

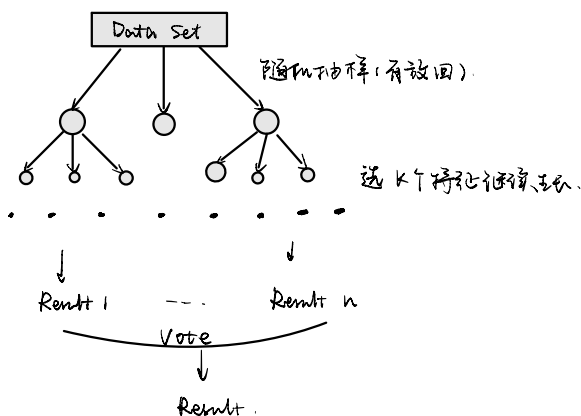
Chapter 1: Ensemble Learning. 集成学习

Lecture 2: Boosting, MoE and Stacking.

1. Ensemble Learning 组合多个模型 $\rightarrow$ better results.

- Bagging. 并行 $\rightarrow$ 投票. 降低 Var, 减少 Overfitting.
- Boosting. 串行 $\rightarrow$ 纠正前一个的错误. 降低 bias, 防止 Underfitting.
- Stacking. 基模型 $\rightarrow$ 元模型 $\rightarrow$ result

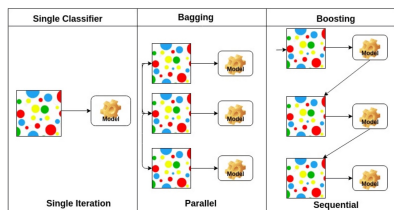
2. Random Forest (Bagging).



优: 参数不敏感, 准确

劣: 对 high-dim sparse data 不适用

3. Boosting. Gradient Boost
AdaBoost.



Ensemble Model: $H_T(x) = \sum_{t=1}^T \alpha_t h_t(x)$.

加法模型.
(区别于 NN, 层层嵌套)

类似 GB, (更新日使 $L(\theta) \rightarrow \min$).

Boosting: 添加 h_{t+1} 使 $L(\theta) \rightarrow \min$.

$$L(H) = \frac{1}{n} \sum_{i=1}^n \ell(H(x_i), y_i)$$

find h_{t+1}, α_{t+1}

更新任务 $\Rightarrow$

$$h_{t+1}, \alpha_{t+1} = \underset{h, \alpha}{\operatorname{argmin}} L(H_t + \alpha h).$$

问题: 怎么解 $h_{t+1}, \alpha_{t+1} = \underset{h, \alpha}{\operatorname{argmin}} L(H_t + \alpha h)$?

① Gradient Boosting.

Use Taylor approximation!

$$L(H + \alpha h) \approx L(H) + \alpha \langle \nabla L(H), h \rangle$$

$$\Rightarrow h = \underset{h}{\operatorname{argmin}} L(H + \alpha h) = \underset{h}{\operatorname{argmin}} \langle \nabla L(H), h \rangle$$

$$= \underset{h}{\operatorname{argmin}} \sum_{i=1}^n \frac{\partial L}{\partial H(x_i)} \cdot h(x_i) = \underset{h}{\operatorname{argmin}} \sum_{i=1}^n \eta_i h(x_i)$$

$\Rightarrow \eta_i$ 与 h 有关 $\rightarrow$ training data i 的权重 (To $\downarrow$ Loss, i should move to where?)

$$h_{t+1} = \operatorname{argmin} \sum q_i h_i$$

Input: $\ell, \alpha, T, \{(x_i, y_i)\}, H, A(\text{an algorithm})$
 1: $H_0 = 0$
 2: **for** $t = 0 : T - 1$ **do**
 3: $q_i = \frac{\partial \ell}{\partial H(x_i)}, i = 1, 2, \dots, n$
 4: $h_{t+1} = \operatorname{argmin}_{h \in H} \sum_{i=1}^n q_i h(x_i)$ via Algorithm A
 5: **if** $\sum_{i=1}^n q_i h_{t+1}(x_i) < 0$ **then**
 6: $H_{t+1} = H_t + \alpha h_{t+1}$ (α may also be optimized)
 7: **else**
 8: **return** (H_t)
 9: **end if**
 10: **end for**
Output: H_T

Ideally, $h = -\nabla L(H)$, 误差 = 残差

$\Rightarrow$ Gradient Boosting for Regression.

For now, we have. $h_{t+1} = \operatorname{argmin}_h \sum_{i=1}^n q_i h(x_i)$, where $q_i = \frac{\partial \ell}{\partial H(x_i)}$.

Assume $\sum_{i=1}^n h(x_i) = \text{constant}$ (easily achieve by normalization)

and replace q_i with $-2r_i$

$$h_{t+1} = \operatorname{argmin}_h \sum_{i=1}^n q_i h(x_i) = \operatorname{argmin}_h -2 \sum_{i=1}^n r_i h(x_i)$$

$$= \operatorname{argmin}_h \sum_{i=1}^n (r_i^2 - 2r_i h(x_i) + (h(x_i))^2)$$

$$= \operatorname{argmin}_h \sum_{i=1}^n (h(x_i) - r_i)^2 \quad \text{Use } h_{t+1} \text{ to predict } r_i$$

I. Let $L(H) = \sum (H(x_i) - y_i)^2$, $r_i = -\frac{1}{2} q_i$

$$r_i = -\frac{1}{2} \times \frac{\partial L}{\partial H(x_i)} = y_i - H(x_i). \quad \Rightarrow \text{ } r_i \text{ is the residual !!}$$

II. Let $L(H) = \sum |H(x_i) - y_i|$

$$q_i = -\operatorname{sign}(y_i - H(x_i))$$

$\Rightarrow$ Gradient Boosting for classification.

Predicting prob. of each K classes.

$$P_K(x) = \frac{e^{h^K(x)}}{\sum e^{h^k(x)}}$$

② AdaBoost

Use exponential loss $L(H) = \sum_{i=1}^n e^{-y_i H(x_i)}$

$$q_i = \frac{\partial L}{\partial H(x_i)} = -y_i e^{-y_i H(x_i)}$$

$$\Rightarrow w_i = \frac{1}{2} e^{-y_i H(x_i)}, \quad z = \sum e^{-y_i H(x_i)}$$

Then
$$h = \underset{h}{\operatorname{argmin}} \sum q_i h(x_i) = \underset{h}{\operatorname{argmin}} - \sum_{i=1}^n w_i y_i h(x_i)$$

$$= \underset{h}{\operatorname{argmin}} \sum_{h(x_i) \neq y_i} w_i - \sum_{h(x_i) = y_i} w_i$$
 分类. if $h(x_i) = y_i \Rightarrow$ 分类正确, $y_i h(x_i) = 1$
 $\neq \Rightarrow$ 错误. $= -1$. $\Rightarrow$ 损失 = 正确 - 错误.

$$= \underset{h}{\operatorname{argmin}} \sum_{h(x_i) \neq y_i} w_i \Rightarrow \text{holds because } \sum w_i = 1.$$

So,
$$h = \underset{h}{\operatorname{argmin}} \sum_{h(x_i) \neq y_i} w_i$$

Given h , find α :

$$\alpha = \underset{\alpha}{\operatorname{argmin}} L(H + \alpha h) = \underset{\alpha}{\operatorname{argmin}} \sum e^{-y_i(H(x_i) + \alpha h(x_i))}$$

Differentiate w.r.t. α

$$\sum y_i h(x_i) e^{-(y_i H(x_i) + \alpha y_i h(x_i))} = 0.$$

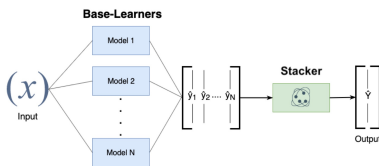
$$\Rightarrow \alpha = \frac{1}{2} \ln \frac{1 - \epsilon}{\epsilon}$$

③ Comparison: Gradient Boosting vs. AdaBoost.

AdaBoost 使用 exponential loss $\rightarrow$ 对 outliers 更敏感.

4. MoE: Mixture of Experts.

5. Stacking.



Lecture 3: Advanced Applications.

1. Recommendation System

Method I: Collaborative Filtering. 协同过滤.

找与你相似的人.

① Nearest-Neighbor Collaborative Filtering.

$$\hat{x}_{ui} = \bar{x}_u + \sum_{v \in \mathcal{N}} \left((x_{vi} - \bar{x}_v) \times \frac{w_{uv}}{\sum_{u' \in \mathcal{N}} w_{u'v}} \right)$$

- $\bar{x}_u$ (基准偏移): 这是用户 u 给所有物品打分的平均值。为什么要加这一项? 因为每个人的打分尺度不同。有的用户是“给分狂魔”, 平均分 4.5; 有的则是“严格判官”, 平均分只有 2.0。加上基准分, 是为了在这个用户自己的尺度上做预测。
- $(x_{vi} - \bar{x}_v)$ (邻居的偏好): 邻居 v 对物品 i 的评价相对于他自己平均分的波动。如果结果是正数, 说明邻居 v 喜欢这个东西; 如果是负数, 说明他不喜欢。
- $\frac{w_{uv}}{\sum_{u' \in \mathcal{N}} w_{u'v}}$ (归一化相似度权重):
 - w_{uv} 通常是用户 u 和 v 之间的 Pearson 相关系数或余弦相似度。
 - 这个权重决定了: 越像你的邻居, 他的意见对你的影响就越大。
 - 分母是为了让所有邻居的权重和为 1。

② Matrix Factorization

将你的喜好与电影标签以匹配程度。

SVD-based recommendation ($R \approx P^T Q$)

$$\min_{P, Q} \sum_{(u, i) \in \Omega} \left\{ (r_{ui} - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2) \right\}$$

Regularization

R : User-item rating matrix

P, Q 分别是 User / Item matrix.

SVD with bias ($b_{ui} = \mu + b_u + b_i$)

$$\min_{P, Q, B} \sum_{(u, i) \in \Omega} \left\{ (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2) \right\}$$

$B = \{\mu, \{b_u\}, \{b_i\}\}$ μ 是 avg. 降低个人打分习惯的影响。
全局用户物品

- 现在的做法 (MF): 我们假设每部电影都有一些隐藏标签 (Latent Factors), 比如“动作成分”、“感人程度”、“科幻感”。
什么算法算出来的
- 用户向量 p_u : 代表用户 u 对这些标签的喜爱程度 (比如你 90% 喜欢科幻, 10% 喜欢感人)。
- 物品向量 q_i : 代表电影 i 在这些标签上的得分 (比如《星际穿越》是 95% 的科幻, 20% 的感人)。

当你把这两个向量相乘 (点积 $p_u^T q_i$), 结果就是你对这部电影的预测分。

优点: 无需领域知识。

缺点: 冷启动 (新用户)

Method II: Content Based

- 内容分析 (Content analysis): 将物品 (Item) 转化为特征向量 v_i 。
 - 例如: 对于一部电影, 特征可以是文本描述的 TF-IDF 权重、海报的图像特征、或者是预定义的标签 (导演: 诺兰, 类型: 科幻)。
- 用户画像学习 (Profile learning): 将用户 (User) 也转化为特征向量 z_u 。
 - 这通常基于用户的显式信息 (年龄、性别、教育程度) 以及用户过去喜欢的物品特征。而非 SVD 中的隐向量。
- 过滤模块 (Filtering module): 这是核心。它训练一个分类或回归模型, 输入用户特征 z 和物品特征 v , 预测用户对该物品的兴趣程度。

- 目标函数: $\minimize_g \sum_{(u, i) \in \Omega} \ell(r_{ui}, g_u(v_i))$
- v_i : 物品 i 的 d' 维特征向量。
- g : 一个映射函数 (如神经网络或线性回归), 输入物品特征, 输出一个 m 维向量 (代表 m 个用户对该物品的预测评分)。
- $g_u(v_i)$: 函数 g 的第 u 个输出, 即用户 u 对物品 i 的预测值。
- ℓ : 损失函数 (通常是平方损失)。

2. Evaluation Metrics

① 预测指标

- RMSE (Root Mean Square Error, 均方根误差):
 - 公式: $RMSE = \sqrt{\frac{1}{R} \sum_{r_{ui} \in R} (r_{ui} - \hat{r}_{ui})^2}$
 - 特点: 它对大误差非常敏感。因为误差被平方了, 所以如果模型把一个 5 分电影猜成了 1 分, RMSE 会飙升得非常厉害。
- MAE (Mean Absolute Error, 平均绝对误差):
 - 公式: $MAE = \frac{1}{R} \sum_{r_{ui} \in R} |r_{ui} - \hat{r}_{ui}|$
 - 特点: 它对误差的反应是线性的, 比 RMSE 更“稳健 (Robust)”。

② 排名指标

- Precision@k (准确率):
 - 含义: 推荐给你的 k 个东西里, 真正相关的 (被你点过的) 占比多少。
 - 逻辑: $\frac{\text{相关且推荐了}}{\text{总共推荐的 } k \text{ 个}}$
- Recall@k (召回率 / HitRatio@k):
 - 含义: 用户真正喜欢的那些东西里, 你成功抓到了多少放进了前 k 名中。
 - 逻辑: $\frac{\text{相关且推荐了}}{\text{用户所有相关的总数(历史)}}$
- Average Precision (AP@N):
 - 核心逻辑: 它不仅看推得对不对, 还看顺序。
 - 特点: 如果你把用户喜欢的商品排在第 1 名, AP 会很高; 如果排在第 10 名, 哪怕抓到了, AP 也会因为位置靠后而被打折扣。

3. Ranking

1. Pointwise Modeling (点对点建模)

这是最简单、最直观的方法。它将排序问题转化为单一对象的回归或分类问题。

- 核心逻辑: 对于给定的查询, 模型独立地为每一个文档打分 (例如: 是/否相关, 或 0 到 100 的连续值)。
- 输入: 单一的 (查询, 文档) 特征向量。
- 缺点: 这是它的致命伤。因为它每次只看一个文档, 所以完全忽略了文档之间的相对顺序。在学习过程中, 模型并不知道“把 A 排在 B 前面”比“给 A 打 0.9 分”更重要。

2. Pairwise Modeling (成对建模)

这种方法更接近“排序”的本质, 因为它关注的是两个文档之间的相对好坏。

- 核心逻辑: 给定一个查询, 模型同时输入两个文档, 并预测哪一个更符合用户的偏好。
- 输入: 包含 (查询, 文档1, 文档2) 的特征向量。
- 输出: 通常是一个二分类结果 (+1 表示文档1更好, -1 表示文档2更好)。
- 缺点: 它无法区分“极好与极坏”和“一般与极坏”的区别 (即区分度缺失)。如果两个文档都很烂, 强制模型选一个更好的可能并没有意义。

3. List-wise Modeling (列表建模)

这是目前最先进、但也最复杂的方法。它直接对整个搜索结果列表进行建模。

- 核心逻辑: 模型一次性考虑所有候选文档, 并优化整个列表的排列顺序。
- 优势: 位置信息对损失函数可见。它能直接优化像 nDCG 或 AP 这种关注“头部排名”的指标。
- 缺点: 训练复杂度极高。计算全列表的排列组合对算力的要求非常大。

Lecture 4-1 Spectral Clustering

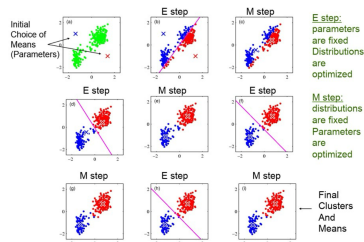
1. Intro

① K-Means

Expectation - Maximization (EM) 的变种

→ E步固定中心，分配数据

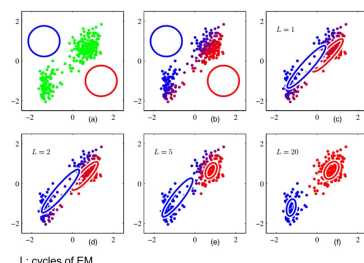
→ M步固定归属，调整中心



② GMM (Gaussian Mixture Model)

类似 K-Means，但引入概率

调整协方差矩阵，拟合簇



Compactness

K-Means 的局限性：- 易于收敛性，处理长数据

非凸 → 谱聚类 (连通性, Connectivity)

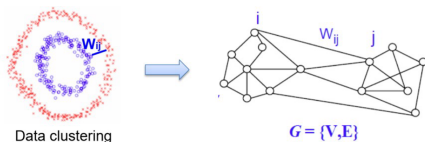
2. Graph Partition

目标：将图分成若干组

- 组内边的权重高 (簇内紧凑)
- 组间边的权重小 (一一稀疏)

Similarity Graph Construction

Given $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, construct a similarity graph.



- k-nearest neighbor graph
- ϵ -neighborhood graph
- Gaussian kernel similarity function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

1. k-nearest neighbor graph (k近邻图):

- 规则：我是点 A，我只连离我最近的 k 个人。
- 特点：这能保证产生的图是稀疏的 (Sparse)，计算效率高，且不容易把两个离得很远的簇意外连起来。这是最常用的方法。

2. ϵ -neighborhood graph (ϵ -邻域图):

- 规则：以我为中心，画一个半径为 ϵ 的圈，圈里的人我都连。
- 缺点： ϵ 很难选。选小了，图会断裂 (很多孤立点)；选大了，整个图都连通了，聚类失效。

3. Gaussian kernel similarity function (全连接 + 高斯核):

- 这是最“数学”的方法，它生成的通常是全连接图 (虽然很多边权重极小，接近 0)，但给每个边赋予了一个精细的权重值。

3. Minimum / Normalized Cut

Cut: 将 V 划分为 A, B . (A, B 不相交).

$$\text{Cut 的大小: } \text{Cut}(A, B) := \sum_{i \in A, j \in B} W_{ij}$$

① Minimum Cut: $\min \text{Cut}(A, B)$.

Easy to solve $O(|V| \cdot |E|)$
但倾向于切出孤立点.

② Normalized Cut:

在 min cut 基础上, 要求 A, B 体积不能太小.

$$N\text{Cut}(A, B) = \text{Cut}(A, B) \cdot \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)$$

$$\text{vol}(A) = \sum_{i \in A} d_i \quad (\text{sum of degrees})$$

NP-hard.

高维 $\xrightarrow{\text{relax}}$ 连续 (spectral relaxation) $\Rightarrow$ eigenvalue problem.

Graph Laplacian.

Given a graph with similarity matrix

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}$$

The degree matrix of the graph is defined as

$$D = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_N \end{bmatrix}$$

where $d_i = \sum_{j=1}^N w_{ij}$. d_i is the degree of vertex i of the graph.

The graph Laplacian is defined as:

$$L = D - W$$

$\rightarrow L$ 对称, 半正定.

最小特征值 = 0. 对应的特征向量是 $\mathbf{1}$

Ncut is equivalent to the minimization of $\frac{\mathbf{u}^T L \mathbf{u}}{\mathbf{u}^T D \mathbf{u}}$, i.e.,

$$\min_{A, B} \text{Ncut}(A, B) \iff \min_{\mathbf{u}} \frac{\mathbf{u}^T L \mathbf{u}}{\mathbf{u}^T D \mathbf{u}}, \quad \mathbf{u} \in \mathbb{R}^N, \quad u_i = \begin{cases} \frac{1}{\text{vol}(A)}, & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)}, & \text{if } i \in B \end{cases}$$

$$\min_u \frac{u^T L u}{u^T D u} \quad \text{s.t. } u^T D \mathbf{1} = 0, \quad u_i \in \{1, -1\}.$$

Relaxation: $u = \{L u = \lambda D u\}$ 的第 n 小特征值时对应的特征向量。

等价于归一化 $\tilde{L} = D^{-1} L = I - D^{-1} W$ 的第 n 小特征值时对应的特征向量

Result: w_0 为界 $u_i \geq 0 \Rightarrow \in A \quad u_i < 0 \Rightarrow \in B$

4. Spectral Clustering Algorithm.

Input: data $X = \{x_1, x_2, \dots, x_N\}$. number K of clusters.

- **Step 1.** Construct a similarity matrix W

e.g. use $w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

k -nearest neighbor graph, or ϵ -neighborhood graph

The eigenvalues satisfy

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \dots \leq \lambda_{N-1} \leq \lambda_N$$

- **Step 2.** Compute the Laplacian matrix L (or normalized L)

- $L = D - W$
- $\tilde{L} = I - D^{-1}W$ (normalized)
- $\hat{L} = I - D^{-1/2} W D^{-1/2}$ (symmetric normalized, recommended)

有 n 个 0 特征值。就有 n 个簇。

- **Step 3.** Perform eigenvalue decomposition on L (or normalized L) and use the first K eigenvectors to form a matrix Z

$$\hat{L} = V \Sigma V^T, \quad Z = [v_1, v_2, \dots, v_K]^T \in \mathbb{R}^{K \times N}$$

- **Step 4.** Normalize the columns of Z to unit L_2 norm, i.e.,

$$z_i \leftarrow z_i / \|z_i\|, \quad i = 1, \dots, N$$

- **Step 5.** Perform K-means on $\{z_1, z_2, \dots, z_N\}$

Output: K of clusters of Z or X

谱聚类的优势 (Advantages)

- **高聚类准确度:** 在许多实际应用中, 谱聚类的准确度非常高。 ^①
- **性能优于 K-means:** 在处理非凸 (non-convex) 或具有复杂拓扑结构的数据分布时, 它的表现通常优于传统的 K-means 算法。 ^{②①}
- **捕捉连通性:** 谱聚类基于“连通性”(Connectivity) 准则, 能够识别 K-means 这种基于“紧凑性”(Compactness) 准则的算法无法处理的形状, 如嵌套圆环或丝状分布。 ^{③③}
- **自动确定簇数 K :** 利用拉普拉斯矩阵的特征值属性, 可以通过最大化“特征间隙”(Eigengap) 来自发式地确定最合理的聚类数量 K 。 ^④

谱聚类的劣势 (Disadvantages)

- **计算成本极高:** 由于涉及稠密矩阵的存储和复杂的特征分解, 该算法不适用于大规模数据 (Big Data) 场景。 ^{⑤②}
- **空间复杂度:** 空间复杂度为 $O(N^2)$, 主要源于存储 $N \times N$ 的相似度矩阵 W 或拉普拉斯矩阵 L 。 ^{⑥①}
- **时间复杂度:** 时间复杂度为 $O(N^3)$, 主要用于对矩阵进行特征值分解。 ^{⑦①}
- **相似度矩阵难以确定:** 选择何种构建方法 (如 KNN、 ϵ -邻域或高斯核) 以及如何设定相应的超参数 (如高斯核的 σ) 并不容易, 且对结果影响显著。 ^⑧
- **最终仍依赖 K-means:** 算法的最后一步通常仍需在特征空间运行 K-means 来获得最终标签, 这意味着谱聚类并没有完全摆脱对局部最优解的敏感性。 ^⑨

Lecture 4 - II : Semi - Supervised Learning.

1. Classification.

- o Self-training algorithm.
- o Graph Based algorithm
- o Graph Convolutional Network based SSL

2. Self-training Algorithm.

假设: 模型自身做出的高江预测是正确的.

步骤：①由 (X_L, Y_L) 训练模型 f .
 ↓ labeled
 ②对 $x \in X_{in}$ 预测
 ↓ unlabeled

重复以上步骤。

③ 将 $(x, f(x))$ 加入 (x_i, y_i)

事件

全部加入
只加入少量自信样本
全部加入，但赋予权重

3. Graph-based SSL Method

假设: 给定包含标签, 无标签数据的图. 通过重边相连的实例往往有相同的标签.

• 圖式化.

- 引入无标签数据进行正则化：如果 x_i 和 x_j 相似（权重 W_{ij} 大），则预测标签 $f(x_i)$ 和 $f(x_j)$ 也应相似。
- 优化目标转换为：
$$\min_f \sum_{i=1}^I l(y_i, f(x_i)) + \lambda \sum_{i=1}^N \sum_{j=1}^N W_{ij} \|f(x_i) - f(x_j)\|^2.$$

Algorithm 11.1 Label propagation (Zhu and Ghahramani [2002])

Compute affinity matrix $\mathbf{W}$ from (11.1)
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Initialize $\hat{\mathbf{Y}}^{(0)} \leftarrow (y_1, \dots, y_l, 0, 0, \dots, 0)$

Iterate

1. $\hat{Y}^{(t+1)} \leftarrow \mathbf{D}^T$
2. $\hat{Y}_l^{(t+1)} \leftarrow Y_l$

until convergence to $\hat{Y}^{(\infty)}$

Label point x_i by the sign of $\hat{y}_i^{(\infty)}$

Lecture 5: Graph Neural Network.

1. Graph Data.

- $G = (V, E)$
- Vertices/nodes $V = \{v_1, v_2, \dots, v_n\}$, edges/links $E = \{e_1, e_2, \dots, e_l\}$
- Affinity matrix $A \in \mathbb{R}^{n \times n}$ 谁和谁相连
- Feature matrix of nodes $X \in \mathbb{R}^{n \times d}$ (may not exist) 包含什么信息

2. Convolution on Graph.

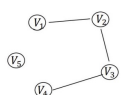
$$H^{(l+1)} = \sigma(\tilde{A} H^{(l)} W^{(l)})$$

- σ 代表激活函数 (如 ReLU 或 Sigmoid)
- $W^{(l)}$ 代表参数矩阵。
- $\tilde{A}$ 为重归一化后的矩阵, 计算方式为 $\tilde{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, 其中引入了自环 $\tilde{A} = A + I$ 。

$$\tilde{D} = \text{diag}(\sum_i \tilde{A}_{i1}, \dots, \sum_i \tilde{A}_{in})$$

Example.

$$\tilde{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}, \tilde{A} = A + I, \tilde{D} = \text{diag}(\sum_i \tilde{A}_{i1}, \dots, \sum_i \tilde{A}_{in})$$



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tilde{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 3 & 3 & 2 & 1 \end{bmatrix}$$

$\Rightarrow$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \hat{A}X = \begin{bmatrix} 0.5x_1 + 0.41x_2 \\ 0.41x_1 + 0.33x_2 + 0.33x_3 \\ 0.33x_2 + 0.33x_3 + 0.41x_4 \\ 0.41x_3 + 0.5x_4 \\ x_5 \end{bmatrix}$$

$$\tilde{D}^{-1/2} = \begin{bmatrix} 0.71 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.58 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.58 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.71 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \tilde{A} = \begin{bmatrix} 0.5 & 0.41 & 0.0 & 0.0 & 0.0 \\ 0.41 & 0.33 & 0.33 & 0.0 & 0.0 \\ 0.0 & 0.33 & 0.33 & 0.41 & 0.0 \\ 0.0 & 0.0 & 0.41 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$\tilde{D}^{-1/2} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 1 \end{bmatrix}, \tilde{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$$

$$H^{(0)} = X$$

$$H^{(1)} = \sigma(\tilde{A} X W^{(1)})$$

$$H^{(2)} = \sigma(\tilde{A} H^{(1)} W^{(2)})$$

$\vdots$

$$H^{(l+1)} = \sigma(\tilde{A} H^{(l)} W^{(l+1)})$$

$\vdots$

3. GCN 架构

常用的 GCN 通常只有 2 层:

图卷积是邻域信息的交换, 如层数过深, 所有节点都会发收敛 → oversmoothing

$$Z = f(X, A) = \text{softmax}(\tilde{A} \text{ReLU}(\tilde{A} X W^{(0)}) W^{(1)})$$

$$\text{Cross-Entropy Loss: } \mathcal{L} = - \sum_{\substack{i \in V_L \\ \text{labelled}}} \sum_{k=1}^K Y_{ik} \ln Z_{ik}$$

4. READOUT.

已给全部节点打完标签, $\Rightarrow$ 把所有节点的特征向量求平均或全局特征向量。

SUM $\rightarrow$ 最好! , Average, Min/Max ...

5. Link Prediction

$$p(A_{ij}) = \sigma(z_i^T z_j)$$

Lecture 6: Nonlinear Dimensionality Reduction.

1. Intro.

Linear DR: PCA (寻找投影后方差最大的方向).
LDA (类内聚集, 类外分离).
MDS (依赖样本间距离还原重构).

NLDR: Kernel PCA.
LLE, Isomap.
t-SNE, UMAP.
AutoEncoder.

2. LLE (Locally Linear Embedding).

① Logic.

Find neighbors $\rightarrow$ calculate relationship $\rightarrow$ mapping.

用邻居的线性组合拼出 x_i .

$$x_i \approx \sum_{j \in \text{Neighbor}_i} w_{ij} x_j, \quad \sum_j w_{ij} = 1$$

②

- Input: D-dimensional data points $x_1, x_2, \dots, x_N$, d , k ($k \geq d+1$)

- For each data point x_i , find its k nearest neighbors N_i^k
- Find the locally linear reconstruction weights by solving

$$W = \argmin_W \sum_{i=1}^N \|x_i - \sum_{j \in N_i^k} w_{ij} x_j\|^2, \quad \text{s.t. } \forall i \sum_{j \in N_i^k} w_{ij} = 1$$

- Use W to compute the low-dimensional projections

$$Z = \argmin_Z \sum_{i=1}^N \|z_i - \sum_{j \in N_i^k} w_{ij} z_j\|^2, \quad \text{s.t. } \sum_{i=1}^N z_i = 0, \quad \frac{1}{N} Z Z^T = I$$

$$\begin{aligned} \sum_{i=1}^N \|z_i - \sum_{j \in N_i^k} w_{ij} z_j\|^2 &= \|Z - ZW\|_F^2 \\ &= \text{trace}(Z(I-W)(I-W^T)Z^T) \end{aligned}$$

- Output: Low-dimensional embedding $Z \in \mathbb{R}^{d \times N}$

3. t-SNE

t-SNE 的核心目标非常纯粹: 把高维空间中距离相近的点, 映射到低维空间 (通常是 2D 或 3D) 时依然保持相近, 同时把原本相距较远的点推开。为了实现这一点, 它通过 **概率分布替换了传统的距离计算**。与 LLE 区别。

1. 高维空间的相似度计算: 高斯分布

在高维空间中, t-SNE 使用高斯分布来定义点与点之间的“邻域概率”。

对于给定数据点 $x_1, x_2, \dots, x_N$, 点 x_i 选择点 x_j 作为其邻居的概率 p_{ji} 为:

②-1

$$p_{ji} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

②

- 困惑度 (Perplexity) 的作用:** 公式中的方差 σ_i^2 决定了我们要考察的“邻域大小”。t-SNE 并不是一刀切地使用固定的方差, 而是允许用户设置一个“困惑度”参数 (推荐值在 5-50 之间)。算法会根据这个参数为每个点动态计算出合适的 σ_i 。②-2

- 对称化处理:** 为了方便后续的矩阵计算和求导, t-SNE 会对概率进行联合对称化处理, 即 $p_{ij} = \frac{p_{ji} + p_{ij}}{2N}$ 。②-3

2. 低维空间的相似度计算: 引入 t-分布

这是整个 t-SNE 算法最天才的“戏眼”。

如果我们在低维空间也沿用高斯分布 (这就是早期的 SNE 算法), 会遇到致命的 **拥挤问题 (Crowding Problem)**。因为高维空间的容积很大, 而低维空间 (比如 2D) 的物理容积很小, 根本没有足够的空间来容纳所有的高维邻居点。

②-4

为了解决这个问题, t-SNE 在低维空间放弃了高斯分布, 转而使用了自由度 $\nu = 1$ 的 **Student-t 分布**。②-5

- 重尾效应 (Heavy Tail):** 你可以参考上面的图像, t-分布的尾部比高斯分布更厚。这意味着, 要在低维空间中表达与高维空间相同的极小概率 (即相距较远的点), 低维空间中的点必须被推得非常远。

- 因此, 低维空间中的概率 q_{ij} 计算公式变成了这样 (没有了指数项, 计算也变简单了):

$$q_{ij} = \frac{(1 + \|z_i - z_j\|^2)^{-1}}{\sum_{k \neq i} \sum_{k \neq j} (1 + \|z_k - z_i\|^2)^{-1}}$$

3. 目标优化：KL 散度

现在我们有了高维的真实概率矩阵 P 和低维的模拟概率矩阵 Q 。我们要做的就是让 Q 尽可能逼近 P 。
t-SNE 使用了 KL 散度 (Kullback-Leibler divergence) 来作为损失函数：

② +1

$$\mathcal{L} := KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

随后，算法利用梯度下降法不断更新低维坐标 z_i ，直到达到收敛状态。

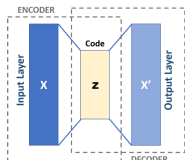
② +1

4. Auto Encoder

Lecture 7. Generative Models

1. Variational AutoEncoder (VAE)

① Review AutoEncoder.



- Encoder network $f_\phi: \mathbf{x} \rightarrow \mathbf{z}$
- Decoder network $g_\theta: \mathbf{z} \rightarrow \mathbf{x}'$
- Optimization objective:

$$\min_{\phi, \theta} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - g_\theta(f_\phi(\mathbf{x}_i))\|^2 + \mathcal{R}(\phi, \theta)$$

但是AE无法生成新 data.

(z 是编码后的点, 在 z 所在空间和原数据未必能还原成原数据)

→ 引入 VAE.

② VAE Motivation.

不再把 z 编码成向量 z'.

AE: $\mathbf{x} \rightarrow \mathbf{z}' \rightarrow \mathbf{x}'$

↓
生成分布!

VAE: $\mathbf{x} \rightarrow q_\phi(\mathbf{z}|\mathbf{x}) \rightarrow \mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{x}) \rightarrow p_\theta(\mathbf{x}|\mathbf{z})$

$p(\mathbf{z})$: 先验分布.

如何生成新样本?

先从 $p(\mathbf{z})$ 中采样 z, 再从解码器 $g_\theta(\mathbf{z})$ 生成样本.

→ 如何得到 decoder $g_\theta(\mathbf{z})$? 训练, 给真实样本尽可能的概率.

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^n \log p(\mathbf{x}_i|\theta) = p_\theta(\mathbf{x})$$

⇒ 如何表示 $p_\theta(\mathbf{x})$?

两种思路: Marginal likelihood.

Bayes' Rule.

积分运算.

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}.$$

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}.$$

证据.

⇒ Learn! 再训练一个 encoder 学习出 z 近似后验.

$$q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$$

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} = \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \log p_\theta(\mathbf{x}|\mathbf{z}) - \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} + \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \end{aligned}$$

⇒ 取期望:

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] = \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} d\mathbf{z} \\ &\quad + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \\ &\quad + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \end{aligned}$$

KL 散度非负 ⇒ 存在下界.

主要操作 分布更简单.

$$\log p_\theta(\mathbf{x}) \geq \underbrace{\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{下界: ELBO}} \triangleq \mathcal{L}_{\phi, \theta}(\mathbf{x})$$

$$\log p_\theta(\mathbf{x}) = \mathcal{L}_{\phi, \theta}(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))$$

VAE maximizes ELBO, i.e.,

$$\phi^*, \theta^* = \arg\max_{\phi, \theta} \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

当 $\log p_\theta(\mathbf{x}) = \mathcal{L}_{\phi, \theta}(\mathbf{x})$ 时: $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$

VAE 目标: MAX ELBO

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

sampling 是随机操作 梯度无法传回 $\Rightarrow$ Reparameterization.

$$\mathbf{z} = \mu_{\phi}(\mathbf{x}) + \sigma_{\phi}(\mathbf{x}) \odot \epsilon, \text{ where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- maximize $\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})]$: reconstruct $\mathbf{x}$
- minimize $D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$: approximate prior

Suppose $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Then let $q_{\phi}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_{\phi}(\mathbf{x}), \text{diag}(\sigma_{\phi}^2(\mathbf{x})))$.

2. Generative Adversarial Networks (GAN).

- 生成器 G : 输入随机噪声 z , 输出伪样本 $G(z)$
- 判别器 D : 输入样本, 输出它来自真实数据的概率

目标是:

- 判别器尽量把真样本判成真, 把假样本判成假
- 生成器尽量生成能骗过判别器的样本

训练:

1. 固定生成器, 训练判别器
2. 固定判别器, 训练生成器
3. 反复迭代

- p_r : 真实数据分布
- p_g : 生成器诱导的数据分布
- p_z : 噪声先验分布

判别器目标

希望真实样本得高分:

$$\max_D \mathbb{E}_{x \sim p_r(x)} [\log D(x)]$$

希望生成样本得低分:

$$\max_D \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

生成器目标

课件写的是:

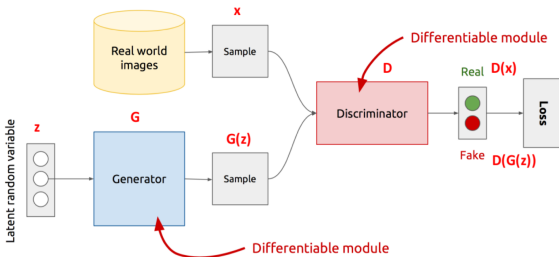
$$\min_G \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

也就是让 $D(G(z))$ 尽量大, 换句话说, 让判别器把假样本当真。

□ CDA4210 Lecture 07 Generative M...

合并成 minimax 游戏

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$



3. Diffusion Model.

前向过程: 逐步加高斯噪声.

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_{t-1}$$

其中:

- $\beta_t \in (0, 1)$ 是每一步噪声强度
- $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$

于是有条件分布:

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I})$$

设

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^t \alpha_s$$

则可以直接写成:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

其中 $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. 因此:

$$q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

这非常重要, 因为它说明:

不必一步一步加噪, 也能直接从 x_0 跳到任意 x_t .

反向过程：逐步消噪。

课件第 33 页说明，如果扩散过程可以反过来，那么从高斯噪声 $x_T \sim N(0, I)$ 出发，就能逐步还原出真实样本。模型学习的就是：

$$p_{\theta}(x_{t-1}|x_t) = N(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

这里神经网络输入是：

- 当前噪声样本 x_t
- 时间步 t

输出是反向过程所需的均值和方差参数。 DDA4210 Lecture 07 Generative M...

训练一个网络去预测噪声 ϵ

最终的简化目标是：

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, x_0, \epsilon} [\|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2]$$

这表示：

- 先随机选一个时间步 t
- 给真实样本 x_0 加噪得到 x_t
- 让网络预测当时加入的噪声 ϵ
- 用均方误差训练

总结， 7.2 生成质量对比

按照课件表达：

- VAE：通常较平滑，容易偏模糊
- GAN：通常更真实、更锐利
- Diffusion：常常最高质量

7.3 训练难度对比

- VAE：相对稳定
- GAN：最难训练，容易不稳定、模式崩塌
- Diffusion：训练公式稳定，但计算量大、时间长

7.4 概率建模能力

- VAE：显式概率建模，有 ELBO
- GAN：隐式分布，不易直接算 $p(x)$
- Diffusion：显式概率建模

7.5 采样速度

- VAE：快，一次解码
- GAN：快，一次前向传播
- Diffusion：慢，要多步迭代去噪

Lecture 08 Foundation Models

1. Fine-tuning. 让模型适应任务.

① LoRA : Low Rank Adaption.

原始更新: $W \rightarrow W + \Delta W$

LoRA : $\Delta W = AB$, 学两个小矩阵.

② Others :

- Partial fine-tuning: Fine-tune only a subset of the model's parameters (e.g., the top layers or specific layers) while freezing the rest.
- Multi-task fine-tuning
- Adversarial fine-tuning
- ...

2. Model Compression

① Knowledge Distillation : Teacher - Student Model.

② Quantization : 把参数精度调低.

③ Factorization : $W \approx AB$. 分解成两个小矩阵.

④ Pruning.

3. Hallucination 幻觉 Detection

① Perplexity. 困惑度. (越困惑越高)

$$PPL_{\theta}(\mathbf{x}) = \exp\left(-\frac{1}{n} \sum_{i=1}^n \log P_{\theta}(x_i | \mathbf{x}_{<i})\right)$$

PPL 考察的是统计流畅性, 而非事实正确性 / 推理能力.

Lecture 9: AI4Science.

无重要知识点.

Lecture 10: Causal ML.

Correlation $\neq$ Causation

1. Causal Inference. 因果推断.

with observational studies
with interventions $\rightarrow$ 可主动施加干预.

Average Treatment Effect.

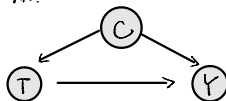
- $Y_i(1)$: 个体 i 在接受 treatment 时的潜在结果
- $Y_i(0)$: 个体 i 在不接受 treatment 时的潜在结果

那么个体的因果效应就是:

$$Y_i(1) - Y_i(0)$$

课件进一步定义:

$$ATE := \mathbb{E}[Y(1) - Y(0)]$$



那么 C 是后门路径上的混杂变量, 需要调整. 课件给出的结论是:

$$\mathbb{E}[Y \mid \text{do}(T=t)] = \mathbb{E}_C \mathbb{E}[Y \mid t, C] = \sum_c \mathbb{E}[Y \mid t, c] P(c)$$

严格执行 $T=t$.

并把这个方法称为 backdoor adjustment. DDA4210 Lecture 10 Causal ML

2. Causal Discovery.

Given dataset D , derive a causal graph.

DAG: Directed Acyclic Graph. 有向无环图.

NOTES

假设每个变量都满足

$$x_j := f_j(x_{\pi_j}) + e_j, \quad e_j \sim \mathcal{N}(0, \sigma_j^2)$$

其中:

- π_j^f 表示节点 j 的父节点集合
- $x_{\pi_j^f}$ 表示父节点对应变量
- f_j 表示生成机制
- e_j 是噪声项 DDA4210 Lecture 10 Causal ML

每个变量都由它的直接原因 + 噪声生成.

一个矩阵 $W \in \mathbb{R}^{d \times d}$ 是 DAG, 当且仅当

$$\text{trace}(e^{W \odot W}) - d = 0$$

其中 $W \odot W$ 表示逐元素平方, e^M 是矩阵指数. DDA4210 Lecture 10 Causal ML

优化:

$$\min_W \frac{1}{2n} \|X - XW\|_F^2 + \lambda \|W\|_1 \quad \text{s.t.} \quad \text{trace}(e^{W \odot W}) - d = 0$$

第一项:

$$\frac{1}{2n} \|X - XW\|_F^2$$

表示重构误差.

意思是: 用其他变量线性解释每个变量.

第二项:

$$\lambda \|W\|_1$$

表示稀疏正则.

因为真实因果图通常比较稀疏, 不希望每个变量都连向所有变量.

约束项:

$$\text{trace}(e^{W \odot W}) - d = 0$$

保证学到的是 DAG.

此处需确认 Causal Learning 考否

Lecture 11 : Privacy in ML

1. Privacy Attack

- 1. Membership Inference Attack (成员推断攻击)
- 2. Unintended Memorization Attack (非预期记忆攻击)
- 3. Model Inversion Attack (模型反演攻击)
- 4. Reconstruction Attack (重构攻击)

3.4 重构攻击 Reconstruction Attack

这是整份课件非常重要的一部分，因为它直接导向差分隐私。

定义

攻击者利用公开的聚合统计查询结果，逐步恢复数据库中的私有信息。

课件例子里有一个小数据库：

- Alice: disease = 1
- Bob: disease = 0
- Charlie: disease = 1
- David: disease = 0
- Ever: disease = 1

攻击者可以问：

- “Alice 或 Charlie 或 David 中，有几个人得病？”
- 答案是 2。

如果再设计一个额外查询，就可能推出 Charlie 的疾病状态。 DDA4210 Lecture 11 DP and FL11

关键思想

即使系统只回答“总共有多少人满足条件”，看起来没暴露单个人数据，多个查询联合起来仍可能恢复个体数据。

这为什么重要

这说明：

只发布 aggregate statistics (聚合统计) 也不一定安全。

3.1 成员推断攻击 Membership Inference

定义

攻击者要判断某个样本 x 是否在训练集 D 中。

为什么能做到

因为模型对“见过的训练样本”往往更自信，输出概率更尖锐、更高。

直觉例子

假设一个分类器对某图片输出：

- 猫: 0.999
- 狗: 0.001

而对别的类似图片只输出：

- 猫: 0.70
- 狗: 0.30

攻击者就可能怀疑前者在训练集中出现过。

本质

这是在利用模型输出分布中的“过拟合痕迹”。

3.2 非预期记忆攻击 Unintended Memorization

定义

模型把训练数据里的某些敏感字符串直接记住了。攻击者通过精心构造 prompt 把它诱导出来。

课件举的例子是：

“Alice’s SSN is ???-??-7452”

让语音模型补全问号。 DDA4210 Lecture 11 DP and FL11

3.3 模型反演攻击 Model Inversion

定义

从训练好的模型中恢复训练数据的某些特征，甚至恢复原始样本近似形态。

课件给出的例子是：对脸部识别分类器微生式模型反演，重建出接近真实训练人脸的图片。 DDA4210 Lecture 11 DP and FL11

直觉

模型学到“某个人长什么样”，攻击者反过来利用模型，把这个“内部表征”解码成一张脸。

本质

模型参数里不仅有泛化知识，也可能含有可逆的个体信息。

2. Random Response .

5.1 基本机制

课件用了“你是否逃过税”的敏感调查问题。机制如下： DDA4210 Lecture 11 DP and FL11

- 1. 先抛一次硬币
 - 正面：如实回答
 - 反面：再抛一次
- 2. 第二次抛硬币
 - 正面：回答 Yes
 - 反面：回答 No

因此：

- 如果真实答案是 Yes，则最终回答 Yes 的概率是 3/4
- 如果真实答案是 No，则最终回答 Yes 的概率是 1/4

5.2 为什么这能保护隐私

因为单个回答不再直接等于真实情况。

如果某人回答 Yes，我们不能确信：

- 他真的做过
- 还是随机机制让他回答了 Yes

所以攻击者的后验信念只会小幅改变。

课件给了一个贝叶斯例子：调查员原本认为 Bob 逃税概率是 0.02，看到他回答 Yes 后，后验只升到约 0.058，而不是接近 1。 DDA4210 Lecture 11 DP and FL11

直觉总结

随机性让单个个体“藏在噪声里”。

代价是精度下降。

3. Differential Privacy. 差分隐私.

6.1 场景设定

课件把 DP 的参与者分成三部分： DDA4210 Lecture 11 DP and FL11

- Database D : 包含个体数据
- Curator (可信数据管理端)
- Analyst (分析者)

分析者不断提问，数据库用某个随机机制 mechanism 回答。

目标是：

- 让回答有用
- 但不能让分析者从回答中推断出某个个体是否在数据中，或者其敏感属性是什么

6.2 邻近数据集 Neighbouring Datasets

两个数据库 D_1, D_2 如果只有一条记录不同，就叫邻近数据集。

DDA4210 Lecture 11 DP and FL11

差分隐私的核心思想就是：

如果一个机制对 D_1 和 D_2 的输出分布几乎一样，那么攻击者难以判断某个人的数据是否被包含在数据库中。

6.3 ϵ -DP 定义

课件给出的原始定义是：

一个机制 M 是 ϵ -differentially private，如果对任意两个邻近数据库 D_1, D_2 ，以及在输出集合 R ，都有：

$$\Pr(M(D_1) \in R) \leq e^\epsilon \Pr(M(D_2) \in R)$$

并且反过来也成立，因此任意输出 y 的概率比被夹在：

$$e^{-\epsilon} \leq \frac{\Pr(M(D_1) = y)}{\Pr(M(D_2) = y)} \leq e^\epsilon$$

之间。 DDA4210 Lecture 11 DP and FL11

直觉解释

把某个人的数据放进去或拿出来，不应让结果分布发生大变化。

ϵ 是什么

ϵ 是隐私参数：

越小：隐私越强，输出分布越接近

越大：隐私越弱，允许差别更明显

6.4 (ϵ, δ) -DP

课件还给了放松版定义：

$$\Pr(M(D_1) \in R) \leq e^\epsilon \Pr(M(D_2) \in R) + \delta$$

这里 δ 是一个很小的失败概率。 DDA4210 Lecture 11 DP and FL11

理解

- ϵ 控制“多数情况下的泄露强度
- δ 允许“极少数坏事”发生

这在 Gaussian 机制中尤其常见。

课件总结了 DP 的优点： DDA421

- 有严格数学定义
- 对攻击者能力不做强假设
- 可组合
- 可量化
- 是工业界事实标准

缺点：

- 有时过于严格
- 会影响效用
- 需要设计合适机制

4. Laplace in DP (适合数值输出).

9.1 核心思想

对于数据型查询 $f(D)$ ，直接输出：

$f(D) + z$

其中 z 是 Laplace 噪声，课件给出 Laplace 分布：

$p(y; \mu, b) = \frac{1}{2b} \exp\left(-\left|\frac{y - \mu}{b}\right|\right)$

关键问题就是：噪声尺度 b 应该多大？ [CS44210 Lecture 11 DP and FL11](#)

9.2 全局 ℓ_1 敏感度

课件定义了数据函数 f 的全局 ℓ_1 敏感度：

$\Delta f = \max_{D, D'} \|f(D) - f(D')\|_1$

其中 D, D' 是邻近数据集。 [CS44210 Lecture 11 DP and FL11](#)

它是什么意思

“改动一个人的数据，查询结果最多能变多少”。

为什么它重要

因为：

- 敏感度越高，单个人对结果影响越大
- 要掩盖这份影响，就需要加更大的噪声

9.3 Laplace 机制公式

课件给出的机制是：

$y_i \sim \text{Laplace}\left(f(D)_i, \frac{\Delta f}{\epsilon}\right)$

也就是：

$M_{\text{Lap}}(D) = f(D) + z, \quad z_i \stackrel{\text{iid}}{\sim} \text{Lap}(\Delta f/\epsilon)$

它满足 ϵ -DP。 [CS44210 Lecture 11 DP and FL11](#)

你必须记住的规律

噪声尺度 = 敏感度 / ϵ

也就是：

- 敏感度大 $\rightarrow$ 噪声大
- ϵ 小 (隐私更强) $\rightarrow$ 噪声也大

5. Exponential in DP (非数值输出).

10.1 核心思想

我们定义一个打分函数/效用函数：

$u(D, o)$

表示：在数据集 D 下，输出 o 有多好。 [CS44210 Lecture 11 DP and FL11](#)

输出空间 O 可以是离散选项集合，比如：

- Google
- GitHub
- Jingdong

Exponential 机制不再“给最优答案直接加噪”，而是：

按分数高低随机抽样，分数越高，被选中的概率越大。

10.3 指数机制公式

输出 $o \in O$ 的概率为：

$Pr[o] \propto \exp\left(\frac{\epsilon u(D, o)}{2\Delta u}\right)$

课件明确指出，这个机制满足 ϵ -DP。 [CS44210 Lecture 11 DP and FL11](#)

直觉

- 分数越高，指数项越大
- 被选中概率越高
- 但不是一定选最好那个，因此能保护隐私

和 softmax 的关系

这本质上就是一个“带温度的 softmax / Boltzmann 分布”。

10.2 Score sensitivity

课件定义打分函数敏感度：

$\Delta u = \max_{o \in O} \max_{D, D'} |u(D, o) - u(D', o)|$

[CS44210 Lecture 11 DP and FL11](#)

这个量表示：

改动一个人的数据后，同一个候选输出 o 的分数最多变化多少。

6. Federate Learning 联邦学习.

Lecture 12: Fairness in ML.

1.

Lecture 13: Explainable AI (XAI)

1. Motivation: 预测准确性 ≠ 人能理解.

Transparency	模型训练/预测过程能否被清晰描述.
Interpretability	人能否理解模型如何做决定.
Explainability	能否给出理由.

White Box / Black Box Model.

Removal-based Explanations. 如果一个特征很重要, 删去它后, 输出应发生明显变化.

2. Permutation Test. 置换特征重要性. → 全局解释.

通过随机打乱某个特征值, 看模型性能下降多少, 评估特征重要性.

① 用原始数据评估 ② 一次只打乱一个特征. ③ 再评估性能下降量

$$a_i = \text{Acc}(\text{Original}) - \text{Acc}(\pi_i \text{Corrupted})$$

优: 适用于任何模型, 特征. 速度快.

劣: 无理论证明. 无法处理相关特征.

3. Occlusion 遮挡法

用于深度学习图像模型. 局部解释.

对一张图片, 逐步把某些区域遮住 (比如设为 0 或灰色块), 观察分类分数如何变化.
如果遮住某块区域后, 目标类别分数下降很多, 说明那块区域重要.

[OCDA210 Lecture 13 Explainable AI](#)

课件给出的公式本质上就是:

$$a_i = f_{\theta}(x_1, \dots, x_i, \dots, x_d) - f_{\theta}(x_1, \dots, 0, \dots, x_d)$$

也就是:

原始预测分数 减去 把该位置遮住后的预测分数. [OCDA210 Lecture 13 Explainable AI](#)

优: 适用于任何模型. 理论上适用于任何数据. 易于实现.

劣: 耗时长.

4. General Framework

8.1 Feature removal (如何移除特征)

例如:

1. 设置默认值 (如 0)
2. 生成随机值
3. 为不同特征子集训练不同模型
4. 使用模型预测和丢失特征时的模型
5. 用任务相关函数 (如图像预测) [OCDA210 Lecture 13 Explainable AI](#)

8.2 Model behavior (看什么变化)

可以看:

- 单个模型变化
- 单个样本损失变化 [OCDA210 Lecture 13 Explainable AI](#)
- 整个数据集损失变化

8.3 Summary technique (怎样总结为解释)

例如:

- 给每个特征一个分数
- 选出高价值子集 [OCDA210 Lecture 13 Explainable AI](#)
- 训练一个线性代理模型

这很重要, 因为它告诉你:

Permutation, Occlusion, LIME, SHAP, 并不是互相完全无关的方法, 而是同一个“移除-观察-总结”范式下的不同实现. [OCDA210 Lecture 13 Explainable AI](#)

3. LIME

虽然原始模型f很复杂，但在某个样本x附近，可能可以用一个简单模型g近似它。

$$\hat{z}(x) = \underset{g}{\operatorname{argmin}} L(f, g, \pi_x) + J_2(g)$$

f: 原始黑盒模型, g: 简单模型, L: 要求g在x附近尽可能拟合f, π_x : 邻近样本x越近, 权重越大, $J_2(g)$: 惩罚复杂度

- 1 Given x , form x' .
- 2 Generate z' : sample instances around x' by drawing nonzero elements of x' uniformly at random (where the number of such draws is also uniformly sampled).
- 3 Map each z' to the original space of x to obtain z .
- 5 Solve, e.g., $\min_w \sum_{(z, x')} \exp(-\|x - z\|^2 / \sigma^2) (f(z) - w^T x')^2 + \lambda \|w\|_1$
- 6 Use w to explain: a larger w_i means the corresponding feature i has a higher contribution to the positive prediction.

注: 模型无关, 可解释单一样本, 直观, 任意数据而用

6. SHAP

Shapley Value.

在博弈论里, 假设玩家集合是 $D = \{1, \dots, d\}$.
任意一个特征子集 $S \subseteq D$ 组成一个联盟, 这个联盟有一个价值 $v(S)$.

□ DDA4210 Lecture 13 Explainer AI

Shapley value 计算第 i 个玩家的贡献方式是:

- 枚举所有不包含 i 的联盟 S ;
- 看 i 加入这个联盟前后, 价值增加多少:

$$v(S \cup \{i\}) - v(S)$$

- 再对所有有这种边际贡献加权平均. □ DDA4210 Lecture 13 Explainer AI

这其实是在回答:

平均而言, 当特征 i 加入各种不同特征组合时, 能带来多少额外贡献?

这比“只看单独加一个特征”的方法更全面, 因为它考虑了交互作用。

$$\phi_i(v) = \sum_{S \subseteq D \setminus i} \frac{|S|! (d - |S| - 1)!}{d!} [v(S \cup \{i\}) - v(S)]$$

在XAI中, 该量 = 特征, 收益 = 模型输出的分数, Shapley value = 每个特征对当前预测的贡献。

为: 计算复杂度 $O(2^d)$, $\Rightarrow$ Kernel SHAP

Kernel SHAP = Linear LIME + Shapley Value.

意思是:

- 它也像 LIME 一样, 用局部线性模型做拟合;
- 但它不是随便选权重, 而是用特殊的核权重 $\pi(S)$;
- 这个权重设计能让线性模型系数正好对应 Shapley values.

□ DDA4210 Lecture 13 Explainer AI

这很重要, 因为它揭示了:

LIME 和 SHAP 不是对立关系, Kernel SHAP 可以看成是在 LIME 的框架里加入了博弈论约束。

• Recall LIME

$$\min_g \sum_{x'} \pi_x(x') (f(h_x(x')) - g(x'))^2 + \Omega(g)$$

- Let $g(x') = w_0 + \sum_{S \subseteq [d]} w_S z'_S$, where $z' \in \{0, 1\}^d$.
- Kernel SHAP solves the following problem:

$$\min_{w_0, \dots, w_S} \sum_{x'} \pi(S) \left(w_0 + \sum_{S \subseteq [d]} w_S (z'_S) \right)^2, \quad \pi(S) = \frac{d-1}{\binom{d}{|S|} |S| (d - |S|)}$$