

# Unsupervised and Semi-Supervised learning

# Overview - Un/Semi/Fully Supervised learning

Assume we have data

$$\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]$$

.

The data can be

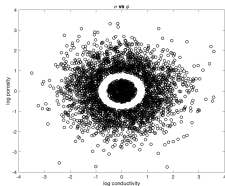
- ▶ Images
- ▶ Text
- ▶ Sound
- ▶ Set of numbers (climate, pressure ...)

We can think of (at least) 3 goals

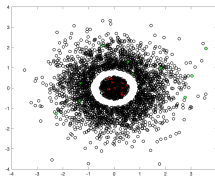
- ▶ Cluster the data (unsupervised)
- ▶ Give meaning to each cluster, label it (semisupervised)
- ▶ Find a functional relation between the cluster and its label (supervised)

# Overview - Un/Semi/Fully Supervised learning

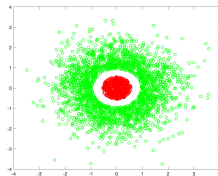
Example - rock conductivity versus porosity data



Unsupervised



Semisupervised



Supervised

- ▶ How many types of rocks we have: **Unsupervised learning**
- ▶ What are their names (Granite, Basalt): **Semisupervised learning**
- ▶ Given  $\sigma, \phi$  can we find a function  $f(\sigma, \phi) = \text{rock type}$ : **Supervised learning**

# Overview - Un/Semi/Fully Supervised learning

- ▶ Supervised learning requires a large labeled data set
- ▶ Gives an "explanation" (a model) between data and label
- ▶ Un/Semisupervised is more modest
- ▶ No model, just label
- ▶ Can be followed by supervised learning

# Overview - Un/Semi/Fully Supervised learning



- ▶ What kind of cars are in the picture
- ▶ What is their colour
- ▶ What is their size
- ▶ Which direction are they going

## Types of learning

- ▶ Unsupervised learning may cluster cars in irrelevant manner
- ▶ Semi-supervised we label a few cars and ask the computer the label the rest
- ▶ In supervised learning someone labeled all the cars

# Overview - Un/Semi Supervised learning

In this module we focus on semisupervised learning and a bit on unsupervised learning.

Main questions

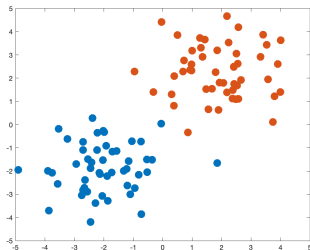
- ▶ Given a data set cluster the data into a few groups
- ▶ Assuming that a few data are labeled, label the rest

# Un/Semisupervised learning: General principle

## Main assumption:

Given the data set  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]$ , if  $\mathbf{y}_i$  is "close" to  $\mathbf{y}_j$  then they belong to the same class.

What is close?



# Measuring closeness - metrics

The question is how to measure the closeness of two vectors in our space.

Assume a function  $D(\mathbf{x}, \mathbf{y}) \rightarrow [0, \infty)$  with the following properties

- ▶  $D(\mathbf{x}, \mathbf{y}) \geq 0$
- ▶  $D(\mathbf{x}, \mathbf{y}) = 0 \rightarrow \mathbf{x} = \mathbf{y}$
- ▶  $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$
- ▶  $D(\mathbf{x}, \mathbf{z}) \leq D(\mathbf{x}, \mathbf{y}) + D(\mathbf{y}, \mathbf{z})$

In many cases we will not keep all the above but will keep most of them



# Measuring closeness - metrics

Most obvious metric based on a norm

$$D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p = \left( \sum |\mathbf{x}_i - \mathbf{y}_i|^p \right)^{\frac{1}{p}}$$

- ▶ Most used is the 2-norm (why?)
- ▶  $p = \infty$  is called the max norm (why?)

A simple modification weighted norms

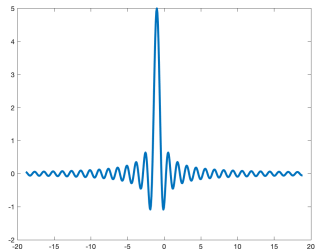
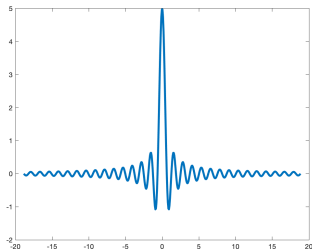
$$D(\mathbf{x}, \mathbf{y}) = \left( \sum \mathbf{w}_i |\mathbf{x}_i - \mathbf{y}_i|^p \right)^{\frac{1}{p}}$$

where  $\mathbf{w} > 0$  is a vector of positive weights.

Allow us to focus on some parts of the vectors and scale it if needed

# Measuring closeness - metrics

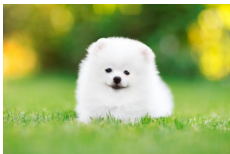
Are these signals similar?



Measure is very sensitive to translation

# Measuring closeness - metrics

Which of these images are closer



# Measuring closeness - metrics

Many different definitions of distance based on the application

- ▶ Normed distance
- ▶ Hamming distance - for strings
- ▶ Wasserstein metric - for probabilities (also applied to images)
- ▶ Riemannian metrics - for data that "lives" on manifolds

Choosing the right distance is the key for the application

In many cases - chicken and egg. If we know the right distance then we know how to cluster.

We will use  $L_2$  distance for conveniency.

# Using metrics - graphical models

A graph is



A vertex set  $\mathcal{V}$

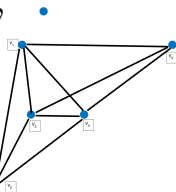


# Using metrics - graphical models

A graph is



A vertex set  $\mathcal{V}$

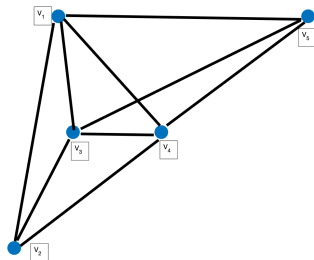


Edge set  $\mathcal{E}$

# The adjacency matrix

The adjacency matrix is defined as

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if there is an edge} \\ 0 & \text{otherwise} \\ 0 & \text{if } i = j \end{cases}$$

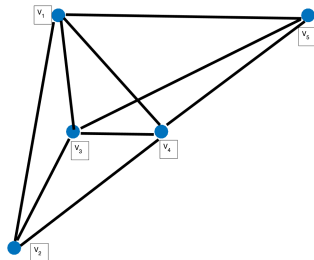


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

# The graph Laplacian

The degree matrix

$$\mathbf{D} = \text{diag} \left( \sum_j \mathbf{A}_{ij} \right)$$



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$



# The graph Laplacian

The graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{pmatrix}$$

Note

$$(\mathbf{L}\mathbf{v})_i = \sum_{i \neq j} \mathbf{v}_i - \mathbf{v}_j \quad \text{and} \quad \mathbf{v}^\top \mathbf{L} \mathbf{v} = \sum_{e_{ij}} (\mathbf{v}_i - \mathbf{v}_j)^2$$

# The weighted graph Laplacian

In the Laplacian above every neighbour gets the same weight. Define the weighted Laplacian, where each edge  $e_{ij}$  is associated with a weight  $0 \leq w_{ij}$ .

$$(\mathbf{L}\mathbf{v})_i = \sum_{i \neq j} \mathbf{w}_{ij}(\mathbf{v}_i - \mathbf{v}_j) \quad \text{and} \quad \mathbf{v}^\top \mathbf{L}\mathbf{v} = \sum_{e_{ij}} \mathbf{w}_{ij}(\mathbf{v}_i - \mathbf{v}_j)^2$$

Typical choice for  $\mathbf{w}$  is

$$\mathbf{w}_{ij} = \exp\left(-\frac{\|\mathbf{v}_i - \mathbf{v}_j\|^2}{\sigma}\right)$$

More appropriate choice

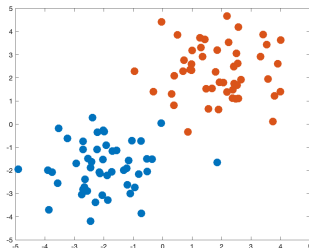
$$\mathbf{w}_{ij} = \exp\left(-\frac{D(\mathbf{v}_i, \mathbf{v}_j)}{\sigma}\right)$$

# The eigenvalues and eigenvectors of the Laplacian

The smallest eigenvalue is always 0 with eigenvector

$\mathbf{v} = [1, \dots, 1]^\top$  (why)

The second to smallest eigenvalue that is not zero is the Fiedler eigenvalue.



It gets the (approximate) value 1 on the first group and  $-1$  (approximately) on the second group.

# The eigenvalues and eigenvectors of the Laplacian

The second eigenvalue is the solution to the optimization problem

$$\begin{array}{ll} \min_{\mathbf{u} \neq \gamma \mathbf{e}} & \frac{1}{2} \mathbf{u}^\top \mathbf{L} \mathbf{u} \\ \text{subject to} & \|\mathbf{u}\| = 1 \end{array}$$

## Interpretation

Try to find the non-constant vector with the minimal energy  
The solution should be "smooth" on the graph

# The eigenvalues and eigenvectors of the Laplacian

The second eigenvalue is the solution to the optimization problem

$$\begin{array}{ll} \min_{\mathbf{u} \neq \gamma \mathbf{e}} & \frac{1}{2} \mathbf{u}^\top \mathbf{L} \mathbf{u} \\ \text{subject to} & \|\mathbf{u}\| = 1 \end{array}$$

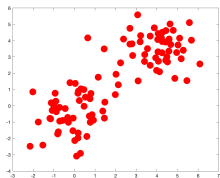
- ▶ For small problems use eig solver of a dense matrix
- ▶ For large problem use iterative methods - inverse iteration, Krylov methods, randomized linear algebra

Power method

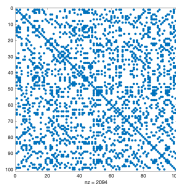
$$\mathbf{v}_k = \mathbf{L}^{-\dagger} \mathbf{u}_k \quad \mathbf{u}_{k+1} = \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|}$$

# The eigenvalues and eigenvectors of the Laplacian

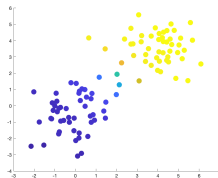
## Example



Data



Graph Laplacian

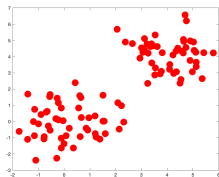


Fidler Vector

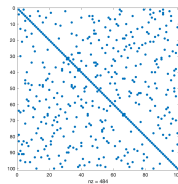
# The eigenvalues and eigenvectors of the Laplacian

Can be very sensitive to parameters

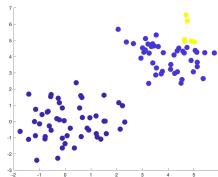
- ▶ Number of neighbors
- ▶ Choice of  $\sigma$
- ▶ Distance measure



Data



Graph Laplacian



Fidler Vector

# From unsupervised to semisupervised learning

- ▶ In most cases it is possible to obtain a few labeled data
- ▶ Goal is to label the whole data given the labeled data
- ▶ Typically, results are much better and more robust than unsupervised learning

Assumption: we are given a set  $\mathcal{S}$  where  $\mathbf{p}_{\mathcal{S}}^{\text{obs}}$  is the probability of each datum to belong to each class



# From unsupervised to semisupervised learning

## Incorporating probabilities - the softmax function

- ▶  $\mathbf{p}^{\text{obs}}$  is a probability,  $0 \leq \mathbf{p}_i^{\text{obs}} \leq 1$  and  $\sum \mathbf{p}^{\text{obs}} = 1$ .
- ▶  $\mathbf{u}$  is unbounded
- ▶ Define  $\mathbf{p}(\mathbf{u}) = \frac{\exp(\mathbf{u})}{\sum \exp(\mathbf{u})}$
- ▶ Use cross entropy to compare probabilities,  $\mathbf{p}(\mathbf{u})$  and  $\mathbf{p}^{\text{obs}}$

$$\ell_{\text{oss}}(\mathbf{u}, \mathbf{p}^{\text{obs}}) = -\frac{1}{n} \mathbf{p}_{\text{obs}}^{\top} \log(\mathbf{p}(\mathbf{u}))$$

# From unsupervised to semisupervised learning

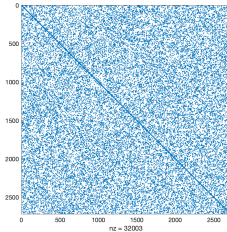
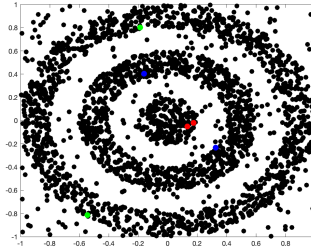
**New goal** - Fit the given labels **and** keep the vector smooth.

$$\min_{\mathbf{u}} \mathcal{E}(\mathbf{u}) = \text{loss}(\mathbf{u}, \mathbf{p}^{\text{obs}}) + \frac{\alpha}{2} \mathbf{u}^{\top} \mathbf{L} \mathbf{u}$$

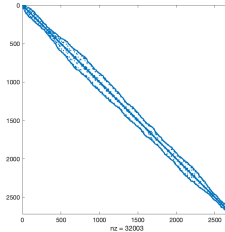
- ▶  $\alpha$  tradeoff parameter
- ▶ Can be estimated using cross validation
- ▶ Solution using inexact Newton's method

# From unsupervised to semisupervised learning

## Example - classifying nonlinear data

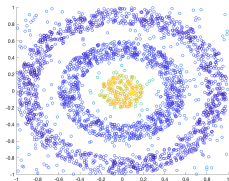


Graph Lap

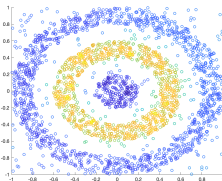


Reordered Graph Lap

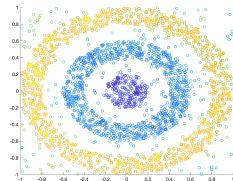
# From unsupervised to semisupervised learning



Class 1



Class 2



Class 3

# Semisupervised learning - challenges

- ▶ Choice of distance function
- ▶ Choice of hyper parameters
- ▶ Naive complexity  $n^2$
- ▶ Sparse linear algebra, preconditioning, eigenvalue solvers