Semi-Supervised Learning

Yanyan Lan

lanyanyan@ict.ac.cn

Supervised Learning

- Supervised Learning models require labeled data
- Learning a reliable model usually requires plenty of labeled data

Human expert/
Special equipment/
Experiment

"Crystal" "Needle" "Empty"

"0" "1" "2" ...

"Sports"
"News"
"Science"
...

Unlabeled data, X_i

Cheap and abundant !

Labeled data, Y_i

Expensive and scarce !

Can we devise ways to utilize unlabeled data with labeled data to learn better models?

Semi-Supervised Learning

- General idea: Learning from both labeled and unlabeled data
- Semi-supervised classification/regression
  - **Given:** labeled training data $D = \{(x_1, y_1), (x_2, y_2) \dots (x_l, y_l)\}$, Unlabeled data $D_u = \{x_{l+1}, x_{l+2}, \dots x_{l+u}\} (usually\ u \gg l)$
  - Goal: Learning a classifier f better than using labeled data alone
- Semi-unsupervised clustering/dimensionality reduction
  - **Given:** unlabeled data $\{x_i\}^m$, and the goal could be to do clustering or dimensionality reduction. Additionally given: some constraints on the data
    - E.g., for clustering: two points must be in the same cluster, or two points must not be in the same cluster; for dimensionality reduction: two points must be close after the projection

Inductive learning vs Transductive learning

- Inductive learning
  - Given labeled training data $D = \{(x_1, y_1), (x_2, y_2) \dots (x_l, y_l)\}$, unlabeled data $D_u = \{x_{l+1}, x_{l+2}, \dots x_{l+u}\} (usually\ u \gg l)$, learn a function f
  - f is used to predict labels for the future test data
  - Learning a function to be applied on the test data
- Transductive learning
  - Given labeled training data $D = \{(x_1, y_1), (x_2, y_2) \dots (x_l, y_l)\}$, unlabeled data $D_u = \{x_{l+1}, x_{l+2}, \dots x_{l+u}\}$
  - No explicit function is learned, all we care about is the predictions for D_u
  - The set D_u is the test data and is available at the training time.

Why the name

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

↕

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

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

↕

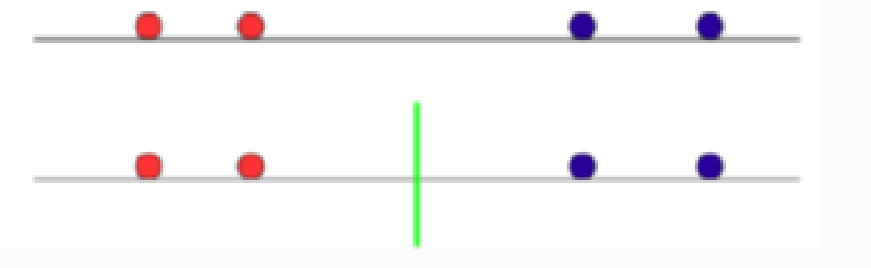
Semi-supervised clustering $\{x_{1:n}, must-, cannot-links\}$

↕

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

How can unlabeled data ever help?

- Red: +1, Dark Blue:-1

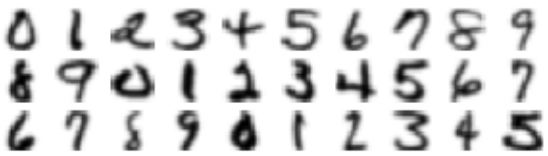


- Let`s include some additional unlabeled data (Light Blue points)

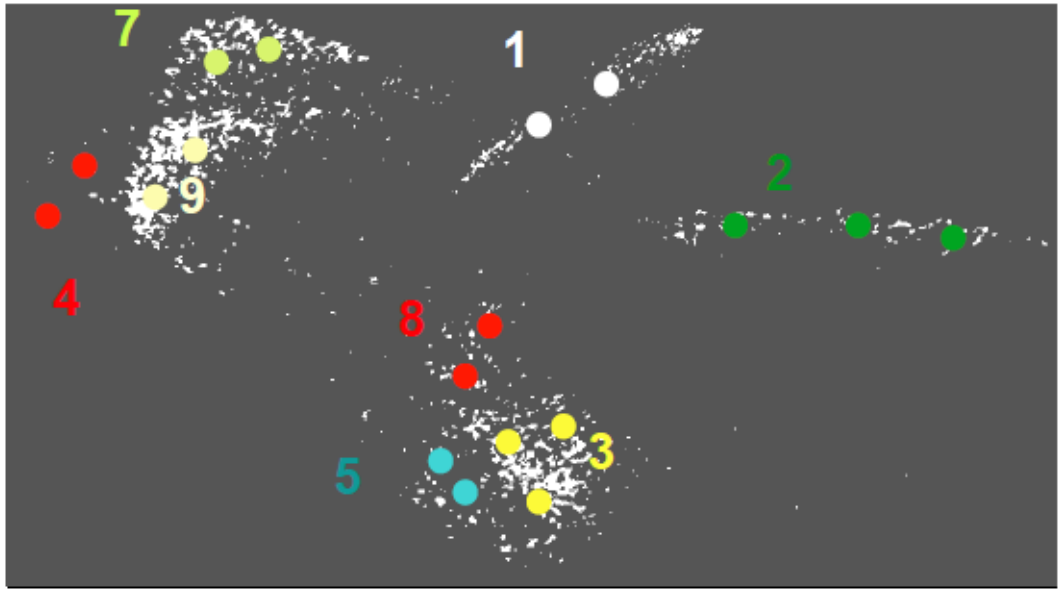
- Assumption: examples from the same class follow a coherent distribution
- Unlabeled data can give a better sense of the class separation boundary

How can unlabeled data ever help?

Unlabeled Images



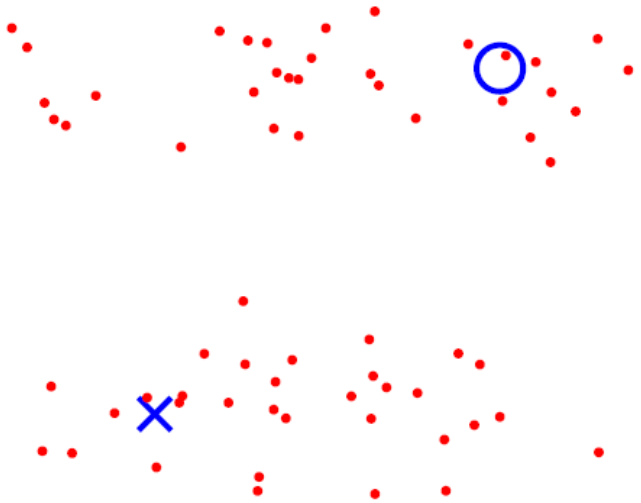
Labels “0” “1” “2” …



- Assumption: “Similar” data points have “similar” labels

Cluster assumption

- Cluster assumption
  - If points are in the same cluster, they are likely to be of the same class.
- Equivalent formulation of cluster assumption:
  - Low density separation:
  - The decision boundary should lie in a low-density region.



Manifold Assumption

- Manifold assumption
  - the high-dimensional data lie roughly on a low-dimensional manifold.
- A problem of many statistical methods and learning algorithms is the curse of dimensionality, due to the fact that an exponentially growing number of examples is required for statistical tasks such as density estimation.
- If the data happen to lie on a low-dimensional manifold, then the learning algorithm can essentially operate in a space of corresponding dimensionality, thus avoid the cure of dimensionality.

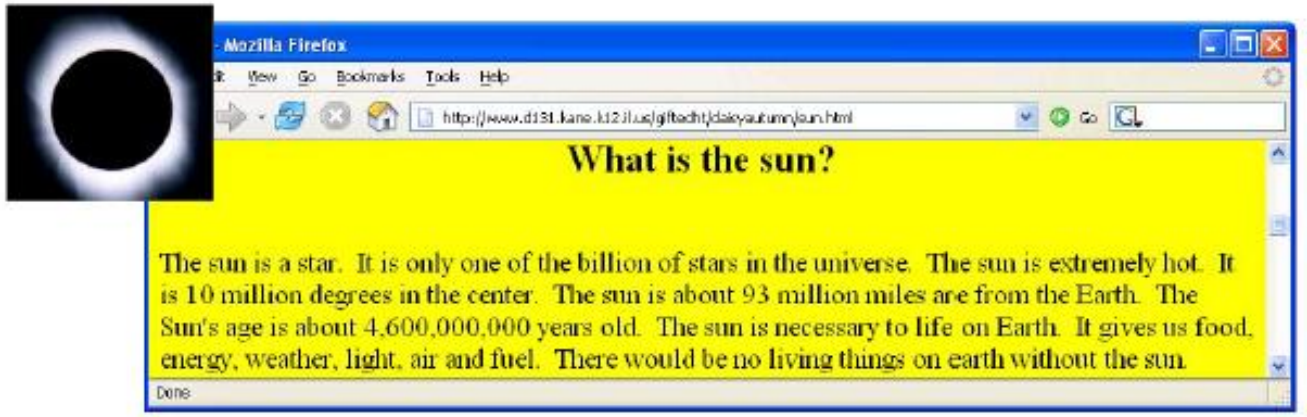
Major Semi-Supervised Learning Models

- Multiview algorithms
  - eg. Co-training [BM98]
- Generative models
  - data are generated from the same generative model.
  - eg. Gauss of Mixture Model
- Low-density separation models
  - eg. Transductive SVM [Joa99]
- Graph-based algorithms
  - The data (both labeled and unlabeled) are represented by the nodes of a graph, the edges of which are labeled with the pairwise distances of the nodes.
  - These methods are based on the manifold assumption.
  - eg. Label Propagation
- Semi-supervised Clustering

Multiview Algorithms

Co-training

- Two views of an item: image and HTML text

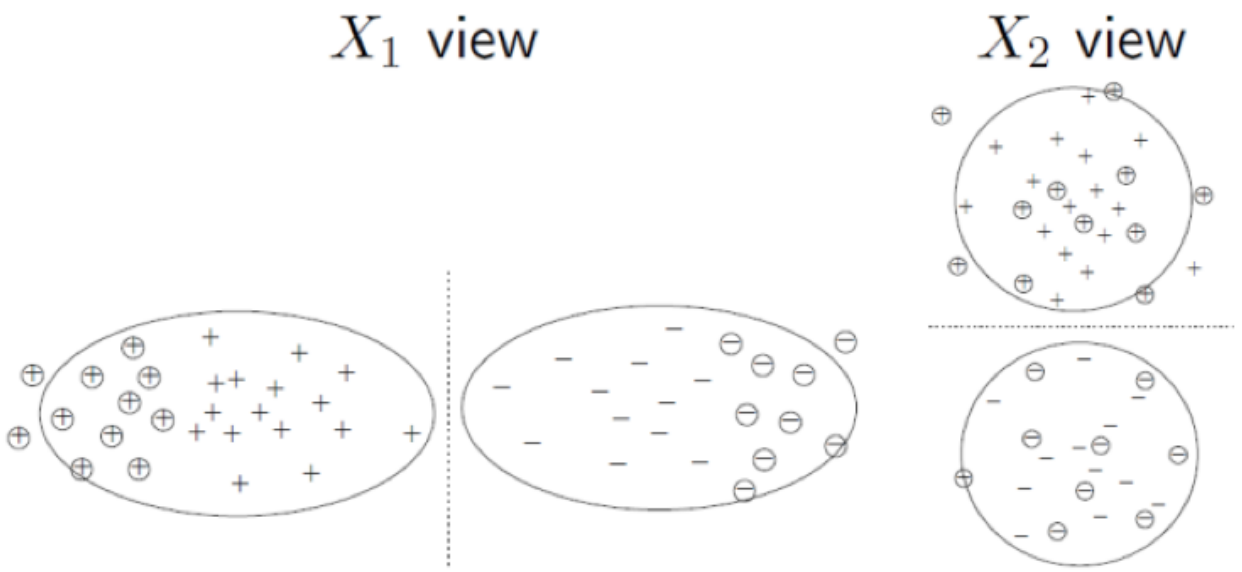


Feature split

- 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 $x = \left[x^{(1)}; x^{(2)}\right]$ 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
  - Train two classifiers: $f^{(1)}$ from $\left(X_l^{(1)}, Y_l\right)$, $f^{(2)}$ from $(X_l^{(2)}, Y_l)$.
  - Classify X_u with $f^{(1)}$ and $f^{(2)}$ separately.
  - Add $f^{(1)}$'s k-most-confident $(x, f^{(1)}(x))$ to $f^{(2)}$'s labeled data
  - Add $f^{(2)}$'s k-most-confident $(x, f^{(2)}(x))$ to $f^{(1)}$'s labeled data
  - Repeat

Multi-view Learning

- A general class of algorithms for semi-supervised learning
- Based on using multiple views (feature representations) of the data
  - Co-training is a special type of multi-view learning algorithm
- General Idea:
  - Train multiple classifiers, each using a different view
  - Multiple classifiers must agree on the unlabeled data

A regularized risk minimization framework to encourage multi-learner agreement:

$$\min_f \sum_{v=1}^{M} \left(\sum_{i=1}^{l} c\left(y_i, f_v(x_i)\right) + \lambda_1 \|f\|_K^2 \right) + \lambda_2 \sum_{u,v=1}^{M} \sum_{i=l+1}^{n} \left(f_u(x_i) - f_v(x_i)\right)^2$$

M learners. C() is the loss function, e.g., hinge loss, square loss

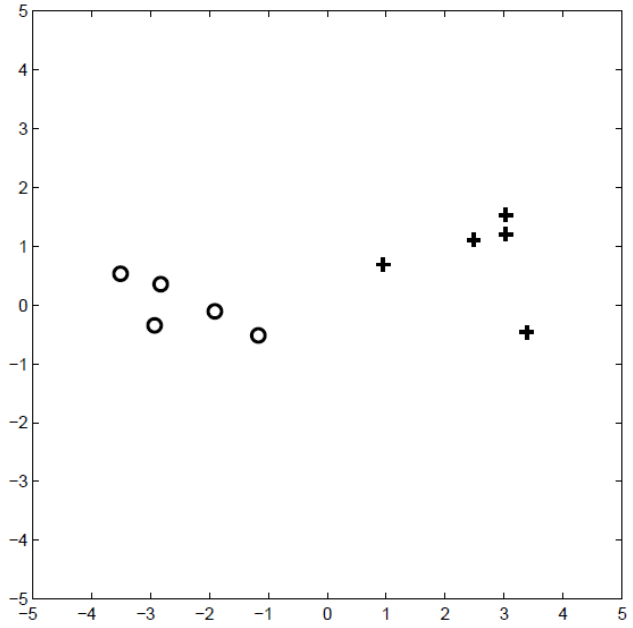
Multi-view Learning

- How might it help learn better?
  - Learning is essentially searching for the best classifier
  - By enforcing agreement among classifiers, we are reducing the search space
  - hope is that the best classifier can be found easily with little labeled data
- For test data, these multiple classifiers can be combined
  - E.g., voting, consensus, etc.
- Backed by a number of theoretical results
- Multiview-based semi-supervised learning is a natural bridge between semi-supervised learning and ensemble learning

Generative Models

An example of generative models

- Labeled data (X_l, Y_l):



Assuming each class has a Gaussian distribution, what is the decision boundary?

An example of generative models

Model parameters $\theta = \{\omega_1, \omega_2, \mu_1, \mu_2, \sum_1, \sum_2\}$

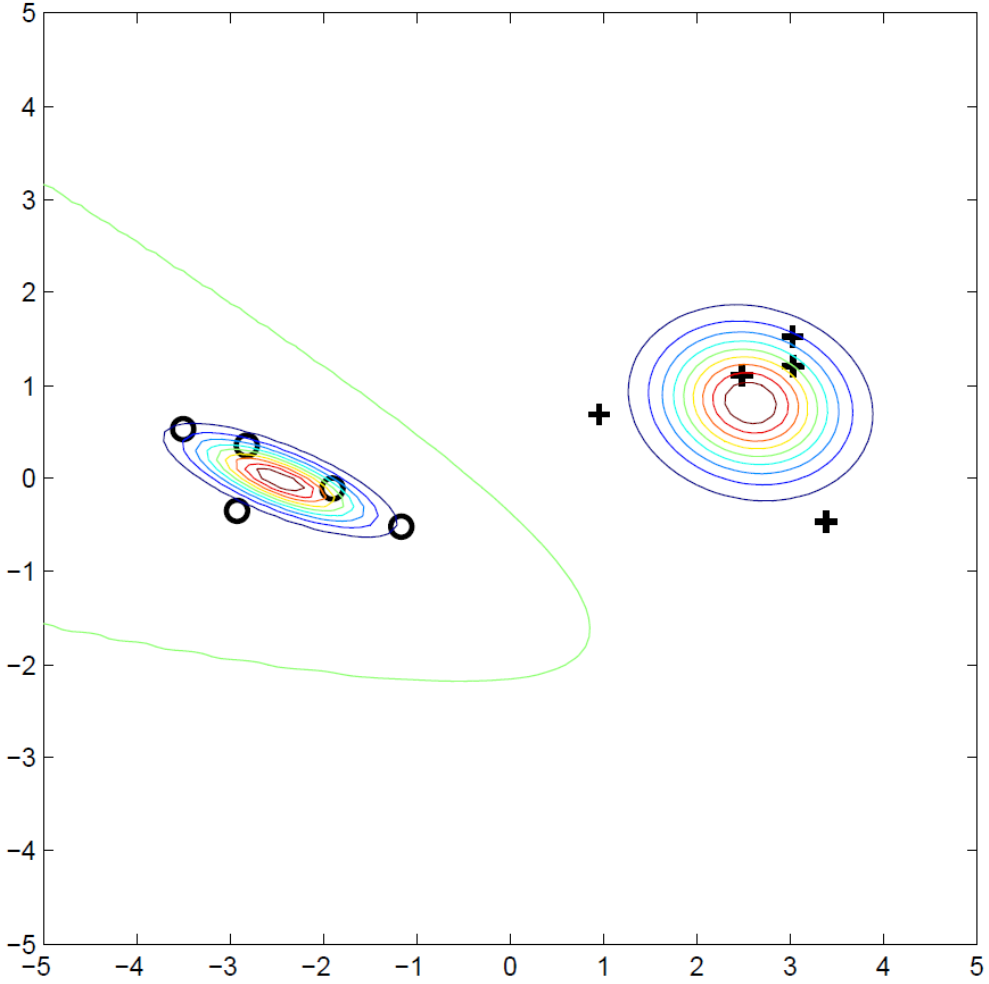
The GMM:

$$p(x, y|\theta) = p(y|\theta)p(x|y, \theta) = \omega_y \mathcal{N}(x; \mu_y, \sum_y)$$

Classification: $P(y|x, \theta) = \frac{p(x,y|\theta)}{\sum_{y'} p(x,y'|\theta)}$

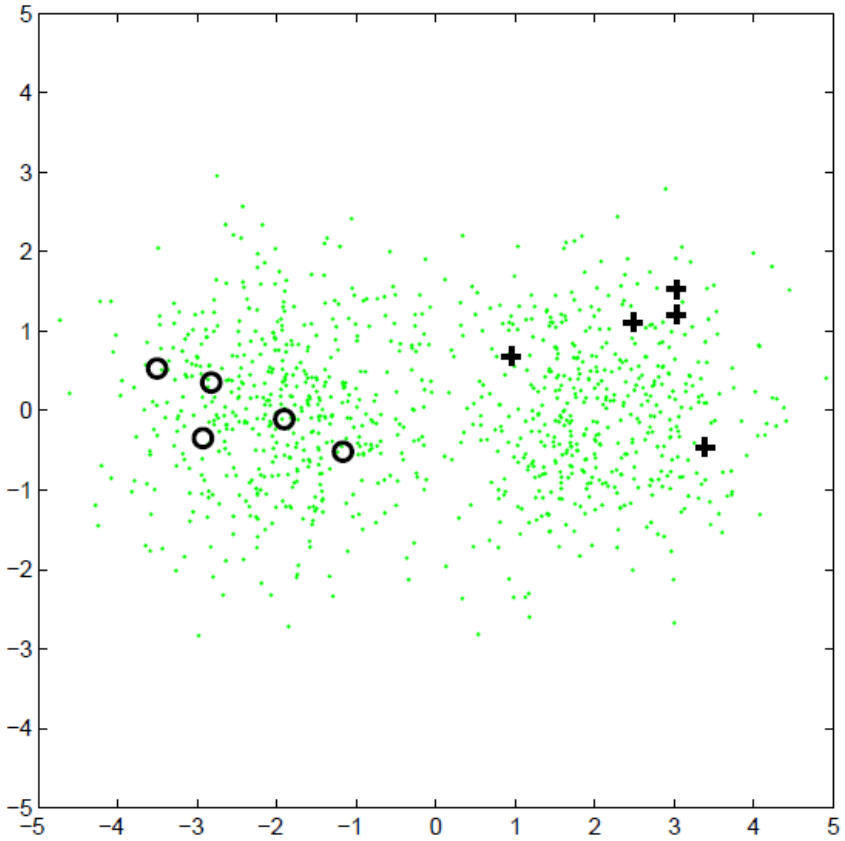
An example of generative models

The most likely model, and its decision boundary:



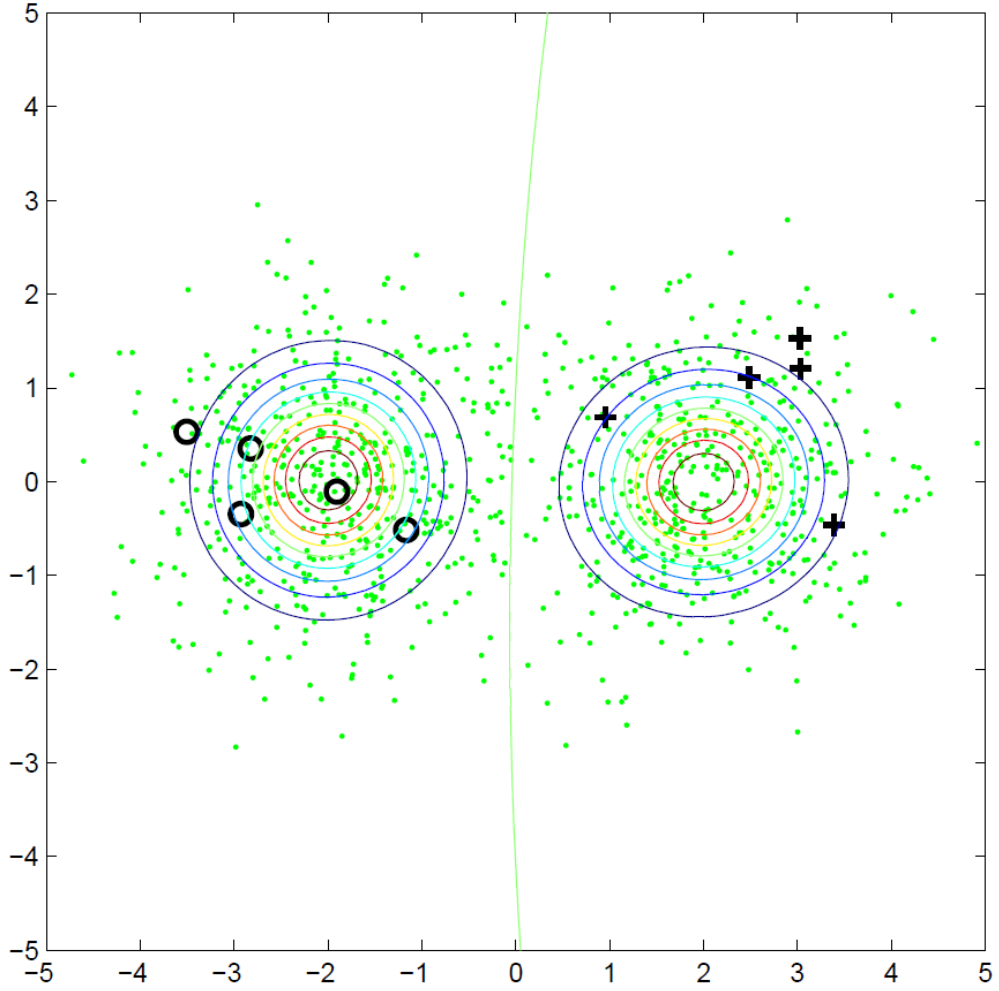
An example of generative models

Adding unlabeled data



An example of generative models

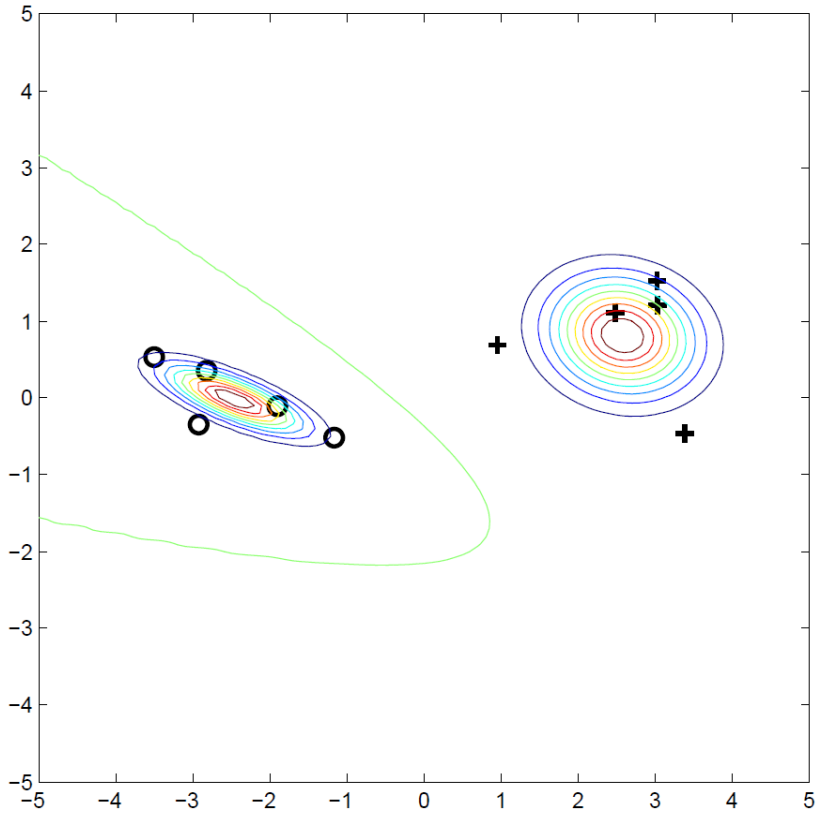
With unlabeled data, the most likely model and its decision boundary:



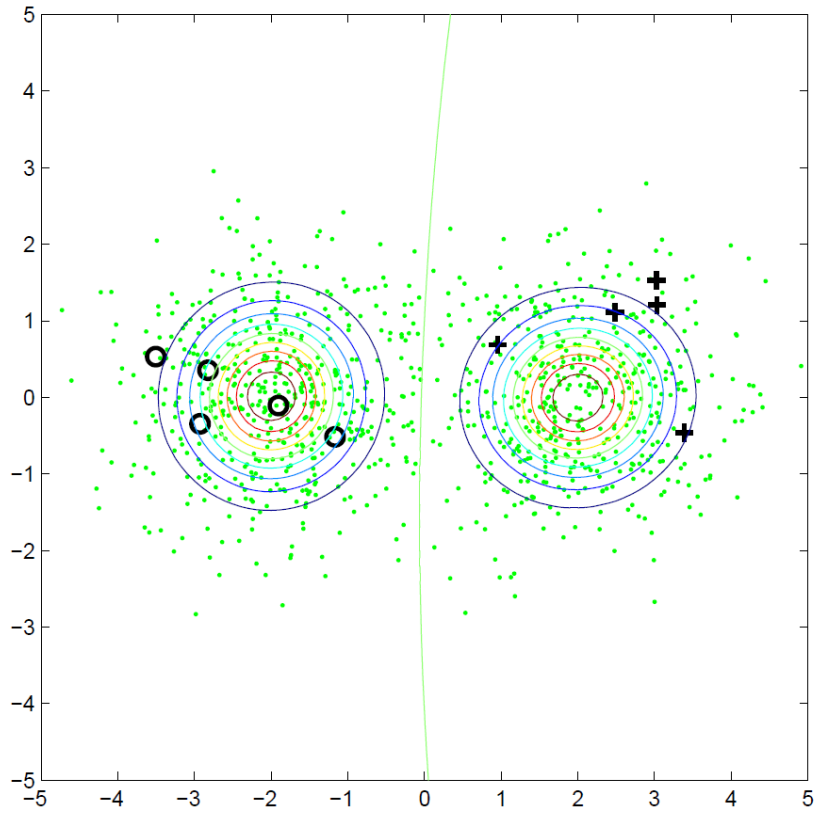
An example of generative models

They are different because they maximize different quantities.

$p(X_l, Y_l|\theta)$



$p(X_l, Y_l, X_u|\theta)$



Generative model for semi-supervised learning

- Assumption
  - The full generative model $P(X, Y|\theta)$
- Generative model for semi-supervised learning:
  - Quantity of interesting: $p(X_l, Y_l, X_u|\theta) = \sum_{Y_u} p(X_l, Y_l, X_u, Y_u|\theta)$
  - Find the maximum likelihood estimate (MLE) of θ , the maximum a posteriori (MAP) estimate, or be Bayesian

Examples of some generative models

Often used in semi-supervised learning:

- Mixture of Gaussian distribution (GMM)
  - image classification
  - the EM algorithm
- Mixture of multinomial distributions (Naïve Bayes)
  - text categorization
  - the EM algorithm
- Hidden Markov Models(HMM)
  - speech recognition
  - Baum-Velch algorithm

Case study: GMM

- For simplicity, consider binary classification with GMM using MLE
- Labeled data only
  - $\log p(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta)$
  - MLE for θ trivial (frequency, sample mean, sample covariance)
- Labeled and unlabeled data

$$\log p(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta)$$
$$+\sum_{i=l+1}^{l+u} \log \sum_{y=1}^{2} p(y|\theta)p(x_i|y, \theta)$$

  - MLE harder (hidden variables)
  - The expectation-maximization(EM) algorithm is one method to find local optimum

The EM algorithm for GMM

- Start from $MLE\ \theta = \{\omega, \mu, \sum\}_{1:2}$ on (X_l, Y_l)
  - w_c=propotation of class c
  - μ_c=sample mean of class c
  - $\sum_c$=sample cov of class c

repeat:

- The E-step: compute the expected label $p(y|x,\theta) = \frac{p(x|y,\theta)}{\sum_{y'} p(x,y'|\theta)}\ for\ all\ x \epsilon X_u$
  - Label $p(y = 1|x, \theta)$-fraction of x with class 1
  - Label $p(y = 2|x, \theta)$-fraction of x with class 2

- The M-step: update MLE θ with (now labeled) X_u

Can be viewed as a special form of self-training

Generative model for semi-supervised learning: beyond EM

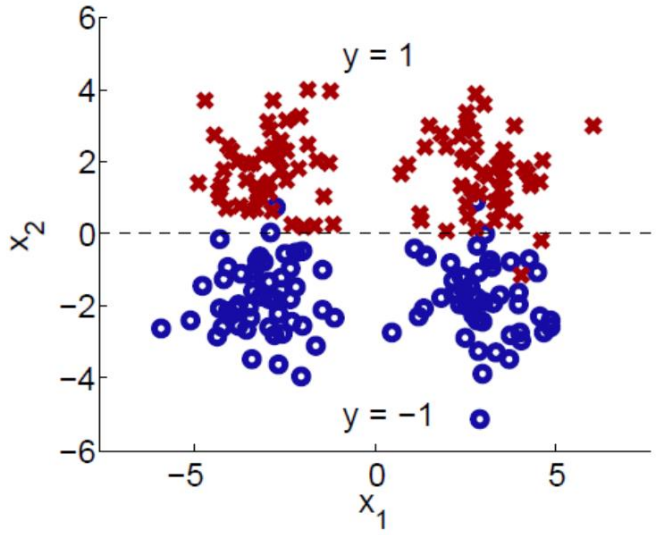
- Key is to maximize $p(X_l, Y_l, X_u|\theta)$
- EM is just one way to maximize it
- Other ways to find parameters are possible too, e.g., variational approximation, or direct optimization

Advantages of generative models

- Clear, well-studied probabilistic framework
- Can be extremely effective, if the model is close to correct

Disadvantages of generative models

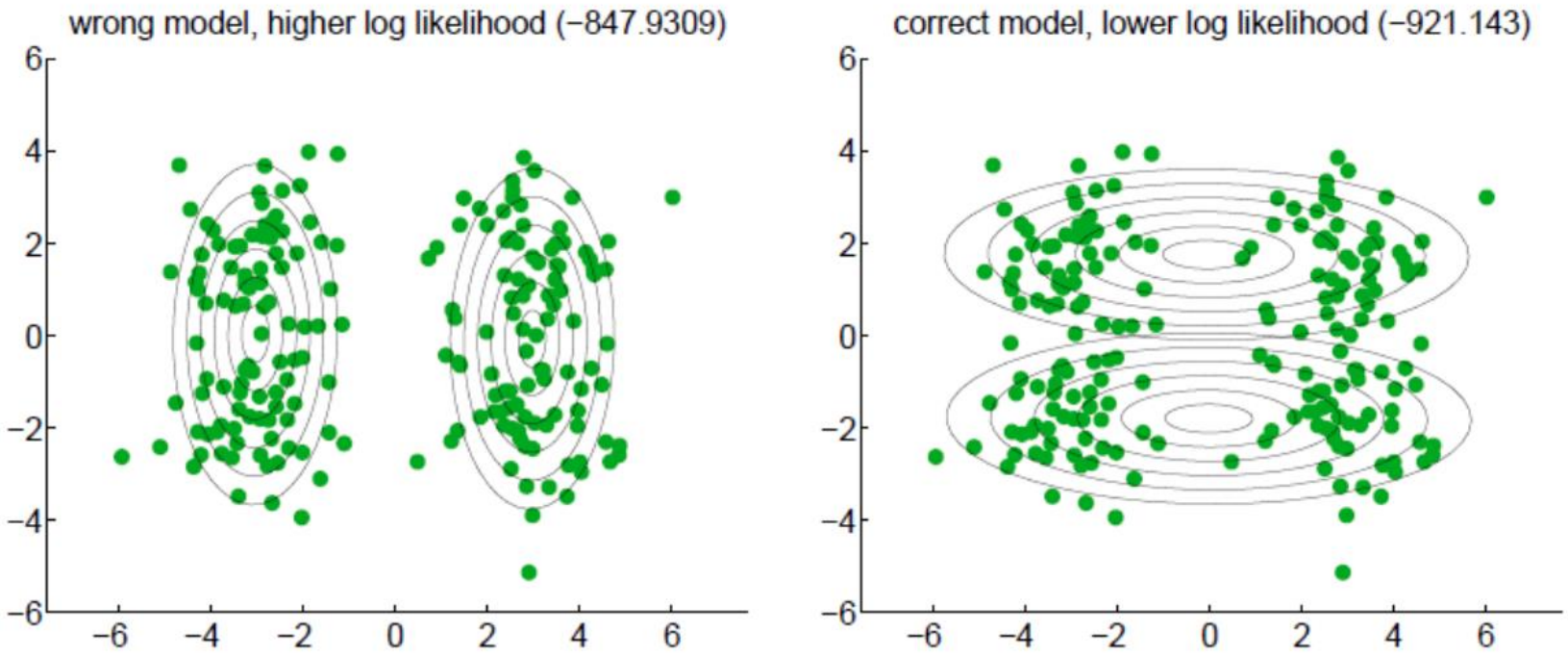
- Often difficult to verify the correctness of the model
- Model identifiability
- EM local optima
- Unlabeled data may hurt if generative model is wrong



For example, classifying text by topic vs by genre.

Unlabeled data may hurt semi-supervised learning

- If the generative model is wrong:



Heuristics to lessen the danger

- Carefully construct the generative model to reflect the task

  e.g., multiple Gaussian distributions per class, instead of a single one

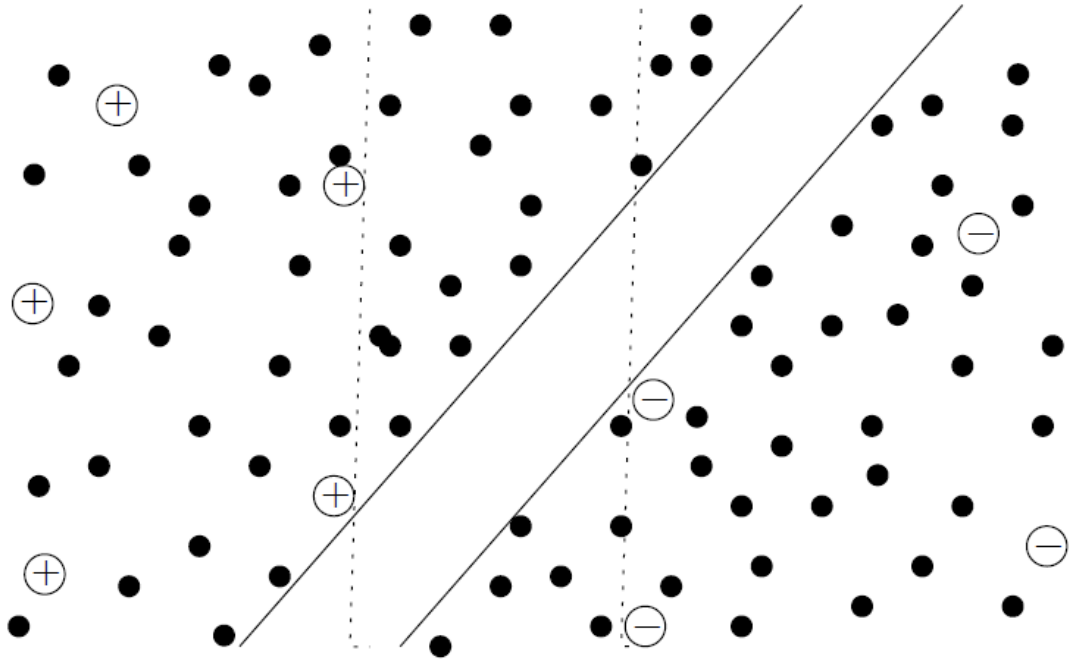
- Down-weight the unlabeled data

$$\log p\,(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i,\theta) + \lambda \sum_{i=l+1}^{l+u} \log \sum_{y=1}^{2} p(y|\theta)p(x_i|y_i,\theta)$$

S3VMS

Semi-supervised support vector machines

- Semi-supervised SVMs (S3VMs) = Transductive SVMs (TSVMs)
- Maximizes “unlabeled data margin”



S3VMs

- Assumption
  - Unlabeled data from different classes are separated with large margin
- S3VMs basic idea:
  - Enumerate all 2^u possible labeling of X_u
  - Build one standard SVM for each labeling (and X_t)
  - Pick the SVM with the largest margin

Standard SVM review

- Problem set up:
  - two classes $y \in \{+1, -1\}$
  - Labeled data $\{X_l, Y_l\}$
  - Weight w
- SVM finds a function $f(x) = w^{\mathsf{T}}x + b$
- Classify x by $sign(f(x))$

Standard soft margin SVMs

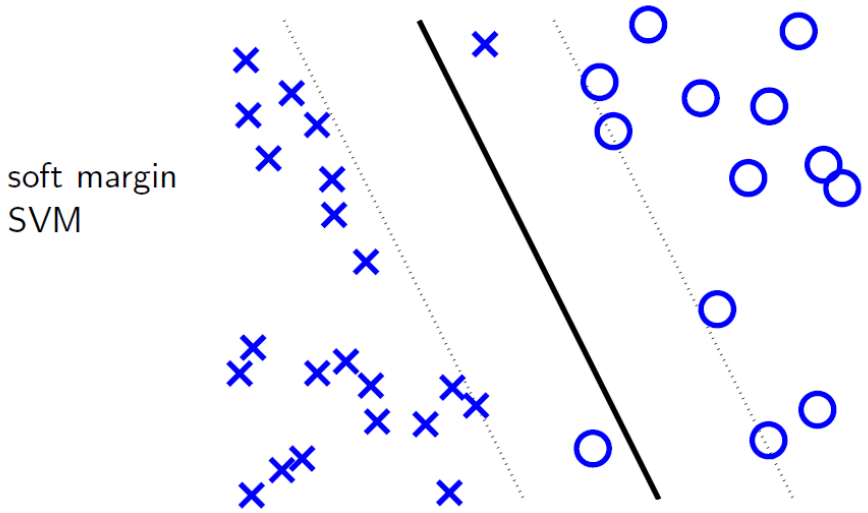
- Try to keep labeled points outside the margin, while maximizing the margin:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$s.t. \quad y_i(w^\intercal x_i + b) \geq 1 - \xi_i, \forall i = 1 \ldots l$$

$$\xi_i \geq 0$$

- The ξ_i's are slack variables



SVM with hinge function

Let $z_i = 1 - y_i\left(w^\intercal x_i + b\right) = 1 - y_i f(x_i)$, the problem

$$\min_{h,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$\text{subject to } y_i(w^\intercal x_i + b) \geq 1 - \xi_i \text{ , } \forall i = 1 \cdots l$$
$$\xi_i \geq 0$$

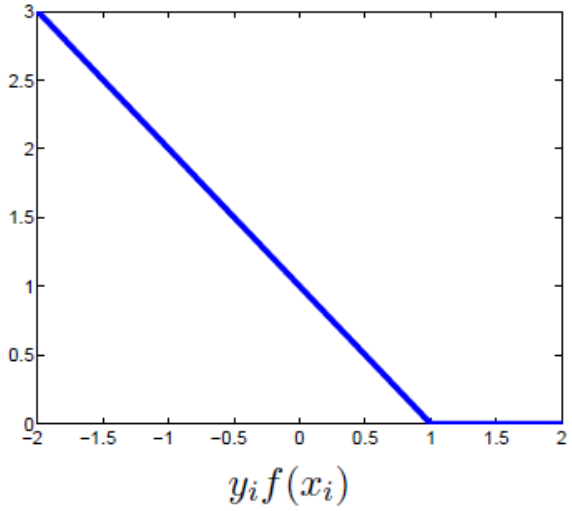
is equivalent to

$$\min_{f} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(1 - y_i f(x_i))_+$$

The hinge loss in standard SVMs

$$\min_f \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(1 - y_i f(x_i))_+$$

$y_i f(x_i)$ known as the margin, $(1 - y_i f(x_i))_+$ the hinge loss



Prefers labeled points on the ‘correct’ side.

S3VM objective function

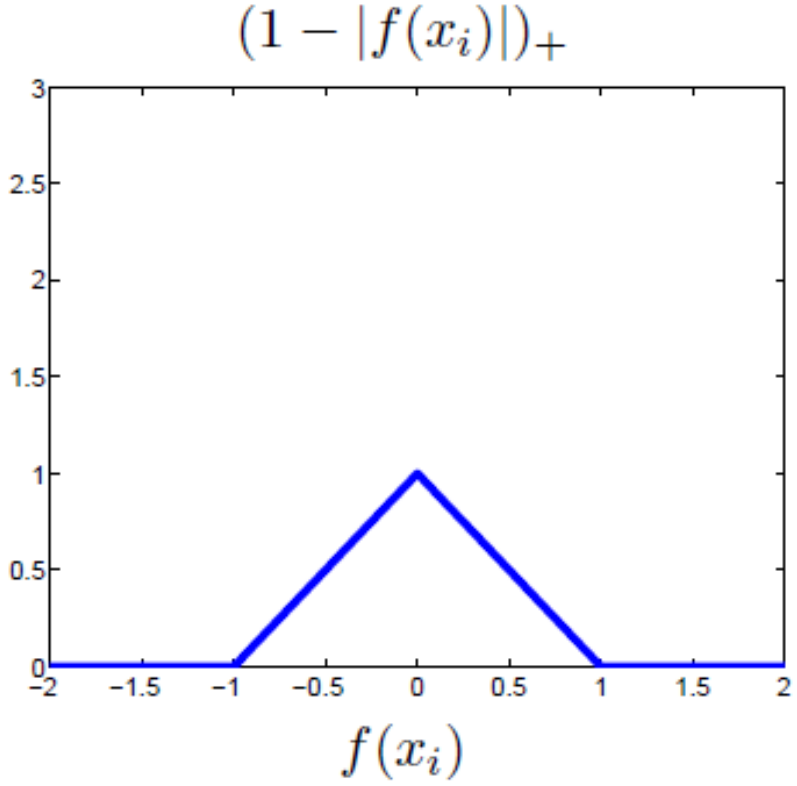
- How to incorporate unlabeled points?
  - Assign putative labels sign$(f(x))$ to $x \in X_u$
  - sign$(f(x))\ f(x) = |f(x)|$
  - The hinge loss on unlabeled points becomes

$$(1 - y_i f(x_i))_+ = (1 - |f(x_i)|)_+$$

- S3VM objective:

$$\min_f \frac{1}{2}\|w\|^2 + C_1 \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + C_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

The hat loss on unlabeled data



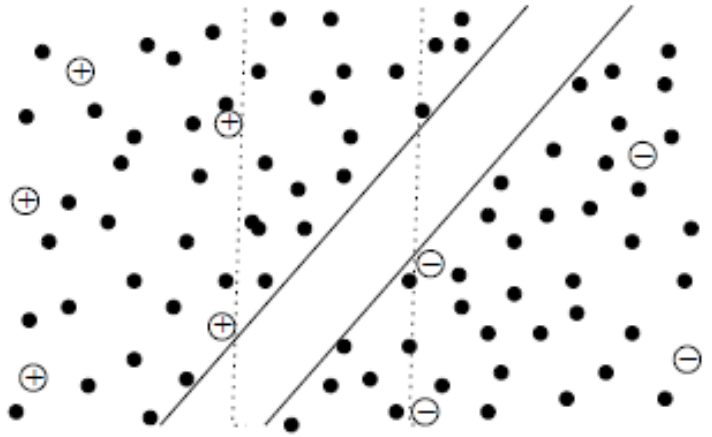
Prefers $f(x) \geq 1$ or $f(x) \leq -1$, i.e., unlabeled instance away from decision boundary $f(x) = 0$.

Avoiding unlabeled data in the margin

S3VM objective

$$\min_f \frac{1}{2}\|w\|^2 + C_1 \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + C_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

The third term prefers unlabeled points outside the margin. Equivalently, the decision boundary $f = 0$ wants to be placed so that there is few unlabeled data near it.



The class balancing constraint

- Directly optimizing the S3VM objective often produces unbalanced classification — most points fall in on class
- Heuristic class balance: $\frac{1}{n-l}\sum_{i=l+1}^{n} y_i = \frac{1}{l}\sum_{i=1}^{l} y_i$.
- Relaxed class balancing constraint: $\frac{1}{n-l}\sum_{i=l+1}^{n} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i$

The S3VM algorithm

- Input: weights w, C_1, C_2, (X_l, Y_l), X_u
- Solve the optimization problem for $f(x) = w^{\mathsf{T}}x + b$

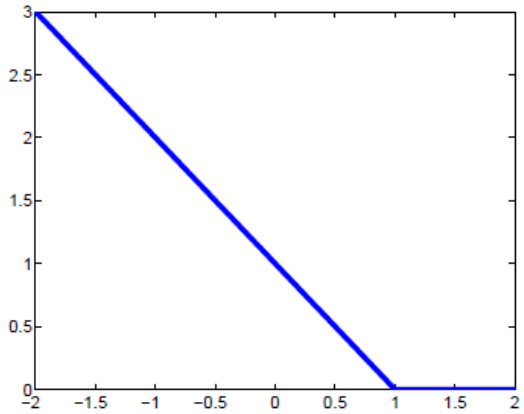
$$\min_f \frac{1}{2}\|w\|^2 + C_1 \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + C_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

$$\text{s.t.} \quad \frac{1}{n-l}\sum_{i=l+1}^{n} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i$$

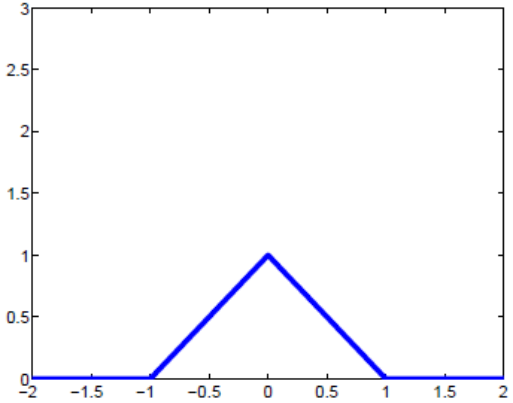
- Classify a new test point x by $\text{sign}(f(x))$

The S3VM optimization challenge

- SVM objective is convex:



- Semi-supervised SVM objective is non-convex:



- Finding a solution for semi-supervised SVM is difficult, which has been the focus for S3VM research.

Optimization methods used for S3VM training

- Exact:
  - Mixed Integer Programming [Bennett, Demiriz; NIPS 1998]
  - Branch & Bound [Chapelle, Sindhwani, Keerthi; NIPS 2006]
- Approximative:
  - self-labeling heuristic S^3VM^{light} [T. Joachims; ICML 1999]
  - gradient descent [Chapelle, Zien; AISTATS 2005]
  - CCCP-S^3VM [R. Collobert et al.; ICML 2006]
  - contS^3VM [Chapelle et al.; ICML 2006]

S3VM implementation 1: $\boldsymbol{SVM^{light}}$

- Local combinatorial search
- Assign hard labels to unlabeled data
- Outer loop: “Anneal” C_2 from zero up
- Inner loop: Pairwise label switch

S3VM implementation 1: SVM^{light}

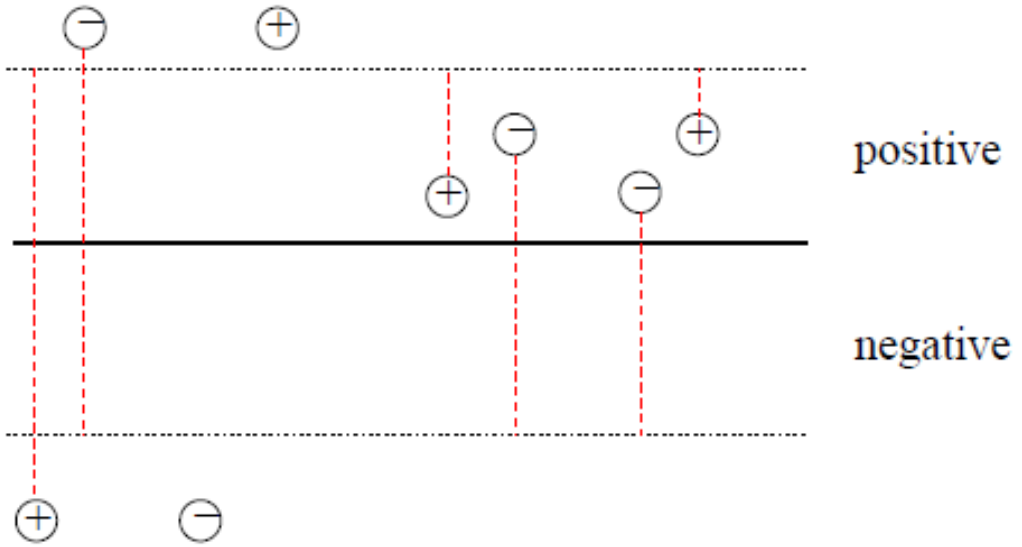
- Train an SVM with (X_l, Y_l).
- Sort X_u by $f(X_u)$. Label $y = 1, -1$ for the appropriate portions.
- FOR $C_2 \leftarrow 10^{-5} C_1 \ldots C_1$
  - REPEAT:
  - $\min_f \frac{1}{2}\|w\|^2 + C_1 \sum_{i=1}^{l}(1 - y_i f(x_i)) + C_2 \sum_{i=l+1}^{n}(1 - |f(x_i)|)_+$
  - IF $\exists (i, j)$ switchable THEN switch y_i, y_j
  - UNTIL No labels switchable

S3VM implementation 1: SVM^{light}

$i, j \in X_u$ switchable if $y_i = 1, y_j = -1$ and

$$loss(y_i = 1, f(x_i)) + loss\left(y_j = -1, f\left(x_j\right)\right)$$
$$> loss(y_i = -1, f(x_i)) + loss(y_j = 1, f(x_j))$$

With the hinge loss $loss(y, f) = (1 - yf)_+$



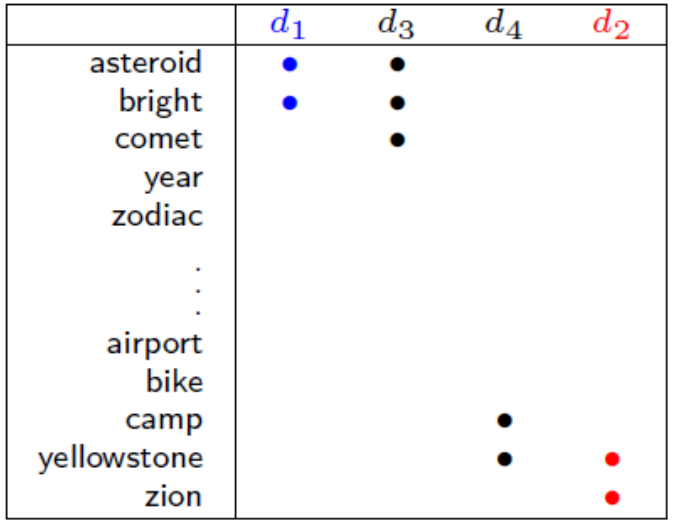
Summary of S3VMs

- Pros:
  - Applicable wherever SVMs are applicable
  - Clear mathematical framework
- Cons:
  - Optimization difficult
  - Can be trapped in bad local optima
  - More modest assumption than generative model or graph-based methods, potentially lesser gain

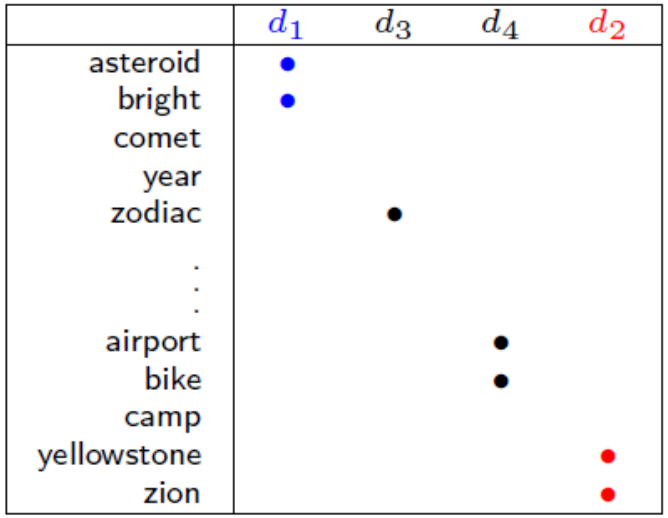
Graph-based Algorithms

Example: text classification

- Classify astronomy vs. travel articles
- Similarity measured by content word overlap



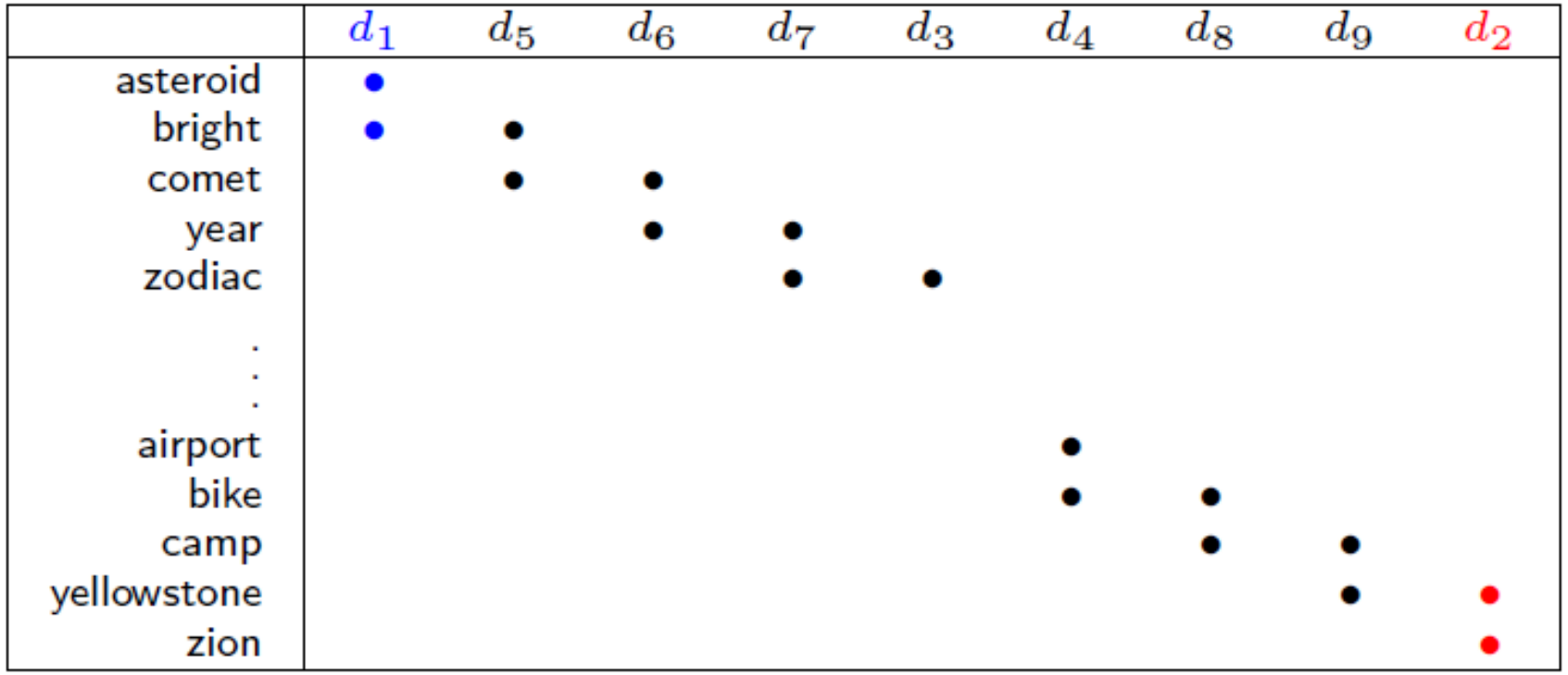
No overlapping words!



When labeled data alone fails

Unlabeled data as stepping stones

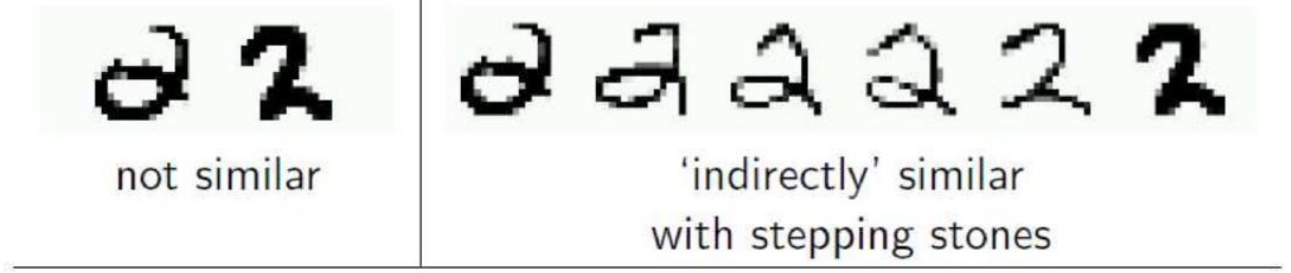
Labels “propagate” via similar unlabeled articles.



Graph-based semi-supervised learning

- Assumption
  - A graph is given on the labeled and unlabeled data. Instances connected by heavy edge tend to have the same label.
- The labels should vary smoothly along the graph
  - Nearby vertices should have similar labels

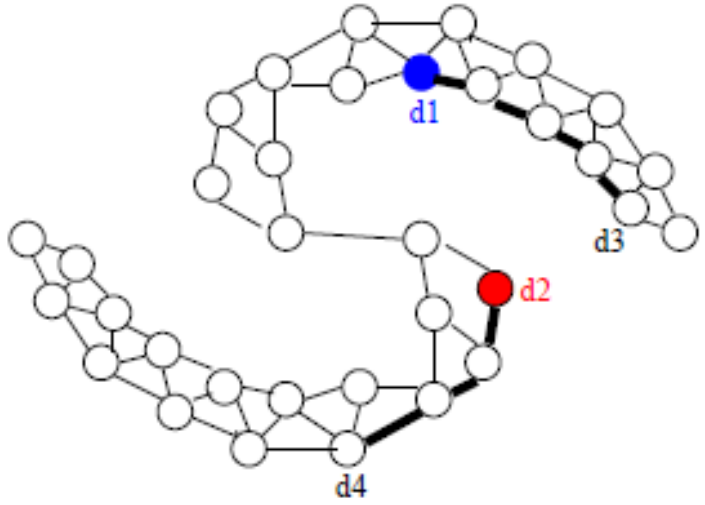
Handwritten digits recognition with pixel-wise Euclidean distance



We can call it label propagation

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 $w = \exp(-\|x_i - x_j\|^2 / \sigma^2)$
  - ε-radius graph
- Want: implied similarity via all paths



Some graph-based algorithms

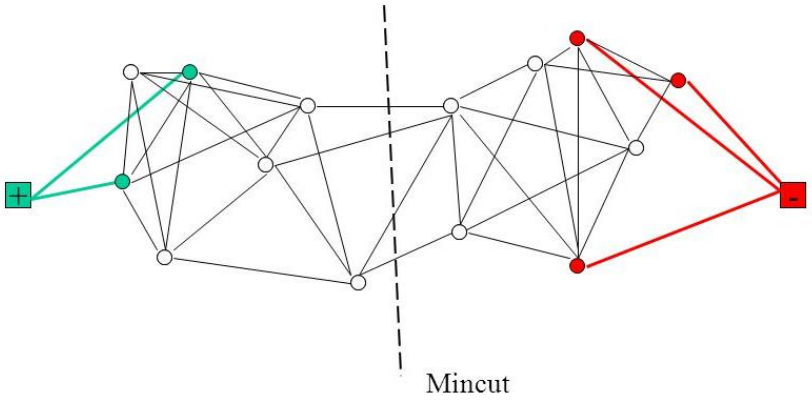
- Mincut
- Harmonic
- Local and global consistency
- Manifold regularization

The mincut algorithm

- The graph mincut problem:
  - Fix Y_l, find $Y_u \in \{0,1\}^{n-l}$ to minimize $\sum_{ij} w_{ij}|y_i - y_j|$
  - Equivalently, solves the optimization problem

$$\min_f \infty \sum_{i=1}^{l} (f(x_i) - Y_{li})^2 + \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2$$

  - Combinatorial problem, but has polynomial time solution

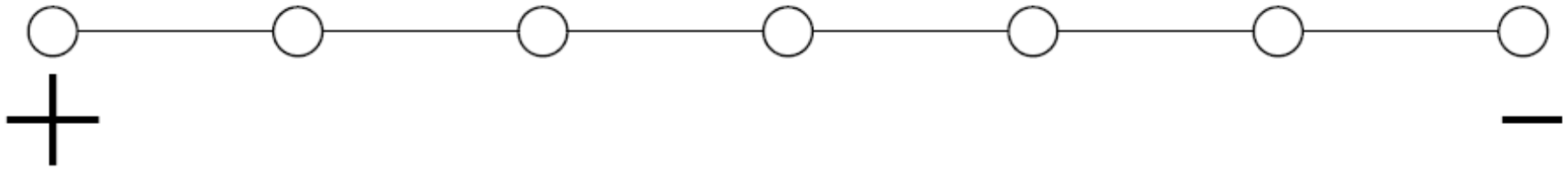


While there are more than 2 vertices:
- Pick a remaining edge (u, v) uniformly at random
- Merge (or "contract") u and v into a single vertex
- Remove self-loops

Return cut represented by final 2 vertices

The mincut algorithm

- Mincut computes the modes of a Boltzmann machine
- There might be multiple modes
- One solution is to randomly perturb the weights, and average the results.



The harmonic function （Ten Year Best Paper Award, ICML’2013）

- Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies
  - $f(x_i) = y_i$ for $i = 1 \cdots l$
  - f minimizes the energy

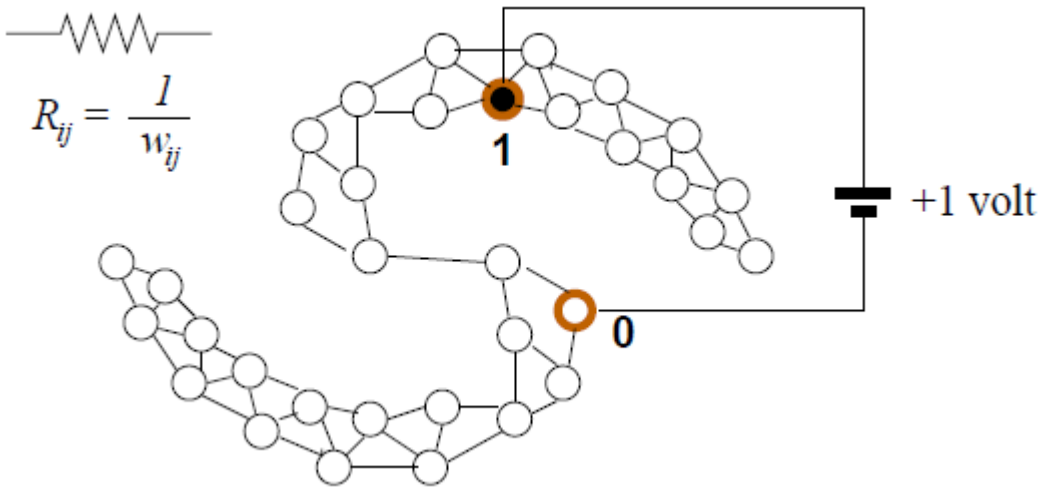
$$\sum_{i \sim j} w_{ij} (f(x_i) - f(x_j))^2$$

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

An electric network interpretation

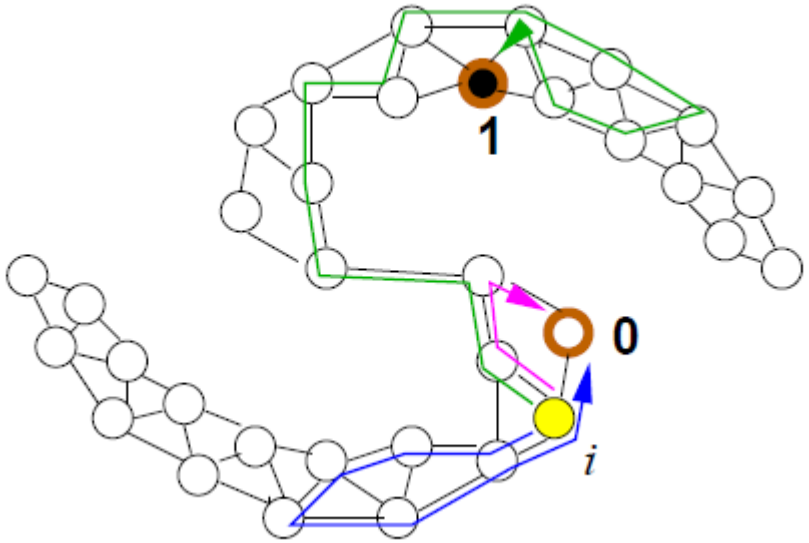
- Edges are resistors with conductance w_{ij}
- 1 volt battery connects to labeled points $y = 0,1$
- The voltage at the nodes is the harmonic function f

Implied similarity: similar voltage if many paths exist



A random walk interpretation

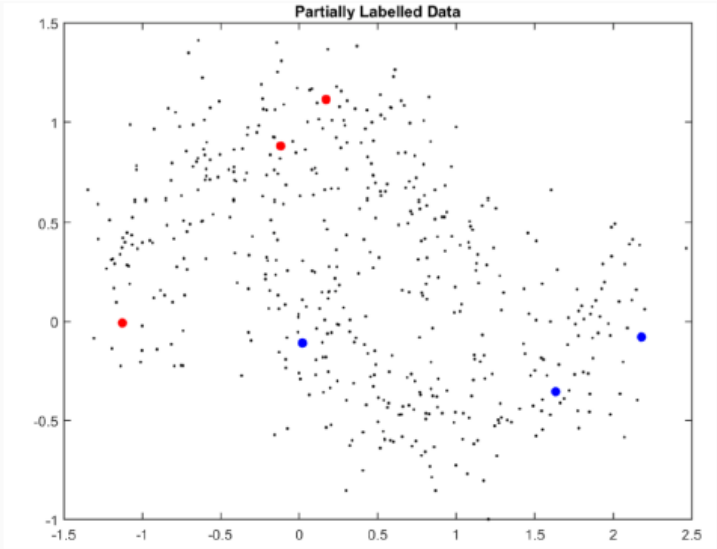
- Randomly walk from node i to j with probability $\frac{w_{ij}}{\sum_k w_{ik}}$
- Stop if we hit a labeled node
- The harmonic function $f = Pr(\text{hit label 1} | \text{start from } i)$

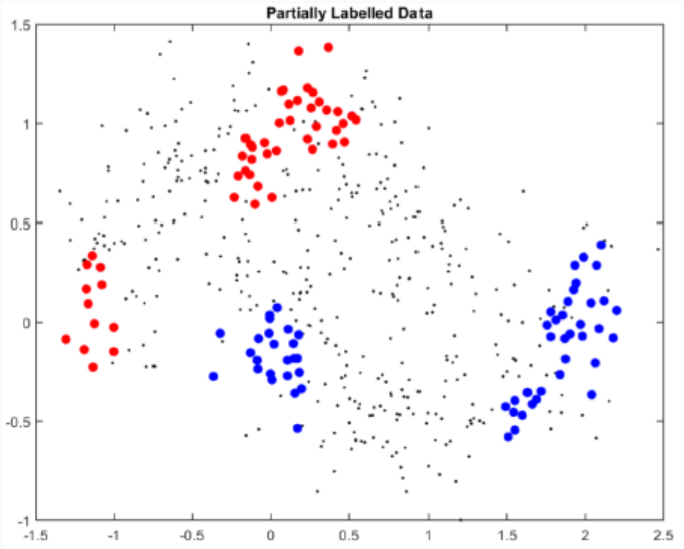


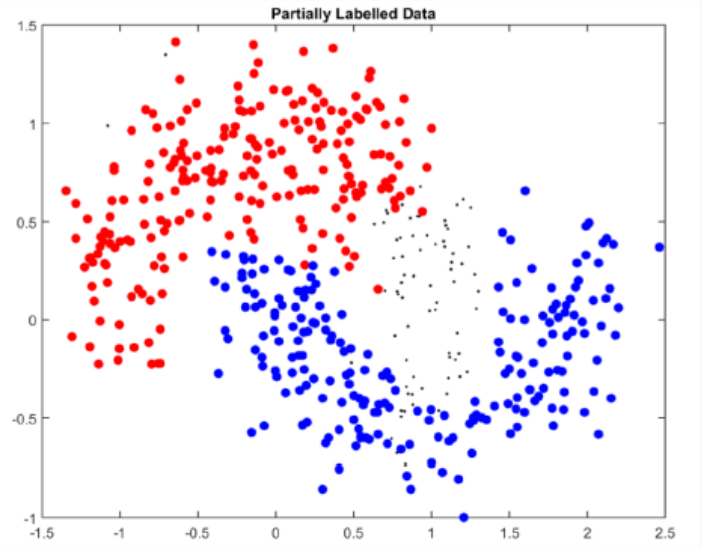
An algorithm to compute harmonic function

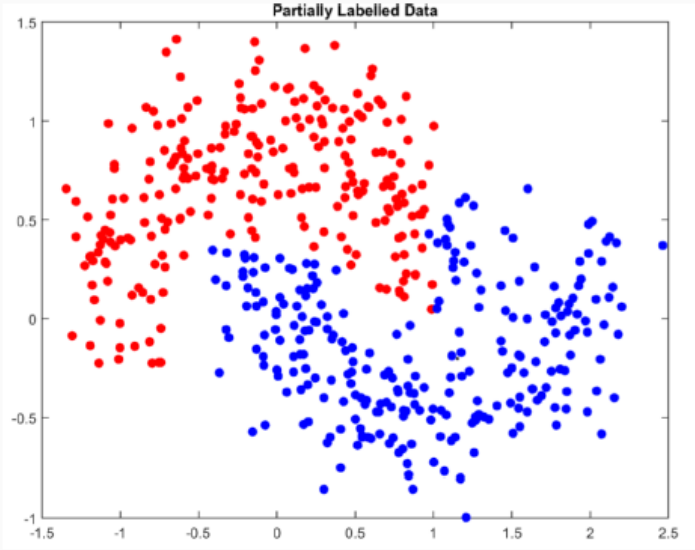
- One way to compute the harmonic function is:
  - Initially, set $f(x_i) = y_i$ for $i = 1 \cdots l$, and $f(x_j)$ arbitrarily (e.g., 0) for $x_j \in X_u$
  - Repeat until convergence: Set $f(x_i) = \frac{\sum_{j \sim i} w_{ij} f(x_j)}{\sum_{j \sim i} w_{ij}}$, $\forall x_i \in X_u$, i.e., the average of neighbors. Note $f(X_l)$ is fixed.
- This can be viewed as a special case of self-training too.

Label Propagation









The graph Laplacian

We can also compute f in closed form using the graph Laplacian.

- $n \times n$ weight matrix W on $X_l \cup X_u$
  - symmetric, non-negative
- Diagonal degree matrix D: $D_{ii} = \sum_{j=1}^{n} W_{ij}$
- Graph Laplacian matrix Δ

$$\Delta = D - W$$

- The energy can be rewritten as

$$\frac{1}{2}\sum_{i \sim j} w_{ij}(f(x_i) - f(x_j))^2 = f^{\intercal}\Delta f$$

Harmonic solution with Laplacian

The harmonic solution minimizes energy subject to the given labels

$$\min_f \infty \sum_{i=1}^{l} (f(x_i) - y_i)^2 + f^\top \Delta f$$

Partition the Laplacian matrix $\Delta = \begin{bmatrix} \Delta_{ll} & \Delta_{lu} \\ \Delta_{ul} & \Delta_{uu} \end{bmatrix}$

Harmonic solution $\Delta f = 0$

$$f_u = -{\Delta_{uu}}^{-1} \Delta_{ul} Y_l = (I - P_{uu})^{-1} P_{ul} Y_l, \text{ where } P = D^{-1} W$$

The normalized Laplacian $\mathcal{L} = D^{-1/2} \Delta D^{-1/2} = I - D^{-1/2} W D^{-1/2}$, or Δ^p, $\mathcal{L}^p$ are often used too ($p > 0$).

Problems with harmonic solution

Harmonic solution has two issues

- It fixes the given labels Y_l
  - What if some labels are wrong?
  - Want to be flexible and disagree with given labels occasionally
- It cannot handle new test points directly
  - f is only defined on X_u
  - We have to add new test points to the graph, and find a new harmonic solution

Local and Global consistency

- Allow $f(X_l)$ to be different from Y_l, but penalize it
- Introduce a balance between labeled data fit and graph energy

$$\min_f \sum_{i=1}^{l} (f(x_i) - y_i)^2 + \lambda f^\intercal \Delta f$$

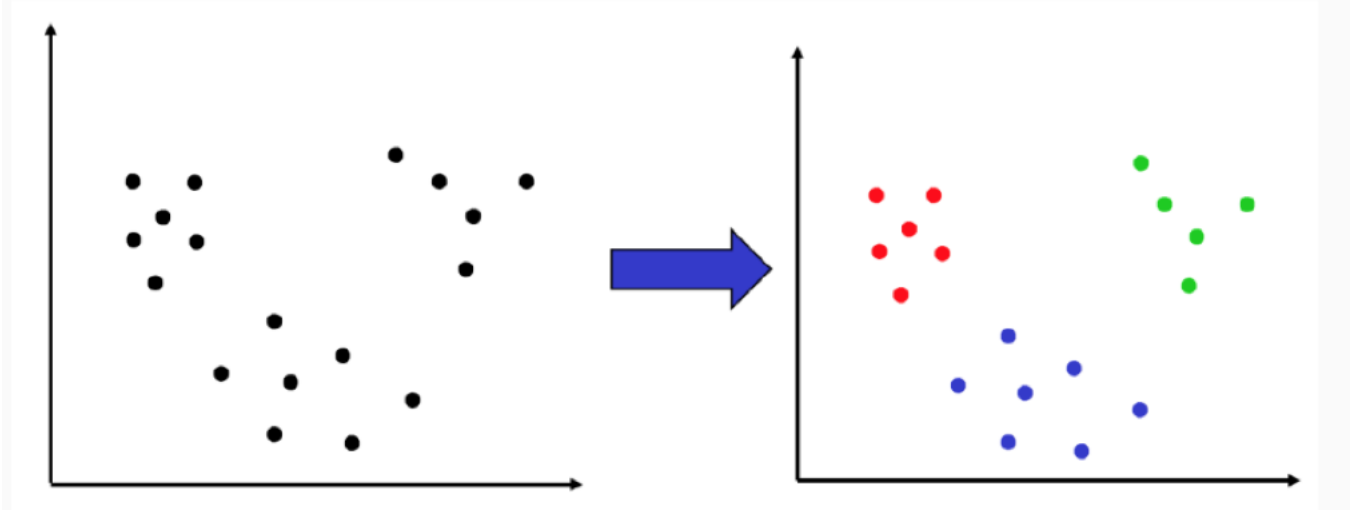
Summary of graph-based method

- Pros:
  - Clear mathematical framework
  - Performance is strong if the graph happens to fit the task
  - Can be extended to directed graphs
- Cons:
  - Performance is bad if the graph is bad
  - Sensitive to graph structure and edge weights

Semi-supervised Clustering

Semi-supervised clustering

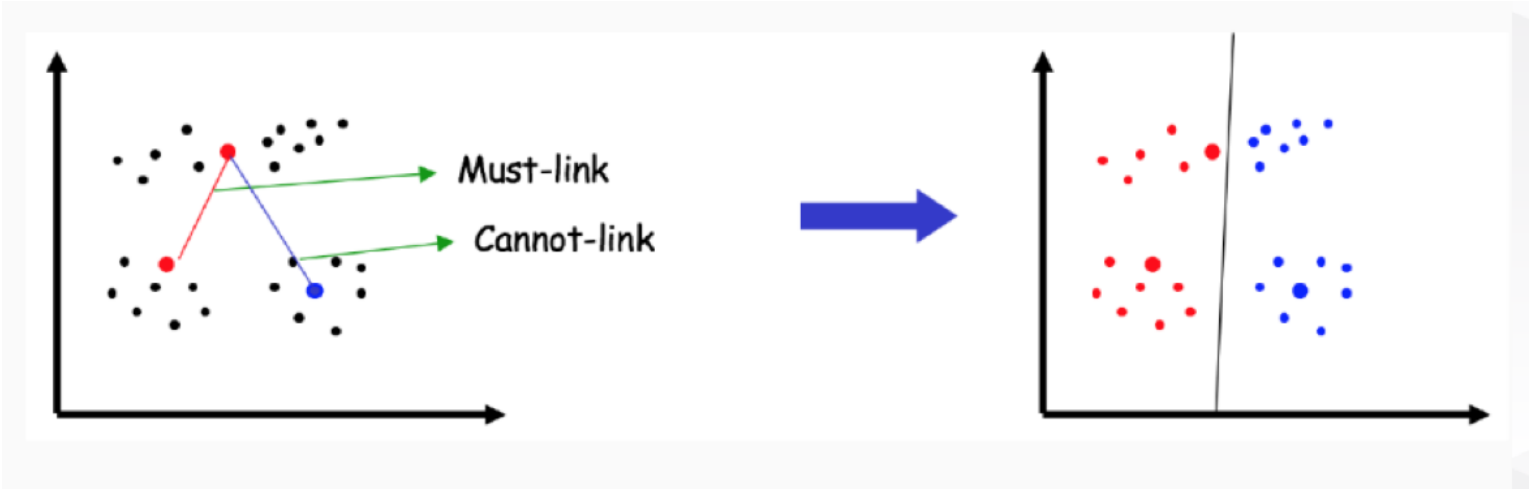
- Clustering is a method of unsupervised learning



- Semi-supervised clustering: clustering with a large amount of domain knowledge

Semi-supervised clustering

- According to different given domain knowledge:
  - Users know about which few documents are related (must-link) or unrelated (cannot-link)



Wagstaff K, Cardie C, Rogers S, et al. Constrained k-means clustering with background knowledge[C]//Icml. 2001, 1: 577-584.

Semi-supervised clustering

- Constrained K-means

Table 1. Constrained 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 $\{C_1 \ldots C_k\}$.

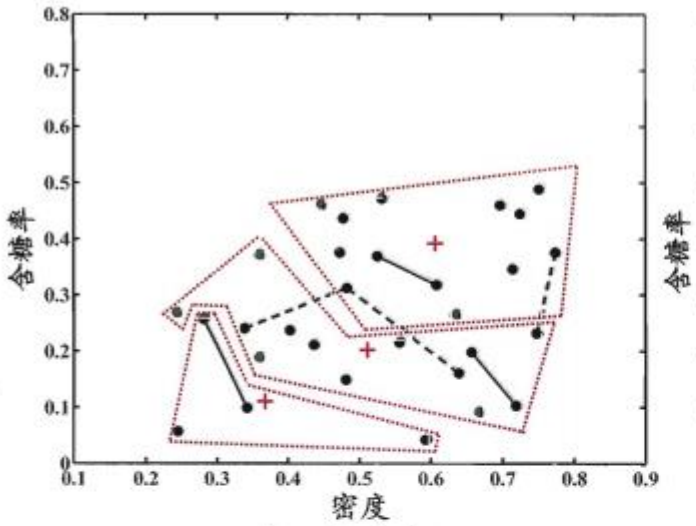
VIOLATE-CONSTRAINTS(data point d, cluster C, must-link constraints $Con_= \subseteq D \times D$, cannot-link constraints $Con_\neq \subseteq D \times D$)

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.

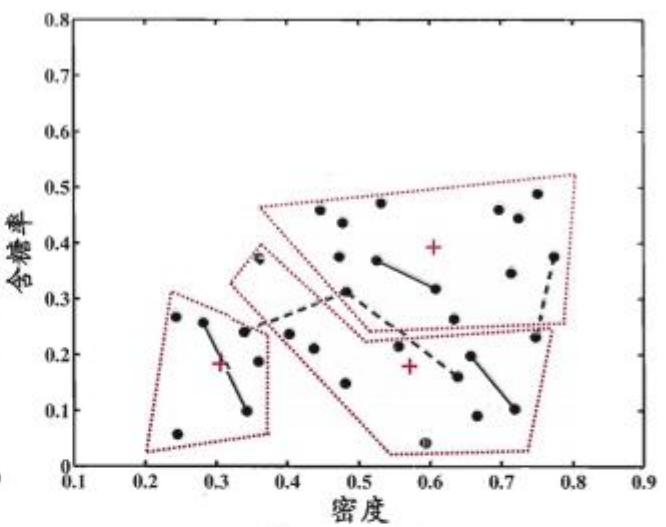
Wagstaff K, Cardie C, Rogers S, et al. Constrained k-means clustering with background knowledge[C]//Icml. 2001, 1: 577-584.

Semi-supervised clustering

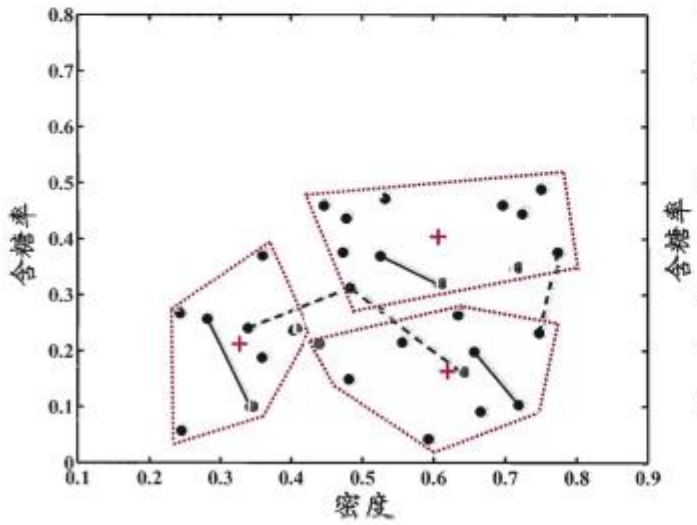
编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	16	0.593	0.042
2	0.774	0.376	17	0.719	0.103
3	0.634	0.264	18	0.359	0.188
4	0.608	0.318	19	0.339	0.241
5	0.556	0.215	20	0.282	0.257
6	0.403	0.237	21	0.748	0.232
7	0.481	0.149	22	0.714	0.346
8	0.437	0.211	23	0.483	0.312
9	0.666	0.091	24	0.478	0.437
10	0.243	0.267	25	0.525	0.369
11	0.245	0.057	26	0.751	0.489
12	0.343	0.099	27	0.532	0.472
13	0.639	0.161	28	0.473	0.376
14	0.657	0.198	29	0.725	0.445
15	0.360	0.370	30	0.446	0.459



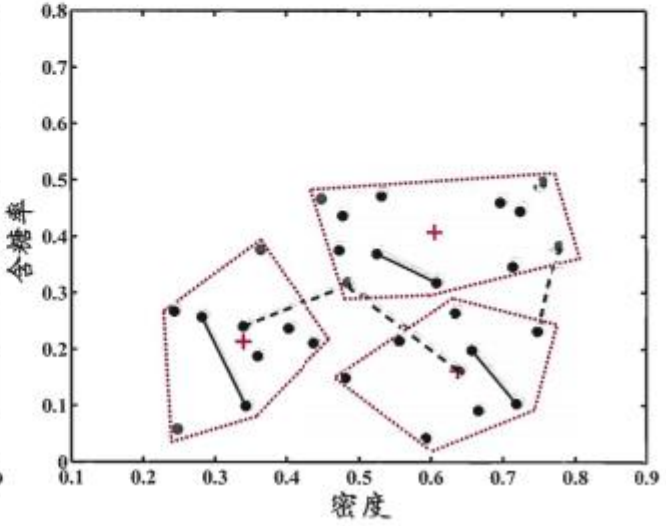
(a) 第 1 轮迭代后



(b) 第 2 轮迭代后



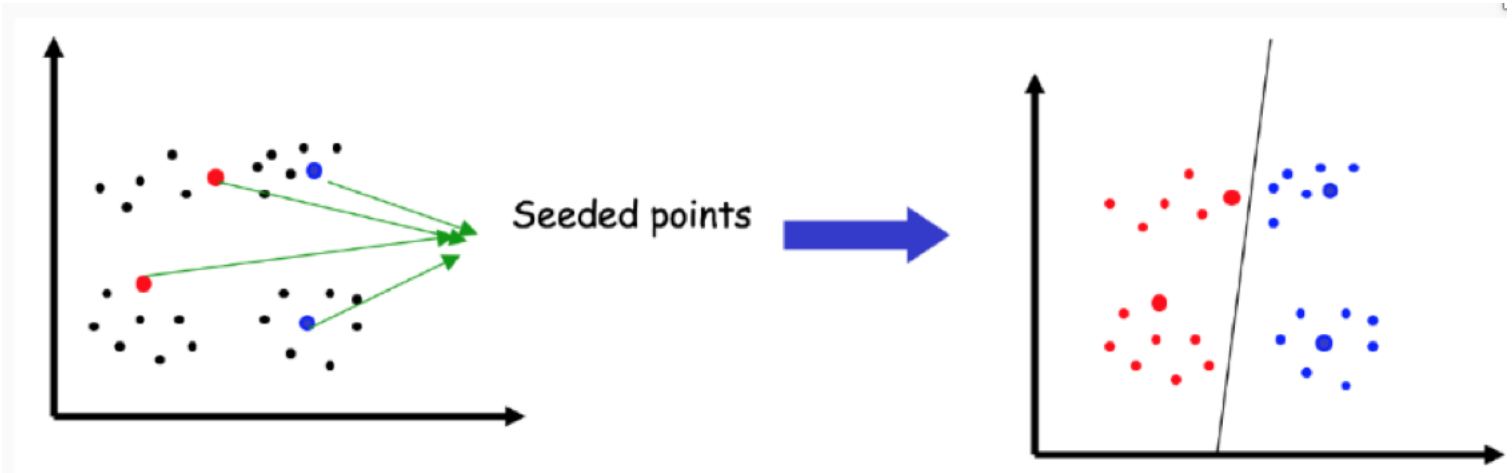
(c) 第 3 轮迭代后



(d) 第 4 轮迭代后

Semi-supervised clustering

- According to different given domain knowledge:
  - Users provide class labels (seeded points) a priori to some of the documents



Basu S, Banerjee A, Mooney R. Semi-supervised clustering by seeding[C]//In Proceedings of 19th International Conference on Machine Learning (ICML-2002. 2002.

Semi-supervised clustering

- Seeded K-means

The label for seeded points may change:

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. $t \leftarrow (t+1)$

Figure 1. Seeded-KMeans algorithm

The label for seeded points will not change:

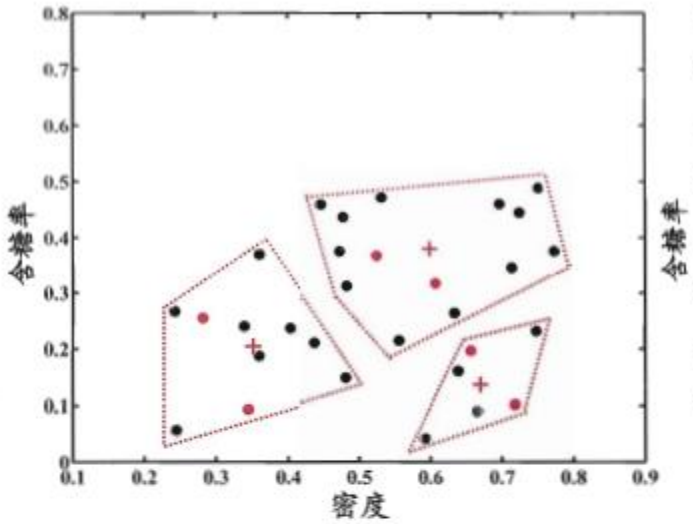
Algorithm: `Constrained-KMeans`
Input: Set of data points $\mathcal{X} = \{x_1, \cdots, x_N\}, x_i \in \mathbb{R}^d$, number of clusters K, set $\mathcal{S} = \cup_{l=1}^{K} \mathcal{S}_l$ of initial seeds
Output: Disjoint K partitioning $\{\mathcal{X}_l\}_{l=1}^{K}$ of $\mathcal{X}$ such that the KMeans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

Figure 2. Constrained-KMeans algorithm

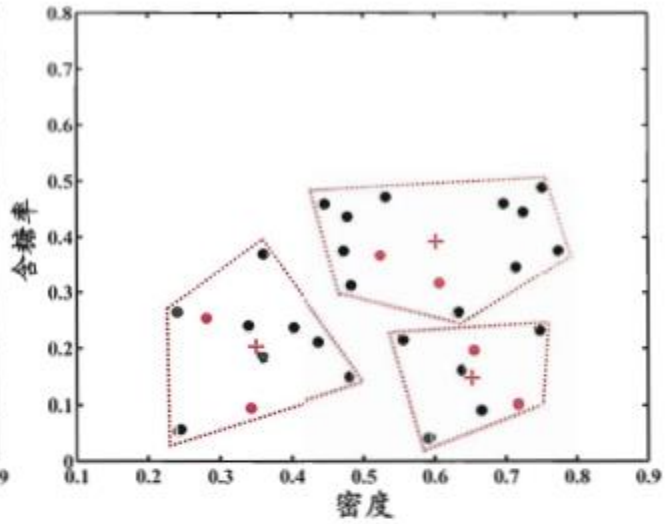
Basu S, Banerjee A, Mooney R. Semi-supervised clustering by seeding[C]//In Proceedings of 19th International Conference on Machine Learning (ICML-2002. 2002.

Semi-supervised clustering

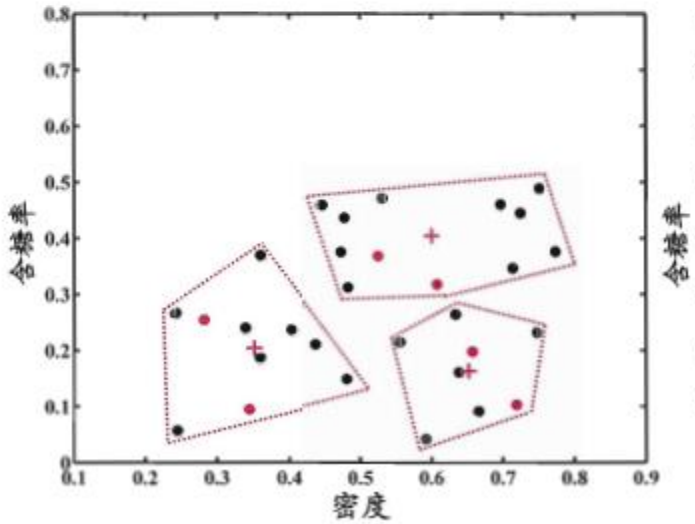
编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	16	0.593	0.042
2	0.774	0.376	17	0.719	0.103
3	0.634	0.264	18	0.359	0.188
4	0.608	0.318	19	0.339	0.241
5	0.556	0.215	20	0.282	0.257
6	0.403	0.237	21	0.748	0.232
7	0.481	0.149	22	0.714	0.346
8	0.437	0.211	23	0.483	0.312
9	0.666	0.091	24	0.478	0.437
10	0.243	0.267	25	0.525	0.369
11	0.245	0.057	26	0.751	0.489
12	0.343	0.099	27	0.532	0.472
13	0.639	0.161	28	0.473	0.376
14	0.657	0.198	29	0.725	0.445
15	0.360	0.370	30	0.446	0.459



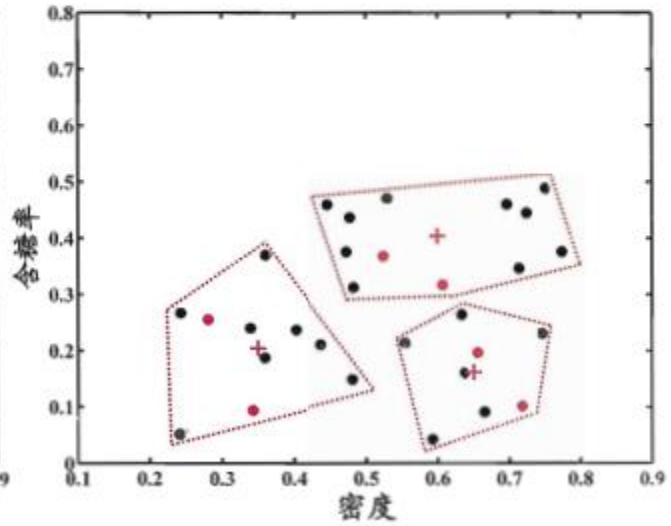
(a) 第 1 轮迭代后



(b) 第 2 轮迭代后



(c) 第 3 轮迭代后



(d) 第 4 轮迭代后

Summary

- General idea: Learning from both labeled and unlabeled data
- Assumption:
  - Smooth assumption (Generative)
  - Cluster assumption (S3VM)
  - Manifold assumption (Graph-based)
  - Independent assumption (Co-training)
- Challenges:
  - Other assumptions?
  - Efficiency
- two ways to use unlabeled data:
  - in the loss function (S3VM, co-training)
    - non-convex – optimization method matters!
  - in the regularizer (graph methods)
    - convex, but graph construction matters