**70. Document Classification in Warehouse Logistics**

Let's move on to a new problem within **warehouse logistics optimization**, which is **document classification**. Document classification can be highly valuable in logistics management for sorting through vast amounts of text data, such as delivery reports, incident logs, maintenance records, and customer feedback. For this example, I'll use a dataset that contains **incident reports** in a warehouse environment, where reports are labeled based on their content—whether they describe a "maintenance issue," "inventory discrepancy," "safety concern," or "delivery problem."

In a typical logistics operation, managers and staff generate various documents, such as incident logs or maintenance records. Each document, or "report," is a short document that describes a specific event or problem in the warehouse. These reports can be either positive (e.g., "all items accounted for") or negative (e.g., "inventory mismatch detected"). In this scenario, a labeled dataset might consist of several thousand reports that have been manually categorized into various classes such as "urgent maintenance," "inventory adjustment needed," "safety hazard," or "normal operations." The challenge is to automatically classify these reports to help prioritize actions.

Document classification might sound like an easy task, but the language used in these reports can be nuanced. For example, a report that states "forklift making strange noises but still operable" might initially seem non-urgent, but it actually implies a potential maintenance issue that needs to be addressed soon. This subtlety in language is what makes document classification both challenging and essential for effective warehouse operations.

Consider an example of a report: "Forklift malfunctioned halfway through unloading and caused a minor delay." This statement clearly indicates an equipment issue and a delay in operations. However, there may be other subtle clues in different reports that require a more sophisticated approach to classification.

**Representing Warehouse Incident Reports for Classification**

The primary question is: how do I represent a report, which is a free-form document, in a way that can be processed by a machine learning algorithm? The goal is to convert these text reports into a format that can be used to automatically classify the type of issue described.

One popular method for transforming documents into a usable format is called the **bag of words model**. In warehouse logistics, reports can vary in length and consist of sequences of words describing different incidents or issues. To create a standardized feature set, I might define a dictionary of 10,000 commonly used terms relevant to warehouse operations (e.g., "forklift," "mismatch," "delay," "leak," "injury," etc.). This number (10,000) is a parameter that I can adjust based on the data's complexity.

For each report, I create a binary vector of length 10,000 where each position corresponds to a word in the dictionary. If a word from the dictionary appears in the report, I mark its position with a 1. If a word appears multiple times, it still only gets a 1 in this basic model. Thus, for nnn documents (e.g., 5,000 incident reports), I end up with an n×pn \times pn×p sparse feature matrix, XXX, where p=10,000p = 10,000p=10,000. Most of the values in this matrix will be 0 because each report will only contain a small subset of the total possible words.

**Comparing Logistic Regression and Neural Networks for Document Classification**

To classify these warehouse reports, I can compare two different models: **lasso logistic regression** and a **two-hidden-layer neural network**. Both models can handle sparse input matrices, but they differ in their approach to capturing the underlying patterns in the text.

First, I apply **lasso logistic regression** using the **glmnet** package in R. This method is efficient for sparse matrices and performs well with high-dimensional data, such as text. The plot below shows the model's performance as a function of the regularization parameter, λ\lambdaλ, which controls the model's complexity:

* **Training Accuracy:** The accuracy on the training data as the model is trained.
* **Validation Accuracy:** Accuracy on a small subset of the data set aside for validation, which helps decide when to stop regularization.
* **Test Accuracy:** Accuracy on a separate test dataset that evaluates how well the model generalizes.

In the plot, the **validation accuracy** and **test accuracy** are closely aligned, which is a good indication that the model is not overfitting. The **lasso** model achieves a high accuracy of around 90%, indicating it can classify the warehouse reports effectively.

Next, I consider a **two-hidden-layer neural network**. This network doesn't use convolutions but instead applies dense layers to learn patterns in the text data. For the neural network, I track **training accuracy**, **validation accuracy**, and **test accuracy** as a function of the number of **epochs**. An epoch refers to one complete pass through the entire training dataset during the optimization process (e.g., **gradient descent**).

As I increase the number of epochs, the **training accuracy** initially improves, similar to lasso. However, both **validation accuracy** and **test accuracy** tend to plateau or decrease after a certain point, indicating overfitting. The performance of the neural network is quite similar to that of the lasso logistic regression, though the neural network might capture some non-linear relationships between words that a linear model like lasso cannot.

**Why Do Neural Networks Perform Similarly to Lasso?**

At first glance, one might expect the neural network to perform significantly better because it can capture non-linearities and interactions between terms. For instance, it could capture interactions like "forklift + leak" suggesting a particular kind of maintenance issue. However, in this case, both models perform similarly. Why?

1. **Feature Sparsity:** The text data's sparsity means that even complex models like neural networks may not gain much over simpler models if the signal is already clear from the unigram features.
2. **Overfitting:** Both models show signs of overfitting as the complexity increases, indicating that the additional parameters in the neural network might not provide significant advantages without more data or additional regularization techniques.
3. **Training Speed:** The lasso model, implemented with **glmnet**, is highly efficient in handling sparse matrices, making it faster than the neural network, which has to go through multiple training epochs.

**Conclusion**

In this chapter, I’ve shown that both **lasso logistic regression** and **neural networks** can be effective tools for classifying documents in a warehouse logistics setting, such as incident reports or maintenance logs. While neural networks offer the ability to capture complex relationships between words, simpler models like lasso can perform just as well, especially when the data is sparse and the relationships are mostly linear.

When considering which model to use, factors like interpretability, speed, and the specific characteristics of the data should guide the decision. In many warehouse logistics applications, where quick insights are needed, and data is plentiful but sparse, logistic regression models might provide a good balance between performance and efficiency.