01 _ Autoregressive

P03 Motivation : AR model for sequence of observations $X_1, X_2, \dots X_T$
↳ continous observations at discrete time-step

P04 AR model : AR model AR(p) : $X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \iff Lag[x_{t-1}, \dots, x_{t-p}]_t \Rightarrow (X_t)_t$

- c: constant
- φ_i: parameters (shared through time)
- X_{t-i} : Lagged value at time t
- ε_t : $\varepsilon_t \sim N(0, \sigma)$ white noise

$$p(X_t | X_{t-1}, \dots, X_{t-p}) \sim N(c + \sum_{i=1}^{p} \varphi_i X_{t-i}, \sigma)$$

P06 AR model property : Mean function : $\mu(t) = E[x_t] = \frac{c}{1 - \sum_{i=1}^{p} \varphi_i}$, $\forall t$

Autocovariance : $\gamma(t, i) = cov(X_t, X_{t-i})$

Person Autocorrelation function : $\rho(t, i) = \frac{cov(x_t, x_{t-i})}{\sqrt{var(x_t)}\sqrt{var(x_{t-i})}} \in [-1, 1]$

P08 φ Parameter Learning :

$$\begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_p \end{bmatrix} = (X^T X)^{-1} X^T y \qquad X = \begin{bmatrix} x_{p-1} & \cdots & x_0 \\ x_p & \cdots & x_1 \\ \cdots & & \cdots \end{bmatrix} \qquad y = \begin{bmatrix} x_p \\ x_{p+1} \\ \cdots \end{bmatrix}$$

P12 AR model - Stationarity : A process is stationarity if : $-1 < \phi < 1$

good modeling assumption

- $E[x_t] = E[x_{t-i}] = \mu$, $\forall t \ \forall i \iff$ constant mean
- $cov(X_t, X_{t-i}) = \gamma_i$, $\forall t \ \forall i \iff$ depend only on i, not t
- $E[|X_t|^2] < \infty$, $\forall t$

$\Rightarrow \gamma_i = cov(x_t, x_{t-i}) = cov(x_{t-i}, x_t) = \gamma_{-i}$

特征值解法

$X_t = c + \phi_1 L X_t + \varepsilon_t$

$(1 - \phi_1 L) = 0$

$L = 1^{\phi}$

$|L| > 1$

P14 Parameter Learning under Stationary : Yule Walker

$$\gamma_0 = \sum_{j=1}^{p} \varphi_j \gamma_{-j} + \sigma^2$$
$$\gamma_1 = \sum_{j=1}^{p} \varphi_j \gamma_{1-j}$$
$$\dots$$
$$\gamma_p = \sum_{j=1}^{p} \varphi_j \gamma_{p-j}$$

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{p-1} \\ \gamma_p \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_{-1} & \cdots & \gamma_{2-p} & \gamma_{1-p} \\ \gamma_1 & & & & \gamma_{2-p} \\ \vdots & & \ddots & & \vdots \\ \gamma_{p-2} & & & & \gamma_{-1} \\ \gamma_{p-1} & \gamma_{p-2} & \cdots & \gamma_1 & \gamma_0 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_{p-1} \\ \varphi_p \end{bmatrix}$$

Yule - Walk matrix

02 _ Markov Chains

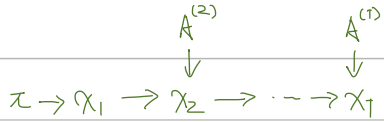
P03 Markov Chains : Markov Chains is a sequence of $X_1, X_2, \cdots, X_T$ which fulfills the Markov property

$p(x_t \mid x_1, \cdots, x_{t-1}) = p(x_t \mid x_{t-1})$ / discrete time t and discrete x_t

Joint Distribution: $p(x_1 = i_1, \cdots, x_T = i_T) = p(x_1 = i_1) \prod_{t=1}^{T-1} p(x_{t+1} = i_{t+1} \mid x_t = i_t)$

P04 General case : $p(x_1 = i) = \pi_i$ and $p(x_{t+1} = j \mid x_t = i) = A_{ij}^{(t+1)}$

prior distribution: $\pi \in R^K$ / Transition matrix $A^{(t)} \in R^{K \times K}$



Joint Distribution: $p(x_1 = i_1, \cdots, x_T = i_T) = \pi_{i_1} \times A^{(2)}_{i_1, i_2} \times \cdots \times A^{(T)}_{i_{T-1}, i_T}$

#paras = $(K-1) + (K-1)K(T-1)$

P05 Stationary : $p(x_1 = i) = \pi_i$ and $p(x_{t+1} = j \mid x_t = i) = A_{ij}$ / $A^{(t)} = A$, doesn't depend on t

Joint Distribution: $p(x_1 = i_1, \cdots, x_T = i_T) = \pi_{i_1} \times A_{i_1, i_2} \times \cdots \times A_{i_{T-1}, i_T}$

#paras = $(K-1) + (K-1)K$

P0 Parameter Learning : Given a set of $\{x_{1:T_n}^{(n)}\}$ → learn π and A Maximum-likelihood

$$\max_{\pi, A} \log P(all) = \sum_{k=1}^{K} L(k) \log(\pi_k) + \sum_{i=1}^{K} \sum_{j=1}^{K} N(i,j) \log(A_{ij})$$

subject to $\sum_k \pi_k = 1$, $\sum_j A_{ij} = 1$ / $A_{ij} = \frac{N(i,j)}{\sum_{j'} N(i,j')}$ $\pi_k = \frac{L(k)}{\sum_{k'} L(k')}$

① compute $A_{ij}(n) = P(x_{t+n} = j \mid x_t = i)$ ⇔ probability n step from i to j

$$A(n) = P(x_{t+n} = j \mid x_t = i) = \sum_{k=1}^{K} A_{kj} A_{ik}(n-1) = A(n-1)A \Rightarrow \boxed{A(n) = A^n}$$

Chapman-Kolmogorov ⇒ $\boxed{A(m+n) = A(m)A(n)}$

(row vector)

② compute $\pi_j(t) = P(x_t = j)$ ⇔ probability t step reaching j

$$\pi_j(t) = P(x_t = j) = \sum_{i=1}^{K} A_{ij} \pi_i(t-1) = A\pi(t-1) \iff \boxed{\pi(t) = \pi A^{(t-1)}}$$

注意 A^T 不

03 _ HMM

P03 Motivation : Probabilistic latent variable models for sequence observations: $X_1, X_2, \dots, X_T$

discrete time step / discrete or continous observations

problem ①: X_t can't capture all relevant information of $[X_1, \dots, X_t]$

②: State is unknown, can only be observed indirectly

↳ not observed / latent state Z_t

Markov-property: $[Z_1, \dots, Z_T]$

No Markov-property: $[X_1, \dots, X_T]$

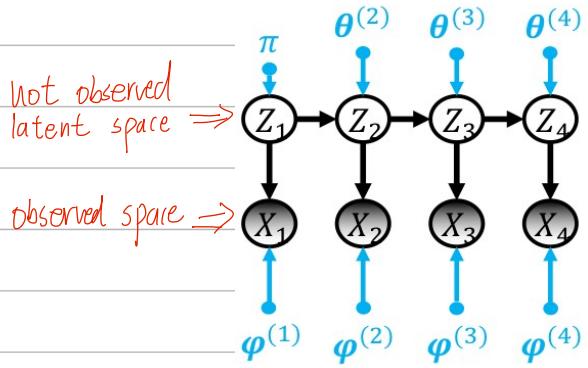
P05 HMM : Composed of a sequence of hidden / latent variables $[Z_1, \dots, Z_T]$ and observed variables $[X_1, \dots, X_T]$

and $[Z_1, \dots, Z_T]$ full-fill Markov-property: $P(Z_{t+1} | Z_t, \dots, Z_1) = P(Z_{t+1} | Z_t)$

Transition probabilities

X_t depend only on Z_t and not full-fill Markov-property: $P(X_{t+1} | Z_1, \dots, Z_T, X_1, \dots, X_T) = P(X_{t+1} | Z_{t+1})$

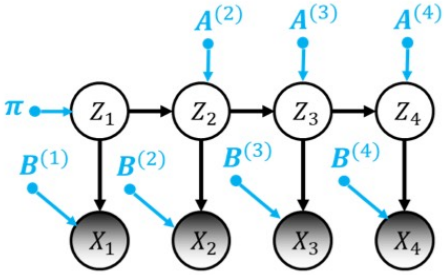
Emission probabilities



Discrete Case

Joint Distribution :

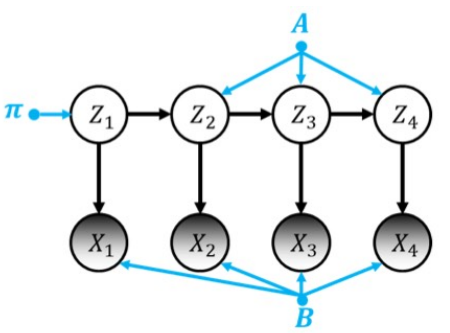
$$P(Z_1 = z_1, \dots, Z_T = z_T, X_1 = x_1, \dots, X_T = x_T)$$
$$= \underbrace{P(Z_1 = z_1; \boldsymbol{\pi})}_{\text{prior}} \prod_{t=1}^{T-1} \underbrace{P(Z_{t+1} = z_{t+1} | Z_t = z_t; \boldsymbol{\theta}^{(t+1)})}_{\text{Transition prob}} \prod_{t=1}^{T} \underbrace{P(X_t = x_t | Z_t = z_t; \boldsymbol{\varphi}^{(t)})}_{\text{condition prob}}$$



Discrete Case

Paras $= K + (T-1)K^2 + T \cdot K \cdot K'$

$$\begin{cases} p(Z_1 = i) = \pi_i \\ p(Z_{t+1} = j | Z_t = i) = A_{ij}^{(t+1)} \in R^{K \times K} \\ p(X_{t+1} = j | Z_t = i) = B_{ij}^{(t+1)} \in R^{K \times K'} \end{cases}$$



Share Parameters:

para $= K + K^2 + K \cdot K'$

Joint Distribution :

$$P(Z_1 = z_1, \dots, Z_T = z_T, X_1 = x_1, \dots, X_T = x_T) = P(Z_1 = z_1; \boldsymbol{\pi}) \prod_{t=1}^{T-1} A_{z_t z_{t+1}} \prod_{t=1}^{T} B_{z_t x_t}$$

Transition Emission

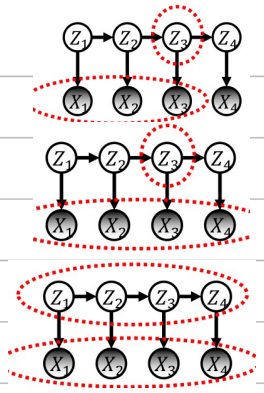
P12 Inference : ① Filtering $P(z_t | x_{1:t})$

online setting

② Smoothing $P(z_t | x_{1:T})$

offline setting

③ MAP Inference $\arg\max_{z_{1:T}} P(z_{1:T} | x_{1:T})$



P13 Algorithm : ① Forward : Goal : $P(z_t = k | x_{1:t}) = \frac{\alpha_t(k)}{sum(\alpha_t)}$

$\alpha_1(k) = \pi_k \odot B_{k x_1} / \alpha_{t+1}(k) = B_k(x_{t+1}) \sum_{j=1}^{k} A_{jk} \alpha_t(j)$

vector $\Rightarrow$ $\alpha_{t+1}(k) = B_k(x_{t+1}) \odot (A' \alpha_t(k))$

② Forward - Backward : Goal : $P(z_t = k | x_{1:T}) \propto \alpha_t(k) \cdot \beta_t(k) = \frac{\alpha_t(k)\beta_t(k)}{sum(\alpha_t \cdot \beta_t)}$

$\beta_T(k) = 1 \quad / \beta_t(j) = \sum_{k=1}^{k} A_{jk} B_k x_{t+1} \beta_{t+1}(k)$

$\beta_t(k) = A(B_k(x_{t+1}) \odot \beta_{t+1}(k))$

③ MAP Inference : Goal : $\arg\max_z P(z_{1:T} | x_{1:T}) = \arg\max_z \log[P(z_1) p(x_1|z_1)] + \sum_{t=2}^{T} \log[p(z_t|z_{t-1}) p(x_t|z_t)]$

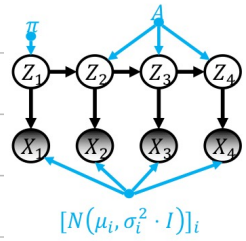
前向算法

- weights of edges connected to the *Start* node: $-\log[\Pr(Z_1 = j) \Pr(X_1 | Z_1 = j)]$ $= g(i)$
- weights of the intermediate layers: $-\log[\Pr(Z_t = j | Z_{t-1} = i) \Pr(X_t | Z_t = j)]$ $= f(i,j)$
- weights of the edges connected to the *End* node: 0

sum of edge weights $= -\log P(z_{1:T}, x_{1:T})$

P25 Parameter Learning : model parameters : $\theta = \{\pi, A, B\}$ / Goal : solve $\max_\theta \log P(x|\theta)$ / EM - step

P30 Continous Data :



Continous case, e.g. time-series segementation

$P(z_1 = i) = \pi_i$

$P(z_{t+1} = j | z_t = i) = A_{ij}$

$P(x_{t+1} = x | z_t = i) = N(x | \mu_i, \sigma_i^2 \cdot I)$

parameter Learning P33

04 _ NN Approaches

P03 Word representation : ① One-hot encoding (word are independent of each other)

Distributional hypothesis : words in similiar contexts should have similiar meanings

similiar words should have vector near each other

P06 Co-occurence Matrix M : aware of contex ⇒ count word appearance

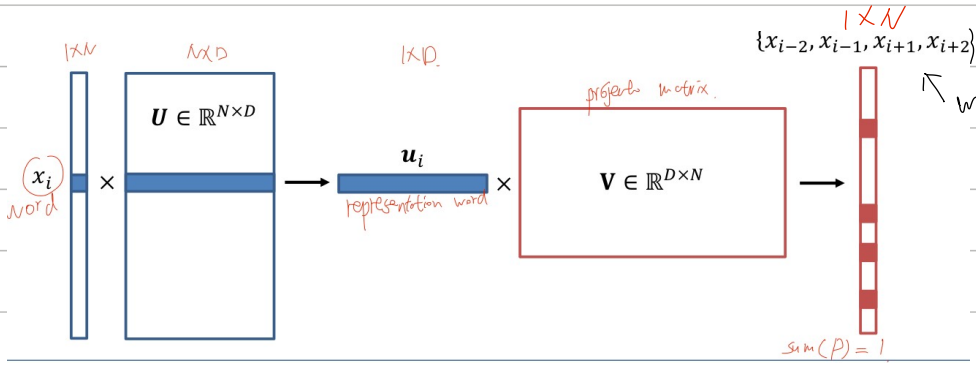
↓ Words $\{x_1, \dots, x_N\}$, window size L, $\{x_{i-L}, \dots, x_{i-1}, x_{i+1}, \dots, x_{i+L}\}$

SVD : $M = U \Sigma V^T$ / first D columns of V = D-dim word vectors

P09 Word2Vec : use NN to get word vectors / predict words based on context

Two approaches : CBow / Skip-gram.

P10 Skip-Gram :



$S = \{x_{i-L}, \dots, x_{i-1}, x_{i+1}, x_{i+L}\}$, θ is the model parameters

$$\max_{\boldsymbol{\theta}} \mathbb{E}[P(S|x_i, \boldsymbol{\theta})] = \min_{\boldsymbol{\theta}}(-\mathbb{E}[P(S|x_i, \boldsymbol{\theta})])$$

$$\text{where } P(S|x_i, \boldsymbol{\theta}) = \prod_{x_k \in S} P(x_k|x_i, \boldsymbol{\theta})$$

$$\text{and } P(x_k|x_i, \boldsymbol{\theta}) = \text{softmax}(\boldsymbol{u_i V})_k$$

corresponding embedding

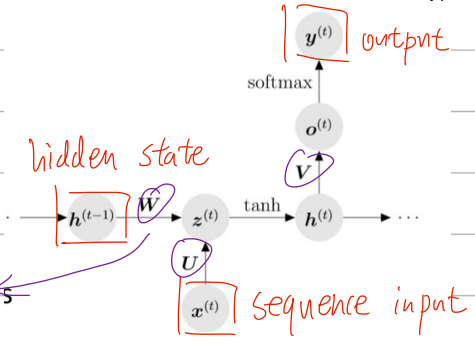
* choose U = V ⇒ less expressiveness / parameters

Inefficient for large vocabularies

* Alternative : Negative Sampling → BCE

P15 RNN : One to Many / Many to One / Many to Many

$$\boldsymbol{z}^{(t)} = \boldsymbol{W h}^{(t-1)} + \mathbf{U x}^{(t)} + \boldsymbol{b}$$
$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{z}^{(t)})$$
$$\boldsymbol{o}^{(t)} = \mathbf{V h}^{(t)}$$
$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)})$$



weights share over time

The gradient $\frac{\partial L}{\partial h^{(t)}}$ depends on future times

$$\frac{\partial L}{\partial \boldsymbol{h}^{(t)}} = \boldsymbol{V}^T(\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}) + \boldsymbol{W}^T \text{diag}\left(1 - (\boldsymbol{h}^{(t+1)})^2\right) \frac{\partial L}{\partial \boldsymbol{h}^{(t+1)}}$$

→ impact may vanish or explode

→ RNN can't retain informations for many steps

↗ skipping / avoid gradient problem

P21 GRU: Add Gate mechanism

↙ train parameter ↘ determine gate which to learn / not to learn.

$$z^{(t)} = \sigma\left(W_z\left[h^{(t-1)}, x^{(t)}\right]\right)$$
$$r^{(t)} = \sigma\left(W_r\left[h^{(t-1)}, x^{(t)}\right]\right)$$

Gates

$$\tilde{h}^{(t)} = \tanh\left(W\left[r^{(t)} \odot h^{(t-1)}, x^{(t)}\right]\right)$$

Simple RNN update – gives candidate state

$$h^{(t)} = (1 - z^{(t)}) \odot h^{(t-1)} + z^{(t)} \odot \tilde{h}^{(t)}$$

new information

How much to take from previous state vs. candidate state

P22 LSTM: Add cell state $c^{(t)}$

$$f^{(t)} = \sigma\left(W_f\left[h^{(t-1)}, x^{(t)}\right]\right)$$ Forget gate

$$i^{(t)} = \sigma\left(W_i\left[h^{(t-1)}, x^{(t)}\right]\right)$$ Input gate

$$o^{(t)} = \sigma\left(W_o\left[h^{(t-1)}, x^{(t)}\right]\right)$$ Output gate

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tanh\left(W\left[h^{(t-1)}, x^{(t)}\right]\right)$$

Previous cell state

Simple RNN update – LSTM treats it as an input

$$h^{(t)} = o^{(t)} \odot \tanh(c^{(t)})$$

Update hidden state (now the output) using a cell state

P25 Non-Recurrent Model: Weight in RNN rely on the whole history

use Convolutional Neural Networks (ConvNets)

P28 Wave Net: use 1-D ConvNet

Causal convolutions: ensure the output only depends on the past

Dilated convolutions: skip some input to increase the receptive field

P30 Transformers & Attention: Attention ⇔ weighting elements

positional Encoding: represent the token order

Training and Inference: During training, all masked embeddings are input

During inference, decode is step by step

GPT: unsupervised pre training to predict next token

task-specific fine tuning

05_Temporal Point Processes

P03 Event Data : Discrete events in continous time

task : when next event happen / how many events happen

different time series : (synchronous) event data : regular intervals

(Asynchronous) event data : Irregular intervals

P05 Temporal Point Process : (TPP) ⇒ probabilistic model ⇒ describe distribution of discret event

⇒ generative model for variable-length sequence $t=\{t_1,\cdots,t_n\}$ in interval $[0,T]$

⇒ Likelihood Function $p(\{t_1,\cdots,t_n\})$

P06 predict next event : ① Conditional density as $p^*(t) := p(t|H(t))$. $H(t)$ is history : $H(t)=\{t_j<t\}$

② Cumulative distribution function (CDF)

$F^*(t) = \int_{t_{i-1}}^{t} p^*(u)du \iff$ Probabilty that next event happen in $[t_{i-1}, t)$

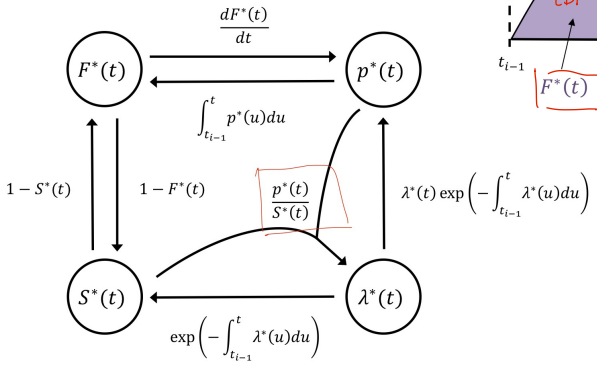
Survival function

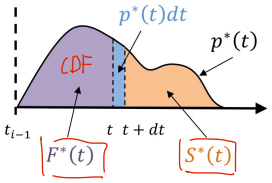
$S^*(t) = 1-F^*(t) = \int_t^{\infty} p^*(u)du \iff$ Probability that next event happen after t

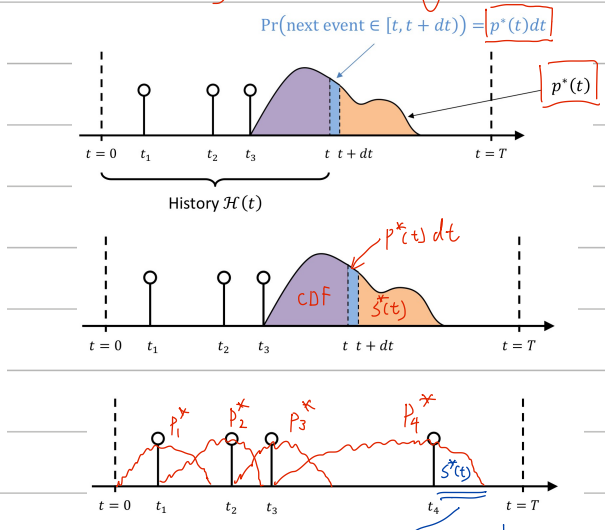
③ Conditional intensity

$\lambda^*(t)dt$ = probability of event in $[t, t+dt)$ given no event in $[t_{i-1}, t) = \frac{p^*(t)dt}{S^*(t)}$

= Expected # of events $\lambda^*(t)dt$



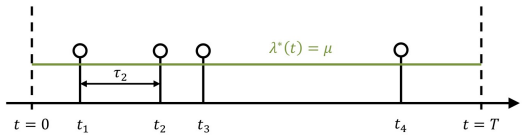




P10 Likelihood of Sequence : $p(\{t_1,t_2,t_3,t_4\}) = p^*(t_1)p^*(t_2)p^*(t_3)p^*(t_4)S^*(T)$

$= \lambda^*(t_1)\lambda^*(t_2)\lambda^*(t_3)\lambda^*(t_4)\exp\left(-\int_0^T \lambda^*(u)du\right)$

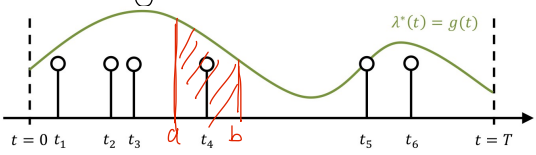
P13 Homogenous Poisson Process : (HPP) Constant intensity : $\lambda^*(t) = \mu$



Inter-event times : $p^*(t) = \mu \exp\left(-\int_{t_{i-1}}^{t} \mu \, du\right) = \mu \exp\left(-\mu(t - t_{i-1})\right)$

P15 Inhomogenous Possion Process : (IPP) Intensity : $\lambda^*(t) = g(t) \geq 0$

↳ independent of History / Capture global trend

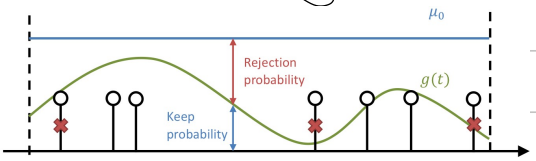


Expected Events : $N([a, b]) \sim \text{Poisson}\left(\int_a^b g(t)\, dt\right)$

Combination of two IPPs : $g(t) + h(t)$

$$p(x = k) = \frac{e^{-\lambda} \lambda^k}{k!}$$

P18 Simulating an IPP : Find upper bound $\mu_0 \geq g(t)$

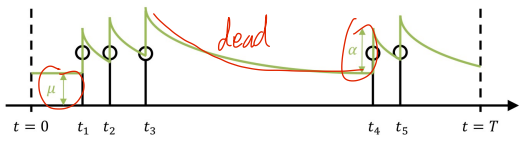


Simulate Event from HPP with μ_0

keep each event probability $g(t_i) / \mu_0$

P19 Hawkes Process : Self-exciting Process : $\lambda^*(t) = \mu + \alpha \sum_{t_j \in H(t)} k_w(t - t_j)$

↳ Triggering kernel $= \exp(-w(t - t_i))$



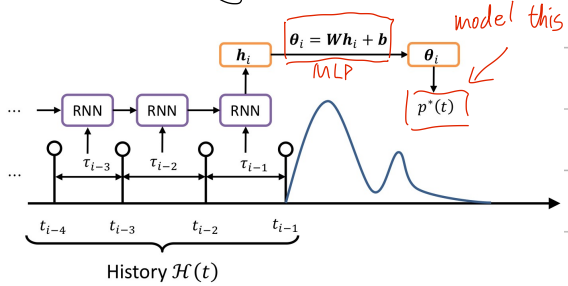
Intensity depend on history

P20 Parameters Training : MLE :

$$\max_{\theta} \sum_{t = \{t_1, \cdots, t_N\} \in D_{train}} \log P_\theta(\{t_1, \cdots, t_N\})$$

$$= \sum_{i=1}^{N} \log \lambda^*(t_i) - \int_0^T \lambda^*(u)\, du$$

P22 Modeling TPPs with RNN

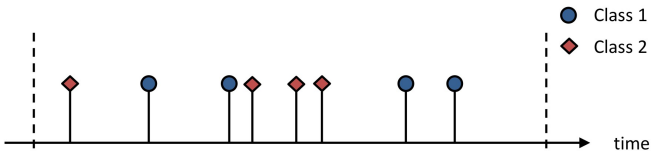


$p^*(t) = P(t \mid H(t)) = p(t \mid h_i)$

P23 Marked TPP : Categorical marks

Events has associated class

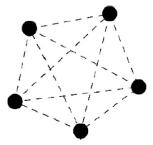
Events may influence each other



01 _ Graphs and Networks

P07 Graphs : $G=(V,E)$ / V: sets of nodes, E: sets of edges $\Rightarrow E \subseteq V \times V$

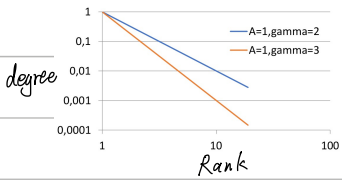
Adjacency matrix : $A \in \{0,1\}^{|V| \times |V|}$



P08 Erdös-Renyi Random Graph Model : Start with N isolated Nodes

Each pair $v_1, v_2 \in V$ add edges with equal probability p

Real world graph is not random — No obvious patterns

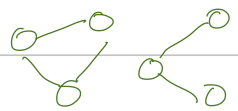


P10 Power Law Distribution : $y(x) = Ax^{-\gamma}$ / x and Y are variables, A and γ are positive constants

↳ Scale-free Law : $y(a \cdot x) = b \cdot y(x)$

↳ A good descriptions / fit but not the data itself

P15 Patterns and Algorithms: Clustering Coefficient $C = \dfrac{3 \cdot \#\text{traingles}}{\#\text{connected triplets}}$



Real world $\Leftrightarrow$ $C \uparrow$ $\Leftrightarrow$ Strong Community // Expensive calculate

Efficient Triangle Counting : $\#\text{traingles} = \frac{1}{6}\text{trace}(A^3) = \frac{1}{6}\sum_i \lambda_i^3$

↳ Eigen values of adjacency matrix A

use power iteration to compute single eigen vector $V \leftarrow \frac{X \cdot V}{\|X \cdot V\|}$

P21 Real Graph : N nodes for N^2 edges

In Real, $E << N^2 \Rightarrow$ Sparse

P23 Characteristic Path Length : For starting Node V consider shortest path to each other

Take average length

Take average length for all starting node and median

$\text{median}\{\frac{1}{|V|}\sum_{v_j \in V} \text{len } p_{min}(v, v_j)\}$

Average Diameter : $\text{mean} = \frac{1}{|V|}\sum_{v \in V} \frac{1}{|V|}\sum_{v_j \in V} \text{len } p_{min}(v, v_j)$

Eccentricity / Hop-plot : # reachable nodes via h hops : $N_h(u) = \{v \in V \mid \text{len}(p_{min}(u,v)) < h\}$

Total neighborhood size : $N_h = \sum_{u \in V} |N_h(u)|$

Effective diameter : 90%

02 _ Generative Models

P02 Generative Model : Statistical model describe data distribution

Task: unsupervised learning $p(x)$ / generating data

GMM : prior probability for each cluster $\pi_k > 0 / \sum_k \pi_k = 1$

cluster indicator $z_i \sim Cat(\pi)$

Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$

→ For Graph : Don't know the latent factors

a) What is the expected number of cliques of size m?

$\binom{n}{m} p^{\binom{m}{2}}$

P05 Erdös-Renyi : Given $p \in [0,1] \rightarrow A_{ij} \sim Bernoulli(p)$ # 6 $\binom{N}{3} p^3$

Probability of vertex has degree k $p_k = \binom{N-1}{k} \cdot p^k \cdot (1-p)^{N-1-k} \approx \frac{z^k e^{-z}}{k!}$, $z = p(N-1)$

$\Rightarrow$ possion $\leftrightarrow$ real world is power-law X

Diameter : $\log(N) / \log(z)$ $\Rightarrow$ grow slowly $\leftrightarrow$ real world is constant or shrink X

Clustering Coefficient : $\Rightarrow p = z/(N-1)$ $\leftrightarrow$ Real world look totally different X

→ Generalization of Erdos-Renyi : node from same community connect with p

node from diff community connect with q, $p > q$

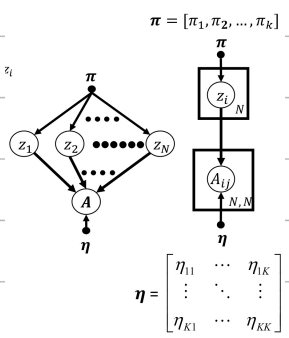
↓ Improvement

P08 Planted Partition Model (PPM) : $V \rightarrow C_1, C_2$, assign node : $z_i \in \{1, -1\}$ ← latent variable

$$p(A_{ij} = 1 \mid z_i, z_j) = \begin{cases} p & \text{if } z_i = z_j \\ q & \text{if } z_i \neq z_j \end{cases}$$

disadvantage : two clusters / same p / ... "boring"

P10 Stochastic Block Model (SBM): advanced PPM ← multiple cluster



Random variables $\Rightarrow$ $z_i \in \{1, \ldots, k\}$ / $A \in \{0,1\}^{N \times N}$

Model parameters $\Rightarrow$ $\pi = [\pi_1, \cdots, \pi_K]$: community proportions / η_{uv} : edge probability in 2 cluster

Condition distributions $\Rightarrow$ $P(z_i = k) = \pi_k$ / $P(A_{ij} \mid z_i, z_j) = Ber(\eta_{z_i, z_j})$

p / q

Not capture all patterns X ∈

$\eta \begin{pmatrix} a & b \\ b & a \end{pmatrix}$ # 6 $= \binom{N}{3} a^3$ 均分

+ × 2

6 $= \binom{N}{2} \binom{N}{1} b^2 \cdot a$ 分

2 in C_1 1 in C_2

$B_{ii} = \frac{\#\text{edges within group } i}{\binom{\#\text{nodes in group } i}{2}}$

$B_{ij} = \frac{\#\text{edges between groups } i, j}{\#\text{nodes in group } i \cdot \#\text{nodes in group } j}$ if $i \neq j$

P15 Preferential Attachment Models: ER / PPM / SBM assume edge generated independently

↳ Generate Network in two steps

↳ Growth: start with a set node, add new nodes and edges

Preferential attachment: "rich get richer", # connecting nodes ∝ degree

P17 Initial Attractiveness: start with m_0 nodes → add a new node w → insert m edges

↳ Probability that endpoint of edge (u,v) corresponds to V

is proportional to $A_v = A + \text{indeg}(V)$ / A: initial attractiveness for all nodes

↳ currently # incoming edges

parameters: m (new edge per step) / A (initial attractiveness)

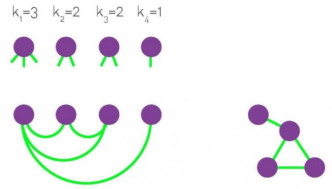
Degree distributions: $\gamma = 2 + \frac{A}{m}$ ⇔ power law ✓

Diameter: increase ⇔ not same as the real world ✗

Average degree: constant ⇔ increase in real world ✗

P19 Configuration Model: assign degree to each node at stub

stub ⇔ half link → random graph

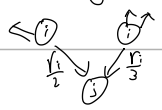


P22 Deep Generative Model: Difficult to hand-craft a graph fit for all real world data

Solution: VAE / GAN / ... / Net GAN

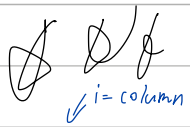
03 _ Ranking

P04 Page Rank : Ranking of Nodes ⇔ pages not equally important



↳ Page is important if many important pages point to it

Rank of page j: $r_j = \sum_{i \to j} \frac{r_i}{d_i}$ / d_i : out-degree of node i



Gaussian elimination → Adjacency Matrix M → sum(column M) = 1 ⇔ $\sum_i r_i = 1$

↳ if $i \to j$, then $M_{ji} = \frac{1}{d_i}$, else = 0

Eigen vector : $r = M \cdot r \Rightarrow 1 \cdot r = Mr \Rightarrow r$ is eigenvector

Finding r = Finding eigenvector of M with largest eigenvalue

↳ power iteration : $r \leftarrow \frac{Mr}{\|Mr\|}$ M is sparse

P10 Random Walk Interpretation : Surfer path ⇔ Markov chain

↓ Start from random pages : $p(X_1 = i) = \pi_i$

↓ Transition probabilities : $p(X_{t+1} = j \mid X_t = i) = M_{ji} \Leftrightarrow B = M^T$

$p(X_t = i) = \pi_i(t) \Leftrightarrow \pi(t) = \pi B^{(t-1)}$

Stationary distribution ⇔ $\pi^\infty = \pi^\infty B$ → r^T

limiting distribution : $\lim_{t \to \infty} \pi(t)$

↳ $r = \lim_{t \to \infty} \pi(t)$ // rank score of page $i = r_i = \lim_{t \to \infty} p(X_t = i)$

$r = \pi B, (\pi B)B, [(\pi B) \cdot B] \cdot B \cdots$

super efficient ⇔ Vector · Matrix

↳ Probability of reaching a node is not depend on Start point

↳ assume $t \to \infty$ B^t has same row

$$\lim \pi(t) = \lim \pi B^{(t-1)} = [\pi_1 \ \pi_2 \ \pi_3] \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \end{bmatrix} = [k a \quad k b \quad k c]$$

$k = \pi_1 + \pi_2 + \pi_3$

Above limiting distribution must be Irreducible and Aperiodic

任意狀態可以到述任意狀態 → Irreducible : possible to get any state from any state

從任意狀態出發可以回到當前狀態 → Aperiodic : exists n → for all $n' \geq n \Rightarrow p(X_{n'} = i \mid X_1 = i) > 0$

$n' \geq n$ 自然就對

↳ Markov chain is Aperiodic if all states are Aperodic

↳ Irreducible markov chain : one Aperiodic → all Aperiodic

P15 Problems and solutions : Dead ends (not irreducible) 详细
Spider traps (not irreducible)
Periodic (not aperiodic)

Random Teleports β: follow a random link
$1-\beta$: jump to random page
$\Rightarrow r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$
$A = \beta M + (1-\beta)[\frac{1}{N}]_{N\times N} \Rightarrow r = Ar$ $\Rightarrow B = M^T \to B = A^T$ $r^{B(t-1)}$
problem: A is dense Matrix
Instead: $r = \beta M r + (1-\beta)[\frac{1}{N}]_N$ ← Sparse
↖ vector

Page Rank Problems: Measure generic popularity / Link Spam / Single measure of importance

P22 Topic-Sensitive PageRank : based on topic popularity
Core idea : Bias the random walk
Standard PageRank : S = all pages with equal probability
Topic-Sensitive PageRank: S = set of "relevant" pages / diff rs

$r = \beta M r + (1-\beta)\pi$ where $\pi_i = \begin{cases} \frac{1}{|S|} & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$
↓
$r = \beta M r + (1-\beta)\pi$ where $\sum_i \pi_i = 1$
↓
$r = (1-\beta)(I-\beta M)^{-1}\pi$

*important: when convergence , set $S=\{i\}$, $r_i \geq 1-\beta$ (jump)

P26 Variants : Normal $\pi_i = c$
Personalized page
Spam ⇔ Trust Rank

等比数列
$\frac{a_0(1-q^n)}{1-q}$
$C(n,k)$ $\frac{n!}{k!\times(n-k)!}$

04 _ Clustering

P03 Clustering : Given $G = (V, E)$, find clusters of vertices

Goal : vertices in same clusters has more edges

vertices in diff clusters has less edges

Two categories : Partitioning approaches : Each vertex belong to one cluster

Non-Partitioning approaches : Cluster can overlap , Vertex ∈ None

P10 Partitioning approaches : Constrained optimization problem

Given $G=(V,E)$, assign vertex V to a set of clusters $C=\{C_1,\cdots,C_K\}$

such that $Q(C)$ is optimized / s.t. $\begin{cases} C_1 \cup C_2 \cup \cdots \cup C_K = V \\ \forall 1 \le i < j \le k : C_i \cap C_j = \emptyset \end{cases}$

P12 Global Minimum Cut : Goal: Minimize the number of edges/weights between clusters

$Cut(C_1, C_2) = \sum_{v_1 \in C_1, v_2 \in C_2} w(v_1, v_2)$ is minimized

Problem : Tend to cut small vertex sets / consider only inter-cluster edges (外边)

P13 Normalized Cut :

Ratio Cut : Minimize $\frac{cut(C_1,C_2)}{|C_1|} + \frac{cut(C_2,C_1)}{|C_2|}$

Normalized Cut : Minimize $\frac{cut(C_1,C_2)}{vol(C_1)} + \frac{cut(C_1,C_2)}{vol(C_2)}$

volume of a set of nodes

$vol(C_i) =$
$assoc(C_i, V) =$
$cut(C_i, V) =$
$\sum_{v_i \in C_i, v_j \in V} w(v_i, v_j) =$
$\sum_{v_i \in C_i} \deg(v_i) =$

P14 Multi-way Cut : $k \ge 2$

- Cut: $\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} cut(C_i, V \backslash C_i)$
- Ratio Cut: $\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{cut(C_i, V \backslash C_i)}{|C_i|}$
- Normalized Cut: $\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{cut(C_i, V \backslash C_i)}{vol(C_i)}$

P15 Graph Laplacin : Effective Computation

Laplacian matrix : $L = D - W$ ｜ D: degree matrix 对角矩阵，该节点的E权重

W: weighted Adjacency matrix 未连接的=0

$$L_{uv} = \begin{cases} -w_{uv}, & (u,v) \in E \Rightarrow u \neq v, D=0 \\ deg(v), & u=v \Rightarrow u=v, D \neq 0, W=0 \\ 0, & else \Rightarrow u \neq v, D=0, (u,v) \notin E, W=0 \end{cases}$$

For any vector $f \Rightarrow f^T \cdot L \cdot f = \frac{1}{2} \sum_{(u,v) \in E} W_{uv} (f_u - f_v)^2$

→ Assign "number" to each node : $f: V \to R$

Laplacian transform f to another function g : $Lf = g$

P17 Properties : ① L is symmetric and PSD $\Rightarrow x^T L x \geq 0$

↳ D and W are symmetric

② smallest eigenvalue of L is 0, vector is $\mathbb{1}$ $\quad \lambda_1 = 0$

$Lx = \lambda x \Rightarrow x^T L x = \lambda = \frac{1}{2}\sum_{(u,v)\in E} W_{uv}(x_u - x_v)^2$

Trivial solution set $x = c\cdot\vec{1}$ and $\lambda = 0$

③ L has n non-negative eigenvalues

④ Additive : $L_{G\cup H} = L_G + L_H$

⑤ Graph laplacian : $L_G = \sum_{(u,v)\in E} L_{(u,v)}$

⑥ # Eigenvector of L with eigenvalue 0 = # connected components $= \#k$

C_k: the set of nodes of the k-th connected component

Let $f_{C_k}[i] = 1$ if $v_i \in C_k$, else $= 0$

$\Rightarrow f_{C_k}^T L f_{C_k} = 0$, $\forall k \Rightarrow f_{C_k}$ ⇐ smallest eigenvector = eigenvalues $= 0$

↑ $f^T \cdot L \cdot f = \frac{1}{2}\sum_{(u,v)\in E} W_{uv}(f_u - f_v)^2 = \boxed{\lambda}$

If graph is connected, $\lambda_2 > 0$ ⇐ $(0 = \lambda_1 < \lambda_2 < \cdots < \lambda_n)$ ⇐ 一整张 graph

↳ $\lambda_2(L)$ ⇔ Fiedler eigenvalue

→ reflect how well the connert is

$\lambda_2(K_n) = n$, if K_n is a fully connected graph with n nodes

⑦ Isoperimetric ratio of S : $\theta(S) = \frac{cut(S,\bar{S})}{|S|} > \lambda_2\left(1 - \frac{|S|}{|V|}\right)$

Cheeger constant of graph $\theta_G = \min_{|S|\leq \frac{|V|}{2}} \theta(S) \geq \frac{\lambda_2}{2}$

↖ cluster smaller than half graph

λ_2 is big = graph is well connected

⑧ Exisit co-spectral graphs, not isomorphic but share same spectrum

P22 Graph Laplacian and Minimum Cut : For $k=2$, $f \in \{-1, 1\}^n$, $f_{C_1}[i] = \begin{cases} +1, & v_i \in C_1 \\ -1, & v_i \in V \setminus C_1 = C_2 = \bar{C_1} \end{cases}$

$f_{C_1}^T L f_{C_1} = 4 \cdot cut(C_1, C_2)$ / Minimize $f_{C_1}^T L f_{C_1}$ → Minimize the cut

P23 Spectral Clustering : 2 Clusters : Ratio Cut for $k=2 \Rightarrow \min_{C_1 \in V} \frac{cut(C_1, C_2)}{|C_1|} + \frac{cut(C_2, C_1)}{|C_2|}$

Indicator vector f_{C_1} for cluster C_1 / $f_{C_1}[i] = \begin{cases} +\sqrt{\frac{|\bar{C}_1|}{|C_1|}} & \text{if } v_i \in C_1 \\ -\sqrt{\frac{|C_1|}{|\bar{C}_1|}} & \text{else} \end{cases}$

minimize cut

① $\sum_i f_{C_1}[i] = 0$ / ② $f_{C_1}^T f_{C_1} = \|f_{C_1}\|_2^2 = |V|$ / ③ $f_{C_1}^T L f_{C_1} = |V| \left[\frac{cut(C_1, C_2)}{|C_1|} + \frac{cut(C_2, C_1)}{|C_2|}\right]$

② constant length

③ $|V| \cdot$ ratio cut

Finally : Minimize $\{f_{C_1}^T L f_{C_1}\}$, s.t. $f_{C_1} \perp \vec{1}$ and $\|f_{C_1}\|_2 = \sqrt{|V|}$

f_{C_1} is the second smallest eigenvector of L

$\Rightarrow L \cdot f_{C_1} = f_{C_1} \cdot \lambda_2 \Leftrightarrow f_{C_1}^T L f_{C_1} = |V| \cdot \lambda_2$ Relax iso constrain

P27 General case : $k>2$ / $h_C[i] = \begin{cases} \frac{1}{\sqrt{|C|}} & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$ $\Rightarrow H = [h_{C_1}; h_{C_2}; \cdots; h_{C_k}]$

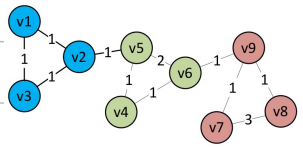
$\Rightarrow H^T H = Id$ / $h_{C_i}^T L h_{C_i} = \frac{cut(C_i, V \setminus C_i)}{|C_i|}$ and $h_{C_i}^T L h_{C_i} = (H^T L H)_{ii}$

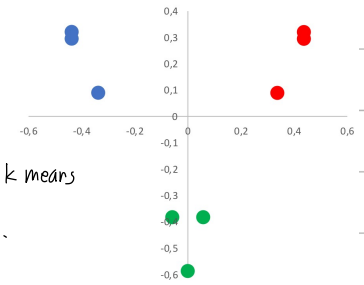
Ratio cut $= \sum_{i=1}^k \frac{cut(C_i, V \setminus C_i)}{|C_i|} = \sum_{i=1}^k (H^T L H)_{ii} = Tr(H^T L H)$

Minimize ratio cut = minimize $Tr(H^T L H)$ s.t. $H^T H = Id$

$\Rightarrow$ optimal H = first k smallest eigenvectors of L

Spectral Layout :





Solution

Unnormalized Laplacian $L = D - W \Rightarrow$ ratio cut

Normalized Laplacian $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$

$\Rightarrow$ normalized cut

x axis	y axis	
-0.3333	-0.4376	0.2939
-0.3333	-0.3370	0.0890
-0.3333	-0.4376	0.2939
-0.3333	0.0000	-0.5878
-0.3333	-0.0584	-0.3829
-0.3333	0.0584	-0.3829
-0.3333	0.4376	0.2939
-0.3333	0.4376	0.2939
-0.3333	0.3370	0.0890

result of k means

vertex 9 →

↑ constant, can be ignored.

P37 Inference in PPM : Likelihood of community assignment $z \in \{1,-1\}^N$ / observed symetric $A \in \{0,1\}^{N \times N}$

$$\Pr(\boldsymbol{A}|\boldsymbol{z}) = \prod_{i<j} \left[p^{A_{ij}}(1-p)^{1-A_{ij}}\right]^{\mathbb{I}(z_i=z_j)} \left[q^{A_{ij}}(1-q)^{1-A_{ij}}\right]^{\mathbb{I}(z_i \neq z_j)}$$

MLE : $z^* = \underset{z \in \{-1,1\}^N}{\operatorname{argmax}} \Pr(A|z)$

P38 PPM and Spectral Clustering : assume balanced cluster $|C_1| = |C_2| = \frac{N}{2} \Leftrightarrow \sum_i z_i = 0$

$$E_{cut}(\boldsymbol{z}) = |\{(i,j) \in E \text{ s.t. } z_i \neq z_j\}| = \sum_{(i,j)\in E} \mathbb{I}(z_i \neq z_j)$$

$$\Pr(\boldsymbol{A}|\boldsymbol{z}) \propto \left(\frac{(1-p)q}{(1-q)p}\right)^{E_{cut}(\boldsymbol{z})} \Leftarrow Pr(A_{ij} = 1|z_i, z_j) = \begin{cases} p & \text{if } z_i = z_j \\ q & \text{if } z_i \neq z_j \end{cases}$$

→ $p > q \Rightarrow$ Find a minimum balanced cut

$p < q \Rightarrow Pr > 1 \Rightarrow$ Find maximum cut

P41 Inference in SBM : Assume η are π are known $\Pr(\boldsymbol{z}|\boldsymbol{A}, \boldsymbol{\eta}, \boldsymbol{\pi}) = \frac{\Pr(\boldsymbol{A}|\boldsymbol{z}, \boldsymbol{\eta}, \boldsymbol{\pi}) \Pr(\boldsymbol{z}|\boldsymbol{\pi})}{\Pr(\boldsymbol{A}|\boldsymbol{\eta}, \boldsymbol{\pi})}$

use variational inference to find approximate solution

$$\Pr(\boldsymbol{z}|\boldsymbol{A}, \boldsymbol{\eta}, \boldsymbol{\pi}) \approx q(\boldsymbol{z}|\boldsymbol{\Psi}) = \prod_{i=1}^{N} q(\boldsymbol{z_i}|\boldsymbol{\psi_i})$$

Learning in SBM : If both z and A are observed, then MLE for η and π

$$\pi_k^{MLE} = \frac{\#\text{ nodes in cluster } k}{N} =: \frac{N_k}{N}$$

$$\eta_{uv}^{MLE} = \frac{\text{observed \# edges between } u \text{ and } v}{\text{possible \#edges between } u \text{ and } v} = \frac{\sum_{(i,j)\in E} \mathbb{I}(z_i = u)\mathbb{I}(z_j = v)}{P_{uv}}$$

where $P_{uv} = \begin{cases} \binom{N_u}{2} & \text{if } u = v \\ N_u \cdot N_v & \text{if } u \neq v \end{cases}$ is the number of possible edges between clusters u and v

If only A are observed, then use variation inference

05 _ Classification

P03 Introduction : Supervised learning problems

P04 collective classification : known → labeled nodes / classify → unlabeled nodes
↳ based on structure / edges / relationship

P05 Label Propagation : Consider binary case
- $V = S \cup U$ / S: Labeled instances S (seeds) ; U: Unlabeled instances
- $W \in R^{|V| \times |V|}$: Symmetric weighted adjacency matrix / $w_{ij} \geq 0$ = similiarity
- $\hat{y}_i \in \{0,1\}$ for $i \in S$: Given class label
- $y_i \in \{0,1\}$ for $i \in U$: Unknow class label

Idea: Energy Minimization : $E(y) = \frac{1}{2} \sum_{ij} w_{ij} (y_i - y_j)^2 \Leftrightarrow \min E(y)$ = min cut
- smoothness : adjacent nodes should have same class label
- matching the seed labels

$$\min_{y \in \{0,1\}^{|V|}} \frac{1}{2} \sum_{ij} w_{ij} (y_i - y_j)^2 \text{, s.t. } y_i = \hat{y}_i \text{ for all } i \in S$$

$$\rightarrow \min_{y \in \{0,1\}^{|V|}} y^T L y \text{, s.t. } y_i = \hat{y}_i \text{ for all } i \in S$$

($y \in R^{|V|}$, $L = D - W$)

solution : assume $L = \begin{bmatrix} L_{ss} & L_{su} \\ L_{us} & L_{uu} \end{bmatrix}$ min $y^T L y$

$$y = \begin{bmatrix} y_s \\ y_u \end{bmatrix} = \begin{bmatrix} \hat{y}_s \\ y_u \end{bmatrix} \Rightarrow y_u = -L_{uu}^{-1} L_{us} \hat{y}_s$$

K labels : one-hot $y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is of class } k \\ 0 & \text{else} \end{cases}$

$$E(Y) = \sum_{i,j} w_{ij} (y_i - y_j)^T (y_i - y_j)$$

Network effect H:

different type

$$E(Y) = \sum_{i,j} w_{ij} (y_i - y_j)^T H (y_i - y_j)$$

estimate paras. ↓

SBM vs LP ⇒ predict label

P10 Transductive Learning : ⇒ Label Propagation (LP)

Given : label training instances T / unlabeled test instances U
other piror knowledge

Goal : predict label only for unlabeled instances U (no new data!)

P12 Semi-supervised learning (SSL) ⇒ Given T, U , Goal: learn a mapping f
inductive ($f: X \to y$) / transductive ($f: U \to y$)

→ Unlabeled data help to model the data distribution

Motivation : Large unlabel data + small label data
label is expensive

P14 Attributed Graph : Each node with a feature vector
⇒ Markov random fields

P15 Graph construction : construct a graph connecting similiar data points
apply LP

P16 Graph classification : $G = (V, E)$ ⇒ classify V

↳ Given : a set of training graphs $\{G_i = (V_i, E_i)\}_{i=1,\dots,N}$
label y_i for each G_i

Goal : Learn a function $f: g \to Y$, map new graph with label

Method : Graph kernel function : $K: g \times g \to R$ / $K(G_1, G_2) = \phi(G_1)^T \phi(G_2)$
SVM / GNN ...

06 _ Node Embeddings

P03 Motivation : Want to apply traditional ML methods to graph
→ have no concept of graphs

P04 Learning Node Representation : Transform graph ⇒ vertex is represented by vector
Node embedding function : $\phi : V \to R^d$
close Node are similiar in embedding space

P05 Application : Clustering / Semi-supervised classification / Link prediction

P06 Type of Embedding : Spectral Embedding → nodes are similiar, if belong to cluster (same)
Deep Embedding → dependen on loss function and architecture

P09 Spectral Embedding : based on the eigenvectors of the graph Laplacian L (first k eigenvector of L)
$|V| \times |V|$ Adjacent matrix → $|V| \times k$ matrix H

P12 Deep Embedding : Capture more complex structure

Deep Walk : Graph ⇒ Random Walk / learn word 2 vec model
Every node sample multiple random walks
Train Word 2 vec
close node has similiar vector representation

Graph 2 Gauss : Learning f_θ to map node v_i to Gaussian distrib. / $v_i \to N(\mu_\theta(x_i), \Sigma_\theta(x_i))$ (attributed features)
1st node closer / 2nd node far away
Loss Function : $\sum_{(v,w)} E_{uv}^2 + e^{-E_{uw}}$ / $E_{uv} = KL(f_\theta(u) \| f_\theta(v))$

P15 Graph Embedding : Aggregate all node embedding into one graph embedding
→ mean pooling

07 _ GNN

P03 Motivation : Label propagation consider only graph structure

Real-world graph have attributes

GNN: NN operates on graph-structured data

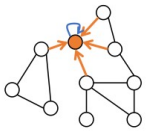
Adjacent matrix is not image fed into CNN

multiple GCN

Two versions: { Spatial GNNs / Spectral GNNs

P07 Spatial GNNs: Differentiable Message Passing Network Framework

Message passing = local aggregation



Given: $G=\{V,E\}$ and node $v=x_v \in R^d \Rightarrow G=(A,X)$

Even with edge features E_{vu}

$h_v^{(k-1)}$: hidden representation of node v at previous k-1 layer

Solve : Massage gather: $m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)}, h_u^{(k-1)}, E_{vu}) = \sum \frac{1}{d_v}(W^{(k)} h_u^{(k-1)} + b^{(k)})$

Hidden representation update: $h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)}) = relu(Q^{(k)} h_v^{(k-1)} + p^{(k)} + m_v^{(k)})$

U and M are trainable NN

P11 GCN : $X_i^{t+1} = X_i^{(t)} + \sum_{(i,j) \in E} X_j^{(t)}$ / $X^{(t+1)} = X^{(t)} + AX^{(t)}$

$H^{(L)} = [\quad]$ encode the hidden representations after layer L

$H^{(L+1)} = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}} H^{(L)} W^{(L)}) \Rightarrow \begin{cases} \hat{A} = A + I_n \\ \hat{D}_{ij} = \sum_j \hat{A}_{ij}, \text{ degree matrix} \end{cases}$

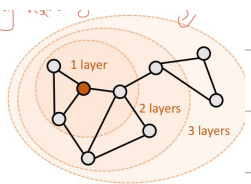
↳ Message passing ↳ Feature Transformation ↳ non linearity

At graph step k, node v aggregates informations from k-hop neighbours

↳ all nodes in its k-hop

Transformation depth and propagation depth are orthogonal

⇒ more layers / more node information



P15 Graph Signal Processing:

Given: Normalized Laplacian matrix $\hat{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Eigen decomposition: $\hat{L} = U \Lambda U^T$, $0 = \lambda_1 \leq \lambda_2 \cdots \leq \lambda_n \leq 2$

$U^T U = I$

Graph Signal: A function of nodes $x: V \to R$, $x \in R^N$

Graph fourier transformer: $\tilde{x} = U^T x$ / $x = U^T \tilde{x}$

Spectral filtering: $y = U g(\Lambda) U^T x$ / g: spectral filter

P16 Spectral Graph NN: $H^{(l+1)} = \phi(U g(\Lambda) U^T \varphi(H^{(l)}))$ $g_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k \Lambda^k$ ∈ polynomial filter

MLP

$= \phi(g_\theta(\hat{L}^k) \varphi(H^{(l)}))$

Computational complexity: avoid eigen decomposition

Receptive field size: whole graph not the neighbour

Transferability:

Application of GNN

P19 Semi-Supervised Node Classification: probabilty of node v belong to class C

$\Rightarrow p_v = \text{softmax}(h_v^{(k)})$

$\Rightarrow$ use CE $\min_{\cdots} - \sum_{v \in S} \sum_{c \in C} y_{vc} \log p_{vc}$

P20 Graph Classification: $H_{G_i} = [h_1^{(k)}, \cdots, h_{|V_i|}^{(k)}]$ of $G_i = \{V_i, E_i\}$

$\Rightarrow$ Aggration function $h_{G_i} = R(H_{G_i}) = \frac{1}{|V_i|} \sum_{v \in V_i} h_v^{(k)}$

$\Rightarrow$ use CE $\min_{\cdots} L(y_{G_i}, \text{softmax}(h_{G_i}))$

P21 Equivariant Machine Learning: Given $G_1 = (A_1, X_1)$, $G_2 = (A_2, X_2)$

isomorphic $\Rightarrow$ exist a permutation matrix P, let $PA_1P^T = A_2$, $PX_1 = X_2$

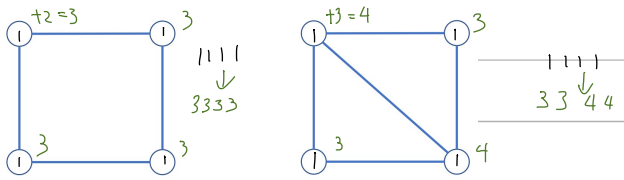
Graph isomorphism $\Rightarrow$ invariance $\Rightarrow f(PX, PAP^T) = f(X, A)$

equivarianve $\Rightarrow f(PX, PAP^T) = Pf(X, A)$

E(n) equivariant: $E(N) = \{Rx + T : R \in O(N)\}$

P24 Limitations of GNN: Expressiveness / Reliability (Non-Robust) / Scalability

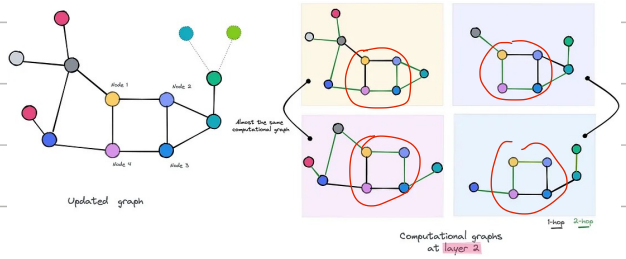
P26 Weisfeiler-Lehman Test:

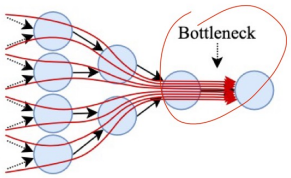


⇒ can distinguish many non-isomorphic graph but not all of them

P28 Oversmoothing: Infinite layers ⇒ stationary distribution

FCN Graph





[Alon2021]

P30 Overqnashing: long distance ⇒ in one node ⇒ bottelneck

tree-based Graph

P31 Page Rank:

Page Rank: $r^{(t+1)} = AD^{-1} \; r^{(t)}$

GCN: $H^{(l+1)} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}} \; \sigma(H^{(l)}W^{(l)})$

Message Passing | Features

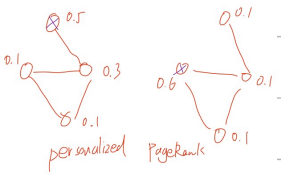
P33 Personalized Propagation of Neural Prediction (PPNP): Personalized teleport $\pi = e_i$

↳ Seperate Transformation and Propagation

① $H^{(0)} = X \Rightarrow H^{(0)}_{i,:} = f_\theta(x_{i,:})$

② $r^{(t+1)} = (1-\alpha)AD^{-1}r^{(t)} + \alpha\pi \;/\; H^{(l+1)} = (1-\alpha)\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)} + \alpha H^{(0)}$

③ PPNP for Node Classification ⇒ $Z = \text{softmax}\left[\alpha(I_N - (1-\alpha)\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})^{-1}H^{(0)}\right]$



prevent oversmoothing / seperation of Transformation and Propagation / Scalability

P37 APPNP: $Z = \text{softmax}\left[\alpha(I_N - (1-\alpha)\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})^{-1}H^{(0)}\right] \Rightarrow Z = H^{(k)}$

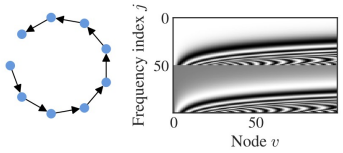
P38 Graph Transformers ⇒ Positional Encodings

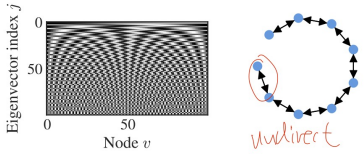
P39 Sequence problem : Sinusoidal Encodings $PE \in \mathbb{R}^{n \times d}$

positional encodings ⇐ Discrete Fourier Transformation

$$\hat{x}_j = \sum_{v=0}^{n-1} x_v \left[\cos\left(v \cdot \frac{2\pi}{n} j\right) - i \cdot \sin\left(v \cdot \frac{2\pi}{n} j\right)\right]$$

- Sinusoidal Encodings ↔ Eigenvectors of Laplacian





unlirect

$$\mathrm{PE}^{(\sin)}_{v,j+d/2} = \sin\left(v \cdot 1/10,000^{2j/d}\right)$$
$$\mathrm{PE}^{(\sin)}_{v,j} = \cos\left(v \cdot 1/10,000^{2j/d}\right)$$

Closed form for $L = D - A = U\Lambda U^T$ of a sequence:

$$U = \mathrm{PE}^{(\mathrm{lap})}_{v,j} = \pm \cos\left(v \cdot \frac{j\pi}{n} + \frac{j\pi}{2n}\right)$$

Eigenvalue

P41 Direct Graph : $L = D - W$ / $\hat{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ are real symmetric

if A is symmetric and graph is undirected

⇒ Direction-aware Magnetic Laplacian $L^{(q)}$ ⇔ encode edge directions

$$L^{(q)} := D_S - A_S \odot exp(i\Theta^{(q)}) \qquad \hat{L}^{(q)} := I - D_S^{-1/2} A_S D_S^{-1/2} \odot exp(i\Theta^{(q)})$$

$$\Theta^{(q)}_{u,v} := 2\pi q (A_{u,v} - A_{v,u})$$ direction