CPSC 340: Machine Learning and Data Mining

Semi-Supervised Learning

Fall 2015

Admin

- Assignment 3 grades posted this weekend (with mark breakdowns).
- Assignment 5:
  - Tutorial slides posted.
  - Due Friday of next week.

Last Time: Loss Functions

- We discussed loss functions:
  - Continuous: Max, Squared, Absolute, Square-Root.
  - Binary labels: logistic, hinge, extreme.
  - Categorical: softmax.
  - Ordinal: ordinal logistic.
  - Counting: Poisson.
- While squared loss is convenient, there are usually better choices.

Last Time: Loss Functions

- We also discussed how to use probabilities to derive loss functions:

If $\hat{y}_i = w^T x_i$ (or any other model) then define some $p(y_i \mid \hat{y}_i)$ and use $-\log(p(y_i \mid \hat{y}_i))$ as loss.

- If you aren't happy with existing losses, use this to derive your own.
- This lets us write training in terms of probabilities:

Instead of $\underset{w \in \mathbb{R}^d}{\text{argmin}} \frac{1}{2} \sum_{i=1}^{n} (y_i - w^T x_i)^2$ we can use $\underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} -\log(p(y_i \mid w, x_i))$

Today: Semi-Supervised Learning

- Our usual supervised learning framework:

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0	0.7	0	0.3	0	0	
0.3	0.7	0	0.6	0	0.01	
0	0	0	0.8	0	0	
0.3	0.7	1.2	0	0.10	0.01	

Sick?
1
1
0
1

- In semi-supervised learning, we also have unlabeled examples:

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…
0.3	0	1.2	0.3	0.10	0.01	
0.6	0.7	0	0.3	0	0.01	
0	0.7	0	0.6	0	0	
0.3	0.7	0	0	0.20	0.01	

Semi-Supervised Learning

- The semi-supervised learning (SSL) framework:

$$X = \underbrace{\begin{bmatrix} & & \\ \end{bmatrix}}_{n \times d} \quad y = \underbrace{\begin{bmatrix} \\ \end{bmatrix}}_{n \times 1} \quad \tilde{X} = \underbrace{\begin{bmatrix} & & \\ & & \\ \end{bmatrix}}_{t \times d}$$

- This arises a lot:
  - Usually getting unlabeled data is easy and getting labeled data is hard.
  - Why build a classifier if getting labels is easy?
- Common situation:
  - A small number of labeled examples.
  - A huge number of unlabeled examples: t >> n.

Transductive vs. Inductive SSL

- Transductive SSL:
  - Only interested in labels of the unlabeled examples.

$$X = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{n \times d} \quad y = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{n \times 1} \quad \tilde{X} = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{t \times d} \quad \tilde{y} = \underbrace{\begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}}_{t \times 1}$$

Transductive vs. Inductive SSL

- Transductive SSL:
  - Only interested in labels of the unlabeled examples.
- Inductive SSL:
  - Interested in the test set performance on new examples.

Training

$$X = \underset{n \times d}{\left[\qquad\right]} \quad y = \underset{n \times 1}{\left[\quad\right]} \quad \tilde{X} = \underset{t \times d}{\left[\qquad\qquad\right]}$$

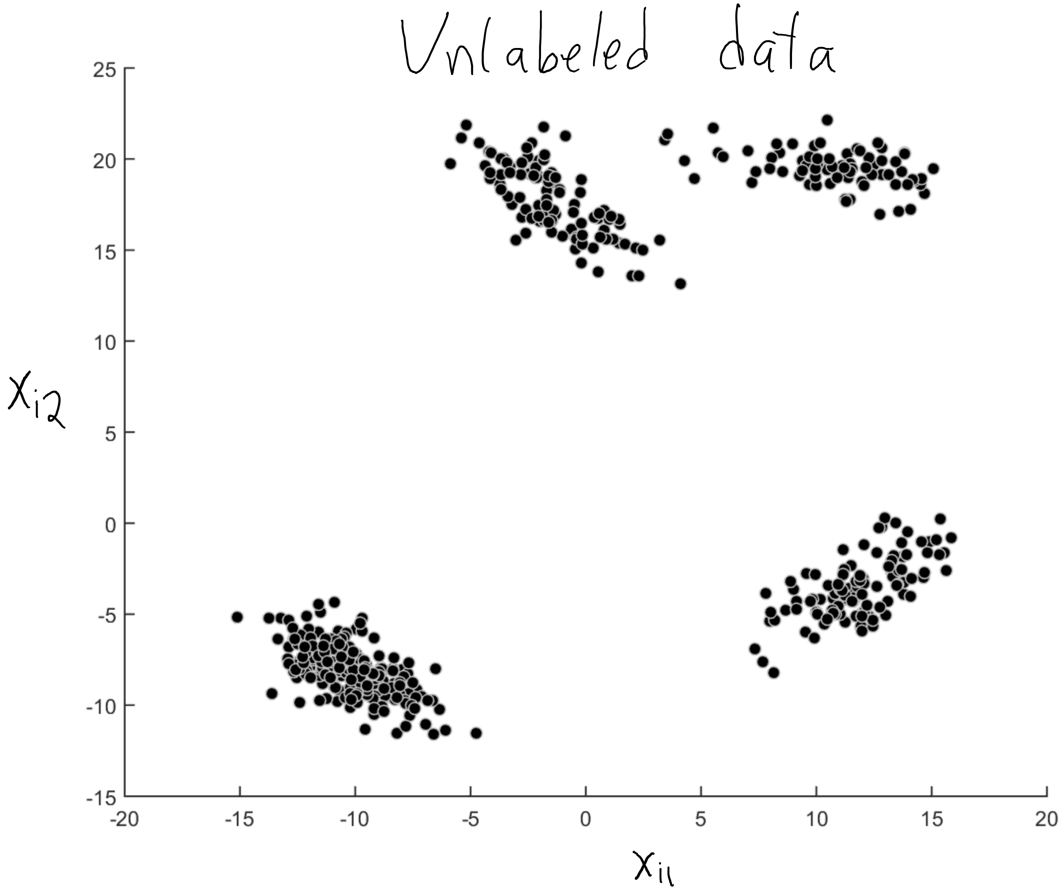
Testing

$$\hat{X} = \left[\qquad\qquad\right] \quad \hat{y} = \left[\quad\right]$$

completely new data

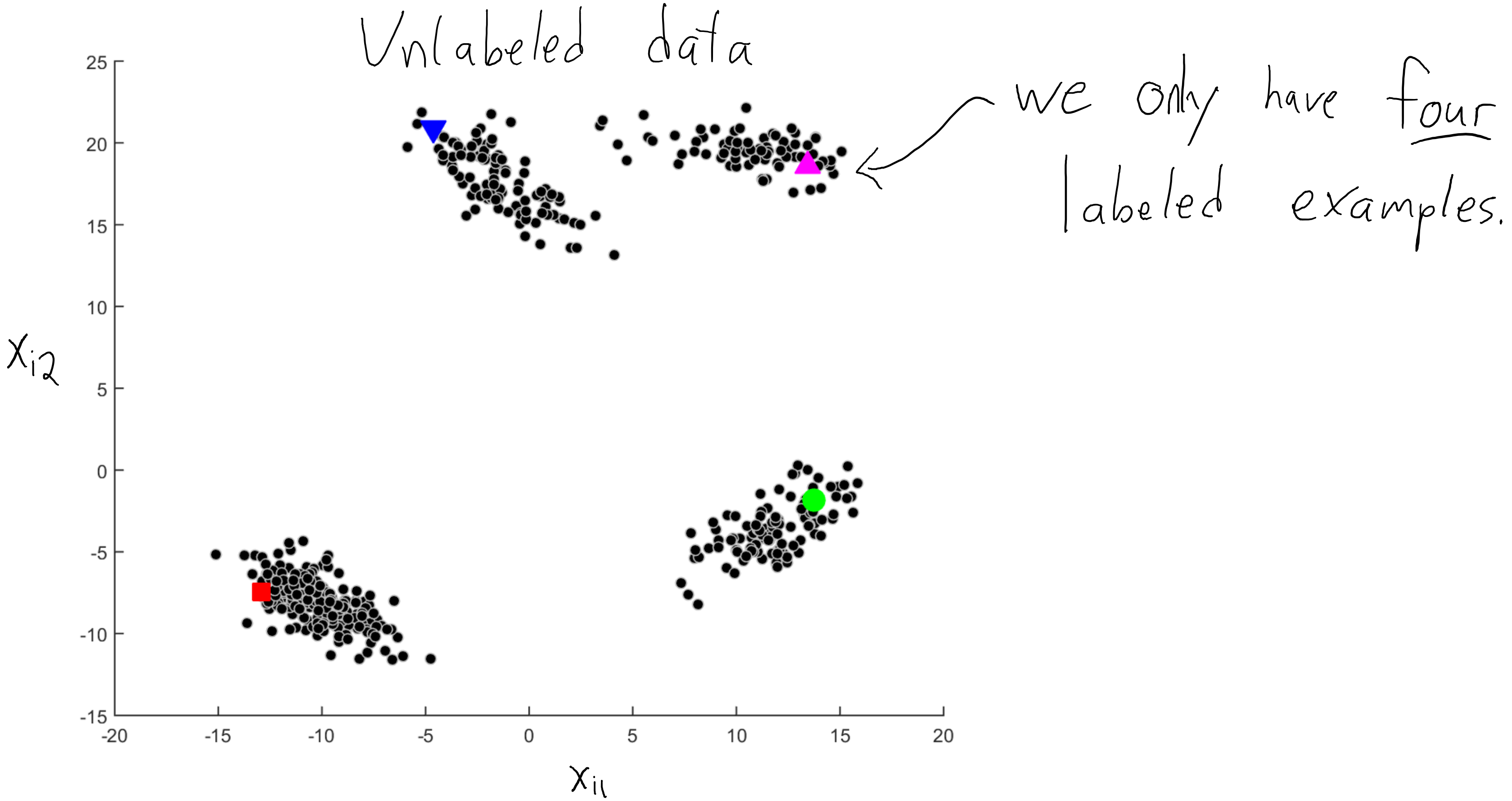
Semi-Supervised Learning

- Why should unlabeled data tell us anything about the labels?
  - Usually, we assume that: (similar features ⇔ similar labels).



Semi-Supervised Learning

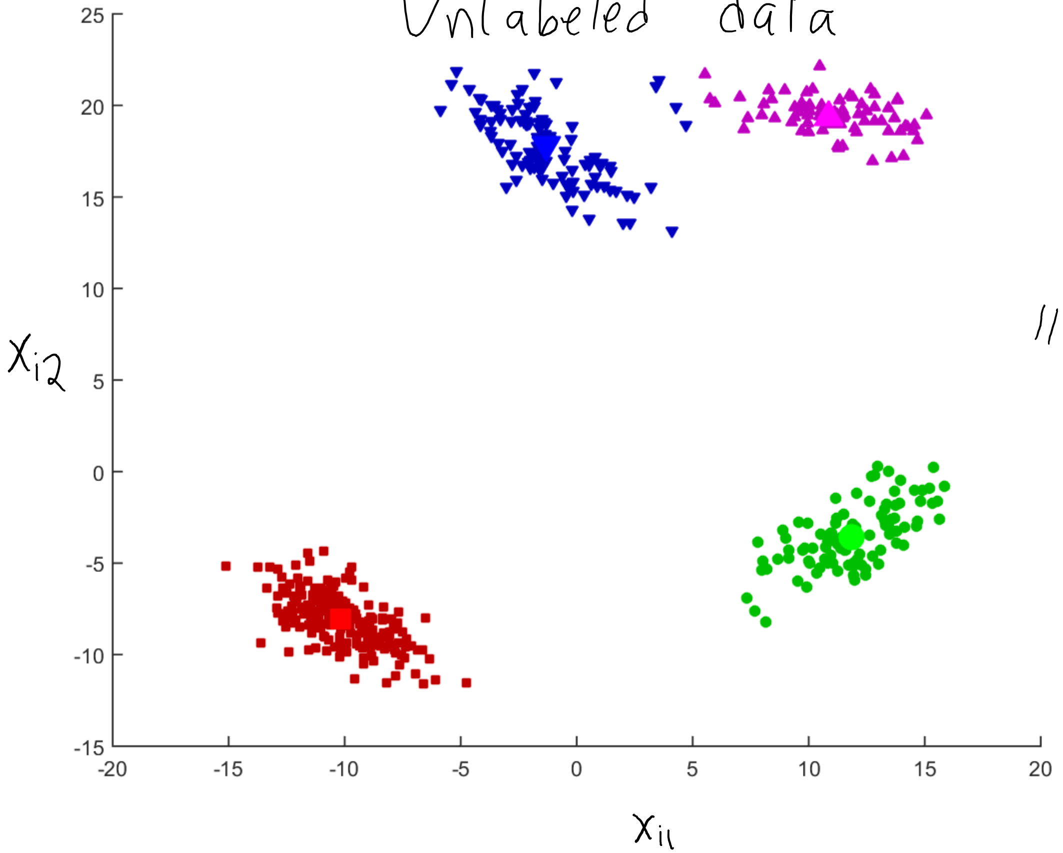
- Why should unlabeled data tell us anything about the labels?
  – Usually, we assume that: (similar features ⇔ similar labels).



Semi-Supervised Learning

- Why should unlabeled data tell us anything about the labels?
  - Usually, we assume that: (similar features ⇔ similar labels).

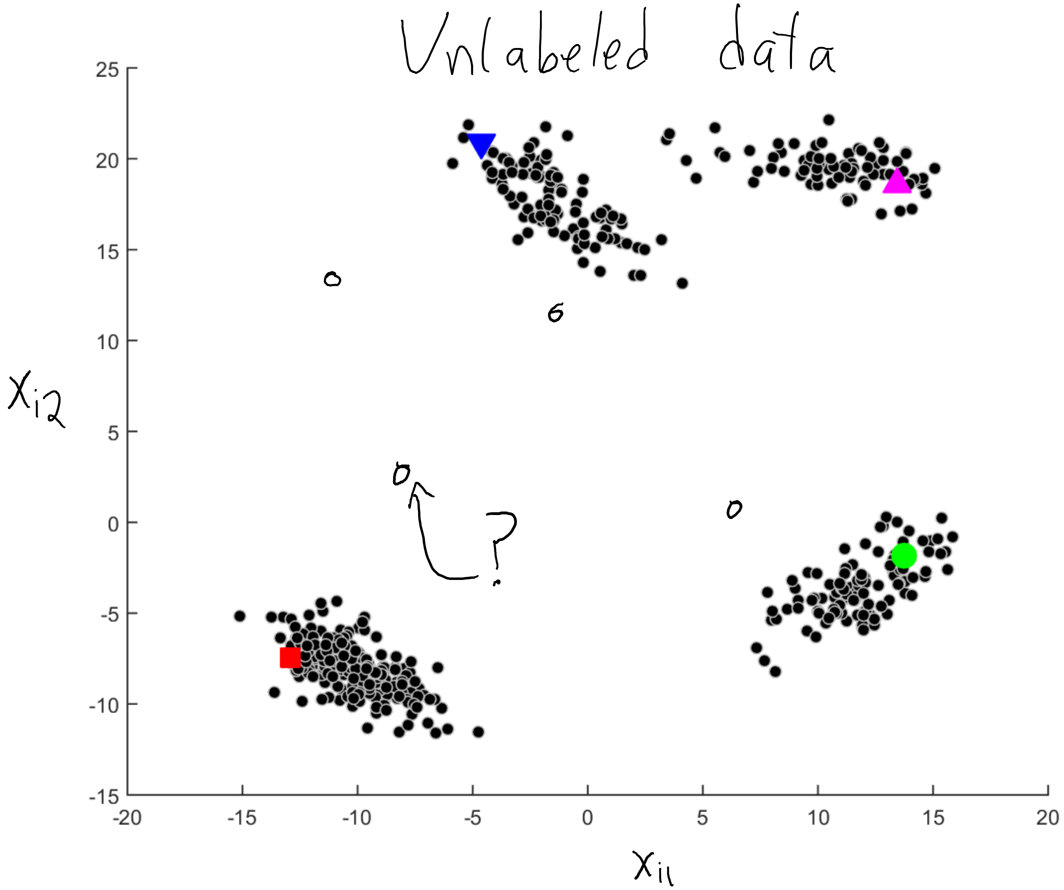
Unlabeled data



"Transductive" SSL:
- label the unlabelled examples only.

Semi-Supervised Learning

- Why should unlabeled data tell us anything about the labels?
  - Usually, we assume that: (similar features ⇔ similar labels).



"Inductive" SSL:
– build model that can label new points.

Philosophical Digression: Can we do SSL?

- Will unlabeled examples help in general?
  - No!
- Consider choosing random 'x_i' values, then computing 'y_i'.
  - Unlabeled examples collected in this way will not help.
  - By construction, distribution of 'x_i' says nothing about 'y_i'.
- Consider 'y_i' somehow influencing data we collect:
  - Now there is information about labels contained in unlabeled examples.
  - Example 1: we try to have an even number of $y_i = +1$ and $y_i = -1$.
  - Example 2: we need to choose non-random 'x_i' to correspond to a valid 'y_i'
  - We are almost always in this case.

Philosophical Digression: Can we do SSL?

- Example where SSL is not possible:
  - Try to detect food allergy by trying ‘random’ combinations of food.
    - The actual ‘random’ process isn’t important, as long it doesn’t depend on ‘y_i’.
  - Unlabeled data would be more random combinations:

$$X = \begin{bmatrix} \text{"random"} \\ \text{values} \end{bmatrix} \quad y = \begin{bmatrix} \text{labels} \\ \text{of} \\ \text{random} \\ \text{values} \end{bmatrix} \quad \tilde{X} = \begin{bmatrix} \text{more} \\ \text{"random"} \\ \text{values} \end{bmatrix}$$

- You can generate all possible unlabeled data, but it says nothing about labels.

Philosophical Digression: Can we do SSL?

- Example where SSL is possible:
  - Trying to classify images as ‘cat’ vs. ‘dog’:
  - Unlabeled data would be images of cats or dogs: not random images.
    - Unlabeled data contains information about what images of cats *and* dogs look like.
    - E.g., clusters or manifolds (or just closeness) in unlabeled images.
- Contrast this with ‘cat’ vs. ‘not cat’:
  - If we generate random images then label them, unlabeled data won’t help.
  - If we know that half our unlabeled images are cats, unlabeled could help.

https://en.wikipedia.org/wiki/Cat
https://en.wikipedia.org/wiki/Dog

SSL Approach 1: Self-Taught Learning

- Self-taught learning is similar to k-means:
  1. Fit a model based on the labeled data.
  2. Use the model to label the unlabeled data.
  3. Use estimated labels to fit model based on labeled and unlabeled data.
  4. Go back to 2.
- Obvious problem: it can reinforce errors and even diverge.
- Possible fixes:
  – Only use labels are you very confident about.
  – Regularize the loss from the unlabeled examples:

$$\operatorname*{argmin}_w \sum_{i=1}^{n} g(y_i, \hat{y}_i) + \lambda \sum_{i=1}^{t} g(\tilde{y}_i, \hat{y}_i)$$

$\tilde{y}_i$: prediction from step 2.

$\hat{y}_i$: prediction from using 'w'

Scalar λ controls how much we trust predicted labels $\tilde{y}_i$

SSL Approach 1: Self-Taught Learning

Input:
- labeled examples $\{X, y\}$
- unlabeled examples $\tilde{X}$

1. Train on $\{X, y\}$:

   model = fit(X, y)

2. Guess labels:

   $\tilde{y}$ = model.predict$(\tilde{X})$

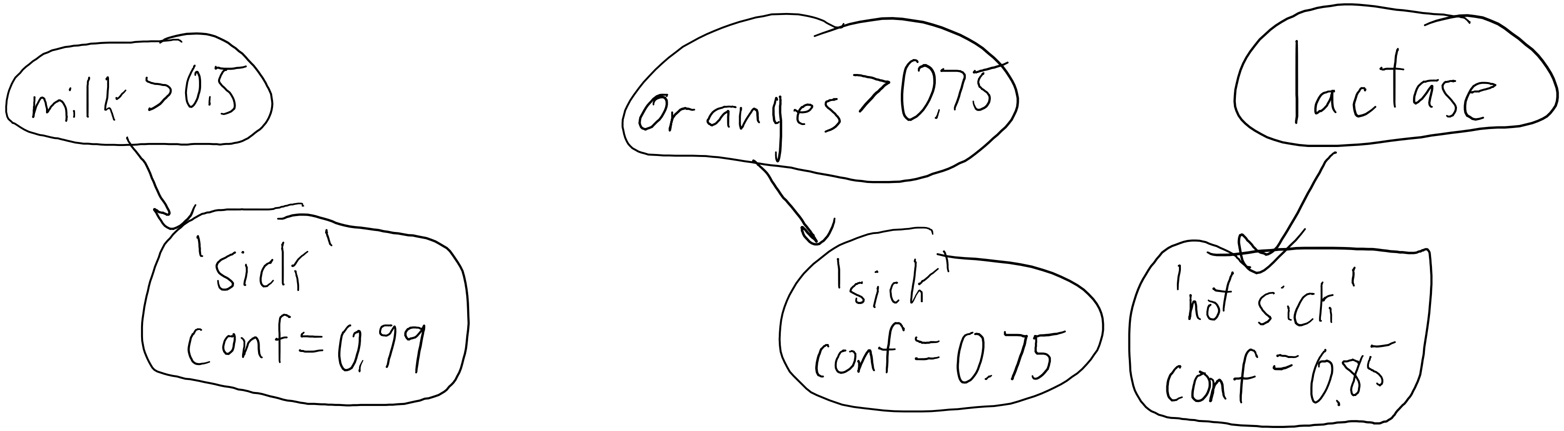
3. Train on bigger data set:

   model = fit$\left(\begin{bmatrix} X \\ \tilde{X} \end{bmatrix}, \begin{bmatrix} y \\ \tilde{y} \end{bmatrix}\right)$

repeat

Yarowsky Algorithm

- Variant of self-taught learning is Yarowsky's algorithm.
- Base classifier is a decision list:
  - List of decision rules and their confidence:



  - Use rule with the highest confidence.
    - Or leave unlabeled if nothing has high confidence.

Yarowsky Algorithm

- Variant of self-taught learning is Yarowsky's algorithm:
    1. Start with a small number of 'seed' rules:

"Word sense diambiguation" for word "sentence"
$\begin{cases} \text{If 'served' is close to 'sentence', label } -1 \text{ (conf=0.99)} \\ \text{If 'read' is close to 'sentence', label } +1 \text{ (conf=0.99)} \end{cases}$

    2. Label unlabeled examples.
    3. Add rules with highest confidence.

If previous word is 'life', label = −1 (conf = 0.986)
If 'page' is close, label = +1 (conf = 0.953)

    4. Go back to 2.

Yarowsky Algorithm

- Variant of self-taught learning is Yarowsky's algorithm:
  - Surprisingly effective in some applications.
  - Seed rules for person/place/company identification:

New York => place

Cailfornia => place

U.S. => place

Microsoft => company

I.B.M. => company

Contains(Incorporated) => company

contains(Mr.) => person

  - Finding rules with 95% confidence lead to 91% test set accuracy.

SSL Approach 2: Co-Training

- Assumes that we have 2 sets of features:
  - The feature sets should be conditionally independent given the label.
  - Both sets are sufficient to give high accuracy.
  - E.g., image features (set 1) and caption features (set 2).

Cats' whiskers are highly sensitive to touch.

- Co-training:
  1. Using labeled set, fit model 1 based on set 1, fit model 2 based on set 2.
  2. Label a random subset of unlabeled examples based on both models.
  3. Move examples where each classifier is most confident to labeled set.
  4. Go back to 1.
- Hope is that models 'teach' each other to achieve consensus.
  - Theoretically works if assumptions are satisfied.

https://en.wikipedia.org/wiki/Cat

SSL Approach 2: Co-Training

0. Split features into X_1 and X_2:

$$X = \left[\begin{array}{c:c} X_1 & X_2 \end{array} \right]$$

1. Train models on X_1 and X_2:

$$\text{model1} = \text{fit}(X_1, y) \qquad \text{model2} = \text{fit}(X_2, y)$$

2. Guess labels:

$$\tilde{y}_1 = \text{model.predict}(\tilde{X}_1) \qquad \tilde{y}_2 = \text{model.predict}(\tilde{X}_2)$$

3. Choose subset of unlabeled, add "most confident" to labeled.

Use random subset to avoid picking similar examples.

repeat

SSL Approach 3: Entropy Regularization

- Self-taught and co-training predictions may propagate errors.
- Instead of making predictions, encourage 'predictability'.
- Entropy is a measure of 'randomness' of a probability:

$$h(p) = -\sum_{i=1}^{K} p(i) \log p(i)$$

high entropy: very "random"

low entropy: very "predictable"

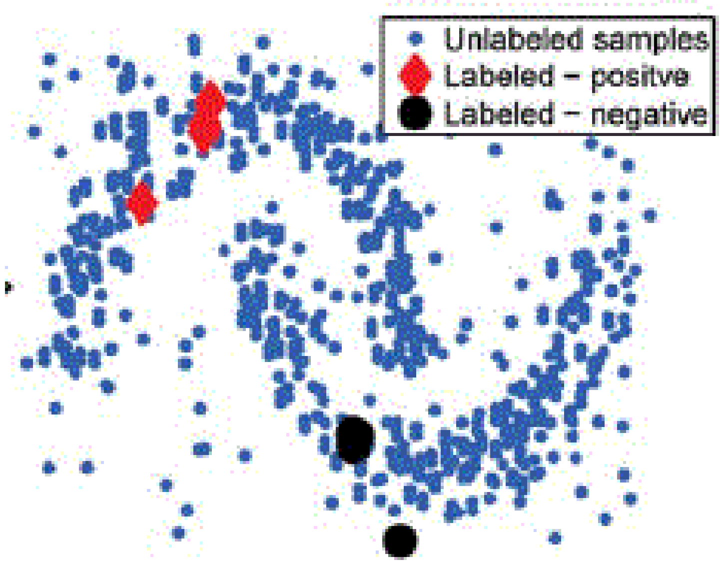
- Entropy regularization of the unlabeled examples' labels:

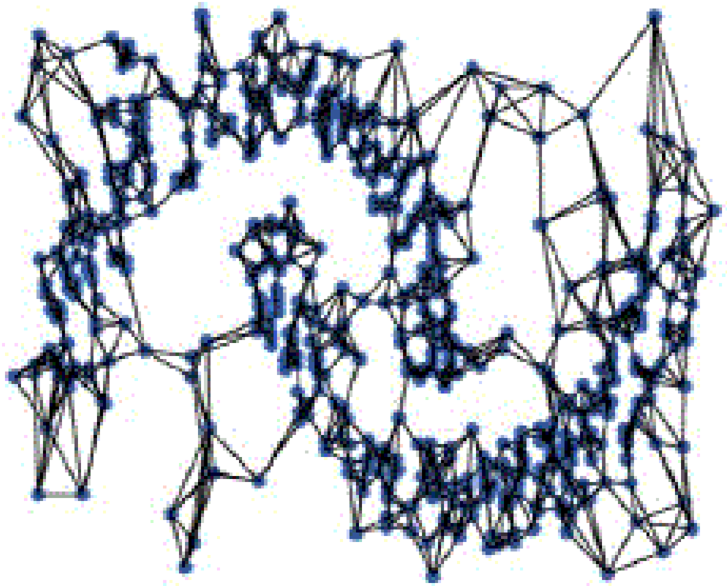
$$\underset{w}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{n} -\log\left(p(y_i \mid w, x_i)\right)}_{\text{usual loss on labeled examples}} \underbrace{- \sum_{i=1}^{t} \sum_{c=1}^{K} p(\tilde{y}_i = c \mid w, \tilde{x}_i) \log\left(p(\tilde{y}_i = c \mid w, \tilde{x}_i)\right)}_{\text{encourage "preditability" in unlabeled examples}}$$

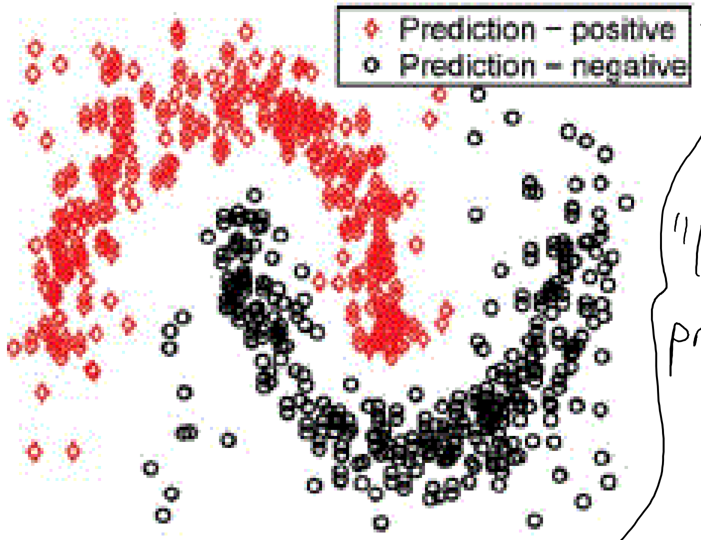
  - Related approach is transductive SVMs: avoid boundaries in dense regions.

SSL Approach 4: Graph-Based Methods

- We can only do SSL because (similar features ⇔ similar labels).
- Graph-based SSL uses this directly:
  - Define weighted graph on training examples (similar to ISOMAP):
    - For example, use k-nearest or r-close neighbours.
    - Weight is how ‘important’ it is for nodes to share label.







"label propagation"

Graph-Based SSL (Label Propagation)

- Treat unknown labels as variables, minimize cost of disagreement:

$$\underset{\tilde{y} \in \{-1,1\}^t}{\operatorname{argmin}} \; \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}(y_i - \tilde{y}_j)^2 + \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} w_{ij}(\tilde{y}_i - \tilde{y}_j)^2$$

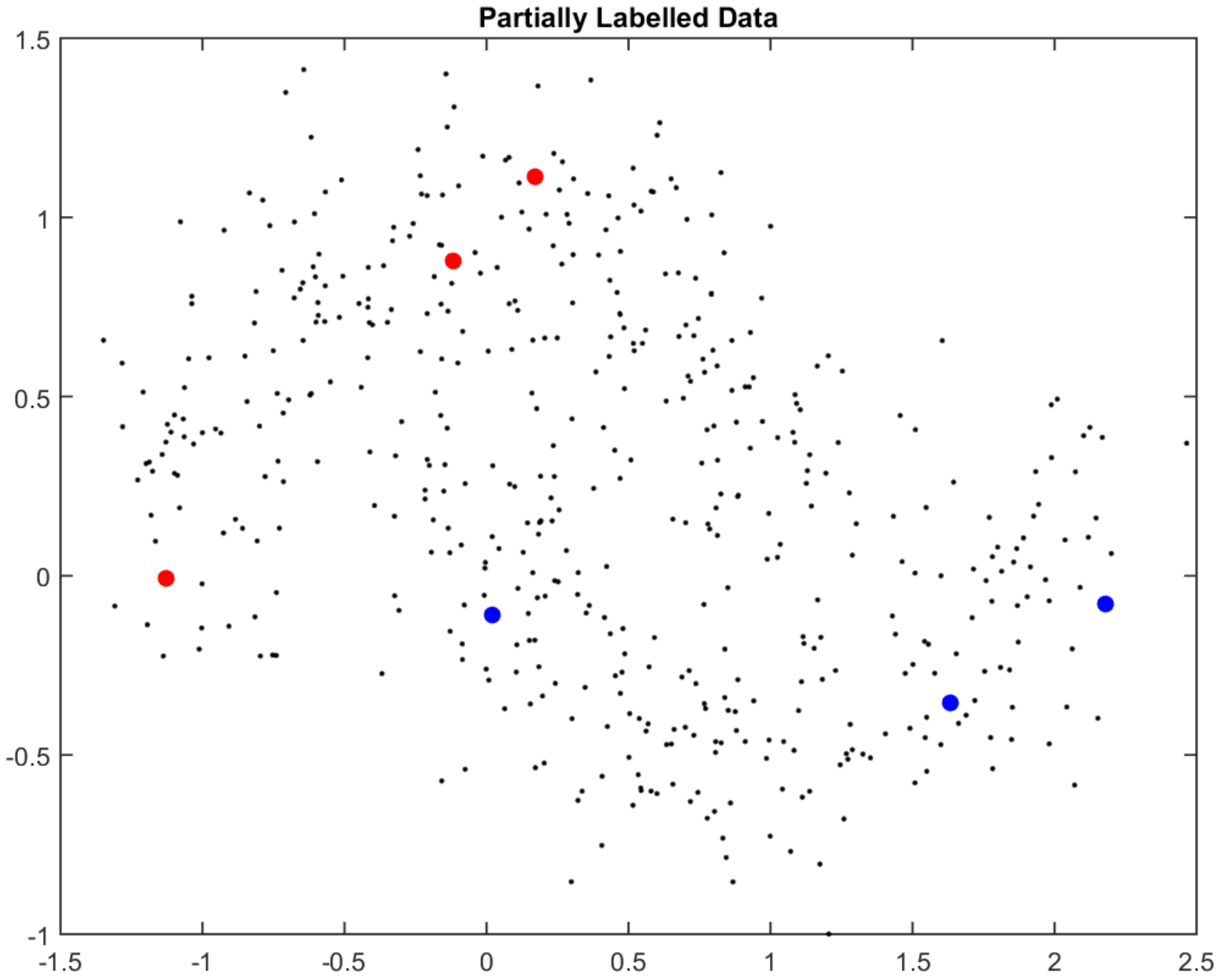
graph weight between labeled and unlabeled.

make $\tilde{y}_j$ similar to its labeled neighbours.

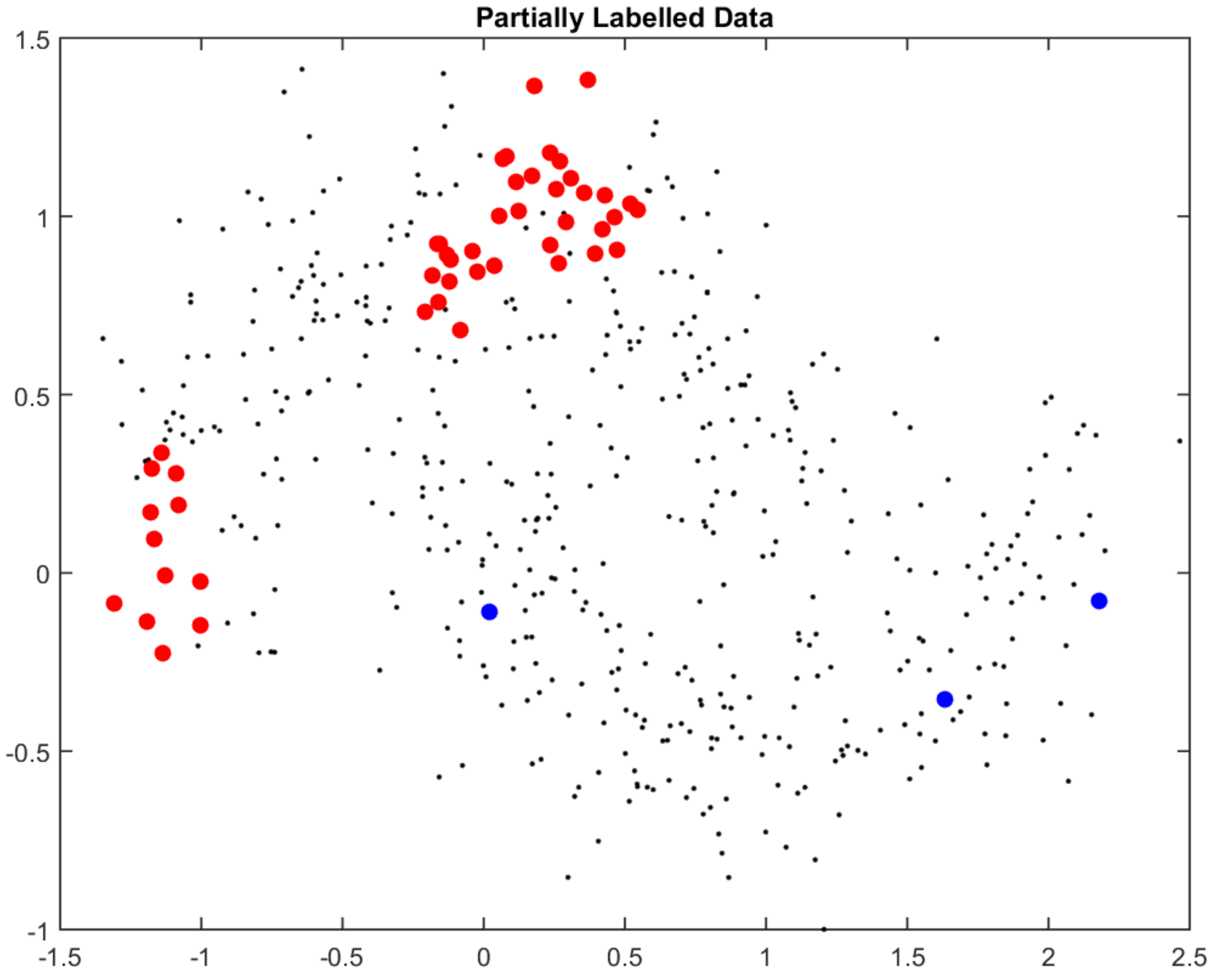
make unlabeled neighbours similar to each other.

Leads to "label propagation" through graph.

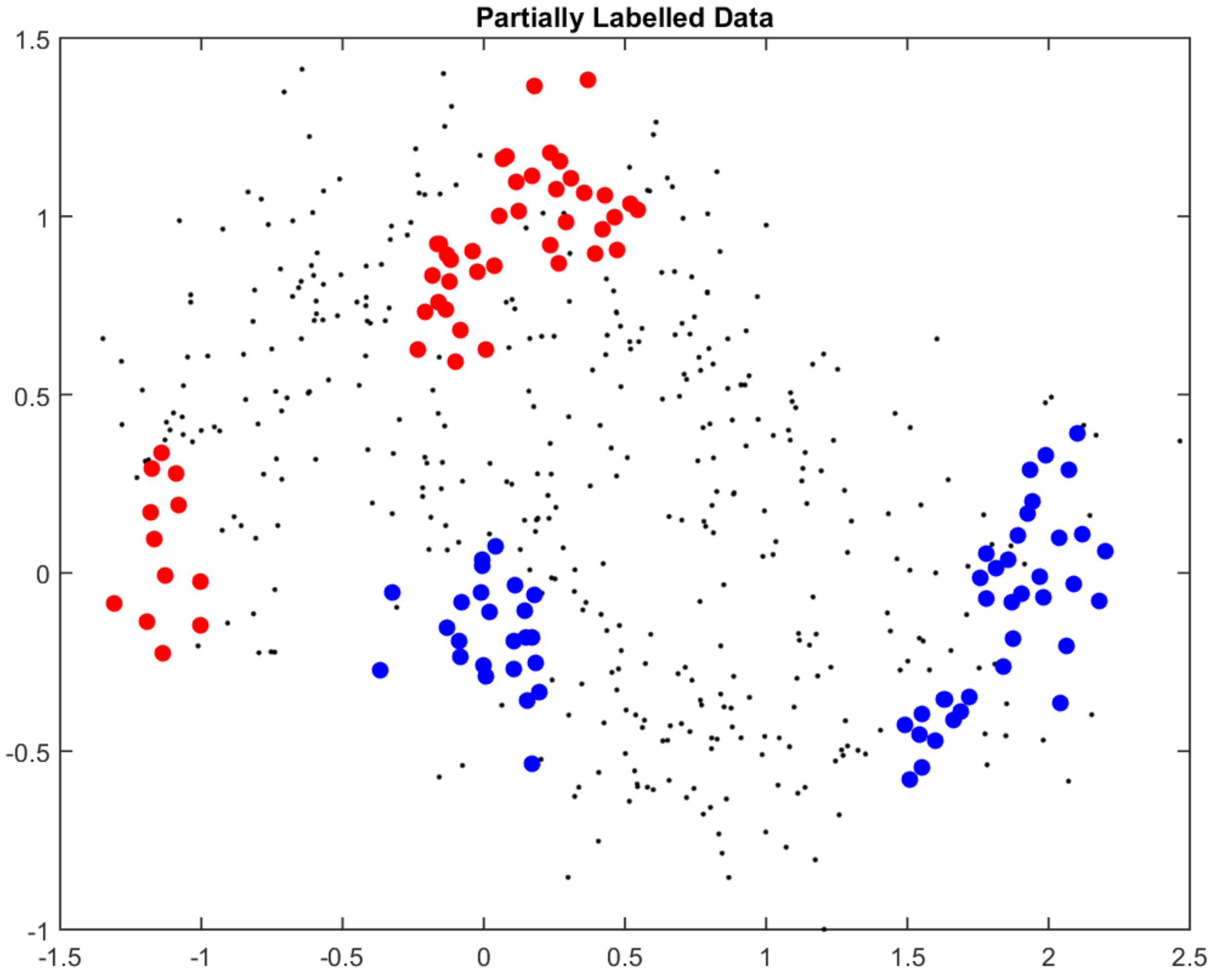
Label Propagation in Action



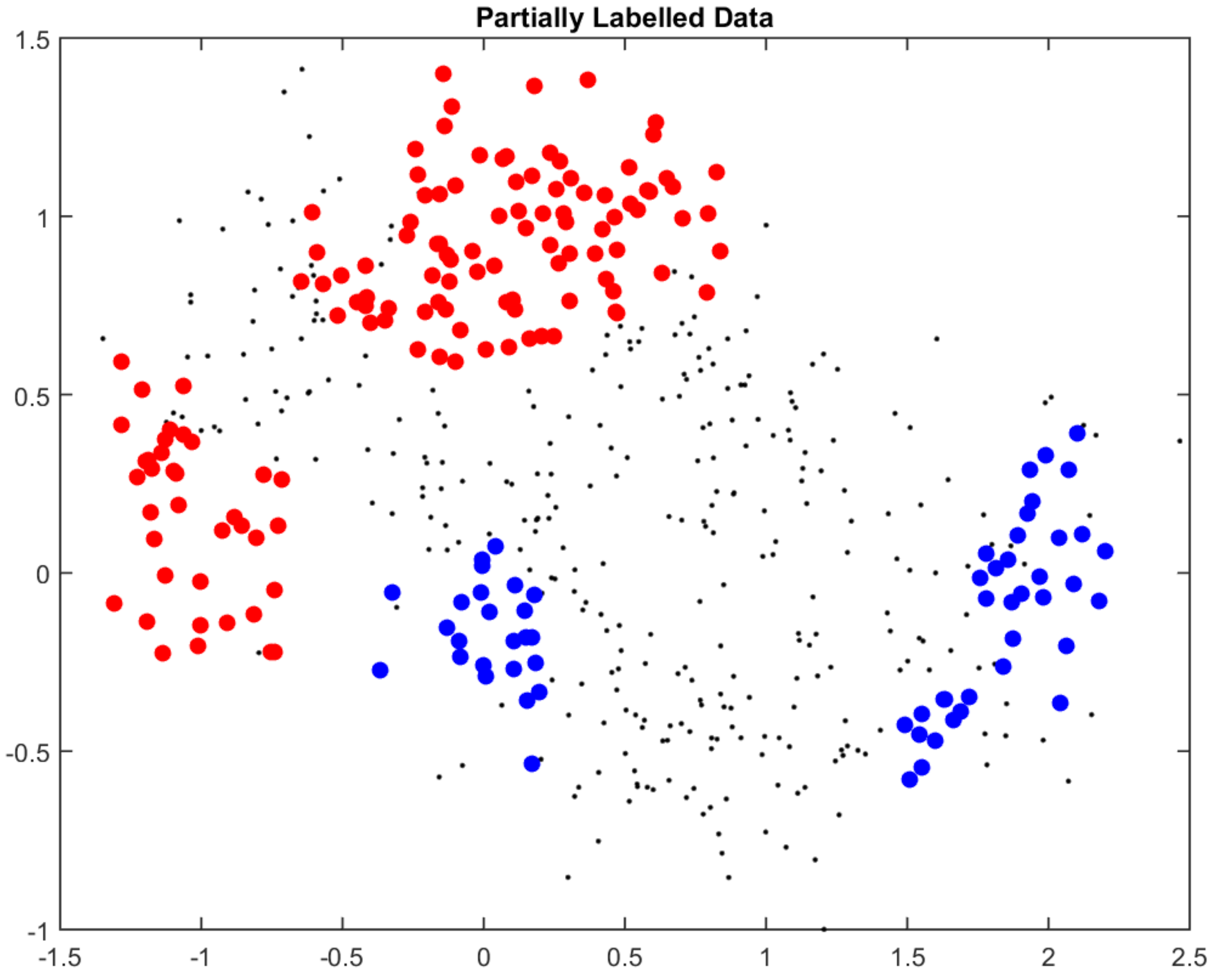
Label Propagation in Action



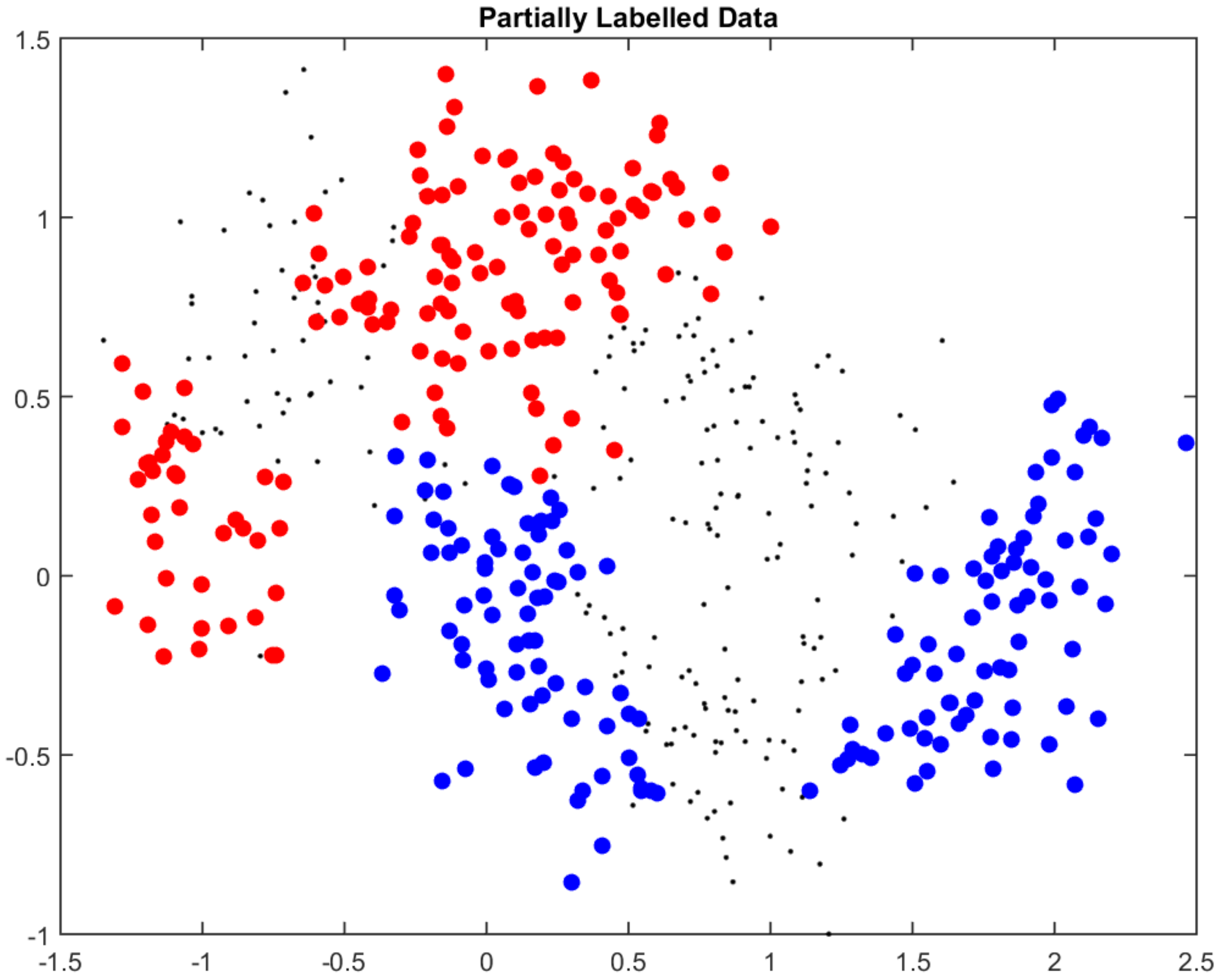
Label Propagation in Action



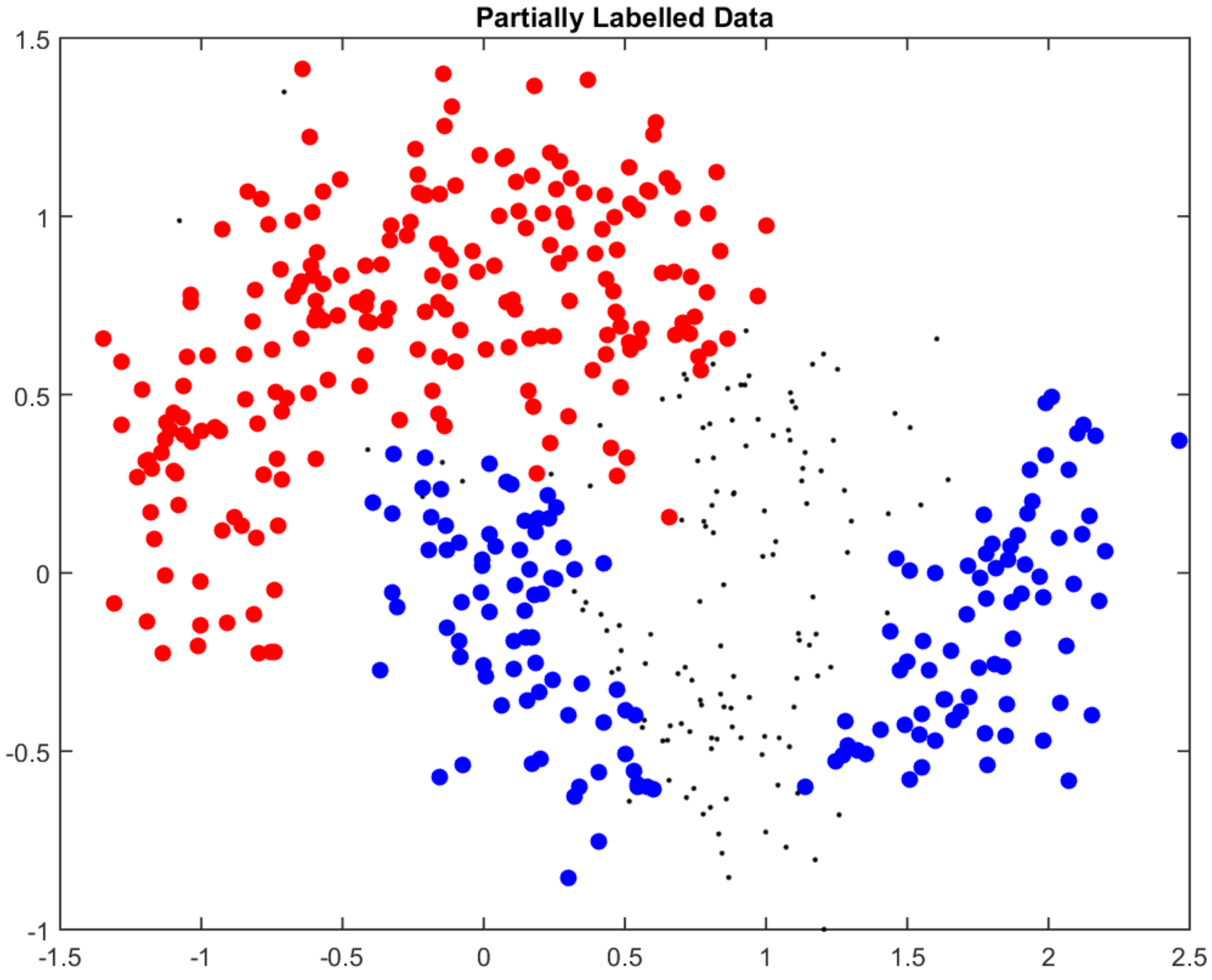
Label Propagation in Action



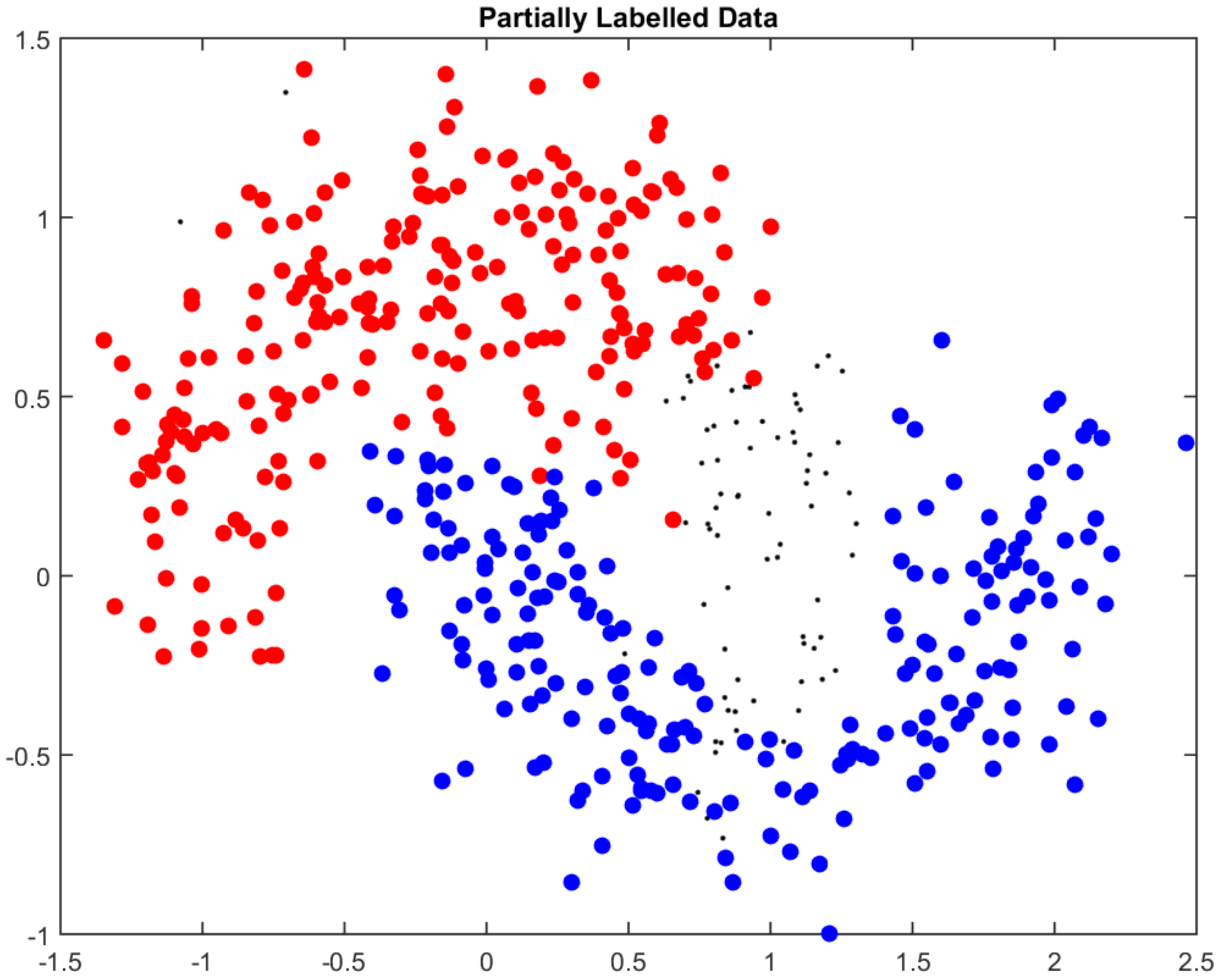
Label Propagation in Action



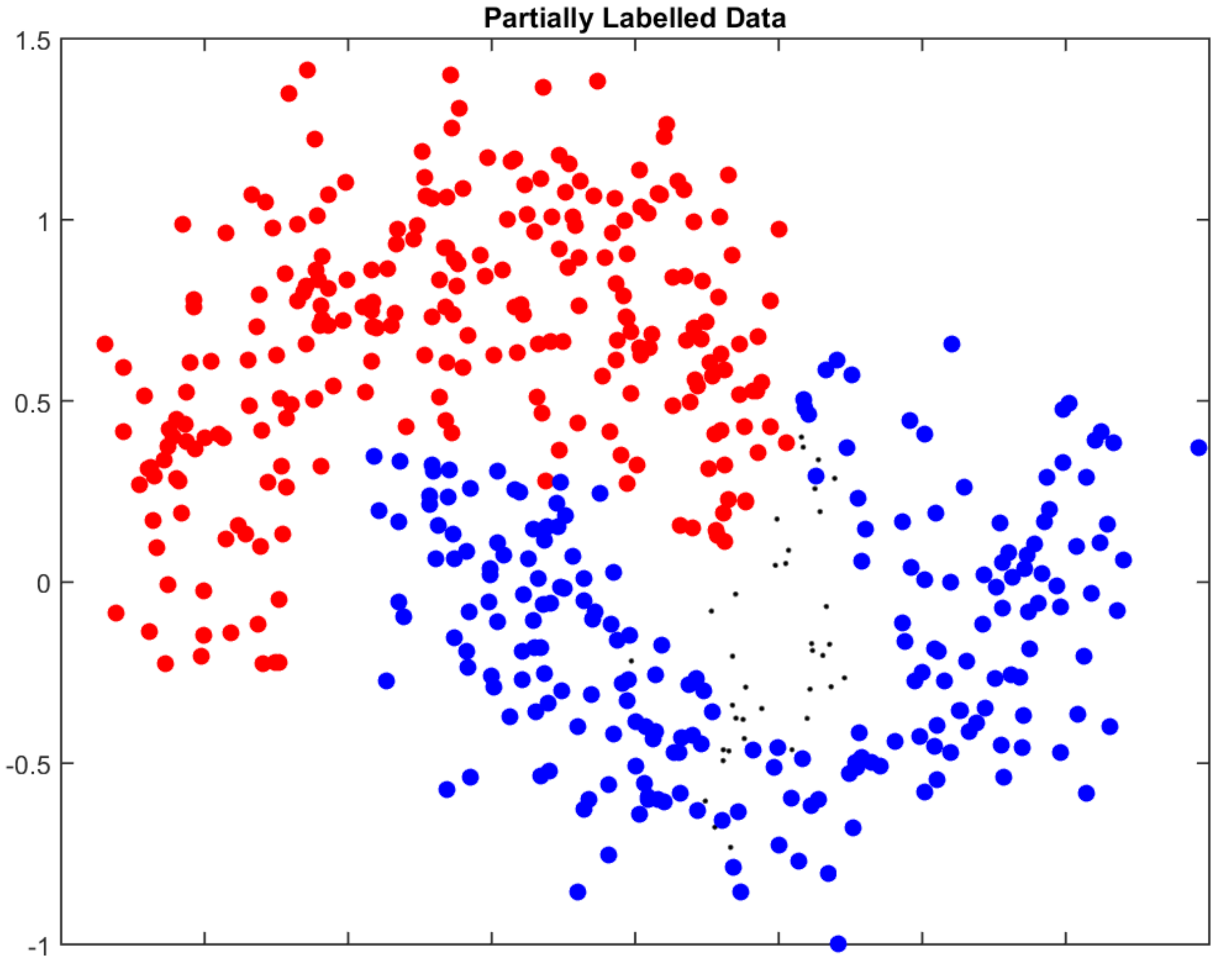
Label Propagation in Action



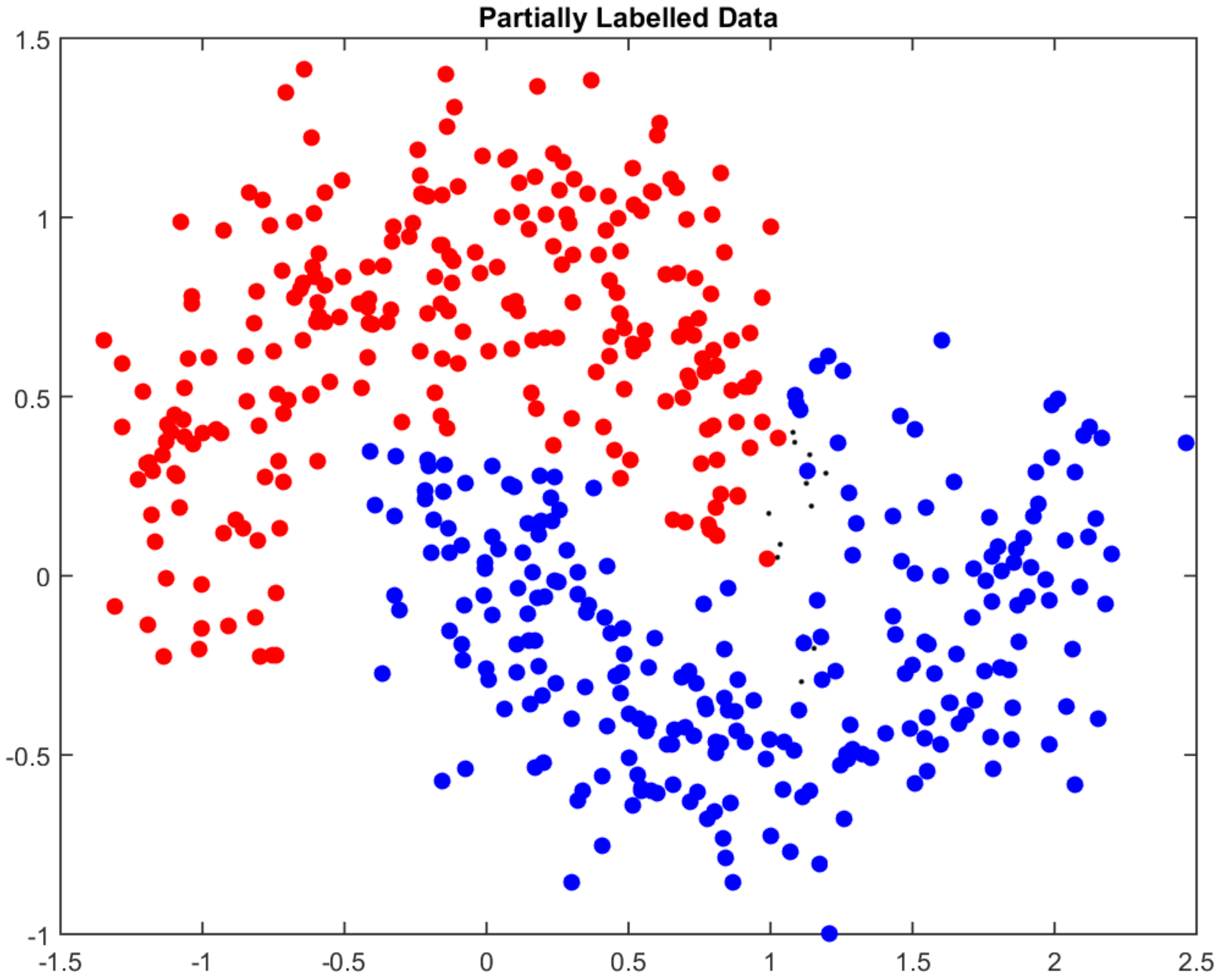
Label Propagation in Action



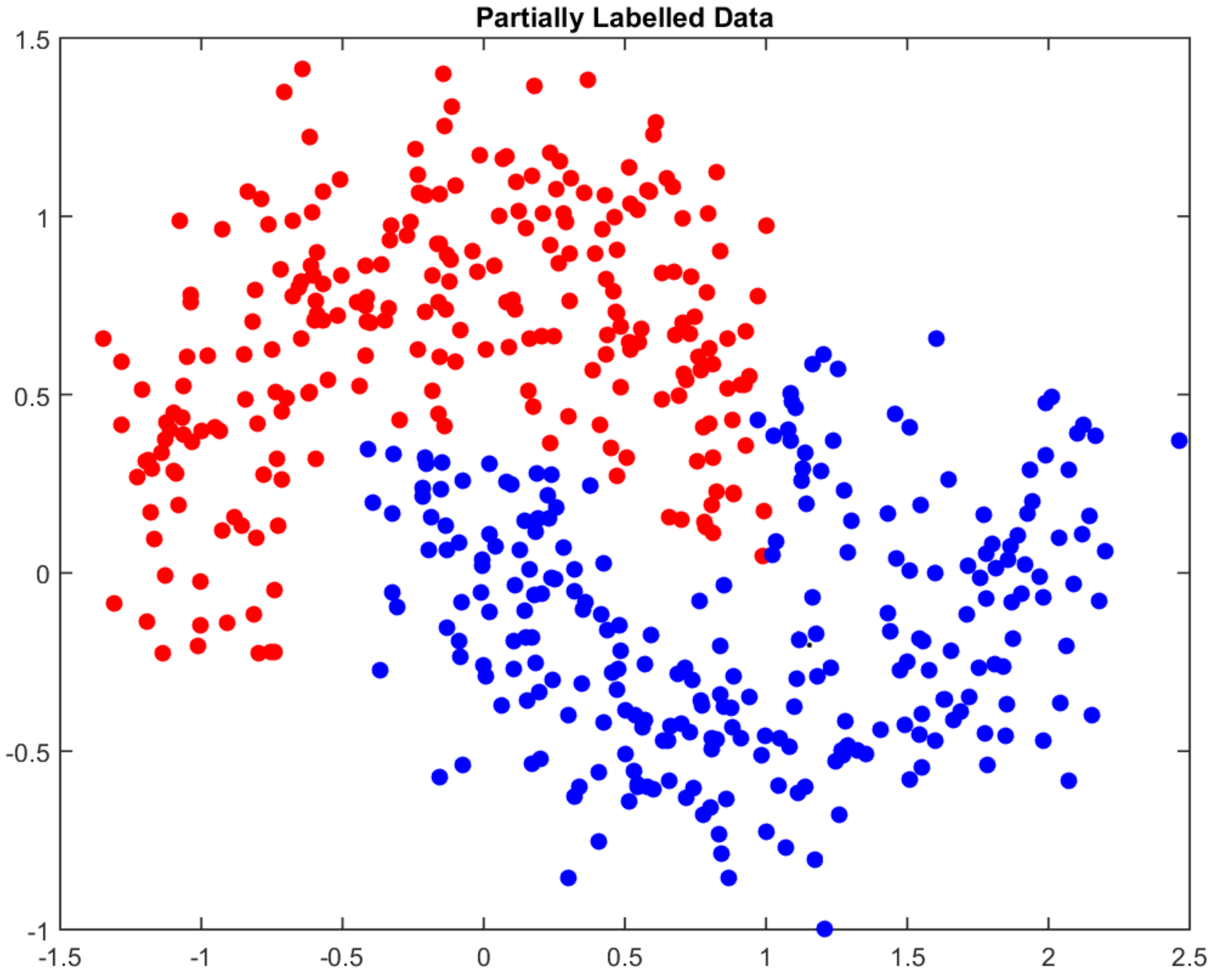
Label Propagation in Action



Label Propagation in Action



Label Propagation in Action



Example: Tagging YouTube Videos

- Comments on graph-based SSL:
  - Transductive method: only estimates the unknown labels.
  - Often surprisingly effective even if you only have a few labels.
  - Do not need features if you have the weighted graph.
  - Often add regularization: encourages ‘no decision’ if far from labels.

- Example:
  - Consider assigning ‘tags’ to YouTube videos (e.g., ‘cat’).
  - Construct a graph based on sequences of videos that people watch.
    - Give high weight if video A is often followed/preceded by video B.
  - Use label propagation to tag all videos.

Summary

- Semi-supervised learning uses unlabeled data in supervised task.
- Transductive learning only focuses on labeling this data.
- SSL may or may not help, depending on structure of data.
- Self-taught/Yarowsky/co-training alternate labeling/fitting.
- Entropy regularization encourages ‘predictable’ labels.
- Graph-based SSL propagates labels in graph (no features needed).

- Next time: ranking and the original Google algorithm.