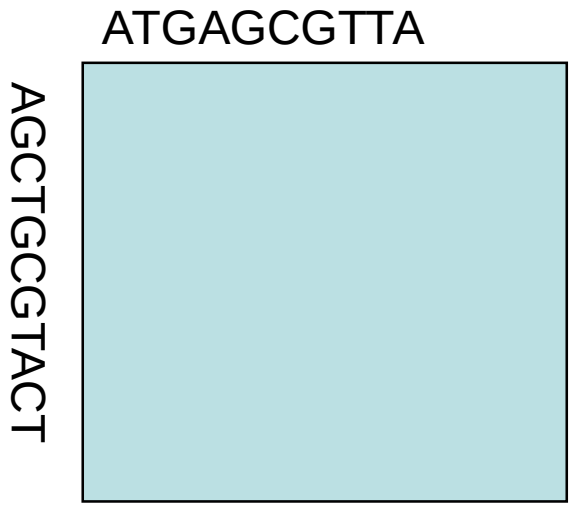
Note on sequences and alignment matrices in exercises

- Example solutions to alignment problems will have sequences arranged like this:
  - Perform global alignment of the sequences
    - s = AGCTGCGTACT
    - t = ATGAGCGTTA

So if you want to be able to compare your solution easily against the example, use this convention.



Tracing back optimal path

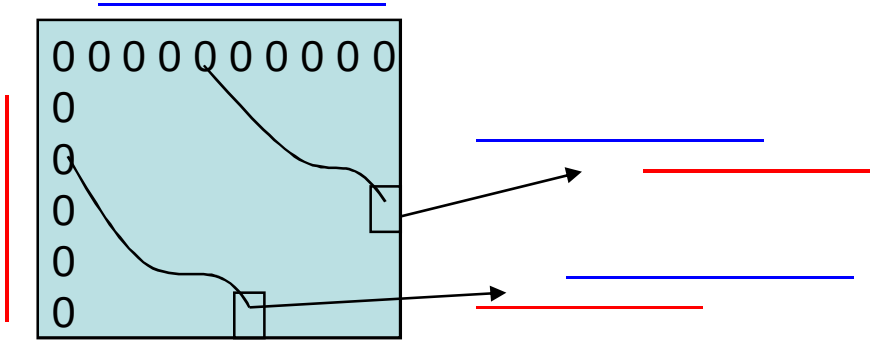
- When filling the DP matrix, one can store to each cell the list of pointers to the cells where the optimal value was computed; then tracing back all the optimal paths is easy.
- Explicit pointers are actually not needed, since one can just reverse the computation; start from cell (m,n), check which are the neighboring cells where the optimum value must have been computed, and continue like this to follow all optimal paths.

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3	-5	-7	-9
T	-4	-1	-2	-4	-4	-6
C	-6	-3	-2	-3	-5	-5
G	-8	-5	-2	-1	-3	-4
T	-10	-7	-4	-3	0	-2

Overlap alignment

- Overlap matrix used by Overlap-Layout-Consensus algorithm can be computed with dynamic programming
- Initialization: $O_{i,0} = O_{0,j} = 0$ for all i, j
- Recursion:

$$O_{i,j} = \max \{ O_{i-1,j-1} + s(a_i, b_i), O_{i-1,j} - \delta, O_{i,j-1} - \delta, \}$$



Best overlap: maximum value from rightmost column and bottom row

⇨ ...more on sequence assembly at the Spring 2010 seminar!

Non-uniform mismatch penalties

- We used uniform penalty for mismatches:

  s('A', 'C') = s('A', 'G') = … = s('G', 'T') = μ
- Transition mutations (A<->G, C<->T) are approximately twice as frequent than transversions (A<->C, A<->T, C<->G, G<->T)
  - use non-uniform mismatch penalties collected into a *substitution matrix*

	A	C	G	T
A	1	-1	-0.5	-1
C	-1	1	-1	-0.5
G	-0.5	-1	1	-1
T	-1	-0.5	-1	1

Gaps in alignment

- Gap is a succession of indels in alignment

```
C T – - - A A
C T C G C A A
```

- Previous model scored a length k gap as $w(k) = -k\delta$
- Replication processes may produce longer stretches of insertions or deletions
  - In coding regions, insertions or deletions of codons may preserve functionality

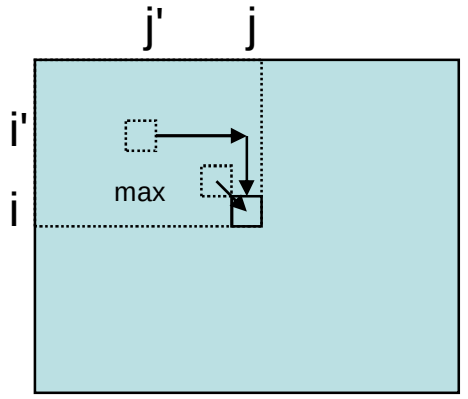
Gap open and extension penalties (2)

- We can design a score that allows the penalty opening gap to be larger than extending the gap:

$$w(k) = -\alpha - \beta(k - 1)$$

- Gap open cost α, Gap extension cost β
- Alignment algorithms can be extended to use $w(k)$ as follows:

$$S_{i,j} = \max_{i'+j'<i+j} \{ S_{i-1,j-1} + s(a_i, b_j), \; S_{i',j'} + w(j - j' + i - i') \}$$

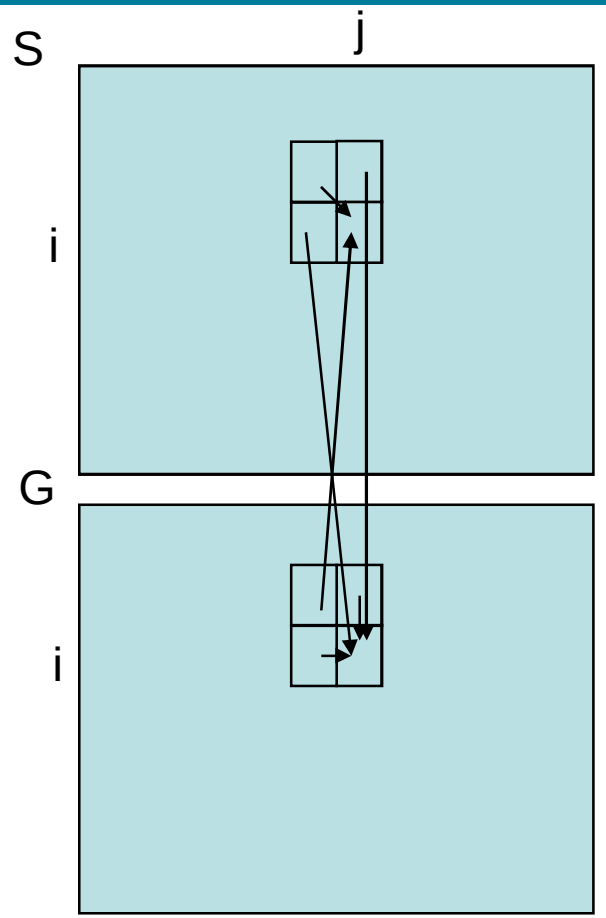


- However, this is too inefficient: $O(m^2n^2)$.

Speeding up gap open and extension alignment

$$S_{i,j} = \max \{ S_{i-1,j-1} + s(a_i, b_i), G_{i-1,j-1} + s(a_i, b_i) \}$$

$$G_{i,j} = \max \{ S_{i-1,j} - \alpha, G_{i-1,j} - \beta, S_{i,j-1} - \alpha, G_{i,j-1} - \beta \}$$



- Equivalent result in O(mn) time.

Demonstration of the EBI web site

- European Bioinformatics Institute (EBI) offers many biological databases and bioinformatics tools at http://www.ebi.ac.uk/
  - Sequence alignment: Tools -> Sequence Analysis -> Align

Sequence Alignment (chapter 6)

- The biological problem
- Global alignment
- Local alignment
- *Multiple alignment*

Multiple alignment

- Consider a set of d sequences on the right
  - Orthologous sequences from different organisms
  - Paralogs from multiple duplications
- How can we study relationships between these sequences?

aggcgagctgcgagtgcta
cgttagattgacgctgac
ttccggctgcgac
gacacggcgaacgga
agtgtgcccgacgagcgaggac
gcgggctgtgagcgcta
aagcggcctgtgtgcccta
atgctgctgccagtgta
agtcgagccccgagtgc
agtccgagtcc
actcggtgc

Optimal alignment of three sequences

- Alignment of $A = a_1a_2\ldots a_i$ and $B = b_1b_2\ldots b_j$ can end either in $(-, b_j)$, (a_i, b_j) or $(a_i, -)$
- $2^2 - 1 = 3$ alternatives
- Alignment of A, B and $C = c_1c_2\ldots c_k$ can end in $2^3 - 1$ ways: $(a_i, -, -)$, $(-, b_j, -)$, $(-, -, c_k)$, $(-, b_j, c_k)$, $(a_i, -, c_k)$, $(a_i, b_j, -)$ or (a_i, b_j, c_k)
- Solve the recursion using three-dimensional dynamic programming matrix: $O(n^3)$ time and space
- Generalizes to d sequences but impractical with even a moderate number of sequences

Multiple alignment in practice

- In practice, real-world multiple alignment problems are usually solved with heuristics
- Progressive multiple alignment
  - Choose two sequences and align them
  - Choose third sequence w.r.t. two previous sequences and align the third against them
  - Repeat until all sequences have been aligned
  - Different options how to choose sequences and score alignments
  - Note the similarity to Overlap-Layout-Consensus

Multiple alignment in practice

- Profile-based progressive multiple alignment: CLUSTALW
  - Construct a distance matrix of all pairs of sequences using dynamic programming
  - Progressively align pairs in order of decreasing similarity
  - CLUSTALW uses various heuristics to contribute to accuracy

Additional material

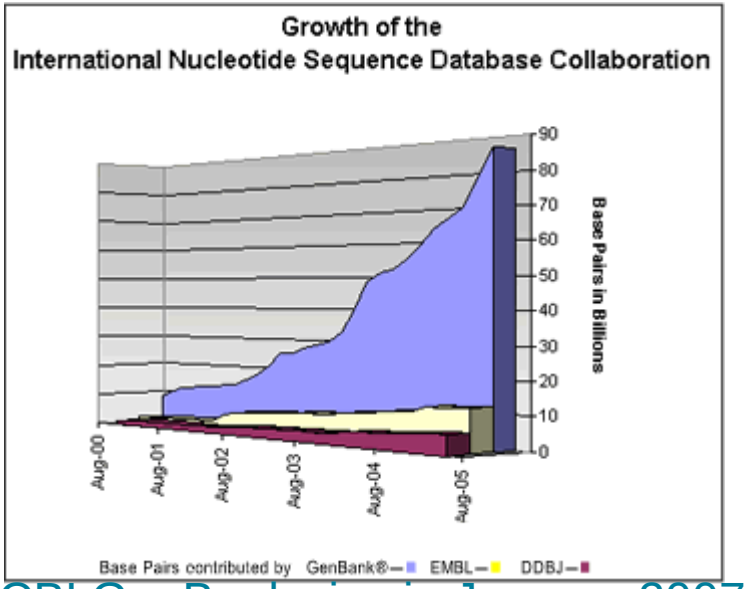
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison: Biological sequence analysis
- N. C. Jones, P. A. Pevzner: An introduction to bioinformatics algorithms
- Biological sequence analysis course, next time in 2010-2011?

Rapid alignment methods: FASTA and BLAST (Section 7)

- *The biological problem*
- *Search strategies*
- FASTA
- BLAST

The biological problem

- Global and local alignment algoritms are slow in practice
- Consider the scenario of aligning a *query sequence* against a large database of sequences
  - New sequence with unknown function



- NCBI GenBank size in January 2007 was 65 369 091 950 bases (61 132 599 sequences)
- Feb 2008: 85 759 586 764 bases (82 853 685 sequences)

Problem with large amount of sequences

- Exponential growth in both number and total length of sequences
- Possible solution: Compare against model organisms only
- With large amount of sequences, chances are that matches occur by random
  - Need for statistical analysis

Rapid alignment methods: FASTA and BLAST

- The biological problem
- Search strategies
- *FASTA*
- BLAST

FASTA

- FASTA is a multistep algorithm for sequence alignment (Wilbur and Lipman, 1983)
- The sequence file format used by the FASTA software is widely used by other sequence analysis software
- Main idea:
  - Choose regions of the two sequences (query and database) that look promising (have some degree of similarity)
  - Compute local alignment using dynamic programming in these regions

Search strategies

- How to speed up the computation?
  - Find ways to limit the number of pairwise comparisons
- Compare the sequences at *k-mer* level:
  - k-mer (or a *k-word*, or a *k-tuple*) is a substring of length k

FASTA outline

- FASTA algorithm has five steps:
  - *1. Identify common k-mers between I and J*
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Analyzing the k-mer content

- Example query string I: TGATGATGAAGACATCAG
- For k = 8, the set of k-mers of I is

```
TGATGATG
 GATGATGA
  ATGATGAA
   TGATGAAG
      ...
          GACATCAG
```

Analyzing the k-mer content

- There are n-k+1 k-mers in a string of length n
- If at least one k-mer of I is not found from another string J, we know that I differs from J
- Need to consider statistical significance: I and J might share k-mers by chance only
- Let m=|I| and n=|J|

Word lists and comparison by content

- The k-mers of I can be arranged into a table of k-mer occurences $L_w(I)$
- Consider the k-mers when k=2 and I=GCATCGGC:

GC, CA, AT, TC, CG, GG, GC

AT: 3
CA: 2
CG: 5
GC: 1, 7 ← Start indecies of k-mer GC in I
GG: 6
TC: 4

Building $L_w(I)$ takes O(n) time

Common k-mers

- Number of common k-mers in I and J can be computed using $L_w(I)$ and $L_w(J)$
- For each k-mer w in I, there are $|L_w(J)|$ occurences in J
- Therefore I and J have $\sum_w |L_w(I)||L_w(J)|$ common k-mer pairs
- This can be computed in $O(m + n + 4^k)$ time
  - $O(m + n + 4^k)$ time to build the lists
  - $O(4^k)$ time to multiply the corresponding list entry sizes (in DNA strings)

Common k-mers

- I = GCATCGGC
- J = CCATCGCCATCG

$L_w(I)$	$L_w(J)$	Common k-mers
AT: 3	AT: 3, 9	2
CA: 2	CA: 2, 8	2
	CC: 1, 7	0
CG: 5	CG: 5, 11	2
GC: 1, 7	GC: 6	2
GG: 6		0
TC: 4	TC: 4, 10	2
		10 in total

Properties of the common word list

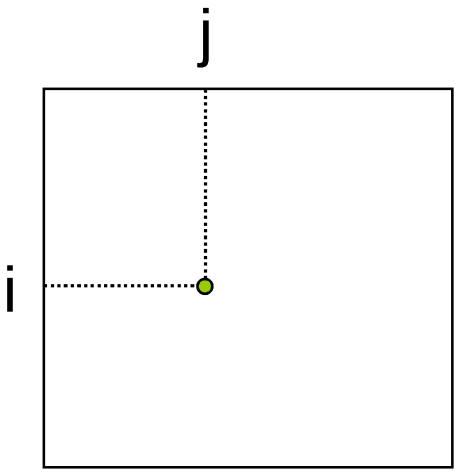
- For large k, the table size $O(4^k)$ is too large to compute the common k-mer pairs count in the previous fashion
- Instead, an approach based on merge sort can be utilised (details skipped)
- The common k-mer technique can be combined with the local alignment algorithm to yield a rapid alignment approach
- Alternative solutions, not part of FASTA:
  - Exercise: Show how generalized suffix tree can be used for counting the common k-mer pairs
  - Generalized suffix tree with some additional data structures can also be used for directly computing all maximal pairs of tuples {(i',i),(j',j)} such that $a_{i'}...a_i = b_{j'}...b_j$ and the ranges cannot be extended left or right (see Gusfield's book).

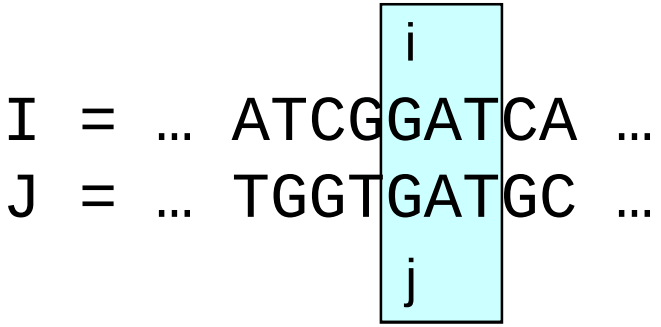
FASTA outline

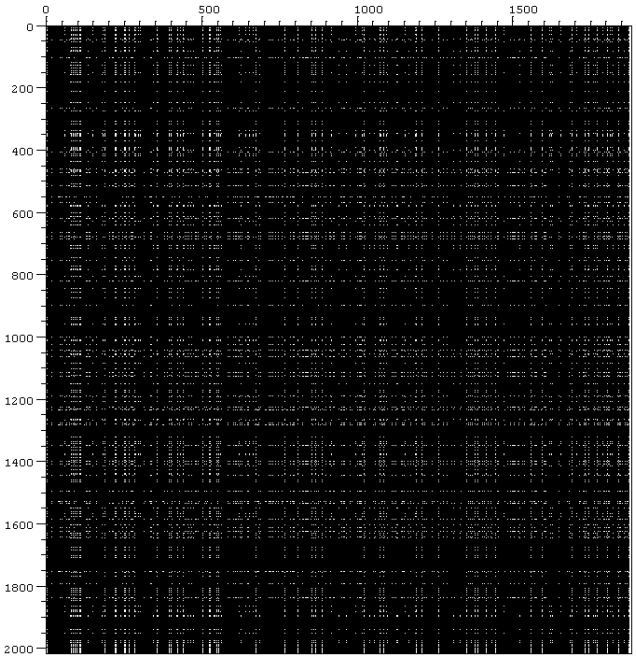
- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - *2. Score diagonals with k-mer matches, identify 10 best diagonals*
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - 5. Perform dynamic programming to find final alignments

Dot matrix comparisons

- k-mer matches in two sequences I and J can be represented as a *dot matrix*
- Dot matrix element (i, j) has ”a dot”, if the k-mer starting at position i in I is identical to the k-mer starting at position j in J
- The dot matrix can be plotted for various k

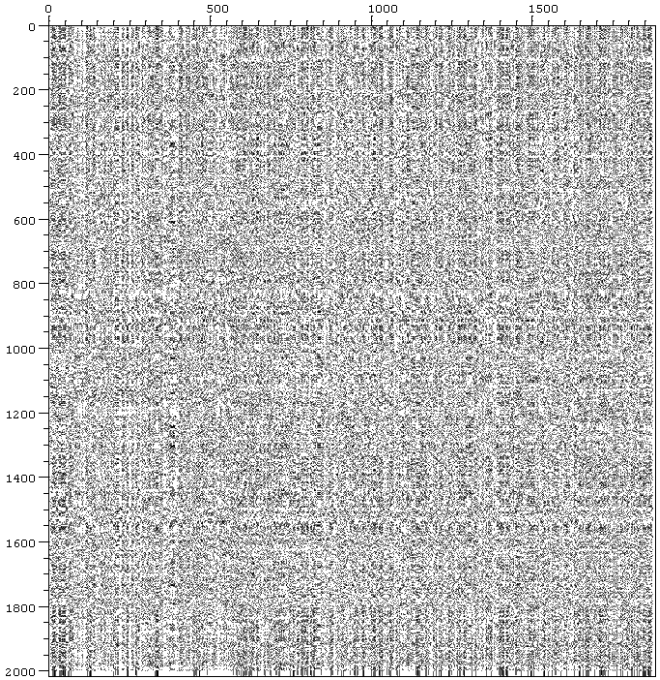




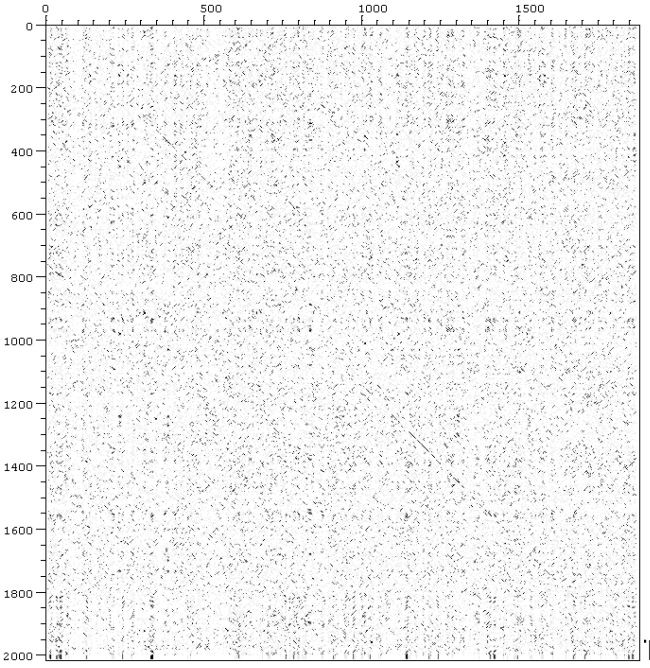


k=1

k=4

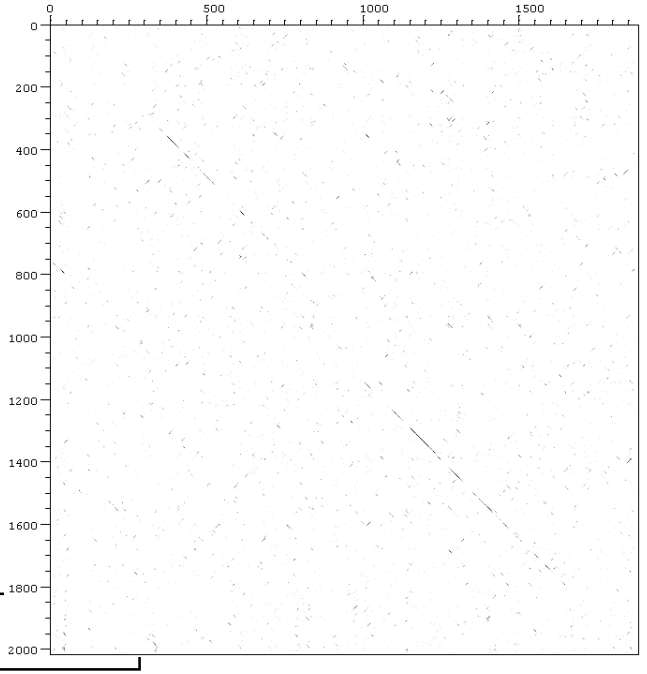


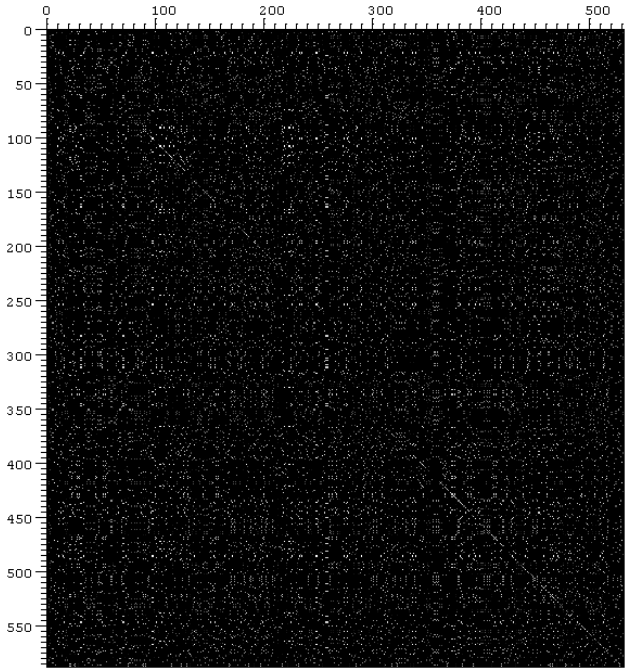
Dot matrix (k=1,4,8,16)
for two **DNA** sequences
X85973.1 (1875 bp)
Y11931.1 (2013 bp)



k=8

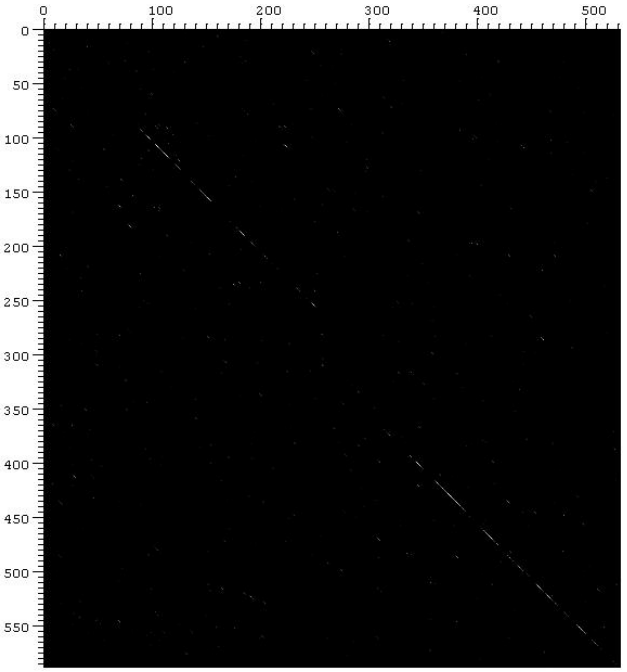
k=16



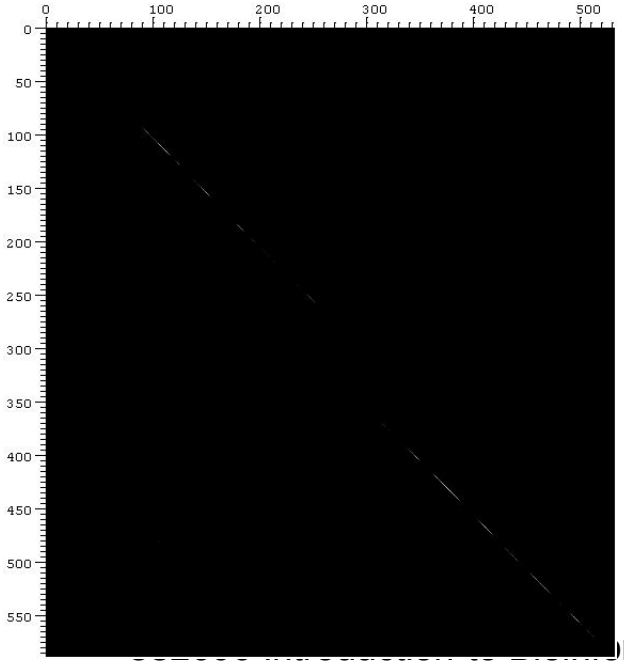


k=1

k=4

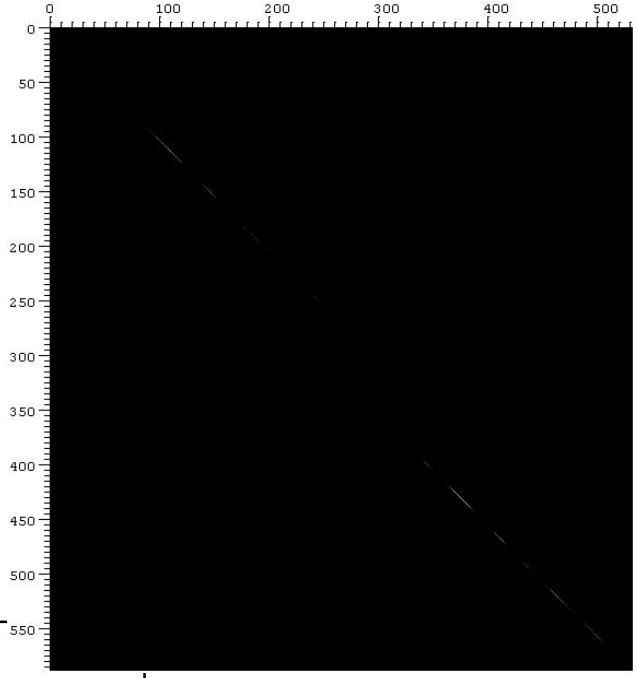


Dot matrix
(k=1,4,8,16) for two
protein sequences
CAB51201.1 (531 aa)
CAA72681.1 (588 aa)



k=8

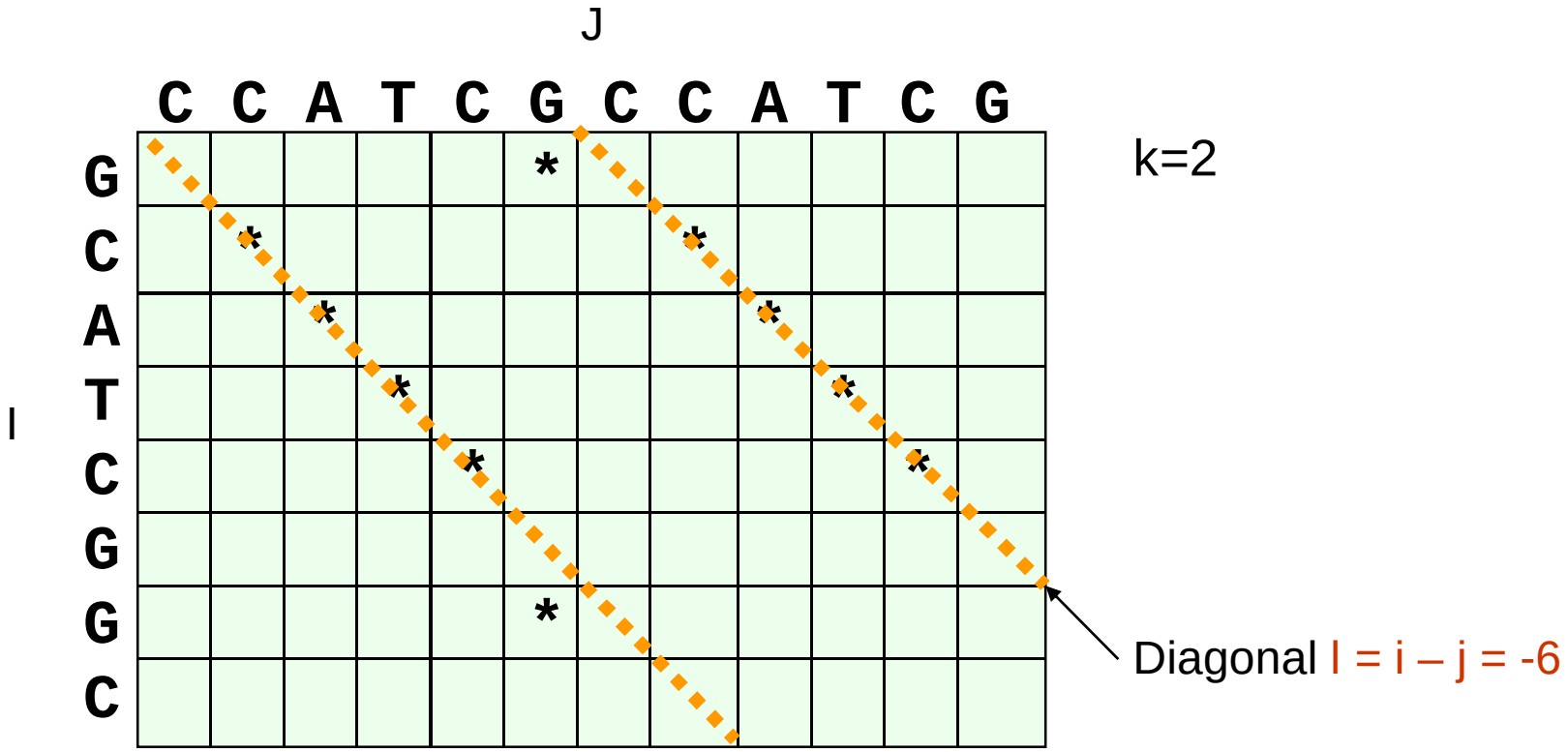
k=16



Shading indicates
now the match score
according to a
score matrix
(Blosum62 here)

Computing diagonal sums

- We would like to find high scoring diagonals of the dot matrix
- Lets index diagonals by the offset, l = i - j



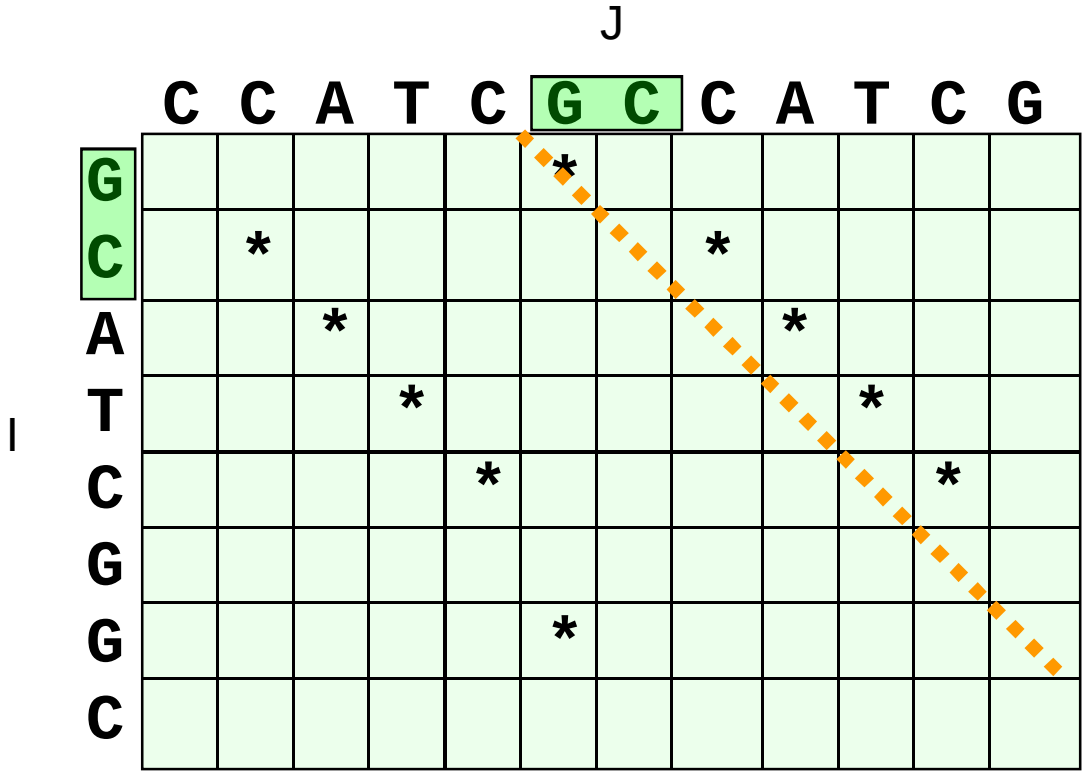
Computing diagonal sums

- As an example, lets compute diagonal sums for I = GCATCGGC, J = CCATCGCCATCG, k = 2
- 1. Construct k-mer list $L_w(J)$
- 2. Diagonal sums S_l are computed into a table, indexed with the offset and initialised to zero

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums

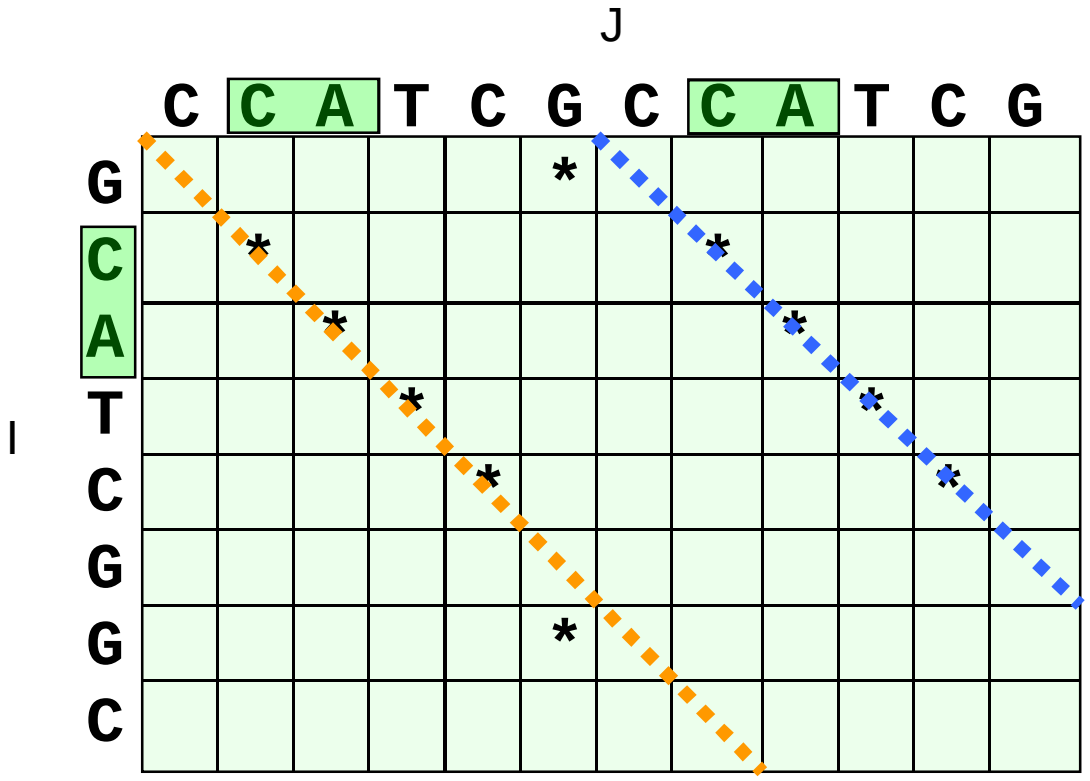


For the first 2-mer in I, GC, $L_{GC}(J) = \{6\}$.

We can then update the sum of diagonal $l = i - j = 1 - 6 = -5$ to $S_{-5} := S_{-5} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-mer in I is CA, for which $L_{CA}(J) = \{2, 8\}$.

Two diagonal sums are updated:

$I = i - j = 2 - 2 = 0$

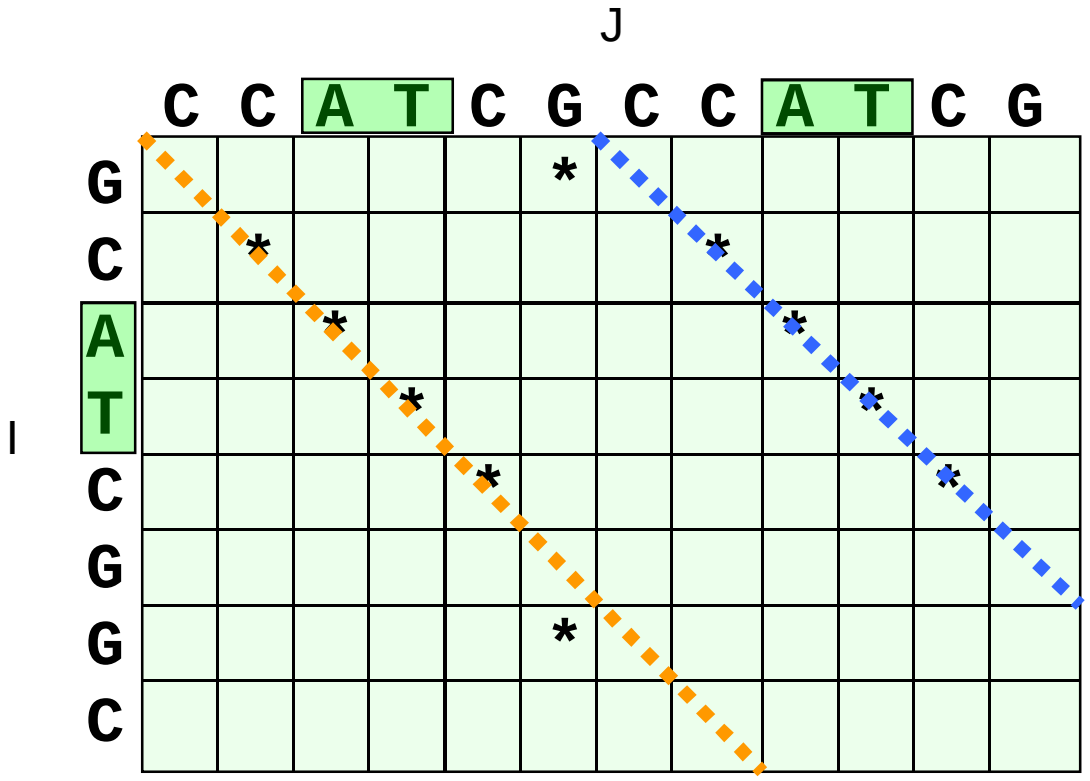
$S_0 := S_0 + 1 = 0 + 1 = 1$

$I = i - j = 2 - 8 = -6$

$S_{-6} := S_{-6} + 1 = 0 + 1 = 1$

Computing diagonal sums

- 3. Go through k-mers of I, look for matches in $L_w(J)$ and update diagonal sums



Next 2-mer in I is AT, for which $L_{AT}(J) = \{3, 9\}$.

Two diagonal sums are updated:

$I = i - j = 3 - 3 = 0$

$S_0 := S_0 + 1 = 1 + 1 = 2$

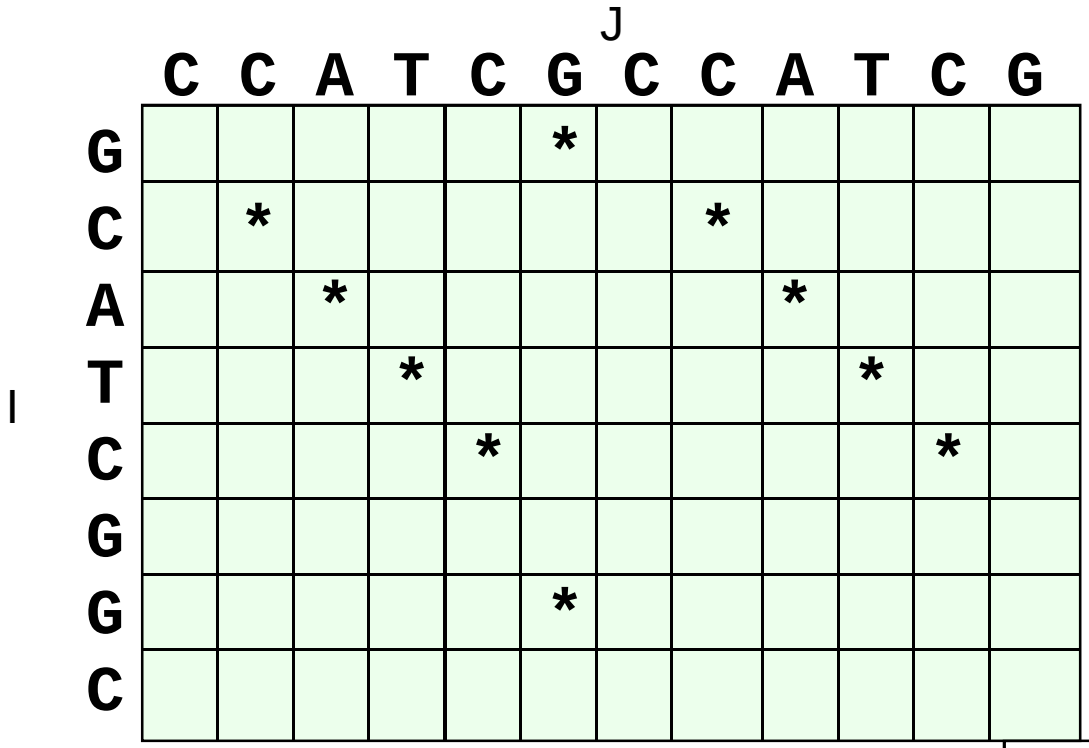
$I = i - j = 3 - 9 = -6$

$S_{-6} := S_{-6} + 1 = 1 + 1 = 2$

Computing diagonal sums

After going through the k-mers of I, the result is:

l	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
S_l	0	0	0	0	4	1	0	0	0	0	4	1	0	0	0	0	0



Algorithm for computing diagonal sum of scores

$S_l := 0$ for all $1 - n < l \leq m - 1$

Compute $L_w(J)$ for all k-mers w

for i := 1 to m – k +1 do

    w := $l_i l_{i+1} \ldots l_{i+k-1}$

    for j $\in$ $L_w(J)$ do

        l := i – j

        $S_l := S_l + 1$ ← Match score is here 1

    end

end

FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - *3. Rescore initial regions with a substitution score matrix*
  - *4. Join initial regions using gaps, penalise for gaps*
  - 5. Perform dynamic programming to find final alignments

Rescoring initial regions

- Each high-scoring diagonal chosen in the previous step is rescored according to a score matrix
- This is done to find subregions with identities shorter than k
- Non-matching ends of the diagonal are trimmed

```
I: C C A T C G C C A T C G
J: C C A A C G C A A T C A
```

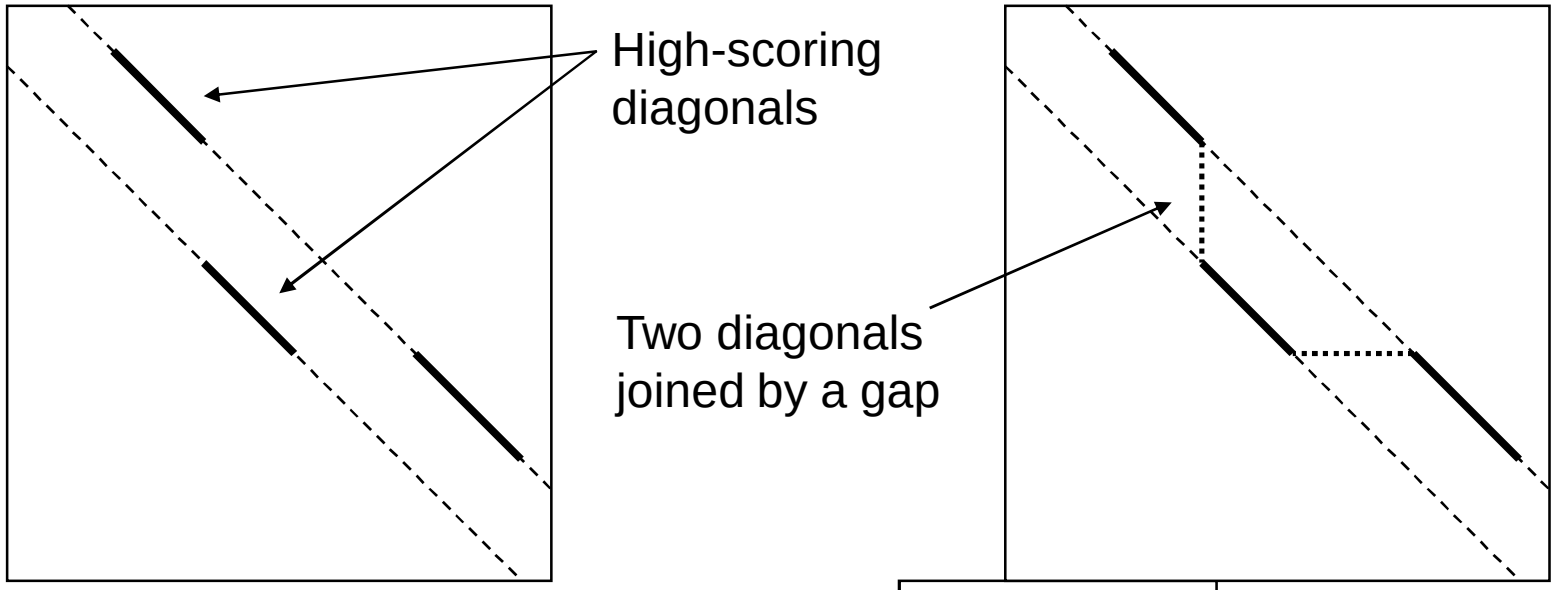
75% identity, no 4-mer identities

```
I': C C A T C G C C A T C G
J': A C A T C A A A T A A A
```

33% identity, one 4-mer identity

Joining diagonals

- Two offset diagonals can be joined with a gap, if the resulting alignment has a higher score
- Separate gap open and extension are used
- Find the best-scoring combination of diagonals

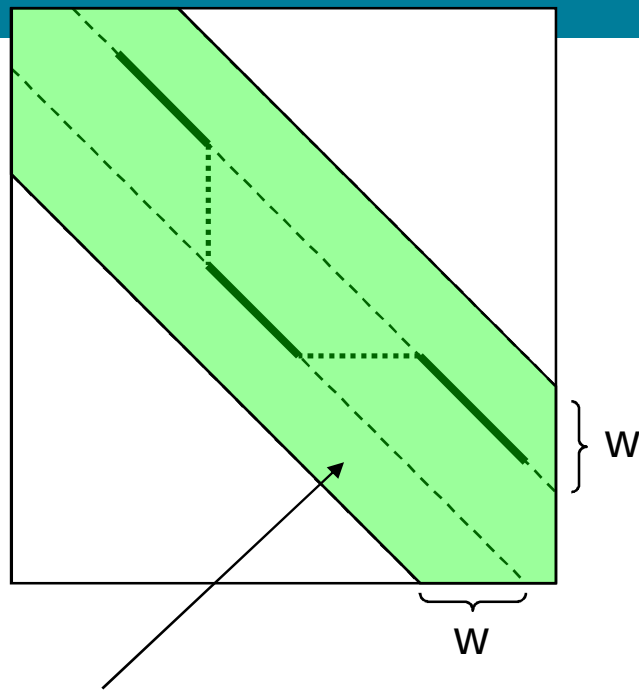


FASTA outline

- FASTA algorithm has five steps:
  - 1. Identify common k-mers between I and J
  - 2. Score diagonals with k-mer matches, identify 10 best diagonals
  - 3. Rescore initial regions with a substitution score matrix
  - 4. Join initial regions using gaps, penalise for gaps
  - *5. Perform dynamic programming to find final alignments*

Local alignment in the highest-scoring region

- Last step of FASTA: perform local alignment using dynamic programming around the highest-scoring diagonals
- Region to be aligned covers –w and +w offset diagonal to the highest-scoring diagonals
- With long sequences, this region is typically very small compared to the whole m x n matrix



Dynamic programming matrix
M filled only for the green region

Properties of FASTA

- Fast compared to local alignment using dynamic programming only
  - Only a narrow region of the full matrix is aligned
- *Lossy filter* : may fail to find some high scoring local alignments
- Increasing parameter k decreases the number of hits:
  - Increases specificity
  - Decreases sensitivity
  - Decreases running time

Properties of FASTA

- FASTA looks for initial exact matches to query sequence
  - Two proteins can have very different amino acid sequences and still be biologically similar
  - This may lead into a lack of sensitivity with diverged sequences

Demonstration of FASTA at EBI

- http://www.ebi.ac.uk/fasta/
- Note that parameter ktup in the software corresponds to parameter k in lectures

Rapid alignment methods: FASTA and BLAST

- The biological problem
- Search strategies
- FASTA
- *BLAST*

BLAST: Basic Local Alignment Search Tool

- BLAST (Altschul et al., 1990) and its variants are some of the most common sequence search tools in use
- Roughly, the basic BLAST has three parts:
  - 1. Find *segment pairs* between the query sequence and a database sequence above score threshold (”seed hits”)
  - 2. Extend seed hits into *locally maximal segment pairs*
  - 3. Calculate p-values and a rank ordering of the local alignments
- Gapped BLAST introduced in 1997 allows for gaps in alignments

Finding seed hits

- First, we generate a set of *neighborhood sequences* for given k, *match score matrix and threshold* T
- Neighborhood sequences of a k-mer w include all strings of length k that, when aligned against w, have the alignment score at least T
- For instance, let I = GCATCGGC, J = CCATCGCCATCG and k = 5, match score be 1, mismatch score be 0 and T = 4

Finding seed hits

- I = GCATCGGC, J = CCATCGCCATCG, k = 5, match score 1, mismatch score 0, T = 4
- This allows for one mismatch in each k-mer
- The neighborhood of the first k-mer of I, GCATC, is GCATC and the 15 sequences

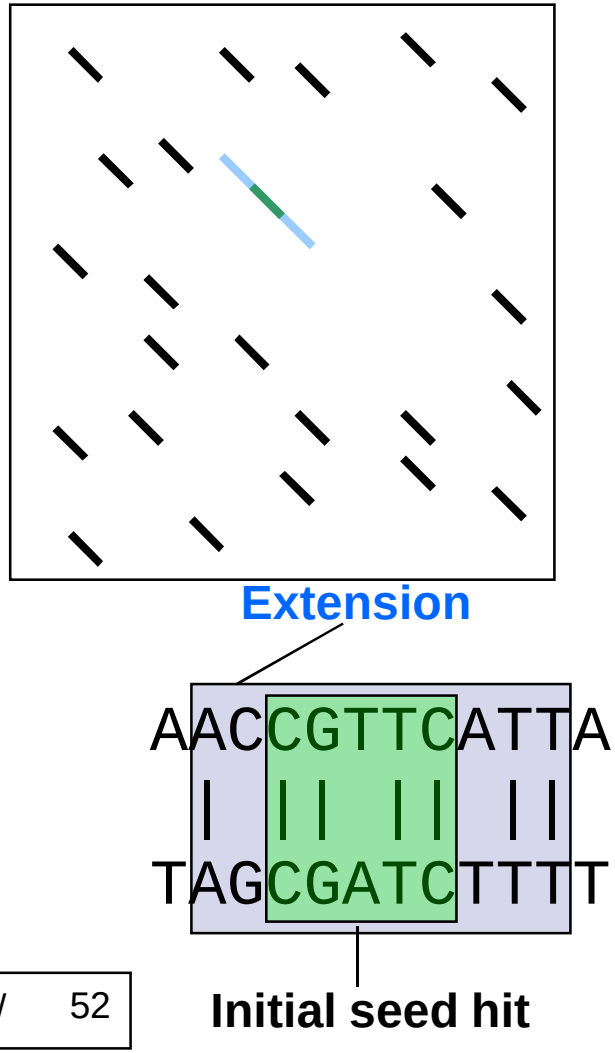
$$\left\{\begin{matrix} A \\ CCATC, G \\ T \end{matrix}\right\}\left\{\begin{matrix} A \\ GATC, GC \\ T \end{matrix}\right\}\left\{\begin{matrix} C \\ GTC, GCA \\ T \end{matrix}\right\}\left\{\begin{matrix} A \\ CC, GCAT \\ G \end{matrix}\right\}\left\{\begin{matrix} A \\ G \\ T \end{matrix}\right\}$$

Finding seed hits

- I = GCATCGGC has 4 k-mers and thus 4x16 = 64 5-mer patterns to locate in J
  - Occurrences of patterns in J are called **seed hits**
- Patterns can be found using exact search in time proportional to the sum of pattern lengths + length of J + number of matches (Aho-Corasick algorithm)
  - Attend 58093 String processing algorithms to learn Aho-Corasick and alike algorithms.
- Compare this approach to FASTA

Extending seed hits: original BLAST

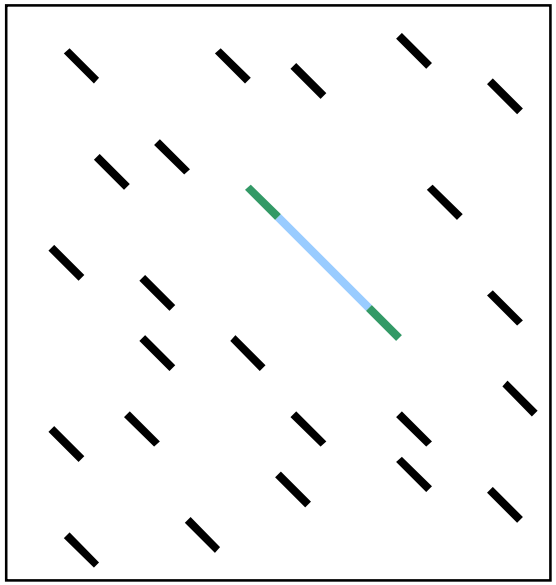
- Initial seed hits are extended into **locally maximal segment pairs** or **High-scoring Segment Pairs** (HSP)
- Extensions do not add gaps to the alignment
- Sequence is extended until the alignment score drops below the maximum attained score minus a threshold parameter value
- All statistically significant HSPs reported



Altschul, S.F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J., *J. Mol. Biol.*, 215, 403-410, 1990

Extending seed hits: gapped BLAST

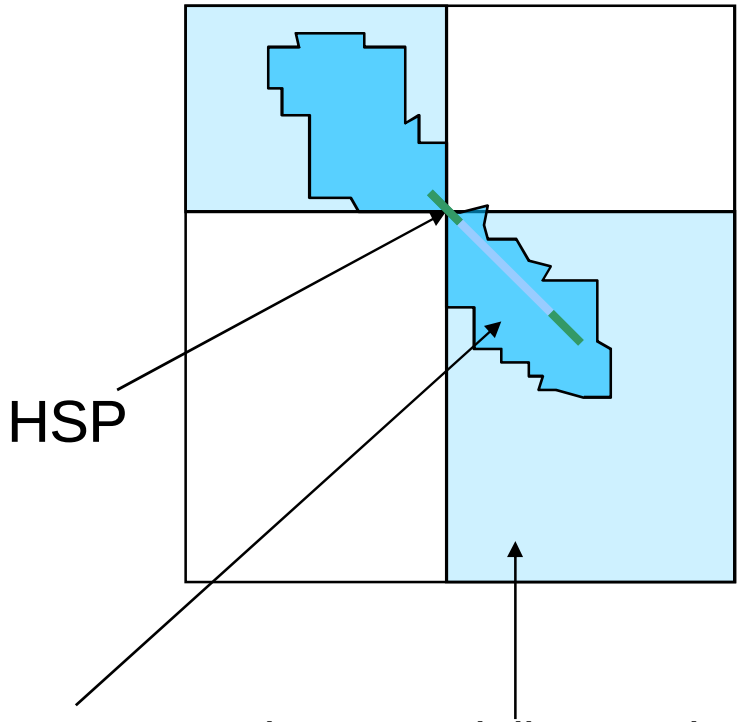
- In a later version of BLAST, two seed hits have to be found on the same diagonal
  - Hits have to be non-overlapping
  - If the hits are closer than A (additional parameter), then they are joined into a HSP
- Threshold value T is lowered to achieve comparable sensitivity
- If the resulting HSP achieves a score at least S_g, a *gapped extension* is triggered



Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, and Lipman DJ, *Nucleic Acids Res*. 1;25(17), 3389-402, 1997

Gapped extensions of HSPs

- Local alignment is performed starting from the HSP
- Dynamic programming matrix filled in ”forward” and ”backward” directions (see figure)
- Skip cells where value would be X_g below the best alignment score found so far



Estimating the significance of results

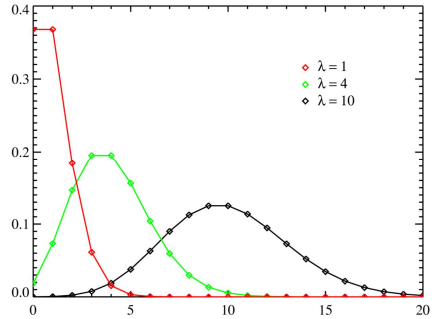
- In general, we have a score $S(D, X) = s$ for a sequence X found in database D
- BLAST rank-orders the sequences found by p-values
- The p-value for this hit is $P(S(D, Y) \geq s)$ where Y is a random sequence
  - Measures the amount of ”surprise” of finding sequence X
- A smaller p-value indicates more significant hit
  - A p-value of 0.1 means that one-tenth of random sequences would have as large score as our result

Estimating the significance of results

- In BLAST, p-values are computed roughly as follows
- There are mn places to begin an optimal alignment in the m x n alignment matrix
- Optimal alignment is preceded by a mismatch and has t matching (identical) letters
  - (Assume match score 1 and mismatch/indel score -∞)
- Let p = P(two random letters are equal)
- The probability of having a mismatch and then t matches is $(1-p)p^t$

Estimating the significance of results

- We model this event by a Poisson distribution (why?) with mean $\lambda = nm(1-p)p^t$
- P(there is local alignment t or longer)
  $\approx 1 - P(\text{no such event})$
  $= 1 - e^{-\lambda} = 1 - \exp(-nm(1-p)p^t)$
- An equation of the same form is used in Blast:
- E-value = $P(S(D, Y) \geq s) \approx 1 - \exp(-mn\gamma\xi^t)$ where $\gamma > 0$ and $0 < \xi < 1$
- Parameters γ and ξ are estimated from data



Properties of BLAST

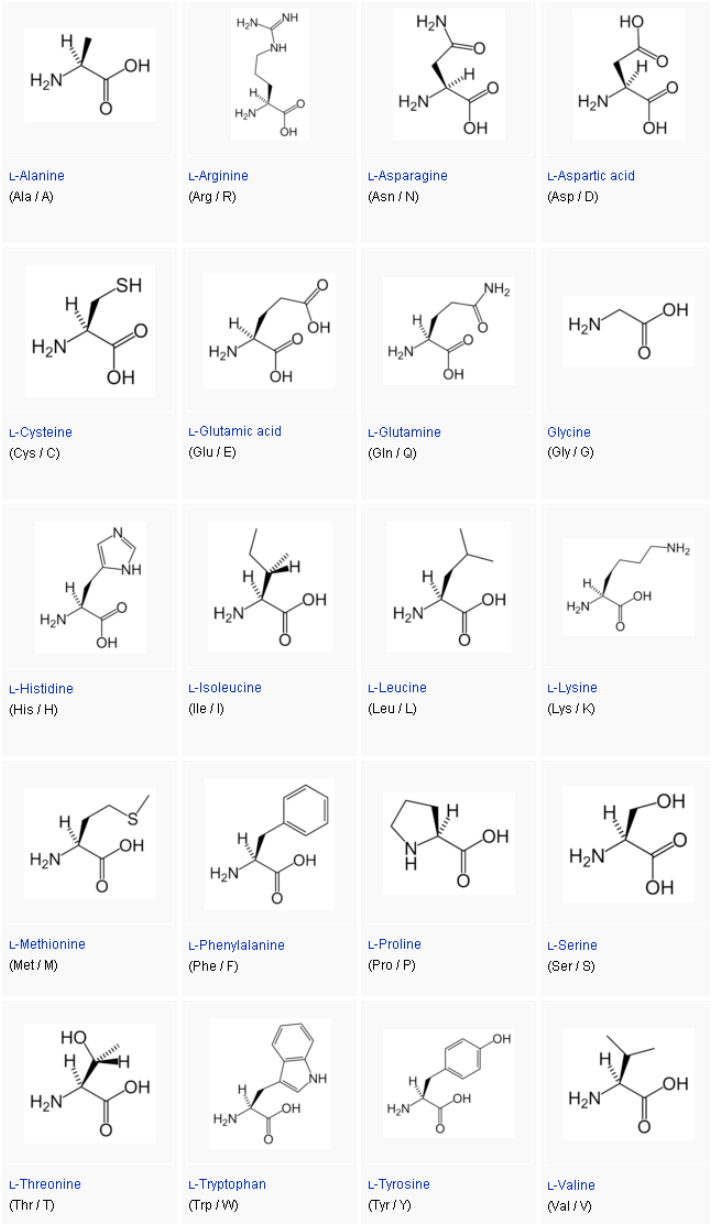
- Better sensitivity than in FASTA
- Still a lossy filter:
  - Alternative lossless filters exist with similar performance!
    - See e.g. Lam et at.: Compressed indexing and local alignment of DNA, *Bioinformatics*, 25:1754-1760, 2008.
- Has become *the standard* in Bioinformatics:
  - This is due to the p-value computation and ranking of results, which give a way to compare the significance of the search results.
    - However, these computations apply to any alignment algorithm not just to BLAST.
    - BLAST may fail to find real occurrences, skewing the comparison of significance.

Amino acid sequences

- We have discussed mainly DNA sequences
- Amino acid sequences can be aligned as well
- However, the design of the substitution matrix is more involved because of the larger alphabet
- More on the topic in the course Biological sequence analysis

Scoring amino acid alignments

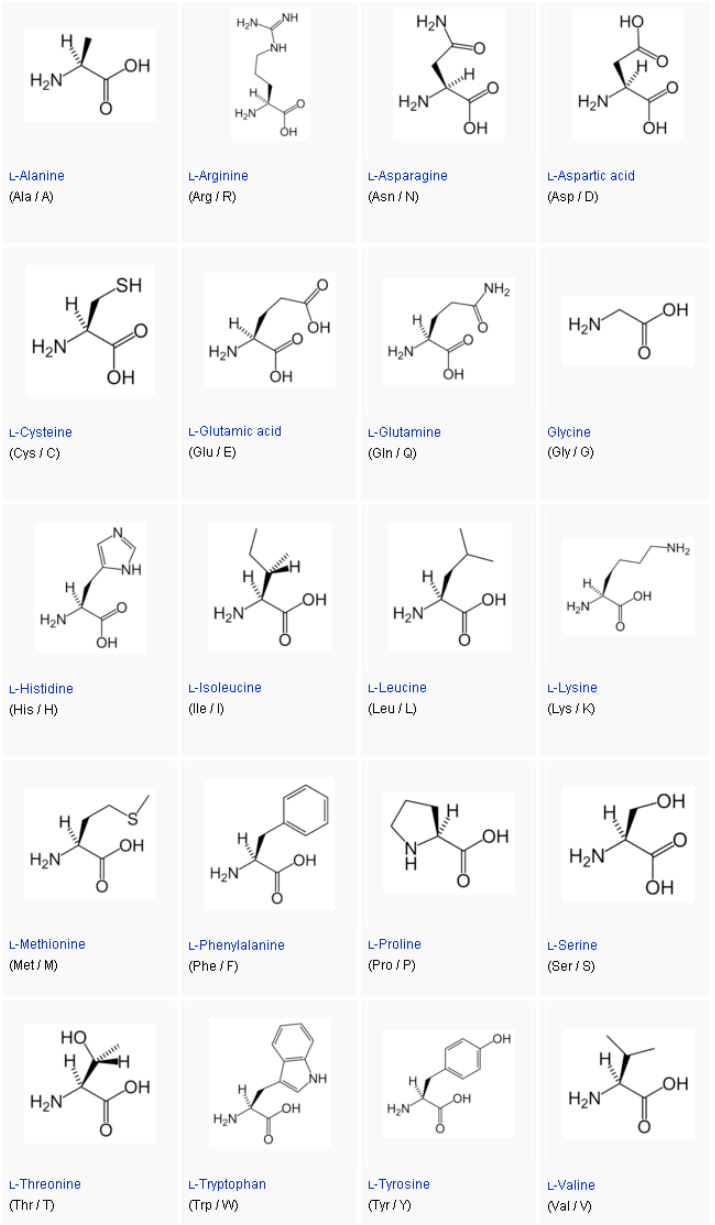
- We need a way to compute the score S(D, X) for aligning the sequence X against database D
- Scoring DNA alignments was discussed previously
- Constructing a scoring model for amino acids is more challenging
  - 20 different amino acids vs. 4 bases
- Figure shows the molecular structures of the 20 amino acids



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Scoring amino acid alignments

- Substitutions between chemically similar amino acids are more frequent than between dissimilar amino acids
- We can check our scoring model against this



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Score matrices

- Scores $s = S(D, X)$ are obtained from score matrices
- Let $A = a_1a_2\ldots a_n$ and $B = b_1b_2\ldots b_n$ be sequences of equal length (no gaps allowed to simplify things)
- To obtain a score for alignment of A and B, where a_i is aligned against b_i, we take the ratio of two probabilities
  - The probability of having A and B where the characters match (*match model M*)
  - The probability that A and B were chosen randomly (*random model R*)

Score matrices: random model

- Under the random model, the probability of having A and B is

$$P(A, B|R) = \prod_i q_{ai} \prod_i q_{bi}$$

where q_{xi} is the probability of occurence of amino acid type x_i

- Position where an amino acid occurs does not affect its type

Score matrices: match model

- Let p_{ab} be the probability of having amino acids of type a and b aligned against each other given they have evolved from the same ancestor c
- The probability is

$$P(A, B|M) = \prod_i p_{a_i b_i}$$

Score matrices: log-odds ratio score

- We obtain the score S by taking the ratio of these two probabilities

$$\frac{P(A,B|M)}{P(A,B|R)} = \frac{\prod_i p_{a_i b_i}}{\prod_i q_{a_i} \prod_i q_{b_i}} = \prod_i \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}}$$

and taking a logarithm of the ratio

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

Score matrices: log-odds ratio score

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

- The score S is obtained by summing over character pair-specific scores:

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

- The probabilities q_a and p_{ab} are extracted from data

Calculating score matrices for amino acids

- Probabilities q_a are in principle easy to obtain:
  - Count relative frequencies of every amino acid in a sequence database

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Calculating score matrices for amino acids

- To calculate p_{ab} we can use a known pool of aligned sequences
- BLOCKS is a database of highly conserved regions for proteins
- It lists multiply aligned, ungapped and conserved protein segments
- Example from BLOCKS shows genes related to human gene associated with DNA-repair defect xeroderma pigmentosum

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

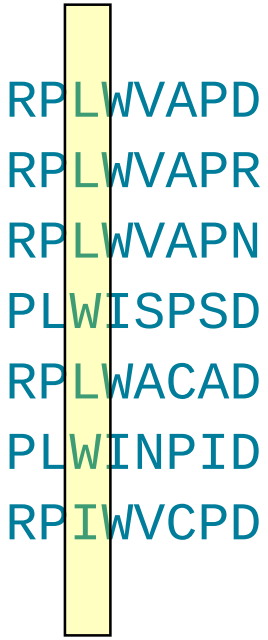
Block PR00851A
ID XRODRMPGMNTB; BLOCK
AC PR00851A; distance from previous block=(52,131)
DE Xeroderma pigmentosum group B protein signature
BL adapted; width=21; seqs=8; 99.5%=985; strength=1287

```
XPB_HUMAN|P19447 ( 74)    RPLWVAPDGHIFLEAFSPVYK 54
XPB_MOUSE|P49135 ( 74)    RPLWVAPDGHIFLEAFSPVYK 54
P91579 ( 80)              RPLYLAPDGHIFLESFSPVYK 67
XPB_DROME|Q02870 ( 84)    RPLWVAPNGHVFLESFSPVYK 79
RA25_YEAST|Q00578 ( 131) PLWISPSDGRIILESFSPLAE 100
Q38861 ( 52)              RPLWACADGRIFLETFSPLYK 71
O13768 ( 90)              PLWINPIDGRIILEAFSPLAE 100
O00835 ( 79)              RPIWVCPDGHIFLETFSAIYK 86
```

http://blocks.fhcrc.org

BLOSUM matrix

- BLOSUM is a score matrix for amino acid sequences derived from BLOCKS data
- First, count pairwise matches $f_{x,y}$ for every *amino acid type pair (x, y)*
- For example, for column 3 and amino acids L and W, we find 8 pairwise matches: $f_{L,W} = f_{W,L} = 8$



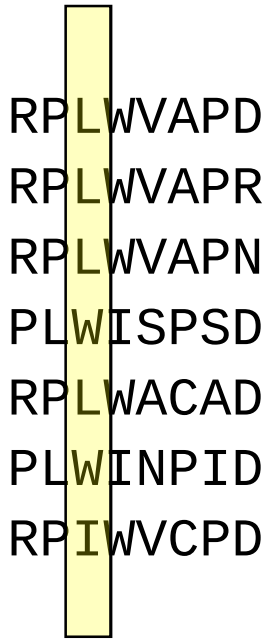
Creating a BLOSUM matrix

- Probability p_{ab} is obtained by dividing f_{ab} with the total number of pairs (note difference with course book):

$$p_{ab} = f_{ab} / \sum_{x=1}^{20} \sum_{y=1}^{x} f_{xy}$$

- We get probabilities q_a by

$$q_a = \sum_{b=1}^{20} p_{ab}$$



Creating a BLOSUM matrix

- The probabilities p_{ab} and q_a can now be plugged into

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

to get a 20 x 20 matrix of scores s(a, b).

- Next slide presents the BLOSUM62 matrix
  - Values scaled by factor of 2 and rounded to integers
  - Additional step required to take into account expected evolutionary distance
  - Described in Deonier's book in more detail

BLOSUM62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Using BLOSUM62 matrix

MQLEANADTSV

| | |

LQEQAEAQGEM

$$s = \sum_{i=1}^{11} s(a_i, b_i)$$

= 2 + 5 – 3 – 4 + 4 + 0 + 4 + 0 – 2 + 0 + 1

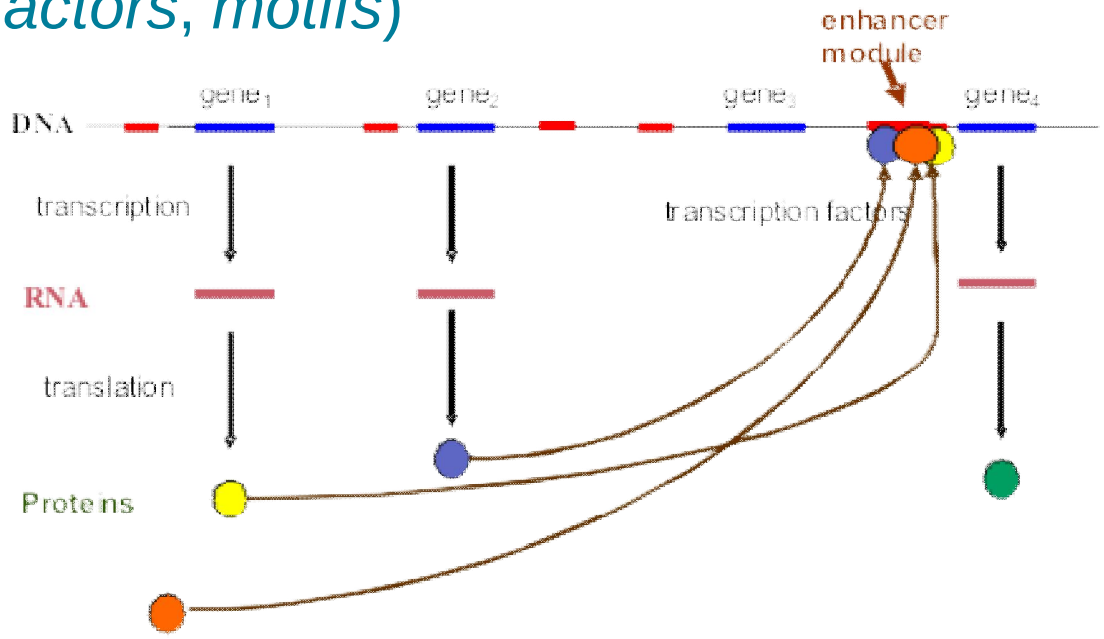
= 7

Demonstration of BLAST at NCBI

- http://www.ncbi.nlm.nih.gov/BLAST/

Signals in DNA (Section 9)

- Genes
- Promoter regions
- Binding sites for regulatory proteins (*transcription factors*, *motifs*)



Gene finding

- Could be done by dynamic programming similar to exercise 3/6 taking into account
  - Sequencing errors
  - Exons / intron gap constraints
  - Total gene length constraints
  - Codon usage optimization
  - Start codons / stop codons.
- In practise, often just the consensus reverse translation is taken and BLAST is used for finding local alignments (exons).

Promoter sequences

- Often immediately before the gene.
- Clear structure in prokaryotes, more complex in eukaryotes.
- An example from *E coli* is shown in next slide (taken from course book).

Promoter example

Table 9.2. A sample of *E. coli* promoter sequences. These sequences have been aligned relative to the transcriptional start site at position +1 (boldface large letter). Sequences from −40 to +11 are shown. Close matches to consensus −35 and −10 hexamers are underlined. See also Appendix C.3 for additional examples and sources of the data.

	−35	−10	−1
ORF83P1	CTCTGCTGGCATTCACAATGCGCAGGGGTAAAACGTTTCCTGTAGCACCG		
ada	GTTGGTTTTTGCGTGATGGTGACCGGGCAGCCTAAAGGCTATCCTTAACCA		
*amn*P4	TTCACATTTCTGTGACATACTATCGGATGTGCGGTAATTGTATGGAACAGG		
araFGH	CTCTCCTATGGAGAATTAATTTCTCGCTAAAACTATGTCAACACAGTCACT		
aroG	CCCCGTTTACACATTCTGACGGAAGATATAGATTGGAAGTATTGCATTCAC		
atpI	TATTGTTTGAAATCACGGGGGCGCACCGTATAATTTGACCGCTTTTTGATG		
caiT	AATCACAGAATACAGCTTATTGAATACCCATTATGAGTTAGCCATTAACGC		
*clpA*P1	TTATTGACGTGTTACAAAAATTCTTTTCTTATGATGTAGAACGTGCAACGC		
*crr*P2-I	GTGGTGAGCTTGCTGGCGATGAACGTGCTACACTTCTGTTGCTGGGGATGG		

Representing signals in DNA

- Consensus sequence:
  - -10 site in E coli: TATAAT
  - GRE half-site consensus: AGAACA
- Simple regular expression:
  - A(C/G)AA(C/G)(A/T)
- Positional weight matrix (PWM):

A	1.00	0.00	1.00	1.00	0.00	0.86
C	0.00	0.14	0.00	0.00	0.86	0.00
G	0.00	0.86	0.00	0.00	0.14	0.00
T	0.00	0.00	0.00	0.00	0.00	0.14

GRE half-sites:

AGAACA
ACAACA
AGAACA
AGAAGA
AGAACA
AGAACT
AGAACA

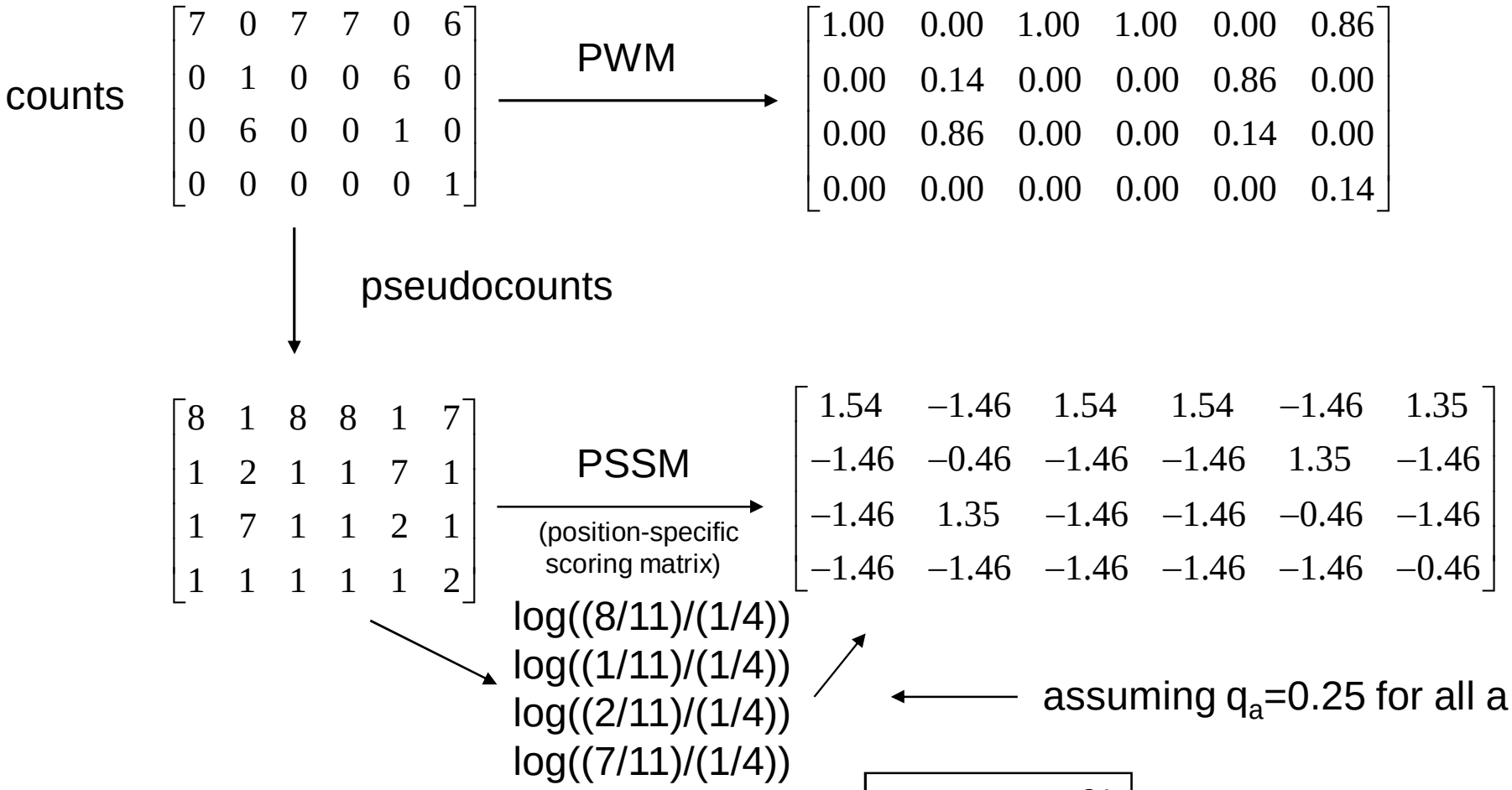
consensus: AGAACA

Position-specific scoring matrix (PSSM)

- PSSM is a log-odds normalized version of PWM. [1]
- Calculated by $\log(p_{ai}/q_a)$, where
  - p_{ai} is the frequency of a at column i in the samples.
  - q_a is the probability of a in the whole organism (or in some region of interest).
- Problematic when some values p_{ai} are zero.
- Solution is to use pseudocounts:
  - add 1 to all the sample counts where the frequencies are calculated.

[1] In the following log denotes base 2 logarithm.

PWM versus PSSM



Extended representations

- PWM representation can be extended to first-order *non-homogenous* Markov chain.
- Identical to what described in lecture 2, except that transition probabilities are position-specific.

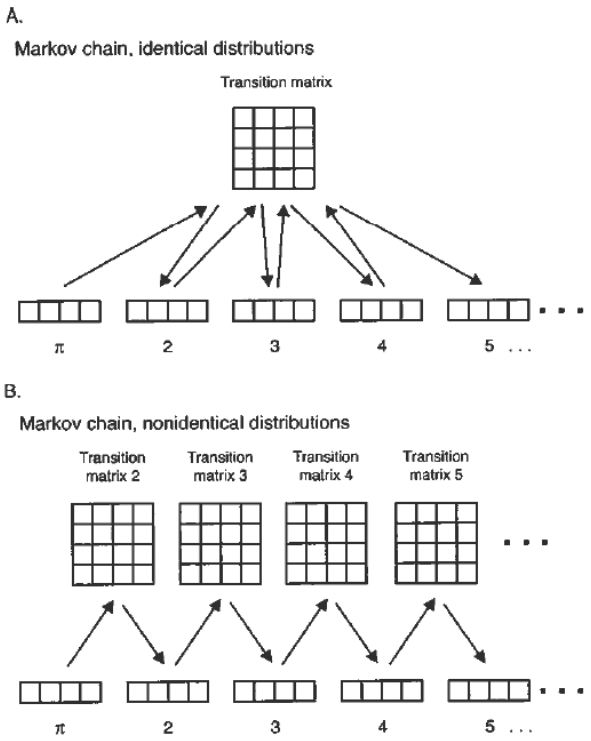


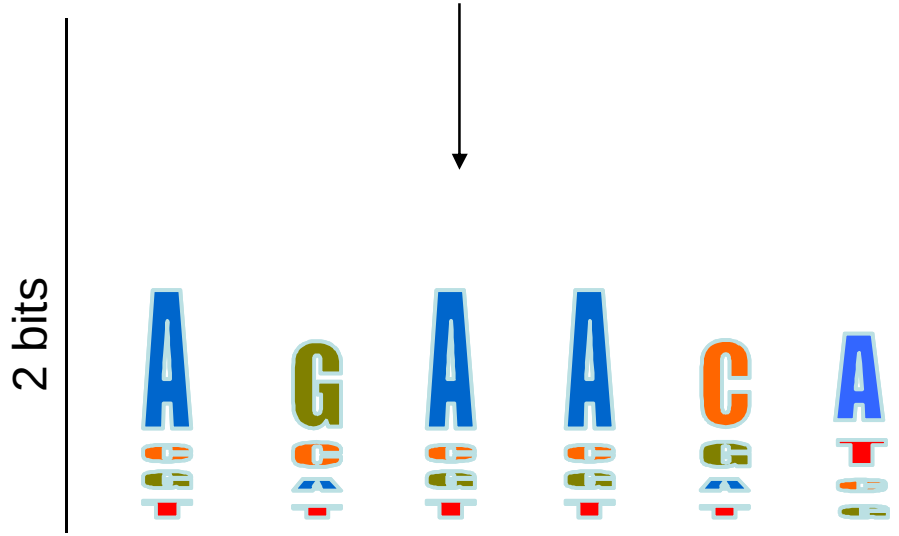
Fig. 9.4. Contrast between the Markov chain representation for positions with identical probability distributions (panel A) and the model for positions in a signal (binding site) sequence (panel B). Numbered four-element horizontal rows are the vectors of probabilities for A, C, G, and T at each position. These vectors are transformed to vectors at the next position by matrix multiplication by a transition matrix. In panel A, a single transition matrix is employed. In panel B, a different transition matrix is required for each successive position.

Sequence logos

- Many known transcription factor binding site PWM:s can be found from JASPAR database (http://jaspar.cgb.ki.se/).
- PWM:s are visualized as *sequence logos*, where the height of each nucleotide equals its proportion of the relative entropy (expected log-odds score) in that column.
  - $\mathrm{E}(S_i) = \sum_a p_{ai} \log(p_{ai} / q_a)$
  - Height of a at column i is $p_{ai}\mathrm{E}(S_i)$

Example sequence logo

$$\begin{bmatrix} 1.54 & -1.46 & 1.54 & 1.54 & -1.46 & 1.35 \\ -1.46 & -0.46 & -1.46 & -1.46 & 1.35 & -1.46 \\ -1.46 & 1.35 & -1.46 & -1.46 & -0.46 & -1.46 \\ -1.46 & -1.46 & -1.46 & -1.46 & -1.46 & -0.46 \end{bmatrix}$$



More on signals in DNA

- Book by Durbin et al.: Biological sequence analysis.
  - Hidden Markov Model (HMM) approach to alignment.
  - Viterbi / forward-backward dynamic programming algorithms for finding most probable paths in the model.
  - Profile-HMMs for sequence family classification.
- 582653 Computational methods of systems biology (4 credits)
  - Professor Esko Ukkonen, II period
  - *The course is an advanced introduction to computational methods for analysing genomic and gene expression data to find different functional units (such as genes) and regulatory structures and relations (such as gene enhancers).*