Applied Machine Learning!!!

W207 Section 9

Rasika Bhalerao
rasikabh@berkeley.edu

Aug 23: Welcome!
Nov 8 and 22: No classes

Schedule

Supervised learning methods

	Sync	Topic
2	Aug 30	Linear Regression / Gradient Descent
3	Sep 6	Feature Engineering Bonus: Naive Bayes
4	Sep 13	Logistic Regression
5	Sep 20	Multiclass classification / Eval Metrics Bonus: Reinforcement learning
6	Sep 27	Neural Networks
7	Oct 4	KNN, Decision Trees, Ensembles

Unsupervised learning methods

	Sync	Topic
8	Oct 11	KMeans and PCA Bonus: LDA
9	Oct 18	Text Embeddings Bonus: Language models
10	Oct 25	CNNs Bonus: GANs
11	Nov 1	EDA, Real data, Baselines
12	Nov 15	Fairness / Ethics
13	Nov 29	Fancy Neural Networks
14	Dec 6	Final Presentations

Assignment Schedule

Due Date	Assignment
Aug 28	HW1
Sep 4	HW2
Sep 11	HW3
Sep 18	HW4
Sep 25	HW5
Oct 2	HW6
Oct 16	Group project baseline
Oct 23	HW8
Nov 6	HW9
Nov 20	HW10
Dec 4	Final project notebook + presentation

Behavior expectations

- Healthy disagreement is expected
- Be mindful of one another’s schedules
- Be a good listener
- Have fun in a professional manner
- Share related real-world experience
- Ask questions when something is confusing
- Keep it 100 but be respectful
- Be open-minded to new ideas in the real world and when coding
- On time for group meetings

Has anyone not signed up for a final project group?

https://docs.google.com/document/d/1R3J_X1Rz6WP8eMQ2cyMC0wAr5iQdhMK_httdoNO6L0w/edit?usp=sharing

Softmax

What? Why?

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

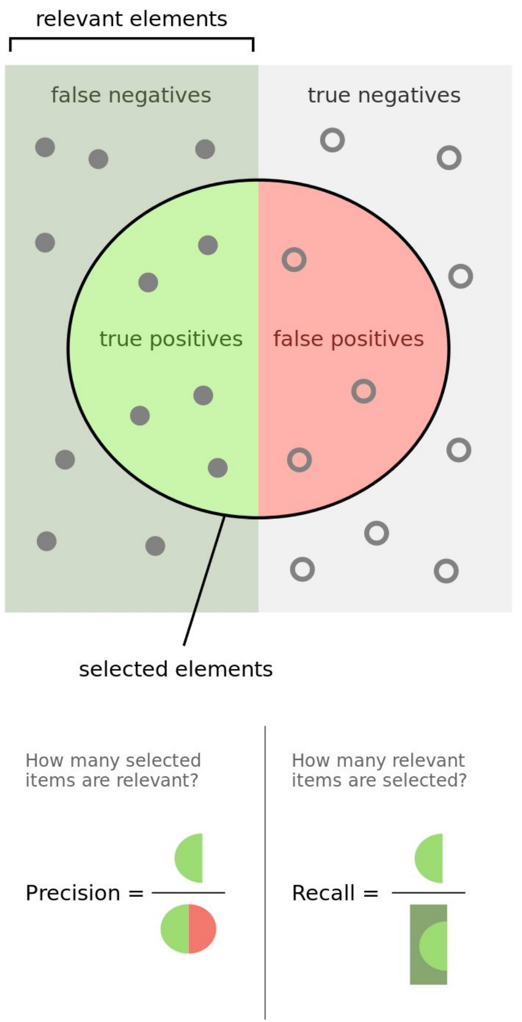
Image of equation: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Precision, Recall, and F1

	Ground truth positive	Ground truth negative
Tested positive	True positive (TP)	False positive (FP)
Tested negative	False negative (FN)	True negative (TN)

- Precision
  - Out of those tested positive, how many are truly positive?
  - TP / (TP + FP)
- Recall
  - Out of those truly positive, how many tested positive?
  - TP / (TP + FN)
- F1

$$\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$$



https://en.wikipedia.org/wiki/F1_score

More than two classes

- **Multivalue classification:** each document can belong to 0, 1, or >1 classes
- **Multinomial classification:** each document belongs to exactly 1 class

- Precision, recall, and F1 score are calculated for each class
  - **Macroaveraging:** get the scores for each class, then calculate the (unweighted) average
  - **Microaveraging:** count all the TP, FP, and FN for all classes and then calculate together
- “Accuracy” is simply the percent of documents classified correctly
  - Why is this less robust?

Cross-entropy

A way to measure the difference between these two:

0.8	1
0.02	0
0.06	0
0.11	0
0.01	0

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Image of equation: https://levelup.gitconnected.com/grokking-the-cross-entropy-loss-cda6eb9ec307

Confusion matrix

For each pair of classes <c_1, c_2>, how many documents from c_1 were assigned to c_2?

	Assigned positive	**Assigned neutral**	**Assigned negative**
True positive	8	3	2
True neutral	4	3	7
True negative	2	6	3

View options (at the top) → Annotate

Async Practice Quiz Questions (vote!)

As you adjust the classification threshold, when precision increases, recall increases too.	True	False
In a multiclass logistic regression model, each class has its own set of parameters.	True	False
Cross-entropy loss is only valid when the label distribution is one-hot.	True	False
Logistic regression output can be computed with a single matrix multiplication (and a sigmoid).	True	False
A linear model must rely on handcrafted input features if it is to understand complex structure.	True	False
A model trained for the MNIST digit classification task could be applied directly to mail sorting by zip code.	True	False

Social bias in classification

No notebook this week (same as last week's binary classification code) https://medium.com/syncedreview/yann-lecun-cake-analogy-2-0-a361da560dae

Reinforcement learning

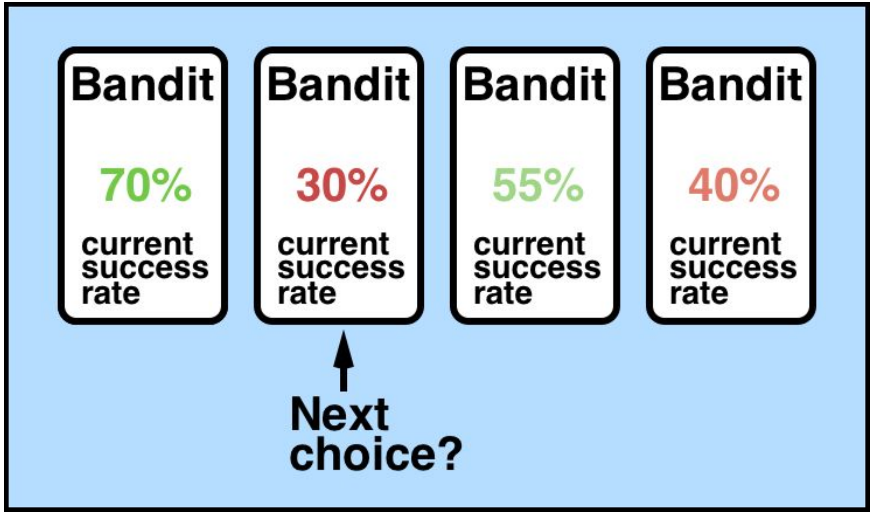
Y. LeCun

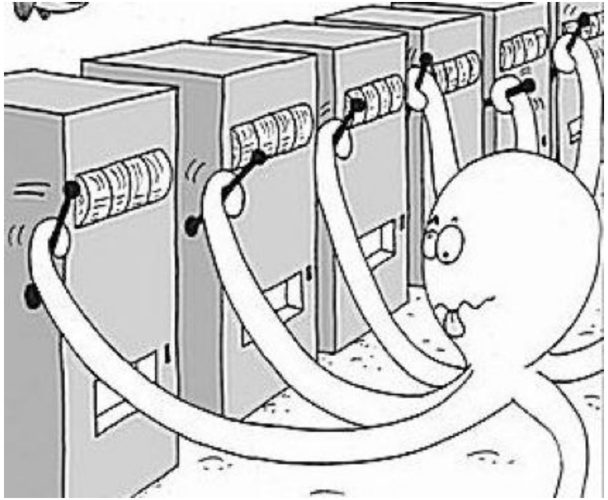
How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Multi-Armed Bandit Problem

- Slot machine with n arms (bandits)
- Each arm has a different probability of giving you the reward
- Pulling an arm stochastically gives a reward (1) or failure (0)
- Task: maximize reward in the long run

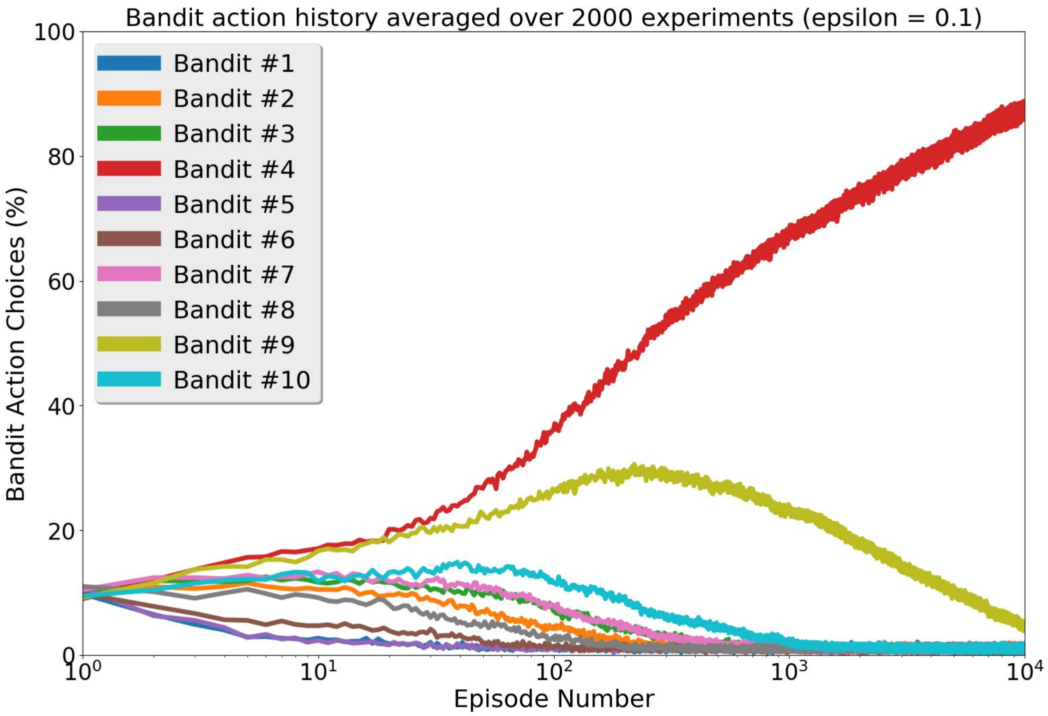




https://towardsdatascience.com/solving-the-multi-armed-bandit-problem-b72de40db97c

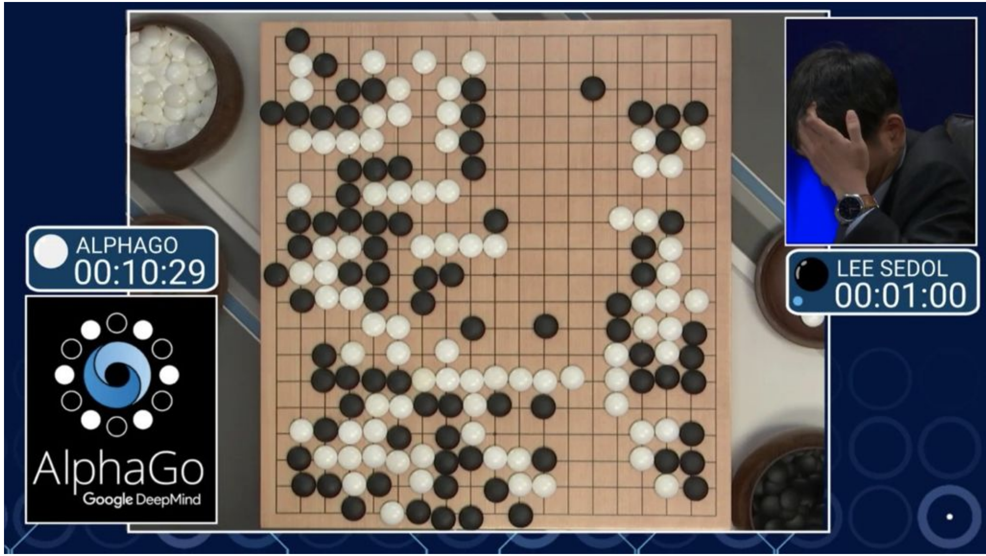
Which arm to pull?

- Pull all arms equally often?
- Only pull the arm that has given the best results so far?
- Mostly pull the “best” arm, but sometimes the others?
- An example of the exploration/exploitation dilemma

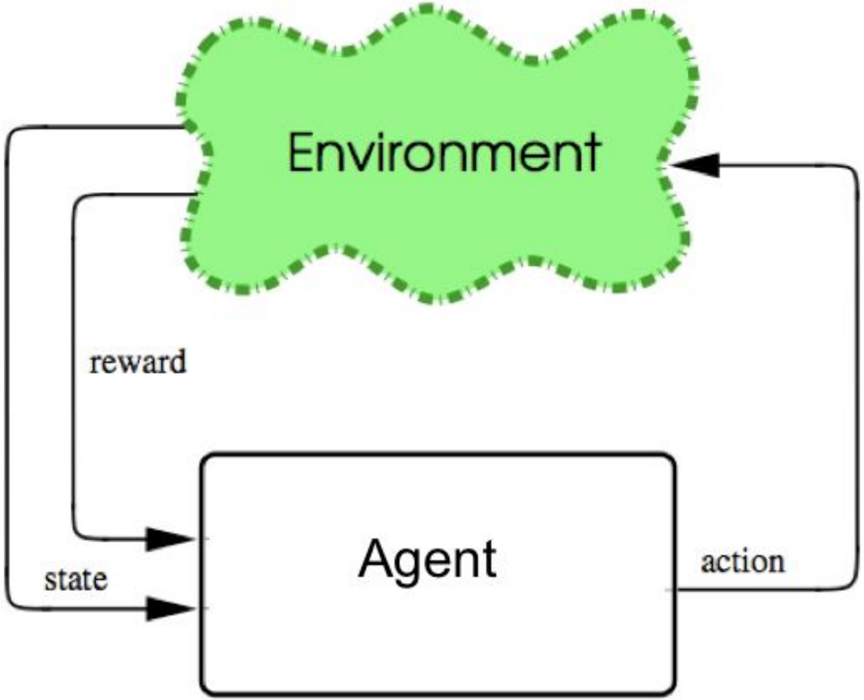


What is reinforcement learning?

- Learning well-performing behavior from state observations and rewards
- Tree search:
    - Huge trees
    - State evaluation is hard
    - Each action selection may take a long time
- Supervised learning:
    - Agent is limited by expertise of expert from whom it learns
- Reinforcement learning:
    - Agent learns just from observations and rewards



Reinforcement Learning



$$s_t \xrightarrow{a_t} \overset{\overset{r_{t+1}}{\uparrow}}{s_{t+1}} \xrightarrow{a_{t+1}} \overset{\overset{r_{t+2}}{\uparrow}}{s_{t+2}} \xrightarrow{a_{t+2}} \overset{\overset{r_{t+3}}{\uparrow}}{s_{t+3}} \xrightarrow{a_{t+3}} \dots$$

Example rewards: PacMan

- One example:
  - 1 if you eat a pill
  - -10 if you get caught by a ghost
  - 2 if you eat a power pill or eat a ghost
  - 0 otherwise
- Another example:
  - -1 at every time step
  - 1,000,000 if you win the level

Markov Decision Processes

- S: finite set of states (state space). s ∈ S
- A: finite set of actions. a ∈ A
- R: finite set of rewards. r ∈ R
- State transition probabilities:

$$P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

- Policy:

$$\pi(s,a) = \Pr\{a_t = a \mid s_t = s\}$$

- Reward function:

$$R^a_{ss'} = \mathrm{E}\{r_{t+1} \mid s_t = s, a_t = a\}$$

s_t = state at time t
a_t = action chosen at time t
r_t = reward at time t (depends on time t-1)

Markov Property

Probability of the next state and reward only depend on the immediately preceding state and action; it doesn't matter what happened before that

$$P_{ss'}^{a} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_0, a_0, r_0\}$$

$$\equiv$$

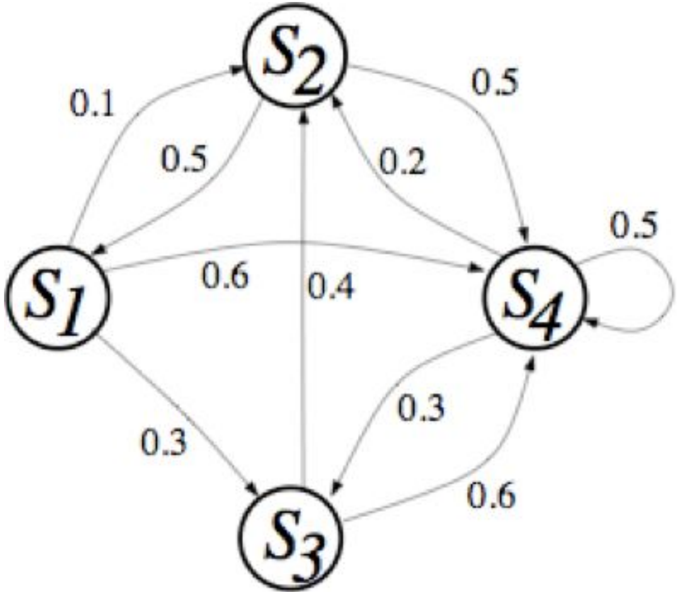
$$P_{ss'}^{a} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

State Transition Probabilities

- Example with four states, considering one action
- Probability of going from one state to the other if the action is taken
- Each row sums to 1, outbound edges for each node sum to 1

	s_1	s_2	s_3	s_4
s_1	0	0.1	0.3	0.6
s_2	0.5	0	0	0.5
s_3	0	0.4	0	0.6
s_4	0	0.2	0.3	0.5

$$P^a_{ss'}$$



Policy

- Determines agent's behavior
- Agent's goal: find the best policy
- Probability of taking action a when in state s
- In general, the policy is stochastic:

$$\pi(s,a) = \Pr\{a_t = a \mid s_t = s\}$$

- If the policy is deterministic (we know which action to use) we can write:

$$\pi(s) \rightarrow a$$

Reward Function

- Aka. expected return
- Agent's overall goal: find the policy that maximizes this
- Cumulative reward that the agent will receive from time t until the end of the game, if they take this action now

$$R^{a}_{ss'} = \mathrm{E}\{r_{t+1} \mid s_t = s, a_t = a\}$$

- γ close to 0 → agent cares more about immediate reward: shortsighted
- γ close to 1 → agent cares more about future rewards: farsighted
- γ = 1 → sum doesn't converge for infinite time steps (fine if each episode always has finite steps)

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \quad 0 \le \gamma \le 1$$

Value Function

Answers question: how "good" is this state or action?

Defined based on rewards expected from that state or action

Uses policy π to determine the value, given values of next states

- State-value:
  - What is the value of this state s, given policy π?
- Action-value:
  - What is the value of taking action a, from state s, given policy π?

State-value function

Value of state s for an agent following policy π

Expected return, starting state s at time t following policy π

$$
\begin{aligned}
v_\pi(s) &= E_\pi\left[G_t \mid S_t = s\right] \\
&= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]
\end{aligned}
$$

Action-value function

Value of action a for an agent in state s following policy π

Expected return, starting state s at time t, taking action a, then following policy π

$$
\begin{aligned}
q_\pi(s,a) &= E_\pi\left[G_t \mid S_t = s, A_t = a\right] \\
&= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]
\end{aligned}
$$

This is the Q-function, and its output is the Q-value!

How does the agent find the best policy?

Optimal policy π: the policy with expected return ≥ all other policies':

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \boldsymbol{S}$$

Based on that, the optimal state-value function for state s is:

$$v_*(s) = \max_\pi v_\pi(s)$$

And the optimal action-value function for state s is:

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

So, v* is the largest expected return possible for each state s

And q* is the largest expected return possible for each state-action pair (s,a).

Bellman optimality equation for q*

q* must satisfy the Bellman equation:

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

Exercise: why is this true?

Q-learning

- Objective: find the optimal policy by learning the optimal Q-values for each action-state pair
- Reminder: Q-function takes action-state pair, and returns expected return starting state s at time t, taking action a, then following policy π

$$q_*(s,a) - q(s,a) = loss$$

$$E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right] - E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] = loss$$

Q-learning: updating Q-values

Learning rate α: number between 0 and 1
Update Q-value q(s,a) with weighted sum of old and learned values
Higher learning rate → more quickly adopt new Q-value

$$q^{new}(s,a) = (1-\alpha)\underbrace{q(s,a)}_{\text{old value}} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)}^{\text{learned value}}$$

Exploration vs. Exploitation

- Exploitation: take good actions in each state already taken before to maximize reward
- Exploration: take a chance on actions that may have lower value in order to learn more, and maybe find true best action to later exploit
- Need to balance the two!

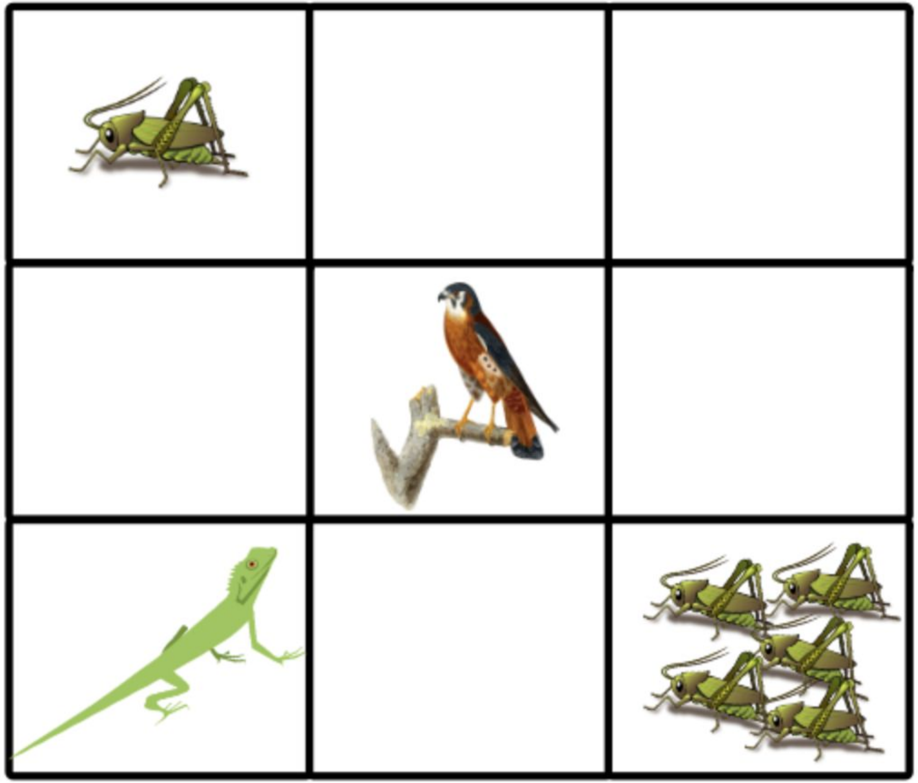
Exploration vs. Exploitation

- ε-greedy policy
  - Select greedy action 1-ε% of the time (exploit), and a random action ε% of the time (explore)
  - ε decays over time
- Stochastic policy
  - Use action values to select actions probabilistically

    $$\pi(s,b) = \frac{e^{Q(s,b)/\tau}}{\sum_a e^{Q(s,a)/\tau}}, \text{ where } \tau > 0 \text{ is the } \textit{temperature}$$

    High temperatures increase exploration by making policy more random
    Low temperatures increase exploitation by making policy more greedy

Q-learning example: The Lizard Game



Agent: lizard

Goal: Eat as many crickets as possible as fast as possible without meeting a bird

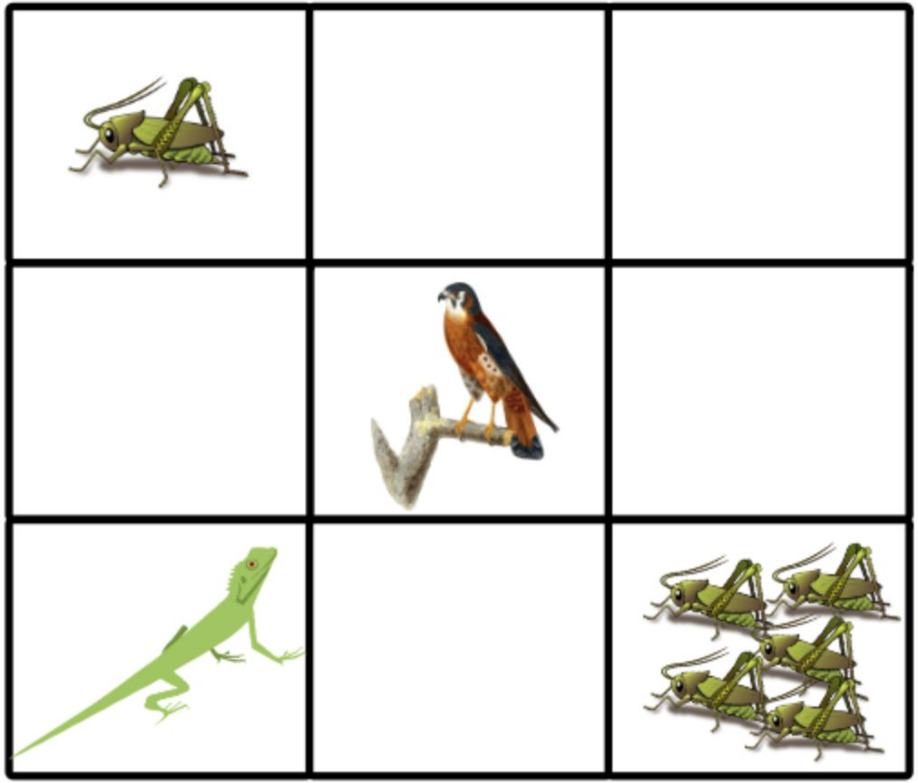
Actions: up, down, left, right

States: tiles

Rewards:

State	Reward	Game over?
1 cricket	1	No
Empty	-1	No
5 crickets	10	Yes
Bird	-10	Yes

Q-learning example: The Lizard Game

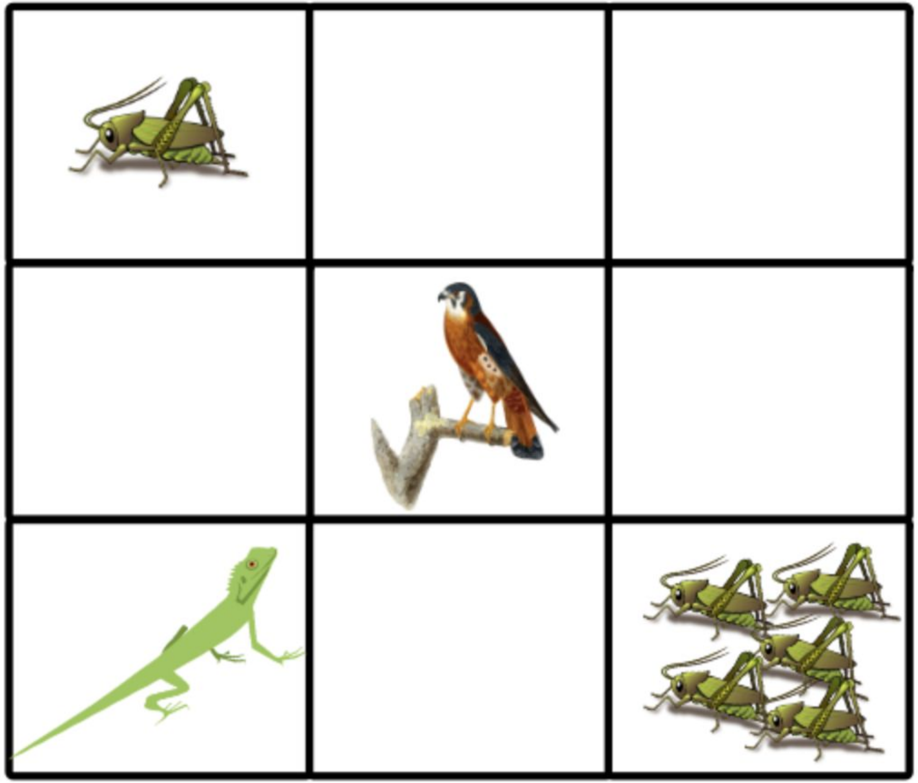


Q-table:

		Actions			
		Left	Right	Up	Down
States	1 cricket	0	0	0	0
	Empty 1	0	0	0	0
	Empty 2	0	0	0	0
	Empty 3	0	0	0	0
	Bird	0	0	0	0
	Empty 4	0	0	0	0
	Empty 5	0	0	0	0
	Empty 6	0	0	0	0
	5 crickets	0	0	0	0

Update Q-values in this table

Q-learning example: The Lizard Game



What would happen if we only did exploitation?

What would happen if we only did exploration?

State	Reward	Game over?
1 cricket	1	No
Empty	-1	No
5 crickets	10	Yes
Bird	-10	Yes

Q-learning example: The Lizard Game

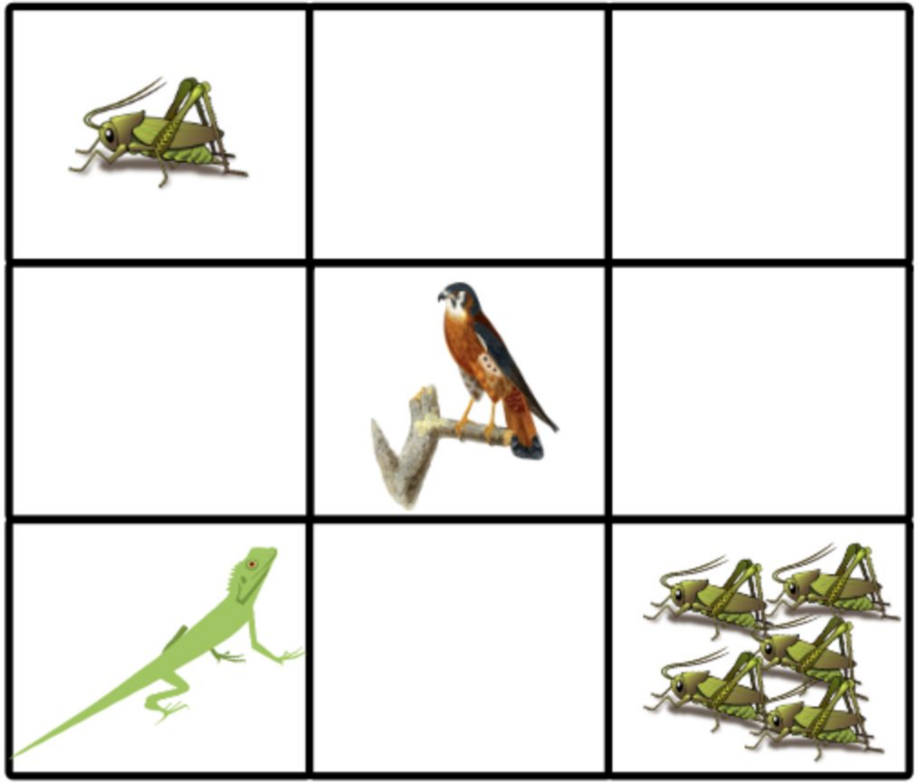
- First step: explore, move one tile to the right. Reward: -1
- Suppose discount rate γ = 0.99 and learning rate α = 0.7

$$q^{new}(s,a) = (1-\alpha)\ \underbrace{q(s,a)}_{\text{old value}} + \alpha \overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)}^{\text{new value}}$$
$$= (1-0.7)(0) + 0.7\left(-1 + 0.99\left(\max_{a'} q(s',a')\right)\right)$$

- To find maximum Q-value over all actions from s', check table (currently all 0)

$$\begin{aligned} &= (1-0.7)(0) + 0.7(-1+0.99(0)) \\ &= 0 + 0.7(-1) \\ &= -0.7 \end{aligned}$$

Q-learning example: The Lizard Game



	Actions			
	Left	Right	Up	Down
1 cricket	0	0	0	0
Empty 1	0	0	0	0
Empty 2	0	0	0	0
States Empty 3	0	0	0	0
Bird	0	0	0	0
Empty 4	0	0	0	0
Empty 5	0	**-0.7**	0	0
Empty 6	0	0	0	0
5 crickets	0	0	0	0