# Sequence alignment and search

## A review on alignment and search tools

Wenyuan Jiang[1]

[1]School of Life Science
Tongji University

Class Presentation, Mar 2021

# Table of Contents

# How LARGE the data is

## Bit

The bit is a basic unit of information in computing and digital communications. The bit represents a logical state with one of two possible values, 0 or 1.

# How LARGE the data is

## Bit

The bit is a basic unit of information in computing and digital communications. The bit represents a logical state with one of two possible values, 0 or 1.

## Byte

The byte is a unit of digital information that most commonly consists of eight bits. Usually, B represents byte and b represents bit.

# How LARGE the data is

## Bit

The bit is a basic unit of information in computing and digital communications. The bit represents a logical state with one of two possible values, 0 or 1.

## Byte

The byte is a unit of digital information that most commonly consists of eight bits. Usually, B represents byte and b represents bit.

## Magnitude

B -> KiB -> MiB -> GiB -> TiB -> PiB -> EiB ...
(Each -> represents a magnitude of 1024.)

# How FAST an algorithm is

## Big O notation

Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

# How FAST an algorithm is

## Big O notation

Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

## Formal definition

Let $f$ be a real or complex valued function and g a real valued function. Let both functions be defined on some unbounded subset of the positive real numbers, and $g(x)$ be strictly positive for all large enough values of x.

$$f(x) = O(g(x))(x \to \infty)$$

# How FAST an algorithm is

## Application of Big O notation

Big O notation can be used to compare the speed of different algorithm.

# How FAST an algorithm is

## Application of Big O notation

Big O notation can be used to compare the speed of different algorithm.

## Example

Informally, especially in computer science, the big O notation often can be used to describe an asymptotic tight bound. For example, when considering a function $T(n) = 73n^3 + 22n^2 + 58$, we can say:

$$T(n) = O(n^3)$$

Where $n$ often represents the data amount of an input, and $T(n)$ is the time for execution of an algorithm.

# How FAST an algorithm is

## Comparing algorithms

Big O notation can compare the absolute speed of an algorithm as well as the relative speed.

- $O(nlog(n))$
- $O(n^k), (k > 1)$
- $O(2^n)$

# How FAST an algorithm is

## Comparing algorithms

Big O notation can compare the absolute speed of an algorithm as well as the relative speed.

- $O(nlog(n))$
- $O(n^k), (k > 1)$
- $O(2^n)$

## Example

Bubble sort has a speed of $O(n^2)$, while Quick sort has a speed of $O(nlog(n))$.
Therefore, on the same machine (with the same amount of input $n$), Quick sort is faster than Bubble sort. If we run Quick sort on a slow machine and Bubble sort on a fast machine, when $n$ gets larger, Quick sort will eventually run faster than Bubble sort.

# How ACCURATE an algorithm is

## Traditional algorithms and Heuristic algorithms

For Traditional algorithms, their output is definite and accurate. While for Heuristic algorithms, especially when ramdomness is involved, their output is not stable.

# How ACCURATE an algorithm is

## Traditional algorithms and Heuristic algorithms

For Traditional algorithms, their output is definite and accurate. While for Heuristic algorithms, especially when ramdomness is involved, their output is not stable.

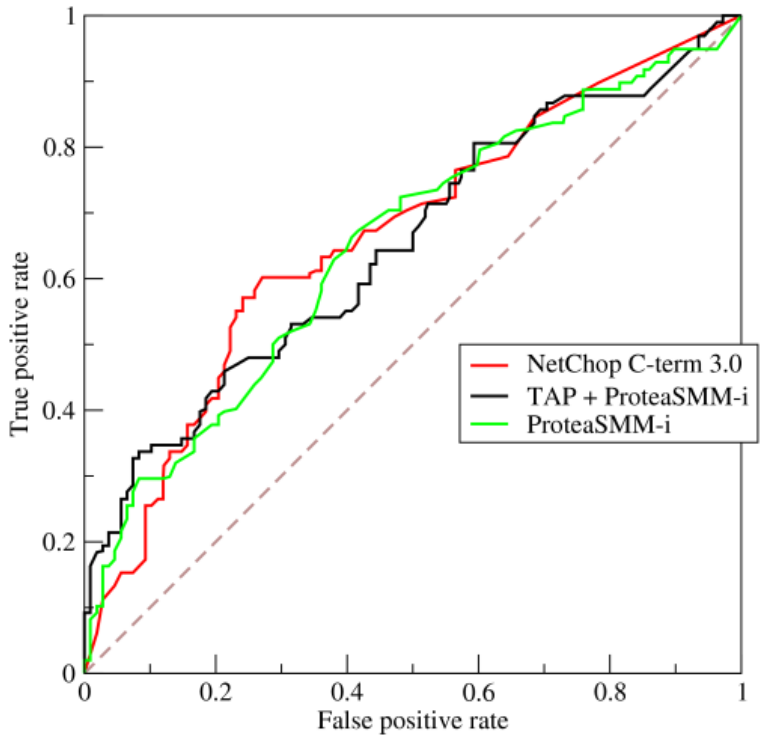
## Terminology

- TN, true negative
- FP, false positive
- FN, false negative
- TPR, true positive rate, $TPR = TP/P = TP/(TP + FN)$
- FPR, false positive rate, $FPR = FP/N = FP/(FP + TN)$
- ACC, accuracy, $ACC = (TP + TN)/(P + N)$

# How ACCURATE an algorithm is

## ROC curve

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

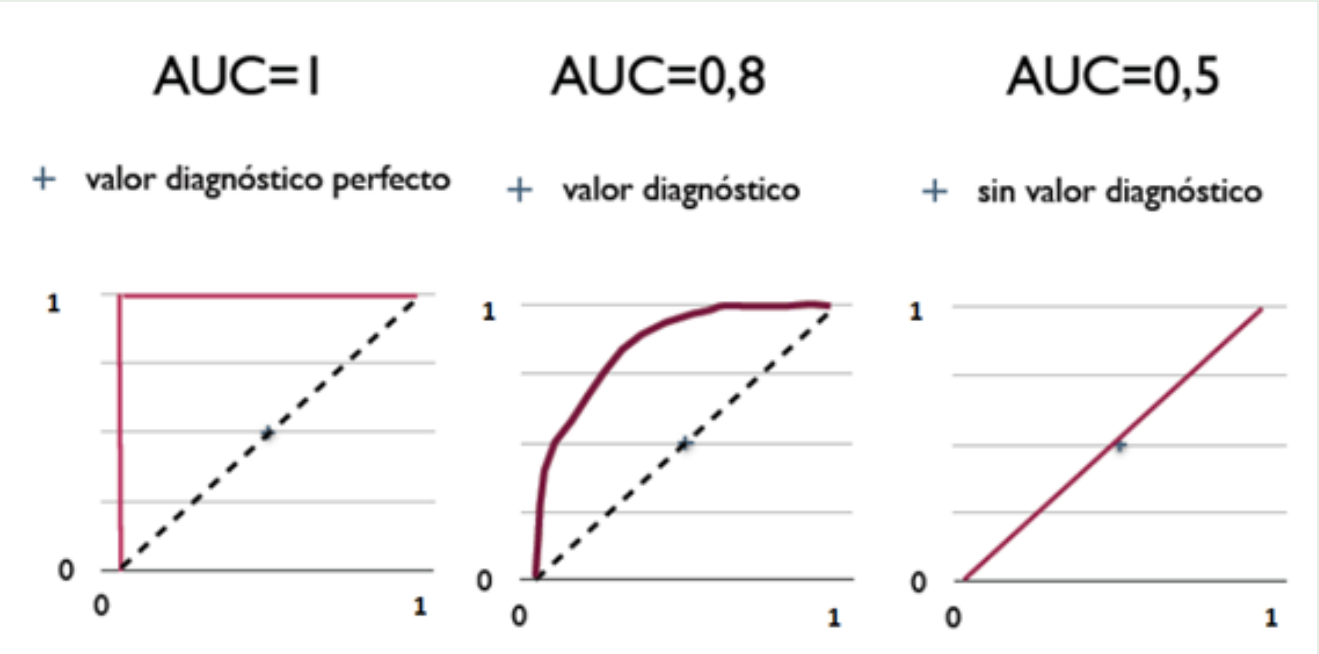


# How ACCURATE an algorithm is

## AUC

AUC is the area under the ROC curve.

## Example



# Table of Contents

# Pairwise alignment

## Pairwise alignment

Pairwise sequence alignment methods are used to find the best-matching piecewise (local or global) alignments of two query sequences.

# Pairwise alignment

## Pairwise alignment

Pairwise sequence alignment methods are used to find the best-matching piecewise (local or global) alignments of two query sequences.

## Usage of Pairwise alignment

Pairwise alignments can only be used between two sequences at a time, but they are efficient to calculate and are often used for methods that do not require extreme precision (such as searching a database for sequences with high similarity to a query).

# Pairwise alignment

## Algorithms for Pairwise alignment

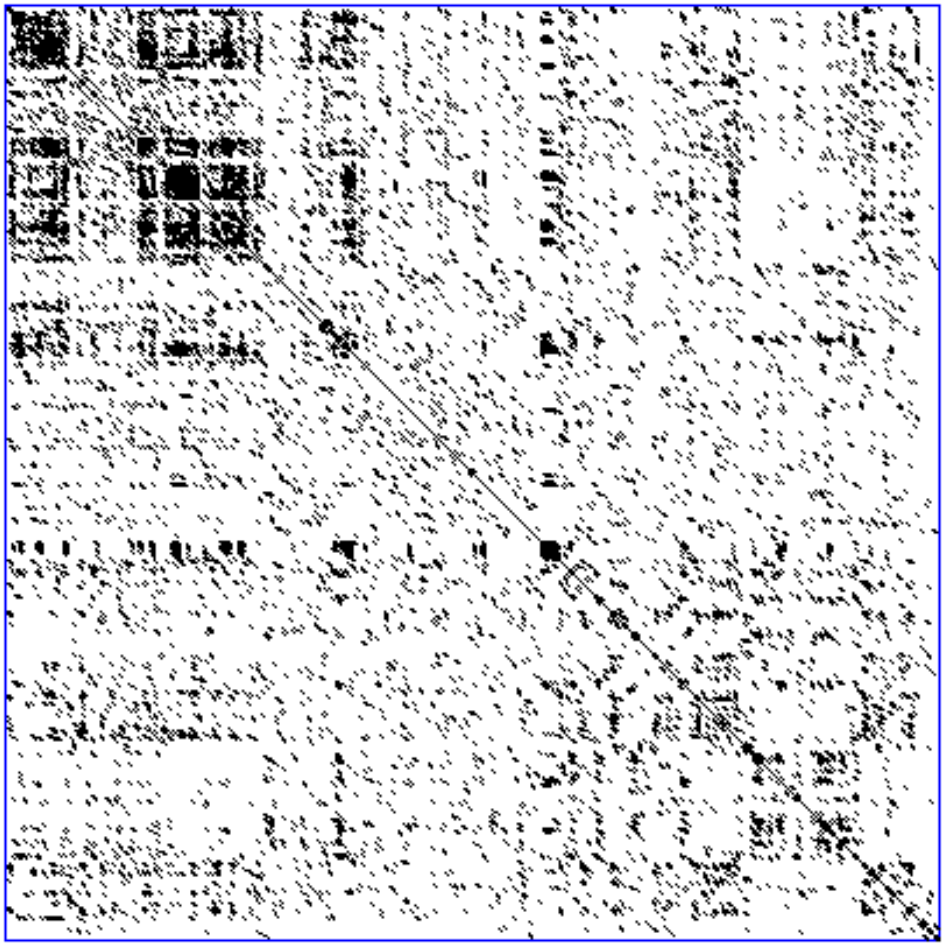
- Dot-matrix methods
- Dynamic programming
- Word methods

# Pairwise alignment: Dot-matrix

## Dot-matrix methods

The dot-matrix approach, which implicitly produces a family of alignments for individual sequence regions, is qualitative and conceptually simple, though time-consuming to analyze on a large scale. In the absence of noise, it can be easy to visually identify certain sequence features—such as insertions, deletions, repeats, or inverted repeats—from a dot-matrix plot. To construct a dot-matrix plot, the two sequences are written along the top row and leftmost column of a two-dimensional matrix and a dot is placed at any point where the characters in the appropriate columns match—this is a typical recurrence plot.

# Pairwise alignment



# Pairwise alignment: Dot-matrix

## Dot-matrix tool

- Dotter: A dot-matrix program with interactive greyscale rendering for genomic DNA and Protein sequence analysis

# Pairwise alignment: Dot-matrix

## Dot-matrix tool

- Dotter: A dot-matrix program with interactive greyscale rendering for genomic DNA and Protein sequence analysis

## Dot-matrix method advantage

- Fairly easy to Implement.
- Easy to understand visually.
- It shows all possible alignment of pairs.
- Good overview of places fot good alignment.

# Pairwise alignment: Dot-matrix

## Dot-matrix tool

- Dotter: A dot-matrix program with interactive greyscale rendering for genomic DNA and Protein sequence analysis

## Dot-matrix method advantage

- Fairly easy to Implement.
- Easy to understand visually.
- It shows all possible alignment of pairs.
- Good overview of places fot good alignment.

## Dot-matrix method speed

For two sequences that have the length of $n$:

$$O(n^2)$$

# Pairwise alignment: Dynamic programming

## Dynamic programming

The technique of dynamic programming can be applied to produce global alignments via the Needleman-Wunsch algorithm, and local alignments via the Smith-Waterman algorithm. In typical usage, protein alignments use a substitution matrix to assign scores to amino-acid matches or mismatches, and a gap penalty for matching an amino acid in one sequence to a gap in the other. DNA and RNA alignments may use a scoring matrix, but in practice often simply assign a positive match score, a negative mismatch score, and a negative gap penalty.

# Pairwise alignment: Dynamic programming

## Input and output

The input of the algorithm is two sequences, and the output is a score representing how similar the sequences are. On most cases, a pattern showing the matches will be given.

For example:

```
TACGGGCCCGCTA-C
||---|-||-|||-|
TA---G-CC-CTATC
```

# Pairwise alignment: Dynamic programming

## Dynamic programming tool

- EMBOSS Water
- EMBOSS Needle
- `http://emboss.sourceforge.net/`

# Pairwise alignment: Dynamic programming

## Dynamic programming tool

- EMBOSS Water
- EMBOSS Needle
- `http://emboss.sourceforge.net/`

## Dynamic programming method speed

For two sequences that have the length of $n$:

$$O(n^2)$$

# Pairwise alignment: Word methods

## Word methods

Word methods, also known as k-tuple methods, are heuristic methods that are not guaranteed to find an optimal alignment solution, but are significantly more efficient than dynamic programming. These methods are especially useful in large-scale database searches where it is understood that a large proportion of the candidate sequences will have essentially no significant match with the query sequence.

# Pairwise alignment: Word methods

## Word methods

Word methods, also known as k-tuple methods, are heuristic methods that are not guaranteed to find an optimal alignment solution, but are significantly more efficient than dynamic programming. These methods are especially useful in large-scale database searches where it is understood that a large proportion of the candidate sequences will have essentially no significant match with the query sequence.

## Word methods speed

For two sequences that have the length of $n$:

$$O(n^k), 1 < k < 2$$

# Pairwise alignment: Word methods

## Word methods tool

- EMBL FASTA http://www.ebi.ac.uk/fasta33/
- NCBI BLAST https://blast.ncbi.nlm.nih.gov/Blast.cgi

# Multiple sequence alignment

## Modified pairwise alignment algorithms

Most Pairwise alignment algorithms can be easily modified for Multiple sequence alignment. However, such algorithms are too slow to handle massive amount of data.

# Multiple sequence alignment

## Modified pairwise alignment algorithms

Most Pairwise alignment algorithms can be easily modified for Multiple sequence alignment. However, such algorithms are too slow to handle massive amount of data.

## Specialized agorithms and tools

Some algorithms are designed for fast multiple sequence alignment, but most of them are not sensitive compared with Pairwise alignment algorithms.

Below are some tools.

- Clustal
- T-Coffee
- LINCLUST
- MMseqs2
- UCLUST

# Table of Contents

# MMseqs2 Publications


- Steinegger M and Soeding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nature Biotechnology, doi: 10.1038/nbt.3988 (2017).
- Steinegger M and Soeding J. Clustering huge protein sequence sets in linear time. Nature Communications, doi: 10.1038/s41467-018-04964-5 (2018).
- Mirdita M, Steinegger M and Soeding J. MMseqs2 desktop and local web server app for fast, interactive sequence searches. Bioinformatics, doi: 10.1093/bioinformatics/bty1057 (2019).
- Mirdita M, Steinegger M, Breitwieser F, Soding J, Levy Karin E: Fast and sensitive taxonomic assignment to metagenomic contigs. bioRxiv, doi: 10.1101/2020.11.27.401018 (2020).

# Install MMseqs2

Before installation, read the documents of the source code carefully, and make sure you understand all the instructions in the doc.

## Install gcc, cmake and libz

To compile MMseqs2 `git`, `g++ (4.9 or higher)` and `cmake (2.8.12 or higher)` are needed. `Zlib` and `Bzlib` are optional.

```
apt-get install build-essential cmake zlib1g zlib1g-devel libbz2-dev
```

# Install MMseqs2

Before installation, read the documents of the source code carefully, and make sure you understand all the instructions in the doc.

## Install gcc, cmake and libz

To compile MMseqs2 `git, g++ (4.9 or higher)` and `cmake (2.8.12 or higher)` are needed. `Zlib` and `Bzlib` are optional.

```
apt-get install build-essential cmake zlib1g zlib1g-devel libbz2-dev
```

## Download source code

```
git clone https://github.com/soedinglab/MMseqs2.git && cd MMseqs2
```

# Install MMseqs2

Before installation, read the documents of the source code carefully, and make sure you understand all the instructions in the doc.

## Install gcc, cmake and libz

To compile MMseqs2 `git, g++ (4.9 or higher)` and `cmake (2.8.12 or higher)` are needed. `Zlib` and `Bzlib` are optional.

```
apt-get install build-essential cmake zlib1g zlib1g-devel libbz2-dev
```

## Download source code

```
git clone https://github.com/soedinglab/MMseqs2.git && cd MMseqs2
```

## Compile and Install

```
mkdir build && cd build/
cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr ..
make -j && make install
```

# Use MMseqs2: Search

## Create a database

You can create a database from a FASTA file.

```
mmseqs createdb examples/DB.fasta targetDB
mmseqs createindex targetDB tmp
```

Or you can download a database online.

```
mmseqs databases UniProtKB/Swiss-Prot swissprot tmp
```

# Use MMseqs2: Search

## Create a database

You can create a database from a FASTA file.

```
mmseqs createdb examples/DB.fasta targetDB
mmseqs createindex targetDB tmp
```

Or you can download a database online.

```
mmseqs databases UniProtKB/Swiss-Prot swissprot tmp
```

## Search

You can search some sequences in a fasta file.

```
mmseqs easy-search examples/QUERY.fasta examples/DB.fasta alnRes.m8 tmp
```

Or search some sequences in a pre-built database, which is much faster.

```
mmseqs easy-search examples/QUERY.fasta targetDB alnRes.m8 tmp
```

# Use MMseqs2: Search

## Search Results

The results of a search is in m8 format, which is a tab-separated ascii file with lines like this:

```
A0A061HXN7 P13020 0.953 780 37 0 1 778 1 780 0.000E+00 1521
```

Each column has the definition listed below:

- Query id
- Subject id
- Identity
- Alignment length
- Mismatches
- Gap Openings
- q. start
- q. end
- s. start
- s. end
- e-value
- bit score

# Use MMseqs2: Cluster

## Clustering FASTA files

For clustering, MMseqs2 `easy-cluster` and `easy-linclust` are available.

```
mmseqs easy-cluster examples/DB.fasta clusterRes tmp
mmseqs easy-linclust examples/DB.fasta clusterRes tmp
```

# Use MMseqs2: Cluster

## Clustering FASTA files

For clustering, MMseqs2 `easy-cluster` and `easy-linclust` are available.

```
mmseqs easy-cluster examples/DB.fasta clusterRes tmp
mmseqs easy-linclust examples/DB.fasta clusterRes tmp
```

## Cluster results

The commands above produces three files: `clusterRes_cluster.tsv`, `clusterRes_all_seqs.fasta` and `clusterRes_rep_seq.fasta`.

# Use MMseqs2: Cluster

## Cluster results

The `clusterRes_cluster.tsv` file follows the following format:

```
#cluster-representative cluster-member
Q0KJ32 Q0KJ32
Q0KJ32 C0W539
Q0KJ32 D6KVP9
...
```

# Use MMseqs2: Cluster

## Cluster results

The `clusterRes_cluster.tsv` file follows the following format:

```
#cluster-representative cluster-member
Q0KJ32 Q0KJ32
Q0KJ32 C0W539
Q0KJ32 D6KVP9
...
```

## Cluster results

The `clusterRes_all_seqs.fasta` is FASTA-like format file, with a new cluster being marked by two identical name lines of the representative sequence.

The `clusterRes_rep_seq.fasta` contains all representative sequences.

# MMseqs2: Performance

## Computing environment

- Hardware: Xeon Gold 5117 @2.00GHz 28C56T, 32GB RAM, SSD
- Software: Ubuntu 18.04, MMseqs2 Release 13 compiled with gcc 7.5.0

# MMseqs2: Performance

## Computing environment

- `Hardware: Xeon Gold 5117 @2.00GHz 28C56T, 32GB RAM, SSD`
- `Software: Ubuntu 18.04, MMseqs2 Release 13 compiled with gcc 7.5.0`

## Data size

The input file is `swiss.fasta`, which contains `1129276` seqs from UniProtKB.
The size of the file is `264MB`

# MMseqs2: Performance

## Performance

Using command

```
time mmseqs easy-cluster examples/swiss.fasta clusterRes tmp
```

to do the clustering, it took: 1m21.693s.

# MMseqs2: Performance

## Performance

Using command

```
time mmseqs easy-cluster examples/swiss.fasta clusterRes tmp
```

to do the clustering, it took: `1m21.693s`.

## Performance

Using command

```
time mmseqs easy-linclust examples/swiss.fasta clusterRes tmp
```

to do the clustering, it took: `0m16.869s`.

# Table of Contents

# Algorithm?

- Parallel computing?
- Machine learning based methods?
- Quantum computing?

# Hardware?

- GPU?
- CPU Clusters?
- FPGA?
- Playstation?
- Wirawan A, Kwoh C K, Hieu N T, et al. CBESW: sequence alignment on the playstation 3

# Table of Contents

# Q & A

# References


- Likic V. The Needleman-Wunsch algorithm for sequence alignment[J]. Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne, 2008: 1-46.
- Nevill-Manning C G, Huang C N, Brutlag D L. Pairwise protein sequence alignment using Needleman-Wunsch and Smith-Waterman algorithms[J]. Personal communication (http://motif.stanford.edu/alion/), 1997.
- Higgins D G, Sharp P M. Fast and sensitive multiple sequence alignments on a microcomputer[J]. Bioinformatics, 1989, 5(2): 151-153.
- Steinegger M, Söding J. Clustering huge protein sequence sets in linear time[J]. Nature communications, 2018, 9(1): 1-8.
- Polyanovsky V O, Roytberg M A, Tumanyan V G. Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences[J]. Algorithms for molecular biology, 2011, 6(1): 1-12.
- Wirawan A, Kwoh C K, Hieu N T, et al. CBESW: sequence alignment on the playstation 3[J]. BMC bioinformatics, 2008, 9(1): 1-10.