User Guide for the `utils.py` Module

Version 1.0

October 18, 2024

Contents

1 Introduction

This module provides a function for visualising and comparing learning metrics against a baseline, particularly useful in reinforcement learning experiments involving algorithms like Q-learning or SARSA. It enables the plotting of multiple metrics on a single graph with multiple y-axes for comprehensive analysis.

2 Prerequisites

Before using the module, ensure you have the following software installed:

- **Python 3.x**
- **Matplotlib**: Required for plotting graphs.
- **Pandas**: Required for handling data in DataFrames.

You can install these packages using `pip`:

```
pip install matplotlib pandas
```

3 Installation

To use the `plot_comparison_with_baseline` function, ensure the `utils.py` file is in your assignment directory.

4 Function Overview

The `plot_comparison_with_baseline` function allows you to compare learning metrics such as average reward, success rate, and learning speed against baseline values on a single plot with multiple y-axes.

Function Signature

```python
def plot_comparison_with_baseline(
    availability,
    df_learning,
    baseline_learning,
    accuracies=None,
    algorithm="Q-learning",
):
    # Function implementation
```

Parameters

- `availability` (**float**): Teacher availability level to filter the data (e.g., `0.8` for 80%).
- `df_learning` (**pd.DataFrame**): DataFrame containing IntRL learning results.

- baseline_learning (**tuple**): Baseline values (avg_reward, success_rate, avg_learning_speed).
- accuracies (**list**, optional): List of accuracies to filter by. If None, all accuracies are used.
- algorithm (**str**, optional): The algorithm type ("Q-learning" or "SARSA").

5 Using the Module

5.1 Importing the Function

First, import the function from the module:

```python
from utils import plot_comparison_with_baseline
```

5.2 Preparing the Data

Ensure your data is organised in a Pandas DataFrame with the following columns:

- "Availability": Teacher availability levels.
- "Accuracy": Accuracy levels to be analysed.
- "Avg Reward": Average rewards obtained.
- "Success Rate (%)": Success rates in percentage.
- "Avg Learning Speed": Average learning speeds.

5.3 Calling the Function

Use the function to generate the comparison plot:

```python
# Example baseline values
baseline_learning = (50, 80, 30)  # (avg_reward, success_rate, avg_learning_speed)

# Call the function
plot_comparison_with_baseline(
    availability=0.8,
    df_learning=your_dataframe,
    baseline_learning=baseline_learning,
    accuracies=[0.7, 0.8, 0.9],
    algorithm="Q-learning"
)
```

Parameters:

- availability: Set to the teacher availability level you wish to analyse.
- df_learning: Your prepared DataFrame containing the learning metrics.

- baseline_learning: A tuple of baseline values for comparison.
- accuracies: (Optional) A list of accuracy levels to include.
- algorithm: (Optional) The algorithm used, default is "Q-learning".

6 Example Usage

Here's an example demonstrating how to use the plot_comparison_with_baseline function:

```
import pandas as pd
from utils import plot_comparison_with_baseline

# Modified sample data
data = {
    "Availability": [0.7, 0.7, 0.7],
    "Accuracy": [0.6, 0.75, 0.85],
    "Avg Reward": [40, 48, 53],
    "Success Rate (%)": [72, 77, 82],
    "Avg Learning Speed": [22, 28, 33],
}

# Create DataFrame
df_learning = pd.DataFrame(data)

# Baseline values
baseline_learning = (45, 78, 29)

# Generate the plot
plot_comparison_with_baseline(
    availability=0.7,
    df_learning=df_learning,
    baseline_learning=baseline_learning,
    algorithm="Q-learning"
)
```

7 Troubleshooting

- **ValueError for Algorithm**: Ensure the algorithm parameter is either "Q-learning" or "SARSA".
- **DataFrame Issues**: Verify that your DataFrame contains all the required columns with correct data types.
- **Plot Not Showing**: Make sure to run the script in an environment that supports plotting (e.g., Jupyter Notebook or a Python script executed in a terminal that supports GUI).