Semi-Supervised Learning: An Overview

深圳大学

SHENZHEN UNIVERSITY

Muhammed Jamshed Alam Patwary

PhD Research Fellow, Big Data Institute

College of Computer Science and Software Engineering

Shenzhen University

Outline

- Introduction to Semi-Supervised Learning
- Semi-Supervised Learning Algorithms
  - Self-training
  - EM with generative mixture models
  - Co-training
  - Fuzziness based Semi-Supervised Learning
  - Transductive support vector machine
- Which semi-supervised learning method should I use?
- Some Challenges for Future Research

Disclaimer:

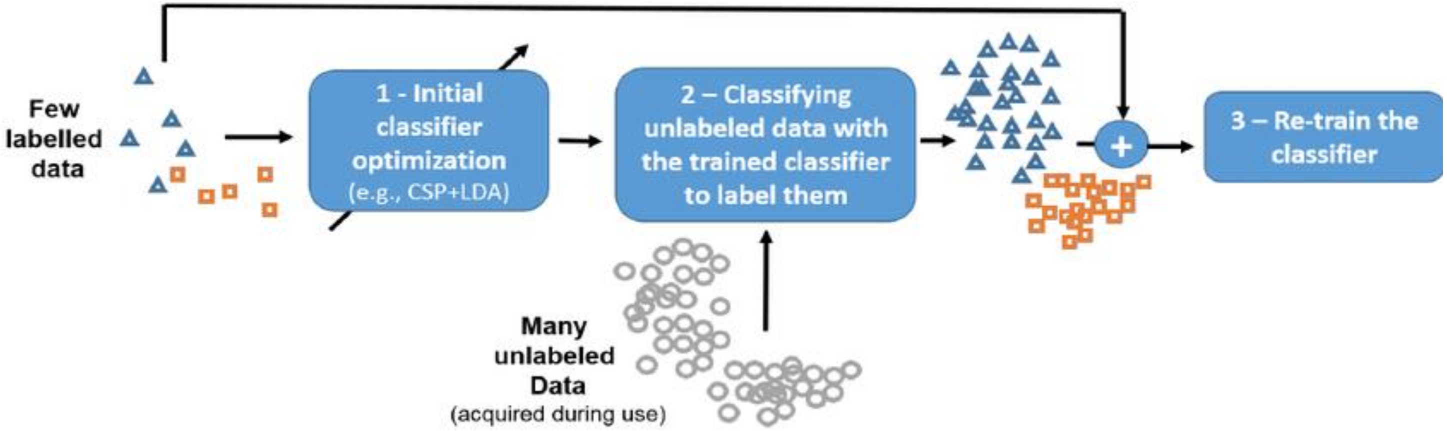
Some of the pictures and slides are taken from Xiaojin Zhu's(University of Wisconsin, Madison, USA) presentation slides.

Introduction to Semi-Supervised Learning

The Traditional View:

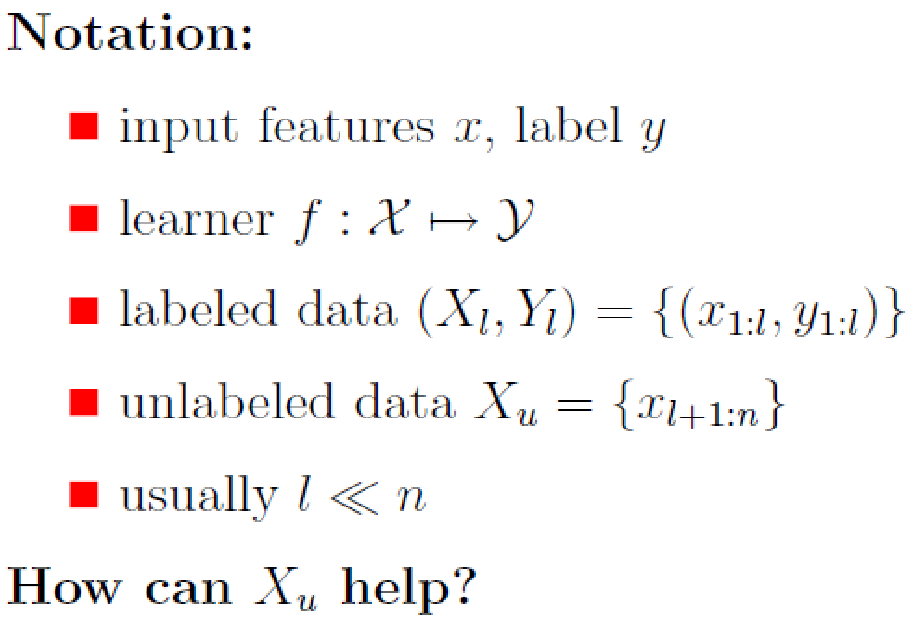
- Labeled instances are difficult to get
  - Expensive and time consuming to obtain.
  - They require the effort of experienced human annotator.
- Unlabeled data is cheap

- **Semi-supervised learning** is a class of supervised learning tasks and techniques that also make use of unlabeled data for training
- 1965, Scudder


Few labelled data
1 - Initial classifier optimization (e.g., CSP+LDA)
2 – Classifying unlabeled data with the trained classifier to label them
+
3 – Re-train the classifier
Many unlabeled Data (acquired during use)

Introduction to Semi-Supervised Learning

- Why Semi-supervised learning?
- The learning problem
  - Goal: Using both labeled and unlabeled data to build better learners, then using each one alone.



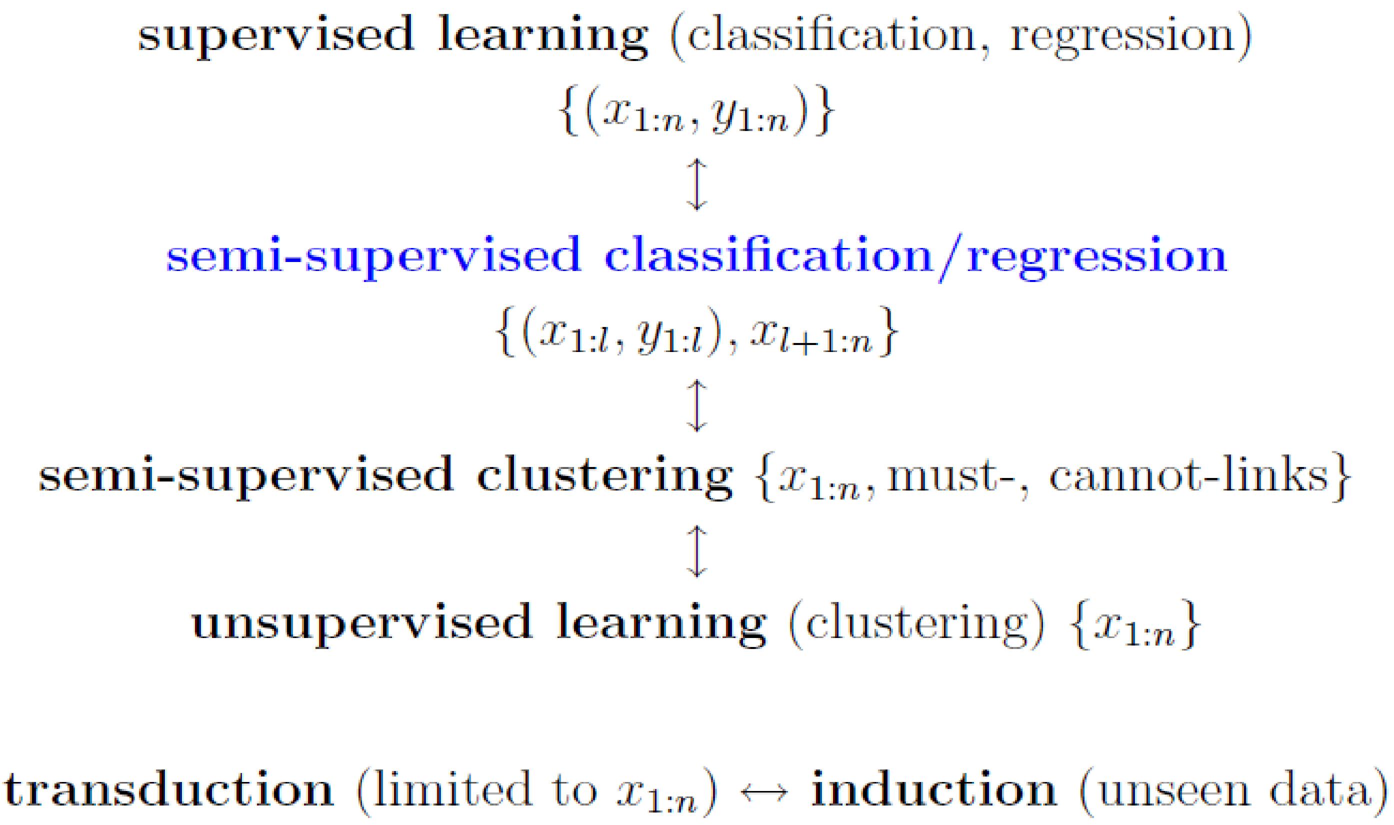
Notation:

- input features x, label y
- learner $f : \mathcal{X} \mapsto \mathcal{Y}$
- labeled data $(X_l, Y_l) = \{(x_{1:l}, y_{1:l})\}$
- unlabeled data $X_u = \{x_{l+1:n}\}$
- usually $l \ll n$

How can X_u help?

Introduction to Semi-Supervised Learning

- The landscape



supervised learning (classification, regression)

$\{(x_{1:n}, y_{1:n})\}$

$\updownarrow$

semi-supervised classification/regression

$\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

$\updownarrow$

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

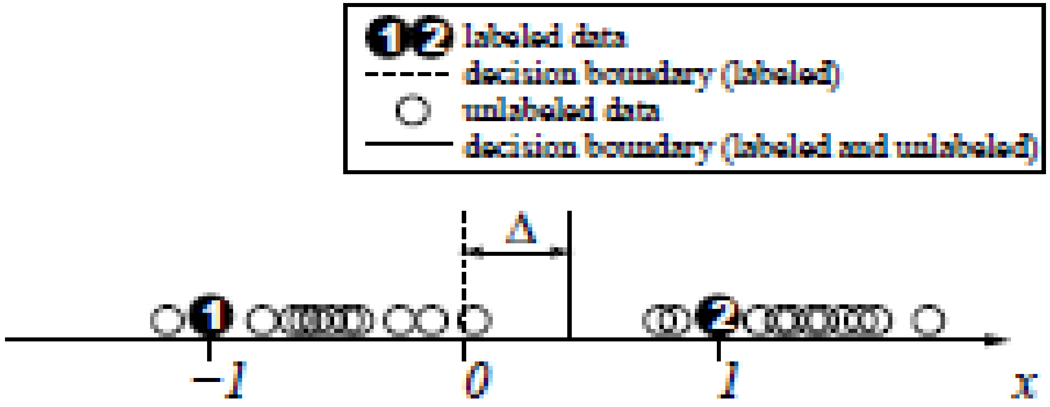
$\updownarrow$

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

transduction (limited to $x_{1:n}$) $\leftrightarrow$ **induction** (unseen data)

Introduction to Semi-Supervised Learning

How can unlabeled data ever help?

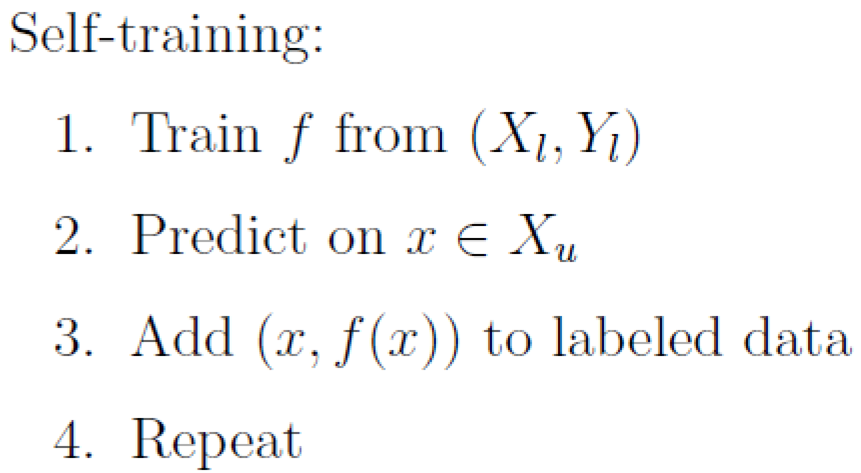


1 2 labeled data
decision boundary (labeled)
unlabeled data
decision boundary (labeled and unlabeled)
Δ
−1
0
1
x

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

Does unlabeled data always help?

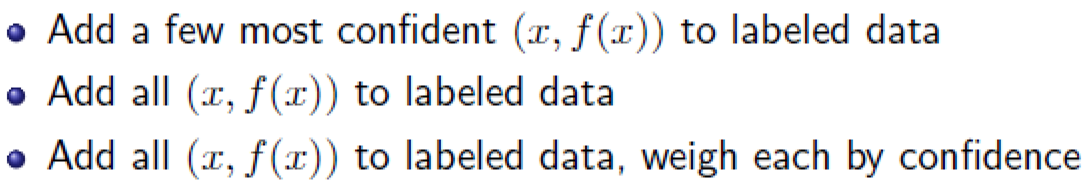
Semi-Supervised Learning Algorithms-Self-training



Self-training:

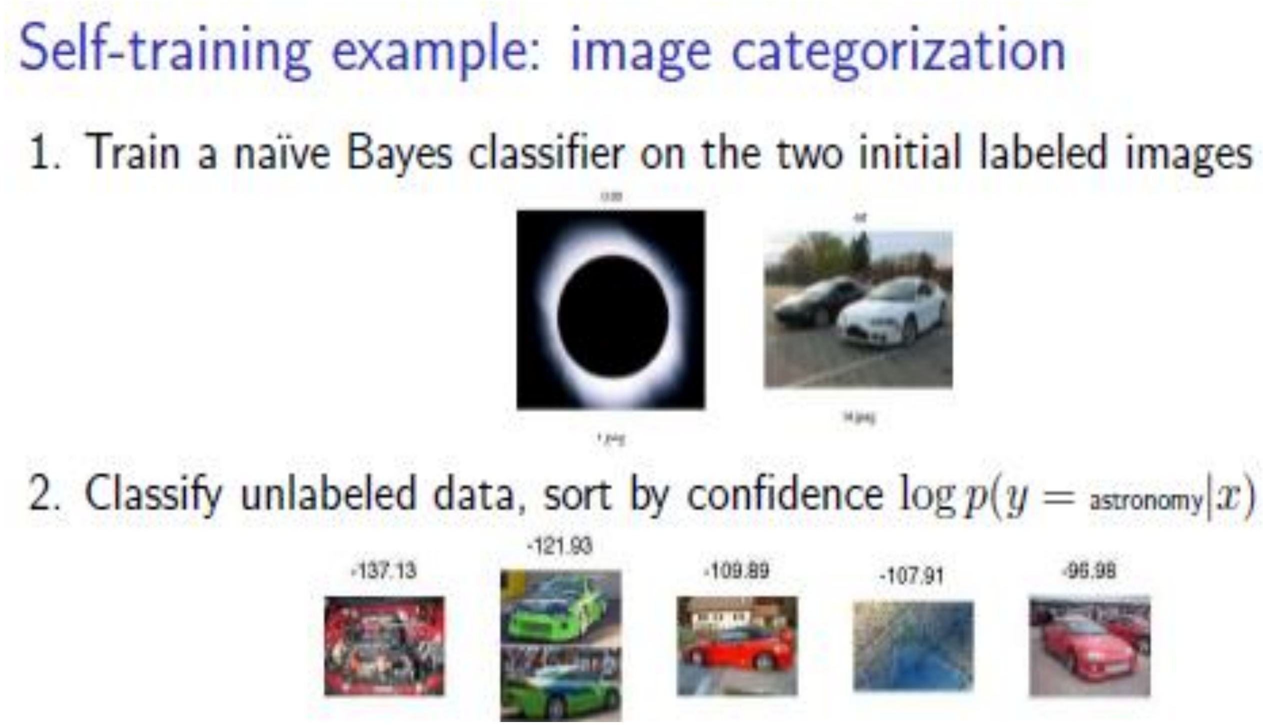
1. Train f from (X_l, Y_l)
2. Predict on $x \in X_u$
3. Add $(x, f(x))$ to labeled data
4. Repeat

- 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

Semi-Supervised Learning Algorithms-Self-training

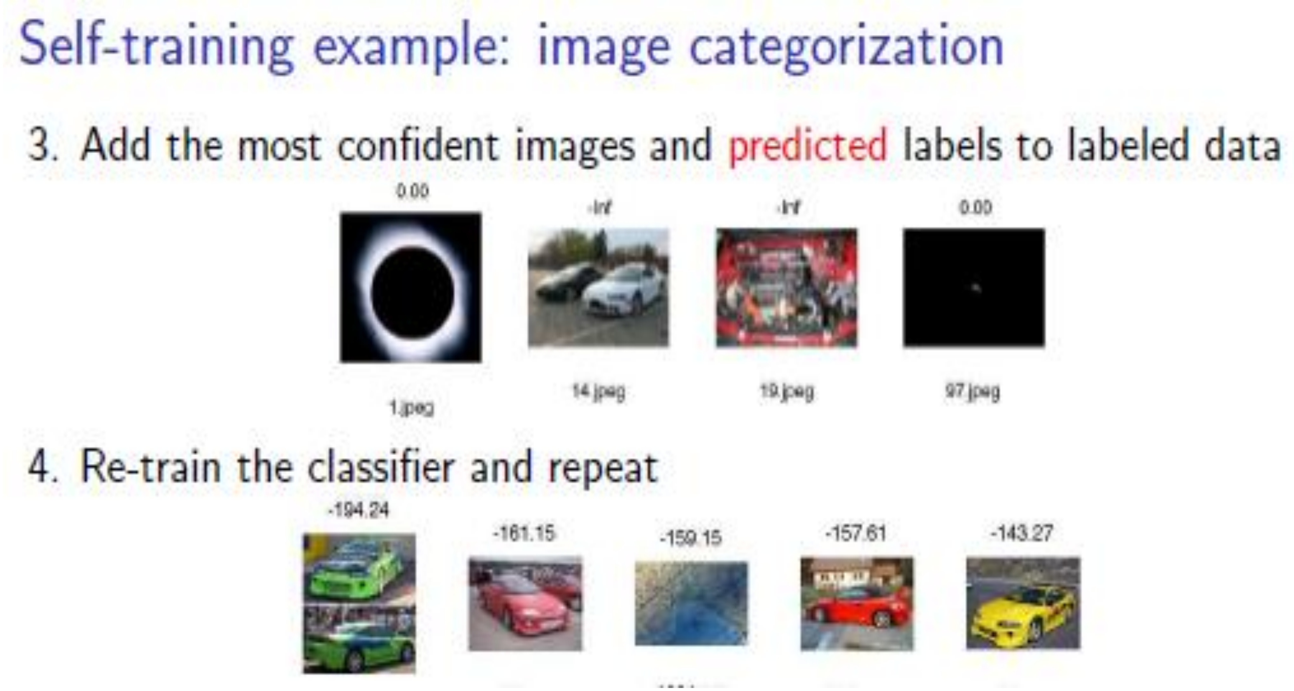


Self-training example: image categorization

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

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

Semi-Supervised Learning Algorithms-Self-training



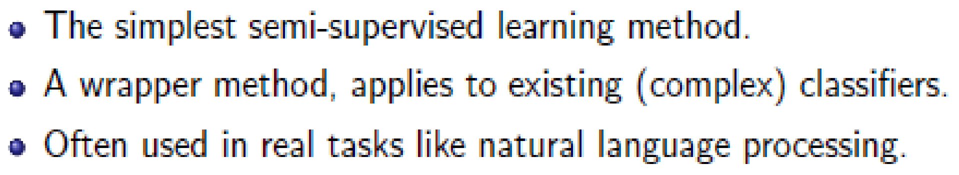
Self-training example: image categorization

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

4. Re-train the classifier and repeat

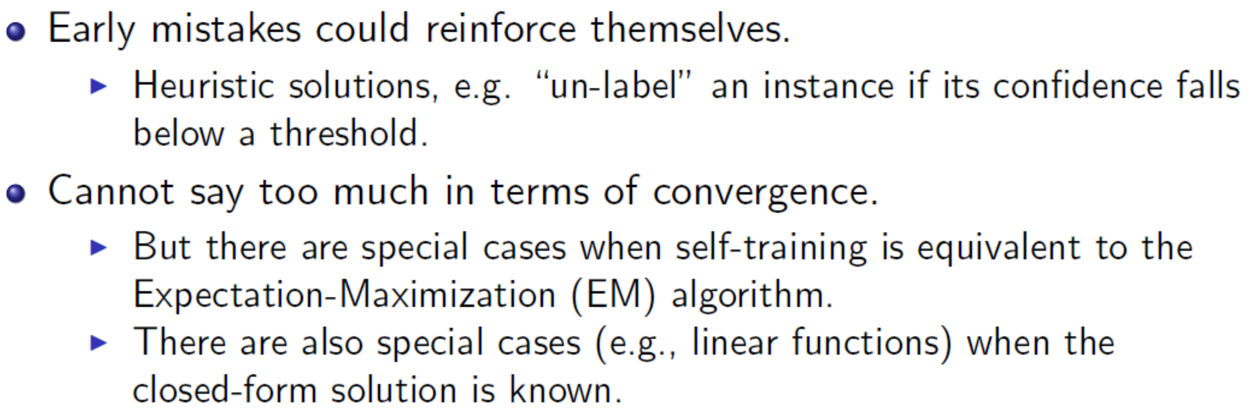
Semi-Supervised Learning Algorithms-Self-training

- 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

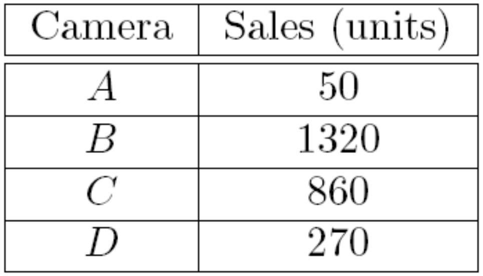


- Early mistakes could reinforce themselves.
  - Heuristic solutions, e.g. "un-label" an instance if its confidence falls below a threshold.
- Cannot say too much in terms of convergence.
  - But there are special cases when self-training is equivalent to the Expectation-Maximization (EM) algorithm.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

- **EM with generative mixture models**

Fuzzy Cluster

- In hard clustering methods
  - Every data object is assigned to exactly one cluster
- Some applications may need for fuzzy or soft cluster assignment
  - Ex. An e-game could belong to both entertainment and software
- Example: Popularity of cameras is defined as a fuzzy mapping



Camera	Sales (units)
A	50
B	1320
C	860
D	270

$$\mathrm{Pop}(o) = \begin{cases} 1 & \text{if } 1,000 \text{ or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \ (i < 1000) \text{ units of } o \text{ are sold} \end{cases}$$

- Then, *A*(0.05), *B*(1), *C*(0.86), *D*(0.27)

Fuzzy (Soft) Clustering

Review-id	Keywords
R_1	digital camera, lens
R_2	digital camera
R_3	lens
R_4	digital camera, lens, computer
R_5	computer, CPU
R_6	computer, computer game

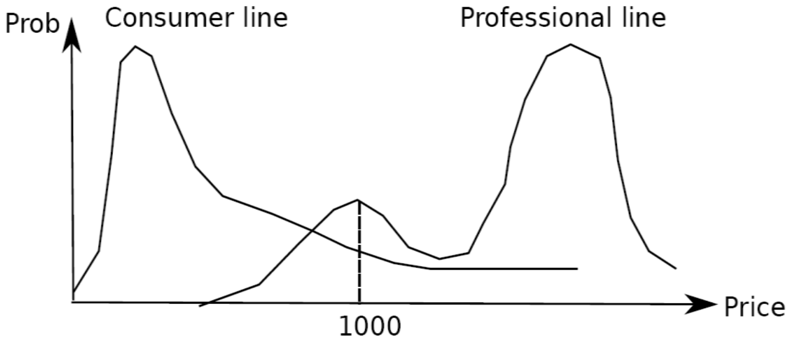
$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

- Example: Let cluster features be
  - C_1 :“digital camera” and “lens”
  - C_2: “computer“
- Fuzzy clustering
  - k fuzzy clusters C_1, …,C_k ,represented as a partition matrix M = $[w_{ij}]$
  - P1: for each object o_i and cluster C_j, $0 \le w_{ij} \le 1$ (fuzzy set)
  - P2: for each object o_i, $\sum^{k} w_{ij} = 1$, equal participation in the clustering
  - P3: for each cluster C_j , $0 < \sum_{i=1}^{n} w_{ij} < n$ ensures there is no empty cluster
- Let c_1, …, c_k as the center of the k clusters
- For an object o_i, sum of the squared error (SSE), p is a parameter:
- For a cluster C_i, SSE: $\mathrm{SSE}(C_j) = \sum_{i=1}^{n} w_{ij}^{p} dist(o_i, c_j)^2$ $\quad \mathrm{SSE}(o_i) = \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$
- Measure how well a clustering fits the data:

$$\mathrm{SSE}(\mathcal{C}) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$$

Probabilistic Model-Based Clustering

- Cluster analysis is to find hidden categories.
- A hidden category (i.e., *probabilistic cluster)* is a distribution over the data space, which can be mathematically represented using a probability density function (or distribution function).
- Ex. 2 categories for digital cameras sold
  - consumer line vs. professional line
  - density functions f_1, f_2 for C_1, C_2
  - obtained by probabilistic clustering



- A **mixture model** assumes that a set of observed objects is a mixture of instances from multiple probabilistic clusters, and conceptually each observed object is generated independently
- **Our task**: infer a set of *k* probabilistic clusters that is most likely to generate *D* using the above data generation process

Probabilistic Model-Based Clustering

- A set *C* of *k* probabilistic clusters $C_1, \ldots, C_k$ with probability density functions $f_1, \ldots, f_k$, respectively, and their probabilities $\omega_1, \ldots, \omega_k$.
- Probability of an object *o* generated by cluster C_j is $P(o|C_j) = \omega_j f_j(o)$
- Probability of *o* generated by the set of cluster ***C*** is $P(o|\mathbf{C}) = \sum_{j=1}^{k} \omega_j f_j(o)$
- Since objects are assumed to be generated independently, for a data set D = {$o_1, \ldots, o_n$}, we have,

$$P(D|\mathbf{C}) = \prod_{i=1}^{n} P(o_i|\mathbf{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

- Task: Find a set *C* of *k* probabilistic clusters s.t. *P*(*D*|***C***) is maximized
- However, maximizing *P*(*D*|***C***) is often intractable since the probability density function of a cluster can take an arbitrarily complicated form
- To make it computationally feasible (as a compromise), assume the probability density functions being some parameterized distributions

Univariate Gaussian Mixture Model

- O = {o_1, …, o_n} (n observed objects), Θ = {θ_1, …, θ_k} (parameters of the k distributions), and $P_j(o_i| \theta_j)$ is the probability that o_i is generated from the j-th distribution using parameter θ_j, we have

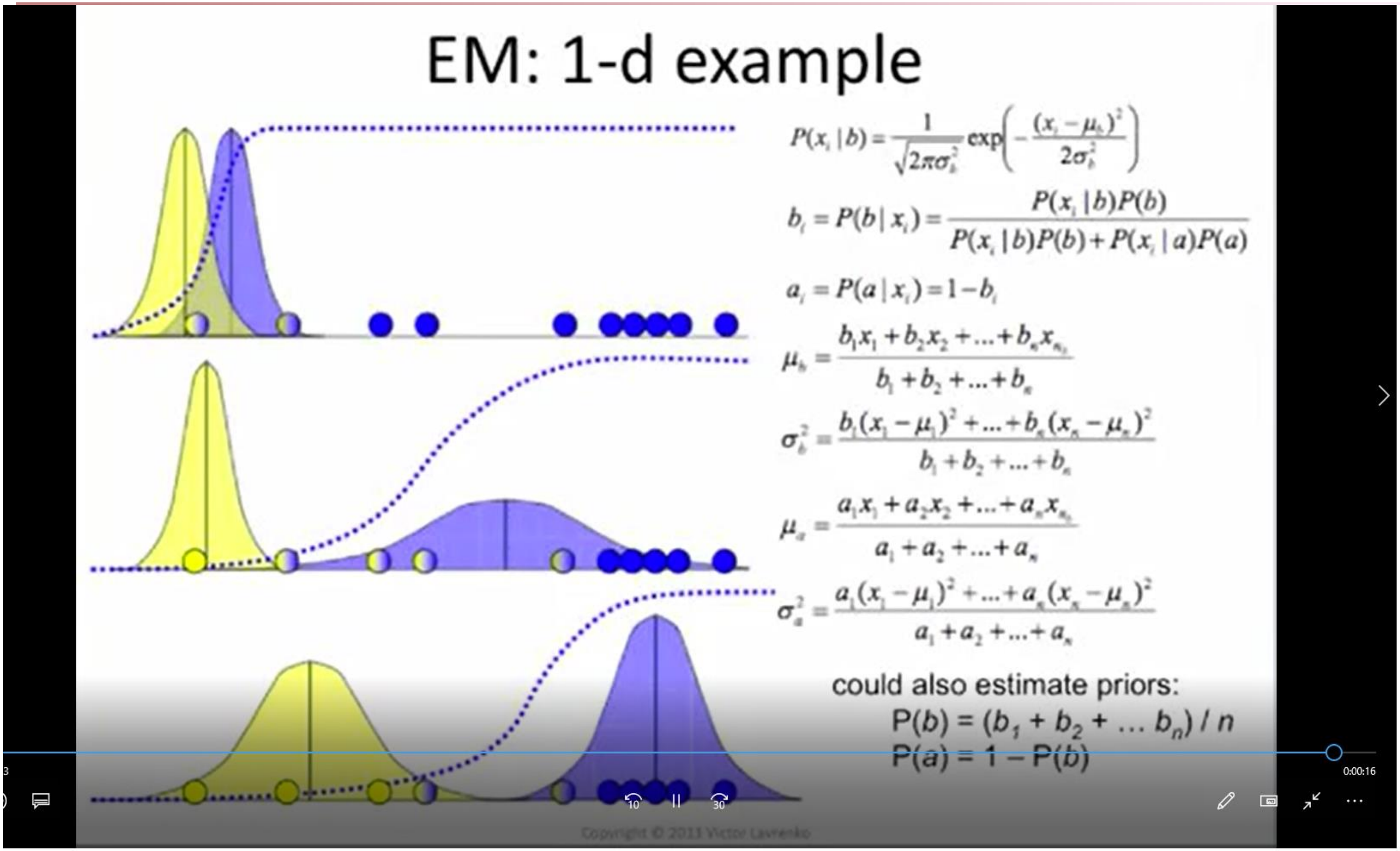
$$P(o_i|\mathbf{\Theta}) = \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j) \qquad P(\mathbf{O}|\mathbf{\Theta}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are k clusters.
  - The probability density function of each cluster are centered at μ_j with standard deviation σ_j, θ_j, = (μ_j, σ_j), we have

$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}} \qquad P(o_i|\mathbf{\Theta}) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

$$P(\mathbf{O}|\mathbf{\Theta}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

Univariate Gaussian Mixture Model



EM: 1-d example

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i-\mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1x_1 + b_2x_2 + \ldots + b_nx_n}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1(x_1-\mu_1)^2 + \ldots + b_n(x_n-\mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1x_1 + a_2x_2 + \ldots + a_nx_n}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1(x_1-\mu_1)^2 + \ldots + a_n(x_n-\mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

could also estimate priors:

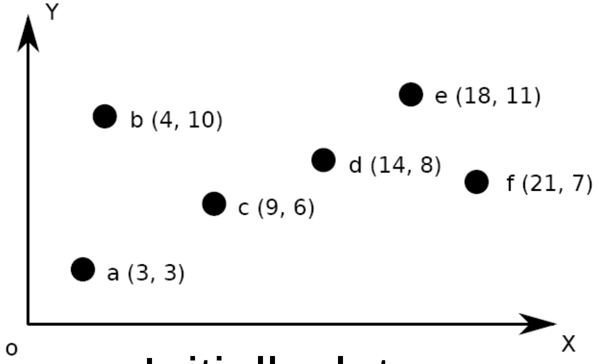
$P(b) = (b_1 + b_2 + \ldots b_n) / n$

$P(a) = 1 - P(b)$

The EM (Expectation Maximization) Algorithm

- The k-means algorithm has two steps at each iteration:
  - **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*
  - **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized
- **The (EM) algorithm:** A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.
  - **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
  - **M-step** finds the new clustering or parameters that maximize the sum of squared error (SSE) or the expected likelihood

Fuzzy Clustering Using the EM Algorithm



Iteration	E-step	M-step
1	$M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$	$c_1 = (8.47, 5.12),$ $c_2 = (10.42, 8.99)$
2	$M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$	$c_1 = (8.51, 6.11),$ $c_2 = (14.42, 8.69)$
3	$M^T = \begin{bmatrix} 0.80 & 0.76 & 0.99 & 0.02 & 0.14 & 0.23 \\ 0.20 & 0.24 & 0.01 & 0.98 & 0.86 & 0.77 \end{bmatrix}$	$c_1 = (6.40, 6.24),$ $c_2 = (16.55, 8.64)$

- Initially, let c_1 = a and c_2 = b
- 1st E-step: assign o to c_1,w. wt = $\frac{\frac{1}{dist(o,c_1)^2}}{\frac{1}{dist(o,c_1)^2} + \frac{1}{dist(o,c_2)^2}} = \frac{dist(o,c_2)^2}{dist(o,c_1)^2 + dist(o,c_2)^2}$
  - $w_{c,c_1} = \frac{41}{45+41} = 0.48$
- 1st M-step: recalculate the centroids according to the partition matrix, minimizing the sum of squared error (SSE)

$$c_j = \frac{\sum\limits_{\text{each point } o} w_{o,c_j}^2 o}{\sum\limits_{\text{each point } o} w_{o,c_j}^2}$$

$$c_1 = \left(\frac{1^2\times 3+0^2\times 4+0.48^2\times 9+0.42^2\times 14+0.41^2\times 18+0.47^2\times 21}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}, \frac{1^2\times 3+0^2\times 10+0.48^2\times 6+0.42^2\times 8+0.41^2\times 11+0.47^2\times 7}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}\right) = (8.47, 5.12)$$

- Iteratively calculate this until the cluster centers converge or the change is small enough

Computing Mixture Models with EM

- Given n objects O = {o_1, …, o_n}, we want to mine a set of parameters Θ = {θ_1, …, θ_k} s.t.,P(**O**|**Θ**) is maximized, where $\theta_j = (\mu_j, \sigma_j)$ are the mean and standard deviation of the j-th univariate Gaussian distribution
- We initially assign random values to parameters θ_j, then iteratively conduct the E- and M- steps until converge or sufficiently small change
- At the E-step, for each object o_i, calculate the probability that o_i belongs to each distribution,

$$P(\Theta_j|o_i,\mathbf{\Theta}) = \frac{P(o_i|\Theta_j)}{\sum_{l=1}^{k} P(o_i|\Theta_l)}$$

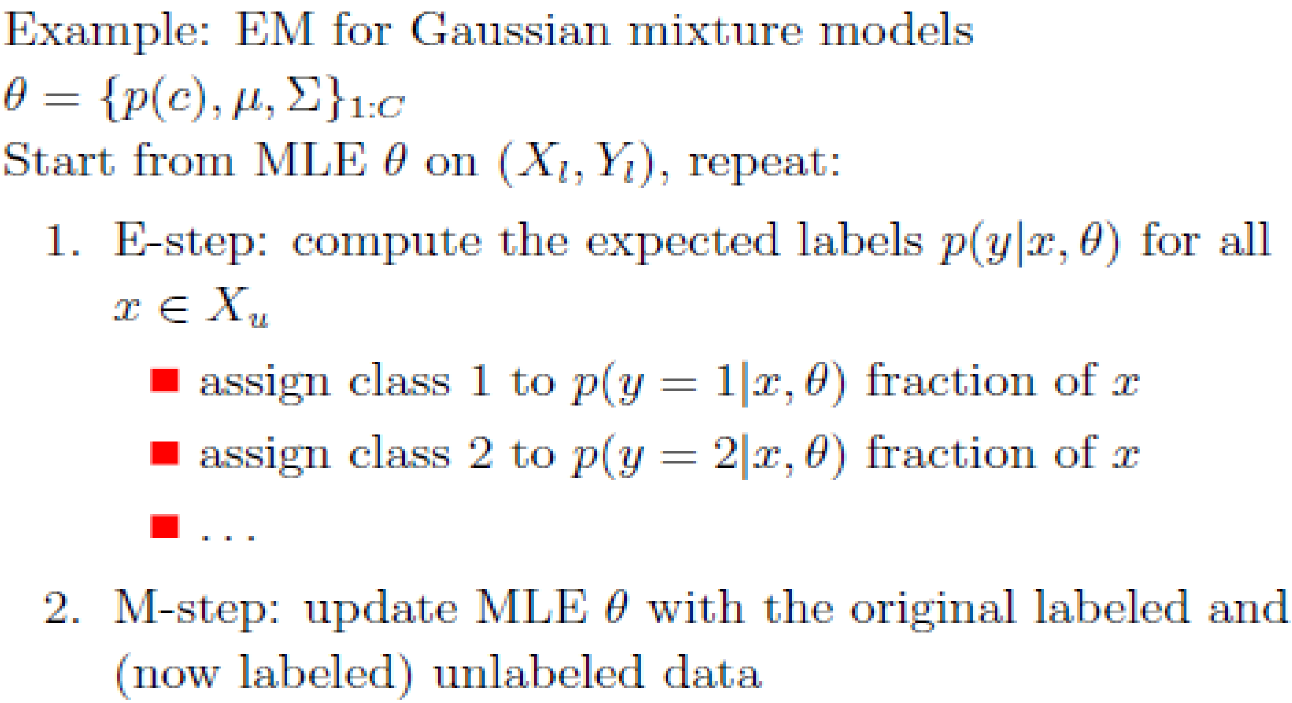
- At the M-step, adjust the parameters $\theta_j = (\mu_j, \sigma_j)$ so that the expected likelihood P(**O**|**Θ**) is maximized

$$\mu_j = \sum_{i=1}^{n} o_i \frac{P(\Theta_j|o_i,\mathbf{\Theta})}{\sum_{l=1}^{n} P(\Theta_j|o_l,\mathbf{\Theta})} = \frac{\sum_{i=1}^{n} o_i P(\Theta_j|o_i,\mathbf{\Theta})}{\sum_{i=1}^{n} P(\Theta_j|o_i,\mathbf{\Theta})} \qquad \sigma_j = \sqrt{\frac{\sum_{i=1}^{n} P(\Theta_j|o_i,\mathbf{\Theta})(o_i - u_j)^2}{\sum_{i=1}^{n} P(\Theta_j|o_i,\mathbf{\Theta})}}$$

Advantages and Disadvantages of Mixture Models

- Strength
  - Mixture models are more general than partitioning and fuzzy clustering
  - Clusters can be characterized by a small number of parameters
  - The results may satisfy the statistical assumptions of the generative models
- Weakness
  - Converge to local optimal (overcome: run multi-times w. random initialization)
  - Computationally expensive if the number of distributions is large, or the data set contains very few observed data points
  - Need large data sets
  - Hard to estimate the number of clusters

EM with generative mixture model



Example: EM for Gaussian mixture models

$\theta = \{p(c), \mu, \Sigma\}_{1:C}$

Start from MLE θ on (X_l, Y_l), repeat:

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

EM with generative mixture model

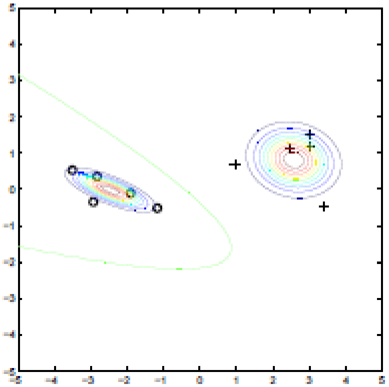
The MLE of θ without and with X_u is different.

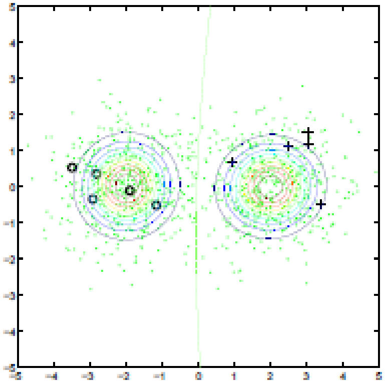
labeled data only

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

labeled and unlabeled

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



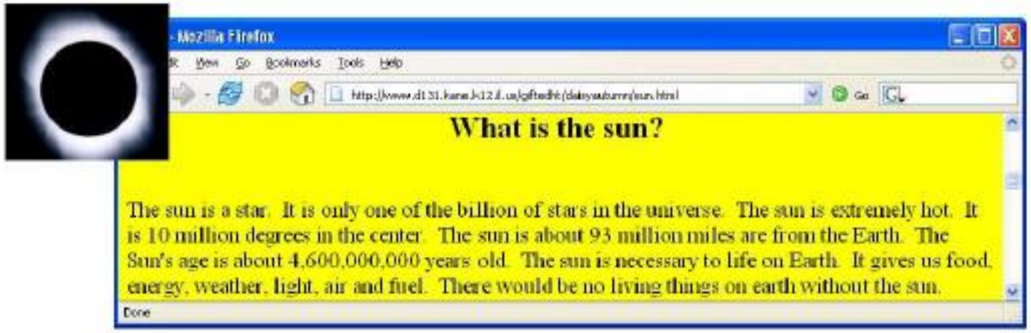


In principle X_u is useful for other generative models too.

Co-training

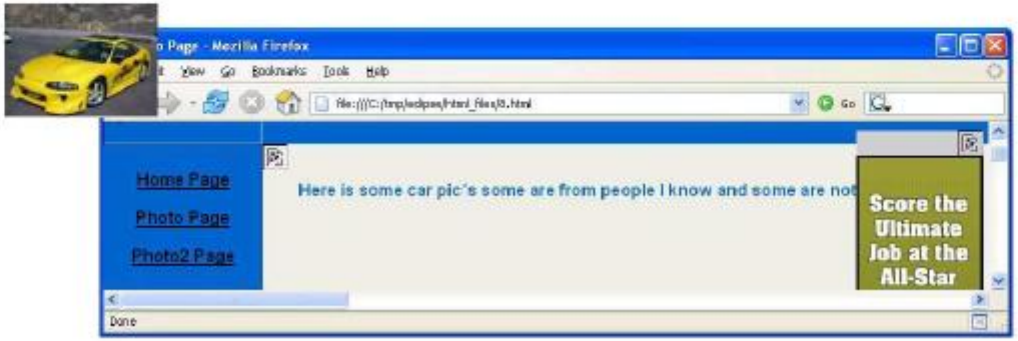
Co-training

Two views of an item: image and HTML text


- Mozilla Firefox
View Go Bookmarks Tools Help
http://www.d131.kana.k12.il.us/giftedht/daisyauburn/sun.html
Go
Done

What is the sun?

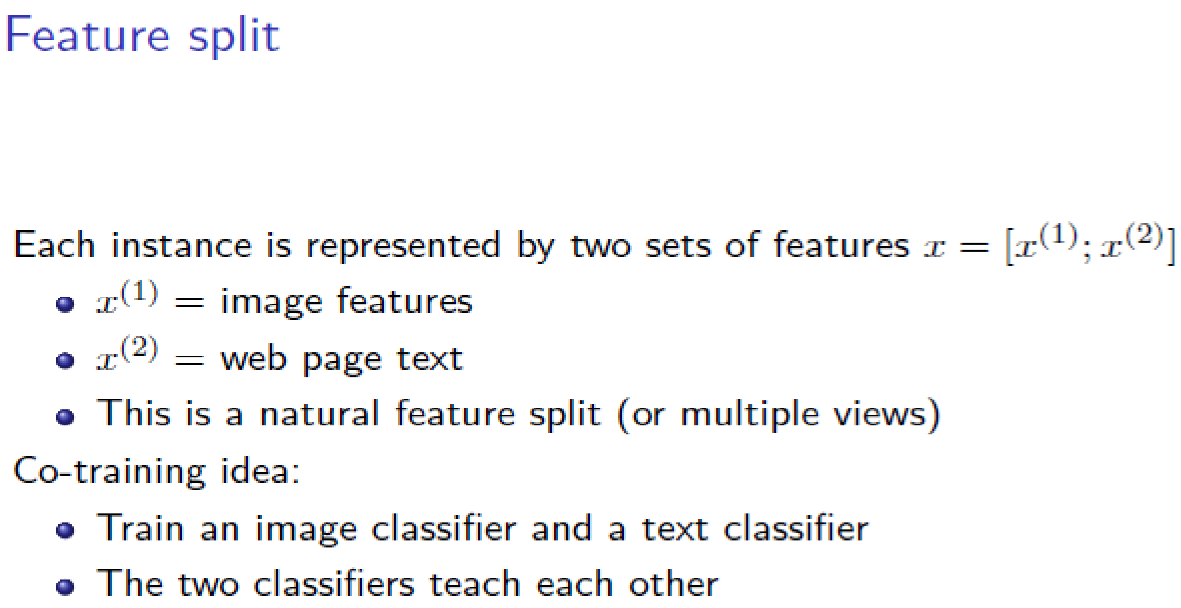
The sun is a star. It is only one of the billion of stars in the universe. The sun is extremely hot. It is 10 million degrees in the center. The sun is about 93 million miles are from the Earth. The Sun's age is about 4,600,000,000 years old. The sun is necessary to life on Earth. It gives us food, energy, weather, light, air and fuel. There would be no living things on earth without the sun.


o Page - Mozilla Firefox
View Go Bookmarks Tools Help
file:///C:/tmp/eclipse/html_files/8.html
Go
Home Page
Photo Page
Photo2 Page
Done

Here is some car pic's some are from people I know and some are not

Score the Ultimate Job at the All-Star

Co-training



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:

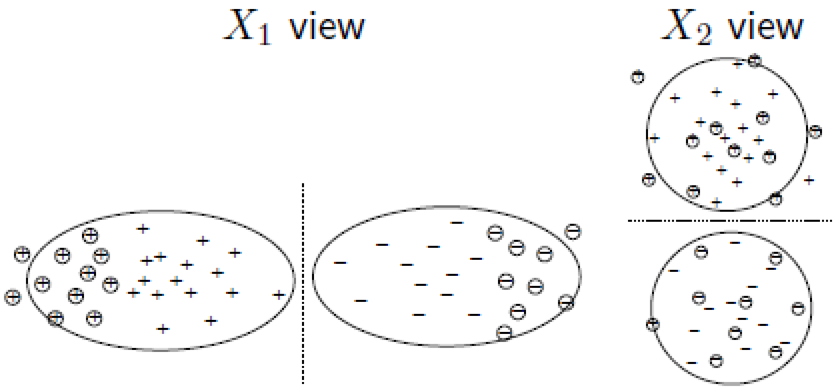
- Train an image classifier and a text classifier
- The two classifiers teach each other

Co-training

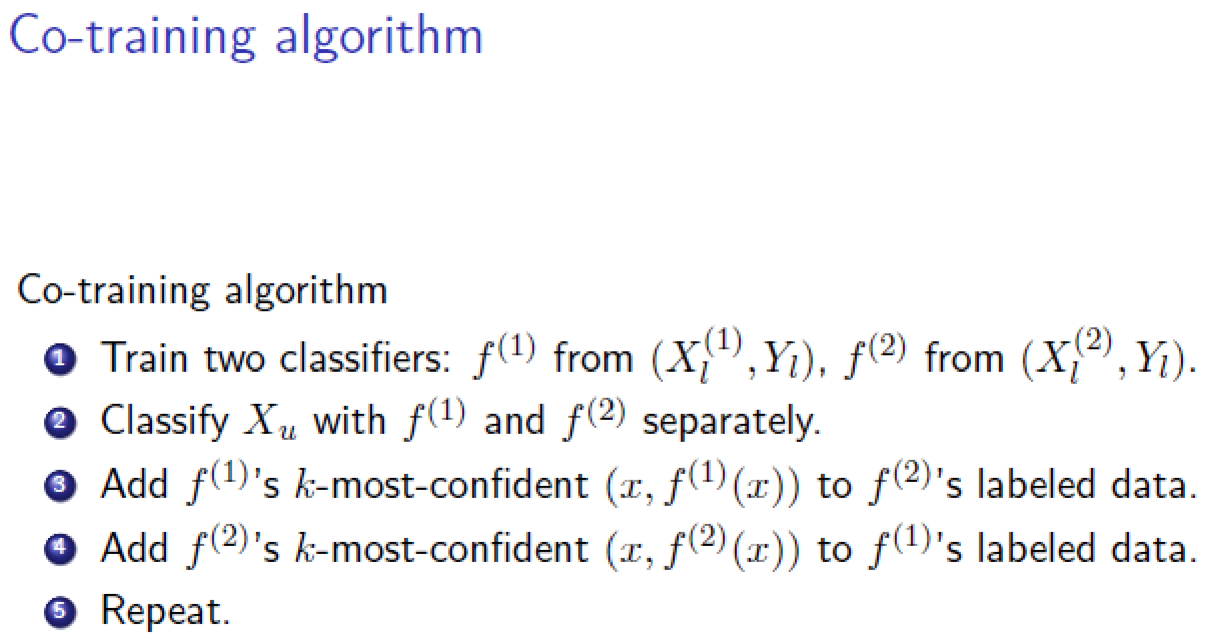
Co-training assumptions

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

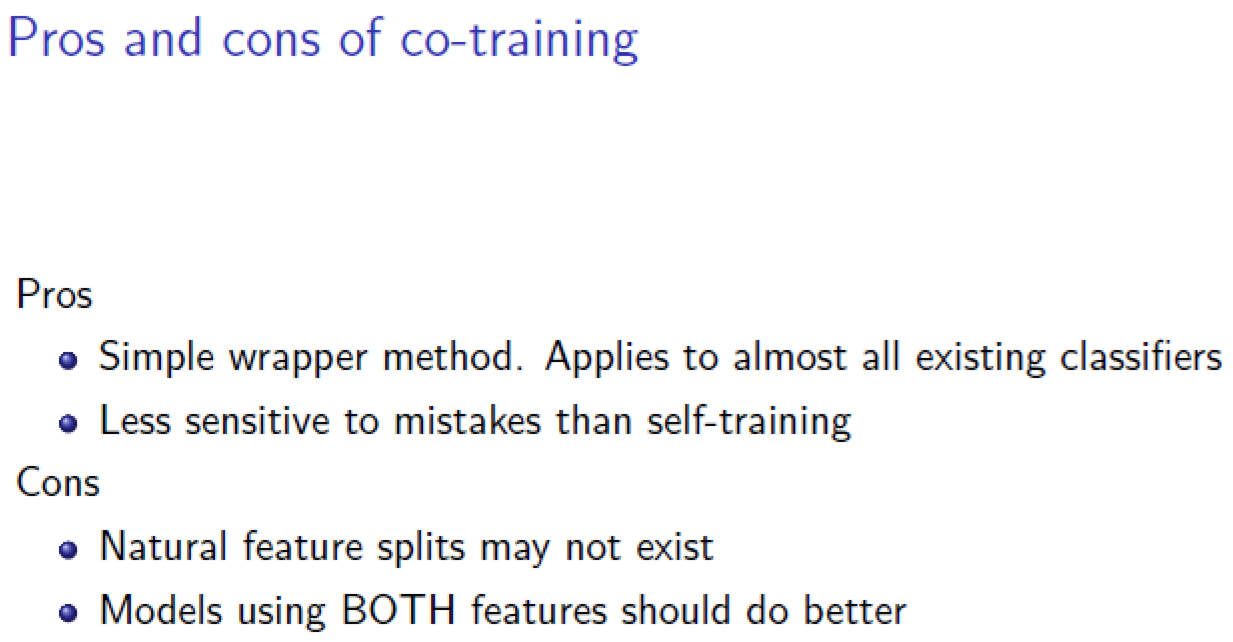


Co-training algorithm

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.

Co-training



Pros and cons of co-training

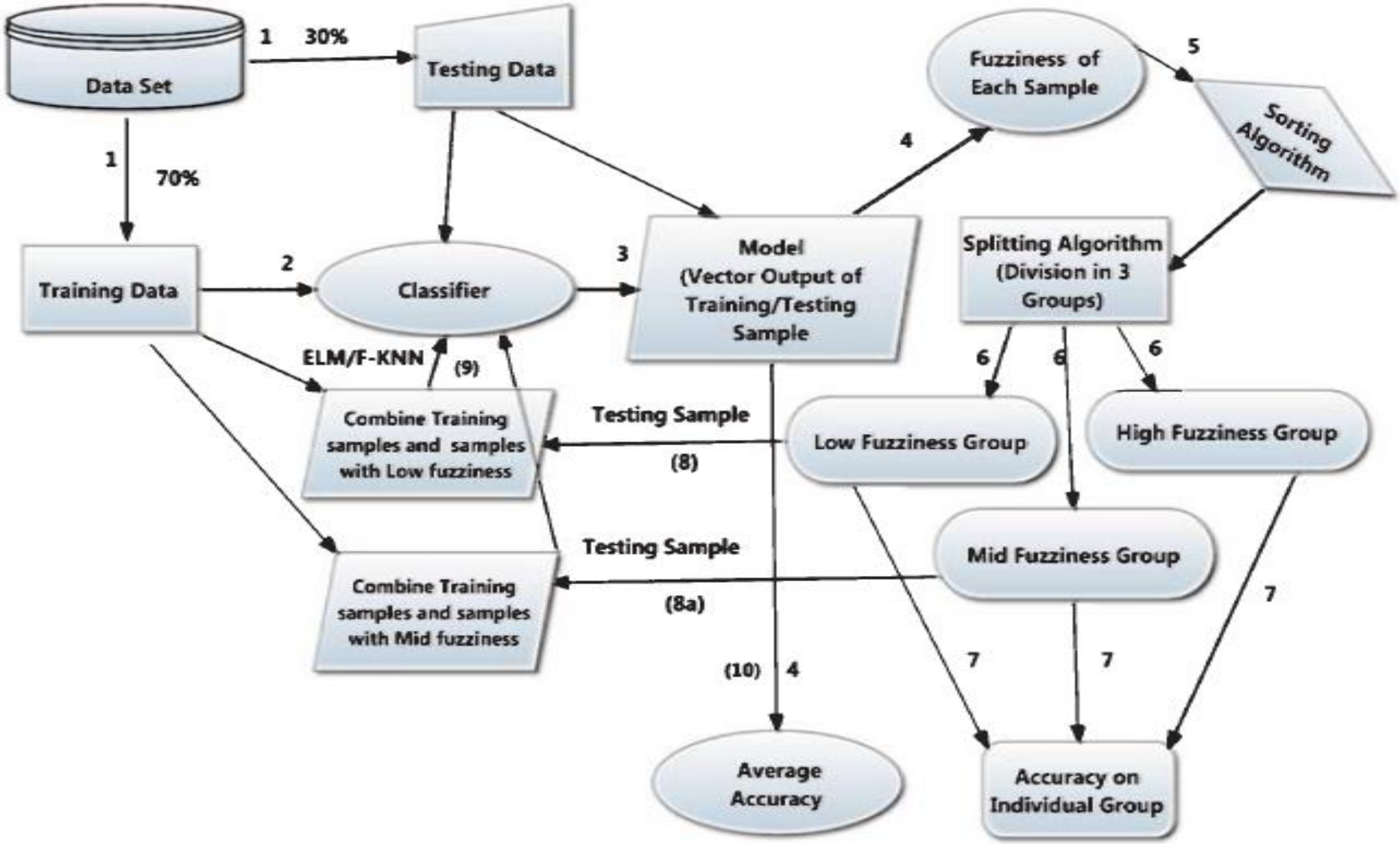
Pros

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

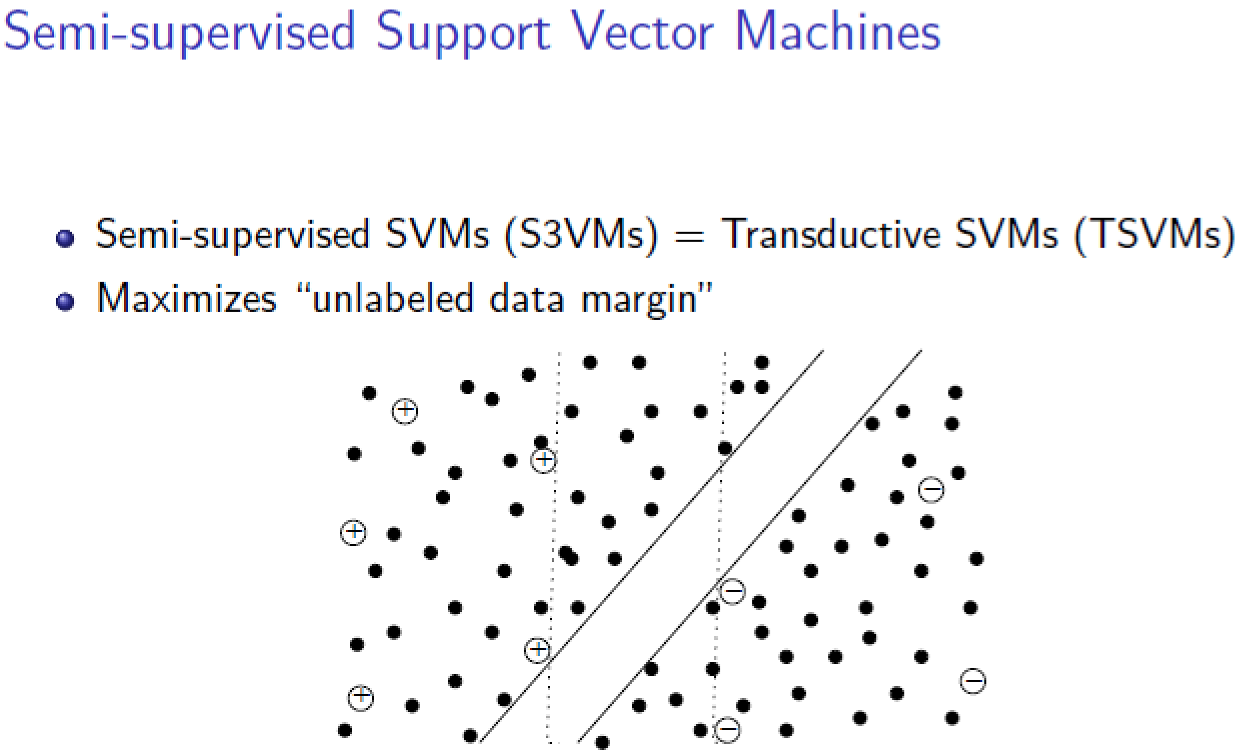
Cons

- Natural feature splits may not exist
- Models using BOTH features should do better

Fuzziness based semi-supervised learning



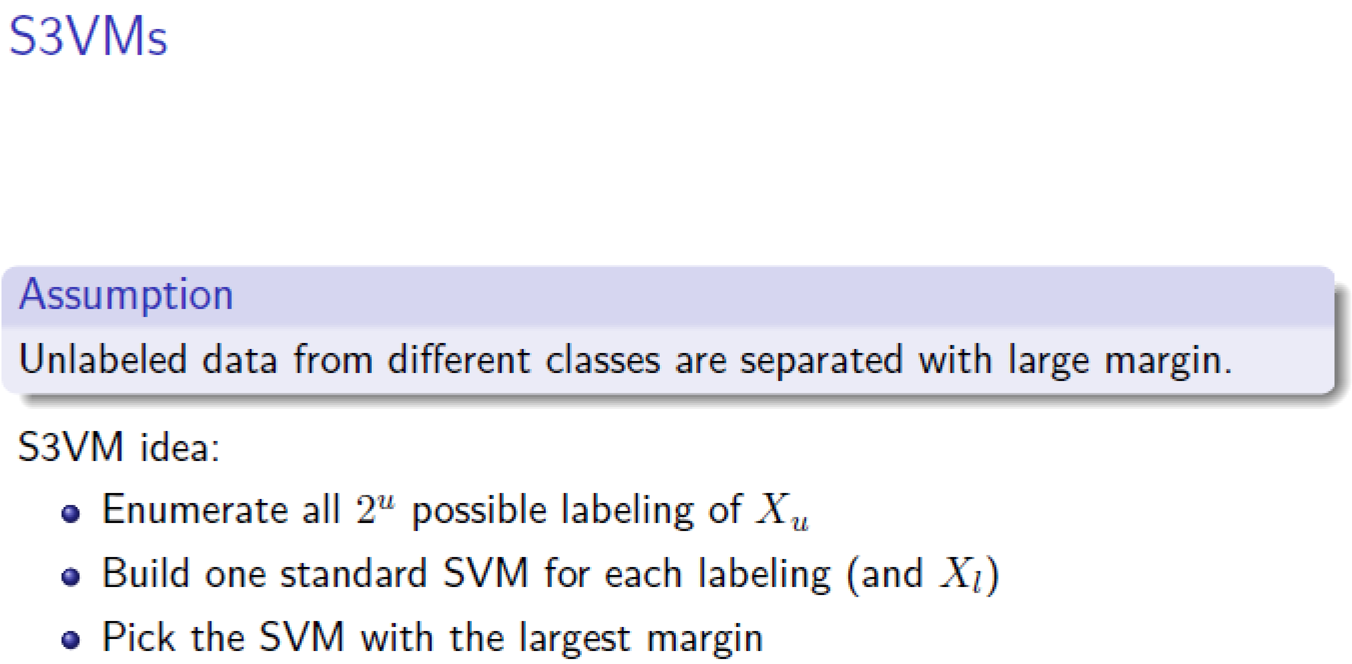
Semi-supervised Support Vector Machines



Semi-supervised Support Vector Machines

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

Semi-supervised Support Vector Machines



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

Which semi-supervised learning method should I use?

Ideally, one should use a method whose assumptions fit the problem structure.

- Do the classes produce well clustered data?

  If yes, EM with generative mixture models may be a good choice.
- Do the features naturally split into two sets?

  If yes, co-training may be appropriate.
- Is it true that two points with similar features tend to be in the same class?

  If yes, graph-based methods can be used.
- Already using SVM?

  Transductive SVM is a natural extension.
- Is the existing supervised classifier complicated and hard to modify?

  Self-training is a practical wrapper method.

Future Direction

- First: We need guarantees that semi-supervised learning will outperform supervised learning.
- Second: We need methods that benefit from unlabeled data when the size of the labeled data is large.
- Third: We need good ways to combine semi-supervised learning and active learning.
- Finally: We need methods that can efficiently process massive unlabeled data, especially in an online setting.

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