Lecture 10

Sequence Alignments

Part 1: Alignments Between Sequences

Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

Sequence Alignments

Alignments Between Sequences

We have two related sequences of possibly different lengths.

How to best line them up using insertions / deletions (i.e., monotonically)?

Alignments Between Sequences

We have two related sequences of possibly different lengths.

How to best line them up using insertions / deletions (i.e., monotonically)?

biology:

DNA, RNA, or protein
sequences:

align `CAAT` and `ATTACA`:

```
--CA-AT
ATTACA-
```

Alignments Between Sequences

We have two related sequences of possibly different lengths.

How to best line them up using insertions / deletions (i.e., monotonically)?

biology:

DNA, RNA, or protein sequences:

align `CAAT` and `ATTACA`:

```
--CA-AT
ATTACA-
```

nlp:

find the best sequence of edits between strings

(e.g., spell checking etc)

```
kitten-
sitting
```

Alignments Between Sequences

We have two related sequences of possibly different lengths.

How to best line them up using insertions / deletions (i.e., monotonically)?

biology:

DNA, RNA, or protein sequences:

align `CAAT` and `ATTACA`:

```
--CA-AT
ATTACA-
```

nlp:

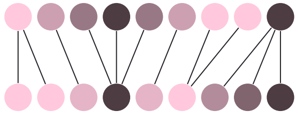
find the best sequence of edits between strings

(e.g., spell checking etc)

```
kitten-
sitting
```

signal processing:

stretch or compress signals (e.g., audio) to match.



Alignment Are Structures

Alignments are structured objects: many possible alignments between same strings.

```
--CA-AT   -CAAT-    CAAT--    CAAT------   CAAT-----
ATTACA-   ATTACA   ATTACA     ----ATTACA   ---ATTACA  . . .
```

Alignment Are Structures

Alignments are structured objects: many possible alignments between same strings.

```
--CA-AT   -CAAT-    CAAT--    CAAT------   CAAT-----
ATTACA-   ATTACA   ATTACA     ----ATTACA   ---ATTACA  ...
```

We need to figure out:

1. How to score an alignment
2. How to compute argmax over alignments
3. How to compute logsumexp over alignments

Alignment Are Structures

Alignments are structured objects: many possible alignments between same strings.

```
--CA-AT    -CAAT-     CAAT--     CAAT------    CAAT-----
ATTACA-    ATTACA    ATTACA      ----ATTACA    ---ATTACA   . . .
```

We need to figure out:

0. How to represent alignments

1. How to score an alignment

2. How to compute argmax over alignments

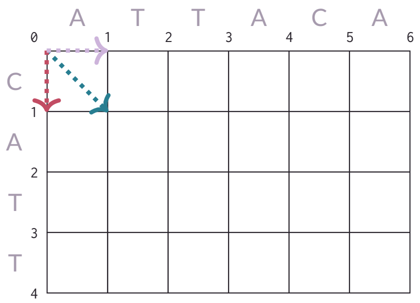
3. How to compute logsumexp over alignments

Machine Learning for Structured Data

Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

Sequence Alignments

Alignment Tables

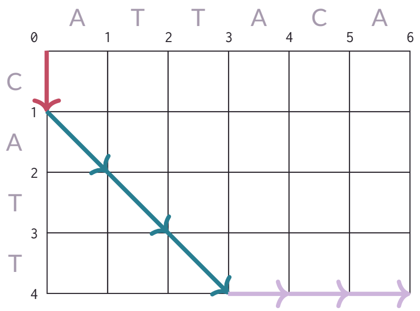


At point (i, j) in the grid we either:

- **M**: match tokens i in `seq1` to j in `seq2`,
- **I**: skip token i in `seq1`,
- **D**: skip token j in `seq2`.

Some alignments and corresponding trajectories:

Alignment Tables



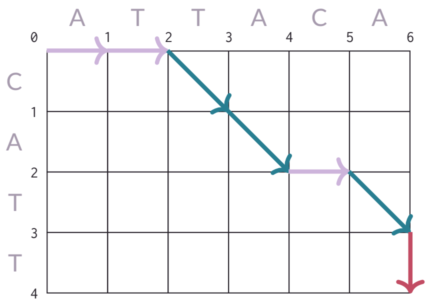
At point (i,j) in the grid we either:

- **M**: match tokens i in `seq1` to j in `seq2`,
- **I**: skip token i in `seq1`,
- **D**: skip token j in `seq2`.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
  ```
  CATT---
  -ATTACA
  ```

Alignment Tables



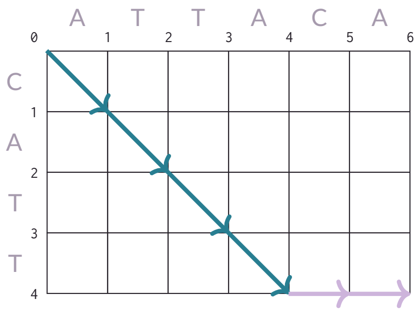
At point (i, j) in the grid we either:

- **M**: match tokens i in seq1 to j in seq2,
- **I**: skip token i in seq1,
- **D**: skip token j in seq2.

Some alignments and corresponding trajectories:

- IMMMDDD:
```
CATT---
-ATTACA
```
- DDMMDMI:
```
--CA-TT
ATTACA-
```

Alignment Tables



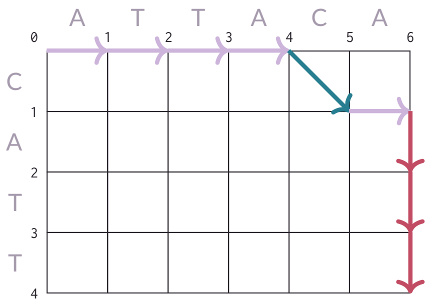
At point (i, j) in the grid we either:

- **M**: match tokens i in `seq1` to j in `seq2`,
- **I**: skip token i in `seq1`,
- **D**: skip token j in `seq2`.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
```
CATT---
-ATTACA
```
- `DDMMDMI`:
```
--CA-TT
ATTACA-
```
- `MMMMDD`:
```
CATT--
ATTACA
```

Alignment Tables



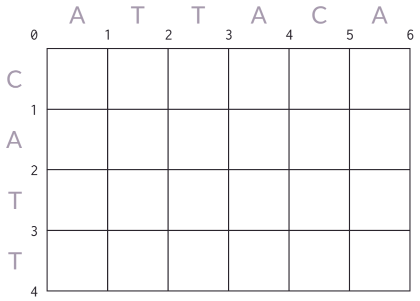
At point (i, j) in the grid we either:

- **M**: match tokens *i* in `seq1` to *j* in `seq2`,
- **I**: skip token *i* in `seq1`,
- **D**: skip token *j* in `seq2`.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
  ```
  CATT---
  -ATTACA
  ```
- `DDMMDMI`:
  ```
  --CA-TT
  ATTACA-
  ```
- `MMMMDD`:
  ```
  CATT--
  ATTACA
  ```
- `DDDDMDIII`:
  ```
  ----C-ATT
  ATTACA---
  ```

Alignment Tables



DAG representation:

Nodes at grid points
$V = \{(i,j) : 0 \leq i \leq n, 0 \leq j \leq m\}$

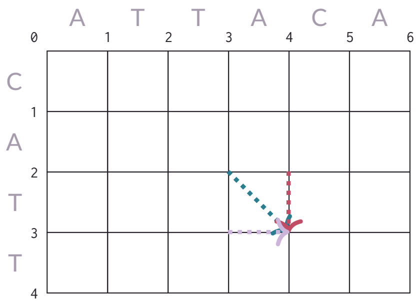
At point (i, j) in the grid we either:

- **M**: match tokens i in seq1 to j in seq2,
- **I**: skip token i in seq1,
- **D**: skip token j in seq2.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
  ```
  CATT---
  -ATTACA
  ```
- `DDMMDMI`:
  ```
  --CA-TT
  ATTACA-
  ```
- `MMMMDD`:
  ```
  CATT--
  ATTACA
  ```
- `DDDDMDIII`:
  ```
  ----C-ATT
  ATTACA---
  ```

Alignment Tables



DAG representation:

Nodes at grid points
$V = \{(i,j) : 0 \leq i \leq n, 0 \leq j \leq m\}$

Three incoming edges for each node.

At point (i,j) in the grid we either:

M: match tokens i in `seq1` to j in `seq2`,

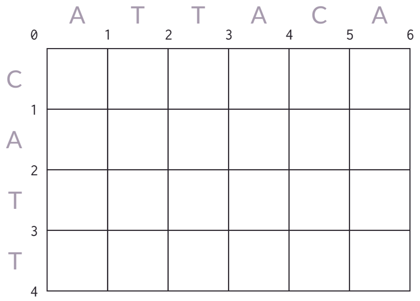
I: skip token i in `seq1`,

D: skip token j in `seq2`.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
  ```
  CATT---
  -ATTACA
  ```
- `DDMMDMI`:
  ```
  --CA-TT
  ATTACA-
  ```
- `MMMMDD`:
  ```
  CATT--
  ATTACA
  ```
- `DDDDMDIII`:
  ```
  ----C-ATT
  ATTACA---
  ```

Alignment Tables



DAG representation:

Nodes at grid points
$V = \{(i, j) : 0 \leq i \leq n, 0 \leq j \leq m\}$

Three incoming edges for each node.

Number of paths from $(0, 0)$ to (n, m):
$D(n, m) = \sum_{k=0}^{\min(n,m)} \binom{m}{k}\binom{n}{k} 2^k$ (Delannoy numbers)

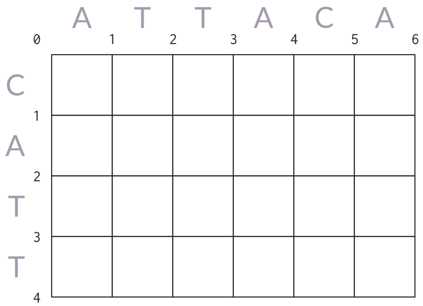
At point (i, j) in the grid we either:

- **M**: match tokens i in `seq1` to j in `seq2`,
- **I**: skip token i in `seq1`,
- **D**: skip token j in `seq2`.

Some alignments and corresponding trajectories:

- `IMMMDDD`:
```
CATT---
-ATTACA
```
- `DDMMDMI`:
```
--CA-TT
ATTACA-
```
- `MMMMDD`:
```
CATT--
ATTACA
```
- `DDDDMDIII`:
```
----C-ATT
ATTACA---
```

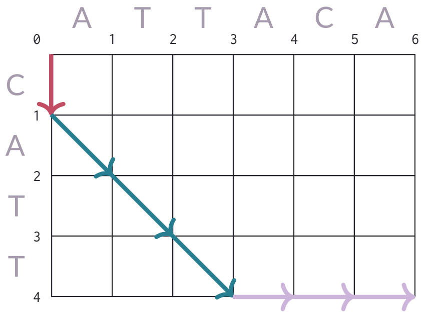
Scoring an alignment



A “default” scoring strategy:

- Get a score of 1 for matching identical characters.
  i.e., if action `M` taken at grid position (i, j) and `seq1[i] == seq2[j]`, add 1 to the score.
- Get a score of -1 for any insertion or deletion.

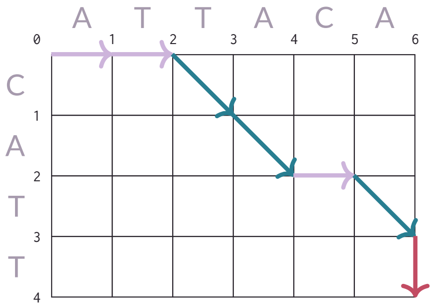
Scoring an alignment



A “default” scoring strategy:

- Get a score of 1 for matching identical characters.
  i.e., if action `M` taken at grid position (i, j) and `seq1[i] == seq2[j]`, add 1 to the score.
- Get a score of -1 for any insertion or deletion.

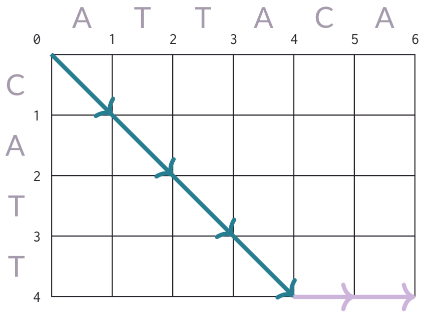
Scoring an alignment



A "default" scoring strategy:

- Get a score of 1 for matching identical characters.
  i.e., if action `M` taken at grid position (i, j) and `seq1[i] == seq2[j]`, add 1 to the score.
- Get a score of -1 for any insertion or deletion.

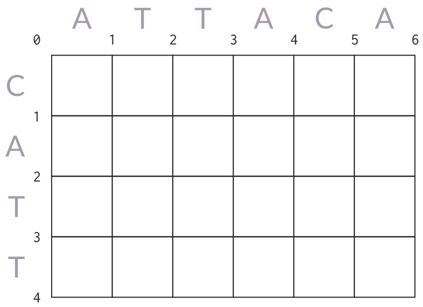
Scoring an alignment



A “default” scoring strategy:

- Get a score of 1 for matching identical characters.
  i.e., if action `M` taken at grid position (i, j) and `seq1[i] == seq2[j]`, add 1 to the score.
- Get a score of -1 for any insertion or deletion.

Parametrized Scoring



let $\boldsymbol{A}$ a score array of shape $(n+1, m+1, 3)$:

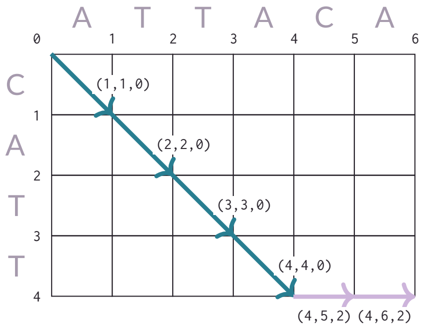
- $a_{i,j,0}$ is the score for Matching token i in `seq1` with token j in `seq2`.
- $a_{i,j,1}$ is the score for an Insertion at (i, j): skipping token i in `seq1` when the cursor is at j in `seq2`.
- $a_{i,j,2}$ is the score for a Deletion at (i, j): skipping token j in `seq2` when the cursor is at i in `seq2`.

note: in these slides, we use zero-indexing into $\boldsymbol{A}$, but one-indexing into the sequences.

We can set the specific values of $\boldsymbol{A}$ to replicate the default scoring from before.

But this parametrized version lets us learn how to score alignments.

Parametrized Scoring



let $\boldsymbol{A}$ a score array of shape $(n+1, m+1, 3)$:

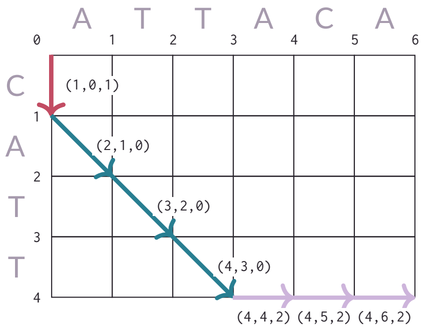
- $a_{i,j,0}$ is the score for `M`atching token i in `seq1` with token j in `seq2`.
- $a_{i,j,1}$ is the score for an `I`nsertion at (i,j): skipping token i in `seq1` when the cursor is at j in `seq2`.
- $a_{i,j,2}$ is the score for a `D`eletion at (i,j): skipping token j in `seq2` when the cursor is at i in `seq2`.

  note: in these slides, we use zero-indexing into $\boldsymbol{A}$, but one-indexing into the sequences.

We can set the specific values of $\boldsymbol{A}$ to replicate the default scoring from before.

But this parametrized version lets us learn how to score alignments.

Parametrized Scoring



let $\boldsymbol{A}$ a score array of shape $(n+1, m+1, 3)$:

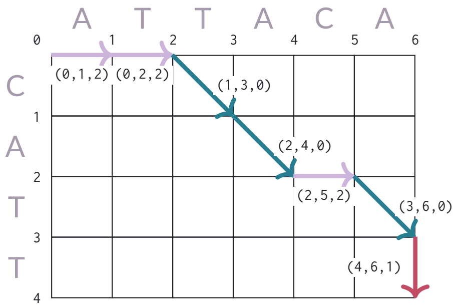
- $a_{i,j,0}$ is the score for Matching token i in `seq1` with token j in `seq2`.
- $a_{i,j,1}$ is the score for an Insertion at (i, j): skipping token i in `seq1` when the cursor is at j in `seq2`.
- $a_{i,j,2}$ is the score for a Deletion at (i, j): skipping token j in `seq2` when the cursor is at i in `seq2`.

  note: in these slides, we use zero-indexing into $\boldsymbol{A}$, but one-indexing into the sequences.

We can set the specific values of $\boldsymbol{A}$ to replicate the default scoring from before.

But this parametrized version lets us learn how to score alignments.

Parametrized Scoring



let $\boldsymbol{A}$ a score array of shape $(n+1, m+1, 3)$:

- $a_{i,j,0}$ is the score for Matching token i in `seq1` with token j in `seq2`.
- $a_{i,j,1}$ is the score for an Insertion at (i,j): skipping token i in `seq1` when the cursor is at j in `seq2`.
- $a_{i,j,2}$ is the score for a Deletion at (i,j): skipping token j in `seq2` when the cursor is at i in `seq2`.

note: in these slides, we use zero-indexing into $\boldsymbol{A}$, but one-indexing into the sequences.

We can set the specific values of $\boldsymbol{A}$ to replicate the default scoring from before.

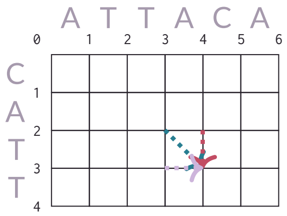
But this parametrized version lets us learn how to score alignments.

Machine Learning for Structured Data

Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

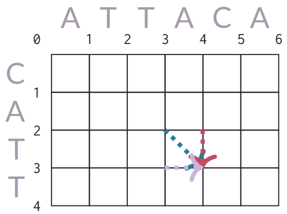
Sequence Alignments

Dynamic Programming for Alignments



Alignments = paths in DAG from $(0, 0)$ to (n, m).

Dynamic Programming for Alignments



F=

0						

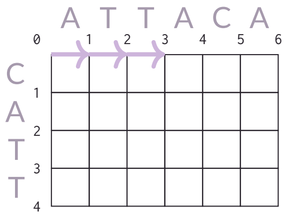
Alignments = paths in DAG from $(0, 0)$ to (n, m).

Computing the max score:
Fill in a table $\boldsymbol{M}$, size $(1 + n, 1 + m)$,
s.t. m_{ij} = the max score up to (i, j).

$$m_{ij} = \begin{cases} m_{i-1,j-1} + a_{i,j,0} \\ m_{i-1,j} + a_{i,j,1} \\ m_{i,j-1} + a_{i,j,2} \end{cases} \quad \text{for any } i > 0, j > 0.$$

What is a topological order?

Dynamic Programming for Alignments



F=

0						

Alignments = paths in DAG from $(0,0)$ to (n,m).

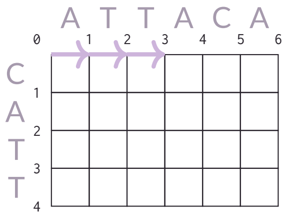
Computing the max score:
Fill in a table $\boldsymbol{M}$, size $(1+n, 1+m)$,
s.t. m_{ij} = the max score up to (i,j).

$$m_{ij} = \begin{cases} m_{i-1,j-1} + a_{i,j,0} \\ m_{i-1,j} + a_{i,j,1} \\ m_{i,j-1} + a_{i,j,2} \end{cases} \quad \text{for any } i > 0, j > 0.$$

What is a topological order?

m_{i0}: only one possible path for any i.

Dynamic Programming for Alignments



F=

0	a_{012}	a_{012} $+a_{022}$	$\cdots$			

Alignments = paths in DAG from $(0, 0)$ to (n, m).

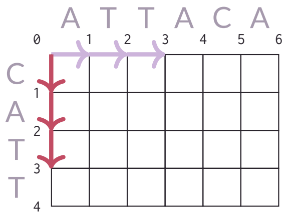
Computing the max score:
Fill in a table $\boldsymbol{M}$, size $(1 + n, 1 + m)$,
s.t. m_{ij} = the max score up to (i, j).

$$m_{ij} = \begin{cases} m_{i-1,j-1} + a_{i,j,0} \\ m_{i-1,j} + a_{i,j,1} \\ m_{i,j-1} + a_{i,j,2} \end{cases} \quad \text{for any } i > 0, j > 0.$$

What is a topological order?

m_{i0}: only one possible path for any i.

Dynamic Programming for Alignments



Alignments = paths in DAG from $(0,0)$ to (n,m).

Computing the max score:
Fill in a table $\boldsymbol{M}$, size $(1+n, 1+m)$,
s.t. m_{ij} = the max score up to (i,j).

$$m_{ij} = \begin{cases} m_{i-1,j-1} + a_{i,j,0} \\ m_{i-1,j} + a_{i,j,1} \\ m_{i,j-1} + a_{i,j,2} \end{cases} \quad \text{for any } i > 0, j > 0.$$

What is a topological order?

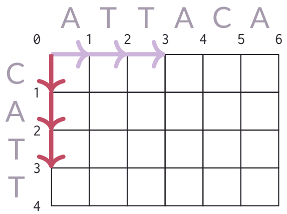
m_{i0}: only one possible path for any i.

m_{0j}: only one possible path for any j.

F=

0	a_{012}	a_{012} $+a_{022}$	$\cdots$			

Dynamic Programming for Alignments



F=

0	a_{012}	a_{012} $+a_{022}$	$\cdots$			
a_{101}						
a_{101} $+a_{201}$						
$\cdots$						

Alignments = paths in DAG from $(0, 0)$ to (n, m).

Computing the max score:
Fill in a table $\boldsymbol{M}$, size $(1 + n, 1 + m)$,
s.t. m_{ij} = the max score up to (i, j).

$$m_{ij} = \begin{cases} m_{i-1,j-1} + a_{i,j,0} \\ m_{i-1,j} + a_{i,j,1} \\ m_{i,j-1} + a_{i,j,2} \end{cases} \quad \text{for any } i > 0, j > 0.$$

What is a topological order?

m_{i0}: only one possible path for any i.

m_{0j}: only one possible path for any j.

History of DP Alignments

Small variants of this algorithm are known by many names and were reinvented many times:

- in biology: Needleman-Wunsch, and (with a small change) Smith-Waterman.
- in compling / information retrieval, Levenshtein / Edit Distance / Wagner-Fischer
- in time series / signal processing: Dynamic Time Warping (DTW)

As far as we know, the first inventor is actually Ukrainian mathematician Taras Vintsiuk, for speech applications.

Viterbi for alignments

```
input: Scores $\boldsymbol{A}$ ($n+1 \times m+1 \times 3$ array), zero-indexed
initialize $\boldsymbol{F}$, same shape as $\boldsymbol{A}$,
$M_{00} = 0, \quad M_{i0} = \sum_{k=1}^{i} a_{k,0,1}, \quad M_{0j} = \sum_{k=1}^{j} a_{0,k,2}.$

*Forward*: compute max. scores recursively
for $i = 1$ to $n$ do
    for $j = 1$ to $m$ do
        $M_{ij} = \max \begin{cases} M_{i-1,j-1} + a_{i,j,0} \\ M_{i-1,j} + a_{i,j,1} \\ M_{i,j-1} + a_{i,j,2} \end{cases} ; \qquad \pi_{ij} = \arg\max \begin{cases} M_{i-1,j-1} + a_{i,j,0} \\ M_{i-1,j} + a_{i,j,1} \\ M_{i,j-1} + a_{i,j,2} \end{cases} ;$
$f^\star = M_{n,m}$

*Backward*: follow backpointers
$i = n, j = m, \boldsymbol{y}^\star = ()$
while $(i,j) \neq (0,0)$ do
    insert $\pi_{ij}$ at the front of $\boldsymbol{y}^\star$,
    decrease $i$, $j$, or both, depending on $\pi_{ij}$

output: The highest-scoring alignment path $\boldsymbol{y}^\star$, and its total score $f^\star$.
```

Forward algorithm for alignments

```
input: Scores $\boldsymbol{A}$ ($n+1 \times m+1 \times 3$ array), zero-indexed
initialize $\boldsymbol{F}$, same shape as $\boldsymbol{A}$,
$F_{00} = 0, \quad F_{i0} = \sum_{k=1}^{i} a_{k,0,1}, \quad F_{0j} = \sum_{k=1}^{j} a_{0,k,2}.$

*Forward*: compute scores recursively
for $i = 1$ to $n$ do
    for $j = 1$ to $m$ do
        $M_{ij} = \log \sum \exp \begin{cases} M_{i-1,j-1} + a_{i,j,0} \\ M_{i-1,j} + a_{i,j,1} \\ M_{i,j-1} + a_{i,j,2} \end{cases} ;$
return $M_{n,m}$
```

Machine Learning for Structured Data

Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

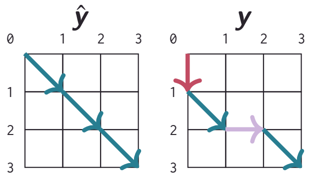
Sequence Alignments

Evaluating Alignments

So far we are representing alignments as sequences of "moves" on a grid.

How to evaluate if we predict $\hat{\boldsymbol{y}}$ = `MMM` when the correct label is $\boldsymbol{y}$ = `IMDM`?

Alignment-level accuracy always an option.
Finer-grained eval?



Evaluating Alignments

So far we are representing alignments as sequences of "moves" on a grid.

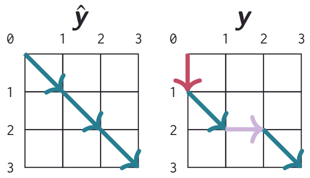
How to evaluate if we predict $\hat{\boldsymbol{y}}$ = MMM
when the correct label is $\boldsymbol{y}$ = IMDM?

Alignment-level accuracy always an option.
Finer-grained eval?

In protein alignment, we care most about getting the aligned indices (i, j) right.

(getting the M–edges right!)

- precision: n. correct M–edges / n. predicted M–edges
- recall: n. correct M–edges / n. true M–edges
- F-score: harmonic average of P and R.



$$\text{indices}(\hat{\boldsymbol{y}}) = \{(1,1),(2,2),(3,3)\},$$
$$\text{indices}(\boldsymbol{y}) = \{(2,1),(3,3)\}.$$

Summary

- Monotonic alignments between two sequences.
- Once again, dynamic programming gives us polynomial-time complexity.
- Algorithm rediscovered many times across many different fields under different names.