Field-aware Factorization Machines for CTR Prediction

0. Summary

1. Research Objective

2. Problem Statement

CTR问题。

3. Methods

在FM中，每个特征值对应一个隐含向量。

比如现有三部电影，One-Hot编码之后：

id	x_0（第一部电影）	x_1（第二部电影）	x_2（第三部电影）
158	0	0	1

ω_2是特征x_2对应的隐含向量，也就是第三部电影对应的和其他特征的交互信息。

如下图，与FM不同的是，FFM增加了特征值对应的领域信息，比如ESPN属于Publisher这个领域，Nike属于Advertiser这个领域，这也是FFM（Field-Aware Factorization Machines）名称的由来之处。

Clicked	Publisher(P)	Advertiser(A)	Gender(G)
Yes	ESPN	Nike	Male

在FFM中，每个特征值包含多个隐含向量，比如ESPN这个特征值，它包含两个隐含向量$\omega_{ESPN,A}$，$\omega_{ESPN,G}$，其中$\omega_{ESPN,A}$对ESPN与Adertiser这个领域内的特征的交互信息进行建模，$\omega_{ESPN,G}$对ESPN与Gender这个领域内的特征的交互信息进行建模。

针对上面这个例子，对FM来说：

$$\phi_{FM} = \omega_{ESPN} \cdot \omega_{Nike} + \omega_{ESPN} \cdot \omega_{Male} + \omega_{Nike} \cdot \omega_{Male}$$

对FFM来说：

$$\phi_{FFM} = \omega_{ESPN,A} \cdot \omega_{Nike,P} + \omega_{ESPN,G} \cdot \omega_{Male,P} + \omega_{Nike,G} \cdot \omega_{Male,A}$$

可以这样理解，FM中的每个隐含向量学到的是该特征值和其他特征值的交互信息，FFM中每个隐含向量学到的是该特征和对应领域内特征的交互信息。

> Mathematically

$$\phi_{FFM}(\omega, x) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} (\omega_{j_1,f_2} \cdot \omega_{j_2,f_1}) x_{j_1} x_{j_2}$$

其中 f_1和f_2是属性j_1和j_2对应的领域（field）。

在FM和FFM中，特征对应的隐含向量的维度为 k，在FFM中，因为每个特征对应的隐含向量（有多个）只需要考虑该特征与某个领域的交互，例如$\omega_{ESPN,A}$只考虑ESPN这个特征与Adertiser这个领域之间的交互，而在FM中，每个特征对应的隐含向量（只有一个），它包含该特征与所有特征之间的交互信息，所蕴含的信息量更大，所以一般来说$k_{FFM} \ll k_{FM}$。

> Optimization

采用SGD优化算法，自适应学习率算法采用的是Adagrad（研究表明Adagrad在矩阵分解中表现出更好的性能）。

每次从训练集中采样一个样本，梯度信息

4. Evaluation

Model and implementation	parameters	training time (seconds)	public set logloss	public set rank	private set logloss	private set rank
LM-SG	$\eta = 0.2, \lambda = 0, t = 13$	527	0.46262	93	0.46224	91
LM-LIBLINEAR-CD	$s = 7, c = 2$	1,417	0.46239	91	0.46201	89
LM-LIBLINEAR-Newton	$s = 0, c = 2$	7,164	0.46602	225	0.46581	222
Poly2-SG	$\eta = 0.2, \lambda = 0, B = 10^7, t = 10$	12,064	0.44973	14	0.44956	14
Poly2-LIBLINEAR-Hash-CD	$s = 7, c = 2$	24,771	0.44893	13	0.44873	13
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 40, t = 8$	2,022	0.44930	14	0.44922	14
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 100, t = 9$	4,020	0.44867	11	0.44847	11
LIBFM	$\lambda = 40, k = 40, t = 20$	23,700	0.45012	14	0.45000	15
LIBFM	$\lambda = 40, k = 40, t = 50$	131,000	0.44904	14	0.44887	14
LIBFM	$\lambda = 40, k = 100, t = 20$	54,320	0.44853	11	0.44834	11
LIBFM	$\lambda = 40, k = 100, t = 50$	398,800	0.44794	9	0.44778	8
FFM	$\eta = 0.2, \lambda = 2 \times 10^{-5}, k = 4, t = 9$	6,587	0.44612	3	0.44603	3

(a) `Criteo`

Model and implementation	parameters	training time (seconds)	public set logloss	public set rank	private set logloss	private set rank
LM-SG	$\eta = 0.2, \lambda = 0, t = 10$	164	0.39018	57	0.38833	64
LM-LIBLINEAR-CD	$s = 7, c = 1$	417	0.39131	115	0.38944	119
LM-LIBLINEAR-Newton	$s = 0, c = 1$	650	0.39269	182	0.39079	183
Poly2-SG	$\eta = 0.2, \lambda = 0, B = 10^7, t = 10$	911	0.38554	10	0.38347	10
Poly2-LIBLINEAR-Hash-CD	$s = 7, c = 1$	1,756	0.38516	10	0.38303	9
Poly2-LIBLINEAR-Hash-Newton	$s = 0, c = 1$	27,292	0.38598	11	0.38393	11
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 40, t = 8$	574	0.38621	11	0.38407	11
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 100, t = 9$	1,277	0.38740	17	0.38531	15
LIBFM	$\lambda = 40, k = 40, t = 20$	18,712	0.39137	122	0.38963	127
LIBFM	$\lambda = 40, k = 40, t = 50$	41,720	0.39786	935	0.39635	943
LIBFM	$\lambda = 40, k = 100, t = 20$	39,719	0.39644	747	0.39470	755
LIBFM	$\lambda = 40, k = 100, t = 50$	91,210	0.40740	1,129	0.40585	1,126
FFM	$\eta = 0.2, \lambda = 2 \times 10^{-5}, k = 4, t = 4$	340	0.38411	6	0.38223	6

(b) `Avazu`

	statistics			logloss			
Data set	# instances	# features	# fields	LM	Poly2	FM	FFM
`KDD2010-bridge`	20,012,499	651,166	9	0.27947	0.2622	0.26372	<u>0.25639</u>
`KDD2012`	149,639,105	54,686,452	11	0.15069	0.15099	0.15004	<u>0.14906</u>
`phishing`	11,055	100	30	0.14211	0.11512	<u>0.09229</u>	0.1065
`adult`	48,842	308	14	0.3097	0.30655	0.30763	<u>0.30565</u>
`cod-rna` (dummy fields)	331,152	8	8	0.13829	0.12874	<u>0.12580</u>	0.12914
`cod-rna` (discretization)	331,152	2,296	8	0.16455	0.17576	0.16570	<u>0.14993</u>
`ijcnn` (dummy fields)	141,691	22	22	0.20093	0.08981	0.07087	<u>0.0692</u>
`ijcnn` (discretization)	141,691	69,867	22	0.21588	0.24578	0.20223	<u>0.18608</u>

Table 4: Comparison between LM, Poly2, FM, and FFMs. The best logloss is underlined.

5. Conclusion

在某些数据集上，FFM比LM，Poly2，FM表现出更好的性能（logloss更小），但是它的训练时间也相对较长。

在某些数据集上，LM/Poly2/FM/FFM性能相当，一个解释是，该数据集包含的特征之间没有交互，或者说交互信息量很小，没法挖掘出信息。

6. Notes

1. 模型汇总
    1. LM，Linear Model，不考虑特征之间的交互，拟合效果不佳。

$$\phi_{\mathrm{LM}}(\boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}.$$

    2. Poly2，考虑特征之间的二阶交叉项，但是在数据十分稀疏的情况下，模型很难学到东西，也就是很难准确估算ω。

$$\phi_{\mathrm{Poly2}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} w_{h(j_1,j_2)} x_{j_1} x_{j_2}$$

    3. FM，Factorization Machines，为了解决Poly2中存在的问题，FM为每个特征值引入隐含向量（embedding vector），通过隐含向量之间的内积来表示特征之间的交互，解决了数据稀疏性的问题。

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

    4. FFM，Field-aware Factorization Machines，引入了域的概念，每个特征都属于一个相应的域，另外每个特征包含多个隐含向量，每个隐含向量表示该特征与其他域内所有特征的交互信息。

$$\phi_{\mathrm{FFM}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} (\boldsymbol{w}_{j_1,f_2} \cdot \boldsymbol{w}_{j_2,f_1}) x_{j_1} x_{j_2}$$

2. 为了降低参数的个数，Poly2采用hash的方法，如下：

$$\phi_{\mathrm{Poly2}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} w_{h(j_1,j_2)} x_{j_1} x_{j_2}$$

$$h(j_1, j_2) = (\frac{1}{2}(j_1 + j_2)(j_1 + j_2 + 1) + j_2) \bmod B$$

    相当于多个特征组合共用同一个参数，其中B为模型参数的个数，用户自己指定。

3. PITF（Pairwise Interaction Tag Factorization）是FM的变体，FFM源自PITF。

4. 采用较小的“基准学习率”（Adagrad会自适应更新学习率，在此记η为基准学习率），模型收敛的速度很慢，采用较大的“基准学习率”模型在验证集上的损失值降低地很快，但是容易出现过拟合现象。本篇文章在模型训练过程中采用了Early Stopping策略来解决这个问题，具体地：
    1. 将数据集分为训练集和验证集两部分
    2. 在每轮训练结束时，计算模型在验证集上的损失值
    3. 如果模型在验证集上的损失值上升，那么记下来当前已经训练的轮数E，转向第四步
    4. 使用全部的数据（包括训练集和验证集）重新训练模型E轮

7. References