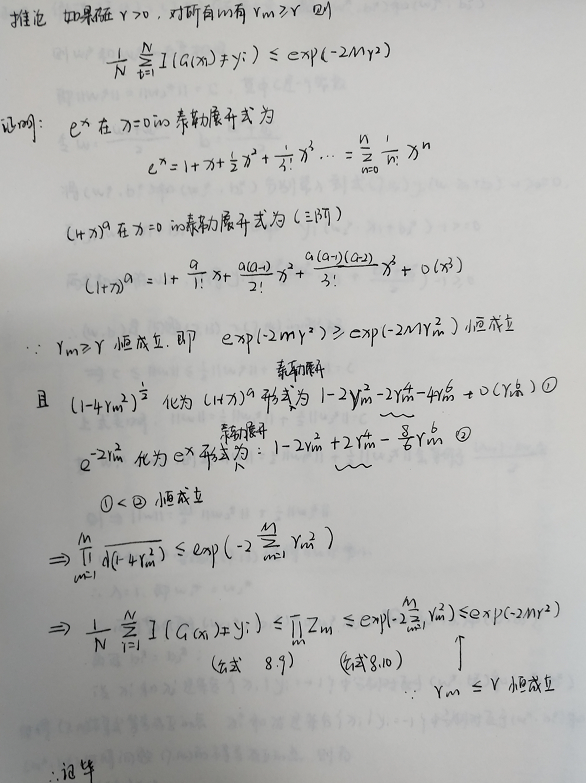
1. （必填）自己提出的问题的理解（罗列全部）：
2. **提出的问题1：**

162页推论8.1是如果如何推出的？

讨论后的理解：



1. **提出的问题2：**

训练样本更新权重为什么要乘以分类器系数？

讨论后的理解：

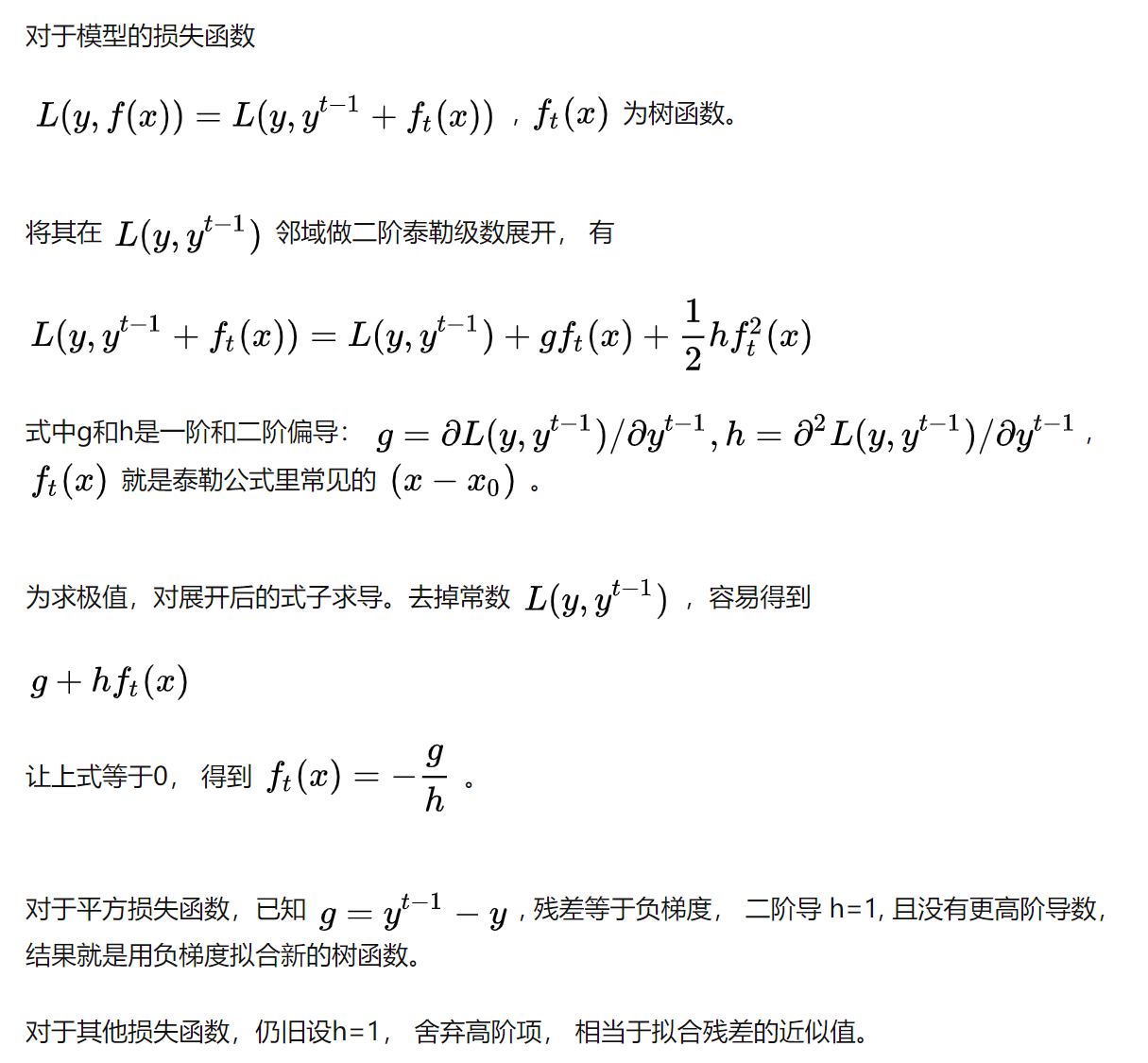
在Adaboost算法中，每一步都会新增加一个弱分类器并且在当前的轮次更新分类器权重，并且在往后的每一步都只修改新加入的弱分类器的权重，不修改原来分类器的权重，因此对于每一个样本来说，加入分类器训练的输入都应该乘上分类器的权重系数，因为根据前面的迭代，具有不同分类效率的分类器都被权重系数修正过，因此为了得到更好的训练结果，训练样本数据也应该乘上分类器的权重系数。

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：

1. **问题3**：

P171 为什么利用损失函数的负梯度作为残差的近似值？

自己的理解：



2. **问题4：**

p167最下面，为什么对回归问题的提升树算法，只需要简单地拟合当前模型的残差？

自己的理解：

对于分类的提升算法，只需要使用优化分类误差率然后再使用加法模型训练分类器就好了，而对于回归问题，没有一个分类问题的那种分类误差率，所以需要寻找一个其他的可以用来优化的量，而这个量就是残差，它是当前分类器的分类结果与数据真实输出值之间的差距，优化这个残差就能够达到优化模型的效果。

1. **问题5：**

170页为什么提升树对指数损失函数和平方损失函数优化简单，损失函数是根据什么选择？

自己的理解：

指数损失函数感觉对应的是Adaboost算法而不是提升树算法，基函数为决策树桩的boost算法（就是提升树）使用了残差的概念来优化提升效果，而平方损失函数的负梯度正好是残差或残差的倍数。优化损失函数与正则项，其实代表的是对参数的极大似然或者极大后验估计，不同的损失函数和正则项，反映的我们对参数先验分布和似然函数的不同假设。

三、（必填）读书计划

1、本周完成的内容章节：统计学习方法（第八章：提升算法）

2、下周计划：统计学习方法（第九章：EM算法及其推广）

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

AdaBoost（adaptive boosting）是元算法，通过组合多个弱分类器来构建一个强分类器。我们为训练数据中的每一个样本都赋予其一个权重，这些权重构成了向量D，一开始，这些权重都初始化成相等值，然后每次添加一个弱分类器对样本进行分类，从第二次分类开始，将上一次分错的样本的权重提高，分对的样本权重降低，持续迭代。此外，对于每个弱分类器而言，每个分类器也有自己的权重，取决于它分类的加权错误率，加权错误率越低，则这个分类器的权重值α越高，最后综合多个弱分类器的分类结果和其对应的权重α得到预测结果，AdaBoost是最好的监督学习分类方法之一。

算法描述：

模型生成

训练数据中的每个样本，并赋予一个权重，构成权重向量D，初始值为1/N

t次循环中的每一次：

在训练数据上训练弱分类器并计算分类器的错误率e

如果e等于0或者大于等于用户指定的阈值：

终止模型，break

重新调整每个样本的权重，其中alpha=0.5\*ln((1-e)/e)

对权重向量D进行更新，正确分类的样本的权重降低而错误分类的样本权重值升高

对于数据集中的每个样例：

如果某个样本正确分类：

权重改为D^(t+1)\_i = D^(t)\_i \* e^(-a)/Sum(D)

如果某个样本错误分类：

权重改为D^(t+1)\_i = D^(t)\_i \* e^(a)/Sum(D)

分类

赋予所有类权重为0

对于t（或小于t）个模型（基分类器）中的每一个：

给模型预测的类加权 -log(e/(1-e))

返回权重最高的类2、伪代码的具体实现(选做)

**梯度提升树**或者梯度提升回归树(GBRT)是任意一个不同损失函数的泛化。GBRT是一个灵敏的并且高效程序，可以用在回归和分类中。梯度提升树模型在许多领域中都有使用，如web搜索排行榜和社会生态学中。它主要的思想是，每一次建立模型是在之前建立模型损失函数的梯度下降方向。这句话有一点拗口，损失函数(loss function)描述的是模型的不靠谱程度，损失函数越大，则说明模型越容易出错（其实这里有一个方差、偏差均衡的问题，但是这里就假设损失函数越大，模型越容易出错）。如果我们的模型能够让损失函数持续的下降，则说明我们的模型在不停的改进，而最好的方式就是让损失函数在其梯度（Gradient)的方向上下降。

GRBT的优势：

混合数据类型的自然处理

预测力强

健壮的输出空间。Boosting主要是一种思想，表示“知错就改”。而Gradient Boosting是在这个思想下的一种函数（也可以说是模型）的优化的方法，首先将函数分解为可加的形式（其实所有的函数都是可加的，只是是否好放在这个框架中，以及最终的效果如何）。然后进行m次迭代，通过使得损失函数在梯度方向上减少，最终得到一优秀的模型。值得一提的是，每次模型在梯度方向上的减少的部分，可以认为是一个“小”的或者“弱”的模型，最终我们会通过加权(也就是每次在梯度方向上下降的距离）的方式将这些“弱”的模型合并起来，形成一个更好的模型。

2、代码实现（选做）

AdaBoost算法实现

def my\_adaboost\_clf(Y\_train, X\_train, Y\_test, X\_test, M=20, weak\_clf=DecisionTreeClassifier(max\_depth = 1)):

n\_train, n\_test = len(X\_train), len(X\_test)

# Initialize weights

w = np.ones(n\_train) / n\_train

pred\_train, pred\_test = [np.zeros(n\_train), np.zeros(n\_test)]

for i in range(M):

# Fit a classifier with the specific weights

weak\_clf.fit(X\_train, Y\_train, sample\_weight = w)

pred\_train\_i = weak\_clf.predict(X\_train)

pred\_test\_i = weak\_clf.predict(X\_test)

# Indicator function

miss = [int(x) for x in (pred\_train\_i != Y\_train)]

print("weak\_clf\_%02d train acc: %.4f"

% (i + 1, 1 - sum(miss) / n\_train))

# Error

err\_m = np.dot(w, miss)

# Alpha

alpha\_m = 0.5 \* np.log((1 - err\_m) / float(err\_m))

# New weights

miss2 = [x if x==1 else -1 for x in miss] # -1 \* y\_i \* G(x\_i): 1 / -1

w = np.multiply(w, np.exp([float(x) \* alpha\_m for x in miss2]))

w = w / sum(w)

# Add to prediction

pred\_train\_i = [1 if x == 1 else -1 for x in pred\_train\_i]

pred\_test\_i = [1 if x == 1 else -1 for x in pred\_test\_i]

pred\_train = pred\_train + np.multiply(alpha\_m, pred\_train\_i)

pred\_test = pred\_test + np.multiply(alpha\_m, pred\_test\_i)

pred\_train = (pred\_train > 0) \* 1

pred\_test = (pred\_test > 0) \* 1

print("My AdaBoost clf train accuracy: %.4f" % (sum(pred\_train == Y\_train) / n\_train))

print("My AdaBoost clf test accuracy: %.4f" % (sum(pred\_test == Y\_test) / n\_test))