Semi-supervised Learning

Learning from a small labeled set

and a large unlabeled set

Wang Houfeng

Institute of Computational Linguistics

Peking University

Outline

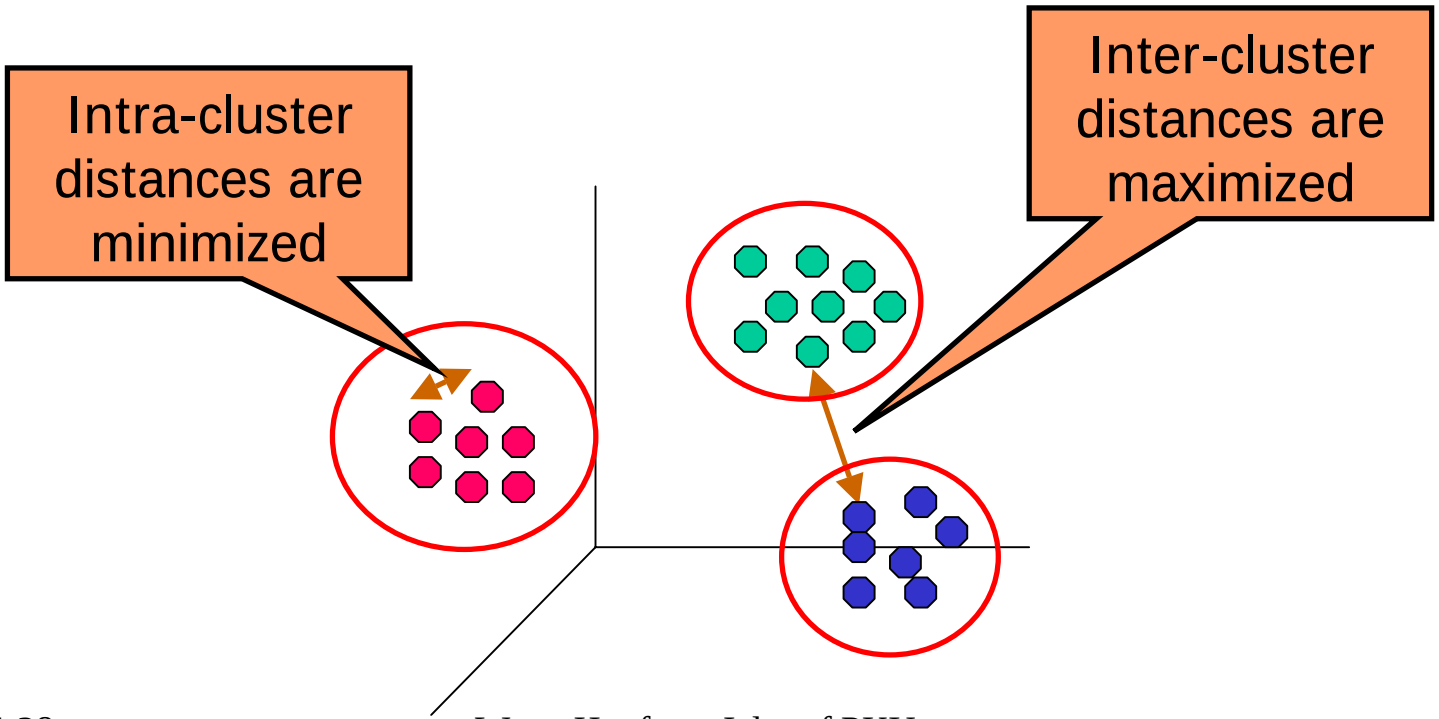
- **Overview**
- Semi-Clustering
- Semi-Classification

Supervised classification vs unsupervised clustering

- Unsupervised clustering
  - Group similar objects together to find clusters
    - Minimize intra-class distance
    - Maximize inter-class distance
- Supervised classification
  - Given: training samples with Class label;
  - Training: a model or classifier
  - Predicting: classifying new instances with the classifier

Clustering

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



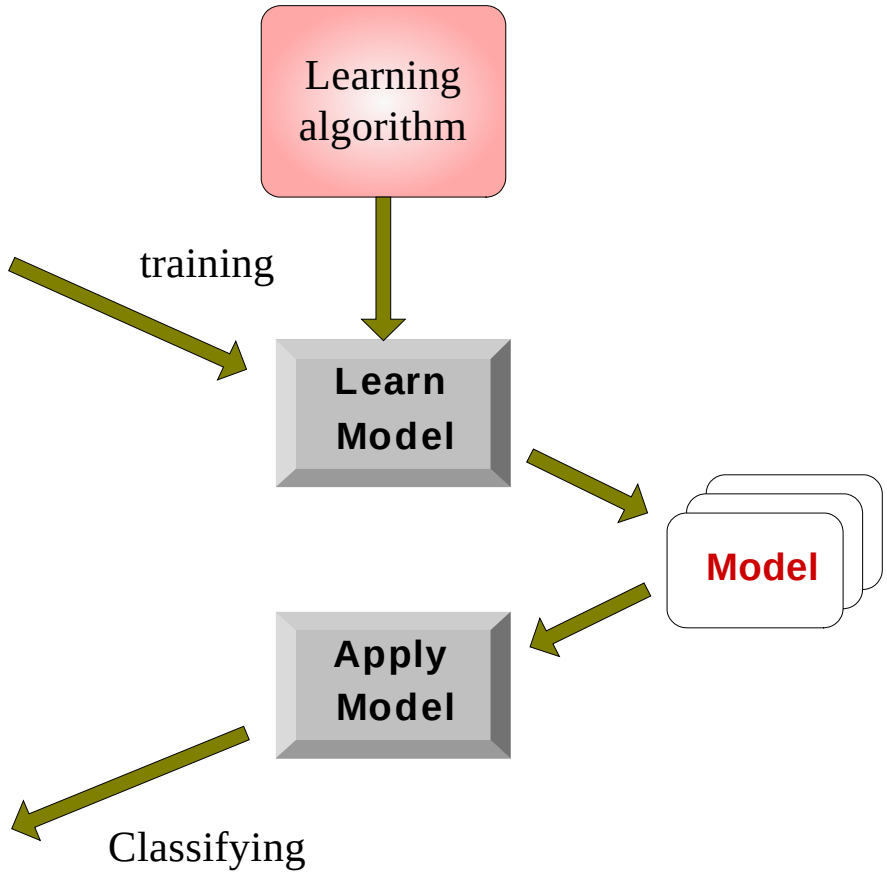
Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Semi-Supervised Learning

- Combines labeled and unlabeled data during training to improve performance:
  - Semi-supervised classification: Training on labeled data exploits additional unlabeled data, frequently resulting in a more accurate classifier.
  - Semi-supervised clustering: Uses small amount of labeled data to aid and bias the clustering of unlabeled data.

Why using unlabeled data

- Labeling a large set of examples is expensive!
  - Often done manually
  - Time consuming
- Unlabeled data are usually plentiful;

Outline

- Overview
- ➢ **Semi-Clustering**
- Semi-Classification

Semi-supervised clustering

- Input:
  - A set of unlabeled objects, each described by a set of attributes
  - A small amount of domain knowledge
- Output:
  - A partitioning of the objects into k clusters (possibly with some discarded as outliers)
- Objective:
  - Maximum intra-cluster similarity
  - Minimum inter-cluster similarity
  - High consistency between the partitioning and the domain knowledge

Semi-Supervised K-Means for partially labeled data

- Seeded K-Means:
  - Labeled data provided by user are used for initialization: initial center for cluster *i* is the mean of the seed points having label *i*.
  - Seed points are only used for initialization, their label may be changed in subsequent steps.
- Constrained K-Means:
  - Labeled data provided by user are used to initialize K-Means algorithm.
  - Cluster labels of seed data are kept unchanged in the cluster assignment steps, and only the labels of the non-seed data are re-estimated.

General K-Means Clustering

- K-Means is a partitional clustering algorithm based on iterative relocation that partitions a dataset into *K* clusters.

Algorithm :

Initialize k clusters centers $\{\mu_l\}_{l=1}^{k}$ randomly, Repeat until convergence.

$\Rightarrow$ Cluster Assignment Step: Assign each point x to the cluster X_l, such that L_2 distance of x from μ_l (center of X_l) is minimum;

$\Rightarrow$ Center Re-estimation Step: Re-estimate each cluster center μ_l as the mean of the points in that cluster.

Seeded K-Means

Algorithm: Seeded-KMeans

Input: Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters K, set $\mathcal{S} = \cup_{l=1}^{K} \mathcal{S}_l$ of initial seeds

Output: Disjoint K partitioning $\{\mathcal{X}_l\}_{l=1}^{K}$ of $\mathcal{X}$ such that KMeans objective function is optimized

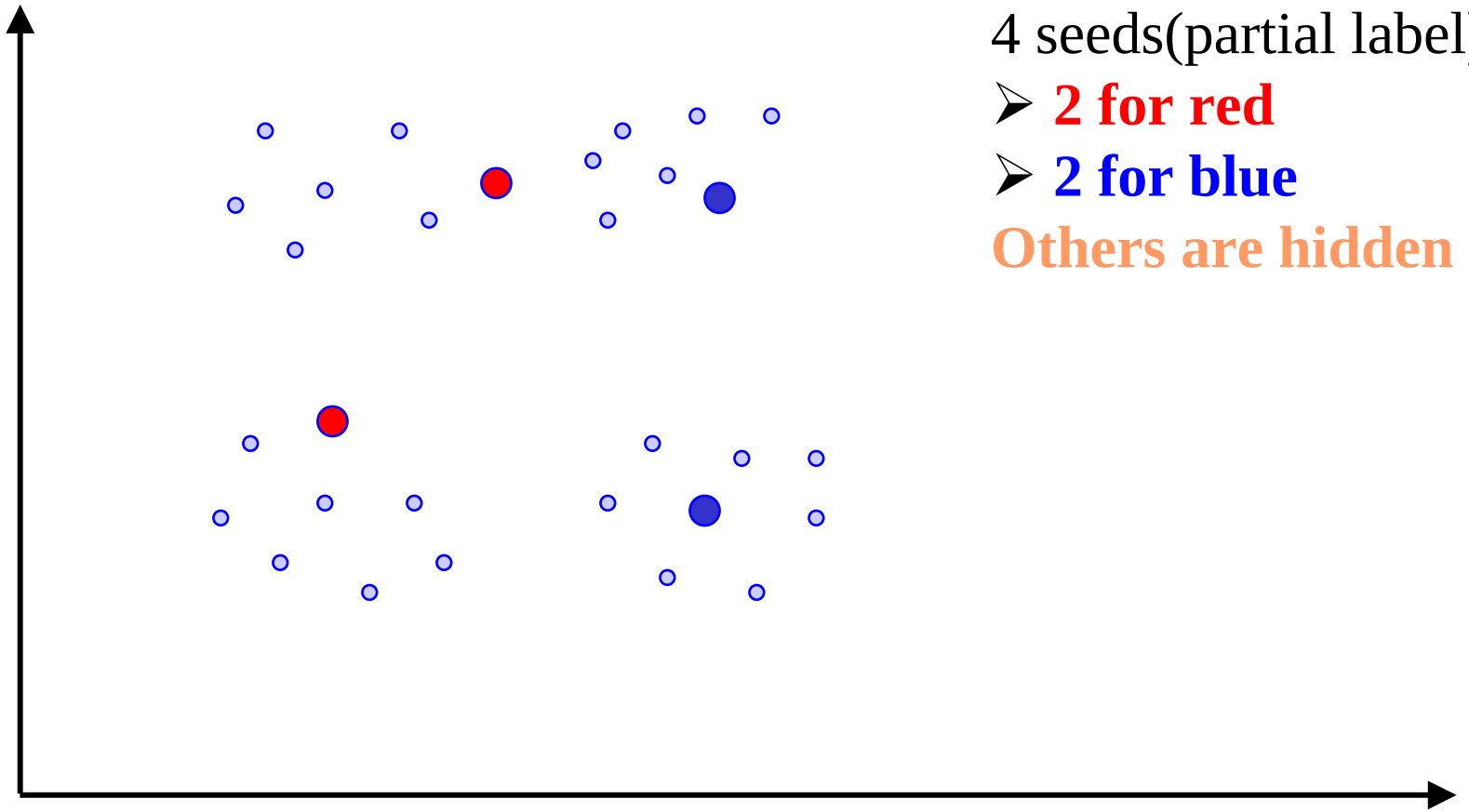
Method:

1. **intialize:** $\mu_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{x \in \mathcal{S}_h} x$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*
   - 2a. **assign_cluster:** Assign each data point x to the cluster h^* (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|x - \mu_h^{(t)}\|^2$
   - 2b. **estimate_means:** $\mu_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x \in \mathcal{X}_h^{(t+1)}} x$
   - 2c. $t \leftarrow (t+1)$

Use partial labeled data to find the initial centroids And then run K-Means.

The labels for seeded points may change.

Seeded K-Means Example



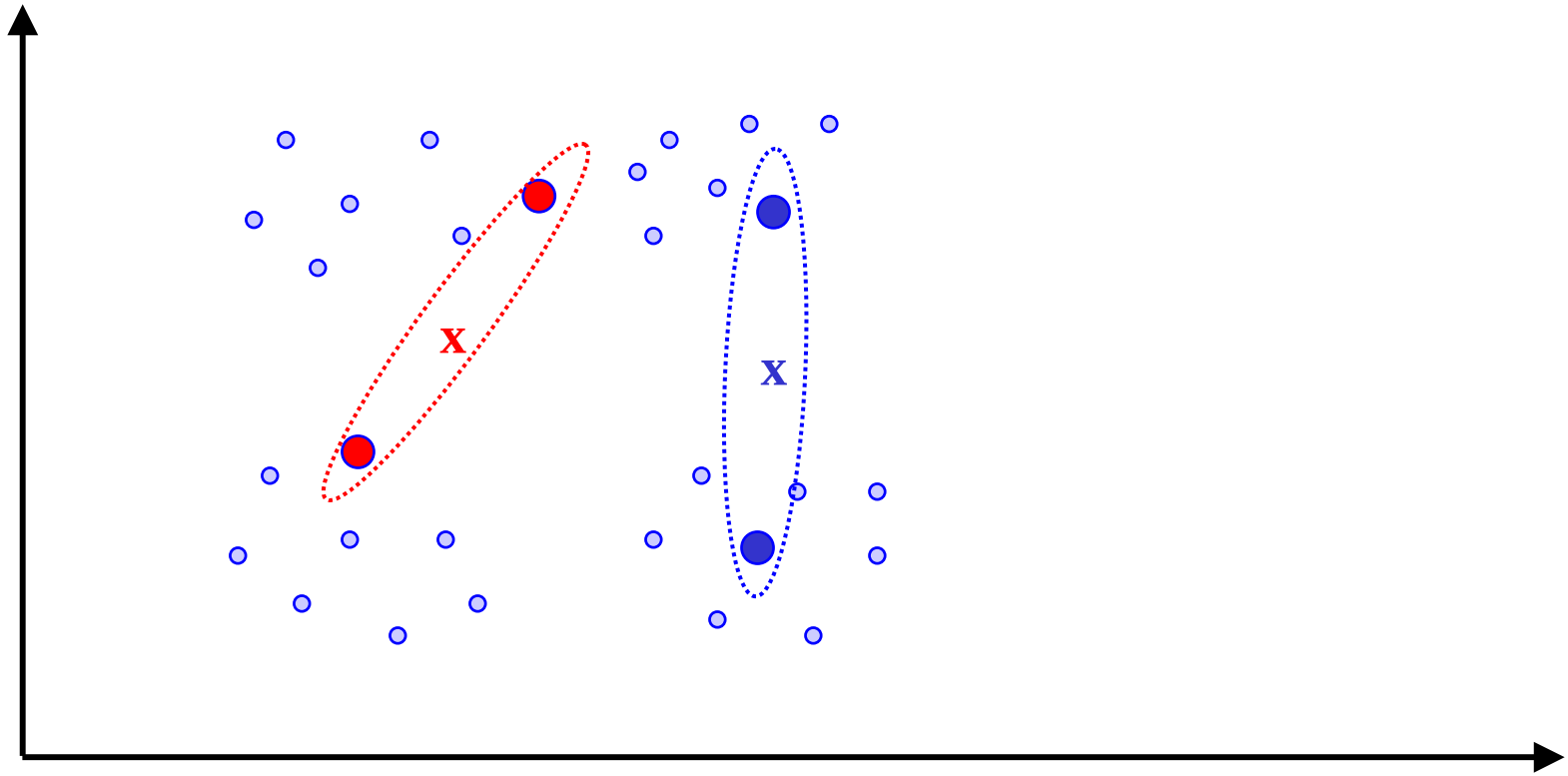
4 seeds(partial label):

- **2 for red**
- **2 for blue**

Others are hidden

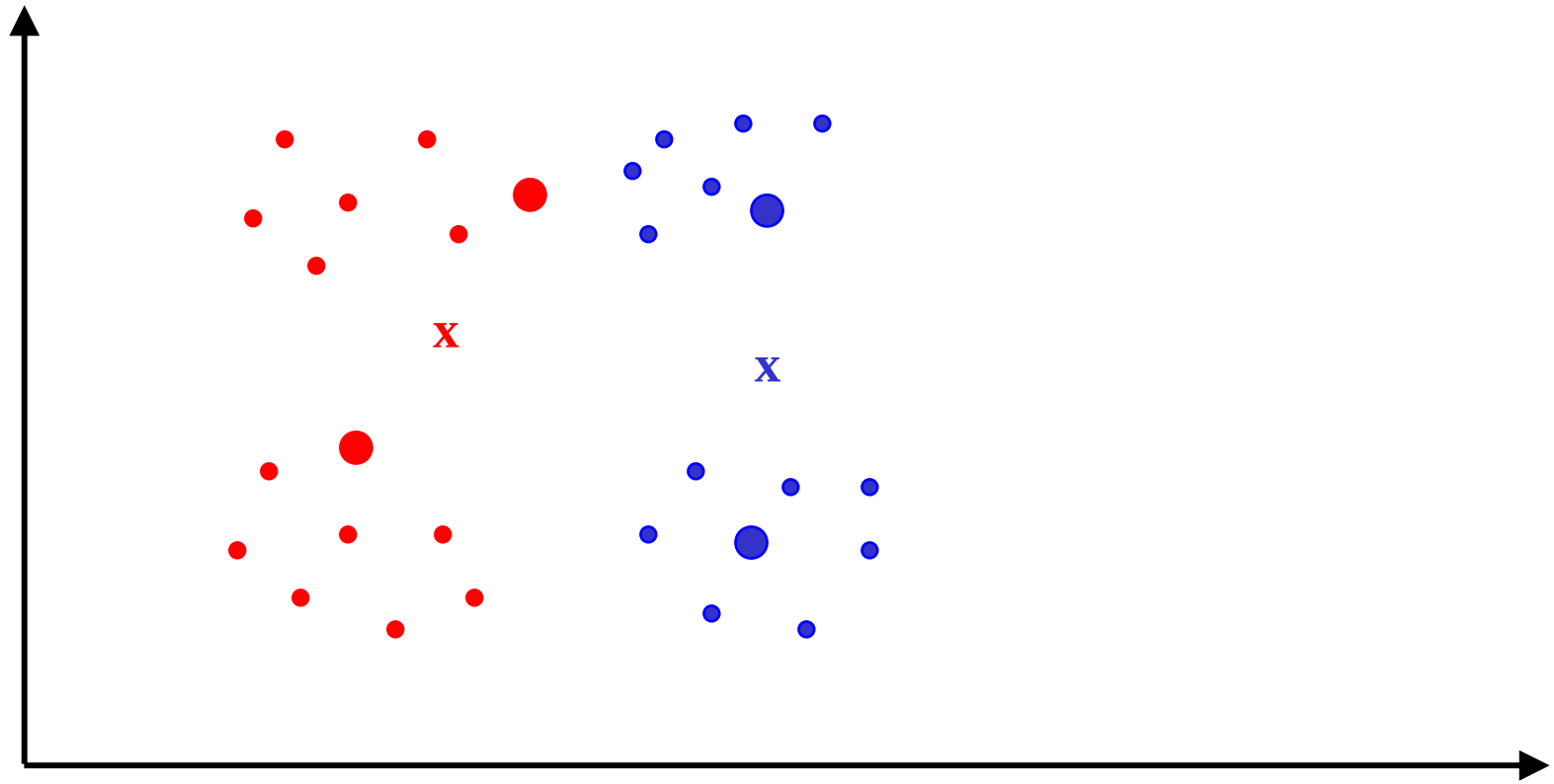
Seeded K-Means Example

- Initialize Means Using Labeled Data (i.e. **seeds**)



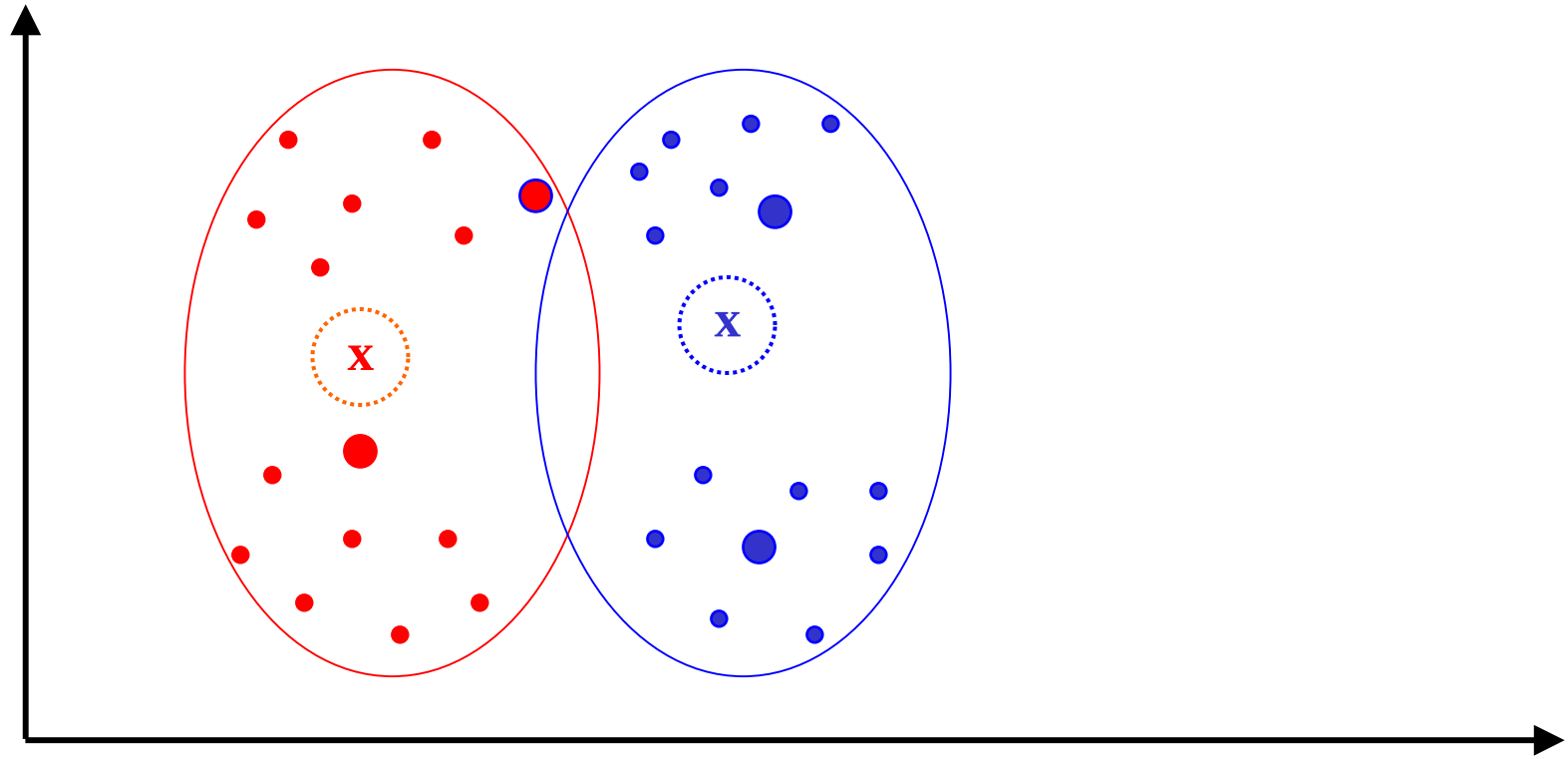
Seeded K-Means Example

- Assign Points to Clusters



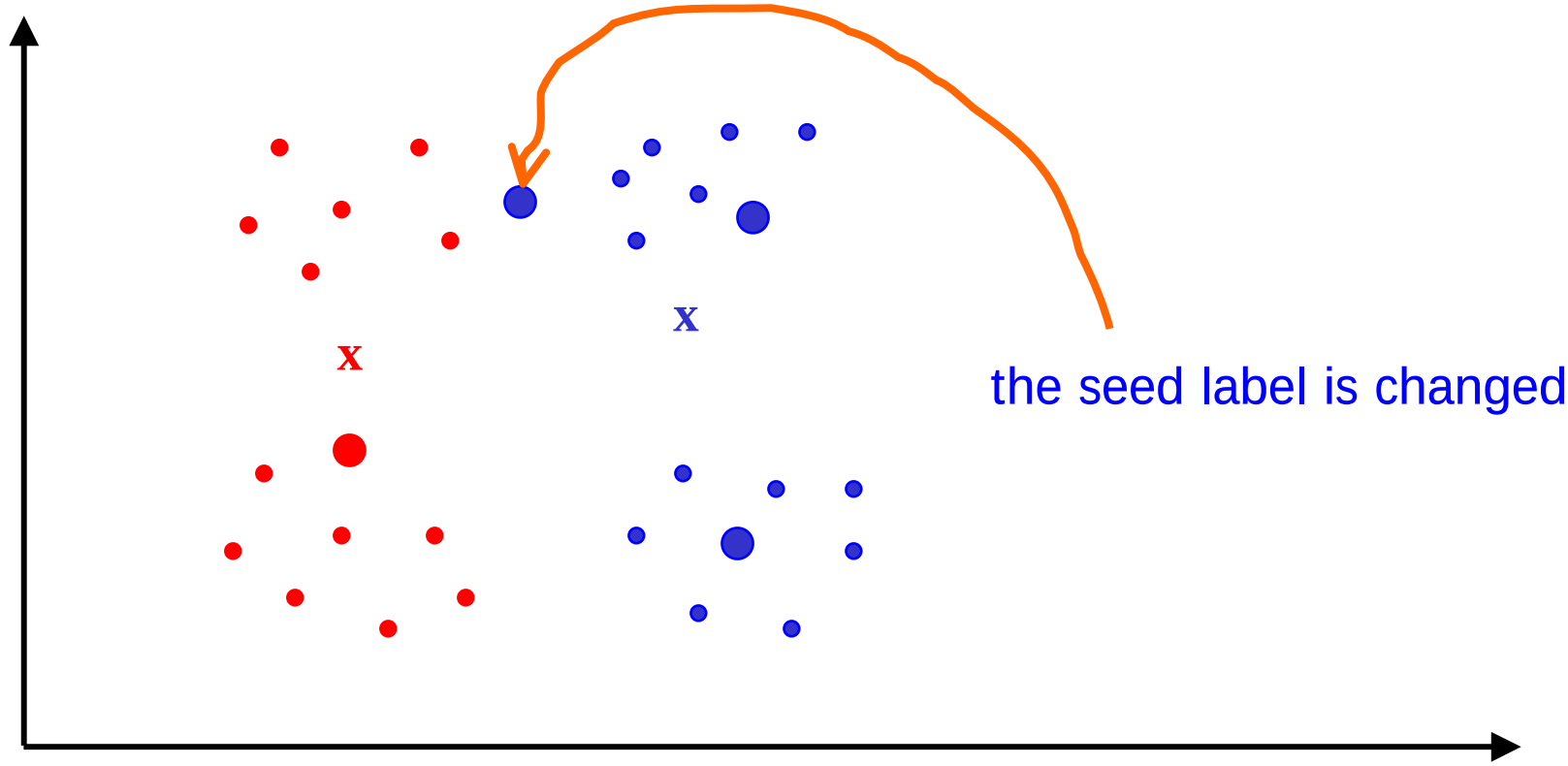
Seeded K-Means Example

- Re-estimate Means



Seeded K-Means Example

- Assign points to clusters until Converge



Constrained K-Means

Algorithm: `Constrained-KMeans`
Input: Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters K, set $\mathcal{S} = \cup_{l=1}^{K} \mathcal{S}_l$ of initial seeds
Output: Disjoint K partitioning $\{\mathcal{X}_l\}_{l=1}^{K}$ of $\mathcal{X}$ such that the KMeans objective function is optimized
Method:

1. `intialize`: $\mu_h^{(0)} \leftarrow \frac{1}{|\mathcal{S}_h|} \sum_{x \in \mathcal{S}_h} x$, for $h = 1, \ldots, K; t \leftarrow 0$
2. Repeat until *convergence*

2a. `assign_cluster`: For $x \in \mathcal{S}$, if $x \in \mathcal{S}_h$ assign x to the cluster h (i.e., set $\mathcal{X}_h^{(t+1)}$). For $x \notin \mathcal{S}$, assign x to the cluster h^* (i.e. set $\mathcal{X}_{h^*}^{(t+1)}$), for $h^* = \arg\min_h \|x - \mu_h^{(t)}\|^2$

2b. `estimate_means`: $\mu_h^{(t+1)} \leftarrow \frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x \in \mathcal{X}_h^{(t+1)}} x$

2c. $t \leftarrow (t+1)$

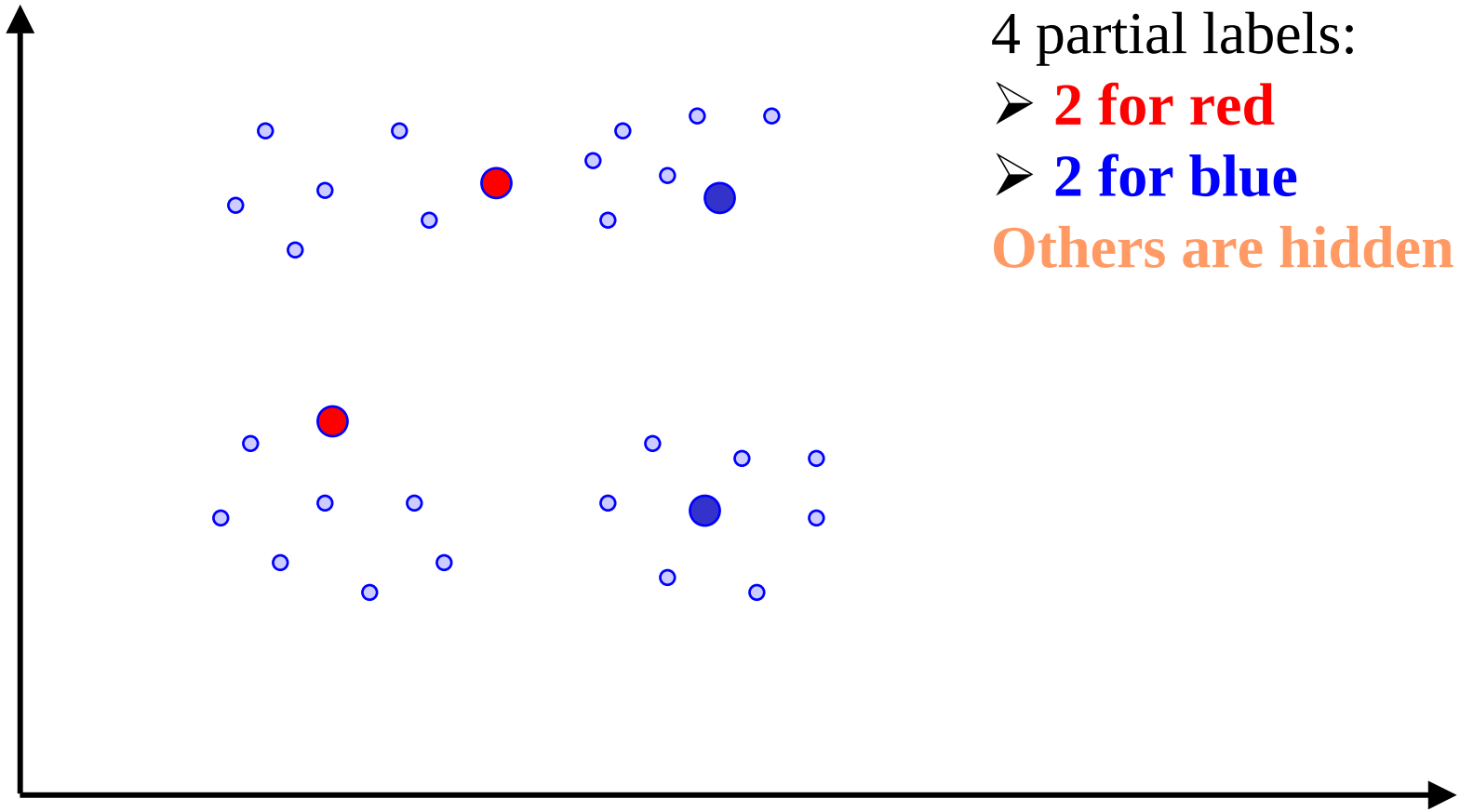
Use labeled data to find the initial centroids and then run K-Means.

The labels for seeded points will not change.

COP K-Means

- COP K-Means [Wagstaff *et al.*: ICML01] is K-Means with must-link (must be in same cluster) and cannot-link (cannot be in same cluster) constraints on data points.
- **Initialization***:* Cluster centers are chosen randomly, but as each one is chosen any must-link constraints that it participates in are enforced (so that they cannot later be chosen as the center of another cluster).
- **Algorithm***:* During cluster assignment step in COP-K-Means, a point is assigned to its nearest cluster without violating any of its constraints. If no such assignment exists, abort.

Constrained K-Means Example



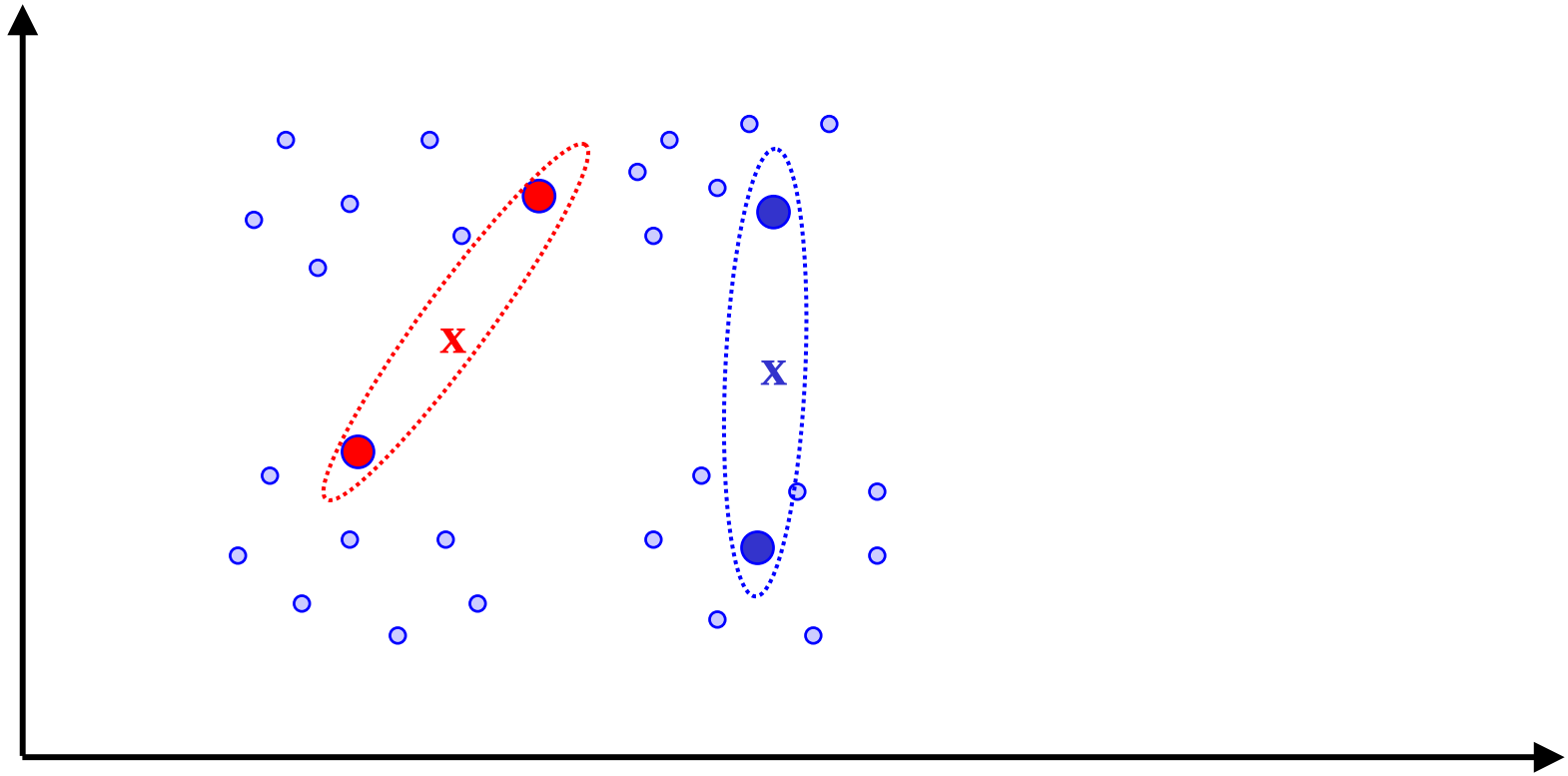
4 partial labels:

- **2 for red**
- **2 for blue**

Others are hidden

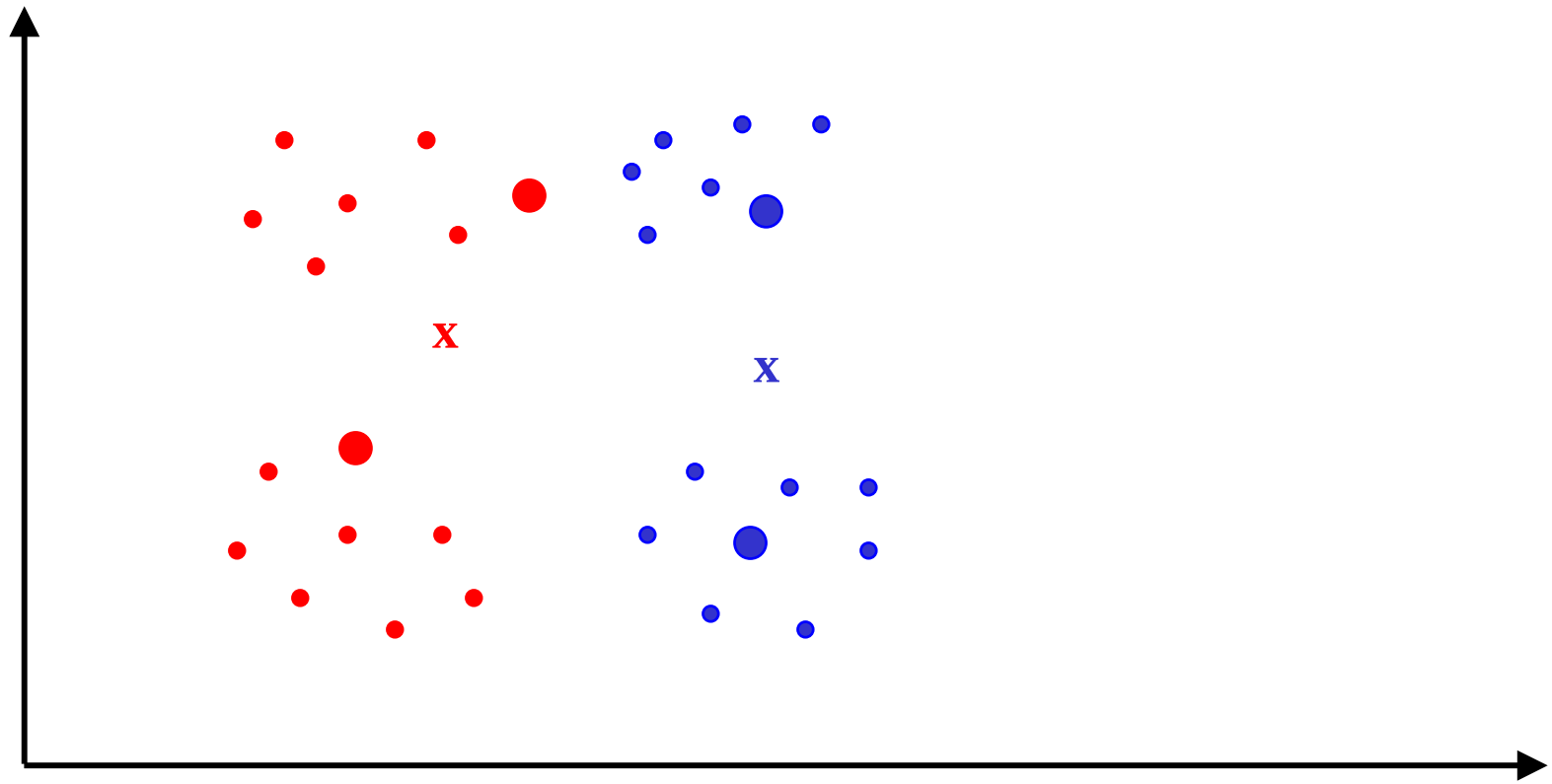
Constrained K-Means Example

- Initialize Means Using Partial Labeled Data



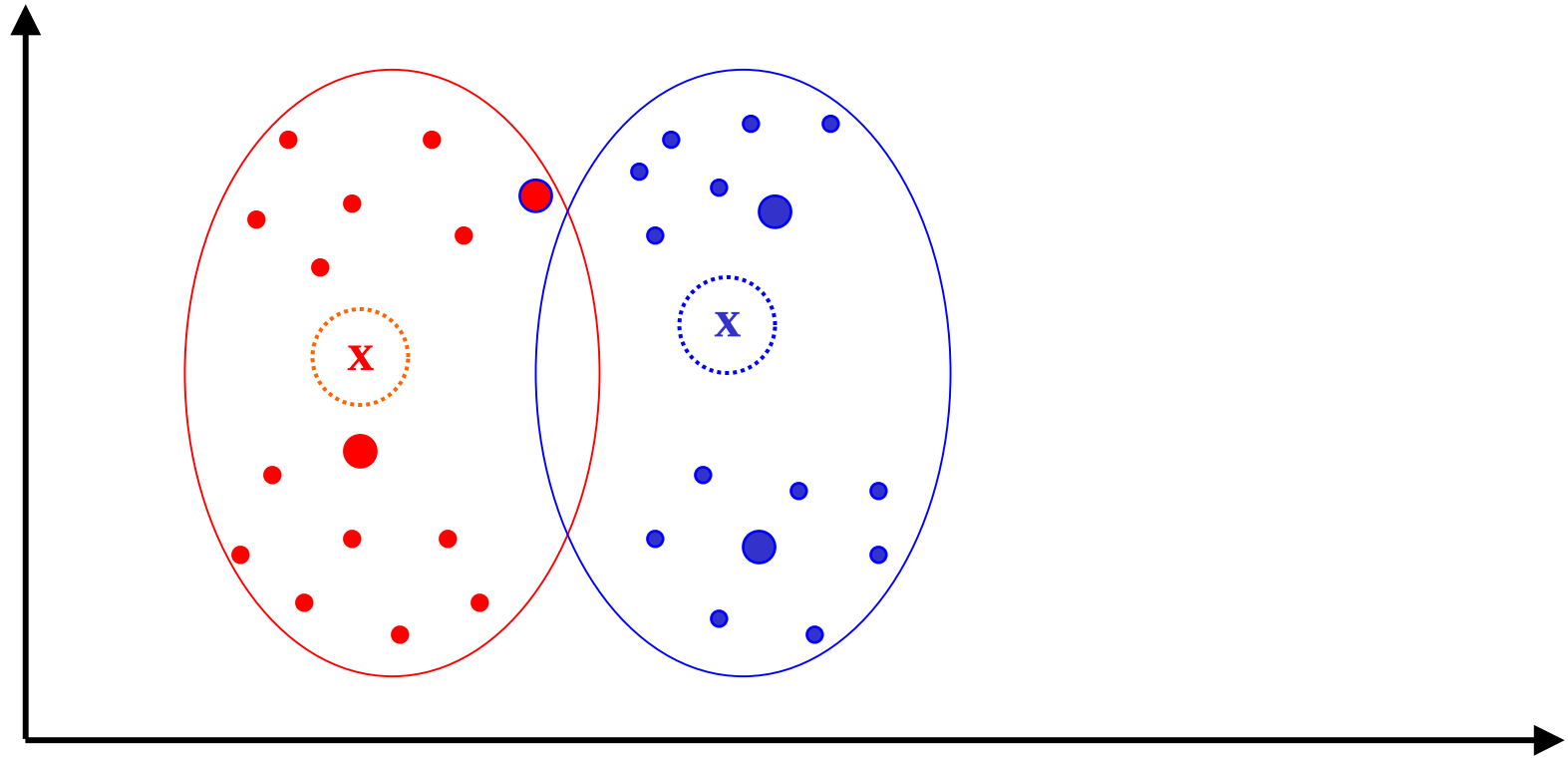
Constrained K-Means Example

- Assign Points to Clusters



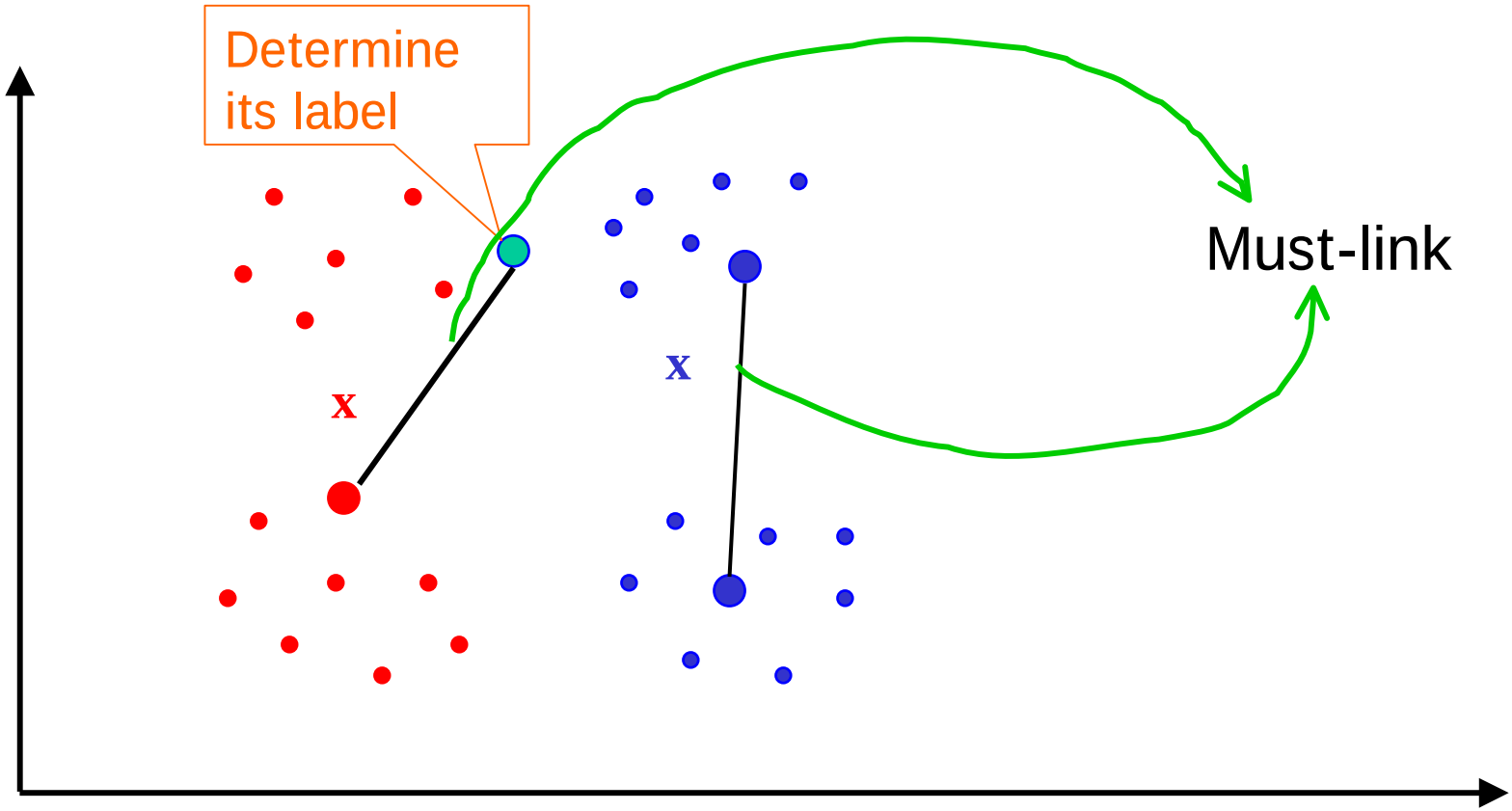
Constrained K-Means Example

- Re-estimate Means

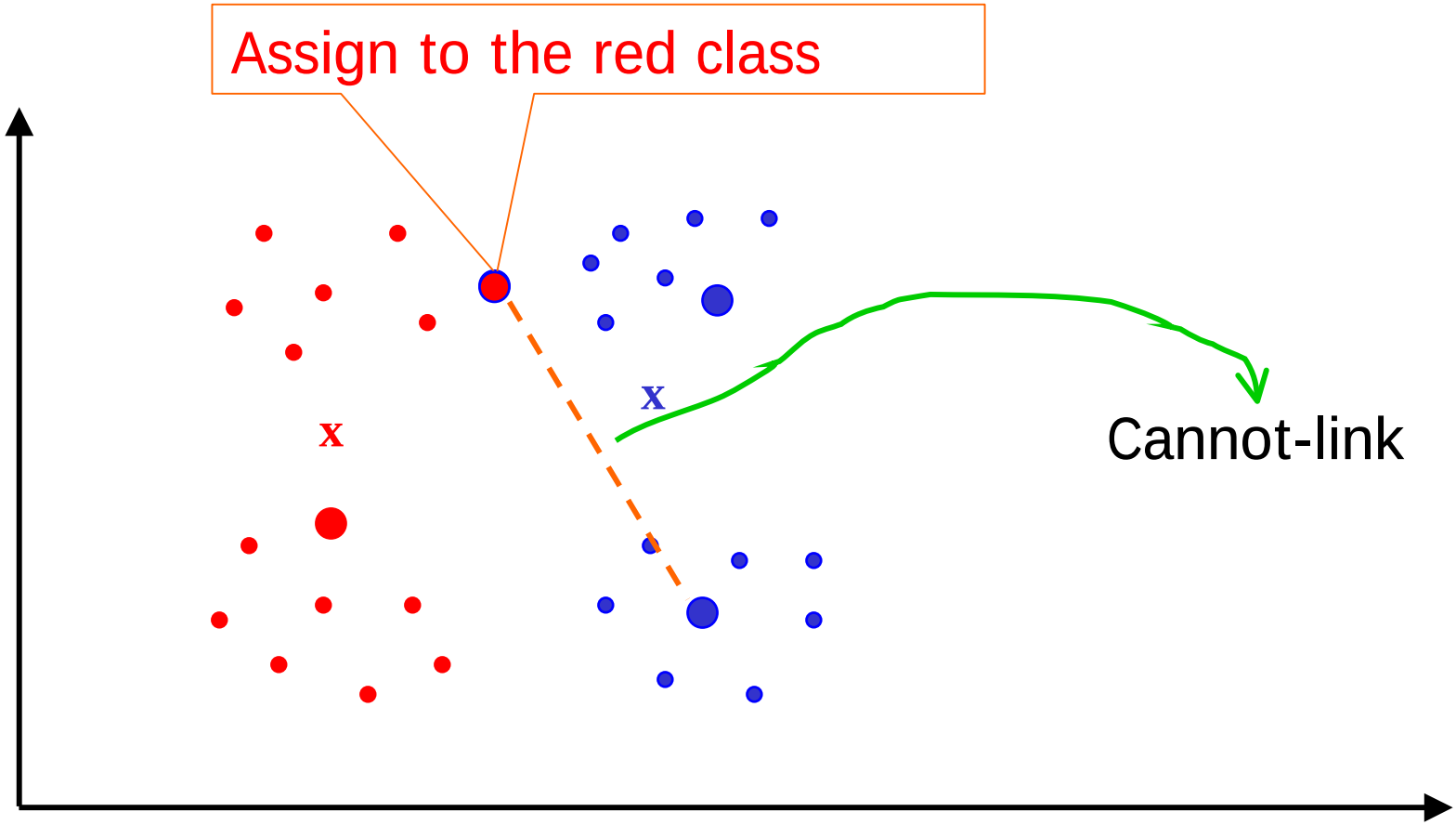


Constrained K-Means Example

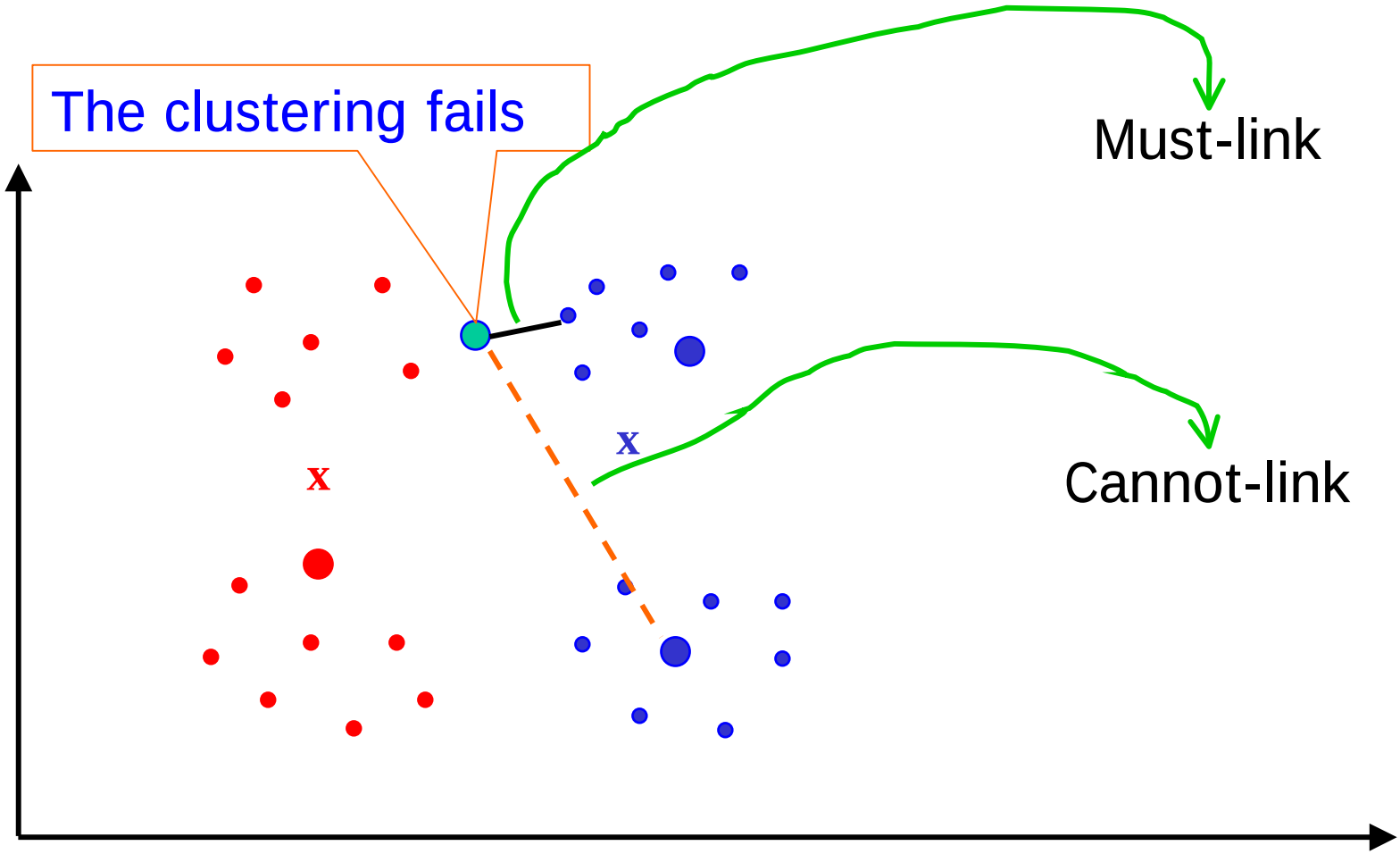
- Assign points to clusters until Converge



Constrained K-Means Example



Constrained K-Means Example



COP K-Means Algorithm

COP-KMEANS(data set D, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. Let $C_1 \ldots C_k$ be the initial cluster centers.
2. For each point d_i in D, assign it to the closest cluster C_j **such that** VIOLATE-CONSTRAINTS(d_i, C_j, $Con_=$, $Con_{\neq}$) **is false. If no such cluster exists, fail (return {}).**
3. For each cluster C_i, update its center by averaging all of the points d_j that have been assigned to it.
4. Iterate between (2) and (3) until convergence.
5. Return $\{C_1 \ldots C_k\}$.

VIOLATE-CONSTRAINTS(data point d, cluster C, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. For each $(d, d_=) \in Con_=$: If $d_= \notin C$, return true.
2. For each $(d, d_{\neq}) \in Con_{\neq}$: If $d_{\neq} \in C$, return true.
3. Otherwise, return false.

Summary I

- Seeded and Constrained K-Means: partially labeled data
- COP K-Means: constraints (Must-link and Cannot-link)
- Constrained K-Means and COP K-Means require all the constraints to be satisfied.
  - May not be effective if the seeds contain noise.
- Seeded K-Means use the seeds only in the first step to determine the initial centroids.
  - Less sensitive to the noise in the seeds.
- Experiments show that semi-supervised k-Means outperform traditional K-Means.

Outline

- Overview
- Semi-Clustering
- ➢ **Semi-Classification**

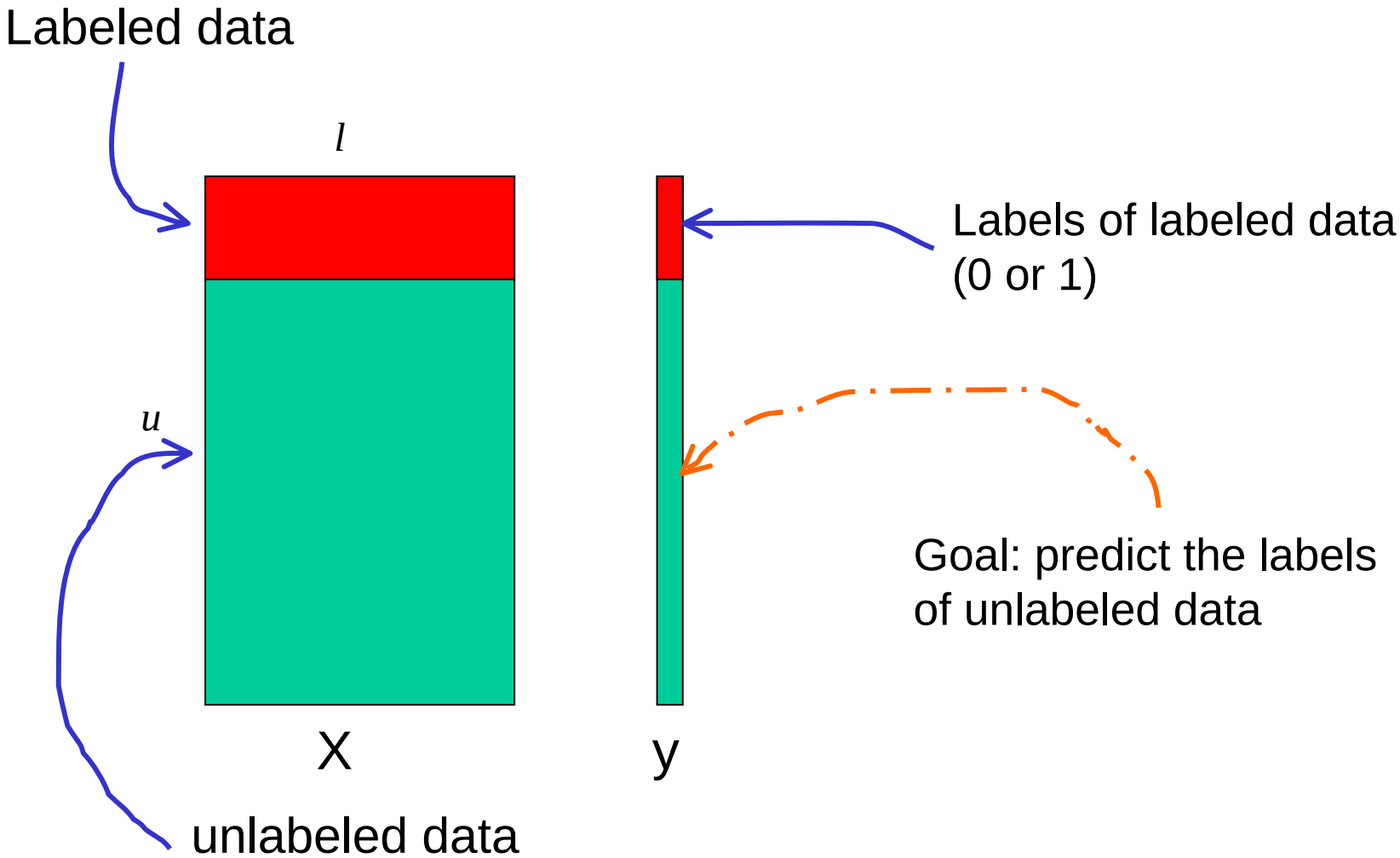
Semi-Supervised Classification

- Use small number of labeled data to label large amount of unlabeled data.
  - Labeling is expensive
- Basic idea
  - Similar data should have the same class label.
- Typical example
  - Web page classification
  - Document classification
  - Protein classification

Problem Setting

- $D = \{d_1 ... d_l, d_{l+1} ... d_m\} \in R^n$
- Label set $L = \{0,1\}$
- The first l points have been labeled $y_i \in \{0,1\}$, i=1,..,l.
- For points with i>l, y_i is unknown.

Problem Setting



More Notation

D^u: Set of unlabeled data points

D^l: Set of labeled data points

$D = D^u \cup D^l$: all data points

$\mathrm{d_i} = (x_{i1}, x_{i2}, ..., x_{in}) \in R^n$, is the ith point

C : Set of categories

Semi-Supervised Classification

- ✓ **Bootstrapping**
- Semi-supervised EM
- Co-training

Bootstrapping

- Probably the earliest Semi-Supervised Learning idea.
- Also called self-teaching or self-training
- First well-known NLP paper: (Yarowsky, 1995)

Bootstrapping

Given : D^l be the set of labeled data,

D^u be the set of unlabeled data.

Repeat :

- Train a classifier f with training data D^l
- Classify data in U with f
- Find a subset $D^{u'}$ of D^u with the most confident scores.

$$D^l \Leftarrow D^l + D^{u'}$$

$$D^u \Leftarrow D^u - D^{u'}$$

Semi-Supervised Classification

- Bootstraping
- ✓ **Semi-supervised EM**
- Co-training
-

Text Classification again

Naïve Bayes Model

- Vocabulary: $V = (w_1, w_2, ..., w_{|V|})$

  w_i is a word in the vocabulary

- Document: $d_i = <w_{d_{i1}}, w_{d_{i2}}, ...>$

  $w_{d_{ij}}$ is a word in position j of document i

- Two more assumptions:
  a. Word independence
  b. Document length independence

Probabilistic Framework

- Two assumptions
  - The data are produced by a mixture model,

θ : *parameter* (*a probability distribution*),

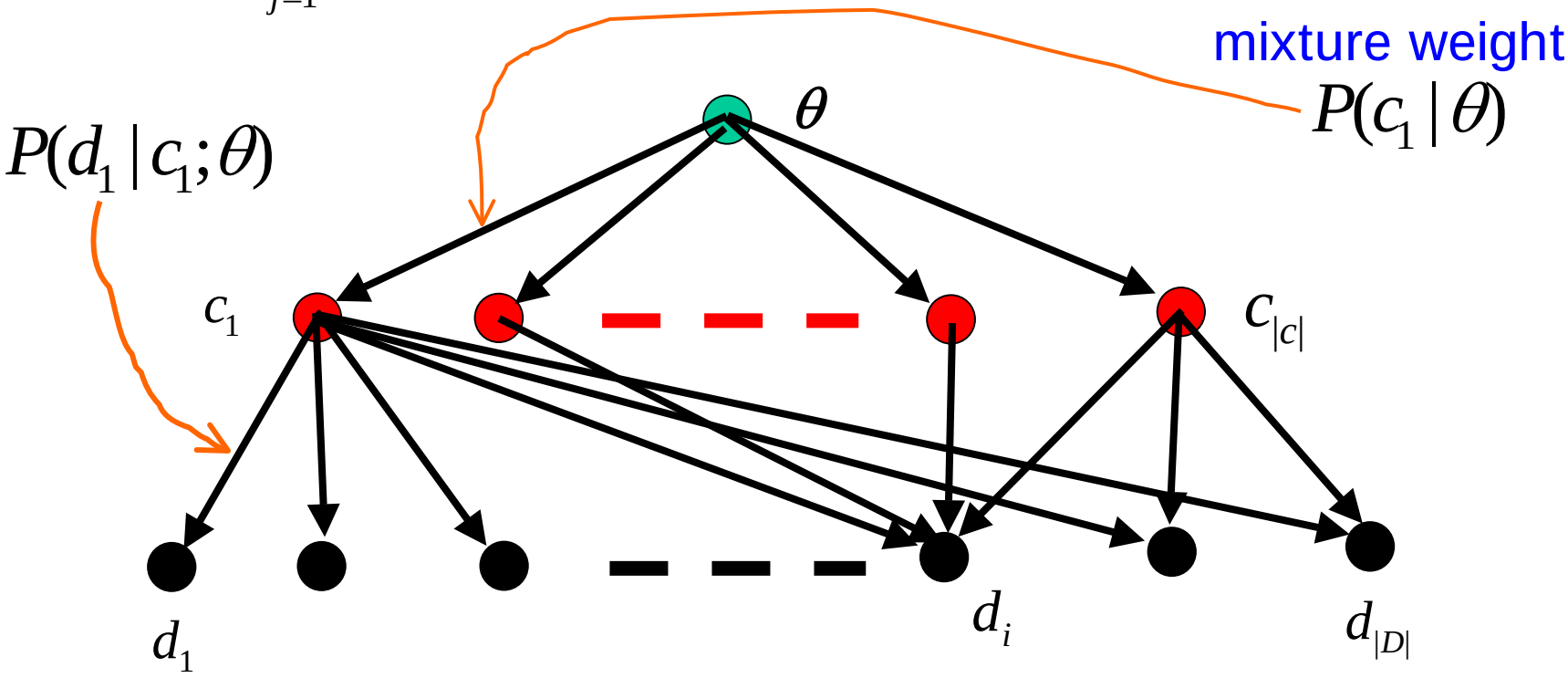
a mixture components and class labels $c_j \in C = \{c_1, ..., c_{|C|}\}$

  - There is a one-to-one correspondence between mixture components and document classes

Probabilistic Framework

The likelihood of a point(document) d_i is:

$$P(d_i \mid \theta) = \sum_{j=1}^{|C|} P(c_j \mid \theta) P(d_i \mid c_j; \theta)$$



Document Generative Model

- Generating a document
  - Step 1. Selecting a mixture component according to the mixture weight $P(c_j \mid \theta)$
  - Step 2. Having the selected mixture component generates a document according to its own distribution $P(d_i \mid c_j; \theta)$
  - Step 3, The likelihood of a document is

$$P(d_i \mid \theta) = \sum_{j=1}^{|C|} P(c_j \mid \theta) P(d_i \mid c_j; \theta)$$

$\hat{\theta}_{c_j}$ $\hat{\theta}_{w_t \mid c_j}$

Supervised: Naïve Bayes Learner

The maximization yields parameters that are word frequency counts:

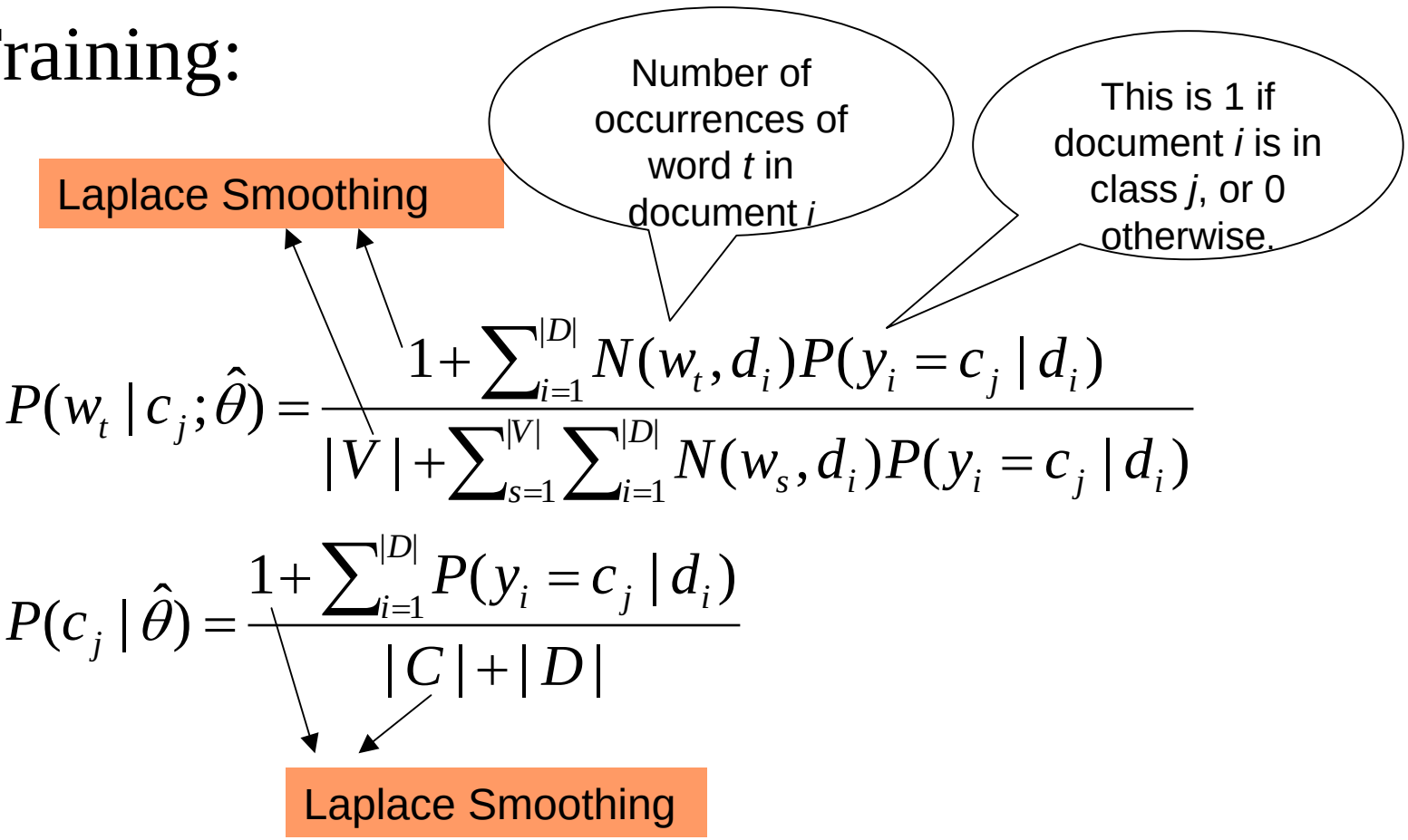
$$\hat{\theta}_{w_t|c_j} = \frac{1 + \text{No. of occurrences of } w_t \text{ in class } j}{|V| + \text{No. of words in class } j}$$

$$\hat{\theta}_{c_j} = \frac{1 + \text{No. of documents in class } j}{|C| + |D|}$$

Laplace smoothing gives each word a prior probability.

Supervised: Naïve Bayes Learner

Training:



Naïve Bayes Classification

Using the classifier

$$\mathrm{P}(y_i = c_j | d_i; \hat{\theta}) = \frac{\mathrm{P}(c_j|\hat{\theta})\mathrm{P}(d_i|c_j;\hat{\theta})}{\mathrm{P}(d_i|\hat{\theta})}$$

$$= \frac{\mathrm{P}(c_j|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_j;\hat{\theta})}{\sum_{r=1}^{|\mathcal{C}|}\mathrm{P}(c_r|\hat{\theta})\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_r;\hat{\theta})}$$

Class priors

Conditional probabilities

Three cases

- If all data is labeled, corresponds to supervised training of Naïve Bayes classifier;
- If all data unlabeled, corresponds to unsupervied, clustering;
- If both labeled and unlabeled data, then unlabeled data helps if the Bayes net model is correct

How to use unlabeled data

- One way is to use the EM algorithm
  - **EM: Expectation Maximization**
- The EM algorithm is a popular iterative algorithm for maximum likelihood estimation in problems with missing data.
- The EM algorithm consists of two steps,
  - ***Expectation* step**, i.e., filling in the missing data
  - ***Maximization* step –** calculate a new maximum *a posteriori* estimate for the parameters.

Incorporating unlabeled Data with EM

- Basic EM
- Augmented EM with weighted unlabeled data
- Augmented EM with multiple mixture components per class

Algorithm Outline

1. Train a classifier with only the labeled documents.
2. Use it to probabilistically classify the unlabeled documents.
3. Use **ALL** the documents to train a new classifier.
4. Iterate steps 2 and 3 to convergence.

General EM

E-step:

$$Q(\theta,\theta^t)=\sum_{m=1}^{M}\sum_{y}P(y\mid D_m,\theta^t)\log P(D_m,y\mid\theta)$$

M-step:

$$\theta^{(t+1)}=\arg\max_{\theta}Q(\theta,\theta^t)$$

EM

E Step: $$P\left(c_j \mid d_i, \hat{\theta}\right) = \frac{P\left(c_j \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_j; \hat{\theta}\right)}{\sum_{r=1}^{|C|} P\left(c_r \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_r; \hat{\theta}\right)}$$

M Step:

$$P(w_t \mid c_j; \hat{\theta}) = \frac{1+\sum_{i=1}^{|D|} N(w_t, d_i)P(y_i = c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N(w_s, d_i)P(y_i = c_j \mid d_i)}$$

$$P(c_j \mid \hat{\theta}) = \frac{1+\sum_{i=1}^{|D|} P(y_i = c_j \mid d_i)}{|C| + |D|}$$

An example

- **C0**: aba ab bc **C1**: bcb cc ac
- Unlabeled data: abc aca
- Vocabulary = {a, b, c}. |V|=3; |C|=2;

Laplace Smoothing

- P(a|**C0**) = (1+3)/(3+7)=2/5 P(a|**C1**)=(1+1)/(3+7)=1/5
- P(b|**C0**)=(1+3)/(3+7)=2/5 P(b|**C1**)=(1+2)/(3+7)=3/10
- P(c|**C0**)= (1+1)/(3+7) = 1/5 P(c|**C1**)=(1+4)/(3+7)=1/2
- P(**C0**) = (1+3)/(2+6)=1/2 P(**C1**) = (1+3)/(2+6)=1/2

$\hat{\theta}$

- Classify unlabeled data: **abc; aca** using NB

An example :E-step

E Step: $$P\left(c_j \mid d_i, \hat{\theta}\right) = \frac{P\left(c_j \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_j; \hat{\theta}\right)}{\sum_{r=1}^{|C|} P\left(c_r \mid \hat{\theta}\right)\prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}} \mid c_r; \hat{\theta}\right)}$$

- P(**C0**|abc) = P(**C0**)P(abc|**C0**)/ (P(**C0**)P(abc|**C0**)+P(**C1**)P(abc|**C1**))
  = (0.5*2/5*2/5*1/5)/(0.5*2/5*2/5*1/5 +0.5*1/5*3/1081/2)
  = 16/31

- P(**C0**|aca) = P(**C0**)P(aca|**C0**)/(P(**C0**)P(aca|**C0**)+P(**C1**)P(aca|**C1**))
  = (0.5*2/5*1/5*2/5)/(0.5*2/5*1/5*2/5 + 0.5*1/5*1/2*1/5)
  = 8/13

- P(**C1**|abc) = 15/31
- P(**C1**|aca) = 5/13

An example: Update parameters

- C0: aba ab bc abc (16/31) aca (8/13)
- C1: bcb cc ac abc (15/31) aca (5/13)
- Learn new parameters
  - P(a|C0) = (1+3 + 1*16/31 + 2*8/13)/(3+7+3*16/31+3*8/13)=?
  - P(a|C1) = (1+1+1*15/31 + 2*5/13)/(3+7+3*15/31+3*5/13) = ?
  - P(b|C0) = ? P(b|C1) = ? P(c|C0) = ? P(c|C1) = ?

  - P(C0) = (1+3+16/31+8/13)/(2+**8**) = ?
  - P(C1) = (1+3+15/31+5/13)/(2+**8**) = ?

$\hat{\theta}$

- Classify unlabeled data
- Repeat until convergence (EM)

$$P(w_t \mid c_j; \hat{\theta}) = \frac{1+\sum_{i=1}^{|D|} N(w_t, d_i) P(y_i = c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) P(y_i = c_j \mid d_i)}$$

$$P(c_j \mid \hat{\theta}) = \frac{1+\sum_{i=1}^{|D|} P(y_i = c_j \mid d_i)}{|C| + |D|}$$

The Problem

- Suppose you have a few labeled documents and many more unlabeled documents.
- Then the algorithm almost becomes unsupervised clustering! The only function of the labeled data is to assign class labels to the mixture components.
- *When the mixture-model assumptions are not true, the basic algorithm will find components that don't correspond to different class labels.*

The solution: EM-λ

Modulate the influence of unlabeled data with a parameter λ, $0 \le \lambda \le 1$. And maximize:

M Step:

Weight

Word count

Probabilistic class assignment

$$P(w_t \mid c_j; \hat{\theta}) = \frac{1 + \sum_{i=1}^{|D|} \Lambda(i) N(w_t, d_i) P(y_i = c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) P(y_i = c_j \mid d_i)}$$

sum over all words and documents

$$P(c_j \mid \hat{\theta}) = \frac{1 + \sum_{i=1}^{|D|} \Lambda(i) P(y_i = c_j \mid d_i)}{|C| + |D^l| + \lambda |D^u|}$$

$$\Lambda(i) = \begin{cases} \lambda, & \text{if } d_i \in D^u; \\ 1, & \text{if } d_i \in D^l; \end{cases}$$

Semi-Supervised Classification

- Bootstraping
- Semi-supervised EM
- ✓ **Co-training**

Co-Training

- In some settings, available data features are redundant and we can train two (or many) classifiers based on disjoint features;
- In this case, the two classifiers should agree on the classification for each unlabeled example;
- Therefore, we can use the unlabeled data to constrain joint training of both classifiers

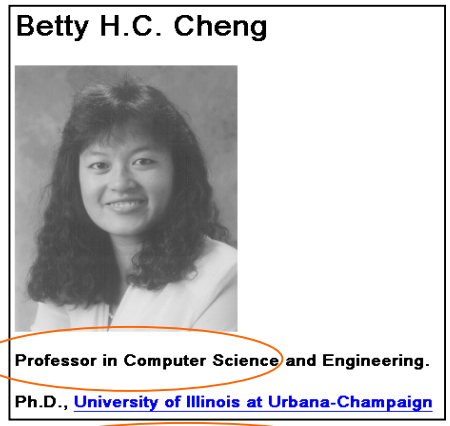
An example

- Web-page classification
  - e.g., find homepages of CS faculty members.
- Two kinds of features are helpful:
  - Page text (content): words occurring on that page

    e.g., “research interest”, “teaching”…
  - Hyperlink text (link) : words occurring in hyperlinks that point to that page:

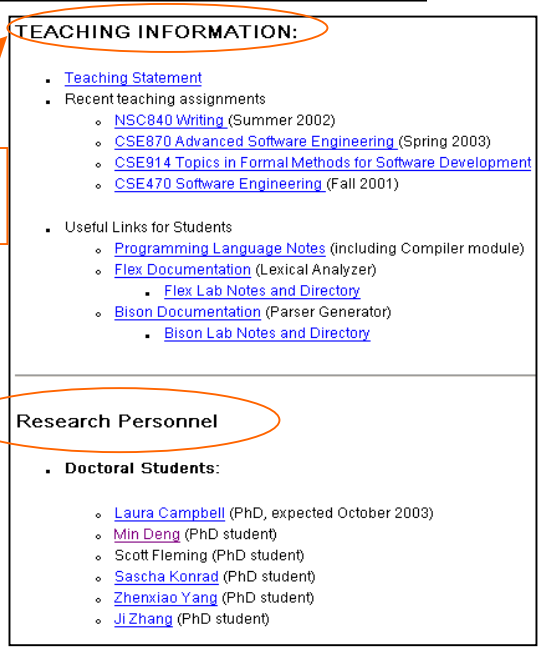
    e.g., “(my)advisor”

An example

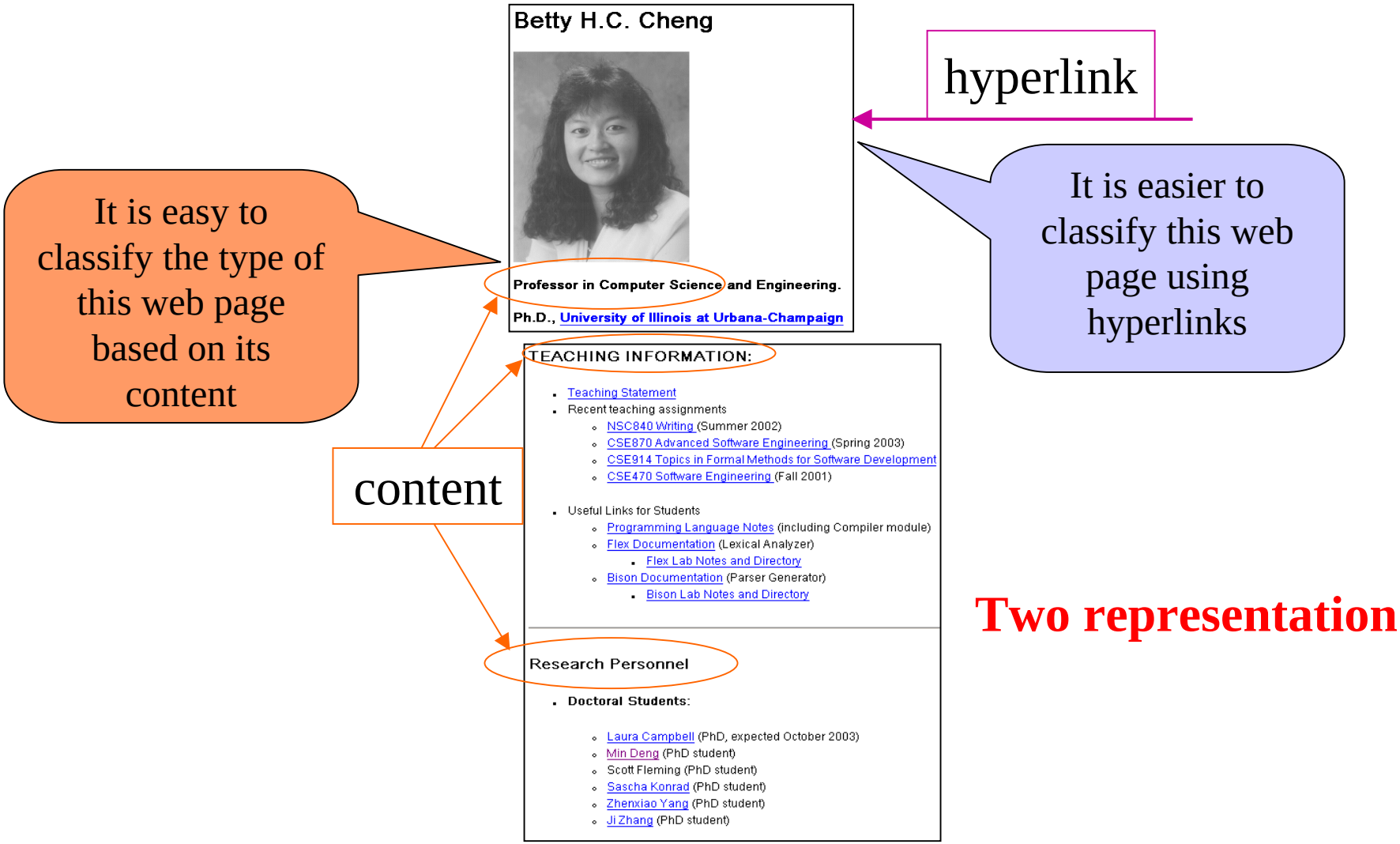


hyperlink

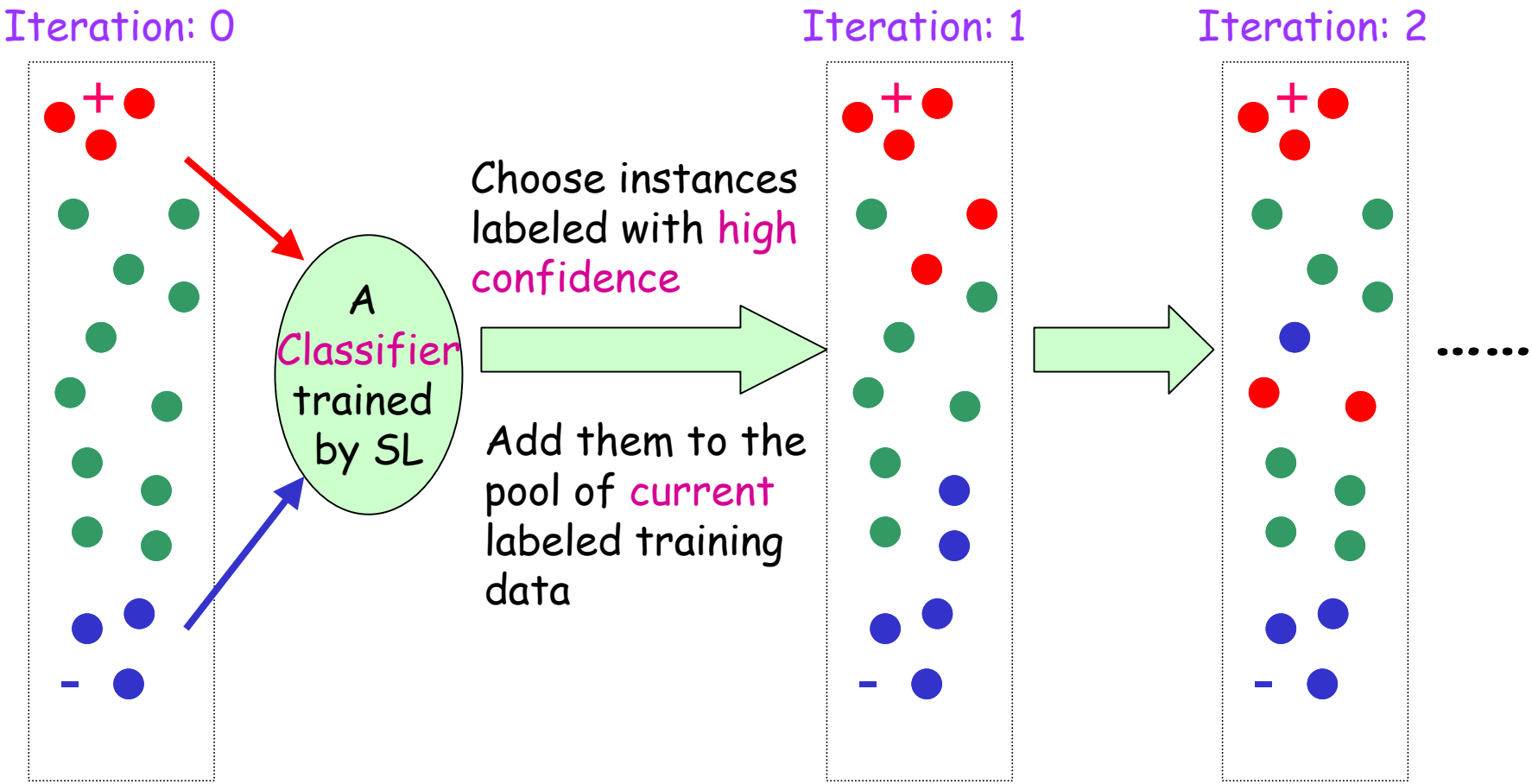
content



Redundantly Sufficient Features



Co-Training


Iteration: 0
Iteration: 1
Iteration: 2
+
-
A Classifier trained by SL
Choose instances labeled with high confidence
Add them to the pool of current labeled training data
.....

Co-Training

- A Problem Setting may Provide Redundantly Sufficient Features

learn $f : D \to Y$

Each point $d_i \in D$ is described as $\mathbf{x}=(x_1, x_2, ..., x_n)$

$\mathbf{x}$ can be splited into two feature set: $\mathbf{x}=\mathbf{x}_1 \times \mathbf{x}_2$

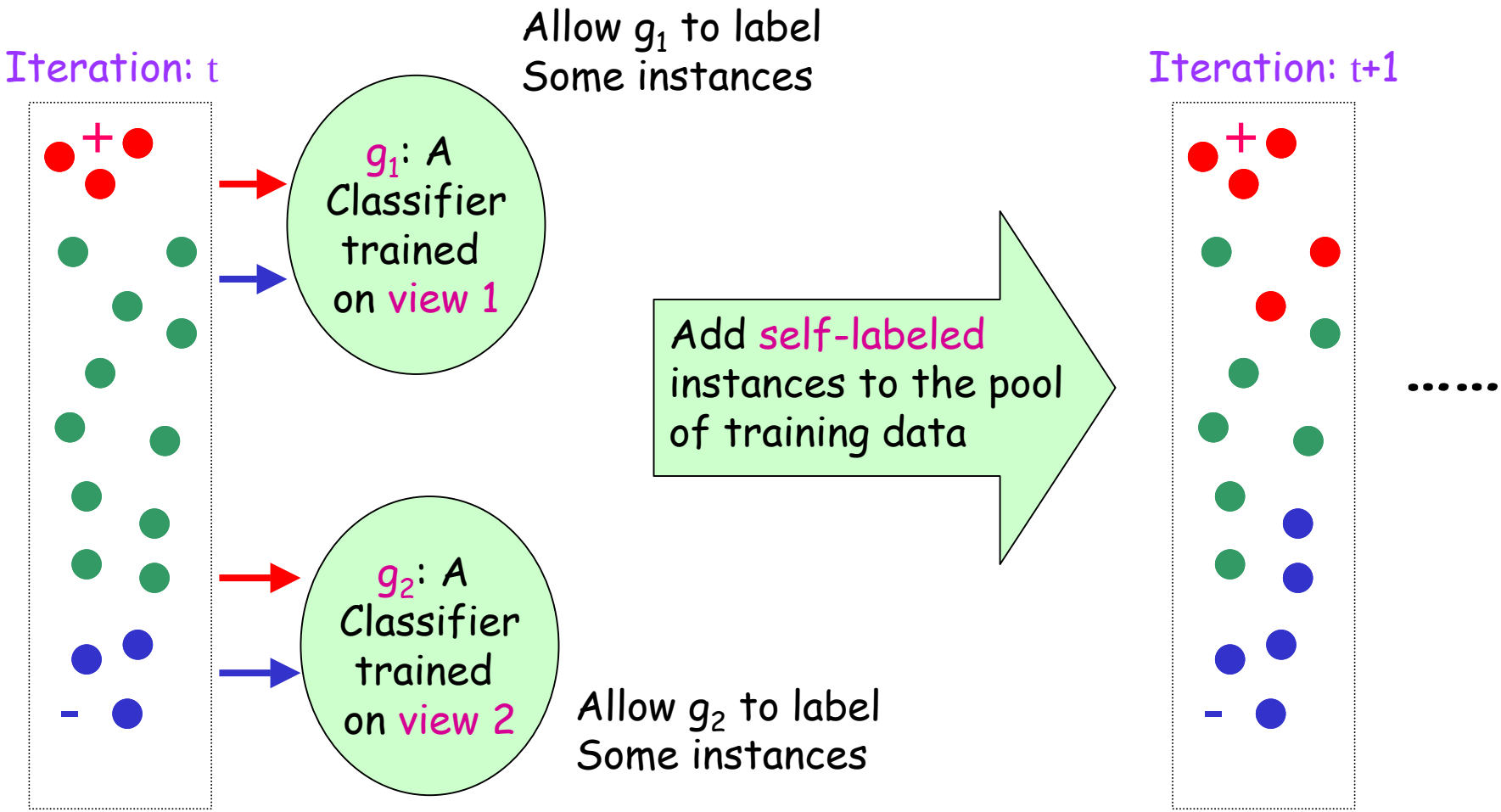
Each feature (e.g. $\mathbf{x}_1$ or $\mathbf{x}_2$) is called a view.

and $\exists g_1, g_2 \quad (\forall \mathbf{x}) g_1(\mathbf{x}_1) = g_2(\mathbf{x}_2) = f(\mathbf{x})$

C_1 : the set of target concept class over $\mathbf{x}_1$, $g_1 \in C_1$;

C_2 : the set of target concept class over $\mathbf{x}_2$, $g_2 \in C_2$;

Co-Training


Iteration: t
Allow g_1 to label
Some instances
g_1: A Classifier trained on view 1
Add self-labeled instances to the pool of training data
Iteration: t+1
.....
g_2: A Classifier trained on view 2
Allow g_2 to label
Some instances

Compatibility

- Let **D** be a distribution over instance space, C_1 and C_2 is concept classes over X_1 and X_2
- Let the combined target concept class be $f=(g_1, g_2)$
- For any example $x=(x_1, x_2)$ observed with label *l* :

  $f(x) = g_1(x_1) = g_2(x_2)=l$. i.e. assign probability zero to any example $x=(x_1, x_2)$ such that $g_1(x_1) \neq g_2(x_2)$

  ➔ Each set of features is sufficient for classification

- The compatibility of f with **D** :

$$p = 1 - \Pr[(x_1, x_2) : f_1(x_1) \neq f_2(x_2)]$$

Conditional independence

Definition: A pair of views $\mathbf{x}_1$ and $\mathbf{x}_2$ satisfy view independence just in case:

$$\Pr(X_1 = x_1 \mid X_2 = x_2, Y = y) = \Pr(X_1 = x_1 \mid Y = y)$$
$$\Pr(X_2 = x_2 \mid X_1 = x_1, Y = y) = \Pr(X_2 = x_2 \mid Y = y)$$

- A classification problem instance satisfies view independence just in case all pairs x_1, x_2 satify view independence.

Co-Training Algorithm

Given: labeled data L,

unlabeled data U,

Loop:

Train g_1 (eg. hyperlink classifier) using L

Allow g_1 to label the examples from U

Label the examples that have been confidently classified and add them into L

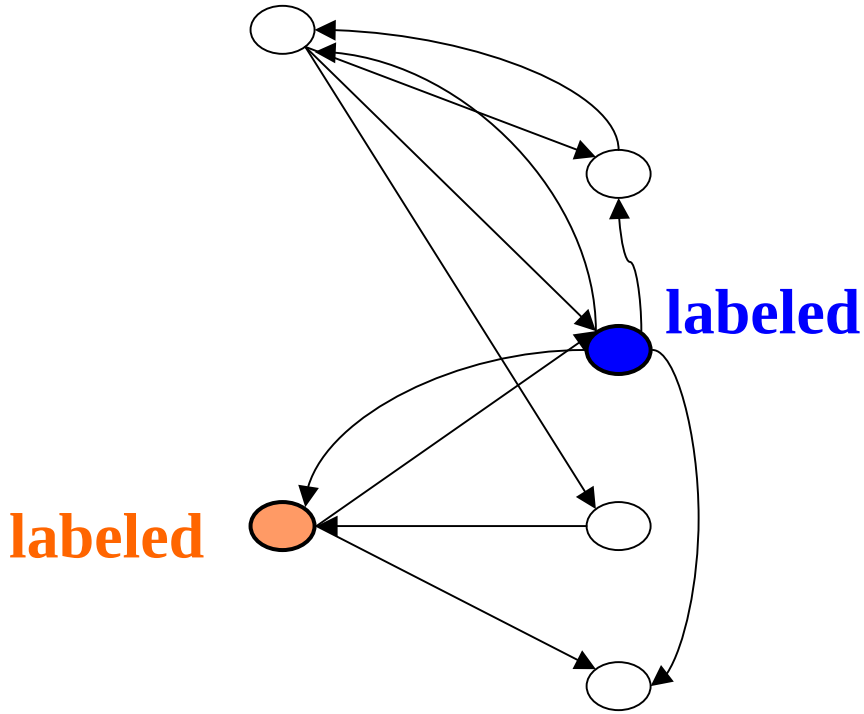
Train g_2 (eg. page classifier) using L

Allow g_2 to label the examples from U

Label the examples that have been confidently classified and add them into L

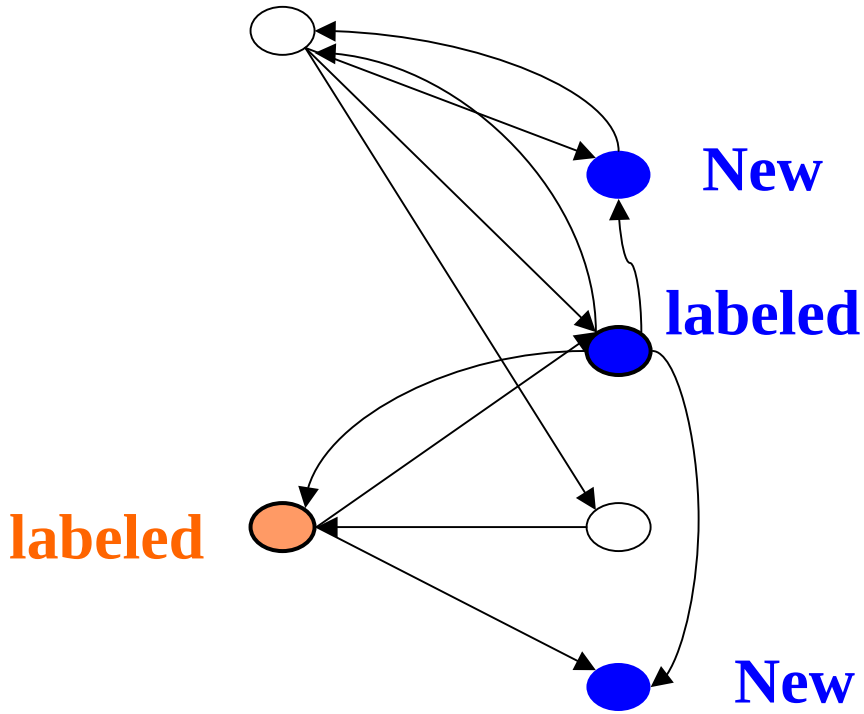
Example: Co-training

- Train classifier g_1



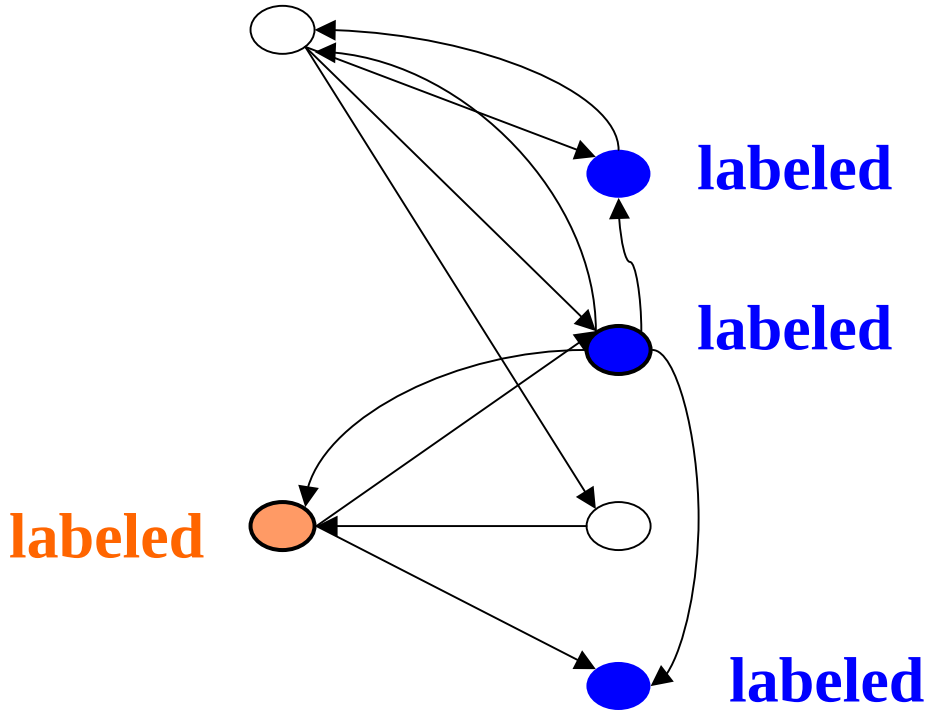
Example: Co-training

- Label the unlabeled examples with g_1 that are confidently classified



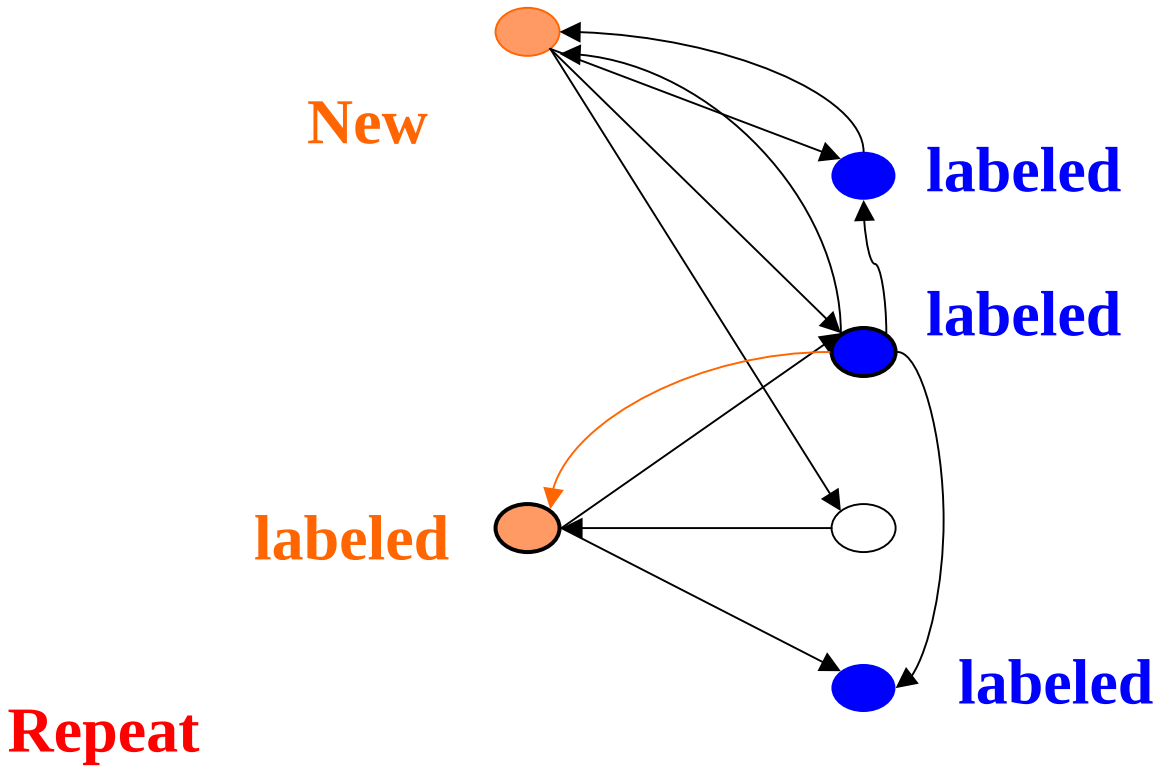
Example: Co-training

- Train classifier g_2



Example: Co-training

- Label the unlabeled examples with g_2 that are confidently classified



Improvement

[Blum and Mitchell, 1998]

Given:

- a set L of labeled training examples;
- a set U of unlabeled examples;

Create a pool **U'** of examples by choosing u examples at random from U

Loop for k iterations

1) Use L to train a classier g_1 that considers only the x_1 portion of x
2) Use L to train a classier g_2 that considers only the x_2 portion of x
3) Allow g_1 to label p positive and n negative examples from **U'**
4) Allow g_2 to label p positive and n negative examples from **U'**
5) Add these self-labeled examples to L
6) Randomly choose (2p+2n) examples from U to replenish **U'**

How to train each classifier?

- Use Naïve Bayes (or others) to train each classifier.

  $P_1(c|\mathbf{x_1})$ and $P_2(c|\mathbf{x_2})$, where, $c \in C$

- Classification:
  - Use each classifier to label p positive example and n negative ones
  - Or, combining both classifiers to one:

    $P(c|x)=P_1(c|x_1)P_2(c|x_2)$

Intuition behind the co-training algorithm

- g_1 adds examples to the labeled set that g_2 will be able to use for learning, and vice verse.
- If the conditional independence assumption holds, the learning will progress.

Co-training vs. EM

- Co-training splits features, EM does not.
- Co-training **incrementally** uses the unlabeled data.
- EM probabilistically labels all the data at each round; EM **iteratively** uses the unlabeled data.

Assumptions

- Assumptions made by the underlying classifier (supervised learner):
  - Naïve Bayes: words occur independently of each other, given the class of the document.
  - Co-training uses the classifier to rank the unlabeled examples by confidence.
  - EM uses the classifier to assign probabilities to each unlabeled example.

- Assumptions made by Semi-Super L method:
  - Co-training: conditional independence assumption.
  - EM: maximizing likelihood correlates with reducing classification errors.

Co-EM: EM with feature split

Idea:

- Like co-training, use one set of features to label the other
- Like EM, iterate
  - Assign probabilistic values to unobserved class labels
  - Updating model parameters (= labels of other feature set)

Goal:

to learn, $X_1 \rightarrow Y$, $X_2 \rightarrow Y$, $X_1 \times X_2 \rightarrow Y$

$$P(Y \mid X_1 = k) = \sum_j P(Y \mid X_2 = j) P(X_2 = j \mid X_1 = k)$$

$$P(Y \mid X_2 = j) = \sum_k P(Y \mid X_1 = k) P(X_1 = k \mid X_2 = j)$$

Co-EM: EM with feature split

- Repeat until converge
  - Train **A-feature-set** classifier using the labeled data and the unlabeded data with **B's** labels
  - Use classifier A to probabilistically label all the unlabeled data
  - Train **B-feature-set** classifier using the labeled data and the unlabeled data with **A's** labels.
  - B re-labels the data for use by A.

Some Applications

- Learning to extract named entities

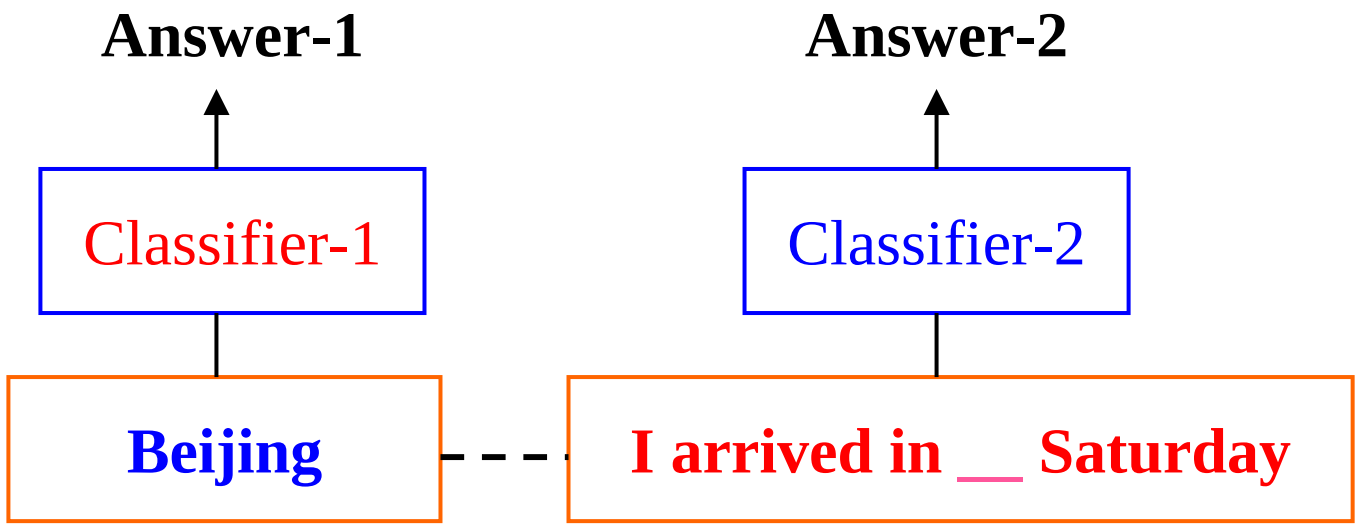
I arrived in Beijing Saturday.

If: "I **arrived in <X>** Saturday."

Then: **Location(X)**

Co-Training for Named Entity Extraction

[Riloff&Jones 98; Collins et al., 98; Jones 05]



I arrived in Beijing Saturday.

Named Entity Extraction

- Bootstrap learning to extract named entities
  - [Riloff and Jones, 1999], [Collins and Singer, 1999],
- CoEM applied to Named Entity Recognition
  - [Rosie Jones, 2005], [Ghani & Nigam, 2000]