Hidden Markov Models and Gene Prediction

Overview

- Sequence profiles
- How hidden Markov models work
- Training HMMs
- HMMs for gene prediction

K-ELQRAASLTIEV

KDEGQK--SLVIDV

If we have an alignment…

…what can we do with it?

For many questions, we would like to know the distribution of residues (and gaps) in a block of sequences

```
CGGCCT
CGAGCT
GATGCA
AAAGCA
ATAGCA
TCTACT
AACATC
TACGCC
AACGAG
AGCTGT
```

Position-specific scoring matrices (PSSM)

PAM, BLOSUM, etc. are position-**independent** scoring matrices

A PSSM is a log-odds matrix of column frequencies

CGGCCT
CGAGCT
GATGCA
AAAGCA
ATAGCA
TCTACT
AACATC
TACGCC
AACGAG
AGCTGT

Frequency Matrix

	1	2	3	4	5	6
A	0.5	0.5	0.3	0.2	0.1	0.3
C	0.2	0.1	0.4	0.1	0.7	0.2
G	0.1	0.3	0.1	0.5	0.1	0.1
T	0.2	0.1	0.2	0.2	0.1	0.4

Background frequencies:
A = 19/60 = 0.317
C = 17/60 = 0.283
G = 12/60 = 0.2
T = 12/60 = 0.2

Frequency Matrix

	1	2	3	4	5	6
A	0.5	0.5	0.3	0.2	0.1	0.3
C	0.2	0.1	0.4	0.1	0.7	0.2
G	0.1	0.3	0.1	0.5	0.1	0.1
T	0.2	0.1	0.2	0.2	0.1	0.4

$\log_n$-odds matrix (n = *e*)

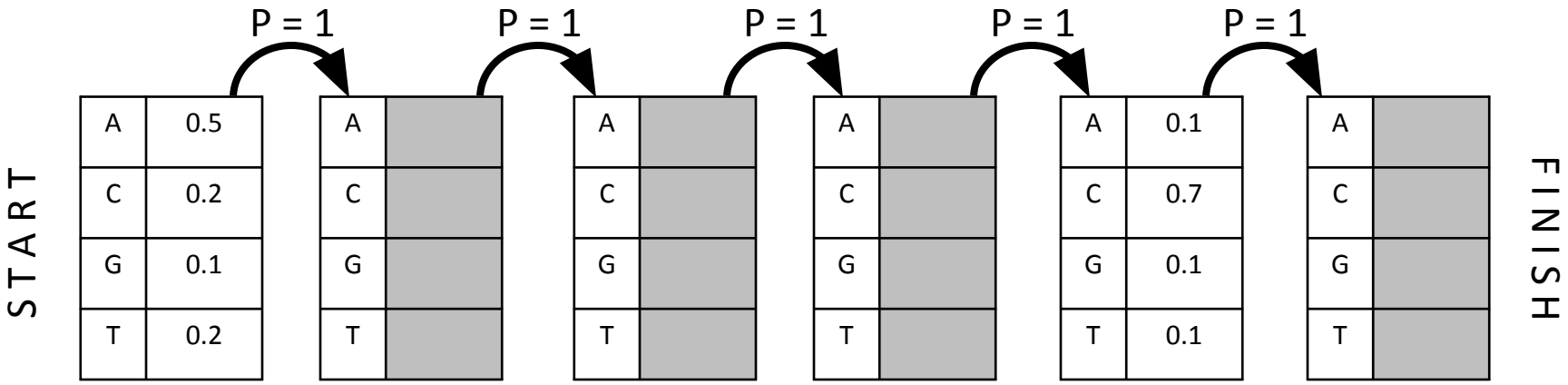
	1	…	5
A	0.18		-0.5
C	-0.15		0.54
G	-0.3		-0.3
T	0		-0.3

Background frequencies:
A = 19/60 = 0.317
C = 17/60 = 0.283
G = 12/60 = 0.2
T = 12/60 = 0.2

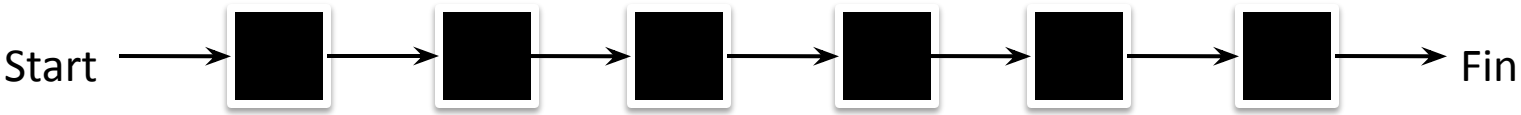
Aligning a sequence against log-odds matrix:
Add scores for residue at each position, then take n^{sum}

How do we represent insertions and deletions?

Transitions in a Probability Matrix

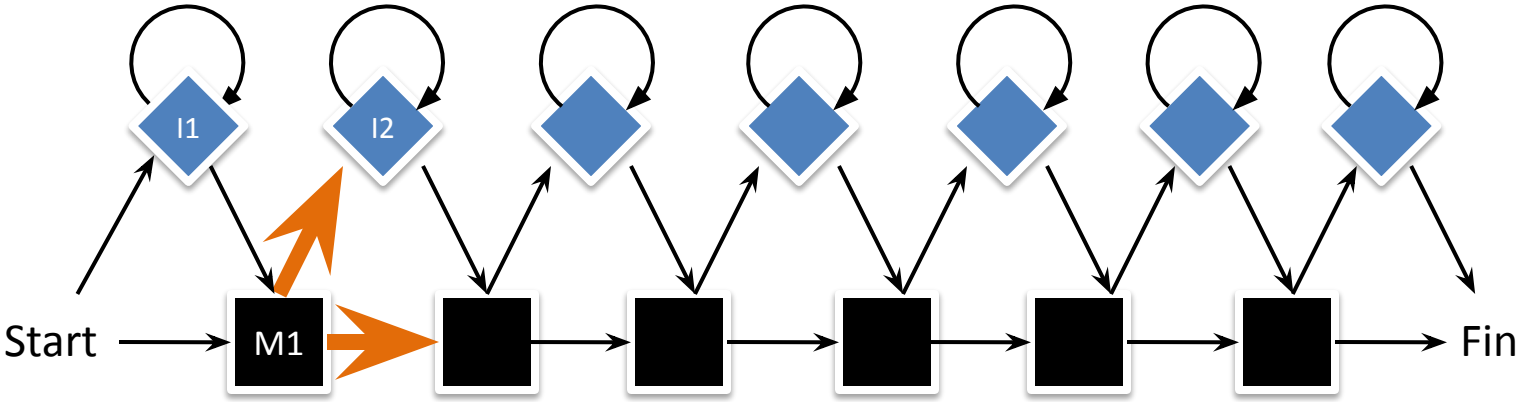


Transition from match state *k* to match state *k* + 1 with probability 1.0



Match states

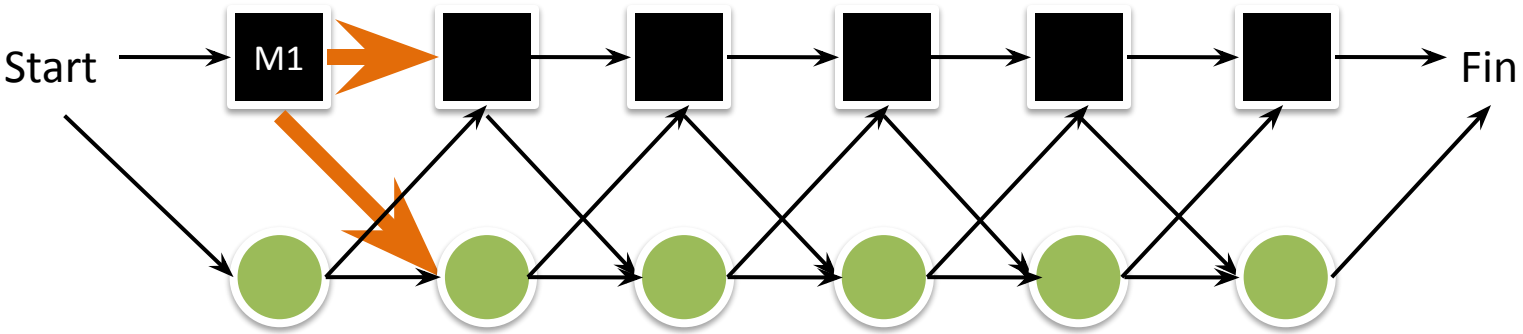
Insertions



Insert states

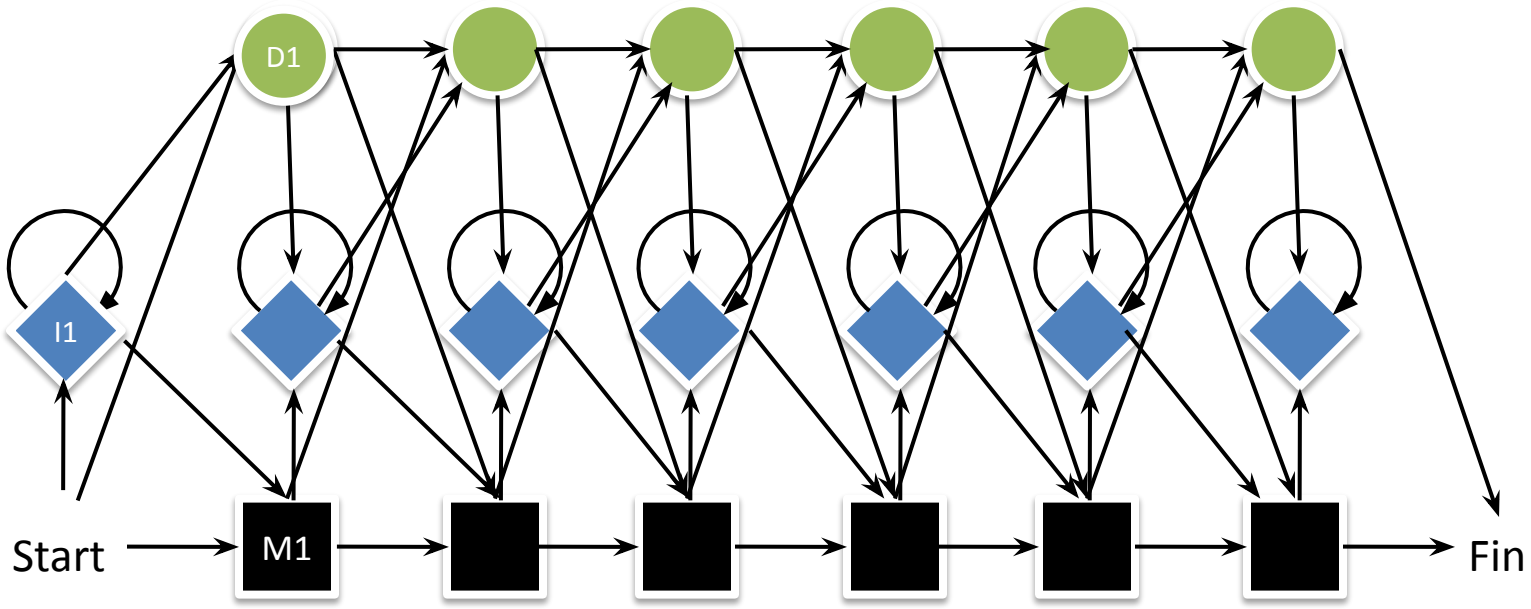
Transition probabilities out of any state must sum to 1.0

Deletions



Delete states

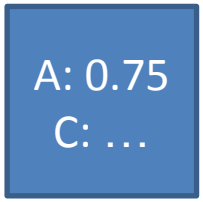
Hidden Markov Model



HIDDEN because we don't actually know the states of the sequence we're looking at
MARKOV because the future does not depend on the past
MODEL because, well, it's a model

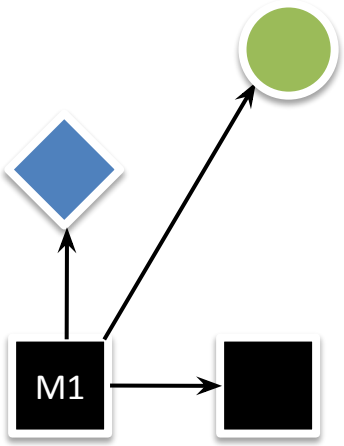
Key components of an HMM

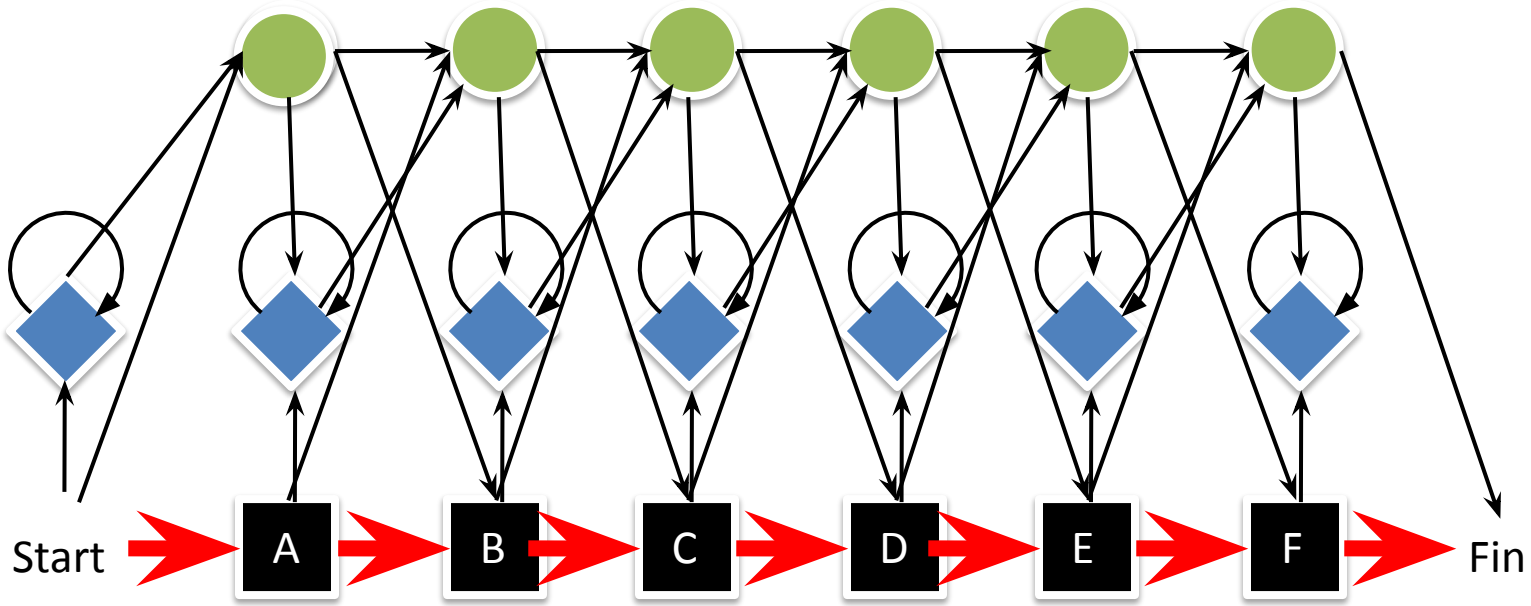
- EMISSIONS: A character (nucleotide or amino acid) produced by a given insertion or match state



emission probability

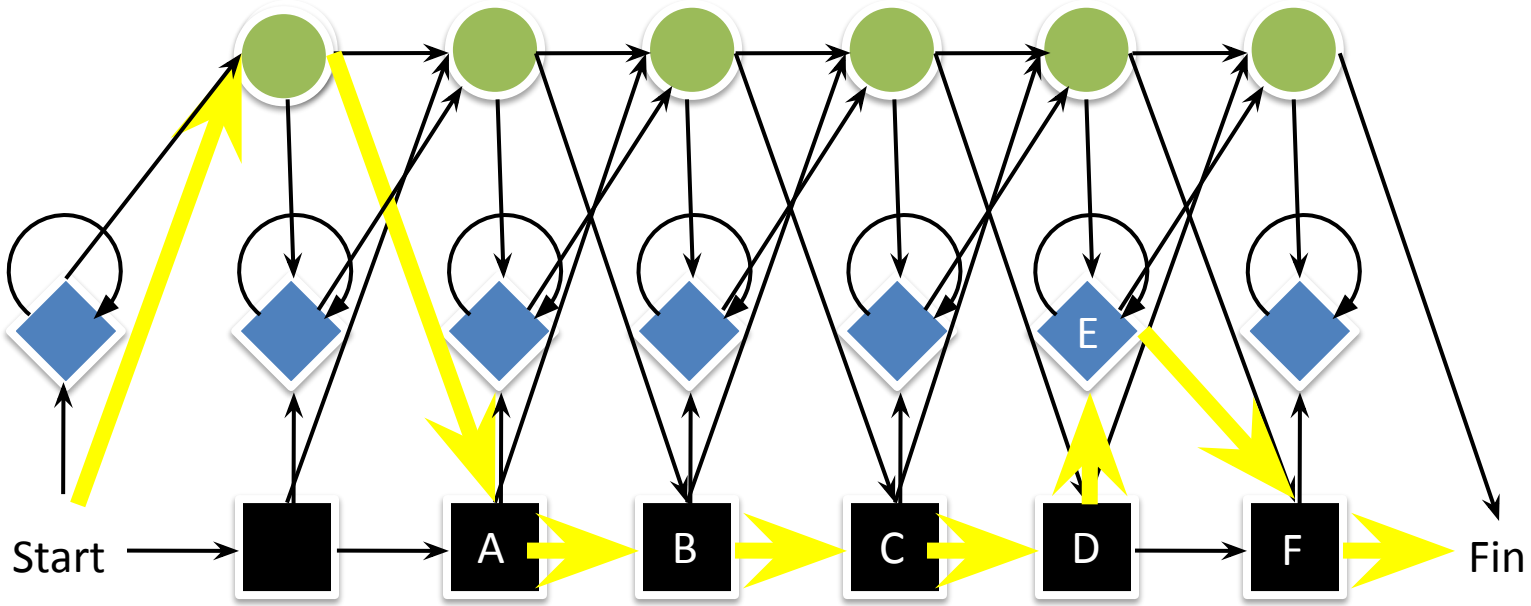
- TRANSITIONS: The probability of going from state *i* to state *j* (sum of all transitions from a given state = 1)





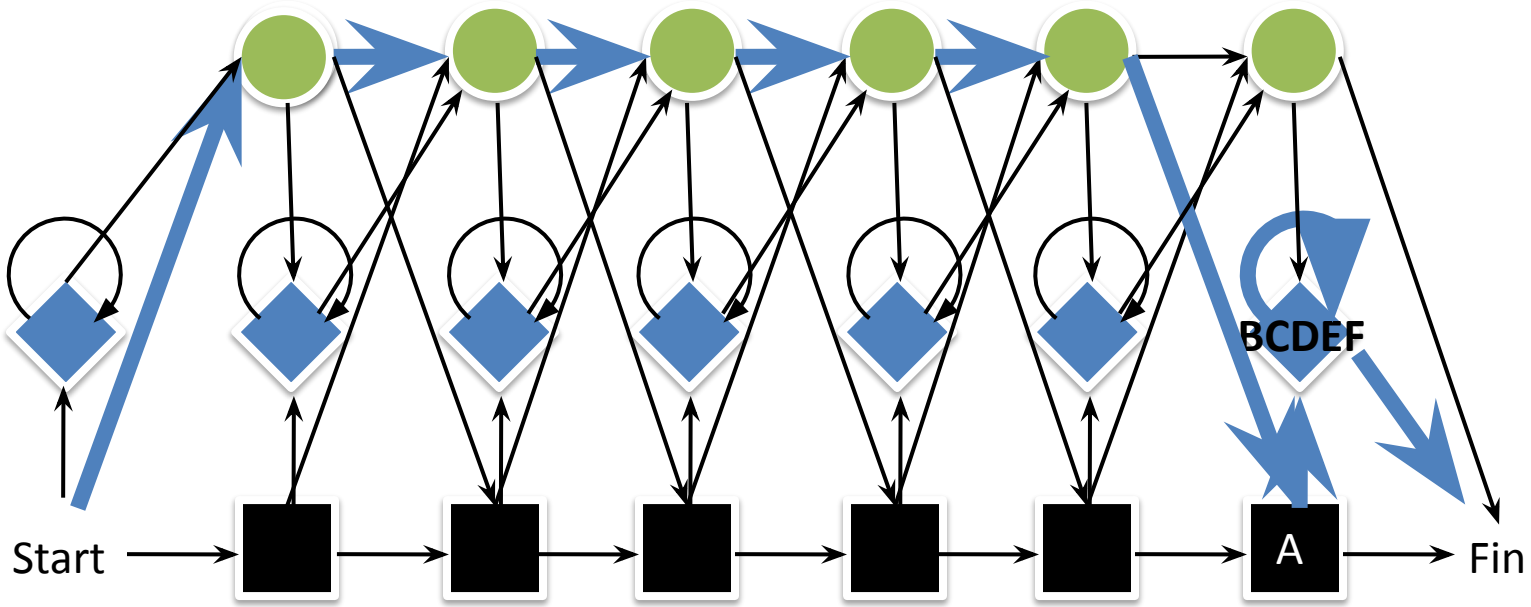
Let's run a sequence through the HMM!

ABCDEF



Let's run a sequence through the HMM!

ABCDEF



Let's run a sequence through the HMM!

ABCDEF

The product of the EMISSION PROBABILITIES *e* and the TRANSITION PROBABILITIES *a* through the model

=

The **joint probability** of the *sequence x* and the *path* π

The product of the EMISSION PROBABILITIES *e* and the TRA[illegible] *a* through the model

Or sum of logs

=

The **joint probability** of the *sequence x* and the *path* π

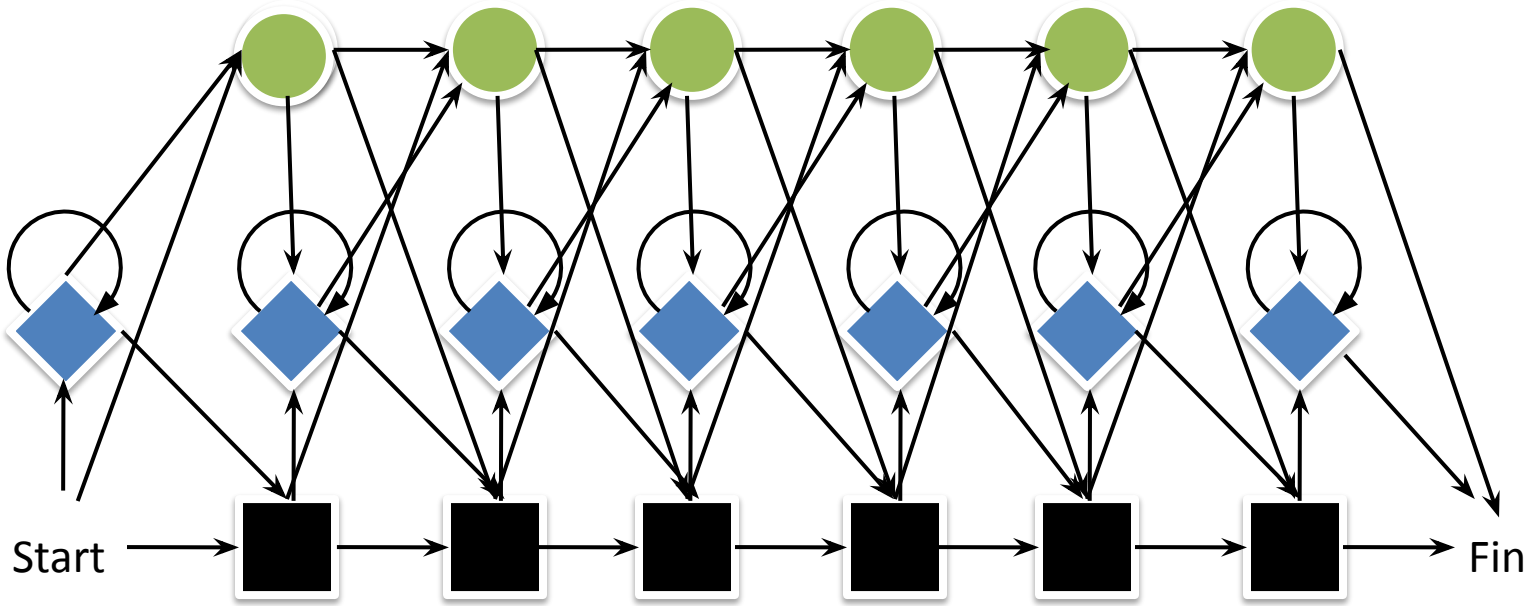
Best path

- There are many paths π through the model for any given sequence *x*

- What is the best path π^* ?

The Viterbi Algorithm

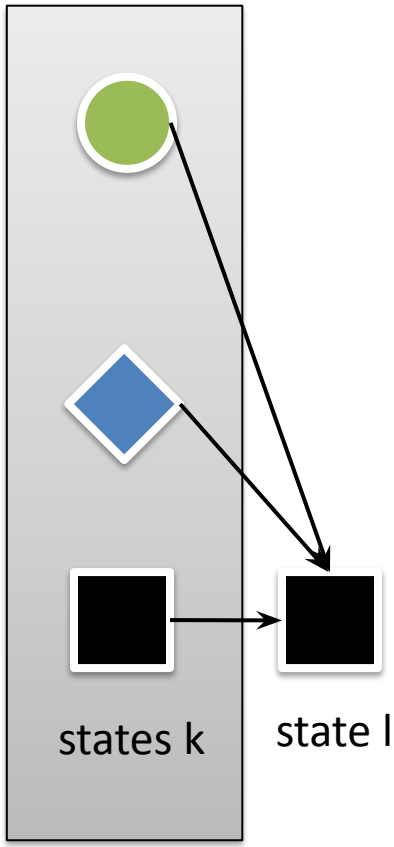
- As with multiple sequence alignment, we cannot be greedy in our choice of path
- But we only need to consider the **best path** to every possible state in the model
- Dynamic programming!



v(Start) = 1

$$v_l(i) = e_i(x_i)\max_k(v_k(i-1)a_{kl})$$

Huh?



i = { A,B,C,D,E,F }

$$v_l(i) = e_i(x_i) \max_k (v_k(i-1)a_{kl})$$

Viterbi score of sequence position *I* at state *l*

Emission probability of x_i

max over all possibilities

Viterbi score at previous state, times the transision probabillity

So we are saving the best path for each **character** { A,B,C,D,E,F } at each **state** in the HMM

When we choose our best incoming path, we save a pointer as before and **backtrace**

Complexity = O(LS) (# of characters x # of states in the HMM structure) – kinda like n^2

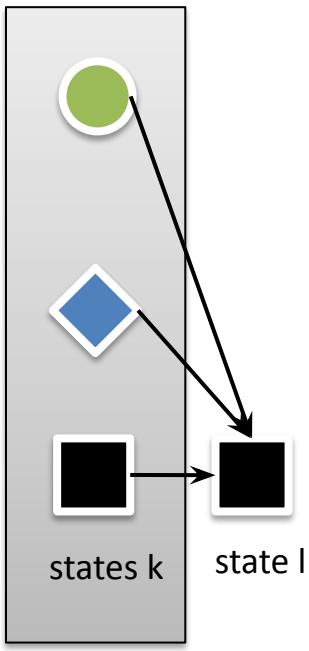
The Viterbi alignment of each member of a set of sequences *X* to a trained HMM yields a ***multiple alignment*** of these sequences

All Paths

FORWARD algorithm

sums over incoming paths

instead of taking max



i = { A,B,C,D,E,F }

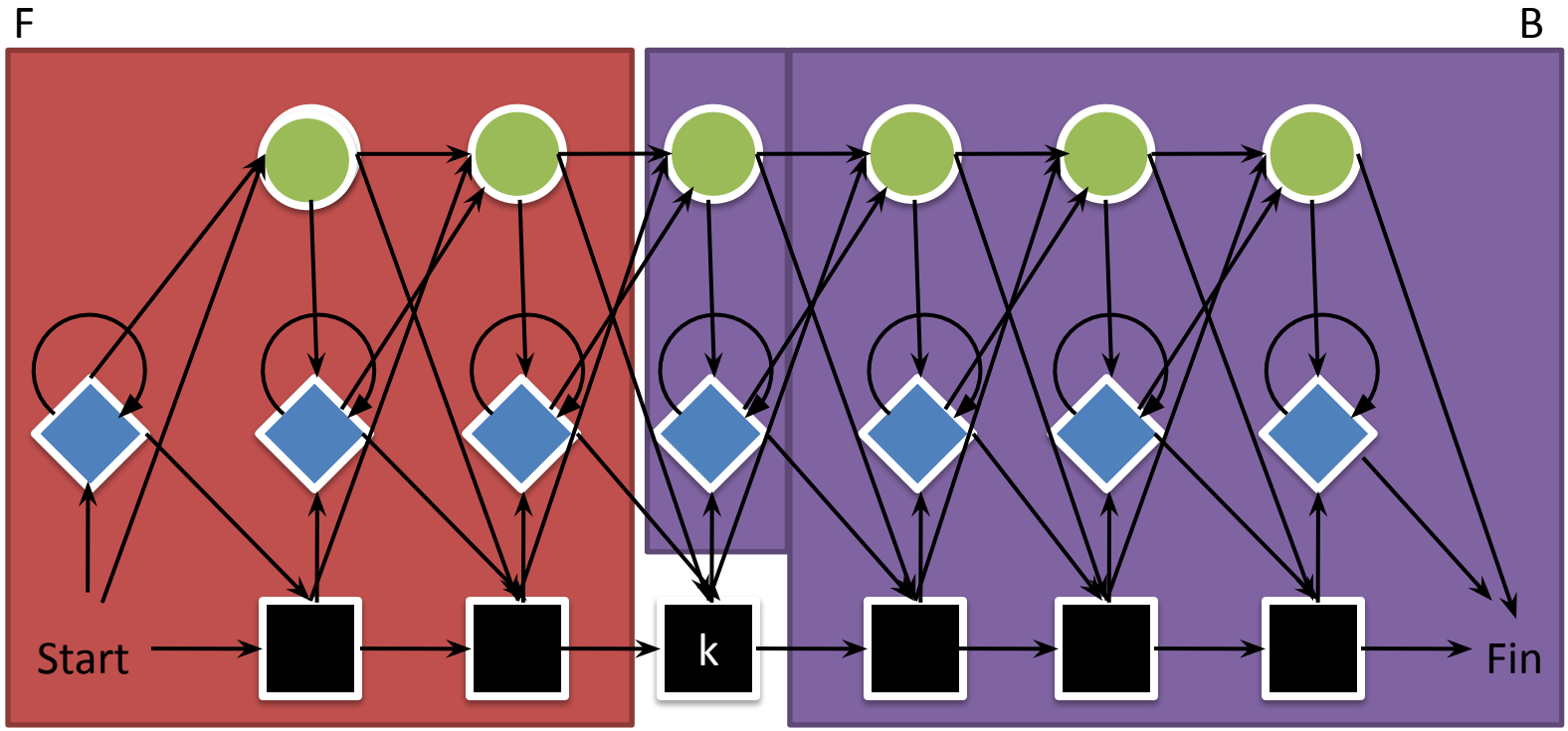
$$f_l(i) = e_i(x_i) \sum_k (f_k(i-1) a_{kl})$$

The Backward Algorithm

- Kind of like the forward algorithm, but starts from the finish and works backward

$$b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$$

- Why would we want to do this?

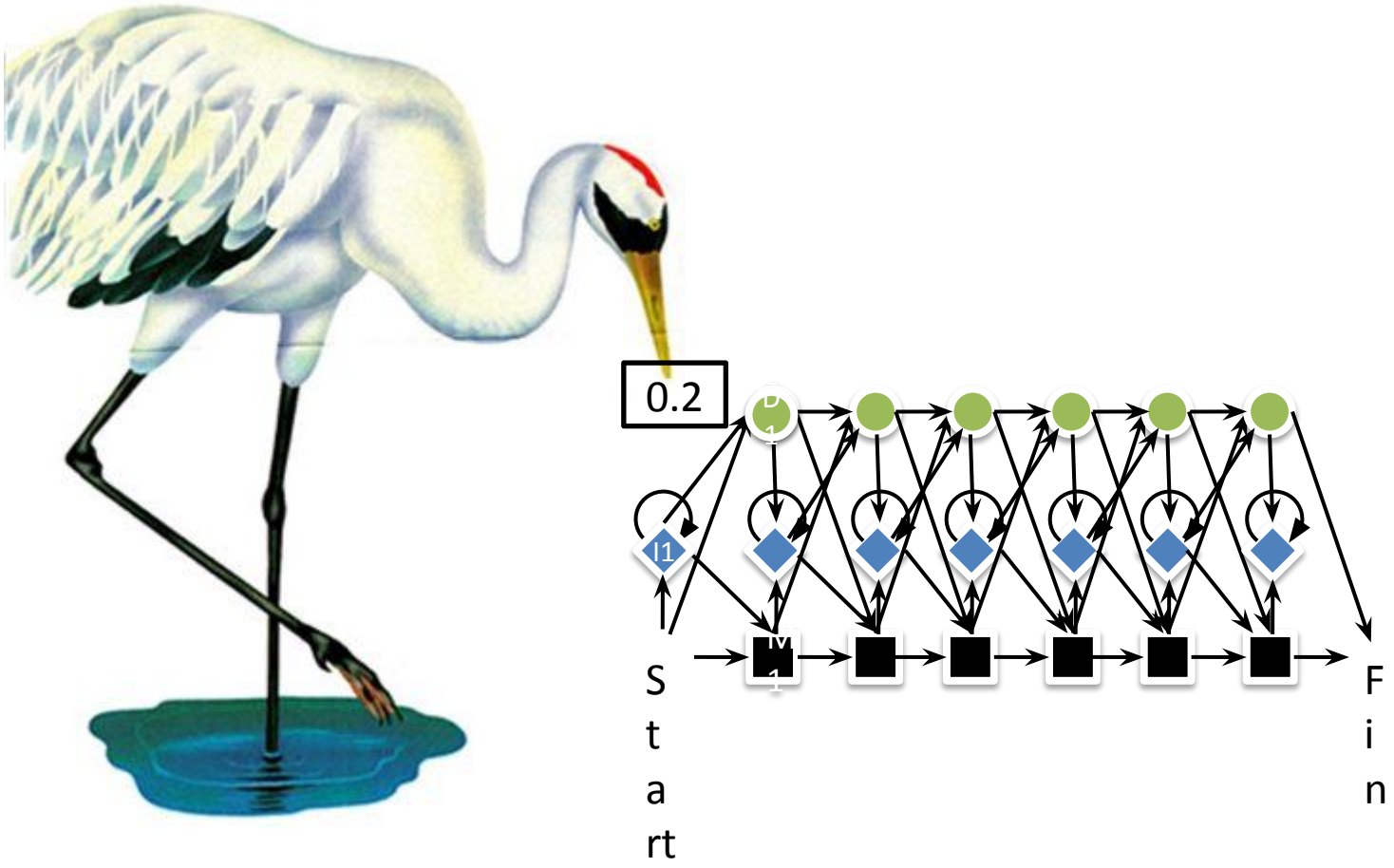


By running the forward and backward algorithms together for a given sequence, we can compute the probability that character *i* in sequence *x* maps to state *k*

ABCDEF

P(x, *k* = D)?

Training HMMs



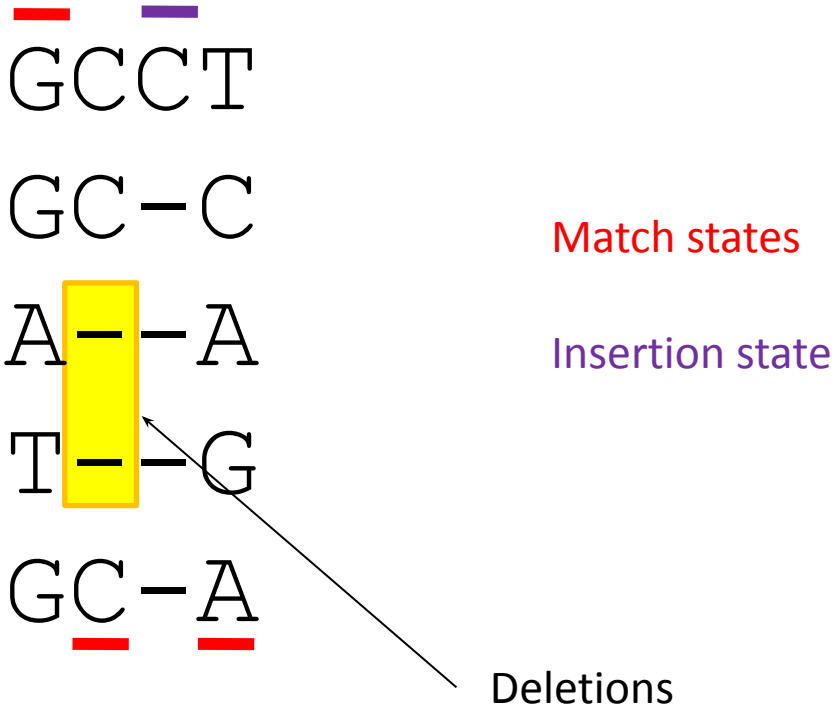
Two components of training

- Build the HMM structure or ‘skeleton’
  - Custom-tailored with exquisite knowledge of the problem to be modelled
  - In ignorance, build a complete model
- Assign transition and emission probabilities to the thing

Training an HMM (supervised)

- Construct a multiple sequence alignment using some method, and build the HMM using empirical frequencies

- Supervised because we're specifying exactly WHAT sequences belong in the model



Note that we now get custom gap costs!

Big Alphabets, Few Sequences

Homologous residues
from a family of sequences

I
N
D
Q
R
S
D
Q
R
N
M
I
I
D
D

Incomplete sampling
in our database

Sampled set

I
N
D
Q
Q
D

Build matrix

A	0
C	0
D	2
E	0
F	0
G	0
H	0
I	1
K	0
L	0
M	0
N	1
P	0
Q	2
R	0
S	0
T	0
V	0
W	0
Y	0

What happens when the probability of
character *i* at position *k* is = 0?

Psolution

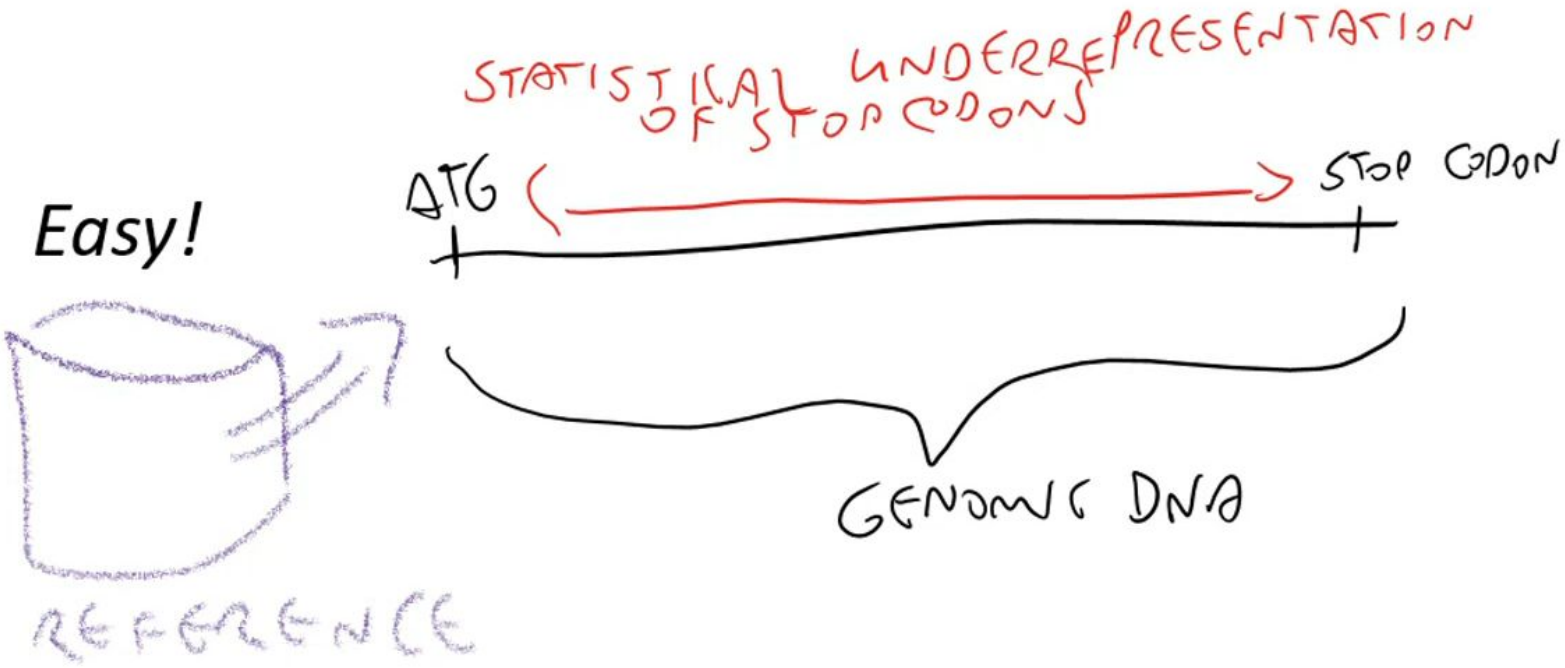
- Add **pseudocounts** to each column of the multiple sequence alignment

- Laplace's Rule: Add 1 to every count (!)
- Add small counts in proportion to background frequencies
- Modify added counts using PAM matrix or other distributions (Dirichlet mixtures)

Beyond sequences:
Other applications of HMMs

HMMs in Gene prediction

Given a genome sequence (complete or draft), identify all of the genes

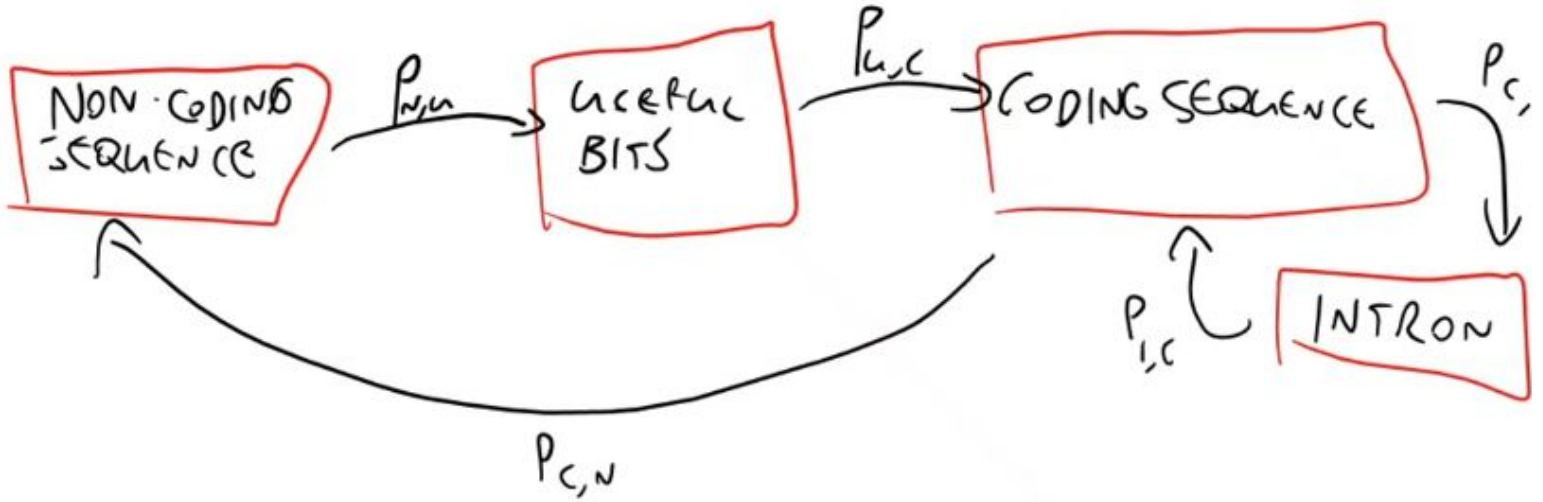


Maybe not so easy

Because:

- Alternative start codons (TTG, GTG)
- Uncertain start codons (which ATG?)
- Introns
- Short genes
- Non-protein-coding genes
- Genes that overlap
- Genes with no known homologs

Hidden Markov Models – the basic idea



[GeneMark.hmm – see bonus slides]

Advantages of HMMs

- Probabilistic framework – the forward algorithm returns the probability of the data (sequence) given the model (the HMM)

- Eminently tweakable – can be designed carefully to capture the patterns in biological sequences

Disadvantages

- Must be designed carefully to adequately capture the patterns in biological sequences
  - Or, use a generic framework
- Can be computationally expensive (kind of like DP for sequence alignment)
- It's Markovian, so you cannot represent correlations of matches at different sites

Implementations

- HMMER (http://hmmer.janelia.org/)

- SAM (http://compbio.soe.ucsc.edu/sam.html)