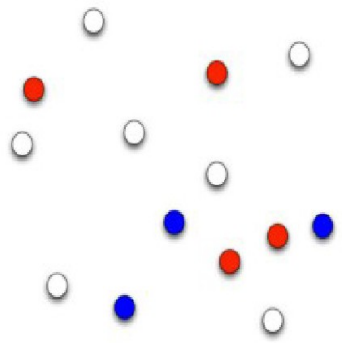
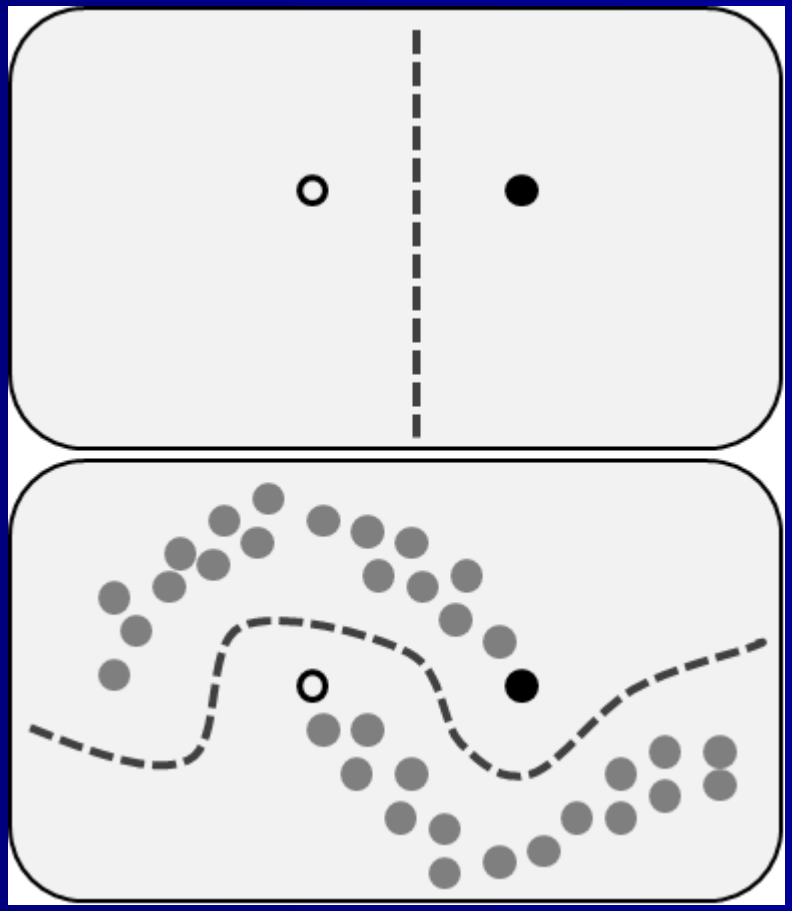
SEMI SUPERVISED LEARNING



Semi-supervised learning

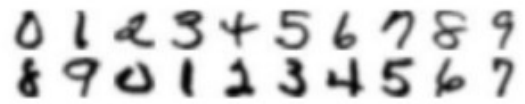
$X_1, X_2, \ldots, X_n$	C
a , b , … , b	+
b , b , … , a	-
a , a , … , b	-
b , a , … , b	+
a , b , … , a	-
b , a , … , a	-
a , a , … , b	+
a , b , … , a	?
a , b , … , b	?
b , a , … , b	?
b , b , … , a	?
a , a , … , b	?
b , a , … , a	?
a , a , … , a	?



OUTLINE

- Semi-supervised learning SSL→ type of data
- "Learning with assumptions"
- Types of semi-supervised learning algorithms
- References and software

SEMI SUPERVISED LEARNING
– DATA TYPE –



Unlabeled data, X_i

Cheap and abundant !

Human expert/
Special equipment/
Experiment

"Crystal" "Needle" "Empty"

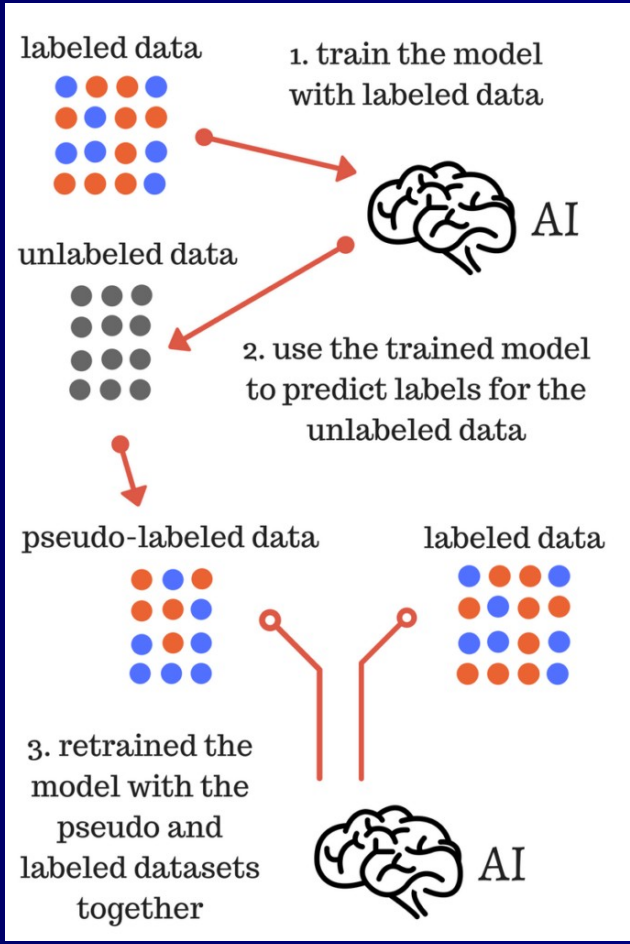
"0" "1" "2" …

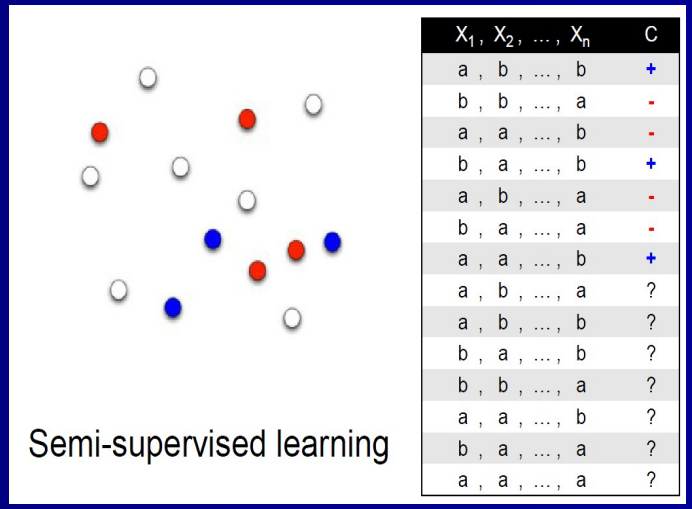
"Sports"
"News"
"Science"
…

Labeled data, Y_i

Expensive and scarce !

SEMI SUPERVISED LEARNING
– DATA TYPE –

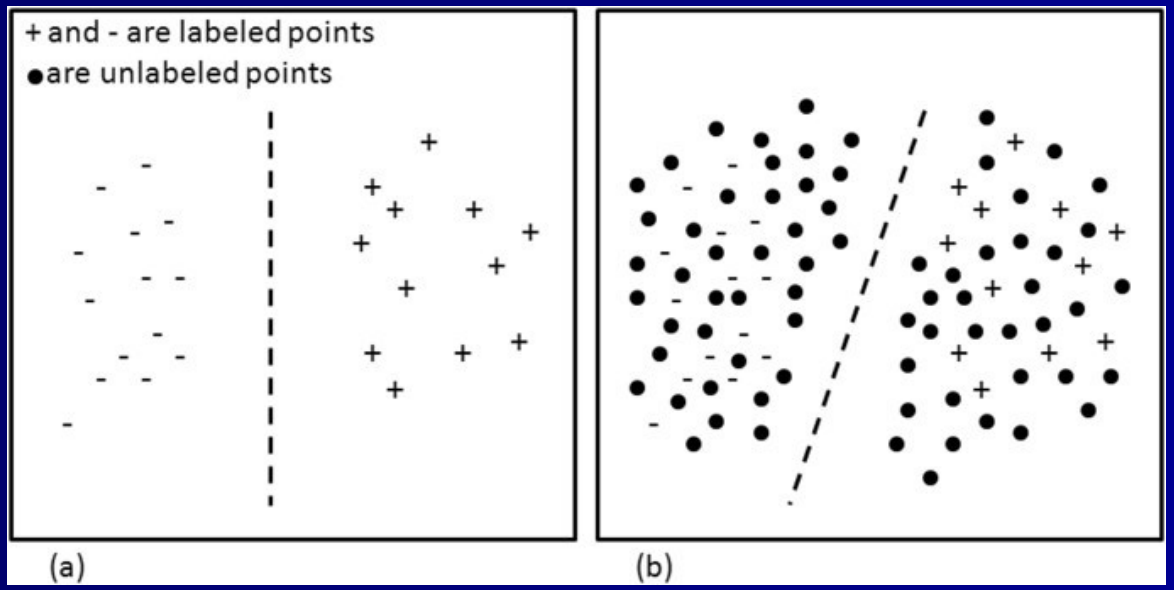




$X_1, X_2, \ldots, X_n$	C
a , b , … , b	+
b , b , … , a	-
a , a , … , b	-
b , a , … , b	+
a , b , … , a	-
b , a , … , a	-
a , a , … , b	+
a , b , … , a	?
a , b , … , b	?
b , a , … , b	?
b , b , … , a	?
a , a , … , b	?
b , a , … , a	?
a , a , … , a	?

SSL
– LEARNING WITH ASSUMPTIONS –

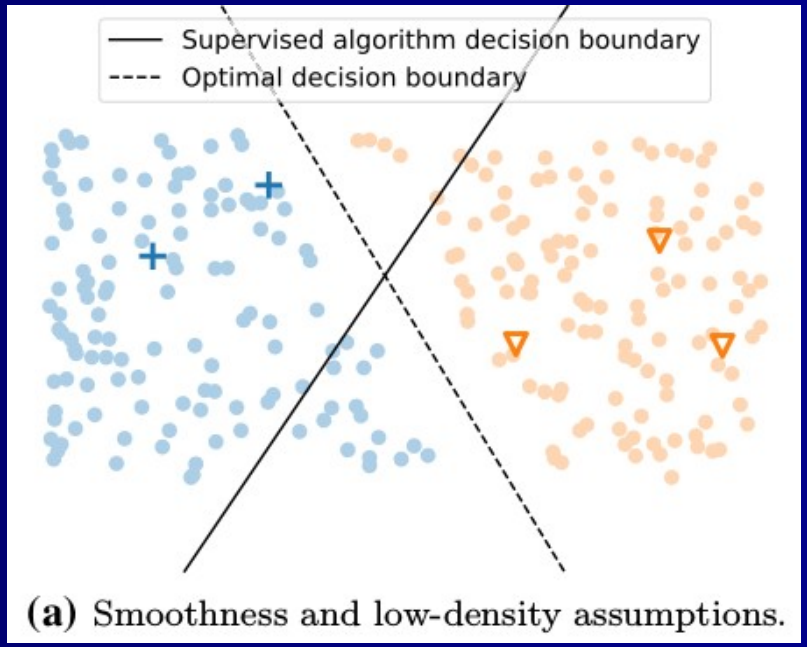
- Adding unlabeled data → no guarantee to improve SL
- Does p(x) contain info about p(y|x)?
- Essential for SSL success



SSL
– SMOOTHNESS ASSUMPTION –

- Two close points → should have same labels
- Transitively applied

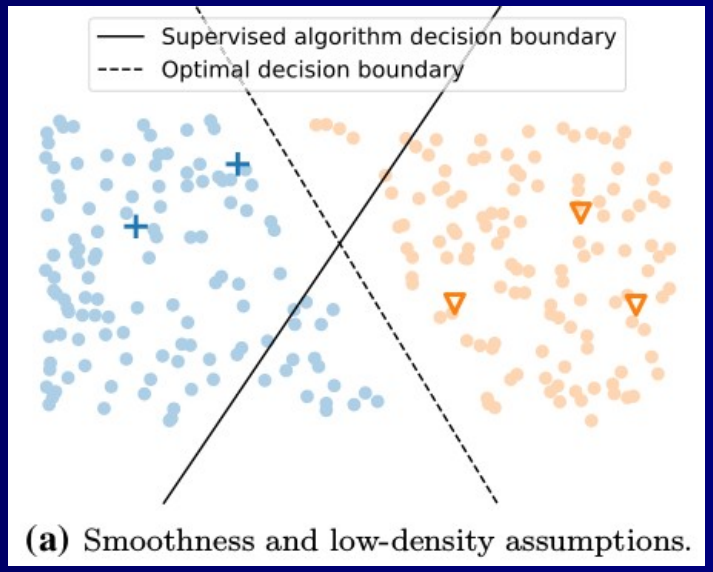
$$x_1 \sim x_2 + x_2 \sim x_3 \rightarrow x_1 \sim x_3$$



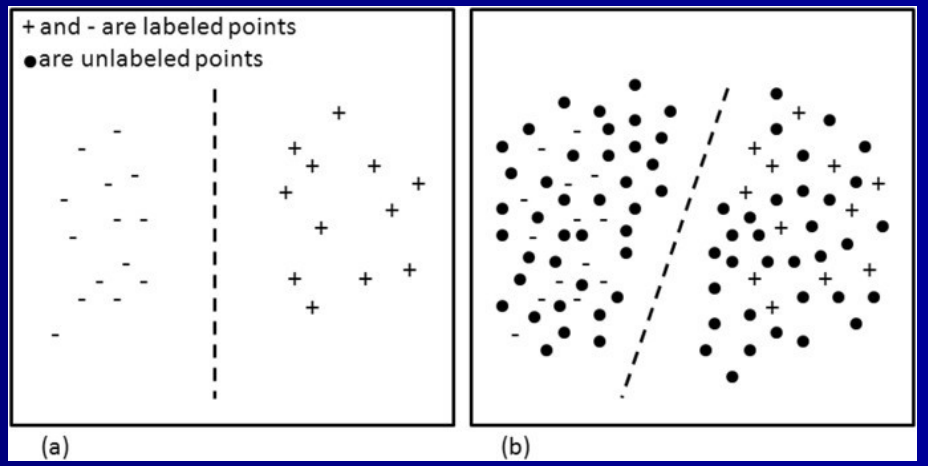
(a) Smoothness and low-density assumptions.

SSL
– LOW-DENSITY ASSUMPTION –

- Boundary → not cross high-density regions

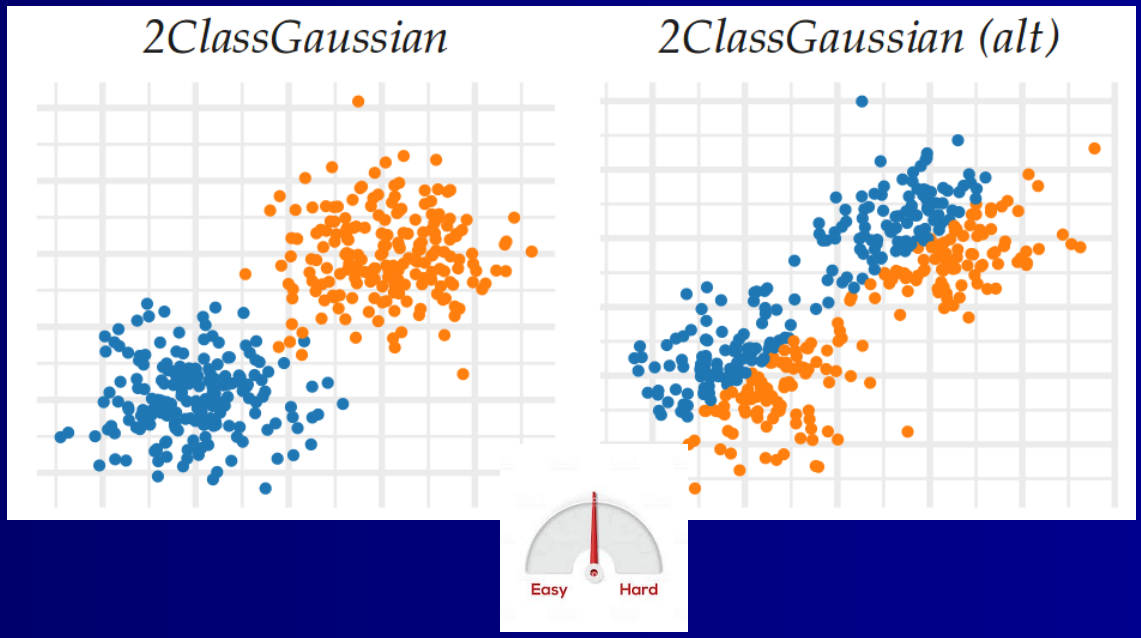


(a) Smoothness and low-density assumptions.



SSL
– LEARNING WITH ASSUMPTIONS –

- Adding unlabeled data → no guarantee to improve SL
- Keypoint → Does *p(x)* contain info about *p(class|x)*?
- Essential for SSL success



SSL
– MANIFOLD ASSUMPTION –

- Manifold definition

In machine learning problems where the data can be represented in Euclidean space, the observed data points in the high-dimensional input space $\mathbb{R}^d$ are usually concentrated along lower-dimensional substructures. These substructures are known as *manifolds*: topological spaces that are locally Euclidean. For instance, when we consider a 3-dimensional input space where all points lie on the surface of a sphere, the data can be said to lie on a 2-dimensional

- Data dispersed in high-dimensions

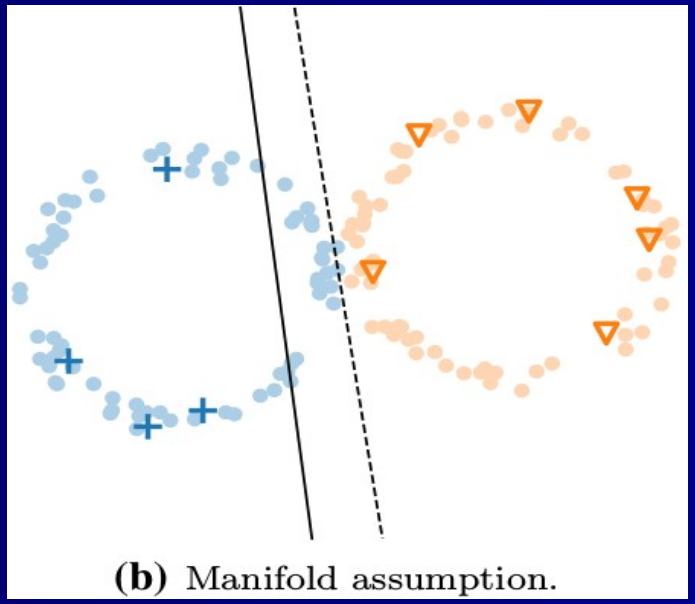
  BUT
- Concentrated in lower-dimensional structures

  → called “manifolds”

SSL
– MANIFOLD ASSUMPTION –

- Assumption → manifold structures exist in data
- Assumption → Points on the same manifold → same label

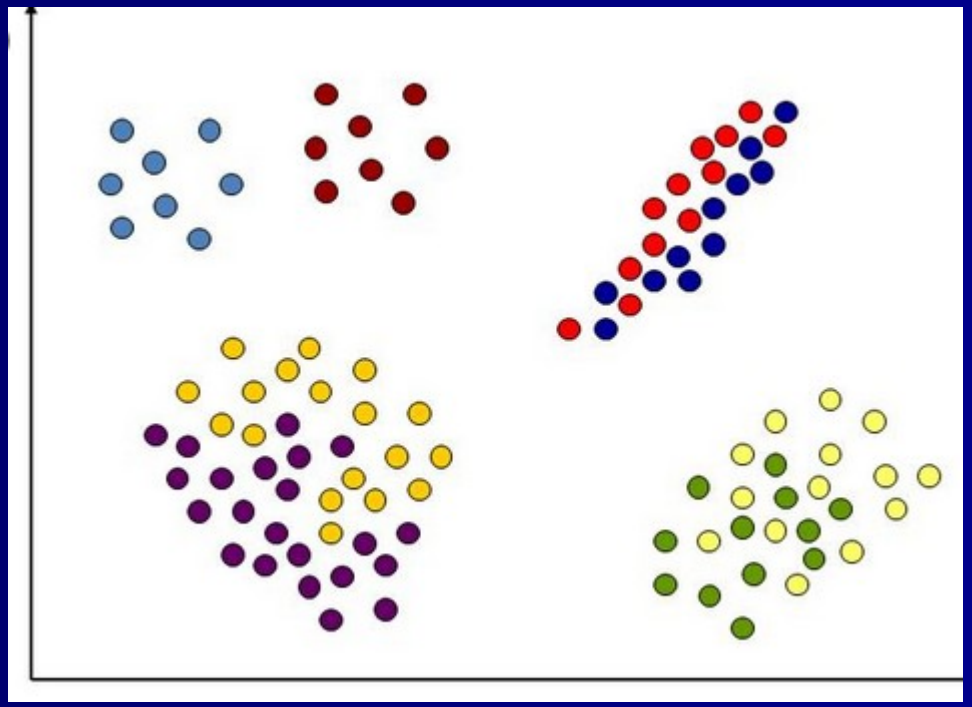
- Learn process → in low-dimension space



(b) Manifold assumption.

- KEY ASSUMPTION-
– CLUSTER ASSUMPTION –

- Does “generalize” previous assumptions? Yes…
- If points no meaningfully clustered → SSL no possible
- Does *p(x)* contain info about *p(class|x)*?

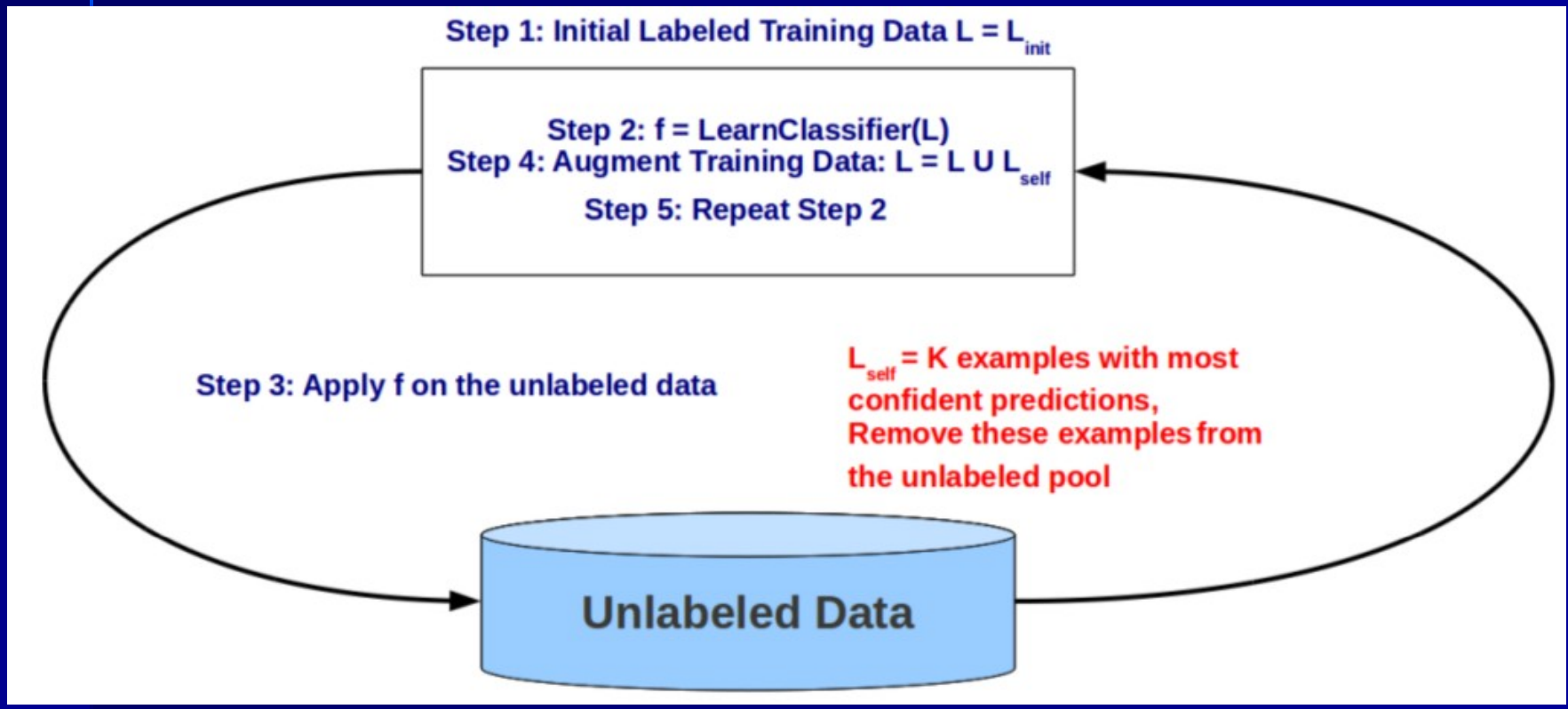


SSL
– MAIN TECHNIQUES –

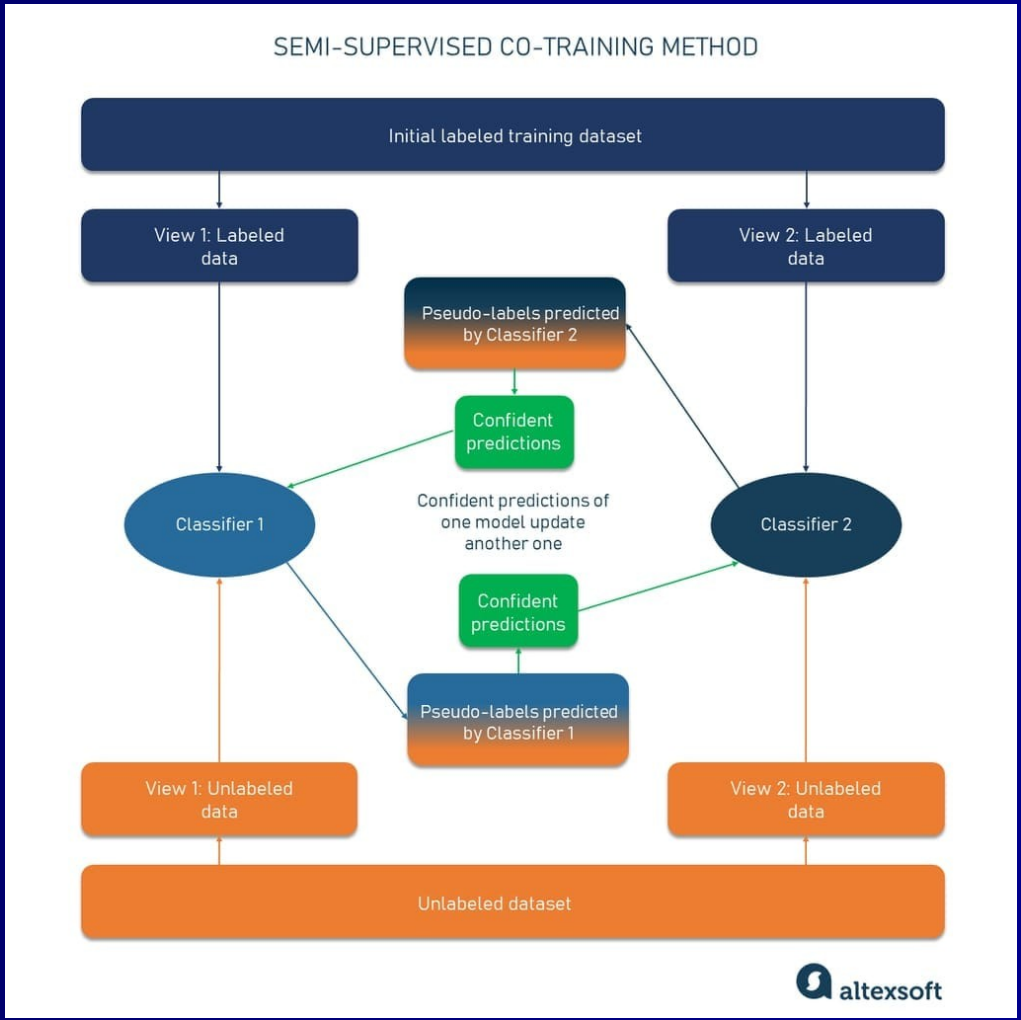
- Self-training
- Co-training
- Clustering + Labeling
- Graph-based methods
- Semi-supervised SVMs
- Gaussian model → EM for parameter learning
- Implicitly constrained by LDA

- 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

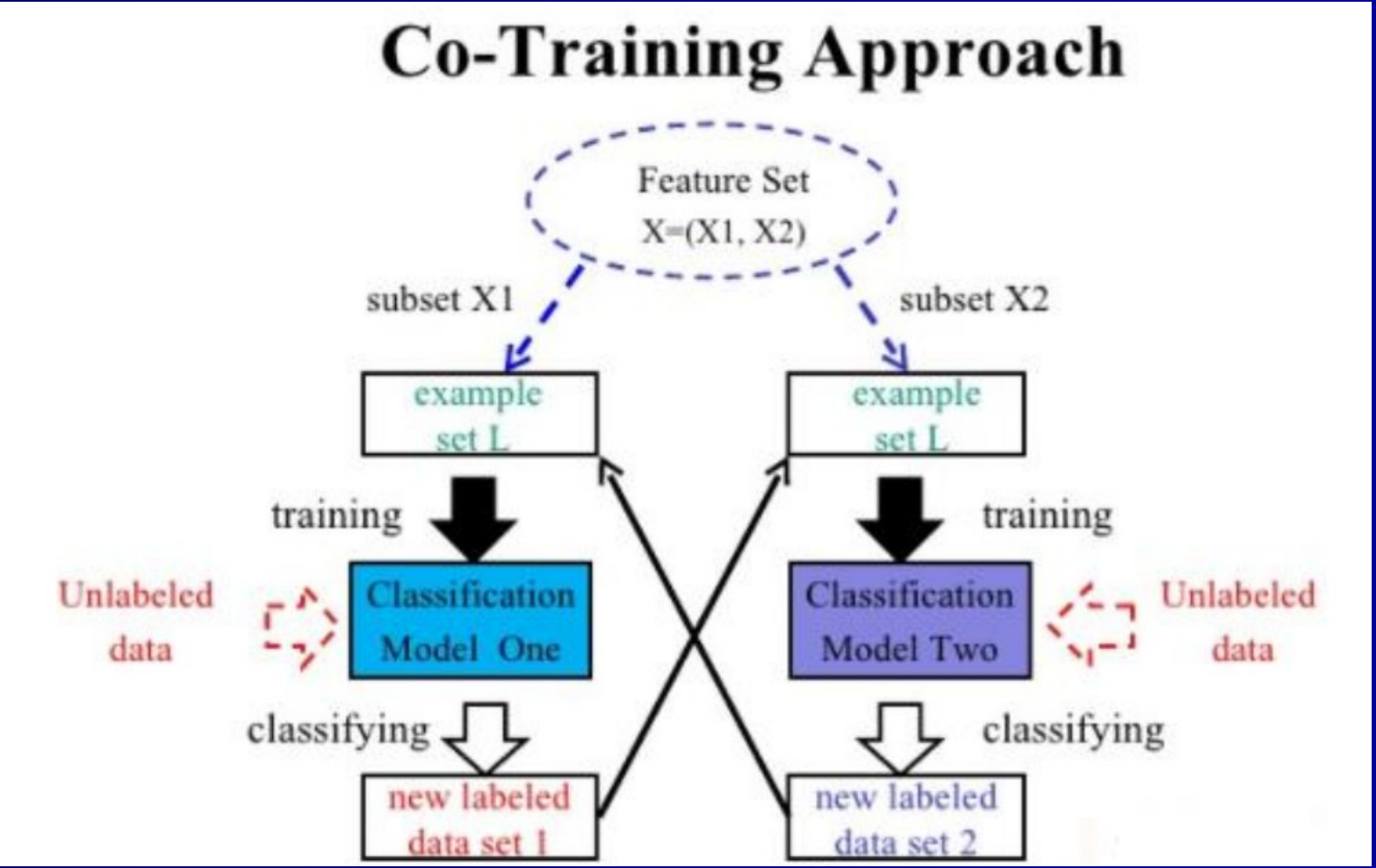
SSL
SELF -TRAINING



SSL
CO -TRAINING



SSL
CO -TRAINING



SSL
CO -TRAINING

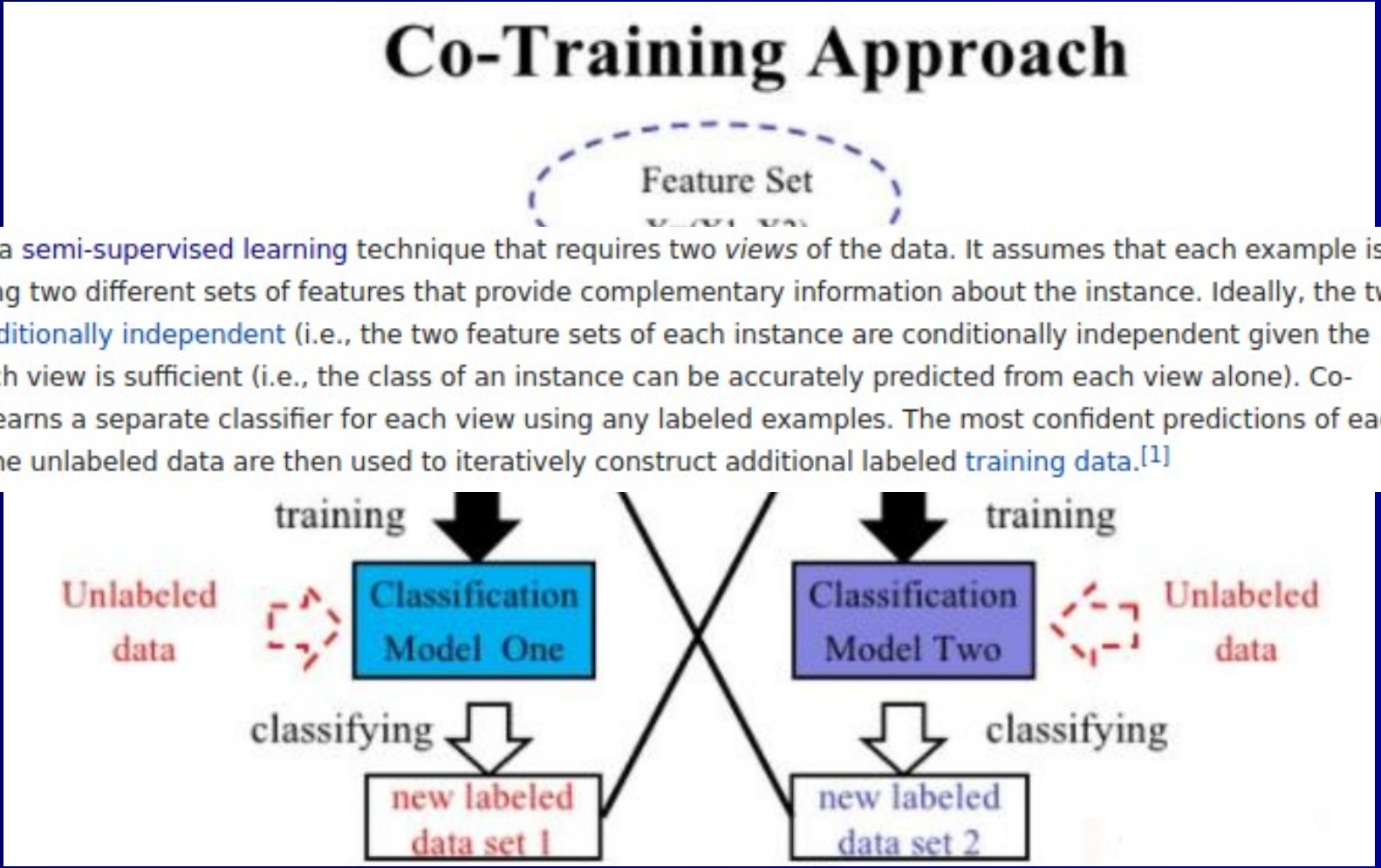
Co-Training Approach

Combining Labeled and Unlabeled Data with Co-Training*†

Avrim Blum
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
avrim+@cs.cmu.edu

Tom Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
mitchell+@cs.cmu.edu

SSL
CO -TRAINING



SSL
– CLUSTERING + LABELING –

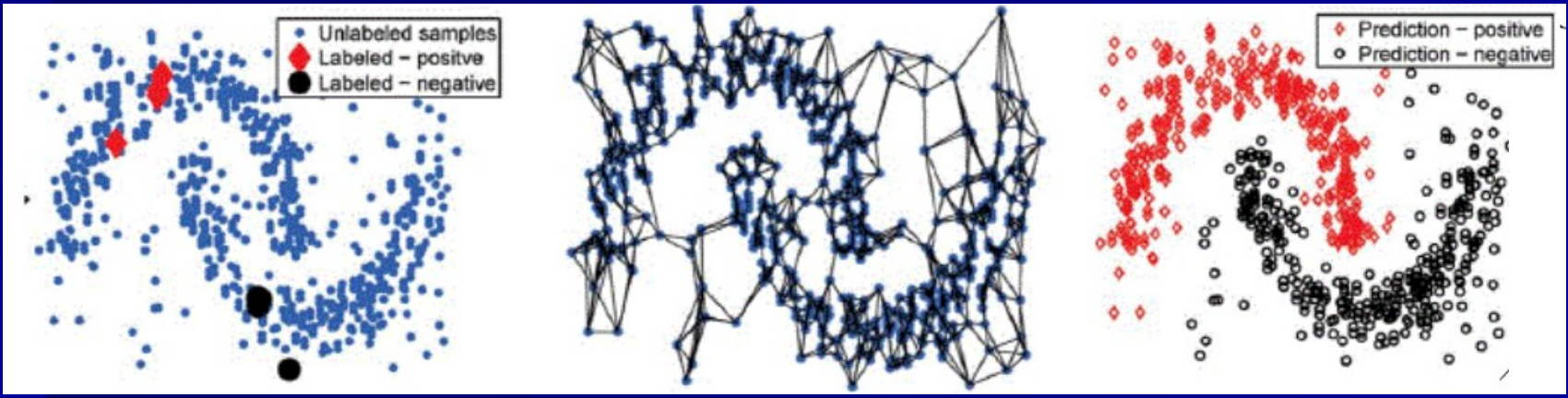
Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$,
a clustering algorithm $\mathcal{A}$, a supervised learning algorithm $\mathcal{L}$

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.

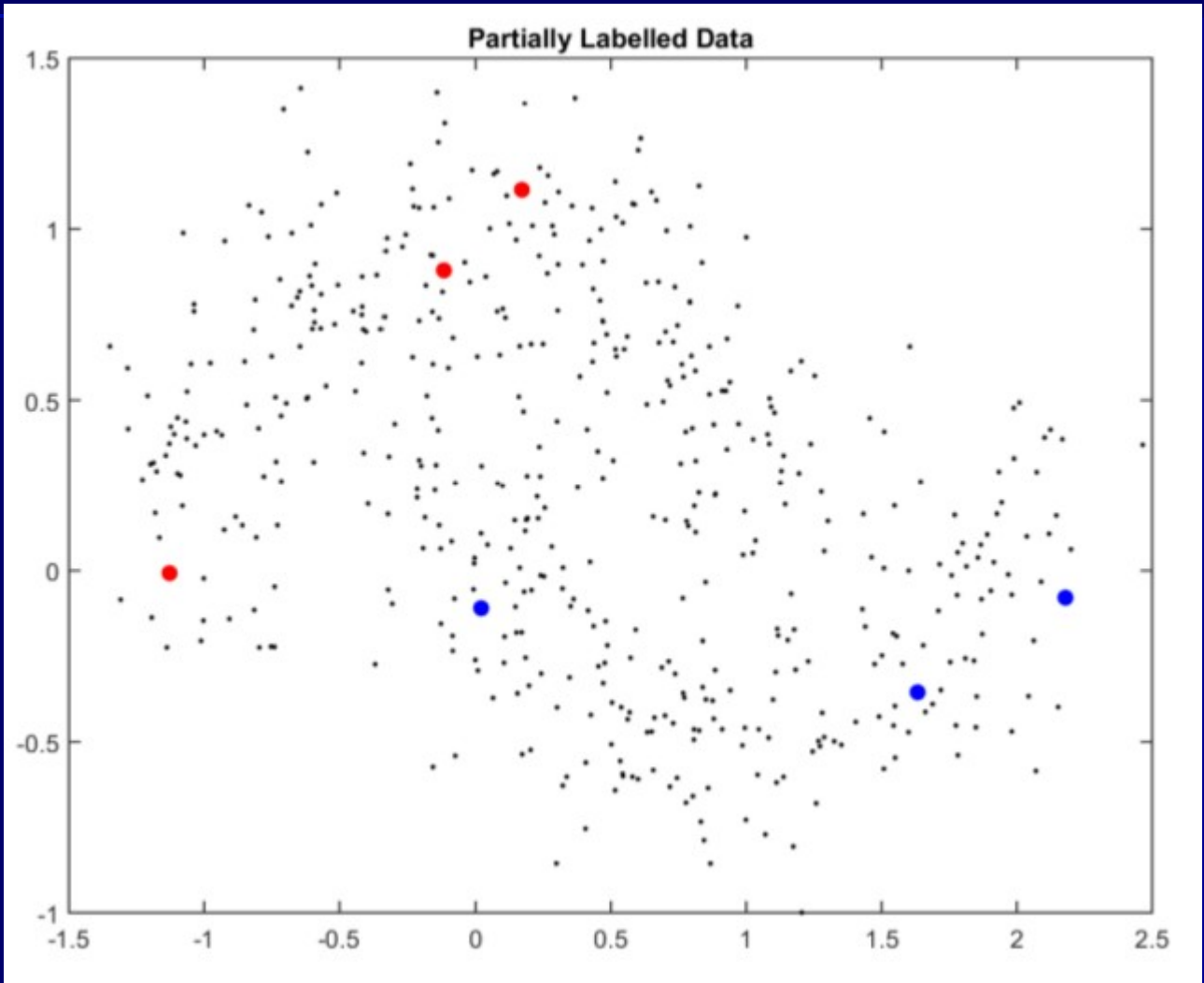
Output: labels on unlabeled data $y_{l+1}, \ldots, y_{l+u}$.

SSL
– GRAPH-BASED METHODS –

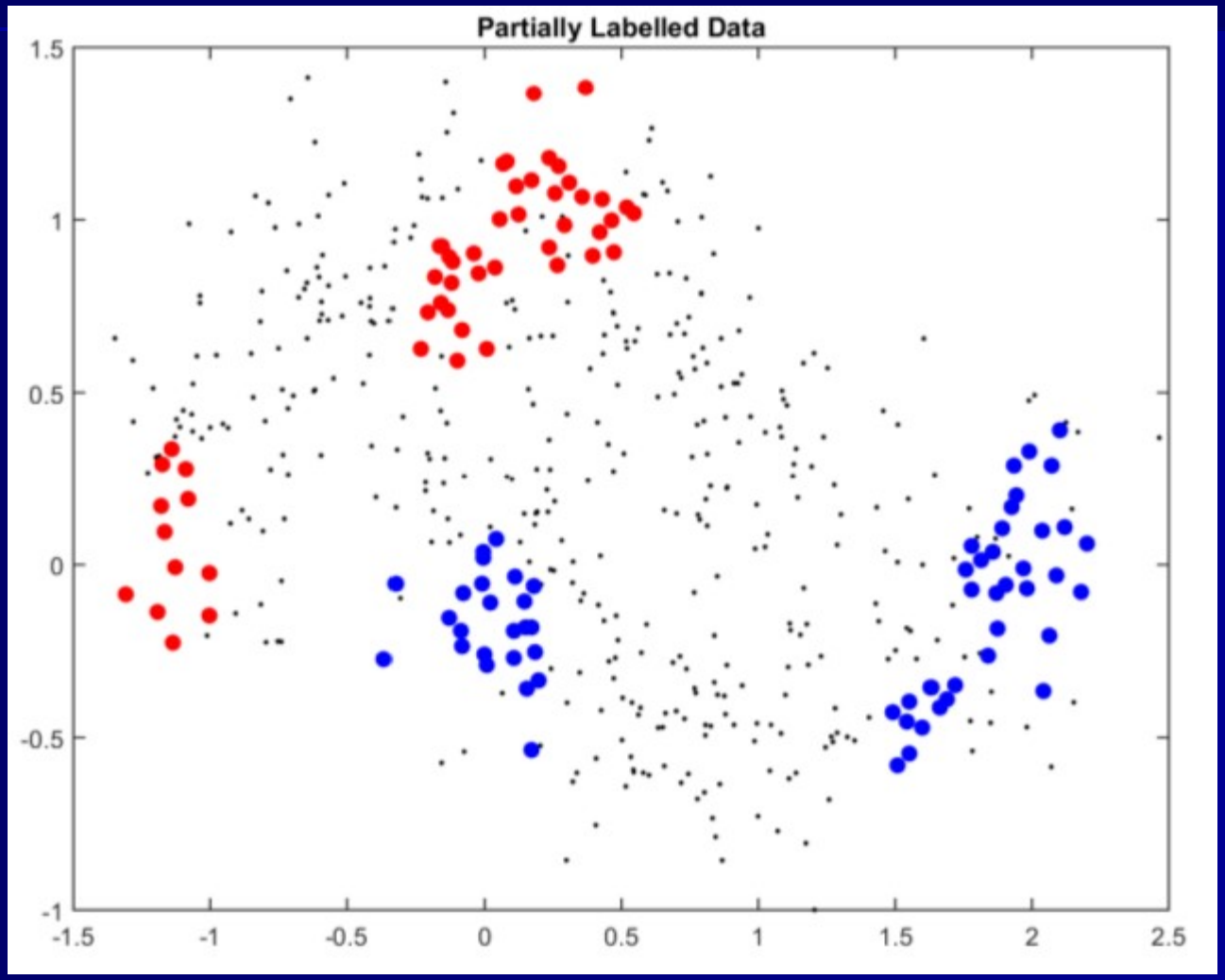
- Construct a graph with
  - nodes → instances
  - arcs → connecting "close" instances
- Propagate labels from labeled → to unlabeled
- {Smoothness + low-density region} assumptions



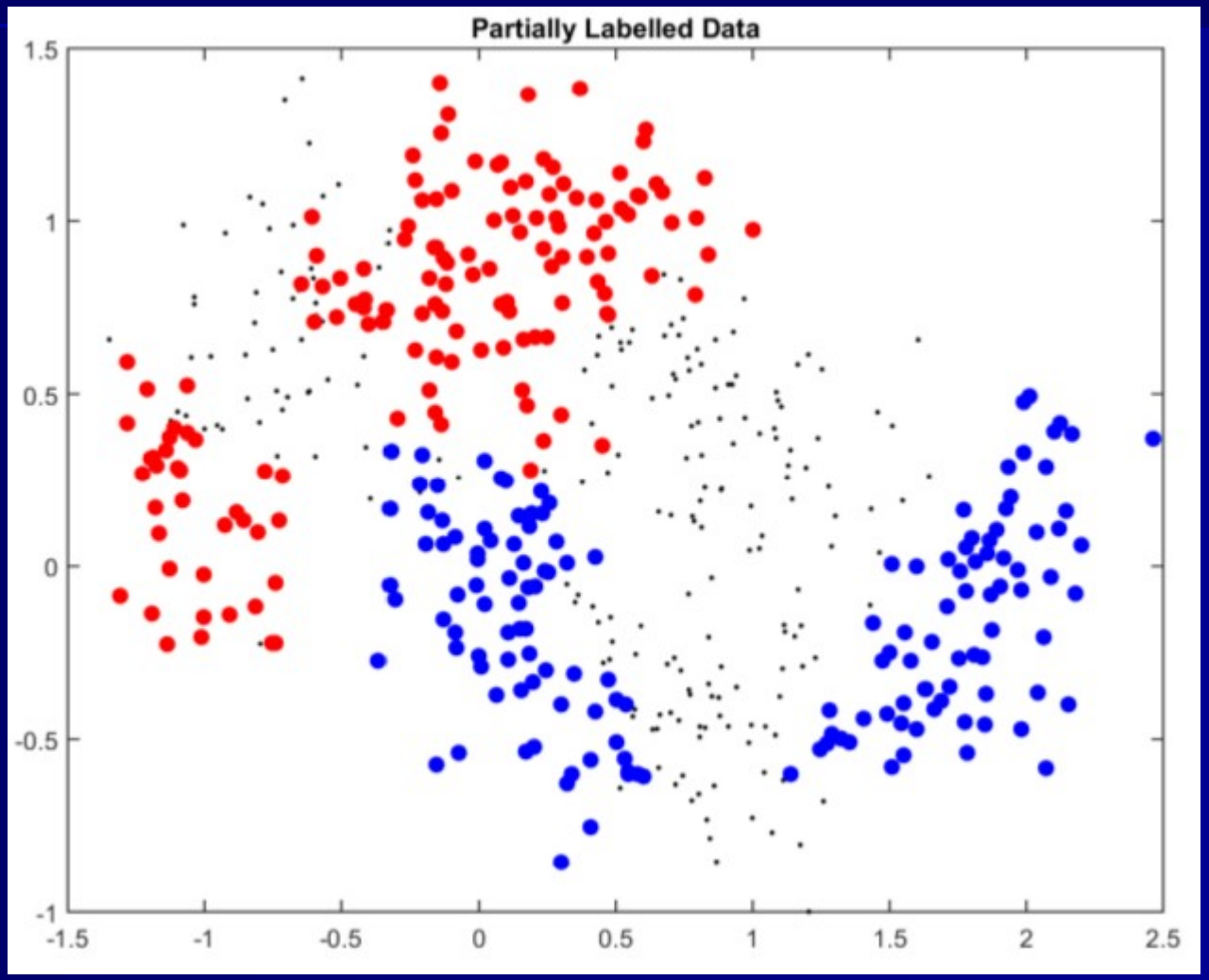
SSL
– GRAPH-BASED METHODS –



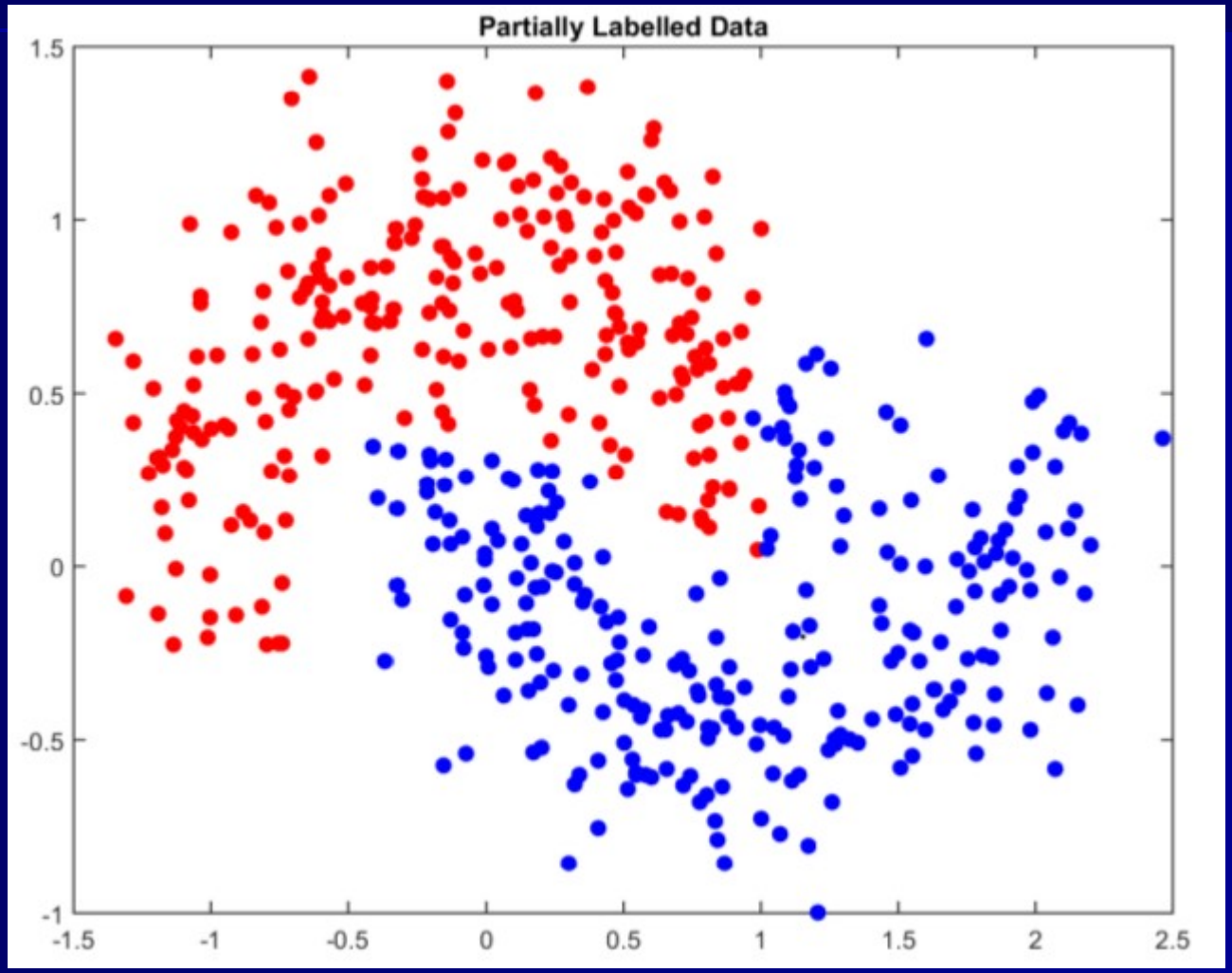
SSL
– GRAPH-BASED METHODS –



SSL
– GRAPH-BASED METHODS –

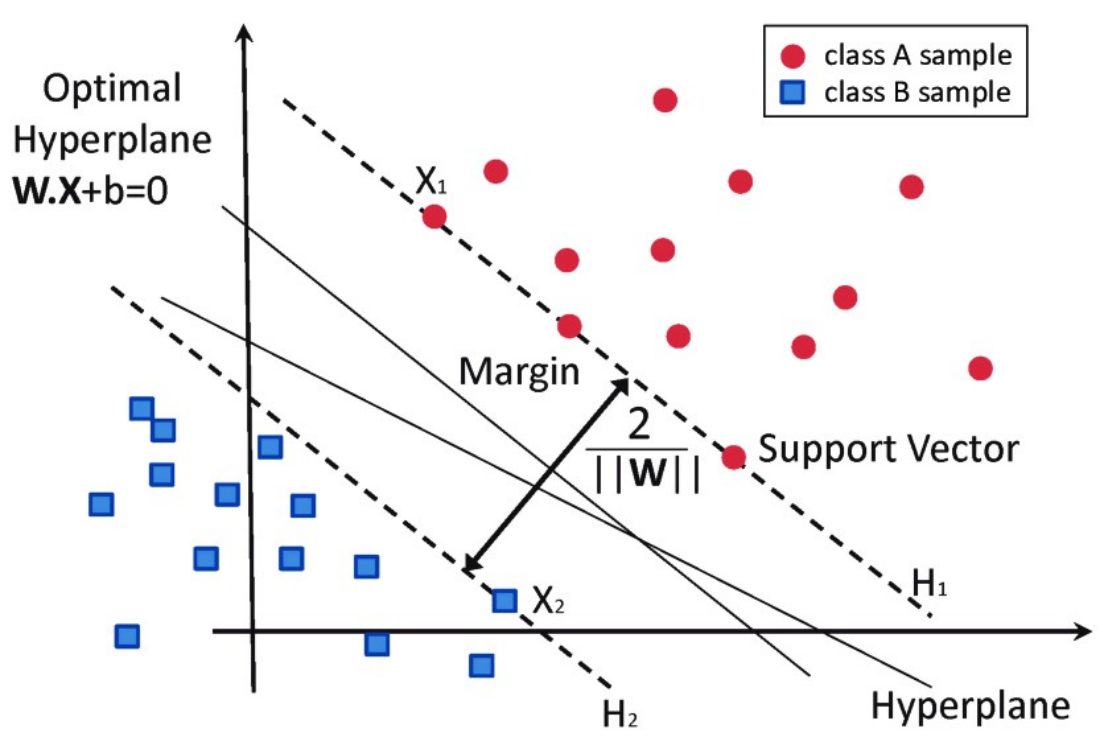


SSL
– GRAPH-BASED METHODS –



SSL
– SEMI-SUPERVISED SVMs –

- SVM formulation → from SL to SSL
- SVM for SL

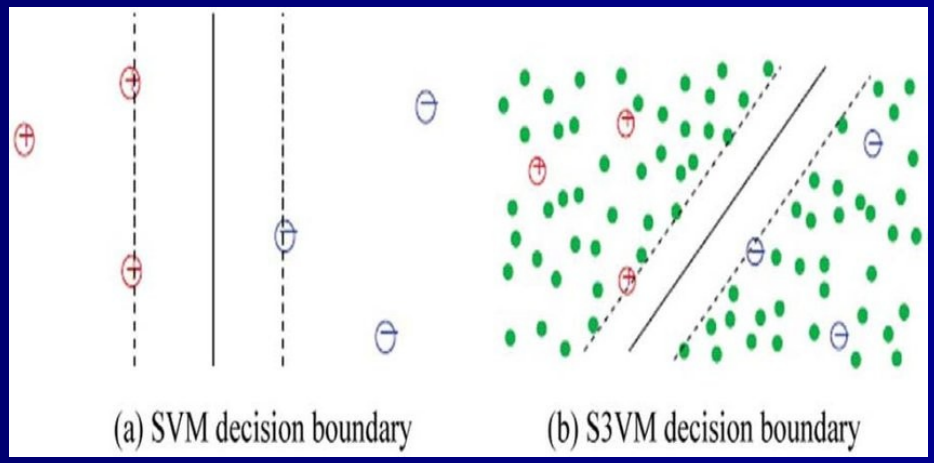


SSL
– SEMI-SUPERVISED SVMs –

- SVM formulation → from SL to SSL
- Incorporating unlabeled data to "margin maximization"

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2} \cdot ||\mathbf{w}||^2 + C \cdot \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i \cdot (\mathbf{w}^\intercal \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\ & \xi \geq 0, \quad i = 1, \ldots, l, \end{aligned}$$

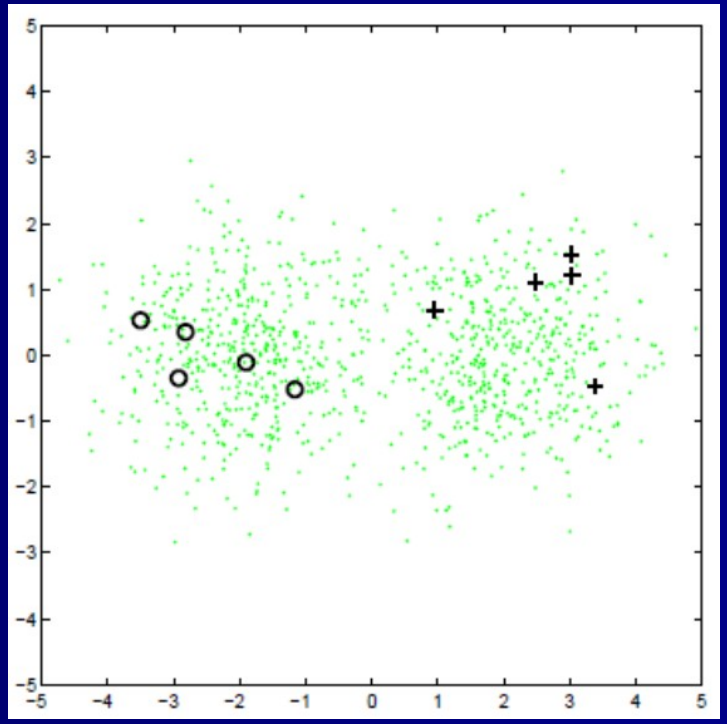
$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2} \cdot ||\mathbf{w}||^2 + C \cdot \sum_{i=1}^{l} \xi_i + C' \cdot \sum_{i=l+1}^{n} \xi_i \\ \text{subject to} \quad & y_i \cdot (\mathbf{w}^\intercal \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\ & |\mathbf{w}^\intercal \cdot \mathbf{x}_i + b| \geq 1 - \xi_i, \quad i = l+1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n, \end{aligned}$$



(a) SVM decision boundary (b) S3VM decision boundary

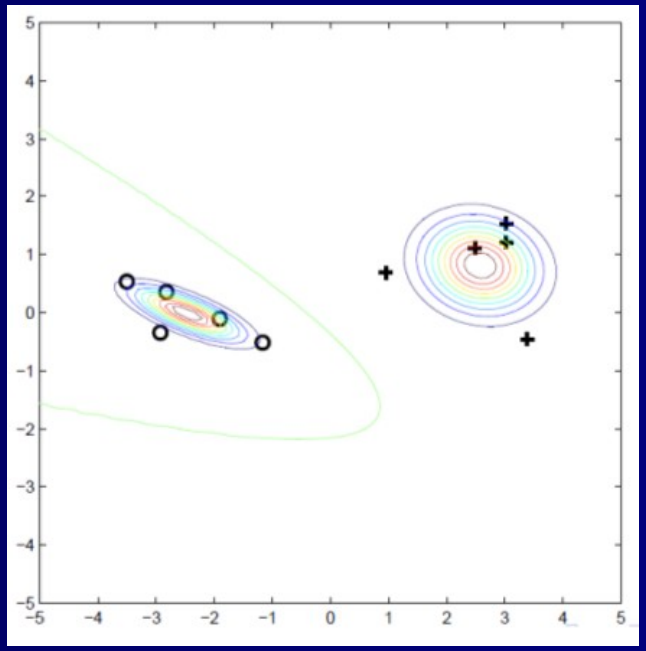
SSL
– GAUSSIAN MODEL –

- Assuming Gaussian mixture model for labeled data
- Assumption → needed to be correct!



SSL
– GAUSSIAN MODEL –

- Assuming Gaussian mixture model for labeled data
- Assumption → needed to be correct!



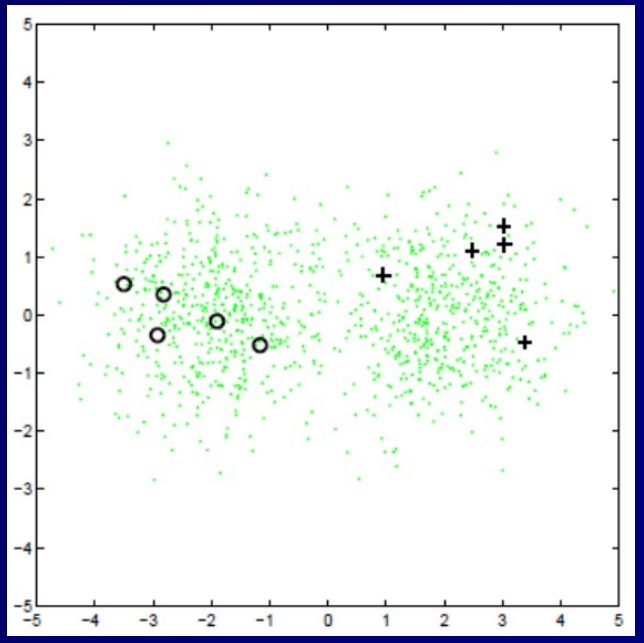
Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$
The GMM:

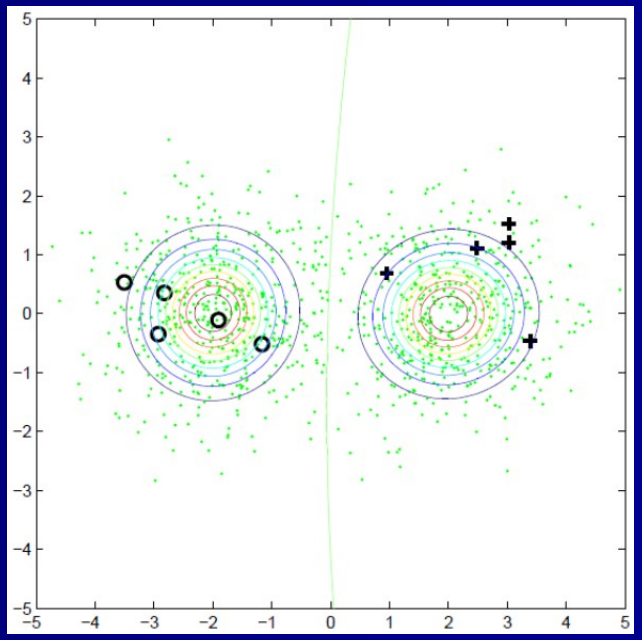
$$\begin{aligned} p(x, y|\theta) &= p(y|\theta)p(x|y, \theta) \\ &= w_y \mathcal{N}(x; \mu_y, \Sigma_y) \end{aligned}$$

Classification: $p(y|x, \theta) = \frac{p(x,y|\theta)}{\sum_{y'} p(x,y'|\theta)} \gtrless 1/2$

SSL
– GAUSSIAN MODEL –

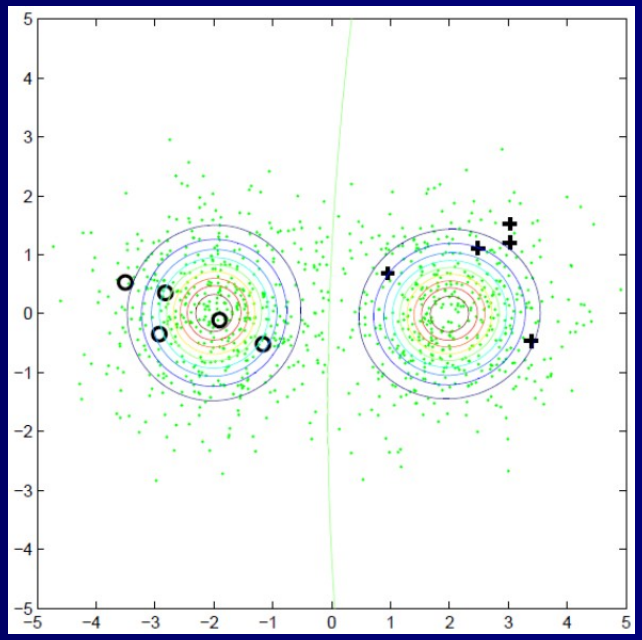
- Adding unlabeled data
- Holding Gaussian mixture assumption





SSL
– GAUSSIAN MODEL –

- How to calculate model parameters with unlabeled data?
- Expectation-Maximization algorithm - EM



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: $p(y|x, \theta) = \frac{p(x,y|\theta)}{\sum_{y'} p(x,y'|\theta)} \gtrless 1/2$

SSL
– GAUSSIAN MODEL –

- EM for parameter learning → Gaussian mixture model
- Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$

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

SSL
– GAUSSIAN MODEL –

- EM for parameter learning → Gaussian mixture model
- Model parameters: $\theta = \{w_1, w_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$

Maximum Likelihood from **Incomplete Data Via** the ***EM*** **Algorithm**

AP Dempster, NM Laird… - Journal of the Royal …, 1977 - Wiley Online Library

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched …

☆ 99 Citado por 60207 Artículos relacionados Las 67 versiones

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

SSL
– IMPLICITLY CONSTRAINED BY LDA –

Pattern Recognition 63 (2017) 115–126

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Robust semi-supervised least squares classification by implicit constraints

Jesse H. Krijthe [a,b,*], Marco Loog [a,c]

[a] Pattern Recognition Laboratory, Delft University of Technology, Mekelweg 4, 2628CD Delft, The Netherlands
[b] Department of Molecular Epidemiology, Leiden University Medical Center, Einthovenweg 20, 2333ZC Leiden, The Netherlands
[c] Image Group, University of Copenhagen, Universitetsparken 5, DK-2100 Copenhagen, Denmark

SSL
– IMPLICITLY CONSTRAINED BY LDA –

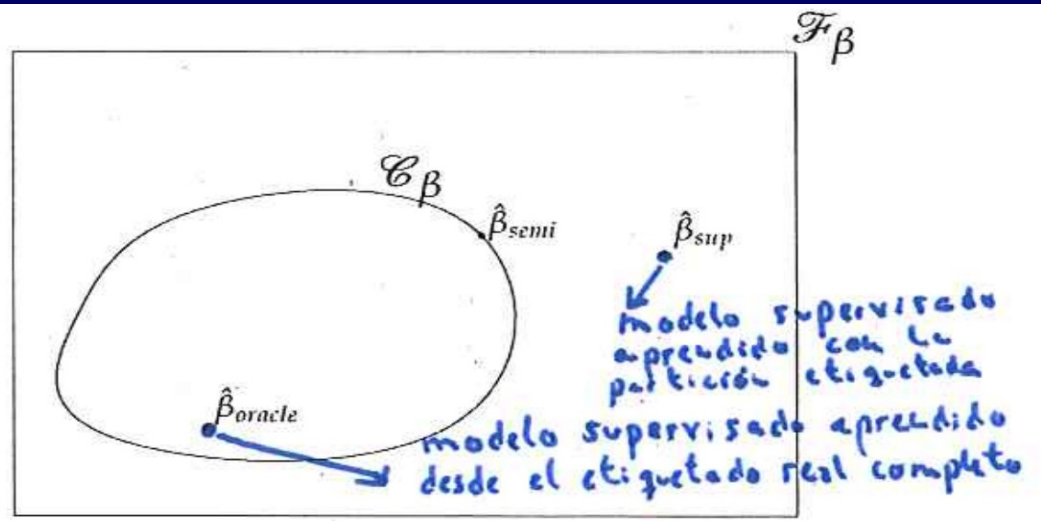


Fig. 1. A visual representation of implicitly constrained semi-supervised learning. $\mathcal{F}_\beta$ is the space of all linear models. $\hat{\beta}_{sup}$ denotes the solution given only a small amount of labeled data. $\mathcal{C}_\beta$ is the subset of the space which contains all the solutions we get when applying all possible (soft) labelings to the unlabeled data. $\hat{\beta}_{semi}$ is the solution in $\mathcal{C}_\beta$ that minimizes the loss on the labeled objects. $\hat{\beta}_{oracle}$ is the supervised solution if we would have the labels for all the objects.

- $\hat{\beta}_{semi}$ → semi supervised model:
  - trained → a (soft-)labeling of unlabeled + labeled
  - minimizes error on labeled partition

SSL

– IMPLICITLY CONSTRAINED BY LDA –

- HOW TO COMPUTE $\hat{\beta}_{semi}$?
- Least squares classification:

$$\hat{R}(\boldsymbol{\beta}) = \frac{1}{L} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \qquad \hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- IDEA → Consider all possible "labelings", $y_u \in [0,1]^U$
- IDEA → find "labeling" whose classifier minimizes empirical risk in the labeled partition

- Set of semi-supervised solutions:

$$C_{\beta} := \left\{ \boldsymbol{\beta} = \left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}_e^T\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_u\end{bmatrix} : \mathbf{y}_u \in [0,1]^U \right\} \qquad \mathbf{X}_e = \begin{bmatrix}\mathbf{X}^T & \mathbf{X}_u^T\end{bmatrix}^T$$

SSL
– IMPLICITLY CONSTRAINED BY LDA –

- IDEA → Consider all possible "labelings", $y_u \in [0,1]^U$ $\hat{R}(\boldsymbol{\beta}) = \frac{1}{L}\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2$ $\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
- IDEA → find the "labeling" whose classifier minimizes the empirical risk in the labeled partition
- Set of semi-supervised solutions: $C_{\hat{\beta}} = \left\{\boldsymbol{\beta} = \left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}_e^T\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_u\end{bmatrix} : \mathbf{y}_u \in [0,1]^U\right\}$ $\mathbf{X}_e = \begin{bmatrix}\mathbf{X}^T & \mathbf{X}_u^T\end{bmatrix}^T$

- HOW TO COMPUTE $\hat{\beta}_{semi}$?
- Closed-convex form of a quadratic problem:
  - $y_u \in [0,1]^U$ → labelings, completions
  - solving for y_u → obtain $ß_{semi}$
  - optimization with constraints
  - convex form → gradient descent

$$\underset{\mathbf{y}_u \in [0,1]^U}{\operatorname{argmin}} \frac{1}{L}\left\|\mathbf{X}\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}_e^T\begin{bmatrix}\mathbf{y}\\ \mathbf{y}_u\end{bmatrix} - \mathbf{y}\right\|_2^2$$

$$\frac{\partial \hat{R}}{\partial \mathbf{y}_u} = \frac{2}{L}\mathbf{X}_u\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}^T\mathbf{y} + \frac{2}{L}\mathbf{X}_u\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}_u^T\mathbf{y}_u + \frac{2}{L}\mathbf{X}_u\left(\mathbf{X}_e^T\mathbf{X}_e\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- "Safety SSL" theoretical result → 1D, unlimited unlabeled

SSL SOFTWARE
– 2 Gaussians artificial dataset –

RSSL: Semi-supervised Learning in R

Jesse H. Krijthe[1,2]

[1] Pattern Recognition Laboratory, Delft University of Technology
[2] Department of Molecular Epidemiology, Leiden University Medical Center
jkrijthe@gmail.com

```r
set.seed(1)
df <- generate2ClassGaussian(2000,d=2,var=0.6)

# define the supervised classifier and SSL strategy;
# and their parameters
classifiers <- list("LS"=function(X,y,X_u,y_u) {
  LeastSquaresClassifier(X,y,lambda=0)},
  "Self"=function(X,y,X_u,y_u) {
    SelfLearning(X,y,X_u,LeastSquaresClassifier)}
)

# define the type of performance metric:
# the vertical axe in the plotted curve
measures <- list("Accuracy" =  measure_accuracy)

# first line --> define unlabeled and labeled part of the data frame
# second line --> fix already defined classifier, SSL and metric
# third line --> 40% of the samples for testing and plotting the curves
# "fraction" of labeled objects varies from 0% to 100%: remaining ones, unlabeled
# repeated the process 10 times
lc1 <- LearningCurveSSL(as.matrix(df[,1:2]), df$Class,
                        classifiers=classifiers, measures=measures,
                        type="fraction", test_fraction = 0.4, repeats=10)

# analyze where the green curve (SSL) is above the red curve (standard supervised)
plot(lc1)
```

A link to this code, in eGela

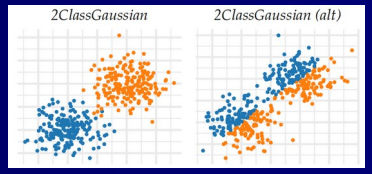


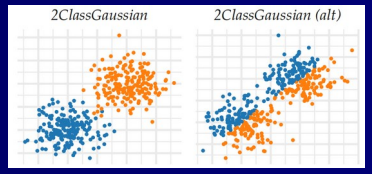
Table 1. Overview of classifiers available in RSSL

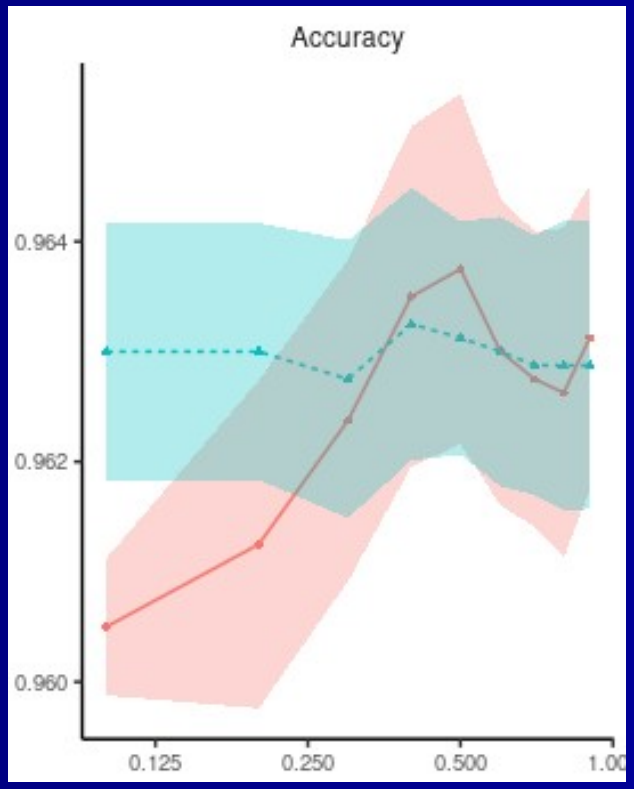
Classifier	R	Interface	Port	Reference
(Kernel) Least Squares Classifier	✓			[8]
Implicitly Constrained	✓			[13]
Implicitly Constrained Projection	✓			[12]
Laplacian Regularized	✓			[1]
Updated Second Moment	✓			[23]
Self-learning	✓			[20]
Optimistic / "Expectation Maximization"	✓			[11]
Linear Discriminant Analysis	✓			[25]
Expectation Maximization	✓			[5]
Implicitly Constrained	✓			[10]
Maximum Constrastive Pessimistic			✓	[18]
Moment Constrained	✓			[17]
Self-learning	✓			[20]
Nearest Mean Classifier	✓			[25]
Expectation Maximization	✓			[5]
Moment Constrained	✓			[16]
Self-learning	✓			[20]
Support Vector Machine	✓			
SVMlin		✓		[24]
WellSVM			✓	[14]
S4VM			✓	[15]
Transductive SVM (Convex Concave Procedure)	✓			[9,3]
Laplacian SVM	✓			[1]
Self-learning	✓			[20]
Logistic Regression	✓			
Entropy Regularized Logistic Regression	✓			[7]
Self-learning	✓			[20]
Harmonic Energy Minimization	✓			[27]

SSL SOFTWARE
– 2 Gaussians artificial dataset –

```r
set.seed(1)
df <- generate2ClassGaussian(2000,d=2,var=0.6)

# define the supervised classifier and SSL strategy;
# and their parameters
classifiers <- list("LS"=function(X,y,X_u,y_u) {
  LeastSquaresClassifier(X,y,lambda=0)},
  "Self"=function(X,y,X_u,y_u) {
    SelfLearning(X,y,X_u,LeastSquaresClassifier)}
)

# define the type of performance metric:
# the vertical axe in the plotted curve
measures <- list("Accuracy" =  measure_accuracy)

# first line --> define unlabeled and labeled part of the data frame
# second line --> fix already defined classifier, SSL and metric
# third line --> 40% of the samples for testing and plotting the curves
# "fraction" of labeled objects varies from 0% to 100%: remaining ones, unlabeled
# repeated the process 10 times
lc1 <- LearningCurveSSL(as.matrix(df[,1:2]), df$Class,
                        classifiers=classifiers, measures=measures,
                        type="fraction", test_fraction = 0.4, repeats=10)

# analyze where the green curve (SSL) is above the red curve (standard supervised)
plot(lc1)
```

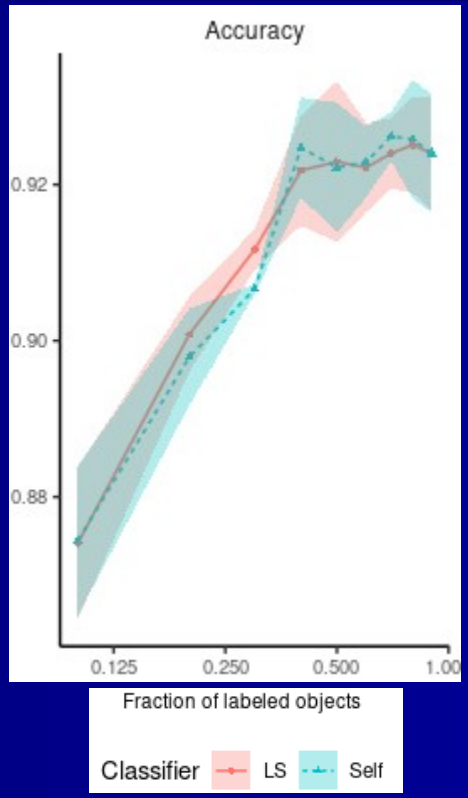
A link to this code, in eGela





SSL SOFTWARE
– Spambase dataset –

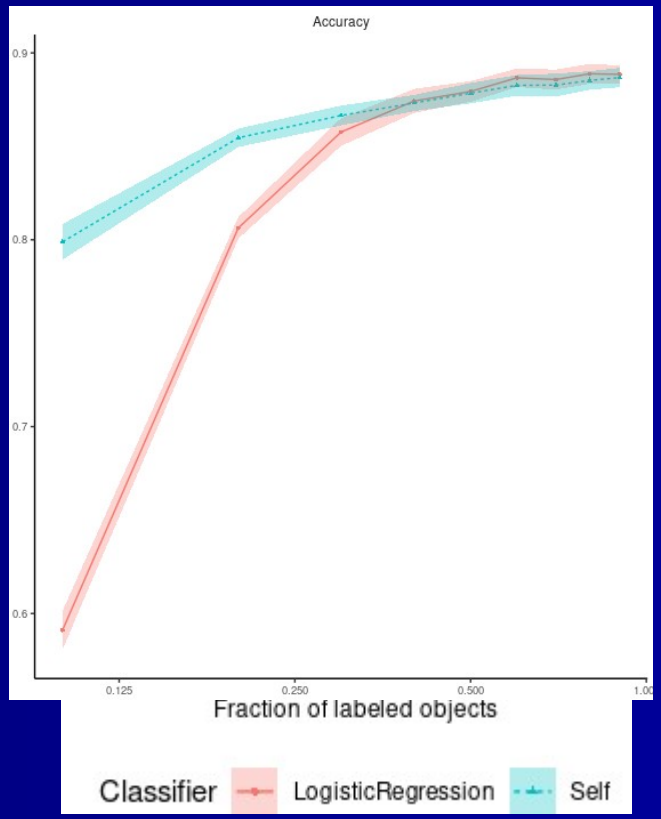
```r
set.seed(1)
# define the supervised classifier (logistic regression) and SSL strategy (self learning);
# and their parameters

classifiers3 <- list("LogisticRegression"=function(X,y,X_u,y_u) {
  LogisticRegression(X,y, lambda=0)},
  "Self"=function(X,y,X_u,y_u) {
    SelfLearning(X,y,X_u,LogisticRegression)}
)

# define the type of performance metric:
# the vertical axe in the plotted curve
measures <- list("Accuracy" =  measure_accuracy)

# first line --> define unlabeled and labeled part of the data frame
# second line --> fix already defined classifier, SSL and metric
# third line --> 40% of the samples for testing and plotting the curves
# "fraction" of labeled objects varies from 0% to 100%: remaining ones, unlabeled
# repeated the process 10 times

spambase = read.csv(file = "http://www.sc.ehu.es/ccwbayes/master/selected-dbs/nlp-naturallanguageprocessing/spambase.csv",
                    header=TRUE, sep=",")
lc2 <- LearningCurveSSL(as.matrix(spambase[,1:57]), as.factor(spambase$class),
                        classifiers=classifiers3, measures=measures,
                        type="fraction", test_fraction=0.5, repeats = 10)

plot(lc2)
```



A link to this code, in eGela

SSL SOFTWARE
– LGTB cyberbullying dataset –
- create a corpus + SSL -

```
# csv file from
# https://www.kaggle.com/datasets/kw5454331/anti-lgbt-cyberbullying-texts
# binary annotation: whether the text is considered anti-LGBT cyberbullying (1) or not (0)
library(tm)
lgtb = read.csv("anti-lgbt-cyberbullying.csv", stringsAsFactors = F, header=T)
lgtb_corpus = Corpus(VectorSource(lgtb$text))
lgtb_corpus = tm_map(lgtb_corpus, content_transformer(tolower))
lgtb_corpus = tm_map(lgtb_corpus, removeNumbers)
lgtb_corpus = tm_map(lgtb_corpus, removePunctuation)
lgtb_corpus = tm_map(lgtb_corpus, removeWords, stopwords("english"))
lgtb_corpus = tm_map(lgtb_corpus, stemDocument)
lgtb.dtm = DocumentTermMatrix(lgtb_corpus)
lgtb.dtm = removeSparseTerms(lgtb.dtm, 0.99)
dim(lgtb.dtm)

labels = lgtb$anti_lgbt
lgtb.dtm = cbind(lgtb.dtm,labels)
lgtb.dtm.matrix = as.matrix(lgtb.dtm)
colnames(lgtb.dtm.matrix)[277] = "label"

library(RSSL)
classifiers <- list("LeastSquaresClassifier"=function(X, y, X_u, y_u) {
LeastSquaresClassifier(X,y)},"Self"=function(X,y,X_u,y_u) {SelfLearning(X,y,X_u, method=LeastSquaresClassifier)})

measures <- list("Accuracy"=measure_accuracy)

lc2 <- LearningCurveSSL(lgtb.dtm.matrix[,1:276], as.factor(lgtb.dtm.matrix[,277]),
                        classifiers=classifiers, measures=measures,
                        type="fraction", test_fraction=0.3, repeats = 5)
plot(lc2)
```



A link to this code, in eGela

SSL
– PROPOSED EXERCISE –

- RSSL package
- Consult its R-vignette [github] [arXiv]

- Choose an artificial dataset offered by the package
- Check the variety of artificial datasets and their shapes
- Functions to generate artificial datasets
- Choose and load a real dataset → csv format

- LearningCurveSSL( ) function
- Understand associated parameters
- Change parameter values and check the result
- Change base classifiers
- Choose different SSL strategies
- Type of measures-metrics offered by RSSL
- Does the SSL strategy improve SL?