Machine Learning on Networks

Esteban Bautista-Ruiz, Lionel Tabourier

LIP6 – CNRS and Université Pierre et Marie Curie

first_name.last_name@lip6.fr

November 16, 2021

Introduction

- Motivation
- Machine Learning on Regular Domains
- Machine Learning on Irregular Domains

Neural Networks

- The multi-layer perceptron

Machine Learning on Graphs

- Spectral Clustering
- Graph Convolutional Networks

Motivation

The big data era

- **36 million** web-sites created per minute
- **188 million** emails sent per minute
- **55 thousand** photos posted per minute
- **5.5 million** videos watched per minute
- **4.3 billion** people connected to Internet

Motivation

The big data era

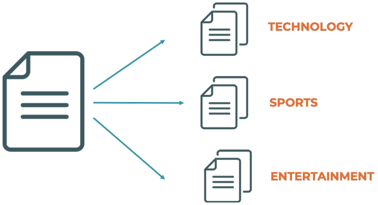
- **36 million** web-sites created per minute
- **188 million** emails sent per minute
- **55 thousand** photos posted per minute
- **5.5 million** videos watched per minute
- **4.3 billion** people connected to Internet

Goal

- **Categorize** websites by topic
- **Categorize** emails as spam or not
- **Categorize** photographies by genre
- **Predict** social trends based on videos watched
- **Predict** political preferences based on navigation patterns

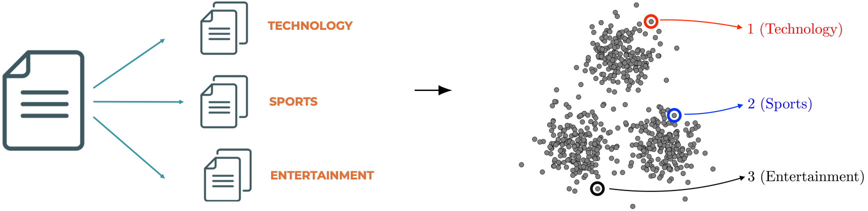
The Goal of Machine Learning

Classification



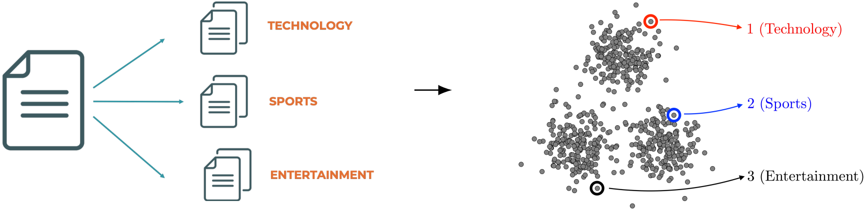
The Goal of Machine Learning

Classification



The Goal of Machine Learning

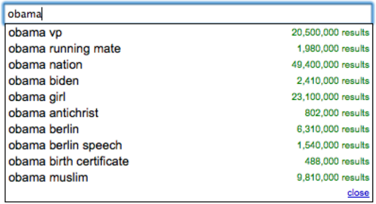
Classification



$$\mathcal{F} : \mathbb{R}^d \to \{1, \ldots, K\}$$

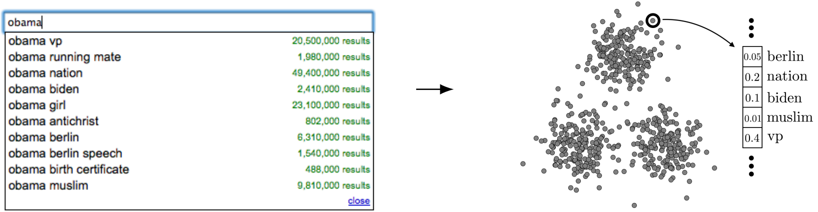
The Goal of Machine Learning

Regression



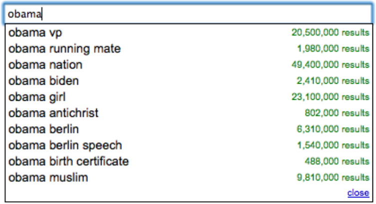
The Goal of Machine Learning

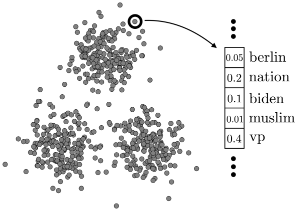
Regression



The Goal of Machine Learning

Regression





$$\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^m$$

The Goal of Machine Learning

The goal in machine learning

How to find $\mathcal{F}$?

The Goal of Machine Learning

The goal in machine learning

How to find $\mathcal{F}$?

Two main approaches:

- Supervised learning : We know $\mathcal{F}$ for some examples

The Goal of Machine Learning

The goal in machine learning

How to find $\mathcal{F}$?

Two main approaches:

- Supervised learning : We know $\mathcal{F}$ for some examples
- Unsupervised learning : We do not have any knowledge about $\mathcal{F}$

Machine Learning on Regular Domains

Supervised Learning

Supervised Learning

The Supervised Learning Paradigm

- **Training dataset**: set of points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $\mathcal{F}$ is known
- **Learning**: use the training dataset to infer $\mathcal{F}$ for the remainder of data

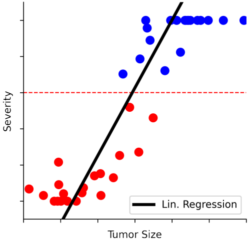
Supervised Learning

The Supervised Learning Paradigm

- **Training dataset**: set of points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $\mathcal{F}$ is known
- **Learning**: use the training dataset to infer $\mathcal{F}$ for the remainder of data

Classical methods

Linear Regression

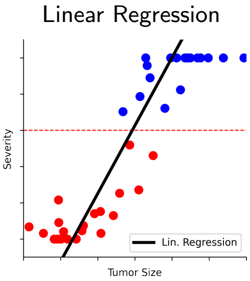


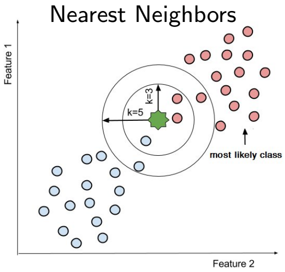
Supervised Learning

The Supervised Learning Paradigm

- **Training dataset**: set of points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $\mathcal{F}$ is known
- **Learning**: use the training dataset to infer $\mathcal{F}$ for the remainder of data

Classical methods





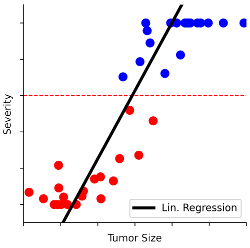
Supervised Learning

The Supervised Learning Paradigm

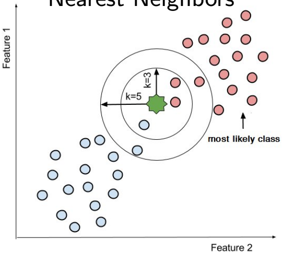
- **Training dataset**: set of points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $\mathcal{F}$ is known
- **Learning**: use the training dataset to infer $\mathcal{F}$ for the remainder of data

Classical methods

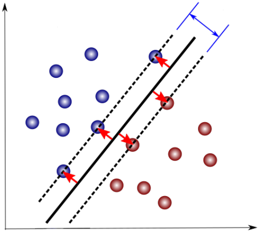
Linear Regression



Nearest Neighbors

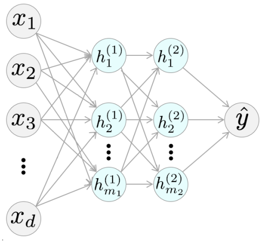


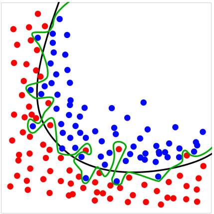
Support Vector Machine

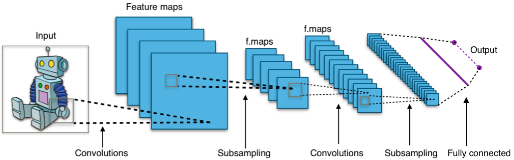


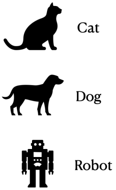
Supervised Learning

Modern methods: Neural networks









Machine Learning on Regular Domains

Unsupervised Learning

Unsupervised Learning

The Unsupervised Learning Paradigm

- No training dataset
- **Learning**: find patterns in the data
- **Hypothesis**: similar points are likely to be of the same category

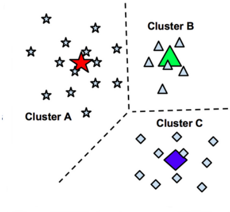
Unsupervised Learning

The Unsupervised Learning Paradigm

- No training dataset
- **Learning**: find patterns in the data
- **Hypothesis**: similar points are likely to be of the same category

Classical methods

K-means



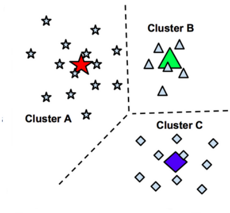
Unsupervised Learning

The Unsupervised Learning Paradigm

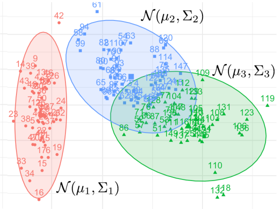
- No training dataset
- **Learning**: find patterns in the data
- **Hypothesis**: similar points are likely to be of the same category

Classical methods

K-means



Mixture models



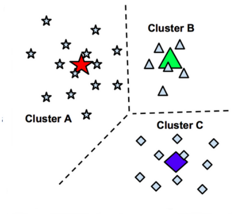
Unsupervised Learning

The Unsupervised Learning Paradigm

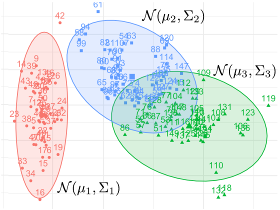
- No training dataset
- **Learning**: find patterns in the data
- **Hypothesis**: similar points are likely to be of the same category

Classical methods

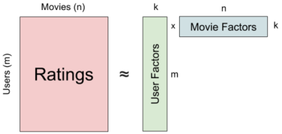
K-means



Mixture models



Matrix factorization



Unsupervised Learning

Modern methods: Autoencoders

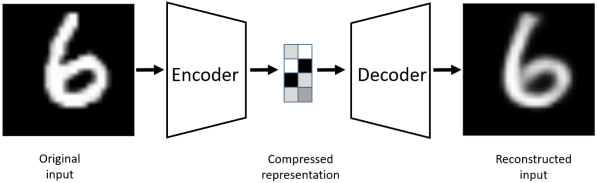


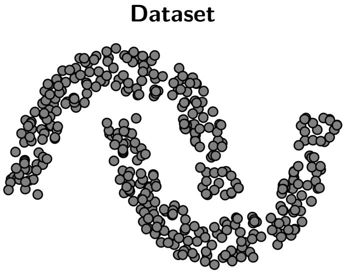
Image taken from [D. Bank et al. Autoencoders. arXiv preprint arXiv:2003.05991]

Machine Learning on Irregular Domains

The Need for Machine Learning on Networks

The Need for Machine Learning on Networks

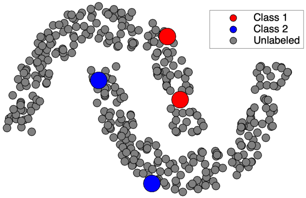
1. **Data often lies in a low dimensional manifold**



The Need for Machine Learning on Networks

1. **Data often lies in a low dimensional manifold**
2. **Labelled data are often expensive to collect**

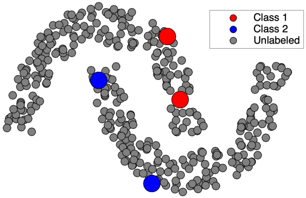
Few training data



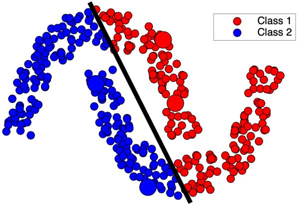
The Need for Machine Learning on Networks

1. **Data often lies in a low dimensional manifold**
2. **Labelled data are often expensive to collect**

Few training data



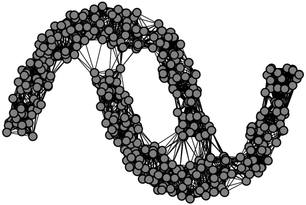
Supervised classifier



The Need for Machine Learning on Networks

1. **Data often lies in a low dimensional manifold**
2. **Labelled data are often expensive to collect**

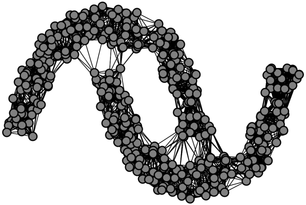
Graph from data



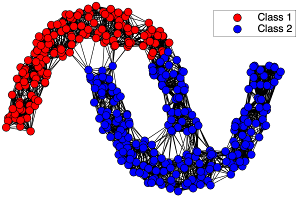
The Need for Machine Learning on Networks

1. **Data often lies in a low dimensional manifold**
2. **Labelled data are often expensive to collect**

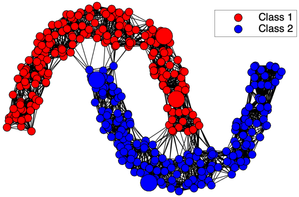
Graph from data



Community detection

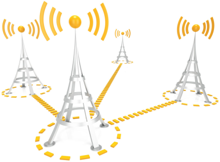


Learning via graph + labels



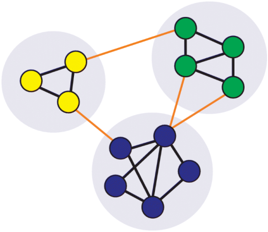
The Need for Machine Learning on Networks

1. **Data often lies in a low dimensional manifold**
2. **Labelled data are often expensive to collect**
3. **Network-structured data are ubiquitous**



Tasks of Interest

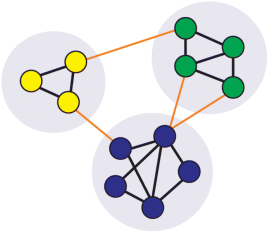
Node classification



- Documents by topic
- Email spam or not
- Power shutdown

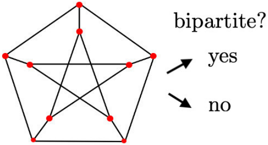
Tasks of Interest

Node classification



- Documents by topic
- Email spam or not
- Power shutdown

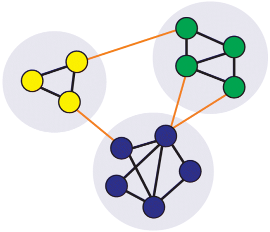
Graph classification



- Graph isomorphism
- Drug synthesis
- Voice recognition

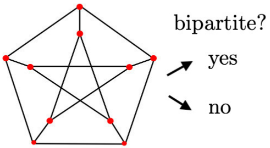
Tasks of Interest

Node classification



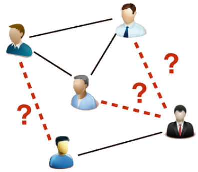
- Documents by topic
- Email spam or not
- Power shutdown

Graph classification



- Graph isomorphism
- Drug synthesis
- Voice recognition

Link prediction



- Predict failures
- Social modeling
- Text generation

Tasks of Interest

In all cases, we still look for a function $\mathcal{F}$. Yet, it is now supported in a graph-based domain instead of a euclidean one.

- Node classification: the domain of $\mathcal{F}$ is the set of vertices
- Graph classification: the domain of $\mathcal{F}$ is the space of all graphs
- Link prediction: the domain of $\mathcal{F}$ is the set of edges

Learning From Graph Data

- How to learn $\mathcal{F}$ supported on graph data?
  - ▷ Cannot use ML models that operate on euclidean domains or regular grids

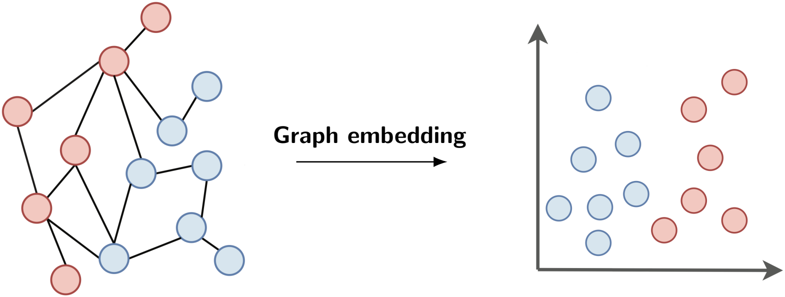
Learning From Graph Data

- How to learn $\mathcal{F}$ supported on graph data?
  - ▷ Cannot use ML models that operate on euclidean domains or regular grids
- Graphs are combinatorial objects where optimization leads to exponential search
  - ▷ Node classification: groups of nodes with small cut
  - ▷ Graph classification: existence of a cycle

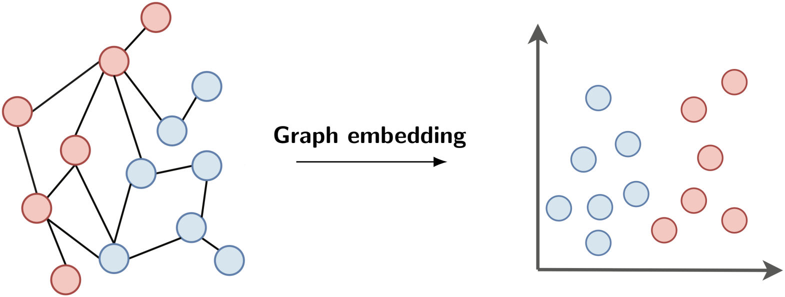
Learning From Graph Data

- How to learn $\mathcal{F}$ supported on graph data?
  - ▷ Cannot use ML models that operate on euclidean domains or regular grids
- Graphs are combinatorial objects where optimization leads to exponential search
  - ▷ Node classification: groups of nodes with small cut
  - ▷ Graph classification: existence of a cycle
- These limitations are relaxed by embedding the graph into an euclidean space
  - ▷ Embeddings can be done via matrix factorization or diffusion processes

Learning From Graph Data



Learning From Graph Data

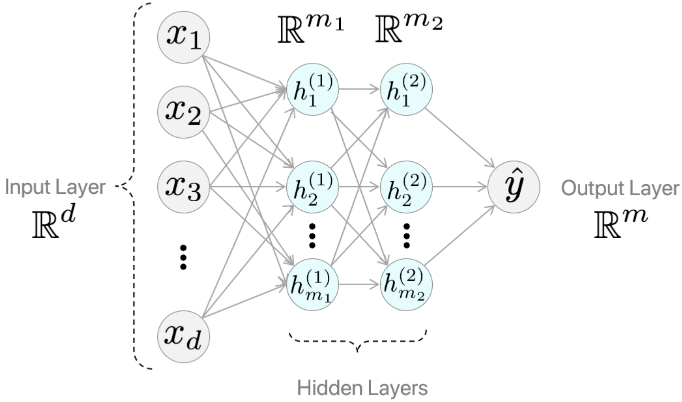


The recipe of graph-based machine learning

Learning on graphs = graph embedding + model for learning on euclidean data

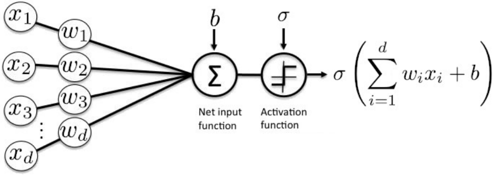
The Multi-Layer Perceptron

The architecture



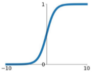
The Multi-Layer Perceptron

The Perceptron rule

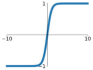


Activation functions

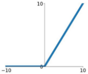
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



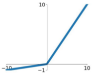
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

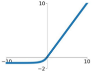


Leaky ReLU
$\max(0.1x, x)$



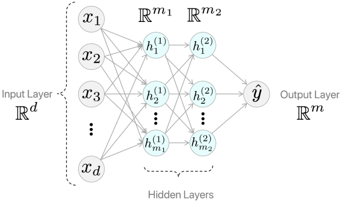
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



The Multi-Layer Perceptron

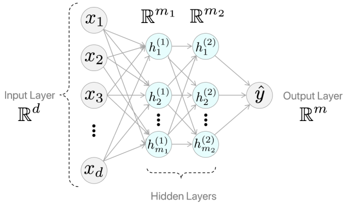
The output



$$\hat{\mathbf{y}} = \sigma^{(l)}(\mathbf{W}^{(l)} \ldots \sigma^{(2)}(\mathbf{W}^{(2)}\sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{x}))\ldots)$$

The Multi-Layer Perceptron

The output

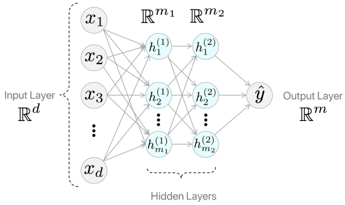


$$\hat{\mathbf{y}} = \sigma^{(l)}(\mathbf{W}^{(l)} h^{(l-1)})$$
$$h^{(k)} = \sigma^{(k)}(\mathbf{W}^{(k)} h^{(k-1)})$$

- **Training**: adjust $\mathbf{W}^{(\cdot)}$ to minimize $E = \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|$ on the training set (SGD algorithm)

The Multi-Layer Perceptron

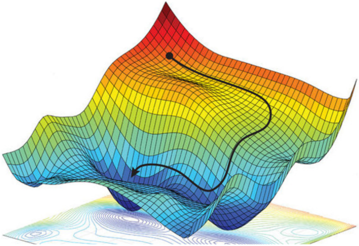
The output



- **Training**: adjust $\mathbf{W}^{(\cdot)}$ to minimize $E = \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|$ on the training set (SGD algorithm)
- **The dilemma**: width vs depth
  - ▷ Deep Neural Network: Networks with more than 2 hidden layers
  - ▷ Empirically more layers improves learning but significantly complicates training

The Gradient Descent Algorithm

The loss $E = \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|$ is a high-dimensional function of the weights $\mathbf{W}^{(\cdot)}$



Algorithm :

1. Initialize weights at random
2. Iteratively update each weight according to the rule:

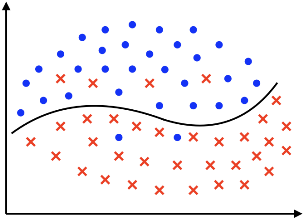
$$w_{i,j}^{(l)} = w_{i,j}^{(l)} - \eta \frac{\partial E}{\partial w_{i,j}^{(l)}}$$

where η is the learning rate.

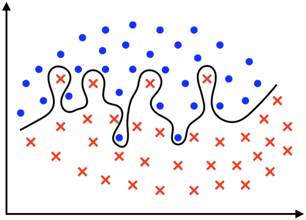
The Overfitting problem

Overfitting: We minimize E very well but generalize poorly

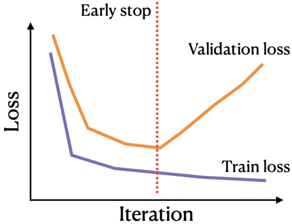
Normal fitting



Overfitting



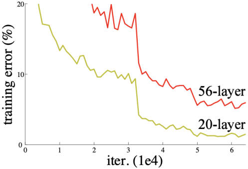
Splittings to avoid overfitting



The Multi-Layer Perceptron

The deep architectures

More layers

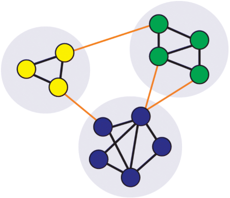


With adaptation

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Spectral Clustering on Graphs

Unsupervised embedding to perform community detection



Recalling conductance-based community detection

Find the partition $\mathcal{V} = S \cup S^c$ satisfying:

$$h_G = \min_S h_S = \min_S \frac{\text{cut}(S, S^c)}{vol(S)}$$

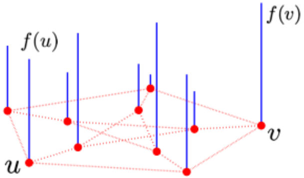
The Laplacian matrix

Laplacian matrix

$$L = D - A$$

- D: Diagonal degree matrix
- A: Adjacency matrix

The Laplacian and functions on the graph



$$f^T L f = \frac{1}{2} \sum_{i,j} a_{ij} \left(f(i) - f(j)\right)^2$$

Relaxation via Spectral Clustering

Rewriting using the Laplacian

$$h_G = \min_S \frac{\mathbb{1}_S^\top \mathbf{L} \mathbb{1}_S}{\mathbb{1}_S^\top \mathbf{D} \mathbb{1}_S}$$

Relaxation via Spectral Clustering

Rewriting using the Laplacian

$$h_G = \min_S \frac{\mathbb{1}_S^\top \mathbf{L} \mathbb{1}_S}{\mathbb{1}_S^\top \mathbf{D} \mathbb{1}_S} \approx \min_g \frac{\mathbf{g}^\top \mathbf{L} \mathbf{g}}{\mathbf{g}^\top \mathbf{D} \mathbf{g}}$$

Relaxation via Spectral Clustering

Rewriting using the Laplacian

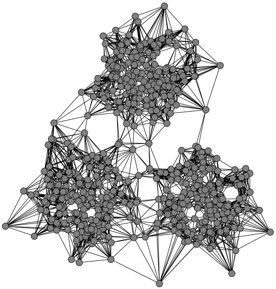
$$h_G = \min_S \frac{\mathbb{1}_S^\top \mathbf{L} \mathbb{1}_S}{\mathbb{1}_S^\top \mathbf{D} \mathbb{1}_S} \approx \underbrace{\min_g \frac{\mathbf{g}^\top \mathbf{L} \mathbf{g}}{\mathbf{g}^\top \mathbf{D} \mathbf{g}}}_{\text{eigenvalue problem}}$$

Relaxation via Spectral Clustering

Rewriting using the Laplacian

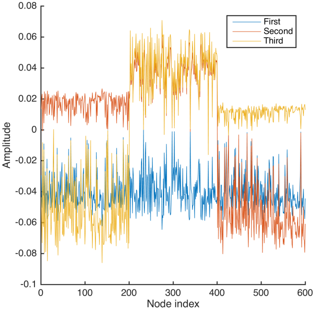
$$h_G = \min_S \frac{\mathbb{1}_S^\top \mathbf{L} \mathbb{1}_S}{\mathbb{1}_S^\top \mathbf{D} \mathbb{1}_S} \approx \underbrace{\min_g \frac{\mathbf{g}^\top \mathbf{L} \mathbf{g}}{\mathbf{g}^\top \mathbf{D} \mathbf{g}}}_{\text{eigenvalue problem}}$$

Graph



Embedding

Eigenvectors of RW-Lap



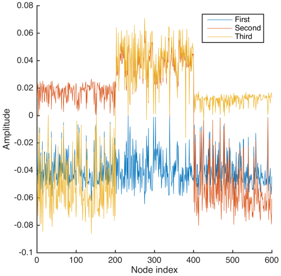
Spectral Clustering

Multi-class clustering

Multi-class Spectral Clustering algorithm:

1. Compute the first K eigevectors of $\mathbf{D}^{-1}\mathbf{L} = \mathbb{I} - \mathbf{P}$

Eigenvectors



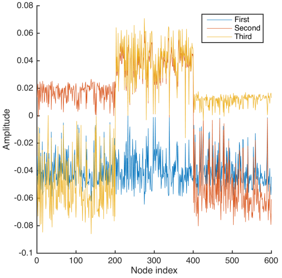
Spectral Clustering

Multi-class clustering

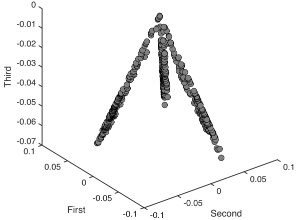
Multi-class Spectral Clustering algorithm:

1. Compute the first K eigevectors of $\mathbf{D}^{-1}\mathbf{L} = \mathbb{I} - \mathbf{P}$
2. Interpret the eigenvectors as coordinates in $\mathbb{R}^K$

Eigenvectors



Interpretation



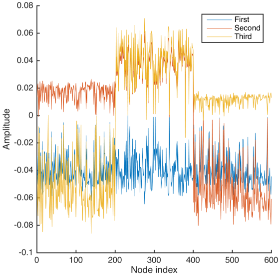
Spectral Clustering

Multi-class clustering

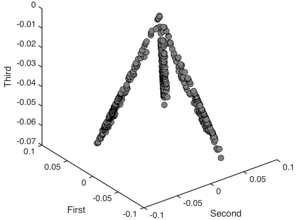
Multi-class Spectral Clustering algorithm:

1. Compute the first K eigevectors of $\mathbf{D}^{-1}\mathbf{L} = \mathbb{I} - \mathbf{P}$
2. Interpret the eigenvectors as coordinates in $\mathbb{R}^K$
3. Apply k-means

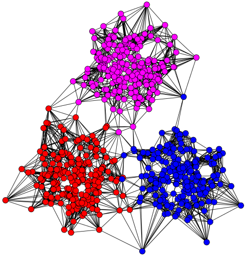
Eigenvectors



Interpretation

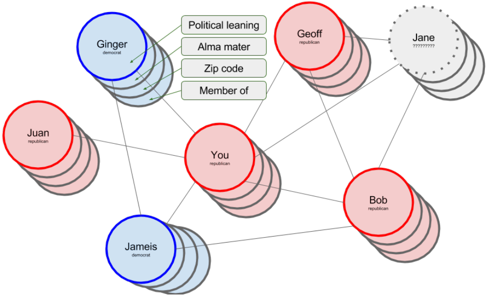


k-means result



Graph Convolutional Neural Networks

Attributed network: Graph + Feature vector on nodes



- $A \in \mathbb{R}^{N \times N}$: Adjacency matrix
- $X \in \mathbb{R}^{N \times D}$: Feature matrix

Graph Convolutional Neural Networks

From :

$$\hat{\mathbf{y}} = \sigma^{(l)}(\mathbf{W}^{(l)} h^{(l-1)})$$
$$h^{(k)} = \sigma^{(k)}(\mathbf{W}^{(k)} h^{(k-1)})$$

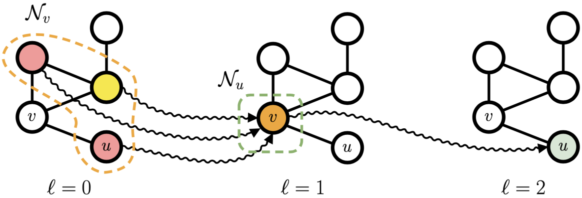
Do :

$$\hat{\mathbf{Y}} = \sigma^{(l)}(\mathbf{W}^{(l)} \widetilde{L} H^{(l-1)})$$
$$H^{(k)} = \sigma^{(k)}(\mathbf{W}^{(k)} \widetilde{L} H^{(k-1)})$$

- $\widetilde{L} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}$
- $\widetilde{A} = A + \mathbb{I}$
- $H^{(0)} = X$

Graph Convolutional Neural Networks

At the node level



$$h_v^{(k)} = \mathbf{W}^{(k)} \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(k-1)}}{\sqrt{d(u)}\sqrt{d(v)}}$$

Graph Convolutional Neural Networks

Node classification

- Output $\hat{\mathbf{Y}} \in \mathbb{R}^{|\mathcal{Y}_L|}$ is a linear layer

$$\hat{\mathbf{Y}} = W^{(l)} H^{(l-1)}$$

- Weight matrices $\mathbf{W}^{(l)}$, $\mathbf{W}^{(l-1)}$, ..., optimized to minimize (Cross-entropy)

$$E = -\sum_{c \in \mathcal{Y}_L} \sum_{f=1}^{F} \mathbf{Y}_{cf} \ln(\hat{\mathbf{Y}}_{cf})$$

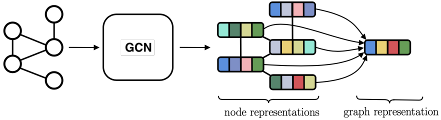
  ▷ We can update the weights in batches (only measuring E in a subset of training examples)

Graph Convolutional Neural Networks

Graph classification

- Output of $\mathcal{G}_i$, $\hat{\mathbf{Y}}_{\mathbf{i},:} \in \mathbb{R}^{|\mathcal{Y}_L|}$, is an embedding average + (opt. linear layer)

$$\hat{\mathbf{Y}}_{\mathbf{i},:} = \sum_{v \in V} h_v^{(l)}$$



- Weight matrices $\mathbf{W}^{(l)}$, $\mathbf{W}^{(l-1)}$, ..., optimized to minimize (Cross-entropy)

$$E = -\sum_{c \in \mathcal{Y}_L} \sum_{f=1}^{F} \mathbf{Y}_{cf} \ln(\hat{\mathbf{Y}}_{cf})$$

Graph Convolutional Neural Networks

Link prediction

- Output is a dot product of embeddings

$$\mathbf{\hat{Y}_{ij}} = \mathbf{\hat{A}_{ij}} = \sigma(h_i^{(l)\top} h_j^{(l)})$$

- Weight matrices $\mathbf{W}^{(l)}$, $\mathbf{W}^{(l-1)}$, ..., optimized to minimize (Cross-entropy)

$$E = -\sum_{i,j} \mathbf{A}_{ij} \ln(\mathbf{\hat{A}}_{ij})$$

Graph Convolutional Neural Networks

The general message passing procedure

- Step 1: Message dispatching
- Step 2: State update

$$h_v^{(l)} = \phi^{(l)}\left(h_v^{(l-1)}, \Psi\left(\{\psi^{(l)} h_u^{(l-1)} | u \in \mathcal{N}_v\}\right)\right)$$

- $h_v^{(l)}$: state of node v at layer l
- ϕ, ψ: arbitrary transformations
- Ψ: permutation invariant function

Graph Convolutional Neural Networks

The general message passing procedure

- Step 1: Message dispatching
- Step 2: State update

$$h_v^{(l)} = \phi^{(l)}\left(h_v^{(l-1)}, \Psi\left(\{\psi^{(l)} h_u^{(l-1)} | u \in \mathcal{N}_v\}\right)\right)$$

GCN is a special case

$$m_u^v = \Psi^{(l)}(h_u^{(l-1)} = \frac{h_u^{(l-1)}}{\sqrt{d(u)}\sqrt{d(v)}}$$

$$M_v = \Psi(\{m_u^v | u \in \mathcal{N}_v\}) = \sum_{u \in \mathcal{N}_v} m_u^v$$

$$h_v^{(l)} = \phi^{(l)}\left(h_v^{(l-1)}, M_v\right) = \sigma(W^{(l)} M_v)$$

Graph Neural Networks

Generalizations (hot research topic)

Model	Neighborhood Aggregation $\mathbf{h}_v^{\ell+1}$
NN4G [88]	$\sigma\Big(\mathbf{w}^{\ell+1^T}\mathbf{x}_v + \sum_{i=0}^{\ell}\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}} w_{c_k}^i * \mathbf{h}_u^i\Big)$
GNN [104]	$\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}\Big(\mathbf{x}_u, \mathbf{x}_v, \mathbf{a}_{uv}, \mathbf{h}_u^{\ell}\Big)$
GraphESN [44]	$\sigma\Big(\mathbf{W}^{\ell+1}\mathbf{x}_u + \hat{\mathbf{W}}^{\ell+1}[\mathbf{h}_{u_1}^{\ell}, \ldots, \mathbf{h}_{u_{\mathcal{N}_v}}^{\ell}]\Big)$
GCN [72]	$\sigma\Big(\mathbf{W}^{\ell+1}\sum_{u\in\mathcal{N}(v)}\mathbf{L}_{vu}\mathbf{h}_u^{\ell}\Big)$
GAT [120]	$\sigma\Big(\sum_{u\in\mathcal{N}_v}\alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1}\mathbf{h}_u\Big)$
ECC [111]	$\sigma\Big(\frac{1}{\lvert\mathcal{N}_v\rvert}\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}(\mathbf{a}_{uv})^T\mathbf{h}_u^{\ell}\Big)$
R-GCN [105]	$\sigma\Big(\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}}\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\mathbf{W}_{c_k}^{\ell+1}\mathbf{h}_u^{\ell} + \mathbf{W}^{\ell+1}\mathbf{h}_v^{\ell}\Big)$
GraphSAGE [54]	$\sigma\Big(\mathbf{W}^{\ell+1}(\frac{1}{\lvert\mathcal{N}_v\rvert}[\mathbf{h}_v^{\ell}, \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{\ell}])\Big)$
CGMM [3]	$\sum_{i=0}^{\ell} w^i * \Big(\sum_{c_k\in\mathcal{C}} w_{c_k}^i * \big(\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\sum_{u\in\mathcal{N}_v^{c_k}}\mathbf{h}_u^i\big)\Big)$
GIN [131]	$MLP^{\ell+1}\Big(\big(1+\epsilon^{\ell+1}\big)\mathbf{h}_v^{\ell} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{\ell}\Big)$