Predicting errors during Pacman for stroke patients

Mentor: Arzu Güneysu Özgür, CHILI Lab
Authors: Irina Bejan, Nevena Drešević & Marija Katanić
School of Computer and Communication Sciences, EPFL Lausanne, Switzerland

Abstract—**This paper aims to present our approach in predicting future errors in the Pacman game played by stroke patients, such as ghosts eats and hitting walls. The input data set consists of data collected on the Cellulo project in the CHILI laboratory at EPFL, and the paper covers our methodology to analyze, preprocess the data, as well as the models we considered to predict future events in the game. Predicting such events is valuable to make the game adaptive in real-time and increase rehabilitation success. Considering a time-series approach, we explored logistic regression models, decision trees, and recurrent neural networks, obtaining promising results.**

I. Introduction

This Pacman game is a part of Cellulo rehabilitation project and provides practical, easy to use, and intuitive gamified rehabilitation by using tangible robots. The participants of the game are stroke patients, for which this is upper arm exercise. With these error predictions, the therapists will be able to tune several game elements such as speed and movement of the ghost, positions of apples, etc. While a patient is playing the game, the difficulty level could be changed in real-time depending on player performance and needs.

The data set used for training the model is original data collected through a handheld robot by players of the Pacman game on a paper maze. Analyzing the semantics of the data, we derive multiple features related to motor performance metrics and spatial information, such as map sectors. Based on the features, we explore using both classification and regression to predict the probability of an error happening in the next couple of time frames in the future. We compare the performance achieved by various machine learning models, including logistic regression, random forests, XGBoost, feed-forward neural networks, recurrent networks, and fine-tuning various problem hyperparameters, such as the window size, number of backward and forward time frames considered. The performance is given by the F1-score and the recurrent neural network leads to best results, with a score of 0.77 for crossing borders and a score of 0.53 for ghost eating. We further analyze the models to explain what features are more indicative of future errors, gaining more insights into the difficulties the stroke patients have.

II. Data

A. Dataset

The data set is collected by playing a total of 810 games on three different types of maps. Maps are different sizes, with each having one or two ghosts who are chasing the player and 6 apples placed on fixed positions. The game ends once the player collects all the apples. A player makes an error and dies in the following events: it is eaten (caught by a ghost) or crossed the border (crashed into a wall). In both cases, the player loses collected apples and continues from the start.

During the game, the system continually sends time-series data (10hz interval) of the players' position data (Pacman robot - the hand of the person) and ghosts. Also, every moment of making an error is captured, which gives two different data sets based on the error type. Besides this real-time information from the system, we receive specific attributes and metrics for each patient, out of which we use the hand it plays with (left or right). The games are played by two groups of patients, namely EIS and ISTB, where groups have 726 and 84 games played. The groups have some differences in their stroke level, age, and rehabilitation location. Hence, patients' performance in these groups varies, and we train on both data sets to better generalize the model for newly arrived patients. We analyze all the data, preprocess it, and extract the features to build the model that predicts if a patient will make an error in the next x seconds of the game.

B. Preprocessing

To make the most of the collected data, we focused on refining it and extracting it with new features. Since stroke is related to motor performance, some of the features we compute using position data of the pacman are motor performance-related metrics. Such features are velocity, acceleration, deviance, and jerk, the latter representing a measurement of the smoothness of the movement. We also add mean, max, and the ratio of these metrics, as well as peaks (sign change of acceleration) per time and time to the highest peak.

Since the two types of errors we want to predict have different prerequisites and causes, we prioritize different features depending on the specific error to better train our model. When predicting if ghosts will catch the player, we take into account the ghosts' positions in time and distances from pacman to the nearest and farthest ghost. To better understand our problem and extract further information, we compared the calculated features of game data from stroke patients with data received when healthy people played the same game. Plotting the data also showed how making the error by crossing the border is dependant on the closeness to the apples. This can be explained by the patients' difficulties in controlling the movement, primarily when their attention is devoted to collecting the apple.

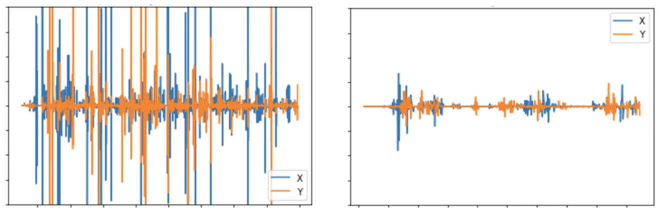

Figure 1: The smoothness of the movement (jerk) during the time for a stroke patient (left) and a healthy person (right).

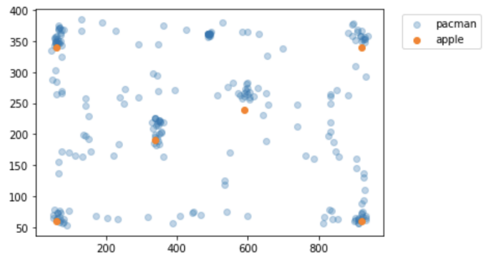

Figure 2: Blue dots represent the positions of players in the moment of crossing a border, while red ones are positions of apples on the map.

In addition, since the map's size depends on the patient's range of motion and size, we split the map into smaller parts. Those parts represent sectors for which we calculate the percentage of time the player spent in each sector. The best results are achieved when splitting all maps into two sectors (left and right half).

Finally, we standardize all features except the hand and sectors, categorical and percentage data, respectively. We give this data to the machine learning model without dropping the outliers because such extreme values might be tied to more severe stroke patients and their struggle to control the movement.

III. Methods

The first challenge was modeling the foretelling of in-game errors as an optimization problem.

A. Imbalance

Given the nature of the movement, our approach relied on time series, and we stated an initial binary classification problem of predicting whether an error will happen in a specific time window from now, considering a fixed number of windows in the past. However, we were facing a trade-off: by decreasing the size of the time frame (10s to 3s, for example), we were able to learn more granular movements, but given the small number of errors that happen within the game, we were increasing the imbalance significantly. For the window size of 3, the imbalance of ghost eating errors was 1 to 48 non-errors. To address it, we considered using cost-sensitive learning by applying weights derived from the data distribution to our models and further use over-sampling of the errors.

However, these efforts did not increase the learning ability of our models. We considered two new views of our problem to increase, artificially, the number of errors: regression, where a fixed number of time frames preceding an error are labeled with an increasing probability of an error to happen, and a threshold is chosen to optimize the target scores, and classification, but predicting if an error will happen or not during the next five time frames (for a window size of 3, that would be in the next 15 seconds). While the regression models were merely unable to distinguish between errors and non-errors, the latter approach proved to work best together with our imbalance mitigation approach, as it was reduced from 15 non-errors to 1 error.

B. Metrics

To compare the implemented models and measure each of them's performance, we take a few different metrics into account. The primary metric we consider is the F1 score, a harmonic mean of the precision and recall, i.e.,

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Also, we look at recall and precision separately. Precision tells us what proportion of positive identifications was actually correct, and recall explains what proportion of actual positives was identified correctly. Furthermore, we take accuracy into account, but this is not the best metric because the data is highly imbalanced with few errors. For this reason, we introduce accuracy1, which tells us only the accuracy of predicting errors. The last metric that we include is the area under the ROC curve (AUC) score, which explains how well our model distinguishes between classes, which means that the higher the AUC, the better the model predicts 0s as 0s and 1s as 1s.

C. Models

We focused our analysis on five models: Logistic Regression, Random Forest, XGBoost (shallow and deep), feed-forward neural network, and a recurrent neural network. We used cross-validation with five splits to choose hyperparameters, such as the number of backward steps considered for learning, the learning rate, the time frame size, the regularization factors, and the recurrent layers' sizes. Therefore, we continued our analysis considering five backward steps (input being a sequence of 5 consecutive time frames), each having a window size of 3 seconds, and using a time frame for our future prediction of 15s.

The logistic regression is run for at most 1000 iterations. For the random forest and XGBoost, we consider 300 estimators with a depth of 5. We also consider a shallow variant of the XGBoost, employing only 200 estimators and a depth of 3. These values were chosen by running a grid search.

The architecture of the neural networks is presented in Figure 3. The feed-forward is composed of two fully connected layers, first of 128 units, using a leaky rectified linear activation for its hidden units to avoid the activation value of 0.0, and one layer for the final classification using a softmax activation to output the probability. Both the feed forward and the recurrent model are trained using a categorical cross-entropy loss.

For the recurrent neural network, we employ a stacked long short-term memory architecture. The LSTM units [2] use a gating mechanism to retain or forgot data from the previous state and preserve information, solving the vanishing gradient from traditional RNNs, outperforming traditional methods employed in time series analysis [5]. Furthermore, stacking LSTM units had better performance in detecting anomalies (abrupt changes) on multiple datasets [4]. As previously, a leaky rectified linear activation for the first layer is used. As we found the sizes of 100 and 80 units respectively for the LSTM layers to perform better, we also greatly increased the model's complexity, so we needed to prevent overfitting. We accomplish this by using dropout with a factor of 0.5 before the classification layer and use in both LSTM layers an L1, and L2 regularization [6] with a factor of 1e-5, which performs better than Lasso and Ridge regression alone on our problem.

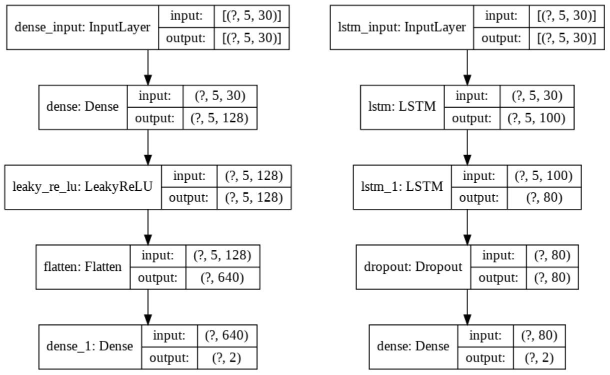

Figure 3: Model architectures for feed-forward neural network (left) and recurrent neural network (right).

For both models, we ensure the batches have equal numbers of errors and non-errors via oversampling, we use a batch size of 32, the Adam optimizer [3] with a learning rate of 0.001, and we train until the validation loss decreases with the patience of 15 epochs, choosing the model with minimum validation loss.

To ensure the correctness of the model, we further did leave-one-out cross-validation by removing each patient from the training set and analyzing the model's ability to generalize the learning on new patients, as it is critical to ensure it will be able to address patients with different stroke levels or age. A similar approach has been applied to leaving a percentage of full games in the validation set to see how the model generalizes new games.

The model were written using Keras, xgboost and sklearn libraries.

IV. RESULTS

The results are presented in Table I for the models trying to predict ghost errors and in Table II for the model trying to predict the cross-border errors. The LSTM-based architecture performs better, followed by the feed forward neural network. We believe they perform better given our oversampling approach to have balanced batches and therefore ease the imbalance much better. As a result, the neural network performs better with lower window sizes since it leads to a bigger dataset for training. Simultaneously, it increases the imbalance, therefore for LR, RF, and XGBoost models, a larger window size will lead to a better result.

Metric	LSTM	Feed Forward	LR	RF	XGBoost	Shallow XGBoost
F1	**0.529**	0.327	0.2644	0.267	0.2198	0.3108
Acc	0.917	0.805	0.7334	0.7558	**0.9326**	0.908
Acc1	0.68	0.689	**0.7194**	0.6702	0.1432	0.312
Precision	0.432	0.214	0.162	0.167	**0.4754**	0.3098
Recall	0.68	0.689	**0.7194**	0.6702	0.1432	0.312
AUC	**0.807**	0.751	0.727	0.7162	0.566	0.6312

Table I: Results for window size of 3 for ghost errors

Metric	LSTM	Feed Forward	LR	RF	XGBoost	Shallow XGBoost
F1	**0.844**	0.533	0.5162	0.5182	0.4688	0.5182
Accuracy	**0.781**	0.515	0.6664	0.6868	0.7366	0.7066
Accuracy1	**0.806**	0.378	0.6446	0.6104	0.421	0.5718
Precision	**0.885**	0.904	0.4308	0.4506	0.5296	0.4738
Recall	**0.806**	0.378	0.6446	0.6104	0.421	0.5718
AUC	**0.759**	0.633	0.66	0.663	0.639	0.6648

Table II: Results for window size of 3 for cross border errors

To better analyze the model's robustness and its ability to generalize, we consider two different ways to split the data into training and validation. First, by leaving each patient out and training on the rest of the patients, which leads to an average validation F1-Score of 0.749, which means it performs reasonably well on new patients. Another attempt is to split by individual games to check if the model's ability to learn from a couple of initial games would improve performance. This approach results in an average F1-Score of 0.717.

V. INTERPRETATION OF RESULTS

To interpret the results, we compute the permutation importance [1] on the best model and measure the increase in the F1 score after permuting each feature's values. Permutation breaks the tie between the feature and outcome, and if the feature is more important, the score will be significantly affected.

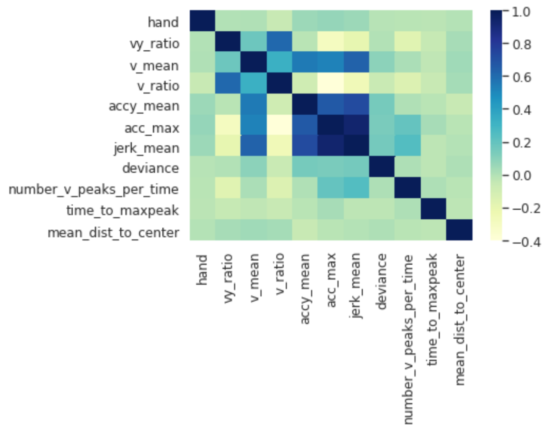

Figure 4: Correlation of a feature subset

The features we derivate from positional data (acceleration, velocity, jerk) are highly correlated, which can be seen in the subset of features presented in Figure 4, and it is not a surprise that a subset of them act as more important. The hand feature tells us that stroke patients are making similar mistakes based on their dominant hand. Further, the distance to the center is very important, given Figure 2, as we can see patients tend to overshoot when trying to eat the apple and therefore crash into the wall.

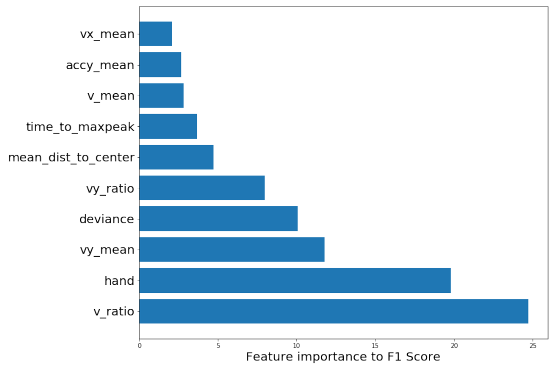

Figure 5: The importance of all features employed

VI. Future work

Our approach based on time series has improved performance, but we are aware that the model is given a much smaller amount of data than we, as players of the Pacman game, can perceive. We suspect that some regions of the map, at a given distance, where the patient needs to put more effort to reach, could cause more errors. However, the model has only two metrics targeting the map, the mean dist to the center and the sectors. An exciting idea to explore from here would be a more computer vision approach, to generate timeframes of the actual games programmatically from games and create images a network could much easier identify patterns that rely on the spatial dimensions.

VII. Conclusion

Predicting future errors in a Pacman game played by stroke patients has inherent challenges: the stroke patients' movements do not have the same mean, making it harder to learn, their movements have very high variance, and the data is not abundant. Furthermore, Pacman is a game, and the times you die is deficient compared to the length of a game, otherwise being very frustrating. This leads to a significant imbalance in our dataset, which we managed to ease, but it is still challenging to assess.

However, we managed to navigate through these issues, modeled the problem as a time-series problem, explored the ability of 5 models to predict two different types of errors, showing promising results on the usage of a LSTM based network. We are hopeful that the model can be used further in the Cellulo project to rehabilitate stroke patients.

References

[1] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, 2019.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[4] P. Malhotra, L. Vig, G. Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *ESANN*, 2015.

[5] S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[6] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.