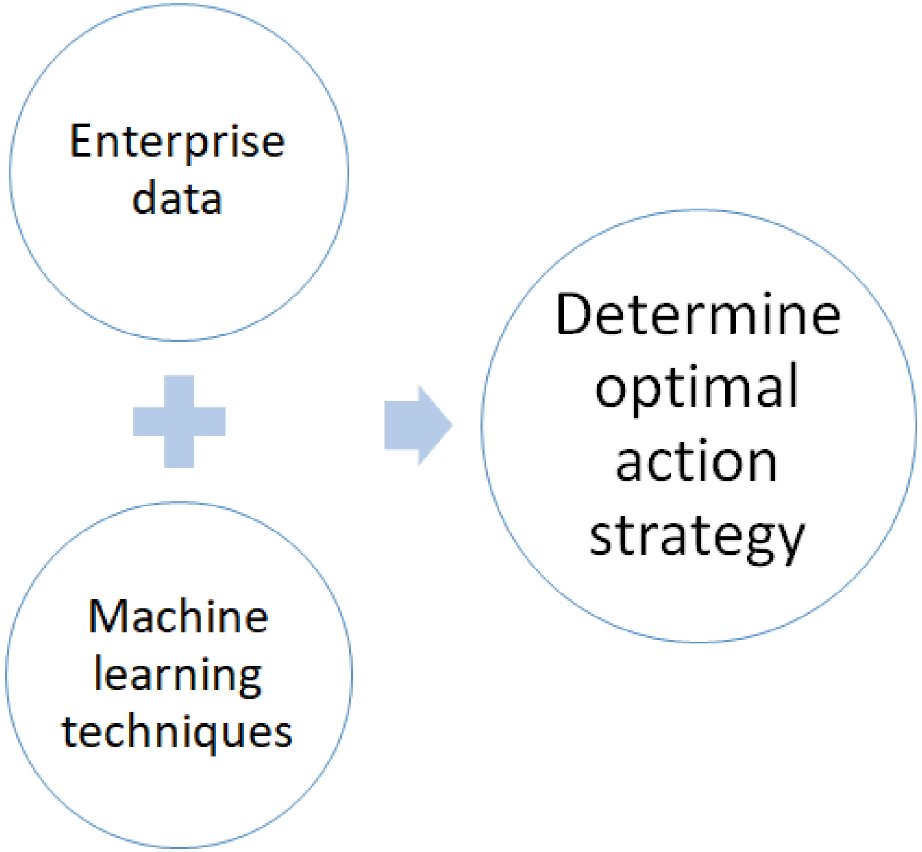
Project 10

Cerebri

Gaël Letarte St-Pierre
Hongshi Li
Tianyi Zhang
Yuelan Qin
Xing Hu

Introduction



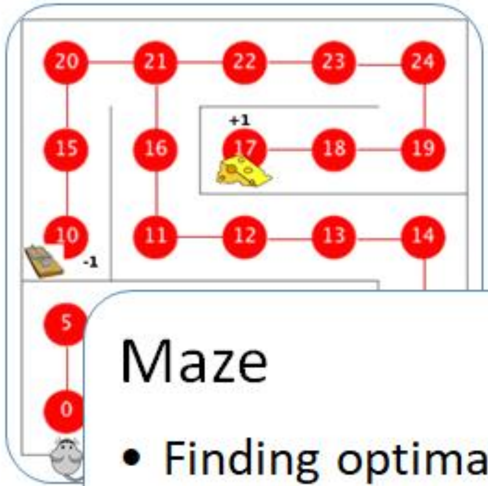
Challenges:

- No perfect information
- Infer rules/rewarding patterns from static data
- Limited available data

Introduction

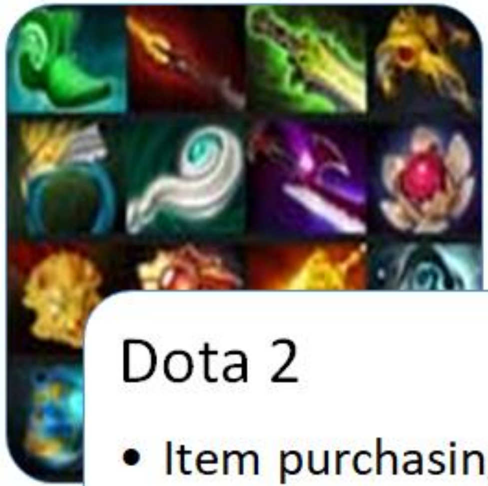
Next event prediction

- Naïve Bayes
- Reinforcement learning



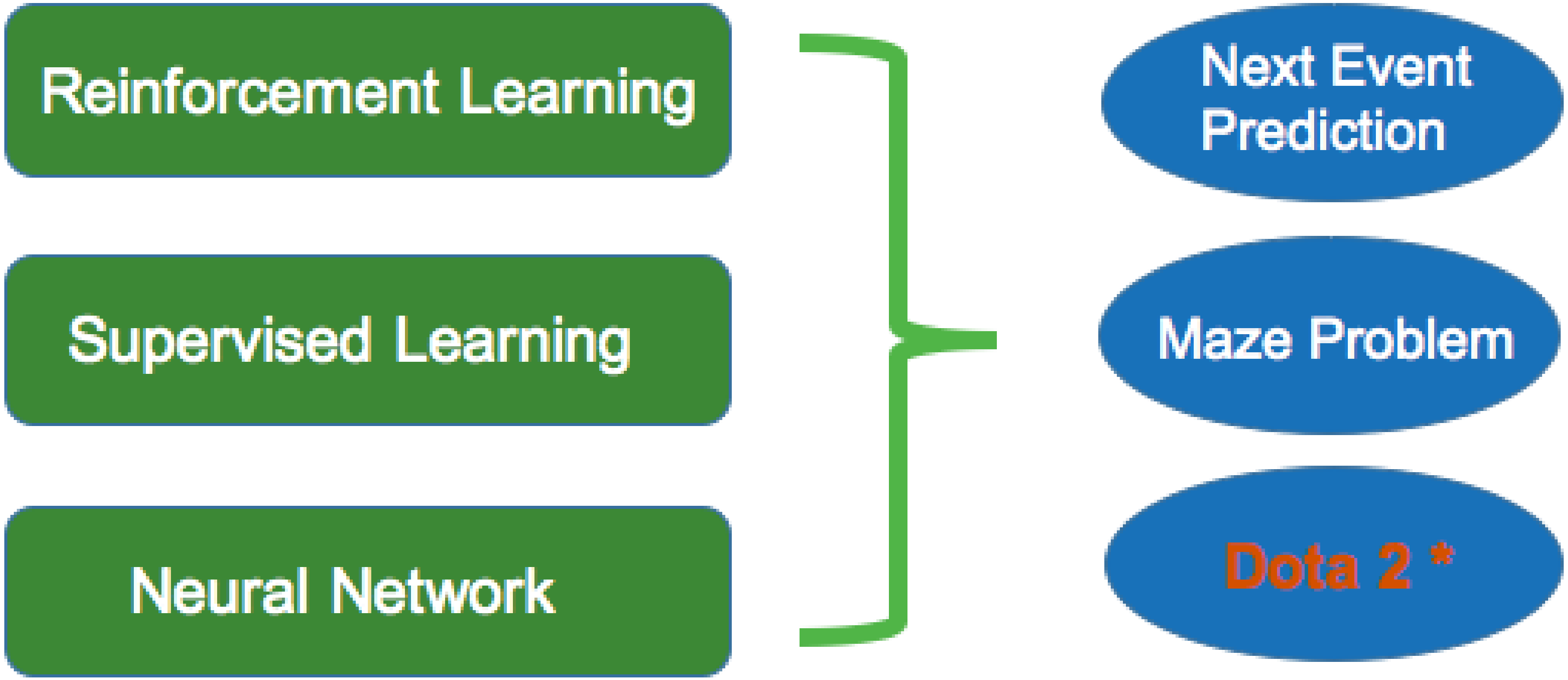
Maze

- Finding optimal path
- Reinforcement learning



Dota 2

- Item purchasing strategy
- Various techniques


Reinforcement Learning
Supervised Learning
Neural Network
Next Event Prediction
Maze Problem
Dota 2 *

Dota 2

Introduction – Dota 2

Data available

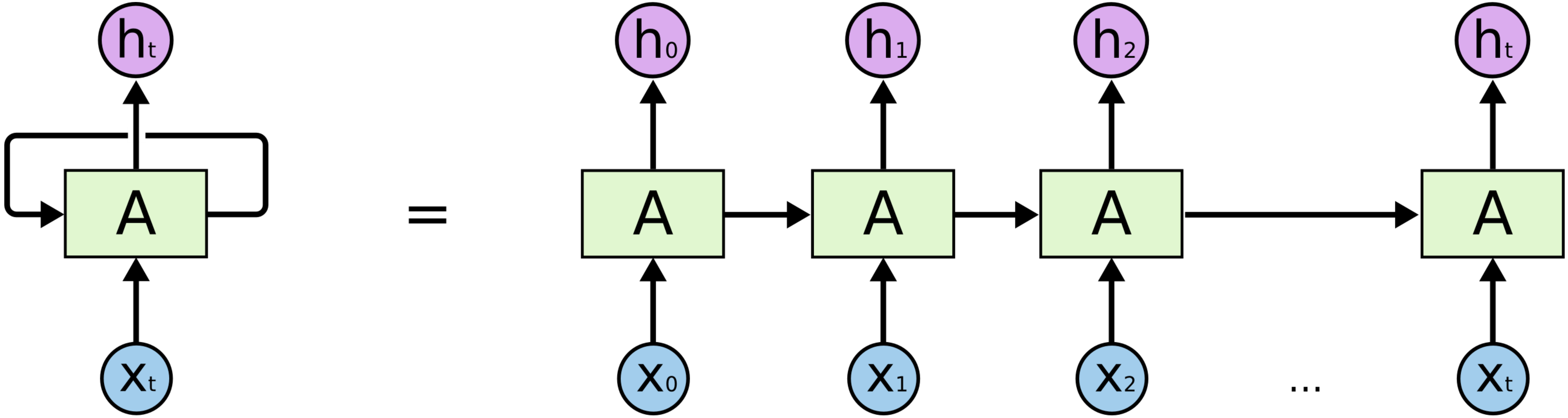
- 50,000 matches
- Item purchasing records of both combating teams
- Final results (winning/losing)

Task

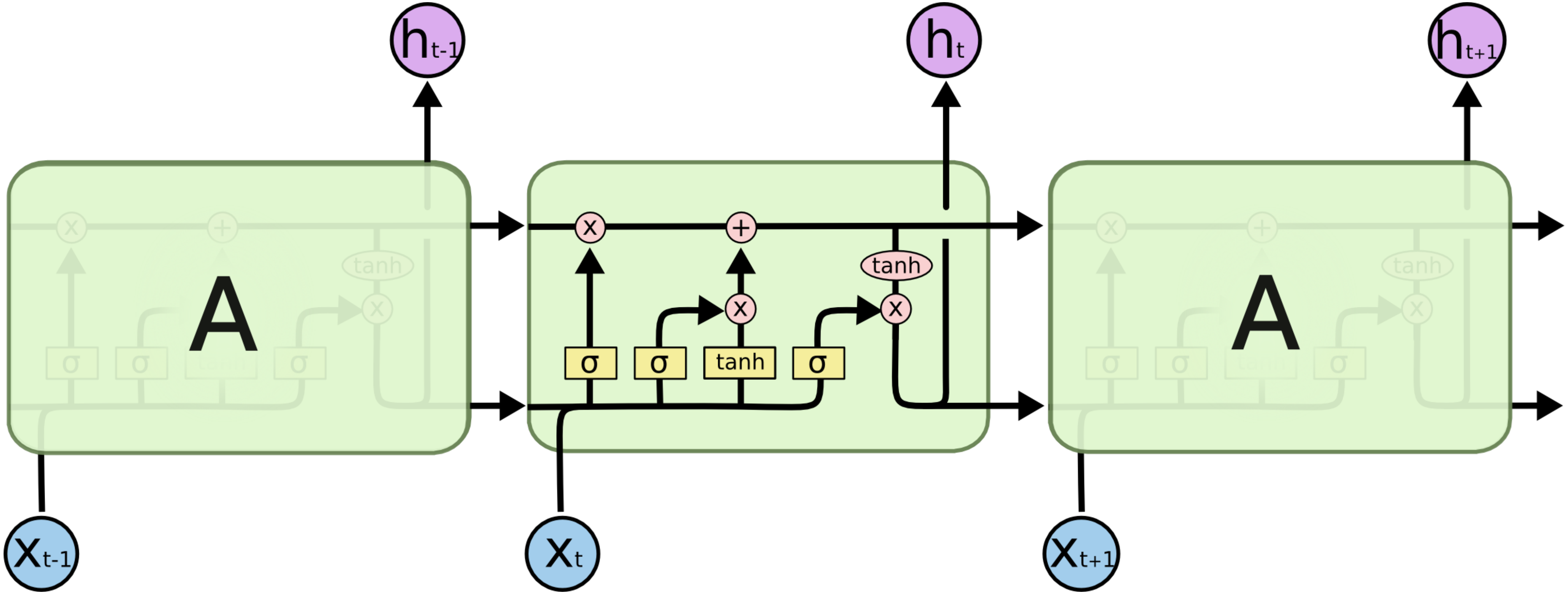
- Find relationship
  - Item purchasing ←→ Outcome
- Find optimal purchasing strategies

{"0": {"a": [[239, 46], [248, -46], [249, -46], [250, -38], [254,
[286, 41], [298, -36], [326, 29], [335, 42], [335, 42], [337, 9
-46], [399, 25], [408, 36], [408, 34], [408, 20], [408, 46], [4
[412, -20], [417, -16], [417, -38], [418, -39], [420, -17], [42
-46], [460, -46], [462, 34], [463, 46], [464, -46], [465, 86],
[478, 36], [495, 63], [495, 17], [497, 34], [501, -2], [501, -2
-25], [540, -65], [541, -46], [543, -29], [551, -46], [555, -46
-46], [637, -42], [637, -42], [641, 46], [643, 46], [643, -14],
-88], [644, -12], [651, -16], [651, -20], [658, -8], [661, -16]
[699, 16], [706, -25], [706, -17], [714, 79], [724, -63], [732,
[772, 42], [777, 40], [781, 46], [783, 42], [784, -13], [787, -
46], [822, 17], [822, 25], [831, -20], [832, 102], [833, -38],
[864, -168], [927, 46], [939, 6], [944, 26], [944, 164], [951,
[980, -69], [990, -46], [1008, 54], [1022, -46], [1026, -46], [
23], [1058, 46], [1062, 46], [1068, -46], [1070, -27], [1070, -
[1103, -46], [1123, -19], [1129, -86], [1129, -4], [1135, 46],
37], [1147, -36], [1151, -46], [1152, -38], [1159, -5], [1173,
[1219, 60], [1222, -46], [1233, 42], [1233, 42], [1243, 46], [1
46], [1258, 43], [1258, 218], [1285, -18], [1314, -170], [1315,
[1339, 162], [1345, -46], [1346, -46], [1380, -46], [1381, 170]
60], [1397, -79], [1399, -46], [1401, -42], [1401, -43], [1401,
[1433, 46], [1447, 22], [1455, -145], [1456, -46], [1463, 154],
-218], [1495, -43], [1496, -218], [1496, -42], [1498, 46], [150
[1599, -42], [1599, -42], [1605, -9], [1613, -21], [1621, 46],
24], [1668, -46], [1671, -46], [1671, 60], [1675, 21], [1675, 1

Recurrent Neural Networks



Long Short-Term Memory



Winner Prediction

Winner Prediction Results

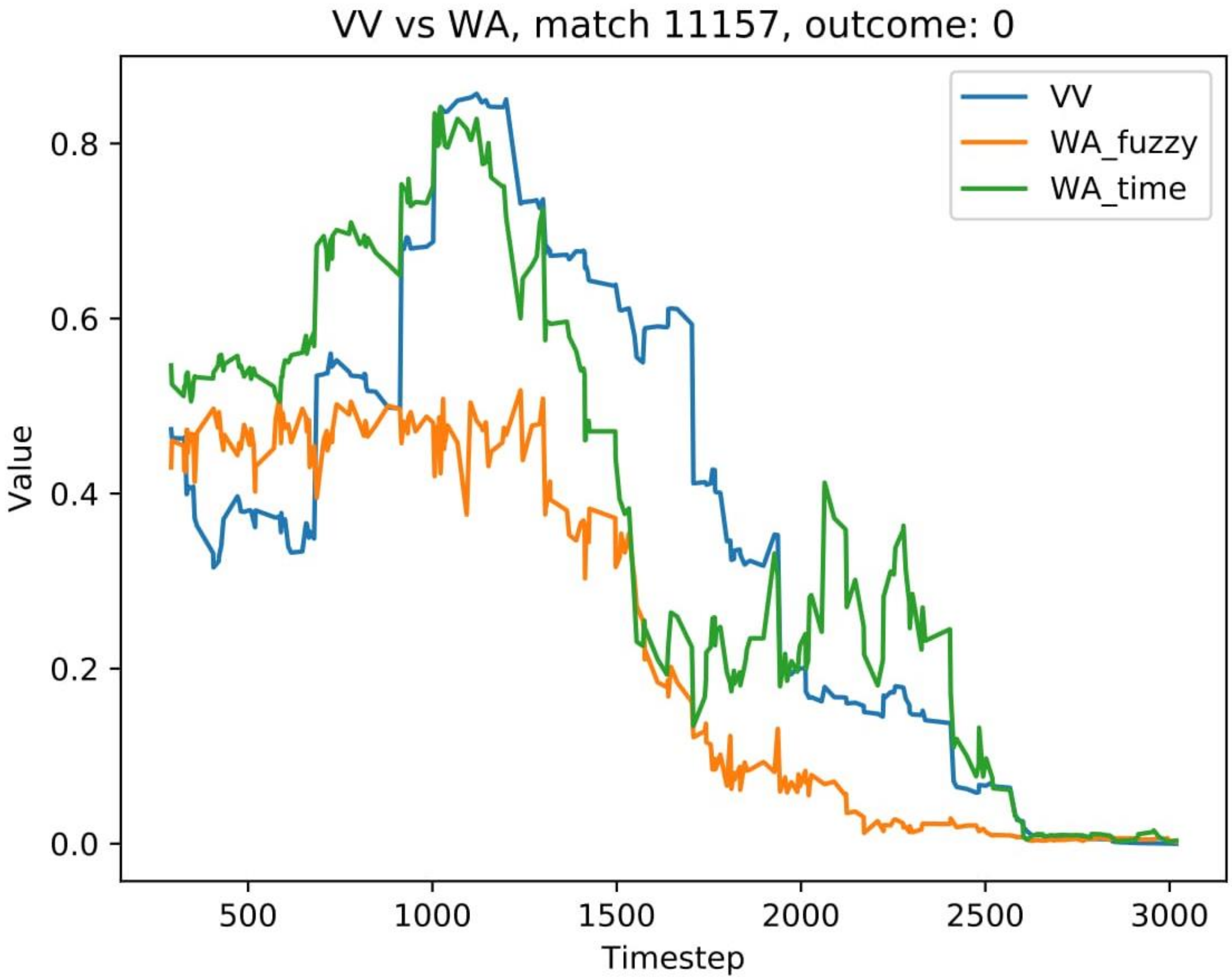
Winning Advantage

Models

Winning Advantage (WA)

- Time Padded Architecture (WA_time)
- Fuzzy Sequences Architecture (WA_fuzzy)

Vendor Value (VV)


VV vs WA, match 11157, outcome: 0
VV
WA_fuzzy
WA_time
0.0
0.2
0.4
0.6
0.8
Value
500
1000
1500
2000
2500
3000
Timestep

Results

Rank	VV (%)	WA_fuzzy (%)	WA_time (%)
Best	**46**	26	28
Middle	32	13	**55**
Worse	22	**61**	17

Reinforcement Learning
Simple Q-learning

Q-learning is a model-free reinforcement learning technique. It can be used to find an optimal action policy for any given (finite) Markov decision process (MDP).

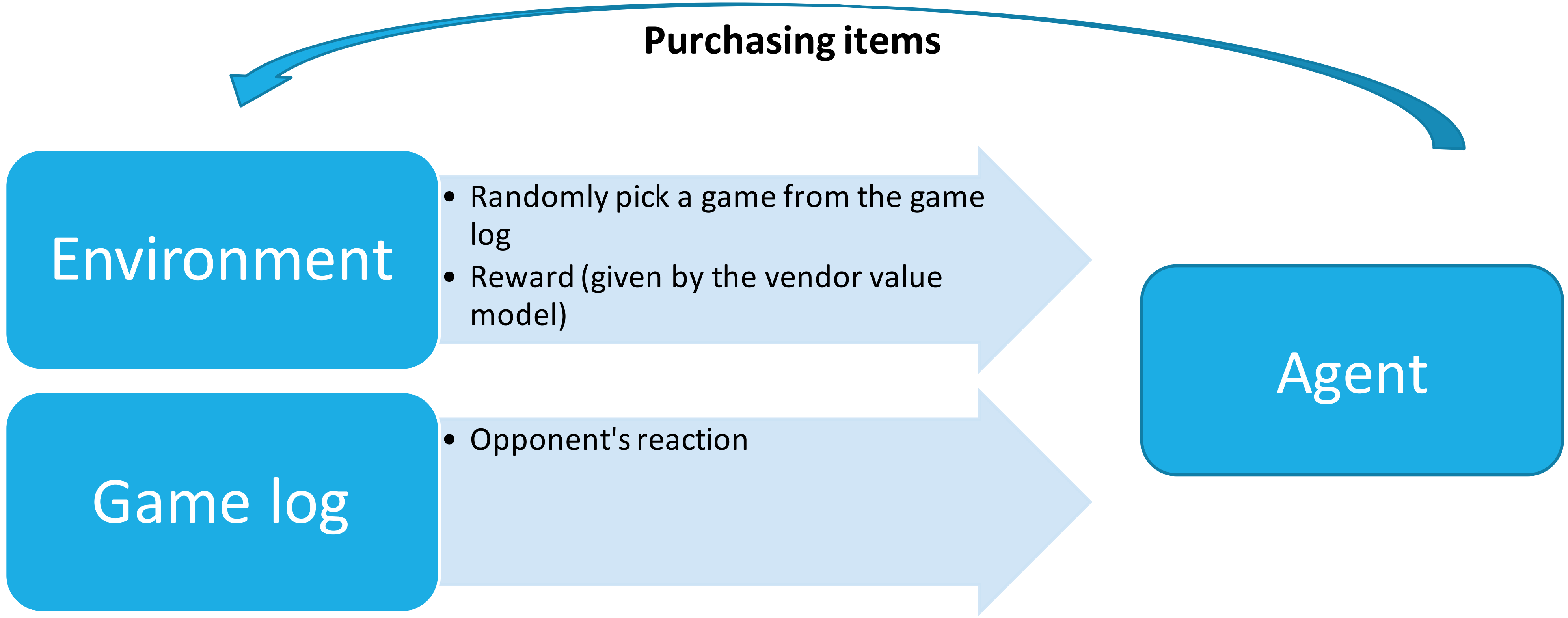
$$R = \begin{array}{c c} & \textbf{Action} \\ \textbf{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Two Models

Dictionary Model

Tree Model

Environment



Dictionary Model

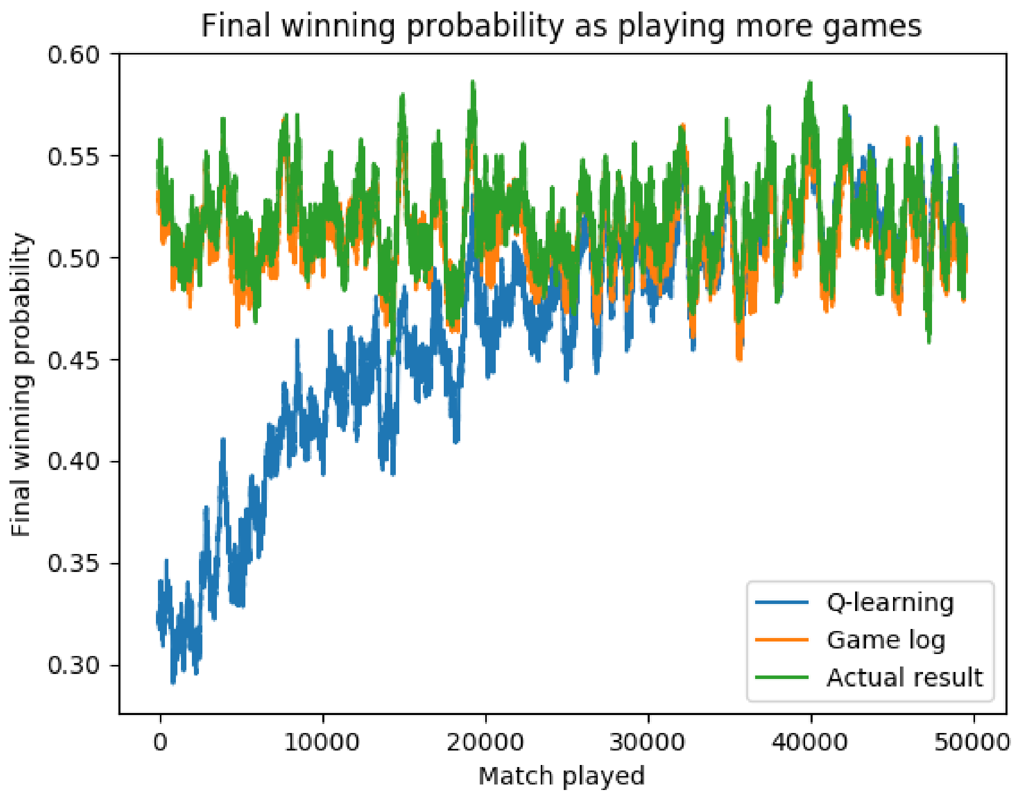
Method

Use a dictionary to store states, actions and corresponding q values, always pick the action with the highest q value.

Limit

For each state, we only allow actions that are in the game log. We can never beat the best game log.

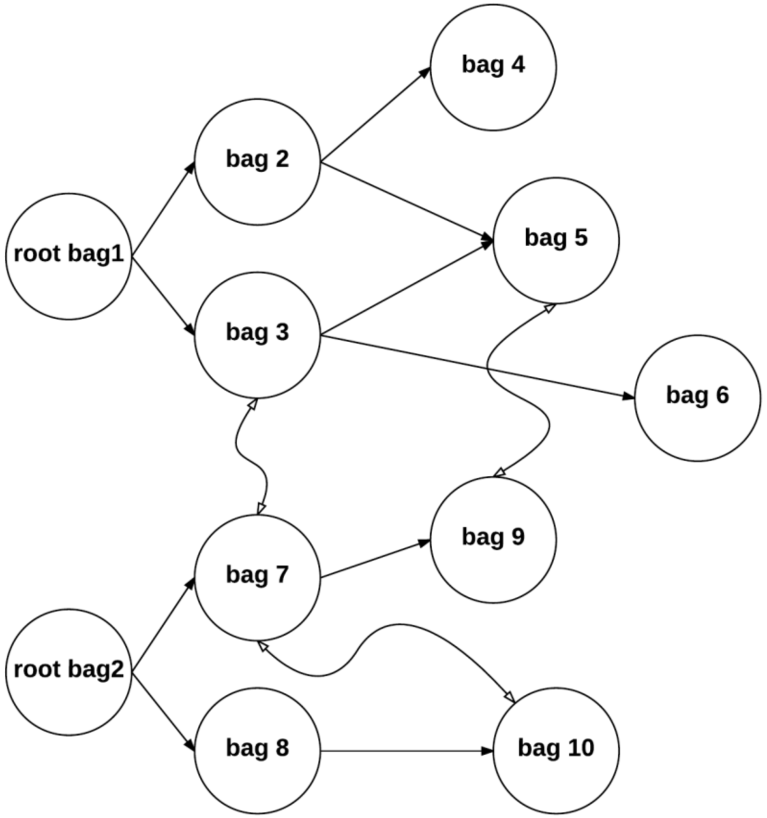
Result of Dictionary Model



As we expected, our model converged to actual results.

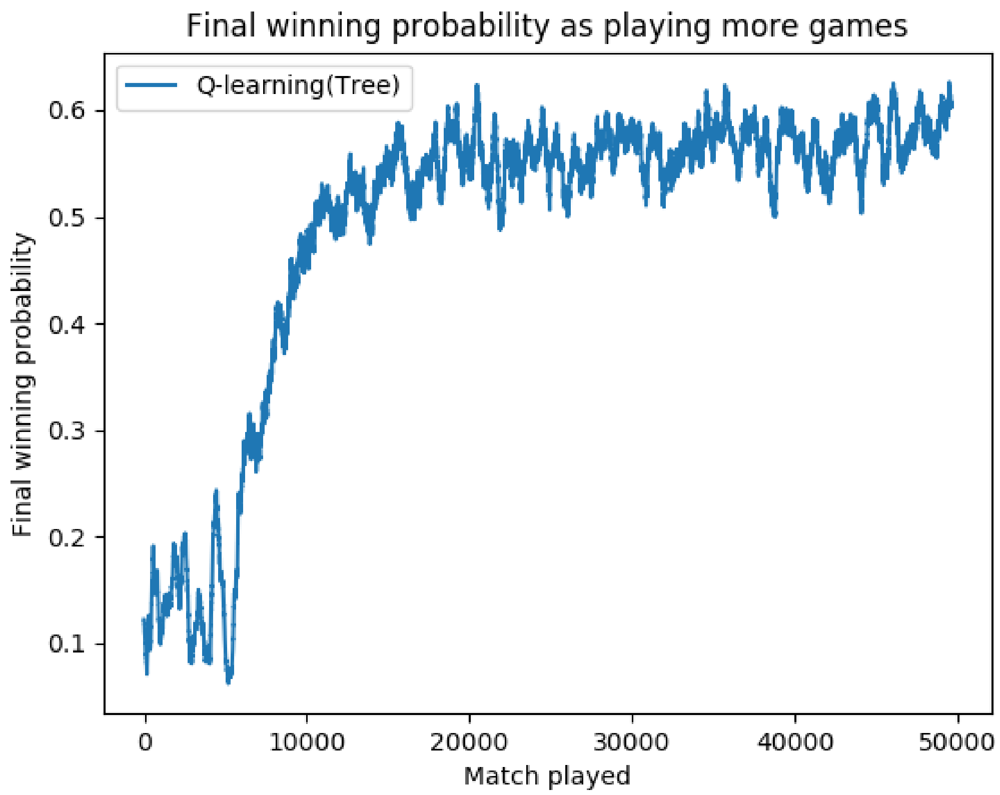
The winning probability is slightly above 50% after we train the model for 50000 games.

Tree Model



- The tree is made up of nodes.
- A node is used to capture all the features of a unique state.
- In the tree model, we have similar nodes.
- We allow jumping among similar nodes.

Result of Tree Model



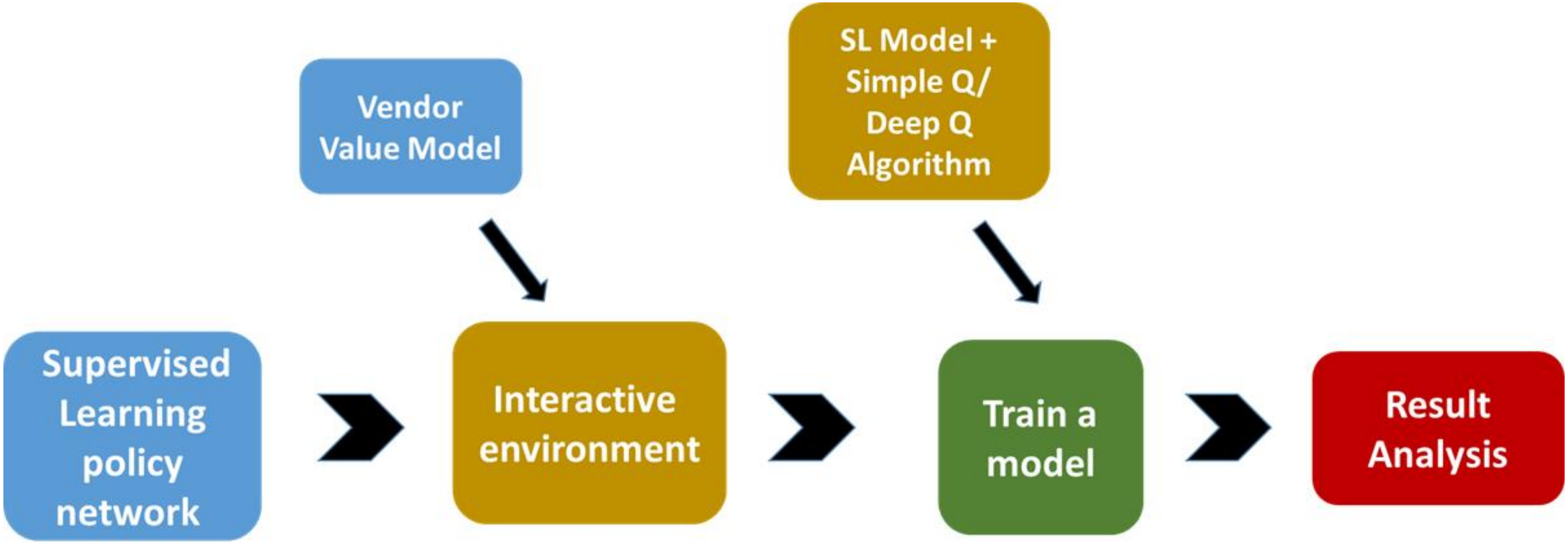
The winning probability is about 60% after

50000 games.

Outperform the dictionary model.

Alpha Go Approach

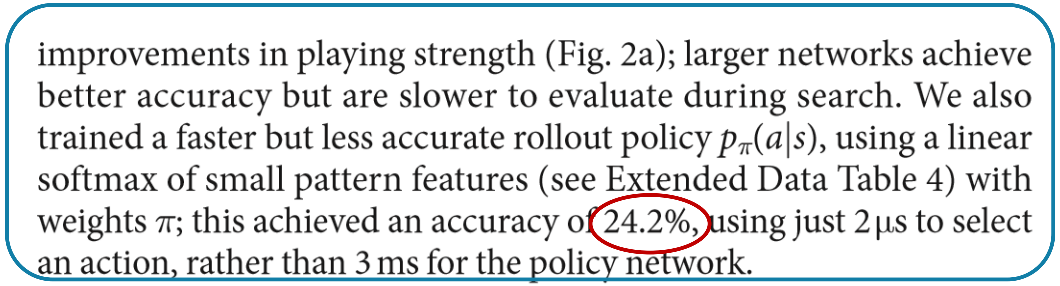
Step		Aim
1. Supervised Learning Policy Network	→	Directly learn from human experts
2. Reinforcement Learning Policy Network	→	Calculate Action Probability
3. Value Network	→	Estimate State Situation

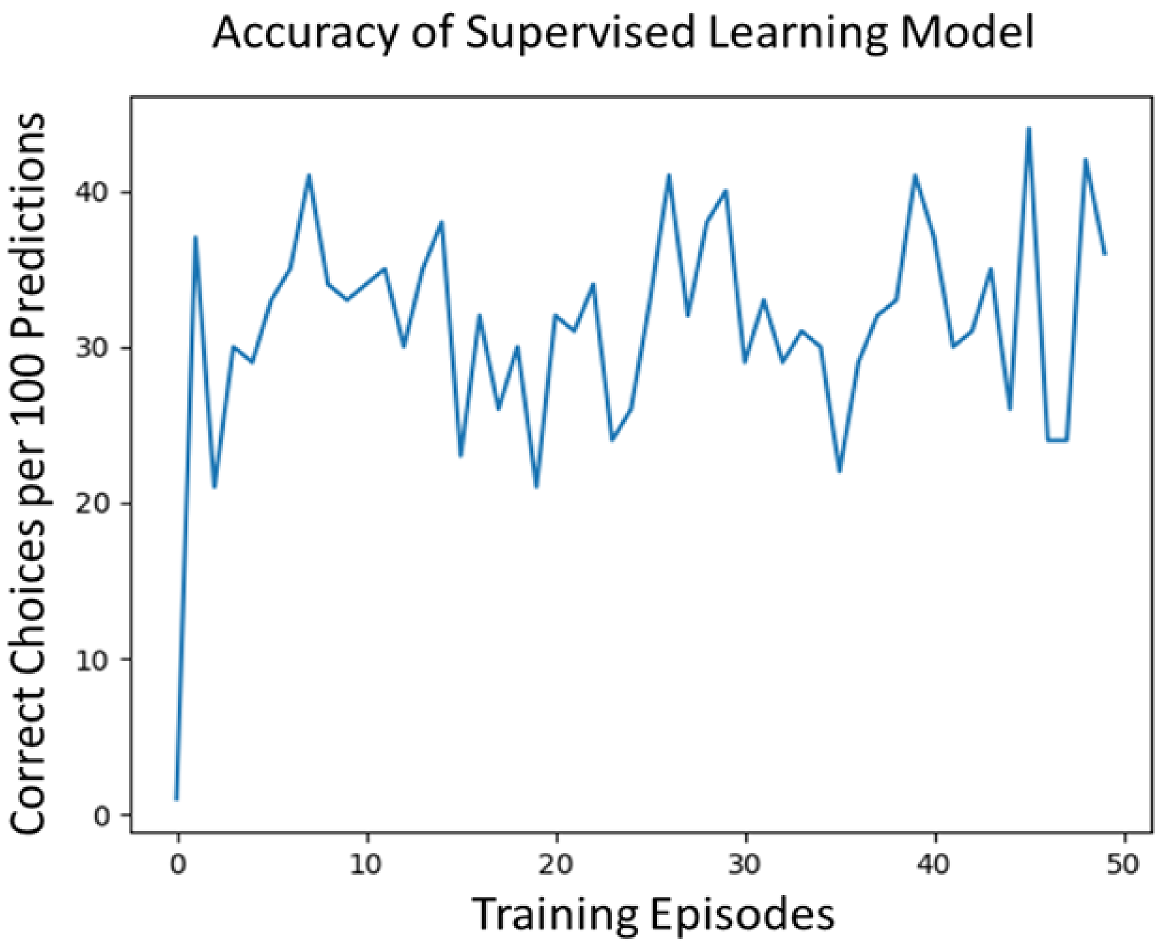


From Alpha Go to Dota2

1. Supervised Learning Model

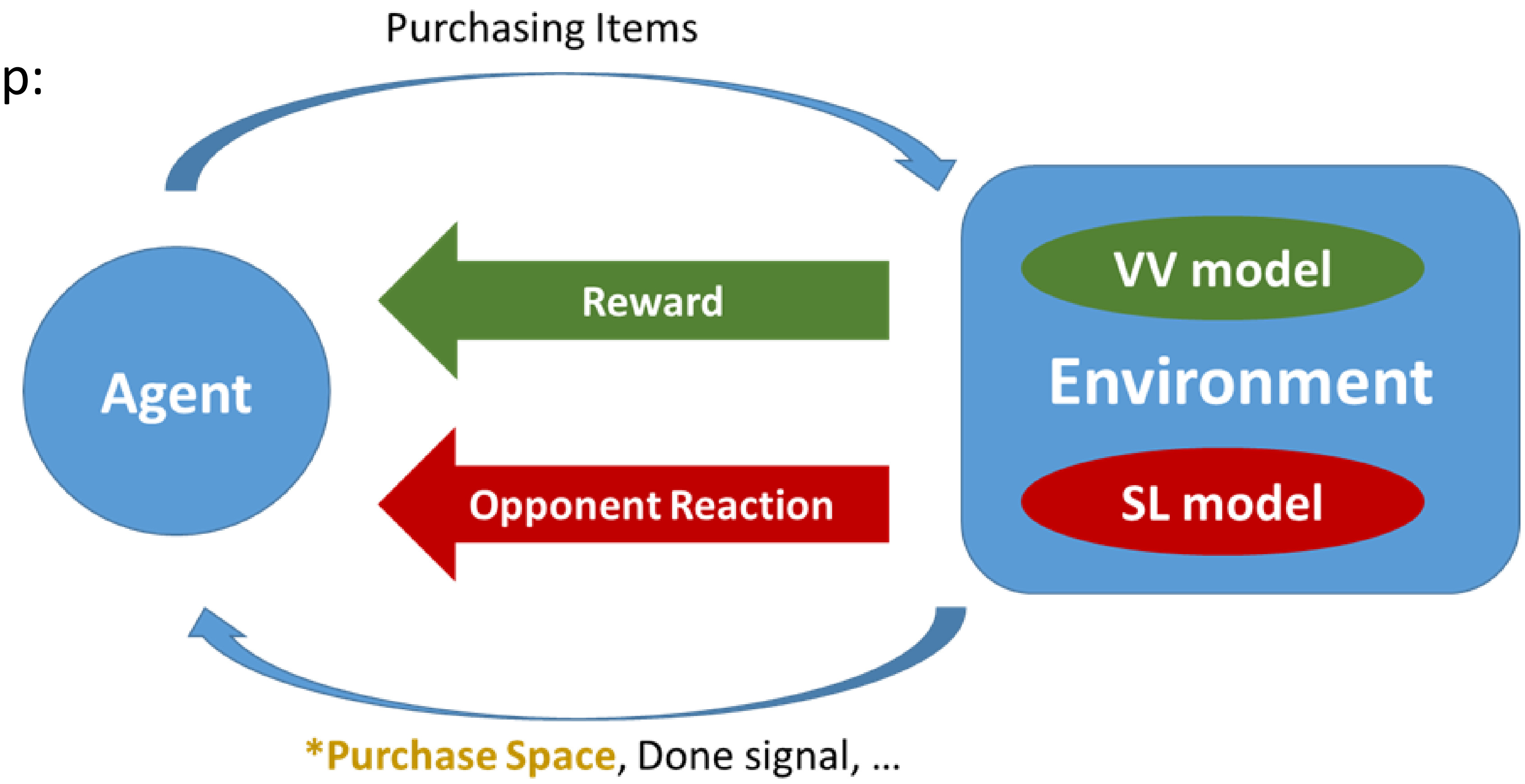
The final accuracy of the supervised learning model reaches 30%


improvements in playing strength (Fig. 2a); larger networks achieve better accuracy but are slower to evaluate during search. We also trained a faster but less accurate rollout policy $p_\pi(a|s)$, using a linear softmax of small pattern features (see Extended Data Table 4) with weights π; this achieved an accuracy of 24.2%, using just 2 μs to select an action, rather than 3 ms for the policy network.



2. Interactive Environment

At each step:



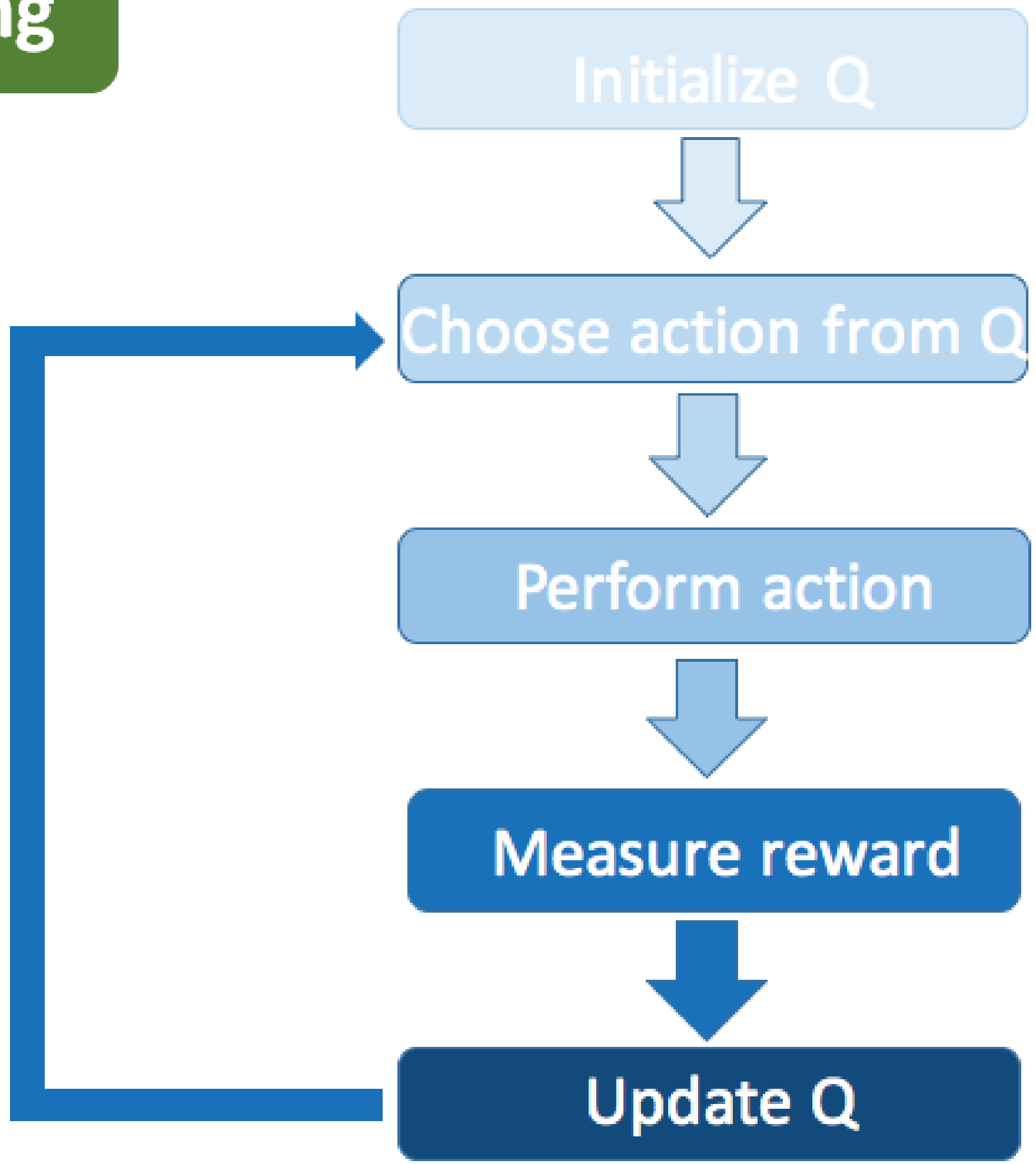
2. Interactive Environment

*Purchase Space Restriction

- By time
- By item bag

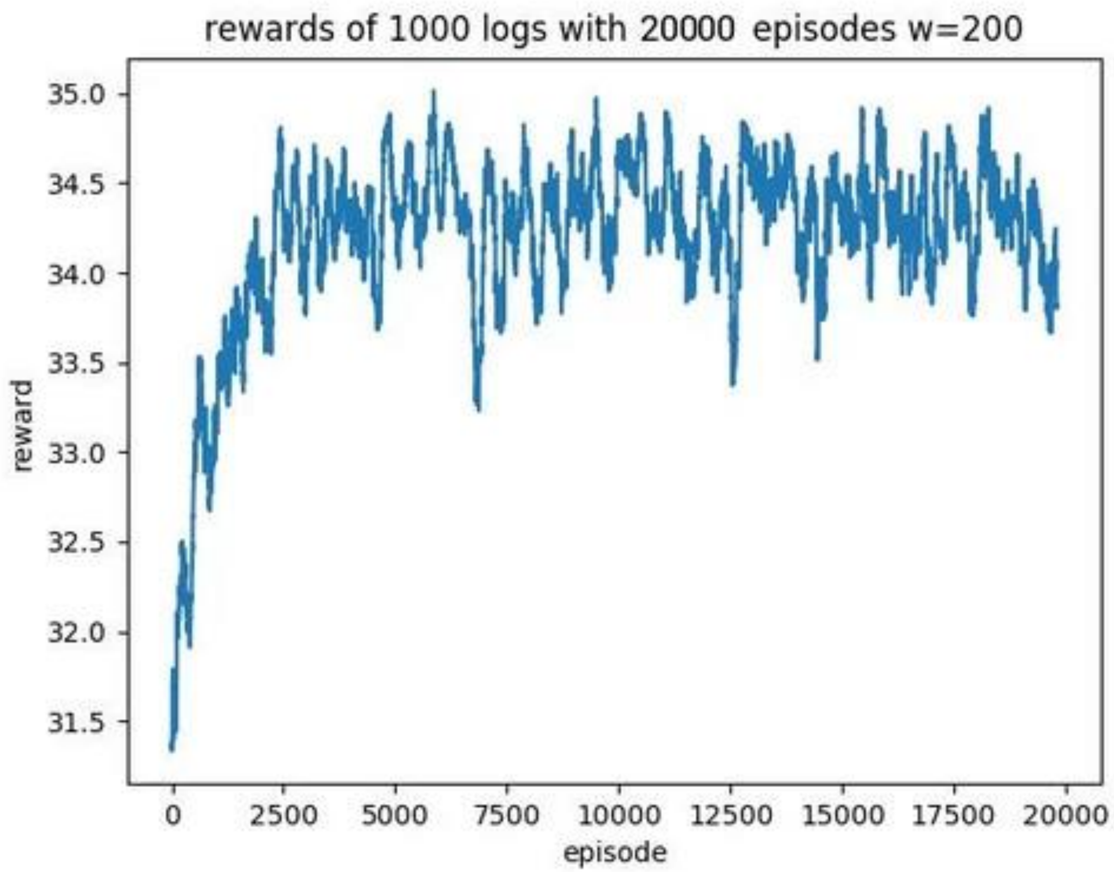
3. Reinforcement Learning

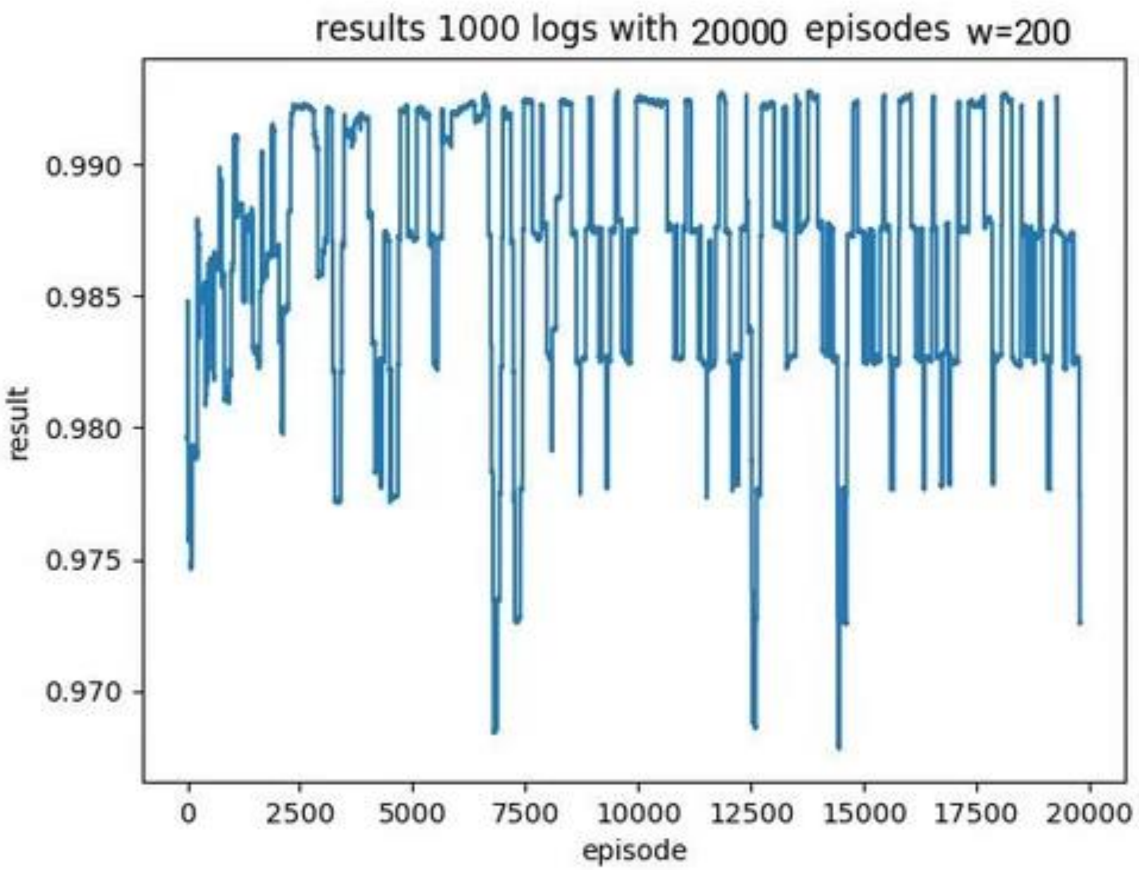
Simple Q-Learning



3. Reinforcement Learning

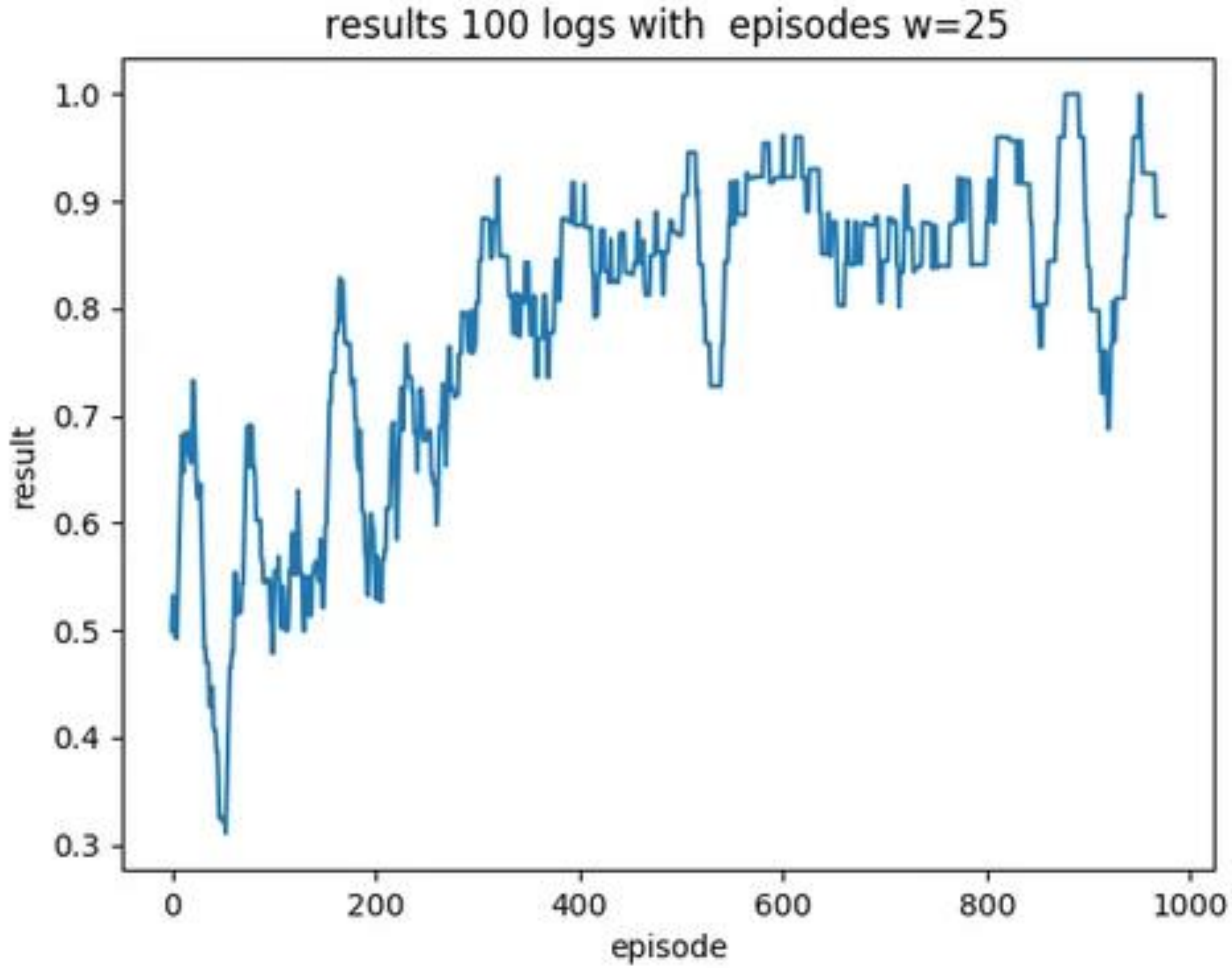
By time



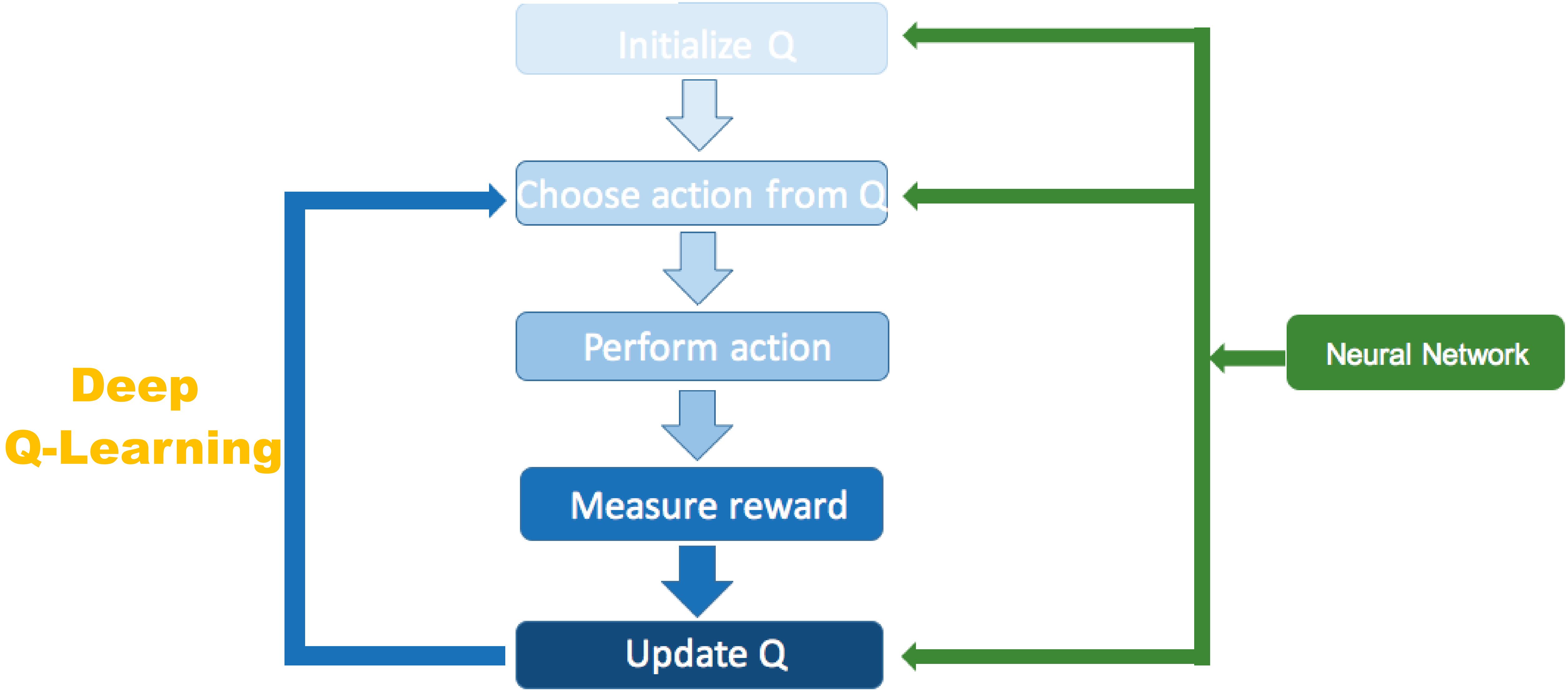


3. Reinforcement Learning

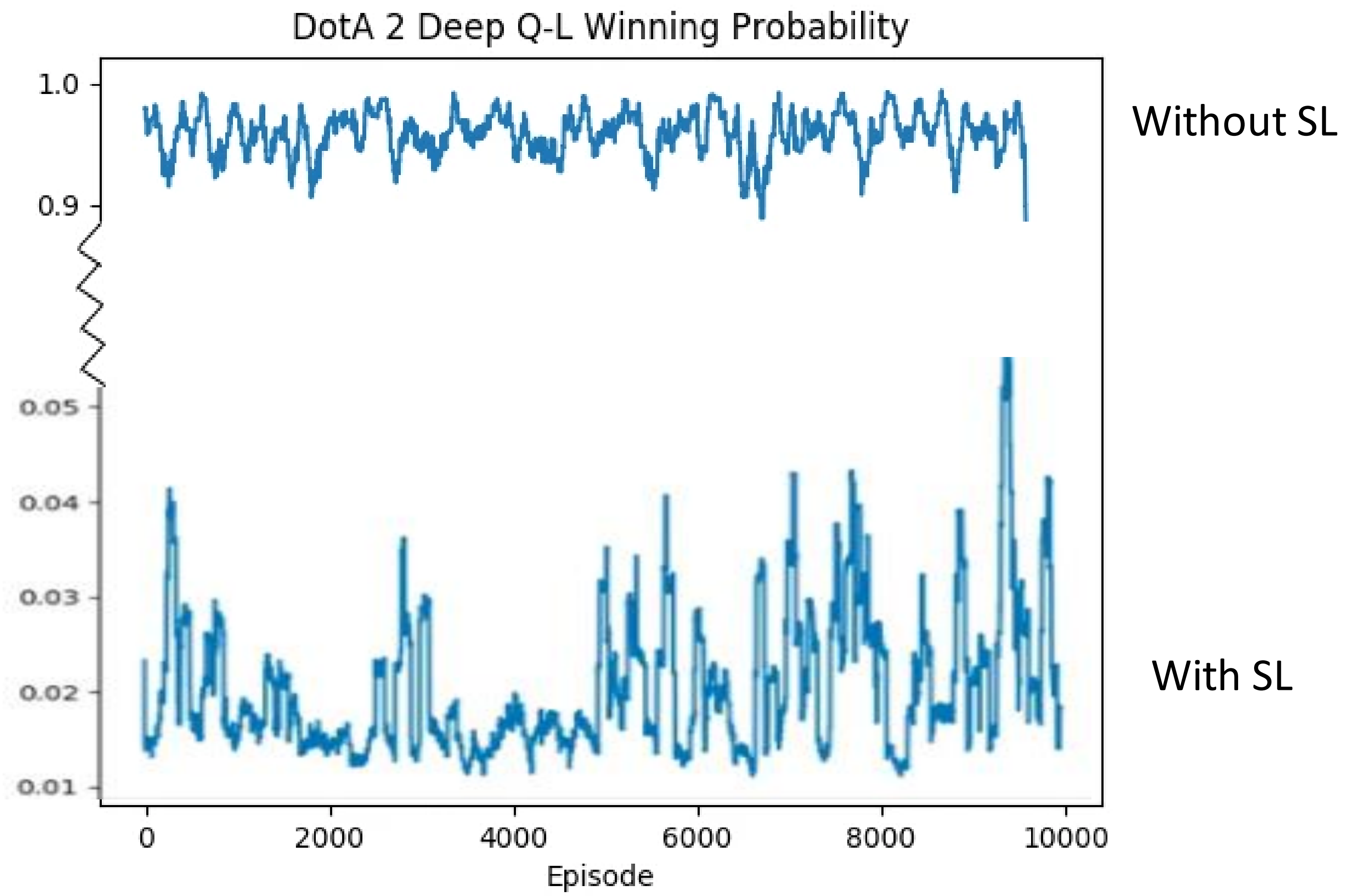
By item bag



3. Reinforcement Learning



3. Reinforcement Learning



4. Summary

Reinforcement Learning

Supervised Learning

Neural Network

Next Event Prediction

Maze Problem

Dota 2 *