# COL 341 - Machine Learning - Assignment 1

**Due Date:**
**9:00 pm on Thursday, 9th September, 2021 - Part 1**
**9:00 pm on Sunday, 26th September, 2021 - Part 2**
**Max Marks : 100**

**Notes:**

- **Copyright Claim:** This is a property of Prof. Rahul Garg Indian Institute of Technology Delhi, any replication of this without explicit permissions on any websites/sources will be considered violation to the copyright.
- This assignment has two parts: 1.1 – Linear Regression and 1.2 – Logistic regression.
- For the first part (A1.1), you should submit three files linear.py, feature_selection.py (on moodle) and report.pdf (on Gradescope). The description of these files are in appendix.
- For the second part (A1.2), you should submit three files logistic.py, logistic_feature_selection.py (on moodle) and report.pdf (on Gradescope). The description of these files are in appendix.
- You are advised to use vector operations (wherever possible) for best performance.
- You should use Python 3 only for all your programming solutions.
- Your assignments will be auto-graded on our servers, make sure you test your programs with the provided evaluation script before submitting. We will use your code to train the model on a different training data set and predict on a different test set as well. 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. No additional resource or library is allowed except for the ones specifically mentioned in the assignment statement. If you choose to use any external resource or copy any part of this assignment, you will be awarded D or F grade or **DISCO**.
- *Graceful degradation:* For each late day from the deadline, there will be a penalty of 10%. So, if you submit, say 4 days after deadline, you will be graded out of 60% of weightage of assignment.
- For doubts in Linear Regression send an email to Abhishek Burnwal and for Logistic Regression to Shivansh Chandra Tripathi.

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

Release date: Sept. 2, 2021, Due date: Sept. 9, 2021)

In this problem, we will use Linear Regression to predict the Total Costs of a hospital patient. You have been provided with training dataset (and evaluation scripts) of the **SPARCS Hospital dataset**. Train Data is given as train.csv with target as last value, please refer to Evaluation Appendix for more details, submission format and evaluation on sample test. The final training and evaluation will be done on an unseen dataset. Make sure your output adheres to the format given in Appendix using the evaluation script.

On a side note, You should know that the dataset above is actually a processed subset of this original dataset, which you are encouraged to look at. The original dataset has been pre-processed by us to make it suitable for linear regression. For those interested, the details of pre-processing can be found in Appendix at the end of document.

Make sure your programs run efficiently and do not end up using too much memory of CPU. For parts (a) and (b), your program should run using less than 8GB of RAM within 15 minutes on an *Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz* (see this link for theoretical peak GFLOPS of this processor). For part (b), you are strongly advised to not shuffle the data to avoid mismatch with the model solution.

(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. You may not use any library other than NumPy or SciPy for this part.

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

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

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. Do not
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. You should use modified Pseudo-Inverse expression as discussed in the class for solving this part. You may not use any library other than NumPy or SciPy for this part.

(c) **(25 points)** Feature creation and selection [1] [2] [3] 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. For this part you can use train_large.csv, which contains about 1.6 million instances as your training dataset. Extend your data by creating as many features as you like. Then use Lasso regression that will automatically select a small number (less than 300) 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. Make a 1-2 page report.pdf file stating the number of features in your final selection and also a brief reason for creating those features and why they are relevant to the underlying dataset. Submit this report on gradescope while the code files need to be submitted on moodle. 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.

**Important**: You can (and should) experiment with any number of features on training data. However, while submitting your work, the total number of features in your final selection must be **less than 300** as we are going to do the final evaluation on a bigger data-set. Your submission file *linear.py* when run with option for part (c) should create your *selected features* and then run the Moore-Penrose Pseudo inverse to make the predictions using the given input training and testing data. You program will be run on very large train and test data – so make sure that you write efficient codes and you don't consume too much memory. We will run a reference code to figure out the memory and running time requirements for the evaluation run and will make sure that the your program is given sufficient memory and CPU time to run up to 300 features. However, if your program doesn't complete in the stipulated time and memory, it may cause an early termination and you may get a 0 for this part. Your submission file *feature_selection.py* should contain all your code for feature creation and selection using Lasso (including cross validation). However, when running linear.py, (to save runtime) you are not expected to run feature_selection.py. You may directly code your finally selected features in linear.py.

Some examples of feature generation are (i) One-Hot encoding (ii) using average/median values of the target for all the examples with a particular value of a categorical variable (e.g., *Facility_id* = 1453) (iii) average/median target value for all the examples with two specified categorical values (e.g., *Facility_id* = 1453 and *CCS Diagnosis Code* = 13) and others examples discussed in the class. Application of common sense and innovative thinking about the domain will greatly help in creating meaningful features.

*Evaluation:*

- **Training:** You are given file train.csv and test.csv. You are to use them 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). The final scores for these parts will be calculated on unseen data. Make sure you name your files exactly as given in the command line arguments (submission details are in Appendix). You can check your output format by testing parts a and b on test.csv, generating outputfile and weightfile and then using the evaluation scripts.
- Use the given script evaluate.sh to check output format and to make sure there is no error during auto grading. Run the evaluation script for part a as:

  python3 grade_a.py outputfile_a.txt weightfile_a.txt model_outputfile_a.txt model_weightfile_a.txt

  Similarly for part 2:

  python3 grade_b.py outputfile_b.txt weightfile_b.txt model_outputfile_b.txt model_weightfile_b.txt

  where model_{outputfile, weightfile}_{a,b} have been provided in the folder Assignment_1.zip
- 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: Sept. 9, 2021, Due date: Sept. 26, 2021)

In this problem, we will use Logistic Regression to build a classifier for **Hospital Inpatient Discharges** training data. You will be building a logistic regression model for 8 class classification to predict the 'Length of Stay' (target column in the given dataset) of the patients in the Hospital. The length of Stay can be any of the one among {1,2,3,4,5,6,7,8} days (here label 8 means $\geq$ 8 days). You need to minimize the logistic loss function using the gradient descent algorithm. Use softmax for the multi-class prediction. Initialize all your weights and biases from zero in all parts a,b,c,d. You will also need to convert all the features (except the column 'Total Costs') to a one hot encoding as follows:

**For part (a) and (b)** Use the train.csv and test.csv files (not train_large.csv) and the code in read_and_encode.py for data read and one-hot encoding for non-target columns. Make sure that after this the total number of non-target columns be 1632. Note that this code is given to ensure your outputs match the model outputs, since the test.csv might have some categorical values in non-target columns although very small in number that are never seen in train.csv and this may make the one-hot encoding inconsistent if done separately for train and test files.

**For part (c) and (d)** You're free to use either the given code or anything but make sure that your code runs correctly and does not throws error with one-hot encoding or anything else that you implemented when we evaluate your solutions on our own test file which may have some rare values in non-target columns (excluding 'Total Costs') that were never seen in train.csv or train_large.csv.

The number of unique values in each non-target column (except for column 'Total Costs') with respect to full dataset is given in unique.txt. You may refer to it and incorporate in your part c) and d) solutions if required.

One-hot encoding is done in read_and_encode.py as follows: Suppose a column C has values from the set {234, 453, 121, 347} then the corresponding 4 columns of the one hot encoded columns of C be encoded in ascending order i.e., the ascending order of the set of values is {121, 234, 347, 453} so 121 be encoded as (1, 0, 0, 0), 234 as (0, 1, 0, 0), 347 as (0, 0, 1, 0), 453 as (0, 0, 0, 1). Next, we drop the first column in the one hot encoded columns of C to avoid the Dummy Variable Trap (For more information click here.) So now, 121 be encoded as (0, 0, 0), 234 as (1, 0, 0), 347 as (0, 1, 0), 453 as (0, 0, 1). You will also need to 'one hot encode' your target while running your algorithms but the final predictions will be a single number representing one of the class {1,2,3,4,5,6,7,8} (Note that there's no issue of Dummy Variable Trap in target one hot encoding so don't drop any of the one hot encoded target columns during this process.).

Append a column of ones as a first column in your data to absorb the bias terms in a single weight matrix. Now, your weight matrix will have shape ([n + 1] x 8) where n is the total number of columns(excluding those of target) after one hot encoding and the first row of the weight matrix will represent the bias terms. Do not shuffle rows or columns of the data within your code, otherwise your solution will not match the reference solution. The input data has already been shuffled. For details of the original dataset, you are encouraged to look at this webpage. You are not allowed to use any library other than NumPy and SciPy for part (a), (b), (c) apart from using pandas for data read and one-hot encoding.

(a) **(12.5 points)** Given a training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, recall that the log-likelihood for logistic regression (k-class) can be written as:

$$L(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \sum_{j=1}^{k} \{y^{(i)} = j\} \log(h_{\theta j}(x^{(i)})) \tag{3}$$

$$h_{\theta j}(x) = \frac{e^{(\theta_j^T x)}}{\sum_{i=1}^{k} e^{(\theta_i^T x)}} \tag{4}$$

($\theta$) represents the decision surface learned by logistic regression and the $1^{st}$ column in X is all ones to take care of constant bias term. $\{y^{(i)} = j\}$ is 1 when $y^{(i)}$ is equal to j and zero otherwise
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 the exact number of iterations provided.

(b) **(12.5 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. (Note that an epoch here means updating gradients with all samples or full data considered once, don't shuffle the data and start picking up batches sequentially from beginning of data matrix, i.e., in a single epoch, first update the gradients using x_index(0 to batchsize-1) then x_index(batchsize to batchsize*2-1) then x_index(batchsize*2 to batchsize*3-1) and so on. Also it maybe possible that the last batch size is smaller than the original batch size if sample size is not a multiple of batch size, in that case ignore the gradient updations with that last smaller batch. )

(c) **(12.5 points)** Find the best algorithm and working hyper-parameters and use the model to train and make predictions on the evaluation data. This part will have a fixed time (10 minute on a CSC baadal machine) to train your model and predict on the test set. (Hint: Plot the loss function, L(w,b) with respect to number of floating point operations, and runtime varying i) Batch size, ii) Variant of the gradient descent algorithm and iii) Value of learning rate parameters ($\eta_0, \alpha, \beta$ for adaptive case), to visualize and select the best algorithm and working hyper-parameters.)

(d) **(12.5 points)** Feature engineering is an important aspect of Machine Learning. Create additional features that you think will be most predictive of the given target. Use a suitable feature selection algorithm along with cross-validation to select the final features that will be most predictive on the test set. Some examples of feature selection algorithms are Large-scale sparse logistic regression, Sparse multinomial logistic regression: fast algorithms and generalization bounds, ANOVA based feature selection.

For this part, you may use any publicly available method or library for feature creation or selection. Your final list of features must be less than 500. With this new data, perform part (c) of the assignment and ensure that the entire process of data reading, training and testing takes less than 15 minutes on the CSC baadal machine.

Make a detailed report stating your parameter tuning method, feature creation, selection and other experiments you performed to arrive at your final solution to parts (c) and (d). Submit this report on gradescope while the code files need to be submitted on moodle.

*Evaluation:*

- 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) Cross-Entropy Loss will be used as evaluation criterion to compare different submissions. There will be relative marking for this part.
- For part (d), marks will be based on your prediction accuracy on the test data. There will be relative marking for this part.
- For parts (a) and (b), evaluation scripts, sample hyper-parameters and sample test data data will be provided. However, the final testing will be on a different test set which will be larger than the test data provided.
- 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.

## 1. Linear Regression

Submit your code in a single executable python3 file called linear.py
Your code will be run as:

python3 linear.py *Mode Parameters*

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

*python3 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 in the very first line) in the weightfile.txt.

*python3 linear.py b trainfile.csv testfile.csv regularization.txt outputfile.txt weightfile.txt bestparameter.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, report the best regularization parameter in the file bestparameter.txt.

*python3 linear.py c trainfile.csv testfile.csv outputfile.txt*
Here, you should create and use your best features (found using Lasso) and calculate your predictions using the same. You will be graded on lowest mean-squared error. In addition, you need to submit the code (on moodle) that generates features and selects a small number of them, in the feature_selection.py. You should select under 300 features. The selected features given by this code must match with the features that you end up using in linear.py.

## 2. Logistic Regression

Submit your code in a single executable python3 file called logistic.py
Your code will be run as:

python3 logistic.py *Mode Parameters*

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

*python3 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 where the first line will correspond to the first row in test.csv and so on. Also output your weight matrix (which includes bias terms in the first row) by flattening the weight matrix rowwise, i.e., write the first row first (1 value per line), then second row and so on and create a line aligned 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 (i)), $\eta_0$ value for adaptive learning rate (for (ii)) or a comma separated (learning rate, $\alpha$, $\beta$) value for $\alpha\beta$ backtracking (for (iii)). The third line will be the exact number of iterations for your gradient updates.

*python3 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) and also the third line is the exact number of epochs to run on.

*python3 logistic.py c trainfile.csv testfile.csv outputfile.txt weightfile.txt*
Here, you are free to use your own learning rate strategies, batch size etc. Again output the outputfile and weightfile as before. For this part, the code would be given 10 minutes to run, and then killed. Make sure that you keep a track of total runtime using *datetime* function and save your output periodically after every few iterations/epoch. If your output files are inconsistent or incomplete, you will get a 0 for this part. Also, update your outputfile.txt and weightfile.txt after every minute.

*python3 logistic.py d trainfile.csv testfile.csv outputfile.txt weightfile.txt*
The logistic.py should create your *selected features* and then run a suitable variant of gradient descent on the given training data that is expected to give best performance on the test data. Here, you are free to use your own learning rate strategies, batch size, feature selection methods etc. Output the outputfile.txt and weightfile.txt as before. For this part, the code would be given 15 minutes to run, and then killed. Make sure that you keep a track of total runtime using *datetime* function and save your output periodically after every few iterations/epoch. If your output files are inconsistent or incomplete, you will get a 0 for this part. Also, update your outputfile.txt and weightfile.txt after every minute.

In addition to this, you are supposed to submit logistic_features_selection.py on moodle that contains your complete code for feature creation and selection. The selected features given by this code must match with the features that you end up using in logistic.py.

3. Data Pre-processing description

Any real dataset will rarely be ideal. In this dataset, we have done the following pre-processing (in order):

- Removed rows(a small fraction) which had one or more values missing.
- Converted the column 'Length of Stay' to integers and an extra step of changing all values greater than 7 to 8 for Logistic Regression.
- Ordinal encoding of all columns which contained categorical data with the following **mapping**.

Note: This section has only been provided to demonstrate the extra efforts that go in data preparation before applying a model.