承 诺 书

我们仔细阅读了大学生数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们的题目是： 基于主成分分析、支持向量机和神经网络的放贷决策模型

参赛年级是（一年级，二年级以上，研究生）： 二年级以上

所属学院（请填写完整的全名，可填多个）： 信软

参赛队员姓名学号：1. 张晋豪（学号：2016220204026）

(打印并签名) 2. 吉普照（学号：2016220203011）

3. 刘昌澍（学号：2016220105001）

指导教师或指导教师组负责人 (有的话打印)： 李厚彪

是否愿意参加国内赛（是，否）： 是

日期：2018 年 5 月 20 日

报名队号（请查阅《2018 校内赛报名队信息-0517》后填写）：

2018 电子科技大学大学生数学建模竞赛

编 号 专 用 页

报名队号（请查阅《2018 校内赛报名队信息-0517》后填写）：

评阅记录：

评阅人				
评分				
备注				

基于主成分分析、支持向量机和神经网络的放贷决策模型

摘要

本文针对以往债务人个人信息和违约情况的历史记录，建立可供保理商参考的债务人个人信用和是否放贷的放贷决策模型。

针对以往的大量客户的个人信息和历史违约记录，本文首先根据已经标准化的各项个人信息的参数，在对所有属性已知的客户的表格项进行主成分分析，得到了各项个人信息中对违约记录具有不同影响大小因素的主成分元素集合。基于此，将每个元素对于历史违约记录的影响权重大小从高到低进行排序，取前几个主成分元素作为主成分集合，使得主成分集合在总体决定权重中的占比超过85%，找出了个人信息中哪些信息主要决定客户借款后是否会违约。自此，针对预测客户违约的可能性问题，我们已从完整数据集合中找出数据规律，生成了违约预测模型，将得到的主要因素输入接下来建立的支持向量机模型中。

针对推测数据完整的客户授信额度的问题，本文首先根据前文所提到的违约预测模型利用神经网络建立起了估算模型。在此基础上，根据算法的结果，进行放贷决策。同时，针对预测最后 80 个 class 条目未知的客户是否违约的问题，利用我们建立起的主成分分析与支持向量机模型，由与超平面之间的距离为参数利用支持向量机计算出的概率作为客户违约的可能性。

针对数据有残缺情形下的确定授信额度的分析问题，本小组采用以 L2 范数偏差为依据的热卡方法对残缺的参数进行填补，即先将残缺的部分用对应列整体数据的均值作为暂时的代替值，寻找与当前残缺的最接近的一组数据中的对应行中的对应元素作为缺失值的替代值，因为若使用填补值的新数据集经过主成分分析之后分布可能与给出 284727+80 条数据不同，所以利用数据完整情况下的神经网络模型进行分析。

针对向公司管理层提供非技术报告的问题，我们将根据分析出的主成分进行分析，并结合列表属性的生活实际进行客观解释，对管理层日后对一组数据决策的迅速决策做出科学指导。

关键词：主成分分析，RBF 支持向量机，机器学习，神经网络

一、问题重述

随着当今国内外贸易活动的日益活跃，赊销的方式越来越流行，这也对保理业务提出了新的要求。在大数据和人工智能高速发展的大背景下，确立更加精准可靠的放贷机制成为可能。现实生活中，客户申请放贷的过程中需要提供大量的个人信息，放贷过程中也会产生相关数据。从分析已有的大量案例数据入手，结合相关资料，我们将解决以下几个问题：

1、根据文件数据，寻找数据规律，建立用来预测客户违约可能性的违约预测模型并用实例验证。

2、根据文件数据，分析所有数据之间的联系，建立用于决策授信额度的授信额度估算模型，并用实例验证。

3、结合生活中实际情况，用户提供的信息存在不完整的可能性。结合参数之间的联系，建立有数据缺失情况下的授信额度估算模型。

4、针对上述三个问题的解答，撰写技术报告，展示建模成果。

二、模型假设

1、对于训练条目，数据越完整，不同类别数据的占比越相近，分析出来的模型越真实；对于测试条目，数据越完整，分析出来的结果越合理。

2、根据我们填补缺省值的评价标准，拥有较好填补缺省值性能的填补方法对于数据不完整条目的决策分析所提供的支持更大。

3、假设客户的每一条残缺数据仍处于真实完整客户数据集属性列的分布中。

三、符号说明

符号	含义	单位
C_{i}	第 i 号客户	无
V_i	第 i 号属性	无
CV_{ij}	第 i 号客户的第 j 号属性	无
$class_i$	第 i 号客户是否违约	无
Acc	总体正确度	无
Cri_i	第 i 号客户的授信额度	万元
M_i	第 i 个主成分	无

四、问题分析

4.1 决策任务分解分析

新客户的放贷决策是典型的分类和回归问题。其核心在于分别针对信息完整和存在数据残缺的两种客户的现有个人信息和历史违约记录，并结合已有数据集合进行分析，从而给出合理的个人信用评价指标体系，对模型进行检验和评估。

4.2 数据划分和数据填充分析

为评价决策模型的优劣，考虑从完整数据和缺失数据两种情况来共同确定评价指标体系。对表格中的整体数据条目划分为以下三类：过去信用记录未知（即最后 80 个 class 条目未知）的客户数据；class 完整条目中的大部分（用于生成对完整数据情况下的分析的训练集）；其余部分（用于对缺失情况下的目录进行模拟测试，即选择其中某几个进行遮蔽，将这些条目视作存在缺失数据的条目）。从数据完整的情况出发，关心的是对于所有已知数据的合理利用得出符合以往信用记录（即违约情况）的决策；从数据存在缺失的情况出发，关心的是针对存在残缺值的数据条目，利用热卡方法算出来的模拟缺失值进行填补，对新的条目做出的决策符合以往完整客户数据集的决策。符合完整客户数据集的决策意味着，若该条目以往存在着违约记录，则做出的决策应该是在已有信用额度的基础上进行减少或根本不予贷款（若已有信用额度缺失则使用神经网络预测出模拟值再进行计算）；若该条目以往不存在违约记录，则做出的决策应该是在已有信用额度的基础上增加或至少不变才是合理的。综合考虑完整数据情况和缺失数据两种情况给出评价指标体系。

4.3 数据归一化处理与分析算法

客户在决策系统中提供的参数由户籍所在地、婚姻状况、年龄等属性构成。统计附录中的数据，发现除了部分数值太小需要处理之外，所有的数据条目均可用于分析。先对数据完整的情况进行决策，即对数据进行归一化处理得到归一化之后的矩阵，将此矩阵用于主成分分析，并以此作为支持向量机的分析预测的输入，并得出支持向量机的决策模型，然后作为做出新的放贷决策的依据。通过比较计算出的结果和阈值的大小关系，如果结果大于阈值则确定给予贷款服务，并把其距离超平面距离的大小作为参数输入支持向量机，将计算出的概率作为客户违约的可能性预测值。

4.4 对缺失数据项决策的单独处理

再对数据存在缺失值的情况进行决策，即先用热卡算法计算出模拟值进行填补，接着，运用 bp 神经网络进行处理。将具有完整数据的条目的 28 个参数作为 bp 神经网络的输入层，将 class 即违约历史记录作为输出层，通过正向和反向两个子过程，以 Widrow-Hoff 学习规则为原则，按照梯度下降法，分别调整输入层和隐含层、隐含层和输出层之间的权值和阈值，不断优化，最终得到神经网络的整体结构，即包含每个结点的权值和阈值。至此，我们便获得了针对数据存在缺失值的放贷决策模型，其中的计算结果就已经包含了授信额度和客户违约的可能性。

最后，我们提出了针对缺失值的统一填补算法（如我们采用的热卡算法）在不同缺失情况下的评价体系，并记录在我们的非技术报告中。

五、模型建立与求解

5.1 基于主成分分析、支持向量机、神经网络方法的在数据完整情况下放贷决策模型（违约可能性预测与授信额度估算模型）

5.1.1 模型建立

根据表格将客户的 29 个个人信息属性分别表示为$V_i, i=1,2,...,29$，例如$V_1=$“户籍所在地”；每个属性决定权大小为$d_i, i=1,2,...,29$，其中 n 个对客户违约情况具有主要决定权的属性表示为$M_j, j=1,2,....,n(n<29)$，对应属性决定权大小为$d_j, j=1,2,...,n(n<29)$，其中这 n 个属性对放贷决策情况的决定权之和占总体决定权的比重（累计贡献率）大于 85%，即

$$\frac{\sum_{j=1}^{n} d_j}{\sum_{i=1}^{29} d_i} \geq 85\%$$

通过分析之前找出来的 n 个具有主要决定权的属性，试图寻找一种分类方式$f(CV_i)(CV_i=\{CV_{i1},CV_{i2},...,CV_{in}\})(n<29)$, 其中此时的$CV_{ij}$为之前寻找出的具有主要决定权的属性，通过$f(CV_i)$将是否违约的两类客户正确分开，并使的两类客户的个人属性集合通过这种分类方式所计算出的差异性尽可能大，从而实现合理划分。

这时，再将属性完整的待预测客户的主要属性输入到$f(CV_i)$中，通过比较计算结果和阈值的大小，判断出*class*的值，即做出是否违约的判断。

授信额度的分析单独使用神经网络进行分析。将所有条目中的 28 个属性作为参数传入输入层，将授信额度作为标签传入输出层，然后根据 Widrow-Hoff 学习规则从正向和反向两个子过程进行回归学习模拟，最终生成完整的神经网络图，将此网络结构图作为授信额度的分析模型，回归分析出新用户 class 未知的 80 个客户的授信额度预测值。

5.1.2 模型求解算法

5.1.2.1 主成分分析算法

对原始客户数据矩阵CV进行标准化处理, 将CV_{ij}变换为$CV_{ij}^{'}$：

$$CV_{ij}^{'}=\frac{CV_{ij}-E(CV_{*j})}{\sqrt{D(CV_{*j})}},$$

$E(CV_{*j})$表示第 j 列属性的均值，$\sqrt{D(CV_{*j})}$表示第 j 列属性的标准差。通过标准

化变化，得到标准化矩阵$CV^{'}$。求解相关系数矩阵R：

$$R=\frac{CV^{'T}\times CV^{'}}{n-1}$$

计算矩阵$CV^{'T}\times CV^{'}$的特征值λ_i（即为各个属性决定权大小）和对应的特征向量。将λ_i从大到小排序，从第一项开始寻找出 n 个λ_j，使得

$$\frac{\sum_{j=1}^{n}\lambda_j}{\sum_{i=1}^{29}\lambda_i}\geq 85\%(n<29)$$

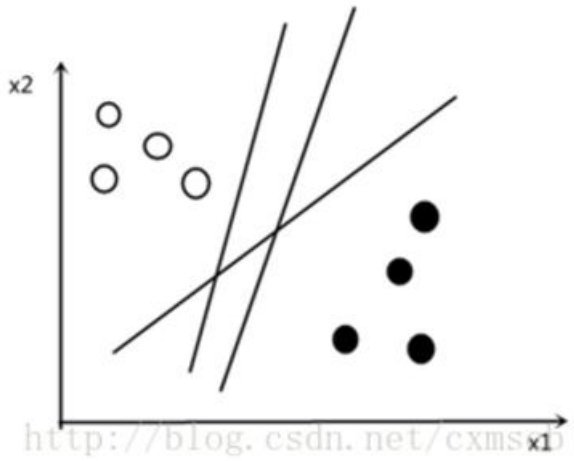
则$\lambda_1...\lambda_n$为分析得出的主成分权重，与$\lambda_1...\lambda_n$对应的$V_j, j=1,2,...,n(n<29)$为分析出的主成分属性。

5.1.2.2 RBF 支持向量机算法

根据给定的训练集

$$T=\{[CV_{1*},class_1],[CV_{2*},class_2],...,[CV_{l*},class_l]\}\in(\Omega,\ Y),$$

式中，1 为客户信息条目数，$CV_{i*}\in CV\subseteq\Omega\subseteq R^l$；$\Omega$称为输入空间，输入空间中的每一个点$CV_{i*}$由 n 个主成分属性组成，$class_i\in Y=\{-1,1\}, i=1,...,l$。下面寻找$R^n$的一个实值函数$g(CV_{i*})$，以便用分类函数



图一 分类

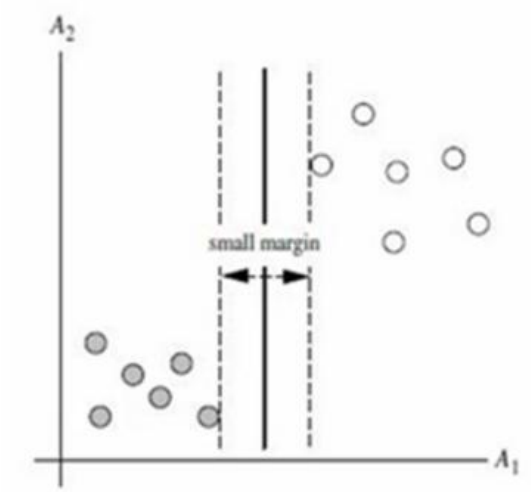
$$f(CV_{i*})=\mathrm{sgn}(g(CV_{i*}))$$

推断任意一个客户的个人信息集合 CV_{i*} 相对应的 $class_i$ 值的问题为分类问题。此分类方式即为支持向量机分类方式，此分类函数即为超平面。下面给出结合本文定义的规范超平面的定义：

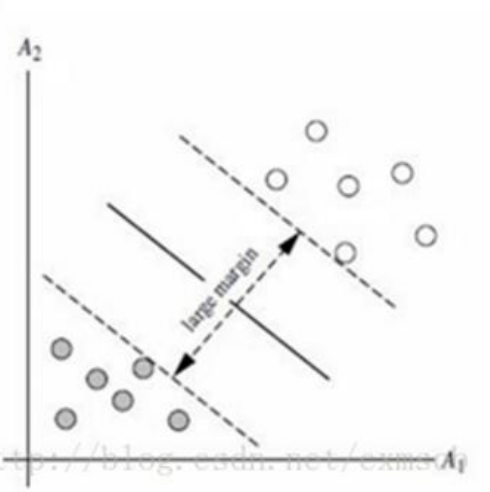
空间 R^n 中超平面都可以写为 $(\omega \times CV_{i*}) + b = 0$ 的形式，参数 (ω, b) 乘以任意一个非零常数后得到的是同一个超平面，定义满足条件

$$\begin{cases} class_i \times [(\omega \times CV_{i*}) + b] \geq 0, \\ \min\limits_{i=1,...,l} |\omega \times CV_{i*} + b| = 1 \end{cases} \quad i = 1,...,l$$

的超平面为训练集 T 的规范超平面。



图二 超平面



图三 规范超平面

定义满足 $(\omega \times CV_{i*}) + b = \pm 1$ 成立的 a 称为普通支持向量，普通支持向量间的间隔为 $\frac{2}{\|\omega\|}$。最有超平面即意味着最大化 $\frac{2}{\|\omega\|}$，$(\omega \times CV_{i*}) + b = \pm 1$ 称为分类边界，于是寻找最优超平面的问题可以转化为如下的二次规划问题：

$$\min \quad \frac{1}{2}\|\omega\|^2,$$

$$s.t. \quad y_i[\omega \cdot a_i + b] \geq 1, i = 1,...,l。$$

该问题的特点是目标函数 $\frac{1}{2}\|\omega\|^2$ 是 ω 的凸函数，并且约束条件都是线性的。

引入 *Lagrange* 函数

$$L(\omega, b, \alpha) = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{l} \alpha_i \{1 - y_i[(\omega \cdot \alpha_i) + b]\},$$

式中：$\alpha=[\alpha_1,...,\alpha_l]^T\in R^{l+}$为*Lagrange*乘子。

根据对偶的定义，通过对原问题中各变量的偏导置零，得

$$\frac{\partial L}{\partial \omega}=0\Rightarrow \omega=\sum_{i=1}^{l}\alpha_i y_i a_i\ ,$$

$$\frac{\partial L}{\partial b}=0\Rightarrow \sum_{i=1}^{l}\alpha_i y_i=0\ ,$$

代入*Lagrange*函数化为原问题的*Lagrange*对偶问题：

$$\max_{\alpha}\ -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_i y_j\alpha_i\alpha_j(a_i\cdot a_j)+\sum_{i=1}^{l}\alpha_i\ ,$$

$$s.t.\begin{cases}\sum_{i=1}^{l}y_i\alpha_i=0,\\ \alpha_i\geq 0,i=1,...,l。\end{cases}$$

求解上述最优化问题，得到最优解$\alpha^*=[\alpha_1^*,...,\alpha_l^*]^T$，计算

$$\omega^*=\sum_{i=1}^{l}\alpha_i^* y_i a_i\ ,$$

由*KKT*互补条件知

$$\alpha_i^*\{1-y_i[\omega^*\cdot a_i+b^*]\}=0,$$

可得只有当a_i为支持向量的时候，对应的α_i^*才为正，否则皆为0。选择α_i^*的一个正分量α_j^*，并以此计算

$$b^*=y_i-\sum_{i=1}^{l}y_i\alpha_i^*(a_i\cdot a_j),$$

于是构造分类超平面$(\omega^*\cdot x)+b^*=0$，并由此求得决策函数

$$g(CV_{i*})=\sum_{i=1}^{l}\alpha_i^* y_i(a_i\times CV_{i*})+b^*,$$

得到分类函数

$$f(CV_{i*})=\mathrm{sgn}\left[\sum_{i=1}^{l}\alpha_i^* y_i(a_i\times CV_{i*})+b^*\right],$$

从而对未知样本进行分类。

5.1.2.3 神经网络分析算法

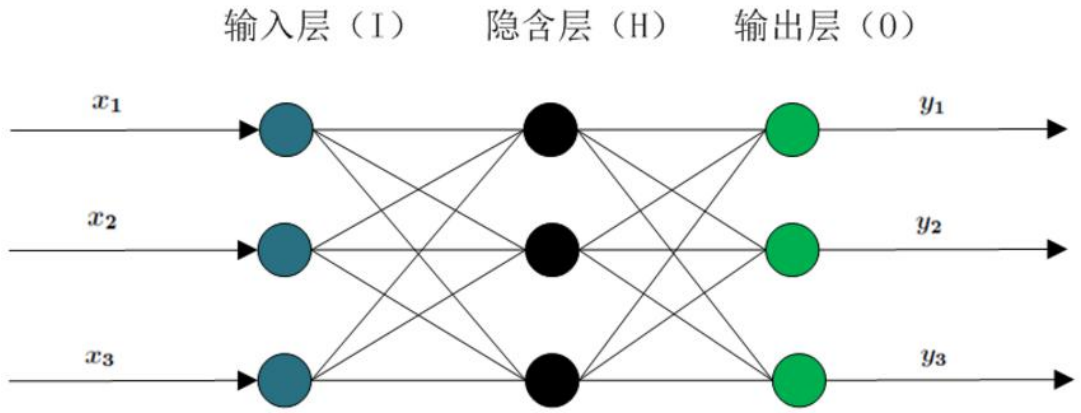
本论文主要应用的神经网络分支是 BP(Back Propagation)神经网络，下面对 BP 神经网络算法进行阐述。

BP 神经网络分为两个过程：

（1）工作信号正向传递子过程

（2）误差信号反向传递子过程

在 BP 神经网络中，单个样本有 m 个输入，有 n 个输出，在输入层和输出层之间通常还有若干个隐含层。实际上，1989 年 Robert Hecht - Nielsen 证明了对于任何闭区间的一个连续函数都可以用一个隐含层的 BP 网络来逼近，即万能逼近定理。所以一个三层的 BP 网络就可以完成任意的 m 维到 n 维的映射。即这三层分别是输入层（I），隐含层（H），输出层（O）。如图所示：



图四 神经网络结构示意图

（3）工作信号正向传递子过程

现在设节点 i 和节点 j 之间的权值为 w_{ij}，节点 b_j 的阀值为 b_j，每个节点的输出值为 x_j，而每个节点的输出值是根据上层所有节点的输出值、当前节点与上一层所有节点的权值和当前节点的阀值还有激活函数来实现的。具体计算方法如下

$$S_j = \sum_{i=0}^{m-1} w_{ij} x_i + b_j$$

$$x_j = f(S_j)$$

其中 f 为激活函数，一般选取 s 型函数或者线性函数。

正向传递的过程比较简单，按照上述公式计算即可。在 BP 神经网络中，输入层节点没有阈值。

（4）反向传递子过程

在 BP 神经网络中，误差信号反向传递子过程比较复杂，它是基于 Widrow-Hoff

学习规则的。假设输出层的所有结果为d_j，误差函数如下

$$E(w,b)=\frac{1}{2}\sum_{j=0}^{n-1}\left(d_j-y_j\right)^2$$

而 BP 神经网络的主要目的是反复修正权值和阀值，使得误差函数值达到最小。Widrow-Hoff 学习规则是通过沿着相对误差平方和的最速下降方向，连续调整网络的权值和阀值，根据梯度下降法，权值矢量的修正正比于当前位置上 E(w,b)的梯度，对于第 j 个输出节点有

$$\Delta w(i,j)=-\eta\frac{\partial E(w,b)}{\partial w(i,j)}$$

假设选择激活函数为

$$f(x)=\frac{A}{1+e^{-\frac{\beta}{B}}}$$

对激活函数求导，得到

$$f'(x)=\frac{Ae^{-\frac{\beta}{B}}}{B\left(1+e^{-\frac{\beta}{B}}\right)^2}=\frac{f(x)[A-f(x)]}{AB}$$

那么接下来针对w_{ij}有

$$\frac{\partial E(w,b)}{\partial wij}=\left(d_j-y_i\right).\frac{f(S_j)[A-f(S_j)]}{AB}\cdot x_i=\delta_{ij}\cdot x_i$$

同样，对于b_j，有

$$\frac{\partial E(w,b)}{\partial b_j}=\delta_{ij}$$

这就是著名的δ学习规则，通过改变神经元之间的连接权值来减少系统实际输出和期望输出的误差，这个规则又叫做 Widrow-Hoff 学习规则或者纠错学习规则。

上面是对隐含层和输出层之间的权值和输出层的阀值计算调整量，而针对输入层和隐含层和隐含层的阀值调整量的计算更为复杂。假设w_{ki}是输入层第 k 个节点和隐含层第 i 个节点之间的权值，那么有

$$\frac{\partial E(w,b)}{\partial w_{ki}}=x_k\cdot\sum_{j=0}^{n-1}\delta_{ij}\cdot w_{ij}\cdot\frac{f(S_i)[A-f(S_i)]}{AB}=\delta_{ki}\cdot x_k$$

$$\delta_{ki}=\sum_{j=0}^{n-1}\delta_{ij}\cdot w_{ij}\cdot\frac{f(S_i)[A-f(S_i)]}{AB}$$

有了上述公式，根据梯度下降法，那么对于隐含层和输出层之间的权值和阀值调整如下

$$w_{ij}=w_{ij}-\eta_1\cdot\frac{\partial E(w,b)}{\partial w_{ij}}=w_{ij}-\eta_1\cdot\delta_{ij}\cdot x_i$$

$$b_j=b_j-\eta_2\cdot\frac{\partial E(w,b)}{\partial b_j}=b_j-\eta_2\cdot\delta_{ij}$$

而对于输入层和隐含层之间的权值和阈值调整同样有

$$w_{ki}=w_{ki}-\eta_1\cdot\frac{\partial E(w,b)}{\partial w_{ki}}=w_{ki}-\eta_1\cdot\delta_{ki}\cdot x_k$$

$$b_i=b_i-\eta_2\cdot\frac{\partial E(w,b)}{\partial b_i}=b_i-\eta_2\cdot\delta_{ki}$$

通过将 29 个属性作为参数输入输入层，真实信用额度作为结果输入输出层，通过神经网络网络多次学习调节不同节点的权重和阈值，形成完整的神经网络图，最终逼近真实的结果。

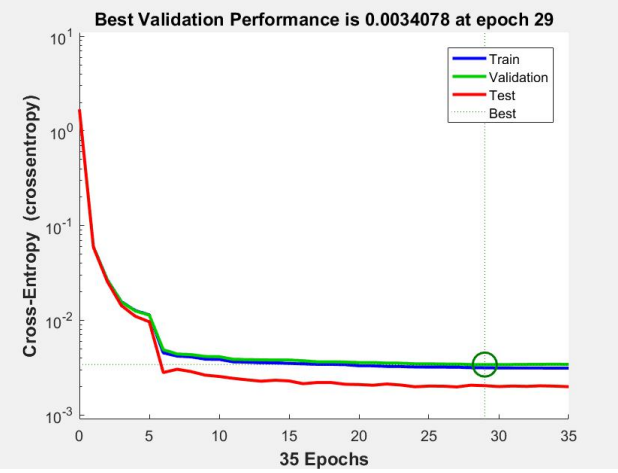
5.1.3 模型求解

5.1.3.1 神经网络分析违约可能性结果

利用神经网络，将 29 个客户属性作为参数输入集，将实际违约结果作为参数作为输出集，根据纠错学习规则，不断调整神经网络节点中的权重值和阈值，使得输入集最终拟合成输出集。即通过比较输出结果与阈值的大小对客户违约的可能性进行分类判断直接做出违约的决策结果。

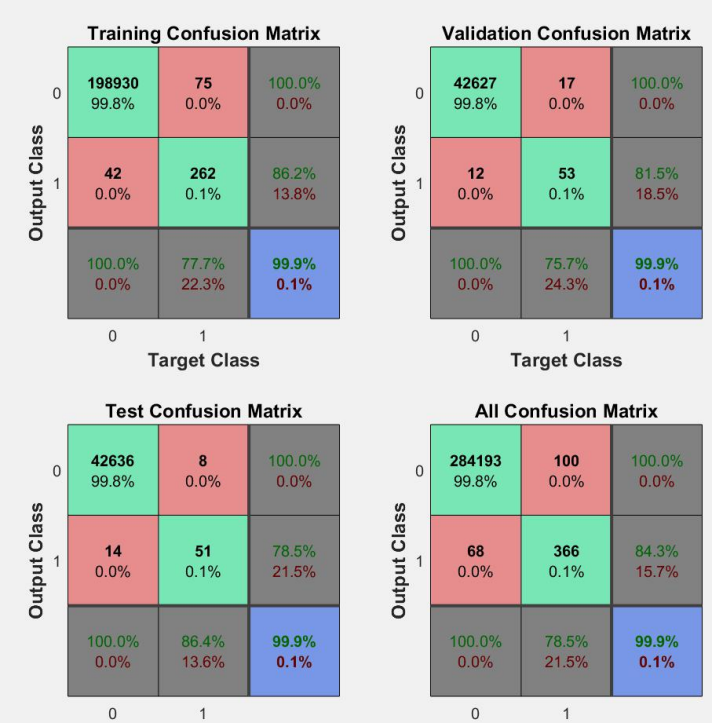
用所有数据训练的神经网络，该网络在测试集上的准确率 99.9%以上。

训练这个神经网络一共使用了 284727 个数据，其中 70%作为训练集，15%作为验证集，15%作为测试集，



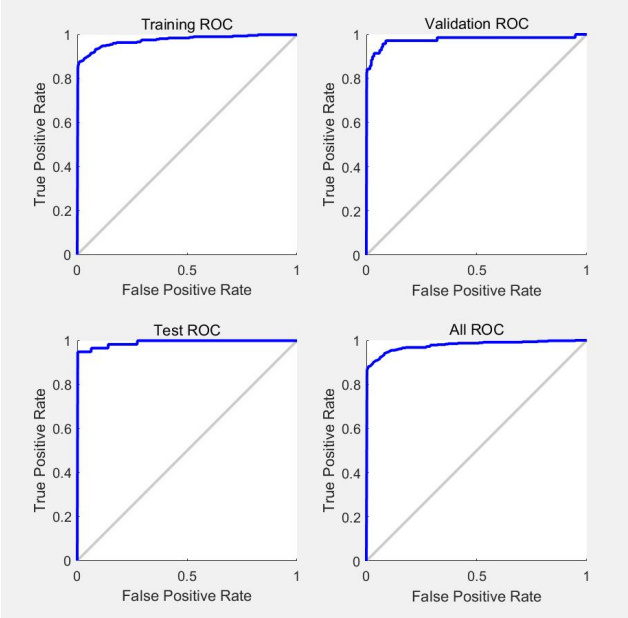
图五 训练过程中的交叉熵变化

图五是训练过程中评价函数的值的变化，值越小，表示误差越小，可以看到我们的神经网络一共训练了 35epoch，在第 29epoch 的时候到达最好的效果，此时的交叉熵为 0.0034078。



图六 混淆矩阵

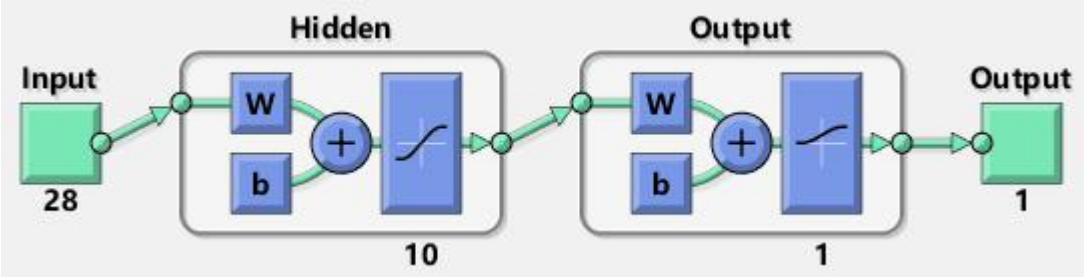
图六是混淆矩阵，可以看出，我们在测试集上的准确率为 99.9%以上，同时可以看出，我们的神经网络具有非常小的概率（小于 0.01%）把正常客户预测为违约客户，我们有 13.6%的概率把违约客户预测为正常客户。



图七 ROC 矩阵

图七是模型的 ROC 曲线，可以看出性能较好。

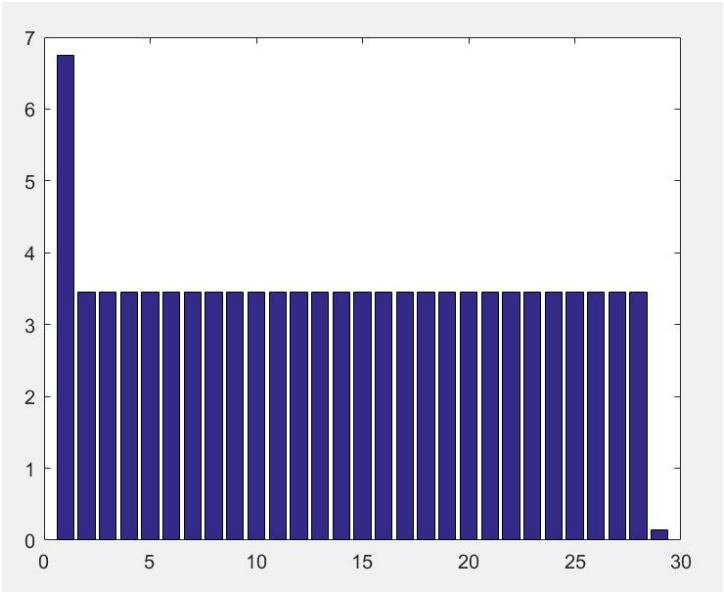
预测信用额度的神经网络结构如下：



图八 预测信用额度的神经网络结构图

5.1.3.2 主成分分析与支持向量机预测 class 未知客户违约结果

根据输入的用户信息属性数据，通过主成分分析，得出如图九结果，找出 24 个主成分并实现了降维效果（原先属性有 29 项）



横坐标：主成分排序序号，纵坐标：主成分贡献值

图九 主成分分析结果

主成分表

权重值排序	主成分
1	授信额度
2	婚姻状况
3	卡均使用率
4	经营合同风险性质
5	同意交易对手频次
6	交易对手数量
7	利息保障倍数
8	毛利率
9	退租率
10	出租率
11	月均还款占比
12	月均银行账户资金留存

13	年龄
14	固定资产
15	大额进出额交易频率
16	贷款类查询记录（含本人）
17	合伙人数量
18	第三方征信风险得分
19	申请人公检法记录
20	户籍所在地
21	申请人占股
22	近 5 年内贷款逾期次数
23	有效信用卡数
24	新开金融类账户数量

表一 主成分排序表

根据分析出的 24 个主成分，立用支持向量机计算出的 80 个 class 未知客户的违约结果如下。（注：0 表示不违约，1 表示违约）

ID	Class	ID	Class
U01	1	U41	1
U02	0	U42	0
U03	0	U43	0
U04	0	U44	0
U05	0	U45	0
U06	1	U46	0
U07	0	U47	0
U08	0	U48	0
U09	1	U49	0
U10	1	U50	0
U11	0	U51	0
U12	1	U52	0
U13	0	U53	0
U14	0	U54	1
U15	1	U55	0
U16	1	U56	0
U17	1	U57	0
U18	0	U58	0
U19	0	U59	0
U20	1	U60	0

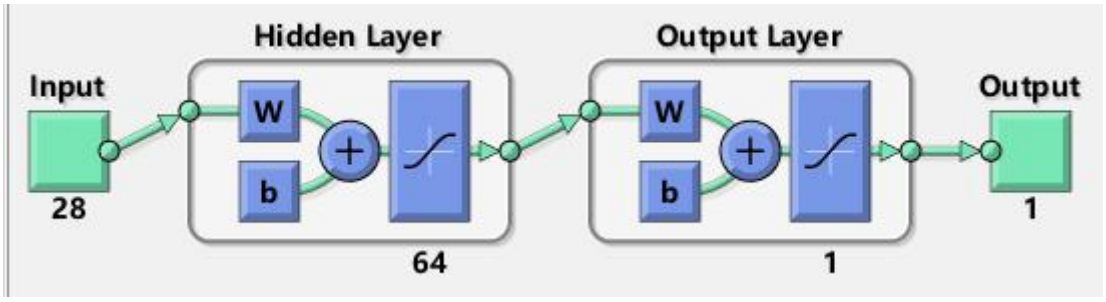
U21	1	U61	0
U22	0	U62	1
U23	1	U63	1
U24	1	U64	0
U25	0	U65	0
U26	0	U66	0
U27	0	U67	0
U28	0	U68	1
U29	0	U69	0
U30	0	U70	1
U31	0	U71	0
U32	0	U72	1
U33	1	U73	0
U34	0	U74	0
U35	0	U75	0
U36	1	U76	0
U37	0	U77	0
U38	0	U78	0
U39	0	U79	1
U40	0	U80	0

表二 class 未知客户的违约判断

5.1.3.3 BP 神经网络进行授信额度估算

为了实现对授信额度的估算，我们训练了用于回归的神经网络，根据已有数据估算授信额度。

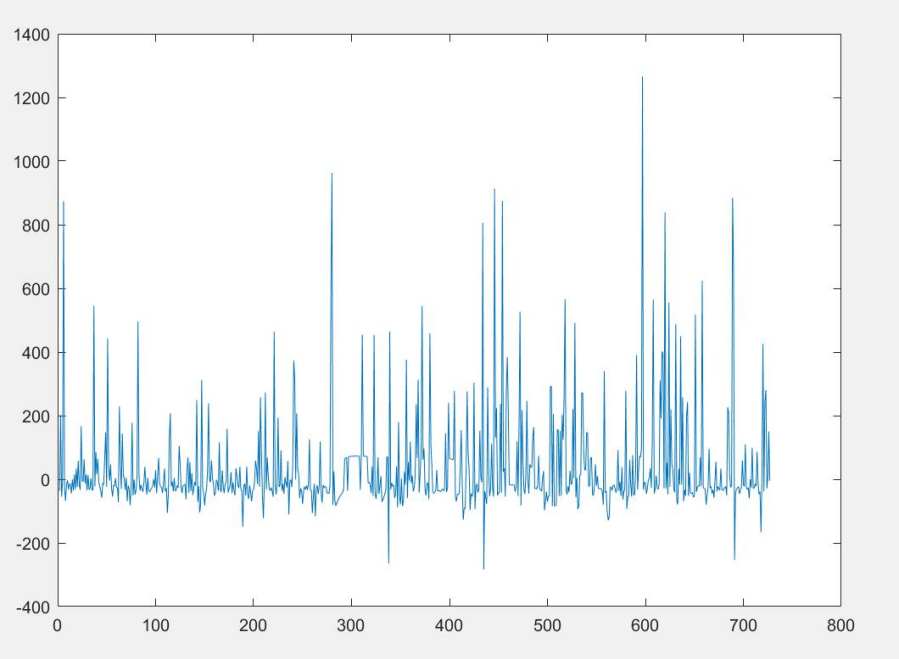
BP（Back Propagation）神经网络传输函数为 *TANSIG* 函数；训练函数为 *TRAINLM* 函数；学习规则为 *LEARNGDM* ；所使用的误差评判标准为 *MSE* 均方误差。神经网络的模型结构和参数如下图所示：



图十 判断是否违约的神经网络结构图

为了对所建立的神经网络模型准确度进行验证，我们划分出所提供的数据集

中的最后 727 条数据进行预测。对于这些数据的误差可视化结构如下图所示，经计算真实值和预测值之间的 *MSE* 误差（均方误差）为 33.659760694303344。



横坐标：测试数据序号，纵坐标：预测值与真实值之间的差值

图十一 神经网络图预测与真实情况误差统计

5.2 基于神经网络的数据残缺情况下客户放贷决策模型（授信额度估算和违约可能性预测）

5.2.1 模型建立

针对数据残缺的情况，根据“将待解决的问题转换为已解决问题”的数学思想，即考虑使用数据完整的情况中神经网络作为模板解决数据残缺情况下客户放贷决策问题。考虑残缺数据项的客户应属于完整数据集合的整体分布，故考虑采用某种方式（即后面提到的然卡方式）填补缺省数据项，使得填补后的数据客户条目项仍属于完整分布。

接着，可考虑使用神经网络模型同时解决放贷决策和违约可能性大小。即将 28 项个人数据集合作为输入集，将 class 和个人信用额度作为输出集，通过神经网络中纠错学习规则不断修改网络中结点的权重值和阈值，使得输入集通过神经网络的输出吻合输出集结果。至此，数据残缺情况下的客户放贷决策模型已经建立，即上文提到的数据完整情况下的用于预测信用额度和预测违约的神经网络。输入含残缺项的客户条目，即可同时获得放贷决策和违约可能性大小。

5.2.2 模型求解算法

5.2.2.1 热卡填补残缺值算法

热卡填充原理：对于一个包含空值的对象，热卡填充法在完整数据中找到一个与它最相似的对象，然后用这个相似对象的值来进行填充。

在缺少数据项的客户条目 C_{i*} 中，假设缺失属性为 C_{ij}，先用缺失属性值所在列的完整数据集的平均值暂时填充得到条目 $C_{i*}^{'}$，根据热卡填充原理，需要寻找

一个总体水平和残缺条目最相近的条目$C_{j*}(i \neq j)$，并用C_{j*}中对应C_{i*}的缺失属性填补，从而得到了用热卡方法填补的数据条目$C_{i*}^{''}$，以用于神经网络分析计算。

总体水平最相近条目的算法定义如下：

对其余的所有完整条目，计算与暂时用均值填补的条目$C_{i*}^{'}$的均方误差和达到最小的数据条目即可，即

对完整条目C_{j*}，$\Delta = \sum_{k=1}^{28}(C_{ik}^{'} - C_{jk})^2$，寻找到对应$\Delta$达到最小的条目$C_{j*}$，即为与残缺条目总体水平最相近的条目。然后填充$C_{j*}$中的对应缺失数据项至$C_{i*}$中即可。

5.2.2.2 残缺值填补性能评估算法

为了进一步验证上述算法在填补残缺值时的性能，我们提出了一种评估算法。

该评估算法建立在以下三条假设的基础上：

假设 1：所有属性值所占权重相等，即：

$$w(V_1) = w(V_2) = ... = w(V_{28})$$

假设 2：缺省一组属性值所占权重，等于组内各个属性权重值之和，即：

$$\text{设属性集} G = V_i + V_j + ... + V_k (i \neq j \neq k)$$

$$\text{则有} w(G) = w(V_i) + w(V_j) + ... + w(V_k)$$

假设 3：缺省属性的个数按照占总属性个数的百分比分为 5%、10%、20%三档

假设有缺省 5%属性的数据a组，缺省 10%属性的数据b组，缺省 30%属性的数据 c 组

缺省 5%属性的填充均方差

$$MSE_1 = \frac{\sum_{i=1}^{a}(V_{pj} - V_{zj})^2}{a}$$

缺省 10%属性的填充均方差

$$MSE_2 = \frac{\sum_{i=1}^{b}(V_{pj} - V_{zj})^2}{b}$$

缺省 20%属性的填充均方差

$$MSE_3 = \frac{\sum_{i=1}^{c}(V_{pj} - V_{zj})^2}{c}$$

总体数据填充均方差

$$MSE = w_1 \times MSE_1 + w_2 \times MSE_2 + w_3 \times MSE_3$$

$$w_1 = \frac{1}{7}; w_2 = \frac{2}{7}; w_3 = \frac{4}{7}$$

注：V_{pj} 指测试数据中某一行有属性残缺的客户数据中残缺的第 j 个属性，V_{zj} 指与填充值对应的真实值。

5.2.2.3 结合热卡算法的神经网络分析算法

由于本文 5.1.2.3 和 5.2.2.1 两小节分别对 bp 神经网络算法和热卡算法的介绍，下面结合本章节需解决的处理存在数据缺失值的数据项的分析方法。

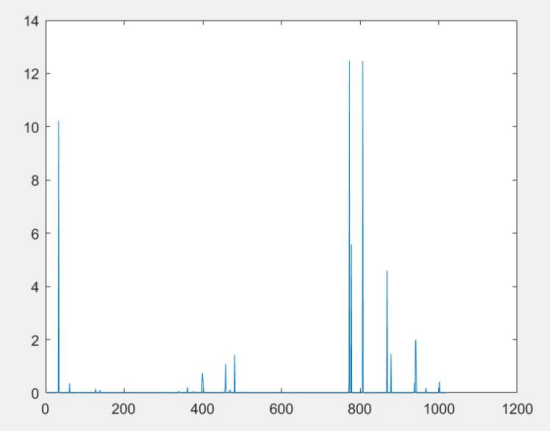
即先利用热卡填补算法填补出与原始数据基本吻合的数据项 $C_{i*}^{''}$，然后利用将其视作数据完整的客户条目数据项，并将其中的对应元素与原始完整条目数据集合一起对应放入 bp 神经网络的输入层和输出层，通过正向传递和反向传递两个子过程，结合 Widrow-Hoff 学习规则得出完整的神经网络结构，并对其进行信用额度和违约可能的决策预判。

5.2.3 模型求解

根据数据完整的客户情况，训练神经网络。具体的神经网络训练情况参见上文 5.1.3.1 模型求解部分。

在本题中我们将解决用户提供数据不完整的情况下对用户授信额度的估算。我们采用热卡填充的算法，寻找“最接近”的数据填入残缺数据项。接下来是我们对热卡填充的准确程度的验证。验证过程分为三个部分：缺失 5%数据、缺失 10%数据和缺失 20%数据。

1）共 272×28 个数据，缺失 5%数据时，随机缺失值 1019 个，经过填补再计算和原来值之间的差值的平方得到下图：

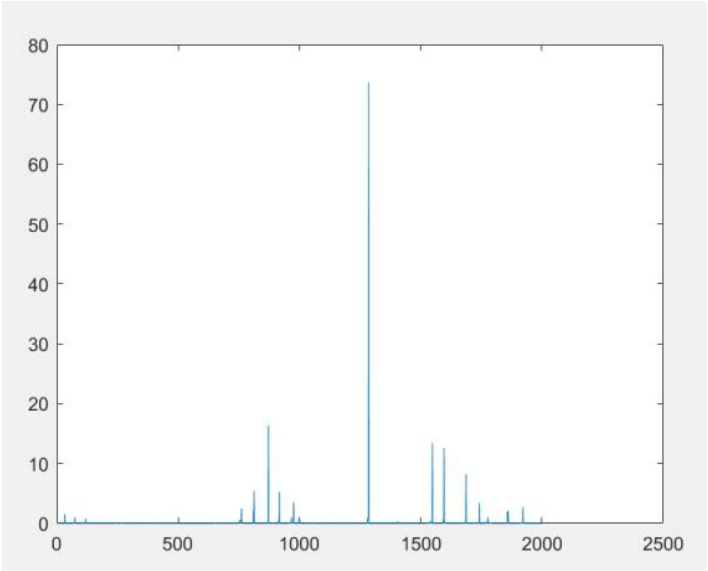


横坐标：缺失值序号，纵坐标：预测值与真实值之间差的平方

图十二 缺失 5%情况下热卡填充效果

经过计算，计算出其均方误差 $MSE_1 = 0.058605618840858$，可见此时热卡填充效果较好。

2）缺失 10%数据时。共 272×28 个数据，随机缺失值 1019 个，经过填补再计算和原来值之间的差值的平方得到下图：

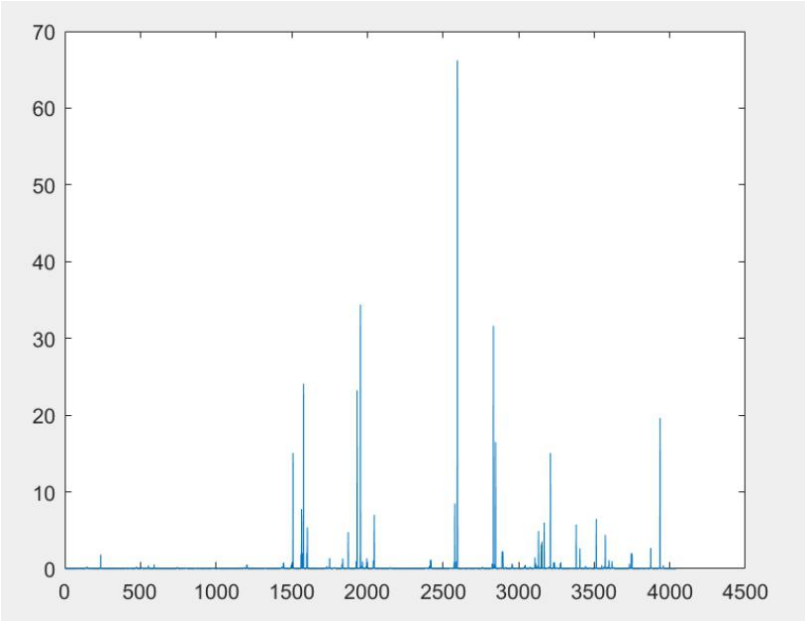


横坐标：缺失值序号，纵坐标：预测值与真实值之间差的平方

图十三 缺失 10%情况下热卡填充效果

经过计算，计算出其均方误差 $MSE_2 = 0.094416578795544$，可见此时热卡填充效果较好。

3）缺失 20%数据时。共 272×28 个数据，随机缺失值 1019 个，经过填补再计算和原来值之间的差值的平方得到下图：



横坐标：缺失值序号，纵坐标：预测值与真实值之间差的平方

图十四 缺失 20%情况下热卡填充效果

经过计算，计算出其均方误差 $MSE_3 = 0.123739775345780$, 此时热卡填充随着缺失值的增大而增加。

$$MSE = w_1 \times MSE_1 + w_2 \times MSE_2 + w_3 \times MSE_3 \approx 0.084843774331643$$

经过如上分析，我们可以得出热卡填充方式的均方误差相对较小，且随着缺失量的增多均方差有变大的趋势的结论，这与我们日常生活中的认知较为符合。

六、模型评价与改进

6.1 模型的优点

1、模型 5.1 中使用了主成分分析的方法，实现了数据降维，提高运算的速率。经验证，对客户是否违约的划分准确度达到 99%以上，具有较高的准确程度。
2、经验证，模型 5.1 中对数据完整情况下对授信额度估算的准确率较高。
3、模型 5.2 中使用热卡方法填充缺失值，该算法运算效率较高、复杂度不高且鲁棒性强。经验证，热卡方式填充准确度很高，具体数据可见 5.2.3。

6.2 模型的缺点

1、模型 5.1 中，对授信额度进行估算时，经验证在特殊情况下估算偏差较大。
2、用于预测在数据不完整情况下违约情况的数据样本正负样本比例差距大。这导致在模型 5.2 中存在一定可能性将违约用户判定为不违约用户（约有 13%的概率）
3、模型 5.2 中采用热卡填充的方式，难以确定相似标准，受主观因素影响较大。同时依赖大量数据，否则影响填充的准确度。

6.3 模型的改进方向

1、对于模型 5.1 中存在的偏差较大的异常值影响，可以用剔除或其他填补方法对个别数据进行处理。
2、对于数据中存在的违约正负样本比例差距大的问题，可以进一步进行数据增强从而实现样本类别比例平衡。
3、对于模型 5.2 中采用的热卡填充的缺陷，可以使用基于朴素贝叶斯的缺失填补方法。

七、参考文献

[1] 姜启源，数学建模（第四版），高等教育出版社，2011.1
[2] 徐全智，杨晋浩，数学建模（第二版），高等教育出版社，2008.6
[3] 司守奎，孙兆亮，数学建模算法与应用（第二版），国防工业出版社，2017.4
[4] 司守奎，数学建模算法与应用习题解答（第二版），国防工业出版社，2017.4
[5] Flexible Imputation of Missing Data (Chapman & Hall/CRC Interdisciplinary Statistics), Stef van Buuren
[6] 电子科技大学数学科学学院，数学实验方法，中国铁道出版社，2013.2

[7] 卓金武，MATLAB 在数学建模中的应用（第二版），北京航空航天大学出版社，2014.9
[8] 潘志成，带有缺失值的多元统计分析研究，2011.5

八、附件清单

附件 1 支持向量机（SVM）相关代码（classify.m）
附件 2 热卡填补缺失值相关代码（missing_value.m）
附件 3 神经网络相关代码（neural_network.m）
附件 4 预测 80 个 class 未知客户的决策情况（predict.xlsx）
附件 5 成分矩阵
附件 6 碎石图
附件 7 主成分排序
附件 8 总方差矩阵

九、附件

附件 1 支持向量机（SVM）相关代码（classify.m）

```
% 此文件针对问题 1 2，用于预测是否违约
filename = 'C:\\Users\\DELL\\Desktop\\2018 年数模校内赛题目\\2018 年数模校内赛题目\\问题 B： 不完全对称信息下的客户授信评估问题\\creditdata.xlsx';
sheet = 1;
xlRange='C2:AF284808'; %AF 后还应加上行数 训练数据共有 284726 个
data = xlsread(filename, sheet, xlRange);
pca_data = data(:,1:29); % 这个数据用于进行 pca，其中 test 数据()同样被 pca，这样方便预测

classify_label=data(1:284727,30); %0 1 需要变成 1 -1 test_data 是从 284728-284807
pos=classify_label==0;
classify_label(pos)=-1;% 将 0 变为-1
%regress_label = train_label(1); %amount

% 下面对数据进行保存，这样下一次读取就直接从 mat 格式数据读取了
%save pca_data.mat train_data
%save classify_label.mat classify_label
%save regress_label.mat regress_label
%save test_data.mat test_data

% 若已经保存，则直接读取
%train_data=load('train_data.mat');
```

```
%classify_label=load('classify_label.mat');
%regress_label=load('regress_label.mat');
%test_data=load('test_data.mat');

stdr = std(pca_data); % 得到标准差
[n, m] = size(pca_data); % 矩阵的行与列
sddata=pca_data./stdr(ones(n, 1), :); %标准化
[p,princ,egenvalue]=princomp(sddata);  %调用主成分
% p 是一个 m*m 方阵，-0 每一列包含一个主成份的系数
% princ 是对主成分的打分，n*m
%p=p(:,1:3);                                %输出前 3 主成分系数；
%sc=princ(:,1:3);                           %前 3 主成分量，后面在做 SVM 的
时候就直接用这个作为输入的特征
egenvalue;                                  %相关系数矩阵的特征值，即各主
成分所占比例；
per=100*egenvalue/sum(egenvalue);          %各个主成分所占百分比；

% 接下来对数据进行一个直观的了解
% 先读取各列，即各类数值的含义,读取的时候遇到了问题，暂时用 1-28 数字来
代替每个原始特征
% V_name = 'd:\\math\\V1-V28.xlsx';
% V1_V28 = xlsread(V_name, 1); % 只包含了 V1-V28 的名称
V1_V29=1:29;
bar(V1_V29, per); % 显示柱状图，柱状图的 x 坐标是 1 到 28，值是每个原始
特征的主成份占比

% 挑选和为前 85%  前 24 个 的主成进行分析，输入到 SVM 中
[after, index_before]=sort(per);
% after 就是进行过升序排序的 per，而 index_before 就是排序后的每个元素的
在原来的向量中的下标
% after 中的每个元素不超过 1，和为 1
sum = 0;
index=[];
for i=fliplr(V1_V28), % 从后往前累加，找到刚好大于 0.85 的那个下标
    sum = sum+after(i);
    if sum>=85,
        index=i:28;
        break;
    end
end

top=index_before(index); % top 是一个向量，包含了和为前 85%占比的主成份
的下标,用这个作为 princ 的索引即可得到和为前 85%的主成份
```

```
princ = princ(:,top); % 得到和为85%以上的主成分
disp(size(princ)); % 显示行数与列数，即样本数与主成成分数

test_data=princ(284728:284807,:);
train_data=princ(1:284727,:);
% 训练SVM,train_label中的标注为1或者-1才行
%                不                适                用                了
SVMStruct=svmtrain(train_data(:,1:24),classify_label,'kernel_function
', 'rbf','rbf_sigma',1,'Showplot',true);      % train


SVMModel                                                              =
fitcsvm(train_data,classify_label,'Standardize',true,'KernelFunction',
'RBF',...
    'KernelScale','auto');
% 进行预测是否可以贷款给这个人，Group中为1表示违约客户，-1表示正常客
户
[class, score]=predict(SVMModel,test_data);
% 其中class(i)是第i个测试样本分的类别 score(i,1)是概率，当score大于
零的时候class为-1（正常客户），score小于零的时候class为1（违约客户）
% 接下来验证模型
CVSVMOdel=crossval(SVMModel);
classLoss=kfoldLoss(CVSMModel);
%classLoss=0.000544381108921811;
% 表示我们在验证集上的准确率已经达到了99.9%以上

save 'classify_model.mat' SVMModel;% 保存训练好的SVM模型

% 可以使用如下语句读取
model = cell2mat(struct2cell(load('classify_model.mat')));
% 保存到excel中
p=class==-1;
class(p)=0;
xlswrite('predict.xlsx', class);
```

附件2 热卡填补缺失值相关代码（missing_value.m）

```
% 此文件针对问题3 ，先进行缺失值填补，再进行回归预测

% 计算每行的12距离和均值并保存
[n,m]=size(train_data);
train_mean=mean(train_data);
train_l2=zeros([n,1]);
for i=1:n,
    train_mean(i)=norm(mean(train_data(i,:)), 2);
```

```
end
save 'train_mean.mat' train_mean
save 'train_norm.mat' train_l2

% 热卡填补方式
data=load('data.mat');
data=cell2mat(struct2cell(data)); % 转换成矩阵
test_data=data(284001:284727,1:28);% test_data 是用于预测的 727 个数据，其中部分数据是 nan，需要进行填补
train_mean=load('train_mean.mat'); % mean 是每一个指标的均值，当用于预测的数据中有缺失值的时候先用均值填补，然后用热卡的方式再次填补
train_mean=cell2mat(struct2cell(train_mean));
train_l2=load('train_norm.mat'); % train_l2 是计算好用于训练的数据中的每条记录（每行的 12 距离），是一个一维向量,
train_l2=cell2mat(struct2cell(train_l2));
train_data=data(1:284000,1:28); % 284001-284727 被用于模拟缺失值

origin_test_data=test_data;


% 找到含有 nan 的行，即找到含有 nan 的一条数据
% 使用热卡的方式进行填补
[n,m]=size(test_data);
flag=0;% 这一行，即这一条数据中是否有缺失值
k=0; % 用于统计每行中的缺失值
index=[]; % 每行中的缺失值的下标
error_count=0;% error 的下标
error=[]; % 统计填补过程中的误差（mse）
for i=1:n,
    for j=1:m,
        if isnan(test_data(i,j)),
            flag=1;
          k=k+1;
            index(k)=j;
            test_data(i,j)=train_mean(j); %先用这一列的均值填补
            % 利用热卡的方式对本条数据进行填补,先计算 L2 距离，和 28 万条数据进行对比得到最相近的那个，然后用那个的值来代替
        end
    end
    if flag==1, % 在检查下一行数据之前要把 k 置为 0，index 置为空
       flag=0; % 清除 flag
        l2=norm(test_data(i,:),2);
        [x,minimum]=min(abs(train_l2-l2));% minimum 是 28 万个数据中 12 距离和这个相差最小的下标
```

```
        for z=index,
            test_data(i,z)=train_data(minimum,z);
            error_count=error_count+1;

error(error_count)=(origin_test_data(i,j)-test_data(i,j))^2;
        end
        k=0;
        index=[];
    end
end


% 下面进行验证，先产生残缺的数值，然后再填补
% 产生随机数和 test_data 的大小一致，用这个作为索引，来进行生成残缺的部分

rand_pos=rand([727,28]);% 产生 727*28 阶离散均匀分布的随机数矩阵；产生一个数值在 0-1 之间的 mm*nn 矩阵

% 缺失 10%的数据,这里的 test_data 未经过归一化
test_data=origin_test_data;
test_data(rand_pos<=0.05)=nan;

plot(1:727,error) % 预测值和真实值之间的差的平方作图
mean(error) % MSE

附件 3 神经网络相关代码（neural_network.m）
% 神经网络过程

% 读取数据
data=load('data.mat'); % 从 284262 开始 class 是 1（违约客户）
data=cell2mat(struct2cell(data)); % 转换为矩阵 284727X30
train_data=data(1:284000,1:1:28);% 用于训练回归和分类的数据 284000 个，其余的 727 个用于缺失值填补测试
regress_label=data(1:284000,29); % 用于回归的标签

% 使用全部数据训练神经网络
classify_label=data(1:284727,30);% 用于分类的标签，一共有 800 个 class 为 0 400 个 为 1 400 个
classify_data=data(1:284727,1:28);


classify_label=data(283796:284727,30); % 用于分类的标签，一共有 800 个 class 为 0 400 个 为 1 400 个
```

```
classify_data=data(283796:284727,1:28); % 用于分类的数据


% 回归训练数据进行归一化才能输入神经网络
stdr = std(train_data); % 得到标准差
[n, m] = size(train_data); % 矩阵的行与列
train_data=train_data./stdr(ones(n, 1), :); %标准化

% 对用于分类的数据进行归一化
stdr = std(classify_data); % 得到标准差
[n, m] = size(classify_data); % 矩阵的行与列
classify_data=classify_data./stdr(ones(n, 1), :); %标准化

% 进行回归训练的时候需要进行转置
train_data=train_data';
regress_label=regress_label';

% 进行分类的时候需要进行转置
classify_label=classify_label';
classify_data=classify_data';

% 下面是用于回归的测试数据
test_data=data(284001:284727,1:28); % 用于测试的 727 个数据
test_regress_label=data(284001:284727,29); % 测试额度的标签

% 下面是用于分类的测试数据
test_classify_data0=data(1:100,1:28);
test_classify_label0=data(1:100,30);
test_classify_data1=data(284663:284727,1:28);
test_classify_label1=data(284663:284727,30); % 65 个违约的用于预测

% 回归测试数据进行归一化才能输入神经网络
stdr = std(test_data); % 得到标准差
[n, m] = size(test_data); % 矩阵的行与列
test_data=test_data./stdr(ones(n, 1), :); %标准化

% 分类测试数据需要进行归一化才能输入神经网络
stdr = std(test_classify_data0); % 得到标准差
[n, m] = size(test_classify_data0); % 矩阵的行与列
test_classify_data0=test_classify_data0./stdr(ones(n, 1), :); %标准化

stdr = std(test_classify_data1); % 得到标准差
[n, m] = size(test_classify_data1); % 矩阵的行与列
test_classify_data1=test_classify_data1./stdr(ones(n, 1), :); %标准化
```

```
% 测试数据进行转置
test_data=test_data';
test_regress_label=test_regress_label';

test_classify_label0=test_classify_label0';
test_classify_data0=test_classify_data0';
test_classify_label1=test_classify_label1';
test_classify_data1=test_classify_data1';


% 建立神经网络中的选项
% Training Function：训练函数
% Adapting learning function：适应性学习函数，即梯度下降的方式
% Performance function：即 loss function，
% Number of layers：隐藏层数
% Number of neurons：隐藏层神经元
% Transfer function：激活函数 transig（双极性函数 BP 一般使用） LOGSIG（逻辑斯蒂曲线）
% 生成神经网络
% 这个网络一共有四层 每层的激活函数都是 tansig（双曲正切 S 型函数），用于训练的是 traingd 梯度下降法

% 验证回归（信用额度） test_data 是已经归一化的结果
net=load('regress_net');
perf=mse(net,test_data,test_regress_label);

% 验证分类（是否违约） test_data 是已经归一化的结果
net=load('classifier_net');
pred=sim(net,test_data);
```

附件 4 预测 80 个 class 未知客户的决策情况（predict.xlsx）

```
% 神经网络过程

% 读取数据
data=load('data.mat'); % 从 284262 开始 class 是 1（违约客户）
data=cell2mat(struct2cell(data)); % 转换为矩阵 284727X30
train_data=data(1:284000,1:1:28);% 用于训练回归和分类的数据 284000 个，其余的 727 个用于缺失值填补测试
regress_label=data(1:284000,29); % 用于回归的标签

% 使用全部数据训练神经网络
classify_label=data(1:284727,30);% 用于分类的标签，一共有 800 个 class 为 0 400 个 为 1 400 个
```

```
classify_data=data(1:284727,1:28);


classify_label=data(283796:284727,30); % 用于分类的标签，一共有 800 个
class 为 0 400 个 为 1 400 个
classify_data=data(283796:284727,1:28); % 用于分类的数据


% 回归训练数据进行归一化才能输入神经网络
stdr = std(train_data); % 得到标准差
[n, m] = size(train_data); % 矩阵的行与列
train_data=train_data./stdr(ones(n, 1), :); %标准化

% 对用于分类的数据进行归一化
stdr = std(classify_data); % 得到标准差
[n, m] = size(classify_data); % 矩阵的行与列
classify_data=classify_data./stdr(ones(n, 1), :); %标准化

% 进行回归训练的时候需要进行转置
train_data=train_data';
regress_label=regress_label';

% 进行分类的时候需要进行转置
classify_label=classify_label';
classify_data=classify_data';

% 下面是用于回归的测试数据
test_data=data(284001:284727,1:28); % 用于测试的 727 个数据
test_regress_label=data(284001:284727,29); % 测试额度的标签

% 下面是用于分类的测试数据
test_classify_data0=data(1:100,1:28);
test_classify_label0=data(1:100,30);
test_classify_data1=data(284663:284727,1:28);
test_classify_label1=data(284663:284727,30); % 65 个违约的用于预测

% 回归测试数据进行归一化才能输入神经网络
stdr = std(test_data); % 得到标准差
[n, m] = size(test_data); % 矩阵的行与列
test_data=test_data./stdr(ones(n, 1), :); %标准化

% 分类测试数据需要进行归一化才能输入神经网络
stdr = std(test_classify_data0); % 得到标准差
[n, m] = size(test_classify_data0); % 矩阵的行与列
```

```
test_classify_data0=test_classify_data0./stdr(ones(n, 1), :); %标准化

stdr = std(test_classify_data1); % 得到标准差
[n, m] = size(test_classify_data1); % 矩阵的行与列
test_classify_data1=test_classify_data1./stdr(ones(n, 1), :); %标准化

% 测试数据进行转置
test_data=test_data';
test_regress_label=test_regress_label';

test_classify_label0=test_classify_label0';
test_classify_data0=test_classify_data0';
test_classify_label1=test_classify_label1';
test_classify_data1=test_classify_data1';


% 建立神经网络中的选项
% Training Function：训练函数
% Adapting learning function：适应性学习函数，即梯度下降的方式
% Performance function：即 loss function，
% Number of layers：隐藏层数
% Number of neurons：隐藏层神经元
% Transfer function：激活函数 transig（双极性函数 BP 一般使用） LOGSIG（逻辑斯蒂曲线）
% 生成神经网络
% 这个网络一共有四层 每层的激活函数都是 tansig（双曲正切 S 型函数），用于训练的是 traingd 梯度下降法

% 验证回归（信用额度） test_data 是已经归一化的结果
net=load('regress_net');
perf=mse(net,test_data,test_regress_label);

% 验证分类（是否违约） test_data 是已经归一化的结果
net=load('classifier_net');
pred=sim(net,test_data);
```

80 个 class 未知客户的违约结果如下。（注：0 表示不违约，1 表示违约）

ID	Class	ID	Class
U01	1	U41	1
U02	0	U42	0
U03	0	U43	0
U04	0	U44	0
U05	0	U45	0
U06	1	U46	0
U07	0	U47	0

U08	0	U48	0
U09	1	U49	0
U10	1	U50	0
U11	0	U51	0
U12	1	U52	0
U13	0	U53	0
U14	0	U54	1
U15	1	U55	0
U16	1	U56	0
U17	1	U57	0
U18	0	U58	0
U19	0	U59	0
U20	1	U60	0
U21	1	U61	0
U22	0	U62	1
U23	1	U63	1
U24	1	U64	0
U25	0	U65	0
U26	0	U66	0
U27	0	U67	0
U28	0	U68	1
U29	0	U69	0
U30	0	U70	1
U31	0	U71	0
U32	0	U72	1
U33	1	U73	0
U34	0	U74	0
U35	0	U75	0
U36	1	U76	0
U37	0	U77	0
U38	0	U78	0
U39	0	U79	1
U40	0	U80	0

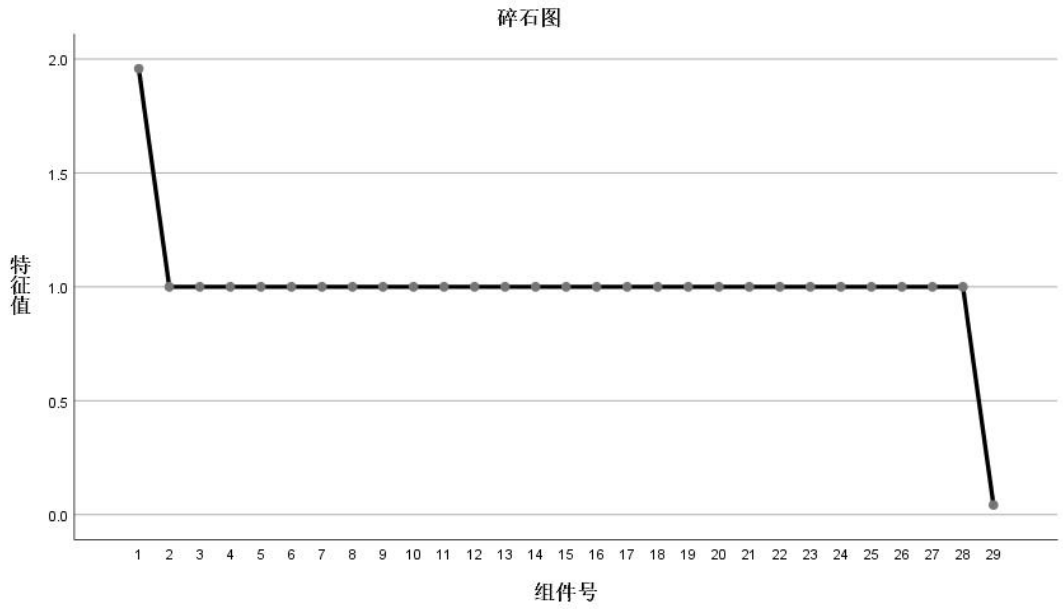
附件 5 成分矩阵

成分矩阵 [a]

	成分						
	1	2	3	4	5	6	7
Zscore(Amount)	.989	.000	.000	.000	.000	.000	.000
Zscore(V2)	-.549	.148	-.175	.209	-.178	.351	.066
Zscore(V7)	.410	.387	-.326	-.265	.052	.182	-.053
Zscore(V20)	.351	.225	-.105	.277	.171	.316	-.096

Zscore(V28)	.011	.500	.366	.094	.084	-.034	.014
Zscore(V27)	.030	-.377	.356	-.023	.226	.257	-.063
Zscore(V24)	.005	.209	.401	.176	-.137	.006	-.035
Zscore(V23)	-.116	-.076	.355	-.461	-.009	-.160	-.077
Zscore(V22)	-.067	.075	.023	.206	.401	-.083	.050
Zscore(V21)	.109	.074	.142	-.153	-.381	.165	.378
Zscore(V8)	-.106	.082	-.128	.024	-.174	-.175	-.505
Zscore(V25)	-.049	-.140	-.251	.090	.256	-.177	.348
Zscore(V3)	-.218	.188	.036	-.020	-.235	-.012	-.016
Zscore(V5)	-.399	.305	-.106	-.387	.286	-.067	.053
Zscore(V26)	-.003	-.233	-.084	.168	-.189	-.111	.168
Zscore(V11)	.000	-.057	-.108	.005	-.227	-.066	-.031
Zscore(V18)	.037	.163	.048	.250	.053	-.266	-.126
Zscore(V12)	-.010	-.120	-.200	-.204	-.031	.170	.024
Zscore(V19)	-.058	.067	.060	.236	.016	-.312	.059
Zscore(V1)	-.235	-.129	-.080	.288	.090	.218	-.251
Zscore(V16)	-.004	-.075	.142	.049	.291	.135	.013
Zscore(V9)	-.046	-.019	.074	.043	-.065	.347	.058
Zscore(V6)	.223	-.079	.141	.119	-.183	.031	.065
Zscore(V14)	.035	.005	-.084	-.071	.213	-.086	.214
Zscore(V15)	-.003	-.006	-.188	-.013	-.150	.046	-.110
Zscore(V4)	.102	-.078	-.155	-.002	.006	-.155	-.138
Zscore(V17)	.008	-.025	-.071	.135	-.061	-.206	.254
Zscore(V13)	.005	-.159	.046	-.140	.127	.126	-.337
Zscore(V10)	-.105	.079	.035	.010	.109	.241	.268

附件 6 碎石图



附件 7 主成分排序

V	中文名
amount	授信额度
2	婚姻状况
7	卡均使用率
20	经营合同风险性质
28	同意交易对手频次
27	交易对手数量
24	利息保障倍数
23	毛利率
22	退租率
21	出租率
8	月均还款占比
25	月均银行账户资金留存
3	年龄
5	固定资产
26	大额进出额交易频率
11	贷款类查询记录（含本人）
18	合伙人数量
12	第三方征信风险得分
19	申请人公检法记录
1	户籍所在地
16	申请人占股
9	近 5 年内贷款逾期次数
6	有效信用卡数
14	新开金融类账户数量
15	业务所处阶段
4	学历
17	申请人出资方式
13	贷款涉及金融机构类型
10	征信五级分类次级以上级别贷款

附件 8 总方差矩阵

总方差解释

	初始特征值			提取载荷平方和		
成分	总计	方差百分比	累积 %	总计	方差百分比	累积 %
1	1.958	6.751	6.751	1.958	6.751	6.751
2	1.000	3.448	10.199	1.000	3.448	10.199
3	1.000	3.448	13.648	1.000	3.448	13.648

4	1.000	3.448	17.096	1.000	3.448	17.096
5	1.000	3.448	20.544	1.000	3.448	20.544
6	1.000	3.448	23.993	1.000	3.448	23.993
7	1.000	3.448	27.441	1.000	3.448	27.441
8	1.000	3.448	30.889	1.000	3.448	30.889
9	1.000	3.448	34.337	1.000	3.448	34.337
10	1.000	3.448	37.786	1.000	3.448	37.786
11	1.000	3.448	41.234	1.000	3.448	41.234
12	1.000	3.448	44.682	1.000	3.448	44.682
13	1.000	3.448	48.130	1.000	3.448	48.130
14	1.000	3.448	51.579	1.000	3.448	51.579
15	1.000	3.448	55.027			
16	1.000	3.448	58.475			
17	1.000	3.448	61.924			
18	1.000	3.448	65.372			
19	1.000	3.448	68.820			
20	1.000	3.448	72.268			
21	1.000	3.448	75.717			
22	1.000	3.448	79.165			
23	1.000	3.448	82.613			
24	1.000	3.448	86.062			
25	1.000	3.448	89.510			
26	1.000	3.448	92.958			
27	1.000	3.448	96.406			
28	1.000	3.448	99.855			
29	.042	.145	100.000			