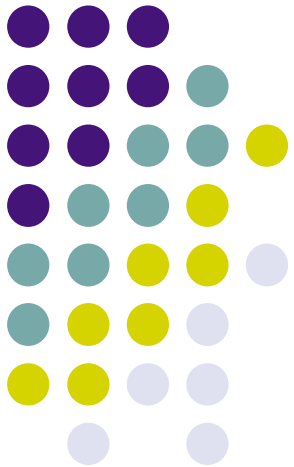
Graduate Computational Genomics

02-710 / 10-810 & MSCBIO2070

Elements of Statistics

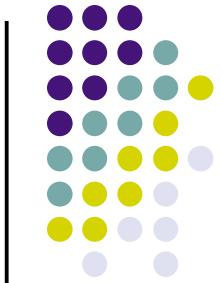


Takis Benos

Lecture #6b, February 1, 2007

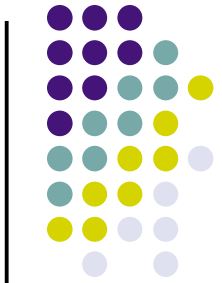
Reading: Durbin *et al.* "Biological Sequence Analysis", Ch. 3
Koski, "Hidden Markov Models for Bioinformatics", Ch. 1, 2

Outline of the Statistics part



- Some definitions
- Information measures
- Markov Chains
- Hidden Markov Models

Some definitions



- *Chain rule*: $P(X,Y,Z) = P(X|Y,Z)\ P(Y|Z)\ P(Z)$
- If $P(X|Y) = P(X)$ then X,Y *independent*

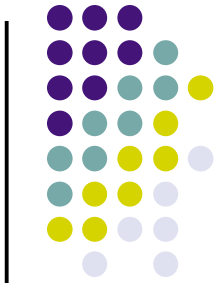
$$P(X,Y) = P(X)\ P(Y)$$

- *Conditional independence:*

$$P(X,Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z)$$

- *Marginal probability*: $P(X) = \sum_{Y} P(X,Y) = \sum_{Y} P(X \mid Y) P(Y)$

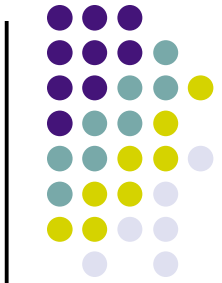
Bayes' Rule



$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} = \frac{P(Y \mid X)P(X)}{\sum_{x} P(Y \mid X)P(X)}$$

- *P(X), P(Y): prior probabilities*
- *P(X|Y), P(Y|X): posterior probabilities*
- *Posterior probabilities are the compromise between data and prior information*

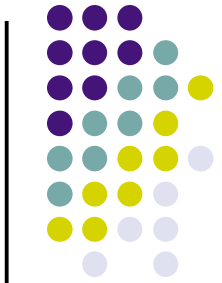
Bayes: application



- Problem:

A rare genetic disease is discovered with population frequency one in 1 million. An *extremely good* genetic test is 100% sensitive (always correct if you have the disease) and 99.99% specific (false positive 0.01%). Will you be willing to take such a test?

Bayes: application (cntd)



- *What are the chances to have the disease if it is positive?*

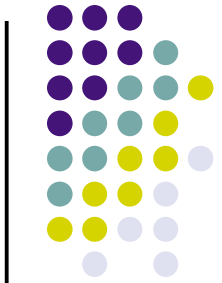
$$P(D \mid +) = P(+ \mid D)\, P(D) / P(+) =$$

$$= 1.0 * 10^{-6} / [1.0 * 10^{-6} + 10^{-4} * (1 - 10^{-6})] =$$

$$= 0.0099$$

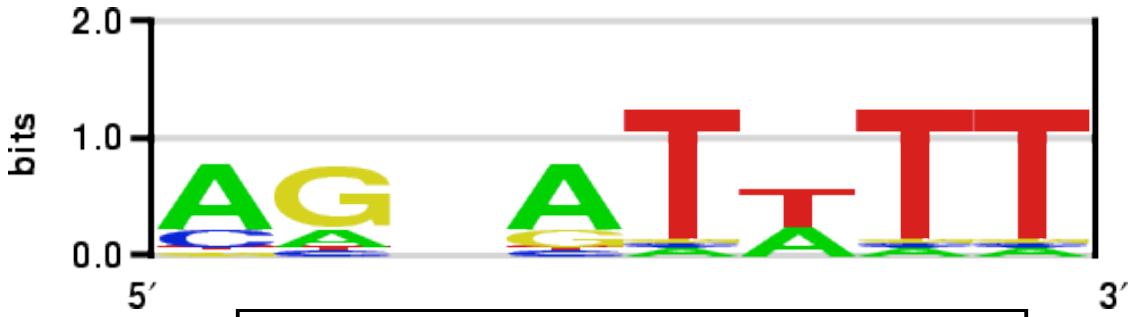
$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{\sum_x P(Y \mid X)P(X)}$$

Measures of information



- *Entropy* $$H(X) = \sum_i f_X(x_i) \cdot \log f_X(x_i)$$
- *Relative entropy* $$KL(X,Y) = \sum_i f_X(x_i) \cdot \log \frac{f_X(x_i)}{f_Y(x_i)}$$
- *Mutual information* $$I(X,Y) = \sum_i \sum_j f_{X,Y}(x_i, y_j) \cdot \log \frac{f_{X,Y}(x_i, y_j)}{f_X(x_i) \cdot f_Y(y_j)}$$

Entropy in LOGOs



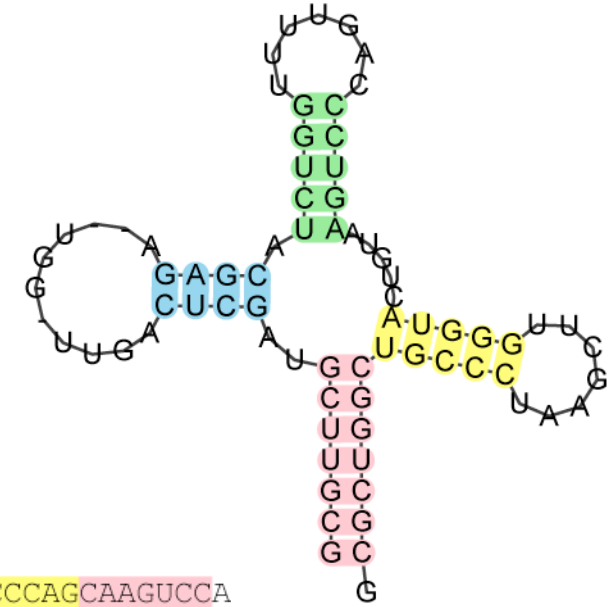
$$Height(i) = 2 + \sum_b f_i(b) \cdot \log_2 f_i(b)$$

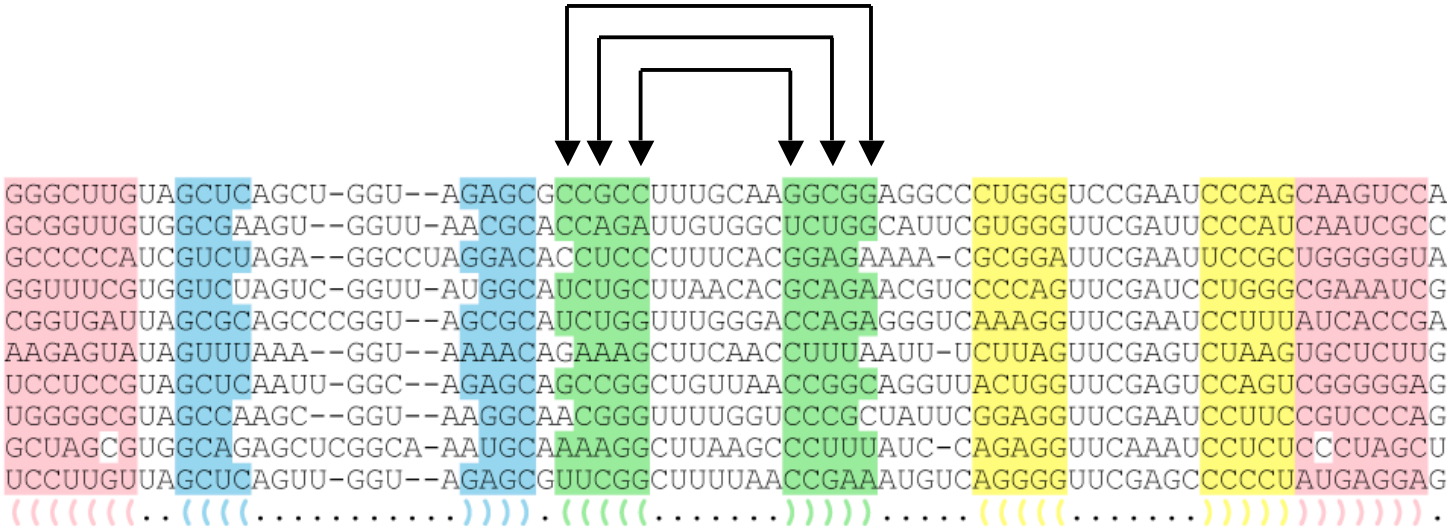
C-26	**AGGATATT**
C-57	**AGGATATT**
C-25	**CATATTTT**
7	**AGAGTTTT**
67	**AGCATTTT**

A	4	1	1	4	0	2	0	0
C	1	0	1	0	0	0	0	0
G	0	4	2	1	0	0	0	0
T	0	0	1	0	5	3	5	5

Mutual Information in RNA structure prediction

$$I(X,Y) = \sum_i \sum_j f_{X,Y}(x_i, y_j) \cdot \log \frac{f_{X,Y}(x_i, y_j)}{f_X(x_i) \cdot f_Y(y_j)}$$



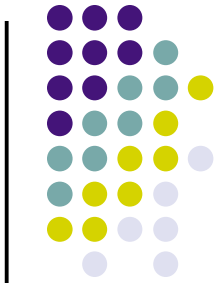


Markov chains

- What is a Markov chain?

- Markov chain of order *n* is a stochastic process of a series of outcomes, in which the probability of outcome *x* depends on the state of the previous *n* outcomes.

Markov chains (cntd)

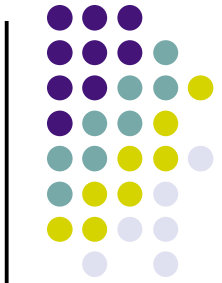


- Markov chain (of *first* order) and the *Chain Rule*

$$P(\vec{x}) = P(X_L, X_{L-1}, \ldots, X_1) =$$
$$= P(X_L \mid X_{L-1}, \ldots, X_1) P(X_{L-1}, X_{L-2}, \ldots, X_1) =$$
$$= P(X_L \mid X_{L-1}, \ldots, X_1) P(X_{L-1} \mid X_{L-2}, \ldots, X_1) \ldots P(X_1) =$$
$$= P(X_L \mid X_{L-1}) P(X_{L-1} \mid X_{L-2}) \ldots P(X_2 \mid X_1) P(X_1) =$$
$$= P(X_1) \prod_{i=2}^{L} P(X_i \mid X_{i-1})$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

Application of Markov chains: CpG islands

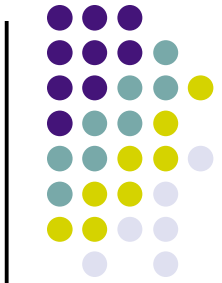


- CG is relatively rare in the genome due to high mutation of methyl-CG to TG
- In promoters, CG is usually unmethylated resulting in high frequency of CG

- Problem:

> Given two sets of sequences from the human genome, one with CpG islands and one without, can we calculate a model that can predict the CpG islands?

Application of Markov chains: CpG islands (cntd)



$P(x_2 \mid x_1, +)$

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

$P(x_2 \mid x_1, -)$

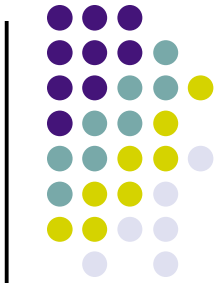
-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

$\log_2(P(x_2 \mid x_1,+)/P(x_2 \mid x_1,-))$

	A	C	G	T
A	-0.740	0.419	0.580	-0.803
C	-0.913	0.302	1.812	-0.685
G	-0.624	0.461	0.331	-0.730
T	-1.169	0.573	0.393	-0.679

$$\log_2 \frac{P(\vec{x} \mid +)}{P(\vec{x} \mid -)} = \sum_{i=1}^{L} \log_2 \frac{P(x_{i+1} \mid x_i, +)}{P(x_{i+1} \mid x_i, -)}$$

Hidden Markov Models (HMMs)

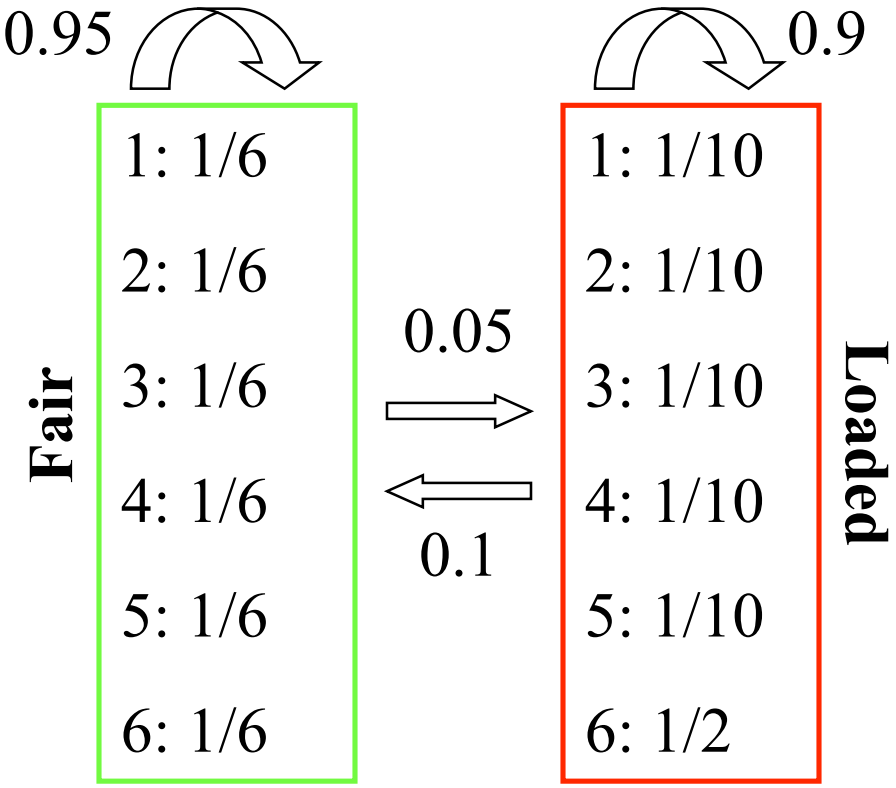


- What is a HMM?

  A Markov process in which the probability of an outcome depends also in a (hidden) random variable (state).

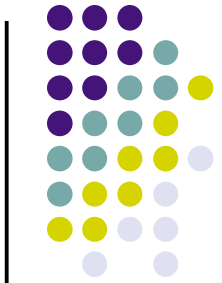
- Memory-less: future states affected only by current state
- We need:
  - ✓ Ω : alphabet of symbols (outcomes)
  - ✓ $\int$: set of states (hidden), each of which emits symbols
  - ✓ $A = (a_{kl})$: matrix of state transition probabilities
  - ✓ $E = (e_k(b)) = (P(x_i=b|\pi=k))$: matrix of emission probabilities

Example: the dishonest casino



- ✓ $\Omega = \{1, 2, 3, 4, 5, 6\}$
- ✓ $\int = \{F, L\}$
- ✓ A : $a_{FF}=0.95$, $a_{LL}=0.9$,
  $a_{FL}=0.05$, $a_{LF}=0.1$
- ✓ E : $e_F(b)=1/6$ $(\forall b \in \Omega)$,
  $e_L(\text{“6”})=1/2$
  $e_L(b)=1/10$ $(\text{if } b \neq 6)$

Three main questions on HMMs



1. **Evaluation**

   GIVEN HMM M, sequence x

   FIND $P(x \mid M)$

   ALGOR. Forward

2. **Decoding**

   GIVEN HMM M, sequence x

   FIND the sequence π of states that maximizes $P(\pi \mid x, M)$

   ALGOR. Viterbi, Forward-Backward

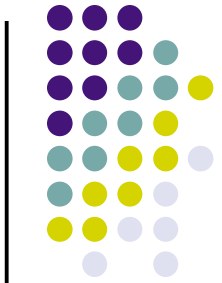
3. **Learning**

   GIVEN HMM M, with unknown prob. parameters, sequence x

   FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$

   ALGOR. Maximum likelihood (ML), Baum-Welch (EM)

Problem 1: Evaluation

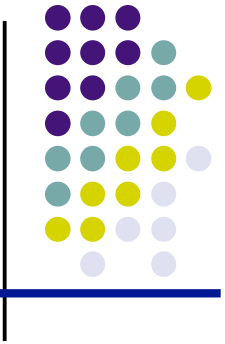


Find the likelihood a given sequence is generated by a particular model

E.g. Given the following sequence is it more likely that it comes from a ***L****oaded* or a ***F****air* die?

12341231626163646162341122134 1

Problem 1: Evaluation (cntd)



12341231626163646162341122134 1

$$P(Data \mid F_1 \ldots F_{30}) = \prod_{i=1}^{30} a_{F,F} \cdot e_F(b_i) =$$

$$= (1/6)^{30} \cdot 0.95^{29} = 4.52 \cdot 10^{-24} \cdot 0.226 =$$

$$= 1.02 \cdot 10^{-24}$$

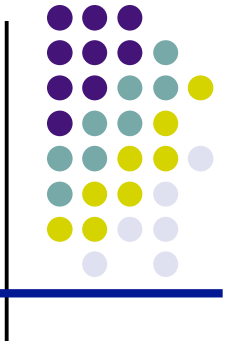
$$P(Data \mid L_1 \ldots L_{30}) = \prod_{i=1}^{30} a_{L,L} \cdot e_L(b_i) =$$

$$= (1/2)^{6} \cdot (1/10)^{24} \cdot 0.90^{29} = 1.56 \cdot 10^{-26} \cdot 0.047 =$$

$$= 7.36 \cdot 10^{-28}$$

What happens in a sliding window?

Problem 2: Decoding

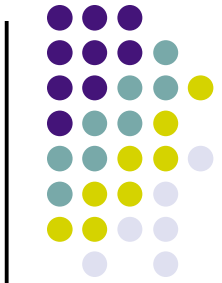


Given a point x_i in a sequence find its most probable state

E.g. Given the following sequence is it more likely that the 3rd observed "6" comes from a ***L****oaded* or a ***F****air* die?

12341231626163646162341122134 1

The Forward Algorithm - derivation



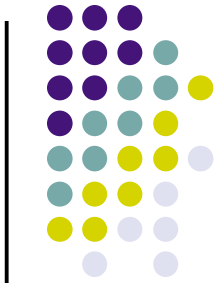
- In order to calculate P(x_i) = probability of x_i, given the HMM, we need to sum over all possible ways of generating x_i:

$$P(x_i) = \sum_{\pi} P(x_i, \pi) = \sum_{\pi} P(x_i \mid \pi) \cdot P(\pi)$$

- To avoid summing over an exponential number of paths π, we first define the *forward probability*:

$$f_k(i) = P(x_1 \ldots x_i, \pi_i = k)$$

The Forward Algorithm – derivation (cntd)

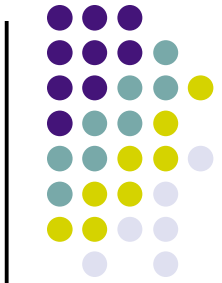


- Then, we need to write the $f_k(i)$ as a function of the previous state, $f_l(i\text{-}1)$.

$$
\begin{aligned}
f_k(i) &= P(x_1,\ldots,x_{i-1},\pi_i = k) \\
&= \sum_{\pi_1,\ldots,\pi_{i-1}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1},\pi_i = k)\cdot e_k(x_i) \\
&= \sum_l \left(\sum_{\pi_1,\ldots,\pi_{i-2}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-2},\pi_{i-1} = l)\cdot a_{l,k} \right) \cdot e_k(x_i) \\
&= \sum_l P(x_1,\ldots,x_{i-1},\pi_{i-1} = l)\cdot a_{l,k}\cdot e_k(x_i) \\
&= e_k(x_i)\cdot \sum_l f_l(i-1)\cdot a_{l,k}
\end{aligned}
$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

The Forward Algorithm



We can compute $f_k(i)$ for all k, i, using dynamic programming

Initialization:

$$f_0(0) = 1$$

$$f_k(0) = 0, \quad \forall k > 0$$

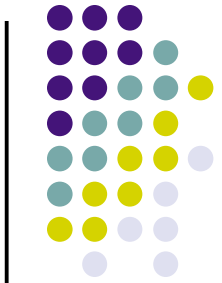
Iteration:

$$f_k(i) = e_k(x_i) \cdot \sum_l f_l(i-1) \cdot a_{l,k}$$

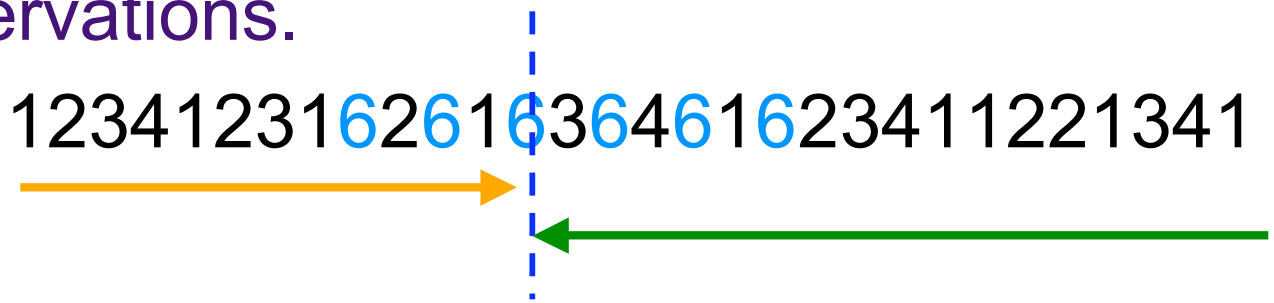
Termination:

$$P(\vec{x}) = \sum_k f_k(N) \cdot a_{k,0}$$

The Backward Algorithm

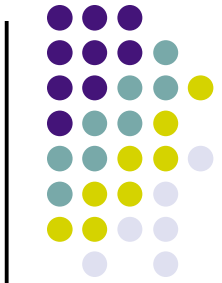


- Forward algorithm determines the most likely state *k* at position *i*, using the *previous* observations.



- What if we started from the end?

The Backward Algorithm – derivation



- We define the *backward probability*:

$$b_k(i) = P(x_{i+1},\ldots,x_N \mid \pi_i = k)$$

$$= \sum_{\pi_{i+1},\ldots,\pi_N} P(x_{i+1},\ldots,x_N,\pi_{i+1},\ldots,\pi_N \mid \pi_i = k)$$

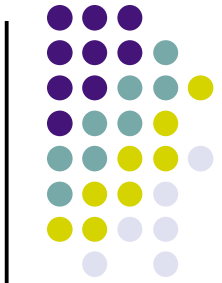
$$= \sum_l \sum_{\pi_{i+1},\ldots,\pi_N} P(x_{i+1},\ldots,x_N,\pi_{i+1} = l,\pi_{i+2},\ldots,\pi_N \mid \pi_i = k)$$

$$= \sum_l e_k(x_{i+1}) \cdot a_{k,l} \cdot \sum_{\pi_{i+2},\ldots,\pi_N} P(x_{i+2},\ldots,x_N,\pi_{i+2},\ldots,\pi_N \mid \pi_{i+i} = l)$$

$$= \sum_l b_l(i+1) \cdot a_{k,l} \cdot e_l(x_{i+1})$$

Chain rule: *P(A,B,C)=P(C|A,B) P(B|A) P(A)*

The Backward Algorithm



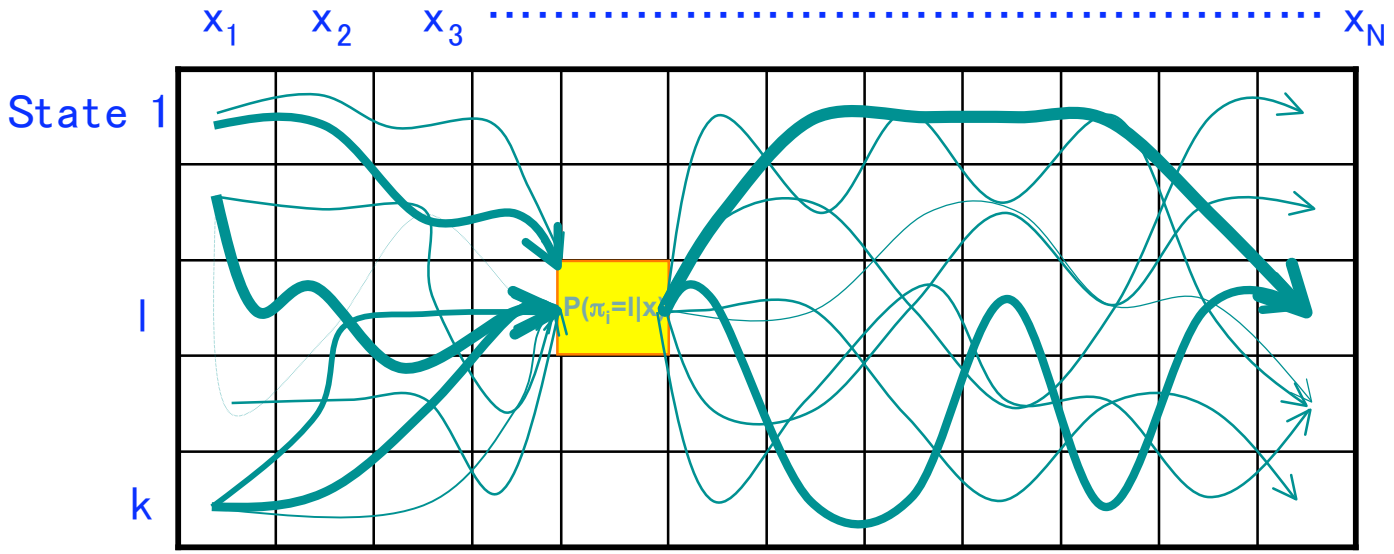
We can compute $b_k(i)$ for all k, i, using dynamic programming

Initialization: $$b_k(N) = a_{k,0}, \quad \forall k$$

Iteration: $$b_k(i) = \sum_l e_k(x_{i+1}) \cdot b_l(i+1) \cdot a_{k,l}$$

Termination: $$P(\vec{x}) = \sum_k b_k(1) \cdot a_{0,k} \cdot e_k(x_1)$$

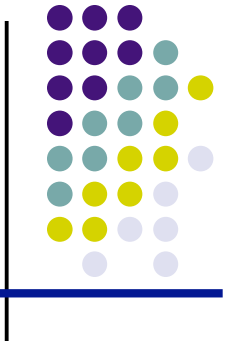
Posterior Decoding



- *Posterior decoding* calculates the optimal path that explains the data.

- For each emitted symbol, x_i, it finds the most likely state that could produce it, based on the *forward* and *backward* probabilities.

Posterior Decoding (cntd)



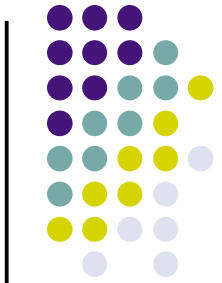
- It finds the $\hat{\pi}_i = \arg\max_k P(\pi_i = k \mid \vec{x})$ where:

$$P(\pi_i = k \mid \vec{x}) = \frac{P(\pi_i = k \wedge \vec{x})}{P(\vec{x})} = \frac{f_k(i) \cdot b_k(i)}{P(\vec{x})}$$

- Note that this is for a single sequence point *x(i)* only.

- *Tip:* these algorithms have a lot of small number calculations
  Solution: use logs!

The Viterbi Algorithm – derivation



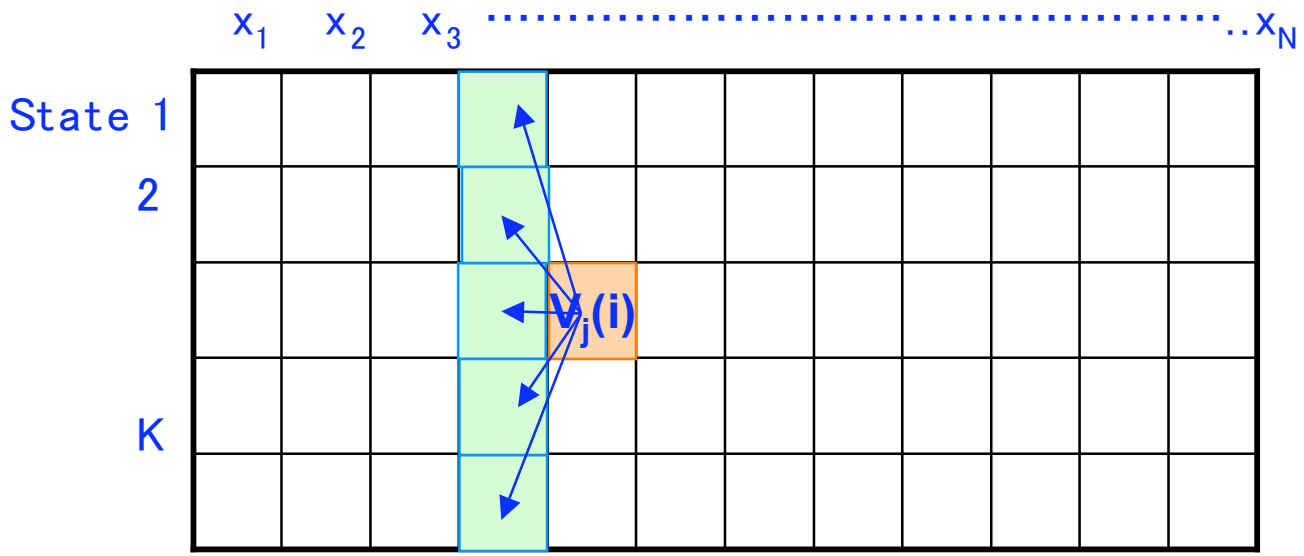
- Now we define:

$$V_k(i) = \max_{\{\pi_1,\ldots,\pi_{i-1}\}} P(x_1,\ldots,x_{i-1},\pi_1,\ldots,\pi_{i-1},\pi_i = k)$$

- Then, we need to write the $V_k(i)$ as a function of the previous state, $V_l(i\text{-}1)$.

$$V_k(i) = \ldots = e_k(x_i) \cdot \max_l \{a_{l,k} \cdot V_l(i-1)\}$$

The Viterbi Algorithm

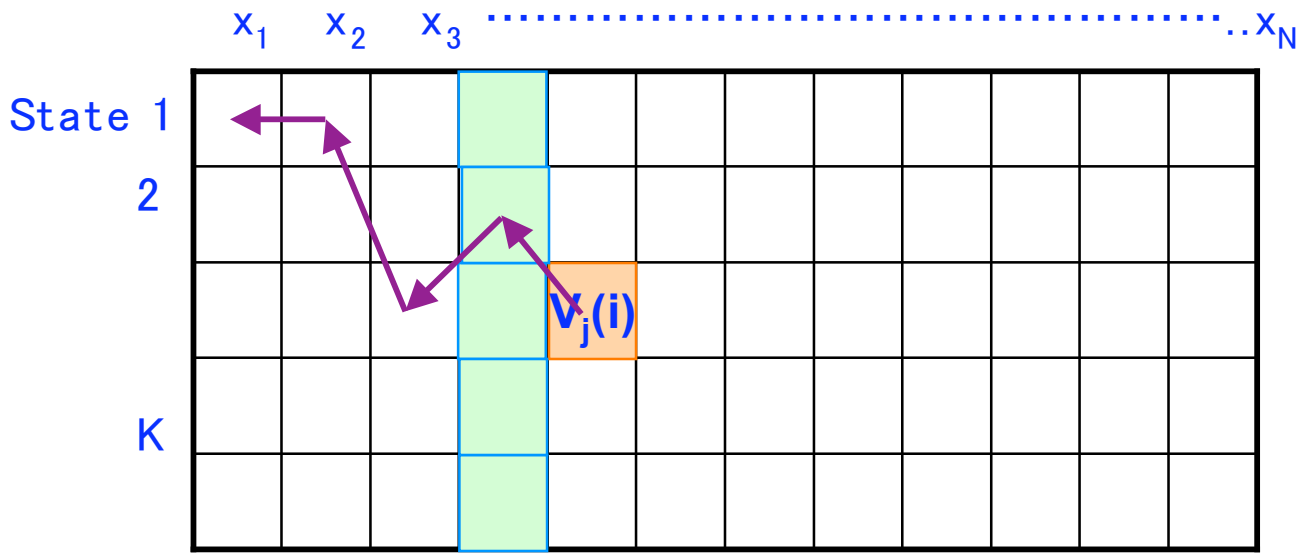


Similar to "aligning" a set of states to a sequence

Dynamic programming!

Viterbi decoding: traceback

The Viterbi Algorithm

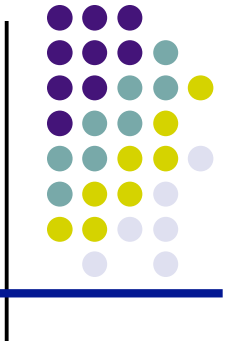


Similar to "aligning" a set of states to a sequence

Dynamic programming!

Viterbi decoding: traceback

Viterbi, Forward, Backward



VITERBI

Initialization:

$V_0(0) = 1$

$V_k(0) = 0$, for all $k > 0$

Iteration:

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

Termination:

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

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

FORWARD

Initialization:

$f_0(0) = 1$

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

Iteration:

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

Termination:

$P(x) = \mathbf{\Sigma_k}\ f_k(N)\ a_{k0}$

Time: $O(K^2N)$

BACKWARD

Initialization:

$b_k(N) = a_{k0}$, for all k

Iteration:

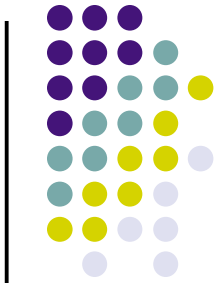
$b_l(i) = \mathbf{\Sigma_k}\ e_l(x_i{+}1)\ a_{kl}\ b_k(i{+}1)$

Termination:

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

Space: $O(KN)$

Three main questions on HMMs



1. **Evaluation**

   GIVEN HMM M, sequence x

   FIND $P(x \mid M)$

   ALGOR. Forward

2. **Decoding**

   GIVEN HMM M, sequence x

   FIND the sequence π of states that maximizes $P(\pi \mid x, M)$

   ALGOR. Viterbi, Forward-Backward

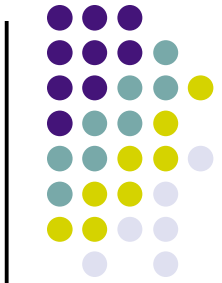
3. **Learning**

   GIVEN HMM M, with unknown prob. parameters, sequence x

   FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$

   ALGOR. Maximum likelihood (ML), Baum-Welch (EM)

Three main questions on HMMs



✓ **Evaluation**

GIVEN HMM M, sequence x
FIND $P(x \mid M)$
ALGOR. Forward

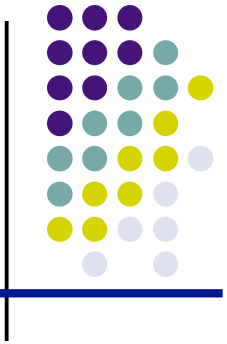
✓ **Decoding**

GIVEN HMM M, sequence x
FIND the sequence π of states that maximizes $P(\pi \mid x, M)$
ALGOR. Viterbi, Forward-Backward

3. Learning

GIVEN HMM M, with unknown prob. parameters, sequence x
FIND parameters $\theta = (\pi, e_{ij}, a_{kl})$ that maximize $P(x \mid \theta, M)$
ALGOR. Maximum likelihood (ML), Baum-Welch (EM)

Problem 3: Learning



Given a model (structure) and data,
calculate model's parameters

Two scenarios:

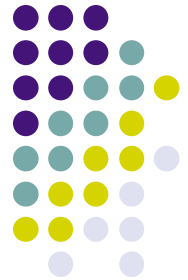
- Labeled data - Supervised learning

12341231	62616364616	23411221341
Fair	Loaded	Fair

- Unlabeled data - Unsupervised learning

12341231626163646162341122134 1

Two learning scenarios - examples



1. **Supervised learning**

   Examples:

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

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

2. **Unsupervised learning**

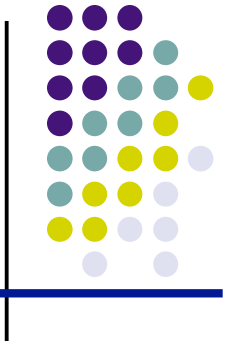
   Examples:

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

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

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

Supervised learning



- Given $x = x_1 \ldots x_N$ for which the true state path $\pi = \pi_1 \ldots \pi_N$ is known
  - **Define:**

    $A_{k,l}$ = # times state transition $k \rightarrow l$ transition occurs in π

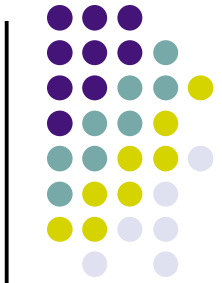
    $E_k(b)$ = # times state k in π emits b in x

  - The maximum likelihood parameters θ *are:*

$$a_{k,l}^{ML} = \frac{A_{k,l}}{\sum_i A_{k,i}} \qquad e_k^{ML}(b) = \frac{E_k(b)}{\sum_c E_k(c)}$$

- Problem: overfitting (when training set is small for the model)

Overfitting



- **Example**
  - Given 10 casino rolls, we observe

$$\mathbf{x} = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3$$
$$\boldsymbol{\pi} = F, F, F, F, F, F, F, F, F, F$$

  - Then:

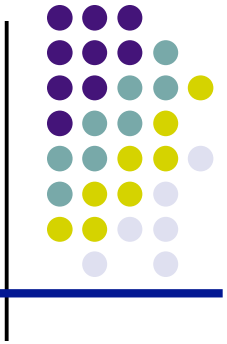
$a_{FF} = 10/10 = 1$; $a_{FL} = 0/10 = 0$

$e_F(1) = e_F(3) = 2/10 = 0.2$;

$e_F(2) = 3/10 = 0.3$; $e_F(4) = 0/10 = 0$; $e_F(5) = e_F(6) = 1/10 = 0.1$

- **Solution:** add *pseudocounts*
  - Larger pseudocounts ⇒ strong prior belief (need a lot of data to change)
  - Smaller pseudocounts ⇒ just smoothing (to avoid zero probabilities)

Unsupervised learning - ML



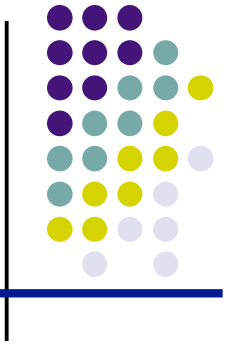
- Given $x = x_1 \ldots x_N$ for which the true state path $\pi = \pi_1 \ldots \pi_N$ is unknown

  - **EXPECTATION MAXIMIZATION (EM) in a nutshell**

  0. Initialize the parameters θ of the model M
  1. Calculate the *expected* values of $A_{k,l}$, $E_k(b)$ based on the training data and current parameters
  2. Update θ according to $A_{k,l}$, $E_k(b)$ as in supervised learning
  3. Repeat #1 & #2 until convergence

  - In HMM training, this is also called the Baum-Welch Algorithm

The Baum-Welch algorithm



- Initialization: pick arbitrary model parameters
- Recurrence:
  1. Set *A* and *E* to pseudocounts
  2. Calculate $f_k(i)$ and $b_k(i)$ for each training sequence *j*
  3. Add the contribution of seq *j* to *A* and *E*:

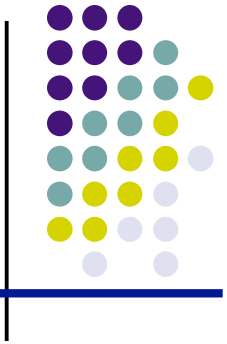
$$P(\pi_i = k, \pi_{i+1} = l \mid x, \theta)$$

$$A_{k,l} = \sum_i f_k(i) \cdot a_{k,l} \cdot e_l(x_{i+1}) \cdot b_l(i+1) / P(\vec{x}) \qquad E_k(b) = \sum_{\{i \mid x_i b\}} f_k(i) \cdot b_k(i) / P(\vec{x})$$

  4. Calculate new model parameters $a_{k,l}$ and $e_k(b)$
  5. Calculate the new (log)likelihood of the model

- Termination: if *Δlog-likelihood < threshold* or Ntimes>max_times

The Baum-Welch algorithm



- Time complexity:
  - # iterations x $O(K^2N)$
- Guaranteed to increase the log-likelihood $P(x \mid \theta)$
- Not guaranteed to find globally optimal parameters
  - Converges to a local optimum, depending on initial conditions
- Too many parameters / too large model $\Rightarrow$ Overtraining

Acknowledgements

Some of the slides used in this lecture are adapted or modified slides from lectures of:

- Serafim Batzoglou, Stanford University
- Eric Xing, Carnegie-Mellon University

Theory and examples from the following books:

- T. Koski, "*Hidden Markov Models for Bioinformatics*", 2001, Kluwer Academic Publishers
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison, "Biological Sequence Analysis", 1998, Cambridge University Press