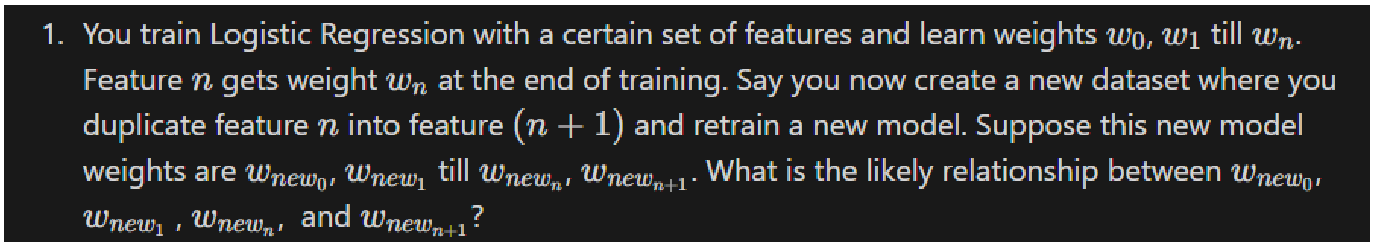
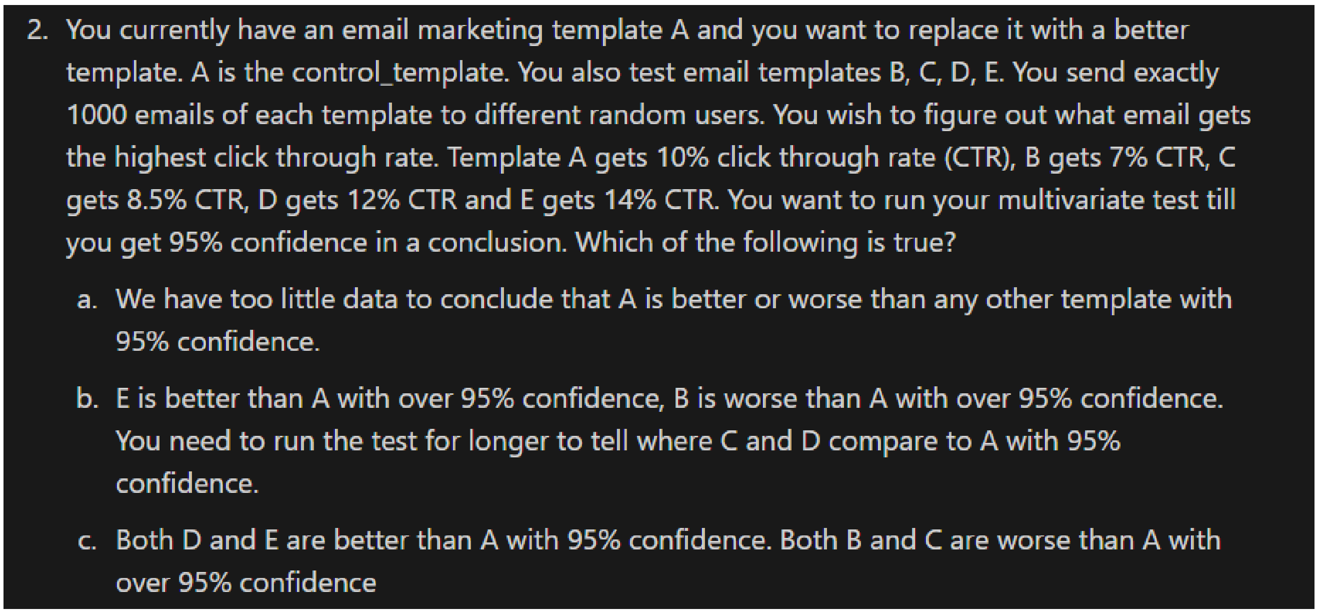


1. You train Logistic Regression with a certain set of features and learn weights w_0, w_1 till w_n. Feature n gets weight w_n at the end of training. Say you now create a new dataset where you duplicate feature n into feature $(n+1)$ and retrain a new model. Suppose this new model weights are w_{new_0}, 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 $w_{new_{n+1}}$?

Answer 1. : - In the scenario where a feature n is duplicated to create a new feature n+1 , and a logistic regression model is retrained, the weights W(new,n) and W(new,n+1) would likely adjust such that their sum is close to the original weight W(n). This is because the logistic regression model would distribute the total importance that was previously assigned to feature n across both the original and the duplicated feature.



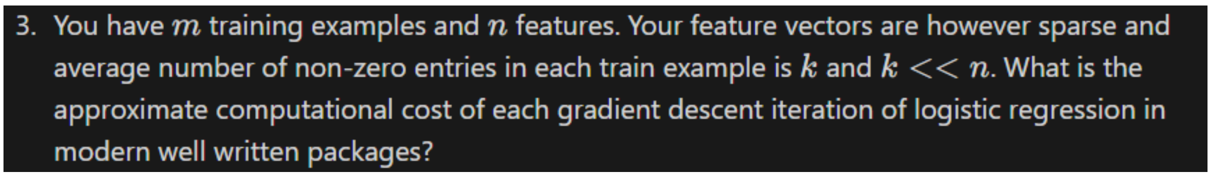
2. You currently have an email marketing template A and you want to replace it with a better template. A is the control_template. You also test email templates B, C, D, E. You send exactly 1000 emails of each template to different random users. You wish to figure out what email gets the highest click 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

Answer 2 :- a. This statement suggests that we do not have enough data to conclude if Template A is better or worse, which is not correct based on the CTRs provided. We can see that B and C are worse than A, and D and E are better than A, even without formal statistical tests.

b. This statement correctly identifies that Template E has a higher CTR than A. However, it incorrectly states that B is worse than A with over 95% confidence—while the CTR is lower, we can't claim 95% confidence without statistical testing. The statement about needing more time to compare C and D to A is reasonable since statistical significance depends on both the difference in CTR and the sample size.

c. This statement claims that both D and E are better than A with 95% confidence and that both B and C are worse than A with over 95% confidence. Without statistical testing, we cannot make these claims with a specified level of confidence.

Based on the CTRs alone, Template E appears to be the best performer, and Template B the worst. However, the assertion of 95% confidence for any comparison requires statistical testing. The most accurate option, given typical statistical standards and without seeing the actual p-values or confidence intervals from a statistical test, would likely be a version of option b.


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?

Answer 3 :-
The computational cost for a single iteration of gradient descent in logistic regression with sparse feature vectors can be calculated by considering the operations involved in computing the gradient of the loss function with respect to the weights.

Given:

- m training examples
- n features
- k average non-zero entries in each feature vector (with k << n)

Here are the steps for a single iteration of gradient descent, focusing on the computational cost:

1. Compute the Prediction:

For each training example, we compute the predicted probability. This involves a dot product between the feature vector and the weight vector. For sparse vectors, this dot product only needs to be taken over non-zero entries.

Cost per example: O(k)

Total cost for all examples: O(m.k)

2. Compute the Gradient:

The gradient of the loss function with respect to the weights is the average over all training examples of the product of the prediction error and the feature vector. Again, for sparse feature vectors, we only consider non-zero entries.

Cost per example: O(k)

Total cost for all examples: O(m.k)

3. Update the Weights:

The weights are updated by subtracting the product of the learning rate and the gradient. The update is only applied to weights corresponding to non-zero features.

Cost: O(k)

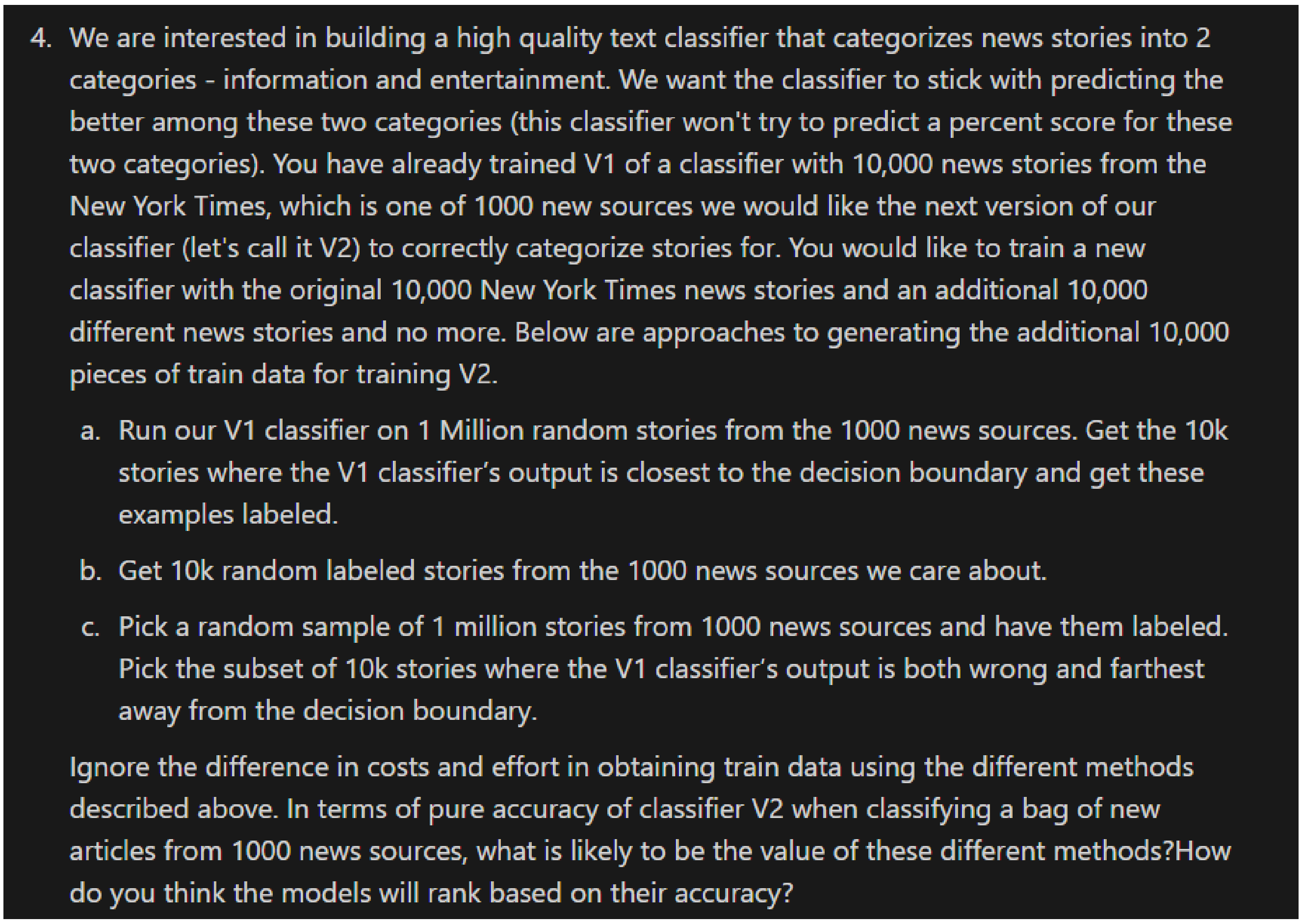
Adding these costs gives us the total computational cost for a single iteration of gradient descent:

O(m.k) (for prediction) + O(m.k) (for gradient) + O(k) (for weight update)

Since O(m.k) is the dominant term, we can approximate the total cost as:

O(m.k)

This means that the computational cost scales linearly with the number of training examples and the average number of non-zero features per example, rather than the total number of features. This is significantly more efficient for sparse datasets where k is much smaller than n.



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?

Answer 4 :- The image contains a description of three different approaches for generating additional training data for a new version (V2) of a text classifier, which has been previously trained (V1) on 10,000 news stories from the New York Times. The approaches described in the image for generating the additional 10,000 pieces of training data are:

a. Use the V1 classifier on 1 million random stories from the 1000 new sources, select the 10k stories where V1's output is closest to the decision boundary, and get these examples labeled.

b. Get 10k random labeled stories from the 1000 news sources.

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

These methods have implications for the type of data that will be used to train V2:

a. Selecting stories close to the decision boundary can be a strategy to improve the classifier's performance on cases where it is most uncertain. This could lead to a model that is better at handling ambiguous cases.

b. Randomly selecting labeled stories provides a more general and possibly more diverse set of training data. This does not specifically target the model's weaknesses but can give an overall performance boost by covering a wide range of examples.

c. Choosing stories where V1's predictions are wrong and farthest from the decision boundary aims to correct the most confident errors made by V1. This can significantly improve the accuracy of V2 by focusing on correcting the cases that V1 got wrong.

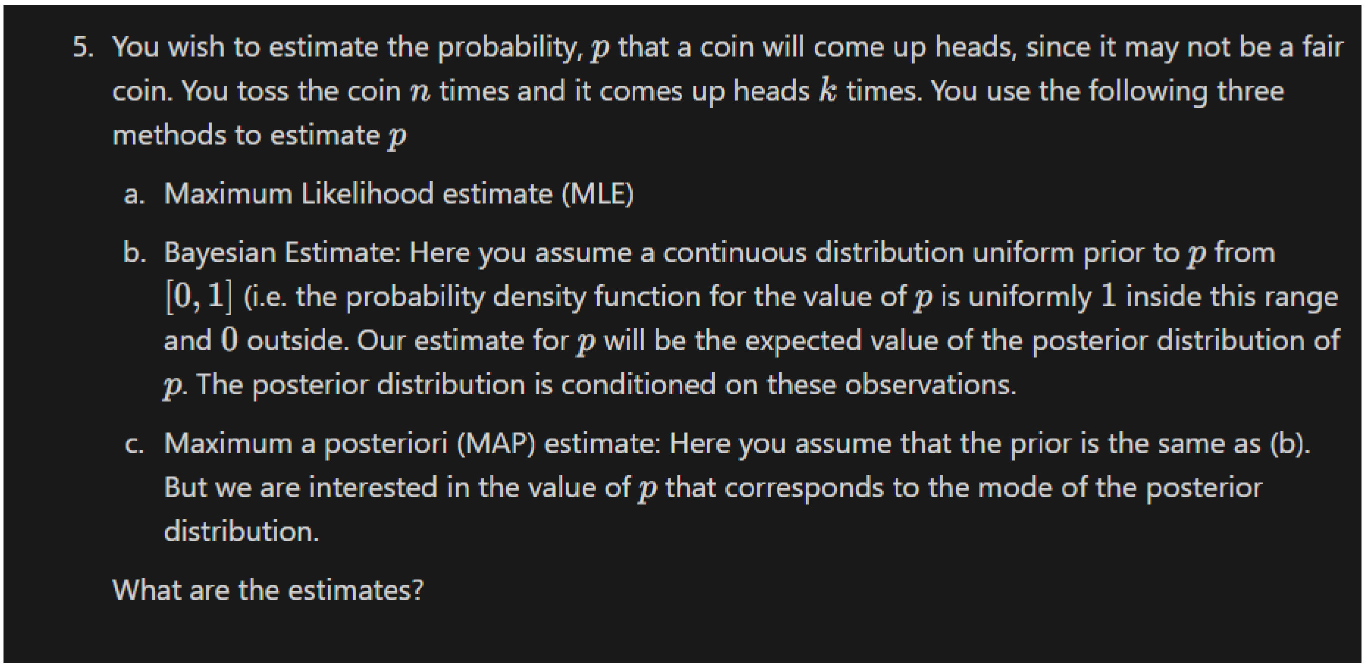
In terms of improving the pure accuracy of classifier V2, we might expect the following ranking:

1. Method c: By focusing on the stories where V1 was most confidently incorrect, V2 can learn from these mistakes. This has the potential to make the most significant improvements in accuracy, assuming these mistakes are representative of common errors that need to be corrected.

2. Method a: Training on stories close to the decision boundary will help V2 to better handle ambiguous cases, which are often the hardest for classifiers and where improvements can be most beneficial. However, this method might not address confidently wrong predictions.

3. Method b: While this method might provide a good general improvement, it does not target specific weaknesses in V1 and might include a lot of 'easy' examples that V1 already handles well.

However, it's important to note that the best method may also depend on the distribution of the data and the specific characteristics of the news stories from the 1000 sources. If the 'hardest' or most ambiguous stories are not representative of the typical data the classifier will encounter, then focusing training on these examples might not result in the highest overall accuracy.



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?

Answer 5 :-

a. Maximum Likelihood Estimate (MLE): (k/n) .

b. Bayesian Estimate (Expected value of the posterior distribution): (k+1)/(n+2)

c. Maximum a Posteriori (MAP) estimate: (k-1)/(n-2) (Note: This formula is not valid if k = 0 or k = n , as the mode of the posterior distribution is not defined in these cases.)