Machine learning for graphs and with graphs

Course 1: introduction

Titouan Vayer & Pierre Borgnat
email: titouan.vayer@inria.fr, pierre.borgnat@ens-lyon.fr

September 9, 2024

ENS DE LYON

CR09: Machine learning for graphs and with graphs

From theory …

1. Basics of machine learning
2. The graph framework
3. Community detection/ graph clustering
4. Graph signal processing
5. Kernels for graphs
6. Graph neural networks
7. Optimal transport for graphs
8. Learning graphs from (unstructured) data

Full description:

`https://tvayer.github.io/courses/coursegraph.html`

… to practice

We will use `Python` and various libraries

Some references for machine learning

Shai Shalev-Shwartz and Shai Ben-David (2014). *Understanding Machine Learning - From Theory to Algorithms*.

Francis Bach (2022). *Learning Theory from First Principles*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*.

Evaluation

- 50 % oral presentation on a selected research article.
- 50 % code associated to the article applied on real data.
- Bonus.

Python installations

- The practical sessions of the course will require to run jupyter notebooks.
- We recommend that you install python through the Anaconda distribution (python 3.7, 3.8 or 3.9 is preferrable) available at `https://www.anaconda.com/products/distribution`

You should check that you are able to create and open a jupyter notebook, and inside, run the following imports:

```python
import matplotlib
import numpy
import sklearn
import pytorch
import pandas
import scipy
```

If any of these packages is missing, it can be installed with ‘conda install numpy‘, the command being run in a terminal or in Anaconda prompt for Windows user.

Basics of machine learning

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions
- Empirical risk minimization

Model selection and validation

- Split your dataset !

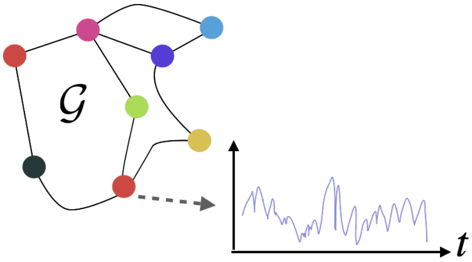
The problems with structured data

- Motivating examples
- A primer on graph theory
- Why “classical ML” struggles with stuctured data

What is machine learning ?

Some applications

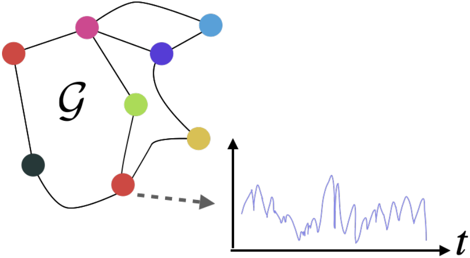
1. Energy networks, disease propagation



What is machine learning ?

Some applications

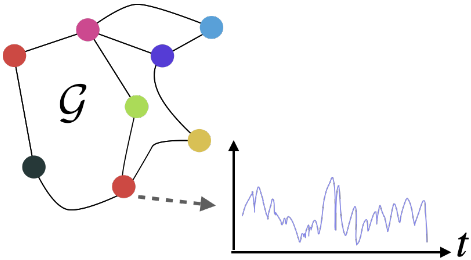
1. Energy networks, disease propagation
2. Image analysis (medical application, web)

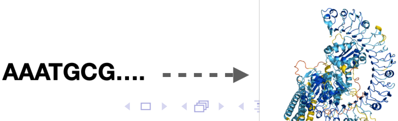


What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding Jumper et al. 2021

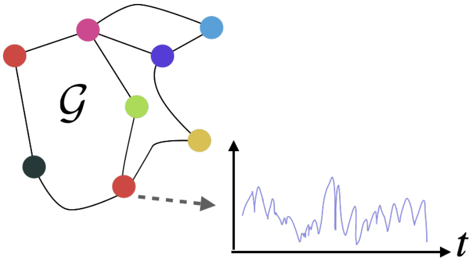


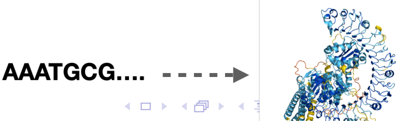


What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding Jumper et al. 2021
4. Generative models `https://stablediffusionweb.com/`

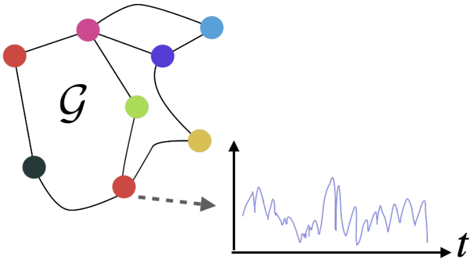


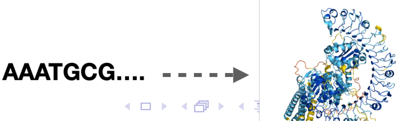


What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding Jumper et al. 2021
4. Generative models `https://stablediffusionweb.com/`
5. Natural language processing `https://chat.openai.com/chat`

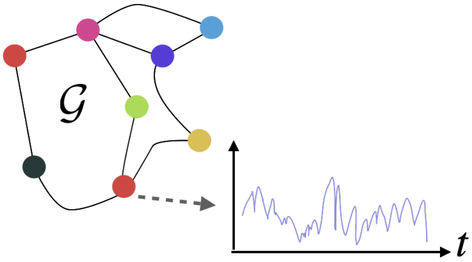




What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding Jumper et al. 2021
4. Generative models https://stablediffusionweb.com/
5. Natural language processing https://chat.openai.com/chat
6. For art https://www.youtube.com/watch?v=MwtVkPKx3RA

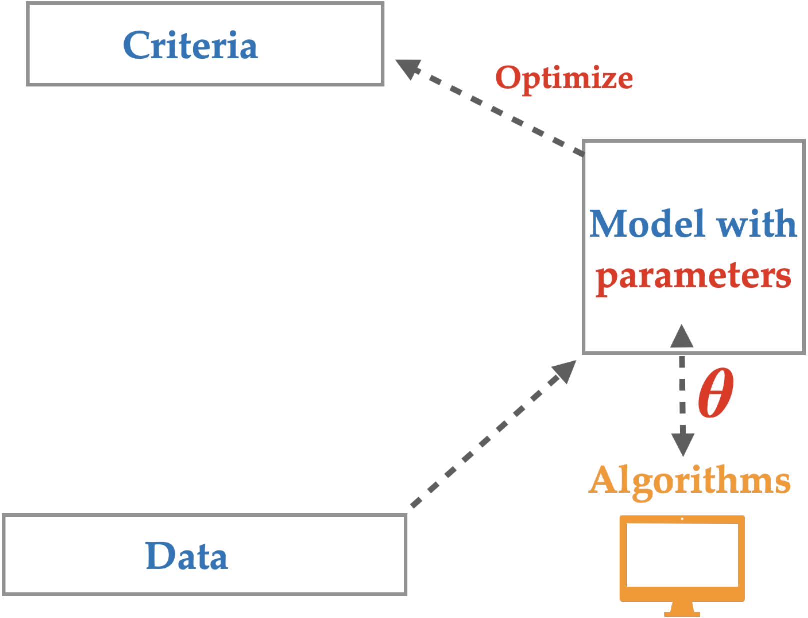


What is machine learning ?

Criteria

Data

What is machine learning ?



What is machine learning ?

The objective of machine learning

Teach a machine to process automatically some data in order to solve a given problem.

What is machine learning ?

The objective of machine learning

Teach a machine to process automatically some data in order to solve a given problem.

Unsupervised learning: understanding the data

- Clustering & probability density estimation
- Dimensionality reduction

What is machine learning ?

The objective of machine learning

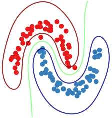
Teach a machine to process automatically some data in order to solve a given problem.

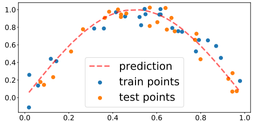
Unsupervised learning: understanding the data

- Clustering & probability density estimation
- Dimensionality reduction

Supervised learning: learning to predict

- Classification: classify points according to some labels
- Regression: predict real (vector) values





Some images and slides have been obtained by the courtesy of Rémi Flamary

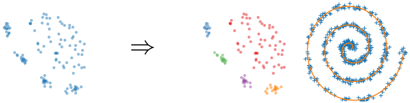
What is machine learning ?

The objective of machine learning

Teach a machine to process automatically some data in order to solve a given problem.

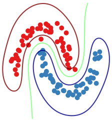
Unsupervised learning: understanding the data

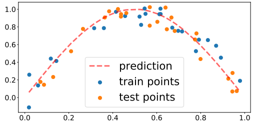
- Clustering & probability density estimation
- Dimensionality reduction



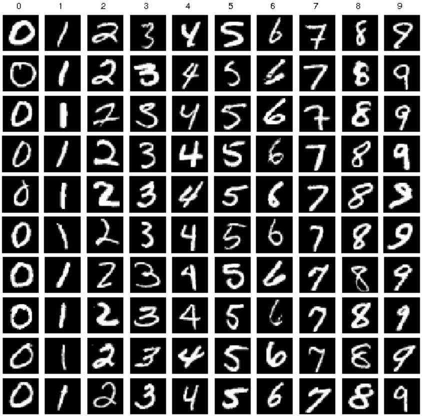
Supervised learning: learning to predict

- Classification: classify points according to some labels
- Regression: predict real (vector) values





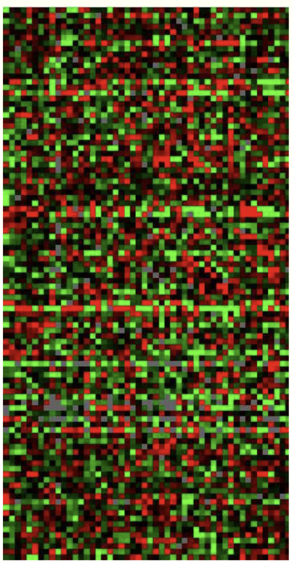
What is machine learning ?



Supervised classification examples

- e.g. to identify the numbers on images from a 16×16 gray level image (image classification)
- SPAM, fraud detection, disease classification ...

What is machine learning ?



Clustering example

- Analyse n sequences (individuals) of d genetical responses
- Groups of similar samples ? Gene with similar expressions ?

Plan

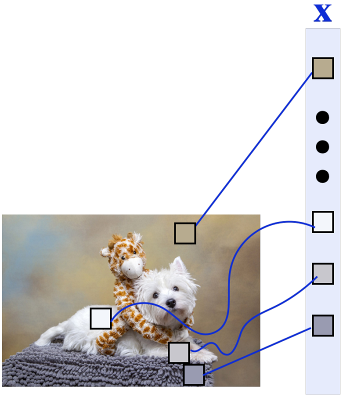
Store a data point

Vectorial representation

One “sample”, “data point”, “individual”:

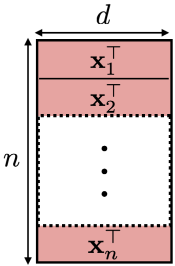
$$\mathbf{x} = (x_1, \cdots, x_d)^\top \in \mathbb{R}^d$$

- d is the dimension, x_i is the ith information i of $\mathbf{x}$
- Can describe information about an individual
- For an image $\mathbf{x}$: each pixel of an image
- Descriptors of a cell, word embedding ...



Unsupervised dataset

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$



Unsupervised learning

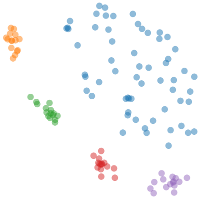
- The dataset contains the samples $(\mathbf{x}_i)_{i=1}^n$ where n is the number of samples of size d.
- d and n define the dimensionality of the learning problem.
- Data stored as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ that contains the training samples as rows.

Supervised dataset

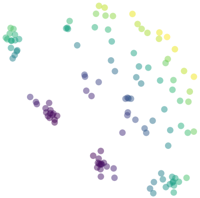
Samples + labels:

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

Classification



Regression



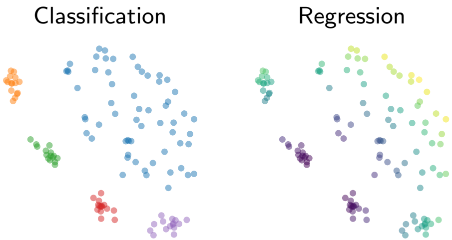
Supervised learning

- 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 $\mathbf{y} \in \mathcal{Y}^n$

Supervised dataset

Samples + labels:

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



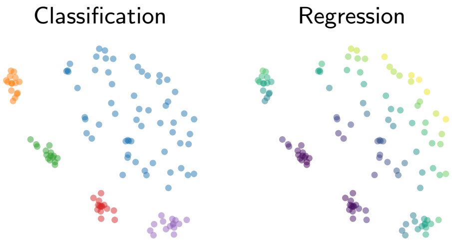
Supervised learning

- 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 $\mathcal{Y} = \{1, \dots, K\}$ for classification problems.
  - $\mathcal{Y} = \mathbb{R}$ for regression problems ($\mathbb{R}^p$ for multi-output regression).

Supervised dataset

Samples + labels:

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



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 $\mathcal{Y} = \{1, \ldots, K\}$ for classification problems.
  - $\mathcal{Y} = \mathbb{R}$ for regression problems ($\mathbb{R}^p$ for multi-output regression).
- Semi-supervised learning: few labeled points are available, but a large number of unlabeled points are given.

Plan

From training data to prediction

Training data

- We have access to n samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n) \sim p$
- $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the data distribution
- p is unknown ! We only have access to samples.

From training data to prediction

Training data

- We have access to n samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n) \sim p$
- $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the data distribution
- p is unknown ! We only have access to samples.
- For unsupervised problem we only have $\mathbf{x}_1, \cdots, \mathbf{x}_n \sim p$ and $p \in \mathcal{P}(\mathcal{X})$

From training data to prediction

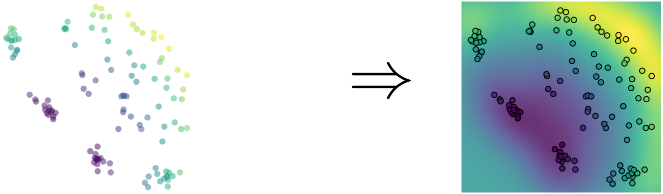
Training data

- We have access to n samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n) \sim p$
- $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the data distribution
- p is unknown ! We only have access to samples.
- For unsupervised problem we only have $\mathbf{x}_1, \cdots, \mathbf{x}_n \sim p$ and $p \in \mathcal{P}(\mathcal{X})$

Objective

- We have a task to solve: classification, regression, clustering ...
- Most ML problems formulate as **finding some function f that "best" solves our task**
- f is called **an hypothesis** and is **implemented by a computer**
- Most of the time f depends on some parameter $\boldsymbol{\theta} \in \Theta$

Regression

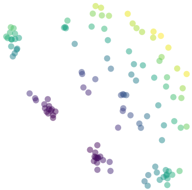


Objective

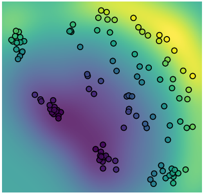
$$(\mathbf{x}_i, y_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- 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 ($\mathcal{Y} = \mathbb{R}^p$).

Regression



$\Longrightarrow$



Objective

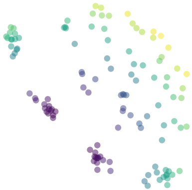
$$(\mathbf{x}_i, y_i)_{i=1}^{n} \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \mathbb{R}$$

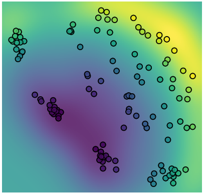
- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

Hyperparameters

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

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 ($\mathcal{Y} = \mathbb{R}^p$).

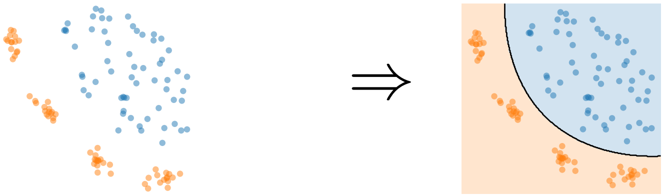
Hyperparameters

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

Methods

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

Binary classification

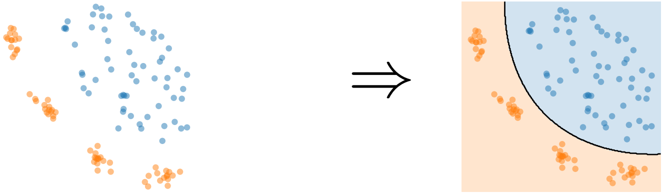


Objective

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

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value ($\mathcal{Y} = \{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

Binary classification



Objective

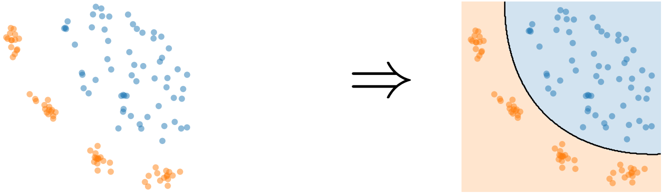
$$(\mathbf{x}_i, y_i)_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value ($\mathcal{Y} = \{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

Hyperparameters

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

Binary classification



Objective

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

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value ($\mathcal{Y} = \{-1, 1\}$).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

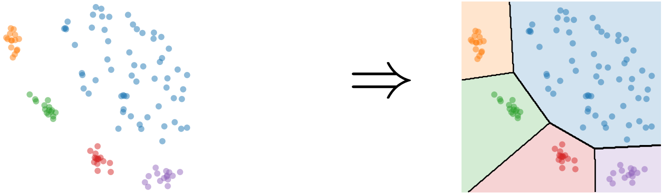
Hyperparameters

- 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



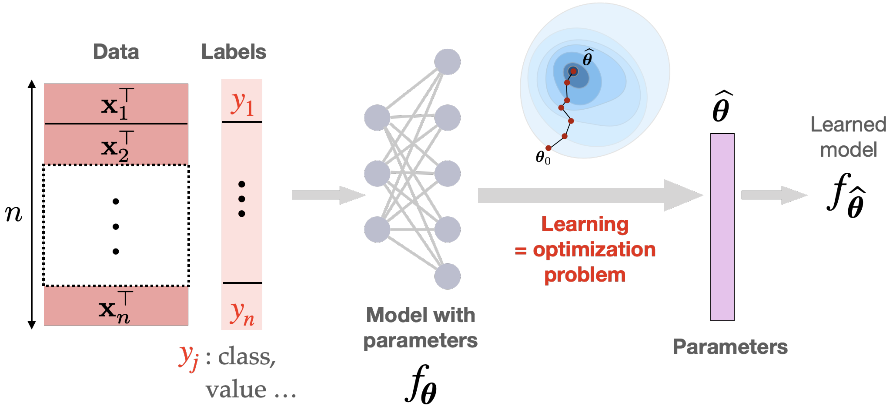
Objective

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

- 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.

The big picture of (parametrized) ML

But how to find this function ?



The goal in the **learning step** will be to find the parameters $\widehat{\theta}$ (hence the function) that **minimizes a measure of error on the data**

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ so that:

$$\ell(\text{true value}\ , \text{predicted value}) = \text{ how good is my prediction}$$

Regression problems

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ so that:

$$\ell\,(\text{true value}\ , \text{predicted value}) = \ \text{how good is my prediction}$$

Regression problems

- E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ so that:

$$\ell\,(\text{true value}\;,\text{predicted value}) = \;\text{how good is my prediction}$$

Regression problems

- E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ so that:

$$\ell\,(\text{true value}\ ,\text{predicted value}) = \ \text{how good is my prediction}$$

Regression problems

- E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)
- E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_q^q$ (ℓ_q loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ so that:

$$\ell\,(\text{true value}\ ,\text{predicted value}) = \ \text{how good is my prediction}$$

Regression problems

- E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)
- E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_q^q$ (ℓ_q loss)

Classification problems

- E.g. $y_i \in \{-1, 1\}$ $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i \neq f(\mathbf{x}_i)}$ (0/1 loss)

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

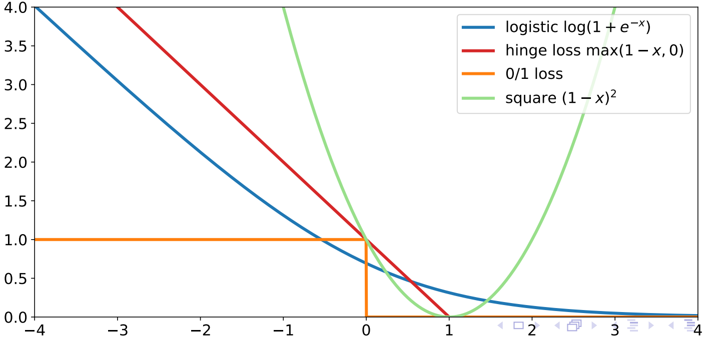
- $y_i f(\mathbf{x}_i)$ is **the margin** (on the board).
- $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ (0/1 loss)
- $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ (hinge loss: **SVM**)
- $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ (logistic loss)

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

- $y_i f(\mathbf{x}_i)$ is **the margin** (on the board).
- $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ (0/1 loss)
- $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ (hinge loss: **SVM**)
- $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ (logistic loss)



Empirical risk minimization

Train by minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) \tag{ERM}$$

Empirical risk minimization

Train by minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) \tag{ERM}$$

- It is called **empirical risk minimization (ERM)**
- Given the loss, finds the "best" f on the training data
- Same idea applies for unsupervised problem (minimizing reconstruction error)

Empirical risk minimization

Train by minimizing the train error

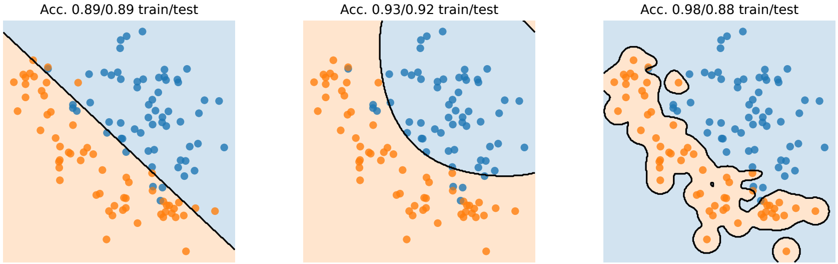
To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, f(\mathbf{x}_i)) \tag{ERM}$$

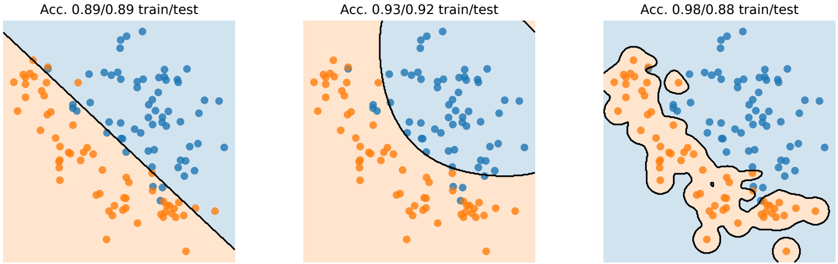
- It is called **empirical risk minimization (ERM)**
- Given the loss, finds the "best" f on the training data
- Same idea applies for unsupervised problem (minimizing reconstruction error)

Once solved how do I know if my model is good ?

Underfitting and overfitting

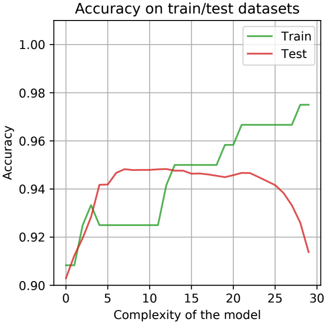


Underfitting and overfitting



Complexity of a model

- Under-fitting when the model is too simple.
- Over-fitting occurs when the model is too complex
- Training data performance is not a good proxy for testing performance.
- We want to predict well on new data!
- Parameter and model validation.



Empirical risk minimization

Train by minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \lambda \, \mathrm{Reg}(f) \qquad \text{(ERM)}$$

- It is called **empirical risk minimization (ERM)**
- Given the loss, finds the "best" f on the training data
- Teacher/student analogy
- Same idea applies for unsupervised problem

... but we want generalization !

- We want f to be good outside the training samples
- Add regularization ! (limit the complexity of f)

Plan

Model selection and validation

Bias-complexity tradeoff

$$\text{generalization error} \;=\; \text{estimation error} \;+\; \text{approximation error}$$

Select a model that is not too complex but not too simple !

Model selection and validation

Bias-complexity tradeoff

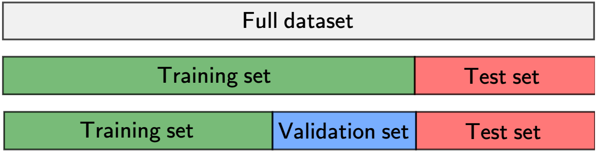
$$\text{generalization error} = \text{estimation error} + \text{approximation error}$$

Select a model that is not too complex but not too simple !

General principle

- Estimate the generalization error **on data not seen during training**
- Is a rough estimate of the test error
- Select the model with the lowest "approximate" test error

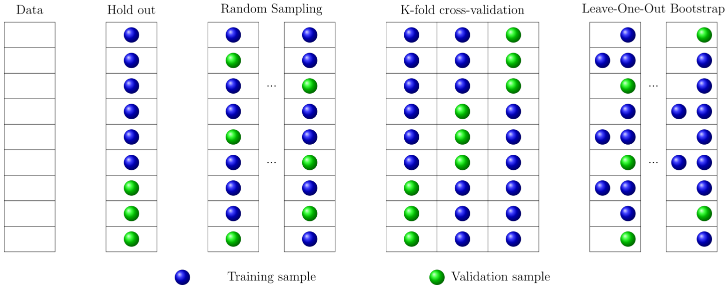
Splitting the data



Principle of Hold-Out cross-validation

- Split the training data in a training and validation sets (non overlapping).
- Train different models (different methods and/or hyperparameters) on the train data.
- Evaluate performance on the validation data and select the method/parameters with best performance.
- **Validation set acts as a proxy of test data**
- But only one split is a poor proxy !

Different ways to split the data



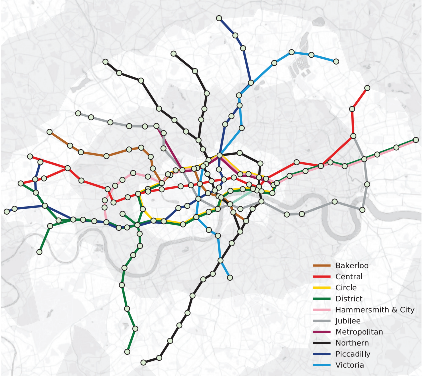
Data splitting for cross-validation Arlot and Celisse 2010

- The training data is split in non-overlapping training/validation sets.
- **Hold-Out** uses a unique split.
- More robust cross-validation approaches actually investigate several splits of the data and compute the average performance:
  - **K-fold** (split in K sets and use one split as test for all k)
  - Random sampling (aka **Shuffle split**) draws several random splittings.
- Scikit-learn implementation : `sklearn.model_selection.cross_validate`

Plan

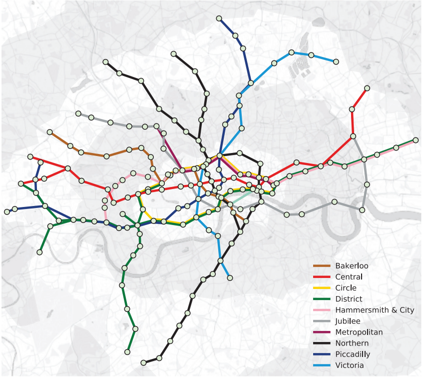
Motivating examples

- Traffic forecasting (e.g. ETA estimation): GNN for Google Maps Derrow-Pinion et al. 2021.



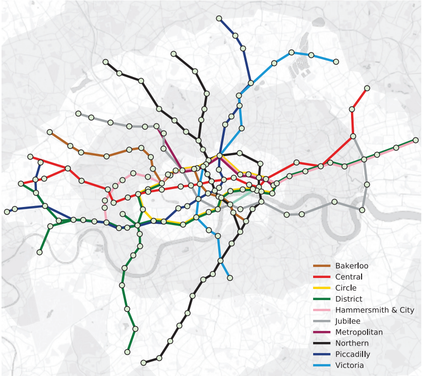
Motivating examples

- Traffic forecasting (e.g. ETA estimation): GNN for Google Maps Derrow-Pinion et al. 2021.
- Chemistry and Drug Design: space of chemically synthesisable molecules is very large (estimated around 10^{60}).
- Drug Repositioning: action of drugs and their interactions $\rightarrow$ graph Barabási, Gulbahce, and Loscalzo 2011.



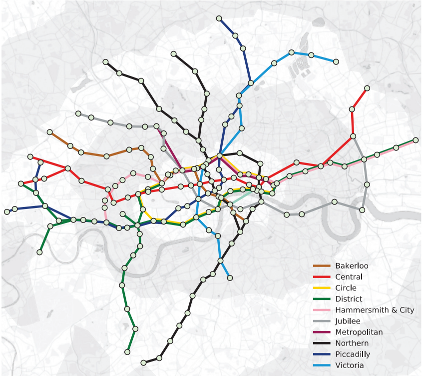
Motivating examples

- Traffic forecasting (e.g. ETA estimation): GNN for Google Maps Derrow-Pinion et al. 2021.
- Chemistry and Drug Design: space of chemically synthesisable molecules is very large (estimated around 10^{60}).
- Drug Repositioning: action of drugs and their interactions $\rightarrow$ graph Barabási, Gulbahce, and Loscalzo 2011.
- Protein biology Jumper et al. 2021.



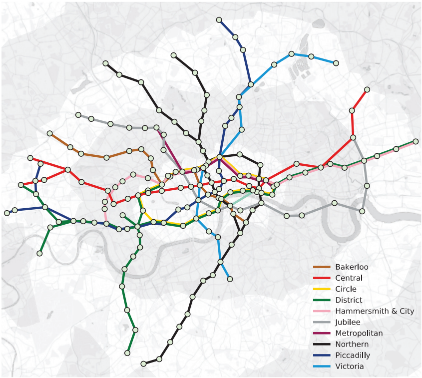
Motivating examples

- Traffic forecasting (e.g. ETA estimation): GNN for Google Maps Derrow-Pinion et al. 2021.
- Chemistry and Drug Design: space of chemically synthesisable molecules is very large (estimated around 10^{60}).
- Drug Repositioning: action of drugs and their interactions → graph Barabási, Gulbahce, and Loscalzo 2011.
- Protein biology Jumper et al. 2021.
- Recommender Systems and Social Networks.
- Healthcare.



Motivating examples

- Traffic forecasting (e.g. ETA estimation): GNN for Google Maps Derrow-Pinion et al. 2021.
- Chemistry and Drug Design: space of chemically synthesisable molecules is very large (estimated around 10^{60}).
- Drug Repositioning: action of drugs and their interactions $\rightarrow$ graph Barabási, Gulbahce, and Loscalzo 2011.
- Protein biology Jumper et al. 2021.
- Recommender Systems and Social Networks.
- Healthcare.
- and more...



What is a graph ?

Definition

A graph $G = (V, E)$ is defined as a set of **vertices** V, which are connected by a set of **edges** $E \subset V \times V$.

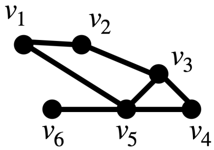
What is a graph ?

Definition

A graph $G = (V, E)$ is defined as a set of **vertices** V, which are connected by a set of **edges** $E \subset V \times V$.

- Example of **undirected** graph

$G = (V, E)$



Adjacency matrix

$$\mathbf{A} = \begin{array}{c c} & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{array}{c} v_1 \\ \\ \\ \\ \\ \\ \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} = \mathbf{A}^\top$$

Adjacency matrix

The adjacency $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ is defined as

$$[\mathbf{A}]_{ij} = \begin{cases} 1 \text{ if } (v_i, v_j) \in E (\text{often noted as } v_i \sim v_j) \\ 0 \text{ otherwise} \end{cases}$$

What is a graph ?

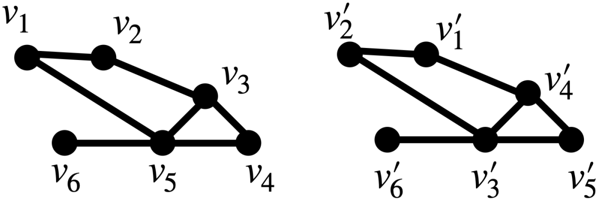
Definition

A graph $G = (V, E)$ is defined as a set of **vertices** V, which are connected by a set of **edges** $E \subset V \times V$.

Isomorphic graphs

The definition depends on the ordering of the nodes.

$$G = (V, E) \qquad G' = (V', E')$$



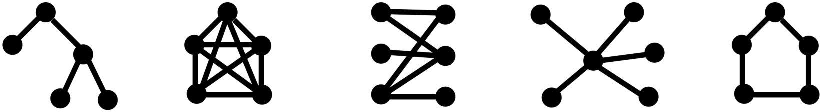
$$\mathbf{A} = \mathbf{P}^\top \, \mathbf{A}' \, \mathbf{P}$$

What is a graph ?

Definition

A graph $G = (V, E)$ is defined as a set of **vertices** V, which are connected by a set of **edges** $E \subset V \times V$.

Some special structures

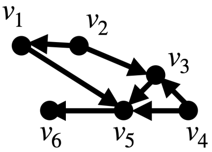


Tree Complete graph Bipartite graph Star graph Circular graph

What is a graph ?

- Example of **directed** graph

$$G = (V, E)$$



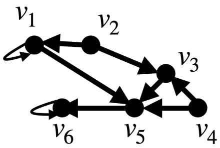
Adjacency matrix

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \neq \mathbf{A}^\top$$

What is a graph ?

- Example of **directed** graph with **self-loops**.

$$G = (V, E)$$



Adjacency matrix

$$\mathbf{A} = \begin{array}{cc} & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{array}{c} v_1 \\ \\ \\ \\ \\ \\ \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \neq \mathbf{A}^\top$$

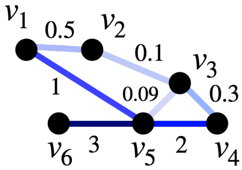
What is a graph ?

Weighted graph

A weighted graph $G = (V, E)$ associates <u>non-negative</u> weights to each edge.

- Example of **weighted** graph

$$G = (V, E)$$



Weight matrix

$$\mathbf{W} = \begin{pmatrix} 0 & 0.5 & 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0.3 & 0.09 & 0 \\ 0 & 0 & 0.3 & 0 & 2 & 0 \\ 1 & 0 & 0.09 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix}$$

What is a graph ?

Degree of a node

The degree of a node v_i is

$$d_i = |\{v \in V : v \sim v_i\}| = \sum_{j=1}^{|V|} A_{ij}$$

The degree matrix is $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_{|V|})$.

What is a graph ?

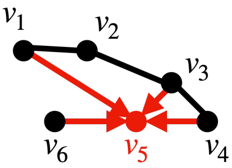
Degree of a node

The degree of a node v_i is

$$d_i = |\{v \in V : v \sim v_i\}| = \sum_{j=1}^{|V|} A_{ij}$$

The degree matrix is $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_{|V|})$.

$$G = (V, E)$$



Adjacency matrix

Degree

$$\mathbf{A} = \begin{array}{cc} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ v_1 & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad \begin{pmatrix} 2 \\ 2 \\ 3 \\ 2 \\ 4 \\ 1 \end{pmatrix} = \mathbf{A1}$$

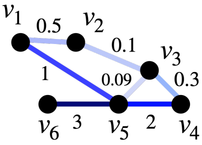
What is a graph ?

Degree of a node

The degree of a node v_i in a weighted graph is

$$d_i = \sum_{j=1}^{|V|} W_{ij}$$

$$G = (V, E)$$



$$\mathbf{W} = \underbrace{\begin{pmatrix} 0 & 0.5 & 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0.3 & 0.09 & 0 \\ 0 & 0 & 0.3 & 0 & 2 & 0 \\ 1 & 0 & 0.09 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix}}_{\text{Weight matrix}} \quad \underbrace{\begin{pmatrix} 1.5 \\ 0.6 \\ 0.49 \\ 2.3 \\ 4.09 \\ 3 \end{pmatrix}}_{\text{Degree}}$$

What is a graph ?

Laplacian matrix

The Laplacian matrix of a undirected graph is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \text{ where } \mathbf{D} \text{ is the degree matrix}$$

Properties

On the board

What is a graph ?

Laplacian matrix

The Laplacian matrix of a undirected graph is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \text{ where } \mathbf{D} \text{ is the degree matrix}$$

Properties

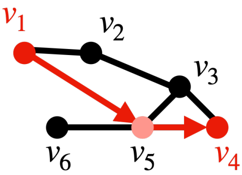
- $\forall \mathbf{x} \in \mathbb{R}^n, \; \mathbf{x}^\top \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{ij} W_{ij}(x_i - x_j)^2$.
- $\mathbf{L}$ is PSD.
- $\ker(\mathbf{L}) = \{c\mathbf{1}_n ; c \in \mathbb{R}\}$.

What is a graph ?

Shortest-path matrix

The shortest-path between $v, v' \in V$ is the path that connects v, v' such that the sum of the weights of its constituent edges is minimized.

$$G = (V, E)$$



Shortest-path matrix

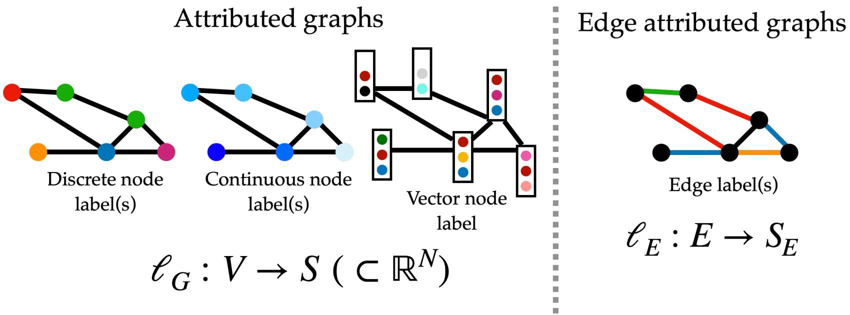
$$\mathbf{S} = \begin{pmatrix} 0 & 1 & 2 & 2 & 1 & 2 \\ 1 & 0 & 1 & 2 & 2 & 3 \\ 2 & 1 & 0 & 1 & 1 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 1 & 1 & 0 & 1 \\ 2 & 3 & 2 & 2 & 1 & 0 \end{pmatrix}$$

- Dijkstra's algorithm computes all the shortest paths from a single node in $O(|E| + |V| \log(|V|))$.
- All-pairs shortest paths with Floyd–Warshall algorithm in $O(|V|^3)$.

What is a graph ?

Attributed graphs

Most graphs encountered in ML also have **attributes**.



ML vs structured data

Problems

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?
- Can we build ML methods with the raw representation of G ? How to adapt ML methods that work on vectors ?

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?
- Can we build ML methods with the raw representation of G ? How to adapt ML methods that work on vectors ?
- How can we handle the combinatoric nature of graphs ?

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?
- Can we build ML methods with the raw representation of G ? How to adapt ML methods that work on vectors ?
- How can we handle the combinatoric nature of graphs ?
- ML outputs should be permutation invariant ? equivariant ?

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?
- Can we build ML methods with the raw representation of G ? How to adapt ML methods that work on vectors ?
- How can we handle the combinatoric nature of graphs ?
- ML outputs should be permutation invariant ? equivariant ?
- When data $=$ vectors in one graph how can we take into account the structure of the graph ?

ML vs structured data

Problems

- Can we encode a graph G as a vector $\in \mathbb{R}^d$ to use standard ML methods ?
- Can we build ML methods with the raw representation of G ? How to adapt ML methods that work on vectors ?
- How can we handle the combinatoric nature of graphs ?
- ML outputs should be permutation invariant ? equivariant ?
- When data = vectors in one graph how can we take into account the structure of the graph ?
- When data = vectors can we find an (interesting) graph that represent these data ?

References I

- Arlot, Sylvain and Alain Celisse (2010). "A survey of cross-validation procedures for model selection". In: *Statistics Surveys* 4.
- Bach, Francis (2022). *Learning Theory from First Principles*.
- Barabási, Albert-László, Natali Gulbahce, and Joseph Loscalzo (2011). "Network medicine: a network-based approach to human disease". In: *Nature reviews genetics* 12.1, pp. 56–68.
- Derrow-Pinion, Austin et al. (2021). "Eta prediction with graph neural networks in google maps". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3767–3776.
- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*.
- Jumper, John et al. (2021). "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873, pp. 583–589.
- Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding Machine Learning - From Theory to Algorithms*.