AI 算法进阶（Advanced AI System）

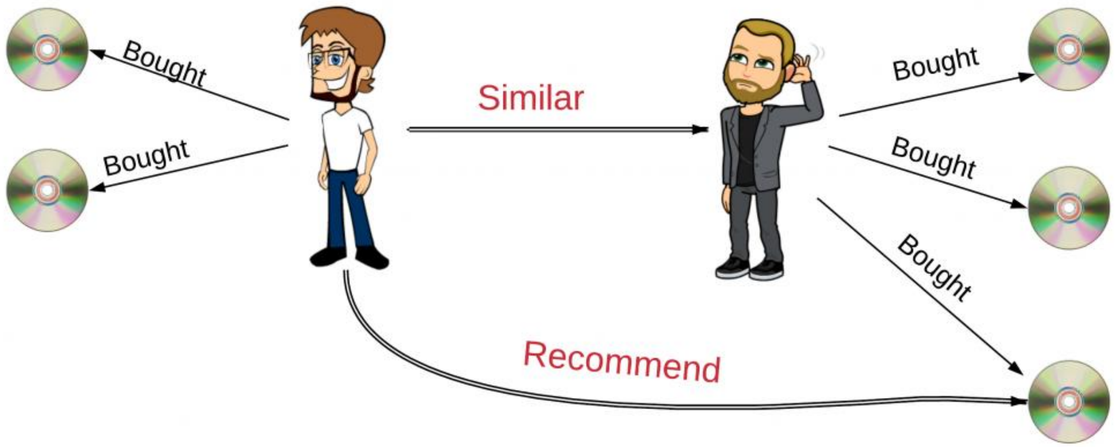
@Tyler

5. 连接主义：感知机与浅层网络

目录

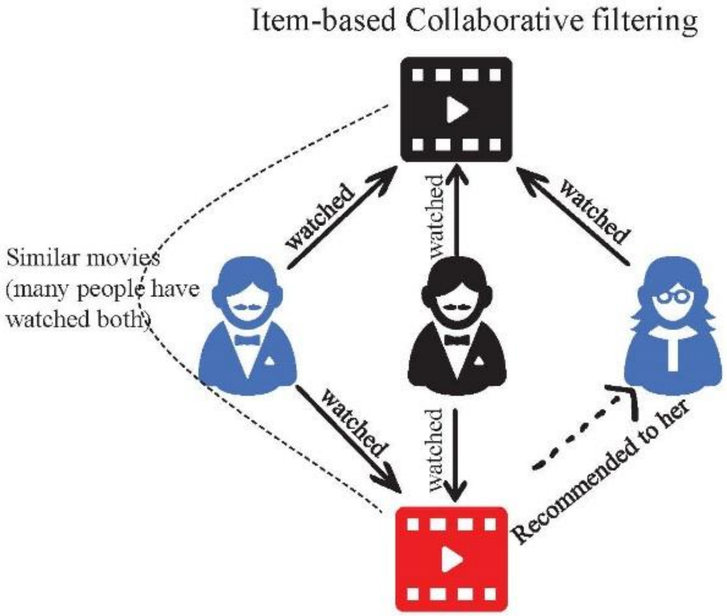
回到我们的推荐系统 CTR 场景

- **核心理念：**协同过滤的本质：发现“品味”相似的人或物
- **人以群分，物以类聚**
- 用户的“品味”通过行为自然分群
- 相似用户往往对同类物品表现出相似偏好
- **行为重合**：
  - **用户A** 和 **用户B** 都购买了多张 CD
  - 由此推断 A、B 口味相近
- **推荐逻辑**：
  - 将 B 购买过、A 尚未购买的 CD 推荐给 A



协同过滤的本质：发现“品味”相似的人或物

- **关键点：历史行为组合越接近的人，购物习惯越一致**
- **不只是同一商品重合**
  - 关注「整体行为模式」而非单次点击或购买
- **组合行为才是关键**
    - 不看单次重叠
- **整体行为序列** 决定口味
    - 购物、浏览、加购皆可用
    - 行为距离越近 → 偏好越一致



协同过滤的本质：发现“品味”相似的人或物

极客时间 训练营

- **不仅看单一特征，而是特征组合的联合效应**
- **特征 A：长途目的地**
  - 例：搜索“上海→夏威夷”（距离 >2000 km）
  - 意图模糊：度假 / 蜜月 / 公差 / 探险?
- **特征 B：家庭出行**
  - 例：选择“3人出行”、浏览“亲子酒店”
  - 意图宽泛：周边游 / 家庭聚会 / 长途旅行?
- **A+B 的联合影响（Joint Impact）**
  - 例：同时满足“长途”+“家庭”，可精准画像“年度重磅家庭度假”
  - **价值信号**：高客单价、提前预订、住宿偏好 → 商业价值极高

如何让 LR 具备协同过滤能力?

- **权重完全隔离：各行其道的参数更新**
- **模型形式**：$P(y=1|x)=\sigma(w1\ x1+w2\ x2+b)$
- **优点**：
  - 训练快速、易部署
  - 权重可解释，每个特征贡献清晰
- **局限（关键）**
- **线性叠加假设**
  - 特征被当作相互独立，模型仅学习加法组合
- **协同贡献难以评估**
  - 只能度量**单一维度行为**对点击率的加性影响

用交叉特征显式表达联合影响

- **在特征工程中，手动将多个二值特征组合为一个新特征，以捕捉特定场景下的“1+1>2”**
- **定义原始二值化特征**
  - x1=is_long_distance∈{0,1}
  - x2=is_family_user∈{0,1}
- **构建交叉特征：**x3=x1×x2
  - 仅当“长途”且“家庭”都为 1 时， x3=1
- **更新后的模型**

$$Score=w1x1+w2x2+w3x3+b$$

- 若用户同时满足 A+B，可学习到独立大权重 w3，显式捕捉“长途家庭游”信号

手工交叉的工程代价

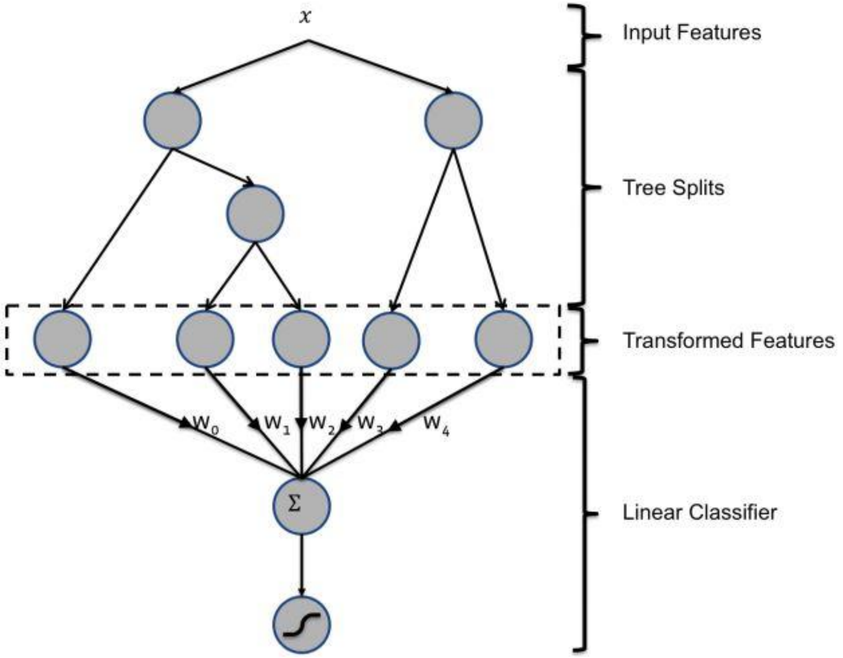
- **数据稀疏，难以学习**
  - 特征交叉组合数百万级，绝大多数从未出现
  - 稀疏维度学习信号微弱，权重无法有效更新
- **运维代价高**
  - 强依赖人工定义与验证，难以维护与上线
  - 业务快速演化，易遗漏关键组合
- **资源开销指数级**
  - 每新增一个特征，组合维度呈指数增长
  - 高维稀疏矩阵 → 存储大、计算重、推理慢

显式权重复用：规则 & GBDT 的低成本套路

- GBDT+LR: 算法模型的强强联合
- 2014年，Facebook 发表了一篇具有里程碑意义的论文，详细介绍了一种巧妙的机器学习模型——GBDT+LR。

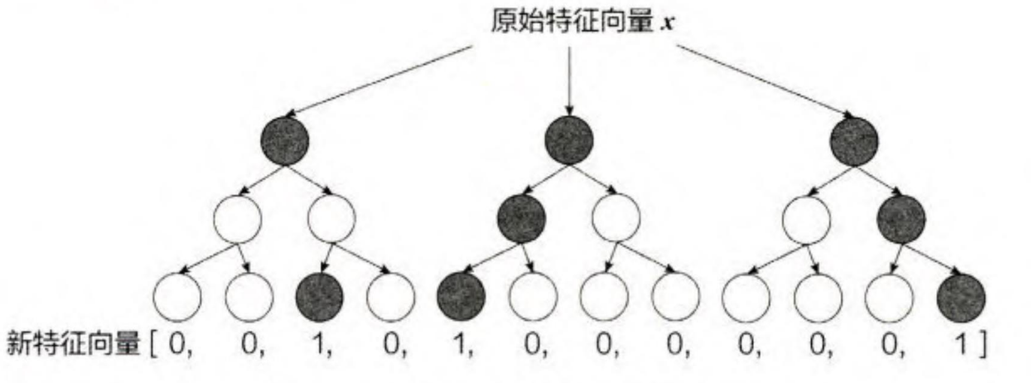
- **提出模型组合**
  - GBDT：自动生成组合特征（“制造逻辑”）
  - LR：对组合特征加权（“分配权重”）

- **为什么强强联合?**
  - GBDT 善于捕捉非线性特征
  - LR 擅长高效计算与可解释性



第一阶段 —— GBDT 构造逻辑组合特征

- **GBDT：让“路径”变成“规则特征”**
  - 每棵树由多个节点构成，每个节点表示一个“条件判断”
  - 样本走一条路径，落入某个叶子节点 → 表示一个“逻辑组合”
  - 每一棵树的叶子节点编号作为新特征
- **举例**：样本 A → Tree1 第3号叶 → Tree2 第1号叶，编码为 [3, 1, 4, ...]



第一阶段 —— GBDT 构造逻辑组合特征

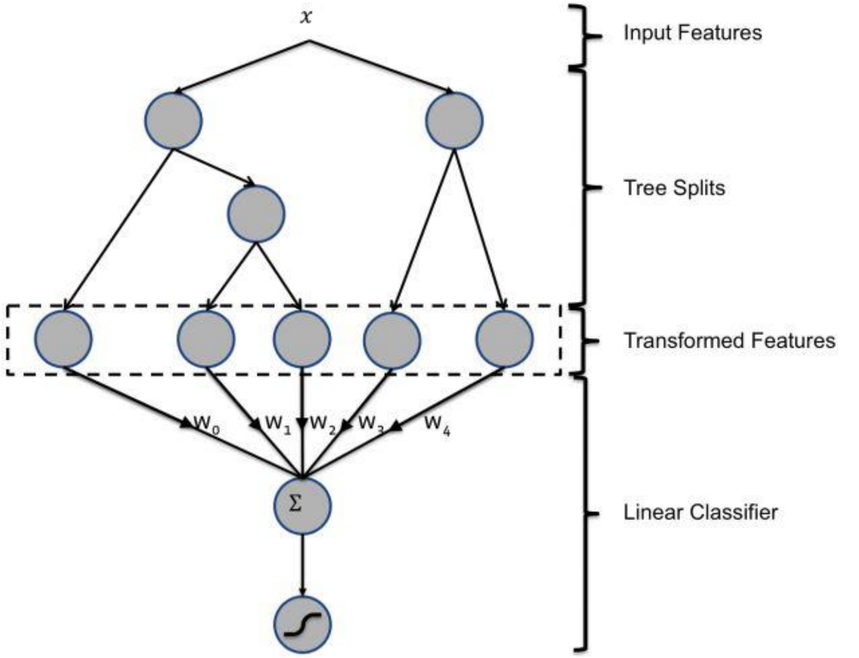
极客时间 训练营

- **One-Hot 编码：把逻辑路径变成“可学习的标签位”**
- 每个叶子节点都变成一个稀疏向量的“开关位”。
- 如果一个样本走到了 Tree1 的第3个叶子 → 就把第3个位设置为 1，其它位为 0。
  - Tree1 第3号叶 → [0, 0, 1, 0] # 命中了“逻辑规则3”
  - Tree2 第5号叶 → [1, 0, 0, 0] # 命中了“逻辑规则1”
  - Tree3 第4号叶 → [0, 0, 0, 1] # 命中了“逻辑规则4”
- **拼接之后：逻辑组合 = 稀疏向量拼接**
- 最终，每个样本都会形成一串“命中的逻辑规则集合”：
- [0, 0, 1, 0, | 1, 0, 0, 0, | 0, 0, 0, 1]
- 这就是它在 GBDT+LR 架构中的新特征表示方式。

第二阶段 —— LR 精准加权高阶特征

- LR：为逻辑特征分配权重，实现预测闭环
- 输入：GBDT生成的 One-Hot 特征向量
- 输出：LR 模型预测结果（如点击率）

- **优势总结**：
- 组合逻辑由数据驱动自动生成
- 权重学习由 LR 高效完成，保持可解释性



显式权重复用：规则 & GBDT 的低成本套路

极客时间 训练营

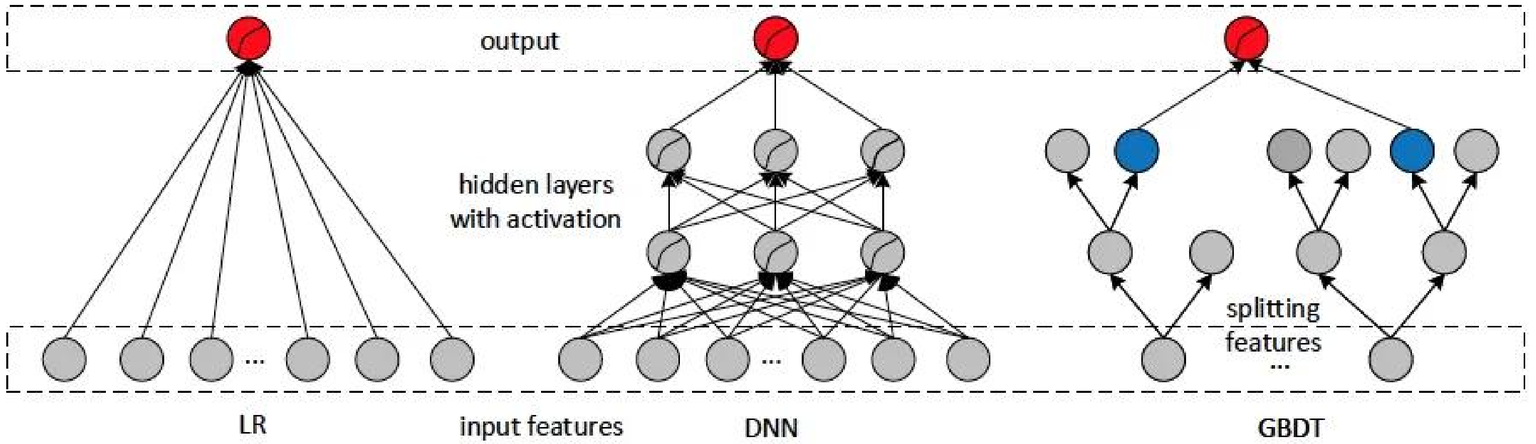
- GBDT+LR 的核心可以看作是一种“显式”的特征工程：GBDT 负责“制造”特征，LR 负责“使用”特征，两个阶段目标不同、过程割裂。
- 这种方式虽然高效，但终究不够优雅和灵活。我们不禁要问：

- **我们能否构建一个统一的模型，让特征的学习、组合与最终的预测在一个框架内完成，让所有参数“心往一处想，劲往一处使”？**
- 这个思考，便引出了以深度学习（如 MLP）为代表的“隐式”特征学习思想——模型不再需要人工设计的两阶段流程，而是通过网络结构在端到端的训练中，自动、协同地学习出最优的特征表示。这正是推荐系统和计算广告领域技术演进的关键一步。

目录

MLP 的“端到端”革命

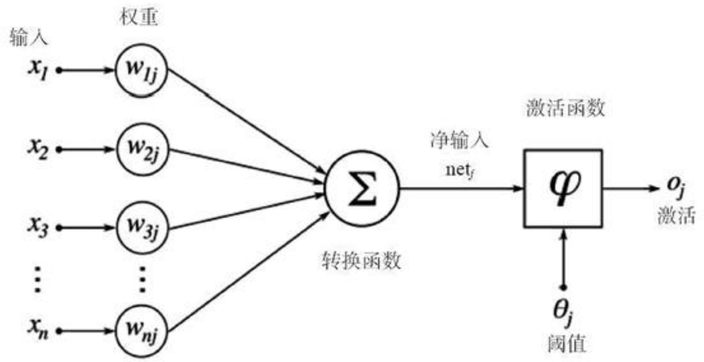
我们能否构建一个“万能”的模型，直接处理所有类型的特征?



对比维度	GBDT + LR 范式	端到端 MLP 范式
建模流程	分阶段：特征工程 + 模型训练	一体化：结构与优化目标统一
特征交叉	半自动：依赖 GBDT 树结构，受限于树的深度和数量	全自动：MLP 网络隐式、高效地学习高阶交叉
优化目标	不一致：GBDT (残差) 与 LR (概率) 目标分离	一致：最小化全局联合损失函数
工程实践	繁琐，依赖经验，维护成本高	简洁，可扩展性强，人力成本低
模型潜力	天花板较低，依赖特征工程的质量	天花板更高，受益于更深、更复杂的网络结构

拥抱现在：MLP 的“端到端”革命

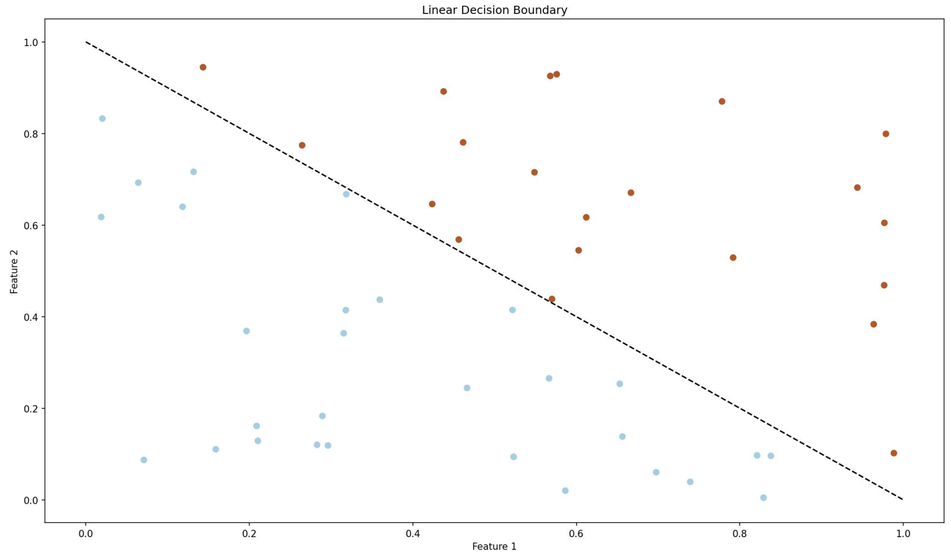
- **最简单的“神经网络”**
- **核心结构：** 一个没有隐藏层的神经网络
- **输入层**接收特征（x_1,x_2）。



- 每个输入通过带**权重 (weights)** 的连接线汇聚到一个**神经元 (neuron)**。
- 神经元内部进行加权求和，并加上一个**偏置 (bias)**。
- 计算结果通过**激活函数 (activation function)** Sigmoid 处理后输出。
- **结论：逻辑回归，本质上就是一个最简单的神经网络。**

逻辑回归的困境——线性边界

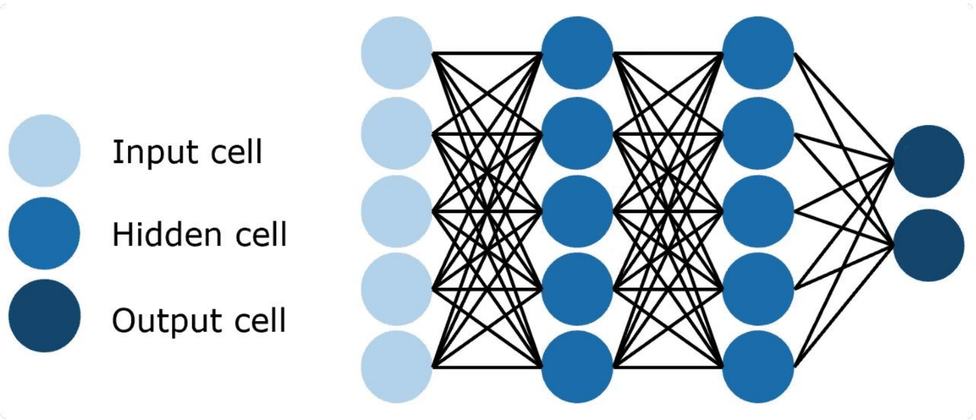
- **核心痛点：** 逻辑回归是**线性模型**，它只能用一条**直线（或高维平面）**来划分数据。



- 不过输入特征往往是**线性不可分**的，“这正是为什么在 GBDT+LR 模式中，我们需要**GBDT先承担‘特征交叉’的繁重工作**，生成类似‘25岁以下的男性’或‘喜欢动作片和科幻片的用户’这样的**组合特征**，再喂给LR。

MLP的解决方案——构建非线性边界

- **核心思想：** 通过加入**隐藏层 (Hidden Layer)**，构建**多层感知机 (Multi-Layer Perceptron, MLP)**。
  - **1. 组合低阶特征：** 隐藏层中的**每一个神经元 (如h_1)**，都会接收所有输入特征。它就像一个“小型LR”，对输入进行自己的一套加权组合，生成一个**新的、更抽象的特征**。
  - **2. 学习非线性关系：** 一个神经元可能学会了识别 (年龄<20 AND 城市=北京)，另一个神经元可能学会了 (性别=女 AND 喜欢购物)。这些都是非线性的组合。
  - **3. 逐层抽象：** 如果有多层隐藏层，后一层可以对前一层输出的“新特征”**再次进行组合**，从而学习到**更高阶、更复杂的模式**。
- **MLP通过引入隐藏层，赋予了模型端到端地、自动地学习特征交叉的能力。隐藏层是深度学习的灵魂。它让模型摆脱了线性的束缚，能够自动、深入地挖掘数据中潜藏的复杂模式，不再需要我们手动设计特征组合。**



MLP是如何学习的?

我们已经设计好了MLP的网络结构，但里面成千上万的“权重”和“偏置”初始都是随机的。模型如何“修炼”成才，找到那一组最佳的权重参数呢？答案是：通过**梯度下降**进行迭代学习。

【Part 1: 温故知新——梯度下降的核心思想】

目标： 找到一组最佳权重 (W, b)，使得模型的损失函数 (Loss Function)值最小。

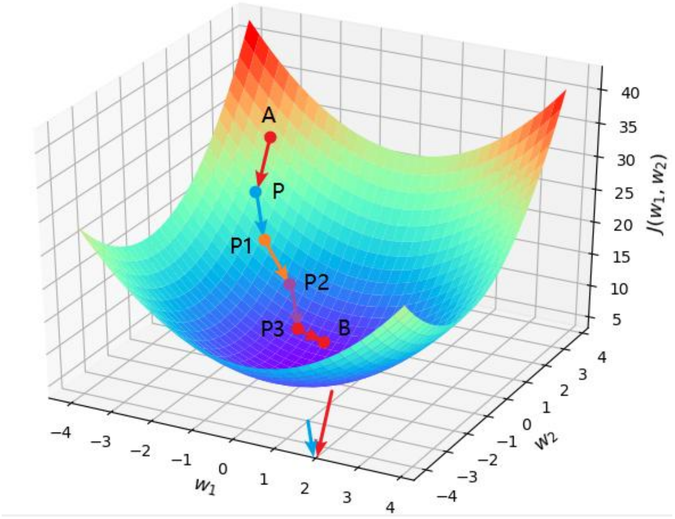
核心思想：

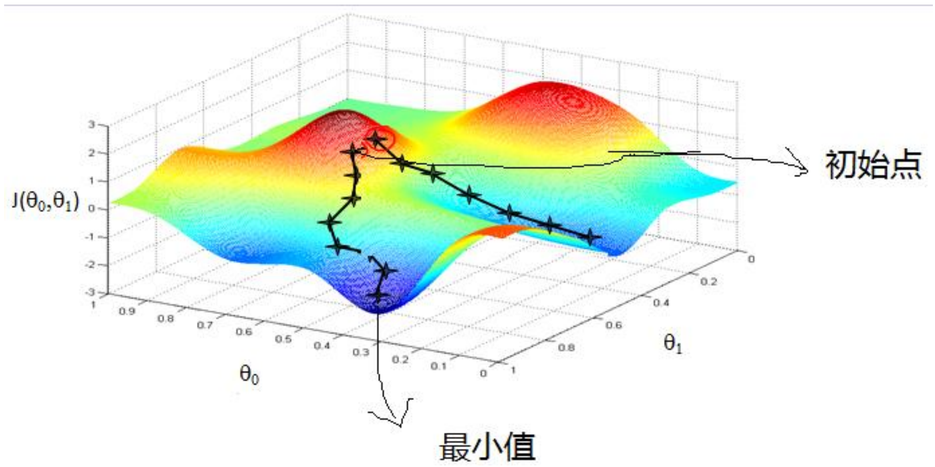
- 把损失函数想象成一座连绵的山脉，我们的目标是走到山脉的最低谷。
- 我们被随机放在山上的一个位置，并且被蒙上了眼睛。
- 我们只能用脚感受**当前位置哪个方向坡度最大（梯度）**，然后朝着**相反的、最陡峭的下坡方向**走一小步。
- 不断重复第3步，最终就有希望走到谷底。

MLP是如何学习的?

【Part 2: 全新挑战——从“一个碗”到“群山”】

逻辑回归 (LR) 的优化世界	多层感知机 (MLP) 的优化世界
损失函数地形：凸函数 (Convex)	损失函数地形：非凸函数 (Non-Convex)
特点： 像一个完美的碗，只有一个最低点（全局最优解）。	特点： 地形复杂，有无数的“小山谷”（局部最优解）和一个真正的“东非大裂谷”（全局最优解）。
结论： “下山”非常简单，无论从哪里出发，保证能到达唯一的谷底。	结论： “下山”充满挑战！我们很可能走到了一个“小山谷”就以为是最低点了，从而陷入局部最优。这是深度学习训练的核心难点。

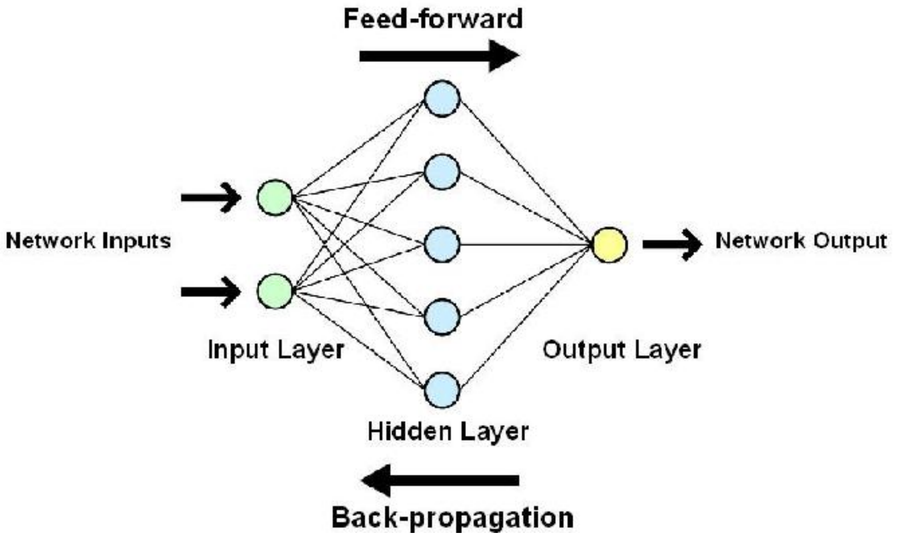




MLP是如何学习的?

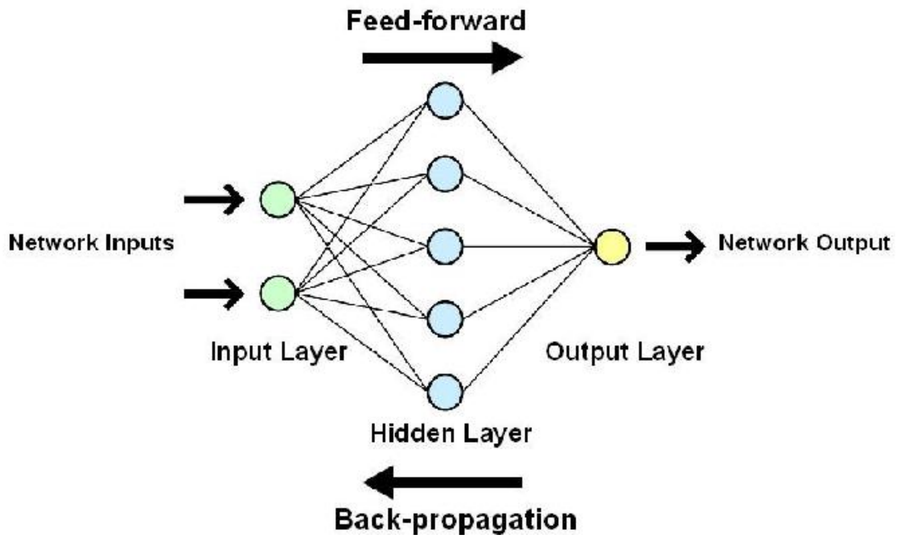
【Part 3: MLP训练的双向核心机制】

- MLP的每一步“下山”（即一次迭代），都包含一进一退两个关键动作：
- **1. 前向传播 (Forward Propagation) → “预测与计算误差”**
- 数据像水流一样，从**输入层**出发，依次流过所有**隐藏层**，最后到达**输出层**，得出一个**预测结果**。
- 将这个“预测结果”与“真实标签”进行比较，通过损失函数计算出模型这次错得有多离谱（即**误差Loss**）。



MLP是如何学习的?

- **2. 反向传播 (Backpropagation) → “分摊责任与更新权重”**
- 这是学习的精髓！计算出的总误差会从**输出层**开始，像回声一样**逐层向后反弹**。
- 每一层的连接权重，会根据自己对最终总误差的“贡献度”**大小，来决定自己应该如何进行梯度调整**（这就是梯度下降的应用）。
- **结论：MLP的学习，就是一个“前向预测 → 反向追责更新”循环往复、不断迭代，最终在崎岖的“群山”中小心翼翼地寻找最低谷的过程。**



MLP的现实挑战：当特征是“海量ID”时

极客时间 训练营

MLP的现实挑战：

在真实世界的模型中，最具价值的特征往往是 **类别型(Categorical)** 的，例如 user_id, item_id 等。这类特征的唯一值数量（我们称之为**词汇表大小**或**基数**）通常非常巨大。

问题： 我们如何将这些高基数类别型特征，有效地输入到MLP中？

【标准但“朴素”的方案：独热编码 (One-Hot Encoding)】

一个直接的想法是使用独热编码。但当我们将此方法应用于拥有百万级甚至更高基数特征的MLP时，会面临两个层面的严峻挑战：一个是**工程实现**上的，另一个是**模型学习效率**上的。

MLP的现实挑战：当特征是“海量ID”时

极客时间 训练营

工程灾难——无法承受的参数量

这是模型在**可行性**层面遇到的硬性障碍。

符号定义：

- 假设一个类别特征（如 item_id）的词汇表大小为 V (例如 10^7)。
- MLP第一层隐藏层的神经元数量为 H (例如 512)。

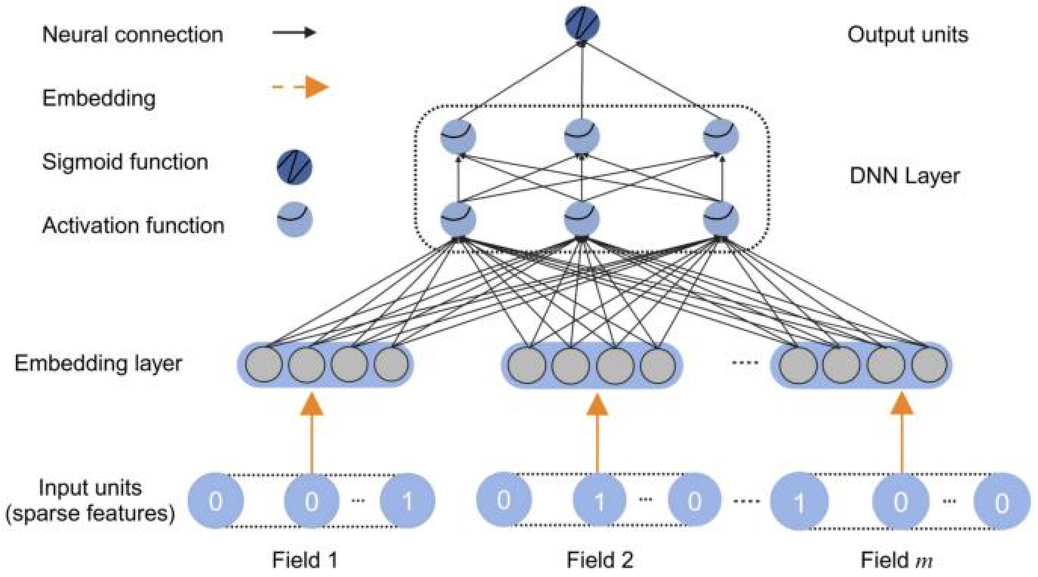
参数计算：

- 输入层（维度为 V）到第一隐藏层（维度为 H）的权重矩阵 W1 的形状为 V×H。
- 这意味着，仅这一层的参数数量就高达：
- $\text{Params}(W_1)=V\times H=10^7\times 512=51.2$亿 **(5B)**

结论： 如此巨大的参数量，无论在**内存存储**还是**计算速度**上都是现代硬件无法承受的。这使得基于One-Hot的MLP方案在处理高基数特征时，工程上基本**不可行**。

MLP的现实挑战：当特征是“海量ID”时

- **结论：** 将高维稀疏的One-Hot特征直接输入MLP，在工程上会遭遇“参数爆炸”。
- **破局之道：** 我们需要一种新的机制，它能够**强制性地让相似ID的参数学习过程“共享”起来**。也就是说，当我们通过训练样本学到了关于“可口可乐”的知识时，我们希望这些知识能被一种高效的结构自动地、低成本地应用到“百事可乐”上。
- **这正是我们将要介绍的 Embedding 层的核心思想——参数共享与知识迁移。**



Embedding 层的参数共享艺术

- 我们可以将Embedding层理解为一个巨大的、可学习的**权重矩阵**，我们称之为 **Embedding 矩阵 W_emb**。
- **矩阵维度：** V x D
  - V：该特征的词汇表大小 (例如，100万个不重复的 item_id)。
  - D：人为设定的**Embedding维度**，一个远小于 V 的超参数 (通常是几十到几百，如 64, 128, 256)。
- **工作机制：** 它将一个ID数字，通过**高效查表 (Lookup)** 的方式，直接映射为矩阵中的某一行。
  - **输入：** 一个具体的ID，例如 item_id = 9527。
  - **操作：** 直接在 Embedding 矩阵 W_emb 中，定位并取出**第 9527 行**。
  - **输出：** 一个 D 维的、低维稠密的向量（我们称之为 **Embedding Vector**）。

方案	进入隐藏层的输入维度	效果
One-Hot 方案	10,000,000	灾难性高维
Embedding 方案	1,024	可行的低维

Embedding语义的来源：监督信号的无声指导

极客时间 训练营

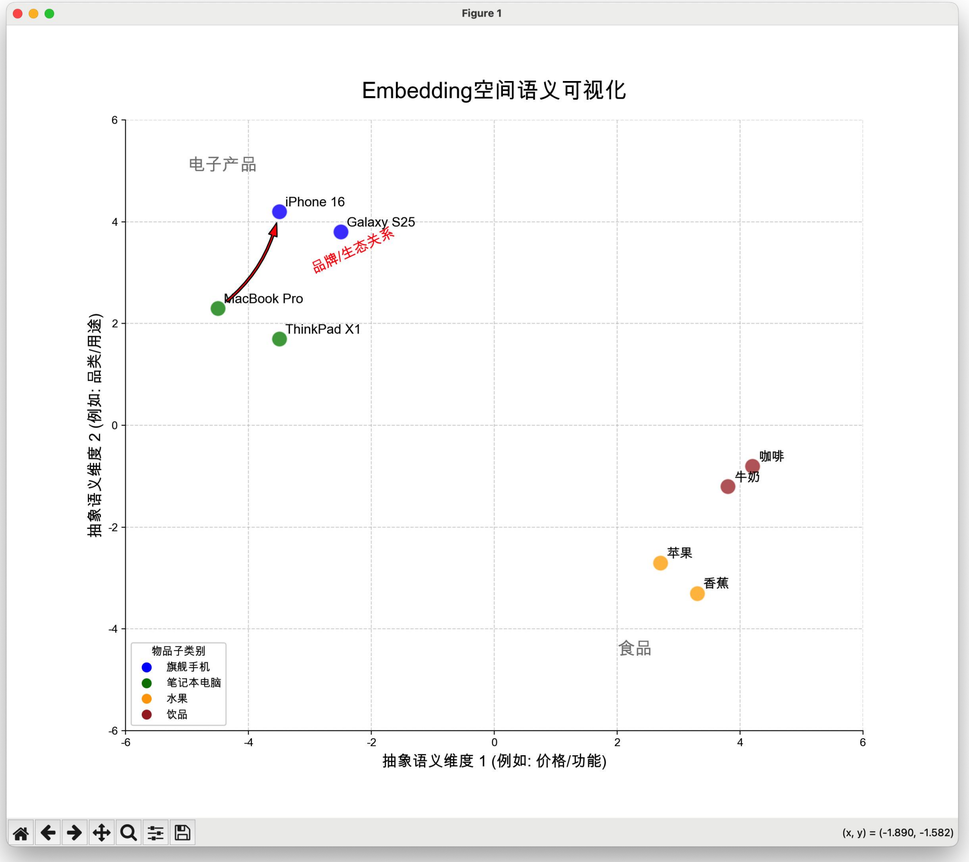
- Embedding层在训练过程中学习到的**稠密向量（Dense Vector）**，并非随机数值的集合。它的最终目标是：**将现实世界中物品的“关系”，映射为向量空间中的“距离”。**

- **什么是空间语义 (Spatial Semantic Meaning)?**
- 简单来说，就是“物以类聚，人以群分”在数学空间的体现。
- **相似的物品，向量相近**: 如果两个item_id代表的物品在功能、类别或用户行为上相似（例如，经常被一起购买、被同类用户查看），那么模型会学习到将它们的向量放置在 embedding 空间中非常接近的位置。
- **向量的“方向”和“距离”具有意义**: 向量之间的算术关系可以揭示物品之间更复杂的类比关系。

Embedding语义的来源：监督信号的无声指导

极客时间训练营

- **一个假想的物品空间：**
- 想象一下，我们将学习到的高维向量（如1024维）投影到一个我们能理解的2D平面上：



- **结论：从“独立ID”到“关系网络”**
- **One-Hot编码**: 每个ID都是孤立的，相互之间没有任何关系可言。模型无法知道 item_id=1001 和 item_id=1002 是否相似。
- **Embedding**: **赋予了每个ID丰富的语义信息**，让模型能够理解物品间的相似、从属、类比等复杂关系，极大地增强了模型的**泛化能力和推理能力**。

Embedding语义的来源：监督信号的无声指导

极客时间 训练营

- 模型是如何知道“可口可乐”和“百事可乐”相似，而和“轮胎”不相似的？它并不具备人类的常识。
- **答案：** Embedding向量所表达的“语义”，并非通用的、先验的知识，而是完全由我们给定的**监督信号**（Supervised Signal）**在训练过程中**“雕刻”出来的。

- **【Part 1: 监督信号——Embedding的唯一“导师”】**
- **监督信号是什么？** 在我们的场景中，监督信号就是**模型的优化目标**，通常由**业务目标**决定。例如：
  - **点击率(CTR)预估任务：** 监督信号就是样本的“**是否点击**”标签 (0或1)。
  - **转化率(CVR)预估任务：** 监督信号就是“**是否购买**”标签 (0或1)。
- **学习的唯一准则：** 整个深度学习网络（**包括Embedding层**在内）的**唯一使命**，就是调整自身所有的参数（包括Embedding矩阵），去**最小化最终的预测损失 (Loss)**。

Embedding语义的来源：监督信号的无声指导

极客时间 训练营

- **【Part 2: 一个CTR预估场景的例子】**
- 让我们看看“是否点击”这个监督信号，是如何塑造Embedding空间的。
- **数据中的行为模式：** 模型在海量的训练数据中观察到以下模式：
  - 看过“可口可乐”广告的用户，如果再给他们看“百事可乐”的广告，**点击率很高**。
  - 看过“可口可乐”广告的用户，如果再给他们看“米其林轮胎”的广告，**点击率极低**。
- **反向传播的“雕刻”过程：**
- **对于模式1：** 为了让模型能准确预测出高点击率，梯度下降算法会发出一个明确的指令：**“拉近‘可口可乐’和‘百事可乐’的Embedding向量！”** 因为这样做，模型就能更容易地将从“可口可乐”学到的用户偏好，泛化到“百事可乐”上，从而做出正确的预测。
- **对于模式2：** 为了让模型能准确预测出低点击率，算法会发出另一个指令：**“推远‘可口可乐’和‘米其林轮胎’的Embedding向量！”** 因为向量距离越远，模型就越容易区分它们，避免做出错误的泛化。

MLP如何实现特征交叉？藏在全连接中的“隐式”组合

- 与 GBDT 那种显式的“分裂+路径”机制不同，MLP 的特征交叉能力是**隐式发生在全连接结构中的**。但它的实现不是一蹴而就的，而是一个**分层嵌套、逐步累积的过程**。
- **Part 1：第一隐藏层 —— 非线性组合，而非真正交叉**
- 我们先看第一隐藏层中的一个神经元 h1，其计算公式为：

$$h_1 = \sigma(w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + b_1)$$

- 其中 x1,x2,x3,…是原始输入特征（或Embedding中的维度），w_i 是神经元的权重，σ(·) 是激活函数（如 ReLU）。
- 虽然多个特征被加权组合在一起，但它们之间**并没有发生真正意义上的“交叉”**，因为每个神经元的计算仍是对输入维度的“线性投影+非线性变换”。所有特征共享同一个权重空间，**但权重之间是彼此独立的**，并未显式建模特征之间的交互逻辑。

MLP如何实现特征交叉？藏在全连接中的“隐式”组合

- **Part 2：多层结构 —— 高阶交叉的指数级生长**
- 从第二层开始，MLP 才真正具备了构建复杂交叉模式的能力。
- 我们来看结构上发生了什么：
- 第二层的每一个神经元，输入的不是原始特征，而是**第一层神经元的输出**，即已经做过“特征加权 + 非线性变换”的组合信号。
- 于是，第二层的计算公式如下：

$$H_2 = \sigma(W_2 \cdot H_1 + b_2) = \sigma(W_2 \cdot \sigma(W_1 \cdot X + b_1) + b_2)$$

- 这种**函数嵌套式的结构**，使得每一层都可以看作是在建构**上一层组合的组合**。原始特征 → 低阶组合 → 高阶组合 → 超高阶交叉，形成一种逐层升级的“表达树”。

MLP如何实现特征交叉？藏在全连接中的“隐式”组合

极客时间 训练营

- 我们可以理解为：

网络层级	学习内容	对应交叉阶数
第1层	原始特征的非线性组合	一阶（线性+非线性）
第2层	第一层组合的再组合	二阶及以上
第L层	L-1层输出的交叉模式再组合	高阶交叉

- 因此，**真正的特征交叉能力，是从第二层才逐渐具备的**，并随着深度的增加而增强。每一个神经元都像一个“模式侦测器”，可以在训练中自动学习哪些特征组合是重要的，从而形成一组**柔性、非线性、可学习的交叉逻辑集合**。

深度结构的“暴力”表达力

极客时间 训练营

- MLP通过逐层组合来构建高阶交叉。这种“深度”（Depth）结构，正是其强大模型能力的核心源泉。它赋予了网络一种近乎无限的、暴力的函数拟合能力。

- **【Part 1: 能力的理论基石 —— 通用逼近定理】**
- **理论核心 (Universal Approximation Theorem):** 一个包含足够多神经元的前馈神经网络（即使只有一个隐藏层），可以以任意精度逼近任意一个连续函数。
- **通俗解读：** 理论上，只要你的网络足够“宽”或足够“深”，它就能模拟出任何你想要的复杂输入–输出关系。因此，**MLP本质上是一个“万能函数逼令器”**。

- **“深度”的独特优势：** 虽然一个很“宽”的浅层网络在理论上可行，但实践证明，“深”层网络在学习具有**层次化结构**的知识时，通常**更有效率、更强大**。因为“组合上一次的组合”是一种更自然的知识抽象方式。

深度结构的“暴力”表达力

极客时间 训练营

- **【Part 2: 表达力为何呈“指数级”增长？】**
- 网络的强大表达能力，源于其组合的层次性，这会形成一棵巨大的、隐式的“组合树”：
- **第 1 层：** 学习原始特征的交叉（例如：用户特征 × 物品特征）。
- **第 2 层：** 将第一层的组合结果进行再组合（例如：(用户×物品) × 上下文特征）。
- **第 L 层：** 将L–1层输出的超高阶组合再次进行组合...

- **结论：** 网络的深度每增加一层，其能够表达的潜在特征交叉模式的数量，就近似于**指数级增长**。这就是“深度”的威力所在，也是“暴力”一词的由来。

深度结构的“暴力”表达力

极客时间 训练营

- 既然MLP理论上如此强大，为什么它不是解决所有问题的“银弹”？因为它的强大能力，必须由三件现实中的“资源”来支撑，这也是它在实践中的能力边界：
  - **参数规模 (Model Size):** 网络必须足够大（足够深/宽）才能拥有足够的“容量”去拟合复杂函数。如果参数太少，表达能力会严重不足。
  - **训练数据量 (Data Volume):** 强大的模型需要海量的数据来“喂养”。如果数据不足，模型会轻易地“记住”训练样本（**过拟合**），而不是学习到通用的规律（**泛化**）。
  - **算力资源 (Computational Power):** 巨大的参数规模和海量数据的结合，意味着高昂的训练成本，需要强大的GPU/TPU集群和漫长的训练时间。

- **总结：多层MLP就像一座“组合特征工厂”，它用暴力而通用的方式，自动学习所有你没想到的交叉逻辑。但要驱动这座工厂，你必须付出数据、参数与算力作为“燃料”。**

MLP对低阶交叉存在“学习盲区”?

- 我们已经盛赞了MLP“暴力”学习所有交叉模式的能力。但这种“一视同仁”的暴力，有时也可能成为一种“盲目”。它是否会因为追求高阶、复杂的组合，而忽略了那些简单、但至关重要的低阶交叉呢?

- **【Part 1: “无偏好”搜索的潜在代价】**
- **MLP的特点：** MLP在学习之初，对所有可能的特征交叉**没有先验的偏好**。它像一个蒙眼的探险家，试图在广袤的土地上摸索出所有的宝藏，无论大小。
- **潜在的浪费：** 这种无偏好的探索，可能导致它将大量的模型容量和计算资源，浪费在探索一些无意义的高阶组合（例如 用户注册城市 × 商品生产批次 × 当前小时）上。 而对业务上真正重要的核心交叉（例如 用户ID × 物品ID），它的“重视度”可能反而不够。

MLP对低阶交叉存在“学习盲区”？

- **【Part 2: 架构的“短板”：缺乏对低阶交叉的“归纳偏置”】**
- **什么是归纳偏置 (Inductive Bias)?** 它是指模型架构中内置的一种“倾向性”或“假设”，使其更容易、更高效地学习到特定类型的模式。
  - **例如：** CNN的卷积核设计，就是一种强大的、针对图像**局部空间关系**的归纳偏置。
- **MLP的缺失：** 标准MLP的全连接结构，**并未内置任何结构上的偏好**来鼓励模型优先学习二阶或三阶这种简单、有效的交叉。所有阶数的交叉都被“一视同仁”，完全依赖反向传播从海量数据中自行发现其重要性。

MLP对低阶交叉存在“学习盲区”？

极客时间 训练营

【Part 3: 一个经典的例子：user_id × item_id 的学习困境】

业务上的重要性： 在推荐系统中，“哪个用户喜欢哪个物品”这种**二阶交叉**，是协同过滤思想的核心，其重要性不言而喻。

MLP的学习方式：

- **对于热门组合 (Dense Data)：** 如果用户A与商品X的历史交互数据**非常丰富**，MLP可以通过“暴力”学习，最终在无数次迭代后捕捉到这个强关联。
- **对于长尾组合 (Sparse Data)：** 如果用户B与商品Y的交互数据**非常稀疏**，MLP在探索海量的参数空间时，很可能**没有足够的数据信号**来让它“注意”到这个微弱但可能很重要的联系。这个关键信号，可能被淹没在无数无意义的组合噪声中。

总结：MLP强大的“通用性”是一把双刃剑。它虽然理论上能学习一切，但也因为它缺乏特定的结构性引导，可能导致它在关键的、显而易见的低阶交叉学习上“投入不足”或“后知后觉”。

这启发了新的模型设计思路：我们能否将MLP强大的**泛化能力（长尾）**，与一种能高效、**显式学习（热门）**关键低阶交叉的结构相结合呢?

Wide & Deep：记忆与泛化的协同建模

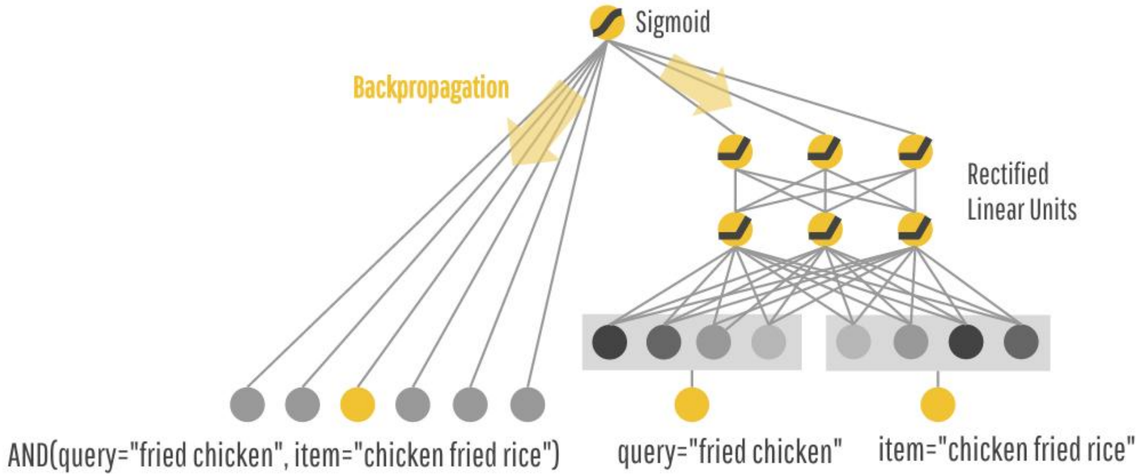
极客时间 训练营

- 一个优秀的推荐系统，需要同时处理两种看似矛盾的需求：
- **记忆能力 (Memorization):** 必须能准确“记住”并利用那些在历史数据中频繁共现的、直接的特征组合。
- **泛化能力 (Generalization):** 必须能探索从未或很少出现过的、更稀疏的特征组合，发现潜在的关联。
- MLP擅长后者，但对前者的学习不够直接高效。为此，Google在2016年提出了Wide & Deep模型，其核心思想就是**让两种能力协同工作**。

Wide & Deep：记忆与泛化的协同建模

【Part 1: 模型的核心结构 —— 两路并进，顶层汇合】

- 模型由两条独立的路径组成：左侧的**Wide部分**和右侧的**Deep部分**，它们的输出在顶层被相加后，共同做出最终预测。



Wide & Deep：记忆与泛化的协同建模

极客时间 训练营

【Part 2: 两大组件的协同思想】

- **Wide 部分 (浅层线性模型)，职责：** 负责**记忆**。
  - **工作方式：** 它直接处理高维稀疏特征（如One-Hot编码的ID及基于**先验知识**进行**人工设计**的交叉特征），通过一个简单的线性模型，快速、直接地学习那些显而易见的、重要的低阶规则。
  - **例如：** 它可以非常高效地“记住”user_id=张三 与 item_id=手机A 这个组合有极强的关联。
- **Deep 部分 (深度神经网络)，职责：** 负责**泛化**。
  - **工作方式：** 它利用我们前面详细讨论的Embedding和MLP结构，将高维稀疏特征转化为低维稠密向量，然后**探索挖掘**这些特征之间潜在的、从未见过的高阶交互关系。
  - **例如：** 它可以发现“年轻数码爱好者”和“新发布的旗舰手机”之间存在潜在的、更抽象的兴趣关联。
- **工程现实驱动的设计：** Google提出此模型时，其线上系统已广泛使用大规模逻辑回归（即Wide部分）。Wide & Deep架构提供了一条**平滑的升级路径**：在保留原有成熟系统的基础上，无缝地“嫁接”上一个提供泛化能力的Deep模块，实现了模型能力的快速迭代。

Wide & Deep 的职责分工

极客时间 训练营

- Wide & Deep模型的精髓，在于其内部清晰的“职责分工”。Wide部分和Deep部分各司其职，像一个经验丰富的老专家和一个充满创造力的新员工，共同组成一个强大的决策团队。

- **【Part 1: 两大组件的详细职责】**

	Wide 部分	Deep 部分
核心职责	记忆 (Memorization)	泛化 (Generalization)
擅长处理	显式的、高频出现的、低阶的已知规则。	隐式的、稀疏的、高阶的未知模式。
输入特征	原始One-Hot特征、人工设计的交叉特征。	类别特征的Embedding向量、连续型数值特征。
工作方式	简单的线性模型，直接、高效。	复杂的多层感知机（MLP），逐层抽象。
优点	速度快，结果稳定，权重可解释。	自动学习交叉，表达能力强，能覆盖长尾。
缺点	依赖人工特征工程，无法发现新模式。	黑箱不可解释，计算开销大。

Wide & Deep 的职责分工

极客时间 训练营

【Part 2: 工业场景下的具体示例】

- **Wide部分如何工作?**
  - **职责：** 显式建模已知规则，擅长“记忆”。
  - **工业示例：** 对于一个在历史数据中被反复验证过的强关联组合，例如 user_id=张三 × item_id=手机A，Wide模块会直接为这个组合学习到一个较高的权重。当这个组合再次出现时，模型能迅速、准确地给出高分预测，如同凭借“肌肉记忆”做出反应。
- **Deep部分如何工作?**
  - **职责：** 自动发现潜在组合，擅长“泛化”。
  - **工业示例：** Deep模块接收代表用户画像（如年龄、性别、兴趣标签）的Embedding向量和代表商品属性的Embedding向量。这些向量在MLP网络中进行复杂的非线性交互，从而可能学习到一种更抽象的模式，比如“年轻男性群体”对“新款游戏笔记本电脑”普遍存在潜在的兴趣偏好，即使某个具体用户和具体电脑的组合从未在数据中出现过。

Wide & Deep 的职责分工

极客时间 训练营

- 为什么需要两者协作?

- 推荐系统面临着一个永恒的“两难”:
- 对于**热门、高频**的模式，我们需要模型能**显式地记住**，保证推荐的准确性和稳定性。
- 对于**长尾、稀疏**的组合，我们需要模型能**自动地泛化**，保证推荐的新颖性和覆盖度。
- **结论：Wide负责记住你见过的，Deep负责猜测你没见过的。二者协同，让推荐系统既稳重又聪明。**

Wide & Deep 的局限：优雅架构下的“历史包袱”

极客时间 训练营

- Wide & Deep模型虽然设计巧妙，但它的成功在很大程度上是其诞生背景（Google已有的大规模线性模型系统）的产物。这也使得它天生带有一些“历史包袱”，使其并非一个完美的“端到端”解决方案。
- **【Part 1: 局限一 · Wide部分依然严重依赖“人工特征交叉”】**
- **核心问题：** 模型中负责“记忆”的Wide部分，其强大的能力依然建立在**手动设计的交叉特征**之上。我们并没有完全摆脱特征工程的桎梏。
- **代价：**
  - **高昂的人力成本：** 特征工程师需要凭业务经验去挖掘、筛选、验证有效的特征组合（例如 gender=F × category=母婴，age_18_24 × time_slot=夜间 等）。
  - **泛化性受限：** 人工设计的特征组合很难覆盖所有情况，尤其是在面对新业务、新场景时，需要重新设计。
  - **维护困难：** 每当需要加入或修改一个交叉特征，往往意味着代码逻辑的重写、数据重跑和重新部署上线，流程繁琐。

Wide & Deep 的局限：优雅架构下的“历史包袱”

极客时间 训练营

- **【Part 2: 局限二 · 输入结构分离，Embedding无法共享】**
- **核心问题：** Wide部分和Deep部分拥有**两套独立的输入通道和特征表示**。
- **具体表现：**
  - **Wide侧：** 接收高维稀疏的**One-Hot表示**。
  - **Deep侧：** 接收低维稠密的**Embedding表示**。
- **后果：** 同一个类别特征（例如 item_id）的信息，在模型的两个部分之间**没有实现参数共享**。这造成了学习上的割裂和参数上的冗余，也增加了模型的复杂度。

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

- 我们分析了Wide & Deep的“历史包袱”：其Wide侧依赖人工，且双路输入架构不够优雅。

- 这些痛点，正是驱动下一代明星模型——**DeepFM**——诞生的核心动因。

目录

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

- **【Part 1: 理论动因 —— 让“显式交叉”也实现自动化】**
- **回顾Wide & Deep的痛点：** Wide部分的强大“记忆”能力，建立在繁琐且容易疏漏的**人工设计交叉特征**之上。

- **核心提问 (The Core Question):**
- 我们能否找到一个组件，来**替代**掉需要人工设计的Wide部分，让模型能够**自动地、高效地**学习所有重要的低阶（尤其是二阶）特征交叉？

- **目标：** **摆脱对人工特征工程的最后依赖**，让低阶交叉的学习也成为模型内建的自动化能力。

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

- **【Part 2: 工程动因 —— 统一Embedding，降低维护成本】**
- **回顾Wide & Deep的痛点：** Wide与Deep两套独立的特征处理路径，导致了参数冗余、特征对齐困难、部署维护复杂等一系列工程问题。

- **核心提问 (The Core Question):**
- 我们能否让模型的不同部分**共享同一套Embedding**？实现真正的特征表示层面的统一，从而简化整个模型的架构和维护流程？
- **目标： 构建一个结构一体化、特征输入统一**的优雅模型。

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

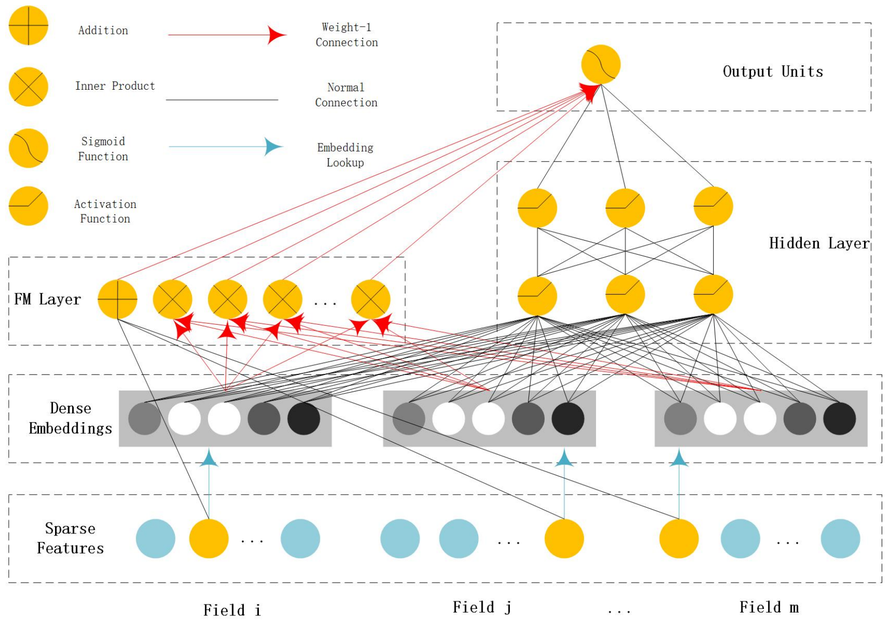
- 总的来说，在DeepFM之前，推荐系统模型普遍面临两大难题：
- **特征交叉的困境**：如何有效学习特征之间复杂的组合关系？是依赖耗时费力的人工设计（如Wide & Deep的Wide部分），还是完全交给“黑盒”的深度网络（DNN）？

- DeepFM给出了一个优雅的解决方案，巧妙地同时解决了上述两个问题，推动了CTR（点击率预估）模型向着更高效、更一体化的方向发展。

DeepFM 架构解析：共享Embedding的优雅设计

极客时间 训练营

- 现在，让我们通过一张架构图，更直观地看看它的内部是如何运作的。



- 整个DeepFM模型可以清晰地分为三大部分：**共享的Embedding层**、**并行的FM部分**和**并行的Deep部分**。

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

- **核心创新 1：用 FM 组件替代 Wide 组件**
- **告别手动，实现特征交叉的自动化学习**
- **引入“因子分解机” (Factorization Machine, FM)**：DeepFM用FM彻底取代了传统Wide & Deep模型中的线性（Wide）部分。
    - 对于一条训练数据（例如：用户A，看了商品X，最终点击了），模型会：
    - 取出“用户A”的隐向量 V_A 和“商品X”的隐向量 V_X。
- **用“内积”自动计算所有交叉关系**： 当需要判断任意两个特征（比如“用户A”和“商品X”）的组合强度时，FM会：
    - 取出它们各自的隐向量 VA 和 VX。
    - **进行内积运算 <V_A, V_X>**，得到一个数值。
    - 这个数值，就**自动成为了**这对特征交叉的权重。

FM的输出值，是“**所有特征的独立影响**”与“**所有特征两两组合的内积影响**”联合起来的总和。

DeepFM 的动因与目标：迈向真正的“端到端”

极客时间 训练营

- **核心创新 2：共享 Embedding 层**
- **连接低阶与高阶，实现信息利用的最大化**
- **精妙的“共享”设计**：这是DeepFM架构的点睛之笔。FM部分与深度网络（Deep）部分共享完全相同的输入特征Embedding层。
- **“一箭双雕”**：原始的、高维稀疏的特征（如用户ID、商品ID）首先被转换成低维稠密的Embedding向量。这个向量将同时被送往两个地方：
  - **送入FM部分**：用于学习上文提到的“二阶特征交叉”。
  - **送入Deep部分（MLP）**：用于通过多层神经网络学习“高阶特征交叉”（例如三个或更多特征的复杂组合关系）。
- **价值**：共享Embedding机制极大地提升了模型的学习效率。模型不再需要为Wide和Deep部分分别学习两套独立的特征表示，而是**用一套统一的、信息密度更高的Embedding**，同时服务于低阶和高阶特征的学习，使得端到端（End-to-End）训练更加高效和自洽。

DeepFM 架构解析：共享Embedding的优雅设计

极客时间 训练营

- **核心创新 3：Deep 组件 (Deep Component)**
- **深入挖掘隐藏的“高阶交叉”关系**
- **它做什么?** ： Deep部分是一个标准的前馈神经网络（MLP），负责**探索三阶、四阶甚至更高阶的**、更抽象、更复杂的特征交叉关系。
- **如何工作?** ：
  - 它同样接收来自Embedding层的所有特征向量。
  - 将这些向量拼接 (Concatenate) 成一个更长的向量。
  - 将这个长向量喂入多层神经网络中。在网络中，神经元之间的交互和非线性激活函数（如ReLU）会帮助模型自动学习到非常复杂和微妙的高阶模式。例如，它可能会学到“（20–25岁的用户）在（一线城市）对（某个特定品牌）的（电子产品）”这种多特征组合下的偏好模式。

DeepFM 架构解析：共享Embedding的优雅设计

极客时间 训练营

- **最终的合力：输出预测结果**

- FM部分和Deep部分的输出是并行计算的，它们各自捕捉了不同维度的特征信息：
- **FM输出**：代表了所有“**显式**”的、可解释性较强的**二阶交叉**关系强度。
- **Deep输出**：代表了所有“**隐式**”的、抽象的**高阶交叉**关系强度。

- 最后，这两个部分的输出结果会被连接在一起，通过一个Sigmoid激活函数，共同给出一个最终的点击率（CTR）预测值。
- 这种结构，使得DeepFM兼具了记忆（Memorization）**和**泛化（Generalization）的能力。

DeepFM的优势总结：为何它成为经典范式?

- DeepFM通过其巧妙的架构，几乎完美地解决了Wide & Deep的核心痛点，使其迅速成为工业界和学术界在CTR预估领域的经典基线模型。其优势主要体现在以下几个方面。

- **【Part 1: 优势一 · 真正的端到端学习，告别特征工程】**
- **关键突破：** FM组件的引入，**彻底替代了依赖人工经验的特征交叉工程**。
- **实现方式：** 模型能自动学习所有特征之间的二阶交叉，无需再手动指定 user_gender × item_category 这类组合。
- **核心价值：** 极大地降低了人力成本，提升了模型迭代的速度和灵活性，让模型能够发现人凭经验难以发现的有效组合。

DeepFM的优势总结：为何它成为经典范式?

极客时间 训练营

- **【Part 2: 优势二 · 高效的架构与参数共享】**
- **共享Embedding (Shared Embedding)：** 这是DeepFM设计的精髓。FM和Deep部分共享同一套Embedding参数，避免了Wide & Deep中的参数冗余和双重计算，统一了**特征处理动作**。
- **训练更充分 (Richer Representation Learning)：** 来自**低阶（FM）**和**高阶（Deep）**交叉的梯度信号会**同时优化**这套共享的**Embedding**，使得特征表示的学习更全面、更高效。

DeepFM的优势总结：为何它成为经典范式?

极客时间 训练营

- **【Part 3: 优势三 · 工程实现的简洁与优雅】**
- **统一的输入：** 模型不再需要为Wide和Deep两部分维护两套不同的**特征处理管道**，所有原始特征都通过统一的Embedding层输入。
- **简化的部署与维护：** 架构的一体化大大降低了线上部署、维护和迭代的复杂度，减少了因特征逻辑不一致而出错的风险。
- **结论：** DeepFM在模型效果和工程效率之间取得了出色的平衡，使其非常适合在工业界大规模落地。
- **总结：DeepFM通过“用FM自动学习低阶交叉”并“让两部分共享Embedding”，构建了一个比Wide & Deep更自动化、更高效、更优雅的端到端模型。它不仅提升了模型效果，更重要的是，它代表了推荐模型架构向着“一体化”和“自动化”方向演进的一大步。**

目录

深度学习的本质：面向场景的“表示”设计

极客时间训练营

【Part 1: 我们的旅程回顾：一条围绕“表示”的进化之路】

我们已经走过了从隔离到共享，从显式到隐式的完整旅程：

GBDT + LR: 依赖**人工设计**的、离散的组合特征表示。

MLP: 探索了**隐式的、高阶的**特征表示。

Embedding: 创造了ID类特征的**低维、稠密**的语义表示。

Wide&Deep/DeepFM: 实现了**低阶显式与高阶隐式**的混合表示。

现在，让我们跳出具体的模型，思考贯穿始终的核心线索：**我们究竟在不断优化什么？**

答案是：**特征的表示 (Feature Representation)**。

深度学习的本质：面向场景的“表示”设计

极客时间 训练营

【Part 2: 核心归纳：深度学习 = 针对场景需求的“表示设计”】

所有复杂的网络架构，其本质都是为了学习一个更优的、可自动从原始数据生成特征向量的“表示函数”。

我们之所以选择或设计不同的模型，完全取决于业务场景对特征表示的**特定需求**：

当业务强依赖于已知的、关键的二阶交叉时...

- 你需要：一种能精准、显式表达 A×B 关系的表示。
- **解决方案：** FM 的**内积表示** ⟨vA,vB⟩ 就是为此而生。

当业务需要探索未知的、复杂的高阶组合模式时...

- 你需要：一种能灵活、强大地组合所有特征的表示。
- **解决方案：** MLP 的**深度非线性表示** (σ(W·X+b)) 提供了这种可能性。

深度学习的本质：面向场景的“表示”设计

当业务特征是序列化的（如用户历史点击行为）...

- 你需要：一种能捕捉时序依赖和兴趣动态演化的表示。
- **解决方案：** Transformer / RNN 的**注意力或循环表示**就派上了用场。

当业务特征是图结构的（如社交网络、知识图谱）...

- 你需要：一种能聚合邻居节点信息的结构化表示。
- **解决方案：** GNN (图神经网络) 的**邻域聚合表示**成为了不二之选。

结论：我们的核心任务不再是“做特征工程”，而是“做表示设计”。我们的工作是从工具箱（MLP, FM, Transformer...）中选择或创造最适合当前业务需求的“表示函数”（即模型架构），并通过数据驱动的方式让它自动学习。

联结主义的实现：“学习”的工程化蓝图

极客时间 训练营

联结主义相信知识可以“生长”在权重中。这听起来很神奇，但深度学习的伟大之处在于，它为这个“生长”过程提供了一套具体、可执行的工程化蓝图——**反向传播 (Backpropagation)** 与 **梯度下降 (Gradient Descent)**。

【一个完整的“学习”周期：从犯错到进步】

让我们来看，当一个网络面对一个训练样本时，它内部究竟发生了什么。

1. 初始状态：混沌的连接 网络刚被创建时，其内部数百万的连接权重是**随机**的。此时它什么都不知道，就像一个“空白的大脑”。

2. 前向传播：接受刺激，产生误差 给网络输入一个信号（例如，一张**猫**的图片）。由于连接是混乱的，网络可能会输出一个错误的结果（例如，识别成了“狗”）。**我们将这个“预测结果”与“真实标签”进行比较，计算出**误差 (Error / Loss) 的大小。

3. 反向传播：追溯责任，分配“贡献” 这是学习的魔法所在！**反向传播算法**会像一个侦探一样，从最终的误差出发，逐层向后“追溯”。它会精确地计算出网络中的**每一个连接（权重）**，对这次的最终错误“贡献”了多少责任。这个“责任”，在数学上就是**梯度 (Gradient)**。

联结主义的实现：“学习”的工程化蓝图

4. 梯度下降：实施“重塑”，微调连接 梯度下降算法根据“责任”大小，对每一个连接的强度进行一次微小的、能让总误差减小的调整。其核心更新公式为：

新权重=旧权重一学习率×梯度

5. 循环往复：千万次的“雕刻” 这个 **“预测 → 犯错 → 追责 → 微调”** 的过程，在整个数据集上重复进行亿万次之后，整个网络的连接权重就被逐渐“雕刻”或“重塑”成了一个能够准确完成任务的结构。

“后天的学习”在深度学习的框架下，不再是一个模糊的概念。它就是指在海量数据的驱动下，通过优化算法（如梯度下降），对网络连接权重进行迭代式、可计算的调整的具体过程。

学习的最终成果：构建优质的“数据表示”

经过亿万次重塑后，我们最终得到了什么？深度学习最强大的地方，就在于它通过这个过程，自动学会了为输入信号构建一个极其优质的**数据表示 (Data Representation)**。

【Part 1: 什么是“表示”？】

定义： 将原始的、复杂的、高维的输入信号，通过一个训练有素的深度网络，转换为一个**低维的、稠密的、且蕴含了解决问题所需的关键语义信息的数学向量**。

这个过程，就是将机器难以理解的原始数据，翻译成它能够轻松计算、比较和推理的“通用语言”。

学习的最终成果：构建优质的“数据表示”

极客时间 训练营

【Part 2: 跨模态的统一逻辑：从原始信号到优质表示】

深度学习的优雅之处在于，这套逻辑适用于所有不同类型的数据（模态）。

视觉信号 (图像):

- **输入：** 一张 224x224x3 的原始像素矩阵。
- **学习成果：** 一个训练好的**卷积神经网络 (CNN)**，其内部连接被重塑。浅层连接学会了识别边缘、颜色、纹理；深层连接学会了识别眼睛、鼻子等部件，最终到完整的物体。
- **输出表示：** 一个 1x1000 的向量，其数值分布直接对应了图像的内容是“猫”还是“狗”。

语言信号 (文本):

- **输入：** 一个孤立的单词，例如“bank”。
- **学习成果：** 一个训练好的 **Transformer 模型 (如BERT)**，其连接被重塑，学会了根据上下文动态地理解词义。
- **输出表示：** 一个 1x768 的词嵌入向量。当上下文是“我去银行存钱”时，该向量的表示会更接近“金融机构”；当上下文是“我坐在河的bank上”时，其表示会更接近“岸边”。

学习的最终成果：构建优质的“数据表示”

在我们讨论的推荐场景中：

- **输入：** 一个高维稀疏的 user_id 或 item_id。
- **学习成果：** 我们之前讨论的 **MLP/DNN 模型**，其Embedding层的连接权重被重塑。
- **输出表示：** 一个 1x128 的Embedding向量，其在空间中的位置和方向，表示了该用户或物品在海量交互数据中体现出的**行为模式、品类归属和潜在兴趣**。

无论是图像、文本还是用户行为，深度学习的魔法在于，它能为不同模态的原始数据，自动学习到一个统一的、可计算、可比较的、解决了核心任务的“通用语言”——即优质的、低维稠密的向量表示。

深度学习“大力出奇迹”的根源

从 GBDT+LR 到 DeepFM，从特征工程到表示设计，我们已经完成了一次完整的认知旅程。现在，让我们将最后几页的核心思想串联起来，形成一条贯穿所有现代深度学习模型的“第一性原理”。

这条从**“哲学思想”**到**“学习过程”**再到**“最终能力”**的逻辑链，正是深度学习能够在图像识别、自然语言处理、语音识别、推荐系统等多个看似不相关的领域取得巨大成功的原因。

因为，它们的底层逻辑是完全相通的：

只要有**足够的数据**和**强大的算力**，一个**通用的学习算法（梯度下降）**，就可以通过**重塑网络连接**的方式，从**任何类型**的原始信号中，**自动学习**出解决特定问题所需要的**最佳数据表示**。

我们从一个具体的模型优化问题出发，最终回归到了人工智能最核心的命题。希望这次旅程能帮助大家不仅理解“术”，更能洞察其后的“道”。

推荐阅读

极客时间训练营

- Pattern Recognition and Machine Learning （PRML） 作者：Christopher Bishop

练习任务

极客时间 训练营

- 将测试床项目中的现有 LR CTR 模型替换为 Wide & Deep 模型，实现特征记忆与泛化能力的融合。
- 保持训练评估逻辑一致，确保与原测试框架兼容。
- 项目地址：github.com/tylerelyt/test bed

THANKS