Tutorial on Semi-Supervised Learning

Xiaojin Zhu

Department of Computer Sciences
University of Wisconsin, Madison, USA

Theory and Practice of Computational Learning
Chicago, 2009

New book

Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.

Outline

1. Part I
   - What is SSL?
   - Mixture Models
   - Co-training and Multiview Algorithms
   - Manifold Regularization and Graph-Based Algorithms
   - S3VMs and Entropy Regularization

2. Part II
   - Theory of SSL
   - Online SSL
   - Multimanifold SSL
   - Human SSL

Outline

Outline

1. Part I
   - What is SSL?
   - Mixture Models
   - Co-training and Multiview Algorithms
   - Manifold Regularization and Graph-Based Algorithms
   - S3VMs and Entropy Regularization

2. Part II
   - Theory of SSL
   - Online SSL
   - Multimanifold SSL
   - Human SSL

What is Semi-Supervised Learning?

Learning from both labeled and unlabeled data. Examples:

- Semi-supervised classification: training data l labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, often $u \gg l$.
  Goal: better classifier f than from labeled data alone.

What is Semi-Supervised Learning?

Learning from both labeled and unlabeled data. Examples:

- Semi-supervised classification: training data l labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, often $u \gg l$. **Goal**: better classifier f than from labeled data alone.
- Constrained clustering: unlabeled instances $\{x_i\}_{j=1}^{n}$, and "supervised information", e.g., must-links, cannot-links. **Goal**: better clustering than from unlabeled data alone.

What is Semi-Supervised Learning?

Learning from both labeled and unlabeled data. Examples:

- Semi-supervised classification: training data l labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, often $u \gg l$. **Goal**: better classifier f than from labeled data alone.
- Constrained clustering: unlabeled instances $\{x_i\}_{j=1}^{n}$, and "supervised information", e.g., must-links, cannot-links. **Goal**: better clustering than from unlabeled data alone.

We will mainly discuss semi-supervised classification.

Motivations

Machine learning

Promise: better performance for free...

- labeled data can be hard to get
  - labels may require human experts
  - labels may require special devices
- unlabeled data is often cheap in large quantity

Motivations

Machine learning

Promise: better performance for free...

- labeled data can be hard to get
  - labels may require human experts
  - labels may require special devices
- unlabeled data is often cheap in large quantity

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

Example of hard-to-get labels

Task: speech analysis

- Switchboard dataset
- telephone conversation transcription
- 400 hours annotation time for each hour of speech

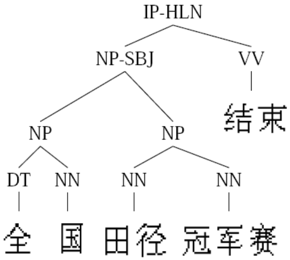
film ⇒ `f ih_n uh_gl_n m`

be all ⇒ `bcl b iy iy_tr ao_tr ao l_dl`

Another example of hard-to-get labels

Task: natural language parsing

- Penn Chinese Treebank
- 2 years for 4000 sentences



"The National Track and Field Championship has finished."

Notations

- 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

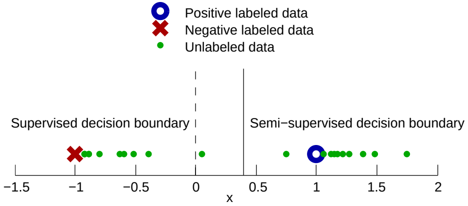
Semi-supervised vs. transductive learning

- Inductive semi-supervised learning: Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, learn $f : \mathcal{X} \mapsto \mathcal{Y}$ so that f is expected to be a good predictor on future data, beyond $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

Semi-supervised vs. transductive learning

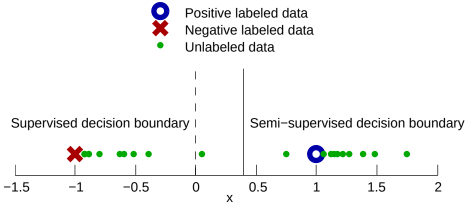
- Inductive semi-supervised learning: Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, learn $f : \mathcal{X} \mapsto \mathcal{Y}$ so that f is expected to be a good predictor on future data, beyond $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
- Transductive learning: Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, learn $f : \mathcal{X}^{l+u} \mapsto \mathcal{Y}^{l+u}$ so that f is expected to be a good predictor on the unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Note f is defined only on the given training sample, and is not required to make predictions outside them.

How can unlabeled data ever help?



- assuming each class is a coherent group (e.g. Gaussian)
- with and without unlabeled data: decision boundary shift

How can unlabeled data ever help?



- assuming each class is a coherent group (e.g. Gaussian)
- with and without unlabeled data: decision boundary shift

This is only one of many ways to use unlabeled data.

Self-training algorithm

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 $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

Self-training algorithm

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 algorithm

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

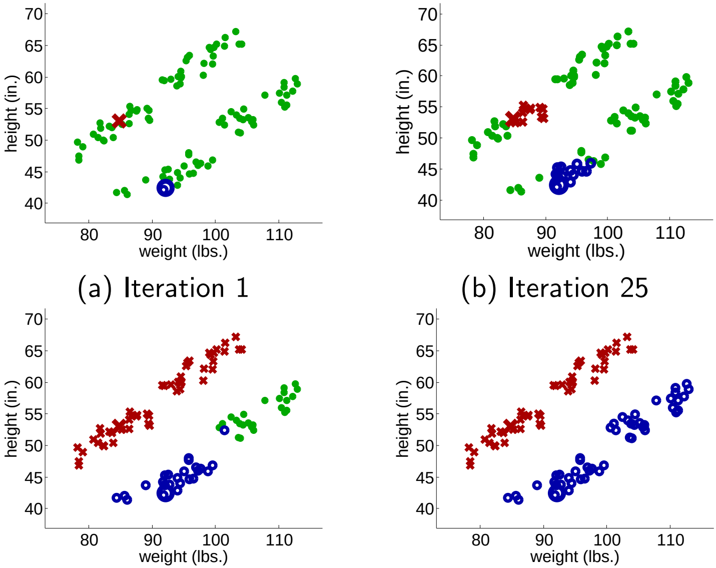
Self-training example: Propagating 1-Nearest-Neighbor

An instance of self-training.

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

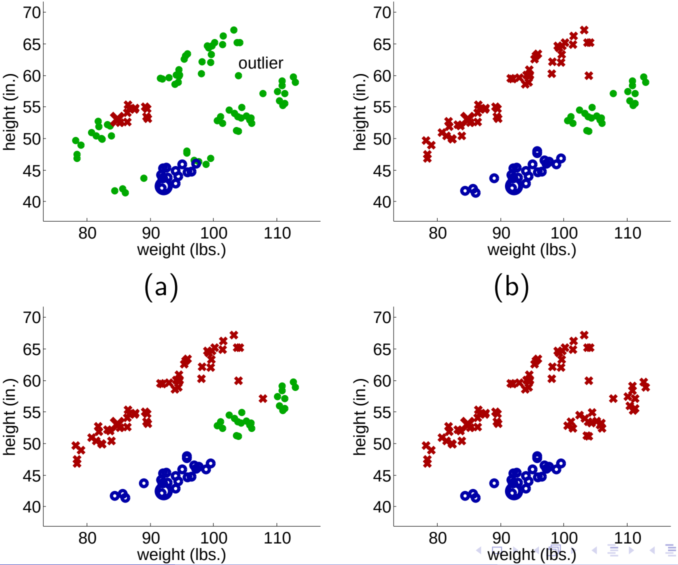
(b) Iteration 25

(c) Iteration 74

(d) Final labeling of all instances

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

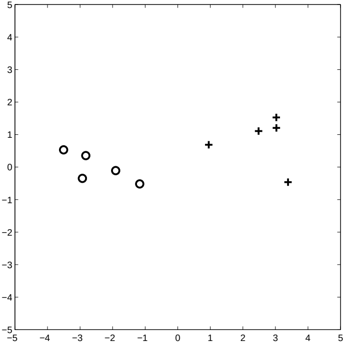
But with a single outlier...



Outline

A simple example of generative models

Labeled data (X_l, Y_l):



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

A simple example of generative models

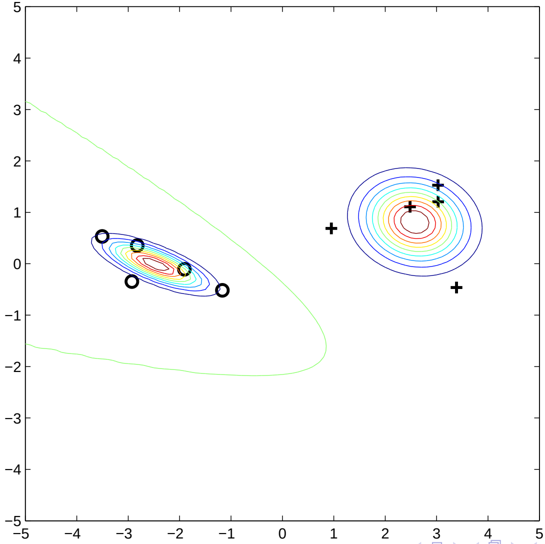
Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$
The GMM:

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

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

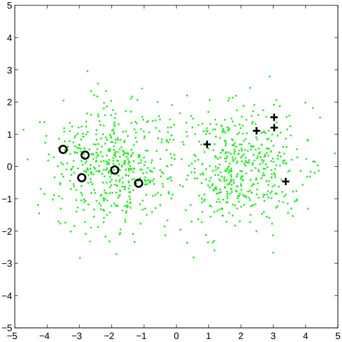
A simple example of generative models

The most likely model, and its decision boundary:



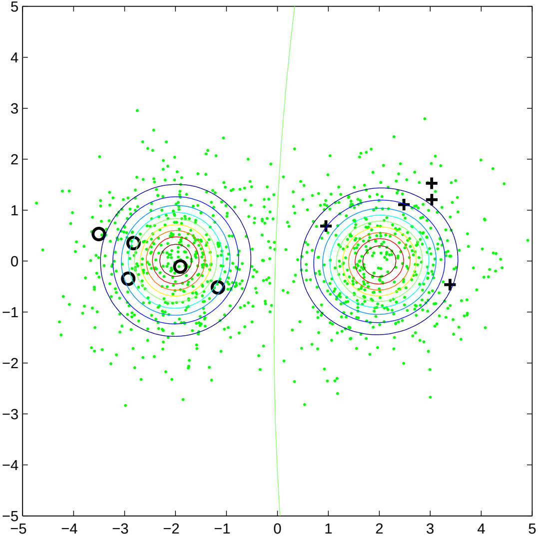
A simple example of generative models

Adding unlabeled data:



A simple example of generative models

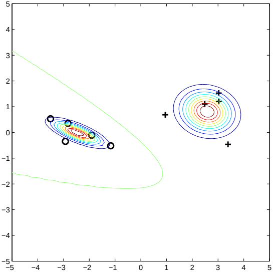
With unlabeled data, the most likely model and its decision boundary:



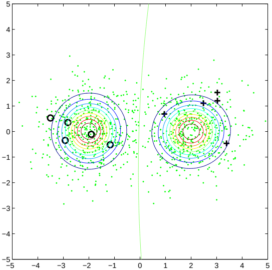
A simple example of generative models

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

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

- joint and marginal likelihood

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

Generative model for semi-supervised learning

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

Generative model for semi-supervised learning

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
- common mixture models used in semi-supervised learning:
  - Mixture of Gaussian distributions (GMM) – image classification
  - Mixture of multinomial distributions (Naïve Bayes) – text categorization
  - Hidden Markov Models (HMM) – speech recognition
- Learning via the Expectation-Maximization (EM) algorithm (Baum-Welch)

Case study: GMM

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)

Case study: GMM

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

The EM algorithm for GMM

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:

The EM algorithm for GMM

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

The EM algorithm for GMM

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

The EM algorithm for GMM

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

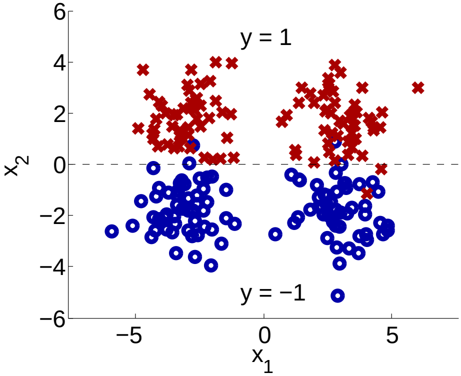
Can be viewed as a special form of self-training.

The assumption of mixture models

- **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.

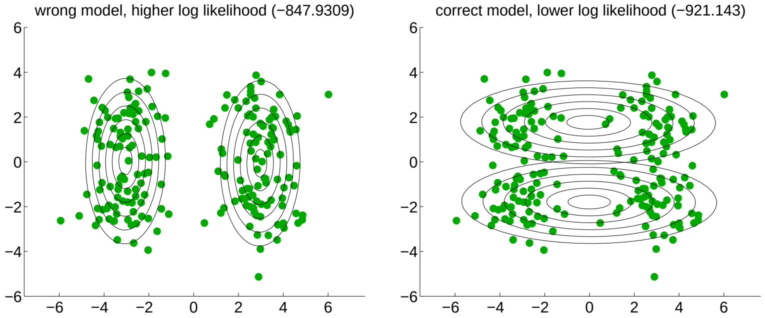
The assumption of mixture models

- **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:

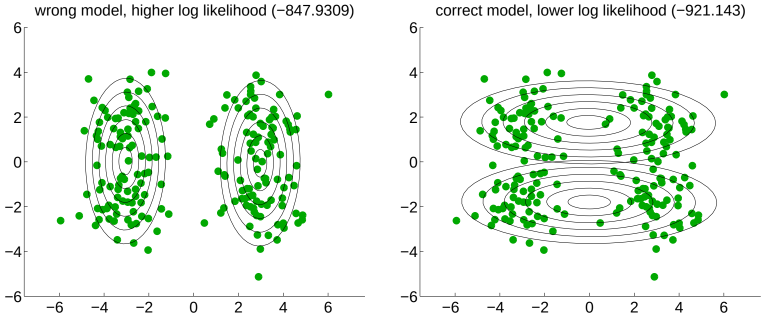


For example, classifying text by topic vs. by genre.

The assumption of mixture models



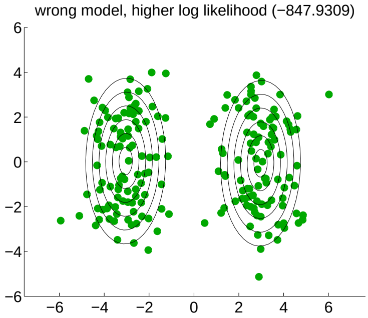
The assumption of mixture models

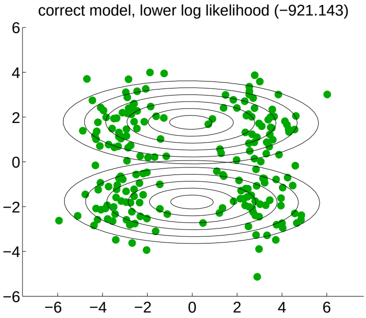


Heuristics to lessen the danger

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

The assumption of mixture models



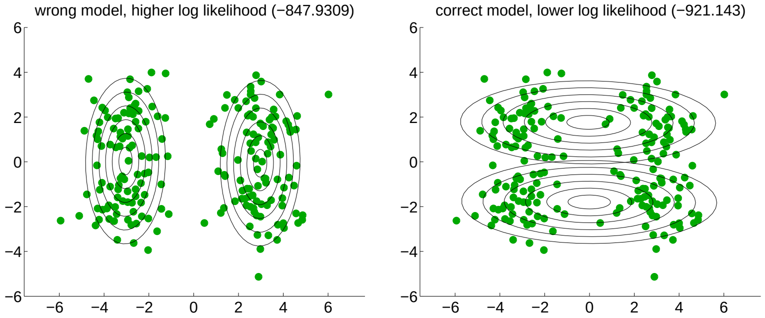


Heuristics to lessen the danger

- Carefully construct the generative model, e.g., multiple Gaussian distributions per class
- Down-weight the unlabeled data ($\lambda < 1$)

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

The assumption of mixture models



Heuristics to lessen the danger

- Carefully construct the generative model, e.g., multiple Gaussian distributions per class
- Down-weight the unlabeled data ($\lambda < 1$)

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

Other dangers: identifiability, EM local optima

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 $\mathcal{L}$

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 $\mathcal{A}$.

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:

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: $f_S = \mathcal{L}(S)$.

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 $y_{l+1}, \ldots, y_{l+u}$.

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.

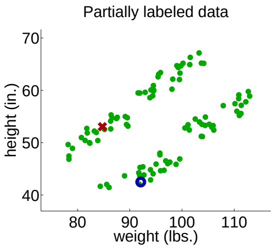
Output: labels on unlabeled data $y_{l+1}, \ldots, y_{l+u}$.

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.

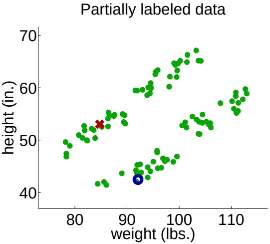
single linkage

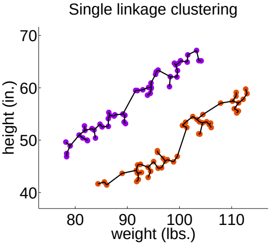


Cluster-and-label: now it works, now it doesn't

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

single linkage

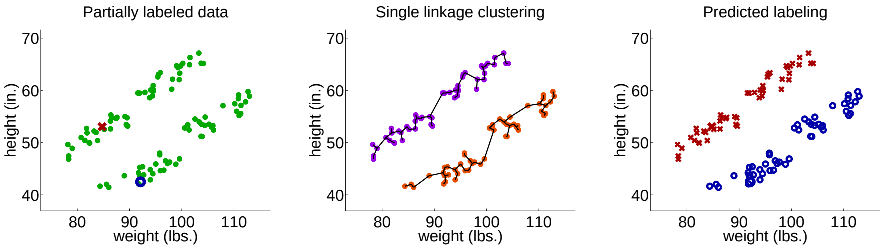




Cluster-and-label: now it works, now it doesn't

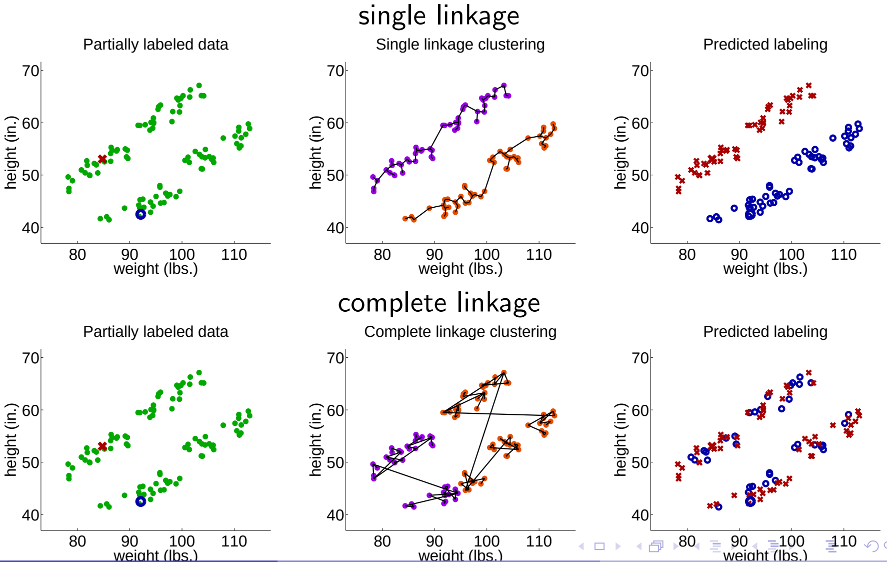
Example: $\mathcal{A}$=Hierarchical Clustering, $\mathcal{L}$=majority vote.

single linkage



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

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



Outline

1. Part I
   - What is SSL?
   - Mixture Models
   - Co-training and Multiview Algorithms
   - Manifold Regularization and Graph-Based Algorithms
   - S3VMs and Entropy Regularization
2. Part II
   - Theory of SSL
   - Online SSL
   - Multimanifold SSL
   - Human SSL

Two Views of an Instance

Example: named entity classification `Person` (Mr. Washington) or `Location` (Washington State)

Two Views of an Instance

Example: named entity classification `Person` (Mr. Washington) or `Location` (Washington State)

instance 1: ... headquartered in (Washington State) ...
instance 2: ... (Mr. Washington), the vice president of ...

- a named entity has two views (subset of features) $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$
- the words of the entity is $\mathbf{x}^{(1)}$
- the context is $\mathbf{x}^{(2)}$

Quiz

instance 1:	. . . headquartered in (Washington State)^L . . .
instance 2:	. . . (Mr. Washington)^P, the vice president of . . .
test:	. . . (Robert Jordan), a partner at . . .
test:	. . . flew to (China) . . .

Quiz

With more unlabeled data

instance 1:	. . . headquartered in (Washington State)^L . . .
instance 2:	. . . (Mr. Washington)^P, the vice president of . . .
instance 3:	. . . headquartered in (Kazakhstan) . . .
instance 4:	. . . flew to (Kazakhstan) . . .
instance 5:	. . . (Mr. Smith), a partner at Steptoe & Johnson . . .
test:	. . . (Robert Jordan), a partner at . . .
test:	. . . flew to (China) . . .

Co-training algorithm

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.

Co-training algorithm

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:

Co-training algorithm

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.

Co-training algorithm

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.

Co-training algorithm

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.

Like self-training, but with two classifiers teaching each other.

Co-training assumptions

Assumptions

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

Co-training assumptions

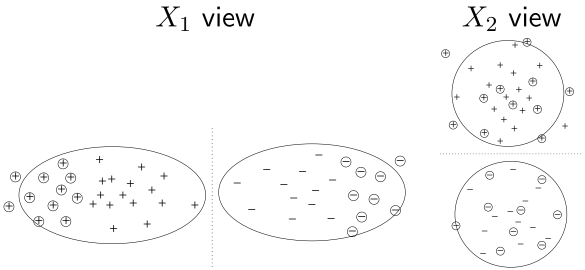
Assumptions

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

Co-training assumptions

Assumptions

- 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



Multiview learning

Extends co-training.

- Loss Function: $c(\mathbf{x}, y, f(\mathbf{x})) \in [0, \infty)$. For example,
  - squared loss $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - 0/1 loss $c(\mathbf{x}, y, f(\mathbf{x})) = 1$ if $y \neq f(\mathbf{x})$, and 0 otherwise.

Multiview learning

Extends co-training.

- Loss Function: $c(\mathbf{x}, y, f(\mathbf{x})) \in [0, \infty)$. For example,
  - squared loss $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - 0/1 loss $c(\mathbf{x}, y, f(\mathbf{x})) = 1$ if $y \neq f(\mathbf{x})$, and 0 otherwise.
- Empirical risk: $\hat{R}(f) = \frac{1}{l} \sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$

Multiview learning

Extends co-training.

- Loss Function: $c(\mathbf{x}, y, f(\mathbf{x})) \in [0, \infty)$. For example,
  - squared loss $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - 0/1 loss $c(\mathbf{x}, y, f(\mathbf{x})) = 1$ if $y \neq f(\mathbf{x})$, and 0 otherwise.
- Empirical risk: $\hat{R}(f) = \frac{1}{l} \sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$
- Regularizer: $\Omega(f)$, e.g., $\|f\|^2$

Multiview learning

Extends co-training.

- Loss Function: $c(\mathbf{x}, y, f(\mathbf{x})) \in [0, \infty)$. For example,
  - squared loss $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - 0/1 loss $c(\mathbf{x}, y, f(\mathbf{x})) = 1$ if $y \neq f(\mathbf{x})$, and 0 otherwise.
- Empirical risk: $\hat{R}(f) = \frac{1}{l} \sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$
- Regularizer: $\Omega(f)$, e.g., $\|f\|^2$
- Regularized Risk Minimization $f^* = \mathrm{argmin}_{f \in \mathcal{F}} \hat{R}(f) + \lambda \Omega(f)$

Multiview learning

A special regularizer $\Omega(f)$ defined on unlabeled data, to encourage agreement among multiple learners:

$$\begin{aligned} \underset{f_1,\ldots,f_k}{\operatorname{argmin}} \quad & \sum_{v=1}^{k}\left(\sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f_v(\mathbf{x}_i)) + \lambda_1 \Omega_{SL}(f_v)\right) \\ & +\lambda_2 \sum_{u,v=1}^{k} \sum_{i=l+1}^{l+u} c(\mathbf{x}_i, f_u(\mathbf{x}_i), f_v(\mathbf{x}_i)) \end{aligned}$$

Outline

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				•

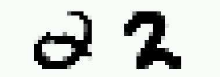
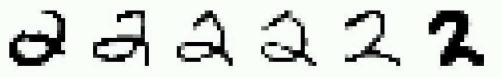
Unlabeled data as stepping stones

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

Graph-based semi-supervised learning

- Nodes: $X_l \cup X_u$
- Edges: similarity weights computed from features, e.g.,

Graph-based semi-supervised learning

- Nodes: $X_l \cup X_u$
- Edges: similarity weights computed from features, e.g.,
  - k-nearest-neighbor graph, unweighted (0, 1 weights)

Graph-based semi-supervised learning

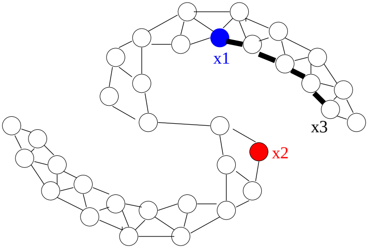
- Nodes: $X_l \cup X_u$
- Edges: similarity weights computed from features, e.g.,
  - k-nearest-neighbor graph, unweighted (0, 1 weights)
  - fully connected graph, weight decays with distance $w = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$

Graph-based semi-supervised learning

- 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 $w = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$
  - ϵ-radius graph

Graph-based semi-supervised learning

- 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 $w = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$
  - ϵ-radius graph
- Assumption Instances connected by heavy edge tend to have the same label.



The mincut algorithm

- Fix Y_l, find $Y_u \in \{0,1\}^{n-l}$ to minimize $\sum_{ij} w_{ij}|y_i - y_j|$.

The mincut algorithm

- 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

$$\min_{Y\in\{0,1\}^n} \infty \sum_{i=1}^{l}(y_i - Y_{li})^2 + \sum_{ij} w_{ij}(y_i - y_j)^2$$

The mincut algorithm

- 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

$$\min_{Y\in\{0,1\}^n} \; \infty \sum_{i=1}^{l}(y_i - Y_{li})^2 + \sum_{ij} w_{ij}(y_i - y_j)^2$$

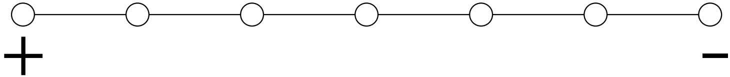
- Combinatorial problem, but has polynomial time solution.

The mincut algorithm

- 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

$$\min_{Y \in \{0,1\}^n} \infty \sum_{i=1}^{l} (y_i - Y_{li})^2 + \sum_{ij} w_{ij}(y_i - y_j)^2$$

- Combinatorial problem, but has polynomial time solution.
- Mincut computes the modes of a discrete Markov random field, but there might be multiple modes



The harmonic function

Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies

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

The harmonic function

Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies

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

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

The harmonic function

Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies

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

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

- the mean of a Gaussian random field

The harmonic function

Relaxing discrete labels to continuous values in $\mathbb{R}$, the harmonic function f satisfies

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

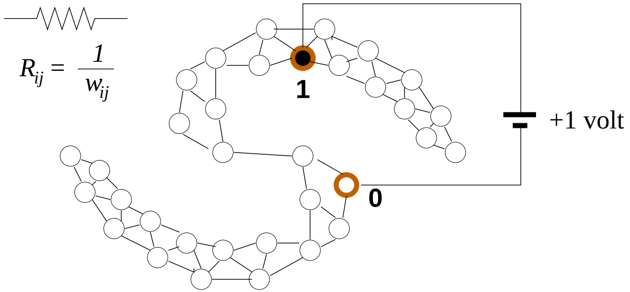
$$\sum_{i \sim j} w_{ij}(f(x_i) - f(x_j))^2$$

- 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

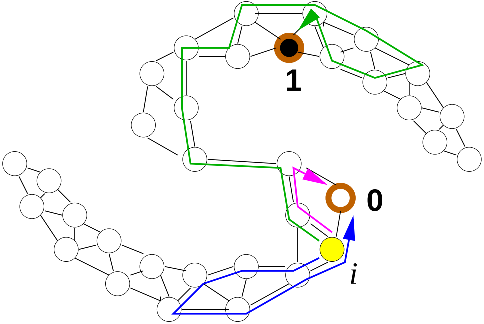
- Edges are resistors with conductance w_{ij}
- 1 volt battery connects to labeled points $y = 0, 1$
- The voltage at the nodes is the harmonic function f

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 iterative way to compute the harmonic function:

1. Initially, set $f(x_i) = y_i$ for $i = 1 \ldots l$, and $f(x_j)$ arbitrarily (e.g., 0) for $x_j \in X_u$.

An algorithm to compute harmonic function

One iterative way to compute the harmonic function:

1. Initially, set $f(x_i) = y_i$ for $i = 1 \ldots l$, and $f(x_j)$ arbitrarily (e.g., 0) for $x_j \in X_u$.
2. 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.

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

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

Harmonic solution with Laplacian

The harmonic solution minimizes energy subject to the given labels

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

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 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

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

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

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

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

Local and Global consistency

- Allow $f(X_l)$ to be different from Y_l, but penalize it

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

Manifold regularization

The graph-based algorithms so far are transductive. Manifold regularization is inductive.

- defines function in a RKHS: $f(x) = h(x) + b, h(x) \in \mathcal{H}_K$
- views the graph as a random sample of an underlying manifold
- regularizer prefers low energy $f_{1:n}^\top \Delta f_{1:n}$

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

Graph spectrum and SSL

Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i {\phi_i}^\top$

Graph spectrum and SSL

Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i {\phi_i}^\top$
- a graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$.

Graph spectrum and SSL

Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i \phi_i^\top$
- a graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$.
- the corresponding eigenvectors are constant on individual connected components, and zero elsewhere.

Graph spectrum and SSL

Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i \phi_i^\top$
- a graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$.
- the corresponding eigenvectors are constant on individual connected components, and zero elsewhere.
- any $\mathbf{f}$ on the graph can be represented as $\mathbf{f} = \sum_{i=1}^{l+u} a_i \phi_i$

Graph spectrum and SSL

Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i \phi_i^\top$
- a graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$.
- the corresponding eigenvectors are constant on individual connected components, and zero elsewhere.
- any $\mathbf{f}$ on the graph can be represented as $\mathbf{f} = \sum_{i=1}^{l+u} a_i \phi_i$
- graph regularizer $\mathbf{f}^\top L \mathbf{f} = \sum_{i=1}^{l+u} a_i^2 \lambda_i$

Graph spectrum and SSL

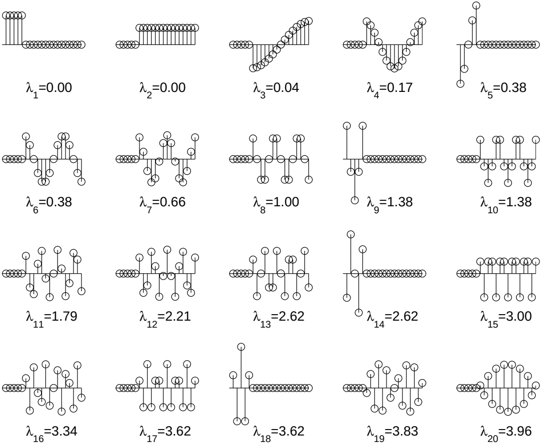
Assumption: labels are "smooth" on the graph, characterized by the graph spectrum (eigen-values/vectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$ of the Laplacian L):

- $L = \sum_{i=1}^{l+u} \lambda_i \phi_i {\phi_i}^\top$
- a graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$.
- the corresponding eigenvectors are constant on individual connected components, and zero elsewhere.
- any $\mathbf{f}$ on the graph can be represented as $\mathbf{f} = \sum_{i=1}^{l+u} a_i \phi_i$
- graph regularizer $\mathbf{f}^\top L \mathbf{f} = \sum_{i=1}^{l+u} a_i^2 \lambda_i$
- smooth function $\mathbf{f}$ uses smooth basis (those with small λ_i)

Example graph spectrum

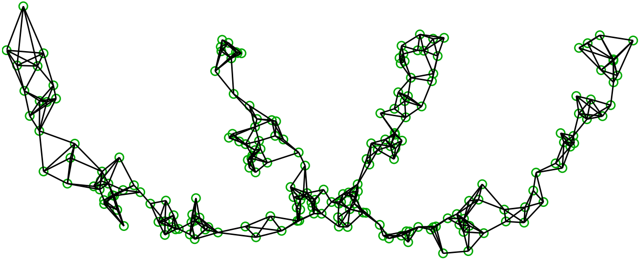
The graph

Eigenvalues and eigenvectors of the graph Laplacian



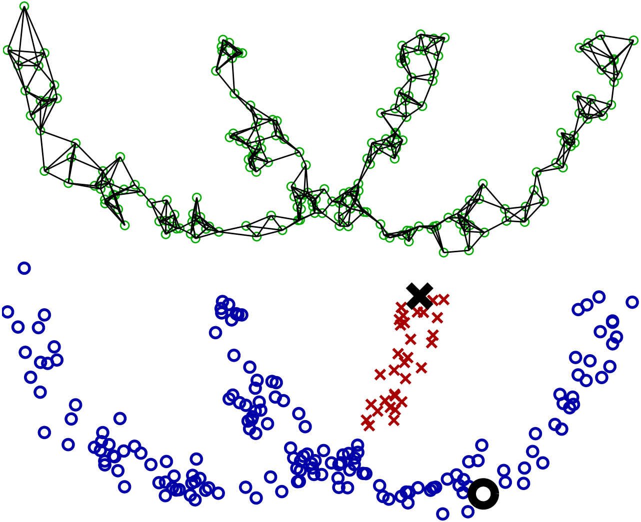
When the graph assumption is wrong

"colliding two moons"



When the graph assumption is wrong

"colliding two moons"

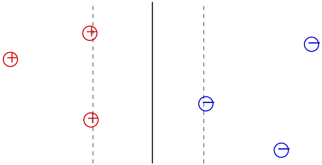


Outline

1. Part I
   - What is SSL?
   - Mixture Models
   - Co-training and Multiview Algorithms
   - Manifold Regularization and Graph-Based Algorithms
   - S3VMs and Entropy Regularization
2. Part II
   - Theory of SSL
   - Online SSL
   - Multimanifold SSL
   - Human SSL

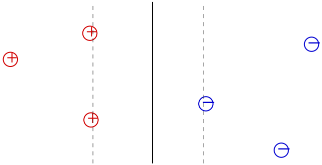
Semi-supervised Support Vector Machines

SVMs

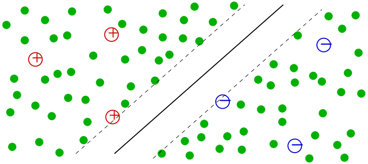


Semi-supervised Support Vector Machines

SVMs



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



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

Standard soft margin SVMs

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

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

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

$$\xi_i \geq 0$$

Standard soft margin SVMs

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

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

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

$$\xi_i \geq 0$$

Equivalent to

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

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

The S3VM objective function

To incorporate unlabeled points,

- assign putative labels sign$(f(x))$ to $x \in X_u$

The S3VM objective function

To incorporate unlabeled points,

- assign putative labels $\text{sign}(f(x))$ to $x \in X_u$
- the hinge loss on unlabeled points becomes

$$(1 - \text{sign}(f(x))f(x_i))_+ = (1 - |f(x_i)|)_+$$

The S3VM objective function

To incorporate unlabeled points,

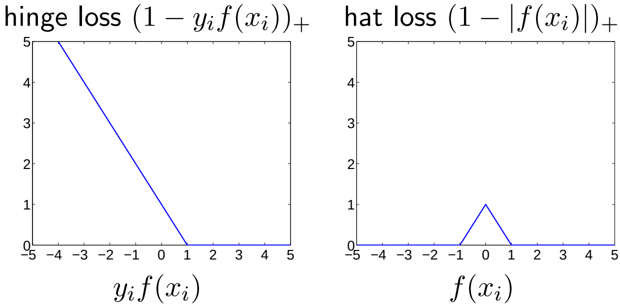
- assign putative labels $\text{sign}(f(x))$ to $x \in X_u$
- the hinge loss on unlabeled points becomes

$$(1 - \text{sign}(f(x))f(x_i))_+ = (1 - |f(x_i)|)_+$$

S3VM objective:

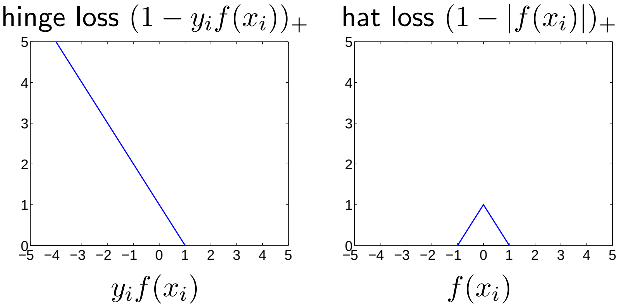
$$\min_f \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n} (1 - |f(x_i)|)_+$$

The hat loss on unlabeled data



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

The hat loss on unlabeled data

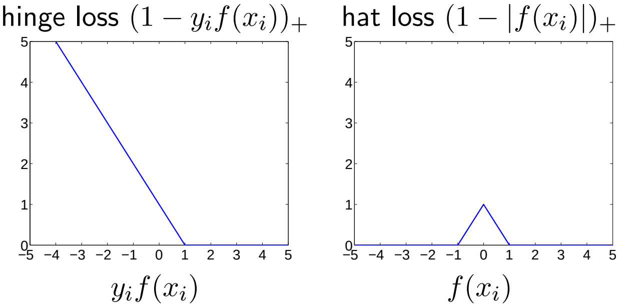


Prefers $f(x) \geq 1$ or $f(x) \leq -1$, i.e., unlabeled instance away from decision boundary $f(x) = 0$.

The class balancing constraint

- often unbalanced – most points classified into one class.

The hat loss on unlabeled data

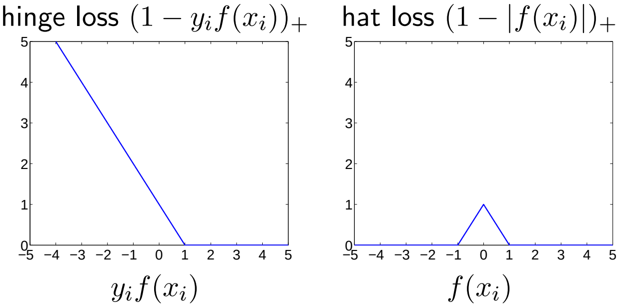


Prefers $f(x) \geq 1$ or $f(x) \leq -1$, i.e., unlabeled instance away from decision boundary $f(x) = 0$.

The class balancing constraint

- often unbalanced – most points classified into one class.
- Heuristic class balance: $\frac{1}{n-l} \sum_{i=l+1}^{n} y_i = \frac{1}{l} \sum_{i=1}^{l} y_i$.

The hat loss on unlabeled data



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

The class balancing constraint

- often unbalanced – most points classified into one 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: $\frac{1}{n-l} \sum_{i=l+1}^{n} f(x_i) = \frac{1}{l} \sum_{i=1}^{l} y_i$.

The S3VM algorithm

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

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

The S3VM algorithm

$$\begin{aligned} \min_f \quad & \sum_{i=1}^{l}(1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \lambda_2 \sum_{i=l+1}^{n}(1 - |f(x_i)|)_+ \\ \text{s.t.} \quad & \frac{1}{n-l}\sum_{i=l+1}^{n} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i \end{aligned}$$

Computational difficulty

- SVM objective is convex
- Semi-supervised SVM objective is non-convex

The S3VM algorithm

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

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

Computational difficulty

- SVM objective is convex
- Semi-supervised SVM objective is non-convex
- Optimization approaches: SVM^{light}, ∇S3VM, continuation S3VM, deterministic annealing, CCCP, Branch and Bound, SDP convex relaxation, etc.

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1 + \exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1+\exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- (conditional) log likelihood $\sum_{i=1}^{l} \log p(y_i|\mathbf{x}_i, \mathbf{w}, b)$

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1 + \exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- (conditional) log likelihood $\sum_{i=1}^{l} \log p(y_i|\mathbf{x}_i, \mathbf{w}, b)$
- prior $\mathbf{w} \sim \mathcal{N}(0, I/(2\lambda))$

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1+\exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- (conditional) log likelihood $\sum_{i=1}^{l} \log p(y_i|\mathbf{x}_i, \mathbf{w}, b)$
- prior $\mathbf{w} \sim \mathcal{N}(0, I/(2\lambda))$
- MAP training $\max_{\mathbf{w},b} \sum_{i=1}^{l} \log\left(1/\left(1+\exp(-y_i f(\mathbf{x}_i))\right)\right) - \lambda\|\mathbf{w}\|^2$

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1+\exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- (conditional) log likelihood $\sum_{i=1}^{l} \log p(y_i|\mathbf{x}_i, \mathbf{w}, b)$
- prior $\mathbf{w} \sim \mathcal{N}(0, I/(2\lambda))$
- MAP training $\max_{\mathbf{w},b} \sum_{i=1}^{l} \log\left(1/\left(1+\exp(-y_i f(\mathbf{x}_i))\right)\right) - \lambda\|\mathbf{w}\|^2$
- logistic loss $c(\mathbf{x}, y, f(\mathbf{x})) = \log\left(1+\exp(-yf(\mathbf{x}))\right)$

Logistic regression

The probabilistic counter part of SVMs.

- $p(y|\mathbf{x}) = 1/\left(1 + \exp(-yf(\mathbf{x}))\right)$ where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- (conditional) log likelihood $\sum_{i=1}^{l} \log p(y_i|\mathbf{x}_i, \mathbf{w}, b)$
- prior $\mathbf{w} \sim \mathcal{N}(0, I/(2\lambda))$
- MAP training $\max_{\mathbf{w},b} \sum_{i=1}^{l} \log\left(1/\left(1 + \exp(-y_i f(\mathbf{x}_i))\right)\right) - \lambda\|\mathbf{w}\|^2$
- logistic loss $c(\mathbf{x}, y, f(\mathbf{x})) = \log\left(1 + \exp(-yf(\mathbf{x}))\right)$

Logistic regression does not use unlabeled data.

Entropy regularization

- Assumption: if the two classes are well-separated, then $p(y|x)$ on any unlabeled instance should be close to 0 or 1.

Entropy regularization

- Assumption: if the two classes are well-separated, then $p(y|x)$ on any unlabeled instance should be close to 0 or 1.
- Entropy $H(p) = -p \log p - (1-p) \log(1-p)$ should be small

Entropy regularization

- Assumption: if the two classes are well-separated, then $p(y|x)$ on any unlabeled instance should be close to 0 or 1.
- Entropy $H(p) = -p \log p - (1-p) \log(1-p)$ should be small
- entropy regularizer $\Omega(f) = \sum_{j=l+1}^{l+u} H(p(y=1|\mathbf{x}_j, \mathbf{w}, b))$

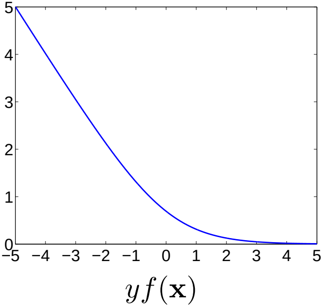
Entropy regularization

- Assumption: if the two classes are well-separated, then $p(y|x)$ on any unlabeled instance should be close to 0 or 1.
- Entropy $H(p) = -p \log p - (1-p)\log(1-p)$ should be small
- entropy regularizer $\Omega(f) = \sum_{j=l+1}^{l+u} H(p(y=1|\mathbf{x}_j, \mathbf{w}, b))$
- semi-supervised logistic regression

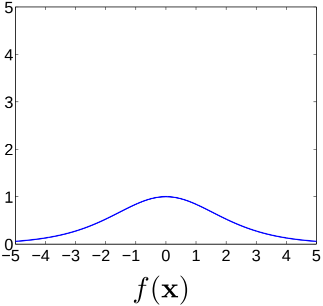
$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{l} \log\left(1+\exp(-y_i f(\mathbf{x}_i))\right) + \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} H\left(1/\left(1+\exp(-f(\mathbf{x}_j))\right)\right)$$

- The probabilistic counter part of S3VMs.

Entropy regularization

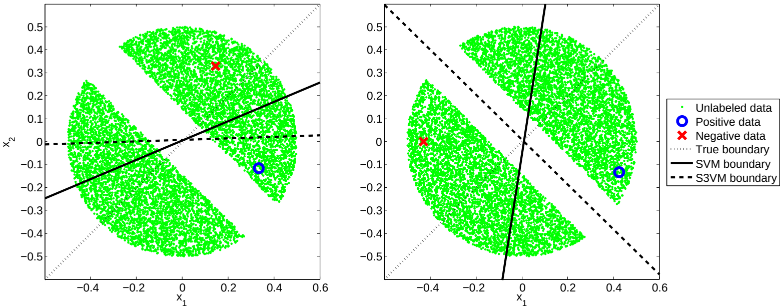


(a) the logistic loss



(b) the entropy regularizer

When the large margin assumption is wrong



S3VM in local minimum

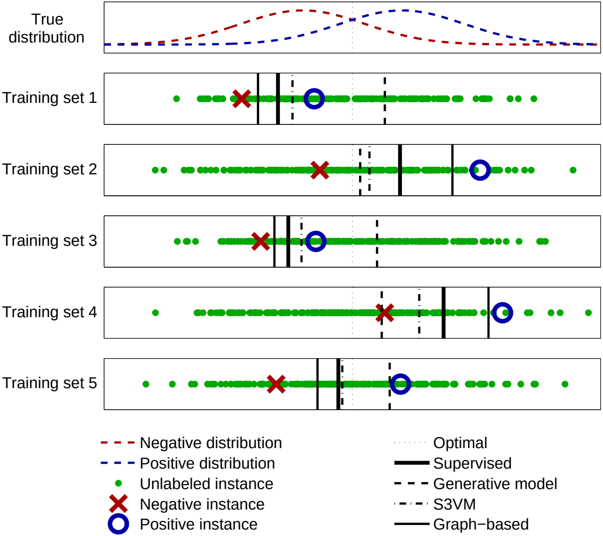
S3VM in wrong gap

SVM error: 0.26 ± 0.13
S3VM error: 0.34 ± 0.19

Outline

Outline

SSL does not always help



Wrong SSL assumption can make SSL worse than SL!

A computational theory for SSL

(Theoretic guarantee of Balcan & Blum)

Recall in supervised learning

- labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \stackrel{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, where P unknown

A computational theory for SSL

(Theoretic guarantee of Balcan & Blum)
Recall in supervised learning

- labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \stackrel{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, where P unknown
- function family $\mathcal{F}$

A computational theory for SSL

(Theoretic guarantee of Balcan & Blum)
Recall in supervised learning

- labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, where P unknown
- function family $\mathcal{F}$
- assume zero training sample error $\hat{e}(f) = \frac{1}{l}\sum_{i=1}^{l}(f(\mathbf{x}_i) \neq y_i)$

A computational theory for SSL

(Theoretic guarantee of Balcan & Blum)
Recall in supervised learning

- labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, where P unknown
- function family $\mathcal{F}$
- assume zero training sample error $\hat{e}(f) = \frac{1}{l}\sum_{i=1}^{l}(f(\mathbf{x}_i) \neq y_i)$
- can we say anything about its true error $e(f_{\mathcal{D}}) = \mathbb{E}_{(\mathbf{x},y)\sim P}\left[f_D(\mathbf{x}) \neq y\right]$?

A computational theory for SSL

(Theoretic guarantee of Balcan & Blum)
Recall in supervised learning

- labeled data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, where P unknown
- function family $\mathcal{F}$
- assume zero training sample error $\hat{e}(f) = \frac{1}{l}\sum_{i=1}^{l}(f(\mathbf{x}_i) \neq y_i)$
- can we say anything about its true error $e(f_{\mathcal{D}}) = \mathbb{E}_{(\mathbf{x},y)\sim P}\left[f_D(\mathbf{x}) \neq y\right]$?
- it turns out we can *bound* $e(f_{\mathcal{D}})$ without the knowledge of P.

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)
- $\{e(f_{\mathcal{D}}) > \epsilon\}$ is a random Boolean event

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)
- $\{e(f_{\mathcal{D}}) > \epsilon\}$ is a random Boolean event
- the probability of this event is $Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right)$. Goal: show that this probability is small

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)
- $\{e(f_{\mathcal{D}}) > \epsilon\}$ is a random Boolean event
- the probability of this event is $Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right)$. Goal: show that this probability is small

$$Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right) \quad \leq \quad Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}:\hat{e}(f)=0\}}\{e(f) > \epsilon\}\right)$$

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)
- $\{e(f_{\mathcal{D}}) > \epsilon\}$ is a random Boolean event
- the probability of this event is $Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right)$. Goal: show that this probability is small

$$\begin{aligned} Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right) &\leq Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}:\hat{e}(f)=0\}}\{e(f) > \epsilon\}\right) \\ &= Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}\}}\{\hat{e}(f) = 0, e(f) > \epsilon\}\right) \\ &= Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}:e(f)>\epsilon\}}\{\hat{e}(f) = 0\}\right) \end{aligned}$$

PAC bound for SL

- training error minimizer $f_{\mathcal{D}}$ is a random variable (of D)
- $\{e(f_{\mathcal{D}}) > \epsilon\}$ is a random Boolean event
- the probability of this event is $Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right)$. Goal: show that this probability is small

$$
\begin{aligned}
Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) > \epsilon\}\right) &\leq Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}:\hat{e}(f)=0\}}\{e(f) > \epsilon\}\right) \\
&= Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}\}}\{\hat{e}(f) = 0, e(f) > \epsilon\}\right) \\
&= Pr_{\mathcal{D}\sim P}\left(\cup_{\{f\in\mathcal{F}:e(f)>\epsilon\}}\{\hat{e}(f) = 0\}\right) \\
&\leq \sum_{\{f\in\mathcal{F}:e(f)>\epsilon\}} Pr_{\mathcal{D}\sim P}\left(\{\hat{e}(f) = 0\}\right)
\end{aligned}
$$

- last step is union bound $Pr(A \cup B) \leq Pr(A) + Pr(B)$

PAC bound for SL

- A biased coin with $P(\text{heads}) = \epsilon$ producing l tails

$$\sum_{\{f \in \mathcal{F} : e(f) > \epsilon\}} Pr_{\mathcal{D} \sim P}\left(\{\hat{e}(f) = 0\}\right) \leq \sum_{\{f \in \mathcal{F} : e(f) > \epsilon\}} (1 - \epsilon)^l$$

PAC bound for SL

- A biased coin with $P(\text{heads}) = \epsilon$ producing l tails

$$\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} Pr_{\mathcal{D} \sim P}\left(\{\hat{e}(f) = 0\}\right) \leq \sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l$$

- if $\mathcal{F}$ is finite, $\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l \leq |\mathcal{F}|(1-\epsilon)^l$

PAC bound for SL

- A biased coin with $P(\text{heads}) = \epsilon$ producing l tails

$$\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} Pr_{\mathcal{D} \sim P}\left(\{\hat{e}(f) = 0\}\right) \leq \sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l$$

- if $\mathcal{F}$ is finite, $\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l \leq |\mathcal{F}|(1-\epsilon)^l$
- by $1 - x \leq e^{-x}$, $|\mathcal{F}|(1-\epsilon)^l \leq |\mathcal{F}| e^{-\epsilon l}$

PAC bound for SL

- A biased coin with $P(\text{heads}) = \epsilon$ producing l tails

$$\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} Pr_{\mathcal{D} \sim P}\left(\{\hat{e}(f) = 0\}\right) \leq \sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l$$

- if $\mathcal{F}$ is finite, $\sum_{\{f \in \mathcal{F}: e(f) > \epsilon\}} (1-\epsilon)^l \leq |\mathcal{F}|(1-\epsilon)^l$
- by $1 - x \leq e^{-x}$, $|\mathcal{F}|(1-\epsilon)^l \leq |\mathcal{F}|e^{-\epsilon l}$
- putting things together, $Pr_{\mathcal{D} \sim P}\left(\{e(f_{\mathcal{D}}) \leq \epsilon\}\right) \geq 1 - |\mathcal{F}|e^{-\epsilon l}$

PAC bound for SL

- A biased coin with $P(\text{heads}) = \epsilon$ producing l tails

$$\sum_{\{f\in\mathcal{F}:e(f)>\epsilon\}} Pr_{\mathcal{D}\sim P}\left(\{\hat{e}(f)=0\}\right) \le \sum_{\{f\in\mathcal{F}:e(f)>\epsilon\}} (1-\epsilon)^l$$

- if $\mathcal{F}$ is finite, $\sum_{\{f\in\mathcal{F}:e(f)>\epsilon\}}(1-\epsilon)^l \le |\mathcal{F}|(1-\epsilon)^l$
- by $1-x \le e^{-x}$, $|\mathcal{F}|(1-\epsilon)^l \le |\mathcal{F}|e^{-\epsilon l}$
- putting things together, $Pr_{\mathcal{D}\sim P}\left(\{e(f_{\mathcal{D}}) \le \epsilon\}\right) \ge 1 - |\mathcal{F}|e^{-\epsilon l}$

Probably (i.e., on at least $1-|\mathcal{F}|e^{-\epsilon l}$ fraction of random draws of the training sample), the function $f_{\mathcal{D}}$, picked because $\hat{e}(f_{\mathcal{D}})=0$, is approximately correct (i.e., has true error $e(f_{\mathcal{D}}) \le \epsilon$).

Simple sample complexity for SL

Theorem Assume $\mathcal{F}$ is finite. Given any $\epsilon > 0, \delta > 0$, if we see l training instances where

$$l = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{1}{\delta}\right)$$

then with probability at least $1-\delta$, all $f \in \mathcal{F}$ with zero training error $\hat{e}(f) = 0$ have $e(f) \le \epsilon$.

- ϵ controls the error of the learned function
- δ controls the confidence of the bound
- proof: setting $\delta = |\mathcal{F}|e^{-\epsilon l}$

A Finite, Doubly Realizable PAC bound for SSL

Plan: make $|\mathcal{F}|$ smaller

A Finite, Doubly Realizable PAC bound for SSL

Plan: make $|\mathcal{F}|$ smaller

- incompatibility $\Xi(f, \mathbf{x}) : \mathcal{F} \times \mathcal{X} \mapsto [0, 1]$ between a function f and an unlabeled instance $\mathbf{x}$

A Finite, Doubly Realizable PAC bound for SSL

Plan: make $|\mathcal{F}|$ smaller

- incompatibility $\Xi(f, \mathbf{x}) : \mathcal{F} \times \mathcal{X} \mapsto [0, 1]$ between a function f and an unlabeled instance $\mathbf{x}$
- example: S3VM wants $|f(\mathbf{x})| \geq \gamma$. Define

$$\Xi_{\text{S3VM}}(f, \mathbf{x}) = \begin{cases} 1, & \text{if } |f(\mathbf{x})| < \gamma \\ 0, & \text{otherwise.} \end{cases}$$

A Finite, Doubly Realizable PAC bound for SSL

Plan: make $|\mathcal{F}|$ smaller

- incompatibility $\Xi(f, \mathbf{x}) : \mathcal{F} \times \mathcal{X} \mapsto [0, 1]$ between a function f and an unlabeled instance $\mathbf{x}$
- example: S3VM wants $|f(\mathbf{x})| \geq \gamma$. Define

$$\Xi_{\mathsf{S3VM}}(f, \mathbf{x}) = \begin{cases} 1, & \text{if } |f(\mathbf{x})| < \gamma \\ 0, & \text{otherwise.} \end{cases}$$

- true unlabeled data error $e_U(f) = \mathbb{E}_{\mathbf{x} \sim P_X} [\Xi(f, \mathbf{x})]$

A Finite, Doubly Realizable PAC bound for SSL

Plan: make $|\mathcal{F}|$ smaller

- incompatibility $\Xi(f, \mathbf{x}) : \mathcal{F} \times \mathcal{X} \mapsto [0, 1]$ between a function f and an unlabeled instance $\mathbf{x}$
- example: S3VM wants $|f(\mathbf{x})| \geq \gamma$. Define

$$\Xi_{\text{S3VM}}(f, \mathbf{x}) = \begin{cases} 1, & \text{if } |f(\mathbf{x})| < \gamma \\ 0, & \text{otherwise.} \end{cases}$$

- true unlabeled data error $e_U(f) = \mathbb{E}_{\mathbf{x} \sim P_X} [\Xi(f, \mathbf{x})]$
- sample unlabeled data error $\hat{e}_U(f) = \frac{1}{u} \sum_{i=l+1}^{l+u} \Xi(f, \mathbf{x}_i)$

A Finite, Doubly Realizable PAC bound for SSL

- by a similar argument, after $u = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)$ unlabeled data, with probability at least $1-\delta/2$, all $f \in \mathcal{F}$ with $\hat{e}_U(f) = 0$ have $e_U(f) \leq \epsilon$.

A Finite, Doubly Realizable PAC bound for SSL

- by a similar argument, after $u = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)$ unlabeled data, with probability at least $1 - \delta/2$, all $f \in \mathcal{F}$ with $\hat{e}_U(f) = 0$ have $e_U(f) \le \epsilon$.
- i.e., if $\hat{e}_U(f) = 0$, then $f \in \mathcal{F}(\epsilon) \equiv \{f \in \mathcal{F} : e_U(f) \le \epsilon\}$

A Finite, Doubly Realizable PAC bound for SSL

- by a similar argument, after $u = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)$ unlabeled data, with probability at least $1-\delta/2$, all $f \in \mathcal{F}$ with $\hat{e}_U(f) = 0$ have $e_U(f) \le \epsilon$.
- i.e., if $\hat{e}_U(f) = 0$, then $f \in \mathcal{F}(\epsilon) \equiv \{f \in \mathcal{F} : e_U(f) \le \epsilon\}$
- apply the SL PAC bound on the (much smaller) $\mathcal{F}(\epsilon)$

A Finite, Doubly Realizable PAC bound for SSL

- by a similar argument, after $u = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)$ unlabeled data, with probability at least $1-\delta/2$, all $f \in \mathcal{F}$ with $\hat{e}_U(f) = 0$ have $e_U(f) \le \epsilon$.
- i.e., if $\hat{e}_U(f) = 0$, then $f \in \mathcal{F}(\epsilon) \equiv \{f \in \mathcal{F} : e_U(f) \le \epsilon\}$
- apply the SL PAC bound on the (much smaller) $\mathcal{F}(\epsilon)$

Theorem (finite, doubly realizable) Assume $\mathcal{F}$ is finite. Given any $\epsilon > 0, \delta > 0$, if we see l labeled and u unlabeled training instances where

$$l = \frac{1}{\epsilon}\left(\log|\mathcal{F}(\epsilon)| + \log\frac{2}{\delta}\right) \text{ and } u = \frac{1}{\epsilon}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right),$$

then with probability at least $1-\delta$, all $f \in \mathcal{F}$ with $\hat{e}(f) = 0$ and $\hat{e}_U(f) = 0$ have $e(f) \le \epsilon$.

Discussions on the PAC bound for SSL

- Good news: can require less labeled data than SL

Discussions on the PAC bound for SSL

- Good news: can require less labeled data than SL
- This particular theorem requires finite $\mathcal{F}$, and doubly realizable f with $\hat{e}(f) = 0$ and $\hat{e}_U(f) = 0$

Discussions on the PAC bound for SSL

- Good news: can require less labeled data than SL
- This particular theorem requires finite $\mathcal{F}$, and doubly realizable f with $\hat{e}(f) = 0$ and $\hat{e}_U(f) = 0$
- More general theorems in (Balcan & Blum 2008):
  - infinite $\mathcal{F}$ is OK: extensions of the VC-dimension
  - agnostic, does not require either realizability: both $e(f)$ and $e_U(f)$ may be non-zero and unknown
  - also tighter ϵ-cover based bounds

Discussions on the PAC bound for SSL

- Good news: can require less labeled data than SL
- This particular theorem requires finite $\mathcal{F}$, and doubly realizable f with $\hat{e}(f) = 0$ and $\hat{e}_U(f) = 0$
- More general theorems in (Balcan & Blum 2008):
  - infinite $\mathcal{F}$ is OK: extensions of the VC-dimension
  - agnostic, does not require either realizability: both $e(f)$ and $e_U(f)$ may be non-zero and unknown
  - also tighter ϵ-cover based bounds
- Most SSL algorithms (e.g. S3VMs) empirically minimize $\hat{e}(f) + \hat{e}_U(f)$: not necessarily justified in theory

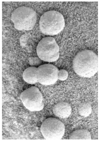
Discussions on the PAC bound for SSL

- Good news: can require less labeled data than SL
- This particular theorem requires finite $\mathcal{F}$, and doubly realizable f with $\hat{e}(f) = 0$ and $\hat{e}_U(f) = 0$
- More general theorems in (Balcan & Blum 2008):
  - infinite $\mathcal{F}$ is OK: extensions of the VC-dimension
  - agnostic, does not require either realizability: both $e(f)$ and $e_U(f)$ may be non-zero and unknown
  - also tighter ϵ-cover based bounds
- Most SSL algorithms (e.g. S3VMs) empirically minimize $\hat{e}(f) + \hat{e}_U(f)$: not necessarily justified in theory
- Incompatibility functions arbitrary. Serves as regularization. There are good and bad incompatibility functions. Example: "inverse S3VM" prefers to cut through dense unlabeled data

$$\Xi_{\text{inv}}(f, \mathbf{x}) = 1 - \Xi_{\text{S3VM}}(f, \mathbf{x})$$

Outline

Life-long learning

x_1	x_2	$\ldots$	x_{1000}	$\ldots$	$x_{1000000}$	$\ldots$
		$\ldots$		$\ldots$		$\ldots$
$y_1 = 0$	-	-	$y_{1000} = 1$	$\ldots$	$y_{1000000} = 0$	$\ldots$

- $n \to \infty$ examples arrive sequentially, cannot store them all
- most examples unlabeled
- no iid assumption, $p(x, y)$ can change over time

This is how children learn, too

x_1	x_2	$\ldots$	x_{1000}	$\ldots$	$x_{1000000}$	$\ldots$
		$\ldots$		$\ldots$		$\ldots$
$y_1 = 0$	-	-	$y_{1000} = 1$	$\ldots$	$y_{1000000} = 0$	$\ldots$

New paradigm: online semi-supervised learning

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t

New paradigm: online semi-supervised learning

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t
2. Learner has classifier $f_t : \mathcal{X} \mapsto \mathbb{R}$, predicts $f_t(x_t)$

New paradigm: online semi-supervised learning

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t
2. Learner has classifier $f_t : \mathcal{X} \mapsto \mathbb{R}$, predicts $f_t(x_t)$
3. With small probability, adversary reveals y_t; otherwise it abstains (unlabeled)

New paradigm: online semi-supervised learning

1. At time t, adversary picks $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$ not necessarily iid, shows x_t
2. Learner has classifier $f_t : \mathcal{X} \mapsto \mathbb{R}$, predicts $f_t(x_t)$
3. With small probability, adversary reveals y_t; otherwise it abstains (unlabeled)
4. Learner updates to f_{t+1} based on x_t and y_t (if given). Repeat.

Online manifold regularization

- Recall (batch) manifold regularization risk:

$$
\begin{aligned}
J(f) \quad = \quad & \frac{1}{l} \sum_{t=1}^{T} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 \\
& + \frac{\lambda_2}{2T} \sum_{s,t=1}^{T} (f(x_s) - f(x_t))^2 w_{st}
\end{aligned}
$$

$c(f(x), y)$ convex loss function, e.g., the hinge loss.

Online manifold regularization

- Recall (batch) manifold regularization risk:

$$J(f) = \frac{1}{l}\sum_{t=1}^{T}\delta(y_t)c(f(x_t),y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \frac{\lambda_2}{2T}\sum_{s,t=1}^{T}(f(x_s)-f(x_t))^2 w_{st}$$

$c(f(x),y)$ convex loss function, e.g., the hinge loss.

- Instantaneous risk:

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t),y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2\sum_{i=1}^{t}(f(x_i)-f(x_t))^2 w_{it}$$

(involves graph edges between x_t and all previous examples)

Online manifold regularization

- Recall (batch) manifold regularization risk:

$$\begin{aligned} J(f) &= \frac{1}{l}\sum_{t=1}^{T}\delta(y_t)c(f(x_t),y_t) + \frac{\lambda_1}{2}\|f\|_K^2 \\ &+\frac{\lambda_2}{2T}\sum_{s,t=1}^{T}(f(x_s)-f(x_t))^2 w_{st} \end{aligned}$$

  $c(f(x),y)$ convex loss function, e.g., the hinge loss.

- Instantaneous risk:

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t),y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2\sum_{i=1}^{t}(f(x_i)-f(x_t))^2 w_{it}$$

  (involves graph edges between x_t and all previous examples)

- batch risk = average instantaneous risks $J(f) = \frac{1}{T}\sum_{t=1}^{T} J_t(f)$

Online convex programming

- Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t: $f_{t+1} = f_t - \eta_t \left. \frac{\partial J_t(f)}{\partial f} \right|_{f_t}$

Online convex programming

- Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t: $f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}$
- Step size η_t decays, e.g., $\eta_t = 1/\sqrt{t}$

Online convex programming

- Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t: $f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}$
- Step size η_t decays, e.g., $\eta_t = 1/\sqrt{t}$
- Accuracy can be arbitrarily bad if adversary flips target often. If so, no batch learner in hindsight can do well either

$$\text{regret} \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t) - J(f^*)$$

Online convex programming

- Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t: $f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}$
- Step size η_t decays, e.g., $\eta_t = 1/\sqrt{t}$
- Accuracy can be arbitrarily bad if adversary flips target often. If so, no batch learner in hindsight can do well either

$$\text{regret} \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t) - J(f^*)$$

- no-regret guarantee against adversary [Zinkevich ICML03]: $\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} J_t(f_t) - J(f^*) \le 0$.

Online convex programming

- Instead of minimizing convex $J(f)$, reduce convex $J_t(f)$ at each step t: $f_{t+1} = f_t - \eta_t \left. \frac{\partial J_t(f)}{\partial f} \right|_{f_t}$
- Step size η_t decays, e.g., $\eta_t = 1/\sqrt{t}$
- Accuracy can be arbitrarily bad if adversary flips target often. If so, no batch learner in hindsight can do well either

$$\text{regret} \equiv \frac{1}{T} \sum_{t=1}^{T} J_t(f_t) - J(f^*)$$

- no-regret guarantee against adversary [Zinkevich ICML03]: $\limsup_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} J_t(f_t) - J(f^*) \leq 0$.
- If no adversary (iid), the average classifier $\bar{f} = 1/T \sum_{t=1}^{T} f_t$ is good: $J(\bar{f}) \to J(f^*)$.

Sparse approximation by buffering

The algorithm is impractical as $T \to \infty$:

- space $O(T)$: stores all previous examples
- time $O(T^2)$: each new instance connects to all previous ones

Sparse approximation by buffering

The algorithm is impractical as $T \to \infty$:

- space $O(T)$: stores all previous examples
- time $O(T^2)$: each new instance connects to all previous ones

Keep a size τ buffer

- approximate representers: $f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot)$

Sparse approximation by buffering

The algorithm is impractical as $T \to \infty$:

- space $O(T)$: stores all previous examples
- time $O(T^2)$: each new instance connects to all previous ones

Keep a size τ buffer

- approximate representers: $f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot)$
- approximate instantaneous risk

$$
\begin{aligned}
J_t(f) &= \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 \\
&\quad + \lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t} (f(x_i) - f(x_t))^2 w_{it}
\end{aligned}
$$

Sparse approximation by buffering

The algorithm is impractical as $T \to \infty$:

- space $O(T)$: stores all previous examples
- time $O(T^2)$: each new instance connects to all previous ones

Keep a size τ buffer

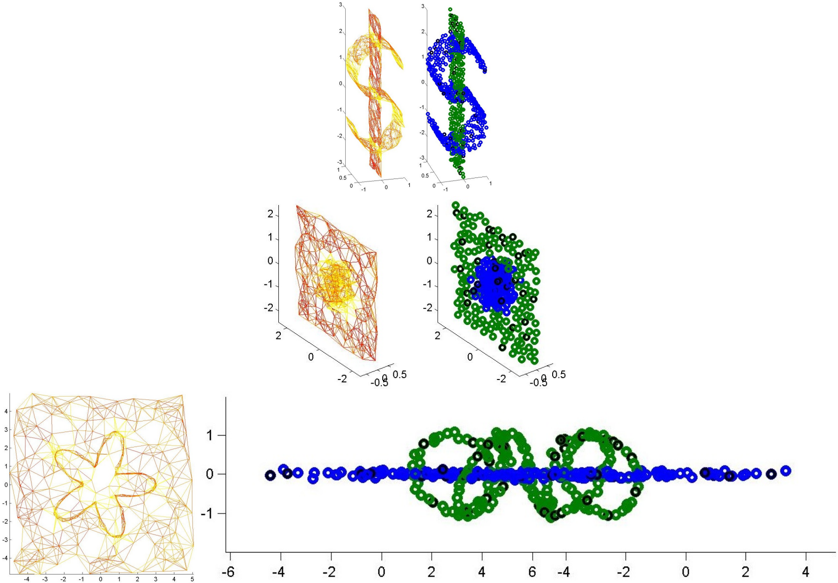
- approximate representers: $f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot)$
- approximate instantaneous risk

$$
\begin{aligned}
J_t(f) &= \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 \\
&\quad +\lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t} (f(x_i) - f(x_t))^2 w_{it}
\end{aligned}
$$

- dynamic graph on instances in the buffer

Outline

Multiple, intersecting manifolds

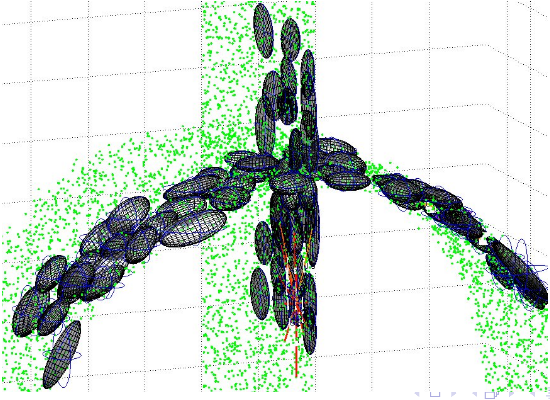


Building Blocks: Local Covariance Matrix

For a sparse subset of points x, the local covariance matrix of the neighbors

$$\Sigma_x = \frac{1}{m-1} \sum_j (x_j - \mu_x)(x_j - \mu_x)^\top$$

captures local geometry.



A Distance on Covariance Matrices

- Hellinger distance

$$H^2(p,q) = \frac{1}{2}\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx$$

A Distance on Covariance Matrices

- Hellinger distance

$$H^2(p,q) = \frac{1}{2} \int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$$

- $H(p,q)$ symmetric, in $[0,1]$

A Distance on Covariance Matrices

- Hellinger distance

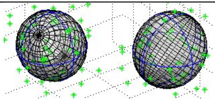
$$H^2(p,q) = \frac{1}{2} \int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$$

- $H(p,q)$ symmetric, in $[0,1]$
- Let $p = N(0, \Sigma_1), q = N(0, \Sigma_2)$. We define

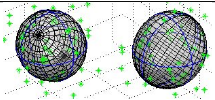
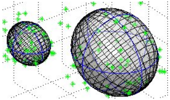
$$H(\Sigma_1, \Sigma_2) \equiv H(p,q) = \sqrt{1 - 2^{\frac{d}{2}} \frac{|\Sigma_1|^{\frac{1}{4}} |\Sigma_2|^{\frac{1}{4}}}{|\Sigma_1 + \Sigma_2|^{\frac{1}{2}}}}$$

(computed in common subspace)

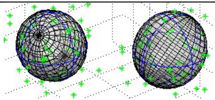
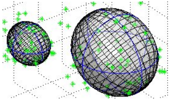
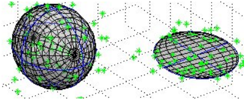
Hellinger Distance

	Comment	$H(\Sigma_1, \Sigma_2)$
	similar	0.02

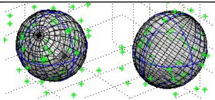
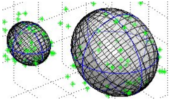
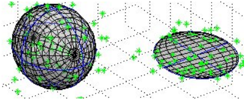
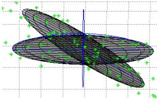
Hellinger Distance

	Comment	$H(\Sigma_1, \Sigma_2)$
	similar	0.02
	density	0.28

Hellinger Distance

	Comment	$H(\Sigma_1, \Sigma_2)$
	similar	0.02
	density	0.28
	dimension	1

Hellinger Distance

	Comment	$H(\Sigma_1, \Sigma_2)$
	similar	0.02
	density	0.28
	dimension	1
	orientation*	1

* smoothed version: $\Sigma + \epsilon I$

A Sparse Graph

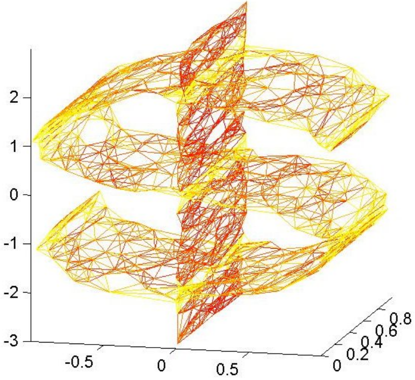
- KNN graph use Mahalanobis distance to trace the manifold
  $d^2(x, y) = (x - y)^\top \Sigma_x^{-1} (x - y)$

A Sparse Graph

- KNN graph use Mahalanobis distance to trace the manifold $d^2(x, y) = (x - y)^\top \Sigma_x^{-1}(x - y)$
- Gaussian edge weight on edges $w_{ij} = e^{-\frac{H^2(\Sigma_{x_i}, \Sigma_{x_j})}{2\sigma^2}}$

A Sparse Graph

- KNN graph use Mahalanobis distance to trace the manifold $d^2(x,y) = (x-y)^\top \Sigma_x^{-1}(x-y)$
- Gaussian edge weight on edges $w_{ij} = e^{-\frac{H^2(\Sigma_{x_i},\Sigma_{x_j})}{2\sigma^2}}$
- Combines locality and shape. Red=large w, yellow=small w



- Manifold Regularization on the graph

Outline

Do we learn from both labeled and unlabeled data?

Learning exists long before machine learning. Do humans perform semi-supervised learning?

Do we learn from both labeled and unlabeled data?

Learning exists long before machine learning. Do humans perform semi-supervised learning?

- We discuss two human experiments:
  1. One-class classification [Zaki & Nosofsky 2007]
  2. Binary classification [Zhu et al. 2007]

Zaki & Nosofsky 2007: self training?

- participants shown training sample $\{(\mathbf{x}_i, y_i = 1)\}_{i=1}^{l}$, all from one class.

Zaki & Nosofsky 2007: self training?

- participants shown training sample $\{(\mathbf{x}_i, y_i = 1)\}_{i=1}^{l}$, all from one class.
- shown u unlabeled instances $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, decide if $y_i = 1$

Zaki & Nosofsky 2007: self training?

- participants shown training sample $\{(\mathbf{x}_i, y_i = 1)\}_{i=1}^{l}$, all from one class.
- shown u unlabeled instances $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, decide if $y_i = 1$
- density level-set problem: learn $\mathcal{X}_1 = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}|y=1) \geq \epsilon\}$, classify $y = 1$ if $\mathbf{x} \in \mathcal{X}_1$

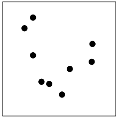
Zaki & Nosofsky 2007: self training?

- participants shown training sample $\{(\mathbf{x}_i, y_i = 1)\}_{i=1}^{l}$, all from one class.
- shown u unlabeled instances $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, decide if $y_i = 1$
- density level-set problem: learn $\mathcal{X}_1 = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}|y = 1) \geq \epsilon\}$, classify $y = 1$ if $\mathbf{x} \in \mathcal{X}_1$
- if $\mathcal{X}_1$ is fixed after training, then test data won't affect classification.

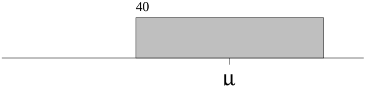
Zaki & Nosofsky 2007: self training?

- participants shown training sample $\{(\mathbf{x}_i, y_i = 1)\}_{i=1}^{l}$, all from one class.
- shown u unlabeled instances $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, decide if $y_i = 1$
- density level-set problem: learn $\mathcal{X}_1 = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}|y = 1) \geq \epsilon\}$, classify $y = 1$ if $\mathbf{x} \in \mathcal{X}_1$
- if $\mathcal{X}_1$ is fixed after training, then test data won't affect classification.
- Zaki & Nosofsky showed this is not true.

The Zaki & Nosofsky 2007 experiment

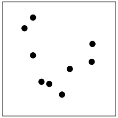


(a) a stimulus

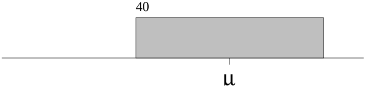


(b) training distribution

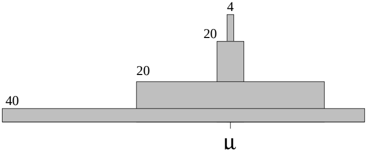
The Zaki & Nosofsky 2007 experiment



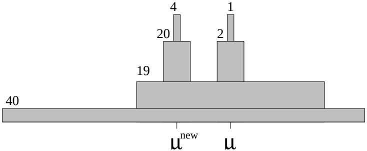
(a) a stimulus



(b) training distribution

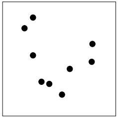


(c) condition 1 test distribution

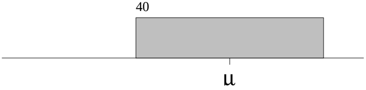


(d) condition 2 test distribution

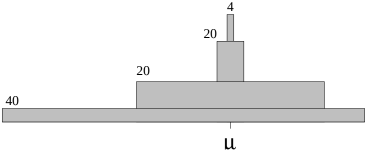
The Zaki & Nosofsky 2007 experiment



(a) a stimulus

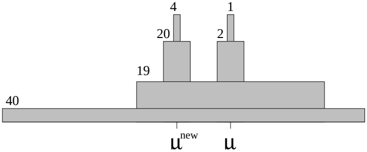


(b) training distribution



(c) condition 1 test distribution

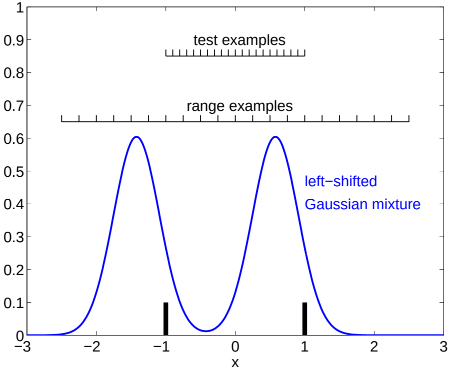
$$\hat{p}(y=1|\mu) > \hat{p}(y=1|\text{low})$$
$$> \hat{p}(y=1|\text{high}) \gg \hat{p}(y=1|\text{random})$$



(d) condition 2 test distribution

$$\hat{p}(y=1|\mu^{\text{new}}) > \hat{p}(y=1|\text{low}^{\text{new}})$$
$$> \hat{p}(y=1|\mu) \approx \hat{p}(y=1|\text{low})$$
$$\approx \hat{p}(y=1|\text{high}) \gg \hat{p}(y=1|\text{random})$$

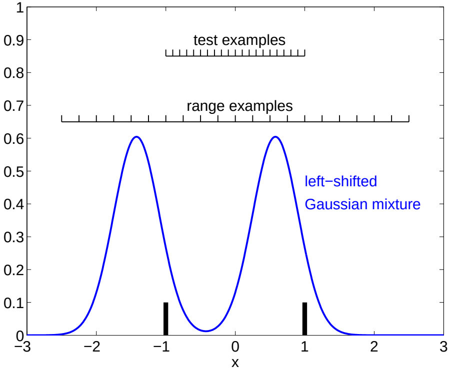
Zhu et al. 2007: mixture model?



blocks

1. 20 labeled points at $x = -1, 1$

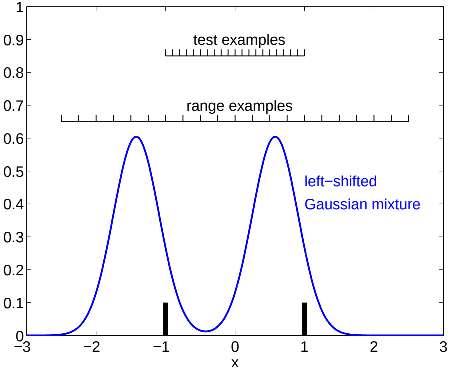
Zhu et al. 2007: mixture model?



blocks

1. 20 labeled points at $x = -1, 1$
2. test 1: 21 test examples in grid $[-1, 1]$

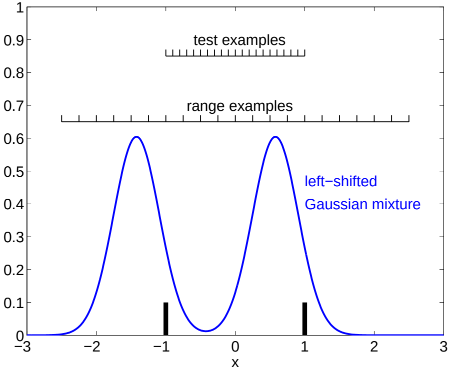
Zhu et al. 2007: mixture model?



blocks

1. 20 labeled points at $x = -1, 1$
2. test 1: 21 test examples in grid $[-1, 1]$
3. 690 examples $\sim$ bimodal distribution, plus 63 range examples in $[-2.5, 2.5]$

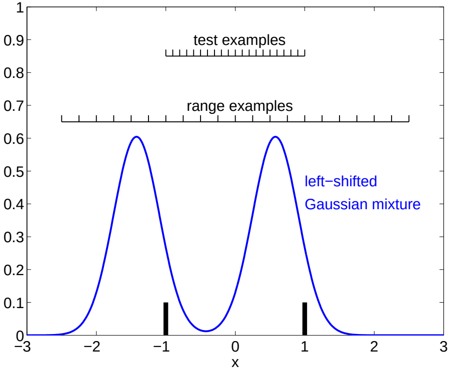
Zhu et al. 2007: mixture model?



blocks

1. 20 labeled points at $x = -1, 1$
2. test 1: 21 test examples in grid $[-1, 1]$
3. 690 examples $\sim$ bimodal distribution, plus 63 range examples in $[-2.5, 2.5]$
4. test 2: same as test 1

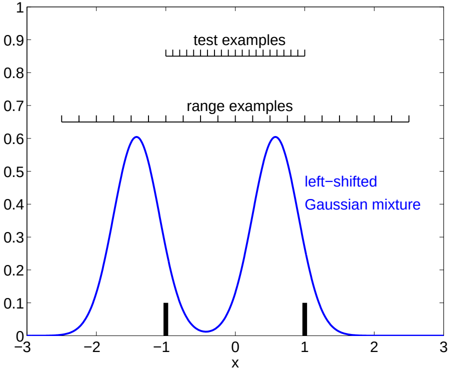
Zhu et al. 2007: mixture model?



blocks

1. 20 labeled points at $x = -1, 1$
2. test 1: 21 test examples in grid $[-1, 1]$
3. 690 examples $\sim$ bimodal distribution, plus 63 range examples in $[-2.5, 2.5]$
4. test 2: same as test 1

Zhu et al. 2007: mixture model?



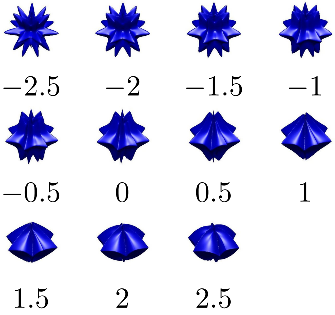
blocks

1. 20 labeled points at $x = -1, 1$
2. test 1: 21 test examples in grid $[-1, 1]$
3. 690 examples $\sim$ bimodal distribution, plus 63 range examples in $[-2.5, 2.5]$
4. test 2: same as test 1

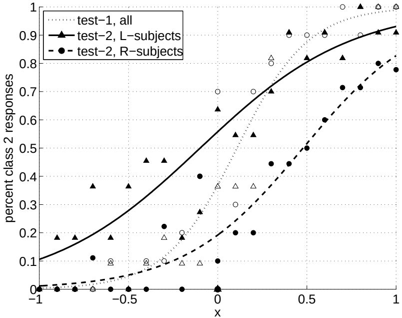
12 participants left-offset, 10 right-offset. Record their decisions and response times.

Visual stimuli

Stimuli parametrized by a continuous scalar x. Some examples:



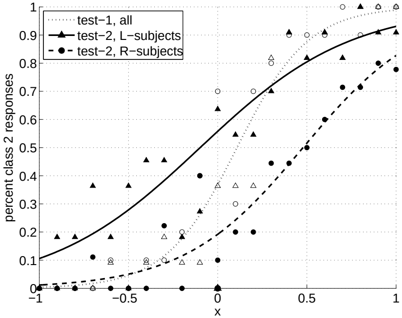
Observation 1: unlabeled data affects decision boundary



average decision boundary

- after seeing labeled data: $x = 0.11$

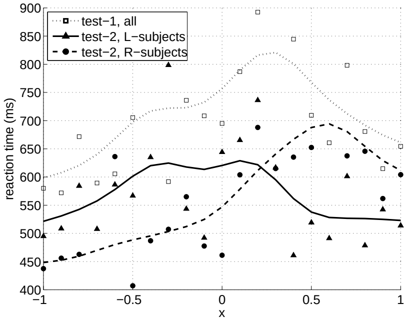
Observation 1: unlabeled data affects decision boundary



average decision boundary

- after seeing labeled data: $x = 0.11$
- after seeing labeled and unlabeled data: L-subjects $x = -0.10$, R-subjects $x = 0.48$

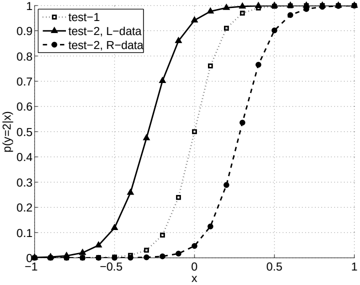
Observation 2: unlabeled data affects reaction time



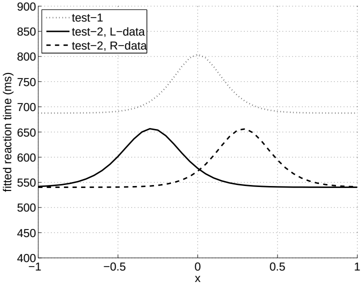
longer reaction time → harder example → closer to decision boundary.
Reaction times too suggest decision boundary shift.

Model fitting

We can fit human behavior with a GMM.



boundary shift



reaction time $t = aH(x) + b$

- Humans and machines both perform semi-supervised learning.
- Understanding natural learning may lead to new machine learning algorithms.

References

See the references in

Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.