电 子 科 技 大 学 实 验 报 告

课程名称：数学实验

实验地点：科 A227

指导教师：杨宇明

评　　分：

完成实验学生信息：

选课序号	姓名	学号	贡献百分比/%	备注
21			100	全部

注：

1. 学生人数按照任课教师要求限定；

2. 对于“评价、改进、总结和体会”都要认真填写，和其他内容是评价实验成绩的重要参考。

目录

1. 问题内容

某个打车平台的数据中心目前有 1000 位正在等待出行的顾客.现在已知每位顾客的出发地点的经度、纬度，到达地点的经度、纬度.这些数据保存在 data_customer.txt 文本文件中(位置数据是虚拟数据,并不一定完全符合实际).每行代表一个顾客的信息,依次表示出发地点经度、出发地点纬度、到达地点经度、到达地点纬度。

如果要把顾客出发地所在区域分为 10 个分区管理,请问每位顾客应分别分在哪个区,每个区的中心在哪里。

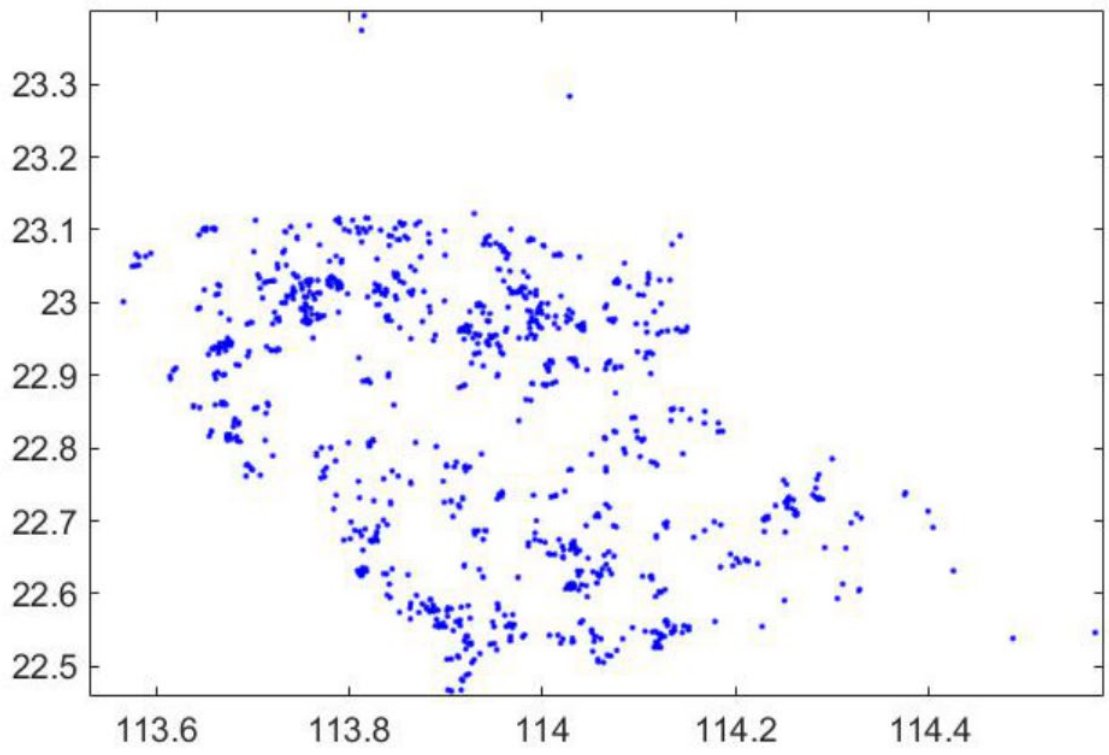


图 1-打车平台顾客发车请求地点散点图

2. 问题分析

2.1. 数据分析

本次建模任务所给数据为一个 1000*4 的矩阵，矩阵的每一行分别有每个顾客的出发地点的经度、纬度，到达地点的经度、纬度，根据对所给问题进行分析，可得本次建模任务为划分顾客出发地所在区域为 10 个分区及根据所划分区域求得区域中心点。故本次建模任务所需数据为所给数据中的每位顾客的出发地点的经度，纬度，即一个二维的矩阵。

2.2. 区域划分分析

已知问题需要对所给的 1000 个二维数据进行 10 个分区的划分，即依据 1000 个二维数据聚类形成 10 个簇，根据数据具有的特征（出发地点的经度、纬度）的某种相似性分别聚成不同的簇。

这种相似性在问题解决过程中体现为与 10 个神经元（即权重）的欧氏距离。

2.3. 区域中心点分析

传统的获取分区中心点的方法是通过聚类形成不同的簇，求取簇内数据在各个维度上的均值。

本次项目的解决方案是设计 10 个神经元（即权重），最初的输入数据在二维空间占据了特殊分布，未学习的 10 个神经元（即权重）随机分布在一个足够小的区域，在输入修改和学习神经元（即权重）后，它在竞争性学习过程中逐步获取输入数据分布

的形状，也即形成了分区中心点，此外，每个神经元及其之间的连接体现了输入数据在空间上拓扑分布。

2.4. 模型假设

1. 假设存在一个片状的浅层人工神经网络，其神经元通过无监督的学习过程对应占据特殊分布的输入数据分别进行专门学习调整。
2. 假设该片状浅层人工神经网络的神经元在对输入数据的学习调整过程中能够逐步获取输入数据分布的特点，且能够对应输入数据的特殊分布形成分区，且最佳匹配的神经元及其拓扑邻域对当前输入进行进一步的敏感化或调整，以保持输入空间的拓扑性质，并以神经元作为分区的中心点。

3. 定义与符号说明

3.1. 名词解释

竞争性学习：假设有一个矢量观测变量 $x=x(t)\in\mathbb{R}^n$的统计样本序列，其中 t 为时间坐标，还有一组参考向量$\{m_i(t):m_i\in\mathbb{R}^n,i=1,2,...,k\}$。假设 mi(0)已经以某种适当方式初始化，通常是以随机方式进行。如果 x(t)能够以某种方式在每个连续的瞬间同时与每个 $m_i(t)$比较，那么最佳匹配的 $m_i(t)$将被更新以更接近当前的 x(t)。如果是基于某种距离度量 $d(x,m_i)$的比较，如果 i=c 是最佳匹配参考向量的索引，那么 $d(x,m_c)$将被减少，而其它 $i\neq c$ 的参考向量将被保留。通过这种方式，不同的参考向量趋于专门调整到输入变量的不同区域。

3.2. 符号说明

符号	说明
j	神经元的序号
i	输入样本的序号
n	可用的输入向量数
l	可用的神经元数
W	神经元的权重向量
i(x)	获胜神经元
x	输入样本
$h_{j,i}(x)$	邻域函数
$d_{j,i}$	序号为 j 的神经元与序号为 i 的输入样本的欧氏距离
$\eta(\tau)$	迭代次数为 τ 的学习率
$L(\tau)$	迭代次数为 τ 的邻域
T	最大迭代次数

4. 模型的建立与求解

4. 1. 自组织神经网络的建模

我们建立一个简单的浅层神经网络，其包含输入层，输出层，神经元，不包含隐含层。

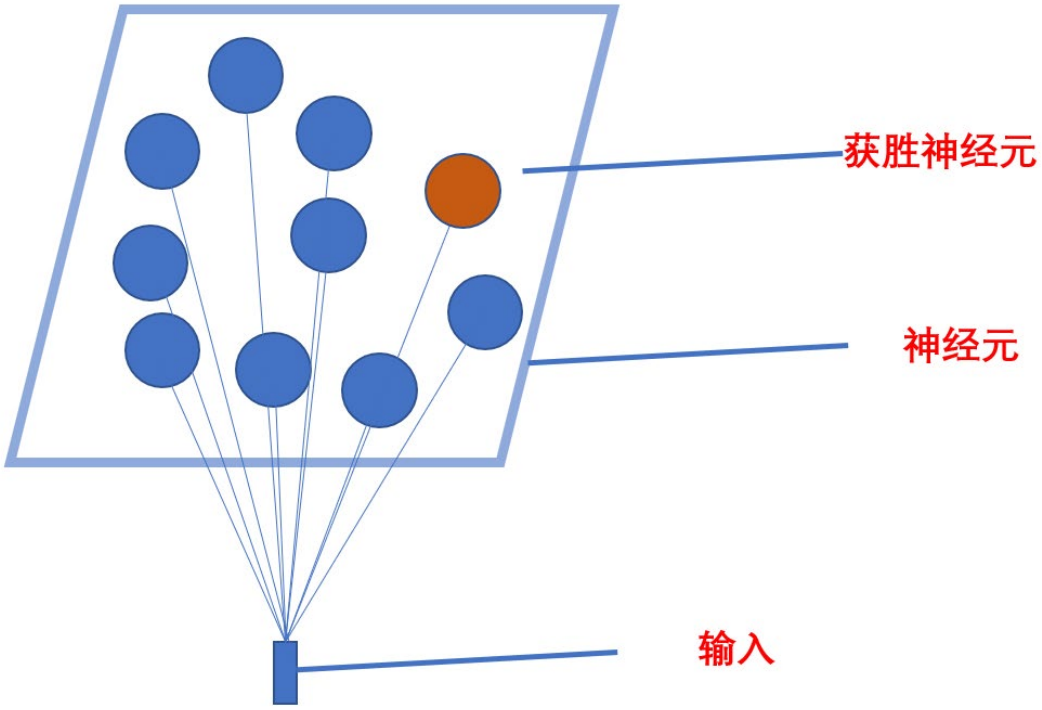


图 1

该神经网络模型功能如下

1．选择与输入模式最佳匹配的神经元

考虑到图 1 描述的二维神经网络，神经元的初始排列是随机的，令 $x=[x_1,x_2,...x_n]T\in\mathbb{R}^n$作为输入向量，为简化运算，假定其与网络中所有神经元 j 平行连接。神经元的权重向量表示为 $mj=[m_{j1},m_{j2},...m_{jn}]T\in\mathbb{R}^n$，而 x 与 m_j 的匹配分析度量，我们采用了基于 x 和 m_j 之间的欧氏距离，而获胜动作定义为 x 和 m_j 之间的最小欧氏距离。

2. 神经元的权重向量的适应

我们假定进行学习的神经元不会独立地受到彼此的影响。而是作为拓扑学上相关子集，对每个子集施加类似的修正。这个过程中，这些选定的子集包含不同的神经元。每个子集的净校正将趋于平滑化。

我们通过定义神经元周围的邻域集 N_c，以一般的形式直接实现横向交互，以适应任意的基础网络结构，在每个学习步骤中，N_c 内的所有神经元都被更新，而 N_c 外的神经元保持不变，这个邻域是以与输入 x 的最佳匹配神经元作为中心。

$$\|x - m_c\| = \min_i\{\|x - m_i\|\}$$

Nc 的半径是时变，且随时间单调地缩小，这是因为在学习过程中，一个较宽的初始 Nc，对应于一个粗略的空间分辨率，首先在 mi 的初始值中推导出一个粗略的全局排列，之后，缩小 Nc 可以提高网络的空间分辨率，而获得的全局排列并不会被破坏。在这个过程中我们只更新最佳匹配子集（获胜子集）。

我们将更新过程表示为

$$m_i(t+1)=\begin{cases} mi(t) + hci(t)[x(t) - mi(t)] & ,if\ i \in Nc(t) \\ mi(t) & ,if\ i \notin \mathrm{Nc}(t) \end{cases}$$

其中，$h_{ci}(t)$是一个标量值的适应性增益，且 $0<h_{ci}(t)<1$,在学习过程中，$h_{ci}(t)$随时间推移而减小。

$$h_{ci} = h_0 \mathrm{e}^{\left(\frac{-\|r_i - r_c\|^2}{\sigma^2}\right)}$$

σ=σ(t)为一个时间递减函数。

4. 2. 自组织神经网络模型的算法实现

1. **初始化**：为所有神经元 j=1,2,...l 的初始权重向量 $W_j(o)$选择足够小的随机值

2. **相似性匹配**：在输入中获取样本 x 后，找到最佳匹配(获胜)的神经元如下所示

$$i(x) = \arg\min_j \|x(h) - w_j\|, j = 0,1,2,3\cdots l, h=0,1,2...n$$

3. **更新**：调整在邻域内(设为 Lo=$\sqrt{l}$)所有激励神经元的权重向量，调整式子如下

$$w_j(\tau+1)=w_j(\tau)+\eta(\tau)h_{j,i(x)}(\tau)(x(\tau)-wj(\tau))$$

其中，hj,i(x)(n)为邻域函数，我们选择了 exp(-$\frac{d2j,i}{2\sigma2(n)}$)作为邻域函数

η(τ)随τ的推移而下降， η(τ)= η0exp(-τ/T)

σ(τ)随τ的推移而下降， σ(τ) = σ0exp(-τ/T)

最大邻域 L 随τ的推移而下降，L(n)=Lo*(1-τ/T)

4. **迭代**：继续迭代至最大迭代次数

4. 3. 自组织神经网络模型的代码实现及聚类检验

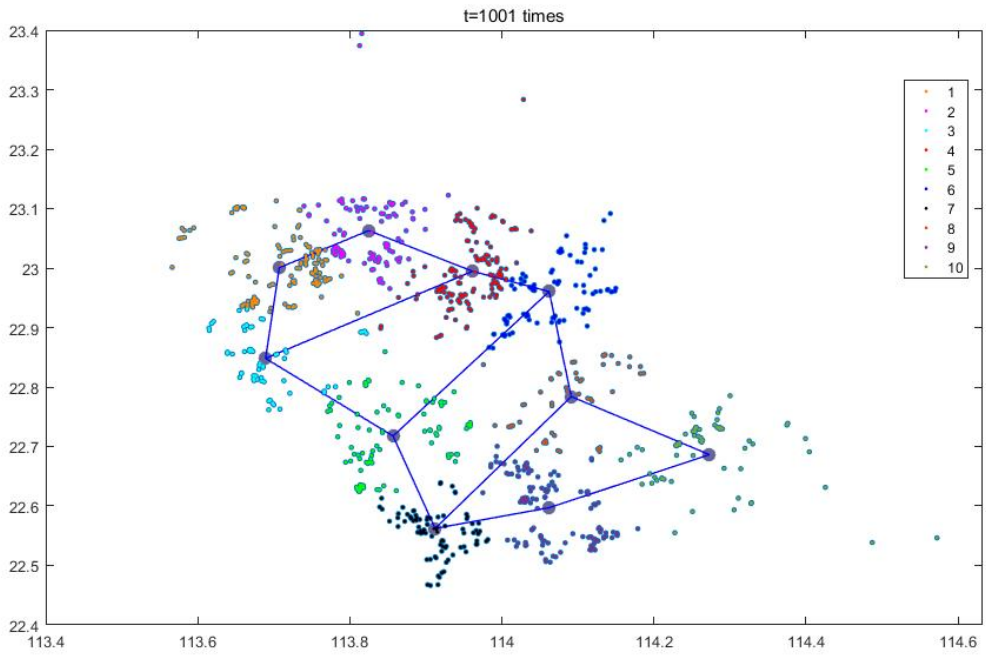
```
load data_customer.txt;
data=data_customer(:,1:2); % 提取出出发的数据
num=length(data);
aini=0.05;
sigmaini=0.005;
LO=sqrt(10)/2;
T=400;  % 迭代次数 可改小 大概 300 次拟合
t=1;
i=1;
for j1=1:5
```

```
    for j2=1:2
        w1(j1,j2)=113.4+(114.6-113.4)*rand;
        w2(j1,j2)=22.4+(23.4-22.4)*rand;
        i=i+1;
    end
end
x1 = data(:,1);
x2 = data(:,2);
while (t<=T)
    a=aini*exp(-t/T);
    sigma=sigmaini*exp(-t/T);
    LN=round(LO*(1-t/T));
    for i=1:num
        e_norm=(x1(i)-w1).^2+(x2(i)-w2).^2;
        minj1=1;minj2=1;
        min_norm=e_norm(minj1,minj2);
        for j1=1:5
            for j2=1:2
                if e_norm(j1,j2)<min_norm
                    min_norm=e_norm(j1,j2);
                    minj1=j1;
                    minj2=j2;
                end
            end
        end
        j1_c= minj1;
        j2_c= minj2;
        bias=[1 2;3 4; 5 6; 7 8;9 10];
        idx(i)=bias(j1_c,j2_c);
        w1(j1_c,j2_c)=w1(j1_c,j2_c) + a * (x1(i) - w1(j1_c,j2_c));
        w2(j1_c,j2_c)=w2(j1_c,j2_c) + a * (x2(i) - w2(j1_c,j2_c));
        for neighbour_radius=1:1:LN
            jj1=j1_c - neighbour_radius;
            jj2=j2_c;
            if ((jj1>=1)&&(jj1<=5))
                e_factor = exp(-((jj1-j1_c).^2+(jj2-j2_c).^2)/2*sigma^2);
```

```
                w1(jj1,jj2)=w1(jj1,jj2) + a * e_factor * (x1(i)-w1(jj1,jj2));
                w2(jj1,jj2)=w2(jj1,jj2) + a * e_factor * (x2(i)-w2(jj1,jj2));
            end
            jj1=j1_c + neighbour_radius;
            jj2=j2_c;
            if ((jj1>=1)&&(jj1<=5))
                e_factor = exp(-((jj1-j1_c).^2+(jj2-j2_c).^2)/2*sigma^2);
                w1(jj1,jj2)=w1(jj1,jj2) + a * e_factor * (x1(i)-w1(jj1,jj2));
                w2(jj1,jj2)=w2(jj1,jj2) + a * e_factor * (x2(i)-w2(jj1,jj2));
            end
            jj1=j1_c;
            jj2=j2_c - neighbour_radius;
            if ((jj2>=1)&&(jj2<=2))
                e_factor = exp(-((j1_c-jj1).^2+(j2_c-jj2).^2)/2*sigma^2);
                w1(jj1,jj2)=w1(jj1,jj2) + a * e_factor * (x1(i)-w1(jj1,jj2));
                w2(jj1,jj2)=w2(jj1,jj2) + a * e_factor * (x2(i)-w2(jj1,jj2));
            end
            jj1=j1_c;
            jj2=j2_c + neighbour_radius;
            if ((jj2>=1)&&(jj2<=2))
                e_factor = exp(-((j1_c-jj1).^2+(j2_c-jj2).^2)/2*sigma^2);
                w1(jj1,jj2)=w1(jj1,jj2) + a * e_factor * (x1(i)-w1(jj1,jj2));
                w2(jj1,jj2)=w2(jj1,jj2) + a * e_factor * (x2(i)-w2(jj1,jj2));
            end
        end
    end
    t=t+1;
    figure(1)
    plot(x1,x2,'.','MarkerSize',10)
    hold on
    plot(w1,w2,'.','MarkerSize',30,'Color',[0.4 0.4 0.6])
    plot(w1,w2,'b','linewidth',1)
    plot(w1',w2','b','linewidth',1)
    hold off
    title(['t=' num2str(t),' times']);
    drawnow
```

```
end
hold on
c =[1 0.5256 0;1 0 1; 0 1 1;1 0 0; 0 1 0;0 0 1;0 0 0;0.8500 0.3250 0.0980;0.4940 0.1840
0.5560;0.4660 0.6740 0.1880];
gscatter(data(:,1),data(:,2),idx,c);
```

模型的聚类检验（迭代次数为 1001）



可以看到图中 10 种颜色代表着计算出来的十个区，点代表着每个区的中心。实际差不多 300 次就完成拟合。

5. 模型的评价与推广

5.1. 模型优点

1. 相较于普遍使用的 KMeans 聚类算法，模型运用了邻域相关权重更新，能够有效避免噪声的影响，具有一定的容错性。
2. 通过竞争性学习，训练权重系数后，自动得出各分群的中心。

3. 网络结构相对简单，能够反映输入空间的拓扑分布。

5.2. 模型缺点

1. 需要手动设置调整参数，参数选择不当会导致无法收敛。
2. 需要预先制定聚类数目和初始的权值矩阵。

5.3. 模型改进方向

猜想可以通过预训练的方式，提前获取输入空间的粗略分布，并根据输入空间的分布情况自动形成输入参数。

6. 心得体会与总结

通过这次数学实验项目的完成，我成功尝试将人工智能课程学习的内容通过 MATLAB 进行实现，在使用课程学习的 MATLAB 代码的同时也巩固了在人工智能课程学习的相关知识。虽然最终参考资料粗略地完成了该项目，但其实完成内容在 MATLAB 中有类似的实现函数，我只是完成了原理到代码的转换，并没有达到内置函数那么好的效果。不过在课程的最后，还是很感谢这学期杨老师上课的指导还有助教学长在汇报评测里面对我错误地方的点评，经过这次课程，我受益匪浅。