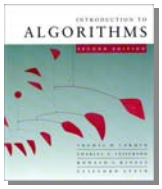
Introduction to Algorithms

6.046J/18.401J



LECTURE 16

Hidden Markov Models I

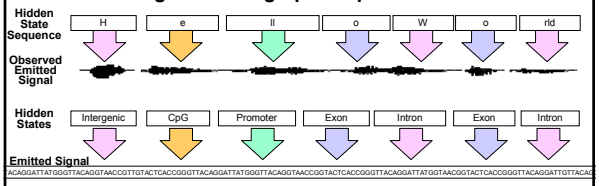
- Motivation and applications
- Defs: Markov Chains HMMs
- Example: the dishonest casino
- $P(x,\pi)$ likelihood calculation
- Viterbi: max likelihood parse π^*
- Forward: total lik. $P(x)$, all paths

Prof. Manolis Kellis

Applications of HMMs: Modeling sequential data

- **Temporal pattern recognition**
  - Speech recognition
  - Handwritten character recognition
  - Body motion and activity recognition
  - Gesture recognition for machine interface
- **Sequential symbolic data**
  - Text and article parsing and interpretation
  - Machine translation
  - Musical score following
- **Computational Biology**
  - Gene structure identification
  - Protein structural domains
  - Protein evolutionary patterns
  - Modeling protein families

HMMs are 'generative' graphical probabilistic models



- Ability to **emit** sequences of a certain ***type***
  - Not exact copy of any one previously-seen example
  - Preserving 'properties' of **type**, not identical sequence
- Ability to **recognize** sequences of a certain type (state)
  - What (hidden) state is most likely to have generated observations
  - Find set of states and transitions that generated a long sequence
- Ability to **learn** distinguishing characteristics of each state
  - Training our generative models on large datasets
  - Learn to classify unlabelled data

Markov Chains and Hidden Markov Models

Definitions: Markov Chain

Definition: A ***Markov chain*** is a triplet (Q, p, A), where:

- Q a finite set of states, each corresponds to a symbol in alphabet Σ
- p is the initial state probabilities.
- A is the state transition probabilities, denoted a_{st} for each s, t in Q.
- For each s, t in Q the transition probability is: $a_{st} \equiv P(x_i = t | x_{i-1} = s)$

Output: The output of the model is the set of states at each instant time → ***the set of states is observable***

Key Property: The probability of each symbol x_i depends only on the value of the preceding symbol x_{i-1} : $P(x_i | x_{i-1}, \ldots, x_1) = P(x_i | x_{i-1})$

Implication: The probability of the sequence:

$P(x) = P(x_L, x_{L-1}, \ldots, x_1) = P(x_L | x_{L-1}) P(x_{L-1} | x_{L-2}) \ldots P(x_2 | x_1) P(x_1)$

Definitions: HMM (Hidden Markov Model)

Definition: An ***HMM*** is a 5-tuple **(*Q, V, p, A, E*)**, where:

- ***Q*** is a finite set of states, **|Q|=N**
- **V** is a finite set of observation symbols per state, **|V|=M**
- ***p*** is the initial state probabilities.
- ***A*** is the state transition probabilities, denoted by a_{st} for each ***s, t*** in ***Q.***
  - For each ***s, t*** in ***Q*** the transition probability is: $a_{st} \equiv P(x_i = t | x_{i-1} = s)$
- **E** is a probability emission matrix, $e_{sk} \equiv P(v_k \text{ at time } t \mid q_t = s)$

Output: Only emitted symbols are observable by the system but not the underlying random walk between states -> "hidden"

Property: Emissions and transitions are dependent on the current state only and not on the past.

The three main questions on HMMs

1. *Decoding*

GIVEN a HMM M (e_i,a_{ij}), and a sequence x,
FIND the sequence π of states that maximizes P[x, π | M]
When was he using a fair die, when was he using a loaded die?
What portion of the sequence was generated by each die?

→ Viterbi algorithm

2. *Evaluation*

GIVEN a HMM M (e_i,a_{ij}), and a sequence x,
FIND Prob[x | M]
Is my model for how the casino works a good representation of reality?
How likely is the sequence of emissions given my model parameters?
(and irrespective of the optimal path, i.e. over all possible paths)

→ Forward algorithm

3. *Learning*

GIVEN a HMM M (?,?), and a sequence x,
(unspecified transition/emission probs)
FIND parameters θ = (e_i(.), a_{ij}) that maximize P[x | θ]
How "loaded" is the loaded die? How "fair" is the fair die?
How often does the casino player change from fair to loaded, and back?

→ Expectation Maximization

The corresponding algorithms for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \Sigma_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^\wedge = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

A simple example of HMMs
The dishonest casino

Example: The Dishonest Casino

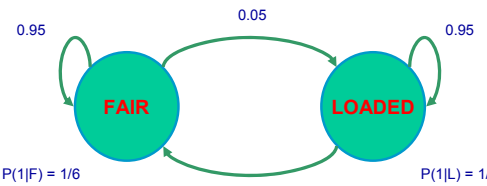
A casino has two dice:
- Fair die
  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- Loaded die
  P(1) = P(2) = P(3) = P(5) = 1/10
  P(6) = 1/2

Casino player switches between fair and loaded die on average once every 20 turns

Game:
1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Question: *Is the Casino player using a fair die?*
More generally: *When to bet, when to stop, when to get up and call management!*

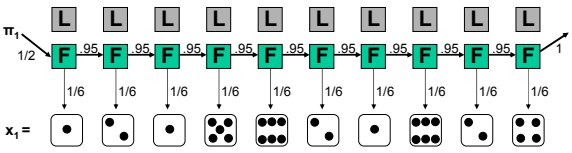
The dishonest casino model



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

When the path is known: P (sequence x_1 , path π_1)



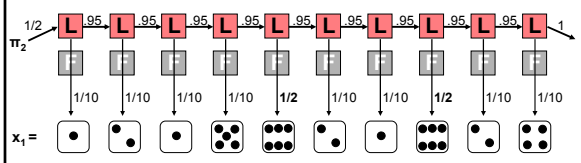
What is the likelihood of
π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair
and rolls
x = 1 , 2, 1, 5, 6, 2, 1, 6, 2, 4

emission transition emission transition emission

$p = ½ \times P(1 \mid \text{Fair}) P(\text{Fair}_{i+1} \mid \text{Fair}_i) P(2 \mid \text{Fair}) P(\text{Fair} \mid \text{Fair}) \ldots P(4 \mid \text{Fair})$
$= ½ \times (1/6)^{10} \times (0.95)^9$
$= 5.2 \times 10^{-9}$

Q: Why is P(x|π) so small?
A: Because all seqs are equally likely, and there's lots and lots of them! (1/6)^10=10^{-8}

When the path is known: P (sequence x_1 , path π_2)



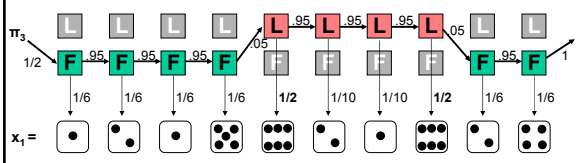
What is the likelihood of
π = Load, Load, Load, Load, Load, Load, Load, Load, Load, Loaded
and rolls
x = 1 , 2, 1, 5, 6, 2, 1, 6, 2, 4

emission transition emission transition emission

$p = ½ \times P(1 \mid Load)\ P(Load_{i+1} \mid Load_i)\ P(2 \mid Load)\ P(Load \mid Load) \ldots P(4 \mid Fair)$
$= ½ \times (1/10)^8 \times (1/2)^2\ (0.95)^9$
$= 7.9 \times 10^{-10}$

Even smaller!
(If indeed the dice was loaded all the way, then we would have expected more 6's)

When the path is known: P (sequence x_1 , path π_3)



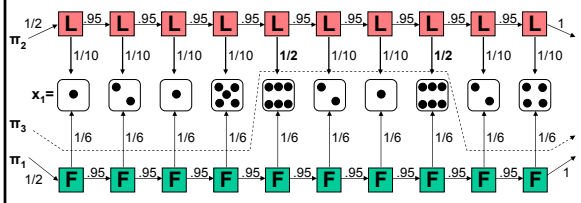
What is the likelihood of
π = Fair, Fair, Fair, Fair, Load, Load, Load, Load, Fair, Fair
and rolls
x = 1 , 2, 1, 5, 6, 2, 1, 6, 2, 4

emission transition emission transition emission

$p = ½ \times P(1 \mid Fair)\ P(Fair_{i+1} \mid Fair_i)\ P(2 \mid Fair)\ P(Fair \mid Fair) \ldots P(4 \mid Fair)$
$= ½ \times (1/10)^2 \times (1/2)^2 \times (1/6)^6 \times (0.95)^7 \times (0.05)^2$
$= 4.6 \times 10^{-11}$

Even smaller! (Having two switches is highly unlikely in so few rolls. Only expect 1 in 20 on average).

Comparing paths to find the most likely one



We calculated the likelihood of three state paths:

π_1: P(x, all-Fair)	$= 0.5 \times 10^{-9}$	(very small)
π_2: P(x, all-Loaded)	$= 7.9 \times 10^{-10}$	(very very small)
π_3: P(x, switched twice)	$= 4.6 \times 10^{-11}$	(very very very small)

➔ Honest!

Likelihood ratios:

P(x, π_1) vs. P(x, π_2):	7	times more likely
P(x, π_2) vs. P(x, π_3):	17	times more likely
P(x, π_1) vs. P(x, π_3):	111	times more likely than

π_1 gives highest P(x,π_i)
Assuming uniform priors, π_3 is most likely path.

Bayes $P(\pi_i|x) = P(\pi_i, x) / P(x)$

Emitted sequence implies most likely state space

A different sequence of rolls:
x_2 = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

The likelihood for π_1 = F, F, …, F?
$P(x_2,\pi_1) = ½ \times (1/6)^{10} \times (0.95)^9 = 0.5 \times 10^{-9}$, as before

The likelihood for π_2 = L, L, …, L?
$P(x_2,\pi_2) = ½ \times (1/10)^4 \times (1/2)^6\ (0.95)^9 = 0.5 \times 10^{-7}$

So, it is 100 times more likely the die is loaded ➔ **Dishonest!**

More generally, how do we find most likely path **π***, over all paths!

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

1. DECODING:
What was the sequence of hidden states?

Given: Model parameters $e_i(.)$, a_{ij}
Given: Sequence of emissions x

Find: Sequence of hidden states π

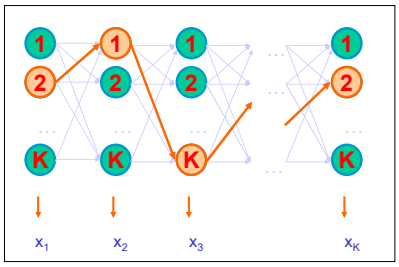
Finding the optimal path

- We can now evaluate any path through hidden states, given the emitted sequences
- How do we find the best path?
- Optimal substructure! Best path through a given state is:
  - Best path to previous state
  - Best transition from previous state to this state
  - Best path to the end state

➔ Viterbi algorithm
- Define $V_k(i)$ = Probability of the most likely path through state π_i=k
- Compute $V_k(i+1)$ as a function of $\max_{k'} \{ V_{k'}(i) \}$
- $V_k(i+1) = e_k(x_{i+1}) * \max_j a_{jk} V_j(i)$

➔ Dynamic Programming

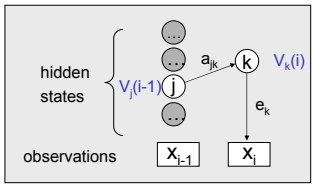
Finding the most likely path



- Find path π^* that maximizes total joint probability P[x, π]
- $P(x,\pi) = a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i \pi_{i+1}}$

start emission transition

Calculate maximum P(x,π) recursively

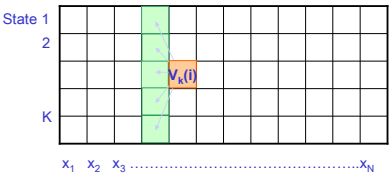


- Assume we know V_j for the previous time step (i-1)
- Calculate $V_k(i) = e_k(x_i) * \max_j (V_j(i-1) \times a_{jk})$

current max | this emission | max ending in state j at step i | Transition from state j

all possible previous states j

The Viterbi Algorithm



Input: x = x1……xN

Initialization:
$V_0(0)$=1, $V_k(0)$ = 0, for all k > 0

Iteration:
$V_k(i) = e_K(x_i) \times \max_j a_{jk} V_j(i-1)$

Termination:
$P(x, \pi^*) = \max_k V_k(N)$

Traceback:
Follow max pointers back
Similar to aligning states to seq

In practice:
Use log scores for computation

Running time and space:
Time: $O(K^2N)$
Space: $O(KN)$

The six algorithmic settings for HMMs

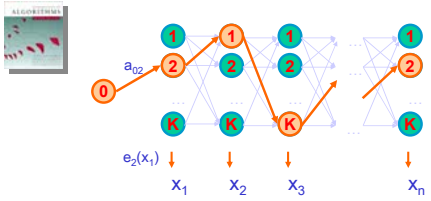
	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \sum_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

2. EVALUATION
(how well does our model capture the world)

Given: Model parameters $e_i(.)$, a_{ij}
Given: Sequence of emissions x

Find: P(x|M), summed over all possible paths π

Simple: Given the model, generate some sequence x

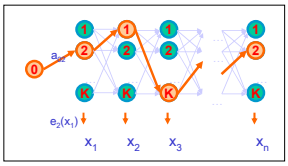


Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi 1}$
2. Emit letter x_1 according to prob $e_{\pi 1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi 1 \pi 2}$
4. … until emitting x_n

We have some sequence x that can be emitted by p. Can calculate its likelihood.
However, in general, many different paths may emit this same sequence x.
How do we find the total probability of generating a given x, over any path?

Complex: Given x, was it generated by the model?



Given a sequence x,

What is the probability that x was generated by the model (using any path)?

– $P(x) = \sum_{\pi} P(x,\pi)$

- Challenge: exponential number of paths

Calculate probability of emission over all paths

- Each path has associated probability
  - Some paths are likely, others unlikely: sum them all up
  - → Return total probability that emissions are observed, summed over all paths
  - Viterbi path is the most likely one
    - How much 'probability mass' does it contain?
- (cheap) alternative:
  - Calculate probability over maximum (Viterbi) path π*
  - Good approximation is Viterbi has highest density
  - BUT: incorrect
- (real) solution
  - Calculate the exact sum iteratively
    - $P(x) = \sum_{\pi} P(x,\pi)$
  - Can use dynamic programming

The Forward Algorithm – derivation

Define the forward probability:

$f_l(i) = P(x_1 \ldots x_i, \pi_i = l)$

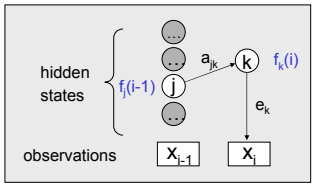
$= \Sigma_{\pi 1 \ldots \pi i-1} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}, \pi_i = l)\, e_l(x_i)$

$= \Sigma_k \boxed{\Sigma_{\pi 1 \ldots \pi i-2} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}=k)}\, a_{kl}\, e_l(x_i)$

$= \Sigma_k \boxed{f_k(i-1)}\, a_{kl}\, e_l(x_i)$

$= e_l(x_i)\, \Sigma_k \boxed{f_k(i-1)}\, a_{kl}$

Calculate total probability $\Sigma_{\pi} P(x,\pi)$ recursively

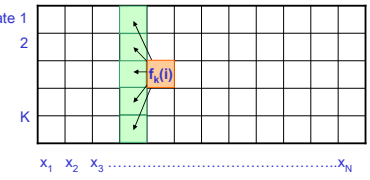


- Assume we know f_j for the previous time step (i-1)
- Calculate $f_k(i)$ = $e_k(x_i)$ * sum_j ($f_j(i-1)$ × a_{jk})

updated sum | this emission | sum ending in state j at step i | transition from state j

every possible previous state j

The Forward Algorithm



Input: x = x1……xN

Initialization:

$f_0(0)=1$, $f_k(0) = 0$, for all k > 0

Iteration:

$f_k(i) = e_K(x_i) \times sum_j\, a_{jk}\, f_j(i-1)$

Termination:

$P(x, \pi^*) = sum_k\, f_k(N)$

In practice:

Sum of log scores is difficult
→ approximate exp(1+p+q)
→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$
Space: O(KN)

What have we learned ?

- Modeling sequential data
  - Recognize a ***type*** of sequence, written, spoken, evolved
- Simple example
  - The dishonest casino
- Definitions
  - Markov Chains
  - Hidden Markov Models (HMMs)
- Our first computations
  - Running the model: know model → generate sequence of a 'type'
  - Evaluation: know model, emissions, states → p?
  - Viterbi: know model, emissions → find optimal path
  - Forward: know model, emissions → total p over all paths
- Next time:
  - Posterior decoding
  - Supervised learning
  - Unsupervised learning: Baum-Welch, Viterbi training

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^\wedge = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths