Bioinformatics (MVE360)

Graham J.L. Kemp

20 January 2014

Content

The course covers basic methods used in sequence analysis such as pairwise and multiple alignment, searching databases for sequence similarity, profiles, pattern matching, hidden Markov models, RNA bioinformatics, gene prediction methods and principles for molecular phylogeny.
The course includes modern high-throughput sequencing techniques and their applications, as well as molecular biology databases and different systems to query such databases.
The course considers theoretical principles as well as how existing programs are being used by bioinformaticians.

Learning outcomes (1)

After completing this course, you should be able to:

- implement solutions to basic bioinformatics problems
- discuss the use of bioinformatics in addressing a range of biological questions
- describe how bioinformatics methods can be used to relate sequence, structure and function
- discuss the technologies for modern high-throughput DNA sequencing and their applications
- use and describe some central bioinformatics data and information resources

Learning outcomes (2)

- describe principles and algorithms of pairwise and multiple alignments, and sequence database searching
- perform pattern matching in biomolecular sequences
- describe how evolutionary relationships can be inferred from sequences (phylogenetics)
- describe the most important principles in gene prediction methods
- describe basic principles of hidden Markov models and their application in sequence analysis

Assessment

Grades will be determined by a written exam at the end of the course.
But in order to pass the course you must also submit solutions to specified exercises:

- one small programming task will be set each week;
- an essay on next generation sequencing and metagenomics.

Course home page

http://www.cse.chalmers.se/edu/year/2014/course/MVE360/

What is bioinformatics?

> *"Research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral or health data, including those to acquire, store, organize, archive, analyze, or visualize such data."*

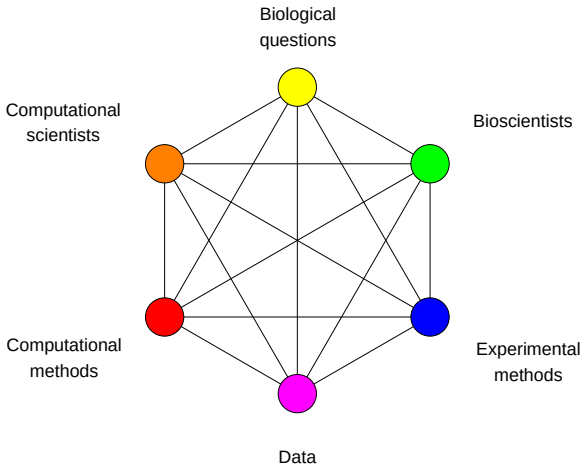
> *"Bioinformatics applies principles of information sciences and technologies to make the vast, diverse, and complex life sciences data more understandable and useful."*

Working definition by the NIH Biomedical Information Science and Technology Initiative Consortium, 2000
http://www.bisti.nih.gov/docs/CompuBioDef.pdf

What is biology?

Ecosystem	Rain forest, desert, fresh water lake, digestive tract of an animal
Community	All species in an ecosystem
Population	All individuals of a single species
Organism	One single individual
Organ System	A specialised functional system of an organism, e.g. nervous system or immune system
Organ	A specialised structural system of an organism, e.g. brain or kidney
Tissue	A specialised substructure of an organ, e.g. nervous tissue, smooth muscle
Cell	A single cell, e.g. neuron, skin cell, stem cell, bacteria
Molecule	e.g. protein, DNA, RNA, sugar, fatty acid, metabolites, pharmaceutical drugs

Addressing biological questions



Sequences, structures and systems

Sequences

- Nucleic acids (DNA and RNA) and proteins are (unbranched) polymers. Their composition can be described by the sequence of units (nucleotides or amino acid residues) in a chain.

Structures

- Three-dimensional structures can give insights into the molecular basis of biological functions.

Systems

- Biological processes consist of the coordinated actions of molecules.

Biological sequences: some experimental methods

- DNA sequencing
- Protein sequencing
- Next-generation sequencing (NGS)

Biological sequences: some questions

- How similar are a pair of sequences?
- Identify the corresponding units in a pair of homologous molecules that have undergone substitutions and insertions/deletions during their evolutionary history (*pairwise sequence alignment*).
- Given a new sequence, has anything similar (in whole or part) been seen before?
- Reconstruct a phylogenetic tree from the sequences of a set of homologous molecules.
- Given the sequences of many overlapping DNA fragments from a single organism, assemble them to reconstruct a full genome.
- Given the sequences of many DNA fragments from a mixture of organisms, identify the species present in the mixture.

Biological structures: some experimental methods

Find the atomic structure of a macromolecule or complex

- X-ray crystallography
- Nuclear magnetic resonance (NMR) spectroscopy

Identify a low-resolution "envelope" enclosing a large macromolecular complex

- Cryo-electron microscopy
- Small-angle x-ray scattering

Biological structures: some questions

- Can differences in the functions of two similar proteins be explained by differences in their structures?
- Can a drug be designed to fit into the active site of a target protein?
- Can the safety and efficacy of a potential therapeutic protein be predicted from its structure?
- Can the function of a protein be altered by changing its composition, and hence its structure?
- Can a protein's structure be predicted from its sequence?
  - the protein folding problem
- Given the structures of two proteins, will they associate with one another? If so, how will they fit together?
  - the protein docking problem

Biological systems: some experimental methods

Which mRNA molecules are being expressed?

- Microarray gene expression
- RNA-Seq

Which proteins are being expressed?

- (2-D) gel electrophoresis
- Mass spectrometry

In which tissue(s) are particular genes expressed?

- *in situ* hybridization

Biological systems: some questions

- Which genes/proteins are co-expressed (i.e. have similar expression profiles)?
- Which genes are expressed in tumour cells but not in healthy cells?
- If a gene is "knocked out", will an organism survive, and how will the expression of other genes be affected?
- Can protein expression profiles identify proteins that could be targets for drug development?
- Can an individual's expression profile indicate whether they are likely to respond to a particular therapeutic treatment?
- How do biological networks respond to injury or to treatment with a therapeutic drug?

Measures of sequence similarity

Hamming distance:

Number of positions with mismatching characters.

Defined for two strings of equal length.

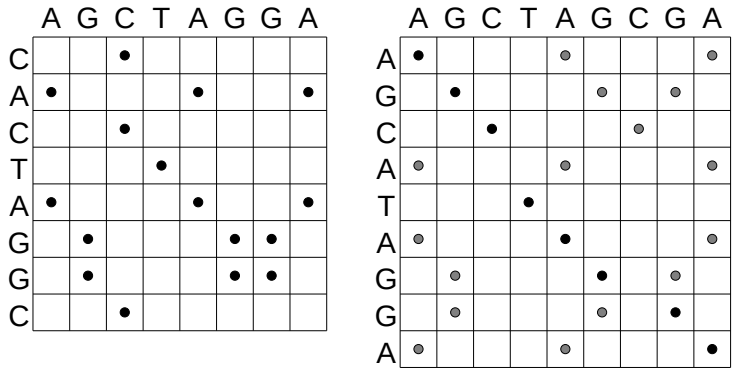
```
agtc
cgta
```

Levenshtein distance:

Minimum number of edit operations (delete, insert, change a single character) needed to change one sequence into another.

```
agtcc
cgctca
```


Dotplots



Dotplots

A pictorial representation of the similarity between two sequences.

Compare a sequence with itself:

Repeats

Palindromic sequences

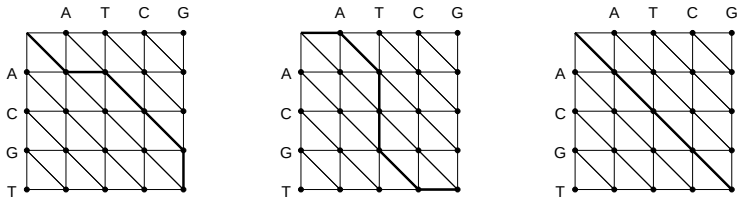
Compare two sequences:

Any path from upper left to lower right represents an alignment.

Horizontal or vertical moves correspond to gaps in one of the sequences.

Path with highest score corresponds to an optimal alignment.

Each path represents an alignment




```
A-CGT
| ||
ATCG-
```



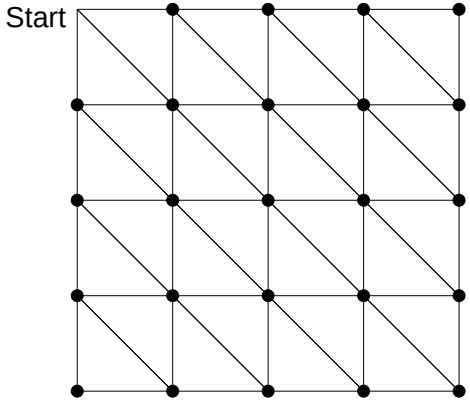
```
-ACGT-

AT--CG
```



```
ACGT
|
ATCG
```

- Vertical steps add a gap to the horizontal sequence
- Horizontal steps add a gap to the vertical sequence

How many paths?



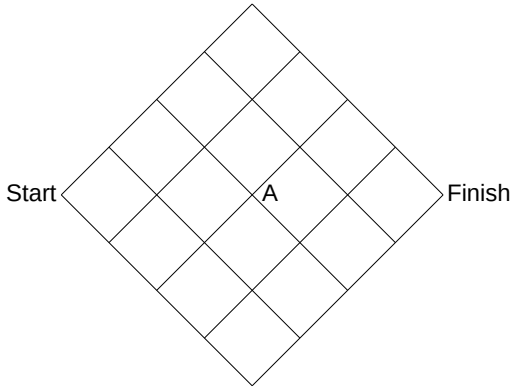
Pairwise global alignment (Needleman-Wunsch algorithm)

Rigorous algorithms use dynamic programming to find an optimal alignment.

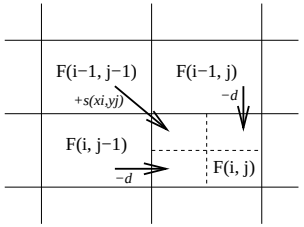
- match score
- mismatch score
- gap penalty

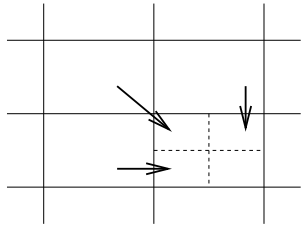
$$F(i,\,j) = \max \begin{cases} F(i-1,\,j-1) + s(x_{i,}y_{j}) \\ F(i-1,\,j) - d \\ F(i,\,j-1) - d \end{cases}$$

Do we have to enumerate all paths?

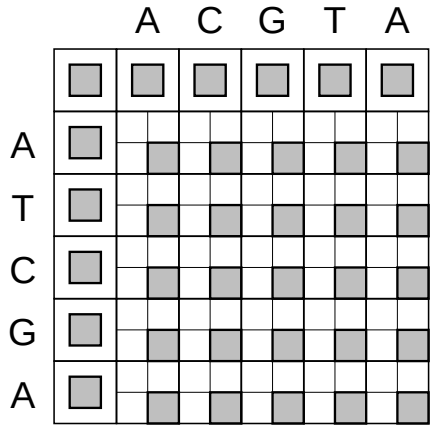


Dynamic programming





Score matrix



Is the similarity significant, or could it be due to chance?

Even if two proteins are unrelated, we would expect some similarity simply by chance.

Is the alignment score significantly higher than random?

Align random permutations of the sequences, and find the mean and standard deviation of the resulting distribution.

The z-score reflects the significance of a global similarity score.

$$z\text{-}score = \frac{score - mean}{standard\ deviation}$$

Larger values imply greater significance.

Percent identity

Having obtained an alignment, it is common to quantify the similarity between a pair of sequences by stating the percent identity.

```
-ACGATAG-CGAAACCAAAA
 ||| ||| |||    |
CAGC-TAGCCGATGTC----
```

Count the number of alignment positions with matching characters and divide by ... *what*?

- the length of the shortest sequence?
- the length of the alignment?
- the average length of the sequences?
- the number of non-gap positions?
- the number of equivalenced positions excluding overhangs?

Pairwise local alignment (Smith-Waterman algorithm)

Local similarities may be masked by long unrelated regions.

A minor modification to the global alignment algorithm.

- If the score for a subalignment becomes negative, set the score to zero.

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

- Trace back from the position in the score matrix with the highest value.
- Stop at cell where score is zero.

global_alignment.c

```
#include <stdio.h>

#define MAX_LENGTH     100

#define MATCH_SCORE      2
#define MISMATCH_SCORE  -1
#define GAP_PENALTY      2

#define STOP             0
#define UP               1
#define LEFT             2
#define DIAG             3


main()
{
    int  i, j;
    int  m, n;
    int  alignmentLength, score, tmp;
    char X[MAX_LENGTH+1] = "ATCGAT";
    char Y[MAX_LENGTH+1] = "ATACGT";

    int  F[MAX_LENGTH+1][MAX_LENGTH+1]; /* score matrix */
    int  trace[MAX_LENGTH+1][MAX_LENGTH+1];
    char alignX[MAX_LENGTH*2];   /* aligned X sequence */
    char alignY[MAX_LENGTH*2];   /* aligned Y sequence */


    /*
     * Find lengths of (null-terminated) strings X and Y
     */
    m = 0;
    n = 0;
    while ( X[m] != 0 ) {
        m++;
    }
    while ( Y[n] != 0 ) {
        n++;
    }


    /*
     * Initialise matrices
     */

    F[0][0] = 0;
    trace[0][0] = STOP;
    for ( i=1 ; i<=m ; i++ ) {
        F[i][0] = F[i-1][0] - GAP_PENALTY;
        trace[i][0] = STOP;
    }
    for ( j=1 ; j<=n ; j++ ) {
        F[0][j] = F[0][j-1] - GAP_PENALTY;
        trace[0][j] = STOP;
    }



    /*
     * Fill matrices
     */

for ( i=1 ; i<=m ; i++ ) {

    for ( j=1 ; j<=n ; j++ ) {

        if ( X[i-1]==Y[j-1] ) {
            score = F[i-1][j-1] + MATCH_SCORE;
        } else {
            score = F[i-1][j-1] + MISMATCH_SCORE;
        }
        trace[i][j] = DIAG;

        tmp = F[i-1][j] - GAP_PENALTY;
        if ( tmp>score ) {
            score = tmp;
            trace[i][j] = UP;
        }

        tmp = F[i][j-1] - GAP_PENALTY;
        if( tmp>score ) {
            score = tmp;
            trace[i][j] = LEFT;
        }

        F[i][j] = score;
     }
}


/*
 * Print score matrix
 */

printf("Score matrix:\n      ");
for ( j=0 ; j<n ; ++j ) {
    printf("%5c", Y[j]);
}
printf("\n");
for ( i=0 ; i<=m ; i++ ) {
    if ( i==0 ) {
        printf(" ");
    } else {
        printf("%c", X[i-1]);
    }
    for ( j=0 ; j<=n ; j++ ) {
        printf("%5d", F[i][j]);
    }
    printf("\n");
}
printf("\n");



/*
 * Trace back from the lower-right corner of the matrix
 */

i = m;
j = n;
alignmentLength = 0;

while ( trace[i][j] != STOP ) {

    switch ( trace[i][j] ) {

        case DIAG:
            alignX[alignmentLength] = X[i-1];
            alignY[alignmentLength] = Y[j-1];
            i--;
            j--;
            alignmentLength++;
            break;

        case LEFT:
            alignX[alignmentLength] = '-';
            alignY[alignmentLength] = Y[j-1];
            j--;
            alignmentLength++;
            break;

        case UP:
            alignX[alignmentLength] = X[i-1];
            alignY[alignmentLength] = '-';
            i--;
            alignmentLength++;
    }
}


/*
 * Unaligned beginning
 */

while ( i>0 ) {
    alignX[alignmentLength] = X[i-1];
    alignY[alignmentLength] = '-';
    i--;
    alignmentLength++;
}

while ( j>0 ) {
    alignX[alignmentLength] = '-';
    alignY[alignmentLength] = Y[j-1];
    j--;
    alignmentLength++;
}



/*
 * Print alignment
 */

for ( i=alignmentLength-1 ; i>=0 ; i-- ) {
    printf("%c",alignX[i]);
}
printf("\n");
for ( i=alignmentLength-1 ; i>=0 ; i-- ) {
    printf("%c",alignY[i]);
}
printf("\n");

return(1);
}
```