

算法部分

概述

在算法部分,我尝试实现了中不同类型的算法,包括强化学习、协同过滤与随机游走

在这一部分,很多算法实际上有一些框架或库存在,但本着了解原理的思想出发,除了xDeepFM使用了pytorch外,我在其余所有的算法实现中均只使用了numpy(一个矩阵运算库,仅用于简化矩阵乘法的三重循环并使用其简便的切片机制).

Thomson Sampling

概述

在之前介绍的Explore模块部分即使用了汤普森采样算法,其主要思想为:为每个user维护一组bandit向量,然后根据尝试结果不断迭代并达到E&E问题的最佳策略

对于这个问题,其难点在于如何生成满足Beta分布的随机变量.

流程

1. 为每个user初始化一系列bandit向量,即初始beta分布的 α 值与 β 值,在本实验中以不同体裁作为bandit,初始参数为(20,20)
2. 对每个bandit的beta分布进行采样,取其中值最大的那一个进行exploit
3. 根据前端的反馈(CTR)对结果进行更新,如果为正则令 $\alpha+1$,否则另 $\beta+1$
4. 重复第2步,不断进行迭代

关于更新方式

汤普森采样使用了贝叶斯推断的思想,并巧妙地使用了beta分布,这使得参数的更新十分简便.

在贝叶斯推断中,后验概率的计算方式为: $p(x|z) = \frac{p(z|x)p(x)}{\int p(z|x)dx}$

其中 $p(x|z)$ 为后验概率, $p(x)$ 为先验概率, $p(z|x)$ 为似然函数,分母处的积分结果 $p(z)$ 为证据因子,为一个常数.

当采用伯努利分布作为似然函数时,beta分布便成为了其共轭先验,即当 $p(x|z)$ 满足伯努利分布时, $p(x)$ 满足beta分布,则 $p(x|z)$ 也满足beta分布.

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)}$$

↗ likelihood
 ↗ prior
 ↙ evidence

$$P(x|z) = K x^{\alpha} (1-x)^{\beta-1} \cdot \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

$$= K \frac{1}{B(\alpha, \beta)} x^{\alpha+\beta-1} (1-x)^{\beta-1}$$

$\because P(x|z)$ is pdf

$$\therefore \int P(x|z) dx = 1$$

$$\frac{B(\alpha, \beta)}{K B(\alpha+\beta-1, \beta-1)} P(x|z) = \frac{1}{B(\alpha+\beta-1, \beta-1)} x^{\alpha+\beta-1} (1-x)^{\beta-1}$$

右边为 Beta($\alpha+\beta-1, \beta-1$) 的 pdf

$$\therefore \frac{B(\alpha, \beta)}{K B(\alpha+\beta-1, \beta-1)} \int P(x|z) dx = \int \frac{1}{B(\alpha+\beta-1, \beta-1)} x^{\alpha+\beta-1} (1-x)^{\beta-1} dx$$

$$\frac{B(\alpha, \beta)}{K B(\alpha+\beta-1, \beta-1)} \times 1 = 1$$

$$K = \frac{B(\alpha, \beta)}{B(\alpha+\beta-1, \beta-1)} \implies P(x|z) = \frac{1}{B(\alpha+\beta-1, \beta-1)} x^{\alpha+\beta-1} (1-x)^{\beta-1}$$

因此, Beta 分布是伯努利分布的共轭先验,
且后验参数根据实验结果的更新方式为:

$$\begin{cases} \text{成功: } \alpha+1 \\ \text{失败: } \beta+1 \end{cases} \Rightarrow \alpha \leftarrow \alpha+1, \beta \leftarrow \beta+1$$

关于 Thompson Sampling

使用 Beta 分布的合理性
与参数更新方式的说明

黑字深

采样方法

根据一个分布的pdf(probability distribution / density function)生成满足其分布的随机变量有下列常用的方法:

1. 反变换定理:

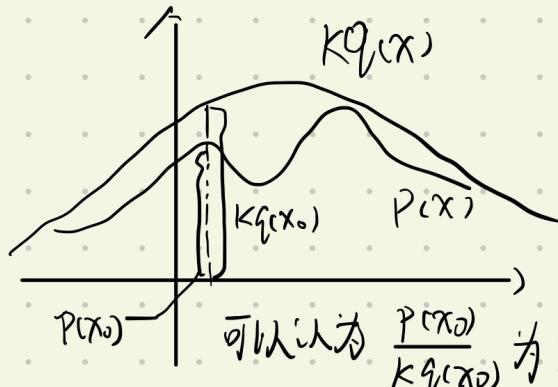
将pdf积分得到CDF,从均匀分布中采样并根据CDF决定拒绝还是接受

反变换方法虽然简单,但是多数情况下积分都很可能没有解析式,因此基本上是一个概念上的方法

2. 接受-拒绝采样:

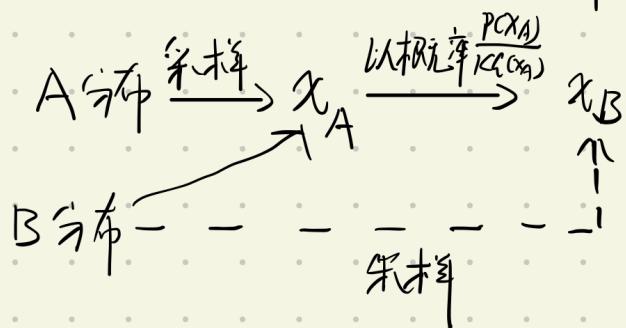
接受-拒绝采样可以根据一个分布A的pdf q去生成满足目标B的pdf p的随机变量,其做法为选择一个k值,使得 $kq(x) \geq p(x)$ 恒成立,然后生成一个满足分布A随机数x,计算 $\frac{p(x)}{kq(x)}$.

得到比值后从U(0,1)中采样得到一个随机数,若比值大于随机数则接受采样x,否则拒绝采样.



可以认为 $\frac{P(x_0)}{Kq(x_0)}$ 为符合分布 A 的随机变量 x_0

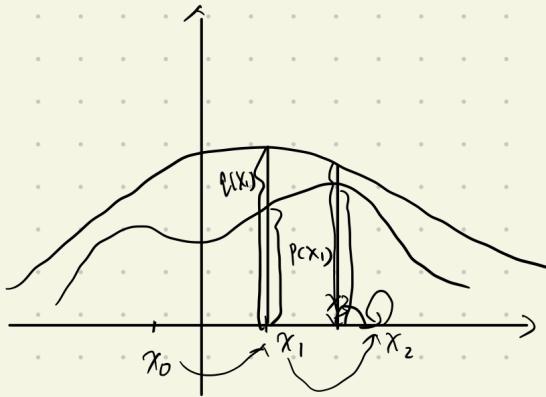
在分布 B 中出现的概率/权重，使用均匀随机数与权重决定是否接受采样



在实验中实现的接受-拒绝采样中,我选择使用高斯分布拟合beta分布,二者曲线较为接近,能够有效减少采样被拒绝的次数.

3. MCMC方法

MCMC方法同用可以使用一个易采样的随机分布来近似目标分布.其认为采样变量的转移过程可以视为一个随机游走过程,在满足细致平稳条件的前提下 x 在游走中其历史取值最终会收敛到目标分布.



在 M-H 算法中, 决定转移与否的概率为 $\min\left(1, \frac{p(x_{n+1})/q(x_{n+1})}{p(x_n)/q(x_n)}\right)$
 也可以将其直观的理解为新采样得到满足简单分布 A 的变量 x_{n+1}
 与目标分布在概率上的接近程度与上一次的比值, 新的采样结果与前一个相比如果越贴近目标分布, 则越倾向于转移,

在实验中实现的Metropolis-Hasting算法同样从高斯分布中采样.

PersonalRank

概述

PersonalRank 算法通过在用户-物品二分图上进行随机游走来寻找相似的用户/物品.

我在实验中实现了基于隐式反馈的无权随机游走的矩阵迭代方法, 并对边权分配方式进行了一些改变, 进而实现了结合显式反馈的随机游走方法.

生成转移矩阵

隐式反馈

在用户返回只有0-1的情况下, 一个用户到达物品的概率通过下面的方式计算: 如果用户对物品没有隐式反馈, 则为0, 否则与其它所有该用户进行过隐式反馈的物品平分转移概率. 物品到用户的概率同理.

显式反馈

在用户进行显式反馈的情况下(0-5分评价), 用户到达物品的概率通过下面的方式计算:

- 首先根据用户评分矩阵, 将所有未进行显式反馈的部分(0分)变为2.5分, 然后将用户评分矩阵中的所有值扩大2倍.
- 在计算转移概率时, 将某用户对所有物品的评分(变换后)使用softmax进行处理, 将分数转化为概率, 作为转移概率.

由softmax的原理可知, 实际上第一步所做的工作为将反馈分数的区间进行放缩而不改变未进行反馈的部分, 从而鼓励对高分游走而抑制低分游走.

如将评分向量[1,2,0,0,4]进行变换后得到[2,4,6,6,8], 因为softmax是平移不变的, 因此该向量在这个概念上而言等价于[-4,-2,0,0,2], 相当于只对显式反馈做变换.

随机游走

随机游走的矩阵表示为 $r_{n+1} = (1 - \alpha)r_0 + \alpha M^T r$,等式右侧第一项代表停止游走返回原点,第二项代表以概率矩阵M中的概率继续游走,两个动作的概率为 $(1 - \alpha)$ 和 α .

经过整理容易得到, $r_{n+1} = (1 - \alpha)(I - \alpha M^T)^{-1} r_n$

在一次矩阵求逆以及矩阵乘法后便可以完成一轮迭代.经过测试,610*9723大小的矩阵经过8轮迭代可使变化小于1e-8.

BiasSVD

概述

BiasSVD是一种基于矩阵分解模型的协同过滤算法,其在基本矩阵分解的基础上考虑到了偏置信息,因此具有更强的泛化能力.

模型参数

- P:用户隐向量矩阵
- Q:商品隐向量矩阵
- B_U :用户偏好偏置向量
- B_I :商品属性偏置向量
- μ :全局偏置
- λ :正则化项

模型推导

BiasSVD 模型基于 SGD 进行参数更新方式的推导 吴学珠

$$\hat{M} = P Q^T + BU + BI + \mu$$

$$\begin{aligned} J_{u,i} &= \frac{1}{2} (\hat{y}_{u,i} - y_{u,i})^2 + \frac{1}{2} \lambda (BU_u^2 + BI_i^2 + \|P_u\|^2 + \|Q_i\|^2 + \mu^2) \\ &= \frac{1}{2} (\hat{y}_{u,i} - y_{u,i})^2 + \frac{1}{2} \lambda (BU_u^2 + BI_i^2 + \|P_u\|^2 + \|Q_i\|^2 + \mu^2) \end{aligned}$$

- $\frac{\partial J_{u,i}}{\partial \mu} = (\hat{y}_{u,i} - y_{u,i}) d\hat{y}_{u,i} + \lambda d\mu / \mu$ 记 $\hat{y}_{u,i} - y_{u,i} \neq g$
 $= g d\mu + \lambda d\mu$
 $= (g + \lambda) d\mu$

$$\frac{\partial dJ_{u,i}}{\partial \mu} = g + \lambda, \quad \mu \leftarrow \mu - \eta(g + \lambda)$$

- $\frac{\partial J_{u,i}}{\partial BU_u} = g d\hat{y}_{u,i} + \lambda BU_u dBU_u$
 $= g BU_u dBV_u + \lambda BV_u dBV_u$
 $= (g BV_u + \lambda BV_u) dBV_u$

$$\frac{\partial dJ_{u,i}}{\partial BV_u} = g BV_u + \lambda BV_u, \quad BV_u \leftarrow BV_u - \eta(g BV_u + \lambda BV_u)$$

同理, $\frac{\partial J}{\partial BI_i} = g BI_i + \lambda BI_i, BI_i \leftarrow BI_i - \eta(g BI_i + \lambda BI_i)$

- $\frac{\partial J_{u,i}}{\partial P_u} = g d\hat{y}_{u,i} + \frac{1}{2} \lambda d(P_u^T P_u)$
 $= g dP_u \cdot Q_i^T + \frac{1}{2} \lambda dP_u^T \cdot P_u + \frac{1}{2} \lambda P_u^T dP_u$
 $= \text{tr}(Q_i^T g dP_u) + \frac{1}{2} \lambda \text{tr}(dP_u^T P_u) + \frac{1}{2} \lambda \text{tr}(P_u^T dP_u)$
 $= \text{tr}(Q_i^T g dP_u) + \frac{1}{2} \lambda \text{tr}(P_u^T dP_u) + \frac{1}{2} \lambda \text{tr}(P_u^T dP_u)$
 $= \text{tr}((Q_i^T g + \lambda P_u^T) dP_u)$

$$\frac{\partial dJ_{u,i}}{\partial P_u} = g Q_i + \lambda P_u, \quad P_u \leftarrow P_u - \eta(g Q_i + \lambda P_u)$$

同理, $\frac{\partial dJ_{u,i}}{\partial Q_i} = g P_u + \lambda Q_i, Q_i \leftarrow Q_i - \eta(g P_u + \lambda Q_i)$

SVD++

概述

SVD++相比BiasSVD,其考虑了隐式反馈的作用,为隐式反馈建模了特征向量.

模型参数

- P:用户隐向量矩阵
- Q:商品隐向量矩阵
- B_U :用户偏好偏置向量
- B_I :商品属性偏置向量
- μ :全局偏置
- λ :正则化项
- Y:商品隐式反馈偏好向量矩阵

模型推导

SVD++

$$\hat{M} = \mu + BV_u + BI_i + (P_u + \frac{\sum_{j \in I_u} y_j}{\sqrt{|I_u|}}) Q_i^T$$

用 P_u 进行补偿
该的商品Id
商品隐式反馈偏好向量

$$\begin{aligned} J_{u,i} &= \frac{1}{2} (\hat{r}_{u,i} - r_{u,i})^2 + \frac{1}{2} \lambda (BV_u^2 + BI_i^2 + \|Q_i\|^2 + \|P_u\|^2 + \sum_{j \in I_u} |y_j|^2) \\ &= \frac{1}{2} (\mu + BV_u + BI_i + (P_u + \frac{\sum_{j \in I_u} y_j}{\sqrt{|I_u|}}) Q_i^T) + \frac{1}{2} \lambda (BV_u^2 + BI_i^2 + \|Q_i\|^2 + \|P_u\|^2 + \sum_{j \in I_u} |y_j|^2) \end{aligned}$$

μ, BV_u, BI_i, P_u 的梯度过程与 BiasSVD 相同：

$$\frac{\partial J_{u,i}}{\partial \mu} = g + \lambda \mu, \quad \mu \leftarrow \mu - \eta(g + \lambda \mu)$$

$$\frac{\partial J_{u,i}}{\partial BV_u} = g BV_u + \lambda BV_u, \quad BV_u \leftarrow BV_u - \eta(g BV_u + \lambda BV_u)$$

$$\frac{\partial J}{\partial BI_i} = g BI_i + \lambda BI_i, \quad BI_i \leftarrow BI_i - \eta(g BI_i + \lambda BI_i)$$

$$\frac{\partial J_{u,i}}{\partial P_u} = g Q_i + \lambda P_u, \quad P_u \leftarrow P_u - \eta(g Q_i + \lambda P_u)$$

$$\begin{aligned} dJ_{u,i}|_{Q_i} &= g d\hat{r}_{u,i} + \lambda Q_i^T dQ_i \\ &= g(P_u + \frac{\sum_{j \in I_u} y_j}{\sqrt{|I_u|}}) dQ_i^T + \lambda Q_i^T dQ_i, \text{ 记 } P_u + \frac{\sum_{j \in I_u} y_j}{\sqrt{|I_u|}} \text{ 为 } PF_u \\ &= \text{tr}(g PF_u dQ_i^T) + \text{tr}(\lambda Q_i^T dQ_i) \\ &= \text{tr}(dQ_i^T g PF_u) + \text{tr}(\lambda Q_i^T dQ_i) \\ &= \text{tr}(PF_u^T g dQ_i + \lambda Q_i^T dQ_i) \\ &= \text{tr}((PF_u^T g + \lambda Q_i^T) dQ_i) \end{aligned}$$

$$\frac{\partial J_{u,i}}{\partial Q_i} = g PF_u + \lambda Q_i, \quad Q_i \leftarrow Q_i - \eta(g PF_u + \lambda Q_i)$$

$$\begin{aligned} dJ_{u,i}|_{y_i} &= g d\hat{r}_{u,i} + \lambda y_i^T dy_i \\ &= g \frac{dy_i}{\sqrt{|I_u|}} \cdot Q_i^T + \lambda y_i^T dy_i \\ &= \text{tr}(\frac{g}{\sqrt{|I_u|}} Q_i^T dy_i + \lambda y_i^T dy_i) \\ &= \text{tr}((\frac{g}{\sqrt{|I_u|}} Q_i^T + \lambda y_i^T) dy_i) \end{aligned}$$

$$\frac{\partial J_{u,i}}{\partial y_i} = \frac{g}{\sqrt{|I_u|}} Q_i + \lambda y_i, \quad y_i \leftarrow y_i - \eta(\frac{g}{\sqrt{|I_u|}} Q_i + \lambda y_i)$$

关于隐式反馈

在测试中,我搜集了ml-ls中用户对电影打标签的行为作为隐式反馈,并进行了统计得到lookup table,格式为用户id->电影id列表,这样就能方便地使用SVD++模型进行训练.

Factorization Machine

概述

与线性回归相比,因子分解机使用隐特征向量的方式进行了二阶特征交叉,能够更有效地学习到不同属性之间的关系

编码

在本实验实现的因子分解机模型中,对电影的体裁属性进行了one-hot编码.在用户方面,尝试了两种不同的编码方案:

- 对年龄进行归一化、将性别转化为1/-1、将职业映射为21个实数并进行归一化
- 将年龄进行概念分层,并以one-hot形式表示,将性别以one-hot形式表示、将职业以one-hot形式表示

模型推导

Factorization Machine

$$\begin{aligned}\hat{y} &= \mu + \sum_{i=1}^n w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^n v_i v_j^\top x_i x_j \\ &= \mu + w^\top x + \frac{1}{2} \sum_{i=1}^n \sum_{j=i+1}^n v_i v_j^\top x_i x_j - \frac{1}{2} \sum_{i=1}^n v_i v_i^\top x_i^2 \\ &= \mu + w^\top x + \frac{1}{2} \left(\sum_{i=1}^n v_i x_i \right) \left(\sum_{i=1}^n v_i x_i \right)^\top - \frac{1}{2} \sum_{i=1}^n v_i v_i^\top x_i^2 \\ J &= (\hat{y} - y)^2\end{aligned}$$

$$\begin{aligned}dJ|_\mu &= (\hat{y} - y) d\hat{y} \quad \text{if } \hat{y} = y \\ &= g d\mu\end{aligned}$$

$$\frac{\partial J}{\partial \mu} = g, \quad \mu \leftarrow \mu - \eta g$$

$$\begin{aligned}dJ|_w &= g d\hat{y} \\ &= \text{tr}(g dW^\top X) \\ &= \text{tr}(g X^\top dW)\end{aligned}$$

$$\frac{\partial J}{\partial w} = g X, \quad w \leftarrow g X$$

$$\begin{aligned}dJ|_{v_i} &= g d\hat{y} \\ &= \frac{1}{2} g dV_i x_i \left(\sum_{i=1}^n v_i x_i \right)^\top + \frac{1}{2} g \left(\sum_{i=1}^n v_i x_i \right) x_i dV_i^\top - \frac{1}{2} g dV_i v_i^\top x_i^2 - \frac{1}{2} g v_i dV_i^\top x_i^2 \\ &= \text{tr} \left(g x_i \left(\sum_{i=1}^n v_i x_i \right)^\top dV_i - g x_i^\top v_i^\top dV_i \right) \\ &= \text{tr} \left(g \left[x_i \left(\sum_{i=1}^n v_i x_i \right)^\top - x_i^\top v_i^\top \right] dV_i \right)\end{aligned}$$

$$\frac{\partial J}{\partial v_i} = g \left[x_i \left(\sum_{i=1}^n v_i x_i \right) - x_i^\top v_i \right], \quad v_i \leftarrow v_i - \eta g \left[x_i \left(\sum_{i=1}^n v_i x_i \right) - x_i^\top v_i \right]$$

关于模型

- 在下面介绍的所有模型都是预测CTR的模型,即输出为0-1的概率,因为我在实验中使用的数据集为ml-100k电影评分数据集,因此在实现中我将最后进行逻辑回归的sigmoid函数进行了删除.
- 实际上对于整数得分的情况也可以使用逻辑回归softmax进行5分类,但考虑到其他数据集的输入如ml-ls包含小数得分,且电影得分的总体水平也是一个连续值,因此我认为还是使用回归模型相对比较自然.

Wide&Deep

概述

wide&deep在使用了并联的结构进行预测,包括输入为稀疏编码的线性回归部分与输入经过dense embedding后稠密编码的多层感知机部分,在输出层对于两个部分的输出进行加权与偏置后输出最终结果.

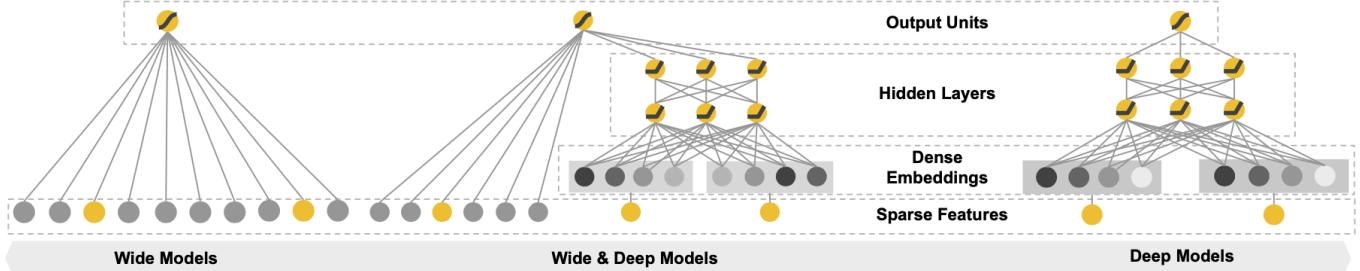


Figure 1: The spectrum of Wide & Deep models.

Wide component

在广度部分,输入的向量不仅是one-hot编码,还包括特征交叉部分.与因子分解不同,wide&deep中使用的方法相对而言更加原始,其公式为.

$$\phi_k(x) = \prod_{i=1}^d x_i^{c_{ki}}$$

其中k代表第k种交叉方式,而 c_{ki} 则为第k种交叉方式的指示参数,取值为0和1,0代表不参与交叉,1代表参与交叉.

简而言之,这种交叉方式类似场感知机制,可人工规定交叉位置从而避免同一个域中的one-hot编码进行无意义的交叉.

Deep component

深度部分为多层感知机,使用ReLU作为激活函数,输入为稠密编码特征.

编码方案

在实验实现的wide&deep中,我使用编码方案如下:

- 稀疏编码:用户与电影的one-hot编码
- 交叉部分:用户每个域的自交叉以及用户域和电影域的交叉
- 稠密部分:用户的非one-hot编码方案(上文提到),对于电影的19维one-hot编码,使用自编码器进行训练的到降维后的5维向量

公式推导

Wide & Deep

Deep: $X_d \rightarrow W_1 X_d + b_1 = z_1 \rightarrow \text{ReLU}(z_1) = x_2 \rightarrow W_2 x_2 + b_2 = z_2 \rightarrow \text{ReLU}(z_2) = x_3 \rightarrow W_3 x_3 + b_3 = y_d$

Wide: $X_s \rightarrow W_s^T X_s + b_s = y_w$

Merge: $\hat{y} = W_w \cdot y_w + W_d \cdot y_d + b$

$$J = (\hat{y} - y)^2$$

BP:

$$dJ = (\hat{y} - y) d\hat{y}, \quad \text{if } \hat{y} - y = g$$

$$= g d\hat{y}$$

$$1. = g dW_w \cdot y_w \rightarrow \frac{\partial J}{\partial W_w} = g y_w dW_w$$

$$2. = g W_w d y_w$$

$$2.1 = g W_w d W_s^T \cdot X_s \rightarrow \frac{\partial J}{\partial W_s} = g W_w X_s$$

$$2.2 = g W_w d b_s \rightarrow \frac{\partial J}{\partial b_s} = g W_w$$

$$3. = g d b \rightarrow \frac{\partial J}{\partial b} = g$$

$$4. = g dW_d y_d \rightarrow \frac{\partial J}{\partial W_d} = g y_d$$

$$5. = g W_d d y_d \quad \text{if } g W_d \neq g_3$$

$$= g_3 d y_d$$

$$5.1 = g_3 d W_3 X_3 \rightarrow \frac{\partial J}{\partial W_3} = g_3 X_3^T$$

$$5.2 = g_3 W_3 d X_3 = g_3 W_3 d \text{ReLU}(z_2)$$

$$= g_3 W_3 \text{diag}\left(\frac{d \text{ReLU}(z_2)}{d z_2}\right) d z_2, \quad \text{if } g_3 W_3 \text{diag}\left(\frac{d \text{ReLU}(z_2)}{d z_2}\right) = g_2$$

$$= g_2 d z_2$$

$$5.2.1 = g_2 d W_2 X_2 \rightarrow \frac{\partial J}{\partial W_2} = g_2^T X_2^T$$

$$5.2.2 = g_2 d b_2 \rightarrow \frac{\partial J}{\partial b_2} = g_2^T$$

$$5.2.3 = g_2 W_2 d X_2$$

$$= g_2 W_2 \text{diag}\left(\frac{d \text{ReLU}(z_1)}{d z_1}\right) d z_1, \quad \text{if } g_2 W_2 \text{diag}\left(\frac{d \text{ReLU}(z_1)}{d z_1}\right) = g_1$$

$$= g_1 d z_1$$

$$5.2.3.1 = g_1 d W_1 \cdot X_d \rightarrow \frac{\partial J}{\partial W_1} = g_1^T X_d^T$$

$$5.2.3.2 = g_1 d b_1 \rightarrow \frac{\partial J}{\partial b_1} = g_1^T$$

$\text{if } \text{diag}\left(\frac{d \text{ReLU}(z_k)}{d z_k}\right)$ 只是一种形式化的表达，其实际意义为：对于 forward 传播响应为 0 的节点，不进行 backward

$$\begin{array}{c} z_k \\ \begin{pmatrix} 6 \\ 7 \\ -2 \\ 0 \end{pmatrix} \end{array} \xrightarrow{\text{ReLU}} \begin{array}{c} x_{k+1} \\ \begin{pmatrix} 6 \\ 7 \\ 0 \\ 0 \end{pmatrix} \end{array}$$

$$g_{k+1} \cdot \text{diag}\left(\frac{d \text{ReLU}(z_k)}{d z_k}\right) \rightarrow g_k$$

$$(-1, 2, -3, 4) \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} \rightarrow (-1, 2, 0, 0)$$

DeepFM

概述

DeepFM在Wide&Deep的基础上进行了改进,将其进行部分交叉的wide部分变为了因子分解机,对于特征具有更好的建模能力.

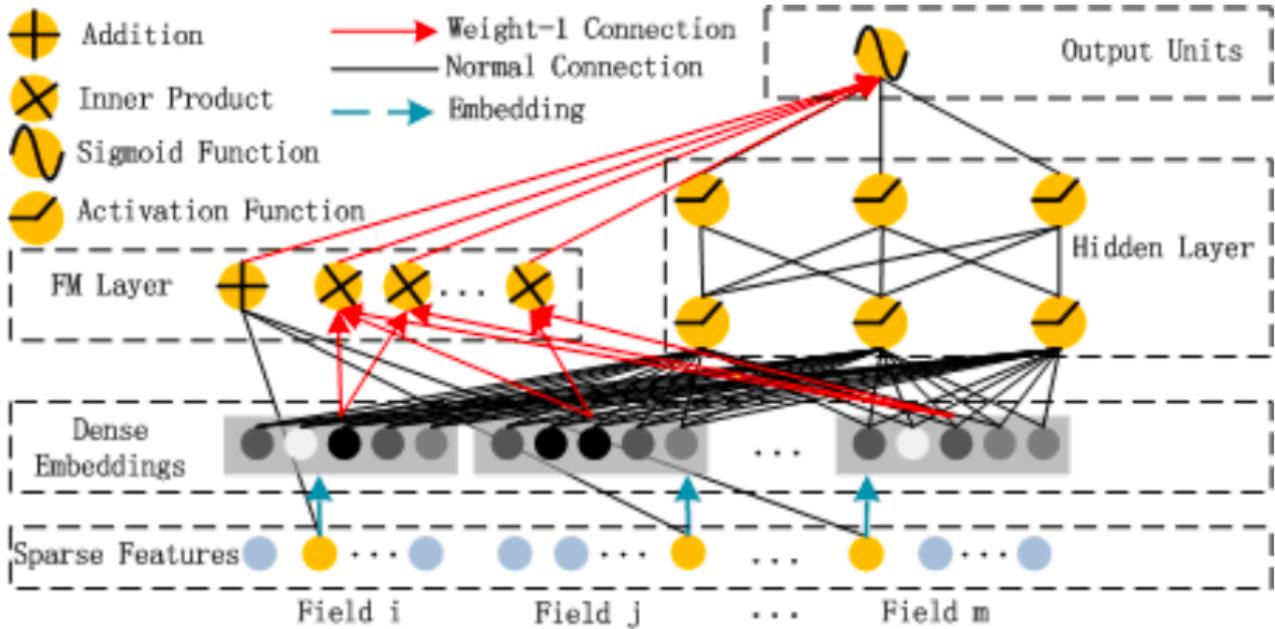


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

编码方案

We would like to point out the two interesting features of this network structure: 1) while the lengths of different input field vectors can be different, their embeddings are of the same size (k); 2) the latent feature vectors (V) in FM now serve as network weights which are learned and used to compress the input field vectors to the embedding vectors.

DeepFM论文中提到,对于FM与MLP都使用编码后的稠密向量作为输入,而FM的特征隐向量则类似神经网络中权重的角色.

因此,在本实验对DeepFM的实现中,我使用自编码器降维后的电影编码与进行特征工程后的非one-hot用户编码拼接后作为稠密向量的输入.

- 由于DeepFM为FM与MLP的线性组合,二者反向传播推导过程上文均已提到,在此不再赘述

Deep & Cross Network

概述

Deep & Cross相比于deepFM,其对于广度部分进行了改进,使用了交叉层,能够使得交叉不仅限于二阶.

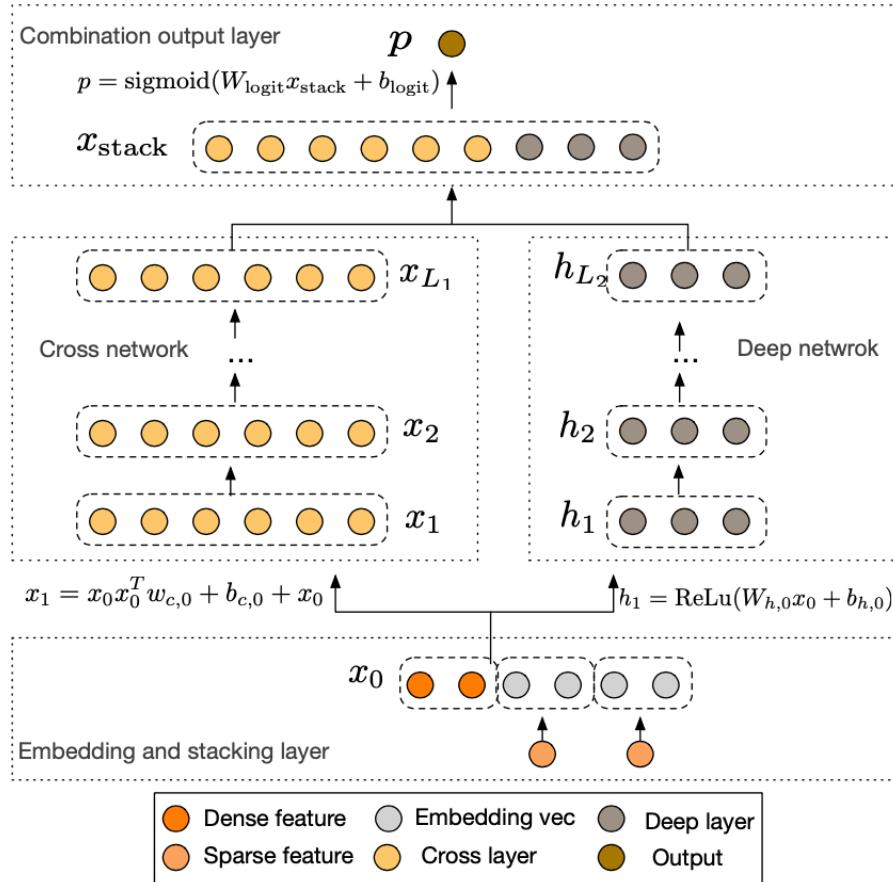


Figure 1: The Deep & Cross Network

CrossLayer

在CrossLayer中,每一层的输入与输出的关系为: $x_{l+1} = x_0x_l^T w_l + b_l + x_l = f(x_l, w_l, b_l) + x_l$

相当于高阶交叉结果与原始特征进行了交叉后又做了残差连接

In particular, the cross network approximates the polynomial class of the same degree in a way that is efficient, expressive and generalizes better to real-world datasets.

在论文中作者提到这么做是基于魏尔斯特拉斯近似理论,并指出这种交叉-连接操作相当于FM的泛化形式.

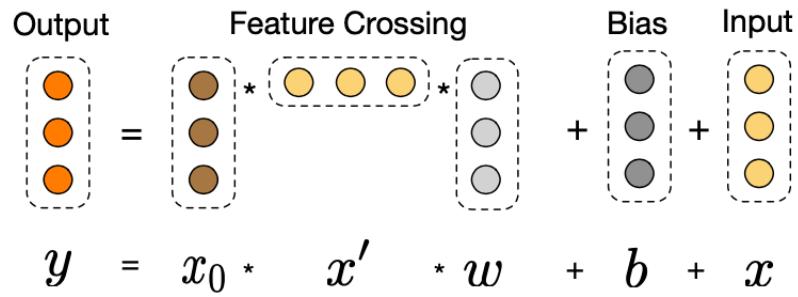
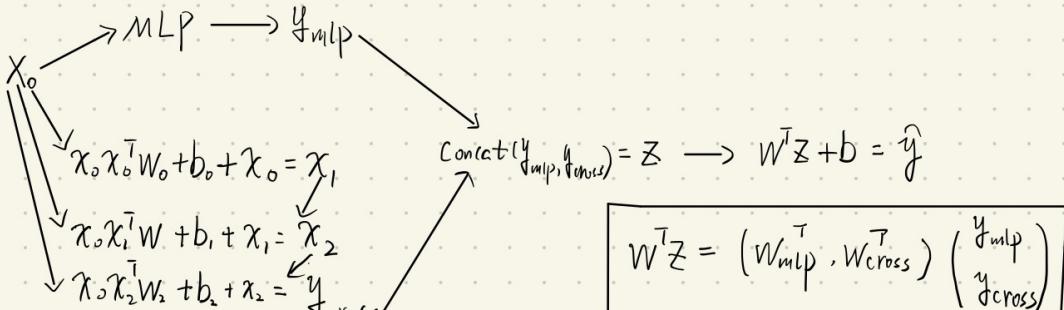


Figure 2: Visualization of a cross layer.

公式推导

Deep & Cross



$$J = (\hat{y} - y)^2$$

$$dJ = (\hat{y} - y) d\hat{y}, \text{ i.e. } \hat{y} - y = g$$

$$1. = g dW_{cross}^T y_{cross} \rightarrow \frac{\partial J}{\partial W_{cross}} = g \cdot y_{cross}$$

$$2. = g W_{cross}^T d y_{cross}, \text{ i.e. } g W_{cross}^T = g_3$$

$$2.1 = g_3 d b_2 \rightarrow \frac{\partial J}{\partial b_2} = g_3^T$$

$$2.2 = g_3 x_0 x_2^T d W_2 \rightarrow \frac{\partial J}{\partial W_2} = x_2 x_0^T g_3^T$$

$$2.3 = g_3 x_0 d x_2^T W_2 + g_3 d x_2$$

$$= \text{tr}(d x_2^T W_2 g_3 x_0) + \text{tr}(g_3 d x_2)$$

$$= \text{tr}((x_0^T g_3^T W_2^T + g_3) d x_2), \text{ i.e. } x_0^T g_3^T W_2^T + g_3 = g_2$$

$$= \text{tr}(g_2 d x_2)$$

$$2.3.1 = \text{tr}(g_2 d b_1) \rightarrow \frac{\partial J}{\partial b_1} = g_2^T$$

$$2.3.2 = \text{tr}(g_2 x_0 x_1^T d W_1) \rightarrow \frac{\partial J}{\partial W_1} = x_1 x_0^T g_2^T$$

$$2.3.3 = \text{tr}(g_2 x_0 d x_1^T W_1 + g_2 d x_1)$$

$$= \text{tr}(d x_1^T W_1 g_2 x_0) + \text{tr}(g_2 d x_1)$$

$$= \text{tr}(x_0^T g_2^T W_1^T + g_2) d x_1, \text{ i.e. } x_0^T g_2^T W_1^T + g_2 = g_1$$

$$2.3.3.1 = \text{tr}(g_1 d b_0) \rightarrow \frac{\partial J}{\partial b_0} = g_1^T$$

$$2.3.3.2 = \text{tr}(g_1 x_0 x_0^T d W_0) \rightarrow \frac{\partial J}{\partial W_0} = x_0 x_0^T g_1^T$$

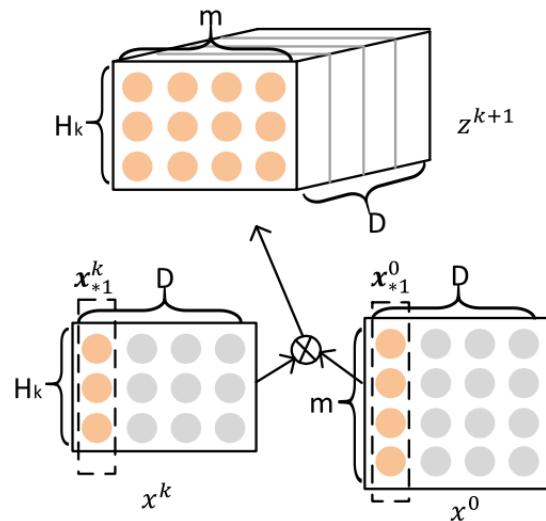
xDeepFM

概述

xDeepFM同时使用了线性层、交叉层和深度层.

在交叉层中,xDeepFM改进了DCN中的Cross Layer结构.在其提出的CIN层中,使用克罗内克积进行特征交叉,使用卷积得到特征图,并使用池化提取特征.

特征交叉



(a) Outer products along each dimension for feature interactions. The tensor Z^{k+1} is an intermediate result for further learning.

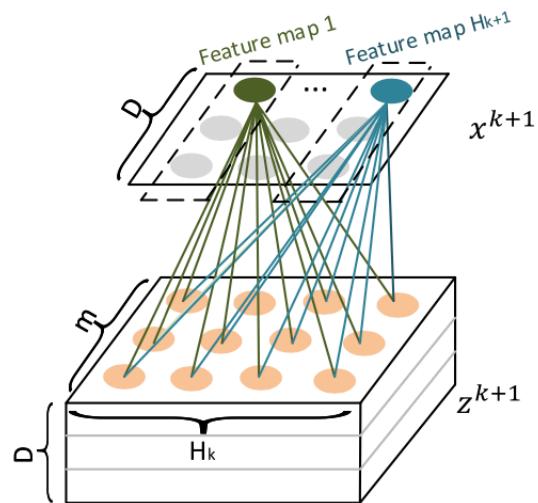
如图所示,左侧 x^k 为k阶特征图, x^0 为原始特征图.论文作者指出,原始特征图的获得方式为根据不同的域,将其特征编码填充至相同长度后并联拼接而成.其中D为域编码维数,m为特征域数量.

在进行外积的过程中,每对embedding维向同列向量做外积,并将结果作为一层,因此最终交叉结果的形状为 $H_k \times m \times D$

生成特征图

在得到交叉结果后,使用卷积操作生成新的特征图.

一个大小为 $H_k \times m$ 的卷积核沿着垂直于该平面的方向卷积,可以得到一个 $D \times 1$ 的向量,因此,使用 H_{k+1} 个该大小的卷积核进行一维卷积,并将结果进行并联拼接.可以得到一个形状为 $H_{k+1} \times D$ 的新的特征图.



(b) The k -th layer of CIN. It compresses the intermediate tensor Z^{k+1} to H_{k+1} embedding vectors (also known as *feature maps*).

提取特征向量

CIN层生成的特征图并不是直接用于最后的回归,而是使用池化操作在每一层提取特征.

在每次卷积完成生成特征图后,对每个域的交叉特征进行求和作为池化结果保存.

在所有的CIN层都结束后,将池化得到的特征串联拼接作为交叉特征输出并参与最终的线性回归.

关于特征

本实验实现的模型中使用的特征如下所示:

稀疏特征

稀疏特征用于进行线性回归,其获得方法如下

- 对电影体裁使用了one-hot编码表示
- 对年龄进行了概念分层并使用one-hot编码表示
- 对职业使用了one-hot编码表示
- 对性别使用了one-hot编码表示

将其拼接得到稀疏向量

稠密特征

稠密特征用于mlp和cin的输入,其获得方法如下

- 使用pytorch的embedding层对职业、性别、年龄、邮编进行了嵌入,其维度为3
- 对电影体裁使用自编码器进行降维,其维度为3

将其并联拼接可得到输入CIN的 5×3 的原始特征图

将其串联拼接可得到输入MLP的 15×1 的向量

实验结果

使用各种模型在ml-100k上得到的结果如下所示

模型名称	BSVD	SVD++	FM	Wide&Deep	deepFM	DCN	xDeepFM
RMSE	1.0399	1.0975	1.1588	1.1491	1.1101	1.1144	1.1429

结论分析

在进行训练与统计结果时我注意到了一个事实,就是随着模型复杂度的提升,训练时间和误差都有所上升.

尤其是后面的Wide&Deep系列模型,不由得让我怀疑正确率全部都是神经网络提供的,而交叉部分根本不work.

为此,我仅仅使用了3层MLP作为baseline进行测试,最终的RMSE误差在1.4左右,确实在一定程度上证明了交叉部分是有效的.

经过反思,我得出的结论如下:

1. Wide&Deep系列在发论文的时候使用的数据应该都是多次尝试调好超参数后进行实验得出的
2. 在进行深度网络训练时,数据量较小可能会甚至会导致欠拟合,由于ml-100k只有100,000条左右的数据,相比起经常用于CTR预测模型训练的criteo数据集包含的40,000,000条数据,实在是太小了
3. 这些论文的所实现的任务是CTR预测,标签只有0和1,而回归得到的值经过sigmoid函数的压缩,只能更倾向于两侧且被限定在了0,1之间.而如果进行回归任务,如本实验进行的1-5分评分预测,则有可能预测分数为9分,而标签为5分,这样就会大大增加了损失.
4. 对于结论3,我曾考虑过使用softmax函数将任务改为逻辑回归,使得结果尽可能贴近1、2、3、4、5这些离散的数值.但在进行模型的实现时我出于前面所说的原因,认为对于电影评分预测,还是使用回归更加自然一些,因此没有尝试分类模型.后续我也会进行一些尝试,验证一下分类能够提高这些模型在评分预测方面的表现.