Q1.MLP W = oldW + learning rate * label y(i) * data x(i). 这里 y 的值是 1or-1. Forward/back 如下

Example: 1 for "Y" and -1 for "N";

$\eta = 0.9$

x0	x1	x2	true label	w before update	predicted label	w after update
1	0	1	Y	(0.0, 0.0, 0.0)	N	(0.9, 0.0, 0.9)
1	1	1	N	(0.9, 0.0, 0.9)	Y	(0.0, -0.9, 0.0)
1	0	0	Y	(0.0, -0.9, 0.0)	N	(0.9, -0.9, 0.0)
1	1	0	Y	(0.9, -0.9, 0.0)	N	(1.8, 0.0, 0.0)

:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Q2 linear Reg(加 1to x for bias)

LogReg(Bernoulli 和 Poisson 的 MLE)

$P_X(x_i; p) = p^{x_i}(1-p)^{(1-x_i)}$, where $p \in [0,1], x_i \in \{0,1\}$.

$$\ln p_X(x_1, \cdots, x_n; p) = \sum_{i=1}^{n} x_i \ln p + \sum_{i=1}^{n}(1-x_i)\ln(1-p)$$

$$\frac{d \ln p_X}{dp} = \frac{\sum_{i=1}^{n} x_i}{p} + \frac{\sum_{i=1}^{n} x_i - n}{1-p} \overset{set}{=} 0$$

$$p_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$P_X(x_i; \lambda) = \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$$

$$\ln p_X(x_1, \cdots, x_n; \lambda) = \sum_{i=1}^{n} x_i \ln(\lambda) - n\lambda - \sum_{i=1}^{n} x_i!$$

$$\frac{d \ln p_X}{d\lambda} = \frac{\sum_{i=1}^{n} x_i}{\lambda} - n \overset{set}{=} 0$$

$$\lambda_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- Y: continuous value $(-\infty, +\infty)$
- $y = x^T\beta + \varepsilon = \beta_0 + x_1\beta_1 + x_2\beta$
- $\varepsilon \sim N(0, \sigma^2)$
- $y|x, \beta \sim N(x^T\beta, \sigma^2)$

$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ MSE

Naive Bayes MLE with Lagrangian Multiplier

Recall that given a corpus and labels for each document $D = \{(x_d, y_d)\}_{d=1}^{|D|}$, the log likelihood function of a Bayes model parameterized by $\Theta = (\beta_1, \beta_2, \beta_3, \pi)$ is:

Full unconstrained objective

$$L^*(\Theta) = L(\Theta) - \sum_j \lambda_j \left(\sum_{n=1}^{N} \beta_{j,n} - 1\right)$$

MLE derivation

$$\frac{\partial L^*(\Theta)}{\partial \beta_{j,n}} = \sum_{d:y_d=j} \beta_{j,n}^{-1} x_{dn} - \lambda_j \overset{set}{=} 0 \implies \beta_{jn} = \frac{\sum_{d:y_d=j} x_{dn}}{\lambda_j} \implies \lambda_j = \sum_{n=1}^{N}\sum_{d:y_d=j} x_{dn}$$

$$\beta_{j,n}^{-1} = \frac{\lambda_j}{\sum_{d:y_d=j} x_{dn}} \implies \sum_{n=1}^{N} \beta_{j,n} = 1 = \sum_{n=1}^{N}\sum_{d:y_d=j} x_{dn}$$

$$\implies \beta_{j,n} = \frac{\sum_{d:y_d=j} x_{dn}}{\sum_{n=1}^{N}\sum_{d:y_d=j} x_{dn}}$$

Step 4: get the full gradient: $\nabla_\beta L^*(\Theta)$ is a 3 by N matrix whose entries are $\beta_{j,n}$, $j \in \{1,2,3,\}, n \in \{1, \cdots, N\}$.

Recall the Dirichlet distribution, a family of continuous multivariate probability distribution used as prior to the multinomial distribution, with the PDF specified as:

$$f(\mathbf{x}; \alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

where $\Gamma(x)$ is the Gamma function with derivative $d\ln(\Gamma(x))/dx = \frac{\Gamma'(x)}{\Gamma(x)} = \psi(x)$, the digamma function. Derive the maximum likelihood estimator for parameter α for the Dirichlet distribution.

找 log-likelihood function

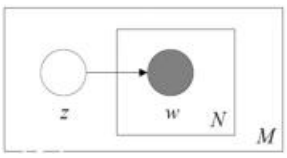
$$\ln f(x; \alpha) = \ln(\Gamma(\tilde{\alpha})) - \sum_{i=1}^{K} \ln \Gamma(\alpha_i) + \sum_{i=1}^{K} (\alpha_i - 1)\ln(x_i)$$

where $\tilde{\alpha} = \sum_{i=1}^{K} \alpha_i$. Now find its derivative w.r.t α_i and set it to zero to find an algebraic equation:

$$\frac{d \ln f(x;\alpha)}{d\alpha_i} = \psi(\tilde{\alpha})\frac{d\tilde{\alpha}}{d\alpha_i} - \frac{d}{d\alpha_i}\left(\sum_{j=1}^{K} \ln \Gamma(\alpha_j)\right) + \frac{d}{d\alpha_i}\left(\sum_{i=1}^{K} (\alpha_j - 1)\ln(x_i)\right)$$

$$= \psi\left(\sum_{i=1}^{K} \alpha_i\right) - \psi(\alpha_i) - \sum_{i \neq j} \ln(x_j) \overset{set}{=} 0$$

Then use a numerical scheme to find root. The full gradient is $\nabla_\alpha \ln f(x;\alpha) = [\frac{d \ln f(x;\alpha)}{d\alpha_1}, \cdots, \frac{d \ln f(x;\alpha)}{d\alpha_K}]$



Q3: N:word 数 M:topic 数.Generative process:先 sample 一个 topic 再产生 N words given that topic

Joint dis 和 conditional dis

$$p(w, z) = p(w|z)p(z)$$

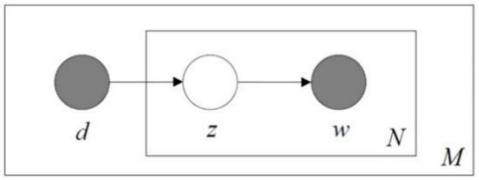
$$p(w = w_i, z = k) = p(w_i = w|z = k)p(z = k),$$

Joint distribution of N variables 和 entire one

$$p(w_{1:N}, z) = \prod_{n=1}^{N} p(w_n|z)p(z)$$

$$p(w_{1:N}, z_{1:M}) = \prod_{m=1}^{M} p(z_m) \prod_{n=1}^{N} p(w_n|z_m) \quad p(w_{1:N}) = \prod_{n=1}^{N} p(w_n) = \prod_{n=1}^{N} \sum_{z} p(w_n, z) = \prod_{n=1}^{N} \sum_{z} p(w_n|z)p(z)$$

pLSA model M doc, N word



Generative process:
- first sample a document from M documents
- For each document, generate N words, each with its own topic. (one Z responsible for 1 w's) e.g., 1-to-1 map between Z and W
- Note that one d is responsible for N z's and N w's. e.g., 1-to-N map between d and Z and between d to w.

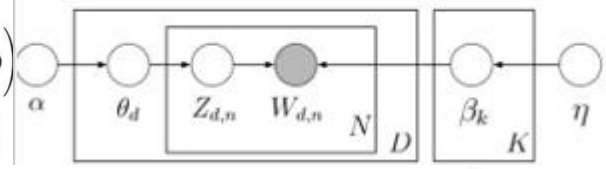
$$p(w, d, z) = p(w|z, d)p(z|d)p(d)$$

$$p(w_{1:N}, z_{1:N}, d) = \prod_{n=1}^{N} p(w_n|z_n, d)p(z_n|d)p(d)$$

$$p(w_{1:N}, z_{1:N}, d_{1:M}) = \prod_{m=1}^{M} p(d_m) \left(\prod_{n=1}^{N} p(w_n|z_n, d_m)p(z_n|d_m) \right)$$

$$p(w_{1:N}, d_{1:M}) = \prod_{m=1}^{M} p(d_m) \prod_{n=1}^{N} \left(\sum_{z} p(w_n|z_n, d_m)p(z_n|d_m) \right)$$

Bayesian pLSA



$$p(\theta, Z, W, \beta) = p(W|Z, \beta)p(Z|\theta)p(\theta)p(\beta) \quad p(\theta_{1:D}, Z_{1:N}, W_{1:N}, \beta_{1:K}) = \prod_{k=1}^{K} p(\beta_k) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(Z_{d,n}|\theta_d)p(W_{d,n}|Z_{d,n}, \beta_k) \right)$$

Derive the following quantities in the pLSA model and show all your steps: (1) The joint density of all random variables $p(w, d, z; \theta, \beta)$; (2) The conditional density $p(z|w, d; \theta, \beta)$. and (3) The conditional density of $p(w|z, d; \theta, \beta)$. (15 pts) (note in this notation, θ, β are given as model hyperparameters, hence they are separated from the random variable using ';')

(a) $p(w, d, z, ; \theta, \beta) = p(w|d, z)p(z|d)p(d)$.

(b) $p(z|w, d; \theta, \beta) = p(w, d, z)/p(w, d) = \frac{p(w|d,z)p(z|d)p(d)}{\sum_{z'} p(w|d,z')p(z'|d)p(d)}$

(c) $p(w|z, d; \theta, \beta)$ already given in lecture slide.

Write the log-likelihood function of the pLSA model as sum of two terms: the Evidence Lower Bound (ELBO) and the KL divergence between a variational distribution and the latent posterior. Point out each term in your answer. (10 pts)

Solution: The general ELBO-KL decomposition for a latent variable model with latent variable Z and data X can be written as:

$$\log p(X) = \mathbb{E}_q[\log p(X, Z) - \log q(Z)] - KL(q(Z)||p(Z|X))$$

In this problem we have $Z = \{z\}$ the topic, and $X = \{w, d\}$. Therefore, for the decomposition in this case we just plug in the values abstractly (this is enough for this problem):

$$\log p(w, d; \theta, \beta) = \mathbb{E}_q[\log p(w, d, z; \theta, \beta) - \log q(z|w, d; \phi)] - KL(q(z|w, d; \phi)||p(z|w, d))$$

Where $q(z|w, d; \phi)$ is a variational family of distribution parameterized by free parameter ϕ. If you attempted to derive it in more detailed form, you also get the full mark.

Use the *pdf kernel method* to derive the posterior distribution the following conjugate pair: prior $p(\theta) = \mathcal{N}(\theta; 0, I)$ and likelihood function $p(x; \theta) = \mathcal{N}(x; \mu, \Sigma)$, show all steps. (5 pts)

Solution: The PDF kernel method simply looks at the kernel part of the pdf. In this case we have:

$$p(\theta) = p(\mu) = N(\mu; 0, I) \propto \exp\{-\frac{1}{2}\mu^T\mu\}$$

$$p(x|\theta) = p(x|\mu) = N(x; \mu, \Sigma) \propto \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

$$p(\theta|x) = p(\mu|x) \propto p(\mu)p(x|\mu) \propto \exp\{-\frac{1}{2}(\mu^T\mu + x^T\Sigma x - \mu^T\Sigma^{-1}x - x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu)\}$$

$$\propto \exp\{(u-\hat{\mu})^T\hat{\Sigma}^{-1}(\mu-\hat{\mu}\}, \hat{\mu} = (\Sigma^{-1}+I)^{-1}\Sigma x, \hat{\Sigma} = (\Sigma^{-1}+I)$$

Given a set of binary outcomes (successes and failures) from independent Bernoulli trials, and assuming a Beta distribution as the prior for the probability of success p, derive the posterior distribution of p after observing the data.

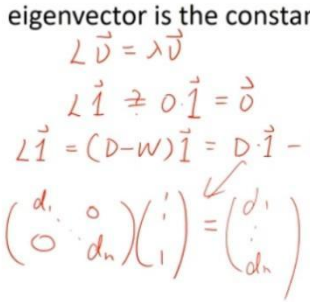
1. **Write down the prior distribution**:
   $p(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}$
   where $B(\alpha, \beta)$ is the beta function.
2. **Write down the likelihood for Bernoulli trials**:
   $p(x|p) = p^s(1-p)^{n-s}$
   where s is the number of successes and n is the number of trials.
3. **Write the posterior distribution**:
   The posterior is proportional to the product of the prior and the likelihood:
   $p(p|x) \propto p(p) \times p(x|p)$
4. **Combine the terms and simplify**:
   $p(p|x) \propto p^{\alpha+s-1}(1-p)^{\beta+n-s-1}$
5. **Recognize the kernel of a Beta distribution**:
   The expression matches the kernel of a Beta distribution with updated par:
   $p(p|x) = \text{Beta}(\alpha+s, \beta+n-s)$

Q4:Eigenval 0 的数量 in L=连接的 Component 数 quadratic form: $x^TLx = \sum_{(i,j)\in E}(x_i - x_j)^2$ normalized graph L:

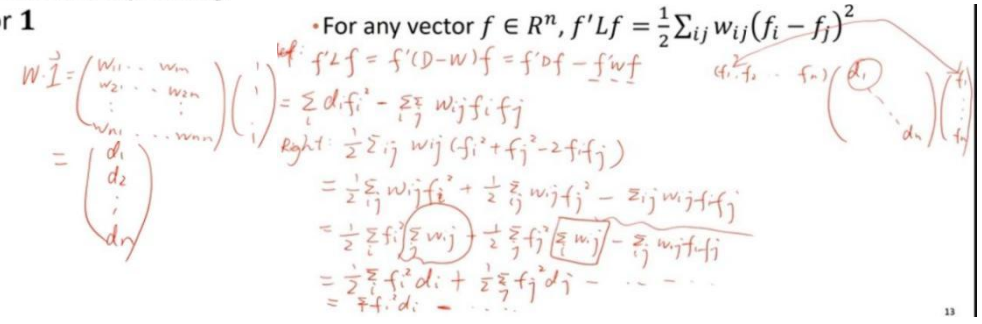
$$L_{\text{rw}} = D^{-1}L = I - D^{-1}W \quad L_{\text{sym}} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$$ Weighted graph: $x^TLx = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2$

Minimization of the Rayleigh quotient relies on the theorem

- The smallest eigenvalue of L is 0, and the corresponding eigenvector is the constant one vector $\mathbf{1}$



- For any vector $f \in R^n$, $f'Lf = \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2$



- L is symmetric and positive semi-definite
  - Undirected graph => symmetric
  - Property 1 => $f'Lf \geq 0$ => positive semi-definite, by definition

LogisReg: **概念：** 属于 **Discriminative** models(比如 logistic regression), model the conditional probability P(Y | X) directly. + 能用于 binary 和 multi-class 分类. 适合(tasks 有大量 labeled data 和 clear decision boundaries)
MSE 不适用(会有 non-convex optimiz 问题) + decision boundary 是 linear, 但可以用 non-linear transform of input features(比如 polynomial features) log reg 能捕捉 feature 间复杂关系(->flexible) + 相当于 1 层 NN + sigmoid(cross-entropy loss 训练). sigmoid activation 输出 prob(0~1) + 2 个 class linearly separable, scale up β变成 cβ, decision boundary 不变,但 likelihood func 会变,prob 更靠近 0 or 1,不会 converge, 最后 explode(所以需要 L1 or L2 regularization) + logistic func $\sigma(z)$ 代表 probability p(属于某 class) (e.g., spam), not log odds. The log odds would be the logarithm of odds ratio $\log(p/1-p)$ + k class 对应 k neuron,每个 neuron 用的 Softmax activation func,输出 prob 加起来是 1 + minimize CE loss = maximize log-likelihood func~~~~Bernoulli dis 用 Maximum Likelihood Estimation(MLE)来确定模型 param，通过 likelihood func 最大化观测到的数据属于其观测 class 的概率。对于 binary 问题, logistic reg 的输出是[0~1]间的一个概率值，表示属于 class 的概率.**Input**: p hat(predicted prob of ith observ belong to class1),观测到的 y 的概率可以用 Bernoulli dis 表示, θ是 model param, x 是 input data,y 是 observed output(target), binary 里 input 是个 neuron $P(y|x;\theta) = \hat{p}^y(1-\hat{p})^{1-y}$ likelihood func(观测值的概率的乘积,n 个 sample): $L(\theta|x,y) = \prod_{i=1}^{n} P(y_i|x_i;\theta)$

log-likelihood func(方便计算和优化，找最佳 param θ) $\ell(\theta|x,y) = \sum_{i=1}^{n}(y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i))$ **Output**: numerical measure of how well the model parameters explain the observed data!! ~~~~The output of the logistic function $\sigma(z)$ represents the log odds (i.e., $\log(\frac{p}{1-p})$), where p is the probability of an email being classified as spam, and $z = \theta^T x$.
对, logistic function (sigmoid function) outputs values 0~1,表示 prob of the positive class (prob email 被分类成 spam). The log odds transformation (logit function) maps these probabilities to the real number line, making it suitable for regression//Decision threshold is 0.5, then model 分类 email 是 spam(if log func ouput>0.5);反之，分类成不是 spam--对，binary classification(用 logistic reg),output=分类成 positive class 的概率(>0.5 的话就属于 positive class) **LinearReg**: 相当于 1 层 NNwith identity activation(no activation,因为 only pass input) (用 MSE loss 训练) + 不用 ReLU acti(会 introduce non-linearities) 需要最小化的 **Optimization func**:y 是真实值,yi hat 是 model 预测值,输出 RSS 和 MSE,公式在 page1 **Naive Bayes**, Bayes' threorem $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$ **概念：Generative models**, model joint probability distribution P(X, Y)--data X, labels Y , 用来产生 new data(by sample from distribution). + 能用于 binary 和 multi-class 分类 + 不属于 latent variable models,因为没 unobserved latent factors. 某个 word a 出现在 class c 的概率β = (word 在 class 出现的次数+1)/c 里的总 word 数+Vocabulary 里的 word 数 With add-1 smoothing 意思是在分子+1，分母+Vocabulary 里的 word 数
Π(某 class) = 属于那个 class 的 doc 数/总 doc 数

Data:

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Vocabulary:

Index	1	2	3	4	5	6
Word	Chinese	Beijing	Shanghai	Macao	Tokyo	Japar

- Learned parameters (with smoothing):

$\hat{\beta}_{c1} = \frac{5+1}{8+6} = \frac{3}{7}$ $\hat{\beta}_{j1} = \frac{1+1}{3+6} = \frac{2}{9}$
$\hat{\beta}_{c2} = \frac{1+1}{8+6} = \frac{1}{7}$ $\hat{\beta}_{j2} = \frac{0+1}{3+6} = \frac{1}{9}$
$\hat{\beta}_{c3} = \frac{1+1}{8+6} = \frac{1}{7}$ $\hat{\beta}_{j3} = \frac{0+1}{3+6} = \frac{1}{9}$
$\hat{\beta}_{c4} = \frac{1+1}{8+6} = \frac{1}{7}$ $\hat{\beta}_{j4} = \frac{0+1}{3+6} = \frac{1}{9}$
$\hat{\beta}_{c5} = \frac{0+1}{8+6} = \frac{1}{14}$ $\hat{\beta}_{j5} = \frac{1+1}{3+6} = \frac{2}{9}$

$\hat{\pi}_c = \frac{3}{4}$ $\hat{\pi}_j = \frac{1}{4}$

- For the test document d=5, compute

$p(y_5 = c|\boldsymbol{x}_5) \propto p(y_5 = c) \times \prod_n \beta_{cn}^{x_{5n}} = \frac{3}{4} \times \left(\frac{3}{7}\right)^3 \times \left(\frac{1}{14}\right) \times \left(\frac{1}{14}\right) \approx 0.0003$

$p(y_5 = j|\boldsymbol{x}_5) \propto p(y_5 = j) \times \prod_n \beta_{jn}^{x_{5n}} = \frac{1}{4} \times \left(\frac{2}{9}\right)^3 \times \left(\frac{2}{9}\right) \times \left(\frac{2}{9}\right) \approx 0.0001$

所以 x5 属于 class c。3/7 因为 doc5 里的 Chinese 的 beta 是 3/7,3 次方因为出现在 Doc5 里出现 3 次, 两个 1/14 因为 Tokyo 和 Japan 的 beta 都是 1/14.**Attributes are conditionally independent given class(class conditional independency)** **Word Embedding** **Skip-gram**:用 target word predict 旁边 word,更好得处理罕见(rare)words,计算效率比 CBOW(平均 context's embedding)低,但对大数据量更 effective 而且能捕捉更 detailed semantic 关系(更高 semantic accuracy,因为预测了很多 context words). **问题**:word 多计算贵(要对每个词评分和 nomal)->用 **Negative Sampling** 把 multi classification 变成 binary classification set(从预测 context 位置->对于 positve sample(actual context word)选 negative sample(随机词,不是 target 的上下文),再最大化 positve sample 的 score, 最小化 negative 的 **Transformer 和 LLM(概念)**Encode(represent)+Decode(Generation).**Input**(Tokenizer-embedd 层(word2vector)-positional encode)-**Encoding**(Self-Attention)-**Decoding**(Self-Attention)-**Output**(Linear Layer+Softmax). En/Decode 步有 add 和 norm 加和 norm 层. Special token EOS 加在末尾,之后开始 decode,但序列 len 太长不行,用 attention 把 input 里的子集 word 去预测 target 而不是全用+不用把序列变成 vector, 计算查询与一系列 key 之间的相似度或关联度决定对应 value 的加权重要性.**BERT:**only Encoder, 随机把 input token 变成 mask,每个 word attain 两侧所有 word,双向 context.**GPT :only Encoder**, pre-train 预测 next token,从左到右每个 word attain 之前的那一个 **Graph Spectral**

Clustering 用 eigenvalue 和 eigenvector 分组 data. node=data,edge=similarity. 算 **Laplacian** 矩阵的前 k 个 eigenvector,每个节点用 k 个 val 表示，用 eigenval 降维，再用如 K-means 来 cluster. 适合复杂 shape 的 data 以及 connectivity or graph structures 更 informative for clustering than distance in the original feature space.**Laplacian** 矩阵 L=D(diagonal matrix,每个点是 node 的度,除了对角线都是 0)-W(adjcency matrix). L 最小的 eigenVal 是 0(其他都大于 0),对应的 eigenVec 是 vector1(all-one vector,n dimension,能用 nonzero 数 scale)

For any vector $f \in R^n$, $f'Lf = \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2$ 对角 matrix L 有 k 个 eigenval 0,eigvectors:(1,1,0,0)(0,1,1,0) **Label Propagatio** 图中两节点紧密连接->很可能属于同一 class. 少数 node 有 class,算法通过图结构把 class 从 labeled 传到 unlabeled node.Laplacian 定义 propagation rule**K-Means** 适合 dataset 的(clusters 要球形+有相似 size 和 density), 不同 initial(可以随机)导致不同结果 + 不是通过 gradient descent 优化 objective func，而是 iteration 分配点和更新 centroid + k 的选择影响 result + converge when 簇心变化小于某值 or 最大迭代次数. 属于 hard classification,一个点属于一个类 **GMM** 不能保证 convergence to global optimal value. + 难估计 cluster 数+cluster 需要很多 param 去 specify+ 有多个 class 所以多个 distribution + dataset cluster 可以非球形和变化 densities,基于 EM 优化 param, 属于 latent variable models + 属于 soft Classificaiton(方差平方=covariance=0 后 soft assign 变成 hard assign) + GMM=soft K-means + covariance matrix 决定分布的形状(方向,宽度) + Joint probability density function(pdf) of marginal distribution of entire dataset under(GMM):

$$f_X(x_1, \cdots, x_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)$$

$f_X(x_1, \ldots, x_n)$: This represents the joint probability density function of the dataset X, where X consists of N data points $(x_1, \ldots, x_n)$ in a multidimensional space (e.g., $\mathbb{R}^d$).

$\prod_{n=1}^{N}$: This is the product operator, indicating that the expression to the right is multiplied together for each data point x_n from 1 to N, where N is the total number of data points in the dataset. This reflects the assumption of independence among the data points in terms of their generation process.

$\sum_{k=1}^{K}$: This is the summation operator, summing over the K components (or clusters) in the mixture model. It represents the mixture aspect of the model, where each data point is

π_k: These are the mixing coefficients (or mixture weights) for each of the K Gaussian components in the mixture model. Each π_k represents the prior probability that a randomly selected data point belongs to cluster k. The mixing coefficients must satisfy two conditions: $0 \leq \pi_k \leq 1$ for all k, and $\sum_{k=1}^{K} \pi_k = 1$, ensuring that they form a valid probability distribution over the clusters.

$\mathcal{N}(x_n; \mu_k, \Sigma_k)$: This denotes the probability density function of a Gaussian (or normal) distribution for the k^{th} component, evaluated at data point x_n. Each Gaussian component is characterized by:

- μ_k: The mean vector of the k^{th} Gaussian component, indicating the center of the cluster in the data space.
- Σ_k: The covariance matrix of the k^{th} Gaussian component, defining the shape and orientation of the cluster. The covariance matrix determines how spread out the cluster is in each dimension and the correlations between dimensions.

$$P(x|\gamma_1) = \frac{1}{\sqrt{2\pi}\sigma_1}\exp(-\frac{(x-\mu_1)^2}{2\sigma_1^2})$$
$$P(x|\gamma_2) = \frac{1}{\sqrt{2\pi}\sigma_2}\exp(-\frac{(x-\mu_2)^2}{2\sigma_2^2})$$

Mutinomial Mixture model Doc=多个 topic，topic=multinomial distribution，描述该主题下每个词出现的概率分布。生成 doc 步骤: 随机选个主题，从该主题的词汇分布生成单词。likelihood func

For each document d
- Sample its cluster label $z \sim Categorical(\boldsymbol{\pi})$
  - $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$, π_k is the proportion of jth cluster
  - $p(z = k) = \pi_k$
- Sample its word vector $\boldsymbol{x}_d \sim multinomial(\boldsymbol{\beta}_z)$
  - $\boldsymbol{\beta}_z = (\beta_{z1}, \beta_{z2}, \ldots, \beta_{zN})$, β_{zn} is the parameter associate with nth word in the vo
  - $p(\boldsymbol{x}_d|z = k) = \frac{(\sum_n x_{dn})!}{\prod_n x_{dn}!}\prod_n \beta_{kn}^{x_{dn}} \propto \prod_n \beta_{kn}^{x_{dn}}$

$$= \prod_d^d \frac{(\sum_n^k x_{dn})!}{\prod_n x_{dn}!} \sum_k p(z = k) \prod_n \beta_{kn}^{x_{dn}}$$

pLSA

Probabilistic Latent Semantic Analysis **概念:**latent variable 是 topic,word 被 latent topic 产生 + 不用 Dirichlet dis + models 每个 doc as mixture of topics(topics are distributions over words) + 不是 generative model,不能 model new doc(因为 pLSA 分配一组 topic distribution param 到每个 doc(在 train set)->param 很难 assign 到 unseen doc(without retrain). LDA 通过 add priors to θ和β变成 generative model 来解决:model doc as random mixtures over latent

- Probability of a word w

$$p(w|d, \theta, \beta) = \sum_k p(w, z = k|d, \theta, \beta) = \sum_k p(w|z = k, d, \theta, \beta)p(z = k|d, \theta, \beta) = \sum_k \beta_{kw}\theta_{dk}$$

- Likelihood of a corpus

$$\prod_{d=1} P(w_1, \cdots, w_{N_d}, d|\theta, \beta, \pi)$$
$$= \prod_{d=1} P(d)\left\{\prod_{n=1}^{N_d}\left(\sum_k P(z_n = k|d, \theta_d)P(w_n|\beta_k)\right)\right\}$$
$$= \prod_{d=1} \pi_d\left\{\prod_{n=1}^{N_d}\left(\sum_k \theta_{dk}\beta_{kw_n}\right)\right\}$$

π_d is usually considered as uniform, i.e., 1/M

topic where the mixture weight drawn from Dirichlet dis)

likelihood func, EM 估计文档-主题分布和主题-词汇分布的参数:通过 Maximize likelihood func 找最可能生成观察到的文本数据的模型参数.E 步:根据参数估计算每个词属于每个主题的概率。M 步更新模型参数来最大化，基于 E 步得到的条件概率分配

LDA 概念： 1.Dirichlet dis + distributions of topics over documents and words over topics 2.能 model unseen doc by model distri of doc. doc=mixtures of topics,topics =mixtures of words. doc 里的 topic dis 用 conjugate pair(简化 infer hidden structure) of Dirichlet dis model(允许 model infer topic dis for doc unseen during train)3.不用 EM(因为 posterior dis 的复杂性),infer 和 learn 用 Variational EM,Variational Bayes,MC(Gibbs Sampling)**EM** 知道 data point 属于哪个 Gaussin distribution,每个 dis 的 Mixing Coefficients 的和=1, GMM 是 unsupervised 分类模型，没有用点的类别学习模型 param. E 步预测每个点属于不同类别的概率(Implicit variable),计算 tight original objective func 的 lower bound at θold;M 步用估计的分类(Implicit variable)来更新 Gaussion 分布的 mean,方差和先验概率来 maximum 数据的 likelihood func,maxinmize lower bound at θnew。用 jensen inequality 找 lower bound. 不同的 initial paramθ会有不同 result，难优化，迭代次数多，容易到 local optim + k-means 属于 EM 特例 **Deep Learning, CNN** Relu 正区间内保持 gradient 恒定，缓解梯度消失问题,input<0 时梯度为 0,可能导致部分神经元不再更新(die). Sigmoid 输出 0~1 适用 binary 的输出层.Tanh 输出-1~1，比 Sigmoid 有更宽的输出范围.后两个 func 在 input 较大或较小时梯度接近 0 可能梯度消失,过大会 gradient explode. **1.**backpropagation 时候 error(loss) 从输出层到第一层,更新 weight 去最小化 loss func **2.**CNN 里 Pooling 层减小 representation 的 spatial size(宽高)和共享 weight 来减小 param 数(不太会 overfitting)而且 contributes to model's invariance to small translations of input image. **3.**convolutional 层作用是通过 filter(检测 patterns or texture)从 input image 学 features representation **4.**CNN 里 Softmax 输出层用 non-linear activation func 确保 output values are distributed(ouput sum=1),适合 multi-class classification **5**.CNN 通过堆叠多个卷积和池化层来增加网络的深度，使得网络能够学习从简单到复杂的特征层次结构 **Graph Embedding(LINE)**降维到 vector 表示(limit:high dimension, No global structure info integrated,permutaion-variant 输入变输出不变),1st order,节点相连则临近性=1,object func 最小化 empirical link dis 和 modeled link dis 的 KL divergence. 2nd:邻居相似则两节点相似(negative sampling 优化 learning efficiency).

GNN weakness： large graphs 导致计算和内存要求高导致 scalability 不好 + 不高效 capture long-range dependencies within graphs because the information needs to propagate over many layers for distant nodes. + sensitive to structure of input graph, making them less robust to changes or noise in the graph's topology. + 选 hyperparameters complex

GAT 用 attention weight nodes' neighbors 重要性, 不用 fixed-weight neighborhood aggregation. 只有 GAT 用 attention, 用来算 edge 的 weight based on node features, 导致更准的 nodes 代表 based on 邻居特征。不是 optimizing computational efficiency through fixed weights. GAT 和 GraphSAGE 都支持 inductive learning(model's ability to generalize from seen to unseen data->predictions on nodes not present during training. GAT achieve by leverage attention on new nodes and their neighbors.

GCN: **Message passing scheme=**邻居和自己的平均 weight(=normalized inverse of the nodes' degrees)更新这个点的 feature(通过用 spectral-based convolution: leverages graph structure encoded in the graph Laplacian matrix or an approximation of it.)

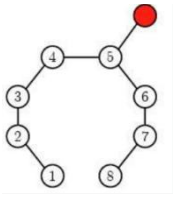
概念：不支持 inductive learning,支持 transductive learning，因为是从 entire graph 学, 包括 labeled+unlabeled nodes, 预测 nodes 也在一样的 graph。+ GCNs 不把 CNN filters 用在 graph structure as if the graph were a regular grid in Euclidean space. 而是用 convolution operation to work on the irregular structure of graphs. + Dynamic routing 更关联 Capsule Networks 而不是 GCNs, which do not base their convolution operations on path lengths or node centrality measures. + [不对]GCNs leverage an eigendecomposition of the graph Laplacian to apply convolution in the spectral domain, requiring computation of the Laplacian for every convolution layer.因为 modern variants 不需要 eigendecomposition for each layer + GCNs aggregate and transform neighbor node features through learnable weights, effectively capturing local graph topology by averaging features from a node's immediate neighbors.

GCN vs. GAT on link prediction tasks 取决于 graph data 和 task. GCNs aggregate neighborhood information based on a normalized sum of neighbor features, 不能总是 capture nuanced relationships between nodes as effectively as GAT's attention-based approach. GCNs 不用 global pooling functions for aggregation; this is a characteristic that can be added to many types of GNNs, including GCNs and GATs, depending on model architecture and task requirements.

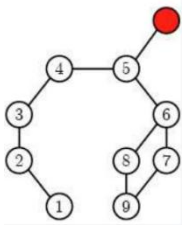
GraphSAGE: Unlike GCNs, GraphSAGE samples a fixed number 邻居而不用 entire 邻居，再 aggregates 他们 features.能用很多 functions 来 aggregation(比如 mean, LSTM, or pooling). 主要是要 generate embeddings by sample and aggregate features from a node's local neighborhood, allowing it to scale to large graphs and support inductive learning, where the model can generalize to unseen nodes.

(2 points) Given the applications of Graph Neural Networks (GNNs) for various tasks, identify which of the following statement regarding training objectives is not correct. Use $\boldsymbol{u}_i$ to denote embedding of node i, y_i for the label of node i, $\boldsymbol{g}$ for the graph embedding, y for the graph label, and $\boldsymbol{A}_{ij} = 1$ to indicate the presence of an edge between nodes i and j and 0 otherwise.

For link prediction, the objective is to maximize the likelihood of correctly predicting Aij based on the embeddings ui and uj of the two involved nodes. + For graph classification, the training objective is to minimize the cross entropy between the predicted probability of graph belonging to each class, which is based on graph-level embedding, and their ground truth class label. + For graph similarity search, the training objective is to minimize the graph-embedding based similarity measure and the ground truth similarity measure between graphs. + in node classification tasks, the training objective involves 最小化 loss between predicted labels (来自 embeddings of nodes by a classification layer, 不是直接来自 embeddings themselves) and the true labels of the nodes. The process 包括 apply a softmax function (or similar) to the node embeddings to get predicted probabilities, then compared to the actual labels using cross-entropy loss.



Consider the 2 graphs in the following figure, where all nodes have a 1-dimensional initial feature vector $x = [1]$. We use a simplified version of Graph Neural Networks (GNNs), with no nonlinearity, no learned linear transformation, and sum aggregation. Specifically, at every layer, the embedding of node v is updated as the sum over the embeddings of its neighbors ($\mathcal{N}(v)$) and its current embedding $\boldsymbol{h}_v^{(t)}$ to get $\boldsymbol{h}_v^{(t+1)}$. We run the GNN to compute node embeddings for the 2 red nodes respectively. Note that the 2 red nodes have different 5-hop neighborhood structure (this is not the minimum number of hops for which the neighborhood structure of the 2 nodes differs). How many layers of message passing are needed so that these 2 nodes can be distinguished (i.e., have different GNN embeddings)? Explain your answer in a few sentences.



To distinguish the two red nodes and ensure they have different GNN embeddings, we need more than k layers of message passing, where k is the minimum number of hops in their neighborhood structure that differ. denote the number of layers of message passing needed to distinguish the two red nodes as k. After k layers, the difference in the embeddings of the two red nodes will be at most k, which means that they will have the same embedding. **Explanation**: Since the GNN model uses a simplified version with no nonlinearity, no learned linear transformation, and sum aggregation, the embedding of each node at each layer is updated by summing the embeddings of its neighbors and its current embedding. To analyze the propagation of information through the graph, let's denote the embedding of node v at layer t as $h_v^{(t)}$ and the set of neighbors of node v as N(v). In each layer of message passing, the embedding of a node v is updated as follows:

$$h_v^{(t+1)} = h_v^{(t)} + sum\left(h_u^{(t)}\right) for\ u\ in\ N(v)$$

Start with an initial feature vector = [1] for all nodes. Therefore, the initial embedding $h_v^{(0)}$ for all nodes, including the red nodes will be [1]. To distinguish the two red nodes, their embeddings need to differ in at least one dimension. Since the initial feature vector is 1-dimensional, the embeddings of the two red nodes will differ by the sum of the values from their respective 5-hop neighborhood structures.

In this problem we explore two approaches to shallow embedding: LINE and Random Walk based embedding methods (DeepWalk). Recall that the central theme to define embedding vectors is to define a measurement of node similarity.

• In the LINE formulation, two nodes are similar if: (a) they are connected, also known as first-order proximity, or (b) their neighbors are similar, also known as second-order proximity.

• In the case of DeepWalk, two nodes are similar if they co-occur frequently in random walks. This random walk procedure is summarized as: given a starting point, randomly select a neighbor node, move to this neighbor, then repeat the process until the generated random path reaches a pre-determined length.

Suppose we want to perform node embedding for a graph $G = (V, E)$ using these two algorithms.

(a) For node embedding using LINE with first-order proximity, what's the time complexity of calculating all pairs of node similarities? How about second-order proximity? Suppose we now use a negative sampling scheme for $K << |V|$ negative samples, what's the time complexity in this case? (10 pts)

(b) Reason about why second-order proximity is required on top of first-order proximity. You may find the discussion in the original paper fruitful. (5 pts)

(c) Read the note from CS224W about Random walk-based approach and reason about comparison with LINE; in particular, the advantanges and disadvantages potentially associated with this approach. (10 pts).

(d) Reason (shortly) about the relation between Node2Vec and Word2Ve, from the perspective of sequence vs. graph data. (5 pts).

Q 1(a):

For node embedding using LINE with first-order proximity, the time complexity of calculating all pairs of node similarities is

$$O(|V|)$$

where $|V|$ is the number of nodes in the graph. This is because need to iterate through all nodes in the graph to calculate their similarities based on the first-order proximity.

For second-order proximity in LINE, the time complexity of calculating all pairs of node similarities depends on the average node degree in the graph. Let's denote the average node degree as d. In this case, the time complexity is approximately

$$O(d * |V|)$$

as that need to consider the neighbors of each node to calculate their similarities based on the second-order proximity.

Suppose a negative sampling scheme for $K << |V|$ negative samples in LINE. The time complexity remains

$$O(d * |V|)$$

because we still need to iterate through the neighbors of each node to calculate their similarities. The negative sampling scheme does not affect the time complexity in this case.

Q 1(b):

Second-order proximity is required on top of first-order proximity in the LINE algorithm to capture higher-order structural information and better reflect the node similarities in the graph. While first-order proximity considers direct connections between nodes, second-order proximity takes into account the similarity of their neighbors.

The inclusion of second-order proximity helps address the limitations of first-order proximity in capturing more complex relationships and structural patterns in the graph. By considering the similarities of neighboring nodes, the algorithm can capture the notion that nodes with similar neighbors are likely to have similar roles or functions in the graph.

TransE RotatE

2 Knowledge Graph Embedding (15 pts)

In this problem, you are going to consider the TransE, DistMult, and RotatE models (slide09 P23-30).

You are going to answer the following questions and provide reasons.

(a) If we have a knowledge graph with friendship and enemy relationship, which model(s) of the TransE, DistMult, and RotatE can we use? Please explain your reason based on the score function of each model. (Hint: Friendship and enemy are symmetric relationships.) (5 pts)

(b) If we have a knowledge graph with father, grandfather, mother, and grandmother relationship, which model(s) can we use? Please explain your reason based on the score function. (Hint: The father of father is grandfather. The mother of mother is grandmother. Which model(s) can model composition relationship? How?) (5 pts)

(c) For each of TransE, DistMult, RotateE, provide an example (different from part (a) and (b)) for a scenario where it cannot model the particular relationship. (5 pts)

Q 2(a)

TransE and RotatE models are suitable for modeling friendship and enemy relationships in a knowledge graph due to their ability to capture symmetric relationships.

TransE uses a score function based on the L1 or L2 distance between the translated head entity and the tail entity, combined with the relation vector. Since friendship and enemy relationships are symmetric, the relation vector would be the same but in opposite directions. The L1 or L2 distance metric in TransE is insensitive to the direction of the relation vector, making it suitable for modeling symmetric relationships.

RotatE, on the other hand, is designed to handle complex-valued embeddings and uses rotation operations in the complex plane to model relations. The scoring function in RotatE calculates the plausibility of a triple based on the rotation angle between the embeddings of the head and tail entities. For friendship and enemy relationships, the rotation angle would be the same but with opposite signs. RotatE's ability to capture rotational patterns makes it well-suited for modeling symmetric relationships.

In contrast, the DistMult model may not be the best choice for modeling symmetric relationships. DistMult uses a simple element-wise multiplication as the scoring function, which lacks the flexibility to differentiate between opposite relations. As a result, DistMult would treat friendship and enemy relationships as identical because their embeddings would have the same magnitude and direction when multiplied element-wise.

Therefore, considering the symmetry of friendship and enemy relationships, both TransE and RotatE models are more appropriate choices compared to DistMult.

Q 2(b)

For a knowledge graph with father, grandfather, mother, and grandmother relationships, we can use the TransE and RotatE models. Both models have the ability to capture composition relationships, which are necessary to represent the hierarchical nature of father, grandfather, mother, and grandmother relationships.

TransE: TransE can model composition relationships by using the addition operation in its scoring function. For example, if we have the triples (father, x, y) and (father, y, z), TransE can infer the triple (grandfather, x, z) by adding the relation vectors of the two father relationships. Similarly, it can infer other composition relationships such as (mother, x, z) and (grandmother, x, z) by adding the corresponding relation vectors. TransE can capture these hierarchical dependencies through the addition operation in its scoring function.

RotatE: RotatE can also model composition relationships by utilizing the rotational patterns in the complex plane. For instance, if we have the triples (father, x, y) and (father, y, z), RotatE can infer the triple (grandfather, x, z) by rotating the relation vector of the father relationship. Similarly, it can infer other composition relationships such as (mother, x, z) and (grandmother, x, z) by applying the appropriate rotations to the relation vectors. RotatE's ability to perform rotations in the complex plane allows it to capture composition relationships in the knowledge graph.

Both TransE and RotatE can effectively model the composition relationships in the given knowledge graph, enabling the inference of higher-level relationships such as grandfather, grandmother, and so on

Q 2(c)

Here are examples

TransE: TransE may struggle to model relationships that involve many-to-many mappings or complex patterns. For example, consider a knowledge graph with a "siblings" relationship. TransE represents relations as vector translations, but it may have difficulty capturing the complexity of sibling relationships, where multiple entities can have a shared sibling. The model's simple translation-based approach may not be able to capture the nuances of such relationships.

DistMult: DistMult may face challenges in modeling relationships that require more expressive interactions between entities and relations. For instance, consider a knowledge graph with a "works-with" relationship that represents collaboration between individuals. DistMult's scoring function, which involves element-wise multiplication, assumes a simple and symmetric interaction pattern. However, the "works-with" relationship may involve more complex and context-dependent interactions that DistMult may not be able to fully capture.

RotatE: RotatE may struggle to model relationships that are inherently asymmetric or anti-symmetric. For example, consider a knowledge graph with an "opposes" relationship that represents opposing viewpoints or ideologies. RotatE's rotational patterns in the complex plane are based on symmetry, which may not align well with the asymmetry or anti-symmetry present in the "opposes" relationship. The model's reliance on complex rotations may not be suitable for accurately representing such relationships.

TransE:1 对 1,1 对多,多对 1 的关系都可以, 但复杂的多对多不好.计算有效，适合大数据集，依靠 distance metrics 导致区别 embedding space 里近距离的 entity 难 **RotatE:**能用很多 symmetric/antisymmetric, inversion, one-to-one, one-to-many, many-to-one, and many-to-many. 用复杂 numbers and rotation in embedding space 使得它 model relation patterns with more nuance 细微差别. Despite its versatility, 不适合 irregular or complex relation patterns.计算量大，不合适大数据集，很复杂 **DistMult** 计算简单，所以适合大的 knowledge graph 通过 element-wise multiplication of embeddings 处理 symmetric relationships 但不是最好的 . 不好:antisymmetric relationships, direction-specific relationships, many-to-one, one-to-many, or

many-to-many relationships due to its inherent symmetry in modeling relations. **TransE:**represents relationships by translating vectors,1 对 1 关系可以，但多对多关系，比如 GrandparentOf 就不行(a head entity 关联于 with 很多 tails and 多个 heads 关联一个 tail entity.) 因为 TransE tries to maintain a fixed distance (translation) between entities in the embedding space, which becomes challenging when an entity participates in multiple relations of the same type but with different entities.**DistMult's** scoring function is symmetric, 这限制了准确 model asymmetric relations. The symmetry implies relation "A related to B" be scored 一样 as "B related to A," which not desirable for asymmetric relationships. RotatE is designed to model various relation patterns 包括 symmetry/antisymmetry, inversion, and composition effectively using complex numbers 但还是有限制

Negative Sampling 用于训练 TransE and RotatE，DistMult. 作用是去 differentiate between true and false (or negative) relations, 不管 relations 是不是 symmetric.

三个模型的计算：

TransE models relationships as translations in the embedding space. For a given triplet (h, r, t), where h is the head entity, r is the relation, and t is the tail entity, TransE assumes $h + r \approx t$ for true triplets. The goal is to learn embeddings such that the head entity vector h added to the relation vector r gets close to the tail entity vector t. The loss function, often using either the L1 or L2 norm, minimizes the distance for positive triplets while ensuring that the distance for negative triplets (incorrect relationships) is larger:

$L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \max(0, \gamma + ||h + r - t|| - ||h' + r - t'||)$

Here, S is the set of positive triplets, S' is a set of negative triplets generated by corrupting positive ones, and γ is a margin hyperparameter.

RotatE models relations by interpreting them as rotations in the complex vector space. For each triplet (h, r, t), RotatE posits:

$h \circ r \approx t$

where $\circ$ denotes the Hadamard (element-wise) product, and h and t are the complex embeddings of the head and tail entities, respectively. r indicates the rotation in the complex plane, with the magnitude and phase of r representing the relation's characteristics. RotatE's loss function is designed as:

$L = -\sum_{(h,r,t)\in S} \log \sigma(f(h, r, t)) - \sum_{(h',r,t')\in S'} \log(1 - \sigma(f(h', r, t')))$

where σ is the sigmoid function, and $f(h, r, t) = \mathrm{Re}(\mathbf{h} \circ \bar{\mathbf{r}} - \mathbf{t})$ calculates the score of a triplet, with Re extracting the real part and $\bar{\mathbf{r}}$ denoting the conjugate of the complex relation vector r.

Each model employs a unique computation method that reflects different intuitions and capabilities for modeling the types of relationships present in data. Choosing the right model often depends on the nature of the data and the specific types of relationships that need to be captured.

DistMult represents each relation as a diagonal matrix, facilitating the modeling of symmetric relationships. For a triplet (h, r, t), DistMult employs the scoring function:

$f(h, r, t) = \mathbf{h}^\top \mathbf{R} \mathbf{t}$

where $\mathbf{h}$ and $\mathbf{t}$ are the vector embeddings of the head and tail entities, respectively, and $\mathbf{R}$ is the diagonal matrix representing relation r. The loss function typically involves negative sampling or cross-entropy to discriminate between positive and negative triplets.

Approach	Scoring function
TransE	$f(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = -\lVert \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} \rVert$
DistMult	$f(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = \langle \boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \rangle$
RotatE	$f(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = -\lVert \boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t} \rVert$, where $\circ$ denotes the element-wise product and $\boldsymbol{r}$ is represented in complex space to enable rotation

Recommendation:

Treat all missing values as 0 can misrepresent data, assuming no interaction to be explicit negative feedback, 所以不是 implicit feedback recommendation 有效的方法. + **BPR** 目的是排序 user's observed interactions over unobserved ones, using a pairwise ranking loss function. + **Implicit matrix factorization** 目的不是 reconstruct original user-item matrix exactly 而是 predict unobserved preferences or rankings. + AUC 能用于 implicit feedback scenarios 通过考虑 task as a binary classification problem (interaction vs. no interaction). Precision@k 也能用于 implicit feedback scenarios 通过定义 interactions as positive instances and non-interactions as negative, though its interpretation might be less straightforward than in explicit feedback settings.

Collaborative Filtering: 假设用户过去和未来喜好一样，分 Memory-based CF 和 Model-based(更快). **Memory-based** 分 **User-based CF**(用来找邻居，计算 user 和 active user 的相似性，用相似 user 的 rating 作为 predict, 大数据集计算大因为要算所有用户 pair 的相似性)和 Item-based CF(计算 item 间相似性，预测 active user rate 过的相似 item 有相似 rating). **Model-based(比如 Matrix Factorization)**:Decomposes the user-item matrix into lower-dimensional latent factors, capturing underlying patterns of interactions.+ based on offline pre-processing or model-learning phase + at run time, only the learned model is used to make prediction + models are updated 周期性 + large variety of techniques used + model build 和 update 的计算很贵。**缺点**：There needs to be enough other users already in the system(cold start) + 如果 item 或者 user 很多，matrix 会 sparse,导致很难找到用户 that have rated the same item + Cannot recommend 之前没被 rate 的 item(new item,Esoteric item) + 不能推荐 item 给有独特 taste 的人(趋向于推荐热门 item) + latent factors might not have a clear interpretation **优点**：model train 了之后，产生 predict 是计算效率高的，即使在大数据集+ 可以 deal with sparse datasets by infer missing entries in the user-item interaction matrix.**Explicit Feedback**:**优点**:直接反应 user's preferences, 数据好的话准, **缺点**：feedback 可能不足 + 会有 biases 比如 user rating inflation or central tendency bias. **Implicit Feedback**:**优点**：Easier to collect at large scales, such as clicks, views, or purchase history. + Reflects the natural interaction of users with the system, without requiring explicit input.**缺点**: Harder to interpret, as actions like viewing an item don't always indicate positive preference.+ Noisy, not all interactions are meaningful indicators of preference.**区别**:explicit feedback is direct user ratings or preferences, whereas implicit feedback is derived from user behavior.**Content-based Recommendation: 优点** No Cold Start for Items: Can recommend new items based on their content features, independent of user interaction. + Specificity: Good at recommending items similar to those the user has liked before.**缺点**:lack of diversity in recommendations, as it focuses on similarity to user's past preferences. + Requires effective methods for extracting and utilizing content features.**Special Features**:Utilizes descriptive attributes of items (e.g., genre, author) to make recommendations, making it less reliant on user interaction data. **Hybrid Method** 好处：Combines the strengths of multiple recommendation approaches, potentially mitigating their individual weaknesses. + higher accuracy and better user satisfaction by integrating different sources of information.**缺点**: complex than single-method approaches. + 会 overfitting 如果没有 regularization.**Special Features**:包含 collaborative filtering, content-based filtering, more robust system. Each of these algorithms has its place depending on the application, available data, and specific requirements of the recommendation system. Hybrid methods, in particular, offer a flexible approach to leverage the strengths of individual methods while addressing their limitations.

Deep neural networks regularization: Add additional weight penalty terms in loss func(L-1 and L-2 norm of weights.)to constrain the size of weight to prevent overfit. + Dropout some weights when forward computation during training -> less sensitive to 一些 features ->减小 overfit + early stop(在 validation set 不再 improve 时候停止 train，防止 learn noise in the train data.) + GD 和 SGD 不算，因为他们是去更新参数最小化 loss func

Problem 11: Mixture Models for Binary Data (20 points)

Suppose we have a dataset of black and white images. The inputs $\boldsymbol{x}^{(i)}$ for the ith image are vectors of binary values corresponding to black and white pixel values, and the goal is to cluster the images into groups. For simplicity, you can use the answer from previous parts of the problem to later parts (e.g., use the answer in Part 1 for Part 2, etc.) For this problem, you simply need to consider a slight change to the Gaussian mixture model.

1. **(5 points)** Consider a binary random vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_D) \in \{0,1\}^D$. What is a choice of probability distribution to model the observed pixel value for each dimension of this random vector? (e.g., model $P(x_d) = 1$, $P(x_d = 0)$ for $d \in \{1, \cdots, D\}$)
2. **(5 points)** Using the proposed probability distribution above, suppose that $P(x_d = 1) = p_d$, where x_d is the random variable for the dth dimension of $\boldsymbol{x}$. Assume that random variables at each dimension are independent with each other (i.e., x_d is independent of x'_d if $d \neq d'$); write down the expression for $P(x_d)$ and $P(\boldsymbol{x})$ using $p_1, \ldots, p_D$.
3. **(5 points)** Suppose that there are K clusters, where kth cluster is associated with distribution described in Part 2 with parameter $\boldsymbol{p}_k$ and p_{kd} denotes the d_{th} dimension in $\boldsymbol{p}_k$. The prior probability each data point belonging to each cluster is $p(z = k) = \pi_k$. Write down the expression for $P(\boldsymbol{x}^{(i)})$ following the mixture model assumption.

   (**Hint:** Compare it with the Gaussian mixture model and consider $P(\boldsymbol{x}^{(i)} \mid z = k)$ and $p(z = k)$.)
4. **(5 points)** Suppose that the dataset is generated i.i.d. $X = \{\boldsymbol{x}^{(i)}\}_{i=1,\ldots,n}$. Write down the expression for the log-likelihood of the dataset. You can use the solution in Part 3.

Part B 11.1.

Choice the Bernoulli distribution, the Bernoulli distribution is a discrete probability distribution that models a binary outcome, which can take on two possible values, typically denoted as 0 and 1.

Part B 11.2.

Assume independence between the random variables at each dimension, the joint probability can be obtained by multiplying the marginal probabilities together. Therefore, the joint probability can be expressed as:

$$P(x) = P(x_1, x_2, \ldots, x_D) = P(x_1)P(x_2) \ldots P(x_D)$$

Substituting the expressions for the marginal probabilities, we have:

$$P(x) = (p_1)^{x_1}(1-p_1)^{1-x_1}(p_2)^{x_2}(1-p_2)^{1-x_2} \ldots (p_D)^{x_D}(1-p_D)^{1-x_D}$$

Part B 11.3.

The $P(x^{(i)})$ can be written by considering the individual cluster probabilities $P(x^{(i)}|z=k)$ and the prior probabilities $p(z=k)$.

Following the analogy with Gaussian mixture models, The $P(x^{(i)})$ can be can be expressed as a weighted sum of the probabilities of $x^{(i)}$ conditioned on each cluster z = k. This can be written as:

$$P(x^{(i)}) = \sum_{k=1}^{K} P(x^{(i)} \mid z=k) \cdot p(z=k)$$

In the the binary data, $P(x^{(i)}|z=k)$ corresponds to the probability of observing the binary vector $x^{(i)}$ given that it belongs to cluster k. The top probability using the Bernoulli distribution is

$$P(x^{(i)} \mid z=k) = \prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{1-x_d^{(i)}}$$

p_{kd} represents the success probability of the dth dimension in cluster k. The product is taken over all dimensions d in the binary vector $x^{(i)}$. $p(z=k)$ represents the probability of data point $x^{(i)}$ belonging to cluster k. It is denoted as $\pi_k$and can be seen as the mixing proportion or the weight assigned to cluster k in the mixture model. Substituting the expressions for $P(x^{(i)} \mid z=k)$ and $p(z=k)$ into the expression for $P(x^{(i)})$,

$$P(x^{(i)}) = \sum_{k=1}^{K} \left[\prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{1-x_d^{(i)}} \right] \cdot \pi_k$$

Part B 11.4

Since the data points in the dataset are assumed to be independently and identically distributed, the likelihood of the dataset can be expressed as the product of the individual data point probabilities.

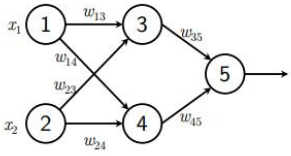
Taking the logarithm of this likelihood gives us the log-likelihood.

The log-likelihood of the dataset is:

$$\log \mathcal{L}(X) = \sum_{i=1}^{n} \log P(x^{(i)}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \left[\prod_{d=1}^{D} (p_{kd})^{x_d^{(i)}} \cdot (1-p_{kd})^{1-x_d^{(i)}}\right] \cdot \pi_k\right)$$

Problem 12: Neural Network (26 points)

As a data scientist working for a hospital, you want to use neural network to assess the health of patients. Consider the fully-connected neural network given below, where input x_1, x_2 are real numbers, the output activation function is the sigmoid function and all the rest activation functions are ReLU.



The weights and biases are given in the following table.

w_{13}	w_{14}	w_{23}	w_{24}	w_{35}	w_{45}	b_3	b_4	b_5
4	−3	4	−3	5	5	−2	5	−5

1. **(6 points)** Compute the model output when the input data is $(x_1, x_2) = (1, 1)$. You can leave the sigmoid function expression without evaluating the numerical value.
2. **(5 points)** What is the kind of supervised learning task this neural network is able to perform? State the reason.
3. **(5 points)** Suppose you want to use this neural network to predict the Cholesterol level (a categorical variable of $K > 2$ categories) of patients given a tabular datasets of real-valued features, what are the modifications you need to make to the network?
4. **(5 points)** Suppose you want to predict the Height (a real value) of patients. What are the modifications you need to make to the network? Which loss function do you like to use?
5. **(5 points)** A medical exam for the brain requires processing of MRI scans, which are images with potential regions of brain tumor. To determine if a user has a tumor, what are the possible modifications you can make to the current network? You are allowed to make major changes.

Part B 12.1

The results is: $x_3 = relu(x_1w_{13} + x_1w_{14})$, $x_4 = relu(x_2w_{23} + x_2w_{24}), x_5 = relu(x_3w_{35} + x_4w_{45})$

$$output = Sigmoid(x_5)$$

Part B 12.2

binary classification. Te reason for this is the output activation function, which is the sigmoid function. The sigmoid function is commonly used for binary classification tasks because it maps the network's output to a value between 0 and 1, representing the probability of belonging to the positive class. By setting an appropriate threshold (e.g., 0.5), we can classify the input into one of the two classes based on the output of the sigmoid function.

Part B 12.3

The following are modifications:

Output Layer Activation Function: Change the activation function of the output layer to a softmax function. The softmax function is commonly used for multiclass classification problems as it produces a probability distribution across all the possible classes. Each output neuron in the softmax layer represents the probability of the input belonging to a specific class.

Output Layer Neurons: Increase the number of neurons in the output layer to K, where K is the number of categories or classes in the Cholesterol level variable. Each neuron in the output layer corresponds to a specific class.

Loss Function: Change the loss function to a suitable loss function for multiclass classification, such as categorical cross-entropy. The categorical cross-entropy loss measures the dissimilarity between the predicted probability distribution and the true class labels.

Part B 12.5

MRI scans are images. The Convolutional Neural Network (CNN) Architecture: CNNs are well-suited for processing image data. We can replace the fully-connected neural network with a CNN architecture, which consists of convolutional layers, pooling layers, and fully-connected layers. The convolutional layers to extract high-level features from the MRI scans. Insert pooling layers (e.g., max pooling or average pooling) after convolutional layers to downsample the feature maps and reduce spatial dimensions. Modify the output layer based on the specific requirements. For binary classification, use a sigmoid activation function and have a single output neuron. For multiclass classification, use softmax activation with multiple output neurons.

Part B 12.4

The following are modifications:

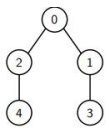
Output Layer Activation Function: Remove the sigmoid activation function from the output layer since it is suitable for binary classification. keep the output layer without any activation function, allowing it to output a continuous value. utput Layer Neurons: Keep a single neuron in the output layer corresponding to the predicted Height value.

Loss Function: The Height is a regression to predict a real value. The loss function for regression is the mean squared error (MSE), which calculates the average squared difference between the predicted and actual Height values.

Problem 13: Label Propagation (10 points)

Consider the following graph G, which consists of 5 nodes labeled from 0 to 4.

1. **(4 points)** Show the adjacency matrix of the graph G.
2. **(6 points)** Using a simple label propagation algorithm, where each unlabeled node adopts the label of the majority of its neighbors (or remains unlabeled if there is a tie), determine the labels of all nodes after one iteration of label propagation.



Part B13.1

the adjacency matrix for the given graph G:

	Node 0	Node1	Node 2	Node 3	Node 4
Node 0	0	1	1	0	0
Node 1	1	0	0	1	0
Node 2	1	0	0	0	1
Node 3	0	1	0	0	0
Node 4	0	0	1	0	0

Initialize the labels for each node:

Node 0: Label A

Node 1: Unlabeled

Node 2: Label B

Node 3: Unlabeled

Node 4: Unlabeled

Iterate through each unlabeled node and update its label based on the majority label of its neighbors.

Node 1: The majority label among its neighbors is A.

Node 3: The majority label among its neighbors is A.

Node 4: The majority label among its neighbors is B.

After the iteration, the updated labels for each node are:

Node 0: Label A

Node 1: Label A

Node 2: Label B

Node 3: Label A

Node 4: Label B

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	5		3		4
User 2	3	3	1	5	2
User 3		5	5	2	
User 4	3	5	1		1
User 5			5	2	5
User 6	5	3	?	5	

In this problem, 2 decimal places are sufficient for your answer, or you may give exact expressions.

1. **(4 points)** You wish to predict the rating of Item 3 by User 6. Suppose we use user-user collaborative filtering and use Pearson correlation as the similarity metric. What is the Pearson correlation coefficient between User 2 and User 6?
2. **(4 points)** Which other users could be used to help predict the rating of Item 3 by User 6?
3. **(4 points)** One problem with pure collaborative filtering is newcomers who have not rated anything. This is where the content-based approach comes in handy. Suppose we have a new user 7 with profile features $[1, 3, 0, 2, -2]$. We also know that the profile features for Item 3 are $[-1, 0, 2, 2, 0]$. What is the cosine similarity between User 7 and Item 3? You should give the value of the cosine itself, rather than the angle with that cosine.

Part B 14.1

Given the ratings:
User 2: [3, 3, 1, 5, 2]
User 6: [5, 3, ?, 5, x]
First, we need to calculate the mean rating for each item:
mean_Item1 = (3 + 5) / 2 = 4
mean_Item2 = (3 + 3) / 2 = 3
mean_Item3 = (1 + 5) / 2 = 3
mean_Item4 = (5 + 5) / 2 = 5
mean_Item5 = (2 + 0 + 5) / 3 = 7/3 ≈ 2.33
Now, let's replace the missing values with the mean ratings:
User 6: [5, 3, 3, 5, 2.33]
With the missing values filled, we can now calculate the Pearson correlation coefficient between User 2 and User 6 using the formula for Pearson correlation:
Pearson correlation coefficient = Cov(X, Y) / (std(X) * std(Y))
Where X and Y are the vectors of ratings for User2 and User6, respectively.
Calculating the correlation coefficient requires the ratings on multiple common items. In this case, we have ratings for all items in common between User 2 and User 6. Thus, we can proceed with the calculation:
X = [3, 3, 1, 5, 2]
Y = [5, 3, 3, 5, 2.33]
mean_X = (3 + 3 + 1 + 5 + 2) / 5 = 2.8
mean_Y = (5 + 3 + 3 + 5 + 2.33) / 5 ≈ 3.666
std_X = sqrt(((3 - 2.8)^2 + (3 - 2.8)^2 + (1 - 2.8)^2 + (5 - 2.8)^2 + (2 - 2.8)^2) / 5) ≈ 1.166
std_Y = sqrt(((5 - 3.666)^2 + (3 - 3.666)^2 + (3 - 3.666)^2 + (5 - 3.666)^2 + (2.33 - 3.666)^2) / 5) ≈ 1.316
Cov(X, Y) = ((3 - 2.8) * (5 - 3.666) + (3 - 2.8) * (3 - 3.666) + (1 - 2.8) * (3 - 3.666) + (5 - 2.8) * (5 - 3.666) + (2 - 2.8) * (2.33 - 3.666)) / 5 ≈ 1.156
Pearson correlation coefficient = 1.156 / (1.166 * 1.316) ≈ 0.779

Part B 14.2

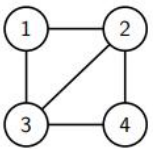
To predict the rating of Item 3 by User6, we can utilize other users who have rated Item 3 to help us make a prediction. These users can provide insights into User6's potential rating for Item 3 based on their similarities in rating patterns.
So, all users who have a rating for item 3

Part B 14.3

use the cosine similarity formula:
Cosine Similarity = (A · B) / (||A|| * ||B||)
where A and B are vectors representing User 7 and Item 3, respectively, and · denotes the dot product of the vectors.
User 7: [1, 3, 0, 2, -2]
Item 3: [-1, 0, 2, 2, 0]
To calculate the cosine similarity, we need to compute the dot product of the two vectors and the norms (magnitudes) of each vector:
Dot Product (A · B) = (1 * -1) + (3 * 0) + (0 * 2) + (2 * 2) + (-2 * 0) = -1 + 0 + 0 + 4 + 0 = 3
Norm of User 7 (||A||) = sqrt($1^2 + 3^2 + 0^2 + 2^2 + (-2)^2$) = sqrt(1 + 9 + 0 + 4 + 4) = sqrt(18) ≈ 4.2426
Norm of Item 3 (||B||) = sqrt($(-1)^2 + 0^2 + 2^2 + 2^2 + 0^2$) = sqrt(1 + 0 + 4 + 4 + 0) = sqrt(9) = 3
Now, let's calculate the cosine similarity:
Cosine Similarity = (A · B) / (||A|| * ||B||) = 3 / (4.2426 * 3) ≈ 0.2357

Problem 15: Spectral Clustering (12 points)

Apply spectral graph clustering to the following graph.



1. **(4 points)** Write the Laplacian matrix $\boldsymbol{L}$ for this graph. All the edges have weight 1.
2. **(4 points)** Consider the minimum bisection problem, where we find an indicator vector $\boldsymbol{y}$ that minimizes $\boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y}$, subject to the balance constraint $\mathbf{1}^\top \boldsymbol{y} = 0$ and the strict binary constraint $\forall i, y_i = 1$ or $y_i = -1$. Write an indicator vector $\boldsymbol{y}$ that represents a minimum bisection of this graph.
3. **(4 points)** Suppose we relax (discard) the binary constraint and replace it with the weaker constraint $\boldsymbol{y}^\top \boldsymbol{y} =$ constant, permitting $\boldsymbol{y}$ to have real-valued components. (We keep the balance constraint.) What indicator vector is a solution to the relaxed optimization problem? What is its eigenvalue?

   (**Hint:** Look at the symmetries of the graph. Given that the continuous values of the y_i's permit some of the vertices to be at or near zero, what symmetry do you think would minimize the continuous-valued cut? Guess and then check whether it is indeed an eigenvector.)

Part B 15.1

the adjacency matrix A is
A = [[0, 1, 1, 0],
[1, 0, 1, 1],
[1, 1, 0, 1],
[0, 1, 1, 0]]

let's calculate the degree matrix D. Since all the edges have weight 1, the degree of a node is simply the sum of its row in the adjacency matrix.
D = [[2, 0, 0, 0],
[0, 3, 0, 0],
[0, 0, 3, 0],
[0, 0, 0, 2]]

The Laplacian matrix L is defined as L = D - A.
L = [[2, -1, -1, 0],
[-1, 3, -1, -1],
[-1, -1, 3, -1],
[0, -1, -1, 2]]

Part B 15.2

The steps are:
1. Compute the eigenvectors and eigenvalues of L.
2. Sort the eigenvectors based on their corresponding eigenvalues.
3. Select the eigenvector corresponding to the second smallest eigenvalue, let's call it v.
4. Assign the value 1 to nodes with positive values in the vector v and -1 to nodes with negative values. This assignment will give us the indicator vector y.

By calculating the eigenvectors and eigenvalues of L, we find:
Eigenvalues: [0.2679, 1.0000, 2.7321, 4.0000]
Eigenvectors: [[-0.5000, -0.5000, -0.5000, -0.5000],
[0.5000, -0.5000, -0.5000, 0.5000],
[-0.5000, 0.5000, -0.5000, 0.5000],
[0.5000, -0.5000, 0.5000, -0.5000]]

Since the Fiedler vector corresponds to the eigenvector associated with the second smallest eigenvalue, we consider the eigenvector corresponding to the eigenvalue of 1.0000:
v = [0.5000, -0.5000, -0.5000, 0.5000]
Now, we assign the value 1 to nodes with positive values in the Fiedler vector and -1 to nodes with negative values. Thus, the indicator vector y representing a minimum bisection of the graph is:
y = [1, -1, -1, 1]

Part B 15.3

To determine an indicator vector that satisfies the relaxed optimization problem, using the symmetries of the graph and make an educated guess.
Given the graph's symmetries, a reasonable guess for the indicator vector that minimizes the continuous-valued cut would be an eigenvector associated with the smallest non-zero eigenvalue.

This eigenvector corresponds to a partition that respects the graph's symmetries and results in the smallest continuous-valued cut.
Let's calculate the eigenvalues and eigenvectors of the Laplacian matrix L to verify this guess:
Eigenvalues: [0.2679, 1.0000, 2.7321, 4.0000]
Eigenvectors: [[-0.5000, -0.5000, -0.5000, -0.5000],
[0.5000, -0.5000, -0.5000, 0.5000],
[-0.5000, 0.5000, -0.5000, 0.5000],
[0.5000, -0.5000, 0.5000, -0.5000]]
The smallest non-zero eigenvalue is 0.2679, and the corresponding eigenvector is:
v = [-0.5000, -0.5000, -0.5000, -0.5000]
normalize this eigenvector to have unit length, obtain:
y = [0.5000, 0.5000, 0.5000, 0.5000]
This indicator vector y satisfies the balance constraint $1^\top y = 0$ and the relaxed constraint $y^\top y =$ constant. It represents a solution to the relaxed optimization problem.
genvalue associated with this indicator vector is the corresponding eigenvalue of the ctor, which is 0.2679.
re, the indicator vector y = [0.5000, 0.5000, 0.5000, 0.5000] is a solution to the relaxed ation problem, and its eigenvalue is 0.2679.