↵

简述 XGBoost。

XGBoost 是陈天奇大牛新开发的 Boosting 库。它是一个大规模、分布式的通用 Gradient Boosting（GBDT）库，它在 Gradient Boosting 框架下实现了 GBDT 和一些广义的线性机器学习算法。

在面试时，可以通过与 GBDT 的对比来介绍 XGBoost（见 **XGBoost 和 GBDT 有什么不同？**），

加餐内容 - 在直播课程中，我还将带大家一起详细了解

- XGBoost 如何在目标函数中加入正则化项；
- 如何通过二阶导数优化目标函数；
- 如何通过目标函数来衡量节点分裂的时的目标增益；
- 以及如何通过贪心算法和分桶策略来优化节点分裂的效率。

XGBoost 和 GBDT 有什么不同？

- GBDT 是机器学习算法，XGBoost 是该算法的工程实现。
- 在使用 CART 作为基础分类器时，XGBoost 显示地加入了正则项来控制模型的复杂度，有利于防止过拟合，从而提高模型的泛化能力。
- GBDT 在模型训练时只使用了损失函数的一阶导数信息，XGBoost 对代价函数进行二阶泰克展开，可以同时使用一阶和二阶导数。
- 传统的 GBDT 采用 CART 作为基础分类器，XGBoost 支持多种类型的基础分类器，比如线性分类器。
- 传统的 GBDT 在每轮迭代时使用全部的数据，XGBoost 则支持对数据进行采样。
- 传统的 GBDT 没有涉及对缺失值进行处理，XGBoost 能够自动学习出缺失值的处理策略。
- XGBoost 还支持并行计算， XGBoost 的并行是基于特征计算的并行，将特征列排序后以 block 的形式存储在内存中，在后面的迭代中重复使用这个结构。

XGBoost 为什么可以并行训练？

注意 xgboost 的并行不是 tree 粒度的并行，xgboost 也是一次迭代完才能进行下一次迭代的（第 t 次迭代的代价函数里包含了前面 t-1 次迭代的预测值）。

xgboost 的**并行是在特征粒度**上的。我们知道，决策树的学习最耗时的一个步骤就是**对特征的值进行排序**（因为要确定最佳分割点），xgboost 在训练之前，预先对数据

进行了排序，然后保存为 block 结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个 block 结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么**各个特征的增益计算就可以开多线程进行**。

树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以 xgboost 还提出了一种**可并行的直方图算法**，用于高效地**生成候选的分割点**。

XGBoost 防止过拟合的方法？

1. **数据**
   - 样本采样 - 在生成每颗树的时候可以对数据进行随机采样。
   - 特征采样 - 而且还可以在生成每棵树的时候，每棵树生成每一层子节点的时候，以及在每次做节点分裂的时候，选择是否对特征进行采样。

2. **模型**
   - 限制树的深度，树的深度越深模型越复杂。
   - 设置叶子节点上样本的最小数量，这个值越小则树的枝叶越多模型越复杂；相反如果这个值越大，则树的枝叶越少，模型越简单。这个值太小会导致过拟合，太大会导致欠拟合。
3. **正则化**
   - L1 和 L2 正则化损失项的权重
   - 叶子节点数量惩罚项的权重值

XGBoost 为什么这么快？

1）同时采用了损失函数的一阶导数和二阶导数，使得目标函数收敛更快。
2）在进行节点分类时采用的贪心算法和直方图算法，大大加速了节点分裂的计算过程。
3）工程优化，使得模型训练时可进行并行计算。

附加题:
BI

6.推荐系统有哪些常见的评测指标?

按照推荐任务的不同，最常用的推荐质量度量方法可以划分为三类： i. 对预测的评分进行评估，适用于评分预测任务。

b. 对预测的item集合进行评估，适用于Top-N推荐任务。

c. 按排名列表对推荐效果加权进行评估，既可以适用于评分预测任务也可以用于Top-N推荐任务。

a. 评分预测指标：如准确度指标：平均绝对误差（MAE）、均方误差根（RMSE）、标准化平均误差（NMAE）；以及覆盖率（Coverage）

b. 集合推荐指标：如精密度(Precision)、召回(Recall)、 ROC和AUC

c. 排名推荐指标：如half-life和discounted cumulative gain等

CV: GRU 和 LSTM 的区别

2. 区别

1. 对memory 的控制

LSTM： 用output gate 控制，传输给下一个unit

GRU：直接传递给下一个unit，不做任何控制

2. input gate 和reset gate 作用位置不同

LSTM: 计算new memory $\hat{c}^{(t)}$时 不对上一时刻的信息做任何控制，而是用forget gate 独立的实现这一点

GRU: 计算new memory $\hat{h}^{(t)}$ 时利用reset gate 对上一时刻的信息 进行控制。

3. 相似

最大的相似之处就是， 在从t 到 t-1 的更新时都引入了加法。

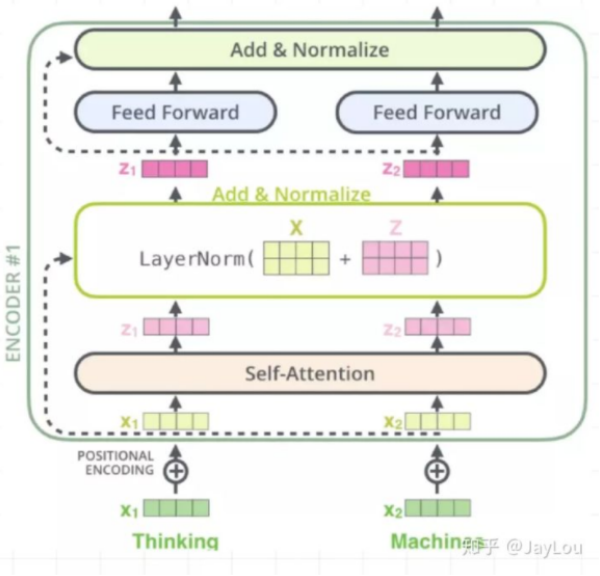
这个加法的好处在于能防止梯度弥散，因此LSTM和GRU都比一般的RNN效果更好。

NLP

Transformer 中的 encoder 和 decoder 的异同点

1.Transformer Encoder

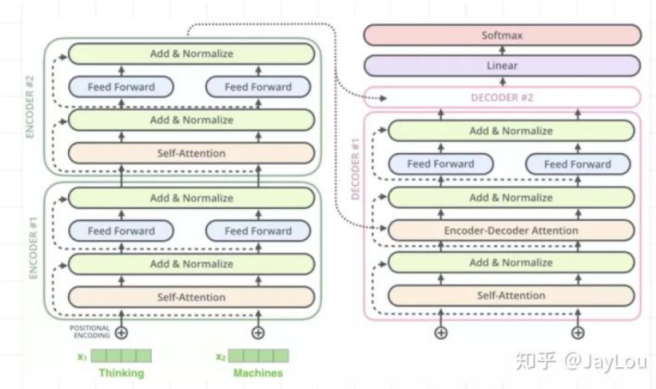
（N=6 层，每层包括 2 个 sub-layers）：



上面这个图真的讲的十分清楚了。

- multi-head self-attention mechanism多头自注意力层： 输出z的shape应该是和x一样的，既然能在残差网络部分相加。
- 全连接网络： $\mathrm{FFN}(x)=\max(0, xW_1+b_1)W_2+b_2$ 使用relu作为激活，并且使用了残差网络： $LayerNorm(X+sublayer(X))$

2.Decoder



- Masked multi-head self-attention mechanism：由于是序列生成过程，所以在时刻 i 的时候，大于 i 的时刻都没有结果，只有小于 i 的时刻有结果，因此需要做 Mask。
- Position-wise Feed-forward Networks全连接层：同 Encoder。
- Encoder-Decoder attention 计算。不同于self-att。

Encoder-Decoder attention与self-att的不同：

前者的q来自解码的输入，kv来自编码输出；后者的qkv来源均是编码的输入。