CSCI-B555 MACHINE LEARNING

PROGRAMMING PROJECT 3

Bayesian Generalized Linear Model

In this experiment we implement GLM for logistic, Poisson and ordinal regression. We implement a generic, main algorithm that works for multiple observation likelihoods. First, we calculate the MAP solution for our parameter vector *w*: W_{map} which we will use for predictions.

Logistic:

Results:

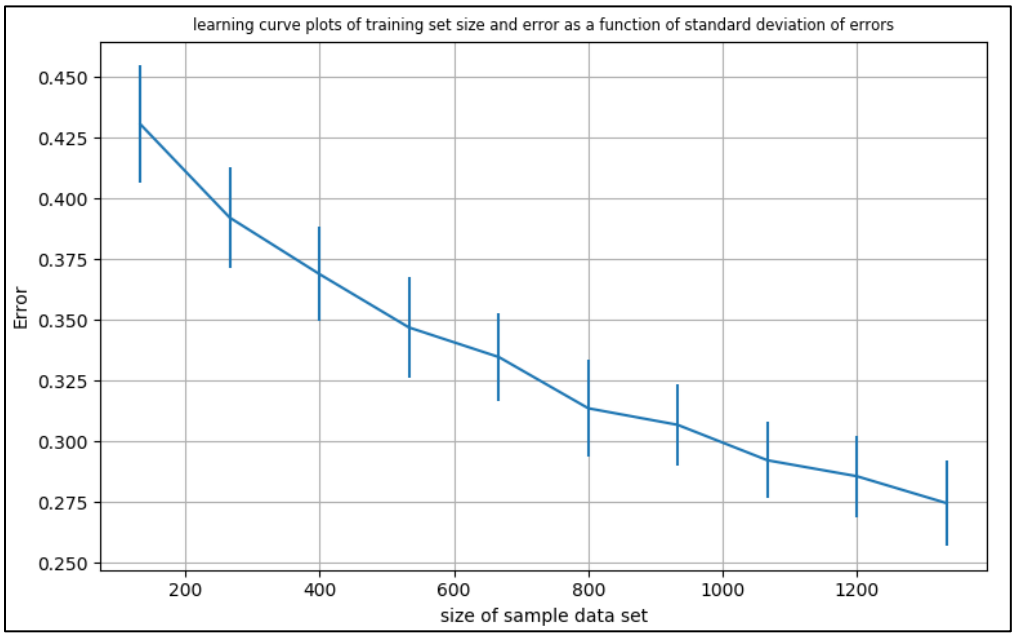

Figure 1: Plot for A data set

Dataset	sizes	Average iterations	Runtime until convergence (in sec)	Overall Run Time (in sec)
A	0.1N	2	0.0012	7.54
	0.2N	2	0.0013	
	0.3N	2	0.0031	
	0.4N	2	0.0044	
	0.5N	2	0.0063	
	0.6N	2	0.0084	
	0.7N	2	0.0104	
	0.8N	2	0.0138	
	0.9N	2	0.0146	
	1N	2	0.0171	

Conclusion:

For the GLM implementation using logistic likelihood (for 'A' dataset) we find that the average iterations for convergence to get W_{map} for each size is 2. Also, the runtime for convergence is gradually increasing as the size of the data is increasing which is obvious and expected. Total runtime for the logistic implementation is 7.54 seconds. Also, we can see that as the size of dataset increases the error reduces. Thus, the learning curves are as expected.

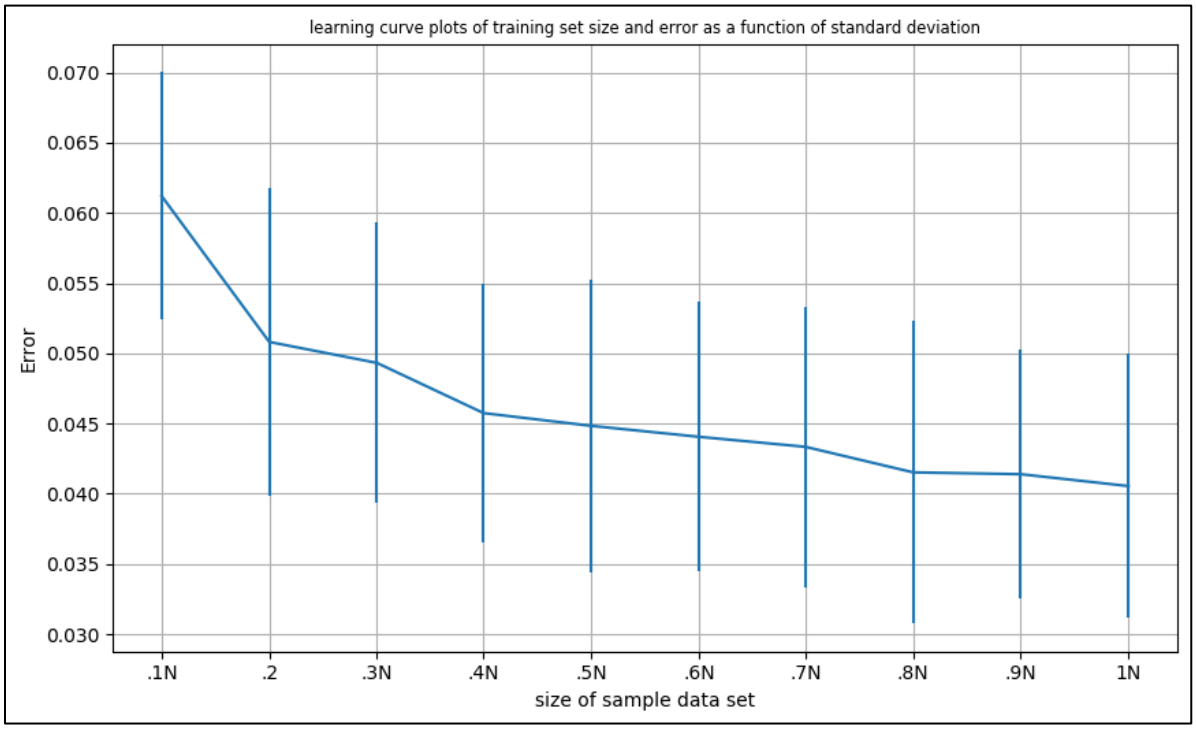

Figure 2: Plot for USPS data set

Dataset	sizes	Average iterations	Runtime until convergence (in sec)	Overall Run Time (in sec)
USPS	0.1N	4	0.0096	13.87
	0.2N	5	0.0137	
	0.3N	5	0.0158	
	0.4N	5	0.0212	
	0.5N	5	0.0278	
	0.6N	6	0.0358	
	0.7N	6	0.0426	
	0.8N	6	0.0494	
	0.9N	6	0.0575	
	1N	6	0.0653	

Conclusion:

For the GLM implementation using logistic likelihood (for 'usps' dataset) we find that the average iterations for convergence to get W_{map} increases with increase in the size of dataset. Also, the runtime for convergence is gradually increasing as the size of the data is increasing which is evident as the data size also increasing and the average iterations also increases. Runtime until convergence depends on both size of the data and the average iterations. Total runtime for the logistic implementation is 13.87 seconds. The overall runtime for this dataset is more compare to the 'A' dataset. This can be attributed to the average iterations for each size, which is more in the 'USPS' than in 'A'. The learning curves are as expected, as the size of dataset increases the error reduces.

Poisson:

Results:

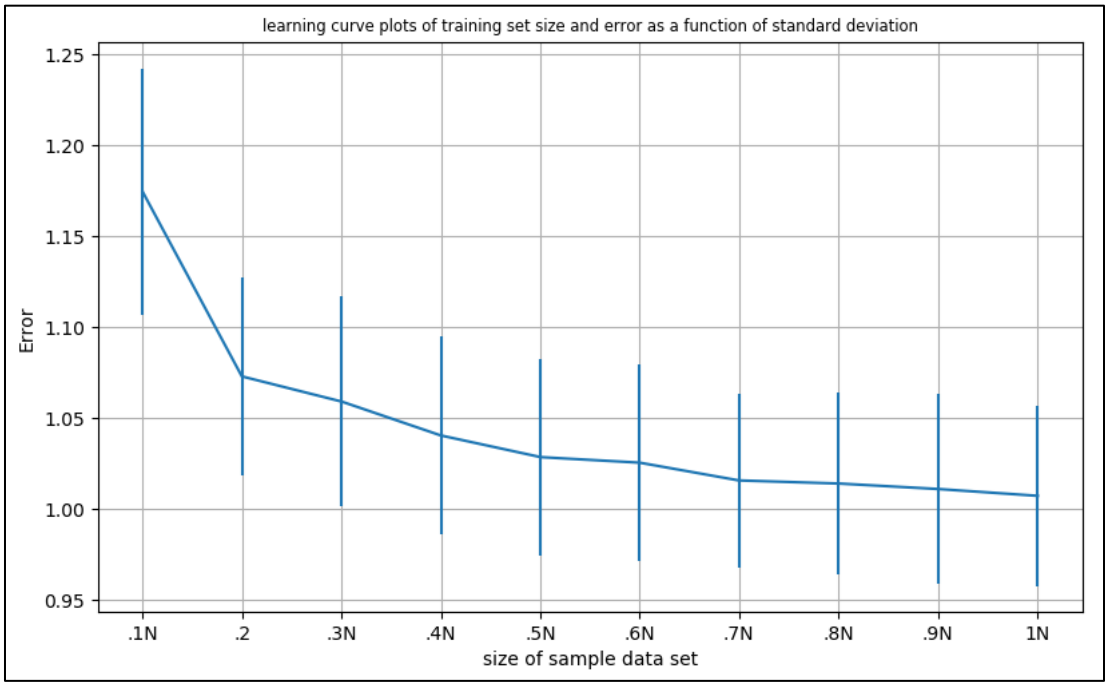

Figure 3: Plot for AP data set

Dataset	sizes	Average iterations	Runtime until convergence (in sec)	Overall Run Time (in sec)
AP	0.1N	6	0.0023	11.2
	0.2N	6	0.0029	
	0.3N	7	0.0080	
	0.4N	6	0.0115	
	0.5N	7	0.0158	
	0.6N	6	0.0199	
	0.7N	7	0.0263	
	0.8N	7	0.0337	
	0.9N	6	0.0356	
	1N	6	0.0437	

Conclusions:

For the GLM implementation using Poisson likelihood (for 'AP' dataset) we find that the average iterations for convergence to get W_{map} for some different sizes of the dataset is either 6 or 7. The runtime for convergence depends on both average iteration as well as the size of the data set. As the size is increasing the runtime for convergence is also increasing. Total runtime for the Poisson implementation is 11.2 seconds. The learning curves are as expected, as the size of dataset increases the error reduces.

Ordinal:

Results:

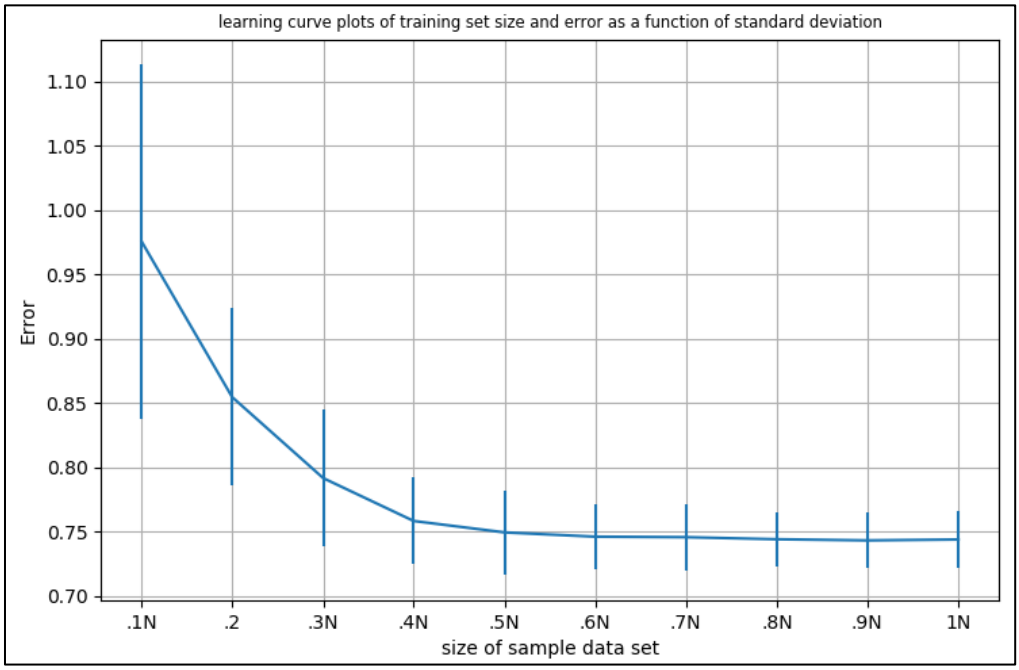

Figure 4: Plot for AO data set

Dataset	sizes	Average iterations	Runtime until convergence (in sec)	Overall Run Time (in sec)
AO	0.1N	3	0.0067	17.20
	0.2N	3	0.0116	
	0.3N	3	0.0187	
	0.4N	3	0.0257	
	0.5N	3	0.0340	
	0.6N	3	0.0426	
	0.7N	3	0.0485	
	0.8N	3	0.0568	
	0.9N	3	0.0650	
	1N	3	0.0707	

Conclusions:

For the GLM implementation using Ordinal likelihood (for 'AO' dataset) we find that the average iterations for convergence to get W_{map} for different sizes of the dataset is 3. Thus, in this case the runtime for convergence will depend only on the size of the data set. As the size is increasing the runtime for convergence is also increasing. Total runtime for the Poisson implementation is 17.20 seconds. The learning curves are as expected, as the size of dataset increases the error reduces.

Model Selection:

We have been asked to explore some model selection approaches. In previous programming programing project, we worked for the model selection for Bayesian linear regression using a linear search along with k-fold cross validation and using the evidence function. While linear search is not efficient, I've tried to work with a version of it called random search. According to the paper by **James Bergstra** and **Yoshua Bengio** "*Random search from Hyper-parameter Optimization*" [1], which states that "randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid." As mention by the professor that an optimal value of alpha would be between 0 to 100, I select 50 random number from 0 to 100 and perform a search using those 50 number. I've used K-fold cross validation to measure the performance of the model. Below are the results which I got for different data sets:

1. **Logistic:**
   'A' dataset:
   Best alpha we got is: 1
   The associated error is: 0.14
   When alpha is 10:
   The associated error is: 0.27

2. **Poisson:**
   'AP' dataset:
   Best alpha we got is: 71
   The associated error is: 0.96
   When alpha is 10:
   The associated error is: 1.01

3. **Ordinal:**
   'AO' dataset:
   Best alpha we got is: 11
   The associated error is: 0.75
   When alpha is 10:
   The associated error is: 0.749

Reference:

1 - http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf