对于“拍照赚钱”应用酬金定制的研究与改进

摘 要

本文针对“拍照赚钱”类应用的酬金制度的研究与改进问题，对题目所给附件中的数据进行了大量的数据处理，以聚类分析、主因素分析、多元回归方程模拟和 BP 神经网络为理论基础进行了建模工作。

针对问题一，对题中提供的经纬度数据进行定位作图，由地理位置推测出与定价有关的相关变量，如当地人口密度以及经济、交通发展情况，查阅资料并采集数据，选取适当变量，应用多元回归方程对数据进行模拟得到定价规律；通过 SPSS 对因素进行相关性检测和聚类分析，得出任务未完成原因：定价过低、距离任务地点太远和用户密集。

针对问题二，将数据归一化后，采用神经网络模型中的 BP 模型模拟得到价格规律模型。经检验，模型符合实际规律，而且使得公司受益。

针对问题三，设计打包领取任务酬金变动规律，即适当减少酬金，利用上题的神经网络模型学习生成任务完成情况的规律，与变动前相对比，发现任务完成情况基本不变，降低价格的改动合理。

针对问题四，对数据进行处理，通过神经网络推出完成度以及价格。为评价其效果，将数据应用于第一题得到的多元回归方程模型，预测任务完成程度及价格。并将两个价格进行比较，得出结论：新建的模型在保持执行率不变的条件下，成功的降低了公司的价格成本。

关键词：定价问题、聚类分析、多元回归方程、神经网络

一、问题重述

基于移动互联网的自助式劳务众包平台“拍照赚钱”，为企业提供各种商业检查和信息搜集，节省调查成本，有效保证调查数据真实性，缩短调查周期。其 APP 中的任务定价的合理性十分重要，是商品检查的成功与否的重要因素。本文通过搜集相关数据，建立合理的数学模型研究如下问题：

问题一 研究项目的任务定价规律，分析任务未完成的原因。

问题二 为项目设计新的任务定价方案，并与原方案对比。

问题三 实际情况下，多个任务可能因为位置比较集中，导致用户争相选择。如果将这些任务联合在一起打包发布，该如何修改前面的定价模型，对最终的任务完成情况有什么影响？

问题四 针对新项目数据给出任务定价方案，并评价该方案的实施效果。

二、问题分析

2.1 问题一

该题旨在寻找任务定价规律，并分析任务未完成的原因。为寻找定价规律，首先应根据题目所给数据，结合合理的常识，找出影响定价规律的多个因素；其次，收集影响因素的详细数据，并对其做适当误差剔除处理，利用 SPSS 找出影响因子的相关系数，挑选合适的影响因子；再者，拟合所挑出的影响因子与任务价格的关系曲线，得到定价方程。

为分析任务未完成的原因，设定影响任务执行情况的众因素为潜在原因，用 SPSS 计算各个潜在原因的相关系数，并按照从大到小进行排序，然后合理对任务未完成的原因进行推测。

2.2 问题二

该题需要找到合适的定价方案，使任务的完成情况整体显著提高，结合问题一的分析，该题考虑变量多且杂，具有不确定性和模糊性，所以考虑使用具有较强自适应性的神经网络法来解决该问题。

2.3 问题三

该题要求基于新的条件，打包任务的出现，修改问题二所生成的模型，并分析新的模型对最终任务完成情况产生的影响。所以在机器学习的输入数据中加入描述打包任务

的变量，以此推断出新的定价规律。接着，为阐释新的规律对任务完成情况的影响，重复解决问题二的方法。

2.4 问题四

题目要求针对新的数据给出任务定价方案，因此基于之前对问题二、三的分析与考虑，决定采用问题二用神经网络生成的模型，纳入新的改动，即问题三提到的对于价格进行处理。将题目中所给的数据进行处理后，应用至新模型中。

三、模型假设

3.1 假设内容

(1) 假设公司在规定价格时，未考虑或未采集任务完成情况，故而任务完成情况对公司原定价无直接影响；

(2) 假设同一城市同一区人口分布是恒定的；

(3) 假设设定标价后，任务的价格不会随着其他因素进行第二次改变。

四、名词解释与符号说明

4.1 变量说明

符号	含义	单位
$l1$	任务 GPS 纬度	度
$l2$	任务 GPS 经度	度
ρ	区域内人口密度	人/平方公里
d	区域任务密集中心	个/区域
y	价格	元
$\beta_1\ \beta_2\ \beta_3$	回归方程常数系数	-
t	区域平均任务开展时间	-
b	区域内所接任务限额	个/区域
f	任务完成情况	-
μ	均值	

五、模型建立与求解

5.1 问题一 回归方程还原定价模型的建立与求解

5.1.1 问题分析及建模前的准备

(1) 研究价格变动的规律

观察题中所给任务数据，由于企业在定价时未采取到任务完成程度信息，所以定价应由经纬度及经纬度所推得的信息来决定。

将价格以不同颜色在经纬图中标出，并作出价格分布等高图（图左）；将任务地点总区域划分成数小格，用 MATLAB 算得单位面积内任务地点数，作其等高图（图中）。两图对比，不难发现，任务密集中心与价格最低中心位置几乎吻合，因此推测任务密度与定价规律有一定联系；同理推测，任务周围会员分布密度与定价相关联。（图右）且价格从中心延外逐渐升高，那么任务与任务密集中心的距离也应和定价有关。

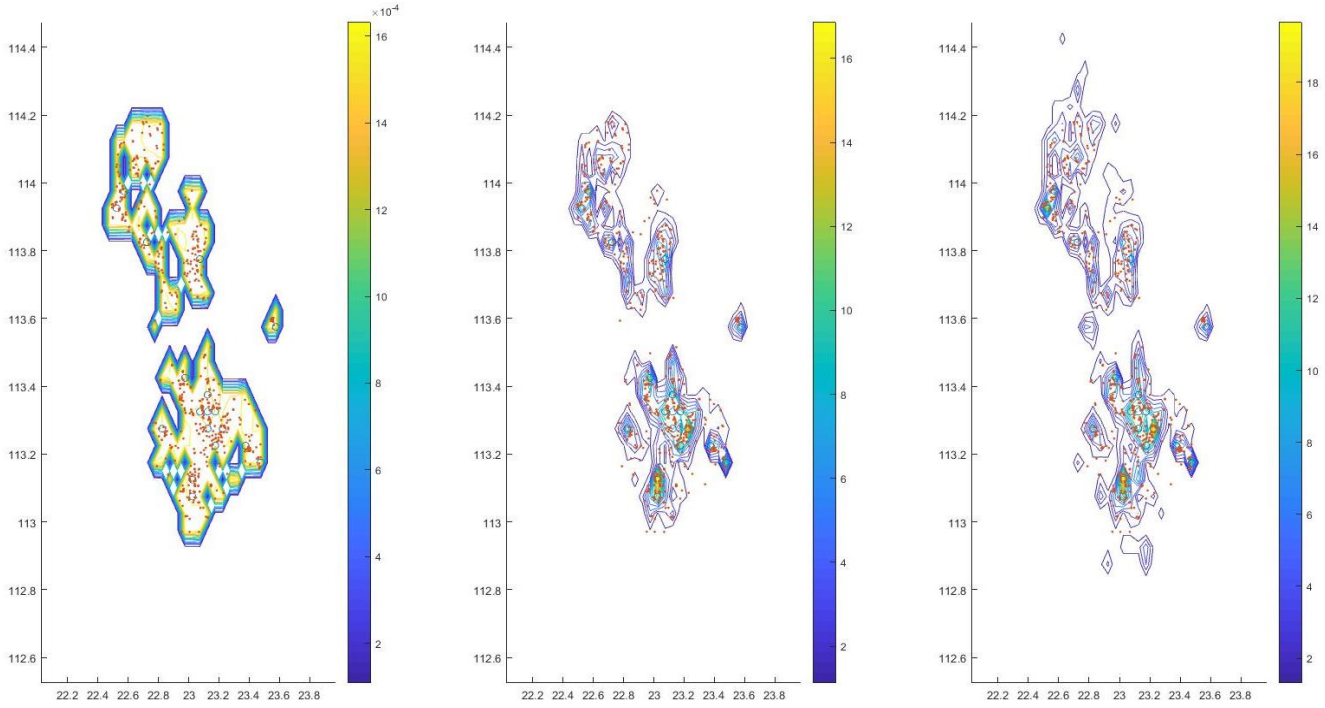

图 1 价格分布等高线 任务分布等高线 会员分布等高线

同时，根据经济学原理[1]，商品定价的高低与产品地点及地点的繁荣程度有关，而人口密度是地区繁荣程度的重要影响因素，所以也应考虑任务地点的人口密度信息。通过软件 XGeoCoding 将经纬度转化成实际的地理位置，发现任务地点分布在广东省，且分布于广州市、深圳市、佛山市、东莞市及清远市，查阅各地政府统计局资料[2-6]，得到每地人口密度。

因此据上述分析，以任务密度、会员密度、人口密度、与密度中心距离作为影响因子，利用 SPSS 找出各影响因子的相关系数；挑选影响程度较大的因子作为自变量，建立回归方程；最终对方程进行检验，判定拟合程度。

(2) 分析任务未完成的原因

考虑影响任务完成程度的因素，有价格、据中心距离、人口密度、会员密度、任务密度、区域任务限额、区域平均任务开始时间。用 SPSS 计算各个潜在原因的相关系数，并按照从大到小进行排序，然后合理对任务未完成的原因进行推测。

5.1.2 模型的建立

5.1.2.1 决策变量的考虑

为寻找定价规律，考虑的变量有：

(1) 区域内任务密度 m

(2) 区域内会员密度 v

(3) 据任务密集中心距离 d

(4) 区域内人口密度 ρ

为分析任务未完成的原因，增加考虑如下变量对任务完成程度的影响：

(5) 区域内所接任务限额 b

(6) 区域平均任务开展时间 t

(7) 价格 y

5.1.2.2 模型求解

(1) 各决策变量的计算

a. 区域内任务密度 m 及区域内会员密度 v

以每个任务点为中心，半径为 r 的圆作为单位区域，即任务的覆盖区域。通过 MATLAB 对每个任务点、每个会员信息进行遍历，对落在该任务区域的点做计数，得到单位区域内任务数量作为区域内任务密度 m，单位区域内会员数量作为会员密度 v。

b. 据任务密集中心距离 d

利用 MATLAB 中用于聚类的 KMEANS 函数，对各任务的经纬度进行分簇，并得到 5 个簇密度中心，即任务密度中心。函数同时生成各任务与密度中心的距离d，即第 i 个任务到第 j 个密度中心的距离d_{ij}：

$$d_{ij} = \sqrt{(110.94 \times (l1_i - l1_j))^2 + (85.276 \times (l2_i - l2_j))^2}$$

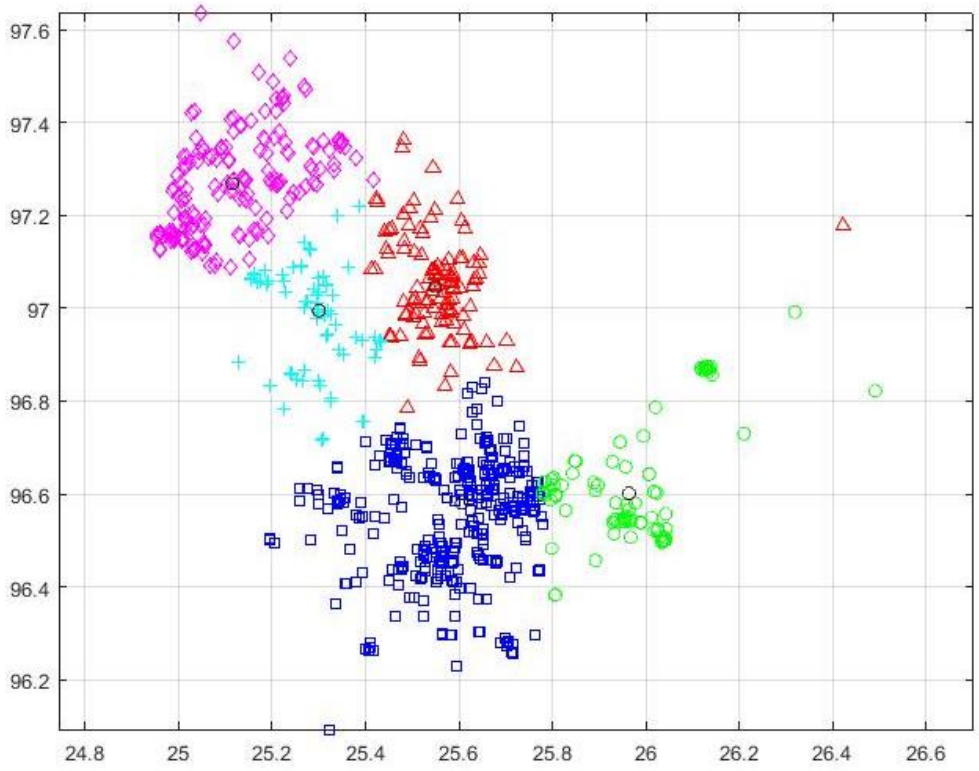

图 2　按经纬度分簇后的任务地点

c. 区域内所接任务限额 b

同 a 中方法，计算落在单位区域内会员预定任务限额总和。

d. 区域平均任务开展时间 t

同 a、c 中方法，计算落在单位区域内会员预定任务开始时间的均值。

(2) 数据处理

a. 预处理：对每个决策变量进行统计之后，考虑到数据为样本量大、影响因素众多的真实数据样本，为保证数据的可靠性，应依次对每类数据进行如下预处理:

利用截割法处理，对数据分别剔除 10 个最大值与最小值；

根据统计学原理拉依达法则，舍去与数据的算术平均值偏差绝对值大于 2 倍偏差的可疑距离，以保留 95%的置信区间，保留部分满足以下条件的数据：

$$|h-\mu| \leq 2\sigma$$

b. 均值计算：同一价格，以每类数据的均值作为最终数据进行分析。

(3) 决策变量对定价的影响程度

利用 SPSS 对每个决策变量进行分析，得出每个决策变量之间相关系数。由表一可见，各决策变量虽对彼此有影响，但互相不可替代；接着用最大系数法对这 4 个决策变量进行系统聚类，发现距离、人口密度为定价主要影响因素，因此考虑用这两个变量来拟合定价曲线。

Proximity Matrix

Matrix File Input

Case	距离	人口密度	会员密度	任务密度
距离	1.000	-.217	-.478	-.424
人口密度	-.217	1.000	.314	.147
会员密度	-.478	.314	1.000	.756
任务密度	-.424	.147	.756	1.000

表一 影响价格的决策变量相关系数

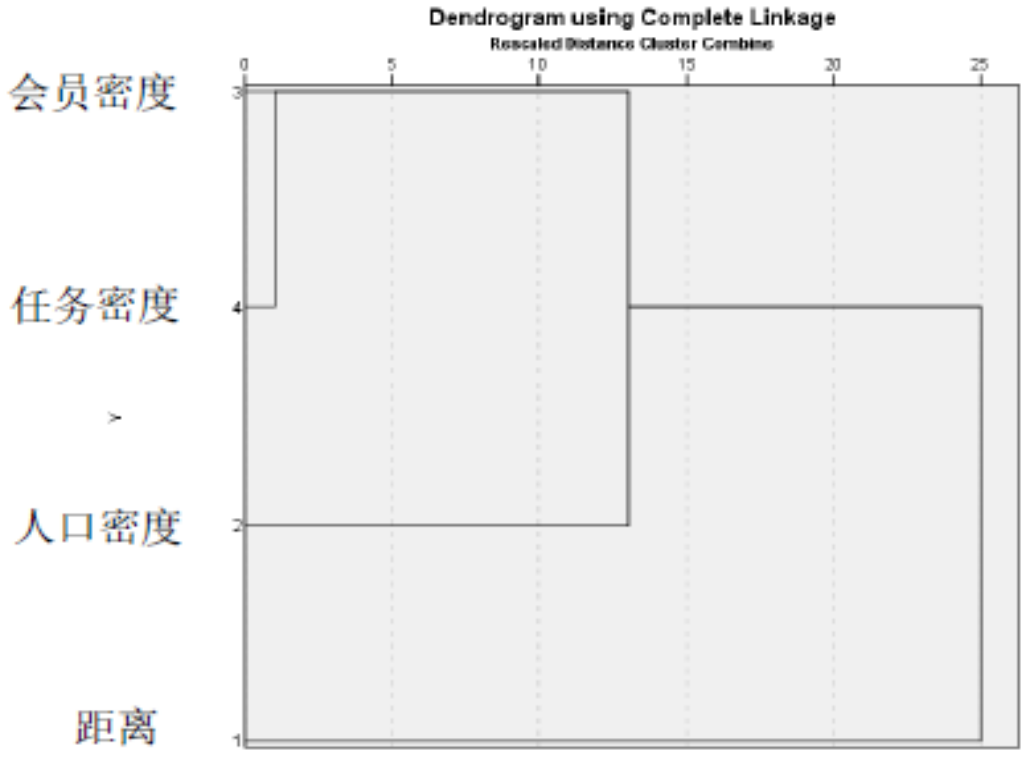

图 3 影响价格的决策变量聚类图

(4) 建立多元回归方程并求解

假设据任务中心距离、区域人口密度与价格为线性相关，建立如下回归方程：

$$y = \beta_1 \rho + \beta_2 d + \beta_3$$

建立回归参数标准方程组，并用矩阵法求解：

$$\begin{cases} \sum y = n\beta_3 + \beta_1 \sum \rho + \beta_2 \sum d \\ \sum \rho y = \beta_3 \sum \rho + \beta_1 \sum \rho^2 + \beta_2 \sum \rho d \\ \sum dy = \beta_3 \sum d + \beta_1 \sum \rho d + \beta_2 \sum d^2 \end{cases} \qquad \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} \sum \rho & \sum \rho^2 & \sum \rho d \\ \sum d & \sum \rho d & \sum d^2 \\ n & \sum \rho & \sum d \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum \rho y \\ \sum dy \\ \sum y \end{bmatrix}$$

根据最小二乘法，利用 MATLAB 求解，可得

$$\beta_1 = -0.0006245, \qquad \beta_2 = 30.28, \qquad \beta_3 = 66.89$$

于是人口密度ρ、距密度中心距离 d 与价格 y 的关系式为：

$$y = -0.0006245\rho + 30.28d + 66.89$$

对回归方程进行检测，得到残差平方和 SSE = 37.94，决定系数$R^2 = 0.8029$，均方差 RMSE = 1.452，因此说明该表达式对真实情况有良好的拟合程度。

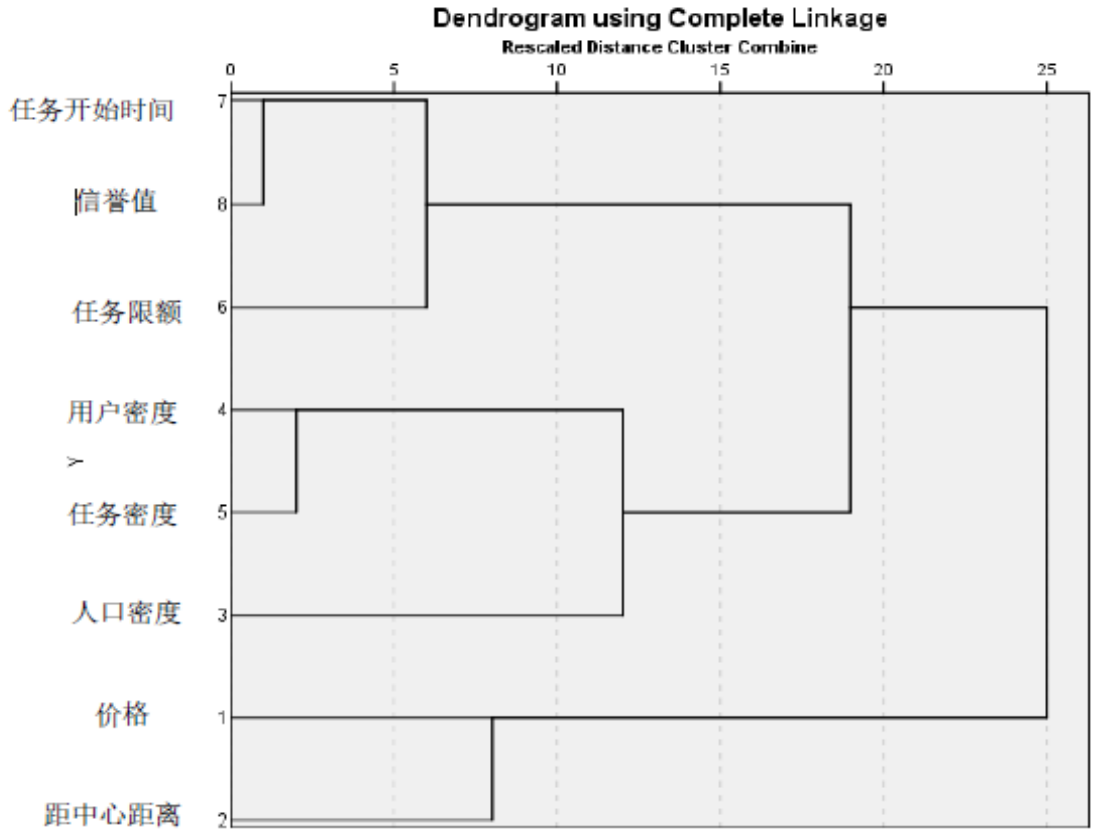

图 4 影响任务完成情况的决策变量聚类图

(5) 考虑影响任务完成情况的因素

由 SPSS 对各影响因素聚类，如右图所示。由图可知，影响任务完成情况，最主要因素是价格、距离，接着是人口密度、用户密度，至于任务密度、任务限额、任务开始、信誉值时间这四个变量，对任务完成情况的影响很小，故不做进一步讨论。

为进一步研究筛选出的四个变量如何影响，用 SPSS 做出各变量与任务完成情况的图像，如下图。故而通过具体分析，可得价格、距离与任务完成程度成正相关，而会员密度、人口密度与任务完成程度呈负相关。

因此可以合理推测，任务没有被成功执行，有可能是因为会员认为定价过低不值得花费精力去做，或会员距离任务地点太近；或者因为用户（人口）密集，会员可选择执行的任务有很多，他们不会介意任务执行率带来的负面影响。

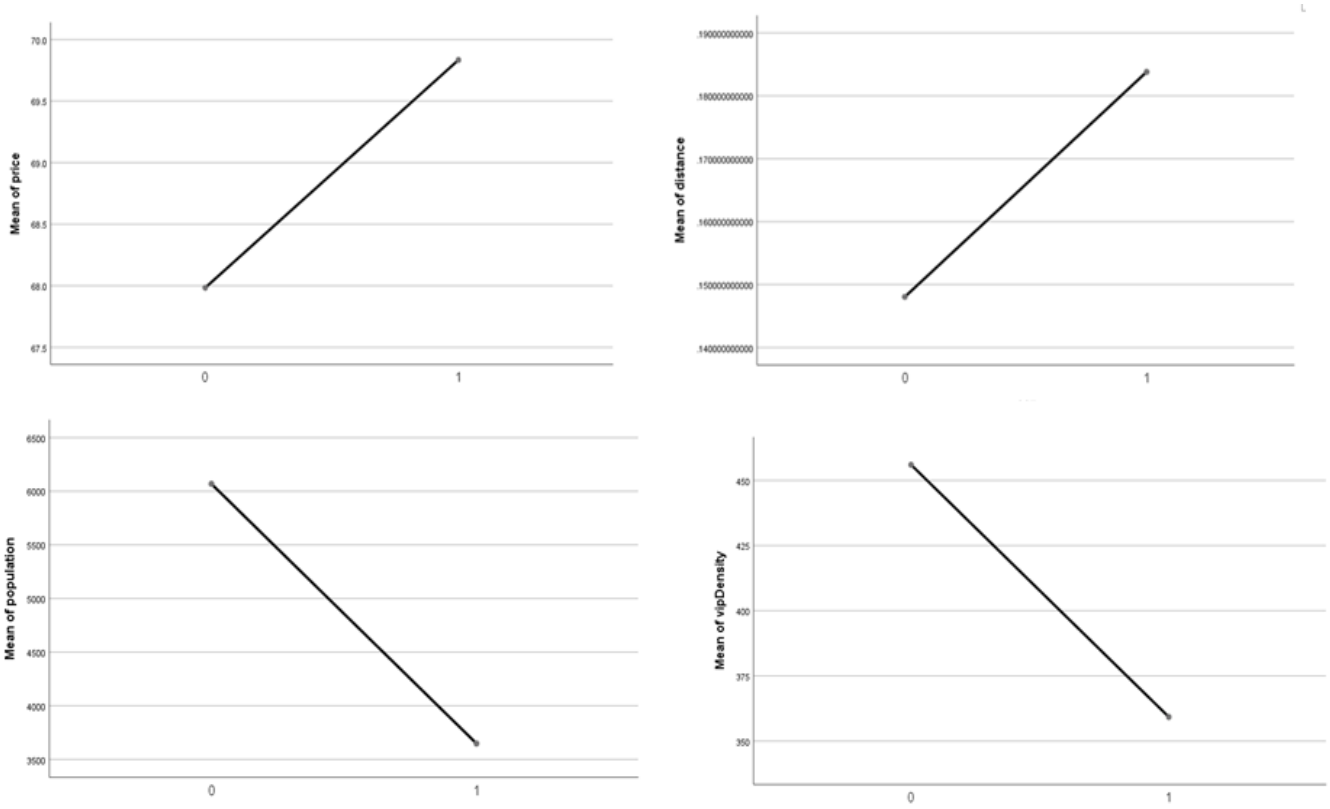

图 5 左上，右上，左下，右下分别为价格、距离、人口密度、用户密度对于任务执行情况的关系图

5.2 问题二 BP 神经网络模型的建立与分析

5.2.1 问题分析

该题需要找到合适的定价方案，使任务的完成情况整体显著提高，结合问题一的分析，该题考虑变量多且杂，有区域内任务密度、区域内会员密度、据任务密集中心距离、区域内人口密度、区域内所接任务限额、区域平均任务开展时间、价格等，这些变量具

有不确定性和模糊性，所以考虑使用具有较强自适应性的神经网络法来解决该问题。

5.2.2 建模背景与理论依据

神经网络的主要特征是大规模的并行处理和分布式的信息储存，具有良好的自适应性以及强大的学习、联想和容错功能[7-8]。因此在科研中，经常使用于高精度拟合任意的连续非线性函数，处理有大量数据并且复杂模糊的问题。目前，80%左右的神经网络采用的是 BP 网络及其衍生网络。

BP 网络具有 n 个输入节点，m 个输出节点，其关系可被看作为一个 n 维欧氏空间到 m 维欧氏空间的高度非线性映射：

$$\mathrm{F}: \mathcal{R}^n \rightarrow \mathcal{R}^m$$

在实用中，将复杂的映射规律作为一个黑箱，以实测的输入、输出数据为学习样本，得到预测值与目标值的差异并由此调整“黑箱”内每一层的权重矩阵，直到差异落入可接受范围之内。

5.2.3 BP 神经网络模型的建立

5.2.3.1 数据预处理及后期处理

由于输入数据的范围、数量级不一，很可能会导致神经网络训练收敛满，时间长，预测误差大，所以训练神经网络前需要对数据进行预处理。本文使用归一化算法：

$$y_i = \frac{x_i - min}{max - min}$$

其中 min，max 分别为为输入向量 x 的最小值、最大值。转化后，所有数据都在[0,1]之间。

而在训练之后，需把归一化的输出数据转为原始数量级。此时用到 MATLAB 的公式 premnmx（p）。

5.2.3.2 BP 网络的参数与结构

由于输入数据本身波动较大，隐藏层选用将波动映射至（-1，1）区间的双曲正切 S 型激励函数 tansig：

$$\mathrm{a} = \frac{2}{1 + e^{2n}} - 1$$

而输出数据仅为一维且波动较小，输出层选用线性激励 purelin：

$$\mathrm{a} = \mathrm{n}$$

同样影响 BP 神经网络的还有隐藏层的神经元节点数 L，已知经验公式：

$$l = \sqrt{m + n} + a$$

其中 m，n 为输入节点数及输出节点数；a 为 1～10 之间的常数。通过逐步增长、逐步修剪法的多次调整，本文设定神经元节点数为 10。

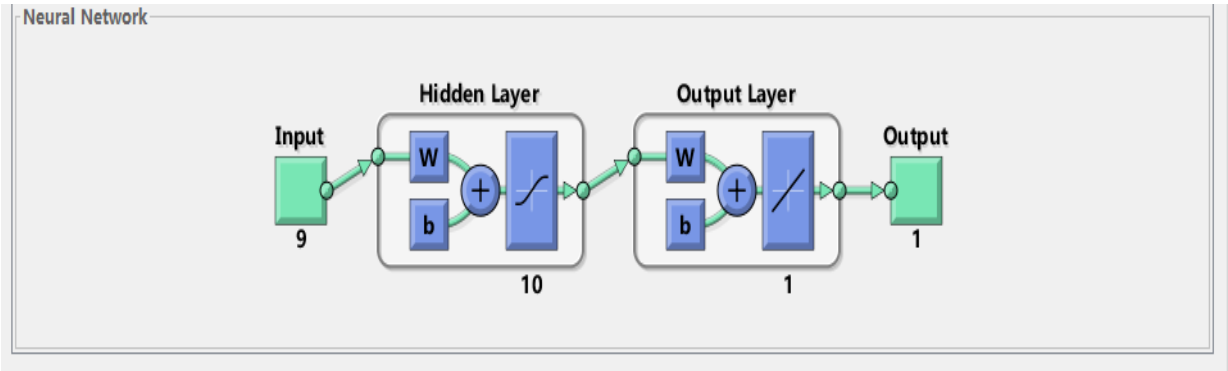

图 6 MATLAB 自带神经元点为 10

此外，根据训练中梯度变化曲线以及均方误差变化曲线，学习速率参数设定为 0.04。

本文使用 MATLAB 的神经网络工具箱（BP 网络）进行网络的代码搭建，并由工具箱自动生成的误差函数判断神经网络的拟合效果。输入数据由一个包含距密度中心距离、人口、会员密度、任务密度以及完成情况的矩阵所构成，而输出数据则仅是一个一维的定价向量。此时输入与输出数据均为真实值，其目的在于训练出一个能联系任务完成情况与定价的网络。在验证时，随机抽取两百个完成情况为 0 的数据，将其完成情况改为 1，输入训练后的神经网络，得到输出的计算价格，与真实价格进行对比，以此评价模型的效用。

5.2.4 BP 神经网络模型的求解

经 MATLAB 的神经网络工具箱学习矩阵，先后得到两组价格的数据。第一组价格数据，是当输入矩阵含有真实任务完成情况列（元为 0 或 1）的情况下得到的输出结果；第二组数据，是当输入矩阵中真实任务完成成功列（元为 1）的情况下得到的输出结果。将两组价格结合绘制散点图，可以发现，两种价格方案几乎吻合，证明神经网络所学习生成的新的价格规律复合实际生活规律。

另外，计算对两组价格的均值，可得（μ_1 为第一组数据的价格均值，μ_2 为第二组数据价格均值）

$$\mu_1 = 69.3430, \quad \mu_2 = 69.0234$$

由此可见，神经网络学习的定价规律不但与实际规律相契合，并且与原方案对比，定价普遍偏低，这也给公司带来了可观利润，所以证明新生成的定价规律比原规律更优。

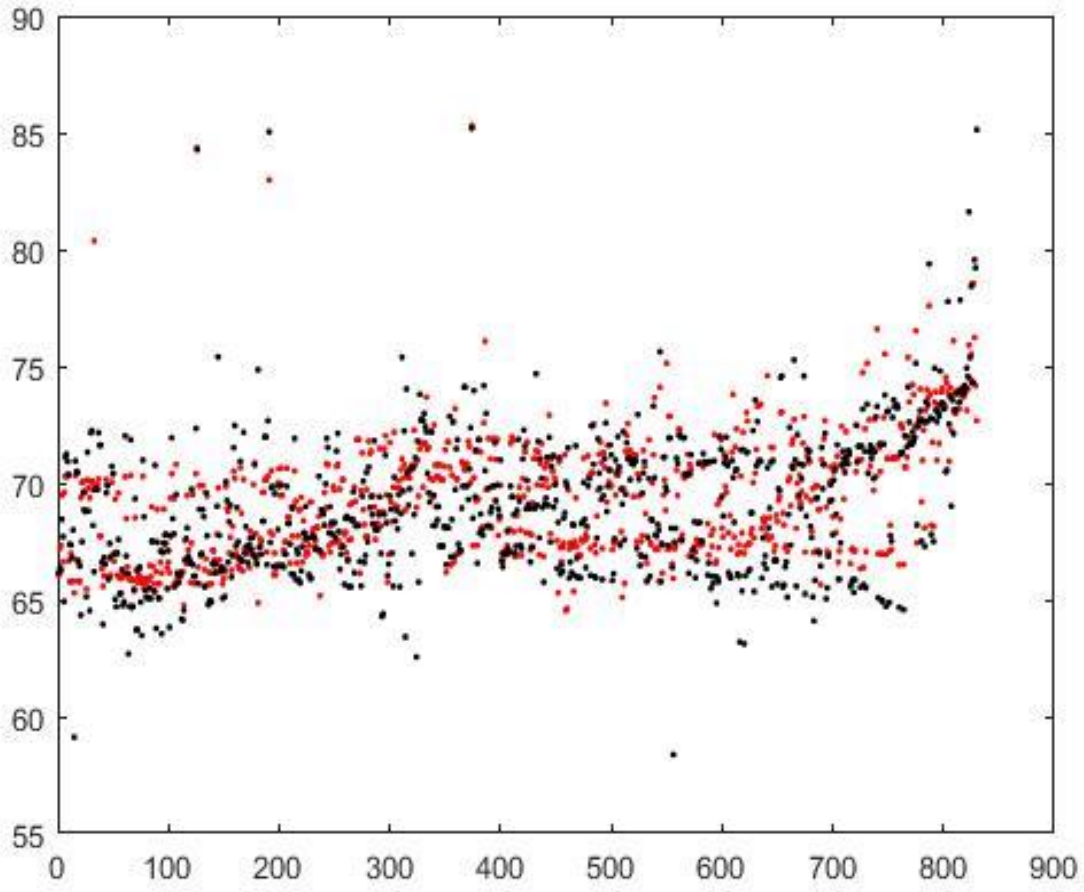

图 7 原价格（红）与任务调整后的价格（黑）对比

5.3 问题三 对 BP 神经网络模型的调整

5.3.1 问题分析

从会员的角度分析，会员一键打包认领大量任务，更多任务会被分发，但领取的任务较多，会员任务的完成程度可能会下降；从企业的角度分析，会员一键打包领取任务，任务的完成程度会下降，所以为了企业维持自身利益，打包后的任务定价应采取降价模式，以减少公司支出。

5.3.2 新变量引入后的价格调整

由于定价区间一定存在，采取以常量降价的方式对价格进行调整，具体措施如下：当该任务点的任务密度大于任务密度均值μ，即 365 时，对该任务点的价格采取降价处理，降价为 2 元。

5.3.3 分析调整后的效果

用 MATLAB 神经元工具箱生成价格调整后的任务完成情况规律，发现价格被调整过后，任务完成情况与未调整之前对比有轻微减少，其均值如下（μ_1为调整之前任务完成情况均值，μ_2为调整之后价格完成情况均值）：

$$\mu_1 = 0.6201， \quad \mu_2 = 0.6170$$

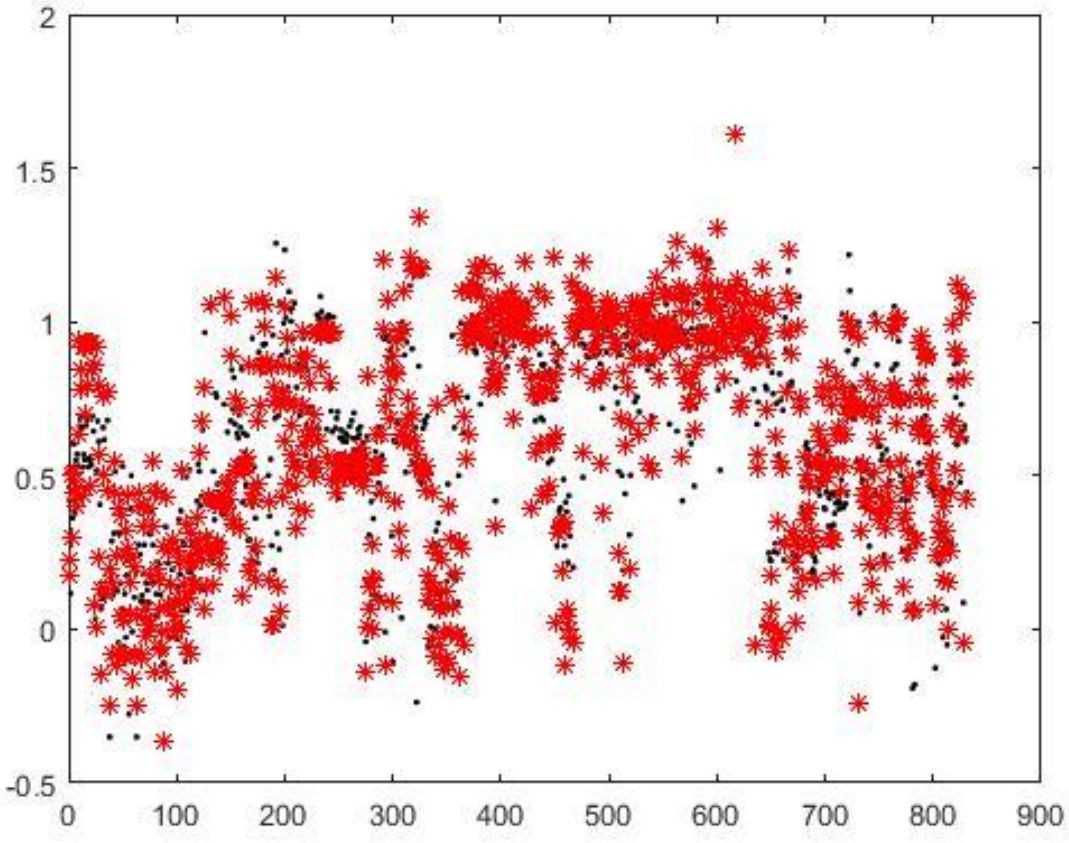

图 8 原价格（黑）与任务调整后的任务完成程度（红）对比

这样公司虽然损失一定任务完成程度，但是公司的利益从价格处得到补偿，即少付佣金以减少支出；客户虽然得到的佣金减少了，但是一键打包为其提供了便捷，所以该模型存在一定合理性。

5.4 问题四 对新数据预测价格

5.4.1 数据预处理

与第一问相似地，使用 XX 软件将坐标定位于具体的城市区域，并通过人口普查等报告确定此处的人口密度。再通过 MATLAB 的 cluster 文件算出距密度中心距离、会员密度、任务密度，将其组合成为输入矩阵。

5.4.2 结果与分析

此时的神经网络经过训练，接受输入矩阵后可以算出新条件下的价格以及完成情况，如图所示，可见在新的价格规律下，价格不再突升至八十以上，而是稳定保持在[60,70]的区间里。同时，任务完成情况也得到了一定的改善。由此可以判定，第二问的模型在对“打包”情况进行进一步的建模之后，在保持任务完成情况无大幅改变的情况下，将公司所需要支付的薪金降低到了一个稳定的水平。

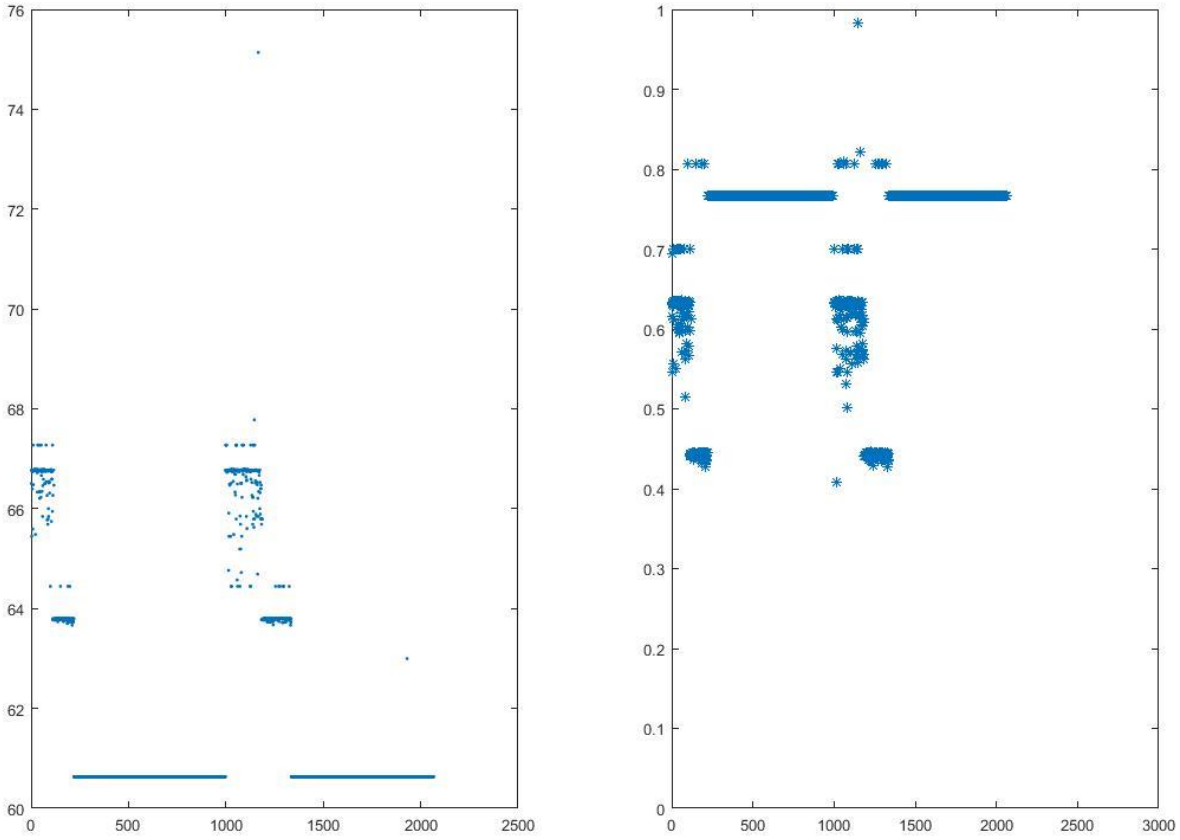

图 9 神经网络模型的预测 价格预测 完成程度模拟

而此时将数据中的距密度中心距离以及人口密度进行组合，输入第一问中得到的价格规律，即回归方程：

$$y = -0.0006245\rho + 30.28d + 66.89$$

比较两组价格分布，第三问中的得到的模型所产生的价格分布显然均值更低，且波动更少，由此可见新模型较原模型更能给公司提供利润。

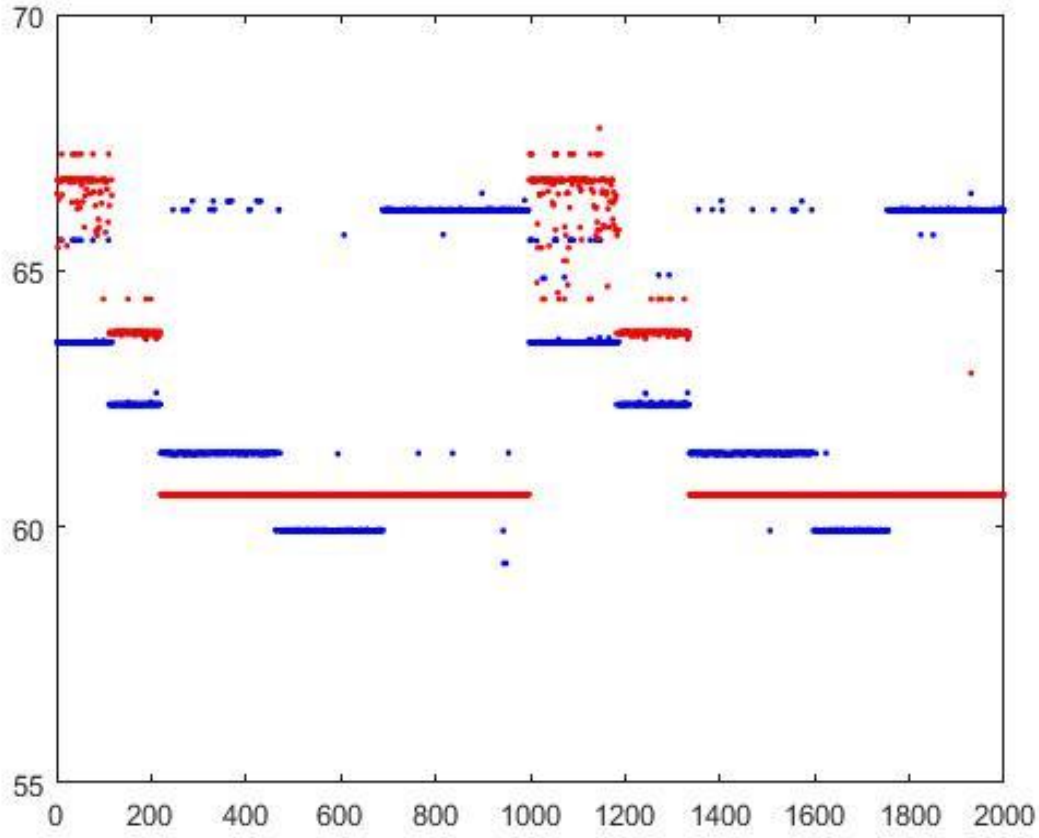

图 10 比较多元回归模型的预测与神经网络模型的预测

六 模型的改进与评价

6.1 模型的优点

(1) 模型是在对附件 1，2 中的数据进行充分的统计后得出的，经过大量的分析，验证与完善使模型的准确度和可信度得到了较强的保证。

(2) 对于问题一，使用了多元回归方程模拟的方法得到模型结果，此方法适用于解决受多个变量所影响的实际经济问题。

(3) 对于问题二，三，四，使用了 BP 神经网络模拟的方法得到模拟结果，此方法的容错能力较强，使得模拟结果不会与现实偏差较大。

(4) 在数据处理及模型求解的过程中充分运用了 MATLAB, SPSS 等数学软件，较为理想地回答了问题，得到了的结果也较为理想。并且充分用了题目及附件中的各种信息，以及较好地结合了对模型的检验。

6.2 模型的缺点

(1) 在问题一中运用的多元回归方程的模拟方法中，只考虑了距离和人口密度两类变量，

使得模拟的结果的准确水平有所下降。

(2) 在问题二，三，四中所使用的 BP 神经网络的模拟方法中，可能会由于方法自身的预测能力与训练问题的矛盾使得学习所得的模型无法反映样本内涵的规律。

6.3 模型的可推广性分析

因为本文对第三题打包问题的解答具有广泛性，可以推广把此应用推广到其他需要解决此类打包问题的 app（如打车类型，快递类型，外卖类型），使得 app 对于有限范围中密集任务的处理效率得到改善，以达到令消费者感到满意，资源得到最大利用的效果。

七、参考文献

[1] 谢文蕙，邓卫 城市经济学.第二版. 北京：清华大学出版社，2008

[2] 深圳统计年鉴 2016.深圳：深圳市统计局，2016

[3] 广州统计年鉴 2016.广州：广州市统计局，2016

[4] 佛山统计年鉴 2016.佛山：佛山市统计局，2016

[5] 东莞统计年鉴 2016.东莞：东莞市统计局，2016

[6] 广东省 2010 年第六次全国人口普查主要数据公报.广东省统计局，2011

[7] 卓金武. MATLAB 在数学建模中的应用. 北京：北京航空航天大学出版社，2011.

[8] 司守奎、孙玺菁 数学建模算法与程序. 北京：国防工业出版社，2011

八、附件

1. myCluster

```
[number,txt,raw] = xlsread('F:\brandNew');
[number1,txt1,raw1] = xlsread('F:\truncked_2');
sr = number(:,4);
vipX = number1(:,4);
vipY = number1(:,5);
vip = [vipX.*1.1094,vipY.*0.85276];
price = number(:,3);
index = 1:length(price);
population = number(:,5);
x = number(:,1).*1.1094;
y = number(:,2).*0.85276;
X = [x,y];
subplot(1,2,1)
[idx,C,sumd,D] = kmeans(X,5,'dist','cityblock');
ptsymb = {'bs','r^','md','go','c+'};
for i = 1:5
    clust = find(idx==i);
    plot(X(clust,1),X(clust,2),ptsymb{i});
    hold on
end
plot(C(:,1),C(:,2),'ko');
axis equal
grid on



mat = [index',idx,price,X,D,population,sr];
matOriginal = mat;
for i = 1:length(matOriginal)
    j = length(matOriginal)-i+1;
    cluster = idx(j);
    if D(j,cluster)>0.5
        mat(j,:) = [];
    end
end
newMat = zeros(length(mat),14);
for i = 1:length(mat)
    dArray = mat(i,6:11);
    d = min(dArray);
    p = mat(i,11);
    ssr = mat(i,12);
```

```
    cx = mat(i,4);
    cy = mat(i,5);
    %density of vip
    deltaVIP = sqrt((vip(:,1)-cx).^2+(vip(:,2)-cy).^2)-0.3;
    v = find(deltaVIP<0);
    vipDensity = length(v);
    %mean credit,
    all_credit = 0;
    all_st = 0;
    all_bound = 0;
    for j = 1:length(v)
        index = v(j);
        all_credit = all_credit+number1(index,3);
        all_st = all_st+number1(index,2);
        all_bound = all_bound+number1(index,1);
    end
    mean_credit = all_credit/length(v);
    mean_st = all_st/length(v);
    %density of other missions
    deltaMission = sqrt((mat(:,4)-cx).^2+(mat(:,5)-cy).^2)-0.3;
    m = find(deltaMission<0);
    mDensity = length(m);
    prob = 0;
    if mDensity > 357.2
        prob = 1;
    end
    newMat(i,:)                                                                    =
[mat(i,1:5),d,p,vipDensity,mDensity,ssr,mean_credit,mean_st,all_bound,prob];
end
xlswrite('F:\truncked',newMat)

scatter(newMat(:,6),newMat(:,3))

%plot
subplot(1,2,2)
Z = mat(:,4:5);
[idx,C,sumd,D] = kmeans(Z,5,'dist','cityblock');
ptsymb = {'bs','r^','md','go','c+'};
for i = 1:5
    clust = find(idx==i);
    plot(Z(clust,1),Z(clust,2),ptsymb{i});
    hold on
end
plot(C(:,1),C(:,2),'ko');
```

```
grid on
axis equal



% Y = [mat1(:,4:5),mat1(:,3)];
% [cidx2,cmeans2] = kmeans(Y,5,'dist','cityblock');
% ptsymb = {'bs','r^','md','go','c+'};
% for i = 1:5
%     clust = find(cidx2==i);
%     plot3(Y(clust,1),Y(clust,2),Y(clust,3),ptsymb{i});
%     hold on
% end
% plot3(cmeans2(:,1),cmeans2(:,2),cmeans2(:,3),'ko');
% plot3(cmeans2(:,1),cmeans2(:,2),cmeans2(:,3),'kx');
% hold off
% xlabel('Sepal Length');
% ylabel('Sepal Width');
% zlabel('Petal Length');
% view(-137,10);
% grid on

2.truncked_attachment2
%process attachment2
[number,txt,raw] = xlsread('F:\attachment2_self');
bound = number(:,1);
start_time = number(:,2);
credit = number(:,3);
number_original = number
credit_bound = polyfit(credit,bound,2);
credit_st = polyfit(credit,start_time,2);

credit_bound_value = polyval(credit_bound,credit);
credit_st_value = polyval(credit_st,credit);

error_cb = abs(bound - credit_bound_value);
error_cst = abs(start_time - credit_st_value);

n = length(number);
for i = 1:n
    j = n-i+1;
    if error_cb(j)>3.5
        number(j,:) = [];
    elseif error_cst(j)>0.35
```

```
        number(j,:) = [];
    end
end

subplot(2,2,1)
plot(credit,bound,'r*',credit,credit_bound_value)
subplot(2,2,2)
plot(credit,start_time,'r*',credit,credit_st_value)

xlswrite('F:\truncked_2',number)

bound = number(:,1);
start_time = number(:,2);
credit = number(:,3);
position = [number(:,4).*1.1094,number(:,2).*0.85276];

credit_bound = polyfit(credit,bound,1);
credit_st = polyfit(credit,start_time,1);

credit_bound_value = polyval(credit_bound,credit);
credit_st_value = polyval(credit_st,credit);

subplot(2,2,3)
plot(credit,bound,'r*',credit,credit_bound_value)
subplot(2,2,4)
plot(credit,start_time,'r*',credit,credit_st_value)

3. interpolation_3d
clc
[number,txt,raw] = xlsread('F:\mean');
price = number(:,1);
distance = number(:,7);
population = number(:,8);
X = [distance,population];
mdl = fitlm(X,price);
mdl.Rsquared.Ordinary
cftool

4. densePoint
[number,txt,raw] = xlsread('F:\newdata');
[number1,txt1,raw1] = xlsread('F:\truncked');

%number = csvread('CHERINE/subfile1.csv',1,0);
location = number(:,1:2);
```

```
prize = number(:,3);
prize = prize./sum(prize);
x = location(:,1);
y = location(:,2);


stepSize = 0.05;
colIndex = floor((x-22)./stepSize)+1;
rowIndex = floor((y-112.5)./stepSize)+1;
index = [rowIndex,colIndex];
[xAxis,yAxis]=meshgrid(22+stepSize/2:stepSize:24-
stepSize/2,112.5+stepSize/2:stepSize:114.5-stepSize/2);
m = zeros(length(xAxis),length(yAxis));
for n = 1:length(index)
    i = index(n,1);
    j = index(n,2);
    m(i,j) = m(i,j)+1;
end


%prize contour
prizeMatrix = zeros(length(xAxis),length(yAxis));
for n = 1:length(index)
    i = index(n,1);
    j = index(n,2);
    prizeMatrix(i,j) = prizeMatrix(i,j)+prize(n);
end
for i = 1:length(prizeMatrix)
    for j = 1:length(prizeMatrix)
        if m(i,j) ~= 0
            prizeMatrix(i,j) = prizeMatrix(i,j)/m(i,j);
        end
    end
end

%vip contour
vip_x = number1(:,4)./1.1094;
vip_y = number1(:,5)./0.85276;
vipMatrix = zeros(length(xAxis),length(yAxis));
colIndex1 = floor((vip_x-22)./stepSize)+1;
rowIndex1 = floor((vip_y-112.5)./stepSize)+1;
index1 = [rowIndex1,colIndex1];
for n = 1:length(index1)
    i = index1(n,1);
```

```
    j = index1(n,2);
    vipMatrix(i,j) = vipMatrix(i,j)+1;
end


%find dense point which has density > 7
k = find(m>7);
krow = mod(k,length(yAxis));
kcol = ceil(k/length(xAxis));
kPositionX = zeros(length(k),1);
kPositionY = zeros(length(k),1);
xAxisRow = xAxis(1,:);
yAxisCol = yAxis(:,1);
for i = 1:length(k)
    kPositionX(i) = xAxisRow(kcol(i));
    kPositionY(i) = yAxisCol(krow(i));
end
kPosition = [kPositionX,kPositionY];



%calculate distance of each point
dx = zeros(length(kPositionX),1);
dy = zeros(length(kPositionY),1);
distance = zeros(length(location),1);
for i = 1:length(location)
    dx = kPositionX - location(i,1);
    dy = kPositionY - location(i,2);
    distance(i) = min(dx.^2+dy.^2);
end
distance =  distance./sum(distance);


%
% p = polyfit(distance,prize,50);
% x1 = linspace(min(distance),max(distance));
% y1 = polyval(p,x1);
plot(distance,prize,'*')

%gray
% syms a b ;
% c = [a b]';
% A = prize';
% B = cumsum(A);
```

```
% n = length(A);
% for i = 1:(n-1)
%     C(i) = (B(i)+B(i+1))/2;
% end
% D = A;
% D(1) = [];
% D = D';
% E = [-C;ones(1,n-1)];
% c = inv(E*(E'))*E*D;
% c = c';
% a = c(1);
% b = c(2);
% F = [];
% F(1) = A(1);
% for i = 2:(n)
%     F(i) = (A(1)-b/a)/exp(a*(i-1))+b/a;
% end
% G = [];
% G(1) = A(1);
% for i = 2:(n)
%     G(i) = F(i) - F(i-1);
% end
% t1 = distance;
% t2 = distance;
% G
% plot(t1,A,'o',t2,G)


 subplot(1,3,1)
 hold on
 contour(xAxis,yAxis, prizeMatrix,15);
 scatter(kPositionX,kPositionY)
 scatter(x,y,'.');
 hold off

subplot(1,3,2);
 hold on
 [C,h] = contour(xAxis,yAxis,m,15);
 scatter(kPositionX,kPositionY)
 scatter(x,y,'.');
 hold off

 subplot(1,3,3);
 hold on
```

```
 [C,h] = contour(xAxis,yAxis,vipMatrix,15);
 scatter(kPositionX,kPositionY)
 scatter(x,y,'.');
 hold off
```

5.interpolation_3d_q2

```
[number,txt,raw] = xlsread('F:\truncked_1_mean');
price = number(:,3);
distance = number(:,6);
population = number(:,7);
cftool
```

6.test

```
[number,txt,raw] = xlsread('F:\test');
price_true = number(:,3);
distance = number(:,6);
population = number(:,7);
[d,p] = meshgrid(distance,population);

p00 = 66.89  ;
p10 = 30.28  ;
p01 = -0.0006254  ;
X = [distance,population];
price_culculated = p00 + p10.*d + p01.*p;
hold on
mesh(d,p,price_culculated)
grid on
plot3(distance,population,price_true,'.');
hold off
```

7.calerr

```
function err = calerr(a)
    ave=mean(a);
    err=a-ave;
    s=std(a);
    n=length(a);
    for i=1:n;
        if abs(err(i))-2*s>0;
            a(i)=0;
        else
            continue;
        end;
    end;
    err=a
```

```
end


8. neural network
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 17-Sep-2017
16:36:14.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = Qx8 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset =
[65;0.00465064365332069;312;28;31;2.77231538461538;1.8;105];
x1_step1.gain = [0.1;4.10517925873271;5.80635796196836e-
05;0.0036101083032491;0.00546448087431694;0.0306905905086998;0.017814753944
7564;0.000160129139900285];
x1_step1.ymin = -1;

% Layer 1
b1 = [1.7852719829729204;-1.5197470178864425;-
1.9718418488661003;0.69051704419472038;0.50561480938113557;-
0.024305272701410235;1.1776300346655073;0.7948135566330734;-
0.88945038613973881;-1.8547301775216027];
IW1_1 = [-1.091005992278762 -0.10025783507112454 -0.59131364854586499
0.097958757070439731 0.55272095410261834 0.95673233027493831 -
0.20878029378250945 0.9345095574314356;-0.27529371164421351
0.14900899429362024 -0.83321834391884719 0.53786332183834984 -
2.7842112403528407 1.4610270227269737 -0.50896191117368506 -
```

```
0.921960086848441;1.3263843837017204 0.13640779231370107 -
1.1117031545097493 0.99474802053505418 -0.31936488277424901
0.1791385250546236 -0.11524012610005542 -0.59143223988338711;-
0.85396144923570227 0.3793033386175908 1.5457755269976969 -
1.3367245345966574 -0.68444294433625352 -0.15630715257634686 -
0.39757093553817524 -0.055080323560860067;-0.026637671088642349 -
0.24533231073045714 -0.12022341985700039 1.3822984498688549
0.11020946401966689 0.1486264328659927 -0.59012963138010655
0.61198368367447542;-0.379176585243306 0.049609954426879731 -
0.62511581125554128 0.56548379751782307 -0.37629962662004551 -
1.11907139820223 -1.4740693580992463 -0.62319202211204938;-
0.2800996947913541 2.6360268675209091 0.20631543554958368
0.40878430509432817 -1.3039497449533193 0.10846878011127528 -
0.73310438926758137 -1.2294150260800851;-0.15055783937315914 -
1.0183133558568098 -0.2460849856515841 -1.8441713006150682
1.8757252467853358 -0.11844189579088241 1.2209296155960132 -
1.0842610266232038;-1.2935669316672287 0.53962180838747309
0.32436890185528577 1.5828773898128345 -1.4582545477752522 -
0.98881671080536371 0.0032979572128915557 -1.3787789298880586;-
0.58630825533844089 0.69531163446165767 0.7746990860751044
0.89605230743041608 -0.24423712307542722 0.67978843196163807 -
0.060226413683026059 0.54502429851041523];

% Layer 2
b2 = -0.63689502169131729;
LW2_1 = [-0.3954205523349788 1.3946209425211586 1.5498814481732615
0.10291769576992099 1.1125816984160137 -1.0521964227031539
0.96745499041115512 3.1069557598535011 1.0619777355736861
0.11839134240774168];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 2;
y1_step1.xoffset = 0;

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
```

```
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
```

```
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
```