COL 774: Machine Learning. Assignment 1

Due Date: 11:59 pm, Friday, Feb 14, 2025. Total Points: 44 (for Q1 and Q2) + _

Notes:

- You should submit all your code as well as any graphs that you might plot.
- Submit all the code in .py files and all answers to theory questions; all graphs should be submitted in a PDF. Do not submit any thoery in the code files.
- Do not submit the datasets.
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use Python for all your programming solutions.
- Your code should have appropriate documentation for readability. This will influence your marks. Demo will also be conducted in which you will have to explain your code
- You will be graded based on what you have submitted as well as your ability to explain your code.
- The readability and presentation in your report will also carry marks. While there is no specific format, you are advised to be clear, concise and complete in your answers with explanations wherever required
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- Your assignment will be run on autograder scripts. Detailed instructions regarding this and submission guidelines of code will follow soon. Till then, please start working on writing the functions for your code.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, an addtional penalty of the negative of the total weightage of the assignment and possibly much stricter penalties (including a **fail grade** and/or referring to a **DisCo**). Please note that using any **AI tools is not allowed**. If the software detects any plagiarism due to it, this will be penalised equivalently.
- Many of the problems below have been adapted from the Machine Learning course offered by Andrew Ng at Stanford.
- For all the parts, the allowed libraries are pandas, numpy, matplotlib, argparse, and sys, along with any other library specifically for plotting. Please note that sklearn or scikit-learn is not allowed. If there are any confusion on whether a library is allowed, feel free to ask on Piazza. We may penalize submission which uses a library outside the above permitted libraries and those approved on piazza.

1 (20 points) Linear Regression

In this problem, we will implement least squares linear regression to predict density of wine based on its acidity. Recall that the error metric for least squares is given by:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)^2$$

where $h_\theta(x) = \theta^T x$ and all the symbols are as discussed in the class. The files linearX.csv and linearY.csv contain the feature ($x^{(i)}$'s, $x^{(i)} \in \mathcal{R}$) and its label ($y^{(i)}$'s, $y^{(i)} \in \mathcal{R}$), respectively, with one training example per row. We will implement least squares linear regression to learn the relationship between $x^{(i)}$'s and $y^{(i)}$'s. **Do not normalize the input features or their label. The input features are already normalized**

1. (**8 points**)
   (a) Implement batch gradient descent method for optimizing $J(\theta)$.
   (b) Choose (and report) an appropriate learning rate and the stopping criteria (as a function of the change in the value of $J(\theta)$). You can initialize the parameters as $\theta = \vec{0}$ (the vector of all zeros). Do not forget to include the intercept term.
   (c) Report your learning rate, stopping criteria, and the final set of parameters obtained by your algorithm.
2. (**3 points**) Plot the data on a two-dimensional graph and plot the hypothesis function learned by your algorithm in the previous part.
3. (**3 points**)
   (a) Draw a 3-dimensional mesh showing the error function ($J(\theta)$) on z axis and the parameters in the $x - y$ plane. Use a colour gradient for different values of the error function.
   (b) On the same plot, display the error value using the current set of parameters at each iteration of the gradient descent. Include a time gap of 0.2 seconds in your display for each iteration so that the change in the function value can be observed by the human eye.
4. (**3 points**) Repeat the part above for drawing the contours of the error function at each iteration of the gradient descent. Once again, chose a time gap of 0.2 seconds so that the change be perceived by the human eye.(Note here plot will be 2-D)
5. (**3 points**) Repeat the part above (i.e. draw the contours at each learning iteration) for the learning rate values of $\eta = \{0.001, 0.025, 0.1\}$. What do you observe? Comment.

2 (24 points) Sampling, Closed Form and Stochastic Gradient Descent

In this problem, we will introduce the idea of sampling by adding Gaussian noise to the prediction of a hypothesis and generate synthetic training data. Consider a given hypothesis h_θ (*i.e.* known $\theta_0, \theta_1, \theta_2$) for a data point $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$. Note that $x_0 = 1$ is the intercept term.

$$y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Adding Gaussian noise, equation becomes

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$

To gain deeper understanding behind Stochastic Gradient Descent (SGD), we will use the SGD algorithm to learn the original hypothesis from the data generated using sampling, for varying batch sizes. We will implement the version where we make a complete pass through the data in a round robin fashion (meaning you have to run through all batches for each epoch), after initially shuffling the examples. If there are r examples in each batch, then there is a total of $\frac{m}{r}$ batches assuming m training examples. For the batch number b ($1 \leq b \leq \frac{m}{r}$), the set of examples is given as: $\{x^{(i_1)}, x^{(i_2)}, \cdots, x^{(i_r)}\}$ where $i_k = (b-1)r + k$. The Loss function computed over these r examples is given as:

$$J_b(\theta) = \frac{1}{2k} \sum_{k=1}^{r} \left(y^{(i_k)} - h_\theta(x^{(i_k)})\right)^2$$

1. **(4 points)** Sample 1 million data points taking values of $\theta = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}3\\ 1\\ 2\end{bmatrix}$, $x_1 \sim \mathcal{N}(3, 4)$ and $x_2 \sim \mathcal{N}(-1, 4)$ independently, and noise variance in y, $\sigma^2 = 2$. Split this data into two parts - keep 80% of the data (800,000 data points) as train set, and the remaning 20% of the data (200,000 data points) as the training set.

2. **(6 points)**
   (a) Implement Stochastic gradient descent method for optimizing $J(\theta)$.
   (b) Relearn $\theta = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}$ using the training set of part a) keeping everything same except the batch size. Keep $\eta = 0.001$ and initialize $\forall j\ \theta_j = 0$. Report the θ learned each time separately for values of batch size $r = \{1, 80, 8000, 800000\}$. **Do not normalize any part of the data.**
   (c) Carefully decide your convergence criteria in each case. Make sure to go through the material on the convergence of SGD algorithm discussed in class.

3. **(6 points)**
   (a) Do different algorithms in the part above (for varying values of r) converge to the same parameter values? Comment on the relative speed of convergence and also on number of iterations in each case.
   (b) We know that a closed form solution exists for Linear Regression, given by:

   $$\theta = (X^T X)^{-1} X^T Y$$

   Relearn $\theta = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}$ using sampled data points of part a) from this closed form.
   (c) Compare the true parameters (from which we sampled), the parameters learned in closed form, and the parameters learnt by SGD. Write what you observe.

4. **(4 points)** Find the mean squared error on the test set (the 20% portion kept in part a)) called the test error. Compare with the error on the training set called training error.

5. **(4 points)** In the 3 dimensional parameter space(θ_j on each axis), plot the movement of θ as the parameters are updated (until convergence) for varying batch sizes. How does the (shape of) movement compare in each case? Does it make intuitive sense? Argue.

3 (15 points) Logistic Regression

... Coming Soon ....

4 (25 points) Gaussian Discriminant Analysis

... Coming Soon ....

5 Submission Instructions

You can find the starter code along with the data at Assignment1 starter code. The directory structure is given below. Your implementations will be autograded, and thus we request you to follow the following instructions while submitting:

- Note that some files will be added later (on the same link) for question 3 and 4.

- You need to implement the functions mentioned in the starter code files (e.g. in `linear_regression.py`). You may add any other functions or methods, but do not change the signature (name and arguments) of the given functions.
- You may add new files for the analysis (for plotting etc.) for each question in their respective folders. You may use a jupyter notebook or python file - whichever you prefer. Note that any and all code used by you (for plotting etc.) needs to submitted.
- While submitting, make sure you follow the same directory structure.

```
Assignment1/
├── data/
│   ├── Q1/
│   │   ├── linearX.csv
│   │   └── linearY.csv
│   ├── Q3/
│   └── Q4/
├── Q1/
│   └── linear_regression.py
├── Q2/
│   └── sampling_sgd.py
├── Q3/
└── Q4/
```