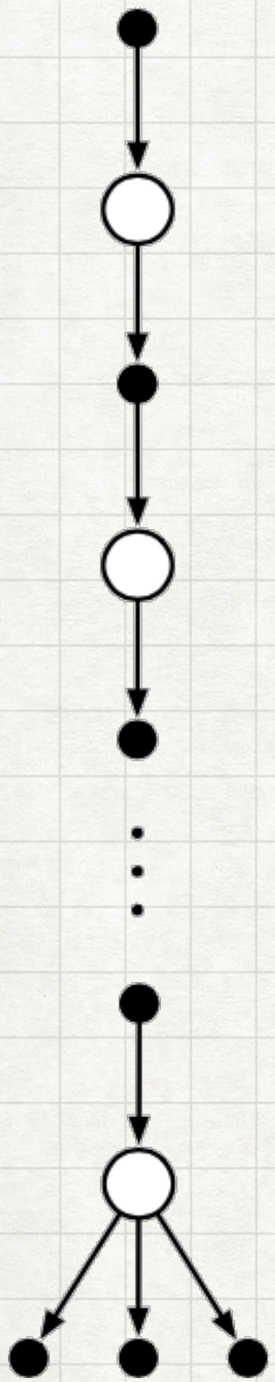
COMPARISON OF N-STEP ALGORITHMS

Jonathan Campbell
COMP-767
February 17, 2017

OVERVIEW

- Comparison of off-policy n-step algorithms to estimate Q
  - Off-policy n-step Expected Sarsa
  - n-step Tree Backup
  - Off-policy n-step Q(σ)
- Values for α and σ will be compared.

N-STEP EXPECTED SARSA



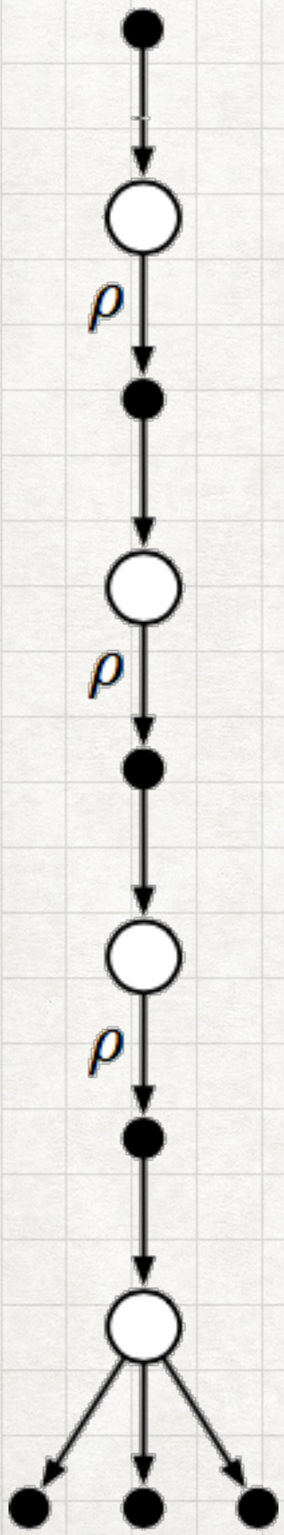
Use n-step return in update rule:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[G_t^{(n)} - Q_{t+n-1}(S_t, A_t)\right]$$

Return is sum of rewards plus expectation at last timestep:

$$G_t^{(n)} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \sum_a \pi(a|S_{t+n}) Q_{t+n-1}(S_{t+n}, a)$$

OFF-POLICY N-STEP EXPECTED SARSA



Weight updates by IS ratio **ρ**:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1}^{t+n-1} \left[G_t^{(n)} - Q_{t+n-1}(S_t, A_t) \right]$$

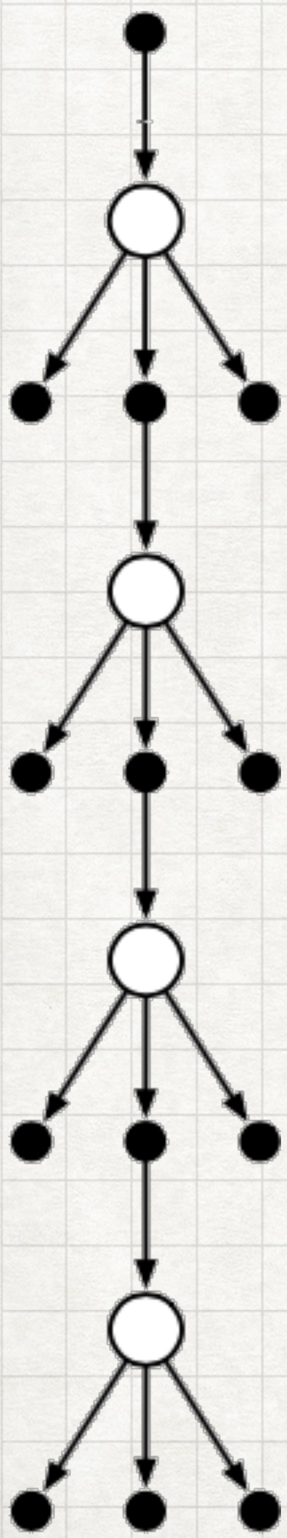
ρ is relative prob. under target/behavior policy of taking the sequence of actions:

$$\rho_t^{t+n} \doteq \prod_{k=t}^{\min(t+n-1, T-1)} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}.$$

Trajectories could be discarded if one action has zero prob. under target policy.

Also: high IS ratios could cause high variance in Q-values, requiring lower learning rate.

N-STEP TREE BACKUP



Off-policy learning without importance sampling.

N-step return considers all possible actions from each state (action q-val multiplied by action prob. under π).

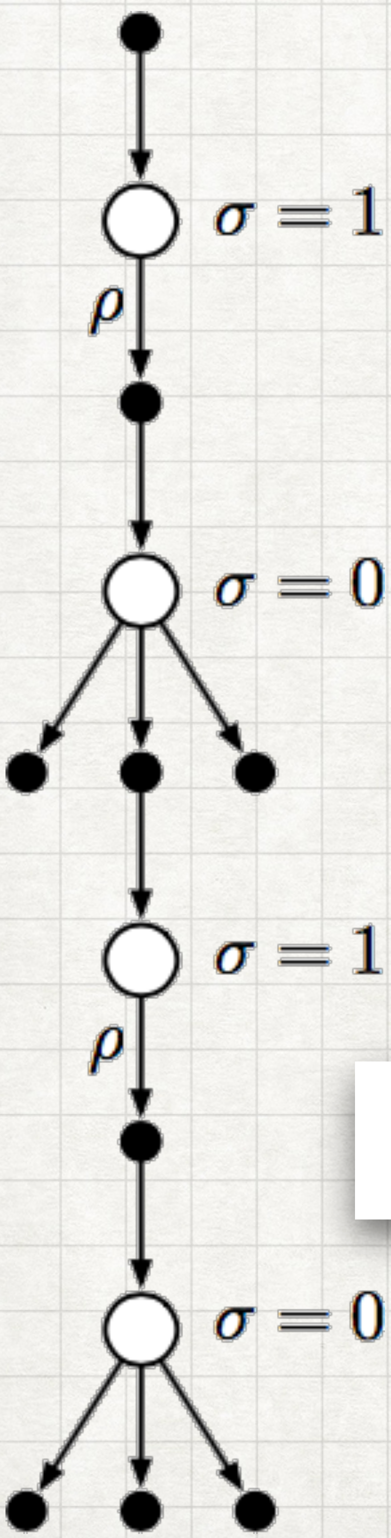
The action taken in a particular state takes into account the reward observed and actions from the next state.

2-step case:

$$\begin{aligned} G_t^{(2)} = &\, R_{t+1} \\ &+ \gamma V_{t+1} \\ &+ \gamma \pi\left(A_{t+1} \middle| S_{t+1}\right)\left[R_{t+2} + \gamma V_{t+2} - Q_t\left(S_{t+1}, A_{t+1}\right)\right] \end{aligned}$$

where $V_t = \sum_a \pi\left(a \middle| S_t\right) Q_{t-1}\left(S_t, a\right)$

N-STEP Q(SIGMA)



Generalization of expected Sarsa and tree-backup algorithm.

Mix Sarsa update and tree backup update, depending on **σ**.

$$G_t^{(n)} \doteq Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^{k} \gamma\big[(1-\sigma_i)\pi(A_i|S_i) + \sigma_i\big]$$

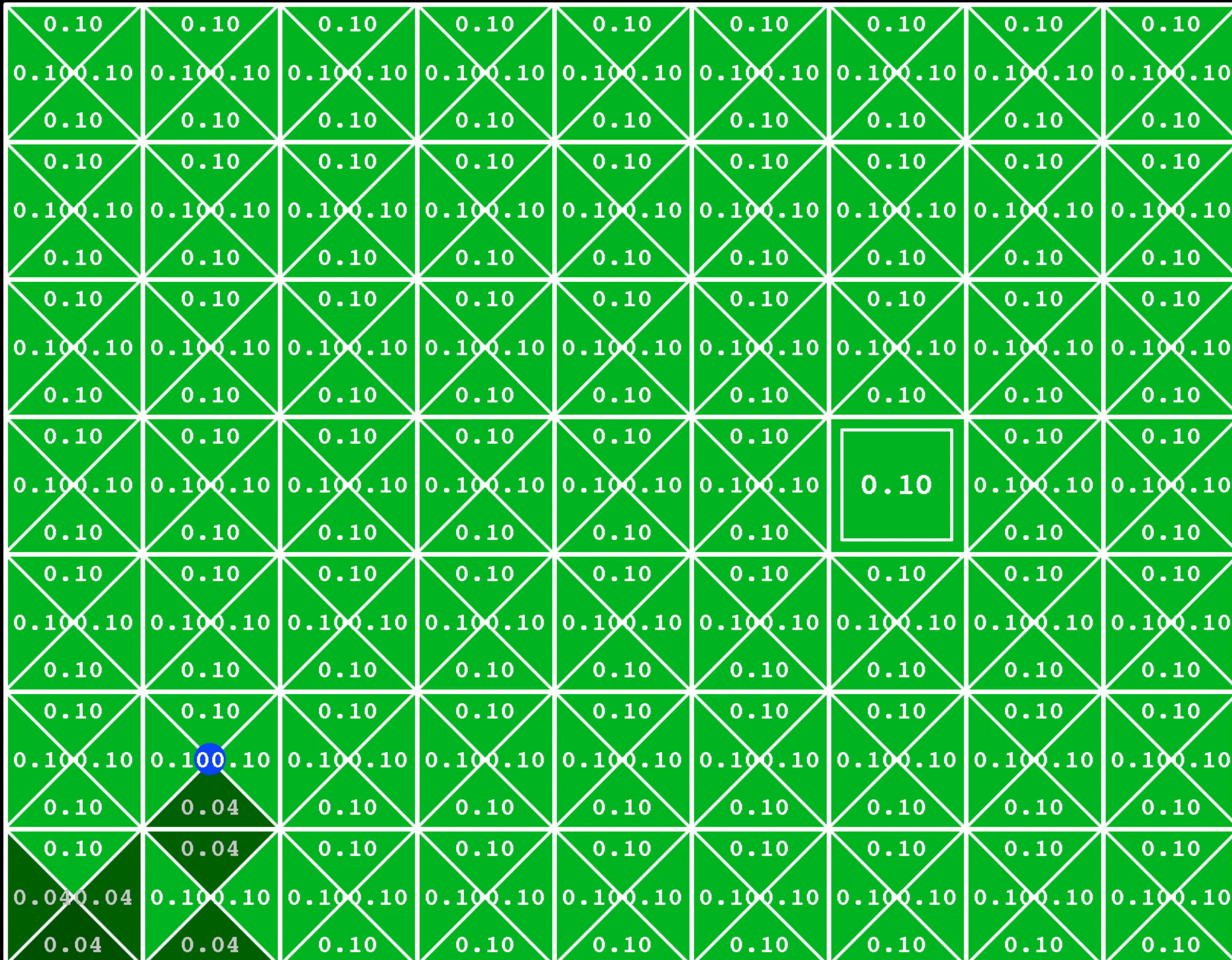
$$\delta_t \doteq R_{t+1} + \gamma\big[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1-\sigma_{t+1})V_{t+1}\big] - Q_{t-1}(S_t, A_t).$$

(Also have to modify IS ratio for off-policy case.)

IMPLEMENTATION

- Uses Gridworld RL framework from UC Berkeley AI course
  - http://ai.berkeley.edu/reinforcement.html
- Extension to windy gridworld:

```
def getWindyGrid():
    grid = [[' ',' ',' ','^','^','^','^',' ',' '],
            [' ',' ',' ','^','^','^','^',' ',' '],
            [' ',' ',' ','^','^','^','^',' ',' '],
            [' ',' ',' ','^','^','^', 1 ,' ',' '],
            [' ',' ',' ','^','^','^','^',' ',' '],
            [' ',' ',' ','^','^','^','^',' ',' '],
            ['S',' ',' ','^','^','^','^',' ',' ']]
    return Gridworld(grid)
```



CURRENT Q-VALUES

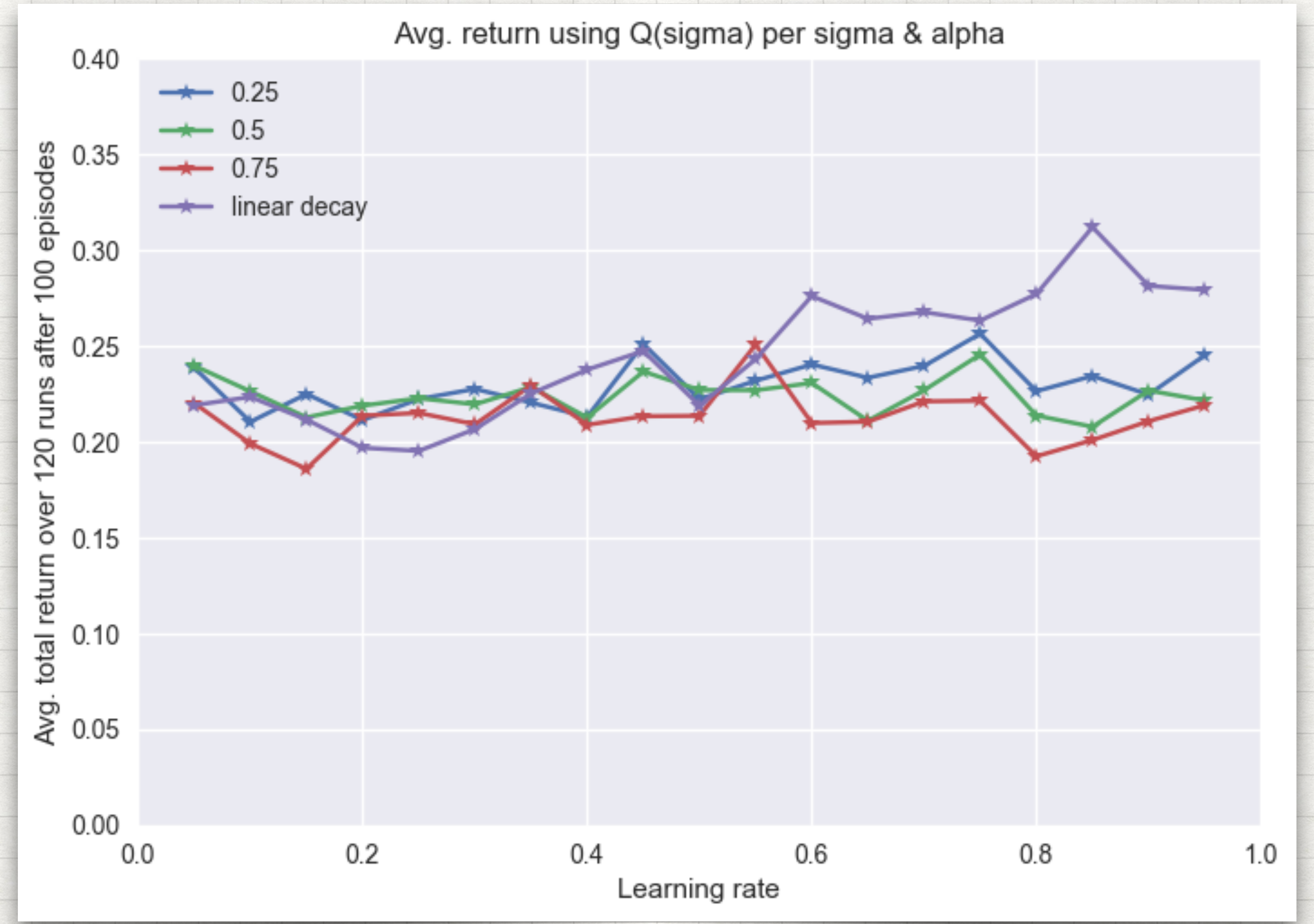
SIGMA PARAMETER

- Sigma parameter controls degree of sampling.
  - **σ** = 0: Tree-backup algorithm
  - **σ** = 1: Sarsa
- Values
  - Fixed
  - Linear/exponential decay
  - Function of current state and/or action

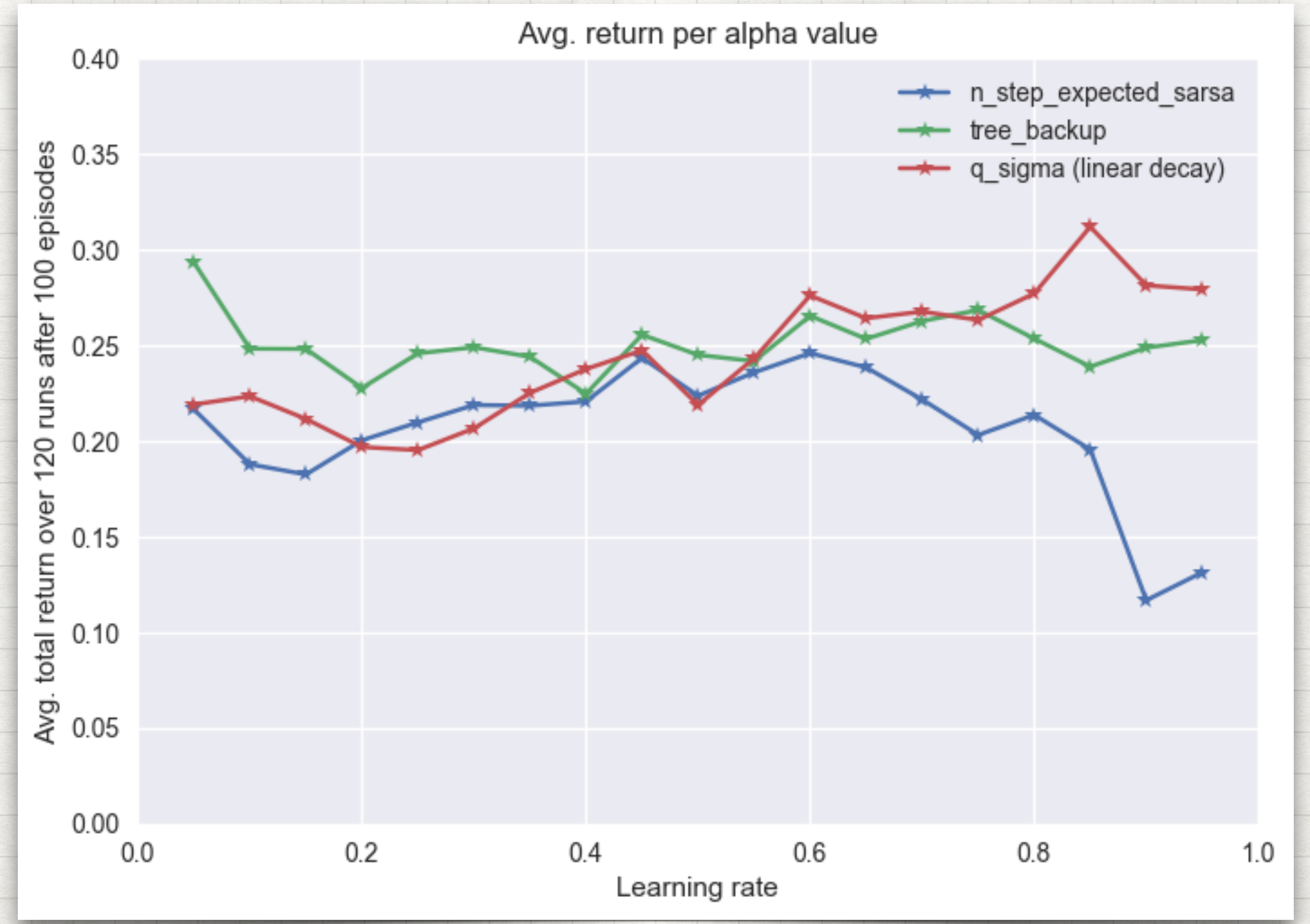
METHOD

- For each alpha/sigma value:
  - For 100 iterations:
    - Run n-step agent (n=10) on windy gridworld for 100 episodes and record total return of last episode.
  - Average the 100 total returns.
- For all algs., behaviour policy is ε-greedy.

SIGMA COMPARISON



N-STEP ALGORITHM COMPARISON



CODE

- Available at course GitHub repo:

  https://github.com/rllabmcgill/rlcourse-february-17-campbelljc