Report for Directed Study

Jessica Pauli de Castro Bonson

Index

1. Tasks Implemented

- 1. Multipurpose python version of SBB (pSBB)
    - Most features of pSBB (diversity, pareto, selection, hall of fame,...) were implemented in a general way, and can be used for other domains. It works for classification and reinforcement learning tasks, and the these environments are also extendable.
    - Since I rushed the code implementation at the end of the summer term, right now the code is a bit messy (mainly the poker part), but in case someone in the lab want to use it, it would take me no more than 1 day to clean it (or around 4 days, if the person wants the poker part too).
    - Features:
        - Diversity metrics implemented: genotype distance, fitness sharing, normalized compression distance, and relative entropy distance.
        - Operations implemented: '+', '-', '*', '/', 'ln', 'exp', 'cos', 'sin', 'if_lesser_than', 'if_equal_or_higher_than'.
        - Selection implemented: proportional selection
        - The runs are seeded, and I ensured that the same seed will produce the exact same run.
        - The hall of fame can be used for the reinforcement learning tasks. At the end of each generation the best team is added to the hall of fame. A pareto between fitness and diversity is used to define which team is going to be removed when the hall of fame is full.
        - At the end of the pSBB run the teams are saved in .json files, one for each team. The saved teams are: the ones in the last generation, the hall of fame, and the last pareto front. The .json file contains the team's programs and its instructions, and is easy to read and process. Eg.:


```
[
  - {
        action: 2,
      - instructions: [
          - {
                source: 6,
                op: "exp",
                mode: "read-input",
                target: 0
            },
          - {
                source: 1,
                op: "+",
                mode: "read-register",
                target: 0
            },
```

- 2. Poker environment
  - I integrated pSBB with the ACPC server, and used the python library pokereval to evaluate the hands.
  - ACPC server:
    - To decode the messages from the ACPC server I just used their protocol. The troublesome part is to handle the communication between the players and the ACPC server, eg.:
      - sometimes the server sends more than one message, merged together; or an empty message
      - the server don't inform you the results of a single hand, just the overall result of all hands played per run of the server, so you have to calculate it yourself (or just execute one hand per run)
      - the server don't inform the current bet and pot
      - when the match finishes, the server just shutdown the connection without warning
  - Inputs:
    - Inputs that lasts and are updated for a single hand: ['hand strength', 'EHS', 'pot', 'bet', 'pot odds', 'betting position', 'round', 'opponent hand aggressiveness']
    - Inputs that lasts and are updated for the whole generation or validation: ['chips', 'opponent long-term aggressiveness', 'opponent short-term aggressiveness']

    - 'hand strength': Works as described in this thesis: http://poker.cs.ualberta.ca/publications/davidson.msc.pdf, page 21
    - 'EHS':
      - Initially I implemented it as in the thesis above. The first problem is that the output for hand potential wasn't normalized, it produced values between 0 and 8. After normalizing it, the hand potential usually outputted results between 0.0 and 0.2, since even very good hands usually don't have that many evolutions, so it wasn't influencing the EHS much. Then I normalized it by its maximum value, but this way I noticed that the hand potential was just messing up the EHS, giving low values for hands that don't have much potential because they already are excellent ones (eg.: pair of A's). So I modified the hand potential formula so it would also give a good potential for already excellent hands. In short:
        - Old hand potential formula:
          - PPOT = (hp[behind][ahead] + hp[behind][tied]/2.0 + hp[tied][ahead]/2.0) / (hp_total[behind] + hp_total[tied]/2.0)
        - New hand potential formula (with normalization and adapted for excellent hands):
          - PPOT = (hp[ahead][ahead] + hp[behind][ahead] + hp[behind][tied]/2.0 + hp[tied][ahead]/2.0) / (hp_total[behind]*1.5 + hp_total[tied]*0.5 + hp_total[ahead]*1.0)
      - I also changed the original formula for EHS (= HS + (1-HS)*PPOT). The first motive was that with the original one PPOT always increase the overall result, even when the PPOT value is poor. The second motive is that even when PPOT had a very good value, it barely affected the EHS result. Currently I am using this formula (= (HS + PPOT * w)/(1+w)), with w = 0.5. But it seems too simple, so I may change it later.
      - The last change, is that PPOT can only be calculated for the flop and turn. So for the river EHS = HS. And for the preflop the normalized equity works as the PPOT, since it also is only relevant for the preflop round.

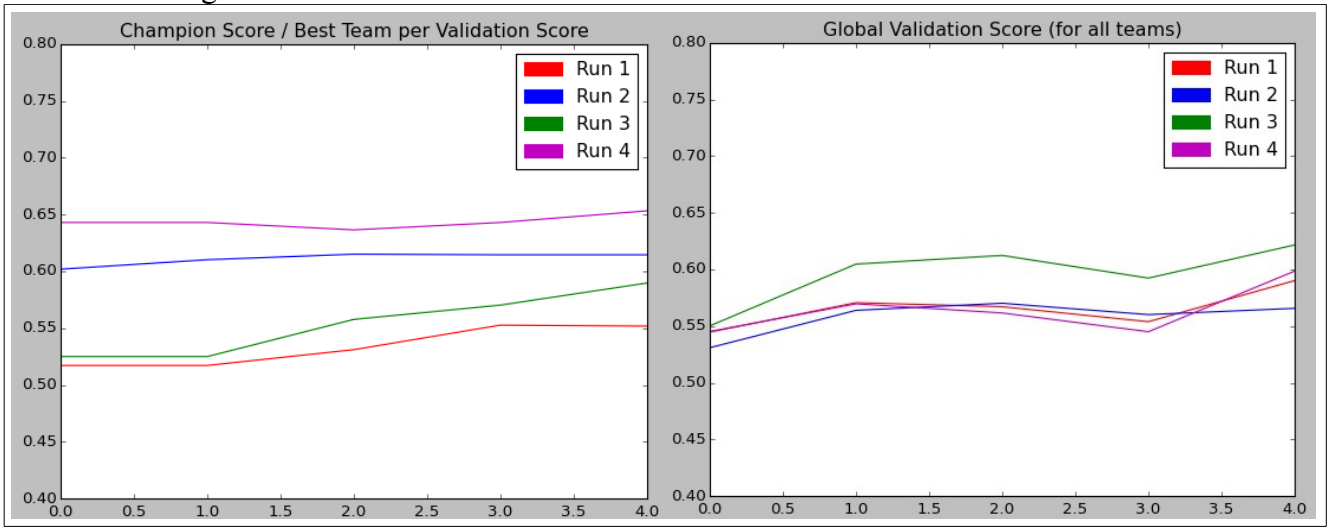
        - 'pot': The total chips currently in the pot.
        - 'bet': The amount to call.
        - 'pot odds': As defined in http://poker.cs.ualberta.ca/publications/davidson.msc.pdf, page 28
        - 'betting position': If the player is the first or the last to act in that round.
        - 'round': If is the preflop, flop, turn, or river
        - 'chips': The total chips the player has won or lost.
        - 'opponent long-term aggressiveness': As described in "Countering Evolutionary Forgetting in No-Limit Texas Hold'em Poker Agents"
        - 'opponent short-term aggressiveness': As described in "Countering Evolutionary Forgetting in No-Limit Texas Hold'em Poker Agents"
        - 'opponent hand aggressiveness': The mean of the actions of the opponent in the current hand. Calling is 0.0, and raising is 1.0.
    - Inputs implemented but not used:
        - Positive potential: used only in the EHS metric.
        - Negative potential: provided bad initial results, and the author of it didn't seem to like it neither.
        - Equity: used only in the EHS metric.
        - self long-term aggressiveness and self short-term aggressiveness: could be used by the teams to avoid being predicted (and thus explored) by complex opponents. Since right now the opponents aren't complex, this metric would be useless.
        - self short-term volatility, self long-term volatility, opponent short-term volatility, opponent long-term volatility: Volatility is a formula for how much the behavior of the opponent is changing before and after the flop. I am not sure if this would be useful, and probably would only be useful for complex opponents, so I am not using it yet.
    - Coded Opponents:
        - Dummy Opponents: Always Call, Always Raise, and Always Fold. Always Fold is not being used anymore since it produces random selection in their generations.
        - Rule-based Opponents: As described in the papers “Bayesian Opponent Modeling in a Simple Poker Environment, 2007” and “Can Opponent Models Aid Poker Player Evolution?, 2008”. I am using different alfa and beta values, since the authors don't say in their paper what is the 'winning probability'. So my 'winning probability' is the EHS formula. Alfa and beta values (may still be tuned later):
            - 'loose_agressive', 0.2, 0.4
            - 'loose_passive', 0.2, 0.9
            - 'tight_agressive', 0.8, 0.85
            - 'tight_passive', 0.8, 0.95
    - Fitness Function:
        - The fitness is the average of earnings in the hands in a generation or validation, given the maximum possible value of the pot (since this is a limited version of poker). The minimal earning (fitness 0.0) occurs if both players always raised, but the opponent wins in the showdown. The maximum earning (1.0) occurs if the opponent loses the showdown. A fitness of 0.5 means that the player neither won or lost chips. Eg.:
            - For a match with 'small_bet': 10 and 'big_bet': 20, the maximum a team can win is 240, and the maximum it can loose is -240.
            - So if a team has a score of 0.6 after 10 matches, it has won an average of 48 chips per match.

- How a run occurs:
  - Generation: In each generation X teams are going to play Y hands against a single opponent, selected at the beginning of the generation. The opponent may be one of the coded opponents, or a team in the hall of fame. At the end of the generation, the teams are processed using selection, part of the hands are replaced using uniform probability, and the hall of fame is updated.
  - Validation: X teams play against Z hands (usually 3Y), but now all the coded opponents are present. So if there is 3 possible coded opponents, each team will play against Z/3 matches against each opponent.
  - Champion: The team with the best validation score is the champion, and will play against W hands (usually 3Z). Again all coded opponents are present. Also, if the hall of fame is being used, the champion will play against each hall of fame opponent in extra matches, with 10 additional hands per opponent. These matches aren't considered to obtain the test score of the champion, they are just used to get an overview of how the champion performs against the previous best teams.

2. Initial Results

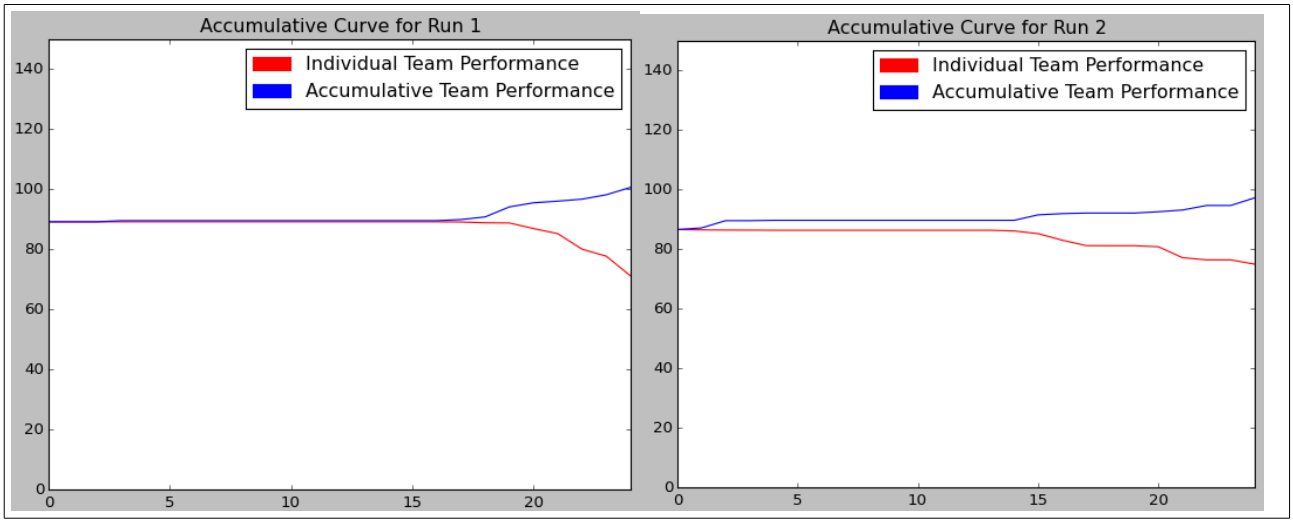
- Previous results:
  - This results will be summarized, to focus on the results that I got from the runs for today.
  - Dummy Opponents x Rule-based Opponents
    - For the dummy opponents, the fitness, validation and champion scores usually would go around 0.54~0.55 in less than 50 generations. And then they would be stuck, no matter the number of additional generations. The teams very rarely folded, and would focus in learning how to balance calls and raises.
    - For the rule-based opponents, the three scores would be around 0.49~0.51. The team behaviors varied more, and folds were more common, but it seemed that they were just behaving randomly. I guess that the tight opponents were too hard to beat, and caused the generation in which they appeared to produce random selection of teams.
  - NCD x Entropy
    - I am not sure if some of these diversity metrics is suitable to differentiate poker behaviors. Both of them sometimes produced weird solutions (entropy couldn't differentiate players that do different actions, but with the same probabilities; and NCD sometimes would produce results that don't make sense). Regarding scores, NCD was slightly better than entropy. I will execute more runs later to compare these metrics regarding the accumulative performance curves.
- Today results:
  - Since I finished implementing it earlier today, I had to run a quicker run, but the results seems to have improved.
  - Modifications:
    - Sampled and categorized 1000 hands. Categorized them by hole cards strength and (hole cards + board cards) strength. Since I am calculating the hand strength and EHS during the sampling, now the hand potential don't use heuristics and is more trustable. I also have access to metrics regarding the full board strength and the showdown. Sadly, this modification barely improved the runtime performance. In the next days I will sample and categorize more hands.
    - Validation and champion point populations are also balanced.
    - Option to balance the hand types by hole cards strength or (hole cards + board cards) strength.
    - Only using the loose opponents, when using the rule-based opponents.
    - Implemented accumulative performance per hand type.

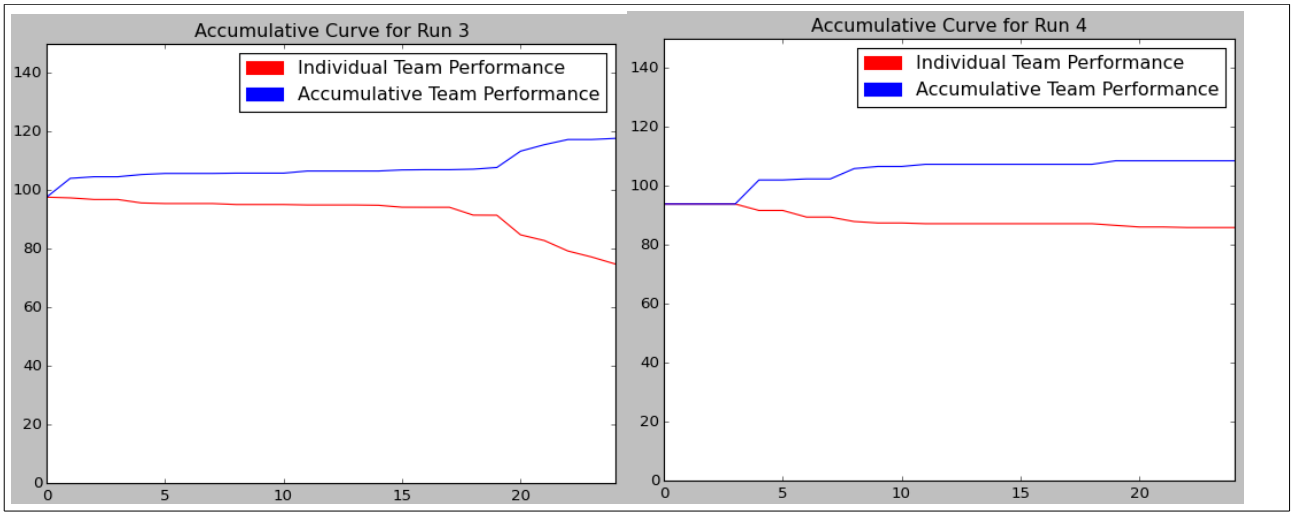
        - Tuned the EHS formula.
    - Parameters:
        - generations: 100
        - teams: 50
        - point population: {training: 50, validation: 150, champion: 500}
        - team size: 2~9,
        - program size: 2~10
        - diversities: NCD and genotype distance
    - I performed 4 runs, with these differing configurations, all with the same seed:
        - Dummy Opponents + hole cards strength balance (Run 1, red)
        - Dummy Opponents + (hole cards + board cards) strength balance (Run 2, blue)
        - Loose Opponents + hole cards strength balance (Run 3, green)
        - Loose Opponents + (hole cards + board cards) strength balance (Run 4, pink)
    - Results:
        - Validation Scores (for champions, and for all teams), for 5 validations between generation 1 and 100:



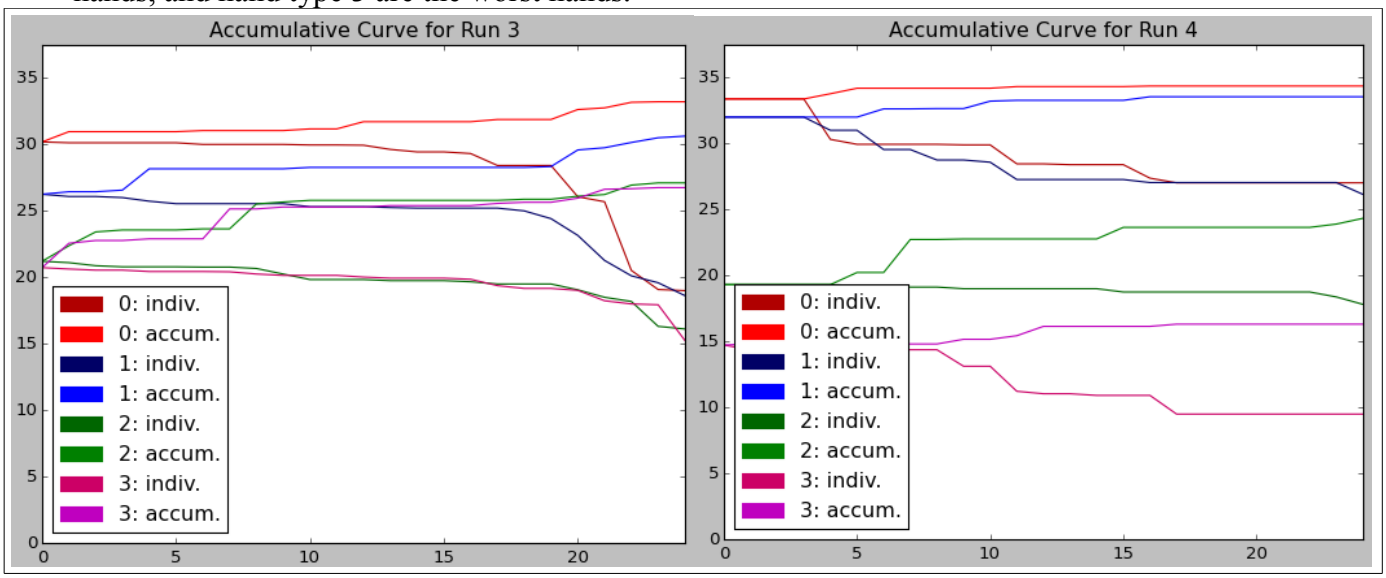
The first thing to notice about these charts is that all of them already performed better than the previous results for the global validation. For the champion score, only the red one didn't outperform the previous results. Also, at least for the global validation, they are improving over time instead of getting stuck around generation 50 (the third validation). I need to perform more and longer runs in order to find which one provides the better scores.

        - Accumulative Performance:





The runs with loose opponents are producing final teams with more diversity than the ones with dummy opponents. It makes sense that more complex opponents enables more varied strategies. So for my future tests I will focus on using the loose opponents. Besides that, I am not entirely sure if the curves for Runs 3 and 4 are good enough, and I would like feedback regarding them. Below there are the accumulative curves per hand type for the runs 3 and 4. Hand type 0 are the better hands, and hand type 3 are the worst hands.



3. Future Work

- In the next weeks:
  - Main Goals:
    - Find out if the current diversity metrics are being able to maintain diversity, based on the accumulative performance curves.
    - Find configurations that produce the best results (based on diversity and test score)
  - Tasks:
    - Implement more metrics (ie. Accumulative performance based on hands won, and with more subdivisions)
    - Try NCD using state information (right now it only uses the sequence of actions)
    - Define a final version for the EHS formula
    - Sample and categorize at least 5000 hands.
    - Run SBB longer for various runs and configurations
    - + fix and tune minor parts of the code
- For the next term (an overall idea, since we would meet before I start them anyway):

- If the diversity metrics aren't being good enough:
  - Implement and test new diversity metrics and/or implement a diversity metric specific for poker behaviors.
- Else:
  - Implement the second layer of SBB, with the coevolution of rule-based opponents and maybe a better opponent model.