Juhi Singh

Problem 1.

Solution.

1. Since feature n and feature $n+1$ are identical, the model will not be able to distinguish between them. As a result, the weight that was initially given to feature n in the original model (weight w_n) will be divided between the duplicated features n and $n+1$ in the new model. This implies that $W_{\text{new } n}$ and $W_{\text{new } n+1}$ will be smaller individually compared to the original weight w_n, but their combined effect should be approximately similar to the original weight of feature n.Addition of a duplicate feature doesn't provide new information to the model; it only redistributes the weight that the model assigns to the duplicated feature.Hence, the weights of other features ($W_{\text{new } 0}, W_{\text{new } 1}, \ldots, W_{\text{new } n-1}$) will not be affected by this change.

2. Possible Model Instability: Duplicating a feature can lead to multicollinearity, which might cause instability in the model's estimation process. This can make the model's output sensitive to changes in the model specification or in the sample data.

□

Problem 2. *Write here the text of the second homework problem.*

Solution. 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. □

Problem 3.

Solution. The computational cost of each gradient descent iteration in logistic regression largel depends on the number of features, the number of training examples, and the sparsit) of the feature vectors. In a situation where the feature vectors are sparse, as mentione in your question, this sparsity can significantly affect the computational cost.

In logistic regression, the cost function is typically defined as:

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

where: - m is the number of training examples. - $x^{(i)}$ is the feature vector of the i-th training example. - $y^{(i)}$ is the label of the i-th training example. - $h_\theta(x)$ is the hypothesis function, $h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$.

The gradient of the cost function with respect to the parameter vector θ is:

$$\nabla J(\theta)=\frac{1}{m} \sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right) x^{(i)}$$

In a well-written logistic regression package that takes advantage of sparse matrices, the computational cost of each gradient descent iteration is proportional to the number of non-zero entries in the feature vectors, rather than the total number of features. If the average number of non-zero entries per training example is s (which seems to be missing in your question), and there are m training examples, then the computational cost for each iteration is approximately proportional to $m \times s$.

In summary, the computational cost of each gradient descent iteration in logistic regression, especially in modern packages that utilize sparse matrix operations, is approximately $O(m \times s)$, where m is the number of training examples and s is the average number of non-zero entries in each feature vector. This is significantly lower than $O(m \times n)$ which would be the cost if all entries were non-zero, with n being the total number of features.

□

Problem 4.

Solution.

- Approach A : The key advantage of this approach is that it targets the ambiguous cases where V1 is most uncertain. If we train on these examples, V2 could potentially become more adept at handling similar cases. But on the other hand, it might also lead to a model that is overly tuned to these edge cases, compromising on the overall accuracy.

- Approach B : This method offers a broad range of examples that cover a wide spectrum of the data distribution and hence, the model can be expected to generalize well across different types of news. This approach ensures that V2 is exposed to a representative sample of the real-world data it will encounter.

- Approach C: This method is designed in a way that corrects the most significant errors of V1. However, this might also lead to a model that is overly tuned to specific types of errors.

Approach B is likely to produce a model (V2) with the best generalization ability since it exposes the model to a wide variety of examples. Approach A and C are focused on specific types of errors (ambiguous cases and most significant errors, respectively). While they might improve performance in these areas, they might not contribute as much to general accuracy. Approach B is likely to result in a more robust model as it is not overly tuned to specific types of errors or uncertainties.

Ranking Based on Potential for Accuracy Improvement:
Approach B > Approach A > Approach C □

Problem 5.

Solution.

1. Maximum Likelihood Estimate (MLE): Since this is a binomial distribution, MLE of p = proportion of heads observed = $\frac{k}{n}$.

2. Bayesian Estimate: In this approach, we assume a uniform prior distribution for p from $[0, 1]$. After observing k heads in n tosses, the likelihood function is a binomial distribution and hence, the posterior distribution for p, is a Beta distribution, $\beta(k + 1, n - k + 1)$. The expected value of the Bayesian estimate of $p = \frac{k+1}{n+2}$.

3. Maximum a Posteriori (MAP) Estimate: Assuming the same uniform prior distribution, MAP estimate is the mode of the posterior distribution. For a $\beta(k+1, n-k+1)$ distribution, mode $= \frac{k}{n+1}$ (except for $k = 0$ or $k = n$, where it is 0 or 1 respectively)

$\square$