Discrete Collaborative Filtering 阅读报告

背景

传统的基于矩阵分解的协同过滤(CF)中，将 m 个用户和 n 个项目的评分矩阵 $\mathbf{S} \in R^{m\times n}$ 分解成两个低维的矩阵 $\mathbf{U} \in R^{r\times m}$ 和$\mathbf{V} \in R^{r\times n}$ ，用户 i 与项目 j 的相似度就可以通过 $U_{*i}^T V_{*j}$ 计算。于是，基于CF的推荐自然会转化成一个相似度搜索问题——对于用户的top-K项推荐可以转换为根据用户查找与其最相似的top-K个项目。

这种方法在性能上存在瓶颈:

- 空间上，需要 $O(mr)$($_{或}$ $O(nr)$) 的空间去储存用户(或项目)向量
- 时间上，相似度搜索需要 $O(n)$

哈希的方法被广泛用于解决上述瓶颈。首先，将实数的向量编码成**二值**的向量，可以大大减少所需的储存空间。其次，相似度计算被 Hamming 空间中的比特运算所取代，线性扫描的时间复杂度显著降低，甚至通过构造查找表，使常数级的扫描时间成为可能。

然而，在本文之前，大家采用的都是一种**两阶段式**的哈希方法：1.**实值优化**(real-valued optimization) 2.**二值量化**(binary quantization)。即先丢弃离散约束，在实数基础上进行优化，再通过舍入、旋转的方法将获得的连续值转化为二值的整数。

作者认为，这种两阶段的方法过渡简化了离散约束，在二值量化的过程中由连续值到整数的偏差会产生较大的**量化损失。**尤其是在大型的系统中，需要用更长的编码长度以提高精度，反而造成了累积的错误，影响推荐的性能。

于是作者提出了称为 Discrete Collaborative Filtering(DCF) 的方法。

问题描述

- 将长度为 r 的的用户和项目的二值码分别表示为 $\mathbf{B} = [\mathbf{b}_1, \dots \mathbf{b}_m] \in \{\pm 1\}^{r\times m}$, $\mathbf{D} = [\mathbf{d}_1, \dots \mathbf{b}_n] \in \{\pm 1\}^{r\times n}$
- Hamming similarity

  $\mathbf{b}_i$ 和 $\mathbf{d}_j$ 的 Hamming similarity 定义为:

$$
\begin{aligned}
sim(i,j) = &\frac{1}{r}\sum_{k=1}^{r}\mathbb{I}(b_{ik} = d_{jk}) \\
=&\frac{1}{2r}(\sum_{k=1}^{r}\mathbb{I}(b_{ik} = d_{jk}) + r - \sum_{k=1}^{r}\mathbb{I}(b_{ik} \neq d_{jk})) \\
=&\frac{1}{2r}(r + \sum_{k=1}^{r} b_{ik}d_{jk}) \\
=&\frac{1}{2} + \frac{1}{2r}\mathbf{b}_i^T\mathbf{d}_j
\end{aligned}
$$

  即$\mathbf{b}_i$ 和 $\mathbf{d}_j$ 相同位的个数再除以编码的长度 r ，使得$sim(i,j) \in [0,1]$。

- 什么是好的编码?
  - 编码应该尽可能**短**(尽可能高效)
    - Balanced Partition：编码的每一位应该有50%的概率为1, 50%的概率为0, 尽量能够平衡的划分数据

      极端来说，如果某一位编码在所有数据中全为1或者全为0，那么这一位就没有意义

      即要满足：$\sum_{k=1}^{r} \mathbf{b}_{ik}$= 0，$\sum_{k=1}^{r} \mathbf{d}_{jk}$ = 0

- Decorrelation: 每一位编码应该是无关的, 尽可能减少冗余的信息

  $\mathbf{b}_i\mathbf{b}_i^T = m\mathbf{I}$, $\mathbf{d}_j\mathbf{d}_j^T = n\mathbf{I}$

- 将相似项映射到相似的二值码

  - $\arg\min_{B,D} \sum_{i,j\in V} (\mathbf{S}_{ij} - \mathbf{b}_i^T \mathbf{d}_j)^2$

    其中 $S_{ij} \leftarrow 2rS_{ij} - r$

- 于是DCF可以描述为:

$$\begin{aligned}&\underset{\mathbf{B},\mathbf{D}}{\operatorname{argmin}} \sum_{i,j\in\mathcal{V}} \left(S_{ij} - \mathbf{b}_i^T\mathbf{d}_j\right)^2, \\ &s.t.\ \mathbf{B} \in \{\pm1\}^{r\times m}, \mathbf{D} \in \{\pm1\}^{r\times n} \\ &\underbrace{\mathbf{B1} = 0, \mathbf{D1} = 0,}_{\text{Balanced Partition}}\ \underbrace{\mathbf{BB}^T = m\mathbf{I}, \mathbf{DD}^T = n\mathbf{I}}_{\text{Decorrelation}}\end{aligned}$$

  在传统的CF中为了防止过拟合，需要添加正则项，但由于 $\mathbf{B} \in \{\pm1\}^{r\times m}$， $\mathbf{D} \in \{\pm1\}^{r\times n}$, 所以正则项已经为常数。

- 由于 Balanced Partition 和 Decorrelation 这两个约束可能会使 DCF 没有可行解，作者建议放宽这两个约束。

  定义:

  - $\mathcal{B} = \{\mathbf{X} \in R^{r\times m} | \mathbf{X1} = 0\, ,\ \mathbf{XX^T} = m\mathbf{I}\}$
  - $\mathcal{D} = \{\mathbf{Y} \in R^{r\times n} | \mathbf{Y1} = 0\, ,\ \mathbf{YY^T} = n\mathbf{I}\}$
  - $d(\mathbf{B}, \mathcal{B}) = min_{\mathbf{X}\in\mathcal{B}} ||\mathbf{B} - \mathbf{X}||_F$
  - $d(\mathbf{D}, \mathcal{D}) = min_{\mathbf{Y}\in\mathcal{D}} ||\mathbf{D} - \mathbf{Y}||_F$

  将原始的 DCF 放宽成:

$$\begin{aligned}&\underset{\mathbf{B},\mathbf{D}}{\operatorname{argmin}} \sum_{i,j\in\mathcal{V}} \left(S_{ij} - \mathbf{b}_i^T\mathbf{d}_j\right)^2 + \alpha d^2(\mathbf{B},\mathcal{B}) + \beta d^2(\mathbf{D},\mathcal{D}) \\ &s.t.,\ \ \mathbf{B} \in \{\pm1\}^{r\times m}, \mathbf{D} \in \{\pm1\}^{r\times n}\end{aligned}$$

  其中:

$$\begin{aligned}d^2(\mathbf{B},\mathcal{B}) &= ||\mathbf{B} - \mathbf{X}||_F^2 \\ &= tr((\mathbf{B} - \mathbf{X})(\mathbf{B} - \mathbf{X})^T) \\ &= tr(\mathbf{BB}^T + \mathbf{XX}^T - \mathbf{BX}^T - \mathbf{XB}^T) \\ &= tr(2mI) - 2tr(\mathbf{BX}^T) \\ &= constant - 2tr(\mathbf{B}^T\mathbf{X})\end{aligned}$$

  因此最终提出的学习模型为:

$$\begin{aligned}&\arg\min_{B,D,X,Y} \sum_{i,j\in\mathcal{V}} (S_{ij} - \mathbf{b}_i^T\mathbf{d}_j) - 2\alpha tr(\mathbf{B}^T\mathbf{X}) - 2\beta tr(\mathbf{D}^T\mathbf{Y}) \\ &s.t.,\mathbf{X1} = 0, \mathbf{XX^T} = m\mathbf{I}, \mathbf{Y1} = 0, \mathbf{YY^T} = n\mathbf{I} \\ &\mathbf{B} \in \{\pm1\}^{r\times m}, \mathbf{D} \in \{\pm1\}^{r\times n}\end{aligned}$$

解决方法

交替地求解方程中DCF模型的四个子问题：$\mathbf{B}$、$\mathbf{D}$、$\mathbf{X}$和$\mathbf{Y}$。

- $\mathbf{B}$-子问题--固定$\mathbf{D}$、$\mathbf{X}$和$\mathbf{Y}$，更新$\mathbf{B}$

  每个 user 是独立的，因此可以并行更新 $\mathbf{b}_i$.

$$\mathop{\arg\min}_{b_i\in\{\pm1\}^r} \mathbf{b}_i^T(\sum_{j\in\mathcal{V}_i}\mathbf{d_j}\mathbf{d_j^T})\mathbf{b}_i - 2(\sum_{j\in\mathcal{V}_i}S_{ij}\mathbf{d}_j^T)\mathbf{b}_i - 2\alpha\mathbf{x}_i^T\mathbf{b}_i$$

由于这个问题是 NP-hard 的，所以作者使用了Discrete Coordinate Descent (DCD) 的方法对 $\mathbf{b}_i$ **逐位**进行更新:

- 令 $\hat{b}_{ik} = \sum_{j\in\mathcal{V}_i}(S_{ij} - \mathbf{d}_{j\bar{k}}^T\mathbf{b}_{i\bar{k}})d_{jk} + \alpha x_{ik}$为 $\mathbf{b}_i$ 的第 k 位，令 $\mathbf{b}_{i\bar{k}}$ 为 $\mathbf{b}_i$ 中不包括 b_{ik} 的剩余项
- 计算 $\hat{b}_{ik} = \sum_{j\in\mathcal{V}_i}(S_{ij} - \mathbf{d}_{j\bar{k}}^T\mathbf{b}_{i\bar{k}})d_{jk} + \alpha x_{ik}$
- 更新 $b_{ik} \leftarrow sgn(K(\hat{b}_{ik}, b_{ik}))$,其中$K(x,y)=x$ if $x\neq 0$,otherwise $K(x,y)=y$

根据附录的推导，当 b_{ik} 与 $\bar{b}_{ik}$ 与同号时目标值最小。

- $\mathbf{X}$-子问题--固定$\mathbf{B}$、$\mathbf{D}$和$\mathbf{Y}$，更新$\mathbf{X}$

$$\mathop{\arg\max}_{\mathbf{X}} tr(\mathbf{B}^T\mathbf{X}), s.t. \mathbf{X1}=0, \mathbf{XX^T}=m\mathbf{I}$$

作者使用了小矩阵**奇异值分解**的方法。由于$\mathbf{X}$-子问题的推导过程，我并没有完全看懂，所以这里就不详细说明了。

扩展至样本外的数据

当一个新的用户产生的时候，不需要重新训练 DCF。只需要根据现有的评分数据，对新的用户求解 $\mathbf{B}$-子问题 即可。并且，对于单个用户来说没有必要考虑 Balanced Partition 和 Decorrelation 这两个约束。

收敛性

作者证明了 DCF 是收敛的，并且在实验中发现大约10~20次迭代之后就会收敛。

初始化

去掉 DCF 的二值约束，再将求解出的实数值转化成整数值，作为 DCF 的初始化值。

$$\mathop{\arg\max}\sum_{i,j\in\mathcal{V}}(S_{ij}-\mathbf{u}_i^T\mathbf{v}_j)^2 + \alpha||\mathbf{U}||_F^2 + \beta||\mathbf{V}||_F^2 - 2\alpha tr(\mathbf{U}^T\mathbf{X}) - 2\beta tr(\mathbf{V}^T\mathbf{Y})$$
$$s.t., \mathbf{X1}=0, \mathbf{XX^T}=m\mathbf{I}, \mathbf{Y1}=0, \mathbf{YY^T}=n\mathbf{I}$$

代码理解

- 目录结构

|-- test.m //入口
|-- DCF.m //DCF
|-- DCFinit.m //DCF初始化
|-- ScaleScore.m //评分放缩
|-- DCDmex.c //$\mathbf{B}/\mathbf{D}$-子问题
|-- UpdateSVD.m //$\mathbf{X}/\mathbf{Y}$-子问题
|-- my_MGS.m //GS 正交化

- 主要功能在于 DCDmex.c 和 UpdateSVD.m 这两个文件上
  - DCDmex.c
    - 这部分的实现不是根据论文的正文而是根据论文的**脚注3**来实现的
    - 标识符(**以解决$\mathbf{B}$-子问题为例**)
      - ss：- $\hat{b}_{ik}$

- MM: $\mathbf{D}_i\mathbf{D}_i^T$
- Ms: $\mathbf{D}_i\mathbf{s}_i$
- x[k]: x_{ik}

- 首先计算$\hat{b}_{ik}$，再根据$\hat{b}_{ik}$的符号对b_{ik}进行更新

```
//DCDmex.c
..............
while (!converge){
    no_change_count = 0;
    for (k = 0; k < r; k ++){//for each bit in b
        ss = 0;
        for (i = 0; i < r; i ++)
            if (i != k)
                ss += MM[k+i*r]*b[i];
        ss -= Ms[k]+x[k];

        //update
        if (ss > 0){
            if (b[k] == -1)
                no_change_count ++;
            else
                b[k] = -1;
        }
        else if (ss < 0){
            if (b[k] == 1)
                no_change_count ++;
            else
                b[k] = 1;
        }
        else
            no_change_count ++;
    }
    if ((it >= (int)maxItr-1) || (no_change_count == r))
        converge = true;
    it ++;
}
..............
```

- UpdateSVD.m
  - 标识符(**以解决$\mathbf{X}$-子问题为例**)
    - b : r 即code length
    - W: $\mathbf{B}$
    - JW: $\overline{\mathbf{B}}^T$
    - P: $[\mathbf{P}_b \quad \hat{\mathbf{P}}_b]$ 特征向量
    - ss: $\sum_\mathbf{b}^\mathbf{2}$ 特征值
    - Q: $[\mathbf{Q}_b \quad \hat{\mathbf{Q}}_b]$
    - H_v: Unpated $\mathbf{X}$
  - 首先，进行奇异值分解，如果得出的特征值有零，则需要通过 GS正交化 补充向量的个数至 r。然后再根据公式计算出更新后的 $\mathbf{X}$
  - ```
    %UpdateSVD.m
    function H_v = UpdateSVD(W)
    %UpdateSVD: update rule in Eq.(16)
    [b,n] = size(W);
    ```
    ```

```
m = mean(W,2);
JW = bsxfun(@minus,W,m);
JW = JW';
[P,ss] = eig(JW'*JW);
ss = diag(ss);
zeroidx = (ss <= 1e-10);
if sum(zeroidx) == 0
    H_v = sqrt(n)*P*(JW*P*diag(1./sqrt(ss)))';
else
    ss = ss(ss>1e-10);
    Q = JW*P(:,~zeroidx)*diag(1./sqrt(ss));
    Q = my_MGS(Q, b);
    H_v = sqrt(n)*P *Q';
end
end
```

我的看法

- 创新点
    - DCF 始终严格执行**二值约束**，没有放宽到连续实值，减少了量化损失。
    - 使用了 Balanced Partition 和 Decorrelation 的约束，以生成紧凑以及信息丰富的编码。
    - 在此基础上提出了较高效的算法，通过 DCD 和 SVD 等方法求解。
- 作者在建模时，通过放宽 Balanced Partition 和 Decorrelation 的约束使得原来无可行解的问题变为有解，我认为是比较巧妙的地方。
- 此外，作者在求解的时候使用了交替优化、DCD 和 SVD的方法，并最终证明了其收敛性。我认为这里是最困难的地方，因为作者也在多处提到之前的工作都是先在实数上优化再rounding off，意味着直接在离散值上求解是困难的。而 作者提出的解法不仅复杂度低，而且性能好，外加上 Balanced Partition 和 Decorrelation 的约束又减少了的编码长度，进一步减少了量化损失。