《多模态机器学习》

第六章 多模态对齐

黄文炳
中国人民大学高瓴人工智能学院
hwenbing@126.com

2024年秋季

课程提纲

单模态表示

- 视觉模态
- 文本模态
- 三维点云
- 动作模态

基本概念

- 神经网络及其优化

经典多模态机器学习

- 多模态表示
- 多模态对齐
- 多模态推理
- 多模态生成
- 多模态迁移

通用多模态机器学习

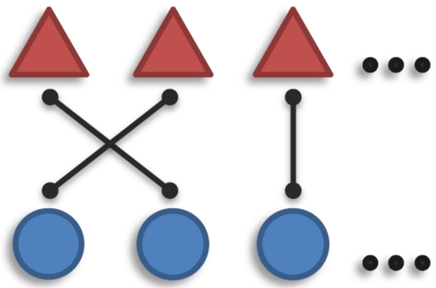
- 通用多模态（大）模型
- 多模态预训练
- 多模态典型应用

Task 2: Alignment

Definition: Identifying and modeling cross-modal connections between all elements of multiple modalities, building from the data structure

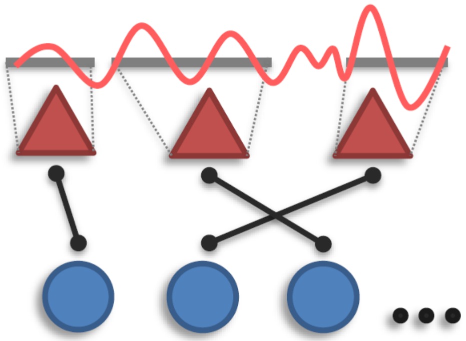
Sub-challenges:

Discrete Alignment



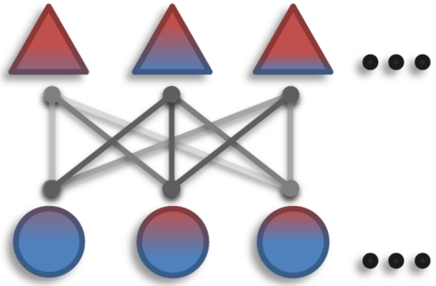
Discrete elements and connections

Continuous Alignment



Segmentation and continuous warping

Contextualized Representation



Alignment + representation

内容提纲

① Discrete alignment

  ① Local alignment

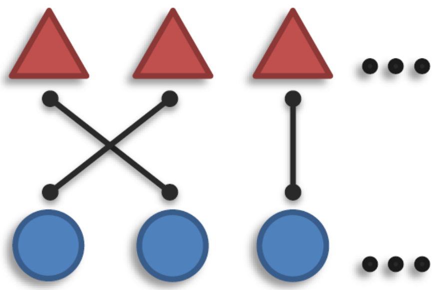
  ② Global alignment

② Continuous alignment

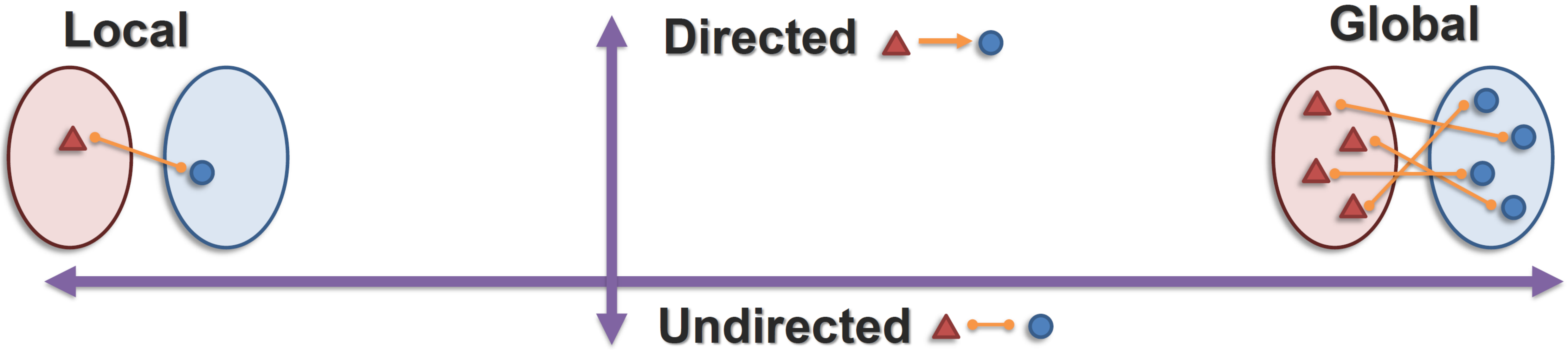
  ① Continuous warping

  ② Discretization and segmentation

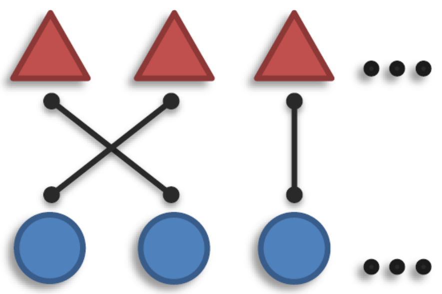
Sub-Challenge 2a: Discrete Alignment



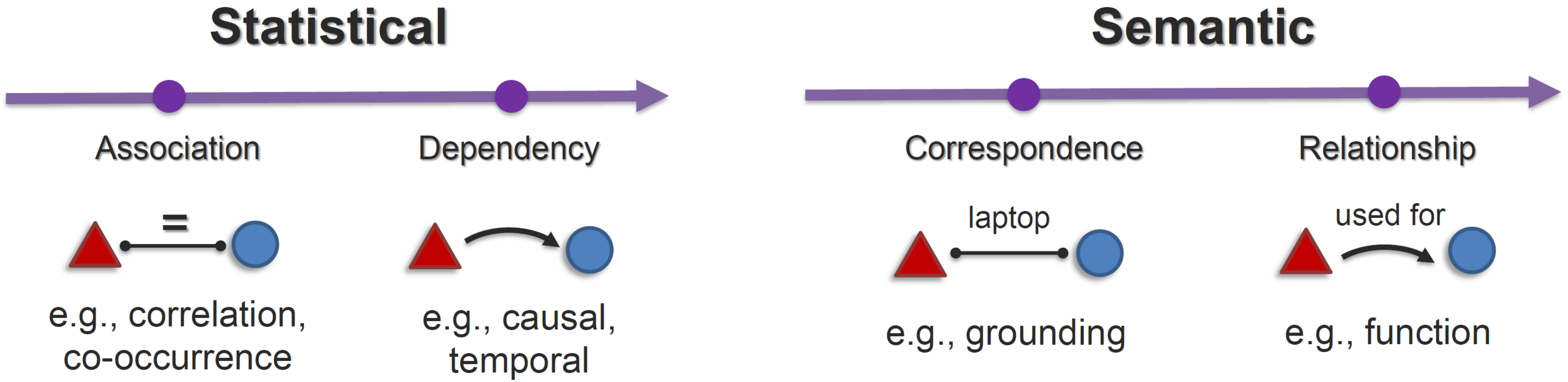
Definition: Identify and model connections between elements of multiple modalities



Connections

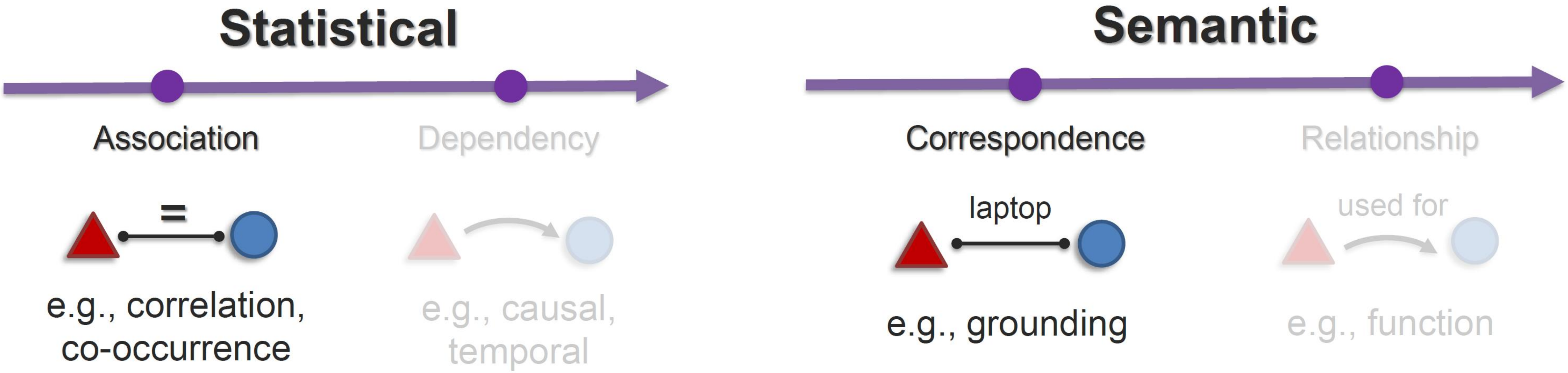


Why should 2 elements be connected?

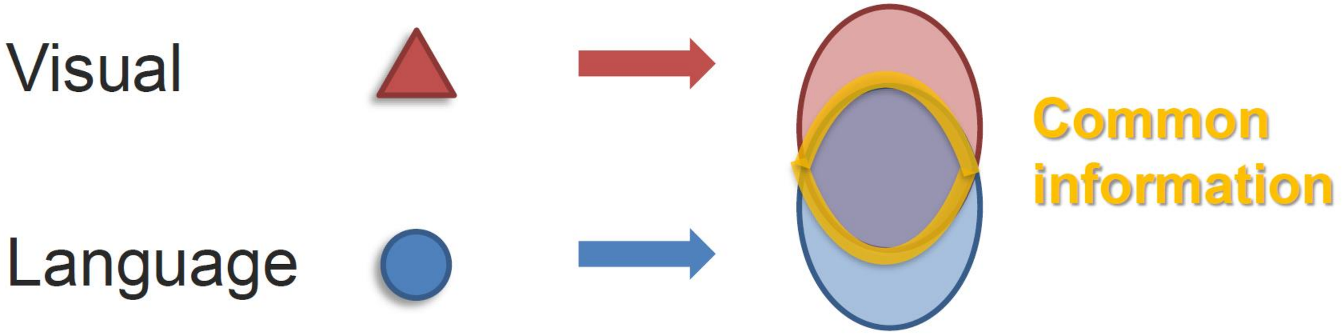


Language Grounding

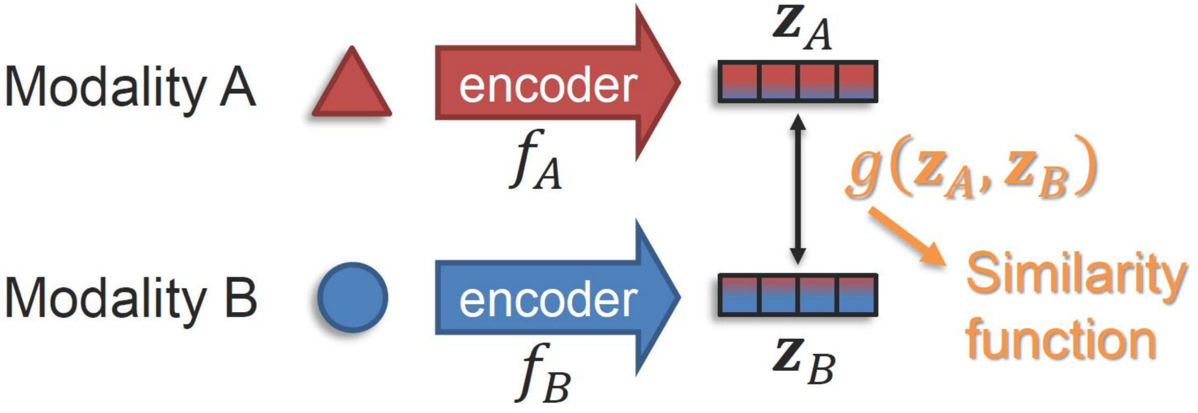
Definition: Tying language (words, phrases,…) to non-linguistic elements, such as the visual world (objects, people, …)



Local Alignment – Coordinated Representations

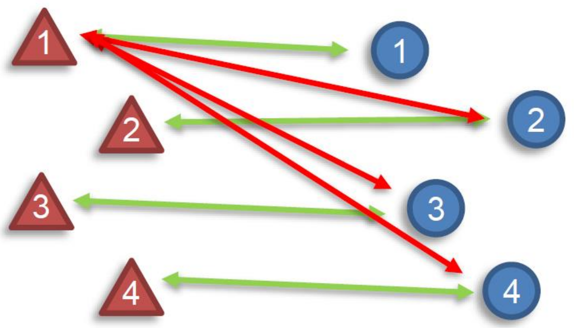


Learning coordinated representations:

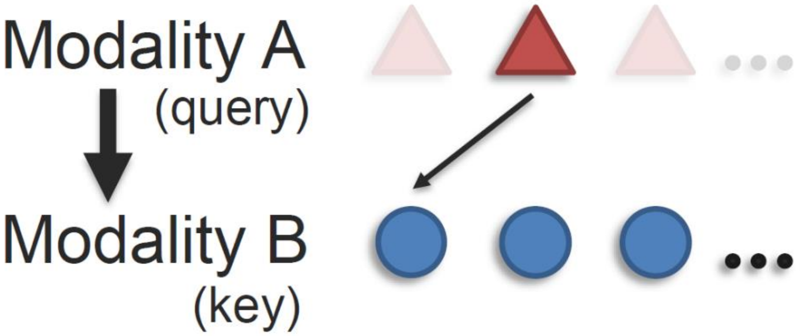


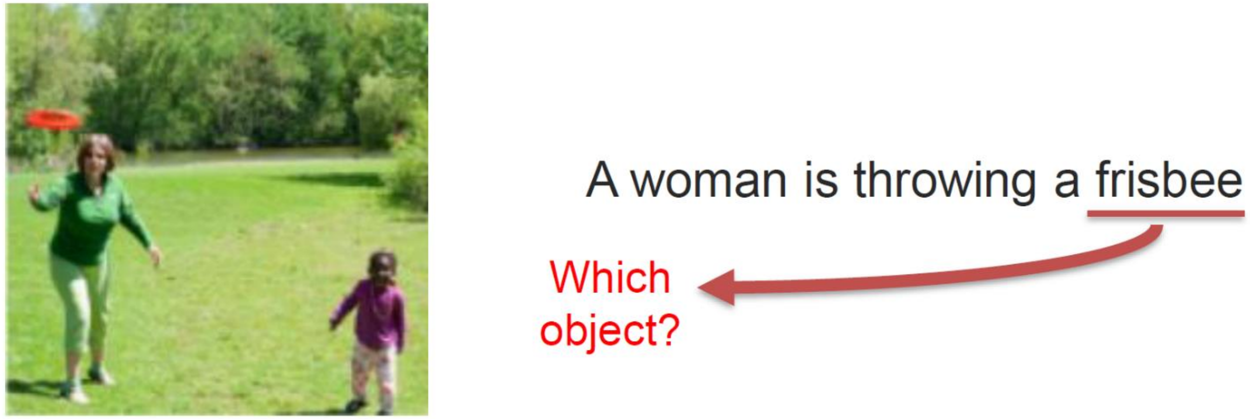
or contrastive learning

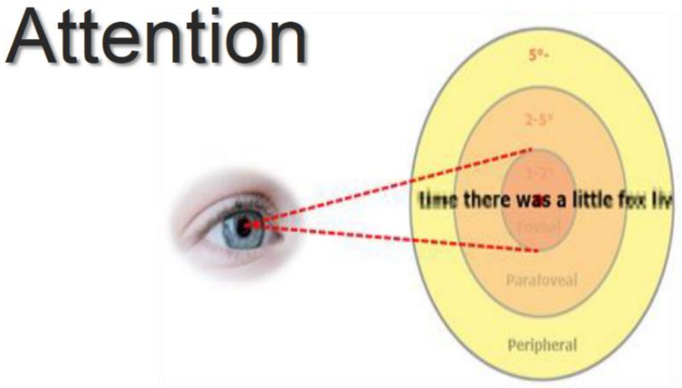
Supervision: Paired data



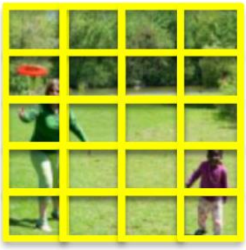
Directed Alignment





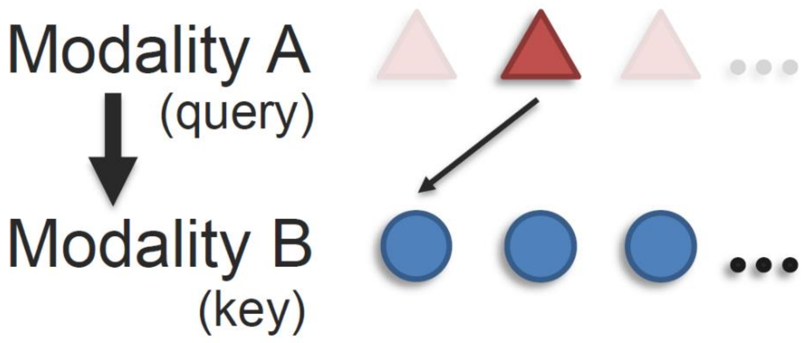


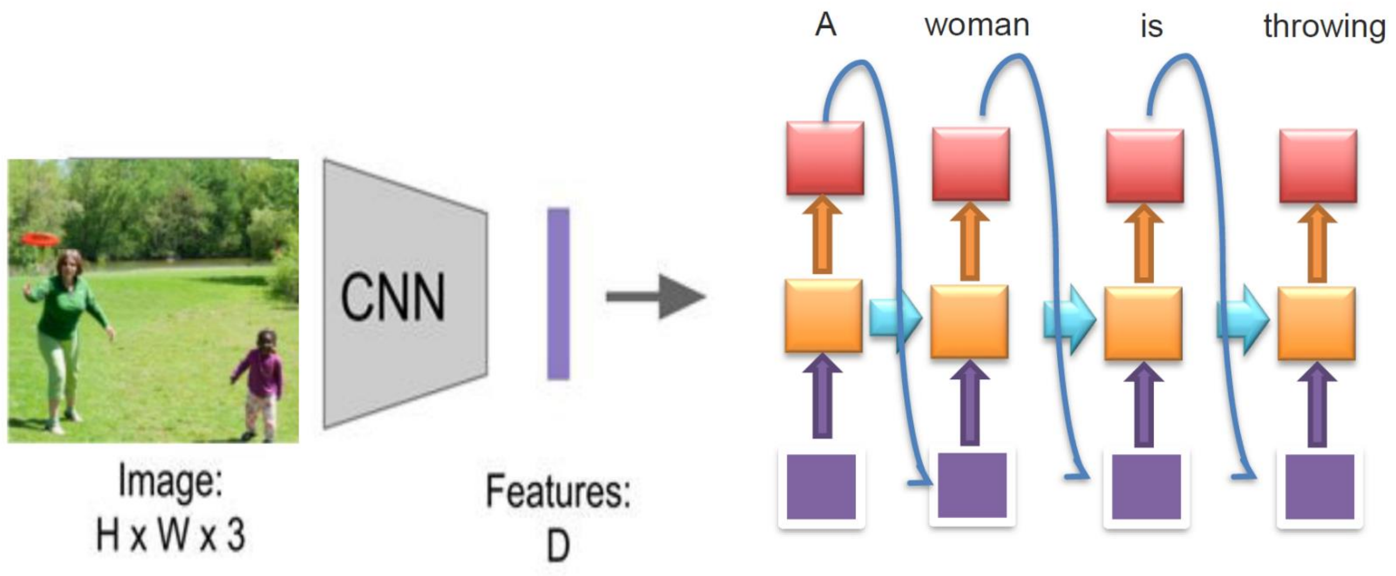
1 Soft attention



2 Hard attention

Cross-modal Interactions

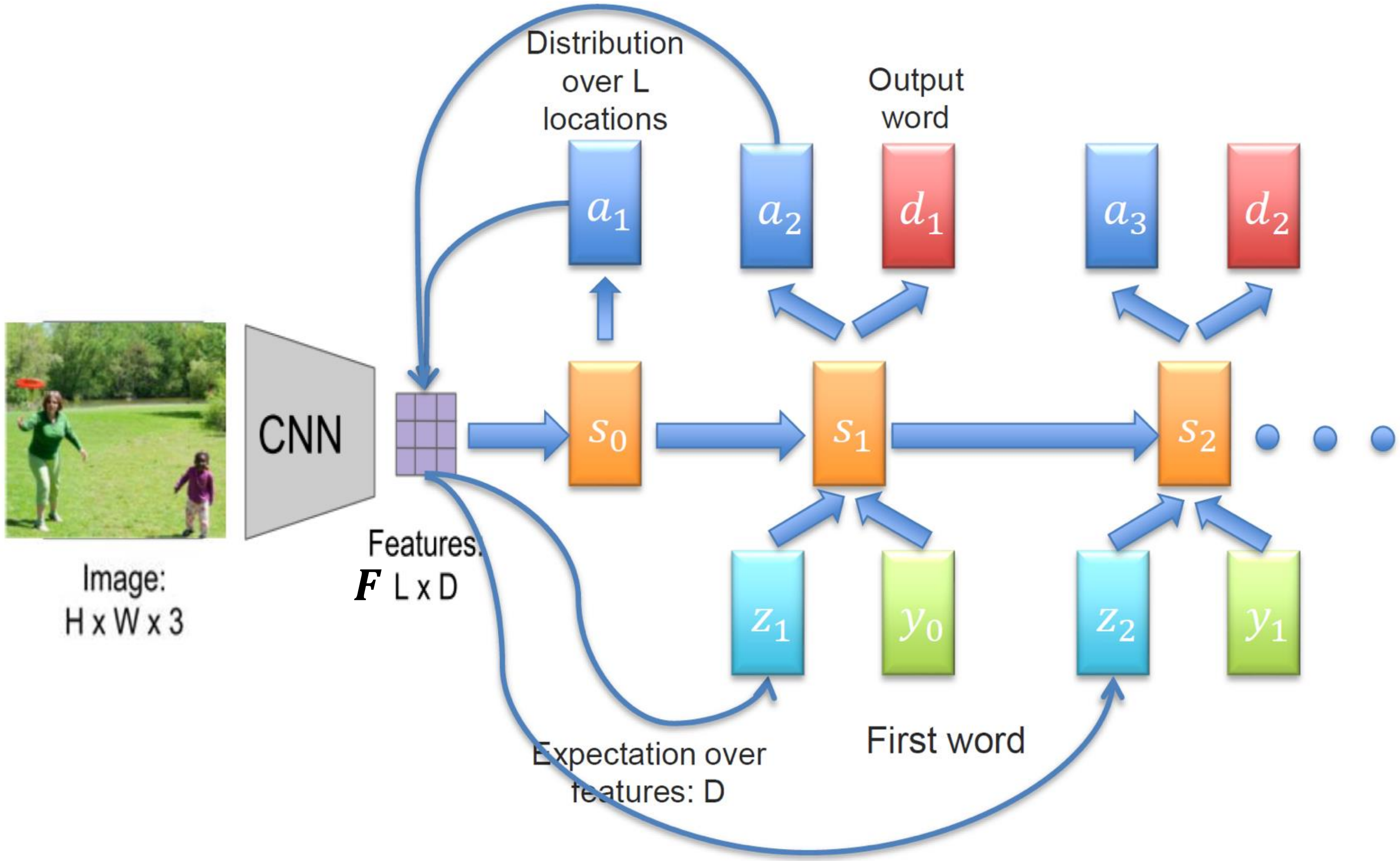




Should we always use the final layer of the CNN for all generated words?

Arevalo et al., Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015. Citations: 12828

Directed Alignment – Image Captioning



Attention Gates

Before:

$$p(y_i | y_1, \dots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, \boldsymbol{s}_i, \boldsymbol{z}),$$

where $\boldsymbol{z} = \boldsymbol{h}_T$, last encoder state and $\boldsymbol{s}_i$ is the current state of the decoder

Now:

$$p(y_i | y_1, \dots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, \boldsymbol{s}_i, \boldsymbol{z}_i)$$

Have an attention "gate"

- A different context $\boldsymbol{z}_i$ used at each time step!
- $\boldsymbol{z}_i = \sum_{j=1}^{L} a_{ij} \boldsymbol{f}_j$

α_{ij} is the (scalar) attention for word i at image position j

Attention Gates

So how do we determine a_{ij}?

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$ => softmax, making sure they sum to 1

where:

$$e_{ij} = \boldsymbol{v}^{\top} \sigma(\boldsymbol{W} \boldsymbol{s}_{i-1} + \boldsymbol{U} \boldsymbol{f}_j)$$

a feedforward network that can tell us how important the current encoding is

$\boldsymbol{v}, \boldsymbol{W}, \boldsymbol{U}$ – learnable weights

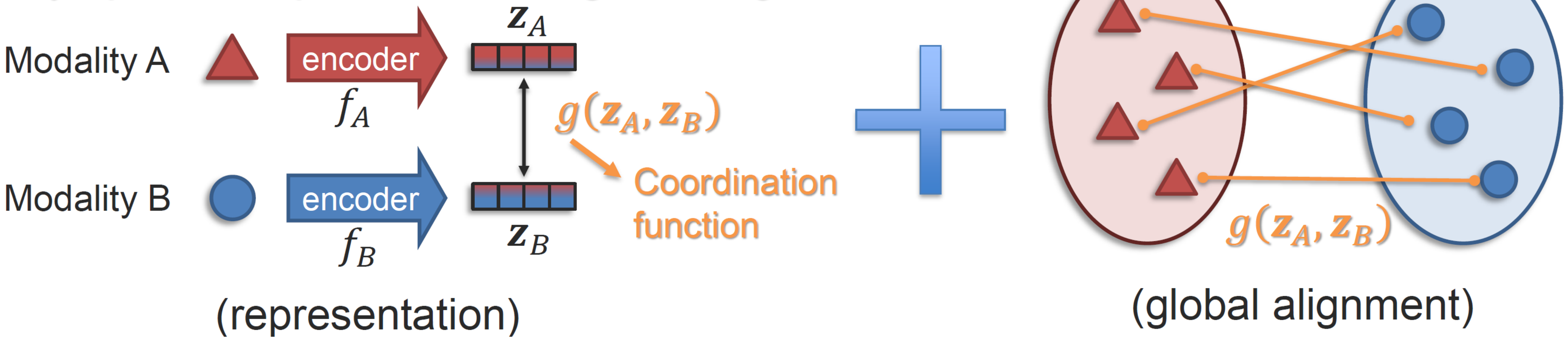
$$\boldsymbol{z}_i = \sum_{j=1}^{L} a_{ij} \boldsymbol{f}_j$$

expectation of the context (a fancy way to say it's a weighted average)

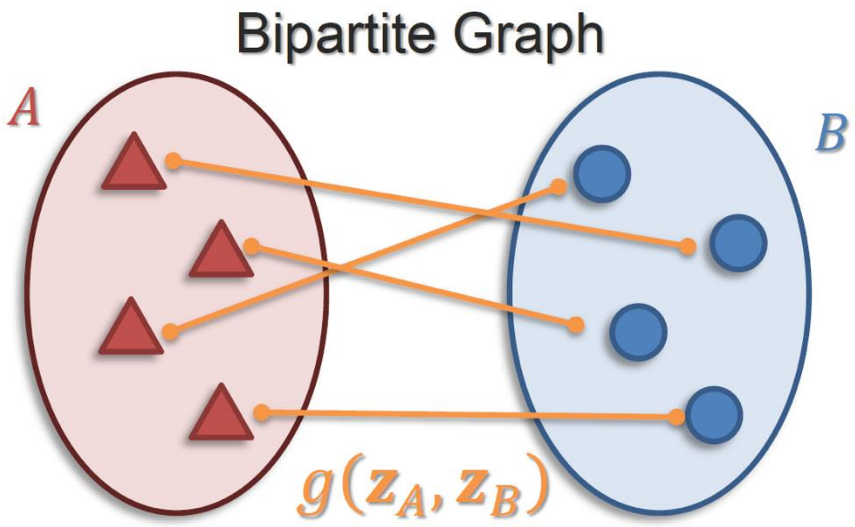
Example – Image Captioning

Global Alignment

Jointly optimize representation + global alignment:



Global Alignment



Initial assumptions:

- Same number of elements in A and B modalities
- 1-to-1 "hard" alignment between elements
- All elements assigned (aka "perfect matching")

➡ How to solve?

Naive solution: check all assignments

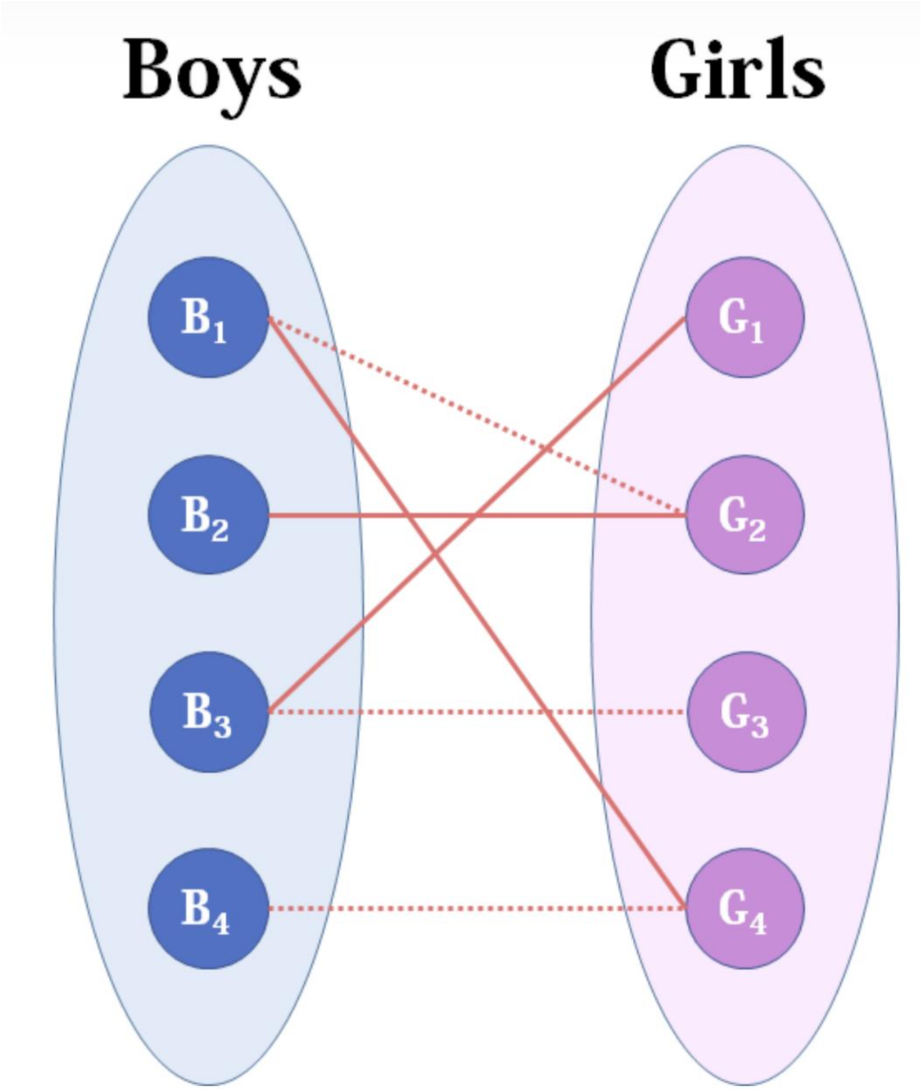
Assignment: $f: A \rightarrow B$
(vector of indices)

Similarity weights: $w_{i,f(i)} = g(\mathbf{z}_A^i, \mathbf{z}_B^{f(i)})$

Maximize: $\max_{f \in S_n} \sum_{i=1}^{N} w_{i,f(i)}$

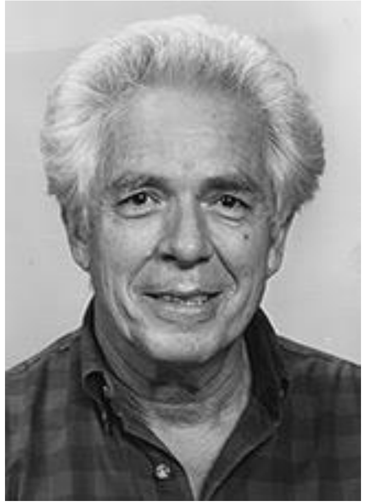
Hungarian algorithm

“红娘算法”



Hungarian algorithm

The Hungarian method, known also as the Kuhn–Munkres algorithm or Munkres assignment algorithm

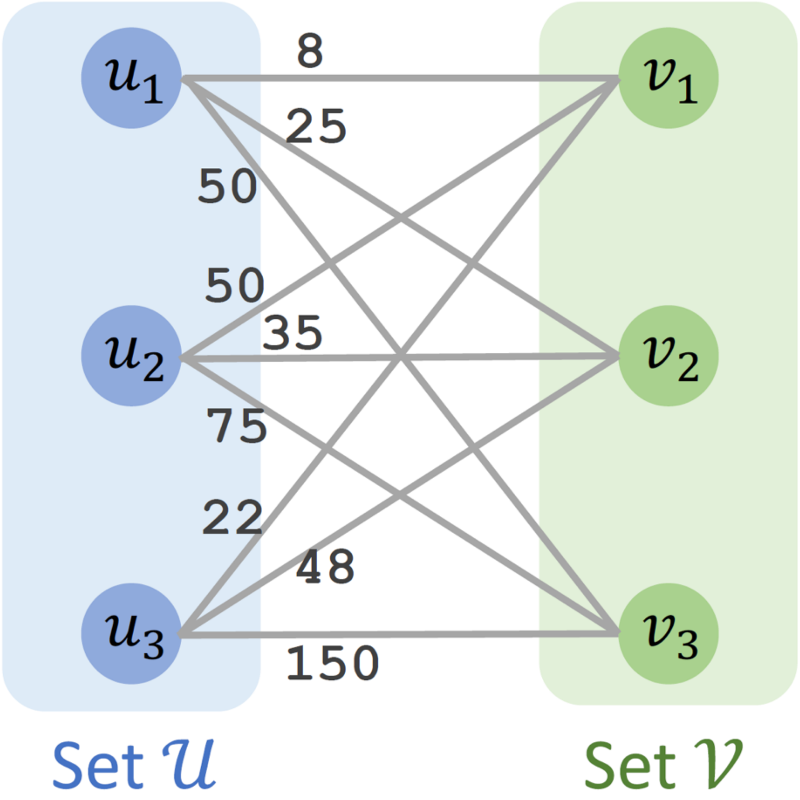


Harold William Kuhn
(July 29, 1925 – July 2, 2014)

Dénes Kőnig
(September 21, 1884 – October 19, 1944)

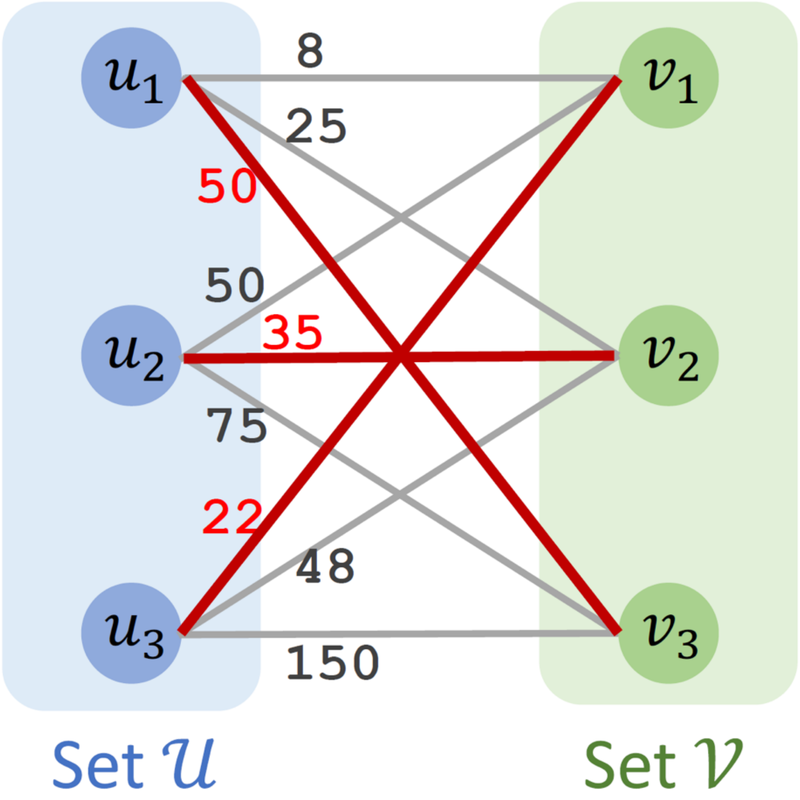
Jenő Elek Egerváry
(April 16, 1891 – November 30, 1958)

Minimum-Weight Bipartite Matching



	v_1	v_2	v_3
u_1	8	25	50
u_2	50	35	75
u_3	22	48	150

Minimum-Weight Bipartite Matching



	v_1	v_2	v_3
u_1	8	25	50
u_2	50	35	75
u_3	22	48	150

The minimum sum of weight is $50 + 35 + 22 = 107$.

Minimum-Weight Bipartite Matching

Subtract Row Minima

	v_1	v_2	v_3
u_1	8	25	50
u_2	50	35	75
u_3	22	48	150

Minimum-Weight Bipartite Matching

Subtract Row Minima

	v_1	v_2	v_3
u_1	8	25	50
u_2	50	35	75
u_3	22	48	150

Minimum-Weight Bipartite Matching

Subtract Row Minima

	v_1	v_2	v_3
u_1	8 −8	25 −8	50 −8
u_2	50 −35	35 −35	75 −35
u_3	22 −22	48 −22	150 −22

Minimum-Weight Bipartite Matching

Subtract Row Minima

Now, the row minima are zeros.

	v_1	v_2	v_3
u_1	0	17	42
u_2	15	0	40
u_3	0	26	128

Minimum-Weight Bipartite Matching

Subtract Column Minima

	v_1	v_2	v_3
u_1	0	17	42
u_2	15	0	40
u_3	0	26	128

Minimum-Weight Bipartite Matching

Subtract Column Minima

	v_1	v_2	v_3
u_1	0	17	42
u_2	15	0	40
u_3	0	26	128

Minimum-Weight Bipartite Matching

Subtract Column Minima

	v_1	v_2	v_3
u_1	0 −0	17 −0	42 −40
u_2	15 −0	0 −0	40 −40
u_3	0 −0	26 −0	128 −40

Minimum-Weight Bipartite Matching

Subtract Column Minima

Now, the column minima are zeros.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	~~0~~	~~17~~	~~2~~
u_2	~~15~~	~~0~~	~~0~~
u_3	~~0~~	~~26~~	~~88~~

Minimum-Weight Bipartite Matching

Iteration 1A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Not optimal!

Minimum-Weight Bipartite Matching

Iteration 1A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1B

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

- If n lines are required, the algorithm stops.
- If less than n lines are required, then continue with Step C.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1B

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

First, find the smallest element (denote k) that is not covered by a line.

	v_1	v_2	v_3
u_1	0	17	2
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

First, find the smallest element (denote k) that is not covered by a line.

	v_1	v_2	v_3
u_1	0	17	2 =k
u_2	15	0	0
u_3	0	26	88

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

Second, subtract k from all uncovered elements.

	v_1	v_2	v_3
u_1	0	17 -2	2 -2
u_2	15	0	0
u_3	0	26 -2	88 -2

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

Second, subtract k from all uncovered elements.

	v_1	v_2	v_3
u_1	0	15	0
u_2	15	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

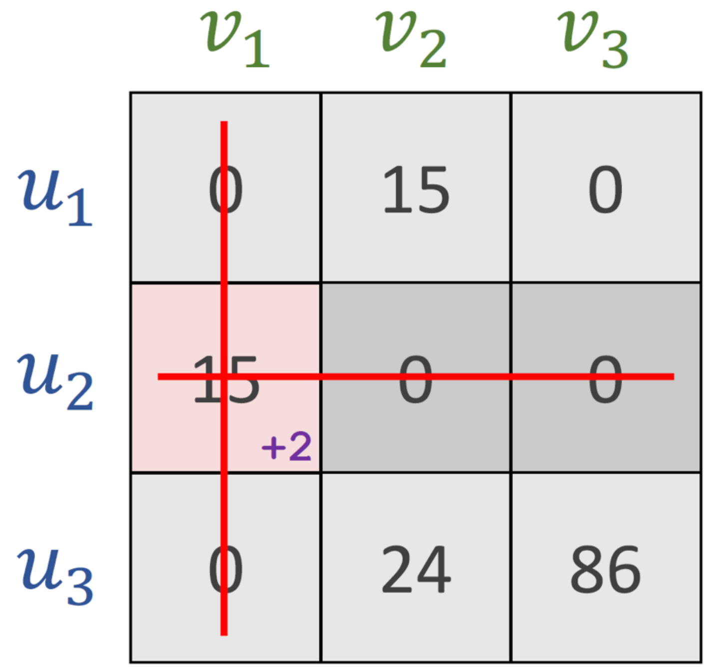
Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.



Third, add k to all the elements that are covered twice.

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

Second, subtract k from all uncovered elements.

	v_1	v_2	v_3
u_1	0	15	0
u_2	15	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

Third, add k to all the elements that are covered twice.

	v_1	v_2	v_3
u_1	0	15	0
u_2	15 +2	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Iteration 1C

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

Third, add k to all the elements that are covered twice.

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Iteration 2

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Iteration 2A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Iteration 2A

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. Decide whether to stop.

C. Create additional zeros.

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

At least 3 lines are needed.

Minimum-Weight Bipartite Matching

Iteration 2B

Repeat the followings:

A. Cover all the zeros with a minimum number of lines.

B. **Decide whether to stop.**

C. Create additional zeros.

If n lines are required, the algorithm stops.

The algorithm stops.

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Output the matching

	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

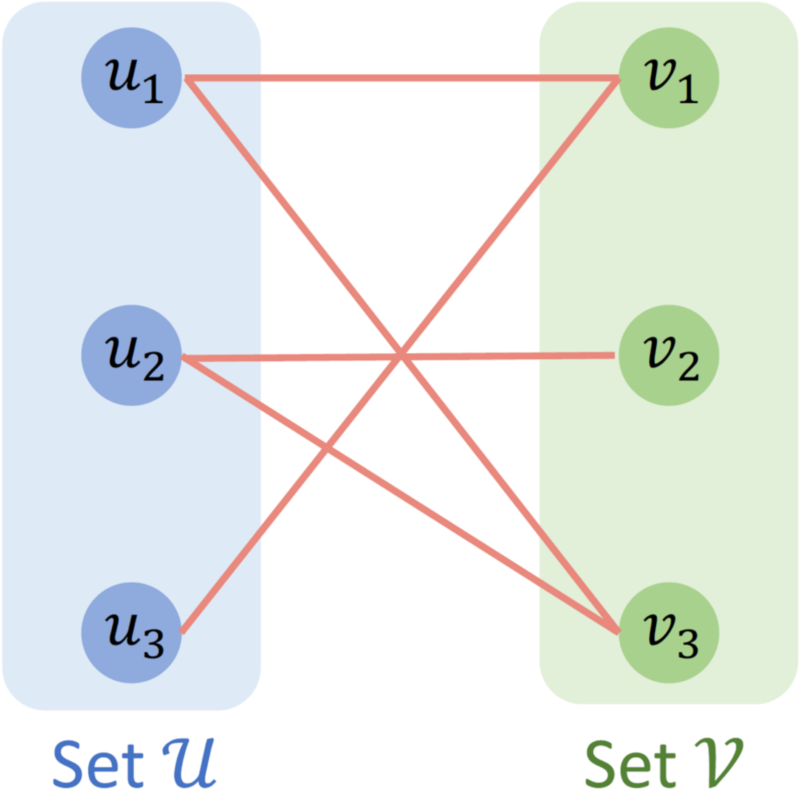
Output the matching

	v_1	v_2	v_3
u_1	**0**	15	**0**
u_2	17	**0**	**0**
u_3	**0**	24	86

- Choose a matching among the zeros.
- Think of the zeros as edges.

Minimum-Weight Bipartite Matching

Output the matching

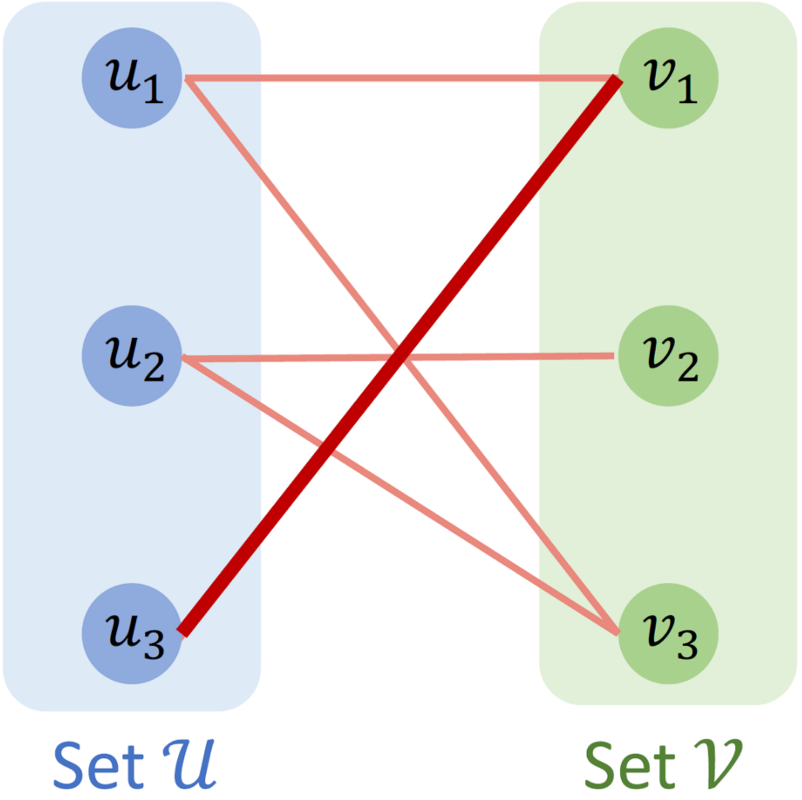


	v_1	v_2	v_3
u_1	**0**	15	**0**
u_2	17	**0**	**0**
u_3	**0**	24	86

- Choose a matching among the zeros.
- Think of the zeros as edges.

Minimum-Weight Bipartite Matching

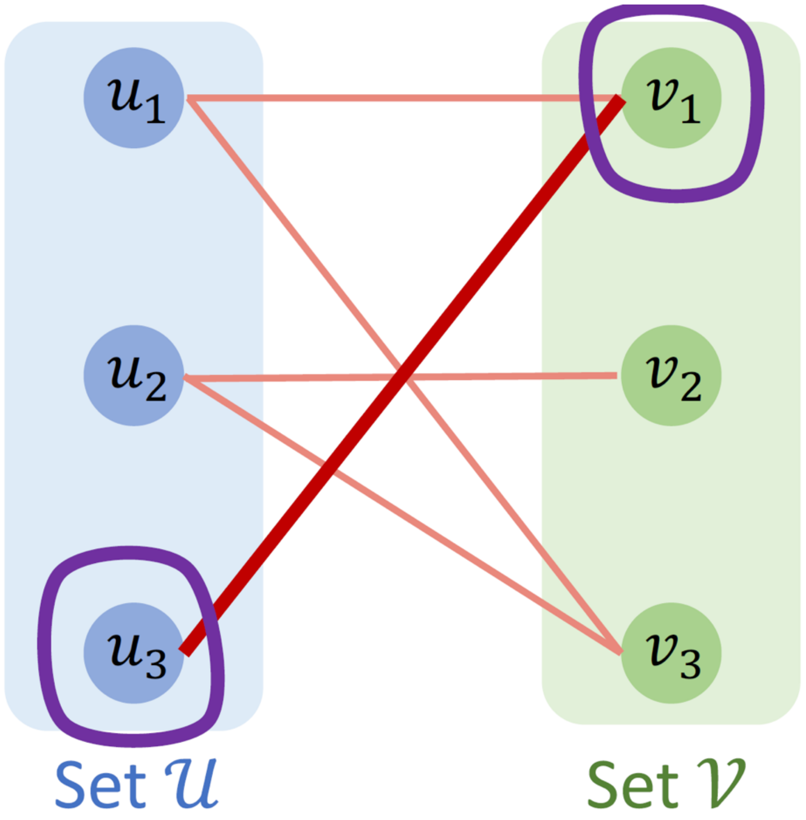
Output the matching



	v_1	v_2	v_3
u_1	**0**	15	**0**
u_2	17	**0**	**0**
u_3	**0**	24	86

Minimum-Weight Bipartite Matching

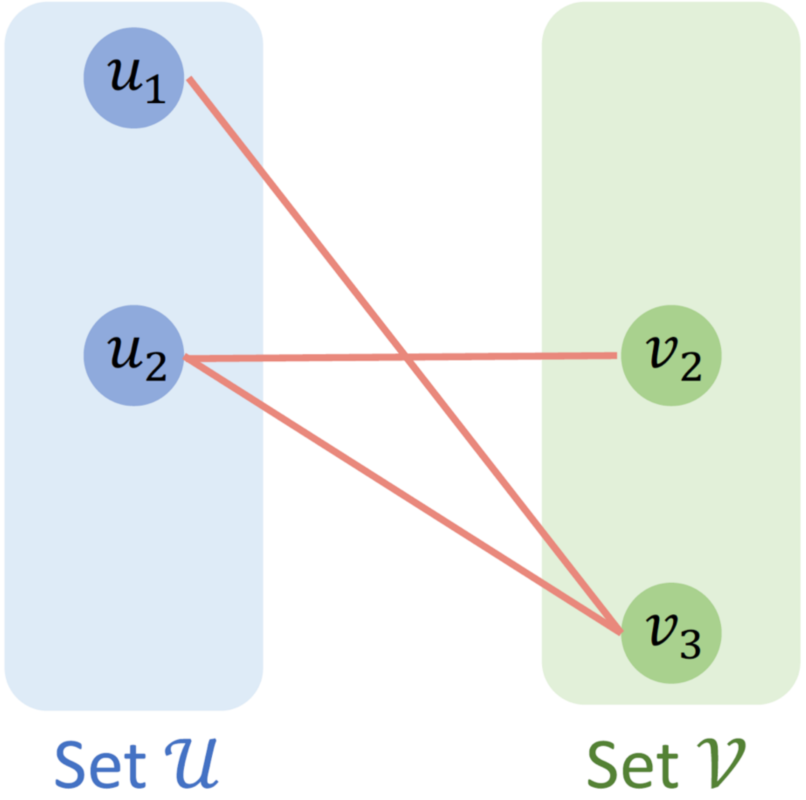
Output the matching



	v_1	v_2	v_3
u_1	**0**	15	**0**
u_2	17	**0**	**0**
u_3	**0**	24	86

Minimum-Weight Bipartite Matching

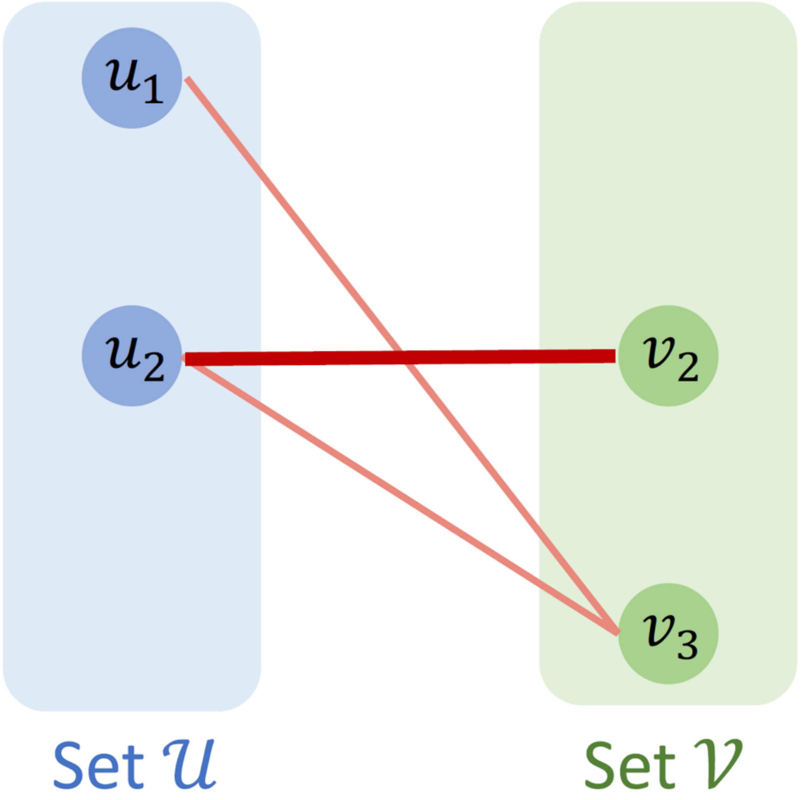
Output the matching



	v_1	v_2	v_3
u_1	0	15	**0**
u_2	17	**0**	**0**
u_3	0	24	86

Minimum-Weight Bipartite Matching

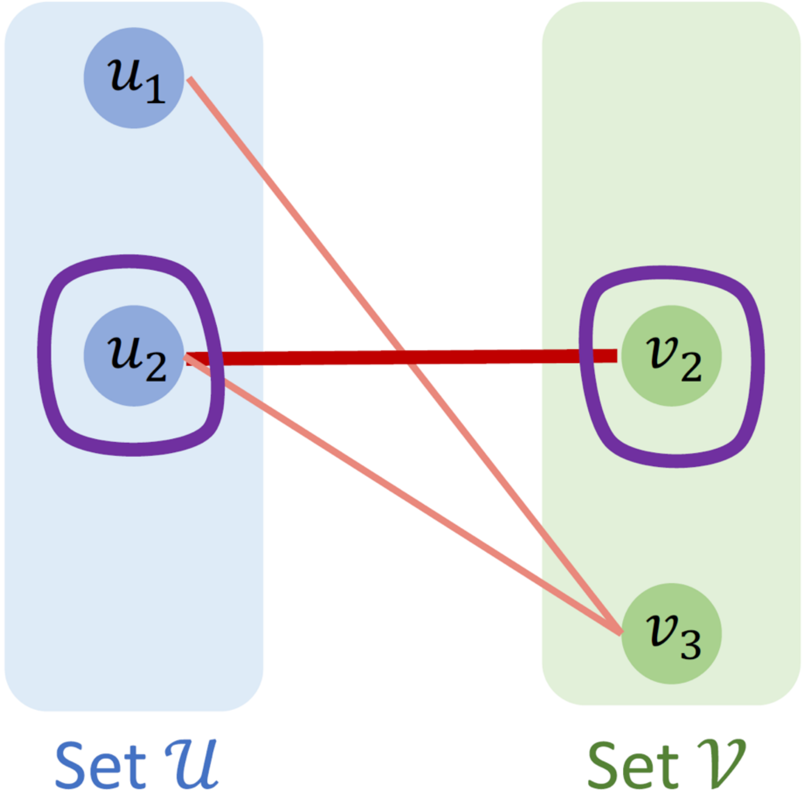
Output the matching



	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

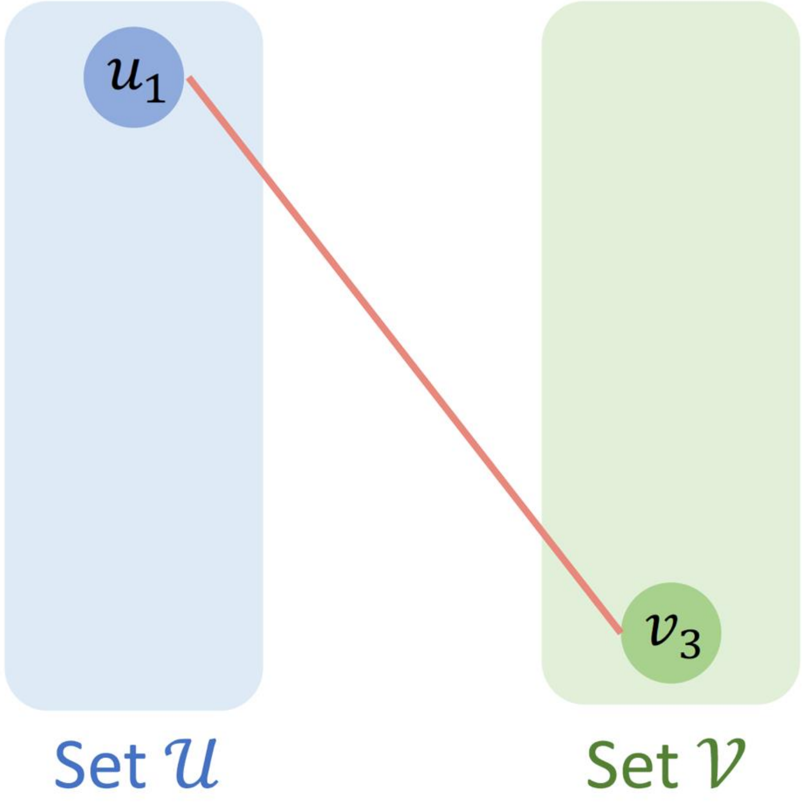
Output the matching



	v_1	v_2	v_3
u_1	0	15	0
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

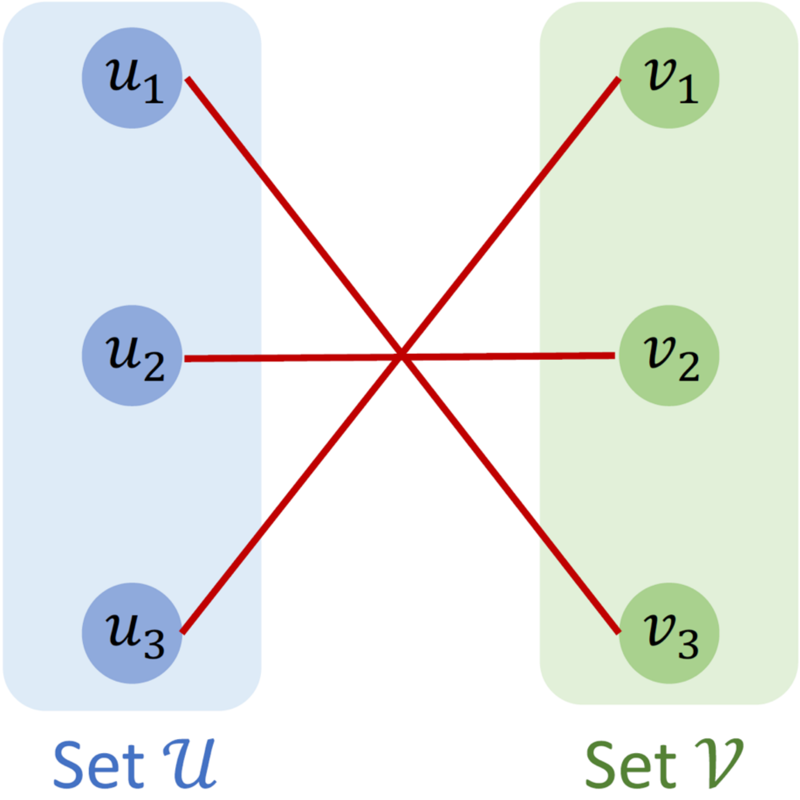
Output the matching



	v_1	v_2	v_3
u_1	0	15	**0**
u_2	17	0	0
u_3	0	24	86

Minimum-Weight Bipartite Matching

Output the matching

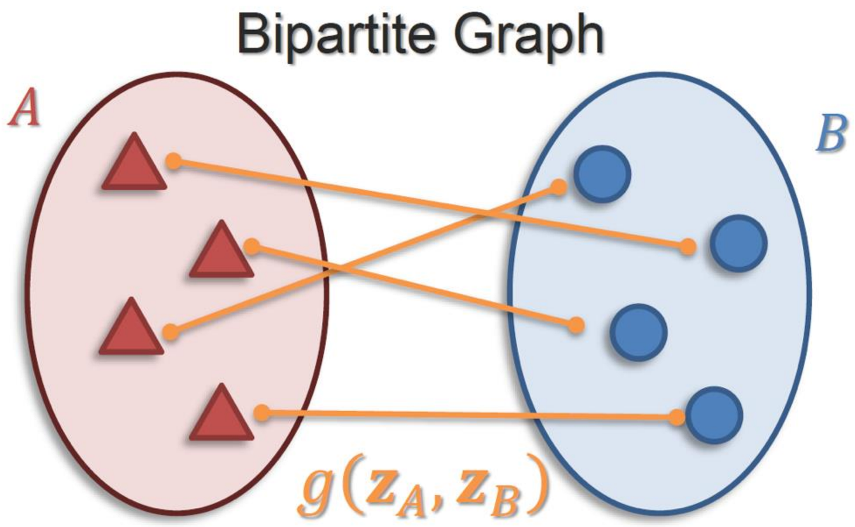


	v_1	v_2	v_3
u_1	**0**	15	**0**
u_2	17	**0**	**0**
u_3	**0**	24	86

The matching is

$$\mathcal{S} = \{(u_3, v_1),\ (u_1, v_3),\ (u_2, v_2)\}.$$

Global Alignment



Initial assumptions:

- Same number of elements in A and B modalities
- 1-to-1 "hard" alignment between elements
- All elements assigned (aka "perfect matching")

How to solve?

Naive solution: check all assignments

Assignment: $f: A \rightarrow B$
(vector of indices)

Similarity weights: $w_{i,f(i)} = g(\mathbf{z}_A^i, \mathbf{z}_B^{f(i)})$

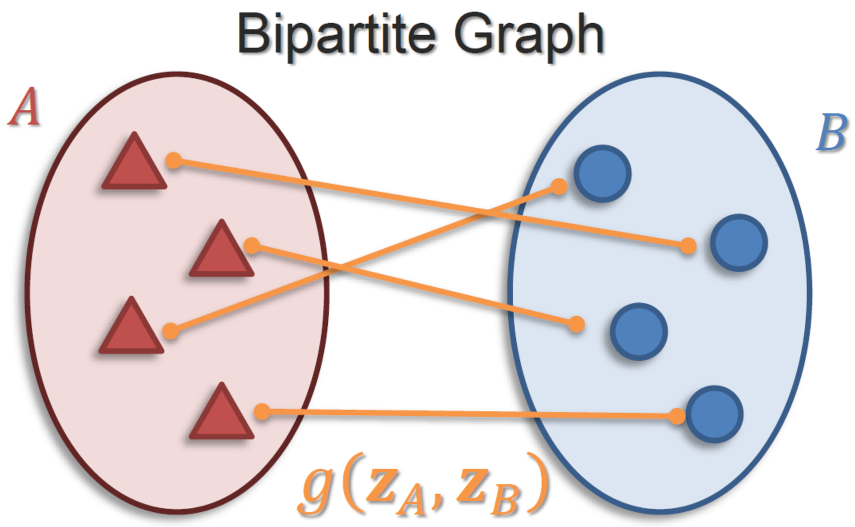
Maximize: $\max_{f \in S_n} \sum_{i=1}^{N} w_{i,f(i)}$

Hungarian algorithm

Why it works?

Complexity?

Global Alignment



Initial assumptions:

- Same number of elements in A and B modalities
- 1-to-1 "hard" alignment between elements
- All elements assigned (aka "perfect matching")

How to solve?

Naive solution: check all assignments

Better solution: Linear Programming

Assignment: (vector of indices) ~~$f: A \rightarrow B$~~

$x_{ij} = 1$ when matching connection, otherwise 0

Similarity weights: ~~$w_{(i,f(i))} = g(\mathbf{z}_A^i, \mathbf{z}_B^{f(i)})$~~

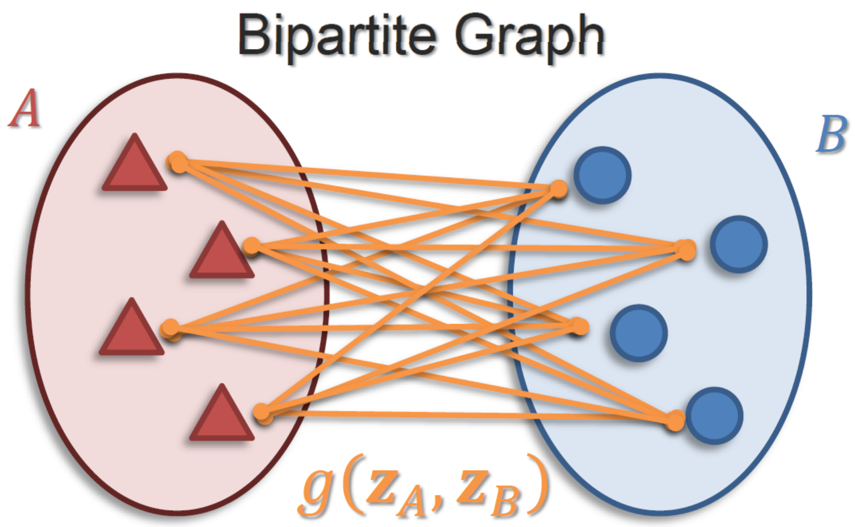
$$w_{(i,j)} = g(\mathbf{z}_A^i, \mathbf{z}_B^j)$$

Maximize: ~~$\max_{f \in \text{Perm}(N)} \sum_{i=1}^{N} w_{i,f(i)}$~~

$$\max_{\{x_{ij}\}} \sum_{(i,j) \in A \times B} w_{i,j} x_{ij}$$

Can be solved with simplex algorithm

Global Alignment



New assumptions:

- Different number of elements in A and B modalities
- Many-to-many “soft” alignment between elements

➡ It can be seen as “transporting” elements from modality A to modality B (and vice-versa)

Assignments: $x_{(i,j)}$: soft alignment between $\mathbf{z}_A^i$ and $\mathbf{z}_B^j$

Similarity weights: $w_{(i,j)} = g(\mathbf{z}_A^i, \mathbf{z}_B^j)$

Maximize: $$\max_{\{x_{ij}\}} \sum_{(i,j)\in A\times B} w_{i,j} x_{ij}$$

➡ Wassertein distance give optimal transport

内容提纲

① Discrete alignment

  ① Local alignment

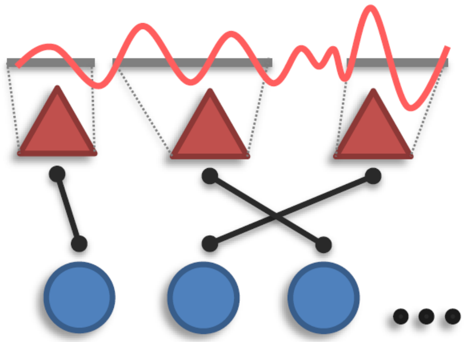
  ② Global alignment

② Continuous alignment

  ① Continuous warping

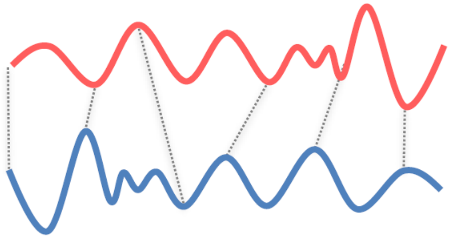
  ② Discretization and segmentation

Challenge 2b: Continuous Alignment

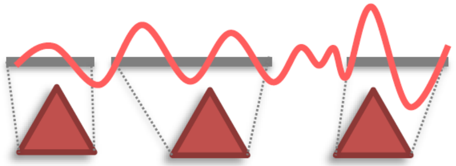


Definition: Model alignment between modalities with continuous signals and no explicit elements

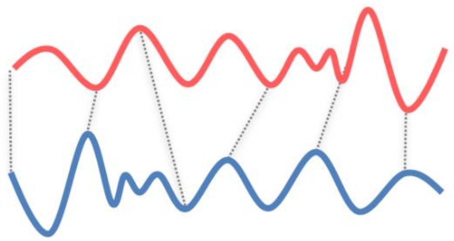
Continuous warping

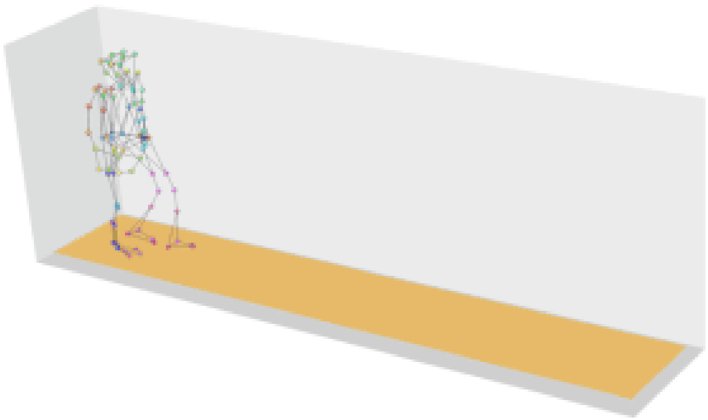


Discretization (segmentation)

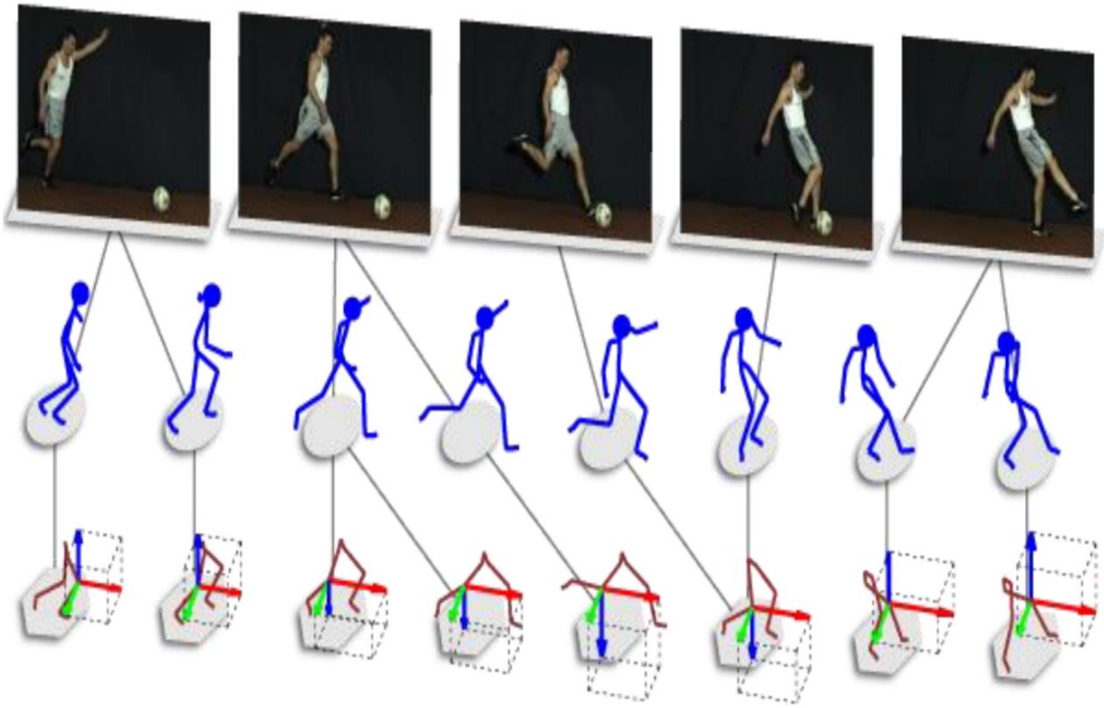


Continuous Warping – Example





Aligning video sequences



Dynamic Time Warping (DTW)

We have two unaligned temporal unimodal signals

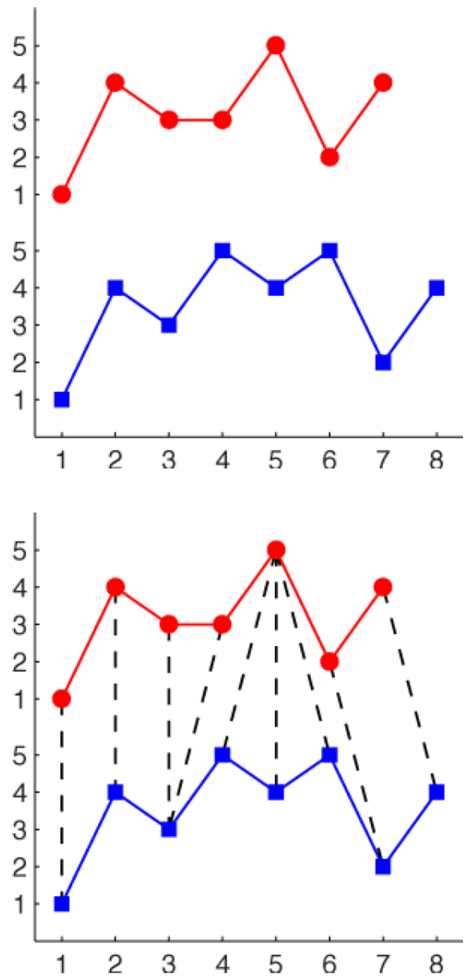
- $\mathbf{X} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{n_x}\right] \in \mathbb{R}^{d \times n_x}$
- $\mathbf{Y} = \left[\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{n_y}\right] \in \mathbb{R}^{d \times n_y}$

Find set of indices to minimize the alignment difference:

$$L(\boldsymbol{p}^x, \boldsymbol{p}^y) = \sum_{t=1}^{l} \left\| \boldsymbol{x}_{\boldsymbol{p}_t^x} - \boldsymbol{y}_{\boldsymbol{p}_t^y} \right\|_2^2$$

where $\boldsymbol{p}^x$ and $\boldsymbol{p}^y$ are index vectors of same length

Dynamic Time Warping is designed to find these index vectors!

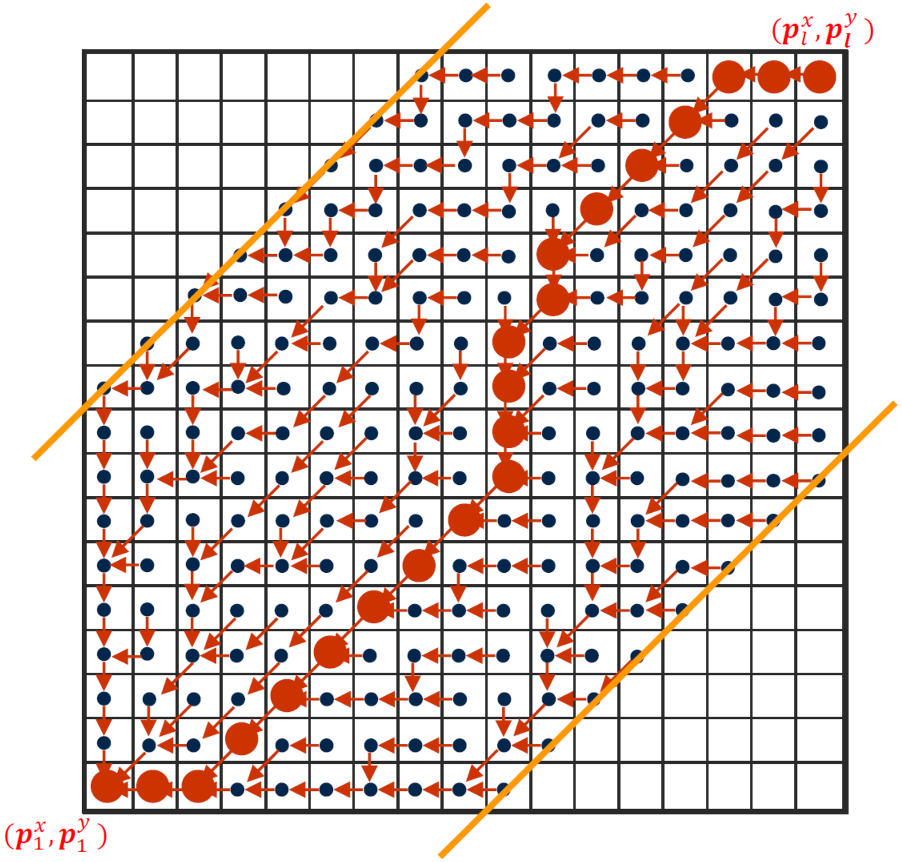


Dynamic Time Warping (DTW)

Lowest cost path in a cost matrix

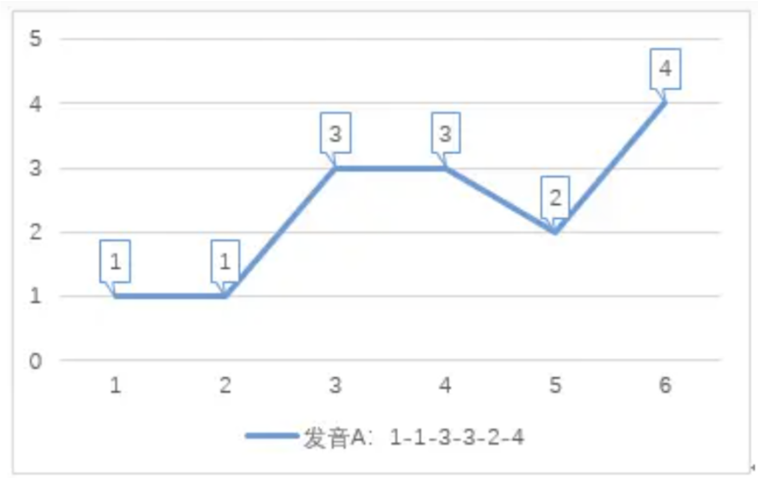
- Restrictions?
  - Monotonicity – no going back in time
  - Continuity - no gaps
  - Boundary conditions - start and end at the same points
  - Warping window - don't get too far from diagonal
  - Slope constraint – do not insert or skip too much

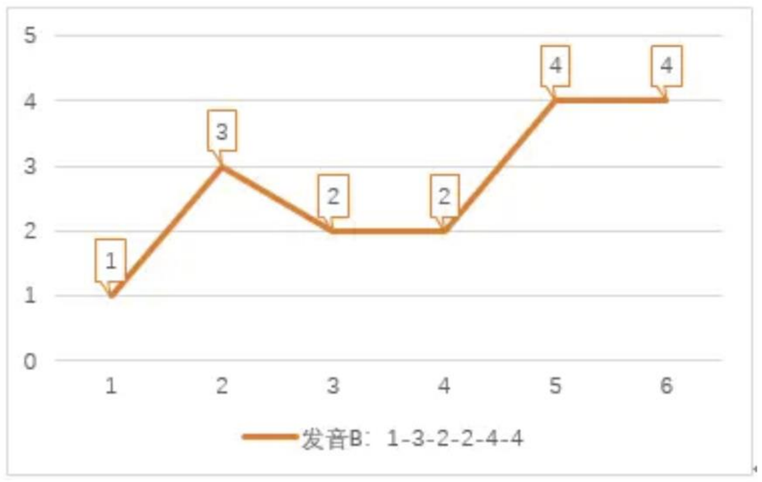
Solved using dynamic programming while respecting the restrictions



Dynamic Time Warping (DTW)

现在有两个人说这个单词，一个人在前半部分拖长，其发音为1-1-3-3-2-4；另一个人在后半部分拖长，其发音为1-3-2-2-4-4





	A(1)=1	A(2) =1	A(3) =3	A(4) =3	A(5) =2	A(6) =4
B(1) =1	0	0	2	2	1	3
B(2) =3	2	2	0	0	1	1
B(3) =2	1	1	1	1	0	2
B(4) =2	1	1	1	1	0	2
B(5) =4	3	3	1	1	2	0
B(6) =4	3	3	1	1	2	0

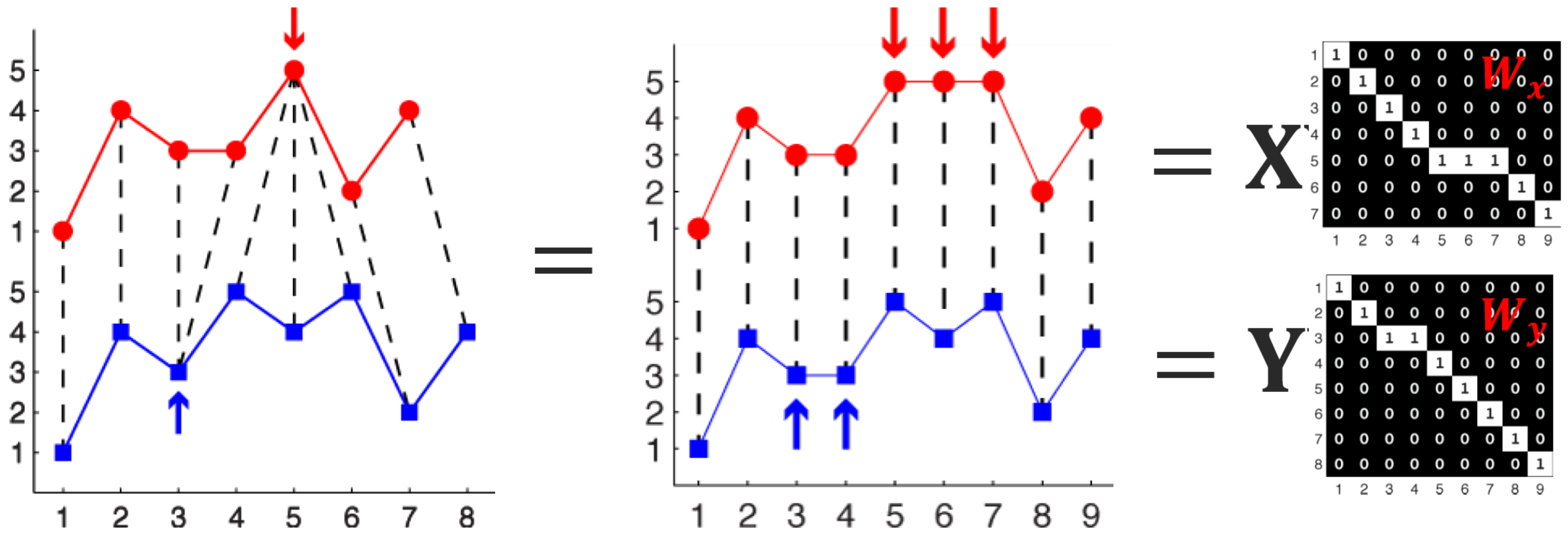
1. 计算两个序列各个点之间的距离矩阵

2. 寻找一条从矩阵左上角到右下角的路径，使得路径上的元素和最小

DTW alternative formulation

$$L(\boldsymbol{p}^x, \boldsymbol{p}^y) = \sum_{t=1}^{l} \left\| \boldsymbol{x}_{p_t^x} - \boldsymbol{y}_{p_t^y} \right\|_2^2$$

Replication doesn't change the objective!



Alternative objective:

$$L(\boldsymbol{W}_x, \boldsymbol{W}_y) = \left\| \boldsymbol{X}\boldsymbol{W}_x - \boldsymbol{Y}\boldsymbol{W}_y \right\|_F^2$$

Frobenius norm $\|\boldsymbol{A}\|_F^2 = \sum_i \sum_j |a_{i,j}|^2$

$\boldsymbol{X}, \boldsymbol{Y}$ – original signals (same #rows, possibly different #columns)

$\boldsymbol{W}_x, \boldsymbol{W}_y$ - alignment matrices

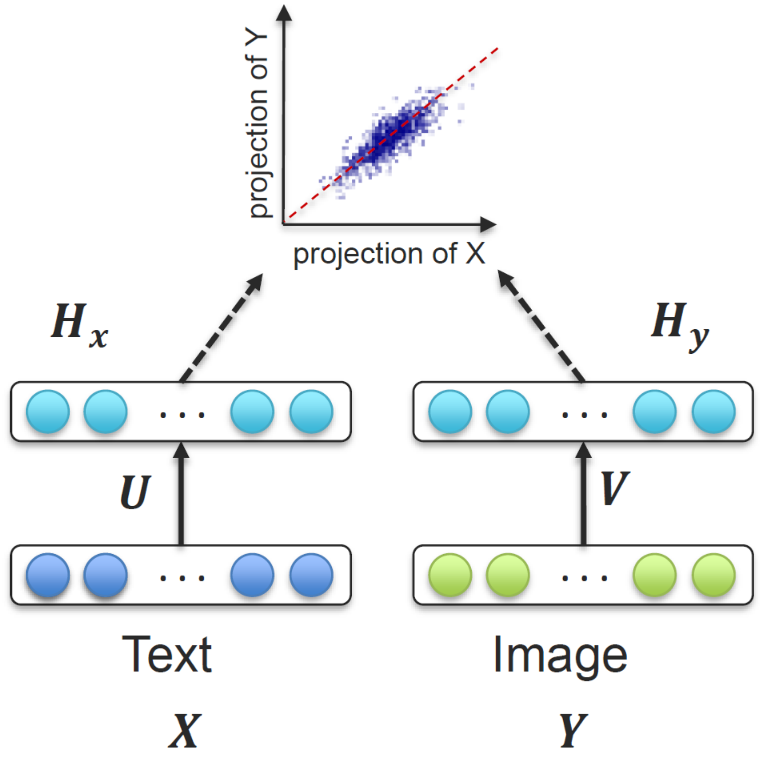
Canonical Correlation Analysis – Reminder

CCA loss can also be re-written as:

$$L(\boldsymbol{U}, \boldsymbol{V}) = \|\mathbf{U}^T\mathbf{X} - \mathbf{V}^T\mathbf{Y}\|_F^2$$

subject to:

$$\boldsymbol{U}^T\boldsymbol{\Sigma}_{YY}\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{\Sigma}_{YY}\boldsymbol{V} = \boldsymbol{I},\ \boldsymbol{u}_{(j)}^T\boldsymbol{\Sigma}_{XY}\boldsymbol{v}_{(i)} = \mathbf{0}$$



Canonical Time Warping

Dynamic Time Warping + Canonical Correlation Analysis = Canonical Time Warping

$$L(U, V, W_x, W_y) = \left\| \mathbf{U}^T \mathbf{X} \mathbf{W_x} - \mathbf{V}^T \mathbf{Y} \mathbf{W_y} \right\|_F^2$$

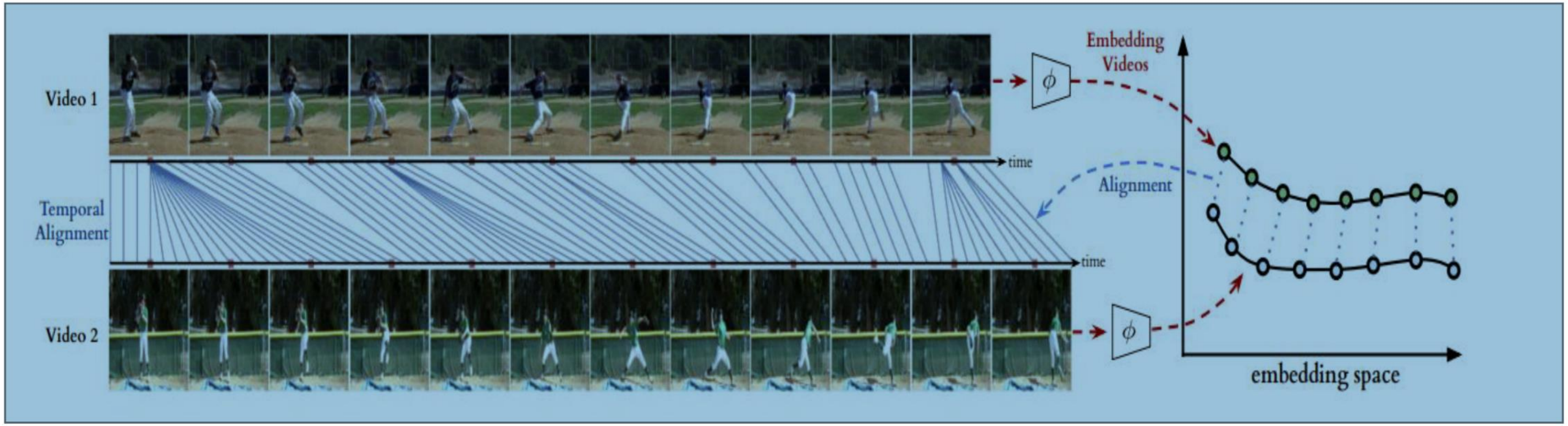
Allows to align multi-modal or multi-view (same modality but from a different point of view)

- W_x, W_y – temporal alignment
- U, V – cross-modal (spatial) alignment

Canonical Time Warping for Alignment of Human Behavior, Zhou and De la Tore, 2009

Temporal Alignment and Neural Representation Learning

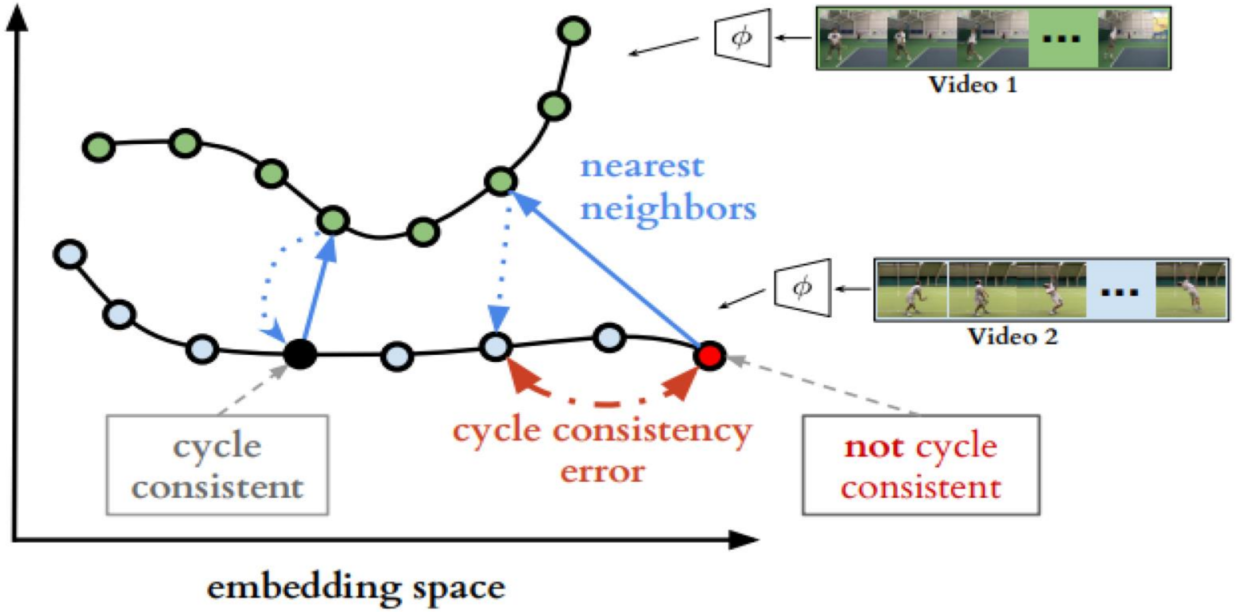
Premise: we have paired video sequences that can be be temporally aligned



How can we define a loss function to enforce the alignment between sequences while at the same time learning good representations?

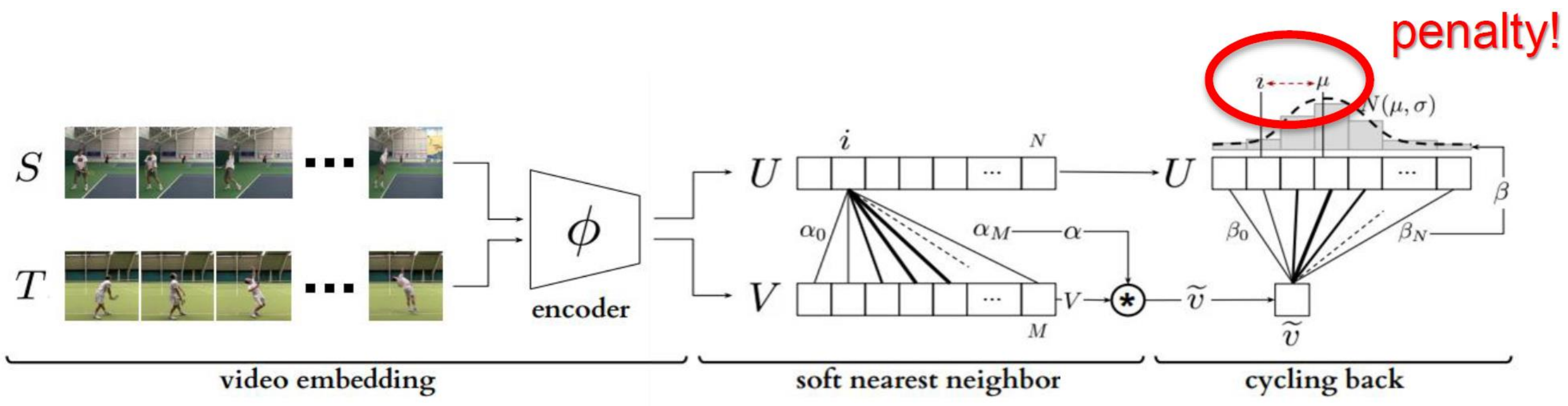
Temporal Cycle-Consistency Learning

Solution: Representation learning by enforcing **Cycle consistency**



Main idea: My closest neighbor also views me as their closest neighbor

Temporal Cycle-Consistency Learning



Compute “soft” / “weighted” nearest neighbour:

distances: $$\alpha_j = \frac{e^{-||u_i - v_j||^2}}{\sum_k^M e^{-||u_i - v_k||^2}}$$

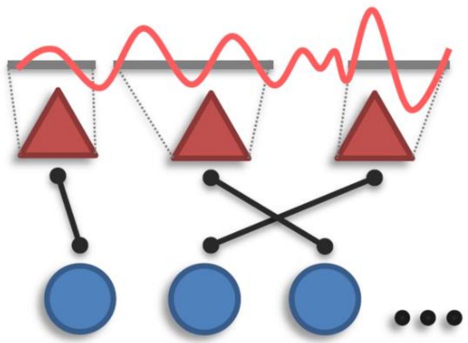
Soft nearest neighbor: $$\widetilde{v} = \sum_j^M \alpha_j v_j,$$

Find the nearest neighbor the other way and then penalize the distance:

$$\beta_k = \frac{e^{-||\widetilde{v} - u_k||^2}}{\sum_j^N e^{-||\widetilde{v} - u_j||^2}}$$

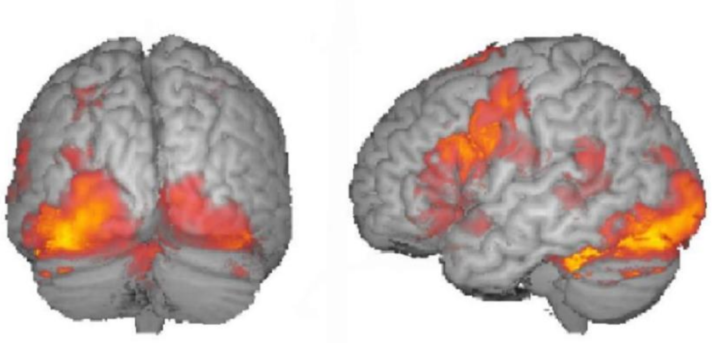
$$L_{cbr} = \frac{|i - \mu|^2}{\sigma^2} + \lambda \log(\sigma)$$

Discretization (aka Segmentation)

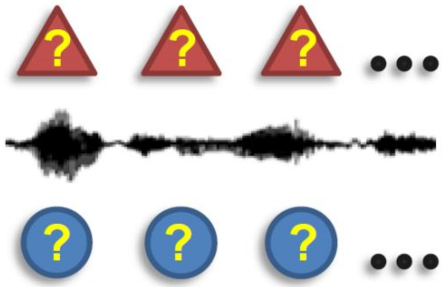


Common assumptions: ① Segmented elements

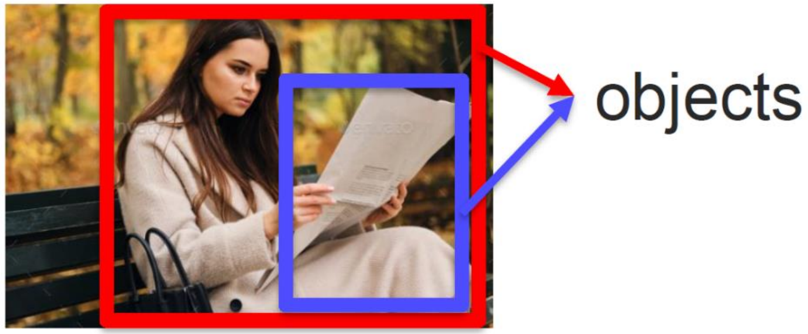
Examples:



Medical imaging



Signals

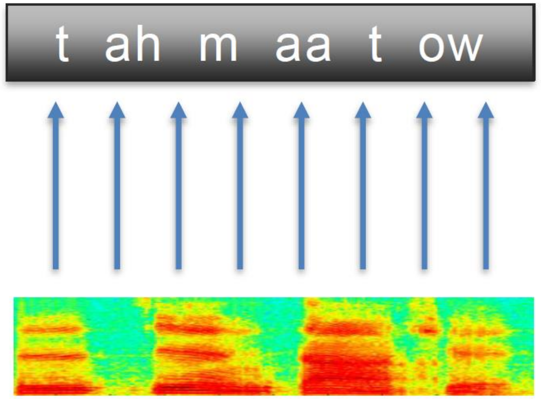


Images

Discretization – Example

Sequence Labeling and Alignment

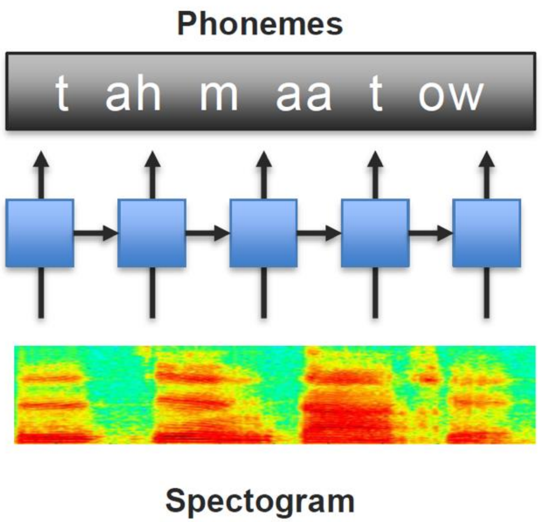
Phonemes

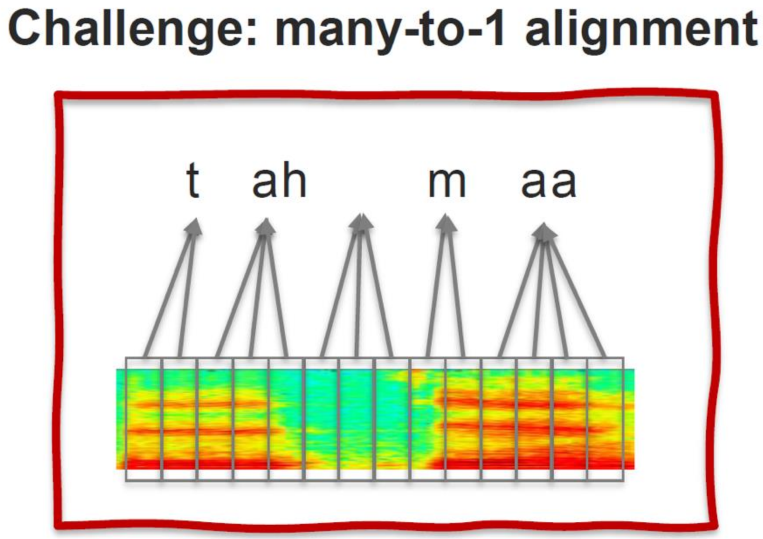


Spectogram

How can we predict the sequence of phoneme labels?

Sequence Labeling and Alignment



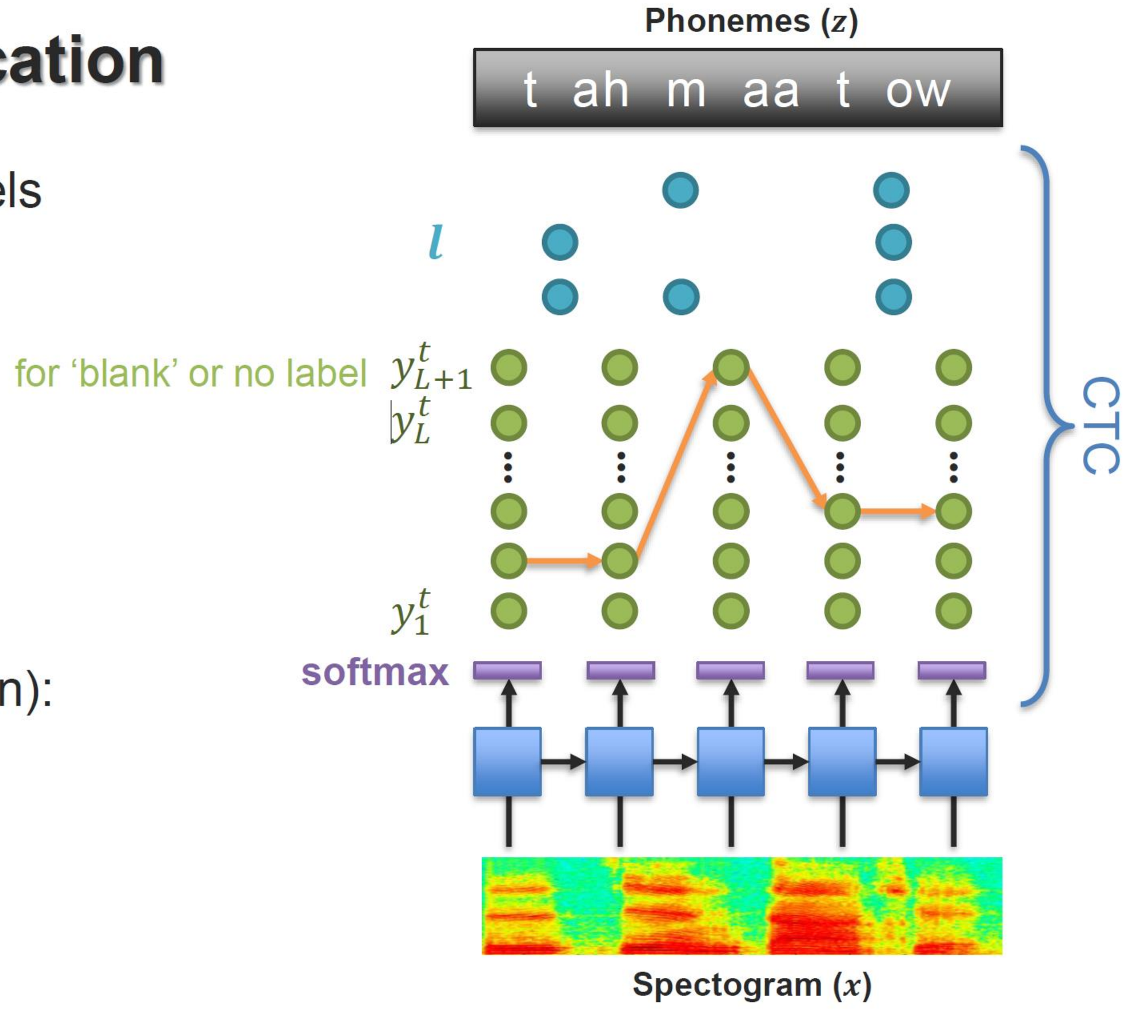


How can we predict the sequence of phoneme labels?

Discretization – A Classification Approach

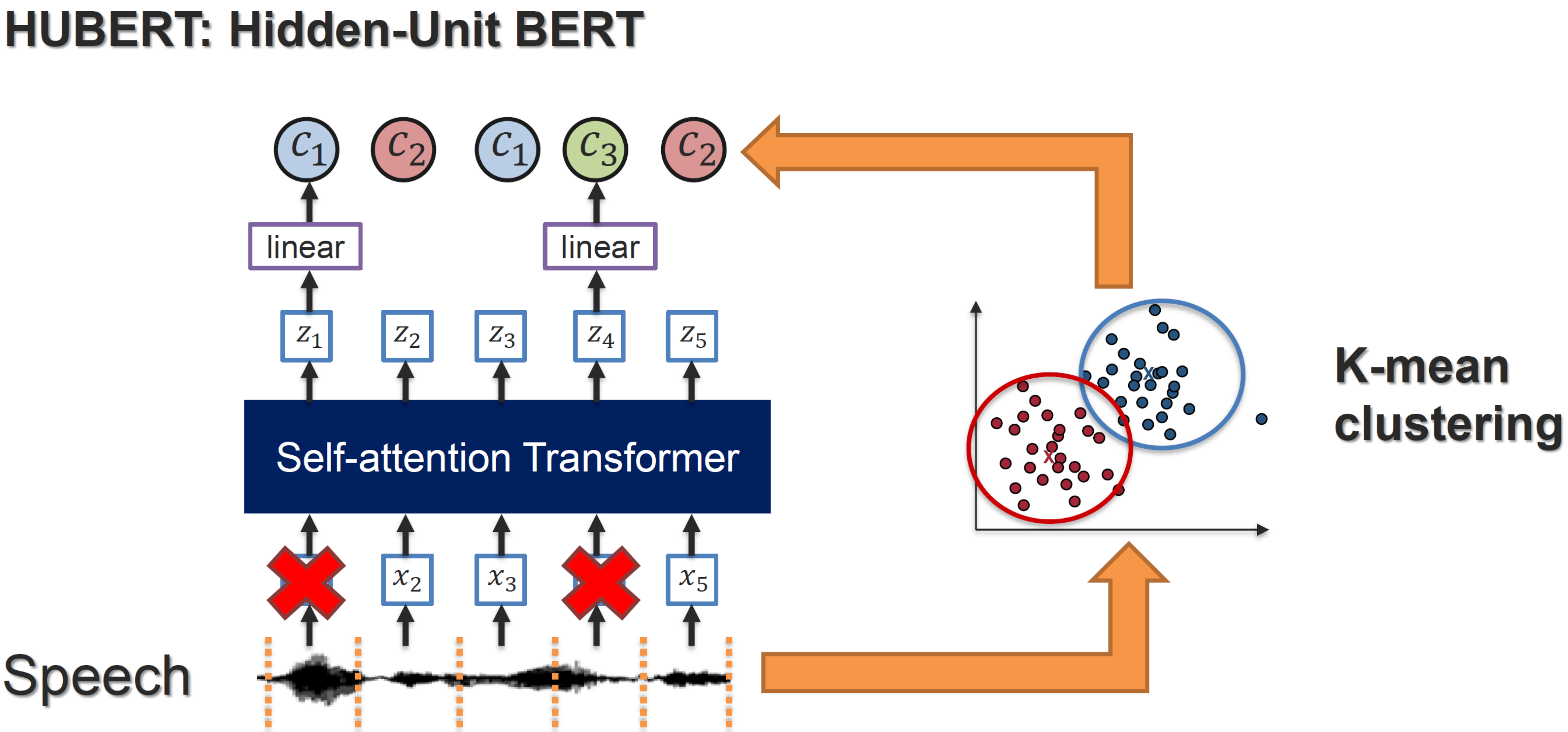
Connectionist Temporal Classification

4. Most probable sequence labels
3. Predicted labels $\boldsymbol{l}$
2. Path π over the activations:
1. Output activations (distribution):



Grave et al., Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks, ICML 2006

Discretization and Representation – Cluster-based Approaches



Hsu et al., HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units, arxiv 2021