Practical introduction to machine learning

Part 1 : Data and Machine Learning problems

Rémi Flamary - CMAP, École Polytechnique

Master Data Science, Institut Polytechnique de Paris

October 11, 2023

Objectives of this course

Objectives

- Introduction to standard Machine Learning methods.
- Allow you to find which problem/method fits your application.
- Provide vocabulary and tools necessary for more in-depth study.
- Promote good practices, interpretation and reproducibility of ML.

What we will do

- Define major ML problems from unsupervised and supervised learning.
- Discuss in more details (optimization problem, parameters, algorithm) some classical approaches.
- Practical sessions on real data with Python/Numpy/Scikit-learn (100% of grade).

What we will not do

- Talk only about deep learning.
- Talk about everything on the slides (some information provided for reference only).
- Discuss in details the theory behind all the methods.
- Teach linear algebra, probability theory and Python programming (requirements).

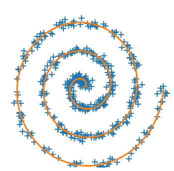
What is machine learning?

Objective of Machine Learning (ML)

Teach a machine to process automatically a large amount of data (signals, images, text, objects) in order to solve a given problem.

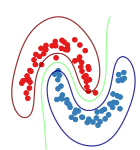
Unsupervised learning: Understanding the data.

- Clustering
- Probability Density Estimation
- Generative modeling
- Dimensionality reduction



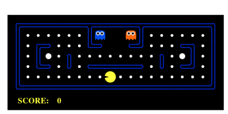
Supervised learning: Learning to predict.

- Classification
- Regression

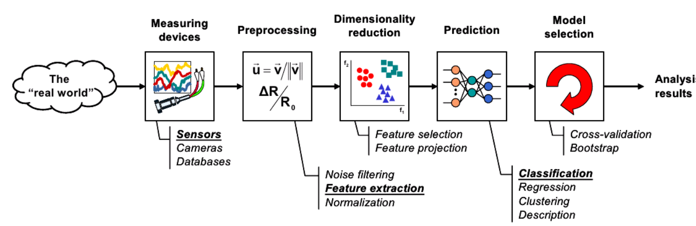


Reinforcement learning: Learn from environment.

Train a machine to choose actions that maximize a reward (games, autonomous vehicles, control).



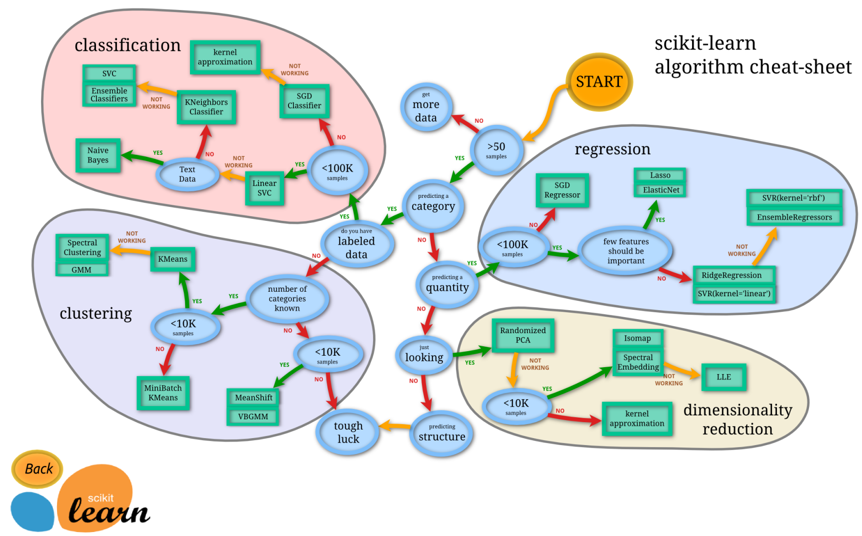
Machine learning in practice



- **Data acquisition** : sensor, databases, manual or automatic labeling
- **Pre-processing** : denoising, formating, numerical conversion, normalization
- **Feature extraction** : manual when prior knowledge, feature selection dimensionality reduction
- **Model estimation** : classification, regression, clustering.
- **Validation** : model and parameter selection.
- **Analysis** : performance, uncertainty, interpretation of the model.

Features extraction, selection and model estimation can be done simultaneously (deep learning, sparse models).

Find your ML method



https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

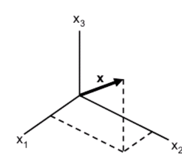
Overview of MAP654I

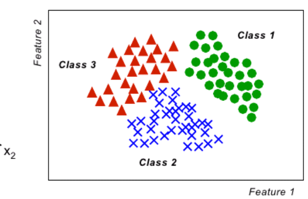
1. **Data and Machine Learning problems**
   - Data properties and visualization
   - Pre-processing
   - Finding your Machine Learning problem
2. **Unsupervised learning**
   - Clustering
   - Density estimation and generative modeling
   - Dictionary learning and collaborative filtering
   - Dimensionality reduction and manifold learning
3. **Supervised learning**
   - Bayesian decision and Nearest neighbors
   - Linear models nonlinear methods for regression and classification
   - Trees, forest and ensemble methods
4. **Validation and interpretation**
   - Performance measures
   - Models and parameter selection (validation)
   - Interpretation of the methods

Overview for the current part

Data description

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$





Vector data

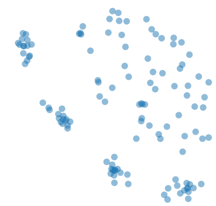
- A feature is a distinct trait, or detail of an object.
- An object is represented as a combination of features *i.e.* a vector $\mathbf{x}$ of dimensionality d.
- The space of size d is called the representation/feature space (often $\mathbb{R}^d$).

Dataset

- An ensemble of objects is often denoted as a data set.
- The individuals objects in a dataset are called examples or samples since they are often supposed to be realizations of probability distributions.
- Samples can be represented as points in this space. This representation is called scatter plot and is usually used for 2D and 3D data.

Unsupervised dataset

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{id} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{nd} \end{bmatrix}$$

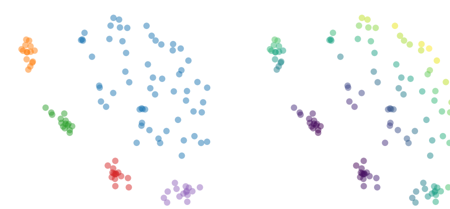


Unsupervised learning

- The dataset contains the samples $\{\mathbf{x}_i\}_{i=1}^n$ where n is the number of samples of size d.
- d and n define the dimensionality of the learning problem.
- Data stored as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$ contains the transposed training samples as lines (features are columns).
- Note: in the course we use 1-based indexing as standard in math but in Python 0-based indexing is used.

Supervised dataset

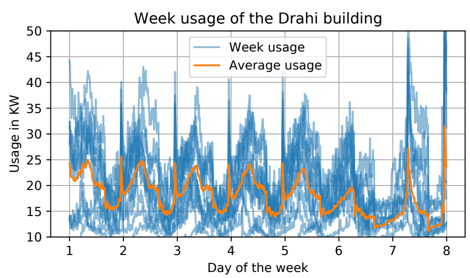
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$



Supervised learning

- The dataset contains the samples $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i$ is the feature sample and $y_i \in \mathcal{Y}$ its label.
- The values to predict (label) can be concatenated in a vector $\mathbf{y} \in \mathcal{Y}^n$
- Prediction space $\mathcal{Y}$ can be:
  - $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{1, \dots, m\}$ for classification problems.
  - $\mathcal{Y} = \mathbb{R}$ for regression problems ($\mathbb{R}^p$ for multi-output regression).
  - Structured for structured prediction (graphs,...).
- Scatter plots for supervised data (`plt.scatter`) use color for the label.

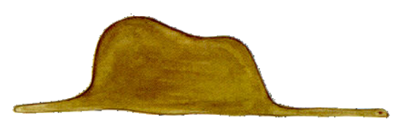
Example of real life dataset

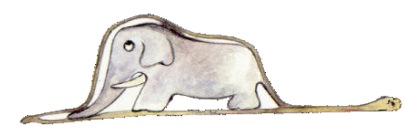


Electrical usage of the Drahi X-Novation Center

- Demonstrator of Energy4Climate of IP Paris.
- Recording of the electrical usage of the building during 1.5 years.
- Can be completed by weather measurement (linked to energy usage).
- Data is a temporal signal that will be used in the following for:
  - Clustering (classification of week usage)
  - Dimensionality reduction (visualization of week usage)
  - Regression/Classification (prediction of usage in the next 24 hours).
- Note that some pre-processing of the data is necessary before getting the unsupervised or supervised datasets.

The Python in the room





Python/Numpy (https://numpy.org/doc/ [Harris et al., 2020])

- Python/Numpy will be used in this course and practical sessions.
- The numerical data will be stored in `np.array` objects.
- We will suggest the name of the functions to use in the practical sessions.

Other libraries

- Scipy (https://www.scipy.org/docs.html)
- Pandas (https://pandas.pydata.org/docs/)
- Matplotlib (https://matplotlib.org/)
- Seaborn (https://seaborn.pydata.org/)
- Scikit-learn (https://scikit-learn.org/)

Installed by default on Anaconda distributions.

Default import

```python
import numpy as np
import scipy as sp
import pandas as pd
import pylab as pl
import seaborn as sns
```

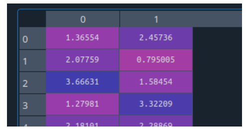
Those modules will be supposed already imported in the course.

Image from "Le petit prince".

Getting to know your data

Basic properties : descriptive statistics

- Look at the arrays (IDE array viewer or print), is the data complete (NaN) ?
- Look at the properties of the features with with `pd.DataFrame(X).describe()`.
- Do the features correspond to physical measurement, are they comparable?
- If very different dynamics (variances and mean/medians) then some pre-processing may be needed.

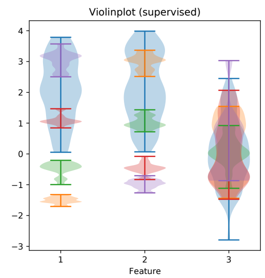


	0	1
0	1.36554	2.45736
1	2.07759	0.795005
2	3.66631	1.58454
3	1.27981	3.32209

```
In [2]: pd.DataFrame(x).describe()
Out[2]:
               0           1
count  120.000000  120.000000
mean     1.297114    1.389048
std      1.625110    1.544902
min     -1.705968   -1.264103
25%     -0.011573    0.030770
50%      1.376426    1.384059
75%      2.783527    2.826683
max      3.786362    3.986649
```

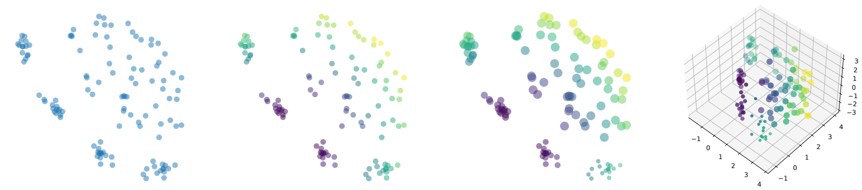
Interpretation: plots

- Compare the features distributions with histograms and violinplots (`pl.violinplot`).
- If images them plot some images (`pl.imshow`).
- If signals of time series then plot some signals (`pl.plot`).
- If unstructured data then use scatterplots (see next slide).
- Dimensionality reduction can be necessary.



Visualizing your data with scatterplot

2D 3D (2D+color) 4D (2D+color+size) 5D (3D+color+size)

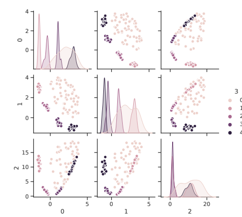


Scatterplot `pl.scatter(X[:,0],X[:,1],c=X[:,2],s=X[:,3])`

- Can be use to see relations between features (and labels) in small dimensions.
- Size (`s=`) and colors (`c=`) of the points can be used to go beyond 2D.
- Sometimes dimensionality reduction is necessary (see next course).

Scatterplot matrix `sns.pairplot(df,hue=key)`

- Provided by Seaborn but requires `pd.DataFrame` data (color label set with `hue=`).
- Pairwise 2D scatterplots between features.
- Visualization of pairwise relationships between features and the target label.



Preprocessing and feature extraction

Objective

Process (transform) the raw data input so that the ML methods will have better performances.

Classical approaches

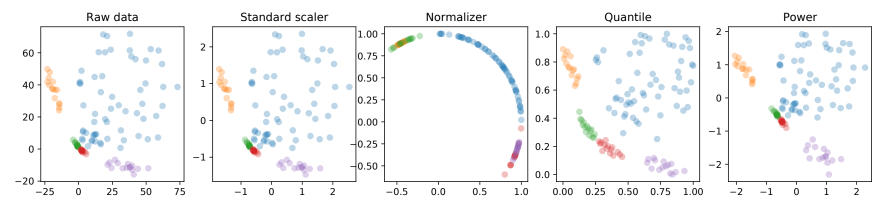
- Scaling (standard, unit, min/max), nonlinear mapping of features
- Data imputation (missing data)
- Encoding (from text or categorical to vector)
- Features selection (prior knowledge, experts, automated)
- Filtering (temporal and spatial, denoising)
- Dimensionality reduction (low dimensional modeling and visualization)

Most unsupervised learning methods can be used for feature extraction.

Warning

- Sensitivity to outliers (`sklearn.preprocessing.RobustScaler`).
- Transformation needed on new data for supervised learning (out-of-sample).
- Invertible transformations allows for better interpretability.

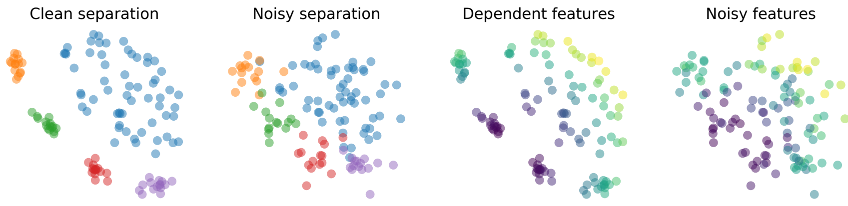
Examples of pre-processing



Preprocessing methods from Scikit-learn [Pedregosa et al., 2011]

- **Standard scaler** `sklearn.preprocessing.StandardScaler()`
  Remove mean and divide by standard deviation for all features (no visible change in scatterplot).
- **Normalizer** `sklearn.preprocessing.Normalizer()`
  Scale individual samples to have a unit norm (projection on the hypersphere).
- **Transform to uniform distribution** `sklearn.preprocessing.QuantileTransformer()`
  Nonlinear mapping of each feature to a uniform distribution.
- **Transform to Normal distribution** `sklearn.preprocessing.PowerTransformer()`
  Nonlinear mapping of each feature to a Normal distribution.

Feature extraction



What are "good" features in supervised learning ?

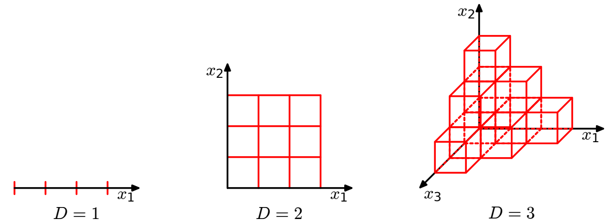
The quality of a feature depends on the learning problem.

- **Classification** Samples from the different classes should be separated in the feature space (clustered classes are simpler to discriminate).
- **Regression** The position in the feature space should help determining the value to predict (correlation or at least non-independence with the value to predict).

How to perform feature extraction?

- **Manually** : prior knowledge of the data, research literature, existing toolboxes.
- **Automatically** : feature extraction part of the ML model (deep neural network, feature explorations).

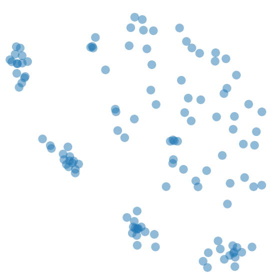
Dimensionality



Curse of dimensionality

- Datasets have n of samples of dimensionality d.
- In order to have a constant sampling of the space, the required number of samples n is exponential in d.
- In high dimension it is easy to overfit a model (predict well on training data but fail on new data).
- High dimensional statistics study the performances in this case.
- Main reason for simple models (linear) and dimensionality reduction.
- State of the art model on Imagenet (14M training images, 469x387 pixels in average) on september 2021 has 2 Billion parameters [Zhai et al., 2021].

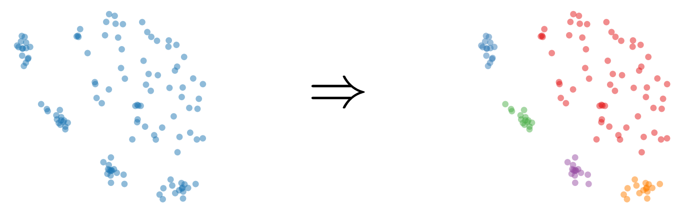
Unsupervised learning, data description/exploration



Different objectives

- **Clustering** : $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow \{\hat{y}_i\}_{i=1}^n$ where $\hat{y}$ is the labels of a group.
- **Probability density estimation** : $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow \hat{p}(\mathbf{x})$.
- **Generative modeling** : $\{\mathbf{x}_i\}_{i=1}^n \Rightarrow G(\mathbf{z})$ such that $p(G(\mathbf{z})) \approx p(\mathbf{x})$ with $\mathbf{z} \sim N(0, \sigma^2)$.
- **Dimensionality reduction** : $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n \Rightarrow \{\tilde{\mathbf{x}}_i \in \mathbb{R}^p\}_{i=1}^n$ with $p \ll d$.

Clustering



Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \{\hat{y}_i\}_{i=1}^n$$

- Organize training examples in groups: Find the labels $\hat{y}_i \in \mathcal{Y} = \{1, \ldots, K\}$.
- Optional : Find a clustering function $\hat{f}(\mathbf{x}) \in \mathcal{Y}$ that can cluster new samples.

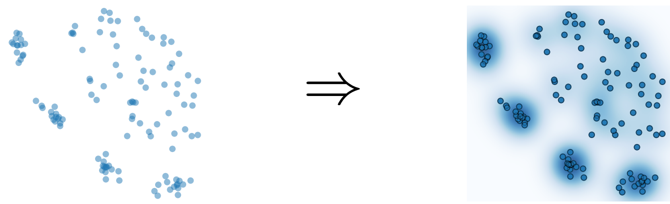
Parameters

- K number of classes.
- Similarity measure between samples.
- Minimal distance between clusters.

Methods

- K-means.
- Gaussian mixtures.
- Spectral clustering.
- Hierarchical clustering.

Probability density estimation



Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \hat{p}$$

- Estimate a probability density $\hat{p}(\mathbf{x})$ from the IID samples in the data.
- Probability density : $\hat{p}(\mathbf{x}) \geq 0,\ \forall \mathbf{x}$ and $\int \hat{p}(\mathbf{x})d\mathbf{x} = 1$.
- Optional : generate new data from $\hat{p}(\mathbf{x})$.

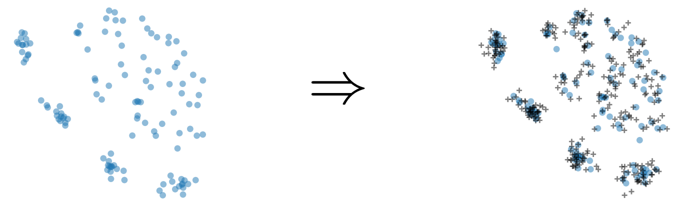
Parameters

- Type of distribution (Histogram, Gaussian, ...).
- Parameters of the law (μ, Σ)

Methods

- Histogram (1D/2D).
- Parzen/kernel density estimation.
- Gaussian mixture.

Generative modeling



Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \hat{g} \text{ such that } p(\hat{g}(\mathbf{z})) \approx p(\mathbf{x}) \text{ with } \mathbf{z} \sim \mathcal{N}$$

- Estimate a mapping function $\hat{g}(\mathbf{z}) \in \mathbb{R}^d$ that generates similar samples to $\{\mathbf{x}_i\}_{i=1}^n$.
- Latent variable $\mathbf{z}$ follows a known Normal or Uniform distribution.
- Optional : recover an estimation of $\hat{p}(\mathbf{x})$ using the change of variable formula.

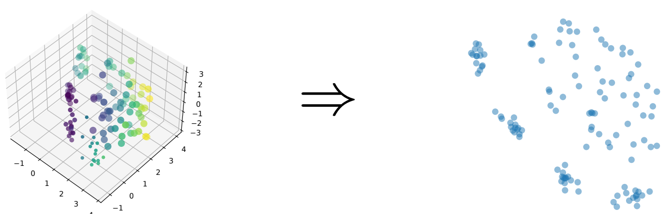
Parameters

- Type of distribution for $\mathbf{z}$ (Gaussian, uniform, ...).
- Type of function for g.

Methods

- PCA (Gaussian data).
- Gen. Adversarial Networks (GAN)
- Variational Auto-Encoders (VAE)
- Diffusion models

Dimensionality reduction



Objective

$$\{\mathbf{x}_i\}_{i=1}^n \quad \Rightarrow \quad \{\tilde{\mathbf{x}}_i \in \mathbb{R}^p\}_{i=1}^n \text{ with } p \ll d$$

- Project the data into a low dimensional space of size $p \ll d$.
- Preserve the information in the data (class, subspace, manifold).
- Optional : Learning a projection function $\hat{m} : \mathbb{R}^d \to \mathbb{R}^p$ for new data.

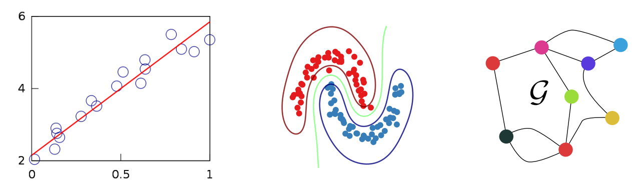
Parameters

- Type of projection (linear, nonlinear).
- Assumptions about the data (subspace, manifold).
- Similarity between samples.

Methods

- Feature selection.
- Principal Component Analysis (PCA).
- Dictionary learning, ICA.
- Non-linear dimensionality reduction (MDS, tSNE, Auto-Encoder)

Supervised learning



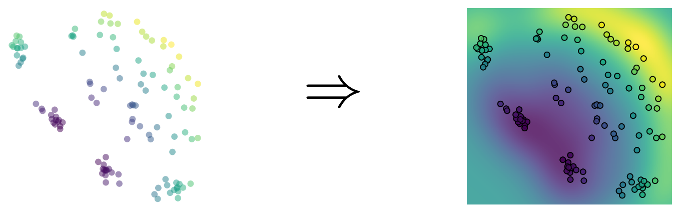
Objective

- Training dataset : $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with observations $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathcal{Y}$.
- Train a function $f(\cdot) : \mathbb{R}^d \to \mathcal{Y}$ on the dataset.

Types of supervised prediction

- **Classification** $f(\cdot)$ predicts a class (discrete output) either binary $\mathcal{Y} = \{-1, 1\}$ or multi-class $\mathcal{Y} = \{1, \dots, K\}$.
- **Regression** $f(\cdot)$ predicts a continuous value ($\mathcal{Y} = \mathbb{R}$) or several ($\mathcal{Y} = \mathbb{R}^p$).
- **Structured prediction** $f(\cdot)$ predicts a structured object (graph, tree, molecule) (not discussed in detail here).

Regression



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

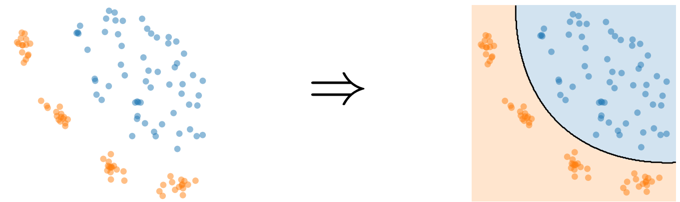
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Least Square (LS).
- Ridge regression, Lasso.
- Kernel regression.
- Deep learning.

Binary classification



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value ($\mathcal{Y} = \{-1, 1\}$).
- In practice, train a continuous function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ and predict with $\text{sign}(f)$.
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.
- Optional: provide uncertainty information such as probabilities of each class.

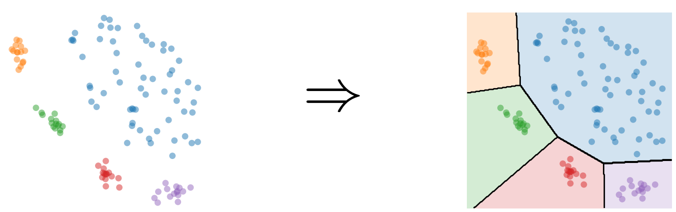
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Bayesian classifier (LDA, QDA)
- Linear and kernel discrimination
- Decision trees, random forests.
- Deep learning.

Multiclass classification



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \{1, \ldots, K\}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting an integer value ($\mathcal{Y} = \{1, \ldots, K\}$).
- In practice K continuous score functions f_k are estimated and the prediction is

$$f(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$$

- Softmax can be used instead of argmax to get probability estimates.

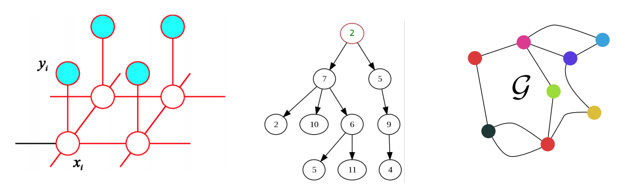
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Bayesian classifier (LDA, QDA)
- Linear and kernel discrimination
- Decision trees, random forests.
- Deep learning.

Structured learning and prediction



Objective

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathcal{X} \rightarrow \mathcal{Y}$$

- Train a prediction function $f(\mathbf{x}) = \mathbf{y} \in \mathcal{Y}$ on stuctured data.
- The structure prediction function is often expressed as

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \tilde{f}(\mathbf{x}, \mathbf{y})$$

- Both $\mathcal{X}$ and $\mathcal{Y}$ can be spaces of structured data (graph, sequence, tree).

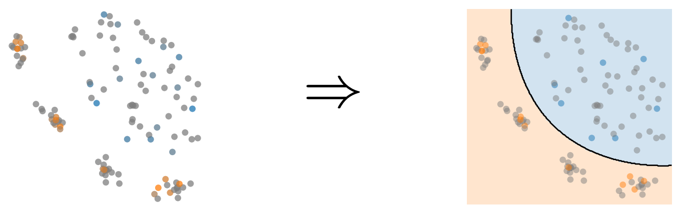
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- How to find the max value in $\mathcal{Y}$.

Methods

- Structured Support Vector Machine (SSVM)
- Conditional Random Fields (CRF)
- Convolutional Graph Networks.

Semi-supervised learning



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^m, \{\mathbf{x}_i\}_{i=m+1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \mathcal{Y}$$

- Train a prediction function $f(\mathbf{x}) = y \in \mathcal{Y}$ from partially labeled data.
- Only $m < n$ labeled samples out of the n total samples.

Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Assumption on label propagation.

Methods

- Low-density separation
- Laplacian regularization
- Heuristic approaches
- Generative models

Other common ML problems

Multi-task learning

$$\{\mathbf{x}_i^t, y_i^t\}_{i=1}^n, \ \forall t \in \{1, \dots, T\} \quad \Rightarrow \quad f_t : \mathcal{X} \to \mathcal{Y}^t, \ \forall t$$

- Train simultaneously T functions f_t and share information between the tasks.

Domain Adapation (unsupervised)

$$\{\mathbf{x}_i^s, y_i^s\}_{i=1}^{n_s}, \{\mathbf{x}_i^t\}_{i=1}^{n_t} \quad \Rightarrow \quad f_t : \mathcal{X} \to \mathcal{Y}, \ \forall t$$

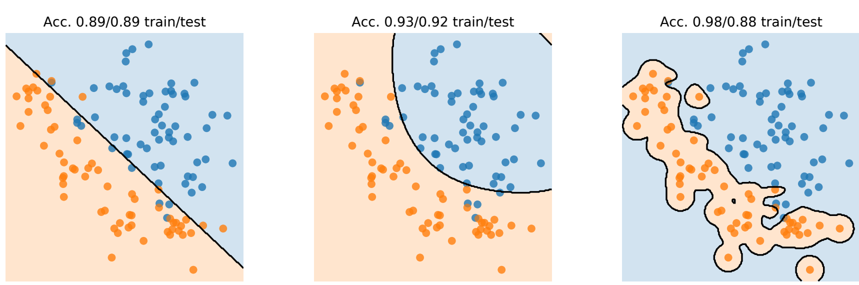
- Train a function f_t on unlabeled target data $\{\mathbf{x}_i^t\}_{i=1}^{n_t}$ and related but different labeled source data $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^{n_s}$.
- Variants include Multi-Source DA (MSDA) and semi supervised DA (few labels available in target).

Transfer Learning

$$\tilde{f}, \{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathcal{X} \to \mathcal{Y}, \ \forall t$$

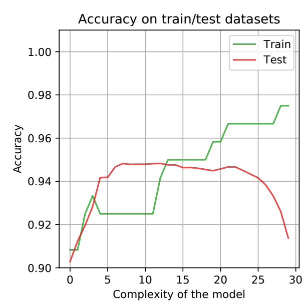
- Train a function f on dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ using a model $\tilde{f}$ already trained on another tasks (benefit from other training experience)

Generalization of a model

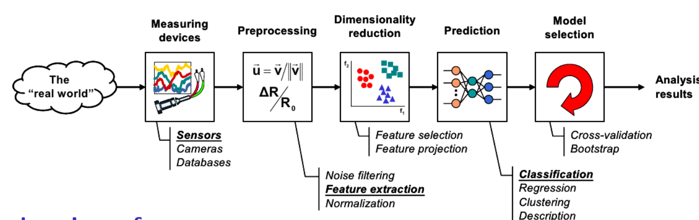


Complexity of a model

- Under-fitting when the model is too simple.
- Over-fitting occurs when the model is too complex (memorization, bad students remember only the training samples).
- Training data performance is not a good proxy for testing performance.
- We want to predict well on new data!
- Parameter and model validation.



General references for this course



Machine learning references

- Elements of statistical learning (free PDF online) [Friedman et al., 2001].
- Pattern recognition and machine learning [Bishop Christopher et al., 2006].
- Deep learning (`https://www.deeplearningbook.org/`) [Goodfellow et al., 2016].
- Probabilistic Machine Learning (`https://probml.github.io/`) [Murphy, 2022]
- ML course of Andrew Ng (free on Coursera and Youtube).

Applied mathematics

- Linear algebra [Petersen et al., 2008] [Golub and Van Loan, 1996].
- Convex Optimization [Boyd et al., 2004] (Free PDF online).
- Statistics [Wasserman, 2013].

Numerical Python

- All documentations.
- `https://scipy-lectures.org/`
- Google and `https://stackoverflow.com/`.

Machine learning

`https://xkcd.com/`

References I

[Bishop Christopher et al., 2006] Bishop Christopher, M. et al. (2006).
Pattern recognition and machine learning.
Information science and statisticsNew York: Springer.

[Boyd et al., 2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004).
Convex optimization.
Cambridge university press.

[Forbes, 2016] Forbes, F. (2016).
Modelling structured data with probabilistic graphical models.
EAS Publications Series, 77:195–219.

[Friedman et al., 2001] Friedman, J., Hastie, T., Tibshirani, R., et al. (2001).
The elements of statistical learning, volume 1.
Springer series in statistics New York.

[Golub and Van Loan, 1996] Golub, G. H. and Van Loan, C. F. (1996).
Matrix computations. johns hopkins studies in the mathematical sciences.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

References II

[Harris et al., 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020).
Array programming with NumPy.
Nature, 585(7825):357–362.

[Hunter, 2007] Hunter, J. D. (2007).
Matplotlib: A 2d graphics environment.
Computing in Science & Engineering, 9(3):90–95.

[Murphy, 2022] Murphy, K. P. (2022).
Probabilistic machine learning: an introduction.
MIT press.

[pandas development team, 2020] pandas development team, T. (2020).
pandas-dev/pandas: Pandas.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011).
Scikit-learn: Machine learning in python.
the Journal of machine Learning research, 12:2825–2830.

References III

[Petersen et al., 2008] Petersen, K. B., Pedersen, M. S., et al. (2008).
The matrix cookbook.
Technical University of Denmark, 7(15):510.

[Virtanen et al., 2020] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020).
SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.
Nature Methods, 17:261–272.

[Waskom, 2021] Waskom, M. L. (2021).
seaborn: statistical data visualization.
Journal of Open Source Software, 6(60):3021.

[Wasserman, 2013] Wasserman, L. (2013).
All of statistics: a concise course in statistical inference.
Springer Science & Business Media.

[Zhai et al., 2021] Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021).
Scaling vision transformers.
arXiv preprint arXiv:2106.04560.