1. 以下说法中正确的是（ ）？

A. SVM 对噪声(如来自其他分布的噪声样本)鲁棒

B. 在 AdaBoost 算法中,所有被分错的样本的权重更新比例相同

C. Boosting 和 Bagging 都是组合多个分类器投票的方法,二者都是根据单个分类器的正确率决定其权重

D. 给定 n 个数据点,如果其中一半用于训练,一般用于测试,则训练误差和测试误差之间的差别会随着 n 的增加而减少

正确答案：（BD）

解析：

SVM 对噪声（如来自其他分布的噪声样本）鲁棒，SVM 本身对噪声具有一定的鲁棒性，但实验证明，是当噪声率低于一定水平的噪声对 SVM 没有太大影响，但随着噪声率的不断增加，分类器的识别率会降低。

在 AdaBoost 算法中所有被分错的样本的权重更新比例相同

AdaBoost 算法中不同的训练集是通过调整每个样本对应的权重来实现的。开始时，每个样本对应的权重是相同的，即其中 n 为样本个数，在此样本分布下训练出一弱分类器。对于分类错误的样本，加大其对应的权重；而对于分类正确的样本，降低其权重，这样分错的样本就被凸显出来，从而得到一个新的样本分布。在新的样本分布下，再次对样本进行训练，得到弱分类器。以此类推，将所有的弱分类器重叠加起来，得到强分类器。

Boost 和 Bagging 都是组合多个分类器投票的方法，二者均是根据单个分类器的正确率决定其权重。

Bagging 与 Boosting 的区别：

取样方式不同。

Bagging 采用均匀取样，而 Boosting 根据错误率取样。

Bagging 的各个预测函数没有权重，而 Boosting 是有权重的。

Bagging 的各个预测函数可以并行生成，而 Boosing 的各个预测函数只能顺序生成。

2. 对于随机森林和 GradientBoosting Trees，下面说法正确的是:（ ）

1.在随机森林的单个树中，树和树之间是有依赖的，而 GradientBoosting Trees 中的单个树之间是没有依赖的

2.这两个模型都使用随机特征子集，来生成许多单个的树

3.我们可以并行地生成 GradientBoosting Trees 单个树，因为它们之间是没有依赖的，GradientBoosting Trees 训练模型的表现总是比随机森林好

A. 2

B. 1 and 2

C. 1 and 3

D. 2 and 3

答案: （A）

解析：

1.随机森林是基于 bagging 的，而 Gradient Boosting trees 是基于 boosting 的，所有说反了,在随机森林的单个树中，树和树之间是没有依赖的，而 GradientBoosting Trees 中的单个树之间是有依赖关系。

2.这两个模型都使用随机特征子集，来生成许多单个的树。

所以 A 是正确的。

3. 对应 GradientBoosting tree 算法， 以下说法正确的是 ：（ ）

1.当增加最小样本分裂个数，我们可以抵制过拟合

2.当增加最小样本分裂个数，会导致过拟合

3.当我们减少训练单个学习器的样本个数，我们可以降低 variance

4.当我们减少训练单个学习器的样本个数，我们可以降低 bias

A. 2 和 4

B. 2 和 3

C. 1 和 3

D. 1 和 4

答案: （C）

解析：

最小样本分裂个数是用来控制“过拟合”参数。太高的值会导致“欠拟合”，这个参数应该用交叉验证来调节。第二点是靠 bias 和 variance 概念的。

4. 简单介绍一下 bagging？

解析：

对数据集进行多次有放回抽样，每次的抽样进行分类计算生成弱分类器，分类问题就是把每一次的计算结果进行投票，看哪一种情况票数多即为最后结果。回归问题就是把所有生成的弱分类器结果进行取平均。

随机森林是 Bagging 的一个拓展变体，在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。泛化误差更低，训练效率更高。

5. 简单介绍一下 boosting，常用 boosting 算法有哪些？

解析：

初始对每个样本分配相同的权重，每次经过分类，把对的结果的权重降低，错的结果权重增高，如此往复，直到阈值或者循环次数。

6. 判断——从偏差-方差分解的角度分析，Boosting 主要关注降低偏差，因此 Boosting 能基于泛化性能相当弱的学习器构建出很强的集成。（ ）

答案：对

7. 判断——从偏差-方差分解的角度看，Bagging 主要关注降低方差，因此它在不剪枝决策树、神经网络等易受扰动的学习器上效用更加明显。（ ）

答案：对

8. 简述 Bagging 和 Boosting 的区别。

解析：

（1） bagging 的训练集是随机的，各训练集是独立的；而 boosting 训练集的选择不是独立的，每一次选择的训练集都依赖于上一次学习的结果；

（2） bagging 的每个预测函数都没有权重；而 boosting 根据每一次训练的训练误差得到该次预测函数的权重；

（3） bagging 的各个预测函数可以并行生成；而 boosting 只能顺序生成。

9. GBDT 和随机森林哪个容易过拟合？

解析：

随机森林，因为随机森林的决策树尝试拟合数据集，有潜在的过拟合风险，而 boosting 的 GBDT 的决策树则是拟合数据集的残差，然后更新残差，由新的决策树再去拟合新的残差，虽然慢，但是难以过拟合。

10. Gradient Boosting 是一种常用的 Boosting 算法，是分析其与 AdaBoost 的异同。

解析：

GradientBoosting 与 AdaBoost 相同的地方在于要生成多个分类器以及每个分类器都有一个权值，最后将所有分类器加权累加起来

不同在于：

AdaBoost 通过每个分类器的分类结果改变每个样本的权值用于新的分类器和生成权值，但不改变每个样本。

GradientBoosting 将每个分类器对样本的预测值与真实值的差值传入下一个分类器来生成新的分类器和权值(这个差值就是下降方向)，而每个样本的权值不变。

11. 试述随机森林为什么比决策树 Bagging 集成的训练速度快。

解析：

随机森林不仅会随机样本，还会在所有样本属性中随机几种出来计算。这样每次生成分类器时都是对部分属性计算最优，速度会比 Bagging 计算全属性要快。

12. 试设计一种能提升 k 近邻分类器性能的集成学习算法。

解析：

可以使用 Bagging 来提升 k 近邻分类器的性能，每次随机抽样出一个子样本，并训练一个 k 近邻分类器，对测试样本进行分类。最终取最多的一种分类。

13. GBDT 的优点和局限性有哪些？

解析：

优点：（1）预测阶段的计算速度快，树与树之间可并行化计算。（2）在分布稠密的数据集上，泛化能力和表达能力都很好，这使得 GBDT 在 Kaggle 的众多竞赛中，经常名列榜首。（3）采用决策树作为弱分类器使得 GBDT 模型具有较好的解释性和鲁棒性，能够自动发现特征间的高阶关系，并且也不需要对数据进行特殊的预处理如归一化等。

局限性：（1）GBDT 在高维稀疏的数据集上，表现不如支持向量机或者神经网络。（2）GBDT 在处理文本分类特征问题上，相对其他模型的优势不如它在处理数值特征时明显。（3）训练过程需要串行训练，只能在决策树内部采用一些局部并行的手段提高训练速度。

14. XGBoost 与 GBDT 的联系和区别有哪些？

解析：

（1）GBDT 是机器学习算法，XGBoost 是该算法的工程实现。

（2）在使用 CART 作为基分类器时，XGBoost 显式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。

（3）GBDT 在模型训练时只使用了代价函数的一阶导数信息，XGBoost 对代价函数进行二阶泰勒展开，可以同时使用一阶和二阶导数。

（4）传统的 GBDT 采用 CART 作为基分类器，XGBoost 支持多种类型的基分类器，比如线性分类器。

（5）传统的 GBDT 在每轮迭代时使用全部的数据，XGBoost 则采用了与随机森林相似的策略，支持对数据进行采样。

（6）传统的 GBDT 没有设计对缺失值进行处理，XGBoost 能够自动学习出缺失值的处理策略。

15. 什么是偏差和方差？

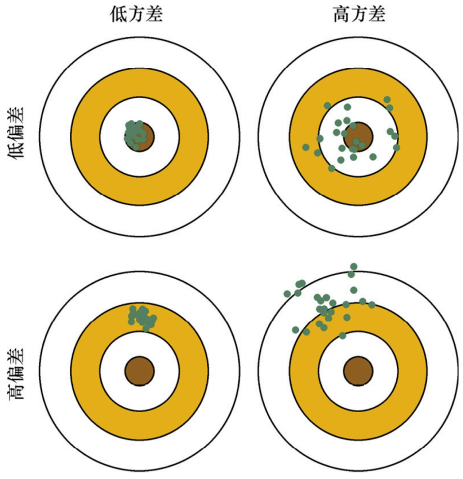
解析：

在有监督学习中，模型的泛化误差来源于两个方面——偏差和方差，具体来讲偏差和方差的定义如下：

偏差指的是由所有采样得到的大小为 m 的训练数据集训练出的所有模型的输出的平均值和真实模型输出之间的偏差。偏差通常是由于我们对学习算法做了错误的假设所导致的，比如真实模型是某个二次函数，但我们假设模型是一次函数。由偏差带来的误差通常在训练误差上就能体现出来。

方差指的是由所有采样得到的大小为 m 的训练数据集训练出的所有模型的输出的方差。方差通常是由于模型的复杂度相对于训练样本数 m 过高导致的，比如一共有 100 个训练样本，而我们假设模型是阶数不大于 200 的多项式函数。由方差带来的误差通常体现在测试误差相对于训练误差的增量上。

上面的定义很准确，但不够直观，为了更清晰的理解偏差和方差，我们用一个射击的例子来进一步描述这二者的区别和联系。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。射中靶心位置代表预测准确，偏离靶心越远代表预测误差越大。我们通过 n 次采样得到 n 个大小为 m 的训练样本集合，训练出 n 个模型，对同一个样本做预测，相当于我们做了 n 次射击，射击结果如图所示。我们最期望的结果就是左上角的结果，射击结果又准确又集中，说明模型的偏差和方差都很小；右上图虽然射击结果的中心在靶心周围，但分布比较分散，说明模型的偏差较小但方差较大；同理，左下图说明模型方差较小，偏差较大；右下图说明模型方差较大，偏差也较大。



16. 如何从减小方差和偏差的角度解释 Boosting 和 Bagging 的原理？

解析：

简单回答这个问题就是：Bagging 能够提高弱分类器性能的原因是降低了方差，Boosting 能够提升弱分类器性能的原因是降低了偏差。

为什么这么讲呢？首先，Bagging 是 BootstrapAggregating 的简称，意思就是再抽样，然后在每个样本上训练出来的模型取平均。假设有 n 个随机变量，方差记为 σ2，两两变量之间的相关性为 ϱ，则 n 个随机变量的均值$\frac{\sum X_i}{n}$的方差为 $\rho*\sigma^2+(1-\rho)*\sigma^2/n$。在随机变量完全独立的情况下，n 个随机变量的方差为 σ2/n，也就是说方差减小到了原来的 1/n。

再从模型的角度理解这个问题，对 n 个独立不相关的模型的预测结果取平均，方差是原来单个模型的 1/n。这个描述不甚严谨，但原理已经讲得很清楚了。当然，模型之间不可能完全独立。为了追求模型的独立性，诸多 Bagging 的方法做了不同的改进。比如在随机森林算法中，每次选取节点分裂属性时，会随机抽取一个属性子集，而不是从所有属性中选取最优属性，这就是为了避免弱分类器之间过强的相关性。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，从而降低 Bagging 后模型的方差。

再看 Boosting，大家应该还记得 Boosting 的训练过程。在训练好一个弱分类器后，我们需要计算弱分类器的错误或者残差，作为下一个分类器的输入。这个过程本身就是在不断减小损失函数，来使模型不断逼近“靶心”，使得模型偏差不断降低。但 Boosting 的过程并不会显著降低方差。这是因为 Boosting 的训练过程使得各弱分类器之间是强相关的，缺乏独立性，所以并不会对降低方差有作用。

关于泛化误差、偏差、方差和模型复杂度的关系如图 12.5 所示。不难看出，方差和偏差是相辅相成，矛盾又统一的，二者并不能完全独立的存在。对于给定的学习任务和训练数据集，我们需要对模型的复杂度做合理的假设。如果模型复杂度过低，虽然方差很小，但是偏差会很高；如果模型复杂度过高，虽然偏差降低了，但是方差会很高。所以需要综合考虑偏差和方差选择合适复杂度的模型进行训练。