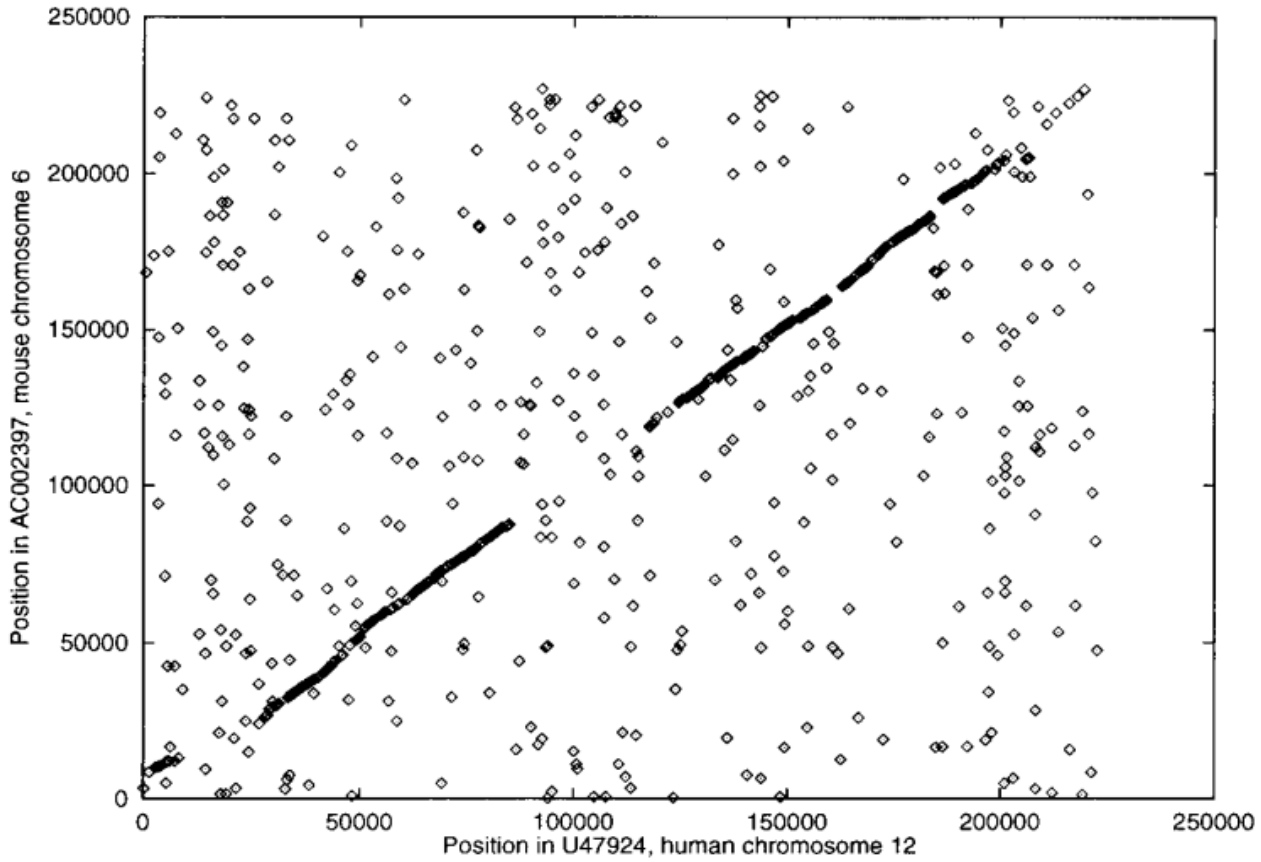
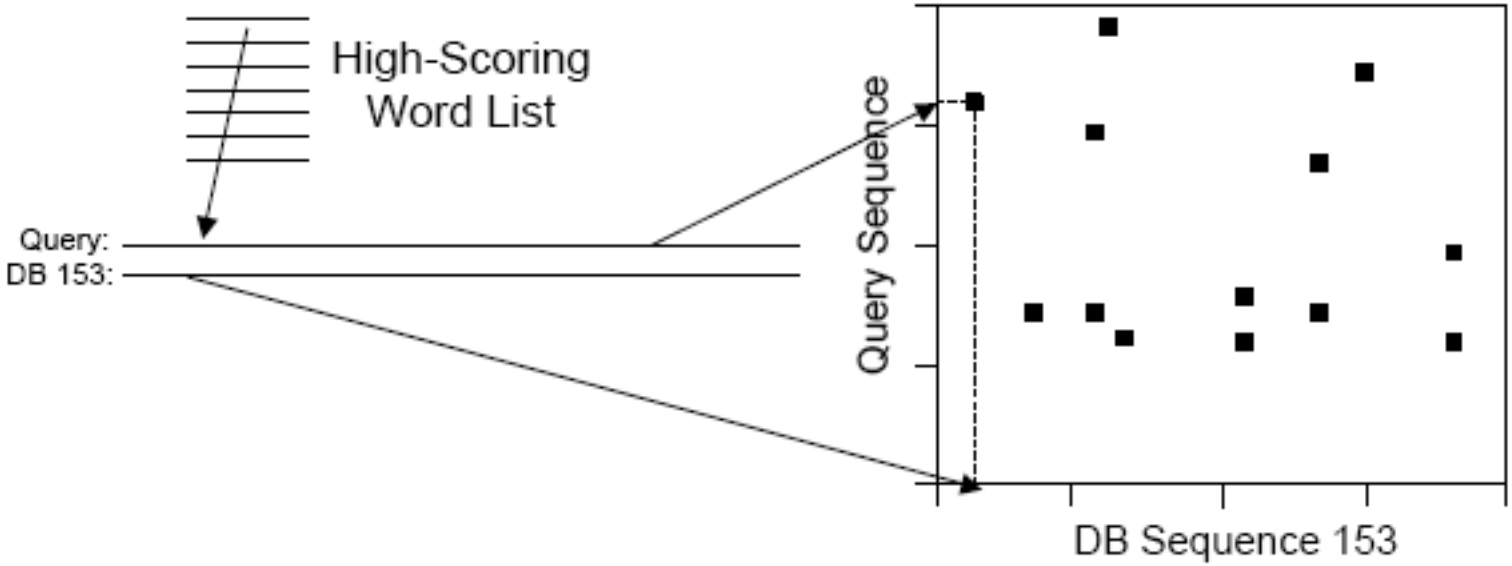
Review: Human-Mouse Dot Plot



BLAST − high level overview



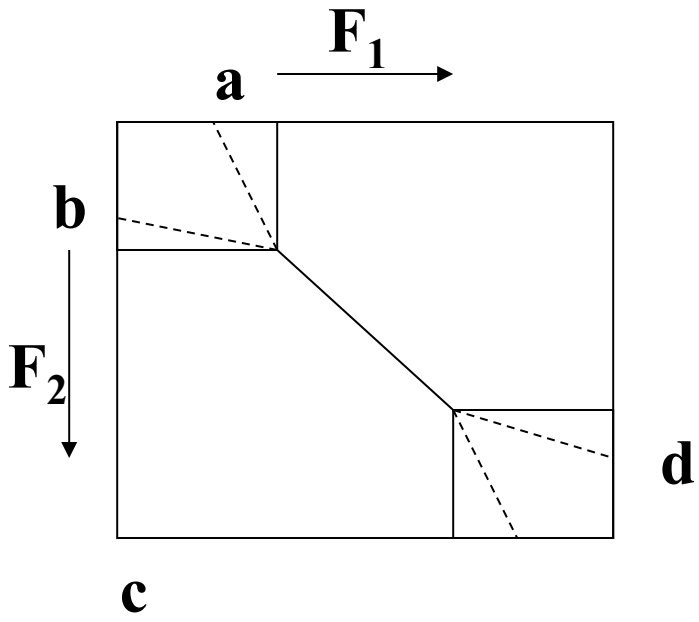
Finding runs and extending them (old version)

- Extend alignments “greedily” off each end; stop when the score drops below a threshold
- Alignments are ungapped, so extension is straightforward.
- All hits whose score is greater than a minimum score S are displayed.

Gapped BLAST

- Require two hits on the same “diagonal”
- Hits must be less than a specified distance away (antidiagonal difference). Alignment is explored in between matches (banded variant)
- Substantially reduces the number of extensions, and is a better heuristic in terms of biological value

Banded Dynamic Programming



– Compute only lower and upper rectangles based on desired percent similarity. Also an *exclusion method*

Finding matches faster/better

Let:

A = 0 (00) C = 1 (01) G = 2 (10) T = 3 (11)

Strings can be converted based on the binary string they represent

String *Binary* *Integer*

AAA = 000000 = 0
ATA = 001100 = 12
AAC = 000001 = 1

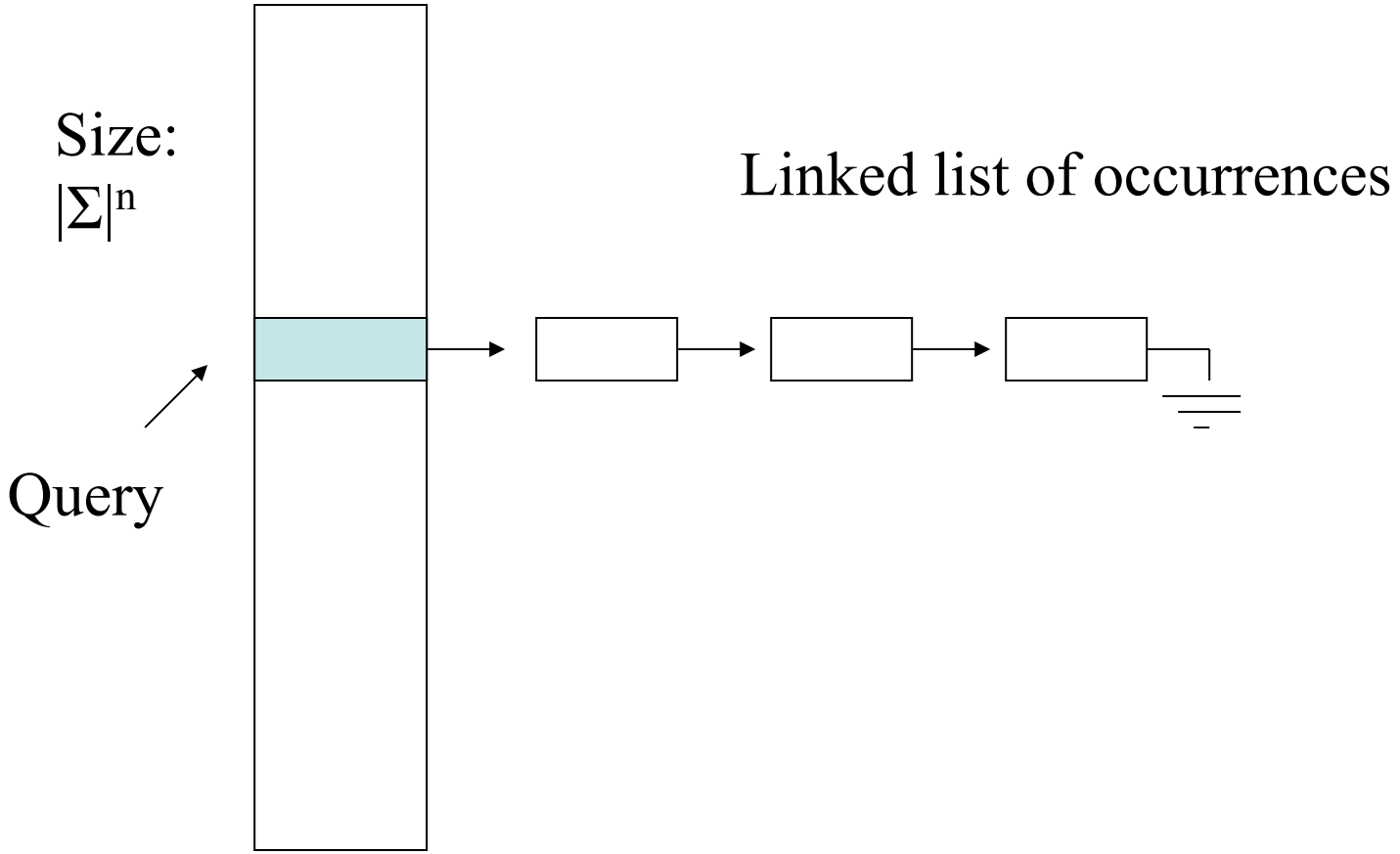
Look-up tables

- Strings of size *k* over Σ can be represented by an integer index i, $0 \leq i \leq |\Sigma|^k - 1$.

- DNA is composed of four characters.
  - $\Sigma = \{A, G, C, T\}$ $|\Sigma| = 4$

- We can preprocess a database into a lookup table to locate all occurrences of a query index.
  - Linear time and linear space

Search using an index



Indexing discussion

- A database can be preprocessed in linear time to allow locating all instances of a short string.

- Major limitation is it limits searching to fixed length strings.
  - This is used heavily, though, in sequence assembly

Affine gap penalties

- To date, we have considered constant gap penalties.

- However, nature doesn't necessarily work this way. For example which do you think is more likely?

```
ATA--GC        ATAG-GC
ATATTGC        AT-GTGC
```

Gap scoring

- Ideally, we would like a gap of length *l* to be penalized by -($a + bl$)
- *a* is called the gap open penalty
- *b* is the gap extension penalty

Runtime

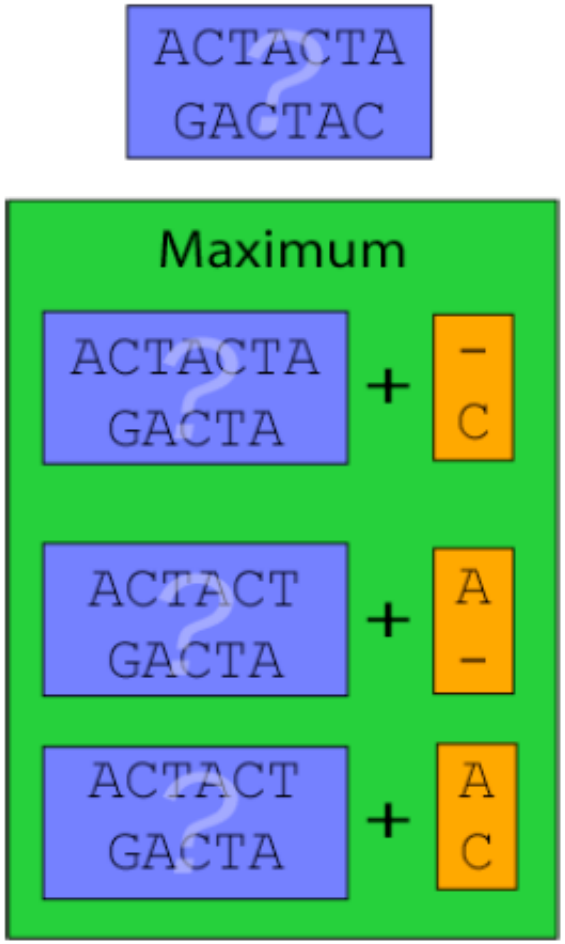
- Implementing an affine gap penalty in the way we understand requires $O(n)$ work per cell to work.

- Given we have $O(n^2)$ cells, this is an $O(n^3)$ algorithm. Can we do better?

Flashback:

We can solve this recursively based on looking at three smaller problems

$$T[i,j] = \max \begin{cases} T[i-1,j-1] + score(s[i], t[j]) \\ T[i-1,j] + g \\ T[i,j-1] + g \end{cases}$$



Solution

- We will use 4 tables instead of 1:
  - *V* - stores best alignment between *s*[1..*i*] and *t*[1..*j*]
  - *G* - stores best alignment between *s*[1..*i*], *t*[1..*j*], *i.e.*, *s*[*i*] aligned to *t*[*j*]
  - *E* - best alignment between *s*[1..*i*], *t*[1..*j*] ending with a gap in *s*
  - *F* - best alignment between s[*1*..i], t[1..*j*], ending with a gap in *t*
- As before, best global alignment is *V*[*m*,*n*] if |s| is *m* and |*t*| = *n*

Updated recurrences

- V[*i,j*] = max{E[*i,j*], F[*i,j*],G[*i,j*]}
- G[*i,j*] = V[*i*-1,*j*-1] + score(*s*[*i*],*t*[*j*])
- E[*i,j*] = max{E[*i,j*-1],V[*i,j*-1] - *gap_open*} - *gap_extend*
- F[*i,j*] = max{F[*i*-1, *j*], V[*i*-1, *j*] - *gap_open*} - *gap_extend*

Gusfield's notation

Table G

- $G[i,j] = V[i-1,j-1] + \text{score}(s[i],t[j])$

  - V is the best one from somewhere
  - We match score($s[i]$,$t[j]$) to diagonal value
  - By design, *G* is the best alignment that ends on a match

Table E

- $E[i,j] = \max\{E[i,j-1], V[i,j-1] - gap\_open\} - gap\_extend$

  - No matter what, we are extending a gap
  - Two options
    - Add a gap to an existing gap in *s* (thus use of E in part 1)
    - Open a new gap in *s*
  - This gives us the best alignment ending in a gap in *s*

Table F

- F[*i*,*j*] = max{*F*[*i*-1, *j*], V[*i*-1,*j*] - *gap_open*} - *gap_extend*

  - No matter what, we are extending a gap here too
  - Two options
    - Add a gap to an existing gap in *t* (thus use of F in part 1)
    - Open a new gap in *t*
  - This gives us the best alignment ending in a gap in *t* per the original idea

What? Four tables?

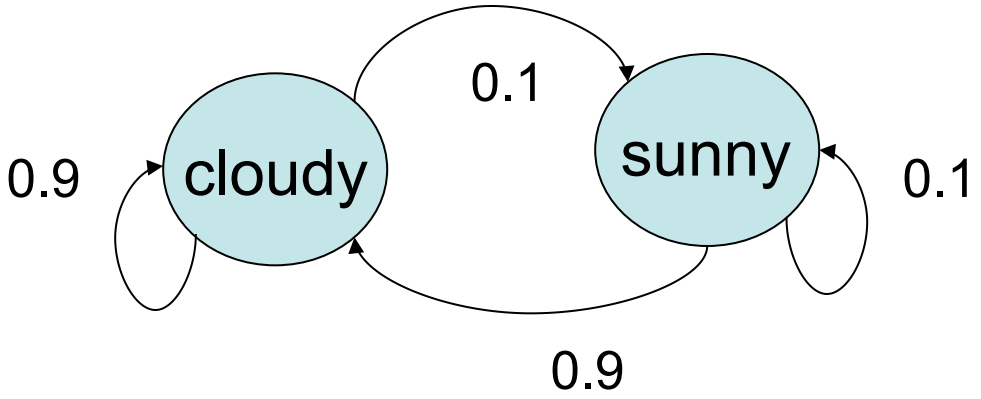
- It is possible to derive a recurrence that uses only two tables, denoted *M* and *I* per Durbin's terminology:
  - *M* = match
  - *I* = insertion
- This approach is only guaranteed to find the optimal alignment when lowest mismatch score is >= -2 *gap_extend*

Applications for next week (hopefully)

- Comparing different genome assemblies
- Gene finding through comparative genomics
- Analyzing pathogenic bacteria against their harmless close relatives

South Bend winters: 1890

Tomorrow's weather



Markov models, revisited

- We can also view Markov models as a discrete (finite) system:
  - *N* total states
  - Start at some initial state (*t* = 1) based on multinomial model
  - System proceeds to the next state based on probabilities given current state
  - Also called a *probabilistic finite automata*

Example: play calling

- Suppose we simplify the ND offensive playbook into three plays:
  - Run
  - Pass short
  - Long pass
- Further, lets suppose there are two at most two offensive coaches:
  - Coach Brian Kelly
  - Offensive coordinator Chip Long

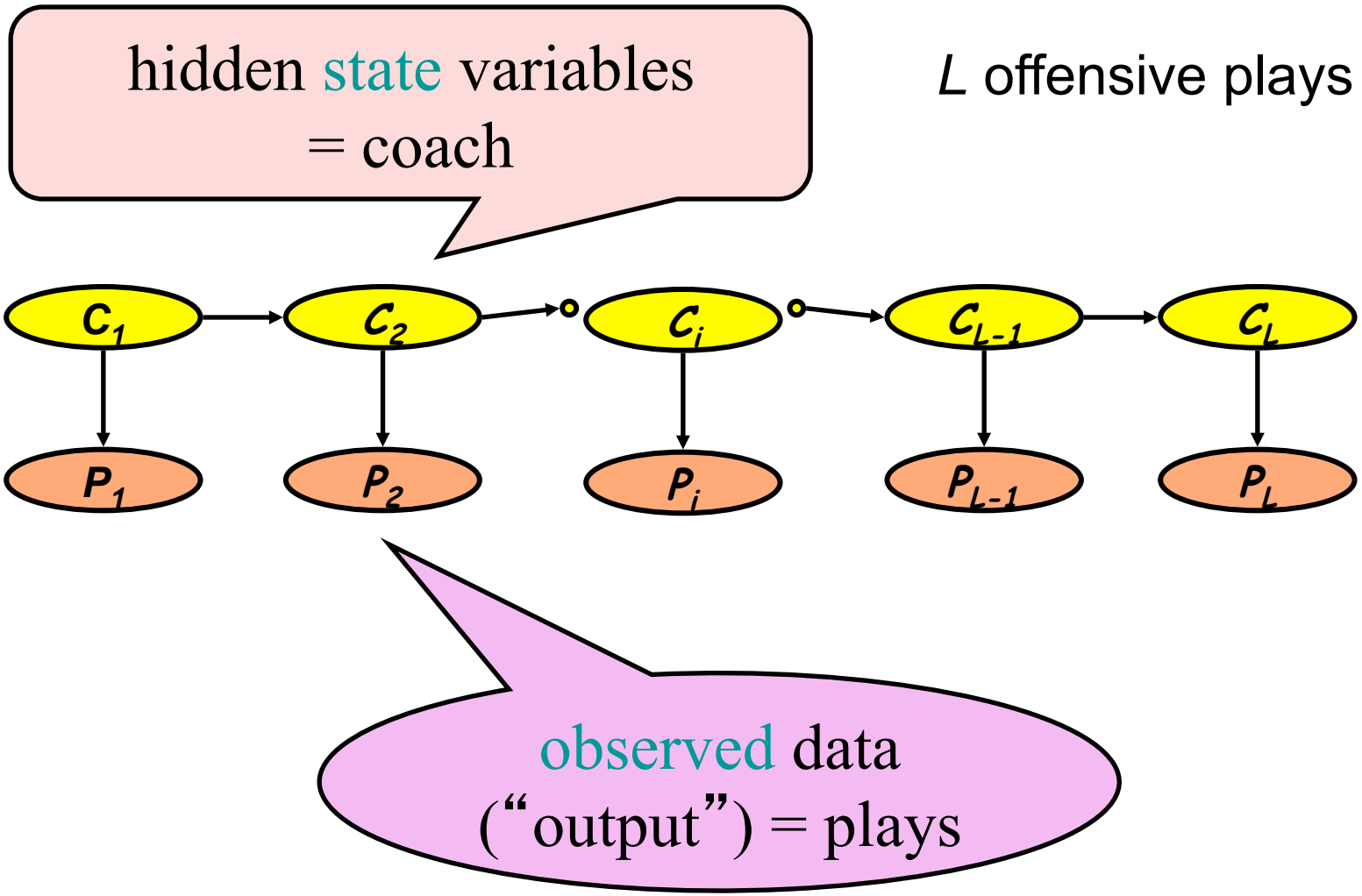
Play calling

- Coach Kelly:
  - P(run) = 0.1
  - P(short pass) = 0.8
  - P(long pass) = 0.1
- Offensive coordinator Molnar:
  - P(run) = 0.8:
  - P(short pass) = 0.05
  - P(long pass) = 0.15

Hidden model

- As fans, we can not tell who is calling the plays, all we can observe is the play called
- We assume play calls are based on coach tendencies (output) probabilities

ND Football Game



Barbecue begging

- A puppy smells a number of neighbors barbecuing. One unsupervised grill is two houses downhill from his yard, and another unsupervised grill is three houses uphill from his yard. Because so many people are barbecuing, he goes randomly from house to house in search of food, going downhill with twice the probability that he goes uphill. We record his progress from house to house, using 0 to stand for one unsupervised grill, 2 to stand for his yard, and 5 to stand for the other unsupervised grill.

www.mathforum.org

Hidden Markov Models

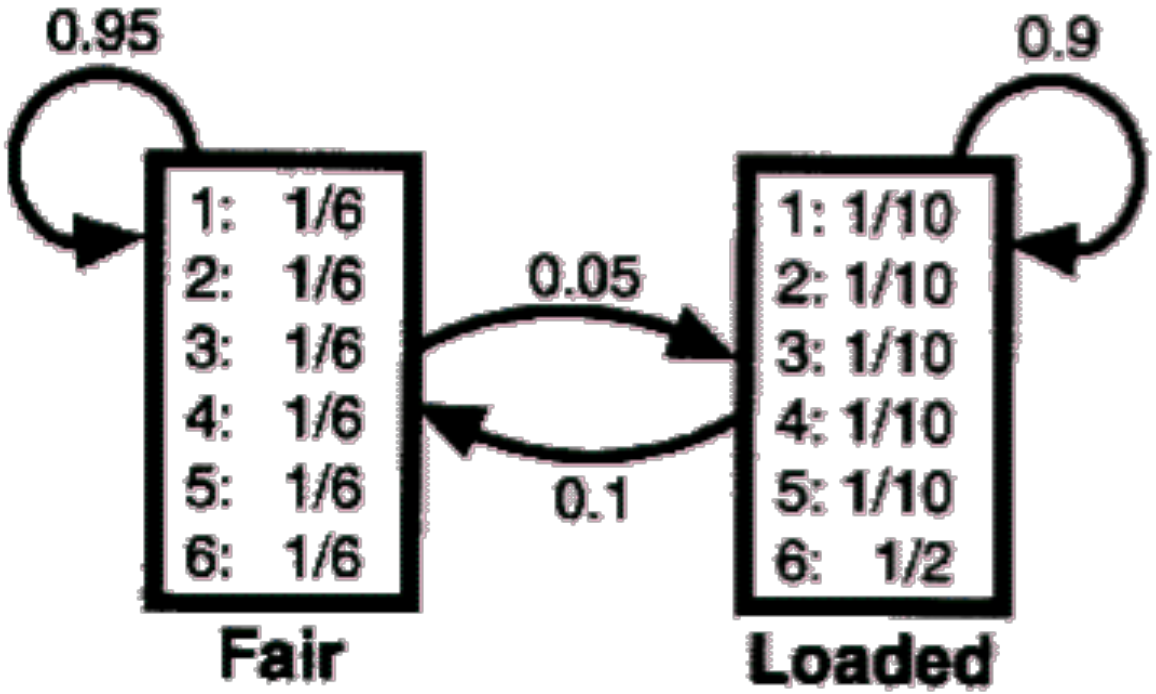
- Used when states can not directly be observed, good for noisy data

- Requirements:
  - A finite number of states, each with an output probability distribution
  - State transition probabilities
  - Observed phenomenon, which can be randomly generated given state-associated probabilities.

Example goals

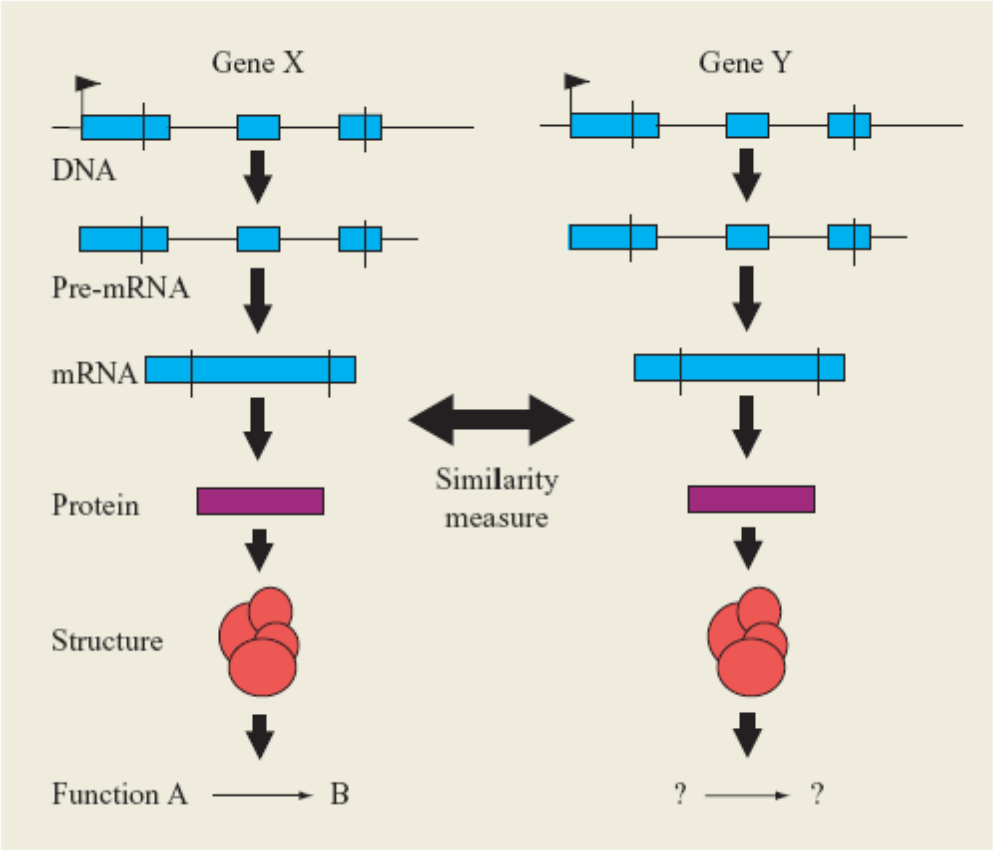
- Suppose we have a text written by Shakespeare and a monkey. Can we tell who wrote what?
- More important: DNA sequences with coding and non-coding sequences. Can we discriminate boundaries?

Example: dishonest casino



From Durbin

Goal



(E. Birney, 2001)

Cases

Example	**Observations**	**Hidden state**
Football	Plays	Coach
Text	Words	Shakespeare / monkey
Casino	Rolled numbers	Fair/loaded
DNA	ACGT	Coding/not