CPSC 340: Machine Learning and Data Mining

Semi-Supervised Learning

Fall 2016

Admin

- Assignment 5:
  - 3 late days to hand in Friday.
- Assignment 6:
  - Due Friday, 1 late day to hand in next Monday, etc.
- Final:
  - December 12 (8:30am – HEBB 100)
  - Covers Assignments 1-6.
  - List of topics posted.
  - Final from last year will be posted Friday.
  - Closed-book, cheat sheet: 4-pages each double-sided.

Last Time: Ranking

- In ranking, goal is to output ordering of objects.
- We discussed supervised ranking:
  - Given item relevance, formulate as regression or ordinal regression.

$$f(w) = \sum_{i=1}^{n} -\log p(y_i \mid x_i, w) + \sum_{j=1}^{d} -\log p(w_j \mid \lambda)$$

  - Given pairwise preferences, define loss by probability ratios.

$$f(w) = \sum_{(i,j) \in R} \max\{0, \underbrace{1 - \log p(y_i \mid x_i, w) + \log p(y_j \mid x_j, w)}_{\text{From } \frac{p(y_i \mid x_i, w)}{p(y_j \mid x_j, w)} \geq e}\} + \sum_{j=1}^{d} -\log p(w_j \mid \lambda)$$

Want $y_i > y_j$

Last Time: PageRank and Markov Chains

- We discussed Markov chains for analysing sequences:
  - Initial distribution over a set of states.
  - Transition probability between each combination of states.
- Typical operations we can perform in Markov chains:
  - Generate sample sequences.
  - Compute marginal probabilities $p(x_t = s)$.
  - Compute stationary distributions.
- We discussed PageRank algorithm for ranking nodes in a graph:
  - Stationary distribution of random walk through webpages:
    - With probability α, go to a random webpage.
    - With probability $1- \alpha$, follow a random link.

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:

$$X = \Big[\qquad\Big]_{n \times d} \qquad y = \Big[\;\Big]_{n \times 1} \qquad \tilde{X} = \Bigg[\qquad\quad\Bigg]_{t \times d}$$

- This arises a lot:
  - Usually getting unlabeled data is easy but 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 **given** unlabeled examples.

$$X = \underset{n \times d}{\begin{bmatrix} & & & \\ & & & \end{bmatrix}} \quad y = \underset{n \times 1}{\begin{bmatrix} \\ \end{bmatrix}} \quad \bar{X} = \underset{t \times d}{\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix}} \quad \bar{y} = \underset{t \times 1}{\begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}}$$

Transductive vs. Inductive SSL

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

Training

$X = [\quad]_{n \times d}$ $\quad y = [\quad]_{n \times 1}$ $\quad \bar{X} = [\quad]_{t \times d}$

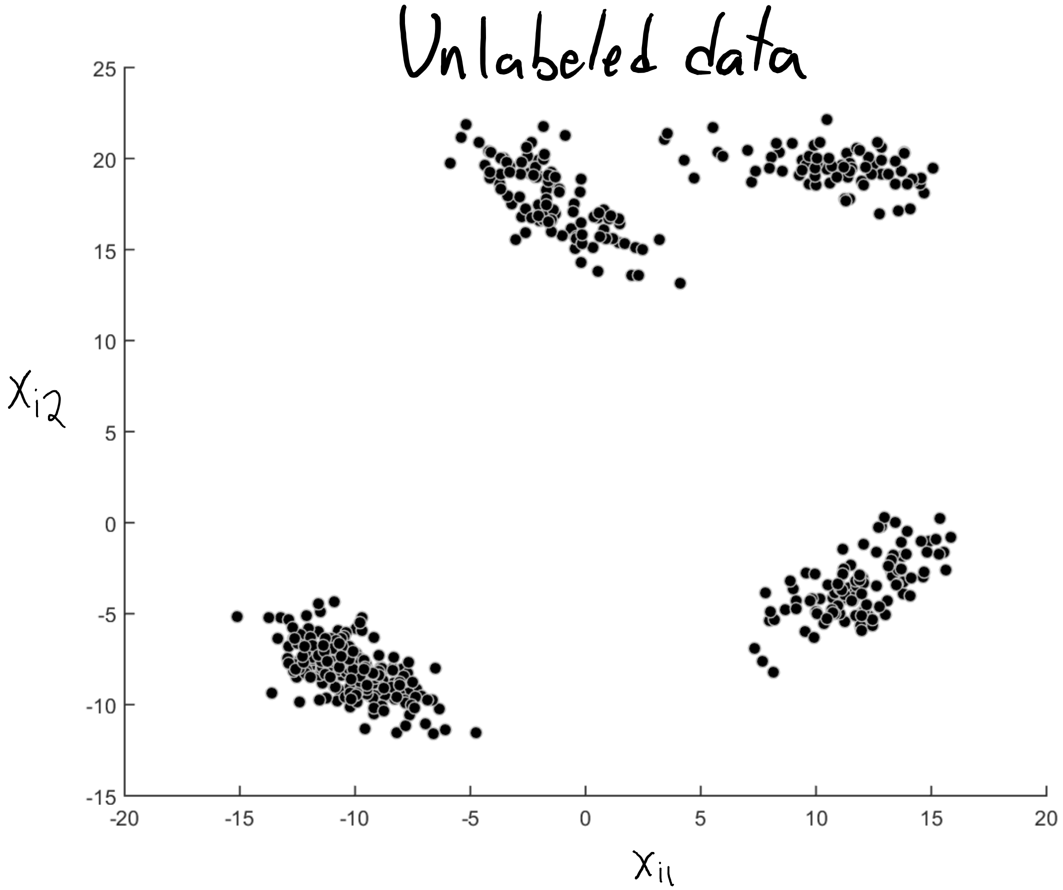
Testing

$\tilde{X} = [\quad]$ $\quad \tilde{y} = [\quad]$

completely new data

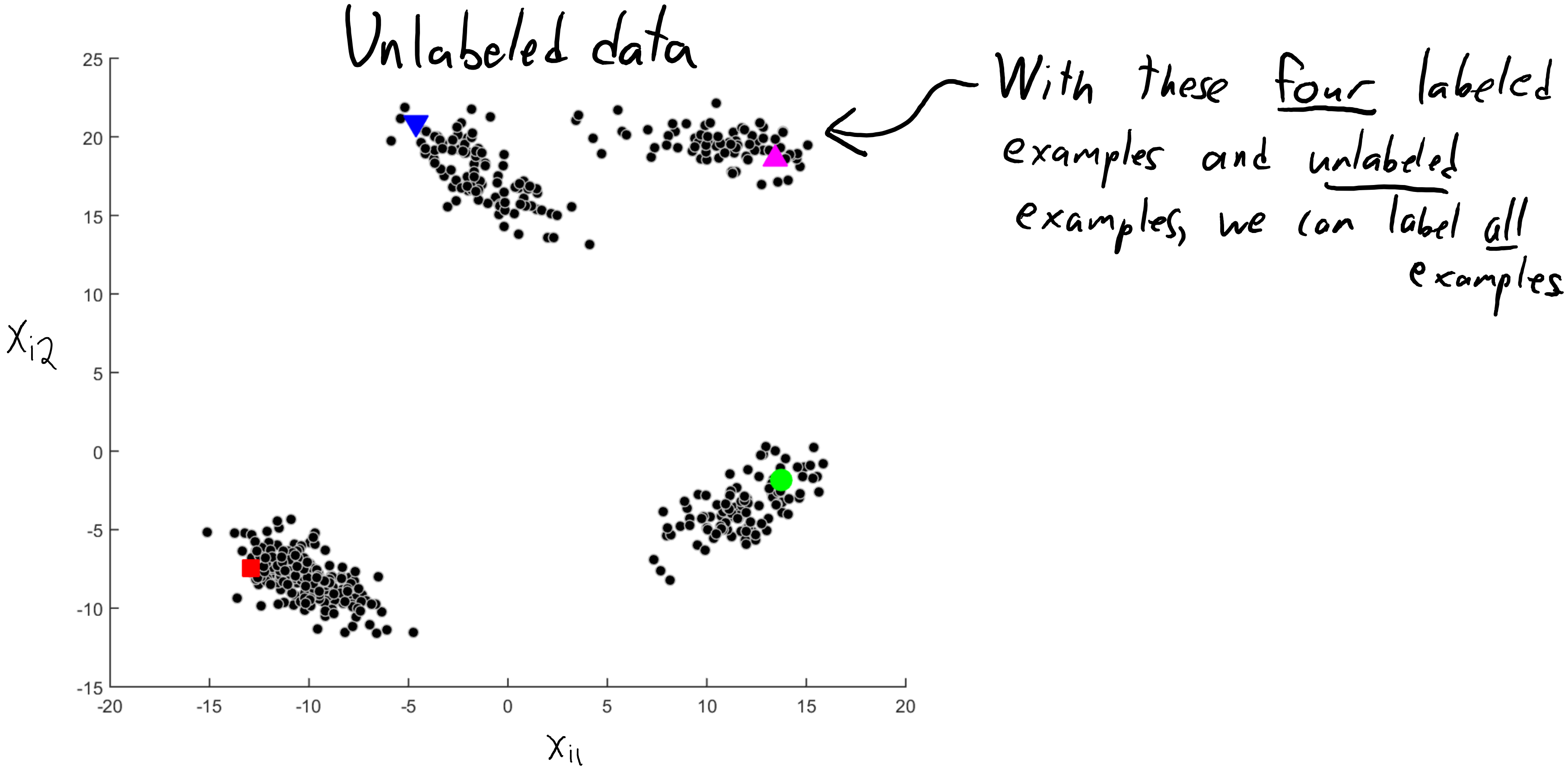
Semi-Supervised Learning

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



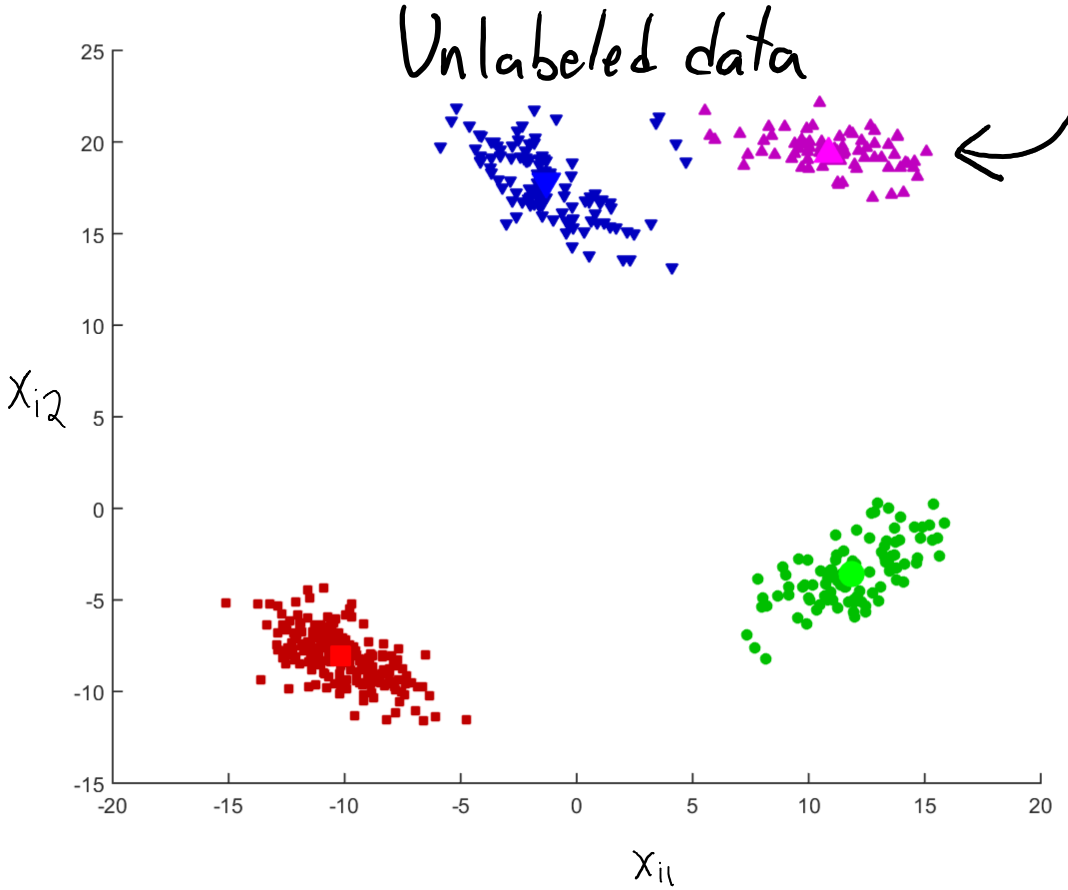
Semi-Supervised Learning

- 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).

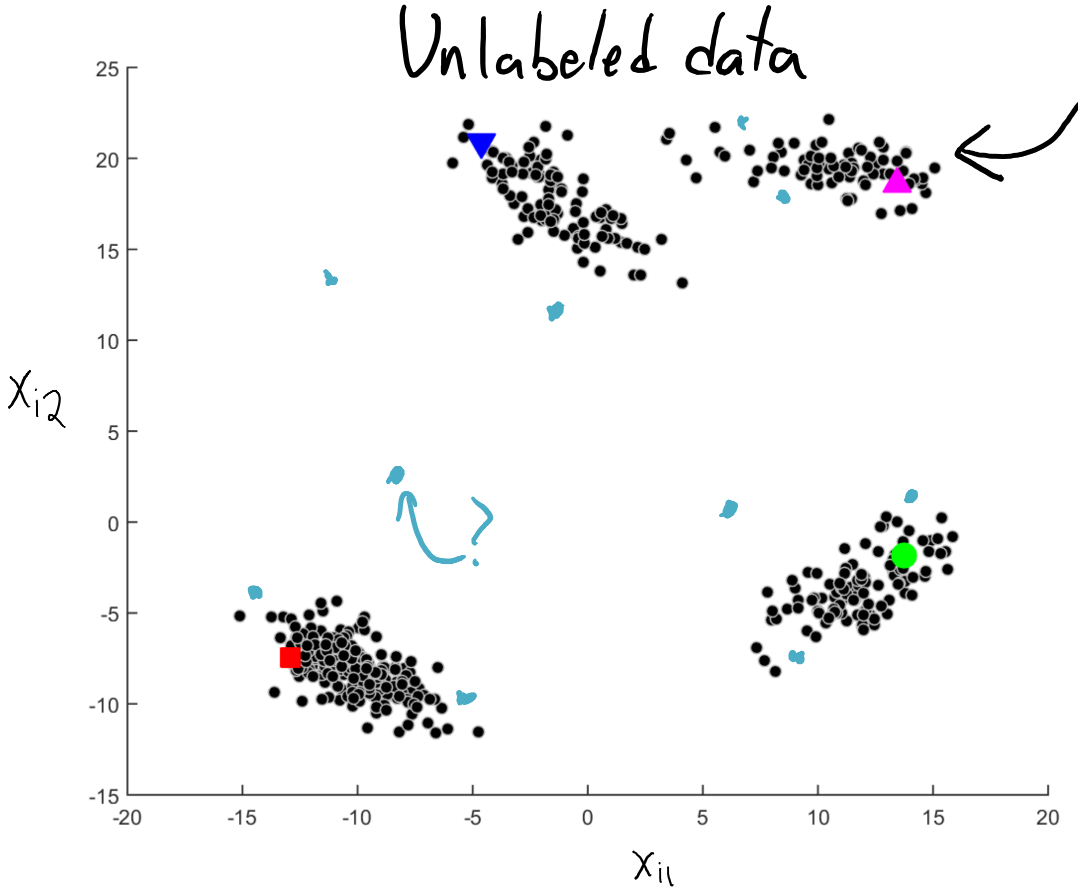


With these four labeled examples and unlabeled examples, we can label all examples

"Transductive" SSL: label the given unlabeled examples

Semi-Supervised Learning

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



With these four labeled examples and unlabeled examples, we can label all examples

"Inductive" SSL:
build a model that can label new examples.

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'.

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:

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

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

Philosophical Digression: Can we do SSL?

- When can unlabeled examples help?

- 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 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 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:

prediction from step 2

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|\tilde{X}w - \hat{y}\|^2$$

λ controls how much we trust guesses on unlabeled data

SSL Approach 1: Self-Taught Learning

Input:
- Labeled examples $\{X, y\}$
- Unlabeled examples $\tilde{X}$

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

$$\text{model} = \text{fit}(X, y)$$

2. Guess labels:

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

3. Train on bigger data set:

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

repeat

Popular variants:
1. "Expectation maximization"
2. "Yarowsky" algorithm (language)

SSL Approach 2: Co-Training

- Assumes that we have 2 sets of features:
  - Both sets are sufficient to give high accuracy.
  - The sets are conditionally independent given the label.
  - 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 above 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:

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

repeat

2. Guess labels of unlabelled examples:

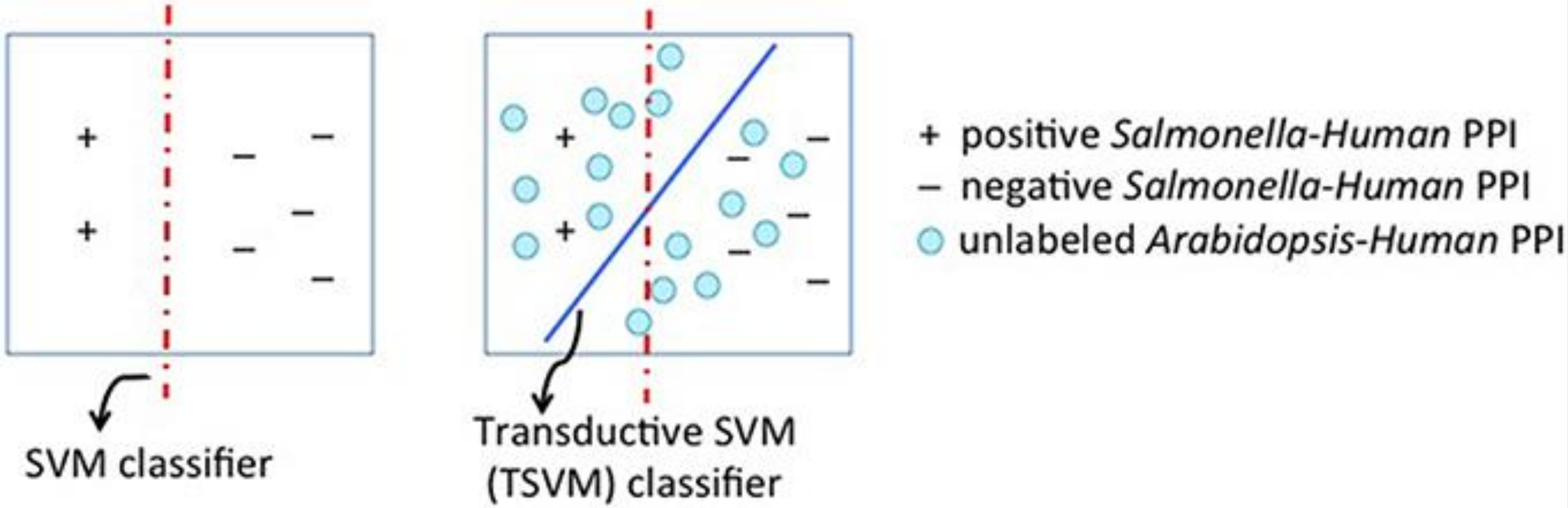
$$\hat{y}_1 = \text{model.predict}(\tilde{X}_1) \qquad \hat{y}_2 = \text{model.predict}(\tilde{X}_2)$$

Use random subset to avoid picking similar examples

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

SSL Approach 3: Entropy Regularization

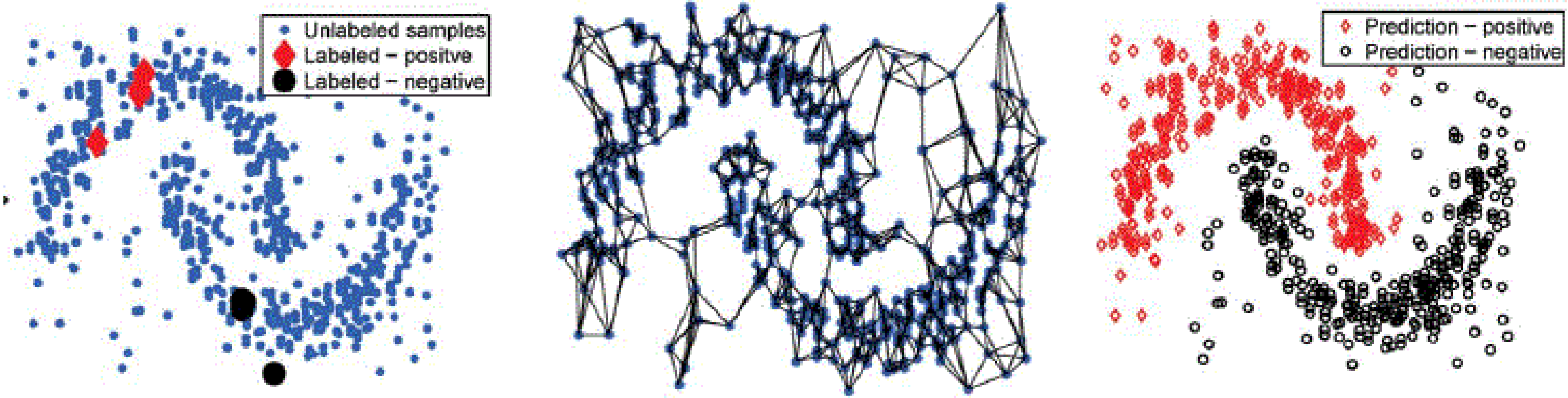
- Self-taught and co-training predictions may propagate errors.
- Instead of making predictions, encourage “predictability”:
  - Entropy regularization: penalize “randomness” of labels on unlabeled.
  - Transductive SVMs: avoid decision boundaries in dense regions.



http://journal.frontiersin.org/article/10.3389/fmicb.2015.00036/full

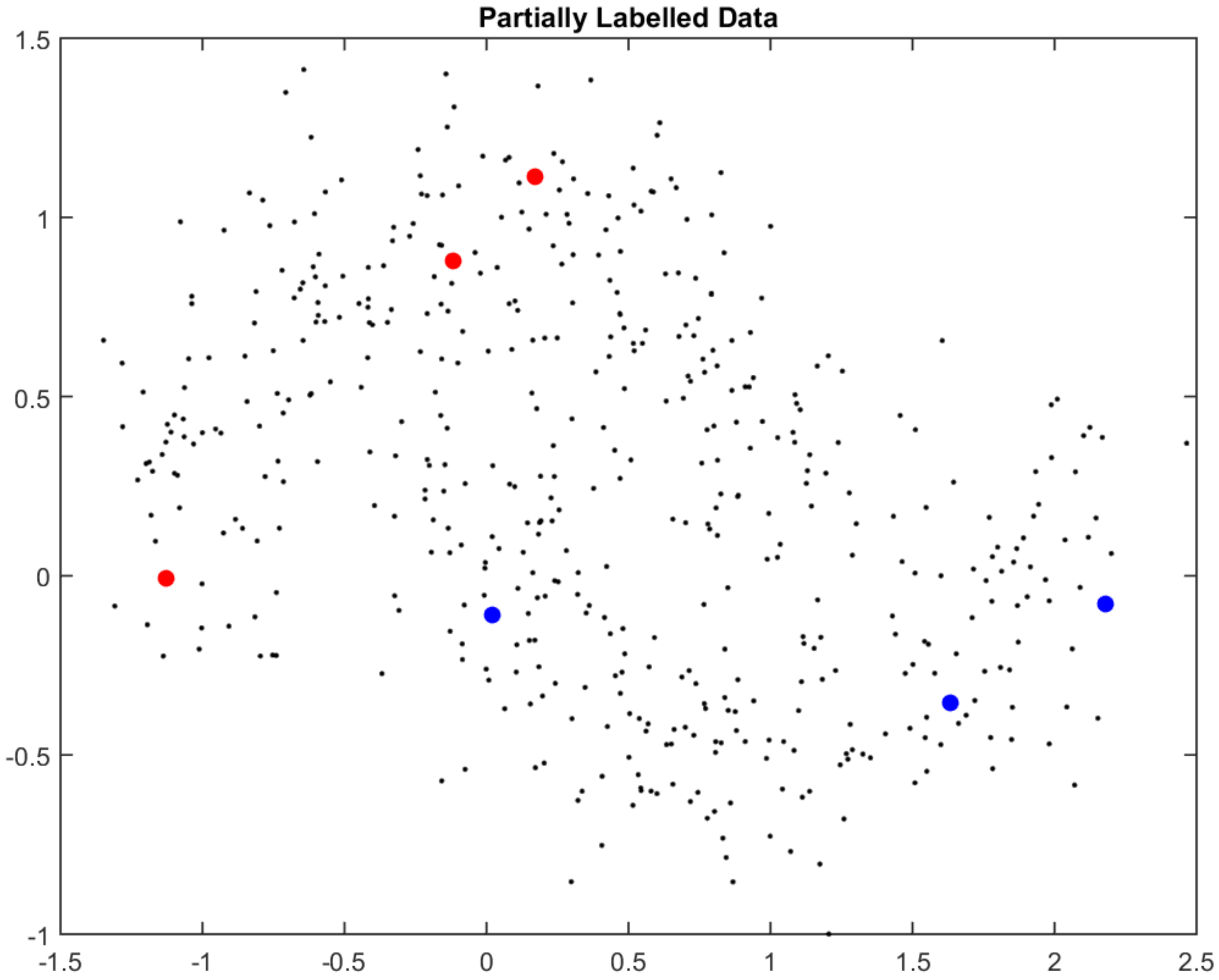
Graph-Based Methods (Label Propagation)

- We can only do SSL because (similar features ⇔ similar labels).
- Graph-based SSL uses this directly.
  - Define weighted graph on training examples:
    - For example, use KNN graph or points within radius ‘ε’.
    - Weight is how ‘important’ it is for nodes to share label.

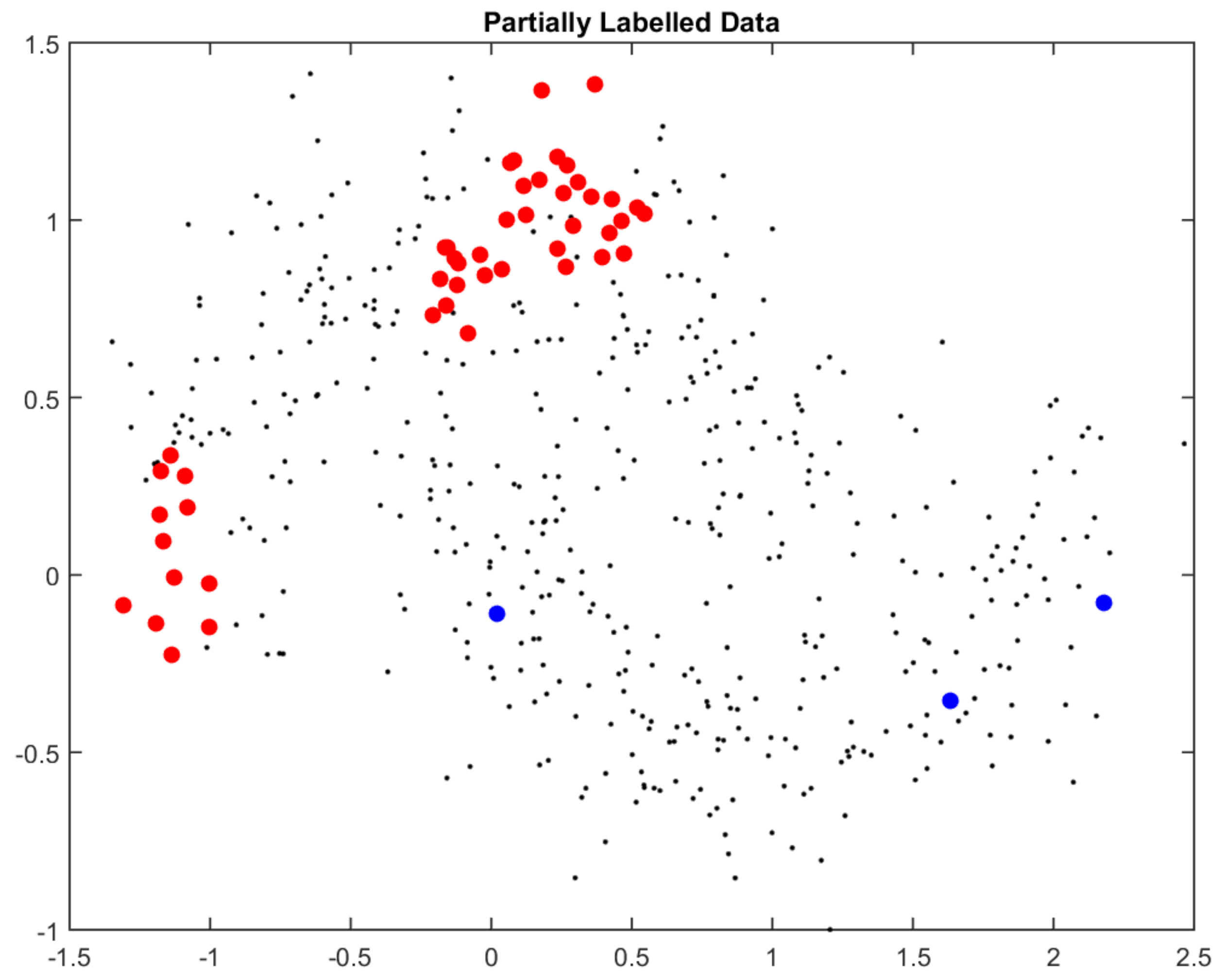


http://www.ee.columbia.edu/ln/dvmm/pubs/publications.html

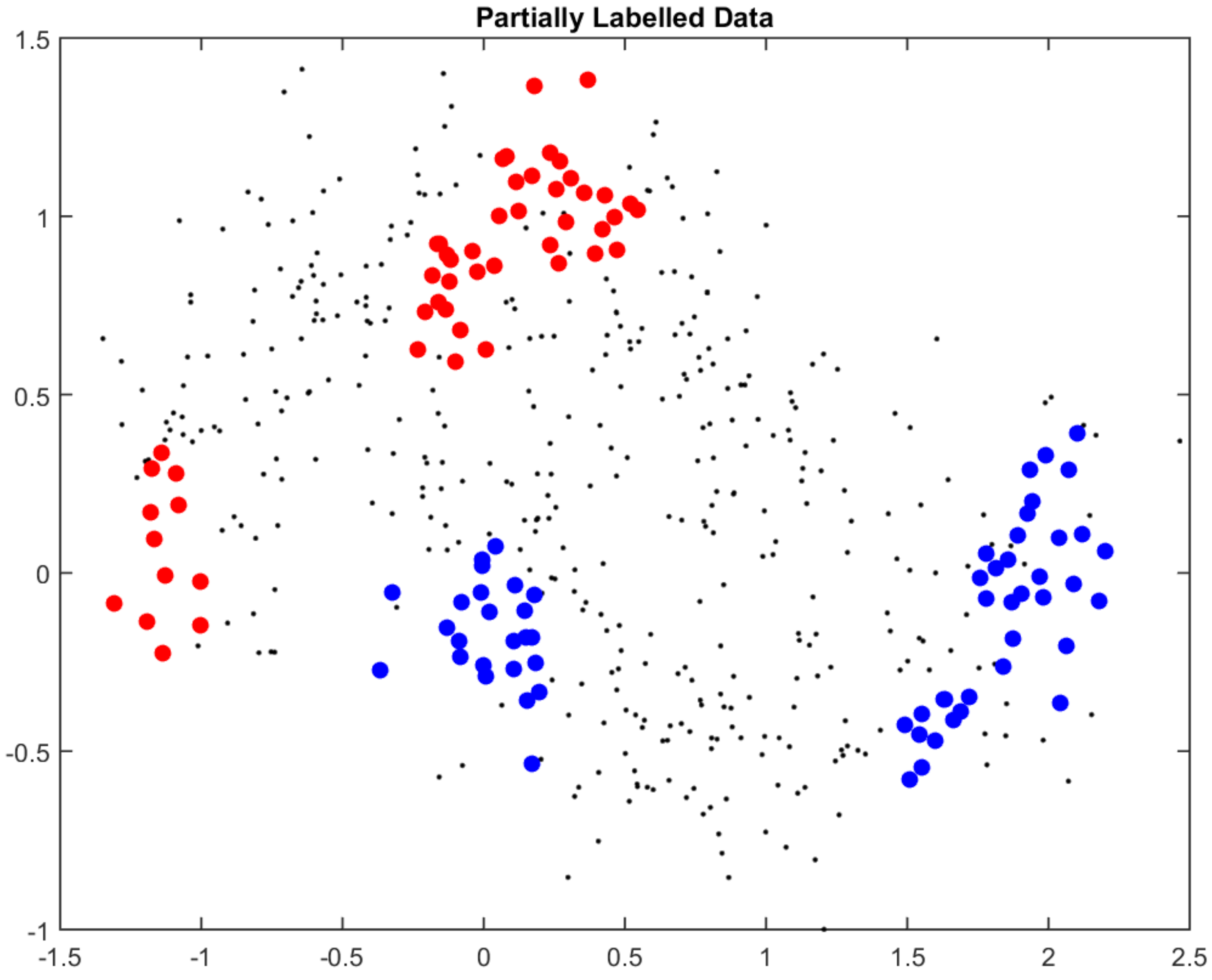
Label Propagation in Action



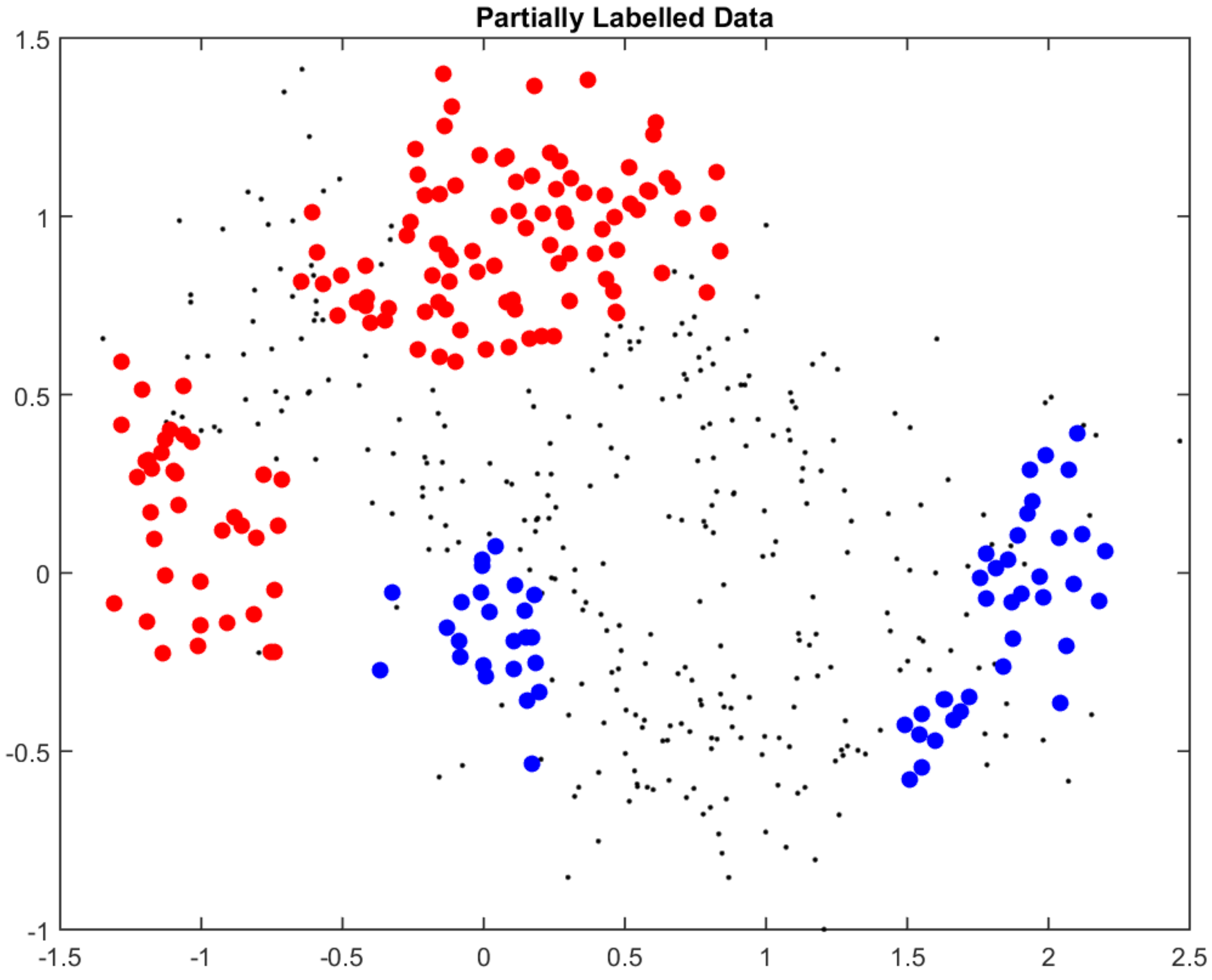
Label Propagation in Action



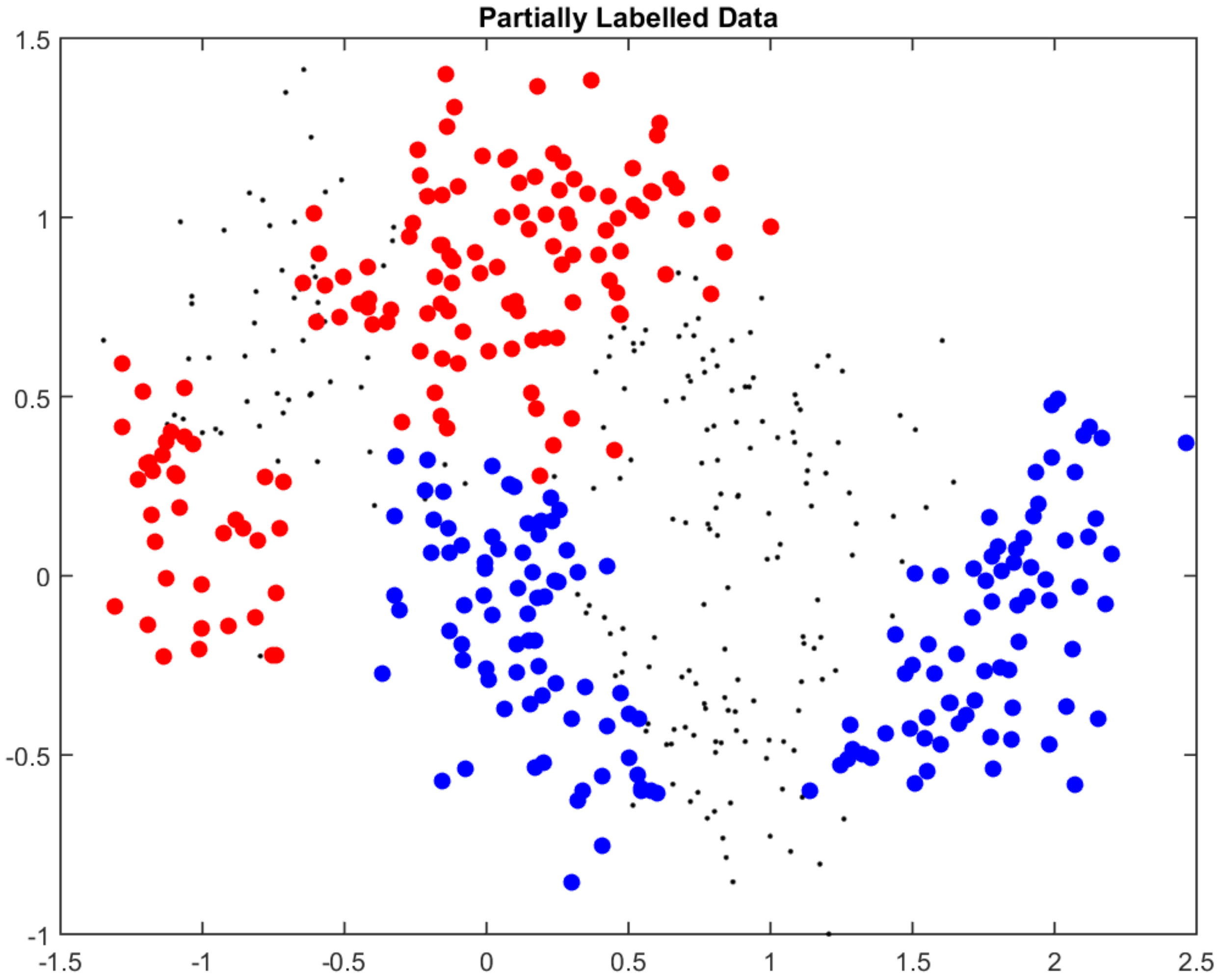
Label Propagation in Action



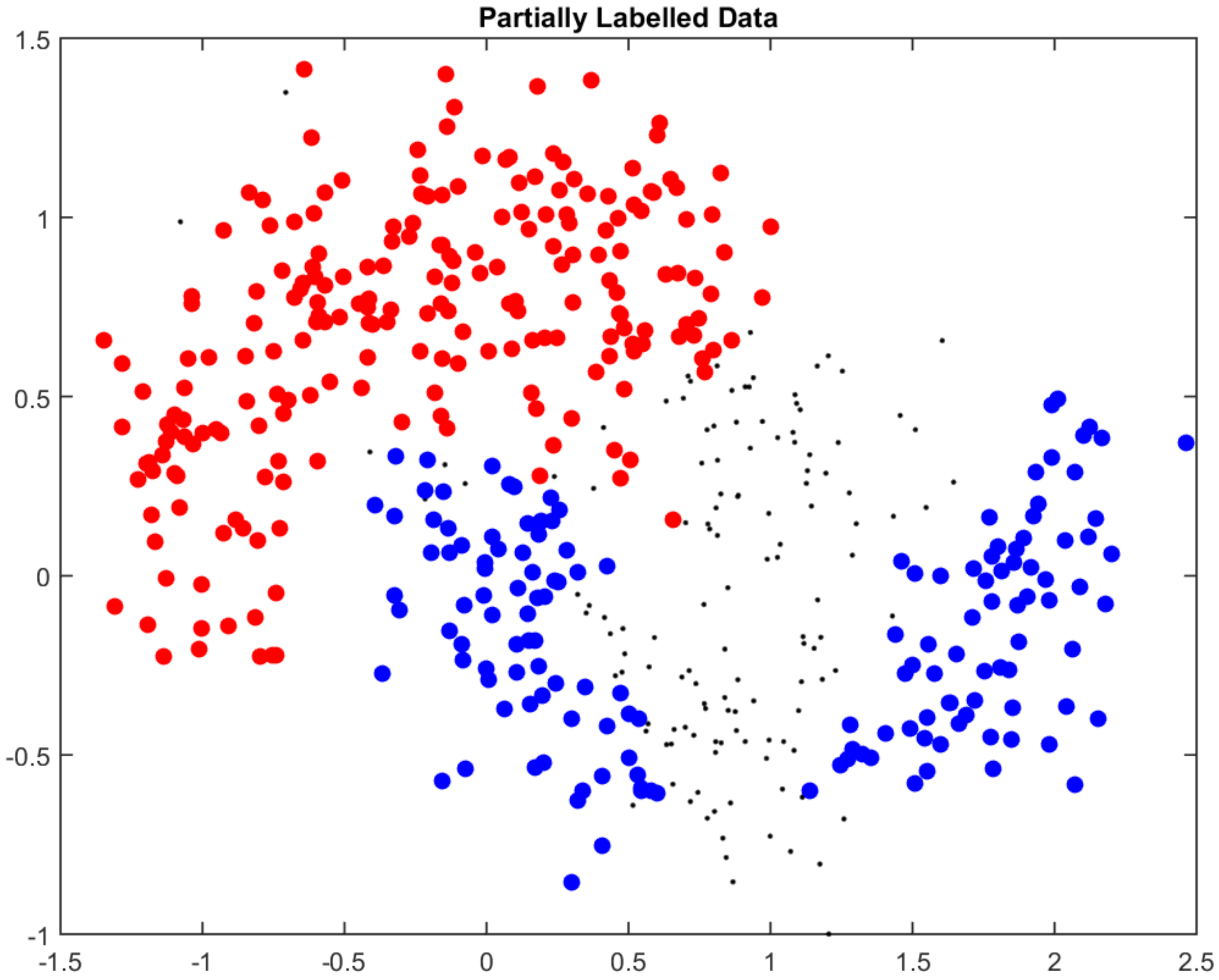
Label Propagation in Action



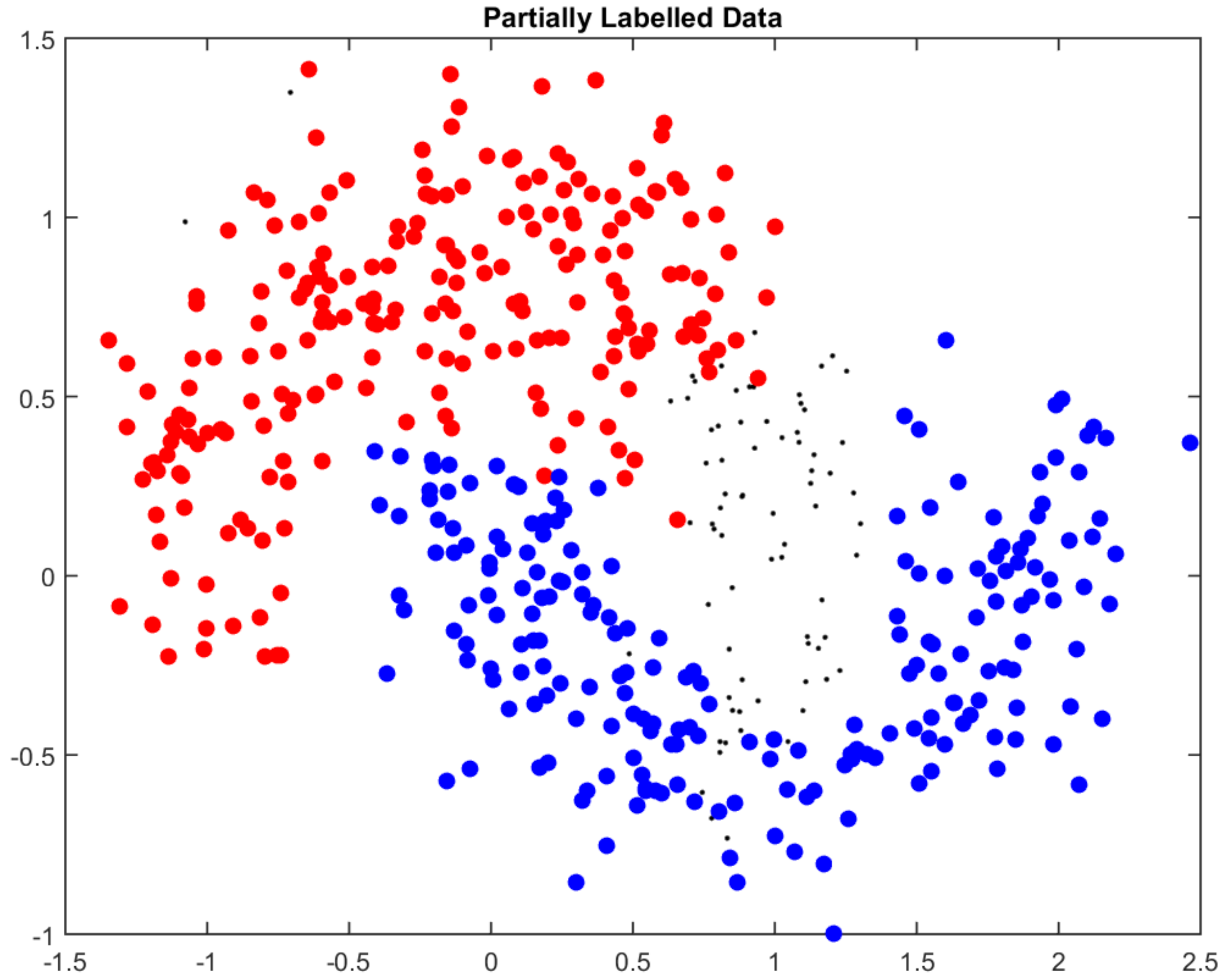
Label Propagation in Action



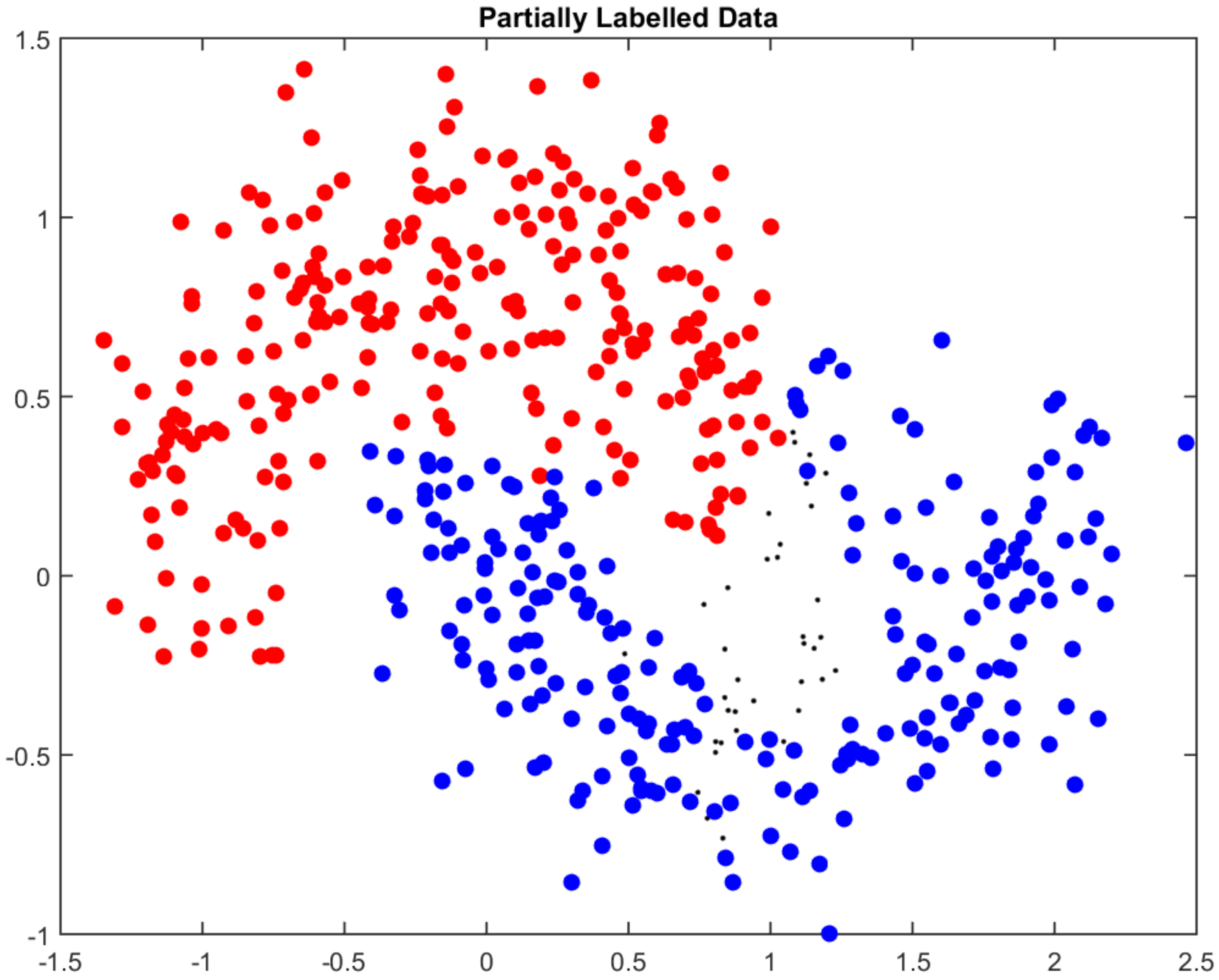
Label Propagation in Action



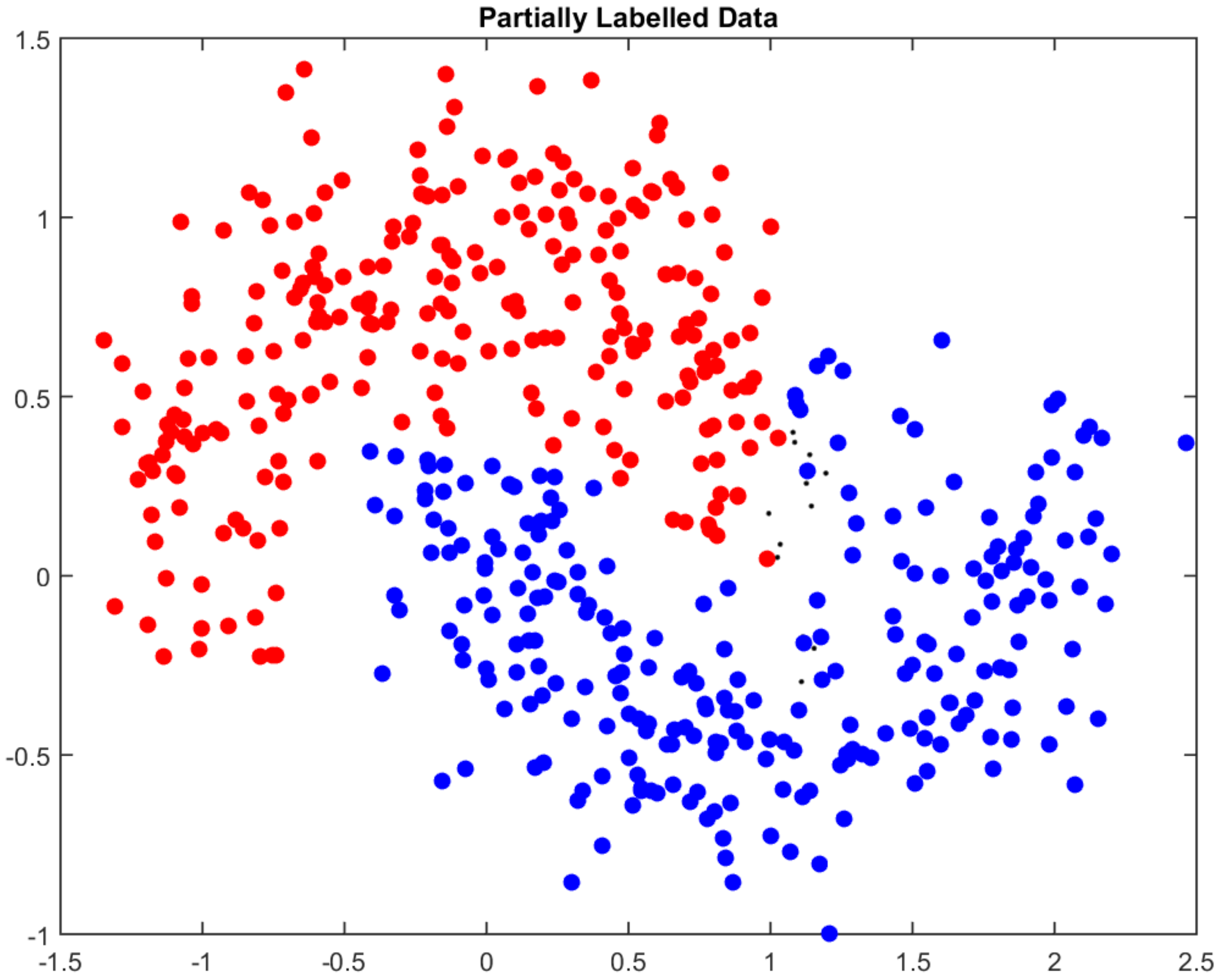
Label Propagation in Action



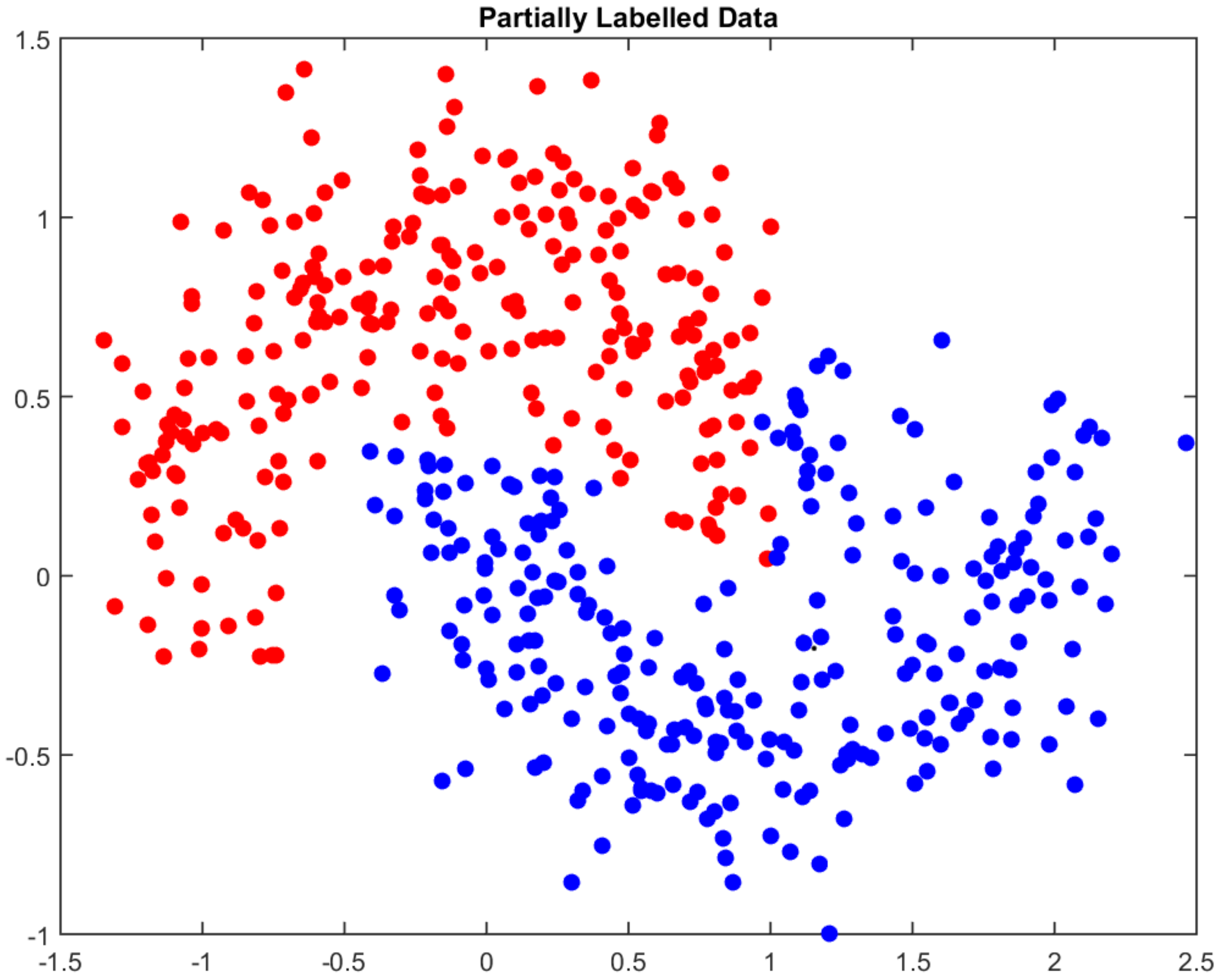
Label Propagation in Action



Label Propagation in Action



Label Propagation in Action



Graph-Based SSL (Label Propagation)

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

$$f(\bar{y}) = \sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}\underbrace{(y_i - \bar{y}_j)^2}_{\text{make } \tilde{y}_j \text{ similar to labeled neighbour}} + \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t}\bar{w}_{ij}\underbrace{(\bar{y}_i - \bar{y}_j)^2}_{\text{make unlabeled neighbours similar to each other}}$$

Do gradient descent on labels of unlabelled examples.

graph weight between labeled and unlabeled

Leads to "label propagation" through graph.

- Common variations:
  - Treat labels y_i as variables (they might be wrong).
    - Weight how much you trust original labels.
  - Regularize the unlabeled $\bar{y}_i$ towards a default value.
    - Can reflect that example is really far from any labeled example.

Example: Tagging YouTube Videos

- 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.

- Becoming popular in bioinformatics:
  - Label a subset of genes using manual experiments.
  - Find out which genes interact using more manual experiments.
  - Predict function/location/etc of genes using label propagation.

- Comments on graph-based SSL:
  - Transductive method: only estimates the unknown labels.
  - Often surprisingly effective even if you only have a few labels.
  - Does not need features if you have the weighted graph.

Graph-Based SSL as Markov Chain

- Standard graph-based SSL has a random walk interpretation:
  - At time t = 0, set your state to the node you want to label.
  - At time t > 0 and on a labeled node, output the label.
  - At time t > 0 and on an unlabeled node:
    - Move to neighbour 'j' with probability proportional to w_{ij}.
- Final predictions are probabilities of outputting each label.

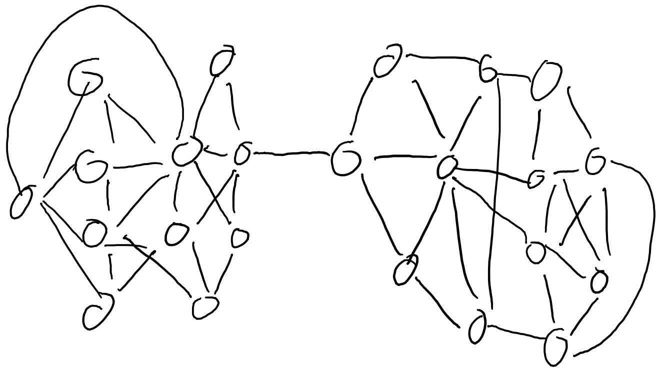
- Labeled nodes are called absorbing states in the Markov chain:
  - States that you can never leave from.
- Common variation where you don't "trust" labels:
  - Include absorbing-state "label node" as a neighbour of labeled nodes.
  - These neighbours get chosen with probability proportional to w_{ii}.

What else can we do with random walks?

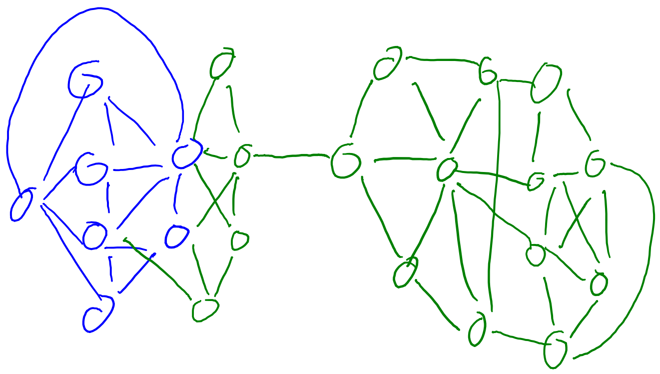
- We've discussed random walks for ranking and SSL.
  - Useful for problems defined on graphs.
  - We can convert from features to graphs using things like KNN graphs.
- Random walks for other tasks:
  - Outlier detection with outrank:
    - Examples with low PageRank are considered outliers (can detect outlier clusters).

What else can we do with random walks?

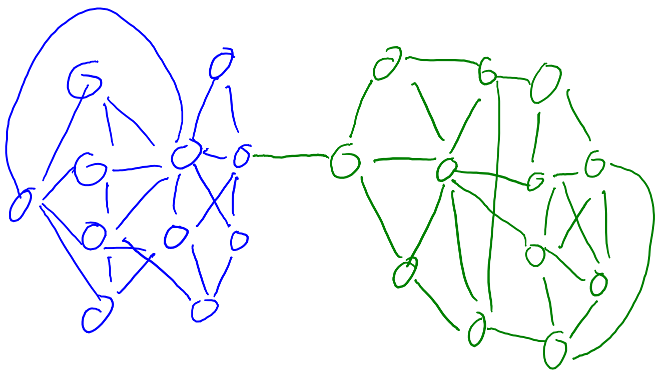
- We've discussed random walks for ranking and SSL.
  - Useful for problems defined on graphs.
  - We can convert from features to graphs using things like KNN graphs.
- Random walks for other tasks:
  - Clustering with spectal clustering (and "spectral graph theory):
    - "If we start in cluster 'c', random walk should stay in cluster 'c'".



Graph representation of data

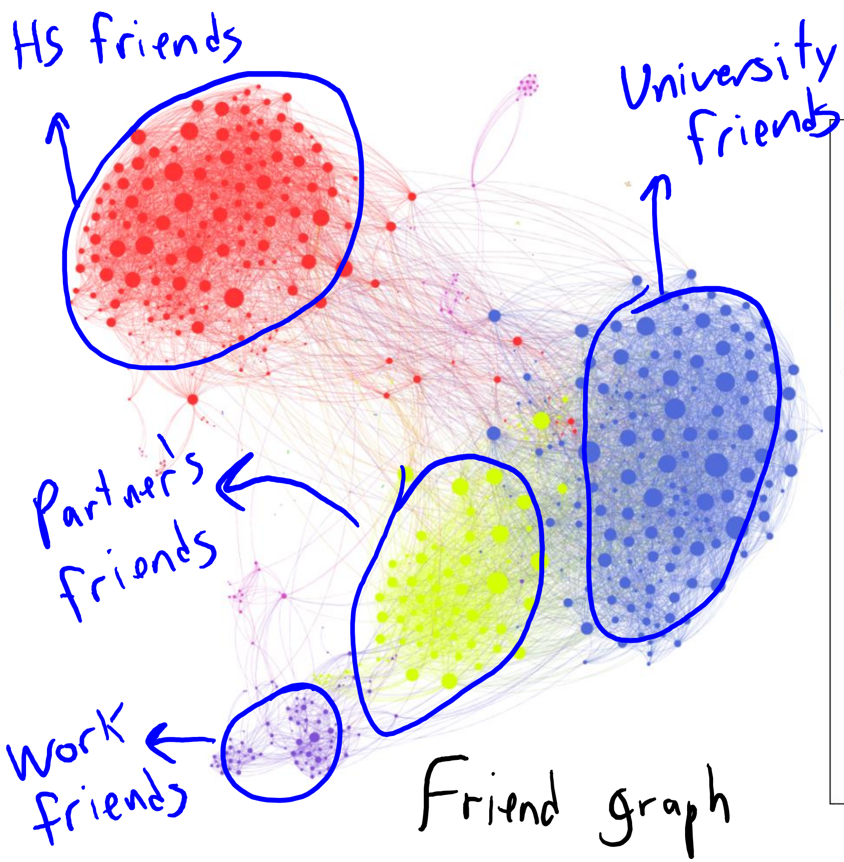


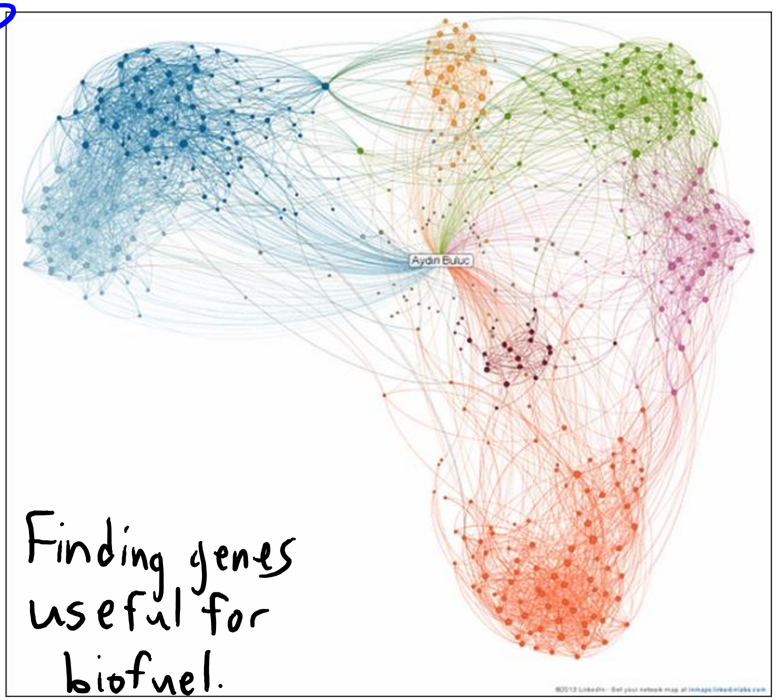
Bad clustering

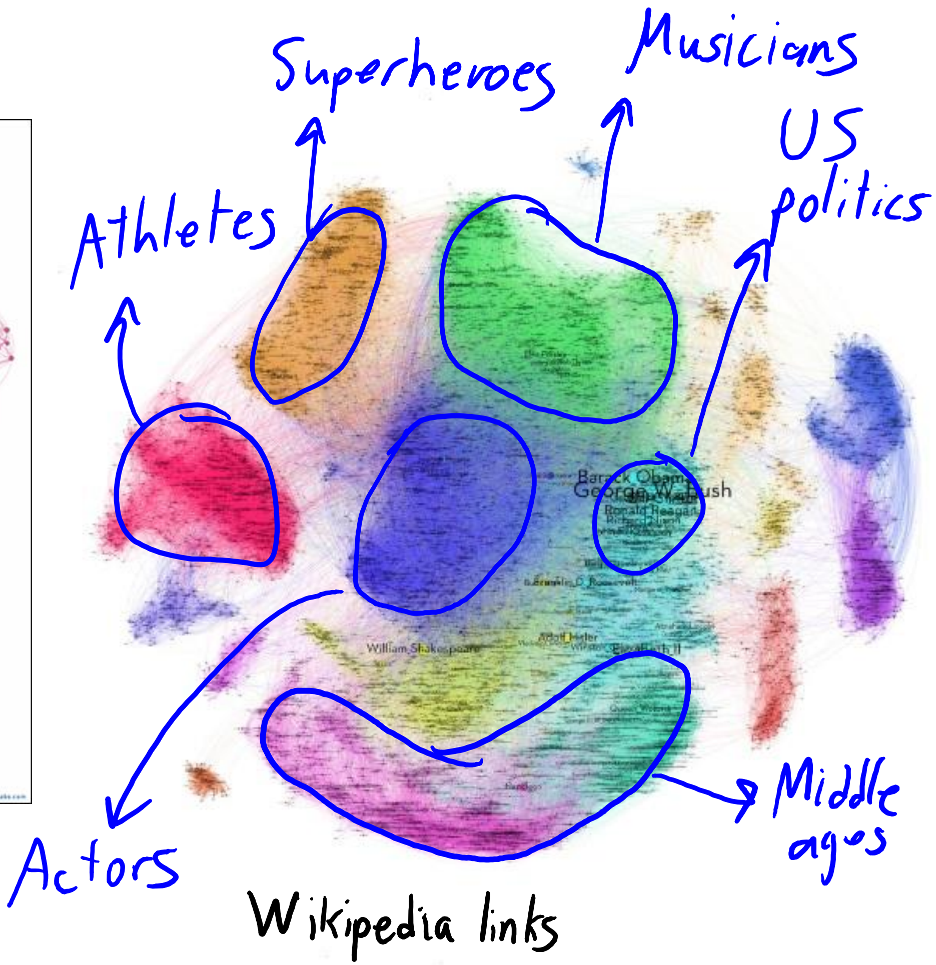


Good clustering.

Graph-Based Clustering Methods







https://griffsgraphs.wordpress.com/tag/clustering/
http://ascr-discovery.science.doe.gov/2013/09/sifting-genomes/
https://www.hackdiary.com/2012/04/05/extracting-a-social-graph-from-wikipedia-people-pages/

What else can we do with Markov chains?

- Common tasks we want to do with Markov chains:
  1. Sampling: given model, simulate from $p(x_t, x_{t-1}, \dots, x_0)$.
  2. Inference: given model, compute $p_t(x_t = s)$.
  3. Stationary distribution: $p_\infty(x_\infty = s)$.
  4. Decoding: find most likely sequence: $\max_{x1,x2,\dots,xt} p(x_t, x_{t-1}, x_{t-2}, \dots, x_0)$.
  5. Conditional inference: $p(x_t = s_1 \mid x_{t-1} = s_2, x_{t+10} = s_3)$.
  6. Learning: estimating $p(x_t = s_1 \mid x_t = s_2)$ from data to make model.
- Each of these is useful in particular applications.
- Generalizations of Markov models (CPSC 540):
  - Hidden Markov models: can't directly observe state of Markov model.
    - Sequence of observations depends on values of hidden states of Markov chain.
  - Graphical models generalize Markov chains beyond sequences:
    - Hierarchies, images, general graphs.

Fun with Markov Chains

- Snakes and ladders:
  - http://datagenetics.com/blog/november12011/index.html
- Candyland:
  - http://www.datagenetics.com/blog/december12011/index.html
- Yahtzee:
  - http://www.datagenetics.com/blog/january42012
- Find your car keys:
  - http://datagenetics.com/blog/november32016/index.html

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/co-training alternate labeling/fitting.
- Graph-based SSL propagates labels in graph (no features needed).
- Random walks can be used for other tasks:
  – Outrank for outlier detection.
  – Spectral clustering for clustering.

- Next time:
  – Review of topics we've covered, overview of topics we didn't.