因子机
Factorization Machine

左元

2016/12

Outline

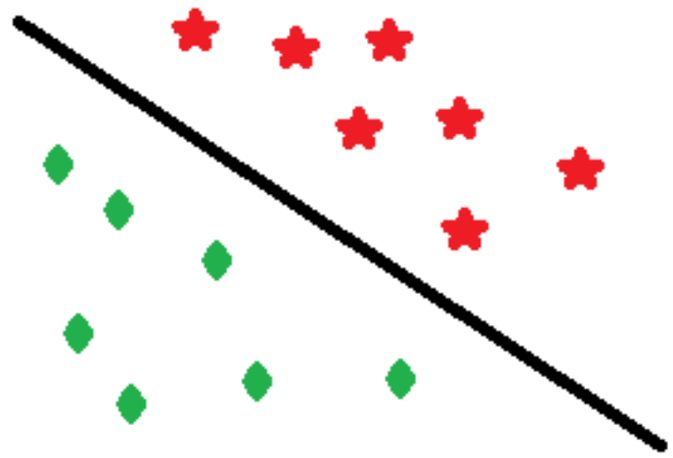
- 因子机简介（FM）
- 因子机 vs 矩阵分解协同过滤
- 因子机 vs 树模型
- 因子机 vs 神经网络
- FM 的变种：FFM，高阶FM

FM：因子机简介

从线性模型说起

- 线性模型

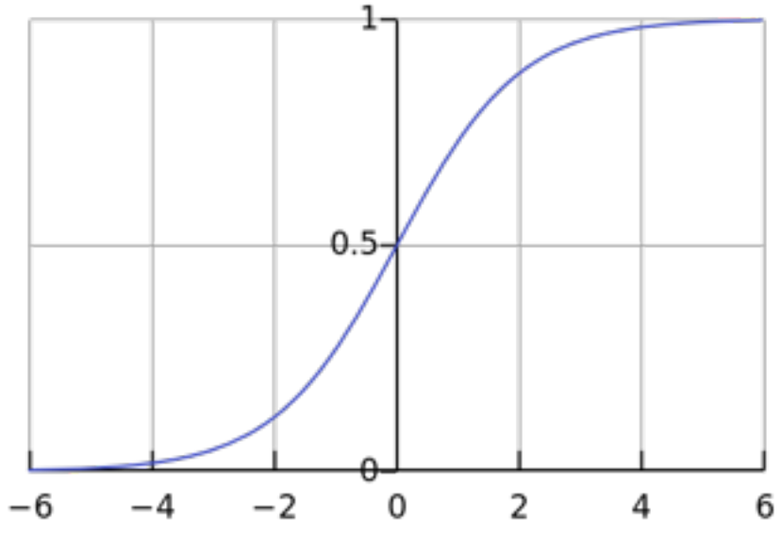
$$\phi(\mathbf{w}, \mathbf{x}) = \mathbf{w}^T \mathbf{x} = \sum_{j \in C_1} w_j x_j,$$



- 逻辑回归：对线性模型的输出做sigmoid变换

$$\hat{y} = \text{sigmoid}(\phi) = \frac{1}{1 + \exp(-\phi(w, x))}$$

$$\min_{\mathbf{w}} \quad \sum_{i=1}^{L} \left(\log\left(1 + \exp(-y_i \phi(\mathbf{w}, \mathbf{x}_i)\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right)$$



线性模型的特点

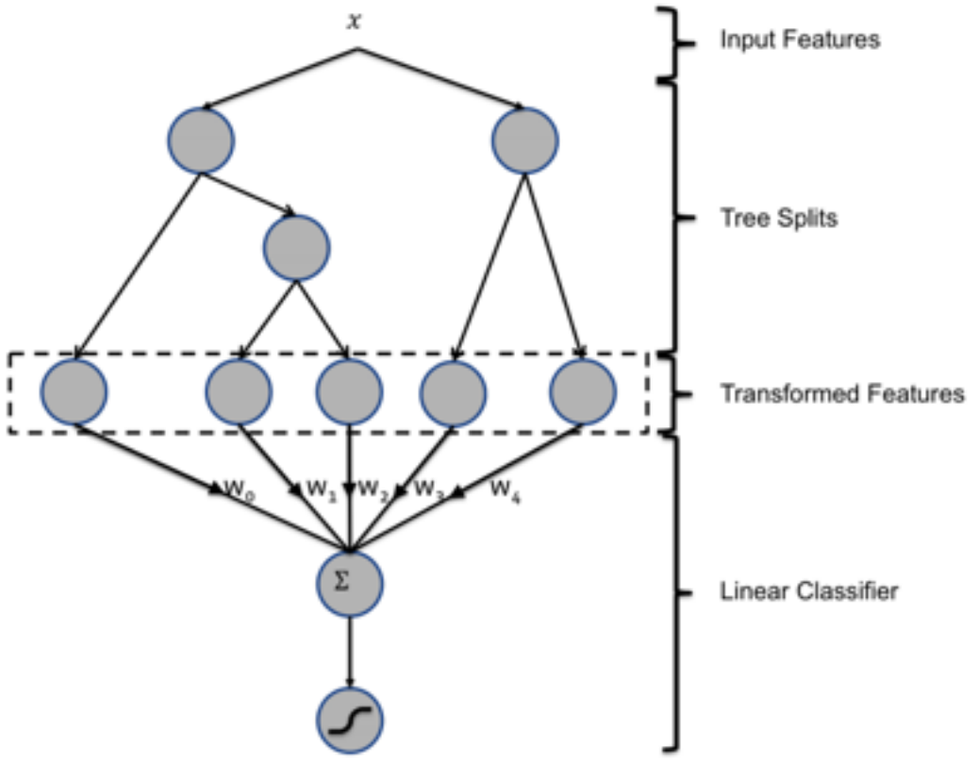
- 判决面是超平面
- 通常会将特征离散化以解决单变量的非线性问题：
  - w_年龄 = 0.5，y = 是否为高收入人群?
- 简单高效，可解释性强
- 早期一统江湖：推荐，排序，广告CTR etc
- 为了实现特征间的非线性，需要人工特征组合，后来演变为决策树、gbdt自动特征组合，DNN特征组合（deep-width model）

非线性改进：离散化

- 将连续变量分成若干段，使得模型可以在每一段学出不同的权值而达到学到非线性关系
  - 年龄：0-18，19-25，25-30，30+
  - onehot编码或序编码
  - eg: 年龄=20: [0, 1, 0, 0] 或 [1, 1, 0, 0]
- 确定离散化分位点方法：
  - 根据特征的特点人工划分：例如年龄分为老中青
  - 等距划分
  - 等频划分
  - 单变量决策树（信息增益，gini系数，属于有监督的方法）

非线性改进：特征组合

- 人工特征组合：中年+性别男=中年大叔，少年+性别女=萝莉 etc
  - 根据业务需求，认为的将两类及以上的特征做交叉
    - 性别+年龄， 时间+空间（职业标签优化）
- 决策树自动特征组合
  - 树的一个叶子结点对应一个路径，也即是一条规则



非线性改进：特征组合

- 二次函数组合（多项式核函数的SVM, d=2）

$$\begin{aligned}\phi(x) &= b + \sum_i w_i x_i + \sum_{i<j} \sum_j W_{ij} x_i x_j \\ &= b + w^T x + x^T W x\end{aligned}$$

- 简单的二次函数组合存在的问题
  - 参数个数随特征维度的平方增长
  - 过多的参数要求更多的数据量
  - 对于高度稀疏数据，数据中可能缺少大量 xi xj != 0的模式
    - 推荐中 xi 代表用户， xj 代表商品，用户没有过行为的商品远远大于有行为的商品
    - 广告 CTR 预估中，xi 代表用户 xj 代表广告，用户没有过行为的广告远远大于有行为的广告
  - 高度稀疏的数据中，维度可能上百万维甚至上亿维，参数数目将是 亿亿 个，往往是没有这么多的样本

改进方案：因子机

- 对权值矩阵施加低秩约束

$$W_{ij} = v_i^T v_j, v_i \in R^k$$

$$W = s(V^T V), V \in R^{k \times n}, W \in R^{n \times n}$$

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

- 好处及解释
  - k << n 极大地减少了参数数目，使模型泛化能力增强
  - 由于W元素不独立，那么没有见到过的模式可以通过见过的模式学习
  - 线性计算时间复杂度（只跟非0特征数目有关）

线性时间复杂度的证明

- 关键技巧：分解为和的平方与平方的和两项

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle\, x_i\, x_j \\
=&\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle\, x_i\, x_j - \frac{1}{2}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_i\rangle\, x_i\, x_i \\
=&\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{j,f}\, x_i\, x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{i,f}\, x_i\, x_i\right) \\
=&\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)\left(\sum_{j=1}^{n} v_{j,f}\, x_j\right) - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right) \\
=&\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)
\end{aligned}
$$

- Rendle S. Factorization Machines[C]. international conference on data mining, 2010.
- Rendle S. Factorization Machines with libFM[J]. ACM Transactions on Intelligent Systems and Technology, 2012, 3(3).

因子机目标函数及梯度

- 目标函数(以二分类为例，回归问题类似)：

$$\hat{y} = \frac{1}{1 + \exp(-\phi)}$$

$$\text{loss} = \frac{1}{N} \sum_i \log(1 + \exp(-y_i \phi_i(x_i))) + \Omega(w, W)$$

- 梯度计算也是线性的（只与非0特征数目有关）

$$\frac{\partial \text{loss}_i}{\partial \theta} = \frac{-y_i \phi(x_i)}{1 + \exp(-y_i \phi(x_i))} \frac{\partial \phi(x)}{\partial \theta} + \frac{\partial \Omega}{\partial \theta}$$

$$\frac{\partial \phi(x)}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

优化方案

- SGD 随机梯度下降
- ALS 适用于回归问题
- MCMC 贝叶斯推断优化方案，详细看论文
- 软件：
    - 单机多线程版本 libFM
    - 网上的Spark版本FM，我增加了自适应学习率，目前DMSPA可用
    - difacto DMLC出品的基于自己实现的PS实现的分布式FM
    - 基于 tensorflow 的分布式FM　https://github.com/kopopt/fast_tffm

Rendle S. Factorization Machines with libFM[J]. ACM Transactions on Intelligent Systems and Technology, 2012, 3(3).

一个例子：推荐

- movielens 评分预测

Fig. 1. Example for sparse real valued feature vectors $\mathbf{x}$ that are created from the transactions of example 1. Every row represents a feature vector $\mathbf{x}^{(i)}$ with its corresponding target $y^{(i)}$. The first 4 columns (blue) represent indicator variables for the active user; the next 5 (red) indicator variables for the active item. The next 5 columns (yellow) hold additional implicit indicators (i.e. other movies the user has rated). One feature (green) represents the time in months. The last 5 columns (brown) have indicators for the last movie the user has rated before the active one. The rightmost column is the target – here the rating.

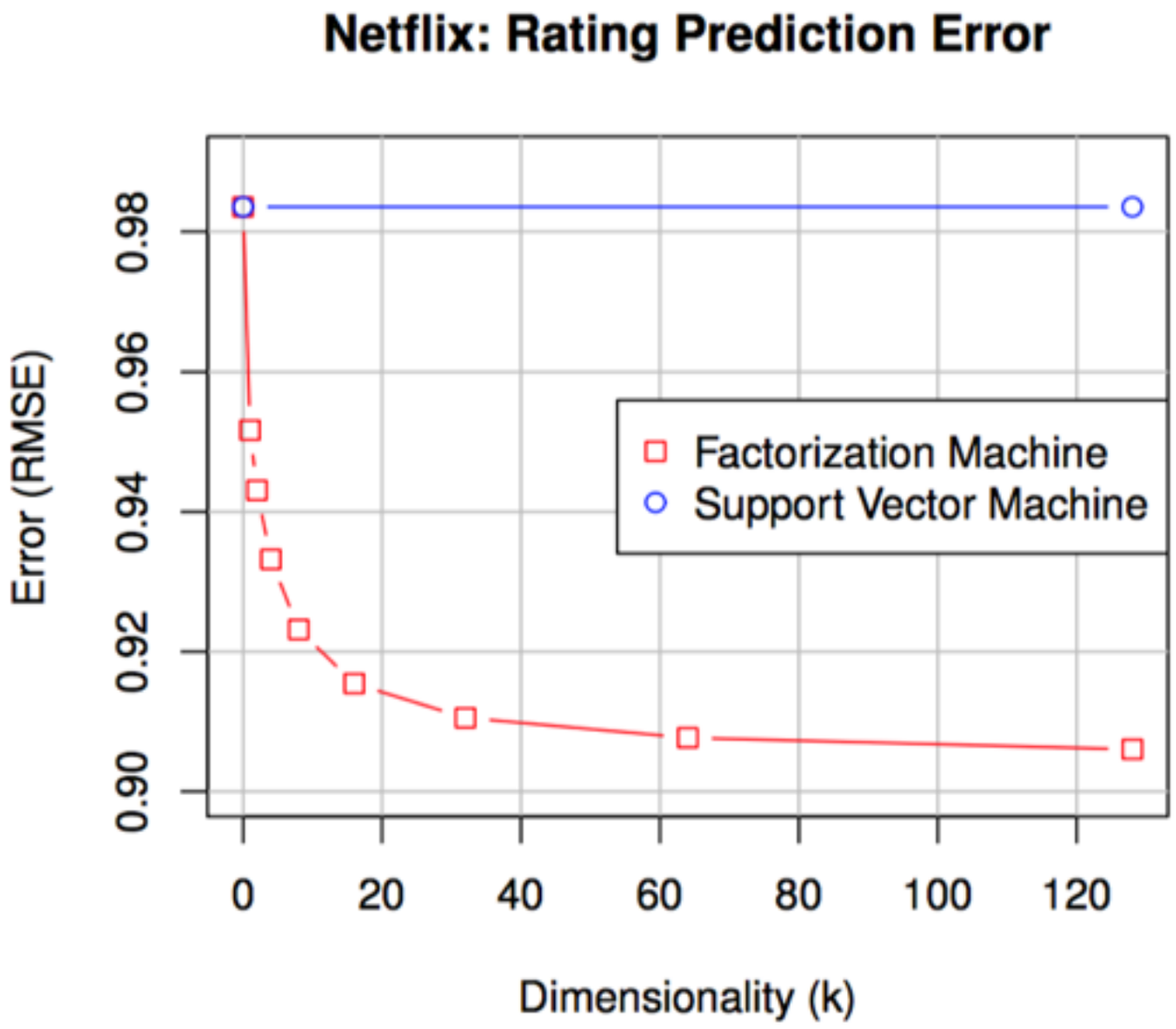
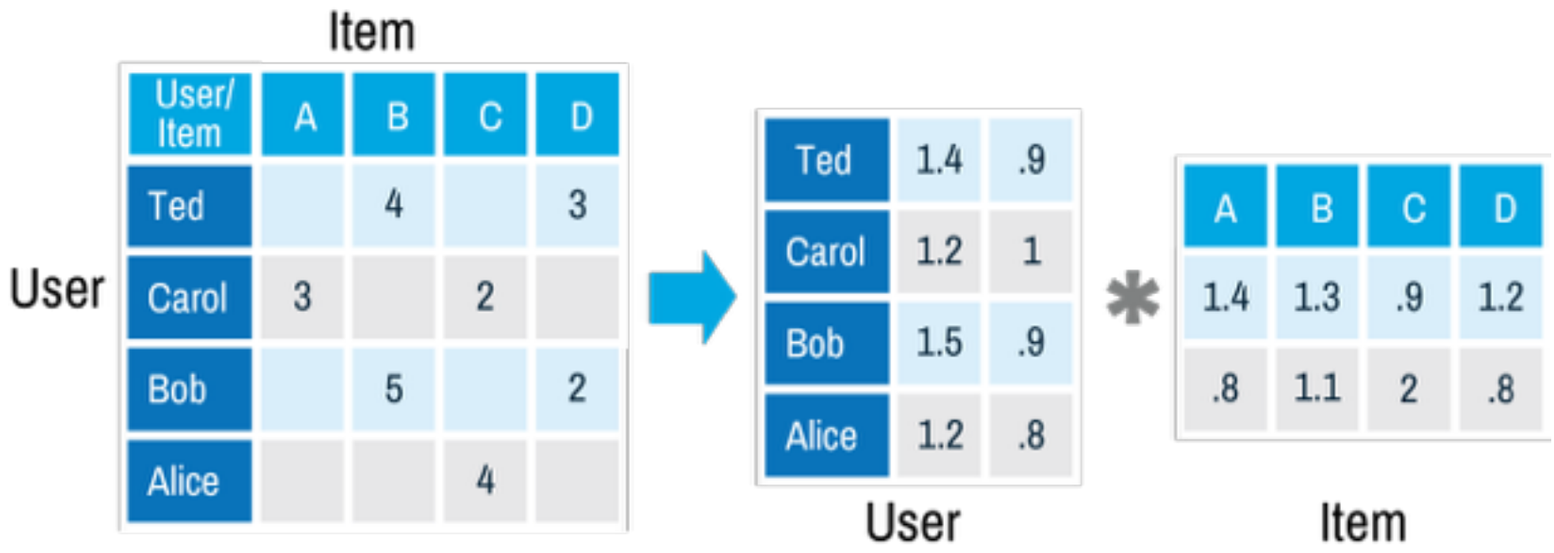


Fig. 2. FMs succeed in estimating 2-way variable interactions in very sparse problems where SVMs fail (see section III-A3 and IV-B for details.)

FM vs 矩阵分解协同过滤

review：矩阵分解CF

- 将评分矩阵分解为两个低秩矩阵的乘积，隐变量解释为兴趣点
- 和FM都用了低秩矩阵分解的方法学习缺失模式



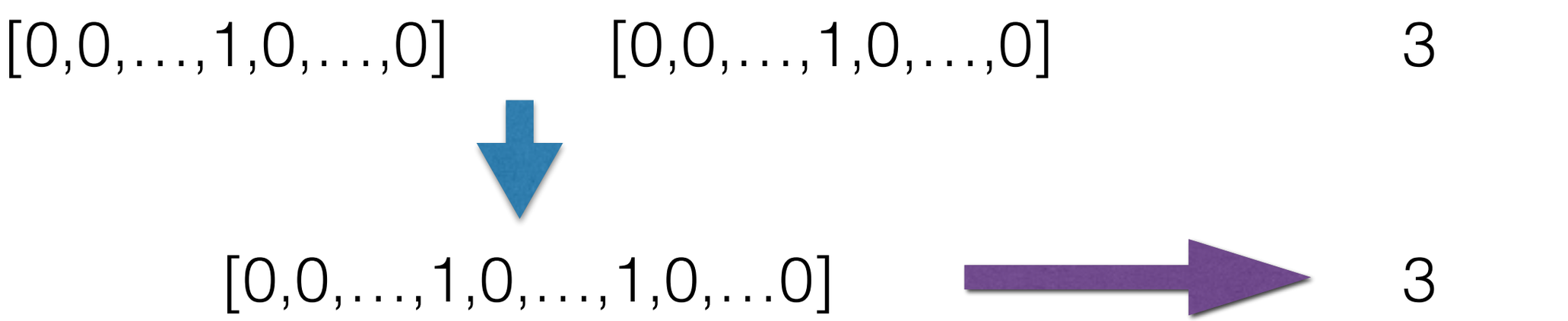
$$\min_{p,q,b} \sum_{(u,i)\in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$

$$s.t. \quad \hat{r}_{ui} = q_i^T p_u + \mu + b_i + b_u$$

矩阵分解CF是FM的一个特例

- 样本：(userID, itemID)，label：score
- 特征向量 f(userID, itemID) = 用户ID onehot ++ 商品ID one hot

用户123对商品45的评分为3分



矩阵分解CF是FM的一个特例

- 优势：FM能够将更多的特征融入这个框架
- 用户画像应用场景：**优化用户poi偏好建模！**

FM

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

协同过滤

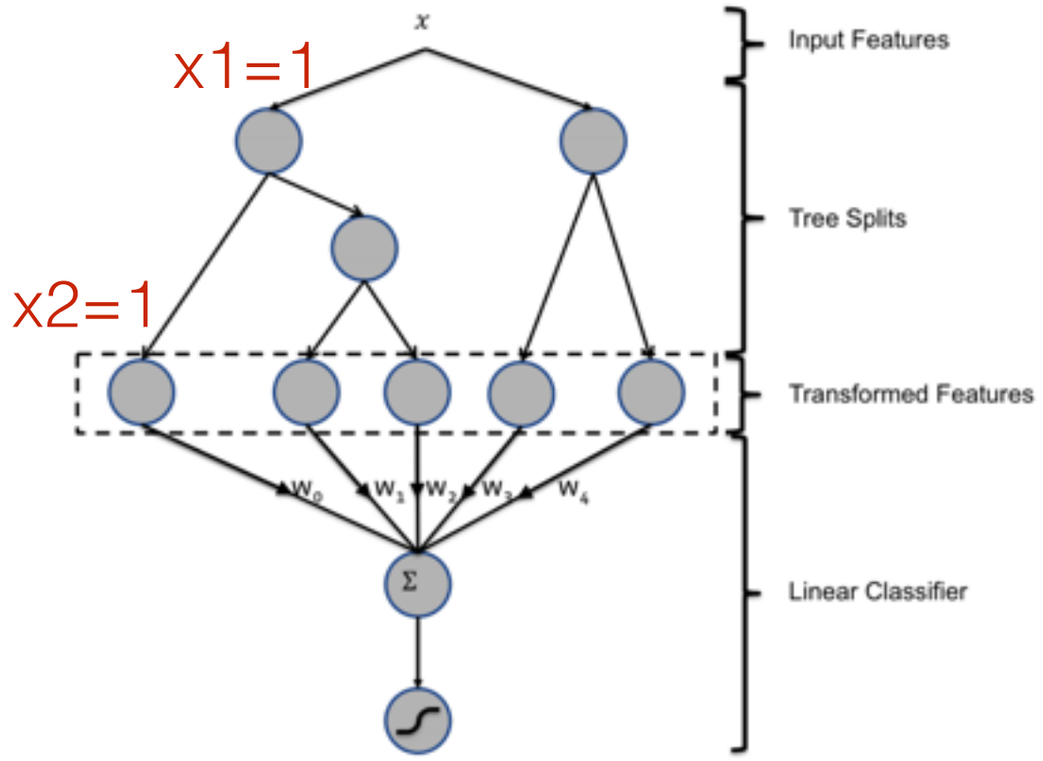
$$\min_{p,q,b} \sum_{(u,i) \in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$

$$s.t. \quad \hat{r}_{ui} = q_i^T p_u + \mu + b_i + b_u$$

FM vs 树模型

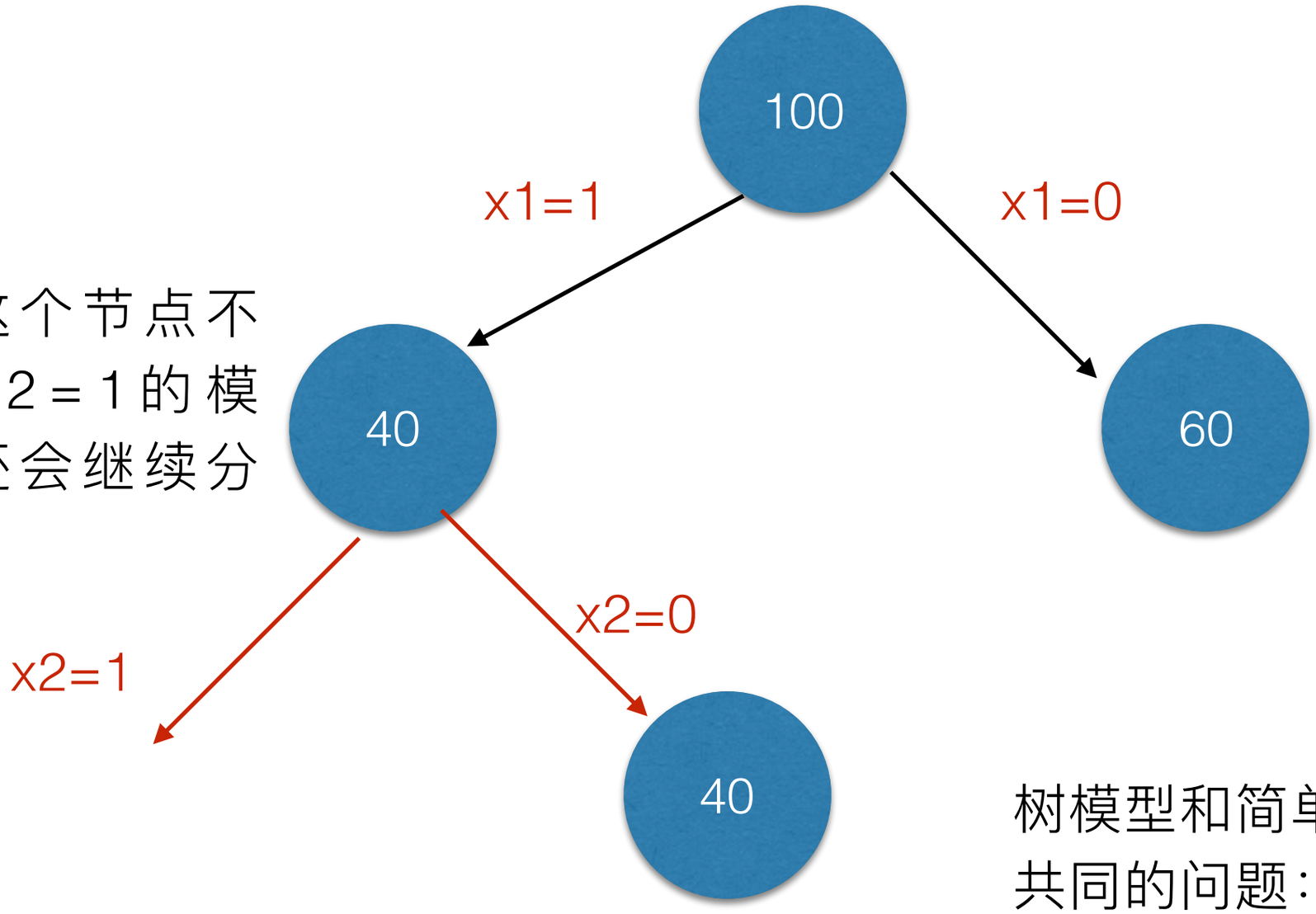
FM vs 树模型

- 都能自动学习特征交叉组合
- 树模型适合连续中低度稀疏数据，容易学到高阶组合
- FM适合高度稀疏数据
- **树模型为什么不适合学习高度稀疏数据的特征组合?**



树模型高度稀疏数据

如果这个节点不存在x2=1的模式，还会继续分裂么?



树模型和简单二阶多项式模型存在共同的问题：不能学到数据中不存在的特征模式！但是FM可以！这是FM在高度稀疏数据建模优势的关键原因！

行业案例：世纪佳缘

杨鹏谈世纪佳缘推荐算法：基于Spark GraphX，弃GBDT和LR用FM - CS...
www.csdn.net › 云计算 ▾
2015年9月30日 - 9月29日20:30-21:30，世纪佳缘算法工程师杨鹏在CSDN人工智能用户群分享了“世纪佳缘推荐和机器学习算法实践”。他主要介绍了基于图算法产生 ...
缺少字词： machine learning

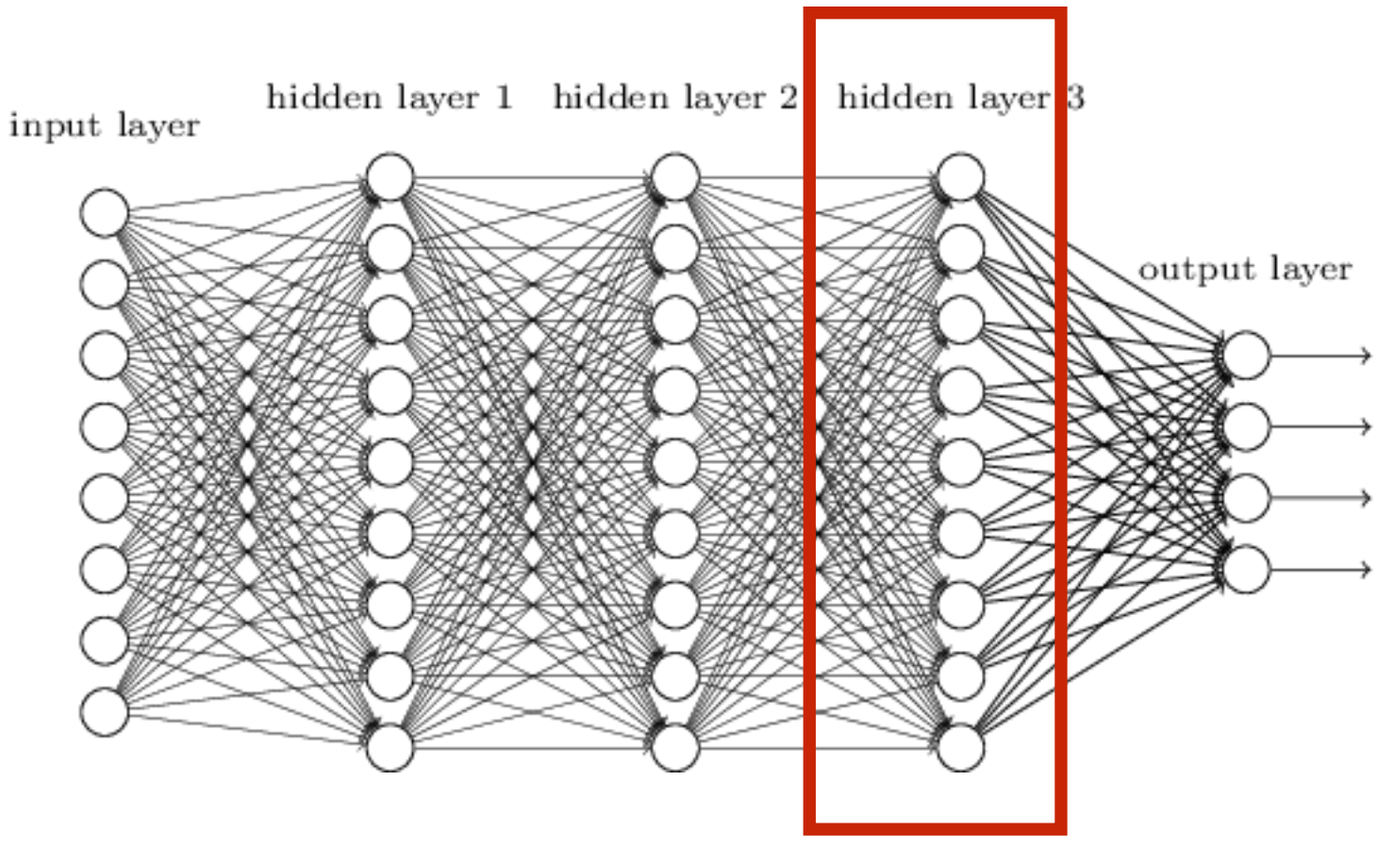
杨鹏谈世纪佳缘推荐算法：基于Spark GraphX，弃GBDT和LR用FM
www.chinacloud.cn/show.aspx?id=21902&cid=6 ▾
2015年9月30日 - 以下为杨鹏分享实录： 大家好，我叫杨鹏，来自世纪佳缘算法组，主要关注于推荐和机器学习方面。我今天分享一下世纪佳缘在推荐方面的尝试和心得 ...
缺少字词： machine learning

FM vs 神经网络

神经网络

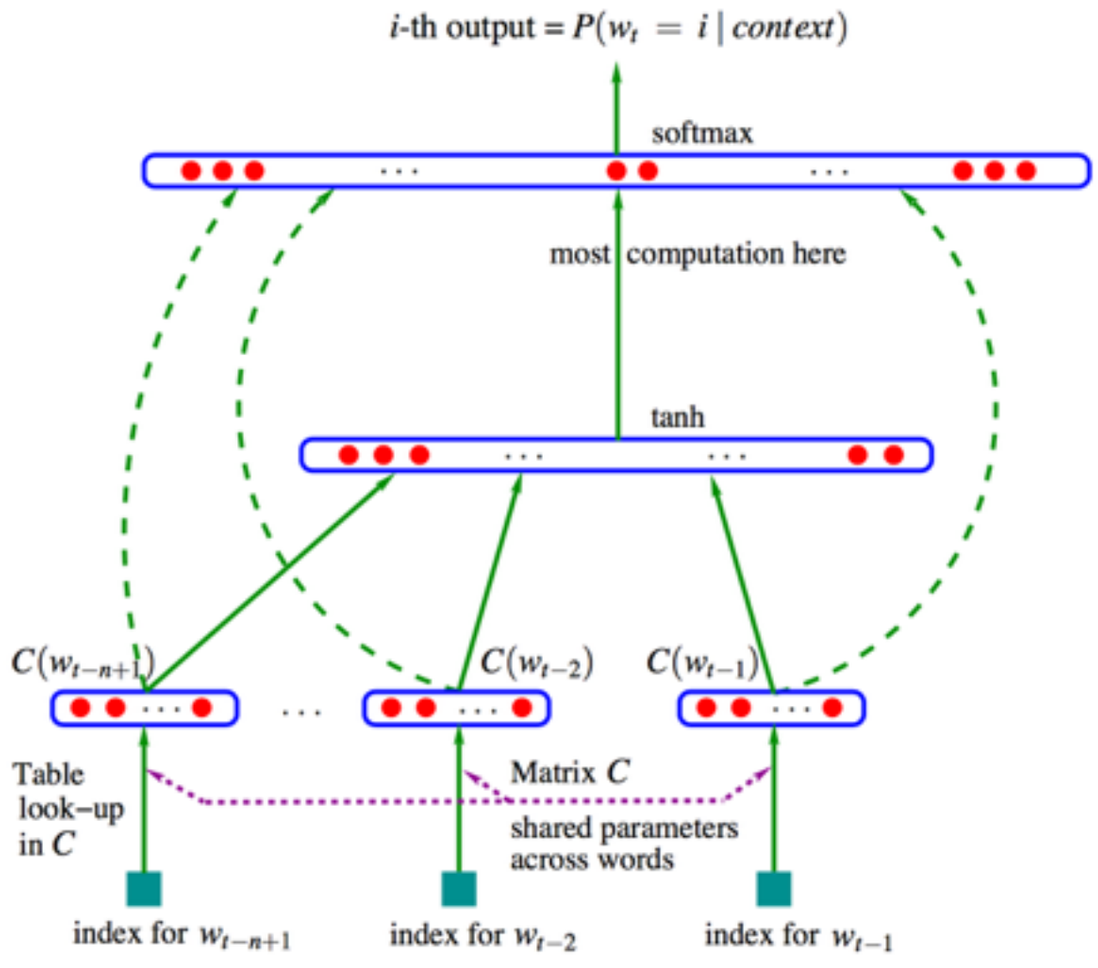
- 神经网络最接近目标的那一层可以看做高级的非线性交叉特征，Hinton把它称作bottleneck特征，又有称作dark knowledge
- 问题：神经网络适合连续低维特征输入，对高维稀疏的离散特征输入难以处理

dark knowledge



低维嵌入embedding

- 神经网络处理高维稀疏的离散特征的一种常用手段，例如：词向量!
- 这个低维嵌入的向量是和模型联合优化得到的
- **FM可以看做一种 embedding 的方法**
- word2vec词出现在上下文的概率跟两个词向量内积正相关；FM用户和商品的偏好与用户向量和商品向量内积正相关
- FM 和 embedding 都是通过将高维稀疏的离散特征映射为低维向量来增强其特征的泛化能力的



deep & wide 模型

- 将离散特征做低维嵌入，和连续特征一起放到神经网络里面学习高度非线性的特征交叉
- 将这样的神经网络和线性模型联合训练优化

$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$$

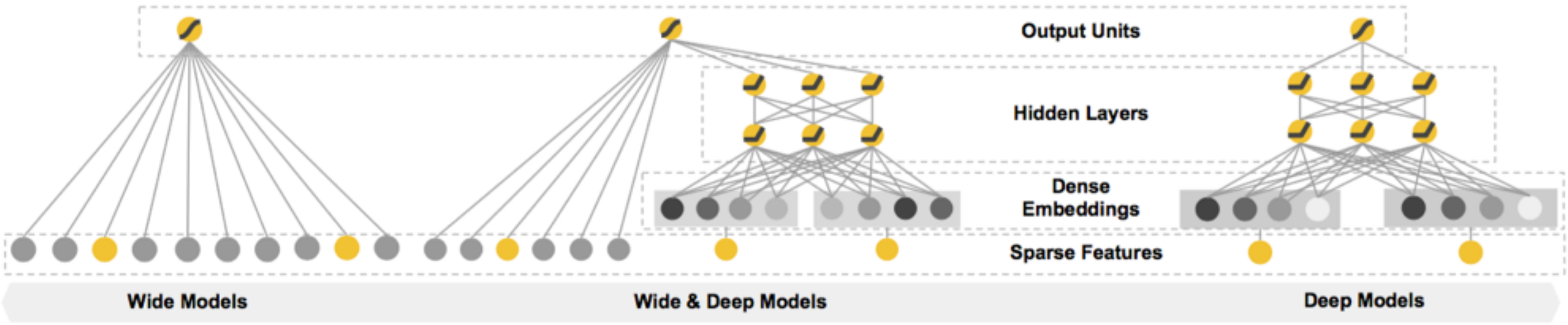


Figure 1: The spectrum of Wide & Deep models.

Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016: 7-10.

deep & wide 模型

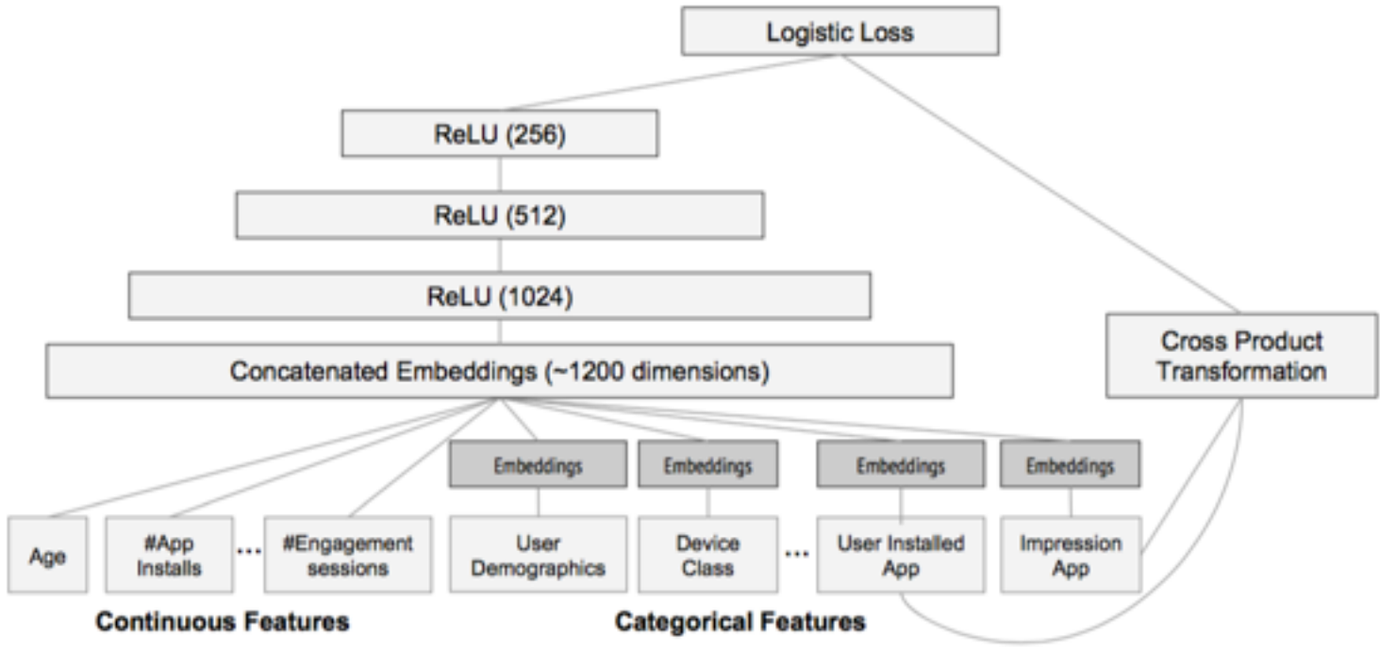


Figure 4: Wide & Deep model structure for apps recommendation.

Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	+3.9%

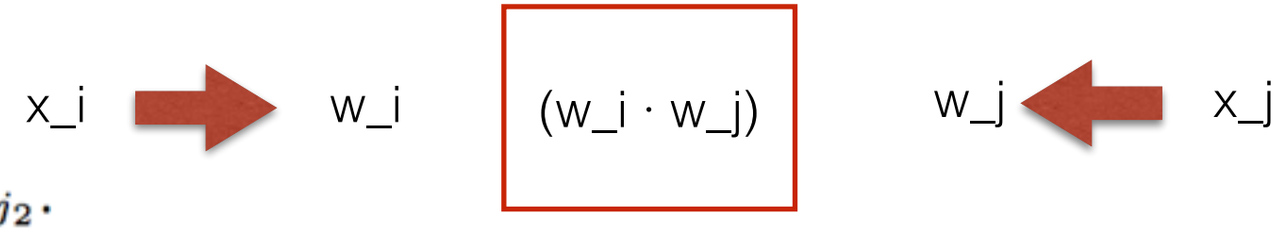
离线结果和在线结果有偏移，如何理解?

Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016: 7-10.

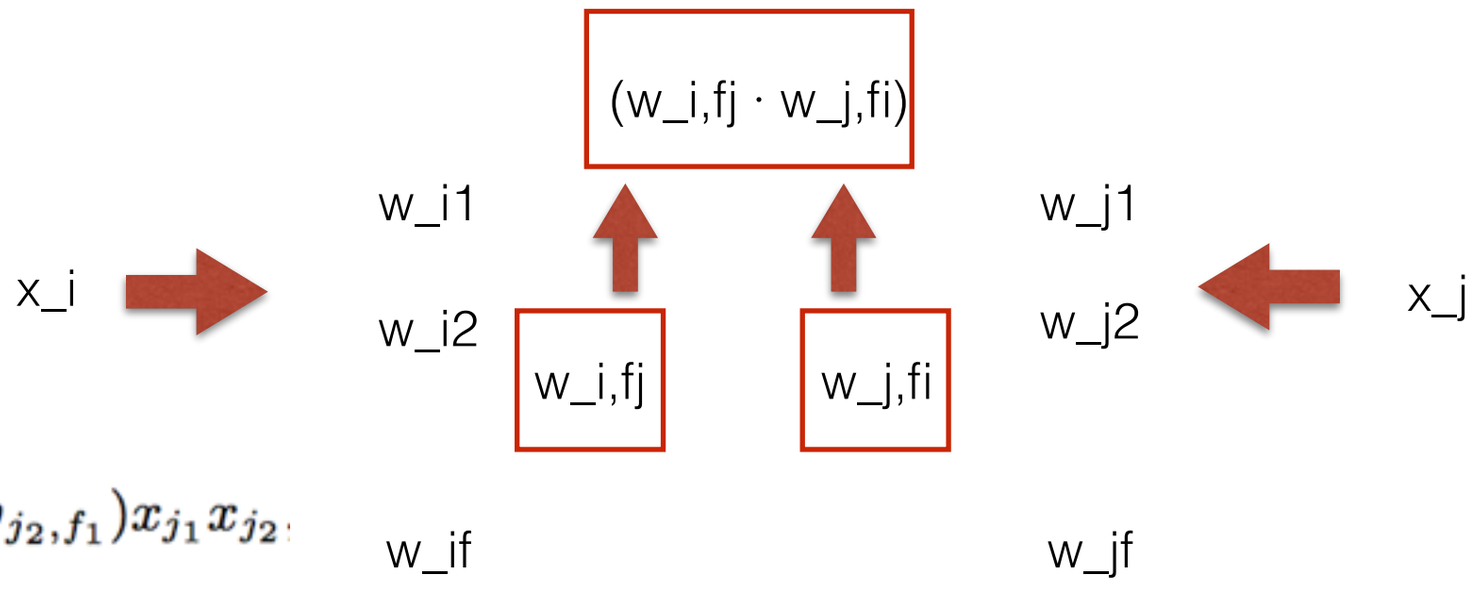
FFM

- 将一个特征映射为多个向量，每个向量对应一个场(FieId)，两个特征交叉时取对应场的向量做内积
- FM可以看做只有一个场的FFM
- 场的意义就是将特征归类，不同类交叉用的向量不同

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



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



FFM

- 场的意义就是将特征归类，不同类交叉用的向量不同

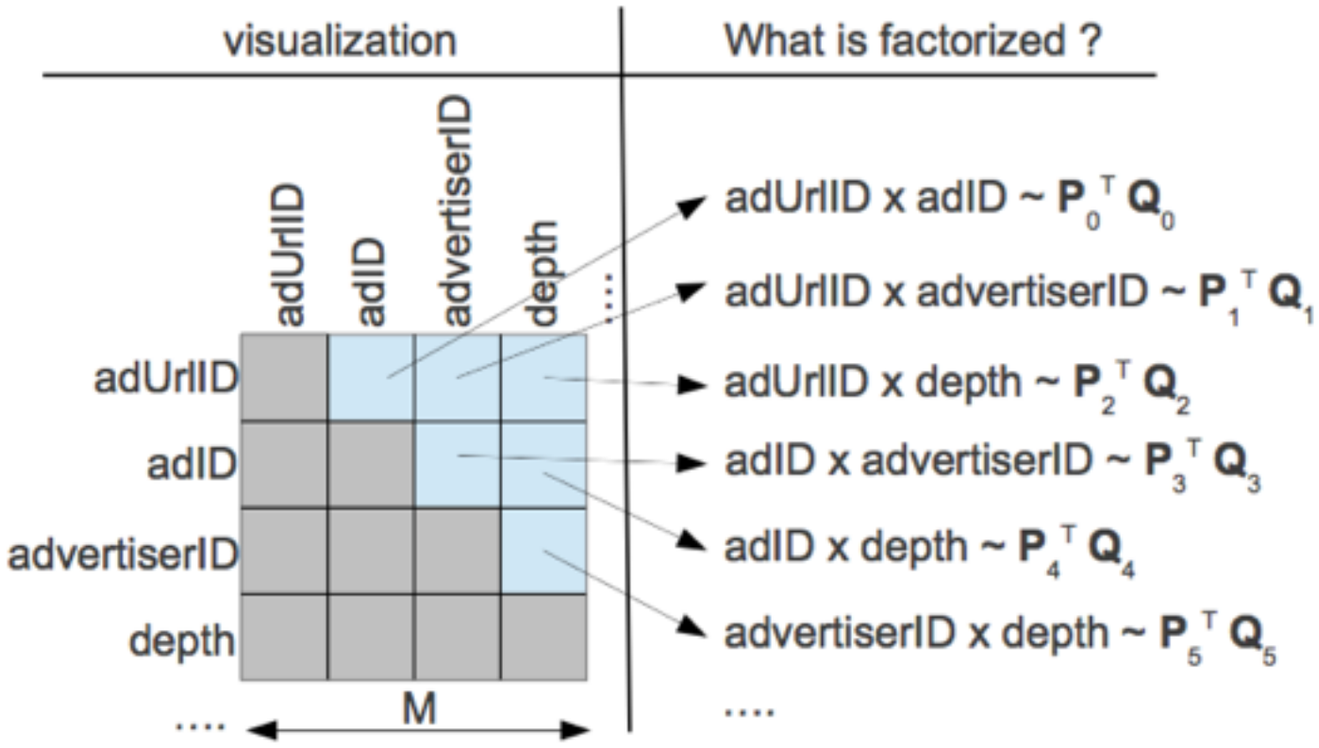


Figure 2: Factors in the factor model. Every cell had its own factor matrices.

M. Jahrer, A. To¨scher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered model for click through rate prediction," in KDD Cup 2012 Workshop, ACM, 2012.

FFM计算性能

- 预测无法通过数学技巧变为线性复杂度，而是O(k·n0^2), n0表示非0元素个数
- 通常 k_ffm << k_fm, n0数目在高度稀疏数据集中一般在几十的样子，k_ffm一般不超过10
- 优化方法：随机梯度
- 软件：单机版libFFM；我自己实现了 ffm 自适应随机梯度的Spark版本，DMSPA可用

$$\boldsymbol{g}_{j_1,f_2} \equiv \nabla_{\boldsymbol{w}_{j_1,f_2}} f(\boldsymbol{w}) = \lambda \cdot \boldsymbol{w}_{j_1,f_2} + \kappa \cdot \boldsymbol{w}_{j_2,f_1} x_{j_1} x_{j_2}, \quad (5)$$

$$\boldsymbol{g}_{j_2,f_1} \equiv \nabla_{\boldsymbol{w}_{j_2,f_1}} f(\boldsymbol{w}) = \lambda \cdot \boldsymbol{w}_{j_2,f_1} + \kappa \cdot \boldsymbol{w}_{j_1,f_2} x_{j_1} x_{j_2}, \quad (6)$$

where

$$\kappa = \frac{\partial \log(1 + \exp(-y\phi_{\mathrm{FFM}}(\boldsymbol{w}, \boldsymbol{x})))}{\partial \phi_{\mathrm{FFM}}(\boldsymbol{w}, \boldsymbol{x})} = \frac{-y}{1 + \exp(y\phi_{\mathrm{FFM}}(\boldsymbol{w}, \boldsymbol{x}))}.$$

Juan Y, Zhuang Y, Chin W, et al. Field-aware Factorization Machines for CTR Prediction[C]. conference on recommender systems, 2016: 43-50.

案例 KDD CUP 2012

- KDD Cup 2012 CTR预
- 234M个样本
- 评估指标 AUC

- **adUrlID** - This is a property of the ad. The URL was shown together with the title and description of an ad. It was usually the shortened landing page URL of the ad, but not always.
- **adID** - The unique ID of an advertisement.
- **advertiserID** - The unique advertiser ID of the **adID**. It is a property of the ad. Some advertisers might produce more attractive advertisements than others.
- **depth** - The number of ads displayed to the user in a query session. The maximum depth is 3.
- **position** - The position of the **adID** in a query session. The maximum position is 3.
- **queryID** - The unique ID of the query.
- **keywordID** - A number representing a keyword used in the ad.
- **titleID** - A number representing the ad title.
- **descriptionID** - A number representing a description of the ad.
- **userID** - The unique ID of a user who conducts the query.

Table 4: Models on raw upsampled data.

MODEL	Ω_{V1}	Ω_{V2}	Ω_{Tst}
BIAS MODEL	0.82719	0.82754	0.788
FACTOR MODEL	0.82989	0.829394	0.7913
AFM MODEL	0.7673	0.7698	0.740
BAYESIAN MODEL	0.8430	0.8460	0.752
GBM	0.7860	0.7836	0.757
SVM^{perf}	0.7961	0.7924	0.764
ANN	0.8012	0.8251	0.765

The bias model benefited greatly from an ID-specific learning rate and L2-regularization values. We found the values in Table 3 by coordinate search. The optimized bias model had leaderboard $AUC = 0.78784$.

M. Jahrer, A. To¨scher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered model for click through rate prediction," in KDD Cup 2012 Workshop, ACM, 2012.

案例

- 两个CTR预估的数据集：Criteo，Avazu （都是Kaggle 上的比赛）

Data Set	# instances	# features	# fields
Criteo	45,840,617	10^7	39
Avazu	40,428,967	10^7	33

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**

Table 3: Comparison among models and implementations on data sets **Criteo** and **Avazu**. The training sets used here are **CriteoTrVa** and **AvazuTrVa**, and the test sets used here are **CriteoTe** and **AvazuTe**. For all experiments, a single thread is used. The public set is around 20% of the test data, while the private set contains the rest. For Criteo, we do not list the result of Poly2-LIBLINEAR-Hash-Newton, because the experiment does not finish after more than 10 days. Note that the we use different stopping conditions for different algorithms, so the training time is only for reference.

Juan Y, Zhuang Y, Chin W, et al. Field-aware Factorization Machines for CTR Prediction[C]. conference on recommender systems, 2016: 43-50.

高阶FM

- d-way FM：只有d阶项，依然可以通过平方分解，在线性时间复杂度计算

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right)$$

- High order FM：不仅仅只有d阶项，还有2,3...,d-1阶项，不能简单地利用平方分解的技巧实现线性时间复杂度

$$\hat{y}_{\mathrm{HOFM}}(\boldsymbol{x}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{j'>j} \langle \bar{\boldsymbol{p}}_j^{(2)}, \bar{\boldsymbol{p}}_{j'}^{(2)} \rangle x_j x_{j'} + \cdots + \sum_{j_m > \cdots > j_1} \langle \bar{\boldsymbol{p}}_{j_1}^{(m)}, \ldots, \bar{\boldsymbol{p}}_{j_m}^{(m)} \rangle x_{j_1} x_{j_2} \ldots x_{j_m}$$

where we defined $\langle \bar{\boldsymbol{p}}_{j_1}^{(t)}, \ldots, \bar{\boldsymbol{p}}_{j_t}^{(t)} \rangle := \mathrm{sum}(\bar{\boldsymbol{p}}_{j_1}^{(t)} \circ \cdots \circ \bar{\boldsymbol{p}}_{j_t}^{(t)})$ (sum of element-wise products).

- Rendle S. Factorization Machines[C]. international conference on data mining, 2010.
- Blondel M, Fujino A, Ueda N, et al. Higher-Order Factorization Machines[C]. neural information processing systems, 2016: 3351-3359.

高阶FM

- 利用动态规划将高阶FM计算复杂度也变为线性时间复杂度

$$\hat{y}_{\mathrm{HOFM}}(\boldsymbol{x}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{j'>j} \langle \bar{\boldsymbol{p}}_j^{(2)}, \bar{\boldsymbol{p}}_{j'}^{(2)} \rangle x_j x_{j'} + \cdots + \sum_{j_m > \cdots > j_1} \langle \bar{\boldsymbol{p}}_{j_1}^{(m)}, \ldots, \bar{\boldsymbol{p}}_{j_m}^{(m)} \rangle x_{j_1} x_{j_2} \ldots x_{j_m}$$

where we defined $\langle \bar{\boldsymbol{p}}_{j_1}^{(t)}, \ldots, \bar{\boldsymbol{p}}_{j_t}^{(t)} \rangle := \mathrm{sum}(\bar{\boldsymbol{p}}_{j_1}^{(t)} \circ \cdots \circ \bar{\boldsymbol{p}}_{j_t}^{(t)})$ (sum of element-wise products).

ANOVA kernel

$$\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) := \sum_{j_m > \cdots > j_1} \prod_{t=1}^{m} p_{j_t} x_{j_t}. \tag{5}$$

For later convenience, we also define $\mathcal{A}^0(\boldsymbol{p}, \boldsymbol{x}) := 1$ and $\mathcal{A}^1(\boldsymbol{p}, \boldsymbol{x}) := \langle \boldsymbol{p}, \boldsymbol{x} \rangle$. Then it is shown that

$$\hat{y}_{\mathrm{HOFM}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{s=1}^{k_2} \mathcal{A}^2\left(\boldsymbol{p}_s^{(2)}, \boldsymbol{x}\right) + \cdots + \sum_{s=1}^{k_m} \mathcal{A}^m\left(\boldsymbol{p}_s^{(m)}, \boldsymbol{x}\right), \tag{6}$$

ANOVA kernel 递推关系 $\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) = \mathcal{A}^m(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j}) + p_j x_j \, \mathcal{A}^{m-1}(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j}),$

- Blondel M, Fujino A, Ueda N, et al. Higher-Order Factorization Machines[C]. neural information processing systems, 2016: 3351-3359.