Introduction to Statistical Learning and Machine Learning

Chap13 - Semi-supervised Learning

Chap12 -
Semi-supervised Learning

Books

Optional subtitle

Introduction to Semi-Supervised Learning

Xiaojin Zhu and Andrew B. Goldberg
University of Wisconsin, Madison

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING #6

MORGAN & CLAYPOOL PUBLISHERS

Contents

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student is on vacation

while it may be dangerous to obtain labelled data.

Optional subtitle

while it may be dangerous to obtain labelled data.

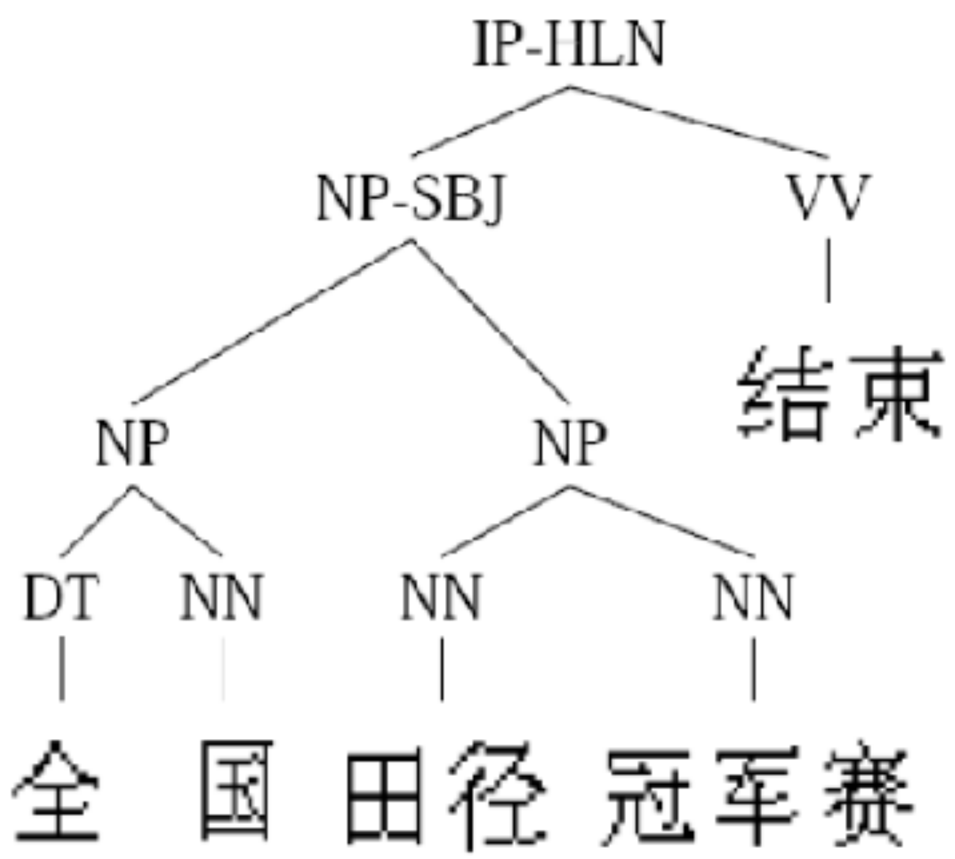
Optional subtitle

Examples of hard-to-get labels

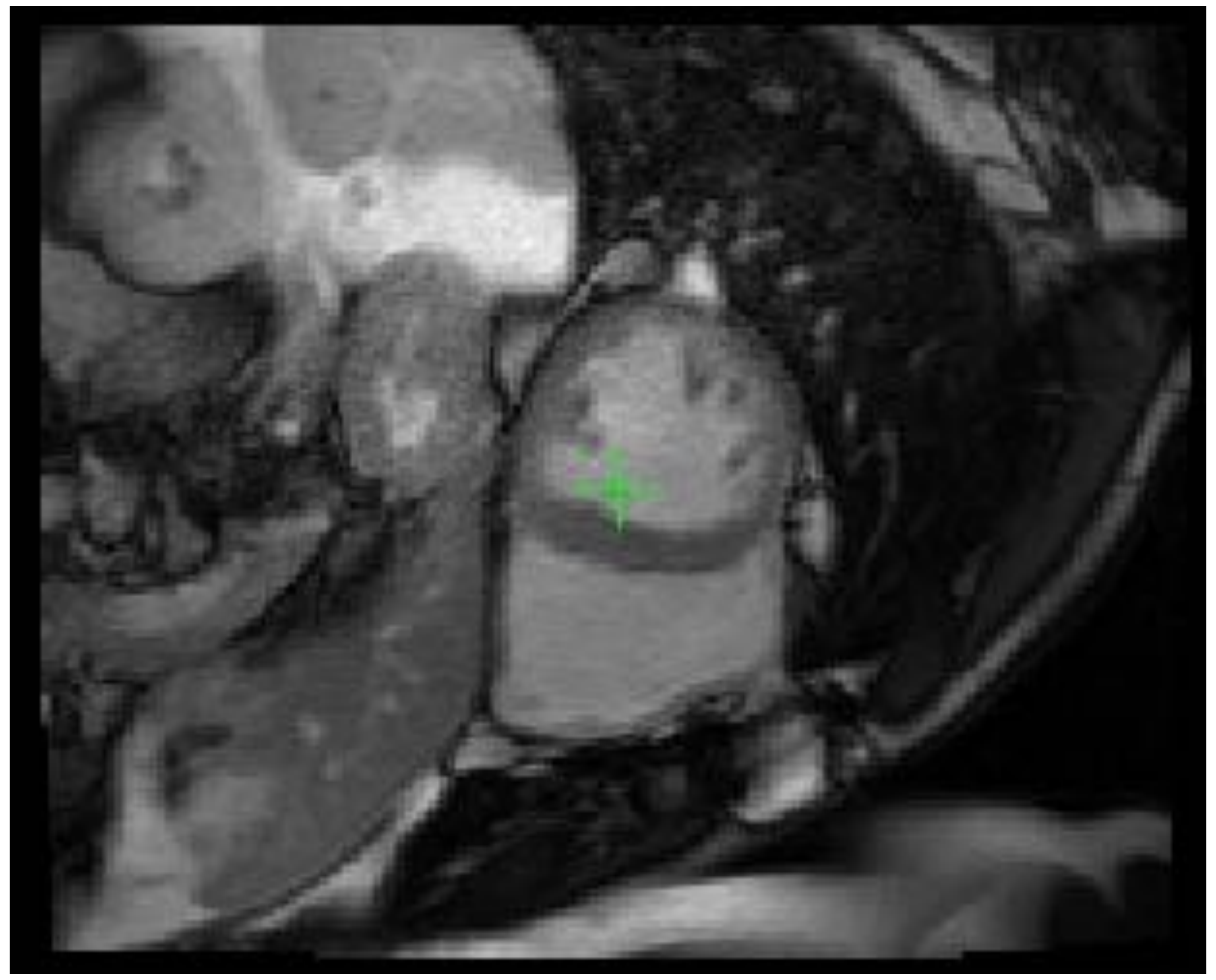
Optional subtitle

Task: natural language parsing

- Penn Chinese Treebank
- 2 years for 4000 sentences



"The National Track and Field Championship has finished."

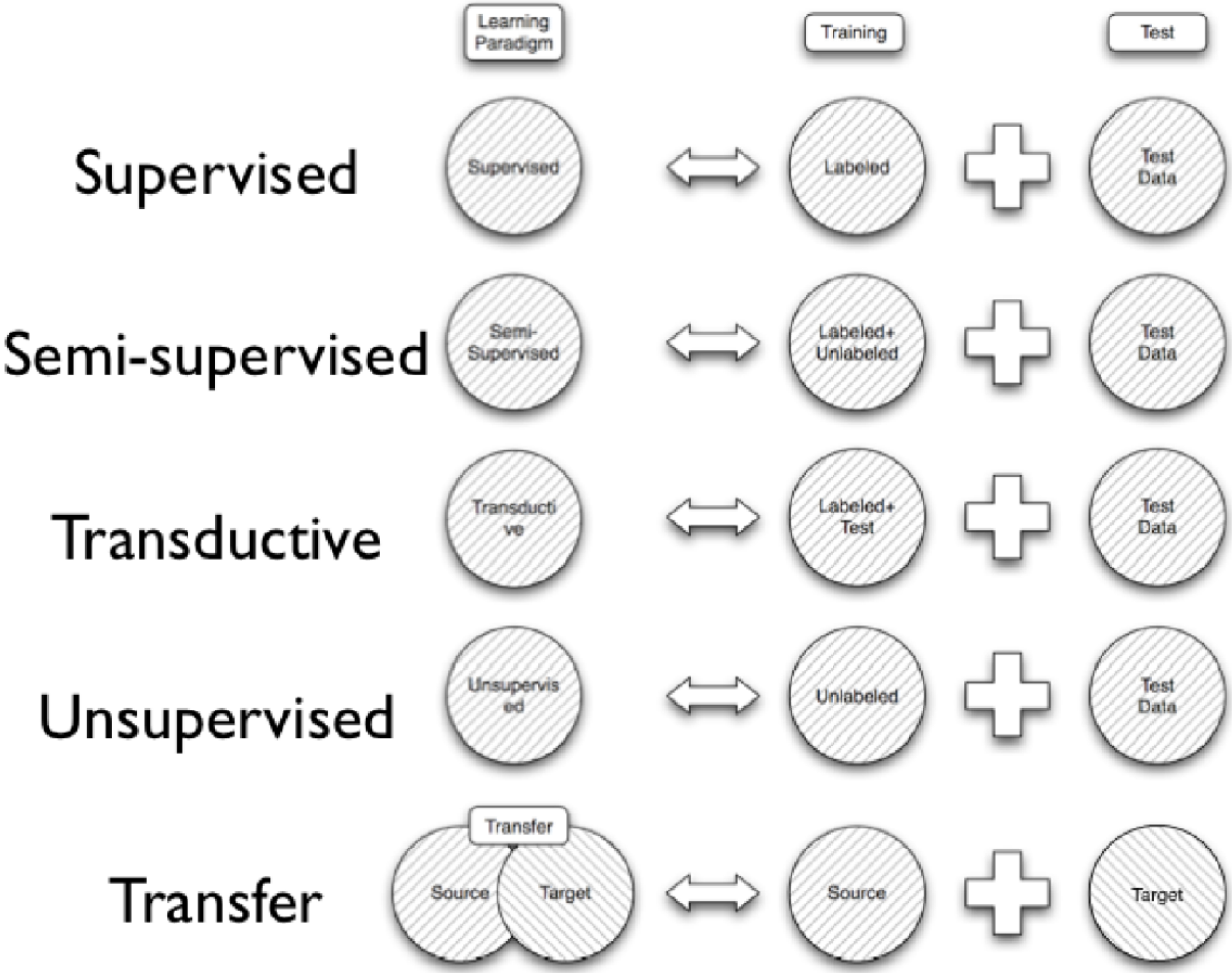


Notations

Optional subtitle

- input instance x, label y
- learner $f : \mathcal{X} \mapsto \mathcal{Y}$
- labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$
- unlabeled data $X_u = \{x_{l+1:n}\}$, available during training
- usually $l \ll n$
- test data $X_{test} = \{x_{n+1:}\}$, not available during training

Learning paradigms



Why the name

Optional subtitle

supervised learning (classification, regression) $\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{test}\}$

transductive classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

semi-supervised clustering $\{x_{1:n}, \text{must-, cannot-links}\}$

$\updownarrow$

unsupervised learning (clustering) $\{x_{1:n}\}$

Why the name

Optional subtitle

supervised learning (classification, regression) $\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{test}\}$

transductive classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

semi-supervised clustering $\{x_{1:n}, \text{must-, cannot-links}\}$

$\updownarrow$

unsupervised learning (clustering) $\{x_{1:n}\}$

We will mainly discuss semi-supervised classification.

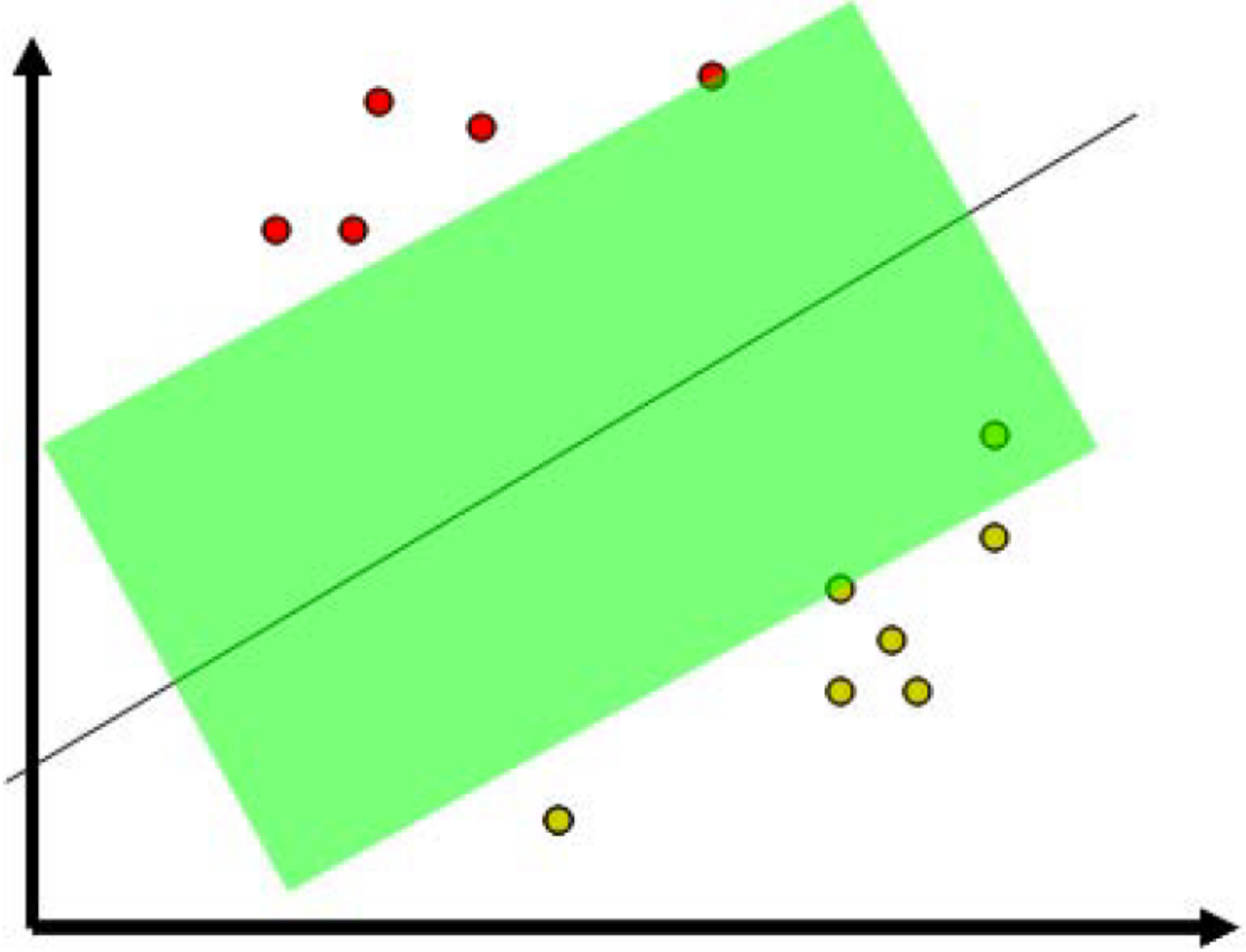
Semi-supervised Learning

Intuition understanding

How semi-supervised learning is helpful?

Optional subtitle

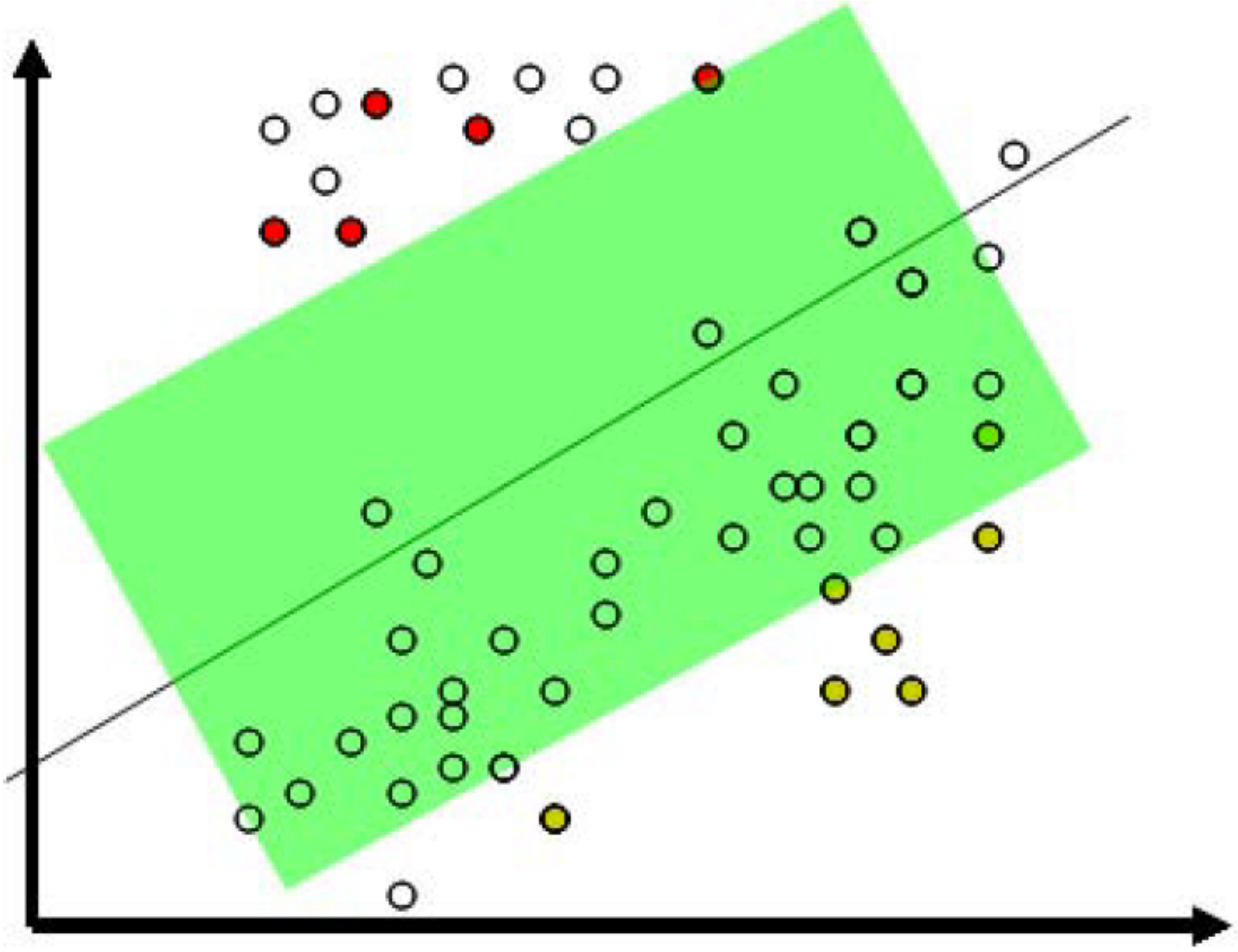
- SVM



How semi-supervised learning is helpful

Optional subtitle

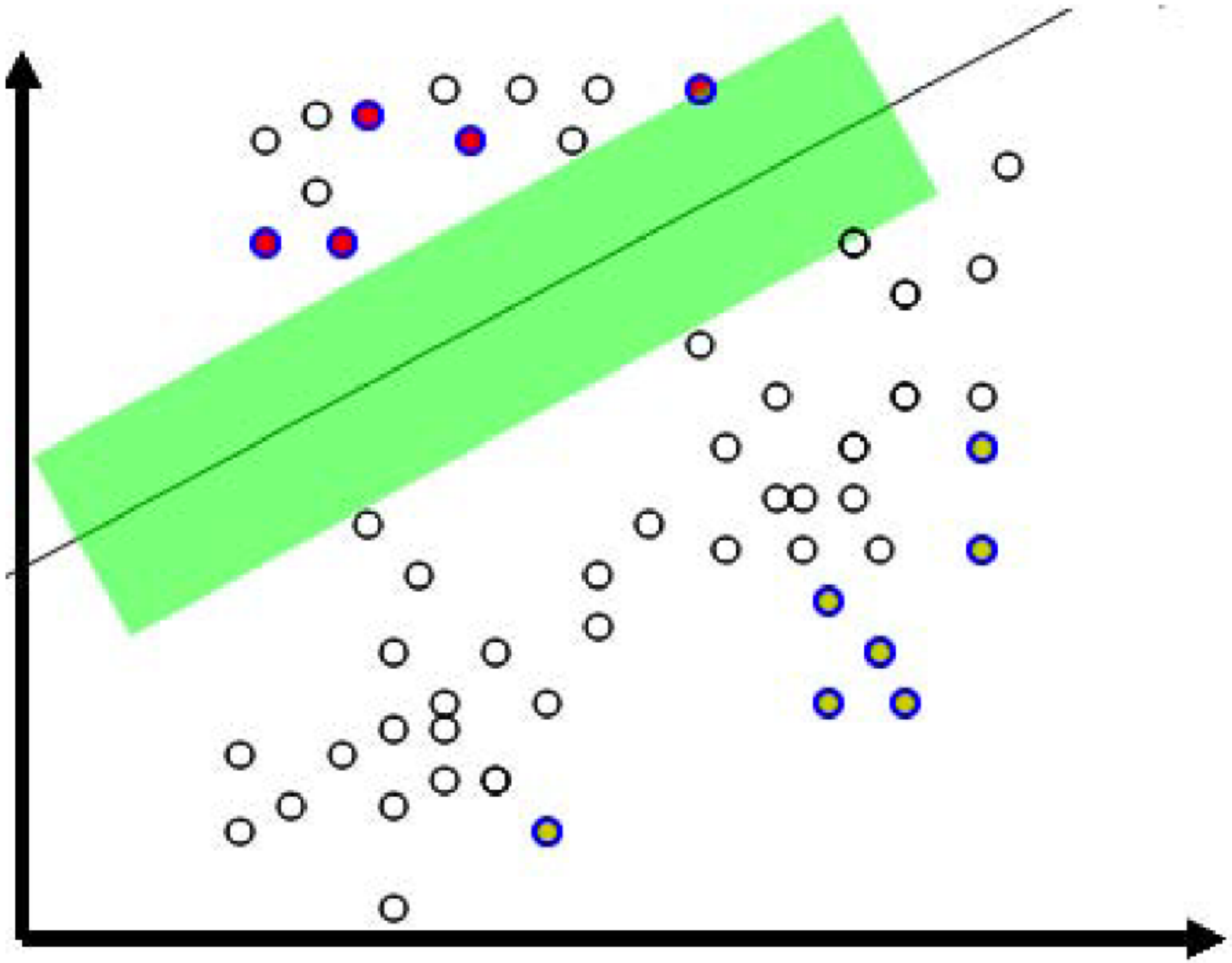
- SVM
- SVM with unlabeled data



How semi-supervised learning is helpful?

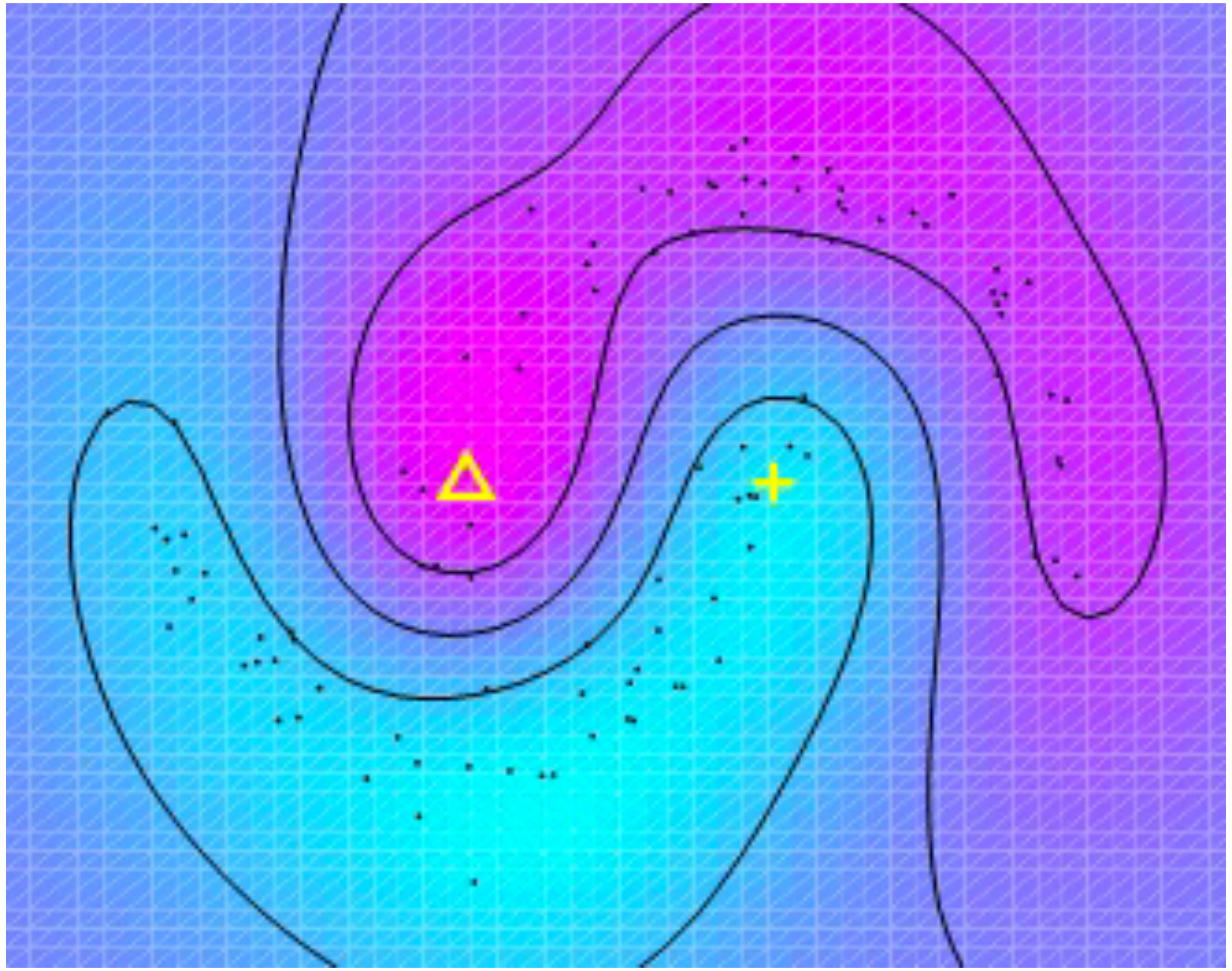
Optional subtitle

- SVM
- SVM with unlabeled data
- Semi-supervised SVM



Two moons

Optional subtitle



There are 2 labeled points (the triangle and the cross) and 100 unlabeled points. The global optimum of S3VM correctly identifies the decision boundary (black line).

This figure comes from Chapelle et al. “Optimization Techniques for Semi-Supervised Support Vector Machines”, JMLR 2007.

Does unlabeled data always help?

Optional subtitle

Does unlabeled data always help?

Optional subtitle

Unfortunately, this is not the case, yet.

Does unlabeled data always help?

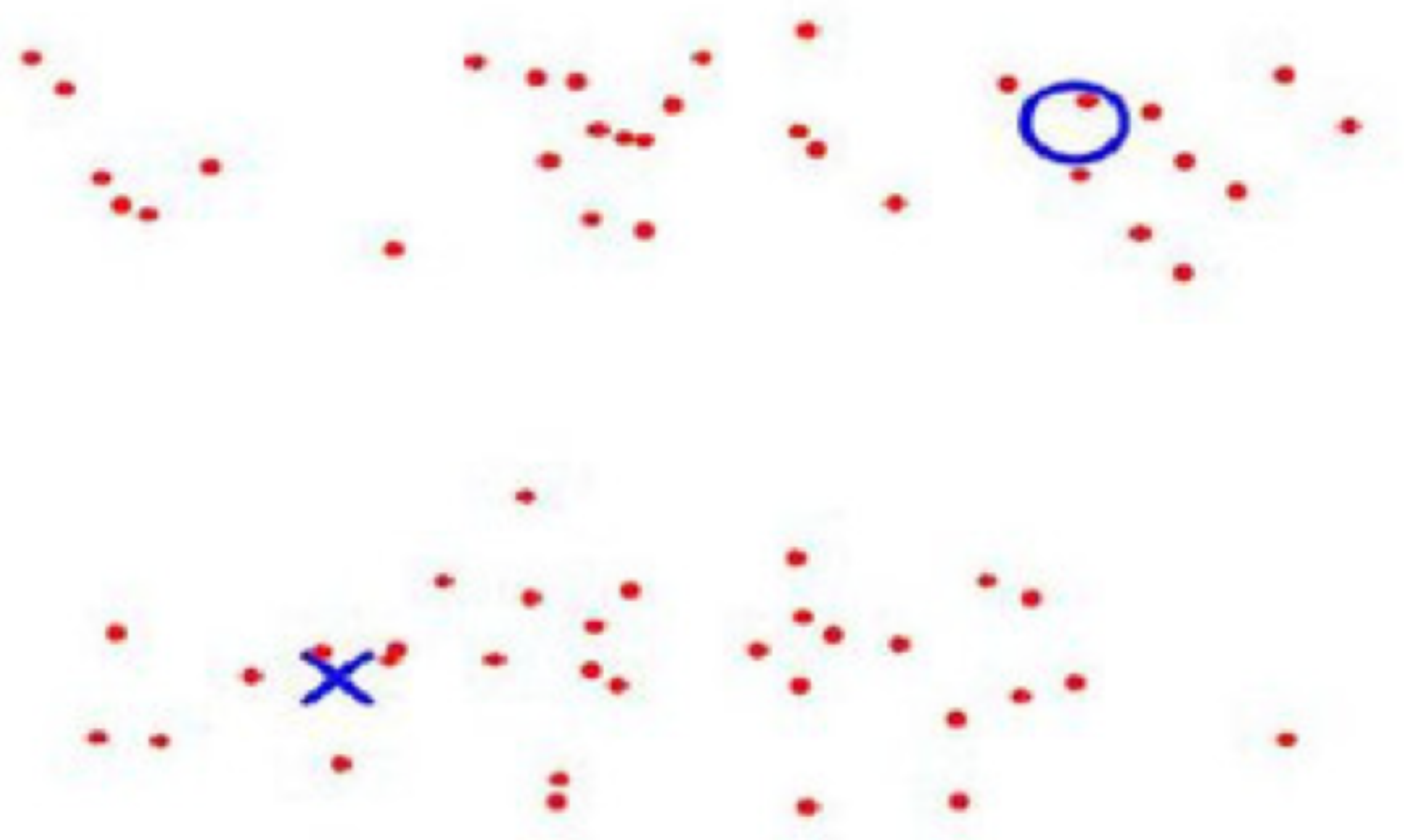
Optional subtitle

Unfortunately, this is not the case, yet.

Not until recently, “**Safe Semi-supervised learning**”,
Yu-Feng Li and Zhi-Hua Zhou. T*owards making unlabeled data never hurt.* IEEE Transactions on Pattern Analysis and Machine Intelligence 2014.

Assumption and Intuitions

Optional subtitle

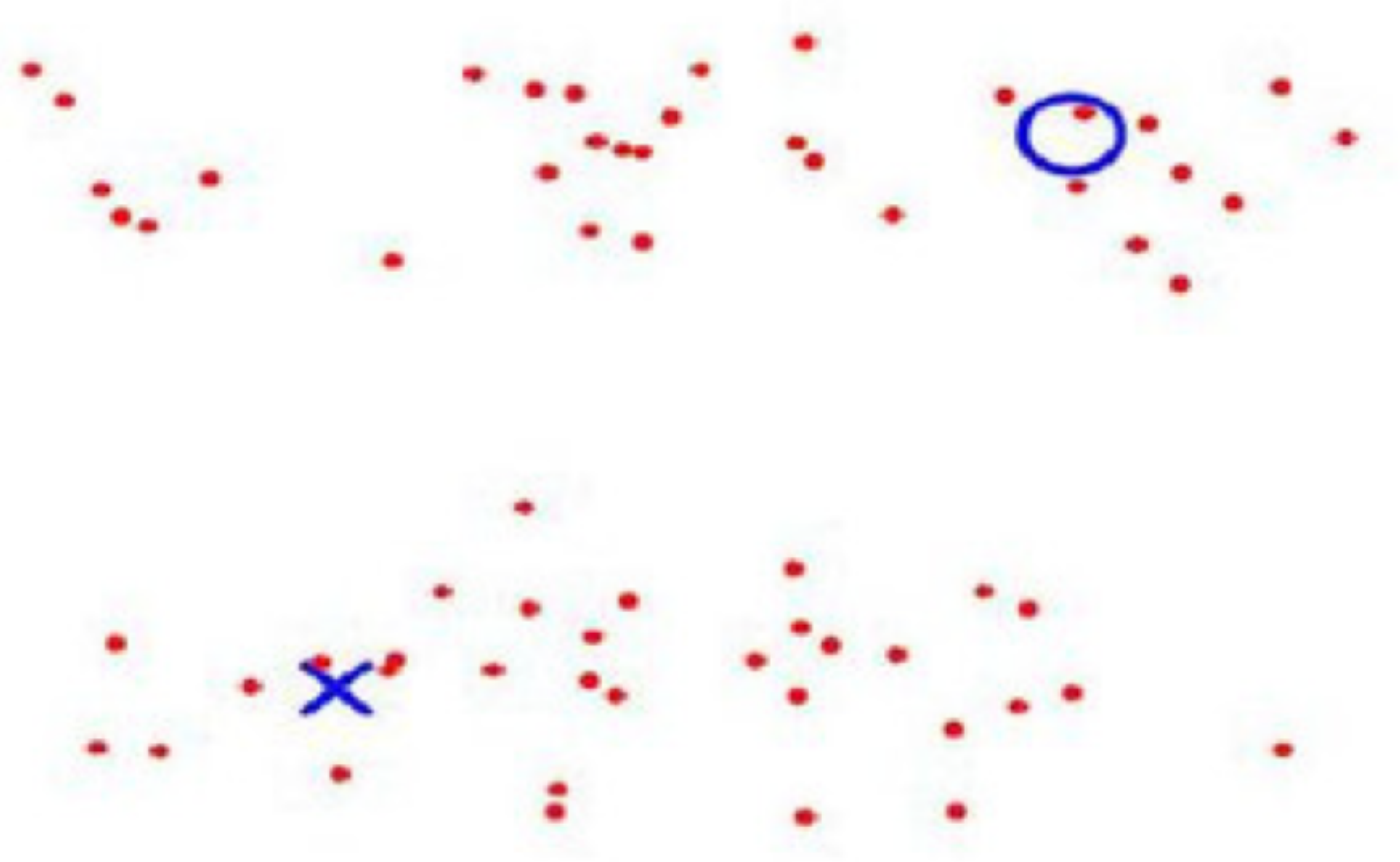


Guess the name?

- Points in the same cluster likely to be of the same class;
- Decision line should lie in a low-density region.

Assumption and Intuitions

Optional subtitle

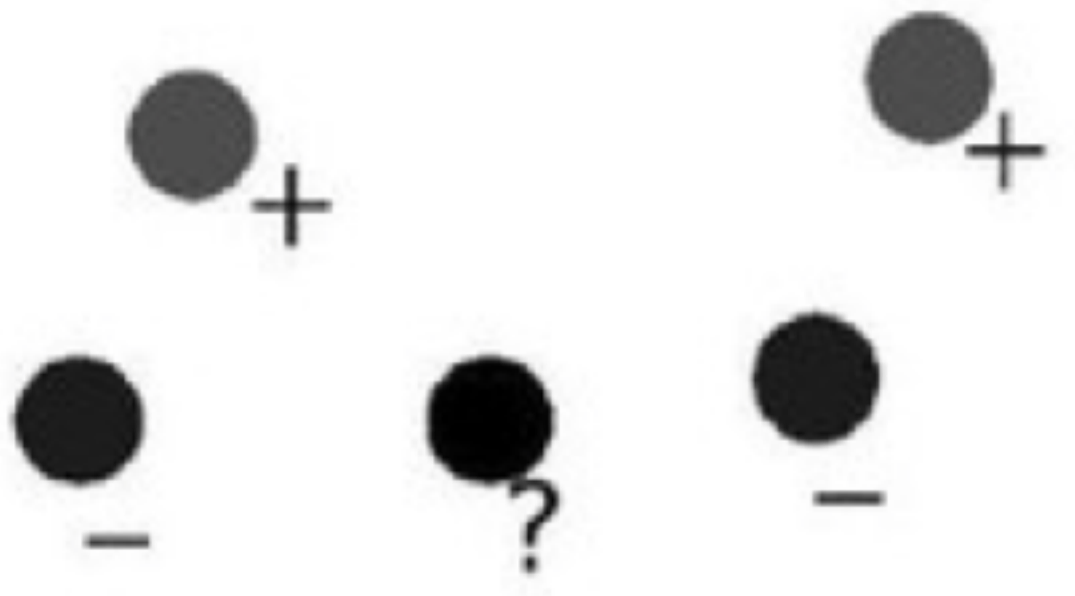


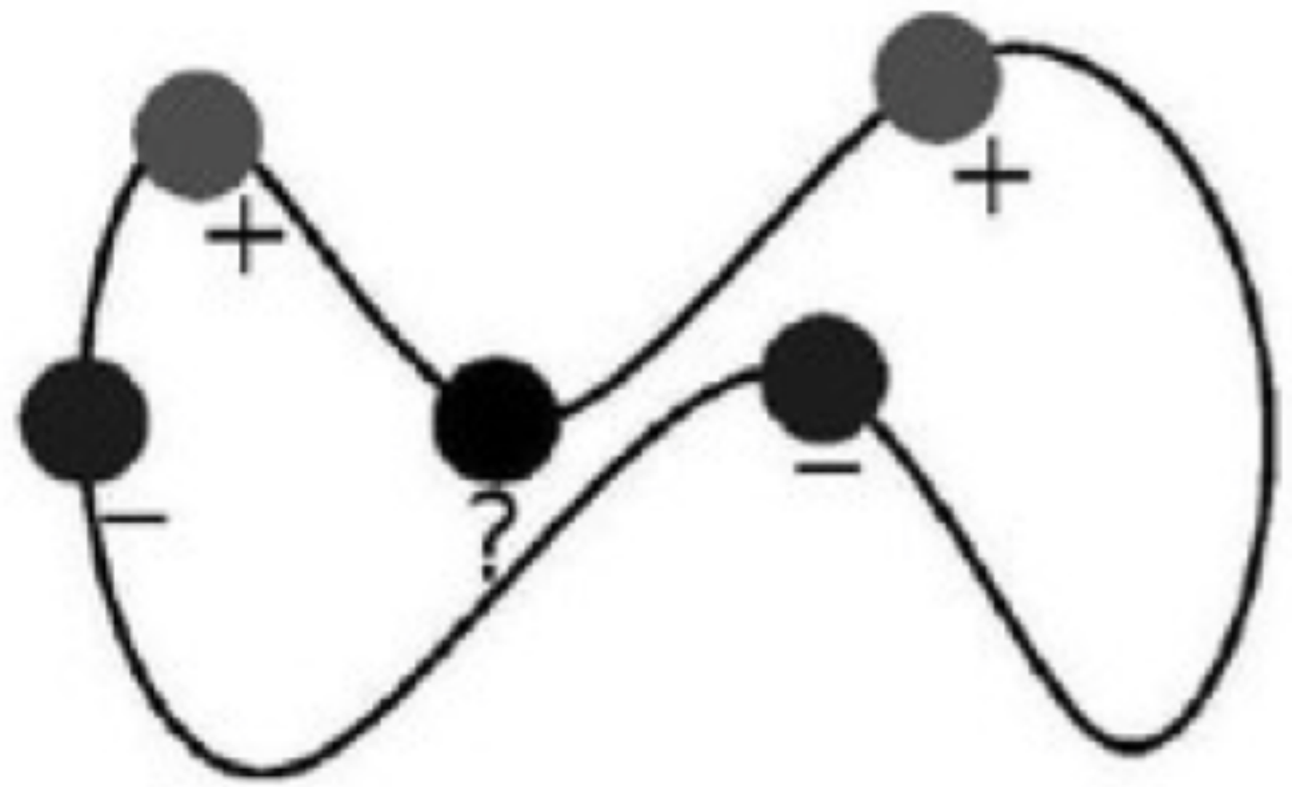
Guess the name?

Cluster Assumption

- Points in the same cluster likely to be of the same class;
- Decision line should lie in a low-density region.

Assumption and Intuitions

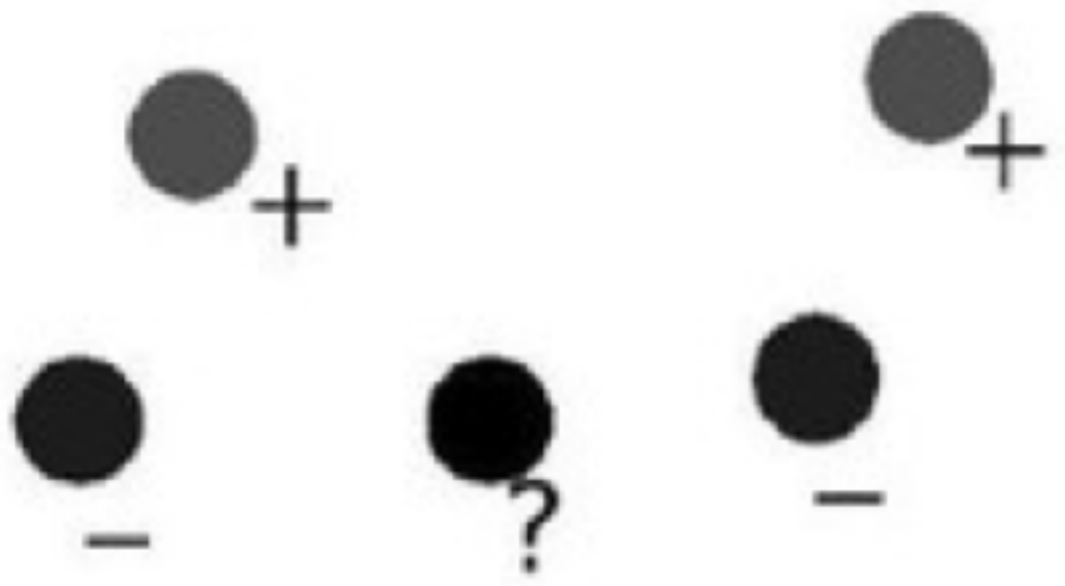


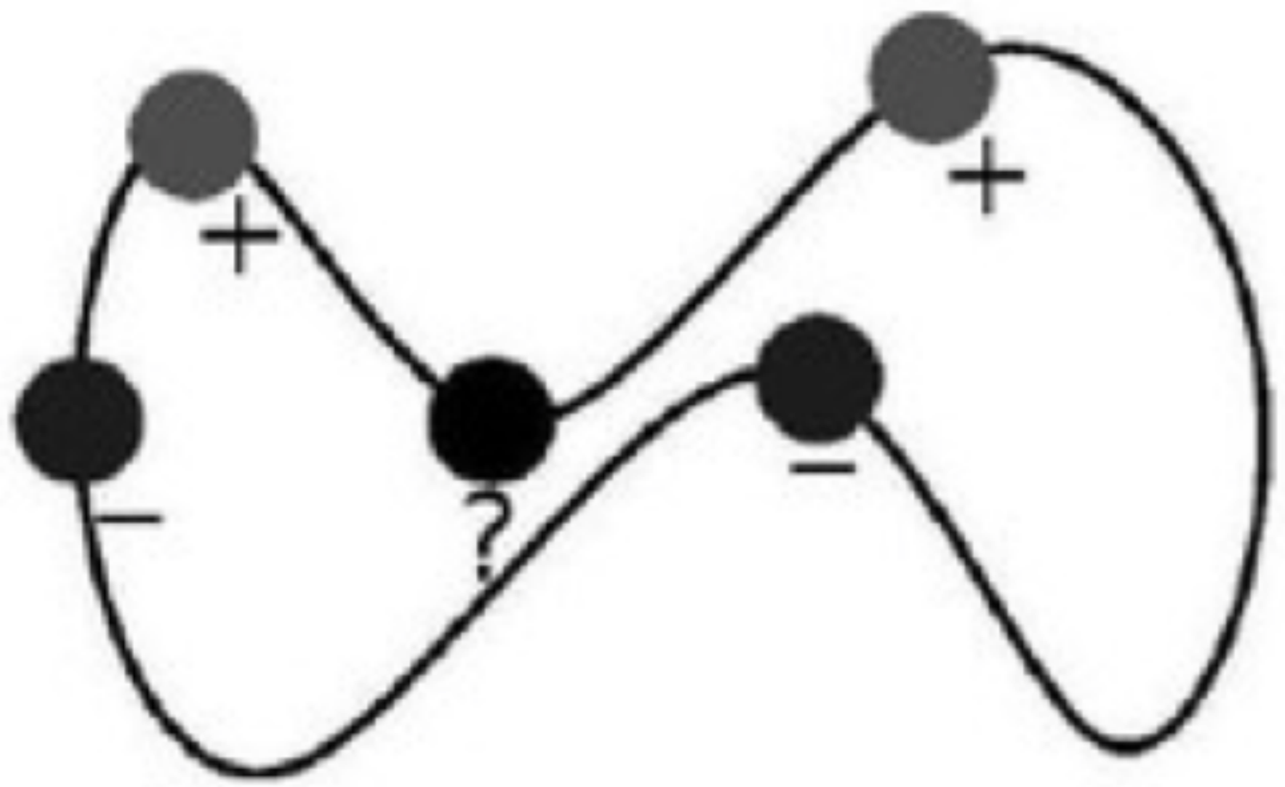


Guess the name?

- The data lie approximately on a manifold of much lower dimension than the input space.
- Swiss Roll

Assumption and Intuitions





Guess the name?

Manifold Assumption

- The data lie approximately on a manifold of much lower dimension than the input space.
- Swiss Roll

Assumption and Intuitions

Optional subtitle

If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

Guess the name?

Points which are close to each other are more likely to share a label. This is also generally assumed in supervised learning and yields a preference for geometrically simple decision boundaries. In the case of semi-supervised learning, the smoothness assumption additionally yields a preference for decision boundaries in low-density regions, so that there are fewer points close to each other but in different classes.

Assumption and Intuitions

Optional subtitle

If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

Guess the name?

Points which are close to each other are more likely to share a label. This is also generally assumed in supervised learning and yields a preference for geometrically simple decision boundaries. In the case of semi-supervised learning, the smoothness assumption additionally yields a preference for decision boundaries in low-density regions, so that there are fewer points close to each other but in different classes.

Smoothness Assumption

Semi-supervised Learning

Key methods of SSL

Methods and Algorithms of SSL

Optional subtitle

- Self-training
- Co-training -Multi-view Algorithm
- Graph-based SVM
- Semi-supervised SVM (S3VM)

Self-training

Self-Training algorithm

Optional subtitle

Assumption

One's own high confidence predictions are correct.

Self-training algorithm:

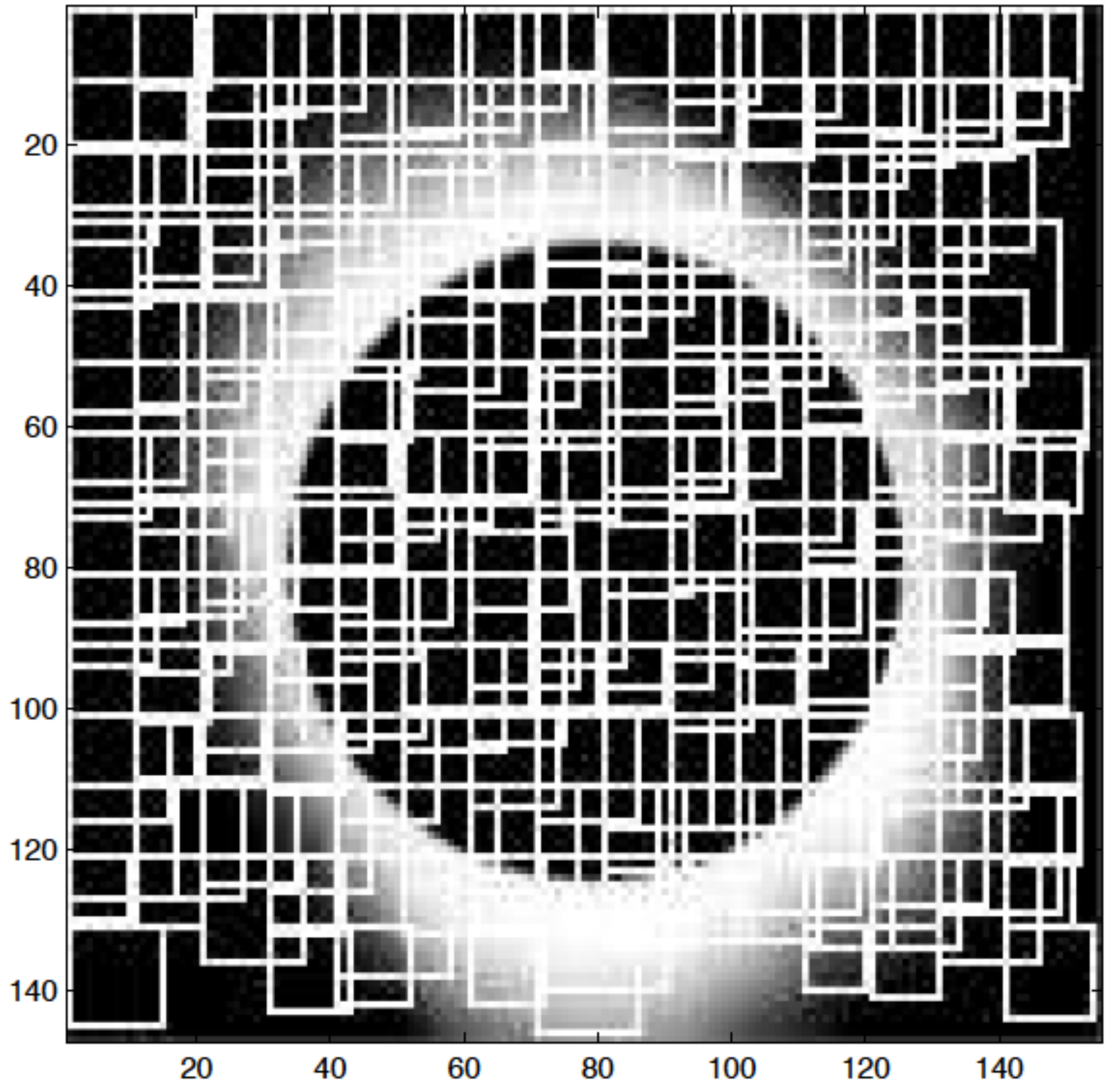
1. Train f from (X_l, Y_l)
2. Predict on $x \in X_u$
3. Add $(x, f(x))$ to labeled data
4. Repeat

Variations in self-training

- Add a few most confident $(x, f(x))$ to labeled data
- Add all $(x, f(x))$ to labeled data
- Add all $(x, f(x))$ to labeled data, weigh each by confidence

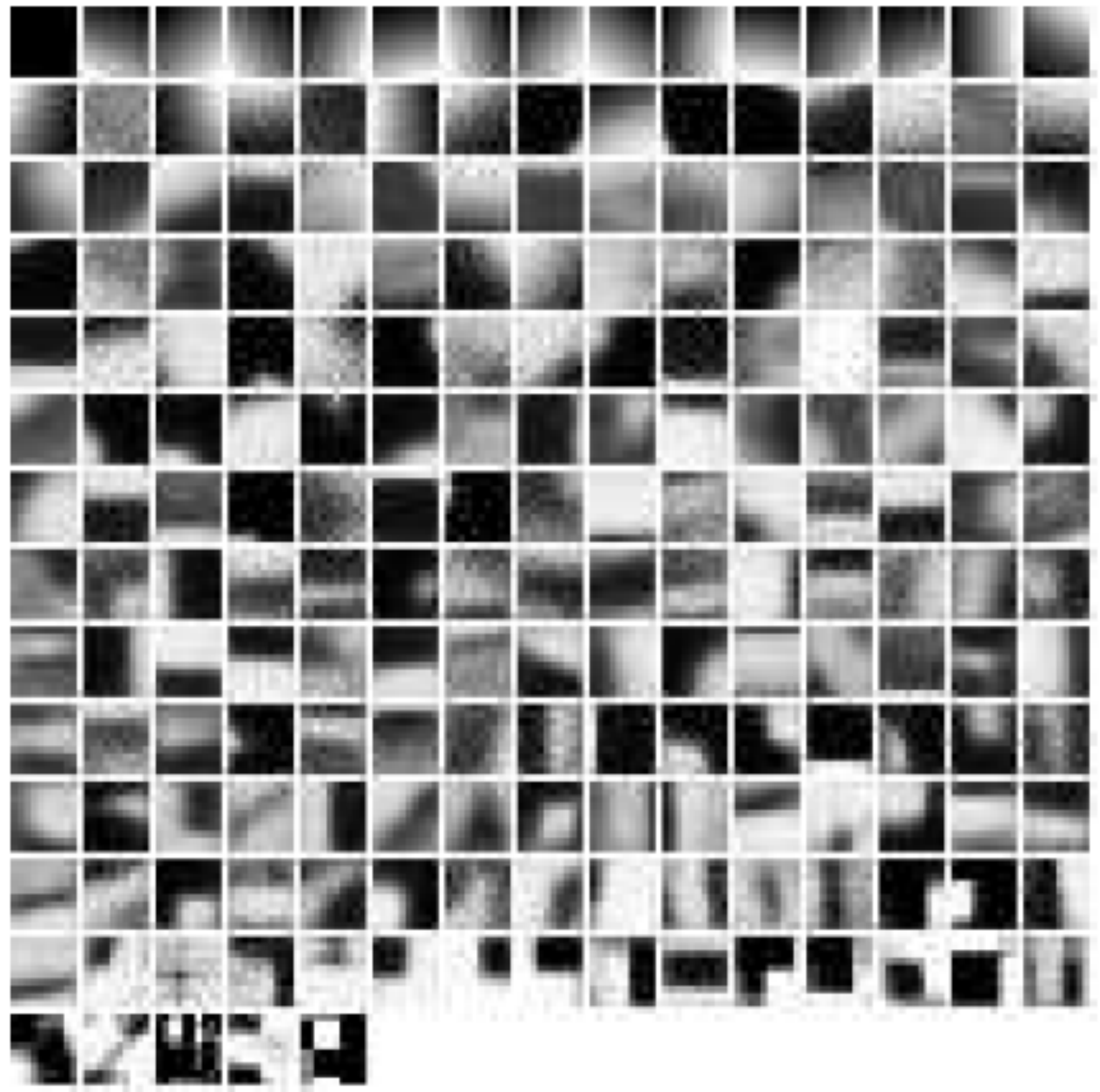
Self-training example: image categorisation

- Each image is divided into small patches
- 10×10 grid, random size in $10 \sim 20$



Self-training example: image categorisation

- All patches are normalized.
- Define a dictionary of 200 'visual words' (cluster centroids) with 200-means clustering on all patches.
- Represent a patch by the index of its closest visual word.



The bag-of-word representation of images

 → 1:0 2:1 3:2 4:2 5:0 6:0 7:0 8:3 9:0 10:3 11:31 12:0 13:0 14:0 15:0 16:9 17:1 18:0 19:0 20:1 21:0 22:0 23:0 24:0 25:6

26:0 27:6 28:0 29:0 30:0 31:1 32:0 33:0 34:0 35:0 36:0 37:0 38:0 39:0 40:0 41:0 42:1 43:0 44:2 45:0 46:0 47:0 48:0 49:3 50:0

51:3 52:0 53:0 54:0 55:1 56:1 57:1 58:1 59:0 60:3 61:1 62:0 63:3 64:0 65:0 66:0 67:0 68:0 69:0 70:0 71:1 72:0 73:2 74:0 75:0

76:0 77:0 78:0 79:0 80:0 81:0 82:0 83:0 84:3 85:1 86:1 87:1 88:2 89:0 90:0 91:0 92:0 93:2 94:0 95:1 96:0 97:1 98:0 99:0 100:0

101:1 102:0 103:0 104:0 105:1 106:0 107:0 108:0 109:0 110:3 111:1 112:0 113:3 114:0 115:0 116:0 117:0 118:3 119:0 120:0

121:1 122:0 123:0 124:0 125:0 126:0 127:3 128:3 129:3 130:4 131:4 132:0 133:0 134:2 135:0 136:0 137:0 138:0 139:0 140:0

141:1 142:0 143:6 144:0 145:2 146:0 147:3 148:0 149:0 150:0 151:0 152:0 153:0 154:1 155:0 156:0 157:3 158:12 159:4 160:0

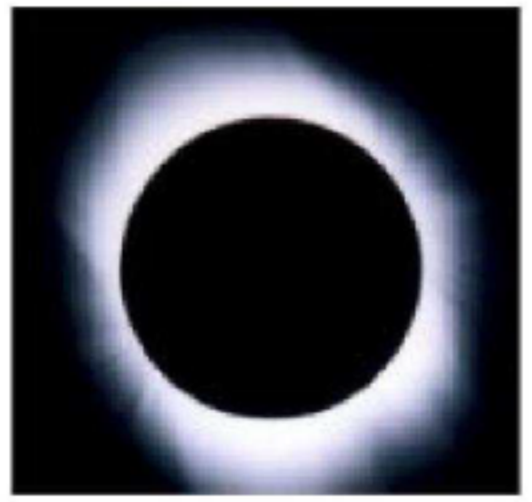
161:1 162:7 163:0 164:3 165:0 166:0 167:0 168:0 169:1 170:3 171:2 172:0 173:1 174:0 175:0 176:2 177:0 178:0 179:1 180:0

181:1 182:2 183:0 184:0 185:2 186:0 187:0 188:0 189:0 190:0 191:0 192:0 193:1 194:2 195:4 196:0 197:0 198:0 199:0 200:0

Self-training example: image categorization

1. Train a naïve Bayes classifier on the two initial labeled images

0.00



1.jpeg

-Inf

14.jpeg

2. Classify unlabeled data, sort by confidence $\log p(y = \text{astronomy} | x)$

-137.13

19.jpeg

-121.93

34.jpeg

-109.89

84.jpeg

-107.91

100.jpeg

-96.98

13.jpeg

. . .

-0.00

91.jpeg

-0.00

94.jpeg

-0.00

95.jpeg

-0.00

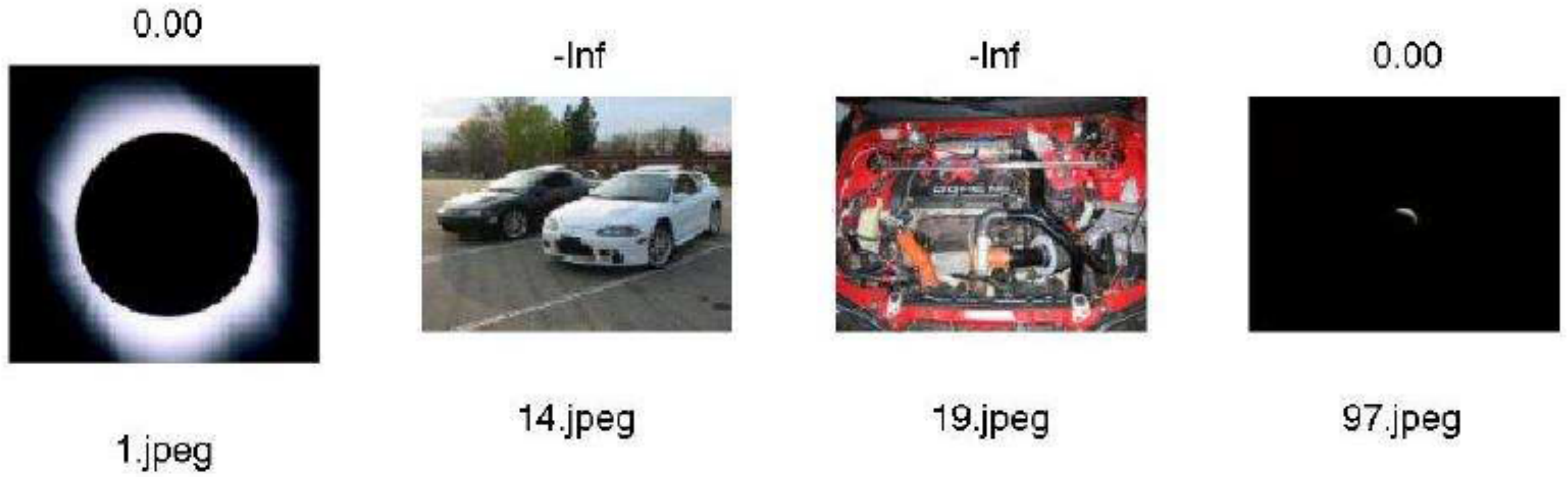
96.jpeg

-0.00

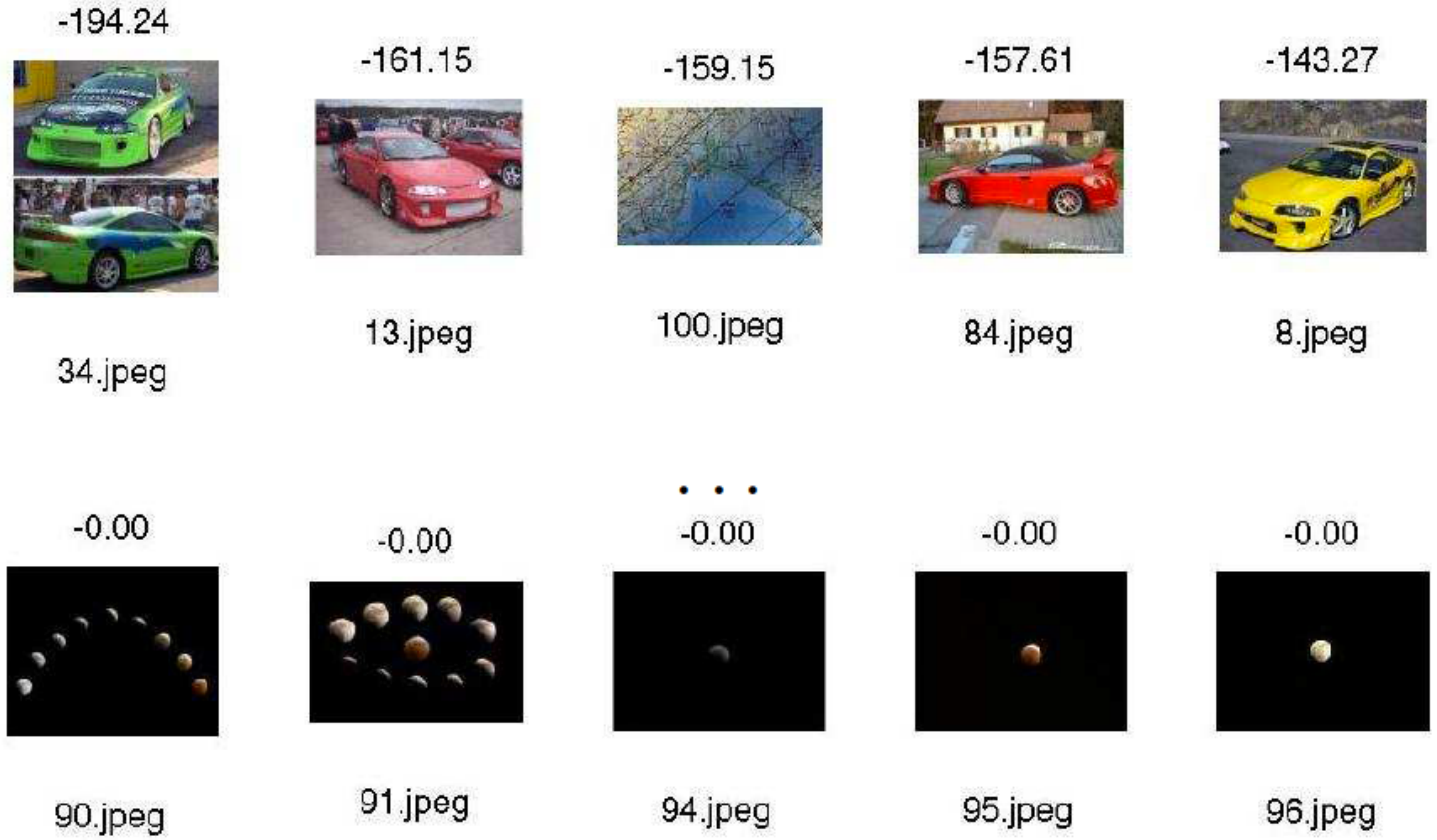
97.jpeg

Self-training example: image categorization

3. Add the most confident images and predicted labels to labeled data



4. Re-train the classifier and repeat



Comments on Self-training

Advantages of self-training

- The simplest semi-supervised learning method.
- A wrapper method, applies to existing (complex) classifiers.
- Often used in real tasks like natural language processing.

Disadvantages of self-training

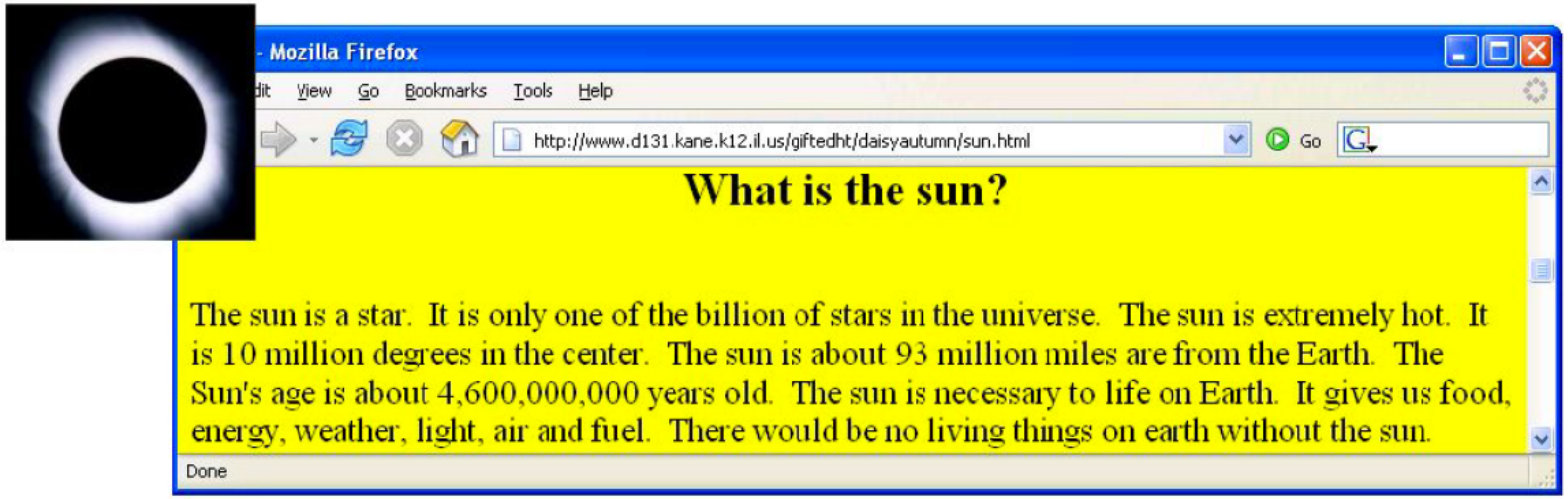
- Early mistakes could reinforce themselves.
  - Heuristic solutions, e.g. “un-label” an instance if its confidence falls below a threshold.
- Cannot say too much in terms of convergence.
  - But there are special cases when self-training is equivalent to the Expectation-Maximization (EM) algorithm.
  - There are also special cases (e.g., linear functions) when the closed-form solution is known.

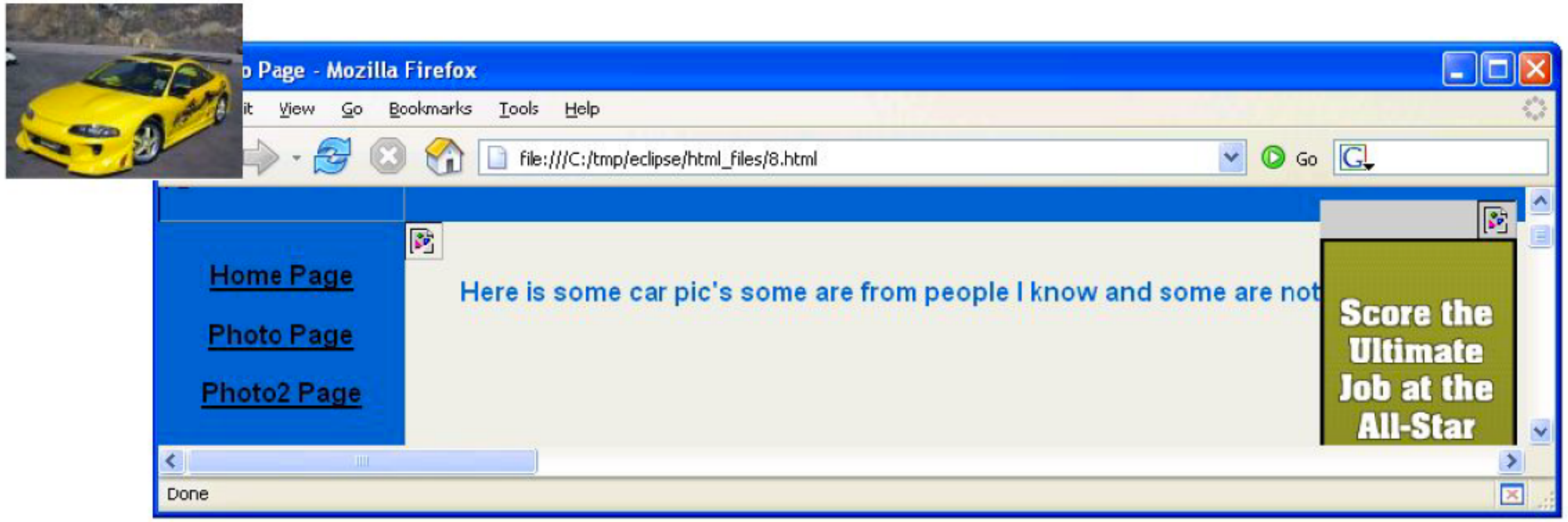
Co-training

one type of Multi-view algorithm

Co-training

Two views of an item: image and HTML text





Feature split

Each instance is represented by two sets of features $x = [x^{(1)}; x^{(2)}]$

- $x^{(1)}$ = image features
- $x^{(2)}$ = web page text
- This is a natural feature split (or multiple views)

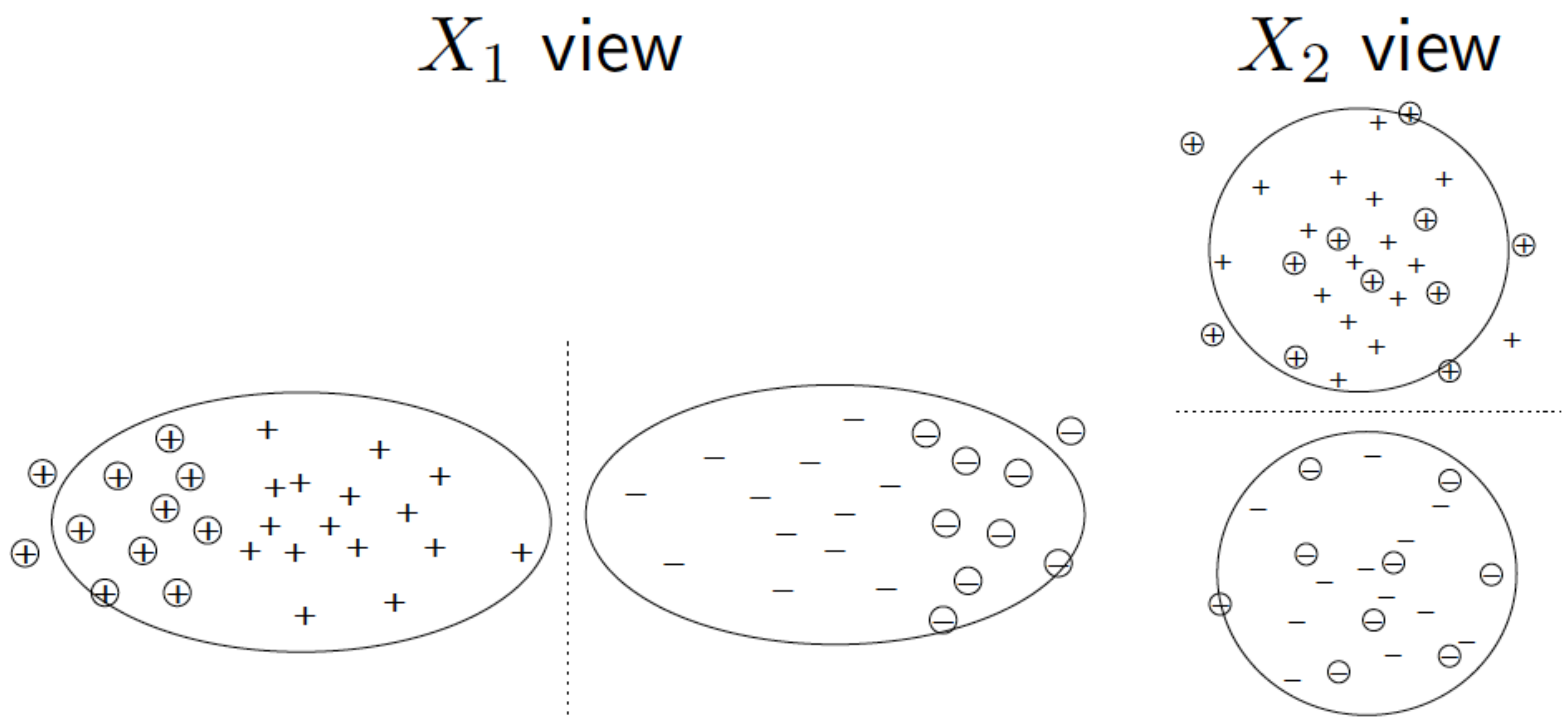
Co-training idea:

- Train an image classifier and a text classifier
- The two classifiers teach each other

Co-training assumptions

Assumptions

- feature split $x = [x^{(1)}; x^{(2)}]$ exists
- $x^{(1)}$ or $x^{(2)}$ alone is sufficient to train a good classifier
- $x^{(1)}$ and $x^{(2)}$ are conditionally independent given the class



Co-training algorithm

Co-training algorithm

1. Train two classifiers: $f^{(1)}$ from $(X_l^{(1)}, Y_l)$, $f^{(2)}$ from $(X_l^{(2)}, Y_l)$.
2. Classify X_u with $f^{(1)}$ and $f^{(2)}$ separately.
3. Add $f^{(1)}$'s k-most-confident $(x, f^{(1)}(x))$ to $f^{(2)}$'s labeled data.
4. Add $f^{(2)}$'s k-most-confident $(x, f^{(2)}(x))$ to $f^{(1)}$'s labeled data.
5. Repeat.

Pros and cons of co-training

Optional subtitle

Pros

- Simple wrapper method. Applies to almost all existing classifiers
- Less sensitive to mistakes than self-training

Cons

- Natural feature splits may not exist
- Models using BOTH features should do better

Variants of co-training

Co-EM: add all, not just top k

- Each classifier probabilistically label X_u
- Add (x, y) with weight $P(y|x)$

Fake feature split

- create random, artificial feature split
- apply co-training

Multiview: agreement among multiple classifiers

- no feature split
- train multiple classifiers of different types
- classify unlabeled data with all classifiers
- add majority vote label

Graph-based semi-supervised learning

Graph-based semi-supervised learning

Key idea

- Construct a graph with nodes being instances and edges being similarity measures among instances
- Look for some techniques to cut the graph
    - Labeled instances
    - Some heuristics, e.g., minimum cut

Graph-based semi-supervised learning

Key idea

- Construct a graph with nodes being instances and edges being similarity measures among instances
- Look for some techniques to cut the graph
  - Labeled instances
  - Some heuristics, e.g., minimum cut

Assumption

A graph is given on the labeled and unlabeled data. Instances connected by heavy edge tend to have the same label.

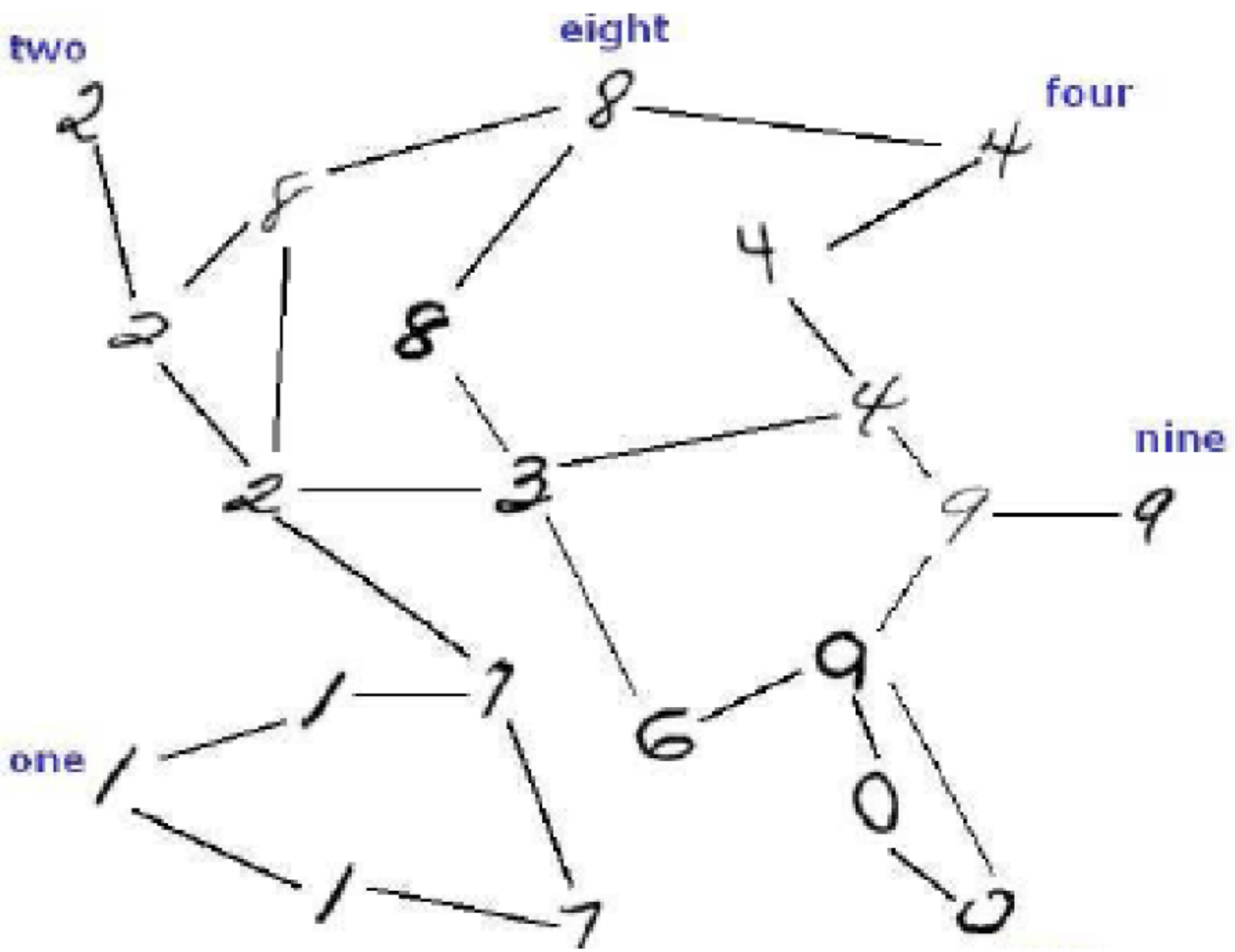
Graph construction

Graph construction

- $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V} = \{\mathbf{x}_i\}_{i=1}^n$
- Build adjacency graph using a heuristic
  - ϵ-NN. $\epsilon \in \mathbb{R}^+$. Nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected if $\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j) \le \epsilon$
  - k-NN. $k \in \mathbb{N}^+$. Nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected if $\mathbf{x}_i$ is among the k nearest neighbors of $\mathbf{x}_j$.
- Graph weighting
  - Heat kernel. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected, the weight $W_{ij} = \exp^{-\frac{dist(\mathbf{x}_i, \mathbf{x}_j)}{t}}$, where $t \in \mathbb{R}^+$.
  - Simple-minded. $W_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected.

Graph-based semi-supervised learning

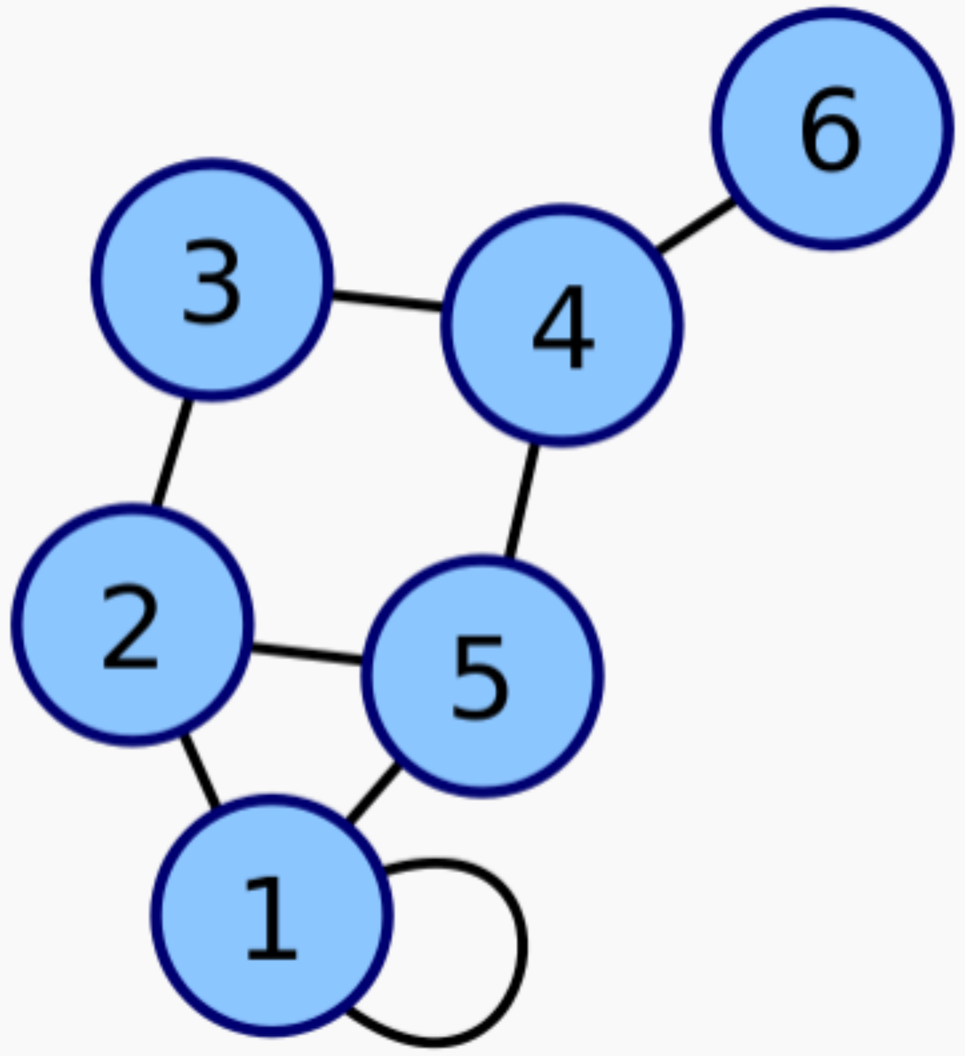
Optional subtitle



- $\mathcal{G} = <\mathcal{V}, \mathcal{E}>$
- W_{ij}: weights on edge $(\mathbf{x}_i, \mathbf{x}_j)$
- $D_{ii} = \sum_{j=1}^{n} W_{ij}$
- Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{W}$
- Weighted graph Laplacian: $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$

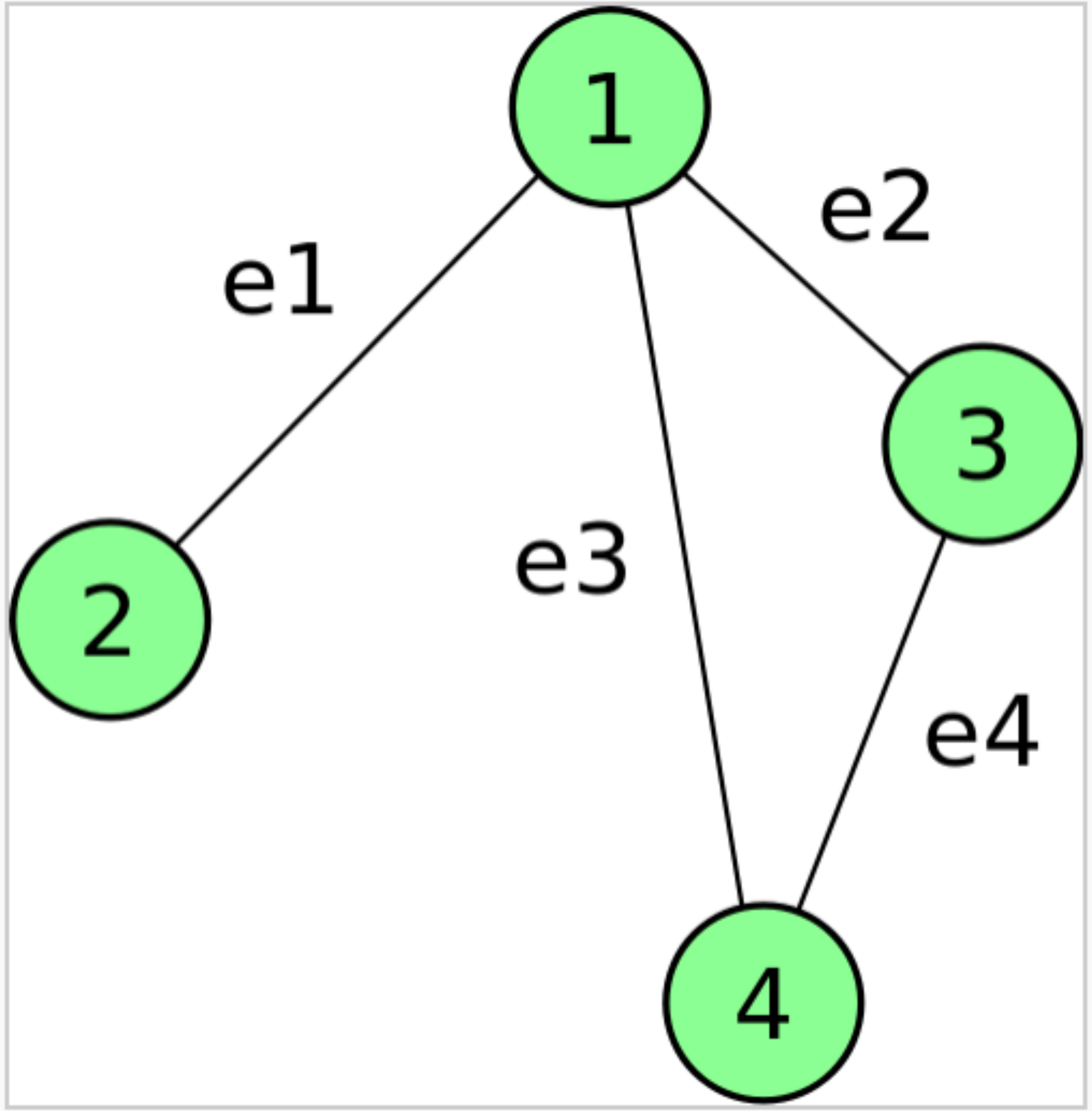
Adjacency matrix

Optional subtitle

Labeled graph	Adjacency matrix
	$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ Coordinates are 1–6.

Incidence matrix

The *unoriented incidence matrix* (or simply *incidence matrix*) of an undirected graph is a $n \times m$ matrix B, where n and m are the numbers of vertices and edges respectively, such that $B_{i,j} = 1$ if the vertex v_i and edge e_j are incident and 0 otherwise.



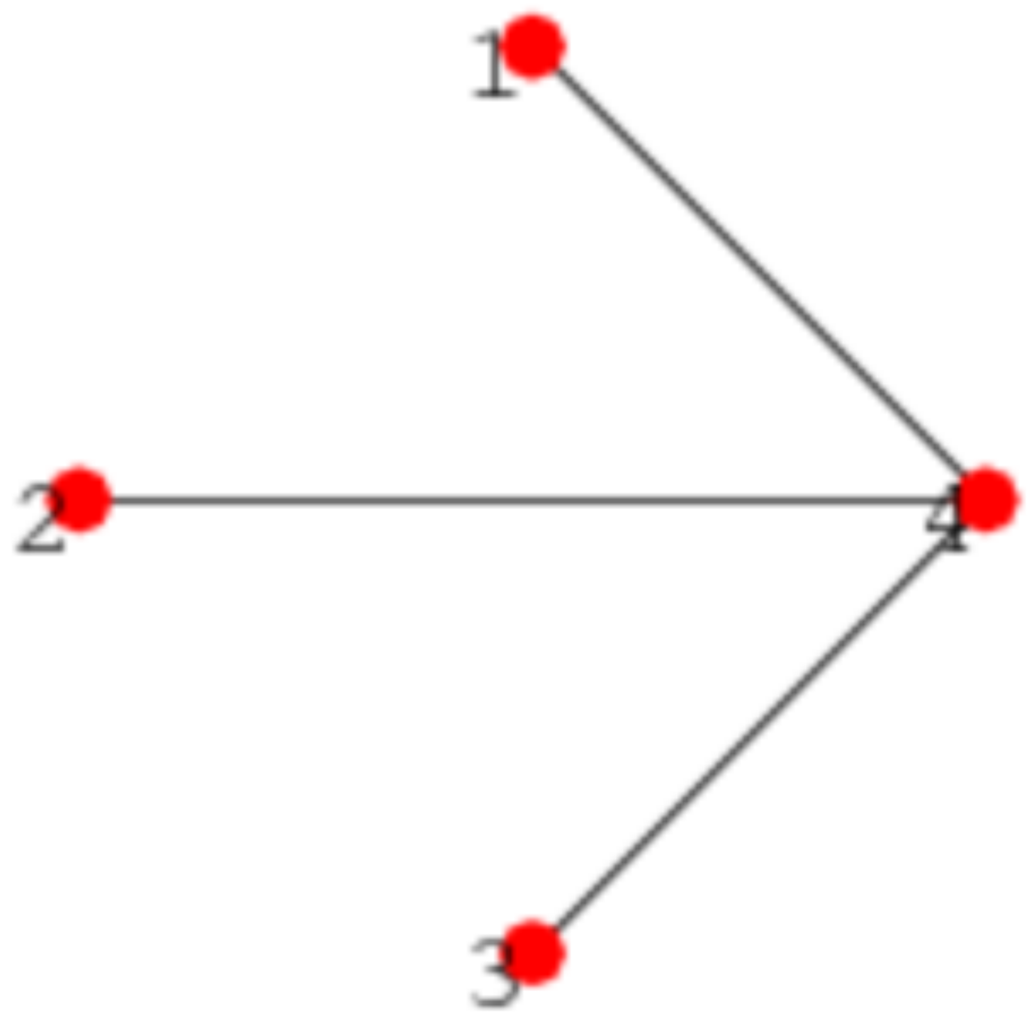
An undirected graph.

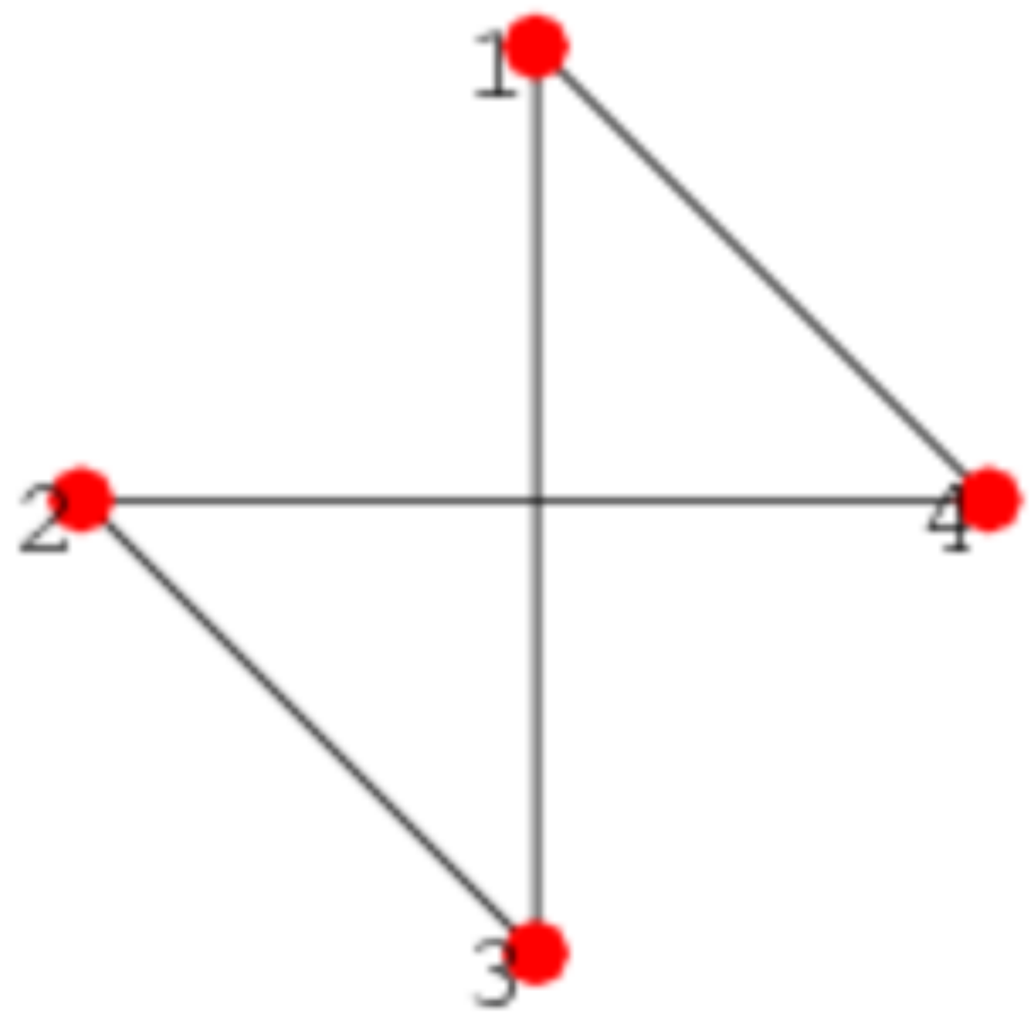
$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

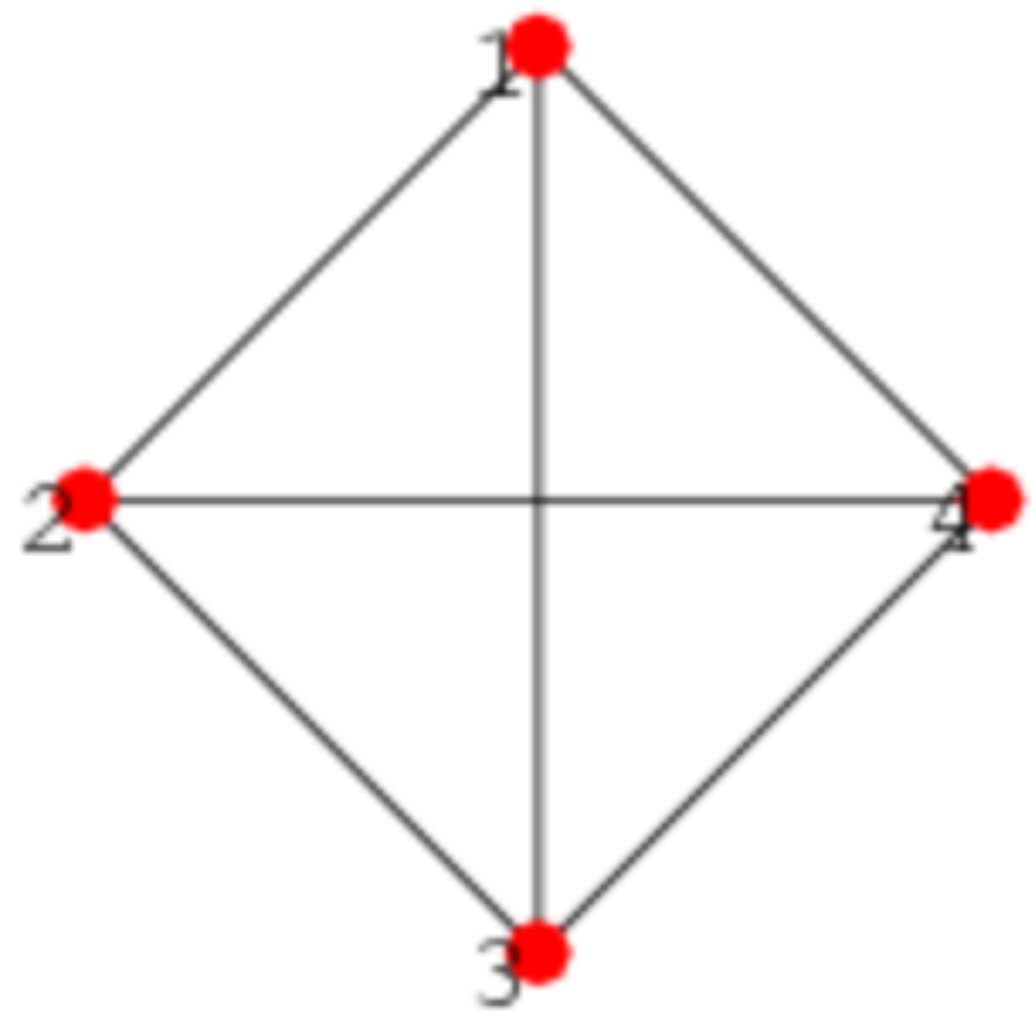
For example the incidence matrix of the undirected graph shown on the right is a matrix consisting of 4 rows (corresponding to the four vertices, 1-4) and 4 columns (corresponding to the four edges, e1-e4):

Questions: Incidence matrix & Adjacency matrix

Optional subtitle

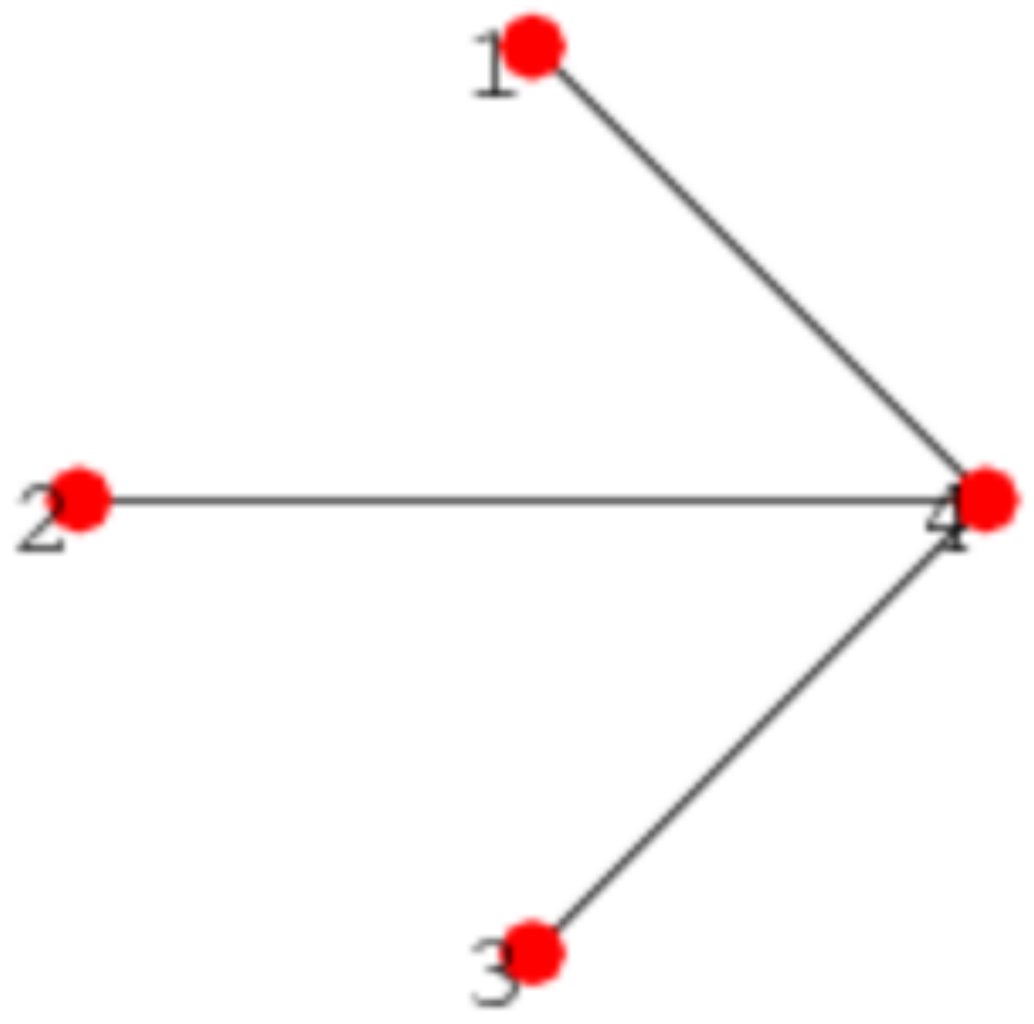


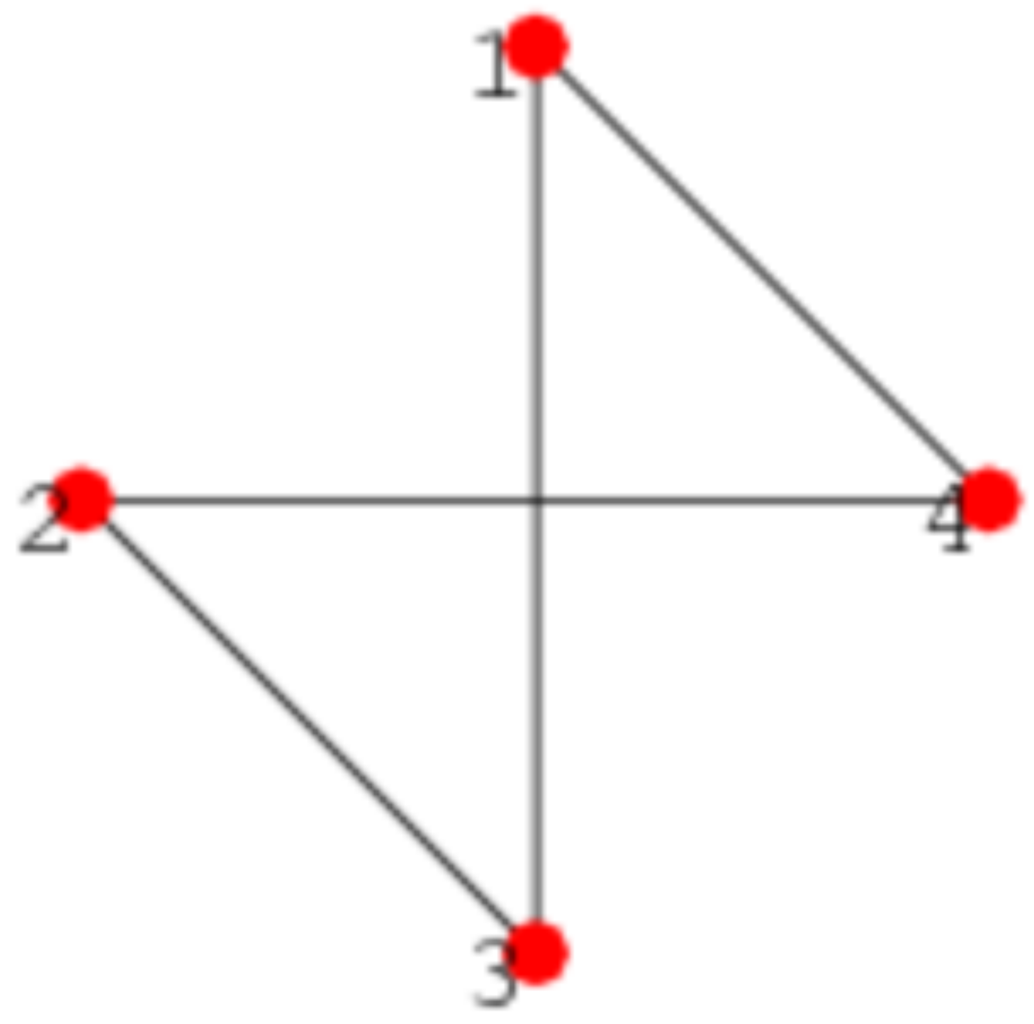


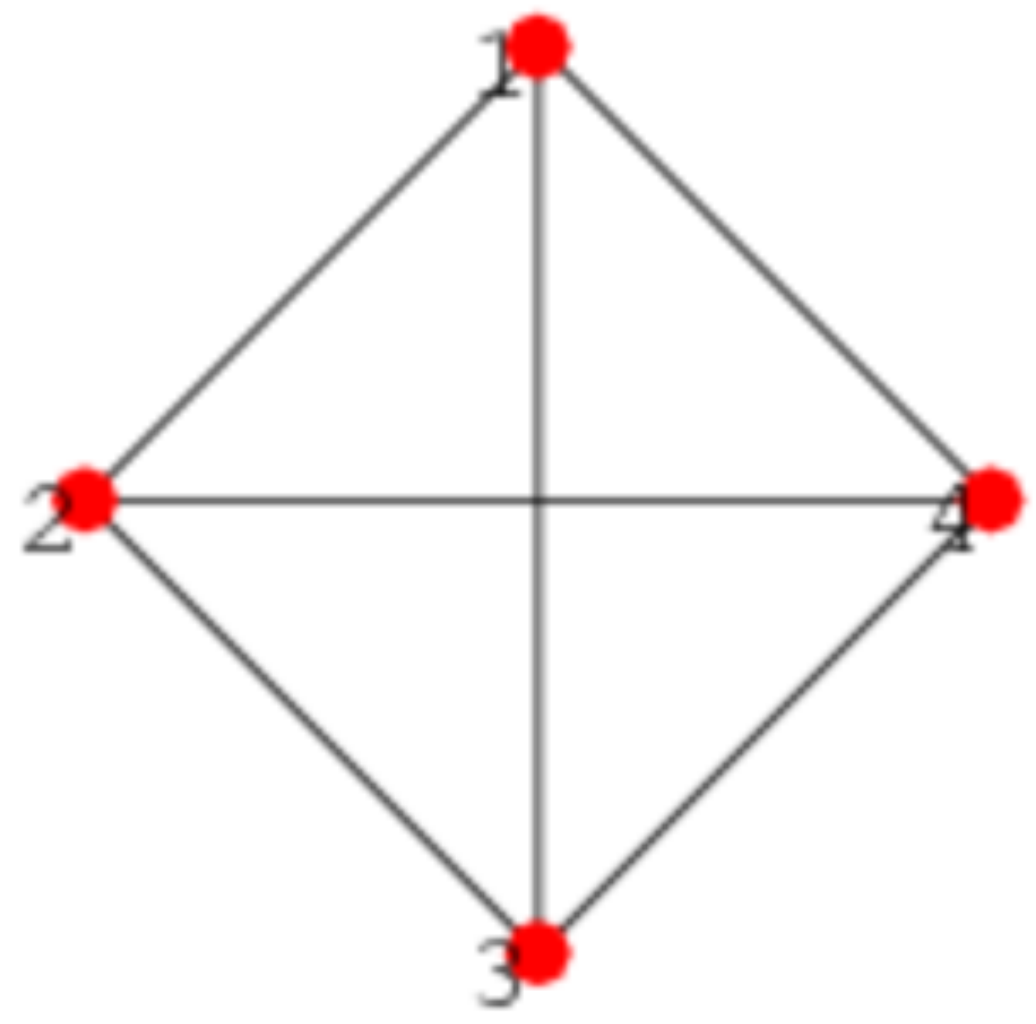


Questions: Incidence matrix & Adjacency matrix

Optional subtitle



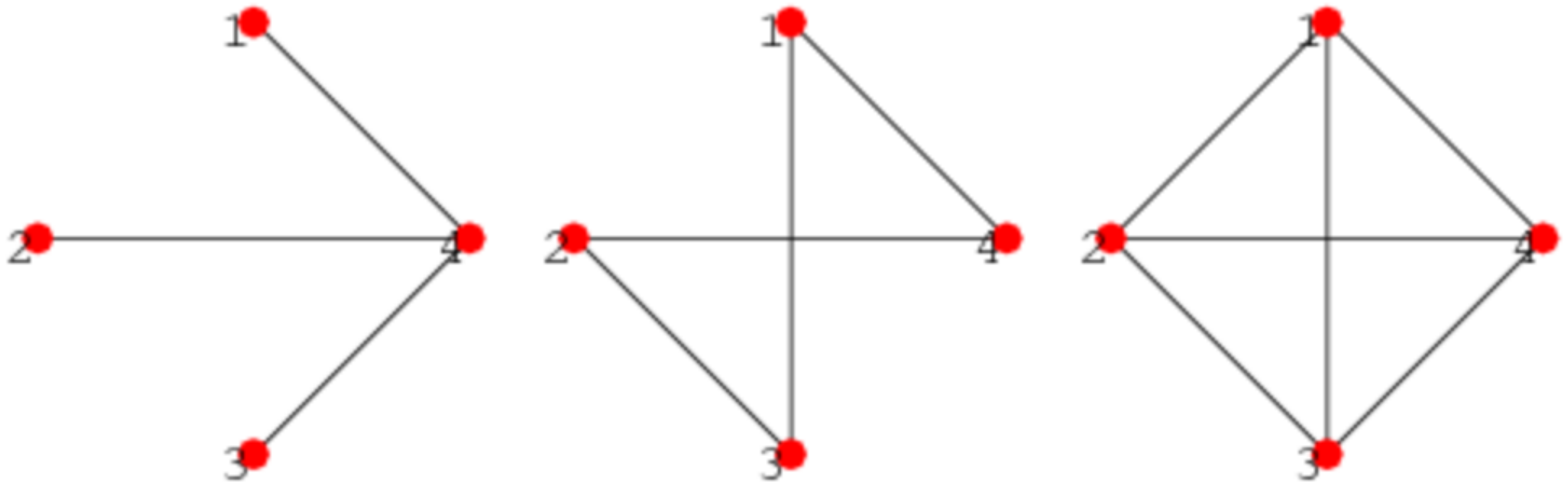




$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Questions: Incidence matrix & Adjacency matrix

Optional subtitle



$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

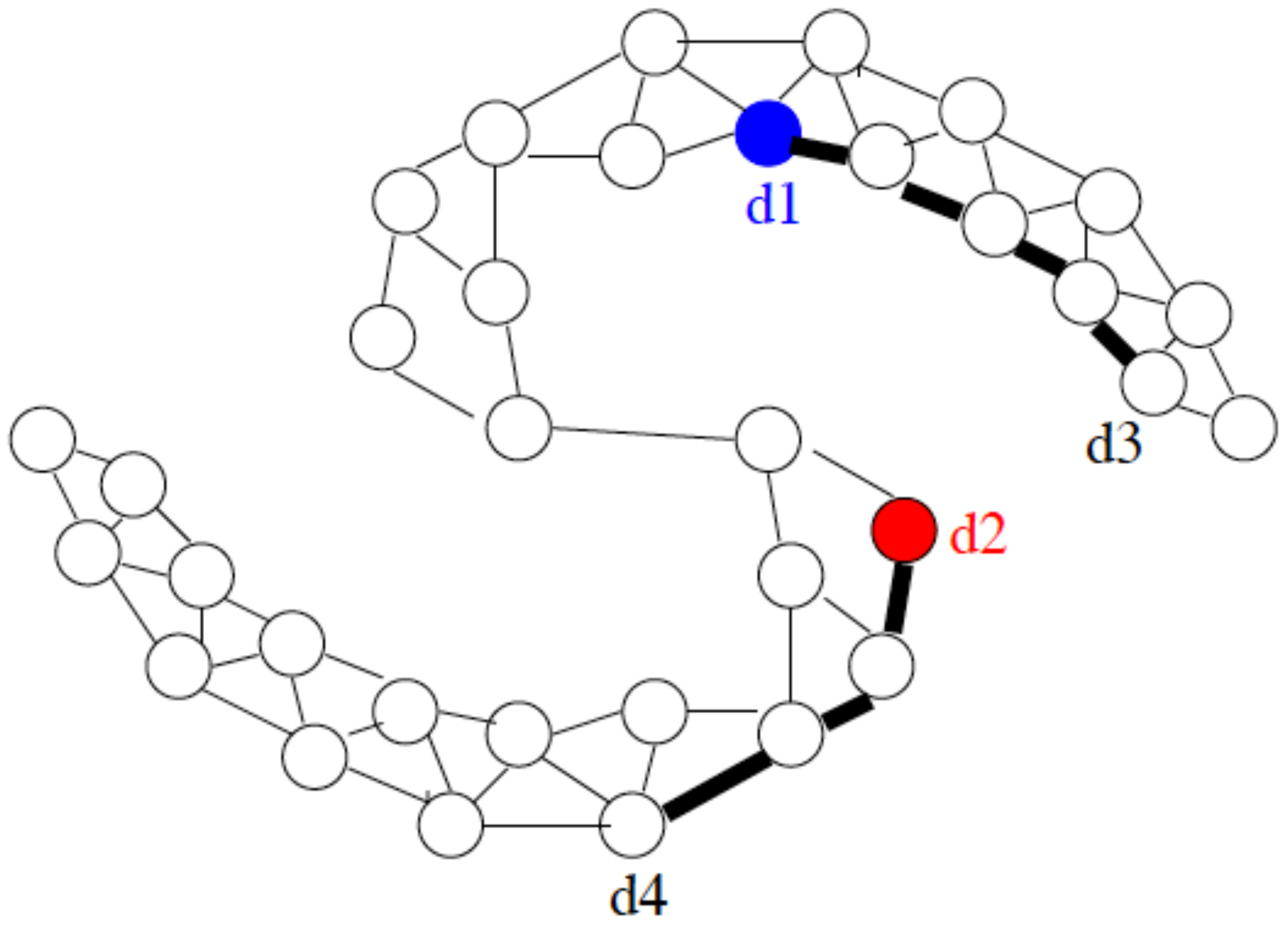
The incidence matrix C of a graph and adjacency matrix L of its line graph are related by

$$L = C^T C - 2 I,$$

where I is the identity matrix (Skiena 1990, p. 136).

The graph

- Nodes: $X_l \cup X_u$
- Edges: similarity weights computed from features, e.g.,
  - k-nearest-neighbor graph, unweighted (0, 1 weights)
  - fully connected graph, weight decays with distance $w = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$
- Want: implied similarity via all paths



Label Propagation on graphs

Optional subtitle

- Initial class assignments $\hat{\mathbf{y}} = \{-1, 0, +1\}^n$

$$\hat{y}_i = \begin{cases} \pm 1 & \forall \mathbf{x}_i \in \mathbf{X}_l \\ 0 & \forall \mathbf{x}_i \in \mathbf{X}_u \end{cases}$$

- Predicted class assignments
  1. Predict the confidence scores $\mathbf{f} = (f_1, \ldots, f_n)$
  2. Predict the class assignments $y_i = \mathrm{sign}\,(f_i)$

Harmonic function f

Optional subtitle

Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies

- $f(x_i) = y_i$ for $i = 1 \ldots l$
- f minimizes the energy

$$\sum_{i \sim j} w_{ij} (f(x_i) - f(x_j))^2$$

- the mean of a Gaussian random field
- average of neighbors $f(x_i) = \frac{\sum_{j \sim i} w_{ij} f(x_j)}{\sum_{j \sim i} w_{ij}}, \forall x_i \in X_u$

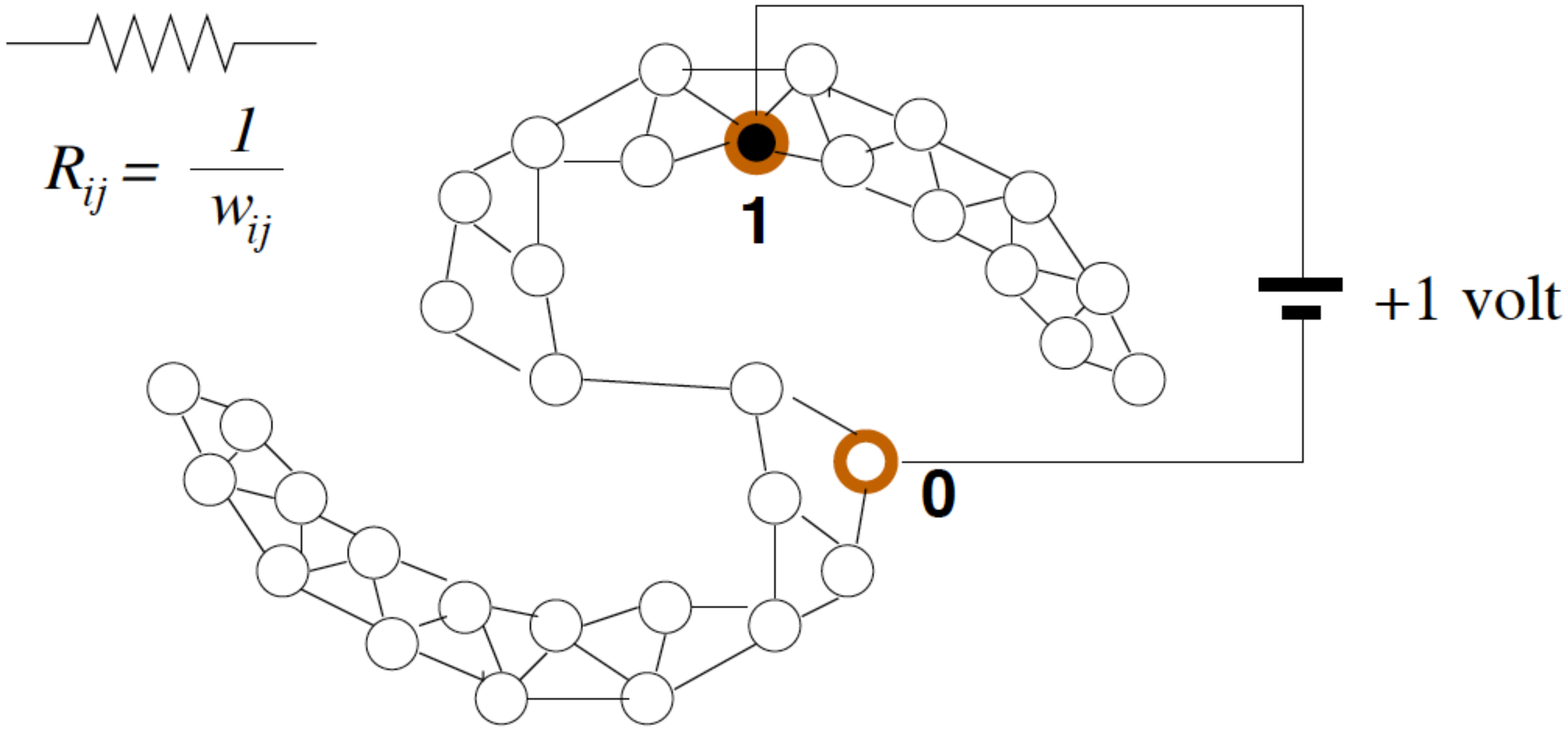
Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. Xiaojin Zhu, Zoubin Ghahramani, John Lafferty. ICML-2003. Awarded the classic paper prize in ICML 2013

An electric network interpretation

Optional subtitle

- Edges are resistors with conductance w_{ij}
- 1 volt battery connects to labeled points $y = 0, 1$
- The voltage at the nodes is the harmonic function f

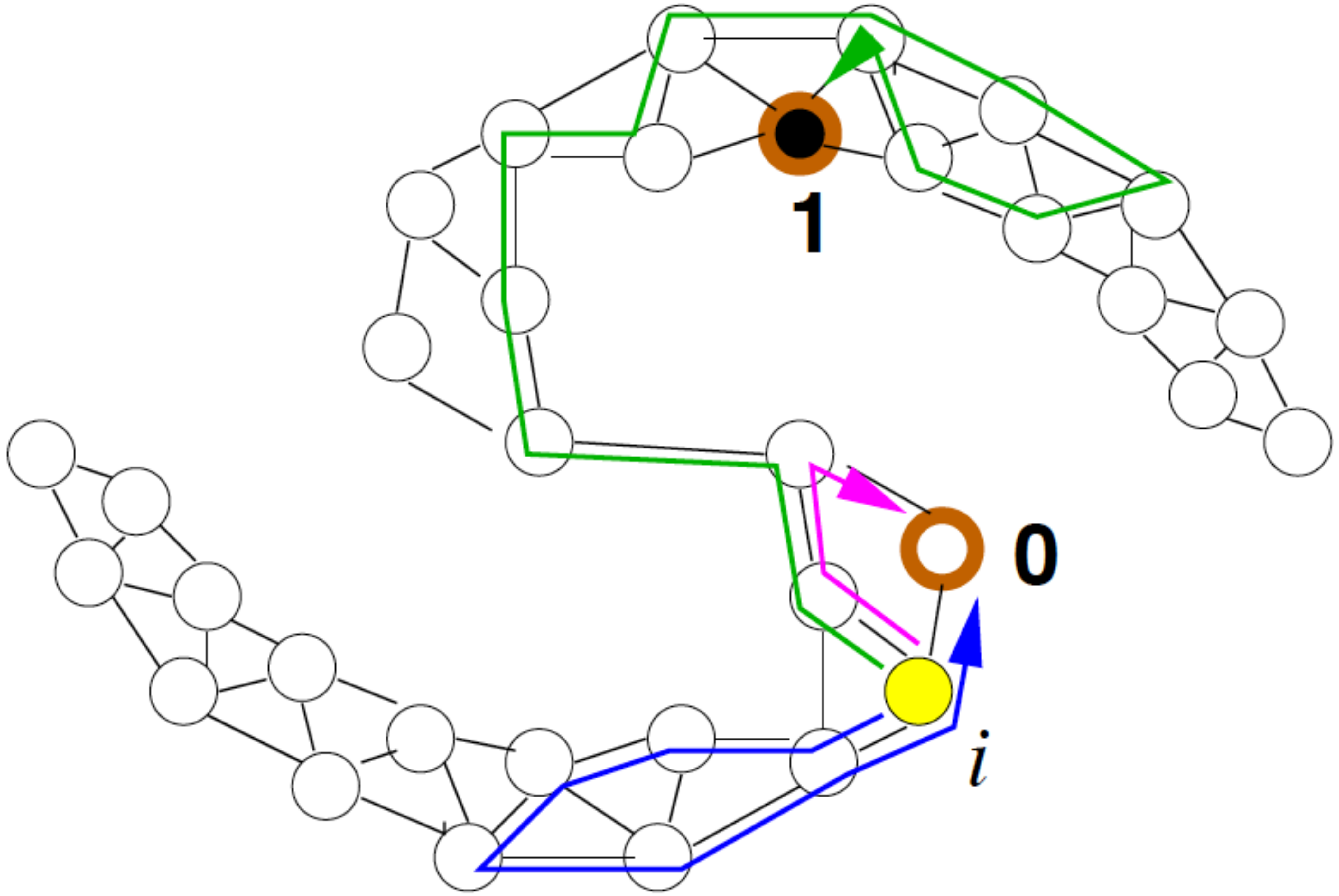
Implied similarity: similar voltage if many paths exist



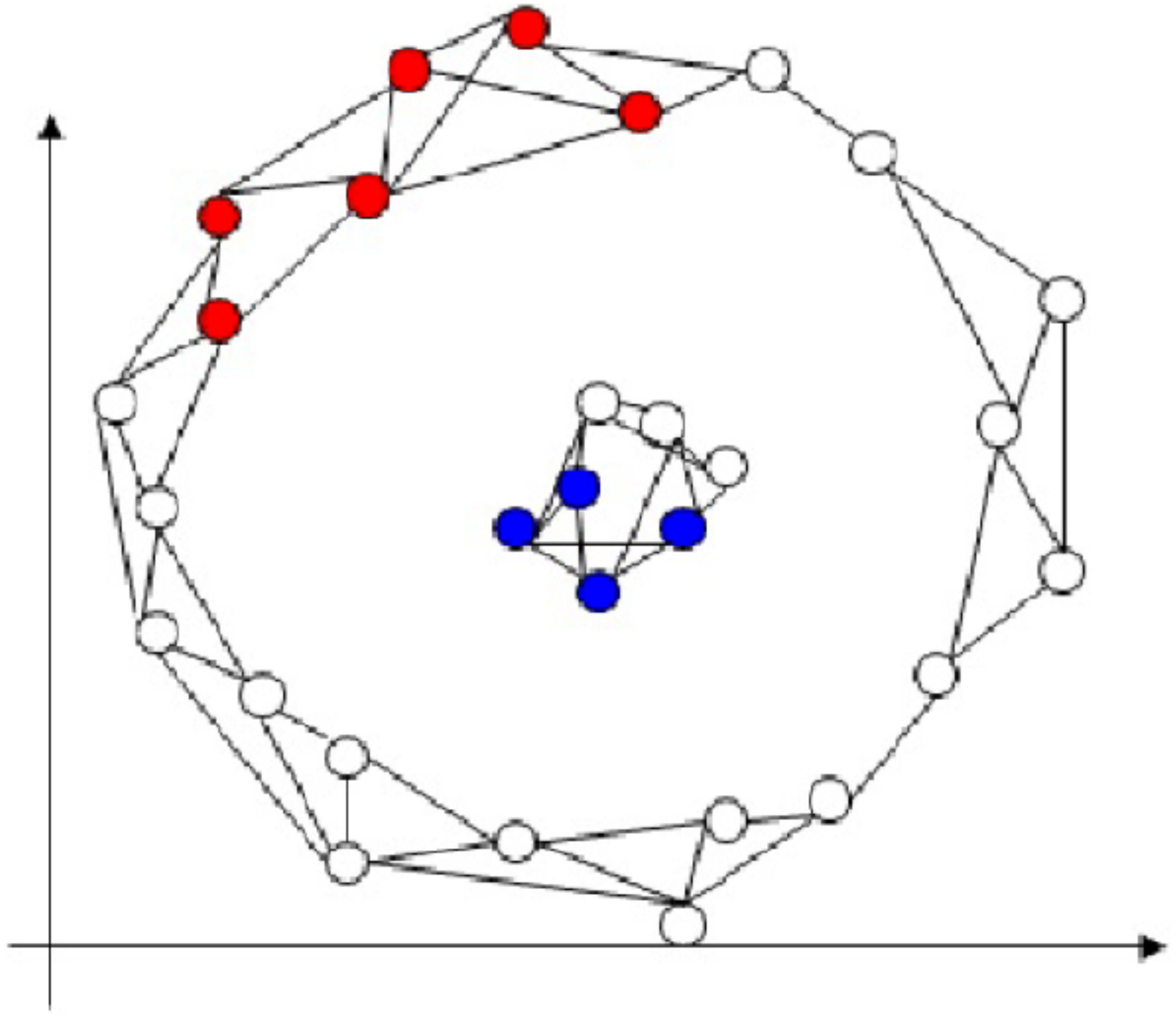
A random walk interpretation

Optional subtitle

- Randomly walk from node i to j with probability $\frac{w_{ij}}{\sum_k w_{ik}}$
- Stop if we hit a labeled node
- The harmonic function $f = Pr(\text{hit label 1}|\text{start from } i)$



Label Propagation



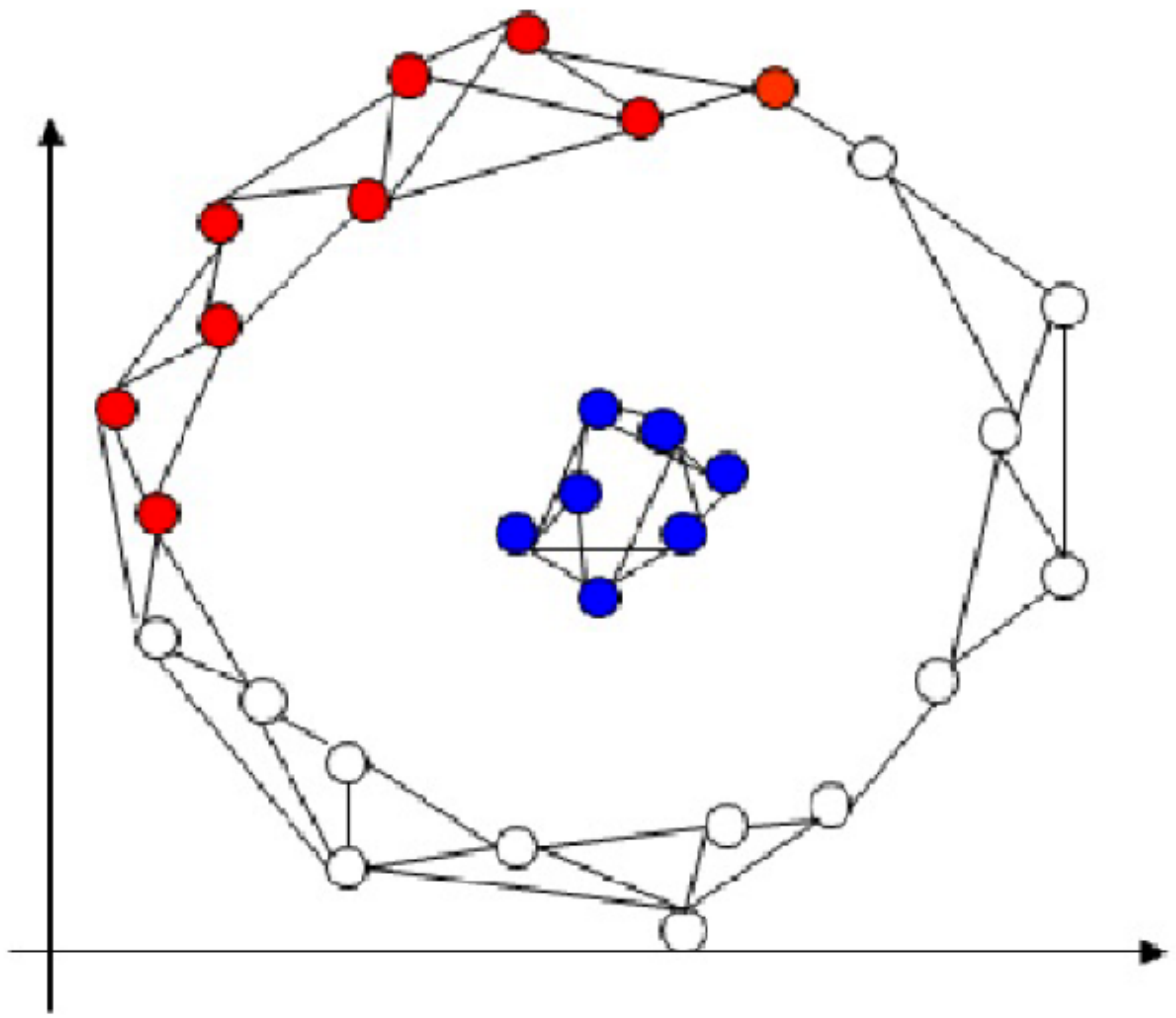
One round of propagation

- $$f_i = \begin{cases} \hat{y}_i & \forall \mathbf{x}_i \in \mathbf{X}_l \\ \alpha \sum_{j=1}^{n} W_{ij}\hat{y}_i & \forall \mathbf{x}_i \in \mathbf{X}_u \end{cases}$$
- $\mathbf{f}^{(1)} = \hat{\mathbf{y}} + \alpha W \hat{\mathbf{y}}$

Zhou D.-Y. Learning with Local and Global Consistency, NIPS 2004.

Label Propagation

Optional subtitle

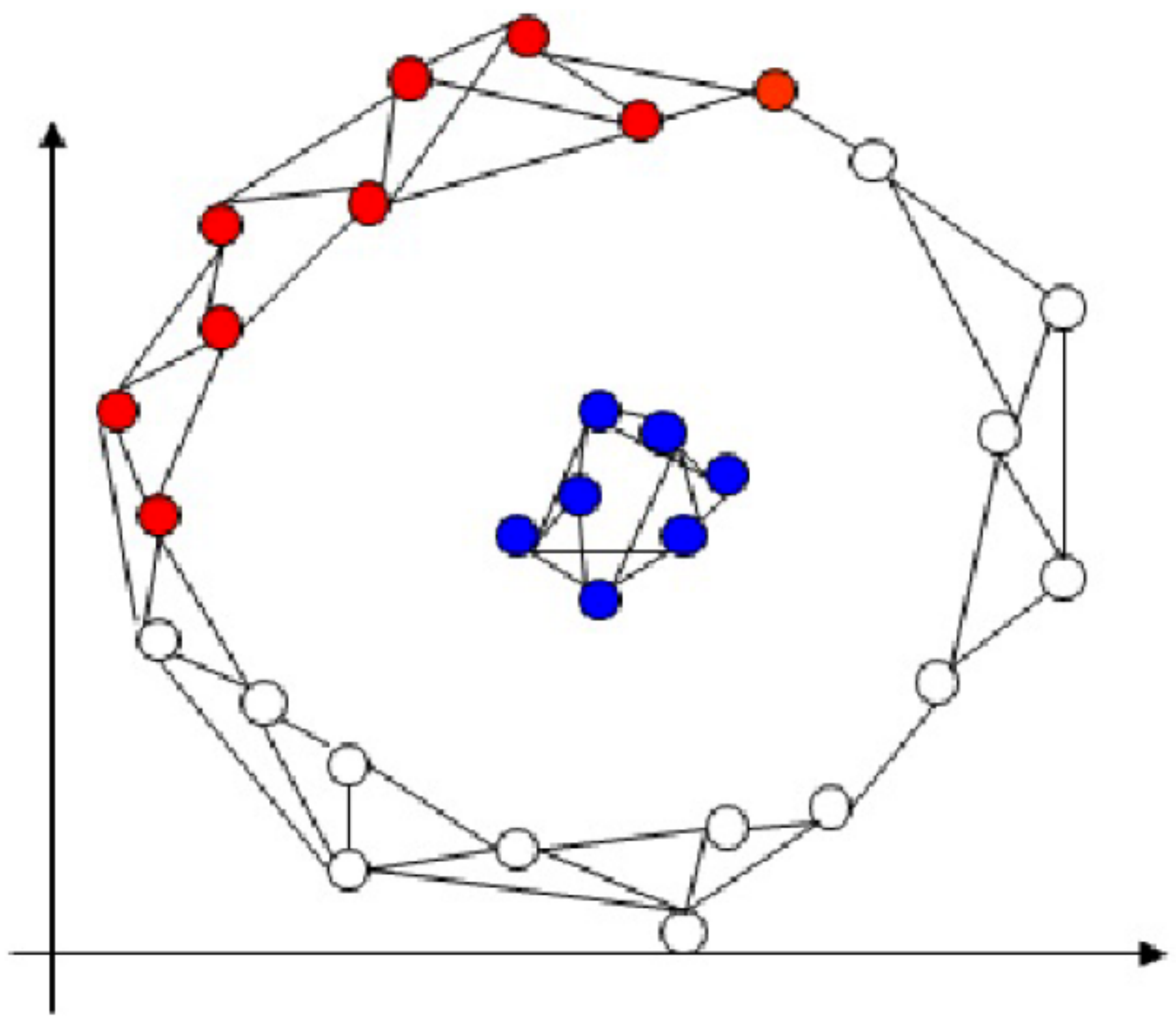


Two rounds of propagation

$$\begin{aligned}\mathbf{f}^{(2)} &= \mathbf{f}^{(1)} + \alpha W \mathbf{f}^{(1)} \\ &= \hat{\mathbf{y}} + \alpha W \hat{\mathbf{y}} + \alpha^2 W^2 \hat{\mathbf{y}}\end{aligned}$$

Label Propagation

Optional subtitle

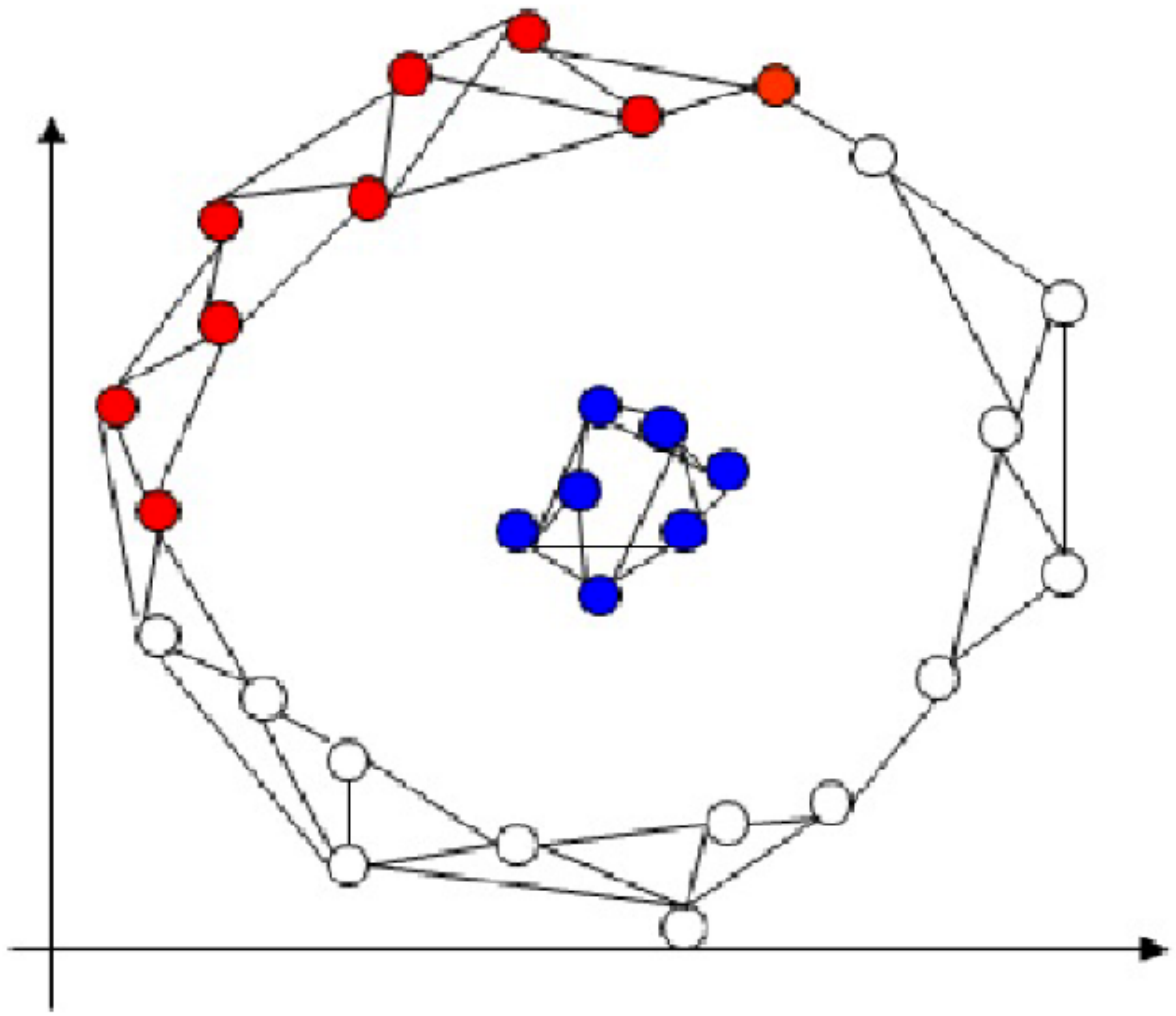


Any rounds of propagation

$$\mathbf{f}^{(t)} = \hat{\mathbf{y}} + \sum_{k=1}^{t} \alpha^k W^k \hat{\mathbf{y}}$$

Label Propagation

Optional subtitle



Infinite rounds of propagation

$$\mathbf{f}^{(\infty)} = \hat{\mathbf{y}} + \sum_{k=1}^{\infty} \alpha^k W^k \hat{\mathbf{y}}$$

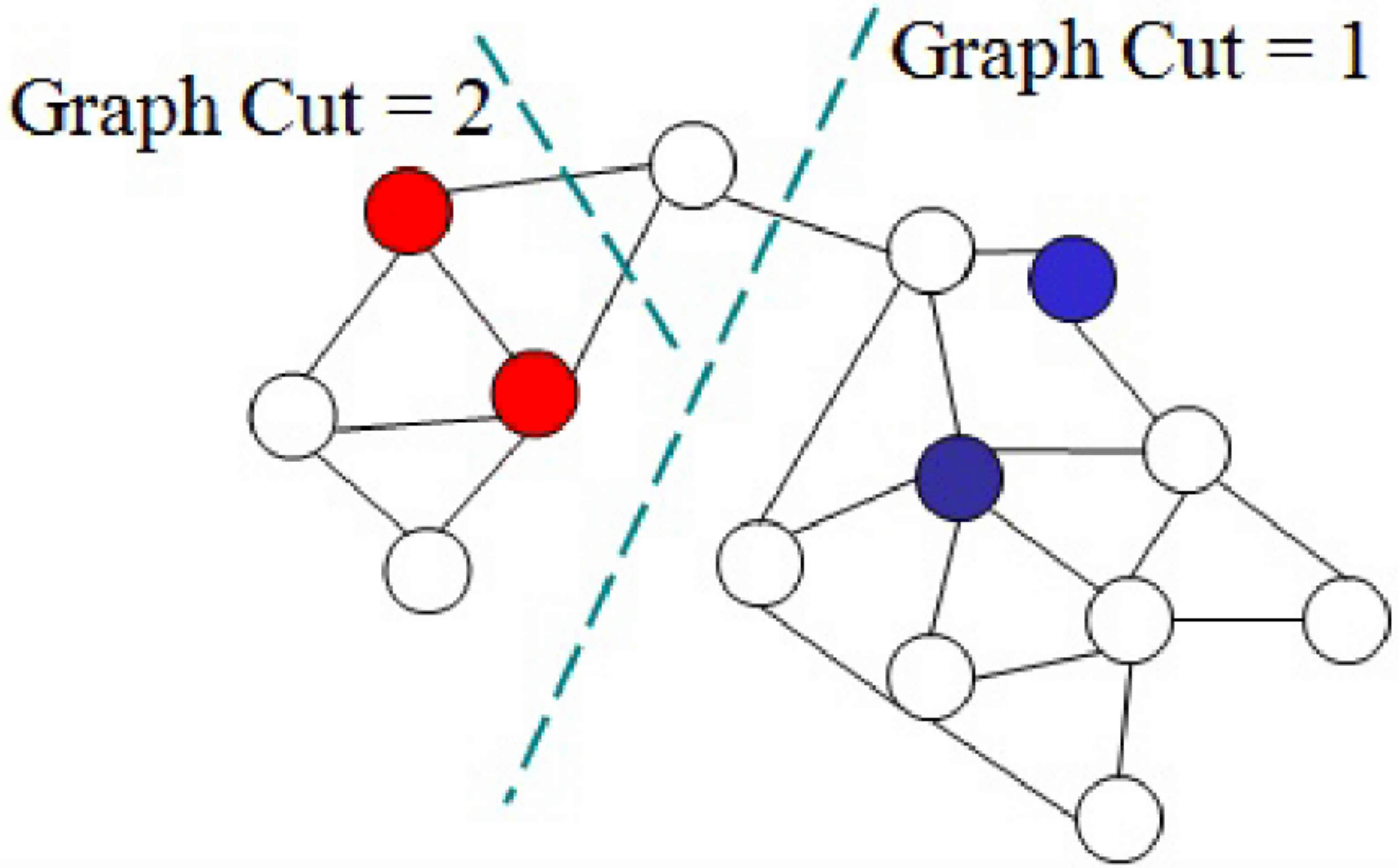
Or equivalently

$$\mathbf{f}^{(\infty)} = (\mathbf{I} - \alpha W)^{-1} \hat{\mathbf{y}}$$

Graph partition

Optional subtitle

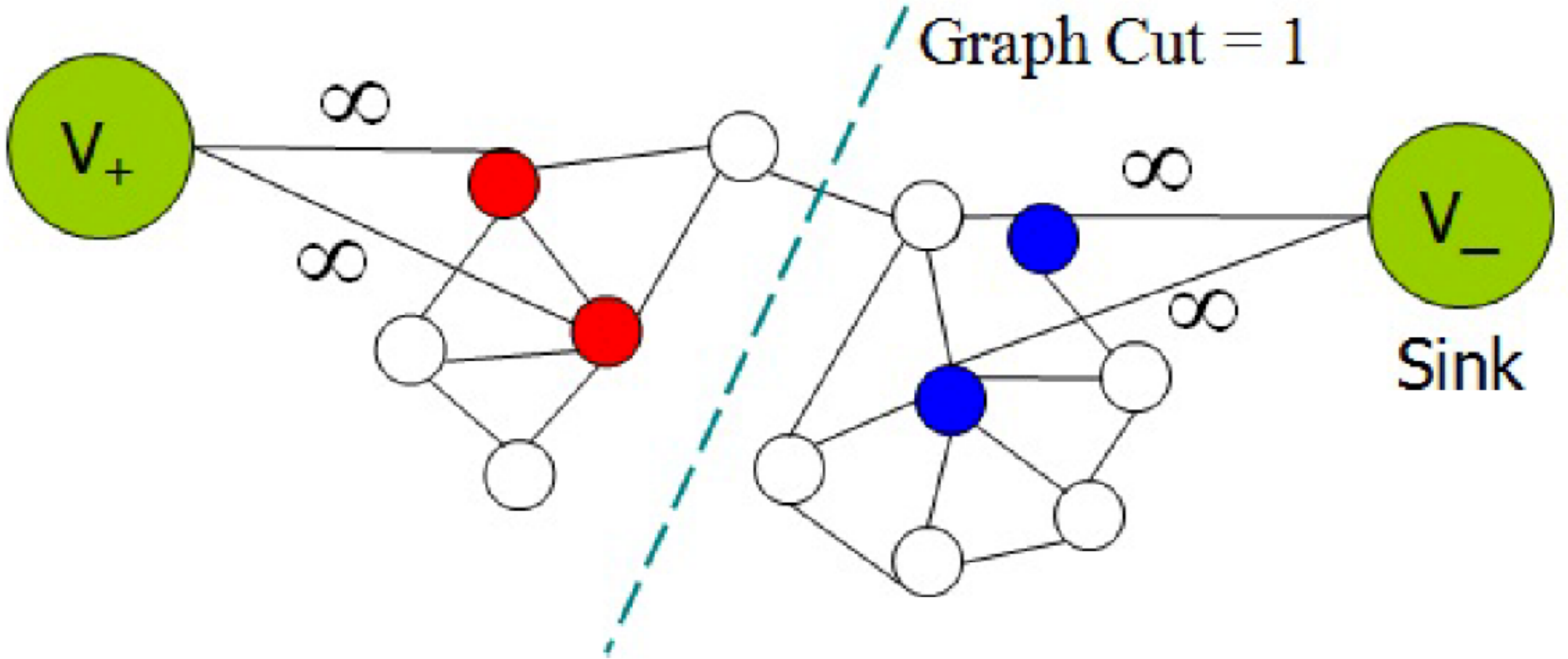
- Key idea
  - Classification as graph partitioning
- Search for a classification boundary
  - Consistent with labeled examples
  - Partition with small graph cut



Min-cuts

- V_+ : source, V_-: sink
- Infinite weights connecting sinks and sources

Karger's Min Cut Algorithm



Max-flow min-cut theorem:

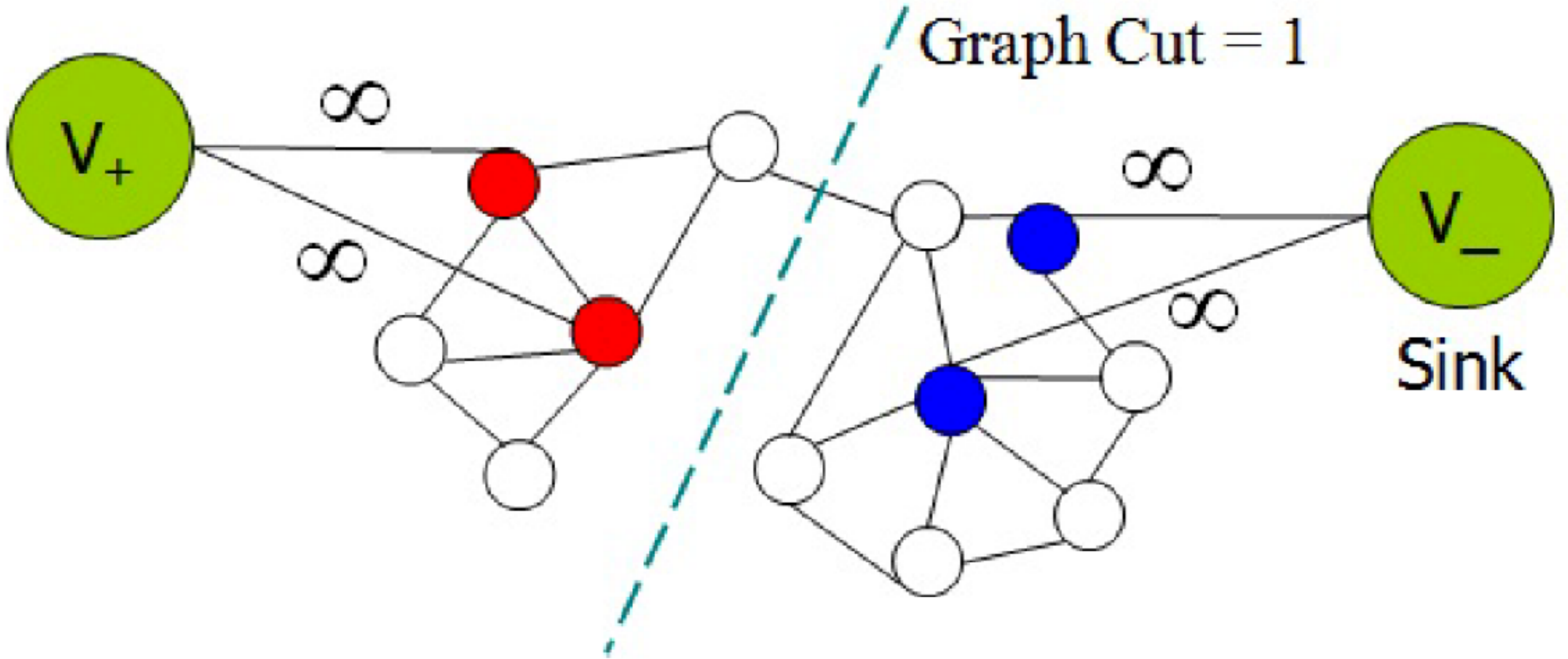
The cut in a flow network that separates the source and sink vertices and minimizes the total weight on the edges that are directed from the source side of the cut to the sink side of the cut. As shown in the max-flow min-cut theorem, the weight of this cut equals the maximum amount of flow that can be sent from the source to the sink in the given network.

In graph theory, a **minimum cut** of a graph is a cut (a partition of the vertices of a graph into two disjoint subsets that are joined by at least one edge) that is minimal in some sense.

Min-cuts

- V_+ : source, V_-: sink
- Infinite weights connecting sinks and sources

Graph-cut
Normalised Cut
Karger's Min Cut Algorithm



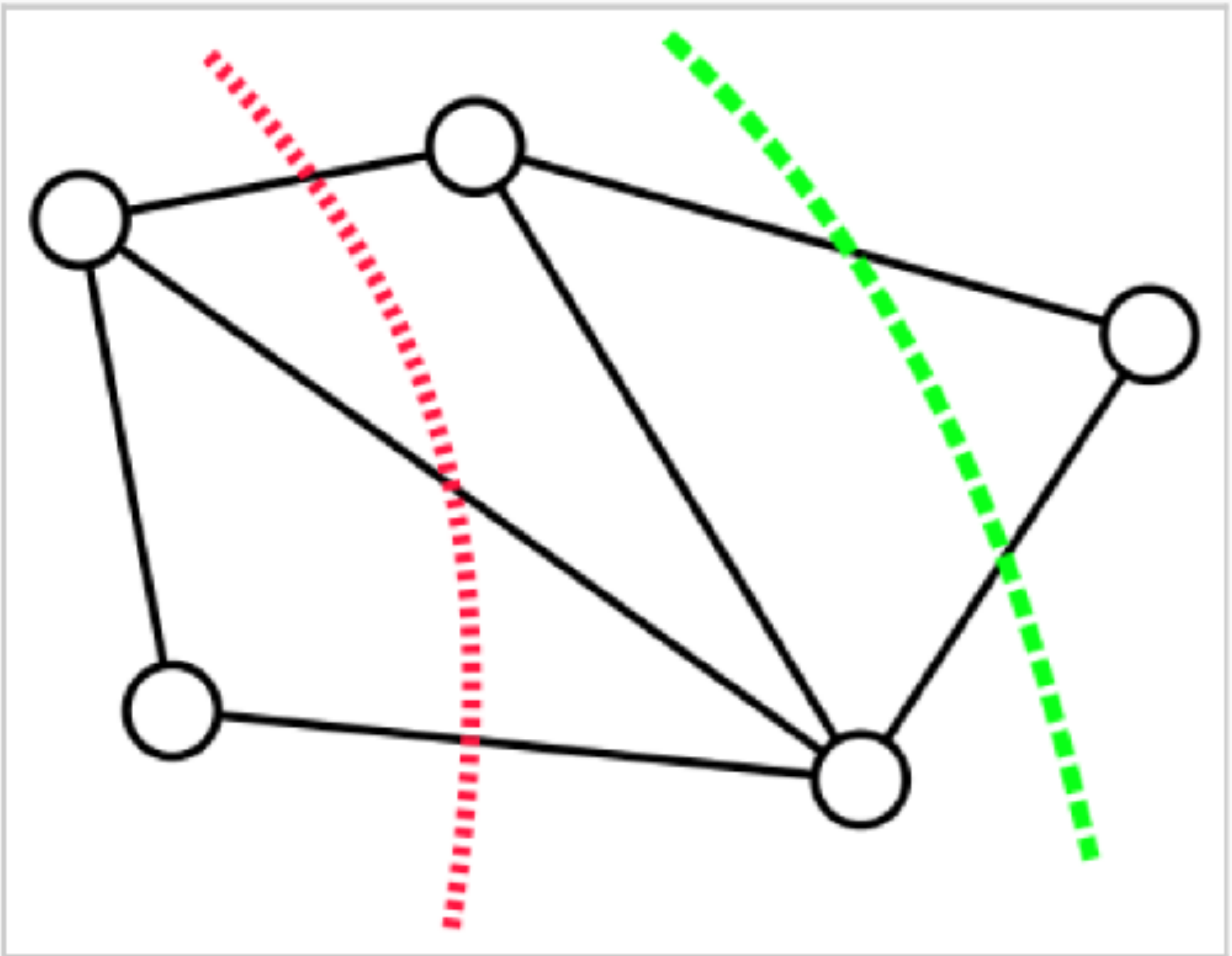
Max-flow min-cut theorem:

The cut in a flow network that separates the source and sink vertices and minimizes the total weight on the edges that are directed from the source side of the cut to the sink side of the cut. As shown in the max-flow min-cut theorem, the weight of this cut equals the maximum amount of flow that can be sent from the source to the sink in the given network.

In graph theory, a **minimum cut** of a graph is a cut (a partition of the vertices of a graph into two disjoint subsets that are joined by at least one edge) that is minimal in some sense.

Karger's **minimum cut** algorithm

The idea of the algorithm is based on the concept of contraction of an edge (u, v) in an undirected graph $G = (V, E)$. Informally speaking, the contraction of an edge merges the nodes u and v into one, reducing the total number of nodes of the graph by one. All other edges connecting either u or v are "reattached" to the merged node, effectively producing a multigraph. Karger's basic algorithm iteratively contracts randomly chosen edges until only two nodes remain; those nodes represent a cut in the original graph. By iterating this basic algorithm a sufficient number of times, a minimum cut can be found with high probability.



A graph and two of its cuts. The dotted line in red is a cut with three crossing edges. The dashed line in green is a min-cut of this graph, crossing only two edges.

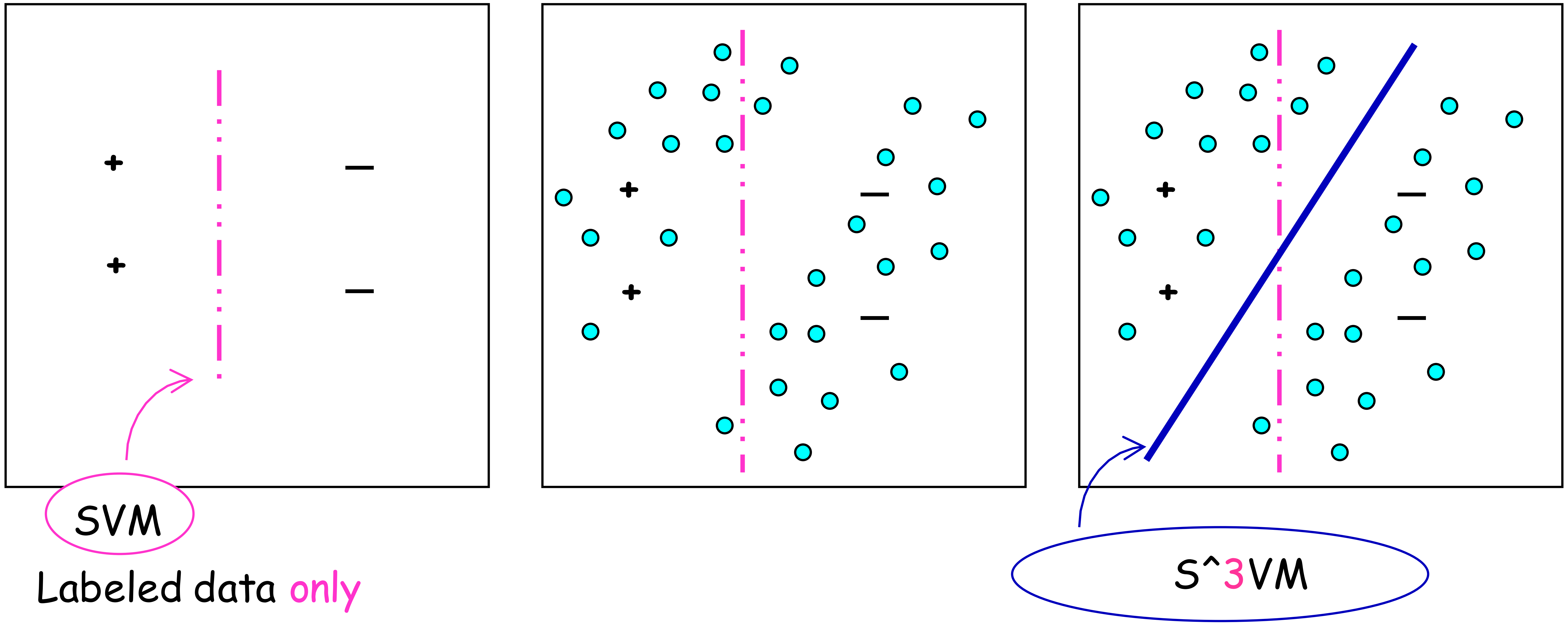
David Karger
ACM Doctoral Dissertation Award
United States – 1994

For his dissertation "Random Sampling in Graph Optimization Problems."

Semi-supervised Learning (Transductive) SVM

S3VM [Joachims98]

- Suppose we believe target separator goes through low density regions of the space/large margin.
- Aim for separator with large margin w.r.t labeled and unlabeled data. (L+U)



Low Density Separation Assumption

The decision boundary should lie in a low-density region, that is the decision boundary does not cut through dense unlabeled data.

Also known as cluster assumption

Semi-supervised SVM

Optional subtitle

S3VM: $\mathbf{y}_u$ for unlabeled data as a free variable

S3VM

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi} \min_{\mathbf{y}_u \in \{-1,+1\}^n} \quad & \|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i \\
\text{s. t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \ i = 1, \ldots, l \\
& y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \ i = l+1, \ldots, n \\
& \xi_i \geq 0, \ i = 1, \ldots, n
\end{aligned}
$$

- No longer convex optimization problem
- Alternating optimization

Semi-supervised SVM

Optional subtitle

Equivalently, unconstrained form:

S3VM

$$\min_{f} \min_{\mathbf{y}_u} \quad \|\mathbf{w}\|_2^2 + C_l \sum_{i=1}^{l} (1 - y_i f(\mathbf{x}_i))_+ + C_u \sum_{i=l+1}^{l+u} (1 - y_i f(\mathbf{x}_i))_+$$

where $(1 - y_i f(\mathbf{x}_i))_+ = \max(0, 1 - y_i f(\mathbf{x}_i))$

Optimize over $\mathbf{y}^u = (y_{l+1}^u, \ldots, y_n^u)$, we have

$$\min_{y_i^u} (1 - y_i f(\mathbf{x}_i))_+ = (1 - \mathrm{sign}\ (f(\mathbf{x}_i)) f(\mathbf{x}_i))_+ = (1 - |f(\mathbf{x}_i)|)_+$$

Semi-supervised SVM

Optional subtitle

S3VM objective

$$\min_{f} \quad \|\mathbf{w}\|_2^2 + C_l \sum_{i=1}^{l} (1 - y_i f(\mathbf{x}_i))_+ + C_u \sum_{i=l+1}^{l+u} (1 - |f(\mathbf{x}_i)|)_+$$

- Non-convex problem
- Optimization methods?

Appendix