Coursework	Fall 2021
Machine Learning (B555 - 17011)	Prof. Roni Khardon
Department of Computer Science, IU Bloomington	

Topic : Programming project 3 – Implementation of variant of GLM
Submitted by : Pratap Roy Choudhury, MS in Data Science, Fall202

Overview: Bayesian GLM

In this programming project we will be working with Generalized Linear Models (specifically, the simpler version with canonical link function) as covered in class, including Logistic regression, Poisson regression and Ordinal regression. The goal is to use one generic implementation for the main algorithm that works for multiple observation likelihoods.

Data

Data for this assignment is provided in a zip file pp3data.zip on Canvas. Each dataset is given in two files with the data in one and the labels in the other file. We will use the datasets A and usps for classification with logistic regression. We will use the dataset AP for count prediction with Poisson regression. We will use the dataset AO for ordinal prediction with ordinal regression.

The datasets A, AP, AO were artificially generated with labels which are not perfectly matched to any linear predictor, yet they are generated to be somewhat predictable. The examples in usps represent 16x16 bitmaps of the characters 3 and 5 and are taken from the well-known usps dataset (representing data originally used for zip code classification).

Implementing our variant of GLM

In this assignment we will use a Bayesian (or regularized) version of the algorithm with $w \sim \mathcal{N}(0, \frac{1}{\alpha}I)$, where $\alpha = 10$, to calculate the MAP solution w_{MAP}.

- As discussed in class logistic regression (and by extension GLM) relies on a free parameter (w_0) to capture an appropriate separating hyperplane. Therefore, you will need to add a feature fixed at one (also known as an intercept) to all datasets in the assignment. To match the test case below please add this as the first column in the data matrix.
- The vector of first derivatives of the log posterior is $g = \frac{\partial LogL}{\partial w} = \sum_i d_i \phi(x_i) - \alpha w = \Phi^T d - \alpha w$ where d is a vector whose elements are d_i.
- The matrix of second derivatives of the log posterior is $H = \frac{\partial LogL}{\partial w \partial w^T} = -\sum_i r_i \phi(x_i)\phi(x_i)^T - \alpha I = -\Phi^T R \Phi - \alpha I$ where R is a matrix with elements r_i on the diagonal.
- The GLM algorithm initializes the weight vector as $w = 0$ and then repeatedly applies an update with Netwon's method $w \leftarrow w - H^{-1}g$ until w converges.
- For this assignment we consider that w has converged if $\frac{\|w_{n+1} - w_n\|_2}{\|w_n\|_2} < 10^{-3}$ where w_n, w_{n+1} are the values of w before and after the update respectively. If w has not converged in 100 iterations we stop and output the last w as our solution.
- The final vector, when the algorithm stops is w_{MAP}. In this assignment we will use w_{MAP} for prediction (i.e. we will not calculate a predictive distribution).

Likelihood models

- In order to apply the algorithm for any likelihood model and to evaluate its predictions we need to specify 4 items: (1) d_i, (2) r_i, (3) how to compute our prediction $\hat{t}$ for test example z, and (4) how to calculate the error when we predict $\hat{t}$ and the true label is t.

- For the logistic likelihood we have: $y_i = \sigma(w^T\phi(x_i))$ and the first derivative term is $d_i = t_i - y_i$ or $d = t - y$. The second derivative term is $r_i = y_i(1 - y_i)$. For test example z we predict $t = 1$ iff $p(t = 1) = \sigma(w_{MAP}^T\phi(z)) \geq 0.5$. The error is 1 if $\hat{t} \neq t$.

- Note that for the logistic model the update formula as developed in class is $w_{n+1} \leftarrow w_n - (-\alpha I - \Phi^T R\Phi)^{-1}[\Phi^T(t - y) - \alpha w_n]$. You might want to start developing your code and testing it with this special case and then generalize it to handle all likelihoods. To help you test your implementation of this algorithm we provide an additional dataset, *irlstest*, and solution weight vector in *irlsw*. The first entry in *irlsw* corresponds to w_0.

- For the Poisson likelihood we have: $y_i = e^{(w^T\phi(x_i))}$ and the first derivative term is $d_i = t_i - y_i$ or $d = t - y$. The second derivative term is $r_i = y_i$. For test example z we have $p(t) = Poisson(\lambda)$ where $a = w_{MAP}^T\phi(z)$ and $\lambda = e^a$. We predict with the mode $t = \lfloor\lambda\rfloor$, the nearest integer $\leq \lambda$. For this assignment we will use the absolute error: $\text{err} = |\hat{t} - t|$.

- For the ordinal model with K levels we have parameters s and $\phi_0 = -\infty < \phi_1 < \ldots < \phi_{K-1} < \phi_K = \infty$ where for this assignment we will use $K = 5$, $s = 1$ and $\phi_0 = -\infty < \phi_1 = -2 < \phi_2 = -1 < \phi_3 = 0 < \phi_4 = 1 < \phi_5 = \infty$. The model is somewhat sensitive to the setting of hyperparameters so it is important to use these settings.

  Here $a_i = w^T\phi(x_i)$ and for $j \in \{0, \ldots K\}$ we have $y_{i,j} = \sigma(s * (\phi_j - a_i))$. Using this notation, for example i with label t_i we have $d_i = y_{i,t_i} + y_{i,(t_i-1)} - 1$. For the second derivative we have $r_i = s^2[y_{i,t_i}(1 - y_{i,t_i}) + y_{i,(t_i-1)}(1 - y_{i,(t_i-1)})]$.

  To predict for test example z we first calculate the y values: $a = w_{MAP}^T\phi(z)$ and for $j \in \{0, \ldots K\}$ we have $y_j = \sigma(s * (\phi_j - a))$. Then for potential labels $j \in \{1, \ldots K\}$ we calculate $p_j = y_j - y_{j-1}$. Finally select $\hat{t} = \text{argmax}_j\ p_j$. For this assignment we will use the absolute error, or the number of levels we are off in prediction, that is, $\text{err} = |\hat{t} - t|$.

Implementation and Evaluation of the Algorithms

The main idea in GLM is to build a generic implementation that in principle can handle any likelihood model. Accordingly, one implementation of the optimization and evaluation process which makes use of functions that compute the 4 items described in the previous section should be created. It can be then called on each likelihood model to obtain results.

The task is to implement the GLM algorithm and generate learning curves with error bars (i.e., +- 1σ) as follows.

Repeat 30 times

- Step 1) Set aside 1/3 of the total data (randomly selected) to use as a test set.
- Step 2) Permute the remaining data and record the test set error rate as a function of increasing training set portion (0.1,0.2, . . . ,1 of the total size).

Calculate the mean and standard deviation for each size and plot the result. In addition, record the number of iterations and the run time until convergence in each run and report their averages. In the submission provide 4 plots, one for each dataset, and corresponding runtime/iterations statistics,

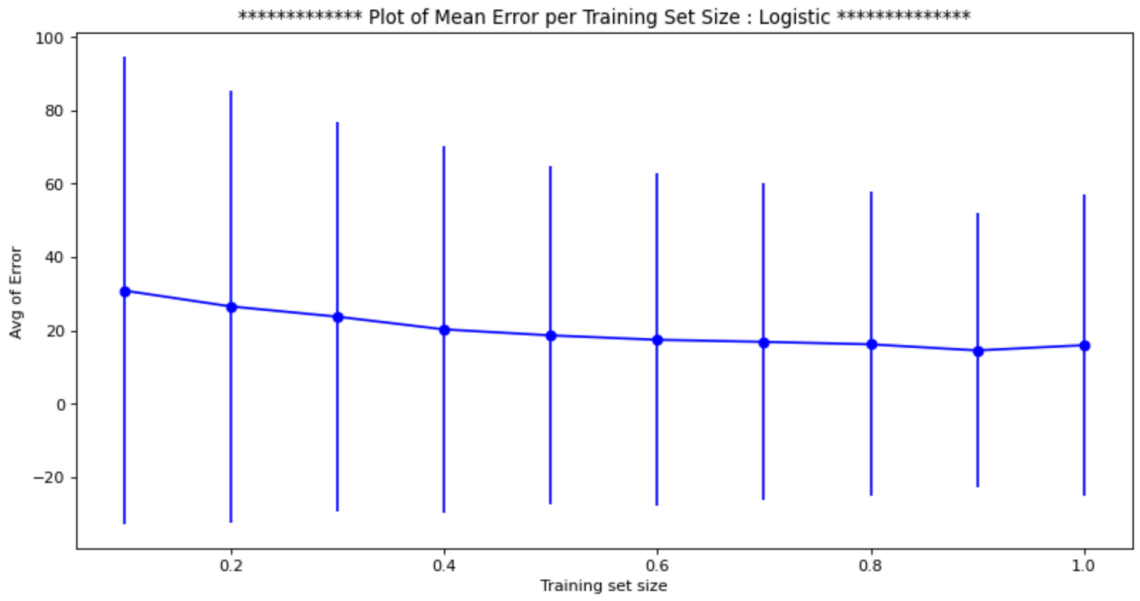
and provide a short discussion of the results. Are the learning curves as expected? How does learning time vary across datasets for classification? and across the likelihood models? What are the main costs affecting these (time per iteration, number of iterations)?

The results and plots of GLM applied on different datasets in different methods are listed below –

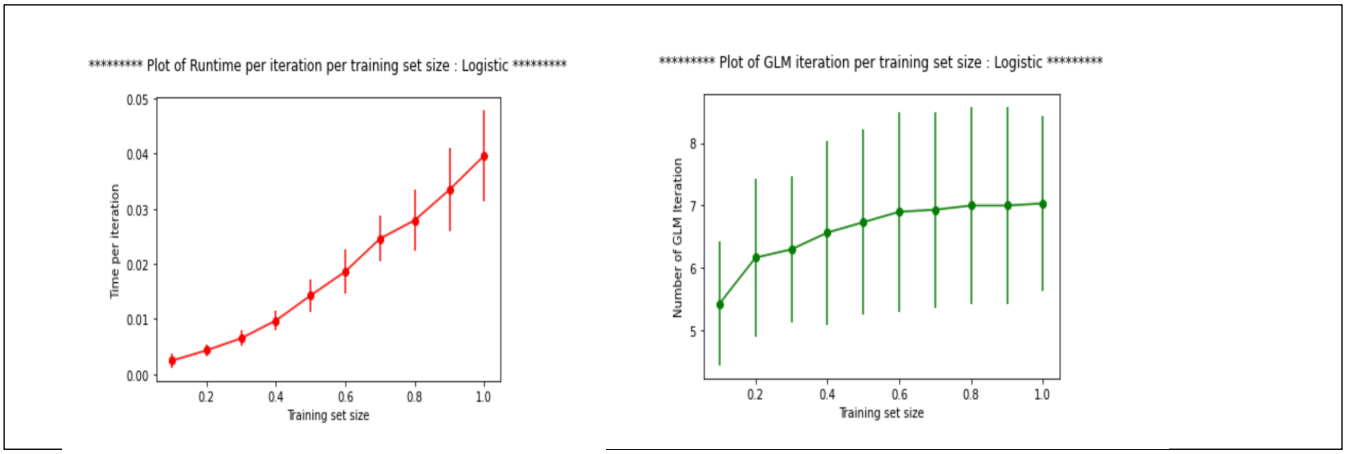
Process Logistic: Data – A

```
TRAINING SET :  (2000, 60) (2000, 1)

        !!!!!!!!!!!!!!!! RESULTS Task 1 - Logistic !!!!!!!!!!!!!!!!


 Mean of training errors per training set size :  [30.9, 26.566666666666666, 23.766666666666666, 20.3, 18.66666666666
6668, 17.466666666666665, 16.9, 16.233333333333334, 14.566666666666666, 16.0]

 SD of training errors per training set size :  [63.81919773861153, 58.80118101610619, 52.89340433572242, 50.11596551
9981735, 46.183210032314655, 45.28850724950966, 43.28999114499024, 41.4404659990959, 37.37528892493303, 41.0105677437
73876]

 Mean of run time per iteration per training set size :  [0.0021529356638590497, 0.0037380297978719073, 0.00632940133
4126791, 0.010774072011311848, 0.014901455243428547, 0.01954357624053955, 0.02672297159830729, 0.028281641006469727,
0.033153049151102704, 0.04046089649200439]

 Mean of GLM iterations per training set size :  [5.4, 6.033333333333333, 6.166666666666667, 6.533333333333333, 6.7,
6.766666666666667, 6.766666666666667, 6.9, 6.9, 6.9]
```

```
************* Plot of Mean Errors **************
```



For Logistic Regression, the error rate decreases as the training set size increases and has the highest error 30.9 at training set size proportion = 0.1. It decreases at a slower rate till 16 when the training size is full. The Standard deviation at each training set size decreases and thus the error curve is decreasing which is as expected.

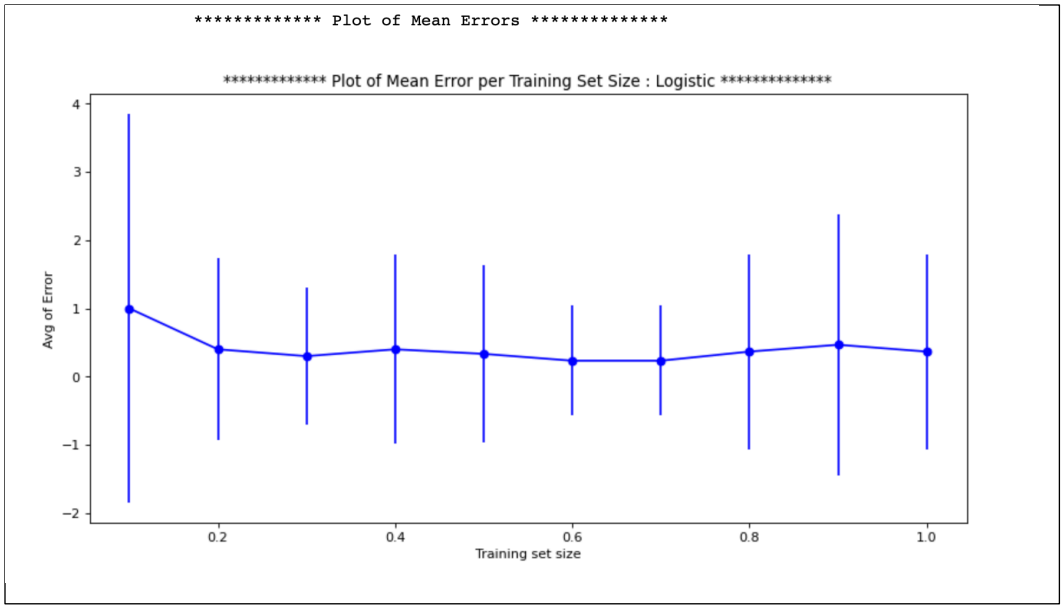


The Run time per iteration for each test set size increases as training set size increases.

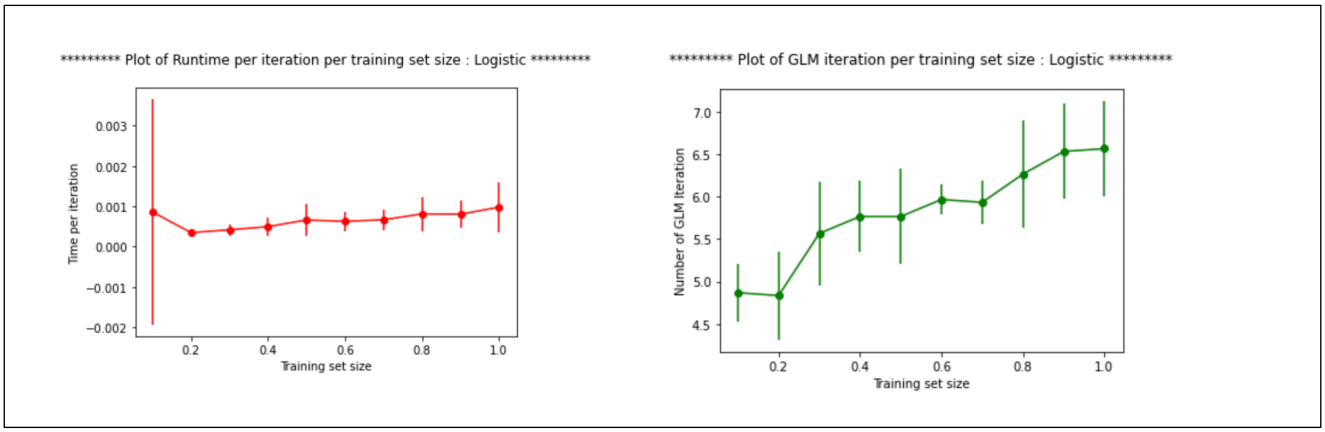
The average number of GLM iterations before convergence for each training set size increases continuously and stays between 5-7 iterations.

Process Logistic: Data – irlstest

```
TRAINING SET :  (200, 10) (200, 1)

        !!!!!!!!!!!!!!!! RESULTS Task 1 - Logistic !!!!!!!!!!!!!!!!


 Mean of training errors per training set size :  [1.0, 0.4, 0.3, 0.4, 0.3333333333333333, 0.23333333333333334, 0.233
33333333333334, 0.36666666666666664, 0.4666666666666667, 0.36666666666666664]

 SD of training errors per training set size :  [2.851899951494325, 1.3316656236958775, 1.004987562112089, 1.38082101
18138642, 1.299572579307863, 0.8034647195462641, 0.8034647195462641, 1.425560318689541, 1.9102065042525884, 1.4255603
18689541]

 Mean of run time per iteration per training set size :  [0.0008588711420694987, 0.0003485838572184246, 0.0004178365
071614583, 0.0004961490631103516, 0.0006618261337280274, 0.0006272077560424804, 0.0006671826044718425, 0.000811656316
1214193, 0.0008063554763793945, 0.0009783109029134114]

 Mean of GLM iterations per training set size :  [4.866666666666666, 4.833333333333333, 5.566666666666666, 5.76666666
6666667, 5.766666666666667, 5.966666666666667, 5.933333333333334, 6.266666666666667, 6.533333333333333, 6.56666666666
6666]
```



The learning curve of error in irlstest dataset decreases from the very first step ie. From 0.1 to 0.2 training set size but then follows the same trend.



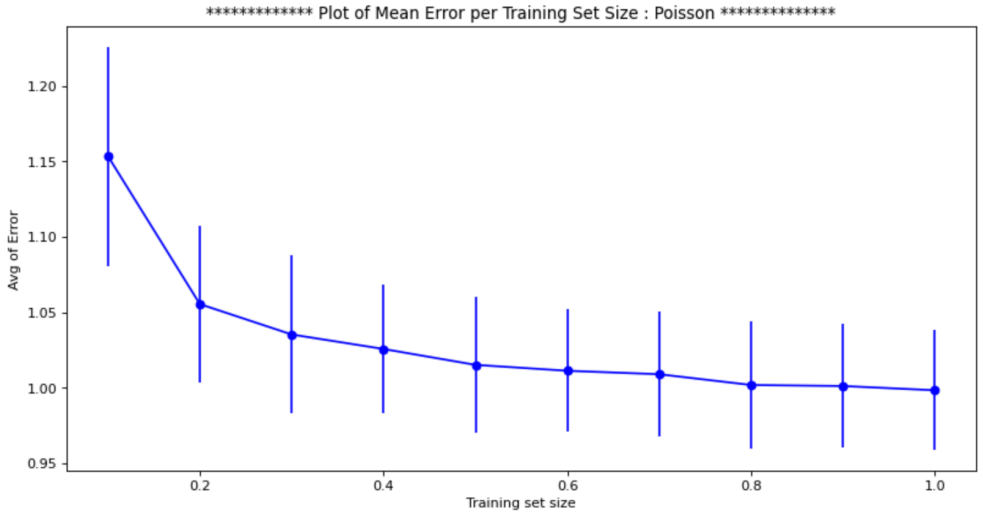
The run time per iteration for each training set size first decreases from 0.1 to 0.2 and then increases at a very slow rate.

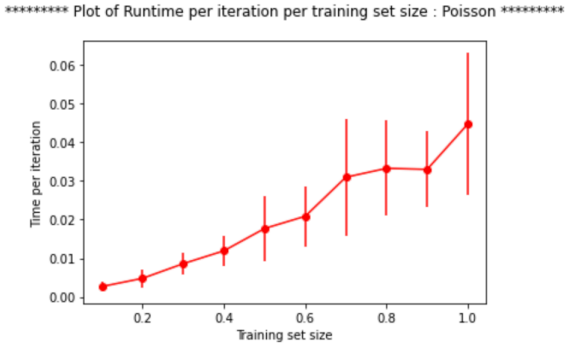
The GLM iteration per training set size actually increases from o.1 to 1 size but initially it decreases from 0.1 to 0.2 and then increases 0.4, then almost equalizes at 0.5, then again increases at 0.6 and further more.

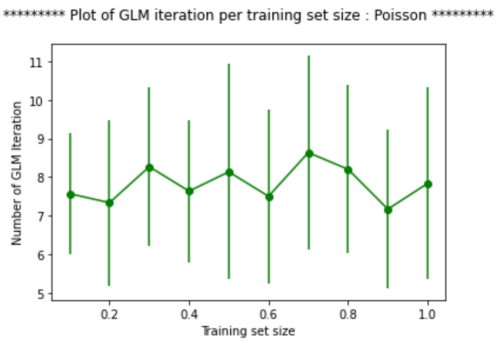
Process Poisson: Data – AP

```
TRAINING SET :  (2000, 60) (2000, 1)

        !!!!!!!!!!!!!!!! RESULTS Task 2 - Poisson !!!!!!!!!!!!!!!!


 Mean of training errors per training set size :  [1.153303303303303, 1.0554054054054054, 1.0352852852852854, 1.02567
56756756757, 1.0151651651651652, 1.0112612612612617, 1.008958958958959, 1.001851851851852, 1.0011511511511513, 0.9983
483483483486]

 SD of training errors per training set size :  [0.07242505112648052, 0.05193451830130117, 0.05211421655597366, 0.042
70595125678881, 0.04475943823614321, 0.040578986700867445, 0.0411650327983673, 0.04186413691111599, 0.040844674151297
194, 0.039709628963391086]

 Mean of run time per iteration per training set size :  [0.002689814567565918, 0.0048056920369466145, 0.008589529991
149902, 0.011913569768269856, 0.017668000857035317, 0.020833667119344076, 0.030910396575927736, 0.033211596806844074,
0.0329428752263387, 0.044661339124043783]

 Mean of GLM iterations per training set size :  [7.566666666666666, 7.333333333333333, 8.266666666666667, 7.63333333
3333334, 8.133333333333333, 7.5, 8.633333333333333, 8.2, 7.166666666666667, 7.833333333333333]
```

********** Plot of Mean Errors **********







For Poisson Regression, the error rate decreases significantly from 0.1 to 0.2 as the training set size increases and has the highest error at training set size proportion = 0.1. The Standard deviation at each training set size decreases and thus the error curve is decreasing which is as expected.

The Run time for each test set size increases as training set size increases.

The average number of GLM iterations before convergence for each training set size varies as it decreases and increases sequentially but on an average stay between 7-8. At training set size 0.7 it's the highest 8.6 which is appx 9.

Process Ordinal: Data – AO

There is a small error present in the script for which the code taking longer time than expected and could not getting the expected result when running for 30 repetitions and for 10 training set segments.