

## Programming Project 3

This assignment is due by Friday 11/5, 11:59pm via Canvas.

You can write your code in any programming language so long as we are able to test it on SICE servers. We plan to run some or all of submitted code for further testing and validation.

### 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. Your 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  $16 \times 16$  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  $g = \frac{\partial \text{Log}L}{\partial w} = \sum_i d_i \phi(x_i) - \alpha w = \Phi^T d - \alpha w$  where  $d$  is a vector whose elements are  $d_i$ .

- The matrix of second derivatives of the log posterior is  $H = \frac{\partial^2 \text{Log} L}{\partial w \partial w^T} = -\sum_i r_i \phi(x_i) \phi(x_i)^T - \alpha I = -\Phi^T R \Phi - \alpha I$  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 Newton'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  $p(t) = \text{Poisson}(\lambda)$  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  $\phi_0 = -\infty < \phi_1 < \dots < \phi_{K-1} < \phi_K = \infty$  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  $j \in \{0, \dots, K\}$  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  $j \in \{0, \dots, K\}$  we have  $y_j = \sigma(s * (\phi_j - a))$ . Then for potential labels  $j \in \{1, \dots, K\}$  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 you should provide one implementation of the optimization and evaluation process which makes use of functions that compute the 4 items described in the last section. You can then call this implementation on each likelihood model to obtain results.

Your task is to implement the GLM algorithm and generate learning curves with error bars (i.e.,  $\pm 1\sigma$ ) 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 your 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)?

## Extra Credit

Explore some approach for model selection for  $\alpha$  in all models and/or for  $s$  and  $\phi$  in the ordinal model and report your results. You may want to generate your own data with known parameters in order to test the success of algorithms in identifying good parameters.

## Submission

Please submit two separate items via Canvas:

(1) A zip file `pp3.zip` with all your work and the report. The zip file should include: (1a) Please write a report on the experiments, include all plots and results, and your conclusions as requested above. Prepare a PDF file with this report. (1b) Your code for the assignment, including a README file that explains how to run it. When run your code should produce all the results and plots as requested above. Your code should assume that the data files will have names as specified above and will reside in sub-directory `pp3data/` of the directory where the code is executed. We will read your code as part of the grading – please make sure the code is well structured and easy to follow (i.e., document it as needed). This portion can be a single file or multiple files.

(2) One PDF “printout” of all contents in 1a, 1b: call this `YourName-pp3-everything.pdf`. One PDF file which includes the report, a printout of the code and the README file. We will use this file as a primary point for reading your submission and providing feedback so please include anything pertinent here.

## Grading

Your assignment will be graded based on (1) the clarity of the code, (2) its correctness, (3) the presentation and discussion of the results, (4) our ability to run the code on SICE servers.