

2024 Fall: Prepare

Yu Sun

November 14, 2024

1 MLSD 一般思路

1.1 Clarify the questions

1. back ground of the system, 应用场景, 是否是 personalized, etc.
2. 可以获得数据: 数据的形式类型, input output, features, label
3. 可以使用的手段的基建支持
4. 有无任何的使用限制, 数据的敏感性, 机器的总数量和计算能力的瓶颈
5. scale of the system 用户数量, 用户群体, 国家, 地区
6. Perfomance: 延时和准确性的 trade off

1.2 Framing the problem into an ML task

1. ML 目标是什么: 对应的业务问题转换为提升的指标, 然后通过显式 (比如之间建模 like) 和隐式 (比如建模 like 提升 retention, 注意 retention 是留存率, 只对于一段时间之前已经注册的老用户看留下了多少, 而 dau 是对于新+老用户一起的) 的方式提升
2. 输入和输出是什么
3. 选择合适的 ML category: 要首先根据能拿到什么样子的信息决定用什么样的 ML策略: supervised (主要的), Unsupervised, Reinforcement

1.3 Data preparation

1. Data engineer:
 - Source
 - Storage: 关系型 SQL 查询慢但是可以复杂, 非关系型 NoSQL 查询快但是不可以复杂查询

2. Feature engineer

- 缺失值处理: 删除 deletion 和插值 imputation
- Scaling: minimax, standardization, log, etc.
- Discretization: 分桶
- Encoding:
 - (a) 常规特征: Integer encoding, one-hot encoding, embedding learning.
 - (b) 序列特征: 序列 encoding
 - (c) 文字特征: 除了 one-hot, word2vec 是常见的技术:
 - CBOW: 滑动窗口的中间词为 target, 周围词的 one-hot 拼接进 NN $-i$ hidden layer $-i$ pred one hot。Inference 的时候的 hidden layer 就是新的 embedding。
 - Skip gram: 中间不变, 输入输出反过来, 周围词是 target, 中间词的 one-hot 是 input。
 - LLM 的 tokenization 也可以。

1.4 Model development

Dimension	Logistic Regression	Linear Regression	Decision Tree	Gradient Boosted Decision Trees	Random Forests	Support Vector Machines	Naive Bayesian	Neural Networks
Pros	Simple, interpretable	Simple, interpretable	Non-linear boundaries	High accuracy	Reduces overfitting	Works well with high-dimensional data	Works well with small datasets	Handles complex data
Cons	Assumes linearity	Sensitive to outliers	Prone to overfitting	Complex, slow, overfitting to outliers	Slow for large forests	Hard to choose kernel	Assumes independence	Black box, needs tuning
Data Needs	Low	Low	Moderate	Large	Large	Moderate	Low	Large
Training Speed	Fast	Fast	Moderate	Slow	Slow	Slow	Fast	Slow
Hyperparameters	Regularization	None	Max depth	Learning rate, trees	Number of trees	Kernel, C	None	Layers, neurons, learning rate
Hyperparameter Tuning	Grid search	N/A	Grid search	Grid search	Grid search	Grid search	N/A	Grid search, random search
Continual Learning	Limited	Limited	Moderate	Limited(树类的都不太行, 静态的)	Moderate(树类的都不太行, 静态的)	Limited, 也相对静态	Limited	Possible with fine-tuning, 流式学习完全没问题
Compute Needs	Low	Low	Moderate	High	High	High	Low	Very high
Interpretability	High	High	Moderate	Low	Low	Moderate	High	Low

Table 1: 常规算法的优缺点

1. 参数选择:

- Grid search: 常规的参数选择方法，最终使得某些指标（recall， precision 之类的指标最好的）

2. 特征和 label 选取

- 特征：是够和目标对齐，相关性，重要度，填充率，以什么结构进入，维护，复用，后续开发进展
- label: handlabeling 手工标签还是 natural labeling 自然的。分桶，bias，是否系数

3. Loss 的选择: 对应不同的场景，目标，stage 都可以不一样，比如召回的 reward softmax

4. 样本的选取: 是否要用所有的样本，例如精排尾部300，ltr 如果是精排所有 candidate 分数，也不用所有

采样方法	中文名称	采样逻辑描述	公式
Convenience Sampling	便利抽样	研究者选择易于接触的个体，通常用于快速、低成本的数据收集。	无公式
Snowball Sampling	雪球抽样	通过现有样本推荐其他参与者，常用于难以接触的群体，如特定社区。	无公式
Stratified Sampling	分层抽样	将总体划分为多个层次，然后在每个层次中随机选择样本，确保样本的代表性。	n_i 是第 i 层的样本量，公式为: $\frac{N_i}{N} \cdot n$
Reservoir Sampling	水库抽样	在处理流数据时，随机选择固定数量的样本，确保每个样本被选中的概率相同。	$\frac{n}{k}$ （选择第 k 个元素的概率）
Importance Sampling	重要性抽样	根据每个样本的重要性进行加权抽样，常用于估计期望值，减少方差。	$\hat{E}[f(X)] = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}$

Table 2: 不同采样方法及其描述

5. 分布式训练

- Data parallelism: 数据大模型小; model parallelism: 模型大, gpt
- Data parallelis 训练框架: 异步 (Synchronous Training 分别更新) 还是同步训练 (Asynchronous Training, 等最慢的一个)
- 如何分 batch, 是否要进行数据打散, 怎么打散
- 如何将样本分为测试, 训练, k folds

6. 数据不均衡

- Resampling: up down 采样
- Data argumentation: 数据增强
- Loss function 的 weight 更多的给 minor class, i.e., focal loss (因为 estimator 前面有 $\frac{1}{n}$), 所以是 $-\log(p)$ 而不是 $-p\log(p)$

Loss Function	Formula
Cross Entropy Loss	Cross Entropy Loss(p) = $-\log(p)$
Focal Loss	Focal Loss(p_t) = $-\alpha_t(1 - p_t)^\gamma \log(p_t)$

Table 3: Cross Entropy Loss 和 Focal Loss 的公式

- 改变样本形式, 变成 listwise contrastive loss 也可以

7. Training from scratch 还是 fine tuning: 取决于应用场景和要解决的问题, 算力

8. 附加 TBD

1.5 Evaluation

- Offline
- Online
- Bias and unfairness 怎么办:
 1. 数据审查与预处理:
 - 对训练数据进行审查, 识别和移除不平衡的性别分布或歧视性样本。确保数据集中包含各种性别的代表性样本。
 - 进行数据增强, 生成更多的少数性别样本, 以平衡性别比例。
 2. 算法选择与调整:
 - 选择对性别偏见不敏感的算法, 如决策树或集成学习方法 (如随机森林)。
 - 使用公平性约束或正则化技术, 减少模型对性别的依赖。
 3. 公平性评估:

Task	Offline Metrics	Definition	Formula
Classification	Precision	正确预测的正例占所有预测为正例的比例	$Precision = \frac{TP}{TP+FP}$
	Recall	正确预测的正例占所有实际正例的比例	$Recall = \frac{TP}{TP+FN}$
	F1 Score	精确率和召回率的调和平均数	$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
	Accuracy	正确预测的样本占总样本的比例	$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
	ROC-AUC	曲线下面积，衡量分类器性能	$AUC = \int_0^1 TPR(FPR) dFPR$
	PR-AUC	精确率与召回率的曲线下面积	$PR-AUC = \int_0^1 Precision(Recall) dRecall$
Regression	Confusion Matrix	实际值与预测值的矩阵对比	-
	MSE	均方误差，预测值与实际值之差的平方的平均值	$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
	MAE	平均绝对误差，预测值与实际值之差的绝对值的平均值	$MAE = \frac{1}{n} \sum_{i=1}^n y_i - \hat{y}_i $
Ranking	RMSE	均方根误差，MSE 的平方根	$RMSE = \sqrt{MSE}$
	Precision@K	前 K 个推荐中相关项目的比例	$Precision@K = \frac{\text{Relevant items in top K}}{K}$
	Recall@K	前 K 个推荐中所有相关项目的比例	$Recall@K = \frac{\text{Relevant items in top K}}{\text{Total relevant items}}$
	MRR	平均倒排排名，相关项的平均排名的倒数	$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{rank_i}$
	mAP	平均精确率，在多个查询上平均的精确率	$mAP = \frac{1}{Q} \sum_{q=1}^Q AP(q)$
	nDCG	归一化折现累计增益，考虑排名顺序的评估指标	$nDCG = \frac{DCG}{IDCG}$
CV	FID	Fréchet Inception Distance，评估生成图像质量的指标	$FID = \ \mu_r - \mu_g\ ^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g})$
	Inception Score	评估生成图像多样性和质量的指标	$IS = \exp(\mathbb{E}_{x \sim p_g}[D_{KL}(p(y x) p(y))])$
NLP	BLEU	用于评估机器翻译质量的指标，基于 n-grams 的匹配	$BLEU = BP \cdot \exp\left(\sum_{n=1}^N w_n \log p_n\right)$
	METEOR	考虑同义词和词形变化的翻译评估指标	$METEOR = \frac{1}{\max(1, P)} \cdot \sum_{g \in \text{matched}} F(g)$
	ROUGE	用于评估自动摘要的指标，基于重叠 n-grams	$ROUGE = \frac{\text{Recall}}{\text{Total n-grams}}$
	CIDEr	考虑上下文信息的图像描述质量评估指标	$CIDEr = \frac{1}{N} \sum_{i=1}^N c(g_i) \cdot \text{IDF}(g_i)$
	SPICE	基于语义的图像描述评估指标	$SPICE = \frac{1}{ D } \sum_{d \in D} \frac{\text{matches}(d)}{\text{predicates}(d)}$

Table 4: 不同任务的离线评估指标及其定义和公式

ab:metrics

- 使用公平性指标（如均衡准确率、均衡预测率等）评估模型在不同性别上的表现。
- 实施后续测试，确保模型在不同人群中的表现一致。

4. 模型校准:

- 在训练后对模型进行校准，调整输出概率，使其在不同性别群体中更为一致。
- 使用 Platt Scaling 或 Isotonic Regression 等技术校正输出概率。

5. 透明性与可解释性:

- 提高模型的可解释性，以了解其在决策中处理性别特征的方式。
- 通过提供透明的文档，增强用户对性别偏见处理方式的信任。

6. 持续监测与反馈:

- 定期监测模型性能，特别是在性别相关任务中的表现，并及时调整。
- 收集用户反馈，了解模型的实际应用中是否存在性别偏见，并进行改进。

1.6 Deployment and Serving

- Could, 自研 cloud, on device
- Model compression:

1. Knowledge distillation: 类似粗排蒸馏精排 cotrain
 2. Pruning: 把不用的 parameters set to 0
 3. Quantization: 精度问题 32 bit
- Test in product:
 - Shadow deployment: 同样的请求打到新的和旧服务，用户推理还是用旧的，新的只是作为开发调研用。
 - Bigbang: 整体停机替换，快但是有风险，场景手机游戏，所有用户在同一时刻需要同一个版本
 - Rolling: 根据流量慢慢替换，比较慢，比如替换推荐策略是可以的
 - Green blue: 搞一套完全相同的服务，新服务只供给开发测试，测试完整后整体迁移到新的服务上。适合部署时间比较慢，需要快速 roll back，大范围的服务更新，而且整体消耗比较大。
 - Canary: 选择一部分机器或者一部分用户做
 - Feature toggle: 只针对某个 feature 的开关是否开启。可以跟上面的四种结合，这个开关针对的某种 deployment 的方式。但是缺点是 toggle debt，要注意清理不用的，记得开发的时候加上新的 toggle。

部署策略	部署方式	风险控制	适用场景
Shadow Deployment	并行运行新旧版本，不影响用户	风险非常低	性能测试、新功能验证，不影响用户体验
Big Bang Deployment	一次性切换到新版本	风险非常高	传统系统或要求简单的项目，但现代软件较少使用
Rolling Deployment	逐步替换部分服务器至新版本	中等风险	微服务、多实例部署，逐步替换以减小风险
Blue-Green Deployment	两个环境，切换流量	低风险	快速切换版本，简化回滚，高可用系统
Canary Deployment	部分用户使用新版本，逐步扩大覆盖范围	低风险	大规模用户验证新版本，便于逐步放大测试

- Example question: How to ensure fast and scalable serving for our machine learning models, we would implement the following technologies and strategies.

1. Model optimization and framework selection:

- Choose lightweight models or compress models (quantization, pruning) for inference efficiency.

- Use inference frameworks like TensorFlow Serving, TorchServe, or ONNX Runtime for high-throughput serving.
2. **Hardware acceleration:**
 - Leverage GPUs (with CUDA or TensorRT) or TPUs to speed up inference for computation-intensive tasks.
 3. **Architecture design:**
 - Use microservice architectures to deploy models independently and enable horizontal scaling.
 - Deploy services using containerization (Docker) and orchestration (Kubernetes) to simplify deployment and scaling.
 - Serverless architectures (AWS Lambda, Google Cloud Functions) can dynamically allocate resources based on traffic.
 4. **Caching and inference optimization:**
 - Use caching mechanisms (e.g., Redis) to store common inference results and avoid redundant computation.
 - Utilize batch processing to handle inference requests in groups, improving resource utilization.
 - Employ asynchronous inference with message queues (e.g., RabbitMQ, Kafka) to handle high concurrency.
 5. **Model management and deployment strategies:**
 - Use a model registry (e.g., MLflow, Sagemaker Model Registry) for version control and dynamic switching.
 - Apply A/B testing or Canary Deployment to validate new models gradually before a full release.
 6. **Scalability:**
 - Implement auto-scaling strategies in cloud environments (e.g., AWS, Google Cloud) to handle peak loads.
 - Use distributed inference frameworks (e.g., Ray Serve or TensorFlow Distributed) for very large models across multiple nodes.
 7. **Monitoring and optimization:**
 - Monitor key metrics (e.g., latency, throughput) with tools like Prometheus and Grafana, and set up alerts to detect performance bottlenecks.
 - Adjust auto-scaling policies based on real-time traffic patterns to ensure smooth operation.

1.7 Monitoring and Infrastructure

两个维度：数值可以看出来的维度和体感看出来的维度

1. Operation metrics: 各个服务各个阶段的各种工程指标
2. Model metrics: 模型的各种指标, 例如各种阶段的打分, 排序源头
3. (optional): 各个业务阶段是不是有强烈耦合

1.8 我自己提出的: 可快速维护和迭代性

两个维度: 数值可以看出来的维度和体感看出来的维度

1. 可快速维护 (Maintainability): 例如链路耦合性
2. 可快速迭代性 (Iterativity): 后续开发和升级
3. 稳固性 (Robustness): 可预防一些恶意攻击

2 经典面试题

2.1 1. CV - Representation Learning—Visual Search System: Pinterest

1. 确认应用场景不需要 personalization 和其他 concern
2. 多用了 CV 信息但是用户交互的信息很少, 模型不怎么更新
3. Training: 对于 CV 图片相似的 representation learning, 可以选择先用 CNN 学习 query, 正样本, 负样本 list 的 embedding; 接着用 softmax 生成 predict label, ground truth 是 $[1, 0, \dots, 0]$, 只有正样本应该是 1, loss 非常朴素的用 CrossEntropy 跟真实 label 算 loss。后续 Rerank 只根据业务逻辑。
4. Office metric: 用一个固定的离线的 evaluation data 去 nDCG 根据排序累积的 relevance 值。(evaluation set 不更新?? recall he rerank 用同一个指标去衡量? 你在逗我???)

2.2 2. CV - Objective Detection – Google Street View Blurring System

1. Two state objective detections: RCNN – 先找边框, 再对边框内的内容进行 multi-class classification
2. Offline metric 是一个融合边框大小和 Precision (Privacy based)
3. 对于 privacy 的东西, 一旦用户给出负反馈, 一定要有机硬隔离和再进模型, 也要有可以放出

2.3 3. CV + NLP- Representation Learning–YouTube Video Search: YouTube

1. 可以拆成两个 search 的子模块对应2种方案

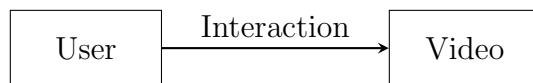
ML Design	Text 部分	CV 部分	Pros n Cons
q to text search + q to CV search, 最后分开融合	text 部分可以选择统计的 1) 倒排2) ElasticSearch 3) Bert or Transformer	双塔, text encoder and video encoder	分阶段可控, 解释能力强, 融合部分可以操作升级
q to (text + CV) search	text 部分可以自己训练 encoder, 也可以 UE	双塔, text encoder and video encoder	提前融合, text + cv encoder 的结构可以在图内融合, 算力要求更大

- 双模块能做的比较多, search 本身也可以比较复杂, 注意模块融合的优劣点的比较

2.4 4. TnS - Harmful Content Detection

2.5 5. Video Recommendation System – Representation Learning

目标是 interaction。可以通过用户已知的信息动作（明确正样本）和潜在有可能带来的所有维度进行拆分（路径非常多，社交，互动，直播）。所有召回可以分为三大类。用内容源头来分



1. 只用 user 信息: 正排, 倒排
2. 只用交互信息: Collaborative based: 跟你相似用户的你没看过的。不需要视频特征, 只用用户交互。Cons: 需要大量交互, Coldstart, niche interest。模型的话老的 Matrix Factorization 也许可以
3. 只用 Video 信息 Content based: 一个交互 item, 找这个item 的相似视频。不需要用户额外的动作, 只用视频特征。但是可能单一而且找“相似”这个事情很有说道。
4.
 1. 种子类
 - 直接动作: 正排, 倒排

- 统计信息: ItemCF, 进阶版的 swing
- 各类 ****2****, i2i, U2i, 相似的定义性

2. 模型类

2.6 6. Event Recommendation System

2.7 7. Ad Click Prediction on Social Platforms

2.8 8. Similar Listing on Vacation Rental Platforms

2.9 9. Personalized News Feed

2.10 10. People You May Know

2.11 title

2.12 title

3 流程图

3.1 中文流程图









图标	含义	说明
	处理动作或操作	使用矩形或圆角矩形表示, 描述具体的操作。
	条件判断	使用菱形表示, 代表需要根据条件作出选择的分支点。
	数据输入或输出	使用平行四边形表示, 表示数据的输入或输出。
	数据库、存储系统	使用圆柱形或堆叠圆盘形表示, 代表持久化的存储。
	模块化流程	用双线矩形表示, 表示一个独立的模块化流程或功能。
	流程的起点和终点	用椭圆表示, 标识流程的开始或结束。
	外部服务或接口	用平行四边形或特定的图标表示, 代表流程外部的系统或服务。
	流程说明	用虚线框或括号状的形状表示, 添加说明或注释。

Table 7: 中文流程图元素及其说明

3.2 英文流程图









Icon	Meaning	Description
	Process action or operation	Represented using a rectangle or rounded rectangle to describe a specific operation.
	Conditional judgment	Represented using a diamond shape, indicating a branching point where a choice needs to be made based on conditions.
	Data input or output	Represented using a parallelogram to indicate the input or output of data.
	Database, storage system	Represented using a cylinder or stacked disks to indicate persistent storage.
	Modular process	Indicated using a double-lined rectangle to represent an independent modular process or function.
	Start and end of the process	Indicated using an ellipse to mark the beginning or end of the process.
	External service or interface	Represented using a parallelogram or specific icon to indicate a system or service external to the process.
	Process description	Indicated using a dashed box or bracket-like shape to add notes or comments.

Table 8: Flowchart Elements and Their Descriptions

4 面试的一般流程

4.1 Step 1. 理清核心问题

不是每个面试官都能用一目了然的方式提问。有的面试官水平差，自己也理不清问题的逻辑。遇到问题很模糊的时候，要尽快理清核心问题。抽象出来，可用信息/输入有哪些，要求的输出是什么样的，这是一个classification的问题，还是regression，还是relevance/matching/ranking？理清楚核心问题，就能判断需要train哪种类型的model，整个pipeline就很容易flow out了。

4.2 Step 2. 白板画图 diagram

typical的解答逻辑包括这几大块：training/testing data, input representation, model, output, evaluation, optimization(parameter estimation). 我一般从model开始画，一个框框摆在中间，这是核心。然后画上游，下游。在这里，只要把框架搭好，告诉面试官，我要讲这些内容，面试官有个心理准备，就可以开始听你讲课了。

4.3 Step 3. 讨论model

为什么我用“讨论”这个词？因为能seriously被考到design的人，都不是entry level。对于更senior的人来说，面试的最好氛围不是你问我答，而是我把我知道的都讲给你听，你看看还有什么想听的。所以你讲的过程中要和面试官互动。要看ta的反应，哪里皱眉

了，哪里表情不轻松了，你就要停下来，问他Is there anywhere that you want me to talk more? 这给面试官一个机会表达自己，也帮助你更好的address面试官的考点。

Model方面，针对task 的类型，propose哪些model可用，把你能想到的都name出来。选择2-3个常用的，比较优劣，然后选择一个大家常用的。不同的model，输入输出可能不一样。所以决定了model，其他的component就很自然的浮现了。这一步，要在你的model框框里，把关键的component列出来，说明它们之间的关系。分析各个model的优劣，可能需要在旁边额外画出model的visualization，比如说到dnn，你就画几层multi perceptron layer，再顺便提一下SGD和ADAM。说到用logistic regression做classification，你就顺手写一下log likelihood，显得你optimization也很懂。说到regularization，你就写写L1 norm和L2 norm。显示你的深度，主要就靠这一步。有时候面试官会告诉你ta想用的model，你就按照ta的来，你也可以在讲解完几个model的优劣后，根据经验自己决定一个model。

4.4 Step 4. Input and output

前面一步把model定下来。根据不同的model，解释一下input 和output的format。比如dnn就有one-hot encoding，这种最好用上embedding，顺便讲一下有什么好处。比如需要自己设计feature的，就重点讲一下有哪些常用的feature。到此，这轮面试的核心你都cover住了，可以得到60分。Step 5和step 6 是能区别ML**和ML老手的部分。如果你做了，答的一般可以再拿20分，答的好可以再拿40分。有ML经验的人在这两个部分，一定要把握。

4.5 Step 5. Data

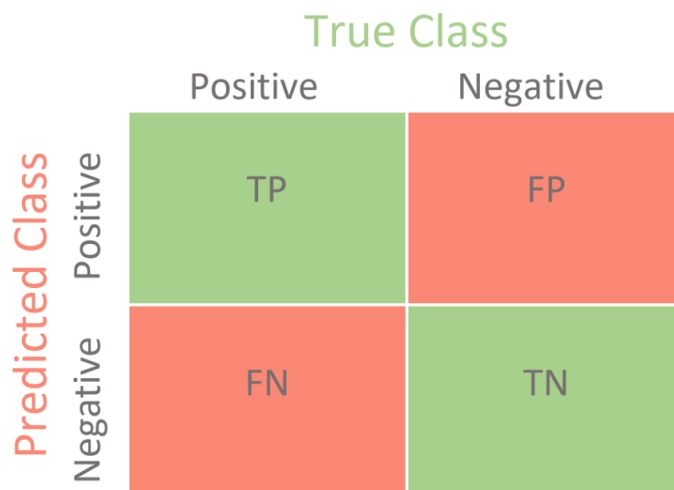
从2个方面 identify data: training + label, testing + ground truth。1. propose 可用的data 来源+data format。2. how to preprocess data → make training data, how to build/create label, etc. 完成建模。根据具体的问题，data的solution可以非常creative。甚至Training data和testing有时候不一致。比如language model方面的问题：decide一个twitter post是什么语言？training 可能就用wikipedia，testing则可以收集user data或者platform-specific的data，这时候也需要指明testing如何get ground truth(testing label)。

4.6 Step 6. Evaluation 模型评估

- **Precision:** $TP/(TP+FP)$ 预测成功的概率
- **Recall:** $TP/(TP+FN)$ actual positive 被识别成功地概率 (=TPR)
- **F1:** $2/(1/Precision+1/Recall)$ Precision 和 Recall 的调和平均。
- **Accuracy:** $(TP + TN)/(TP+TN+FP+FN)$ 所有预测对的比例
- **Confusion Matrix:**

Table 9: Classification 分类模型评估

指标	描述	Scikit-learn函数
Precision	精准度	from <i>sklearn.metrics</i> import <i>precision_score</i>
Recall	召回率	from <i>sklearn.metrics</i> import <i>recall_score</i>
F1	F1值	from <i>sklearn.metrics</i> import <i>f1_score</i>
Accuracy	Accuracy 值	from <i>sklearn.metrics</i> import <i>accuracy_score</i>
Confusion Matrix	混淆矩阵	from <i>sklearn.metrics</i> import <i>confusion_matrix</i>
ROC	ROC曲线	from <i>sklearn.metrics</i> import <i>roc</i>
AUC	ROC曲线下的面积	from <i>sklearn.metrics</i> import <i>auc</i>
Calibration	模型输出概率的校准	from <i>sklearn.calibration</i> import <i>CalibratedClassifierCV</i>

Figure 1: \mathcal{S}^{π}

- ROC(Receiver Operating Characteristic):
 1. **Sensitivity** TPR (True Postive Rate): $TP/(TP+FN)$, 代表分类器预测的正类中实际正实例占有所有正实例的比例。
 2. **1-Specificity** FPR (False Postive Rate): $FP/(FP+TN)$, 代表分类器预测的正类中实际负实例占有所有负实例的比例。
 3. **Specificity** TNR (True Negative Rate): $TN/(FP+TN)$,代表分类器预测的负类中实际负实例占有所有负实例的比例, $TNR=1-FPR$ 。
- AUC(Area Under the Curve): AUC用于衡量“二分类问题”机器学习算法性能(泛化能力)。
- UAUC(Unbiased Area Under the Curve): 在每个用户维度做 AUC, 在某一类用户主导的情况下, 例如高热远多余低热用户的时候, UAUC 效果显然更好。

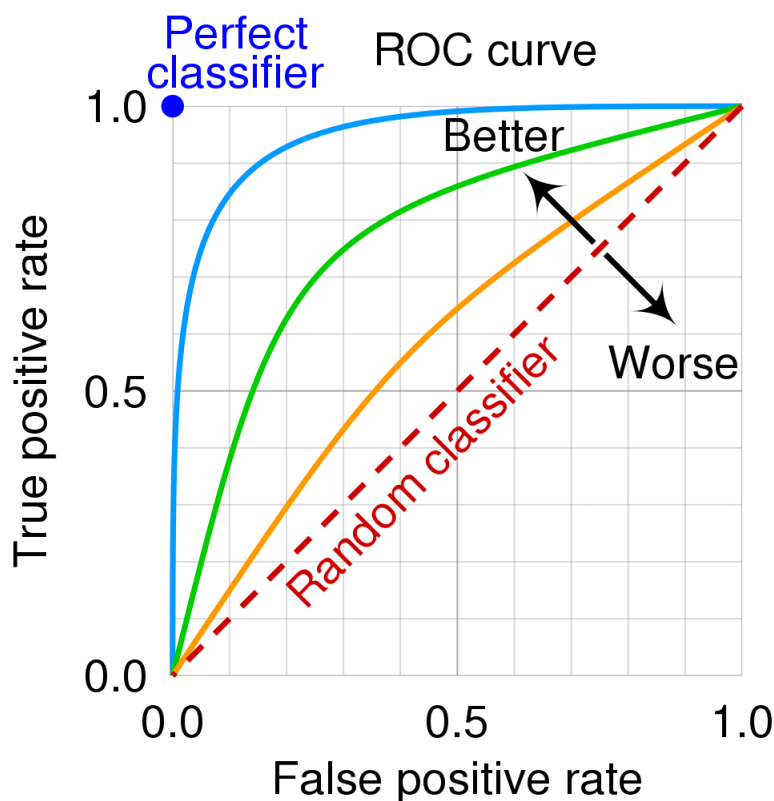


Figure 2: ROC and AUC

- ROC 是就是 x 假阳率 vs y 真阳率
- Recall 和 Precision 是互相拮抗的，调整方法
 1. 阈值：增大阈值，增加 Precision，减少 Recall；减小阈值，增加 Recall，减少 Precision
 2. 样本：如果新样本被正确预测为正类，精确率可能增加，召回率会增加；如果新样本被错误预测为负类（且假设不影响 FP），则精确率可能降低，召回率保持不变。
 3. 更改 loss function 的形式有可能有用
- Calibration: 以下是校准（calibration）与常见校准偏差解法的公式及描述：校准的目的是将分类器的预测概率与实际标签的分布进行匹配。即使分类器经过良好的训练，其输出的概率（如 C）仍可能存在偏差。以 SVM 为例，SVM 的输出并不是预测的概率，因此，尽管预测分类可能准确，预测的平均值与真实正样本的占比之间仍会存在差距。因此，为了将预测值用作概率，需要进行校准。校准的逻辑类似于残差补全的思想，尽管在 ResNet 或 XGBoost 中，模型的提升是通过学习残差使损失逐渐减小，而校准的过程则是通过调整输出概率以提高其与真实标签的匹配度。换句话说，校准旨在将输出的概率分布拟合为真实值。常见的方法包括 Platt Scaling 和 Isotonic Regression，前者通过将 SVM 的对

数几率拟合为一个逻辑函数，后者则通过逐段常数的方式调整输出概率，从而将 SVM 等模型的输出转换为可靠的概率预测。

1. 校准公式:

校准曲线用于衡量分类器预测概率与实际观察到的结果的匹配度。如果模型被良好校准，预测事件发生的概率将与实际发生的概率一致。

$$C(f(x)) = P(Y = 1 | \hat{p} = f(x)) \quad (1)$$

其中:

- $C(f(x))$ 是校准函数。
- $f(x)$ 是预测的概率。
- $P(Y = 1 | \hat{p} = f(x))$ 表示给定预测概率时事件发生的实际概率。

2. 校准偏差的常见解决方法:

(a) **Platt 缩放 (Platt Scaling)** :

$$\hat{p} = \frac{1}{1 + \exp(Az + B)} \quad (2)$$

其中:

- A 和 B 是通过训练数据学到的参数。
- z 是分类器的对数几率。

(b) **等分回归 (Isotonic Regression)** :

$$\hat{p}_{\text{calibrated}} = g(\hat{p}) \quad (3)$$

其中 $g(\hat{p})$ 是一个逐段常数、单调递增的函数，具体形式如下:

$$g(\hat{p}) = \begin{cases} a_1, & \text{if } \hat{p} \in [p_0, p_1) \\ a_2, & \text{if } \hat{p} \in [p_1, p_2) \\ \vdots & \vdots \\ a_n, & \text{if } \hat{p} \in [p_{n-1}, p_n] \end{cases} \quad (4)$$

其中 p_0, p_1, \dots, p_n 是分段点, a_1, a_2, \dots, a_n 是对应的输出值。

(c) **Beta 校准 (Beta Calibration)** :

$$\hat{p}_{\text{beta}} = \frac{1}{1 + \frac{1-\hat{p}}{\hat{p}} \cdot \exp(A + B \cdot \log(\hat{p}) + C \cdot \log(1 - \hat{p}))} \quad (5)$$

其中 A 、 B 和 C 是通过训练数据拟合的参数。

Table 10: Regression 分类模型评估

指标	描述	Scikit-learn函数
Mean Square Error (MSE, RMSE)	平均方差	from <i>sklearn.metrics</i> import <i>mean_squared_error</i>
Absolute Error (MAE, RAE)	绝对误差	from <i>sklearn.metrics</i> import <i>mean_absolute_error</i> or <i>mean_absolute_percentage_error</i>
R-Squared	R平方值	from <i>sklearn.metrics</i> import <i>r2_score</i>

5 统计基础知识

5.1 Bootstrap

自助法, 拔靴法 Bootstrap Method, Bootstrapping或自助抽样法, 均匀的有放回抽样 (sample size n , 做 n 次有放回的抽样, 得到新的数据集), 根本原因在于如果总体是正态分布的 (百度自助法), 那么sample distribution 也是正态分布的。根据 $(1 - \frac{1}{n})^n \rightarrow \frac{1}{e} \approx 0.368$, 那么大概会有1/3 的样本不会出现在训练集里。

- 跟cross validation 一样, 是模型检验的一种方式
- 优点: 每次1/3 当成 test set
- 缺点: 这样产生的训练集的数据分布和原数据集的不一样了, 只是在当数据集非常大, 且高斯分布的时候会近似, 会引入估计偏差。
- 用途: 自助法在数据集较小, 难以有效划分训练集/验证集时很有用; 此外, 自助法能从初始数据集中产生多个不同的训练集, 这对集成学习(bagging (bootstrap aggregating的缩写, 也称作“套袋法”))等方法有很大的好处。
- 总结: Bootstrapping通过重复抽样, 避免了Cross Validation造成的样本减少的问题。其次, Bootstrapping也可以用于随机创造数据。比如, 随机森林算法就是从原始训练数据中, 用bootstrap sampling的方法有放回地随机抽取 k 个新的自助样本集, 并由此构建 k 棵分类回归树。但由于其训练集有重复数据, 这会改变数据的分布, 因而导致训练结果有估计偏差, 因此这种方法不是很常用, 除非数据量真的很少。
- [boosting 博客](#), [Cross Validation 博客](#)

6 特征工程

- 编码方式 [【机器学习】特征工程中常见的特征编码](#)
 1. One-hot encoder: 特征多的时候维度大 (可以尝试用 PCA 降维度), 维度固定时有冲突, 而且不可学习; 但是对于不可比较的类别特征, 比如 country 就还行

2. Label encoder: 变成 0, 1, 2, etc., 没有标签的具体含义, 适用于支持类比性算法的模型, XGboot
3. Ordinal encoder: 拍个顺, 本科0, 研究生1, 博士2 等等, 依赖先验知识
4. Frequence encoder: 出现几次就用几, 对于离群值敏感, 而且容易出现冲突
5. Mean encoder: target 频次的 mean, 也容易出现冲突
6. 分箱取整缩放

7 Generative VS discriminative

- Generative: learn $\mathbb{P}(x, y)$, i.e. GDA, Native bayes. 学习联合分布
- Discriminative: learn $\mathbb{P}(y|x)$, Regression, SVMs. 学习边界
- Generative VS discriminative
- why generative mode 更不容 overfit
- Discriminative: 直接学习 $p(y|x)$, 学习 decision boundary, i.e. Regression, SVM
- Generative: 学习 $p(x|y)$ 然后用bayesian 推出 $p(y|x)$, 学习的其实是 data 的分布, i.e. GDA, Naive Bayes

7.1 Generative 1. Gaussian Discriminant Analysis

label 0 和 1 服从 $Ber(P)$, 在固定的 label 下所有 data 服从 Gaussian (不同的mean), 对于所有 parameter 的 MLE 都是 canonical unbiased estimator。

7.2 Generative 2. Naive Bayes

- Assumption: The Naive Bayes model supposes that the features of each data point are all independent。为什么一定要是独立的
- MLE 就是 highest count。
- Naive Bayes is widely used for text classification and spam detection.

8 Generalization

模型泛化的能力决定了模型能多有用。

$$Error = Bia + Variance$$

准与确

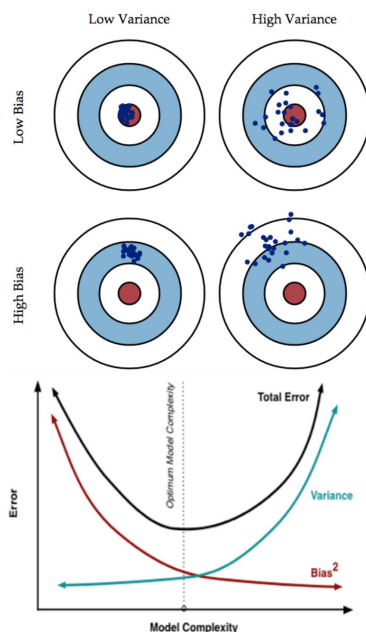


Figure 3: Bias n Variance

- Variance 过大，是模型 overfitting
- 利用 k-fold cross validation 一定会增加 bias，因为利用所有数据训练一定是 bias 最低的，不过相应的 variance 会降低。Trade off is art.
- Bias and Variance trade-off

9 Regularization

- 最主要的目的：解决 overfitting
- 同时满足了 Occam's Razor
- **l1 (Lasso)**: 稀疏化，特征的自动选取，模型解释性
- **l2 (Ridge)**: 不会稀疏，但是 robust 很好，同时对于 outlier 更敏感
- 对于 logistic regression，特殊的解释就是，l1, l2 其实是 prior distribution, l1 对应的是 Lapacian l2 对应的是 Gaussian

10 linear Regression

- 5个基本假设：线性可加性，线性独立，误差 ϵ 的相互独立，误差的方差为常数，而且误差出现正态分布

- 如何利用残差等检验模型是否是线性的

11 聚类

11.1 Kmeans

- Kmeans特点, 更加泛化的 EM特点

12 Logistic Regression

Logistic Regression 详解 Classification, supervised, discriminative, Linear regression 是 maximal entropy 在二分的特例, loss 是 $-\log \text{likelihood}$, 其实就是 logistic loss(cross entropy)。参数模型。

- 逻辑回归模型中的最大化似然函数与最小化损失函数的等价性: 逻辑回归用于二分类问题, 其输出为样本属于某一类的概率。假设我们有一个样本特征 x 和其对应的标签 $y \in \{0, 1\}$, 逻辑回归模型的形式为:

$$P(y = 1|x) = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}} \quad (6)$$

这里, w 是模型的权重, σ 是逻辑函数。

- 最大化似然函数: 似然函数定义为在给定参数 w 的情况下, 观测到的样本数据的概率。对于 n 个样本, 其似然函数可以表示为:

$$L(w) = \prod_{i=1}^n P(y_i|x_i) = \prod_{i=1}^n \sigma(w^T x_i)^{y_i} (1 - \sigma(w^T x_i))^{1-y_i} \quad (7)$$

为了方便计算, 我们通常取对数, 得到对数似然函数:

$$\ell(w) = \log(L(w)) = \sum_{i=1}^n (y_i \log(\sigma(w^T x_i)) + (1 - y_i) \log(1 - \sigma(w^T x_i))) \quad (8)$$

- 最小化损失函: 逻辑回归中的损失函数通常定义为负对数似然损失 (Cross-Entropy Loss) :

$$J(w) = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\sigma(w^T x_i)) + (1 - y_i) \log(1 - \sigma(w^T x_i))) \quad (9)$$

- 等价关系的解释: - **最大化似然与最小化损失**: 从对数似然函数中可以看出, 最大化似然函数相当于最小化负对数似然损失。因此, 在训练逻辑回归模型时, 最大化似然函数和最小化损失函数是等价的。

- **优化目标**：在优化过程中，最大化 $\ell(w)$ （对数似然）等同于最小化 $J(w)$ （损失函数），因此在使用梯度下降等优化算法时，选择其中一种方法都能得到相同的参数估计。
- **结论**：综上所述，逻辑回归模型中，最大化似然函数和最小化损失函数实际上是两个不同的优化目标，但它们在数学上是等价的。这使得在实现逻辑回归模型时，可以灵活选择优化方法。

12.1 与 SVM 对比

- 相同：

1. 都是分类算法，本质上都是在找最佳分类超平面；
2. 都是监督学习算法；
3. 都是判别式模型，判别模型不关心数据是怎么生成的，它只关心数据之间的差别，然后用差别来简单对给定的一个数据进行分类；
4. 都可以增加不同的正则项。

- 不同：

1. LR 是一个统计的方法，SVM 是一个几何的方法；
2. SVM 的处理方法是只考虑 Support Vectors，也就是和分类最相关的少数点去学习分类器。而逻辑回归通过非线性映射减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重；
3. 损失函数不同：LR 的损失函数是交叉熵，SVM 的损失函数是 HingeLoss，这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。对 HingeLoss 来说，其零区域对应的正是非支持向量的普通样本，从而所有的普通样本都不参与最终超平面的决定，这是支持向量机最大的优势所在，对训练样本数目的依赖大减少，而且提高了训练效率；
4. LR 是参数模型，SVM 是非参数模型，参数模型的前提是假设数据服从某一分布，该分布由一些参数确定（比如正太分布由均值和方差确定），在此基础上构建的模型称为参数模型；非参数模型对于总体的分布不做任何假设，只是知道总体是一个随机变量，其分布是存在的（分布中也可能存在参数），但是无法知道其分布的形式，更不知道分布的相关参数，只有在给定一些样本的条件下，能够依据非参数统计的方法进行推断。所以 LR 受数据分布影响，尤其是样本不均衡时影响很大，需要先做平衡，而 SVM 不直接依赖于分布；
5. LR 可以产生概率，SVM 不能；
6. LR 不依赖样本之间的距离，SVM 是基于距离的；
7. LR 相对来说模型更简单好理解，特别是大规模线性分类时并行计算比较方便。而 SVM 的理解和优化相对来说复杂一些，SVM 转化为对偶问题后，分类只需要计算与少数几个支持向量的距离，这个在进行复杂核函数计算时优势很明显，能够大大简化模型和计算。

13 SVM

SVM 详解 把所有点分成 $x_i^T W + b \geq 1$ 和 $x_i^T W + b \leq -1$ 两部分，并要求两平行直线距离最短

$$L(W, b, \alpha) = \frac{1}{2} \|W\|^2 - \sum_{i=1}^n \alpha_i [y_i (x_i^T W + b) - 1] \quad (10)$$

解为

$$\begin{cases} \hat{W} = \sum_{i \in SV} \hat{\alpha}_i y_i x_i \\ \hat{b} = y_j - \sum_{i \in SV} \hat{\alpha}_i y_i x_i^T x_i \\ f(x) = \text{sign}(\sum_{i \in SV} \hat{\alpha}_i y_i x^T x_i + \hat{b}) \end{cases} \quad (11)$$

只与 support vector 有关。需要的点比较少。决策函数为

- 线性时

$$f(x) = \hat{W}^T x + b \quad (12)$$

- 非线性 kernel 时

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x_i, x) + b \quad (13)$$

13.1 Soft Margin

对于软性间隔问题 (soft margin) :

$$\min_{W, b} \frac{1}{2} \|W\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i (x_i^T W + b)) \quad (14)$$

前一项可以理解为“结构风险(structural risk)”，用来描述所求模型的某些性质(SVM就是要求间隔最大)；第二项称为“经验风险(empirical risk)”

13.2 Nonlinear

Kernel trick. 决策函数可以变为 i.e. 用 $[K(x, z)]$ 去代替 $x^T x$

- 多项式核: $K(x, z) = (x \cdot z + 1)^p$
- 高斯核(RBF: Radial Basis Function, 除了前面的常数项, 类 Gaussian Kernel): $K(x, z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$

13.3 SVM 的 SGD 的实现

- [Pegasos算法](#)
- [机器学习 — 一文看懂SVM算法从原理到实现全解析](#)
- [svm随机次梯度下降算法-pegasos](#)
- [ml-deep pegasos](#)

13.4 总结

- 优点:
 1. 由于SVM是一个凸优化问题，所以求得的解一定是全局最优而不是局部最优。
 2. 不仅适用于线性线性问题还适用于非线性问题(用核技巧)。
 3. 拥有高维样本空间的数据也能用SVM，这是因为数据集的复杂度只取决于支持向量而不是数据集的维度，这在某种意义上避免了“维数灾难”。
 4. 理论基础比较完善(例如神经网络就更像一个黑盒子)。
- 缺点:
 1. 二次规划问题求解将涉及m阶矩阵的计算(m为样本的个数)，因此SVM不适用于超大数据集。(SMO算法可以缓解这个问题)
 2. 只适用于二分类问题。(SVM的推广SVR也适用于回归问题；可以通过多个SVM的组合来解决多分类问题)。

14 分类问题和回归问题的却别

- 分类问题 的 loss 应该是 cross entropy, 这取决于本身分类问题的返回应该是不同类别的概率，那么天生应该比较的是概率分布是否相似。如果用 MSE + softmax 会出现梯度消失的情况

15 优化器的选择

15.1 GD 家族

15.2 SGD 到 Adam

SGD-ADAM SGD, ADAM 主要区别：加入了一阶和二阶动量，一阶解决了陷入local min 和震荡的问题，二阶动量继承了历史上的gradient 改动，自适应，改动越多的我们不希望改动太多。同时二阶动量是一个窗口，避免了出现学习率为0的情况。

1. 核心思路：动量法和 RMSProp

Table 11: GD、SGD 和 Mini-batch SGD 的优缺点

方法	优点	缺点
GD	- 全局精确更新 - 稳定的收敛	- 计算代价高 - 内存占用大 - 收敛速度慢
SGD	- 更新速度快 - 内存占用小 - 更容易跳出局部最优解	- 更新噪声大 - 收敛精度差 - 较小的学习率，并且可能采用学习率衰减
Mini-batch SGD	- 计算效率较高 - 收敛过程较平稳 - 利用硬件并行	- 仍有一定噪声 - 批次大小调整困难

2. 动量法：一阶动量平滑了梯度的方向
3. RMSProp：二阶动量避免了大步长更新带来的不稳定性
4. beta 的滑动窗口平均，使得近期 grad 的影响更大

16 Tree-based and ensemble methods

16.1 Decision Tree

决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构，决策树的属性分裂选择是“贪心”算法，也就是没有回溯的。选择每次让信息熵增加最多的 feature 作为 parent node，同时也可以搜索分裂点

- 优点
 - 可解释性强
 - 可以同时处理数值和类别特征
- 缺点
 - 过度拟合：噪音数据，缺少代表性数据，多重比较
 - 大量的 if else，不好并行
 - 不稳定
- 设定阈值及终止条件
- 可以用 K-Fold Cross Validation
- 划分点的方法：回归树和决策树划分节点的方法

- 树模型可以一定程度上对于不平衡数据不敏感，本质上不平衡数据的问题在于没办法对于小类别有很好的估计。但是树模型可以通过类似信息增益比等划分节点的方式一定程度上注重小类别。
- DT 总结
 1. IC3: 用信息熵的增益
 2. IC4.5: 避免某些feature 下面分类过多，做归一化，信息熵的增益比
 3. CART: 分类问题采用基尼系数来选择最优特征和分裂点，回归问题采用平方误差的总值（总方差）来选择最优特征和分裂点
- 解决过拟合的方法， regularization: pruning DT overfitting
 1. pre-pruning: early stop, max depth, max num of samples
 2. post-pruning: grow to full, then 同时平衡num of node 和 误差 前后剪纸示意图

16.2 Random Forest(Bagging)

步骤

1. 一个样本容量为N的样本，有放回的抽取N次，每次抽取1个，最终形成了N个样本。这选择好了的N个样本用来训练一个决策树，作为决策树根节点处的样本。
2. 当每个样本有M个属性时，在决策树的每个节点需要分裂时，随机从这M个属性中选取m个属性，满足条件 $m \ll M$ 。然后从这m个属性中采用某种策略（比如说信息增益）来选择1个属性作为该节点的分裂属性。
3. 决策树形成过程中每个节点都要按照步骤2来分裂（很容易理解，如果下一次该节点选出来的那一个属性是刚刚其父节点分裂时用过的属性，则该节点已经达到了叶子节点，无须继续分裂了）。一直到不能够再分裂为止。注意整个决策树形成过程中没有进行剪枝。
4. 按照步骤1-3建立大量的决策树，这样就构成了随机森林了。

优点

1. 它可以处理很高维度（特征很多）的数据，并且不用降维，无需做特征选择
2. 它可以判断特征的重要程度
3. 可以判断出不同特征之间的相互影响
4. 不容易过拟合
5. 训练速度比较快，容易做成并行方法
6. 实现起来比较简单

7. 对于不平衡的数据集来说，它可以平衡误差。
8. 如果有很大一部分的特征遗失，仍可以维持准确度。

缺点

1. 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟合。
2. 对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的

16.3 Boosting

步骤

1. 先通过对N个训练数据的学习得到第一个弱分类器 h_1 ;
2. 将 h_1 分错的数据和其他的新数据一起构成一个新的有N个训练数据的样本，通过对这个样本的学习得到第二个弱分类器 h_2 ;
3. 将 h_1 和 h_2 都分错了的数据加上其他的新数据构成另一个新的有N个训练数据的样本，通过对这个样本的学习得到第三个弱分类器 h_3 ;
4. 最终经过提升的强分类器 $h_{final} = \text{Majority Vote}(h_1, h_2, h_3)$ 。即某个数据被分为哪一类要通过 h_1, h_2, h_3 的多数表决。
5. bagging 和 boosting 的优缺点，典型案例 GBDT, 和他的进阶版本 xgboost

16.4 Adaboost

High weights are put on errors to improve at the next boosting step.

17 PCA and ICA

17.1 PCA

1. 对所有的样本 $D_{n \times m} = (x_1, \dots, x_m)$ 进行中心化
2. 计算样本的协方差矩阵 $(XX^T)_{n \times n}$
3. 对矩阵 $XX_{n \times n}^T$ 进行特征值分解
4. 取出最大的 $n' < n$ 个特征值对应的特征向量 $w_1, \dots, w_{n'}$, 将所有的特征向量标准化后，组成特征向量矩阵 $W_{nn'}$ 。
5. 对样本集中的每一个样本 ,转化为新的样本 $(z_i)_{n' \times 1} = W^T x_i$
6. 得到输出样本集 $D' = (z_i, \dots, z_m)_{n' \times m}$
7. PCA 对于稀疏特征有作用，避免了由于数据缺失导致特征系数而影响了模型效果

17.2 ICA: Independent component analysis

步骤 ICA 一般只对Gaussian denoise 比较有效。

18 Unsupervised

Motivation: The goal of unsupervised learning is to find hidden patterns in unlabeled data.

18.1 Expectation-Maximization: EM

步骤 EM

- Expectation: 对于初始的 θ , 算出数据对应的 posterior distribution
- 对于所有cluster, 按权重用 MLE 更新 θ

18.2 K-means

选定K 个簇, 选取中中心, 每一个点选择最近的中心, 划分完更新中心, 知道终止条件打成。

- 需要用户事先指定类簇个数K
- 聚类结果对初始类簇中心的选取较为敏感;
- 容易陷入局部最优;
- 只能发现球型类簇;

18.3 Hierarchical Clustering 层聚类

跟 NJ method 思路一致, 适用于树状结构。

- Single Linkage: 两个组合数据点中距离最近的两个数据点间的距离作为这两个组合数据点的距离。两个不同组数据点可能由于其中的某个极端的数据点距离较近而组合在一起。
- Complete Linkage: 将两个组合数据点中距离最远的两个数据点间的距离作为这两个组合数据点的距离。两个相似组数据点可能由于其中的某个极端的数据点距离较远而分开。
- 平均值, 计算量大, 但是少偏差。

19 Entropy

Entropy

$$H(P, Q) = H(P) + KL(P||Q) \quad (15)$$

19.1 定义

1. **熵 (Entropy)**: 对于离散随机变量 X 的熵定义为:

$$H(X) = - \sum_i p(x_i) \log p(x_i) \quad (16)$$

其中 $p(x_i)$ 是随机变量 X 的概率分布。

2. **交叉熵 (Cross Entropy)**: 随机变量 X 和 Y 的交叉熵定义为:

$$H(p, q) = - \sum_i p(x_i) \log q(x_i) \quad (17)$$

其中 $p(x_i)$ 是真实分布, $q(x_i)$ 是模型分布。

3. **KL 散度 (KL Divergence aka Relative Entropy)**: 随机变量 X 和 Y 之间的 KL 散度定义为:

$$D_{KL}(p \parallel q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)} = H(p, q) - H(p) \quad (18)$$

19.2 Cross Entropy 和 Relative Entropy 的对比

为什么 Logistic Regression 使用交叉熵而不是 KL 散度

1. KL 散度用于衡量两个完整概率分布之间的差异, 而分类问题中的真实标签并不构成概率分布。
2. 交叉熵在分类问题中更直接、更有效地计算模型输出概率与真实标签之间的误差。
3. 使用交叉熵损失简化了优化过程, 与 Logistic 回归的输出形式 (Sigmoid 概率) 高度匹配。

19.3 KL 的正定性

为了证明 KL 散度 $D_{KL}(p \parallel q) \geq 0$, 我们可以使用 Jensen 不等式。以下是证明过程:

1. **定义**: 由于 $q(x_i) > 0$, 我们可以考虑:

$$D_{KL}(p \parallel q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)} = \sum_i p(x_i) (\log p(x_i) - \log q(x_i)) \quad (19)$$

2. **应用 Jensen 不等式**: 设 $f(x) = \log x$, 这是一个凹函数, 因此根据 Jensen 不等式:

$$f\left(\sum_i p(x_i) q(x_i)\right) \geq \sum_i p(x_i) f(q(x_i)) \quad (20)$$

Table 12: Relative Entropy (相对熵) vs Cross Entropy (交叉熵) 总结

指标	KL散度 (相对熵)	交叉熵
定义	衡量两个分布之间的差异, 用于描述模型估计与真实分布的偏差	衡量模型分布与真实分布之间的总损失, 包含模型误差和数据本身的不确定性
公式	$D_{KL}(P \parallel Q) = \sum P(x) \log \frac{P(x)}{Q(x)}$	$H(P, Q) = - \sum P(x) \log Q(x)$
是否包含 $H(P)$	不包含	包含 $H(P)$ 和 $D_{KL}(P \parallel Q)$
应用场景	用于优化模型概率输出, 使其更接近真实分布	在分类问题等监督学习中作为常用的损失函数
直观理解	度量模型估计的分布与真实分布之间的差异	衡量模型预测误差与数据固有不确定性的综合损失

于是我们得到:

$$\log \left(\sum_i p(x_i) q(x_i) \right) \geq \sum_i p(x_i) \log q(x_i) \quad (21)$$

3. 利用上面的不等式: 这表明:

$$D_{KL}(p \parallel q) \geq 0 \quad (22)$$

综上所述, 我们可以得出结论:

$$D_{KL}(p \parallel q) \geq 0 \quad (23)$$

并且等号成立当且仅当 $p = q$ 时。

20 NN

20.1 Activation function

- ReLu over Sigmoid:

- Sigmoid: 梯度消失, 求导缓慢, 0点非对称导致梯度下降不平衡
- ReLU: 稀疏大部分 Neuron 不激活, 求导快, 正部梯度一直为1, 0点虽然不可导但是可以利用单边的导数; 负值如果没有梯度可以用 leaky ReLU 加一个很小的梯度

- 常见激活函数的导数

- Sigmoid 激活函数

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (24)$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \quad (25)$$

– Tanh 激活函数

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1 \quad (26)$$

$$\tanh'(x) = 1 - \tanh^2(x) \quad (27)$$

– ReLU 激活函数

$$\text{ReLU}(x) = \max(0, x) \quad (28)$$

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (29)$$

– Leaky ReLU 激活函数

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \quad (30)$$

其中 α 是一个很小的常数（通常设置为 0.01）。

$$\text{Leaky ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ \alpha & \text{if } x \leq 0 \end{cases} \quad (31)$$

– Softmax 激活函数

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (32)$$

$$\text{softmax}'(z_i) = \text{softmax}(z_i)(1 - \text{softmax}(z_i)) \quad \text{for } i = j \quad (33)$$

$$\text{softmax}'(z_i) = -\text{softmax}(z_i)\text{softmax}(z_j) \quad \text{for } i \neq j \quad (34)$$

20.2 初始化

- 部分参数例如 bias 初始置0的话可能使得网络收敛比较慢
- 如果全部置 0 的话，模型就是对称的，那么本质上就相当于只有一个神经元了！

20.3 Dropout layer

Dropout 为了解决 overfitting 的问题，从 bagging 的角度来说，自网络的ensemble。

20.4 NLP

- 基础知识 word2vec，最基本的利用skip gram, CBOW, 一个是当前词预测周围，一个是周围词汇预测当前 N gram
- RNN and LSTM, RNN 只有短期记忆，LSTM有长期
- seq2seq 利用encoder 和 decoder，但是同样面临遗忘的问题

- attention 机制解决了 seq2seq 的遗忘问题，代价是巨大的计算量，因为每一次解码都与要计算与之前所有 encoder 状态向量的相似度 [注意力机制视频](#)
- [Transformer 相比 RNN 和 LSTM 的优势](#)，更无序，句子整体处理
- [关于 Transformer 的总结](#)

21 NLP 类别

- N-gram: markov 假设，通过统计前后出现pair的频率计算一句话的概率。用于句子生成，文本补全，机器翻译
- BOW: Bag of words, 统计句子里面的词频生成 vector，太稀疏太大还没有前后语义关系
- TF-IDF (Term Frequency-Inverse Document Frequency)：是一种常见的文本表示方法，广泛用于[信息检索](#)和[文本挖掘](#)，特别是文本的特征提取和相似性计算任务。它通过衡量一个词在文档中的重要性来表示词与文本的关联程度。
 - **TF (Term Frequency)**：词频，表示某个词在文档中出现的次数。
 - **IDF (Inverse Document Frequency)**：逆文档频率，表示某个词在整个文档集中出现的频率。IDF 通过降低那些在大多数文档中都频繁出现的词（如“the”、“is”等）的权重，突出那些更能区分文档的词。

对于某个词 t 在文档 d 中的 TF-IDF 计算公式为：

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

其中：

- **TF**：词 t 在文档 d 中出现的频率。
- **IDF**：定义为

$$\text{IDF}(t) = \log \left(\frac{N}{1 + n_t} \right)$$

其中 N 是文档总数， n_t 是包含词 t 的文档数。

- BM25: TF-IDF 升级版，直接计算 query 和文档分数，综合了文档长度
- Bert and Transformer: 可以进行文本分类，语句生成，匹配等更多功能; attention 的 QKV 线性捕捉全局依赖，和 MLP 的特征交叉结合，生成更好的 embedding，然后比较 CLS 的相似性 cos。

22 数学问题， 概率论

1. 20 个人生日不相同的概率是多少？如何估计：准确答案 $1 - \frac{365!}{365^n(365-20)!}$ ，近似的话，首先设 $Y_{i,j}$ 表示 i,j 生日相同，所有的 $Y_{i,j}$ are almost independent, then $Y_{i,j} \sim Ber(1/365)$, sum of bernulli $\rightarrow Binomial(n^2/2, 1/365)$, and Bin 的极限是 $Possion(\lambda)$,

$$\lambda = \mathbb{E}[Pos] \approx \mathbb{E}[Bin] = \frac{n^2}{2} \frac{1}{365} = n^2/730 \quad (35)$$

Then

$$\mathbb{P}[\sum Y_{i,j} = 0] = \exp^{-\lambda} = \exp^{-\frac{n^2}{730}} \quad (36)$$

生日问题， 小数定律

23 推荐系统

- 链路：召回 \rightarrow 粗排 \rightarrow 精排 \rightarrow 重排

– 链路讲解

- 常见的 embedding 手段：w2v, item2v, graph embedding(i.e.w2v to DeepWalk to Node2v to EGES(适合冷启动))

- 召回模型 从NFEP的视角理解召回。

Near——定义向量距离近，取决于不同的召回方式。

Far——定义向量距离远，取决于负样本的采样方式。

Embedding——如何生成embedding，这被归结到用户/物品画像的问题。

Pairwise-loss——召回阶段无法追求预测值与标签值的绝对准确性，所以往往采用Pairwise (user,item+,item-) 追求排序之间的相对准确性。召回策略

1. 以矩阵分解为基础的：矩阵分解详解，实际业务中已经过时
2. 因子分解机(FM): FM 的基本原理，如何将交叉项计算复杂度降到 $O(kn)$ ，以及如何解决了 LR 里面面临的1：特征不够，如需要人工制作特征； 2：稀疏特征的问题。FM 为什么会解决多回路召回的问题
3. 双塔 DSSM 双塔介绍, 双塔在实际业务中的应用
 - 后期融合模型，Pros: 适合召回，快速, Cons: 没有用户特征和物品特征的交互
 - 正负样本选取，pointwise, pairwise and listwise 三种训练 三种训练的优劣
 - 简单负样本：未被召回的，困难负样本：未进入粗排，非常困难负样本：未进入精排
 - 曝光但未点击的样本，只可以作为排序模型的负样本，但不可以作为retrival 的负样本，原因很简单，没被点击的原因只是因为兴趣没有那么高，但不代表没有兴趣！！

- DSSM 实现的时候，用户向量是随时计算的，而物品向量是pre cal的，既平衡了计算难度，同时也符合人是动态变化的，但是物品是相对稳定的

● 推荐模型:

1. LS-PLM(MLR大规模分段线性模型): 阿里曾用，先聚类，再在聚类的基础上对每一类做 LR。 Pros: 快 Cons: 对非线性拟合有限

● 深度网络: 线性模型, 深度模型, 树模型, attention 模型1, 和 线性模型, 深度模型, 树模型, attention 模型2。总结贴 深度学习网络总结, 包括传统的多目标模型和 MMoE

1. embedding: LR 到 FM/FFM/biFFM 充分利用了交叉特征
2. DNN 下的: FNN PNN 和 DeepCrossing 的通用Pros: 抓住了高阶交叉特征, cons: 对于低阶特征利用不足。简单来说, 线性模型以浅层形式直接学习稀疏组合特征权重, 对训练数据中出现过的组合特征具有很好的记忆能力。而深度模型, 稀疏特征被映射成低维稠密embedding向量, 随后在深层全连接网络中获得充分交互, 对具体的特征组合的“记忆”能力会减弱, 但换来了更好的泛化效果。
 - FNN(Factorisation-machine supported Neural Networks, 以 FM 的隐藏层作为输入)
 - PNN(Product-based Neural Network, 在 FNN 的基础上增加了 product layer)
 - DeepCrossing (借助了 ResNet residual unit 的概念, 将传统的 MLP 改为了 residual unit)
3. Wide&Deep: 联合学习, LR(Wide 记忆性) + DNN(Deep, 泛化能力)。Wide 和 Deep 输入值是不同的, 因为 LR 需要人工构造特征。衍生品
 - DeepFM: FM + DNN
 - DeepCross Network(DCN): Cross(多项式快速计算, $O(d)$) + DNN。 $x_{i+1} = x_0 x_i^T W_i + x_i$
 - Neural Factorization Machines(NFM): DNN 会梯度/爆炸和过拟合等问题。NFM的主要改进点是, 引入Bi-Interaction Pooling层, 替代经典DNN的concat层, 在底层增加足够的特征交互信息后, 再馈入到MLP网络做进一步的高阶非线性建模。
4. Attention 机制:
 - AFM: 相比于 NFM, 通过 attention net 计算 attention 权重因子。cons: 缺少高阶交叉信息
5. 精排 LHUC(PPNet Parameter Personalized Net): 从语音识别的 LHUC 出发, 将个性化(UserID) 的embedding 用来放大或者缩小物品(ItemID)的 embedding, 从而实现个性化
6. FiBiNET: SeNet本质上同过压缩再放缩实现对 feature fieldwise 的加权 + Bilinear cross。具体结构为 concatenation + bilinear cross + SeNet bilinear cross

7. 序列模型(Last N):

- 简单的 last N 取平均作为feature, 平均池化, 作为用户特征, 可以放入召回, 粗排, 精排
 - DIN: 利用 attention 机制, 计算候选物品与 last N 的相似度然后做加权平均。因为要用到候选物品只适用于精排
 - SIM: 在DIN 模型的基础上, 试图增加 n, 并且减少last n 中与候选物品完全不相关的物品, 减少计算成本, last N to top K. Hard search 和 soft search 用来找 top K, soft 结果更好, 但是计算成本更大。同时加入物品交互时间信息。
- 推荐链路中的重排: 多样性抽样, 尽量多利用内容特征(CV, NLP)而不是用ID 从双塔中学习的特征, 因为头部现象明显, 对于长尾和新进物品都不好。
 1. MMR: Maximal Marginal Relevance : 通过 λ 控制一遍满足和 query 的相关性, 一遍降低同推荐列表里最相似物品的相关性, 实践中可以通过滑动窗口增加相似性的有效性(如果对于所有物品, 那么相似性很容易是1)。同时可以加入规则约束同时优化。
 2. DPP: Determinantal Point Process local repulsive 的性质, 使得 sampling 的时候不会过于相似。快手加Hulu: **DPP**: 用 $\log(\det V^T V)$ 代替了MMR 中的相似惩罚项(常规的DPP), 创新在于提出了贪心思路下快速的行列式计算
 3. **MMR, DPP 总结 1** 和 **MMR, DPP 总结 2**
 - 冷启动的问题:
 1. 评价指标: 对于 UGC(user generated content) 的平台, 区别于 PGC(platform generated content), 可以通过作者侧指标(渗透率), 用户侧指标(新物品消费指标, 大盘指标 DAU etc), 还有内容指标(高热笔记占比)
 2. 召回通道: ItemCF 不行, 因为没有用户交互, 直接用双塔也不好, 因为用户item embedding 不对
 - 改进双塔, Item embedding 从其他相似物品中学习
 - 通过类目和关键词
 - 聚类召回, 把所有item cluster 然后召回每个感兴趣 cluster 中最新的几个
 - Look alike 人群扩散
 - 业界实例:
 1. Facebook: **FB 2020**
 2. Youtube: **YouTube 2019**
 3. 快手:
 - 快手的 PPNet, 源自 Learning Hidden Unit Contribution(LHUC)
 - 快手加Hulu: **DPP**
 4. 微博 **FiBiNET**
 5. 阿里 **DIN**, **SIM**, **EGES**

24 搜索系统

24.1 文本查询分类

文本查询可以分为头部查询、中间查询和尾部查询。每种情况下通常使用的不同方法如下：

1. 头部查询 (Head Queries) :

- **特征选择和加权**：对于高频查询，通常会使用特征选择技术来识别关键字，并为其分配较高的权重。
- **搜索引擎优化 (SEO)**：应用广泛的SEO策略，以确保这些查询在搜索结果中优先展示。
- **聚合和推荐**：利用历史数据和用户行为进行结果聚合和个性化推荐。

2. 中间查询 (Torso Queries) :

- **语义理解**：使用自然语言处理 (NLP) 技术来理解查询的意图，并提供相关结果。
- **上下文建模**：根据用户的上下文信息（如位置、搜索历史）调整查询结果。
- **混合搜索策略**：结合关键词匹配和语义匹配，提供更相关的搜索结果。

3. 尾部查询 (Tail Queries) :

- **长尾优化**：对于这些低频查询，采用特定的内容推荐策略，以确保即使是冷门查询也能找到相关结果。
- **用户反馈机制**：通过用户的点击率和反馈数据，逐步优化尾部查询的结果。
- **内容扩展**：通过增加相关内容或使用用户生成的内容来增强搜索引擎的结果，以满足更广泛的查询。

25 训练架构

- **Jax** : Google 可以用 cpu 和 gpu 加速

26 链接

- [Amazon 八股文1](#), and [八股文1 noion](#)
- [八股文2](#)
- [最新 Pinterest OA 面经](#)
- [八股文3](#)
- [推荐系统面试集锦](#)