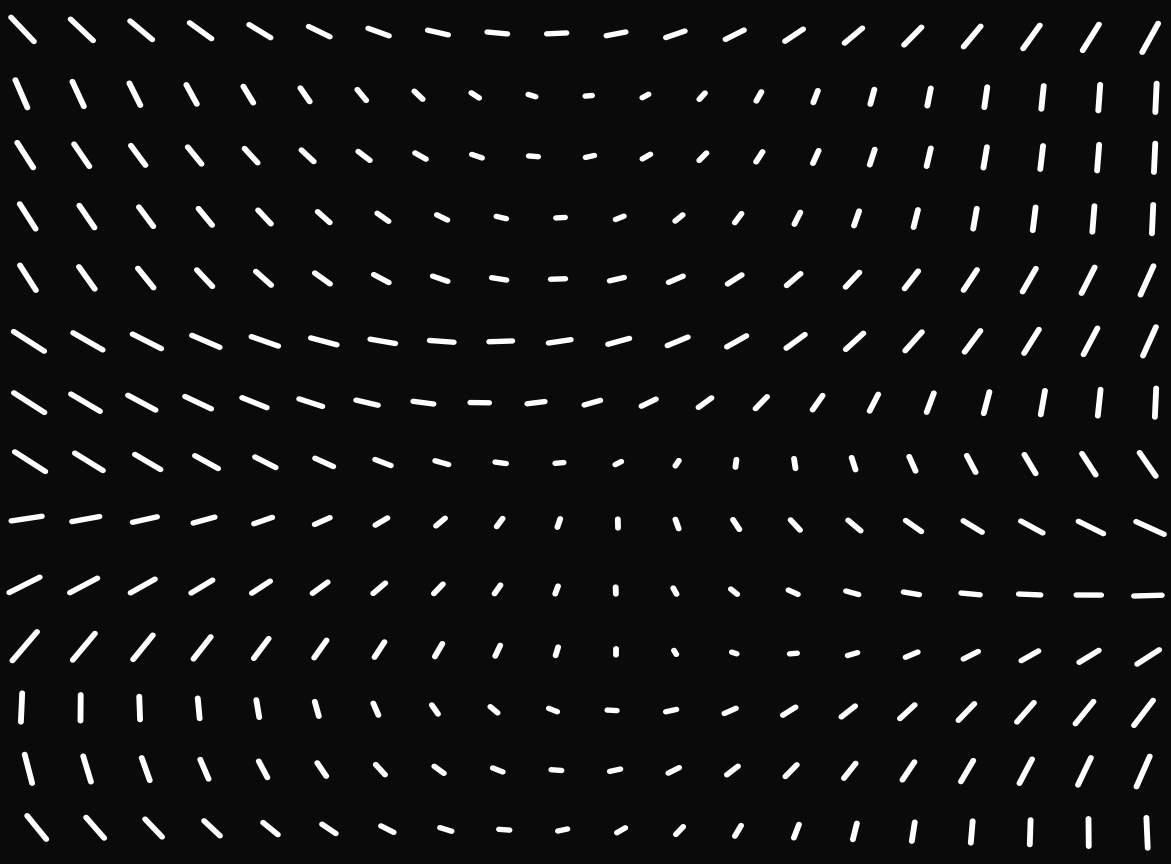
Machine learning for graph and sequential data

Notes, exercise solutions and previous year exams



Chapter 1 - Normalizing Flows:

probability distribution $p(x)$

ie $p(x)$ where $x = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$

① Data sampling: generate x_i following distr. $p(x)$

② Density evaluation: given any x_i compute the prob. density at this point $p(x_i)$

Main Idea

* NF can model flexible distributions for data sampling and density evaluation.

Idea → • Normalizing Flows are based on change of variable
→ • useful in simple computation like integration.

the probability in input space should be the same as in the outer space

$$p_2(x) = p_1(z) \left| \frac{\partial z}{\partial x} \right|$$

ex 1) shift $p_1(z) = \text{Unif}([0,1]^2)$, $f(z) = z + \text{shift} = x$

$p_2(x) = p_1(z)$

ex 2) Linear tr. $p_1(z) = \text{Unif}([0,1]^2)$, $f(z) = Mz = x$

$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$p_2(x) = p_1(z) \frac{1}{\det(M)}$

Change of variables: if $D \in \mathbb{N}$, $p_1(z)$ a D-dimensional distribution $f(z) = x$ is an invertible and differentiable transformation, then distribution $p_2(x)$ is

$$p_2(x) = p_1(f^{-1}(x)) \left| \det\left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

- determinant term : accounts for "distortion rate"
- $\frac{\partial f(x)}{\partial x}$: Jacobian ; D×D matrix $\frac{\partial f^{-1}(x)}{\partial x} = \left(\frac{\partial f(x)}{\partial x}\right)^{-1}$
- tranformation f should be valid (invertible and differentiable).

conditions : (sufficient) condition for valid transform

- invertibility : → i/p & o/p mapping have same dimension
  - if D = 1, sufficient if f is strictly monotonic
  - if the transformation f is linear, its determinant should be nonzero.
- differentiability → Both f & f^{-1} are continuously differentiable
  - can be violated in practice, but we assume this.

stacking : stacking transformations f_i :

apply change of variable multiple times
- the first variables z_0 is transformed by $z_1 = f(z_0)$
- the variables z_{i-1} is transformed $z_i = f(z_{i-1})$
- the last variable z_{K-1} is transformed by $x = z_K = f_K(z_{K-1})$

$$p_K(x) = p_0(z_0) \prod_{i=1}^{K} \left| \det \left(\frac{\partial f_i^{-1}(z_i)}{\partial z_i} \right) \right|$$

Log version

$$\log(p_K(x)) = \log(p_0(z_0)) + \sum_{i=1}^{K} \log \left| \det \left(\frac{\partial f_i^{-1}(z_i)}{\partial z_i} \right) \right|$$

Forward and Reverse Parametrization :

the change of variable does a mapping b/w two distro.

$$p_2(x) = p_1(f^{-1}(x)) \cdot \left| \det\left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

- Reverse Parametrization : evaluate $p_2(x)$ at x
- Forward Parametrization : sample points from $p_2(x)$

Reverse Parametrization :

parametrize inverse transform $g = f^{-1}$

- $g_\phi(x) = z$ is computable and parameter ϕ can be learned.

$f = g^{-1}$ might not be easily computable

$$p_2(x) = p_1(g_\phi(x)) \left| \det\left(\frac{\partial g_\phi(x)}{\partial x} \right) \right|$$

Stacking : $g_\phi = g_{\phi_1} \circ \cdots \circ g_{\phi_K}$

to compute density :

1. For any $x^{(i)}$, we can get $x^{(i)} = z_K$
2. Compute transformations $z_{i-1}^{(i)} = g_{\phi_i}(z_i^{(i)})$ and $\left| \det\left(\frac{\partial \phi_i(z_i^{(i)})}{\partial z_i^{(i)}} \right) \right|$
3. $z_0^{(i)}$, we compute $p_0(z_0^{(i)})$ / gaussian or uniform and thus $p_K(x^{(i)})$

- we aim to learn the transformation g_ϕ
- we can call the distro. $p_\phi(x)$ to make the dependency on ϕ clear.

Learning :

$\mathcal{D} = \{x^{(j)}\}_{j=1}^{N}$ (considering iid)

$$\max_{\phi} \log p_\phi(\mathcal{D}) = \max_{\phi} \frac{1}{N} \sum_{x^{(j)} \in \mathcal{D}} \log p_\phi(x^{(j)})$$

Forward Parameterization :

- Parameterize the transformation f that we know analytically
  - $f_\theta(z) = x$ is computable and θ is learned.
- we know that the inverse f^{-1} exists, but we might not know it analytically.
  - $f_\theta^{-1}(x) = z$ might not be (easily) computable
- the change of variable

$$p_2(x) = p_1(z) \left| \det \left(\frac{\partial f_\theta(z)}{\partial z} \right)^{-1} \right|$$

  - the formula only uses the known parameterization function f_θ
  - given a sample $z^{(j)}$, we can compute samples $x^{(j)} \sim p_2(x)$ & $p_2(x^{(j)})$
- we can also stack transformations and forward param.

$$f_\theta = f_{\theta_K} \circ \cdots \circ f_{\theta_1}$$

- forward parameterization enables sampling from distro $p_{\theta}(x)$
  1. sampling $z_0^{(j)} \sim p_0(z_0)$ (eg. gaussian & uniform)
  2. compute the transform.

$$z_i^{(j)} = f_{\theta_i}(z_{i-1}^{(j)}) \text{ and } \left| \det \left(\frac{\partial f_{\theta_i}(z_{i-1}^{(j)})}{\partial z_{i-1}^{(j)}} \right) \right|$$

  3. For the particular sample $x^{(j)} = z_K^{(j)}$, we can compute $p_K(x^{(j)})$

Forward pointer ; this is exactly what we need in VI.

- sample x from q and compute prob. $q(x)$

Jacobian Determinent Computation :

$$J_g = \begin{bmatrix} \frac{\partial g_1(x)}{\partial x_1} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \vdots & & \\ \frac{\partial g_n(x)}{\partial x_1} & \cdots & \frac{\partial g_n(x)}{\partial x_n} \end{bmatrix}$$

① $|A^{-1}| = \frac{1}{|A^{-1}|}$ ② $|A| = \prod_{i=1}^{n} a_i$

③ $\begin{vmatrix} B & 0 \\ 0 & C \end{vmatrix} = |B|\,|C|$

Diagonal Jacobian

① $J_g = \begin{bmatrix} \frac{\partial g_1(x_1)}{\partial x_1} & & 0 \\ & \ddots & \\ 0 & & \frac{\partial g_D(x_D)}{\partial x_D} \end{bmatrix}$

② $|J_g| = \prod_{i=1}^{D} \frac{\partial g(x_i)}{\partial x_i}$

Triangular Jacobian

$g(x) = \begin{bmatrix} g_1(x_1) \\ \vdots \\ g_D(x_1, \ldots x_D) \end{bmatrix}$

① $J_g = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & & 0 \\ \vdots & & \\ \frac{\partial g(x_1, \ldots x_D)}{\partial x_1} & \cdots & \frac{\partial g(x_1 \ldots x_D)}{\partial x_D} \end{bmatrix}$

② $|J_g| = \prod_{i=1}^{D} \frac{\partial g(x)}{\partial x_i}$

Full Jacobian

$J_g = LV \qquad |J_g| = |L||V|$

Questions

1. Is $f(z) = 1 - z$ a valid transformation?

1) $f(z) = 1 - z$ is bijective mapping and differentiable

$z = 1 - f(z) \qquad f^{-1}(z) = f(x) = 1 - x$

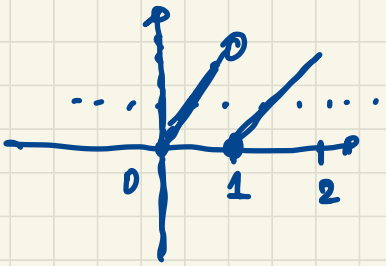
2. Is $f(z) = 2 - 3z$ a valid transformation?

2) $f(z) = 2 - 3z$ if bijective and is cont. and differentiable $\forall z$.

$\therefore$ it is a valid mapping

3. Is $f(z) = \begin{cases} -z, z \in [0,1[\\ 1 - z, z \in [1,2] \end{cases}$ a valid transformation?

3)



the transform is not a bijection, also it is not cont. & differentiable hence not a valid transform

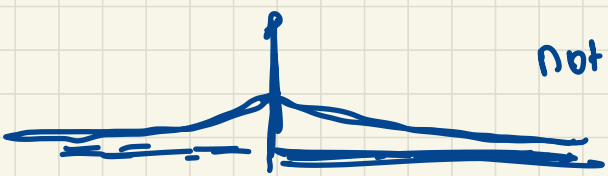
1. For which x is it possible to compute $p(x)$ with the forward parametrization?

1) the point x can be used to calculate the density for a point x for which corresponding $z = g^{-1}(x)$

2. Suppose the forward transformation is defined as $f(z) = \exp(-z^n)$.
What is the corresponding inverse transformation $g(x)$?
Does $g(x)$ exist for any n?

2) there will be two cases, $n = \begin{cases} \text{even} \\ \text{odd.} \end{cases}$

for n → even



not bijective

3. Suppose the forward transformation is defined as the sigmoid function.
What is the corresponding inverse transformation?

3) $f(z) : \frac{1}{1+e^{-z}}$ continuous and differen.

$$1 + e^{-z} = \frac{1}{f(z)}$$

$$1 - \frac{1}{f(z)} = e^{-z}$$

$$\ln\left(1 - \frac{1}{f(z)}\right) = -z$$

$$z = \ln(f(z)) - \ln(1 - f(z))$$

1. Let's assume you get the following Jacobian:

$$\begin{bmatrix} \frac{\partial g_1(x_1)}{\partial x_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial g_D(x_1, \ldots, x_D)}{\partial x_1} & \cdots & 0 \end{bmatrix}$$

How expensive is it to compute the determinant?
Can you comment on this in the context of NFs?

determinant is = 0.
∴ O(1)

2. What is the complexity to compute the Jacobian determinant of an arbitrary valid transformation?

$O(D^3)$ → complexity of computing an arbitrary determinant.

3) Considering high-dimensional data (i.e. D is high), what type of Jacobian would you use?

3) triangular / Diagonal Jacobian

Problem 1: Consider the following transformation $f: \mathbb{R}^3 \to \mathbb{R}^3$

$$f(z) = \begin{bmatrix} 2z_1 \\ e^{z_1} z_2 \\ e^{-z_1 - z_2} z_3 \end{bmatrix}.$$

Prove or disprove whether the transformation f is invertible.

$$f\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} 2z_1 \\ e^{z_1} z_2 \\ e^{-z_1 - z_2} z_3 \end{bmatrix}$$

$f(z_2) = e^{z_1} z_2$

$z_2 = e^{-z_1} \cdot f(z_2)$.

$z_2 = e^{-f(z_1)/2} f(z_2)$

$f^{-1}(z_2) = e^{-z_1/2} z_2$ — ②

$f(z_1) = 2z_1$

$f^{-1}(z_1) = z_1/2$ — ①.

$f(z_3) = e^{-z_1 - z_2} \cdot z_3$

$z_3 = e^{z_1 + z_2} \cdot f(z_3)$

$z_3 = f(z_3) e^{\frac{f(z_1)}{2}} + e^{-\frac{f(z_1)}{2}} \cdot f(z_3)$

$f^{-1}(z_3) = z_3 e^{\frac{z_1}{2}} + e^{-z_1/2} z_2$ — ③

$$f^{-1}\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} z_1/2 \\ e^{-z_1/2} z_2 \\ z_3 e^{z_1/2} + e^{-z_1/2} \cdot z_2 \end{bmatrix}$$

There inverse is a valid mapping

Problem 2: Consider the following transformation $f: \mathbb{R}^2 \to \mathbb{R}^2$:

$$f(z) = \begin{bmatrix} z_1^2 z_2 \\ z_2^3 \end{bmatrix}.$$

Prove or disprove whether the transformation f is invertible.

$$f\left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) = \begin{bmatrix} z_1^2 z_2 \\ z_2^2 \end{bmatrix}$$

$f(z_2) = z_2^2$

$f(z_2) = \pm\sqrt{z_2}$ — not a unique image,

∴ Therefore it is not a valid transformation

violates the one to one mapping condition

Problem 3: Consider the tranformation $f(z) = Az + b$ from $\mathbb{R}^2$ to $\mathbb{R}^2$, where $A \in \mathbb{R}^{2\times 2}$ and $b \in \mathbb{R}^2$. Under what conditions on A and b is this tranformation invertible? Justify your answer.

$$f(z) = Az + b$$

$$\frac{\partial f(z)}{\partial z} = A^T$$

$$\det\left(\frac{\partial f(z)}{\partial z}\right) = \det(A^T) = \det(A)$$

necessary conditions of invertibility in case of linear transform is that $\det(J) \neq 0$.

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0 \Rightarrow \boxed{a_{11}a_{22} - a_{12}a_{21} \neq 0.}$$

Problem 4: We consider the following forward tranformation $f: \mathbb{R}^3 \to \mathbb{R}^3$

$$\boldsymbol{x} = f(\boldsymbol{z}) = \begin{bmatrix} z_1 \\ e^{z_1} z_2 \\ \sqrt[3]{e^{-z_1} z_3 + z_1^2} \end{bmatrix}.$$

We assume a uniform base distribution $p_1(\boldsymbol{z}) = U([0,2]^3)$. Evaluate the density $p_2(\boldsymbol{x})$ at the points $\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 1 \\ \frac{1}{3} \end{bmatrix}$ and $\boldsymbol{x}^{(2)} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$.

4) $p_2(x) = p_1(f^{-1}(x_1)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|$

$z_1 = x_1$
$z_2 = x_2 e^{-x_1}$
$z_3 = (x_3^3 - x_1^2) e^{x_1}$

$f^{-1}(x) = \begin{bmatrix} x_1 \\ x_2 e^{-x_1} \\ e^{x_1}(x_3^3 - x_1^2) \end{bmatrix}$

$\frac{\partial f^{-1}(x)}{\partial x} = \begin{bmatrix} 1 & 0 & 0 \\ -x_2 e^{-x_1} & e^{-x_1} & 0 \\ * & * & 3x_3^2 e^{x_1} \end{bmatrix}$

$1 \cdot (e^{-x_1} \cdot 3x_3^2 e^{x_1}) = 3x_3^2$

$f_1^{-1}(x^{(1)}) = \begin{bmatrix} 0 \\ 1 \\ 1/27 \end{bmatrix} = z^{(1)}$

$f^{-1}(x^{(2)}) = \begin{bmatrix} -1 \\ 2e \\ 0 \end{bmatrix} = z^{(2)}$

$$p_2(x^{(1)}) = p_1(f^{-1}(x^{(1)})) \left| \det \frac{\partial f^{-1}(x^{(1)})}{\partial x} \right|$$

$$= \frac{1}{23} \cdot 3 \cdot \frac{1}{3^2} = \boxed{\frac{1}{24}}$$

$$p_2(x^{(2)}) = p_2(f^{-1}(x^{(2)})) \left| \det \frac{\partial f^{-1}(x^{(2)})}{\partial x} \right|$$

$$= \frac{1}{2^3} \cdot 2 \cdot 0 = \boxed{0}$$

Problem 5: We consider the following forward transformation $x = f(z) = \sum_{k=1}^{K} \sigma(kz)$ from $\mathbb{R}$ to $(0, K)$ with $\sigma(z) = \frac{1}{1+e^{-z}}$. We assume a Gaussian base distribution $p_1(z) = \mathcal{N}(0, 1)$. We sampled one point from the base distribution $z^{(1)} = 0$. Compute the corresponding sample $x^{(1)}$ from the transformed distribution and evaluate its density $p_2(x^{(1)})$.

5) $$f(z^{(1)}) = \sum_{k=1}^{K} \sigma(k \times 0) = \frac{K}{2}$$

$$\left| \det \left(\frac{\partial f(z)}{\partial z} \right) \right| = f'(z) = \sum_{k=1}^{K} k\,\sigma(kz)(1 - \sigma(kz))$$

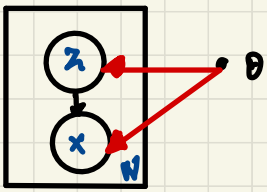
$$p_2(x^{(1)}) = p_1(z^{(1)}) \left| \det \left(\frac{\partial f(z)}{\partial z} \right) \right|^{-1}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-(0)^2/2} \cdot \frac{1}{\sum_{k=1}^{K} k\,\sigma(k \times 0)(1 - \sigma(k \times 0))}$$

$$= \frac{1}{\sqrt{2\pi}} \frac{4}{\sum_{k=1}^{K} k} = \boxed{\frac{8}{\sqrt{2\pi}\, K(K+1)}}$$

Chapter 2 : Variational Inference

Latent variable Models (LVMs)



1) generate latent variable $z \sim p_\theta(z)$
2) generate data x conditional on z $x \sim p_\theta(x|z)$

Joint probability distribution : $p_\theta(x,z) = p_\theta(z)\, p_\theta(x|z)$

Marginal Distribution : $p_\theta(x) = \int p_\theta(x,z)\, dz = \int p_\theta(z)\, p_\theta(x|z)\, dz$

$= \mathbb{E}_{z \sim p_\theta(z)}[p_\theta(x|z)]$

Tasks in LVMs :

1) Inference : $p_\theta(z|x) = \dfrac{p_\theta(x|z)\, p_\theta(z)}{p_\theta(x)}$

2) Learning : $\max_\theta \log p_\theta(x) \approx \max_\theta \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i)$

iid assumption

- Normally $\int p_\theta(x,z)$ doesn't have a closed form solution, numerical integration is infeasible.

we can use Normalizing flows :

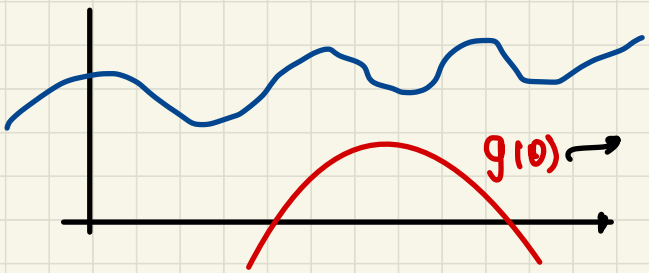
using reverse parametrization with parametric function $g_\theta(x)$ and base distribution p_1 we obtain

(to evaluate $p_\theta(x)$ at a point.

reverse parametrization with parametric function $g_\theta(x)$ and base distribution p_1 we obtain

$$\max_\theta \log p_\theta(x) = \max_\theta \left[\log p_1(g_\theta(x)) + \log \left| \det \left(\frac{\partial g_\theta(x)}{\partial x} \right) \right| \right]$$

Maximization using Lower Bounds :

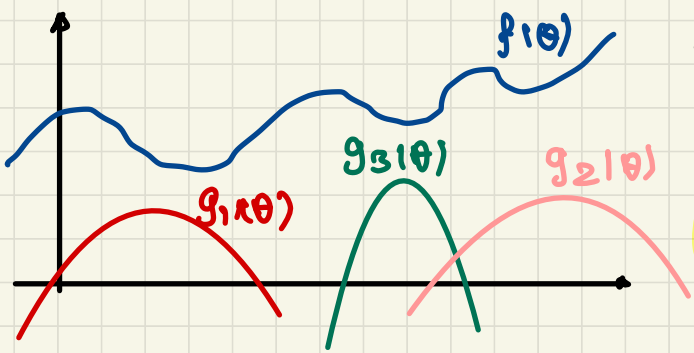


→ f & ∇f(θ) are intractable.

→ find a nice g(θ) that is lower bound of f.

- Maximizing g(θ) would give us lower bound on the solution

$$\max_\theta f(\theta) \geq \max_\theta g(\theta)$$



- using multiple lower bounds is even better.

$$\max_\theta f(\theta) \geq \max_{g \in \mathcal{G}} \max_\theta g(\theta)$$

algorithm: approx. solving $\max_\theta f(\theta)$ ~~→~~ intractable f

1. construct lower bound $g(\theta)$ s.t. $f(\theta) \geq g(\theta)$ $\forall g \in \mathcal{G}$ & $\forall \theta$
2. solve optimization problem $\max_{g \in \mathcal{G}, \theta} g(\theta)$

Lower Bound for Marginal log likelihood

$$\log p_\theta(x) = \mathbb{E}_{z \sim q(z)}\left[\log \frac{p_\theta(x,z)}{q(z)}\right] + KL\left(q(z) \,||\, p_\theta(z|x)\right)$$

arbitrary dist.

Kullback - Leibler Distribution

$$KL(q(z)\,||\,p(z)) = \int q(z) \log \frac{q(z)}{p(z)}\, dz$$

Properties:
- asymmetric $KL(q(z)||p(z)) \neq KL(p(z)||q(z))$
- non negative
- if $KL(p(z)||q(z)) = 0 \Rightarrow p(z) = q(z)$ almost everywhere

Evidence Lower BOund (ELBO)

$$\log p_\theta(x) = \mathbb{E}_{z\sim q(z)}\left[\log \frac{p_\theta(x,z)}{q(z)}\right] + KL(q(z)\,||\,p(z))$$

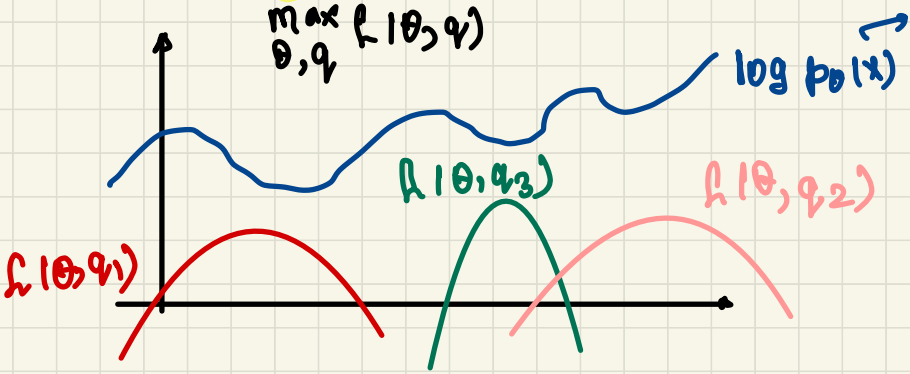
≥ 0

$\mathcal{L}(\theta, q)$ → lower bound of $\log p_\theta(x)$ → called evidence

* Tightness depends on how close $q(z)$ is to posterior $p_\theta(z|x)$ in terms of KL divergence

Note: Different choices of $q(z)$ leads to different lower bounds

objective: param θ and distribution $(q(z))$ that maximizes lower bound



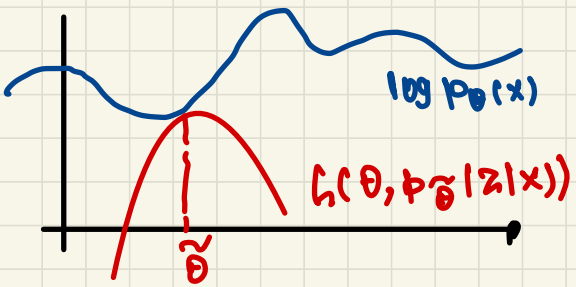
Alternative interpretation of the ELBO

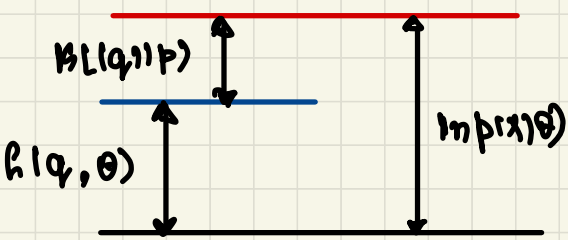
$$\log p_\theta(x) = \mathcal{L}(\theta, q) + KL(q(z) \,||\, p_\theta(z|x))$$

$$\Rightarrow \mathcal{L}(\theta, q) = -KL(q(z)||p_\theta(z|x)) + \log p_\theta(x)$$

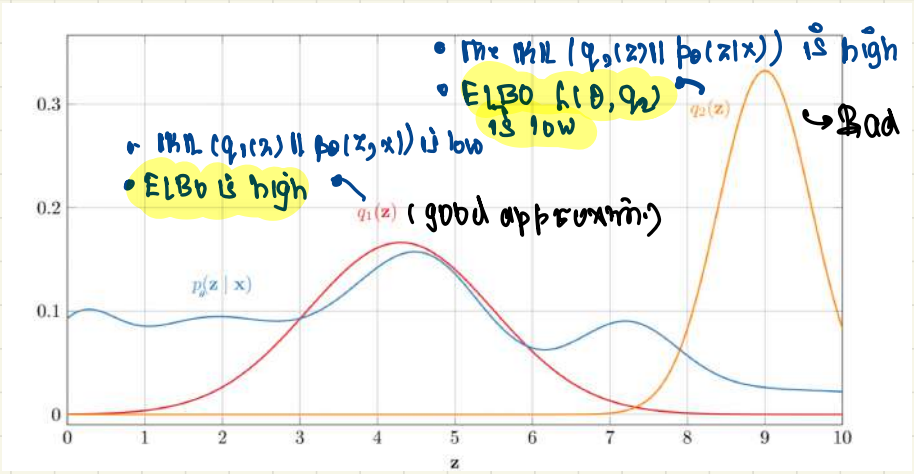
- for fixed θ, $q(z) = p_\theta(z|x)$

makes ELBO exactly equal to $\log p_\theta(x)$





for a fixed θ : maximizing the ELBO w.r.t q is equivalent to making $q(z)$ as close as possible to $p_\theta(z|x)$



#EM algorithm and variation Inference

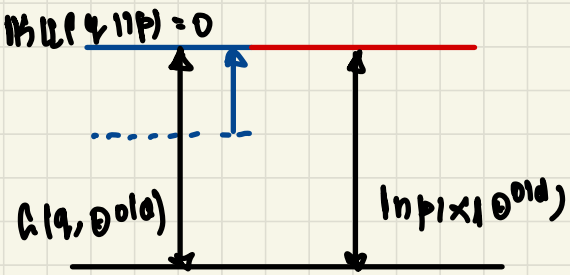
- true posterior $p_\theta(z|x)$ → intractable
- EM algorithm
  - E step: $q(z) = p_\theta(z|x)$
  - M step: $\theta^{new} = \arg\max_\theta \mathbb{E}_{z\sim q(z)}[\log p_\theta(x,z)]$

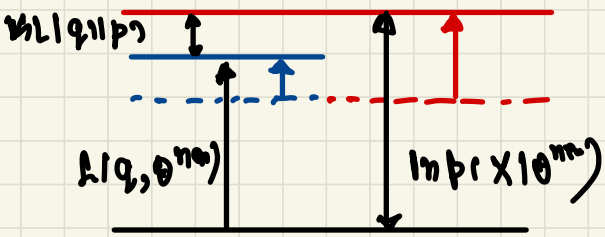
E-step

$$q(z) = p_\theta(z|x)$$
$$= \arg\min_q KL(q(z)\|p_\theta(z|x))$$
$$= \arg\max_q f(\theta, q)$$

M-step

$$\theta^{new} = \arg\max_\theta \mathbb{E}_{z\sim q(z)}[\log p_\theta(x,z)]$$
$$= \arg\max_\theta f(\theta, q)$$





optimizing the ELBO:

$$\max_{\theta, q} f(\theta, q)$$

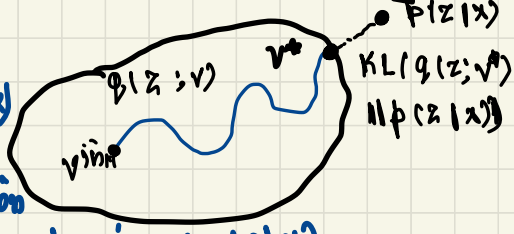
$q(z)$ is prob. distr.
∴ 1) what is the domain
2) How to opt. w.r.t a prob. distr.

Parametric family of distributions:

Pick tractable family of distribution Q
- 1) we can draw samples
- 2) every distribution in Q is specified over a parameter vector ϕ.

our problem becomes:

$$\max_{\theta \in \mathbb{R}^M, \phi \in \mathbb{R}^K} \mathbb{E}_{z \sim q_\phi(z)} \left[\log p_\theta(x,z) - \log q_\phi(z) \right]$$



1) max ELBO = min $KL(q_\phi(z) \| p_\theta(z|x))$
2) true posterior intractable
3) optimizing over ϕ → distribution $q \in Q$ that is closest to true posterior $p_\theta(z|x)$

$$\max_{\theta,\phi} \mathbb{E}_{z \sim q_\phi(x)} \left[\log p_\theta(x,z) - \log q_\phi(z) \right] = \max_{\theta;\phi} L(\theta, \phi)$$

use gradient ascent → need to compute $\nabla_\theta L(\theta, \phi)$, $\nabla_\phi L(\theta, \phi)$

In some cases integral can be calculated, but in case it can't be

→ assume ϕ is fixed

① Approximating $\nabla_\theta L(\theta, \phi)$:

$$\mathbb{E}_{z \sim q_\phi(z)}[f_\theta(z)] = \frac{1}{S} \sum_{i=1}^{S} f_\theta(z_i) \quad z_i \sim q_\phi(z) \text{ for } i = 1, \dots S$$

$$\nabla_\theta \mathbb{E}_{z \sim q_\phi(z)}[f_\theta(z)] = \nabla_\theta \int q_\phi(z) f_\theta(z) \, dz$$

$$= \int q_\phi(z) \nabla_\theta f_\theta(z) \, dz = \mathbb{E}_{z \sim q_\phi(z)}[\nabla_\theta f_\theta(z)] = \frac{1}{S} \sum_{i=1}^{S} \nabla_\theta f_\theta(z_i) \quad z_i \sim q_\phi(z)$$

② Approximating $\nabla_\phi \mathcal{L}(\theta, \phi)$: → assume θ is fixed

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z)}[h_\phi(z)] = \int q_\phi(z)\, h_\phi(z)\, dz$$

In our case

$$h_\phi(z) = \log p_\theta(x, z) - \log q_\phi(z)$$

* cannot "push" the gradient inside integral

Reparameterization Trick:

- sampling from many distributions $q_\phi(z)$ → can be represented as a deterministic transformation $T(\epsilon, \phi)$ of some base distribution $b(\epsilon)$
- $q_\phi(z) = \mathcal{N}(z \mid \mu, RR^T)$ multivariate normal

sampling → Drawing sample from $\epsilon \sim \mathcal{N}(0, I)$

$z = T(\epsilon, \phi = \{\mu, R\}) = R\epsilon + \mu$

distribution $b(\epsilon)$ does not depend on ϕ.

$$\mathop{\mathbb{E}}_{z \sim q_\phi(z)}[h_\phi(z)] = \mathop{\mathbb{E}}_{\epsilon \sim b(\epsilon)}[h_\phi(T(\epsilon, \phi))]$$

$$\nabla_\phi \mathop{\mathbb{E}}_{\epsilon \sim b(\epsilon)}[h_\phi(T(\epsilon, \phi))] = \int b(\epsilon) \nabla_\phi h_\phi(T(\epsilon, \phi))\, d\epsilon$$

$$= \mathop{\mathbb{E}}_{\epsilon \sim b(\epsilon)}[\nabla_\phi h_\phi(T(\epsilon, \phi))] \approx \frac{1}{S} \sum_{i=1}^{S} \nabla_\phi h_\phi(T(\epsilon_i, \phi))$$

$\epsilon_i \sim b(\epsilon)$

Putting it all together

1. We define a latent variable generative model for our data $\boldsymbol{x}$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) d\boldsymbol{z}$$

2. We are interested in maximum likelihood estimation of model parameters $\boldsymbol{\theta}$

$$\max_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$$

3. We can obtain a lower bound on $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ using some distribution $q(\boldsymbol{z})$

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}(\boldsymbol{\theta}, q) \coloneqq \mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z})]$$

4. Our original optimization problem can be approximately solved as

$$\max_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \max_{\boldsymbol{\theta}, q} \mathcal{L}(\boldsymbol{\theta}, q)$$

5. We pick a parametric family of variational distributions $\mathcal{Q}$

$$\mathcal{Q} = \{q_{\boldsymbol{\phi}}(\boldsymbol{z}) \text{ for } \boldsymbol{\phi} \in \mathbb{R}^K\}$$

6. Our optimization problem now becomes

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) \coloneqq \max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z})\right]$$

7. We obtain the gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ and $\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ using Monte Carlo (with the reparametrization trick)

8. We find $\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star$ by maximizing our objective function using gradient ascent

Mean Field assumption

In practice we make a simplifying assumption that $q(z)$ factorizes:

$$q(z) = \prod_{i=1}^{N} q_i(z_i)$$

$$\frac{1}{N} \mathbb{E}_{z \sim q(z)} \left[\sum_{i=1}^{N} \log p_\theta(x_i, z_i) - \log q(z) \right]$$

$$= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{z_i \sim q_i(z_i)} \left[\log p_\theta(x_i, z_i) - \log q_i(z_i) \right]$$

Questions

1. Assume that we have an LVM, where $p_\theta(x|z)$ and $p_\theta(z)$ are tractable (i.e. we can compute them). Can it happen, that we can also compute $p_\theta(z|x)$ for this model, but cannot compute $p_\theta(x)$? Why or why not?

1) $p_\theta(z|x) = \dfrac{p_\theta(x|z)\, p_\theta(z)}{p_\theta(x)}$ (Bayes' Theorem)

- No, this cannot happen.

2. Why is it always possible to compute $\log p_\theta(x)$ in a latent variable model, where z can take only finitely many values?

2) $\log p_\theta(x) = \sum_{c=1}^{C} \log p_\theta(x, z = c)$

$$= \sum_{c=1}^{C} \log \left[p_\theta(x|z=c)\, p(z=c) \right]$$

1. Slide 57: Why is it necessary for all functions $g \in \mathcal{G}$ to be lower bounds on f? What happens if some functions in $\mathcal{G}$ are not lower bounds?

The $\max_{g \in \mathcal{G}, \theta} g(\theta)$ will be meaningless

2. Slide 57: Can we use a similar approach if we want to approximately minimize some intractable function f? What changes need to be done in this case?

either we can use $-f$ or we take upper bounds and minimize them

3. Assume that $p_\theta(z|x)$ is a distribution on $[0, \infty)$ (e.g. exponential distribution), and our variational distribution $q(z)$ is a distribution on all of $\mathbb{R}$ (e.g. normal distribution).
   - What happens to the ELBO in this case?
   - Why is the optimization problem of maximizing the ELBO ill-defined?
   - How can we fix this problem?

3) $\mathbb{E}_{z \sim q(z)} [\log p(z|x) - \log q(z)] + \text{cons.}$

$= \int_{-\infty}^{\infty} q(z) [\log p(z|x) - \log q(z)] \, dz + \text{cons.}$

for $x \leq 0$ $p(z|x)$ is not defined.

$\therefore$ ELBO does not exist

to fix this is to take $q(z)$ s.t it is defined in $[0, \infty)$

1. Which of the following conditions have to be satisfied by a distribution $q(\boldsymbol{z})$, such that it's possible to use it in variational inference (as described in our recipe on slides 80-81)
   a) We can compute the expected value of $\boldsymbol{z}$ in closed form
   b) We can compute the entropy of $q(\boldsymbol{z})$ in closed form
   c) We can draw samples from $q(\boldsymbol{z})$ with reparametrization
   d) We can compute the density $\log q(\boldsymbol{z})$ for an arbitrary $\boldsymbol{z}$
   e) We can compute $\log q(\boldsymbol{z})$ for a sample $\boldsymbol{z}$ drawn from $q(\boldsymbol{z})$
   f) The distribution can be factorized as $q(\boldsymbol{z}) = \prod_i q_i(\boldsymbol{z}_i)$

1) only c) and e) are necessary as we need to draw samples for $q(z)$ and we compute $q(z)$ & check the ELBO formulation.

2. Think of a probabilistic model with two latent variables $z_1, z_2 \in \mathbb{R}$ and an observed variable $x \in \mathbb{R}$ (i.e. write down $p_{\boldsymbol{\theta}}(x|z_1, z_2)$ and $p_{\boldsymbol{\theta}}(z_1, z_2)$), where the posterior can be factorized as $p_{\boldsymbol{\theta}}(z_1, z_2|x) = p_{\boldsymbol{\theta}}(z_1|x)p_{\boldsymbol{\theta}}(z_2|x)$.

3. Same as question 3, but now the posterior cannot be factorized.

Problem 1: Consider the following latent variable model.

$$p_\theta(z) = \text{Exp}(z|\theta) = \begin{cases} \theta \exp(-\theta z) & \text{if } z > 0, \\ 0 & \text{else.} \end{cases}$$

$$p(x|z) = \mathcal{N}(x|z, 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x-z)^2\right),$$

where $x \in \mathbb{R}$ is the observed data and $z \in \mathbb{R}_+$ is the latent variable. We have observed a single data point x and now would like to maximize the marginal log-likelihood $\log p_\theta(x) = \log\left(\int p(x|z)p_\theta(z)dz\right)$ w.r.t. the model parameters $\theta \in \mathbb{R}_+$. For this we will use variational inference.

We define the following parametric family of variational distributions

$$q_\phi(z) = \text{Exp}(z|\phi) = \begin{cases} \phi \exp(-\phi z) & \text{if } z > 0 \\ 0 & \text{else;} \end{cases}$$

that is parametrized by $\phi \in \mathbb{R}_+$. We are interested in solving the optimization problem

$$\max_{\theta>0, \phi>0} \mathcal{L}(\theta, \phi).$$

a) Assume that θ is known and fixed. Does there exist a value of ϕ such that the ELBO is tight, i.e. $\log p_\theta(x) = \mathcal{L}(\theta, \phi)$? Justify your answer.

b) Write down the ELBO $\mathcal{L}(\theta, \phi)$ for the above probabilistic model $p_\theta(x, z)$ and the variational distribution $q_\phi(z)$ and simplify it as far as you can. Your final answer should be a closed-form expression (no integrals or expectations).

c) Compute the gradients of the ELBO $\nabla_\theta \mathcal{L}(\theta, \phi)$ and $\nabla_\phi \mathcal{L}(\theta, \phi)$.

1) $\log p_\theta(x) = \mathcal{L}(\theta, \phi) + KL(q(z) \| p(z|x))$

as θ is fixed

should be zero to satisfy the condition

for this to happen $q(z) = p(z|x)$

$p(z|x) \propto p(x|z) \cdot p(z)$

$\propto \exp\left(-\frac{1}{2}(x-z)^2\right) \exp(-\theta z) \mathbb{1}(z>0)$

$\propto \exp\left(-\frac{1}{2}z^2 + (x-\theta)z\right) \mathbb{1}(z>0)$

$q_\phi(z) \propto \exp(-\phi z)\, \mathbb{1}(z > 0)$

$-\phi z = -\frac{1}{2} z^2 + (x-\theta) z$

quadratic term on RHS

(b) $f(\theta, \phi) = \mathbb{E}_{z \sim q_\phi(z)} [\log p(x|z) + \log p(z) - \log q(z)]$

$= \mathbb{E}_{z \sim q_\phi(z)} [-\frac{1}{2}\log(2\pi) - \frac{1}{2}(x-z)^2 + \log(\theta) - \theta z - \log(\phi) + \phi z]$

$= \mathbb{E}_{z \sim q_\phi(z)} [-\frac{z^2}{2} + xz + \log\theta - \theta z - \log\phi - \phi z] + \text{const.}$

$\mathbb{E}_{z \sim q_\phi(z)}[z] = \frac{1}{\phi}$

$= -\frac{1}{\phi^2} + \frac{x}{\phi} + \log(\theta) - \frac{\theta}{\phi} - \log(\phi)$

$+ \text{const}$

$= -\frac{1}{\phi^2} + \frac{x}{\phi} + \log(\theta) - \frac{\theta}{\phi} - \log(\phi) + \text{const.}$

1(c) $\frac{\partial}{\partial \theta} f(\theta, \phi) = -\frac{1}{\phi} + \frac{1}{\theta}$

$\frac{\partial}{\partial \phi} f(\theta, \phi) = \frac{2}{\phi^3} - \frac{x-\theta}{\phi^2} - \frac{1}{\phi}$

Problem 2: You want to draw samples from an exponential distribution with rate ϕ with reparametrization. Assume that

$$q_\phi(z) = \text{Exp}(z|\phi) = \begin{cases} \phi \exp(-\phi z) & \text{if } z > 0 \\ 0 & \text{else;} \end{cases}$$

where $\phi \in \mathbb{R}_+$.

a) You have access to an algorithm that produces samples ϵ from an exponential distribution with unit rate, that is

$$b(\epsilon) = \text{Exp}(\epsilon|1) = \begin{cases} \exp(-\epsilon) & \text{if } \epsilon > 0 \\ 0 & \text{else.} \end{cases}$$

Find a deterministic transformation $T_\phi \colon \mathbb{R}_+ \to \mathbb{R}_+$ that converts a sample $\epsilon \sim b(\epsilon)$ into a sample from $q_\phi(z)$.

b) Now, you have access to an algorithm that produces samples u from a uniform distribution on $[0, 1]$, that is

$$b(u) = \begin{cases} 1 & \text{if } u \in [0, 1] \\ 0 & \text{else.} \end{cases}$$

Find a deterministic transformation $S_\phi \colon [0, 1] \to \mathbb{R}_+$ that converts a sample $u \sim b(u)$ into a sample $z = S_\phi(u) \sim q_\phi(z)$.

a) $b(\varepsilon) = \text{Exp}(\varepsilon|1) = \begin{cases} \exp(-\varepsilon) & \varepsilon > 0 \\ 0 & \text{otherwise} \end{cases}$

T_ϕ tranforms $\varepsilon \sim \text{Exp}(1)$ into $z = T_\phi(\varepsilon) \sim \text{Exp}(\phi)$

taking into consideration the cumulative density function $F_\varepsilon(a) = Pr(\varepsilon \le a)$ and CDF of z $F_z(a) = Pr(z \le a)$

$$F_z(a) = Pr(z \le a) = Pr(T_\phi(\varepsilon) \le a)$$
$$Pr(\varepsilon \le T_\phi^{-1}(a))$$
$$= F_\varepsilon(T_\phi^{-1}(a))$$

$F_z(a) = 1 - \exp(-\phi a)$

$F_z(T_\phi^{-1}(a)) = 1 - \exp(-T_\phi^{-1}(a))$

$T_\phi^{-1}(a) = \phi a \Rightarrow$ $\boxed{T_\phi(x) = \frac{x}{\phi}}$

$\therefore$ the desired transformation is $\boxed{T_\phi(x) = \frac{x}{\phi}}$

(b) given $u \sim b(u)$ → uniform distribution in $[0,1]$

find $S_\phi : [0,1] \to \mathbb{R}_+$ s.t it converst sample $u \sim b(u)$ into $z = S_\phi(u) \sim q_\phi(z)$

CDF of $u \to F_u(a) = Pr(u \le a)$
CDF of $z \to F_z(a) = Pr(z \le a)$

$$\begin{aligned} Pr(z \le a) &= Pr(S_\phi(u) \le a) \\ &= Pr(u \le S_\phi^{-1}(a)) \\ &= F_u(S_\phi^{-1}(a)) \end{aligned}$$

$1 - \exp(-\phi a) = S_\phi^{-1}(a)$

$1 - S_\phi^{-1}(a) = \exp(-\phi a)$

$\ln(1 - S_\phi^{-1}(a)) = -\phi a$

$\boxed{S_\phi(a) = -\frac{1}{\phi} \ln(1-a)}$

Problem 3: You are given two distributions $q(\boldsymbol{z})$ and $p(\boldsymbol{z})$ over some random vector $\boldsymbol{z} \in \mathbb{R}^D$. Assume that both distributions can be factorized as

$$q(\boldsymbol{z}) = \prod_{i=1}^{D} q_i(z_i) \qquad\qquad p(\boldsymbol{z}) = \prod_{i=1}^{D} p_i(z_i).$$

(This is equivalent to saying that each component z_i is independent of z_j for $j \neq i$ under the distributions q and p). Your task is to prove that in this case the following equality holds

$$\mathbb{KL}(q(\boldsymbol{z})\|p(\boldsymbol{z})) = \sum_{i=1}^{D} \mathbb{KL}(q_i(z_i)\|p_i(z_i)).$$

3) $\mathbb{KL}(q(z)\|p(z)) = \int q(z) \ln \frac{q(z)}{p(z)} \, dz$

$= \int\int \cdots \int q_1(z_1) q_2(z_2) \cdots q_D(z_D) \log\left(\prod_{i=1}^{D} \frac{q_i(z_i)}{p(z_i)}\right) dz_1 dz_2 \cdots dz_D$

$= \sum_{i=1}^{D} \left(\int \cdots \int q_1(z_1) \cdots q_D(z_D) \log \frac{q_i(z_i)}{p(z_i)} \, dz_1 \cdots dz_D \right)$

$= \sum_{i=1}^{D} \int q(z_i) \ln \frac{q(z_i)}{p(z_i)} \left(\int \cdots \int q_1(z_1) \cdots q_{i-1}(z_{i-1}) q_{i+1}(z_{i+1}) \cdots q_D(z_D) \, dz_1 \cdots dz_{i-1} dz_{i+1} \cdots dz_D \right) dz_i$

$= 1$

$$= \boxed{\sum_{i=1}^{D} \int q(z_i) \ln \left(\frac{q(z_i)}{p(z_i)} \right) dz_i}$$

Chapter 3: Variational Autoencoders

In variational inference, our optimization problem is

$$\max_{\theta,\phi} \mathbb{E}_{z\sim q_\phi(z)}[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z)]$$

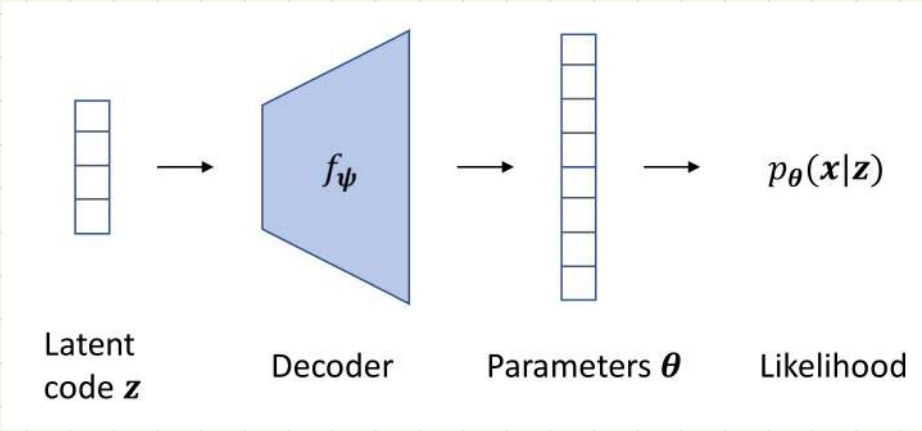
- **choosing $p_\theta(z)$**: pick simple $p(z) = \mathcal{N}(z|0,I)$
- **choosing $p_\theta(x|z)$**: idea: pick a parametric distribution $p_\theta(x|z)$ whose parameters are produced by some function f_ψ that takes z as input.

Choosing $p_\theta(x|z)$ involves trade-off between expressiveness and efficiency

$$p_\theta(x|y) = \mathcal{N}(x|\mu = f_\psi(z), I) = \prod_{j=1}^{D} \mathcal{N}(x_j|\mu = f_\psi(z)_j, 1)$$

The decoder f_ψ:

The neural network f_ψ is often called "the decoder" since it converts the latent variable z into the parameter θ of $p_\theta(x|z)$



• Choosing $q_\phi(z)$:

Dataset : N (iid samples)

variational distribution is over $z^{(i)}$'s : $q_\phi(z^{(1)}, \ldots, z^{(N)})$

making the mean field assumption

$$q_\phi(z^{(1)}, \ldots, z^{(N)}) = \prod_{i=1}^{N} q_{\phi^{(i)}}(z^{(i)})$$

choice of $q_{\phi^{(i)}}(z^{(i)})$ $\quad \phi^{(i)} : \{\mu^{(i)}, \Sigma^{(i)}\}$ ← diagonal

① multivariate normal : $q_{\phi^{(i)}}(z^{(i)}) = \mathcal{N}(z^{(i)} \mid \mu^{(i)}, \Sigma^{(i)})$

② Normalizing flow → forward parametrization

Why is Normal $q_\phi(z)$ a good choice?

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{z \sim q_\phi(z)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z)]$$

$$= \mathbb{E}_{z \sim q_\phi(z)}[\log p_\theta(x|z)] - KL(q_\phi(z) \| p(z))$$

KL divergence b/w two multivariate normal dist:

$$KL(q_\phi(z) \| p(z)) = \frac{1}{2}\left(\mathrm{tr}(\Sigma) + \mu^T\mu - \log|\Sigma| - J\right)$$

even simpler for diagonal covariance $\Sigma = \mathrm{diag}(\sigma^2)$

$$KL(q_\phi(z) \| p(z)) = \frac{1}{2}\left(\sum_{j=1}^{J}(\sigma_j^2 + \mu_j^2 - \log\sigma_j^2 - 1)\right)$$

for multiple parameters :

$$\mathcal{L}(\theta, \phi) = \frac{1}{N}\sum_{i}^{N}\mathcal{L}_i(\theta, \phi^{(i)}) = \frac{1}{N}\sum_{i}^{N}\left[\mathbb{E}_{z^{(i)} \sim q_{\phi^{(i)}}(z^{(i)})}\left[\log p_\theta(x^{(i)}|z^{(i)})\right]\right.$$

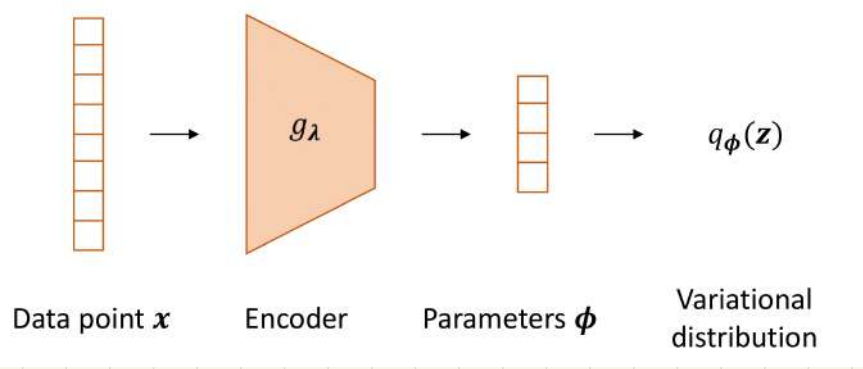
- But we have to learn separate parameters $\phi^{(i)}$ for each instance.

Ammortized Inference :

- we need to find $\phi^{(i)}_{optimal}$ for every sample $x^{(i)}$
- standard approach : $\phi^{(i)}_{optimal} = \arg\max_{\phi^{(i)}} f_i(\theta, \phi^{(i)})$
- Better idea : train a NN g_λ that tries to map every $x^{(i)}$ in the training set to its optimal parameters $\phi^{(i)}_{opt}$

$$\max_{\lambda} \frac{1}{N} \sum_{i}^{N} f_i(\theta, g_\lambda(x^{(i)}))$$

Encoder g_λ



"encoder" converts data point x into the parameter ϕ that defines the distribution $q_\phi(z)$ over the latent code z.

Putting it all together:

ELBO is a single sample

$$L(\psi, \lambda) = \mathbb{E}_{z \sim q_\phi(z)}[\log p_\theta(x|z)] - KL(q_\phi(z) \| p(z))$$

where $\theta = f_\psi(z)$ and $\phi = g_\lambda(x)$

Recipe for optimizing the ELBO:

1. Compute $\phi = g_\lambda(x)$
2. Compute an MC estimate of ELBO; often done using a single sample
   a) Draw $z' \sim q_\phi(z)$ with reparameterization
   b) Compute $\theta = f_\psi(z')$
   c) ELBO: $L(\psi, \lambda) \approx \log p_\theta(x|z') - KL(q_\phi(z) \| p(z))$
3. Backprop (compute $\nabla_\psi L(\psi, \lambda)$ and $\nabla_\lambda L(\psi, \lambda)$)
4. Update the NN weights ψ and λ using gradient ascent.

Reparametrization:

$$L(\psi, \lambda) = \mathbb{E}_{z \sim q_\phi(z)}[\log p_\theta(x|z)] - KL(q_\phi(z) \| p(z))$$

$\theta = f_\psi(z)$ $\phi = g_\lambda(x)$

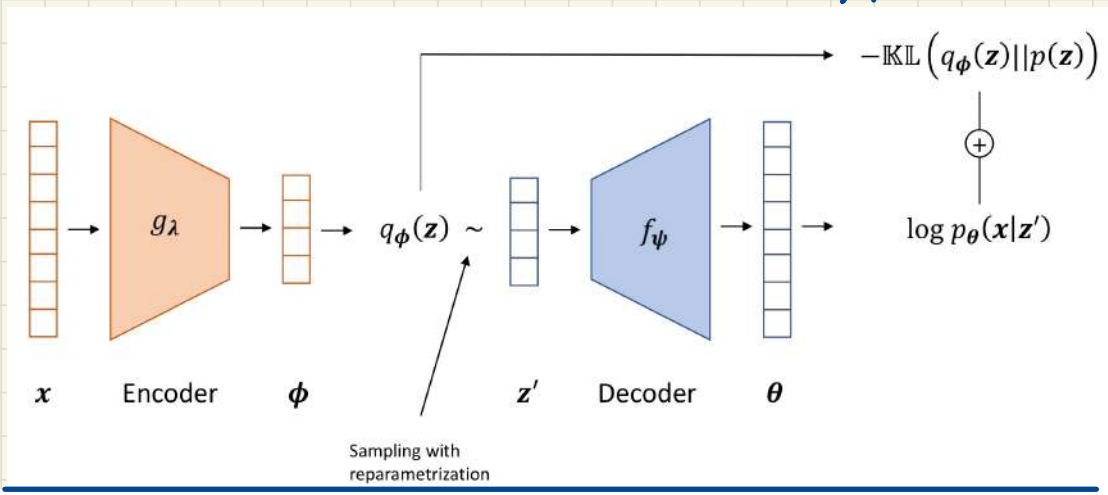
This means we need to sample from $q_\phi(z)$ with reparameter.

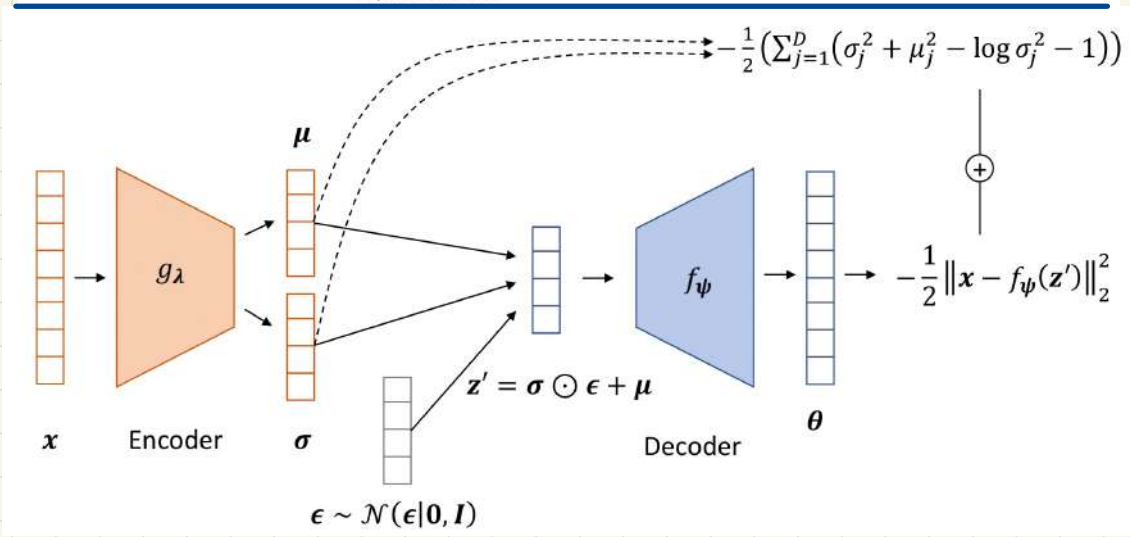
1. $\varepsilon \sim b(\varepsilon)$
2. $z' = T(\varepsilon, \phi)$

Variational Autoencoder (VAE)

$L(\psi, \lambda) = \mathbb{E}_{z \sim q_\phi(z)}[\log p_\theta(x|z)] - \mathbb{KL}(q_\phi(z) \| p(z))$

$\phi = g_\lambda(x)$ and $\theta = f_\psi(z)$





$f(\psi, \lambda) = \log p_\theta(x|z')$

$z' = g_\lambda(x)$ $\theta = f_\psi(z')$

03 - Generative Adversarial Networks :

idea : Transform given initial noise using flexible function

Resulting Distribution :

1) Draw an initial noise vector from the prior $z \sim p(z)$
2) Deterministically transform z into the target data / o/p space $x = f_\theta(z)$

$$p_\theta(x) = \frac{d}{dx_1} \cdots \frac{d}{dx_d} \int_{f_\theta(z) \le x} p(z)\, dz$$

intractable

- sampling is easy ; density evaluation is hard.
without likelihood, many inference and parameter learning tools become unavailable.

Forward Pointer : generative adversarial network :

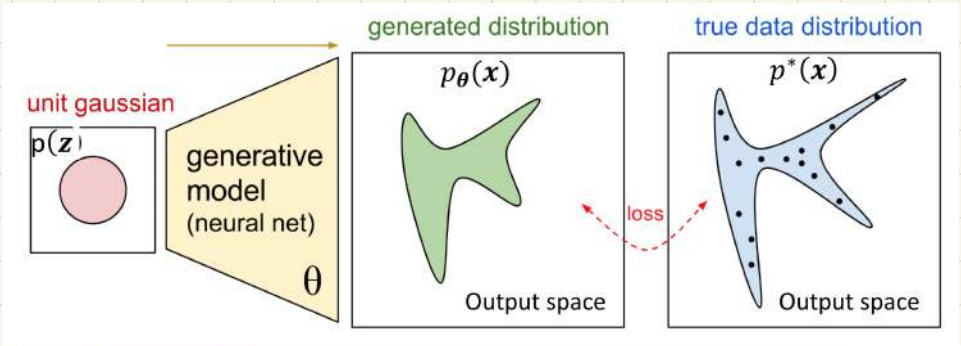
main idea:

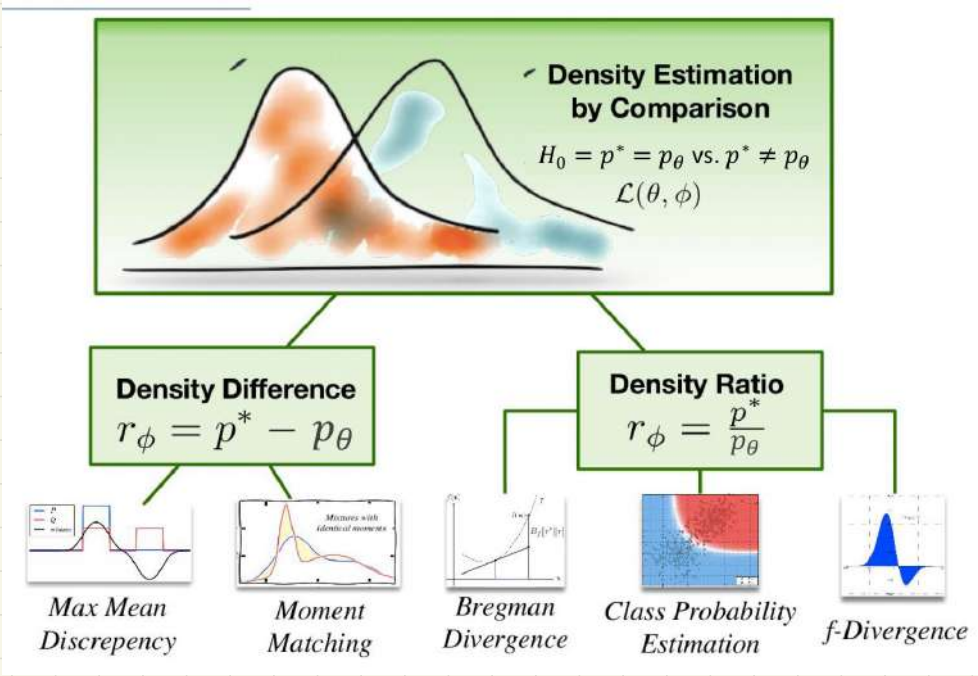
- two competing NN models
- generator : takes noise as i/p and generate "fake" samples
- discriminator : receives samples from both generator and training data and has to distinguish b/w the two "real" or "fake"
- goal : train the generator in such a way that the discriminator can not distinguish b/w "real" or "fake"

Likelihood free inference :

idea : any method that compares two set of samples this is called density estimation by comparison.

hypothesis → • true data distribution $p^*(x)$ and the model distribution $p_\theta(x)$ are equal.





Density difference : approach 1

1) Moment matching → p^* and p_θ are identical iff the expectations of any test statistic says are identical.

- thus, minimize the gap $\min_\theta \left(\mathbb{E}_{x \sim p^*}[s(x)] - \mathbb{E}_{z \sim p(z)}[s(f_\theta(z))] \right)^2$

approx. expectation using Monte Carlo

2) Max Mean Discrepancy:

- When $g(x)$ are defined within a reproducing kernel Hilbert Space, we obtain kernel based forms of these objectives
- These kernels are highly flexible and allow easy handling of data such as images, strings & graphs.

2) Ratio Base Approach

- Density ratio $r^*(x) = p^*(x)/p_\theta(x)$.
  cannot compute closed form → approximating means solving a learning problem
  approximate $r^*(x)$ by $r_\phi(x)$

general principle
1) optimize ratio loss: approx. true density ratio $r^*(x)$ (learning ϕ)
2) optimize generative loss: drive the density ratio towards 1 (learning θ)
can be solved alternatively.

Following three methods are instantiation of this principle

1) Class Probability estimation: • density ratio via classifier that distinguishes b/w observed and generated data. GAN
2) Divergence Maximization: • min f-divergence b/w true density and generation density
3) Ratio Matching: Directly minimize the error b/w the true density ratio and estimate of it
   $L = E_{p_\theta}[(r_\phi(x) - r^*(x))]^2$

Learning via Class Probability :

$p(x|Y=1) = p^*(x)$ and $p(x|Y=0) = p_\theta(x)$

(true) (generator)

$p(Y=1) = \pi$ $\quad p(Y=0) = 1-\pi$

$$r^*(x) = \frac{p^*(x)}{p_\theta(x)} = \frac{p(Y=1|x)}{p(Y=0|x)} \cdot \frac{1-\pi}{\pi}$$

density ratio estimation = class probability estimation

scoring function or discriminator $\quad D_\phi(x) = p(Y=1|x)$

$$f_{\theta,\phi} = \mathbb{E}_{(x,y)\sim p(x,y)}\left[-y \log[D_\phi(x)] - (1-y)\log(1-D_\phi(x))\right]$$

$$= \pi\, \mathbb{E}_{p^*(x)}\left[-\log D_\phi(x)\right] + (1-\pi)\,\mathbb{E}_{p(z)}\left[-\log[1-D_\phi(f_\theta(z))]\right]$$

1) solving : $\phi^*(\theta) = \arg\min_\phi f_{\theta,\phi}$

leads to the "best" discrimination for given generation model.

$$r^*(x) = \frac{p^*(x)}{p_\theta(x)} = \frac{p(Y=1|x)}{p(Y=0|x)} \approx \frac{D_{\phi^*(\theta)}(x)}{1-D_{\phi^*(\theta)}(x)}$$

$(\pi = .5)$

aim to drive $r^*(x) \to 1$

solving $\theta^* = \arg\max_\theta f_{\theta,\phi^*(\theta)}$

- generator and discriminator play a min-max game

$$\min_{\theta}\max_{\phi} \pi \mathbb{E}_{p^*(x)}[\log D_\phi(x)] + (1-\pi)\mathbb{E}_{p(z)}[\log[1 - D_\phi(f_\theta(z))]$$

- Discriminator → aims to distinguish b/w $p^*(x)$ and $p_\theta(x)$ (max)
  - min - cross entropy
- generator → aims to generate samples that are indistinguishable. (min)
  - max (min cross entropy)

Learning via Class Probability Estimation:

* Ratio Loss (discriminator loss) optimization:

$$\min_{\phi} \pi \mathbb{E}_{p^*(x)}[-\log D_\phi(x)] + (1-\pi)\mathbb{E}_{p(z)}[-\log[1 - D_\phi(f_\theta(z))]$$

* generative loss optimization:

$$\min_{\theta} \mathbb{E}_{p(z)}[\log[1 - D_\phi(f_\theta(z))]]$$

Questions

1. Assume that each data point is represented by a vector $x \in \{1, 2, \ldots, C\}^D$. What distribution would you pick for $p_\theta(x|z)$? How can we parametrize this distribution with a neural network?

1) Categorical distribution
   $D \times C$ matrix of logits A

$$p(x_i = c) = \frac{\exp(A_{ic})}{\sum_{j=1}^{C} \exp(A_{ij})}$$

2. Assume that each datapoint $\boldsymbol{x}$ is represented by a variable-length sequence of real numbers $\boldsymbol{x}_i \in \mathbb{R}^{D_i}$ (D_i might be different for different i's). What NN architecture could we use for the encoder g_λ in this case?

for variable lengths using RNN would be a good choice → we can use the hidden states as $q(z)$

3. Slide 94: Assume that we choose to model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with a normalizing flow. Should we use forward or reverse parametrization? Why?

we should choose reverse parametrization as we need $p_\theta(x|z)$ for single instances

4. Slide 97: Assume that we choose to model $q_\phi(\boldsymbol{z})$ with a normalizing flow. Should we use forward or reverse parametrization? Why?

we should use forward parametrization because we want to sample from $q(z)$

5. Slide 106: How can we ensure that the vector $\boldsymbol{\sigma}$ produced by the encoder g_λ is always positive?

we can use element-wise strictly positive function like softplus or exp.

1. What can we say about the ratio $r^*(\boldsymbol{x}) = p^*(\boldsymbol{x})/\, p_\theta(\boldsymbol{x})$ when:
   a) The generator and the discriminator are optimal
   b) The generator only is optimal ↓ depends on generator
   c) The discriminator only is optimal

↓ depends on generator

→ cannot make any remark

1) $r^*(x) = \frac{p^*(x)}{p_\theta(x)}$ density ratio only depends on the generator

2) For a given data $\boldsymbol{x}$, what is the probability for the discriminator to be correct when:

Ambi-guous

a) The generator and the discriminator are optimal $\approx 50\%$.

b) The generator only is optimal $\leq 50\%$.

c) The discriminator only is optimal $\geq 50\%$.

Problem 1: Below we show pseudocode for implementing 3 autoencoder-like neural net architectures. The observed data is denoted as $\boldsymbol{x} \in \mathbb{R}^D$. Here, $g_{\boldsymbol{\lambda}} : \mathbb{R}^D \to \mathbb{R}^L$ and $f_{\boldsymbol{\psi}} : \mathbb{R}^L \to \mathbb{R}^D$ are fully connected feedforward neural networks with learnable parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$. The output layers of $g_{\boldsymbol{\lambda}}$ and $f_{\boldsymbol{\psi}}$ have no (i.e. have linear) activation functions. $\mathcal{N}$ denotes the normal distribution, $\boldsymbol{I}_N$ is the $N \times N$ identity matrix, and $\mathbf{0}_N$ is the vector of all zeros of length N.

For each of the architectures below, explain whether it's **necessary** to use the reparametrization trick to compute the gradient of the loss $\mathcal{L}$ w.r.t. **both** $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$. Answer "Yes" or "No" and provide a justification. If the answer is "Yes", modify the code to implement the reparametrization trick.

a) Model 1

$$
\begin{aligned}
\boldsymbol{z}_i &\sim \mathcal{N}(\boldsymbol{x}_i, \boldsymbol{I}_D) \\
\boldsymbol{h}_i &= g_{\boldsymbol{\lambda}}(\boldsymbol{z}_i) \\
\tilde{\boldsymbol{x}}_i &= f_{\boldsymbol{\psi}}(\boldsymbol{h}_i) \\
\mathcal{L} &= \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2
\end{aligned}
$$

a) No, the parameters do not depend on learnable parameters

b) Model 2

$$
\begin{aligned}
\boldsymbol{h}_i &= g_{\boldsymbol{\lambda}}(\boldsymbol{x}_i) \\
\boldsymbol{z}_i &\sim \mathcal{N}(\boldsymbol{h}_i, \boldsymbol{I}_L) \\
\tilde{\boldsymbol{x}}_i &= f_{\boldsymbol{\psi}}(\boldsymbol{z}_i) \\
\mathcal{L} &= \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2
\end{aligned}
$$

b) Yes

$\varepsilon_i \sim \mathcal{N}(0_L, I_L)$

$z_i = h_i + \varepsilon_i$

c) Model 3

$$
\begin{aligned}
\boldsymbol{h}_i &= g_{\boldsymbol{\lambda}}(\boldsymbol{x}_i) \\
\boldsymbol{z}_i &\sim \mathcal{N}(\mathbf{0}_L, \boldsymbol{I}_L) \\
\tilde{\boldsymbol{x}}_i &= f_{\boldsymbol{\psi}}(\boldsymbol{h}_i + \boldsymbol{z}_i) \\
\mathcal{L} &= \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2
\end{aligned}
$$

→ no sampling required

→ no sampling req.

no.

Problem 2: Consider the same setup as in the previous problem. The model specified below is **not well defined**. Your task is to find the problem with the model and modify the pseudo code to fix it.

In addition, if you think it's **necessary** to use the reparametrization trick, include it in your implementation.

$$
\begin{aligned}
\boldsymbol{h}_i &= g_\lambda(\boldsymbol{x}_i) \\
\boldsymbol{z}_i &\sim \mathcal{N}(\mathbf{0}_L, \operatorname{diag}(\boldsymbol{h}_i)) \\
\tilde{\boldsymbol{x}}_i &= f_\psi(\boldsymbol{z}_i) \\
\mathcal{L} &= \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2^2
\end{aligned}
$$

covariance matrix should be +ve definite
∴ Entries should be +ve.

$h_i = \exp(g_\lambda(x_i))$

as z_i samples depends on λ of encoder we need to use the reparameterization technique when sampling z_i.

$\varepsilon_i \sim N(0_L, I_L)$
$z_i \sim \varepsilon_i \odot h_i$

Problem 3: The loss used in generative adversarial networks (GANs) can be written in the following form:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}[\log D_{\boldsymbol{\phi}}(\boldsymbol{x})] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(\boldsymbol{z})))]$$

where $p^*(\boldsymbol{x})$ is the true data distribution, $p(\boldsymbol{z})$ is the distribution of the noise, $f_{\boldsymbol{\theta}}$ is the generator, and $D_{\boldsymbol{\phi}}$ is the discriminator.

a) For a given generator (fixed parameters $\boldsymbol{\theta}$) assume there exists a discriminator $D_{\boldsymbol{\phi}^*}(\boldsymbol{x})$ with parameters $\boldsymbol{\phi}^*$ such that for all $\boldsymbol{x}$:

$$D_{\boldsymbol{\phi}^*}(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x}) + p_{\boldsymbol{\theta}}(\boldsymbol{x})}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the distribution learned by the generator. Show that $D_{\boldsymbol{\phi}^*}$ is **optimal**, i.e. $\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$.

Hint: $\arg\max_y [a \log(y) + b \log(1-y)] = \frac{a}{a+b}$ for any $a, b \in \mathbb{R}_0^+, a + b > 0$.

$$\max_{\phi} L(\theta,\phi) = \max_{\phi} \mathbb{E}_{p^*(x)}[\log D_\phi(x)] + \mathbb{E}_{p(z)}[\log(1 - D_\phi(f_\theta(z)))]$$

$$\max_{\phi} \mathbb{E}_{p^*(x)}[\log D_\phi(x)] + \mathbb{E}_{p_\theta(x)}[\log(1 - D_\phi(x))]$$

$$\max_{\phi} \int [p^*(x) \log D_\phi(x) + p_\theta(x) \log(1 - D_\phi(x))] dx$$

its not obvious to find D_{ϕ^*} from this integral but what if we found optimal discriminator for every single x? this will achieve higher value for our optimizer

$$\leq \int \max_{D_\phi} \left[p^*(x) \log D_\phi(x) + p_\theta(x) \log (1 - D_\phi(x)) \right] dx$$

$$y = D_\phi(x) \quad a = p^*(x) \quad b = p_\theta(x).$$

$$D_{\phi^*} = \arg\max_{D_\phi(x)} \left[p^* \log D_\phi(x) + p_\theta(x) \log(1 - D_\phi(x)) \right]$$

$$= \frac{p^*(x)}{p^*(x) + p_\theta(x)}$$

∴ the discriminator maximizes the value for expression, as this expression is greater than our previous expression so D_{ϕ^*} should also maximize that

b) What is value of the optimal $D_{\phi^*}(\boldsymbol{x})$ when:

- The generator is optimal i.e. $p_\theta(\boldsymbol{x}) = p^*(\boldsymbol{x})$
- The generator assigns a zero probability $p_\theta(\boldsymbol{x}) = 0$ to a sample $\boldsymbol{x}$ whereas $p^*(\boldsymbol{x}) \neq 0$
- The generator assigns a non-zero probability $p_\theta(\boldsymbol{x}) \neq 0$ to a sample $\boldsymbol{x}$ whereas $p^*(\boldsymbol{x}) = 0$

b) $D_{\phi^*}(x) = \frac{1}{2}$

$D_{\phi^*}(x) = 1$ all such samples as real.

$D_{\phi^*}(x) = 0$ all such samples fake.

Chapter 4: Robustness Part 1

Adversarial Examples - Definition

- Dataset : $(x_i, y_i) \sim \mathbb{P}_{data}$, $(x_i, y_i) \in \mathbb{R}^d \times y$
- Classifier : $f : \mathbb{R}^d \to y$

specify a perturbation set $\mathcal{P}(x)$, i.e set of perturbations which when applied to x do not change its semantic

we say $\tilde{x} \in \mathcal{P}(x)$ is an adversarial example for f at (x, y)
if
- $f(x) = y$ i.e f correctly classifies x
- $f(\tilde{x}) \neq y$ i.e fails to correctly classify $\tilde{x}$

- Perturbation should not change semantic content
- L_p norm : $\mathcal{P}_{\epsilon,p}(x) = \{\tilde{x} : \|\tilde{x} - x\|_p \leq \epsilon\}$

Adversarial Attacks : Objective Function :

$$\tilde{x}^*_x = \underset{\tilde{x} \in \mathcal{P}(x)}{\arg\max}\ l_{0/1}(f(\tilde{x}), y)$$

Recall : $l_{0/1}$ is the zero one loss
However : $l_{0/1}$ has either zero or undefined gradient

cross entropy loss L is often :

$$\tilde{x}^*_x = \underset{\tilde{x} \in \mathcal{P}(x)}{\arg\max}\ L(f(\tilde{x}), y)$$

Projected Gradient Descent:

$$\tilde{x}^*_x = \arg\max_{\tilde{x} \in P(x)} L(f(\tilde{x}), y)$$

$$x_{t+1} = \Pi(x_t + \eta_t \nabla_x L(f(x_t), y))$$

- train model: But update data instead of weights

Fast Gradient-Sign Method (FGSM):

$$\tilde{x} = \Pi(x + \eta \cdot \text{sign}(\nabla_x L(f(x), y)))$$

$P(x)$ is a ball with radius ε measured by the L_∞ norm, setting $\eta = \varepsilon$ yields

Alternative optimization problem:

- $\min_{\tilde{x}} D(x, \tilde{x})$ s.t $\text{loss}(f(\tilde{x}), y) > 0$
- D is large when $\tilde{x}$ is far from x

$$\min_{\tilde{x}} D(x, \tilde{x}) + \lambda L(\tilde{x}, y)$$ (converting to unconstrained optimization)

an effective loss fn:

$$L(\tilde{x}, y) = \left[Z(\tilde{x})_y - \max_{i \neq y} (Z(\tilde{x})_i) \right]_+$$

- $y \sim$ original class
- $[x]_+ = \max(x, 0)$
- $Z(\tilde{x})_i = \log f(\tilde{x})_i$ (log prob. of class i)
- loss L is positive if $\tilde{x}$ is classified as y and 0 otherwise

- Fixing a "bad" model does not seem to be a feasible approach when trying to increase the robustness

Robust Training: aim to produce models that are robust to adversarial perturbations

Common theme: optimize "worst case" loss

- Let $l(\hat{y}, y)$ be some loss (for ex. cross entropy)
- The (non-robust) training tries to find an f that minimizes the expected loss

$$R = \mathbb{E}_{(x,y)\in P_{data}} [\, l(f(x), y)]$$

- The robust version of the problem.

$$R = \mathbb{E}_{(x,y)\in P_{data}} \Big[\sup_{\tilde{x}\in\mathcal{P}(x)} l(f(\tilde{x}), y) \Big]$$

* = loss achieved by worst-case perturbation in $\mathcal{P}(x)$

Adversarial Training:

Idea: perform stochastic gradient descent (SGD) on the robust loss R_{rob}.

$$R_{rob} = \mathbb{E}_{(x,y)\in P_{data}} \Big[\sup_{\tilde{x}\in\mathcal{P}(x)} l(f(\tilde{x}), y) \Big]$$

$f = f_\theta$ (NN) parameterized by weights θ

$$\nabla_\theta R_{rob} = \mathbb{E}_{(x,y)\in P_{data}} \Big[\nabla_\theta \Big(\sup_{\tilde{x}\in\mathcal{P}(x)} l(f(\tilde{x}), y) \Big) \Big]$$

use Daskin's Theorem for calculating the gradient of worst case loss.

Daskin's Theorem: $\Delta(\theta)$ → set of $\tilde{x}$ for which supremum is obtained.

if $\Delta(\theta)$ contains a single element, i.e $\Delta(\theta) = \{\tilde{x}^*_\theta\}$, then sup of differential exists

$$\nabla_\theta \left(\sup_{\tilde{x} \in P(x)} l(f_\theta(\vec{x}), y) \right) = \nabla_\theta l(f_\theta(\tilde{x}^*_\theta), y)$$

Problem: 1) finding worst case perturbed example $\tilde{x}$ is intractable.

idea: create adversarial examples as proxy of the worst case examples.

- Adversarial training algorithm:

1) $(x_i, y_i) \sim \hat{P}_{data}$
2) using adversarial attack procedure, find an $\tilde{x}_i$ with high loss $l(f_\theta(\tilde{x}_i), y_i)$
3) update the weights via gradient descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta l(f_\theta(\tilde{x}_i), y_i)$$

04 - Exact certification:

goal: Develop an algorithm that answers the question "Is the classifier f_θ around the sample x adversarial-free (within an ϵ-ball measured by some norm".

- NN → sequence of functions
- each layer: $f_i(x) = \sigma(W_i x + b_i)$
- ReLu activation: $\sigma(x) = \max(0, x)$
- overall function: $F : \mathbb{R}^d \to \mathbb{R}^{|y|}$

$$F(x) = W_L f_{L-1} \circ f_{L-2} \circ \cdots \circ f_1(x) + b_L$$

Theorem: Exact certification of NN with ReLU activation function and L_∞-bounded perturbation is NP Hard.

Mixed integer Linear programming (MILP):

$$\min: \quad c^T x$$
$$\text{subject to}: \quad Ax \leq b \; ; \; x \geq 0$$

} can be solved efficiently

MILP: some variables constrained to integers and some not.

NP-complete

Exact certification as MILP:

- suppose our classifier predicts class $c^* = \text{argmax}\, F(x)_c$
- $m_t = F(x)_{c^*} - F(x)_t$: classification margin b/w c^* and t
- worst case margin:

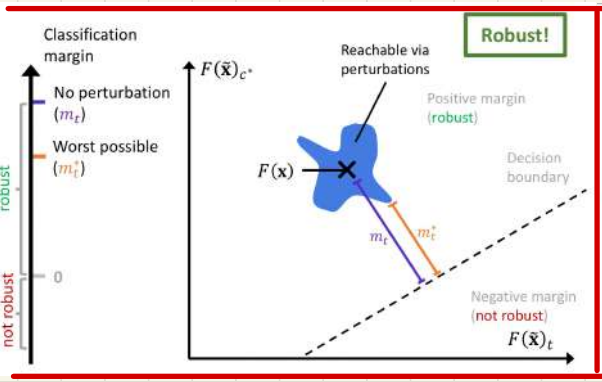
$$m_t^* = \min_{\tilde{x}} F(\tilde{x})_{c^*} - F(\tilde{x})_t$$

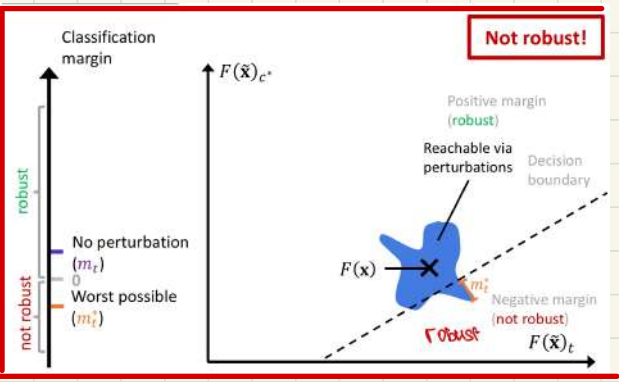
$$\text{subject to} \quad \|\tilde{x} - x\|_p \leq \epsilon$$

$m^*_t > 0$ → prediction cannot be changed from c^* to t

- if $\forall$ classes $t \neq c^*$, $m^*_t > 0$ → certify robustness
- if for any class $t \neq c^*$, $m^*_t < 0$ → $\exists$ adversarial example $\hat{x} \in \mathcal{P}_{\epsilon,p}(x)$

(1) (2)





optimization Problem :

$$m^*_t = \min_{\tilde{x}} [\hat{x}^{(L)}]_{c^*} - [\hat{x}^{(L)}]_t$$

subject to $\|\tilde{x} - x\|_p \leq \epsilon$

$y^{(0)} = \tilde{x}$

$\hat{x}^{(i)} = W_i y^{(i-1)} + b_i, \forall i = 1 \ldots L$

$y^{(i)} = \text{ReLU}(\hat{x}^{(i)}) \quad \forall i = 1 \ldots L-1$

we need to convert to MILP encode:

1) L_p constraints → adversarial perturbations
2) the nonlinear ReLU constraints

L_p constraints :

$x_i - \tilde{x}_i \leq \epsilon \quad \forall i$

$\tilde{x}_i - x \leq \epsilon \quad \forall i$

L_1 : straightforward

L_2 : mixed integer quadratic program.

ReLU :

↳ naive way makes problem nonlinear

↳ $y_i = \text{ReLU}(x_i) \Leftrightarrow$ ($y_i \leq a_i x_i$ and $y_i \geq 0$ & $y_i \geq x_i$)

$(y_i \leq x_i - l_i(1 - a_i)) \wedge (y_i \leq a_i u_i) \quad \forall i$

$\wedge (y_i \geq x_i) \quad \wedge (y_i \geq 0) \quad \wedge (a_i \in \{0,1\})$

overall MILP

$$m^*_t = \min_{\tilde{x}, y^{(l)}, \hat{x}^{(l)}, a^{(l)}} [\hat{x}^{(L)}]_{c^t} - [\hat{x}^{(L)}]_t$$

subject to:

$$\tilde{x}_i - \hat{x}_i \le \varepsilon \quad \forall i$$
$$\hat{x}_i - \tilde{x}_i \le \varepsilon \quad \forall i$$

$$y^{(0)} = \tilde{x}$$
$$\hat{x}^{(l)} = W_l y^{(l-1)} + b_l \quad \forall l = 1 \dots L$$

$$y_i^{(l)} \le \hat{x}_i^{(l)} - l_i^{(l)} (1 - a_i^{(l)})$$
$$y_i^{(l)} \ge \hat{x}_i^{(l)}$$
$$y_i^{(l)} \le u_i^{(l)} \cdot a_i^{(l)}$$
$$y_i^{(l)} \ge 0$$
$$a_i^{(l)} \in \{0, 1\}$$

on the lower and upper Bounds : interval arithmetic

$$u^{(l)} = [W_l]_+ u^{(l-1)} - [W_l]_- l^{(l-1)} + b_l$$
$$l^{(l)} = [W_l]_+ l^{(l-1)} - [W_l]_- u^{(l-1)} + b_l.$$

$$[w]_+ = \max\{w, 0\}$$
$$[w]_- = \max\{-w, 0\}$$

l_0 constraints

$$u_i^{(0)} = x_i + \varepsilon$$
$$l_i^{(0)} = x_i - \varepsilon$$

stable and unstable units :

$$u_i^{(l)} \ge l_i^{(l)} \ge 0$$ — stably active
$$0 \ge u_i^{(l)} \ge l_i^{(l)}$$ — stably inactive
$$u_i^{(l)} \ge 0 \ge l_i^{(l)}$$ — unstable

} can be removed from optim

\# tighter bounds has no influence on correctness
- they just lead to more stable units → faster optimization.

1. Given an arbitrary binary classifier f for an input domain $\mathbb{R}^d$ and the perturbation set $\mathcal{P}_{\epsilon,p}(\mathbf{x})$ as defined before. Is it possible that every $\mathbf{x} \in \mathbb{R}^d$ is "robust", i.e. no adversarial example exists?

1) is it possible that every $x \in \mathbb{R}^d$ is "robust"

$\mathcal{P}_{\epsilon,p}(x) = \{\tilde{x} : \|\tilde{x} - x\|_p < \epsilon\}$

yes a constant classifi would not be affected by perturbations

2. Will the fast gradient-sign method (FGSM) always find an adversarial example (assuming there exist some in the set of perturbations $\mathcal{P}_{\epsilon,\infty}(\mathbf{x})$)?

no, it is not gaurantteed as FGSM only takes the sign of gradient can overshoot.

Problem 1: Suppose we have a trained binary logistic regression classifier with weight vector $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$. Given a sample $\boldsymbol{x} \in \mathbb{R}^d$ we want to construct an adversarial example via gradient descent on the binary cross entropy loss:

$$\mathcal{L}(\boldsymbol{x}, y) = -y \log(\sigma(z)) - (1-y)\log(1-\sigma(z)),$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid function, $z = \boldsymbol{w}^T\boldsymbol{x} + b$, and $y \in \{0, 1\}$ is the class label of the sample at hand.

a) Derive the gradient $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, y)$. How do you interpret the result?

Hint: You may use the relation $1 - \sigma(z) = \sigma(-z)$.

b) Provide a closed-form expression for the worst-case perturbed instance $\tilde{\boldsymbol{x}}^*$ (measured by the loss $\mathcal{L}$) for the perturbation set $\mathcal{P}(x) = \{\tilde{\boldsymbol{x}} : \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2 \leq \epsilon\}$, i.e.

$$\tilde{\boldsymbol{x}}^* = \underset{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2 \leq \epsilon}{\arg\max}\ \mathcal{L}(\tilde{\boldsymbol{x}}, y)$$

c) What is the smallest value of ϵ for which the sample $\boldsymbol{x}$ is misclassified (assuming it was correctly classified before)?

d) We would now like to perform adversarial training. Provide a closed-form expression of the worst-case loss

$$\hat{\mathcal{L}}(\boldsymbol{x}, y) = \max_{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2 \leq \epsilon} \mathcal{L}(\tilde{\boldsymbol{x}}, y)$$

as a function of $\boldsymbol{x}$ and $\boldsymbol{w}$. How do you interpret the results?

a) $z = w^T x + b$

$\nabla_x h(x,y)$: $\frac{-y}{\sigma(z)} \frac{\partial(\sigma(z))}{\partial z} \cdot \nabla_x z - \frac{1-y}{\sigma(-z)} \frac{\partial(\sigma(-z))}{\partial z} \nabla_x z$

$= \frac{-y}{\sigma(z)} \sigma(z)\sigma(-z)\, w + \frac{1-y}{\sigma(-z)} \sigma(-z)\sigma(z) \cdot w$

$= -y\,\sigma(-z)\, w + (1-y)\,\sigma(z)\, w$

gradient is orthogonal and points in wrong direction

b) $\tilde{x}^* = x - \varepsilon \frac{w}{\|w\|_2}$ if $y = 1$

$\tilde{x}^* = x + \varepsilon \frac{w}{\|w\|_2}$ if $y = 0$

c) for classification to change classification

$$\sigma(z) = 0.5 \Rightarrow w^T \tilde{x} + b = 0$$

$$w^T x - w^T \varepsilon \frac{w}{\|w\|_2} + b = 0$$

$$w^T x - \varepsilon \|w\|_2 + b = 0$$

$$\frac{1}{\|w\|_2}(w^T x + b) = \varepsilon$$

for misclassification $\varepsilon > \frac{1}{\|w\|_2}(w^T x + b)$

analogous for $y = 0$ $\quad \varepsilon > \frac{1}{\|w\|_2}(-w^T x - b)$

d) $$\tilde{f}(x, y) = \max_{\|\tilde{x} - x\|_2 \leq \varepsilon} f(\tilde{x}, y)$$

$$= f(\tilde{x}^*, y)$$

$$= -y \log\left(\sigma\left(w^T x - \varepsilon \frac{w^T w}{\|w\|_2} + b\right)\right) + (1-y) \log\left(\sigma\left(-w^T x - \varepsilon \frac{w^T w}{\|w\|_2} - b\right)\right)$$

$$= -y \log\left(\sigma\left(w^T x - \|w\|_2 \varepsilon + b\right)\right) + (1-y) \log\left(\sigma\left(-w^T x + \|w\|_2 \varepsilon - b\right)\right)$$

Problem 2: In the lecture on exact certification of neural network robustness we have considered $K-1$ optimization problems (one for each incorrect class) of the form (c.f. slide 42):

$$m_t^* = \min_{\hat{\boldsymbol{x}}, \boldsymbol{y}^{(l)}, \hat{\boldsymbol{x}}^{(l)}, \boldsymbol{a}^{(l)}} [\hat{\boldsymbol{x}}^{(L)}]_{c^*} - [\hat{\boldsymbol{x}}^{(L)}]_t \quad \text{subject to MILP constraints.}$$

That is, for each class $t \neq c^*$, we optimize for the **worst-case margin** m_t^*, and conclude that the classifier is robust if and only if

$$\min_{t \neq c^*} m_t^* \geq 0.$$

However, we can equivalently solve the following single optimization problem:

$$m^* = \min_{\hat{\boldsymbol{x}}, \boldsymbol{y}^{(l)}, \hat{\boldsymbol{x}}^{(l)}, \boldsymbol{a}^{(l)}} \left([\hat{\boldsymbol{x}}^{(L)}]_{c^*} - y \right) \quad \text{subject to } y = \max_{t \neq c^*} [\hat{\boldsymbol{x}}^{(L)}]_t \wedge \text{MILP constraints},$$

where we have introduced a new variable y into the objective function.

Express the equality constraint

$$y = \max(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{K-1})$$

using only linear and integer constraints. To simplify notation, here $\boldsymbol{x}_k \in \mathbb{R}$ denotes the logit corresponding to the k-th incorrect class, and $\boldsymbol{l}_k$ and $\boldsymbol{u}_k$ its corresponding lower and upper bound.

Hint: You might want to introduce binary variables to indicate which logit is the maximum.

$u_{max} = \max_k u_k$ i.e largest upper bound

$y \leq x_k + (1 - b_k)(u_{max} - l_k)$ $\quad \forall\ 1 \leq k \leq K-1$

$y \geq x_i$ $\quad \forall\ 1 \leq k \leq K-1$

$b_k \in \{0, 1\}$ $\quad \forall\ 1 \leq k \leq K-1$

$\sum_{k=1}^{K-1} b_k = 1$ → one element = 1 all others are zero.

valid assignment of b is to have $b_k = 1$ for (unique) maximum value $x_k = \max(x_1, \ldots x_{k-1})$

to see this, consider the case $b_k = 1$, but x_k is not max., then $y \leq x_k$, however from ② we would get $x_{max} \geq x_i \ \forall\ 1 \leq k \leq K-1$. contradiction

consider the case $b_k = 1$ and corresponding value x_k is indeed the (unique) maximum. from ① & ② $y = x_k$.

03 - Convex Relaxation :

Relaxed Classification Margin

Instead of solving:

$$m^*_t = \min_{\tilde{x}} F(\tilde{x})_{c^*} - F(\tilde{x})_t$$

$$\text{subject to } \|\tilde{x} - x\|_p \leq \epsilon$$

$$y^{(0)} = \tilde{x}$$

$$z^{(l)} = W_l y^{(l-1)} + b_l \quad \forall l = 1, \dots L$$

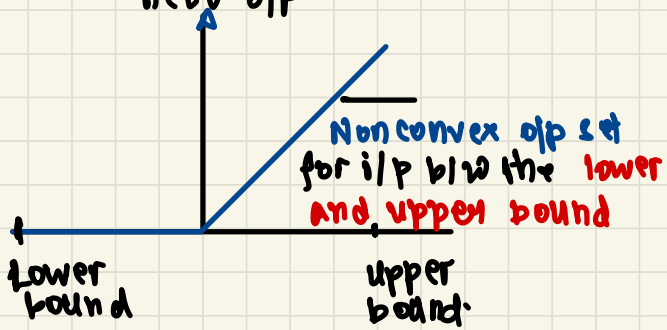
$$y^{(l)} = \text{ReLU}(z^{(l)}) \quad \forall l = 1 \dots L-1$$

We solve a relaxed problem

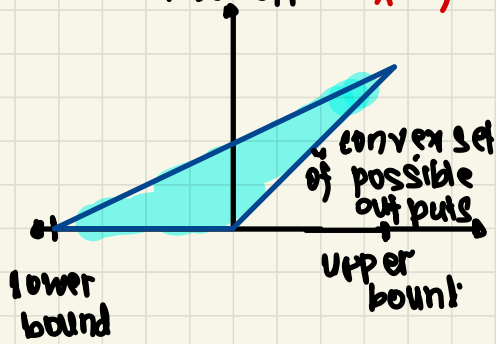
- Results in lower bound $m^{*(b)}_t$ on the true minimum m^*_t
  - if $m^{*(b)}_t > 0$ → classifier prediction cannot be changed
  - can make optimization possible in polynomial time
  - price → cannot make YES or NO statement in some cases.

original constraint: $y^{(l)} = \text{ReLU}(\hat{x}^{(l)})$

ReLU: Nonconvex o/p set



Relaxed constr. $y^{(l)} \in$ Triangle $(\hat{x}^{(l)})$



Note: O/p of the ReLU activation is no longer deterministic but a variable to optimize over.

Convex ReLU relaxation: Formal definition

$y^{(k)} = \text{ReLU}(\hat{x}^{(k)})$

1) For unstable units ($l_i^{(k)} < 0 \wedge u_i^{(k)} > 0$)
   - $y_i^{(k)} \geq 0$
   - $y_i^{(k)} \geq \hat{x}_i^{(k)}$
   - $(u_i^{(k)} - l_i^{(k)})\, y_i^{(k)} - u_i^{(k)} \hat{x}_i^{(k)} \leq -u_i^{(k)} l_i^{(k)}$

Note: The relaxation is equivalent to relaxing the integer constraint $a_i \in \{0,1\}$ to $a_i \in [0,1]$

2) For stable active units ($l_i^{(k)} \geq 0$):
   - $y_i^{(k)} = \hat{x}_i^{(k)}$

3) For stably inactive units ($u_i^{(k)} \leq 0$):
   - $y_i^{(k)} = 0$

Note everything is linear here.

overall LP:

$$m_t^* = \min_{\tilde{x}, y^{(k)}, \hat{x}^{(k)}} [\hat{x}^{(L)}]_{c^*} - [\hat{x}^{(L)}]_t$$

subject to

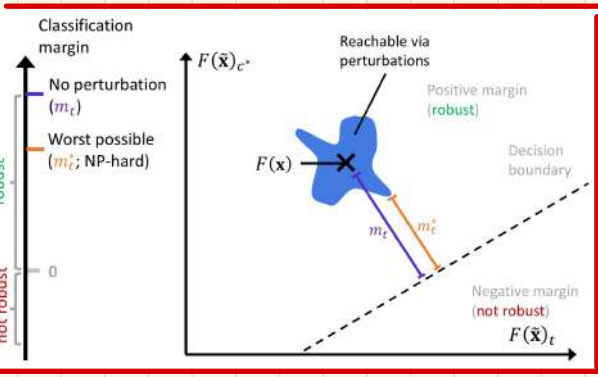
$$x_i - \tilde{x}_i \leq \epsilon$$
$$\tilde{x}_i - x_i \leq \epsilon$$

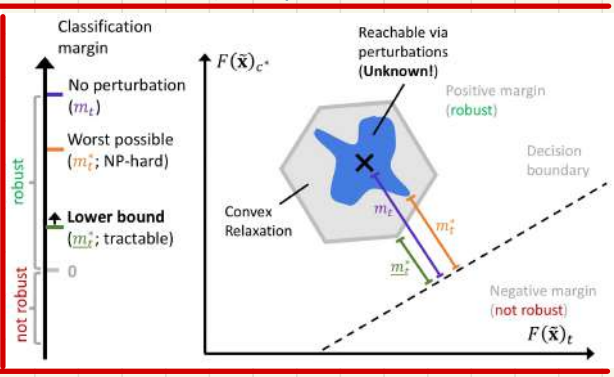
$$\hat{x}^{(0)} = \tilde{x}$$
$$\hat{x}^{(k)} = W_k y^{(k-1)} + b_k \quad \forall k = 1, \dots L$$

$$\left.\begin{aligned} & y_i^{(k)} \geq \hat{x}_i^{(k)} \\ & (u_i^{(k)} - l_i^{(k)}) y_i^{(k)} - u_i^{(k)} \hat{x}_i^{(k)} \leq -u_i^{(k)} l_i^{(k)} \end{aligned}\right\} \begin{aligned} & \forall k = 1 \dots L-1, \\ & \forall i : l_i^{(k)} < 0 \wedge u_i^{(k)} > 0 \end{aligned}$$

$$\begin{aligned} & y_i^{(k)} = \hat{x}_i^{(k)} && \forall k = 1 \dots L, \forall i : l_i^{(k)} \geq 0 \\ & y_i^{(k)} = 0 && \forall k = 1 \dots L, \forall i : u_i^{(k)} \leq 0 \end{aligned}$$

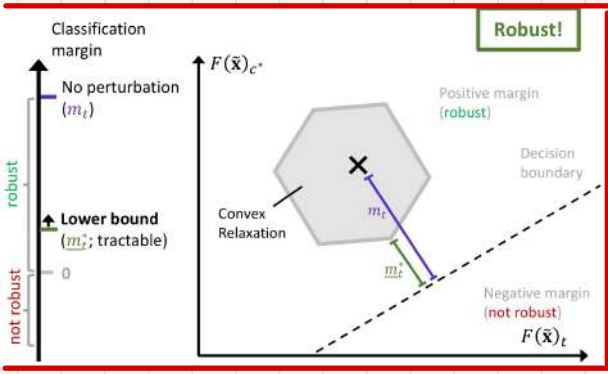
not robust

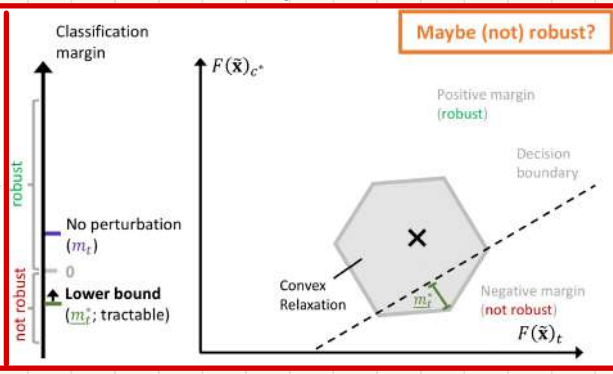




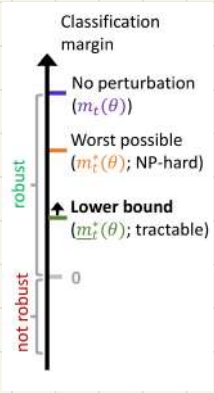
Robust

maybe case





Note: tightness of the lower and upper bound influence the quality of relaxation, i.e. how often we return MAYBE.



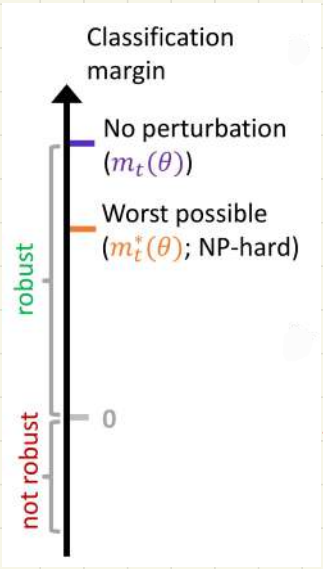
$\theta \rightarrow$ weights of NN.

the optim problem's solution depends on this

During robust training we aim to find a good θ (via gradient descent)

$m_t^*(\theta)$: dependent on θ.

- challenge is to calculate ∇_θ([illegible])



ReLU relaxation via an LP helps in efficiently optimize a robust loss based on the certification

keeping discussion simple → correctly classified (loss = 0)
↳ misclassified : loss = margin to decision boundary

$$\ell(f_\theta(\tilde{x}), y) = \max_{t \neq y} \max\left(F_\theta(\tilde{x})_t - F_\theta(\tilde{x})_y, 0\right)$$

thus, the supremum evaluates to:

$$\sup_{\tilde{x} \in \mathcal{P}(x)} \ell(f_\theta(\tilde{x}), y) = \max_{t \neq y} \max(-m_t^*(\theta), 0)$$

To make it tractable, we instead optimize via lower bound (→ more pessimistic) loss

$$\sup_{\tilde{x} \in \mathcal{P}(x)} \ell(f_\theta(\tilde{x}), y) = \max_{t \neq y} \max(-m_t^*(\theta), 0)$$

$$\leq \max_{t \neq y} \max(-\underline{m_t^*}(\theta), 0)$$ ← lower bound

challenge : difficult to calculate $\nabla_\theta \underline{m_t^*}(\theta)$

idea : can we solve another lower bound which does not require us to solve an optimization problem.

Recap: Strong Duality

primal

$\min_x h_0(x)$

s.t $h_i(x) \leq 0 \quad i = 1 \dots m$

in our case, the primal is the LP form slide, overall LP

dual

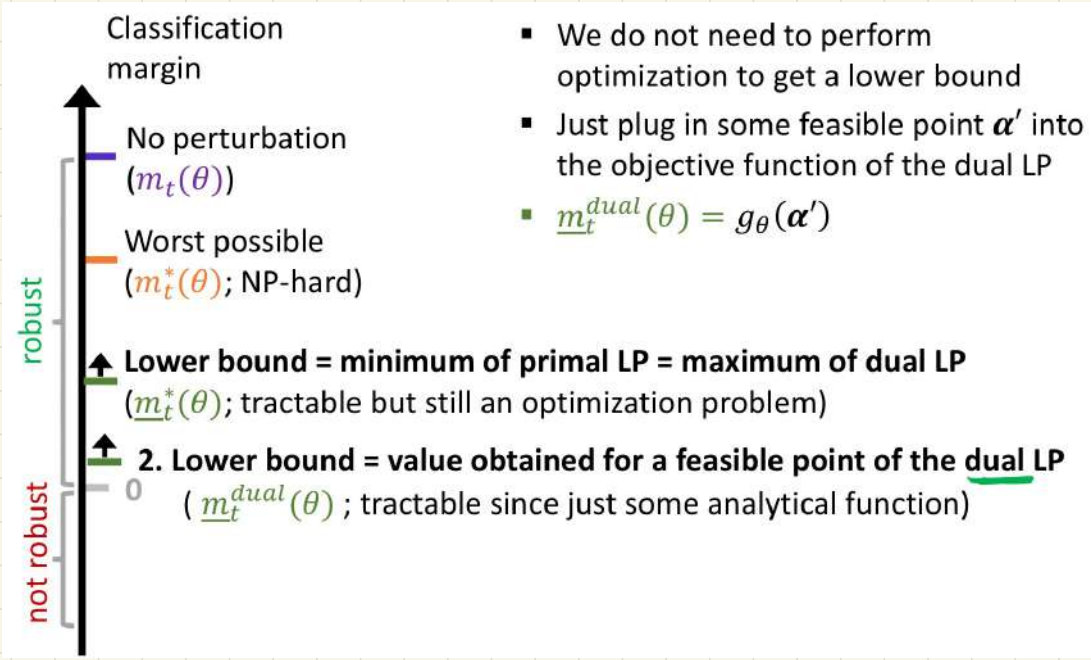
$\max_\alpha g(\alpha)$

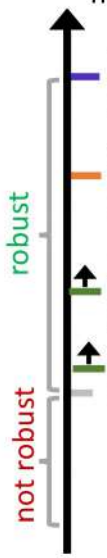
s.t $\alpha_i \geq 0 \quad i = 1 \dots M$

in our case, both primal and dual depend additionally on θ. Thus it would be more accurate to write $g_\theta(\alpha)$

$$h_0(x') \geq h_0(x^*) = g(\alpha^*) \geq g(\alpha')$$

any feasible x' — optimal x^* — optimal α^* — any feasible α'





$$\sup_{\tilde{\mathbf{x}} \in \mathcal{P}(\mathbf{x})} \ell\left(f_\theta(\tilde{\mathbf{x}}), y\right) = \max_{t \neq y} \max(-m_t^*(\theta), 0)$$

$$\leq \max_{t \neq y} \max(-\underline{m}_t^*(\theta), 0)$$

$$\leq \max_{t \neq y} \max(-\underline{m}_t^{dual}(\theta), 0)$$

$$= \max_{t \neq y} \max(-g_\theta(\boldsymbol{\alpha}'), 0)$$

We reached our goal: $\nabla_\theta g_\theta(\boldsymbol{\alpha}')$ can be easily computed!

08 - Lipschitz Continuity for Robustness Certification

a function $f: \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous if

$$D_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K \cdot D_{\mathcal{X}}(x_1, x_2) \quad \forall x_1, x_2 \in \mathcal{X}$$

- Idea is to bound how much the output of the network can change for a given perturbation of the input.

- Given a function $f(x) = (\phi_L \circ \phi_{L-1} \cdots \circ \phi_1)(x)$, the Lipschitz constant of f is bounded from above by:

$$L(f) \leq \prod_{i=1}^{L} L(\phi_i)$$

Recall: $F(x) = W_2 f_{L-1} \circ f_{L-2} \cdots \circ f_1(x) + b_L$

* we can compute the Lipschitz constant of individual layers to upper bound the Lipschitz constant of neural network F.

$$f(x) = Wx + b$$

$$\|(Wx_1 + b) - (Wx_2 + b)\|_p \leq K \|x_1 - x_2\|_p$$

let $a = x_1 - x_2$; and rearranging

$$L(f) = \sup_{a \neq 0} \frac{\|Wa\|_p}{\|a\|_p} \quad \{\text{operator norm}\}$$

- $p = 1$, $L(f) \to$ max L1 norm of columns of W
- $p = \infty$, $L(f) \to$ max L1 norm of rows of W
- $p = 2$, $L(f) \to$ max singular value of W.

Convolution layers:

- computing Lipschitz constant of conv layers is bit trickier in case of $p = 2$, but approx efficiently.

Activation function

- Typical activation functions (ReLU, softmax, ..) have a Lipschitz constant of at worst 1.

ReLU → has Lipschitz constant of exactly 1

For feed forward NN F with ReLU activation upper bound is Lipschitz constant

$$L(F) = \prod_{i=1}^{L} \|W_i\|_p$$

By regularizing norm of weight matrices : we can enforce any Lipschitz constant.

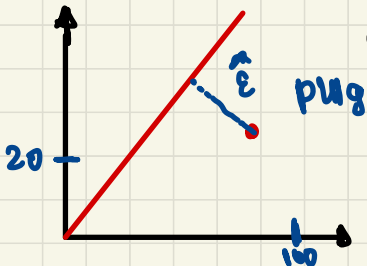
certifying robustness via Lipschitz continuity :

- Neural net F → with Lipschitz constant K.
- given an i/p sample x and corresponding logits $\hat{y} = F(x)$ we can compute the maximum norm ε of a perturbation δ.
  $\delta : \|\delta\|_p \leq \varepsilon$ for which we can gaurantee robustness of classifier

Approach:

1. compute the min. perturbation norm $\hat{\varepsilon} = \|\hat{\delta}\|_p$ in logit space to change classification
2. Compute the corresponding perturbation norm $\varepsilon = \|\delta\|_p$ based on the Lipschitz constant

1)



2) $\|F(x) - F(\tilde{x})\|_p \leq K \|x - \tilde{x}\|$

plugging in the max change we can certify $\hat{\varepsilon}$

$$\varepsilon = \|x - \tilde{x}\|_p \leq \hat{\varepsilon} / K$$

$$\Rightarrow K\|x - \tilde{x}\|_p \leq \hat{\varepsilon}$$

$$\Rightarrow \|F(x) - F(\tilde{x})\|_p \leq \hat{\varepsilon}$$

- we can certify that for any $\tilde{x} \in p(x) = \{\tilde{x} : \|x - \tilde{x}\|_p \leq \frac{\hat{\varepsilon}}{K}\}$ the predicted class for sample x does not change
- if K is small we get more robustness certificates

Partial Considerations

- we want our classifier to have small Lipschitz constant
- However small K limits the expressiveness of F
- There is a tradeoff b/w expressiveness and robustness

05 - Randomized Smoothing

idea: transform base classifier into a smoothed classifier by randomly adding noise to the input and predicting majority class given many samples.

general idea: for any classifier $f(x) = \arg\max_{c \in Y} F(x)_c$ into a smooth classifier g

$g(x)$: most probable prediction of f under Gaussian noise

Formal definition:

$(g(x))_c$: Probability that f classifies a sample from $N(x, \sigma^2 I)$ (or equivalently $x+\varepsilon$ where $\varepsilon \sim N(0, \sigma^2 I)$) as class c

$$\mathbb{P}_\varepsilon(f(x+\varepsilon) = c) = \mathbb{E}_\varepsilon(\mathbb{I}[f(x+\varepsilon) = c]) = \int \mathbb{P}(x=z)\,\mathbb{I}[f(z)=c]\,dz$$
$$= g(x)_c$$

- output of smooth classifier $g(x)$ is a vector with entries $g(x)_c = \mathbb{P}_\varepsilon(f(x+\varepsilon) = c)$
- Now denote with $c^* = \arg\max_c g(x)_c$ the most likely class
  $p_x^* = g(x)_{c^*}$ probability of observing c^*.

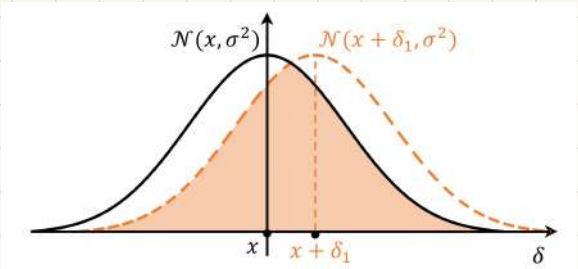
goal: We want to certify that for any admissable perturbation δ it holds

$$\arg\max_c g(x+\delta)_c = c^* \quad \forall \; \|\delta\|_2 \leq r$$

Perturbation of Input Space - 1D Case

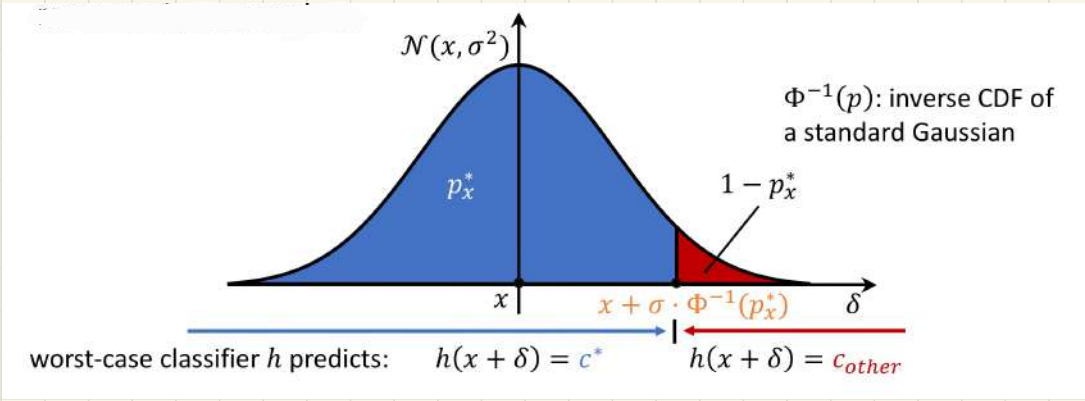
when we slightly perturb x the samples from $\mathcal{N}(x,\sigma^2)$ and $\mathcal{N}(x+\delta,\sigma^2)$ have an overlap ⇒ $g(x+\delta)$ and $g(x)$ produce similar output

How large can we make δ and still gauranlee that $\text{argmax}\ g(x+\delta) = \text{argmax}\ g(x)$



Worst case view on randomized Sampling

- suppose that with a probability p_x^* the classifier predicts class c^* for a sample from $\mathcal{N}(x,\sigma^2)$.
- in the worst case all the samples from c^* are concentrated on one side, and all samples from $c_{other} \neq c^*$ are concentrated on the other side.
- In the worst case: perturbation is orthogonal to the boundry → we move straight towards it

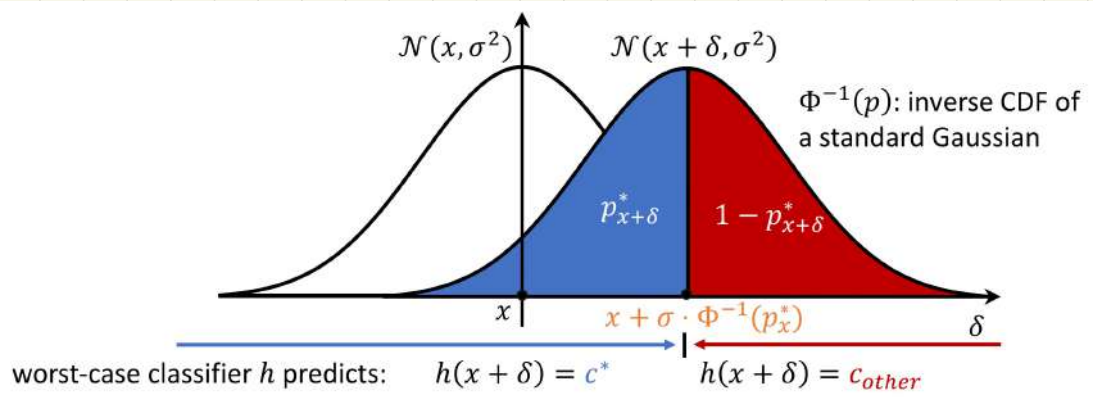


Largest certifiable radius

- if $\|\delta\|_2 < \sigma \Phi^{-1}(p^*_x)$: in worst case
- For and perturbation $\|\delta\|_2 < \sigma \Phi^{-1}(p^*_x)$, the smoothened classifier will not change prediction. i.e $\arg\max g(x+\delta) = \arg\max g(x)$

$p^*_{x+\delta} = 1 - p^*_{x-\delta} = 0.5$

- Large variance σ^2 could lead to large radii $\|\delta\|_2$, it could also lead to reproduce p^*_x by introducing too much noise



How to determine p^*_x ?

in neural nets cannot estimate true p^*_x to find proportion c^* → solution → use Monte Carlo estimation : sample large number of samples from gaussian to estimate $\hat{p}_x$ as number of samples → ∞ we get true proportion p^*_x

- ensure not to overestimate p^*_x

$A = \mathbb{I}[f(x+\varepsilon) = c^*] \in \{0,1\}$ for $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ denote random variable corresponding to event of observing c^*

$A \rightarrow$ Bernoulli r.v with prob. of observing $1 = p^*_x$

- can compute one-sided $(1-\alpha)$ lower confidence interval for p^*_x

- If we sample n times and $A=1$ in $\frac{m}{n}$ the $PCA=m$)

  = Binomial (n, p^*_x)

- We get a lower bound p^v_x from confidence interval of the Binomial.

Practical considerations :

For certification, determine → most likely class c^*
→ Lower bound for p^*_x

- We need to take care from a statistical point of view.
- Using a small set of samples we first take a guess at c^*
- Then, using a large set of samples we compute $\underline{p^*_x}$ based on the confidence interval

Training for Randomized Smoothing

data augmentation
Instead of training on original data
train on perturbed data

1. Suppose we define a **local** variant of the Lipschitz constant around a given point $\mathbf{x}_0$ as follows

$$\mathcal{D}_Y\big(f(\mathbf{x}_0), f(\mathbf{x})\big) \leq k_{\mathbf{x}_0} \cdot \mathcal{D}_X(\mathbf{x}_0, \mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}$$

How does the local constant $k_{\mathbf{x}_0}$ relate to the global constant k of f? Which one would provide better guarantees and why?

1) $k_{x_0} \leq k$ ← global const

local L-constant provides better guarantees assuming $P_{adv}(x_0) \subseteq \mathcal{X}$ i.e L-const holds ∀ inputs -

2. For which class of classifiers does the certificate for the smoothed classifier g equal the certificate for the underlying base classifier f and why?

2)

3. Given two classifiers f_1 and f_2 with Lipschitz constants k_1 and k_2 with $k_1 < k_2$ which one would provide better guarantees? What if we form smoothed classifiers g_1 and g_2?

Problem 1: On slide 15 of the robustness chapter, we have defined an optimization problem for untargeted attacks, i.e. we aim to have the sample $\hat{\boldsymbol{x}}$ classified as **any** class other than the correct one:

$$\min_{\hat{\boldsymbol{x}}} \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda \cdot L(\hat{\boldsymbol{x}}, y)$$

The loss function is defined as:

$$L(\hat{\boldsymbol{x}}, y) = \left[Z(\hat{\boldsymbol{x}})_y - \max_{i \neq y} Z(\hat{\boldsymbol{x}})_i \right]_+ ,$$

where $[\boldsymbol{x}]_+$ is shorthand for $\max(\boldsymbol{x}, 0)$ and $Z(\boldsymbol{x})_i = \log f(\boldsymbol{x})_i$ (i.e. log probability of class i. Here, $L(\hat{\boldsymbol{x}}, y)$ is positive if $\hat{\boldsymbol{x}}$ is classified correctly and 0 otherwise.

Provide an alternative loss function to turn this attack into a targeted attack, i.e. we aim to have the sample $\boldsymbol{x}$ classified as a *specific* target class t.

P1) $\min_{\hat{x}} D(x,\hat{x}) + \lambda L(\hat{x}, y)$

$L(\hat{x}, y) = \left[\max_{i \neq t} Z(\hat{x})_i - Z(\hat{x})_t \right]$

Problem 2: Recall from slide 41 the MILP constraints expressing the ReLU activation function:

$$\begin{aligned} y_i &\leq x_i - l_i(1 - a_i), \\ y_i &<= a_i \cdot u_i, \\ y_i &\geq x_i, \\ y_i &\geq 0, \\ a_i &\in \{0, 1\}, \end{aligned}$$

where $u_i, l_i \in \mathbb{R}$ are upper and lower bounds on the value of the ReLU input x_i.

Show that – for an unstable unit (i.e. $u_i > 0 \wedge l_i < 0$) – a continuous relaxation on a leads to the convex relaxation constraints on slide 54. That is, replacing the constraint $a_i \in \{0, 1\}$ with $a_i \in [0, 1]$ yields

$$(u_i - l_i)y_i - u_i x_i \leq -u_i l_i.$$

$y_i \leq x_i - l_i(1-a_i)$

$y_i \leq u_i a_i$

$y_i \leq \min(x_i - l_i(1-a_i),\ u_i a_i)$

choose a_i s.t it leaves max. leeway for optimization.

$y_i \leq \max_{a_i} \min(x_i - l_i(1-a_i),\ u_i a_i)$

negative slope; -ve; +ve slope

max at intersection

$y_i \leq a_i u_i$ → plugging

$\rightsquigarrow y_i(y_i - l_i) - y_i x_i \leq -u_i l_i$

$a_i = \frac{x_i - l_i}{u_i - l_i}$

plugging in eq

Problem 3: Convex relaxations of non-linearities are not limited to ReLU. For this exercise, we consider the ReLU6 non-linearity

$$\text{ReLU6}(x) = \min(\max(0, x), 6),$$

which is used in MobileNet models performing low-precision computations on mobile devices.

Given input bounds l and u with $l \leq x \leq u$, provide a set of linear constraints corresponding to the convex hull of $\left\{ \begin{pmatrix} x & \text{ReLU6}(x) \end{pmatrix}^T \mid l \leq x \leq u \right\}$.

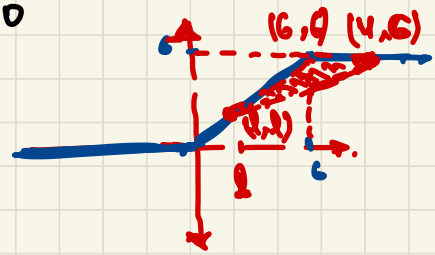
Hint: You have to make a case distinction over different ranges of l and u.

ReLU6(x) = { 0 , x < 0 ; x , 0 ≤ x < 6 ; 6 , x > 6

* if $u_i < 0$ ($l_i < 0$) → $y = 0$

* If $l_i > 0$, $u_i < 0$ $y = x$

* if $l \geq 6$ $y = 6$



* $l < 0$ $0 \leq u \leq 6$ (Behaves like unstable)

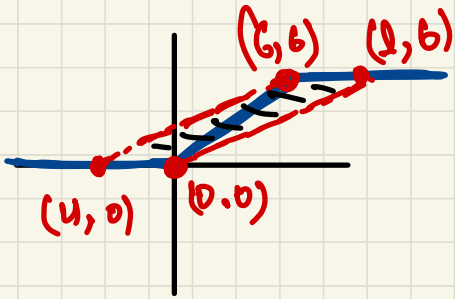
using convex hull to determine $(l,0)^T, (0,0)^T$ & $(u,u)^T$

$y \geq 0$ $y \geq x$ $y \leq \frac{u}{u-l}(x-l)$

* $0 \leq l < 6$ and $u > 6$ convex hull is Δ^2 spanned by

$(l,l)^T$, $(6,6)^T$, $(u,6)^T$

$y \leq x$ $y \leq u$ $y \geq l + \frac{6-l}{u-l}(x-l)$



find the conditions

Problem 4: Last week, we investigated the adversarial robustness of linear classifiers

$$f(x) = \mathrm{I}[\boldsymbol{w}^T\boldsymbol{x} + b > 0]$$

with weight vector $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, mapping samples from $\mathbb{R}^d$ to binary labels $\{0, 1\}$.

Given such a linear classifier f, we can define the randomly smoothed classifier $g : \mathbb{R}^d \mapsto \{0, 1\}$ with

$$g(\boldsymbol{x}) = \mathrm{argmax}_{c\in\{0,1\}} g_c(\boldsymbol{x})$$

and

$$g_c(\boldsymbol{x}) = \Pr_{\boldsymbol{\epsilon}}(f(\boldsymbol{x}+\boldsymbol{\epsilon}) = c) = \begin{cases} \Pr_{\boldsymbol{\epsilon}}\left(\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\epsilon}) + b \le 0\right) & \text{if } c = 0 \\ \Pr_{\boldsymbol{\epsilon}}\left(\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\epsilon}) + b > 0\right) & \text{else} \end{cases},$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^d$ is a random variable.

For this exercise, we assume that $\boldsymbol{\epsilon}$ follows an isotropic normal distribution, i.e. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ with elementwise standard deviation $\sigma \in \mathbb{R}_+$.

As discussed in the lecture, evaluating randomly smoothed classifier is typically not tractable and requires sampling. This is however now the case for our simple linear classifier.

Given input $\boldsymbol{x} \in \mathbb{R}^d$, weights $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, show that $g_0(\boldsymbol{x}) = \Phi_{0,1}\left(-\frac{\boldsymbol{w}^T\boldsymbol{x}}{\sigma\|\boldsymbol{w}\|_2} - \frac{b}{\sigma\|\boldsymbol{w}\|_2}\right)$, where $\Phi_{0,1} : \mathbb{R} \mapsto [0, 1]$ is the cumulative distribution of the standard normal distribution $\mathcal{N}(0, 1)$.

Hint: $\Pr_{\boldsymbol{\epsilon}}\left(\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\epsilon}) + b \le 0\right)$ can alternatively be written as:

$$\int_{\mathbb{R}^d} \mathrm{I}\left[\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\epsilon}) + b \le 0\right] \mathcal{N}(\boldsymbol{\epsilon} \mid \mathbf{0}, \sigma^2\mathbf{I})\, d\boldsymbol{\epsilon}.$$

P4)

$$g_0(x) = \Pr_\epsilon\left(W^T(x+\epsilon) + b \le 0\right)$$

$$= \int_{\mathbb{R}_d} \mathrm{I}\left[W^T(x+\epsilon)+b \le 0\right] \mathcal{N}(\epsilon \mid 0, \sigma^2 I)\, d\epsilon$$

integrating over all possible ϵ

$$z = W^T(x+\epsilon) + b$$

$y \sim \mathcal{N}(\mu, \Sigma)$ Affine transfor $Ax + c$

$\mathcal{N}(A\mu + c,\ A\Sigma A^T)$

$$x \sim N(\mu, \Sigma)$$

$$u = w^T \mu + w^T x + b = w^T x + z$$

$$\sigma = \sqrt{w^T \Sigma w} = \sqrt{w^T w} = \|w\|_2$$

$$\vdots$$

NTF

Chapter 6 : Autoregressive Models

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t$$

↳ autoregressive model of order 'p'

$\phi_1, \phi_2 \ldots \rightarrow$ parameters

$c \rightarrow$ constant

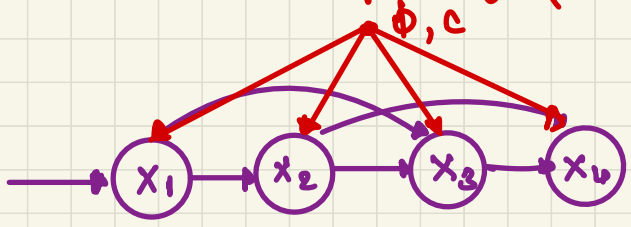
$\varepsilon_i \sim N(0, \sigma)$ is white noise

$X_{t-i} \rightarrow$ lagged value at time i

- perform regression on $([X_{t-1} \ldots X_{t-p}])$ are the inputs and are $(X_t)_t$ the outputs
- Modifications on X_t have repercussions, $(X_t)_t$ are not independent.

Rewrite AR model :

$$P(X_t \mid X_{t-1}, \ldots X_{t-p}) \sim N\left(c + \sum_{i=1}^{p} \phi_i X_{t-1}, \sigma\right)$$

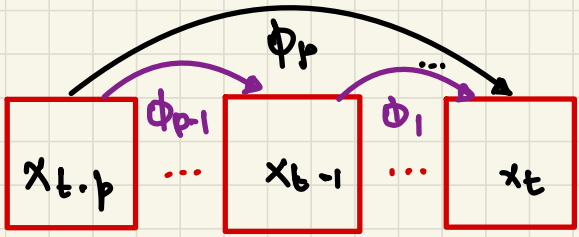


- mean function : $\mu(t) = \mathbb{E}[x_t]$, depends on t

- autocovariance $\gamma(t, i) = \text{cov}(x_t, x_{t-i})$, by default depends on t and i.

- Pearsons autocorrelation function

$$\rho(t, i) = \frac{\text{cov}(x_t, x_{t-i})}{\sqrt{\text{var}(x_t)}\sqrt{\text{var}(x_{t-i})}}$$



Stationary Process:

(i) $\mathbb{E}[x_t] = \mathbb{E}[x_{t-i}] = \mu$, $\forall t, \forall i$ → mean is constant

(ii) $\text{cov}(x_t, x_{t-i}) = \gamma_i$, $\forall t, \forall i$ → autocovariance only depends on time lag.

(iii) $\mathbb{E}[|x_t|^2] < \infty$, $\forall t$

↳ possible to estimate mean and autocovariance by averaging measure over time

- Moments of stationary process AR(p):

- $\mathbb{E}[x_t] = \mu = \dfrac{c}{1 - \sum_{i=1}^{p} \phi_i}$, $\forall t$

- $\text{Var}(x_t) = \gamma_0 = \sum_{j=1}^{p} \phi_j \gamma_{-j} + \sigma^2$, $\forall t$

- $\text{Cov}(x_t, x_{t-i}) = \gamma_i = \sum_{j=1}^{p} \phi_j \gamma_{i-j}$, $\forall t, \forall i \in \mathbb{N}$

- $\rho_i = \dfrac{\gamma_i}{\gamma_0}$

conditions for process to be stationary

AR(p) is stationary iff roots of characteristic polynomial:

- $\Phi(L) = 1 - \sum_{i=1}^{p} \phi_i L^i$ lie outside the circle

Example:

- AR(1): $x_t = c + \phi_1 x_{t-1} + \varepsilon_t$ is stationary if
  - $|\phi_1| < 1$

- AR(2): $x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \varepsilon_t$ (from quadratic equations)

  $\phi_1 + \phi_2 < 1$, $\phi_2 - \phi_1 < 1$, $|\phi_2| < 1$

Parameter Learning

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} = (X^T X)^{-1} X^T y$$

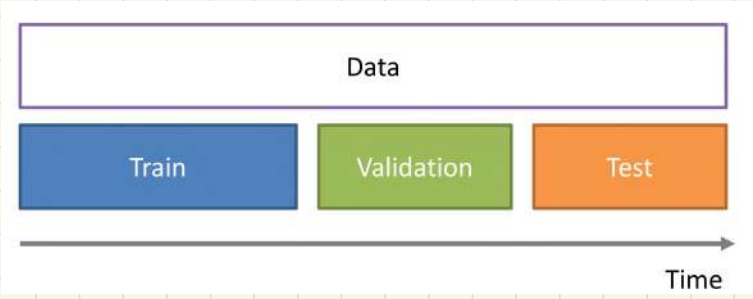
and

$$X = \begin{bmatrix} x_{p-1} & \cdots & x_0 \\ x_p & \cdots & x_1 \\ \vdots & & \vdots \end{bmatrix} \qquad y = \begin{bmatrix} x_p \\ x_{p+1} \\ \vdots \end{bmatrix}$$

Parameter learning from Yule Walker equations

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{p-1} \\ \gamma_p \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_{-1} & \cdots & \gamma_{2-p} & \gamma_{1-p} \\ \gamma_1 & & & & \gamma_{2-p} \\ \vdots & & \ddots & & \vdots \\ \gamma_{p-2} & & & & \gamma_{-1} \\ \gamma_{p-1} & \gamma_{p-2} & \cdots & \gamma_1 & \gamma_0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{p-1} \\ \phi_p \end{bmatrix}$$

Note when splitting the data always maintain the temporal order.



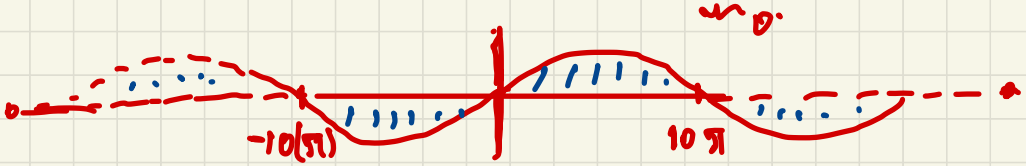
Questions

1. What is the mean $E[X_t]$ of the following processes:
   a) $X_t = \sin(t/10) + \varepsilon_t$ where $\varepsilon_t \sim N(0, \sigma)$
   b) $X_t = 4 - 0.8 * X_{t-1} - 0.1 * X_{t-2} + \varepsilon_t$ where $\varepsilon_t \sim N(0, \sigma)$
   c) $X_t = 4 + X_{t-1} + \varepsilon_t$ where $\varepsilon_t \sim N(0, \sigma)$ and $X_0 \sim N(0, \sigma)$

1) a) $X_t = \sin(t/10) + \varepsilon_t \quad \varepsilon_t \sim N(0, \sigma)$

→ not an autoregressive model.

$E[X_t] = E[\sin(t/10)] + E(\varepsilon_t)$ (0)



we can expect the $\sin\left(\frac{t}{10}\right)$ to evenly cover

$\therefore E\left(\sin\left(\frac{t}{10}\right)\right) = 0$

b) $X_t = 4 - 0.8 * X_{t-1} - 0.1 * X_{t-2} + \varepsilon_t$

$\phi_1 = -0.8 \quad \phi_2 = -0.1$

① $\phi_1 + \phi_2 < 1 \Rightarrow -0.9 < 1$ ✓
② $\phi_2 - \phi_1 < 1 \Rightarrow .7 < 1$ ✓
③ $|\phi_2| = 0.1 < 1$

} satisfied

it autoregressive and stationary process

$E[X_t] = \frac{c}{1 - \sum \phi_i} = \frac{4}{1 + .8 + .1} = \frac{4}{1.9} = \boxed{\frac{40}{19}}$

c) $x_t = y + x_{t-1} + \varepsilon_t$

this is an autoregressive model.

$\phi_1 = 1$ $\quad |\phi_1| < 1$ for process to be stationary

∴ The value keeps on increasing unboundedly. So $\mathbb{E}[x_t]$ does not exist.

2. Does Yule Walker parameter learning assume a stationary process? Why ?

2) Yes, as we need to ensure covariance only depends upon the time lag and not on x_t.

Problem 1: Consider the stationary AR(p) process $X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We denote by μ the mean $E[X_t]$ and by γ_i the autocovariance $Cov(X_t, X_{t-i})$. Show:

1. $\mu = \frac{c}{1-\sum_{i=1}^{p} \phi_i}$, for all t
2. $\gamma_0 = \sum_{j=1}^{p} \phi_j \gamma_{-j} + \sigma^2$
3. $\gamma_i = \sum_{j=1}^{p} \phi_j \gamma_{i-j}$, for all $t, i \in [1, p]$

① $x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon$

taking expectation both sides

$\mathbb{E}[x_t] = \mathbb{E}[c] + \mathbb{E}\left[\sum_{i=1}^{p} \phi_i x_{t-i}\right] + \mathbb{E}[\varepsilon]$

μ $\quad$ c $\quad$ as this is stationary process. $\quad$ 0

$\mu = c + \sum_{i=1}^{p} \phi_i \mathbb{E}[x_{t-i}] \rightarrow = \mu \ \forall\ x_{t-i}$

$\mu\left(1 - \sum_{i=1}^{p} \phi_i\right) = c \rightarrow \boxed{\mu = \frac{c}{1 - \sum_{i=1}^{p} \phi_i}}$

② $\text{cov}(x_t, x_t) = \underbrace{\text{cov}(c, x_t)}_{0} + \text{cov}\left(\sum_{i=1}^{p} \phi_i x_{t-1}, x_t\right) + \underbrace{\text{cov}(\varepsilon_t, x_t)}_{\sigma^2}$

take the cov inside

$= \sum_{i=1}^{p} \phi_i \underbrace{\text{cov}(x_{t-i}, x_t)}_{\gamma_i} + \sigma^2$

$= \sum_{i=1}^{p} \phi_i \gamma_i + \sigma^2$

③ $\text{cov}(x_t, x_{t-j}) = \underbrace{\text{cov}(c, x_{t-j})}_{0} + \sum_{i=1}^{p} \phi_i \text{cov}(\underbrace{x_{t-i}}_{\gamma_{j-i}}, x_{t-j}) + \underbrace{\text{cov}(\varepsilon, x_{t-j})}_{\sigma^2}$

$= \sum_{i=1}^{p} \phi_i \gamma_{j-i} + \sigma^2$

Problem 2:

a) Consider the following AR(1) process $X_t = c + \phi_1 X_{t-1} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Show that this process is stationary iff the following condition is fulfilled

$$|\phi_1| < 1.$$

b) Consider the following AR processes:

- $X_t = c + .8 \times X_{t-1} + .1 \times X_{t-2} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- $X_t = -\sum_{k=1}^{p} \binom{p}{k} X_{t-k} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Are these processes stationary ?

2) AR(1) : $x_t = c + \phi_1 x_{t-1} + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$

for a process to be stationary (AR(1))

$\Phi(L) = 1 - \phi_1 L$ → roots should lie outside unit circle

$1 - \phi L = 0 \Rightarrow$ $\boxed{L = \frac{1}{\phi}}$

$\frac{1}{\phi} > 1 \Rightarrow \phi < 1$

& $\frac{1}{\phi} < -1 \Rightarrow \phi > -1$

$\Rightarrow \boxed{|\phi| < 1}$ final condition

b) (i) $x_t = c + 0.8 x_{t-1} + 0.1 x_{t-2} + \varepsilon$

$\phi_1 = 0.8 \quad \phi_2 = 0.1$

① $\phi_1 + \phi_2 < 1 \Rightarrow .9 < 1$ ✓

② $\phi_2 - \phi_1 < 1 \Rightarrow -.7 < 1$ ✓

③ $|\phi_2| < 1 \Rightarrow .1 < 1$ ✓

} all conditions are met, hence it is a stationary process.

(iii) $x_t = -\sum_{k=1}^{p} \binom{p}{k} x_{t-k} + \varepsilon$

$p(x) = 1 + \sum_{k=1}^{p} \binom{p}{k} x^k$ → roots should lie outside unit circle

$= (1+x)^p$ $x = -1$ (not outside unit circle, ∴ not stationary)

Chapter 6: Markov Chains

Markov Property: $P(x_t \mid x_1, .. x_{t-1}) = P(x_t \mid x_{t-1})$

$t \in \{1, \dots T\}$ (Discrete values)

$x_t \in \{1, 2, .. K\}$

joint distribution

$$P(x_1 = i_1, \dots x_T = i_T) = P(x_1 = i_1) \prod_{t=1}^{T-1} P(x_{t+1} = i_{t+1} \mid x_t = i_t)$$

general case of Markov chain:

$$P(x_1 = i) = \pi_i \qquad P(x_{t+1} = j \mid x_t = i) = A_{ij}^{(t+1)}$$

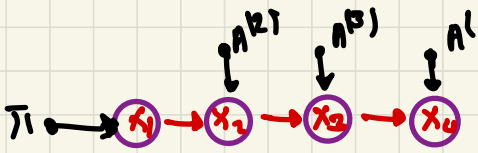
$\pi \in \mathbb{R}^K$ → prior probability

$A^{(t)} \in \mathbb{R}^{K \times K}$ → transition matrix.

consequently joint probability is

$$P(x_1 = i_1, .. x_T = i_T) = \pi_{i_1} \times A^{(2)}_{i_1, i_2} \times \dots \times A^{(T)}_{i_{T-1}, i_T}$$

Parameters $= (K-1) + (T-1)\, K\, (K-1)$



Markov Chain - Stationary case

- time homogenous and stationary Markov Chain

$$P(x_1 = i) = \pi_i \qquad P(x_{t+1} = j \mid x_t = i) = A_{ij}$$

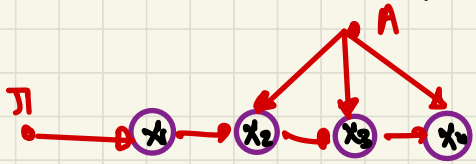
* Transition Matrix $A^{(t)} = A$

↳ INDEPENDENT OF TIME

- joint probability becomes

$$P(x_1 = i_1, \ldots x_T = i_T) = \pi_{i_1} \times A_{i_1, i_2} \ldots \times A_{i_{T-1}, i_T}$$

Parameters $= (K-1) + K(K-1)$



Markov Chains - Learning of Model Parameters

$$P(\text{all}) = \prod_{n=1}^{N} P(x_1^{(n)}) \prod_{t=1}^{T_n - 1} P(x_{t+1}^{(n)} \mid x_t^{(n)}) = \left(\prod_{k=1}^{K} \pi_k^{L(k)} \right) \left(\prod_{i=1}^{K} \prod_{j=1}^{K} A_{ij}^{N(i,j)} \right)$$

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

minimizing $P(\alpha, \beta)$ s.t $\sum_k \pi_k = 1$ and $\sum_j A_{ij} = 1$

$$A_{ij} = \frac{N(i,j)}{\sum_{j'} N(i,j')} \qquad \pi_k = \frac{L(k)}{\sum_{k'} L(k')}$$

More insights :

Task 1 $\quad A_{ij}(n) = P(x_{t+n} = j \mid x_t = i)$

Probability of going from state i to j in exactly n steps

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

$A(n) = A(n-1)\, A \xrightarrow{A(1) = A} A(n) = A^n$

Chapman Kolgomorov Equation

$$A_{ij}(m+n) = \sum_{k=1}^{K} A_{ik}(m)\, A_{kj}(n) = A(m+n) = A(m)\, A(n)$$

Task 2 $\quad \pi_j(t) = Pr(x_t = j)$

Probability of reaching state j in t timesteps

$$Pr(x_t = j) = \sum_{i=1}^{K} Pr(x_t = j \mid x_{t-1} = i)\, Pr(x_{t-1} = i)$$

$$= \sum_{i=1}^{K} A_{ij}\, \pi_i(t-1)$$

$$\Rightarrow \pi(t) = \pi(t-1)\, A$$

$$\pi(t) = \pi A^{t-1}$$

* $\pi(t)$ and π are row vectors.

Questions

1. We assume that $X_t \in \{1, 2, 3\}$. We consider $\boldsymbol{\pi} = \begin{bmatrix} 0.0 \\ 0.5 \\ 0.5 \end{bmatrix}$ and $\boldsymbol{A} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$.

   a) What is the probability to observe the sequence $\boldsymbol{X}^{(1)} = [1, 2, 3]$?

   b) What is the probability to observe the sequence $\boldsymbol{X}^{(2)} = [2, 2, 3]$?

a) $x^{(1)} = [1 \;\; 2 \;\; 3]$.

$= \pi(1) * A_{12}\, A_{23}. \quad = 0.$ ($\pi(1) = 0$)

b) $x^{(2)} = [2, 2, 3]$

$= \pi(2) * A_{22} * A_{23}. = 0.5 \times 0.5 \times 0.4$

$= 0.1$

2. We assume that $X_t \in \{1, 2, 3\}$ and we observed three sequences:
   - $\boldsymbol{X}^{(1)} = [1, 3, 2]$
   - $\boldsymbol{X}^{(2)} = [3]$
   - $\boldsymbol{X}^{(3)} = [1, 1, 3, 2]$

   What is the MLE of the transition matrix $\boldsymbol{A} \in \mathbb{R}^{3\times 3}$?

2)

transition	occurances
1 → 1	# 1
1 → 2	# 0
1 → 3	# 2

transition
2 → 1
2 → 2
2 → 3

} no occurances

transition	occurances
3 → 1	0
3 → 2	2
3 → 3.	0

$\pi(1) = \#2$ $\quad \pi(2) = 0$ $\quad \pi(3) = 1.$

$$\pi = \begin{bmatrix} 2/3 \\ 0 \\ 1/3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1/2 & 0 & 2/3 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1 & 0 \end{bmatrix}$$

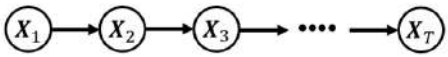
there is no data available for state 2

so, we make a reasonable assumption that transitions are equally likely.

Problem 3: Let $\mathbf{X}_t$ be a 2-D random vector:

$$\mathbf{X}_t = \begin{bmatrix} u_t \\ v_t \end{bmatrix}, \quad \text{where } u_t, v_t \in \{1, 2, ..., K\}. \tag{1}$$

Consider the following Markov chain.



Model parameters are as follows:

- initial distribution $\boldsymbol{\pi}_x \in \mathbb{R}^{K \times K}$ that parametrizes $Pr(\mathbf{X}_1)$:

$$Pr(\mathbf{X}_1 = \begin{bmatrix} i \\ j \end{bmatrix}) = \boldsymbol{\pi}_x(i, j). \tag{2}$$

- transition probability matrix $\mathbf{A}_x \in \mathbb{R}^{K \times K \times K \times K}$ that parametrizes $Pr(\mathbf{X}_{t+1} | \mathbf{X}_t)$:

$$Pr(\mathbf{X}_{t+1} = \begin{bmatrix} i_{t+1} \\ j_{t+1} \end{bmatrix} \mid \mathbf{X}_t = \begin{bmatrix} i_t \\ j_t \end{bmatrix}) = \mathbf{A}_x(i_t, j_t, i_{t+1}, j_{t+1}). \tag{3}$$

The joint probability can be factorized as:

$$Pr(\mathbf{X}_1, ..., \mathbf{X}_T) = Pr(\mathbf{X}_1) \prod_{t=1}^{T-1} Pr(\mathbf{X}_{t+1} | \mathbf{X}_t).$$

In this task, we refer to this model as *"2-D first-order Markov chain"*.

a) Does the sequence $[u_1, ..., u_T]$ (where $u_t \in \{1, 2, ..., K\}$ is defined in Eq. (1)) have the first-order Markov property? Why or why not?

a) u_t depends on both u_{t-1} and v_{t-1} and v_{t-1} depends on u_{t-2} & v_{t-2}.

∴ The first order Markov property would not hold

b) Let $[Y_1, ..., Y_T]$ be a 1-D first-order Markov chain with the following initial and transition probabilities ($Y_1, ..., Y_T$ are binary-valued).

$$\boldsymbol{\pi}_y = [0.5 \quad 0.5], \qquad \mathbf{A}_y = \begin{bmatrix} 0.2 & 0.8 \\ 0.5 & 0.5 \end{bmatrix}.$$

- Briefly explain why the sequence $\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} Y_2 \\ Y_3 \end{bmatrix}, ..., \begin{bmatrix} Y_{T-1} \\ Y_T \end{bmatrix}$ is a 2-D first-order Markov chain.
- Compute initial and transition probabilities, $\boldsymbol{\pi}_x$ and $\mathbf{A}_x$ (defined in Eqs. 2 and 3) for the sequence $\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \begin{bmatrix} Y_2 \\ Y_3 \end{bmatrix}, ..., \begin{bmatrix} Y_{T-1} \\ Y_T \end{bmatrix}$.

b) (i) $\left\{ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \begin{bmatrix} y_2 \\ y_3 \end{bmatrix}, \ldots \begin{bmatrix} y_{T-1} \\ y_T \end{bmatrix} \right\}$ → 2D Markov chain

$$\Pr\left(\begin{bmatrix} y_{T-1} \\ y_T \end{bmatrix} \middle| y_{t-1}, y_{t-2}, \ldots y_1 \right) \longrightarrow (1)$$

as $\{y_1, y_2, \ldots y_T\}$ is 1-D Markov

$$\Pr(y_t \mid y_{t-1}, \ldots y_1) = \Pr(y_t \mid y_{t-1})$$

using the results in (1)

$$\Pr\left(\begin{bmatrix} y_{t-1} \\ y_t \end{bmatrix} \middle| y_{t-1}, \ldots y_1 \right) = \Pr\left(\begin{bmatrix} y_{t-1} \\ y_t \end{bmatrix} \middle| \begin{bmatrix} y_{t-1} \\ y_{t-2} \end{bmatrix} \right)$$

(ii) $\pi_x(i,j) = \Pr\left(\begin{bmatrix} y_2 \\ y_1 \end{bmatrix} = \begin{bmatrix} j \\ i \end{bmatrix} \right) = A_y(j,i)\,\pi_y(i)$

$$A_x(i,j,i',j') = \Pr\left(\begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix} = \begin{bmatrix} i \\ j \end{bmatrix} \middle| \begin{bmatrix} y_{t-1} \\ y_{t-2} \end{bmatrix} = \begin{bmatrix} i' \\ j' \end{bmatrix} \right)$$

$$= \begin{cases} 0 & j \neq i' \\ A_y(i,j) & \text{otherwise} \end{cases}$$

Chapter 7 : Hidden Markov Models :

Hidden Markov Model (HMM) is composed of a sequence of hidden / latent variables $[z_1, \dots z_T]$ and a sequence of observed variables $[x_1, \dots x_T]$

- r.v $z_1, z_2, \dots z_T$ satisfy the Markov property

$$p(z_{t+1} | z_t, \dots z_1) = p(z_{t+1} | z_t)$$

transition probability

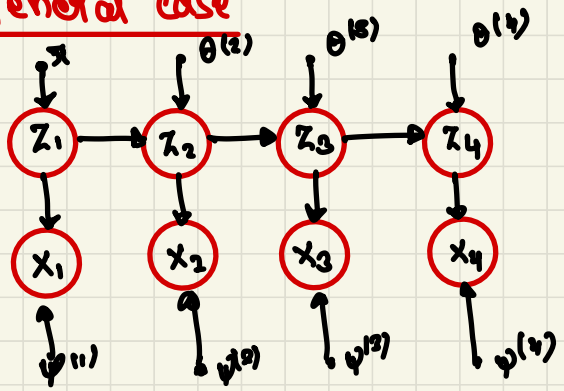
- Distribution of x_t only depends on z_t.

$$p(x_{t+1} | z_1, \dots z_T, x_1, \dots x_T) = p(x_{t+1} | z_{t+1})$$

emission probability

- discrete time $t \in \{1, 2, \dots T\}$ and discrete r.v $z \in \{1, 2, \dots K\}$
- observed data can be discrete or continuous

HMM : general case



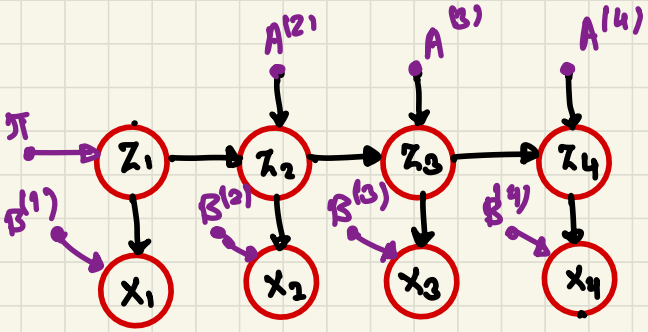
- joint distribution can be written as:

$$P(Z_1 = z_1 \dots, Z_T = z_T, X_1 = x_1, \dots X_T = x_T)$$
$$= P(Z_1 = z_1 ; \pi) \prod_{t=1}^{T} \left(P(Z_{t+1} = z_{t+1} \mid Z_t = z_t ; \theta^{(t+1)}) \right) * \prod_{t=1}^{T} P(X_t = x_t \mid Z_t = z_t ; \varphi^{(t)})$$

HMM : Discrete Case :

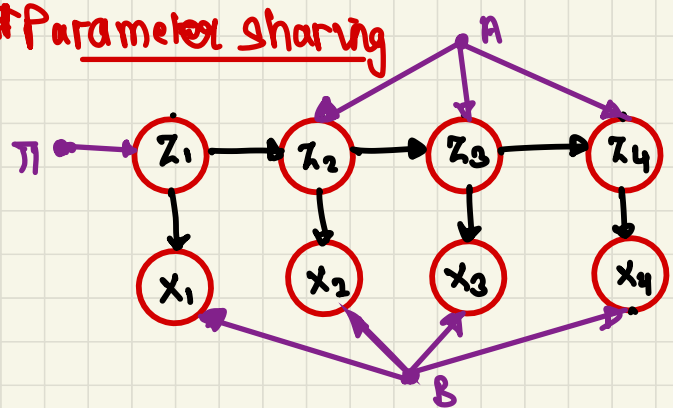
$$x_t \in \{1, 2, \dots K\}$$

$$P(z_1 = i) = \pi_i$$
$$P(z_{t+1} = j \mid z_t = i) = A_{ij}^{(t+1)}$$
$$P(x_{t+1} = j \mid z_{t+1} = i) = B_{ij}^{(t+1)}$$



Parameters
$= K + (T-1)K^2 + TKK'$

Parameter sharing



Parameters

$K + K^2 + KK'$

From now on assume parameter sharing in the Markov Chains

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

Tasks concerning HMM :

- **Inference :**
  - model parameters are fixed
  - info from posterior $Pr(z_{1:T} \mid x_{1:T})$
- **Parameter Learning :**
  - we seek to learn the model parameters
  - $x_{1:T}$ is observed
  - $z_{1:T}$ is (usually) not observed

Inference

- **Filtering :** $Pr(z_t \mid x_{1:t})$ → incrementally as data streams in
- **Smoothing :** $Pr(z_t \mid x_{1:T})$ Offline past and future
- **MAP Inference :** computes $\arg\max_{z_{1:T}} Pr(z_{1:T} \mid x_{1:T})$ → mode of posterior, viterbi decoding

The Forward Algorithms :

- Goal : incrementally compute $P(z_t | x_{1:t})$

Bayes rule gives

$$P(z_t = k | x_{1:t}) = \frac{P(z_t = k, x_{1:t})}{\sum_{j=1}^{K} P(z_t = j, x_{1:t})}$$

- for convinience, we denote

$$\alpha_t(k) \overset{def}{=} P(z_t = k, x_{1:t}) \quad \text{and} \quad \alpha_t = \begin{bmatrix} \alpha_t(1) \\ \vdots \\ \alpha_t(K) \end{bmatrix}$$

Hence, we have

$$P(z_t = k | x_{1:t}) = \frac{\alpha_t(k)}{sum(\alpha_t)} = \frac{\alpha(k)}{\sum_{k'} \alpha_t(k')}$$

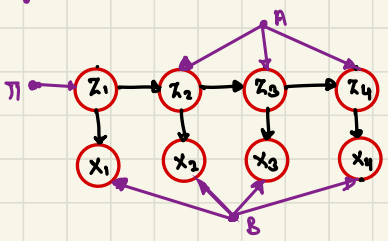
Forward algo computes recursively

1) computes α_1 (initialization)

2) Given α_t, compute α_{t+1} (recursion)

initialization : computation of parameter α_1

$$\alpha_1(k) = P(z_1 = k, x_1)$$
$$= P(z_1 = k) P(x_1 | z_1 = k)$$
$$= \pi_k B_{k x_1}$$



Recursion : given α_t, we can compute α_{t+1}

$$\alpha_{t+1}(k) = P(z_{t+1} = k, x_{1:t+1})$$

$$\alpha_{t+1}(k) = B_{k x_{t+1}} \sum_{j=1}^{K} A_{jk} \alpha_t(j)$$

$$\alpha_{t+1} = B_{:(x_{t+1})} \odot A^T \alpha_t$$

$O(TK^2)$: linear in T

$$\begin{bmatrix} \alpha_{t+1}(1) \\ \alpha_{t+1}(2) \\ \vdots \\ \alpha_{t+1}(K) \end{bmatrix} = \begin{bmatrix} B(1, x_{t+1}) \\ B(2, x_{t+1}) \\ \vdots \\ B(K, x_{t+1}) \end{bmatrix} \odot \left[\boxed{A^T} \begin{bmatrix} \alpha_t(1) \\ \alpha_t(2) \\ \vdots \\ \alpha_t(K) \end{bmatrix} \right]$$

The Forward Backward Algorithm:

incrementally compute $P(z_t \mid x_{1:T})$ ⟵ full data this time

Bayes Rule :

$$P(z_t = k \mid x_{1:T}) = \frac{P(z_t = k, x_{1:T}) P(x_{t+1:T} \mid z_t = k)}{\sum_{j=1}^{K} P(z_t = j, x_{1:T})}$$

for convinience

$\beta_t(k) = P(x_{t+1:T} \mid z_t = k)$ and $\beta_t = \begin{bmatrix} \beta_t(1) \\ \vdots \\ \beta_t(K) \end{bmatrix}$

using $\alpha_t(k)$ and $\beta_t(k)$

$$P(z_t = k \mid x_{1:T}) \propto \alpha_t(k)\, \beta_t(k)$$

Backward algorithm compulates recursively

1) computes β_T (initialization)
2) given $\beta_{t+1} \rightarrow$ compute β_t

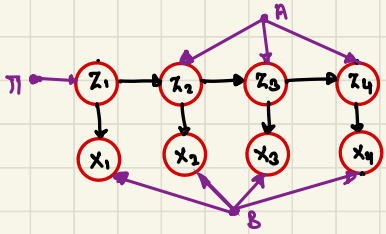
initialization $\beta_T(k) = 1$

for $t = T$ term is completely captured.

Recursion: given β_{t+1} we compute β_t

$$\beta_t(j) = P(x_{t+1:T} \mid z_t = j)$$

$$= \sum_{k=1}^{K} A_{jk} B_{k x_{t+1}} \beta_{t+1}(k)$$



$$\beta_t = A\,(B_{:(x_{t+1})} \odot \beta_{t+1})$$

$$\begin{bmatrix} \beta_t(1) \\ \beta_t(2) \\ \vdots \\ \beta_t(K) \end{bmatrix} = \boxed{A} \left(\begin{bmatrix} B(1, x_{t+1}) \\ B(2, x_{t+1}) \\ | \\ | \\ B(K, x_{t+1}) \end{bmatrix} \odot \begin{bmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \\ | \\ | \\ \beta_{t+1}(K) \end{bmatrix} \right)$$

$\beta_{1:T}$ takes $O(TK^2)$ operations, linear in T

Forward Backward : Application I

① Prob. of being in state k at time t online:

$$P(z_t = k \mid x_{1:t}) = \frac{\alpha_t(k)}{\sum_s \alpha_t(s)}$$

via argmax we can get most likely state

② compute the probability of being in state k at time t offline

$$\gamma_t(k) = p(z_t = k \mid x_{1:T}) = \frac{\alpha_t(k)\,\beta_t(k)}{\sum_s \alpha_t(s)\,\beta_t(s)}$$

③ compute probability of two "adjacent" states having specific realizations.

$$\xi_t(i,j) = P(z_t = i, z_{t+1} = j \mid x_{1:T}) = \frac{\alpha_t(i)\, A_{ij}\, \beta_{t+1}(j)\, B_{j x_{t+1}}}{\sum_u \sum_v \alpha_t(u)\, A_{uv}\, \beta_{t+1}(v)\, B_{v x_{t+1}}}$$

MAP inferences in HMM

given $x_{1:T}$ → find most probable hidden state $z_1, \dots z_T$

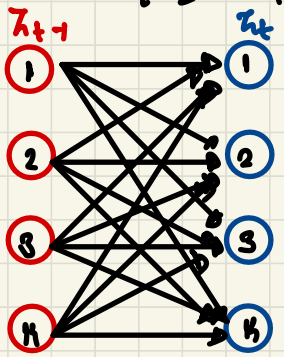
Posterior distribution $Pr(z_{1:T} \mid x_{1:T})$

$$\arg\max_z P(z_{1:T} \mid x_{1:T}) = \arg\max_z \log\left[P(z_{1:T}, x_{1:T})\right]$$

$$= \arg\max_z \log\left[P(z_1) P(x_1 \mid z_1)\right] + \sum_{t=2}^{T} \log\left[P(z_t \mid z_{t-1}) P(x_t \mid z_t)\right]$$

$\log\left[P(z_t \mid z_{t-1}) P(x_t \mid z_t)\right]$

depends on z_{t-1} and z_t



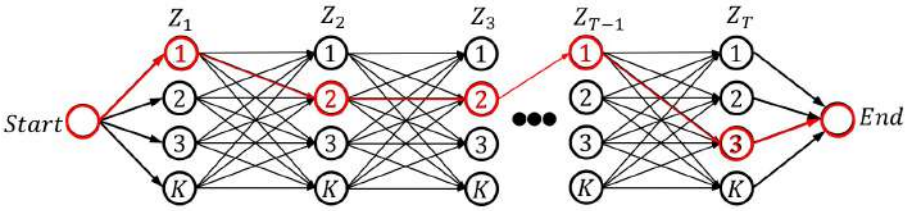
think of bi-partite graph

weight of the edge $i \to j$

$$= -\log\left[P(z_t = j \mid z_{t-1} = i)\, P(x_t \mid z_t = j)\right]$$

Formulate : Viterbi algorithm :

- We can formulate the MAP inference as a shortest-paths problem.
  - weights of edges connected to the *Start* node: $-\log[\Pr(Z_1 = j)\Pr(X_1|Z_1 = j)]$
  - weights of the intermediate layers: $-\log[\Pr(Z_t = j|Z_{t-1} = i)\Pr(X_t|Z_t = j)]$
  - weights of the edges connected to the *End* node: 0



Each directed path corresponds to an assignment to variables $Z_{1:T}$.
Sum of edge weights $= -logPr(Z_{1:T}, X_{1:T})$

complexity: $O(TK^2)$

Called Viterbi algorithm

Parameter learning :

model parameters $\theta = \{\pi, A, B\}$

goal : Solve $\max_{\theta} \log p_\theta(x)$

using ELBO :

$$\mathbb{E}_{P(Z_{1:T}|X_{1:T},\theta^{old})}[\log P(X_{1:T}, Z_{1:T}, \theta)] = \sum_k P(Z_1 = k|X_{1:T}, \theta^{old}) \log(\pi_k)$$
$$+ \sum_{i,j}\sum_t P(Z_t = i, Z_{t+1} = j|X_{1:T}, \theta^{old}) \log(A_{ij})$$
$$+ \sum_i \sum_t P(Z_t = i|X_{1:T}, \theta^{old}) \mathbb{I}(x_t = j) \log(B_{ij})$$

using forward-Backward, blue terms can be computed.

Questions

1. Does the sequence $[Z_1, \ldots, Z_T]$ fullfill the Markov property ? Why ?

1) yes as $Pr(z_t \mid z_{t-1}, \ldots z_1) = P(z_t \mid z_{t-1})$.

2. Does the sequence $[X_1, \ldots, X_T]$ fullfill the Markov property ? Why ?

2) $Pr(x_t \mid x_{1:t}, z_{1:t}) = Pr(x_t \mid z_t)$.

$z_t \rightarrow z_{t-1}$ ∴ not Markovian

3. In Variational Inference we sometimes need to approximate the ELBO (for example by sampling from the the latent Z from the variational distribution). For learning HMM parameters, do we need to sample Z as well?

3) No, as closed form soln of forward backward is possible

Problem 1: Consider an HMM where hidden variables are in $\{1, 2\}$ and observed variables are in $\{a, b, c\}$. Let the model parameters be as follows:

$$A = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} 0.2 & 0.8 \\ 0.5 & 0.5 \end{bmatrix} \end{array} \quad B = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 0.2 & 0 & 0.8 \\ 0.4 & 0.6 & 0 \end{bmatrix} \end{array} \quad \pi = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

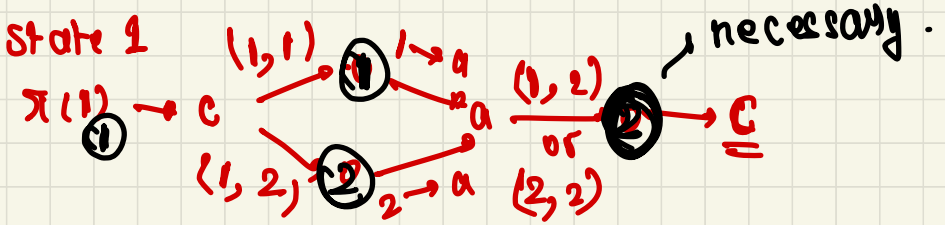
Assume that the sequence $X_{1:5} = [cabac]$ is observed.

1. Filtering: find the distribution $P(Z_3|X_{1:3})$.
2. Smoothing: find the distribution $P(Z_3|X_{1:5})$.
3. Viterbi algorithm: find the most probable sequence $[Z_1, \ldots, Z_5]$.

1) Filtering $P(Z_3 | X_{1:3})$ use forward algorithm

$X_{1:5} = [c \; a \; b \; a \; c]$

c → a → b



necessarily $P(Z_3 | X_{1:3}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

As State 1 → b is not possible

naive soln: $\alpha_1 = \pi \odot B_{:c} = \begin{bmatrix} .2 \\ .5 \end{bmatrix} \odot \begin{bmatrix} .8 \\ 0 \end{bmatrix} = \begin{bmatrix} .4 \\ 0 \end{bmatrix}$

$\alpha_2 = B_{:a} : A^T \alpha_1 =$ — —

2) smoothing : We will use Forward Backward Algorithm

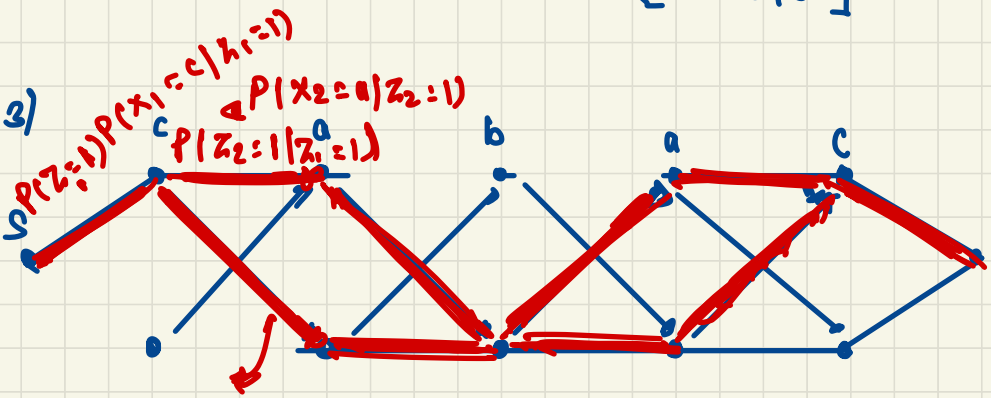
clever way is by observation because $x_3 = b$

$$P(z_3 \mid x_{1:3}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\beta_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \beta_4 = A(B_{:c} \odot \beta_5) = \begin{bmatrix} 0.16 \\ 0.4 \end{bmatrix}$$

$$\beta_3 = A(B_{:a} \odot \beta_4) = \begin{bmatrix} 0.1344 \\ 0.096 \end{bmatrix}$$

3)

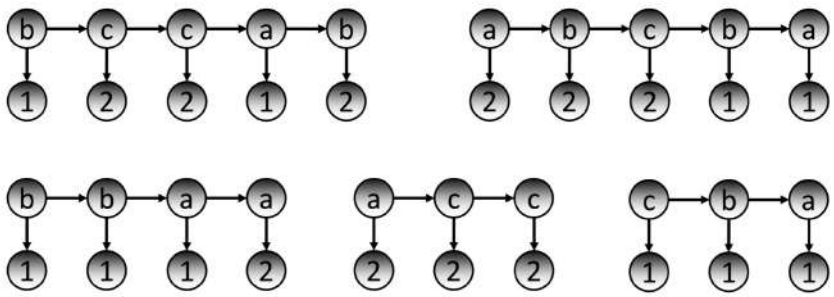


$P(z_2=2 \mid z_1=1)$

* $P(x_2=a \mid z_2=2)$

• calculate the shortest path.

Problem 2: Consider an HMM where states Z_t are in $\{a, b, c\}$ and emissions X_t are in $\{1, 2\}$. Given is the following set of fully-observed instances (two sequences of length 5, one sequence of length 4, and two sequences of length 3):



Learn the parameters of the HMM (i.e. $\pi \in \mathbb{R}^3, \mathbf{A} \in \mathbb{R}^{3\times3}$, and $\mathbf{B} \in \mathbb{R}^{3\times2}$) using maximum-likelihood estimation.

$$\log P(S_1) = \log P(z_1) + \sum_{t=2}^{T_1} \log P(z_t \mid z_{t-1}) + \sum_{t=1}^{T_1} \log P(x_t \mid z_t)$$

using indicator function and A, B, π

$$= \sum_k \mathbb{I}(z_1 = k) \log \pi(k) + \sum_{t=2}^{T_1} \sum_{i,j} \mathbb{I}(z_t = j, z_{t-1} = i) \log A(i,j)$$

$$+ \sum_{t=1}^{T_1} \sum_{i',j'} \mathbb{I}(x_t = j' \mid z_t = i') \log B(i',j')$$

$$\log P(D): \sum_{n=1}^{S} \sum_k \mathbb{I}(z_1 = k) \log \pi(k) + \sum_{n=1}^{S} \sum_{t=2}^{T_n} \sum_{i,j} \mathbb{I}(z_t = j, z_{t-1} = i) \log A(i,j)$$

$$+ \sum_{n=1}^{S} \sum_{t=1}^{T_n} \sum_{i',j'} \mathbb{I}(x_t = j' \mid z_t = i') \log B(i',j')$$

$L(k)$ $\qquad$ $N(i,j)$

$$= \sum_k \sum_{n=1}^{S} \mathbb{I}(z_1 = k) \log \pi(k) + \sum_{i,j} \sum_{n=1}^{S} \sum_{t=2}^{T_n} \mathbb{I}(z_t = j, z_{t-1} = i) \log A(i,j)$$

$M(i,j')$

$$+ \sum_{i',j'} \sum_{n=1}^{S} \sum_{t=1}^{T_n} \mathbb{I}(x_t = j' \mid z_t = i') \log B(i',j')$$

$$L = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \qquad N = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix} \qquad M = \begin{bmatrix} 5 & 3 \\ 2 & 5 \end{bmatrix}$$

Neural sequence Model

Word2vec:

① CBOW: → • Predict a word from word surrounding
• Not good for rare words

② skip gram → • Predict context given center word
• good for rare words

Skip grams:

- i/p : one hot vector with N dimension
- embedding : project the word to D-dimensional space with $U \in \mathbb{R}^{N \times D}$
- Prediction : multiply embedding with $V \in \mathbb{R}^{N \times D}$ and apply softmax

- for $S = \{x_{i-1}, \ldots x_{i-1}, x_{i+1}, \ldots x_{i+2}\}$ is a window of size 2 around the word x_i

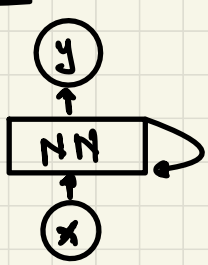
$$\max E[P(S|x_i, \theta)] = \min_{\theta} (-E[P(S|x_i, \theta)])$$

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

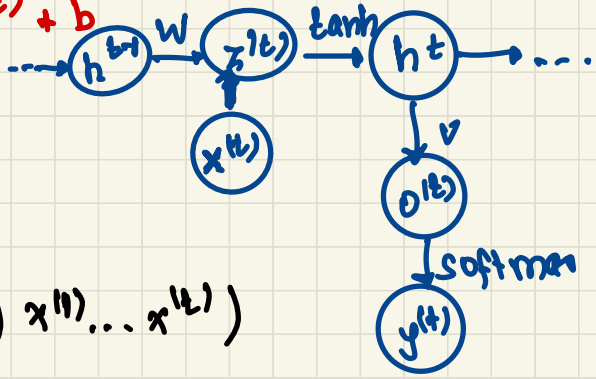
and $P(x_k|x_i, \theta) = \text{Softmax}(u_i v)$

RNN:

- given sequence $\{x^{(1)}, \ldots x^{(n)}\}$ and o/p $\{y^{(1)} \ldots y^{(n)}\}$ predict prob $P(y^{(t)}|x^{(1)} \ldots x^{(t)})$



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



weights are shared

$$L = -\log \prod_t p_{\text{model}}(y^{(t)} \mid x^{(1)} \dots x^{(t)})$$

$$= -\sum_t \log p_{\text{model}}(y^{(t)} \mid x^{(1)} \dots x^{(t)}) = -\sum_t L^{(t)}$$

$$\frac{\partial L}{\partial V} = \sum_t (\hat{y}^{(t)} - y^{(t)})(h^{(t)})^T$$

$$\frac{\partial L}{\partial W} = \sum_t \text{diag}(1 - h^{(t)2}) \frac{\partial L}{\partial h^{(t)}} (h^{(t-1)})^T$$

$$\frac{\partial L}{\partial U} = \sum_t \text{diag}(1 - (h^{(t)})^2) \frac{\partial L}{\partial h^{(t)}} (x^{(t)})^T$$

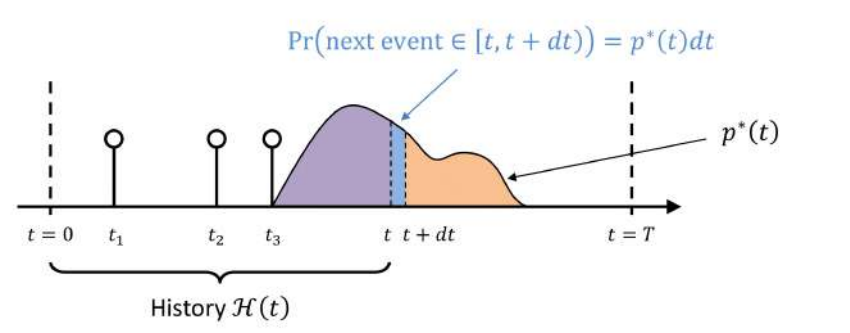
Chapter 9

Temporal point processes

class of probabilistic models that describe the distributio of discrete event sequence in continuous time

defines a generative model for variable-length sequences $t = \{t_1, t_2, .. t_N\}$ on the interval $[0, T]$

Modelling the time of next event:



model the distribution $p(\{t_1, .. t_N\})$ autoregressively

- predict the time of next event t_i given history $\mathcal{H}(t) = \{t_j < t\}$
- important: $\mathcal{H}(t)$ depends on the specific sample $\{t_1, .. t_N\}$!

conditional density $p^*(t) = p(t \mid \mathcal{H}(t))$

Accumulative density function

$$F^*(t) = \int_{t_{i-1}}^{t} p^*(u)\,du = \Pr(\text{next event happens in } [t_{i-1}, t))$$

t_{i-1} is the last event ~~that~~ before t happens

Survival function :

- $S^*(t) = 1 - F^*(t) = \int_t^\infty p^*(u)\,du$ → Pr. that next event does not take place before t.

or probability that next event happens after t.

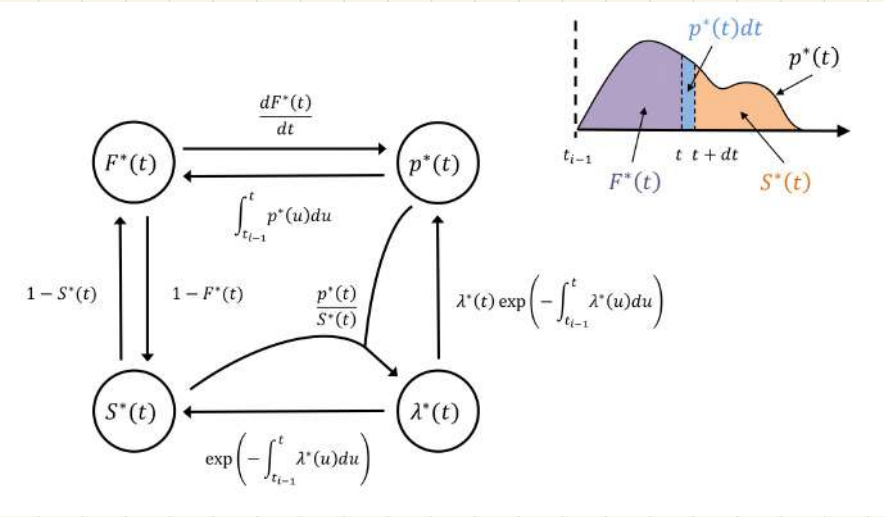
conditional intensity :

- $\lambda^*(t)\,dt$: Pr. of event in $[t, t+dt)$ given no event in $[t_{i-1}, t)$

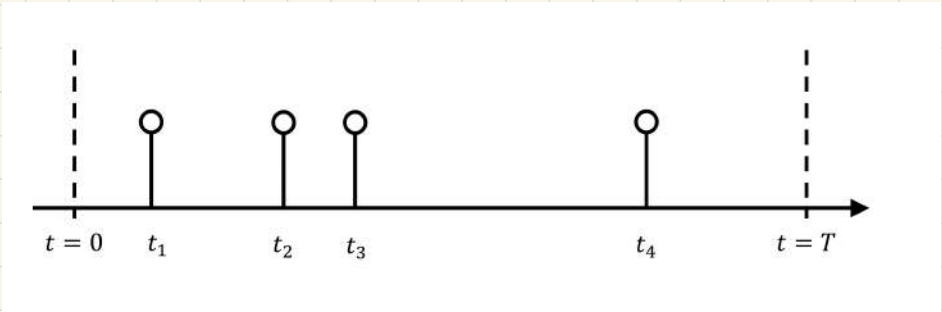
- $\lambda^*(t)\,dt = \dfrac{Pr[\text{event in } [t, t+dt) \text{ \& no event in } [t_{i-1}, t)]}{Pr(\text{no event in } [t_{i-1}, t))}$

$\lambda^*(t)$ → expected # of events / unit time $= \dfrac{p^*(t)\,dt}{S^*(t)}$

Relationship b/w p^*, F^*, S^*, λ^*



* Likelihood of entire sequence

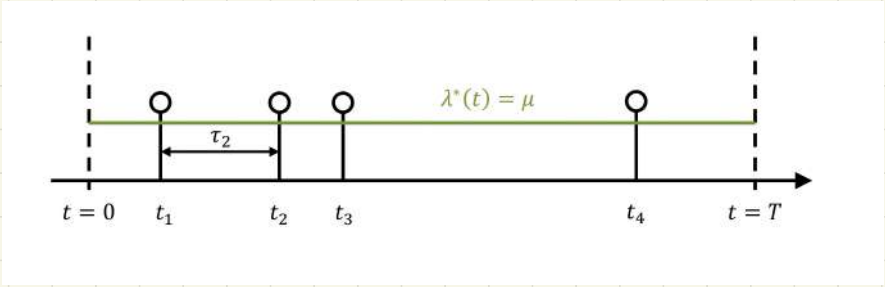


$$p(\{t_1, t_2, t_3, t_4\}) = p^*(t_1)\, p^*(t_2)\, p^*(t_3)\, p^*(t_4)\, S^*(T)$$

$$= \lambda_1^*(t_1)\, \lambda^*(t_2)\, \lambda^*(t_3)\, \lambda^*(t_4) \exp\left(-\int_0^T \lambda^*(u)\, du\right)$$

* number of events can vary

* Homogenous Poisson Process (HPP) :



- simplest model : constant intensity
  $\lambda^*(t) = \mu$
- inter event times follow exponential distro.

$$p^*(t) = \mu \exp\left(-\int_{t_{i-1}}^{t} \mu\, d\mu\right) = \mu \exp(-\mu(t - t_i))$$

inter-event time τ_i

* How to simulate HPP? → generating inter-event times

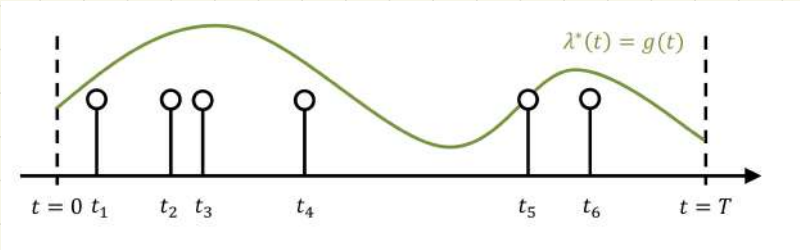
```
arrival_times = []
t = 0
while t < T:
    tau ~ Exponential(mu)
    t += tau
    if t < T:
        arrival_times.append(t)
```

* Sampling from exponential distro : inverse CDF

$$u = F(\tau) = 1 - \exp(-\mu\tau) \qquad u \sim \text{Uniform}(0,1)$$

$$\tau = F^{-1}(u) = -\frac{1}{\tau}\log(1-u)$$

* Inhomogenous Point Process (IPP)



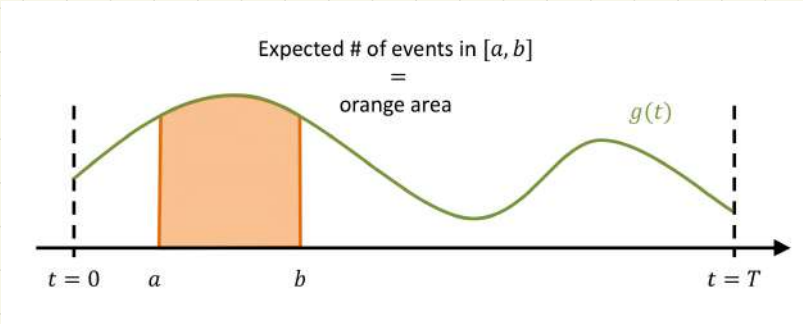
- intensity changes over time : $\lambda^*(t) = g(t) \geq 0$
- intensity is independent of history

Expected number of events

Number of events in $[a,b] \subseteq [0,T]$ follow Poisson

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

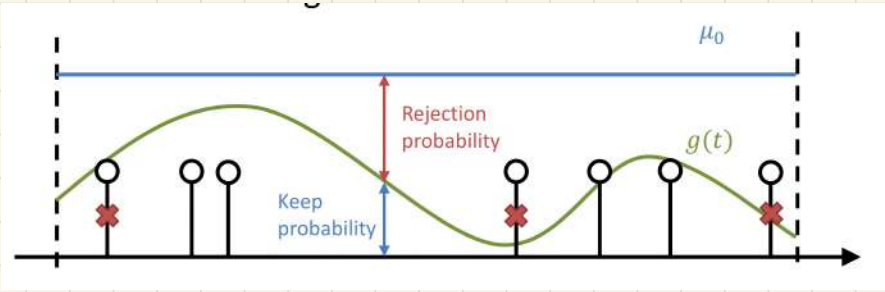
* the expected number of events inside $[a,b]$ is equal to total intensity over the region



combination of two IPPs ($g(t)$ & $h(t)$) is also an IPP with intensity $g(t) + h(t)$

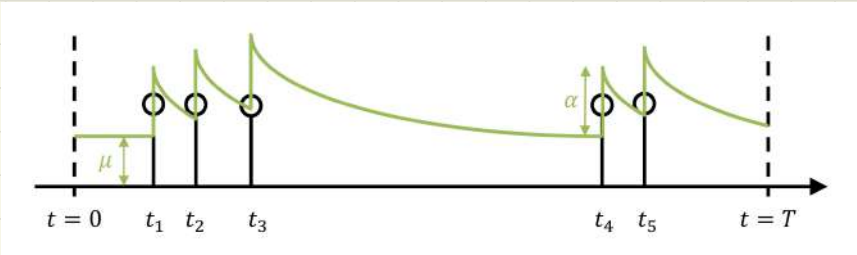
simulating an IPP

integration is hard → use the approach: thinning



1) find an upper bound $\mu_0 \geq g(t)$ $\forall$ t
2) simulate candidate events $\{t_1, \ldots\}$ from a HPP with rate μ_0.
3) keep each t_i with probability $g(t_i) / \mu_0$.

Hawke's Process



self-exciting process

$$\lambda^*(t) = \mu + \alpha \sum_{t_i \in H(t)} \kappa_\omega(t - t_j)$$

- Triggering kernel $\kappa_\omega(t - t_i) = \exp(-\omega(t - t_i))$
- Parameters $\mu, \alpha, \omega \geq 0$
- intensity depends on History

Parameter Estimation in TPPs:

- Parametric conditional intensity $\lambda^*_\theta(t)$
- Maximize the log-likelihood

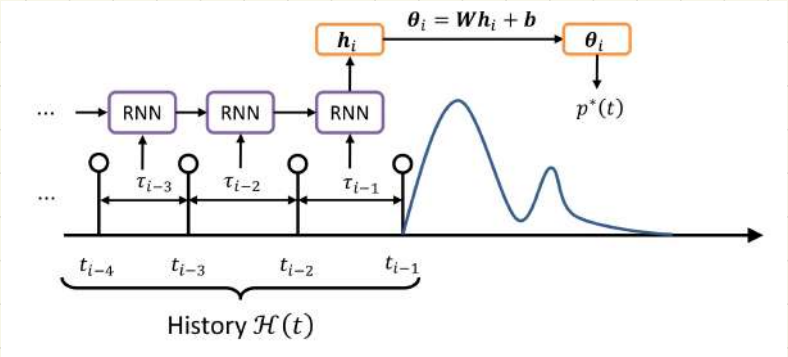
$$\max_\theta \sum_{t \in \{t_1, \ldots t_N\} \in \mathcal{D}_{train}} \log p_\theta(\{t_1, \ldots t_N\})$$

- log likelihood of single sequence

$$\log p_\theta(\{t_1, \ldots t_N\}) = \sum_{i=1}^{N} \log \lambda^*(t_i) - \int_0^T \lambda^*(u)\, du$$

different sequences can have different lengths

Modelling TPPs with RNNs :



directly model the conditional distribution $p^*(t)$ using RNN

1) Every time an event happens, we feed τ_i into RNN
2) hidden state $h_i \in \mathbb{R}^D$ of RNN as history embedding
3) use h_i to generate parameters θ_i of distro $p^*(t)$

$$p^*(t) = p(t \mid \mathcal{H}(t)) = p(t \mid h_i)$$

Modelling $p^*(t)$

Questions:

1. Is it possible to obtain the conditional intensity $\lambda^*(t)$ if you know only the survival function $S^*(t)$ and don't know the conditional PDF $p^*(t)$?

1) $S^*(t) = \exp\left(-\int_{t_{i-1}}^{t} \lambda^*(u)\,du\right)$ ≠

2. Would you use (a) Hawkes process or (b) inhomogeneous Poisson process to model the following event data?
   - Customers visiting a supermarket (event = customer enters the supermarket)
   - Messages sent by a single user on WhatsApp (event = message sent)
   - Taxi rides in a city (event = a trip starts)

2) (i) inhomogeneous point process → History should not typically excite more occurrences

(ii) Hawke's process as one message excites and would increase chances of more messages back and forth

(iii) Taxi rides will be inhomogeneous point process again as does not depend on history.

3. What can you say about a TPP with the following conditional intensity function? What kind of behavior does it model?

$$\lambda^*(t) = \exp\left(t - \sum_{t_i \in \mathcal{H}(t)} 1\right)$$

3) $\exp\left(t - \sum_{t_i \in \mathcal{H}(t)} (1)\right)$

• the intensity decreases as # occurrences increases. This is called inhibiting process.

Problem 1: Consider a temporal point process, where all the inter-event times $\tau_i = t_i - t_{i-1}$ are sampled i.i.d. from the distribution with the survival function

$$S(\tau) = \exp\left(-(e^{b\tau} - 1)\right)$$

with a parameter $b > 0$.

a) Write down the closed-form expression for the conditional intensity function $\lambda^*(t)$ of this TPP. Simplify as far as you can.

b) Write down the closed-form expression for the log-likelihood of a sequence $\{t_1, ..., t_N\}$ generated from this TPP on the interval $[0, T]$. Simplify as far as you can.

1) (a) $S(\tau) = \exp(1 - e^{b\tau})$

$F(\tau) = 1 - \exp(1 - e^{b\tau})$

$p(\tau) = b \exp(1 - e^{b\tau}) \exp(b\tau)$

$p(\tau) = b \exp(1 + b\tau - e^{b\tau})$

$h(\tau) = \frac{p(\tau)}{S(\tau)} = b \frac{\exp(1 - e^{b\tau}) \cdot \exp(b\tau)}{\exp(1 - e^{b\tau})}$

$h(\tau) = b e^{b\tau}$

$\lambda^*(t) = b e^{b(t_i - t_{i-1})}$

(b) $p(\{t_1, ..., t_N\}) = \left(\prod_{i=1}^{N} p^*(t_i)\right) S^*(T)$

$= \prod_{i=1}^{N} \left(b \exp(1 + b\tau - e^{b\tau}) \right).$

(solve ahead) should be doable

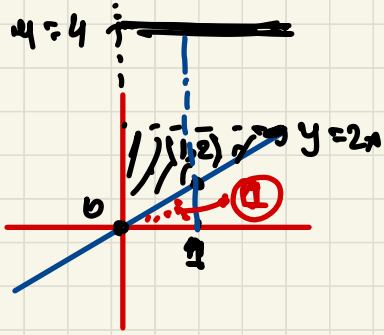
Problem 2: Consider an inhomogeneous Poisson process (IPP) on $[0, 1]$ with the intensity function $\lambda(t) = 2t$. We simulate a sample from this IPP using thinning. For this, we first simulate a *homogeneous* Poisson process (HPP) with intensity $\mu = 4$ and apply the thinning procedure described in the lecture. What is the expected number of events from the HPP that will be rejected when using this procedure?

P2) Inhomogenous Poisson Process

$\lambda(t) = 2t$

approximating using thinning.

approximate $g(t)$



no. of expected events $= \int_0^1 \lambda(t)\,dt$

$= \int_0^1 2t\,dt = \boxed{1}$

for HPP $\int_0^1 \lambda(t)\,dt = \boxed{4}$

on an average 3 events will be discarded

Problem 3: Consider an inhomogeneous Poisson process on $[0, 4]$ with the intensity function $\lambda(t) = \beta t$, where $\beta > 0$ is a parameter that has to be estimated. You have observed a single sequence $\{1, 2.1, 3.3, 3.8\}$ generated from this IPP. What is the maximum likelihood estimate of the parameter β?

P3) $\lambda(t) = \beta t$ $\quad S = \{1, 2.1, 3.3, 3.8\}$

$S(t) = \exp\left(-\int_{t_{i-1}}^{t} \beta u\,du\right)$

$= \exp\left(-\int_0^t \beta u^2\right) = e^{-\beta t^2/2}$

$F^*(t) = 1 - e^{-\beta t^2/2}$ $\quad f^*(t) = \beta e^{-\beta t^2/2}$

$$p(\{t_i\}) = \prod_{i=1}^{4} \lambda^*(t_i) \cdot S(T)$$

$$\log p(\{t_i\}|\beta) = \sum_{i=1}^{4} \log(\lambda^*(t_i)) + \log(S(T))$$

$$= \sum_{i=1}^{4} \left[\log(\beta e^{-\beta t_i^2/2}) \right] + \log(e^{-8\beta})$$

$$\therefore n\log(\beta) - \frac{T^2}{2}\beta$$

$$\frac{n}{\beta} - \frac{T^2}{2} = 0$$

$$\boxed{\beta = \frac{2n}{T^2}} \rightarrow \boxed{\frac{1}{2}}$$

Problem 4: Consider a *neural* temporal point process where the conditional intensity function is defined with a neural network. In particular, for a time point t_i, we represent the history $\{t_1, t_2, \ldots, t_{i-1}\}$ with a fixed-sized vector $\boldsymbol{h}_i \in \mathbb{R}^d$. The conditional intensity function $\lambda^*(t)$ is defined as a function of $\boldsymbol{h}_i$. We will use the transformer architecture (see previous lecture). We propose the following implementation.

Given the full sequence $\{t_1, t_2, \ldots, t_n\}$, we calculate all $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ in parallel. We first calculate vectors $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i \in \mathbb{R}^d$ as a function of t_i. We stack these vectors into matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n \times d}$. The output of the transformer is: $\boldsymbol{H} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T)\boldsymbol{V}$, then $\boldsymbol{h}_i$ is the ith row of $\boldsymbol{H}$.

Identify the errors in this implementation compared to the original definition of $\boldsymbol{h}_i$. Propose a solution.

$\lambda^*(t) = f(h_i)$

Masking

Impose right shift and masking (for future)

maybe get back to this

Chapter 10 - Graphs & Networks :

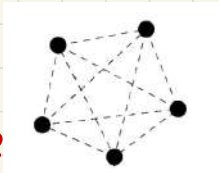
Plain / Simple graph $G = (V, E)$

- set of nodes V
- set of edges $E \subseteq V \times V$

May be represented : using binary adjacency matrix

Are real graphs random?

1) Erdös- Renyi Random Graph Model



- Start with N (isolated) nodes
- for every pair v_1, v_2 add an edge with probability p
- every edge occurs with equal prob.

• Are real graphs random : No!

Power law distribution :

Definition : - two variables are related by the power law when $y(x) = Ax^{-\gamma}$, A & γ (power law constant) are positive constants

- A random variable is distributed according to a power law when pdf is given by :

$$p(x) = Ax^{-\gamma}, \quad \gamma > 1$$

Note : Power law distribution looks like a line on log-log.

Characteristic : decay of polynomial is only polynomial

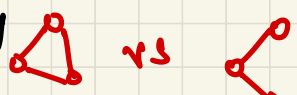
Note: graphs with degree distributions following a power law is called scale free.

Remark: We don't claim that the underlying distro is power law distribution

instead: in many cases a power law distribution is a good description / fit

Gaussian Trap: Be careful while writting algorithm

Patterns and Algorithms: Patterns allow new algos.

- Real social networks have many triangles  vs
- clustering coefficient: $C = \frac{3 * \text{number of } \Delta s}{\text{number of connected triples}}$
- computing clustering coefficient requires computing Δs.
  but, triangles are expensive to compute

Efficient triange counting:

Recap: Eigen decomposition $Ax = \lambda x$

observed: Power law in the eigenvalues & adjacency matrix.

- number of triangles $= \frac{1}{6} \text{trace}(A^3) = \frac{1}{6} \sum \lambda_i^3$

$\lambda_i \rightarrow$ eigenvalue of adjacency matrix

- Eigen values follow power law (highly skewed)

Power Iteration : iterative approach to compute a single eigenvector

v ← arbitrary vector, x ← Matrix

iteratively compute : $v \leftarrow \frac{x \cdot v}{\|x \cdot v\|}$ (until convergence)

* efficient for sparse data

for multiple eigen values : Deflate the matrix

eigen decomposition x : $\Gamma \Lambda \Gamma^T = \sum \lambda_i \gamma_i \gamma_i^T$

$\hat{x} = x - \lambda_1 \gamma_1 \gamma_1^T$

* Real graphs are sparse :

* characteristic path length :
  - for each starting node : shortest path to every other node
  - Take the average length of all these paths
  - consider avg length of all starting nodes and then take the median.

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

* average diameter (mean instead of median in prev)

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

Effective diameter / Eccentricity and Hotplot :

- $N_h(u)$: number of nodes reachable from u via h hops

$$N_h(u) = \{ v \in V \mid len(path(u,v)) \leq h \}$$

- Total neighbourhood size N_h is the sum over all the starting nodes

$$N_h = \sum_{u \in V} |N_h(u)|$$

Hotplot : plot of N_h vs h.

- effective diameter (or eccentricity) :
  - Minimum number of hops in which some fraction of all connected pairs of nodes can reach each other.

$$\min \{ h \in \mathbb{N} \mid N_h \geq 0.9 |V|^2 \}$$

Chapter 10 : Graphs generated models :

Models assuming (conditional) independent edges

1) Erdös-Renyi Graph Model — corresponds to Poisson Distro.

$A_{i,j} \sim Bernoulli(p)$

- Degree distribution :
  - probability of a vertex having degree p is :

$$p_k = \binom{N-1}{k} \cdot p^k \cdot (1-p)^{N-1-k} \approx \frac{z^k e^{-z}}{k!}$$

* $z \rightarrow$ avg degree of graph

But in real world we have power-law distributions

- Diameter : $\log(N) / \log(z)$ — N : avg node degree
  - ↳ Diameter increase slowly with no. of nodes

But in real world it is constant or it shrinks

- Clustering coefficient :
  connection probability $p = \frac{2}{N-1}$

To generalize to real world data

Idea : → nodes of same community connect with prob. p
→ nodes of different community connect with prob. q. $p > q$

Planted Partition Model :

- set of nodes V, partitioned into 2 communities C_1 and C_2
- generate edge b/w every pair of nodes with prob.

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

Stochastic Block Model :

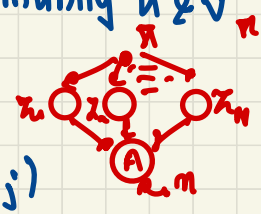
generalizes PPM to graphs with arbitrary numbers and sizes of communities, and varying edge densities

- random variables :
  - $z_i \in \{1, .. K\}$ node i belongs to block / community z_i
  - $A \in \{0,1\}^{N \times N}$: adjacency matrix.

- Model Params :
  - $\pi = [\pi_1, .. \pi_K]$: community parameters
  - η_{uv} : edge probabilities b/w two nodes that are in community u & v

$\pi = [\pi_1, .. \pi_K]$

- conditional distributions :
  - $Pr(z_i = k) = \pi_k$
  - $Pr(A_{ij} \mid z_i, z_j) = \text{Bernoulli}(\eta_{z_i z_j})$



$$\eta = \begin{bmatrix} \eta_{11} \dots \eta_{1K} \\ \eta_{K1} \dots \eta_{KK} \end{bmatrix}$$

Preferential Attachment Model :

- ER / PPM / SBM assume edges are generated independently. nodes at beginning ; edges Bernoulli distro.

Now generate network based on two processes:

(i) growth : instead of starting with all nodes, start with a small set of nodes and let the networks grow over time by adding new nodes and edges

(ii) preffemential treatment : "rich get richer" probability of connecting nodes is proportional to the current degree of the nodes.

Note: "Rich get richer" principle leads to power law

Intial attractiveness is an extension of power law degree distribution.

algorithm:

1) start with m_0 nodes
2) add new node w
3) simultaneously insert m directed edges (u, v)
   probability that the endpoint of an edge (u, v) corresponds to $A_v = A + indeg(v)$.
   A : initial attractiveness
   indeg(v) is the number of currently !) incoming edges.
4) goto 2 until required number of nodes are obtained

Initial attractiveness : Properties :

Params: m (new edges per step) A (initial attractiveness)

Degree distribution : follows a power law with exponent $\gamma = 2 + A/m$

↳ Matches real world data

Diameter : Diameter grow $\sim \left(\frac{\log N}{\log \log N} \right)$

- Matches small world effect
- But : still slightly increasing in contrast to real world data.

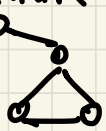
Average degree : Remain constant over time

Configuration Models :

generating networks with arbitrary specified degree distribution.

1) assign degree to each nodes, represented as stubs or half-links
2) Randomly select a stub pair and connect them
   - Depending on the order and way in which the stubs were chose, we obtain different

Preserves degree structure



Deep Generative Networks :

all classical approaches → "hand crafted features"
need something to learn from data.

NetGAN : First GAN for networks.

- How to adopt GAN ideas to graph?

Naive $O(N^2)$ solution : learning a distribution over adjacency matrix. Not permutation invariant

solution: → Learn distribution over Random walks

Random Walk :

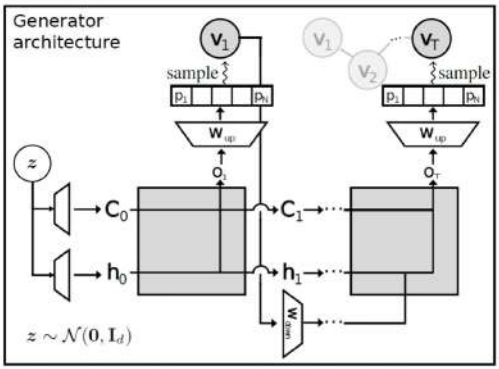
- Graph $G = (V, E)$, walk length t
- Define $W_{v_i} = (W_j : 0 \leq j < t) \; ; \; W_0 = v_i)$ a random walk with starting node v_i
- W_j is the j-th node in the random walk, and the nodes are sampled from

$$Pr(W_{j+1} = v_u \mid W_j) = \begin{cases} 1/d_{W_j} & \text{if } v_u \in N(W_j) \\ 0, & \text{else} \end{cases} \qquad \forall j : 0 < j < t$$

- $N(v_x)$ is the set of neighbours of node v_x & d_{v_x} its degree

NetGAN : generator :

$$\begin{aligned} & & z &\sim \mathcal{N}(0, I_d) \\ & & m_0 &= g_{\theta'}(z) \\ v_1 &\sim \mathrm{Cat}(\sigma(p_1)), & (p_1, m_1) &= f_\theta(m_0, 0) \\ v_2 &\sim \mathrm{Cat}(\sigma(p_2)), & (p_2, m_2) &= f_\theta(m_1, v_1) \\ & \vdots & & \vdots \\ v_T &\sim \mathrm{Cat}(\sigma(p_T)), & (p_T, m_T) &= f_\theta(m_{T-1}, v_{T-1}) \end{aligned}$$



generative process is as follows

1) sample latent noise from normal distribution $z \sim N(0, I_d)$
2) Pass z through NN to obtain starting states c_0 and h_0 for LSTMs
3) Run the LSTM from this state to generate a sequence of nodes (random walk)

Problem: Categorical distribution is not differentiable with respect to parameter π.
- same as variational inference.

We need → a) Discrete samples to feed into discriminator
→ b) gradients to train generators

Solution: Categorical reparameterization trick

- random variable g is said to have standard Gumbel distribution if
  $g = -\log(-\log(u))$ with $u \sim \text{Uniform}[0,1]$

- Let v be discrete random variable $P(v = k) \propto \pi_k$ and let $\{g_k\}_{1 \le k \le K}$ be i.i.d sequence from Gumbel random variables.

gumbel trick

$$v = \arg\max_k [g_k + \log \pi_k]$$

- recipe for (reparameterization) sampling from categorical distribution.

1) Draw Gumbel noise by transforming uniform samples
2) add it to $\log \pi_k$ which only has to be known up to a normalizing constant.
3) Take the value k that produces the maximum

Problem: argmax in $v = \arg\max_k [g_k + \log \pi_k]$ is still not differentiable.

Solution: Approximate it with softmax + temperature parameter τ.

so $v^* = \sigma\left(\frac{\log \pi_k + g}{\tau}\right)$ → Note: this is only for backward pass; forward pass is $\arg\max(v)$

as $\tau \to 0$, $v^* \to v$

Questions :

- How much memory do you need to store the edges of a graph with 1000 nodes and 10,000 edges in a dense adjacency matrix? How much for a sparse matrix?

in dense → $|V| \times |V| = 1000 \times 1000 = 10^6$

4 Bytes / memory ≈ 4MB

sparse

1 entry = 2 pointers → (10^4 edges * 16)

8 Byte per pointer ≈ 0.16 MB.

- What is the average degree in an Erdös-Renyi graph with edge probability p? And in a real world sparse graph with $O(E) = O(N^{\alpha})$?

each vertex can have upto (N-1) edges each with a probability p.

Avg degree : $(N-1)p$

In real world : $O(E) = O(N^{\alpha})$

no. of vertex per edge = 2

$\frac{2 * N^{\alpha}}{2} = 2N^{\alpha-1}$

- Can the Erdös-Renyi model generate all graphs that the Stochastic Block Model can generate?

yes as all the edges have an equal probability p.

there is a possibility that ER model generates all possible combinations in SBM model.

- Is the Initial Attractiveness model equal to the Erdös-Renyi model as $A \to \infty$?

Doubt

- Why does π not need to be normalized for the Gumbel trick to work?

let $\hat{\pi} = c \cdot \pi$ $\quad \log(\hat{\pi}) = \log(c) + \log(\pi)$

argmax is insensitive to constant additions

Problem 1: An unweighted undirected graph represented by an adjacency matrix $A \in \{0,1\}^{N \times N}$ is given. Prove that the number of triangles in the graph is equal to $\frac{1}{6}\text{trace}(A^3)$ and that this term is in turn equal to $\frac{1}{6}\sum_i \lambda_i^3$ where λ_i are the eigenvalues of the adjacency matrix A. *Hint:* Show first that A^k_{ij} is the number of walks of length k from node i to node j.

P1) base case $A^0 = I_N$ is true

$$A_{ij}^{k} = \sum_{v=1}^{N} A_{iv} A_{vj}^{k-1} = \sum_{v=1}^{N} A_{vj}^{k-1} \cdot \begin{cases} 1 & \text{if } \exists \text{ edge b/w } i \& v \\ 0 & \text{otherwise} \end{cases}$$

for A_{ii}^3 ; number of walks of length 2 from node i to i.

in undirected graph this walk is a triangle

so each triangle is counted 3 * 2 *

3 nodes, 2 directions

$$\frac{1}{6}\sum_i A_{ii}^3 = \frac{1}{6}\text{trace}(A^3)$$ is number of triangles

$\text{trace}(A) = \sum_i \lambda_i$ (sum of eigenvalues)

if v is e.v of A then for A^3

$A^3 v = \lambda^3 v$ (v is eigenvector with e.v λ^3)

$\sum \lambda_i^3$

Problem 2: Given is an Erdös-Renyi graph consisting of N nodes, with the edge probability $p \in [0, 1]$. Derive the probability p_k that a node in the graph has degree equal to exactly k.

$d \sim$ Binomial $(N-1, p)$

$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k}$

Problem 3: Given is an Erdös-Renyi graph consisting of N nodes with edge probability $p \in [0, 1]$. What is the expected number of triangles in this graph?

total number of possible triangles $\binom{N}{3}$
we will this set T
let $X_t = 1$ if triplet $t \in T$
else 0
$X_t \sim$ Bernoulli with prob (p^3)

$$\mathbb{E}\left[\sum_{(i,j,k) \in T} X_{(i,j,k)}\right] = \sum_{(i,j,k) \in \text{set}} \mathbb{E}[X_{i,j,k}] = \sum_{(i,j,k) \in T} p^3 = \binom{N}{3} p^3$$

Problem 5: Compare the two following graph generation processes.

- Graph G_1 is generated by a stochastic block model. It consists of N nodes partitioned into $K = 2$ communities. Both communities consist of exactly $N/2$ nodes, and $\eta = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$.
- Graph G_2 is an Erdös-Renyi graph of N nodes and edge probability p.

Given the probabilities a and b, for which values of p will the expected number of triangles in G_2 be *larger* than the expected number of triangles in G_1?

$2\left[\binom{N/2}{3} a^3 + \binom{N/2}{2}\binom{N/2}{1} ab^2\right]$

$G_2 \rightarrow {}^{N}C_3\, p^3$ (no. of Δ^s)

SBM → N nodes $K = 2$ partitions $\eta = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$

$\binom{N}{3} p^3 > 2\left[\binom{N/2}{3} a^3 + \binom{N/2}{2}\binom{N/2}{1} ab^2\right]$

Discuss tomorrow.

Problem 4: Given are 6 graphs $\{G_1, \ldots, G_6\}$, which exhibit the properties listed in Table 1. Five of them have been synthetically generated, while one is a real graph. Assign the graphs $\{G_1, \ldots, G_6\}$ to the following models (one each) and briefly justify each answer!

a) Erdös-Renyi model

b) Stochastic block model with 5 clusters

c) Stochastic block model with 10 clusters

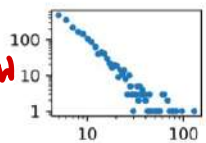
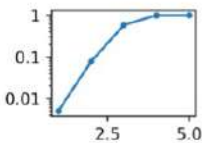
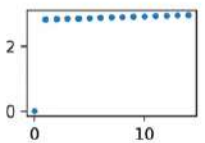
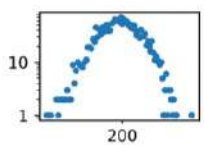
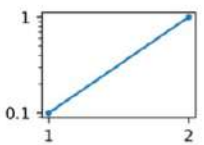
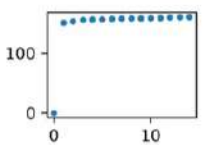
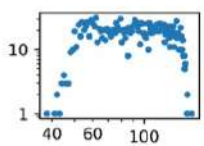
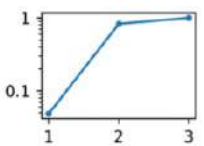
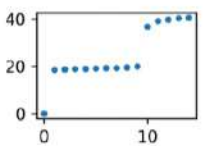
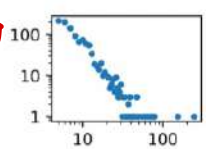
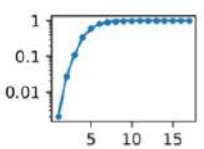
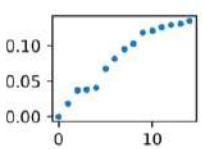
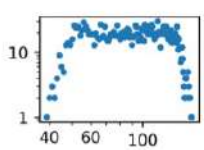
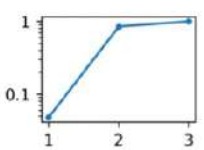
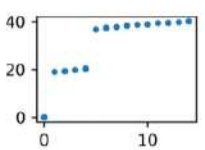
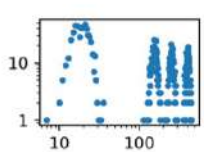
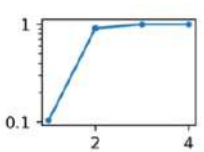
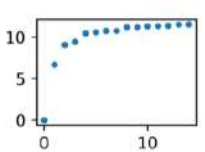
d) Stochastic block model with core-periphery structure

e) Initial attractiveness model

f) Real graph

these two follow power law G_1 or G_4

Table 1: Graphs $\{G_1, \ldots, G_6\}$

ID	Degree distribution	Hop plot	Smallest eigenvalues	Clustering coeff.
G_1				0.013
G_2				0.100
G_3				0.275
G_4				0.278
G_5				0.145
G_6				0.191

Chapter 11 : Graph Clustering :

input graph: $G = (V, E)$
aim : Find clusters of vertices in the graph.

clustering in network data :

Basic aim →
- vertices in the same clusters must be connected by many edges
- only few edges b/w different clusters

Two main clustering algorithms are.
- Partition approaches: each vertex to exactly one cluster
- Non Partition approaches: clusters can overlap, outliers that belong to no cluster possible.

Partitioning Approaches :

idea : constrained optimization problem
given $G = (V, E)$ partition vertex set into clusters $C = \{C_1, .. C_k\}$ s.t
- given objective function $\phi(C)$ is optimized
- subject to constraints
  - $C_1 \cup C_2 ... \cup C_k = V$ (Every vertex from V belongs to a cluster)
  - $\forall 1 \le i < j \le k : C_i \cap C_j = \emptyset$ (clusters are disjoint; no overlap)
- usually NP-hard → discrete optimization
- eg- k Means : objects are partitioned such that Total distance is maximized.

Global Minimum Cut :

First idea : minimize number of edges/weights b/w clusters
- small value of cut

Task : Partition V into two sets C_1 and C_2, such that the sum of inter cluster edge weights
$cut(C_1, C_2) = \sum_{v_1 \in C_1, v_2 \in C_2} w(v_1, v_2)$ is minimized

here : undirected edges, i.e $w(a,b) = w(b,a)$

Note : s-t cut (where source and sink are given) can be done in **Ford Fulkerson**. (Polynomial time)

However in graph clustering no source/sink is given ∴ any partitioning is possible

Normalized cut criteria :

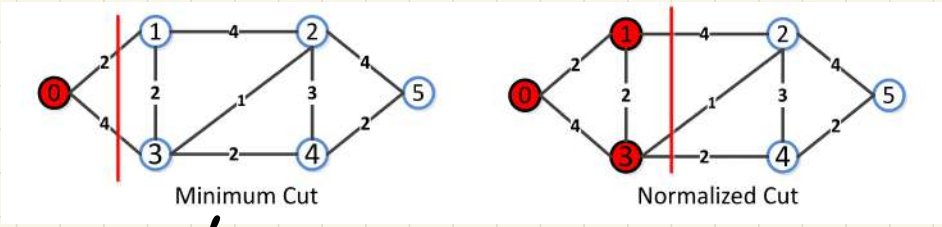
Drawbacks of Minimum Cut
- → cut small vertex set
- → only inter cluster edges are considered.

∴ normalized cut criterion were introduced

- Ratio cut : minimize $\frac{cut(C_1, C_2)}{|C_1|} + \frac{cut(C_2, C_1)}{|C_2|}$ (* no. of nodes)
- Normalized cut : minimize $\frac{cut(C_1, C_2)}{vol(C_1)} + \frac{cut(C_1, C_2)}{vol(C_2)}$

Note : $vol(C_i) = assoc(C_i, V) = cut(C_i, V) =$

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



only minimize inter cluster interactions
hence produces a not so useful cut as small clusters would form

* $\frac{cut(C_1, C_2)}{|C_1|} + \frac{cut(C_2, C_1)}{|C_2|}$

Multiway Graph Partitioning :

- generalization to $k \geq 2$ clusters.
- Partition V into $C_1, .. C_k$ such that:
  - cut : $\min_{C_1, .. C_k} \sum_{i=1}^{k} cut(C_i, V \setminus C_i)$
  - Ratio cut : $\min_{C_1, .. C_k} \sum_{i=1}^{k} \frac{cut(C_i, V \setminus C_i)}{|C_i|}$
  - Normalized cut : $\min_{C_1, .. C_k} \sum_{i=1}^{k} \frac{cut(C_i, V \setminus C_i)}{vol(C_i)}$

Optimal solution is N-P Hard.

Graph Laplacian :

Definition : Laplacian matrix $L = D - W$
- W = (weighted) adjacency matrix, D = degree matrix

- i.e $L_{uv} = \begin{cases} -w_{uv} & \text{if } (u,v) \in E \\ deg(v) & \text{if } u = v. \\ 0, & \text{otherwise} \end{cases}$

- observation 1: For any vector f we have

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

Graph Laplacian:

- discrete analogous of the Laplacian $\sum \frac{\partial^2 f}{\partial x_i^2}$ measures extent a function differs at a point from its value at nearby points
- (discrete) Laplacian: operator on n-dim vector space of functions $f: V \to \mathbb{R}$
- Laplacian transforms f to another function g

$$g(v) = (Af)(v) = \sum_{(u,v) \in E} w_{uv} \cdot [f(v) - f(u)]$$

- L is symmetric and positive semi-definite (as D & W are symmetric) L is PSD (quadratic)
- smallest eigen value of L is 0. Corresponds to $\vec{1}$
- L has n - non negative, real valued eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

- in general symmetric matrices has real-valued eigenvalues
- $\lambda_1 = 0$ is smallest ent rest are larger)

- Laplacian is additive $L_{G \cup H} = L_G + L_H$ (G and H are 2 graphs on the same vertex set with disjoint edge sets)

- Laplacian of a graph is sum of Laplacians for each edge
  - $L_G = \sum_{(u,v) \in E} L(u,v)$
  - allows us to prove properties for single edge and add them up.

Laplacian for an edge i.e graph with single edge (u,v)

$$L(u,v) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Observation: No. of eigenvectors of L with eigenvalue 0 corresponds to the number of connected components

* corrollary: if the graph is connected : $\lambda_2(L) > 0$

Properties of the Graph Laplacian:

Algebraic connectivity of a graph is $\lambda_2(L)$
- also known as Fiedler eigenvalue.
- magnitude represents how well connected graph is
- $\lambda_2(K_n) = n$, where K_n is complete graph with n nodes

- for every $S \subset V$ we have

$$\theta(S) = \frac{cut(S, \bar{S})}{|S|} \geq \lambda_2 \left(1 - \frac{|S|}{|V|}\right)$$

  - $\theta(S)$ is isoperimetric ratio of S
  - $\theta_G \overset{def}{=} \min_{|S| \leq \frac{|V|}{2}} \theta(S) \geq \lambda_2/2$
  - θ_G is called the Cheeger constant of graph or conductance of a graph

- Inequality implies if λ_2 is big, graph is well connected boundry of each small set of vertices is at least λ_2 times a value close to 1.

conclusion: spectrum of L encodes useful info in graph

unfortunately: co-spectral graphs exist
- graphs that are not isomorphic but share same spectrum
- implies that the spectrum doesn't completely characterize the graph.

so spectrality can be useful: find sparse graph with identical the same spectrum.

Graph Laplacian and the minimum cut:

- for $k=2$ clusters let $f \in \{-1, 1\}^n$ be indicator vector,

$$f_{c_1}[i] = \begin{cases} +1 & \text{if } v_i \in C_1 \\ -1 & \text{if } v_i \in V \setminus C_1 = C_2 \end{cases}$$

- Then $f_{c_1}^T L f_{c_1} = 4 \cdot \text{cut}(C_1, C_2)$
- Thus we can minimize $f_{c_1}^T L f_{c_1}$ to minimize the cut.

But min cut is not ideal so we expand this to ratio cut.

ratio-cut ($k=2$): $\min_{C_1 \subset V} \frac{\text{cut}(C_1, C_2)}{|C_1|} + \frac{\text{cut}(C_2, C_1)}{|C_2|}$

where $C_2 = V \setminus C_1 = \bar{C_1}$

indicator vectors

$$f_{c_1}[i] = \begin{cases} +\sqrt{\frac{|\bar{C_1}|}{|C_1|}} & \text{if } v_i \in C_1 \\ -\sqrt{\frac{|C_1|}{|\bar{C_1}|}} & \text{else} \end{cases}$$

1. $\sum_i f_{c_1}[i] = 0$ — f_{c_1} is orthogonal to $\vec{1}$.

2. $f_{c_1}^T \cdot f_{c_1} = \|f_{c_1}\|^2 = |V|$ — length is constant

3. $f_{c_1}^T \cdot L \cdot f_{c_1} = \dots = |V| \cdot \left[\frac{cut(c_1, c_2)}{|c_1|} + \frac{cut(c_2, c_1)}{|c_1|} \right]$

→ $|V| \cdot$ ratio_cut.

Minimizing the ratio cut is equivalent to:

$\min_{c_1 \subset V} \{ f_{c_1}^T \cdot L \cdot f_{c_1} \}$ subject to $f_{c_1} \perp \vec{1}$ and $\|f_{c_1}\|_2 = \sqrt{|V|}$

and f_{c_1} is defined as before

Discrete optim → NP Hard

idea: constraint relaxation

drop the discretness condition (i.e f_{c_1} can take any values).

- Result: $\min \{ f_{c_1}^T \cdot L \cdot f_{c_1} \}$ subject to $f_{c_1} \perp \vec{1}$ and $\|f_{c_1}\| = \sqrt{|V|}$

Relaxed constraint — $f_{c_1} \in \mathbb{R}^{|V|}$ in real space

- For symmetric matrix L we have:

$\lambda_1 = \min_{\|x_1\|=1} x_1^T L_1 x_1$ $\quad \lambda_2 = \min_{\substack{\|x_2\|=1 \\ x_2 \perp x_1}} x_2^T L x_2 \dots$

- f_{c_1} is the second smallest eigenvector of L
  - $\vec{1}$ is smallest
  - $L \cdot f_{c_1} = f_{c_1} \cdot \lambda_2 \Rightarrow f_{c_1}^T \cdot L \cdot f_{c_1} = |V| \lambda_2$

in case of 2 clusters: sign of f_{c_1} gives clusters

general case (spectral clustering):

↳ ($K > 2$) : clusters $C_1, C_2, \ldots$

$$h_c[i] = \begin{cases} 1/\sqrt{|C|} & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$

- let $H = [h_{C_1}; h_{C_2} \cdots; h_{C_K}]$

- observations:

  - $H^T H = Id$
  - $h_{C_i}^T . L . h_{C_i} = \dfrac{cut(C_i, V \setminus C_i)}{|C_i|}$ and $h_{C_i}^T . L . h_{C_i} = (H^T L H)_{ii}$

Ratio cut $(C_1, \ldots C_K) = \sum_{i=1}^{K} \dfrac{cut(C_i, V \setminus C_i)}{|C_i|} = \sum_{i=1}^{K} (H^T L H)_{ii}$

$= trace(H^T L H)$

- Minimize ratio-cut

$$\min_{C_1, \ldots C_K} trace(H^T L H) \quad s.t \quad H^T H = Id$$

constraint : allow arbitrary value for H

Result: $\min_{H \in \mathbb{R}^{V \times K}} trace(H^T L H)$ s.t. $H^T H = Id$

- To find clusters based on eigen value
→ Represent vertex by a vector of its corresponding components (spectral embeddings)
- clustering in embedding space yields final result.

Also known as spectral layout.

Remarks:

- spectral clustering using unnormalized Laplacian:
$L = D - W$
→ approx. of ratio cut
- spectral clustering using normalized Laplacian
$L_{sym} = D^{-1/2} L D^{1/2}$ → approx. of normalized cut.

Probabilistic Approaches:

- consider graph as realisation (sample) drawn from a generative model
- generative process is controlled by set of params
- communities are modelled with generative process.

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

- likelihood of a community assignment

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

- assume p & q are known.

simplest soln (MLE) $z^* = \arg\max_{z \in \{1,\dots,k\}^N} Pr(A|z)$

PPM and Spectral Clustering:

ML estimate for z for planted partition model?
let us assume that communities are balanced $(C_1 = C_2 = N/2)$

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

likelihood: $Pr(A|z) \propto \left(\frac{p(1-q)}{q(1-p)}\right)^{E_{cut}(z)}$

since $p > q$, MLE is equivalent to finding a minimum balanced cut

How can we solve? Spectral clustering

② SBM generalizes PPM with arbitrary number and size of communities

r.v → $z_i \in \{1,\dots,k\}$ node i belongs to block z_i
→ $A \in \{0,1\}^{n \times n}$, adjacency matrix

Model Params: → $\pi = [\pi_1, \dots \pi_k]$: community params
→ η_{uv} : edge prob. b/w two nodes.

conditional distn: $Pr(z_i = k) = \pi_k$

$Pr(A_{ij}|z_i, z_j) = \text{Bernoulli}(\eta_{z_i z_j})$

inference

- if η and π are given

$$Pr(z|A,\eta,\pi) = \frac{Pr(A|z,\eta,\pi)\,Pr(z|\pi)}{Pr(A|\eta,\pi)}$$

intractable $O(K^N)$ ops.

use variational inference:

find $q(z)$ similar to true Posterior $Pr(z|A,\eta,\pi)$ (mean field assumption)

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

- if both z and A are observed

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

$$\eta_{uv}^{MLE} = \frac{\text{observe \# edge btw } u \text{ and } v}{\text{possible \# edge btw } u \text{ and } v} = \frac{\sum_{i,j:i<j} \mathbb{I}(z_i=u)\mathbb{I}(z_j=v)}{P_{uv}}$$

$$P_{uv} = \begin{cases} \binom{N_u}{2} & u=v \\ N_u \cdot N_v & u \neq v \end{cases}$$

no. possible edges

- if only A is observed:

use ELBO wrt $q(z)$ as well as η and π

Questions

- How can you find the connected components of a graph from its Laplacian?

# of eigenvalues which have value = 0 are the number of connected components

- Consider a graph with n arbitrarily connected nodes and k disconnected nodes. What are the first $k+1$ clusters that spectral clustering finds? Why?

spectral clustering would return the k solo nodes and the connected graph.

Problem 2: Consider minizing the ratio cut on a graph with two clusters C_1 and C_2 and N nodes in total. The indicator vector

$$f_{C_1,i} = \begin{cases} +\sqrt{\frac{|\overline{C_1}|}{|C_1|}} & \text{if } v_i \in C_1 \\ -\sqrt{\frac{|C_1|}{|\overline{C_1}|}} & \text{otherwise} \end{cases}$$

is defined as in the lecture. Prove the following three properties about $\boldsymbol{f}_{C_1}$.

a) $1^T \boldsymbol{f}_{C_1} = \sum_i f_{C_1,i} = 0$

b) $\boldsymbol{f}_{C_1}^T \boldsymbol{f}_{C_1} = \|\boldsymbol{f}_{C_1}\|_2^2 = |V|$

c) $\boldsymbol{f}_{C_1}^T L \boldsymbol{f}_{C_1} = |V| \left[\frac{\text{cut}(C_1,C_2)}{|C_1|} + \frac{\text{cut}(C_1,C_2)}{|\overline{C_1}|} \right]$

2) a) $1^T f_{C_1} = \sum_i f_{C,i} = \sqrt{M}\sqrt{\frac{N-M}{M}} - \sqrt{N-M}\sqrt{\frac{M}{N-M}}$

$= 0$

b) $f_{C_1}^T f_{C_1} = \|f_{C_1}\|_2^2 = \sum_i f_{C_1,i}^2$

$= \sum_i \begin{cases} \sqrt{\frac{|\overline{C_1}|}{|C_1|}} & v_i \in C_1 \\ \sqrt{\frac{|C_1|}{|\overline{C_1}|}} & \text{other.} \end{cases}$

$M \cdot \frac{N-M}{M} + (N-M)\frac{M}{N-M}$

$= N = |V|$

c) $f_{c_1}^T L f_{c_1} = \frac{1}{2} \sum_{(u,v) \in E} w_{uv} (f_{c_1,u} - f_{c_1,v})^2$

Ch-12 - Graph Node Embeddings :

Learning Node Representations :

- transform the graph s.t each vertex is represented by a vector.

$\Phi : V \rightarrow \mathbb{R}^d$ maps each node to $\mathbb{R}^d$

- Nodes close in the embedding space are "similar" w.r.t. graph structure.

Types of embeddings :

- difference come from how we define similarity b/w nodes in embedding space.

spectral clustering : (Recap)

- construct the graph Laplacian L
- compute first k eigenvectors v_i of L in cols.
- Represent the i-th node as i^{th} row of H
- cluster the vector representation, for ex. k means

Spectral clustering : embedding view

- Spectral clustering is based on eigenvectors of the graph Laplacian L.
  - L encodes structural behaviour of graph G
  - $|V| \times |V|$ adjacency matrix is transformed and reduce to $|V| \times k$ matrix H.

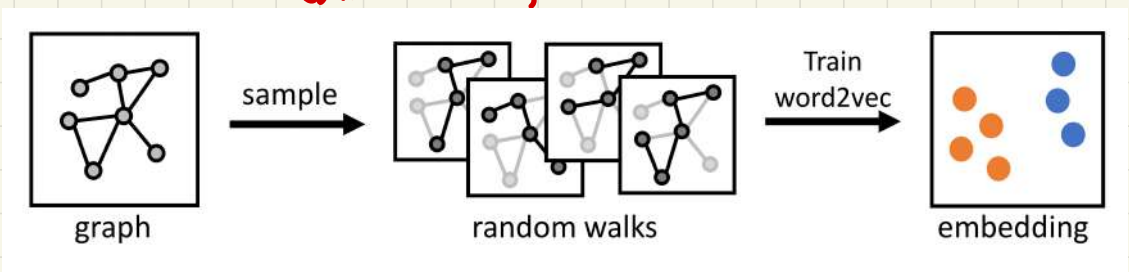
- Relation to PCA and dimensionality reduction
  - transformation is based on eigenvectors of data-matrix and one retains only eigenvectors with smallest / largest eigenvalue.

12 "Deep" Node embedding approaches :

goal : Try to capture more complex structures than spectral clustering.
- thus, hopefully getting better results for specific downstream tasks.

DeepWalk :

idea : transform graph into a set of random walks and learn a word2vec model.
- For every node v_i sample multiple random walks
- Random walks corresponds to sentences in word2vec both are sequences of discrete tokens
- Train this collection of sentences



Results: nodes that are close to each other in graph and share many neighbours get similar vector representations (embeddings)

Graph2Gauss

idea : map each node to gaussian distro., in the embedding space such that a node's 1st neighbour are closer than its 2nd neighbours and so forth.

- learn a mapping : $f_\theta(v) \rightarrow \mathcal{N}(\mu_\theta(v), \Sigma_\theta(v))$

- for each node u define a loss $\sum_{(u,v)} E_{uv}^2 + e^{-E_{uv}}$

where $E_{uv} = \kappa_L(f_\theta(u), f_\theta(v))$

u and v are all node pairs such that v is closer to u than v (ranking loss)

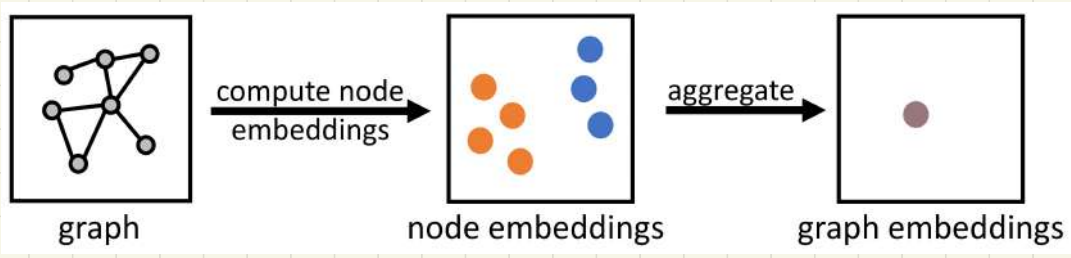
Result: distance in embedding space correlates with distances in the graph and the Gaussian variances express how certain the model is about the embedding.

Graph Embedding (vs Node embedding):

idea: Leverage node embeddings to embed graphs

Problem: → graphs have different number of nodes
simple concatenation produces incompatible dimensionalities

solution: Aggregate all node embeddings into one graph embedding, for example with mean pooling



#12 : Graph Ranking

core idea : A page is important if many important pages point to it.

"voting" principle :
- each page votes for importance of pages it points to
- link's vote α importance of source page
- if page j, with importance r_j has n-outclicks, each get r_j/n votes
- page j's own importance is sum of votes on in-links.

rank of page j : $r_j = \sum_{i \to j} \frac{r_i}{d_i}$ $d_i \to$ out degrees

PageRank : Matrix Formulation :

Stochastic adjacency matrix : M

- if $i \to j$, then $M_{ji} = 1/d_i$ else $M_{ji} = 0$
- M : column stochastic matrix

rank vector : r
- r_i is the importance score of page i
- $\sum_i r_i = 1$

Equation $r_j = \sum_{i \to j} \frac{r_i}{d_i}$ can be written as

$$r = M \cdot r$$

Notes: 1) r is eigenvector of M
- eigenvalue 1
- largest eigen value is 1 (M is column stochastic)

2) Finding x = finding eigenvector of M corresponding to largest eigenvalue.

Computation

Power Iteration: • $x \leftarrow \frac{M \cdot x}{\|M \cdot x\|}$ until convergence.

• required for page rank: $\sum_i x_i = 1$

• let $y = Mx$ $\sum x_i = 1$ • M is column stochastic $\sum y_i = 1$

• no need to normalize

iterate $x \leftarrow Mr$

• start with random (normalized vector) x

Iterate till convergence. $r_j \leftarrow \sum_{i \to j} x_i / d_i$

Random walk interpretation:

surfer's path $\{x_1, x_2, \ldots\} \rightarrow$ Markov chain

$Pr(x_1 = i) = \pi_i$ $Pr(x_{t+1} = j \mid x_t = i) = M_{ji}$

transition probability matrix of Markov chain is $B = M^T$

under some technical conditions:

rank score of page i = $x_i = \lim_{t \to \infty} Pr(x_t = i)$

or in vector form: $r = \lim_{t \to \infty} \pi(t)$

$\pi(t) = \pi B^{t-1}$ = limit of sequence $\pi B, (\pi B)B, \ldots$

- $\lim_{t\to\infty} \pi(t)$ is called limiting distribution

- $t \to \infty$, B^t converges to a matrix whose rows are the same

one row of $\lim_{t\to\infty} B^t$ → limiting distr

probability of reaching a node does not depend on starting point.

$$\lim_{t\to\infty} B^{t+1} = \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \end{bmatrix} \Rightarrow \lim_{t\to\infty} \pi(t) = \lim_{t\to\infty} \pi B^{t+1} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \end{bmatrix}$$

$$= \boxed{[a \; b \; c]}$$

Stationary distribution: $\pi(\infty)$ is called stationary distribution

$$\pi(\infty) = \pi(\infty) B$$

* $\pi(\infty)$ if exists = transpose of rank vector 1.
$\pi(\infty)$ can be computed by
1) getting eigenvector associated with $\lambda = 1$
2) normalizing it to 1.

under "technical conditions": Markov chain has a limiting distribution = unique stationary distribution

technical conditions: irreducible & aperiodic

possible to get any state from any state

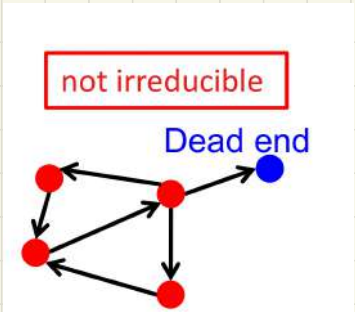
$\exists n$ s.t. $\forall n' \geq n$

$P(X_{n'} = i \mid X_1 = j) > 0$

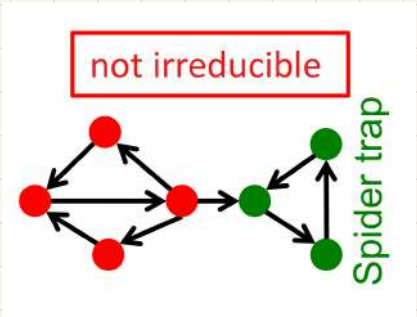
* we might have to prove this

- Markov chain is aperiodic if every state is aperiod.
- irreducible markov chain only need one aperiodic state to imply all states are aperiodic.

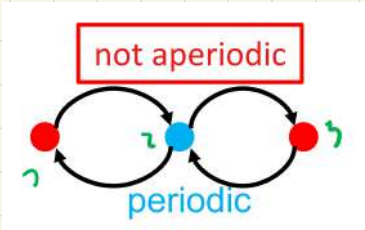
Problems in Page Rank



random walk has nowhere to go
no outlinks



- random walk stuck
- eventually absorbs all importance
spide trap



if we start at state, we will return to the state in fixed periods

solution : random teleportation

two options
- β follow a link at random
- $1-\beta$ jump to some random page

Page rank equation :

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \qquad // \; : \sum_{i \to j} \beta \frac{r_i}{d_i} + \sum_i (1-\beta)\frac{r_i}{N}$$

in matrix notation

$$A = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

$[\frac{1}{N}]_{N \times N}$: $(N \times N)$ matrix with all entries $\frac{1}{N}$

final solution : $r = Ar$

Notes on computation

- attention : the matrix A is dense! (N^2 non zero entri)
  - you should never compute it in such way

- consider the teleport by adding constant penalty to each term

▸ iterate $r_j \leftarrow \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$ until convergence

only neighbours need to be considered

- To maintain sparsity in matrix form multiply βM then add a vector

$$r = \beta M r + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

- vertex oriented computations
  - each vertex performs local computation.

crucial aspects of vertex oriented programming

GAS principle - gather (a), apply (b), scatter (c)

a) gathers info from the adjacent vertex / edge
b) apply transforms
c) scatters info to adjacent vertices

Page rank only requires a) and b)

Problems related to page rank :

- Measure generic popularity of a page
  - Biased against topic-specific authorities
  solution : topic-sensitive Page Rank

- susceptible to link spam
  - artificial link topographies created in order to boost Pagerank.
  solution : use trust rank

- use single measure of importance ✓ sol : Hubs and Authorities

Topic Sensitive Page Rank :

- goal: Evaluate pages based not just on their popularity, but by how close they are to a particular topic.
- allow search queries to be answered based on the interest of the user

- core idea: Bias the random walk
  when the walker teleports → pick page from S.
  - standard page rank: S = all pages
    (all pages with equal probability)
  - Topic sensitive page Rank: S = set of relevant pages

  for each teleport get a different vector r_S.

- topic sensitive page rank takes the following:

$$r = \beta M r + (1-\beta)\pi \qquad \pi_i = \begin{cases} \frac{1}{|S|} & i \in S \\ 0 & \text{oth.} \end{cases}$$

- generalizing to arbitrary teleports

$$r = \beta M r + (1-\beta)\pi \qquad \sum_i \pi_i = 1$$

exact solution is $r = (1-\beta)(I-\beta M)^{-1}\pi$

Discovering relevent set:
→ create page ranks of different topics
→ use context of the query

- **"Normal" PageRank:**
  - Teleports uniformly at random to any node
  - All nodes have the same teleport probability of surfer landing there:
    $\pi = (0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1)^T$
- **Topic-Sensitive PageRank**:
  - Teleports to a topic specific set of pages
  - Nodes can have different probabilities of surfer landing there:
    $\pi = (0.1 \quad 0 \quad 0 \quad 0.2 \quad 0 \quad 0.5 \quad 0 \quad 0 \quad 0 \quad 0.2)^T$
- **Personalized PageRank (Random Walk with Restarts):**
  - Teleport is always to the same node:
    $\pi = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$

Questions :

Ch-13 : Semi supervised Learning

Collective classification :

- Labels are known for some labeled nodes
- goal is to classify the unlabeled nodes

standard assumption : Homophily aka assortativity

Label propagation :

- consider the binary case
- Formal definition / seeds of labelled instance
  - Nodes $V = S \cup U$ → unlabeled instance
  - symmetric weighted edge matrix $W \in \mathbb{R}^{|V| \times |V|}$
  - $\hat{y}_i \in \{0, 1\}$ for $i \in S$ // given
  - $y_i \in \{0, 1\}$ for $i \in V$ // predict

idea : Energy minimization $E(y) = \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2$

- Smoothness : adjacent nodes should have same class label

Energy minimization as MAP :

- goal : $\min_y E(y) = \min_y \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2$
- this is equivalent to optimization problem

$$\arg\min_{y \in \{0,1\}^N} E(y) = \arg\max_{y \in \{0,1\}^N} \exp(-E(y))$$

- consider the following approximation to posterior

$$p(y_U \mid W, y_L) = \frac{1}{Z} \exp(-E(y))$$

$Z = \sum_y \exp(-E(y))$ → normalizing const

Label Propagation

Two aspects: smoothness, matching the seed labels

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

constrained integer optimization problem

we know how to rewrite the problem

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

$L = D - W$: graph Laplacian

Solution:

without loss of generality assume that Laplacian matrix is partitioned into blocks for labeled and unlabeled nodes

$$L = \begin{bmatrix} L_{ss} & L_{su} \\ L_{us} & L_{uu} \end{bmatrix}$$

accordingly let $y = \begin{bmatrix} y_s \\ y_u \end{bmatrix} = \begin{bmatrix} \hat{y}_s \\ y_u \end{bmatrix}$ vector of labels to be learned.

$$y_u = -L_{uu}^{-1} \cdot L_{us} \cdot \hat{y}_s$$

Label Propagation: generalization:

What if we have k labels?

- one hot encoding $y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is of class } k \\ 0 & \text{else} \end{cases}$
- energy function: $E(Y) = \sum_{i,j} w_{ij} (y_i - y_j)^T (y_i - y_j)$

- other types of effects encode with compatibility matrix H
- Energy function $E(Y) = \sum_{i,j} w_{ij} (y_i - y_j)^T H (y_i - y_j)$

Label propagation vs SBM :

LP is a discriminative model → models the conditional distribution of labels given the similarity graph $p(Y|W)$

SBM is a generative model → models $Pr(A|Z)$ and $Pr(Z)$

For SBM we get posterior using Baye's Theorem.

SBM and LP solve a different problem.

SBM → estimate parameters generated given graph A (unsupervised)

LP → predict labels of the nodes in U given observed labels in L.

Non Graph Data : if we only have vector data
- construct a graph connecting similar points (ex. kNN)
- apply label propagation
- Features are used to construct the graph, but then only graph is used for classification.

Transductive Learning :

LP is a special case of Transductive Learning

given : (i) set of labelled data $T = \{(x_i, y_i), i=1..n\} \subset X \times Y$

(ii) set of unlabeled data $U = \{(x_i)_{i=1,..m}\} \subset X$

goal : • predict labels only for unlabelled instance U

(AND NOT SOLVE HARDER INTERMEDIATE PROBLEM)

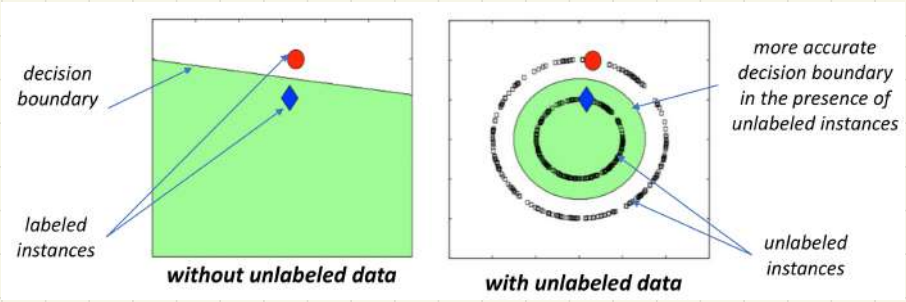
supervised learning is inductive.

- general idea: Semi supervised learning

definition: given labelled data (T) and unlabeled data (U)

main idea: use both T and U to learn mapping f

How does semi supervised learning help?



(Semi supervised) Deep Learning on Graphs:

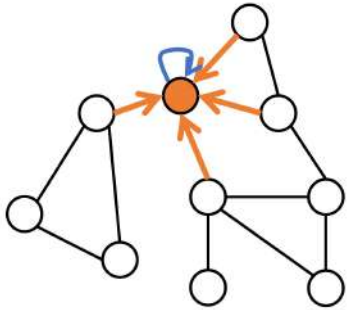
Differential message passing framework

given:
- $G = (V, E)$ nodes V, Edges E, E_{uv} → edge weights
- each node v has features $x_v \in \mathbb{R}^d$
- subset of labelled nodes $S \subseteq V$, y_v → labels of nodes

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

- For each node do:
  1) gather message from all neighbours
  $$m_v^{(k)} = \sum_{u \in N(v)} M(h_v^{(k-1)}, h_u^{(k-1)}, E_{vu})$$
  2) update the hidden representation
  $$h_v^{(k)} = U(h_v^{(k-1)}, m_v^{(k)})$$

M & U are differentiable fn → Neural Nets.



1. Gather message from all neighbors
2. Update hidden representation

instantiation :

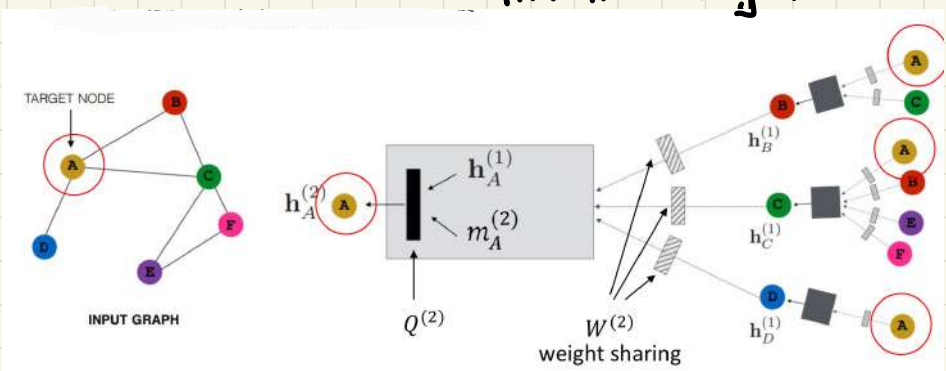
- let $h_v^{(0)} = x_v$ → at the first layer the representation are node features
- set of message aggregate functions to be the average over the neighbours

$$m_v^{(k)} = \sum_{u \in N(v)} \frac{1}{d_v} \left(W^{(k)} h_u^{(k-1)} + b^{(k)} \right)$$

- calculate hidden representations as simple NNs with a non linearity

$$h_v^{(k)} = \text{relu}\left(B^{(k)} h_v^{(k-1)} + p^{(k)} + m_v^{(k)} \right)$$

$W^{(k)}$, $b^{(k)}$, $B^{(k)}$, $p^{(k)}$ are trainable parameters of the k-th layer.



How do Neighbours Influence a Given Node

observation 1: K hidden layers equal to K steps of message passing

- at step 1, node v aggregates info of 1hop neighbour
- at step 2, node v (implicitly) aggregates info from neighbour of 1hop neigh = 2 hop

at step K : implicitly K hop

observation 2: K hidden layers → representation $h_v^{(K)}$ has info -based on K-hop neighbour.

How to perform semi-supervised Node classification

- $h_v^{(K)}$ at final layer → K logits
- $p_v = \text{softmax}(h_v^{(K)})$ → probability that node v belongs to class c

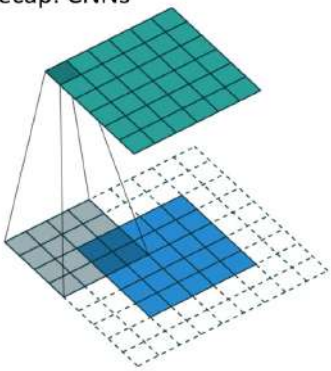
$$\min_{\{W^{(k)}, B^{(k)}, p^{(k)}, b^{(k)}\}_{k=1\ldots K}} -\sum_{v \in S}\sum_{c \in C} y_{vc} \log p_{vc}$$

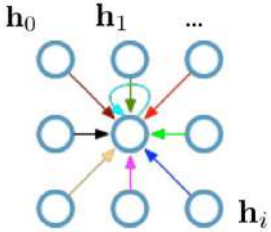
y_{vc}: one hot encoded; probability of node v belonging to class c

S: labelled nodes

C: set of classes

- Alternative view of message passing framework is that we are essentially performing a "graph convolution"
- Recap: CNNs

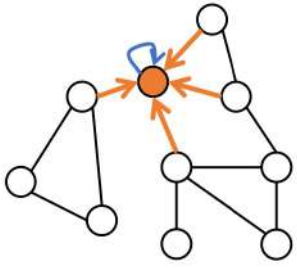


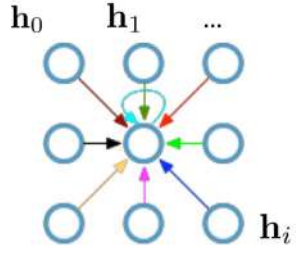


Update for a single pixel:

- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

- Images are a special kind of graph: every pixel is a node connected to 8 other nodes (up, down, left, right, etc. pixels)
  - Note: Unlike sequences/images, where the convolution operation is clearly defined, there are various versions for graphs. Indeed, one often distinguishes between spatial and spectral approaches.





In spatial domain, graph conv. updates each node's features by considering the local neighbourhood.
- In effect it is some kind of message passing

$$x_i^{(t+1)} = x_i^{(t)} + \sum_{(i,j)\in E} x_j^{(t)}$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Non-linearity · Message Passing · Feature Transformation

Normalizing propogation matrix avoids gradient explosion
$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ where $\tilde{A} = A + I_n$ $D_{ii} = \sum_j \tilde{A}_{ij}$ is degree matrix $\tilde{A}$

#How deep are GNN?

furthest distance that graph propagates info to

$$H^{(k+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(k)}W^{(k)}\right)$$

GCNs apply 1 transformation per message passing step

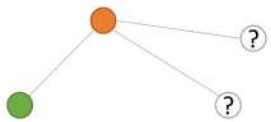
Transformation depth and propagation depth are orthogonal in GNNs.

- We could transform the messages with a multi-layer network in each step or propagate without transforming for multiple steps

Multi graph

- Given:
  - A set of graphs $\mathcal{G} = \{G_i = (V_i, E_i)\}_{i=1..N}$
  - A subset of labeled graphs $\mathcal{H} \subseteq \mathcal{G}$, with y_{G_i} denotes the graph level target of the graph $G_i \in \mathcal{H}$
- Obtain the hidden representation of the last layer for all nodes in all graphs according to the message passing framework
  - Let $H_{G_i} = \left[h_1^{(K)}, h_2^{(K)}, \ldots, h_{|V_i|}^{(K)}\right]$ be a matrix where we stacked the final representations **for all nodes** for graph G_i
- Define an aggregation function $R(H_{G_i})$ over the node representations of a given graph that produces a representation for the **entire** graph $h_{G_i} = R(H_{G_i})$
- For classification: consider h_{G_i} as the logits and train a model using standard cross-entropy loss, i.e. $\mathcal{L}(y_{G_i}, softmax(h_{G_i}))$
  - For regression, e.g. predicting functional properties of molecules you can use squared loss
- Example agg. function $R(H_{G_i}) = \frac{1}{|V_i|}\sum_{v \in V_i} h_v^{(K)}$ is the average of the node embedding

- Consider the graph below. What is the influence of the green node on the unlabeled nodes in Label Propagation? Why? How about GNNs with $K = 2$?



- LP does not care about connections b/w labelled nodes.
- In GNN., 2 hop influence *

- Does semi-supervised learning exist outside of learning on graphs?

yes on vector data as well

- Can you apply GNNs to vector data without a specified graph structure?

GNN is then equivalent to MLP which classifies nodes based on own features

Problem 1: The goal in Label Propagation is to find a labeling $\boldsymbol{y} \in \{0,1\}^N$ that minimizes the energy $\min_{\boldsymbol{y}} \frac{1}{2}\sum_{ij} \boldsymbol{w}_{ij}(y_i - y_j)^2$ subject to $y_i = \hat{y}_i \; \forall i \in S$ where the set of nodes V has been partitioned into the labeled nodes S and the unlabeled nodes U, $w_{ij} \geq 0$ is the non-negative edge weight and $\hat{y}_i$ are the observed labels.

Following from the first observation regarding the Laplacian, the minimization problem can be rewritten and then relaxed to $\min_{\boldsymbol{y}\in\mathbb{R}^N} \boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y}$ subject to the same constraints. Show that the closed form solution is

$$\boldsymbol{y}_U = -\boldsymbol{L}_{UU}^{-1} \cdot \boldsymbol{L}_{US} \cdot \hat{\boldsymbol{y}}_S$$

where w.l.o.g. we assume that the Laplacian matrix is partitioned into blocks for labeled and unlabeled nodes as

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_{SS} & \boldsymbol{L}_{SU} \\ \boldsymbol{L}_{US} & \boldsymbol{L}_{UU} \end{pmatrix}.$$

P1) $y^T L y = y^T \begin{pmatrix} L_{SS} & L_{SU} \\ L_{US} & L_{UU} \end{pmatrix} y$

$= \hat{y}_S^T L_{SS} \hat{y}_S + \hat{y}_S^T L_{SU} y_U + y_U^T L_{US} y_S + y_U^T L_{UU} y_U$

Laplacian is symmetric

$= \hat{y}_S^T L_{SS} \hat{y}_S + 2 y_U^T L_{US} \hat{y}_S + y_U^T L_{UU} y_U = f(y_U)$

$\frac{\partial f}{\partial y_U} = 2 L_{US} \hat{y}_S + (L_{UU} + L_{UU}^T) y_U = 2 L_{US} \hat{y}_S + 2 L_{UU} y_U = 0$

$y_U = -L_{UU}^{-1} L_{US} \hat{y}_S$

Data Analytics and Machine Learning Group
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Machine Learning for Graphs and Sequential Data

Exam:	IN2323 / Endterm	**Date:**	Wednesday 5th August, 2020
Examiner:	Prof. Dr. Stephan Günnemann	**Time:**	11:30 – 12:45

Working instructions

- This exam consists of **14 pages** with a total of **10 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 43 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - all materials that you will use on your own (lecture slides, calculator etc.)
  - **not allowed are any forms of collaboration between examinees and plagiarism**
- You have to sign the code of conduct.
- Make sure that the **QR codes are visible** on every uploaded page. Otherwise, we cannot grade your exam.
- Only write on the provided sheets, **submitting your own additional sheets is not possible**.
- Last two pages can be used as scratch paper.
- All sheets (including scratch paper) have to be submitted to the upload queue. Missing pages will be considered empty.
- **Only use a black or blue color (no red or green)!**
- Write your answers only in the provided solution boxes or the scratch paper.
- **For problems that say "Justify your answer" you only get points if you provide a valid explanation.**
- **For problems that say "Prove" you only get points if you provide a valid mathematical proof.**
- If a problem does not say "Justify your answer" or "Prove" it's sufficient to only provide the correct answer.
- Exam duration - 75 minutes.

Left room from _____ to _____ / Early submission at _____

Problem 1 Normalizing Flows (4 credits)

We consider two transformations $f_1(\boldsymbol{z}) = \begin{bmatrix} z_1 \\ z_2^{1/3} \end{bmatrix}$ and $f_2(\boldsymbol{z}) = \begin{bmatrix} z_1(|z_2|+1) \\ z_2 \end{bmatrix}$ from $\mathbb{R}^2$ to $\mathbb{R}^2$.

The respective inverse transformation are $f_1^{-1}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ x_2^3 \end{bmatrix}$ and $f_2^{-1}(\boldsymbol{x}) = \begin{bmatrix} \frac{x_1}{|x_2|+1} \\ x_2 \end{bmatrix}$.

The respective Jacobians are

$$J_{f_1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{3}z_2^{-\frac{2}{3}} \end{bmatrix} \qquad J_{f_2} = \begin{bmatrix} |z_2|+1 & \text{sign}(z_2)z_1 \\ 0 & 1 \end{bmatrix}$$

$$J_{f_1^{-1}} = \begin{bmatrix} 1 & 0 \\ 0 & 3x_2^2 \end{bmatrix} \qquad J_{f_2^{-1}} = \begin{bmatrix} \frac{1}{|x_2|+1} & \frac{-\text{sign}(x_2)x_1}{(|x_2|+1)^2} \\ 0 & 1 \end{bmatrix}$$

0 1 2 3 4

We assume a Gaussian base distribution $p_1(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We observed one point $\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

We propose to stack the transformations f_1, f_2 to transform the base distribution p_1 in the distribution p_2 with normalizing flows. Compute the likelihood for $\boldsymbol{x}$ under the transformed distribution p_2 if the order of transformations is f_1 followed by f_2.

Hint: You might use the density of the unit variate Gaussian $p = \mathcal{N}(0, 1)$ at the following points: $p(1/2) = 0.3521$, $p(1/3) = 0.3774$, $p(1/9) = 0.3965$, $p(5) = 1.4867e^{-06}$, $p(8) = 5.0523e^{-15}$, $p(10) = 7.6946e^{-23}$

$f_2 \circ (f_1 \circ z) = x \qquad x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$z = f_1^{-1} \circ (f_2^{-1} \circ x)$ —— (1)

$f_2^{-1}(x) = \begin{bmatrix} 1/3 \\ 2 \end{bmatrix} \qquad f_1^{-1} \circ f_2^{-1}(x) = \begin{bmatrix} 1/3 \\ 8 \end{bmatrix}$

$p_2(x) = p_1\left(\begin{bmatrix} 1/3 \\ 8 \end{bmatrix}\right) \times \left(\frac{1}{|2|+1}\right) \times (3 \times 4)$

$= p_1\left(\begin{bmatrix} 1/3 \\ 8 \end{bmatrix}\right) \times 4 = 7.6266e^{-15}$

Problem 2 Variational Inference (5 credits)

We are performing variational inference in some latent variable model $p_\theta(x, z)$ using the following family of variational distributions $\mathcal{Q}_1 = \{\mathcal{N}(z|\phi, 1) : \phi \in \mathbb{R}\}$.

a) Assume that the variational distribution $q \in \mathcal{Q}_1$ is fixed, and we are trying to maximize the ELBO w.r.t. θ using gradient ascent. Is it necessary to use the reparametrization trick in this case? If yes, explain how to do it for our family of distributions $\mathcal{Q}_1$; if not, provide a justification.

0 1 2 3

No, we don't need the reparameterization trick for calculating ELBO when performing gradient descent with θ.

We can estimate

$$\mathbb{E}_{z\sim q_\phi(x)}[f_\theta(z)] \simeq \frac{1}{S}\sum_{i=1}^{S} f_\theta(x_i)$$

$$\nabla \mathbb{E}_{z\sim q_\phi(z)}[f_\theta(z)] = \mathbb{E}_{z\sim q_\phi(z)}[\nabla f_\theta(z)]$$

$$\simeq \frac{1}{S}\sum_{i=1}^{S} \nabla_\theta f_\theta(x_i)$$

b) Consider another family of distributions $\mathcal{Q}_2 = \{\mathcal{N}(z|0, s^2) : s \in (0, \infty)\}$. Which of the following statements is true? Justify your answer.

0 1 2

1. $\max_{\theta, q\in\mathcal{Q}_1} \text{ELBO}(\theta, q) < \max_{\theta, q\in\mathcal{Q}_2} \text{ELBO}(\theta, q)$
2. $\max_{\theta, q\in\mathcal{Q}_1} \text{ELBO}(\theta, q) = \max_{\theta, q\in\mathcal{Q}_2} \text{ELBO}(\theta, q)$
3. $\max_{\theta, q\in\mathcal{Q}_1} \text{ELBO}(\theta, q) > \max_{\theta, q\in\mathcal{Q}_2} \text{ELBO}(\theta, q)$
4. It's impossible to tell without additional information.

- impossible to tell without any knowledge of $p_\theta(x, z)$
- Depends on the model.

Problem 3 Robustness of Machine Learning Models (6 credits)

Suppose we have trained a binary classifier $f : \mathbb{R}^d \to \{0, 1\}$ and want to certify its robustness via randomized smoothing. Therefore, the *smoothed classifier* $g_{\sigma^2}(\boldsymbol{x}) = \mathbb{E}[\mathbb{I}[f(\boldsymbol{x} + \varepsilon) = 1]]$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$.

Fact: $\Phi^{-1}(g_{\sigma^2}(\boldsymbol{x}))$ is $1/\sigma$-**Lipschitz** w.r.t. $\boldsymbol{x}$ and the L_2 norm, where $\Phi(z)$ denotes the cumulative distribution function (CDF) of the standard normal distribution.

0 1 2 3 4

a) **Using the above fact about the Lipschitz-continuity** of $\Phi^{-1}(g_{\sigma^2}(\boldsymbol{x}))$, show that the largest certifiable L_2 radius r around a sample $\boldsymbol{x}$ is identical to the result shown in the lecture. More precisely, show that

$$r = \sigma \Phi^{-1}(g_{\sigma^2}(\boldsymbol{x})).$$

Hint: You may assume we can evaluate $g_{\sigma^2}(\boldsymbol{x})$ in closed form and you may use the following results: $\lim_{z \to 0} \Phi^{-1}(z) = -\infty$, $\Phi^{-1}(0.5) = 0$, $\lim_{z \to 1} \Phi^{-1}(z) = \infty$.

$$\| \Phi^{-1}(g_{\sigma}(x)) - \underbrace{\Phi^{-1}(0.5)}_{0} \|_2 \leq \frac{1}{\sigma} \| x - \tilde{x} \|_2$$

$$\Phi^{-1}(g_{\sigma}(x)) \leq \frac{1}{\sigma} \| x - \tilde{x} \|$$

$$\| x - \tilde{x} \|^2 \geq \sigma \Phi^{-1}(g_{\sigma}(x))$$

0 1 2

b) A fellow student has a promising idea: by letting $\sigma \to \infty$ we can make the Lipschitz constant of the smoothed classifier arbitrarily small, leading to arbitrarily large certifiable radii, i.e. a very robust model. Is this a good idea? Why or why not?

① if σ goes to infinity then the variance of the noise is arbitrarily large ; and signal / noise ≈ 0 ; we lose all our info

② for large σ we sample a really huge space for which large fraction of samples belong to other states.

Problem 4 **Markov Property (3 credits)**

We consider the following sequences of random variables $U_0, U_1, \ldots, U_t$.

a) $U_t = \begin{bmatrix} X_t \\ Z_t \end{bmatrix}$ where X_t are observed variables and Z_t are latent variables of an Hidden Markov Model. Does the sequence of variables U_t fulfill the Markov property i.e. $P(U_t|U_{t-1}) = P(U_t|U_{t-1}, ..., U_0)$? Justify your answer.

0
1

$P(U_t \mid U_{t-1}, .. U_1, U_0) = P\left(\begin{bmatrix} X_t \\ Z_t \end{bmatrix} \mid \begin{bmatrix} X_{t-1} \\ Z_{t-1} \end{bmatrix}, \ldots \begin{bmatrix} X_0 \\ Z_0 \end{bmatrix}\right)$

$P(X_t, Z_t \mid X_{t-1}, Z_{t-1}, \ldots X_0 Z_0) = P(X_t, Z_t \mid X_{t-1}, Z_{t-1})$

all info is retained, hence this is true

b) We consider an AR(p) process X_t. Under what condition on p and k does the sequence of variables $U_t = [X_{t-1}, ..., X_{t-k}]$ fulfill the Markov property i.e. $P(U_t|U_{t-1}) = P(U_t|U_{t-1}, ..., U_0)$? Justify your answer.

0
1

c) We consider a recurrent neural network which produces X_t. Does the sequence of variables $U_t = X_t$ fulfill the Markov property i.e. $P(U_t|U_{t-1}) = P(U_t|U_{t-1}, ..., U_0)$? Justify your answer.

0
1

Recurrent neural networks have all the past info encoded in the hidden state h_t and both x_t and x_{t-2} depends on h_{t-2} hence not possible

Problem 5 **Markov Chain (3 credits)**

0 1 2 3

We consider a Markov chain X_t in $\{1, C\}$ with parameters $\pi, \boldsymbol{A}$. We assume we observed the sequence $S_k = [\underbrace{v_0, \dots, v_0}_{k \text{ times}}, \underbrace{v_1, \dots, v_1}_{k \text{ times}}, \dots, \underbrace{v_T, \dots, v_T}_{k \text{ times}}]$ where each value is observed k times. The parameter k can be seen as a discretization parameter of the time space.

Compute the likelihood of the sequence under the parameters $\pi, \boldsymbol{A}$ i.e. $P_{\pi,\boldsymbol{A}}(S_k)$. What happens to this quantity if you increase the discretization parameter from k to $k' > k$ but keep the same model parameter $\pi, \boldsymbol{A}$?

$$P_{\pi,A}(S_k) = \pi_{v_0} \times \prod_{t=0}^{T-1} A_{v_t, v_{t+1}} \prod_{t=0}^{T} A_{v_t, v_t}^{k-1}$$

Since $A_{i,j} < 1$ → likelihood decreases as k increases

Problem 6 Temporal Point Process (6 credits)

Consider an inhomogeneous Poisson process (IPP) on the interval $[0, 4]$ with the intensity function

$$\lambda(t) = \begin{cases} a & \text{if } t \in [0, 3] \\ b & \text{if } t \in (3, 4] \end{cases}$$

where $a > 0$, $b > 0$ are some positive parameters.

a) Assume that you observed a sequence of events $\{0.2, 1.0, 1.5, 2.9, 3.1, 3.8\}$ generated by the above IPP. What is the maximum likelihood estimate of the parameters a and b?

0 1 2 3 4

$\log P(\{t_1, \ldots t_n\}) = \sum_{i=1}^{N} \log \lambda(t_i) - \int_0^T \lambda(t).dt$

$= 4\log(a) + 2\log b - (3a + b).$

$(LL) = 4\log(a) + 2\log(b) - 3a - b.$

$\frac{\partial LL}{\partial a} = \frac{4}{a} - 3 \Rightarrow a^* = \frac{4}{3}.$

$\frac{\partial LL}{\partial b} = \frac{2}{b} - 1 \Rightarrow b^* = 2.$

b) Assume that $a = 1$ and $b = 5$. What is the expected number of events generated by the IPP in this case?

0 1 2

Expected number of events : $\int_0^T \lambda^*(t)\, dt$

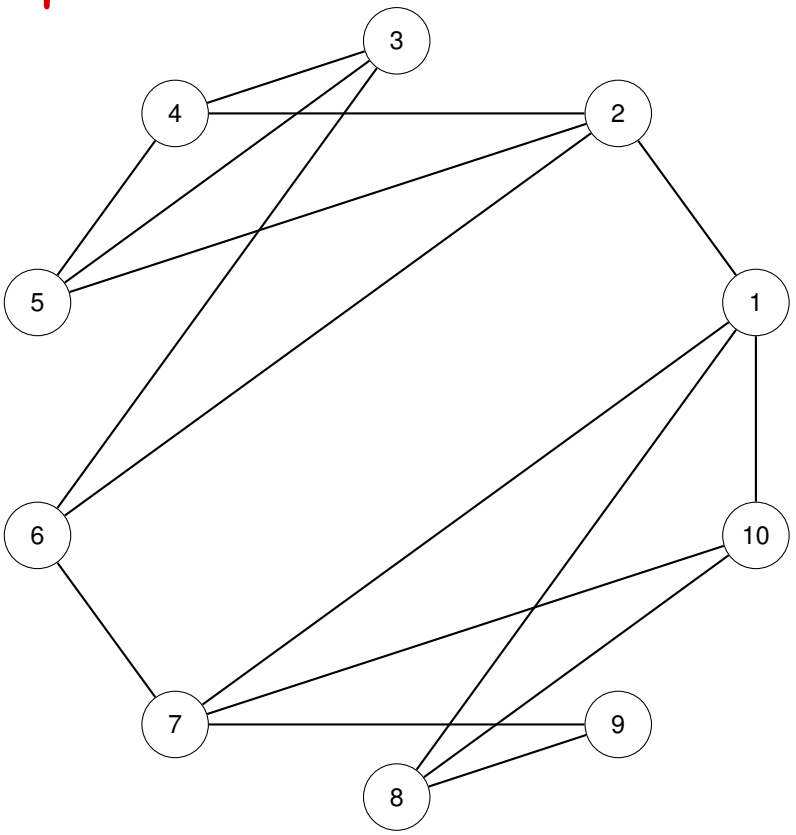
$= \int_0^3 a.dt + \int_3^4 b.dt$

$= 3*(a) + b = 3 + 5 = 8$

Problem 7 Clustering with the Planted Partition Model (4 credits)

0 1 2 3 4

The following graph has been generated from a planted partition model with in-community edge probability p and between-community edge probability q.



Assuming $p < q$, find the maximum likelihood community assignments under a PPM.
Give your solution as two sets of node labels making up the two discovered communities. Justify your answer.

Likelihood $\propto \frac{q(1-p)}{p(1-q)}$ - $\left(\frac{q}{p}\right)\left(\frac{1-p}{1-q}\right)$ >1 >1

for $p < q$ likelihood > 1

∴ likelihood is maximized by max cut size

Problem 8 PageRank in a Wheel (6 credits)

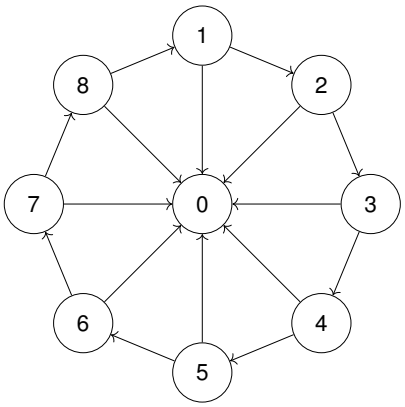


Figure 8.1: Example of a directed wheel graph with $n+1 = 9$ nodes

Consider a directed graph of size $n+1$ with a cycle of n nodes and an additional central node that every other node connects to (see figure). So we have a graph with node set $\mathcal{V} = \{0, 1, \dots, n\}$ and edge set

$$\mathcal{E} = \{(i, i+1) \mid i \in \{1, \dots, n-1\}\} \cup \{(n, 1)\} \cup \{(i, 0) \mid i \in \{1, \dots, n\}\}\,.$$

We want to compute the PageRank scores with a link-follow probability of β (a teleport probability of $1-\beta$) and some arbitrary teleport vector π, $\sum_{i=0}^{n} \pi_i = 1$. Note that we index π from 0 to n.
We define the predecessor function pa as the index of the predecessor of a node in the directed cycle, i.e.

$$\text{pa}(1) = n \quad \text{and} \quad \text{pa}(i) = i - 1 \;\; \forall i \in \{2, \dots, n\}\,.$$

You can write $\text{pa}^k(i)$ for the k-th predecessor of node i, i.e. $\text{pa}^3(i) = \text{pa}(\text{pa}(\text{pa}(i)))$ and $\text{pa}^0(i) = i$.

a) Set up the PageRank equations for all nodes in scalar form, i.e. each r_i separately instead of matrix form.

0
1
2

node 1, 2, .. n have two outgoing edges

$r_i = \beta \frac{r_{\text{pa}(i)}}{2} + (1-\beta)\pi_i \quad \forall \{1, \dots n\}$

central node only has incoming nodes from all the outer nodes

$r_0 = \beta \sum_{i=1}^{n} \left(\frac{r_i}{2}\right) + (1-\beta)\pi_0$

b) Why is this graph problematic for PageRank without random teleportation ($\beta = 1$)?

0
1

dead end at center. Random walks will be stuck there

0
1
2
3

c) Show that the PageRank for node $i \in \{1, \dots, n\}$ in the outer cycle is given by

$$r_i = \frac{(1-\beta)}{1-\left(\frac{\beta}{2}\right)^n} \sum_{j=0}^{n-1} \left(\frac{\beta}{2}\right)^j \pi_{\mathrm{pa}^j(i)}.$$

$$\pi_i^{(1)} = \frac{\beta}{2}\pi_{\mathrm{pa}(i)} + (1-\beta)\pi_i^{(0)}$$

$$= \frac{\beta}{2}\left(\frac{\beta}{2}\pi_{\mathrm{pa}(\mathrm{pa}(i))} + (1-\beta)\pi_{\mathrm{pa}(i)}\right) + (1-\beta)\pi_i$$

$$= \left(\frac{\beta}{2}\right)^2 \pi_{\mathrm{pa}(\mathrm{pa}(i))} + \frac{\beta}{2}(1-\beta)\pi_{\mathrm{pa}(i)} + (1-\beta)\pi_i$$

$$= \left(\frac{\beta}{2}\right)^2 \left[\frac{\beta}{2}\pi_{\mathrm{pa}(\mathrm{pa}(\mathrm{pa}(i)))} + (1-\beta)\pi_{\mathrm{pa}(\mathrm{pa}(i))}\right] + \frac{\beta}{2}(1-\beta)\pi_{\mathrm{pa}(i)} + (1-\beta)\pi_i$$

$$= \left(\frac{\beta}{2}\right)^3 \pi_{\mathrm{pa}(\mathrm{pa}(\mathrm{pa}(i)))} + \left(\frac{\beta}{2}\right)^2 (1-\beta)\pi_{\mathrm{pa}(\mathrm{pa}(i))} + \frac{\beta}{2}(1-\beta)\pi_{\mathrm{pa}(i)} + (1-\beta)\pi_i$$

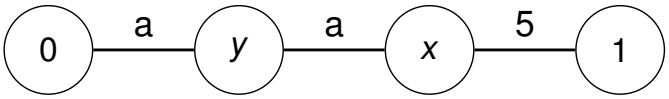
$$+ \dots$$

$$\pi_i^{(k)} = \left(\frac{\beta}{2}\right)^k \pi_{\mathrm{pa}^k(i)} + (1-\beta)\sum_{j=0}^{k-1}\left(\frac{\beta}{2}\right)^j \pi_{\mathrm{pa}^j(i)}$$

Problem 9 **Label Propagation (4 credits)**

0
1
2
3
4

Consider the following graph



The nodes labeled 0 and 1 are observed and from class 0 and 1, respectively. One edge has a fixed weight, the other two have a variable edge weight of $a \geq 0$. The two center nodes are unobserved and we call their labels x and y.

We want to predict classes for the two center nodes that minimize the Label Propagation objective exactly,

$$\frac{1}{2}\sum_{ij} W_{ij}\left(y_i - y_j\right)^2$$

where W is the weighted adjacency matrix and y_i, y_j are the labels of the nodes.

Find the set of all possible edge weights a that guarantee that node x is assigned to class 0. Justify your answer.

$5(x-1)^2 + a(x-y)^2 + a(y-0)^2 = (a+5)x^2 - 10x + 5$
$- 2axy + 2ay^2 = E(a,x,y)$

x	y	E(a,0,0)
0	0	5
1	0	a
0	1	5+2a
1	1	a

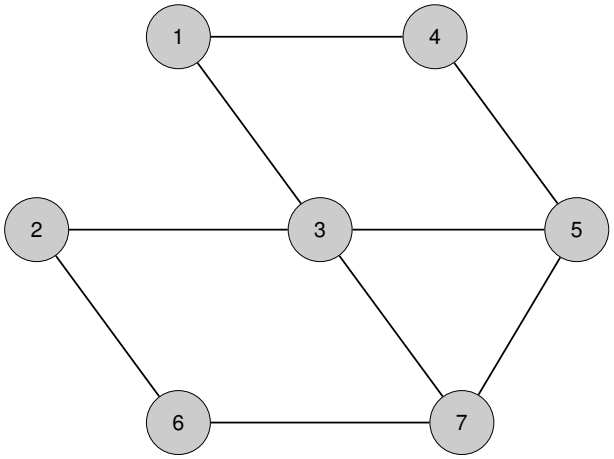
We set $x=0$ when $a \geq 5$

Problem 10 Adversarial Attacks on Graph Neural Networks (2 credits)

Suppose you are given the following two-layer graph neural network.

$$f(\boldsymbol{A}, \boldsymbol{X}) = \boldsymbol{Z} = \text{Softmax}\left(\hat{\boldsymbol{A}}\text{ReLU}\left(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_1\right)\boldsymbol{W}_2\right)$$

$\boldsymbol{X} \in \mathbb{R}^{N \times D}$ are the node features, $\boldsymbol{Z}$ are the node predictions, $\boldsymbol{W}_x$ are weight matrices of appropriate dimensions and $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the propagation matrix as defined for GCNs. Here, $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, where $\boldsymbol{A}$ is the adjacency matrix and $\boldsymbol{I}$ is the identity matrix, and $\tilde{\boldsymbol{D}}$ is a diagonal matrix of node degrees $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$.
The model was trained for the task of semi-supervised node classification, and we want to predict a class c for node 6 in the following graph $\boldsymbol{A}$:



0 1 a) An adversary with complete knowledge about the graph $\boldsymbol{A}$ and the trained model $f(\boldsymbol{A}, \boldsymbol{X})$ may delete one edge to perturb the prediction for node 6. Deleting which of the following edges would lead to a greater change to the prediction for node 6? Justify your answer.

1. The edge connecting node 5 and 7
2. The edge connecting node 3 and 5
3. There is not enough information to determine which deletion leads to a greater change.

5 → 7 as it is two hops away

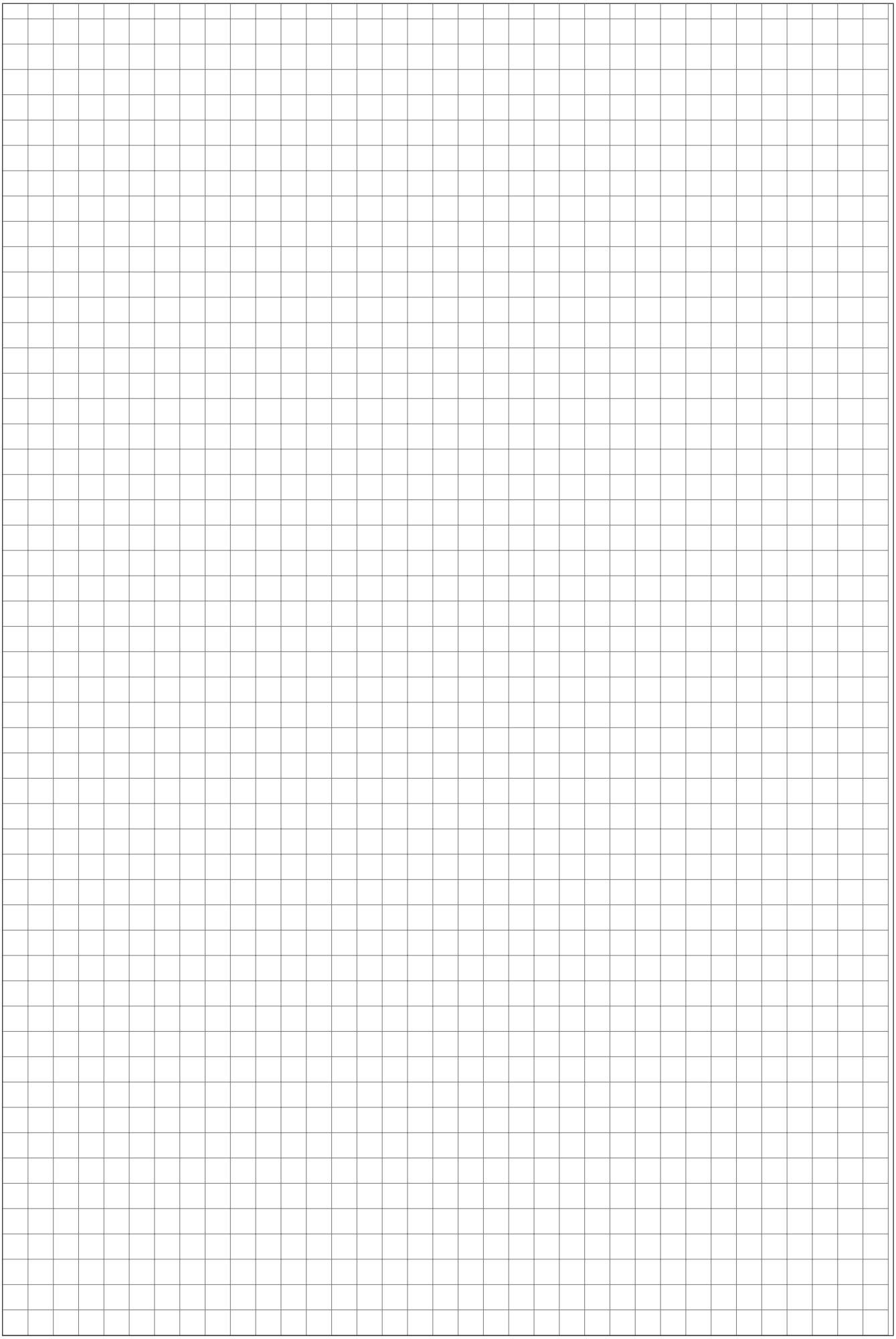
3 → 5 won't affect

0 1 b) Assume we instead have a Personalized Propagation of Neural Predictions (PPNP) model instead of the two-layer GCN. How does this affect your choice? Justify your answer.

PPNP is infinite propagation with teleport

not enough info

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.



Data Analytics and Machine Learning Group
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Machine Learning for Graphs and Sequential Data

Exam:	IN2323 / Retake	**Date:**	Wednesday 7th October, 2020
Examiner:	Prof. Dr. Stephan Günnemann	**Time:**	14:15 – 15:30

Working instructions

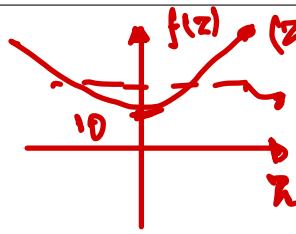
- This exam consists of **14 pages** with a total of **10 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 38 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - all materials that you will use on your own (lecture slides, calculator etc.)
  - **not allowed are any forms of collaboration between examinees and plagiarism**
- You have to sign the code of conduct.
- Make sure that the **QR codes are visible** on every uploaded page. Otherwise, we cannot grade your exam.
- Only write on the provided sheets, **submitting your own additional sheets is not possible**.
- Last two pages can be used as scratch paper.
- All sheets (including scratch paper) have to be submitted to the upload queue. Missing pages will be considered empty.
- **Only use a black or blue color (no red or green)!**
- Write your answers only in the provided solution boxes or the scratch paper.
- **For problems that say "Justify your answer" you only get points if you provide a valid explanation.**
- **For problems that say "Prove" you only get points if you provide a valid mathematical proof.**
- If a problem does not say "Justify your answer" or "Prove" it's sufficient to only provide the correct answer.
- Instructor announcements and clarifications will be posted **on Piazza** with email notifications
- Exam duration - 75 minutes.

Left room from _____ to _____ / Early submission at _____

Problem 1 Normalizing Flows (5 credits)

0 1

a) Let $k \in \mathbb{N}$. We consider the transformation $f(\boldsymbol{z}) = (z - 10)^{2k}$ from $\mathbb{R}$ to $\mathbb{R}$. Is this transformation invertible ? Justify your answer. If yes, compute the determinant of its Jacobian and its inverse.



the function is not a bijection
hence not invertible

0 1 2

b) We consider the transformation $f(\boldsymbol{z}) = \begin{bmatrix} z_n \\ z_1 \\ : \\ : \\ z_{n-1} \end{bmatrix}$ from $\mathbb{R}^n$ to $\mathbb{R}^n$. Is this transformation invertible ? Justify your answer. If yes, compute the determinant of its Jacobian and its inverse. *Hint: The determinant of one elementary permutation (i.e. a permutation which interchanges any two rows) is* -1

Yes, it can be seen as (n-1) stacked permutations

determinant $(-1)^{n-1}$

inverse $f^{-1}(z) = \begin{bmatrix} x_2 \\ x_3 \\ : \\ x_1 \end{bmatrix}$

how is this (n-1) stacked permutation?

0 1 2

c) We consider the transformation $f(\boldsymbol{z}) = \begin{bmatrix} z_1 + 2z_2 \\ 3z_1 + 4z_2 \\ z_3 - 2z_4 \\ 3z_3 - 4z_4 \end{bmatrix}$ from $\mathbb{R}^4$ to $\mathbb{R}^4$. Is this transformation invertible ? Justify your answer. If yes, compute the determinant of its Jacobian and its inverse.

4

$z_2 = x_1 - x_2$

$z_1 + 2z_2 = x_1$

$3x_1 + 4z_2 = x_2$

$z_1 = 2x_2 - x_1$

$J = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 3 & -4 \end{bmatrix}$

$|J| = (-2)(2) = -4$

as determinant $\neq 0$ invertible

Problem 2 Variational Inference (3 credits)

0 1 2 3

We are performing variational inference in some latent variable model $p_\theta(x, z)$ using the following family of variational distributions $\mathcal{Q}_1 = \{\mathcal{N}(z|\phi, 1) : \phi \in \mathbb{R}\}$.
Assume that

$$p(z) = \mathcal{N}(z|0, 1) \propto \exp\left(-\frac{1}{2}z^2\right)$$
$$p(x|z) = \text{Bernoulli}(x \mid \sigma(z)) \propto \exp\left(xz - \log(1 + \exp(z))\right)$$

and x is known and fixed and $\sigma(z) = \frac{\exp(z)}{1+\exp(z)}$ is the sigmoid function. Does there exist a $q^\star \in \mathcal{Q}_1$, such that $\mathbb{KL}(q^\star(z) \parallel p(z|x)) = 0$? Justify your answer.

for $KL(q^*(z) \parallel p(z|x)) = 0$

$q^*(z) = p(z|x)$

$p(z|x) \propto p(x|z) \cdot p(z)$

$\propto \exp(xz - \log(1+e^z)) \cdot \exp(-\frac{z^2}{2})$

$\propto \exp(xz - \log(1+e^z) - \frac{z^2}{2})$

this is not normal in z.

as $\log(1+e^z)$ → exponential term is present

Problem 3 Robustness of Machine Learning Models (3 credits)

Consider a trained binary logistic regression model with weight vector $\boldsymbol{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, where d is the data dimensionality. That is, the predicted probability of a sample $\boldsymbol{x} \in \mathbb{R}^d$ belonging to class 1 is:

$$p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b),$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid function. An input sample is assigned to class 1 if $p(y = 1|\boldsymbol{x}) > 0.5$, else it is assigned to class 0.
We would like to perform **robustness certification** of the logistic regression model using its **Lipschitz constant**. That is, we want to certify that the predicted class of the input sample does not change w.r.t. some radius in the input space.

0 1

a) Briefly explain why it is sufficient to consider $F(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$, i.e. the model without the sigmoid activation function $\sigma(\cdot)$ for this purpose.

Because we just need to know which side of boundary is the input mapped to. (classification)

0 1 2

b) Derive the (smallest possible) Lipschitz constant of $F(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ w.r.t. the L_2 norm. Justify your answer.

$\|w^Tx - w^T\tilde{x}\| \leq L\|x - \tilde{x}\|$

let $a = x - \tilde{x}$

$\|w^Ta\| \leq L\|a\|$

$L \geq \left(\frac{\|w^Ta\|}{\|a\|}\right)\|w\|$

$\frac{\|w^Ta\|}{\|a\|} \leq 1$

$L \geq \|w\|$

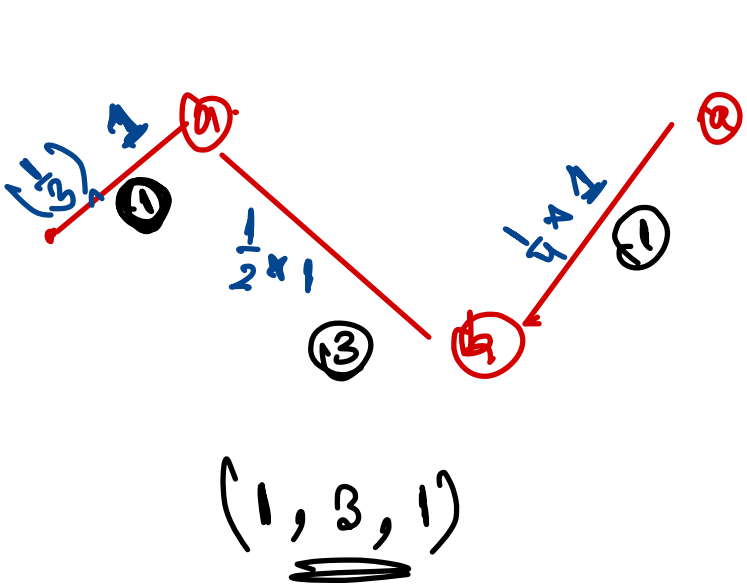
smallest value

Problem 4 Hidden Markov Model (4 credits)

Consider an HMM where hidden variables are in $\{1, 2, 3\}$ and observed variables are in $\{a, b\}$. Let the model parameters be as follows:

$$\pi = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \quad A = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \end{bmatrix}$$

Assume that the sequence $X_{1:3} = [aba]$ is observed. Find the most probable sequence $[Z_1, Z_2, Z_3]$ and compute its likelihood. Justify your answer.



Problem 5 Temporal Point Process (3 credits)

0 1 2 3

Consider a temporal point process defined on the interval $[0, 10\pi]$ with the conditional intensity function

$$\lambda^*(t) = \sin(t) + 2$$

What is the expected number of events that will be generated from this TPP on the interval $[0, 2\pi]$? Justify your answer.

$$\mathbb{E}[N[0,2\pi]] = \int_0^{2\pi} \lambda^*(t)\,dt = \int_0^{2\pi} (\sin(t)+2)\,dt$$

$$= -\cos t + 2t \Big|_0^{2\pi} = \boxed{4\pi}$$

Problem 6 Neural Network approaches for temporal data (4 credits)

0 1 2 3 4

We trained the following models to produce latent representations:

- M1: Skip-gram word2vec model
- M2: LSTM
- M3: Self-attention without positional encoding
- M4: Self-attention with positional encoding
- M5: A convolutional NN which produces the latent representation at time t based on the input at time t and $t-1$.

At **inference time**, we use the following 6 sentences as inputs:

- S1: "I left"
- S2: "They left"
- S3: "I left yesterday"
- S4: "I go left"
- S5: "left I go"
- S6: "go left"

For each model (i.e. M1, M2, M3, M4, M5), which sets of sentences (e.g. (S1, S2), (S3, S4, S6)) are guaranteed to produce the same latent representation for the word "left" ? Note that the answer may be zero, one or more sets of sentences. Justify your answer.

M1) skip gram : embeddings are the same, irrespective of the context

M2) LSTM : "I left" and "I left yesterday" produce same embedding → left context

M3) self attention without positional encoding:
"I go left" and "left I go" are the same

M4) self attention with position encoding :
embeddings are different

M5) 1-D CNN with window 2
I go left , go left same
I left ≠ I left yesterday.

Problem 7 PageRank Lollipop (7 credits)

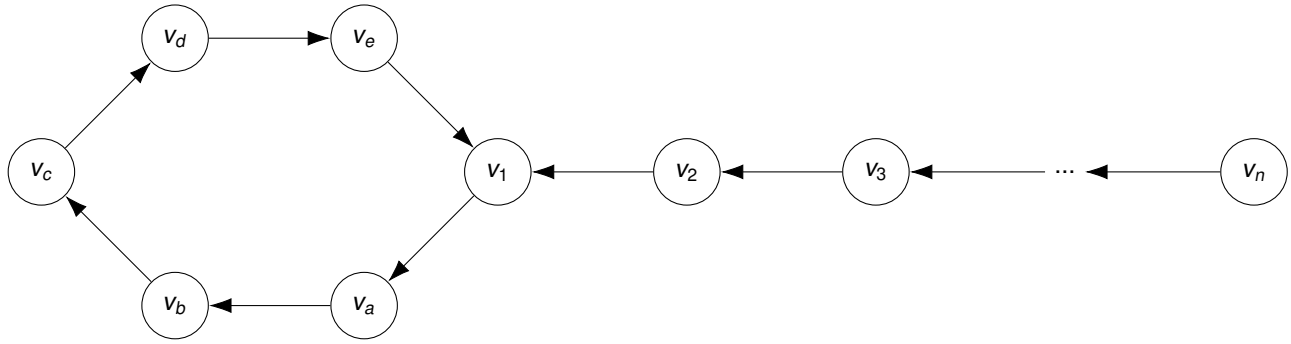


Figure 7.1: A directed "lollipop" graph with a tail of length n

Consider the directed, unweighted graph in Figure 7.1 with a "head" consisting of the six nodes v_1, and v_a through v_e. Its "tail" consists of n nodes $v_1, \ldots, v_n$ where n is a parameter. Note, that we consider v_1 to be part of the head as well as the tail.

0 1 2

a) Which of the PageRank problems of *dead end*, *spider trap*, and *periodic states* apply here? Justify your answer. For each problem that applies, give a set of edges (at most 3 per problem) that would resolve that problem if they were inserted.

① this is a spider trap, as some nodes become unreachable once we enter the loop.
connect and edge b/w the node in loop and open node

② Periodic problem exists in the loop (periodicity 6)
self loop to break periodicity.

b) In the following, we want to compute the topic-sensitive PageRank of the nodes. For that, we introduce the topic-sensitive teleport vector π where the topic is the "tail", i.e.

$$\pi_a = ... = \pi_e = 0 \quad \text{and} \quad \pi_1 = ... = \pi_n = \frac{1}{n}.$$

State each node's PageRank equation for a teleport probability of $1 - \beta$.

0 1 2

$M_n = \frac{1-\beta}{n}$ $\quad M_i = \beta M_{i+1} + \frac{1-\beta}{n}$ $\quad i \in (2 \dots n-1)$

$M_a = \beta M_1$ $\quad M_b = \beta M_a$ $\quad M_c = \beta M_b$ $\quad M_d = \beta M_c$

$M_e = \beta M_d$.

finally the node connecting the two.

$M_1 = \beta(M_e + M_2) + \frac{1-\beta}{n}$

c) Derive the sum of the PageRank of the "head" of the graph $v_1, v_a, \dots, v_e$ as a function of n. Justify your answer.

Reminder: $\sum_{i=0}^{k} \beta^i = \frac{1-\beta^{k+1}}{1-\beta}$ for $\beta \neq 1$

0 1 2 3

$M_a = \beta M_1$ $\quad M_b = \beta^2 M_1$ $\quad M_c = \beta^3 M_1$ $\quad M_d = \beta^4 M_1$ $\quad M_e = \beta^5 M_1$.

$M_2 = \beta M_3 + \left(\frac{1-\beta}{n}\right) = \beta\left(\beta M_4 + \left(\frac{1-\beta}{n}\right)\right) + \left(\frac{1-\beta}{n}\right) = \beta^2 M_4 + \beta\left(\frac{1-\beta}{n}\right)$

$= \beta^{n-2} M_n + \left(\frac{1-\beta}{n}\right) \sum_{i=0}^{n-2} \beta^i = \left(\frac{1-\beta}{n}\right)\left[\beta^{n-2} + \sum_{i=0}^{n-3} \beta^i\right] + \left(\frac{1-\beta}{n}\right)$

$M_1 = \beta(M_e + M_2) + \frac{1-\beta}{n}$

$= \beta\left(\beta^5 M_1 + 1 - \beta\right)$

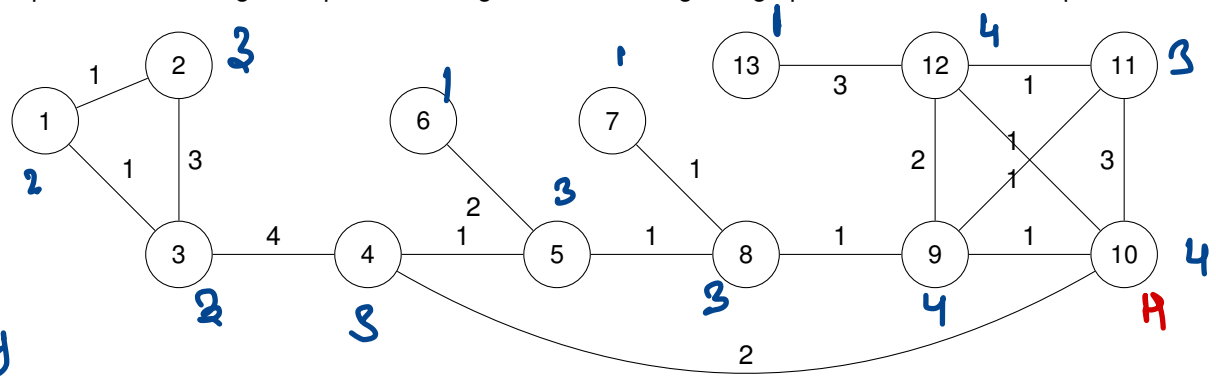
$1 + \beta + \dots \beta^{n-2}$

$= \left(\frac{1-\beta^{n-2}}{1-\beta}\right) \cdot \frac{1-\beta}{n}$

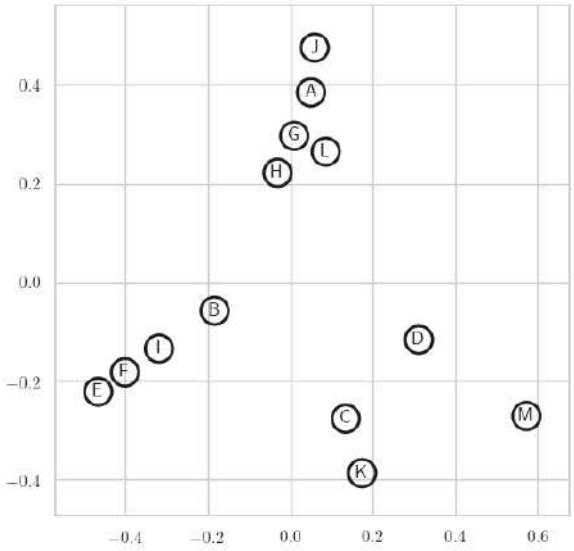
$\left\{1 - \frac{\beta^{n-3}}{n}\right.$

Problem 8 Spectral Embeddings (3 credits)

We use spectral embedding to map the following undirected, weighted graph into 2-dimensional space.



The following plot shows the embeddings as given by the eigenvectors belonging to the second and third smallest eigenvalues of the unnormalized Lapalacian.



Fill out the following table that maps nodes in the graph to points in the plot.

Node	1	2	3	4	5	6	7	8	9	10	11	12	13
Point	E	F	I	B	C	K	M	D	L	H	G	A	J

Problem 9 **Graph Neural Networks (4 credits)**

a) Consider the following graph neural network. 4 hop

0 1 2

$$\boldsymbol{Z} = \sigma\left(\sigma\left(\hat{\boldsymbol{A}}\sigma\left(\hat{\boldsymbol{A}}\sigma\left(\hat{\boldsymbol{A}}\boldsymbol{X}\right)\boldsymbol{W}_a\right)\right)\boldsymbol{W}_b\right)\boldsymbol{W}_c$$

$\boldsymbol{X} \in \mathbb{R}^{N\times D}$ are the node features, $\boldsymbol{Z}$ are the node predictions, σ is the sigmoid function, $\boldsymbol{W}_x$ are weight matrices of appropriate dimensions and $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the propagation matrix as defined for GCNs, i.e. $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix with added self-loops and $\tilde{\boldsymbol{D}}_{ii} = \sum_{j=1}^{N}\tilde{\boldsymbol{A}}_{ij}$ is the degree matrix of the amended adjacency matrix. Give the transformation and propagation depths of this model. Justify your answer.
Hint: We define

- *propagation depth* as the maximum distance that a model propagates information in the graph,
- *transformation depth* as the number of non-linear transformations that are applied to the features.

Each application of A propagates info 1 level
So propogation depth = 4

transformation depth = no. of nonlinear transforms = 4

b) Write down a formula for the node predictions $\boldsymbol{Z}$ of a graph neural network with propagation depth 2 and transformation depth 2 where the transformations use fully connected layers with weight matrices W_i and biases b_i. *Note*: You can assume that operations such as matrix-vector summation broadcast as expected.

0 1 2

$\hat{A}\,\hat{A}\,\sigma(\sigma(XW_1+b_1)W_2+b_2)$

Problem 10 Randomized Smoothing on Graph Neural Networks (2 credits)

Consider an arbitrary but fixed graph neural network $f(\boldsymbol{X})$ as base classifier. Here, $\boldsymbol{X}$ *denotes the adjacency matrix* (i.e. the features are assumed to be constant).
For the the smooth classifier $g(\boldsymbol{X})$ we use the randomization scheme $\phi(\mathbf{x})$:

$$g(\mathbf{x})_c = \mathcal{P}(f(\phi(\mathbf{x})) = c) = \sum_{\tilde{\mathbf{x}} \text{ s.t. } f(\tilde{\mathbf{x}})=c} \prod_{i=1}^{n^2} \mathcal{P}(\tilde{\mathbf{x}}_i|\mathbf{x}_i) \tag{1}$$

with

$$\mathcal{P}(\tilde{\mathbf{x}}_i|\mathbf{x}_i) = \begin{cases} p & \tilde{\mathbf{x}}_i = 1 - \mathbf{x}_i \\ 1-p & \tilde{\mathbf{x}}_i = \mathbf{x}_i \end{cases} \tag{2}$$

Note that in contrast to the lecture here we do not distinguish between a probability for deleting p_d or adding p_a an edge. We simply use the "flip" probability p.

Furthermore, we consider the special case of $p = 0.5$.

0 ☐
1 ☐

a) What is the probability $\mathcal{P}(\tilde{\mathbf{x}}|\mathbf{x})$ for a directed graph or is there not enough information to determine it? Justify your answer.

0 ☐
1 ☐

b) We are given the prediction for the original input $f(\boldsymbol{X}) = c$ and for a slightly different version $f(\boldsymbol{X}') \neq c$.
What does the randomization scheme entail about the (exact) prediction of the smooth classifier $g(\boldsymbol{X})_c$ and $g(\boldsymbol{X}')_c$ (i.e. no Monte Carlo approximation)? Specifically, how do the probabilities for class c of the smooth classifier $g(\boldsymbol{X})_c$ and $g(\boldsymbol{X}')_c$ relate?

1. $g(\boldsymbol{X})_c = g(\boldsymbol{X}')_c$
2. $g(\boldsymbol{X})_c < g(\boldsymbol{X}')_c$
3. $g(\boldsymbol{X})_c > g(\boldsymbol{X}')_c$
4. There is not enough information to determine how $g(\boldsymbol{X})_c$ and $g(\boldsymbol{X}')_c$ relate

Justify your answer.

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

Professorship of Data Mining and Analytics
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:
- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Mining Massive Datasets

Exam:	IN2323 / Endterm	**Date:**	Friday 9th August, 2019
Examiner:	Prof. Dr. Stephan Günnemann	**Time:**	13:30 – 15:00

	P 1	P 2	P 3	P 4	P 5	P 6	P 7
I							

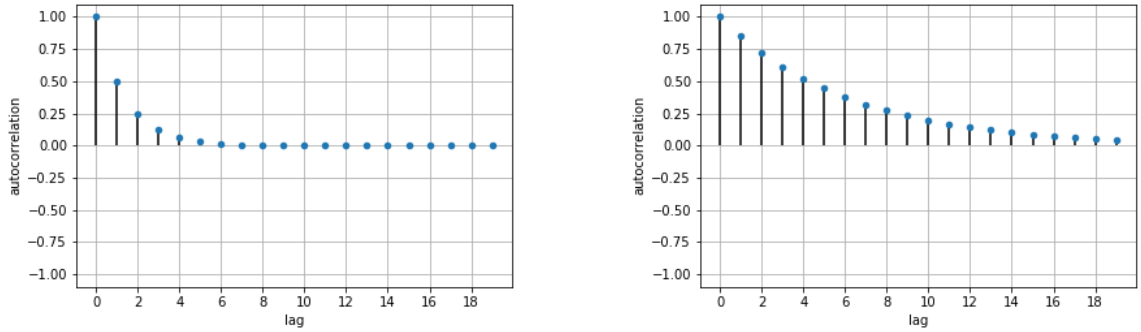
Working instructions

- This exam consists of **8 pages** with a total of **7 problems**.
  Please make sure that you received a complete copy of the exam.
- You can earn 38 points.
- **Detaching pages from the exam is prohibited!**
- Allowed resources:
  - A4 sheet of handwritten notes (two sides)
  - **no other materials (e.g. books, cell phones, calculators) are allowed!**
- Only write on the sheets given to you by supervisors. If you need more paper, ask the supervisors.
- Last two pages can be used as scratch paper.
- All sheets (including scratch paper) have to be returned at the end.
- **Only use a black or a blue pen (no pencils, red or green pens)!**
- Write your answers only in the provided solution boxes or the scratch paper.
- **For problems that say "Justify your answer" or "Show your work" you only get points if you provide a valid explanation.** Otherwise it's sufficient to only provide the correct answer.
- Exam duration - 90 minutes.

Left room from _____ to _____ / Early submission at _____

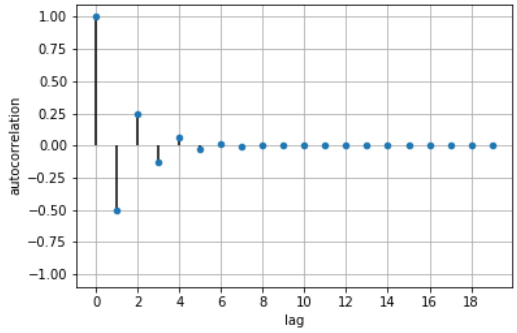
Problem 1 AR models: correlation function (4 points)

For each of the AR(1) and AR(2) models below find the corresponding autocorrelation function plot from Figure 1.1. Each plot was generated using one of the AR models from the list so that there is a one-to-one correspondence between the plots (1)-(4) and the AR processes (a)-(d). Everywhere $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ with a positive σ.

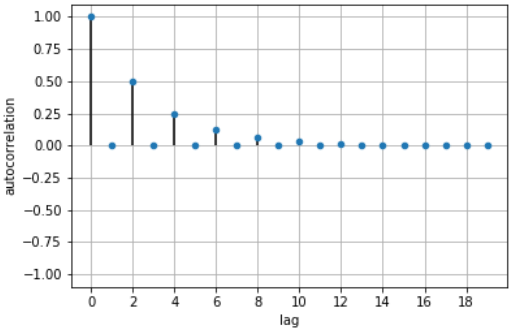


(1)

(2)



(3)



(4)

Figure 1.1: AR models: correlation function

a) $\mathcal{X}_t = -0.5\mathcal{X}_{t-1} + \epsilon_t$

☐ (1) ☐ (2) ☒ (3) ☐ (4)

b) $\mathcal{X}_t = 0.85\mathcal{X}_{t-1} + \epsilon_t$

☐ (1) ☒ (2) ☐ (3) ☐ (4)

c) $\mathcal{X}_t = 0.5\mathcal{X}_{t-2} + \epsilon_t$

☐ (1) ☐ (2) ☐ (3) ☒ (4)

d) $\mathcal{X}_t = 0.5\mathcal{X}_{t-1} + \epsilon_t$

☒ (1) ☐ (2) ☐ (3) ☐ (4)

Problem 2 **Hidden Markov Models (6 points)**

0 1 2 3 4 5 6

Consider the following Hidden Markov Model where Z_t are latent variables and X_t are continuous observed variables. We parametrize the prior and transition probabilities $P(Z_1 = i) = \pi_i$ and $P(Z_{t+1} = j|Z_t = i) = A_{ij}$ by:

$$\pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad A = \begin{bmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{bmatrix}$$

We parametrize the emission probabilities with $X_t|Z_t = 0 \sim Uniform([8, 12])$ and $X_t|Z_t = 1 \sim Uniform([11, 13])$. We assume we observed $X = [8.5, 11, 13, 9.5]$.

Perform the Forward algorithm, compute α_t and compute the most probable state Z_t given $X_1, ..., X_t$ at every step t.

Hint 1: You can check your computations by comparing with $\alpha_4 = \begin{bmatrix} 3/3200 \\ 0 \end{bmatrix}$

Hint 2: Maintaining non-decimal fractions makes calculations easier (e.g. use $8/10$ *instead of* 0.8*).*

$\pi = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$ $\quad Pr(x=8.5 \mid z_1=0) = \frac{1}{4}$

$x_1 = 8.5$ $\quad Pr(x=8.5 \mid z_1=1) = 0$

$\alpha_1 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \odot \begin{bmatrix} 1/4 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/8 \\ 0 \end{bmatrix}$

$x_2 = 11$ $\quad Pr(x=11 \mid z_2=0) = 1/4$

$Pr(x=11 \mid z_2=1) = 1/2$

$\alpha_2 = \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix} \odot \left(\begin{bmatrix} 0.8 & .2 \\ 0.5 & .5 \end{bmatrix} \begin{bmatrix} 1/8 \\ 0 \end{bmatrix} \right)$

$= \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix} \odot \begin{pmatrix} 1/10 \\ 1/40 \end{pmatrix} = \begin{bmatrix} 1/40 \\ 1/80 \end{bmatrix}$

$x_3 = 13$ $\quad Pr(x_3=13 \mid z_3=0) = 0$

$Pr(x_3=13 \mid z_3=1) = 1/2$

$\alpha_3 = \begin{bmatrix} 0 \\ 1/2 \end{bmatrix} \odot \begin{bmatrix} 4/5 & 1/2 \\ 1/5 & 1/2 \end{bmatrix} \begin{bmatrix} 1/40 \\ 1/80 \end{bmatrix}$

$= \begin{bmatrix} 0 \\ 9/1600 \end{bmatrix}$

$x_4 = 11$

$\alpha_4 = \begin{bmatrix} 9/12800 \\ 0 \end{bmatrix}$

Probable states

0, 0, 1, 0

Problem 3 Recurrent neural networks (8 points)

Given a sequence of integers $\{x_1, \ldots, x_n\}, x_t \in \{0, 1\}$, the goal is to output 1 at step t if the sum of all the inputs up to step t is even. That is, $h_t = 1$ if $\sum_{i=0}^{t} x_i$ is even, otherwise 0. For example, the outputs for a sequence 01101 are 10110. To model this you use an RNN with the following update equations:

$$\tilde{h}_t = a \cdot h_{t-1} + b \cdot x_t$$
$$z_t = f(c \cdot h_{t-1} + d \cdot x_t)$$
$$h_t = f((1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t)$$

where $h_0 = 1$ and $a, b, c, d \in \mathbb{R}$ and $f(x) = 1$ if $x > 0.5$, otherwise 0.

0 1 2

a) Fill out the table with all the possible input values x_t and input hidden state values h_{t-1}, and calculate the corresponding output h_t according to the given requirements.

sum upto t-1 → value added

h_{t-1}	x_t	h_t
0	0	0
0	1	1
1	0	1
1	1	0

0 1 2 3 4 5 6

b) Find the values for a, b, c and d such that this model achieves 100% accuracy. Show your work.
Hint: think about what can happen with a new input and what should gate z_t do in those cases.

$z_t = f(c \cdot h_{t-1} + d \cdot x_t)$.

we know $x_t = 0$ retains the input

$x_t = 1$ flips the input

$c = 0 \quad d = 1$

$z_t = f(x_t)$ just care whether $x_t = 1$ or 0

$z_t = 1 \Rightarrow$ past setting is forgotten

$h_t = f(\tilde{h}_t)$

$f(b) = 1$

$f(a) = 1$

$f(a+b) = 0$

Problem 4 **Graph laws (4 points)**

You are given an Erdős-Rényi graph $G(n, p)$, where n is the number of nodes and p is the probability of an edge.

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

0
1
2

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

b) We add a new node and connect it to every existing node with probablity q. What is the expected number of newly created triangles?

0
1
2

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

Problem 5 Deep Generative Models (6 points)

The loss used in generative adversarial networks (GANs) can be written in the following form:

$$\min_{\theta}\max_{\phi}\mathcal{L}(\theta,\phi)=\min_{\theta}\max_{\phi}\mathbb{E}_{p^*(\boldsymbol{x})}[\log D_\phi(\boldsymbol{x})]+\mathbb{E}_{p(\boldsymbol{z})}[\log(1-D_\phi(f_\theta(\boldsymbol{z})))]$$

where $p^*(\boldsymbol{x})$ is the true data distribution, $p(\boldsymbol{z})$ is the distribution of the noise, f_θ is the generator, and D_ϕ is the discriminator.

From sheet

0 1 2 3 a) For a given generator (fixed parameters θ) assume there exists a discriminator $D_{\phi^*}(\boldsymbol{x})$ with parameters ϕ^* such that for all $\boldsymbol{x}$:

$$D_{\phi^*}(\boldsymbol{x})=\frac{p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})+p_\theta(\boldsymbol{x})}$$

where $p_\theta(\boldsymbol{x})$ is the distribution learned by the generator. Show that D_{ϕ^*} is **optimal**, i.e. $\phi^*=\arg\max_\phi \mathcal{L}(\theta,\phi)$. Hint: $\max_y[a\log(y)+b\log(1-y)]=\frac{a}{a+b}$ for any $a,b\in\mathbb{R}_0^+, a+b>0$.

$$\max_\phi L(\theta,\phi)=\max_\phi \mathbb{E}_{p^*(x)}[\log D_\phi(x)]+\mathbb{E}_{p(z)}[\log(1-D_\phi(f_\theta(z)))]$$

$$=\max_\phi\left[\int (p^*(x)\log D_\phi(x))dx+\mathbb{E}_{p_\theta(x)}[\log(1-D_\phi(x))]\right]$$

$$=\max_\phi\left[\int (p^*(x)\log D_\phi(x)+p_\theta(x)\log(1-D_\phi(x))]dx\right.$$

($p^*(x)$: a, $D_\phi(x)$: y, $p_\theta(x)$: b, $1-D_\phi(x)$: $1-y$)

$$=\int\left[\max_\phi\left(p^*(x)\log D_\phi(x)+p_\theta(x)\log(1-D_\phi(x))\right]dx\right.$$

$$=\int\left(\frac{p^*(x)}{p^*(x)+p_\theta(x)}\right)dx$$

0 1 2 3 b) Show that $\mathcal{L}(\theta,\phi^*)=-\log(4)+2\cdot\mathbb{JSD}(p^*(\boldsymbol{x})||p_\theta(\boldsymbol{x}))$ where

$$\mathbb{JSD}(p^*(\boldsymbol{x})||p_\theta(\boldsymbol{x}))=\frac{1}{2}\left[\mathbb{KL}(p^*(\boldsymbol{x})||m(\boldsymbol{x}))+\mathbb{KL}(p_\theta(\boldsymbol{x})||m(\boldsymbol{x}))\right]$$

is the Jensen–Shannon divergence, $\mathbb{KL}$ is the Kullback–Leibler divergence, and $m(\boldsymbol{x})=\frac{p_\theta(\boldsymbol{x})+p^*(\boldsymbol{x})}{2}$.

$$L(\theta,\phi^*)=\mathbb{E}_{p^*(x)}[\log D_\phi(x)]+\mathbb{E}_{p(z)}[\log(1-D_\phi(f_\theta(z)))]$$

$$2\,JSD(p^*(x)\,||\,p_\theta(x))=\left[\mathbb{KL}(p^*(x)||m(x))+\mathbb{KL}(p_\theta(x)||m(x)\right]$$

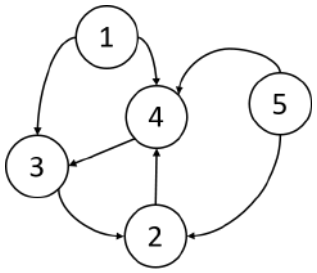
$$=\int p^*(x)\cdot\log\left(\frac{2p^*(x)}{p_\theta(x)+p^*(x)}\right)dx+p_\theta(x)\log\left(\frac{2p_\theta(x)}{p_\theta(x)+p^*(x)}\right)$$

$$=\int\frac{2p^*(x)}{2}\cdot dx+\int\frac{2p_\theta(x)}{2}dx+\cdots$$

$$=e-\log(4)+\cdots$$

Problem 6 Ranking (6 points)

Given the following graph G let $e = (s, t)$ be a new edge, i.e. an edge which is **not** yet in the graph G. We denote with G_{new} the resulting graph when adding e to G. Your task is to identify the edge e based on the following observation:
The PageRank vector of G_{new} based on a teleport set $S = \{1, 5\}$ and $\beta = 0.85$ (i.e. the probability to teleport is 0.15) is given by



$$\pi = [0.0750, 0.1935, 0.2668, 0.2764, 0.1884]$$

What is the edge e (specify the source node s and target node t) that has been added? Justify your answer.

0 1 2 3 4 5 6

$r_1 = \frac{0.1\pi_5}{|S|} = \frac{0.1\pi_5}{2} = 0.07\pi_5$ $\quad r_5 = 0.07\pi_5$.

does not match for node 5, hence node 5 is involved as target node.

now we need to determine the source node

for $1 \to 5$ $\quad \pi_5 = 0.07\pi_5 + \beta\frac{\pi_1}{2} = 0.0963$

for $2 \to 5$ $\quad \pi_5 = 0.07\pi_5 + \beta\frac{\pi_2}{2}$

for $3 \to 5$ ✓ matches.

Problem 7 Spectral clustering (4 points)

SBM

You are given a graph G with $N = 1000$ nodes that is generated from a Stochastic Block Model with the following parameters

$$\pi = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \qquad \eta = \begin{bmatrix} 0.05 & 0.5 \\ 0.5 & 0.05 \end{bmatrix}$$

Assume that G is connected. Do you expect spectral clustering to recover the true communities $\boldsymbol{z}$ when applied to the graph G (yes or no)? Justify your answer.

0 1 2 3 4

as off diagonal in η is more than leading diagonals there are more edges connecting the different communities

Spectral clustering minimizes the inter community edges → won't work.

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

Professorship of Data Mining and Analytics
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:
- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Mining Massive Datasets

Exam:	IN2323 / Retake	**Date:**	Monday 30th September, 2019
Examiner:	Prof. Dr. Stephan Günnemann	**Time:**	08:00 – 09:30

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8
I								

Working instructions

- This exam consists of **12 pages** with a total of **8 problems**.
  Please make sure that you received a complete copy of the exam.
- You can earn 43 points.
- **Detaching pages from the exam is prohibited!**
- Allowed resources:
  - A4 sheet of handwritten notes (two sides)
  - **no other materials (e.g. books, cell phones, calculators) are allowed!**
- Only write on the sheets given to you by supervisors. If you need more paper, ask the supervisors.
- Last two pages can be used as scratch paper.
- All sheets (including scratch paper) have to be returned at the end.
- **Only use a black or a blue pen (no pencils, red or green pens)!**
- Write your answers only in the provided solution boxes or the scratch paper.
- **For problems that say "Justify your answer" or "Show your work" you only get points if you provide a valid explanation.** Otherwise it's sufficient to only provide the correct answer.
- Exam duration - 90 minutes.

Left room from _____ to _____ / Early submission at _____

Problem 1 AR models: stationarity (5 points)

Decide whether the following AR models are stationary or not. Everywhere $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ with a positive σ.

Mark correct answers with a cross ☒
To undo a cross, completely fill out the answer option ■
To re-mark an option, use a human-readable marking ×■

a) $\mathcal{X}_t = c + 0.1\mathcal{X}_{t-1} + \epsilon_t$ with some $c \in \mathbb{R}$ $|\phi_1| < 1$

- ☐ stationary for $|c| > 0.1$, otherwise non-stationary
- ☒ yes, always stationary
- ☐ no, always non-stationary

b) $\mathcal{X}_t = -3 + 0.2\mathcal{X}_{t-1} - 0.01\mathcal{X}_{t-2} + c\epsilon_t$ with some $c \in \mathbb{R} \setminus \{0\}$ $\phi_1 = .2$ $\phi_2 = -0.01$ $\phi_1 + \phi_2 < 1$ $\phi_2 - \phi_1 < 1$ $|\phi_2| < 1$

- ☐ stationary for $c > 0$, otherwise non-stationary
- ☒ yes, always stationary
- ☐ no, always non-stationary

$\Phi(L) = c - \sum_{i=1}^{3} \phi_i L^i = 1 - .2L + 0.03L^2 + 0.001L^3$

$|1 - 0.1L^3| \rightarrow L = 10$ outside unit circle

c) $\mathcal{X}_t = 1 + 0.3\mathcal{X}_{t-1} - 0.03\mathcal{X}_{t-2} + 0.001\mathcal{X}_{t-3} + \epsilon_t$

- ☐ no, non-stationary
- ☒ yes, stationary

$\Phi(L) = -2 - 0.5L^n = 0$

d) $\mathcal{X}_t = -2 + 0.5\mathcal{X}_{t-n} + \epsilon_t$ with some $n \in \mathbb{N}$

- ☐ no, always non-stationary
- ☐ stationary for $n \leq 2$, otherwise non-stationary
- ☒ yes, always stationary

e) $\mathcal{X}_t = 2019 - \sum_{i=1}^{n} a^i \mathcal{X}_{t-i} + \epsilon_t$ with some $n \in \mathbb{N}$, $a \in \mathbb{R} \setminus \{0\}$ * const. can be ignored

- ☒ stationary for $|a| < 1$, otherwise non-stationary
- ☐ stationary for $|a|^n < 2019$, otherwise non-stationary
- ☐ stationary for $n = 1$, $|a| < 1$, otherwise non-stationary

$\Phi(L) = 2019 + \sum_{i=1}^{n} (aL)^i$

$2019 + \frac{1 - (aL)^{n+1}}{1 - aL}$ $|x| = \frac{1}{|x|}$ if $a \leq 1$

Problem 2 **Hidden Markov Models (7 points)**

Consider the following Hidden Markov Model where $Z_t \in \{0, 1\}$ are latent variables and $X_t \in \{a, b\}$ are discrete observed variables. We parametrize the prior and transition probabilities $P(Z_1 = i) = \pi_i$, $P(Z_{t+1} = j|Z_t = i) = A_{ij}$ and $P(X_t = j|Z_t = i) = B_{ij}$ by:

$$\pi = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \quad A = \begin{bmatrix} \alpha & 1-\alpha \\ 1/2 & 1/2 \end{bmatrix}, \quad B = \begin{bmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{bmatrix}$$

We assume we observed $X = [a, b, a]$.

a) Compute $P(Z_2|X_1, X_2)$ and $P(Z_2|X_1, X_2, X_3)$ as a function of α.

0
1
2
3
4
5

a)(i) Forward algorithm :

$$\alpha_1 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \odot \begin{bmatrix} 3/4 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 3/8 \\ 1/8 \end{bmatrix}$$

$$\alpha_2 = \begin{bmatrix} 1/4 \\ 3/4 \end{bmatrix} \odot \begin{bmatrix} \alpha & 1/2 \\ 1-\alpha & 1/2 \end{bmatrix} \begin{bmatrix} 3/8 \\ 1/8 \end{bmatrix} = \frac{1}{32} \begin{bmatrix} 3\alpha + 1/2 \\ 21/2 - 9\alpha \end{bmatrix}$$

(ii) Backward

$$\beta_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\beta_2 = \begin{bmatrix} \alpha & 1-\alpha \\ 1/2 & 1/2 \end{bmatrix} \left(\begin{bmatrix} 3/4 \\ 1/4 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} \alpha & 1-\alpha \\ 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 3/4 \\ 1/4 \end{bmatrix}$$

$$= \frac{1}{4} \begin{bmatrix} 2\alpha + 1 \\ 2 \end{bmatrix}$$

0
1
2

b) What is the most probable state Z_2 according to $P(Z_2|X_1, X_2)$ and $P(Z_2|X_1, X_2, X_3)$ for any value of $\alpha \in [0, 1]$.
Hint: $\sqrt{244} \approx 15.62$

Problem 3 RNNs & Word vectors (7 points)

You are solving a question-answering task. Given a context and a question, the goal is to find the answer **inside** the context. Bellow are two examples (1 and 2).

id	Context	Question	Answer
1	Mary was in the bathroom. Then she moved to the hallway.	Where is Mary?	hallway
2	John is in the hallway. Mary is there as well.	Where is Mary?	hallway

Assume that the question is represented with the vector $\boldsymbol{q}$. We want to know what is the probability that a word from the context is the answer. We decide to somehow represent every word w_i with an embedding $\boldsymbol{h}_i$ and pass it together with $\boldsymbol{q}$ through a neural network to get the probabilities. The only thing left to do is to decide how to get $\boldsymbol{h}_i$. We propose two approaches: sliding window and RNN.

a) **Sliding window** — Every word w_i is represented with a pretrained word vector $\boldsymbol{v}_i$. A sliding window of size 2 takes the neighbouring words and constructs the embedding for w_i as a sum of vectors: $\boldsymbol{h}_i = \sum_{j=i-2}^{i+2} \boldsymbol{v}_j$. Is it possible for this model to find the right answer in example 1? What about example 2? Justify.

☐ 0
☐ 1
☐ 2

b) **RNN** — As an alternative we use an RNN that takes pretrained word vectors $\boldsymbol{v}_i$ from left to right and outputs $\boldsymbol{h}_i$ as a word embedding. Why is this model able to output the right answer in example 1? Explain why it could fail answering example 2?

☐ 0
☐ 1
☐ 2

c) Propose another model that overcomes the shortcomings of the sliding window and the RNN. Describe what the input would be and how would you calculate the embedding $\boldsymbol{h}_i$. Explain why this model could work on both examples.

☐ 0
☐ 1
☐ 2
☐ 3

Problem 4 Deep Generative Model (5 points)

0 1 2 3

a) You are given a pseudo code implementation of 4 different variants of an AutoEncoder. Here, $\boldsymbol{x}_i \in \mathbb{R}^d$ is the input data, and $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}^k$ and $g_\phi : \mathbb{R}^k \mapsto \mathbb{R}^d$ are fully connected feed-forward neural networks where θ and ϕ are learnable parameters. The final layers of f_θ and g_ϕ have no (i.e. have linear) activation functions. $\boldsymbol{I}_k$ is a $k \times k$ identity matrix, $\mathbf{0}_k$ is a k-dimensional vector of zeros, $\mathcal{N}$ is the Normal distribution, σ is the softmax function, and $diag(\boldsymbol{x})$ takes a vector $\boldsymbol{x} \in \mathbb{R}^k$ and returns a $k \times k$ diagonal matrix with the vector values on the diagonal.

AutoEncoder 1

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{x}_i, \boldsymbol{I}_d)$$
$$\boldsymbol{h}_i = f_\theta(\boldsymbol{\epsilon}_i)$$
$$\tilde{\boldsymbol{x}}_i = g_\phi(\boldsymbol{h}_i)$$
$$\mathcal{L} = \sum_i \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2$$

AutoEncoder 2

$$\boldsymbol{h}_i = f_\theta(\boldsymbol{x}_i)$$
$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{h}_i, \boldsymbol{I}_k)$$
$$\tilde{\boldsymbol{x}}_i = g_\phi(\boldsymbol{\epsilon}_i)$$
$$\mathcal{L} = \sum_i \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2$$

AutoEncoder 3

$$\boldsymbol{h}_i = f_\theta(\boldsymbol{x}_i)$$
$$\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}_k, \boldsymbol{I}_k)$$
$$\tilde{\boldsymbol{x}}_i = g_\phi(\boldsymbol{h}_i + \boldsymbol{\epsilon}_i)$$
$$\mathcal{L} = \sum_i \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2$$

Is it **necessary** to use the reparametrization trick in order to compute the gradients of $\mathcal{L}$ w.r.t. to **both** θ and ϕ in the above implementations? Answer with Yes or No and provide a justification. If the answer is Yes, modify the pseudo code to implement the reparametrization trick.

AutoEncoder 1

reparametrization not required as sampling does not depend on parameters.

AutoEncoder 2

sampling $\epsilon_i \sim \mathcal{N}(h_i, I_k)$ $h_i = f_\theta(x_i)$ ∴ reparametrization is required

$\epsilon_i \sim \mathcal{N}(0, I_k)$

$\epsilon_i = h_i + \mathcal{N}(0, I_k)$

AutoEncoder 3

Not required

b) Assume the same setup as in a). The model specified by the following pseudo code is **not well defined**. Specify the reason why, and modify the pseudo code such that the model becomes well-defined. In addition, if you think it is **necessary** to use the reparametrization trick, please include it in your implementation.

0
1
2

$$
\begin{aligned}
\boldsymbol{h}_i &= f_\theta(\boldsymbol{x}_i) \\
\boldsymbol{\epsilon}_i &\sim \mathcal{N}(\mathbf{0}_k, diag(\boldsymbol{h}_i)) \\
\tilde{\boldsymbol{x}}_i &= g_\phi(\boldsymbol{\epsilon}_i) \\
\mathcal{L} &= \sum_i \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|_2
\end{aligned}
$$

Problem 5 Spectral clustering (3 points)

0 1 2 3

Given is the following matrix $\boldsymbol{M} \in \mathbb{R}^{9\times 9}$.

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Write down the **exact** value of the three smallest eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\boldsymbol{M}$ and the respective eigenvectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$.

2 connected components

$\therefore \lambda_1 = \lambda_2 = \lambda_3 = 0.$

Problem 6 Spectral clustering (3 points)

0 1 2 3

You are given an undirected graph $G = (V, E)$. It is known that the second smallest eigenvalue of the unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is equal to 10.
Let $\phi(G)$ denote the best possible ratio cut achievable on the graph G

$$\phi(G) = \min_{S \subset V} \text{ratio-cut}(S, \bar{S})$$

What are possible values of $\phi(G)$ for the given graph? Select all that apply. Justify your answer.

a) 1

b) 2

c) 4

d) 8

e) 16

$\lambda_2 = 10.$

relaxation of min ratio cut problem.

$\text{ratio-cut}(S,\bar{S}) \geq \lambda_2\left(1 - \frac{|S|}{|V|}\right)$

$\text{ratio-cut}(S,\bar{S}) \geq 10\left(1 - \frac{|S|}{|V|}\right)$

This should be ≥ 0

$10\left(-\right)$

Problem 7 Ranking (5 points)

Given a graph G with 5 nodes, assume that you have access to several topic-sensitive PageRank vectors, each pre-computed using a **different** teleport set S, and the **same (fixed)** teleport parameter β, $0 < \beta < 1$.

- $\pi_{235} \in \mathbb{R}^5$, with teleport set $S = \{2, 3, 5\}$
- $\pi_{124} \in \mathbb{R}^5$, with teleport set $S = \{1, 2, 4\}$
- $\pi_{134} \in \mathbb{R}^5$, with teleport set $S = \{1, 3, 4\}$
- $\pi_3 \in \mathbb{R}^5$, with teleport set $S = \{3\}$

Assume that the random walker always teleports **uniformly at random** to each node in the teleport set.

Is it possible to compute each of the following PageRank vectors without access to the graph G, i.e. using only the above pre-computed vectors? If so, specify the exact equation as a function of $\pi_{235}, \pi_{124}, \pi_{134}$ and π_3. If not, justify why not.

a) Is it possible to compute $\pi_{14} \in \mathbb{R}^5$ with teleport set $S = \{1, 4\}$?

0 1

$$\pi_{14} = \left(3 * \frac{\pi_{134} - \pi_3}{2}\right)$$

b) Is it possible to compute $\pi_5 \in \mathbb{R}^5$ with teleport set $S = \{5\}$?

0 1

$$\pi_{14} = 3 * \frac{\pi_{134} - \pi_3}{2} \qquad \pi_2 = 3 * \pi_{124} - 3 * \pi_{134} + \pi_3$$

$$\pi_5 = 3 * \pi_{235} - 3\pi * 124 + 3 * \pi_{134} - 2\pi_3$$

c) Is it possible to compute $\pi_1 \in \mathbb{R}^5$ with teleport set $S = \{1\}$?

0 1

No we can't

d) Is it possible to compute $\pi_w \in \mathbb{R}^5$ with teleport set $S = \{1, 2, 3, 4, 5\}$, where we do not teleport to each node uniformly at random but rather with weights 0.2, 0.3, 0.1, 0.2, 0.2, respectively?

0 1 2

Problem 8 Graph Neural Networks (8 points)

Given an unweighted, undirected graph G with adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$ and node attribute matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, your task is to perform semi-supervised node classification with C classes using a graph convolutional network (GCN).
In matrix notation, a GCN with K layers is recursively defined as follows.

$$\boldsymbol{H}^{(0)} = \boldsymbol{X}$$
$$\boldsymbol{H}^{(k)} = \sigma^{(k)}\left(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \qquad \text{for } k \in 1, ..., K$$

That is, $\boldsymbol{H}^{(K)} \in \mathbb{R}^{N \times C}$ contains the class predictions for **all nodes** stacked in a matrix. Here, $\sigma^{(k)}$ is the ReLU$(\cdot)$ activation function for $k \in \{1, ..., K-1\}$ and the softmax$(\cdot)$ function for the final (output) layer $k = K$. $\boldsymbol{W}^{(k)}$ is the weight matrix of layer k.
$\tilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ is a degree-normalized version of the adjacency matrix whose entries are

$$\tilde{\boldsymbol{A}}_{uv} = \begin{cases} \frac{1}{\sqrt{d_u d_v}} & \text{if } \boldsymbol{A}_{uv} = 1 \\ \frac{1}{d_u} & \text{if } u = v \\ 0 & \text{else,} \end{cases}$$

where d_u is the degree of node u.

0 1 2 3 a) GCN is an instance of the differentiable message passing framework. Provide the corresponding message function M and update function U of GCN. For simplicity, you may write $\sigma^{(k)}$ for all layers.

0 1 b) Assume you have a GCN with 3 layers, i.e. $K = 3$. Provide the non-recursive (unrolled) expression for $\boldsymbol{H}^{(3)}$. That is, write down the single-line equation of $\boldsymbol{H}^{(3)}$ in matrix form.

c) We consider now the two following situations:

0
1
2

(1) $\boldsymbol{A}_{uv} = 1$ for $u = v$ and 0 else.

(2) $\sigma^{(k)}(x) = x$, i.e. identity activation function, for $k \in \{1, 2\}$

Apart from the additional information in each situation, the models are identical to the GCN defined above. Simplify the single-line, matrix-form expression of $\boldsymbol{H}^{(3)}$ given the additional information provided in each situation.

d) For each situation, find one equivalent model in the table below. **You may select each option only once**. Briefly justify your answer for each situation.

0
1
2

Recurrent neural network (RNN)	Linear regression
Feed-forward neural network (FFNN)	Label Propagation (LP)
Linear function	Deep Generative Model
Logistic regression on $\boldsymbol{X}$	Logistic regression on pre-processed features $\tilde{\boldsymbol{X}}$

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

Mining Massive Datasets — Final Exam

1	2	3	4	5	6	7	8	9	10	11	$\sum$
/8	/10	/11	/10	/8	/4	/8	/6	/4	/4	/12	/48

Do not write anything above this line

Name:

Student ID: Signature:

- Only write on the sheets given to you by supervisors. If you need more paper, ask the supervisors.
- Pages 14-16 can be used as scratch paper.
- All sheets (including scratch paper) have to be returned at the end.
- **Do not unstaple the sheets!**
- Wherever answer boxes are provided, please write your answers in them.
- Please write your student ID (*Matrikelnummer*) on every sheet you hand in.
- **Only use a black or a blue pen (no pencils, red or green pens!).**
- You are allowed to use your A4 sheet of handwritten notes (two sides). **No other materials (e.g. books, cell phones, calculators) are allowed!**
- Exam duration - 90 minutes.
- This exam consists of 12 pages, 7 problems. You can earn 48 points.

1 Hidden Markov Models

Problem 1 [10 points] In this question, we will discuss hidden Markov models with **continuous** observations. We will use the notation from the lecture, where $Z_t \in \{1, ..., K\}$ denotes the state at time t and $X_t \in \mathbb{R}$ denotes the observation at time t. The conditional probability of an observation at a state k is $\Pr(X_t \mid Z_t = k, \boldsymbol{\theta}) = \mathcal{N}(X_t \mid \mu_k, \sigma_k^2)$, i.e. a Gaussian distribution parametrized by mean μ_k and variance σ_k^2. $\boldsymbol{\theta}$ is the set of parameters of the HMM, which includes the initial probabilities $\boldsymbol{\pi} \in \mathbb{R}^K$, transition probability matrix $\boldsymbol{A} \in \mathbb{R}^{K \times K}$ and the means and variances $\{\mu_1, ..., \mu_K, \sigma_1^2, ..., \sigma_K^2\}$.

a) Write down the log-likelihood $\log \Pr(Z_1, ..., Z_T, X_1, ..., X_T \mid \boldsymbol{A}, \boldsymbol{\pi}, \mu_1, ..., \mu_K, \sigma_1^2, ..., \sigma_K^2)$ as a function of $\boldsymbol{A}, \boldsymbol{\pi}, \mu_k$'s and σ_k^2's.

b) Write the forwards and backwards update equations for this HMM. Explain in a single line how they are different from the updates we studied in class (discrete observations).

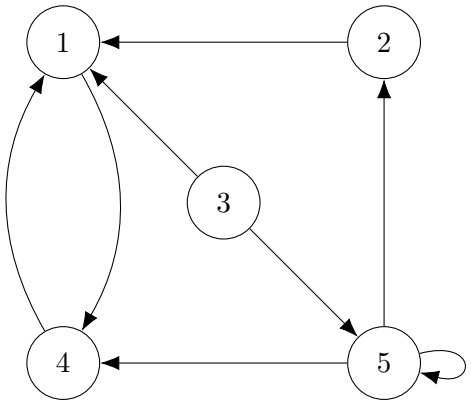
c) You are given a sequence of observations $\{X_1, ..., X_T\}$ and the corresponding states $\{Z_1, ..., Z_T\}$.

Are the maximum likelihood estimates of A_{ij} and π_i for this model different from the ones for HMM with discrete observations (that we studied in class)? Explain why or why not.

d) You are given a sequence of observations $\{X_1, ..., X_T\}$ and the corresponding states $\{Z_1, ..., Z_T\}$. Write down the closed-form maximum likelihood estimates for the parameters μ_k and σ_k^2. You don't have to derive the closed-form MLE, stating it with an intuitive explanation is enough.

2 PageRank

Problem 2 [4 points] Given the graph above and the topic-specific PageRank vector $\boldsymbol{r} = [0.3662, 0.0442, 0.0800, 0.3519, 0.1578]$ using the teleport set $S = \{3, 5\}$. What is the value of β (corresponding to the teleport probability $1 - \beta$)? Justify your answer.



3 Graph Clustering

Problem 3 [8 points] You are given a connected undirected unweighted graph with a set of nodes V. Let S_n^* be the partitioning minimizing the **normalized cut**, and let S_r^* be the partitioning minimizing the **ratio cut**. That is:

$$S_n^* = \arg\min_S \text{n-cut}(S) \qquad S_r^* = \arg\min_S \text{r-cut}(S)$$

where $S = \{C_1, V \backslash C_1\}$ is a two-way partitioning of the nodes in the graph.

Prove or disprove that $\text{n-cut}(S_n^*) \leq \text{r-cut}(S_r^*)$.

Hint: Think about the values of the cuts when we plug in the same partitioning S.

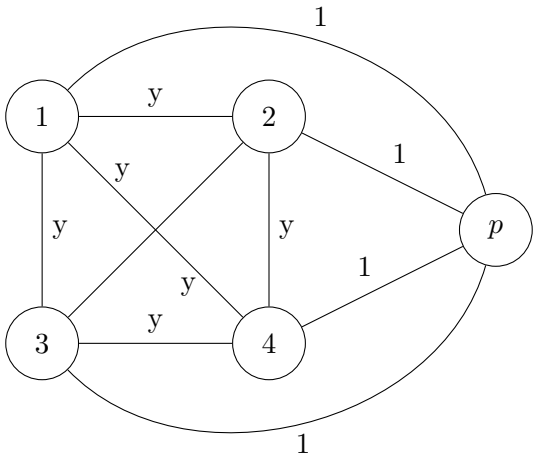
Furthermore, by definition we have that $\text{n-cut}(S_n^*) \leq \text{n-cut}(S)$. Combining these facts we obtain for any S:

$$\text{n-cut}(S_n^*) \leq \text{n-cut}(S) \leq \text{r-cut}(S)$$

Finally, replacing S with S_r^* above we obtain $\text{n-cut}(S_n^*) \leq \text{r-cut}(S_r^*)$.

You are given an undirected, weighted graph with a set of nodes $V = \{1, 2, \ldots, n, p\}$. It contains one clique of n nodes that is fully interconnected, each edge having weight y, and one additional node p that is connected to all nodes in the clique with weight 1.

See the example figure below for $n = 4$ as an illustration. We want to partition the nodes into **two** clusters.

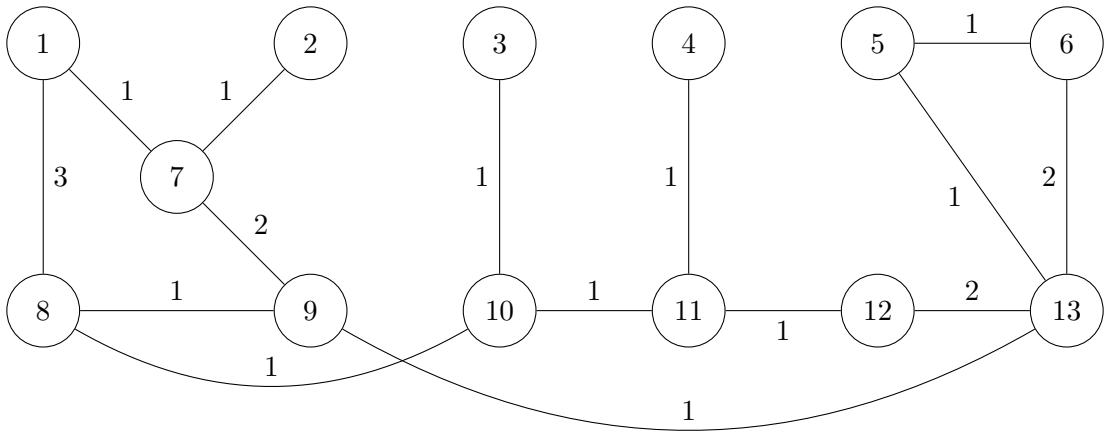


a) What is the ratio cut when one of the clusters contains only the node p? Provide the solution as a function of y and n.

b) What is the ratio cut if one of the clusters corresponds to $S \subseteq \{1, 2, \ldots, n\}$, with $|S| = k > 0$? Provide the solution as a function of y, n, and k.

c) When is the partitioning $\{p\}$ vs $\{1, 2, \ldots n\}$ the optimal solution for the minimum ratio cut? Specify all values of y and n, such that this holds.

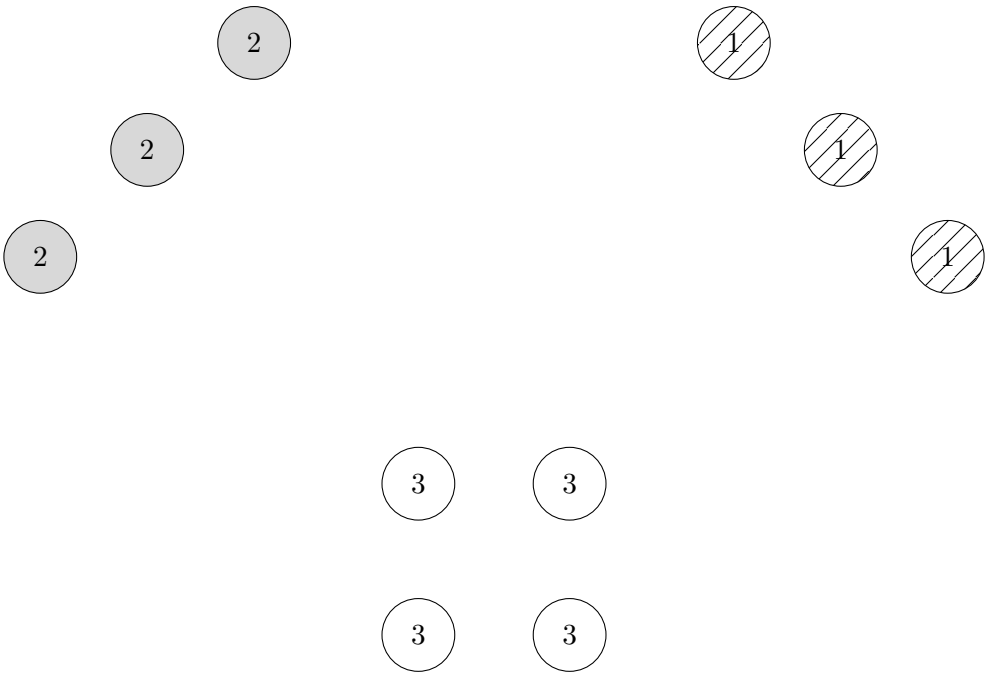
Problem 4 [6 points] Spectral embedding has been applied to the following undirected weighted graph



The plot below demonstrates the coordinates of the nodes in the embedding space (as defined by the second and the third eigenvectors of the unnormalized graph Laplacian).

Your task is to annotate the nodes in the plot below with their corresponding node IDs from the figure above.

Problem 5 [4 points] The graph below has been generated using the Stochastic block model with $K = 3$ communities (edges are hidden in the drawing). The community assignments of the 10 nodes in the graph are known (number inside each circle indicates the community ID z_i).



(a) Draw the edges in the graph such that performing maximum likelihood estimation of the parameter $\boldsymbol{\eta}$ will yield

$$\boldsymbol{\eta} = \begin{bmatrix} 1 & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

In case you draw some edges incorrectly and need to correct your solution, please redraw the entire graph from scratch on pages 14-16. Note that there are multiple correct solutions and specifying any of them is enough.

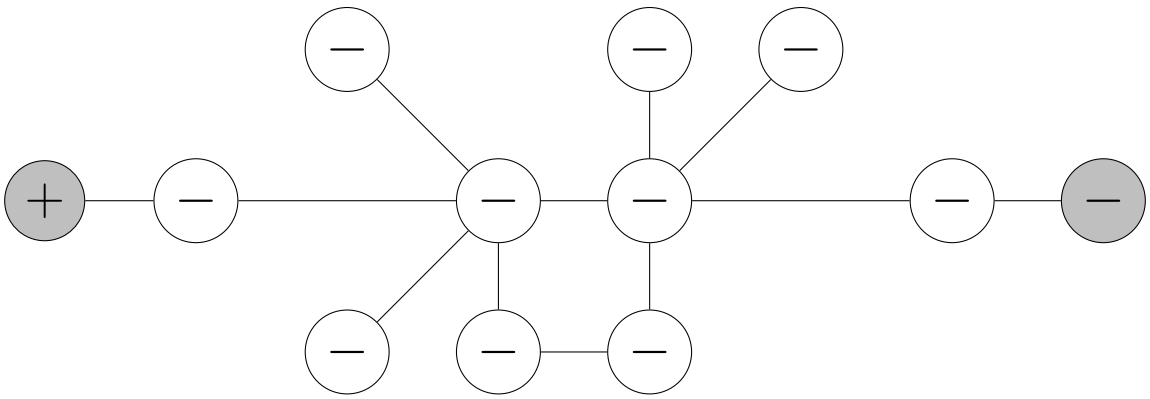
Provide a brief explanation below.

(b) What is the maximum likelihood estimate of $\boldsymbol{\pi}$ for the given graph and communities?

4 Label Propagation

Problem 6 [4 points] You are given the graph below with two labeled seed nodes (shaded), belonging to two classes: + and −. Specify the labels of the remaining nodes (by writing + or − inside each node) that would be obtained with an **exact** solution of the standard binary label propagation problem (i.e. assuming label smoothness).

Justify your answer. Multiple correct solutions are possible, it is enough to provide one.



5 Deep Learning on Graphs

Problem 7 [12 points] You are given an undirected unweighted graph G with N nodes. Your task is to instantiate the differentiable message passing framework to compute PageRank. More specifically you have to specify:

a) The input features $\boldsymbol{x}_v \in \mathbb{R}^2$ of each node v.

b) The function M that computes the message from node u to its neighbor v.

c) The function U that updates the hidden representation of a node v.

such that the hidden representations in the last layer approximately recover the PageRank score of the nodes in the graph G.

Hint 1: Increasing the number of layers in this formulation should improve the approximation of the PageRank scores. Hint 2: You do not need any trainable parameters.