应用机器学习概述

吴建军

2020.08

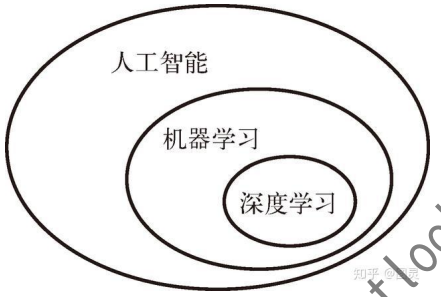
目录

- 总体概述
- 理论基础
- 常用算法
- 平台工具
- 一个例子

总体概述

总体概述

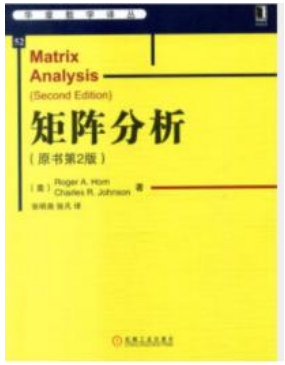
- 机器学习是人工智能的一个分支，人工智能这个概念在1956 年首次提出。
- 人工智能、机器学习、深度学习相互之间的关系为：



- 关于机器学习与人工智能之间的关系还有一个戏谑但是精辟的论述：

总体概述

- 机器学习是交叉学科的领域，涉及统计学，最优化，代数分析，计算机等学科。



- 现在流行机器学习基本都是所谓的统计机器学习，这类方法从大量数据中学习可用规律，它可以分为下面4类：
  - 监督学习：训练集要求包括特征和label，如分类，回归。
  - 无监督学习：训练集没有label，只有特征，如聚类，降维。
  - 半监督学习：部分样本有label，剩下的样本没有label。
  - 强化学习：为了达成目标，随着环境的变动，而逐步调整其行为。

总体概述

- 机器学习目前在工业界有着广泛的应用场景:
  - 物品推荐，
  - 风控，
  - 客服机器人，
  - etc。
- 机器学习应用中涉及的数据类型化主要有下面几种：
  - 结构化数据，
  - NLP，
  - 图像/视频，
  - 轨迹/序列，
- 机器学习应用中难点不少：
  - 高精度-性能，
  - 海量数据-分布式，
  - 稳健-抗攻击，
  - 工程落地，
  - etc。

理论基础

理论基础

- 机器学习建模过程：

  整个机器学习建模过程可以分为如下步骤：

  - 数据准备：我们需要首选获取数据，进行必要的预处理：

    假设我们有数据：$(x_1, y_1), (x_2, y_2), \ldots, (x_K, y_K)$

    我们把它分为训练集合测试集：

    $$\text{train set: } (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

    $$\text{test set: } (x_{N+1}, y_{N+1}), \ldots, (x_K, y_K)$$

  - 问题定义：

    机器学习就是学习一个函数。

    1.我们首先选定一个带参数的函数族来拟合样本：$f(x_n; \theta)$

    比如线性回归中：$f(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} + \theta_0$ 注意并不是所有的函数都能写出其解析式。

    我们希望找到一组最优的参数，使得在每个样本上都拟合良好：$f(\boldsymbol{x}_n, \boldsymbol{\theta}^*) \approx y_n$

    2.接着我们选择一个损失函数来刻画我们总体的拟合误差：

    $$R_{emp} = \frac{1}{N} \sum_{i=1}^{N} l(y_n, \hat{y}_n)$$

    其中，$\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta})$表示了我们的预测值，而 $l(y_n, \hat{y}_n)$表示了预测值和真实值之间的误差。这就是所谓的empirical risk。

理论基础

3.然后，我们选择一个正则项，$\Omega(\theta)$ 。
比如，最简单的二范数正则项，$\|\theta\|^2$
正则项促使模型不要过度拟合数据。
此时我们就得到了完整目标函数：

$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N} l(y_n, \hat{y}_n) + \lambda\Omega(\theta)$$

我们接下来要做的就是求解这个目标函数。

◆ 问题求解：
现在我们就要求解最优的参数 θ^*，使得

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} l(y_n, f(x_n;\theta)) + \lambda\Omega(\theta)$$

这是一个最优化的问题，我们需要采用最优化算法来求解。

◆ 模型评价：
在完成参数的求解后，我们还需要从应用价值上来衡量模型的精度。
此时我们可以使用很多评价指标，如F1，AUC，RMSE等。

◆ 模型部署：
最后一步是模型的部署，我们可以分为在线部署和离线部署。

理论基础

◆ 拟合函数

我们最常见的是拟合函数为线性函数，

$$\theta^T x + b$$

可以用于线性回归(包括SVM)： $y = \theta^T x + b$

还有logistic回归：

$$y = \frac{1}{1 + e^{-\theta^T x - b}}$$

而最简单的两层神经网络的拟合函数可以示例如下：

$$W_2^T(W_1^T x + b_1) + b_2$$

另外，用于回归的决策树的拟合函数可以示例如下：

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

其中M表示叶子节点的个数，R_m表示叶子节点m对应的区域。

理论基础

- 损失函数

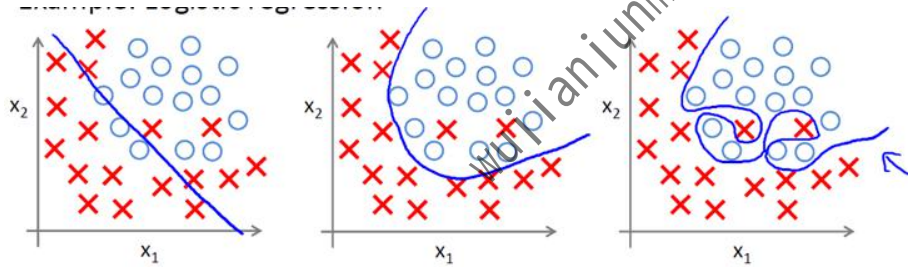
  损失函数有多种，如：

  - 线性回归常用的均方误差： $(y-\hat{y})^2$
  - 二分类常用的logloss： $y\log\hat{y}+(1-y)\log(1-\hat{y})$
  - SVM使用的Hinge损失： $\max(0,1-y\hat{y})$
  - AdaBoost使用的指数损失： $\exp^{-y\hat{y}}$

- 正则化

  正则化技术用于防止过度拟合训练数据导致泛化性能太低。

  下面分别展示了欠拟合，恰拟合和过拟合：



  可以在目标函数中添加正则项，如：

  L2范数： $\|\theta\|^2$

  L1范数： $|\theta|$

  除此之外，还有很多正则化手段。

理论基础

◆ 最优化

我们这里讲的是基于数值计算的最优化，也就是迭代地更新参数，直到收敛。

鉴于绝大多数机器学习算法是基于梯度的最优化求解，所以我们这里只考虑基于梯度的方法：

一个向量到实值的函数，其梯度为：

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \cdots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

基于梯度的方法再分为一阶和二阶，令g_t为梯度：

一阶方法有：

SGD：$\theta_{t+1} = \theta_t - \eta_t g_t$

momentum：$v_t = \gamma v_{t-1} + \eta_t g_t$

$\theta_{t+1} = \theta_t - v_t$

Adam：略

二阶方法就要计算hession矩阵：

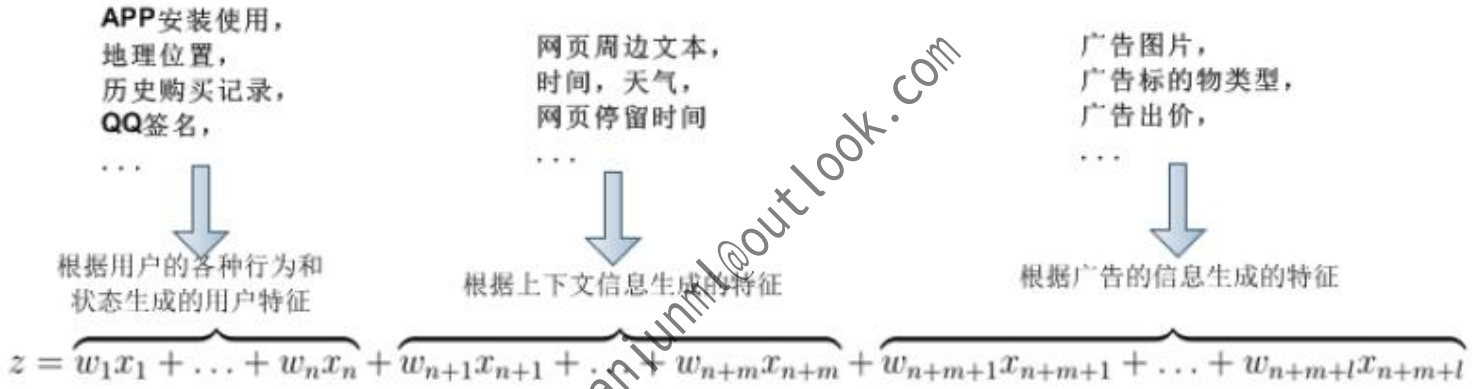
代表算法有：BFGS和L-BFGS

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

常用算法

常用算法

- Logistic Regression
  - LR在评分卡和点击率预测中都是经典算法，
  - 它假设样本为1的概率为：$y = \frac{1}{1+\exp(-z)}$



  - 通常采用交叉熵作为损失函数：

$$L = -[y log\ \hat{y} + (1-y) log\ (1-\hat{y})]$$

  - 采用SGD或BFGS求解。
  - 通常会加入L1或L2范数作为正则项。

常用算法

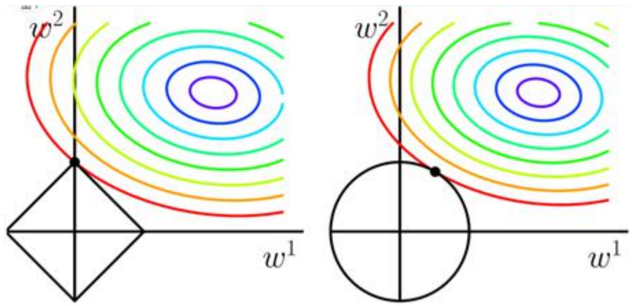
- Logistic Regression
  - 有很多扩展，如：
    - FM：对特征做二阶交叉。

$$\widehat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j \qquad \widehat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{v}_i^T \mathbf{v}_j x_i x_j$$

    - FFM：每个特征属于某个field，每个特征都学习一个相对其他filed的隐向量。

$$\widehat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + r \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbf{v}_{i,f_j}^T \mathbf{v}_{j,f_i} x_i x_j$$

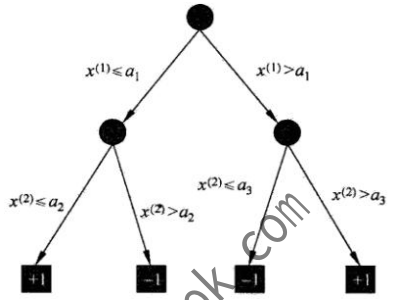
    - FTRL：一种专门针对高维稀疏数据。
      A.它是online learning的算法。
      B.它能够给出最优且高度稀疏的解。
      C.在继承和吸收先前诸多算法，比如TG，FOBOS，RDA。



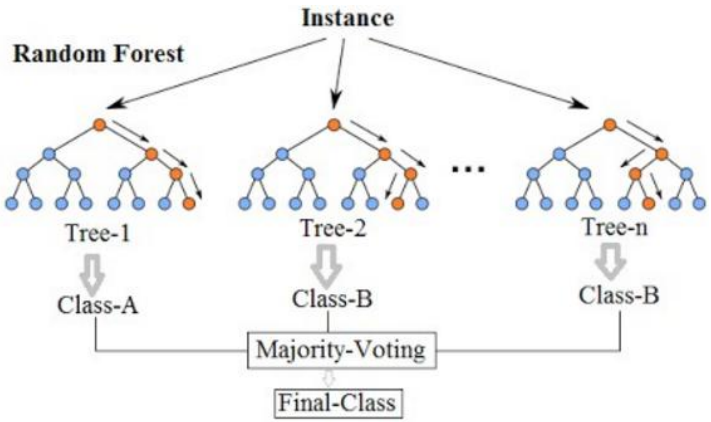
常用算法

- Decision Tree

  决策树可以看做一堆if-then规则的集合：



  - 它不断地对样本做划分，划分的原则很多，比如：信息增益，Gini指数。
  - CART树既可以用于回归，也可以用于分类。做回归时采用平方误差作为划分准则，做分类时采用gini指数做划分。
- 集成学习
  - 单颗决策树能力有限，常常采用集成学习方法学习出一组决策树形成森林。
  - 集成学习又分为bagging和boosting两种。
  - Random Forest就是一种bagging算法。
    它对样本做多次的行列随机采用，
    然后每次独立学习一颗树。



常用算法

- Decision Tree
  - boosting算法比较多，而且在实际中用得更多，我们这里讲讲其中两个
    - XGBoost：
      - 由Tianqi Chen提出，在许多场景下都能取得良好的效果。
      - 他是一种基于梯度的boosting：

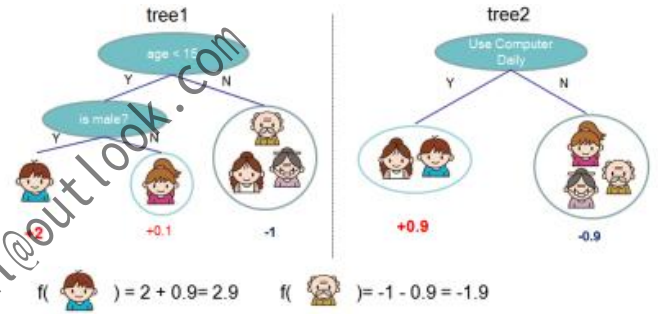


Figure 1: Tree Ensemble Model. The final prediction for a given example is the sum of predictions from each tree.

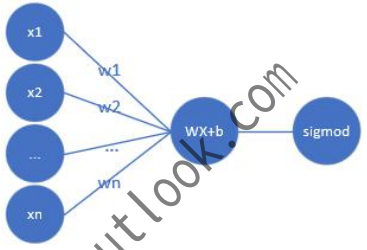
      - 它有很多优点：
        - 正则化：它在代价函数里加入了正则项，是其优于传统GBDT的一个点。
        - 目标函数做二阶展开。
        - 全新的分裂规则，能够直接优化目标函数，而且能够应对稀疏数据。
    - LightGBM
      - 由微软提出，主要有两个优点：
        - Histogram 算法，这可以大大加快算法的训练速度。
        - Leaf-wise 的叶子生长策略，每次从当前所有叶子中，找到分裂增益最大的一个叶子，然后分裂。

常用算法

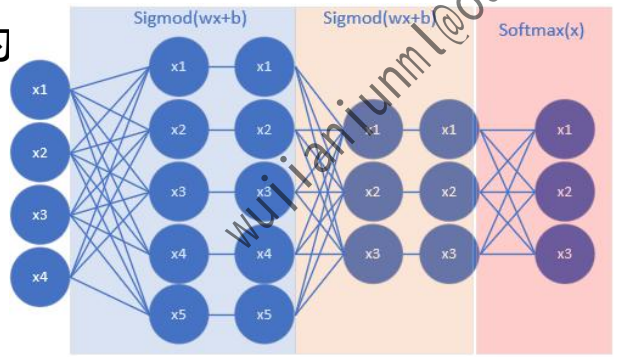
- 神经网络
  - 概述

其实LR就是一个神经网络，只是他仅有一个隐层，一个输出节点：

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots\beta_n X_n)}} = \frac{1}{1+e^{-(\beta_0+\sum \beta_i X_i)}}$$



更复杂一点的神经网络为

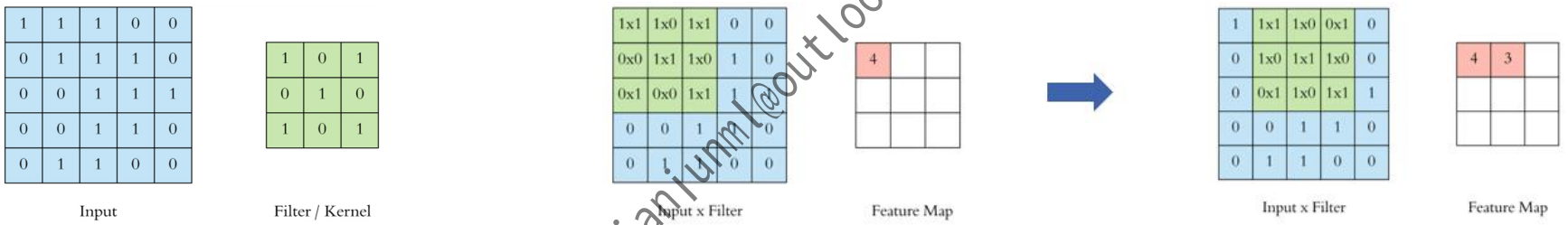


这个网络，首先做5个独立的LR，然后再接上3个独立的LR，最后用softmax输出每个类别的概率。我们再论述下LR与神经网络的关系：

- 所有广义线性模型(LR,SVM等)都是只有一层的神经网络，
- 在神经网络中，LR的使用样式大大扩展(并联，级联，多分类等)，
- 在神经网络中，LR的优化工具更丰富易用(稀疏优化，多种正则化，不平衡学习等)，

常用算法

- 神经网络
  - CNN：
    - 1986年出现第一款成功的CNN：Lenet，
    - 2012年CNN：Alexnet，可以说引爆了眼下的深度学习热潮。
    - CNN的关键是卷积运算，以二维图像上的2D卷积为例，将卷积核(Filter)然后沿着图像的坐标轴逐元素移动，每次把卷积核和卷积核覆盖的子图像矩阵做hadamard 乘积：

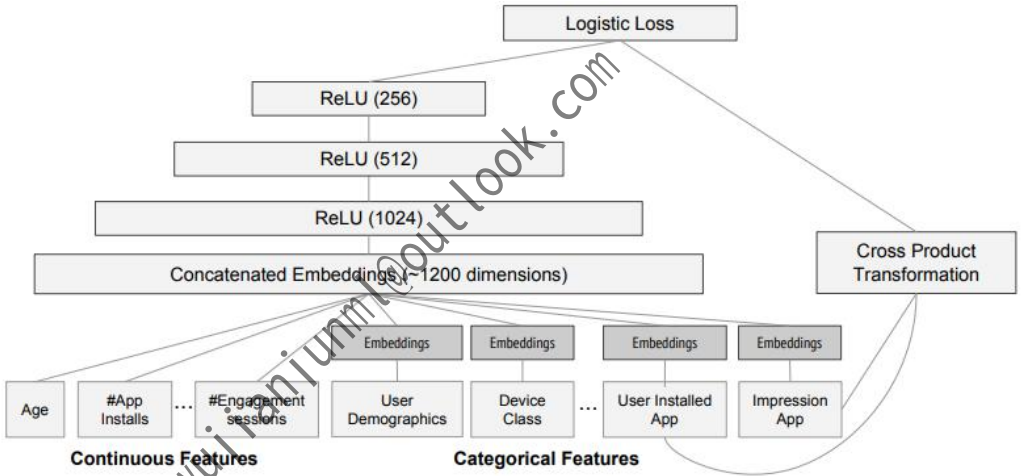


    - 卷积很早就用于提取图像的特征，不同的卷积操作可以提取图像中不同的特征：

    - 发展到现在有很多著名的CNN网络，如ResNet，SENet等。

常用算法

- 神经网络：
  - MLP
    - wide & deep：由Google 提出，结构如下：



    - wide部分是一个线性模型，输入为稀疏标称特征，且包括了交叉特征，
    - deep部分是一个常见的MLP
    - 最后两部分相加得预测结果：

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

  - 还有很多相似的网络，如：
    - deepFM，
    - Deep&Cross，
    - etc。

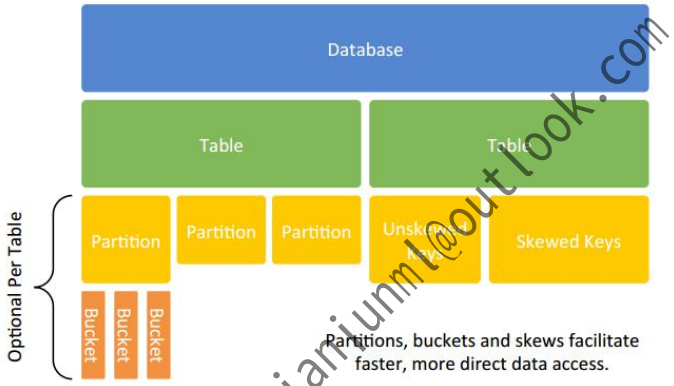
平台工具

平台工具

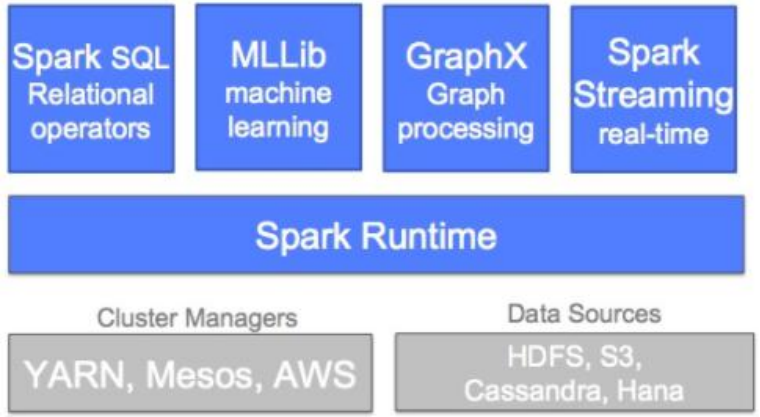
- Hive/Spark

  工业界中数据一般存储在大数据平台中，我们需要进行对其做预处理，常见的工具为：

  - Hive：
    - 使用门槛低，提供类似SQL的HQL预研，自动将HQL转为MR任务，
    - 支持多种文件格式，



  - Spark：
    - 速度快：基于纯内存计算，DAG等。
    - 易于编程：函数式，支持scala，python等语言。
    - 组件丰富：批处理，实时处理，机器学习，图计算。
    - RDD作为核心数据结构。



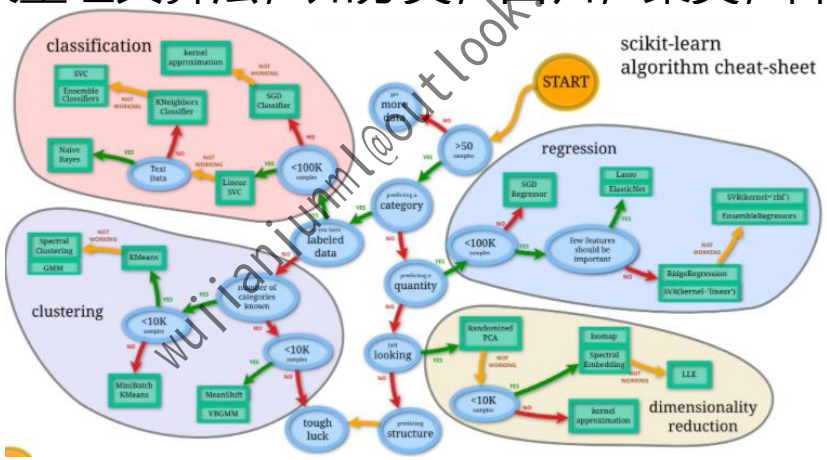
平台工具

- Numpy/Pandas
  - Numpy
    - python的一个库，是科学计算的必备库。
    - 可以非常高效地执行数组类运算，调用大量C/Fortran库。
    - 是最流行的CPU数值计算库。
    - 在机器学习中用于样本数据的存储和计算。
    - 是python环境中最广泛使用的数据格式，几乎所有算法都支持。

  - Pandas
    - 是numpy的一层封装。
    - 支持dataframe：是有多个列的数据表，每个列拥有一个 label，每行有索引.。
    - 主要用于模型训练前的数据分析。

平台工具

- Scikit-learn/scipy
  - Scikit-learn
    - 是python中最著名的机器学习包。
    - 包含大量功能：
      - 数据预处理：规范化，标准化，标称编码，连续分箱等。
      - 模型算法：包含大量经典算法，如分类，回归，聚类，降维等。



      - 选择与评价：超参数搜索，模型指标计算等。
  - scipy
    - 基于NumPy之上，
    - 主要用于高级的数值运算，如微积分，最优化等，
    - 其稀疏格式是python环境中的标准。

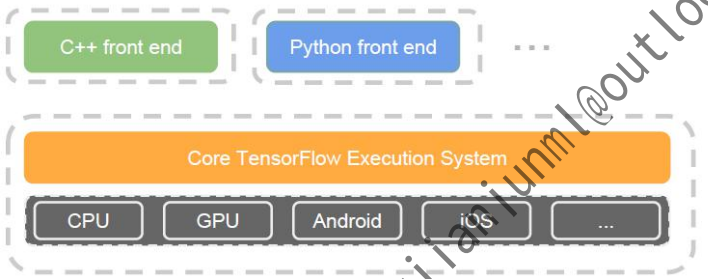
平台工具

- Tensorflow

TensorFlow

TensorFlow is an end-to-end open source platform for machine learning

- 一款自称机器学习平台的软件，实际中主要用做深度学习框架。
- 2015年11月由Google开源。



- Tensorflow入门难度大，系统庞大复杂。
- 主要数据结构为：
  - 使用graph来表示计算任务，图中的节点称为operation，边称为tensor。
  - 在Session中执行图的计算，会话将计算分发到CPU、GPU等设备执行。
  - 使用tensor表示数据。
  - 通过Variable)维护状态。
  - 使用feed 和fetch可以为任意的操作赋值或者从其中获取数据。

平台工具

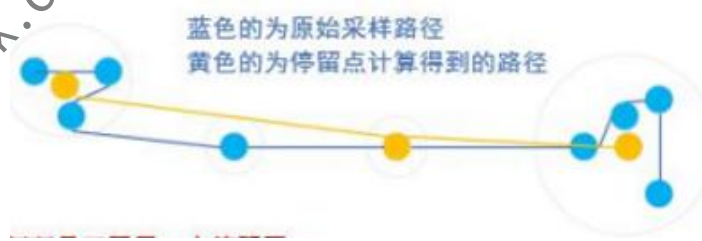
- PyTorch
  - 2017年由facebook开源。
  - 与python生态圈无缝融合，学习门槛低很多。
  - 相比起tensorfow，系统/文档都更简洁。
  - 右边就是一个pytorch的LR实现：


```
class MyDS(Dataset):
    def __init__(self):
        self.x = np.random.randn(500, 2)
        self.y = 5 * self.x[:, 0] + 2 * self.x[:, 1] + 0.001
    def __len__(self):
        return self.x.shape[0]
    def __getitem__(self, i):
        return self.x[i], self.y[i]
myDs = MyDS()
train_loader = DataLoader(myDs, batch_size=128, shuffle=True)
# 实现一个LR
class Net(nn.Module):
    def __init__(self, col_num):
        self.fc = nn.Linear(col_num, 1)
        self.sigmod = nn.Sigmoid()

    def forward(self, x):
        return self.sigmod(self.fc(x))
# 训练LR
def trainNet(net, train_loader, optimizer, epoch_num):
    net.train()
    for epoch in range(epoch_num):
        running_loss = 0.0
        batch_size = 0.0
        for i, (x, y) in enumerate(train_loader):
            optimizer.zero_grad()
            outputs = net(x)
            loss = 0.5 * torch.norm(y - outputs, 2) ** 2
            running_loss += loss.item()
            batch_size += x.shape[0]
            loss.backward()
            optimizer.step()
        running_loss = running_loss / batch_size
        print("epoch=%s, running_loss=%s" % (epoch, running_loss))
    torch.save(net.state_dict(), model_%08d.ckp" % epoch)
#
net = Net(2)
optimizer_sgd = optim.SGD(net.parameters(), lr=0.001)
losses_sgd = trainNet(net, train_loader, optimizer_sgd, 100)
```

一个例子

一个例子

- 问题背景
  - 根据用户LBS轨迹信息预测其预期风险。
- 数据源：
  - 原始轨迹表，(imei，经度，维度，时间)。
  - POI点信息表，(经度，维度，类别，名字)。
- 数据处理过程：
  - 停留点检测：要检测轨迹上停留时间较长的区域。



  - poi点匹配：为每个停留点匹配一个最佳的poi点。分为poi点的初选和精选两个阶段。

```
select imei,
 from_unixtime(cast(cast(inTimestamp as bigint)/1000 as int)) as inTimestamp,
 from_unixtime(cast(cast(outTimestamp as bigint)/1000 as int)) as outTimestamp,
 (outTimestamp-inTimestamp)/1000/60 as staytime,
 arriveType,
 poitag,
 poiInfo,
 RANK() OVER (PARTITION BY imei ORDER BY inTimestamp asc) AS rnk
from risk.tmp_trajectory_semantic
where day = '2019-09-30' and imei = '860032040556419'
limit 200
```

执行记录　查询3

结果　日志　开始时间: 2020-05-09 19:58:43　执行耗时: 1分钟15秒

序号	imei	intimestamp	outtimestamp	staytime	arrivetype	poitag	poiinfo
1	860032040556419	2019-09-30 00:01:47	2019-09-30 06:22:05	380.30033333333336	static	公司企业;公司	中国;浙江省;嘉兴市;秀洲区;浙江省嘉兴市秀洲区欣悦路289号;嘉兴市福海机械有限公司;162.2
2	860032040556419	2019-09-30 06:37:31	2019-09-30 09:32:06	174.58333333333334	walk	公司企业;公司	中国;浙江省;嘉兴市;秀洲区;王江泾镇王江泾开发区新南洋路398号;嘉兴市南洋印染有限公司;24.0
3	860032040556419	2019-09-30 09:37:11	2019-09-30 18:00:32	503.35	walk	公司企业;公司	中国;浙江省;嘉兴市;秀洲区;王江泾镇新南洋路398号南洋印染三楼;美雅花边有限公司;29.5
4	860032040556419	2019-09-30 18:23:49	2019-09-30 21:12:15	168.4396	bus	公司企业;公司	中国;浙江省;嘉兴市;秀洲区;浙江省嘉兴市秀洲区欣悦路289号;嘉兴市福海机械有限公司;151.7

一个例子

◆ 特征生成：下面左图展示了我们完整的特征生成过程

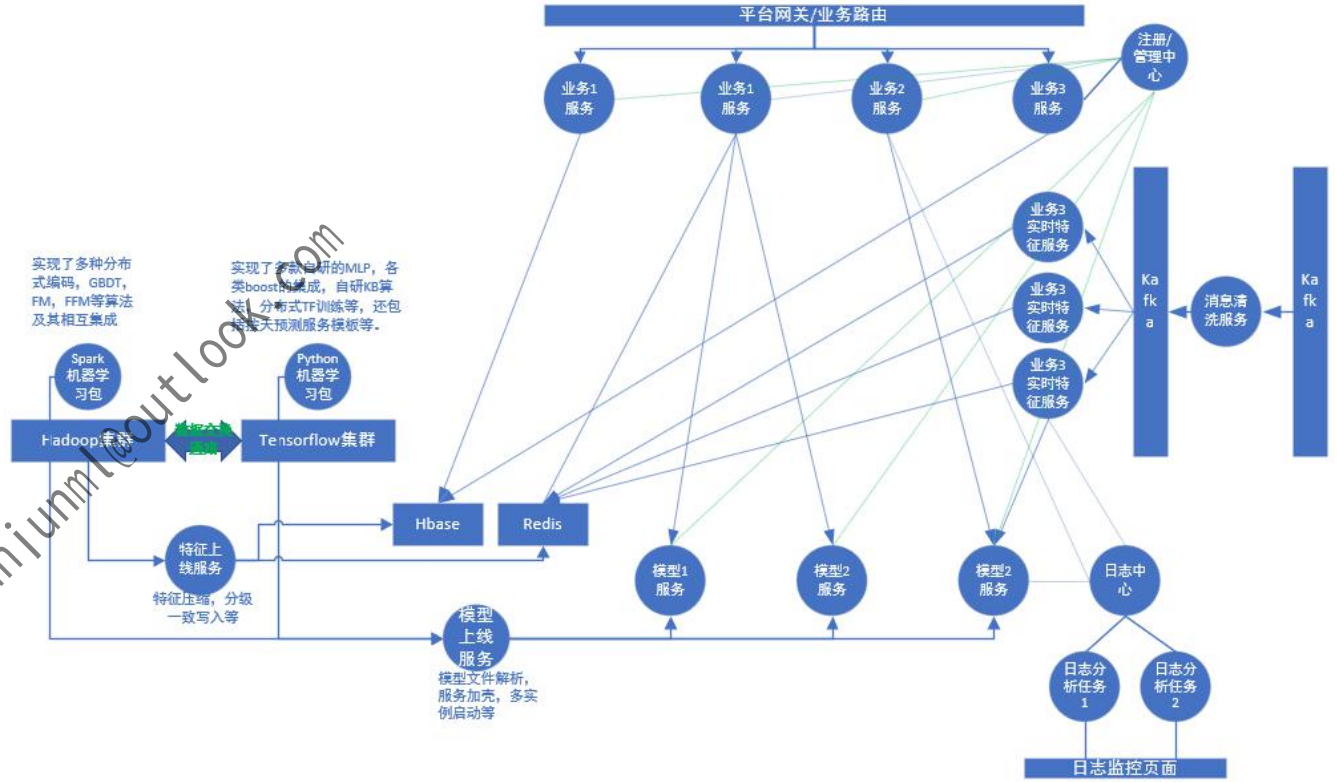
◆ 模型训练：我们采用XGBoost训练模型，如右图：
- ◆ 对Categorical 特征进行TargetEncode编码。
- ◆ 对浮点型特征做标准化。


```python
def trainningXGboost(train_X, train_Y, test_X, test_Y):
    train_data = xgb.DMatrix(train_X, label=train_Y)
    test_data = xgb.DMatrix(test_X)
    num_round = 150
    param = {'booster': 'gbtree',
             'objective': 'binary:logistic',
             'max_depth': 8,
             'gamma': 5,
             'lambda': 50,
             'subsample': 0.7,
             'colsample_bytree': 0.7,
             'eta': 0.05,
             'min_child_weight': 50,
             'seed': 2020,
             'silent': 0}

    bst = xgb.train(param, train_data, num_round)

    y_predict_test = bst.predict(test_data)
    auc_test = roc_auc_score(test_Y, y_predict_test)

    y_predict_train = bst.predict(train_data)
    auc_train = roc_auc_score(train_Y, y_predict_train)

    Log.info("XGBClassifier: auc_test=%s, auc_train=%s" % (auc_test, auc_train))
```

一个例子

- 模型部署：
  - 在线架构示例：
  - 在线模型服务主要由三部分组成：
    - 算法/模型
    - 通信/服务
    - 数据

    实现方法有如下几种：
    - 算法框架本身支持，如TFserving。
    - 借助开源服务框架，如Tornado，dubbo等。
    - 自研一整套服务框架。
    - 自研算法转换器。
    - 存储采用开源KV型DB。



  - 部署步骤一般为：
    - step1.数据由hive注入在线存储，如Hbase/Redis。
    - step2.模型文件在线加载，注意，现在大部分算法都可以实现java/c++对其进行调用。
    - step3.启动与监控。

Q&A