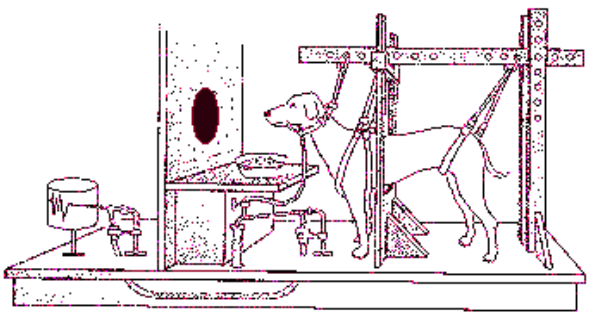
CSE/NEURO 528 Lecture 13: Reinforcement Learning & Course Review

(Chapter 9)



Animation: Tom Creed, SJU

Early Results: Pavlov and his Dog

- Classical (Pavlovian) conditioning experiments
- Training: Bell →Food
- After: Bell → Salivate
- Conditioned stimulus (bell) predicts future reward (food)

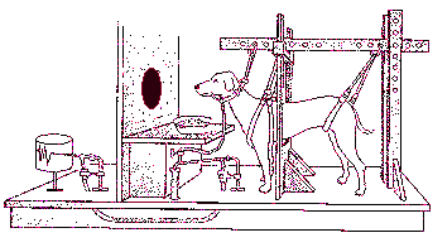


Image: Wikimedia Commons; Animation: Tom Creed, SJU

Predicting Delayed Rewards

- How do we predict rewards delivered *some time after* a stimulus is presented?
- Given: Many trials, each of length T time steps
- Time within a trial: $0 \le t \le T$ with stimulus $u(t)$ and reward $r(t)$ at each time step t (Note: $r(t)$ can be zero for some t)
- We would like a neuron whose output $v(t)$ predicts the *expected total future reward* starting from time t

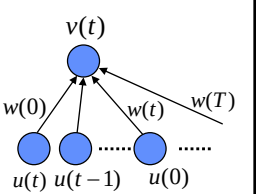
$$v(t) \approx \left\langle \sum_{\tau=0}^{T-t} r(t+\tau) \right\rangle_{trials}$$

Learning to Predict Future Rewards

- Use a set of synaptic weights $w(t)$ and *predict based on all past stimuli* $u(t)$:

$$v(t) = \sum_{\tau=0}^{t} w(\tau) u(t-\tau)$$

(Linear filter!)



- Learn weights $w(\tau)$ that minimize error:

$$\left(\sum_{\tau=0}^{T-t} r(t+\tau) - v(t) \right)^2$$

(Can we minimize this using gradient descent and delta rule?)

Yes, BUT future rewards are not yet available!

Temporal Difference (TD) Learning

- **Key Idea**: Rewrite error function to get rid of future terms:

$$\left(\sum_{\tau=0}^{T-t} r(t+\tau) - v(t)\right)^2 = \left(r(t) + \sum_{\tau=0}^{T-t-1} r(t+1+\tau) - v(t)\right)^2$$

$$\approx \left(r(t) + v(t+1) - v(t)\right)^2$$

Minimize this using gradient descent!

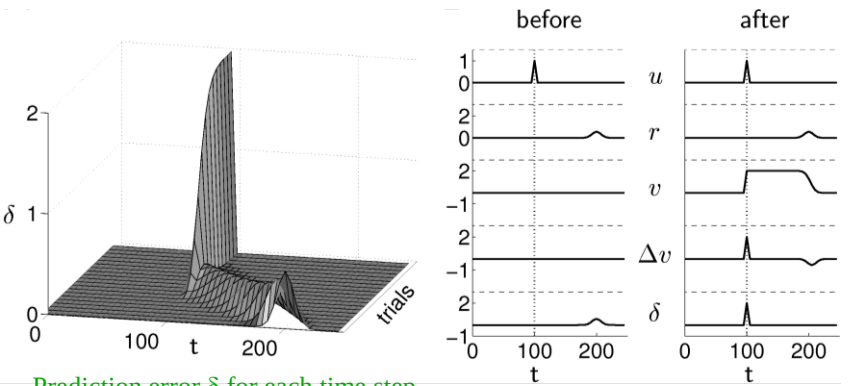
- **Temporal Difference (TD) Learning**:

$$\Delta w(\tau) = \varepsilon \underbrace{[\overbrace{r(t) + v(t+1)}^{\text{Expected future reward}} - v(t)]}_{\delta} u(t-\tau)$$

Prediction

Predicting Future Rewards: TD Learning

Stimulus at t = 100 and reward at t = 200



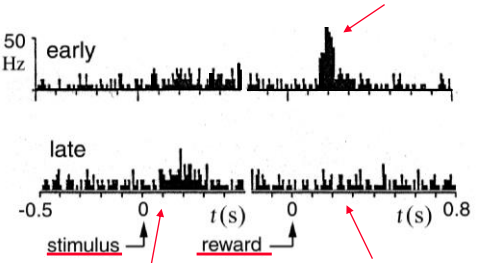
Prediction error δ for each time step (over many trials)

Image Source: Dayan & Abbott textbook

Possible Reward Prediction Error Signal in the Primate Brain

Dopaminergic cells in Ventral Tegmental Area (VTA)

Reward Prediction error δ? $[r(t)+v(t+1)-v(t)]$



Before Training

After Training

No error

$[0+v(t)-v(t-1)]$

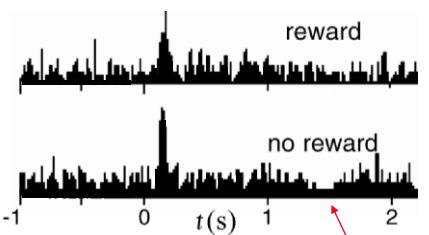
$v(t) \approx r(t)+v(t+1)$

$[r(t)+v(t+1)-v(t)] \approx 0$

Image Source: Dayan & Abbott textbook

More Evidence for Prediction Error Signals

Dopaminergic cells in VTA after Training



Negative error

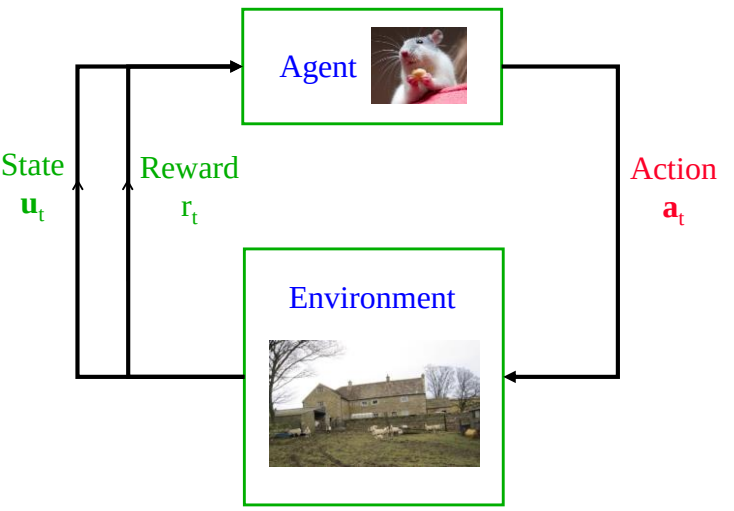
$r(t)=0, v(t+1)=0$

$[r(t)+v(t+1)-v(t)]=-v(t)$

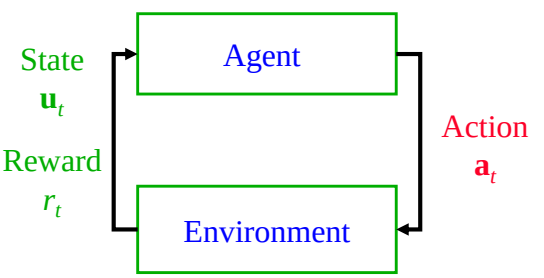
Reward expected but not delivered

Image Source: Dayan & Abbott textbook

Reinforcement Learning: Acting to *Maximize Rewards*



The Problem



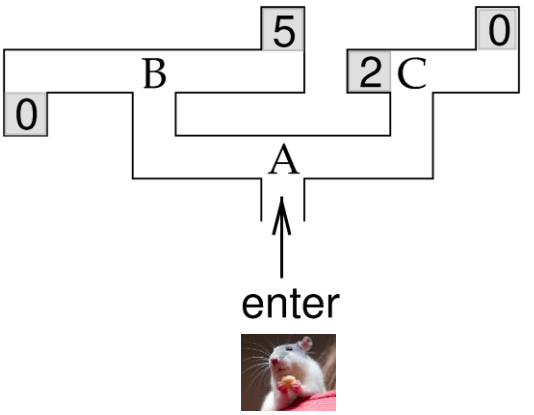
Learn a state-to-action mapping or "policy":

$$\pi(\mathbf{u}) = \mathbf{a}$$

which maximizes the expected total future reward:

$$\left\langle \sum_{\tau=0}^{T-t} r(t+\tau) \right\rangle_{trials}$$

Example: Rat in a barn



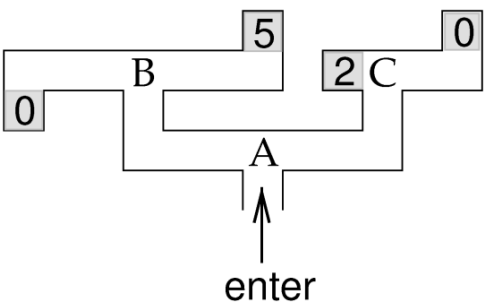
States = locations A, B, or C

Actions= L (go left) or R (go right)

If the rat chooses L or R *at random* (random "policy"), what is the expected reward (or "*value*") *v* for each state?

Image Source: Dayan & Abbott textbook

Policy Evaluation



For random policy:

$$v(B) = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 5 = 2.5$$

$$v(C) = \frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 0 = 1$$

$$v(A) = \frac{1}{2}\cdot v(B) + \frac{1}{2}\cdot v(C) = 1.75$$

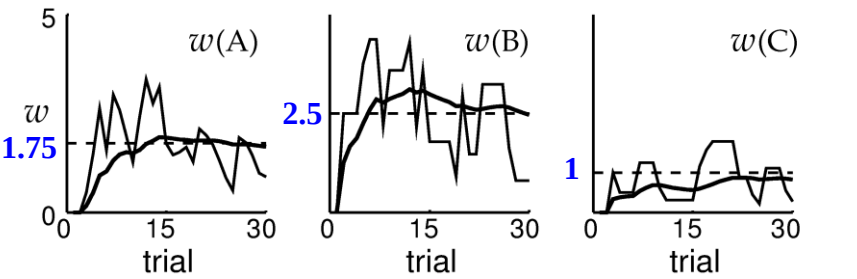
Let value of state u
$v(u)$ = weight $w(u)$

Can learn value of states using TD learning:

$$w(u) \leftarrow w(u) + \varepsilon\,[r(u) + v(u') - v(u)]$$

(Location, action) ⇨ new location i.e., (u,a) ⇨ u'

TD Learning of Values for Random Policy

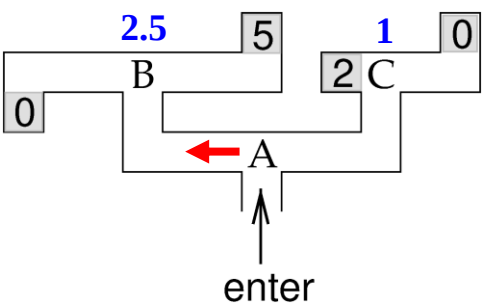


(For all three, ε = 0.5)

Once I know the values, I can pick the action that leads to the higher valued state!

Image Source: Dayan & Abbott textbook

Selecting Actions based on Values



Values act as surrogate immediate rewards → Locally optimal choice leads to globally optimal policy for "Markov" environments (Related to *Dynamic Programming*)

Putting it all together: Actor-Critic Learning

- Two separate components: Actor (selects action and maintains policy) and Critic (maintains value of each state)

1. Critic Learning ("Policy Evaluation"):
   Value of state u = $v(u)$ = $w(u)$

   $$w(u) \leftarrow w(u) + \varepsilon\,[r(u) + v(u') - v(u)]$$ (same as TD rule)

2. Actor Learning ("Policy Improvement"):

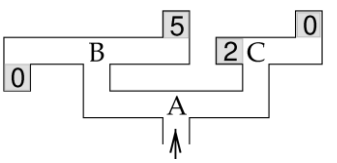
   $$P(a;u) = \frac{\exp(\beta Q_a(u))}{\sum_b \exp(\beta Q_b(u))}$$ Probabilistically select an action a at state u

   For all actions a':

   $$Q_{a'}(u) \leftarrow Q_{a'}(u) + \varepsilon[r(u) + v(u') - v(u)](\delta_{aa'} - P(a';u))$$

3. Repeat 1 and 2

Actor-Critic Learning in our Barn Example



Probability of going Left at each location

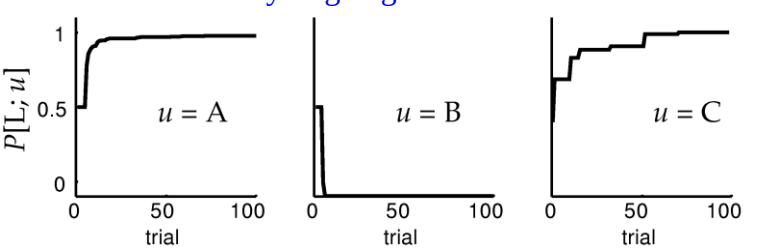
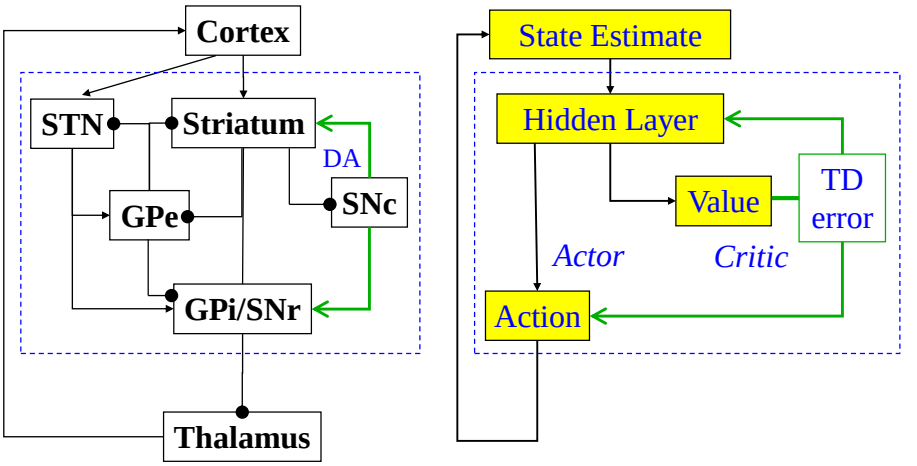


Image Source: Dayan & Abbott textbook

Possible Implementation of the Actor-Critic Model in the Basal Ganglia



(See Supplementary Materials for references)

Reinforcement learning has been applied to many real-world problems!

Example:

Google's AlphaGo beats human champion in Go,

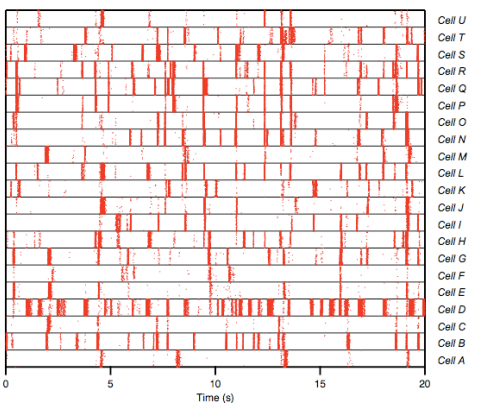
Autonomous Helicopter Flight
(learned from human demonstrations)

(Videos and papers at: http://heli.stanford.edu/)

Course Summary

- **Where have we been?**
  - Course Highlights
- **Where do we go from here?**
  - Challenges and Open Problems
- **Further Reading**

What *is* the neural code?



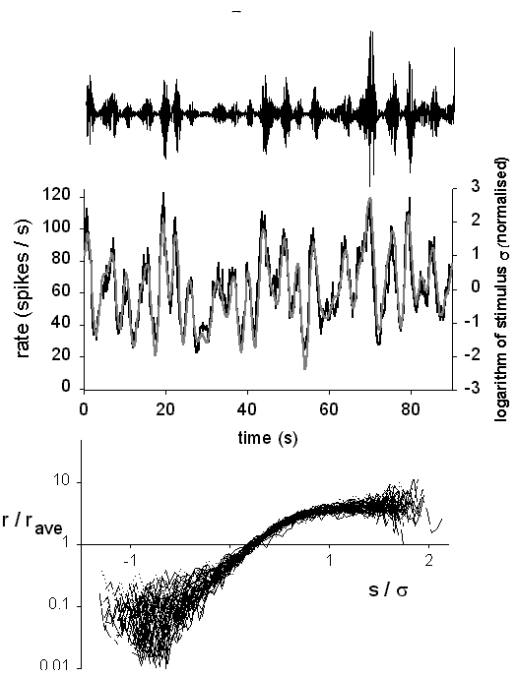
What is the nature of the code?
Representing the spiking output:
single cells vs populations
rates vs spike times vs intervals
What features of the stimulus does the neural system represent?

Encoding and decoding neural information

Encoding: building functional models of neurons/neural systems and predicting the spiking output given the stimulus

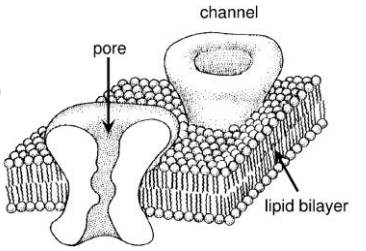
Decoding: what can we say about the stimulus given what we observe from the neuron or neural population?

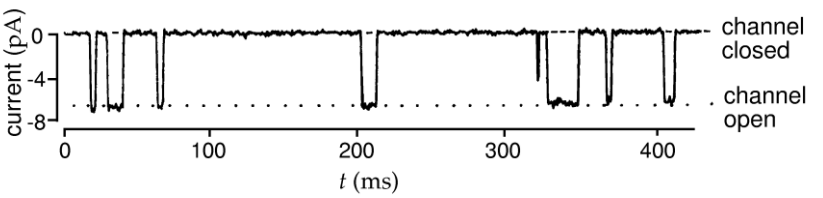
Information maximization as a design principle of the nervous system



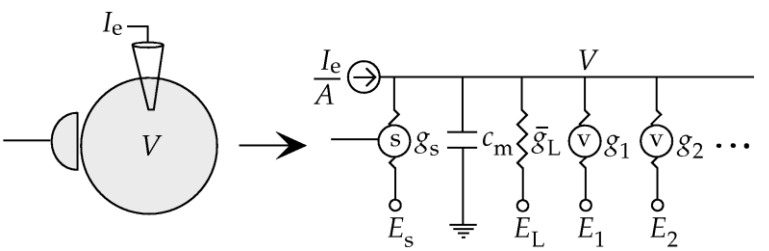
Biophysical Models of Neurons

- Voltage dependent
- transmitter dependent (synaptic)
- Ca dependent





The neural equivalent circuit



Ohm's law: $V = IR$ and Kirchhoff's law

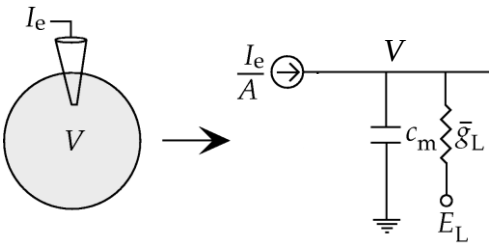
$$-C_m \frac{dV}{dt} = \sum_i g_i(V - E_i) + I_e$$

Capacitive current

Ionic currents

Externally applied current

Simplified models: integrate-and-fire

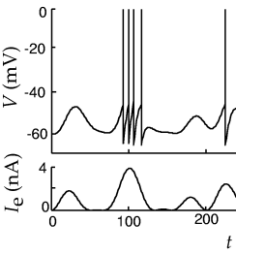


Integrate-and-Fire Model

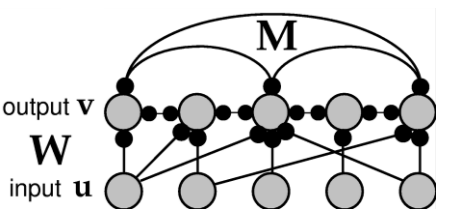
$$\tau_m \frac{dV}{dt} = -(V - E_L) + I_e R_m$$

If V > $V_{threshold}$ → Spike

Then reset: V = V_{reset}



Modeling Networks of Neurons



$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + F(\mathrm{W}\mathbf{u} + \mathrm{M}\mathbf{v})$$

Output Decay Input Feedback

Unsupervised Learning

- For linear neuron $v = \mathbf{w}^T\mathbf{u} = \mathbf{u}^T\mathbf{w}$
- Basic Hebb Rule: $\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{u}v$

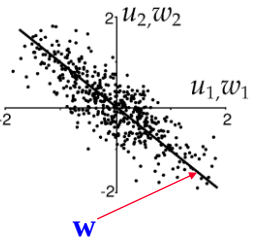
- Average effect over many inputs:

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle \mathbf{u}v \rangle = Q\mathbf{w}$$

- Q is the input correlation matrix:

$$Q = \langle \mathbf{u}\mathbf{u}^T \rangle$$

Hebb rule performs principal component analysis (PCA)



The Connection to Statistics

Unsupervised learning = learning the *hidden causes* of input data

Causes **v** $p[\mathbf{v}\,|\,\mathbf{u};G]$ (posterior)

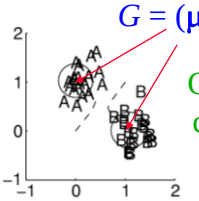
Generative model

Recognition model

Data **u** $p[\mathbf{u}\,|\,\mathbf{v};G]$ (data likelihood)

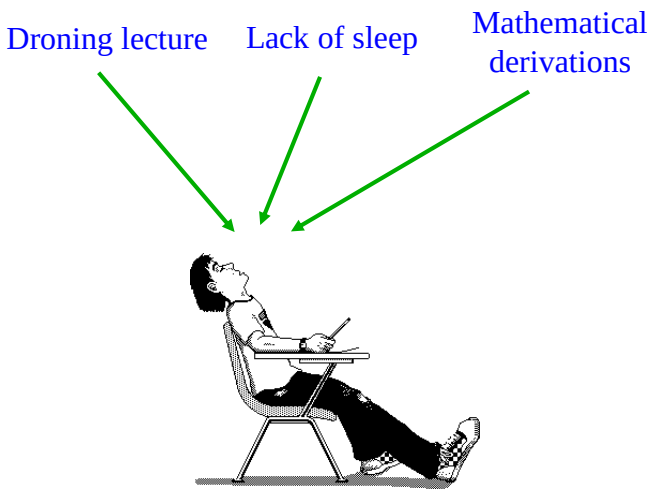
Use EM algorithm for learning

$G = (\mu_v, \sigma_v)$



Causes of clustered data

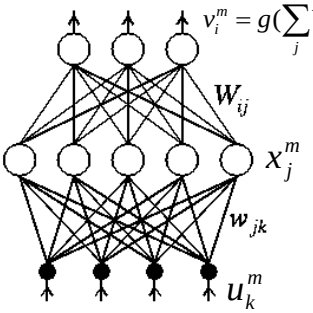
Generative Models



Supervised Learning

Backpropagation for Multilayered Networks

$$v_i^m = g(\sum_j W_{ij}\, g(\sum_k w_{jk} u_k^m))$$



Goal: Find **W** and **w** that minimize errors:

$$E(W_{ij}, w_{jk}) = \frac{1}{2}\sum_{m,i}(d_i^m - v_i^m)^2$$

Desired output

Gradient descent learning rules:

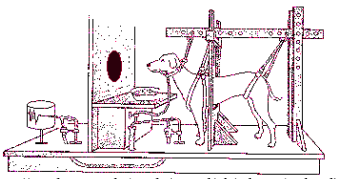
$$W_{ij} \rightarrow W_{ij} - \varepsilon \frac{\partial E}{\partial W_{ij}}$$ (Delta rule)

$$w_{jk} \rightarrow w_{jk} - \varepsilon \frac{\partial E}{\partial w_{jk}} = w_{jk} - \varepsilon \frac{\partial E}{\partial x_j^m} \cdot \frac{\partial x_j^m}{\partial w_{jk}}$$ (Chain rule)

Reinforcement Learning

- Learning to predict rewards:

$$w \rightarrow w + \varepsilon(r - v)u$$

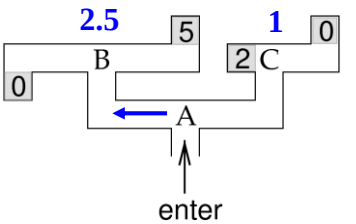


(http://employees.csbsju.edu/tcreed/pb/pdoganim.html)

- Learning to predict **delayed rewards** (TD learning):

$$w(\tau) \rightarrow w(\tau) + \varepsilon[r(t) + v(t+1) - v(t)]u(t-\tau)$$

- Actor-Critic Learning:
  - Critic learns value of each state using TD learning
  - Actor learns best actions based on value of next state (using the TD error)



The Future: Challenges and Open Problems

- How do neurons encode information?
  - **Topics**: Synchrony, Spike-timing based learning, Dynamic synapses
- Does a neuron's structure confer computational advantages?
  - **Topics**: Role of channel dynamics, dendrites, plasticity in channels and their density
- How do networks implement computational principles such as efficient coding and Bayesian inference?
- How do networks learn "optimal" representations of their environment and engage in purposeful behavior?
  - **Topics**: Unsupervised/reinforcement/imitation learning

Further Reading (for Spring and beyond)

- *Spikes: Exploring the Neural Code,* F. Rieke et al., MIT Press, 1997
- *The Biophysics of Computation,* C. Koch, Oxford University Press, 1999
- *Large-Scale Neuronal Theories of the Brain*, C. Koch and J. L. Davis, MIT Press, 1994
- *Probabilistic Models of the Brain*, R. Rao et al., MIT Press, 2002
- *Bayesian Brain*, K. Doya et al., MIT Press, 2007
- *Reinforcement Learning: An Introduction*, R. Sutton and A. Barto, MIT Press, 1998

Next two classes: Project presentations!

- Keep your presentation short: **~7-8 slides, 8 + 3 mins mins/group (with questions)**
  - Introduction, Background, Methods, Results, Conclusion
- Slides:
  - Bring your slides on a USB stick to use the class laptop (Windows machine)

  OR
  - Bring your own laptop (esp if you have videos etc.)
- Projects reports (10-15 pages total) due **March 12** (by email to both Adrienne, Rich, and Raj before midnight)


Have a great weekend!
Neural Encoding for Dummies