Learning Path Optimization Using Regression & Neural Networks

IE402 - Optimization

By

Kevin Menpara (202101079)
Krisha Brahmbhatt (202201164)
Dipesh Verma (202201126)
Visvas Solanki (202101138)
Pooja Yogi (202321012)

1 Introduction

In today's competitive hiring landscape, companies are increasingly focusing on attracting the best talent with skills that align directly with their needs. However, the complexity of interview processes, where companies often repeatedly test candidates on similar topic areas within Data Structures & Algorithms (DSA) and Core Computer Science (CS) subjects, poses a challenge. This project addresses this challenge by optimizing a learning path for students preparing for company interviews, focusing on the most relevant topics to improve their interview performance.

Let $C = \{c_1, c_2, ..., c_n\}$ represent a set of companies, each with unique interview requirements based on topic areas, difficulty, and frequency of occurrence in past interviews. The goal of this project is to develop an optimal learning path $L = \{l_1, l_2, ..., l_k\}$ for students, where each l_i is a topic area that maximizes the likelihood of success $S(L)$ in interviews with specific companies. For each company $c_i \in C$, historical interview data includes the frequency, difficulty, and recency of questions asked from various topics, forming a data vector X. The problem is then formulated as:

$$S(L) = f(X)$$

Where f is a predictive model that links the candidate's preparedness (based on past data) to their likelihood of interview success. By leveraging machine learning models such as linear regression and neural networks, we aim to generate a learning path that is either generalized for multiple companies or tailored to a specific company's interview pattern.

2 Problem Statement

Given a set of companies $C = \{c_1, c_2, ..., c_n\}$, each with associated interview data $D = \{d_1, d_2, ..., d_n\}$, where each d_i includes features such as topic frequency F, difficulty D, and recency R in past interview questions, the objective is to develop a predictive model that optimizes the learning path for candidates. Specifically, for a target company c_j, we aim to find a learning path L^* that maximizes the success score $S(L)$.

Formally, let $X = [F, D, R]$ denote the candidate feature set derived from data D. The goal is to identify the optimal sequence of topics L that maximizes the likelihood of success in interviews:

$$S(L) = f(X)$$

Where f is a predictive model trained on historical interview data from various companies. The model will predict the most relevant topics based on their frequency, difficulty, and recency, thus guiding the student towards an optimal learning path.

3 Algorithm

The algorithm trains a neural network to predict the relevance of topics based on three features: Frequency, Difficulty, and Recency. First, it normalizes the feature values to a range between 0 and 1 and encodes the topic labels into numerical values. The neural network consists of two hidden layers with ReLU activations and an output layer that produces the predicted relevance. During training, the algorithm performs a forward pass to calculate the network's predictions, then computes the loss using Mean Squared Error (MSE), and updates the weights and biases by applying backpropagation. This process is repeated for several epochs to minimize the loss.

Require: A dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i = [x_{i1}, x_{i2}, x_{i3}]$ represents the features $(\text{Frequency}, \text{Difficulty}, \text{Recency})$ of the i-th sample, and y_i is the corresponding label (topic relevance).

Algorithm 1 Topic Relevance Prediction Using Neural Network

Initialize:

- Normalize the features $x_i = [x_{i1}, x_{i2}, x_{i3}]$ for each sample i using `Normalize(x)`.
- Encode the categorical variable 'Topic' into numerical values using `LabelEncode(y)`.
- Initialize the weights W_1, W_2, W_3 and biases b_1, b_2, b_3 for the neural network model randomly.

Main Loop:

1. **Training:** For each epoch $t = 1, 2, \ldots, T$:
   - For each sample $i = 1, 2, \ldots, N$:
     (a) Compute the input vector $x_i = [x_{i1}, x_{i2}, x_{i3}]$.
     (b) Compute the output of the first hidden layer $h_1 = \texttt{ReLU}(W_1 \cdot x_i + b_1)$.
     (c) Compute the output of the second hidden layer $h_2 = \texttt{ReLU}(W_2 \cdot h_1 + b_2)$.
     (d) Compute the predicted output $\hat{y}_i = W_3 \cdot h_2 + b_3$.
   - Compute the Mean Squared Error (MSE) loss using `ComputeMSE(`$\hat{y}_i$`,`y_i`)`.
   - Perform backpropagation to compute the gradients using `Backpropagate(`L`,` $W_1, W_2, W_3, b_1, b_2, b_3$`)`.
   - Update the weights and biases using the optimizer `UpdateWeights(`$W_1, W_2, W_3, b_1, b_2, b_3$`)`.
2. **Prediction:** After training, for each test sample x_{test}:
   - Normalize the input features $x_{\text{test}} = [x_1, x_2, x_3]$ using the same MinMaxScaler.
   - Compute the hidden layers outputs:

$$h_1 = \texttt{ReLU}(W_1 \cdot x_{\text{test}} + b_1)$$

$$h_2 = \texttt{ReLU}(W_2 \cdot h_1 + b_2)$$

   - Compute the predicted output $\hat{y}_{\text{test}} = W_3 \cdot h_2 + b_3$.

4 Mathematical Formulation

The model described in the previous section can be mathematically formalized as follows, considering its neural network architecture and training process.

4.1 Data Preprocessing

1. Normalization of Features: The numerical features (Frequency, Difficulty, Recency) are normalized to the range $[0, 1]$ using the MinMaxScaler:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where x represents each feature value (Frequency, Difficulty, Recency) and $\min(x)$ and $\max(x)$ are the minimum and maximum values of the feature. This operation is performed by the function `Normalize(x)`.

2. Label Encoding for Topic: The categorical feature 'Topic' is label encoded into numerical values. If $T = \{t_1, t_2, \ldots, t_k\}$ represents the set of unique topics, the label encoder maps each topic t_i to an integer y_i. This operation is performed by the function `LabelEncode(y)`.

4.2 Model Definition

The core neural network model consists of three layers:

Input Layer: The input consists of three features: Frequency (x_1), Difficulty (x_2), and Recency (x_3). Thus, the input vector is:

$$x = [x_1, x_2, x_3]$$

Hidden Layers:

- **First Hidden Layer:** The first hidden layer contains 64 neurons. The outputs of the first hidden layer h_1 can be represented as:

  $$h_1 = \text{ReLU}(W_1 \cdot x + b_1)$$

  where W_1 is the weight matrix of the first hidden layer, b_1 is the bias term, and ReLU is the activation function:

  $$\text{ReLU}(z) = \max(0, z)$$

  This is implemented by the function `ReLU`(z).

- **Second Hidden Layer:** The second hidden layer contains 32 neurons. The outputs of the second hidden layer h_2 are:

  $$h_2 = \text{ReLU}(W_2 \cdot h_1 + b_2)$$

  where W_2 is the weight matrix of the second hidden layer and b_2 is the bias term.

Output Layer: The output layer consists of a single neuron, and the predicted output $\hat{y}$ is computed as:

$$\hat{y} = W_3 \cdot h_2 + b_3$$

where W_3 is the weight matrix of the output layer and b_3 is the bias term. This operation is done by the function `OutputLayer`(h_2).

$$\hat{y} = W_3 \cdot \text{ReLU}(W_2 \cdot \text{ReLU}(W_1 \cdot x + b_1) + b_2) + b_3$$

4.3 Loss Function

The model is trained using **Mean Squared Error (MSE)** as the loss function. The MSE loss for a dataset of N samples is given by:

$$L = \frac{1}{N} \sum_{i=1}^{N} (\hat{y_i} - y_i)^2$$

where:

- N is the number of samples,
- $\hat{y_i}$ is the predicted value for the i-th sample,
- y_i is the actual value for the i-th sample.

This is computed by the function `ComputeMSE(`$\hat{y_i}$`, `y_i`)`.

4.4 Backpropagation

Let the loss function be L, and let the parameters of the network be W_1, W_2, W_3 and b_1, b_2, b_3. The gradients are computed as follows:

1. Gradient of the Loss with respect to the Output Layer: The output of the network is $\hat{y} = W_3 \cdot h_2 + b_3$. The gradient of the loss with respect to the output layer parameters is computed using the chain rule:

$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_3} = 2(\hat{y} - y) \cdot h_2^T$$

$$\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

2. Gradient of the Loss with respect to the Second Hidden Layer: The second hidden layer output is $h_2 = \text{ReLU}(W_2 \cdot h_1 + b_2)$. The gradient of the loss with respect to W_2 and b_2 is:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial h_2} \cdot \frac{\partial h_2}{\partial W_2} = \frac{\partial L}{\partial \hat{y}} \cdot W_3 \cdot \text{ReLU'}(W_2 \cdot h_1 + b_2) \cdot h_1^T$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial h_2} \cdot \text{ReLU'}(W_2 \cdot h_1 + b_2)$$

3. Gradient of the Loss with respect to the First Hidden Layer: The first hidden layer output is $h_1 = \text{ReLU}(W_1 \cdot x + b_1)$. The gradient of the loss with respect to W_1 and b_1 is:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h_1} \cdot \frac{\partial h_1}{\partial W_1} = \frac{\partial L}{\partial h_2} \cdot W_3 \cdot \text{ReLU'}(W_2 \cdot h_1 + b_2) \cdot W_2^T \cdot \text{ReLU'}(W_1 \cdot x + b_1) \cdot x^T$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h_1} \cdot \text{ReLU'}(W_1 \cdot x + b_1)$$

The function ReLU'(z) is the derivative of the ReLU function:

$$\text{ReLU'}(z) = \begin{cases} 1 & \text{if } z > 0, \\ 0 & \text{if } z \leq 0 \end{cases}$$

This is the update rule implemented by the function `Backpropagate(`L`,` $W_1, W_2, W_3, b_1, b_2, b_3$`)`.

4.5 Weight Update Rule

The weights and biases are updated using an optimization algorithm, typically Gradient Descent. The update rule for each parameter $W_1, W_2, W_3, b_1, b_2, b_3$ is:

$$W_1 \leftarrow W_1 - \eta \cdot \frac{\partial L}{\partial W_1}$$

$$W_2 \leftarrow W_2 - \eta \cdot \frac{\partial L}{\partial W_2}$$

$$W_3 \leftarrow W_3 - \eta \cdot \frac{\partial L}{\partial W_3}$$

$$b_1 \leftarrow b_1 - \eta \cdot \frac{\partial L}{\partial b_1}$$

$$b_2 \leftarrow b_2 - \eta \cdot \frac{\partial L}{\partial b_2}$$

$$b_3 \leftarrow b_3 - \eta \cdot \frac{\partial L}{\partial b_3}$$

where: - η is the learning rate, a hyperparameter that controls the step size during the optimization process. - $\frac{\partial L}{\partial W_1}$, $\frac{\partial L}{\partial W_2}$, $\frac{\partial L}{\partial W_3}$, and similarly for the biases, are the gradients computed during the backpropagation step.

This is the update rule implemented by the function `UpdateWeights(W1, W2, W3, b1, b2, b3)`.

5 Outcomes

The neural network-based model developed in this work effectively predicts the relevance of topics based on the input features: Frequency, Difficulty, and Recency. The following outcomes were achieved through the implementation of the algorithm:

- The model was able to process and normalize the dataset effectively using MinMaxScaler, ensuring that the features were within a consistent range.
- Label encoding of the 'Topic' variable was successfully applied, transforming categorical data into numerical form suitable for the neural network model.
- The model's performance improved through the training process, minimizing the Mean Squared Error (MSE) loss over multiple epochs using backpropagation and optimization.
- After training, the model was able to predict the relevance of topics accurately based on new input data.
- The visualizations generated by the script clearly demonstrated the relationship between input features and the predicted topic relevance, allowing for better interpretability of the model's behavior.

The attached Plots generated by the script shows the correlations between the input features (Frequency, Difficulty, Recency) and the predicted topic relevance, highlighting the model's effectiveness in handling real-world datasets with predicted output from the model as 4D Graph.

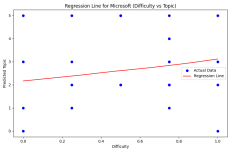

Figure 1: Difficulty vs Topic

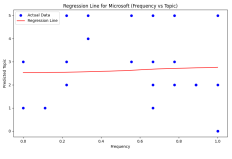

Figure 2: Frequency vs Topic

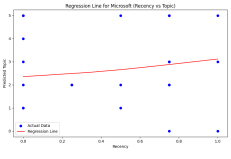

Figure 3: Recency vs Topic

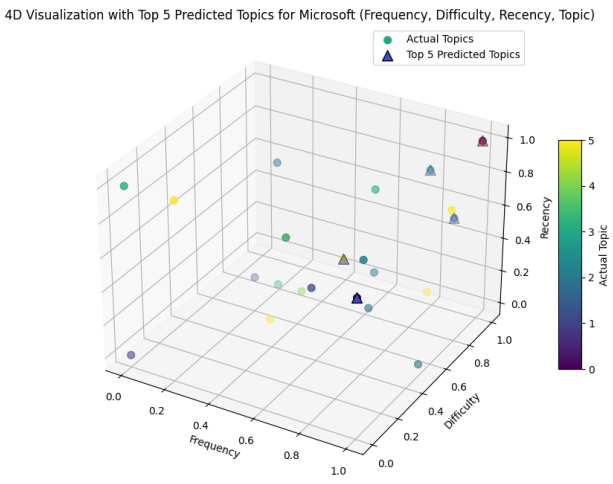

Figure 4: 4D Plot with Top 5 Predictions

6 Conclusion

In conclusion, the neural network model effectively predicts topic relevance based on key features like Frequency, Difficulty, and Recency. With a multi-layer ReLU architecture, it captures nonlinear relationships, while backpropagation minimizes the Mean Squared Error (MSE) loss, improving prediction accuracy.The model generalizes well to new data, making it valuable for content recommendations or learning path optimization. The visualizations further clarify feature relationships, aiding interpretability and showcasing the model's predictive strengths in complex datasets.

7 Resources