Lecture 28: Exam 2 Review

Mark Hasegawa-Johnson, 4/2022

Exam 2 Mechanics

- If you’re in the online section, or have signed up to take the exam online, you should get an email sometime during the day Friday (4/1) giving you a zoom URL
- If you’ve signed up for a conflict exam, you should get email sometime during the day Friday (4/1) to schedule it
- Otherwise, please show up here (1002 ECEB) on Monday 4/4 at 1:00pm.

Exam 2 Mechanics

- Permitted: one page of handwritten notes, front & back
- Not permitted: calculators, computers, textbook

Exam 2 Content

- Lectures 15-18: Search
  - Search: BFS, DFS, Explored set, Explored dict
  - UCS, Greedy, A* Search
  - Heuristics: admissible, consistent, dominant
  - Constraint satisfaction problems
- Lectures 20-24: Bayesian networks
  - Bayesian networks; inference by enumeration
  - Learning: Laplace smoothing, Expectation maximization
  - Hidden Markov models
  - Viterbi algorithm
  - Part of speech tagging

Search

- BFS: frontier is a queue
  - Time complexity = space complexity = $O\{b^d\}$
  - Complete and, if every step has the same cost, optimal
- DFS: frontier is a stack
  - Time complexity = $O\{b^m\}$, space complexity = $O\{mb\}$
  - Neither optimal nor even (if there are loops or an infinite search space) complete
- UCS: frontier = priority queue sorted by g(n)
  - Complete and optimal
  - Time complexity = space complexity = # states with g(n) < best path to goal
- Greedy: frontier = priority queue sorted by h(n)
  - Neither complete nor optimal
  - Time complexity = space complexity = $O\{b^m\}$
- A*: frontier = priority queue sorted by h(n)+g(n). Complete and optimal if:
  - h(n) admissible and explored set not used (explored dict is OK), or if
  - h(n) consistent
  - Time complexity = space complexity = # states with g(n)+h(n) < best path to goal

Explored set vs. Explored dict

- Explored set
  - Advantage: complexity is never worse than exhaustive search
  - Disadvantage: suboptimal if there is any reason to think the first path to a state might ever not be the best path to that state
- Explored dict
  - Advantage: guaranteed to be optimal
  - Disadvantage: not guaranteed to limit complexity below that of an exhaustive search, if there is any reason to think the first path to a state might ever not be the best path to that state

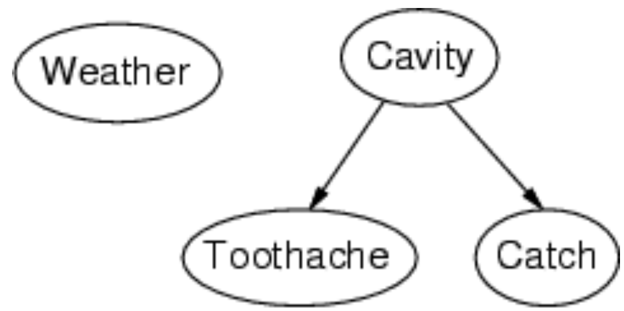
Heuristics

- Admissible: $h(n) \leq d(n)$
  - Guarantees that the first time you expand the goal state, it will be the best path
- Consistent: $h(m) - h(n) \leq d(m) - d(n)$ if $d(m) - d(n) \geq 0$
  - Guarantees that the first time you expand any state, it will be the best path to that state
- Dominant: $h_1(n)$ dominates $h_2(n)$ if $h_1(n) \geq h_2(n)$ for all n.
  - If both are admissible, it guarantees that search using $h_1(n)$ will be faster than search using $h_2(n)$

Constraint satisfaction problem

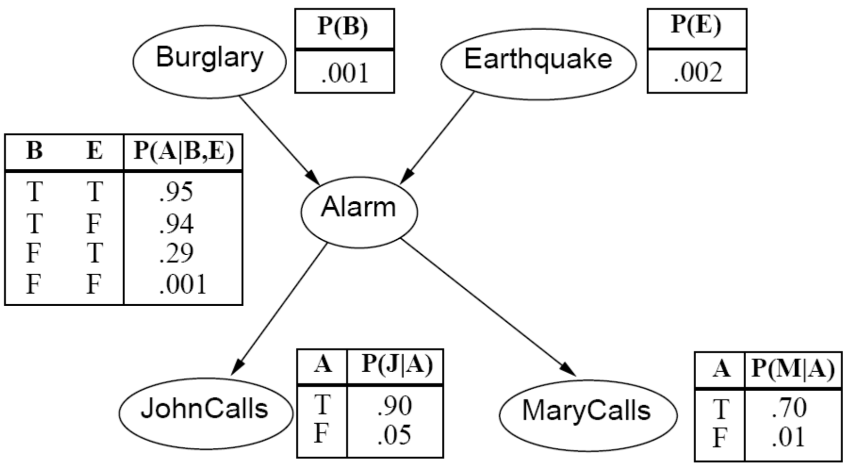
- Every path to goal has the same depth, so DFS is as good as BFS
- Every successful path has the same cost, so the A* complexity guarantee (# states with $g(n)+h(n)<$best cost) is trivial and useless
- Instead, we use heuristics that rank-order search candidates based on rough estimates of probability of success
- LRV: choose the variable with the fewest remaining values
  - Minimize the current branching factor
- MCV: choose the variable that causes the most constraints
  - Because you'll have to solve that variable eventually
- LCV: choose the value that causes the fewest constraints
  - Because it's most likely to be the correct answer

Bayesian networks: Structure



- **Nodes:** random variables
- **Arcs:** interactions
  - An arrow from one variable to another indicates direct ***causal*** influence of variable #1 on variable #2
  - Must form a directed, acyclic graph

Conditional Independence ≠ Independence



- B and E (no common ancestor, common descendant A):
  - Independent
  - Not conditionally independent given A
- J and M (common ancestor A, no common descendant):
  - Not independent
  - Conditionally independent given A
- B and M (B is the ancestor of M):
  - Not independent
  - Conditionally independent given A

Belief propagation, step by step

1. Identify a path through the Bayesian network that includes all variables, including the query variable and all observed variables, starting at their common ancestor
2. Calculate the joint probability of the query variable and all observed variables, iteratively marginalizing out all intermediate variables step-by-step along the path.
    1. Product Step: $P(A,B,C) = P(A,B)P(C|A,B)$
    2. Sum Step: $P(A,C) = \sum_b P(A,B=b,C)$
3. Apply Bayes' rule to get the desired conditional probability

Laplace smoothing

Just like in naïve Bayes:

- Laplace smoothing makes it possible for things to happen in the test data that never happened in the training data. For example, maximum likelihood resulted in $P(F = F|S = T, A = T) = 0$, but with Laplace smoothing, we smooth that parameter to $P(F = F|S = T, A = T) = \frac{k}{1+2k}$
- This smoothing improves generalization from training data to test data.

Laplace smoothing

Unlike naïve Bayes:

- In Bayesian networks, we usually assume that we know the cardinality of each random variable in advance, so no extra probability mass is kept aside for OOV events.

$$P(X = x|H = h) = \frac{(\#\text{ observations of }(\mathrm{H}{=}h, \mathrm{X}{=}x)){+}k}{(\#\text{ observations of }(\mathrm{H}{=}h)) + k \cdot (\#\text{ distinct values of }X)}$$

Expectation Maximization (EM): Main idea

Remember that maximum likelihood estimation counts examples:

$$P(F = T | A = a, S = s) = \frac{\#\text{ days } A{=}a, \; S{=}s, \; F{=}T}{\#\text{ days } \; S{=}s, A{=}a}$$

Expectation maximization is similar, but using "expected counts" instead of actual counts:

$$P(F = T | A = a, S = s) = \frac{E[\#\text{ days } A = a, S = s, F = T]}{E[\#\text{ days } A = a, S = s]}$$

Where E[X] means "expected value of X".

Expectation Maximization (EM) is iterative

INITIALIZE: **guess** the model parameters.

ITERATE until convergence. If F and A are fully observed on each day, but S is sometimes unobserved, then:

1. **E-Step**: $E[\#\text{ days } S = s, A = a, F = f] = \sum_{i:a_i=a,f_i=f} P(S = s|a, f)$
2. **M-Step:** $P(F = f|S = s, A = a) = \frac{E[\#\text{ days } S=s,A=a,F=f]}{E[\#\text{ days } S=s,\ A=a]}$

Continue the iteration, shown above, until the model parameters stop changing.

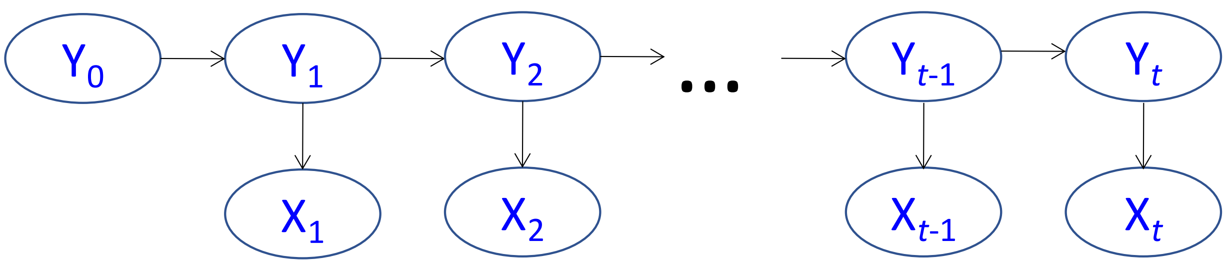
Hidden Markov Model

- A hidden Markov model assumes that both the state and the observation are Markov.
- **<u>State Transitions</u>:** the Markov assumption means that each state variable depends only on the preceding time step:

$$P(Y_t \mid Y_0, \ldots, Y_{t-1}) = P(Y_t \mid Y_{t-1})$$

- **<u>Observation model</u>:** the Markov assumption means that each state variable depends only on the current state:

$$P(X_t \mid Y_0, \ldots, Y_t, X_1, \ldots, X_{t-1}) = P(X_t \mid Y_t)$$

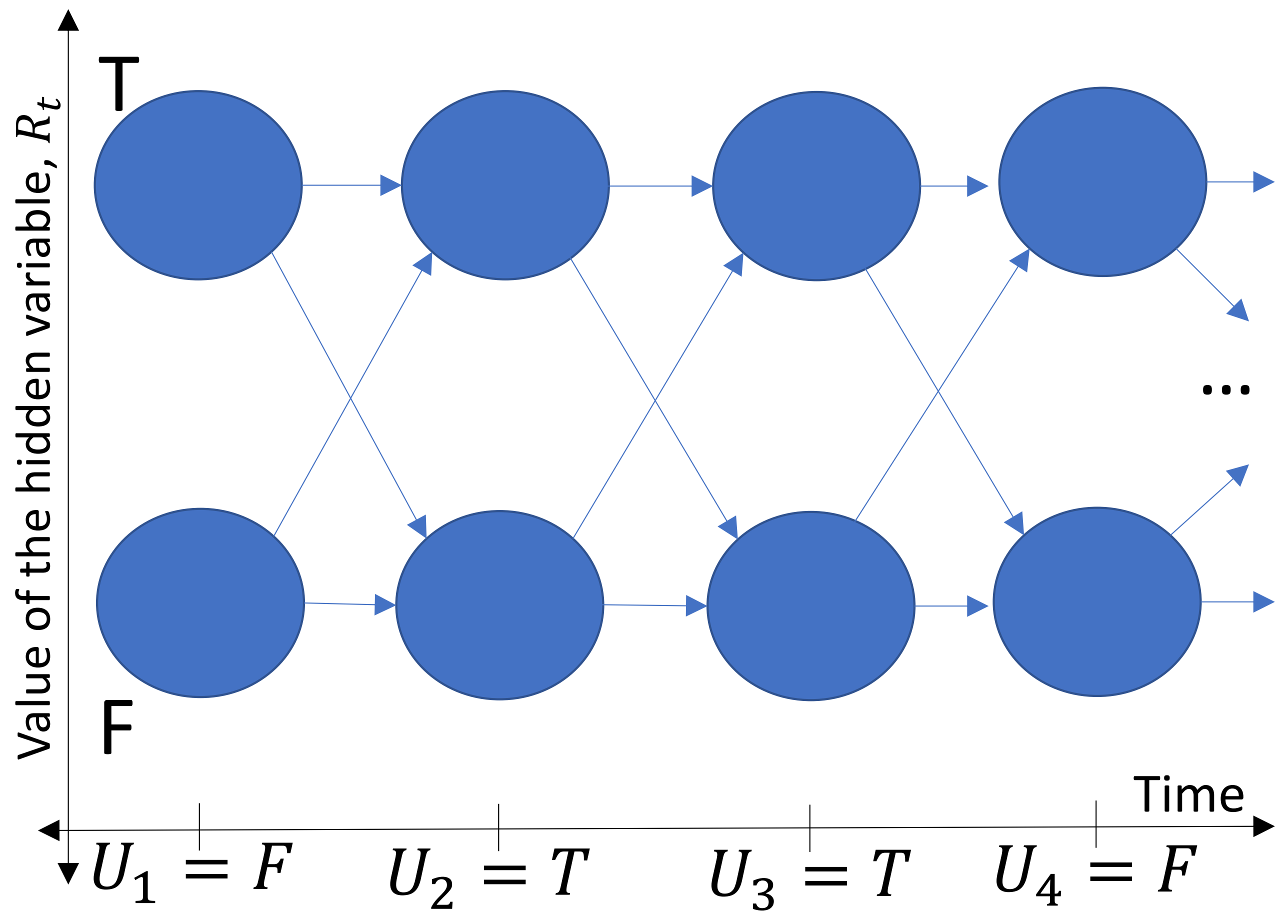


Outline

- Belief propagation
    - What is $P(Y_t | X_1 = x_1, \ldots, X_T = x_T)$?
- Viterbi Algorithm
    - What is the most probable sequence $\{Y_1, \ldots, Y_T\}$ given observations $\{X_1 = x_1, \ldots, X_T = x_T\}$?

The Trellis

- X-Axis = time
- Y-Axis = state variable (R_t)
- Node = a particular state at a particular time
- Edge = possible transition from R_{t-1} to R_t



Viterbi Algorithm: Key concepts

Nodes and edges have numbers attached to them:

- **Edge Probability**: Probability of taking that transition, and then generating the next observed output

$$e_{ijt} = P(R_t = j, U_t = u_t | R_{t-1} = i)$$

- **Node Probability**: Probability of the best path until node j at time t

$$v_{jt} = \max_{r_1,\ldots,r_{t-1}} P(U_1 = u_1 \ldots, U_t = u_t, R_1 = r_1, \ldots, R_t = j)$$

Viterbi Algorithm: the iteration step

Given edge probabilities defined as

$$e_{i,j,t} = P(R_t = j, U_t = u_t | R_{t-1} = i)$$

and node probabilities defined as

$$v_{j,t} = \max_{r_1,\dots,r_{t-2},i} P(U_1 = u_1 \dots, U_t = u_t, R_1 = r_1, \dots, R_{t-1} = i, R_t = j)$$

The node probability can be efficiently computed as

$$v_{j,t} = \max_i v_{i,t-1} e_{i,j,t}$$

Parts of speech

Most modern English dictionaries use these POS tags.

- **Open-class words** (anybody can make up a new word, in any of these classes, at any time): nouns, verbs, adjectives, adverbs, interjections
- **Closed-class words** (it's hard to make up a new word in these classes): pronouns, prepositions, conjunctions, determiners

Most published, tagged data use POS tags that are finer-grained than the nine tags listed above. For example, the next few slides describe the Penn Treebank POS tag set.

Why do POS tagging?

- Because it's highly accurate, typically 97%. That means you can run a POS tagger as a pre-processing step, before doing harder natural language understanding tasks.
- Because it's necessary, if you want to know what the words in the sentence mean.

Will Will ? Will will . Will will will Will 's will to Will .

MD NNP SYM NNP MD SYM NNP MD VB NNP POS NN TO NNP SYM

Viterbi algorithm key formulas

Initial Node Probability:

$$\log v_{j,1} = \log \pi_i + \log b_{j,x_1}$$

Edge Probability:

$$\log e_{ijt} = \log a_{ij} + \log b_{j,x_t}$$

Node Probability:

$$\log v_{j,t} = \max_i \left(\log v_{i,t-1} + \log e_{ijt} \right)$$

Backpointer:

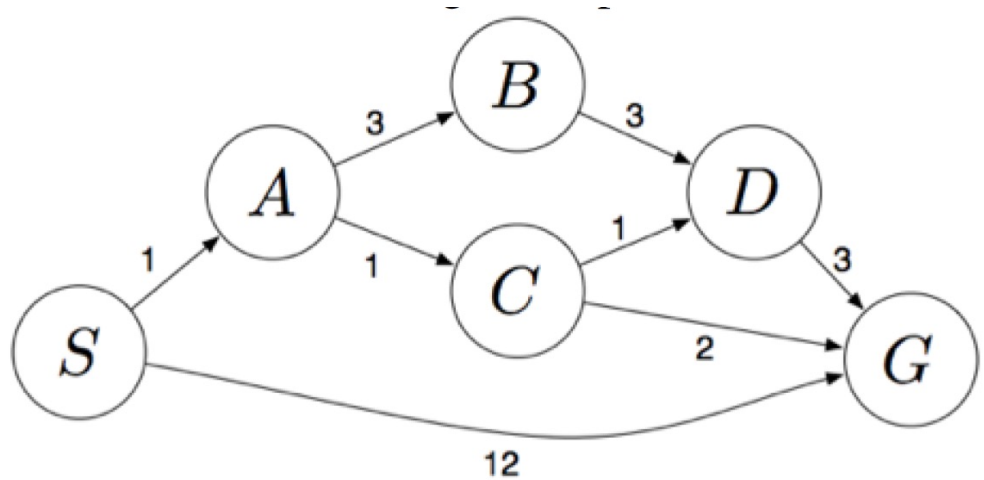
$$i^*_{j,t} = \operatorname{argmax}_i \left(\log v_{i,t-1} + \log e_{ijt} \right)$$

Some sample problems

- Search
- BN
- HMM

Sample problem: Search

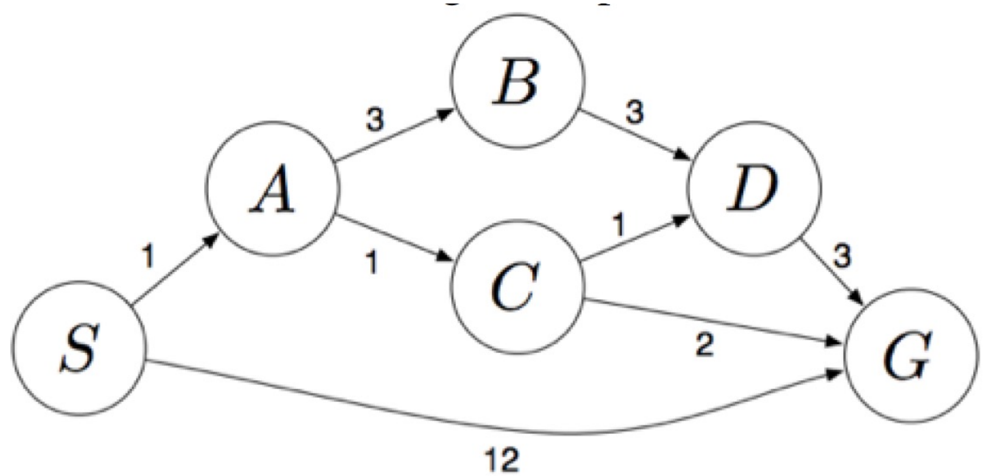
(Assume ties are resolved in alphabetical order)



- What path would BFS return? Answer: SG
  - What states would be expanded? Answer: S, A, and G
- What path would DFS return? Answer: SABDG
  - What states would be expanded? Answer: S, A, B, D, and G
- What path would UCS return? Answer: SACG
  - What states would be expanded? Answer: S, A, C, D, B, and G

Sample problem: Search

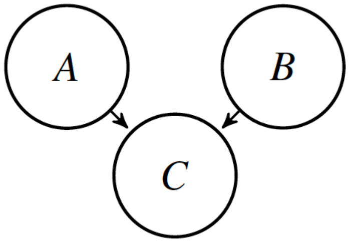
(Assume ties are resolved in alphabetical order)



State	H1	H2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

- Find the smallest possible modification that makes h1 admissible
  - Answer: h1(S)=4
- After your modification, which of these two heuristics will result in the fastest run-time for A* search?
  - Answer: h1, because it still dominates h2
- Find the smallest possible modification that makes h2 both admissible and consistent
  - Answer: h2(S)=3

Sample problem: Bayesian network



P(A=T)=0.4,
P(B=T)=0.1, and

A, B	P(C=T|A,B)
F,F	0.7
F,T	0.7
T,F	0.1
T,T	0.9

What is P(A=T|B=T,C=T)?

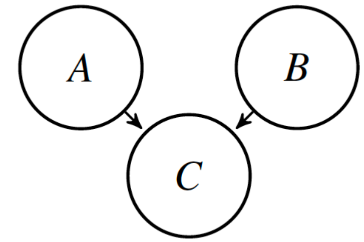
Answer:

$$P(A=T|B=T,C=T)=\frac{P(A=T,B=T,C=T)}{P(B=T,C=T)}$$

$$=\frac{P(A=T)P(B=T)P(C=T|A=T,B=T)}{\sum_{a=T}^{T}P(A=a)P(B=T)P(C=T|A=a,B=T)}$$

$$=\frac{(0.4)(0.1)(0.9)}{(0.4)(0.1)(0.9)+(0.6)(0.1)(0.7)}$$

Sample problem: Bayesian network



$P(A{=}T){=}0.4$, $P(B{=}T){=}0.1$, and

A, B	P(C=T|A, B)
F,F	0.7
F,T	0.7
T,F	0.1
T,T	0.9

Day	A	B	C
1	T	T	F
2	T	F	T
3	F	T	T
4	T	T	T
5	?	T	T

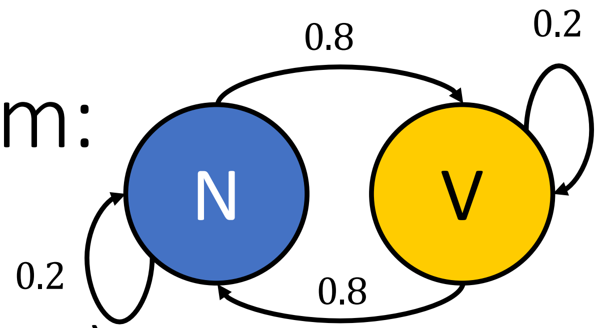
Suppose we have a series of observations with one missing value for A, as shown. What is the expected number of days on which A is true?

Answer:

$$3 + P(A = T | B = T, C = T)$$

$$= 3 + \frac{(0.4)(0.1)(0.9)}{(0.4)(0.1)(0.9) + (0.6)(0.1)(0.7)}$$

Sample problem: HMM



E_t	rose	bill	likes
$P(E_t \mid X_t = N)$	0.4	0.4	0.2
$P(E_t \mid X_t = V)$	0.2	0.2	0.6

- Find P(X2=V|E1=bill, E2=rose)

Solution: Using the forward algorithm, we can compute the joint probabilities as

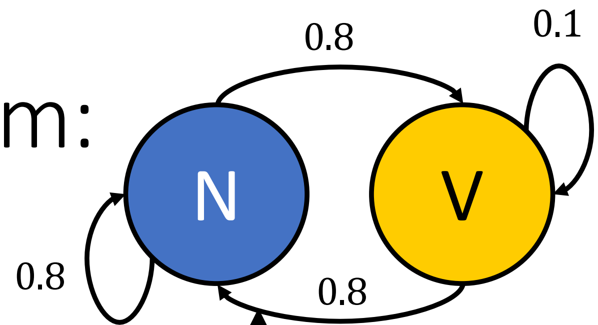
$$\begin{aligned}
P(E, X_2 = V) &= P(X_1 = N, E_1, X_2 = V, E_2) + P(X_1 = V, E_1, X_2 = V, E_2) \\
&= (0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2) \\
P(E, X_2 = N) &= P(X_1 = N, E_1, X_2 = N, E_2) + P(X_1 = V, E_1, X_2 = N, E_2) \\
&= (0.8)(0.4)(0.1)(0.4) + (0.2)(0.2)(0.9)(0.4)
\end{aligned}$$

Dividing the first row by the sum of the two rows, we get

$$P(X_2 = V \mid E) = \frac{(0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2)}{(0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2) + (0.8)(0.4)(0.1)(0.4) + (0.2)(0.2)(0.9)(0.4)}$$

Sample problem: HMM

- Find the most likely state sequence



E_t	rose	bill	likes
$P(E_t \mid X_t = N)$	0.4	0.4	0.2
$P(E_t \mid X_t = V)$	0.2	0.2	0.6