COMP517 Multi-Agent Systems

Straygent: An agent with multiple personalities

C. Theodoris, 2011030032

Introduction

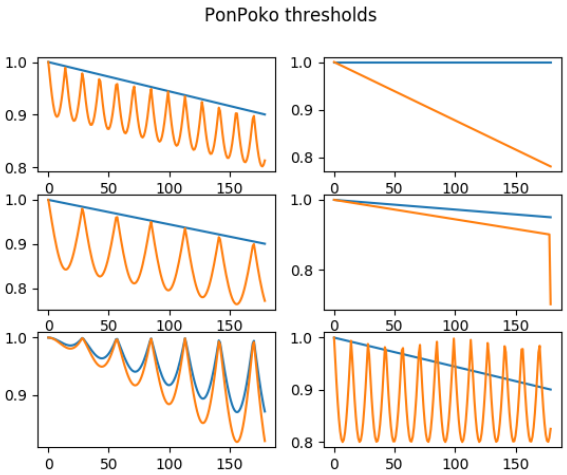
As a class project, we were asked to build an agent to compete in a local tournament, based on the Automated Negotiating Agents Competition (ANAC). The tools I used for this project were Java as the agent's language, Genious as the platform, Python and Excel for data extraction, analysis and plotting. After an extensive search on existing agents and papers, I was drawn towards the idea of multiple threshold profiles, as used on the PonPoko Agent[1]. The agent was simple, based on a randomized algorithm, emphasizing on the thresholds rather than an opponent model to offer and accept bids. I chose this approach, given that it was the winner of the 2017 competition, even without an opponent model and because I believe randomized algorithms to be fairly powerful. Adding an opponent model was attempted, but the results were worse on average.

General Idea

Straygent is a slowly conceding agent, that uses a set of four threshold profiles called psyches (as in mentality). Those profiles are chosen every round, by a weighted random selection, with weights that represent the mood of the agent. As game time passes, the agent changes mood and we see the appearance of some psyches more than others. Using this simple concept, the agent makes a selection of bids within its thresholds from a bid list, prepopulated when the game starts, choosing randomly one to offer and accepting bids higher than its lower threshold.

Psyches

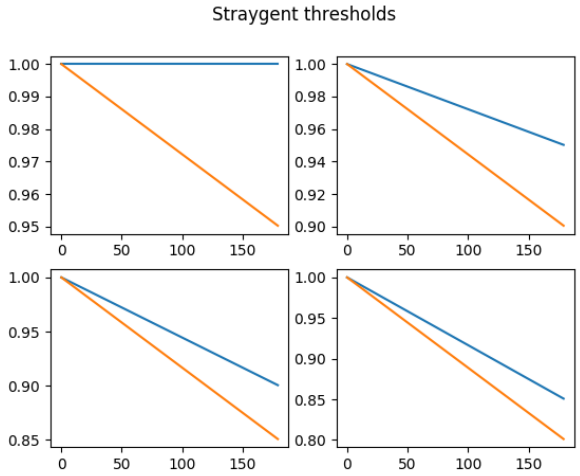
Psyches are multiple threshold profiles, similar to PonPoko Agent's[1], key difference being that the former chooses a **game** profile in each game, while Straygent uses a **round** profile in each round. This small difference changes the whole scope of the agent; Ponpoko uses a straight-forward profile in each game, with a fixed pattern, whereas Straygent is "wild" (within bounds of probabilities). The randomized play style achieves a greater defense to Bayesian Agents, who reverse engineer your offers, to exploit your strategy. The difficulty in playing randomized is to bind the choices to your general strategy (for this agent, a slowly conceding strategy). This will be addressed in the next paragraph. Pondering on what thresholds are good for Straygent, I plotted PonPoko's threshold profiles:



Ponpoko's thresholds are complex and different among them. This allowed the agent to be prone to exploiting, if and only if an opponent remembered the agent's playstyle. Trying these thresholds on Straygent yielded bad results, simply because at any time it could hit the wider spread threshold regions, making the list of valid bids between the thresholds really big, with a wide range of utilities. With these results in mind and numerous tries to make them mathematically fancy failing, I returned to the simplest form of lines:

$$f(x) = a \cdot x$$

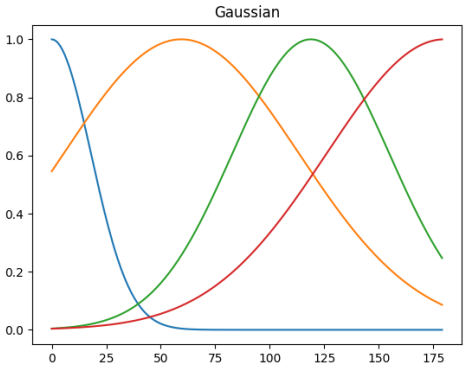
Using these simple lines, I could easily control at my will the spread and conceding rate over time. The final psyches are:



They have a 5% spread and cover the region 1 - 0.8 utility (in the last round) in four steps. This means that the agent should accept and offer bids of utility of 0.8 at least. To cover the scenarios where Straygent doesn't have a bid in the valid region (which happens a lot in the early game), and to be able to make an offer every round, the lower threshold is gradually lowering by 0.01%, until we meet at least one bid. This usually results in a minimum of 2-4 bids, since the bid space is huge, and even this small decrement is enough to find many new bids.

Mood

The way the above thresholds are weighted in the random selection is similar to an average person's mood in games like this. One starts hopeful that he can win the game with huge earnings, bidding his higher utility bids, then tries to bluff midgame his way to victory, trying different bids of high utility to see how the others react and finally, a little dishearted, lowers the value as much as he thinks is still ok. To modelize this kind of behaviour, I thought of the probabilities of each psyche as Gaussian probabilities, each stronger in one of the 4 stages of the game (early/arrogant, early-mid/condescending, late-mid/confused, late/desperate). After tinkering with the parameters, until a satisfying result, the Gaussians are:



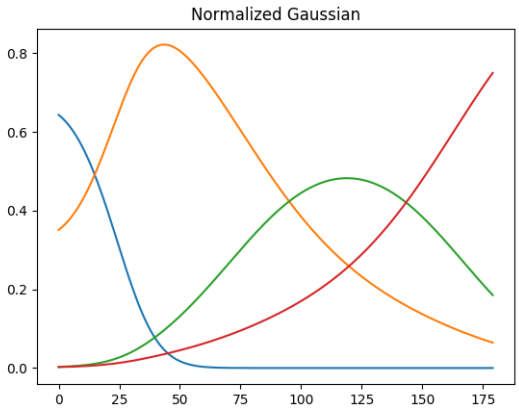
Here, we can see the agent's mind. Starts strong (1-0.95 utility), then realizes that he has to lower his standards (0.95-0.9), then is a little frustrated (0.9-0.85 with a high chance of previous and next mood) and finally trying to close a deal (0.85-0.8). The only problem with these functions is that at any point in time, they don't sum up to 1, which calls for a normalization in the range 0-1.

Simple Normalization

The first method that came to my mind to normalize the Gaussians, was:

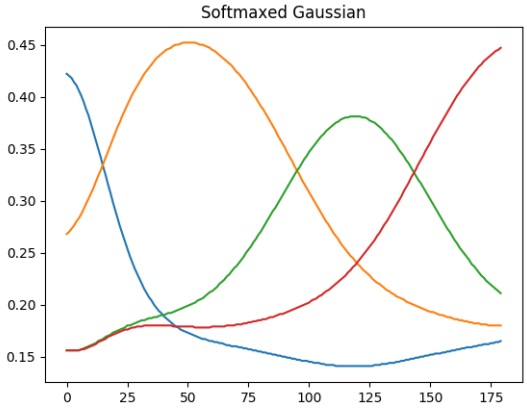
$$Value_{Norm}(t) = \frac{Value_{Gauss}(t)}{\sum(Values_{Gauss}(t))}$$

This transforms the Gaussian values to the percentage of the Sum of all the values at a specific point in time. This effectively makes them probabilities out of raw values.



While this approach yielded good results on average, it was too absolute. It led to guaranteed worse results at the passage of game time, making it vulnerable to Boulware strategies. To make the agent more agile and unpredictable, especially in the later stages of the game, I sought another way to translate my data in percentages, one that, at any point in time, values all the moods as somewhat valid options. This healthy dose of uncertainty and insanity for my agent was found in sigmoid functions[2] and specifically in softmax function[2].

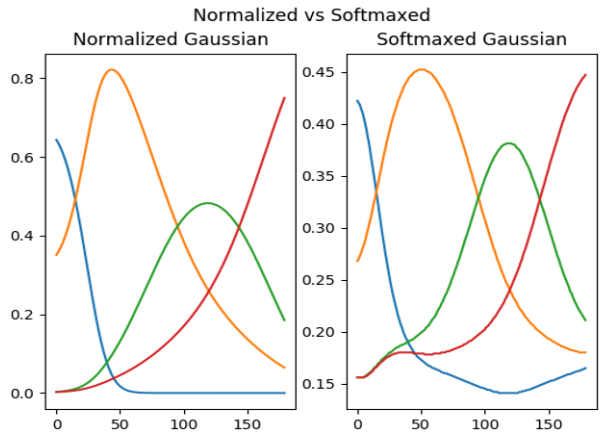
Softmax Regression



Softmax function is the generalized Sigmoid or Logistic function for multiple classes in regression. In Machine Learning[3], it is often needed to classify inputs in many classes, based on the probability of the input being one or the other. The job to assign the probabilities is done by Softmax. It basically transfers the values given in the range 0-1, using exponentials in the simple method that I tried first:

$$Value_{SM}(t) = \frac{\exp(Value_{Gauss}(t))}{\sum_i (\exp(Value_{Gauss(i)}(t))}$$

Using this method, I achieved lower highs and higher lows, meaning that every psyche can come out, at any point in time, but on average the most probable are still those that follow the mood of the agent. It is also admiring how Softmax transfers the essence of the Gaussians better (i.e. the third mood) than the simple normalization, which is obviously why it is used in regression in the first place:



Weighted Selection

Using the probabilities given by the softmax, I implemented a simple weighted random selection:

```
weightList.add[weight(0)]
for i in [1:3]:
    weightList.add(weightList(i-1) + weight(i))
mood = rand(0,1)
```

Which then finds when the mood is less than the weight and gives the appropriate psyche.

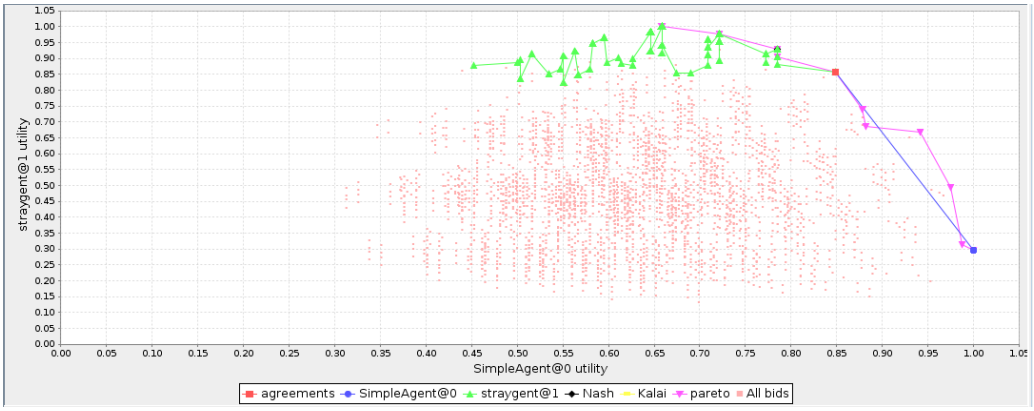
Opponent Modeling

There was an attempt to model the game using a frequency model, where the agent saved which bids have appeared before, putting them on top of the list of all the bids and trying to play the best of those (utility-wise) at the late stages of the game. It didn't assign weights to each element of the bid based on the frequency (as could be done), because the complexity of the task would burden the agent's times. Refinement could certainly be possible, but all the research and the other implementations took up all of the available time.

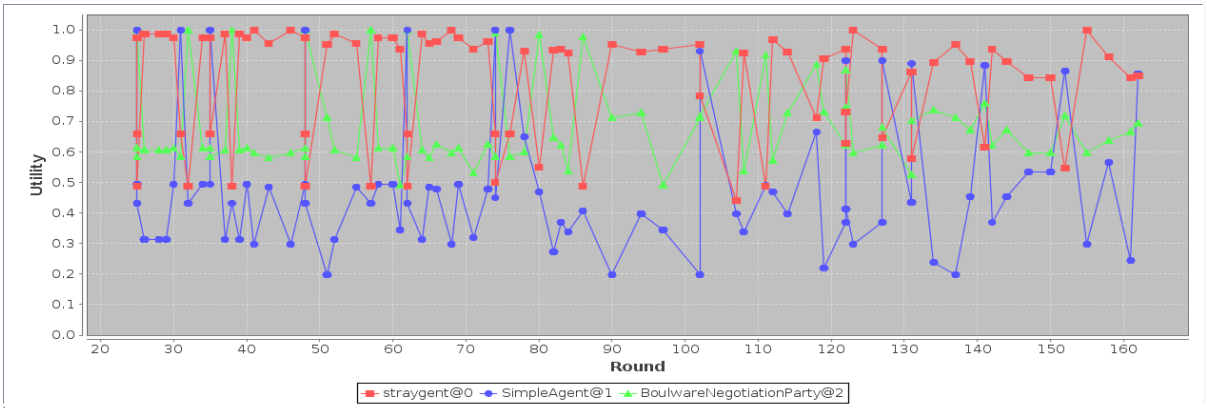
Testing

Most of the testing was done in the party domain, against several agents. Straygent's main opponents were 5 agents: PonPokoAgent, SimpleAgent, BoulwareNegotiationParty, ConcederNegotiationParty and RandomParty. They were categorized into two categories, weak and strong, based on the results of their matchups. The first two were considered strong opponents, with PonPoko being hard headed and SimpleAgent (a Bayesian agent) often closing slightly better deals for himself. The other three were weak opponents, being simpler agents that mostly closed deals fast with lower utility. Straygent was competing with the other agents in 1v1 matches (to monitor the exploration of its bids, Pareto and Nash distances), or in 1v1v1 matches, as ANAC indicated, both in Negotiation and Tournament modes, to monitor its overall score.

Some of the results (picked at random, not the best or worse) are given below:



A game against SimpleAgent, showing that Straygent explores well the bid space and hits the Pareto Frontier



Time (s):	1.632804893
Rounds:	162
Agreement?:	Yes
Discounted?:	No
3	
Min. utility:	0.69241
Max. utility:	0.85649
Distance to pareto:	0.00000
Distance to Nash:	0.09921
Social welfare:	2.39723
Agent utility:	0.84833 (straygent@0)
Agent utility:	0.85649 (SimpleAgent@1)
Agent utility:	0.69241 (BoulwareNegotiationParty@2)

A game against SimpleAgent and BoulwareNegotiationParty, showing that Straygent has a little less utility than SimpleAgent and achieving to hit the Pareto Frontier while having great social welfare

Agreements:	110
Rounds:	504
%:	21.82539683

Utility of agreements:	
Stray	0.8908369397
PonPoko	0.8645651794
Simple	0.8913359688

Avg. Min:	0.846627
Avg. Max:	0.917739273
Mean:	0.882183136

Utility of all rounds:	
Stray	0.8908369397
PonPoko	0.8645651794
Simple	0.8913359688

Agreement Rounds		
Avg. Pareto	Avg. Nash	Social Welfare
0.0050938181818	0.057332909	2.64651090909

Total Rounds		
Avg. Pareto	Avg. Nash	Social Welfare
4.273481727	5.220236364	2.6465109091

Results from a tournament with strong Opponents

Agreements:	504
Rounds:	504
%:	100

Utility of agreements:	
Stray	0.9012367937
Conceeder	0.63543212094
Boulware	0.80926218168

Avg. Min:	0.615078988
Avg. Max:	0.908712758
Mean:	0.761895873

Utility of all rounds:	
Stray	0.9012367937
Conceeder	0.63543212094
Boulware	0.80926218168

Agreement Rounds		
Avg. Pareto	Avg. Nash	Social Welfare
0.0241769047619	0.261514802	2.3459311706

Total Rounds		
Avg. Pareto	Avg. Nash	Social Welfare
0.024176905	0.261514802	2.34593117063

Results from a tournament with weak Opponents

As we can see from the results, Straygent is a hard-headed agent, even more than his predecessor PonPoko, as is shown here. When he agrees, the overall game is Pareto optimal, but matching against hard-headed opponents as well, offers the worst results. Those results were expected and wanted, as I aimed to make the agent value its utility levels more than the need to close a deal. Straygent overall is a fair winner and a sore loser.

Local Tournament

In the local tournament, Straygent held the third position in his bracket and the fourth position in the finals. A major reason why this happened is the hard-headedness. I can only assume that the utility measure we were evaluated with, was the total utility of all the rounds, including agreements and disagreements. Even though I disagree with such a "flat" metric, when there exist so many data to evaluate, I am proud of the fourth place, given how stubborn my agent is.

Final Thoughts

A posteriori I would probably fine-tune the thresholds to get more agreements at the late game, as many other agents do. One way to implement this, which is a window for experimentation and improvement in the future, would be to use the frequency based opponent model to change the hyperparameters of the psyches and moods, dynamically, after a certain stage of the game, while it also adds weights to the bids' elements to improve the bid selection algorithm, which currently is totally random.

References

[1] PonPokoAgent, Takaki Matsune (Tokyo University of Agriculture and Technology, Japan), ANAC 2017

[2] Sigmoids in Logistic Regression: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

[3] Softmax in ML: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax