Personalizing the Learning Experience:
An Adaptive Algorithm Model Based on K-NN

Hicham ER-RADI[1], Souhaib AAMMOU[1], Zakaria TAGDIMI[1], Ikram AMZIL[1]

[1] Abdelmalek Essaadi University, S2IPU, Morocco
erradihicham.doc@gmail.com

Abstract. This paper proposes the use of the k-NN algorithm as a method for adjusting the level of difficulty of educational content based on learner preferences and performance. The goal is to achieve personalized learning experiences that can improve learner engagement and performance. The paper provides a theoretical framework on adaptive learning systems and content difficulty adjustment, which involves using data analytics and machine learning algorithms to adapt educational content to the individual learner's abilities and knowledge level. The k-NN algorithm is explained in detail, along with its steps and design process. The paper proposes a general design process that involves collecting data, preprocessing data, training the k-NN model, and adjusting content difficulty based on the predictions generated by the k-NN model.

Keywords: Adaptive Learning Systems, K-nearest neighbors (k-NN) algorithm, Content difficulty adjustment.

1 Introduction

Personalizing the learning experience can lead to improved learner performance and engagement. Adaptive Learning Systems can help achieve personalization by adjusting the content and difficulty level of learning tasks based on the learner's performance and preferences.

Adjusting the difficulty level of content to improve learner performance can be a challenging task for educators. It can be difficult to identify the appropriate level of difficulty for each learner, especially in a large classroom with a diverse group of learners. The main objective of this paper is to propose an adaptive algorithm model based on the k-NN algorithm to personalize the learning experience for learners. This model aims to adjust the level of difficulty of content based on learner preferences and performance to improve learner engagement and performance.

M. Khaldi et al. (eds.), *Proceedings of the E-Learning and Smart Engineering Systems (ELSES 2023)*, Atlantis Highlights in Social Sciences, Education and Humanities 14,
https://doi.org/10.2991/978-94-6463-360-3_7

2 Theoretical framework

2.1 Adaptive learning systems

Adaptive learning systems are computer-based educational platforms that use data analytics and machine learning techniques to personalize learning experiences for individual learners [1]. These systems use data about a learner's learning style, performance, and knowledge level to tailor instruction to their specific needs and abilities.

Adaptive learning systems can be used in a variety of educational settings, from K-12 classrooms to higher education and corporate training environments. They can provide learners with individualized instruction, feedback, and assessments, which can improve their learning outcomes and engagement.

Adaptive learning systems use a range of data sources to create personalized learning experiences, including learner performance data, user interaction data, and demographic data [2]. They can also incorporate other factors, such as learning goals, preferences, and interests.

Adaptive learning systems can be integrated into existing learning management systems or used as standalone platforms. They can be used to deliver a wide range of content, from simple quizzes to complex simulations and games.

One of the key benefits of adaptive learning systems is that they can help educators to identify areas where learners are struggling and provide targeted support to help them improve [3].

2.2 Content difficulty adjustment

Content difficulty adjustment is a process used in adaptive learning systems to adapt educational content to the individual learner's abilities and knowledge level [4]. The goal is to ensure that the content is challenging enough to promote learning, but not so difficult that the learner becomes frustrated or disengaged.

Content difficulty adjustment involves using data analytics and machine learning algorithms to assess the learner's performance and knowledge level. This information is then used to adjust the difficulty of the educational content in real-time.

Content difficulty adjustment can be used with the k-NN (k-nearest neighbors) algorithm to personalize the learning experience for individual learners. The k-NN algorithm is a type of machine learning algorithm that can be used to predict a learner's performance based on their previous interactions with the educational content.

3 Methodology

3.1 k-NN algorithm

The k-nearest neighbors (KNN) algorithm is a type of supervised learning algorithm used for classification and regression [5]. It is a non-parametric method that works by finding the k closest data points in the training set to a new, unseen data point and

using the majority vote (for classification) or averaging (for regression) of their labels or values as the prediction for the new point. The steps for the KNN algorithm:

1. Choose a value for k.
2. For each new, unseen data point, calculate the distance to all the points in the training set.
3. Choose the k nearest neighbors based on their distance to the new point.
4. For classification, predict the label of the new point as the majority label among the k neighbors. For regression, predict the value of the new point as the average of the values among the k neighbors.

The choice of k is important in the KNN algorithm, as it determines the number of neighbors that will contribute to the prediction. A small value of k may lead to overfitting, while a large value of k may lead to underfitting. The optimal value of k can be determined by cross-validation or other methods.

3.2 Design process

Content difficulty adjustment can be used with the k-NN algorithm according to these general phases:

1. Collect data: Collect data on the learner's previous interactions with the educational content. This could include data on their performance, engagement, learning style, and goals.
2. Preprocess data: Preprocess the data to remove any noise or irrelevant information. This could involve removing outliers or normalizing the data.
3. Train the k-NN model: Use the preprocessed data to train the k-NN model. The model will learn to identify patterns in the data and make predictions about the learner's performance based on their previous interactions with the educational content.
4. Adjust content difficulty: Use the predictions generated by the k-NN model to adjust the difficulty of the educational content. For example, if the model predicts that the learner will struggle with a particular topic, the system can adjust the content to provide additional explanations or examples. Alternatively, if the model predicts that the learner will find a topic easy, the system can adjust the content to provide more challenging material.
5. Evaluate the model: Evaluate the performance of the k-NN model and adjust it as necessary. This could involve tuning the model parameters or collecting additional data to improve the model's accuracy.

4 Implementation

4.1 Implementation process

We can implement the k-nearest neighbors (k-NN) algorithm to adjust the level of difficulty of content based on learner preferences and performance to improve learner engagement and performance:

1. Collect Data: Collect data on the learner's preferences and performance, such as their interaction history with the educational content, the amount of time they spend on different topics, their performance on assessments, and any feedback they provide.
2. Preprocess Data: Preprocess the data to prepare it for analysis. This could involve removing missing or irrelevant data, normalizing the data, and feature scaling.
3. Train the Model: Use the preprocessed data to train the k-NN model. The model will learn to identify patterns in the data and predict the learner's performance based on their previous interactions with the educational content.
4. Adjust Content Difficulty: Use the predictions generated by the k-NN model to adjust the difficulty of the educational content. For example, if the model predicts that the learner will struggle with a particular topic, the system can adjust the content to provide additional explanations or examples. Alternatively, if the model predicts that the learner will find a topic easy, the system can adjust the content to provide more challenging material.
5. Evaluate the Model: Evaluate the performance of the k-NN model and adjust it as necessary. This could involve tuning the model parameters, collecting additional data to improve the model's accuracy, or using other machine learning algorithms in combination with the k-NN algorithm.

4.2 Python code

Collect Data:

. We have collected data on the learner's preferences and performance on a set of educational quizzes. The dataset contains the following columns:

- Gender: Gender of the learner (0 = male, 1 = female)
- Age: Age of the learner
- Quiz1_Score: Score of the learner on the first quiz (out of 10)
- Quiz2_Score: Score of the learner on the second quiz (out of 10)
- Quiz3_Score: Score of the learner on the third quiz (out of 10)
- Difficulty_Level: Difficulty level of the quiz (0 = easy, 1 = medium, 2 = hard)

We'll save this dataset as a CSV file named "learner_data.csv".

Preprocess Data:

. We'll load the dataset into a Pandas dataframe, and preprocess the data by normalizing the numerical columns (Age, Quiz1_Score, Quiz2_Score, Quiz3_Score) using MinMaxScaler from sklearn.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
data = pd.read_csv('learner_data.csv')

# Normalize the numerical columns using MinMaxScaler
scaler = MinMaxScaler()
data[['Age', 'Quiz1_Score', 'Quiz2_Score', 'Quiz3_Score']] = scaler.fit_transform
    (data[['Age', 'Quiz1_Score', 'Quiz2_Score', 'Quiz3_Score']])
```

Train the Model:

. Next, we'll train the k-NN model on the preprocessed data. In this example, we'll use the KNeighborsClassifier from sklearn with k=5.

```
from sklearn.neighbors import KNeighborsClassifier

# Split the data into features (X) and target variable (y)
X = data[['Gender', 'Age', 'Quiz1_Score', 'Quiz2_Score', 'Quiz3_Score']]
y = data['Difficulty_Level']

# Train the k-NN model with k=5
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)
```

Adjust Content Difficulty:

. We can now use the trained k-NN model to adjust the difficulty level of a quiz question based on the learner's predicted performance.

```
# Collect input from the learner on the quiz question
learner_input = pd.DataFrame({
    'Gender': [0],
    'Age': [25],
    'Quiz1_Score': [8],
    'Quiz2_Score': [7],
    'Quiz3_Score': [9]
})

# Normalize the learner input using the same scaler used to preprocess the data
learner_input[['Age', 'Quiz1_Score', 'Quiz2_Score', 'Quiz3_Score']] = scaler.transform
    (learner_input[['Age', 'Quiz1_Score', 'Quiz2_Score', 'Quiz3_Score']])

# Use the trained k-NN model to predict the learner's difficulty level
predicted_difficulty_level = knn.predict(learner_input)[0]

# Adjust the difficulty level of the quiz question based on the predicted difficulty level
if predicted_difficulty_level == 0:
    print("This question is already easy.")
elif predicted_difficulty_level == 1:
    print("Adjusting difficulty level of the question to make it easier.")
elif predicted_difficulty_level == 2:
    print("Adjusting difficulty level of the question to make it harder.")
```

In this example, we've assumed that the learner is attempting a quiz question that has a difficulty level of 1 (medium). We've collected information from the learner about their gender, age, and scores on three previous quizzes. We've then used the trained k-NN model to predict the learner's difficulty level, based on their input. The model predicts a difficulty level of 2 (hard), so we adjust the difficulty level of the question to make it harder.

Evaluate the Model:

. To evaluate the performance of the k-NN model, we can use a hold-out dataset. We'll split the original dataset into training and test sets using train_test_split from sklearn.

```
from sklearn.model_selection import train_test_split

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the k-NN model on the training set
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Evaluate the performance of the model on the test set
accuracy = knn.score(X_test, y_test)
print("Accuracy:", accuracy)
```

The content difficulty adjustment using k-NN algorithm can help to personalize the learning experience and improve learner engagement and performance. However, it's important to note that the quality of the predictions and adjustments will depend on the quality and quantity of the data used to train the model.

5 Conclusion

Using the k-NN algorithm to personalize the learning experience and adjust the difficulty of educational content has shown promising results in the development of adaptive learning systems. By implementing the k-NN algorithm to adjust the level of difficulty based on learner preferences and performance, these systems can offer a more effective and engaging learning experience. The personalized learning experience can motivate learners and keep them engaged, while the adaptive content can provide the appropriate level of challenge and support for more effective learning. Overall, these systems have the potential to improve learner engagement and performance by providing personalized and adaptive educational experiences.

References

1. Taylor, D. L., Yeung, M., & Bashet, A. Z. (2021). Personalized and adaptive learning. Innovative Learning Environments in STEM Higher Education: Opportunities, Challenges, and Looking Forward, 17-34.
2. Daghestani, L. F., Ibrahim, L. F., Al-Towirgi, R. S., & Salman, H. A. (2020). Adapting gamified learning systems using educational data mining techniques. *Computer Applications in Engineering Education*, *28*(3), 568-589.
3. Pak, K., Polikoff, M. S., Desimone, L. M., & Saldívar García, E. (2020). The adaptive challenges of curriculum implementation: Insights for educational leaders driving standards-based reform. *AERA Open*, *6*(2), 2332858420932828.
4. Alshammari, M. T., & Qtaish, A. (2019). Effective Adaptive E-Learning Systems According to Learning Style and Knowledge Level. *Journal of Information Technology Education*, *18*.
5. Bansal, M., Goyal, A., & Choudhary, A. (2022). A comparative analysis of K-Nearest Neighbour, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning. *Decision Analytics Journal*, 100071.