SuperAGI Assignment

Name: Jasjot Singh

Roll No: 2022EET2105

**Q1)** You train Logistic Regression with a certain set of features and learn weights w\_0, w\_1 till w\_n. Feature n gets weight w\_n at the end of training. Say you now create a new dataset where you duplicate feature n into feature (n+1) and retrain a new model. Suppose this new model weights are w\_{new\_0}, w\_{new\_1} till w\_{new\_n}, w\_{new\_{n+1}}. What is the likely relationship between w\_{new\_0}, w\_{new\_1} , w\_{new\_n}, and w\_{new\_{n+1}}?

**Ans:**

In a logistic regression model, each weight corresponds to the importance or contribution of the associated feature to the prediction. The weight determines the impact of that feature on the predicted outcome. If you duplicate feature n into a new feature n+1 and then retrain the model, it is likely that the weights associated with these duplicated features will be similar. This is because both features now provide essentially the same information to the model during training, and the model will try to distribute the importance of representing that information across both weights. This is based on the assumption that the duplicated feature provides similar information to the original feature and, as a result, the model assigns similar weights to both during the training process.

If the duplicated feature provides new information or complements the existing information in the original feature, it might impact the weights of the features. The specific relationship, however, would depend on the dataset and the training process.

**Q2)** You currently have an email marketing template A and you want to replace it with a better template. A is the control\_template. You also test email templates B, C, D, E. You send exactly 1000 emails of each template to different random users. You wish to figure out what email gets the highest click through rate. Template A gets 10% click through rate (CTR), B gets 7% CTR, C gets 8.5% CTR, D gets 12% CTR and E gets 14% CTR. You want to run your multivariate test till you get 95% confidence in a conclusion. Which of the following is true?

* We have too little data to conclude that A is better or worse than any other template with 95% confidence.
* E is better than A with over 95% confidence, B is worse than A with over 95% confidence. You need to run the test for longer to tell where C and D compare to A with 95% confidence.
* Both D and E are better than A with 95% confidence. Both B and C are worse than A with over 95% confidence

**Ans:**

E is better than A with over 95% confidence, B is worse than A with over 95% confidence. You need to run the test for longer to tell where C and D compare to A with 95% confidence.

**Q3)** You have m training examples and n features. Your feature vectors are however sparse and average number of non-zero entries in each train example is k and k << n. What is the approximate computational cost of each gradient descent iteration of logistic regression in modern well written packages?

**Ans:**

In well-written packages that leverage sparse data structures, such as SciPy's sparse matrix representation, the computational cost is often closer to *O*(*m*⋅*k*) rather than *O*(*m*⋅*n*). This is because the sparse matrix multiplication is optimized to only consider non-zero entries.

So, the approximate computational cost of each gradient descent iteration for logistic regression in modern well-written packages with sparse features is around *O*(*m*⋅*k*).

**Q4)** We are interested in building a high quality text classifier that categorizes news stories into 2 categories - information and entertainment. We want the classifier to stick with predicting the better among these two categories (this classifier won't try to predict a percent score for these two categories). You have already trained V1 of a classifier with 10,000 news stories from the New York Times, which is one of 1000 new sources we would like the next version of our classifier (let's call it V2) to correctly categorize stories for. You would like to train a new classifier with the original 10,000 New York Times news stories and an additional 10,000 different news stories and no more. Below are approaches to generating the additional 10,000 pieces of train data for training V2.

1. Run our V1 classifier on 1 Million random stories from the 1000 news sources. Get the 10k stories where the V1 classifier’s output is closest to the decision boundary and get these examples labeled.

2. Get 10k random labeled stories from the 1000 news sources we care about.

3. Pick a random sample of 1 million stories from 1000 news sources and have them labeled. Pick the subset of 10k stories where the V1 classifier’s output is both wrong and farthest away from the decision boundary.

**Ans:**

Approach 1 focuses on challenging examples near the decision boundary, potentially improving the model's performance on ambiguous cases. It provides diversity in the dataset.

Approach 3 targets cases where the current model is wrong, potentially improving its weaknesses. Prioritizes cases far from the decision boundary, which may be more confidently misclassified.

Considering the goal of improving the model's performance on challenging cases and near the decision boundary, a combination of Approach 1 and Approach 3 might be beneficial. This way, you can ensure diversity by selecting cases near the decision boundary (Approach 1) and explicitly target misclassified cases (Approach 3).

Ranking based on potential impact on accuracy:

1. Approach 3
2. Approach 1
3. Approach 2

**Q5)** You wish to estimate the probability, p that a coin will come up heads, since it may not be a fair coin. You toss the coin n times and it comes up heads $k$ times. You use the following three methods to estimate p

1. Maximum Likelihood estimate (MLE)

2. Bayesian Estimate: Here you assume a continuous distribution uniform prior to p from [0,1] (i.e. the probability density function for the value of p is uniformly 1 inside this range and 0 outside. Our estimate for p will be the expected value of the posterior distribution of p. The posterior distribution is conditioned on these observations.

3. Maximum a posteriori (MAP) estimate: Here you assume that the prior is the same as (b). But we are interested in the value of p that corresponds to the mode of the posterior distribution. What are the estimates?

**Ans:**

1. P(MLE) = k/n
2. P(Bayesian) = (k+1)/(n+2)
3. P(MAP) = k/n