什么是集成学习算法？

集成学习(Ensemble Learning)就是将多个机器学习模型组合起来，共同工作已达到优化算法的目的。集成学习的一般步骤是：1.生产一组 “个体学习器”(individual leaner);2.用某种策略将它们结合起来。

个体学习器通常由一个现有的学习算法从训练数据产生。在同质集成（系统中个体学习器的类型相同）中，个体学习器又被称为“基学习器”而在异质集成（系统中个体学习的类型不同）中，个体学习器又被称为“组件学习器”(component learner)。

集成学习的思想类似我们俗话常说的“三个臭皮匠胜过一个诸葛亮”。

不好意思，输入错误。Stacking 是异质的，Boosting 是同质的。

集成学习主要有哪几种框架？分别简述这几种集成学习框架的工作过程？

集成学习主要的集成框架有，Bagging，Boosting 和 Stacking。其中 Bagging 和 Boosting 为同质集成，而 Stacking 和 Boosting 为异质集成。

- **Bagging （Bootstrap Aggregating）**：Bagging 的核心思想为**并行地训练一系列各自独立的同类模型**，然后再将各个模型的输出结果按照某种策略进行聚合（例如分类中可采用投票策略，回归中可采用平均策略）。Bagging 方法主要分为两个阶段：
  - Bootstrap 阶段，即采用有放回的抽样方法，将训练集分为 n 个子样本集；并用基学习器对每组样本分别进行训练，得到 n 个基模型。
  - Aggregating 阶段，将上一阶段训练得到的 n 个基模型组合起来，共同做决策。在分类任务中，可采用投票法。比如相对多数投票法，即将结果预测为得票最多的类别。而在回归任务中可采用平均法，即将每个基模型预测得到的结果进行简单平均或者加权平均来获得最终的预测结果。
- **Stacking**：Stacking 的核心思想为**并行地训练一系列各自独立的不同类模型**，然后通过训练一个元模型（meta-model）来将各个模型的输出结果进行结合。也可以用两个阶段来描述 Stacking 算法：
  - 第一阶段，分别采用全部训练样本训练 n 个组件模型，要求这些个体学习器必须是异构的，也就是说采用的学习方法不同；比如可以分别是线性学习器，SVM，决策树模型和深度模型。
  - 第二阶段，训练一个元模型（meta-model）来将各个组件模型的输出结果进行结合。具体过程是，将各个学习器在训练集上得到的预测结果作

为训练特征和训练集的真实结果组成新的训练集；用这个新组成的训练集来训练一个元模型。这个元模型可以是线性模型或者树模型。

- **Boosting**：Boosting 的核心思想为**串行地训练一系列前后依赖的同类模型**，即后一个模型用来对前一个模型的输出结果进行纠正。Boosting 算法是可将弱学习器提升为强学习的算法。学习过程是：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到实现指定的值 T，最终将这 T 个基学习器进行结合。

Boosting 算法有哪两类，它们之间的区别是什么？

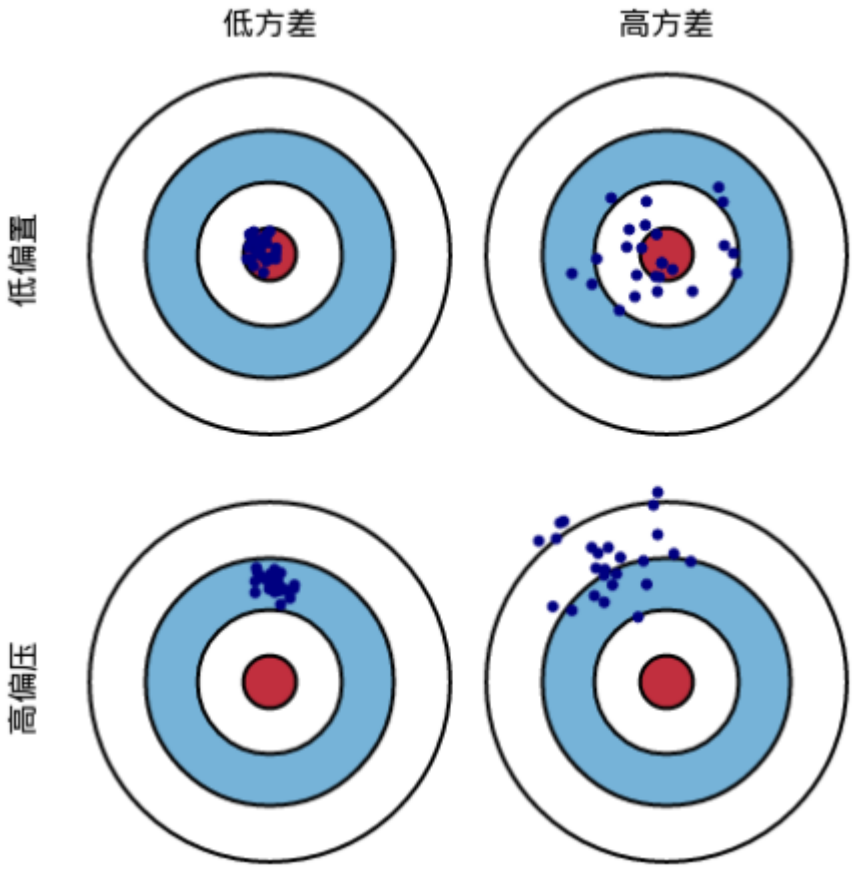
Boosting 算法主要有 AdaBoost（Adaptive Boost）自适应提升算法和 Gradient Boosting 梯度提升算法。最主要的区别在于两者如何识别和解决模型的问题。AdaBoost 用分错的数据样本来识别问题，通过调整分错的数据样本的权重来改进模型。Gradient Boosting 通过负梯度来识别问题，通过计算负梯度来改进模型。

什么是偏差和方差？

偏差指的是预测值的期望与真实值之间的差距，偏差越大，预测值越偏离真实数据的标签。

方差描述的是预测值的变化范围，离散程度，也就是离预测值期望的距离，方差越大，数据的分布越分散。

可通过打靶射击的例子来做类比理解，我们假设一次射击就是一个机器学习模型对一个样本进行预测，射中红色靶心位置代表预测准确，偏离靶心越远代表预测误差越大。偏差则是衡量射击的蓝点离红圈的远近，射击位置即蓝点离红色靶心越近则偏差越小，蓝点离红色靶心越远则偏差越大；方差衡量的是射击时手是否稳即射击的位置蓝点是否聚集，蓝点越集中则方差越小，蓝点越分散则方差越大。



为什么说 Bagging 可以减少弱分类器的方差，而 Boosting 可以减少弱分类器的偏差？

Bagging 算法对数据重采样，然后在每个样本集训练出来的模型上取平均值。假设有 n 个随机变量，方差记为δ^2，两两变量之间的相关性是 0< ρ <1，则 n 个随机变量的均值的方差为：

$$var\left(\frac{1}{n}\sum_{i=1}^{n}X_i\right)=\frac{\delta^2}{n}+\frac{n-1}{n}\rho\delta^2$$

上式中随着 n 增大，第一项趋于 0，第二项趋于$\rho\delta^2$，所以 Bagging 能够降低整体方差。在随机变量完全独立的情况下，n 个随机变量的方差为$\frac{\delta^2}{n}$，n 个随机变量的方差是原来的 1/n。

Bagging 算法对 n 个独立不相关的模型的预测结果取平均，方差可以得到减少，如果模型之间相互独立，则集成后模型的方差可以降为原来的$\frac{1}{n}$，但是在现实情况下，模型不可能完全独立，为了追求模型的独立性，Bagging 的方法做了不同的改进，比如随机森林算法中，每次选取节点分裂属性时，会随机抽取一个属性子集，而不是从所有的属性中选最有属性，这就为了避免弱分类器之间过强的关联性，通过训练集的重采样也能够带来弱分类器之间的一定独立性，这样多个模型学习数据，不会因为一个模型学习到数据某个特殊特性而造成方差过高。

设单模型的期望为 μ，则 Bagging 的期望预测为：

$$E\left(\frac{1}{n}\sum_{i=1}^{n}X_i\right)=\frac{1}{n}E\left(\sum_{i=1}^{n}X_i\right)=E(X_i)\approx\mu$$

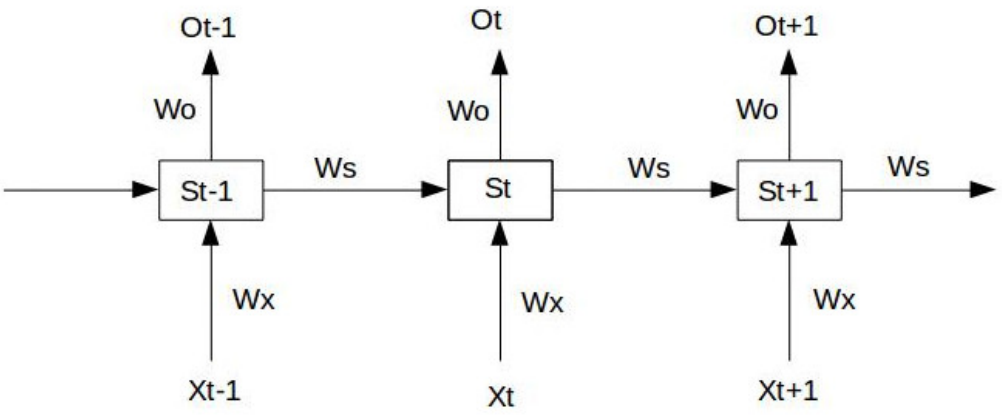
说明 Bagging 整体模型的期望近似于单模型的期望，这意味整体模型的偏差也与单模型的偏差近似。所以 Bagging 不能减少偏差。

Boosting 算法训练过程中，我们计算弱分类器的错误和残差，作为下一个分类器的输入，这个过程本身就在不断减小损失函数，其偏差自然逐步下降。但由于是采取这种串行和自适应的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低方差。所以说 Boosting 主要还是靠降低偏差来提升预测精度。

附加题：

CV：

RNN 发生梯度消失的原因是什么？



假设我们的时间序列只有三段， S_0 为给定值，神经元没有激活函数，则RNN最简单的前向传播过程如下：

$$S_1 = W_x X_1 + W_s S_0 + b_1 \quad O_1 = W_o S_1 + b_2$$

$$S_2 = W_x X_2 + W_s S_1 + b_1 \quad O_2 = W_o S_2 + b_2$$

$$S_3 = W_x X_3 + W_s S_2 + b_1 \quad O_3 = W_o S_3 + b_2$$

假设在t=3时刻，损失函数为 $L_3 = \frac{1}{2}(Y_3 - O_3)^2$ 。

则对于一次训练任务的损失函数为 $L = \sum_{t=0}^{T} L_t$ ，即每一时刻损失值的累加。

使用随机梯度下降法训练RNN其实就是对 W_x 、 W_s 、 W_o 以及 b_1 b_2 求偏导，并不断调整它们以使L尽可能达到最小的过程。

现在假设我们我们的时间序列只有三段，t1，t2，t3。

我们只对t3时刻的 W_x、W_s、W_0 求偏导（其他时刻类似）：

$$\frac{\partial L_3}{\partial W_0} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial W_o}$$

$$\frac{\partial L_3}{\partial W_x} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial W_x} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial W_x} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial S_1}\frac{\partial S_1}{\partial W_x}$$

$$\frac{\partial L_3}{\partial W_s} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial W_s} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial W_s} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial S_1}\frac{\partial S_1}{\partial W_s}$$

可以看出对于 W_0 求偏导并没有长期依赖，但是对于 W_x、W_s 求偏导，会随着时间序列产生长期依赖。因为 S_t 随着时间序列向前传播，而 S_t 又是 W_x、W_s 的函数。

根据上述求偏导的过程，我们可以得出任意时刻对 W_x、W_s 求偏导的公式：

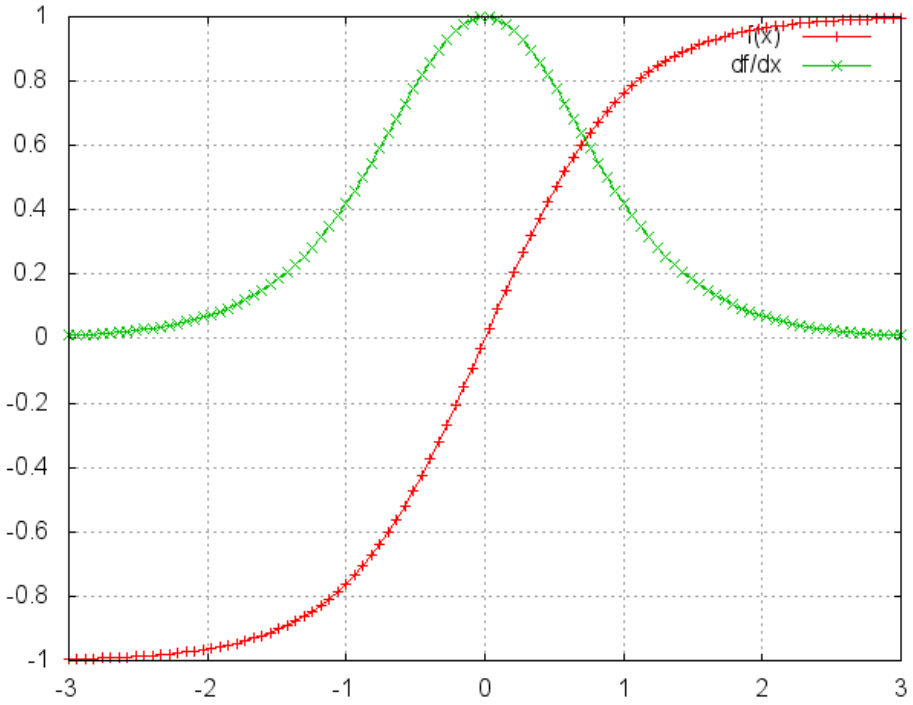
$$\frac{\partial L_t}{\partial W_x} = \sum_{k-0}^{t}\frac{\partial L_t}{\partial O_t}\frac{\partial O_t}{\partial S_t}(\prod_{j=k+1}^{t}\frac{\partial S_j}{\partial S_{j-1}})\frac{\partial S_k}{\partial W_x}$$

任意时刻对 W_s 求偏导的公式同上。

如果加上激活函数， $S_j = tanh(W_x X_j + W_s S_{j-1} + b_1)$ ，

则 $\prod_{j=k+1}^{t}\frac{\partial S_j}{\partial S_{j-1}} = \prod_{j=k+1}^{t} tanh' W_s$

激活函数tanh和它的导数图像如下。



由上图可以看出 $tanh' \leq 1$ ，对于训练过程大部分情况下tanh的导数是小于1的，因为很少情况下会出现 $W_x X_j + W_s S_{j-1} + b_1 = 0$ ，如果 W_s 也是一个大于0小于1的值，则当t很大时 $\prod_{j=k+1}^{t} tanh' W_s$ ，就会趋近于0，和 0.01^{50} 趋近与0是一个道理。同理当 W_s 很大时 $\prod_{j-k+1}^{t} tanh' W_s$ 就会趋近于无穷，这就是RNN中梯度消失和爆炸的原因。

NLP：

写出Attention的公式

Attention机制，里面的q,k,v分别代表什么

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

10.Attention机制，里面的q,k,v分别代表什么

- Q：指的是query，相当于decoder的内容 l K：指的是key，相当于encoder的内容
- V：指的是value，相当于encoder的内容

q和k对齐了解码端和编码端的信息相似度，相似度的值进行归一化后会生成对齐概率值（注意力值）。V对应的是encoder的内容，刚说了attention是对encoder对重编码， qk完成权重重新计算，v复制重编码

BI:

问题:在实时竞价场景中,制定广告主的出价策略是一个什么问题?

答案:

在实时竞价场景中,流量交易平台会把广告流量实时发给广告主;广告主根据流量信息,给出一个竞价,这个竞价实时产生,每次出价时广告主可以决定合适的竞价;流量交易平台接收到所有广告主的竞价之后,把广告位分配给出价最高的广告主,广告主为了广告付出的价格是第二高的竞价,或者平台事先给出的底价.广告主的付费方式有很多种,业内主流的方法是按广告点击收费,即只有用户点击了广告才对广告主收费,没点击则不收费.广告主的出价策略,即为每一个符合广告主条件的广告位设计一个合适的竞价.

广告主在设计竞价时,面临的主要约束是预算.广告主一般会把广告花费按照一次次的营销活动进行分配,一次具体的营销活动会限制在一个时间段内,并且也会有一个预算.广告主希望在这个时间段内花掉这些预算,并取得最好的广告效果.衡量一次营销活动的效果方法有很多,针对在线广告实时竞价场景,我们可以简单地把总的用户点击次数作为衡量指标.这样,制定广告主的出价策略问题就是一个在预算约束下,最大化广告效果的优化问题.