 1_p$ (the vector of ones) and $\phi = 0.5$ and run the algorithm for 5 epochs (this means a total of $5n$ updates of β, where n is the size of the training set). Repeat this for the following step sizes:

$$\alpha \in \{0.000001, 0.000005, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.006, 0.02\}$$

Present an analogous 3×3 grid plot as in the previous question. Instead of choosing an index randomly at each step of SGD, we will cycle through the observations in the order they are stored in X_train to ensure consistent results. Report the best step-size choice and the corresponding train and test MSEs. In some cases you might observe that the value of $\Delta^{(k)}$ jumps up and down, and this is not something you would have seen using batch GD. Why do you think this might be happening?

What to submit: a single plot, the train and test MSE requested and some commentary. Include a screen shot of any code used for this section and a copy of your python code in solutions.py.

(h) Based on your GD and SGD results, which algorithm do you prefer? When is it a better idea to use GD? When is it a better idea to use SGD? *What to submit: some commentary*

(i) Note that in GD, SGD and mini-batch GD, we always update the entire p-dimensional vector β at each iteration. An alternative popular approach is to update each of the p parameters individually.

To make this idea more clear, we write the ridge loss $L(\beta)$ as $L(\beta_1, \beta_2, \dots, \beta_p)$. We initialize $\beta^{(0)}$, and then solve for $k = 1, 2, 3, \dots,$

$$\beta_1^{(k)} = \arg\min_{\beta_1} L(\beta_1, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \dots, \beta_p^{(k-1)})$$

$$\beta_2^{(k)} = \arg\min_{\beta_2} L(\beta_1^{(k)}, \beta_2, \beta_3^{(k-1)}, \dots, \beta_p^{(k-1)})$$

$$\vdots$$

$$\beta_p^{(k)} = \arg\min_{\beta_p} L(\beta_1^{(k)}, \beta_2^{(k)}, \beta_3^{(k)}, \dots, \beta_p).$$

Note that eac[illegible] is over a single (1-dimensional) coordinate of β, and also that as as soon as [illegible]e the new value when solving the update for $\beta_{j+1}^{(k)}$ and so on. The idea is then to cycle through these coordinate level updates until convergence. In the next two parts we will implement this algorithm from scratch for the Ridge regression problem:

$$L(\beta) = \frac{1}{n}\|y - X\beta\|_2^2 + \phi\|\beta\|_2^2$$

Note that we can write the $n \times p$ matrix $X = [X_1, \dots, X_p]$, where X_j is the j-th column of X. Find the solution of the optimization

$$\hat{\beta}_1 = \arg\min_{\beta_1} L(\beta_1, \beta_2, \dots, \beta_p).$$

Based on this, derive similar expressions for $\hat{\beta}_j$ for $j = 2, 3, \dots, p$.

Hint: Note the expansion: $X\beta = X_j\beta_j + X_{-j}\beta_{-j}$, where X_{-j} denotes the matrix X but with the j-th column removed, and similarly β_{-j} is the vector β with the j-th coordinate removed. *What to submit:* your working out.

(j) Implement the algorithm outlined in the previous question on the training dataset. In your implementation, be sure to update the $\hat{\beta}_j$'s in order and use an initial estimate of $\beta^{(0)} = 1_p$ (th vector of ones), and $\phi = 0.5$. Terminate the algorithm after 10 cycles (one cycle here is p updates, one for each β_j), so you will have a total of $10p$ updates. Report the train and test MSE of your resulting model. Here we would like to compare the three algorithms: new algorithm to batch GD and SGD from your previous answers with optimally chosen step sizes. Create a plot of k vs. $\Delta^{(k)}$ as before, but this time plot the progression of the three algorithms. Be sure to use the same colors as indicated here in your plot, and add a legend that labels each series clearly. For your batch GD and SGD include the step-size in the legend. Your x-axis only needs to range from $k = 1, \dots 10p$. Further, report both train and test MSE for your new algorithm. *Note: Some of you may be concerned that we are comparing one step of GD to one step of SGD and the new aglorithm, we will ignore this technicality for the time being. What to submit: a single plot, the train and test MSE requested.*

Question 2

Given $\lambda > 0$ and $v \in \mathbb{R}$, consider the following optimization problem:

$$\min_{\beta \in \mathbb{R}} \left\{ |\beta| + \frac{1}{2\lambda}(\beta - v)^2 \right\}.$$

(a) Denote the solution to the above problem by $\hat{\beta}$. Write down an expression for $\hat{\beta}$. Your answer should be of the form:

$$\hat{\beta} = \mathcal{T}_\lambda(v) := \begin{cases} ? & \text{if} \quad v > \lambda, \\ ? & \text{if} \quad |v| \leq \lambda, \\ ? & \text{if} \quad v < -\lambda. \end{cases}$$

What to subm[illegible] You must include all working out to receive credit.

(b) Using the ab[illegible] for any $\lambda > 0$ and $v = (v_1, \ldots, v_p) \in \mathbb{R}^p$, the solution of the minimizatior[illegible]

$$\min_{\beta \in \mathbb{R}^p} \left\{ \|\beta\|_1 + \frac{1}{2\lambda} \|\beta - v\|_2^2 \right\}$$

is

$$\hat{\beta} = \mathcal{T}_\lambda(v) := (\mathcal{T}_\lambda(v_1), \mathcal{T}_\lambda(v_2), \ldots, \mathcal{T}_\lambda(v_p)).$$

What to submit: your working out.

(c) Let $v = (1, 2, 4, -7, 2, 4, -1, 8, 4, -10, -5)$. What are the results for $\mathcal{T}_\lambda(v)$ with $\lambda = 1, 3, 6, 9$? What do you observe? *What to submit: your results and some commentary*