# CSCI 5582
# Artificial Intelligence

Lecture 16

Jim Martin



# Today 10/24

- Review basic reasoning about sequences
- Break
- Hidden events
- 3 Problems

# Chain Rule

P(E1,E2,E3,E4,E5)

P(E5|E1,E2,E3,E4)P(E1,E2,E3,E4)

P(E4|E1,E2,E3)P(E1,E2,E3)

P(E3|E1,E2)P(E1,E2)

P(E2|E1)P(E1)



# Chain Rule

Rewriting that's just

P(E1)P(E2|E1)P(E3|E1,E2)P(E4|E1,E2,E3)P(E5|E1,E2,E3,E4)

The probability of a sequence of events is just the product of the conditional probability of each event given it's predecessors (parents/causes in belief net terms).

# Markov Assumption

- This is just a sequence based independence assumption just like with belief nets.
  - Not all the previous events matter

    P(EventN|Event1 to Event N-1)=

    P(EventN|EventN-1+K to Event N-1)



# First Order Markov

P(E1)P(E2|E1)P(E3|E1,E2)P(E4|E1,E2,E3)P(E5|E1,E2,E3,E4)

P(E1)P(E2|E1)P(E3|E2)P(E4|E3)P(E5|E4)

# Markov Models

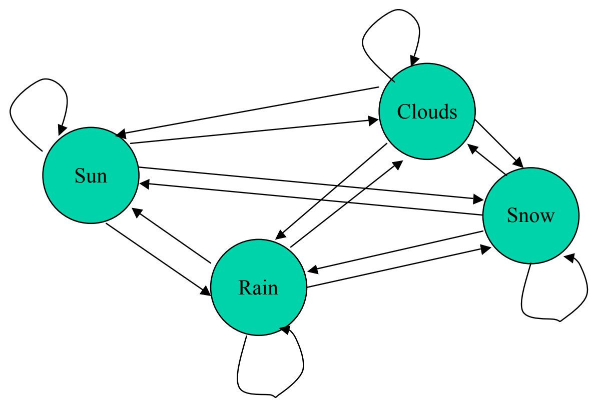
- You can view simple Markov assumptions as arising from underlying probabilistic state machines.
- In the simplest case (first order), events correspond to states and the probabilities are governed by probabilities on the transitions in the machine.



# Weather

- Let's say we're tracking the weather and there are 4 possible events (each day, only one per day)
  - Sun, clouds, rain, snow

# Example






# Example

- In this case we need a 4x4 matrix of transition probabilities.
  - For example P(Rain|Cloudy) or P(Sunny|Sunny) etc
- And we need a set of initial probabilities P(Rain). That's just an array of 4 numbers.

# Example

- So to get the probability of a sequence like
  - Rain rain rain snow
  - You just march through the state machine
  - P(Rain)P(rain|rain)P(rain|rain)P(snow|rain)



# Example

- Say that I tell you that
  - Rain rain rain snow has happened
  - How would you answer
    - What's the most likely thing to happen next?
  - Say I set this all up, gave you a big history of weather events, but I didn't give you the probabilities in the model?

## Hidden Markov Models

- Add an output to the states. I.e. when a state is entered it outputs a symbol.
- You can view the outputs, but not the states directly.
  - States can output different symbols at different times
  - Same symbol can come from many states.



## Hidden Markov Models

- The point
  - The observable sequence of symbols does not uniquely determine a sequence of states.
- Can we nevertheless reason about the underlying model, given the observations.

# Hidden Markov Model Assumptions

- Now we're going to make two independence assumptions
  - The state we're in depends probabilistically only on the state we were last in (first order Markov assumpution)
  - The symbol we're seeing only depends probabilistically on the state we're in



# Hidden Markov Models

- Now the model needs
  - The initial state priors
    - $P(State_i)$
  - The transition probabilities (as before)
    - $P(State_j | State_k)$
  - The output probabilities
    - $P(Observation_i | State_k)$

# HMMs

- The joint probability of a state sequence X and an observation sequence E is…

$$P(X_0, X_1, \ldots X_t, E_1, \ldots E_t) = P(X_0)P(E_0 \mid X_0)\prod_{i=1}^{t} P(X_i \mid X_{i-1})P(E_i \mid X_i)$$



# Noisy Channel Applications

- The hidden model represents an original signal (sequence of words, letters, etc)
- This signal is corrupted probabilistically. Use an HMM to recover the original signal
- Speech, OCR, language translation, spelling correction,…

# Noisy Channel Basis

Decoding…

Argmax P(state seq|obs) =

P(obs | state seq)P(state seq)

Now make 2 First Order Markov assumptions:

Outputs depend only on the state

Current state depends only on the previous state

$$P(X_0, X_1, \ldots X_t, E_1, \ldots E_t) = P(X_0)P(E_0 \mid X_0)\prod_{i=1}^{t} P(X_i \mid X_{i-1})P(E_i \mid X_i)$$



# Three HMM Problems

- The probability of an observation sequence given a model
  - Forward algorithm
  - Prediction falls out from this
- The most likely path through a model given an observed sequence
  - Viterbi algorithm
  - Sometimes called decoding
- Finding the most likely model (parameters) given an observed sequence
  - EM Algorithm

## Problem 1

- What's the probability assigned to a given sequence of observations given a model
  - P(Output sequence|Model)



## Problem 1

- Solution:
  - Enumerate all the possible paths through a model and calculate the probability that each path could have produced the observed sequence.
  - Sum them all; that's the probability that this model could have produced the observed output

## Problem 2

- This is really diagnosis over again. What state sequence is most likely to have caused this observed sequence?
  - Argmax P(State Sequence | Observations)



## Problem 2

- Solution:
  - Enumerate all the paths through the model and calculate the probability that each path could have produced the observed output.
  - Pick the path with the highest probability (argmax)

# Problem 3

- This turns out to be a simple local optimization (hill-climbing) search for the set of parameters (A, B, π) that maximizes the probability of the observed sequence.



# Problems

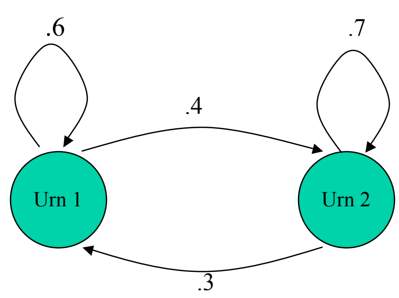
- Of course, there's a minor problem with our solutions to Problems 1 and 2.
  - There are too many paths to enumerate them all and calculate their probabilities
  - The solution is to use the Markov assumption to get a dynamic programming solution to each

# Urn Example

- A genie has two urns filled with red and blue balls. The genie selects an urn and then draws a ball from it (and replaces it). The genie then selects either the same urn or the other one and then selects another ball…



# Urn Example



# Urns and Balls

- Π Urn 1: 0.9; Urn 2: 0.1
- A

| | Urn 1 | Urn 2 |
|---|---|---|
| Urn 1 | 0.6 | 0.4 |
| Urn 2 | 0.3 | 0.7 |

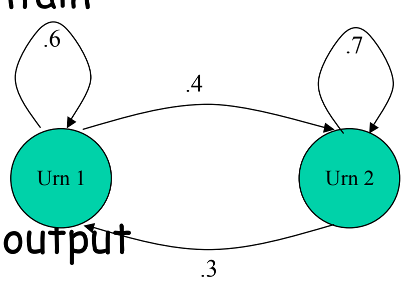
- B

| | Urn 1 | Urn 2 |
|---|---|---|
| Red | 0.7 | 0.4 |
| Blue | 0.3 | 0.6 |



# Urns and Balls: Problem 1

- Let's assume the input (observables) is Blue Blue Red (BBR)
- Since both urns contain red and blue balls any path through this machine could produce this output



# Urns and Balls

- But those paths are not equally likely
  - We need the probability of either urn starting the string
  - The probability of the next urn given the first one
  - The probability of the given urn giving up either a red or blue ball
  - For each possible path



# Urns and Balls

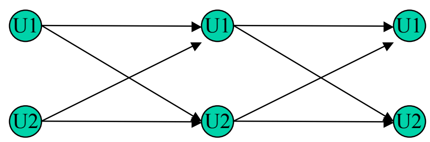
Blue Blue Red: We want P(this seq | model)

| | |
|---|---|
| 1 1 1 | (0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204 |
| 1 1 2 | (0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077 |
| 1 2 1 | (0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136 |
| 1 2 2 | (0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181 |

| | |
|---|---|
| 2 1 1 | (0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052 |
| 2 1 2 | (0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020 |
| 2 2 1 | (0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052 |
| 2 2 2 | (0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070 |

# Urns and Balls

- Another view of this



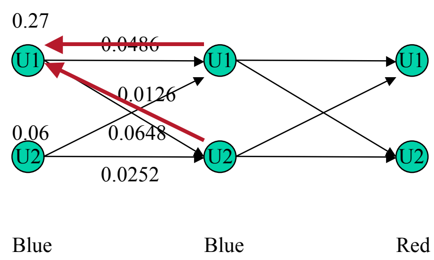



# Urns and Balls: Viterbi

- Problem 2: Most likely path?
  - Argmax P(Path|Observations)
- Sweep through the columns left to right computing the partial path probabilities
  - Keep track of the best (MAX) path to each node as you go

# Urns and Balls

- Another view of this



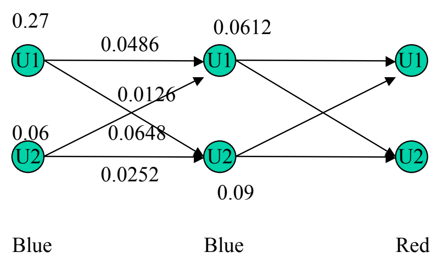



# Urns and Balls: Forward

- Problem 1: Probability of a input sequence given a model
  - P(Inputs | Model)
- Sweep through the columns, left to right, summing the partial path probabilities as you go

# Urns and Balls

- Another view of this






# Urns and Balls

- EM
  - What if I told you I lied about the numbers in the model (π,A,B).
  - Can I get better numbers just from the input sequence?

## Urns and Balls

- Yup
  - Just count up and prorate the number of times a given transition was traversed while processing the inputs.
  - Use that number to re-estimate the transition probability



## Urns and Balls

- But… we don't know the path the input took, we're only guessing
  - So prorate the counts from all the possible paths based on the path probabilities the model gives you
- But you said the numbers were wrong
  - Doesn't matter; use the original numbers then replace the old ones with the new ones.