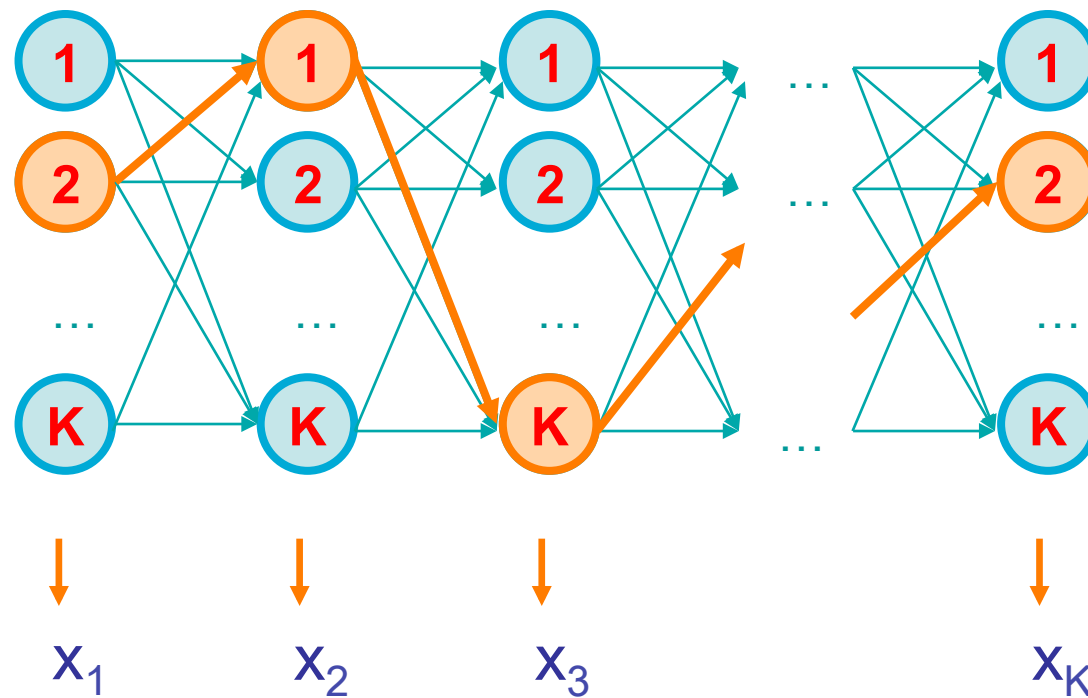




# Hidden Markov Models



# HiSeq X & NextSeq



## NextSeq 500 Sequencing System Performance Parameters

### NEXTSEQ 500 HIGH OUTPUT KIT \*

READ LENGTH	TOTAL TIME†	OUTPUT
2 × 150 bp	~29 hrs	100-120 Gb
2 × 75 bp	18 hrs	50-60 Gb
1 × 75 bp	11 hrs	25-30 Gb

### NEXTSEQ 500 MID OUTPUT KIT \*

READ LENGTH	TOTAL TIME†	OUTPUT
2 × 150 bp	26 hrs	32.5-39 Gb
2 × 75 bp	15 hrs	16.25-19.5 Gb

## Reads Passing Filter

### NEXTSEQ 500 HIGH OUTPUT KIT

Single Reads	Up to 400 Million
Paired-End Reads	Up to 800 million

### NEXTSEQ 500 MID OUTPUT KIT

Single Reads	Up to 130 Million
Paired-End Reads	Up to 260 Million

# Viterbi, Forward, Backward



## VITERBI

### Initialization:

$$V_0(0) = 1$$

$$V_k(0) = 0, \text{ for all } k > 0$$

### Iteration:

$$V_l(i) = e_l(x_i) \max_k V_k(i-1) a_{kl}$$

### Termination:

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

## FORWARD

### Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \text{ for all } k > 0$$

### Iteration:

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

### Termination:

$$P(x) = \sum_k f_k(N)$$

## BACKWARD

### Initialization:

$$b_k(N) = 1, \text{ for all } k$$

### Iteration:

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

### Termination:

$$P(x) = \sum_k a_{0k} e_k(x_1) b_k(1)$$



---

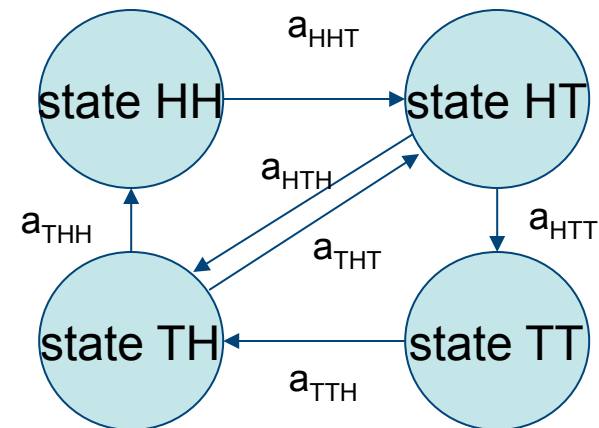
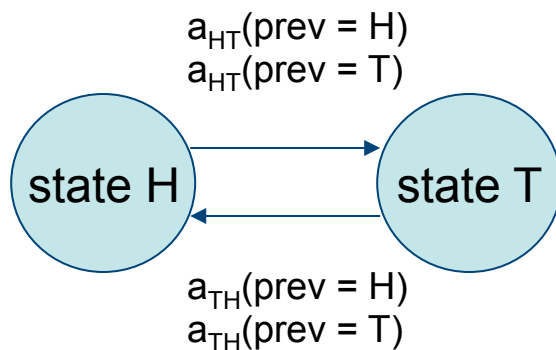
# Variants of HMMs



# Higher-order HMMs



- How do we model “memory” larger than one time point?
- $P(\pi_{i+1} = l \mid \pi_i = k)$   $a_{kl}$
- $P(\pi_{i+1} = l \mid \pi_i = k, \pi_{i-1} = j)$   $a_{jkl}$
- ...
- A second order HMM with K states is equivalent to a first order HMM with  $K^2$  states





# Similar Algorithms to 1<sup>st</sup> Order

---

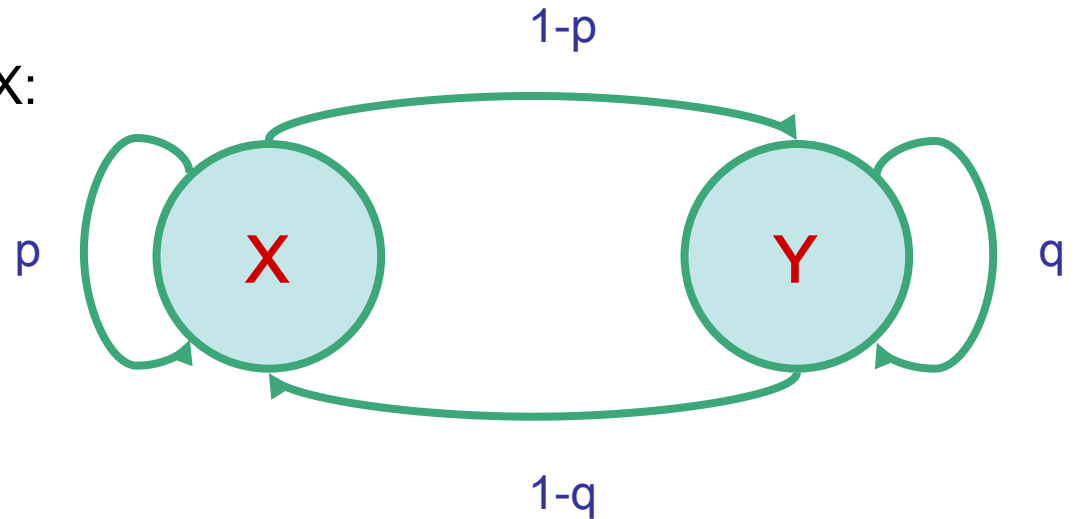
- $P(\pi_{i+1} = l \mid \pi_i = k, \pi_{i-1} = j)$ 
  - $V_{lk}(i) = \max_j \{ V_{kj}(i-1) + \dots \}$
  - Time? Space?

# Modeling the Duration of States



Length distribution of region X:

$$E[l_x] = 1/(1-p)$$

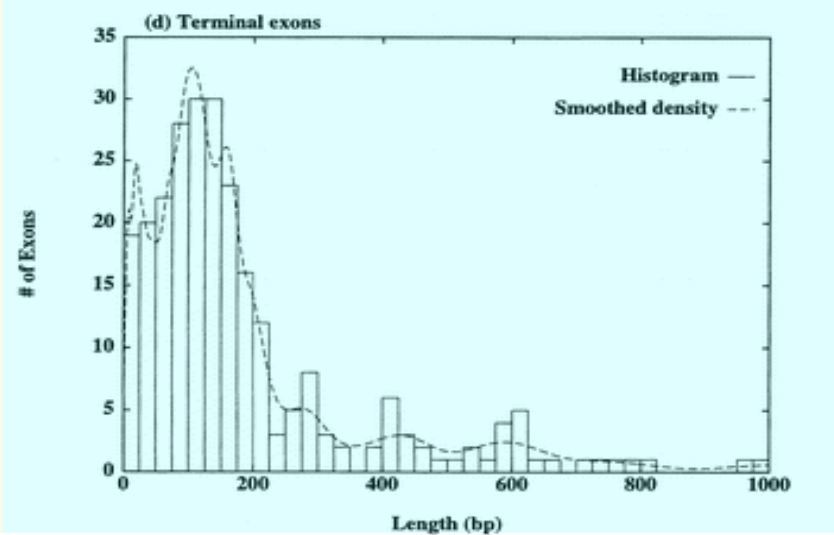
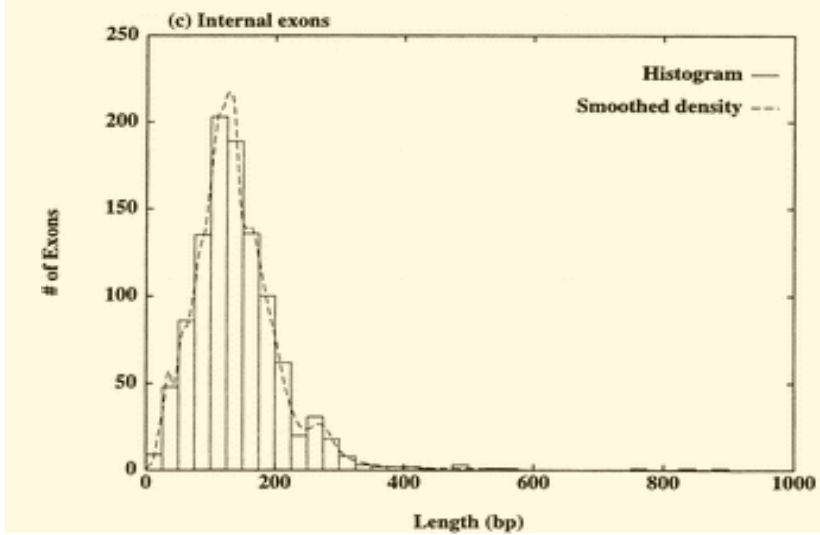
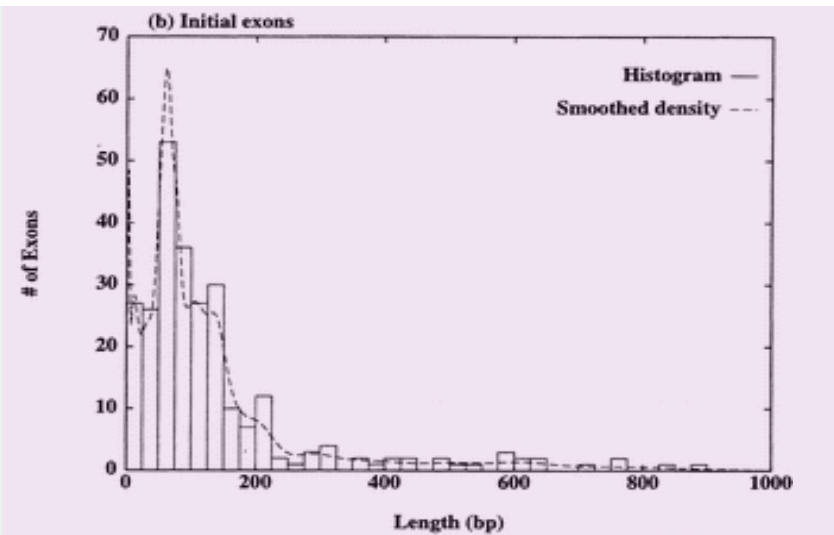
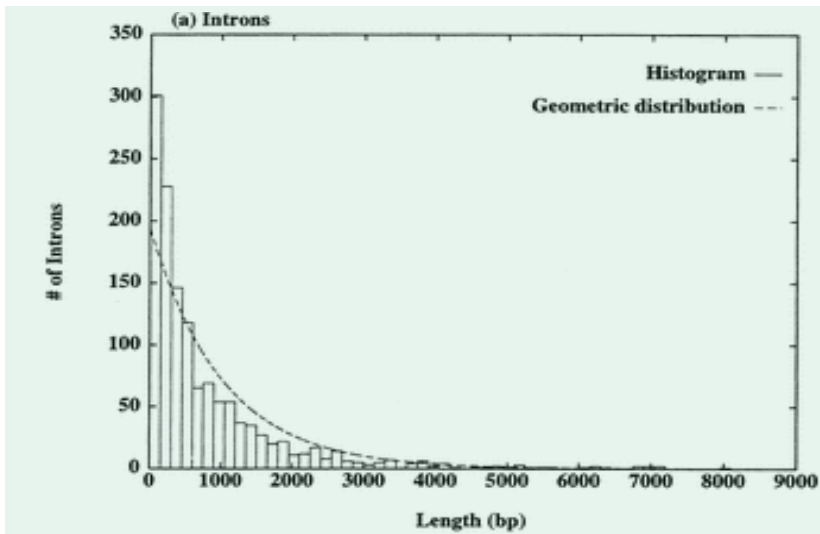
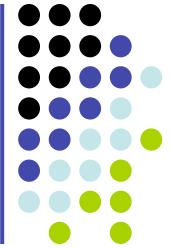


- Geometric distribution, with mean  $1/(1-p)$

This is a significant disadvantage of HMMs

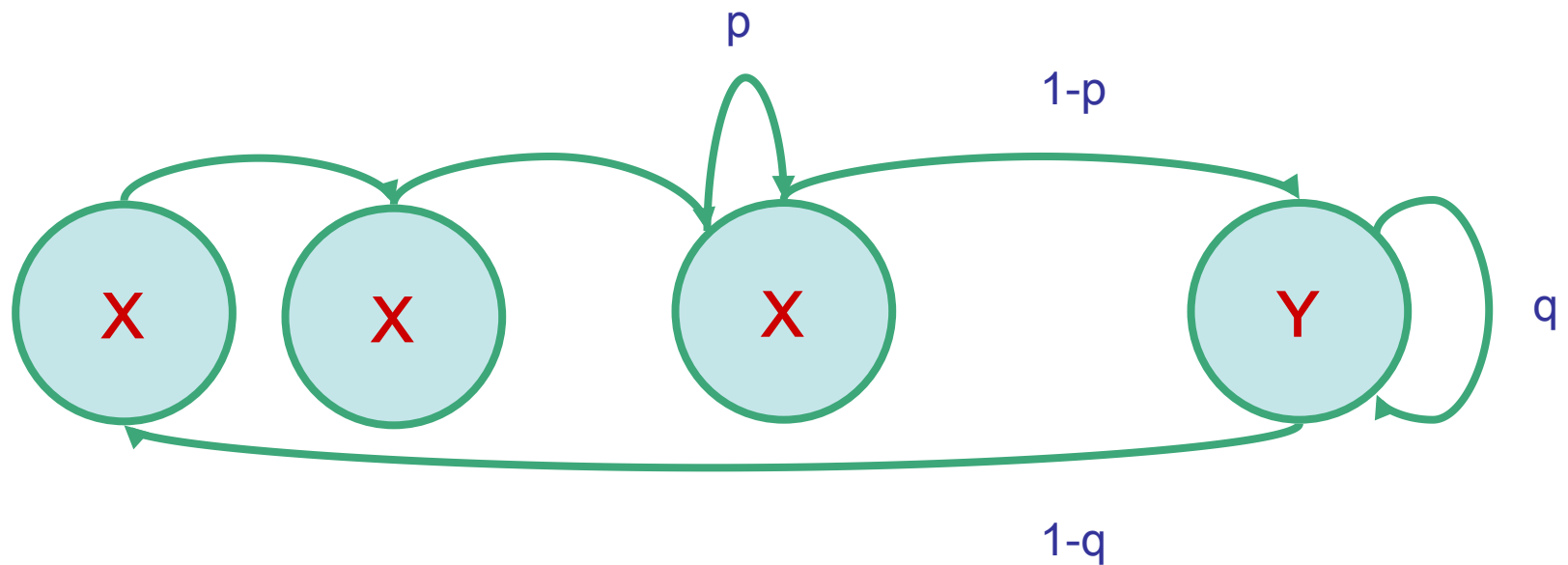
Several solutions exist for modeling different length distributions

# Example: exon lengths in genes





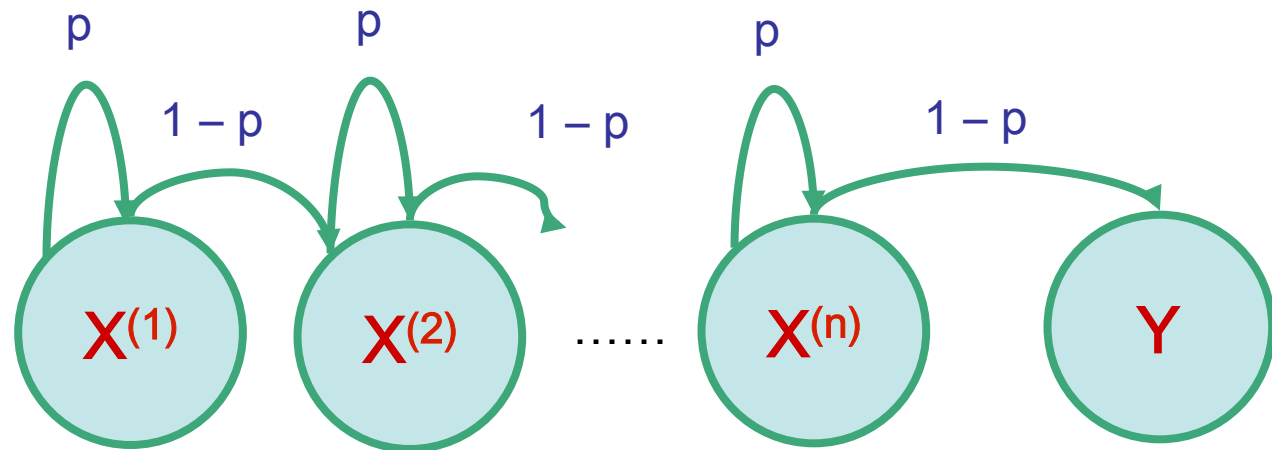
# Solution 1: Chain several states



Disadvantage: Still very inflexible

$$I_X = C + \text{geometric with mean } 1/(1-p)$$

# Solution 2: Negative binomial distribution



Duration in  $X$ :  $m$  turns, where

- During first  $m - 1$  turns, exactly  $n - 1$  arrows to next state are followed
- During  $m^{\text{th}}$  turn, an arrow to next state is followed

$$P(l_X = m) = \binom{m-1}{n-1} (1-p)^{n-1+1} p^{(m-1)-(n-1)} = \binom{m-1}{n-1} (1-p)^n p^{m-n}$$

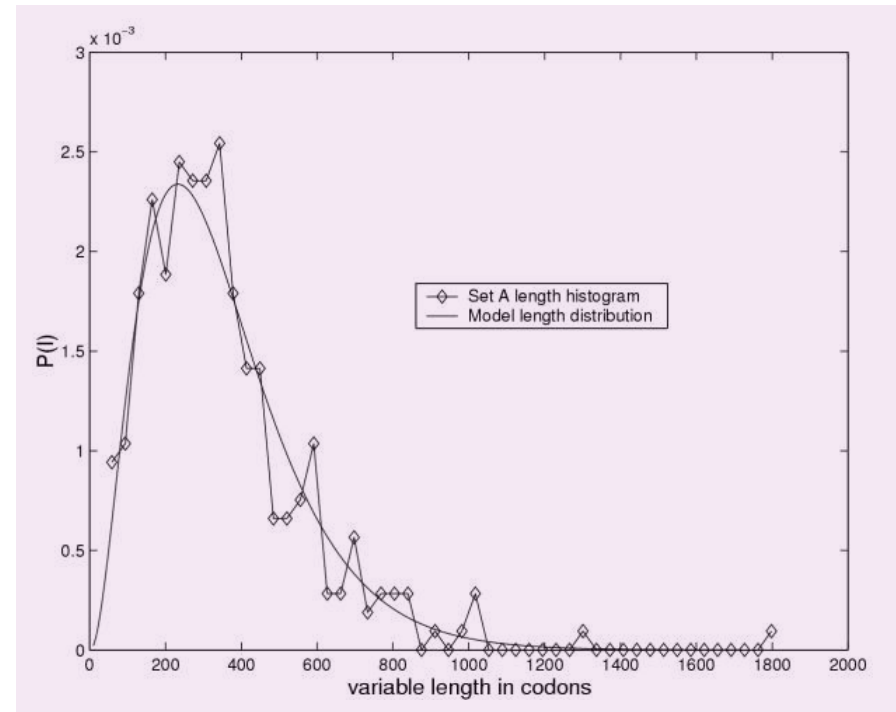
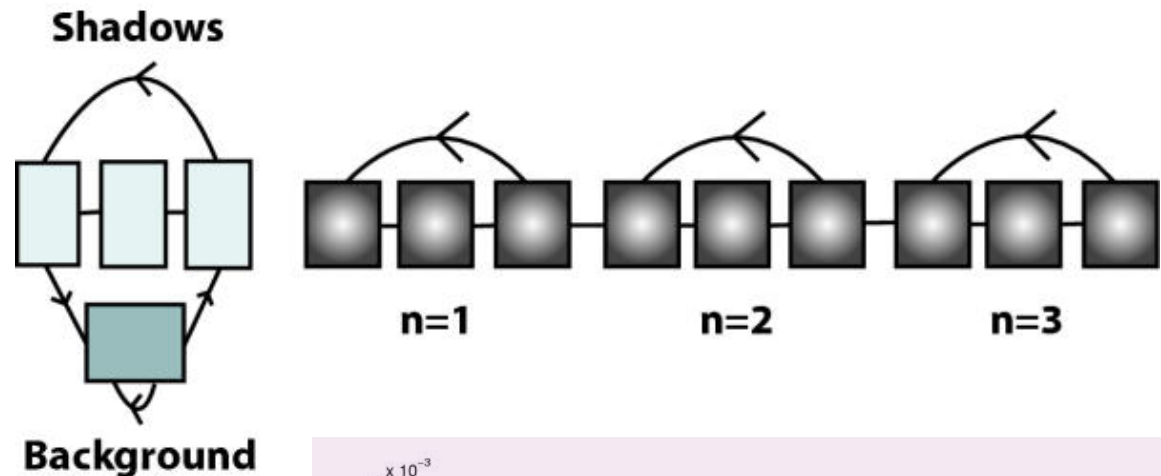
# Example: genes in prokaryotes



- EasyGene:  
Prokaryotic  
gene-finder

Larsen TS, Krogh A

- Negative binomial with  $n = 3$

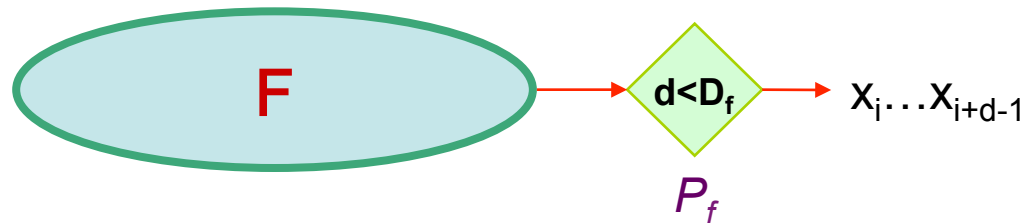


# Solution 3: Duration modeling



Upon entering a state:

1. Choose duration  $d$ , according to probability distribution
2. Generate  $d$  letters according to emission probs
3. Take a transition to next state according to transition probs



Disadvantage: Increase in complexity of Viterbi:

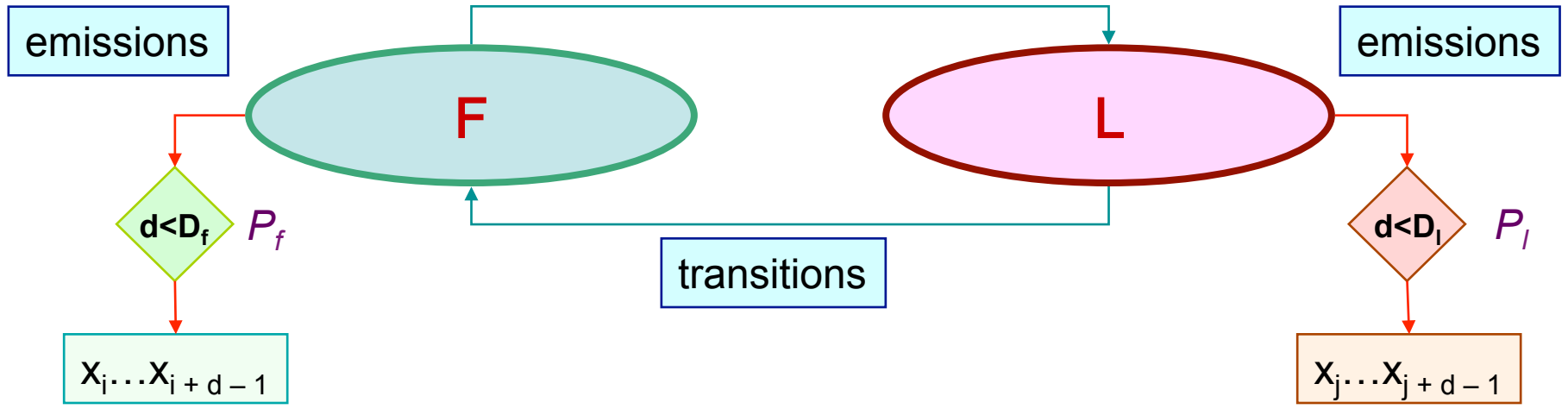
Time:  $O(D)$  ○ ○ ○  
Space:  $O(1)$

*Warning, Rabiner's  
tutorial claims  $O(D^2)$   
&  $O(D)$  increases*

where  $D$  = maximum duration of state



# Viterbi with duration modeling



Recall original iteration:

$$V_l(i) = \max_k V_k(i-1) a_{kl} \times e_l(x_i)$$

*Precompute cumulative values*

New iteration:

$$V_l(i) = \max_k \max_{d=1 \dots D_l} V_k(i-d) \times P_l(d) \times a_{kl} \times \prod_{j=i-d+1 \dots i} e_l(x_j)$$



---

# Learning

Re-estimate the parameters of the model based on training data

# Two learning scenarios

---



## 1. Estimation when the “right answer” is known

### Examples:

**GIVEN:** a genomic region  $x = x_1 \dots x_{1,000,000}$  where we have good (experimental) annotations of the CpG islands

**GIVEN:** the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

## 2. Estimation when the “right answer” is unknown

### Examples:

**GIVEN:** the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

**GIVEN:** 10,000 rolls of the casino player, but we don't see when he changes dice

**QUESTION:** Update the parameters  $\theta$  of the model to maximize  $P(x|\theta)$



# 1. When the states are known

Given  $x = x_1 \dots x_N$

for which the true  $\pi = \pi_1 \dots \pi_N$  is known,

## Define:

$A_{kl}$  = # times  $k \rightarrow l$  transition occurs in  $\pi$

$E_k(b)$  = # times state  $k$  in  $\pi$  emits  $b$  in  $x$

We can show that the maximum likelihood parameters  $\theta$  (maximize  $P(x|\theta)$ ) are:

$$a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)}$$





# 1. When the states are known

**Intuition:** When we know the underlying states,  
Best estimate is the normalized frequency of  
transitions & emissions that occur in the training data

**Drawback:**

Given little data, there may be **overfitting**:  
 $P(x|\theta)$  is maximized, but  $\theta$  is unreasonable  
**0 probabilities – BAD**

**Example:**

Given 10 casino rolls, we observe

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

$\pi = F, F, F, F, F, F, F, F, F, F$

Then:

$$a_{FF} = 1; \quad a_{FL} = 0$$

$$e_F(1) = e_F(3) = .2;$$

$$e_F(2) = .3; e_F(4) = 0; e_F(5) = e_F(6) = .1$$

# Pseudocounts



Solution for small training sets:

Add pseudocounts

$$\begin{array}{lll} A_{kl} & = \# \text{ times } k \rightarrow l \text{ transition occurs in } \pi & + r_{kl} \\ E_k(b) & = \# \text{ times state } k \text{ in } \pi \text{ emits } b \text{ in } x & + r_k(b) \end{array}$$

$r_{kl}$ ,  $r_k(b)$  are pseudocounts representing our prior belief

Larger pseudocounts  $\Rightarrow$  Strong prior belief

Small pseudocounts ( $\epsilon < 1$ ): just to avoid 0 probabilities

# Pseudocounts



**Example:** dishonest casino

We will observe player for one day, 600 rolls

Reasonable pseudocounts:

$$r_{0F} = r_{0L} = r_{F0} = r_{L0} = 1;$$

$$r_{FL} = r_{LF} = r_{FF} = r_{LL} = 1;$$

$$r_F(1) = r_F(2) = \dots = r_F(6) = 20 \quad (\text{strong belief fair is fair})$$

$$r_L(1) = r_L(2) = \dots = r_L(6) = 5 \quad (\text{wait and see for loaded})$$

Above #s are arbitrary – assigning priors is an art



## 2. When the states are hidden

---

We don't know the true  $A_{kl}$ ,  $E_k(b)$

Idea:

- We estimate our “best guess” on what  $A_{kl}$ ,  $E_k(b)$  are
  - Or, we start with random / uniform values
- We update the parameters of the model, based on our guess
- We repeat



## 2. When the states are hidden

---

Starting with our best guess of a model  $M$ , parameters  $\theta$ :

Given  $x = x_1 \dots x_N$

for which the true  $\pi = \pi_1 \dots \pi_N$  is unknown,

We can get to a provably more likely parameter set  $\theta$

*i.e.,  $\theta$  that increases the probability  $P(x | \theta)$*

Principle: **EXPECTATION MAXIMIZATION**

1. Estimate  $A_{kl}$ ,  $E_k(b)$  in the training data
2. Update  $\theta$  according to  $A_{kl}$ ,  $E_k(b)$
3. Repeat 1 & 2, until convergence

# Estimating new parameters



To estimate  $A_{kl}$ : (assume “ $\theta_{\text{CURRENT}}$ ”, in all formulas below)

At each position  $i$  of sequence  $x$ , find probability transition  $k \rightarrow l$  is used:

$$P(\pi_i = k, \pi_{i+1} = l \mid x) = [1/P(x)] \times P(\pi_i = k, \pi_{i+1} = l, x_1 \dots x_N) = Q/P(x)$$

$$\begin{aligned} \text{where } Q &= P(x_1 \dots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \dots x_N) = \\ &= P(\pi_{i+1} = l, x_{i+1} \dots x_N \mid \pi_i = k) P(x_1 \dots x_i, \pi_i = k) = \\ &= P(\pi_{i+1} = l, x_{i+1} x_{i+2} \dots x_N \mid \pi_i = k) f_k(i) = \\ &= P(x_{i+2} \dots x_N \mid \pi_{i+1} = l) P(x_{i+1} \mid \pi_{i+1} = l) P(\pi_{i+1} = l \mid \pi_i = k) f_k(i) = \\ &= b_l(i+1) e_l(x_{i+1}) a_{kl} f_k(i) \end{aligned}$$

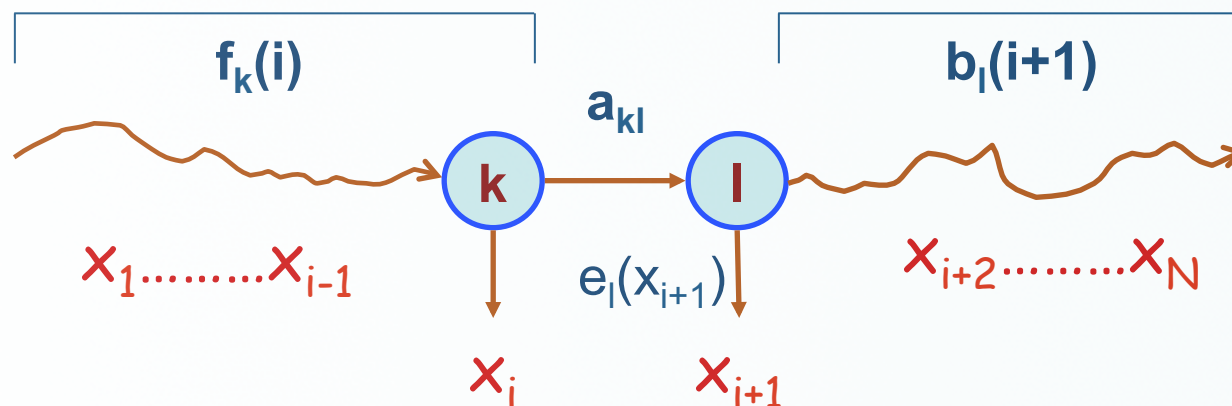
$$\text{So: } P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x \mid \theta_{\text{CURRENT}})}$$



# Estimating new parameters

- So,  $A_{kl}$  is the  $E[\# \text{ times transition } k \rightarrow l, \text{ given current } \theta]$

$$A_{kl} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_i \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x \mid \theta)}$$



- Similarly,

$$E_k(b) = [1/P(x \mid \theta)] \sum_{\{i \mid x_i = b\}} f_k(i) b_k(i)$$



# The Baum-Welch Algorithm

---

## Initialization:

Pick the best-guess for model parameters  
(or arbitrary)

## Iteration:

1. Forward
2. Backward
3. Calculate  $A_{kl}, E_k(b)$ , given  $\theta_{\text{CURRENT}}$
4. Calculate new model parameters  $\theta_{\text{NEW}} : a_{kl}, e_k(b)$
5. Calculate new log-likelihood  $P(x | \theta_{\text{NEW}})$

**GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION**

Until  $P(x | \theta)$  does not change much



# The Baum-Welch Algorithm

---



Time Complexity:

# iterations  $\times O(K^2N)$

- Guaranteed to increase the log likelihood  $P(x | \theta)$
- Not guaranteed to find globally best parameters

Converges to local optimum, depending on initial conditions

- Too many parameters / too large model: **Overtraining**

# Alternative: Viterbi Training



**Initialization:** Same

**Iteration:**

1. Perform Viterbi, to find  $\pi^*$
2. Calculate  $A_{kl}$ ,  $E_k(b)$  according to  $\pi^*$  + pseudocounts
3. Calculate the new parameters  $a_{kl}$ ,  $e_k(b)$

Until convergence

**Notes:**

- Not guaranteed to increase  $P(x | \theta)$
- Guaranteed to increase  $P(x | \theta, \pi^*)$
- In general, worse performance than Baum-Welch



---

# Pair-HMMs and CRFs

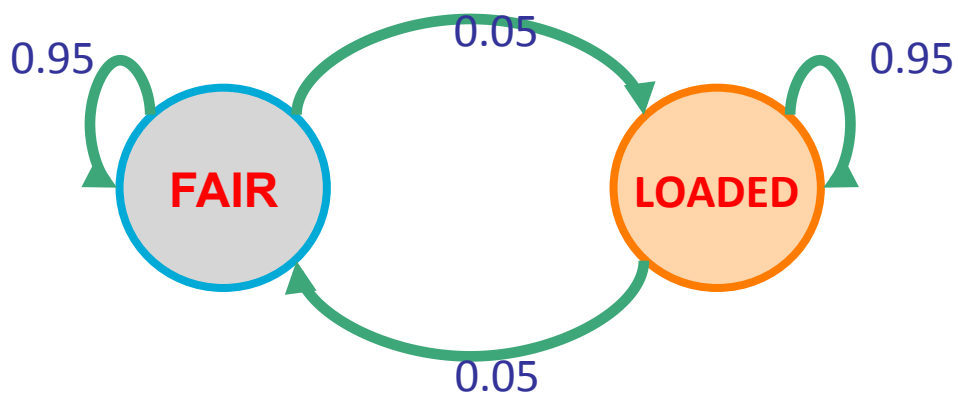
Slide Credits: Chuong B. Do



# Quick recap of HMMs

- Formally, an HMM =  $(\Sigma, Q, A, a_0, e)$ .
  - alphabet:  $\Sigma = \{b_1, \dots, b_M\}$
  - set of states:  $Q = \{1, \dots, K\}$
  - transition probabilities:  $A = [a_{ij}]$
  - initial state probabilities:  $a_{0i}$
  - emission probabilities:  $e_i(b_k)$

- Example:



# Pair-HMMs

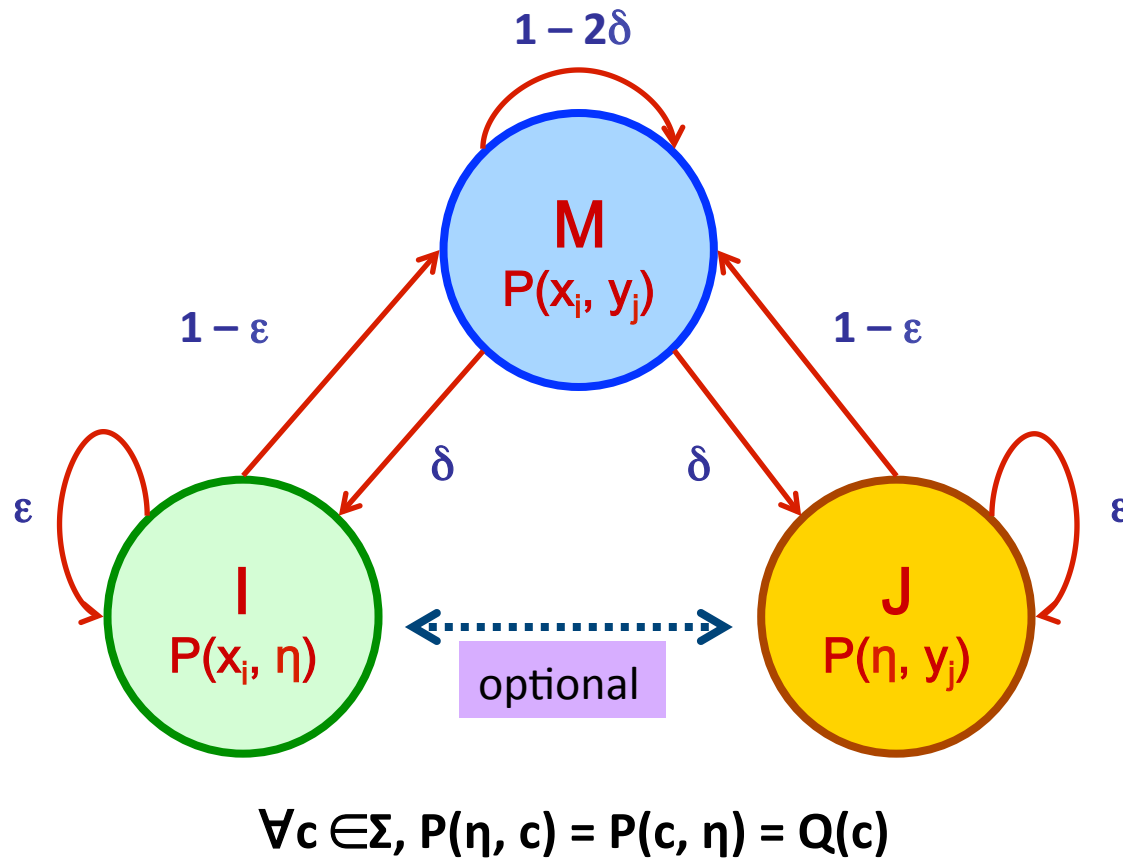


- Consider the HMM =  $((\Sigma_1 \cup \{\eta\}) \times (\Sigma_2 \cup \{\eta\}), Q, A, a_0, e)$ .
- Instead of emitting a pair of letters, in some states we may emit a letter paired with  $\eta$  (the empty string)
  - For simplicity, assume  $\eta$  is never emitted for both observation sequences simultaneously
  - Call the two observation sequences  $x$  and  $y$



# Application: sequence alignment

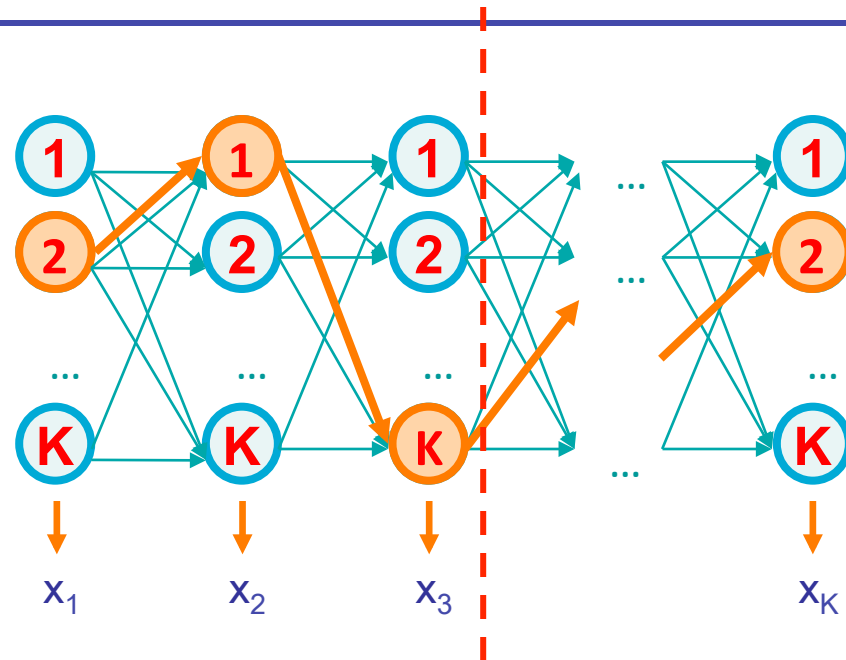
- Consider the following pair-HMM:



- QUESTION:** What are the interpretations of  $P(c,d)$  and  $Q(c)$  for  $c,d \in \Sigma$ ?
- QUESTION:** What does this model have to do with alignments?
- QUESTION:** What is the average length of a gapped region in alignments generated by this model? Average length of matched regions?



# Recap: Viterbi for single-sequence HMMs



- Algorithm:
  - $V_k(i) = \max_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1 \dots \pi_{i-1}, x_i, \pi_i = k)$
  - Compute using dynamic programming!

# (Broken) Viterbi for pair-HMMs



- In the single sequence case, we defined

$$\begin{aligned} V_k(i) &= \max_{\pi_1 \dots \pi_{i-1}} P(x_1 \dots x_{i-1}, \pi_1 \dots \pi_{i-1}, x_i, \pi_i = k) \\ &= e_k(x_i) \cdot \max_j a_{jk} V_j(i-1) \end{aligned}$$

- In the pairwise case,

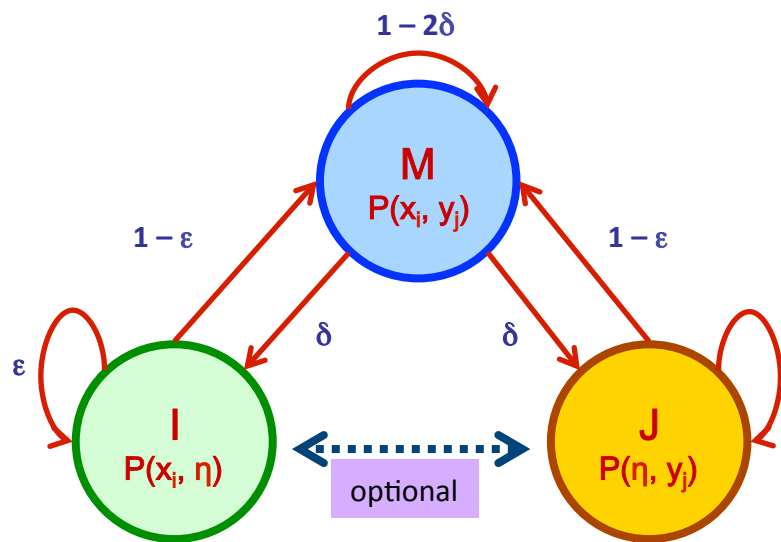
$(x_1, y_1) \dots (x_{i-1}, y_{i-1})$  no longer correspond to the first  $i - 1$  letters of  $x$  and  $y$





# (Fixed) Viterbi for pair-HMMs

- Consider this special case:



$$\forall c \in \Sigma, P(\eta, c) = P(c, \eta) = Q(c)$$

$$V_M(i, j) = P(x_i, y_j) \max \begin{cases} (1 - 2\delta) V_M(i - 1, j - 1) \\ (1 - \epsilon) V_I(i - 1, j - 1) \\ (1 - \epsilon) V_J(i - 1, j - 1) \end{cases}$$

$$V_I(i, j) = Q(x_i) \max \begin{cases} \delta V_M(i - 1, j) \\ \epsilon V_I(i - 1, j) \end{cases}$$

$$V_J(i, j) = Q(y_j) \max \begin{cases} \delta V_M(i, j - 1) \\ \epsilon V_J(i, j - 1) \end{cases}$$

- Similar for **forward/backward** algorithms
  - (see Durbin et al for details)

**QUESTION:** What's the computational complexity of DP?

# Connection to NW with affine gaps



$$V_M(i, j) = P(x_i, y_j) \max \begin{cases} (1 - 2\delta) V_M(i - 1, j - 1) \\ (1 - \varepsilon) V_I(i - 1, j - 1) \\ (1 - \varepsilon) V_J(i - 1, j - 1) \end{cases}$$

$$V_I(i, j) = Q(x_i) \max \begin{cases} \delta V_M(i - 1, j) \\ \varepsilon V_I(i - 1, j) \end{cases}$$

$$V_J(i, j) = Q(y_j) \max \begin{cases} \delta V_M(i, j - 1) \\ \varepsilon V_J(i, j - 1) \end{cases}$$

- **QUESTION:** How would the optimal alignment change if we divided the probability for every single alignment by  $\prod_{i=1}^{|x|} Q(x_i) \prod_{j=1}^{|y|} Q(y_j)$ ?

# Connection to NW with affine gaps



$$V_M(i, j) = \frac{P(x_i, y_j)}{Q(x_i) Q(y_j)} \max \begin{cases} (1 - 2\delta) V_M(i - 1, j - 1) \\ (1 - \varepsilon) V_I(i - 1, j - 1) \\ (1 - \varepsilon) V_J(i - 1, j - 1) \end{cases}$$

$$V_I(i, j) = \max \begin{cases} \delta V_M(i - 1, j) \\ \varepsilon V_I(i - 1, j) \end{cases}$$

$$V_J(i, j) = \max \begin{cases} \delta V_M(i, j - 1) \\ \varepsilon V_J(i, j - 1) \end{cases}$$

- Account for the extra terms “along the way.”



# Connection to NW with affine gaps

$$\log V_M(i, j) = \log \frac{P(x_i, y_j)}{Q(x_i) Q(y_j)} + \max \begin{cases} \log(1 - 2\delta) + \log V_M(i - 1, j - 1) \\ \log(1 - \varepsilon) + \log V_I(i - 1, j - 1) \\ \log(1 - \varepsilon) + \log V_J(i - 1, j - 1) \end{cases}$$

$$\log V_I(i, j) = \max \begin{cases} \log \delta + \log V_M(i - 1, j) \\ \log \varepsilon + \log V_I(i - 1, j) \end{cases}$$

$$\log V_J(i, j) = \max \begin{cases} \log \delta + \log V_M(i, j - 1) \\ \log \varepsilon + \log V_J(i, j - 1) \end{cases}$$

- Take logs, and ignore a couple terms.



# Connection to NW with affine gaps

---

$$M(i, j) = S(x_i, y_j) + \max \begin{cases} M(i - 1, j - 1) \\ l(i - 1, j - 1) \\ J(i - 1, j - 1) \end{cases}$$

$$l(i, j) = \max \begin{cases} d + M(i - 1, j) \\ e + l(i - 1, j) \end{cases}$$

$$J(i, j) = \max \begin{cases} d + M(i, j - 1) \\ e + J(i, j - 1) \end{cases}$$

- Rename!