CS 136 - 2022s - Checkpoint 2

Student Names: Nate Davis and Alexander Lobo

Collaboration Statement:

Turning in this assignment indicates you have abided by the course Collaboration Policy:

`www.cs.tufts.edu/comp/136/2022s/index.html#collaboration-policy`

Total hours spent: TODO

We consulted the following resources:

- TODO
- TODO
- . . .

These are the official instructions for checkpoint 2. You can find instructions on how to submit at `www.cs.tufts.edu/comp/136/2022s/checkpoint2.html`

Please consult the full project description at `https://www.cs.tufts.edu/comp/136/2022s/project.html` in addition to this document when working on this checkpoint. It gives details on what we expect.

Applying Model to Dataset

For this project, we have decided to implement a Bayesian Logistic regression model to predict the class of conformations of molecules as either ”musk” or ”non-musk”.

We used the `MAPEstimator.py` class from CP2 as a template to set up our model. In our python file, we have a `MAPEstimator` class with an initialization (`_init_`). Various methods (`fit()`, `predict_proba()`, and `score()`) are also implemented to train and test the model. Details of our implementation are below.

Initialization

The `MAPEstimator` class has the following attributes that are defined in the initialization:

- `w_D` : defines the first instance of the weight vector
  - is a vector of all zeros since our prior (both trivial and SAS) assumes a zero mean
- `prior` : determines which prior to use (`'trivial'` or `'sas'`)
  - `'trivial'` prior :
$$p(\mathbf{w}|\alpha) = \mathcal{N}(\vec{0}, \alpha^{-1} I_{166}) \quad (1)$$
  - `'sas'` prior (sas stands for ”spike-and-slab” [1]):
$$p(\mathbf{w}|\alpha, \beta) = 0.8\mathcal{N}(\vec{0}, \alpha^{-1} I_{166}) + 0.2\mathcal{N}(\vec{0}, \beta^{-1} I_{166}) \quad (2)$$
- `alpha` : A real value that determines the precision of either the trivial prior or first MVN in the SAS prior
- `beta` : A real value that determines the precision of the second MV in the SAS prior (should only be specified if `prior='sas'`

[1] `https://en.wikipedia.org/wiki/Spike-and-slab_regression`

- `max_iter` : A positive integer that determines the maximum number of iterations allowed for the stochastic gradient descent
- `tol` : A real positive value that determines the tolerance for convergence for the stochastic gradient descent
- `step_size` : A positive real value that determine how large the weight update is per iteration

Training

The `fit()` method is used to train the Bayesian logistic regression model with the provided training data. Since stochastic gradient descent is being used, a while loop iterates over each training example to compute the gradient of the posterior distribution. This gradient is then used to update to update the weight vector and bias term according to the following equations, where c is the bias term:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta\left[\left(\sigma(\mathbf{w}^t \cdot x^{(i)}) - y^{(i)}\right) x^{(i)} + \alpha \mathbf{w}^t\right] \tag{3}$$

$$c^{t+1} \leftarrow c^t - \eta(\sigma(\mathbf{w}^t \cdot x^{(i)}) - y^{(i)}) \tag{4}$$

After each update, the log loss for that example is computed using the following equation:

$$L = y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \tag{5}$$

This loss is then appended to a vector of losses. Since stochastic gradient descent is prone to fluctuations in the loss at each step, after 10 iterations, the previous ten losses are averages and then compared to the 10 before that to see if the average loss change is reducing. If the average loss change becomes less than the tolerance, then it is said to have converged and the loop halts. If the average change in loss does not converge, then the loop halts when the maximum number of iterations are reached.

Evaluation

The `predict_proba()` method is used to predict the probability of each example being in the ”musk” and ”non-musk” class while the `score()` method uses a threshold of 0.5 to characterize each example and then compute the accuracy of the predictions by comparing to the true class labels. After splitting our data set into a training and testing set and training the model on the training set, we can evaluate the models performance by calculating the prediction score with the testing set. We can also use the `iter_count` attribute to evaluate many iterations it took for the training to converge.

Bottlenecks

We decided to first test our model with a learning rate and α value of both 1.0. However, upon implementation, we noticed that the model would not converge, instead, the loss would shoot up to infinity after a few iterations. We then normalized the data using a standard scaler, assuming that the large feature values were causing large steps in the weight updates. However, this led to the same result. After deliberating, we decided to reduce our step-size to 0.001. After this change, the average loss started decreasing with each iteration and took on average about 100 iterations to converge with an average testing accuracy of 75%.***

***Need to update this once we finalize our testing

Evaluating Hypotheses from Checkpoint 1

We tweaked our hypotheses from Checkpoint 1. All three are now the following:

- We hypothesize that the time and iterations needed to compute the weight vectors will decrease after upgrading from a first order to second order gradient descent
- We hypothesize that our multivariate logistic regression model with first-order stochastic gradient descent will converge in fewer iterations when we use a spike and slab prior as we assume that, with 166 features, some should conceivably have little to no weight while others will bear the brunt of prediction.
- We hypothesize that our multivariate logistic regression model with first-order stochastic gradient descent will converge in fewer iterations when we use different step sizes for misclassifying the two output classes as our output classes are highly imbalanced.

However, at this stage (pre-upgrade), we can only evaluate the last hypothesis.

Hypothesis 1

Hypothesis:

We hypothesize that our multivariate logistic regression model with first-order stochastic gradient descent will converge in fewer iterations when we use different step sizes for misclassifying the two output classes as our output classes are highly imbalanced.

Evaluating Hypothesis:

We chose to evaluate our hypotheses by training the model using K-fold cross validation with 10 folds, noting how many iterations it took the model to converge for each fold, and averaging those numbers.

Result:

Hypothesis:

Proposing an Upgrade to your Model or Learning Method

The purpose of this study is to explore how our model performs when upgrading our learning method from a first order to a second order gradient descent algorithm to solve the weight vector MAP estimate for logistic regression. We propose the following hypotheses for how we expect our model to perform: