Industry Implementation of Machine Learning

0. Introduction: Establish Correct Technical Concepts

无论是 **传统算法**， **数据结构**， **机器学习**， **深度学习**这些计算机科学的内容在面对应用时，**都是手段**而不是目的，换言之，这四个之间哪个都不是绝对万能的，在面对到不同的场景和不同的问题时，都要结合现实生活中的实际应用情况，来决定使用哪个工具，或者说哪几个工具一起组合使用

To summary: 学习这些技术，不论难度复杂与否，**并没有先进落后之分**，只是发展的时间先后顺序差异，以及**应用场景的差距**。 至于我们现在看到的**误区：Alought 现在许多技术应用，例如 ChatBox, AIGC, 文生图，文生视频... 看似这些很 fasion 的应用遍布在我们的身边，就会产生一种 '能处理这些应用的技术才是先进'技术' 的错觉，认为 DeepLearning 才是最为先进的技术。** 但是实际上并非如此，还有其他我们所 '看不见' 的领域工业级应用，仍然使用着机器学习的经典算法。所以：总结而言深度学习并不是高人一等的技术，也不是最先进的技术，只不过是作为一种新的处理 **特定数据类型** 的新的，可供选择的趁手的工具而已。

这里就不得不让我们再次 **审视深度学习和机器学习的局限性**，虽然它们在很多问题上表现得相当出色，但并不是所有场景都适用。深度学习模型，为了达到最佳性能，常常需要依赖大量的标记数据。在一些领域，例如稀有疾病的医学研究或某些特定领域的学术研究中，**获取大量、高质量的数据几乎是不可能的**。此外，深度学习模型复杂，需要高昂计算成本，而且它们的**"黑盒"性质**使得它们在某些需要高度透明度的应用中，例如法律和医疗决策，变得难以应用

So, 面对现实工业界应用，机器学习仍然尤其存在意义，甚至可以说：绝大情况下，使用深度学习的占比仍然不如机器学习. 接下来就简单介绍一下各个领域中，现在仍然运用机器学习进行

1. Finance Tech (Fin Tech)

- Finance 行业特点
    - (1) 高监管属性，需要可解释性 (LR特征工程很清晰，每个特征的贡献度也可以统计出来)
        - Additional: "为什么LR具有很高的**可解释性？怎么理解模型的可解释性来源？**"
        - Ans: 因为LR的本质是一个 'Linear Model', **核心假设**是特征与目标变量之间存在线性关系。具体来说，逻辑回归通过将特征的加权和传递到sigmoid函数中，来预测事件的发生概率。由于是线性结构，所以 **可以得知每一个特征对应到的权重 (weight系数)，从而可以很直观地得知每一个特征相对的贡献度**
- Finance 领域业务

1.1 风控

(1) Fin Tech 使用机器学习建模最多的场景就是风控。当然风控也要进行细分，主要应用机器学习建模的细分场景如下：
- **信用卡交易反欺诈**
- **信用卡申请反欺诈**
- **贷款申请反欺诈**
- **反洗钱**

上面这些任务的进行，都是使用朴素的 **GBDT (GBM) / (ad: XGBoosting, 优化内存和并行模型)+ LR (till 2024/3)**

1.2 金融产品销售

(2) Fin Sells 使用理财产品、基金产品、保险产品或者邀请用户办理信用卡账单分期
推荐算法，基于 **协同过滤CF算法** .
- **协同过滤CF算法**:
- 基于用户的协同过滤（User-based CF） / 基于物品的协同过滤（Item-based CF）
- **(User-Based) 该 User-Based 法假设** 有多个用户，他们的行为或偏好相似（**用户间的相似性：运用用户对于物品的打分评价来定义相似性，**），那么他们可能会喜欢相似的物品。基于用户的协同过滤就是通过找到与你行为相似的用户，并推荐这些用户喜欢的物品. (**推荐和你相似的人的likes**)
- 核心：相似度的计算

$$sim(U_i, U_j) = \frac{\sum_k r_{ik} r_{jk}}{\sqrt{\sum_k r_{ik}^2} \cdot \sqrt{\sum_k r_{jk}^2}}$$

公式解释：

- **sim(U_i, U_j)**：表示用户 U_i 和用户 U_j 之间的余弦相似度。
- **$\sum_k r_{ik} r_{jk}$**：表示用户 U_i 和用户 U_j 对共同评分物品的评分乘积之和。
- **$\sum_k r_{ik}^2$**：表示用户 U_i 对所有评分物品的评分平方之和。
- **$\sum_k r_{jk}^2$**：表示用户 U_j 对所有评分物品的评分平方之和。
- **r_{ik}**：表示用户 U_i 对物品 k 的评分。
- **r_{jk}**：表示用户 U_j 对物品 k 的评分。

用途： 余弦相似度常用于推荐系统中，衡量用户之间的相似性，从而进行协同过滤推荐。
- **(Item-Based)**: **该 Item-Based 法假设**，如果用户喜欢某个物品，那么他们也很可能喜欢与该物品相似的其他物品
- 这是计算两个物品 i 和 j 的余弦相似度公式:

$$sim(i, j) = \frac{\sum_u r_{ui} r_{uj}}{\sqrt{\sum_u r_{ui}^2} \cdot \sqrt{\sum_u r_{uj}^2}}$$

公式解释：

- **sim(i, j)**：表示物品 i 和物品 j 之间的余弦相似度。
- **$\sum_u r_{ui} r_{uj}$**：表示所有用户对物品 i 和物品 j 的评分乘积之和。
- **$\sum_u r_{ui}^2$**：表示所有用户对物品 i 的评分平方之和。
- **$\sum_u r_{uj}^2$**：表示所有用户对物品 j 的评分平方之和。
- **r_{ui}**：表示用户 u 对物品 i 的评分。
- **r_{uj}**：表示用户 u 对物品 j 的评分。

用途：

这个公式用于计算物品之间的相似性，常用于基于物品的协同过滤推荐系统中。通过计算物品之间的余弦相似度，可以找到与用户已评分物品相似的物品，从而进行推荐。

详细解释：

余弦相似度衡量的是两个向量在多维空间中的夹角余弦值。在这个公式中：

- 每个物品都被表示为一个向量，向量的每个维度对应一个用户对该物品的评分。
- 公式计算了这两个向量之间的夹角余弦值，从而得到物品之间的相似度。

余弦相似度的取值范围是 [-1, 1]：

- 1 表示完全相似。

- -1 表示完全不相似。
- 0 表示没有相关性。
    - 推荐算法的 process: 召回+排序+业务规则
    - 召回（从所有可能的候选项中**筛选出一小部分可能相关的项目**）：利用协同过滤CF算法、FM算法
    - 排序（**根据用户**信息，收集的数据来进行排序内容的推荐）：LR （由于线性模型，模型的权重可以作为特征贡献度...从而有排序功能）
    - 业务规则（确保前面筛选出的推荐内容符合要求，是对召回和排序结果的**进一步筛选和规范**）：

---

2. Media (Text Data Content)

基于内容item的推荐、基于知识图谱的推荐、基于协同过滤算法的推荐。资讯信息物料的推荐，这里面会涉及到Doc2Vec、Lsi等算法，因为涉及到一部分对于物料语义的理解 (推荐系统)

2.1 推荐算法1: 基于内容的推荐（Content-Based Recommendation）

分析内容本身 (内容/文字文段本身的信息) 的特征 (**海量新内容库中，提取到的特征1**). 再结合用户过去感兴趣的内容特征 (**感兴趣的特征2**), 寻找2和1的重合点，从而进行推荐。

- **特征提取**：Text 数据的特征提取，常见的特征提取方法包括了：词袋模型（BoW）、TF-IDF、Word2Vec（单词级别的词数值向量化表示）、Doc2Vec（生成文档级别 / 全文的向量表示）等.
- **相似度计算**: 通过计算上面通过特征提取方法得到的 Feature1 和 Feature2... 找到任两者之间相关性，相关性较高的组合，那么这个组合就是相似度较高的一对组合。常用的相似度计算方法有余弦相似度、欧几里得距离等...

2.2 推荐算法2: 基于协同过滤的推荐（Collaborative Filtering Recommendation）

同于前面的 CF 推荐算法： 两个种类:

- **(1) User-Based**: 用户打分相似性，来推荐别人的物品給你
- **(2) Item-Based**: 基于 '假设用户喜欢物品A, 那么也喜欢和A类似的物品' 先验条件, 计算物品之间的相似性

2.3 推荐算法3: 基于知识图谱的推荐（Knowledge Graph-Based Recommendation）

知识图谱是一种通过图结构来表示实体及其之间关系的方式。在推荐系统中，**知识图谱** (**构建方式**：它通过将用户、物品、属性、类别等信息构建成图结构，) 可以用来捕捉**物品之间的关联**，基于图中的知识推理出用户可能感兴趣的物品。

- **Graph Processing**

1. **构建图谱**：通过将物品的属性、用户信息、物品类别等多维度数据映射为图结构，节点代表实体（如用户、物品、标签等），边代表它们之间的关系（如用户购买物品、物品属于某类别等）。
2. **图谱推理**：通过图谱推理，找到用户感兴趣物品与其他物品的关系，从而进行推荐。例如，基于用户已购买的物品和物品间的关联关系，可以为用户推荐其他相似或相关的物品。

2.4 推荐算法4: 基于LSI的推荐（Latent Semantic Indexing）

一种用于文本数据降维的技术，它通过对文档和词语的矩阵进行奇异值分解（SVD），发现文本中潜在的语义结构。在推荐系统中，LSI可以帮助我们理解物料或资讯的语义层次，发现文本间的隐含主题。

如何运作：

1. **文档-词语矩阵**：首先，根据物料内容构建文档-词语的矩阵，矩阵中的每个元素表示词语在文档中的出现频率。
2. **奇异值分解（SVD）**：对文档-词语矩阵进行奇异值分解，将数据降维，从而提取文本的潜在语义结构。
3. **相似度计算**：基于降维后的文档向量，可以计算不同物料或资讯之间的相似度，推荐相似的内容给用户。

---

3. Retail Field

推荐场景和搜索场景中的排序

3.1 Factorization Machine FM算法

核心思想是通过因子分解（类似于矩阵分解）来**捕捉特征之间的高阶交互关系** .

特征之间的相互关系: **特征交互建模**, FM能够捕捉特征之间的高阶交互关系 关键项：

v_iv_j：特征之间的相互关系$(FM$模型除了能够捕捉到线性关系，还能捕捉特征之间的高阶交互关系$)$

FM算法的基本模型如下：

$$\hat{y} = w_0 + \sum_{i=1}^{N} w_i x_i + \sum_{1 \le i < j \le N} v_i v_j x_i x_j$$

其中：

- w_0 是全局偏置。
- w_i 是每个特征的权重。
- x_i 是第 i 个特征的值。
- v_i 是第 i 个特征的因子向量。
- v_iv_j 是用于捕捉特征 i 和特征 j 之间的交互关系的向量积。

3.2 Logistic Regression Sorting Model 基于逻辑回归的排序算法

在**排序任务**中，我们通常将它用于二分类任务（例如，判断某个物品是否适合推荐给某个用户）。为了进行排序，可以使用**多个二分类逻辑回归子模型**来预测不同物品的得分，并根据得分进行排序。

3.3 Deep FM（深度因子分解机）

旨在提高模型对特征交互的捕捉能力 (利用深度学习的非线性建模能力)，同时避免了传统FM模型中使用手工设计的特征交互的需求.

Deep FM的结构：

$$\hat{y} = \mathrm{FM}(X) + \mathrm{DNN}(X)$$

其中，FM部分学习的是特征之间的高阶交互，而DNN部分学习的是特征的低阶交互。

- **FM部分**：与传统的FM模型类似，通过因子分解来捕捉特征之间的高阶交互。
- **DNN部分**：利用深度神经网络来学习特征的低阶交互和非线性关系。

优势：

- **结合了深度学习和因子分解的优点**：Deep FM能够同时捕捉特征之间的高阶和低阶交互，不仅能处理稀疏数据，还能够利用神经网络的强大能力来捕捉复杂的非线性关系。
- **端到端训练**：Deep FM可以端到端训练，避免了手工特征设计，所有的特征交互都可以自动学习。

4. AutoML

自动机器学习技术，不需要机器学习专家参与建模，**机器全自动完成整个建模过程（从数据处理到模型部署落地）**

4.1 Question

实际工业界落地时，遇到的 **最大问题就是数据治理**。科学家们很多时候80%建模的工作在做数据整合和数据清洗等，然而很多AI公司对外输出AutoML解决方案时，主要提供先进的AutoML算法，但是不负责数据治理。实际开展时会发现甲方公司数据管理一片混乱，建模需要的数据分散在各个地方，统计口径以及字段含义等均没有对齐，导致部署AutoML模型前需要花费大量科学家的人力和甲方对齐数据口径，构建数据流，然后很可能还会出现模型上线后线上线下效果不一致的情况

4.2 组件

一条龙服务， 完全自动化处理整个建模流程

1. **数据预处理**：
   - 自动处理缺失值、异常值检测和处理。
   - 自动化的特征选择和特征工程。
   - 自动化的数据归一化、标准化、离散化等。
2. **模型选择**：
   - 自动选择最适合任务的模型（如回归、分类、聚类等），例如，自动选择决策树、支持向量机、神经网络等模型。
3. **超参数优化**：
   - 自动调节模型的超参数（如学习率、树的深度、正则化参数等）以获得最佳性能。
4. **模型训练和验证**：
   - 自动化的训练过程，自动进行交叉验证、性能评估等。
5. **模型部署**：
   - 在最终模型确定后，自动化地将模型部署到生产环境中。

4.3 常见方法论

1. **模型选择与组合**：
   - 自动化的模型选择算法（如**集成学习**）会自动评估多种模型，选择最合适的模型。
   - **集成学习**（例如，Stacking、Bagging、Boosting等）通过组合多个模型的预测来提高性能。
2. **特征工程**：
   - 特征工程是机器学习中至关重要的一步，AutoML通常会自动选择合适的特征，进行特征生成、转换、编码等。
3. **超参数优化**：
   - 常用的优化方法包括：
     - **网格搜索（Grid Search）**：遍历超参数空间的每一个组合。
     - **随机搜索（Random Search）**：随机选择超参数进行训练。
     - **贝叶斯优化**：利用贝叶斯推断来智能选择超参数。
4. **神经架构搜索（NAS，Neural Architecture Search）**：
   - 通过自动化搜索不同的神经网络架构以找到最适合任务的结构。

4.4 流行的AutoML Framework

1. **Google AutoML**：
    - Google提供了多种AutoML工具，如AutoML Vision、AutoML Tables等，专注于自动化图像、表格数据和文本数据的机器学习模型开发。
    - **TensorFlow AutoML**：Google的TensorFlow提供了一些AutoML工具来优化深度学习模型。
2. **TPOT**（Tree-based Pipeline Optimization Tool）：
    - 一个基于遗传算法的自动化机器学习工具，可以自动化特征工程、模型选择、超参数优化等。
3. **Auto-sklearn**：
    - 基于Scikit-learn的自动化机器学习工具，支持模型选择和超参数调优。它通过贝叶斯优化方法来选择合适的模型。
4. **H2O.ai**：
    - H2O.ai 提供了自动化的机器学习工具，支持机器学习模型的自动选择、训练、调参等。它的AutoML平台H2O AutoML被广泛应用于企业和数据科学项目中。
5. **Microsoft Azure AutoML**：
    - 微软Azure的AutoML服务提供了自动化的机器学习管道，可以自动执行数据预处理、模型选择、超参数调优等。