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_{new0}, w_{new_1} , w_{new_n}, and w_{new_n+1}?**

ANSWER:
When duplicating a feature in your dataset and retraining a logistic regression model, the associated weights for the duplicated features may undergo changes. Let's delve into an analysis of the probable relationship between the original and new weights.

Assume you initially possess a logistic regression model with weights denoted as w_0, w_1, till w_n. Following the duplication of feature n to create feature (n+1) and subsequent model retraining, the new weights might be represented as $\mathbf{w_{new\_0}}$, $\mathbf{w_{new\_1}}$ **till** $\mathbf{w_{new\_n}}$, $\mathbf{w_{new\_n+1}}$.

The interplay between the original and new weights is contingent upon various factors:

Correlation between duplicated features:

If the duplicated features exhibit high correlation, the model may distribute the weights between them.
Consequently, both $\mathbf{w_{new\_n}}$ and $\mathbf{w_{new\_n+1}}$ could be nonzero, and the combined magnitudes may closely align with $|w_n|$.
Effect of duplication on other features:

The duplication of a feature has the potential to impact the weights assigned to other features in the model.
The model might redistribute the weights to maintain equilibrium between the duplicated features and other pertinent features.
Regularization:

Logistic regression models incorporating regularization (e.g., L1 or L2 regularization) see the regularization term influencing the weights.
Regularization tends to penalize larger weights, potentially prompting the model to distribute weights more evenly, especially among correlated features.
In summary, the precise relationship between $\mathbf{w_{new\_0}}$, $\mathbf{w_{new\_1}}$, $\mathbf{w_{new\_n}}$, and $\mathbf{w_{new\_n+1}}$ hinges on data specifics, the extent of correlation among duplicated features, and the regularization techniques employed. Predicting exact values without this detailed information is challenging, but it can be anticipated that the weights will be influenced by the introduction of duplicated features, potentially resulting in a redistribution compared to the original model.

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?**
    1. We have too little data to conclude that A is better or worse than any other template with 95% confidence.
    2. 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.
    3. Both D and E are better than A with 95% confidence. Both B and C are worse than A with over 95% confidence

ANSWER:

The correct response is:

2. E is superior to A with a confidence level exceeding 95%, while B performs worse than A with a confidence level surpassing 95%. Further testing is required to determine the relative performance of C and D compared to A with a confidence level of 95%.

Explanation:

The confidence interval aids in gauging the likely range of the true population parameter (in this case, the click-through rate). However, statistical significance is also crucial in drawing conclusions.

Template A boasts a 10% click-through rate (CTR).
Template B registers a 7% CTR.
Template C achieves an 8.5% CTR.
Template D records a 12% CTR.
Template E demonstrates a 14% CTR.
Key Observations:

Comparison between A and B:

Given that B's CTR is lower than A's, and the difference is substantial, it is probable that B performs worse than A with statistical significance.
Comparison between A and E:

Considering E's higher CTR in comparison to A, and the notable difference, it is likely that E outperforms A with statistical significance.
Comparison between A and C/D:

The provided information does not specify the CTR difference between A and C/D, making it challenging to draw conclusive statements regarding the statistical significance of these differences.
In summary, option 2 accurately reflects the given information. It asserts that E surpasses A with over 95% confidence, B falls short of A with over 95% confidence, and additional testing is essential to ascertain how C and D compare to A with a confidence level of 95%.

3. **You have m training examples and n features. Your feature vectors are however sparse and the 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:
In logistic regression, the computational cost of each gradient descent iteration depends on the number of non-zero entries in each training example, denoted as k, as well as the number of features n and the number of training examples m.

The standard form of logistic regression cost function is:

$J(\theta) = -\frac{1}{m}\Sigma[y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)}))]$

Here, $h_\theta(x^{(i)})$ is the sigmoid function applied to the linear combination of the parameters θ and the feature vector $x^{(i)}$.

In modern well-written packages that leverage sparse matrix representations, the computational cost of logistic regression is often optimized to handle sparse data efficiently. Specifically, the sparse matrix representation allows the algorithm to operate only on the non-zero entries of the feature vectors, reducing the computational complexity.

The approximate computational cost of each gradient descent iteration in logistic regression can be expressed as O(k. nnz),
where:

- k is the average number of non-zero entries in each training example,
- nnz is the total number of non-zero entries in the entire dataset (sum of non-zero entries across all training examples and features).

This is an approximation, and the actual implementation details may vary between different packages. However, the use of sparse matrix representations significantly reduces the computational cost, making logistic regression with sparse data feasible even when 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.**
   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.

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:

Let's assess three approaches for generating additional training data (10,000 news stories) to build classifier V2 and explore their potential impact on model accuracy:

Approach 1:

Description: Apply the V1 classifier to 1 million random stories from 1000 news sources. Select 10,000 stories where V1's output is closest to the decision boundary and label them.
Analysis:
These examples pose challenges for V1 as they are close to the decision boundary.
Focusing on challenging instances aids V2 in learning from V1's difficulties.
This approach may enhance V2's performance on ambiguous cases.
Approach 2:

Description: Obtain 10,000 randomly labeled stories from the 1000 news sources.
Analysis:
This approach introduces labeled data without considering V1's performance.
Effectiveness depends on the randomness of selection and coverage of diverse scenarios.
It contributes to a balanced and diverse training set.
Approach 3:

Description: Label a random sample of 1 million stories from 1000 news sources. Select 10,000 stories where V1's output is both wrong and farthest from the decision boundary.
Analysis:
Focuses on cases where V1 is incorrect, providing training examples far from the decision boundary.
Addresses situations where V1 is confidently wrong, aiding V2 in correcting errors.
May improve V2's performance on instances where V1 was overly confident but incorrect.
Conclusion:
Ranking based on potential impact:

Approach 3: Focusing on cases where V1 is both wrong and far from the decision boundary may substantially impact V2's accuracy.

Approach 1: Selecting examples close to the decision boundary could enhance accuracy on ambiguous cases.
Approach 2: Randomly selecting labeled stories contributes to diversity but may not specifically target challenging cases.
Overall:
Combining these approaches is beneficial, as each addresses different aspects of the classification problem. A well-rounded training set, encompassing challenging examples and instances where V1 struggled, is likely to contribute to the overall accuracy of the V2 classifier.

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**
   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?

ANSWER:
Let's derive the estimates for the probability p using the three methods:

1. Maximum Likelihood Estimate (MLE):
The Maximum Likelihood Estimate for p is obtained by maximizing the likelihood function, which is the probability of observing the given data under a certain value of p. For a binomial distribution (coin tosses), the MLE for p is given by:

$\hat{p}MLE = \frac{k}{n}$

2. Bayesian Estimate:
With a continuous uniform prior on p from 0 to 1, and assuming a binomial likelihood, the posterior distribution is a Beta distribution. The Bayesian Estimate is the expected value (mean) of the posterior distribution:

$$\hat{p}Bayesian = \frac{k+1}{n+2}$$

3. Maximum a Posteriori (MAP) Estimate:
The MAP Estimate is the mode of the posterior distribution. In the case of a continuous uniform prior and binomial likelihood, the posterior distribution is again a Beta distribution. The mode of a Beta distribution with parameters

$\hat{p}MAP = \frac{k}{n}$

Summary:
Maximum Likelihood Estimate (MLE):
$\hat{p}MLE = \frac{k}{n}$
Bayesian Estimate:

$$\hat{p}Bayesian = \frac{k+1}{n+2}$$

Maximum a Posteriori (MAP) Estimate:
$\hat{p}MAP = \frac{k}{n}$

These are the estimates for the probability p using the Maximum Likelihood, Bayesian, and Maximum a Posteriori methods, based on the given assumptions and prior distribution.