IT540 – Assignment 3 - Instructions

Title: Predicting Chess Game Outcomes

[Exercise in Neural Networks]

Total Marks: 80
Tools: Python (Keras, TensorFlow, scikit-learn, matplotlib)

Research Objective and Goal

Research Objective: To design, implement, and evaluate predictive models using neural networks for classifying outcomes of professional-level chess games based on numerical features related to material strength and mobility.

Research Goal: To build a high-performing neural network-based classification system that can predict whether a chess game will result in a White win, Black win, or Draw using player-specific in-game metrics. This includes conducting cross-validation, optimizing hyperparameters, and deriving meaningful insights about model behavior and strategic gameplay prediction.

Assignment Overview

You are provided with a dataset of 100,000 expert-level chess games. Your objective is to predict the outcome of a chess game (White win, Black win, or Draw) using neural networks implemented in Python. This assignment will assess your skills in deep learning model construction, performance evaluation, and hyperparameter optimization. It also requires meaningful interpretation and discussion of your results.

You must work in Python. This assignment is designed to reflect the analytical depth and methodological rigor expected at the doctoral level. Learners are expected to demonstrate clear modeling rationale, critical interpretation of results, and a strong understanding of the broader implications of their models.

Instructions and Marking Scheme

1. Data Loading and Exploration (5 Marks)

- Load matmob.data.sample.csv using pandas. **(1 mark)**
- Print shape and one example row. **(1 mark)**
- Show distribution of the result variable using value_counts(). **(1 mark)**
- Plot a labeled bar chart of the result distribution. **(2 marks)**

2. Data Preparation (10 Marks)

- Retain only the following 6 columns: **(2 marks)**
    - w.mat.mean, b.mat.mean, w.mob.mean, b.mob.mean, half.moves, result
- Encode result to numeric using the following mapping: **(2 marks)**
    - '0-1' → 0, '1-0' → 1, '1/2-1/2' → 2
- Scale the 5 numeric predictor columns using StandardScaler(). **(3 marks)**
- Perform an 85% training / 15% test split using train_test_split(). **(3 marks)**

3. Baseline Neural Network Model (10 Marks)

- Use Sequential() to define a neural network with: **(2 marks)**
    - One hidden layer (50 units, activation='sigmoid')
    - Output layer (3 units, activation='softmax')
- Compile using: **(2 marks)**
    - Loss: sparse_categorical_crossentropy
    - Optimizer: SGD()
    - Metrics: accuracy
- Train with: **(2 marks)**
    - epochs = 100, batch_size = 128
- Evaluate and print test accuracy. **(2 marks)**
- Plot the training history (loss and accuracy). **(2 marks)**

4. Hyperparameter Tuning via Manual Grid Search (20 Marks)

- Define a list of at least **8 different configurations**. **(5 marks)**
    - Vary units, activation, optimizer, learning rate, batch size, and epochs.
- For each configuration: **(5 marks)**
    - Build, compile, and train a model.
    - Evaluate performance on the test set.
- Record accuracy for each configuration. **(5 marks)**
- Identify and display the best performing configuration. **(5 marks)**

5. Improved Model (10 Marks)

- Rebuild and retrain a new model using the best configuration from step 4. **(3 marks)**
- Evaluate performance. **(3 marks)**
- Plot the training history (loss and accuracy). **(2 marks)**
- Clearly compare its accuracy to the baseline model. **(2 marks)**

6. Interpretation and Application (15 Marks)

Prepare a clear, critical discussion covering:

- Why did the best configuration outperform the rest? **(4 marks)**
- Did activation functions or optimizers make a significant difference? **(3 marks)**
- Was overfitting observed? How was it mitigated? **(2 marks)**
- What patterns is the model likely learning? **(2 marks)**
- How could this system be applied in practice (e.g., chess tutoring, game analysis, strategic AI tools)? **(2 marks)**

- Could this methodology transfer to other domains (e.g., Go, finance, strategic planning)? **(2 marks)**

7. Code Quality and Submission Format (10 Marks)

- Ensure your code is well-commented, clean, and logically organized.
- Submit the following:
    - A .ipynb file or .py script
    - A structured PDF project report that includes the following sections**: (10 marks)**
        - **Title Page**: Project title, your name, course, and date.
        - **Abstract**: A brief summary of your approach, models used, and main findings.
        - **Introduction**: Overview of the objective, dataset, and modeling goals.
        - **Methodology**: Detailed explanation of model architecture, preprocessing steps, tuning approach, and evaluation strategy.
        - **Results**: Presentation of final test accuracies for both models, along with comparison plots.
        - **Visualizations**: Include training loss and accuracy curves, result distribution, and any other insightful charts.
        - **Discussion and Interpretation**: Critical analysis of model performance, interpretation of learned patterns, overfitting mitigation, and real-world applicability (Task 6).
        - **Conclusion**: Summary of key takeaways and possible future enhancements.
- Submit everything as a **single zip folder** named:
    - Assignment3_YourName.zip

Final Notes:

- This is an individual assignment.
- You are expected to demonstrate **depth of understanding** and **critical reasoning**, not just code execution.
- All results must be reproducible.
- Plagiarism or uncredited reuse of code will result in disqualification.

=====================Happy Learning! ==============================