Hidden Markov Models

- Sometimes we need to model things that change over time. We will assume that time is discrete, divided into individual timesteps. At each timestep t, the world (modeled as a set of random variables X_t) can be in a certain **state** (assignment of the variables), and it can **transition** to a different state at the next timestep.

- A **stationary process** is one in which the transition probabilities and the distributions of the random variables do not change over time.

- A **Markov process** or **Markov chain** is a sequence of random variables $X_0, X_1, ..., X_t$ that follows the **Markov assumption**:
  $P(X_t|X_{0:(t-1)}) = P(X_t|X_{(t-k):(t-1)})$
  or in other words, the distribution of X_t depends on only k of the previous X's (where k is some fixed finite number, usually 1; the process is called a kth order Markov process). In a first-order Markov model, this distribution is the **transition model**.

- We will assume that the state variables X_t are unobservable, but there are some evidence variables E_t that we observe. The **sensor Markov assumption** is:
  $P(E_t|X_{0:t}, E_{1:(t-1)}) = P(E_t|X_t)$
  or in other words, the evidence at the current timestep depends only on the current state, not any previous states or previous evidence. This distribution is the **observation model**.

- In a **Hidden Markov Model (HMM)**, we model the unobserved state of the world as a single discrete variable (X_t is one random variable instead of a set of them), and we have a process that follows the Markov assumption and the sensor Markov assumption.

- There are several types of inference tasks that we might want to do:
  - In **filtering** (also called **state estimation**), we compute $P(X_t|e_{1:t})$, the distribution of values for the current state given all the observations.
  - In **prediction**, we compute $P(X_{t+k}|e_{1:t})$, the distribution of values for some state that is k steps into the future, given all the observations up to the present.
  - In **smoothing**, we compute $P(X_k|e_{1:t})$, the distribution of values for some past state ($k < t$), given all the observations up to the present.
  - In finding the **most likely explanation**, we compute $\arg\max_{x_{0:t}} P(x_{0:t}|e_{1:t})$, the most likely values for all states up to the present, given all the observations up to the present.

Exercises

1. Knowing that you're an expert on machine learning techniques, you're called in by the Medical Association of Sports Scientists to investigate the link between sprains and knee injuries over time. You observe individual athletes on a monthly basis and observe whether or not the athlete has a sprain (S = true if a sprain is present); from this you wish to infer the condition of the athlete's patella (P = true if a patellar injury is present; patellar injuries are known to cause sprains[1]). Your friend who is in medical school stops by and tells you the following useful information:

$$\begin{aligned} P(P_0) &= 0.5 \\ P(P_t|P_{t-1}) &= 0.7 \\ P(P_t|\neg P_{t-1}) &= 0.3 \end{aligned}$$

$$\begin{aligned} P(S_t|P_t) &= 0.9 \\ P(S_t|\neg P_t) &= 0.2 \end{aligned}$$

You quickly realize that P follows a first-order Markov process, and S satisfies a sensor Markov assumption based on P. Determine the following:

(a) $P(P_1)$

(b) $P(P_1|S_1 = true)$

(c) $P(P_2|S_1 = true)$

(d) $P(P_2|S_1 = true, S_2 = true)$

(e) Write a formula for computing $P(P_3|S_{1:3})$ using only values that you have calculated above.

(f) In addition to the above observations, you find out $S_3 = false$. What is the probability of $(P_3 = false|S_{1:3})$?

(g) Suppose that you make no further sprain measurements beyond S_3. Give an approximation for the probability distribution of P_{10}.

[1] The 6.034 staff will not be held responsible for any liabilities caused by use of this rule in medical applications.

Search Review

- Basic search algorithms aim to find the shortest path in a graph with undirected, unlabeled edges. They follow a common structure:

  ```
  Until we find a goal or the agenda is empty:
      Extract a node from the agenda
      Expand it (find its children)
      Add its children to the agenda
  ```

- **Breadth-First Search** implements the agenda as a queue, whereas **Depth-First Search** implements the agenda as a stack.
- **Pruning Rule 1**: Don't consider any path that visits the same state twice.
- **Pruning Rule 2**: Don't consider any path that visits a state that you have already visited via some other path. (BFS only.)
- Often we will want to assign weights to edges (eg. Google Maps; some streets might be physically longer or might be more congested.) **Uniform-Cost Search** implements the agenda as a priority queue sorted by path length and extracts the minimum length path at each step.
- Search can often be sped up through use of **heuristics**. Popular heuristics are Euclidean distance, Manhattan distance, and Hamming distance.
- **Hill-Climbing Search** is similar to DFS, except it picks the child with the lowest heuristic cost at each step. It is not guaranteed to find a global optimum.
- **A* Search** is similar to Uniform-Cost Search, except it sorts paths in the agenda by their total cost so far plus their remaining heuristic cost.
- An A* search is only guaranteed to return the shortest path if its heuristic is **admissible**, and can be made much faster if its heuristic is also **consistent**.
- Suppose G is our goal node, $h(N)$ is the heuristic cost to reach G from N, $C(N)$ is the length of the shortest path between N and G, and $d(A, B)$ is the distance from A to B, assuming B is a direct child of A. A heuristic is admissible if $\forall N, h(N) \leq C(N)$. A heuristic is consistent if $h(N) \leq d(N, P) + h(P)$ and $h(G) = 0$. Consistent heuristics are also admissible.