Quickstart Guide

IT things

Access to Infrastructure

Request access to SC here. You can then access the server from within the uni vpn.

Python Environment

Install conda here.

Then install and activate the environment so that you have all the required packages:

```
conda env create -f environment

conda activate chess_ml
```

Download and Preprocess Data

1. Download and unzip the data with:

   ```
   mkdir data
   cd data
   curl -L -o lichess-chess-puzzle-dataset.zip\
     https://www.kaggle.com/api/v1/datasets/download/tianmin/lichess-
         chess-puzzle-dataset
   unzip lichess-chess-puzzle-dataset.zip
   ```

2. Preprocess the data on you local machine you can run:

   ```
   python -m chess_ml.data.transform \
          -i data/lichess_puzzle_transformed.csv \
          -o data/lichess_puzzle_labeled.csv
   ```

   if you are working on SC run:

   ```
   sbatch ./sbatch-data-preparation.sh
   ```

Library Usage

There are two core components you will use for training models: Imitation Learning and Reinforcement Learning.

Imitation Learning

You can either train a model locally with this:

```
# get help and see parameters to change
python -m chess_ml.train.imitation -h

# training a model
python -m chess_ml.train.imitation
```

Or you can run it on the slurm cluster

```
sbatch ./sbatch-training-imitation.sh
```

Reinforcement Learning

Again you can either train a model locally with this:

```
# get help and see parameters to change
python -m chess_ml.reinforcement.imitation -h

# training a model
python -m chess_ml.train.reinforcement
```

Or you can run it on the slurm cluster

```
sbatch ./sbatch-training-reinforcement.sh
```

Adjusting the Code

Not all parameters can be accessed from the cli, thus you may need to delve into the code. As a rule of thumb, those are the files and directories you may be interested in:

- `./chess_ml/train/imitation.py`: all the parameters for imitation training are here
- `./chess_ml/train/reinforcement.py`: all the parameters for imitation training are here
- `./chess_ml/env/Rewards.py`: contains reward functions you can use for reference
- `./chess_ml/model/`: contains model implementations:
    1. `./chess_ml.model.Convolution.ChessCNN`: cnn class
    2. `./chess_ml.model.FeedForward.ChessFeedForward`: linear layer class
    3. `./chess_ml.model.ResBlock.ChessResBlock`: residual block class

Environment and rewards

You can create a new environment with:

```
from chess_ml.env import Rewards
from  chess_ml.env.Environment import Environment

env = Environment(rewards=[...])
```

Where the reward parameter is a list of function names that should be evaluated at each step. The rewards functions are expected to have this function signature:

```
def reward(state: chess.Board, move: chess.Move, result: chess.Board):
    -> float
    '''
    state : state of the board before the player moved
    move  : move model suggested
    result: state of teh board after the opponent player
    '''
    return 0.0
```

The board is represented by with python-chess. Here is a quickguide, but feel free to look at the documentaion:

```
from chess import BLACK, WHITE, PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING
from chess import Board, Move
import chess

board = chess.Board()

# you can select squares
center = [chess.D4, chess.D5, chess.E4, chess.E5]

# you can get
# this is either chess.BLACK or chess.WHITE
current_player = board.turn
```