Q1.You train Logistic Regression with a certain set of features and learn weights W0. W1 till Wn. Feature n gets weight Wn 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 Wnew0, Wnew1 till Wnewn, Wnewn+1. What is the likely relationship between Wnew0 Wnew1, Wnew, and Wnewn+1?

Answer:

When duplicating a feature (n) and retraining a Logistic Regression model, the relationship between the new weights (Wnew) is as follows:

1. Weights for Duplicated Features (Wn and Wn+1):  
Since features n and (n+1) contain identical information, it is expected that the model will assign them very similar weights. It is also possible that these weights are identical (Wnewn = Wnewn+1).

2. Weights for Original Features (Wnew0 and Wnew1):   
The weights for the original features x1 and x2 (Wnew0 and Wnew1) are likely to differ from the original weights (W0 and W1). This is due to a new weight being introduce thus creating repetition. To adjust this new weight all the other weights would be changed. Thus Wnew0 and Wnew1 would be different from W0 and W1.

3. Factors Influencing the Relationship:

Data Distribution:

The distribution of features and the target variable can influence how the model adjusts weights for the duplicated feature. If the original dataset already contained duplicate features the model may learn different weights for feature n and n+1 since it had already faced duplicate features. In this case Wnew0 and Wnew1 may have different value. And there might not be much change in Wnew0 and Wnew1.

Model Complexity:

More complex models have greater flexibility in adjusting weights, potentially resulting in smaller differences between Wnew0 and Wnew1. It may also be able to learn different weights for Wnewn and Wnewn+1.

In summary, the weights for duplicated features i.e. Wnewn and Wnewn+1 are likely to be very similar and the weights Wnew0 and Wnew1 are expected to be different.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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?

* 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:

In a multivariate test, we are comparing the performance of multiple versions to determine which one performs the best. The click-through rate (CTR) is the key metric used to evaluate the effectiveness of each template.  
Here are the given click-through rates for each template:

Template A: 10%

Template B: 7%

Template C: 8.5%

Template D: 12%

Template E: 14%

We are using Z-score for statistical significance i.e for confidence.

The formula for Z-score is:

Z= (p1-p2)/[p(1-p)(1/n1+1/n2)]^½  
Where:  
p1 is the CTR in each group (templates B, C, D, E).  
p2 is the CTR of standard template i.e template A.

p=(1000⋅p1)+(1000⋅p2)+(1000⋅p3)+(1000⋅p4)+(1000⋅p5)/5000  
We get p=0.103  
n1 and n2 is the sample size of each template

For having confidence 95% the Z score should be 1.96.  
  
We the the following Z-score:  
Template B:

Z-score(ZB)=-2.20(approx)

Template C:

Z-score(ZC)=-1.10(approx)

Template D:

Z-score(ZD)=+1.47(approx)

Template E:

Z-score(ZE)=+2.9(approx)

As we can see that Z-score for Template B and E is greater than 1.96 therefore we can say with 95% confidence that:

Template B is worse than Template A

Template E is better than Template A   
  
Therefore

Statement a. Is false:

As we can say with 95% confidence that Template B is worse and Template E is better though we cannot say for Template C and D.

Statement b. is true:

As we can say that Template E is better than A with over 95% confidence, B is worse than A with over 95% confidence. But we need to run the test for longer to tell whether C and D compare to A with 95% confidence.

Statement c. is false:

As we cannot say with 95% confidence whether Template C is worse than A and if Template D is better 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?

Answer:  
In a sparse implementation of logistic regression, which efficiently handles sparse feature vectors, specialized data structures and optimization techniques, such as coordinate descent and stochastic gradient descent, are employed. This results in a substantial reduction in computational cost per iteration compared to dense implementations. The time complexity per iteration is approximately O(m \* k), where m is the number of training examples, and k is the average number of non-zero entries in each example.  
  
Modern machine learning libraries further enhance efficiency through parallelization and optimizations. These include leveraging specialized hardware, employing parallel processing for gradient computations, and using efficient sparse matrix operations. The outcome is a streamlined logistic regression implementation that scales effectively, making it well-suited for real-world applications with large and sparse datasets.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

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:  
  
Certainly! Let's provide a comprehensive summary:

To build a high-quality text classifier (V2) for categorizing news stories into "information" and "entertainment," three different approaches are considered for generating an additional 10,000 training examples. The goal is to compare these approaches in terms of their potential impact on V2's accuracy when classifying a diverse set of news articles from 1000 different sources.

Approach A involves running the V1 classifier on 1 million random stories and selecting the 10,000 examples closest to the decision boundary, focusing on uncertainties. This approach aims to identify cases where V1 struggles and is unsure about the correct classification.

Approach B focuses on obtaining 10,000 randomly labeled stories from the 1000 news sources. The goal is to introduce diversity and guaranteed labels to the training data, potentially mitigating biases and improving generalizability.

Approach C includes selecting 10,000 stories from a random sample of 1 million and labeling them. The focus here is on V1's mistakes by choosing stories where it was wrong and farthest from the decision boundary, aiming to address specific weaknesses in V1.

Each approach has its pros and cons. Approach A emphasizes uncertain cases and may reduce bias but could be limited by V1's existing biases. Approach B introduces diverse examples with guaranteed labels, potentially reducing bias and improving generalization. Approach C targets V1's mistakes, addressing specific weaknesses but requiring additional labeling effort.

The suggested best strategy depends on the project's goals:

1. Reducing Overall Bias and Generalizability:

- Best Approach: B (Randomly labeled stories from all sources) Introduces diversity and guaranteed labels, reducing reliance on V1's biases and potentially improving overall performance.

2. Rectifying V1's Weaknesses and Boosting Accuracy in Specific Areas:

- Best Approach: C (V1's misclassified stories furthest from the decision boundary)

Focuses on learning from V1's mistakes and challenging cases, potentially leading to significant improvements in areas where V1 struggles.

3. Balance between Bias Mitigation and Addressing Weaknesses: - Best Approach: A (V1's uncertain cases closest to the decision boundary)

Balances exploring diverse examples with focusing on areas where V1 is unsure, mitigating bias while providing targeted learning opportunities.

Combining these approaches, using B for general improvement and bias reduction, C for targeted improvement in specific areas where V1 struggles, and A for a balanced approach, provides a comprehensive strategy that considers both generalization and targeted enhancement.

It's crucial to experiment with these approaches and evaluate their performance on a held-out test set to make informed decisions and optimize the performance of the V2 text classifier. Real-world results may vary based on specific data characteristics and training details.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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?  
  
Answer:

Maximum Likelihood Estimate (MLE):

This straightforward method calculates the proportion of heads (k) to the total number of tosses (n), providing a simple ratio (n/k) as the estimate.

Bayesian Estimate:

Incorporating a prior belief about

p, this method assumes a uniform prior distribution. It calculates the posterior distribution by multiplying the likelihood function (based on observed data) with the prior. The expected value of the posterior distribution yields the Bayesian estimate (k+1/n+2).

Maximum a Posteriori (MAP) Estimate: Similar to the Bayesian estimate, it assumes a uniform prior but focuses on the mode of the posterior distribution as the estimate. In cases with a uniform prior, the MAP estimate coincides with the MLE (k/n).

The choice among these methods depends on the availability of prior information. If there is no prior knowledge, MLE is suitable. However, if there is prior information, Bayesian or MAP estimates may be more appropriate.

Suppose you toss a coin 20 times, and it comes up heads 14 times (n=20,k=14).

Maximum Likelihood Estimate (MLE):

MLE= n/k

MLE=14/20=0.7

Bayesian Estimate:

Assuming a uniform prior, the Bayesian estimate is calculated as:

Bayesian Estimate= n+2/k+1

​ = 20+2/14+1

​ = 22/15

≈0.6818

Maximum a Posteriori (MAP) Estimate:

In this case, with a uniform prior, the MAP estimate is the same as the MLE:

MAP Estimate= n/k

​ = 20/14

=0.7

In this example, the estimates differ slightly due to the Bayesian approach incorporating a prior belief. The MLE and MAP estimates align when a uniform prior is assumed.

This illustrates how different values for

n and k can lead to varied probability estimates using these methods. The choice between MLE and Bayesian/MAP depends on the context and available prior information.