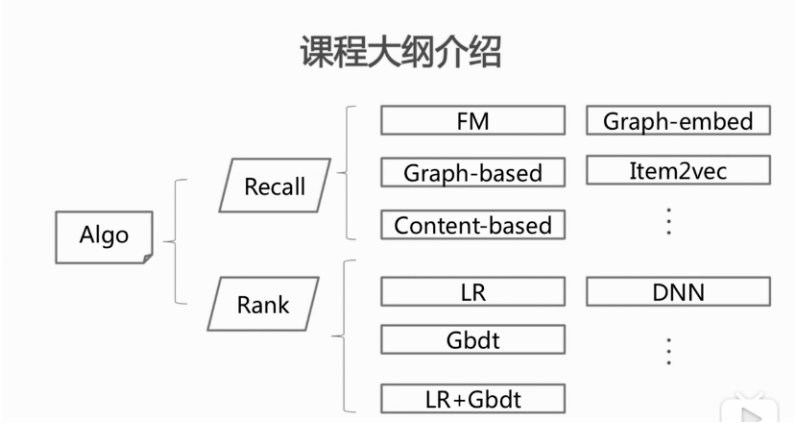
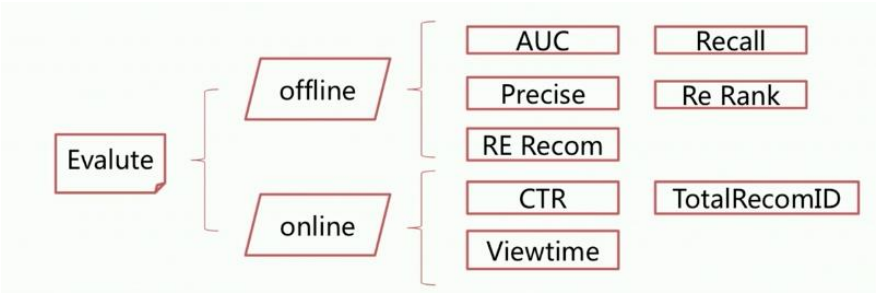
个性化推荐算法实战

1 推荐算法综述

1.1 个性化推荐算法综述





环境：python TensorFlow word2vec xgboost

高数微积分、线性代数、概率论相关知识

机器学习基本知识，数据挖掘大体了解

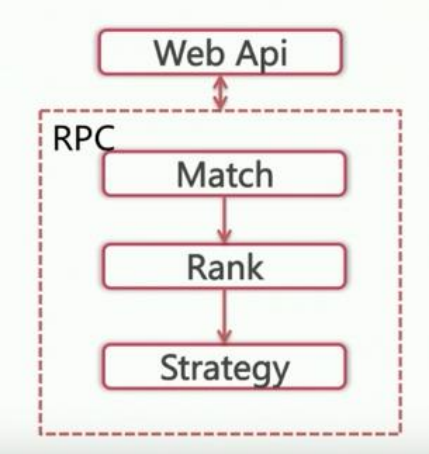
1.2 个性化召回算法综述

1.2.1 什么是个性化召回

根据用户的属性行为上下文等信息从物品全集中选取其感兴趣的物品作为候选集

1.2.2 召回在推荐系统中的重要作用

召回决定了最终推荐结果的天花板

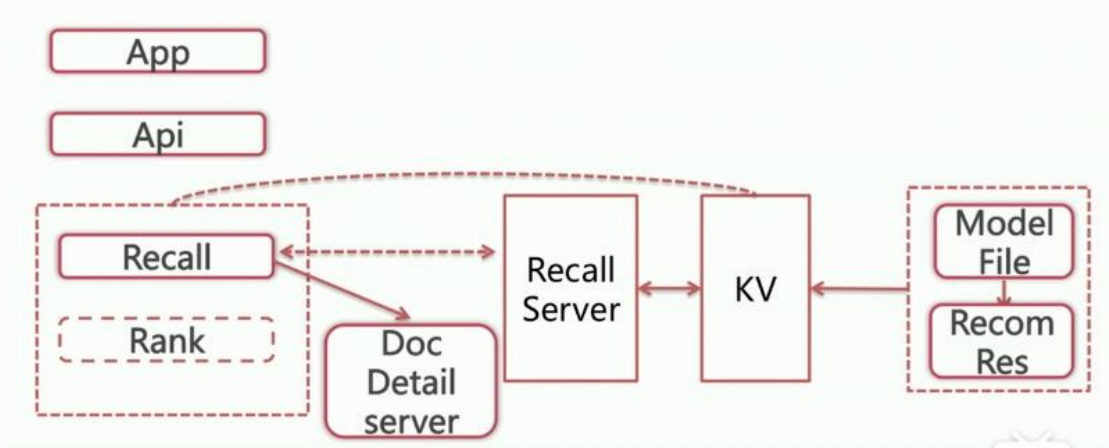


1.2.3 工业界个性化召回解析

分类

- 基于用户行为的
- 基于 user profile 的
- 基于隐语义的

工业界个性化召回架构



2 个性化推荐召回算法

2.1 LFM 算法综述

2.1.1 个性化召回算法(latent factor model)综述

LFM 算法的背景

https://blog.csdn.net/bbbeoy/article/details/78646576

应用于计算用户的 toplike、计算 item 的 topsim、计算 item 的 topic

2.1.2 LFM 理论知识与公式推导

LFM 建模公式：

$$\mathrm{p(u,i)} = p_u^T q_i = \sum_{f=1}^{F} p_{uf} q_{if}$$

LFM loss function

$$\mathrm{loss} = \sum_{(u,i)\in D} (p(u,i) - p^{LFM}(u,i))^2$$

$$\mathrm{loss} = \sum_{(u,i)\in D} (p(u,i) - \sum_{f=1}^{F} p_{uf} q_{if})^2 + \alpha_1 |p_u|^2 + \alpha_2 |q_i|^2$$

LFM 算法迭代

对上式求偏导：

$$\frac{\partial loss}{\partial p_{uf}} = -2\big(p(u,i) - p^{LFM}(u,i)\big)q_{if} + 2\alpha_1 p_{uf}$$

$$\frac{\partial loss}{\partial q_{if}} = -2\big(p(u,i) - p^{LFM}(u,i)\big)p_{uf} + 2\alpha_2 q_{if}$$

使用梯度下降算法

$$p_{uf} = p_{uf} - \beta \frac{\partial loss}{\partial p_{uf}}$$

$$q_{if} = q_{if} - \beta \frac{\partial loss}{\partial q_{if}}$$

参数设定影响效果

- 负样本选取
- 隐特征 F（10~32）、正则参数、learning rate

2.1.3 LFM 算法和 CF 算法的优缺点比较

理论基础

离线计算空间时间复杂度

在线推荐与推荐解释

3 personal rank 算法

3.1 personal rank 算法的背景和物理意义

3.1.1 背景

用户行为很容易表示为图

图推荐在个性化推荐领域效果显著

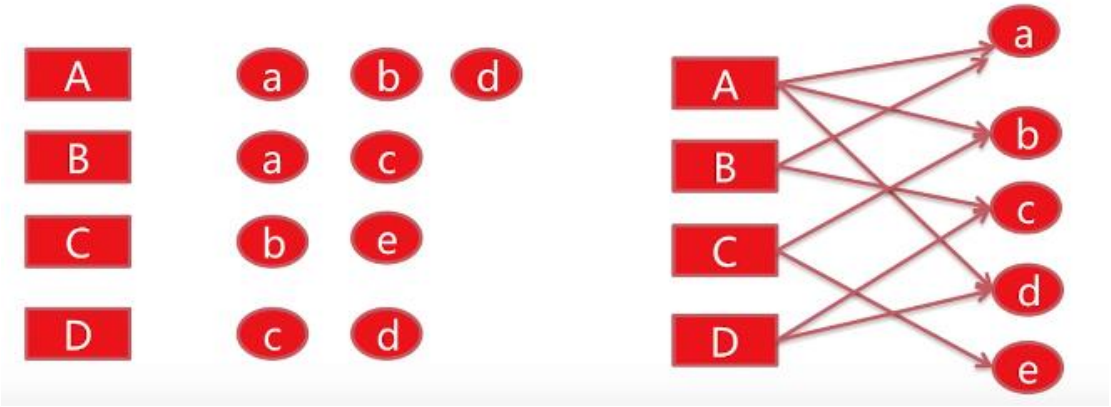
3.1.2 二分图

二分图又称为二部图，是图论中的一种特殊模型。设 G=(V,E)是一个无向图，如果顶点 V 可分割为两个互不相交的子集(A,B)，并且图中的每条边(i,j)所关联的两个顶点 i 和 j 分别属于这两个不同的顶点集(i

in A, j in B)，则称图 G 为一个二分图。

Example：

对 userA 来说，item_c 和 item_e 哪个更值得推荐？



3.1.3 物理意义

两个顶点之间连通路径数

两个顶点之间连通路径长度

两个顶点之间连通路径经过顶点的出度

Example:

分别由几条路径连通？

连通路径的长度分别是多少？

连通路径经过的顶点的出度分别是多少？

3.2 personal rank 算法的数学公式推导

3.2.1 算法抽象-文字阐述

对用户 A 进行个性化推荐，从用户 A 结点开始在用户物品二分图

random walk，以 alpha 的概率从 A 的出边中等概率选择一条游走过去，到达该顶点后（举例顶点 a），有 alpha 的概率继续从顶点 a 的出边中等概率选择一条继续游走到下一个结点，或者（1-alpha）的概率回到起点 A，多次迭代。直到各顶点对于用户 A 的重要度收敛。

3.2.2 算法抽象-数学公式

$$\mathrm{PR(v)} = \begin{cases} \alpha * \sum_{\tilde{v}\in v} \frac{PR(\tilde{v})}{|out(\tilde{v})|} \cdots (v! = v_A) \\ (1-\alpha) + \alpha * \sum_{\tilde{v}\in v} \frac{PR(\tilde{v})}{|out(\tilde{v})|} \cdots (v = v_A) \end{cases}$$

3.2.3 算法抽象-矩阵式

$$\mathrm{r} = (1-\alpha) r_0 + \alpha M^T r \qquad M_{ij} = \frac{1}{|out(i)|} j \in out(i) else 0$$

$$(\mathrm{E} - \alpha M^T) * \mathrm{r} = (1-\alpha) r_0$$

$$\mathrm{r} = (\mathrm{E} - \alpha M^T)(1-\alpha) r_0$$

4 item2vec 算法

4.1 个性化召回算法 Item2vec 背景与物理意义

4.1.1 背景

Item2item 的推荐方式效果显著

NN model 的特征抽象能力

算法论文：ITEM2VEC: NEURAL ITEM EMBEDDING FOR COLLABORATIVE FILTERING

https://blog.csdn.net/qq_35771020/article/details/89137392

4.1.2 物理意义

将用户的行为序列转化成 item 组成的句子

模仿 word2vec 训练 word embedding 将 item embedding 进行训练

4.1.3 缺陷

用户的行为序列时序性缺失

用户行为序列中的 item 强度是无区分性的

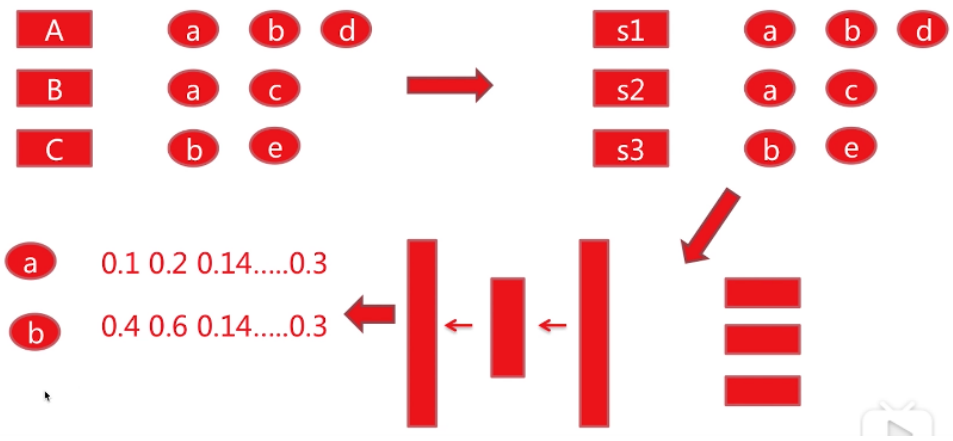
4.2 Item2vec 算法应用主流程

从 log 中抽取用户行为序列

将行为序列当成语料训练 word2vec 得到 item embedding

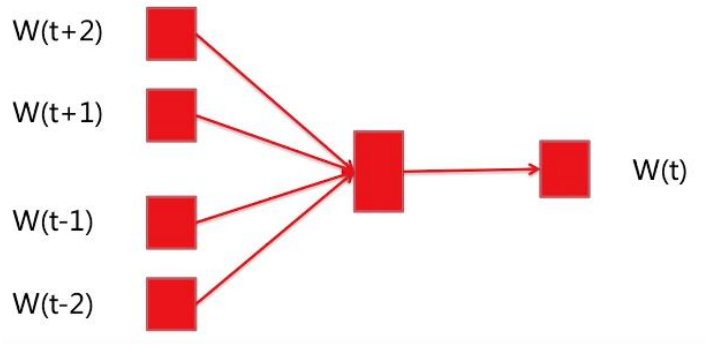
得到 item sim 关系用于推荐

Example:

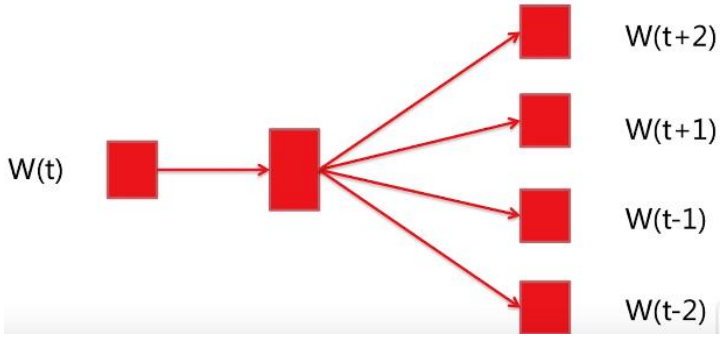


4.3 Item2vec 算法依赖 model word2vec 介绍

4.3.1 CBOW(continuous bag of words)



4.3.2 Skip Gram



4.3.3 CBOW Word2vec 数学原理

问题抽象：

$$\mathrm{g(w)} = \prod_{u \in w \cup NEG(w)} p\big(u \big| Context(w)\big)$$

$$\mathrm{p\big(u\big|Context(w)\big)} = \sigma(X_w^T \theta^u)^{L^w(u)} (1 - \sigma(X_w^T \theta^u))^{(1-L^w(u))}$$

Loss Function

$$\mathrm{Loss} = \log(g(w))$$

$$\mathrm{Loss} = \sum \big(L^w(u) * \log\big(\sigma(X_w^T \theta^u)\big) + \big(1 - L^w(u)\big) * \log\big(1 - \sigma(X_w^T \theta^u)\big)\big)$$

梯度：

$$\frac{\partial Loss}{\partial \theta^u} = \left(L^w(u) - \delta(x_w^T \theta^u)\right) x_w \qquad \theta^u = \theta^u + \alpha * \frac{\partial Loss}{\partial \theta^u}$$

$$\frac{\partial Loss}{\partial x_w} = \left(L^w(u) - \delta(x_w^T \theta^u)\right) \theta^u$$

$$\mathrm{v}(w_{context}) = \mathrm{v}(w_{context}) + \sum_{u \in w \cup NEG(w)} \alpha * \frac{\partial Loss}{\partial x_w}$$

CBOW 训练主流程

- 选取中心词 w 以及负采样 NEG(w)
- 分别获得损失函数对 x_w 与 $theta^u$ 的梯度
- 更新 $theta^u$ 以及中心词对应的 context(w)的每一个词的词向量

4.3.4 Skip Gram Word2vec 数学原理

问题抽象

$$\mathrm{G} = \prod_{u \in Context(w)} \prod_{z \in u \cup NEG(u)} p(z|w)$$

$$\mathrm{p(z|w)} = (\delta(\mathrm{v(w)}^T \theta^z))^{L^u(z)} * (1 - \delta(\mathrm{v(w)}^T \theta^z))^{1 - L^u(z)}$$

Loss Function

$$\mathrm{Loss} = \sum_{u \in Context(w)} \sum_{z \in u \cup NEG(u)} L^u(z) * \log(\delta(v(w)^T \theta^z)) + \left(1 - L^u(z)\right) * \log(1 - \delta(v(w)^T \theta^z))$$

$$\mathrm{G} = \prod_{w^c \in context(w)} \prod_{u \in w \cup NEG(w)} p(u|w^c)$$

$$\mathrm{Loss} = \sum_{w^c \in Context(w)} \sum_{u \in w \cup NEG(w)} L^w(u) * \log(\delta(v(w^c)^T \theta^u)) + \left(1 - L^w(u)\right) * \log(1 - \delta(v(w^c)^T \theta^u))$$

Skip Gram word2vec 训练主流程

- 对于 context(w)中任何一个词 w^c 选取 w 的正负样本
- 计算 Loss 对 theta 以及对 w^c 的偏导
- 更新 w^c 对应的词向量

负采样算法

$$\text{len(word)} = \frac{(counter(word))^{\alpha}}{\sum_{w \in D}(counter(word))^{\alpha}}$$

5 content based 算法

5.1 个性化召回算法 Content Based 背景介绍

思路极简，可解释强

用户推荐的独立性

问世较早，流行度高

5.2 Content Based 算法的主体流程介绍

5.2.1 Item Profile

Topic Finding

Genre Classify

5.2.2 User Profile

Genre/Topic

Time Decay

5.2.3 Online Recommendation

Find top K Genre/Topic

Get the best n item from fixed Genre/Topic

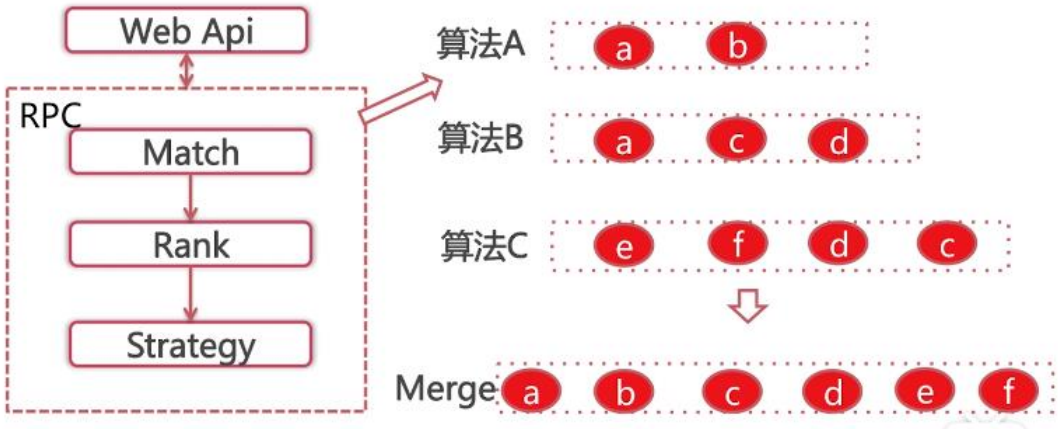
6 个性化召回算法总结与评估方法的介绍

6.1 个性化召回算法总结

基于领域的

基于内容的

基于 neural network 的



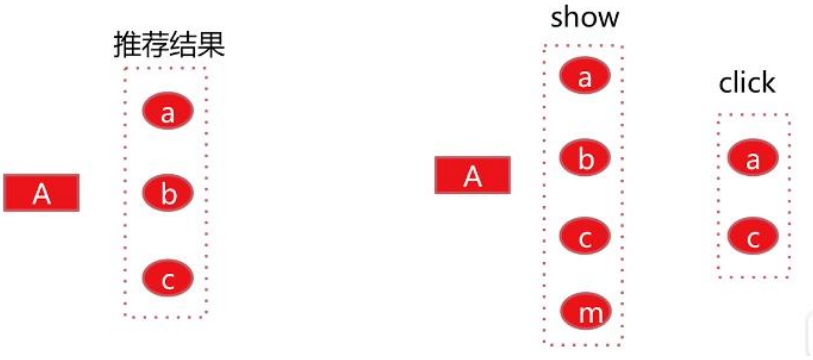
6.2 个性化召回算法评估

离线评价准入

在线评价收益

6.2.1 Offline 评价方法

评测新增算法推荐结果在测试集上的表现



6.2.2 Online 评价方法

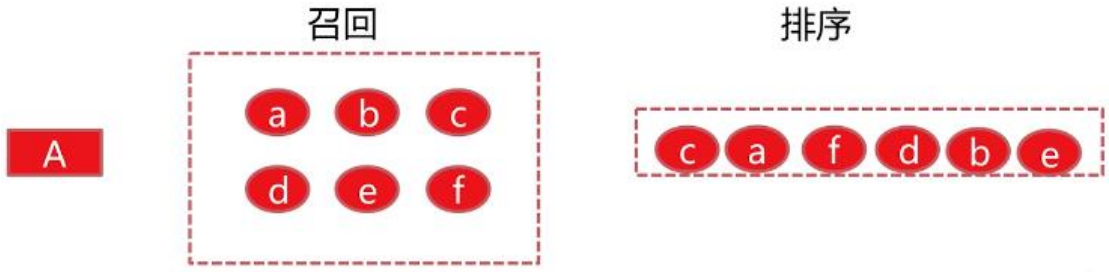
定义指标

生产环境 A/B test

7 排序综述

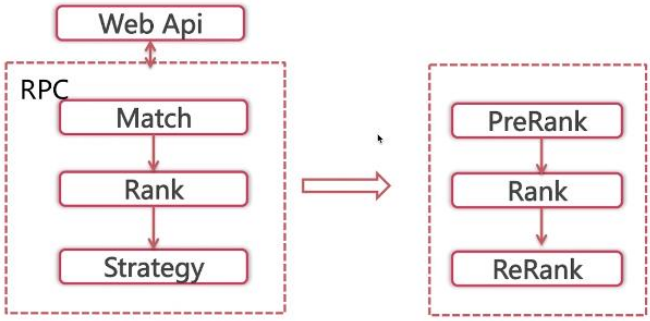
7.1 什么是 learn to Rank?

将个性化召回的物品候选集根据物品本身的属性结合用户的属性，上下文等信息给出展现优先级的过程



7.2 排序在个性化推荐系统中的重要作用

Rank 决定了最终的推荐效果

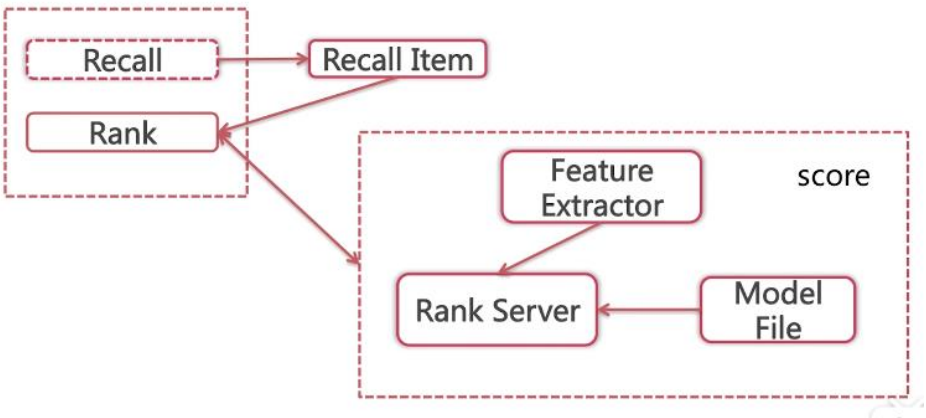


7.3 工业界推荐系统中排序架构解析

单一的渐层模型

浅层模型的组合

深度学习模型



8 逻辑回归模型

8.1 逻辑回归(logistic regression, LR)模型的背景知识介绍

8.1.1 点击率预估与分类模型

https://www.cnblogs.com/qcloud1001/p/7513982.html

https://blog.csdn.net/starzhou/article/details/51769561

8.1.2 什么是 LR

https://blog.csdn.net/weixin_39445556/article/details/83930186

8.1.3 Sigmoid 函数

https://www.jianshu.com/p/d4301dc529d9

https://blog.csdn.net/weixin_39445556/article/details/83930186

Example

LR 模型训练流程

- 从 Log 中获取训练样本与特征
- Model 参数学习
- Model 预测

8.1.4 LR Model 优点与缺点

优点：易于理解，计算代价小

缺点：容易欠拟合，需要特征工程

8.2 逻辑回归模型的数学原理

8.2.1 阶跃函数及其导数

$$\mathrm{f(x)} = \frac{1}{1+\exp(-x)}$$

$$\mathrm{f'(x)} = \frac{\exp(-x)}{(1+\exp(-x))^2}$$

$$\mathrm{f'(x)} = \frac{1}{1+\exp(-x)} * \frac{1+\exp(-x)-1}{1+\exp(-x)}$$

8.2.2 LR Model Function

Model function

$$\mathrm{w} = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n$$

$$\mathrm{y} = \mathrm{sigmoid(w)}$$

8.2.3 Loss Function

$$\mathrm{loss} = \log \prod_{i=1}^{n} p(y_i|x_i)$$

$$\mathrm{p}(y_i|x_i) = h_w(x_i)^{y_i}(1-h_w(x_i))^{1-y_i}$$

$$\mathrm{loss} = -\big(y_i \log h_w(x_i) + (1-y_i)\log\big(1-h_w(x_i)\big)\big)$$

8.2.4 梯度

$$\frac{\partial loss}{\partial w_j} = \frac{\partial loss}{\partial h_w(x_i)} \frac{\partial h_w(x_i)}{\partial w} \frac{\partial w}{\partial w_j}$$

$$\frac{\partial loss}{\partial h_w(x_i)} = -\left(\frac{y_i}{h_w(x_i)} + \frac{y_i - 1}{1-h_w(x_i)}\right)$$

$$\frac{\partial h_w(x_i)}{\partial w}\frac{\partial w}{\partial w_j} = h_w(x_i)\big(1 - h_w(x_i)\big)x_i^j$$

梯度下降

$$\frac{\partial loss}{\partial w_j} = (h_w(x_i) - y_i)x_i^j$$

$$w_j = w_j - \alpha\frac{\partial loss}{\partial w_j}$$

8.2.5 正则化

什么是过拟合

https://blog.csdn.net/qq_18254385/article/details/78428887

L1 正则化与 L2 正则化

https://blog.csdn.net/zhaomengszu/article/details/81537197

L1：

$$\text{loss}_{\text{new}} = loss + \alpha\sum_{i=1}^{n}|w_i|$$

L2:

$$\text{loss}_{\text{new}} = \text{loss} + \alpha|w|^2$$

8.3 样本选择与特征选择相关知识

8.3.1 Corpus

样本选择规则

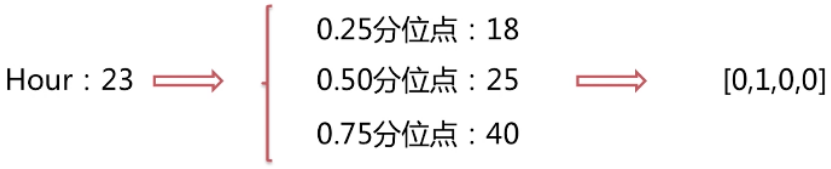
样本过滤规则

8.3.2 Feature

特征的统计与分析

特征的选择

特征的预处理



9 决策树算法

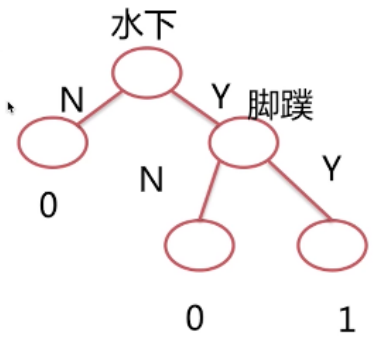
9.1 决策树背景知识介绍

9.1.1 什么是决策树

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。

https://www.cnblogs.com/yonghao/p/5061873.html

水下生活	有脚蹼	是鱼类
1	1	1
1	1	1
1	0	0
0	1	0
0	1	0



9.1.2 决策树构造原理

回归树的函数表示

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

$$\sum_{x_i \in R_m} (y_i - f(x_i))^2$$

$$c_m = ave(y_i | x_i \in R_m)$$

最优特征选取

$$\min_{j,s}[\min_{c_1} \sum_{x_1 \in R_1} (y_i - C_1)^2 + \min_{c_2} \sum_{x_1 \in R_2} (y_i - C_2)^2]$$

$$R_1 = \{x | x^j \leq s\}, R_2 = \{x | x^j > s\}$$

$$c_1 = ave(y_i | x_i \in R_1), c_2 = ave(y_i | x_i \in R_2)$$

构建树的流程

- 遍历所有特征，特征的最佳划分对应的得分，选取最小得分的特征
- 将数据依据此选取的特征划分分成两部分
- 继续在左右两部分遍历变量找到划分特征直到满足停止条件

CART 生成

https://www.jianshu.com/p/b90a9ce05b28

回归树：平方误差最小化原则

分类树：基尼指数

基尼指数

$$\text{Gini(D)} = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

$$D_1 = \{(x, y) \in D | A(x) \geq a\}, D_2 = D - D_1$$

$$\text{Gini(D, A)} = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

水下生活	有脚蹼	是鱼类	
1	1	1	G(D,水下)=3/5*4/9+2/5*0=12/45
1	1	1	
1	0	0	G(D,脚蹼)=4/5*1/2+1/5*0=4/10
0	1	0	
0	1	0	

9.2 梯度提升树的数学原理与构建方法

9.2.1 什么是 boosting

一种用来提高弱分类算法准确度的方法,这种方法通过构造一个预测函数系列,然后以一定的方式将他们组合成一个预测函数

如何改变训练数据的权重

如何组合多个基础 model

9.2.2 Boosting Tree 模型函数

$$f_M(\text{x}) = \sum_{m=1}^{M} T(x; \theta_m)$$

$$f_m(\text{x}) = f_{m-1}(x) + T(x; \theta_m)$$

$$\theta_m = arg\min_{\theta_m} \sum_{i=1}^{N} L\left(y_i, f_{m-1}(x_i) + T(x_i, \theta_m)\right)$$

迭代损失函数

$$\mathrm{L}\left(\mathrm{y}, f(\mathrm{x})\right) = \left(y - f(x)\right)^2$$

$$\mathrm{L}\left(\mathrm{y}, f_m(\mathrm{x})\right) = [y - f_{m-1}(x) - T(x;\theta_m)]^2$$

9.2.3 提升树的算法流程

- 初始化$f_0(x) = 0$
- 对 m=1,2,3,……计算残差 r_m，拟合 rm，得到 T_m
- 更新$f_m = f_{m-1} + T_m$

Example

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

$S=1, R_1=\{1\}, R_2=\{2,3...10\}, c_1=5.56, c_2=7.50, m(s)=0+15.72=15.72$

s	1	2	3	4	5	6	7	8	9
m(s)	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

$T_1(x)=6.24$ x<=6; $T_1(x)=8.91$ x>6 $f_1(x)=T_1(x)$

x	1	2	3	4	5	6	7	8	9	10
y	-0.68	-0.54	-0.33	0.16	0.56	0.81	-0.01	-0.21	0.09	0.14

$T_2(x)=-0.52$ x<=3; $T_2(x)=0.22$ x>3 $f_2(x)=f_1(x)+T_2(x)$

9.2.4 梯度提升树

残差的数值改变

$$r_m = -[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$$

9.3 XGBoost 数学原理与构建方法

9.3.1 XGBoost 模型函数

$$f_M(x) = \sum_{m=1}^{M} T(x; \theta_m)$$

$$f_m(x) = f_{m-1}(x) + T(x; \theta_m)$$

$$\arg\min_{\theta_m} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(x_i) + T(x_i; \theta_m)\big) + \Omega(T_m)$$

9.3.2 优化目标的泰勒展开

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$$

$$\min_{\theta_m} \sum_{i=1}^{N} [g_i T_m + 0.5 * h_i T_m^2] + \Omega(T_m)$$

$$g_i = \frac{\partial L(y_i, f_{m-1})}{\partial f_{m-1}}, h_i = \frac{\partial^2 L(y_i, f_{m-1})}{\partial f_{m-1}}$$

9.3.3 定义模型复杂度

$$f(x) = \sum_{j=1}^{Q} c_j I\left(x \in R_j\right)$$

$$\Omega(T_m) = \partial \mathrm{Q} + 0.5\beta \sum_{j=1}^{Q} c_j^2$$

9.3.4 目标转化

$$\min_{\theta_m} \sum_{i=1}^{N} [g_i T_m + 0.5 * h_i T_m^2] + \Omega(T_m)$$

$$\min_{\theta_m} \sum_{i=1}^{N} [g_i T_m + 0.5 * h_i T_m^2] + \partial Q + 0.5\beta \sum_{j=1}^{Q} c_j^2$$

$$\min_{\theta_m} \sum_{j=1}^{Q} \left[\left(\sum_{i \in R_j} g_i \right) c_j + 0.5 \left(\sum\nolimits_{i \in R_j} h_i + \beta \right) c_j^2 \right] + \partial Q$$

9.3.5 目标函数最优解

$$G_j = \sum\nolimits_{i \in R_j} g_i, H_j = \sum\nolimits_{i \in R_j} h_i$$

$$\min_{\theta_m} \sum_{j=1}^{Q} [G_j c_j + 0.5(H_j + \beta) c_j^2] + \partial Q$$

$$c_j = -\frac{G_j}{H_j + \beta}, obj = -0.5 \sum_{i=1}^{Q} \frac{G_j^2}{H_j + \beta} + \partial Q$$

9.3.6 最佳划分特征选取

$$c_j = -\frac{G_j}{H_j + \beta}, obj = -0.5\sum_{i=1}^{Q}\frac{G_j^2}{H_j + \beta} + \partial Q$$

$$G_{ain} = \left(\frac{G_L^2}{H_L + \beta} + \frac{G_R^2}{H_R + \beta} - \frac{(G_R + G_L)^2}{H_R + H_L + \beta}\right) - \partial$$

9.3.7 XGBoost 总流程

- 初始化$f_0(x) = 0$
- 对 m=1,2,3,……M 应用选择最优划分特征的方法构造树
- 更新$f_m = f_{m-1} + \text{learning_rate} + T_m$

9.4 gbdt 与 lr 混合模型网络介绍

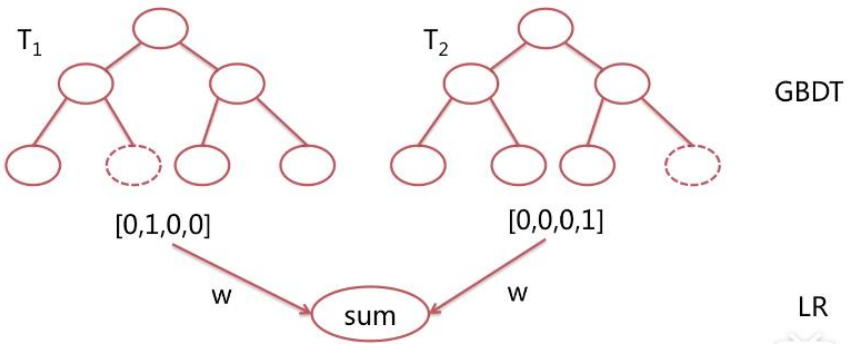
9.4.1 背景知识

Practical Lessons from Predicting Clicks on Ads at Facebook

逻辑回归需要繁琐的特征处理

树模型的 feature transform 能力

9.4.2 模型网络



9.4.3 优缺点总结

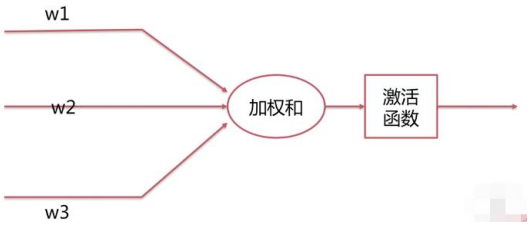
优点：利用树模型做特征转化

缺点：两个模型单独训练不是联合训练

10 深度学习

10.1 深度学习背景介绍

10.1.1 神经元



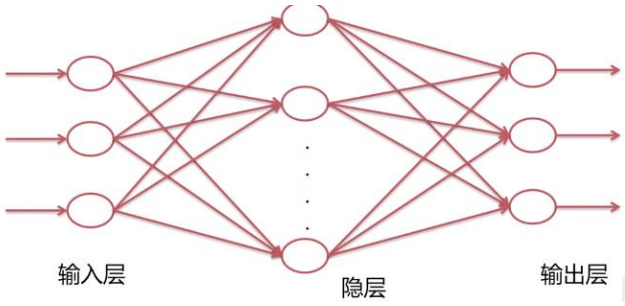
10.1.2 激活函数

sigmoid

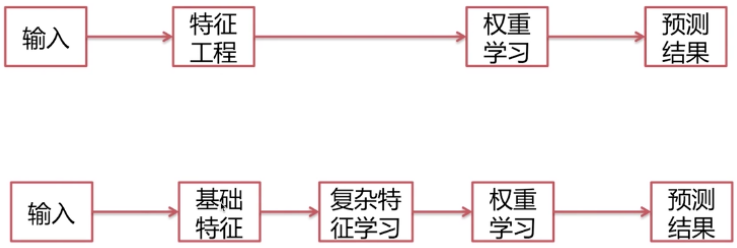
tanh

relu

10.1.3 神经网络

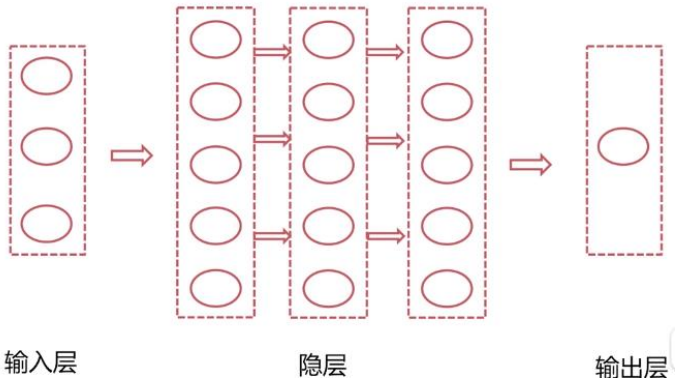


10.1.4 DL & ML difference



10.2 DNN 网络结构与数学原理

10.2.1 DNN 网络结构



10.2.2 DNN 模型参数

隐层层数，每个隐层神经元个数，激活函数

输入输出层向量维度

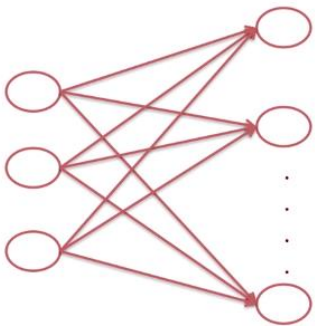
不同层之间神经元的连接权重 W 和偏移值 B

10.2.3 前向传播

节点的输出值

$$a_j^t = f(\sum_k w_{jk}^t \, a_k^{t-1} + b_j^t)$$

$$z_j^t = \sum_k w_{jk}^t \, a_k^{t-1} + b_j^t$$



10.2.3 反向传播

Our Target

$$\frac{\partial L}{\partial w_{jk}^t} \qquad \frac{\partial L}{\partial b_j^t}$$

What we have

$$\frac{\partial L}{\partial a_j^T} \qquad \frac{\partial L}{\partial z_j^T}$$

推导

$$\frac{\partial L}{\partial b_j^t} = \frac{\partial L}{\partial z_j^t} * \frac{\partial z_j^t}{\partial b_j^t} = \frac{\partial L}{\partial z_j^t} \qquad z_j^t = \sum_k w_{jk}^t \, a_k^{t-1} + b_j^t$$

$$\frac{\partial L}{\partial w_{jk}^t} = \frac{\partial L}{\partial z_j^t} * \frac{\partial z_j^t}{\partial w_{jk}^t} = \frac{\partial L}{\partial z_j^t} * a_k^{t-1}$$

核心

$$\frac{\partial L}{\partial z_j^{t-1}} = \sum_k \frac{\partial L}{\partial z_k^t} \frac{\partial z_k^t}{\partial z_j^{t-1}}$$

$$z_j^t = \sum_k w_{jk}^t \, a_k^{t-1} + b_j^t$$

$$\frac{\partial z_k^t}{\partial z_j^{t-1}} = \frac{\partial z_k^t}{\partial a_j^{t-1}} \frac{\partial a_j^{t-1}}{\partial z_j^{t-1}} = w_{jk}^t \frac{\partial a_j^{t-1}}{\partial z_j^{t-1}}$$

- 对输入 x，设置合理的输入向量
- 前向传播逐层逐个神经元求解加权和与激活值
- 对于输出层求解输出层损失函数对于 z 值的偏导
- 反向传播逐层求解损失函数对 z 值的偏导
- 得到 w 与 b 的梯度

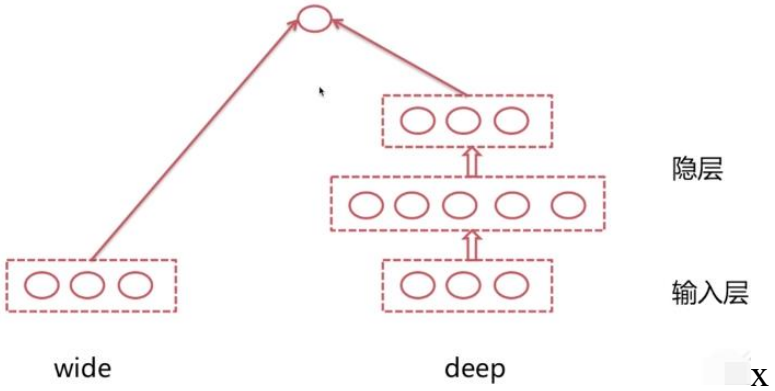
10.3 WD(wide and deep)网络结构与数学原理

10.3.1 w&d 的物理意义

论文：wide & deep learning for Recommender Systems

Generalization and Memorization

10.3.2 w&d 的网络结构



模型输出

$$a_{out}^{T} = h(w_{wide}, w_{deep}) = \sigma(w_{wide}[x, x_{cross}] + w_{deep}a_{out}^{T-1} + b^{T})$$

10.3.3 WD model 的反向传播

Wide 侧参数学习

$$\frac{\partial L}{\partial w_{widej}} = \frac{\partial L}{\partial a^T}\frac{\partial a^T}{\partial z^T}\frac{\partial z^T}{\partial w_{widej}} = \frac{\partial L}{\partial a^T}\sigma'(z^T)x_{widej}$$

Deep 侧参数学习

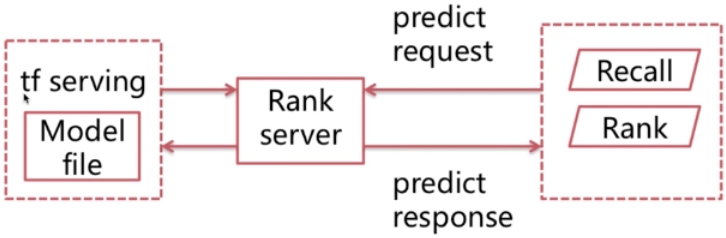
$$\frac{\partial L}{\partial z_j^{t-1}} = \sum_k \frac{\partial L}{\partial z_k^t}\frac{\partial z_k^t}{\partial a_j^{t-1}}\frac{\partial a_j^{t-1}}{\partial z_j^{t-1}} = \sum_k \frac{\partial L}{\partial z_k^t} w_{deepkj}^t \frac{\partial a_j^{t-1}}{\partial z_j^{t-1}}$$

$$z_k^t = \sum_j w_{deepkj}^t\, a_j^{t-1} + b_k^t \rightarrow t \neq T$$

$$z_k^t = \left(\sum_j w_{deepkj}^t\, a_j^{t-1} + b_k^t\right) + w_{wide} * X \rightarrow t = T$$

$$\frac{\partial L}{\partial b_j^t} = \frac{\partial L}{\partial z_j^t} * \frac{\partial z_j^t}{\partial b_j^t} = \frac{\partial L}{\partial z_j^t} \quad \frac{\partial L}{\partial w_{jk}^t} = \frac{\partial L}{\partial z_j^t} * \frac{\partial z_j^t}{\partial w_{jk}^t} = \frac{\partial L}{\partial z_j^t} * a_k^{t-1}$$

10.3.4 server 架构



11 学习排序部分总结

11.1 效果回顾

model	model cv auc	test data auc
LR	0.89908	0.89734
GBDT	0.91179	0.91114
LR+GBDT	0.91605	0.91254
WD	-	0.90888

大规模数据集上表现 WD>LR+GBDT>GBDT>LR

11.2 离线评估

Model cv

Model test and data performance

11.3 在线评估

业务指标

平均点击位置

11.4 特征维度浅析

特征维度

User 、Item、Context、UI Relation、Statics Supplement

特征数目

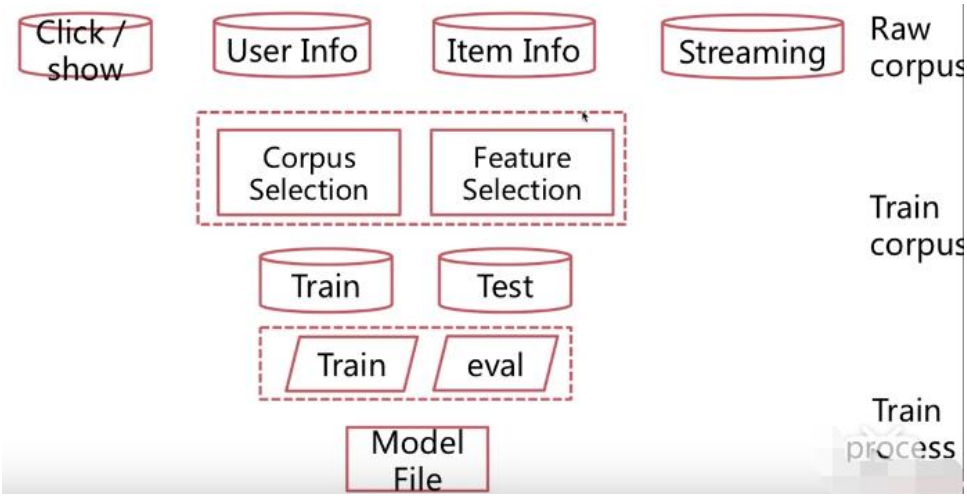
11.5 Rank 技术展望

多目标学习

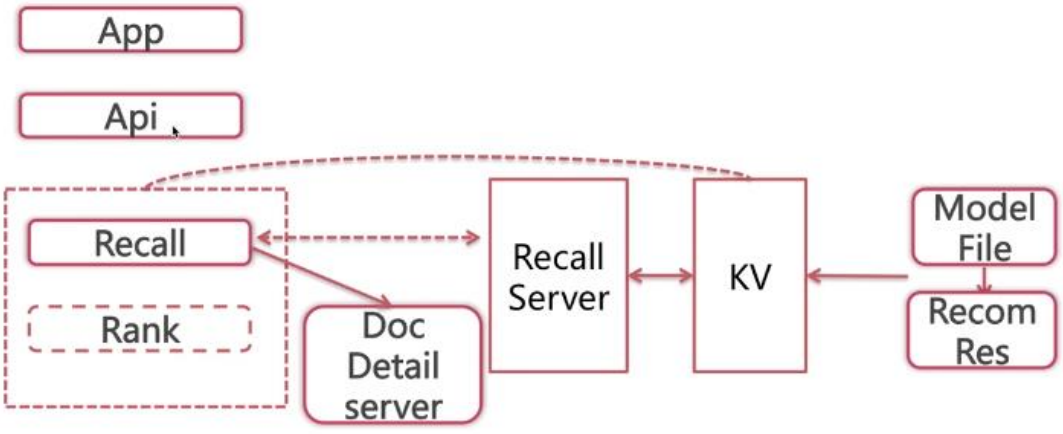
强化学习

12 总结

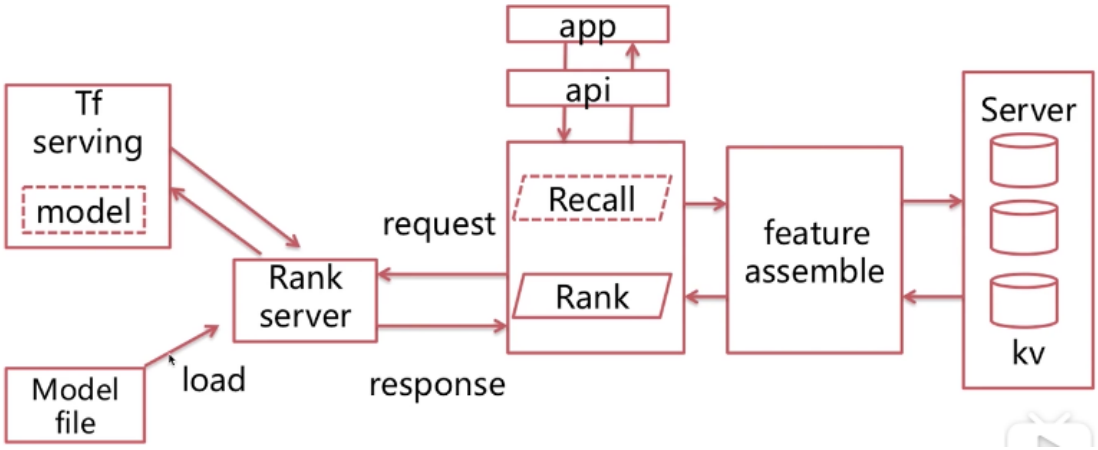
12.1 个性化推荐算法的离线架构



12.2 个性化推荐算法的在 Recall 线架构



12.3 个性化推荐算法的 Rank 在线架构



12.4 个性化推荐算法的回顾

Recall：CF, LFM, Personal Rank, Item2vec, ContentBased

Rank: LR, GBDT, LR + GBDT, WD