

Unsupervised and semi-supervised learning

Aude Sportisse

`<aude.sportisse@univ-grenoble-alpes.fr>`

These slides are based on the course provided by Massih-Reza Amini.

Plan

Unsupervised learning

Introduction

Clustering

- Hierarchical clustering

- K-means algorithm

- Mixture models

- Clustering evaluation

Dimensionality reduction techniques

- Principal Component Analysis

- Variational autoencoders

- Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

What is unsupervised learning ?

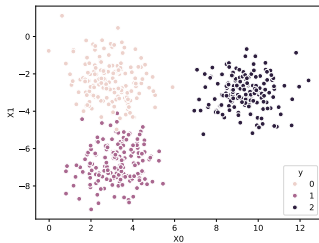


Figure: Access to the labels

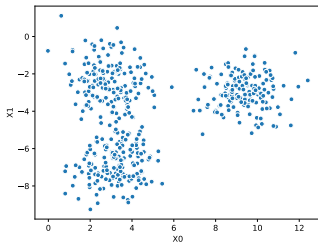


Figure: No labels

Supervised vs unsupervised learning

Supervised learning:

Unsupervised learning:

Goals of unsupervised learning

- ▶ Access to n observations: $(X_1, \dots, X_n)$
- ▶ i.i.d. observations drawn from the distribution $p(X)$ with $X \in \mathcal{X}^d$ a random vector.

Focus on two main tasks

Clustering

Identify and separate distinct groups within the data

Each data point is assigned to a cluster

Dimensionality reduction

Summarize, reduce noise, leverage associations among the variables

New representation of the dataset in a lower-dimensional space

Goals of unsupervised learning

- ▶ Access to n observations: $(X_1, \dots, X_n)$
- ▶ i.i.d. observations drawn from the distribution $p(X)$ with $X \in \mathcal{X}^d$ a random vector.

Focus on two main tasks

Clustering

Identify and separate distinct groups within the data

Each data point is assigned to a cluster

Dimensionality reduction

Summarize, reduce noise, leverage associations among the variables

New representation of the dataset in a lower-dimensional space

Plan

Unsupervised learning

- Introduction

Clustering

- Hierarchical clustering

- K-means algorithm

- Mixture models

- Clustering evaluation

Dimensionality reduction techniques

- Principal Component Analysis

- Variational autoencoders

- Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Clustering

- ▶ Goal: intelligently organizing the data, i.e. find a partition of the data such that the points are:
 - similar within clusters,
 - not similar between the clusters
- ▶ Let G be a partition found over the collection $\mathcal{C}$ of N observations. An element of G is called *group* or *cluster*. A group, G_k , $1 \leq k \leq |G|$, corresponds to a subset of observations in $\mathcal{C}$.
- ▶ A representative of a group G_k , generally its center of gravity $\mathbf{r}_k$, is called prototype.

📌 Example:

Clustering

- ▶ Goal: intelligently organizing the data, i.e. find a partition of the data such that the points are:
 - similar within clusters,
 - not similar between the clusters
- ▶ Let G be a partition found over the collection $\mathcal{C}$ of N observations. An element of G is called *group* or *cluster*. A group, G_k , $1 \leq k \leq |G|$, corresponds to a subset of observations in $\mathcal{C}$.
- ▶ A representative of a group G_k , generally its center of gravity $\mathbf{r}_k$, is called prototype.

🔗 Example:

Clustering

- ▶ Goal: intelligently organizing the data, i.e. find a partition of the data such that the points are:
 - similar within clusters,
 - not similar between the clusters
- ▶ Let G be a partition found over the collection $\mathcal{C}$ of N observations. An element of G is called *group* or *cluster*. A group, G_k , $1 \leq k \leq |G|$, corresponds to a subset of observations in $\mathcal{C}$.
- ▶ A representative of a group G_k , generally its center of gravity $\mathbf{r}_k$, is called prototype.

🔗 Example:

Classification vs clustering

- ▶ In **classification** : we have pairs of examples constituted by observations and their associated class labels $(X, y) \in \mathbb{R}^d \times \{1, \dots, K\}$.
 - The class information is provided by an expert and the aim is to find a prediction function $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ that makes the association between the inputs and the outputs
- ▶ In **clustering** : we observe X but we do not observe y .
 - the aim is to find homogeneous clusters reflecting the relationship between observations.

The problem is difficult:

- **the partition is predicted up to a permutation**, i.e. we can predict y given X up to a permutation of the modalities of y .
- **we do not know the exact number of clusters K** . It is often considered as an hyperparameter (fixed before hand to some arbitrary value or using prior information).

Classification vs clustering

- ▶ In **classification** : we have pairs of examples constituted by observations and their associated class labels $(X, y) \in \mathbb{R}^d \times \{1, \dots, K\}$.
 - The class information is provided by an expert and the aim is to find a prediction function $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ that makes the association between the inputs and the outputs
- ▶ In **clustering** : we observe X but we do not observe y .
 - the aim is to find homogeneous clusters reflecting the relationship between observations.

The problem is difficult:

- **the partition is predicted up to a permutation**, i.e. we can predict y given X up to a permutation of the modalities of y .
- **we do not know the exact number of clusters K** . It is often considered as an hyperparameter (fixed before hand to some arbitrary value or using prior information).

Different forms of clustering

There are two main forms of clustering:

1. *Flat clustering* produces a partition of the data points into K clusters, independent one from another.
 - ✓ easy to interpret, faster than hierarchical methods
 - ✗ Main drawback: lack of hierarchy in clusters (equal size and variance), the number of clusters K is required
2. *Hierarchical clustering* forms a *structured* partition, organized as a tree.
 - ✓ no need to set the number of clusters K
 - ✗ their complexity is in general quadratic in the number of observations !

Different forms of clustering

There are two main forms of clustering:

1. *Flat clustering* produces a partition of the data points into K clusters, independent one from another.
 - ✓ easy to interpret, faster than hierarchical methods
 - ✗ Main drawback: lack of hierarchy in clusters (equal size and variance), the number of clusters K is required
2. *Hierarchical clustering* forms a *structured* partition, organized as a tree.
 - ✓ no need to set the number of clusters K
 - ✗ their complexity is in general quadratic in the number of observations !

Plan

Unsupervised learning

Introduction

Clustering

Hierarchical clustering

K-means algorithm

Mixture models

Clustering evaluation

Dimensionality reduction techniques

Principal Component Analysis

Variational autoencoders

Application to impute missing values

Semi-supervised learning

Introduction and classical methods

Deep semi-supervised learning

Hierarchical clustering

Hierarchical representation in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level.

- ▶ Lowest level: each data point has its individual clusters
- ▶ Highest level: there is only one cluster containing all the data points.

→ Hierarchy of clusters is represented as a tree: dendrogram.

We can "cut" the hierarchical clustering at some similarity point to obtain a partition of K disjoint clusters as for the flat clustering.

Hierarchical clustering

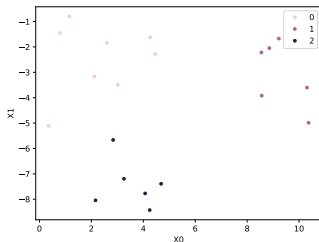


Figure: Partition for 20 points if we cut the hierarchical clustering to $K = 3$ clusters

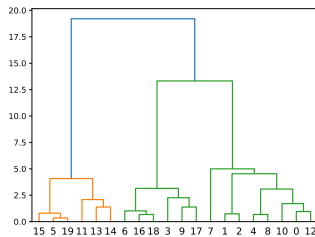


Figure: Dendrogram for 20 points

Plan

Unsupervised learning

- Introduction

Clustering

- Hierarchical clustering

- K-means algorithm

- Mixture models

- Clustering evaluation

Dimensionality reduction techniques

- Principal Component Analysis

- Variational autoencoders

- Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Cost minimization methods

- ▶ *K*-means algorithm: find the partition which minimizes the *within-cluster sum-of-squares criterion*:

$$\operatorname{argmin}_G \sum_{k=1}^K \sum_{X \in G_k} \underbrace{\|X - \mathbf{r}_k\|_2^2}_{\text{Euclidean distance between a point and the center}}$$

- ▶ Euclidean distance:

$$\operatorname{dist}_{\text{eucl}}(x, x') = \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

It minimizes the average dissimilarity of the observations from the cluster mean.

Cost minimization methods

- ▶ *K*-means algorithm: find the partition which minimizes the *within-cluster sum-of-squares criterion*:

$$\operatorname{argmin}_G \sum_{k=1}^K \sum_{X \in G_k} \underbrace{\|X - \mathbf{r}_k\|_2^2}_{\text{Euclidean distance between a point and the center}}$$

- ▶ Euclidean distance:

$$\operatorname{dist}_{\text{eucl}}(x, x') = \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

It minimizes the average dissimilarity of the observations from the cluster mean.

K-means algorithm

Algorithm K-means algorithm (MacQueen, 1967)

Initialisation: initial centers $(\mathbf{r}_k^{(1)})_{1 \leq k \leq K}$, number of clusters K

Iteratively do:

Finds new clusters by affecting each observation to the center to which it is the closest;

$$G_k^{(t)} = \{X_i \text{ such that } \|X_i - \mathbf{r}_k^{(t)}\|^2 \leq \|X_i - \mathbf{r}_j^{(t)}\|^2, \forall j \neq k, 1 \leq j \leq K\}$$

Estimates new centers for the clusters that have been found:

$$\mathbf{r}_k^{(t+1)} = \frac{1}{|G_k^{(t)}|} \sum_{i \in G_k^{(t)}} X_i$$

K-means algorithm

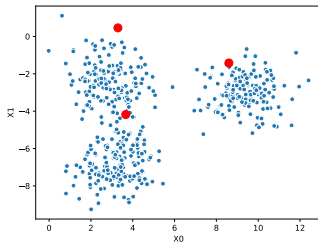


Figure: Initialization

K-means algorithm

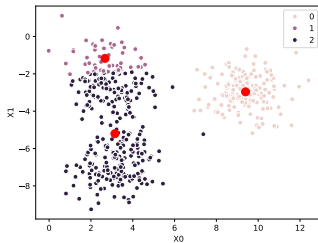


Figure: Iteration 1

K-means algorithm

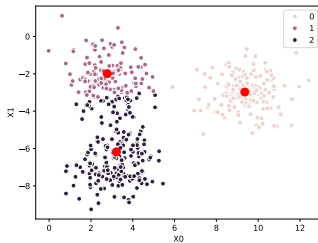


Figure: Iteration 2

K-means algorithm

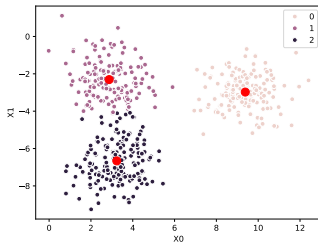


Figure: Iteration 3

K-means algorithm

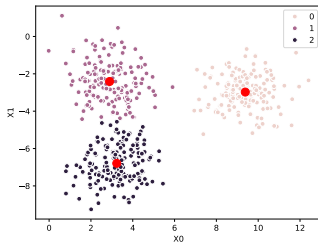


Figure: Iteration 4

K-means algorithm

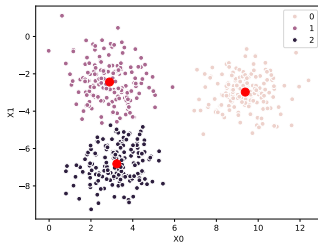


Figure: Iteration 5

Plan

Unsupervised learning

- Introduction

- Clustering**

 - Hierarchical clustering

 - K-means algorithm

 - Mixture models**

 - Clustering evaluation

- Dimensionality reduction techniques

 - Principal Component Analysis

 - Variational autoencoders

 - Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Mixture models

- ▶ Assume that each group G_k is generated by a probability density p_k of parameters θ_k
- ▶ Following the law of total probability, an observation X is then supposed to be generated with a probability

$$P(X, \Theta) = \sum_{k=1}^K \pi_k p_k(X|\theta_k),$$

where π_k is the probability of belonging to group k such that $\pi_k > 0$ and $\sum_{k=1}^K \pi_k = 1$. The parameters of the mixture are $\Theta = \{\pi_k, \theta_k; k \in \{1, \dots, K\}\}$.

Mixture models

- ▶ The aim is then to find the parameters Θ^* that maximize the log-likelihood

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \mathcal{L}_M(\Theta),$$

$$\mathcal{L}_M(X|\Theta) = \sum_{i=1}^n \ln \left[\sum_{k=1}^K \pi_k p_k(X_i|\theta_k) \right]$$

- The direct maximisation of this criterion is impossible because it implies a sum of a logarithm of a sum.
- We can use iterative methods such as the EM algorithm.

Expectation Maximization algorithm

Idea: "augment the data", i.e. considering that we have access to (X, z) where z is a latent variable which represents the class membership.

$$z_{ik} = \begin{cases} 1 & \text{if } X_i \text{ belongs to group } k \\ 0 & \text{otherwise.} \end{cases}$$

We assume that $(z_i)_{i=1,\dots,K}$ are independent and identically distributed, and each z_i follows a multinomial distribution of one draw from K categories.

EM algorithm

We have:

$$\begin{cases} z_i \sim \mathcal{M}(1, \pi) \\ X_i | z_{ik} = 1 \sim P(X_i, \theta_k) \end{cases}$$

and **the complete log-likelihood is**

$$\ell_C(X, z|\Theta) = \prod_{i=1}^n \prod_{k=1}^K [\pi_k f(X_i | z_{ik} = 1, \theta_k)]^{z_{ik}}$$

$$\mathcal{L}_C(X, z|\Theta) = \sum_{i=1}^n \sum_{k=1}^K z_{ik} \ln [\pi_k f(X_i | z_{ik} = 1, \theta_k)]$$

EM algorithm

The complete log-likelihood $\mathcal{L}_C(X, z|\Theta)$ is a lower bound of $\mathcal{L}_M(X|\Theta)$. Optimizing $\mathcal{L}_C(X, z|\Theta)$ over Θ will automatically lead in the optimization of $\mathcal{L}_M(X|\Theta)$.

$$\begin{aligned}\mathcal{L}_C(X, z|\Theta) &= \sum_{i=1}^n \log p((X_i, z_i)|\Theta) \\ &= \sum_{i=1}^n [\log(p(z_i|X_i, \Theta)) + \log p(X_i|\Theta)] \\ &= \mathcal{L}_M(X|\Theta) + \sum_{i=1}^n \log(p(z_i|X_i, \Theta))\end{aligned}$$

It implies

$$\mathcal{L}_M(X|\Theta) = \mathcal{L}_C(X, z|\Theta) - \sum_{i=1}^n \log(p(z_i|X_i, \Theta))$$

EM algorithm

Algorithm EM algorithm for clustering (Dempster et al., 1977)

Initialisation: $\Theta^{(0)}$

Iteratively do:

Expectation step (E-step):

Compute the **expected** likelihood given the **current information**

$$Q(\Theta; \Theta^{(t)}) = \mathbb{E}[\mathcal{L}_C(X, z|\Theta) \mid \underbrace{X, \Theta^{(t)}}_{\text{observed data and current value of } \Theta}]$$

Maximization step (M-step):

$$\Theta^{(t+1)} \in \operatorname{argmax}_{\Theta} Q(\Theta, \Theta^{(t)})$$

-
- ▶ Converges to a local optimum
 - ▶ Dependence of the converged estimate on the initial values
 - ▶ One EM algorithm for each statistical model

EM algorithm

Algorithm EM algorithm for clustering (Dempster et al., 1977)

Initialisation: $\Theta^{(0)}$

Iteratively do:

Expectation step (E-step):

Compute the **expected** likelihood given the **current information**

$$Q(\Theta; \Theta^{(t)}) = \mathbb{E}[\mathcal{L}_C(X, z|\Theta) \mid \underbrace{X, \Theta^{(t)}}_{\text{observed data and current value of } \Theta}]$$

Maximization step (M-step):

$$\Theta^{(t+1)} \in \operatorname{argmax}_{\Theta} Q(\Theta, \Theta^{(t)})$$

-
- ▶ Converges to a local optimum
 - ▶ Dependence of the converged estimate on the initial values
 - ▶ One EM algorithm for each statistical model

Mixture models

After estimating the parameters, we can assign each observation to a group using the Bayesian decision rule:

$$X_i \in G_k \Leftrightarrow k = \underset{\ell}{\operatorname{argmax}} P(z_{i\ell} = 1 \mid X_i, \Theta^*)$$

where

$$\begin{aligned} \forall \ell \in \{1, \dots, K\}, P(z_{i\ell} = 1 \mid X_i, \Theta^*) &= \frac{\pi_\ell^* P(X_i \mid z_{i\ell} = 1, \theta_\ell^*)}{P(X_i, \Theta^*)} \\ &\propto \pi_\ell^* P(X_i \mid z_{i\ell} = 1, \theta_\ell^*) \end{aligned}$$

EM for Gaussian mixture

Comments on the EM algorithm

- ▶ EM algorithm can converge to a local optimum (and not global)
 - k -means algorithm can be used or other clustering methods
 - random initialization several times and at the end keep only the initialization which has the highest log-likelihood
- ▶ Stopping criteria
- ▶ EM algorithm does not learn directly the partition: estimation of Θ + Partition

Classification EM algorithm

Idea: the cluster assignments, denoted as z , are parameters to be learned: a partition into K clusters of the observations is defined in a classification step.

- ▶ Directly learns the partition
- ▶ Converges in a finite number of iterations
- ▶ K -means-like algorithm

Classification EM algorithm

Algorithm CEM algorithm for clustering (Celeux and Govaert, 1992)

Initialisation: initial partition $z^{(0)}$.

Iteratively do:

E-step: Estimate the posterior probabilities using $\Theta^{(t)}$:

$$\forall \ell \in \{1, \dots, K\}, \mathbb{E}[z_{i\ell} \mid X_i, z^{(t)}, \Theta^{(t)}] = \frac{\pi_\ell^{(t)} P(X_i \mid z_{i\ell}^{(t)} = 1, \theta_\ell^{(t)})}{\sum_{k=1}^K \pi_k^{(t)} P(X_i \mid z_{ik}^{(t)} = 1, \theta_k^{(t)})}$$

Classification step (C-step): Assign to each example X_i its partition, the one for which the posterior probability is maximum. Note $z^{(t+1)}$ this new partition

M-step:

$$\Theta^{(t+1)} \in \operatorname{argmax}_{\Theta} \mathcal{L}_C(X, z^{(t+1)} \mid \Theta)$$

Comparison of the algorithms

EM-CEM	k -means
2 steps for EM 1 step for CEM	Only partitioning
Well fitted to the data	Small number of parameters
1 alg. for each statistical model	
Initialization is difficult	
Number of clusters K required	

Plan

Unsupervised learning

- Introduction

Clustering

- Hierarchical clustering

- K-means algorithm

- Mixture models

- Clustering evaluation

Dimensionality reduction techniques

- Principal Component Analysis

- Variational autoencoders

- Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Purity measure

The **purity measure** tends to quantify the ability of the clustering method to group the observations of the same class into the same partitions .

Let G be the partition found and C the set of classes found over G . The purity measure is then defined by:

$$\text{pure}(G, C) = \frac{1}{n} \sum_k \max_l |G_k \cap C_l|$$

✎ Example:

Mutual Information

The **Mutual Information** (MI) measures the agreement of two independent partitions U and V .

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{n} \log \frac{n|U_i \cap V_j|}{|U_i||V_j|},$$

where

- ▶ $|U_i|$ is the number of samples in cluster U_i
- ▶ $|V_j|$ is the number of samples in cluster V_j
- ▶ $|U_i \cap V_j|$ is the number of samples in both cluster U_i and V_j

→ A normalized version exists.

Bayesian information criterion

The **Bayesian information criterion** (BIC) is a criterion for model selection.

$$\text{BIC} = \frac{b}{2} \log(n) - \hat{\mathcal{L}}_c(X, \mathbf{z}|\Theta),$$

with

- ▶ b : the number of parameters estimated by the model
- ▶ n : the number of samples
- ▶ $\hat{\mathcal{L}}_c$: the maximized value of the likelihood function

Plan

Unsupervised learning

- Introduction

- Clustering

 - Hierarchical clustering

 - K-means algorithm

 - Mixture models

 - Clustering evaluation

- Dimensionality reduction techniques

 - Principal Component Analysis

 - Variational autoencoders

 - Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Matrix approximation

- ▶ Singular Value Decomposition (SVD) : r is the rank of $\mathbb{X}$

$$\text{SVD}(\mathbb{X}) = \underbrace{\left[\mathbf{U}_1 \mid \dots \mid \mathbf{U}_n \right]}_{\text{new coordinates}} \underbrace{\begin{bmatrix} \sigma_1(\mathbb{X}) & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & \sigma_r(\mathbb{X}) \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}}_{\text{singular values}} \underbrace{\begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_d^\top \end{bmatrix}}_{\text{new basis}}$$

$\mathbb{R}^{n \times d}$

$\mathbb{R}^{n \times n}$

$\mathbb{R}^{n \times d}$

$\mathbb{R}^{d \times d}$

Principal Component Analysis

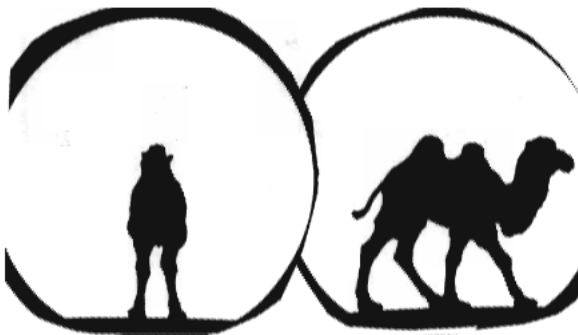


Figure: Two different projections.

Principal Component Analysis

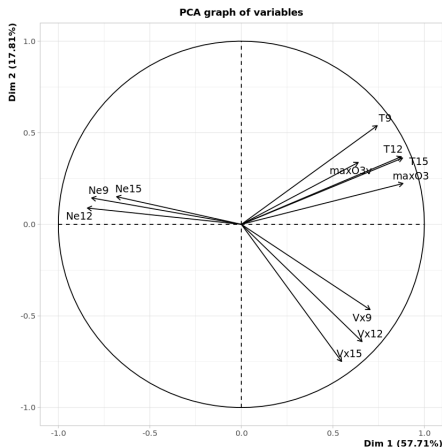
How to perform Principal Component Analysis ?

- ▶ Subtract off the mean for each variable of the data matrix
- ▶ Calculate the SVD or the eigenvectors of the covariance matrix: they are the **principal components** of the data matrix.

Goals:

- ▶ Summarize the data, reduce the dimension
- ▶ Use and represent the correlations between the variables

Principal Component Analysis



- ▶ First principal component of PCA appears to be closely related to cloudiness measurements (Ne12, Ne9, Ne15), while the second principal component appears to be constructed using a mixture of all other variables.
- ▶ Cloudiness variables are highly correlated with each other, as are wind variables and temperature variables with maximum ozone.

Plan

Unsupervised learning

Introduction

Clustering

Hierarchical clustering

K-means algorithm

Mixture models

Clustering evaluation

Dimensionality reduction techniques

Principal Component Analysis

Variational autoencoders

Application to impute missing values

Semi-supervised learning

Introduction and classical methods

Deep semi-supervised learning

To PCA from variational autoencoders

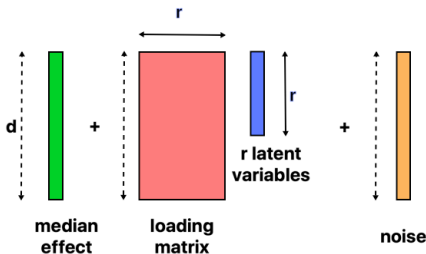
- Perform PCA = estimate a low-rank matrix

$$\underbrace{\mathbf{X}}_{\text{data matrix}} = \underbrace{\mathbf{\Theta}}_{\text{low-rank matrix to estimate}} + \underbrace{\epsilon}_{\text{noise}}$$

- PPCA: consider random effects

$$\mathbf{X} = \boldsymbol{\alpha} + \mathbf{B}\mathbf{Z} + \epsilon \quad (\text{for one row})$$

with $\mathbf{Z} \in \mathbb{R}^{r \times 1}$ (latent variables), $\mathbf{B} \in \mathbb{R}^{d \times r}$ (loading matrix), $\boldsymbol{\alpha} \in \mathbb{R}^d$ (median effect) and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (noise).



To PCA from variational autoencoders

- ▶ Perform PCA = estimate a low-rank matrix

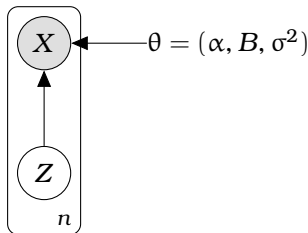
$$\underbrace{\mathbb{X}}_{\text{data matrix}} = \underbrace{\Theta}_{\text{low-rank matrix to estimate}} + \underbrace{\epsilon}_{\text{noise}}$$

- ▶ PPCA: consider random effects

$$X = \alpha + BZ + \epsilon \quad (\text{for one row})$$

with $Z \in \mathbb{R}^{r \times 1}$ (latent variables), $B \in \mathbb{R}^{d \times r}$ (loading matrix), $\alpha \in \mathbb{R}^d$ (median effect) and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (noise).

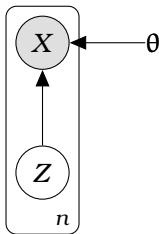
$$\begin{cases} Z \sim \mathcal{N}(0, I_{r \times r}) \\ X|Z \sim \mathcal{N}(\alpha + BZ, \sigma^2 I_{d \times d}) \end{cases}$$



Variational autoencoders

$$\begin{cases} Z \sim p(Z) \\ X \sim p_{\theta}(X|Z) = \Phi(X|f_{\theta}(Z)) \quad (\text{decoder}) \end{cases}$$

Consider $\Phi(\cdot; \tau)$ the Gaussian distribution of parameter τ .
Typically: $f_{\theta}(Z)$ is a deep neural network



Plan

Unsupervised learning

Introduction

Clustering

Hierarchical clustering

K-means algorithm

Mixture models

Clustering evaluation

Dimensionality reduction techniques

Principal Component Analysis

Variational autoencoders

Application to impute missing values

Semi-supervised learning

Introduction and classical methods

Deep semi-supervised learning

Change of basis with SVD

Truncated Singular Value Decomposition (SVD) with rank $\mathbf{r}$

$$\text{SVD}_{\mathbf{r}}(\mathbb{X}) = \underbrace{\begin{bmatrix} \mathbf{U}_1 & \cdots & \mathbf{U}_{\mathbf{r}} \end{bmatrix}}_{\text{new coordinates}} \underbrace{\begin{bmatrix} \sigma_1(\mathbb{X}) & 0 & \cdots \\ 0 & \ddots & 0 \\ \vdots & 0 & \sigma_{\mathbf{r}}(\mathbb{X}) \end{bmatrix}}_{\text{singular values}} \underbrace{\begin{bmatrix} \mathbf{V}_1^\top \\ \vdots \\ \mathbf{V}_{\mathbf{r}}^\top \end{bmatrix}}_{\text{new basis}}$$

$\mathbb{R}^{n \times d}$
 $\mathbb{R}^{n \times \mathbf{r}}$
 $\mathbb{R}^{\mathbf{r} \times \mathbf{r}}$
 $\mathbb{R}^{\mathbf{r} \times d}$

Some notations

- ▶ We assume that we have NAs in the data matrix.
- ▶ We consider the missing-data indicator M : $M_{ij} = 1$ if X_{ij} is missing and $M_{ij} = 0$ otherwise.

Iterative PCA with NA

Algorithm Iterative Principal Component Analysis (simplest version)

Input: r (number of kept dimensions)

Initialisation: $X^{(0)}$ (mean imputation)

Iteratively do:

Estimation step:

Compute the **SVD** with r dim. on the completed dataset:

$$\mathbf{SVD}_r(X^{(t)}) = U_r D_r V_r^t,$$

Imputation step:

Predict the missing values with the **SVD**

$$X^{(t+1)} = \underbrace{X \odot (\mathbf{1}_{n \times d} - M)}_{\text{on observed values}} + \underbrace{\mathbf{SVD}_r(X^{(t)}) \odot M}_{\text{on missing values}}$$

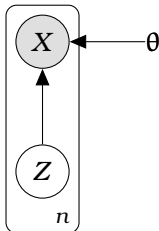
Change of basis with deep learning

Deep latent variable model

$$\begin{cases} Z \sim p(Z) \\ X \sim p_{\theta}(X|Z) = \Phi(X|f_{\theta}(Z)) \quad (\text{decoder}) \end{cases}$$

Consider $\Phi(\cdot; \tau)$ the Gaussian distribution of parameter τ .

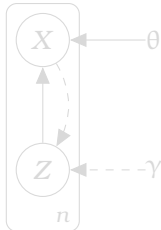
Typically: $f_{\theta}(Z)$ is a deep neural network



DLVM with NA: MIWAE (?)

$$\begin{cases} Z \sim q_{\gamma}(Z|X_{\text{obs}(M)}) = \Phi(Z|g_{\gamma}(X_{\text{obs}(M)})) & \text{(encoder)} \\ X \sim p_{\theta}(X|Z) = \Phi(X_{\text{obs}(M)}|f_{\theta}(Z)) & \text{(decoder)} \end{cases}$$

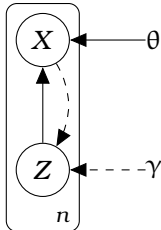
$g_{\gamma}(X)$ is also a deep neural network



DLVM with NA: MIWAE (?)

$$\begin{cases} Z \sim q_{\gamma}(Z|X_{\text{obs}(M)}) = \Phi(Z|g_{\gamma}(X_{\text{obs}(M)})) & \text{(encoder)} \\ X \sim p_{\theta}(X|Z) = \Phi(X_{\text{obs}(M)}|f_{\theta}(Z)) & \text{(decoder)} \end{cases}$$

$g_{\gamma}(X)$ is also a deep neural network



Impute with DLVM (?)

$$\begin{aligned} & \mathbb{E}[X_{\text{mis}(M)} | X_{\text{obs}(M)}] \\ &= \int X_{\text{mis}(M)} p_{\theta}(X_{\text{mis}(M)} | X_{\text{obs}(M)}) dX_{\text{mis}(M)} \\ &= \int \int X_{\text{mis}(M)} p_{\theta}(X_{\text{mis}(M)} | X_{\text{obs}(M)}, Z) p_{\theta}(Z | X_{\text{obs}(M)}) dZ dX_{\text{mis}(M)} \\ &= \int \int X_{\text{mis}(M)} \frac{p_{\theta}(Z | X_{\text{obs}(M)})}{q_{\gamma}(Z | X_{\text{obs}(M)})} p_{\theta}(X_{\text{mis}(M)} | X_{\text{obs}(M)}, Z) q_{\gamma}(Z | X_{\text{obs}(M)}) dZ dX_{\text{mis}(M)} \end{aligned}$$

Estimate with self-normalised importance sampling:

$$\sum_{l=1}^L w_l X_{\text{mis}(M)}^{(l)}, \quad (\text{Section 9.2 of } \text{?})$$

with $w_l = \frac{r_l}{\sum_{l=1}^L r_l}$, $r_l = \frac{p_{\theta}(X_{\text{obs}(M)} | Z^{(l)}) p(Z^{(l)})}{q_{\gamma}(Z^{(l)} | X_{\text{obs}(M)})}$, and

$$(X_{\text{mis}(M)}^{(l)}, Z^{(l)}) \stackrel{\text{iid}}{\sim} p_{\theta}(X_{\text{mis}(M)} | X_{\text{obs}(M)}, Z) q_{\gamma}(Z | X_{\text{obs}(M)})$$

Plan

Unsupervised learning

- Introduction

- Clustering

 - Hierarchical clustering

 - K-means algorithm

 - Mixture models

 - Clustering evaluation

- Dimensionality reduction techniques

 - Principal Component Analysis

 - Variational autoencoders

 - Application to impute missing values

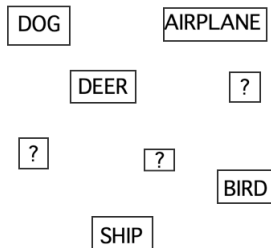
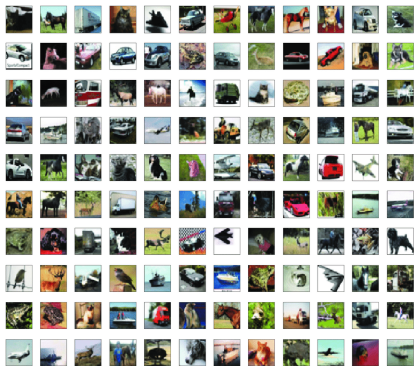
Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Context

- ▶ Huge amount of data is available.
- ▶ Labeling the data is costly and time-consuming.



Semi-Supervised Learning (SSL)

(Amini and Usunier, 2015) We observe:

- ▶ m labeled data:



DEER

- ▶ and u unlabeled data:



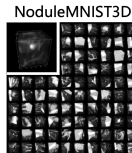
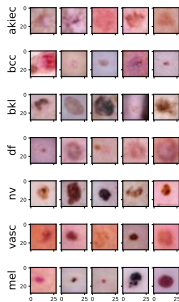
?

Typically: $m \ll u$

The goal is to leverage from the unlabeled data to construct a predictive model.

Semi-Supervised Learning (SSL)

- ▶ NoduleMNIST3D: images from thoracic scans
- ▶ dermaMNIST: 10,015 dermatoscopic images, 7 categories of skin diseases (benign naevi: most frequent class (71%))



Formally

We consider an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and an output space $\mathcal{Y}$.
We observe

- ▶ m pairs of examples (labeled data)
 $\mathcal{S} = \{(X_i, y_i), i \in \{1, \dots, m\}\}$ generated i.i.d from a probability distribution $\mathcal{D}$
- ▶ u observations (unlabeled data)
 $\mathcal{U} = \{X_i; i \in \{m+1, \dots, m+u\}\}$ also generated i.i.d from a marginal $\mathcal{D}_{\mathcal{X}}$

The goal is to **construct a prediction function** $f : \mathcal{X} \rightarrow \mathcal{Y}$ which predicts an output y for a given new X with a minimum probability of error.

Formally

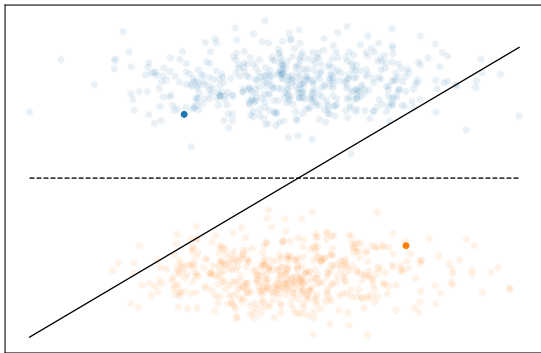
We consider an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and an output space $\mathcal{Y}$.
We observe

- ▶ m pairs of examples (labeled data)
 $\mathcal{S} = \{(X_i, y_i), i \in \{1, \dots, m\}\}$ generated i.i.d from a probability distribution $\mathcal{D}$
- ▶ u observations (unlabeled data)
 $\mathcal{U} = \{X_i; i \in \{m+1, \dots, m+u\}\}$ also generated i.i.d from a marginal $\mathcal{D}_{\mathcal{X}}$

The goal is to **construct a prediction function** $f : \mathcal{X} \rightarrow \mathcal{Y}$ which predicts an output y for a given new X with a minimum probability of error.

SSL intuition

- ▶ Darker points = labeled points
- ▶ Solid line: decision boundary if we only use the 2 labeled points
- ▶ Dashed line: decision boundary if we use all the points



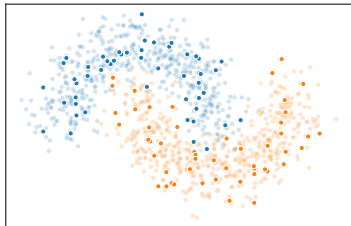
Assumptions

Why do we need assumptions? We want that unlabelled data improve our knowledge of $p(X)$ (and therefore of $p(y|X)$)

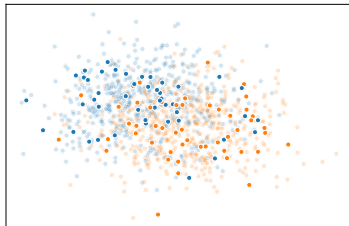
→ $p(X)$ should contain information about $p(y|X)$

- ▶ **Cluster assumption:** if points are in the same cluster, they are likely to be of the same class
- ▶ **Low-density assumption:** decision boundary of a classifier should preferably pass through low-density regions in the input space
- ▶ **Manifold assumption:** observations are roughly contained in a low dimensional manifold
- ▶ **Smoothness assumption:** two observations which are close in a high-density region have closed outputs

Which assumption ?



All assumptions hold.



The smoothness
assumption is uncertain,
and the cluster and
low-density assumptions do
not hold

Generative vs discriminative approaches

- ▶ **generative** : model $p(X|y)$ using unsupervised learning and then infer the predictive density $p(y|X)$ using the Bayes theorem.

Semi-supervised CEM algorithm

- ▶ **discriminative** : directly estimate the predictive density $p(y|X)$

self-training approach

Semi-supervised CEM

- ▶ The density probabilities $P(X | y = k, \theta_k^{(0)})$ are respectively estimated on the K classes from the labeled data $\mathcal{S}$, and $z^{(0)}$ is defined accordingly.
- ▶ Then, the algorithm is iterative:
 - Estimate the posterior class probability for each unlabeled data,
 - Assign each $X_i \in \mathcal{U}$ to the cluster $z_k^{(t+1)}$ with maximal posterior probability according to the posterior class probabilities,
 - Estimate the new parameters

Graph-based methods

- ▶ transductive paradigm: the primary goal is to use both labelled and unlabelled data to predict labels on the unlabelled data
 - define a graph of similarity between pairs of the input dataset (labelled and unlabelled) possibly weighted
 - predictor is trained for (i) predicting the true labels for the labelled samples and (ii) predicting the same labels for points that are highly similar with respect to the weights in the graph

Example of similarity matrix: k -nearest neighbours binary matrix, $\forall (i, j) \in \{1, \dots, m + u\}^2$:

$$W_{ij} = 1 \text{ iff } X_i \text{ is among the } k^{\text{th}}\text{-nearest neighbours of } X_j$$

Graph-based methods

- ▶ transductive paradigm: the primary goal is to use both labelled and unlabelled data to predict labels on the unlabelled data
 - define a graph of similarity between pairs of the input dataset (labelled and unlabelled) possibly weighted
 - predictor is trained for (i) predicting the true labels for the labelled samples and (ii) predicting the same labels for points that are highly similar with respect to the weights in the graph

Example of similarity matrix: k -nearest neighbours binary matrix, $\forall (i, j) \in \{1, \dots, m + u\}^2$:

$$W_{ij} = 1 \text{ iff } X_i \text{ is among the } k^{th}\text{-nearest neighbours of } X_j$$

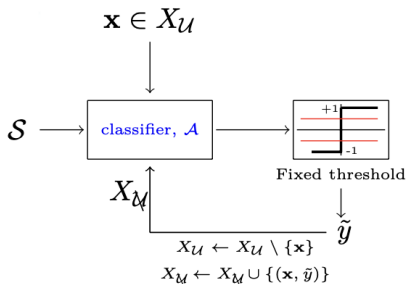
Label propagation

A simple idea to take advantage of the graph G built is to propagate the labels of labeled examples across the graph.

1. Initialization of the labels of all nodes
2. Each node updates its label to the one that the maximum numbers of its neighbours belongs to (at random if there is equality)
3. Convergence when each node has the majority label of its neighbours

Self-training algorithm

- ▶ train the algorithm on the labeled data
- ▶ use the prediction of the model to obtain pseudo-labels
- ▶ only keep the pseudo-labels which are superior to a given threshold



Plan

Unsupervised learning

- Introduction

- Clustering

 - Hierarchical clustering

 - K-means algorithm

 - Mixture models

 - Clustering evaluation

- Dimensionality reduction techniques

 - Principal Component Analysis

 - Variational autoencoders

 - Application to impute missing values

Semi-supervised learning

- Introduction and classical methods

- Deep semi-supervised learning

Mathematical formalism

- ▶ Unlabeled data can be seen as observations having a **missing label**.
- ▶ $R \in \{0, 1\}^n$ indicates where are the missing values in Y

$$\forall i \in \{1, \dots, n\}, R_i = \begin{cases} 1 & \text{if } Y_i \text{ is observed} \\ 0 & \text{otherwise.} \end{cases}$$

- ▶ Remark: Y is partially missing, but R is fully observed.

Mathematical formalism

- ▶ Unlabeled data can be seen as observations having a **missing label**.
- ▶ $R \in \{0, 1\}^n$ indicates where are the missing values in Y

$$\forall i \in \{1, \dots, n\}, R_i = \begin{cases} 1 & \text{if } Y_i \text{ is observed} \\ 0 & \text{otherwise.} \end{cases}$$

- ▶ Remark: Y is partially missing, but R is fully observed.

Mathematical formalism

- ▶ Unlabeled data can be seen as observations having a **missing label**.
- ▶ $R \in \{0, 1\}^n$ indicates where are the missing values in Y

$$\forall i \in \{1, \dots, n\}, R_i = \begin{cases} 1 & \text{if } Y_i \text{ is observed} \\ 0 & \text{otherwise.} \end{cases}$$

- ▶ Remark: Y is partially missing, but R is fully observed.

Mathematical formalism

- ▶ Objective: train a predictive model $p(Y|X; \theta)$
- ▶ Two modeling components:
 - Negative log-likelihood: $L(\theta, X, Y) = -\log p(Y|X; \theta)$.
 - $p(Y|X; \theta)$ is typically a convolutional neural network .
- ▶ The oracle estimate is the minimizer of the theoretical risk:

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{R}(\theta) := \mathbb{E}_{(X,Y) \sim p(X,Y)} [L(\theta; X, Y)].$$

The theoretical risk is always intractable.

Mathematical formalism

- ▶ Objective: train a predictive model $p(Y|X; \theta)$
- ▶ Two modeling components:
 - Negative log-likelihood: $L(\theta, X, Y) = -\log p(Y|X; \theta)$.
 - $p(Y|X; \theta)$ is typically a **convolutional neural network**.
- ▶ The oracle estimate is the minimizer of the **theoretical risk**:

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{R}(\theta) := \mathbb{E}_{(X,Y) \sim p(X,Y)} [L(\theta; X, Y)].$$

The theoretical risk is always intractable.

Mathematical formalism

- ▶ Objective: train a predictive model $p(Y|X; \theta)$
- ▶ Two modeling components:
 - Negative log-likelihood: $L(\theta, X, Y) = -\log p(Y|X; \theta)$.
 - $p(Y|X; \theta)$ is typically a convolutional neural network .
- ▶ The oracle estimate is the minimizer of the **theoretical risk**:

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{R}(\theta) := \mathbb{E}_{(X,Y) \sim p(X,Y)} [L(\theta; X, Y)].$$

The theoretical risk is always intractable.

Complete-case: learning with labeled data

- ▶ Minimize the empirical risk:

$$\hat{\theta} = \operatorname{argmin}_{\theta \in \Theta} \quad \hat{\mathcal{R}}(\theta) := \frac{1}{m+u} \sum_{i=1}^{m+u} L(\theta; X_i, Y_i).$$

The empirical risk is **unobserved** in presence of missing labels.

- ▶ Minimize the complete-case empirical risk:

$$\hat{\mathcal{R}}^{\text{CC}}(\theta) := \frac{1}{m} \underbrace{\sum_{i=1}^{m+u} R_i L(\theta; X_i, Y_i)}_{\text{only the labeled data are used}}$$

Complete-case: learning with labeled data

- ▶ Minimize the empirical risk:

$$\hat{\theta} = \operatorname{argmin}_{\theta \in \Theta} \quad \hat{\mathcal{R}}(\theta) := \frac{1}{m+u} \sum_{i=1}^{m+u} L(\theta; X_i, Y_i).$$

The empirical risk is **unobserved** in presence of missing labels.

- ▶ Minimize the complete-case empirical risk:

$$\hat{\mathcal{R}}^{\text{CC}}(\theta) := \frac{1}{m} \underbrace{\sum_{i=1}^{m+u} R_i L(\theta; X_i, Y_i)}_{\text{only the labeled data are used}}$$

Complete-case: learning with labeled data

- ▶ Minimize the empirical risk:

$$\hat{\theta} = \operatorname{argmin}_{\theta \in \Theta} \quad \hat{\mathcal{R}}(\theta) := \frac{1}{m+u} \sum_{i=1}^{m+u} L(\theta; X_i, Y_i).$$

The empirical risk is **unobserved** in presence of missing labels.

- ▶ Minimize the complete-case empirical risk:

$$\hat{\mathcal{R}}^{\text{CC}}(\theta) := \frac{1}{m} \underbrace{\sum_{i=1}^{m+u} \mathbf{R}_i L(\theta; X_i, Y_i)}_{\text{only the labeled data are used}}$$

Semi-supervised learning estimator

The goal is to **incorporate** the **unlabeled data**

$$\hat{\mathcal{R}}^{\text{SSL}}(\theta) := \underbrace{\frac{1}{m} \sum_{i=1}^{m+u} \textcolor{red}{R}_i L(\theta; X_i, Y_i)}_{\text{term on labeled data}} + \underbrace{\frac{\lambda}{u} \sum_{i=1}^{m+u} (\textcolor{blue}{1} - \textcolor{blue}{R}_i) H(\theta; X_i)}_{\text{term on unlabeled data}}$$

where:

- ▶ $\lambda > 0$: regularization parameter
- ▶ H : surrogate of L : $H \approx \mathbb{E}[L(\theta; X, Y)|X]$

Choice of the regularization

High-confident imputations for the unlabeled data

- ▶ Shannon entropy (Grandvalet and Bengio, 2004):

$$H(\theta; X) = - \sum_Y p(Y|X; \theta) \log(p(Y|X; \theta)).$$

- ▶ Pseudo-labels (Rizve et al., 2021):

- choose the class with the maximum predicted probability

$$c \in \operatorname{argmax}_Y p(Y|X; \theta)$$

- only the pseudo-labels which have a maximum predicted probability larger than a predefined threshold τ are used as target

$$H(\theta; X) = - \log p(c|X; \theta) \mathbf{1}_{\max_Y p(Y|X; \theta) > \tau}$$

Choice of the regularization

High-confident imputations for the unlabeled data

- ▶ Shannon entropy (Grandvalet and Bengio, 2004):

$$H(\theta; X) = - \sum_Y p(Y|X; \theta) \log(p(Y|X; \theta)).$$

- ▶ Pseudo-labels (Rizve et al., 2021):

- choose the class with the maximum predicted probability

$$c \in \operatorname{argmax}_Y p(Y|X; \theta)$$

- only the pseudo-labels which have a maximum predicted probability larger than a predefined threshold τ are used as target

$$H(\theta; X) = - \log p(c|X; \theta) \mathbf{1}_{\operatorname{max}_Y p(Y|X; \theta) > \tau}$$

Choice of the regularization

Robustness of the model to data augmentation

- ▶ Fixmatch (Sohn et al., 2020):
 - compute a pseudo-labels predicted using a weakly-augmented version of X .
 - minimize the likelihood with predictions of the model on a strongly-augmented version of X .

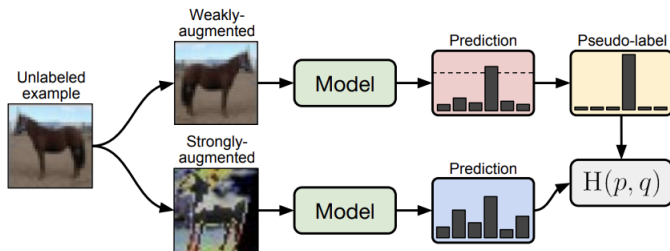


Figure: Credits (Sohn et al., 2020)

Limits of SSL

100 labeled images per class for CIFAR10

Error with supervised learning (neural network): 12%

Error using a large unlabeled dataset (FixMatch SSL): **2,5%**

But...

- ▶ Popular deep SSL techniques are generally not *safe*, meaning that their theoretical guarantees are not stronger than the complete case baseline (Schmutz et al., 2022).
- ▶ Without data augmentation, the gap in performance between using SSL and using only labeled data is smaller.
- ▶ Performances of SSL classical techniques are degraded when the labeled and unlabeled set have different distributions (Oliver et al.,).
- ▶ Many papers perform not realistic numerical experiments (too large complete validation set, costly hypertuning parameters,...) (Oliver et al.,).

The classical estimator is biased

- ▶ This estimator is biased !

$$\mathbb{E}[\hat{\mathcal{R}}^{\text{SSL}}(\theta)] \neq \mathcal{R}(\theta)$$

- ▶ Exercise: prove that the complete-case empirical risk is unbiased.
- ▶ Debiased estimator (Schmutz et al., 2022):

$$\frac{1}{m} \sum_{i=1}^{m+u} R_i L(\theta; X_i, Y_i) + \lambda \left(\frac{1}{u} \sum_{i=1}^{m+u} (1 - R_i) H(\theta; X_i) - \underbrace{\frac{1}{m} \sum_{i=1}^{m+u} R_i H(\theta; X_i)}_{\text{pseudo-labels on labeled data}} \right)$$

The classical estimator is biased

- ▶ This estimator is biased !

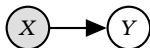
$$\mathbb{E}[\hat{\mathcal{R}}^{\text{SSL}}(\theta)] \neq \mathcal{R}(\theta)$$

- ▶ Exercise: prove that the complete-case empirical risk is unbiased.
- ▶ Debiased estimator (Schmutz et al., 2022):

$$\frac{1}{m} \sum_{i=1}^{m+u} \mathbf{R}_i L(\theta; X_i, Y_i) + \lambda \left(\frac{1}{u} \sum_{i=1}^{m+u} (1 - \mathbf{R}_i) H(\theta; X_i) - \underbrace{\frac{1}{m} \sum_{i=1}^{m+u} \mathbf{R}_i H(\theta; X_i)}_{\text{pseudo-labels on labeled data}} \right)$$

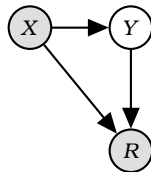
Towards realistic assumptions

$$p(R|X, Y) = p(R) \Leftrightarrow R \perp\!\!\!\perp X, Y$$



Informative labels

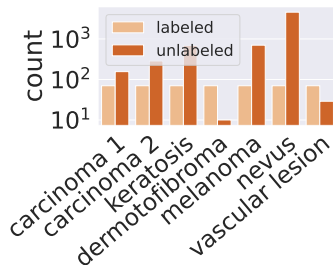
$$p(R|X, Y) = p(R|X, Y)$$



one should
model $p(R|X, Y)$

Towards realistic assumptions

- ▶ dermaMNIST: medical doctor would like to classify the conditions equally and select 70 images per class for labeling
- ▶ Labeled and unlabeled set have different distributions: we do not have $R \perp\!\!\!\perp X, Y$



References I

- ▶ Amini, M.-R. and Usunier, N. (2015). *Learning with partially labeled and interdependent data*. Springer.
- ▶ Celeux, G. and Govaert, G. (1992). A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332.
- ▶ Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.
- ▶ Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17.

References II

- ▶ [MacQueen, J. \(1967\)](#). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*.
- ▶ [Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I.](#) Realistic evaluation of deep semi-supervised learning algorithms, 2018.
- ▶ [Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. \(2021\)](#). In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*.
- ▶ [Schmutz, H., Humbert, O., and Mattei, P.-A. \(2022\)](#). Don't fear the unlabelled: safe semi-supervised learning via debiasing. In *The Eleventh International Conference on Learning Representations*.

References III

- Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608.