CS 295A/395D: Artificial Intelligence

Partial Observability and Markov Decision Processes (MDPs)

Prof. Emma Tosch

25 April 2022

The University of Vermont

Logistics

- Worksheet due Wednesday, programming assignment soft deadline Friday
- Discussion about final topics in the class & final in-class exam
  - Do this at the end of class
- Final exam plan
  - Especially encourage graduate students (model/dry run for qualifying exam)
  - Will cover entire course
  - 40 Multiple choice
  - 2 design questions (pick one)
  - 2 computational questions (pick one)

Agenda

- Handout question 18
- Review: Markov Chains
- Partially observed MDPs (Hidden Markov Models – HMMs)
- Markov Decision Processes (MDPs)
- Introduce Reinforcement Learning

Recall: kth-order Markov Chain

A kth-order Markov chain is defined by the model $\langle S, T, v_0 \rangle$ such that:

$S = \{s_1, \dots, s_n\}$ is the set of *n* states

T is a $n^k \times n$ matrix of transition probabilities such that:

… and let $v_i = \langle s^1, \dots, s^k \rangle$

$$p_{ij} = P\big(X_t = s_j \mid X_{t-1} = v_i(k), \dots ,\ X_{t-2} = v_i(1) \big)$$

$$v_0 = \langle P(X_2 = s_1, X_1 = s_1), P(X_2 = s_1, X_1 = s_2), \dots, P(X_2 = s_n, X_1 = s_n) \rangle$$

Recall: Important MC properties

- Irreducibility (reachability)
  - For all pairs of states (s, s’) , there exists a sequence of states from s to s’ such that the transition probability from s to s’ is not zero.
- Aperiodicity (coverage)
  - Beyond an index k, it must be possible to observe all states in all indices starting with k+1.

Recall: MCMC

These properties ensure there is a unique stable distribution.

The existence of a unique stable distribution makes *Markov Chain Monte Carlo (MCMC)* sampling possible.

MCMC is a **family** of sampling algorithms. Most famous:

- Gibbs Sampling
- Hamiltonian MCMC

Unobserved or Partially Observed Markov Chains

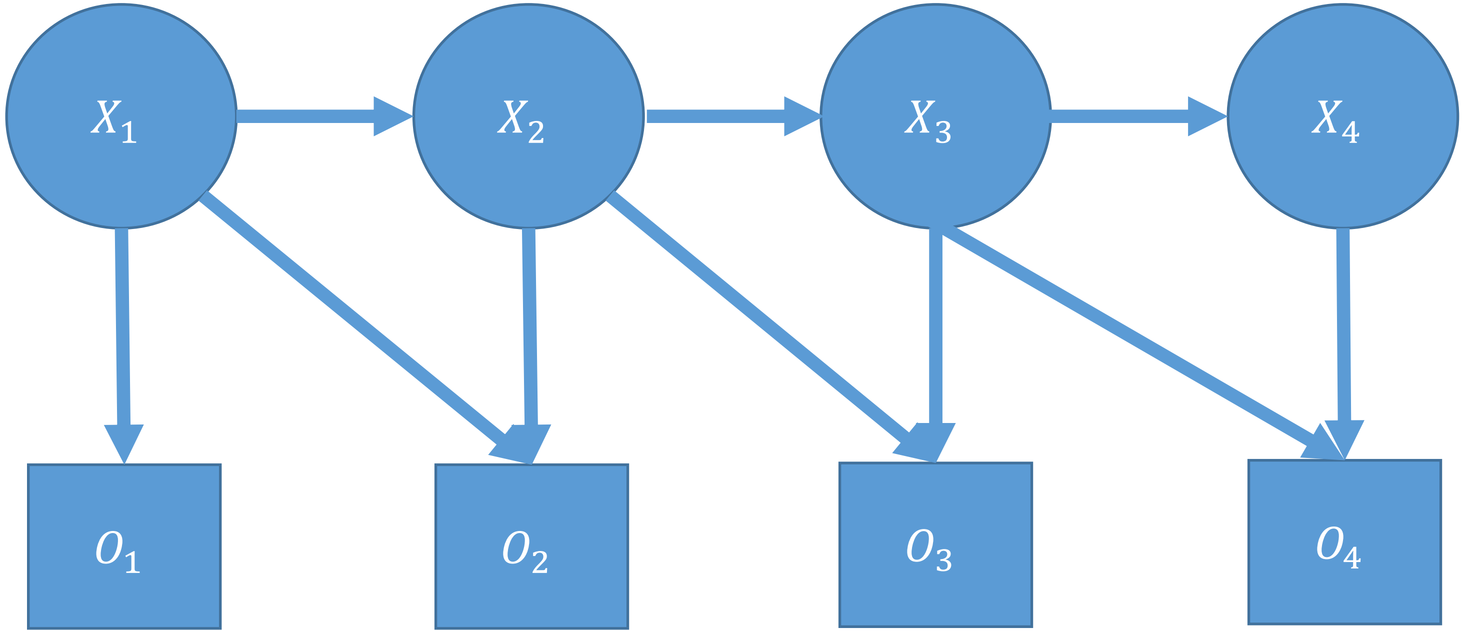
Thus far: assumed we knew what state we were in (“observed”)

Sometimes we don’t know!

Example: Part of speech tagging

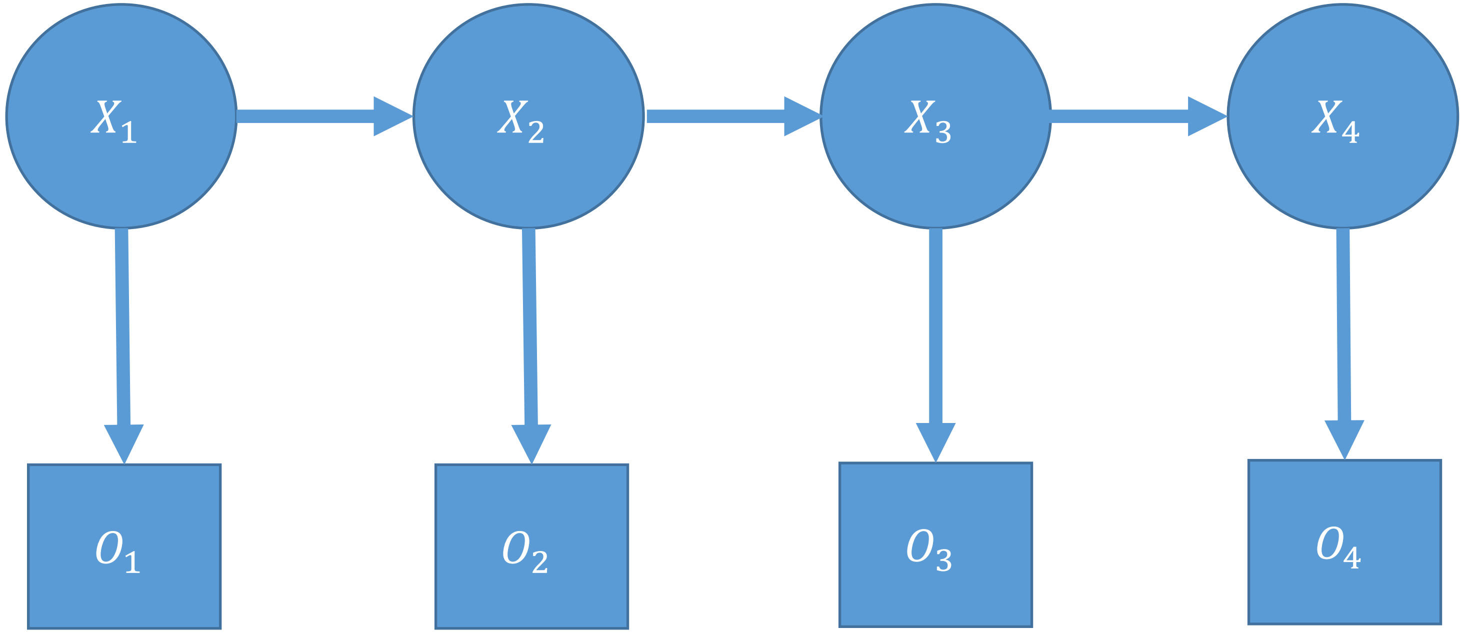
- Parts of speech (e.g., noun, verb, adjective) are necessary for *parsing*
- Parsing is the classical backbone of text analysis

Example POS tagging



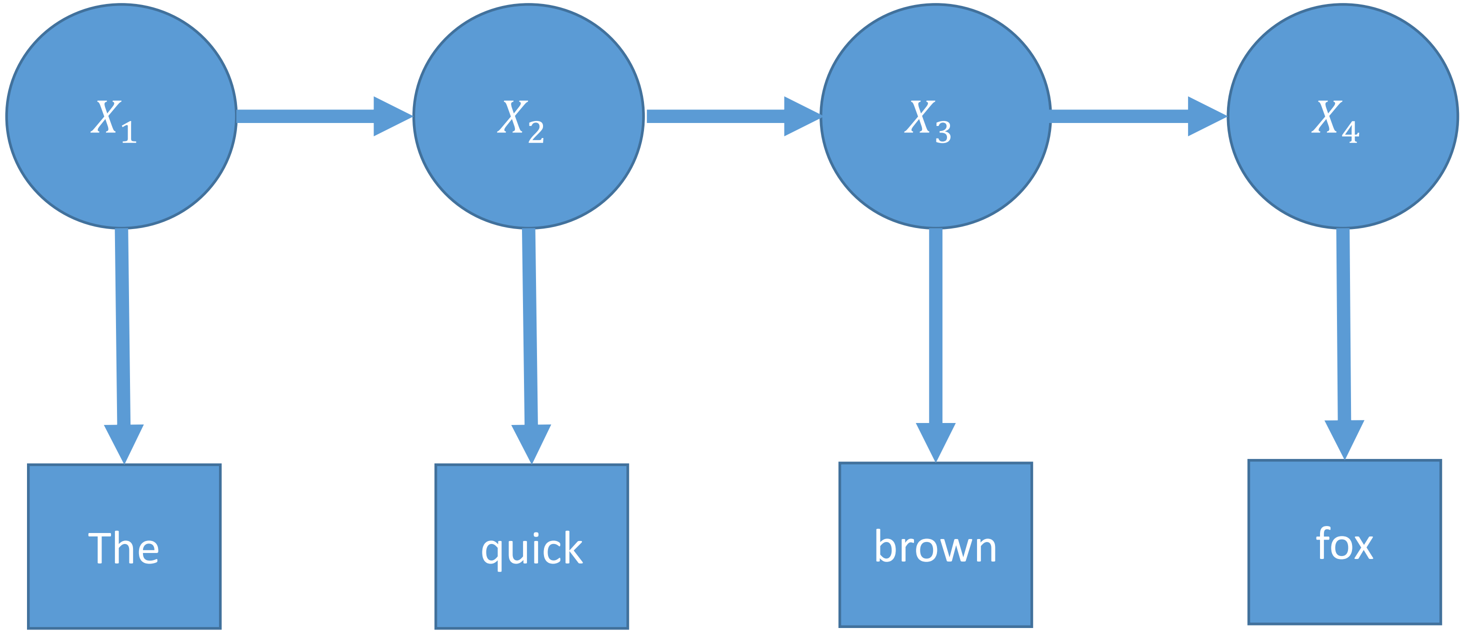
Example POS tagging

$O_t \mid X_t \sim Distr(\theta)$

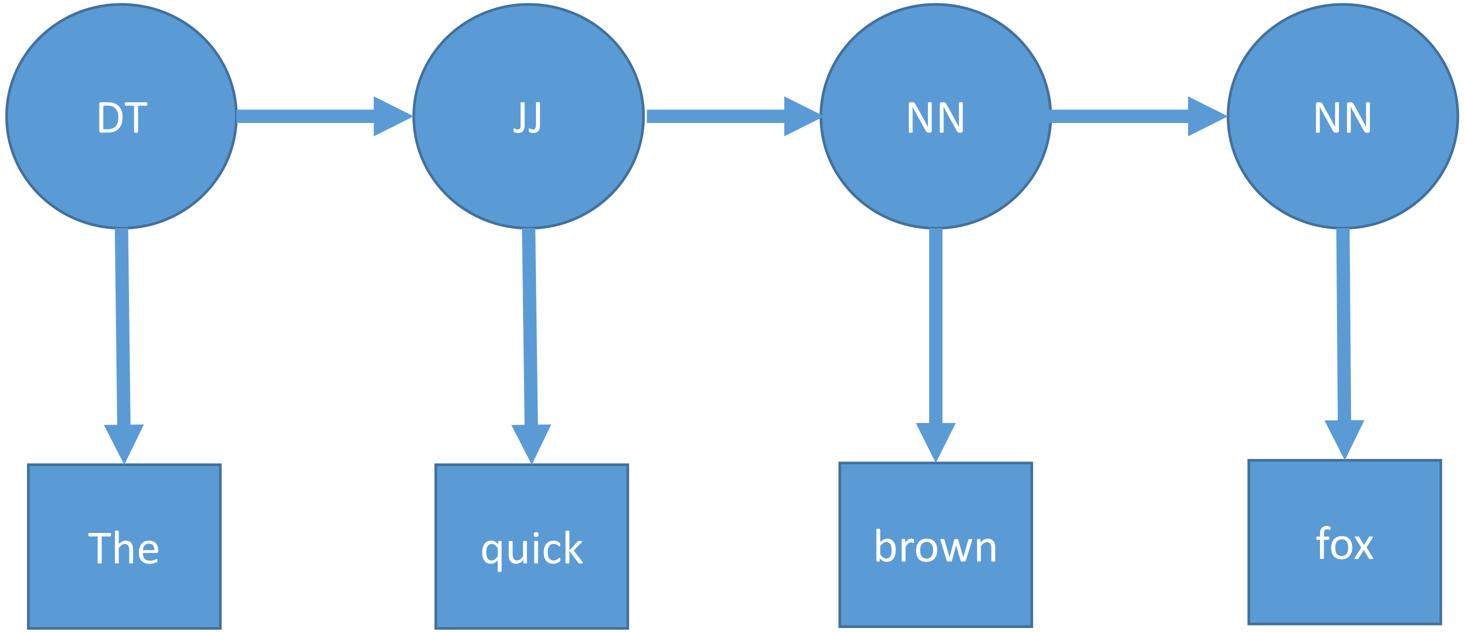


Need to augment model $\langle S, T, v_0 \rangle$ with *emission probabilities* for the observations

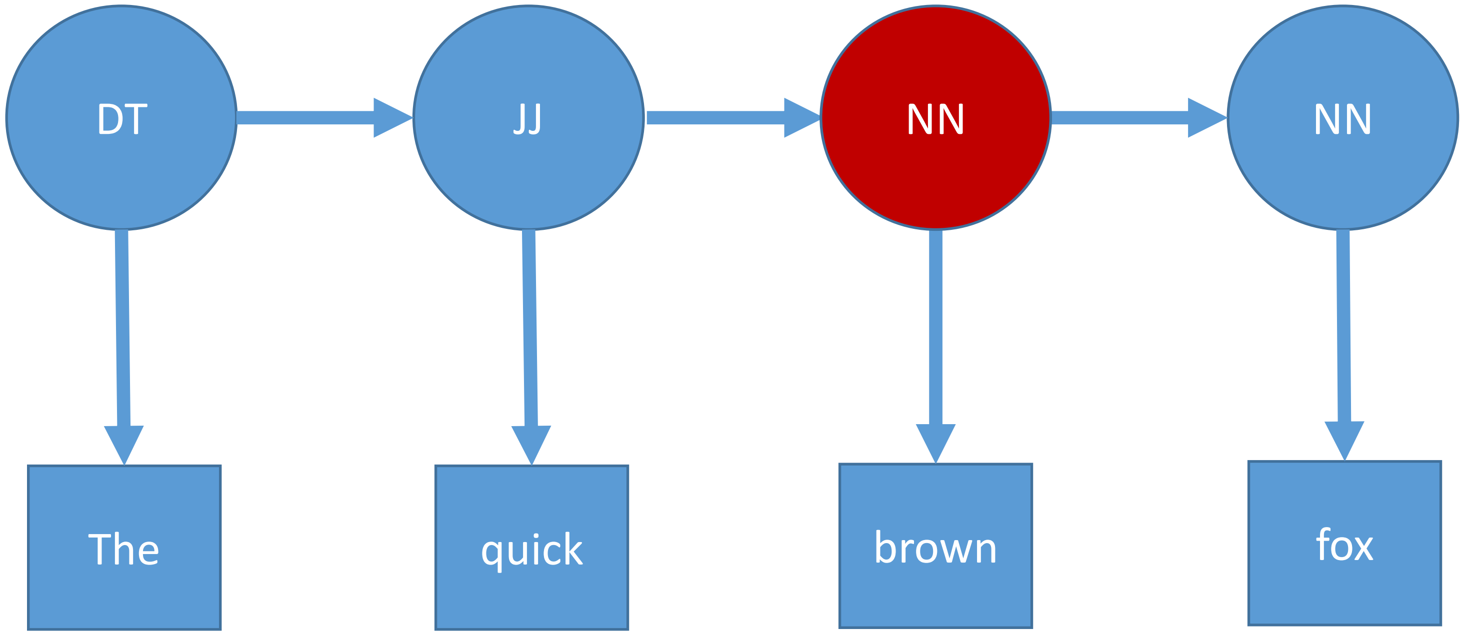
Example POS tagging



Example POS tagging



Example POS tagging



Planning vs. Inference vs. Learning

So far we have focused on planning:

- We know the structure of the problem space
- We need to navigate the problem space, e.g.
    - Searching for proof trees
    - Robot navigating a room
    - Making optimal choices under uncertainty
- Primary challenge has been to navigate a very large space of options

Planning vs. Inference vs. Learning

We have recently encountered *inference*:

- Given a model that may contain uncertainty, draw conclusions from that model
- Planning uses inference to *selection actions*

Inference relies on us having a good enough model of the problem

- *Learning* is the study of procedures for crafting that model from *data*

Inference problem for HMMs

Given:

1. A model $\langle S, T, \mathrm{v}_0, F \rangle$ (where F is the emission function)

2. An input $\langle o_1, o_2, \ldots, o_m \rangle$

Output the hidden states (e.g., parts of speech) $\langle x_1, x_2, \ldots, x_m \rangle$

How might we do this?

Exercise: suboptimality of greedy inference

Solution: forward-backward reasoning

Idea:

- Use dynamic programming (i.e., memorization) to find most likely state
- Computes *forward* and *backward* passes
    - Uses Bayes rule and law of total probability to get the marginal likelihood over a single state
- Naïve approach highly localized – does not consider sequence
- Max sequential path: Viterbi algorithm

Machine Learning vs. AI for text

POS tagging less popular proportionally

- Requires expensive data sets to train
- Mature field (fewer opportunities to publish in top conferences)
- Requires domain knowledge for new languages (linguistics)
- Off-the-shelf pre-trained software works, can build into pipelines

Now: deep neural networks for text analysis

- Can skip intermediate steps (don’t need the POS in the first place)
- Thus, can often skip domain-specific knowledge of language structure
- Uninterpretable, hard to debug
- Power usage to train

MDPs, formally

Reward vs. utility?

Early in the semester:

- Cost — ML, from statistics

Later:

- Utility — Utility theory (economics)

Now:

- Reward — Reinforcement (psychology)

There are effectively all the same; why use different terms?

Action selection representation

Recall: in game theory, the action or actions we took were a “strategy”

Here: an action sequence is a “policy” (confusingly denoted π)

- Typical representations:
  - Action sequence as a matrix of probabilities
    - Finite number of states and actions!
  - Action sequence as a function

Similarities with decision theory

- Finite state space (at least in our course)
- Finite action space (again, in our course)
- Objective: maximize utility (minimize cost, maximize reward)
- Elides aleatory and epistemic uncertainty

Differences compared with decision theory

Decision trees

- Finite horizon (tree has finite size)
- Small branching factor
- State nodes not necessarily ordered
- Decisions/actions decoupled from state
- Purely associational

MDPs

- Infinite horizon (potentially infinite chain)
- Potentially very large branching factor
- State nodes must be ordered
- Decisions/actions tightly coupled with state
- Sometimes causal (maybe not identifiable)

Do actions mutate state?

MDPs vs. Reinforcement Learning (RL)

Who here has heard of reinforcement learning?

- MDPs are about the *problem space*
- RL is about *learning* an optimal policy (i.e. strategy) for acting in a dynamic environment

RL can be used to learn over MDPs, but these two concepts are **not** synonymous!

- If the environment can be modeled as an MDP, certain things become *easier*
- Classical RL assumed MDPs
- Modern RL does not