Semi-Supervised Classification

learning from labeled and unlabeled data

Xiaojin "Jerry" Zhu

jerryzhu@cs.wisc.edu

Computer Science Department
University of Wisconsin, Madison

"Eclipse"


1.jpeg
2.jpeg
3.jpeg
4.jpeg
5.jpeg
6.jpeg
7.jpeg
8.jpeg
9.jpeg
10.jpeg
11.jpeg
12.jpeg
13.jpeg
14.jpeg
15.jpeg
16.jpeg
17.jpeg
18.jpeg
19.jpeg
20.jpeg

"Eclipse"

A good classifier needs good labeled training data.

Labels can be hard to get

- Human annotation is slow, boring

Labels can be hard to get

- Human annotation is slow, boring
- It may require expert knowledge

Labels can be hard to get

- Human annotation is slow, boring
- It may require expert knowledge
- It may require special, expensive devices

Labels can be hard to get

- Human annotation is slow, boring
- It may require expert knowledge
- It may require special, expensive devices
- Your graduate student is on vacation

Labels can be hard to get

- Human annotation is slow, boring
- It may require expert knowledge
- It may require special, expensive devices
- Your graduate student is on vacation

Unlabeled data often easy to obtain in large amount.

In this class we will learn how to use them for classification.

Semi-supervised classification

Goal:

Using both labeled and unlabeled data to build better classifiers (than using labeled data alone).

Notation:

- input x, label y
- classifier $f : \mathcal{X} \mapsto \mathcal{Y}$
- labeled data $(X_l, Y_l) = \{(x_1, y_1), \ldots, (x_l, y_l)\}$
- unlabeled data $X_u = \{x_{l+1}, \ldots, x_n\}$
- usually $n \gg l$

Outline

We will discuss some representative semi-supervised learning methods

1. Self-training and Co-training
2. Generative probabilistic models
3. Semi-supervised support vector machines
4. Graph-based semi-supervised learning

1. Self- and Co- Training

Self-training

Algorithm: Self-training

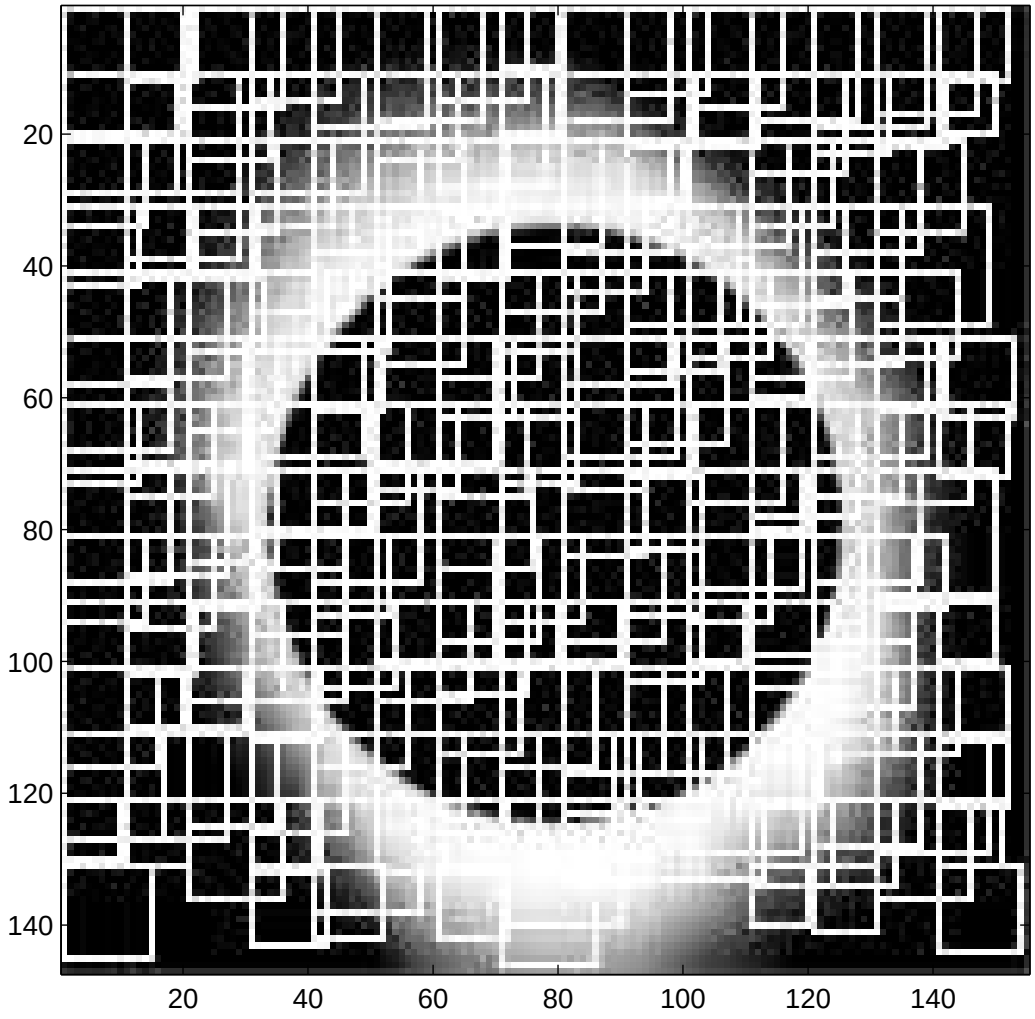
1. Pick your favorite classification method. Train a classifier f from (X_l, Y_l).
2. Use f to classify all unlabeled items $x \in X_u$.
3. Pick x^* with the highest confidence, add $(x^*, f(x^*))$ to labeled data.
4. Repeat.

The simplest semi-supervised learning method.

An example

Each image is divided into small patches

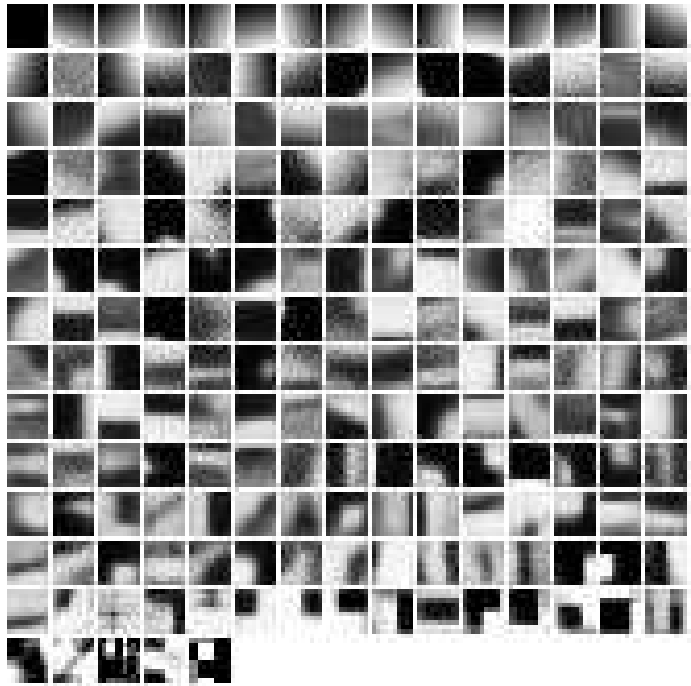
(10×10 grid, random size in $10 \sim 20$)



An example

All patches are normalized.

Define a dictionary of 200 'visual words' (cluster centroids) with k-means clustering on all patches.



Patches represented by the index of the closest visual word.

An example

Images represented by bag-of-word  →

1:0 2:1 3:2 4:2 5:0 6:0 7:0 8:3 9:0 10:3 11:31 12:0 13:0 14:0 15:0 16:9 17:1 18:0 19:0

20:1 21:0 22:0 23:0 24:0 25:6 26:0 27:6 28:0 29:0 30:0 31:1 32:0 33:0 34:0 35:0 36:0

37:0 38:0 39:0 40:0 41:0 42:1 43:0 44:2 45:0 46:0 47:0 48:0 49:3 50:0 51:3 52:0 53:0

54:0 55:1 56:1 57:1 58:1 59:0 60:3 61:1 62:0 63:3 64:0 65:0 66:0 67:0 68:0 69:0 70:0

71:1 72:0 73:2 74:0 75:0 76:0 77:0 78:0 79:0 80:0 81:0 82:0 83:0 84:3 85:1 86:1 87:1

88:2 89:0 90:0 91:0 92:0 93:2 94:0 95:1 96:0 97:1 98:0 99:0 100:0 101:1 102:0 103:0

104:0 105:1 106:0 107:0 108:0 109:0 110:3 111:1 112:0 113:3 114:0 115:0 116:0 117:0

118:3 119:0 120:0 121:1 122:0 123:0 124:0 125:0 126:0 127:3 128:3 129:3 130:4 131:4

132:0 133:0 134:2 135:0 136:0 137:0 138:0 139:0 140:0 141:1 142:0 143:6 144:0 145:2

146:0 147:3 148:0 149:0 150:0 151:0 152:0 153:0 154:1 155:0 156:0 157:3 158:12 159:4

160:0 161:1 162:7 163:0 164:3 165:0 166:0 167:0 168:0 169:1 170:3 171:2 172:0 173:1

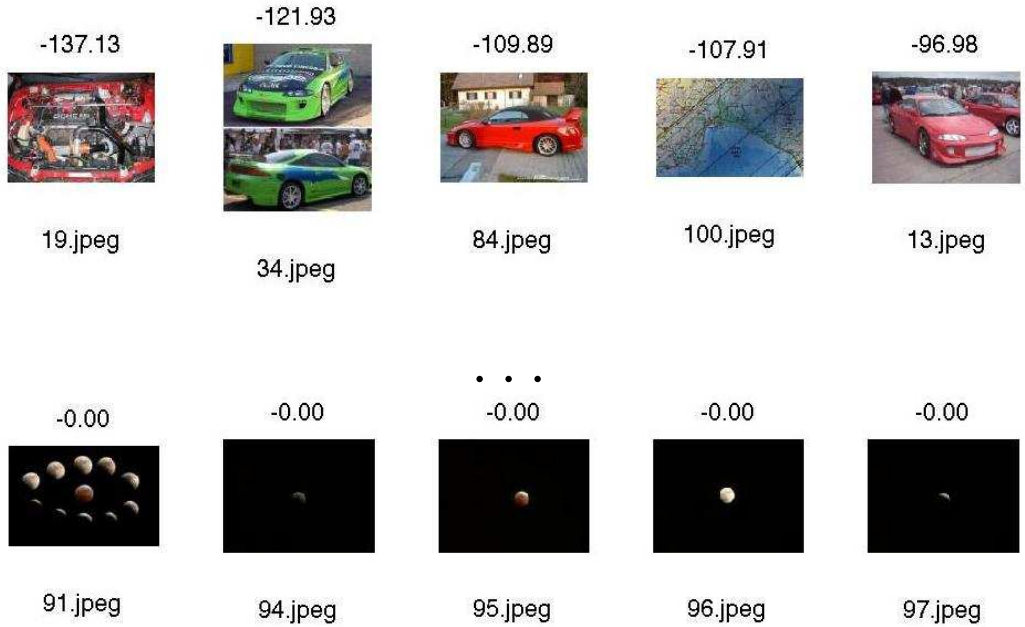
174:0 175:0 176:2 177:0 178:0 179:1 180:0 181:1 182:2 183:0 184:0 185:2 186:0 187:0

188:0 189:0 190:0 191:0 192:0 193:1 194:2 195:4 196:0 197:0 198:0 199:0 200:0

An example

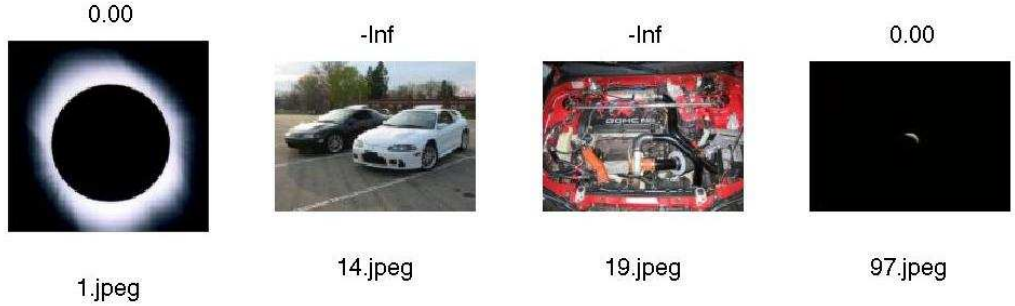
1. Train a naïve Bayes classifier on initial labeled data

2. Classify unlabeled data, sort by confidence $\log p(y = a|x)$

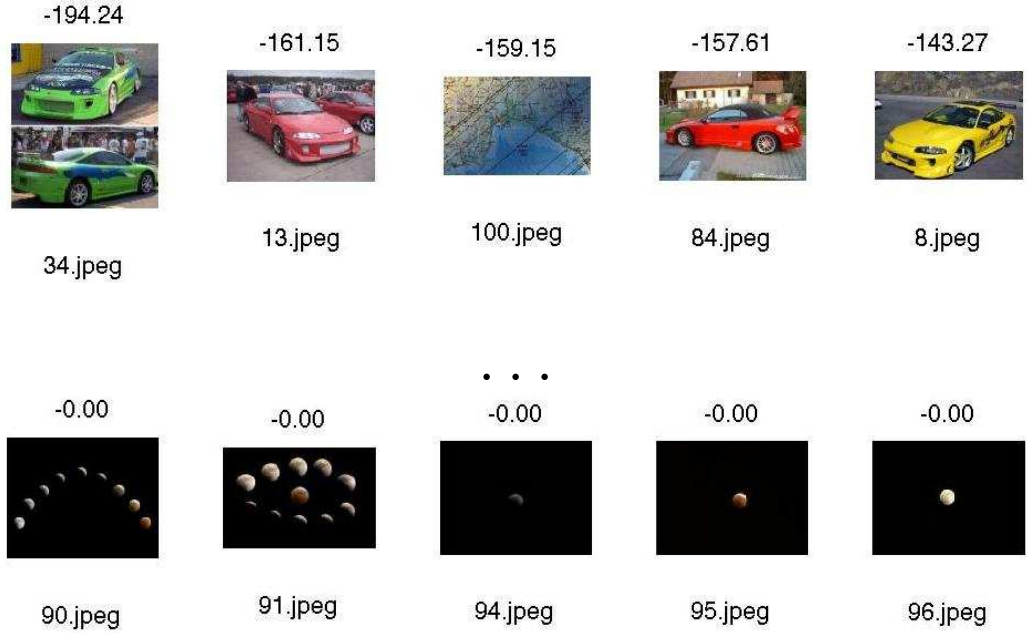


An example

3. Add the most confident images (and computed labels) to labeled data



4. Re-train classifier, classify unlabeled data, repeat



Pros and cons of self-training

Pros

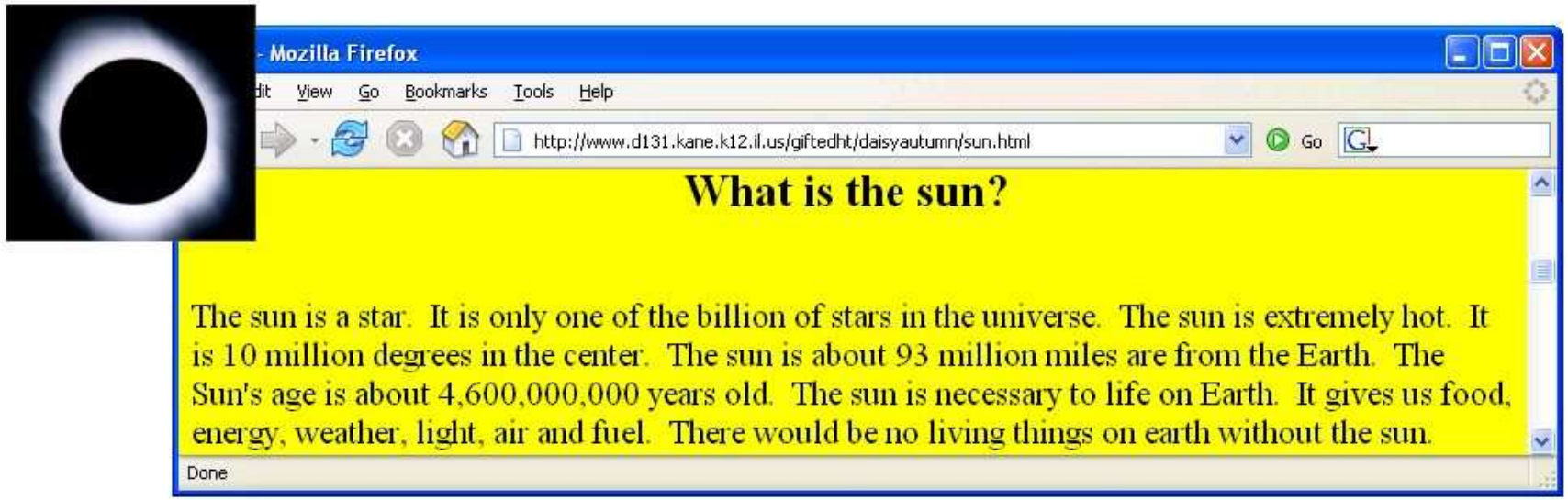
- Simple
- Applies to almost all existing classifiers

Cons

- Mistakes reinforce themselves. Heuristics against pitfalls
  - 'Un-label' a training point if its classification confidence drops below a threshold
  - Randomly perturb learning parameters
- Can't say too much

Co-training

Two views of an item: image and HTML text



Feature split

Each item is represented by two kinds 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 algorithm

Algorithm: Co-training

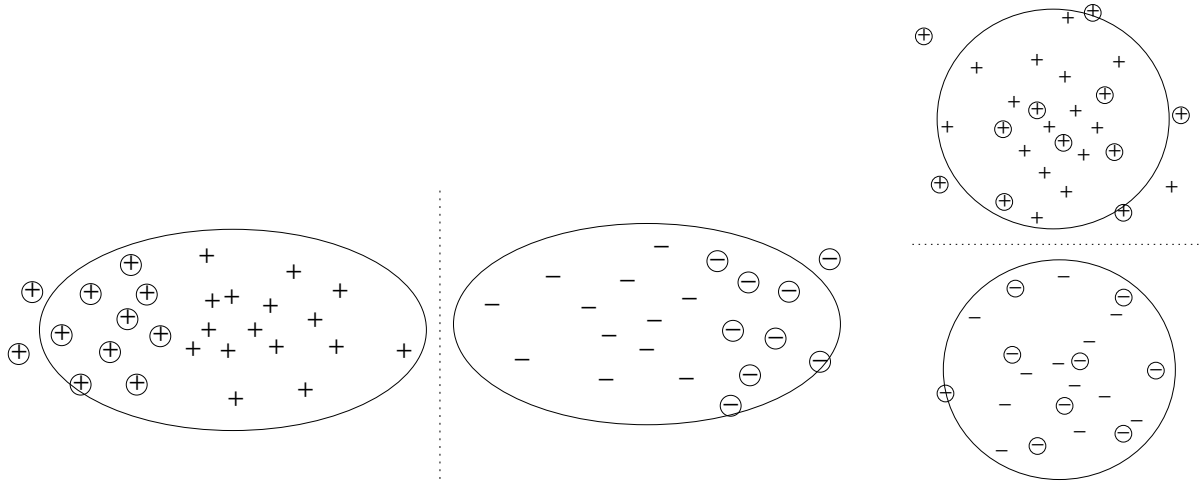
1. Train two classifiers: $f^{(1)}$ from $(X_l^{(1)}, Y_l)$, $f^{(2)}$ from $(X_l^{(2)}, Y_l)$.
2. Classify X_u with $f^{(1)}$ and $f^{(2)}$ separately.
3. Add $f^{(1)}$'s k-most-confident $(x, f^{(1)}(x))$ to $f^{(2)}$'s labeled data.
4. Add $f^{(2)}$'s k-most-confident $(x, f^{(2)}(x))$ to $f^{(1)}$'s labeled data.
5. Repeat.

Co-training assumptions

Co-training assumes that

- feature split $x = [x^{(1)}; x^{(2)}]$ 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

X_1 view X_2 view



Pros and cons of co-training

Pros

- Simple. Applies to almost all existing classifiers
- Less sensitive to mistakes

Cons

- Feature split may not exist
- Models using BOTH features should do better

Variants of co-training

Co-EM: add all, not just top k

- Each classifier probabilistically label X_u
- Add (x, y) with weight $P(y|x)$

Single-view: fake feature split

- create random, artificial feature split
- apply co-training

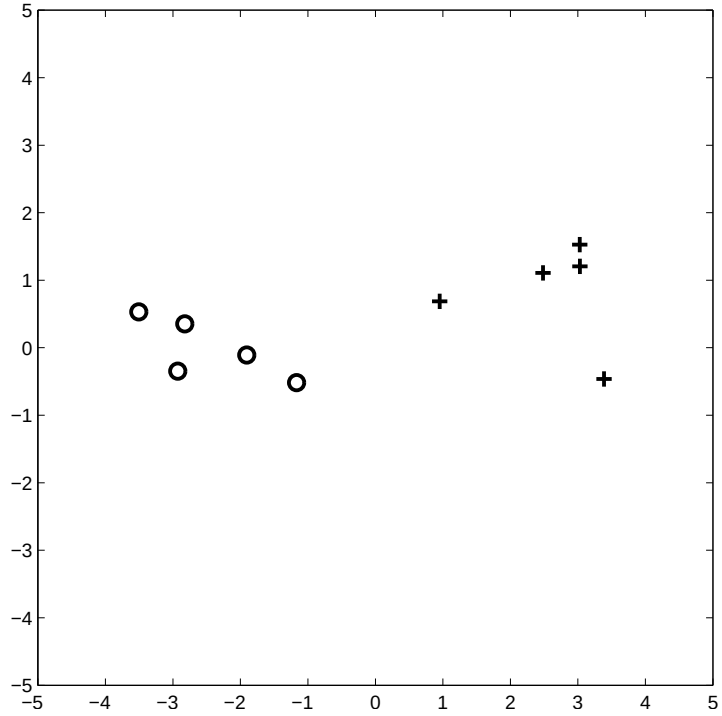
Single-view: agreement among multiple classifiers

- train multiple classifiers of different types
- classify unlabeled data with all classifiers
- add majority vote label

2. Generative probabilistic models

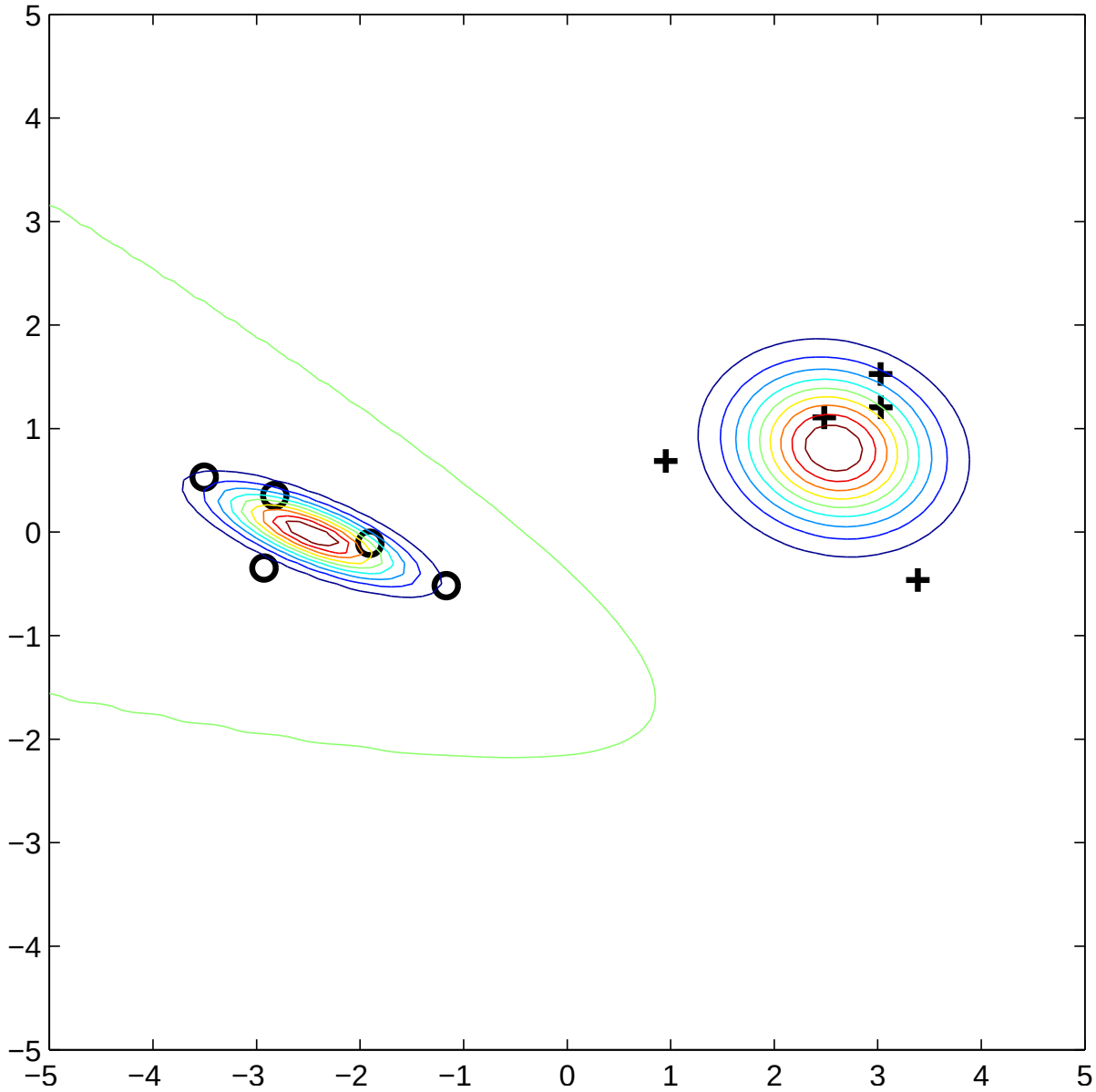
A simple example

Labeled data (X_l, Y_l)



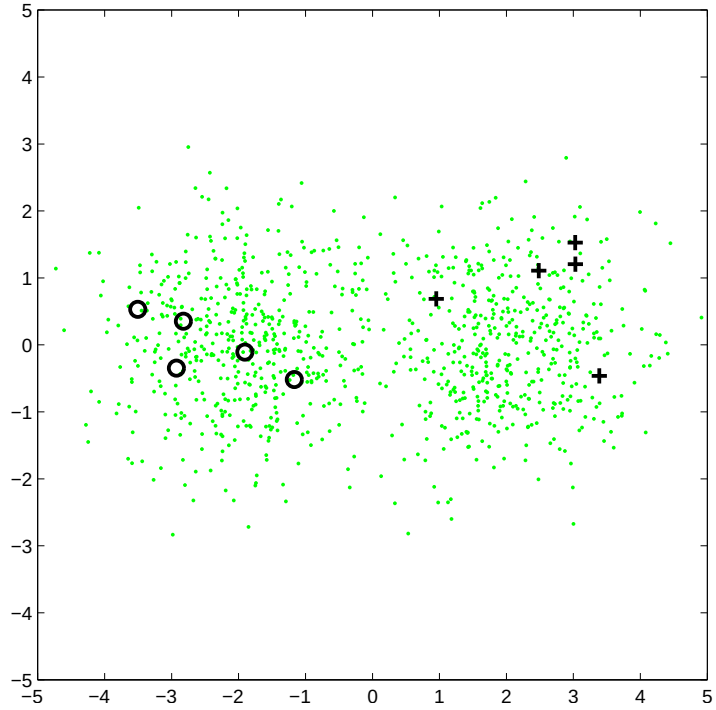
Assuming each class has a Gaussian distribution in feature space, what is the most likely decision boundary?

The most-likely model



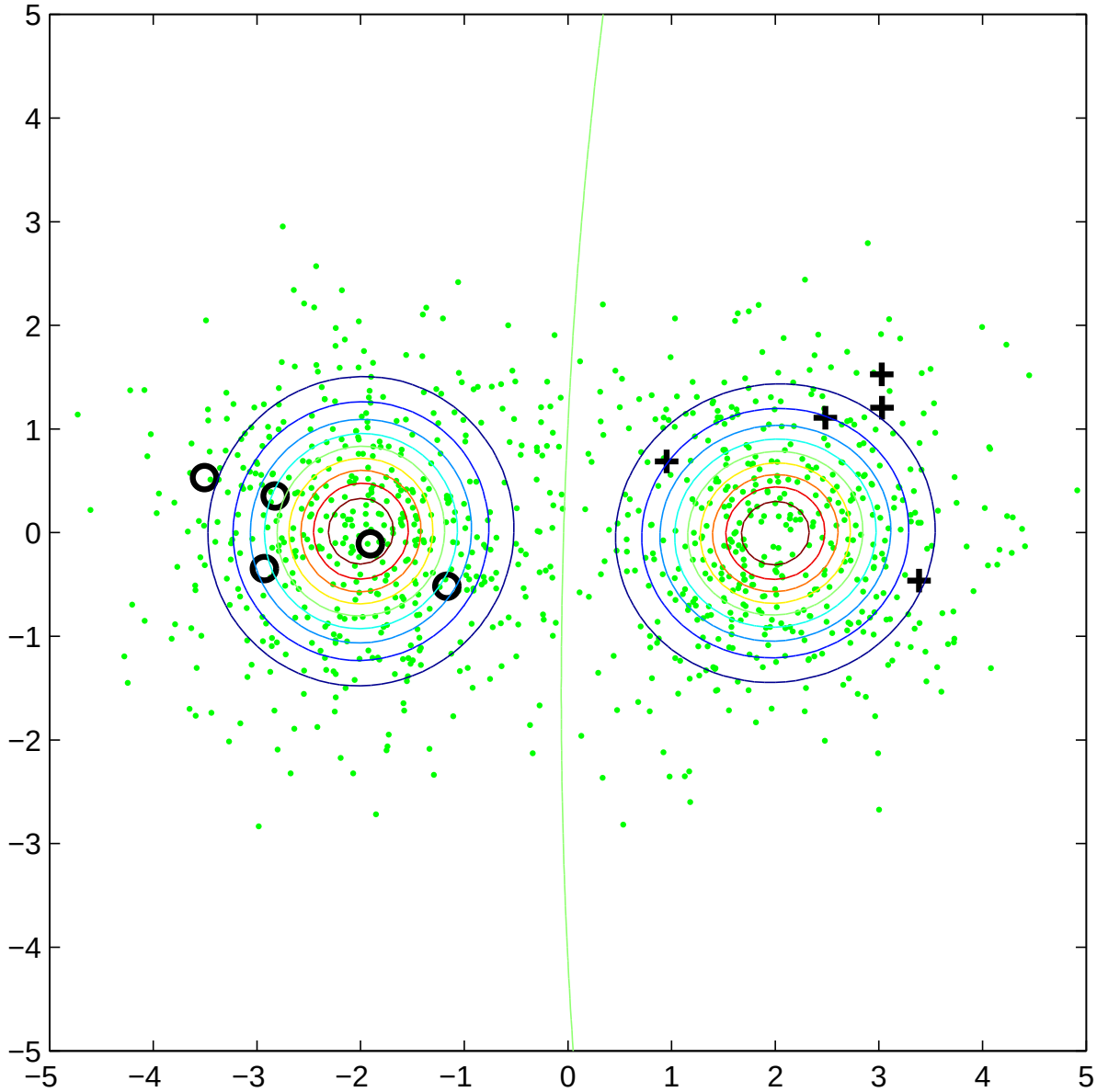
Adding unlabeled data

Labeled data (X_l, Y_l) and unlabeled data X_u



What is the most likely decision boundary now?

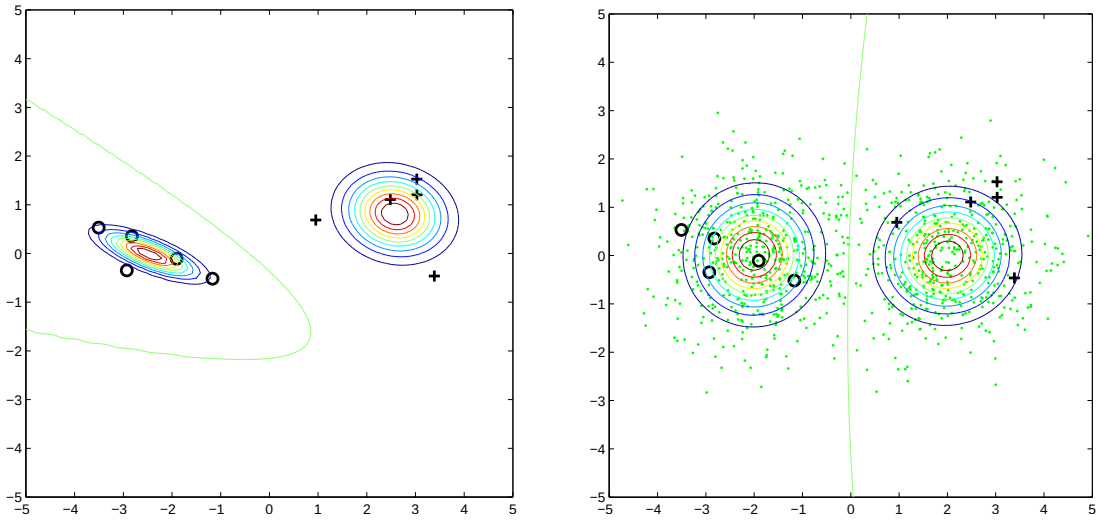
The most-likely model



Why are they different?

- This is a Gaussian mixture model
- Parameters θ:
  - weight of each class $p(y|\theta)$
  - mean and covariance of each Gaussian $p(x|y,\theta)$
- maximum likelihood estimate of parameters θ

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



Maximizing different likelihood

With labeled data only

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

- maximum likelihood estimate (MLE) for θ trivial

With 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)
- Expectation-Maximization (EM), variational approximation, etc.

Maximum a posteriori (MAP) possible with prior $p(\theta)$

Generative probabilistic models

A joint probabilistic model $p(x, y|\theta)$, e.g.,

- Gaussian mixture models
- Multinomial mixture models (Naive Bayes)
- Latent Dirichlet allocation variants
- Hidden Markov models (HMMs)

In contrast to discriminative models which model $p(y|x)$ directly (logistic regression, support vector machines, conditional random fields etc.)

Generative models algorithm

Algorithm: Generative models

1. Choose a generative model $p(x, y|\theta)$
2. Find the MLE on labeled and unlabeled data

$$\theta^* = \arg\max_\theta p(X_l, Y_l, X_u|\theta)$$

3. Compute class distribution using Bayes' rule

$$p(y|x, \theta^*) = \frac{p(x, y|\theta^*)}{\sum_{y'} p(x, y'|\theta^*)}$$

We will discuss one method for finding θ^*: the EM algorithm.

EM for Gaussian mixture models

Start from MLE θ on (X_l, Y_l)

- $p(y|\theta)$: proportion of data with label y
- $p(x|y,\theta)$: mean and covariance of data with label y

Repeat the two steps

1. E-step: compute the expected labels $p(y|x,\theta)$ for all $x \in X_u$
   - assign class 1 to $p(y=1|x,\theta)$ fraction of x
   - assign class 2 to $p(y=2|x,\theta)$ fraction of x
2. M-step: update MLE θ with the original labeled and (now labeled) unlabeled data

EM algorithm in general

Problem set up

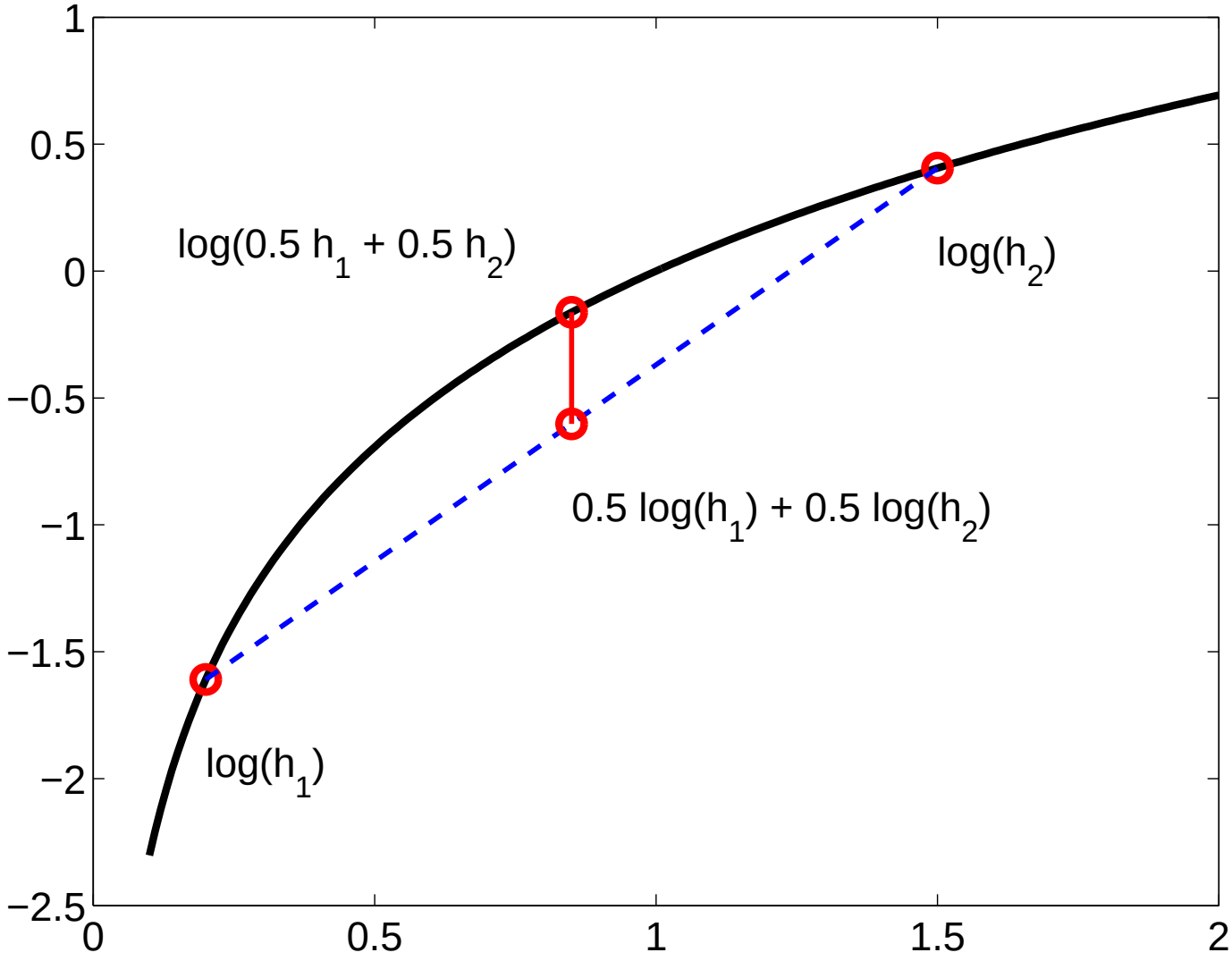
- observed data $\mathcal{D} = (X_l, Y_l, X_u)$
- hidden data $\mathcal{H} = Y_u$
- $p(\mathcal{D}|\theta) = \sum_{\mathcal{H}} p(\mathcal{D}, \mathcal{H}|\theta)$
- goal is to find θ^* to maximize $p(\mathcal{D}|\theta)$
- EM starts from an arbitrary θ_0
- EM iteratively improves $p(\mathcal{D}|\theta)$
- EM converges to a local maximum

How EM works

- Instead of $p(\mathcal{D}|\theta)$, EM works on $\log p(\mathcal{D}|\theta) \equiv \mathcal{L}(\theta)$
- EM constructs a lower bound $\mathcal{F}(q, \theta) \leq \mathcal{L}(\theta)$
  - auxiliary distribution q
  - Jensen's inequality, concavity of log
- $\mathcal{F}(q, \theta)$ is easier to optimize than $\mathcal{L}(\theta)$
  - coordinate ascent
  - fix θ, optimize q: the E-step
  - fix q, optimize θ: the M-step

Jensen's inequality on log()

$$\forall \sum q_i = 1, q_i \geq 0, \log \sum q_i h_i \geq \sum q_i \log h_i$$



The lower bound

Introducing an arbitrary auxiliary distribution on hidden data $q(\mathcal{H})$,

$$\begin{aligned}
\mathcal{L}(\theta) &\equiv \log p(\mathcal{D}|\theta) \\
&= \log \sum_{\mathcal{H}} p(\mathcal{D}, \mathcal{H}|\theta) \\
&= \log \sum_{\mathcal{H}} q(\mathcal{H}) \frac{p(\mathcal{D}, \mathcal{H}|\theta)}{q(\mathcal{H})} \\
&\geq \sum_{\mathcal{H}} q(\mathcal{H}) \log \frac{p(\mathcal{D}, \mathcal{H}|\theta)}{q(\mathcal{H})} \\
&\equiv \mathcal{F}(q, \theta)
\end{aligned}$$

Coordinate ascent on q

Fixing θ, $\mathcal{F}(q, \theta)$ is maximizes when

$$\frac{\partial}{\partial q(\mathcal{H})} q(\mathcal{H}) \log \frac{p(\mathcal{D}, \mathcal{H}|\theta)}{q(\mathcal{H})} = 0$$

$$\text{s.t.} \sum_{\mathcal{H}} q(\mathcal{H}) = 1$$

The maximizing $q^*(\mathcal{H}) = p(\mathcal{H}|\mathcal{D}, \theta)$

- $q^*(\mathcal{H})$ is the expected label under θ
- The E-step
- Under $q^*(\mathcal{H})$, $\mathcal{F}(q^*, \theta) = \mathcal{L}(\theta)$: the lower bound is tight

Coordinate ascent on θ

Fixing q^*, $\mathcal{F}(q^*, \theta)$ is maximized by maximizing

$$\sum_{\mathcal{H}} q^*(\mathcal{H}) \log p(\mathcal{D}, \mathcal{H}|\theta)$$

- 'Fractional' labels with probability $q^*(\mathcal{H})$
- The M-step

EM never decreases likelihood:

$$\mathcal{L}(\theta) \overset{\text{E-step}}{=} \mathcal{F}(q^*, \theta) \overset{\text{M-step}}{\leq} \mathcal{F}(q^*, \theta') \overset{\text{Jensen}}{\leq} \mathcal{L}(\theta')$$

EM converges to a local maximum.

Review: Generative models algorithm

Algorithm: Generative models

1. Choose a generative model $p(x, y|\theta)$
2. Find the MLE on labeled and unlabeled data

$$\theta^* = \arg\max_\theta p(X_l, Y_l, X_u|\theta)$$

   EM is one method.

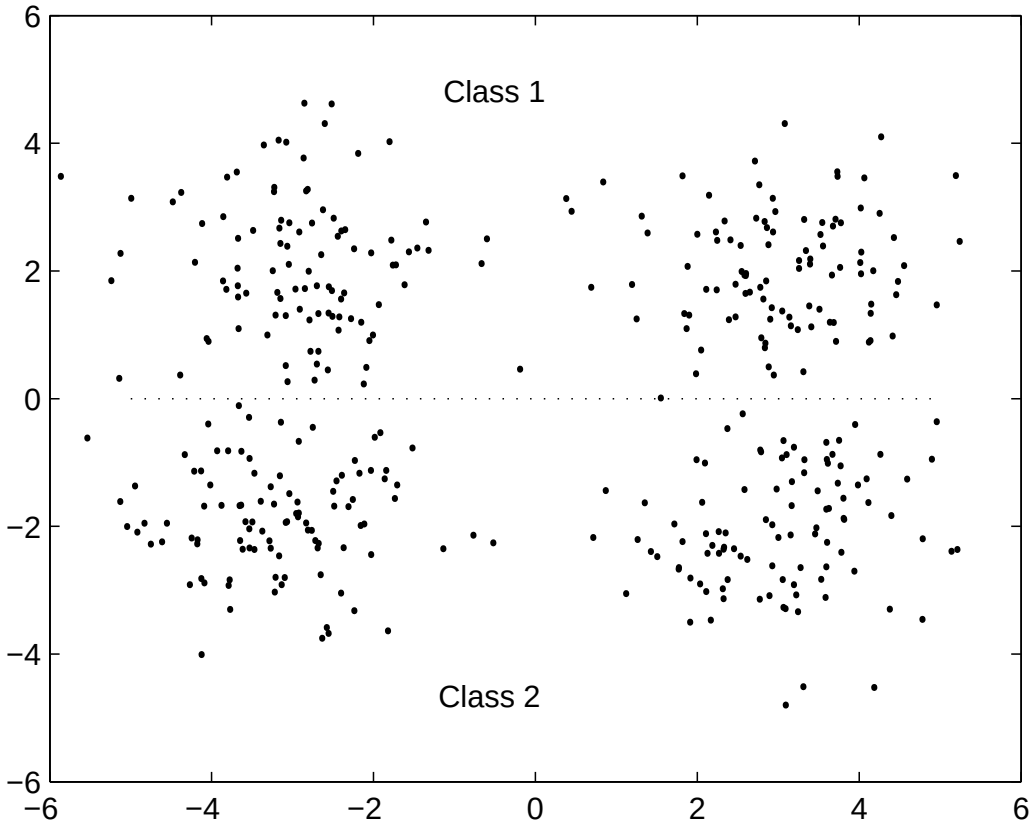
3. Compute class distribution using Bayes' rule

$$p(y|x, \theta^*) = \frac{p(x, y|\theta^*)}{\sum_{y'} p(x, y'|\theta^*)}$$

The Baum-Welch algorithm for HMMs is a form of EM, or semi-supervised learning method.

Pros and cons of generative models

- Pro: clear probabilistic framework
- Con: unlabeled data may hurt if generative model is wrong

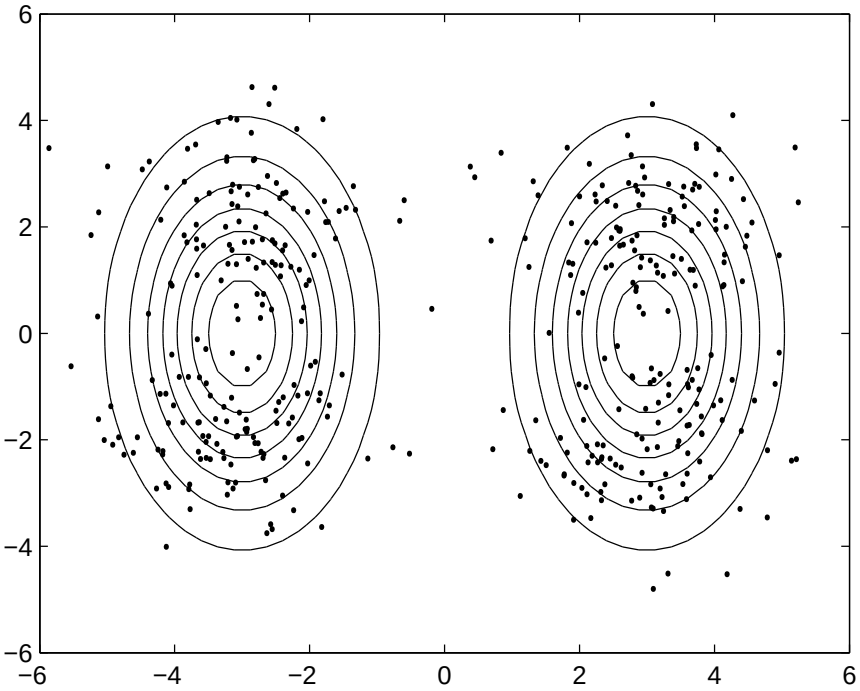


Unlabeled data hurts

If generative model is wrong ...

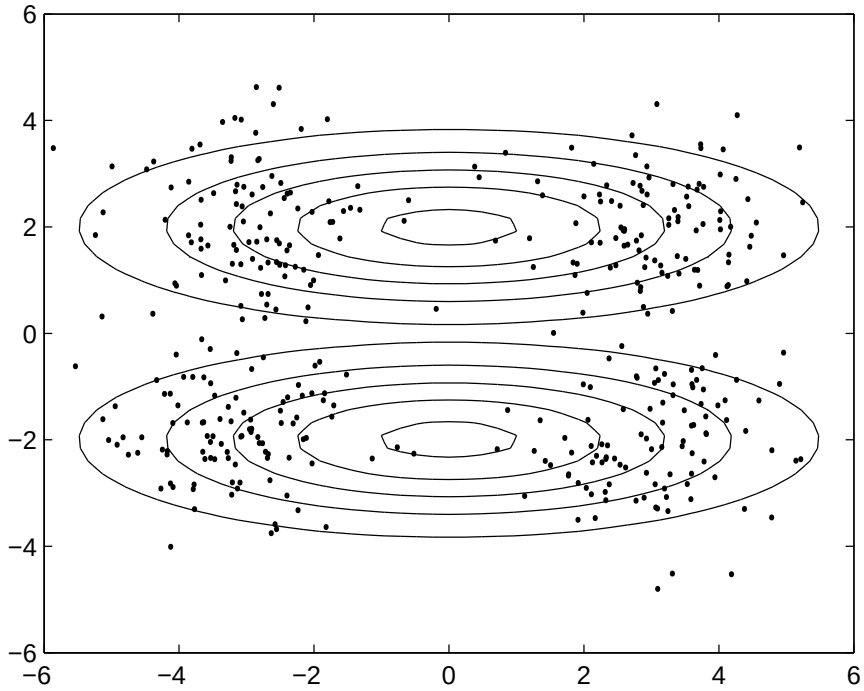
high likelihood

wrong



low likelihood

correct



Briefly mentioned

Clustering algorithms can be used for semi-supervised classification too:

- Run your favorite clustering algorithm on X_l, X_u.
- Label all points within a cluster by the majority of labeled points in the cluster.

Pro: Yet another simple method using existing algorithms.

Con: Hard to analyze.

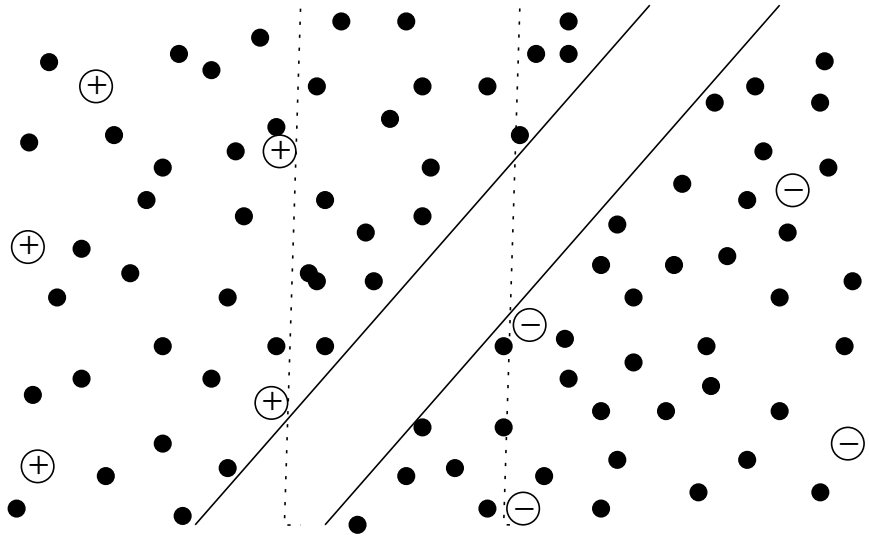
3. Semi-Supervised Support Vector Machines

Maximizing unlabeled margin

Standard SVMs: maximizing labeled margin

Semi-supervised SVMs (S3VMs, transductive SVMs): maximizing unlabeled margin

- Enumerate all 2^u possible labeling of X_u
- Build one standard SVM for each labeling
- Pick the SVM with the largest margin



SVM review

The setting of support vector machines

- two classes $y \in \{+1, -1\}$
- labeled data (X_l, Y_l)
- a kernel K
- the reproducing Hilbert kernel space $\mathcal{H}_K$
- SVM finds a function $f(x) = h(x) + b$ with $h \in \mathcal{H}_K$
- Classify x by $\text{sign}(f(x))$

Linearly separable SVMs

The linearly separable SVM: all training points must be outside the corresponding margin.

$$\min_{h,b} \|h\|^2_{\mathcal{H}_K}$$

$$\text{subject to } \; h(x_i) + b \geq 1 \quad \text{, if } y_i = 1 \quad \forall i = 1 \ldots l$$

$$h(x_i) + b \leq -1 \quad \text{, if } y_i = -1$$

The two constraints can be written as

$$y_i(h(x_i) + b) \geq 1 \quad \forall i = 1 \ldots l$$

Data may not be linearly separable, even in $\mathcal{H}_K$.

Soft margin SVMs

Training points may violate the margin.

$$\min_{h,b,\xi} \sum_{i=1}^{l} \xi_i + \lambda \|h\|^2_{\mathcal{H}_K}$$

$$\text{subject to } \; y_i(h(x_i) + b) \geq 1 - \xi_i \quad , \forall i = 1 \ldots l$$

$$\xi_i \geq 0$$

ξ's are slack variables, penalized.

Hinge function

$$\min_{\xi} \xi$$

$$\text{subject to } \xi \geq z$$

$$\xi \geq 0$$

If $z \leq 0$, $\min \xi = 0$

If $z > 0$, $\min \xi = z$

Therefore the constrained optimization problem above is equivalent to the hinge function

$$(z)_+ = \max(z, 0)$$

SVM with hinge function

Let $z_i = 1 - y_i(h(x_i) + b) = 1 - y_i f(x_i)$, the problem

$$\min_{h,b,\xi} \sum_{i=1}^{l} \xi_i + \lambda \|h\|^2_{\mathcal{H}_K}$$

$$\text{subject to } \; y_i(h(x_i) + b) \geq 1 - \xi_i \quad , \forall i = 1 \ldots l$$

$$\xi_i \geq 0$$

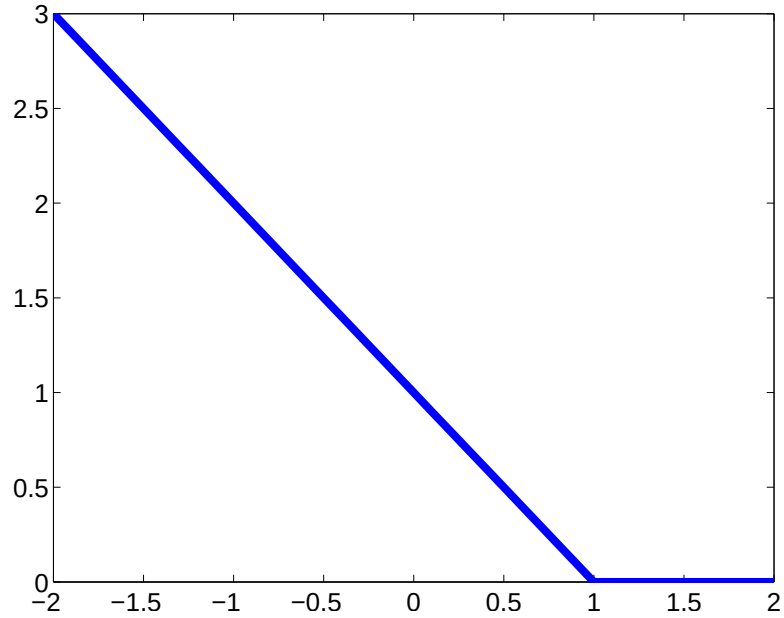
is equivalent to

$$\min_{f} \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda \|h\|^2_{\mathcal{H}_K}$$

The hinge loss

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda \|h\|^2_{\mathcal{H}_K}$$

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



$y_i f(x_i)$

Prefers labeled points on the 'correct' side.

Semi-supervised SVMs

How to incorporate unlabeled points?

- Assign putative labels $\text{sign}(f(x))$ to $x \in X_u$
- $\text{sign}(f(x))f(x) = |f(x)|$
- The hinge loss on unlabeled points

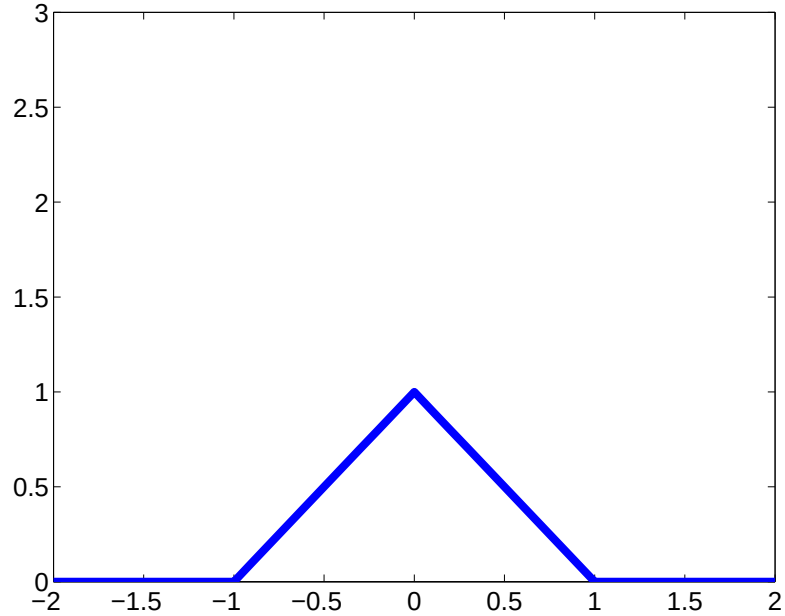
$$(1 - y_i f(x_i))_+ = (1 - |f(x_i)|)_+$$

Semi-supervised SVMs

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

Hinge loss on unlabeled data

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



$$f(x_i)$$

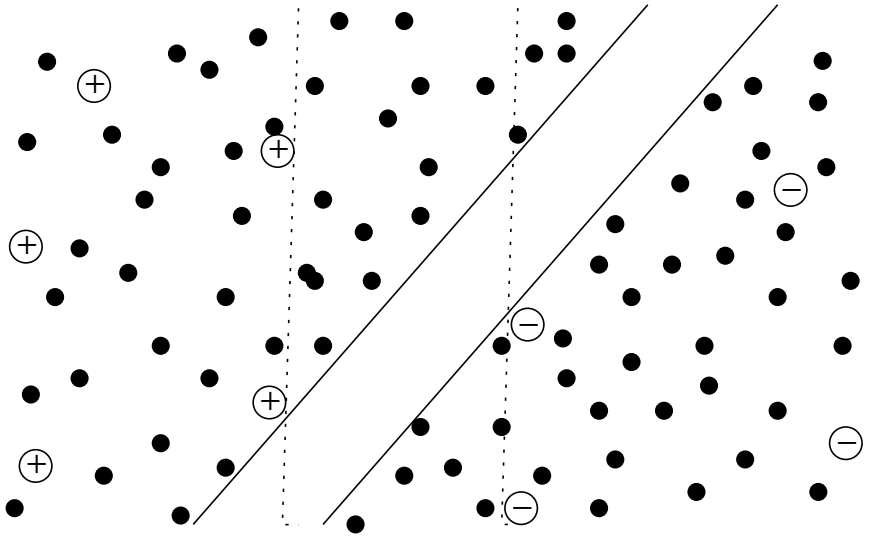
Prefers $f(x) \geq 1$ or $f(x) \leq -1$.

Avoiding unlabeled data

$$\min_f \sum_{i=1}^{l}(1-y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n}(1-|f(x_i)|)_+$$

The third term prefers unlabeled points outside the margin.

- Decision boundary $f = 0$ avoids dense regions.



Semi-supervised SVM algorithm

Algorithm: Semi-supervised SVM

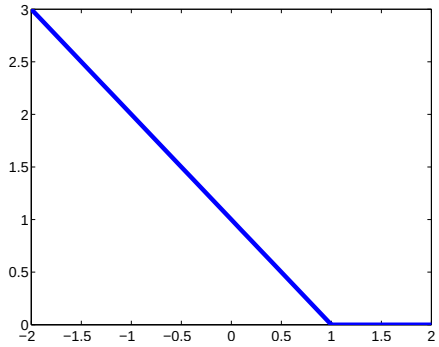
1. Input: kernel K, weights λ_1, λ_2, (X_l, Y_l), X_u
2. Solve the optimization problem for $f(x) = h(x) + b, h(x) \in \mathcal{H}_K$

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

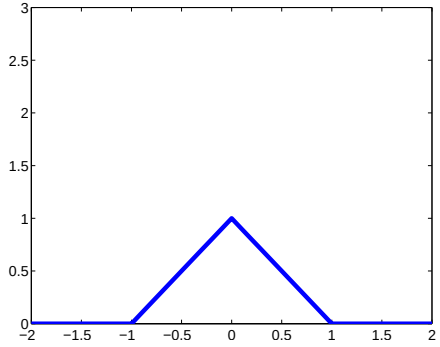
3. Classify a new test point x by $\text{sign}(f(x))$

The optimization challenge

SVM objective is convex:



Semi-supervised SVM objective is not convex:



Finding a solution for semi-supervised SVM is difficult.

- Heuristics, approximations, local optima

Pros and Cons of S3VMs

Pros:

- Applicable wherever SVMs are applicable
- Clear mathematical framework

Cons:

- Avoiding unlabeled dense region may not be the right assumption
- Optimization difficult, (currently) slow

4. Graph-Based Semi-Supervised Learning

Example: text classification

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

	d_1	d_3	d_4	d_2
asteroid	•	•		
bright	•	•		
comet		•		
year				
zodiac				
⋮				
airport				
bike				
camp			•	
yellowstone			•	•
zion				•

When labeled data alone fails

No overlapping words!

	d_1	d_3	d_4	d_2
asteroid	●			
bright	●			
comet				
year				
zodiac		●		
⋮				
airport			●	
bike			●	
camp				
yellowstone				●
zion				●

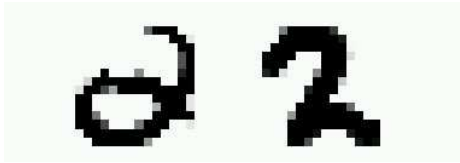
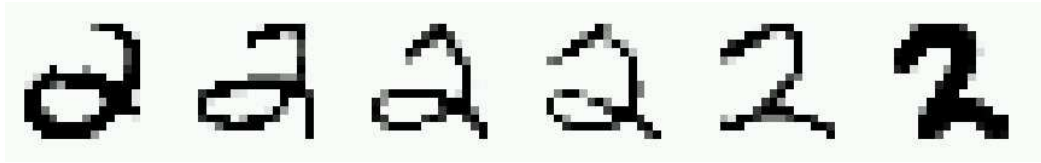
Unlabeled data: stepping stones

Labels propagate via similar unlabeled articles.

	d_1	d_5	d_6	d_7	d_3	d_4	d_8	d_9	d_2
asteroid	●								
bright	●	●							
comet		●	●						
year			●	●					
zodiac				●	●				
⋮									
airport						●			
bike						●	●		
camp							●	●	
yellowstone								●	●
zion									●

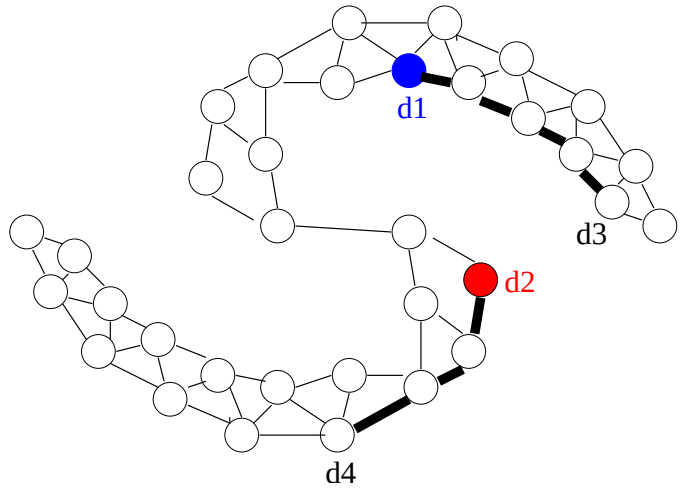
Another example

Handwritten digits recognition with pixel-wise Euclidean distance

	
not similar	'indirectly' similar with stepping stones

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



An example graph

A graph for person identification: time, color, face edges.

image 4005

neighbor 1: time edge

neighbor 2: color edge

neighbor 3: color edge

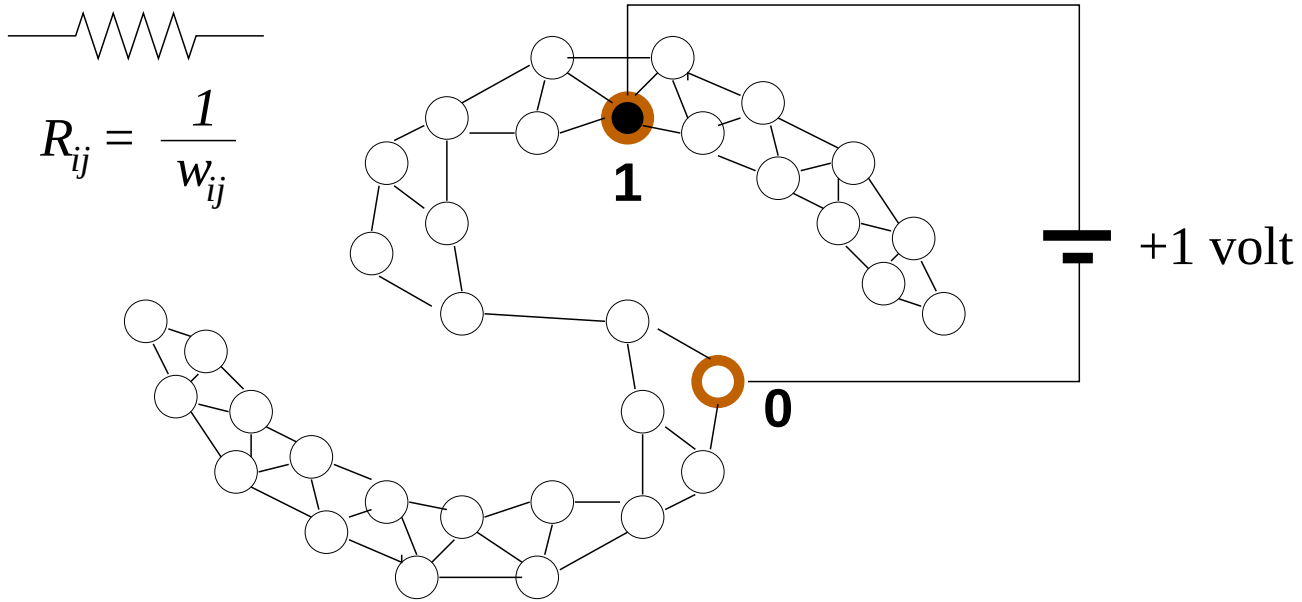
neighbor 4: color edge

neighbor 5: face edge

An electric network interpretation

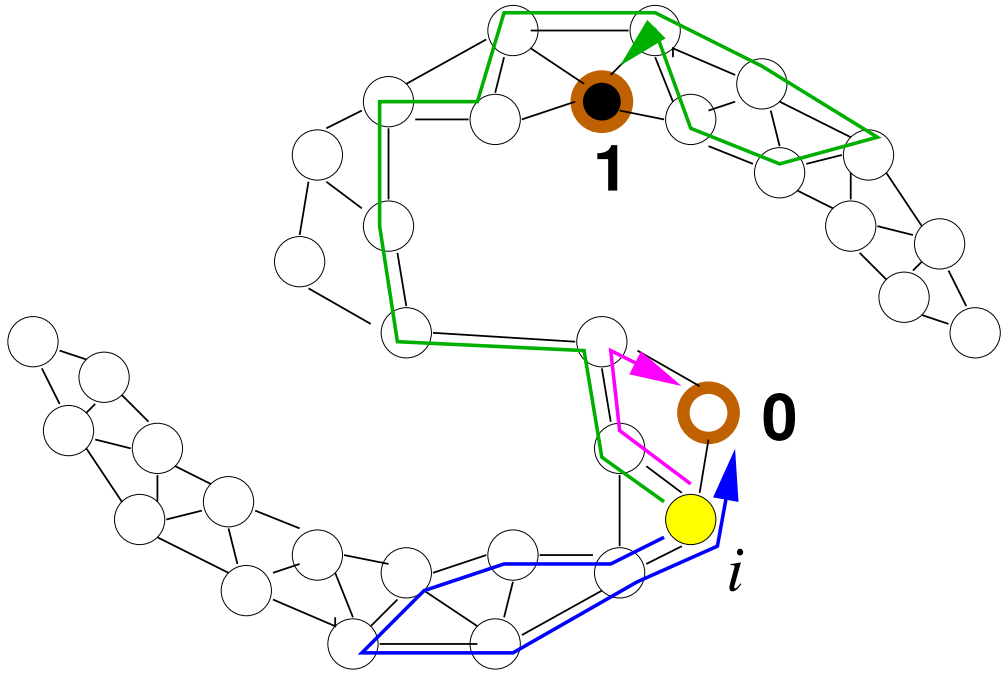
- Edges are resistors with conductance w_{ij}
- 1 volt battery connects to labeled points $y = 0, 1$
- The voltage at the nodes is the harmonic function f

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 $f = Pr(\text{hit label 1}|\text{start from } i)$



The harmonic function

The harmonic function f satisfies

- $f(x_i) = y_i$ for $i = 1 \ldots l$
- f minimizes the energy

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

- average of neighbors $f(x_i) = \frac{\sum_{j \sim i} w_{ij} f(x_j)}{\sum_{j \sim i} w_{ij}}, \forall x_i \in X_u$

We compute f using the graph Laplacian.

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

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

Harmonic solution with Laplacian

The harmonic solution minimizes energy subject to the given labels

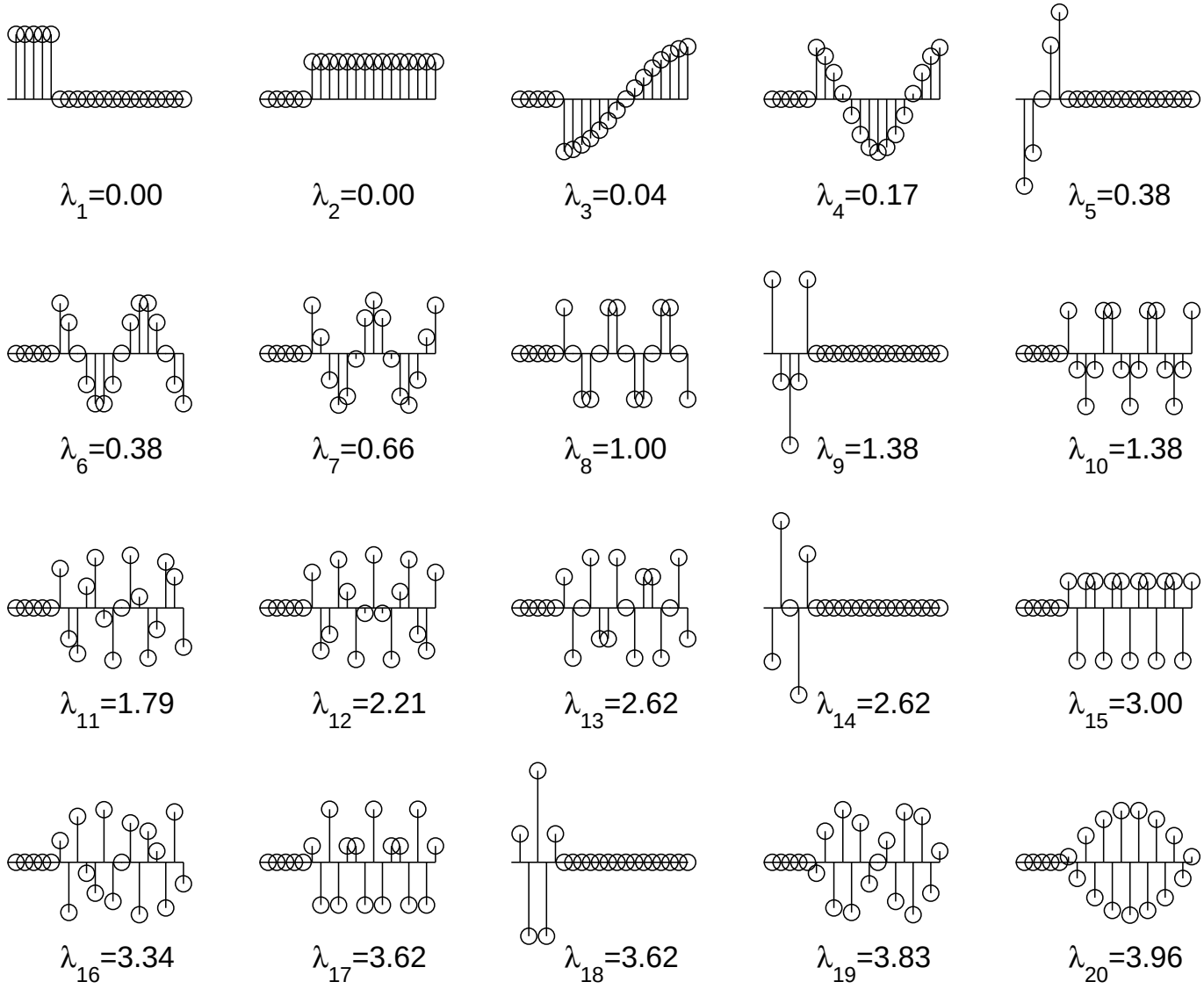
$$\min_f \infty \sum_{i=1}^{l} (f(x_i) - y_i)^2 + f^\top \Delta f$$

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

Harmonic solution

$$f_u = -{\Delta_{uu}}^{-1} \Delta_{ul} Y_l$$

Graph spectrum $\Delta = \sum_{i=1}^{n} \lambda_i \phi_i \phi_i^\top$



Relation to spectral clustering

f can be decomposed as $f = \sum_i \alpha_i \phi_i$

$$f^\top \Delta f = \sum_i \alpha_i^2 \lambda_i$$

- f wants basis ϕ_i with small λ
- ϕ's with small λ's correspond to clusters
- f is a balance between spectral clustering and obeying labeled data

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

Manifold regularization

Manifold regularization solves the two issues

- Allows but penalizes $f(X_l) \neq Y_i$ using hinge loss
- Automatically applies to new test data
  - Defines function in kernel K induced RKHS: $f(x) = h(x) + b, h(x) \in \mathcal{H}_K$
- Still prefers low energy $f_{1:n}^\top \Delta f_{1:n}$

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|_{\mathcal{H}_K}^2 + \lambda_2 f_{1:n}^\top \Delta f_{1:n}$$

Manifold regularization algorithm

Algorithm: Manifold regularization algorithm

1. Input: kernel K, weights λ_1, λ_2, (X_l, Y_l), X_u
2. Construct similarity graph W from X_l, Xu, compute graph Laplacian Δ
3. Solve the optimization problem for $f(x) = h(x) + b, h(x) \in \mathcal{H}_K$

$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 f_{1:n}^\top \Delta f_{1:n}$$

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

Pros and Cons of graph-based method

Pros:

- Clear mathematical framework
- Performance is good if the graph is good

Cons:

- Performance is bad if the graph is bad
- How to construct a good graph?

Summary

Which method is the best?

They make different assumptions

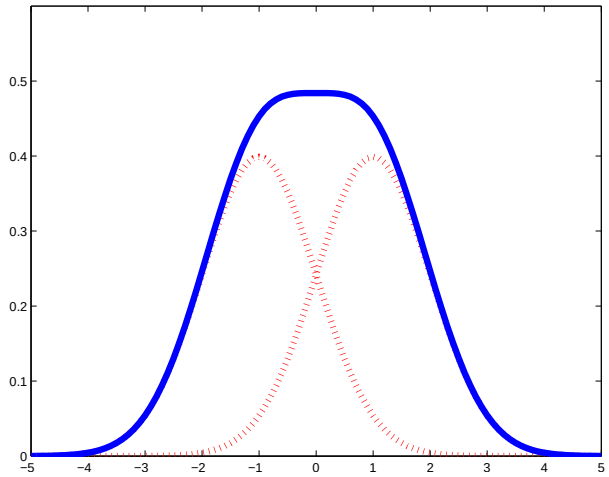
- co-training: feature split, conditional independence
- generative model: the particular model
- S3VM: avoid dense regions
- graph-based: edge weights

On-going research.

Does unlabeled data always help?

NO.

- Use model assumptions to make up for the lack of labeled data
- Assumptions may be wrong



References

1. Xiaojin Zhu (2005). *Semi-supervised learning literature survey*. TR-1530. University of Wisconsin-Madison Department of Computer Science.

2. Olivier Chapelle, Alexander Zien, Bernhard Schölkopf (Eds.). (2006). *Semi-supervised learning*. MIT Press.

3. Matthias Seeger (2001). *Learning with labeled and unlabeled data*. Technical Report. University of Edinburgh.