5.6 推荐缓存服务

- 目的：对待推荐结果进行二级缓存，多级缓存减少数据库读取压力
- 1、获取redis结果，进行判断
  - 如果redis有，读取需要推荐的文章数量放回，并删除这些文章，并且放入推荐历史推荐结果中
  - 如果redis当中不存在，则从wait_recommend中读取
    - 如果wait_recommend中也没有，直接返回
    - 如果wait_recommend有，从wait_recommend取出所有结果，定一个数量(如100篇)存入redis,剩下放回wait_recommend,不够100，全部放入redis，然后清空wait_recommend
    - 从redis中拿出要推荐的文章结果，然后放入历史推荐结果中

'reco:{}:{}art'

5.7 排序模型在线预测

5.7.1排序模型服务

- 1、读取用户特征中心特征

```
ctr_feature_user, '{}', 'channel:{}'
```

- 2、读取文章特征中心特征、合并用户文章特征构造预测样本

```
ctr_feature_article, '{}', 'article:{}'
```

  - 两部分特征通过 3.8.1 特征服务中心读取 提高构建样本预测的速度
- 3、加载训练好的历史模型，进行排序

6.1.1 深度学习到推荐系统

6.1.2 推荐系统为什么加入深度学习?

推荐系统的应用需求

- 能够直接从内容中提取特征，表征能力强
- 容易对噪声数据进行处理，抗噪能量强

6.2 深度学习应用简介

1、特征提取方面

- 机器学习的**特征工程步骤是要靠手动完成的，而且需要大量领域专业知识**
- **深度学习：通过大量数据的训练自动得到模型，不需要人工设计特征提取环节**

2、 数据量

3、算法代表

- 机器学习
  - 朴素贝叶斯、决策树、SVM、逻辑回归、聚类算法
- 神经网络系列

6.2.2 深度学习主要应用与技术

6.2.3 常见深度学习框架对比

- 不需要手写CUDA 代码能跑GPU；
- 容易构建大的计算图（computational graphs）

1.2.2 TensorFlow的特点

1.2.3 TensorFlow的安装

2.1 TF数据流图

2.1.1 案例：TensorFlow实现一个加法运算

2.1.1.2 TensorFlow结构分析

- 分成两个阶段
  - TensorFlow构建模型图阶段
  - TensorFlow主动执行图阶段
- 基础概念:
  - 图：代表程序运算过程，会话：执行程序一个入口
  - 其它：张量(TF当中最基础数据类型)，Tensor类

2.1.2 数据流图介绍

- 其它概念
  - 节点（Operation）在图中表示数学操作
  - 数据在节点之间进行流动计算(Tensor)
  - Tensor + flow

2.2 图与TensorBoard

2.2.1 什么是图结构

图包含了一组tf.Operation代表的计算单元对象和tf.Tensor代表的计算单元之间流动的数据

2.2.2 图相关操作

- 默认图
- 创建图:
  - 1、一个程序定义多个图
  - 可以通过tf.Graph()自定义创建图
  - 2、会话属于默认的get_default_graph, 只能去运行计算默认图的程序，只能运行一个图
    - 切换运行图：session(graph=new_g)

2.2.3 TensorBoard:可视化学习

- 目的：方便 TensorFlow 程序的理解、调试与优化
- **1 数据序列化-events文件**
- **2 启动TensorBoard**

图中的OP

- operation:TensorFlow API使用大多数都是属于OP(操作)
  - 包含了数学运算接口、模型存储保存接口、常见定义方法
- OP可以
  - 包含一个或多个Tensor输入(数据)
  - 包含一个或多个Tensor输出(数据)
  - 所以OP操作返回的是Tensor类
  - 每一个Tensor，都有一个唯一的节点名称，如果节点方法一样，会给字符串名称增加_+数字

2.2.4.2 指令名称

2.3 会话、张量

2.3.1 会话

会话初始化

- session:掌握会话可能拥有的资源，如 tf.Variable，tf.QueueBase和tf.ReaderBase
  - session关闭，默认这些资源释放
- config=tf.ConfigProto(allow_soft_placement=True, log_device_placement=True)
  - 可以打印显示，图中节点运行的设备信息

运行run()函数

- fetches:要运行的节点输出(tensor)的名称
- 还可以用节点输出的eval()方法运行，必须有session上下文环境
  - `c.eval(session=sess)`

2.3.1.3 feed操作

- placeholder提供占位符，run时候通过feed_dict指定参数
  - 函数的参数
  - 定义个placeholder()方法，实现函数传入参数功能

2.3.2 张量(Tensor)

- 张量的类型和形状
- 创建张量
  - 2.3.3 创建张量的指令, 同numpy创建数组API
  - 固定值
  - 随机值
  - 创建张量特殊的API tf.Variable 变量op
- 变换
  - 类型改变,tf.cast
  - 形状：
    - 动态修改形状:tf.reshape
    - 静态修改形状:tf.set_shape
  - 例子：
    - `[None, None]`
- 静态形状
  - 转换静态形状的时候，1-D到1-D，2-D到2-D，不能跨阶数改变形状
  - 对于已经固定的张量的静态形状的张量，不能再次设置静态形状
- 动态形状
  - tf.reshape()动态创建新张量时，张量的元素个数必须匹配

图、OP、会话、张量()

2.3.6 变量(特殊的张量)

- **存储持久化**
- **可修改值**
- **可指定被训练**

2.3.6.1 创建变量

- tf.Variable这个OP返回不是一个tensor类， 而是返回一个variable类别

- 结论：
  - <tf.Variable 'Variable_1:0' shape=() dtype=float32_ref> Tensor("Add:0", shape=(), dtype=float32)
  - 大部分节点操作接受可以是tensor, variable类型的数量
    - 返回都是tesnor数组类型
  - 节点操作：tf.Variable
    - 接受可以是tensor
    - 返回：variable数组类型, 需要手动去初始化这个数组

```
# 初始化变量
    init = tf.global_variables_initializer()
    # 运行一下init
    sess.run(init)
```

2.4 案例：实现线性回归

- 根据训练数据，训练一个w1x1+w2x2+.....+b = y

2.4.2 案例：实现线性回归的训练

- 训练数据：
  - 特征值：100个点，只有一个特征， 100个样本[100, 1]
  - 目标值：y =(0.8 *x + 0.7)这个模型参数的值，是不知道的
- 1 准备好数据集：y = 0.8x + 0.7 100个样本
- 2 建立线性模型
  - 随机初始化W1和b1
  - y = W·X + b，目标：求出权重W和偏置b
  - 模型需要优化的参数：必须使用tf.Variable节点OP定义
  - tf.Variable是可以被梯度下降选择优化
- 3 确定损失函数（预测值与真实值之间的误差）-均方误差
- 4 梯度下降优化损失：需要指定学习率（超参数）

总结：学习率和步长是会决定你的训练最后时间