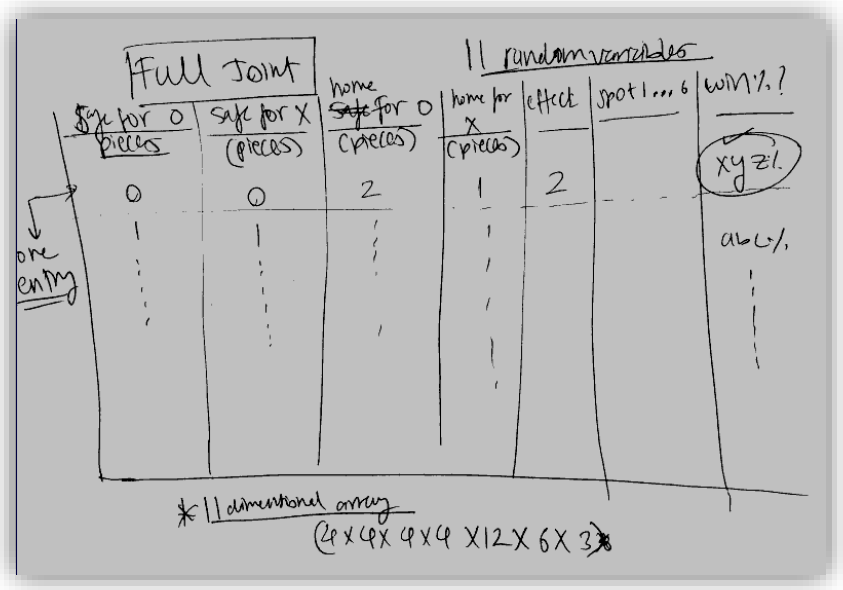
Shyamal H Anadkat : HW3 Discussion

1) FullJointProbTablePlayer Discussion



My full joint represents various random variables which I assumed would be an important factor in winning Nannon. I made two 11 dimensional arrays - one for representing the state for wins, and one for losing states. The 11 dimensions (direct mapping) I considered as my random variables are : pieces at Home for X [4], pieces at Safe for X [4], pieces at Home for O[4], pieces at Safe for O[4], effects[12], and the state of each 6 positions [each with 3 possible values] on the resulting board. Note that I considered the resulting board and the next move while making a good choice for the best move. With the resulting board instead of considering the current board, I get the freedom of logging even the last piece on the winning team that reaches the safe position. I tried a full joint with current board earlier but the success ratio was a bit lower. In the choose-move method, for every legal move, I determine the best winning ratio for the config of the resulting board while in update statistics, I update/increment the win and lose state tables in case of a win and lose respectively. After learning, the table more or less becomes a look up table for best winning config for a particular legal move.

My learned model outputs the best and worst configs (global) for the win - and again this is a good sanity check and helped me understand how to debug probabilistic models in future.

Some of the things that I realized : initially I considered the die value as one of the random variable for evaluating the next move; however, I attained more success when I replaced the die value with the effect. In case of full joint, my choice of random variables to be considered mainly relied on the success attained as well as intuition of what would determine the next best move.

Overall, my full joint is a clear winner and beats all the other players. One of the reason is that I

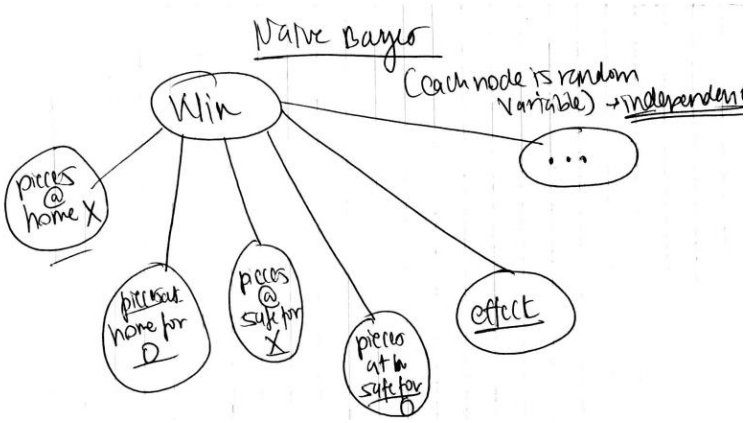
have pretty much covered all the relevant random variables which could map to the next move in the game, and hence the model becomes smart pretty quick. Also I would say for a game like Nannon, there aren't much dependencies and hence a Full Joint would perform better than Bayes Net in general.

Here's a snippet demonstrating learned model output:

```
Shyamal's FullJointProbTable Player learned and reporting a full joint probablity reasonor for Nannon.

Best Values for Position 1 through 6 on board: 2,0,2,0,0,1
Best Value for effect: 0
Best value for pieces at Home for X: 0
Best value for pieces at Safe for X: 2
Best value for pieces at Home for O: 0
Best value for pieces at Safe for O: 0
Best Die Value: 6
Bad Values for Position 1 through 6 on board: 0,2,2,0,0,1
Bad Value for effect: 0
Bad value for pieces at Home for X: 2
Bad value for pieces at Safe for X: 0
Bad value for pieces at Home for O: 0
Bad value for pieces at Safe for O: 0
Bad Die Value: 1
--------------------------------------------------------------------------------
```

2) Naïve Bayes Discussion



For my NB player, as suggested, I have random variables other than win. I look at each individual random variable and determine which has the highest ratio of prob(randVar=value | win) / prob(randVar=value| loss). I considered all the possible values for the random variables when computing this ratio of probabilities. The random variables (arrays) which represented my NB model includes: home pieces for X, home pieces for O, safe pieces for X, safe pieces for O, effect[12] for win, and effect[12] for loss. Further, I calculate their conditional probability : P(random variable given win/lose | win/lose). Then, to get the best odds for win, we apply the Naïve Bayes independence assumption and multiply the winning conditional probabilities and divide that by the product of losing conditional probabilities (win product/lose product ratio).

When we win a game, the update-statistics method will increment the win counts or rather success occurrences of the random variables and vice-versa for when a game is lost.

My NB model works effectively overall, and beats the random player around 63% of the time. Note that this is lower than Full joint as I have not considered the board position states in this model.

One thing I realized about this model is that the training can be done in linear time, and also that we use lesser space (and is scalable more easily) than we used in full joint as we are only storing the RV probabilities here and assuming complete independence between all the random variables considered.

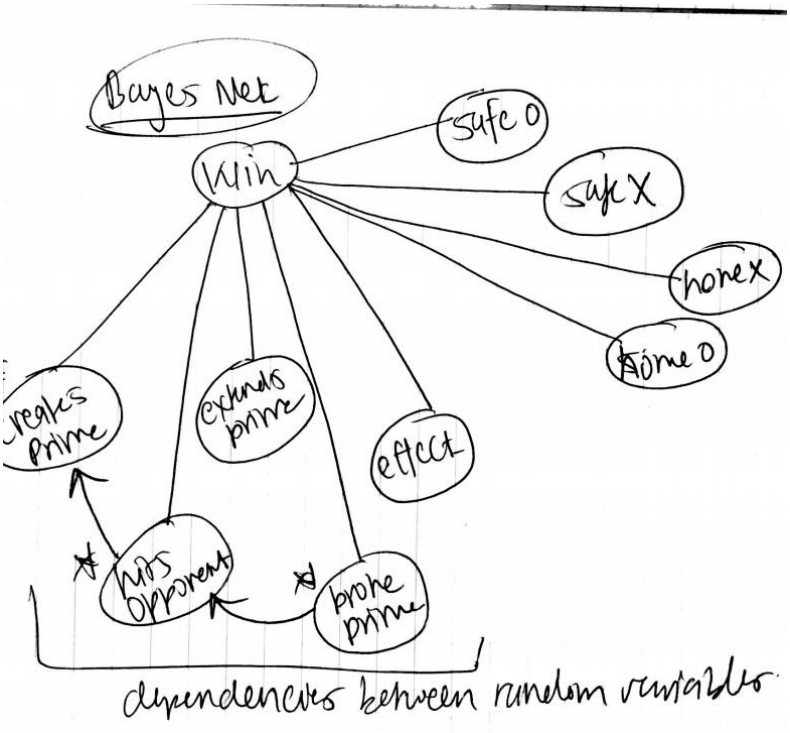
Random variables considered for NB summarized:
* effect for win
* effect for lose
* home pieces for X
* safe pieces for X
* home pieces for O
* safe pieces for O

Here's a snippet demonstrating learned model output:

```
------------------------------------------------
Shyamal's Naive Bayes Playerlearning model !!

Best Winning Ratio for effect: 3.107246376811594
Best Winning Ratio for pieces at Home for X: 1.6089922100226954
Best Winning Ratio for pieces at Safe for X: 1.4627254071519016
Best Winnning Ratio for pieces at Home for O: 33.10664605873261
Best Winning Ratio for pieces at Safe for O: 1.5273751924994818
Worst Ratio for effect: 0.5072298943948009
Worst Ratio for pieces at Home for X: 0.29854444154719917
Worst Ratio for pieces at Safe for X: 0.6621975147155003
Worst Ratio for pieces at Home for O: 0.7036082201576053
Worst Ratio for pieces at Safe for O: 0.5305015241726528
------------------------------------------------
```

3) Bayes Net Discussion



Here I look at the part of the non-naïve Bayesian net that is different, and only consider nodes that have more than WIN as their parent, as suggested on the Piazza forum. While I didn't include more dependencies (or rather a complete directed acyclic graph), I made sure I was getting satisfactory success ratio using the dependencies I considered for the chosen move.

The design for Bayes net is similar to Naïve Bayes model considered previously; however, we consider joint probabilities - the 2 dependencies I considered is between certain move effects – hit opponent and broke my prime, and between hit opponent and create my prime.While choosing a next move (for each win and lose case), I considered probability(joint) of both hit opponent and broke my prime happening and multiplied that with other conditional probabilities for the independent variables (again for lose and win case) and then the best odds is given by win/lose ratio as earlier. The only additional thing we multiplied numerator and denominator by is the joint conditional probability of both effect moves happening.If I were to extend the scope of my model, I would also have considered dependencies between the various values for pieces at Home and Safe for X and O.Overall, the Bayes Net is satisfactory, and wins the random player around 60% of the time, but I would say it is not yet strong as I haven't considered more dependencies and the state of the individual board positions.

Random variables or nodes used for my Bayes Net summarized:

- Home pieces for X [4] – 0, 1, 2, or 3
- Home pieces for O [4]

- Safe pieces for X [4] *Note: using resulting board*
- Safe pieces for O [4]
- Hit opponent, broke my prime, extends prime of mine, create prime
- Hit opponent and broke my prime (NON-NB dependency)
- effect_lose[12], effect_win[12]
- home pieces for X and O lose, home pieces for X and O win
- safe pieces for X and O lose, safe pieces for X and O lose

Here's a snippet demonstrating learned model output:

```
Shyamal's Bayes Net Playerlearning model !!
Best winning odds: 40.32554609293061

Best Winning Ratio for effect: 2.993192823300934
Best Winning Ratio for pieces at Home for X: 6.504728676812708
Best Winning Ratio for pieces at Safe for X: 1.4258593796194192
Best Winnning Ratio for pieces at Home for O: 1.6198445818266458
Best Winning Ratio for pieces at Safe for O: 1.6455818235770503
Best Winning Ratio for Break and Hit Opponent : 1.2789362416984649
Best Winning Ratio for Create and Hit Opponent dependency : 1.703998228182368
Worst Ratio for effect: 0.6191296029094445
Worst Ratio for pieces at Home for X: 0.8130106959147764
Worst Ratio for pieces at Safe for X: 0.662633870744753
Worst Ratio for pieces at Home for O: 0.3838776607384299
Worst Ratio for pieces at Safe for O: 0.6935476967740715
Worst Ratio for Break and Hit Opponent : 1.0
Worst Ratio for Create and Hit Opponent dependency : 1.0
-------------------------------------------------
```

Having a dependency between hit opponent and broke my prime move is a good idea as they are intuitively related plus including them boosted my success ratio for this model. Here is a small figure that illustrates my BN model with nodes:

4) Performance of players against each other

	FullJoint	NaiveBayes	BayesNet	HandCoded	Random
FullJoint		55. 9%	60.05%	53.22%	66.77 %
NaiveBayes	44.03%		55.61%	48.85%	62.49%
BayesNet	39.95%	44.59%		44.17%	59.81%

Brief discussion of results: My FullJoint, NB as well as Non-NB Bayes net performs better than the random by at least more than 10%, Full Joint wins overall with around 67% against random, 62.5% against NaiveBayes and almost 60% against BayesNet. We see that Bayes Net loses against greedy, this is again because my Bayes net only considers two dependencies and could have been more stronger. Also, its important to note that I do not consider the positions of pieces on the board as random variables in the case of Naïve Bayes and Bayes net otherwise they would have likely performed better against random as well as greedy. There are two reasons why I

didn’t consider the board states in case of NB and Bayes: firstly, I wanted to explore other variables like the effect-moves and their dependencies, and second the test-bed could run on a bigger size Nannon board and I did not want to be limited by that factor, hence I tried to make my model to be board-size-agnostic. The clear winner is Full Joint.

Snippet of random vs Naïve-Bayes.