Gossip-Based Machine Learning and Matrix Decomposition

István Hegedűs

Róber Ormándi

Márk Jelasity

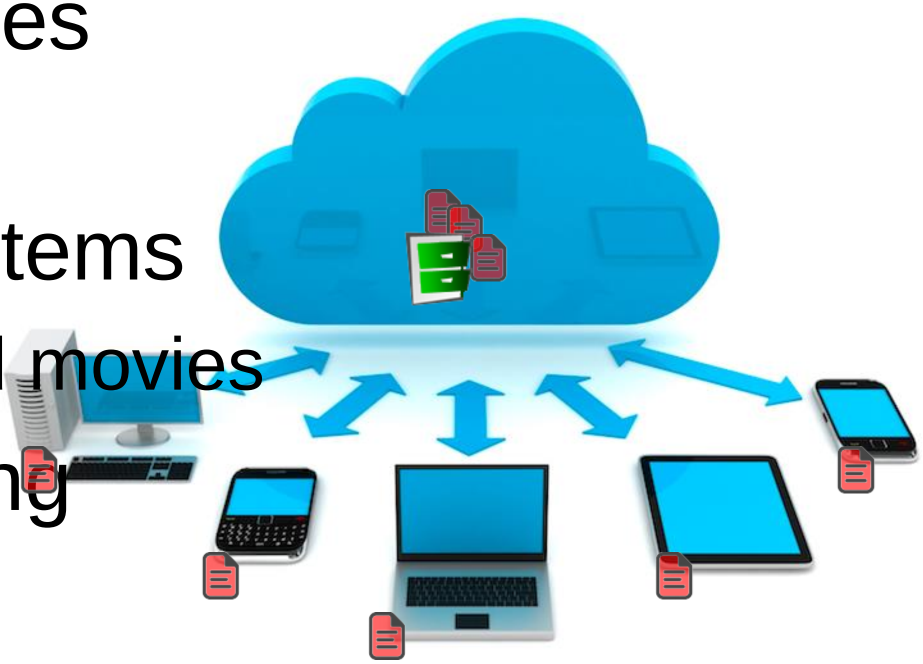
University of Szeged
MTA-SZTE Research Group on AI
Hungary

Motivation

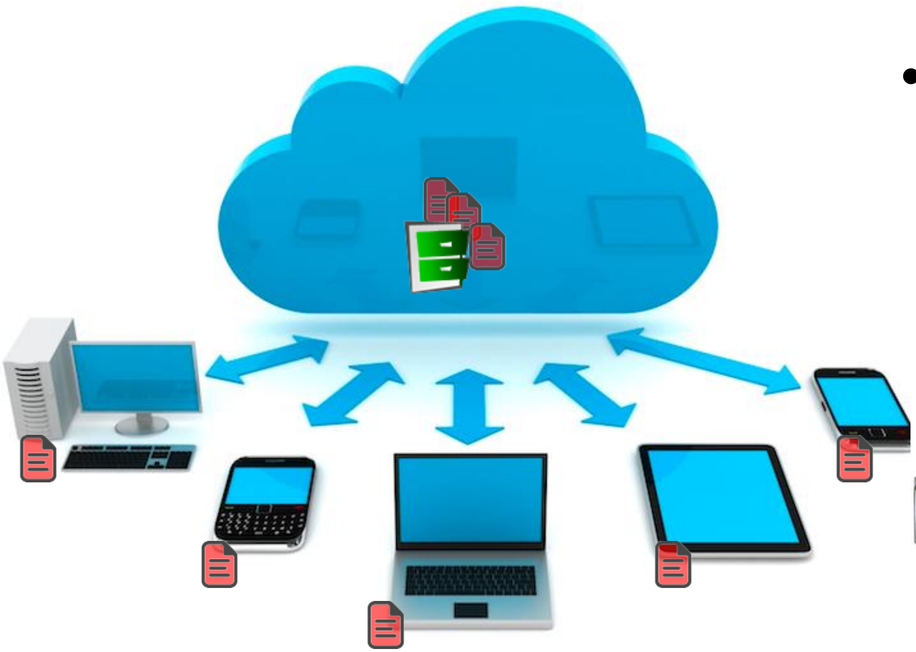
- Data is accumulated in data centers
- Costly storage and processing
  - Maintenance, Infrastructure, Privacy
- Limited access
  - For researchers as well
- But, data was produced by us

Motivation – ML Applications

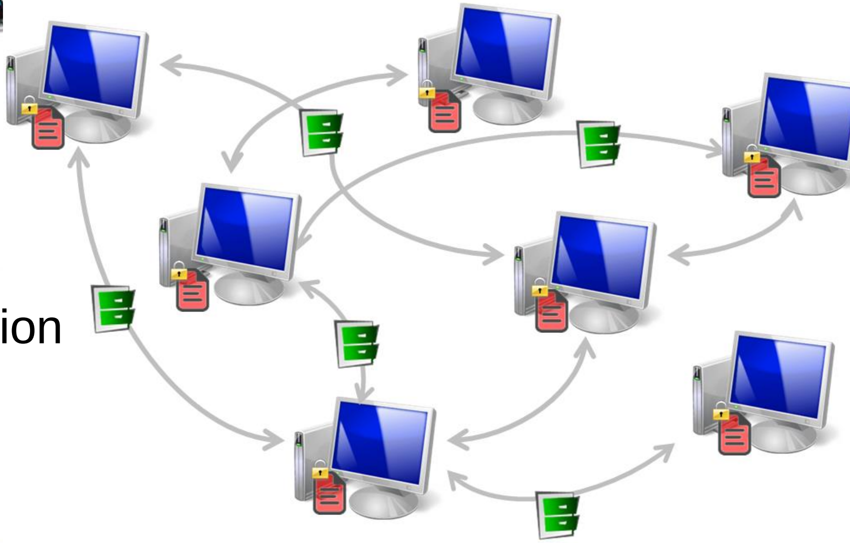
- Personalized Queries
  - Web search history
- Recommender Systems
  - Clicks on items and movies
- Document Clustering
- Spam Filtering
- Image Segmentation
  - We annotate pictures



Motivation – ML Applications

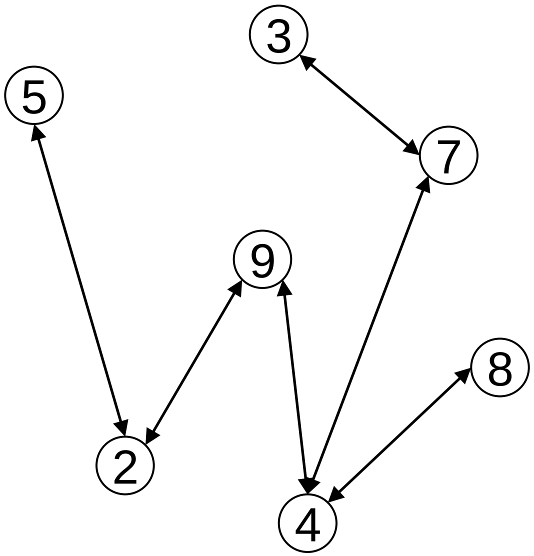


- Build models on the data collaboratively

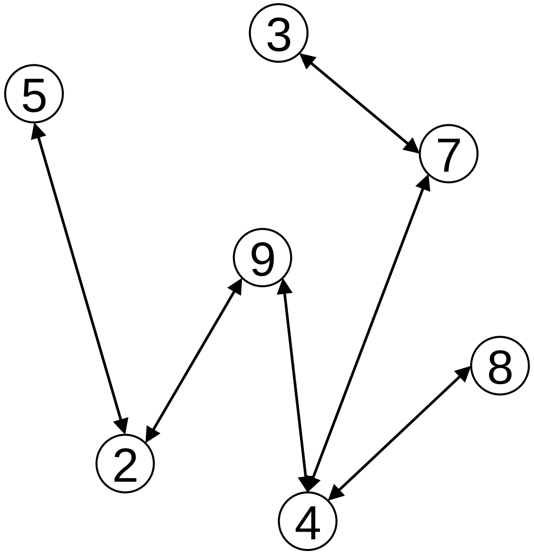


- Devices have internet connection
- Can communicate each other
- Store data as well

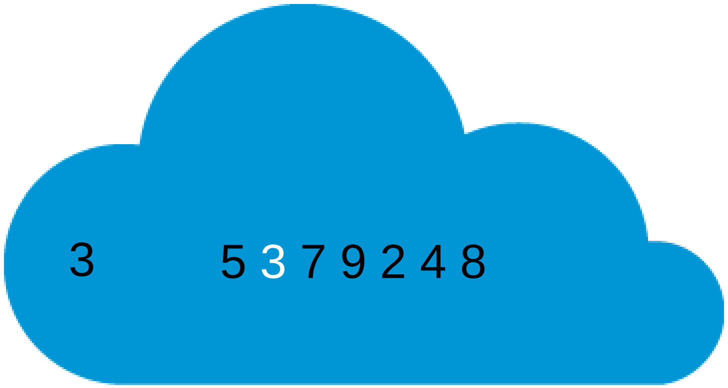
Example: compute min value

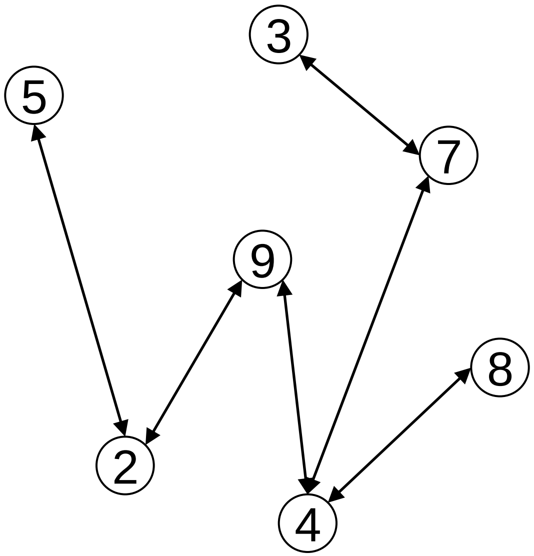


Example: compute min value

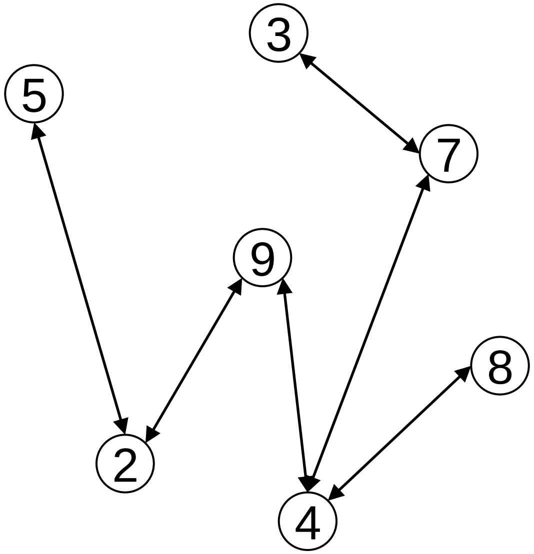


Example: compute min value

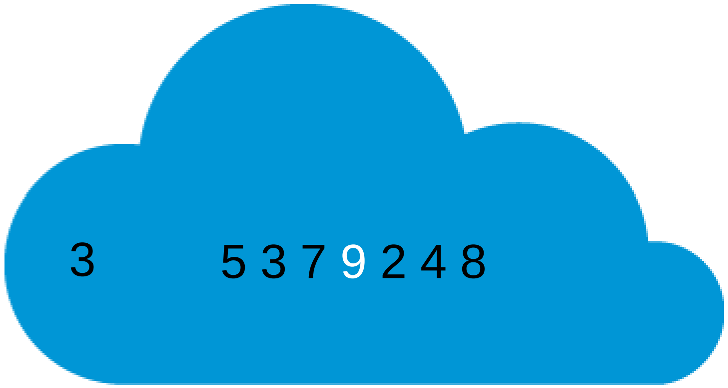


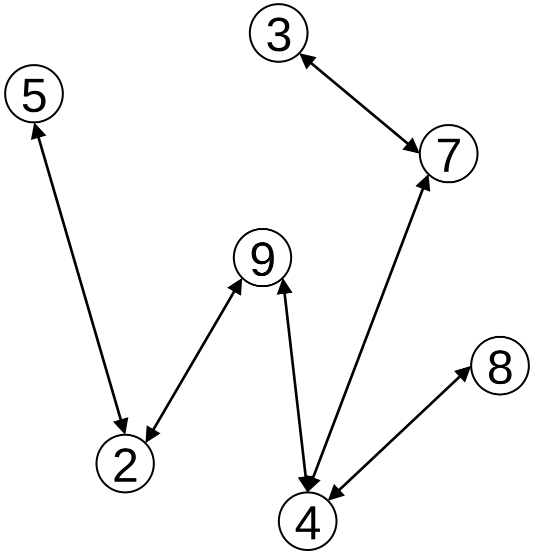


Example: compute min value

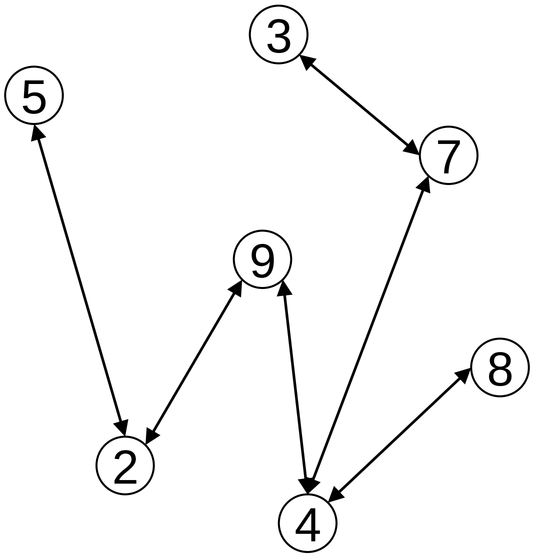


Example: compute min value

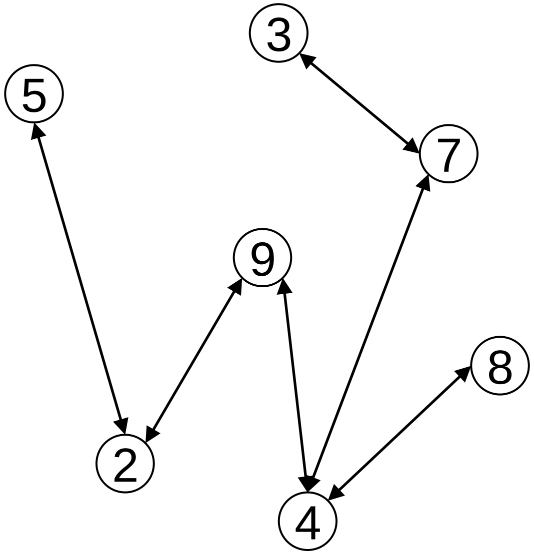




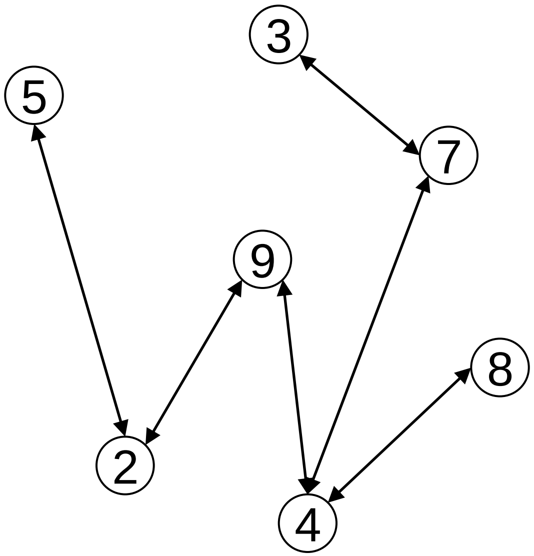
Example: compute min value



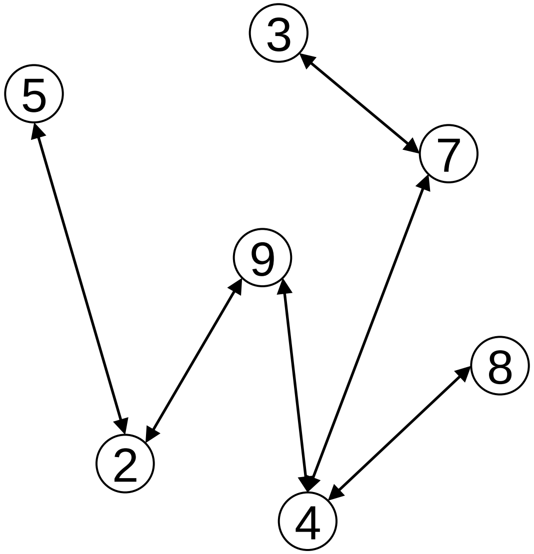
Example: compute min value



Example: compute min value

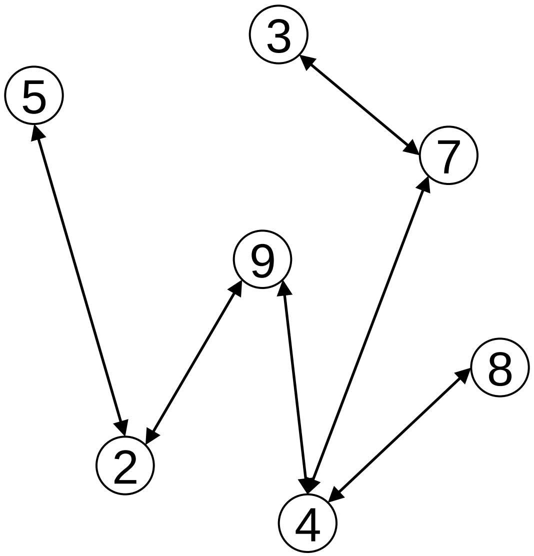


Example: compute min value

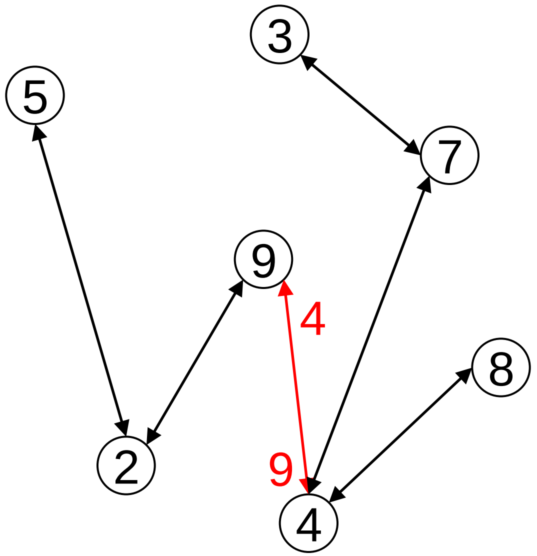


Centralized Algorithm
- Upload data
- Can not use the model without communication

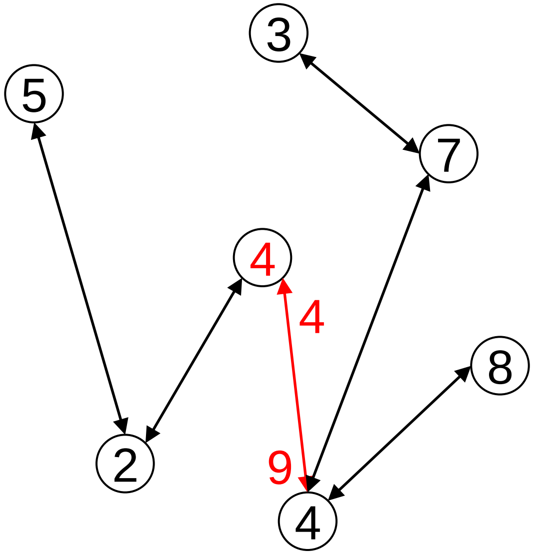
Example: compute min value



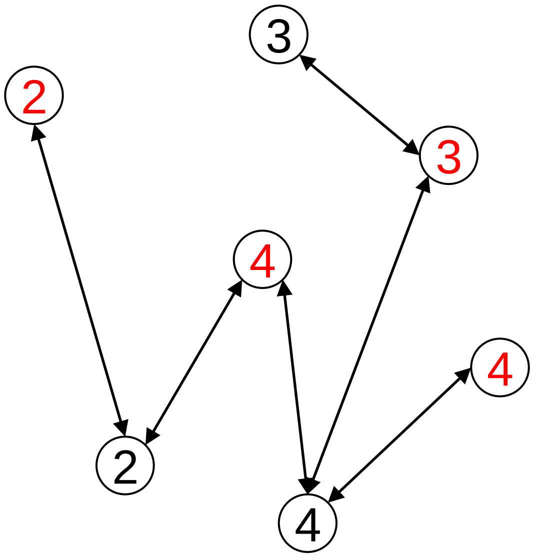
Example: compute min value



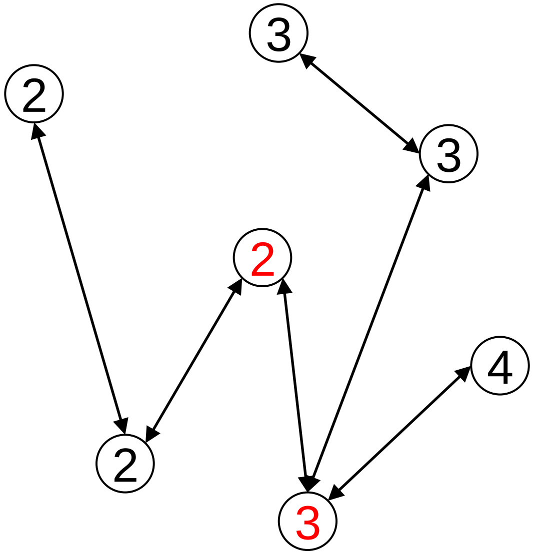
Example: compute min value



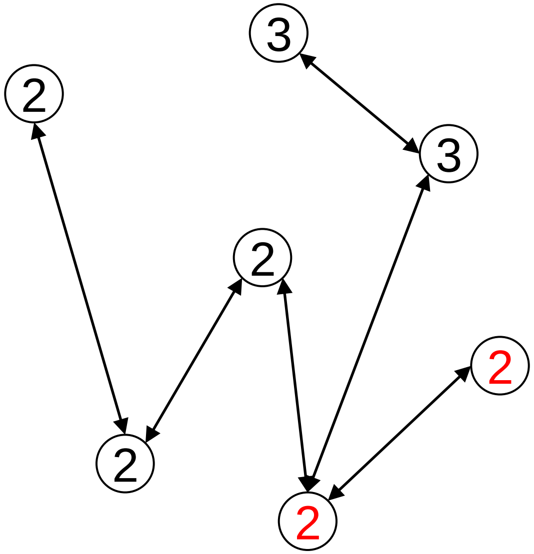
Example: compute min value



Example: compute min value



Example: compute min value

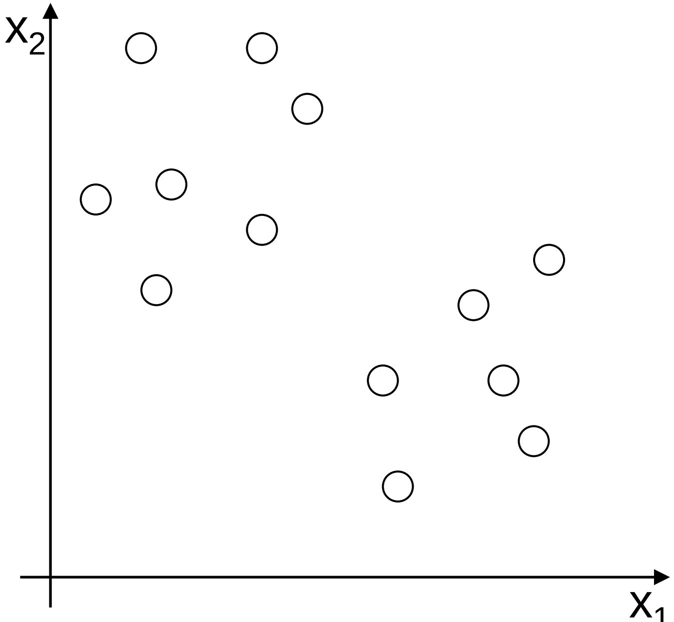


Distributed Algorithm

- More communication
- Can use the model without communication

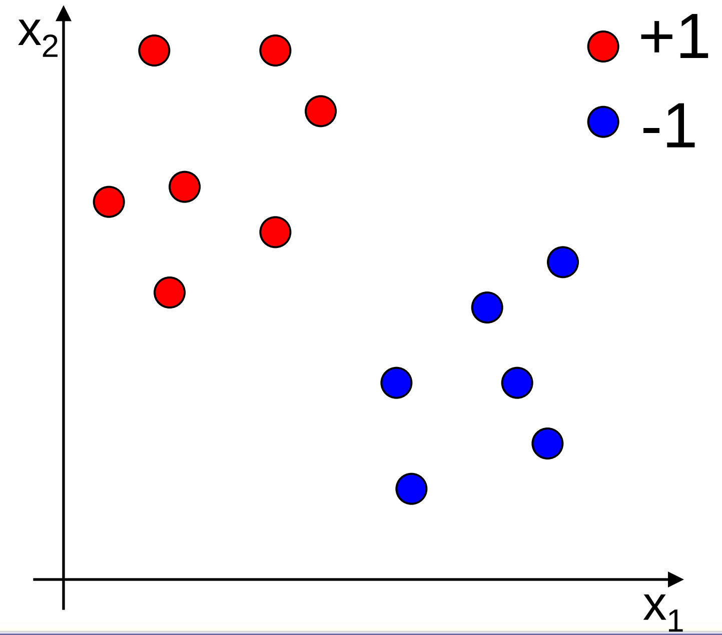
Classification

- Binary classification
  - Given a set of training samples: $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$



Classification

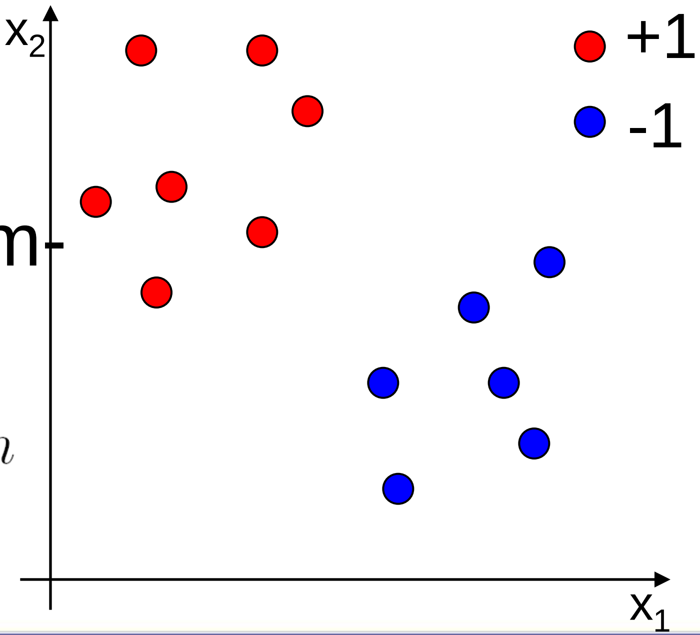
- Binary classification
  - Given a set of training samples: $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$



Classification

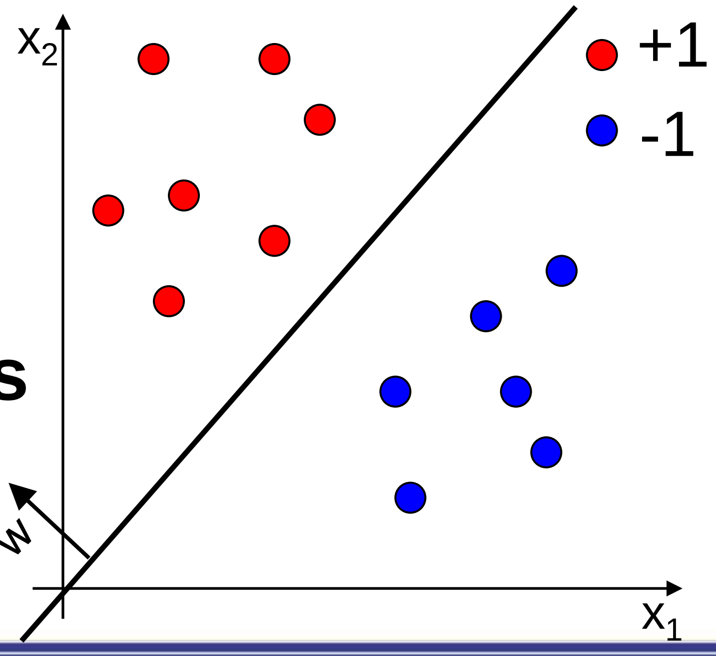
- Binary classification
  - Given $(x_1, y_1), \ldots, (x_n, y_n)$ training samples, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
  - Task: looking for a **model** $f : \mathbb{R}^d \to \{-1, 1\}$ that correctly separates the samples from different classes (minimizes the number of misclassifications)

$$\min_f \sum_i (f(x_i) - y_i)^2 \quad i = 1, \ldots, n$$



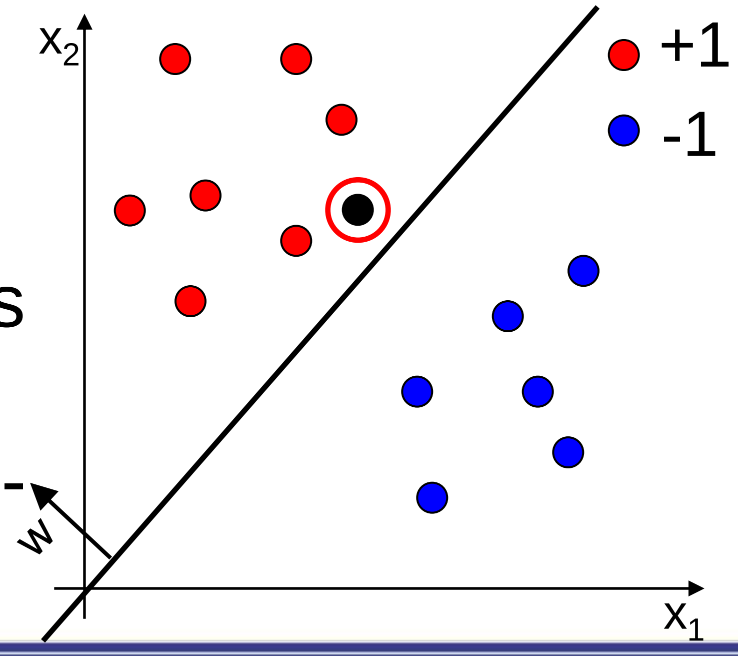
Classification

- Binary classification
  - Given $(x_1, y_1), \ldots, (x_n, y_n)$ training samples, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
  - Task: looking for a model
    $$f : \mathbb{R}^d \to \{-1, 1\}$$
    minimizes the error
    $$\min_f \sum_i (f(x_i) - y_i)^2 \quad i = 1, \ldots, n$$
  - In linear case the **model is a hyper-plane** (w)



Classification

- Binary classification
  - Given $(x_1, y_1), \ldots, (x_n, y_n)$ training samples, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
  - Task: looking for a model
    $$f : \mathbb{R}^d \to \{-1, 1\}$$
    minimizes the error
    $$\min_f \sum_i (f(x_i) - y_i)^2 \quad i = 1, \ldots, n$$
  - In linear case the model is a hyper-plane (w)
  - The **label** of a new instance **can be predicted**

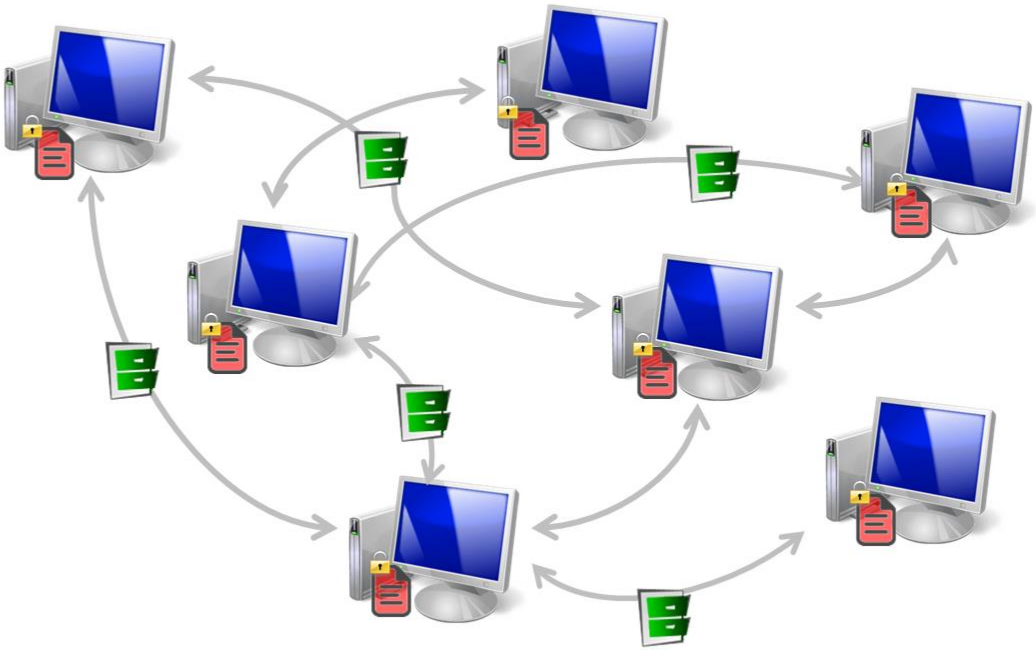


Gossip Learning

- ML is often an optimization problem
- Local data is not enough

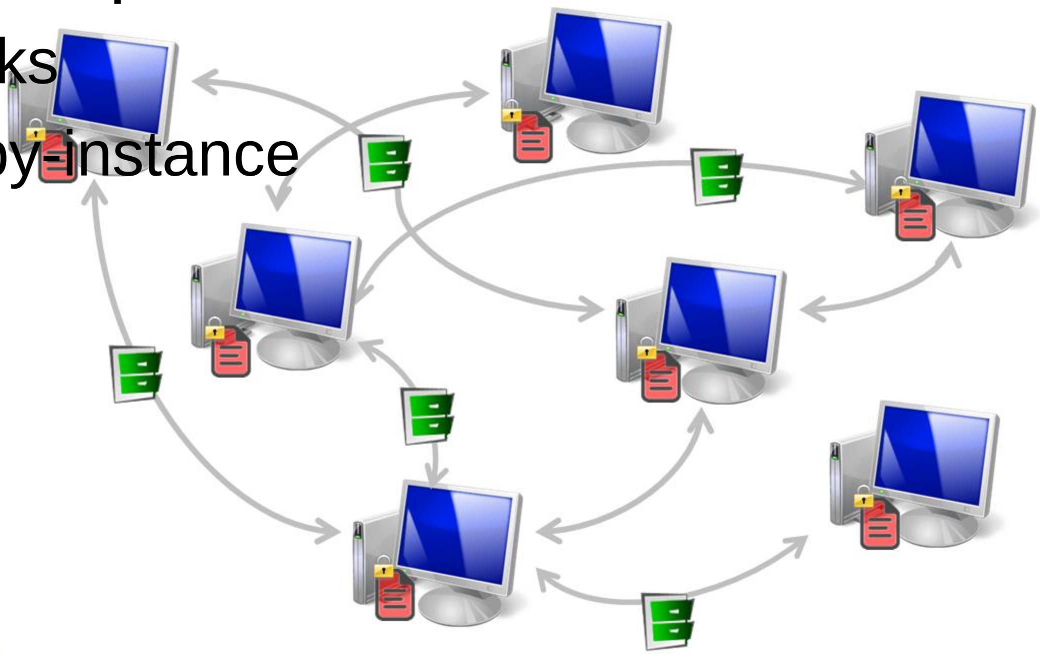
Gossip Learning

- ML is often an optimization problem
- Local data is not enough
- Models are sent and updated on nodes



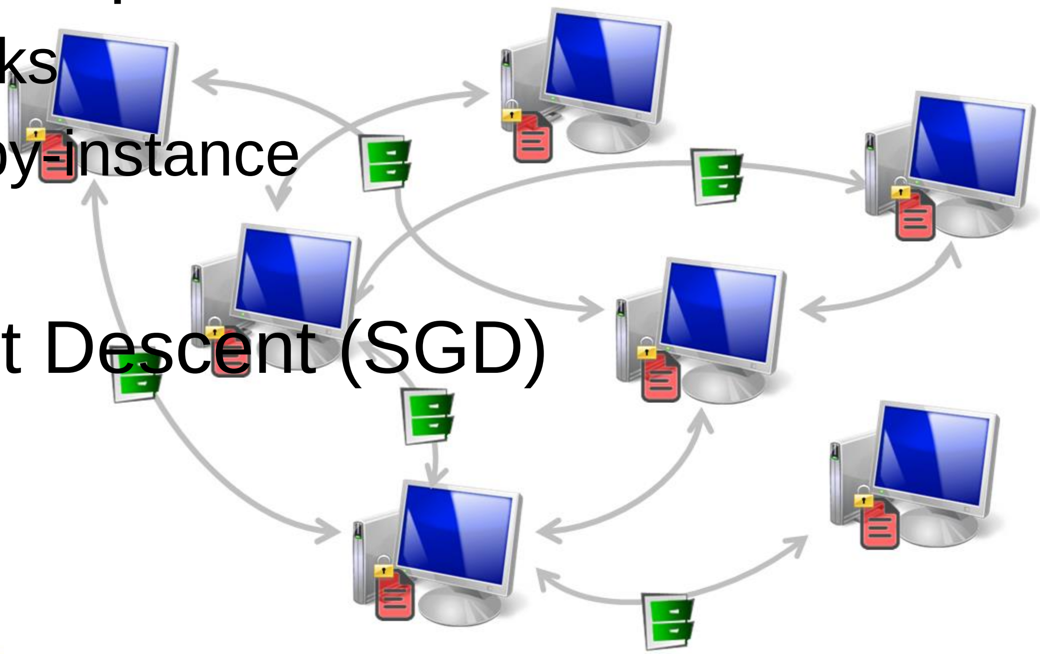
Gossip Learning

- ML is often an optimization problem
- Local data is not enough
- Models are sent and updated on nodes
  - Taking random walks
  - Updated instance-by-instance
  - Data is never sent



Gossip Learning

- ML is often an optimization problem
- Local data is not enough
- Models are sent and updated on nodes
  - Taking random walks
  - Updated instance-by-instance
  - Data is never sent
- Stochastic Gradient Descent (SGD)



SGD

- Objective function

$$w = \arg\min_{w} J(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \frac{\lambda}{2}\|w\|^2$$

SGD

- Objective function

$$w = \arg\min_w J(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \frac{\lambda}{2}\|w\|^2$$

- Gradient method

$$w_{t+1} = w_t - \eta_t(\frac{\partial J}{\partial w})$$

$$= w_t - \eta_t(\lambda w + \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(f_w(x_i), y_i))$$

SGD

- Objective function

$$w = \arg\min_w J(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \frac{\lambda}{2}\|w\|^2$$

- Gradient method

$$\begin{aligned} w_{t+1} &= w_t - \eta_t(\frac{\partial J}{\partial w}) \\ &= w_t - \eta_t(\lambda w + \frac{1}{n}\sum_{i=1}^{n} \nabla \ell(f_w(x_i), y_i)) \end{aligned}$$

- SGD, data can be processed online (instance by instance)

$$w_{t+1} = w_t - \eta_t(\lambda w + \nabla \ell(f_w(x_i), y_i))$$

SGD

- Objective function

$$w = \arg\min_w J(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \frac{\lambda}{2}\|w\|^2$$

- Gradient method

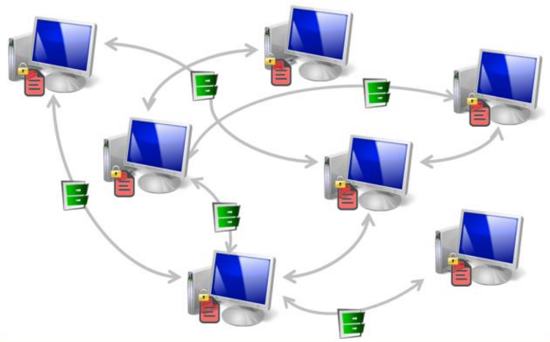
$$w_{t+1} = w_t - \eta_t(\frac{\partial J}{\partial w})$$

$$= w_t - \eta_t(\lambda w + \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(f_w(x_i), y_i))$$

- SGD, data can be processed online (instance by instance)

$$w_{t+1} = w_t - \eta_t(\lambda w + \nabla\ell(f_w(x_i), y_i))$$

- Gossip Learning

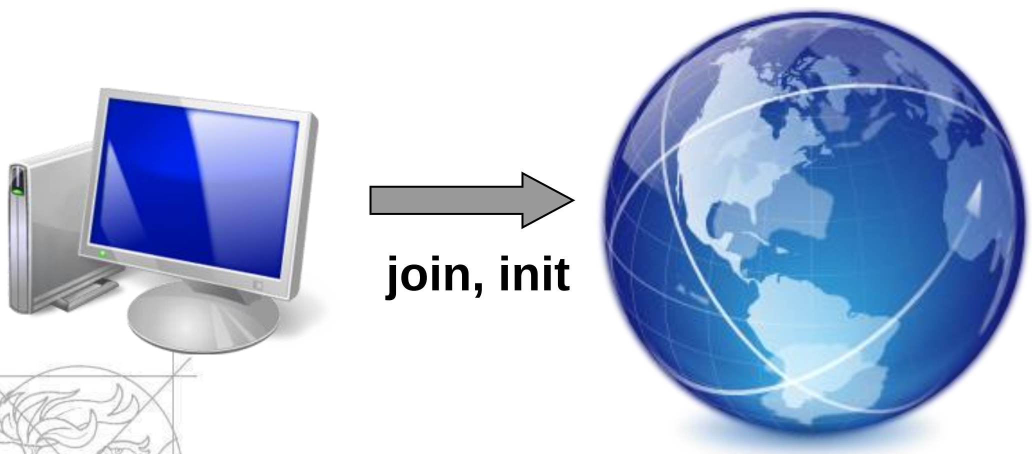


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

```
currentModel ← initModel()
loop
    wait(Δ)
    p ← selectPeer()
    sendModel(p, currentModel)

procedure onReceiveModel(m)
    m.updateModel(x, y)
    currentModel ← m
```

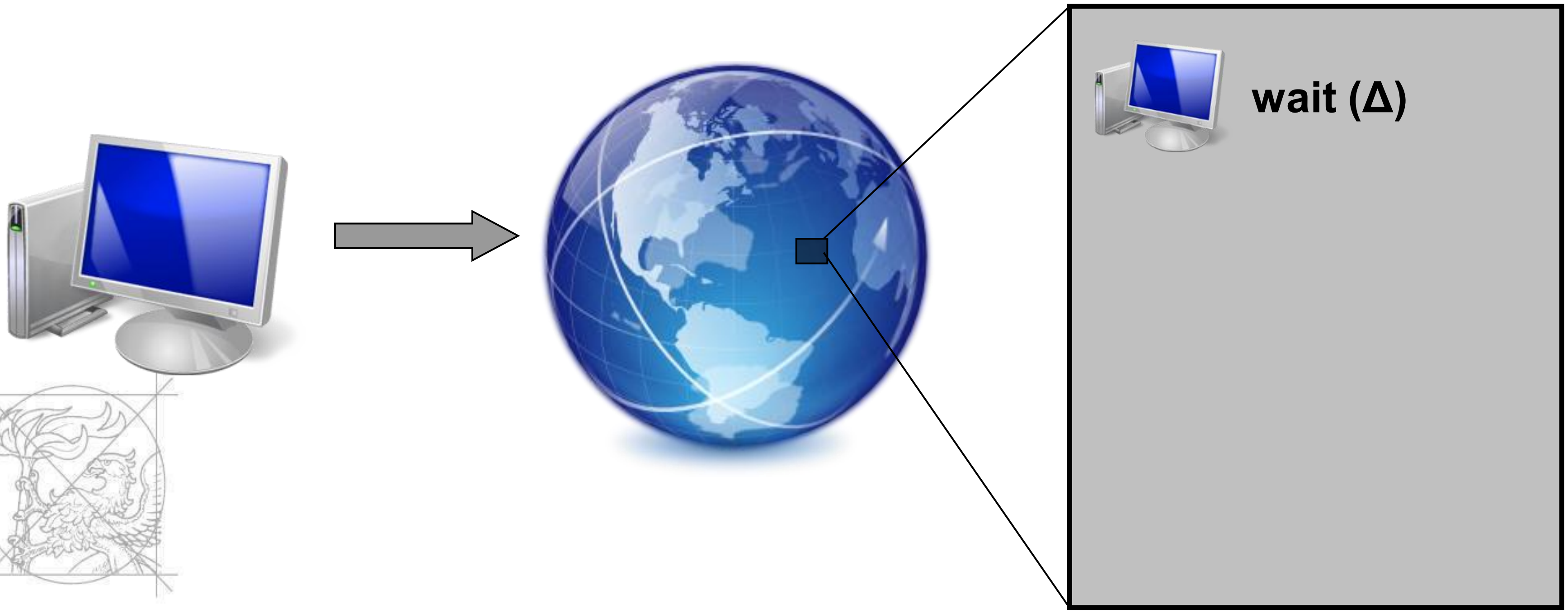


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
    wait(Δ)
    $p \leftarrow$ selectPeer()
    sendModel(p, $currentModel$)

procedure ONRECEIVEMODEL(m)
    m.updateModel(x, y)
    $currentModel \leftarrow m$

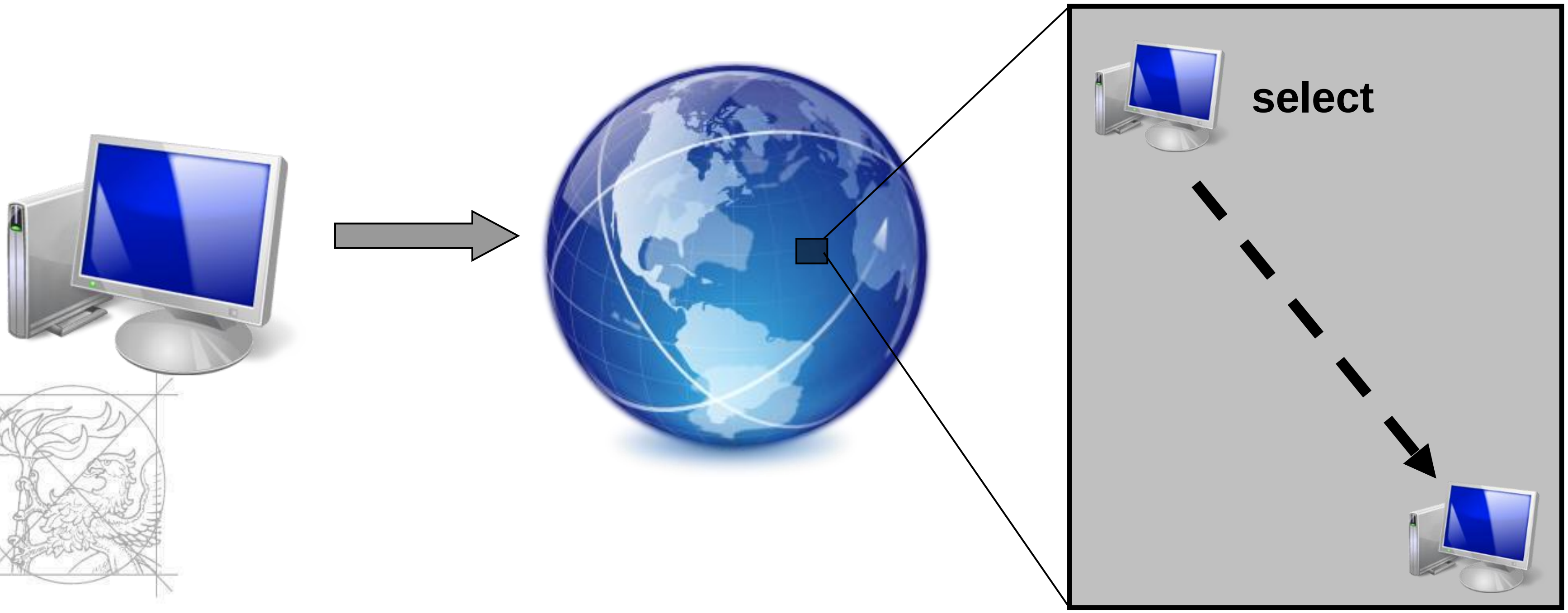


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

```
currentModel ← initModel()
loop
    wait(Δ)
    p ← selectPeer()
    sendModel(p, currentModel)

procedure onReceiveModel(m)
    m.updateModel(x, y)
    currentModel ← m
```

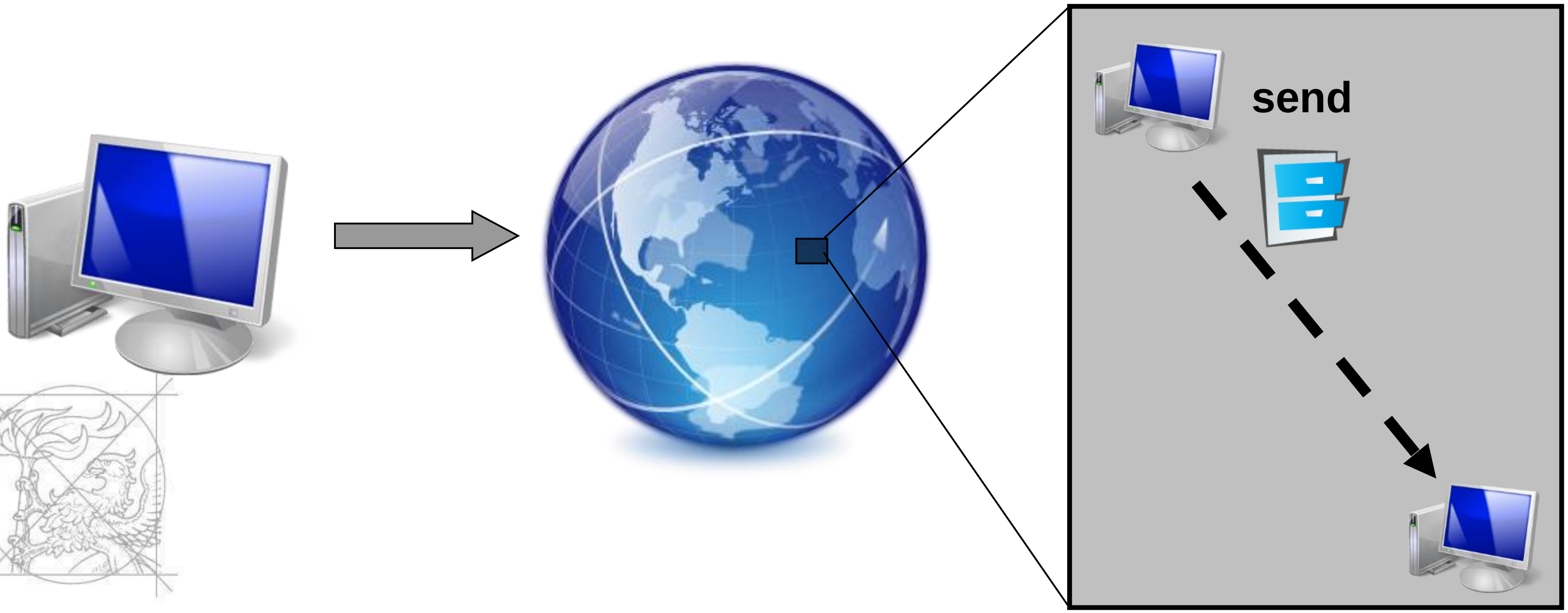


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
    wait(Δ)
    $p \leftarrow$ selectPeer()
    sendModel(p, $currentModel$)

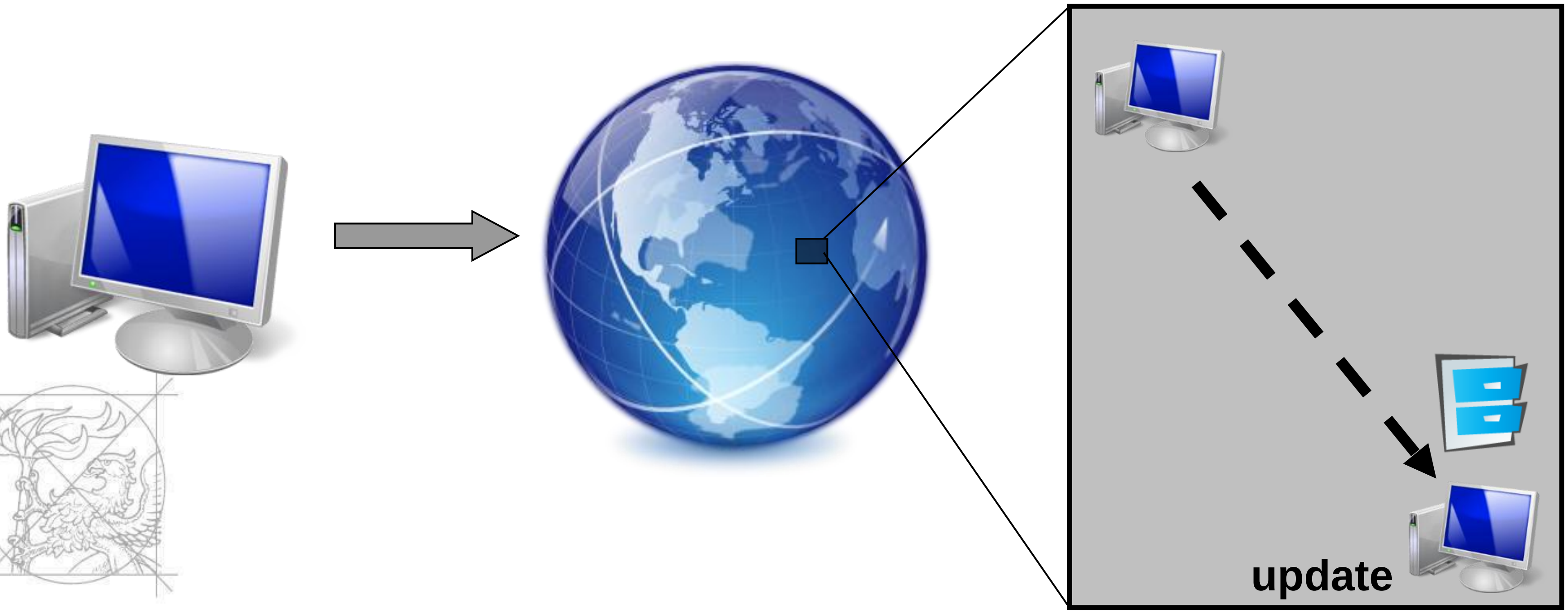
procedure ONRECEIVEMODEL(m)
    m.updateModel(x, y)
    $currentModel \leftarrow m$



GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
    wait(Δ)
    $p \leftarrow$ selectPeer()
    sendModel(p, $currentModel$)
procedure ONRECEIVEMODEL(m)
    m.updateModel(x, y)
    $currentModel \leftarrow m$

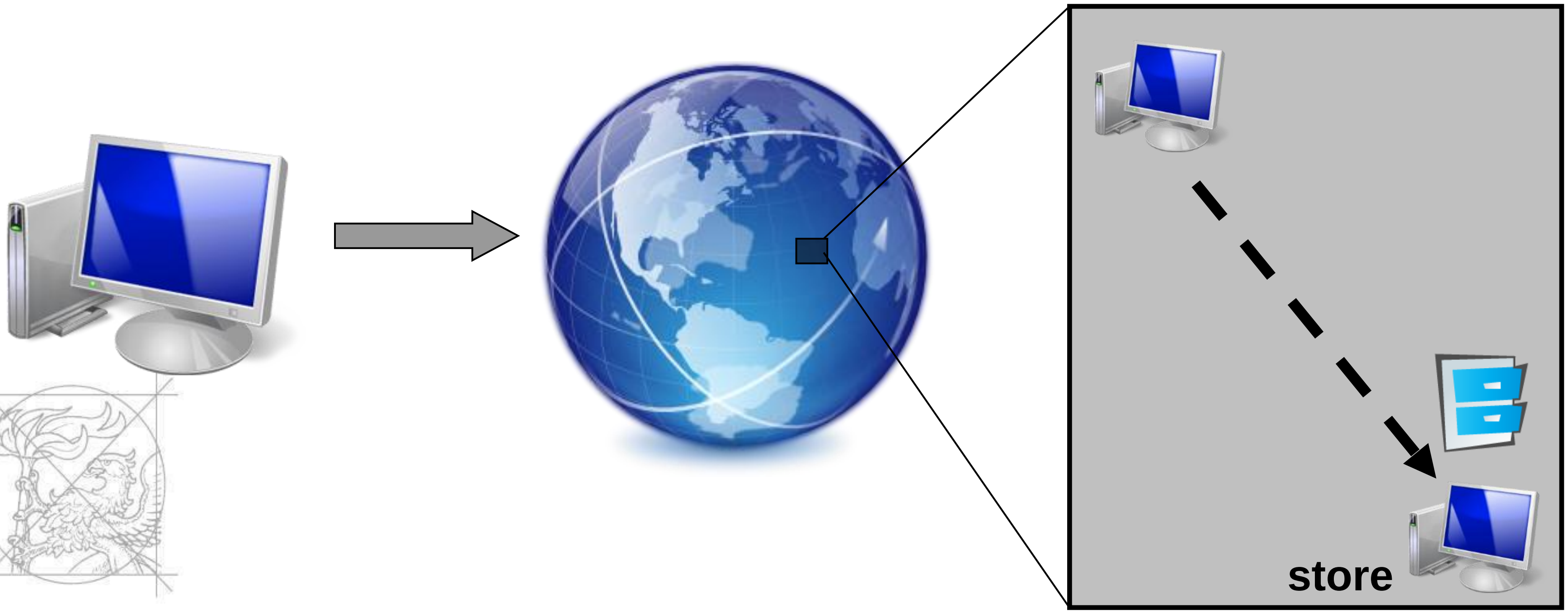


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
  wait(Δ)
  $p \leftarrow$ selectPeer()
  sendModel(p, $currentModel$)

procedure OnReceiveModel(m)
  m.updateModel(x, y)
  $currentModel \leftarrow m$



Gossip-Based Learning

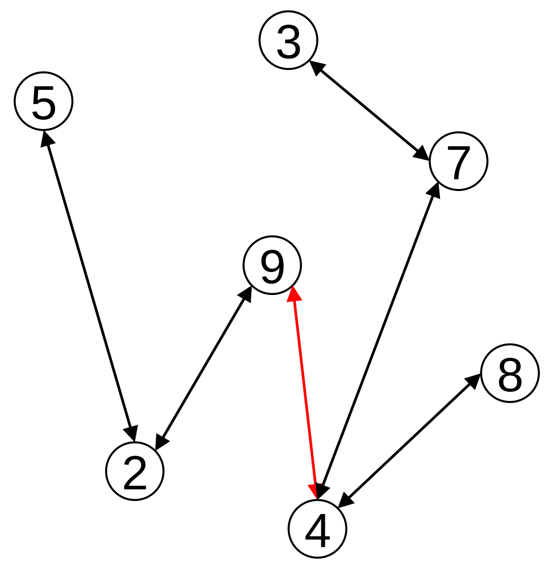
- SGD-based machine learning algorithms can be applied, e.g.
  - Logistic Regression
  - Support Vector Machines
  - Perceptron
  - Artificial Neural Networks
- Training data never leave the nodes
- Models can be used locally additional communication is not required

GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

```
currentModel ← initModel()
loop
    wait(Δ)
    p ← selectPeer()
    sendModel(p, currentModel)

procedure ONRECEIVEMODEL(m)
    m.updateModel(x, y)
    currentModel ← m
```

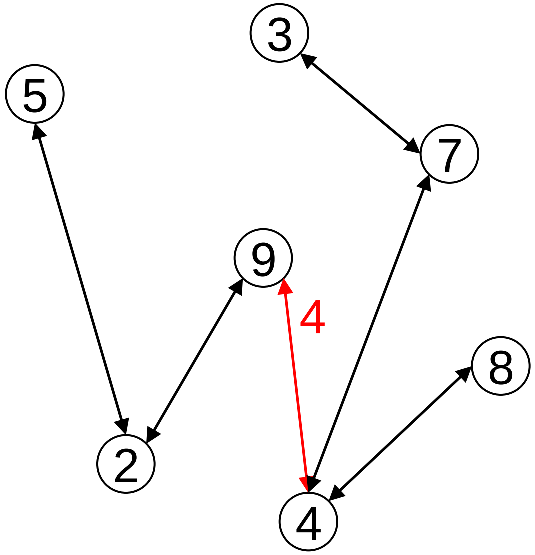


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
  wait(Δ)
  $p \leftarrow$ selectPeer()
  sendModel(p, $currentModel$)

procedure onReceiveModel(m)
  m.updateModel(x, y)
  $currentModel \leftarrow m$

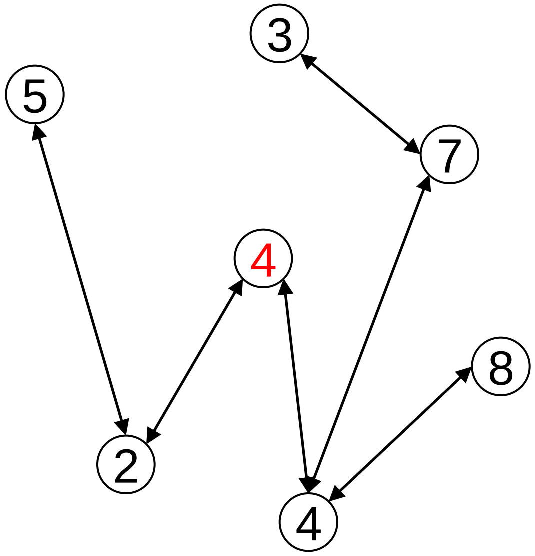


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

$currentModel \leftarrow$ initModel()
loop
  wait(Δ)
  $p \leftarrow$ selectPeer()
  sendModel(p, $currentModel$)

procedure onReceiveModel(m)
  m.updateModel(x, y)
  $currentModel \leftarrow m$

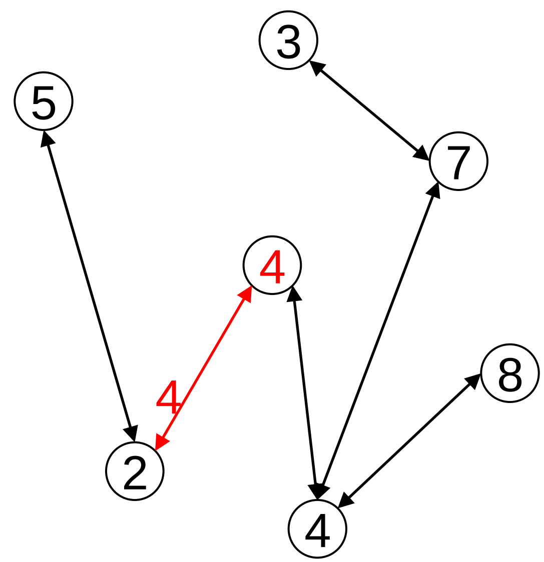


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

```
currentModel ← initModel()
loop
    wait(Δ)
    p ← selectPeer()
    sendModel(p, currentModel)

procedure onReceiveModel(m)
    m.updateModel(x, y)
    currentModel ← m
```

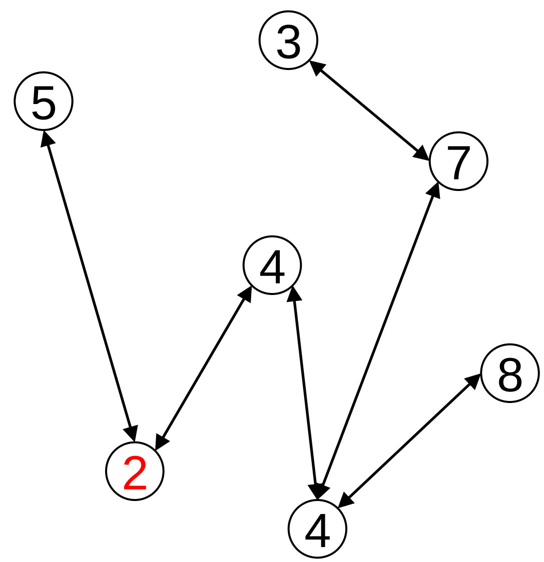


GoLF (Gossip Learning Framework)

Algorithm 1 Skeleton of original GoLF learning protocol

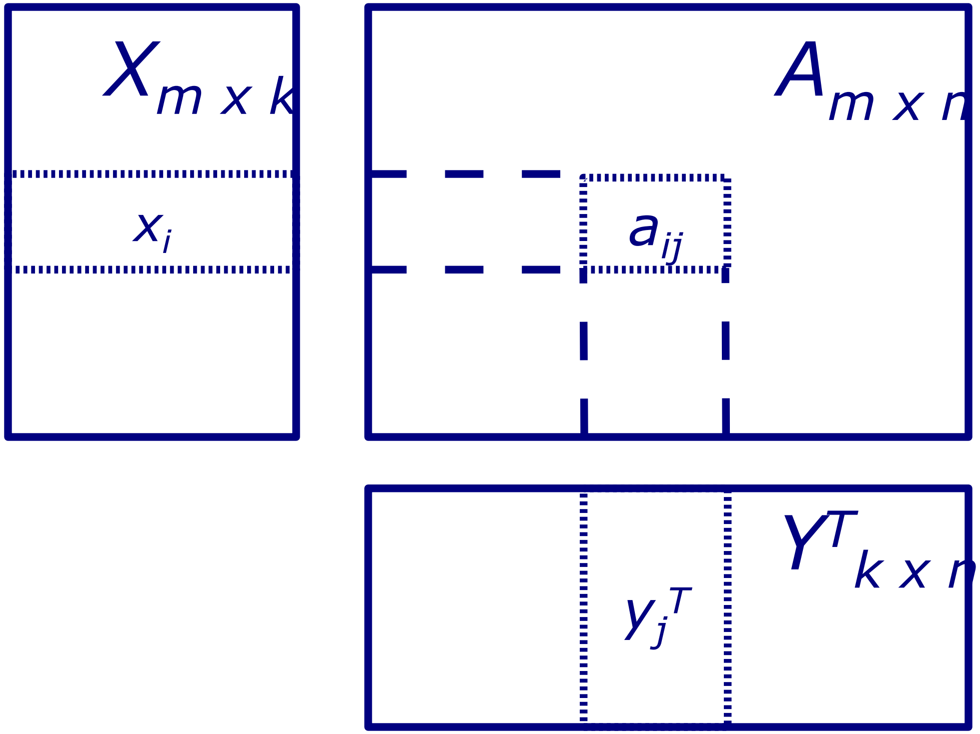
```
currentModel ← initModel()
loop
    wait(Δ)
    p ← selectPeer()
    sendModel(p, currentModel)

procedure ONRECEIVEMODEL(m)
    m.updateModel(x, y)
    currentModel ← m
```



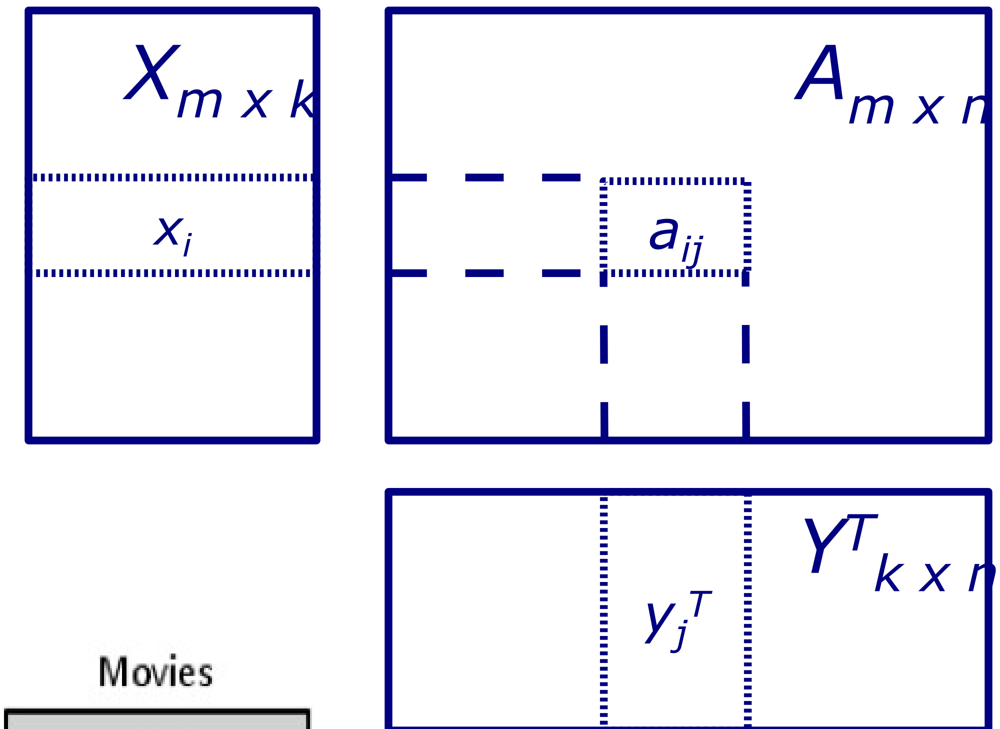
Low-rank Matrix Decomposition

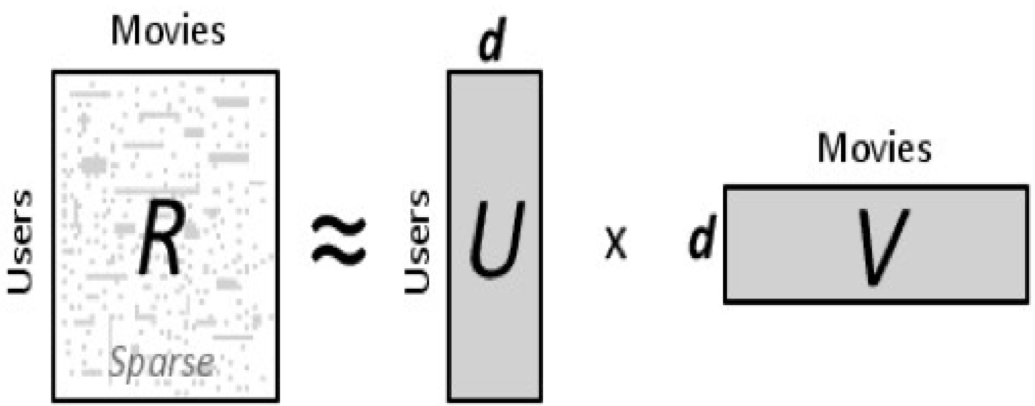
- Given a matrix A
- Looking for matrices X, Y that $XY^T \sim A$
- $k \ll \min(n,m)$
- Data compression



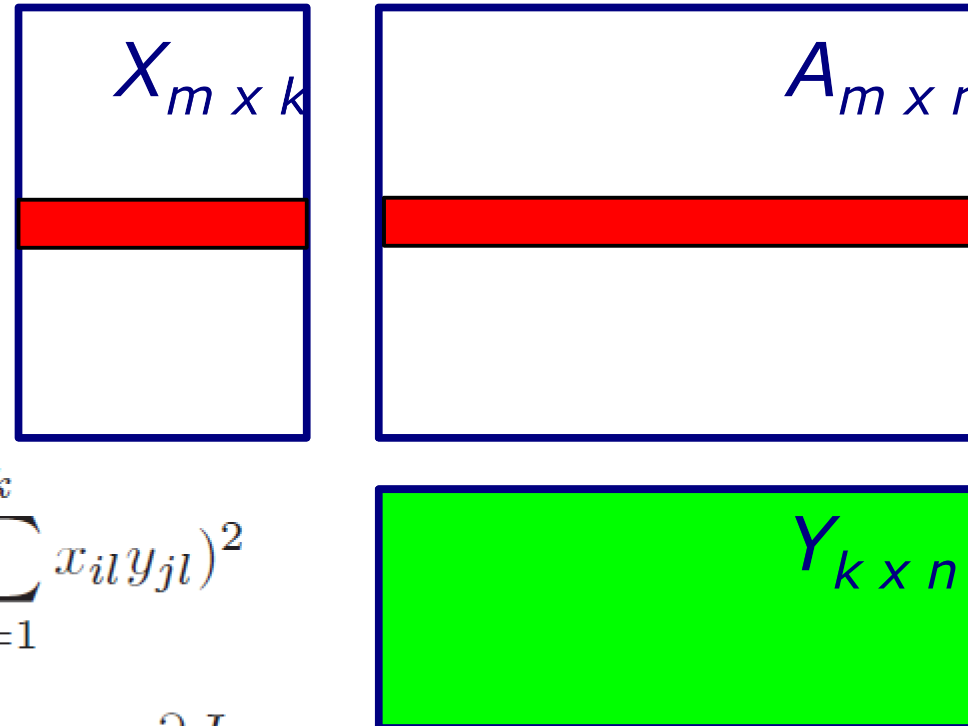
Low-rank Matrix Decomposition

- Given a matrix A
- Looking for matrices X, Y that $XY^T \sim A$
- $k \ll \min(n,m)$
- Data compression





Low-rank Matrix Decomposition



$$J(X,Y) = \frac{1}{2}\|A - XY^T\|_F^2$$

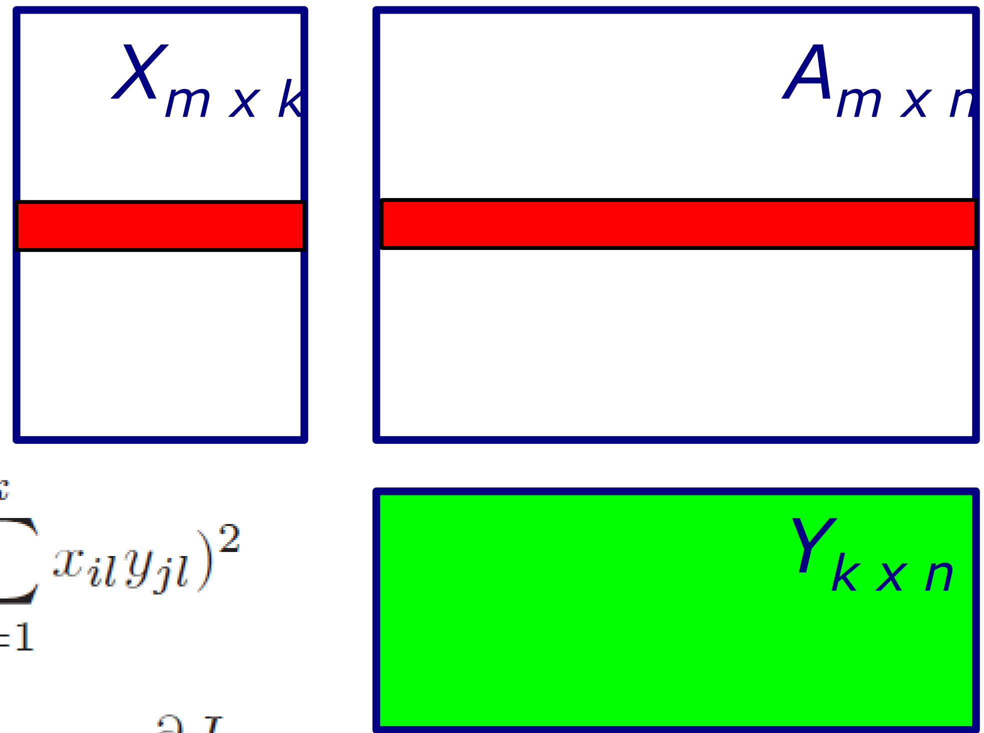
$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(a_{ij} - \sum_{l=1}^{k} x_{il}y_{jl}\right)^2$$

$$\frac{\partial J}{\partial X} = (XY^T - A)Y, \qquad \frac{\partial J}{\partial Y} = (YX^T - A^T)X$$

SGD: $w_{t+1} = w_t - \eta_t \nabla_w E_w$

Low-rank Matrix Decomposition

- User related data:
  a row of X and A
- Random walk: Y



$$J(X,Y) = \frac{1}{2}\|A - XY^T\|_F^2$$

$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij} - \sum_{l=1}^{k} x_{il}y_{jl})^2$$

$$\frac{\partial J}{\partial X} = (XY^T - A)Y, \qquad \frac{\partial J}{\partial Y} = (YX^T - A^T)X$$

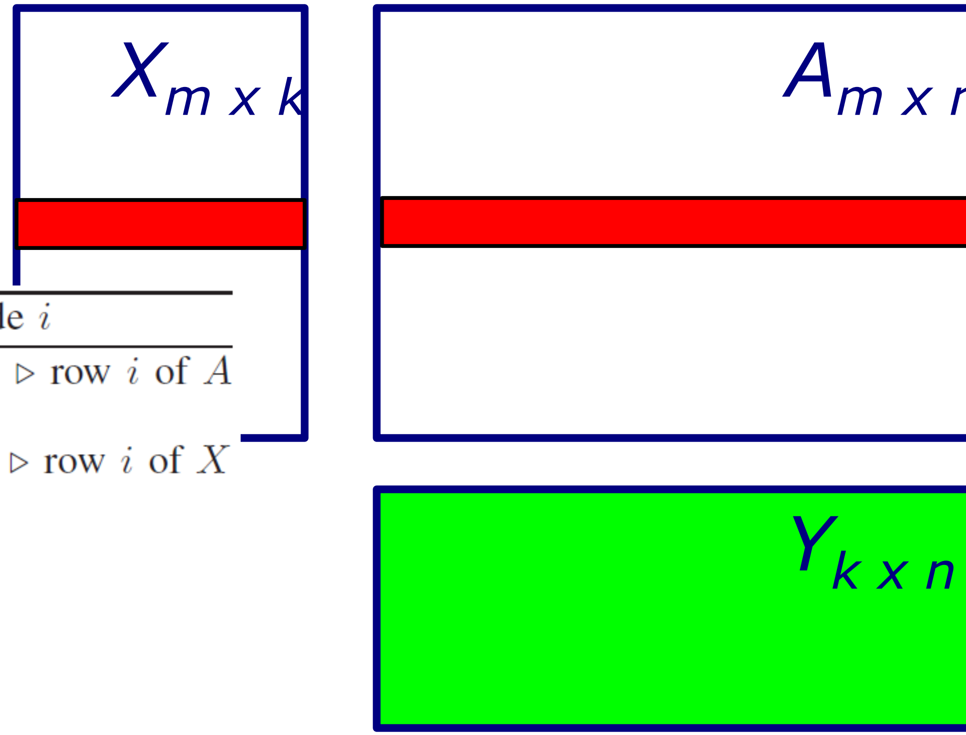
SGD: $w_{t+1} = w_t - \eta_t \nabla_w E_w$

Research Group on Artificial Intelligence

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS

Low-rank Matrix Decomposition

- User related data:
  a row of X and A
- Random walk: Y



Algorithm 1 P2P low–rank factorization at node i

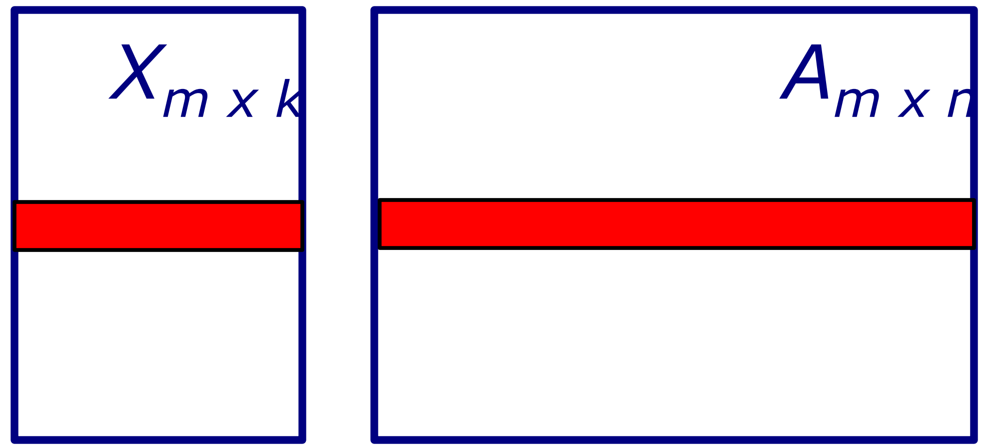
```
a_i                                  ▷ row i of A
initialize Y
initialize x_i                       ▷ row i of X
loop
    wait(Δ)
    p ← selectPeer()
    send Y to p
end loop

procedure ONRECEIVEY(Ỹ)
    Y ← Ỹ
    (Y, x_i) ← update(Y, x_i, a_i)
end procedure
```

Low-rank Matrix Decomposition

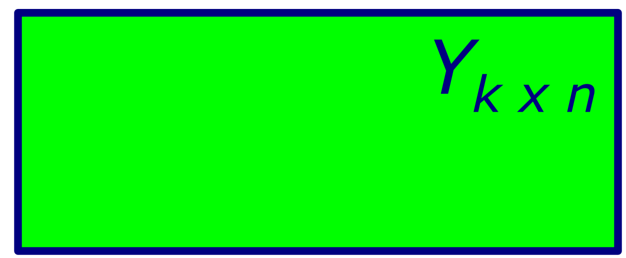
Algorithm 1 P2P low–rank factorization at node i

```
a_i                                   ▷ row i of A
initialize Y
initialize x_i                        ▷ row i of X
loop
    wait(Δ)
    p ← selectPeer()
    send Y to p
end loop
procedure ONRECEIVEY(Ỹ)
    Y ← Ỹ
    (Y, x_i) ← update(Y, x_i, a_i)
end procedure
```



Algorithm 2 rank-k update at node i

1: η ▷ learning rate
2: **procedure** UPDATE(Y, x_i, a_i)
3: $\quad \text{err} \leftarrow a_i - x_i Y^T$
4: $\quad x'_i \leftarrow x_i + \eta \cdot \text{err} \cdot Y$
5: $\quad Y' \leftarrow Y + \eta \cdot \text{err}^T \cdot x_i$
6: $\quad$ **return** (Y', x'_i)
7: **end procedure**



$$\frac{\partial J}{\partial X} = (XY^T - A)Y, \qquad \frac{\partial J}{\partial Y} = (YX^T - A^T)X$$

Singular Value Decomposition

- Columns of X and Y should have the same directions as singular vectors.
- We use:

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

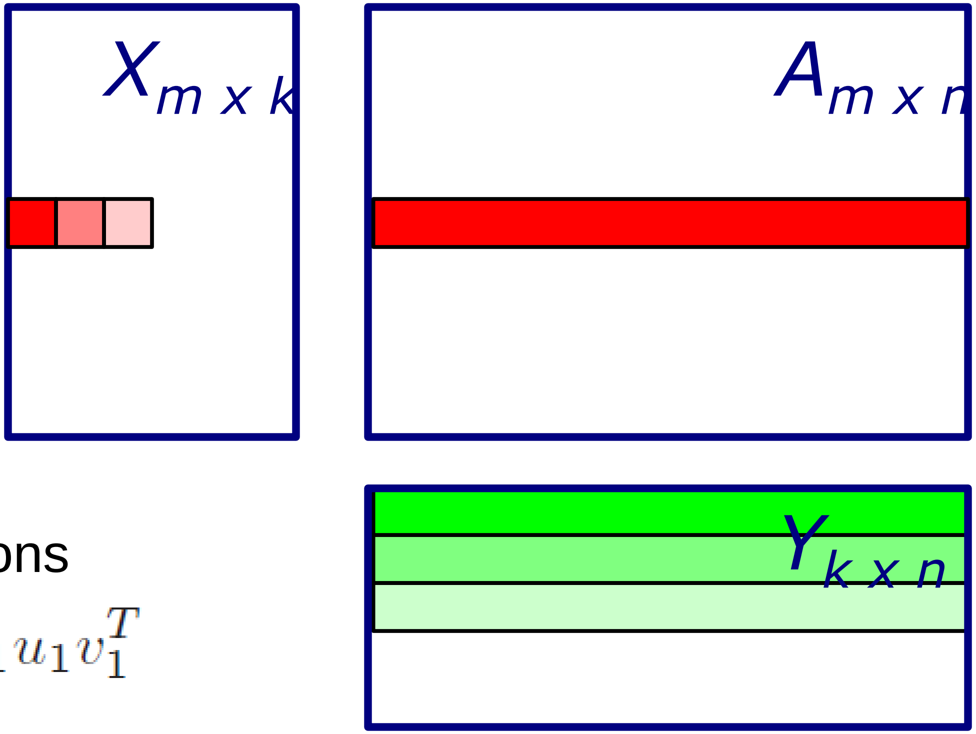
$$X^* = U_k \Sigma_U, \quad Y^* = V_k \Sigma_V$$

- Compute rank-1 approximations

$$X_1 Y_1^T = X^* {Y^*}^T = \sigma_1 u_1 v_1^T$$

$$A - X_1 Y_1^T = \sum_{i=2}^{r} \sigma_i u_i v_i^T$$

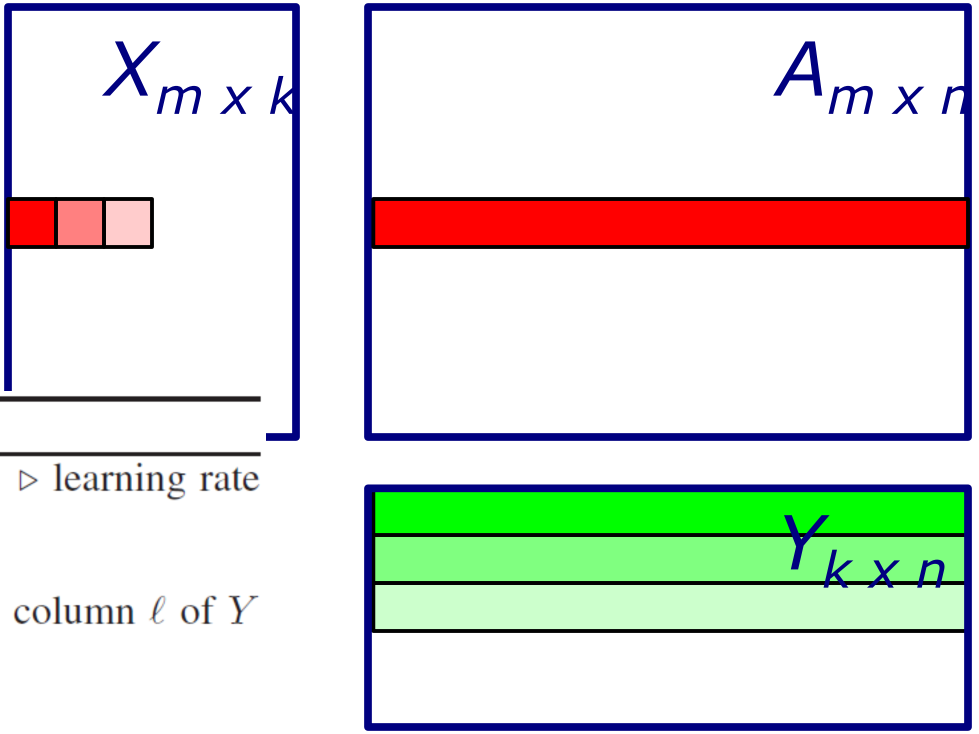
sequentially



Singular Value Decomposition

- Columns of X and Y should have the same directions as singular vectors.
- We use:

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

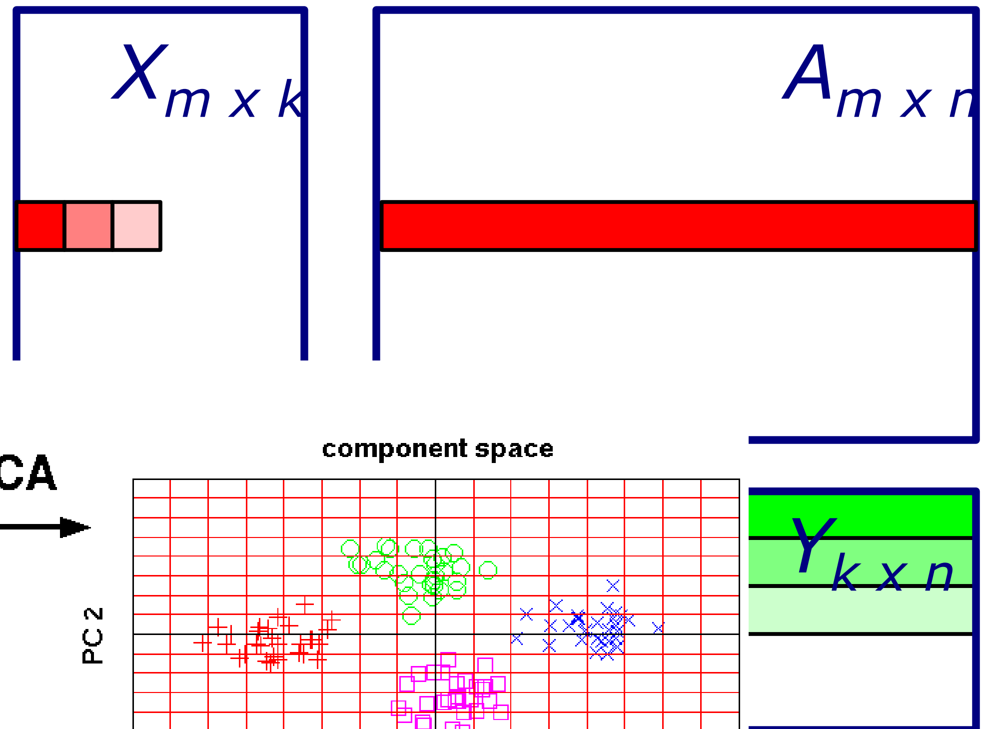


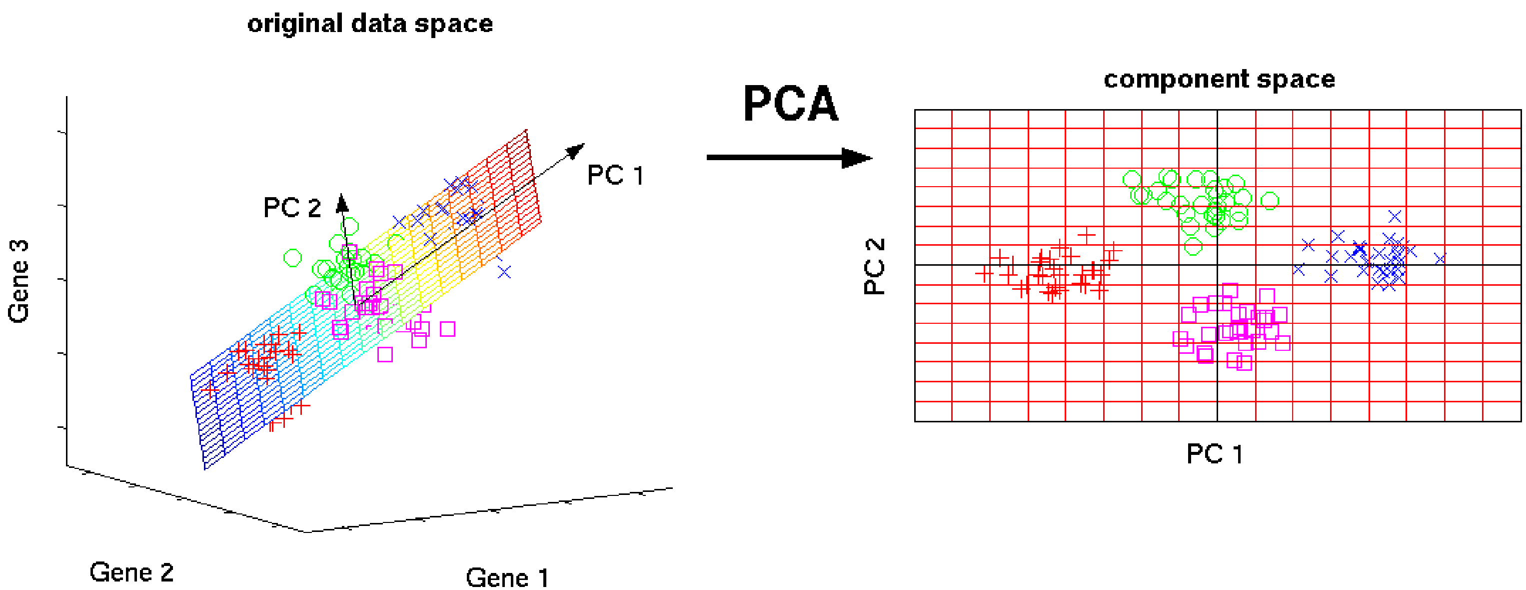
Algorithm 3 rank-k SVD update at node i

```
η                                          ▷ learning rate
procedure UPDATE(Y, x_i, a_i)
    a'_i ← a_i
    for ℓ = 1 to k do                      ▷ y_ℓ : column ℓ of Y
        err ← a'_i − x_iℓ · y_ℓ^T
        x'_iℓ ← x_iℓ + η · err · y_ℓ
        y'_ℓ ← y_ℓ + η · err^T · x_iℓ
        a'_i = a'_i − x_iℓ · y_ℓ^T
    end for
    return (Y', x'_i)
end procedure
```

SVD in Practice

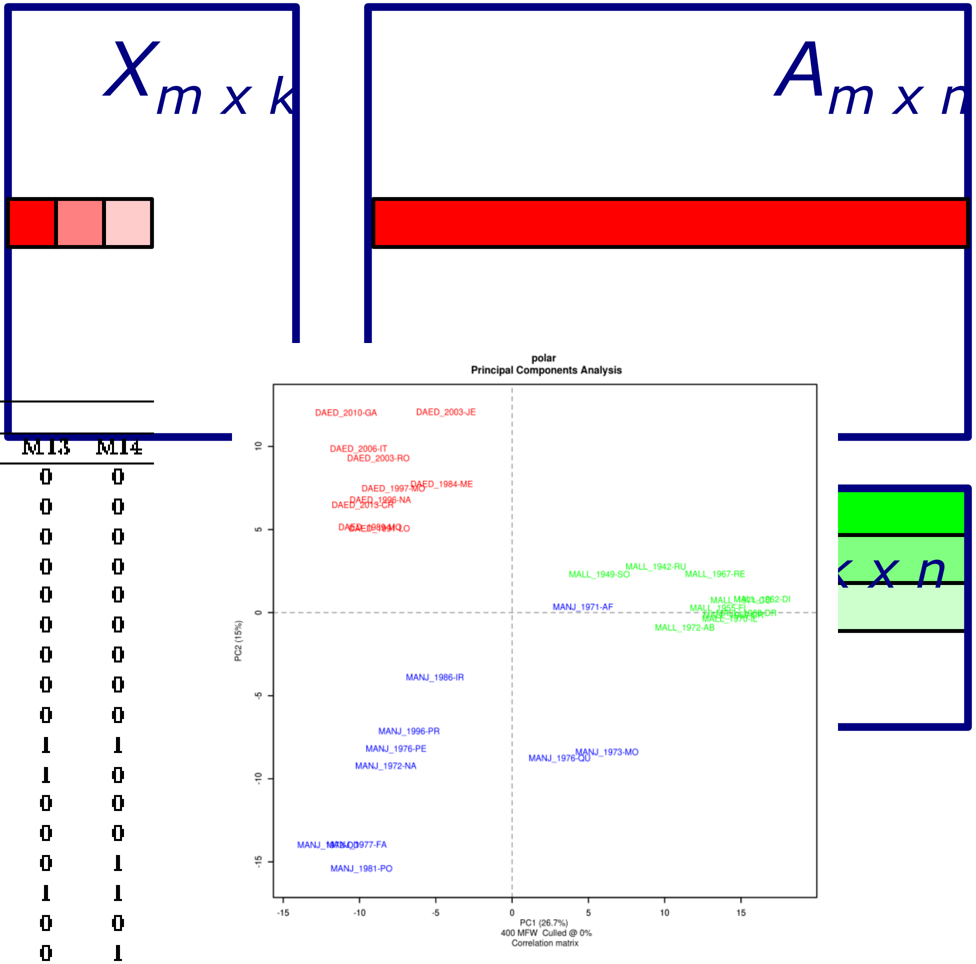
- Dimensionality reduction (PCA)





SVD in Practice

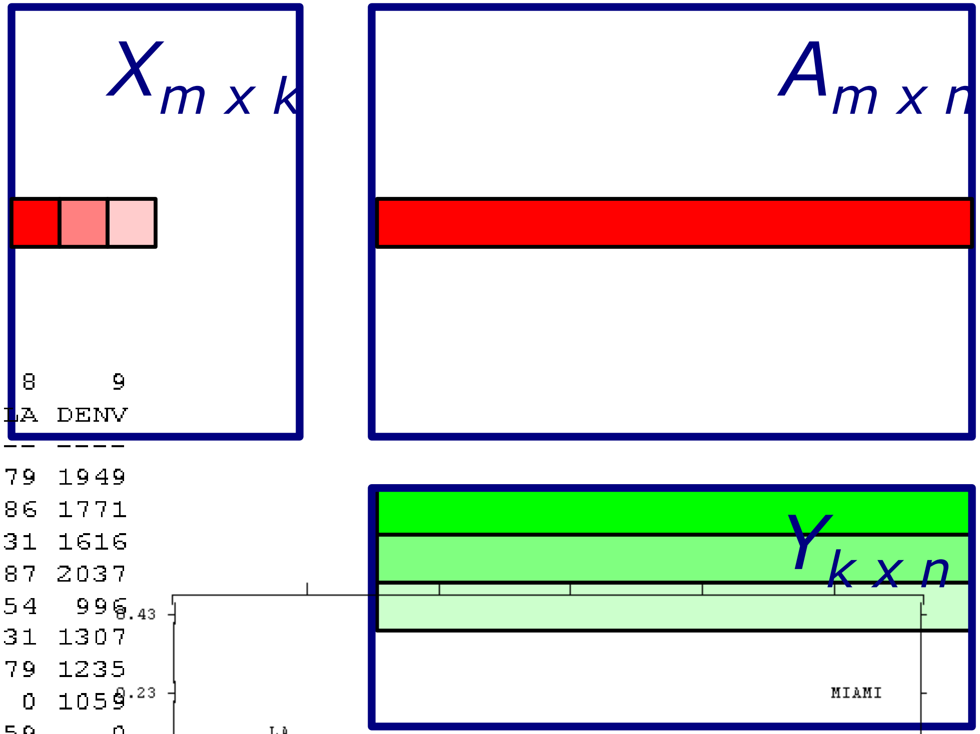
- Topic modeling (LSI)



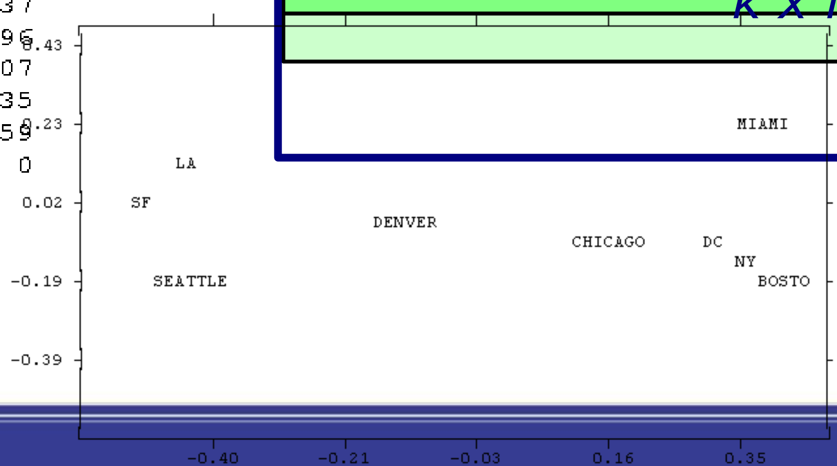
Terms	Documents													
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
abnormalities	0	0	0	0	0	0	0	1	0	1	0	0	0	0
age	1	0	0	0	0	0	0	0	0	0	0	1	0	0
behavior	0	0	0	0	1	1	0	0	0	0	0	0	0	0
blood	0	0	0	0	0	0	0	1	0	0	1	0	0	0
close	0	0	0	0	0	0	1	0	0	0	1	0	0	0
culture	1	1	0	0	0	0	0	1	1	0	0	0	0	0
depressed	1	0	1	1	1	0	0	0	0	0	0	0	0	0
discharge	1	1	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	1	0	1	0	0	0
fast	0	0	0	0	0	0	0	0	0	1	0	1	1	1
generation	0	0	0	0	0	0	0	0	1	0	0	0	1	0
oestrogen	0	0	1	1	0	0	0	0	0	0	0	0	0	0
patients	1	1	0	1	0	0	0	1	0	0	0	0	0	0
pressure	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rats	0	0	0	0	0	0	0	0	0	0	0	0	1	1
respect	0	0	0	0	0	0	0	1	0	0	0	1	0	0
rise	0	0	0	1	0	0	0	0	0	0	0	0	0	1
study	1	0	1	0	0	0	0	0	1	0	0	0	0	0

SVD in Practice

- Multidimensional Scaling (MDS)

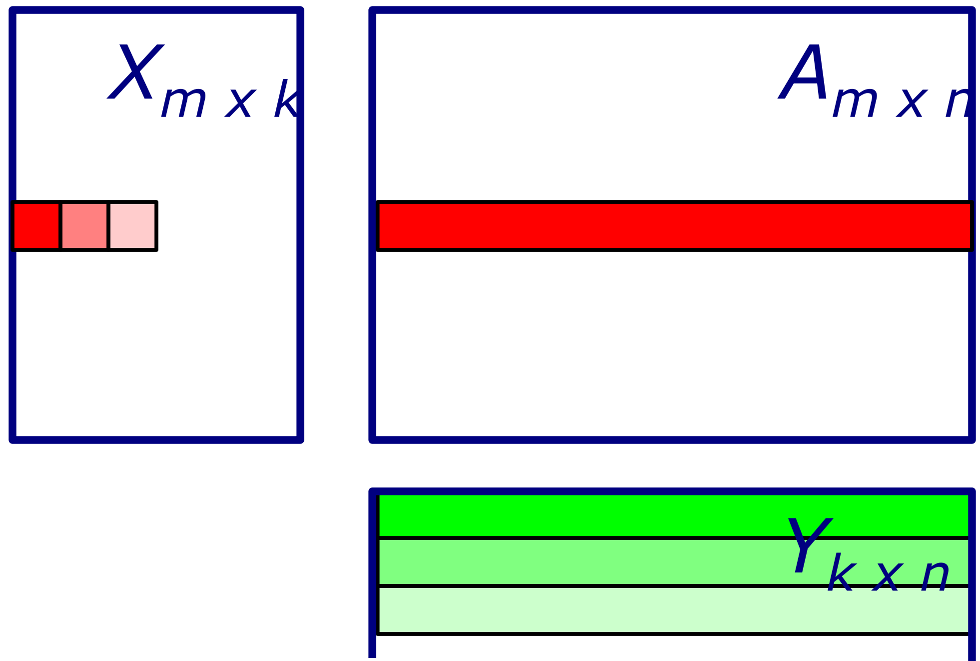


		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0



SVD in Practice

- Graph Spectral Clustering



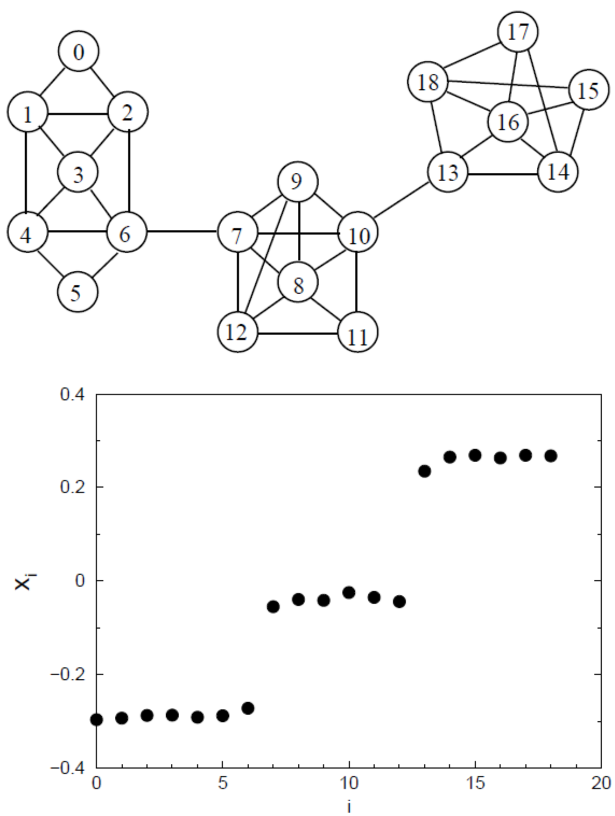


FIG. 18 Basic principle of the spectral algorithm by Capocci et al. (Capocci *et al.*, 2005). The bottom diagram shows the values of the components of the second eigenvector of the right stochastic matrix for the graph drawn on the top. The three plateaus of the eigenvector components correspond to the three evident communities of the graph. Reprinted figures with permission from Ref. (Capocci *et al.*, 2005). ©2005 by Elsevier.

Conclusion

- Machine learning without collecting data
- Prediction without extra communication

- Federated Learning https://research.googleblog.com/2017/04/federated-learning-collaborative.html