Deep Learning for Connect Four

Max Marcussen

Project goals

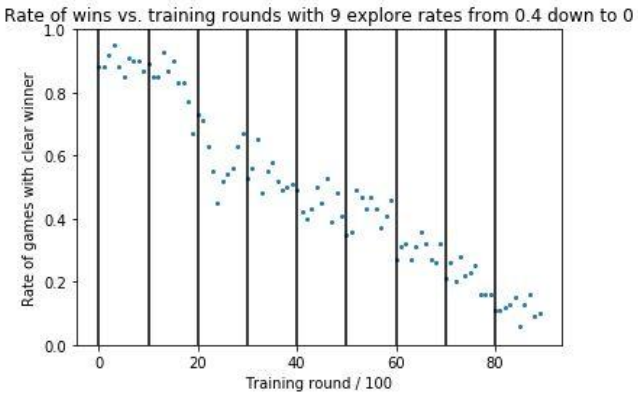
- Train as large of a game board with tabular Q-learning as possible, as efficiently as possible.
- When tabular Q-learning fails, move to deep Q-learning.
- Learn game sizes up to Connect Four (6x7, 4 in a row to win) as efficiently as possible, with good gameplay.
- Determine a success metric for Connect Four algorithm’s performance.

Recap of tabular Q-learning

- Good for small games, scales exponentially for large ones.
- Time and space complexity make anything over a 4x4 board not worth training.

Tic-Tac-Toe, adding gravity, 3x4

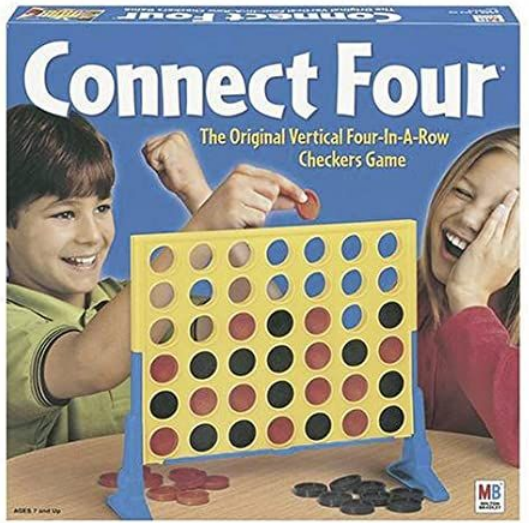
- Very small versions of Connect Four.
- 3^9 = ~20,000 possible states
  - An exploration rate of 0.2 should discover all states within 100,000 training rounds.
- If both players play perfectly, a tie can be reached in every game.
- Can do 10,000 rounds of training in 1 minute with regular Q learning.
- Adding gravity and adding a moving learning rate and exploration rate increases efficiency tenfold.
- Tabular Q-learning worked remarkably well.



Connect Four

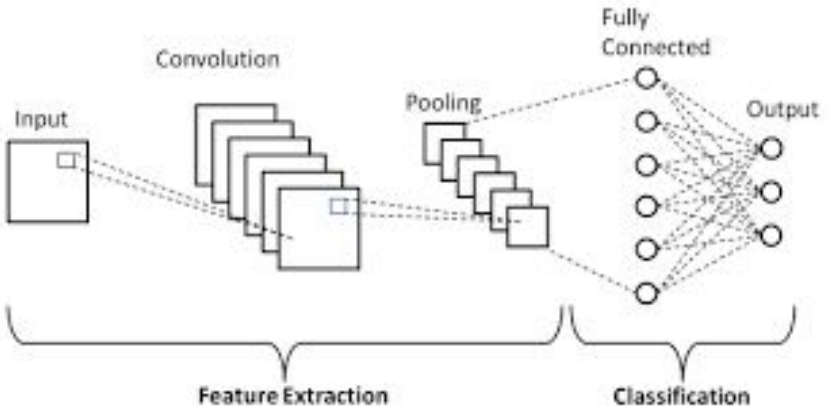
Actually the same game as Tic-Tac-Toe with gravity, just with slightly updated format

- Larger board (3^{42} states vs. 3^{9})
- 4 in a row vs. 3 in a row
- Gravity restriction on movement



Deep Q-learning: network architecture

- Settled on CNN as my architecture.
  - Most often used for image processing, but board is a 2x2 image.
- Filters equal in square size to number of pieces needed to win.
- After convolutional layer, two fully connected layers with # neurons == board size.
- Output layer with # neurons == # columns (possible actions).
- ELU activation functions throughout.
  - Common practice, others performed worse in testing
- Sigmoid output activation
  - All rewards are between 0 and 1

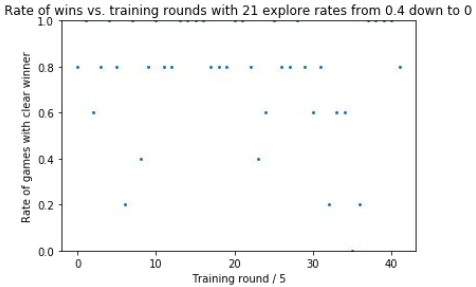


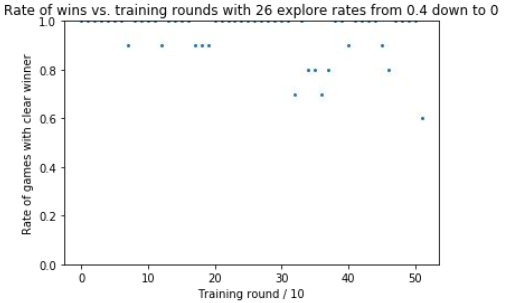
DeepQ: how the update step works

- Neural network outputs a set of action values given a state.
  - For a 6x7 board, will output 7 action values.
- Compute “real” action values for this state from “predicted” action values and reward from next state.
- Fit neural network to updated action values.

How does DeepQ perform on a 3x4 board?

- Limited myself to gravity games.
- Converges to a good mix of wins and ties.
  - Never happened for tabular Q-learning.
  - Learns generally good, not perfect, strategies
- Still exponential (but not as large) increase in time with board size
  - 3 more spaces ~= 6x train time (5 mins vs. 30)
- Overfitting is a huge problem
- Development process extremely buggy overall
- Performance very much based on where weights randomly start



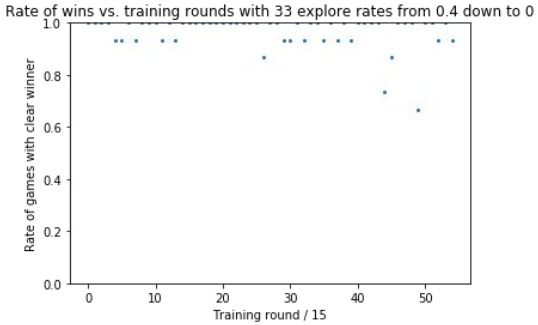


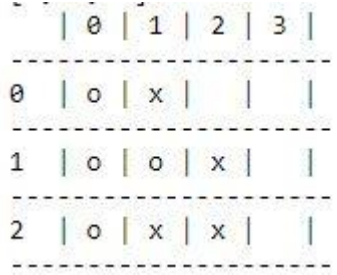

```
  | 0 | 1 | 2 |
---------------
0 | o | o | x |
---------------
1 | x | x | o |
---------------
2 | x | x | o |
---------------
```

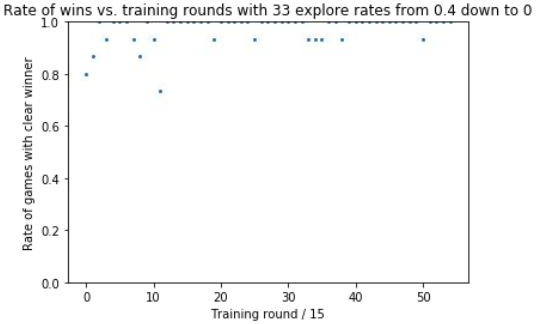
```
  | 0 | 1 | 2 | 3 |
-------------------
0 | o | x |   |   |
-------------------
1 | o | o | x |   |
-------------------
2 | o | x | x |   |
-------------------
```

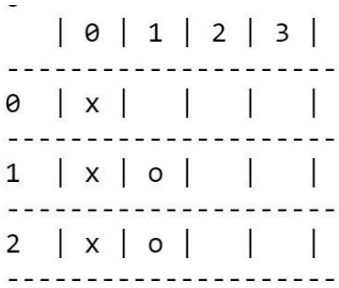
Impact of randomization

- Two training sessions can be given exact same parameters and come up with completely different gameplay.
- Issue that plagues deep reinforcement learning — a lot of researchers will aggregate over many training sessions or will provide random seeds that work well.
- Difficult since training times are so long



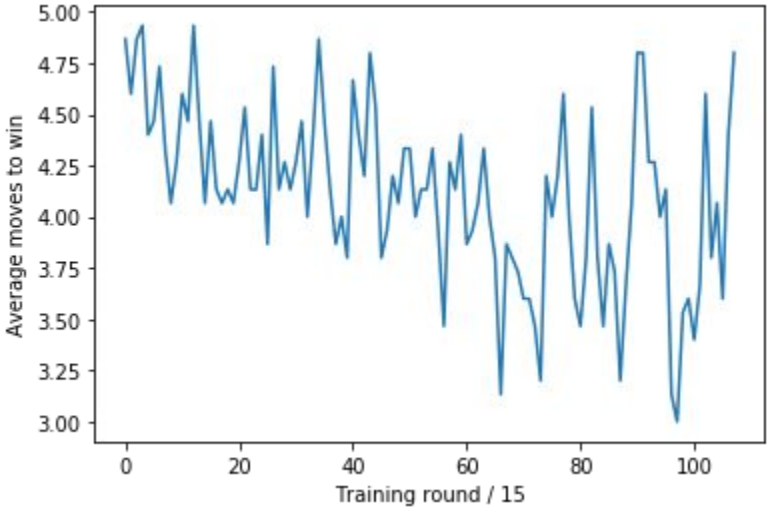






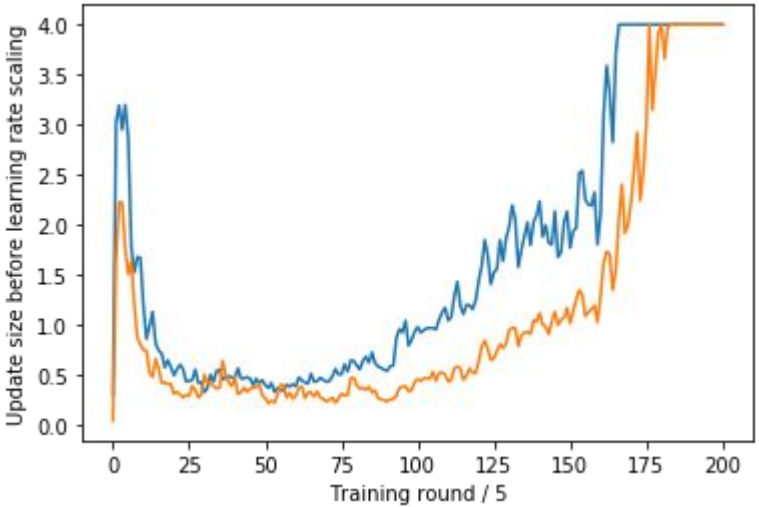
New success metric

- Tie rate not necessarily the best. Both players can play perfectly and one can still be winner.
- New metric: moves to win
  - Players do better if they make the game last longer — prevent other player from winning, build their own long-term strategies, etc.



Overfitting

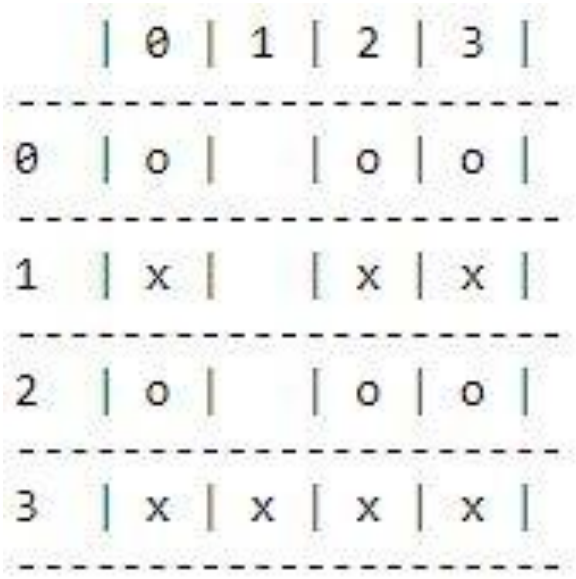
- Updates to action values might balloon out of control if we've trained for too loing and some network update goes the wrong direction or if not enough information.
  - Was more frequent at lower exploration rates.
- Always maintain a slight exploration rate.
  - Would be problematic for tabular, not problematic for deep
- Randomly initialize board state a bit.
- Don't train for too long.
  - Also don't train for too short.
- Set a maximum on update size
  - Prevents even larger future updates — brings updates back to realm of normalcy
- Change activation function to sigmoid




```
   | 0 | 1 | 2 |
-----------------
0  |   |   | x |
-----------------
1  | o |   | x |
-----------------
2  | o |   | x |
-----------------
```

DeepQ on a 4x4 board

- Difficult to avoid overfitting and hard to properly specify training length
- DeepQ is highly sensitive to even small changes in starting learning rate, exploration rate, or even where weights randomly start.



Findings from Deep Q-learning experiments

- Number of games far fewer for Deep Q
  - Gravity-based 3x3 board took 4500 games to play for tabular Q. Only 500 for DeepQ with same results.
  - Number of games needed doesn’t increase exponentially with board size
  - Drawback is that each game takes far longer (milliseconds vs. seconds)
- Game complexity is taken care of by network complexity rather than training time
  - If we scale up our network complexity with game complexity, we’re able to handle larger boards easily.

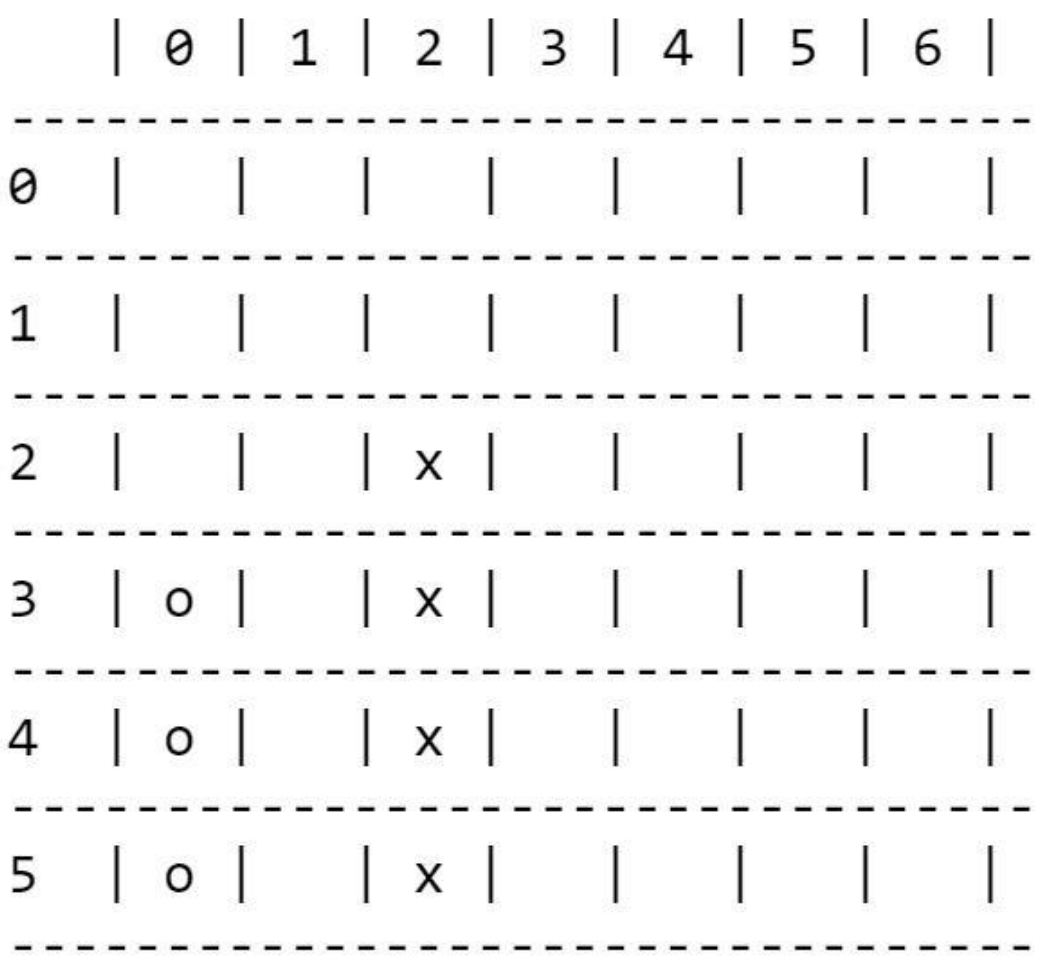
Findings from Deep Q-learning experiments

- Important to not use an excessive number of convolutional filters, especially for small board sizes
  - Can lead to confusion
  - More filters requires lower learning rate
- Reward scheme is important
  - Need to make sure player 1 (first mover) is relatively OK with ties
- Randomness is a huge issue for DeepQ
- High exploration rates are important
  - Helps to learn more possible board states, more strategies that can lead to victory

Let’s play Connect Four!

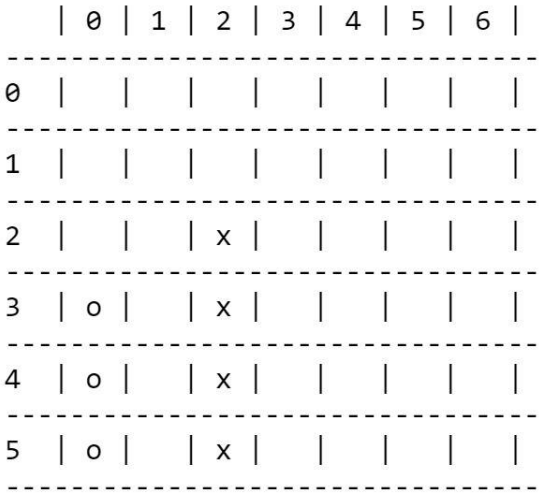
16 filters, 4x4 each, two FC layers with 42 neurons each, overfitting controls, maintain slight consistent exploration rate.

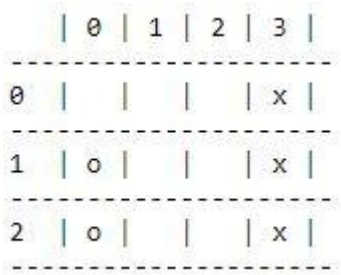
How did we do?


| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
0
1
2 x
3 o x
4 o x
5 o x

What happened?

- This is actually a stable strategy for both players.
  - Constant exploration rate makes this feasible
  - Discount factor (gamma) makes getting wins sooner preferable.
  - This happens with smaller games as well if not trained for long enough (3x4 with <500 games, for instance).
- Solutions
  - Randomly initialize arbitrary number of moves at higher exploration rate
  - Penalize winning very quickly
  - Give higher rewards for winning after more moves

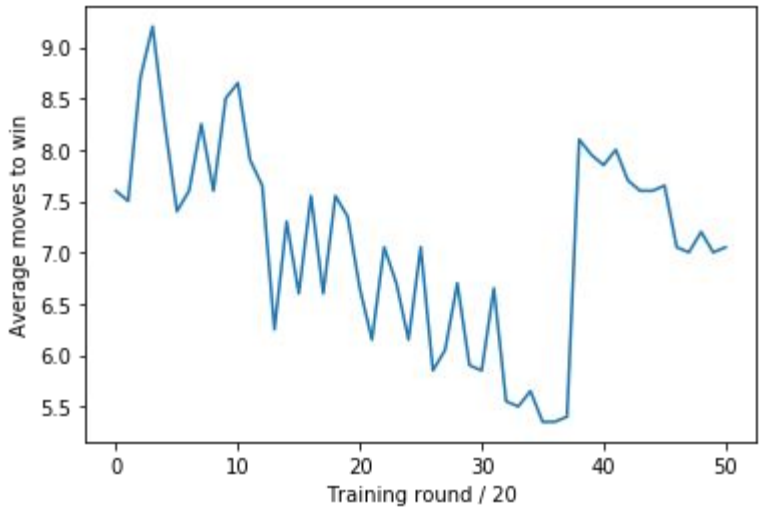




	0	1	2	3	4	5	6
0							o
1							x
2				x			o
3	o			x			x
4	o			x			o
5	o			x			x

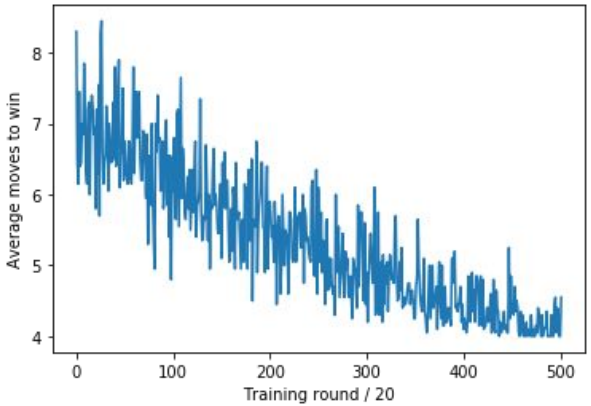
What happened?

- When we penalize winning early, network just stalls until it can evade penalties.
  - Strong penalty may not be correct approach.
- Do we just need more training time?
  - More training time will allow network to learn better strategies.
  - Slightly modified penalty as well
  - Increased gamma more so planning ahead is even better
  - Try both less random initialization and way more.

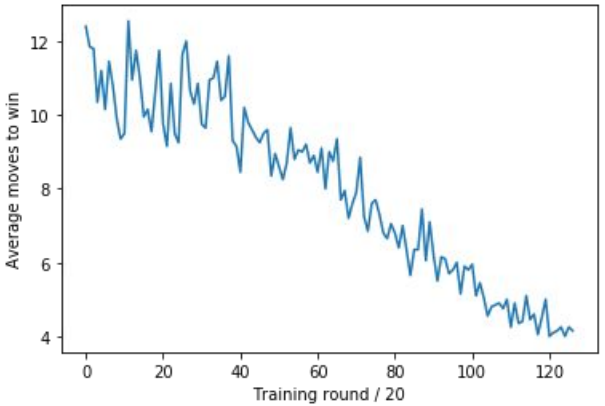


Higher reward for later wins

No strong penalty for winning early, not much random initialization:



Shorter train time, strong penalty, way more random initialization:



Vanishing gradient issue.

Conclusions

- DeepQ more suitable for larger games — but takes a long time for them.
  - Training time long, but more consistent.
  - Learns general good features, not strategies that will win all the time as necessary for tiny games.
- For any board size, can find a way to “cheat” the system.
  - Long training times necessary to evade. Penalizing bad play doesn’t really work.
- Small board sizes can train relatively quickly, large board sizes take a while.
  - Not exponential increase, but in range of dozens of hours.
- Reward scheme is extremely important.
- Randomness is a huge issue.
  - Performance and training time can vary wildly between different seeds.
- Connect Four needs way more training time than I’m able to provide.