6.047/6.878/HST.507
Computational Biology: Genomes, Networks, Evolution

Lecture 4

Modeling Biological Sequences using Hidden Markov Models

Module 1: Aligning and modeling genomes

- Module 1: Computational foundations
  - Dynamic programming: exploring exponential spaces in poly-time
  - Linear-time string matching, Hashing, Content-based indexing
  - Hidden Markov Models: decoding, evaluation, parsing, learning
- Last week: Sequence alignment / comparative genomics
  - Local/global alignment: infer nucleotide-level evolutionary events
  - Database search: scan for regions that may have common ancestry
- This week: Modeling genomes / exon / CpG island finding
  - Modeling class of elements, recognizing members of a class
  - Application to gene finding, conservation islands, CpG islands

We have learned how to align sequences to other sequences

- L2: Sequence alignment
  - Dynamic programming, duality path ⇔ alignment
  - Global / local alignment, general gap penalties
- L3: Rapid string search
  - Exact string match, semi-numerical matching
  - Database search: Hashing, BLAST, variations
- L15:Comparative genomics: evolutionary signatures
  - Tell me how you evolve, I'll tell you what you are
  - Identifying conserved elements through evolution
- L16: Whole-genome assembly/alignment/duplication:
  - Finding all common substrings within/across species
  - Contigs/scaffolds, string graphs, glocal alignmt paths
- Problem set 1, project planning, Problem set 2 out

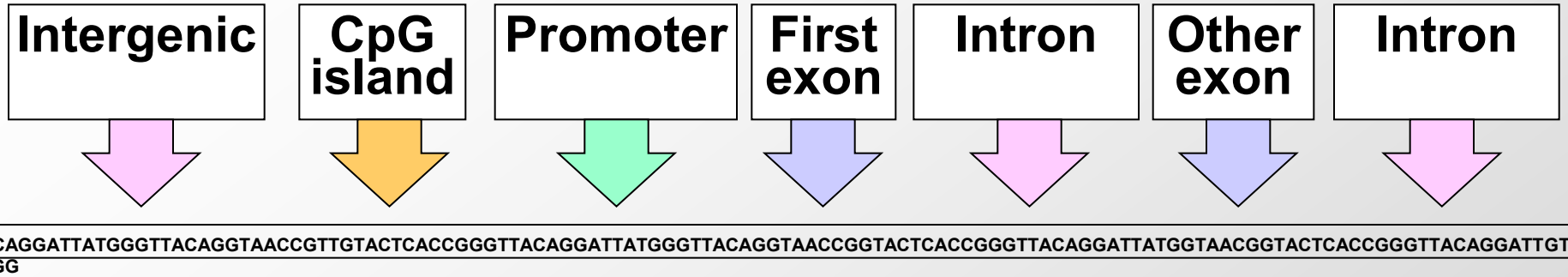
Today: apply these ideas to model DNA sequences

…GTACTCACCGGGTTACAGGATTATGGGTTACAGGTAACCGTT…

- What to do with a completely new piece of DNA
  - Align it to things we know about (database search)
  - Align it to things we don't know about (assembly)
- Stare at it
  - Non-standard nucleotide composition?
  - Interesting k-mer frequencies?
  - Recurrent patterns?
- Model it
  - Make some hypotheses about it
  - Build a 'generative model' to describe it
  - Find sequences of similar ***type***

➔ How do we model DNA sequences?

Modeling biological sequences with HMMs
(a.k.a. What to do with big unlabelled chunks of DNA)



- Ability to **emit** DNA sequences of a certain ***type***
  - Not exact alignment to previously known gene
  - Preserving ‘properties’ of **type**, not identical sequence
- Ability to **recognize** DNA 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

Why Probabilistic Sequence Modeling?

- Biological data is noisy
- Probability provides a calculus for manipulating models
- Not limited to yes/no answers – can provide "degrees of belief"
- Many common computational tools based on probabilistic models
- Our tools:
  - Markov Chains and Hidden Markov Models (HMMs)

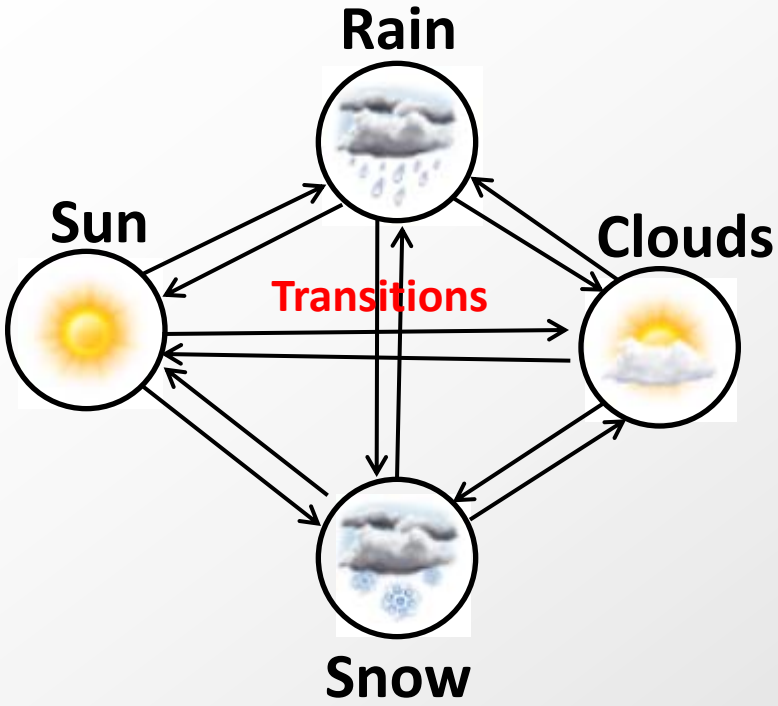
Markov Chains and Hidden Markov Models

Andrey Markov (1856-1922)

Image in the public domain.

Predicting tomorrow's weather

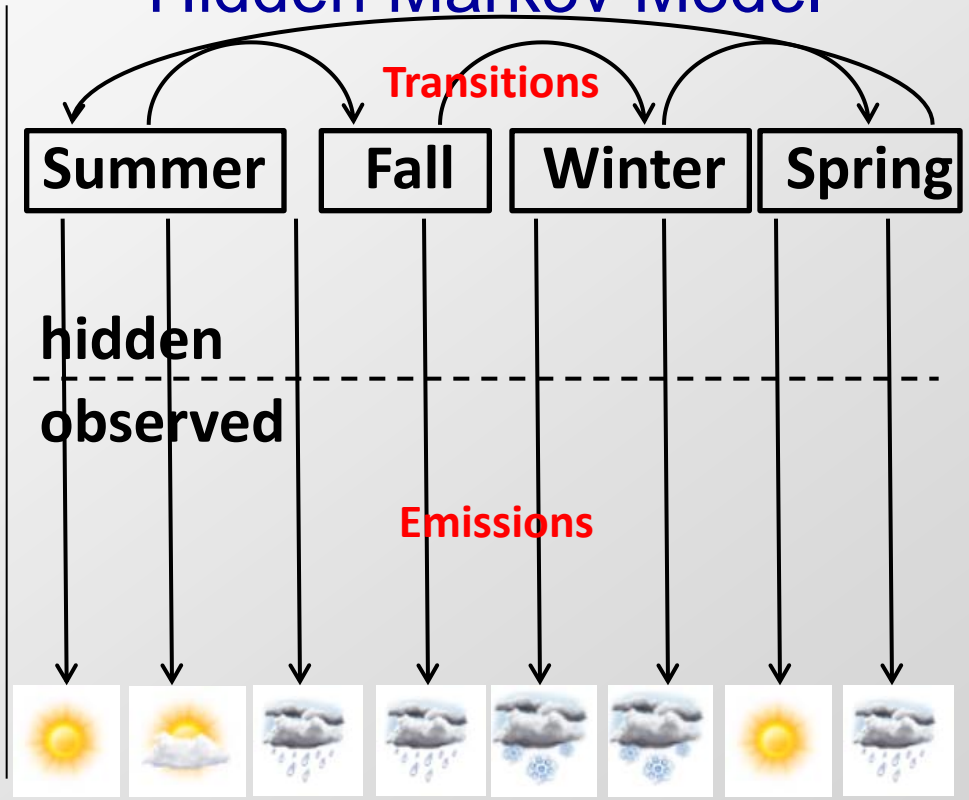
- Markov Chain



All observed

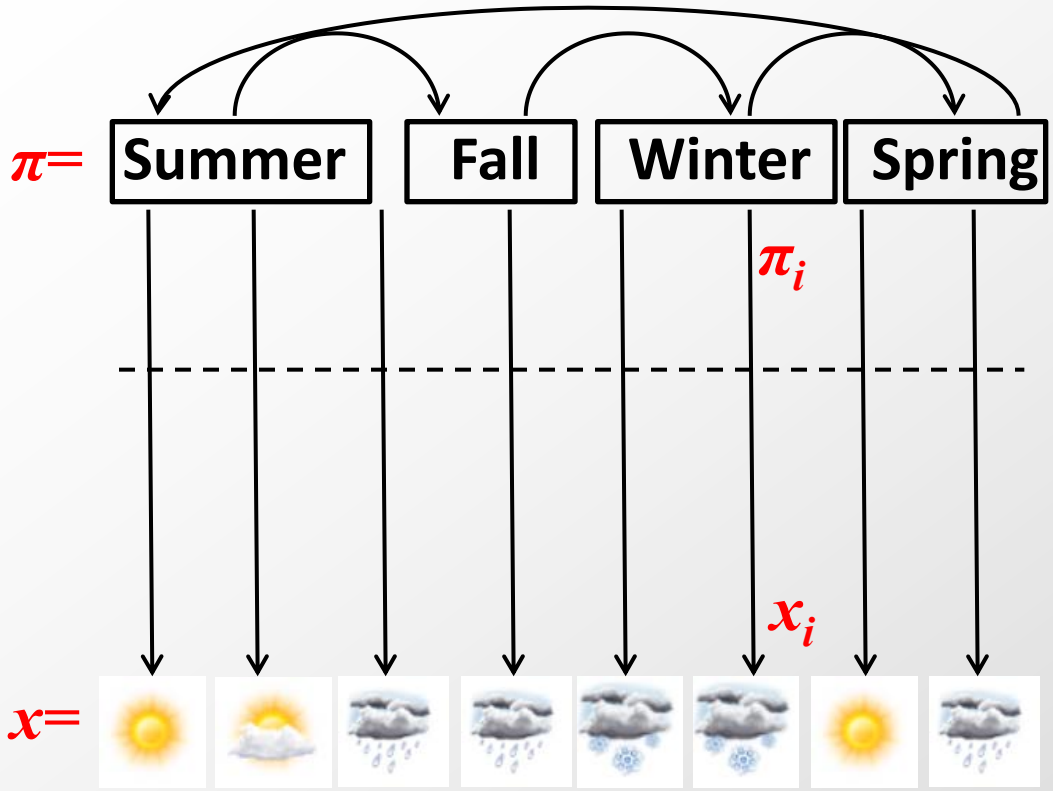
- What you see is what you get: next state only depends on current state (no memory)

- Hidden Markov Model



- Hidden state of the world (e.g. storm system) determines emission probabilities
- State transitions governed by a Markov chain

HMM nomenclature for this course



Transitions: $a_{kl}=P(\pi_i=l|\pi_{i-1}=k)$

Transition probability from state k to state l

Emissions: $e_k(x_i)=P(x_i|p_i=k)$

Emission probability of symbol x_i from state k

- Vector x = Sequence of observations
- Vector π = Hidden path (sequence of hidden states)
- Transition matrix A=a_{kl}=probability of $k \rightarrow l$ state transition
- Emission vector $E=e_k(x_i)$ = prob. of observing x_i from state k
- Bayes's rule: Use $P(x_i|\pi_i=k)$ to estimate $P(\pi_i=k|x_i)$

Components of a Markov Chain

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

- ***Q*** is a finite set of states. Each state corresponds to a symbol in the alphabet Σ
- ***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: $a_{st} \equiv P(x_i = t \mid x_{i-1} = s)$

Output: The output of the model is the set of states at each instant time => the set of states are observable

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

Formula: The probability of the sequence:

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

Slide credit: Serafim Batzoglou

Components of an 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: $a_{st} \equiv P(x_i = t \mid x_{i-1} = s)$
- **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.

Slide credit: Serafim Batzoglou

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 $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ 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

Examples of HMMs

The dishonest casino

The dishonest genome

… and many more

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(4) = P(5) = 1/10

  P(6) = 1/2

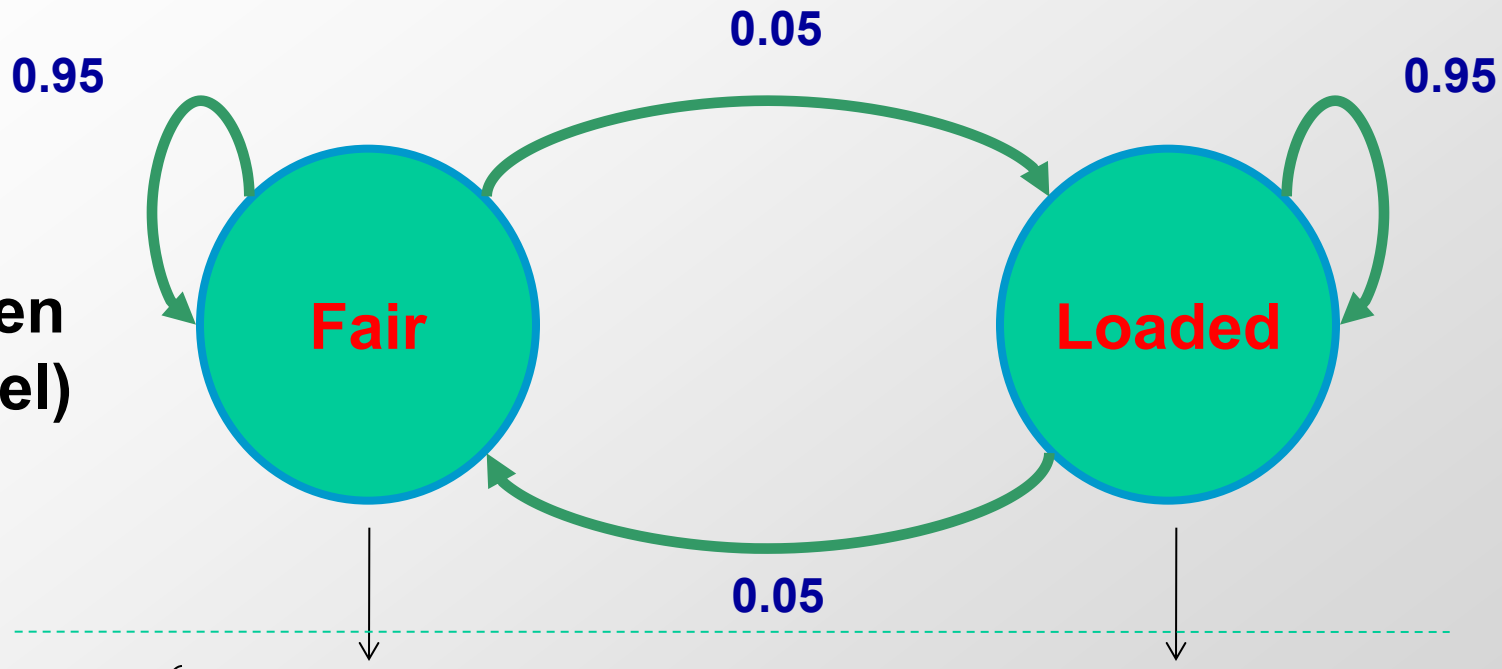
Casino player switches between fair and loaded die on average once every 20 turns

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

Slide credit: Serafim Batzoglou

The dishonest casino model



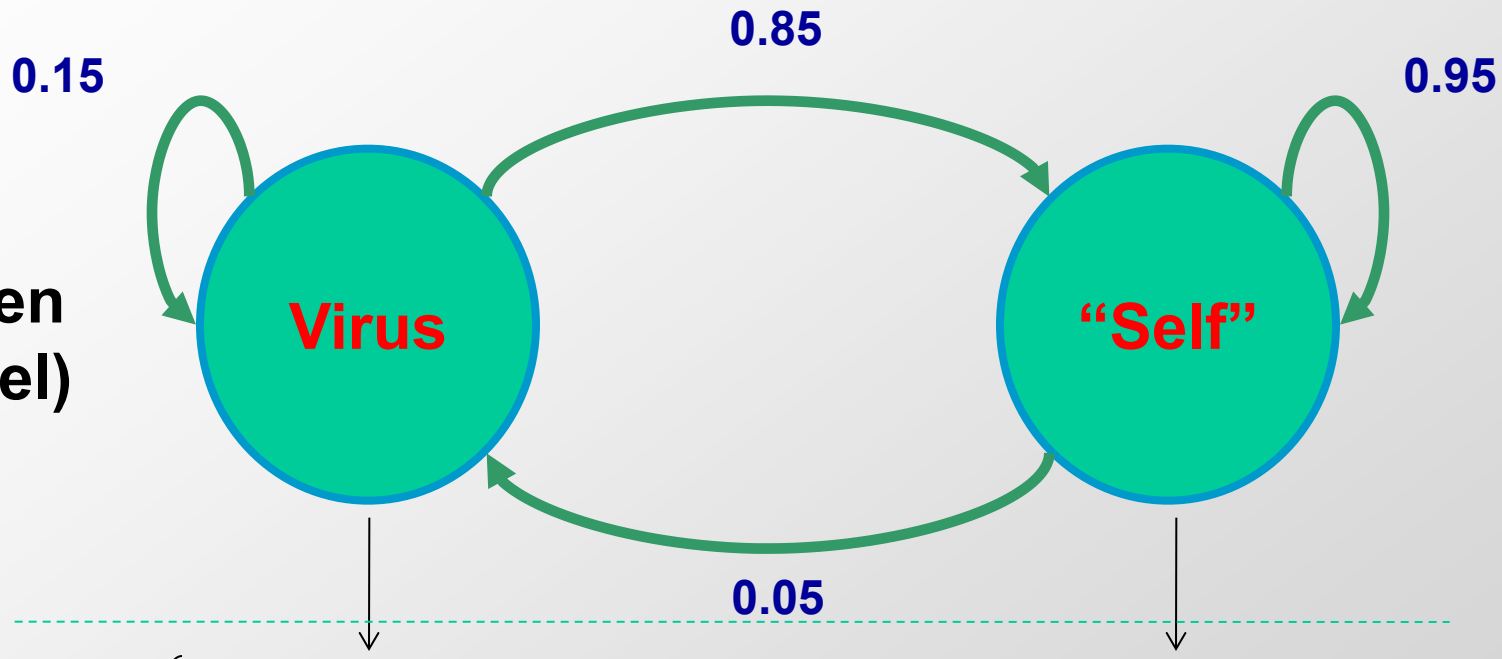
Hidden (model)

Observed (world)

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

Slide credit: Serafim Batzoglou

The dishonest genome model



Hidden (model)

Observed (world)

P(A|Virus) = 1/6
P(T|Virus) = 1/6
P(C|Virus) = 1/3
P(G|Virus) = 1/3

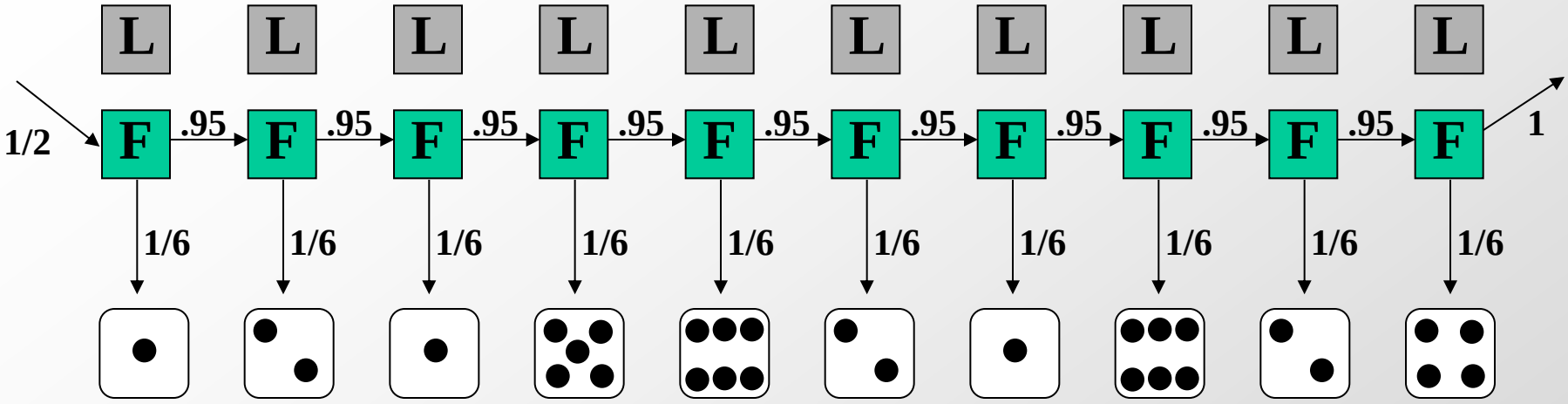
P(A|Self) = 1/4
P(T|Self) = 1/4
P(C|Self) = 1/4
P(G|Self) = 1/4

Slide credit: Serafim Batzoglou

Examples of HMMs for genome annotation

Application	Detection of GC-rich regions	Detection of conserved regions	Detection of protein-coding exons	Detection of protein-coding conservation	Detection of protein-coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	Nucleotide triplets, conservation levels	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

Running the model: Probability of a sequence



What is the joint probability of observing x and a specific path π:

π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair

and rolls

x = 1 , 2, 1, 5, 6, 2, 1, 6, 2, 4

Joined probability $P(x,\pi)=P(x|\pi)P(\pi)=P(\text{emissions}|\text{path})*P(\text{path})$

emission transition emission transition emission

$p = ½ \times P(1 \mid Fair)\ P(Fair_{i+1} \mid Fair_i)\ P(2 \mid Fair)\ P(Fair \mid Fair) \ldots P(4 \mid Fair)$

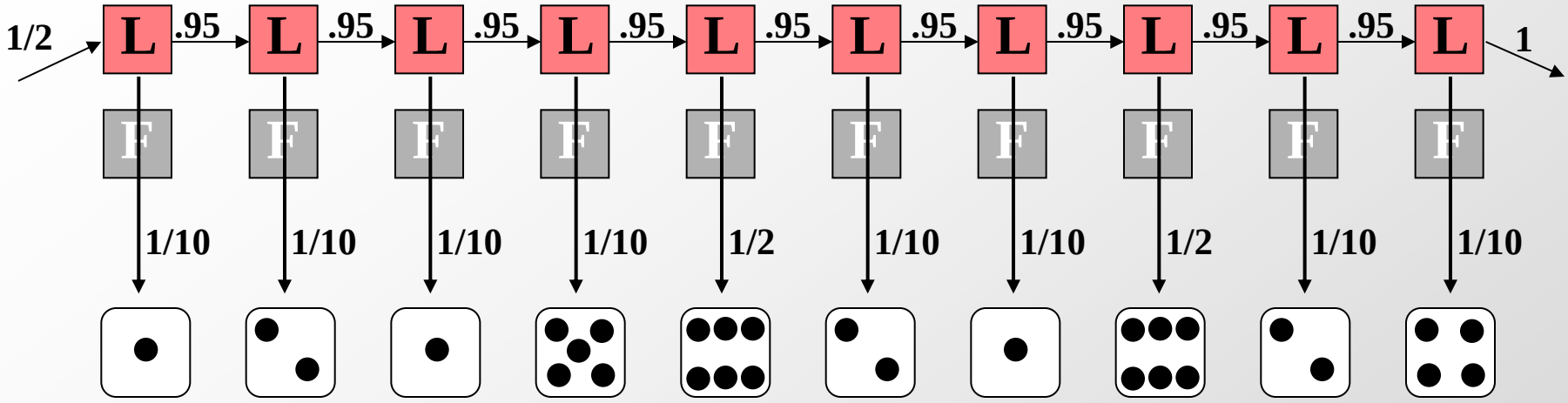
$= ½ \times (1/6)^{10} \times (0.95)^9$

$= 5.2 \times 10^{-9}$

Why is p so small?

Slide credit: Serafim Batzoglou

Running the model: Probability of a sequence



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

$p = ½ \times P(1 \mid \text{Load})\, P(\text{Load}_{i+1} \mid \text{Load}_i)\, P(2 \mid \text{Load})\, P(\text{Load}|\text{Load}) \ldots P(4 \mid \text{Fair})$

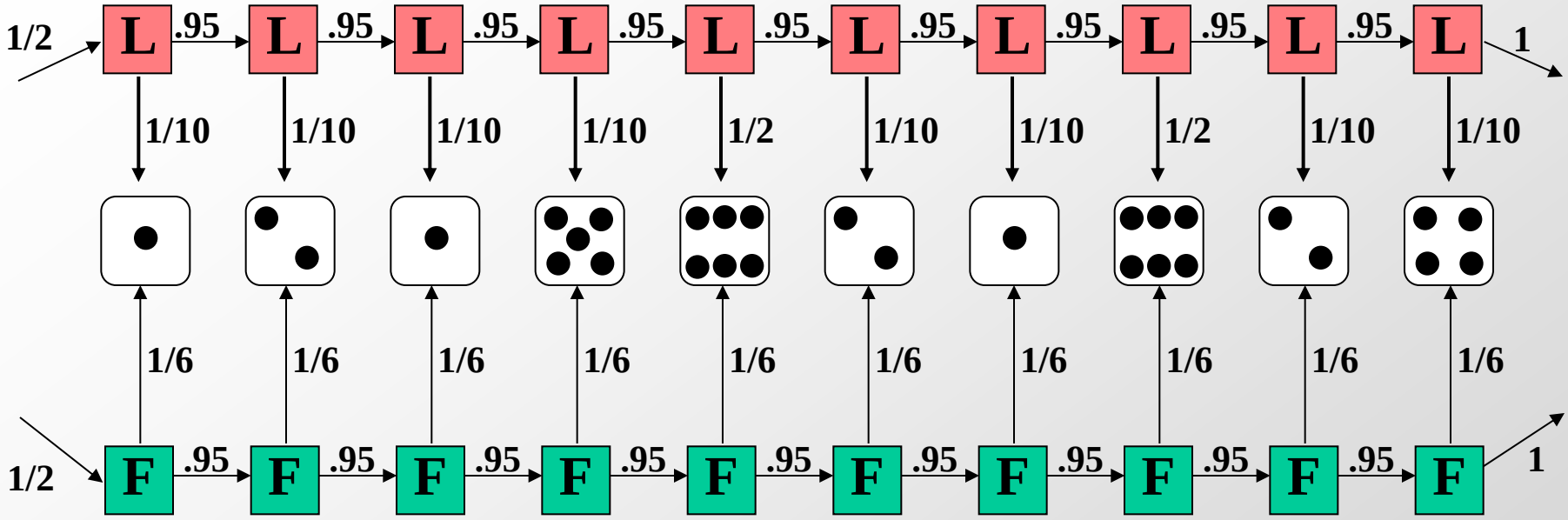
$= ½ \times (1/10)^8 \times (1/2)^2\, (0.95)^9$

$= 7.9 \times 10^{-10}$

Compare the two!

Slide credit: Serafim Batzoglou

Comparing the two paths



Two sequence paths:

P(x, all-Fair) = 5.2×10^{-9} (very small)

P(x, all-Loaded) = 7.9×10^{-10} (very very small)

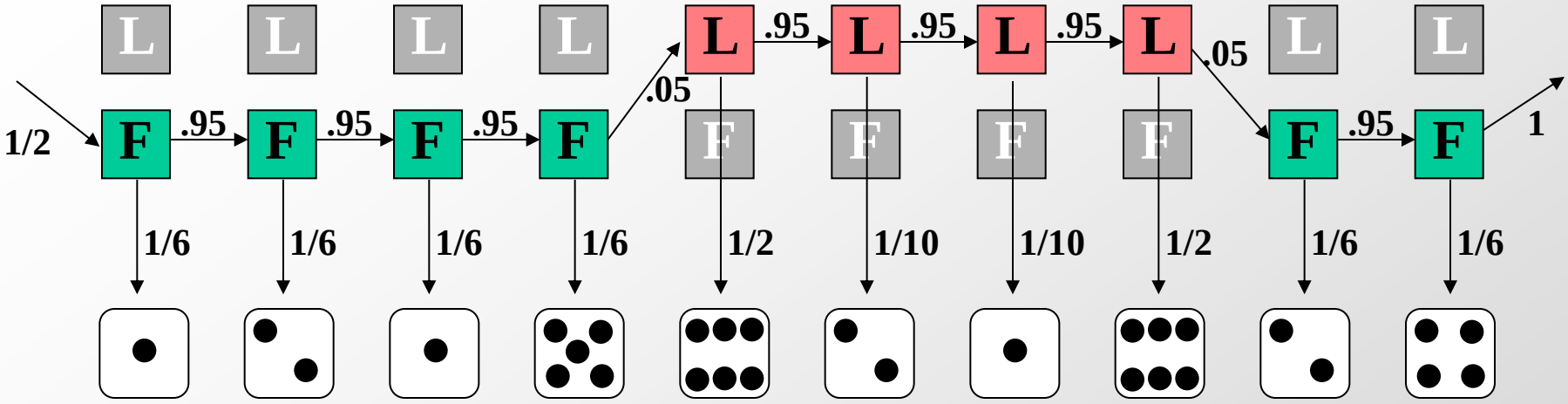
Likelihood ratio:

P(x, all-Fair) is 6.59 times more likely than P(x, all-Loaded)

It is 6.59 times more likely that the die is fair all the way, than loaded all the way.

Slide credit: Serafim Batzoglou

What about partial runs and die switching



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

$p = ½ \times P(1 \mid Fair)\ P(Fair_{i+1} \mid Fair_i)\ P(2 \mid Fair)\ P(Fair \mid Fair) \ldots P(4 \mid Fair)$

$= ½ \times (1/10)^2 \times (1/2)^2 \times (1/6)^5 \times (0.95)^7 \times (0.05)^2$

$= 2.8 \times 10^{-10}$

Much less likely, due to high cost of transitions

Model comparison

Let the sequence of rolls be:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

Now, what is the likelihood π = F, F, …, F?

$\frac{1}{2} \times (1/6)^{10} \times (0.95)^{9} = 0.5 \times 10^{-9}$, same as before

What is the likelihood π = L, L, …, L?

$\frac{1}{2} \times (1/10)^{4} \times (1/2)^{6} (0.95)^{9} = 0.5 \times 10^{-7}$

So, it is 100 times more likely the die is loaded

Model evaluation

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 $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ 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

3. 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

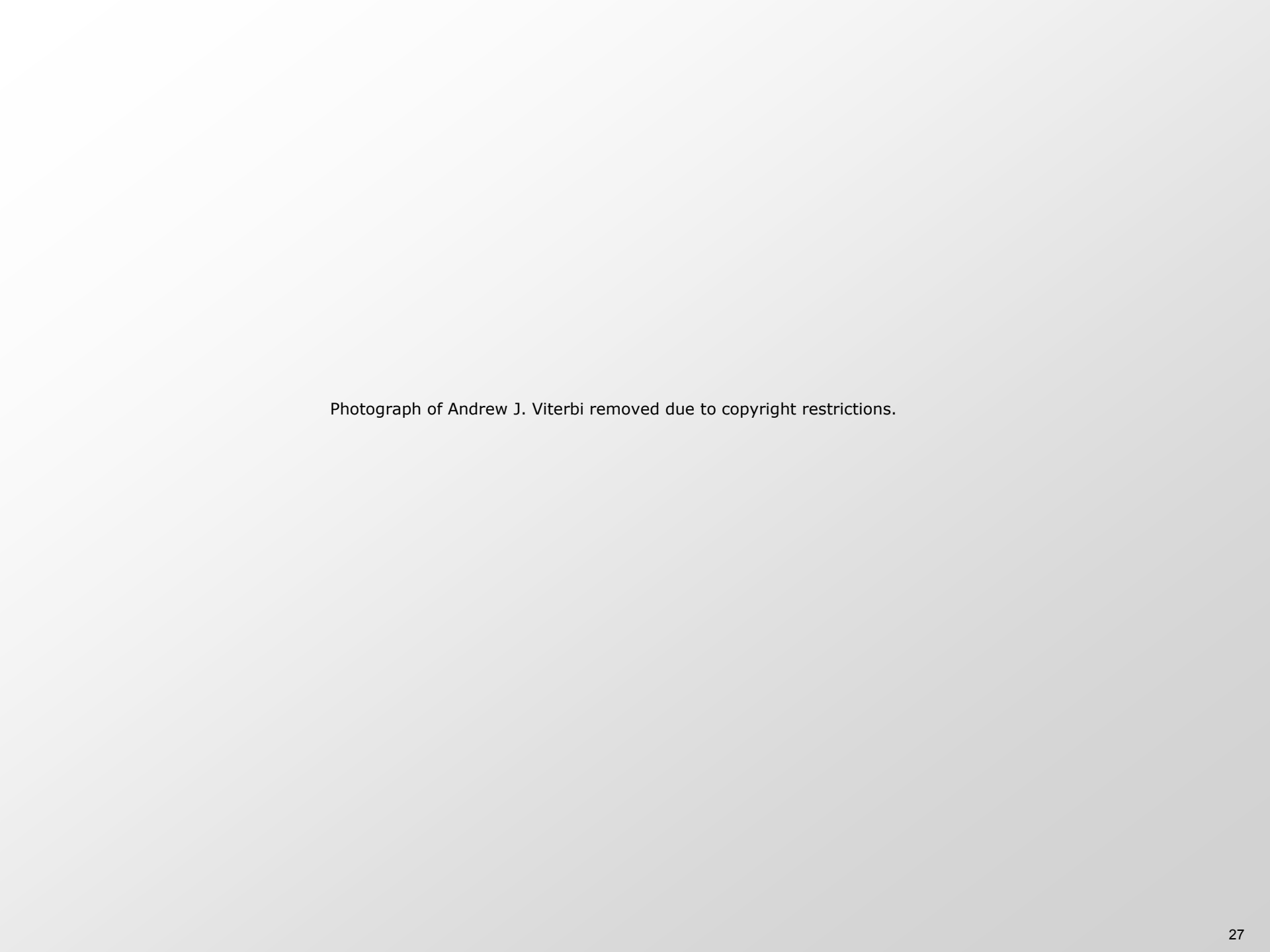
Finding the optimal path

- 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

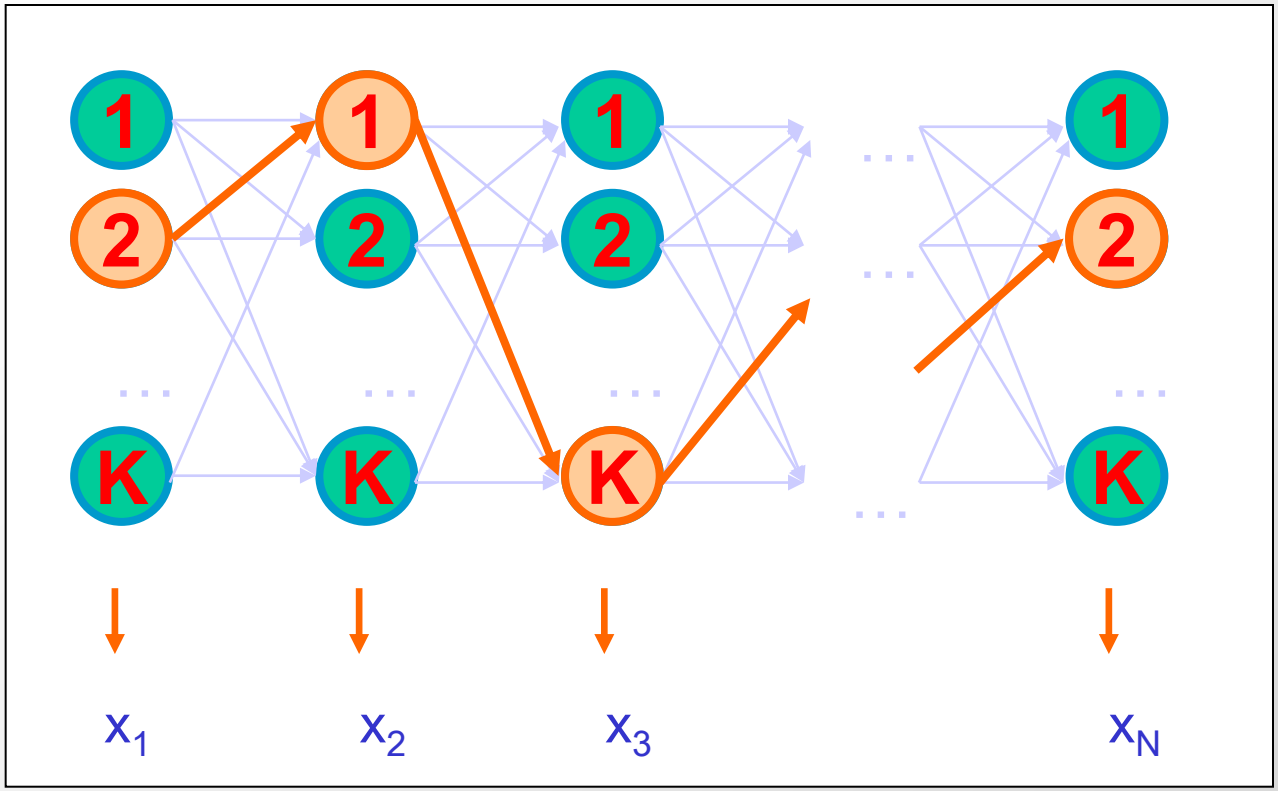
➔ Viterbi algortithm

- Define $V_k(i)$ = Probability of the most likely path through state $\pi_i=k$
- 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


Photograph of Andrew J. Viterbi removed due to copyright restrictions.

Finding the most likely path

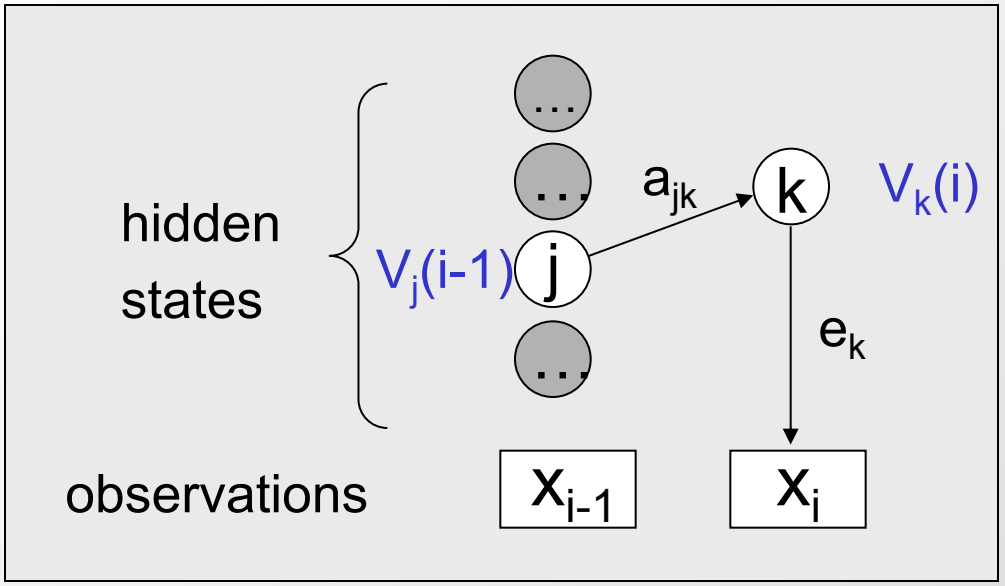


- Find path π^* that maximizes total joint probability $P[x, \pi]$
  - $P(x,\pi) = a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i \pi_{i+1}}$

start emission transition

Slide credit: Serafim Batzoglou

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

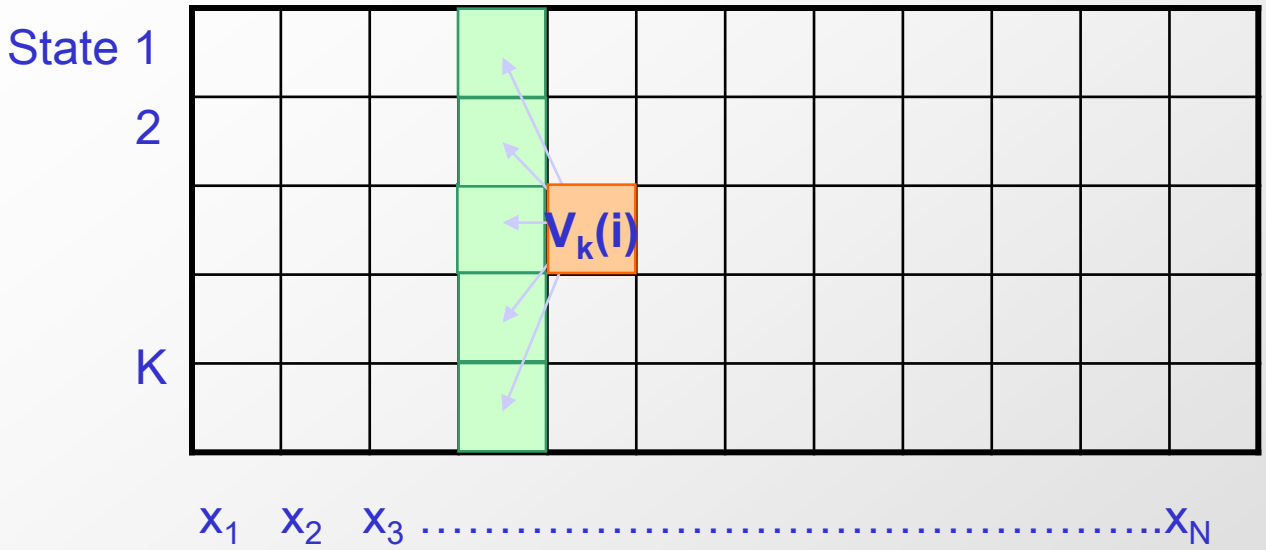
max ending in state j at step i

Transition from state j

all possible previous states j

Slide credit: Serafim Batzoglou

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: $O(KN)$

Slide credit: Serafim Batzoglou

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 $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ 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

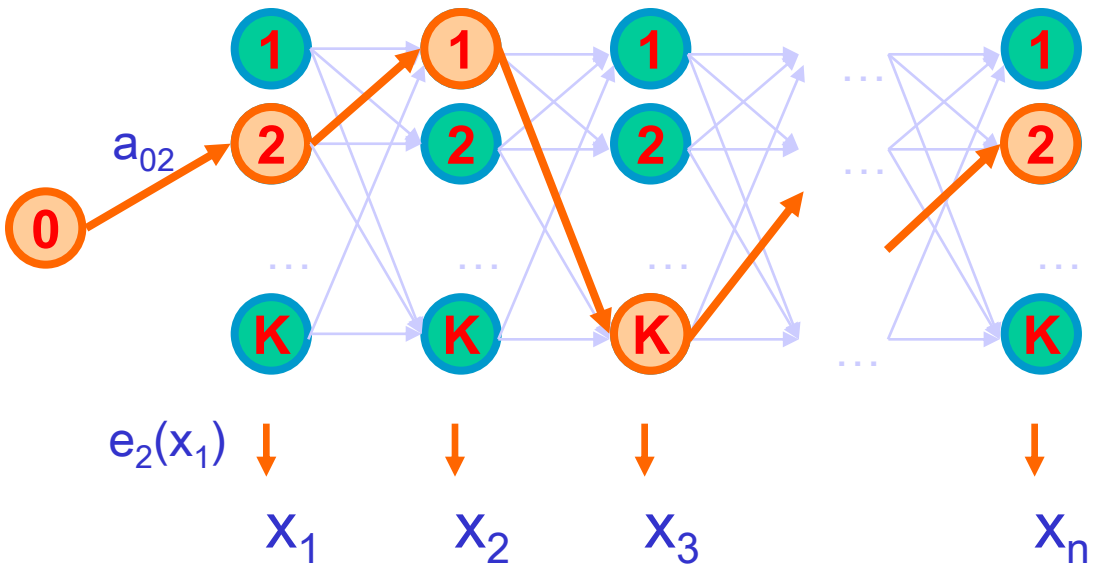
2. EVALUATION
(how well does our model capture the world)

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



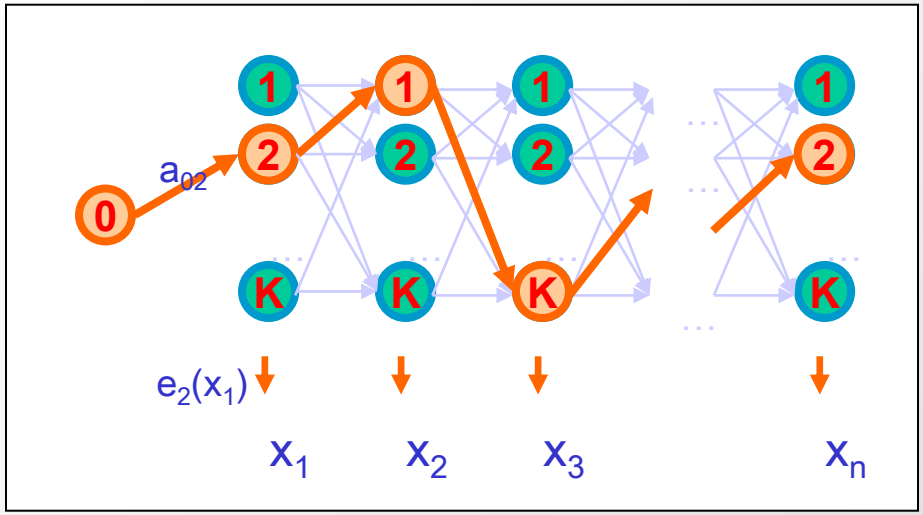
Given a HMM, we can generate a sequence of length n as follows:

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?

Slide credit: Serafim Batzoglou

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)?

- $P(x) = \sum_{\pi} P(x,\pi) = \sum_{\pi} P(x|\pi)\, P(\pi)$
- (weighted average of conditional probability, summed over all paths, weighted by each path's probability)

- Challenge: exponential number of paths

Slide credit: Serafim Batzoglou

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 if Viterbi has highest density
  - BUT: incorrect
- (real) solution
  - Calculate the exact sum iteratively
    - $P(x) = \sum_{\pi} P(x,\pi)$
  - Can use dynamic programming

Slide credit: Serafim Batzoglou

The Forward Algorithm – derivation

Define the forward probability:

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

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

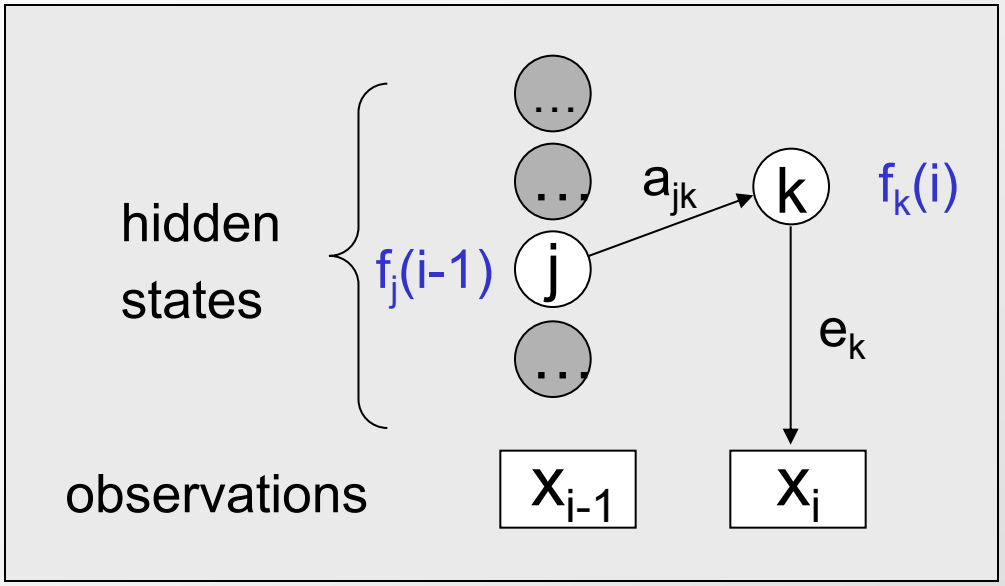
$$= \sum_k \boxed{\sum_{\pi 1 \ldots \pi i-2} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1} = k)}\, a_{kl}\, e_l(x_i)$$

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

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

Slide credit: Serafim Batzoglou

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



- Assume we know f_j for the previous time step (i-1)
- Calculate $f_k(i) = e_k(x_i) * \text{sum}_j (f_j(i-1) \times a_{jk})$

updated sum

this emission

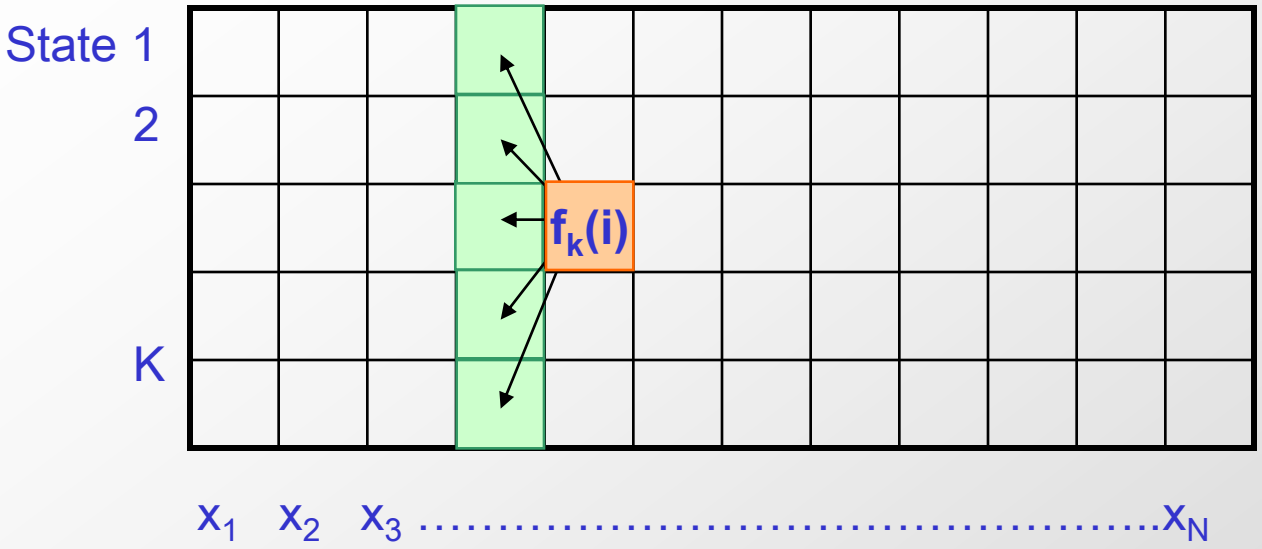
sum ending in state j at step i

transition from state j

every possible previous state j

Slide credit: Serafim Batzoglou

The Forward Algorithm



Input: x = x1……xN

Initialization:

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

Iteration:

$f_k(i) = e_K(x_i) \times \text{sum}_j\ a_{jk}\ f_j(i\text{-}1)$

Termination:

$P(x, \pi^*) = \text{sum}_k\ f_k(N)$

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: $O(KN)$

Slide credit: Serafim Batzoglou

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 $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ 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

Examples of HMMs for genome annotation

Application	Detection of GC-rich regions	Detection of conserved regions	Detection of protein-coding exons	Detection of protein-coding conservation	Detection of protein-coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	64x64 matrix of codon substitution frequencies	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

What have we learned ?

- Modeling sequential data
  - Recognize a ***type*** of sequence, genomic, oral, verbal, visual, etc…
- Definitions
  - Markov Chains
  - Hidden Markov Models (HMMs)
- Examples of HMMs
  - Recognizing GC-rich regions, preferentially-conserved elements, coding exons, protein-coding gene structures, chromatin states
- 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

6.047 / 6.878 / HST.507 Computational Biology
Fall 2015