Cross Validation, Reinforcement Learning

Lecture 27:

Intro to Reinforcement Learning

Cross Validation

ECE/CS 498 DS

Professor Ravi K. Iyer

Department of Electrical and Computer Engineering

University of Illinois

Announcements

- MP 2 final grades and MP 3 checkpoint 1 grades have been released
- MP 3 final submission due **tonight 4/29 @ 11:59 PM** via Compass
- HW 5 released, due **Mon 5/4 @ 11:59 PM** via Compass
  - Covers SVM, neural networks, and random forest
- Discussion section on **Fri 5/1** will be additional practice with neural networks

Final Exam Logistics

Final Exam Logistics

- Final Exam will **be Friday 5/15 from 8-11:10 AM**
- Exam length:
  - You will have 10 minutes of dedicated reading time from 8-8:10 AM
  - After that, exam is designed to take ~2.5 hours (you have 3 hours)
  - You need to scan and submit your solutions by 11:10 AM on Compass
- You are allowed 3 physical, double-sided, 8.5x11 cheat sheets (can be typed or handwritten)
- **No calculators** or other electronic devices are allowed
- **No reference** materials/books are allowed

Final Exam Logistics

- Content:
  - **All course material is fair game**
  - **Material from after midterm exam will be emphasized**:
    - HMMs and Forward-Backward Algorithm
    - Factor Graphs and Belief Propagation
    - SVM
    - Neural Networks
    - Random Forest
    - Cross Validation, etc.
  - **Earlier course material may be included**:
    - Basic Probability
    - Naïve Bayes and Bayesian Networks
    - Clustering (K-means, GMM, Hierarchical)
    - Linear/Logistic Regression
    - PCA, etc.

Final Exam Logistics

- Exam will be released as PDF via Compass, and you will have until 11:10 AM to (i) complete it and (ii) scan and submit your solutions on Compass
  - **Submission format** – preferably .pdf, but .jpg/.jpeg will be accepted as well.
  - **Submission quality** – Make sure to use the highest-quality files available to you. You will be responsible for ensuring that all scanned files are clearly legible/readable by course staff. We cannot grade what we cannot read…
  - **Strict deadline** – You must scan and submit your solutions by 11:10 AM
  - **Smartphone Scanning** – If needed, you may use your smartphone to scan your exam after you have finished writing
    - Writing after using smartphone will be considered cheating, and you are not allowed to use your smartphone at any other point during the exam
    - Recommended smartphone apps for scanning: Genius Scan, CamScanner

Final Exam Logistics

- Exam will be proctored remotely using **Proctorio**. Requirements:
  - Install **Google Chrome**
  - Install **Proctorio extension** for Chrome (https://getproctorio.com/)
  - Have your **iCard** (or some other official ID) available to show the camera at the beginning of the exam
  - Have **~20 blank sheets** of white printer paper to write on during the exam. You may not need all 20, but it's better to have extra
  - Have a **webcam** available during the exam (laptop webcam is fine)
    - All students must have a webcam to use during the exam for us to verify academic integrity
  - Have a **microphone** available during the exam (laptop microphone is fine)
- Webcam should be setup to show you working at a desk/table

Cross Validation

Validation: Binary Classification Performance

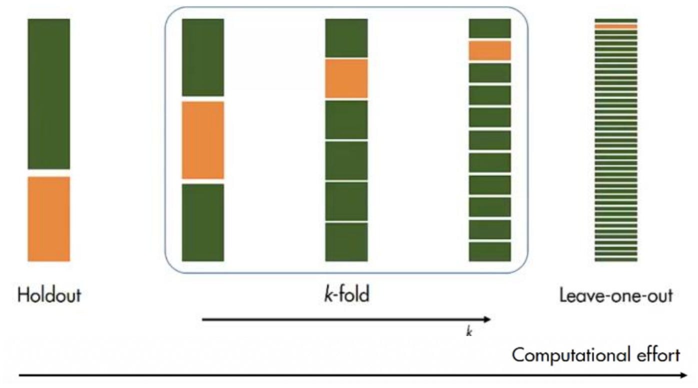
You are solving a Binary Classification Problem, Predict if a person has cancer (1) or not (0). You have learned a model using your training dataset.
How do you test your model?

Explanation:

1) **Holdout**: Randomly divide dataset into 2 parts, learn model on training data and validate on the test data.
2) **K-fold**: Divide data into K equal parts, run the following K-times: (hold out one of the K parts of the dataset, train on the remaining, test on the holdout). Every data-sample is part of the training data K-1 times and part of the test data 1 time.
3) **Leave-one-out**: For every data sample, leave it out, train the model on the remaining data and predict for the left-out data sample.
4) **Monte Carlo Cross Validation**: In each of multiple trials, randomly divide the dataset into training and testing sets (of prespecified sizes). Always retrain the model on the trial's training set and validate on the trial's testing set.

Cross Validation: For a given training iteration, divide the dataset into 2 parts:
- training set (for training the model)
- test set (for validating the model)

Different ways of doing this:
(1) Holdout
(2) K-fold
(3) Leave-one-out
(4) Monte Carlo

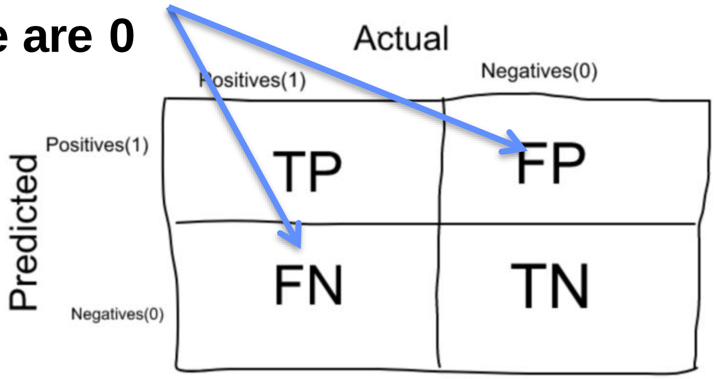


green – training data
orange - test data

What does it mean to **validate** on the test data?

Ideal if these are 0

Confusion Matrix



Depending on the **cost of making a mistake**, minimize the number of FN or FP

Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{F1-score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

Harmonic mean of Precision and Recall (when large disparity between the 2 values, the score is closer to the smaller value)

Common issues

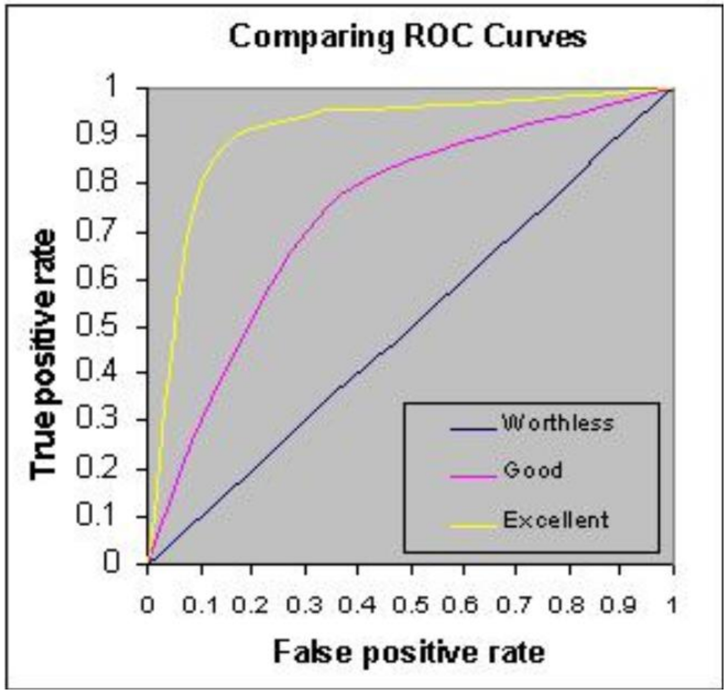
Precision = 1 if FP=0, but FN may be high
Sensitivity = 1 if predict everything as positive
Specificity = 1 if predict everything as negative
How to strike a **balance**? Use F1-score!

Why not Always Use Accuracy?

- Suppose you need to perform binary classification, where the classes are either positive or negative
- If the class distribution is imbalanced, another metric called AUC may be a better performance evaluation criterion
  - Let's say 90% of the test labels are positive
  - A very simple model that only predicts positive labels (and doesn't use the training data at all) would achieve 90% accuracy
    - If the training data is also just as skewed, then even a more complex model may end up predicting positive most of the time
  - This high accuracy is misleading since it depends solely on the distribution of the test dataset
  - By capturing results after varying the decision threshold τ in the range $[0, 1]$, AUC measures discriminative performance of the model
    - Thus, AUC is generally preferred for evaluating binary classifiers with imbalanced datasets
    - The very simple model would report a poor AUC score

ROC and AUC

Receiver Operating Characteristic curve: A plot of **False Positive Rate** Vs **True Positive Rate**

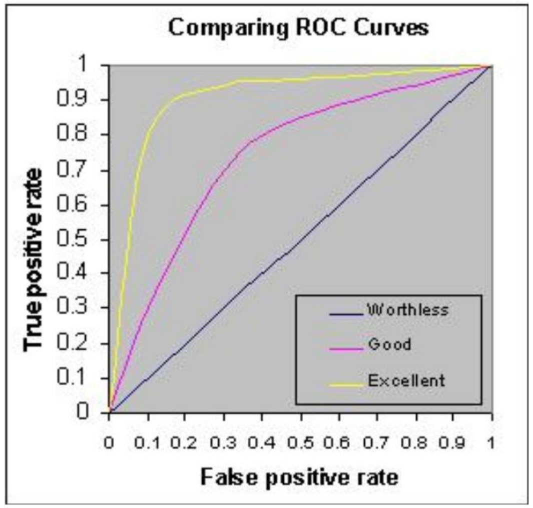


$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

- Typically, a binary classifier has a decision boundary of $\tau = 0.5$
  - If $P(sample\ has\ positive\ label) \geq \tau$, then predict a positive label for the sample. Else, predict a negative label.
- An ROC curve is traced by tracking (FPR, TPR) for a variety of classification thresholds τ in the range $[0, 1]$
  - All ROC curves pass through the points (1,1) and (0,0)
  - $\tau = 0 \Rightarrow$ Only positive labels predicted $\Rightarrow$ (FPR, TPR) = (1,1)
  - $\tau = 1 \Rightarrow$ Only negative labels predicted $\Rightarrow$ (FPR, TPR) = (0,0)

ROC and AUC

Receiver Operating Characteristic curve: A plot of **False Positive Rate** Vs **True Positive Rate**



$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

An ML model has parameters and for different values of the parameters, the model gives different FPR and TPR.

An **ROC (Receiver Operating Characteristic) curve** can be plotted for these different settings. A **good setting** of the parameters i.e. a good classifier, would have a **high TPR and low FPR**.

Area Under Curve (AUC) in [0,1]:
A good classifier would have the largest area under the ROC curve.
The random predictor would have an ROC of 0.5 (the curve of the 'worthless' model in fig.)

AUC value **interpretation** –
If a pair of data samples are drawn independently, one each from the positive and negative sets, AUC gives the probability that the classifier will predict a lower score for the negative sample as compared to the positive sample

Be Careful with Interpreting Metrics…

- When comparing accuracies/AUCs between classifiers, make sure that they are operating on the same (or sufficiently similar) datasets
  - Otherwise, the implicit bias in the datasets can affect prediction outcomes

- If the AUC scores for two classifiers are similar, the individual class probabilities reported for samples by each classifier can be examined to determine which classifier is more discriminative

- AUC is only used with binary classifications
  - For multinomial classifications with AUC, a “one vs rest” scheme may be needed

Reinforcement Learning

Adapted from slides by Sergey Levine
Dept of eecs UC Berkeley

From Markov Chains to RL: Markov Model

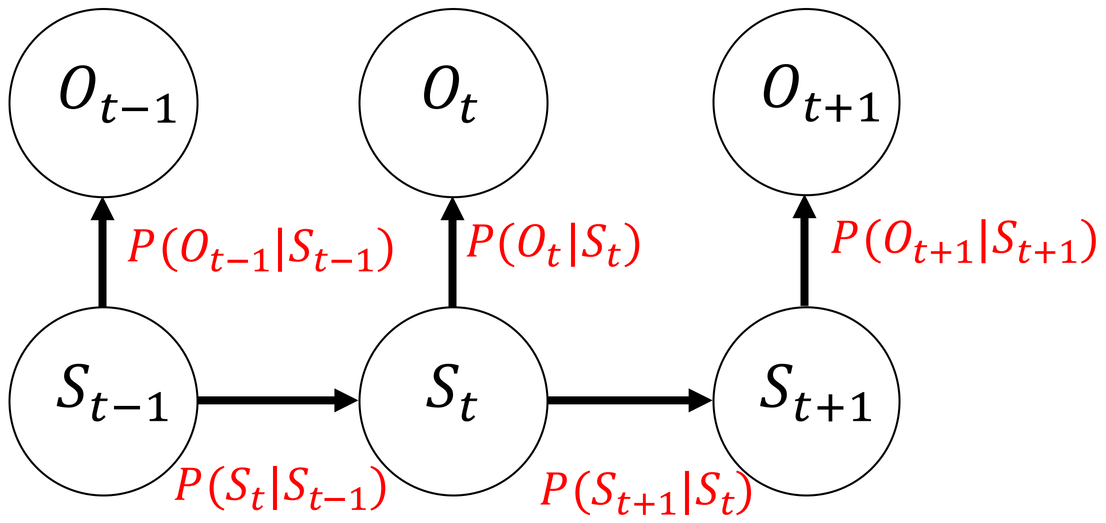
We can use causal relationships to specify conditional probabilities (like in Bayesian networks)

$$\begin{tikzcd} S_{t-1} \arrow[r, "P(S_t|S_{t-1})"'] & S_t \arrow[r, "P(S_{t+1}|S_t)"'] & S_{t+1} \end{tikzcd}$$

S_t is the state at time step t

Goal: Determine state using $P(S_{t+1}|S_t)$

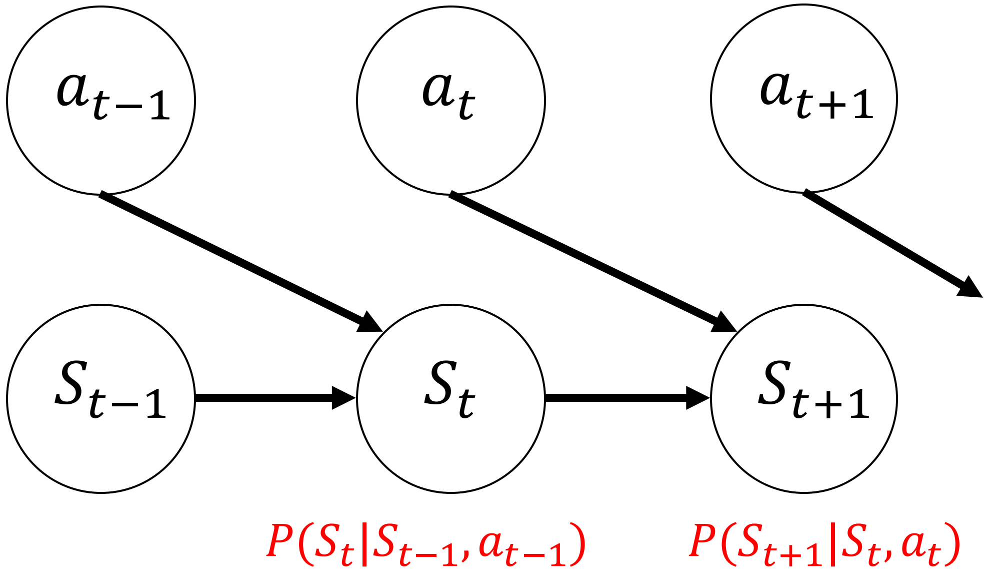
From Markov Chains to RL: HMM



O_t are observations at time step t

Goal: Determine hidden states using $P(S_{t+1}|S_t), P(O_{t+1}|S_{t+1})$

From Markov Chains to RL: Markov Decision Process



a_t is the action taken at time step t

Goal: Identify a **policy** $\pi_\theta(a_t|S_t)$, which is the action a_t to take at state S_t

In an MDP, we **fully observe** S_t

From Markov Chains to RL: Partially Observed Markov Decision Process

$$\begin{tikzcd}
O_{t-1} \arrow[r, "P(a_{t-1}|O_{t-1})"] & a_{t-1} \arrow[dr] & O_t \arrow[r, "P(a_t|O_t)"] & a_t \arrow[dr] & O_{t+1} \arrow[r, "P(a_{t+1}|O_{t+1})"] & a_{t+1} \\
S_{t-1} \arrow[u, "P(O_{t-1}|S_{t-1})"'] \arrow[rr, "P(S_t|S_{t-1},a_{t-1})"'] & & S_t \arrow[u, "P(O_t|S_t)"'] \arrow[rr, "P(S_{t+1}|S_t,a_t)"'] & & S_{t+1} \arrow[u, "P(O_{t+1}|S_{t+1})"'] &
\end{tikzcd}$$

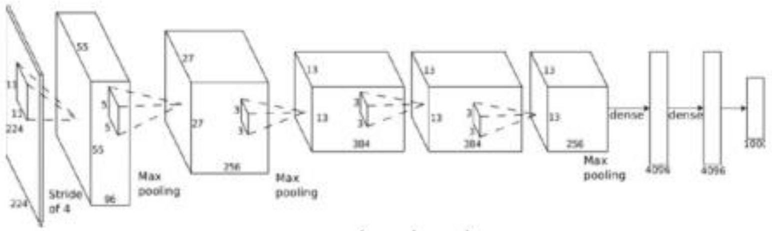
O_t are observations at time step t

a_t is the action taken at time step t

Goal: Identify a **policy** $\pi_\theta(a_t|O_t)$, which is the action a_t to take at state S_t

In an POMDP, we **observe** O_t (S_t is hidden)

Terminology & Notation



$$\begin{tikzcd}
\mathbf{o}_t & \pi_\theta(\mathbf{a}_t|\mathbf{o}_t) & \mathbf{a}_t \arrow[ll, bend left]
\end{tikzcd}$$

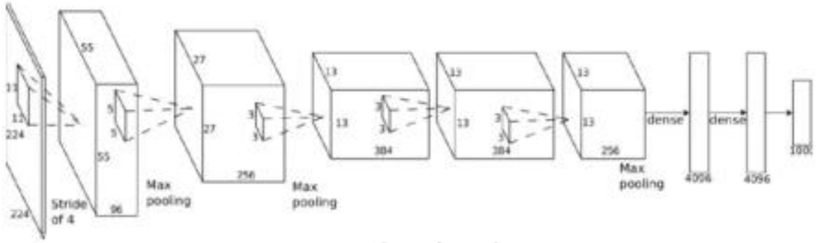
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

$$\begin{tikzcd}
\mathbf{o}_1 \arrow[r, "\pi_\theta"] & \mathbf{a}_1 \arrow[dr] & \mathbf{o}_2 \arrow[r, "\pi_\theta"] & \mathbf{a}_2 \arrow[dr] & \mathbf{o}_3 \arrow[r, "\pi_\theta"] & \mathbf{a}_3 \\
\mathbf{s}_1 \arrow[u] \arrow[rr, "p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)"'] & & \mathbf{s}_2 \arrow[u] \arrow[rr, "p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)"'] & & \mathbf{s}_3 \arrow[u] & \text{Markov property independent of } \mathbf{s}_{t-1}
\end{tikzcd}$$

Reward functions

$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define Markov decision process

high reward

low reward

Definitions

Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$

$\mathcal{S}$ – state space — states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space — actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a tensor!)

let $\mu_{t,j} = p(s_t = j)$

let $\xi_{t,k} = p(a_t = k)$

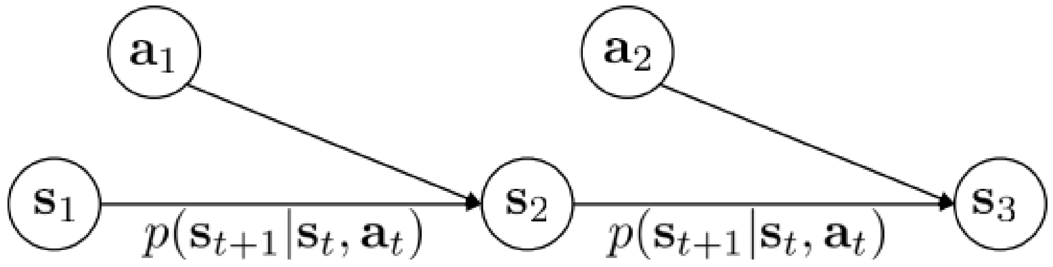
let $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$

$$\mu_{t+1,i} = \sum_{j,k} \mathcal{T}_{i,j,k} \mu_{t,j} \xi_{t,k}$$

$$= \sum_{j,k} \begin{matrix} p(s_{t+1} = i | s_t = j, a_t = k) \\ p(s_t = j) p(a_t = k) \end{matrix}$$

$$= \sum_{j,k} p(s_{t+1} = i, s_t = j, a_t = k)$$

All possible ways of getting $s_{t+1} = i$


$$\begin{tikzcd} \mathbf{a}_1 \arrow[dr] & & \mathbf{a}_2 \arrow[dr] & \\ \mathbf{s}_1 \arrow[r, "p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)"'] & \mathbf{s}_2 \arrow[rr, "p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)"'] & & \mathbf{s}_3 \end{tikzcd}$$

Finite horizon case: state-action marginal

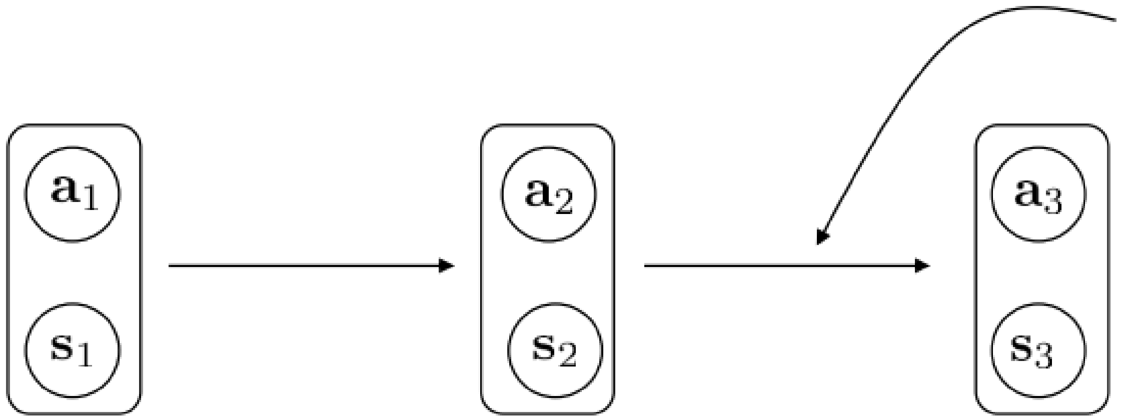
$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Expected cumulative reward under probability distribution $p_\theta(\mathcal{T})$

$$= \arg\max_\theta \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)]$$

$p_\theta(\mathbf{s}_t,\mathbf{a}_t)$ state-action marginal

$$p((\mathbf{s}_{t+1},\mathbf{a}_{t+1})|(\mathbf{s}_t,\mathbf{a}_t)) = p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi_\theta(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$$



Transition tensor and policy

Infinite horizon case: stationary distribution

$$\theta^\star = \arg\max_\theta \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)]$$

what if $T=\infty$?

does $p(\mathbf{s}_t,\mathbf{a}_t)$ converge to a *stationary* distribution?

$\mu = \mathcal{T}\mu$

stationary = the same before and after transition

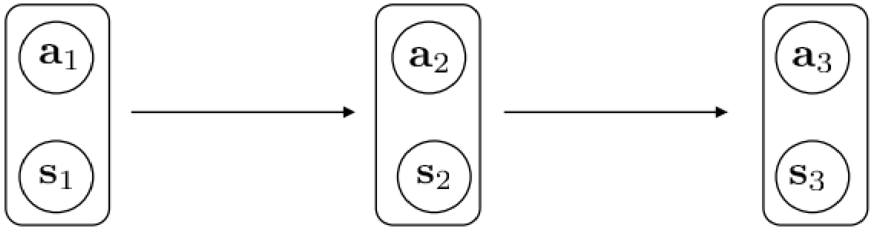
$(\mathcal{T}-\mathbf{I})\mu = 0$

μ is eigenvector of $\mathcal{T}$ with eigenvalue 1!

(always exists under some regularity conditions)

$\mu = p_\theta(\mathbf{s},\mathbf{a})$ stationary distribution

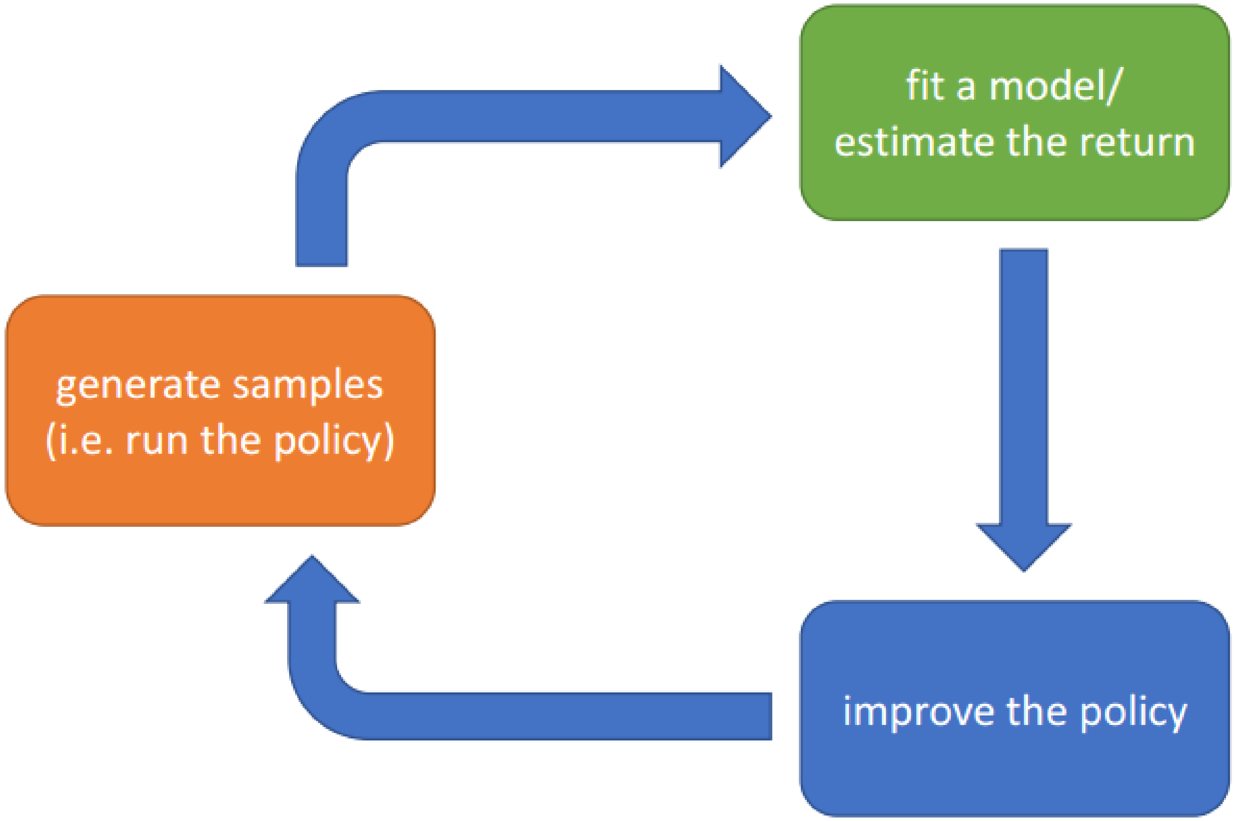
state-action transition operator



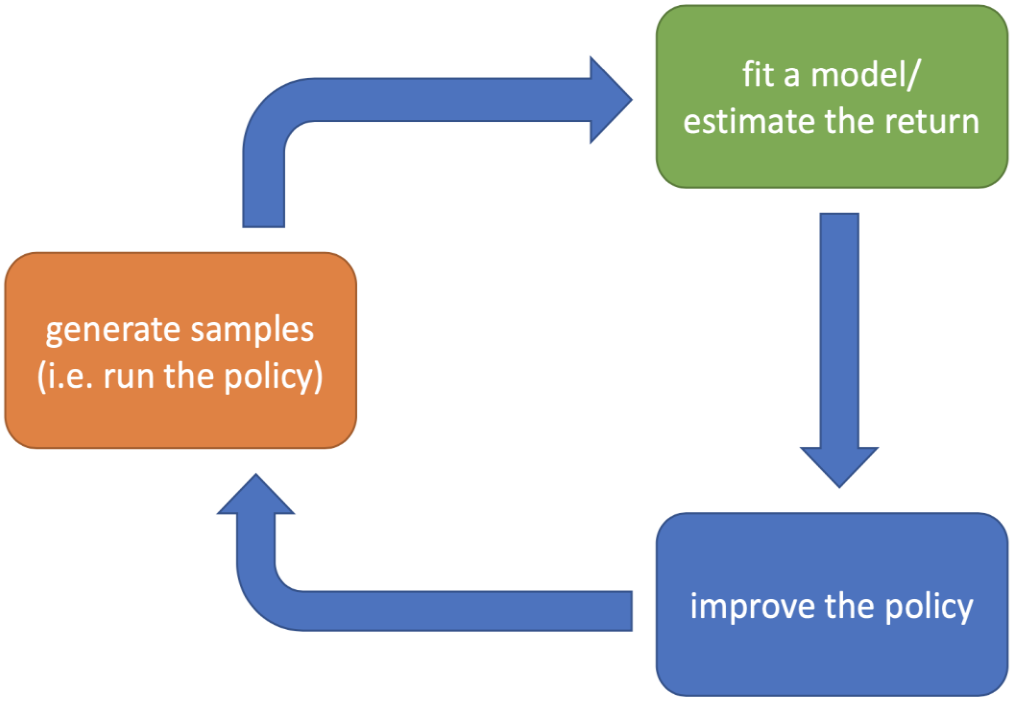
$$\begin{pmatrix}\mathbf{s}_{t+1}\\ \mathbf{a}_{t+1}\end{pmatrix} = \mathcal{T}\begin{pmatrix}\mathbf{s}_{t}\\ \mathbf{a}_{t}\end{pmatrix} \quad \begin{pmatrix}\mathbf{s}_{t+k}\\ \mathbf{a}_{t+k}\end{pmatrix} = \mathcal{T}^k\begin{pmatrix}\mathbf{s}_{t}\\ \mathbf{a}_{t}\end{pmatrix}$$

- The computation of the expected reward is difficult since $p_\theta(s_t,a_t)$ changes at each t
- By taking $T\to\infty$, $p_\theta(s_t,a_t)\to\mu = p_\theta(s,a)$; now it is easier to compute the expected reward

The anatomy of a reinforcement learning algorithm



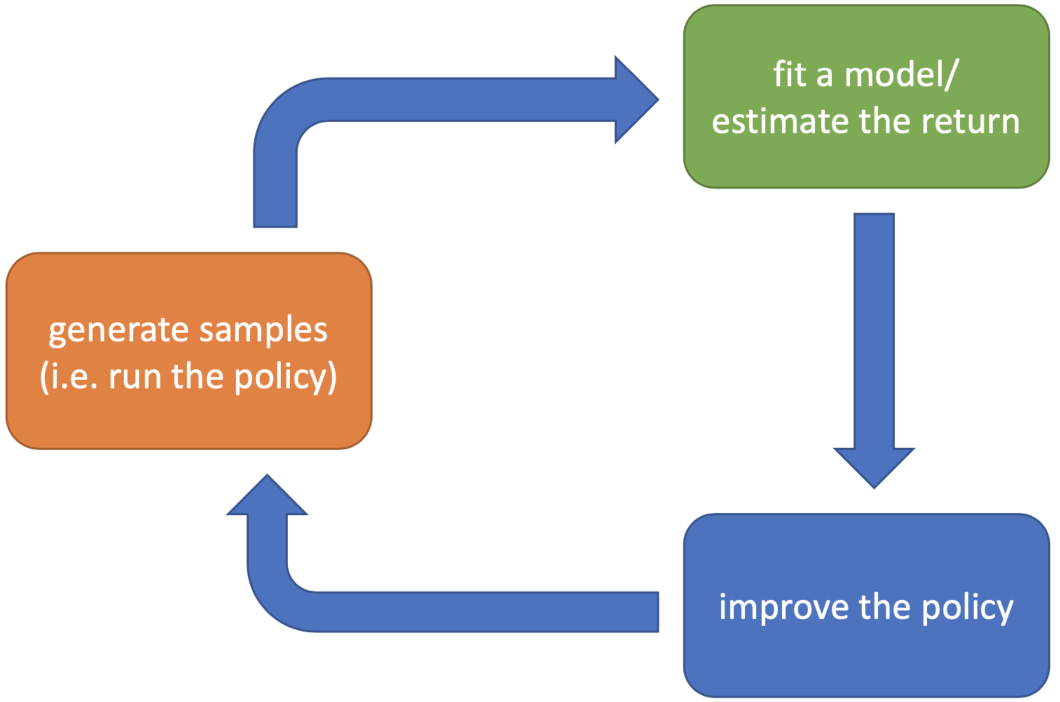
A Simple RL Example


fit a model/ estimate the return
improve the policy
generate samples (i.e. run the policy)

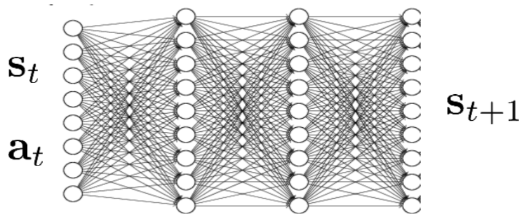
$$J(\theta) = E_\pi\left[\sum_t r_t\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_t r_t^i$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Backpropagation in RL

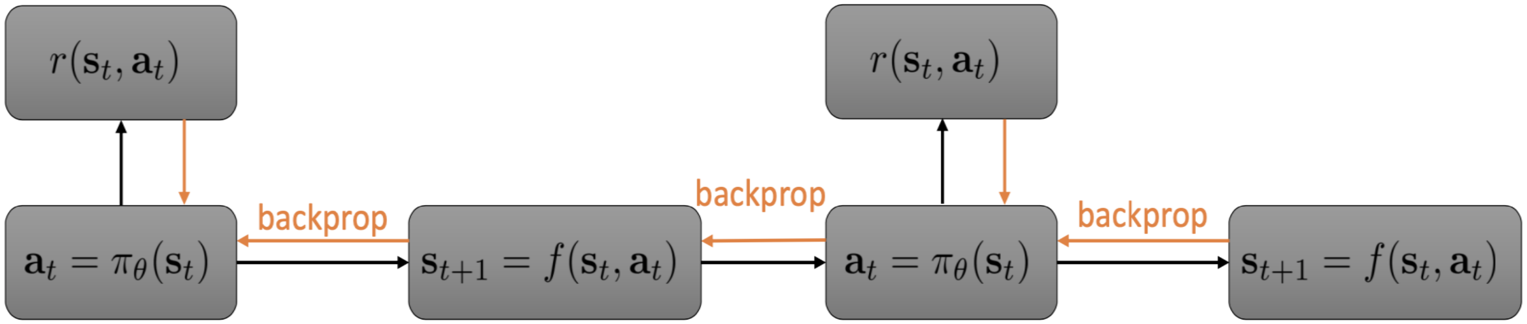


learn f_ϕ such that $\mathbf{s}_{t+1} \approx f_\phi(\mathbf{s}_t, \mathbf{a}_t)$



backprop through f_ϕ and r to train $\pi_\theta(\mathbf{s}_t) = \mathbf{a}_t$

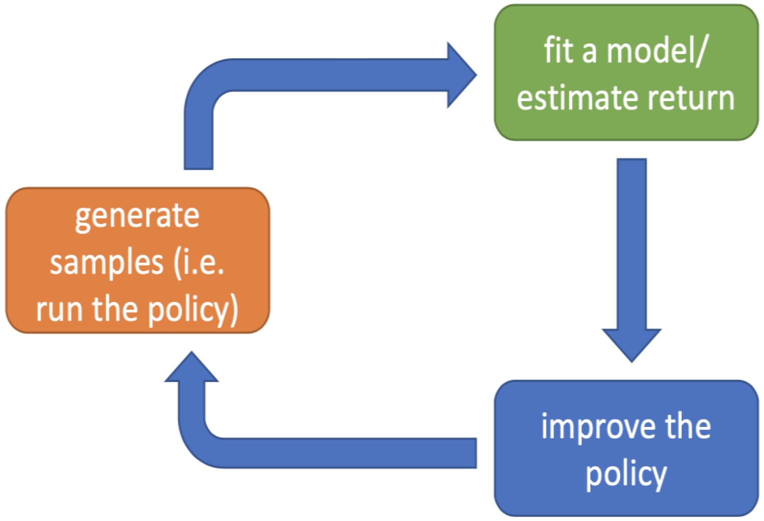
Backpropagation in RL: Example


$r(\mathbf{s}_t, \mathbf{a}_t)$
$r(\mathbf{s}_t, \mathbf{a}_t)$
backprop
backprop
backprop
$\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$
$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$
$\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$
$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$

collect data

update the model *f*

update the policy with backprop

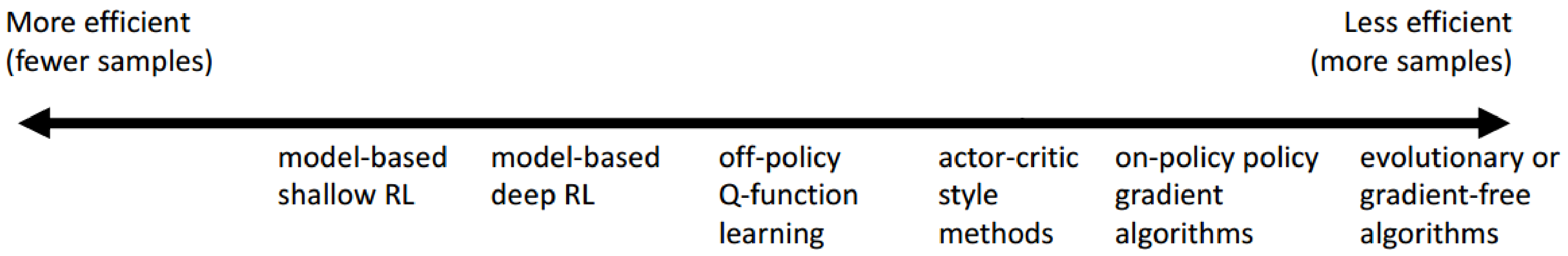

fit a model/ estimate return
improve the policy
generate samples (i.e. run the policy)

Types of RL algorithms

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

- Policy gradients: directly differentiate the above objective
- Value-based: estimate value function or Q-function of the optimal policy (no explicit policy)
- Actor-critic: estimate value function or Q-function of the current policy, use it to improve policy
- Model-based RL: estimate the transition model, and then…
  - Use it for planning (no explicit policy)
  - Use it to improve a policy
  - Something else

Comparison: sample efficiency


More efficient (fewer samples)
Less efficient (more samples)
model-based shallow RL
model-based deep RL
off-policy Q-function learning
actor-critic style methods
on-policy policy gradient algorithms
evolutionary or gradient-free algorithms

Why would we use a *less* efficient algorithm?