Outline

• Matching

• Hungarian algorithm

• DTW
Matching: Formulation 4

Given two sets of elements to be matched

\[ V = \{v_1, v_2, \ldots, v_N\}, \text{ and } V' = \{v'_1, v'_2, \ldots, v'_M\} \]

Find the mapping

\[ f : U \rightarrow U', \text{ where } U \subseteq V \text{ and } U' \subseteq V' \]

\[ f := \{(v, v') : v \in U \subseteq V, v' \in U' \subseteq V'\} \]

Which maximizes the objective function

\[ \hat{f} = \max_f \left[ \sum_{(u, u') \in f} \psi(u, u') \right] \]
Linearization

Linearization by introducing an indicator matrix

\[ X = \begin{bmatrix}
0 & 0 & 1 & \ldots & 0 & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 \\
\vdots & & \ddots & \& \vdots & \vdots \\
0 & 0 & \ldots & 0 & 1 & 0
\end{bmatrix}_{N \times M} \]

\[ x(v, v') = 1, \text{ if } (v_i, v'_i) \text{ matched pair} \]

\[ x(v, v') = 0, \text{ if } (v_i, v'_i) \text{ unmatched pair} \]
Linearization

\[ \Psi = \begin{bmatrix} 
\psi_{11}' & \psi_{12}' & \psi_{13}' & \cdots & \psi_{1M} \\
\psi_{21}' & \psi_{22}' & \psi_{23}' & \cdots & \psi_{2M} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}_{N \times M} \]

\[ \sum_{(v,v') \in f} \psi(v, v') = \sum_{(v,v')} x(v, v') \psi(v, v') = \text{tr}(\Psi^T X) \]
Relaxation

\[
\max_X \text{tr}(\Psi^T X)
\]

subject to:

\[
\forall u \in V, \forall u' \in V', \quad x_{uu'} \in [0, 1]
\]

\[
\forall u, \quad \sum_{u'} x_{uu'} = 1
\]

\[
\forall u', \quad \sum_u x_{uu'} = 1
\]

what is the meaning of this constraint?
Relaxation

$$\max_X \text{tr}(\Psi^T X)$$

subject to:

$$\forall u \in V, \forall u' \in V', x_{uu'} \in [0, 1]$$

$$\forall u, \sum_{u'} x_{uu'} = 1$$

$$\forall u', \sum_u x_{uu'} = 1$$

one-to-one matching
Linear Assignment Problem

$$\max_X \tr(\Psi^T X)$$

subject to:

$$\forall u \in V, \forall u' \in V', x_{uu'} \in [0, 1]$$

$$\forall u, \sum_{u'} x_{uu'} = 1$$

$$\forall u', \sum_u x_{uu'} = 1$$

optimal solution exists for the balanced problem when $|V| = |V'|$
Linear Assignment Problem

\[
\max_X \text{tr}(\Psi^T X)
\]

subject to:

\[
\forall u \in V, \forall u' \in V', \ x_{uu'} \in [0, 1]
\]

\[
\forall u, \sum_{u'} x_{uu'} = 1
\]

\[
\forall u', \sum_u x_{uu'} = 1
\]

The Hungarian algorithm may give false solutions for unbalanced problems when |V| \neq |V'|
The Hungarian Algorithm

\[ A = -\Psi \]

\[
\min_X \text{tr}(A^T X)
\]

subject to:

\[ \forall u \in V, \forall u' \in V', \ x_{uu'} \in [0, 1] \]

\[ \forall u, \sum_{u'} x_{uu'} = 1 \]

\[ \forall u', \sum_u x_{uu'} = 1 \]
The Hungarian Algorithm

1. Find the min element along each row (column) of $A$, and subtract it from all elements in the respective rows (columns). Replace $A$ with the resulting matrix.

2. Cross out the minimum number of rows and columns in $A$ to cover all zero elements of $A$.

3. If min($N,M$) rows and columns of $A$ are crossed out, then go to step 5.

4. Otherwise, find the minimal entry of $A$ that is not crossed out. Add this entry to all elements that are doubly crossed out (by both a horizontal and vertical line), and subtract it from all entries of $A$ that are not crossed out. Return to step 2 with the new matrix.

5. Solutions are zero elements of $A$. Go first for the zero element which is unique in its row and column. Then, delete that row and column from $A$. Repeat until you delete all rows or columns from $A$. 
Example -- The Hungarian Algorithm

given a cost matrix

\[
A = \begin{bmatrix}
14 & 5 & 8 & 7 \\
2 & 12 & 6 & 5 \\
7 & 8 & 3 & 9 \\
2 & 4 & 6 & 10
\end{bmatrix}
\]

step 1: find minimums and subtract

\[
A = \begin{bmatrix}
9 & 0 & 3 & 2 \\
0 & 10 & 4 & 3 \\
4 & 5 & 0 & 6 \\
0 & 2 & 4 & 8
\end{bmatrix}
\]
go for the unique solution first

step 5: \[ A = \begin{bmatrix} 10 & 0 & 3 & 0 \\ 0 & 9 & 3 & 0 \\ 0 & 1 & 3 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 5 & 5 & 0 & 4 \\ 0 & 1 & 3 & 5 \end{bmatrix} \]

\[ x_{33} = 1 \]
Example -- The Hungarian Algorithm -- Solution

go for the unique solution first

step 5: \( A = \begin{bmatrix}
10 & 0 & 3 & 0 \\
0 & 9 & 3 & 0 \\
5 & 5 & 0 & 4 \\
\rightarrow & 0 & 1 & 3 & 5
\end{bmatrix} \)

\[ x_{33} = 1 \]

\[ x_{41} = 1 \]
Example -- The Hungarian Algorithm -- Solution

Go for the unique solution first

\[ A = \begin{bmatrix}
10 & 0 & 3 & 0 \\
0 & 9 & 3 & 0 \\
5 & 5 & 0 & 4 \\
0 & 1 & 3 & 5
\end{bmatrix} \]

Step 5:

\[ x_{33} = 1 \]

\[ x_{41} = 1 \]

\[ x_{12} = 1 \]
Example -- The Hungarian Algorithm -- Solution

go for the unique solution first

step 5: \[ A = \begin{bmatrix} 10 & 0 & 3 & 0 \\
0 & 9 & 3 & 0 \\
5 & 5 & 0 & 4 \\
0 & 1 & 3 & 5 \end{bmatrix} \]

\[ x_{33} = 1 \]

\[ x_{41} = 1 \]

\[ x_{12} = 1 \]

\[ x_{24} = 1 \]
Dynamic Time Warping
Alignment of Two Sequences of Points

arrows show the points of alignment
Alignment of Two Sequences of Points

Cost matrix
Building a Cost Matrix of Pairs of Points

\[ c_{ij} = \| x_i - y_j \|, \quad i = 1: N, \quad j = 1: M \]

goal: find the min-cost path
Building the Accumulated Cost Matrix

Cost matrix \( c_{ij} \) \( \rightarrow \) Accumulated cost matrix \( D(i, j) \)

- \( c_{ij} \): element of the cost matrix
- \( D(i, j) \): element of the accumulated cost matrix
Building the Accumulated Cost Matrix

1. First row:

\[ D(1, j) = \sum_{k=1}^{j} c_{1k}, \quad j = 1 : M \]
Building the Accumulated Cost Matrix

1. First row: 
   \[ D(1, j) = \sum_{k=1}^{j} c_{1k}, \quad j = 1 : M \]

2. First column: 
   \[ D(i, 1) = \sum_{k=1}^{i} c_{k1}, \quad i = 1 : N \]
3. Compute:

\[
D(i, j) = c_{ij} + \min\{D(i - 1, j - 1), D(i, j - 1), D(i - 1, j)\}
\]
Building the Accumulated Cost Matrix

4. Find min in the last row, and then back-track the path
Building the Accumulated Cost Matrix

4. Find min in the last row, and then back-track the path
Dynamic Time Warping

```c
int DTWDistance(char s[1..n], char t[1..m]) {
    int DTW[0..n, 0..m]
    int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity;

    for i := 1 to n
        DTW[i, 0] := infinity;

    DTW[0, 0] := 0;

    for i := 1 to n
        for j := 1 to m

            cost := d(s[i], t[j]);

            DTW[i, j] := cost + minimum(DTW[i-1, j], // insertion
                                          DTW[i, j-1],  // deletion
                                          DTW[i-1, j-1]); // match

    return DTW[n, m];
}
```
Cyclic Dynamic Time Warping -- Region Boundaries

DTW cannot be directly used for cyclic sequences

different CDTW algorithms