**68. Neural Networks in Warehouse Logistics Optimization**

Welcome back to the course. In this series of sessions, I will cover **deep learning**, an advanced technique in machine learning that has significantly impacted various fields, including **warehouse logistics optimization**. This topic is part of the new chapters in the second edition of "Introduction to Statistical Learning." Deep learning, which is essentially a modern evolution of neural networks popularized in the 1980s, has experienced a resurgence since the 2010s and has become highly influential in optimizing logistical operations today.

Back in the 1980s and 1990s, there were many successes and a lot of hype around neural networks. Researchers like Rob and I were regular attendees of conferences like NeurIPS (originally known as the Neural Information Processing Systems conference) held in ski resorts. We would discuss the latest developments in neural networks, while also getting some skiing in. However, with the advent of methods like support vector machines, random forests, and boosting—which you've either learned about in this course or will soon learn about—neural networks took a back seat. They re-emerged around 2010 under the name **deep learning**. By the 2020s, deep learning became dominant, widely used across industries for complex optimization problems, including those in warehouse management and logistics.

Part of the resurgence was due to vast improvements in computing power, much larger training datasets, and better software. The main software platforms include **TensorFlow** from Google and **PyTorch** from Facebook. Much of the credit goes to three pioneers and their students: Yann LeCun, Geoffrey Hinton, and Yoshua Bengio, who received the 2019 ACM Turing Award for their groundbreaking work on neural networks. On our course website, Trevor and I have an interview with Geoff Hinton discussing the history of neural networks and their development since the 1980s.

**Single-Layer Neural Network for Logistics Optimization**

Let's begin with a simple example: a **single-layer neural network**, sometimes called a **feedforward neural network**. Neural networks are often visualized using **network diagrams**. In warehouse logistics, these networks can be used to predict outcomes like demand forecasting, inventory optimization, or equipment failure. In the network diagram, the input layer could consist of variables such as daily order volume, product type, weather conditions, or equipment status. Here, we see an example where we have four input variables (such as order volume, inventory level, delivery time, and equipment health), a hidden layer with five units, and an output layer that predicts the optimal action, such as “replenish stock,” “schedule maintenance,” or “prioritize delivery.”

The hidden layer represents transformations of the inputs. For instance, each hidden unit could represent a complex combination of input features that help predict the output action more accurately. These transformations, or **activations**, are computed using linear combinations of the inputs followed by a non-linear function. The arrows in the diagram represent the weights of these linear combinations, which are learned during training to optimize decision-making in the logistics network.

**Understanding Activations and Activation Functions in Logistics Models**

Activations, denoted by aka\_kak​, are essentially non-linear transformations of a linear function of the inputs. They are known as **activation functions**, and in the context of logistics, they help capture non-linear relationships between different variables. Two popular choices for activation functions are the **sigmoid** and **Rectified Linear Unit (ReLU)**. In the early neural networks, the sigmoid function was popular. It smoothly maps the input range to an interval between 0 and 1, which could represent probabilities such as the likelihood of stockouts or equipment failures.

Today, the ReLU function is more popular. It is simple—it returns zero for negative inputs and a linear value for positive inputs. This property makes it highly effective for large-scale optimization problems in logistics, where relationships between variables are often piecewise linear rather than smoothly curved. Non-linearity is crucial because if the activation functions were purely linear, the entire network would reduce to a simple linear model, which might not capture the complexity of logistical decision-making.

**Fitting Neural Networks in Logistics Using Regression Loss**

Once the network is structured, I fit the model by minimizing a loss function. For regression-based logistics problems, such as predicting delivery times or inventory levels, I might use a squared error loss function, minimizing the sum of squared differences between predicted and actual values. This loss function fff encompasses many parameters, including weights from both the hidden layer to the output layer and from the inputs to the hidden units. These weights (or parameters) are crucial to the predictive power of the model.

**Example: Predicting Warehouse Demand with Neural Networks**

Let's move on to a more practical example. Consider a problem in warehouse logistics where I want to predict **future demand** for various products based on historical data, seasonal trends, and promotional activities. Similar to how neural networks are applied in digit recognition tasks, I can use a two-layer feedforward neural network with multiple hidden units for demand forecasting.

For this example, I use a neural network with two hidden layers: the first with 256 units and the second with 128 units. The input features could be time-series data such as daily sales, inventory levels, promotions, supplier lead times, and external factors like weather. The output layer could have units representing different product categories, each with a predicted demand level.

One might wonder how such a network avoids overfitting, given the number of parameters involved. For instance, with 60,000 daily observations and 235,000 weights, it might seem prone to overfitting. However, techniques such as **regularization** and **dropout** (which I will cover later) help mitigate overfitting by penalizing overly complex models and improving generalization.

**Network Architecture for Predicting Demand**

The network architecture consists of an input layer (all the relevant logistical variables), the first hidden layer with 256 units, a second hidden layer with 128 units, and the output layer with several units representing different logistical decisions or demand predictions. Each arrow represents a weight that needs to be learned, connecting every input to every hidden unit and so on.

The output layer uses a **softmax function**, which is common in multi-class classification problems. For example, each output unit could represent the probability of high, medium, or low demand for a product category. The softmax function transforms the network's linear outputs into probabilities that sum to one, making it easier to interpret the predictions and decide which logistical action to take.

**Fitting the Neural Network: The Softmax Function and Cross-Entropy Loss**

The output layer uses the softmax activation to compute probabilities. This approach is similar to **multi-class logistic regression**, where each product category is associated with a probability. To fit the model, I minimize the **negative multinomial log-likelihood**, also known as **cross-entropy loss** in the field of deep learning. This loss function is essential to optimize because it focuses on the most probable outcomes, aligning with the needs of a warehouse logistics system that must prioritize the most critical actions.

**Results and Performance on Warehouse Data**

After fitting the model, I evaluate its performance on the logistics data. For instance, the neural network might achieve 2.3% prediction errors, compared to 7.2% for multinomial logistic regression or 12.7% for linear discriminant analysis (LDA). These results demonstrate substantial improvements in predictive accuracy, which can translate to better inventory management, reduced stockouts, and optimized warehouse operations.

While these results are promising, I should note that warehouse logistics problems can vary widely. The model's performance depends on the specific context and data. Advanced techniques like **ridge regularization** and **dropout regularization** are critical to preventing overfitting and enhancing the model's ability to generalize to new situations.

**Conclusion**

The application of neural networks in warehouse logistics can significantly enhance predictive capabilities for demand forecasting, inventory management, and operational efficiency. While neural networks may have seemed overly complex or prone to overfitting in the past, modern techniques have made them invaluable tools in optimizing complex supply chain processes. With further exploration and training, deep learning can become a cornerstone of advanced logistics strategies.