Q/A Assignment

1. **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 $w_{new_{n+1}}$?

   Since feature n and feature $n+1$ contain the same information (feature $n+1$ is a duplicate of feature n), we would expect the Logistic Regression model to distribute the weight across these two identical features. So:

   - $w_{new_0}, w_{new_1}, ..., w_{new_n}$ **will likely stay approximately the same as in the original model.** Adding the duplicate feature does not change the informativeness of the other existing features.
   - $w_{new_n} + w_{new_{n+1}}$ **will likely be approximately equal to w_n from the original model.** The total weight assigned to that feature information will be preserved, just divided across two features now.
   - Individually, w_{new_n} and $w_{new_{n+1}}$ will likely receive similar values since the features contain identical information. If w_n was originally high, the new weights will be distributed so their sum is still high.

2. **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?**

   a. **We have too little data to conclude that A is better or worse than any other template with 95% confidence.**

b. **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.**

c. **Both D and E are better than A with 95% confidence. Both B and C are worse than A with over 95% confidence**

Okay, let's analyze this statistically:

- Template A (control) has a 10% CTR
- 1000 emails were sent for each template
- We want 95% confidence in comparisons to A

To compare CTRs, we can use a significance test like a z-test. The key thing we need is the standard error of each CTR percentage, which depends on the sample size and observed percentage.

With a sample size of 1000 and percentages ranging from 7% to 14%, the standard errors will be small enough to conclude some statistically significant differences at 95% confidence:

- Template E at 14% CTR is very likely to be significantly higher than A's 10% CTR.
- Template B at 7% is likely significantly lower than A.

However, templates C and D are more borderline:

- C is at 8.5%, which may or may not be significantly lower than A's 10%
- D is at 12%, which may or may not be significantly higher

So, choice 2 best reflects the statistical conclusions we can draw:

1. 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.

We likely have enough data to determine E and B's relationships with 95% confidence due to more extreme CTR differences from A, but we need more data for the more borderline differences with C and D.

3. **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?**

   Well-written packages leverage efficient sparse matrix operations and optimizations specific to sparse datasets. So, focusing on other operations, we have:

   1. **Computational complexity of logistic regression iteration:**
      - The computational complexity of a single iteration of gradient descent in logistic regression is generally **linear** with respect to the **number of training examples m.**
      - For each training example, the cost mainly **depends on the number of non-zero features k** rather than the total number of features n. This is because, in sparse representations, calculations skip zero-valued features.
   2. **Sparse matrix operations:**
      - The computational complexity of matrix-vector multiplication involving sparse matrices is often proportional to the number of non-zero elements in the matrix, which is approximately $O(k \cdot m)$ for each iteration.
   3. **Overall computational cost per iteration:**
      - Considering the sparsity and optimized operations for sparse matrices, the approximate computational cost per iteration for logistic regression in well-written packages would be around $O(k \cdot m)$.

   Factors like regularization, convergence criteria, and specific optimization algorithms (e.g., stochastic gradient descent, mini-batch gradient descent) can also influence the overall computational complexity.

4. **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.

a. **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.**
b. **Get 10k random labeled stories from the 1000 news sources we care about.**
c. **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.**

Ignore the difference in costs and effort in obtaining train data using the different methods described above. In terms of pure accuracy of classifier V2 when classifying a bag of new articles from 1000 news sources, what is likely to be the value of these different methods?How do you think the models will rank based on their accuracy?

All three methods have potential pros and cons for improving the accuracy of classifier V2:

1. Getting 10k stories close to the decision boundary can help improve performance on borderline cases. However, it may overfit to those specific types of examples.
2. Getting a random sample of labelled stories provides more diversity but doesn't specifically target areas the V1 classifier struggled with.
3. Getting misclassified examples far from the decision boundary ensures a diversity of mistakes, but some may be outliers that don't generalize.

Ranking them:

1. Method 3 potentially provides the most useful new examples by specifically targeting cases the V1 classifier got very wrong. Focusing on clear mistakes far

from the decision boundary means they are likely systemic issues rather than outliers. This provides high-value training data.

2. Method 1 gets borderline cases right, improving the accuracy of hard decisions but could lead to overfitting. Still likely valuable for boosting V2 accuracy.
3. Method 2 provides a random new sample, adding diversity. But without targeting areas for improvement, it may not lead to accuracy gains over V1.

So the would rank be:
3 > 1 > 2

Method 3 targets clear systemic mistakes to provide the highest value new training data, and is likely to improve classifier V2's accuracy the most. Method 1 helps borderline cases, also boosting accuracy. Method 2 adds some diversity but may not improve over V1 without targeting weak areas.

5. **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**
   a. **Maximum Likelihood estimate (MLE)**
   b. **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.**
   c. **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?

   $p_{MLE} = \frac{k}{n}$
   $p_{Bayes} = \frac{k+1}{n+2}$
   $p_{MAP} = \frac{k}{n}$

   The MLE and MAP happen to give the same estimate here, while the Bayesian estimate with a uniform prior gives a slightly different estimate due to the +1

smoothing from the prior.