**Reinforcement Learning: Presentation (Key ideas)**

*Title Page:*

* Topic: TD Learning combined with echolocation
* Path navigation

*Outline:*

* Introduce background information
* Motivation + significance of methodology used and results
* Results obtained
* Future ideas and significance of results

*Temporal Difference Learning:*

* Classic algorithm at the root of RL (1991, Sutton)
* Idea: each time step -> take action, predict ensuing reward
* Goal: TD target as close to actual value as possible
* Use TD error to update value function
* Prediction gets closer to outcome as training continues

*Actor-Critic Architecture:*

* Extension of TD Learning
* Policy and Value function represented independently
* Actor: encodes policy
  + Takes actions at each time step
* Critic: encodes value function
  + Criticizes actions taken by actor
  + Using TD error most of the time
* Same principle as TD Learning:
  + Actor actions become less criticisable as training occurs

*Temporal-Difference Learning in Neuroscience:*

* Dopamine: hormone at the heart of learning
  + Encodes reward and punishment in the brain
  + Drives most of our behaviour (basic such as hunger state, complex such as addiction)
* Figure:
  + Monkey conditioned by fruit juice stimuli
  + 1) no conditioning, neurons that produce dopamine fire a lot when reward occurs
  + 2) conditioning, neurons fire to anticipate stimuli, no firing when reward occurs
  + 3) conditioning, neurons that produce dopamine stop firing greatly when reward does not occur
* Hebbian learning: “neurons that fire together, wire together”
* Dopamine predicts next step, re-calibrates the brain if change occurs in environment

*The Actor-Critic in Neuroscience:*

* States become perceptions
* Dopamine encodes TD error
* Big milestone in computational neuroscience
* One of the key ways AI has inspired neuroscience (was the opposite only for a long time)

*Bat Echolocation:*

* Multiple types of cries in one direction
* Important: echolocation used for…
  + Object Detection
  + Object recognition
  + Object Recognition
* Used for spatial navigation at night!

*The Environment:*

* Start: basic water maze map
  + Water maze: basin with platform, used in neuroscience to study mammalian behaviour
  + Platform is translucent to mimic blind-folded navigation
* Us: extension to perception-based navigation
* Add-ons:
  + Obstacles
  + Sound waves generated by agent every time step (45 degrees)
  + Choice of 8 or 16 directions
* Goal: mimic how a bat would travel in an environment

*The Foster Model (Part 1):*

* Place cells: activated at each step
* Map to a location (center)
* Have a standard deviation
* The closer the agent, the bigger the activation

*The Foster Model (Part 2):*

* Critic: vector length N
* Value function: weighted sum of place cell activations
* TD prediction error used for weight update
* As training occurs, convergence towards value at position pt is equal to TD target with mean reward

*The Foster Model (Part 3):*

* Actor: matrix z of dimensions 8xN or 16xN
* Actor operations:
  + Action vector at each position
    - Preferences for next state
  + Action probabilities
    - Generate actions based on this distribution
  + Weight update using TD error and place cell activations
    - Hebbian learning! Only selected action’s weights are updated

*The Foster Model (Algorithm):*

* Standard:
  + Determine action using actor
  + Determine actual reward from action
  + Compute TD error
  + Update critic weights
  + Update actor weights

*The Foster Model (Add-on):*

* Two networks for X and Y coordinates
* Weight vectors of length N
* Output at each position determined by weighted sums of vectors and place cell activations
* Weights updated using differences between new positions and old positions
* Summation of past place cell activations with more emphasis on recent ones
  + Acts as a memory component
* MENTION MISTAKE

*The Foster Model (Add-on Algorithm)*

* Same as regular Actor-Critic, ONLY REALLY WORKS WHEN GOAL HAS BEEN REACHED (GOAL MEMORY)
* But…when coordinate system is chosen by actor
  + Use X(p) and Y(p) to determine movement direction towards goal coordinate (cosine similarity in my code)
  + Compute TD error (always)
  + Update coordinate system weights (always, even when not selected)
  + Update actor in following fashion is Coordinate System is chosen

*Incorporating Echolocation:*

* Idea: re-balance action probabilities according to sound waves reflected upon agent
* Only two alterations
  + Wall and obstacle -> 0
  + Platform -> not 0
* Normalize by sum of probabilities in order to make it an accurate distribution for sampling
* NOTE: Coordinate System dimension is left untouched
  + Role of CS: memory!

*Pure TD Learning vs Coordinate System Add-On:*

* Explain:
  + One day: 100 trials
  + Escape latency: duration of trial
  + Maximum: 59
  + Platform and obstacle locations changed every 8 days
* Coordinate system performs better
  + Not drastic

*Pure TD Learning and Echolocation:*

* Major upgrade in results
* Faster adaptation to platform change
* Less catastrophic forgetting (loops)

*Coordinate System and Echolocation:*

* Similar upgrade, not as drastic because coordinate system eventually takes over
* Key: echolocation mostly restricts action space at each time step
  + Less useless actions
  + Sometimes action is optimal because reflected off platform
  + Useless actions that occur far away are also eliminated! Very powerful in that sense

*Significance and Future Work:*

* Echolocation can be applied
  + More interaction with environment, faster learning
* Incorporation of perception must happen eventually (use vision or sound to drive learning in RL agents)
* Computational neuroscience: more thorough understanding of how mammals navigate the world!
* Future work:
  + Environment: video games, 3D, good acoustics
  + Learn sound waves classification during simulations (very hard)
  + Test more complex Actor-Critics (SAC? Deep RL?)
    - Novice in the field