Mood Detection : 522 Final Project

David Feldt
Department of System Design Engineering
Waterloo, ON
dfeldt@uwaterloo.ca

Abstract

This study explores the prediction of individual mood based on behavioral patterns, encompassing habits and activities. This report compares the performance of Convolutional Neural Networks (CNNs) and linear regression models in capturing the intricate relationships between behavioral features and mood states. Through testing and evaluation that CNNs outperform linear regression in this predictive task, demonstrating the superior ability of CNNs to discern complex spatial dependencies and hierarchical patterns within behavioral data.

1 Introduction

This report explores the relationship between daily habits, time allocation, and mood. By journaling each day a mood ratings is gathered on a scale of 1-10. As well as tracking time spent on various activities such as sleeping, phone usage, and productivity. The project also tracks habits including working out, meditation, and reading. The primary aim is to utilize this dataset, encompassing over three years of recorded habits and activities, to predict mood fluctuations. Notably, the tracked habits and activities have changed over time, providing a dynamic and comprehensive dataset for analysis. This data collection and analysis offer invaluable insights into the complex dynamics of lifestyle habits and their impact on mood.

This report dives deeper into the fundamental understanding of mood dynamics. Mood, a complex and multifaceted aspect of human psychology, is influenced by a myriad of factors, including daily activities, sleep patterns, and personal habits. There is always a growing interest in improving mood and mental well-being, acknowledging mood's pivotal role in overall health and productivity as well as preventing low mood. The problem is multifaceted: identifying which specific factors most accurately predict mood variations. By analyzing over three years of detailed data on activities and habits, the goal is to uncover patterns and correlations that can influence mood. Over with these insights improved mental health and quality of life.

2 Experimental Setup and Data

In the data collection was done through daily journaling. The primary focus was on mood tracking, where the individual rated their mood on a scale from 1 to 10 each day, providing a quantitative measure of their emotional state. Additionally, the hours spent on various activities were recorded in detail, quantified in hourly units. The tracking extended to personal habits, which were monitored in a binary fashion, marked as 0 or 1 to denote whether a particular habit was not completed or completed, respectively. This systematic approach to data collection aimed to capture a holistic view of the participants' daily routines and their impact on mood. The data be analyzed over 3 years with over 1000 entries.

The experimental setup for our mood data analysis involved several meticulous steps to ensure data integrity and usefulness:

2.1 Manual Data Entry to Excel

The collected data was manually in a physical and was entered into an Excel spreadsheet. This process allowed for a structured and organized dataset, vital for subsequent analysis.

2.2 Data Label Modification

To address gaps in the dataset, the data labels were refined. For instance, diverse activities like skateboarding and working out were categorized under a unified label, 'Exercise'. This step was crucial to reduce fragmentation in the data. For categories that were logically similar, such as 'Reading' and 'Meditating' into 'Solitude' or 'Study' and 'Course Learning' into 'Productivity', the entries were aggregated to provide a more cohesive analysis. Rigorous checks were performed to correct any typographical errors in the dataset, ensuring the accuracy of the data.

2.3 Finalized Data Categories

The dataset was distilled into specific categories, including both activities tracked by hours and binary habit tracking. The final categories were: ['Mood', 'Sleep (hrs)', 'Phone Usage (hrs)', 'Productivity (hrs)', 'Productivity', 'Solitude', 'Exercise', 'Leetcode', 'Journal', 'Profile Pic', 'Bad Habit', 'Other (hrs)', 'Exercise (hrs)', 'Media', 'Duolingo'].

2.4 Data Augmentation for Missing Columns

To handle missing data, the dataset was augmented by adding columns with all zeros where data was absent. Furthermore, a new column called column exits was introduced to track the presence of specific data points over time.

2.5 Data Preperation

The data was organized by month and needed to be combined into 1 continuous stream. Finally the data was split into testing and training with a 80/20 split. Additionally the 2 previous rows were added to each entry to provide more information for each entry and the first 2 entries were dropped.

3 Methodology

Selecting the most suitable machine learning model is a critical aspect of this project. The model needs to be able to handle with a dataset characterized by 30 variables per entry, each ranging from 0 to 10, and an output variable within the same range. In this context, two distinct models, Linear Regression and Convolutional Neural Network (CNN), have been chosen based on their inherent strengths. Linear Regression, a fundamental statistical model, proves valuable when the relationship between input features and the target variable is presumed to be linear. Its simplicity and interpretability make it a pragmatic choice for cases where a straightforward understanding of variable interactions is desired. In contrast, the CNN, originally designed for image processing, is brought into consideration due to its capability to capture complex patterns and spatial relationships within data. Comparing the two will aim to harness the unique advantages each model offers for optimal predictive performance.

3.1 Linear Regression

Linear regression is a fundamental and widely-used machine learning algorithm that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It is particularly effective when there is a linear relationship between the input features and the target variable. However, linear regression does have its limitations.

Linear regression provides a straightforward interpretation of the relationships between input features and the target variable. Each coefficient represents the change in the target variable for a unit change in the corresponding input variable. Linear regression is a simple and computationally efficient algorithm. It is easy to implement, understand, and doesn't require complex hyperparameter tuning. When the underlying relationships in the data are approximately linear, linear regression performs well and provides reliable predictions. Linear regression does not make strong assumptions about the distribution of the input features.

Linear regression assumes a linear relationship between input features and the target variable. If this assumption is violated, the model may provide inaccurate predictions. Linear regression is sensitive to outliers, and the presence of extreme values can significantly impact the model's coefficients and predictions. Linear regression may struggle to capture complex, non-linear relationships present in the data. It assumes that the errors are independent, which may not always be true in real-world scenarios.

To test linear regression 3 different models were implemented. Below is an example on how the model was implemented in python using the sklearn library.

```
# Linear Regression for default parameters
model_default = LinearRegression()
model_default.fit(X_train, y_train)
y_pred_default = model_default.predict(X_test)


# Linear Regression with no intercept
model_no_intercept = LinearRegression(fit_intercept=False)
model_no_intercept.fit(X_train, y_train)
y_pred_no_intercept = model_no_intercept.predict(X_test)


# Linear Regression with normalization
model_normalized = LinearRegression(normalize=True)
model_normalized.fit(X_train, y_train)
```

```
y_pred_normalized = model_normalized.predict(X_test)
```

All three approaches were evaluated using metrics such as R-squared (coefficient of determination) and Mean Squared Error (MSE) to assess the model's accuracy. R-squared (R^2) for all three approaches: 0.67 and Mean Squared Error (MSE) for all three approaches: 1.3. Despite using different approaches (default parameters, no intercept, and normalization), the models exhibited the same accuracy. This suggests that, in this specific case, the variations in model configuration did not significantly impact the model's performance. The R^2 of 0.67 indicates that the model explains 67

3.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a deep learning architecture designed for processing spatial hierarchies of features, commonly applied to image and sequence data. In a sequential CNN, layers are stacked in a linear fashion, enabling the network to automatically learn hierarchical representations, making it effective for tasks such as image recognition and sequential pattern analysis.

Convolutional Neural Networks (CNNs) excel at hierarchical feature learning. Each layer extracts increasingly complex features from the input, allowing the network to capture intricate patterns in the data. CNNs are well-suited for tasks where translation invariance is important. This is achieved through the use of convolutional and pooling layers, enabling the model to recognize patterns regardless of their spatial location in the input. CNNs automatically learn relevant features from the data without manual feature engineering. This is especially advantageous when dealing with high-dimensional datasets, such as those with 30 variables per entry. CNNs leverage parallel processing, making them efficient for training and inference on hardware with parallel computation capabilities like GPUs.

Training deep CNNs can be computationally intensive, requiring substantial computational resources. This may limit their feasibility for certain applications without access to powerful hardware. Deep CNNs are prone to overfitting, especially when dealing with limited datasets. Regularization techniques and appropriate dropout layers are often necessary to mitigate overfitting. The complex architectures of deep CNNs can make them challenging to interpret. Understanding the inner workings of the model may be less straightforward compared to simpler models like linear regression.

To test CNNs 3 different models were implemented. Below is an example on how the model was implemented in python using the tensorflow library.

```
# CNN variation 1
model_1 = Sequential()
model_1.add(Conv1D(filters=32, kernel_size=3, activation='relu',
input_shape=(X.shape[1], 1)))
model_1.add(MaxPooling1D(pool_size=2))
model_1.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model_1.add(MaxPooling1D(pool_size=2))
model_1.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
model_1.add(MaxPooling1D(pool_size=2))
model_1.add(Flatten())
model_1.add(Dense(64, activation='relu'))
```

```
model_1.add(Dense(1))

  # CNN variation 2
  model_2 = Sequential()
  model_2.add(Conv1D(filters=32, kernel_size=3, activation='relu',
  input_shape=(X.shape[1], 1)))
  model_2.add(MaxPooling1D(pool_size=2))
  model_2.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
  model_2.add(MaxPooling1D(pool_size=2))
  model_2.add(Flatten())
  model_2.add(Dense(128, activation='relu'))
  model_2.add(Dense(1))

    # CNN variation 3
    model_3 = Sequential()
    model_3.add(Conv1D(filters=32, kernel_size=3, activation='relu',
    input_shape=(X.shape[1], 1)))
    model_3.add(MaxPooling1D(pool_size=2))
    model_3.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model_3.add(MaxPooling1D(pool_size=2))
    model_3.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
    model_3.add(MaxPooling1D(pool_size=2))
    model_3.add(Conv1D(filters=256, kernel_size=2, activation='relu'))
    model_3.add(MaxPooling1D(pool_size=1))
    model_3.add(Flatten())
    model_3.add(Dense(64, activation='relu'))
    model_3.add(Dense(1))
```

The accuracies (R^2 and MSE) for the three models are as follows: Variation 2 (R^2: 0.6506, MSE: 1.3235) Variation 1(R^2: 0.6689, MSE: 1.2539), Variation 3 (R^2: 0.6813, MSE: 1.2072) The results indicate that as the CNN architecture becomes more complex (from 2 layers to 4 layers), both R^2 values increase, indicating improved model fit, and MSE values decrease, indicating better predictive performance. This suggests that a more intricate model is better suited to capture the complexities present in the data, making it more effective for the given regression task. However, it's essential to strike a balance to avoid overfitting, especially when working with limited datasets.

4 Results and Discussion

The graphs outputted below show actual mood vs the predicted mood. When a point is on the red line it means that the model correctly predicted mood. Dots above the line are when predicted mood are higher than the actual mood. And when the dots are below the line are when the predicted mood are lower than the actual mood. Each model seemed to have its own bias for predicting higher. Graphs can be seen in Figure 1 and Figure 2.

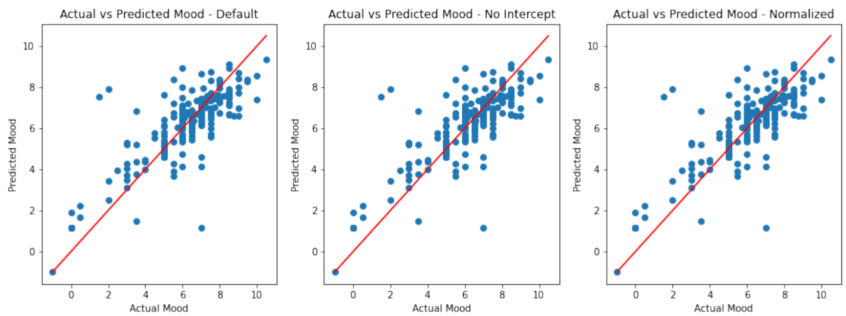

Figure 1: Linear Regression Output

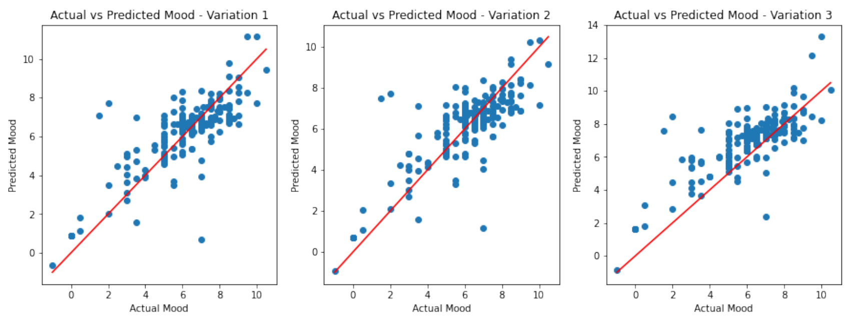

Figure 2: CNN Output

.

In comparing Convolutional Neural Networks (CNNs) and linear regression, both models demonstrated similar performance on a given task, yet the more complex CNN exhibited superior predictive accuracy. While both models achieved comparable R-squared values, the CNN's higher complexity enabled it to capture more intricate patterns in the data, resulting in a lower Mean Squared Error (MSE) and enhanced overall predictive capability. This highlights the CNN's capacity to excel in tasks with complex relationships and spatial dependencies, showcasing its effectiveness compared to the linear regression model.

5 Conclusion

In conclusion, the comparison between Convolutional Neural Networks (CNNs) and linear regression underscores the trade-offs between model complexity and efficiency. While linear regression stands out as a fast and sometimes effective method, suitable for simpler relationships, CNNs offer a more flexible modeling approach. The latter's capacity to capture intricate patterns and spatial dependencies makes it particularly well-suited for tasks where the underlying data relationships are complex.

Looking to the future, the potential applications of CNNs extend beyond traditional regression tasks. For instance, they could be leveraged to predict and understand behavioral patterns, such as sleep cycles. By integrating diverse data sources and leveraging the spatial hierarchies learned by CNNs, it becomes conceivable to develop models that predict, influence, and optimize individual behaviors related to mood and well-being. Such models could provide insights into how to enhance mood or prevent low moods, paving the way for personalized interventions and well-being strategies based on data-driven approaches. This not only demonstrates the adaptability of CNNs but also highlights their potential impact in addressing complex challenges related to human behavior and mental health.