APRENDIZAJE SEMI-SUPERVISADO

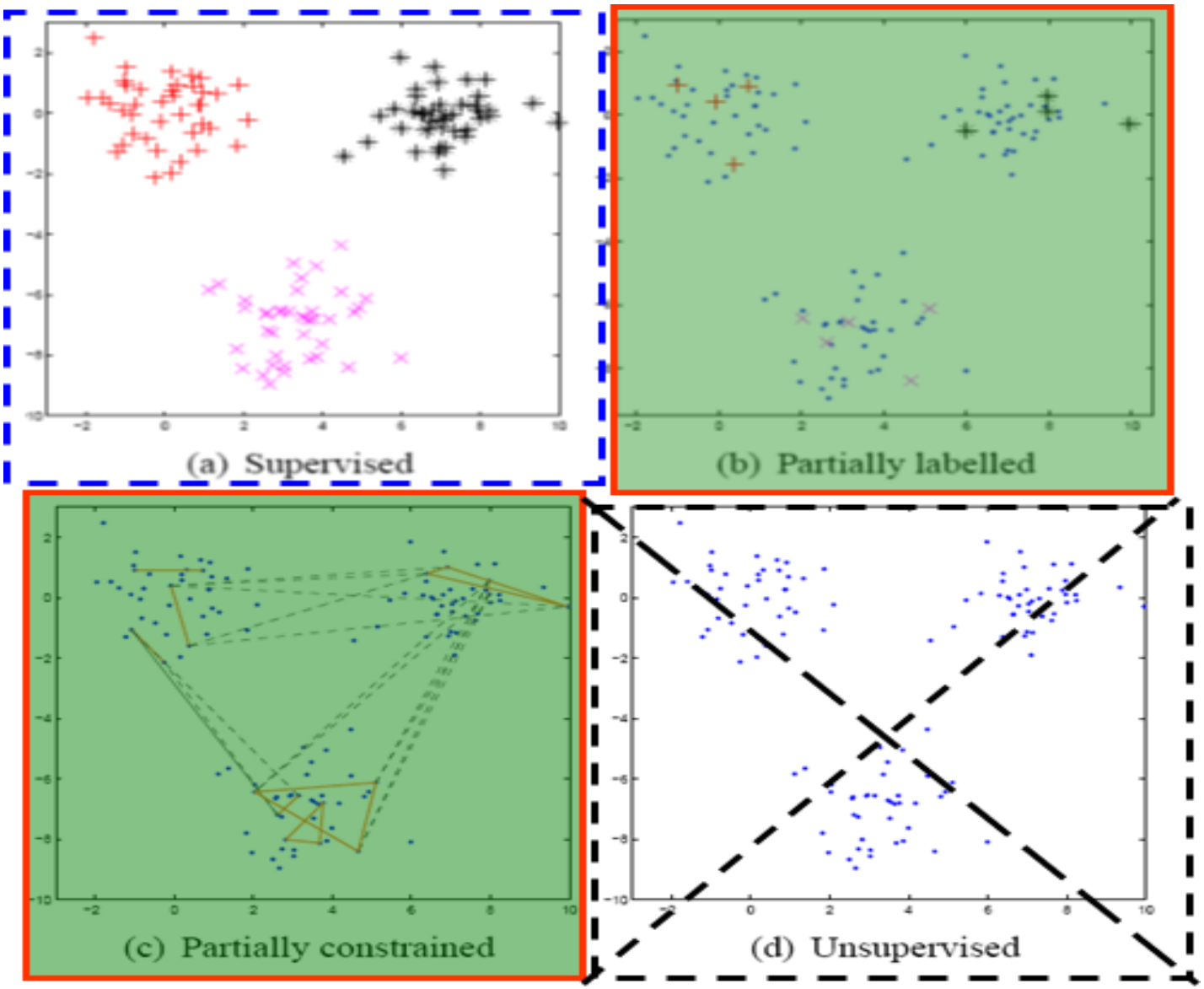
Minería de Datos: Aspectos Avanzados

Salvador García
salvagl@decsai.ugr.es

Objetivos

- Entender el problema del aprendizaje semi-supervisado situarlo en el contexto de la predicción.
- Conocer las diferentes estrategias que se emplean en el campo del aprendizaje semi-supervisado.
- Estudiar diferentes medidas y técnicas de evaluación en aprendizaje semi-supervisado.
- Presentar algunas propuestas clásicas en el enfoque de self-labelling, grafos y label propagation.

Spectrum of Learning Problems



Semi-Supervised Learning: SSL

In many problems, labeled data can be rare or expensive.

Need to pay someone to do it, requires special testing,...

Unlabeled data is much cheaper.

Speech

Customer modeling

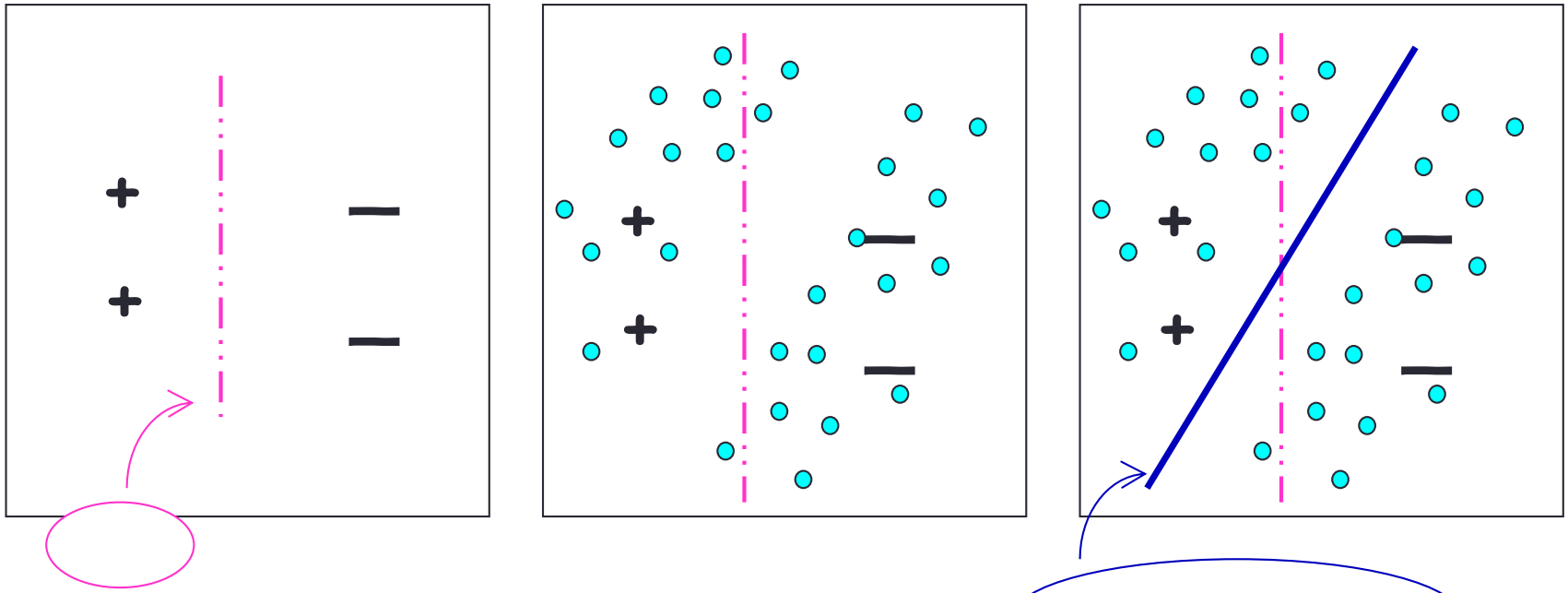
Images

Protein sequences

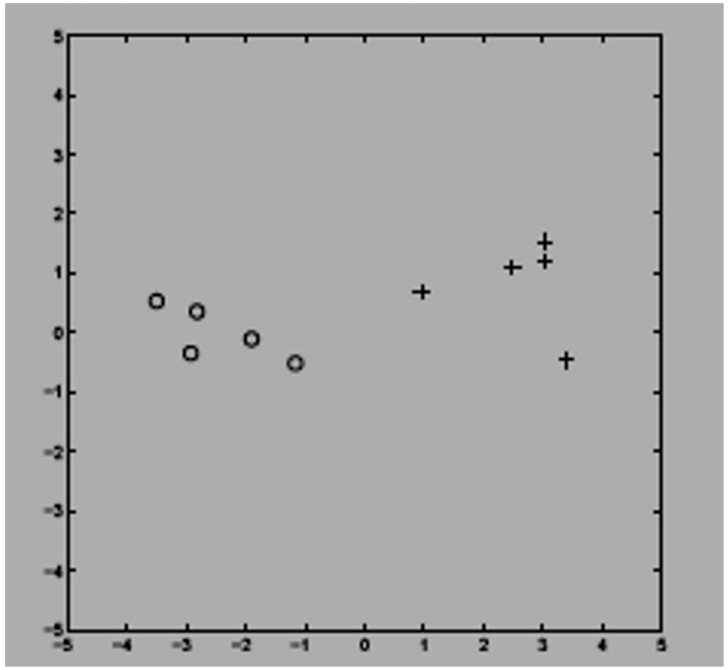
Medical outcomes

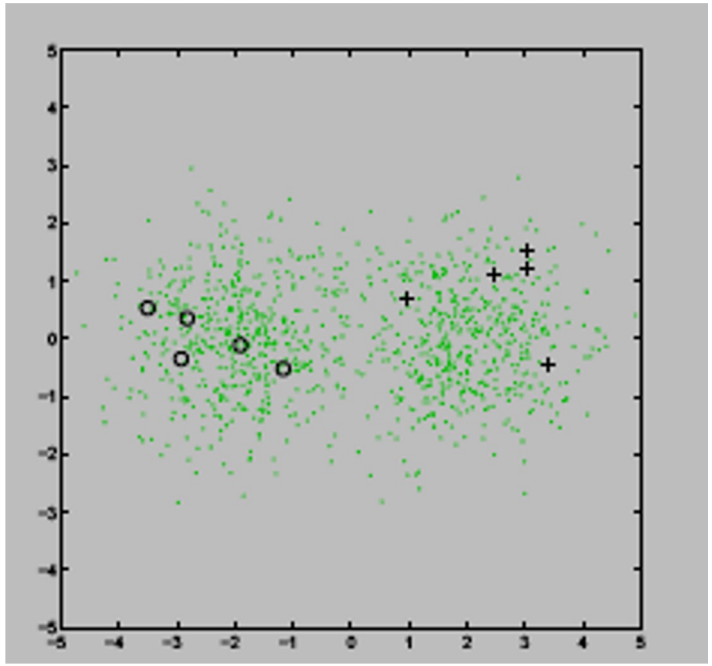
Web pages

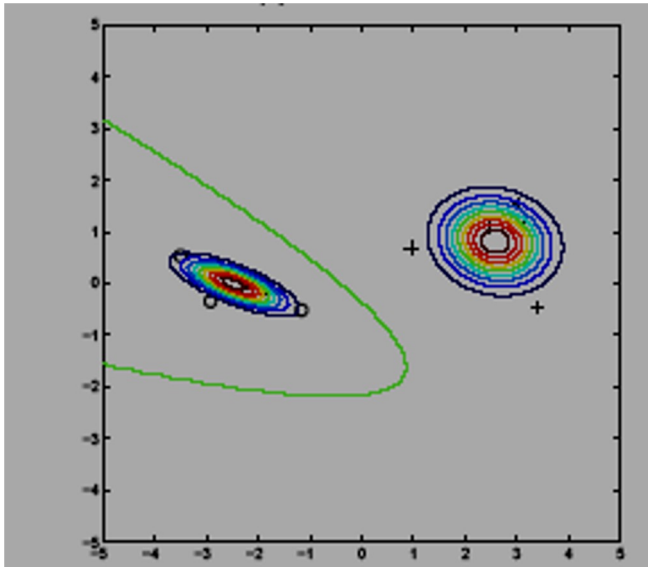
Intuition

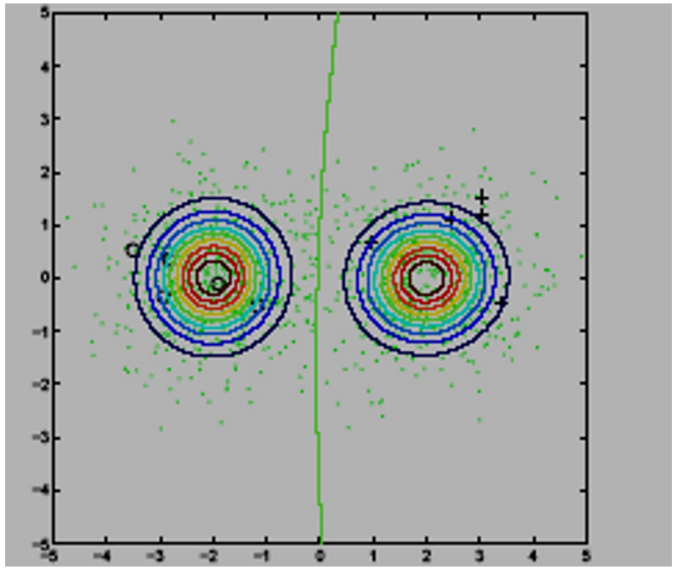


Intuition



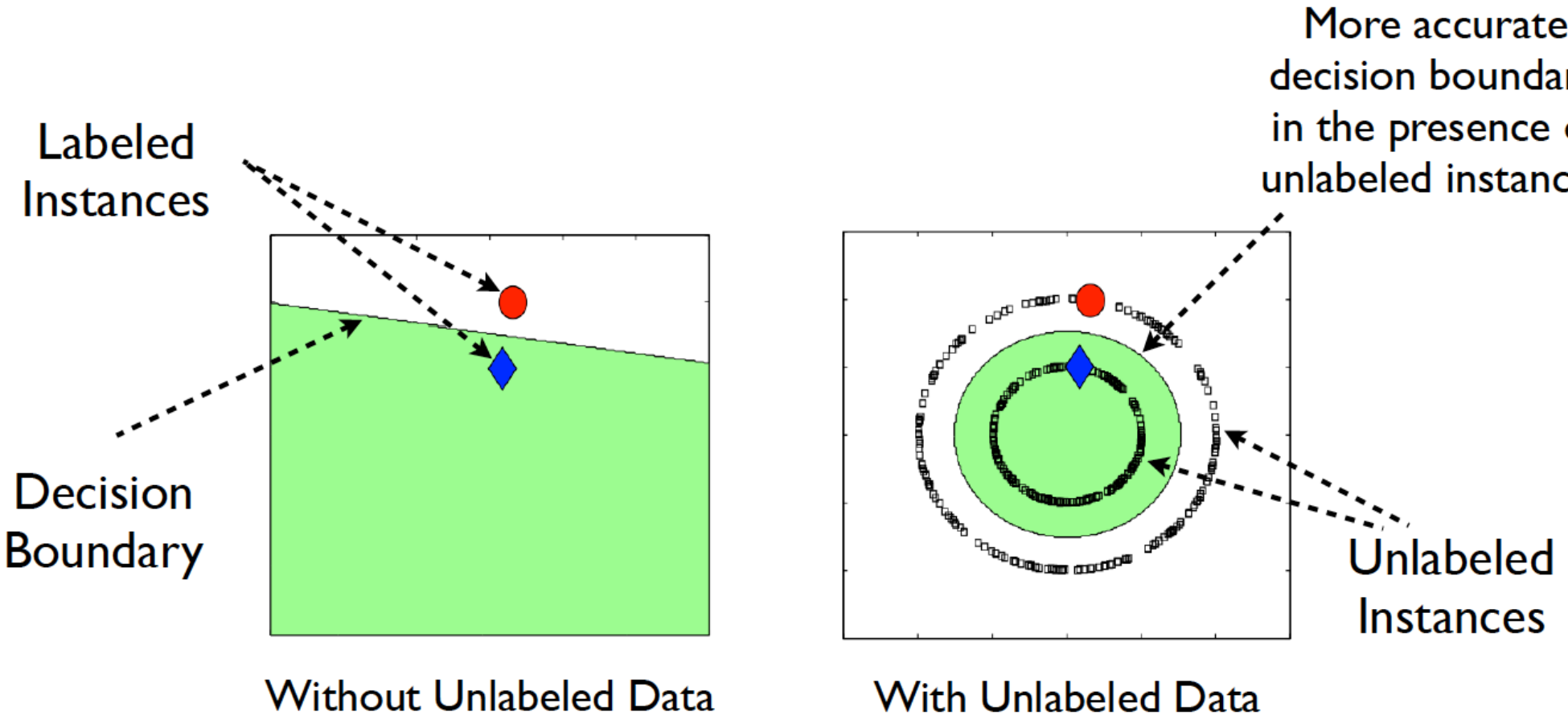






Intuition

How can unlabeled data be helpful?



Example from [Belkin et al., JMLR 2006]

Semi-Supervised Learning: SSL

Can we use unlabeled data to augment a small labeled sample to improve learning?

But unlabeled data is missing the most important info!!

But maybe still has useful regularities that we can use.

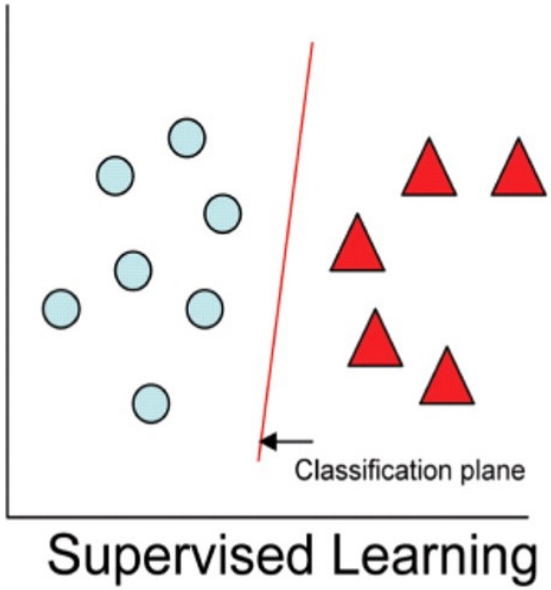
But

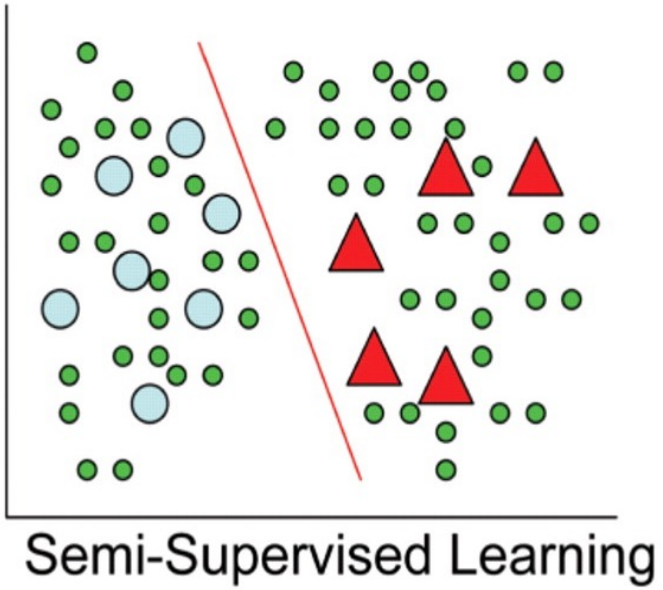
But..

But...

Semi-Supervised Learning: SSL

- **What is semi-supervised learning?**
  - Supervised learning + Additional unlabeled data
  - Unsupervised learning + Additional labeled data



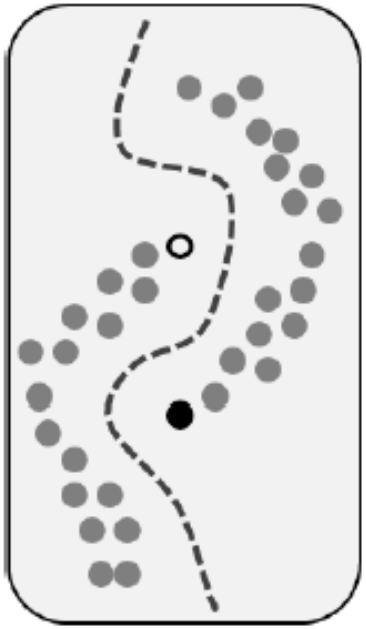


Semi-Supervised Learning: SSL

- Classification
  - **Transductive** – predict labels of unlabeled data
  - **Inductive** – learn a classification function
- Clustering (constrained clustering)
- Ranking (semi-supervised ranking)
- Almost every learning problem has a semi-supervised counterpart.

What is semi-supervised learning?

- Supervised learning + Additional unlabeled data
- Unsupervised learning + Additional labeled data



Why semi-supervised learning?

In many domains:

- Labeling could be expensive and difficult
- Unlabeled examples are easier to obtain
- Examples:
  - Web page classification
  - Speech recognition
  - Bioinformatics (e.g. protein sequences)
  - ...

Semi-Supervised Learning: SSL

- Forms:
  - **Classification,** clustering, regression, etc
- Two main settings for classification:
  - **Transductive**: Produce label only for the available unlabeled data.
    - The output of the method is not a classifier.
  - **Inductive**: Not only produce label for unlabeled data, but also produce a classifier.

Chapelle, Olivier, Schölkopf, Bernhard, and Zien, Alexander. Semi-Supervised Learning. The MIT Press, first edition, 2006.

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

Semi-Supervised Learning: SSL

- Familias de métodos en clasificación SS:
  - S3VMs
  - Métodos basados en Grafos y Label Propagation
  - Self Labeling
- Algoritmos de Self Labeling:
  - Self-Training
  - Co-Training
  - Tri-Training

Blum, Avrim and Mitchell, Tom. Combining labeled and unlabeled data with Co-Training. In Proceedings of the Annual ACM Conference on Computational Learning Theory, pp. 92–100, 1998.

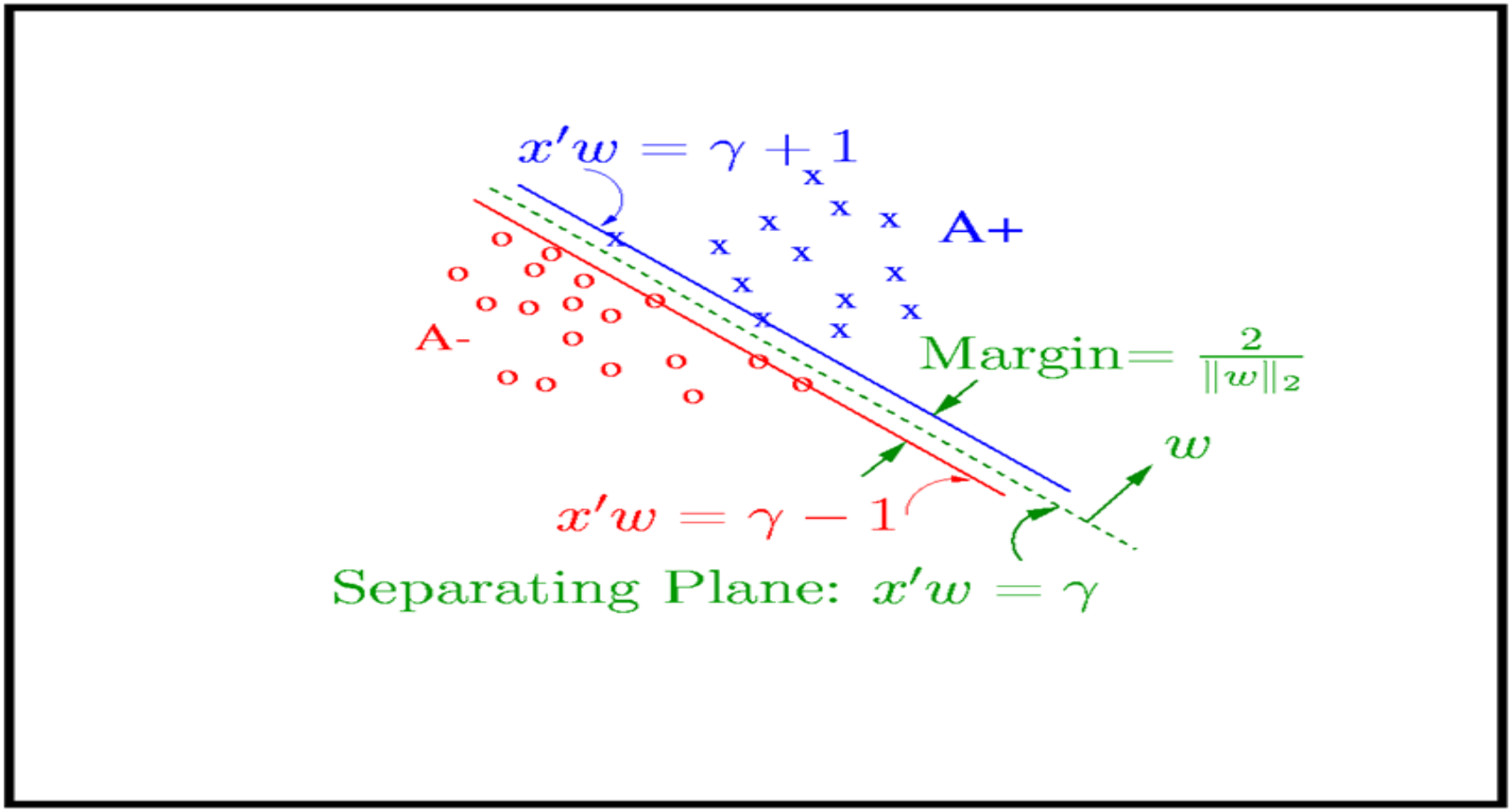
Zhou, Zhi-Hua and Li, Ming. Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transactions on Knowledge and Data Engineering, 17:1529–1541, 2005. ISSN 1041-4347.

Semi-Supervised Learning: SSL Notación

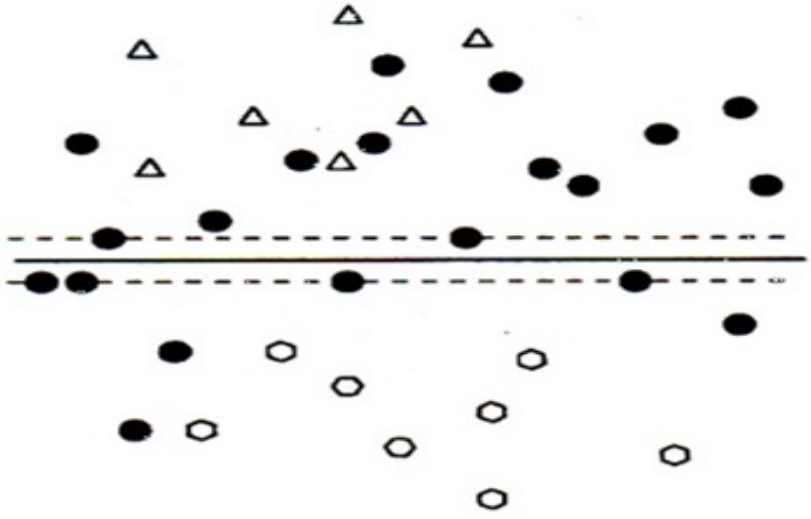
- $X_p = \{X_{p1}, X_{p2}, \ldots, X_{pD,}\ \omega\}$
- Conjunto ejemplos etiquetados *L*: *n* ejemplos X_p con ω conocida.
- Conjunto ejemplos no etiquetados *U*: *m* ejemplos X_q con ω desconocida, *m* >> *n*.
- *TR* = *L* U *U*
- Aprendizaje Transductivo:
  - Etiquetar *m* ejemplos X_q de *U*
- Aprendizaje Inductivo:
  - Clasificar los ejemplos de *TS* a partir de lo aprendido desde *TR*

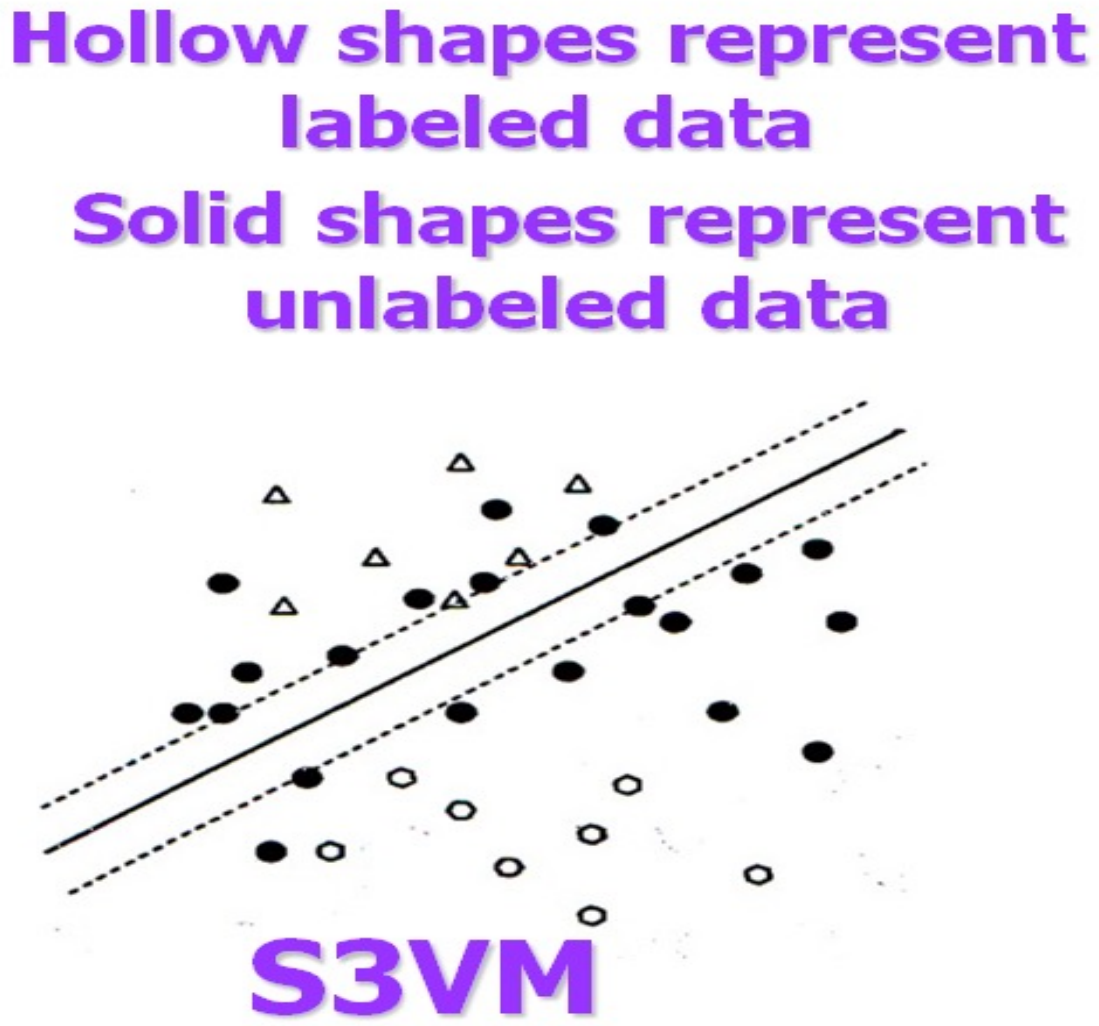
Semi-Supervised Support Vector Machines: S3VMs

Linear SVM:



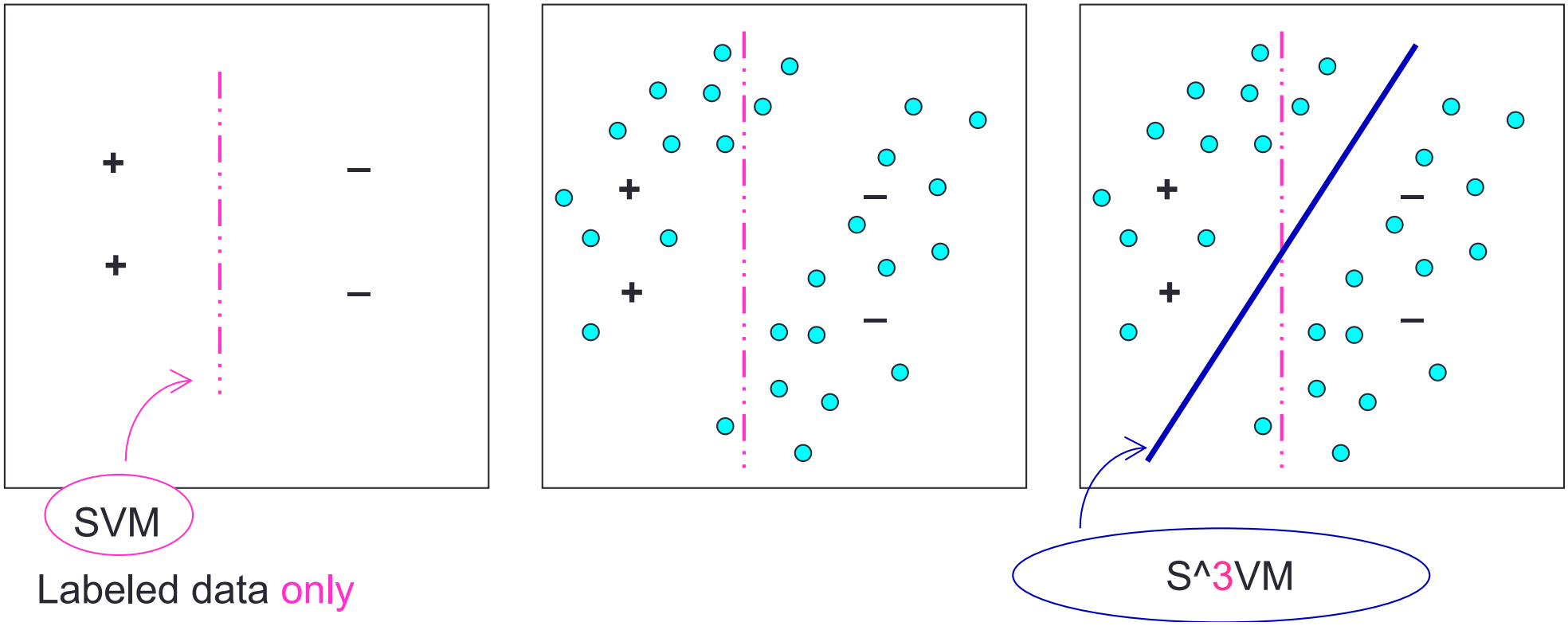
Semi-Supervised Support Vector Machines: S3VMs





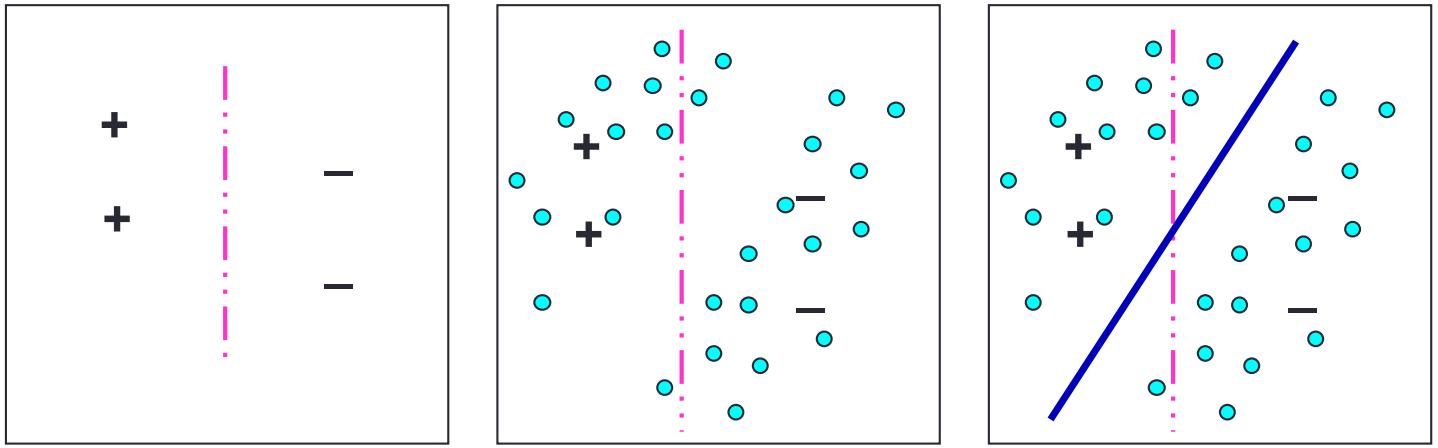
Semi-Supervised Support Vector Machines: S3VMs

- Suppose we believe target separator goes through low density regions of the space/large margin.
- Aim for separator with large margin wrt labeled and unlabeled data. (L+U)



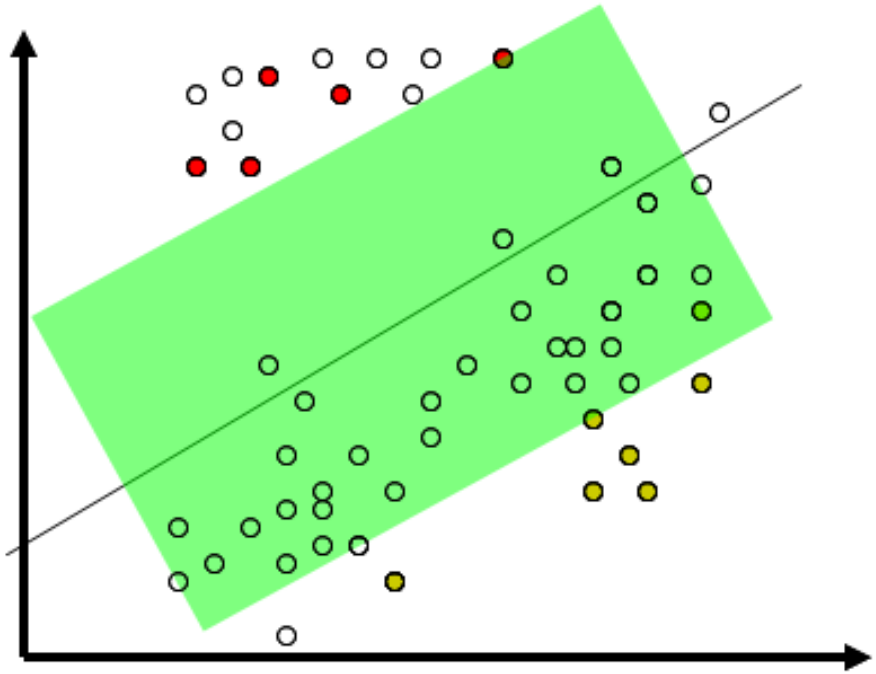
Semi-Supervised Support Vector Machines: S3VMs

- Unfortunately, optimization problem is now NP-hard. Algorithm instead does local optimization.
  - Start with large margin over labeled data. Induces labels on U.
  - Then try flipping labels in greedy fashion.
  - Or, branch-and-bound, other methods (Chapelle etal06)
- Quite successful on text data.



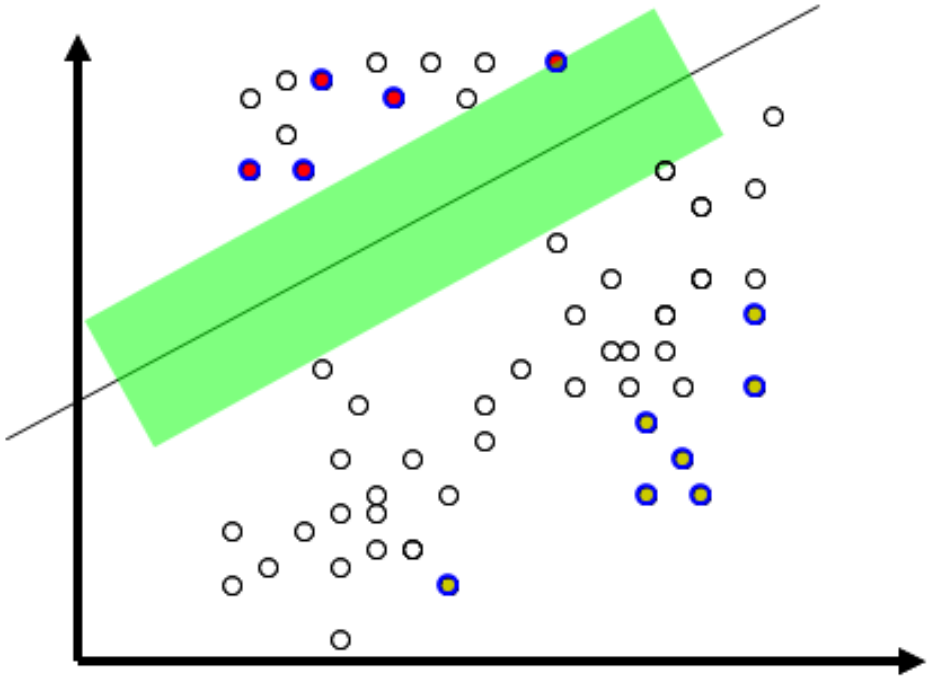
Transductive SVM

- Decision boundary given a small number of labeled examples
- How to change decision boundary given both labeled and unlabeled examples ?

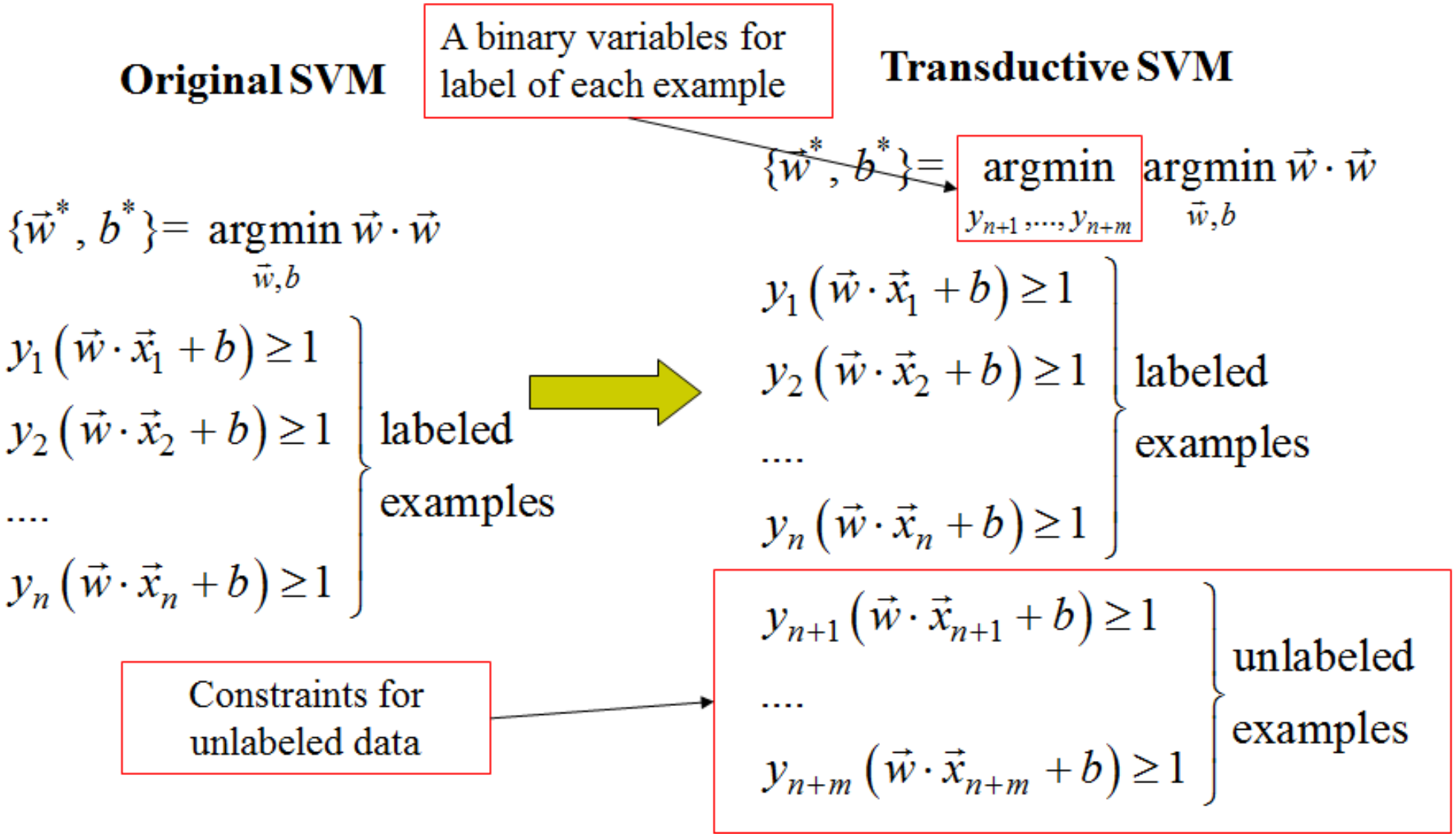


Transductive SVM

- Decision boundary given a small number of labeled examples
- Move the decision boundary to low local density



Transductive SVM



Transductive SVM

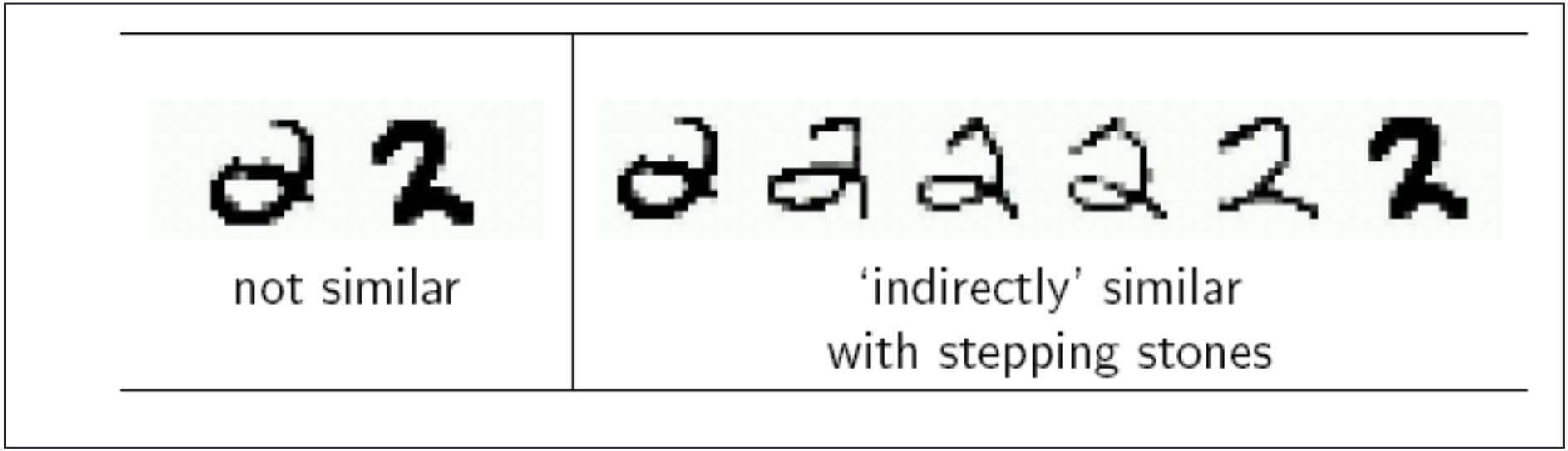
$$\{\vec{w}^*, b^*\} = \underset{y_{n+1},\ldots,y_{n+m}}{\operatorname{argmin}} \; \underset{\vec{w},b}{\operatorname{argmin}} \; \vec{w}\cdot\vec{w} + \sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\eta_i$$

$$\left.\begin{array}{l} y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1-\xi_1 \\ y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1-\xi_2 \\ \ldots. \\ y_n\left(\vec{w}\cdot\vec{x}_n + b\right) \geq 1-\xi_n \end{array}\right\} \text{labeled examples} \qquad \left.\begin{array}{l} y_{n+1}\left(\vec{w}\cdot\vec{x}_{n+1} + b\right) \geq 1+\eta_1 \\ \ldots. \\ y_{n+m}\left(\vec{w}\cdot\vec{x}_{n+m} + b\right) \geq 1+\eta_m \end{array}\right\} \text{unlabeled examples}$$

- No longer convex optimization problem.
- Alternating optimization

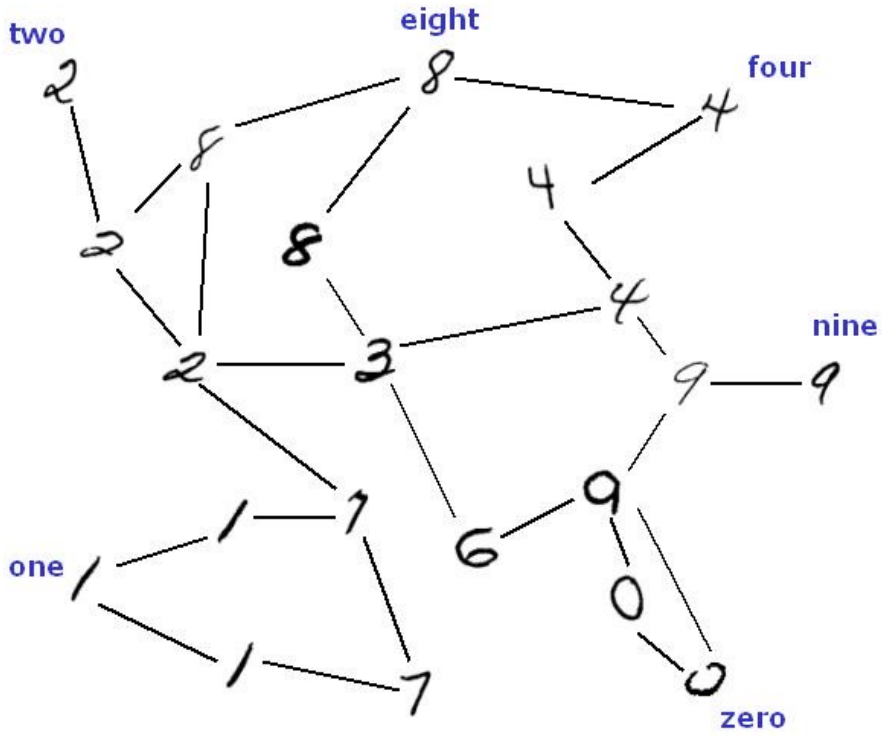
SSL: Graph-based Methods

- Suppose we believe that very similar examples probably have the same label.
- If you have a lot of labeled data, this suggests a Nearest-Neighbor type of alg.
- If you have a lot of unlabeled data, perhaps can use them as "stepping stones"



SSL: Graph-based Methods

- Idea: construct a graph with edges between very similar examples.
- Unlabeled data can help "glue" the objects of the same class together.



Solve for:

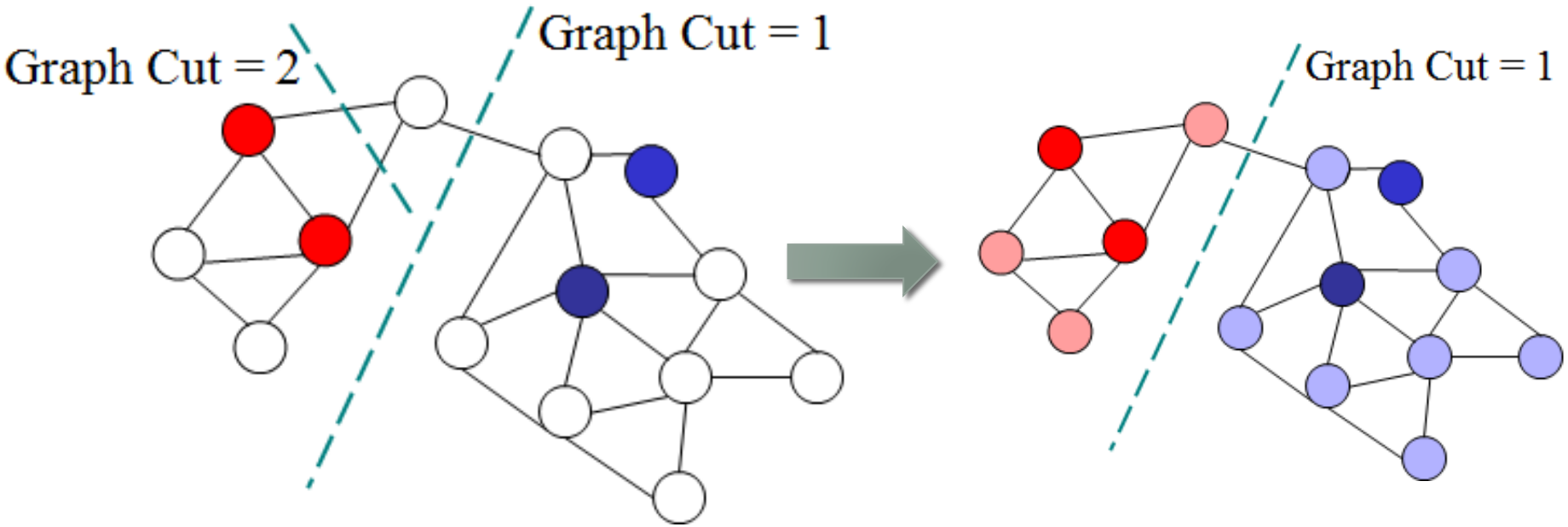
- Minimum cut
- Minimum "soft-cut"

$$\sum_{e=(u,v)}(f(u)-f(v))^2$$

- Spectral partitioning
- …

SSL: Graph-based Methods

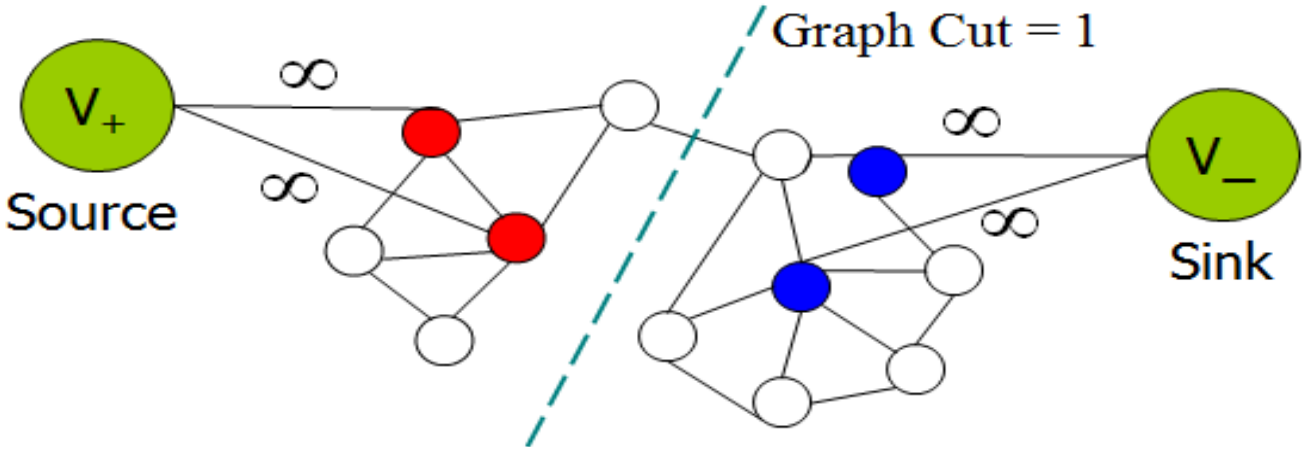
- Classification as graph partitioning
- Search for a classification boundary
  - Consistent with labeled examples
  - Partition with small graph cut



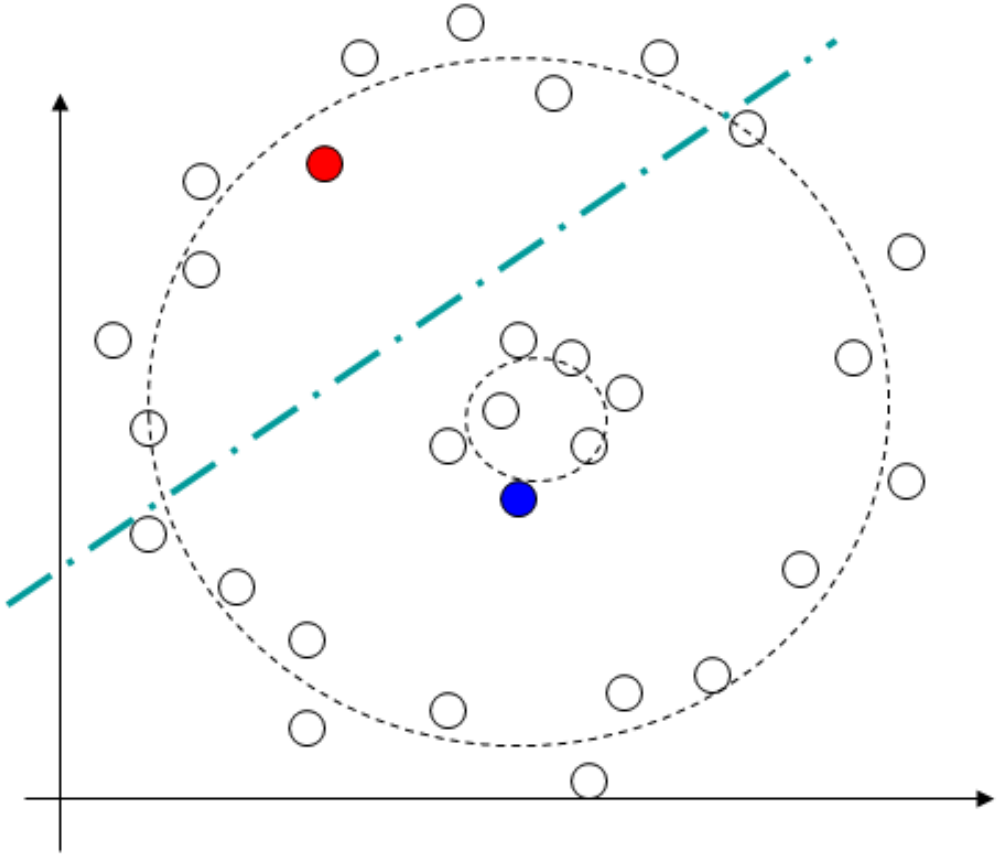
SSL: Graph-based Methods

Min-cuts [Blum and Chawla, ICML 2001]

- Additional nodes
  - V_+: source, V-: sink
  - Infinite weights connecting sinks and sources
- High computational cost

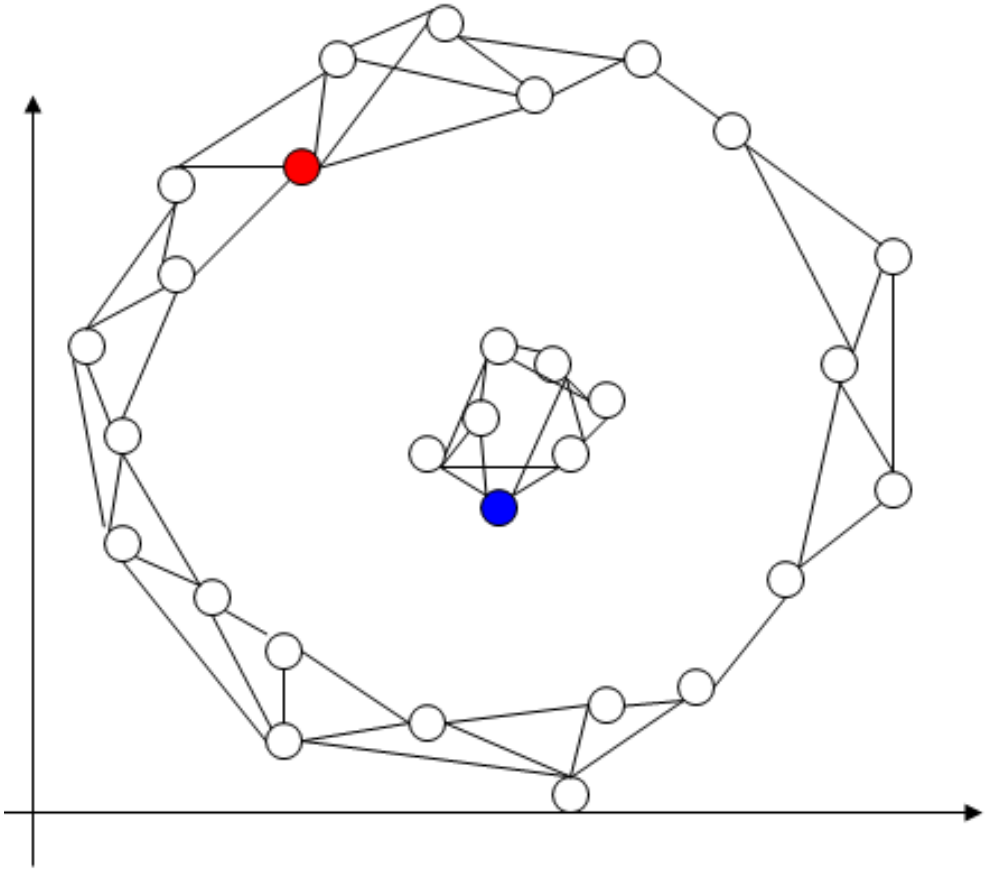


SSL: Label Propagation



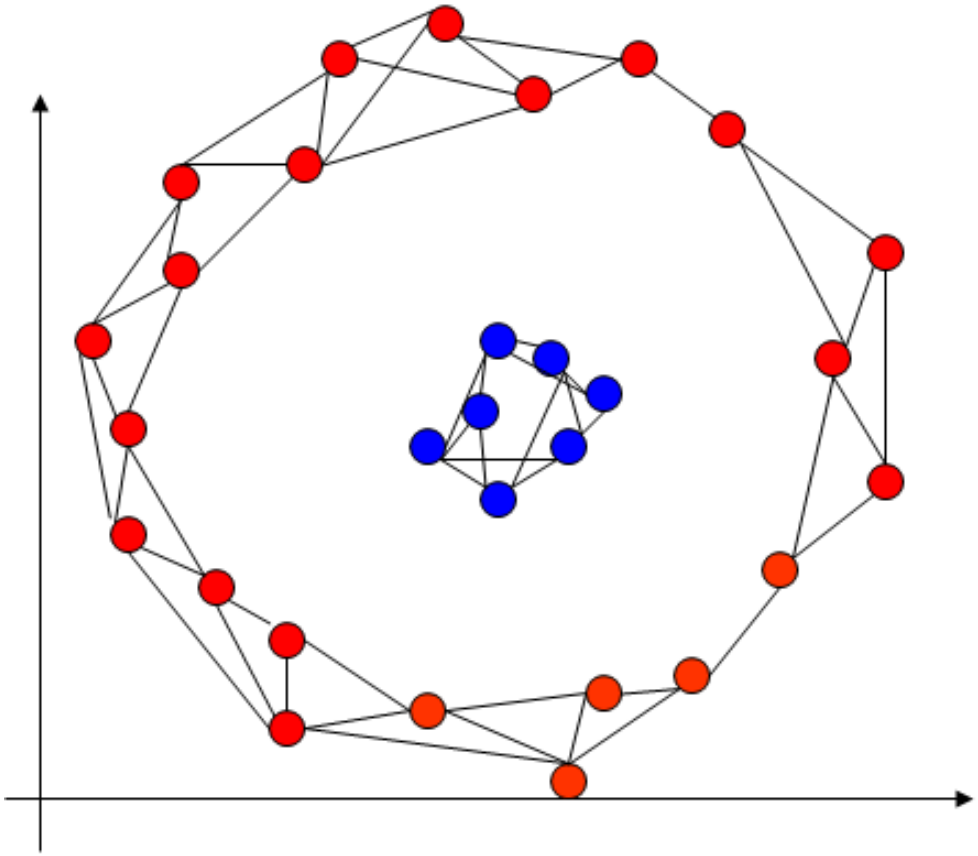
- A decision boundary based on the labeled examples is unable to take into account the layout of the data points
- How to incorporate the data distribution into the prediction of class labels?

SSL: Label Propagation



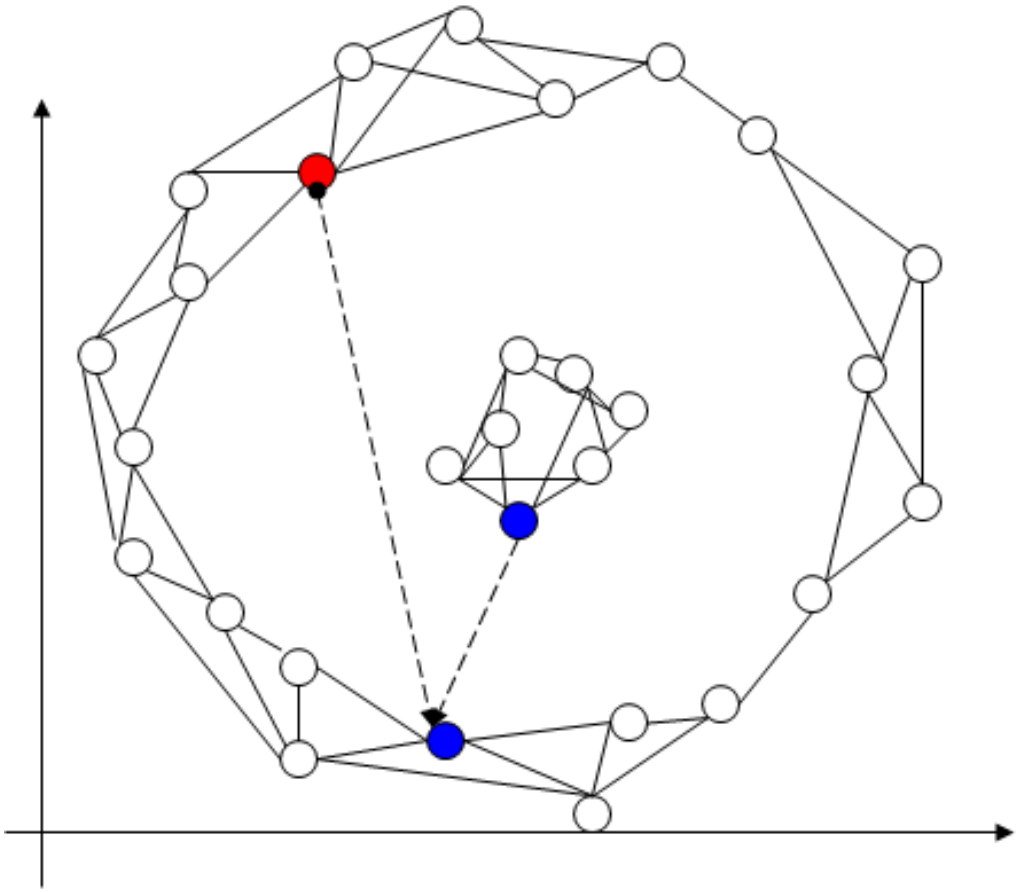
- Connect the data points that are close to each other

SSL: Label Propagation



- Connect the data points that are close to each other
- Propagate the class labels over the connected graph

SSL: Label Propagation



- Connect the data points that are close to each other
- Propagate the class labels over the connected graph
- Different from the K Nearest Neighbor

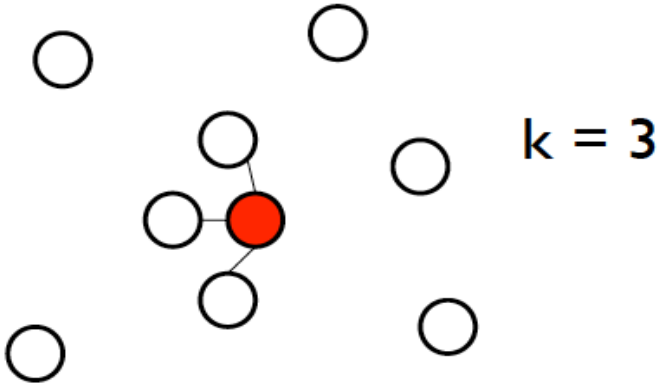
SSL: Label Propagation

Graph Construction

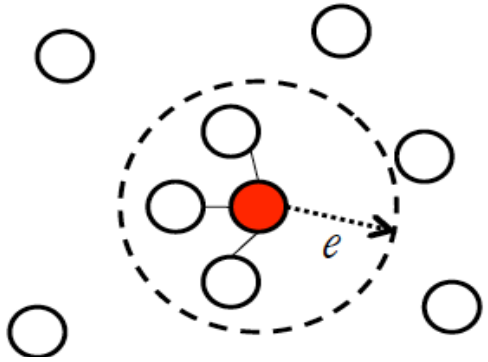
- Neighborhood Methods
  - k-NN Graph Construction (k-NNG)
  - e-Neighborhood Method
- Metric Learning
- Other approaches

SSL: Label Propagation

- k-Nearest Neighbor Graph (k-NNG)
  - add edges between an instance and its k-nearest neighbors



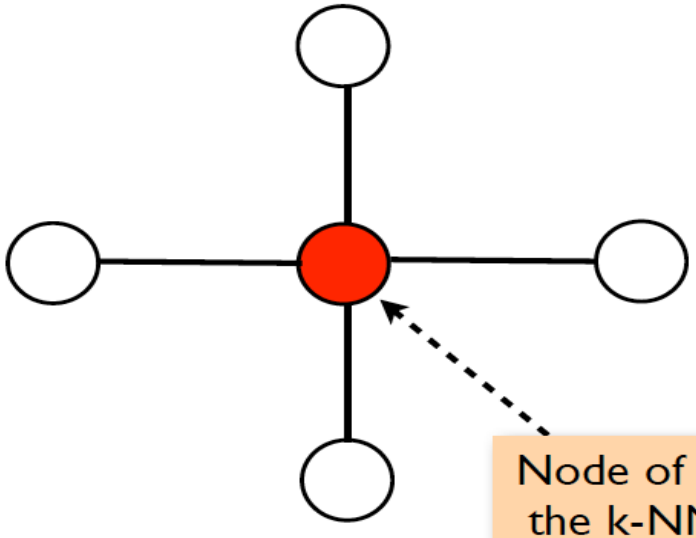
- e-Neighborhood
  - add edges to all instances inside a ball of radius e



SSL: Label Propagation
Issues with k-NNG

- Not scalable (quadratic)
- Results in an asymmetric graph
  - b is the closest neighbor of a, but not the other way

- Results in irregular graphs
  - some nodes may end up with higher degree than other nodes



SSL: Label Propagation
Issues with *e*-Neighborhood

- Not scalable
- Sensitive to value of e : not invariant to scaling
- Fragmented Graph: disconnected components

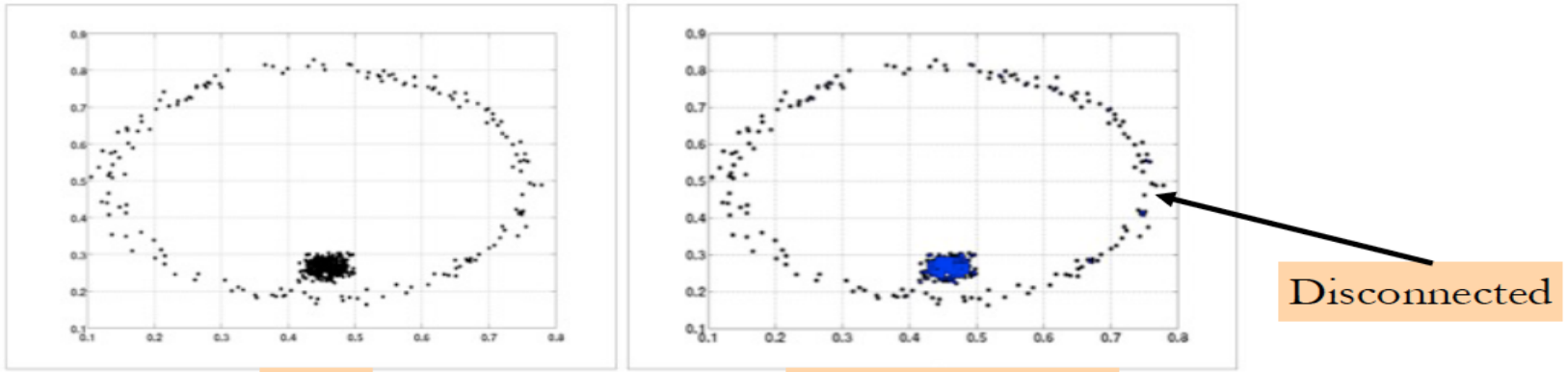
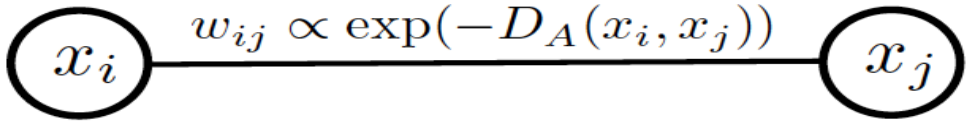


Figure from [Jebara et al., ICML 2009]

SSL: Label Propagation
Graph Construction using Metric Learning



$$D_A(x_i, x_j) = (x_i - x_j)^T A (x_i - x_j)$$

Estimated using Mahalanobis metric learning algorithms

- Supervised Metric Learning
  - ITML [Kulis et al., ICML 2007]
  - LMNN [Weinberger and Saul, JMLR 2009]
- Semi-supervised Metric Learning
  - IDML [Dhillon et al., UPenn TR 2010]

The Self-Training Algorithm

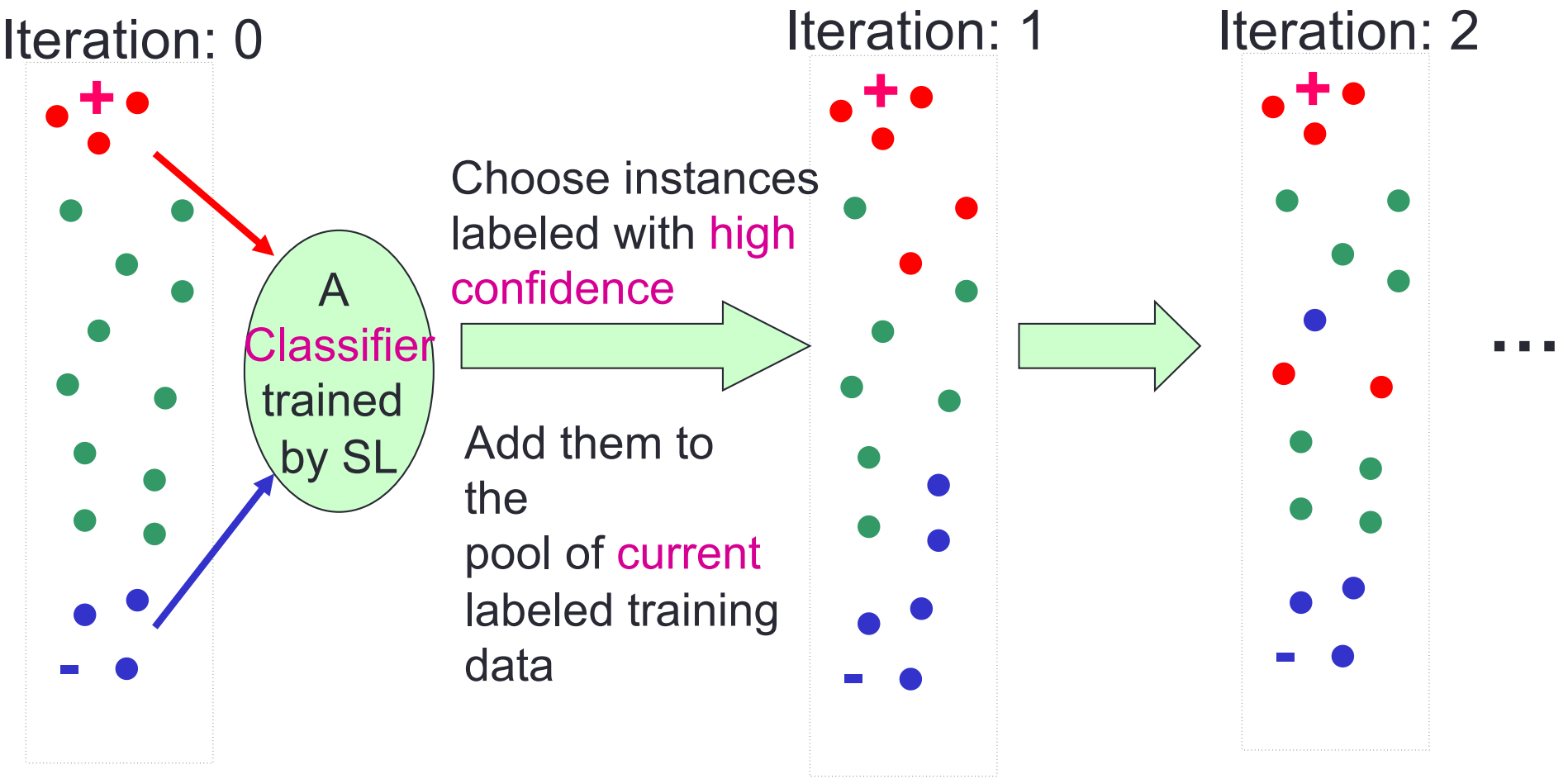
(Yarowsky 1995)

- Suposición:
  - Las propias prediciones con alta confianza son correctas.
- Algoritmo Self-training:
  1. Entrena f desde $\{(x_{1:n}, y_{1:n})\}$
  2. Predice sobre U
  3. Añade (x, f(x)) a los datos etiquetados
     A. Añade todo
     B. **Añade los más confiables**
     C. Añade pesos
  4. Repite

- Ventajas:
  - Simple
  - Wrapper
- Inconvenientes:
  1. Errores tempranos
  2. No se puede conocer su convergencia

The Self-Training Algorithm

(Yarowsky 1995)



The Co-Training Algorithm

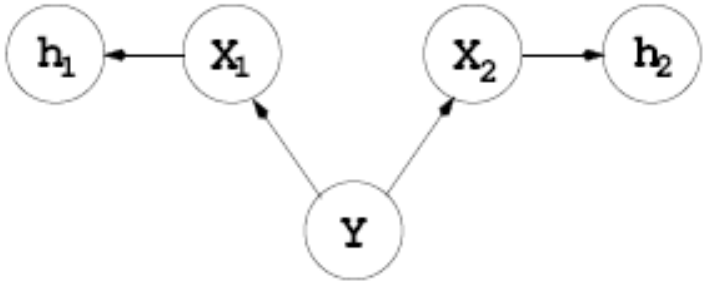


Figure 1: The co-training scenario with rules h_1 and h_2.

Two views – X1, X2

Two distinct hypothesis classes H1, H2 consisting of functions predicting Y from X1 and X2 respectively

Bootstrap using h1єH1, h2єH2

"*If X1 is conditionally independent of X2 given Y then given a weak predictor in H1 and given an algorithm which can learn H2 under random misclassification noise, then it is possible to learn a good predictor in H2*"

The Co-Training Algorithm
(Blum and Mitchell 1998)

- Instances contain two sufficient sets of features
  - i.e. an instance is $x=(x_1,x_2)$
  - Each set of features is called a View

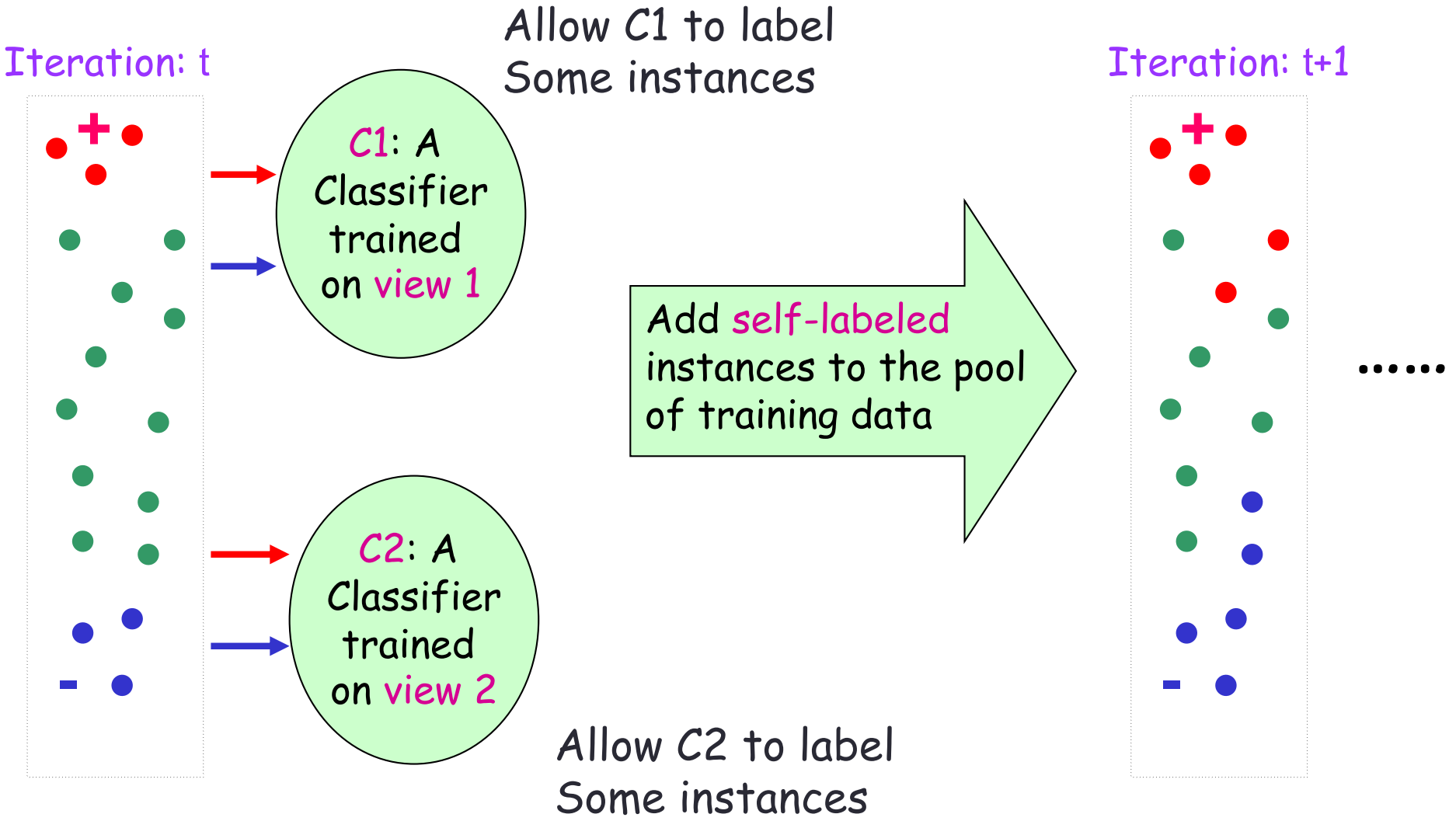
- Two views are independent given the label:

$$P(x_1|x_2,y) = P(x_1|y)$$
$$P(x_2|x_1,y) = P(x_2|y)$$

- Two views are consistent:

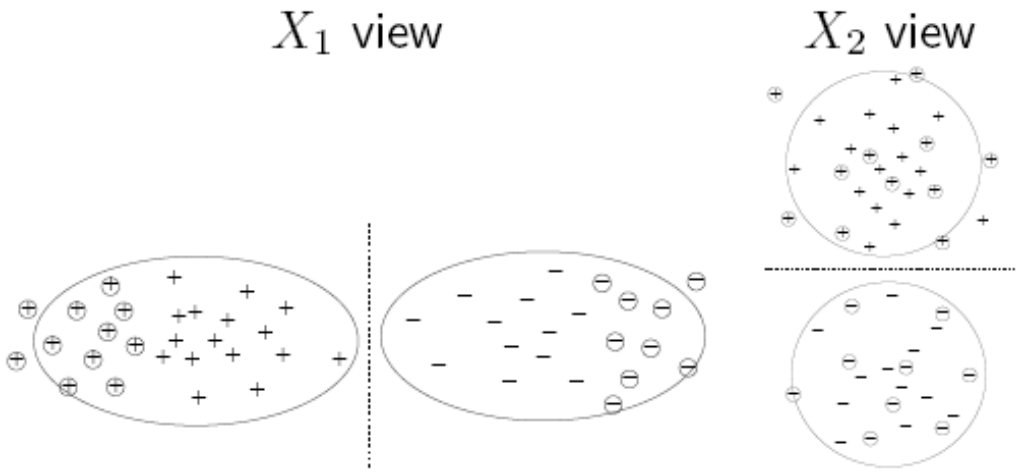
$$\exists c_1, c_2 : c^{opt}(x) = c_1(x_1) = c_2(x_2)$$

The Co-Training Algorithm



The Co-Training Algorithm

- Suposición:
  - División atributos $x=[x^{(1)}; x^{(2)}]$ existe
  - $x^{(1)}$ ó $x^{(2)}$ solo es sufficiente para entrenar un clasificador
  - $x^{(1)}$ ó $x^{(2)}$ son condicionalmente independientes dada la clase



- Algoritmo Co-Training:

1. Entrena 2 clasificadores:
   $f^{(1)}$ desde $(X_l^{(1)}, Y_l)$
   $f^{(2)}$ desde $(X_l^{(1)}, Y_l)$
2. Clasifica X_u con $f^{(1)}$ y $f^{(2)}$ separadamente.
3. Añade $f^{(1)}$'s k-más-confidentes (x, $f^{(1)}(x)$) a $f^{(2)}$'s datos etiquetados.
4. Añade $f^{(2)}$'s k-más-confidentes (x, $f^{(2)}(x)$) a $f^{(1)}$'s datos etiquetados.
5. Repite.

The Tri-Training Algorithm

- Se propuso para mejorar Co-Training, eliminando la posibilidad de dividir los atributos en dos subconjuntos.
- Utiliza tres clasificadores, h_1, h_2, h_3 ; del mismo algoritmo de aprendizaje supervisado.
- La diversidad entre clasificadores solo se puede conseguir manipulando *L (conjunto etiquetado)*.

The Tri-Training Algorithm

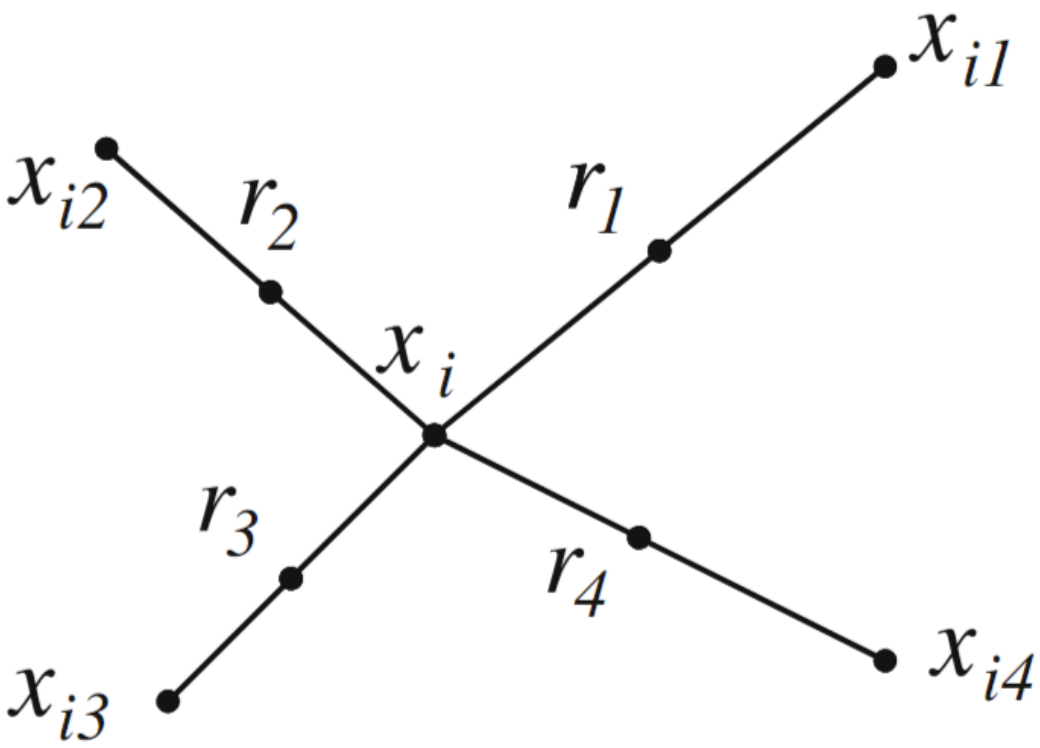
- Algoritmo
  - Muestreo con reemplazo de *L*.
  - Repite
    - Determinar los ejemplos no etiquetados para los que se obtiene mayor confianza.
    - Cada ejemplo de *U* tiene la oportunidad de ser etiquetado por h_1.
    - Un ejemplo no etiquetado X_q puede ser etiquetado para h_i, siempre que haya consenso por parte de los otros dos clasificadores. X_q se usará para modificar el modelo aprendido por h_i.
    - Se puede dar el caso que dos clasificadores predigan una clase incorrecta para el ejemplo no etiquetado, añadiendo ruido en la hipótesis aprendida por el otro clasificador. Para resolverlo, el modelo añade un mecanismo para compensar la influencia negativa de los ejemplos mal etiquetados.
  - Hasta que el modelo aprendido por los tres clasificadores no varía.
- La clasificación final se realiza mediate voto por mayoría de los clasificadores entrenados.

The Tri-Training Algorithm: TriSM

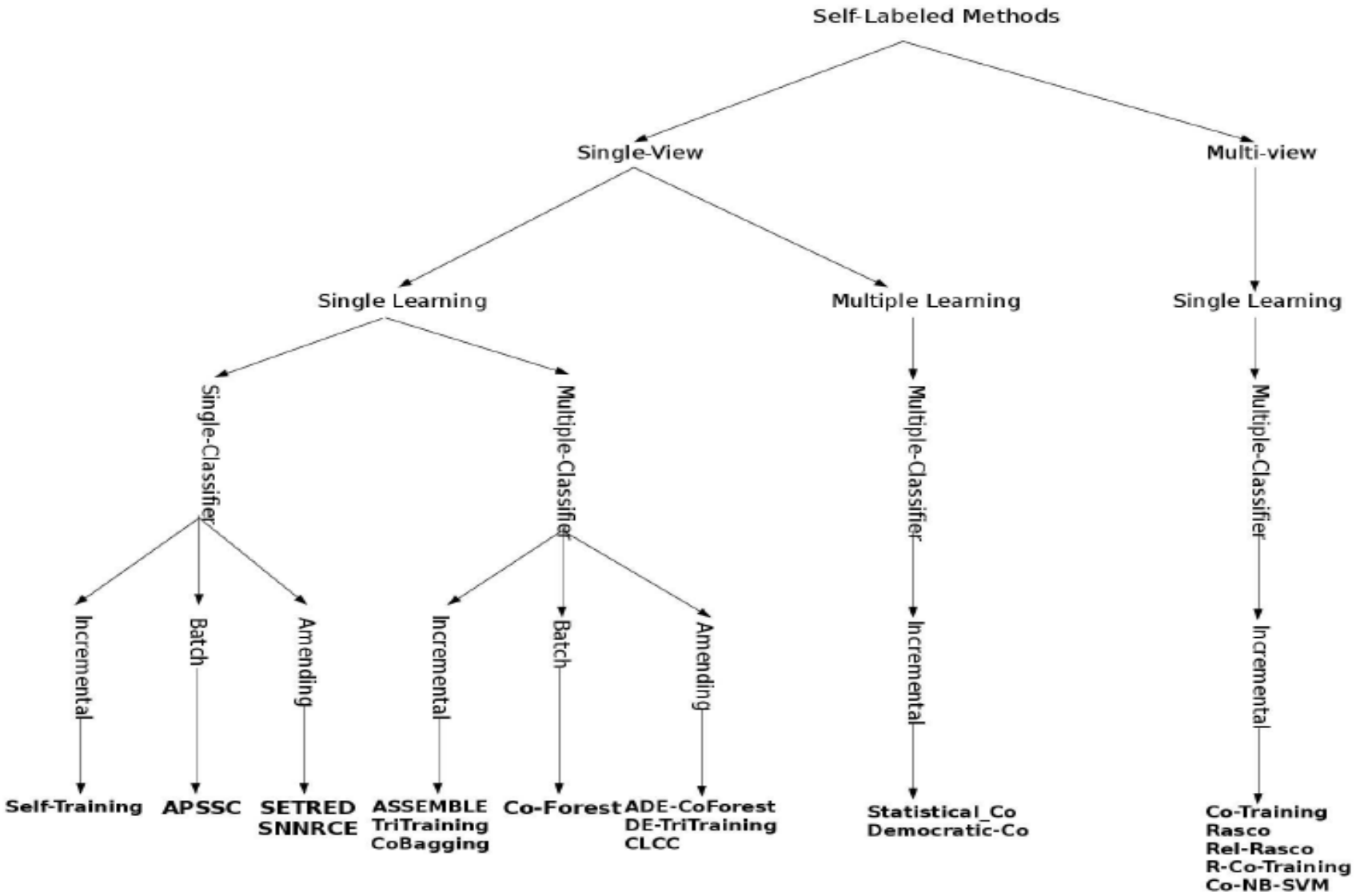
- Tri-Training presenta problemas en la búsqueda de diversidad, limitadas a los datos de entrada. El modelo de muestreo con reemplazo es insuficiente.
- Debido a que L << U, las diferencias de los muestreos no son muy significativas y se producen outliers de forma frecuente.
- Para resolver este problema, se introduce **TriSM**: Tri-training con sobremuestreo, introduciendo ejemplos sintéticos etiquetados.

The Tri-Training Algorithm: TriSM

- Con *L* completo, sobremuestreamos con una técnica basada en SMOTE, pero aplicada a todas las clases.
- Cada clasificador usará los datos etiquetados originales junto a algunos datos sintéticos.
- El ratio de sobremuestreo es constante para todas las clases.



Taxonomy of Self-Labeling Techniques



Experiments on Self-Labeling

- 55 data sets from the UCI repository.
- Base classifiers: *k*-NN, C4.5, Naive Bayes, SMO.
- Performance measures: Accuracy, Kappa, and runtime.
- 10 fold-cross validation
- We conduct different analyses:
  - Transductive and inductive capabilities.
  - Different labeled ratios: 10%, 20%, 30% and 40%
  - Number of classes: binary and multi-class problems.
  - 9 high dimensional data sets with small labeled ratio.
  - How far is SSL from supervised learning?

Associated web-page:

`http://sci2s.ugr.es/SelfLabeled`

Experiments on Self-Labeling

10%			20%			30%			40%		
	Acc	**△K**		**Acc**	**△K**		**Acc**	**△K**		**Acc**	**△K**
Democratic-Co	0.7304	-2	***TriTraining (C4.5)***	0.7609	-1	***Democratic-Co***	0.7744	-1	***Democratic-Co***	0.7834	-1
TriTraining (C4.5)	0.7297	+1	***Co-Bagging (C45)***	0.7597	+1	***TriTraining (C4.5)***	0.7708	-2	**TriTraining (C4.5)**	0.7812	-3
Co-Bagging (C4.5)	0.7278	+1	**Democratic-Co**	0.7545	-3	***Co-Bagging (C4.5)***	0.7707	0	*Co-Training (SMO)*	0.7781	+2
Self-Training (C4.5)	0.7214	0	Self-Training (C4.5)	0.7526	-1	Self-Training (C4.5)	0.7652	-2	Self-Training (C4.5)	0.7754	-3
SETRED	0.7178	0	Co-Training (C4.5)	0.7524	-2	*Co-Training (SMO)*	0.7642	+4	Co-Training (C4.5)	0.7754	-3
TriTraining (KNN)	0.7176	-11	*Co-Training (SMO)*	0.7447	+3	Co-Training (C4.5)	0.7639	-3	*Co-Bagging (SMO)*	0.7738	+3
SNNRCE	0.7151	-2	Co-Bagging (KNN)	0.7431	-8	Co-Bagging (SMO)	0.7568	-1	*TriTraining (SMO)*	0.7707	+3
CoForest	0.7124	+1	*TriTraining (SMO)*	0.7414	+4	Self-Training (SMO)	0.7553	+1	Self-Training (SMO)	0.7700	+2
Self-Training (KNN)	0.7101	-3	SETRED	0.7406	-2	SETRED	0.7550	-2	Co-Bagging (C4.5)	0.7638	0
Co-Bagging (KNN)	0.7097	-3	DE-TriTraining (SMO)	0.7394	0	TriTraining (SMO)	0.7545	+5	SETRED	0.7628	-1
Co-Training (C4.5)	0.7088	-3	DE-TriTraining (C4.5)	0.7387	-6	Co-Bagging (KNN)	0.7536	-3	Co-Bagging (KNN)	0.7628	-7
Co-Training (SMO)	0.7069	+6	Co-Bagging (SMO)	0.7386	+4	Self-Training (KNN)	0.7530	0	DE-TriTraining (C4.5)	0.7605	-5
DE-TriTraining (C4.5)	0.7058	-5	CoForest	0.7372	-1	DE-TriTraining (SMO)	0.7527	+3	DE-TriTraining (SMO)	0.7593	+3
DE-TriTraining (SMO)	0.7019	-1	Self-Training (KNN)	0.7367	+1	DE-TriTraining (C4.5)	0.7500	-1	Rasco (C4.5)	0.7591	+2
DE-TriTraining (KNN)	0.7016	-7	SNNRCE	0.7350	-4	CoForest	0.7497	+2	Self-Training (KNN)	0.7586	0
TriTraining (SMO)	0.6988	+6	TriTraining (KNN)	0.7303	-6	SNNRCE	0.7474	-4	Rel-Rasco (C4.5)	0.7578	+2
Self-Training (SMO)	0.6969	+9	Co-Training (KNN)	0.7302	-3	Co-Training (KNN)	0.7454	-2	CoForest	0.7565	-2
Co-Training (KNN)	0.6958	-5	Self-Training (SMO)	0.7298	+9	DE-TriTraining (KNN)	0.7426	-8	SNNRCE	0.7545	-3
Co-Training (NB)	0.6949	0	DE-TriTraining (KNN)	0.7294	-4	TriTraining (KNN)	0.7412	-8	Co-Training (KNN)	0.7539	-1
...	...	...	...	...	...	...	...	...	...	...	...

Experiments on Self-Labeling

Statistical comparison: Friedman test + Bergmann-Hommel to find out distinctive algorithms in $n * n$ comparisons.

Figure: Transductive

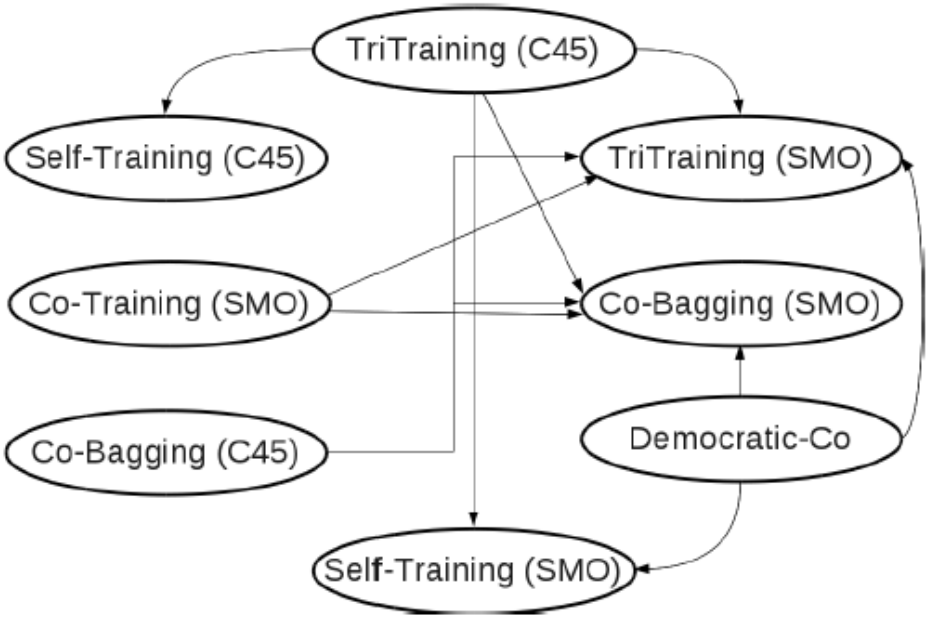
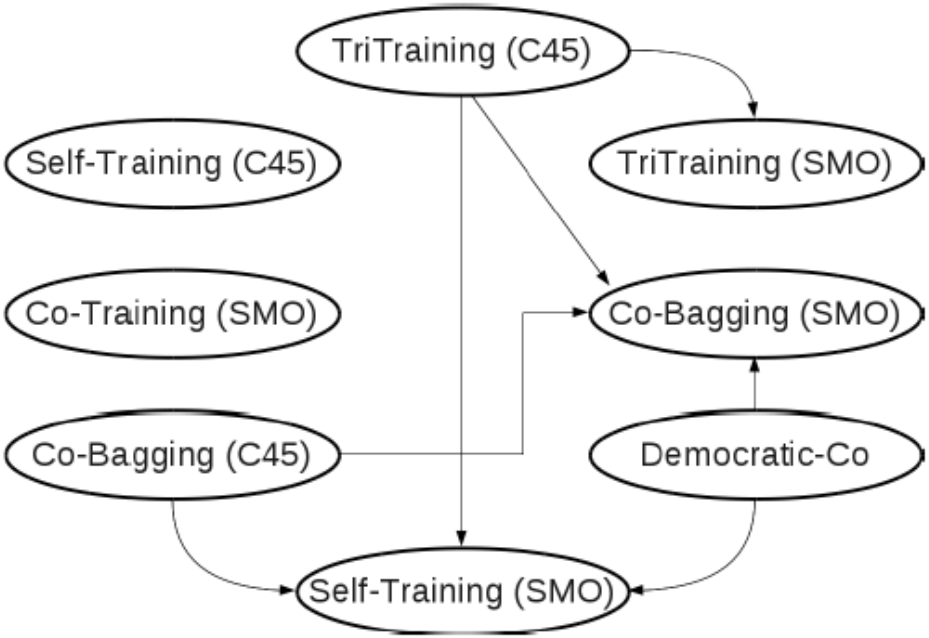


Figure: Inductive



Experiments on Self-Labeling Lessons Learned

- Multiple-classifier+single view approaches have shown the best behavior.
- Experiments on high dimensional data with very reduced labeled data denote that much more work is required.
- SSL techniques are quite far from supervised learning, especially with a reduced labeled ratio (10%).

Self-Labeling: Our ideas

Two different works:

- **On the Characterization of Noise Filters for Self-Training Semi-Supervised Learning**.
- A Framework based on Synthetic Examples Generation for Self-Labeled Semi-Supervised Classification.

Associated web-page:

`http://sci2s.ugr.es/SelfTraining+Filters/`

Self-Labeling: Our ideas

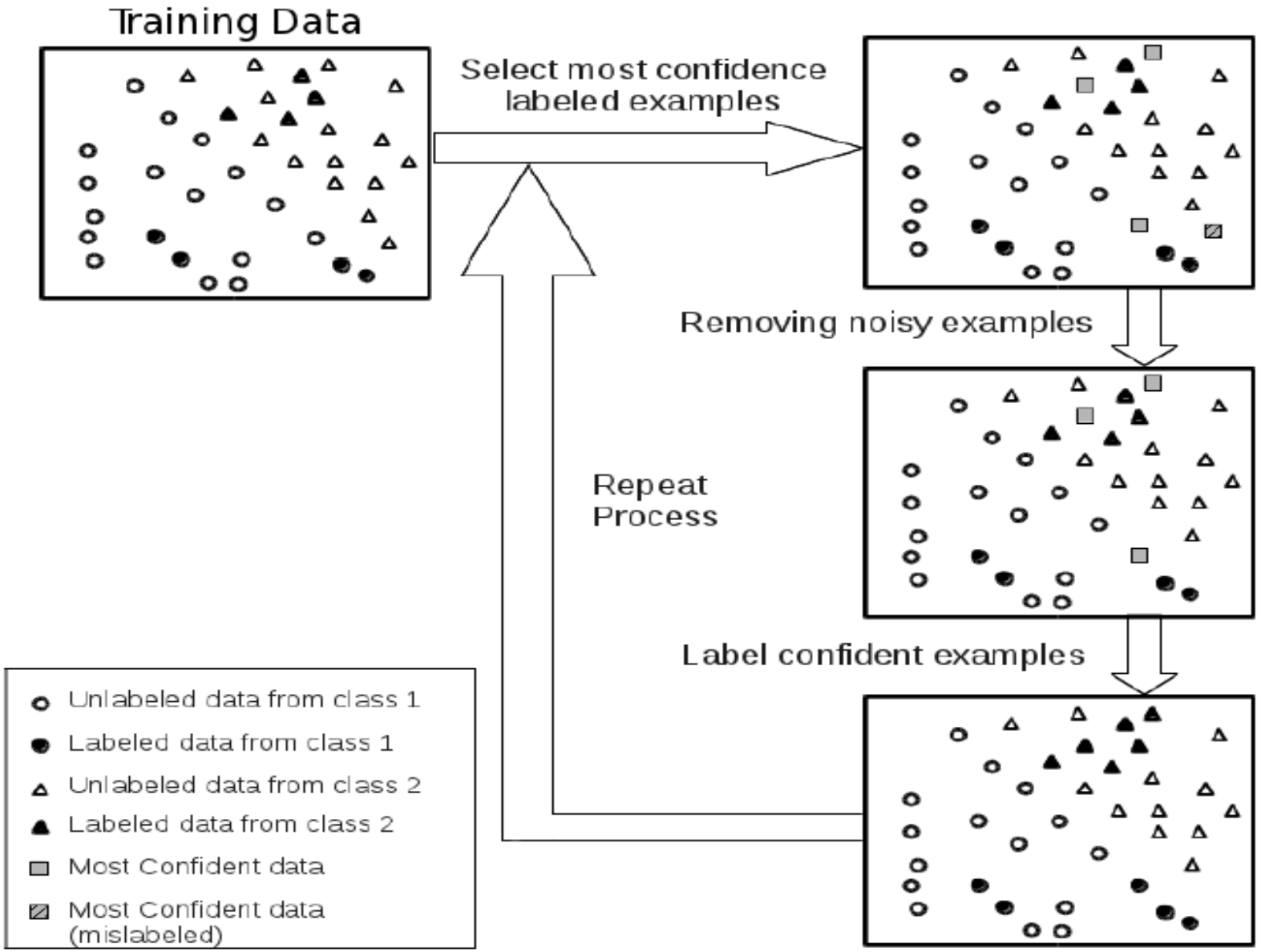
Motivation:

- Self-Training approach exemplifies the behavior of self-labeling techniques.
- Two types of noisy instances may appear:
  - Labeled data distribution may lead to wrong classifications.
  - There may be outliers within the original unlabeled data.
- Prototype selection models may detect noisy data.

Objectives:

- To remove noisy instance during the self-training process.
- To characterize which noise filters are more appropriate.

Self-Labeling: Our ideas



Self-Labeling: Our ideas

Two different works:

- On the Characterization of Noise Filters for Self-Training Semi-Supervised Learning.
- **A Framework based on Synthetic Examples Generation for Self-Labeled Semi-Supervised Classification.**

Self-Labeling: Our ideas

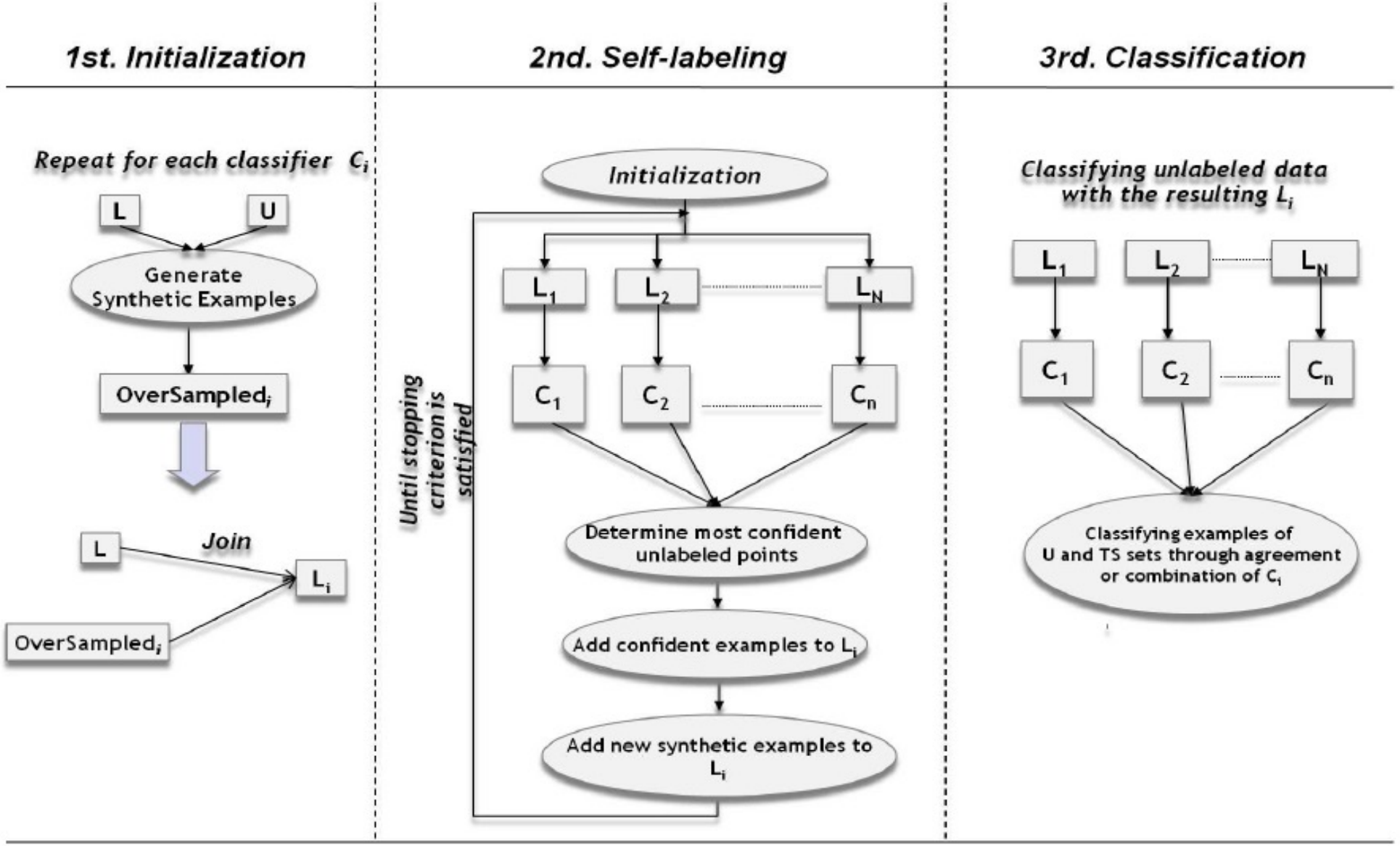
Motivation:

- Self-labeled techniques are limited by the number of labeled points and their distribution.
- Multiple classifier models use diversity mechanisms (bootstrapping). They behave as classical approaches when the number of labeled data is insufficient.

Objectives:

- To generate synthetic labeled data, aiming to:
  - Introduce diversity to multiple classifier approaches.
  - Fulfill the labeled data distribution.
- This synthetic data will be generated by PG models.

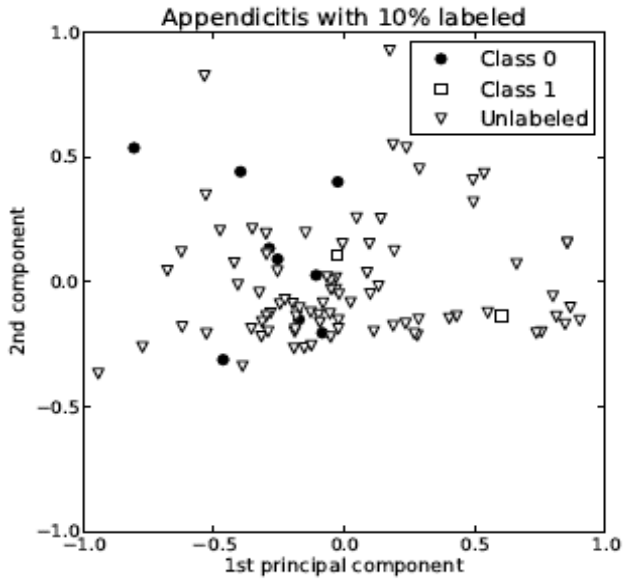
Self-Labeling: Our ideas

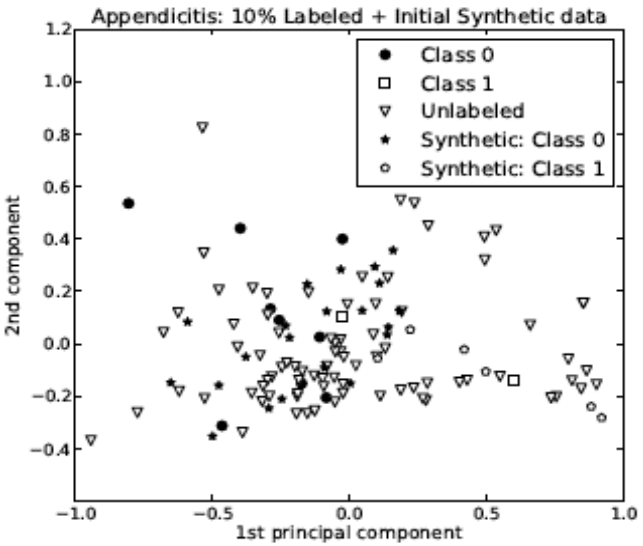


Self-Labeling: Our ideas

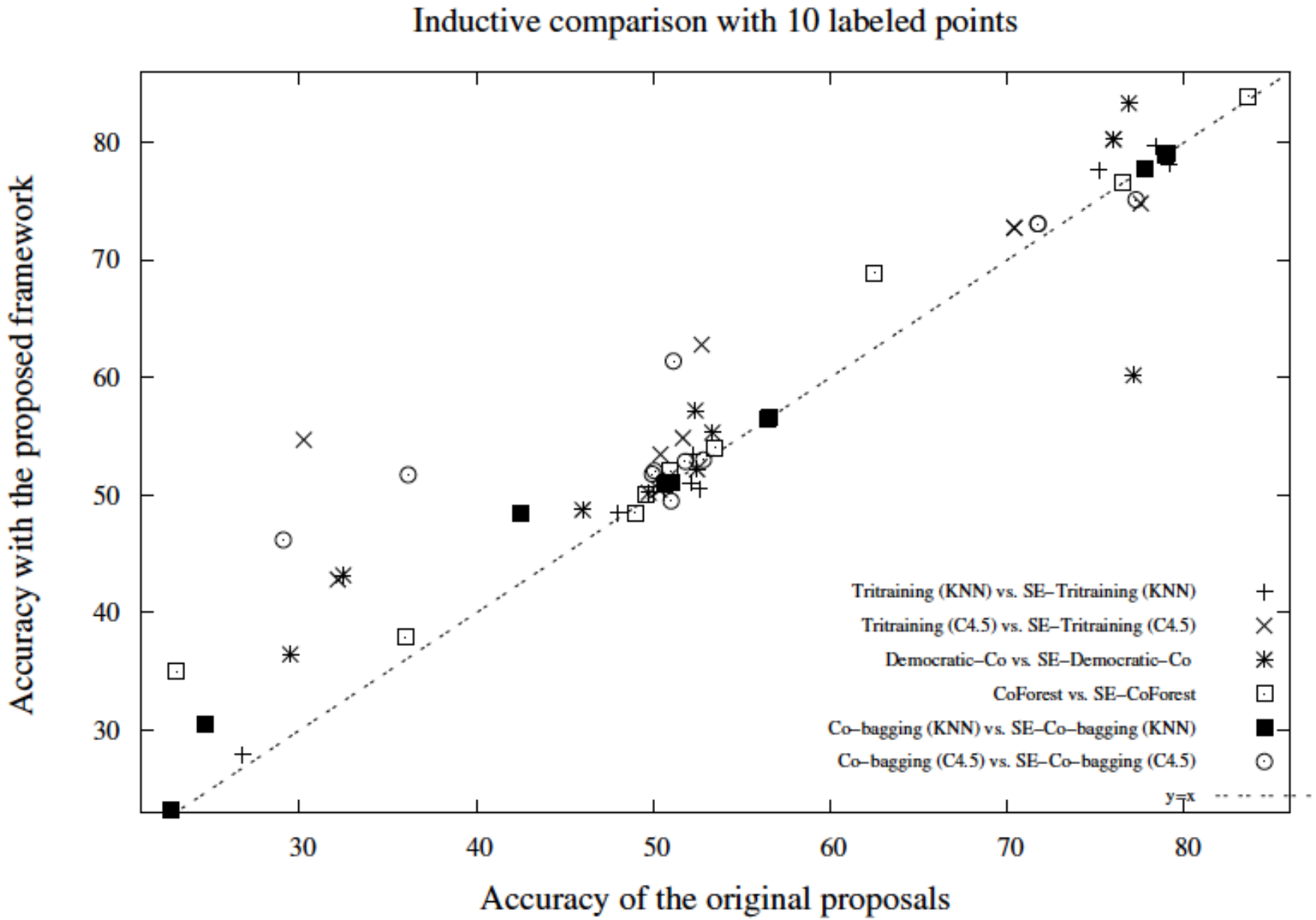
How can we generate new labeled data?

- Multi-class oversampling by using labeled and unlabeled examples.
- **Caveat**: It may generate noisy examples.
- **Solution**: Positioning adjustment based on DE.





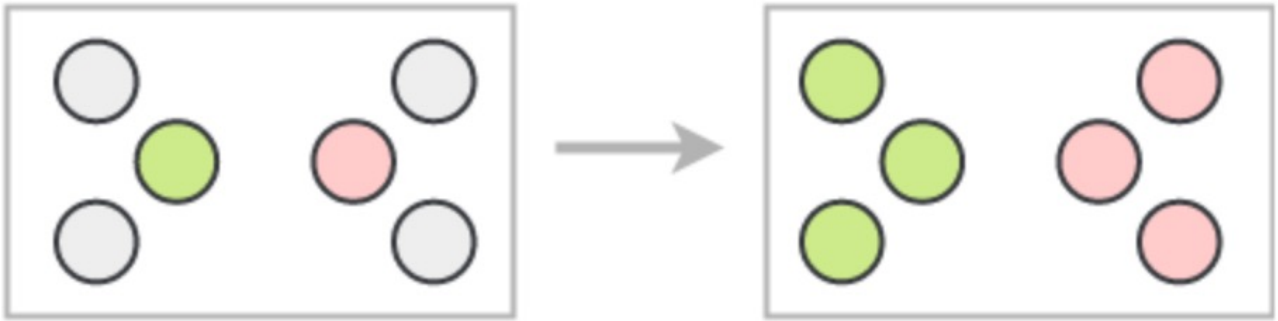
Experiments on high dimensional data sets:



Semi-supervised Learning in Computer Vision

Self-Training

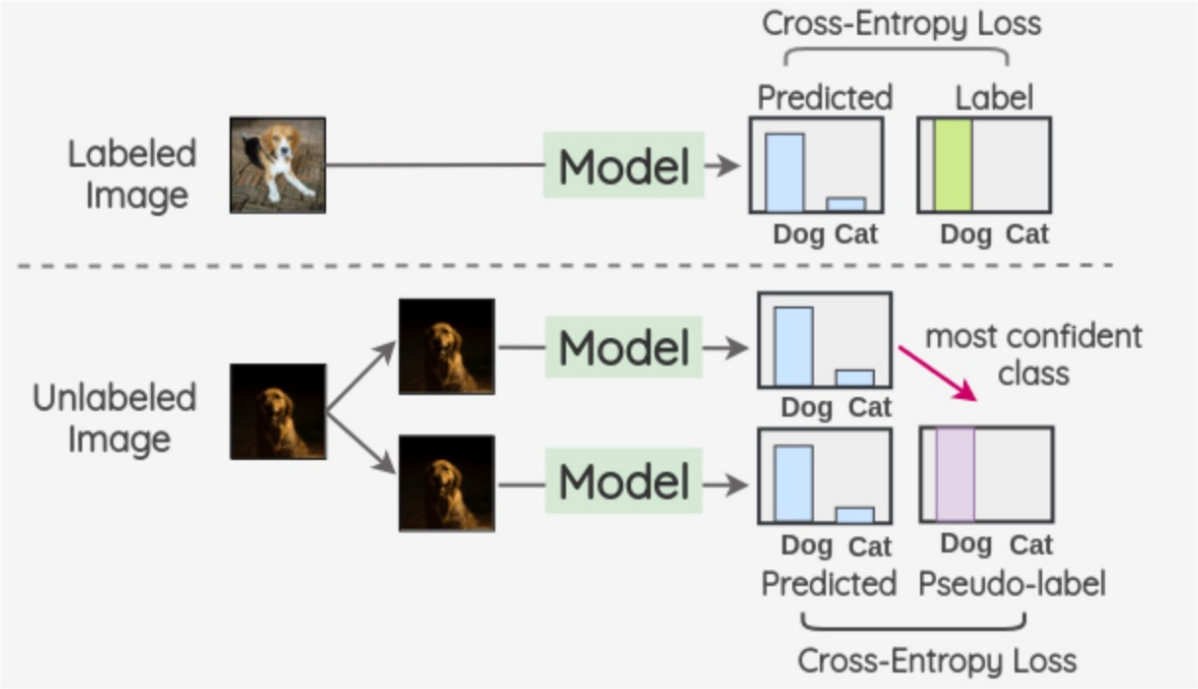
In this semi-supervised formulation, a model is trained on labeled data and used to predict pseudo-labels for the unlabeled data. The model is then trained on both ground truth labels and pseudo-labels simultaneously.



Semi-supervised Learning in Computer Vision

Pseudo-Label

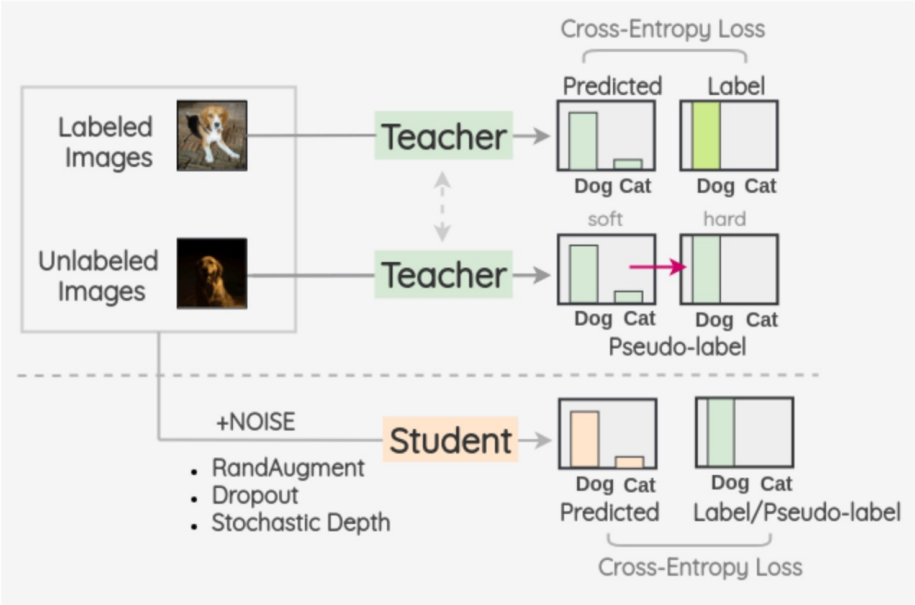
The idea is to train a model simultaneously on a batch of both labeled and unlabeled images. The model is trained on labeled images in usual supervised manner with a cross-entropy loss. The same model is used to get predictions for a batch of unlabeled images and the maximum confidence class is used as the pseudo-label. Then, cross-entropy loss is calculated by comparing model predictions and the pseudo-label for the unlabeled images.



Semi-supervised Learning in Computer Vision

Noisy Student

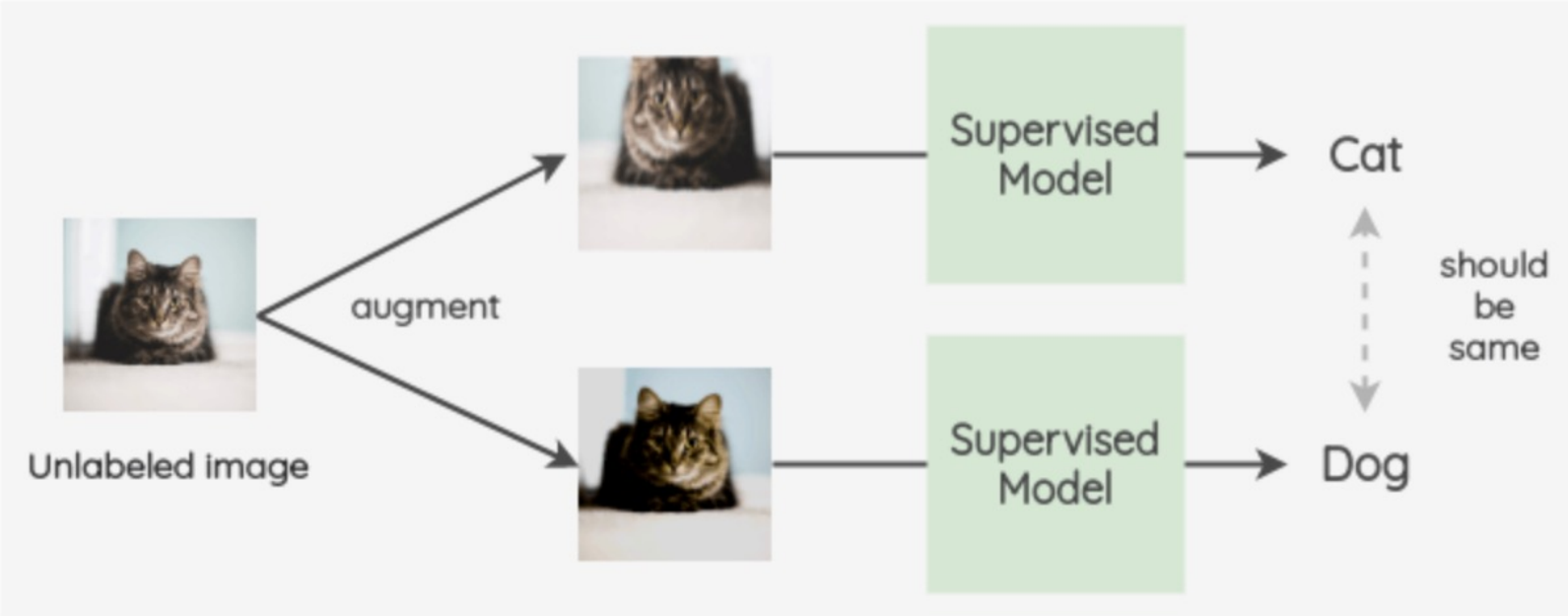
The key idea is to train two separate models called “Teacher” and “Student”. The teacher model is first trained on the labeled images and then it is used to infer the pseudo-labels for the unlabeled images. These pseudo-labels can either be soft-label or converted to hard-label by taking the most confident class. Then, the labeled and unlabeled images are combined together and a student model is trained on this combined data. The images are augmented using RandAugment as a form of input noise. Also, model noise such as Dropout and Stochastic Depth are incorporated in the student model architecture.



Semi-supervised Learning in Computer Vision

Consistency Regularization

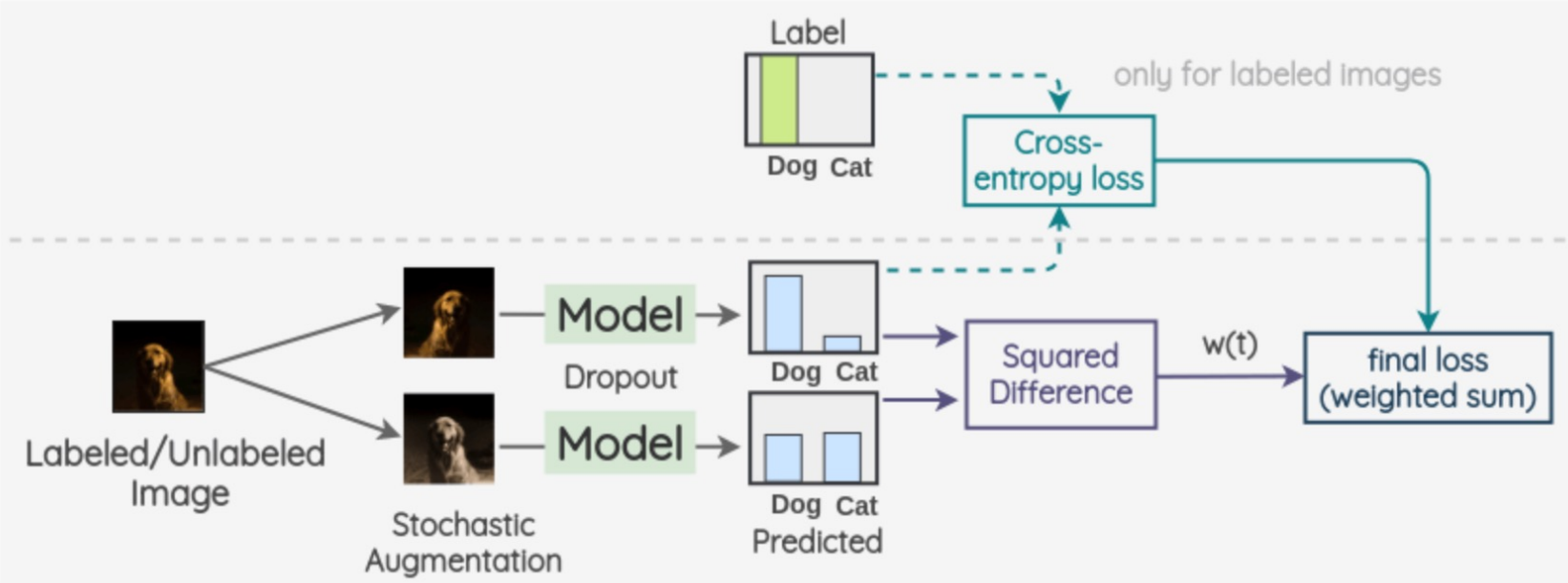
This paradigm uses the idea that model predictions on an unlabeled image should remain the same even after adding noise. We could use input noise such as Image Augmentation and Gaussian noise. Noise can also be incorporated in the architecture itself using Dropout.



Semi-supervised Learning in Computer Vision

π-model

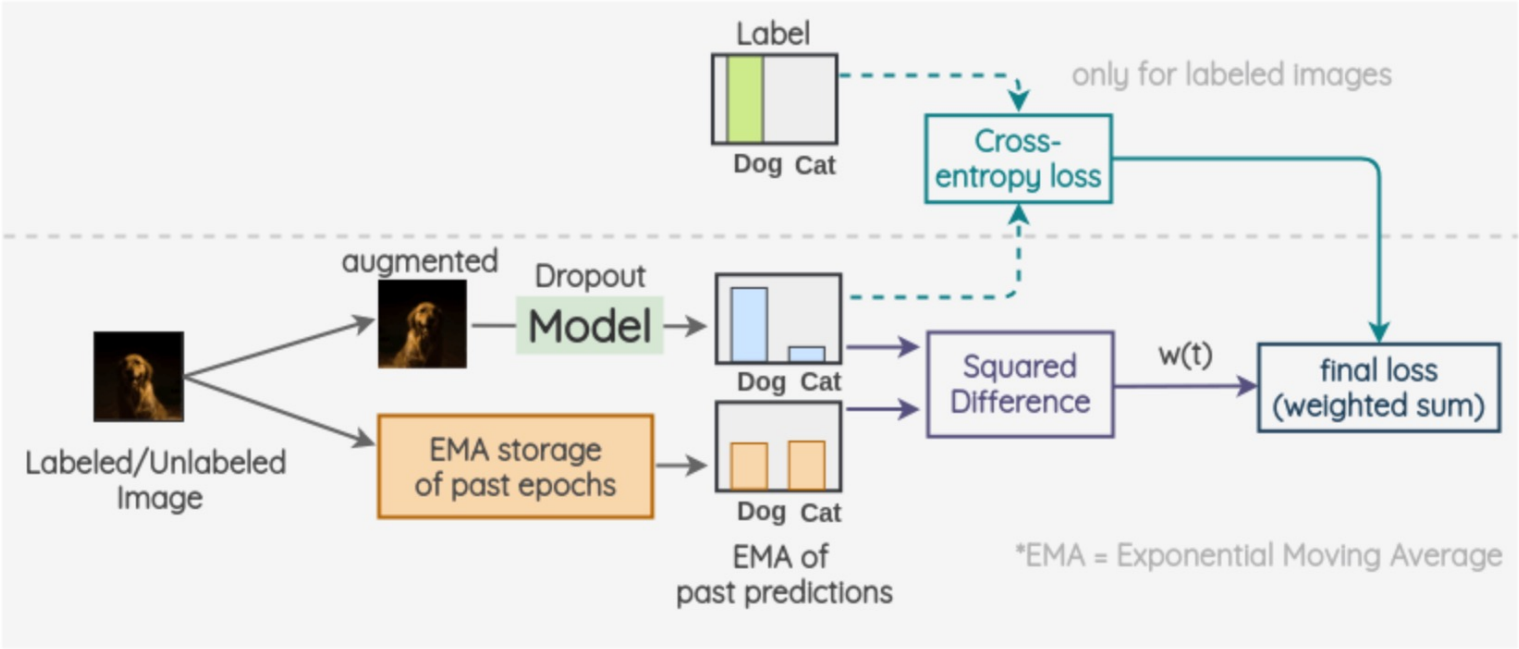
The key idea is to create two random augmentations of an image for both labeled and unlabeled data. Then, a model with dropout is used to predict the label of both these images. The square difference of these two predictions is used as a consistency loss. For labeled images, we also calculate the cross-entropy loss. The total loss is a weighted sum of these two loss terms. A weight w(t) is applied to decide how much the consistency loss contributes in the overall loss.



Semi-supervised Learning in Computer Vision

Temporal Ensembling

The key idea is to use the exponential moving average of past predictions as one view. To get another view, we augment the image as usual and a model with dropout is used to predict the label. The square difference of current prediction and EMA prediction is used as a consistency loss. For labeled images, we also calculate the cross-entropy loss. The final loss is a weighted sum of these two loss terms. A weight w(t) is applied to decide how much the consistency loss contributes in the overall loss.

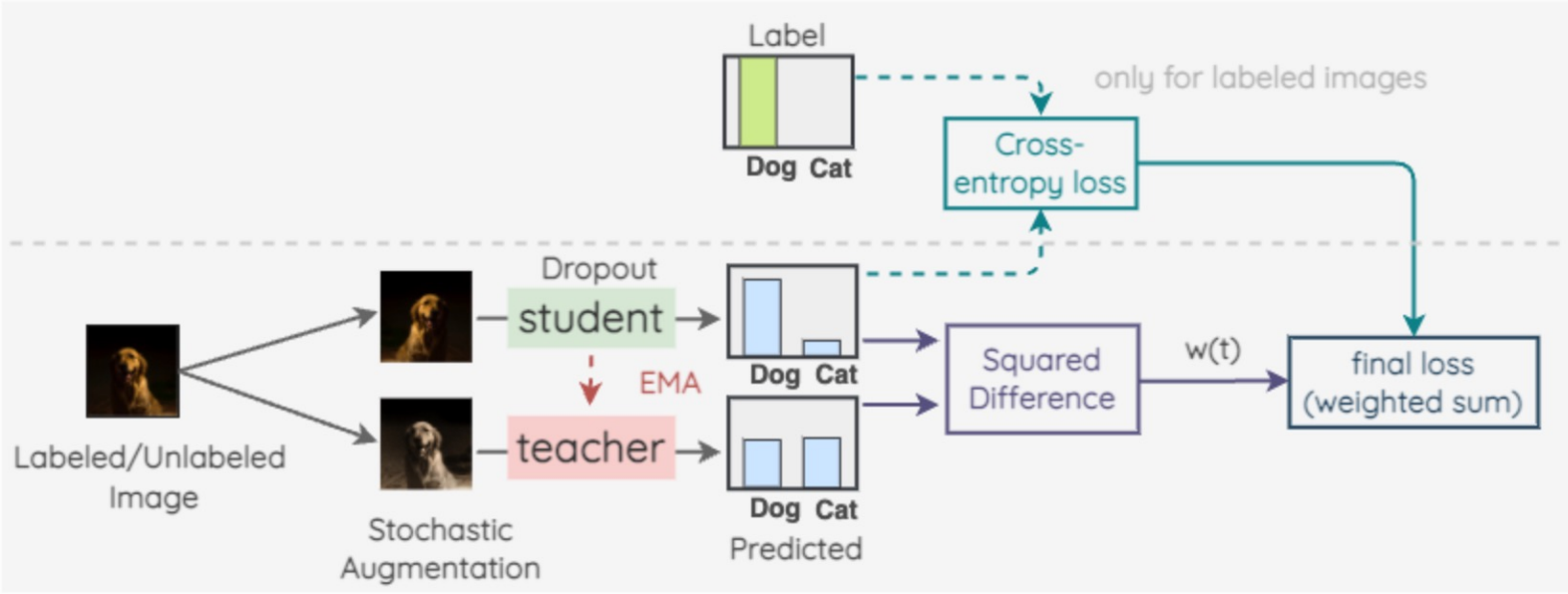


Semi-supervised Learning in Computer Vision

Mean Teacher

The general approach is similar to Temporal Ensembling but it uses Exponential Moving Average(EMA) of the model parameters instead of predictions.

The key idea is to have two models called “Student” and “Teacher”. The student model is a regular model with dropout. And the teacher model has the same architecture as the student model but its weights are set using an exponential moving average of the weights of student model. For a labeled or unlabeled image, we create two random augmented versions of the image. Then, the student model is used to predict label distribution for first image. And, the teacher model is used to predict the label distribution for the second augmented image. The square difference of these two predictions is used as a consistency loss. For labeled images, we also calculate the cross-entropy loss. The final loss is a weighted sum of these two loss terms. A weight w(t) is applied to decide how much the consistency loss contributes in the overall loss.

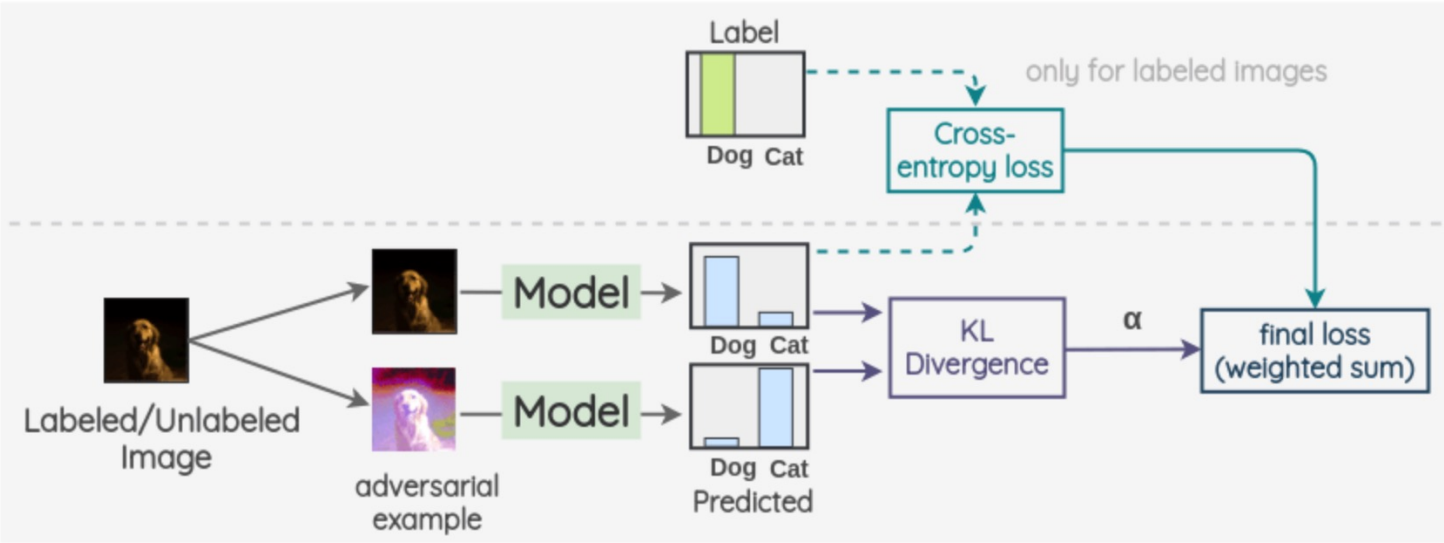


Semi-supervised Learning in Computer Vision

Virtual Adversarial Training

The key idea is to generate an adversarial transformation of an image that will change the model prediction. To do so, first, an image is taken and an adversarial variant of it is created such that the KL-divergence between the model output for the original image and the adversarial image is maximized.

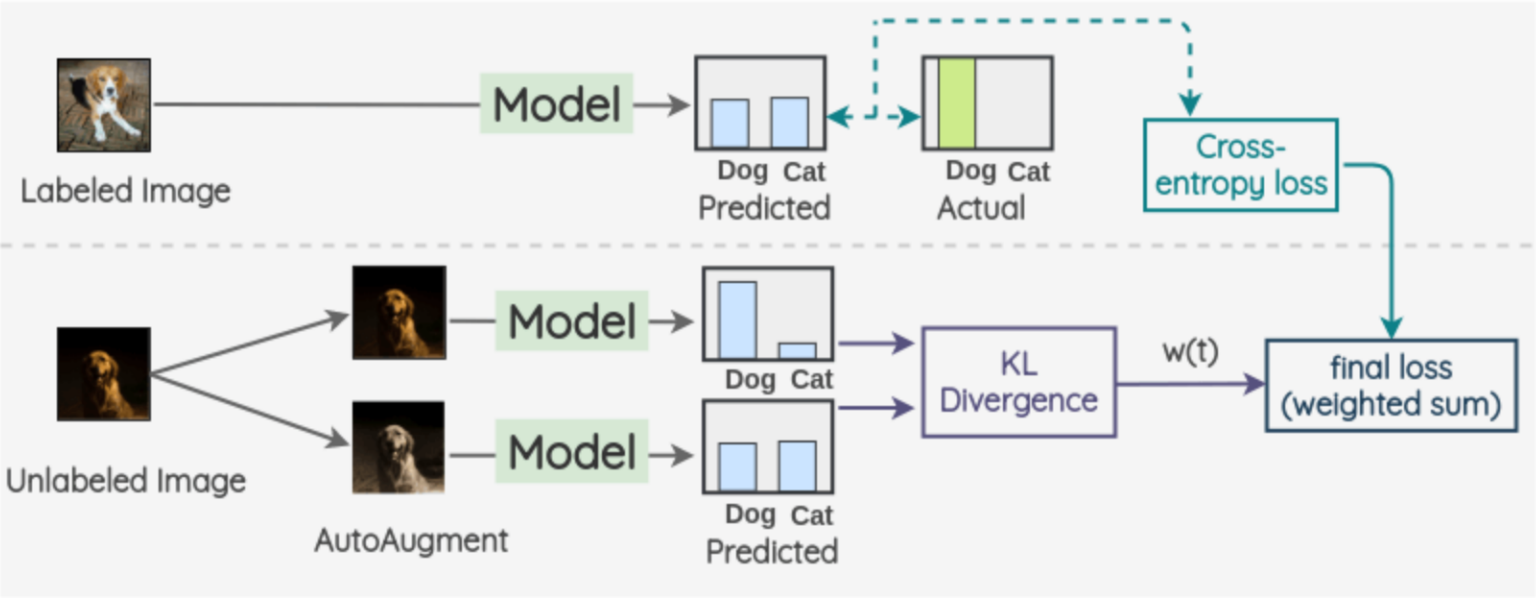
Then we proceed as previous methods. We take a labeled/unlabeled image as first view and take its adversarial example generated in previous step as the second view. Then, the same model is used to predict label distributions for both images. The KL-divergence of these two predictions is used as a consistency loss. For labeled images, we also calculate the cross-entropy loss. The final loss is a weighted sum of these two loss terms. A weight α is applied to decide how much the consistency loss contributes in the overall loss.



Semi-supervised Learning in Computer Vision

Unsupervised Data Augmentation

The key idea is to create an augmented version of a unlabeled image using AutoAugment. Then, a same model is used to predict the label of both these images. The KL-divergence of these two predictions is used as a consistency loss. For labeled images, we only calculate the cross-entropy loss and don't calculate any consistency loss. The final loss is a weighted sum of these two loss terms. A weight w(t) is applied to decide how much the consistency loss contributes in the overall loss.

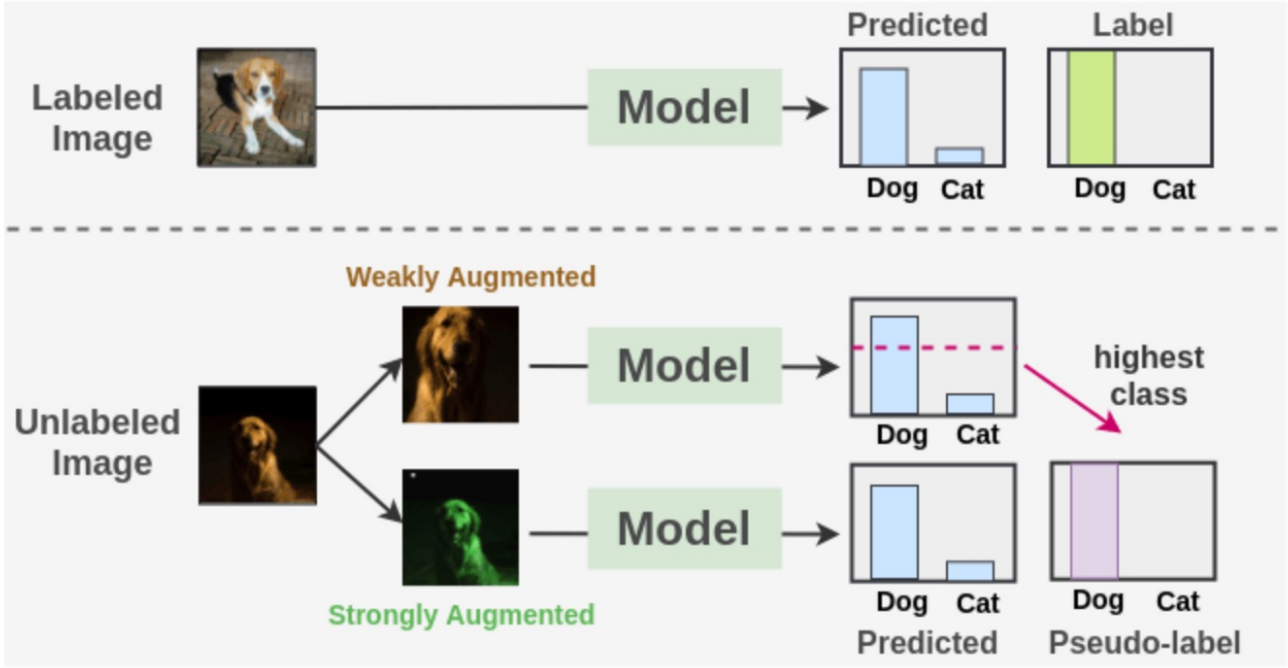


Semi-supervised Learning in Computer Vision

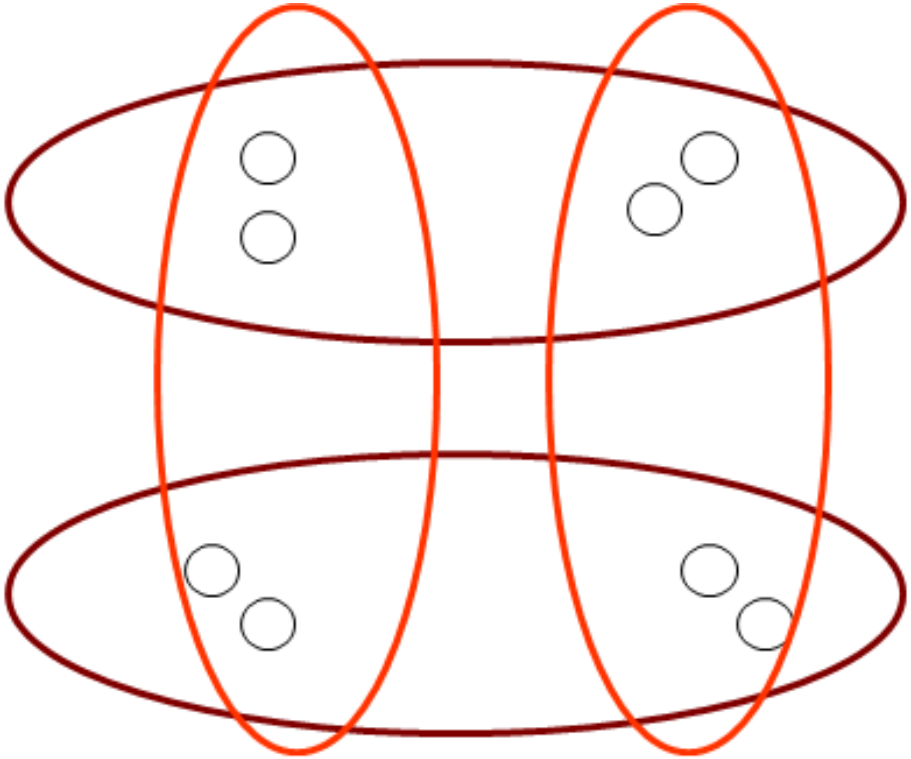
Hybrid - FixMatch

This method combines pseudo-labeling and consistency regularization while vastly simplifying the overall method. It got state of the art results on a wide range of benchmarks.

We train a supervised model on our labeled images with cross-entropy loss. For each unlabeled image, weak augmentation and strong augmentations are applied to get two images. The weakly augmented image is passed to our model and we get prediction over classes. The probability for the most confident class is compared to a threshold. If it is above the threshold, then we take that class as the ground label i.e. pseudo-label. Then, the strongly augmented image is passed through our model to get a prediction over classes. This prediction is compared to ground truth pseudo-label using cross-entropy loss. Both the losses are combined and the model is optimized.

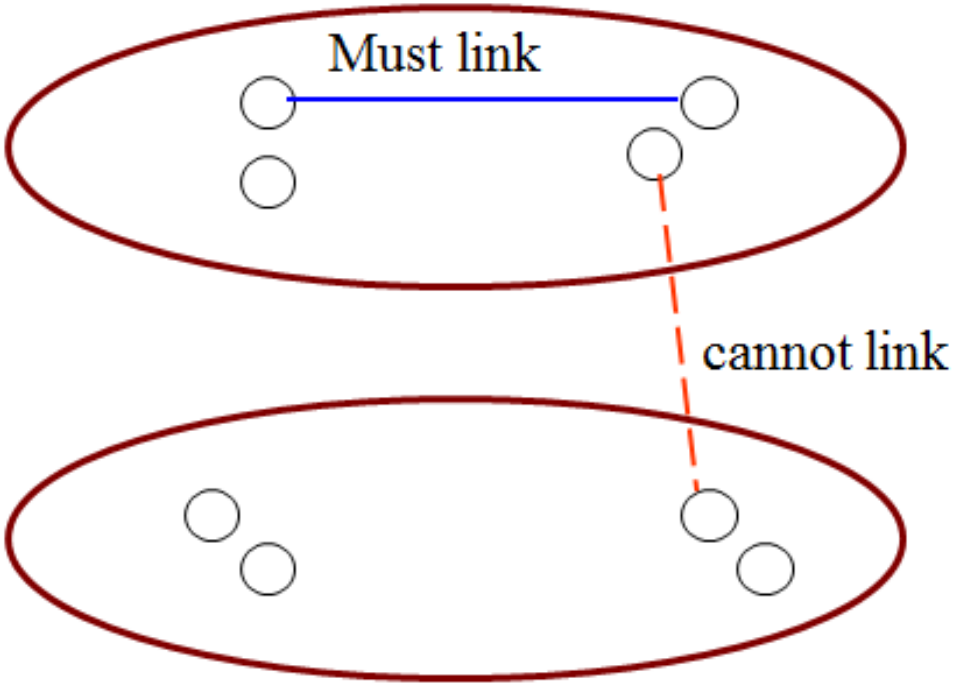


Semi-Supervised Clustering



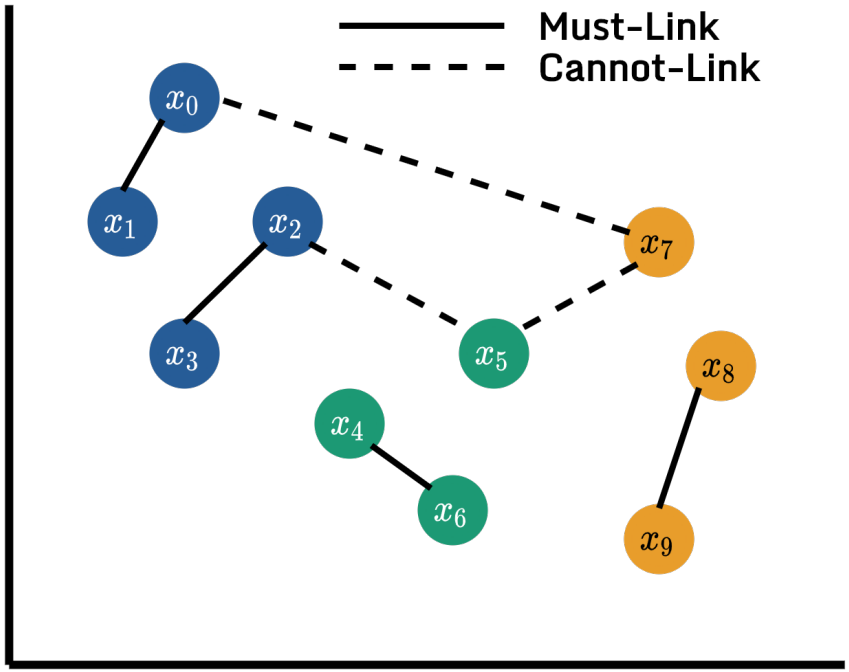
- Clustering data into two clusters

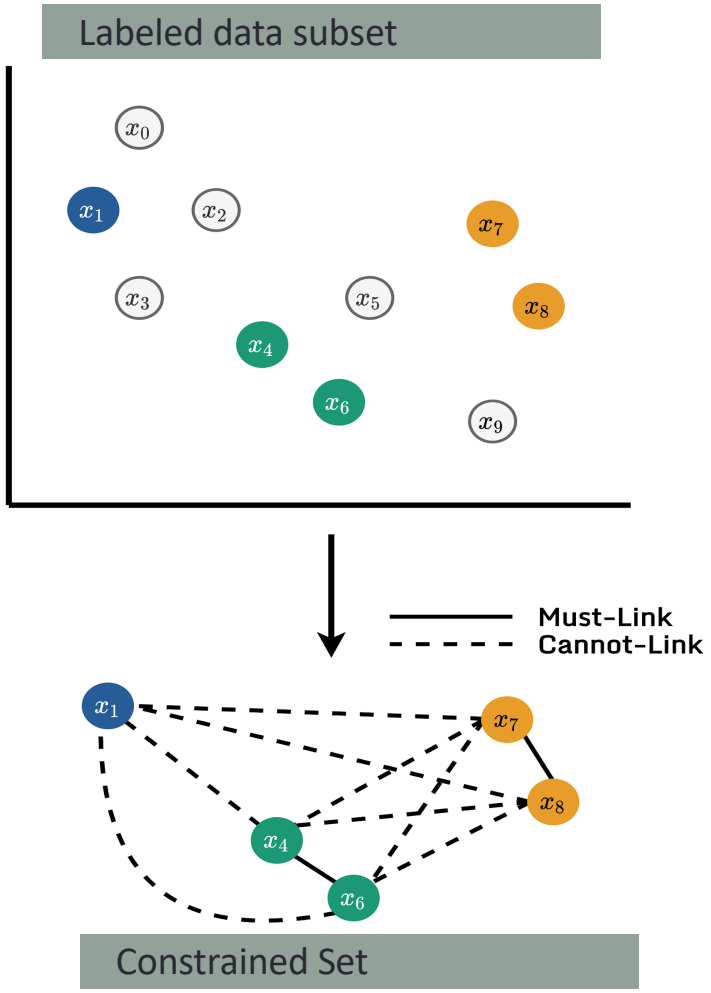
Semi-Supervised Clustering



- Clustering data into two clusters
- Side information:
  - Must links vs. cannot links

Constrained Clustering





Semi-Supervised Clustering

- Also called constrained clustering
- Two types of approaches
  - Restricted data partitions
  - Distance metric learning approaches

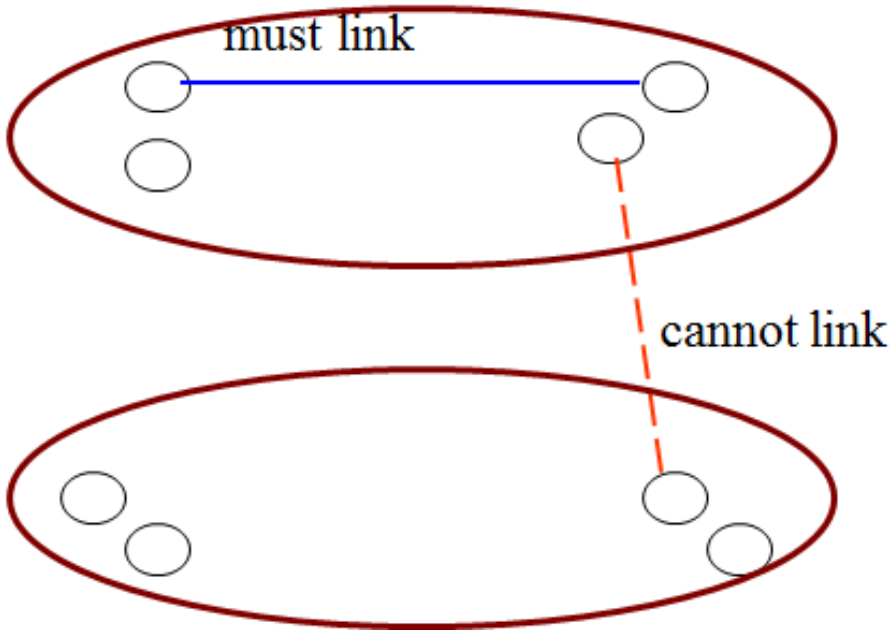
Restricted Data Partitions

- Require data partitions to be consistent with the given links
- Links → hard constraints
  - E.g. constrained K-Means (Wagstaff et al., 2001)
- Links → soft constraints
  - E.g., Metric Pairwise Constraints K-means (Basu et al., 2004)

Restricted Data Partitions

- Hard constraints
  - Cluster memberships must obey the link constraints

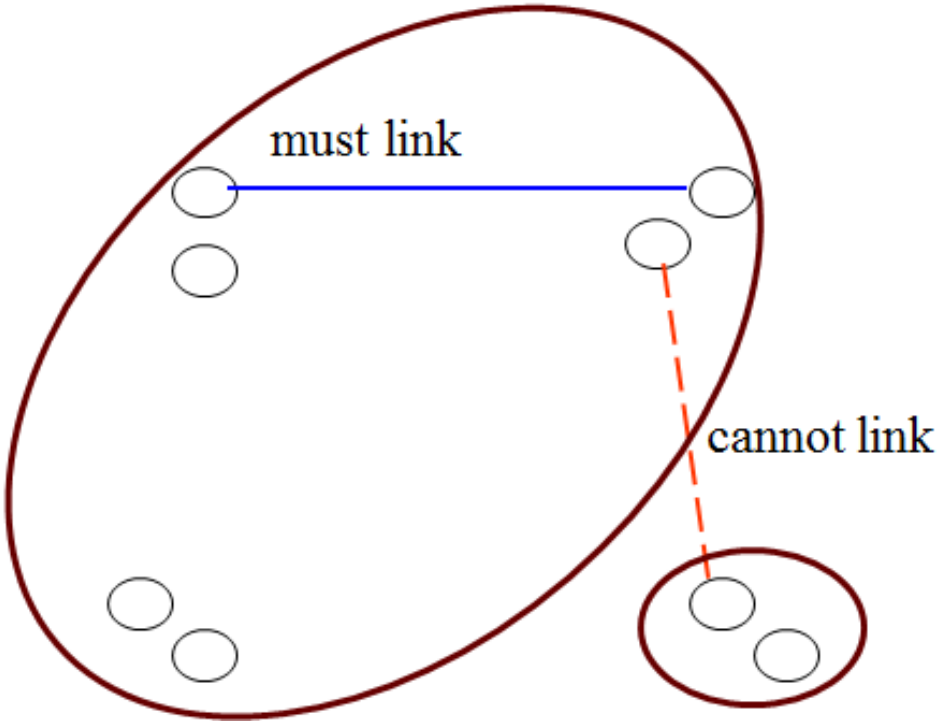
Yes



Restricted Data Partitions

- Hard constraints
  - Cluster memberships must obey the link constraints

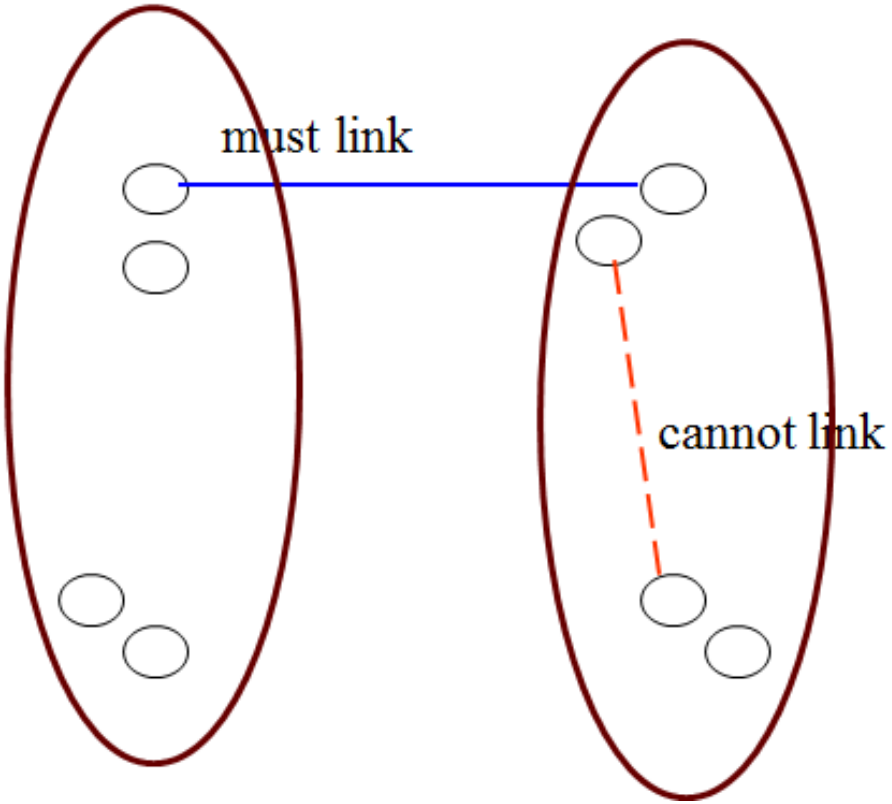
Yes



Restricted Data Partitions

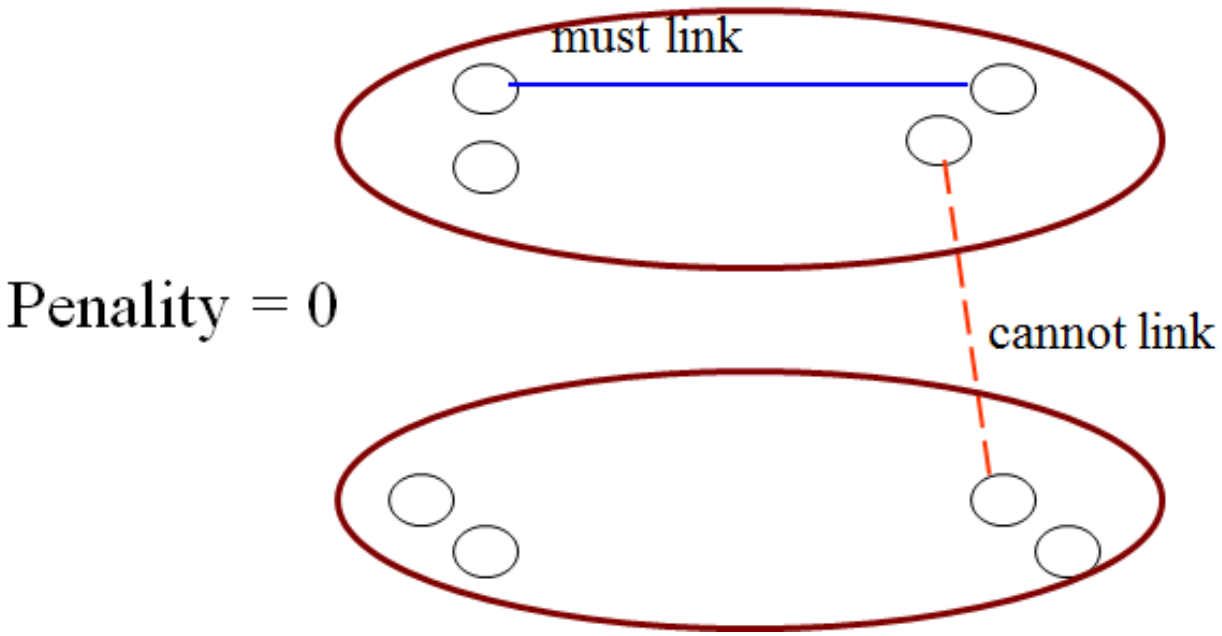
- Hard constraints
  - Cluster memberships **must** obey the link constraints

No



Restricted Data Partitions

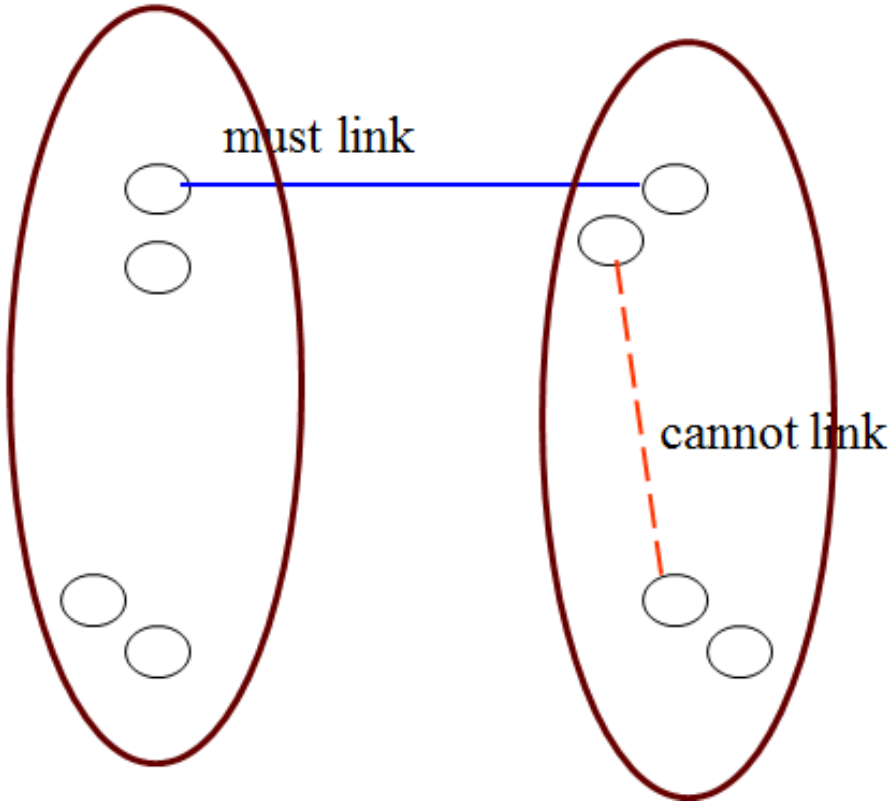
- Soft constraints
  - **Penalize** data clustering if it violates some links



Restricted Data Partitions

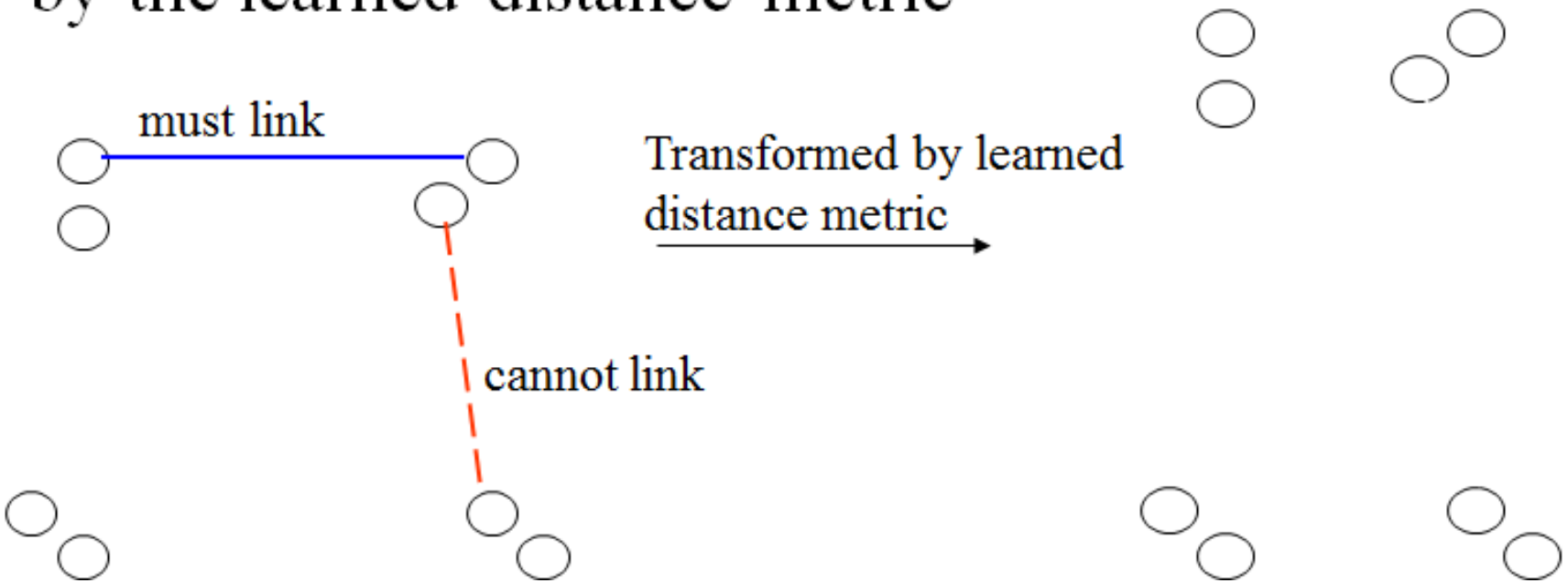
- Soft constraints
  - Cluster memberships must obey the link constraints

Penality = 1



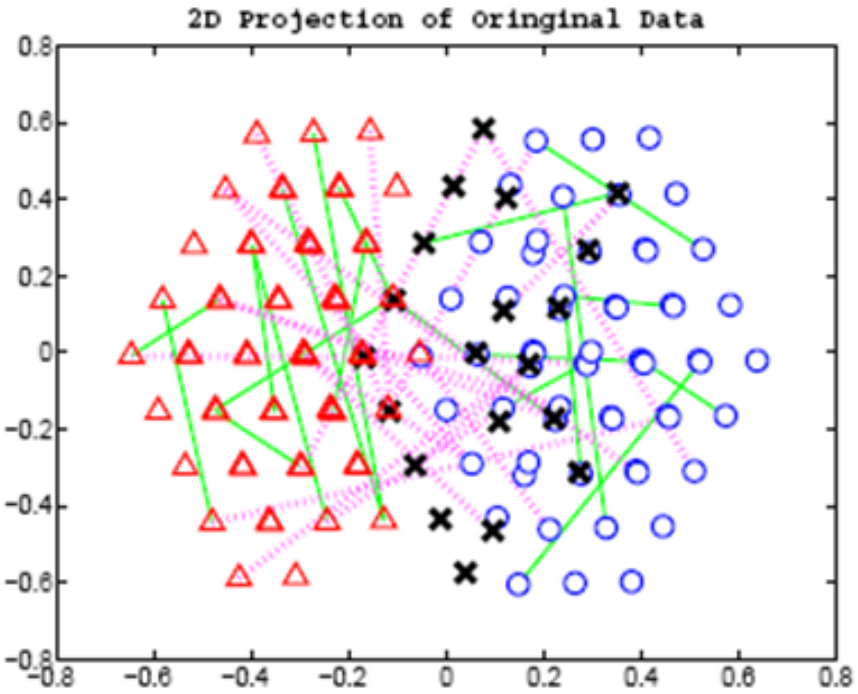
Distance Metric Learning

- Learning a distance metric from pairwise links
  - Enlarge the distance for a cannot-link
  - Shorten the distance for a must-link
- Applied K-means with pairwise distance measured by the learned distance metric

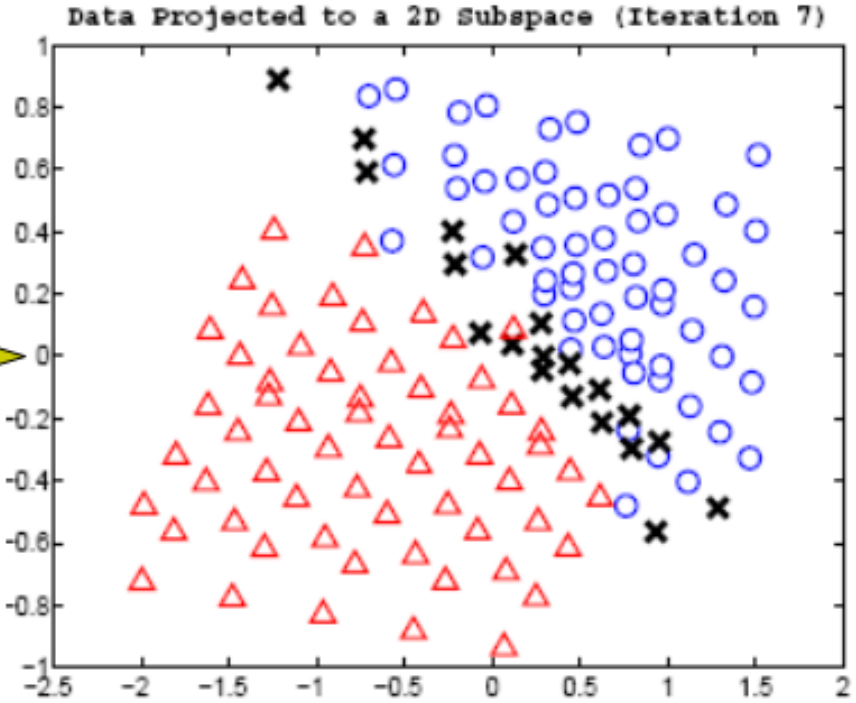


Distance Metric Learning

2D data projection using Euclidean distance metric



2D data projection using learned distance metric



Solid lines: must links

dotted lines: cannot links

INTRODUCTION

Distance Metric Learning (DML) is an area of machine learning with the purpose of learning distances from the data.

Definition

Let X be a non-empty set. A ***distance*** on X is a map d: $X \times X \to \mathbb{R}$ verifying:

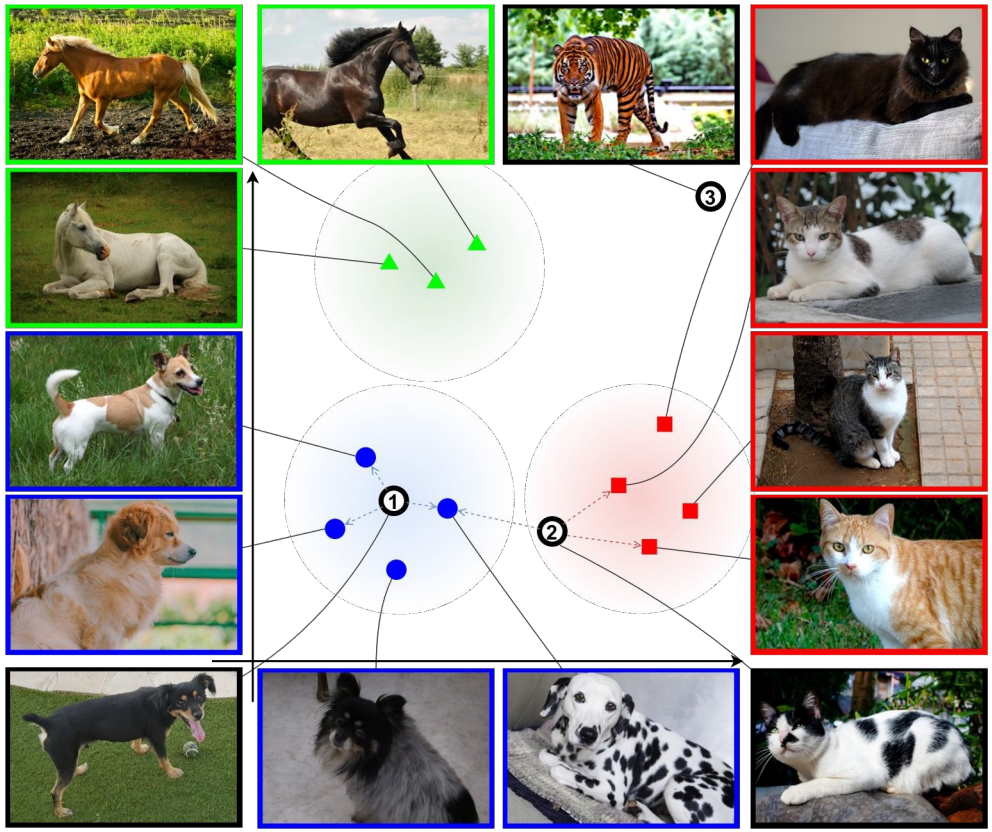
1. $d(x,y) = 0 \Leftrightarrow x = y$ for any $x, y \in X$ (coincidence)
2. $d(x,y) = d(y,x)$ for any $x, y \in X$ (symmetry)
3. $d(x,z) \leq d(x,y) + d(y,z)$ for any $x, y, z \in X$ (triangle inequality).

➢ d is said to be a ***pseudodistance*** if $d(x,x) = 0$ instead of 1.

EXAMPLE: THE k-NEAREST NEIGHBORS CLASSIFIER

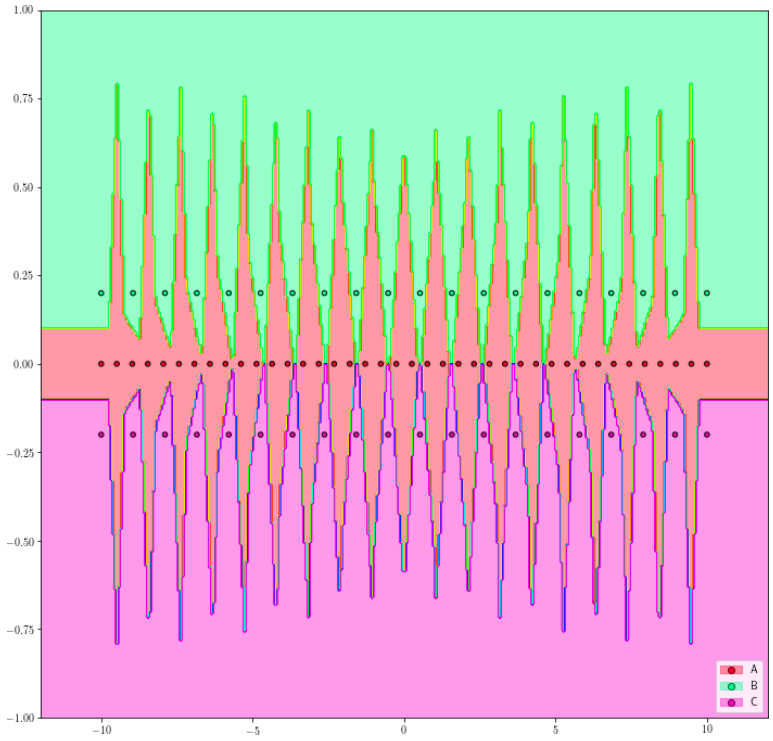
Problem: similarity-based classifiers usually use standard distances (Euclidean, etc.).

Solution: learn a distance from the data themselves.

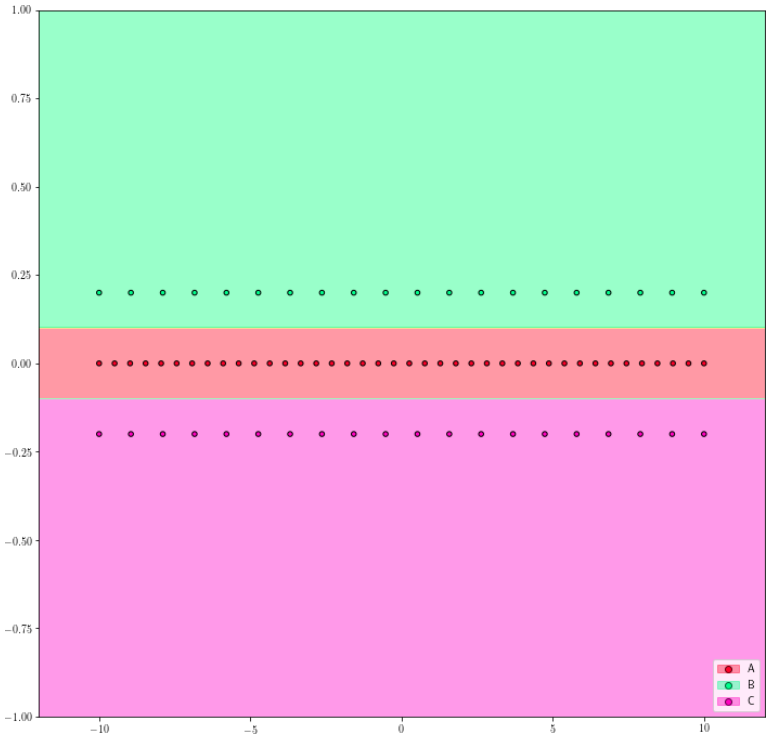


EXAMPLE

Standard distance (Euclidean)



After learning a distance



HOW TO LEARN A DISTANCE?

1. **Linear approaches**.

Let $M \in \mathcal{M}_d(\mathbb{R})$ be positive semidefinite. Then, d_M: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, given by

$$d_M(x, y) = \sqrt{(x-y)^T M (x-y)}$$

is a (pseudo-)distance, called ***Mahalanobis distance.***

Traditional approach: Learn Mahalanobis distances on d-dimensional euclidean spaces.

MAHALANOBIS DISTANCES

➢**Two ways to learn a Mahalanobis distance:**

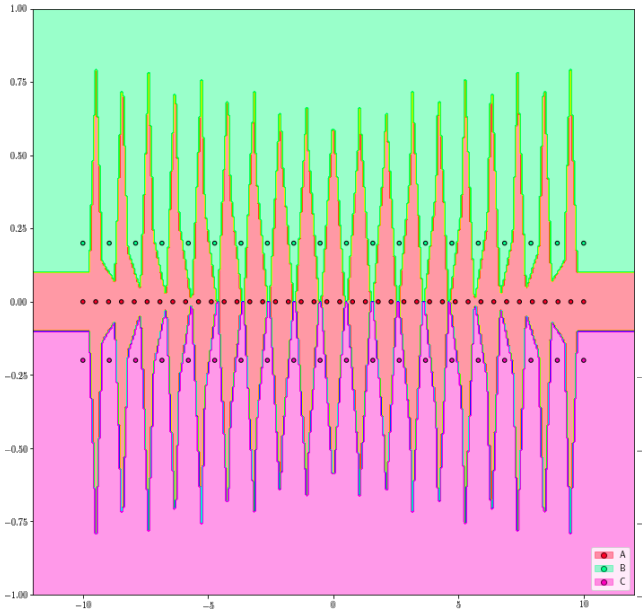
- Learn M.
- Learn a linear map (embedding) L.
  Then, $M = L^T L$ and $d_M(x, y)^2 = \|L(x - y)\|_2^2$.

➢**Optimization: loss function** $\ell(d, S, D, R)$

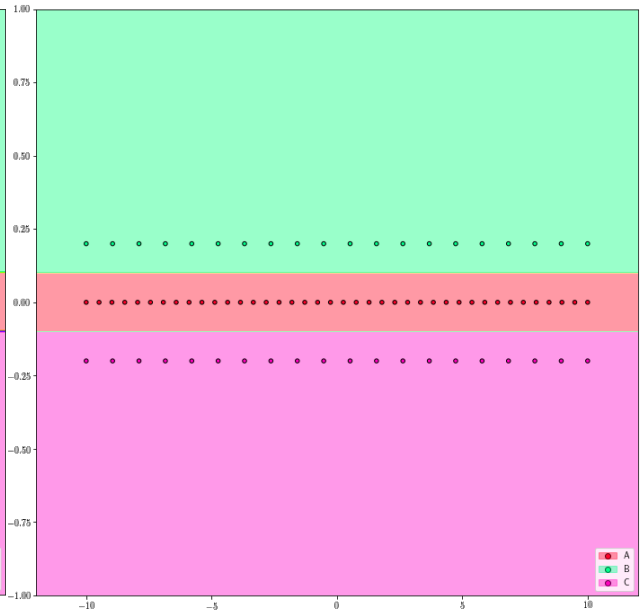
- d: distance to optimize ($d \cong M$ or $d \cong L$).
- S: similar data pairs should be close.
- D: dissimilar data pairs should be far apart.
- R: any sample should be closer to a similar sample than a disimilar one.

MAHALANOBIS DISTANCES

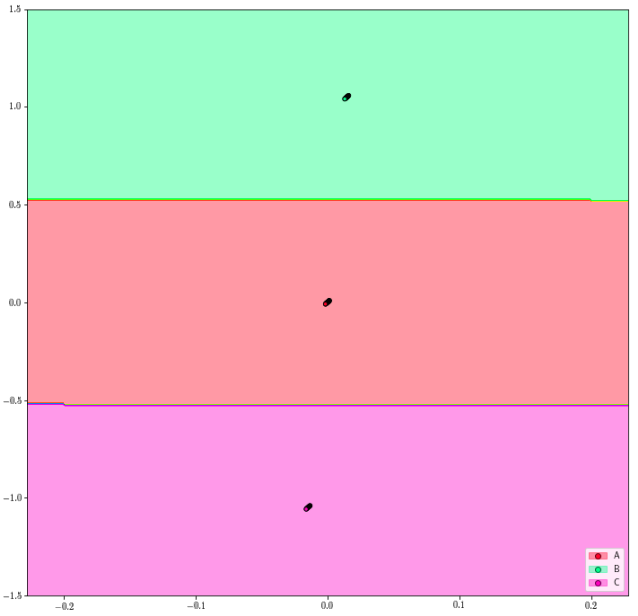
$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



$M \approx \begin{pmatrix} 0 & -0.004 \\ -0.004 & 27.5 \end{pmatrix}$



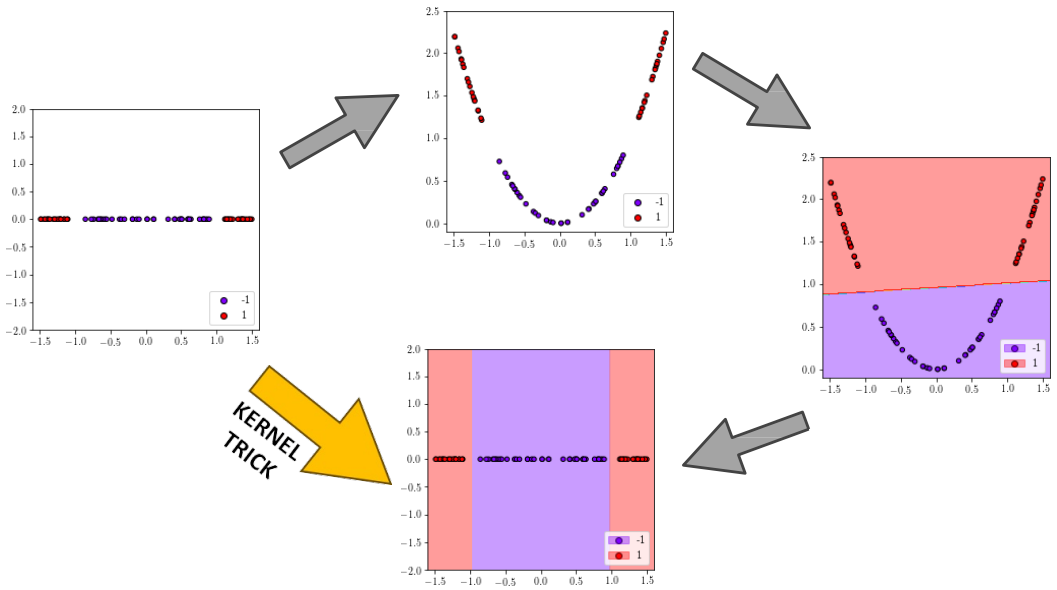
$L \approx \begin{pmatrix} -0.0001 & 0.073 \\ -0.0008 & 5.24 \end{pmatrix}$



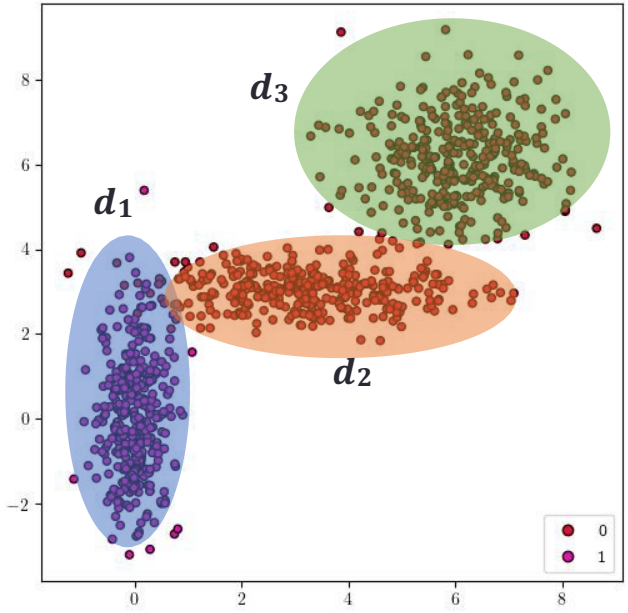
HOW TO LEARN A DISTANCE?

2. Non-linear approaches:

- **Kernel distance metric learning.**

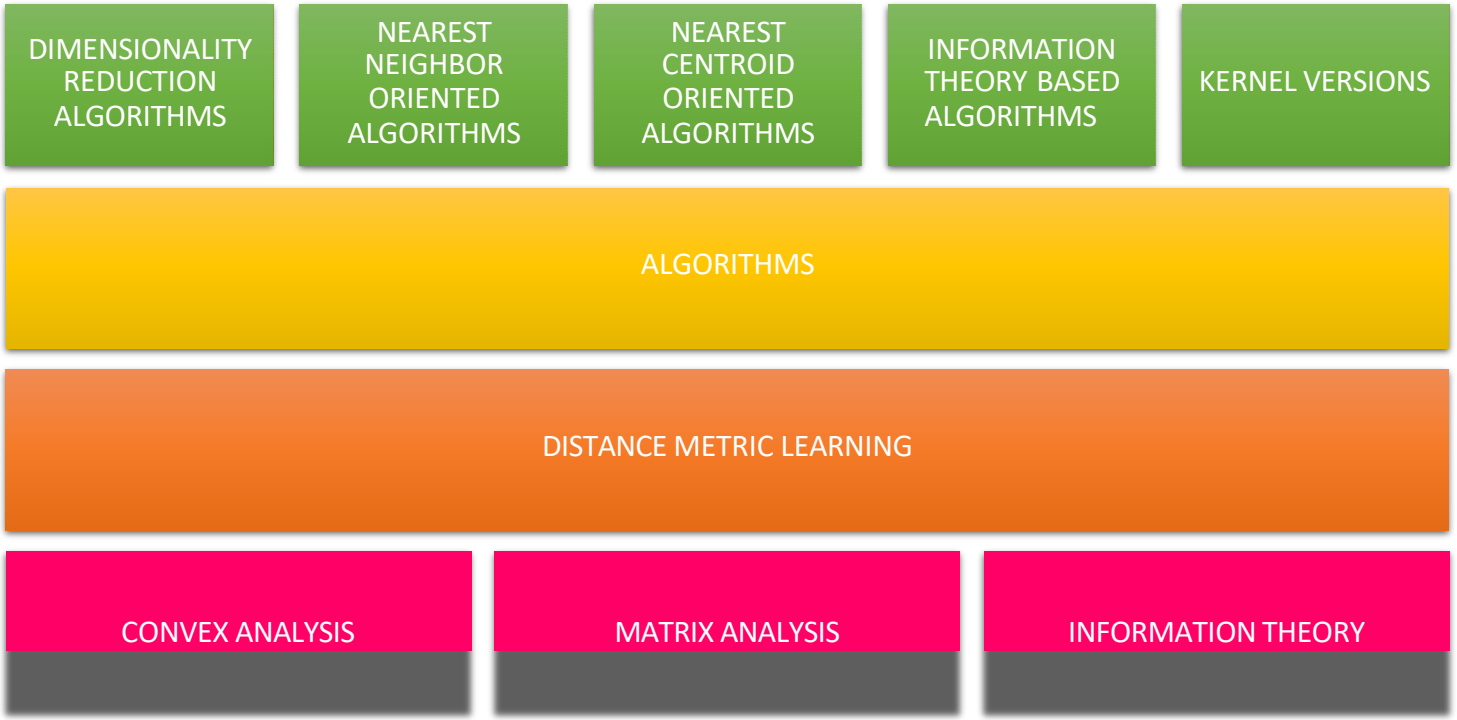


- **Local distance metric learning.**

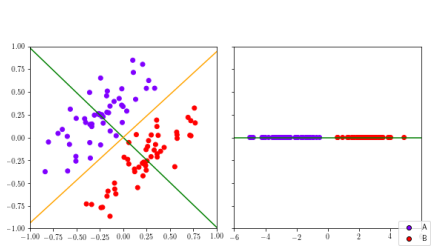


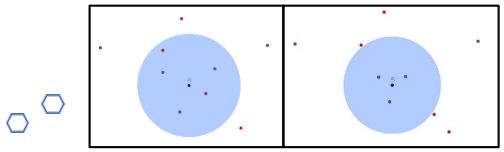
- **Deep metric learning**

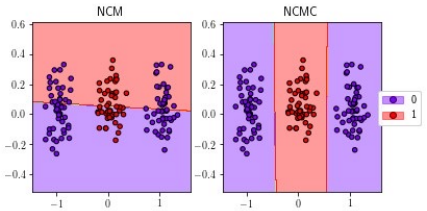
FOUNDATIONS

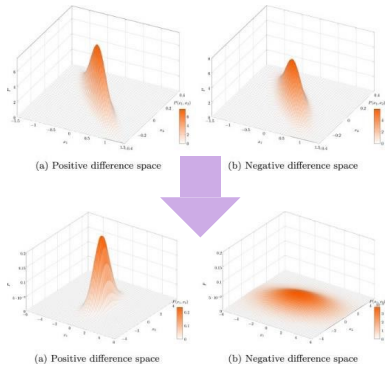


ALGORITHM CLASSIFICATION









Unsupervised

LDA
LMNN
PCA
NCA
NCMC
ANMM
MCML
LSI
DMLMJ
NCMML
ITML
KLMNN
DML-eig
KANMM
KDA
LDML
KDMLMJ

Weak-supervised

Supervised

Dimensionality reduction

Nearest neighbor classification

Nearest centroids classification

Information theory based

Kernel versions

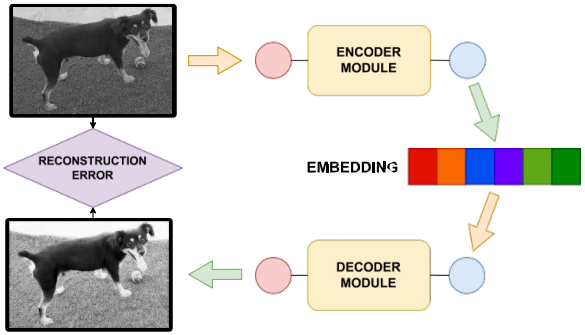
DEEP METRIC LEARNING

- Deep learning models can also be trained to learn distances.
- Similar data should be near, dissimilar data should be far away.
- Learning to **discriminate** instead of learning to **recognize**.

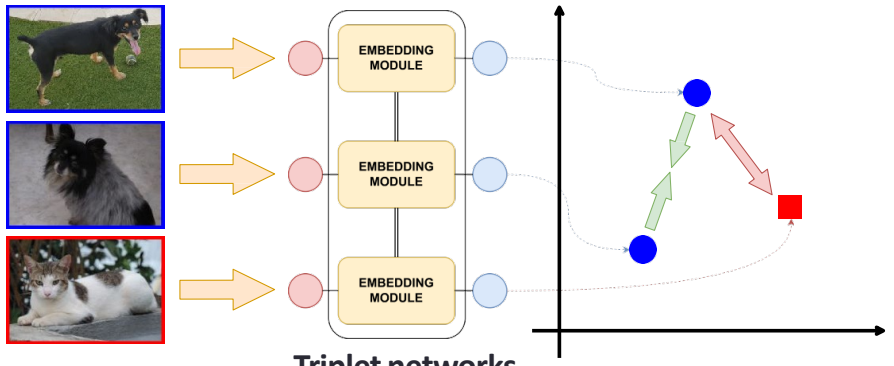
✓ Optimal projections.
✓ Models independent of the number of classes.
✓ Better generalization: few-shot, zero-shot, imbalanced data, …
✓ Aggregation with similarity-based (explainable) algorithms.

DEEP METRIC LEARNING

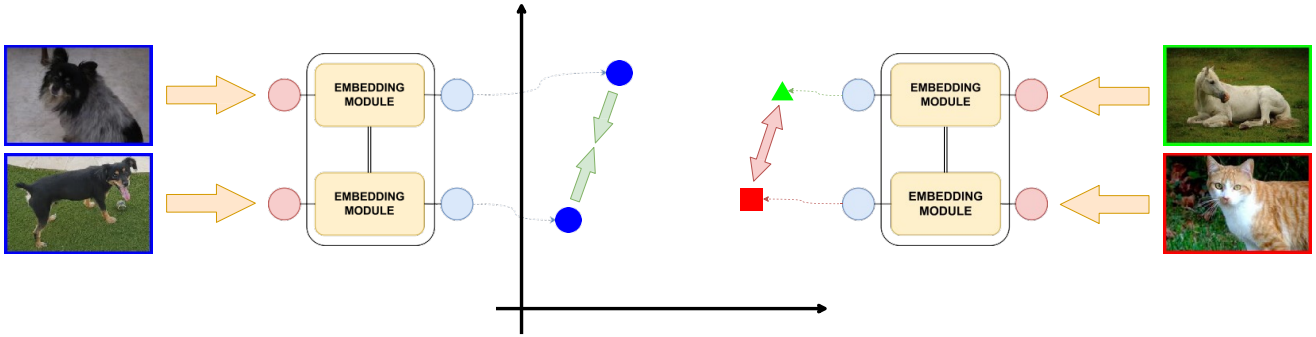
- Key factors: loss function, base architectures, mining strategies, **meta-architectures**, …



Autoencoders

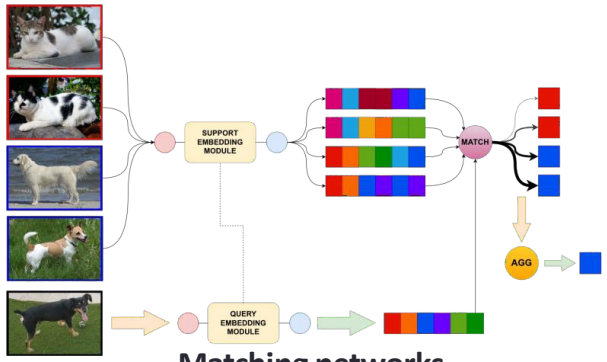


Triplet networks

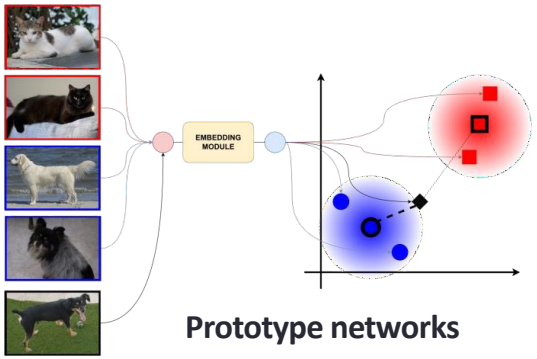


Siamese networks

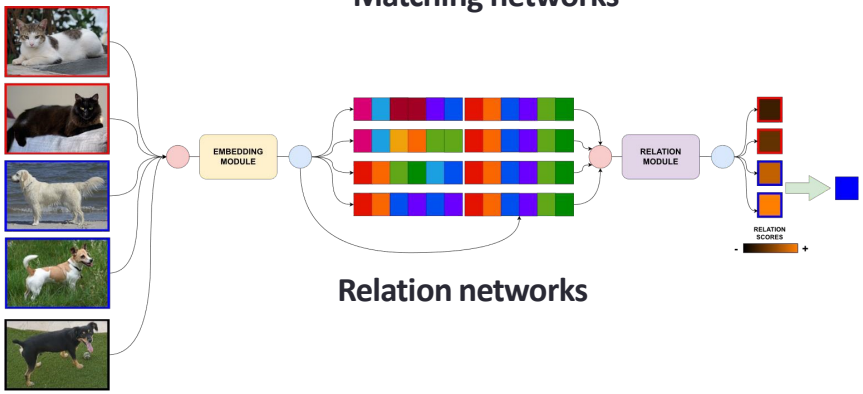
DEEP METRIC LEARNING



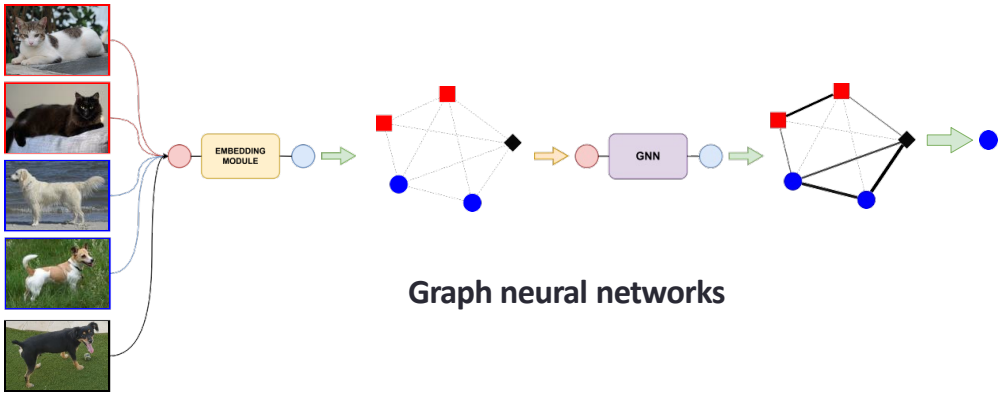
Matching networks



Prototype networks



Relation networks



Graph neural networks

Distance Metric Learning

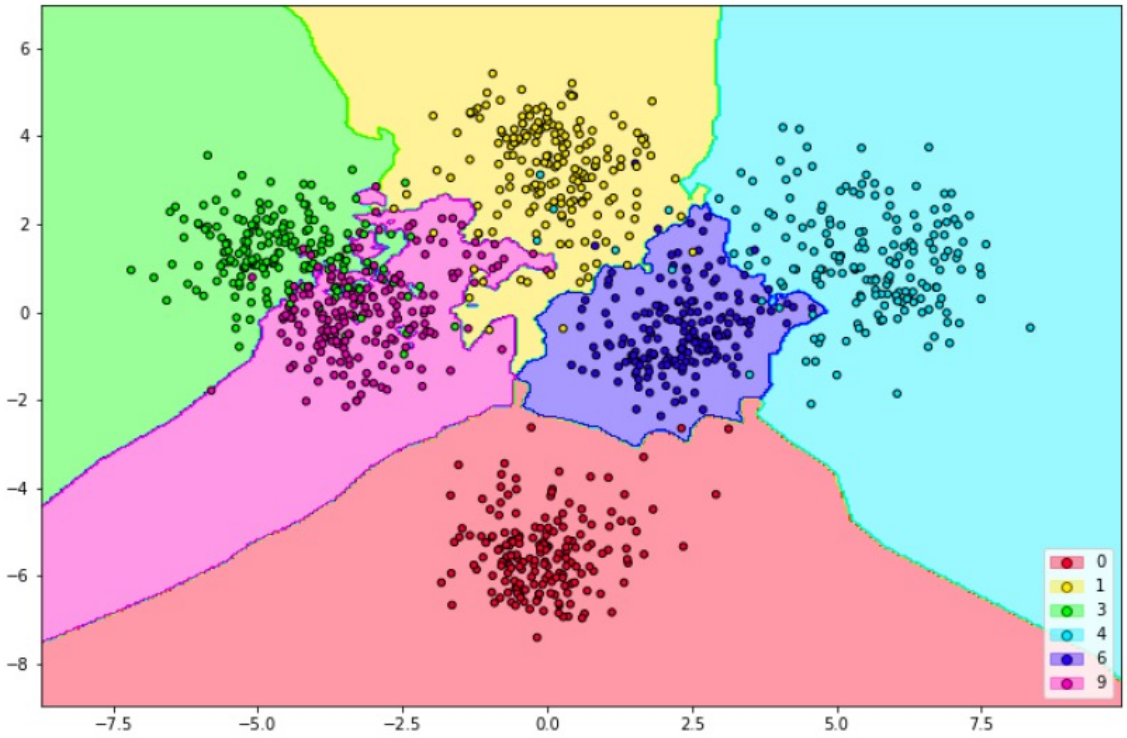


Figure 2: 'Digits' dataset consists of 1797 examples. Each of them consists of a vector with 64 attributes, representing intensity values on an 8x8 image. The examples belong to 10 different classes, each of them representing the numbers from 0 to 9. By learning an appropiate transformation we are able to project most classes on the plane, so that we perceive clearly differentiated regions associated with each of the classes.

Variaciones no estándares

Tipos de variación comunes:

- Información parcial: en la información de entrenamiento hay instancias sin etiquetar

Positive-unlabeled learning

Aprendizaje semi-supervisado

o no hay instancias de algunas clases

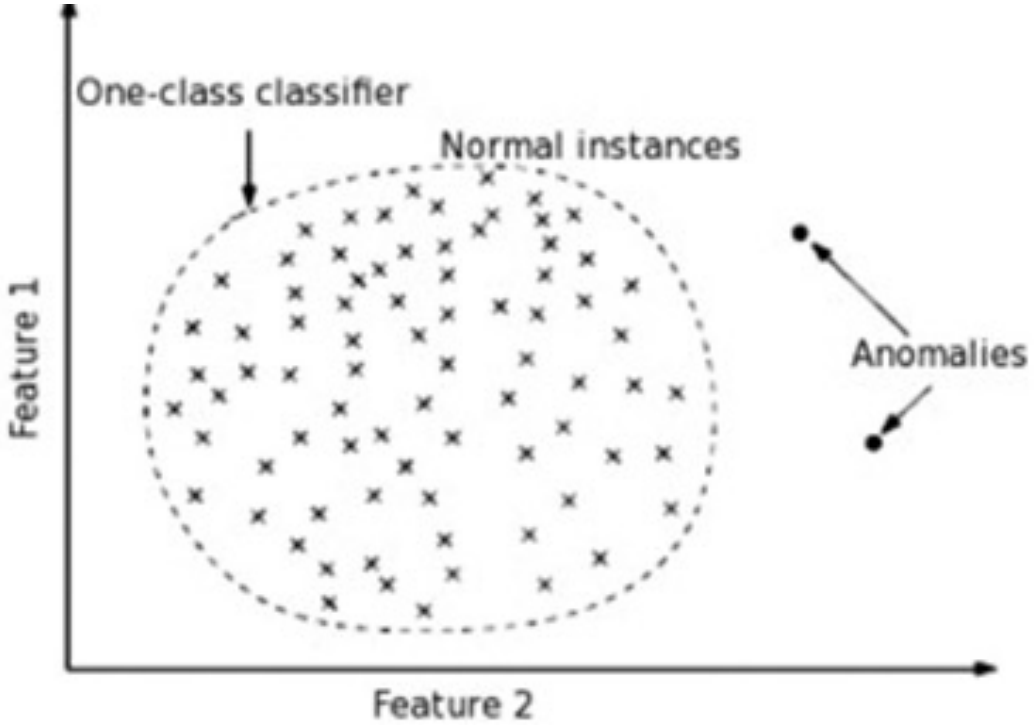
Zero-shot learning

One-class classification

One-shot learning

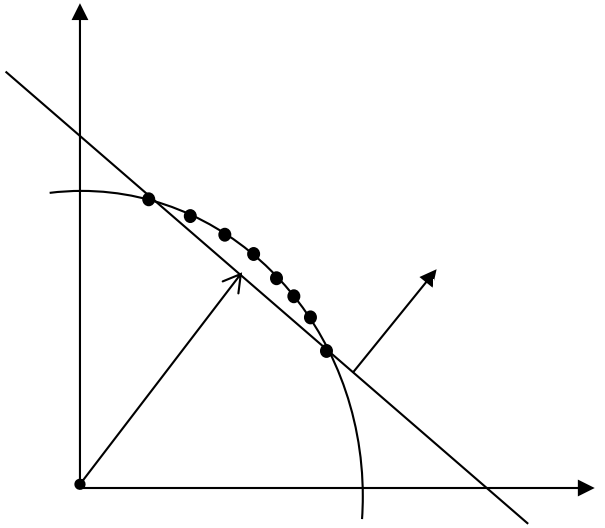
One-class (1-NN)

One-class 1-NN **Es un algoritmo semi-supervisado que aprende una función de decisión para la detección de novedades: la clasificación de nuevos datos como similares o diferentes al conjunto de entrenamiento.**

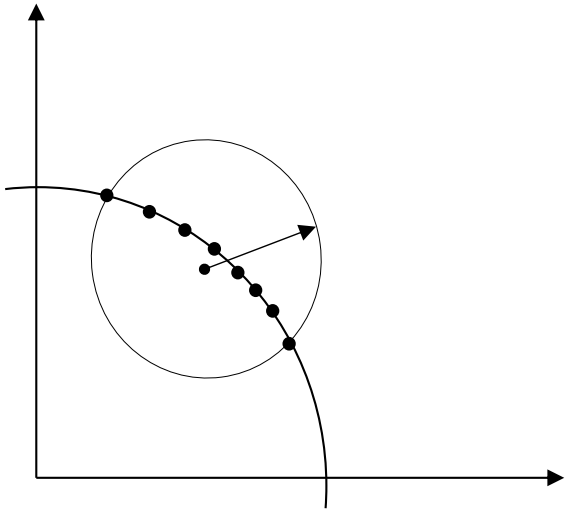


Two models for One Class Classification

- One Class SVM
  - Find the optimal ***hyperplane*** to separate the target class from the origin with maximum margin



- Support Vector Data Description
  - Use the minimum hyperspere to enclose the target class

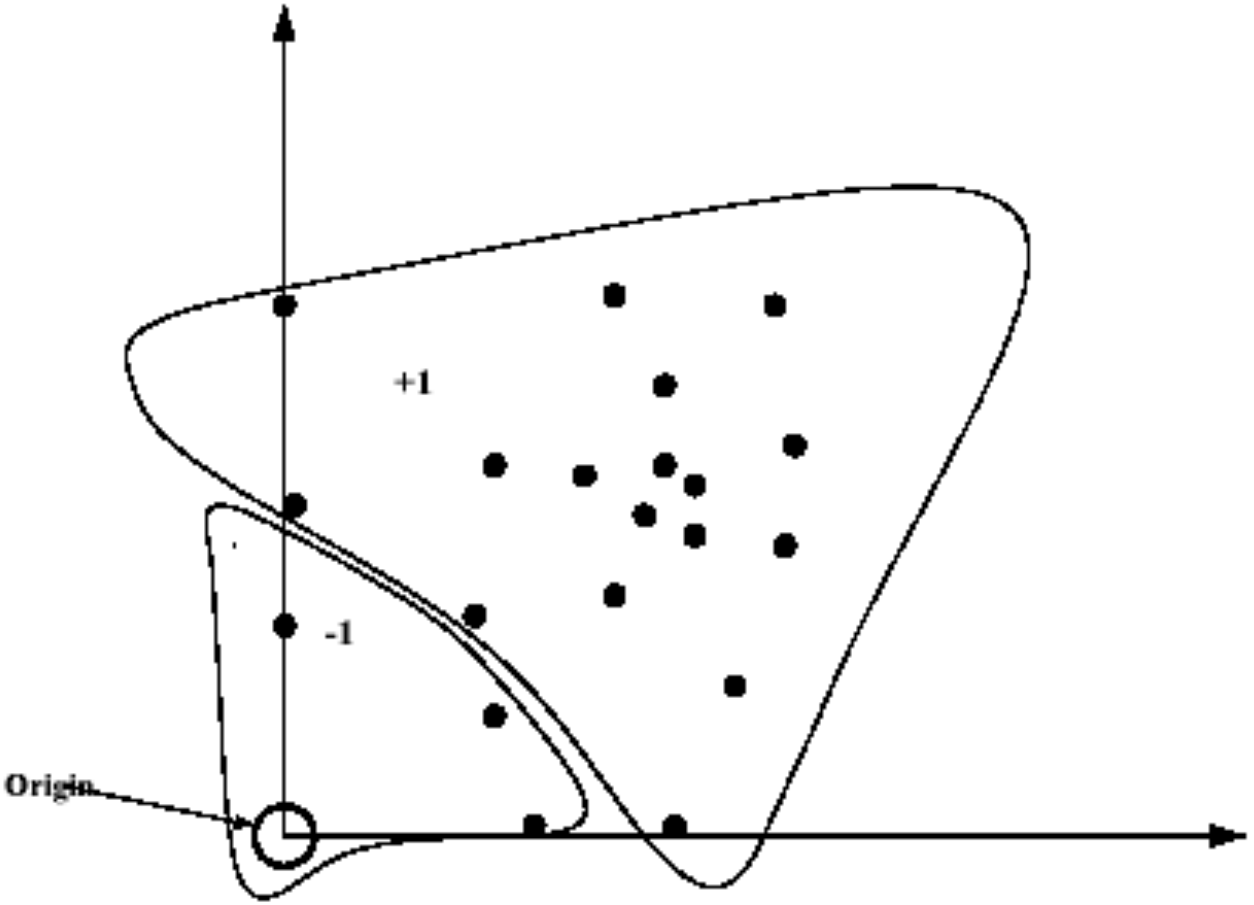


One Class Support Vector Machine (OCSVM) Algorithm

- Maps input data into a high dimensional feature space
- Iteratively finds the maximal margin in the hyperplane which best separates the training data from the origin
- Solves optimization problem to find rule f with maximal margin
  - $f(\mathbf{x})=\langle \mathbf{w},\mathbf{x} \rangle+b$
  - If $f(\mathbf{x})<0$, label **x** as anomalous

OCSVM

Figure 1: One-class SVM



One-Class SVM Classifier. The origin is the only original member of the second class.

OCSVM: Kernels

- Equivalent to solving the dual quadratic programming problem
  - $\min_{\boldsymbol{\alpha}} \bar{(}1/2) \sum_{i,j} \alpha_i \alpha_j K(x_i,x_j)$ s.t. $0 \le \alpha_i \le 1/(v\ell)$, $\sum_i \alpha_i = 1$
    - where α_i - Lagrange multiplier

      v - parameter to control trade-off between distance of hyperplane from the origin and number of points in training dataset

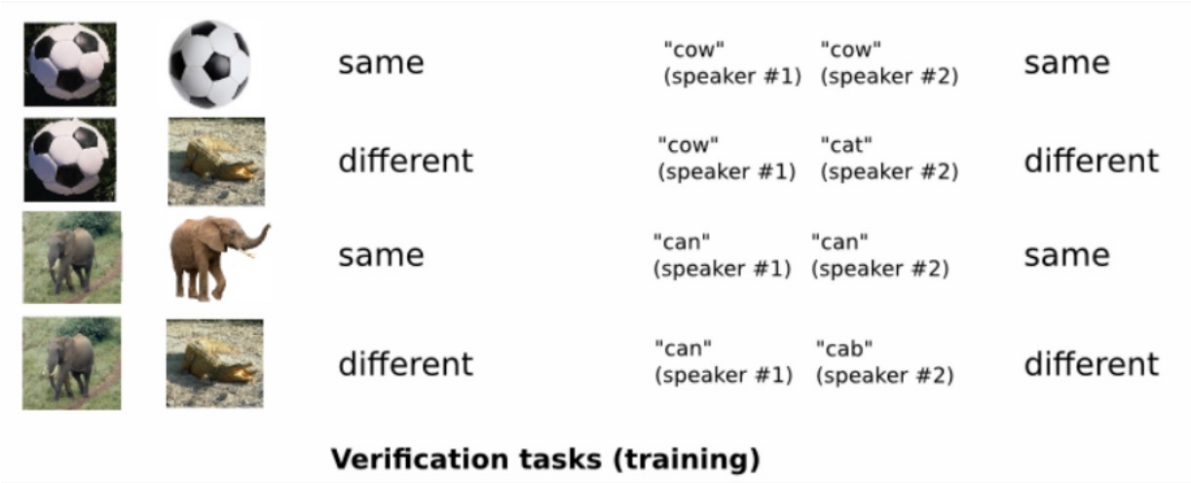
      ℓ - number of points in training dataset

- Kernel function projects input vectors into a feature space allowing for nonlinear decision boundaries
  - Feature map: $\Phi: X \rightarrow R^N$
  - Kernel Function: $K(x_i,x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$

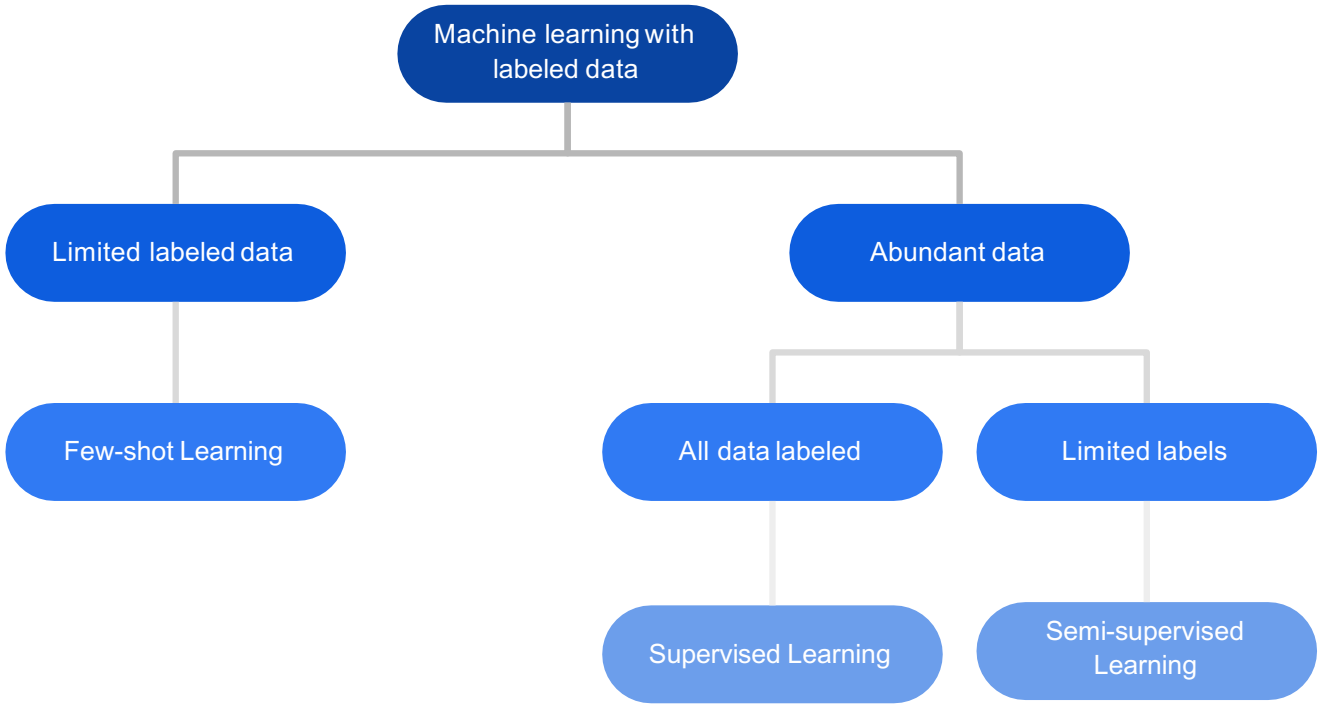
Few-Shot Learning

F-SL es una tarea de clasificación en la que se utilizan uno o unos pocos ejemplos para clasificar muchos ejemplos nuevos en el futuro.
Esto caracteriza a las tareas que se ven en el campo del reconocimiento facial, como la identificación y verificación de rostros, donde las personas deben ser clasificadas correctamente con diferentes expresiones faciales, condiciones de iluminación, accesorios y peinados dándoles una o unas pocas fotos de plantilla.
Red Siamesa: Los CNN profundos son entrenados primero para discriminar entre los ejemplos de cada clase. La idea es que los modelos aprendan vectores de características que sean eficaces para extraer características abstractas de las imágenes de entrada.

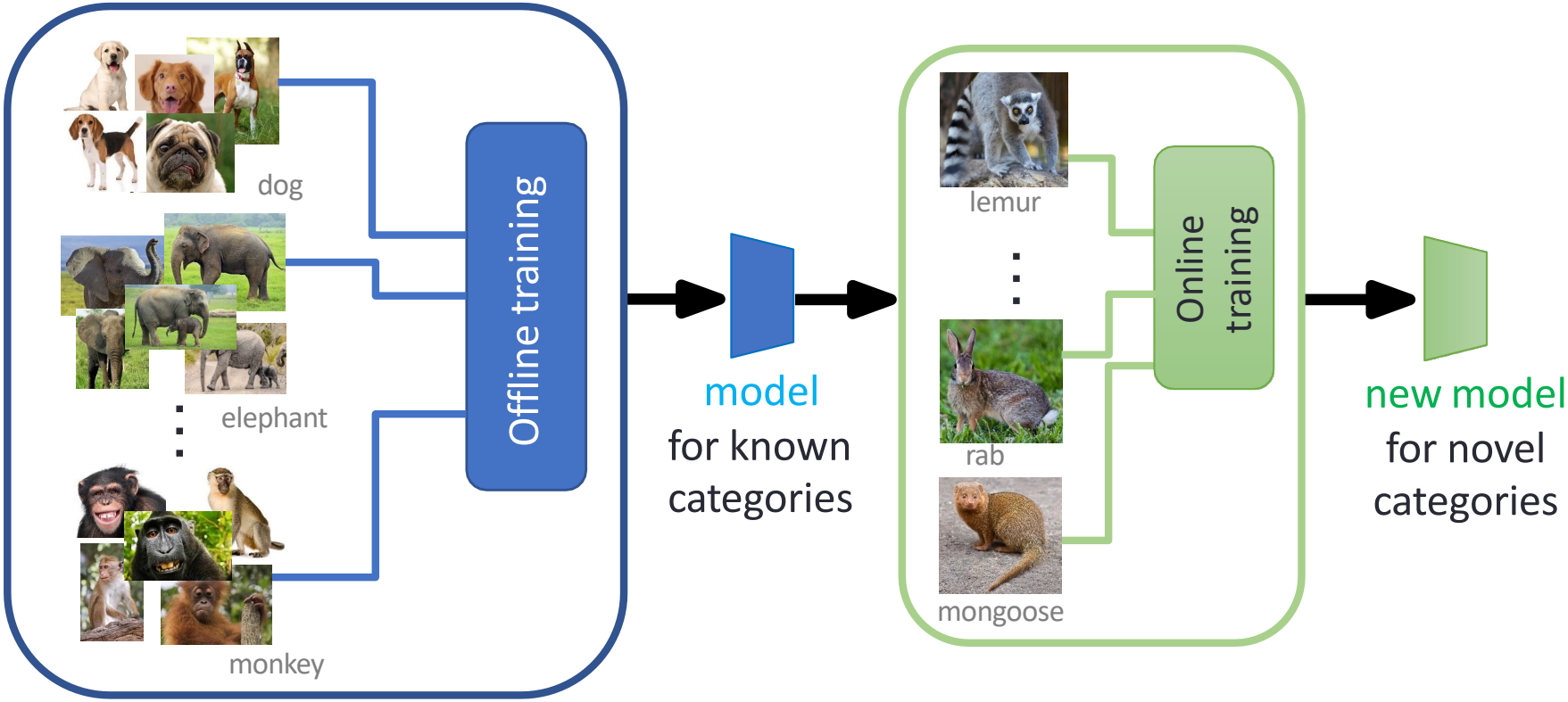
Los modelos se vuelven a proponer para su verificación con el fin de predecir si los nuevos ejemplos coinciden con una plantilla para cada clase.



Learning with labeled data



Few-shot Learning (FSL)

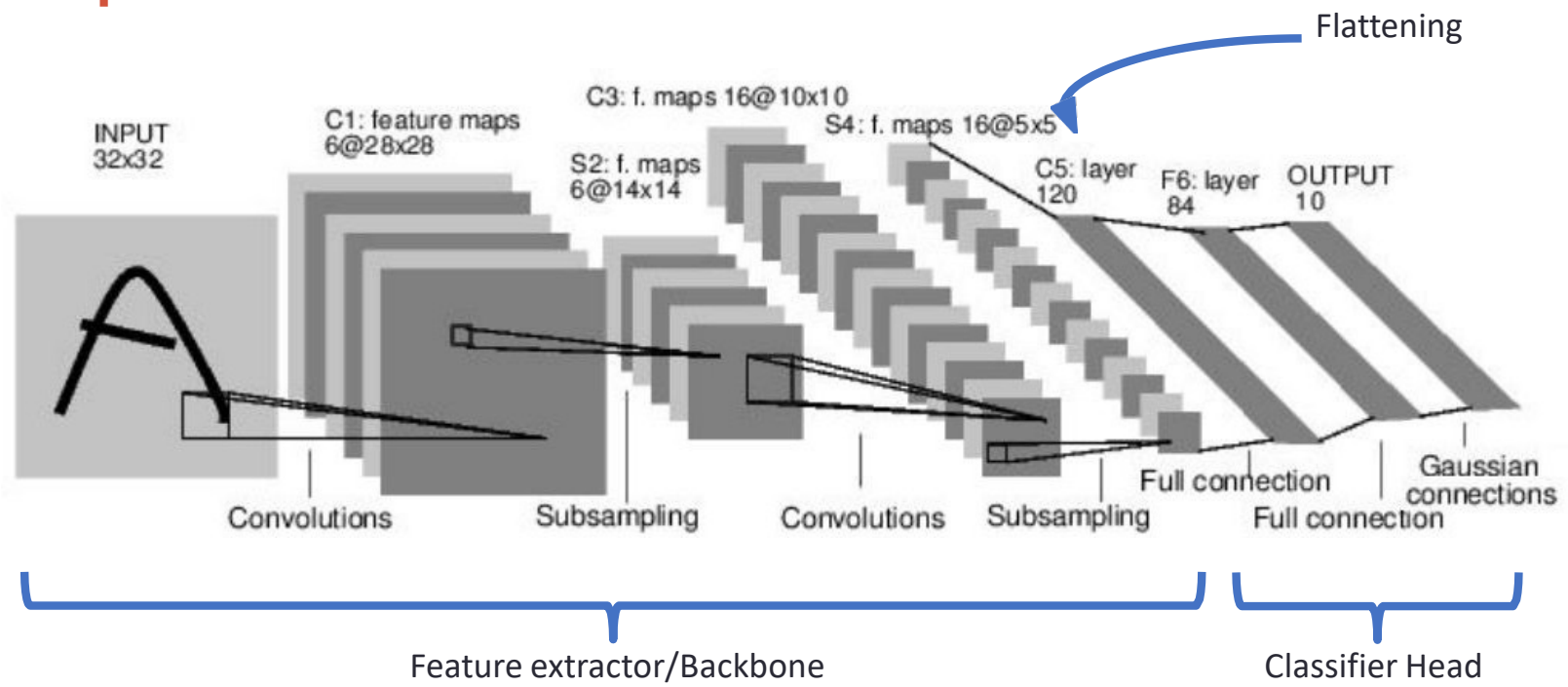


Source: Few-shot Learning, Joseph Shtok, IBM AI Research (slides)

Why care about Few-shot Learning?

- Humans **do not need** thousands of examples to learn.
- It is not always feasible to spend **much time and money** on **annotated data** for a use case.
- Objects are continuously replaced with new ones in real world use cases. Additional **time and money** is required to **retrain** the networks.

Deep Neural Networks: Quick recap



Source: LeNet, *Yann LeCun*, AT&T Bell Labs, 1980

Deep Neural Networks: Quick recap (contd.)

- Requires a huge amount of data for each task
- Same task for training and testing
- Retraining is needed for every new task

What is a task?

- Task: A tuple: $\tau = \{\mathcal{L}(.,.), D_s: (X_s, Y_s), D_q: (X_q, Y_q)\}$
  - $\mathcal{L}$ is a loss function
  - D_s is the **support set** (same as training set for regular training)
  - D_q is the **query set** (same as test set for regular training)

- Example:
  - Domain: Natural animal images
  - Task: Lion-Shark-Dog classification
  - Loss function: CE loss

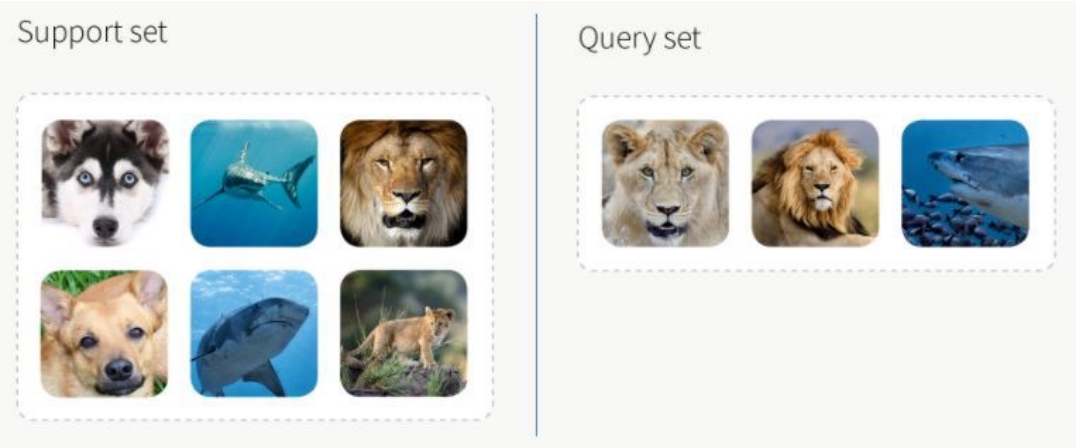
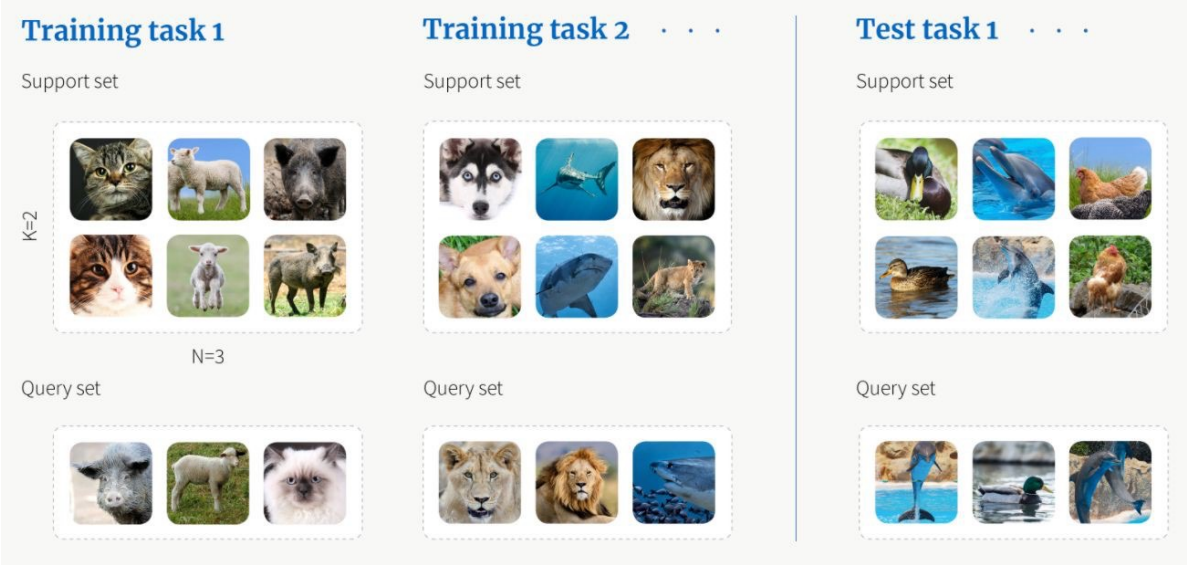


Image source: https://www.borealisai.com/en/blog/tutorial-2-few-shot-learning-and-meta-learning-i/

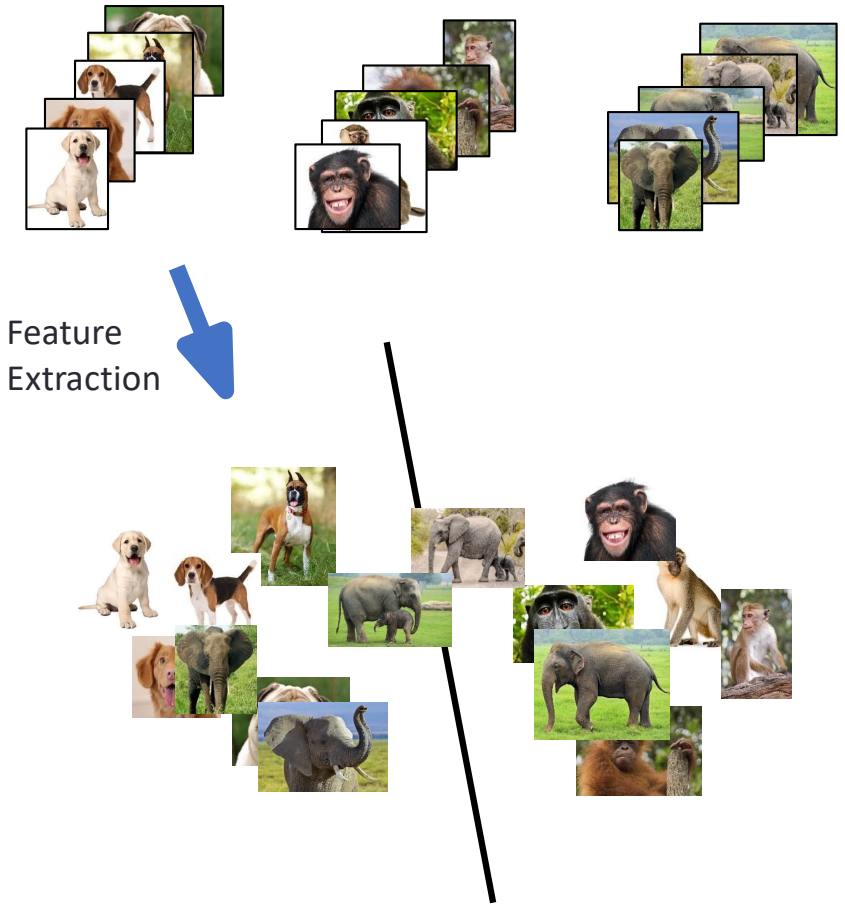
The Few-shot Learning framework

- Task distribution – the underlying space from which the tasks are drawn
  - For example: the space of all possible natural images of animals
- Training set/test set
  - can use both support and query sets within the training set to train



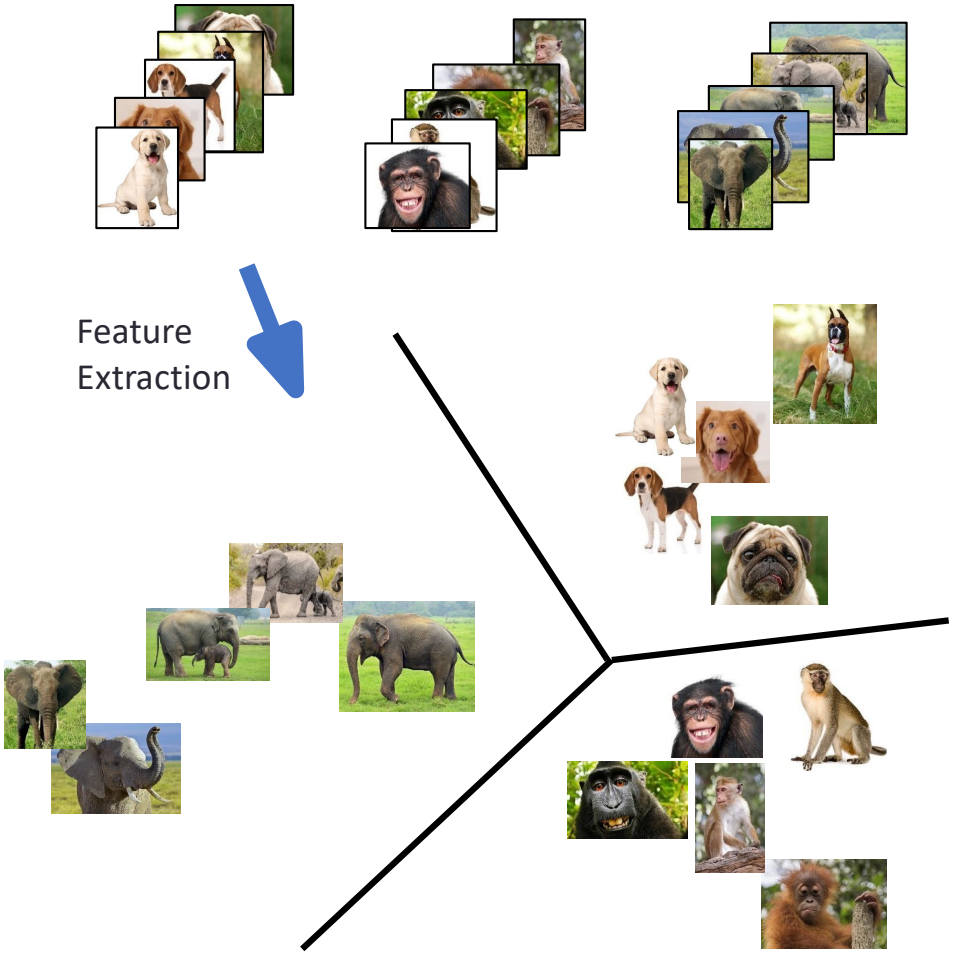
The Rationale

- If the feature extractor can already extract useful features for any new task, only the classification/ regression head needs to be completely retrained for a new task.
  - Training the feature extractor may help further improve performance.
- It is indeed possible to train a classifier/regressor with few samples, e.g. SVM/SVR.

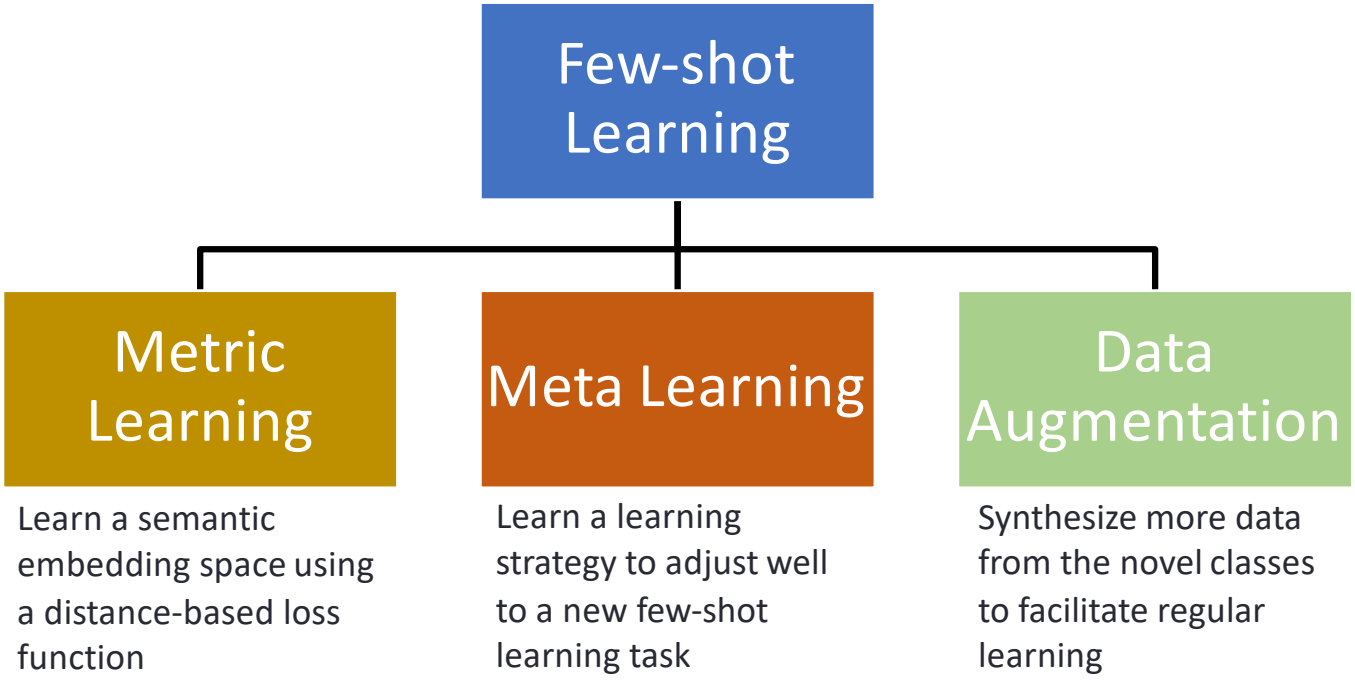


The Rationale

- If the feature extractor can already extract useful features for any new task, only the classification/ regression head needs to be completely retrained for a new task.
  - Training the feature extractor may help further improve performance.
- It is indeed possible to train a classifier/regressor with few samples, e.g. SVM/SVR.



Taxonomy of Few-shot Learning



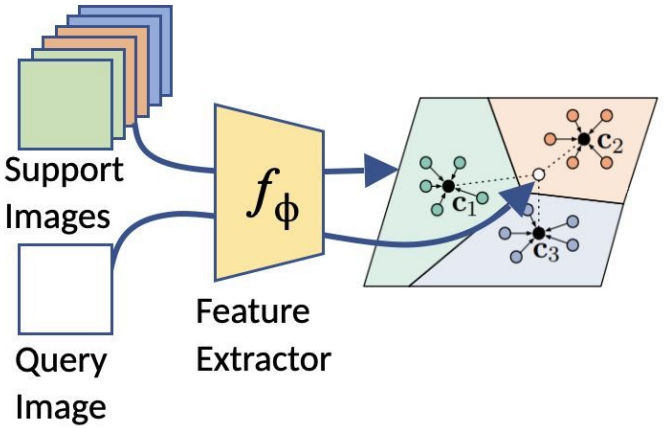
Metric Learning

"Learning to compare"

- Train the deep neural network to obtain a mapping from the high-dimensional data manifold to a low-dimensional space.

- Use a simple distance-based classification function to match the query points to the support set data.
  - The simple non-parametric classification function forces the feature extractor to learn a semantic low-dimensional embedding where similar data are mapped close to each other.

Prototypical Networks



- Prototype calculation: $c_k = \frac{1}{N_k} \sum_{i=1}^{N_k} f_\phi(x)$
- Label calculation: $p_\phi(y = k|x) = \frac{\exp(-d(x,c_k))}{\sum_K \exp(-d(x,c_i))}$
- Repeat until convergence:
  - Randomly sample a task from the task distribution
  - for each class in the support set
    - Calculate the prototype using the support samples from the class.
  - for each class in the query set
    - Update the feature extractor weights to minimize the distance of the query points to the corresponding prototype

[Prototypical Networks for Few-shot Learning, *Jake Snell, Kevin Swersky, Richard Zemel,* NIPS 2017]

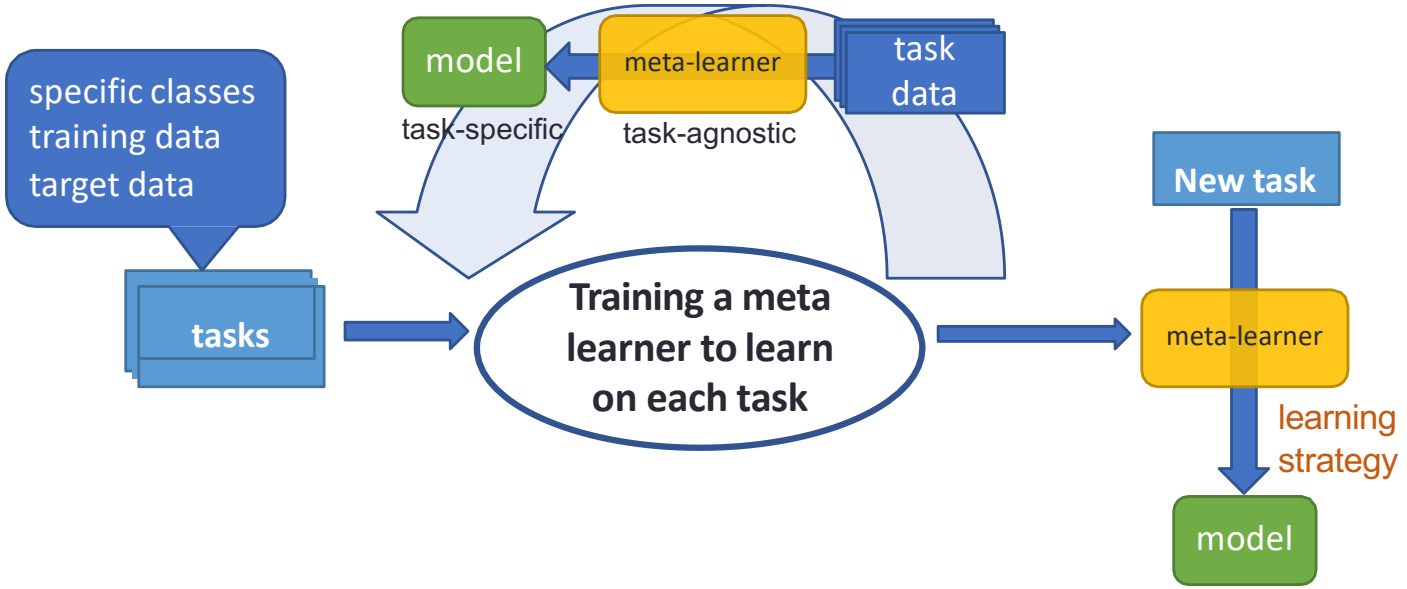
Prototypical Networks

- Other ProtoNet-like algorithms:
  - Matching Networks
    - Finds the similarity of each query point to every support point and calculates the label as a weighted combination of the labels of the support points
    - Uses cosine similarity
    - [Matching Networks for One Shot Learning, *Vinyals et al.*, NIPS 2016]
  - Relation Networks
    - Also finds the similarities between a query point and all support points
    - Finds the similarity between using a learned fully connected layer instead of a predefined distance/similarity measure
    - [Learning to Compare: Relation Network for Few-Shot Learning, *Sung et al.*, CVPR 2018]

Meta Learning

"Learning to learn"

- Learn from other tasks how to effectively learn for a target task.



Source: Few-shot Learning, *Joseph Shtok*, IBM AI Research (slides)

Model Agnostic Meta Learning (MAML)

- Learn parameters that easily transfer to novel tasks via fine-tuning on a few labeled examples.
  - Fine-tune on the support set, then attempt to minimize the loss on the query set at the parameters found using the support set.

- This forces the network to learn a feature embedding which is highly-transferrable across tasks so that the classifier can quickly adapt to any novel task.

[Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, *Finn et al.*, ICML 2017]

Model Agnostic Meta Learning (MAML)

- Start with random meta parameters θ and repeat until convergence:
  - Sample a batch of T tasks
  - For each task T_i in batch:
    - Fine-tune θ on the support set for k steps to obtain θ_{ik} (**inner loop**)
  - Update θ based on gradients for the query set losses at θ_{ik} (**outer loop**)

- Inner loop update:

$$\theta_0 = \theta; \quad \theta_{ik} = \theta_{i(k-1)} - \alpha \nabla_{\theta_{i(k-1)}} L_{\theta_{i(k-1)}} (x_s, y_s)$$

- Outer loop update:

$$\theta = \theta - \beta \nabla_{\theta} \sum_{T_i} L_{\theta}(x_q, y_q)$$

[Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, *Finn et al.*, ICML 2017]

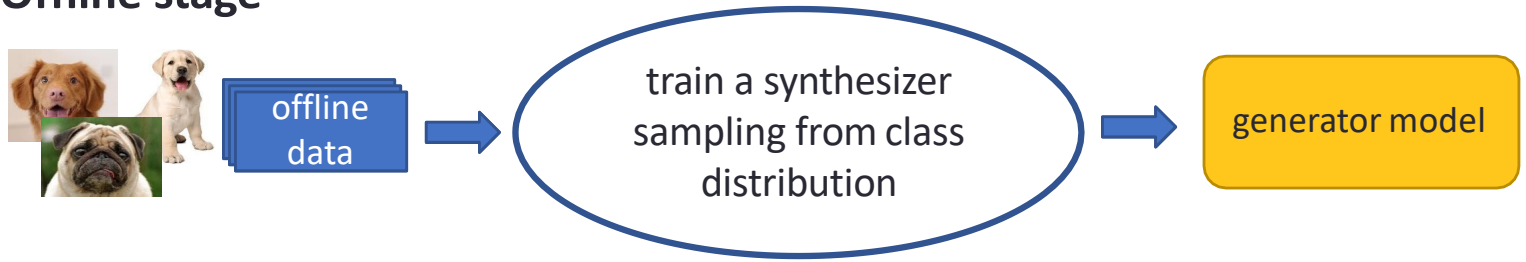
Model Agnostic Meta Learning (MAML)

- Other MAML-like algorithms:
  - MAML is slow and memory-intensive as it must calculate a gradient through a gradient.
  - FO-MAML
    - Uses the gradients wrt θ_{ik} to update θ.
    - [Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, *Finn et al.*, ICML 2017]
  - Reptile
    - Updates θ without gradients by moving them towards θ_{ik} by a finite step.
    - [On First-Order Meta-Learning Algorithms, *Nichol et al.*, arXiv preprint 1803.02999]

Data Augmentation

"Learning to generate"

Offline stage



- Relegate the task of learning the data manifold to the synthesizer model

On new task data

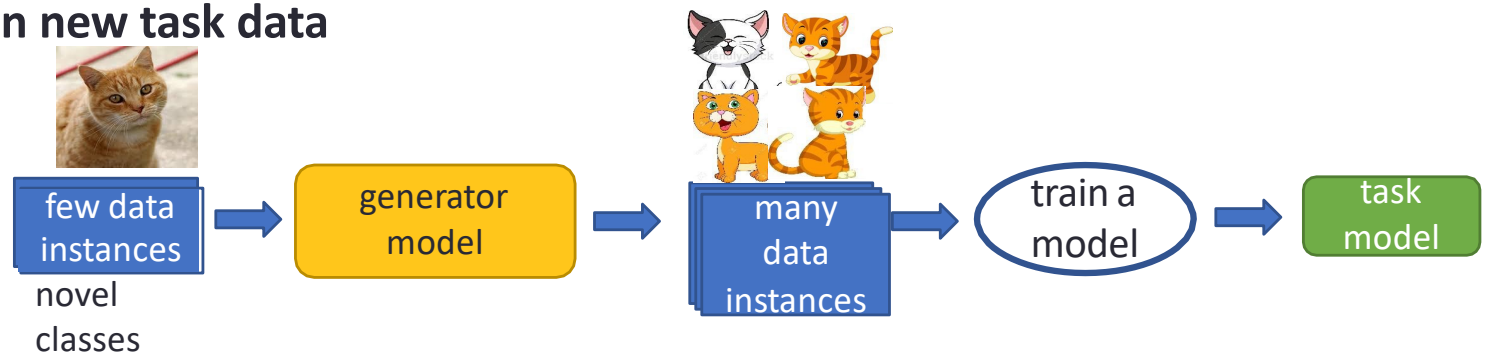
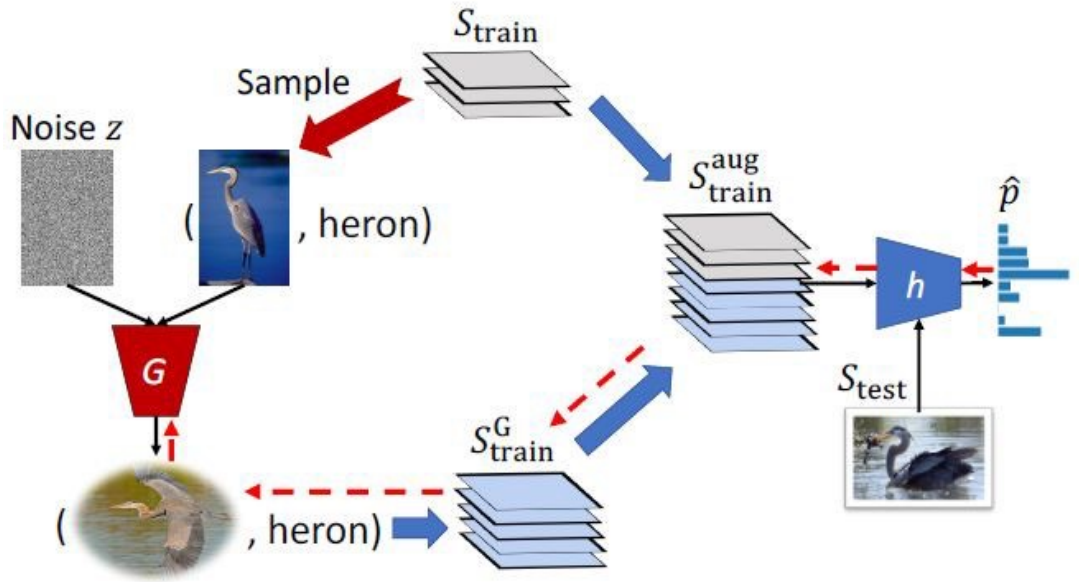


Image source: Few-shot Learning, *Joseph Shtok*, IBM AI Research (slides)

Feature Hallucination

- The generator is part of the classifier pipeline trained end-to-end
  - The classifier h is differentiable w.r.t. the training data
- Once the generator is trained in this setup, it can be used to generate new instances for novel classes
  - Using the (novel input, class label) tuple as input



[Low-Shot Learning from Imaginary Data, *Wang et al.*, CVPR 2018]

Complementary References

- Pedro Antonio Gutiérrez, María Pérez-Ortiz, Javier Sánchez-Monedero, Francisco Fernández-Navarro, César Hervás-Martínez: Ordinal Regression Methods: Survey and Experimental Study. IEEE Trans. Knowl. Data Eng. 28(1): 127-146 (2016).

- José Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michal Wozniak, Salvador García: Monotonic classification: An overview on algorithms, performance measures and data sets. Neurocomputing 341: 168-182 (2019).

- Juan Luis Suárez, Salvador García, Francisco Herrera. A Tutorial on Distance Metric Learning: Mathematical Foundations, Algorithms, Experimental Analysis, Prospects and Challenges. Neurocomputing 425: 300-322 (2021).

- Germán González-Almagro, Daniel Peralta, Eli De Poorter, José Ramón Cano, Salvador García: Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions. CoRR abs/2303.00522 (2023)

- Yaqing Wang, Quanming Yao, James Kwok, Lionel M. Ni. Generalizing from a Few Examples: A Survey on Few-Shot Learning. arXiv:1904.05046v3, 2020.