Semi-Supervised Learning

Barnabas Poczos

Slides Courtesy: Jerry Zhu, Aarti Singh

Supervised Learning

Feature Space $\mathcal{X}$ **Label** Space $\mathcal{Y}$

Goal: Construct a **predictor** $f : \mathcal{X} \to \mathcal{Y}$ to minimize

$$R(f) \equiv \mathbb{E}_{XY}\left[\text{loss}(Y, f(X))\right]$$

Optimal predictor (Bayes Rule) depends on unknown P_{XY}, so instead *learn* a good prediction rule from training data $\{(X_i, Y_i)\}_{i=1}^{n} \overset{\text{iid}}{\sim} P_{XY}$(unknown)

Training data $\{(X_i, Y_i)\}_{i=1}^{n}$ $\Rightarrow$ **Learning algorithm** $\Rightarrow$ Prediction rule $\widehat{f}_n$

Labeled

Labeled and Unlabeled data

“Crystal” “Needle” “Empty”

“0” “1” “2” …

Human expert/
Special equipment/
Experiment

“Sports”
“News”
“Science”
…

Unlabeled data, X_i

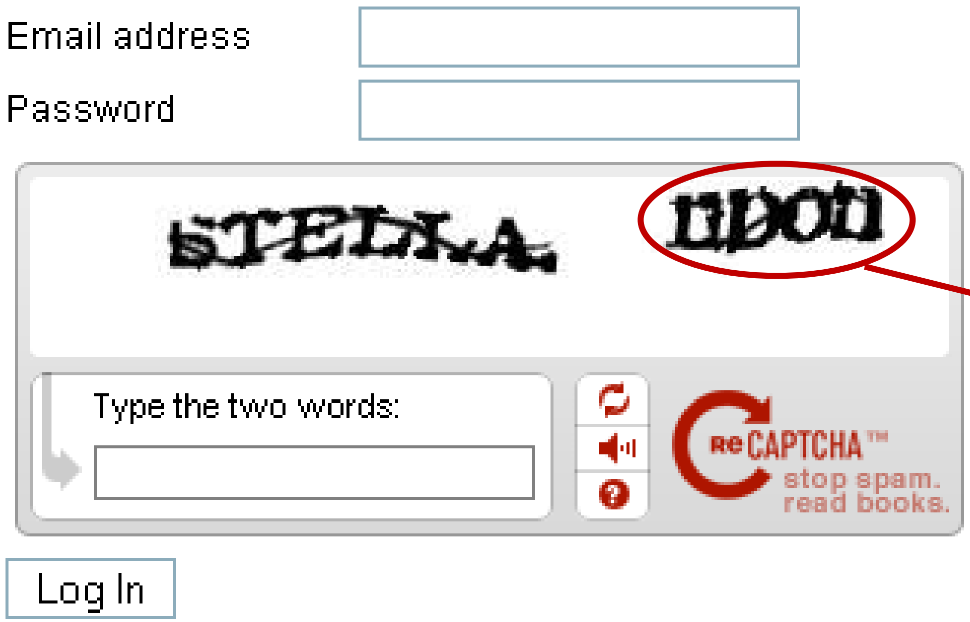
Labeled data, Y_i

Cheap and abundant !

Expensive and scarce !

Free-of-cost labels?

Luis von Ahn: Games with a purpose (ReCaptcha)



Word challenging to OCR
(Optical Character Recognition)

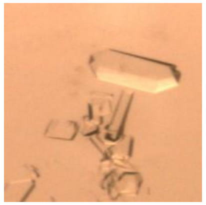
You provide a free label!

Semi-Supervised learning

Training data $\{(X_i, Y_i)\}_{i=1}^n$ $\{X_i\}_{i=1}^m$ ⇨ **Learning algorithm** ⇨ Prediction rule $\widehat{f}_{n,m}$

Supervised learning (SL)

Labeled data $\{X_i, Y_i\}_{i=1}^n$



X_i

"Crystal"

Y_i

Semi-Supervised learning (SSL)

Labeled data $\{X_i, Y_i\}_{i=1}^n$ **and** Unlabeled data $\{X_i\}_{i=1}^m$ $m \gg n$

Goal: Learn a better prediction rule than based on labeled data alone.

Semi-Supervised learning in Humans

Cognitive science

Computational model of how humans learn from labeled and unlabeled data.

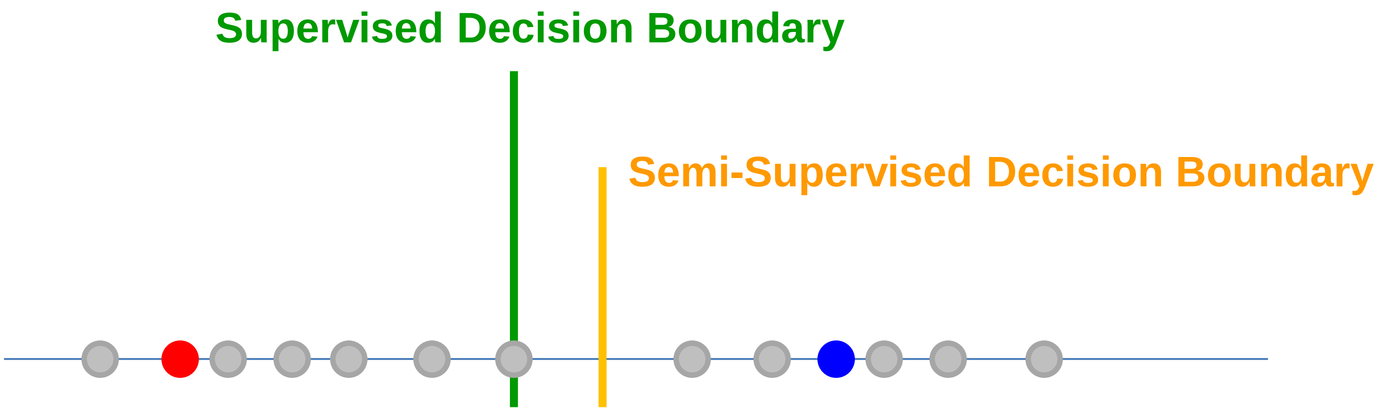
- concept learning in children: $\mathbf{x}$=animal, y=concept (e.g., dog)
- Daddy points to a brown animal and says “dog!”
- Children also observe animals by themselves

Can unlabeled data help?

Positive labeled data

Negative labeled data

Unlabeled data



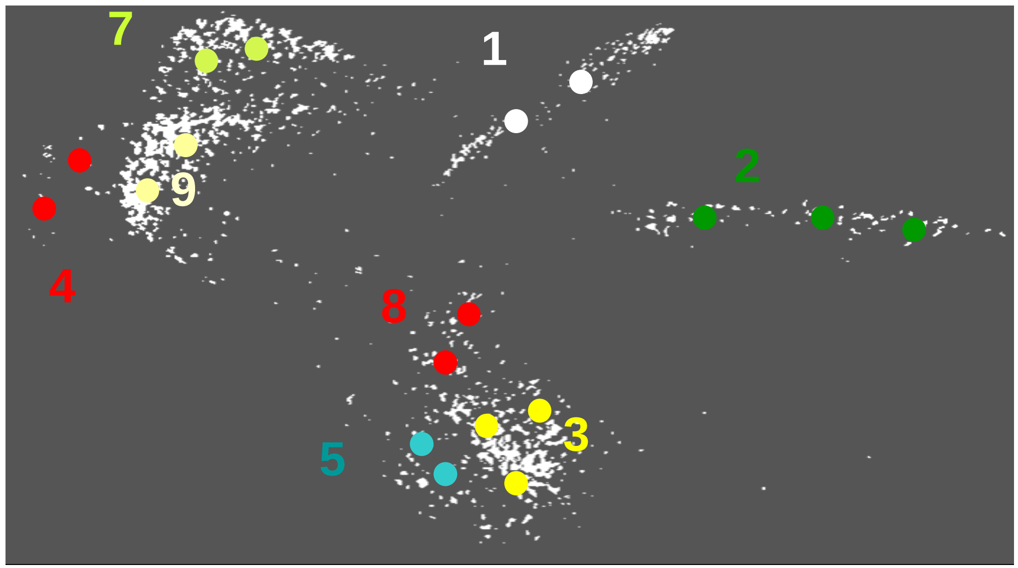
Assume each class is a coherent group (e.g. Gaussian)

Then unlabeled data can help identify the boundary more accurately.

Can unlabeled data help?

Unlabeled Images

Labels "0" "1" "2" …



This embedding can be done by manifold learning algorithms

"Similar" data points have "similar" labels

Some SSL Algorithms

- Self-Training
- Generative methods, mixture models
- Graph-based methods
- Co-Training
- Semi-supervised SVM
- Many others

Notation

- instance $\mathbf{x}$, label y
- learner $f : \mathcal{X} \mapsto \mathcal{Y}$
- labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$
- unlabeled data $X_u = \{\mathbf{x}_{l+1:l+u}\}$, available during training. Usually $l \ll u$. Let $n = l + u$
- test data $\{(x_{n+1\ldots}, y_{n+1\ldots})\}$, not available during training

Self-training

Our first SSL algorithm:

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat:
3. Train f from L using supervised learning.
4. Apply f to the unlabeled instances in U.
5. Remove a subset S from U; add $\{(\mathbf{x}, f(\mathbf{x})) | \mathbf{x} \in S\}$ to L.

Self-training is a *wrapper* method

- the choice of learner for f in step 3 is left completely open
- good for many real world tasks like natural language processing
- but mistake by f can reinforce itself

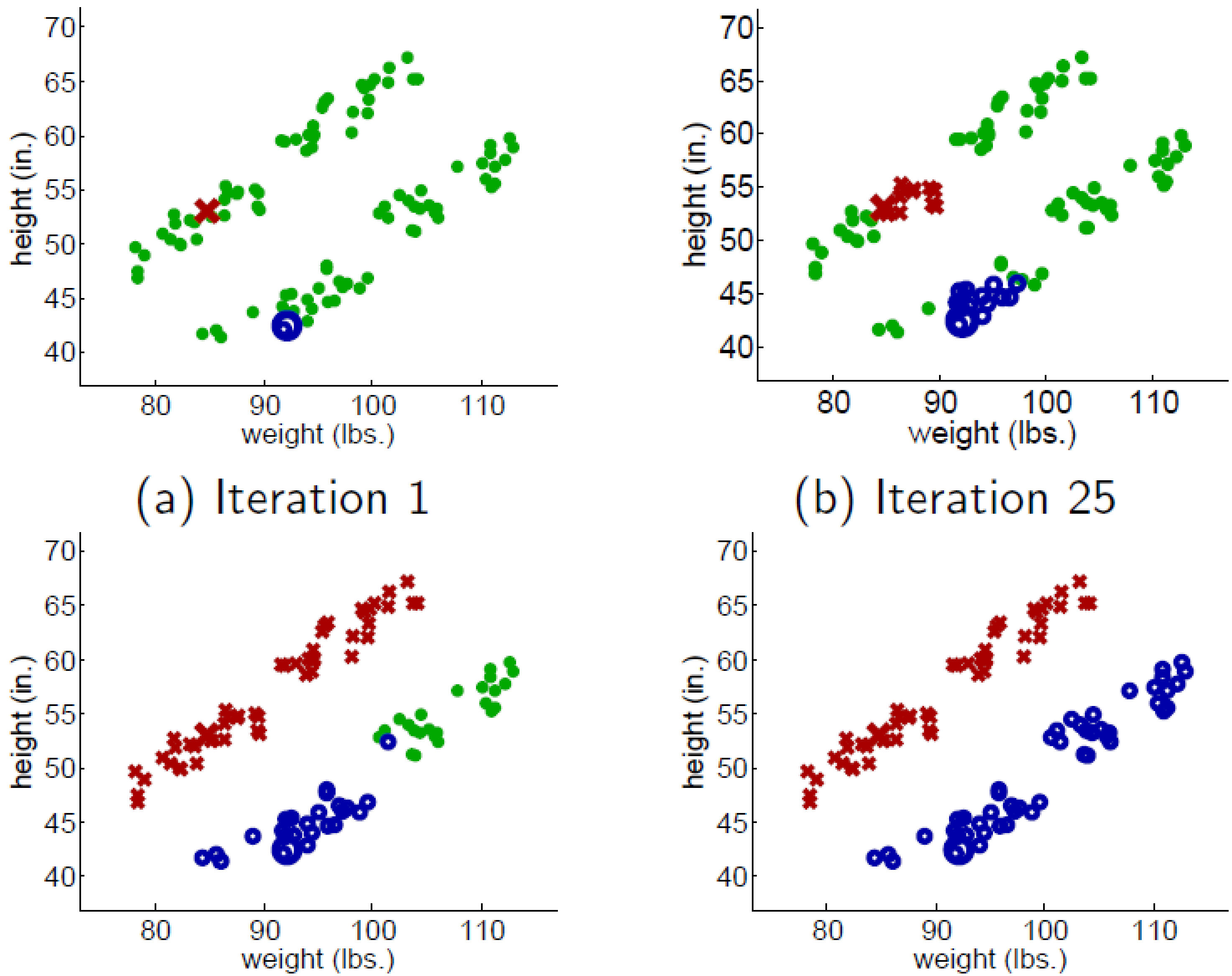
Self-training Example

Propagating 1-NN

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, distance function $d()$.

1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat until U is empty:
3. Select $\mathbf{x} = \mathrm{argmin}_{\mathbf{x} \in U} \min_{\mathbf{x}' \in L} d(\mathbf{x}, \mathbf{x}')$.
4. Set $f(\mathbf{x})$ to the label of $\mathbf{x}$'s nearest instance in L. Break ties randomly.
5. Remove $\mathbf{x}$ from U; add $(\mathbf{x}, f(\mathbf{x}))$ to L.

Propagating 1-Nearest-Neighbor: now it works



(a) Iteration 1

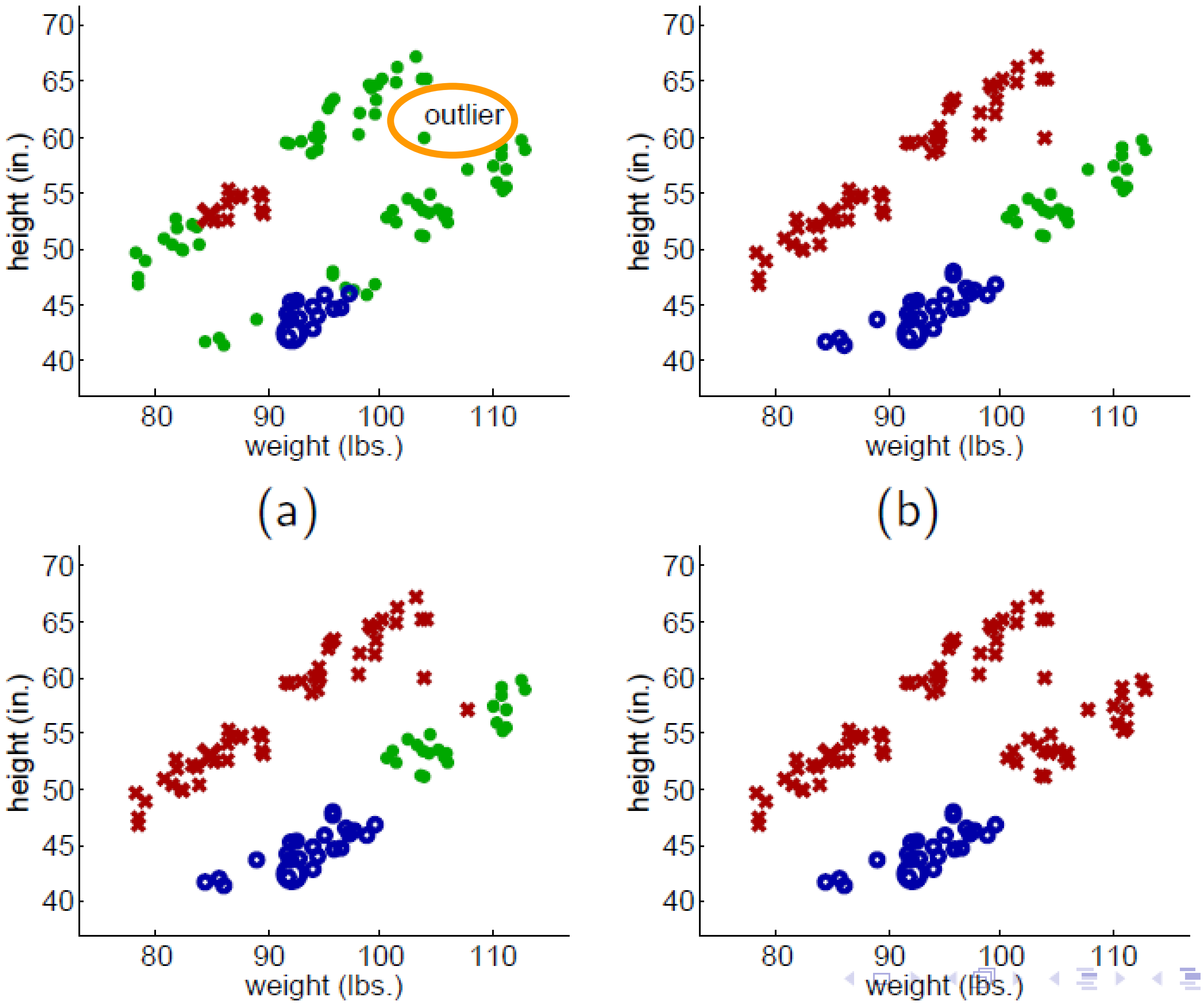
(b) Iteration 25

(c) Iteration 74

(d) Final labeling of all instances

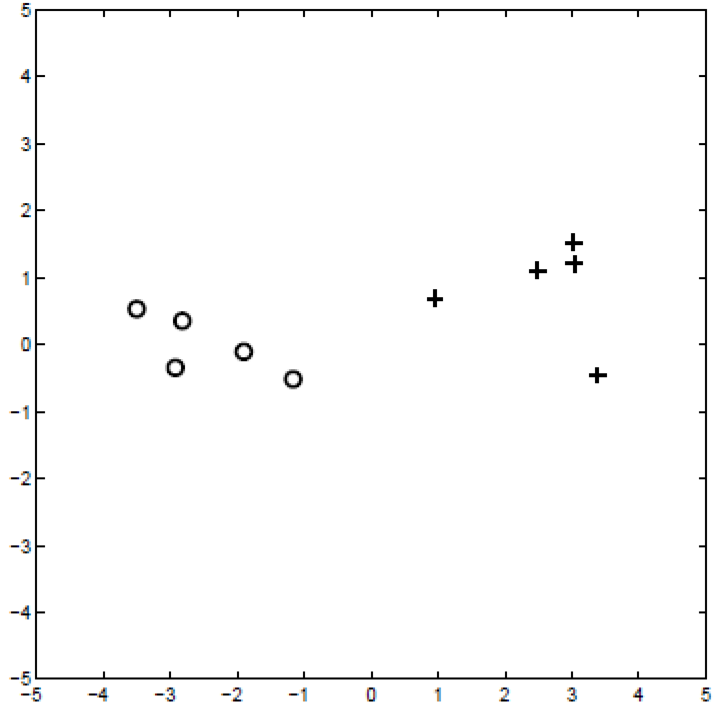
Propagating 1-Nearest-Neighbor: now it doesn't

But with a single outlier...



Mixture Models for Labeled Data

Labeled data (X_l, Y_l):



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

Mixture Models for Labeled Data

Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$
The GMM:

Estimate the parameters from the labeled data

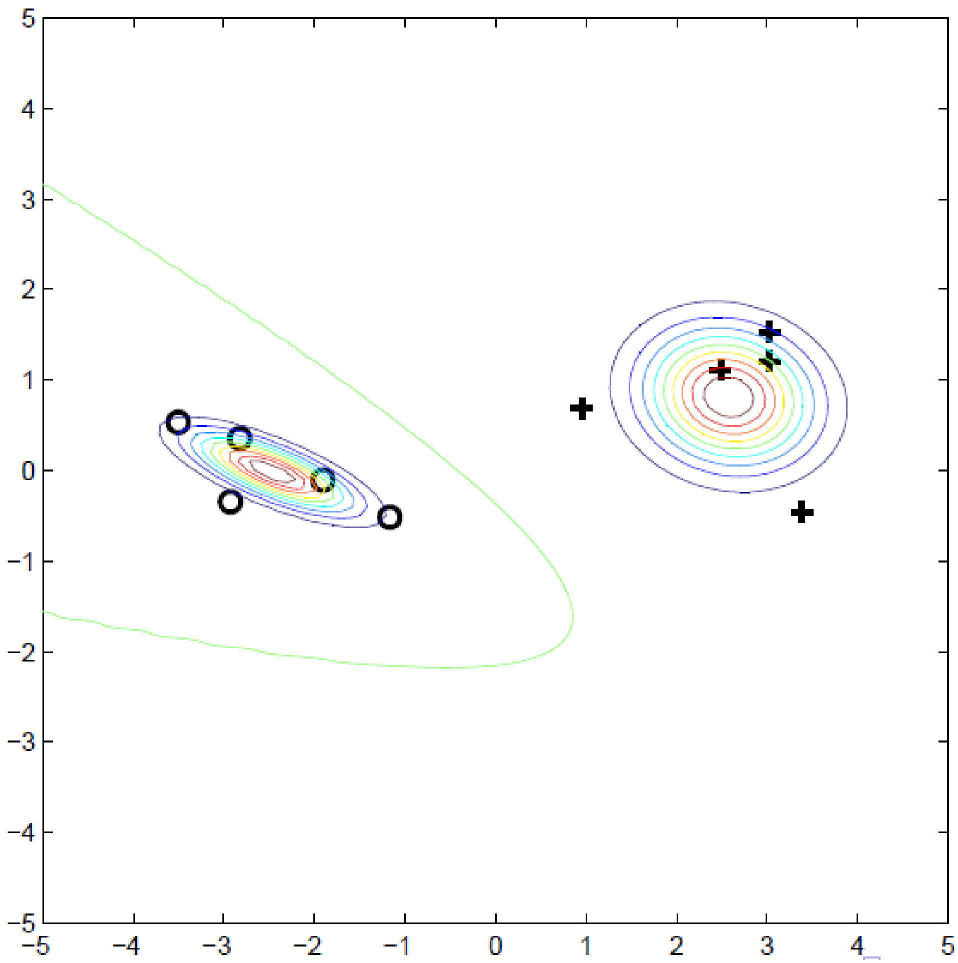
$$\begin{aligned} p(x, y|\theta) &= p(y|\theta)p(x|y, \theta) \\ &= w_y \mathcal{N}(x; \mu_y, \Sigma_y) \end{aligned}$$

Classification: $p(y|x, \theta) = \frac{p(x,y|\theta)}{\sum_{y'} p(x,y'|\theta)} \gtrless 1/2$

Decision for any test point not in the labeled dataset

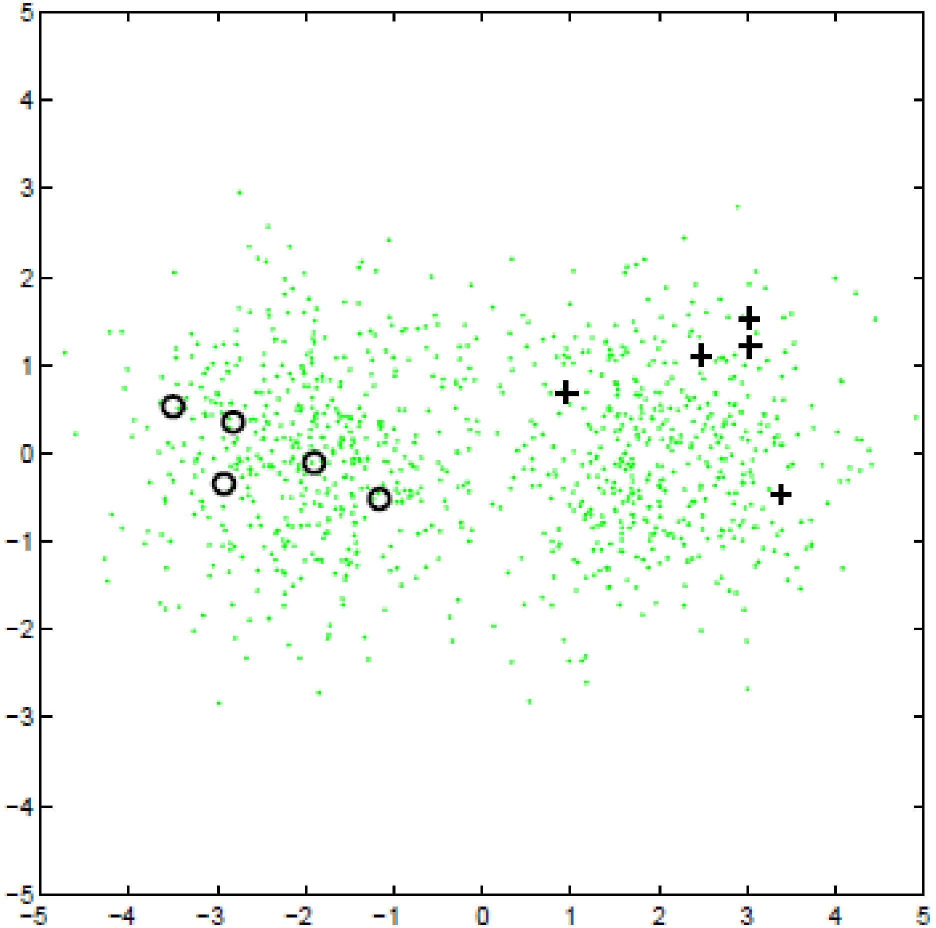
Mixture Models for Labeled Data

The most likely model, and its decision boundary:



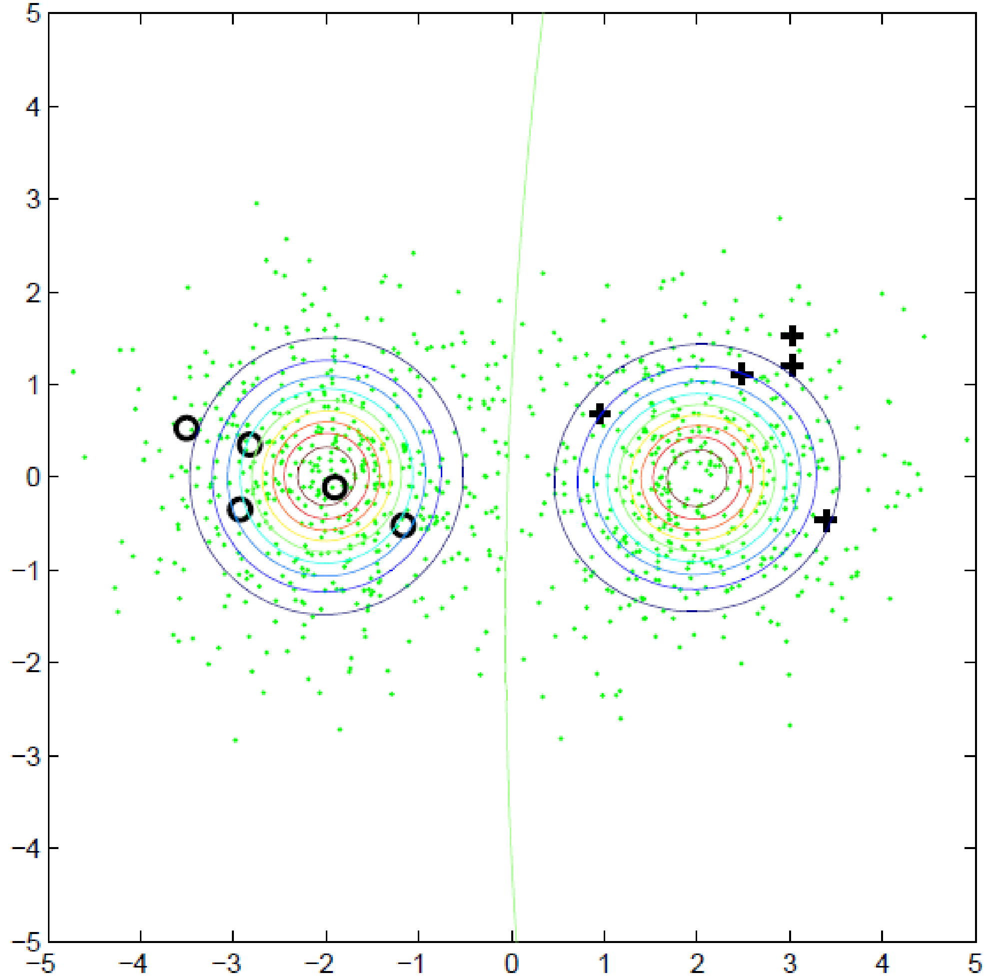
Mixture Models for SSL Data

Adding unlabeled data:



Mixture Models

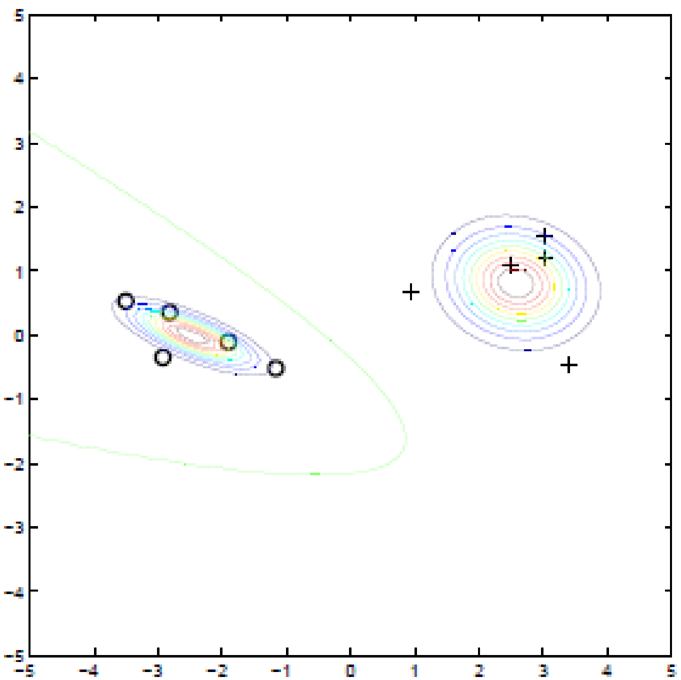
With unlabeled data, the most likely model and its decision boundary:

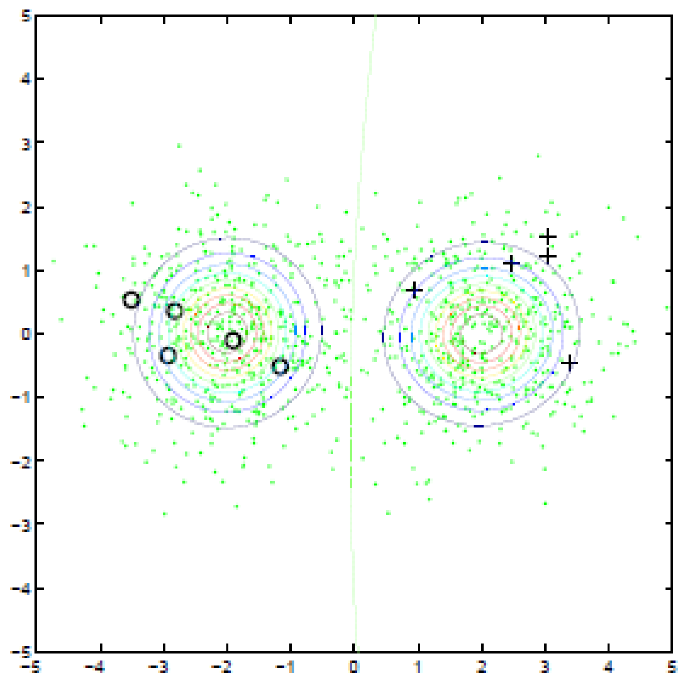


Mixture Models
SL vs SSL

They are different because they maximize different quantities.

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





Mixture Models

Assumption

knowledge of the model form $p(X, Y|\theta)$.

- joint and marginal likelihood

$$p(X_l, Y_l, X_u|\theta) = \sum_{Y_u} p(X_l, Y_l, X_u, Y_u|\theta)$$

- find the maximum likelihood estimate (MLE) of θ, the maximum a posteriori (MAP) estimate, or be Bayesian

Gaussian Mixture Models

Binary classification with GMM using MLE.

- with only labeled data
  - $\log p(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i,\theta)$
  - MLE for θ trivial (sample mean and covariance)

- with both labeled and unlabeled data

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

  - MLE harder (hidden variables): EM

EM for Gaussian Mixture Models

1. Start from MLE $\theta = \{w, \mu, \Sigma\}_{1:2}$ on (X_l, Y_l),
   - w_c=proportion of class c
   - μ_c=sample mean of class c
   - Σ_c=sample cov of class c

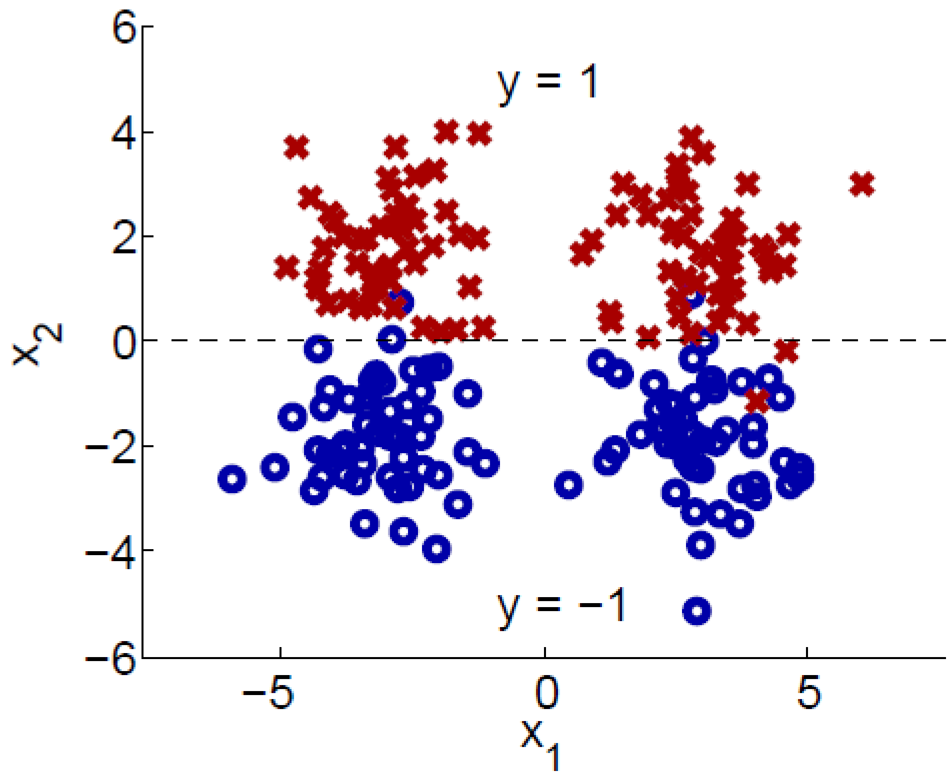
   repeat:

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

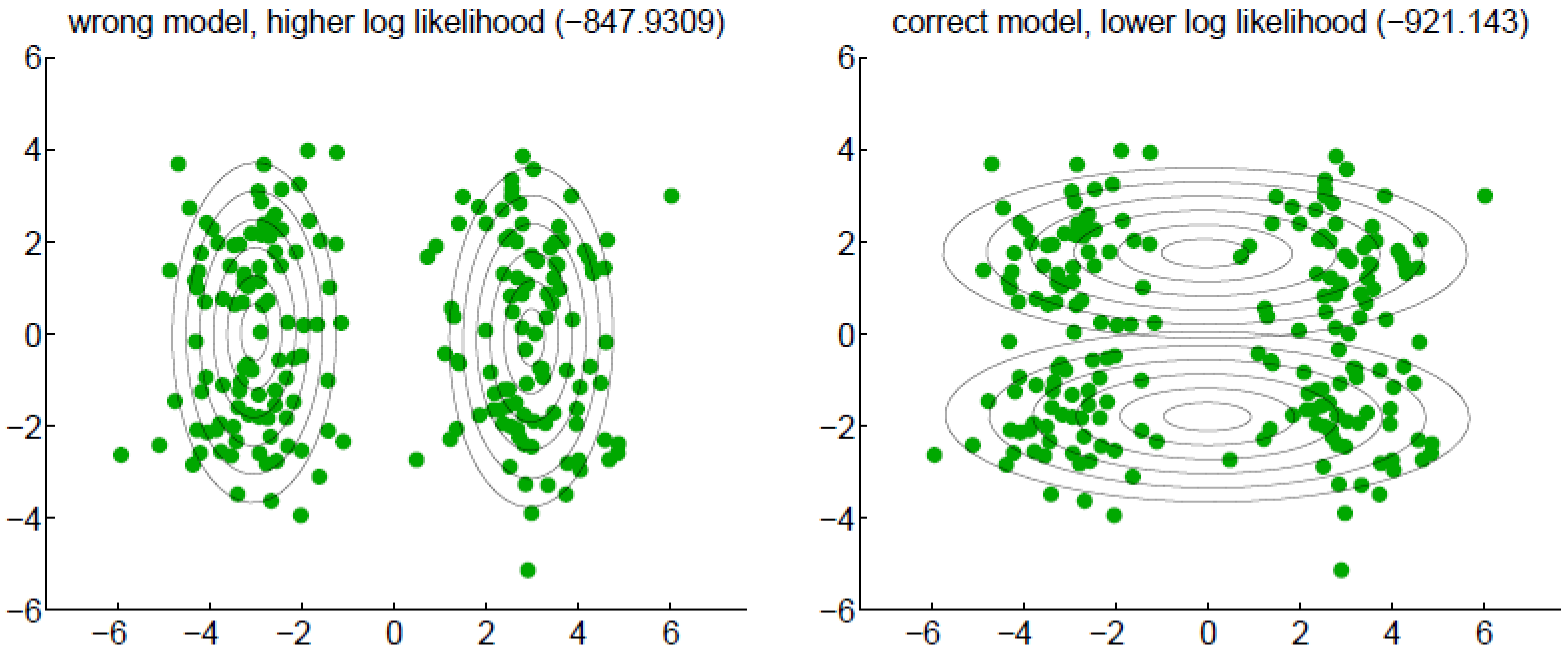
3. The M-step: update MLE θ with (now labeled) X_u

Assumption for GMMs

- **Assumption**: the data actually comes from the mixture model, where the number of components, prior $p(y)$, and conditional $p(\mathbf{x}|y)$ are all correct.
- When the assumption is wrong:



Assumption for GMMs



Assumption for GMMs

Heuristics to lessen the danger

- Carefully construct the generative model, e.g., multiple Gaussian distributions per class

- Down-weight the unlabeled data ($\lambda < 1$)

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

Related: Cluster and Label

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$,
a clustering algorithm $\mathcal{A}$, a supervised learning algorithm $\mathcal{L}$

1. Cluster $\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}$ using $\mathcal{A}$.
2. For each cluster, let S be the labeled instances in it:
3. Learn a supervised predictor from S: $f_S = \mathcal{L}(S)$.
4. Apply f_S to all unlabeled instances in this cluster.

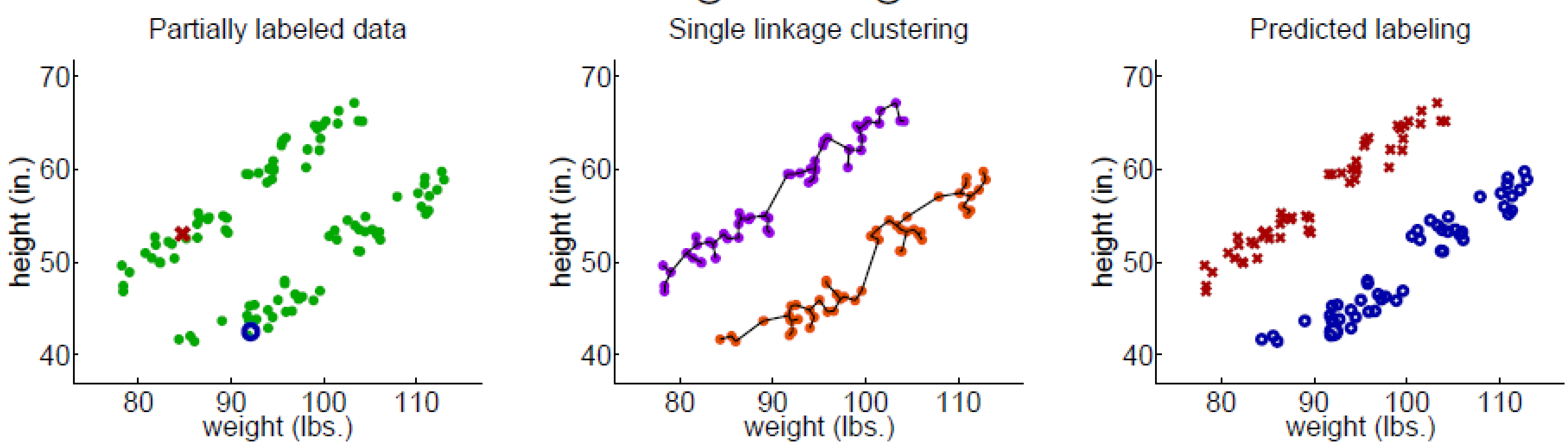
Output: labels on unlabeled data $y_{l+1}, \ldots, y_{l+u}$.

But again: **SSL sensitive to assumptions**—in this case, that the clusters coincide with decision boundaries. If this assumption is incorrect, the results can be poor.

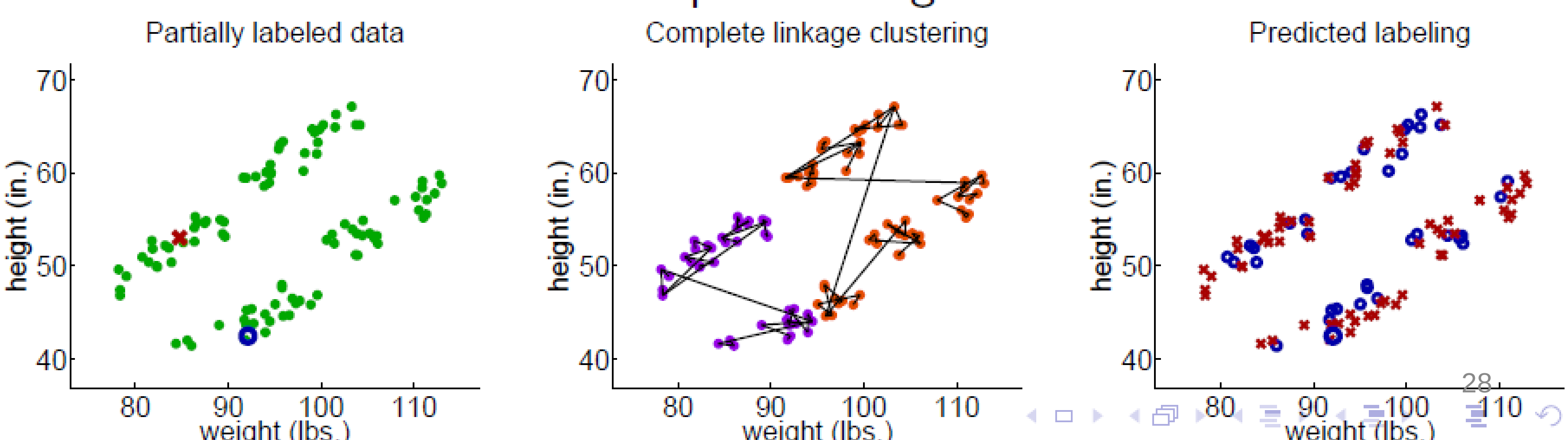
Cluster-and-label: now it works, now it doesn't

Example: $\mathcal{A}$=Hierarchical Clustering, $\mathcal{L}$=majority vote.

single linkage



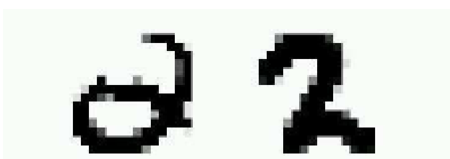
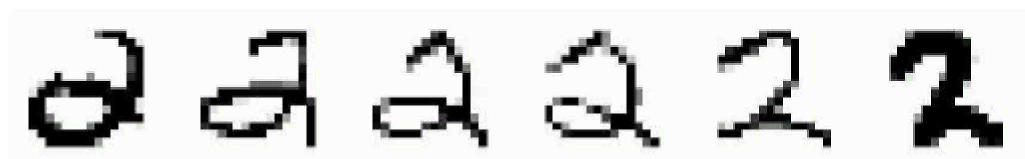
complete linkage



Graph Based Methods

Assumption: Similar unlabeled data have similar labels.

Handwritten digits recognition with pixel-wise Euclidean distance

 not similar	 'indirectly' similar with stepping stones

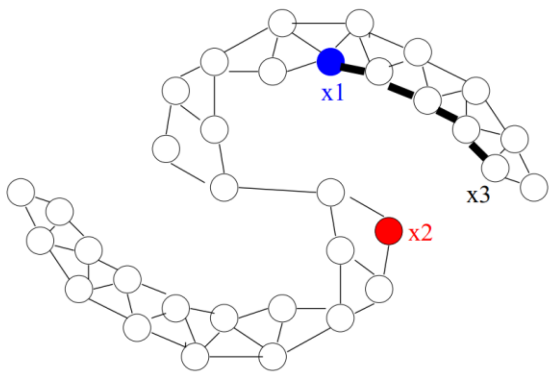
Graph Regularization

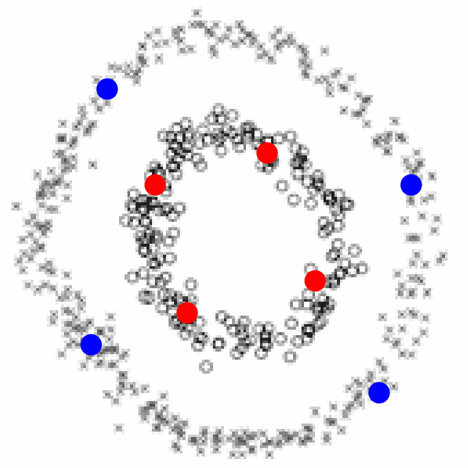
Similarity Graphs: Model local neighborhood relations between data points

- 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

    $w_{ij} = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$
  - ϵ-radius graph

Assumption:
Nodes connected by heavy edges tend to have similar label





Graph Regularization

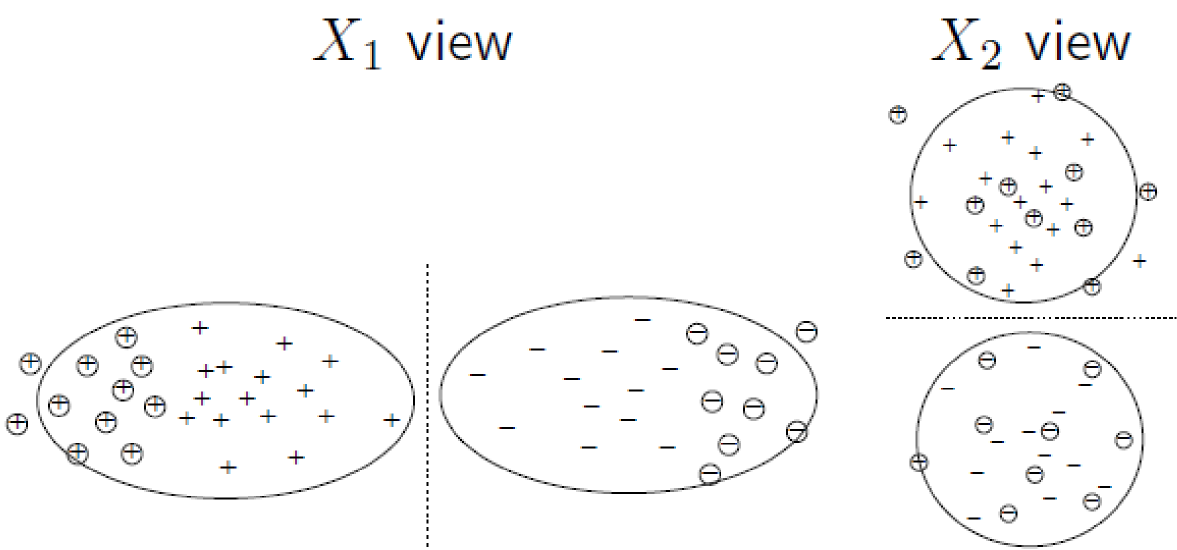
If data points i and j are similar (i.e. weight w_{ij} is large), then their labels are similar $f_i = f_j$

$$\min_f \underbrace{\sum_{i \in l} (y_i - f_i)^2}_{\substack{\text{Loss on labeled data} \\ \text{(mean square,0-1)}}} + \lambda \underbrace{\sum_{i,j \in l,u} w_{ij} (f_i - f_j)^2}_{\substack{\text{Graph based smoothness prior} \\ \text{on labeled and unlabeled data}}}$$

Co-training

Assumptions

- feature split $x = [x^{(1)}; x^{(2)}]$ exists
- $x^{(1)}$ or $x^{(2)}$ alone is sufficient to train a good classifier



Co-training Algorithm

Co-training (Blum & Mitchell, 1998) (Mitchell, 1999) assumes that
(i) features can be split into two sets;
(ii) each sub-feature set is sufficient to train a good classifier.

- Initially two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively.
- Each classifier then classifies the unlabeled data, and 'teaches' the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident.
- Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats.

Co-training Algorithm

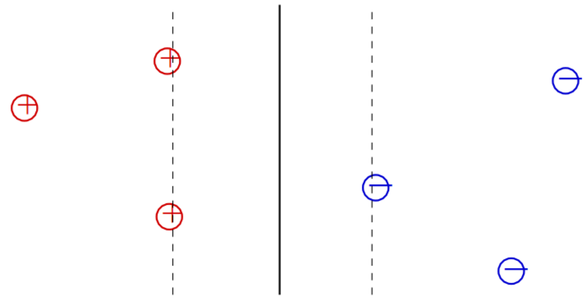
Blum & Mitchell'98

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$
each instance has two views $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$,
and a learning speed k.

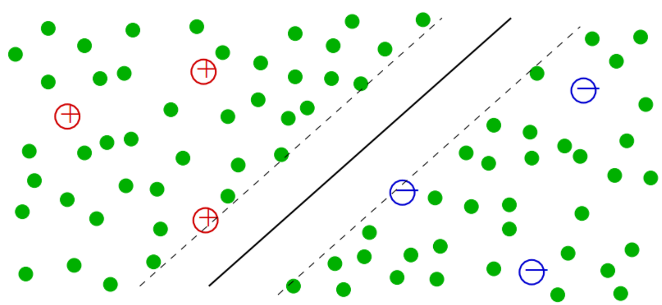
1. let $L_1 = L_2 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$.
2. Repeat until unlabeled data is used up:
3. Train view-1 $f^{(1)}$ from L_1, view-2 $f^{(2)}$ from L_2.
4. Classify unlabeled data with $f^{(1)}$ and $f^{(2)}$ separately.
5. Add $f^{(1)}$'s top k most-confident predictions $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ to L_2.
   Add $f^{(2)}$'s top k most-confident predictions $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ to L_1.
   Remove these from the unlabeled data.

Semi-Supervised SVMs

SVMs



Semi-supervised SVMs (S3VMs) = Transductive SVMs (TSVMs)



Assumption: Unlabeled data from different classes are separated with large margin.

Semi-Supervised Learning

- Generative methods
- Graph-based methods
- Co-Training
- Semi-Supervised SVMs
- Many other methods

SSL algorithms can use unlabeled data to help improve prediction accuracy **if data satisfies appropriate assumptions**