第三章 表格化数据挖掘

极客时间

扫码试看/订阅

《NLP 实战高手课》视频课程

经典的结构化数据挖掘方法

内容概述

- 什么是结构化数据
- 结构化数据的传统建模流程
- 传统建模流程的问题

什么是结构化数据

- 结构化数据指的主要是表格数据（Tabular Data）
- 最简单的想象：可以记载在 Excel 表格中的均是表格数据
  - 每列数据一般称之为一个变量（字段）
  - 变量可以分为离散型变量和连续型变量
- 绝大多数的数据都是表格数据（或可以转化为表格数据）

结构化数据的传统建模流程

- 传统来说（在某种意义上仍是如此），结构化数据常常要求大量的业务理解
- 探索性数据分析往往占很大成分
- 90% 时间花在清洗数据和探索性分析上

传统建模流程的问题

- 最核心的悖论：业务理解从哪里来？
- 一些实际的问题
  - 高维稀疏变量
  - 较差的变量质量
  - 类似的业务理解能力
  - 业务的多变性

表格化数据挖掘基本流程

内容概述

- 数据挖掘竞赛和新的建模流程
- 新的流程
- 关于新的流程的一些说明

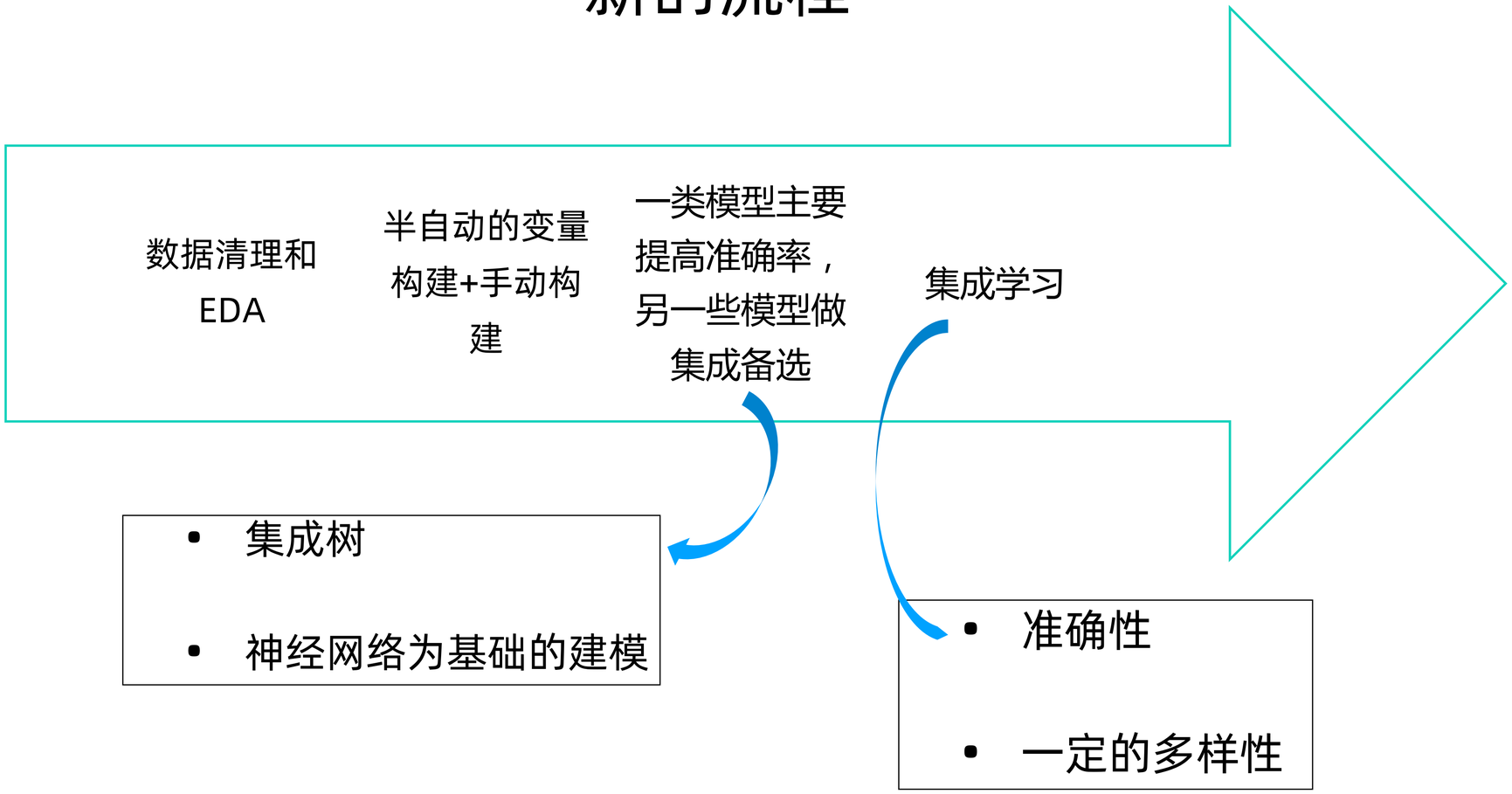
数据挖掘竞赛和新的建模流程：竞赛基本流程和挑战

- 竞赛基本流程
- 挑战
  - 无业务理解
  - 少数提交机会
  - 测试集和训练集可能不一样-> 建模结果必须稳定
  - 时间短暂，必须充分利用时间
- 数据挖掘竞赛和实际业务中的建模有类似挑战，但有一点重要不同
  - 实际业务建模必须要逐步引入更多变量，而竞赛中变量已经给定

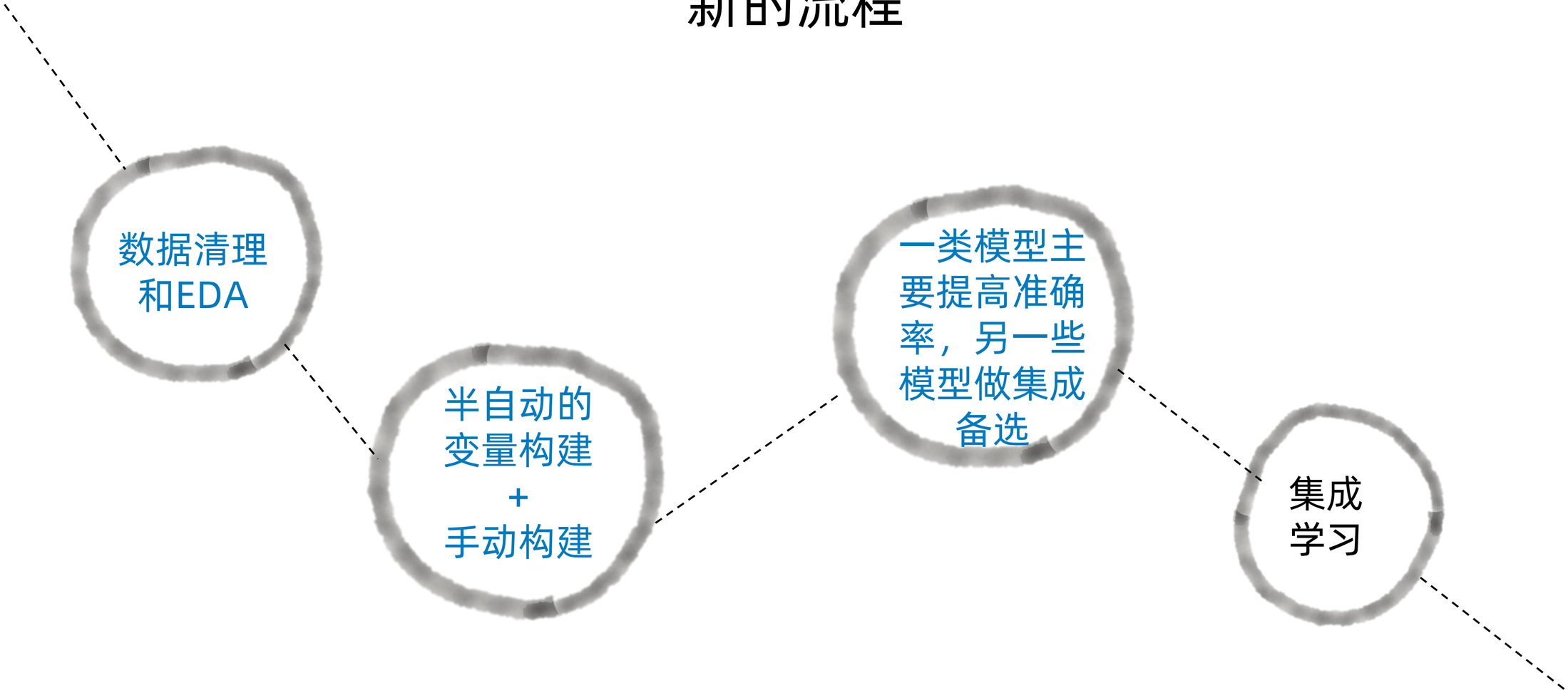
新的流程

- 数据清理和 EDA
- 半自动的变量构建 + 手动构建
- 一类模型主要提高准确率，另一些模型做集成备选
  - 传统来说主要用于提高准确率的是靠集成树的方法
  - 近年来神经网络为基础的建模越来越多
- 集成学习
  - 被集成的模型必须有一定准确性
  - 被集成的模型必须有一定的多样性

新的流程


数据清理和EDA
半自动的变量构建+手动构建
一类模型主要提高准确率，另一些模型做集成备选
集成学习
集成树
神经网络为基础的建模
准确性
一定的多样性

新的流程



关于新的流程的一些说明

- 前三个环节经常需要反复进行
- 基本原则：尽可能利用算力和时间
- 不同人对于 EDA 重要性说法不同，个人建议初学者以实验为主
- 非常重要的原则：结合全局和局部信息

半自动特征构建：Target Mean Encoding

内容概述

- Target Mean Encoding 简介
- 其他的 Groupby 函数应用

Target Mean Encoding 简介

- 对于离散变量最有效编码方法之一
- 对于维度较高的离散性变量效果很好
- 重点在于防止过拟合

Groupby 函数的其他应用

- 实现 Target Mean Encoding 可以使用 df.groupby([A,B])[C].agg(func)
- Groupby 函数可以应用于任何其他变量，在某种程度上捕捉了多变量之间的交叉效应
- Func 可以是多种函数，如 mean，std，kurtosis，min，max 等等
- 可以对残差进行以上操作

半自动特征构建：Categorical Encoders

内容概述

- One-hot Encoder
- Ordinal Encoder
- 其他 Encoder

One-hot Encoder 简介

Ordinal Encoder 简介

其他 Encoder

- 其他 Encoder 包括
  - Count Encoder
  - HashMap
  - ...

半自动特征构建：连续变量的离散化方法

内容概述

- 为什么要对连续变量进行离散化
- 常见的离散化方法
- 树模型的简介

为什么要对连续变量进行离散化

- 捕捉非线性效应
- 捕捉交叉效应

常见的离散化方法

- Uniform
- 基于 Quantile
- 基于聚类
- 基于树

树模型简介

半自动特征构建：Entity Embedding

内容概述

- Entity Embedding 基础
- 如何加入 Vincinal Information

半自动特征构建：连续变量的转换

内容概述

- 常见的数据转换方法
- 基于 ECDF 的方法
- Box-Cox 变换和 Yeo-Johnson Transform

半自动特征构建：缺失值和异常值的处理

内容概述

- 异常值和缺失值的处理概述
- 一些异常值和缺失值的处理方法

概述

- 异常值和缺失值的定义常常是难以确定的
- 异常值的检验最可靠的方法：
  - EDA 加上业务逻辑
  - 可以根据分位数或其他方法（其他方法未必靠谱）
  - 如果异常值是由于输入失误造成的，则可以将之作为缺失值处理
- 缺失值的填充往往可以根据业务决定，其他可以采用平均值、中位数或众数进行填充；也可以单独构建模型进行预测

常见处理方法

- 缺失值的填充往往可以根据业务决定
- 缺失和异常本身可能是有信息量的，可以构造哑变量进行处理
    - 成组的缺失值和异常值本身可能是有信息的
- 对于部分异常值，还需要进行截断处理
- 对于重要的缺失变量，很有可能需要进行预测

自动特征构建方法：Symbolic Learning 和 AutoCross

内容概述

- 自动特征挖掘
- 遗传算法简介
- Symbolic Learning 简介
- AutoCross 简介

自动特征挖掘

- 应用
- 难点：组合优化问题

遗传算法简介

Symbolic Learning 简介

- 采用遗传算法找寻如何构造衍生变量
- 在 gplearn 库当中已经实现
  - 代码：https://gplearn.readthedocs.io/en/stable/examples.html

AutoCross 简介

- 第四范式开发（未开源）
- 主要目的：寻找交叉效应
- 创新
  - Beam Search
  - 简化的逻辑回归求解方式
- 可以进行提升
  - Meta Feature
  - 更好的优化方法

降维方法：PCA，NMF 和 tSNE

内容概述

- 为什么要降维
- PCA 和 NMF 简介
- tSNE 简介
- 实现

为什么要降维

- 找到宏观信息
- 找到交叉效应
- **不建议先降维再拟合模型**

PCA 和 NMF 简介

tSNE 简介

应用

- 在 sklearn 当中均有实现
  - PCA
    - https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html
  - NMF
    - https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.NMF.html
  - tSNE
    - https://scikitlearn.org/stable/modules/generated/sklearn.manifold.TSNE.html

应用

- 隐藏维度的选择取决于数据
- 数据需要进行预处理
  - 标准化
  - 选取重要变量
  - 去掉过于稀疏的个别变量
  - 可构建 2 折和 3 折交叉效应
- 降维方法的参数并不十分重要，一般来说如果有时间，选取所有参数并拟合模型进行测试

降维方法：Denoising AutoEncoder

内容概述

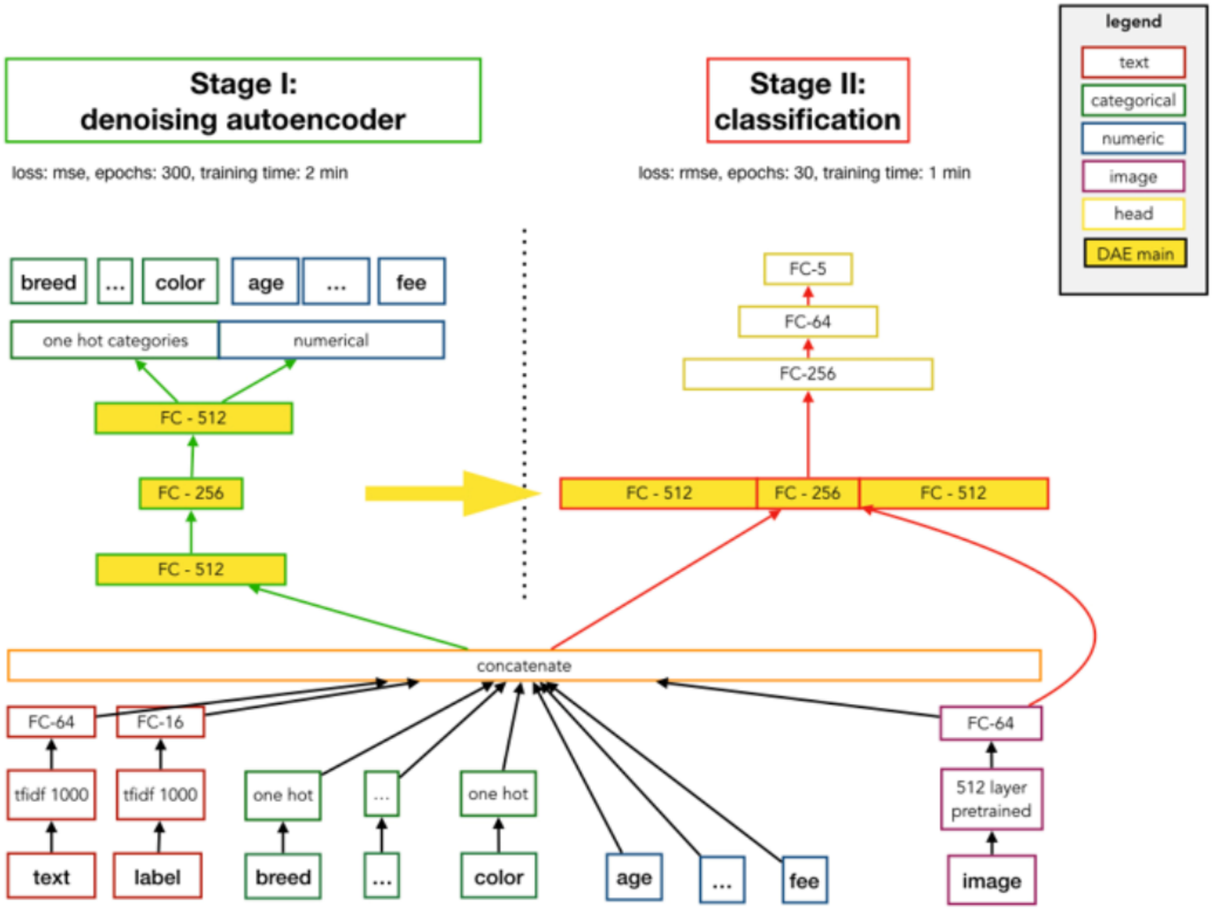
- Denoising AutoEncoder 简介
- 实践中的注意事项

Denoising AutoEncoder 简介

实践中的注意事项

- DAE 一般对多模态有很好效果
- 一般的噪声范畴：5%-20%
- 其他降维方法的 trick 也适用于 DAE
- 注意将中间层均拿出来进行预测

实践中的注意事项


Stage I:
denoising autoencoder
loss: mse, epochs: 300, training time: 2 min
Stage II:
classification
loss: rmse, epochs: 30, training time: 1 min
legend
text
categorical
numeric
image
head
DAE main
breed
...
color
age
...
fee
one hot categories
numerical
FC - 512
FC - 256
FC - 512
FC-5
FC-64
FC-256
FC - 512
FC - 256
FC - 512
concatenate
FC-64
FC-16
tfidf 1000
tfidf 1000
one hot
...
one hot
FC-64
512 layer pretrained
text
label
breed
...
color
age
...
fee
image

降维方法：Variational AutoEncoder

内容概述

- Variational AutoEncoder 的数学形式
- 实践中的注意事项

实践中的注意事项

- VAE 的实现可见
  - https://github.com/1Konny/Beta-VAE
- VAE 是很火的研究领域
  - 训练常常难以收敛
  - 在实际应用中较罕见

变量选择方法

内容概述

- 变量选择概述
- “去一”选择法

变量选择方法

- 变量重要 = 哲学问题
- 优化角度 = 组合优化问题
- 初步选择可根据数据特点
  - 最重要的指标为缺失值和变化率
- 其他的一些选择方法：
  - “去一”的选择方法（主要方法）
  - 模型相关的方法->和模型高度相关，不可靠
  - 其他优化方法->不成熟

“去一”选择法

- 整体流程
- 问题

集成树模型概述

内容概述

- 树模型的缺点和优点
- 集成树模型分类

树模型的优点和缺点

- 优点
  - 非线性效应
  - 交叉效应
  - 稀疏
- 缺点
  - 不稳定
  - 表现力差
  - 精度差

集成树模型分类

- 基本思路：将多个树模型构成进行平均
- 方法：
  - 随机森林类
    - 随机森林（https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html）
    - ExtraTrees（https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html）
  - 梯度提升树：GBDT，XGBoost，LightGBM，CatBoost 等

集成树模型：GBDT 和 XGBoost

内容概述

- GBDT 的数学
- XGBoost 的数学
- XGBoost 实现
- XGBoost 的重要参数

XGBoost 的重要参数

- 最重要的参数：树的深度
- 其他参数：
  - eta：一般选取为 0.01-0.2
  - min_child_weight：建议进行 CV finetune
  - gamma：建议进行 CV finetune
  - Dart 模式：建议选择为 True
- 树的数量可以先少一些，在最终进行增加

补充材料

- GBDT 数学推导：
  - https://towardsdatascience.com/demystifying-maths-of-gradient-boosting-bd5715e82b7c
- XGBoost 数学推导
  - https://xgboost.readthedocs.io/en/latest/tutorials/model.html
- Dart：
  - https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

集成树模型：LightGBM

内容概述

- LightGBM 概述
- LightGBM 具体算法贡献
- LightGBM 参数

LightGBM 概述

- 原始目的->提高 GBDT 的运行效率
- 实际效果->提高了准确性
- 主要贡献
  - Gradient Based One-side Sampling
  - Exclusive Feature Bundling

Gradient Based One-side Sampling

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow \{\}$, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
    preds $\leftarrow$ models.predict(I)
    g $\leftarrow$ $loss$(I, preds), w $\leftarrow$ {1,1,...}
    sorted $\leftarrow$ GetSortedIndices(abs(g))
    topSet $\leftarrow$ sorted[1:topN]
    randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
    usedSet $\leftarrow$ topSet + randSet
    w[randSet] $\times =$ fact ▷ Assign weight $fact$ to the small gradient data.
    newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
    models.append(newModel)

Exclusive Feature Bundling

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges $\leftarrow \{0\}$, totalBin $\leftarrow 0$
for f ***in*** F **do**
    totalBin += f.numBin
    binRanges.append(totalBin)
newBin $\leftarrow$ new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
    newBin[i] $\leftarrow 0$
    for $j = 1$ ***to*** $len(F)$ **do**
        if $F[j].bin[i] \neq 0$ **then**
            newBin[i] $\leftarrow$ F[j].bin[i] + binRanges[j]

Output: $newBin$, $binRanges$

重要参数

- 官方文档：
  - https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

补充材料

- 原始 paper 地址：
  - https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

集成树模型：CatBoost 和 NGBoost

内容概述

- CatBoost 简介
- NGBoost 简介
- 集成树模型回顾

CatBoost 简介

- 适用于离散变量
- 核心思想：Ordered Target Mean Encoding
- 参数：
  - 树的数量
  - 树的深度
  - 学习率
- 注意：不要进行 One-hot 编码
- 实现：https://github.com/catboost/catboost

NGBoost 简介

- 与 GBDT 类似，但是用 Natural Gradient 替代原始的梯度
- Natural Gradient 实际为一阶导数除以二阶导数（和牛顿法类似）
- 实现：https://stanfordmlgroup.github.io/projects/ngboost/
- 计算成本很高

集成树模型总结

- 一般来说，使用 XGBoost 和 LightGBM 作为初始分类器
- CatBoost 和 NGBoost 可作为补充
- 特征工程的效果大于调参

补充材料

- CatBoost 论文：https://arxiv.org/pdf/1706.09516.pdf

神经网络建模：概述

内容概述

- 历史
- 整体方法论
- 计划

历史

- 针对结构化数据，传统的网络主要是 MLP
- 效果不好

新的方法

- 核心
  - 尽可能捕捉不同层次的信息
    - 全局：降维后应用 MLP
    - 重要变量：Transformer
    - 高维稀疏：xDeepFM
  - 尽可能在保留树模型优点的基础上进行提升
- 经常可以将问题进行转换

计划

- 神经网络的常见设计模式
- 神经网络的构成和训练审视
- 常见的网络

神经网络建模：Residual Connection 和 Dense Connection

内容概述

- 核心问题
- Residual Connection
- Dense Connection
- 应用

核心问题

- 信息传递
- 网络深度和预测精度

Residual Connection

Dense Connection

应用

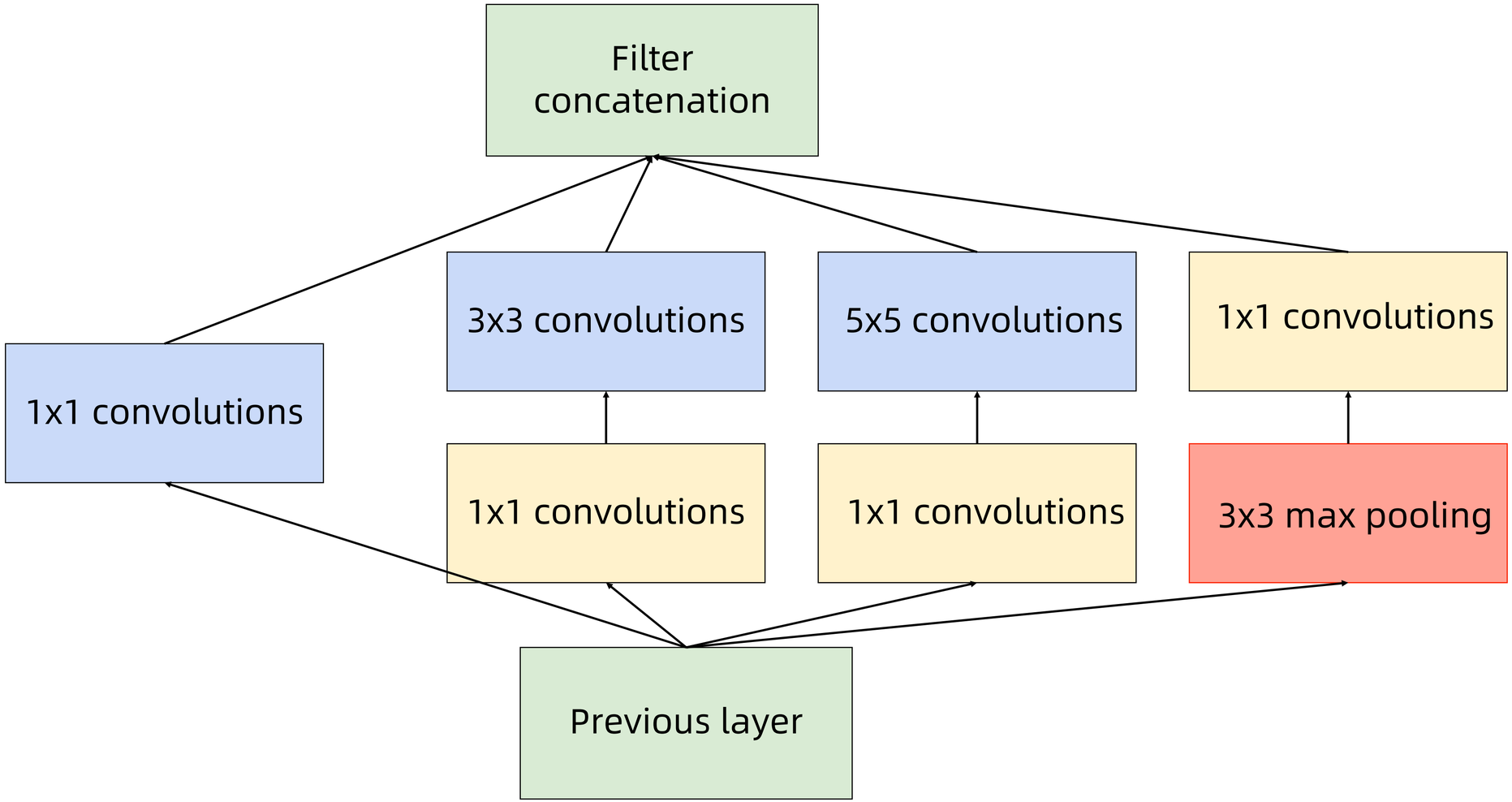
- Residual Connection 可以保证维度
- Dense Connection 将会使维度扩大

神经网络建模：Network in Network

内容概述

- Network in Network
- 应用

Network in Network



应用

- 一种非常有效的提升精度的方法
- 设计会比较耗费时间
- 一般和 AutoML 结合

神经网络建模：Gating 和 Attention

内容概述

- Gating Mechanism 回顾
- Attention 综述和 Multi-head Self-Attention
- Attention 在推断时的一些应用

神经网络建模：Memory

内容概述

- Memory 机制综述
- 综合应用：Compositional Attention Network

神经网络建模：Activation Function

内容概述

- 激活函数回顾
- 一些比较重要的激活函数
- Gradient Clipping

激活函数回顾

一些比较重要的激活函数

Gradient Clipping

- 当出现梯度爆炸时候也许有用
- 在 loss.backward() 和 optimizer.step() 之间调用 torch.nn.clip_grad_value_()
  - （文档见 https://pytorch.org/docs/stable/nn.html?highlight=clip_grad#torch.nn.utils.clip_grad_norm_）
- Value 需要进行尝试

补充材料

- PReLu：https://arxiv.org/pdf/1502.01852.pdf
- ELU：https://arxiv.org/pdf/1511.07289.pdf
- GeLu：https://arxiv.org/pdf/1606.08415.pdf
- Swish：https://arxiv.org/pdf/1710.05941.pdf
- Mish：https://arxiv.org/ftp/arxiv/papers/1908/1908.08681.pdf

神经网络建模：Normalization

内容概述

- Batch Normalization
- 其他 Normalization

Batch Normalization

- 放在 activation 之前还是之后?
- 和 Dropout 的关系?

其他 Normalization

- Layer Normalization
- Group Normalization

补充材料

- Layer Normalization：https://arxiv.org/pdf/1607.06450.pdf
- Instance Normalization：https://arxiv.org/pdf/1607.08022.pdf
- Group Normalization：https://arxiv.org/pdf/1803.08494.pdf

神经网络建模：Activation Function

内容概述

- 初始化的重要性
- 常见初始化方法
- 初始化的 PyTorch 实现

初始化的重要性

常见初始化方法

- 常数初始化->一般效果不佳
- 随机初始化：均匀分布和正态分布
- Xavier 初始化和 Kaiming 初始化
- 一般来说 gain 需要进行调整

初始化的 PyTorch 实现

补充材料

- 大部分实现可以见 PyTorch 官方文档 nn.init
- Xavier Initialization：http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf?source=post_page----------
- Kaiming Initialization：https://arxiv.org/pdf/1502.01852.pdf

神经网络训练：学习率和 Warm-up

内容概述

- 学习率设定
- Warm-up
- PyTorch 实现

学习率设定

- 神经网络炼丹最重要的参数，没有之一
- 理论来说，学习率过小->收敛过慢，学习率过大->错过局部最优
- 实际上来说（可能），学习率过小->不收敛，学习率过大->不收敛
- 设定学习率的一些常用 trick：
  - 首先寻找 ok 的学习率，然后调整其他参数
  - 不同层采用不同的学习率
  - 在最终阶段降低学习率；或者 baby-sitting
  - Warm-up

学习率设定

- 常见的学习率：
  - Finetune：1e-5,2e-5,5e-5
  - 重新训练：没有公认的界定，一般 0.01 开始尝试

Warm-up

- 理论上来说，小学习率有助于模型训练稳定
- 实际对随机初始化的网络而言，开始过小的学习率会导致训练不稳定
- 一般采取 Warm-up：学习率从小到大再到小

PyTorch 实现

神经网络训练：新的训练框架

内容概述

- 分布式训练
- 半精度训练
- 梯度累积
- PyTorch 实现

Transformer：如何通过 Transformer 榨取重要变量

内容概述

- 为什么需要 Transformer
- Transformer 整体架构
- Transformer 技术细节

为什么需要 Transformer

Transformer 整体架构

Transformer 技术细节

xDeepFM：如何用神经网络处理高维的特征

内容概述

- 为什么需要 xDeepFM
- xDeepFM 的整体架构
- xDeepFM 的数学形式

极客时间

扫码试看/订阅

《NLP 实战高手课》视频课程