# COL 341 - Machine Learning - Assignment 1

**Due Date:**
**9:00 pm on Monday, 12$^{th}$ August, 2019 - Part 1**
**9:00 pm on Monday, 19$^{th}$ August, 2019 - Part 2**
**Max Marks : 100**

**Notes:**

- This assignment has two parts - Linear Regression and Logistic regression.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- You are advised to use vector operations (wherever possible) for best performance.
- You should use Python only for all your programming solutions.
- Your assignments will be auto-graded, make sure you test your programs before submitting. We will use your code to train the model on training data and predict on test set. You will need to perform cross validation to figure out regularization parameters during auto-grading as well.
- Input/output format, submission format and other details are included. Your programs should be modular enough to accept specified parameters.
- You should submit work of your own. You should cite the source, if you choose to use any external resource. You will be awarded F grade or **DISCO** in case of plagiarism.
- You can use total of **7** buffer days across all assignments.
- For doubts send an email to Nilaksh Agarwal

1. **Linear Regression - (50 points)**

   Release date: Aug. 5, 2018, Due date: Aug. 12, 2018)

   In this problem, we will use Linear Regression to predict the number of comments on a blog post in 24 hours (from basetime). You have been provided with train and validation split of the **BlogFeedback dataset**. Train Data is given in csv format with target as last value, please refer to Appendix for more details and submission format. For the details of the original dataset, you are encouraged to look at this webpage.

   (a) **(12.5 points)** Given a training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, recall that linear regression optimization problem can be written as:

   $$\min_{w,b} \frac{1}{2n} \sum_{i=1}^{n} ||w^T x^{(i)} + b - y^{(i)}||^2 \tag{1}$$

   Here, $(w, b)$ represents the hyper-plane fitting the data. Implement linear regression on this dataset using normal equations (analytic solution). Note that you may add a column with all ones feature in the matrix $x$ to absorb the parameter $b$ in the weight vector $w$. In this case you may simply use the expression for the Moore-Penrose pseudo inverse covered in the class.

   (b) **(12.5 points)** Implement ridge regression using normal equations to find the optimal hyper-plane. The ridge regression formula is:

   $$\min_{w,b} \frac{1}{2n} \sum_{i=1}^{n} ||w^T x^{(i)} + b - y^{(i)}||^2 + \frac{\lambda}{2} \Big( \sum_{i=1}^{m} ||w_i||^2 + ||b||^2 \Big) \tag{2}$$

   Here, $\lambda$ is the regularization parameter. Again, for simplicity and elegance, you may add an all ones features to all the examples $(x)$ such that the parameter $b$ is absorbed in $w$ which now becomes $m+1$ dimensional vector. You should use 10 fold cross-validation to determine optimal value of regularization parameter. Don't shuffle the data and divide the data sequentially into 10 equal folds.
   Try the following values for $\lambda$ : { 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000 } You are free to experiment with more values if you wish.

   (c) **(25 points)** Feature creation and selection [1] [2] are important part of machine learning. The objective of this part of the assignment is to get the best possible prediction on unseen data by creating additional non-linear features. Extend your data by creating as many features as you like. Then use Lasso regression that will automatically select a small number of features. You will need to select the optimal regularization penalty $\lambda$ for lasso regression that will give the best performance on unseen data. Use cross-validation on the training data for choosing the best regularization parameter $\lambda$. You should try out different transformations to get best performance. The grading for this part will be relative and depend very heavily on the accuracy of your predictions.
   You are encouraged to use 'Lasso model fit with Least Angle Regression (Lars)' package/function for this part. *Note*: LARS package is available for Python. Click here for more details.

   *Evaluation:*

   - For parts (a) and (b), you can get 0 (in case your program given an error), partial marks (if your code runs fine but predictions are incorrect within some predefined threshold) and full (if your code works exactly as expected).
   - For part (c), marks will be based on the prediction errors on the unseen data. There will be relative grading for this part.
   - You can view marks for part[a,b] on Moodle, however, the final grading will be on a different data.
   - Please refer to the submission instructions, any submissions not following the guidelines will fail the auto-grading and be awarded 0. There will be **NO DEMO** for this assignment.

**Extra Readings (highly recommended for part (c)):**

(a) Regression Shrinkage and Selection Via the Lasso

(b) Compressive Sensing Resources

(c) Least Angle Regression

2. **Logistic Regression (50 points)**

Release date: Aug. 5, 2018, Due date: Aug. 19, 2018)

In this problem, we will use Logistic Regression to build a classifier for **Nursery Data Set**. You will be building a logistic regression model to predict the final outcome of the students who applied for nursery admissions to schools in Ljubljana, Slovenia. The final outcome can be one of the following: `not_recom, recommend, very_recom, priority, spec_prior`. You need to minimize the logistic loss function using the gradient descent algorithm. Do not shuffle data within your code, input data will already be shuffled. For details of the original dataset, you are encouraged to look at this webpage.

(a) **(15 points)** Given a training dataset $D = \{(x^{(i)}, t^{(i)})\}_{i=1}^{m}$, recall that the log-likelihood for logistic regression can be written as:

$$L(w,b) = \frac{1}{2m}\sum_{i=1}^{m} t^{(i)} \log(h_{w,b}(x^{(i)})) + (1 - t^{(i)}) \log(1 - h_{w,b}(x^{(i)})) + \frac{\lambda}{2}\Big(\sum_{i=1}^{n} ||w_i||^2 + ||b||^2\Big) \tag{3}$$

$$h_{w,b}(x) = \frac{1}{1 + e^{-w^T x + b}} \tag{4}$$

$(w, b)$ represents the decision surface learned by logistic regression.
Implement gradient descent algorithm and solve the logistic regression problem using it. Learning rate is a critical element in gradient descent based algorithms, you should experiment with following learning rate strategies: i) Constant learning rate, ii) Adaptive learning rate as $\eta_t = \frac{\eta_0}{\sqrt{t}}$, iii) Adaptive learning rate using $\alpha\beta$ backtracking line search algorithm.
Here, $\eta_0$ is the initial learning rate and $t$ is the time step i.e. iteration. Your code should output weights and run for a defined number of iterations.

(b) **(15 points)** Implement mini-batch gradient descent algorithm and solve the logistic regression problem using it. Use batch size as a user input. Again, run it for a defined number of epochs.

(c) **(10 points)** Plot the loss function, L(w,b) with respect to number of floating point operations, varying i) Batch size, ii) Value of learning rate ($\eta_0$ for adaptive case). Also include the confusion matrix, F1 scores for each class as well as the Micro and Macro F1 scores

(d) **(10 points)** Find the best working hyper-parameters and use the model to make predictions on the test set. This part will have a fixed time (1 minute) to train your model, before predicting it on the test set.

*Evaluation:*

- Cross-Entropy Loss will be used as evaluation criterion.
- For part-a, part-b, you can get 0 (error), half (code runs fine but predictions are incorrect within some predefined threshold) and full (works as expected).
- For part-c, marks will be based on the report.
- For part-d, marks will be based on error on test data. There will be relative marking for this part.
- You can view marks for part[a,b] on Moodle, however, the final testing will be on a different test set
- Please refer to the submission instructions, any submissions not following the guidelines will fail the auto-grading and be awarded 0. There will be **NO DEMO** for this assignment.

**Submission Instructions:**

1. Linear Regression

Submit your code in a single executable python file called linear.py
Your code will be run as:

python linear.py *Mode Parameters*

The mode corresponds to part [a-c] of the assignment.
The parameters are dependant on the mode:

*python linear.py a trainfile.csv testfile.csv outputfile.txt weightfile.txt*
Here you have to write the predictions (1 per line) and create a line aligned outputfile.txt. Also output your weights (including intercept) int the weightfile.txt

*python linear.py b trainfile.csv testfile.csv regularization.txt outputfile.txt weightfile.txt*
The parameters are the same as mode 'a', with the additional Regularization Parameter List ($\lambda$) being an input. You have to perform 10-Fold Cross validation for all the $\lambda$ values and report the output and weights as in part 1. Additionally, print the best $\lambda$ value.

*python linear.py c trainfile.csv testfile.csv outputfile.txt*
Here, you should use your best features (found using Lasso) and calculate your predictions on the same. You will be graded on lowest mean-squared error.

2. Logistic Regression

Submit your code in a single executable python file called logistic.py
Your code will be run as:

python logistic.py *Mode Parameters*

The mode should corresponds with part [a-d] of the assignment.
The parameters are dependant on the mode:

*python logistic.py a trainfile.csv testfile.csv param.txt outputfile.txt weightfile.txt*
Here you have to write the predictions (1 per line) and create a line aligned outputfile.txt. Also output your weights (including intercept) int the weightfile.txt. Here, param.txt will contain three lines of input, the first being a number [1-3] indicating which learning rate strategy to use and the second being the fixed learning rate (for "1"), seed value for adaptive learning rate (for "2") or $\beta$ value for $\alpha\beta$ backtracking (assume $\alpha = 0.5$). The third line will be the max number of iterations. You are free to implement an additional stopping criteria as well. Also print the number of iterations your program takes.

*python logistic.py b trainfile.csv testfile.csv param.txt outputfile.txt weightfile.txt*
The arguments mean the same as mode a, with an additional line 4 in param.txt specifying the batch size (int). Again, print the number of iterations.

*python logistic.py c trainfile.csv testfile.csv param.txt*
Here, param.txt will have the same specifications as part-b. Print the confusion matrix, F1 scores (Micro and Macro as well). You are not to use any existing libraries for the same.

*python logistic.py d trainfile.csv weightfile.txt*
Here, you are free to use your own learning rate, batch size etc. The weight file will have one weight per line (with indexes in ascending order) The first line would correspond to the intercept term. Your output would be tested using these weights. For this part, the code would be given 1 minute to train, and then killed, so be sure to save your weights after every iteration/epoch. If you don't output a weightfile you will get a 0.