机器学习（周志华）参考答案

第一章 绪论（略）

第二章 模型评估与选择

1.数据集包含 1000 个样本，其中 500 个正例，500 个反例，将其划分为包含 70%样本的训练集和 30%样本的测试集用于留出法评估，试估算共有多少种划分方式。

一个组合问题，从 500500 正反例中分别选出 150150 正反例用于留出法评估，所以可能取法应该是$(C_{500}^{150})^2$。

2.数据集包含 100 个样本，其中正反例各一半，假定学习算法所产生的模型是将新样本预测为训练样本数较多的类别（训练样本数相同时进行随机猜测），试给出用 10 折交叉验证法和留一法分别对错误率进行评估所得的结果。

10 折交叉检验：由于每次训练样本中正反例数目一样，所以讲结果判断为正反例的概率也是一样的，所以错误率的期望是 5050%。

留一法：如果留下的是正例，训练样本中反例的数目比正例多一个，所以留出的样本会被判断是反例；同理，留出的是反例，则会被判断成正例，所以错误率是 100%。

3.若学习器 A 的 F1 值比学习器 B 高，试析 A 的 BEP 值是否也比 B 高。

两个分类器的F_1值得大小与他们的BEP值大小并没有明确的关系(没去找)

这道题这里用反推，设计两个BEP值相同的分类器，如果他们的F_1值不一样，那么这道题的结论就是否定的

再加点我看了评论后的疑惑：

BEP值就是F_1值吗？

BEP值是在P=R时取到的，也就是BEP=P=R。如果在计算F时也要定义P=R，那么F_1和F_β将会恒等于BEP，那么P, R, F在这里有什么意义呢？

这里分两种情况：

第一就是我的理解，在计算F1时就是按照分类器真实的分类结果来计算P, R，再根据PR计算F1。当这个分类器正好P=R时，有P=R=BEP=F1。否则BEP的计算不能用当前的PR，而是通过一步一步尝试到查准率=查全率时，P’=R’=BEP。

第二种就是不存在我下面假设的分类器，分类器始终会在P=R的位置进行截断(截断指的是分类器将所有样本按分为正例的可能性排序后，选择某个位置。这个位置前面分类为正，后面分类为负)。但是这个可能吗？这种情况下$F_1 = F_\beta = BEP$恒成立，分类器的评价本质将会变成了样本的正例可能性排序，而不是最终的样本划分结果。

分类器将所有训练样本按自己认为是正例的概率排序，排在越前面分类器更可能将它判断为正例。按顺序逐个把样本标记为正，当查准率与查全率相等时，BEP=查准率=查全率。当然分类器的真实输出是在这个序列中的选择一个位置，前面的标记为正，后面的标记为负，这时的查准率与查全率用来计算$F1$值。可以看出有同样的BEP值的两个分类器在不同位置截断可能有不同的$F1$值，所以$F1$值高不一定BEP值也高。

比如：

1/+	2/+	3/+	4/+	5/+	6/-	7/-	8/-	9/-	10/-
1/+	2/+	3/+	4/+	6/-	5/-	7/-	8/-	9/-	10/-
1/+	2/+	3/+	4/+	6/+	5/-	7/-	8/-	9/-	10/-

第一行是真实的测试样本编号与分类，第二三行是两个分类器对所有样本按为正例可能性的排序，以及判断的结果。显然两个分类器有相同的BEP值，但是他们的$F1$值一个是0.89，一个是0.8。

4.试述真正例率（TPR）、假正例率（FPR）与查准率（P）、查全率（R）之间的联系。

查全率：真实正例被预测为正例的比例

真正例率：真实正例被预测为正例的比例

显然查全率与真正例率是相等的。

查准率:预测为正例的实例中真实正例的比例

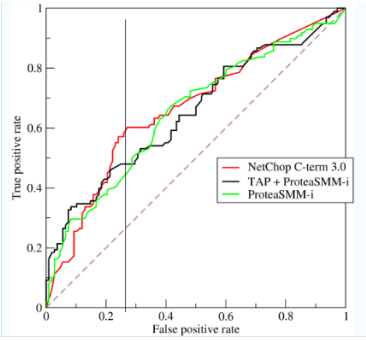
假正例率：真实反例被预测为正例的比例

两者并没有直接的数值关系。

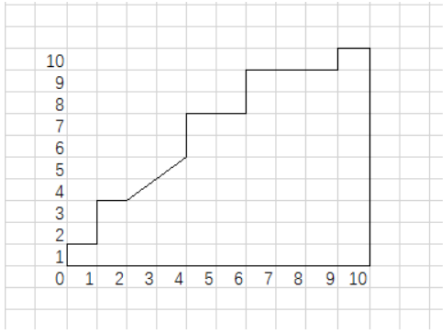
5.试证明(2.22)$AUC = 1 - l_{rank}$

从书34页b图看来，AUC的公式不应该写的这么复杂，后来才发现原来这个图并没有正例反例预测值相等的情况。当出现这种情况时，ROC曲线会呈斜线上升，而不是这种只有水平和垂直两种情况。

由于一开始做题时并没有想过ROC曲线不可以是斜线，所以画了这张图，如果不存在正例反例预测值相等的情况，那么斜线也没必要存在。
但是在维基百科上看到一副图，貌似也存在斜线的ROC，但是不知道含义是否和我这里写的一样。
https://en.wikipedia.org/wiki/Receiver_operating_characteristic
引用一幅有斜线的ROC曲线



与BEP一样，学习器先将所有测试样本按预测概率排序，越可能是正的排在越前面。然后依次遍历，每扫描到一个位置，里面如果只有正例，则ROC曲线垂直向上，如果只有反例，曲线水平往右，如果既有正例也有反例，则斜向上。如图所示



由于TPR与FPR的分母是常数，所以这里按比例扩大了坐标(分别是真实正例和真实反例的数目倍)，可以更好看出曲线走势。

可以看出一共有20个测试样本，10个正，10个反。学习器排序的结果是
$+,-,(+,+),(+,-),(+,-),(+,+),(-,-),(+,+),(-,-,-),+,-$。其中括号内的样本排在相同的位置。
<$(+,+,-,-)$与$(+,-),(+,-)$是同样的效果>

公式2.21累加了所有不在正例的反例数目，其中同样的位置标记为0.5，在正例前面标记为1。从图中可以看出，折线每次向右(右上)延伸，表示扫描到了反例，折线上方对应的面积，就是该反例后面有多少个正例，每个正例是一个正方形，对应的面积是1。同位置上的正例是个三角形，对应的面积是0.5。计算出总面积后，由于ROC图的坐标是归一化的，所以总面积要除以一开始放大的倍数，也就是m^+m^-。

6.试述错误率与ROC曲线之间的关系

ROC曲线每个点对应了一个TPR与FPR，此时对应了一个错误率。
$E_{cost} = (m^+ * (1 - TPR) * cost_{01} + m^- * FPR * cost_{10})/(m^+ + m^-)$
学习器会选择错误率最小的位置作为截断点。

7.试证明任意一条ROC曲线都有一条代价曲线与之对应，反之亦然。

由定义可以知道TPR与FPR都是由0上升到1，那么FNR则是由1下降到0。
每条ROC曲线都会对应一条代价曲线，由于第一条代价线段的是$(0,0),(1,1)$，最后是$(0,1)(1,0)$,
所有代价线段总会有一块公共区域，这个区域就是期望总体代价，而这块区域的边界就是代价曲线，且肯定从$(0,0)$到$(1,0)$。
在有限个样本情况下，ROC是一条折线，此时根据代价曲线无法还原ROC曲线。但若是理论上有无限个样本，ROC是一条连续的折线，代价曲线也是连续的折线，每个点的切线可以求出TPR与FNR，从而得到唯一的ROC曲线。

8.Min-Max规范化与z-score规范化如下所示。试析二者的优缺点。

$Min - max$规范化方法简单，而且保证规范化后所有元素都是正的，每当有新的元素进来，只有在该元素大于最大值或者小于最小值时才要重新计算全部元素。但是若存在一个极大(小)的元素，会导致其他元素变的非常小(大)。
$z - score$标准化对个别极端元素不敏感，且把所有元素分布在0的周围，一般情况下元素越多，0周围区间会分布大部分的元素，每当有新的元素进来，都要重新计算方差与均值。

Max-min	z-score
方法简单	计算量相对大一些
容易受高杠杆点和离群点影响	对离群点敏感度相对低一些
当加入新值超出当前最大最小范围时重新计算所有之前的结果	每加入新值都要重新计算所有之前结果

9.试述卡方检验过程。

步骤

编辑

（1）提出原假设：

H_0：总体X的分布函数为$F(x)$.

如果总体分布为离散型，则假设具体为

H_0：总体X的分布律为$P\{X=x_i\}=p_i$，$i=1,2,\dots$

（2）将总体X的取值范围分成k个互不相交的小区间A1，A2，A3，…，Ak，如可取

A1=（a0，a1]，A2=(a1，a2]，...，Ak=(ak-1,ak)，

其中a0可取-∞，ak可取+∞，区间的划分视具体情况而定，但要使每个小区间所含的样本值个数不小于5，而区间个数k不要太大也不要太小。

（3）把落入第i个小区间的Ai的样本值的个数记作fi，成为**组频数（真实值）**，所有组频数之和f1+f2+...+fk等于样本容量n。

（4）当H0为真时，根据所假设的总体理论分布，可算出总体X的值落入第i 个小区间Ai的概率pi，于是，npi就是落入第i个小区间Ai的样本值的**理论频数（理论值）**。

（5）当H0为真时，n次试验中样本值落入第i个小区间Ai的频率fi/n与概率pi应很接近，当H0不真时，则fi/n与pi相差很大。基于这种思想，皮尔逊引进如下检验统计量 $\chi^2 = \sum_{i=1}^{k} \frac{(f_i - np_i)^2}{np_i}$ ，在0假设成立的情况下服从自由度为k-1的卡方分布。

10.试述在使用$Friedman$检验中使用式(2.34)与(2.35)的区别

书上说$Friedman$检验，在Nk比较大时，平均序值r_i近似于正态分布，均值为$\frac{k+1}{2}$，
方差为$\frac{k^2-1}{12}$ ~~(其实我觉得r_i的方差是$\frac{k^2-1}{12N}$)。~~

即：$r_i \sim N(\frac{k+1}{2}, \frac{k^2-1}{12})$

所以$\frac{12N}{k^2-1}(r_i - \frac{k+1}{2})^2 \sim \chi^2(1)$

统计量$\frac{12N}{k^2-1}\sum_k(r_i - \frac{k+1}{2})^2$由于$k$个算法的平均序值$r_i$是有关联的，知道其中$k-1$个就能推出最后一个，所以自由度为$k-1$,在前面乘上$\frac{k-1}{k}$，最终得到$Friedman$统计量为
$fri = \frac{k-1}{k} * \frac{12N}{k^2-1}\sum_k(r_i - \frac{k+1}{2})^2$

猜测:由于$Friedman$统计量只考虑了不同算法间的影响，而没去考虑不同数据集(其他方差)所带来的影响，所以书上说这个Friedman统计量太保守。
对序值表做方差分析:
总方差$SST = N*(E(X^2) - (EX)^2) = N*k*(k^2-1)/12$自由度$N*(k-1)$
算法间方差$SSA = N*\sum_k(r_i - \frac{k+1}{2})^2$ 自由度$k-1$
其他方差 $SSE = SST - SSA$自由度$(N-1)*(k-1)$

做统计量$f = \frac{SSA/(k-1)}{SSE/((N-1)*(k-1))} = \frac{(N-1)fri}{N(k-1)-fri}$,f服从$(k-1)$和$(N-1)*(k-1)$的F分布

第三章 线性模型

1.试分析在什么情况下，在以下式子中不比考虑偏置项b。

线性模型$y = w^tx + b$,两个实例相减得到$y_i - y_0 = w^t(x_i - x_0)$,以此消除了$b$。所以可以对训练集每个样本都减去第一个样本，然后对新的样本做线性回归，只需要用模型$y = w^tx$。

2.试证明，对于参数 w，对率回归（logistics 回归）的目标函数（式 1）是非凸的，但其对数似然函数（式 2）是凸的。

如果一个多元函数是凸的，那么它的 Hessian 矩阵是半正定的。

$$y = \frac{1}{1+e^{-(w^Tx+b)}}$$
$$\frac{dy}{dw} = \frac{xe^{-(w^Tx+b)}}{(1+e^{-(w^Tx+b)})^2} = x(y-y^2)$$
$$\frac{d}{dw^T}(\frac{dy}{dw}) = x(1-2y)(\frac{dy}{dw})^T = xx^Ty(y-1)(1-2y)$$

xx^T合同于单位矩阵，所以xx^T是半正定矩阵
y的值域为$(0,1)$,当$y \in (0.5,1)$时，$y(y-1)(1-2y) < 0$,导致$\frac{d}{dw^T}(\frac{dy}{dw})$半负定，所以$y = \frac{1}{1+e^{-(w^Tx+b)}}$是非凸的。

$$l(\beta) = \sum_{i=1}^m(-y_i\beta^Tx_i + ln(1+e^{\beta^Tx}))$$
$$\frac{d}{d\beta^T}(\frac{dl}{d\beta}) = xx^Tp1(x;\beta)(1-p1(x;\beta))$$

显然概率$p1 \in (0,1)$，则$p1(x;\beta)(1-p1(x;\beta)) \geq 0$，所以$l(\beta) = \sum_{i=1}^m(-y_i\beta^Tx_i + ln(1+e^{\beta^Tx}))$是凸函数。

3.编程实现对率回归，并给出西瓜数据集 3.0α 上的结果

http://blog.csdn.net/icefire_tyh/article/details/52068844

4.选择两个 UCI 数据集，比较 10 折交叉验证法和留一法所估计出的对率回归的错误率。

http://blog.csdn.net/icefire_tyh/article/details/52068900

5.编程实现线性判别分析，并给出西瓜数据集 3.0α 上的结果。

http://blog.csdn.net/icefire_tyh/article/details/52069003

6. LDA 仅在线性可分数据上能获得理想结果，试设计一个改进方法，使其能较好地用于非线性可分数据。

在当前维度线性不可分，可以使用适当的映射方法，使其在更高一维上可分，典型的方法有KLDA，可以很好的划分数据。

7.令码长为9，类别数为4，试给出海明距离意义下理论最优的EOOC二元码并证明之。

对于$ECOC$二元码，当码长为2^n时，至少可以使$2n$个类别达到最优间隔，他们的海明距离为$2^{(n-1)}$。比如长度为8时，可以的序列为

1	1	1	1	-1	-1	-1	-1
1	1	-1	-1	1	1	-1	-1
1	-1	1	-1	1	-1	1	-1
-1	-1	-1	-1	1	1	1	1
-1	-1	1	1	-1	-1	1	1
-1	1	-1	1	-1	1	-1	1

其中$4,5,6$行是对$1,2,3$行的取反。若分类数为4，一共可能的分类器共有2^4-2种(排除了全1和全0)，在码长为8的最优分类器后添加一列没有出现过的分类器，就是码长为9的最优分类器。

8.EOOC编码能起到理想纠错作用的重要条件是：在每一位编码上出错的概率相当且独立。试析多分类任务经ECOC编码后产生的二类分类器满足该条件的可能性及由此产生的影响。

理论上的$ECOC$码能理想纠错的重要条件是每个码位出错的概率相当，因为如果某个码位的错误率很高，会导致这位始终保持相同的结果，不再有分类作用，这就相当于全0或者全 1的分类器，这点和NFL的前提很像。但由于事实的样本并不一定满足这些条件，所以书中提到了有多种问题依赖的$ECOC$被提出。

9.使用 OvR 和 MvM 将多分类任务分解为二分类任务求解时，试述为何无需专门针对类别不平衡性进行处理。

书中提到，对于 OvROvR，MvMMvM 来说，由于对每个类进行了相同的处理，其拆解出的二分类任务中类别不平衡的影响会相互抵消，因此通常不需要专门处理。以 ECOCECOC 编码为例，每个生成的二分类器会将所有样本分成较为均衡的二类，使类别不平衡的影响减小。当然拆解后仍然可能出现明显的类别不平衡现象，比如一个超级大类和一群小类。

10.试推出多分类代价敏感学习(仅考虑基于类别的错误分类代价)使用“再缩放”能获得理论最优解的条件。

题目提到仅考虑类别分类的误分类代价，那么就默认正确分类的代价为0。
于是得到分类表,(假设为3类)

0	c_{12}	c_{13}
c_{21}	0	c_{23}
c_{31}	c_{32}	0

对于二分类而言，将样本为正例的后验概率设为是p,那么预测为正的代价是$(1-p)*c_{12}$，
预测为负的代价是$p*c_{21}$。当$(1-p)*c_{12} \leq p*c_{21}$样本会被预测成正例，因为他的代价更小。当不等式取等号时，得到了最优划分，这个阀值$p_r = \frac{c_{12}}{c_{12}+c_{21}}$，这表示正例与反例的划分比例应该是初始的$\frac{c_{12}}{c_{21}}$倍。假设分类器预设的阀值是$p_o$,不考虑代价敏感时，当$\frac{y}{1-y} > \frac{p_o}{1-p_o}$时取正例。当考虑代价敏感，则应该是$\frac{y}{1-y} > \frac{1-p_r}{p_r} * \frac{p_o}{1-p_o} = \frac{c_{21}}{c_{12}} * \frac{p_o}{1-p_o}$。
推广到对于多分类，任意两类的最优再缩放系数$t_{ij} = c_{ij}/c_{ji}$,然而所有类别的最优缩放系数并不一定能同时满足。当代价表满足下面条件时，能通过再缩放得到最优解。
设$t_{ij} = w_i/w_j$，则$w_i/w_j = c_{ij}/c_{ji}$对所有i,j成立，假设有k类，共C_k^2个等式，此时代价表中$k*(k-1)$个数，最少只要知道$2*(k-1)$就能推出整张表。

第四章 决策树

4.1.试证明对于不含冲突数据（即特征向量完全相同但标记不同）的训练集，必存在与训练集一致（即训练误差为 0）的决策树。
因为决策树是通过属性来划分，相同属性的样本最终肯定会进入相同的叶节点。一个叶节点只有一个分类，如果样本属性相同而分类不同，必然产生训练误差。反之，决策树只会在当前样本集合是同一类或者所有属性相同时才会停止划分，最终得到训练误差为 0 的决策树。

4.2.试析使用“最小训练误差”作为决策树划分选择的缺陷。
从机器学习最开始就讲起，最小训练误差并不可靠，由于过度学习样本特性最终导致严重的过拟合，而没有泛化能力。

4.3.试编程实现基于信息熵进行划分选择的决策树算法，并为表 4.3 中数据生成一棵决策树。
http://blog.csdn.net/icefire_tyh/article/details/52081556

重写的不剪枝的决策树
http://blog.csdn.net/icefire_tyh/article/details/54575527
即 ID3 算法
4.4.试编程实现基于基尼指数进行划分选择的决策树算法，并为表 4.2 中数据生成预剪枝、后剪枝决策树，并与未剪枝决策树进行比较。
http://blog.csdn.net/icefire_tyh/article/details/52081879
即 CART 算法
4.5.试编程实现基于对率回归进行划分选择的决策树算法，并为表 4.3 中数据生成一棵决策树。
http://blog.csdn.net/icefire_tyh/article/details/52081770
思路：参考书 p90-91 的**多变量决策树**模型，这里我们将每个非叶节点作为一个对率回归分类器，输出为”是”、”否”两类，形成形如二叉树的决策树。
4.6.试选择 4 个 UCI 数据集，对上述 3 种算法所产生的未剪枝、预剪枝、后剪枝决策树进行

实验比较，并进行适当的统计显著性检验。

答案一

简要的分析一下:

ID3 算法基于信息熵增益，CART 算法则采用了基尼系数。两种划分属性选择均是基于数据纯度的角度，方法差距应该不大（CART 可能要好一点）。而对率回归进行划分选择，以斜划分的方式，实现了多变量参与划分，其模型决策边界更光滑。

相比于决策树的生成算法，剪枝操作更影响模型性能。

答案二

这里要对上面三种实现的算法进行未剪枝，预剪枝，后剪枝做比较，对率回归划分就算了，都不知道是个什么情况，信息增益和基尼指数的差别并不大，其实就是为了比较未剪枝，预剪枝，后剪枝对测试样本的输出结果。显著性分析，对 2 种算法，3 种剪枝方式的错误数做方差分析，信息增益和基尼指数有显著区别是拒绝的，未剪枝，预剪枝，后剪枝有显著区别是接受的。

4.7.图 4.2 是一个递归算法，若面临巨量数据，则决策树的层数会很深，使用递归方法易导致“栈”溢出，试使用“队列”数据结构，以参数 maxDepth 控制数的最大深度，写出与图 4.2 等价、但不使用递归的决策树生成算法。

答案一

直接用递归会导致大量的临时变量被保存，当层数过深时会导致“栈”溢出。

用队列对决策树进行层次遍历来生成，用 Max_Depth 来控制树的最大层数。队列中每个元素代表着决策树的每个节点，它必要的属性有：样本集合、剩余属性集合，当前层数指示，父节点序号。队列一开始里面只有一个元素，就是最初初始化，带着所有样本的根节点。然后当队列不为空的时候开始循环，每次取出一个元素，判断是否需要划分，如果不要，就是一个叶节点，出队列就不用管了；如果需要划分，那么找出最好的划分属性，然后划分成 n 个子区间，依次送入队列，继续循环，直到队列为空。

是否需要划分有 3 个依据:

当前所有样本属于一类

当前所有样本属性完全相同

达到了 Max_Depth 的深度

这样就完成了层次遍历(广度优先搜索)对决策树的构建。

显然由于每次出队的元素要先完全划分，那么如果是进行预剪枝算法的决策树，用队列结构是非常方便的。

如果是后剪枝，那必须要等到最终整棵树完全生成，才能进行。

答案二

首先做一些分析:

从数据结构算法的角度来看，生成一棵树常用递归和迭代两种模式。

采用递归时，由于在递归时要存储程序入口出口指针和大量临时变量等，会涉及到不断的压栈与出栈，当递归层次加深，压栈多于出栈，内存消耗扩大。

这里要采用队列数据结构来生成决策树，虽然避免了递归操作产生的内存消耗，但需要更大的额外存储空间。

用 MaxDepth 来控制树的深度，即深度优先（Depth Fisrt）的形式，一般来说，使用递归实现相对容易，当然也可以用非递归来实现。

4.8.试将决策树生成的深度优先搜索过程修改为广度优先搜索，以参数 MaxNode 控制树的最大结点数，将题 4.7 中基于队列的决策树算法进行改写。对比题 4.7 中的算法，试分析哪种方式更易于控制决策树所需储存不超过内存。
本题实际上是 BFS 与 DFS 的比较:

对于深度优先搜索，每深入一层需要存储上一层节点的信息以方便回溯遍历（其存储的是一条路径）;
对于广度优先搜索，每深入一层需要存储当前层兄弟节点信息以实现遍历（其存储的是每层信息，存储量会大一些）;
两种方法各自有防止队列过大化的阈值（即 MaxDepth 和 MaxNode），所以两种方法均可将内存消耗控制在一定范围之内。

当数据属性相对较多，属性不同取值相对较少时，树会比较宽，此时深度优先所需内存较小，反之宽度优先较小。

4.9.试将 4.4.2 节对缺失值的处理机制推广到基尼指数的计算中去。
只需要把信息增益的公式换成基尼指数就行，包括扩展到连续参数，缺失参数，都是很直观的方法。

4.10.从网上下载或自己编程实现任意一种多变量决策树算法，并观察其在西瓜数据集 3.0 上产生的结果。
http://blog.csdn.net/icefire_tyh/article/details/52082051

第五章 神经网络

1.试述将线性函数$f(x)=w^tx$用作神经元激活函数的缺陷。

必须要强调的是，神经网络中必须要有非线性的激活函数，无论是在隐层，还是输出层，或者全部都是。如果单用w^tx作为激活函数，无论多少层的神经网络会退化成一个线性回归，只不过是把他复杂化了。

2.试述使用图5.2(b)激活函数的神经元与对率回归的联系。

两者都是希望将连续值映射到{0,1}上，但由于阶跃函数不光滑，不连续的性质，所以才选择了sigmoid作为映射函数。不同之处在于激活函数不一定要使用sigmoid，只要是非线性的可导函数都可以使用。

3.对于图5.7中v_{ih},试推导出BP算法中的更新公式。

$$-\frac{\partial E_k}{\partial v_{ih}}=-\frac{\partial E_k}{\partial b_h}\frac{\partial b_h}{\partial a_h}\frac{\partial a_h}{\partial v_{ih}}$$

$$\frac{\partial a_h}{\partial v_{ih}}=x_i$$

$e_h=-\frac{\partial E_k}{\partial b_h}\frac{\partial b_h}{\partial a_h}$在书中5.15已经证明

所以得到更新公式ve_hx_i

4.试述学习率的取值对神经网络训练的影响。

如果学习率太低，每次下降的很慢，使得迭代次数非常多。

如果学习率太高，在后面迭代时会出现震荡现在，在最小值附近来回波动。

5.试编程实现标准BP算法与累积BP算法，在西瓜数据集3.0上分别用这个算法训练一个单隐层网络，并进行比较。

6.试设计一个BP改进算法，能通过动态学习率显著提升收敛速度。

```
1  这真是一个蛋疼的题，本来以为方法很多，结果没有一个能用的。
```

固定的学习率要么很慢要么在后期震荡，设计一种自适应的动态学习率算法是有必要的。

- 对四种参数的最速方向做一位搜索
  这是很直观的一种方法，已知f(x)在x_0的导数为d,那么下降方向就是-d。一位搜索就是求试$f(x+td)$最小的t，也就是当前的学习率。
  然而这方法的t用解析法并不好求，$f'_t(x+td)=0$也是无解的。
  使用近似方法尝试了下收敛速度并没有显著提升

- 对四种参数做牛顿迭代
  虽然不符合题目改学习率的要求，但是牛顿法肯定能大大提高收敛速度，只是没有了学习率这个概念。

7.根据式5.18和5.19，试构造一个能解决异或问题的RBF神经网络。

8.从网上下载或自己编程实现一个SOM网络，并观察在西瓜数据集$3.0a$上产生的结果。

9.试推导用于Elman网络的BP算法。

Elman比正常网络多了个反馈，把前一次的b_h作为隐层的输入来调节隐层。
假设用u_{ih}来表示反馈输入与隐层连接的参数，由于前一次计算的b_h作为常数输入，uij与vij的计算方法一样，$\Delta u_{ih}=\eta e_h b_h$，其中e_h书上5.15给出。就是相当于多了几个输入会变的输入层神经元。

10.实现一个卷积神经网络。

第六章 支持向量机

1.试证明样本空间中任意点x到超平面(w,b)的距离为式(6.2)。

超平面(w,b)的平面法向量为w，任取平面上一点x_0，有$w^Tx_0+b=0$。x到平面的距离就是x到x_0的距离往w方向的投影，就是$\frac{|w^T(x-x_0)|}{|w|}=\frac{|w^Tx+b|}{|w|}$。

2.使用libsvm,在西瓜数据集3.0α上分别用线性核和高斯核训练一个SVM,并比较其支持向量的差别。

3.选择两个UCI数据集，分别用线性核和高斯核训练一个SVM，并与BP神经网络和C4.5决策树进行实验比较。