Reinforcement Learning

CMSC 678

UMBC

Announcement 1: Exam 2

Friday May 18th, 1pm-3pm

You can bring any of *your own* hard-copy notes (or course material photocopies)

you don't need to turn them in

Allowed	Not Allowed
• Your notes • CIML, ESL, UML, ITILA photocopies/printouts • Slide printouts • Cheat sheets that *you've* written (or typed) yourself • Notes you've transcribed from online sources (w/ citation)	• Your computer/phone • Your friend's notes • Any full (bound) textbooks • *Printouts* of non-course materials (e.g., printouts from Coursera's ML course)

(don't assume you'll have time to continuously flip through your notes)

Announcement 2: Final Project

Due: Wednesday May 23, 11:59 AM

Turn in the report, code, (best) model(s), and any new data

Recap from last time…

Ensembles

“Wisdom of the crowd:" groups of people can often make better decisions than individuals

Reuse previous classifiers

Boosting — a method that takes classifiers that are only slightly better than chance and learns an arbitrarily good classifier

Courtesy Hamed Pirsiavash

Voting Multiple Classifiers

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging

Courtesy Hamed Pirsiavash

Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

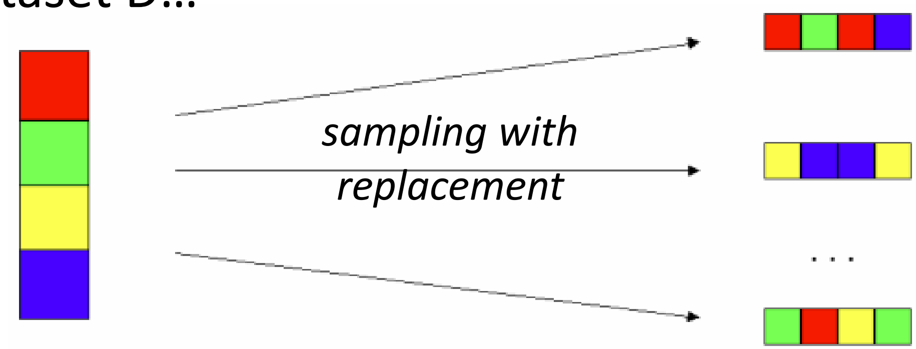
Q: What can go wrong with option 1?

A: Small sample → poor performance

Option 2: Bootstrap aggregation (bagging) resampling

- Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D
- Train classifiers on each dataset and average their predictions

Given a dataset D…



get new datasets $\hat{D}$ by random sampling with replacement from D

Courtesy Hamed Pirsiavash

Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size **k** and pick the best among these

Train decision trees of depth **d**

Average results from multiple randomly trained trees

Q: What's the difference between bagging decision trees and random forests?

A: Bagging → highly correlated trees (reuse good features)

Boosting weak learners

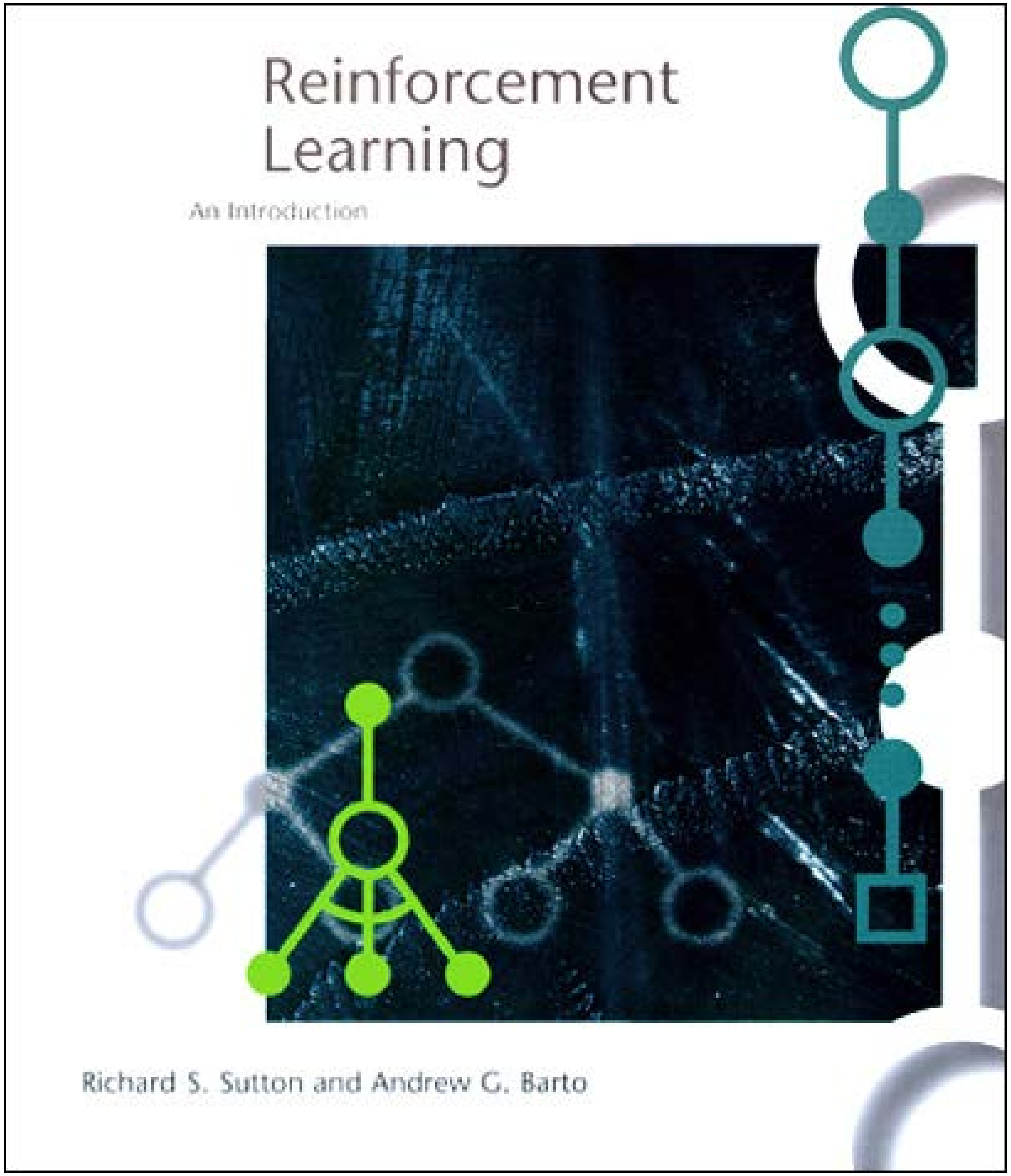
Boosting takes a poor learning algorithm (weak learner) and turns it into a good learning algorithm (strong learner)

Intuition behind AdaBoost: study for an exam by taking past exams

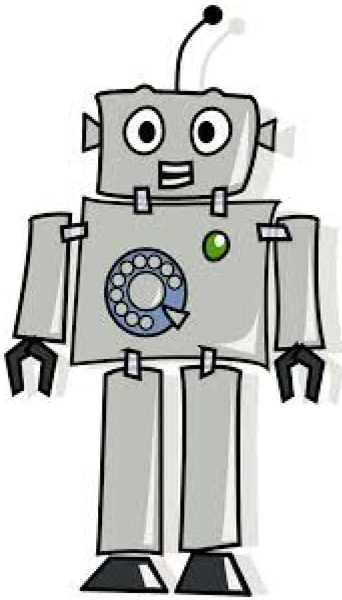
1. Take the exam
2. Pay less attention to questions you got right
3. Pay more attention to questions you got wrong
4. Study more, and go to step 1

There's an entire book!

http://incompleteideas.net/book/the-book-2nd.html



Reinforcement Learning



agent

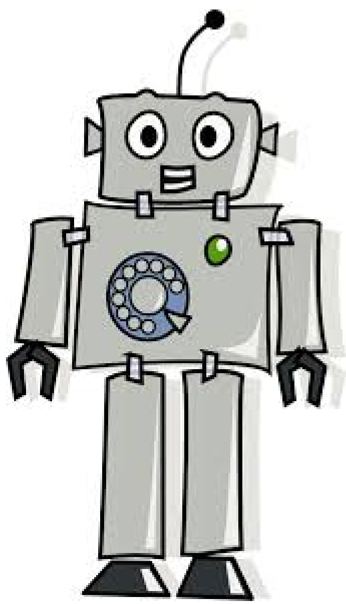
environment

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Reinforcement Learning

take action



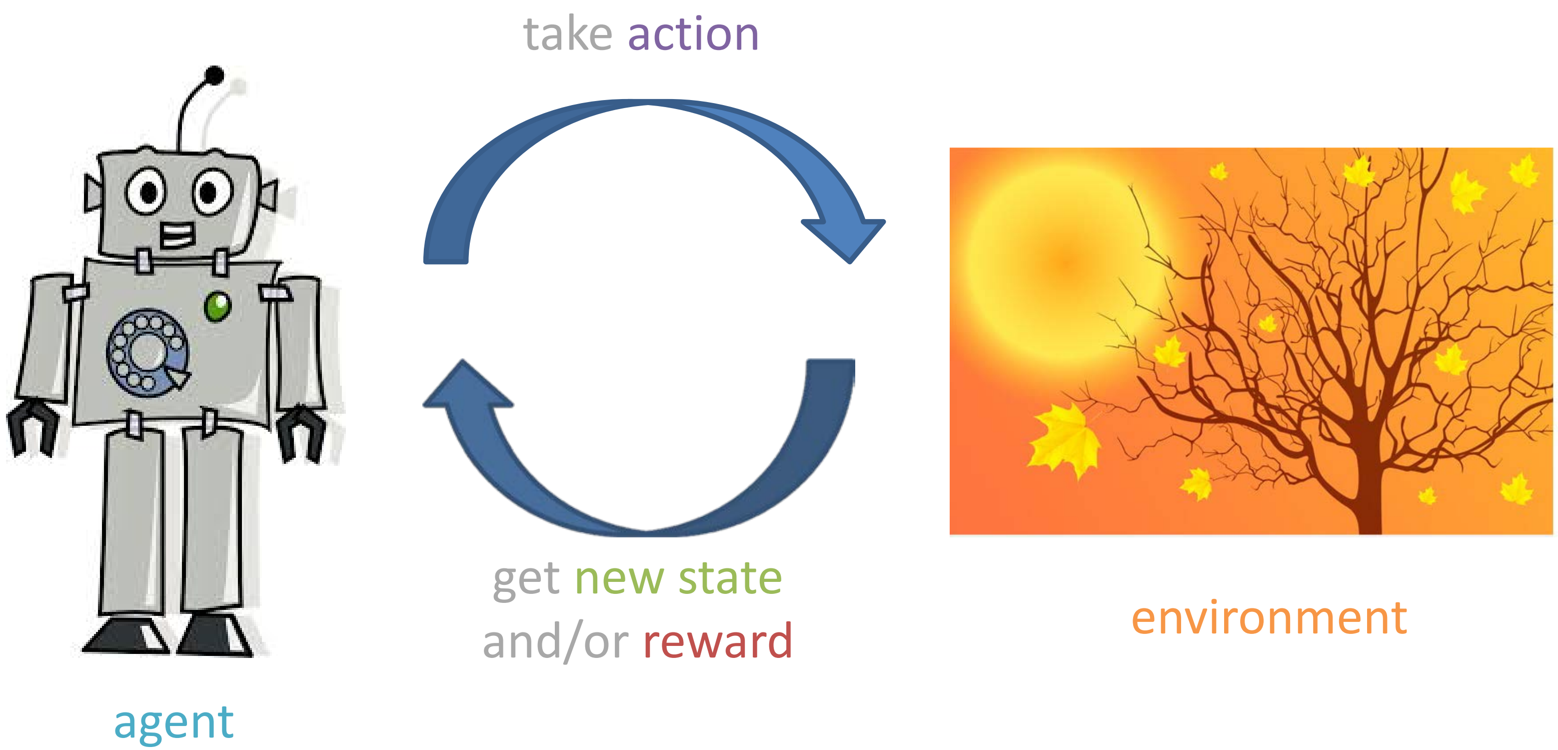
agent

environment

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Reinforcement Learning



Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Reinforcement Learning

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action

get new state
and/or reward

environment

agent

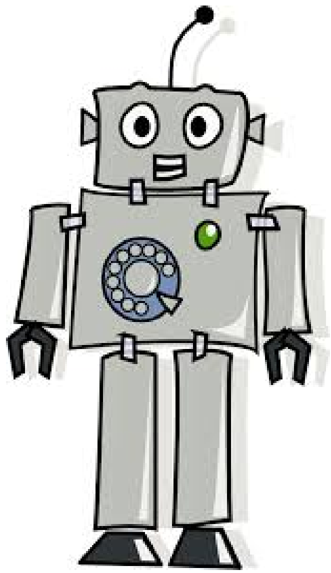
Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$

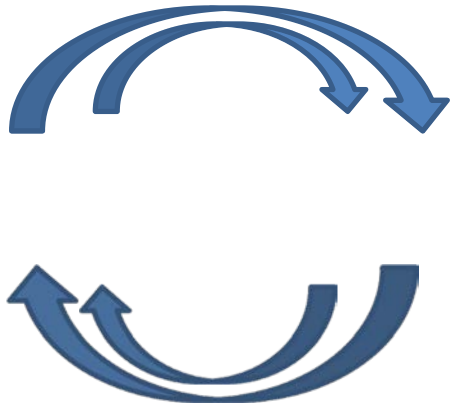
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action





get new state and/or reward

agent

environment

Markov Decision Process:

set of possible actions

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$$

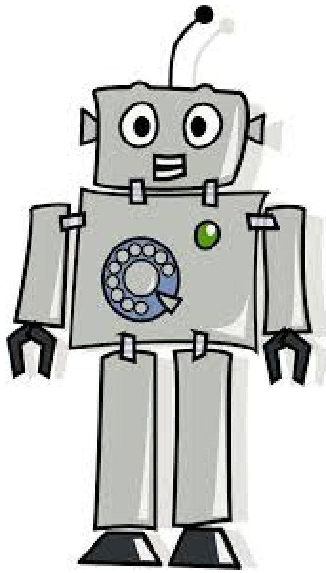
set of possible states

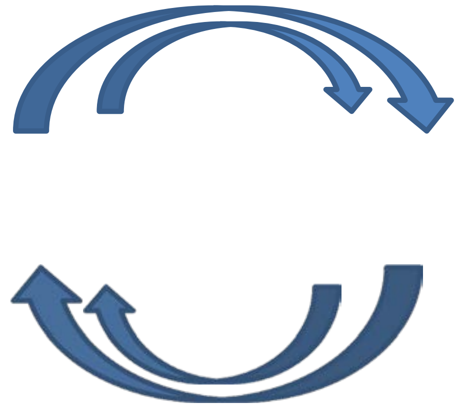
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action





get new state and/or reward

environment

agent

set of possible actions

Markov Decision Process:

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$$

set of possible states

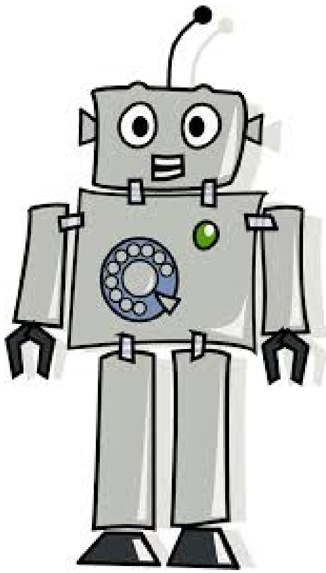
reward of (state, action) pairs

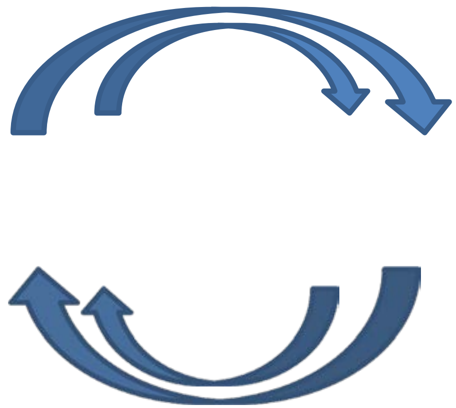
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process: Formalizing Reinforcement Learning

take action





get new state and/or reward

environment

agent

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$

set of possible states

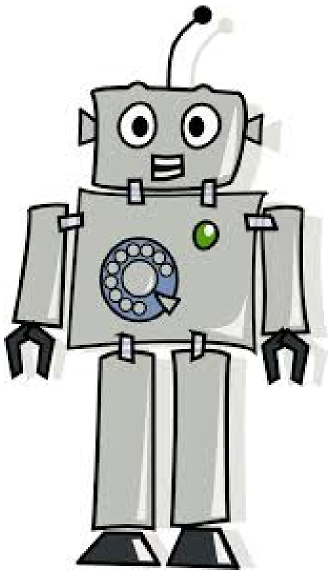
reward of (state, action) pairs

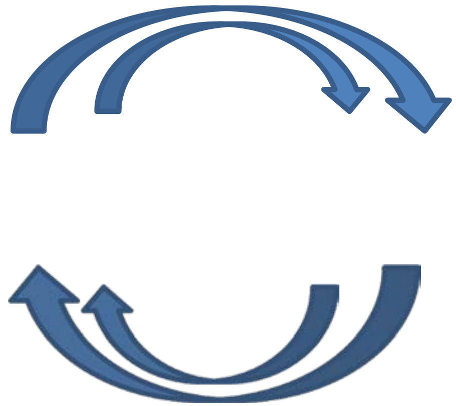
Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

Markov Decision Process:
Formalizing Reinforcement Learning

take action





get new state
and/or reward

environment

agent

Markov Decision Process:

set of possible actions

state-action transition distribution

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$$

set of possible states

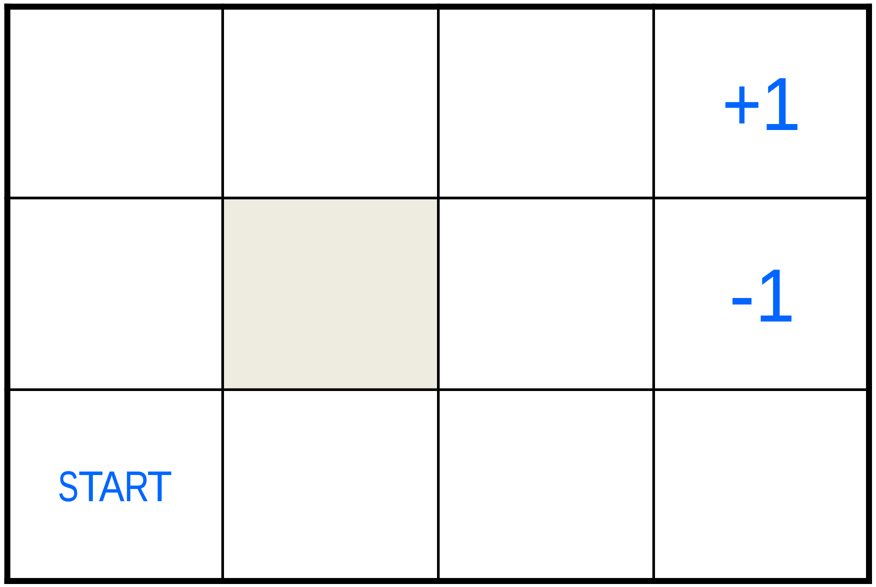
reward of (state, action) pairs

discount factor

Robot image: openclipart.org

https://static.vecteezy.com/system/resources/previews/000/090/451/original/four-seasons-landscape-illustrations-vector.jpg

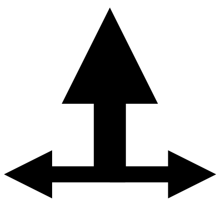
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

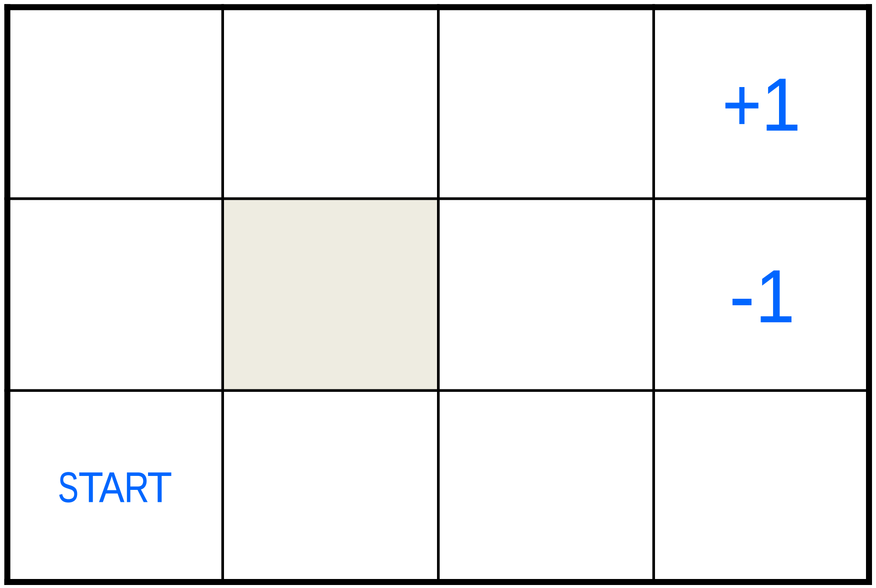


reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

Goal: what's the strategy to achieve the maximum reward?

Slide courtesy Peter Bodík

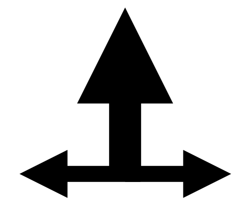
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

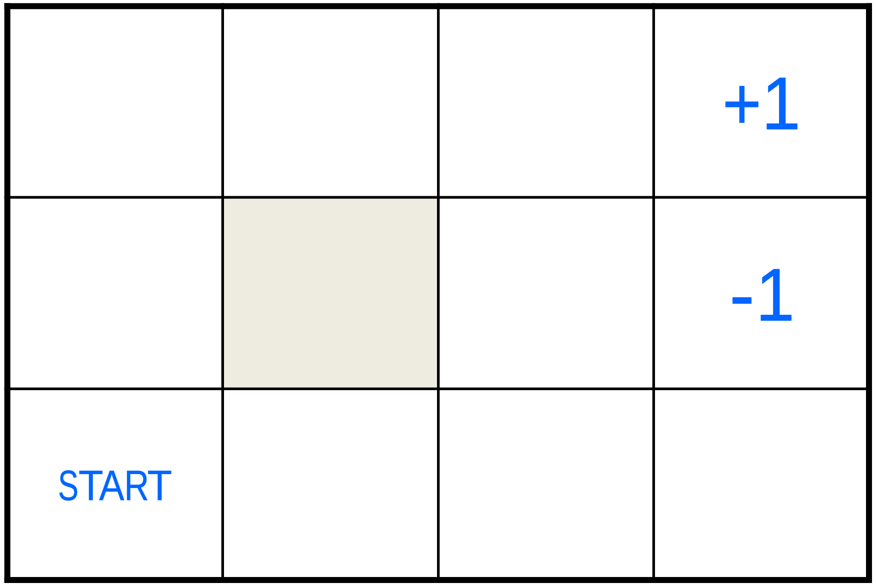


reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

states: current location
actions: where to go next
rewards

what is the solution? map each state to an action

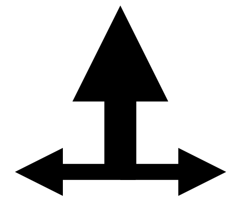
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT



reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

states: current location

actions: where to go next

rewards

a distribution over actions

what is the solution? map each state to ~~an action~~

Slide courtesy Peter Bodík

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
choose action a_t

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- π: state-action transition distribution
- γ: discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process:

set of possible actions

state-action transition distribution

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot|s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

Markov Decision Process: Formalizing Reinforcement Learning

set of possible actions

state-action transition distribution

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

set of possible states

reward of (state, action) pairs

discount factor

Start in initial state s_0
for t = 1 to …:
    choose action a_t
    "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
    get reward $r_t = \mathcal{R}(s_t, a_t)$

objective:

$$\max_{\pi} \sum_{t} \gamma^{t} r_t$$

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward of (state, action) pairs
- π: state-action transition distribution
- γ: discount factor

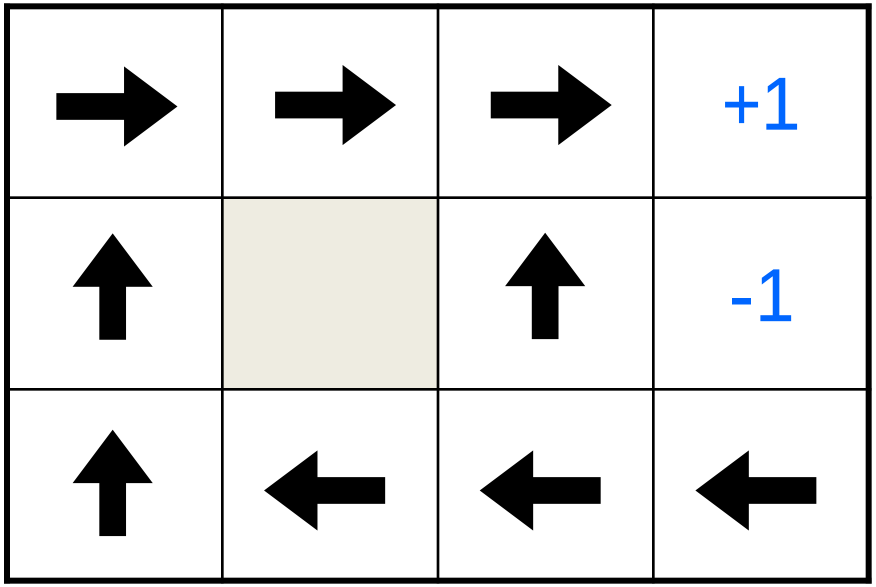
Start in initial state s_0
for t = 1 to …:
  choose action a_t
  "move" to next state $s_t \sim \pi(\cdot | s_{t-1}, a_t)$
  get reward $r_t = \mathcal{R}(s_t, a_t)$

objective:

$$\max_{\pi} \sum_t \gamma^t r_t$$

"solution" $\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}\left[\sum_t \gamma^t r_t \,; \pi\right]$

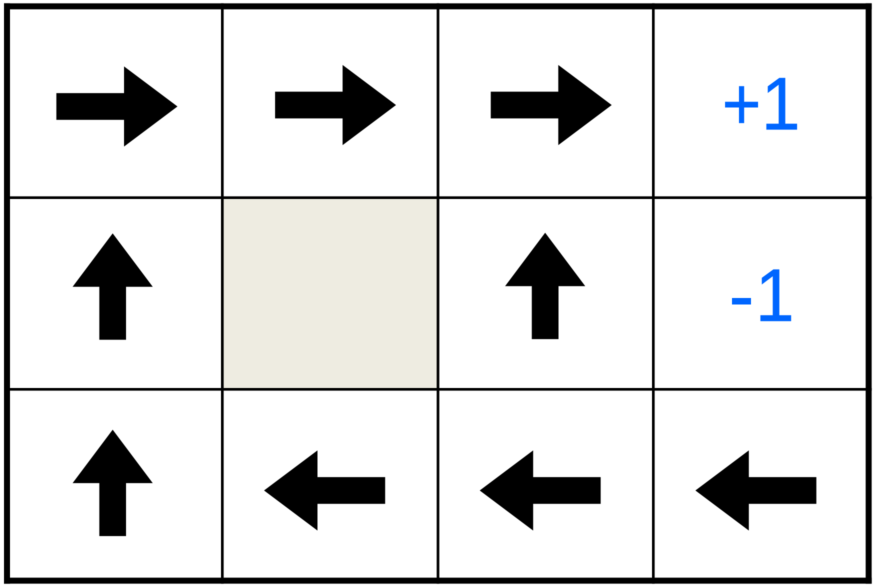
Step Rewards Change the Optimal Solution



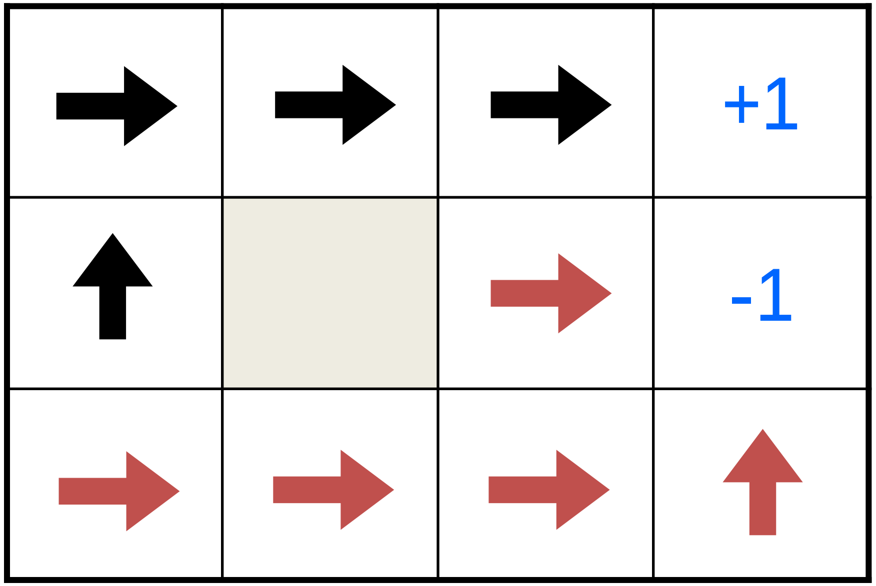
-0.04

Examples courtesy Peter Bodík

Step Rewards Change the Optimal Solution



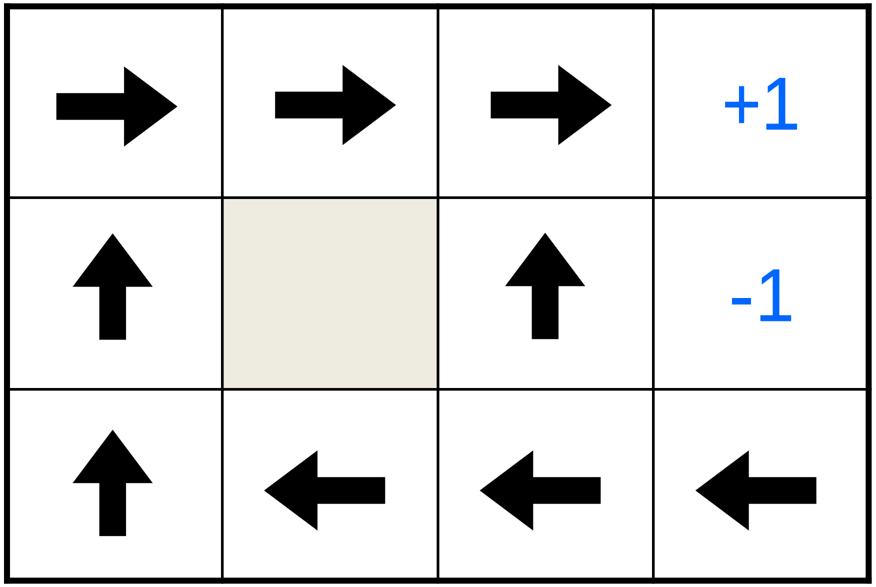
-0.04



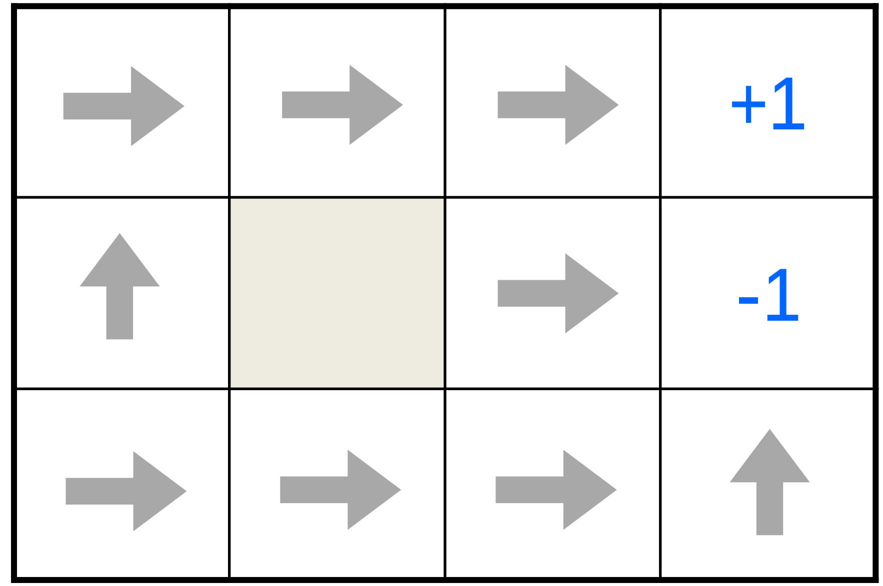
-2

Examples courtesy Peter Bodík

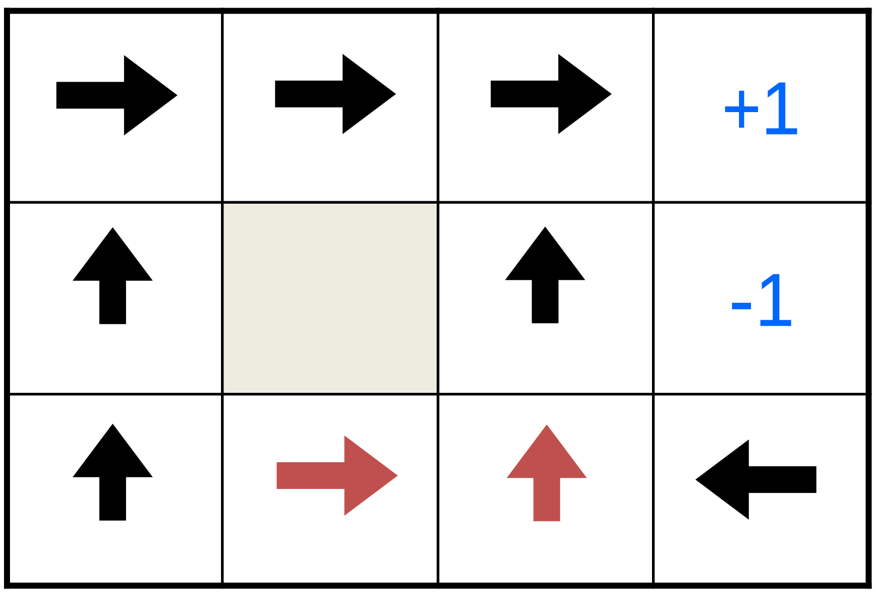
Step Rewards Change the Optimal Solution



-0.04



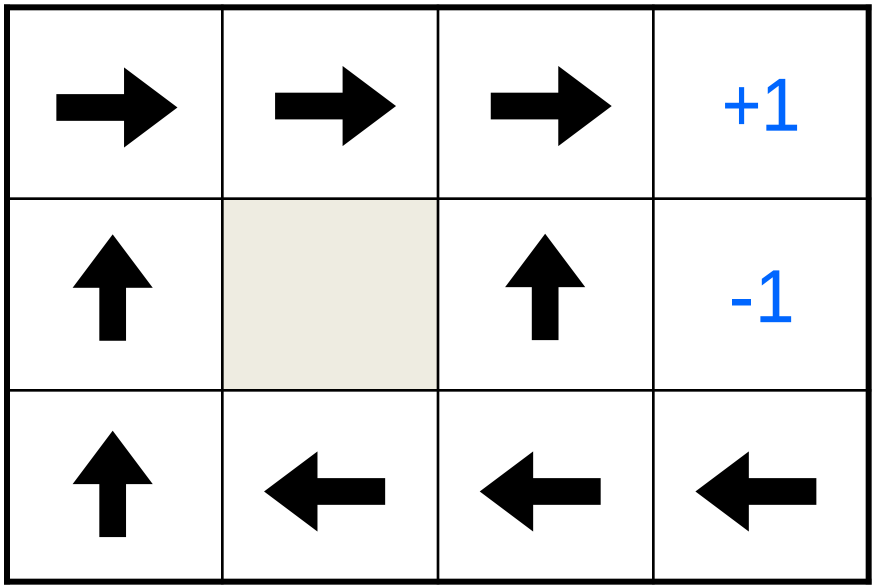
-2



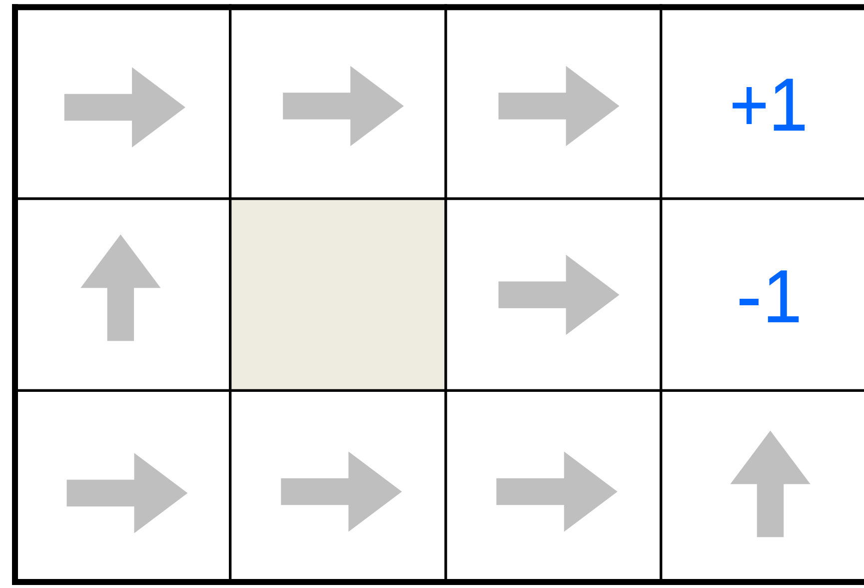
-0.1

Examples courtesy Peter Bodík

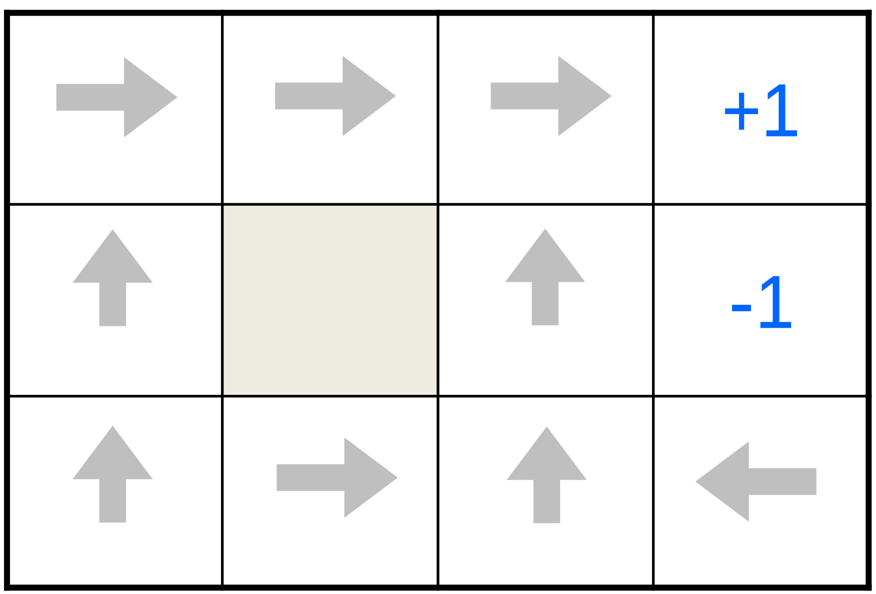
Step Rewards Change the Optimal Solution



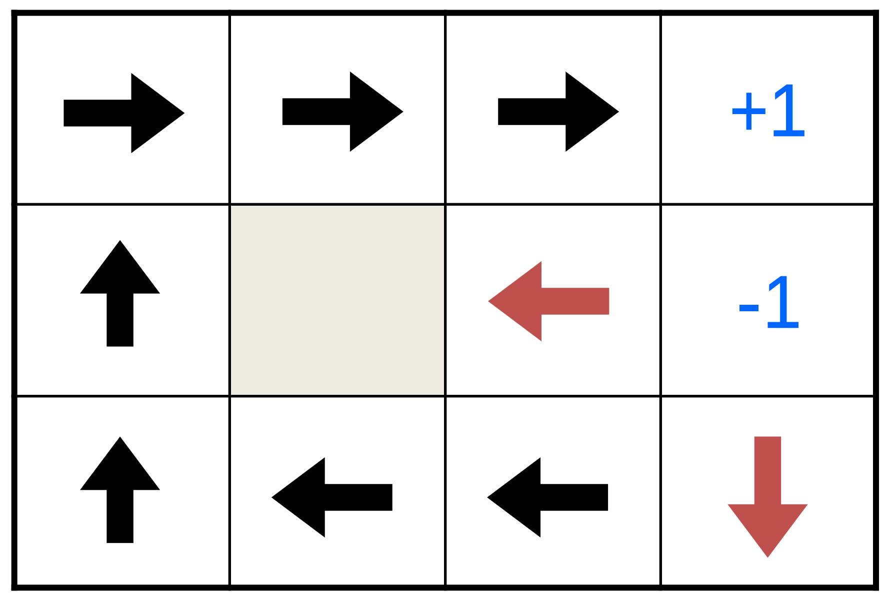
-0.04



-2



-0.1

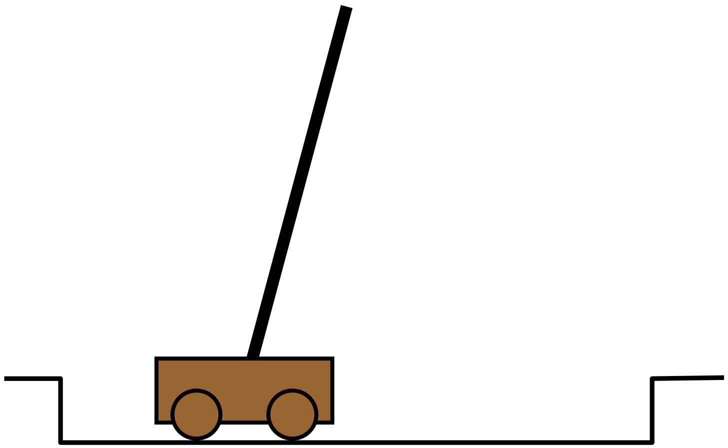


-0.01 Examples courtesy Peter Bodík

State Representation

Task: pole-balancing

state representation?



move car left/right to
keep the pole balanced

State Representation

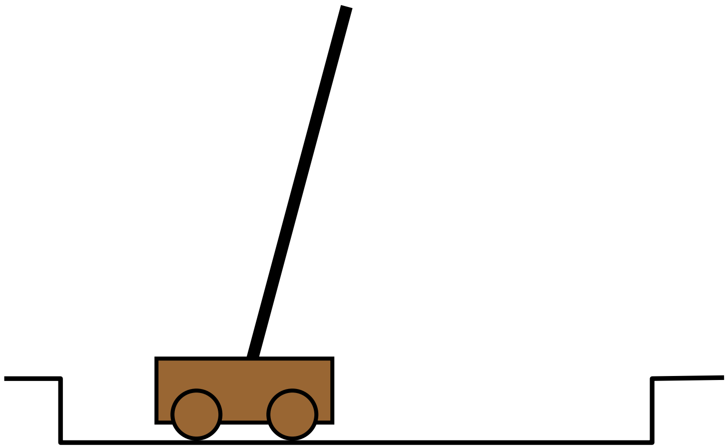
Task: pole-balancing

state representation

position and velocity of car

angle and angular velocity of pole

what about *Markov property*?



move car left/right to
keep the pole balanced

State Representation

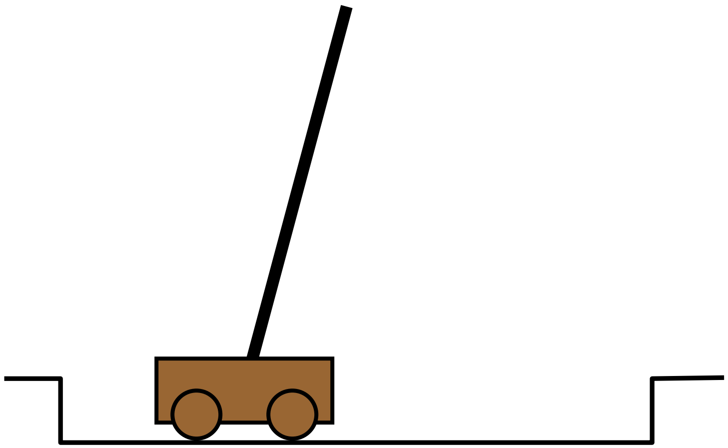
Task: pole-balancing

state representation

- position and velocity of car
- angle and angular velocity of pole

what about *Markov property*?

- would need more info
- noise in sensors, temperature, bending of pole



move car left/right to keep the pole balanced

Designing Rewards

robot in a maze

- episodic task, not discounted, +1 when out, 0 for each step

chess

- GOOD: +1 for winning, -1 losing
- BAD: +0.25 for taking opponent's pieces
  - high reward even when lose

Designing Rewards

robot in a maze

- episodic task, not discounted, +1 when out, 0 for each step

chess

- GOOD: +1 for winning, -1 losing
- BAD: +0.25 for taking opponent's pieces
  - high reward even when lose

rewards

- rewards indicate what we want to accomplish
- NOT how we want to accomplish it

Designing Rewards

robot in a maze

- episodic task, not discounted, +1 when out, 0 for each step

chess

- GOOD: +1 for winning, -1 losing
- BAD: +0.25 for taking opponent's pieces
  - high reward even when lose

rewards

- rewards indicate what we want to accomplish
- NOT how we want to accomplish it

shaping

- positive reward often very "far away"
- rewards for achieving subgoals (domain knowledge)
- also: adjust initial policy or initial value function

Slide courtesy/adapted: Peter Bodík

Value functions

state value function: $V_\pi(s)$

expected return when starting in s and following π

$$V_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \, ; s_0 = s, \pi\right]$$

Value functions

state value function: $V_\pi(s)$

expected return when starting in *s* and following π

$$V_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,; s_0 = s, \pi\right]$$

state-action value function: $Q_\pi(s,a)$

expected return when starting in *s*, performing *a,* and following π

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,; s_0 = s, a_0 = a, \pi\right]$$

Value functions

state value function: $V_\pi(s)$

expected return when starting in *s* and following π

$$V_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \, ; s_0 = s, \pi\right]$$

state-action value function: $Q_\pi(s,a)$

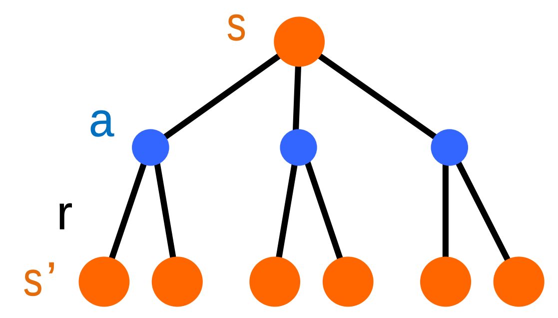
expected return when starting in *s*, performing *a,* and following π

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \, ; s_0 = s, a_0 = a, \pi\right]$$

useful for finding the optimal policy

can estimate from experience

pick the best action using $Q_\pi(s,a)$



Value functions

state value function: $V_\pi(s)$

expected return when starting in *s* and following π

$$V_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,;\, s_0 = s, \pi\right]$$

use V to define Q

state-action value function: $Q_\pi(s,a)$

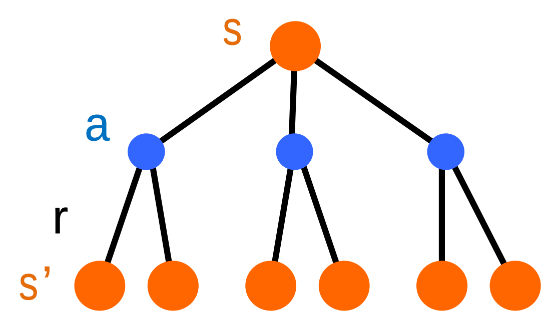
expected return when starting in *s*, performing *a*, and following π

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,;\, s_0 = s, a_0 = a, \pi\right]$$

useful for finding the optimal policy

can estimate from experience

pick the best action using $Q_\pi(s,a)$



Optimal value functions

definition

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,; s_0 = s, a_0 = a, \pi\right]$$

Optimal value functions

definition

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,; s_0 = s, a_0 = a, \pi\right]$$

satisfies

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \,; s_0 = s, a_0 = a, \pi\right]$$

Bellman equation

Optimal value functions

definition

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \,;\, s_0 = s, a_0 = a, \pi\right]$$

satisfies

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \,;\, s_0 = s, a_0 = a, \pi\right]$$

Bellman equation

idea: if you know the best action to take, then the
best strategy maximizes the overall expected reward

Overview: Learning Strategies

Dynamic Programming

Q-learning

Monte Carlo approaches

Dynamic programming

use value functions to structure the search for good policies

Dynamic programming

use value functions to structure the search for good policies

policy evaluation: compute V^{π} from π

policy improvement: improve π based on V^{π}

Dynamic programming

use value functions to structure the search for good policies

policy evaluation: compute V^{π} from π

policy improvement: improve π based on V^{π}

start with an arbitrary policy

repeat evaluation/improvement until convergence

Policy evaluation/improvement

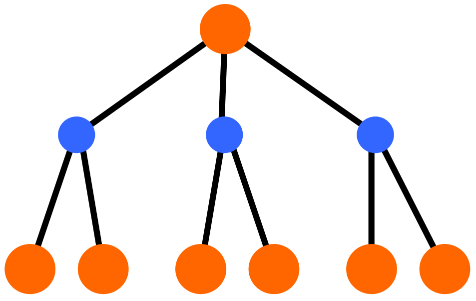
policy evaluation: $\pi \rightarrow V_\pi$

Bellman equations define a *system* of equations

could solve, but will use iterative version

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{k'} P^a_{ss'} \left[r^a_{ss'} + \gamma V_k(s') \right]$$

start with an arbitrary value function V_0, iterate until V_k converges



policy improvement: $V_\pi \rightarrow \pi'$

$$\pi'(s) = \arg\max_a Q^\pi(s,a)$$

$$= \arg\max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^\pi(s') \right]$$

π' either strictly better than π, or π' is optimal (if $\pi = \pi'$)

Q-learning

previous algorithms: on-policy algorithms

- start with a random policy, iteratively improve
- converge to optimal

Q-learning: off-policy

- use any policy to estimate Q

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

- Q directly approximates Q* (Bellman optimality equation)
- independent of the policy being followed
- only requirement: keep updating each (s,a) pair

Q-learning

previous algorithms: on-policy algorithms

- start with a random policy, iteratively improve
- converge to optimal

Q-learning: off-policy

- use any policy to estimate Q

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

- Q directly approximates Q* (Bellman optimality equation)
- independent of the policy being followed
- only requirement: keep updating each (s,a) pair

Deep/Neural Q-learning

$$Q(s, a; \theta) \approx Q^*(s, a)$$

neural network desired optimal solution

Deep/Neural Q-learning

$$Q(s, a; \theta) \approx Q^*(s, a)$$

neural network desired optimal solution

Q: What's a good loss function?

Deep/Neural Q-learning

$$Q(s, a; \theta) \approx Q^*(s, a)$$

neural network

desired optimal solution

Q: What's a good loss function?

A: Squared expectation loss

Monte Carlo policy evaluation

don't need full knowledge of environment (just (simulated) experience)

want to estimate $V^{\pi}(s)$

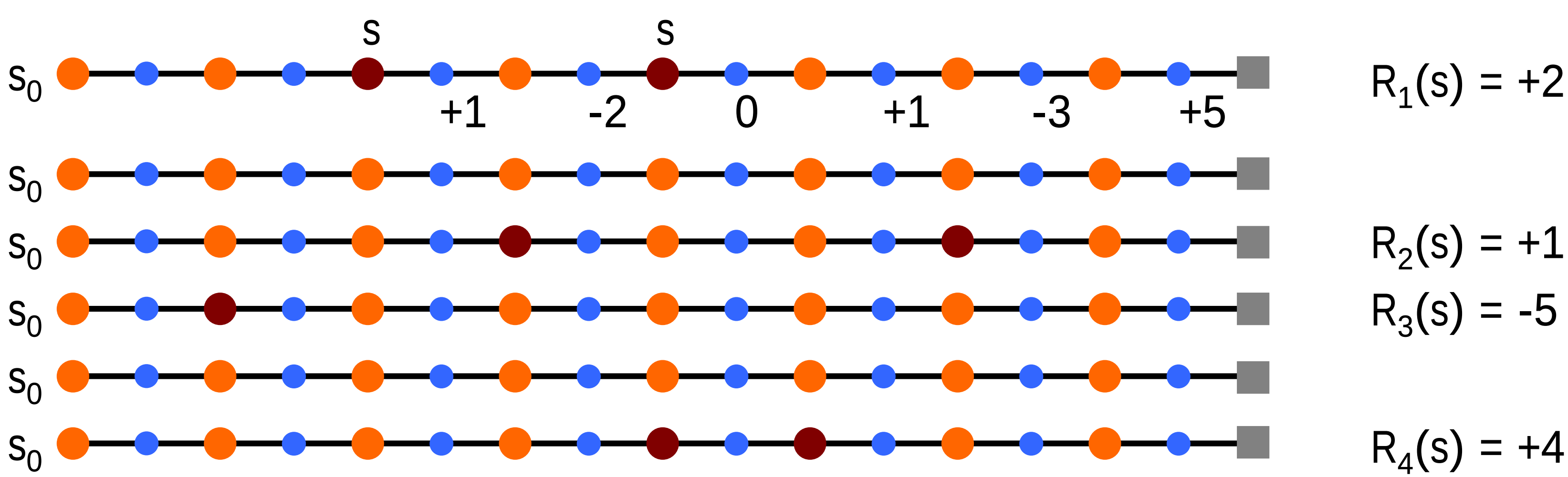
Slide courtesy/adapted: Peter Bodík

Monte Carlo policy evaluation

don't need full knowledge of environment (just (simulated) experience)

want to estimate $V^\pi(s)$

- expected return starting from s and following π
- estimate as average of observed returns in state s



$$V^\pi(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$$

Slide courtesy/adapted: Peter Bodík

Maintaining exploration

key ingredient of RL

deterministic/greedy policy won't explore all actions
- don't know anything about the environment at the beginning
- need to try all actions to find the optimal one

maintain exploration
- use *soft* policies instead: $\pi(s,a)>0$ (for all s,a)

ε-greedy policy
- with probability 1-ε perform the optimal/greedy action
- with probability ε perform a random action

- will keep exploring the environment
- slowly move it towards greedy policy: ε -> 0

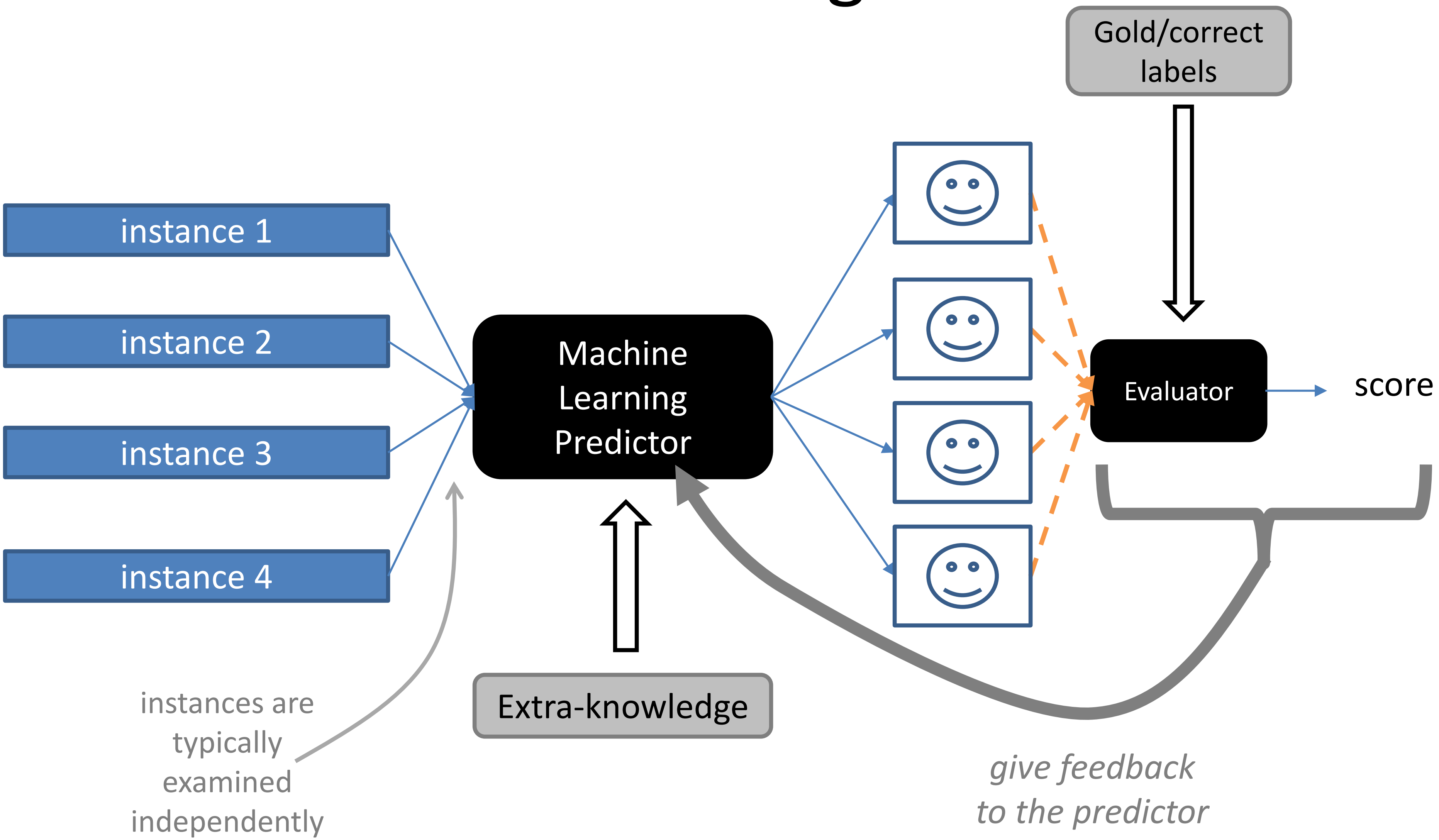
RL Slides Credit

https://people.eecs.berkeley.edu/~jordan/courses/294-fall09/lectures/reinforcement/slides.pptx

Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Machine Learning Framework: Learning

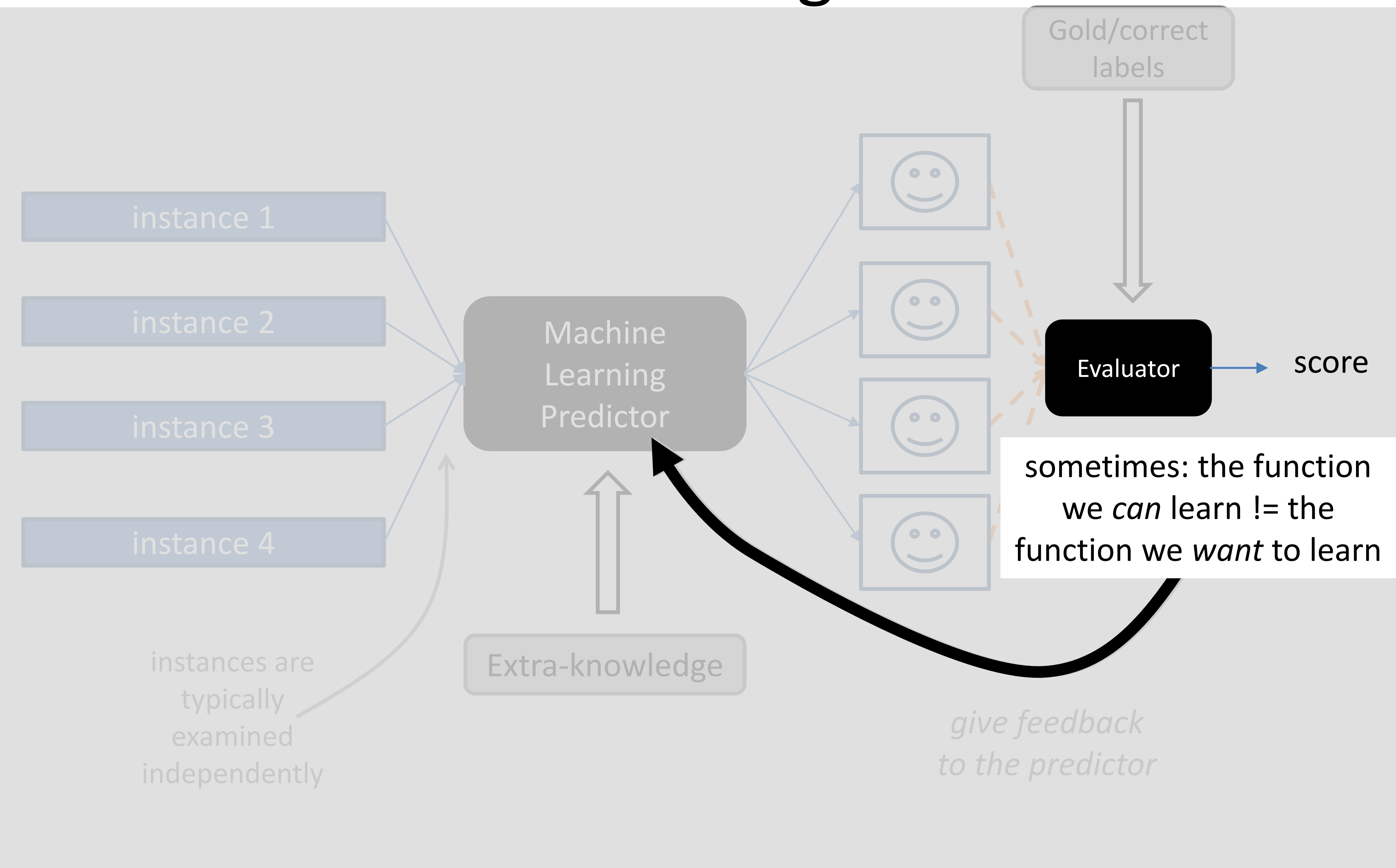


Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Learn different ways that success and progress can be measured in ML

Machine Learning Framework: Learning



Optimize Empirical Risk of Surrogate Loss

$$\underset{\text{h}}{\operatorname{argmin}} \sum_{i=1}^{N} \ell\left(y_i, h_\theta(\boldsymbol{x}_i)\right)$$

empirical risk minimization

$$\Downarrow$$

$$\nabla_\theta F = \sum_i \frac{\partial \ell\left(y_i, \hat{y} = h_\theta(\boldsymbol{x}_i)\right)}{\partial \hat{y}} \nabla_\theta h_\theta(\boldsymbol{x_i})$$

approximate loss in a computable way

Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Learn different ways that success and progress can be measured in ML

Relate to statistics, AI [671], and specialized areas (e.g., NLP [673] and CV [691])

Implement ML programs

Course Goals

Be introduced to some of the core problems and solutions of ML (big picture)

Learn different ways that success and progress can be measured in ML

Relate to statistics, AI [671], and specialized areas (e.g., NLP [673] and CV [691])

Implement ML programs

Read and analyze research papers

Practice your (written) communication skills

Remember from the first day: A Terminology Buffet

the *task*	the *data*	the *approach*
Classification	Fully-supervised	Probabilistic, Neural
Regression	Semi-supervised	Generative, Memory-based
Clustering	Un-supervised	Conditional, Exemplar
		Spectral, …

Classification

Regression

Clustering

*the **task**: what* kind *of problem are you solving?*

Fully-supervised

Semi-supervised

Un-supervised

*the **data**: amount of human input/number of labeled examples*

Probabilistic

Neural

Generative

Memory-based

Conditional

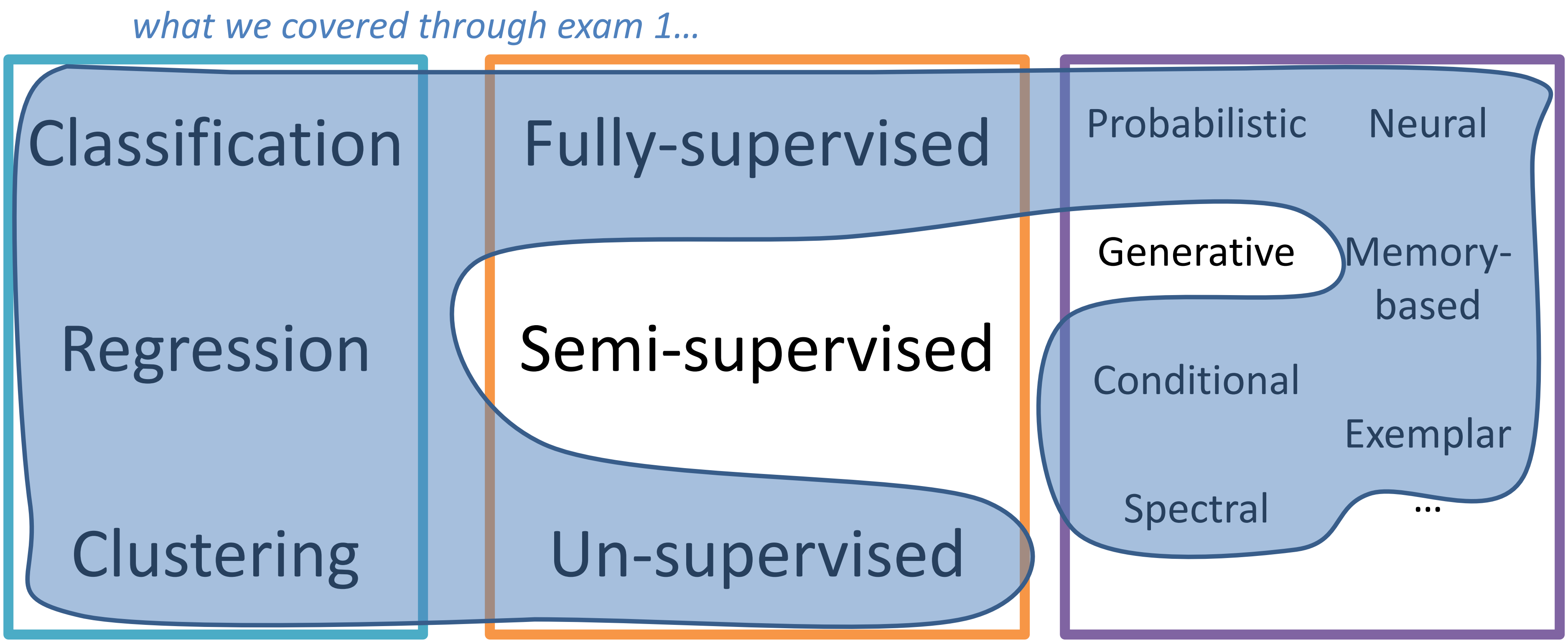
Exemplar

Spectral

…

*the **approach**: how any data are being used*

A Terminology Buffet



*the **task**: what kind of problem are you solving?*

*the **data**: amount of human input/number of labeled examples*

*the **approach**: how any data are being used*

Course Overview (Part 1)

Basics of Probability

- Requirements to be a distribution (“proportional to”, $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps”)
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

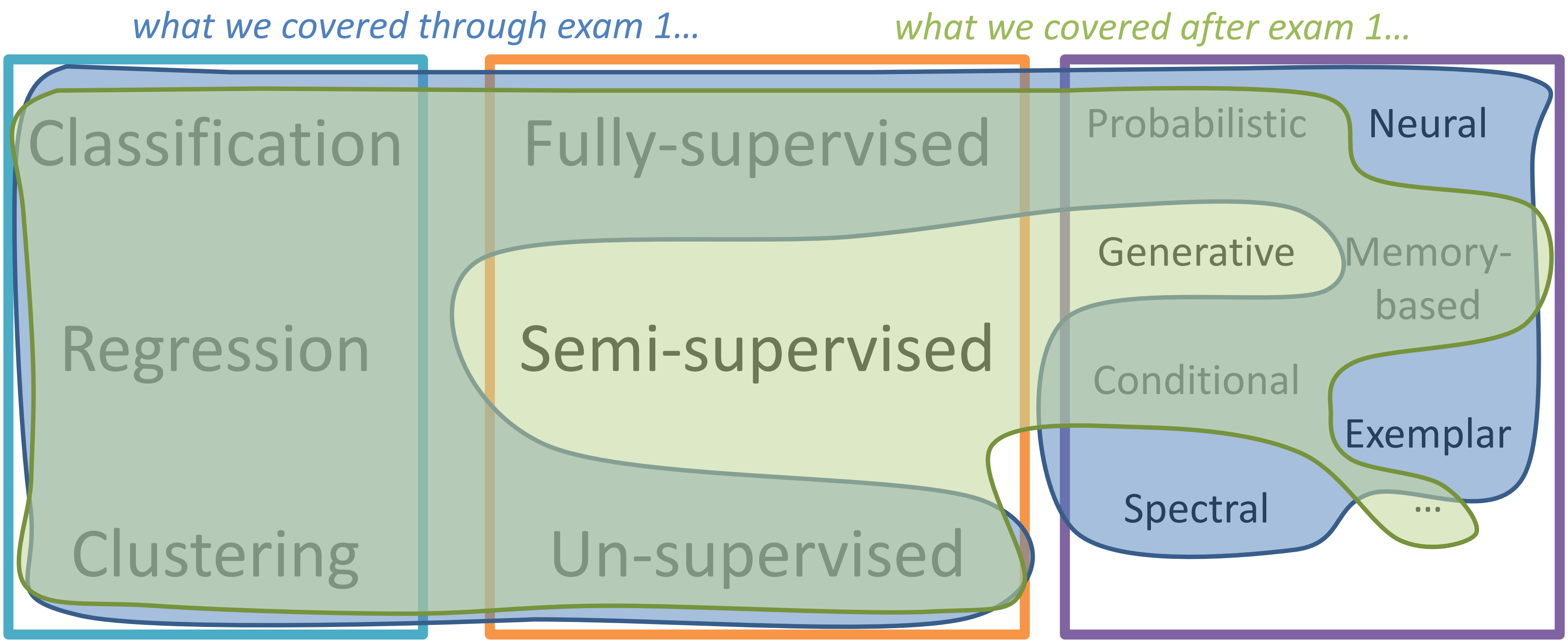
Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

A Terminology Buffet



*the **task**: what kind of problem are you solving?*

*the **data**: amount of human input/number of labeled examples*

*the **approach**: how any data are being used*

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution (“proportional to”, $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

leveraging large, unannotated data

Tasks (High Level)

- Data set splits: training v
- Classification: Posterior
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps”)
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training v
- Classification: Posterior
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

leveraging large, unannotated data

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Course Overview (Part 2)

adding structure

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs
- Classification: Posterior decoding
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models

- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn't)

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models

- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn't)

Distributions as Graphs

- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models

- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn't)

Distributions as Graphs

- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference

- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", ∝)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models

- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn't)

Distributions as Graphs

- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference

- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods

Structured Prediction

- General formulation
- Porting existing approaches: non-separability, slack, and sub-gradients
- Learning vs. inference
- General inference technique: ILP

Course Overview (Part 2)

Basics of Probability

- Requirements to be a distribution ("proportional to", $\propto$)
- Definitions of conditional probability, joint probability, and independence
- Bayes rule, (probability) chain rule
- Expectation (of a random variable & function)

Empirical Risk Minimization

- Gradient Descent
- Loss Functions: what is it, what does it measure, and what are some computational difficulties with them?
- Regularization: what is it, how does it work, and why might you want it?

Tasks (High Level)

- Data set splits: training vs. dev vs. test
- Classification: Posterior decoding/MAP classifier
- Classification evaluations: accuracy, precision, recall, and F scores
- Regression (vs. classification)
- Comparing supervised vs. Unsupervised Learning and their tradeoffs: why might you want to use one vs. the other, and what are some potential issues?
- Clustering: high-level goal/task, K-means as an example
- Tradeoffs among clustering evaluations

Linear Models

- Basic form of a linear model (classification or regression)
- Perceptron (simple vs. other variants, like averaged or voted)
- When you should use perceptron (what are its assumptions?)
- Perceptron as SGD

Maximum Entropy Models

- Meanings of feature functions and weights
- How to learn the weights: gradient descent
- Meaning of the maxent gradient

Neural Networks

- Relation to linear models and maxent
- Types (feedforward, CNN, RNN)
- Learning representations (e.g., "feature maps")
- What is a convolution (e.g., 1D vs 2D, high-level notions of why you might want to change padding or the width)
- How to learn: gradient descent, backprop
- Common activation functions
- Neural network regularization

Dimensionality Reduction

- What is the basic task & goal in dimensionality reduction?
- Dimensionality reduction tradeoffs: why might you want to, and what are some potential issues?
- Linear Discriminant Analysis vs. Principal Component Analysis: what are they trying to do, how are they similar, how do they differ?

Kernel Methods & SVMs

- Feature expansion and kernels
- Two views: maximizing a separating hyperplane margin vs. loss optimization (norm minimization)
- Non-separability & slack
- Sub-gradients

Latent Variable Models

- Meaning: what's a latent variable?
- General problem: latent variables are (often) not labeled (or difficult)
- General algorithm: expectation maximization
- Problem EM optimizes (and what it doesn't)

Distributions as Graphs

- Directed graphical models: Bayesian networks, HMMs, MEMMs
- Undirected graphical models: CRFs, MRFs
- Message passing: Viterbi (max-product), Forward-backward (sum-product)

Bayesian Inference

- Posterior inference (priors all around)
- Reliance on fundamental statistical distributions
- Variational inference
- Sampling methods

Structured Prediction

- General formulation
- Porting existing approaches: non-separability, slack, and sub-gradients
- Learning vs. inference
- General inference technique: ILP

Ensembles & RL

- Combining approaches