Semi-Supervised Learning Tutorial

Xiaojin Zhu

Department of Computer Sciences
University of Wisconsin, Madison, USA

ICML 2007

Outline

Outline

Disclaimer

- This tutorial reflects my subjective opinions.
- Many work cannot be included.

Thank Olivier Chapelle for some of the S3VM figures.

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get
  - human annotation is boring

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices

Why bother?

Because people want better performance for free.

the traditional view

- unlabeled data is cheap
- labeled data can be hard to get
  - human annotation is boring
  - labels may require experts
  - labels may require special devices
  - your graduate student is on vacation

Example of hard-to-get labels

Task: speech analysis

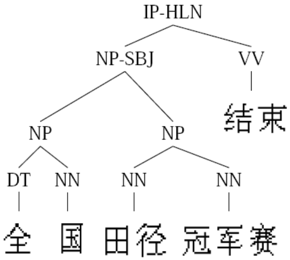
- Switchboard dataset
- telephone conversation transcription
- 400 hours annotation time for each hour of speech

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



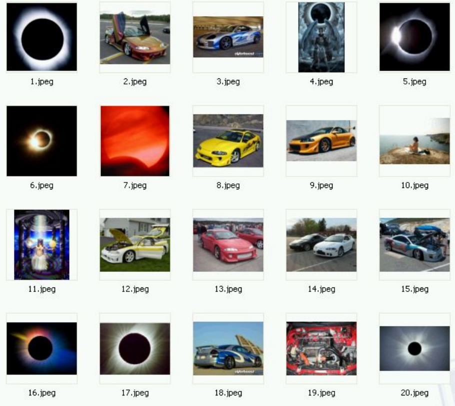
"The National Track and Field Championship has finished."

Example of not-so-hard-to-get labels

a little secret

For some tasks, it may not be too difficult to label 1000+ instances.

Task: image categorization of "eclipse"

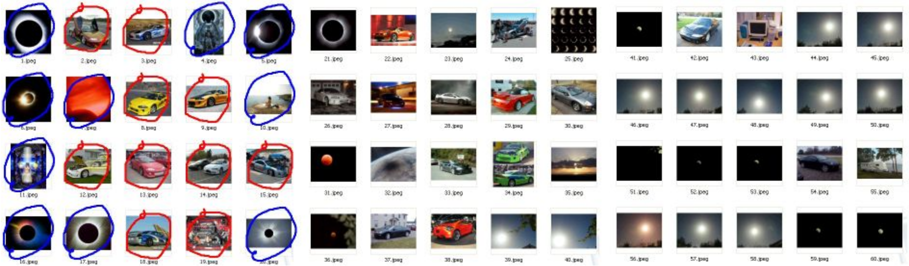


Example of not-so-hard-to-get labels

There are ways like the ESP game (`www.espgame.org`) to encourage "human computation" for more labels.

Example of not-so-hard-to-get labels

nonetheless...



In this tutorial we will learn how to use unlabeled data to improve classification.

The Learning Problem

Goal

Using both labeled and unlabeled data to build better learners, than using each one alone.

Notations

- input instance x, label y
- learner $f : \mathcal{X} \mapsto \mathcal{Y}$
- labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$
- unlabeled data $X_u = \{x_{l+1:n}\}$, available during training
- usually $l \ll n$
- test data $X_{test} = \{x_{n+1:}\}$, not available during training

Semi-supervised vs. transductive learning

- labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$
- unlabeled data $X_u = \{x_{l+1:n}\}$, available during training
- test data $X_{test} = \{x_{n+1:}\}$, not available during training

Semi-supervised learning

is ultimately applied to the test data (inductive).

Transductive learning

is only concerned with the unlabeled data.

Why the name

supervised learning (classification, regression) $\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{test}\}$

transductive classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

semi-supervised clustering $\{x_{1:n},$ must-, cannot-links$\}$

$\updownarrow$

unsupervised learning (clustering) $\{x_{1:n}\}$

Why the name

supervised learning (classification, regression) $\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{test}\}$

transductive classification/regression $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

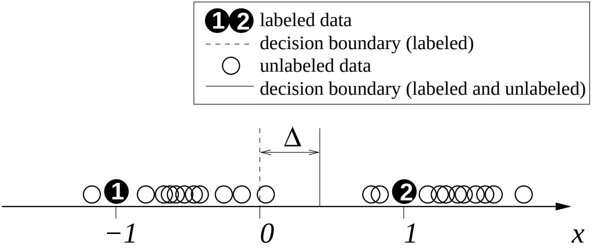
semi-supervised clustering $\{x_{1:n}, \text{must-, cannot-links}\}$

$\updownarrow$

unsupervised learning (clustering) $\{x_{1:n}\}$

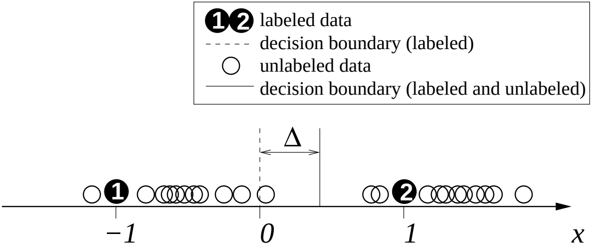
We will mainly discuss semi-supervised classification.

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.

Does unlabeled data always help?

Unfortunately, this is not the case, yet.

Outline

Outline

1. Introduction to Semi-Supervised Learning
2. Semi-Supervised Learning Algorithms
   - Self Training
   - Generative Models
   - S3VMs
   - Graph-Based Algorithms
   - Multiview Algorithms
3. Semi-Supervised Learning in Nature
4. Some Challenges for Future Research

Self-training algorithm

Assumption

One's own high confidence predictions are correct.

Self-training algorithm:

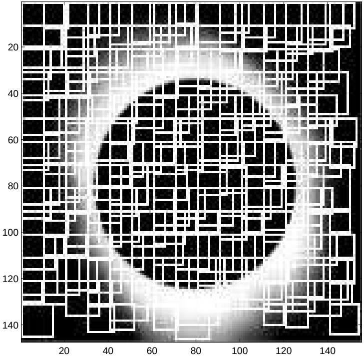
1. Train f from (X_l, Y_l)
2. Predict on $x \in X_u$
3. Add $(x, f(x))$ to labeled data
4. Repeat

Variations in self-training

- Add a few most confident $(x, f(x))$ to labeled data
- Add all $(x, f(x))$ to labeled data
- Add all $(x, f(x))$ to labeled data, weigh each by confidence

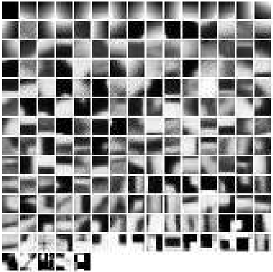
Self-training example: image categorization

- Each image is divided into small patches
- 10×10 grid, random size in $10 \sim 20$



Self-training example: image categorization

- All patches are normalized.
- Define a dictionary of 200 'visual words' (cluster centroids) with 200-means clustering on all patches.
- Represent a patch by the index of its closest visual word.



The bag-of-word representation of images

 ⟶ 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

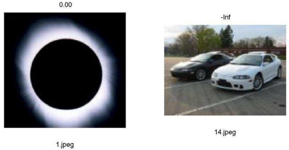
141:1 142:0 143:6 144:0 145:2 146:0 147:3 148:0 149:0 150:0 151:0 152:0 153:0 154:1 155:0 156:0 157:3 158:12 159:4 160:0

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

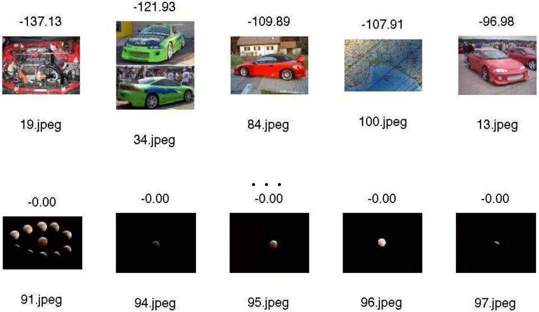
181:1 182:2 183:0 184:0 185:2 186:0 187:0 188:0 189:0 190:0 191:0 192:0 193:1 194:2 195:4 196:0 197:0 198:0 199:0 200:0

Self-training example: image categorization

1. Train a naïve Bayes classifier on the two initial labeled images



2. Classify unlabeled data, sort by confidence $\log p(y = \text{astronomy}|x)$



Self-training example: image categorization

3. Add the most confident images and predicted labels to labeled data

0.00

1.jpeg

-Inf

14.jpeg

-Inf

19.jpeg

0.00

97.jpeg

4. Re-train the classifier and repeat

-194.24

34.jpeg

-161.15

13.jpeg

-159.15

100.jpeg

-157.61

84.jpeg

-143.27

8.jpeg

. . .

-0.00

90.jpeg

-0.00

91.jpeg

-0.00

94.jpeg

-0.00

95.jpeg

-0.00

96.jpeg

Advantages of self-training

- The simplest semi-supervised learning method.
- A wrapper method, applies to existing (complex) classifiers.
- Often used in real tasks like natural language processing.

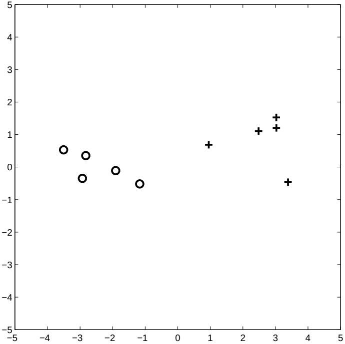
Disadvantages of self-training

- Early mistakes could reinforce themselves.
  - Heuristic solutions, e.g. "un-label" an instance if its confidence falls below a threshold.
- Cannot say too much in terms of convergence.
  - But there are special cases when self-training is equivalent to the Expectation-Maximization (EM) algorithm.
  - There are also special cases (e.g., linear functions) when the closed-form solution is known.

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

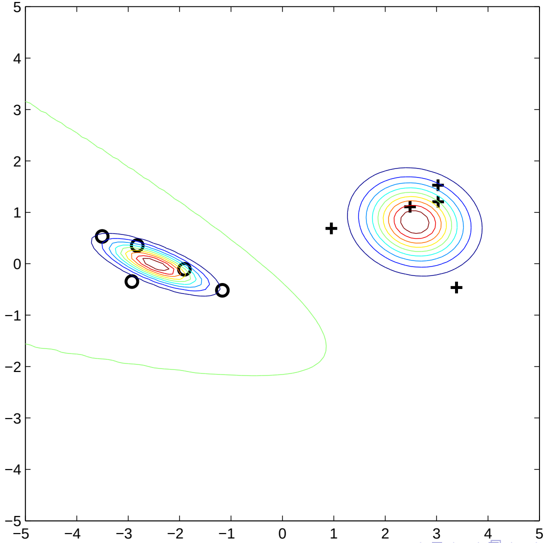
Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$
The GMM:

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

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

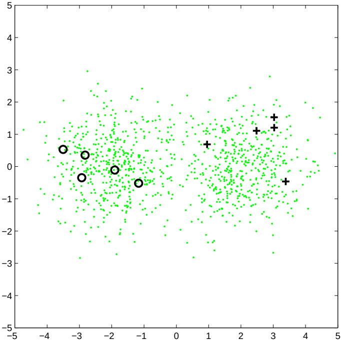
A simple example of generative models

The most likely model, and its decision boundary:



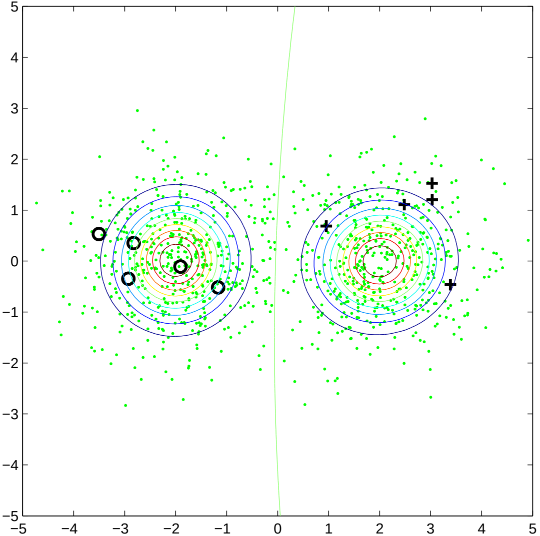
A simple example of generative models

Adding unlabeled data:



A simple example of generative models

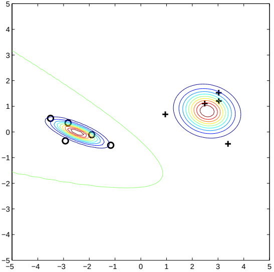
With unlabeled data, the most likely model and its decision boundary:

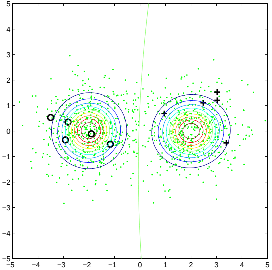


A simple example of generative models

They are different because they maximize different quantities.

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





Generative model for semi-supervised learning

Assumption

The full generative model $p(X, Y|\theta)$.

Generative model for semi-supervised learning:

- quantity of interest: $p(X_l, Y_l, X_u|\theta) = \sum_{Y_u} p(X_l, Y_l, X_u, Y_u|\theta)$
- find the maximum likelihood estimate (MLE) of θ, the maximum a posteriori (MAP) estimate, or be Bayesian

Examples of some generative models

Often used in semi-supervised learning:

- Mixture of Gaussian distributions (GMM)
  - image classification
  - the EM algorithm
- Mixture of multinomial distributions (Naïve Bayes)
  - text categorization
  - the EM algorithm
- Hidden Markov Models (HMM)
  - speech recognition
  - Baum-Welch algorithm

Case study: GMM

For simplicity, consider binary classification with GMM using MLE.

- labeled data only
  - $\log p(X_l, Y_l|\theta) = \sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i, \theta)$
  - MLE for θ trivial (frequency, sample mean, sample covariance)
- 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)
  - The Expectation-Maximization (EM) algorithm is one method to find a local optimum.

The EM algorithm for GMM

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

The EM algorithm for GMM

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

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

The EM algorithm in general

- 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: find θ to maximize $p(\mathcal{D}|\theta)$
- Properties:
  - EM starts from an arbitrary θ_0
  - The E-step: $q(\mathcal{H}) = p(\mathcal{H}|\mathcal{D}, \theta)$
  - The M-step: maximize $\sum_{\mathcal{H}} q(\mathcal{H}) \log p(\mathcal{D}, \mathcal{H}|\theta)$
  - EM iteratively improves $p(\mathcal{D}|\theta)$
  - EM converges to a local maximum of θ

Generative model for semi-supervised learning: beyond EM

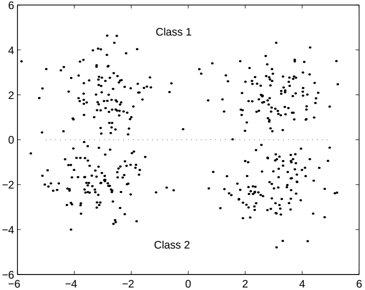
- Key is to maximize $p(X_l, Y_l, X_u|\theta)$.
- EM is just one way to maximize it.
- Other ways to find parameters are possible too, e.g., variational approximation, or direct optimization.

Advantages of generative models

- Clear, well-studied probabilistic framework
- Can be extremely effective, if the model is close to correct

Disadvantages of generative models

- Often difficult to verify the correctness of the model
- Model identifiability
- EM local optima
- Unlabeled data may hurt if generative model is wrong



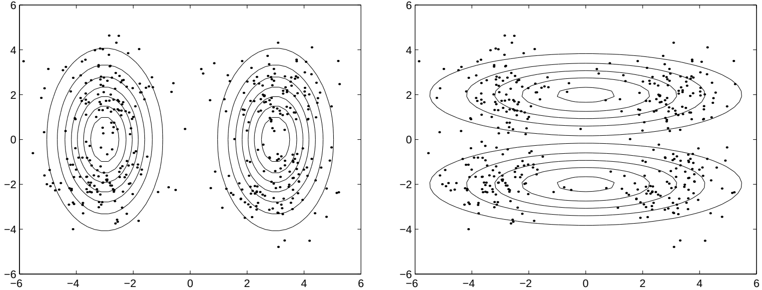
For example, classifying text by topic vs. by genre.

Unlabeled data may hurt semi-supervised learning

If the generative model is wrong:

high likelihood
wrong

low likelihood
correct



Heuristics to lessen the danger

- Carefully construct the generative model to reflect the task
  - e.g., multiple Gaussian distributions per class, instead of a single one
- Down-weight the unlabeled data ($\lambda < 1$)

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

Related method: cluster-and-label

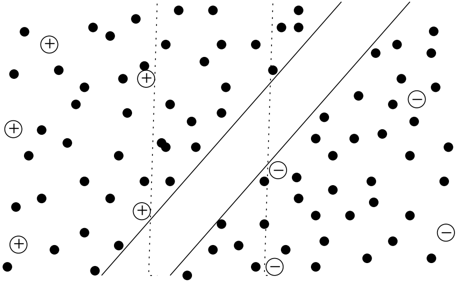
Instead of probabilistic generative models, any clustering algorithm can be used for semi-supervised classification too:

- Run your favorite clustering algorithm on X_l, X_u.
- Label all points within a cluster by the majority of labeled points in that cluster.
- Pro: Yet another simple method using existing algorithms.
- Con: Can be difficult to analyze.

Outline

Semi-supervised Support Vector Machines

- Semi-supervised SVMs (S3VMs) = Transductive SVMs (TSVMs)
- Maximizes “unlabeled data margin”



S3VMs

Assumption

Unlabeled data from different classes are separated with large margin.

S3VM idea:

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

Standard SVM review

- Problem set up:
  - 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 sign$(f(x))$

Standard soft margin SVMs

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

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

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

$$\xi_i \geq 0$$

The ξ's are slack variables.

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

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

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

$$\xi_i \geq 0$$

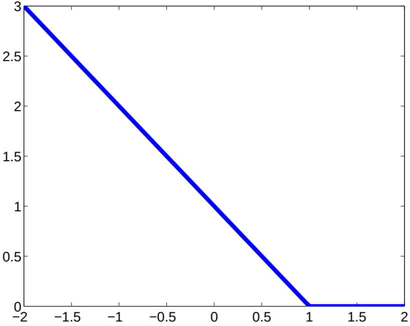
is equivalent to

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

The hinge loss in standard SVMs

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

$y_i f(x_i)$ 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.

S3VM objective function

How to incorporate unlabeled points?

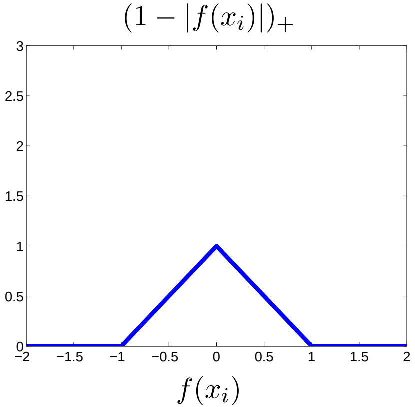
- Assign putative labels $\text{sign}(f(x))$ to $x \in X_u$
- $\text{sign}(f(x))f(x) = |f(x)|$
- The hinge loss on unlabeled points becomes

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

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



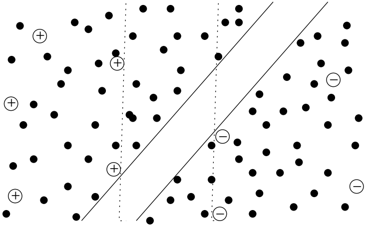
Prefers $f(x) \geq 1$ or $f(x) \leq -1$, i.e., unlabeled instance away from decision boundary $f(x) = 0$.

Avoiding unlabeled data in the margin

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 third term prefers unlabeled points outside the margin. Equivalently, the decision boundary $f = 0$ wants to be placed so that there is few unlabeled data near it.



The class balancing constraint

- Directly optimizing the S3VM objective often produces unbalanced classification – most points fall in 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 class balancing constraint: $\frac{1}{n-l}\sum_{i=l+1}^{n} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i$.

The S3VM algorithm

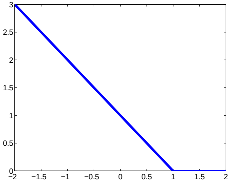
1. Input: kernel K, weights λ_1, λ_2, (X_l, Y_l), X_u
2. Solve the optimization problem for $f(x) = h(x) + b, h(x) \in \mathcal{H}_K$

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

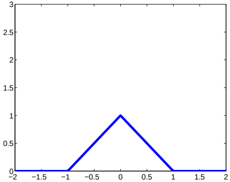
3. Classify a new test point x by sign$(f(x))$

The S3VM optimization challenge

SVM objective is convex:



Semi-supervised SVM objective is non-convex:



Finding a solution for semi-supervised SVM is difficult, which has been the focus of S3VM research. Different approaches: SVM^{light}, ∇S3VM, continuation S3VM, deterministic annealing, CCCP, Branch and Bound, SDP convex relaxation, etc.

S3VM implementation 1: SVM^{light}

- Local combinatorial search
- Assign hard labels to unlabeled data
- Outer loop: "Anneal" λ_2 from zero up
- Inner loop: Pairwise label switch

S3VM implementation 1: SVM^{light}

1. Train an SVM with (X_l, Y_l).
2. Sort X_u by $f(X_u)$. Label $y = 1, -1$ for the appropriate portions.
3. FOR $\tilde{\lambda} \leftarrow 10^{-5}\lambda_2 \ldots \lambda_2$
    1. REPEAT:
    2. $\min_f \sum_{i=1}^{l}(1 - y_i f(x_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \tilde{\lambda} \sum_{i=l+1}^{n}(1 - y_i f(x_i))_+$
    3. IF $\exists(i, j)$ switchable THEN switch y_i, y_j
    4. UNTIL No labels switchable

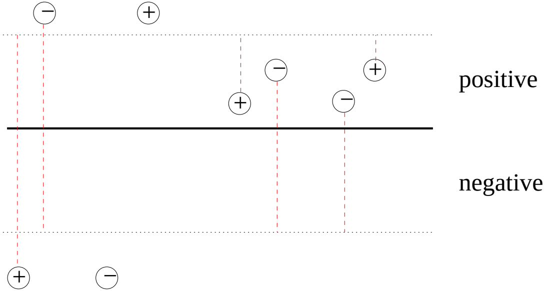
S3VM implementation 1: SVM^{light}

$i, j \in X_u$ switchable if $y_i = 1, y_j = -1$ and

$$loss(y_i = 1, f(x_i)) + loss(y_j = -1, f(x_j))$$

$$> loss(y_i = -1, f(x_i)) + loss(y_j = 1, f(x_j))$$

With the hinge loss $loss(y, f) = (1 - yf)_+$



S3VM implementation 2: ∇S3VM

Make S3VM a standard unconstrained optimization problem:

- Revert kernel to primal space
- Trick to make class balancing constraint implicit
- Smooth the hat loss so it is differentiable (though still non-convex)

S3VM implementation 2: ∇S3VM

Revert kernel to primal space:

- Given kernel $k(x_i, x_j)$, want z s.t. $z_i^\top z_j = k(x_i, x_j)$
- Cholesky factor of Gram matrix $K = B^\top B$, or
- Eigen-decomposition $K = U \Lambda U^\top, B = \Lambda^{1/2} U^\top$ (Kernel PCA map)
- The z's are columns of B
- $f(x_i) = w^\top z_i + b$, where w is the primal parameter

S3VM implementation 2: ∇S3VM

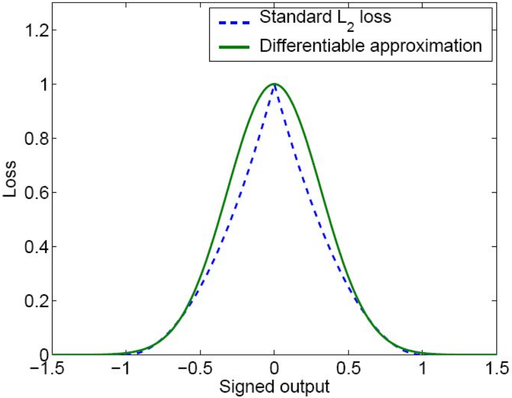
Hide class balancing constraint:

- $\frac{1}{n-l}\sum_{i=l+1}^{n}(w^\top z_i + b) = \frac{1}{l}\sum_{i=1}^{l} y_i$
- We can center the unlabeled data $\sum_{i=l+1}^{n} z_i = 0$, and
- Fix $b = \frac{1}{l}\sum_{i=1}^{l} y_i$
- The class balancing constraint is automatically satisfied.

S3VM implementation 2: ∇S3VM

Smooth the hat loss $(1 - |f|)_+$ with a similar-looking Gaussian curve

$$\exp\left(-5f^2\right)$$



S3VM implementation 2: ∇S3VM

The ∇S3VM problem ($b = \frac{1}{l}\sum_{i=1}^{l} y_i$):

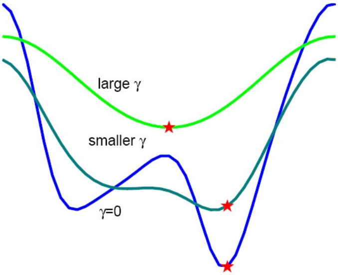
$$\min_w \quad \sum_{i=1}^{l}(1 - y_i(w^\top z_i + b))_+ + \lambda_1 \|w\|^2 \\ +\lambda_2 \sum_{i=l+1}^{n} \exp(-5(w^\top z_i + b)^2)$$

Again, increasing λ_2 gradually as a heuristic to try to avoid bad local optima.

S3VM implementation 3: Continuation method

Global optimization on the non-convex S3VM objective function.

- Convolve the objective with a Gaussian to smooth it
- With enough smoothing, global minimum is easy to find
- Gradually decrease smoothing, use previous solution as starting point
- Stop when no smoothing



S3VM implementation 3: Continuation method

1. Input: S3VM objective $R(w)$, initial weight w_0, sequence $\gamma_0 > \gamma_1 > \ldots > \gamma_p = 0$
2. Convolve: $R_\gamma(w) = (\pi\gamma)^{-d/2} \int R(w-t) \exp(-\|t\|^2/\gamma) dt$
3. FOR $i = 0 \ldots p$
    1. Starting from w_i, find local minimizer w_{i+1} of R_γ

S3VM implementation 4: CCCP

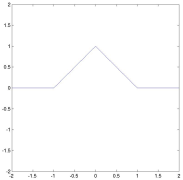
The Concave-Convex Procedure

- The non-convex hat loss function is the sum of a convex term and a concave term
- Upper bound the concave term with a line
- Iteratively minimize the sequence of convex functions

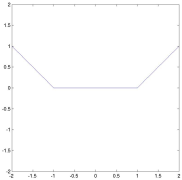
S3VM implementation 4: CCCP

The hat loss

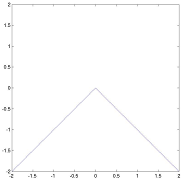
$$(1 - |f|)_+ = (|f| - 1)_+ + (-|f|) + 1$$



$=$



$+$



$+1$

S3VM implementation 4: CCCP

To minimize $R(w) = R_{vex}(w) + R_{cave}(w)$:

1. Input starting point w_0
2. $t = 0$
3. WHILE $\nabla R(w_t) \neq 0$
    1. $w_{t+1} = \arg\min_z R_{vex}(z) + \nabla R_{cave}(w_t)(z - w_t) + R_{cave}(w_t)$
    2. $t = t + 1$

S3VM implementation 5: Branch and Bound

- All previous S3VM implementations suffer from local optima.
- BB finds the exact global solution.
- It uses classic branch and bound search technique in AI.
- Unfortunately it can only handle a few hundred unlabeled points.

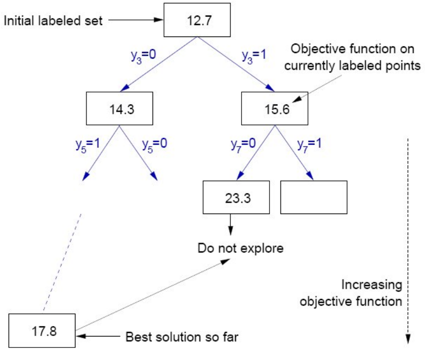
S3VM implementation 5: Branch and Bound

- Combinatorial optimization.
- A tree of partial labellings on X_u.
  - Root node: nothing in X_u labeled
  - Child node: one more $x \in X_u$ in parent node labeled
  - leaf nodes: all $x \in X_u$ labeled
- Partial labellings have non-decreasing S3VM objective

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

S3VM implementation 5: Branch and Bound

- Depth-first search on the tree
- Keep the best complete objective so far
- Prune internal node (and its subtree) if it's worse than the best objective



Advantages of S3VMs

- Applicable wherever SVMs are applicable
- Clear mathematical framework

Disadvantages of S3VMs

- Optimization difficult
- Can be trapped in bad local optima
- More modest assumption than generative model or graph-based methods, potentially lesser gain

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				•

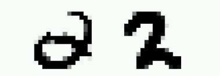
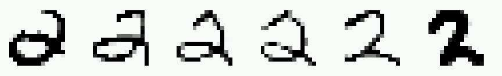
Unlabeled data as stepping stones

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

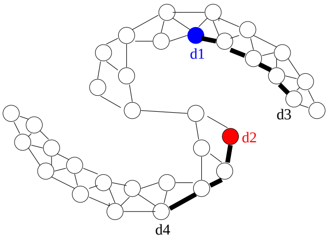
Graph-based semi-supervised learning

Assumption

A graph is given on the labeled and unlabeled data. Instances connected by heavy edge tend to have the same label.

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

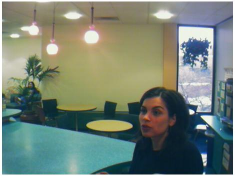


An example graph

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

Some graph-based algorithms

- mincut
- harmonic
- local and global consistency
- manifold regularization

The mincut algorithm

The graph mincut problem:

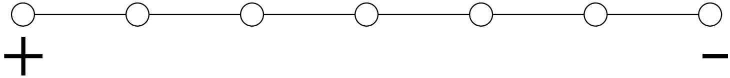
- Fix Y_l, find $Y_u \in \{0,1\}^{n-l}$ to minimize $\sum_{ij} w_{ij}|y_i - y_j|$.
- Equivalently, solves the optimization problem

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

- Combinatorial problem, but has polynomial time solution.

The mincut algorithm

- Mincut computes the modes of a Boltzmann machine
- There might be multiple modes
- One solution is to randomly perturb the weights, and average the results.



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

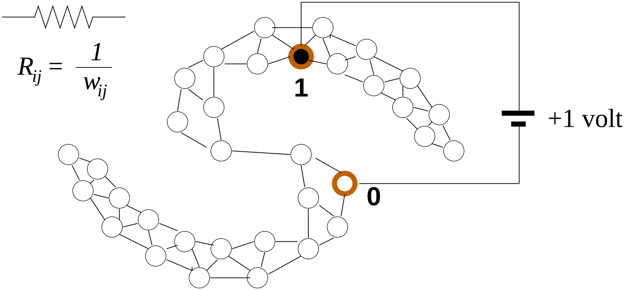
$$\sum_{i \sim j} w_{ij} (f(x_i) - f(x_j))^2$$

- the mean of a Gaussian random field
- 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

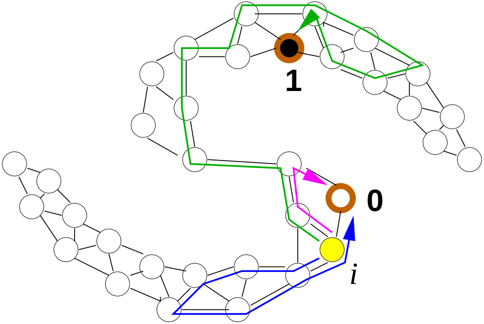
- Edges are resistors with conductance w_{ij}
- 1 volt battery connects to labeled points $y = 0, 1$
- The voltage at the nodes is the harmonic function f

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 way to compute the harmonic function is:

1. Initially, set $f(x_i) = y_i$ for $i = 1 \dots l$, 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.

This can be viewed as a special case of self-training too.

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

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

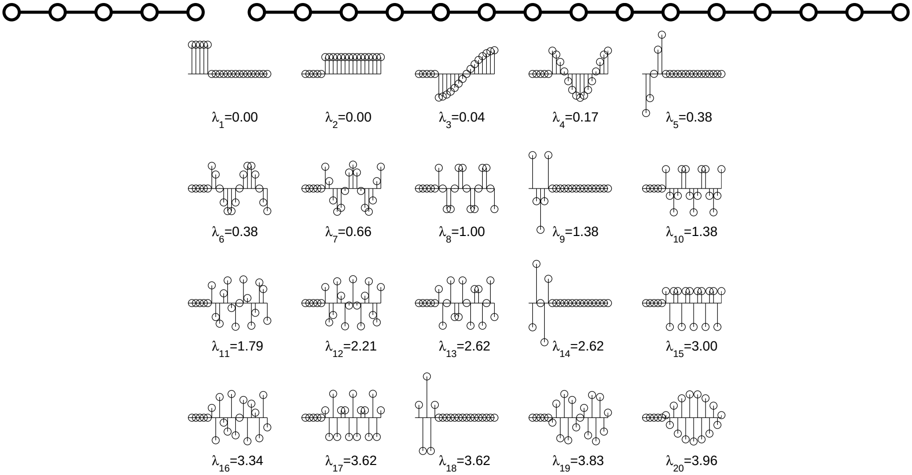
Partition the Laplacian matrix $\Delta = \begin{bmatrix} \Delta_{ll} & \Delta_{lu} \\ \Delta_{ul} & \Delta_{uu} \end{bmatrix}$

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 ($p > 0$).

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

Local and Global consistency

- Allow $f(X_l)$ to be different from Y_l, but penalize it
- Introduce a balance between labeled data fit and graph energy

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

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 $f_{1:n}^\top \Delta f_{1:n}$

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

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 sign$(f(x))$

Advantages of graph-based method

- Clear mathematical framework
- Performance is strong if the graph happens to fit the task
- The (pseudo) inverse of the Laplacian can be viewed as a kernel matrix
- Can be extended to directed graphs

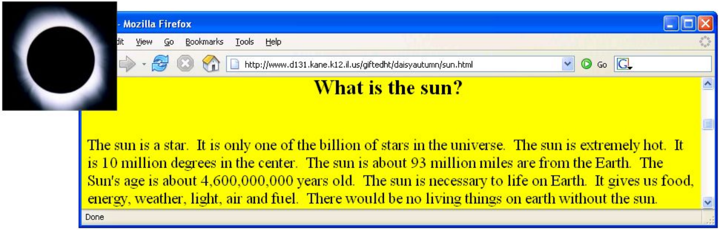
Disadvantages of graph-based method

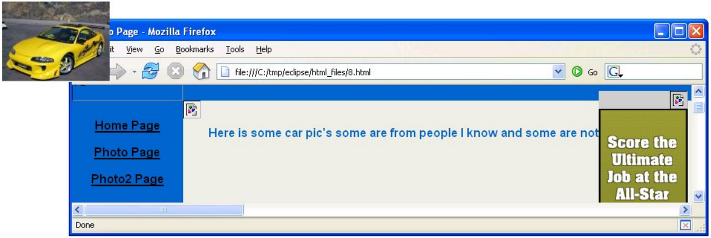
- Performance is bad if the graph is bad
- Sensitive to graph structure and edge weights

Outline

Co-training

Two views of an item: image and HTML text





Feature split

Each instance is represented by two sets 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)

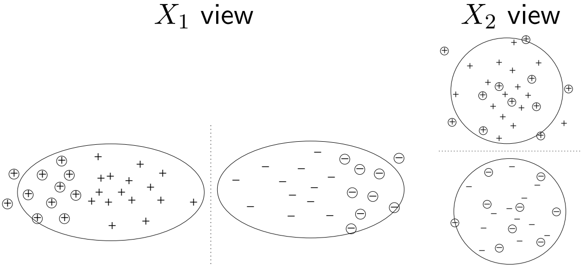
Co-training idea:

- Train an image classifier and a text classifier
- The two classifiers teach each other

Co-training 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



Co-training algorithm

Co-training algorithm

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.

Pros and cons of co-training

Pros

- Simple wrapper method. Applies to almost all existing classifiers
- Less sensitive to mistakes than self-training

Cons

- Natural feature splits 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 $P(y|x)$

Fake feature split

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

Multiview: agreement among multiple classifiers

- no feature split
- train multiple classifiers of different types
- classify unlabeled data with all classifiers
- add majority vote label

Multiview learning

A regularized risk minimization framework to encourage multi-learner agreement:

$$\min_f \sum_{v=1}^{M} \left(\sum_{i=1}^{l} c(y_i, f_v(x_i)) + \lambda_1 \|f\|_K^2 \right) + \lambda_2 \sum_{u,v=1}^{M} \sum_{i=l+1}^{n} (f_u(x_i) - f_v(x_i))^2$$

M learners. $c()$ is the loss function, e.g., hinge loss.

Outline

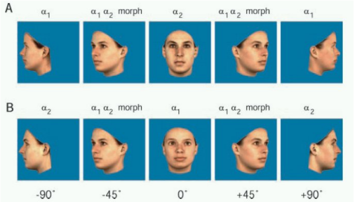
Do we learn from both labeled and unlabeled data?

Learning exists long before machine learning.

- Do humans perform semi-supervised learning?
- Yes, it seems. We discuss three human experiments:
    1. visual recognition with temporal association
    2. infant word-object mapping
    3. novel object categorization

Visual recognition with temporal association

- A face from two angles are very different, but we can easily associate it.
- The image sequence (unlabeled data) might be the glue.
- Artificial wrong sequences (person A's profile morphs to B's frontal) damage people's ability to match test profile and frontal images.

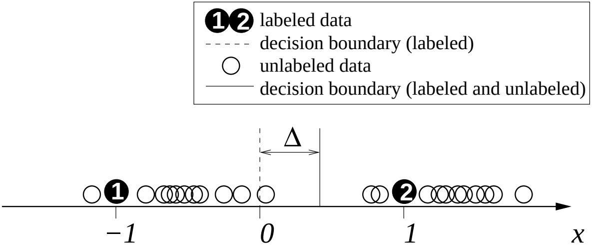


Infant word-object mapping

- 17-month infants listen to a word, see an object
- Measure their ability to associate the word and object
  - If the word heard many times before (without seeing the object; unlabeled data), association is stronger.
  - If the word not heard before, association is weaker.

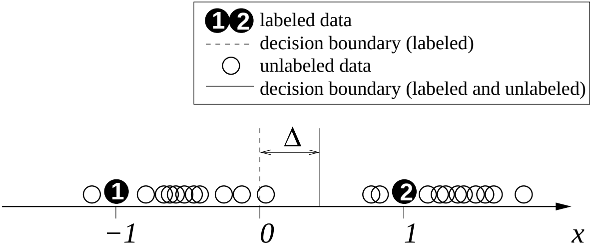
Similar to cluster-then-label.

Novel object categorization



- assuming each class is a coherent group (e.g. Gaussian)
- machine learning: decision boundary shift

Novel object categorization



- assuming each class is a coherent group (e.g. Gaussian)
- machine learning: decision boundary shift

Do we humans shift decision boundary too?

Human learning: a behavioral experiment

Determine human decision boundary

- labeled data only
- labeled and unlabeled data

Human learning: a behavioral experiment

Determine human decision boundary

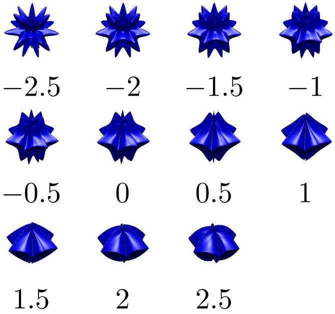
- labeled data only
- labeled and unlabeled data

Participants and materials

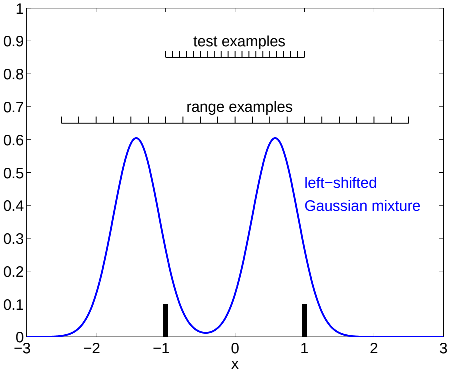
- 22 UW students
- told visual stimuli (examples) are microscopic pollens
- stimuli displayed one at a time
- press ‘b’ or ‘n’ to classify
- label is audio feedback

Visual stimuli

Stimuli parameterized by a continuous scalar x. Some examples:



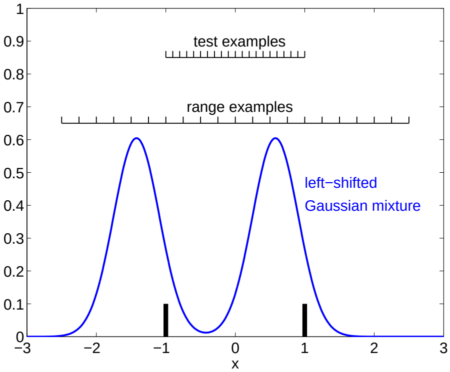
Experiment procedure



6 blocks

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

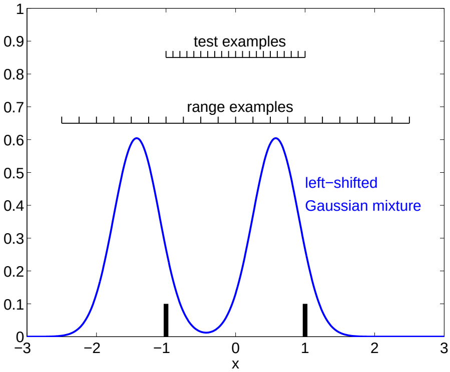
Experiment procedure



6 blocks

1. 20 labeled points at $x = -1, 1$
2. 21 test examples in $[-1, 1]$ (all unlabeled from now on)

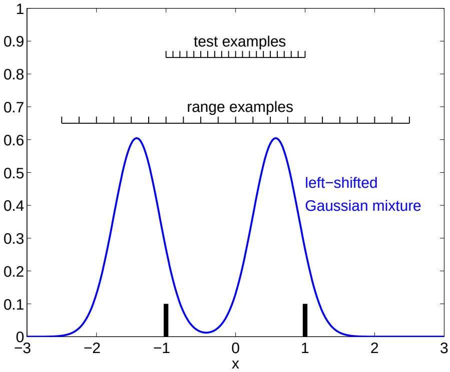
Experiment procedure



6 blocks

1. 20 labeled points at $x = -1, 1$
2. 21 test examples in $[-1, 1]$ (all unlabeled from now on)
3. 230 examples $\sim$ offset GMM, plus 21 range examples in $[-2.5, 2.5]$

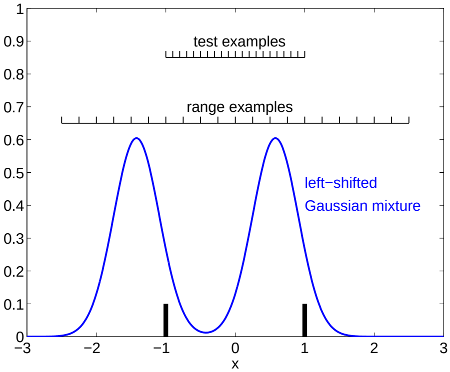
Experiment procedure



6 blocks

1. 20 labeled points at $x = -1, 1$
2. 21 test examples in $[-1, 1]$ (all unlabeled from now on)
3. 230 examples $\sim$ offset GMM, plus 21 range examples in $[-2.5, 2.5]$
4. similar to block 3
5. similar to block 3

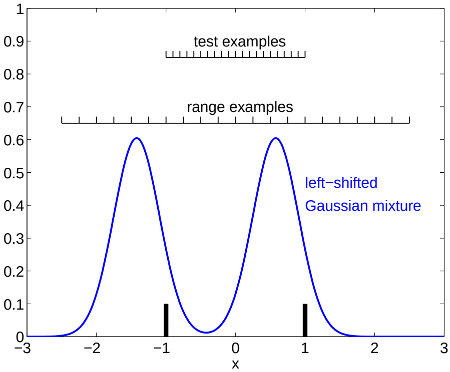
Experiment procedure



6 blocks

1. 20 labeled points at $x = -1, 1$
2. 21 test examples in $[-1, 1]$ (all unlabeled from now on)
3. 230 examples $\sim$ offset GMM, plus 21 range examples in $[-2.5, 2.5]$
4. similar to block 3
5. similar to block 3
6. 21 test examples in $[-1, 1]$ again

Experiment procedure

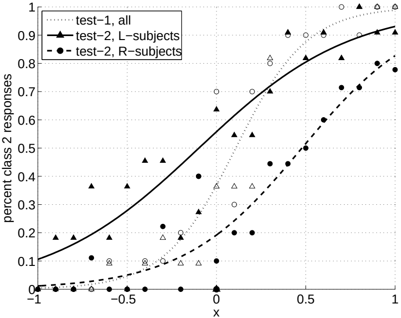


6 blocks

1. 20 labeled points at $x = -1, 1$
2. 21 test examples in $[-1, 1]$ (all unlabeled from now on)
3. 230 examples $\sim$ offset GMM, plus 21 range examples in $[-2.5, 2.5]$
4. similar to block 3
5. similar to block 3
6. 21 test examples in $[-1, 1]$ again

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

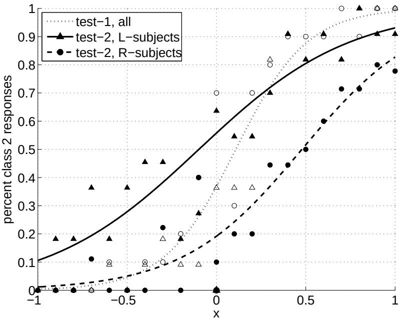
Observation 1: unlabeled data affects decision boundary



average decision boundary

- after seeing labeled data (block 2): $x = 0.11$

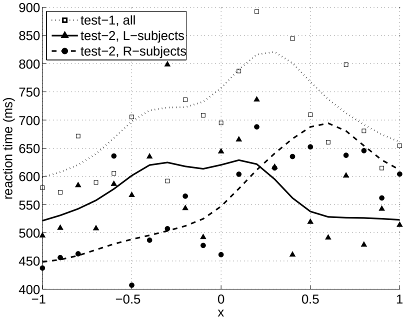
Observation 1: unlabeled data affects decision boundary



average decision boundary

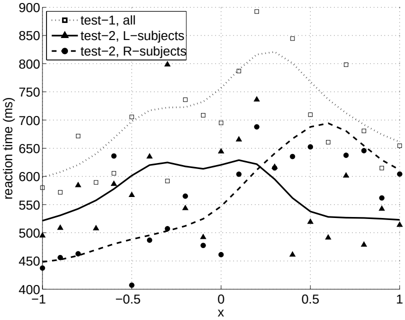
- after seeing labeled data (block 2): $x = 0.11$
- after seeing labeled and unlabeled data (block 6): L-subjects $x = -0.10$, R-subjects $x = 0.48$

Observation 2: unlabeled data affects reaction time



longer reaction time → harder example → closer to decision boundary

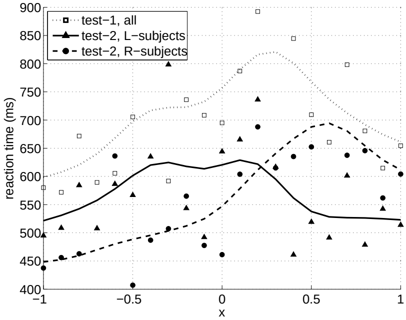
Observation 2: unlabeled data affects reaction time



longer reaction time → harder example → closer to decision boundary

- block 2: reaction time peak near $x = 0.11$

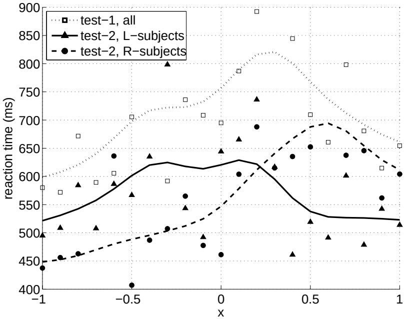
Observation 2: unlabeled data affects reaction time



longer reaction time → harder example → closer to decision boundary

- block 2: reaction time peak near $x = 0.11$
- block 6: overall faster, familiarity with experiment

Observation 2: unlabeled data affects reaction time



longer reaction time → harder example → closer to decision boundary

- block 2: reaction time peak near $x = 0.11$
- block 6: overall faster, familiarity with experiment
- L-subjects reaction time plateau around $x = -0.1$, R-subjects peak around $x = 0.6$

Reaction times too suggest decision boundary shift.

Machine learning: Gaussian Mixture Model

We can explain the human experiment with a semi-supervised machine learning model.

A Gaussian Mixture Model $\theta = \{w_1, \mu_1, \sigma_1^2, w_2, \mu_2, \sigma_2^2\}$ with 2 components

$$w_1 N(\mu_1, \sigma_1^2) + w_2 N(\mu_2, \sigma_2^2) \quad , w_1 + w_2 = 1, w_i \geq 0$$

Prior $w_k \sim \mathrm{Uniform}[0,1], \mu_k \sim \mathrm{N}(0, \infty), \sigma_k^2 \sim \mathrm{Inv}{-}\chi^2(\nu, s^2), k = 1, 2$
Data (assume: remember all, order independent)

$$D = \{(x_1, y_1), \ldots, (x_l, y_l), x_{l+1}, \ldots, x_n\}$$

Goal: find $\theta^{MAP} = \arg\max_\theta p(\theta) p(D|\theta)$

EM

Maximize the objective ($\lambda \leq 1$ weight on unlabeled example)

$$\log p(\theta) + \sum_{i=1}^{l} \log p(x_i, y_i|\theta) + \lambda \sum_{i=l+1}^{n} \log p(x_i|\theta)$$

E-step

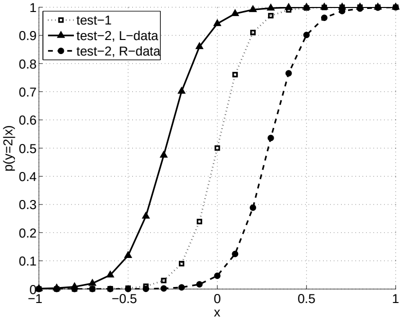
$$q_i(k) \propto w_k \mathrm{N}(x_i; \mu_k, \sigma_k^2), \quad i = l+1, \ldots, n; k = 1, 2$$

M-step

$$\begin{aligned}
\mu_k &= \frac{\sum_{i=1}^{l} \delta(y_i, k) x_i + \lambda \sum_{i=l+1}^{n} q_i(k) x_i}{\sum_{i=1}^{l} \delta(y_i, k) + \lambda \sum_{i=l+1}^{n} q_i(k)} \\
\sigma_k^2 &= \frac{\nu s^2 + \sum_{i=1}^{l} \delta(y_i, k) e_{ik} + \lambda \sum_{i=l+1}^{n} q_i(k) e_{ik}}{\nu + 2 + \sum_{i=1}^{l} \delta(y_i, k) + \lambda \sum_{i=l+1}^{n} q_i(k)} \\
w_k &= \frac{\sum_{i=1}^{l} \delta(y_i, k) + \lambda \sum_{i=l+1}^{n} q_i(k)}{l + \lambda(n - l)}
\end{aligned}$$

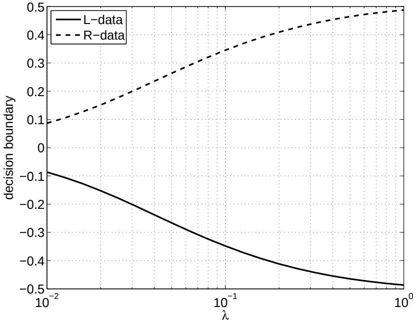
Model fitting result 1

GMM predicts decision boundary shift:



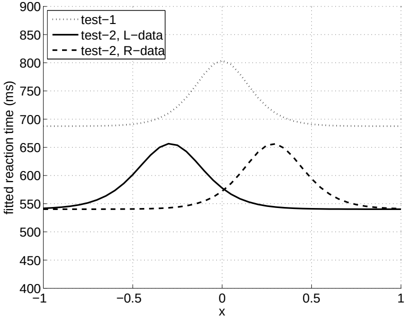
Model fitting result 2

Unlabeled data seem to worth less than labeled data ($\lambda = 0.06$)



Model fitting result 3

GMM explains reaction time:



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

Findings

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

Outline

1. Introduction to Semi-Supervised Learning
2. Semi-Supervised Learning Algorithms
   - Self Training
   - Generative Models
   - S3VMs
   - Graph-Based Algorithms
   - Multiview Algorithms
3. Semi-Supervised Learning in Nature
4. Some Challenges for Future Research

Challenge 0: Real SSL tasks

- What tasks can be dramatically improved by SSL, so that new functionalities are enabled?
- Move from two-moon to the real world

Challenge 1: New SSL assumptions

Generative models, multiview, graph methods, S3VMs

$$\sum_{i=1}^{l} \log p(y_i|\theta)p(x_i|y_i,\theta) + \lambda \sum_{i=l+1}^{n} \log\left(\sum_{y=1}^{c} p(y|\theta)p(x_i|y,\theta)\right)$$

$$\min_f \sum_{v=1}^{M}\left(\sum_{i=1}^{l} c(y_i, f_v(x_i)) + \lambda_1 \|f\|_K^2\right) + \lambda_2 \sum_{u,v=1}^{M}\sum_{i=l+1}^{n} \left(f_u(x_i) - f_v(x_i)\right)^2$$

$$\min_f \sum_{i=1}^{l} c(y_i, f(x_i)) + \lambda_1 \|f\|_K^2 + \lambda_2 \sum_{i,j=1}^{n} w_{ij}(f(x_i) - f(x_j))^2$$

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

Challenge 1: New SSL assumptions

What other assumptions can we make on unlabeled data? For example:

- label dissimilarity $y_i \neq y_j$

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

  w_{ij} edge confidence; $s_{ij} = 1$: same label, -1: different labels

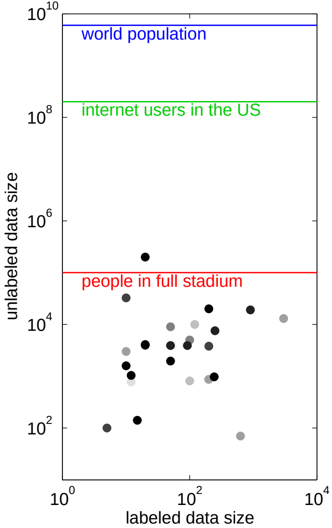
- order preference $y_i - y_j \geq d$ for regression

$$(d - (f(x_i) - f(x_j))_+$$

New assumptions may lead to new SSL algorithms.

Challenge 2: Efficiency on huge unlabeled datasets

Some recent SSL datasets as reported in research papers:



Challenge 3: Safe SSL

- no pain, no gain

Challenge 3: Safe SSL

- no pain, no gain
- no model assumption, no gain

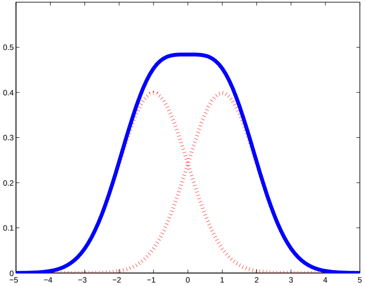
Challenge 3: Safe SSL

- no pain, no gain
- no model assumption, no gain
- wrong model assumption, no gain, a lot of pain

Challenge 3: Safe SSL

- no pain, no gain
- no model assumption, no gain
- wrong model assumption, no gain, a lot of pain

An example where S3VM, graph methods will not work, but GMM will:



Challenge 3: Safe SSL

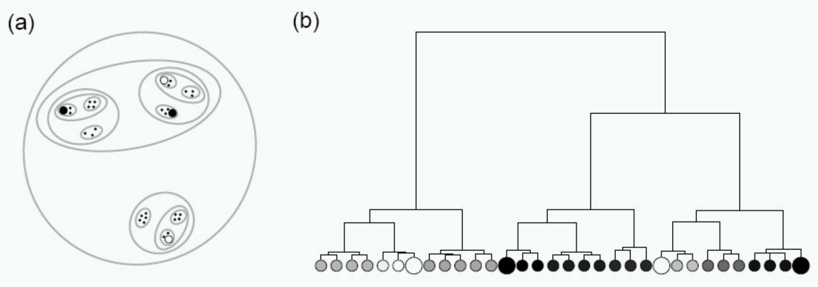
- How do we know that we are making the right model assumptions?
- Which semi-supervised learning method should I use?
- If I have labeled AND unlabeled data, I should do at least as well as only having the labeled data.

How can we make sure that SSL is "safe"?

Challenge 4: What can we borrow from Natural Learning?

Example: Semi-supervised learning with trees

- Tree over labeled and unlabeled data (inspired by taxonomy)
- Label mutation process over the edges defines a prior



References

1. Olivier Chapelle, Alexander Zien, Bernhard Schölkopf (Eds.). (2006). *Semi-supervised learning*. MIT Press.
2. Xiaojin Zhu (2005). *Semi-supervised learning literature survey*. TR-1530. University of Wisconsin-Madison Department of Computer Science.
3. Matthias Seeger (2001). *Learning with labeled and unlabeled data*. Technical Report. University of Edinburgh.

... and the references therein.

Thank you