大家好，这篇是有关台大机器学习课程作业八的详解。

我的github地址：
https://github.com/Doraemonzzz

个人主页：
http://doraemonzzz.com/

作业地址:
https://www.csie.ntu.edu.tw/~htlin/course/ml15fall/

参考资料:
https://blog.csdn.net/a1015553840/article/details/51085129
http://www.vynguyen.net/category/study/machine-learning/page/6/
http://book.caltech.edu/bookforum/index.php
http://beader.me/mlnotebook/
https://blog.csdn.net/qian1122221/article/details/50130093
https://acecoooool.github.io/blog/

Problem 1

考虑一般情形，假设一共有L层，第l层有$d^{(\ell)}$个节点，激活函数为$\theta(x)$，损失函数为$J(x,y)$。

首先要进行前向传播，那么第$\ell-1$层到第l层一共需要的计算次数为

$$(d^{(\ell-1)}+1)\times d^{(\ell)}$$

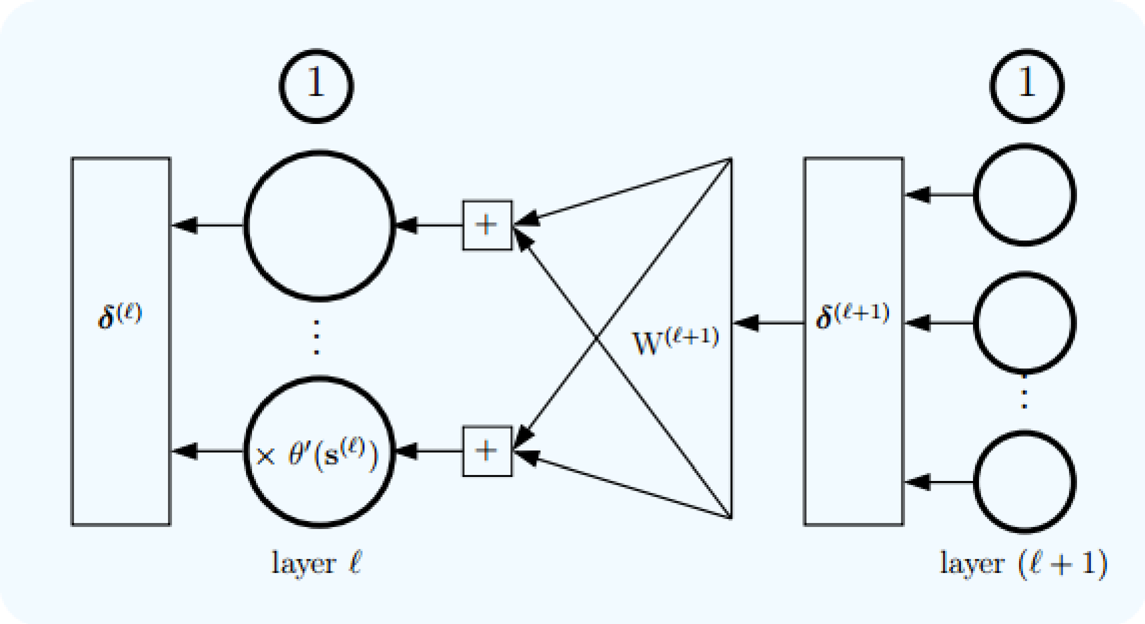
所以前向传播需要的计算次数为

$$\sum_{\ell=1}^{L}(d^{(\ell-1)}+1)\times d^{(\ell)}$$

对于此题来说，前向传播需要的计算次数为

$$(A+1)\times B+(B+1)\times 1$$

接着进行反向传播，先看计算图



接着回顾公式

$$\frac{\partial e}{\partial w_{ij}^{(\ell)}}=x_i^{(\ell-1)}\delta_j^{(\ell)}$$

$$\delta_j^{(\ell)}=\theta^{'}\left(s_j^{(\ell)}\right)\sum_{k=1}^{d^{(\ell+1)}}w_{jk}^{(\ell+1)}\delta_k^{(\ell+1)}$$

从上述递推式可以看到，计算$\delta_j^{(\ell)}$需要的总计算次数和前向传播是一样的，计算$\frac{\partial e}{\partial w_{ij}^{(\ell)}}$总共需要的次数为第1层到第$L-1$层的节点总数（注意$\delta_j^{(L)}$是直接利用求导计算出来的，在这题中计算次数不计算在内），所以总共需要计算的次数为

$$\sum_{\ell=0}^{L}(d^{(\ell-1)}+1)\times d^{(\ell)}+\sum_{\ell=1}^{L-1}d^{(\ell)}$$

对于此题来说，反向传播需要的计算次数为

$$(A+1)\times B+(B+1)\times 1+B$$

所以进行一轮随机梯度下降法总共需要的次数为

$$(A+1)\times B+(B+1)\times 1+(A+1)\times B+(B+1)\times 1+B=2AB+5B+2$$

Problem 2

假设一共有L层，第ℓ层有$d^{(\ell)}$个节点（包括偏置项）。

注意此题每一层的节点数量算上了偏置项1，但是偏置项只有输出，没有输入，所以第ℓ层到第$\ell+1$层的权重数量为

$$d^{(\ell)}\times(d^{(\ell+1)}-1)$$

此题$d^{(0)}=10,d^{(L)}=1$（注意这里我和题目中的$d^{(0)}$不一致，我的$d^{(0)}$是包括偏置项的），所以我们需要计算的量为

$$S=\sum_{\ell=0}^{L-2}d^{(\ell)}\times(d^{(\ell+1)}-1)+d^{(L-1)}$$

如果我们将$d^{(L)}$改为2，那么上述求和式就更整齐

$$S=\sum_{\ell=0}^{L-1}d^{(\ell)}\times(d^{(\ell+1)}-1)=\sum_{\ell=0}^{L-1}d^{(\ell)}d^{(\ell+1)}-\sum_{\ell=0}^{L-1}d^{(\ell)}$$

现在的约束条件为

$$d^{(0)}=10,d^{(L)}=2,\sum_{\ell=1}^{L-1}d^{(\ell)}=36,d^{(\ell)}\geq 2$$

所以我们只要考虑

$$S_1=\sum_{\ell=0}^{L-1}d^{(\ell)}d^{(\ell+1)}$$

这里要解释一下最后一个约束条件$d^{(\ell)}\geq 2$：假设有L层就要求每一层至少有一个神经元，加上偏置项每一层就至少有两个神经元。

这个问题很难，没法用穷举法，因为数量太多了，一定要注意，这里$(10,2,34,2)$和$(10,34,2,2)$是两种结构，之前网上参考的答案就没有考虑次序问题，所以答案是不对的。这里直接给出最小值的答案，来自Learning from data作业6的习题9,10。

最小值发生在一共19层，第1层到18层都是2个节点，所以权重的数量为

$$10+2\times 18=46$$

Problem 3

最大值发生在一共4层，设第1层有x个节点，第2层有y个节点，$x+y=36$，由上一题的讨论，我们要求下式的最大值

$$\begin{aligned}T&=10x+xy+2y\\&=10x+x(36-x)+2(36-x)\\&=10x+36x-x^2+72-2x\\&=-x^2+44x+72\end{aligned}$$

所以当$x=22$时，上式取最大值，所以神经网络的结构为$(10,22,14,1)$，权重的数量为

$$10\times(22-1)+22\times(14-1)+14=510$$

Problem 4

先对$\mathrm{err}(w)$进行化简

$$
\begin{aligned}
\mathrm{err}(w) &= ||x_n - ww^T x_n||^2 \\
&= (x_n - ww^T x_n)^T (x_n - ww^T x_n) \\
&= x_n^T x_n - 2(w^T x_n)^T w^T x_n + x_n^T ww^T ww^T x_n \\
&= x_n^T x_n - 2(w^T x_n)^2 + (x_n^T w)(w^T w)(w^T x_n)(\text{注意 } w^T x_n = x_n^T w\text{，并且都为一个数}) \\
&= x_n^T x_n - 2(w^T x_n)^2 + (x_n^T w)^2 (w^T w)
\end{aligned}
$$

接着求梯度

$$
\begin{aligned}
\nabla_w \mathrm{err}(w) &= \nabla_w \Big(x_n^T x_n - 2(w^T x_n)^2 + (x_n^T w)^2 (w^T w) \Big) \\
&= -4 w^T x_n \nabla_w (w^T x_n) + 2 x_n^T w (w^T w) \nabla_w (x_n^T w) + (x_n^T w)^2 \nabla (w^T w) \\
&= -4 w^T x_n x_n + 2 x_n^T w (w^T w) x_n + 2 (x_n^T w)^2 w
\end{aligned}
$$

Problem 5

$$
\begin{aligned}
E_{\mathrm{in}}(w) &= \frac{1}{N}\sum_{n=1}^{N} \|x_n - ww^T(x_n+\epsilon_n)\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} \big(x_n - ww^T(x_n+\epsilon_n)\big)^T \big(x_n - ww^T(x_n+\epsilon_n)\big) \\
&= \frac{1}{N}\sum_{n=1}^{N} \Big(x_n^T x_n - 2(x_n+\epsilon_n)^T ww^T x_n + (x_n+\epsilon_n)^T ww^T ww^T (x_n+\epsilon_n) \Big) \\
&= \frac{1}{N}\sum_{n=1}^{N} \Big(x_n^T x_n - 2x_n^T ww^T x_n - 2\epsilon_n^T ww^T x_n + x_n^T ww^T ww^T x_n + \epsilon_n^T ww^T ww^T \epsilon_n + 2\epsilon_n^T ww^T ww^T x_n \Big) \\
&= \frac{1}{N}\sum_{n=1}^{N} \Big(x_n^T x_n - 2x_n^T ww^T x_n + x_n^T ww^T ww^T x_n \Big) + \frac{1}{N}\sum_{n=1}^{N} \Big(-2\epsilon_n^T ww^T x_n + \epsilon_n^T ww^T ww^T \epsilon_n + 2\epsilon_n^T ww^T ww^T x_n \Big) \\
&= \frac{1}{N}\sum_{n=1}^{N} \|x_n - ww^T x_n\|^2 + \frac{1}{N}\sum_{n=1}^{N} \Big(-2\epsilon_n^T ww^T x_n + \epsilon_n^T ww^T ww^T \epsilon_n + 2\epsilon_n^T ww^T ww^T x_n \Big)
\end{aligned}
$$

由于ϵ_n服从标准正态分布，所以

$$
\mathbb{E}\Big[\epsilon_n \epsilon_n^T\Big] = I_n
$$

对上式取期望可得

$$
\begin{aligned}
\mathbb{E}[E_{\text{in}}(w)] &= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\left[\epsilon_n^T ww^T ww^T \epsilon_n\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\text{trace}\Big(\mathbb{E}\left[\epsilon_n^T ww^T ww^T \epsilon_n\right]\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\Big[\text{trace}\Big(\epsilon_n^T ww^T ww^T \epsilon_n\Big)\Big] \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\Big[\text{trace}\Big(ww^T \epsilon_n \epsilon_n^T ww^T\Big)\Big] \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\text{trace}\Big(\mathbb{E}\Big[ww^T \epsilon_n \epsilon_n^T ww^T\Big]\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\text{trace}\Big(ww^T\mathbb{E}\Big[\epsilon_n \epsilon_n^T\Big] ww^T\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\text{trace}\Big(ww^T ww^T\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + \frac{1}{N}\sum_{n=1}^{N}\text{trace}\Big(w^T ww^T w\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}||x_n - ww^Tx_n||^2 + (w^Tw)^2
\end{aligned}
$$

因此

$$
\Omega(w) = (w^Tw)^2
$$

Problem 6

由几何意义可知，两个点的$1NN$算法推导出来的边界为过这两个点中点的中垂面，所以可得超平面方程为：

$$
\begin{aligned}
(x^+ - x^-)^T(x - \frac{x^+ + x^-}{2}) &= 0 \\
(x^+ - x^-)^T x - \frac{(x^+ - x^-)^T(x^+ + x^-)}{2} &= 0 \\
(x^+ - x^-)^T x - \frac{||x^+||^2 - ||x^-||^2}{2} &= 0
\end{aligned}
$$

从而分类器为

$$
g_{\text{LIN}}(x) = \text{sign}\left((x^+ - x^-)^T x - \frac{||x^+||^2 - ||x^-||^2}{2}\right)
$$

Problem 7

由$\text{sign}(x)=\text{sign}(ax)(a>0)$可得:

$$
\begin{aligned}
g_{\text{RBFNET}}(x) &= \text{sign}\left(\beta_+ \exp(-\|x-\mu_+\|^2)+\beta_- \exp(-\|x-\mu_-\|^2)\right)\\
&= \text{sign}\left(\exp(-\|x-\mu_-\|^2)\left(\beta_+ \exp(\|x-\mu_-\|^2-\|x-\mu_+\|^2)+\beta_-\right)\right)\\
&= \text{sign}\left(\beta_+ \exp(\|x-\mu_-\|^2-\|x-\mu_+\|^2)+\beta_-\right)
\end{aligned}
$$

我们来计算$||x-\mu_-||^2-||x-\mu_+||^2$

$$
\begin{aligned}
||x-\mu_-||^2-||x-\mu_+||^2 &= (x-\mu_-)^T(x-\mu_-)-(x-\mu_+)^T(x-\mu_+)\\
&= x^Tx-2\mu_-^Tx+\mu_-^T\mu_- -(x^Tx-2\mu_+^Tx+\mu_+^T\mu_+)\\
&= 2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+
\end{aligned}
$$

所以

$$
g_{\text{RBFNET}}(x)=\text{sign}\left(\beta_+ \exp(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+)+\beta_-\right)
$$

我们来求解$\beta_+ \exp(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+)+\beta_- >0$

$$
\begin{aligned}
\beta_+ \exp(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+)+\beta_- >0 \Leftrightarrow\\
\exp(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+) > -\frac{\beta_-}{\beta_+} \Leftrightarrow\\
2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+ > \ln(-\frac{\beta_-}{\beta_+}) \Leftrightarrow\\
2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+ -\ln(-\frac{\beta_-}{\beta_+})>0
\end{aligned}
$$

同理可得

$$
\begin{aligned}
\beta_+ \exp(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+)+\beta_- <0 \Leftrightarrow\\
2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+ -\ln(-\frac{\beta_-}{\beta_+})<0
\end{aligned}
$$

从而

$$
g_{\text{RBFNET}}(x)=\text{sign}\left(2(\mu_+-\mu_-)^Tx+\mu_-^T\mu_- -\mu_+^T\mu_+ -\ln(-\frac{\beta_-}{\beta_+})\right)
$$

Problem 8

回顾定义:

$$
z_n=[\text{RBF}(x_n,x_1),\text{RBF}(x_n,x_2),\ldots,\text{RBF}(x_n,x_N)]
$$

由于

$$\text{RBF}(x,\mu)=[[x=\mu]]$$

所以z_n为第n个元素为1，其余元素均为0的向量，从而$Z=I$，回顾β的计算公式

$$\beta=(Z^TZ)^{-1}Z^Ty=y$$

Problem 9

回顾我们的计算式：

$$\begin{aligned}\min_{W,V}E_{in}(w_m,v_n)&\propto\sum_{\text{user n rated movie m}}\Big(r_{nm}-w_m^Tv_n\Big)^2\\&=\sum_{m=1}^{M}\Big(\sum_{(x_n,r_{nm})\in\mathcal{D}_m}\Big(r_{nm}-w_m^Tv_n\Big)^2\Big)\end{aligned}$$

现在固定了v_n，我们要对每个m，求$\sum_{(x_n,r_{nm})\in\mathcal{D}_m}\Big(r_{nm}-w_m^Tv_n\Big)^2$的最小值，计算其梯度：

$$\nabla_{w_m}\sum_{(x_n,r_{nm})\in\mathcal{D}_m}\Big(r_{nm}-w_m^Tv_n\Big)^2=\sum_{(x_n,r_{nm})\in\mathcal{D}_m}-2\Big(r_{nm}-w_m^Tv_n\Big)v_n$$

令式为0可得

$$\begin{aligned}\sum_{(x_n,r_{nm})\in\mathcal{D}_m}\Big(r_{nm}-w_m^Tv_n\Big)v_n&=0\\\sum_{(x_n,r_{nm})\in\mathcal{D}_m}r_{nm}v_n&=\sum_{(x_n,r_{nm})\in\mathcal{D}_m}(w_m^Tv_n)v_n\end{aligned}$$

注意此处$\tilde d=1$，从而w_m,v_n都为实数，并且由题意，$v_n=1$，从而

$$\begin{aligned}\sum_{(x_n,r_{nm})\in\mathcal{D}_m}r_{nm}&=\sum_{(x_n,r_{nm})\in\mathcal{D}_m}w_m\\w_m&=\frac{\sum_{(x_n,r_{nm})\in\mathcal{D}_m}r_{nm}}{\sum_{(x_n,r_{nm})\in\mathcal{D}_m}1}\end{aligned}$$

这个量即为电影的平均分数。

Problem 10

根据定义计算v_{N+1}^T

$$v_{N+1}^Tw_m=\frac{1}{N}\sum_{n=1}^{N}v_n^Tw_m=\frac{1}{N}\sum_{n=1}^{N}r_{nm}$$

所以我们要找使得$\frac{1}{N}\sum_{n=1}^{N}r_{nm}$最大的电影，即评分最高的电影。

Problem 11-12

注意KNN中我们需要计算每个$||x^{(i)}-x^{(j)}||^2$，这就涉及到计算效率的问题，下面介绍如何使用向量化的方法来实现knn。

我们假设

$$X=\begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}\in \mathbb{R}^{m\times d}, Y=\begin{bmatrix} —(y^{(1)})^T— \\ —(y^{(2)})^T— \\ \vdots \\ —(y^{(n)})^T— \end{bmatrix}\in \mathbb{R}^{n\times d}$$

其中$x^{(i)},y^{(i)}\in\mathbb{R}^d$，现在的问题是如何高效计算矩阵$D\in\mathbb{R}^{m\times n}$，其中

$$D_{i,j}=||x^{(i)}-y^{(j)}||^2$$

首先对$D_{i,j}$进行处理

$$\begin{aligned} D_{i,j} &= ||x^{(i)}-y^{(j)}||^2 \\ &= (x^{(i)}-y^{(j)})^T(x^{(i)}-y^{(j)}) \\ &= (x^{(i)})^Tx^{(i)}-2(x^{(i)})^Ty^{(j)}+(y^{(j)})^Ty^{(j)} \end{aligned}$$

那么

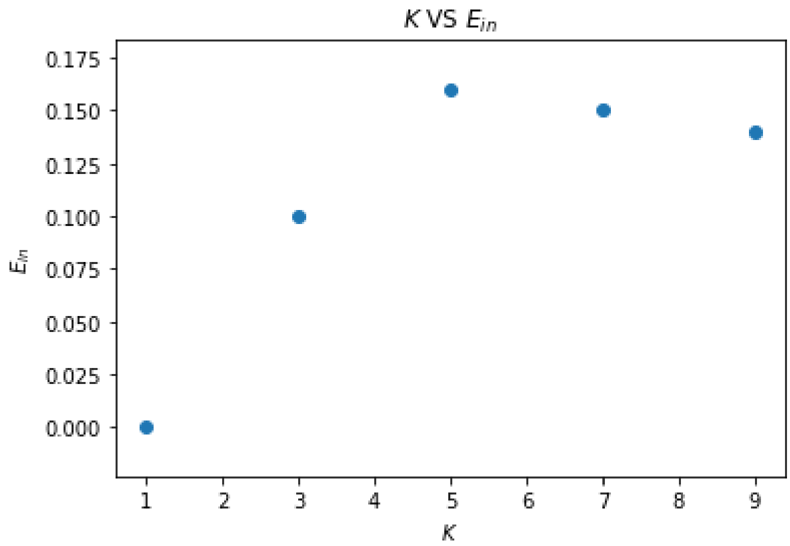
$$\begin{aligned} D &= \begin{bmatrix} D_{1,1} & \ldots & D_{1,n} \\ \ldots & \ldots & \ldots \\ D_{m,1} & \ldots & D_{m,n} \end{bmatrix} \\ &= \begin{bmatrix} (x^{(1)})^Tx^{(1)}-2(x^{(1)})^Ty^{(1)}+(y^{(1)})^Ty^{(1)} & \ldots & (x^{(1)})^Tx^{(1)}-2(x^{(1)})^Ty^{(n)}+(y^{(n)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Tx^{(m)}-2(x^{(m)})^Ty^{(1)}+(y^{(1)})^Ty^{(1)} & \ldots & (x^{(m)})^Tx^{(m)}-2(x^{(m)})^Ty^{(n)}+(y^{(n)})^Ty^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} (x^{(1)})^Tx^{(1)} & \ldots & (x^{(1)})^Tx^{(1)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Tx^{(m)} & \ldots & (x^{(m)})^Tx^{(m)} \end{bmatrix} + \begin{bmatrix} (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \end{bmatrix} - 2\begin{bmatrix} (x^{(1)})^Ty^{(1)} & \ldots & (x^{(1)})^Ty^{(n)} \\ \ldots & \ldots & \ldots \\ (x^{(m)})^Ty^{(1)} & \ldots & (x^{(m)})^Ty^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} (x^{(1)})^Tx^{(1)} \\ \ldots \\ (x^{(m)})^Tx^{(m)} \end{bmatrix} \underbrace{\begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}}_{1\times n\text{矩阵}} + \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{m\times 1\text{矩阵}} \begin{bmatrix} (y^{(1)})^Ty^{(1)} & \ldots & (y^{(n)})^Ty^{(n)} \end{bmatrix} - 2XY^T \end{aligned}$$

利用numpy的广播机制上式可以简写如下：

```
#计算距离矩阵
d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
d2 = np.sum(cluster_centers ** 2, axis=1).reshape(1, -1)
dist = d1 + d2 - 2 * X.dot(cluster_centers.T)
```

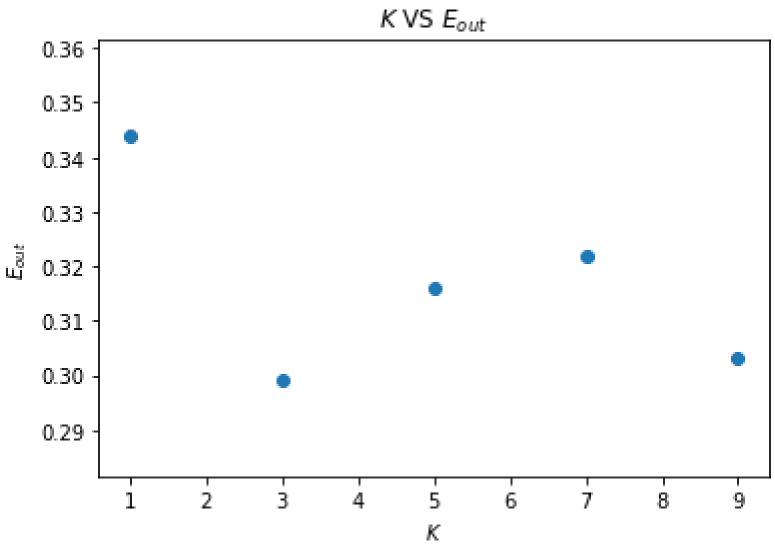
完整代码如下：

```
import numpy as np
import matplotlib.pyplot as plt

#读取数据
train = np.loadtxt("hw8_train.dat")
test = np.loadtxt("hw8_test.dat")
X_train, y_train = train[:, :-1], train[:, -1]
X_test, y_test = test[:, :-1], test[:, -1]

class KNeighborsClassifier_():
    def __init__(self, n_neighbors):
        self.n_neighbors = n_neighbors


    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, X):
        #计算距离矩阵
        d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
        d2 = np.sum(self.X ** 2, axis=1).reshape(1, -1)
        dist = d1 + d2 - 2 * X.dot(self.X.T)

        #找到最近的k个点的索引
        index = np.argsort(dist, axis=1)[:, :self.n_neighbors]
        #计算预测结果
        y = np.sign(np.sum(self.y[index], axis=1))

        return y

# Q11-12
K = [1, 3, 5, 7, 9]
Ein = []
for k in K:
    #训练模型
    knn = KNeighborsClassifier_(n_neighbors=k)
    knn.fit(X_train, y_train)
    #预测结果
    y = knn.predict(X_train)
    ein = np.mean(y != y_train)
    Ein.append(ein)

plt.scatter(K, Ein)
plt.title("$K$ vs $E_{in}$")
plt.xlabel("$K$")
plt.ylabel("$E_{in}$")
plt.show()
```



Problem 13-14

```
# Q13-14
K = [1, 3, 5, 7, 9]
Eout = []
for k in K:
    #训练模型
    knn = KNeighborsClassifier_(n_neighbors=k)
    knn.fit(X_train, y_train)
    #预测结果
    y = knn.predict(X_test)
    eout = np.mean(y != y_test)
    Eout.append(eout)

plt.scatter(K, Eout)
plt.title("$K$ VS $E_{out}$")
plt.xlabel("$K$")
plt.ylabel("$E_{out}$")
plt.show()
```



Problem 15-16

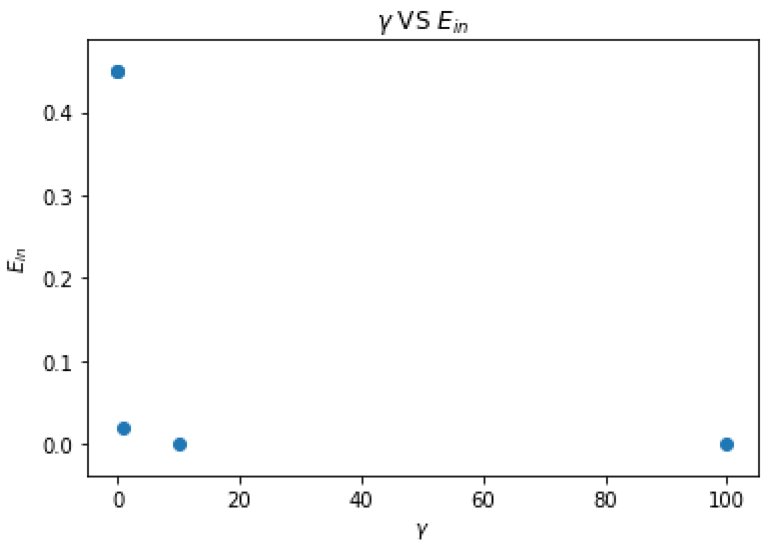
回顾计算公式：

$$g_{\text{uniform}}(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{N} y_m \exp\left(-\gamma\|\mathbf{x}-\mathbf{x}_m\|^2\right)\right)$$

这里依旧可以使用之前介绍的向量化方法：

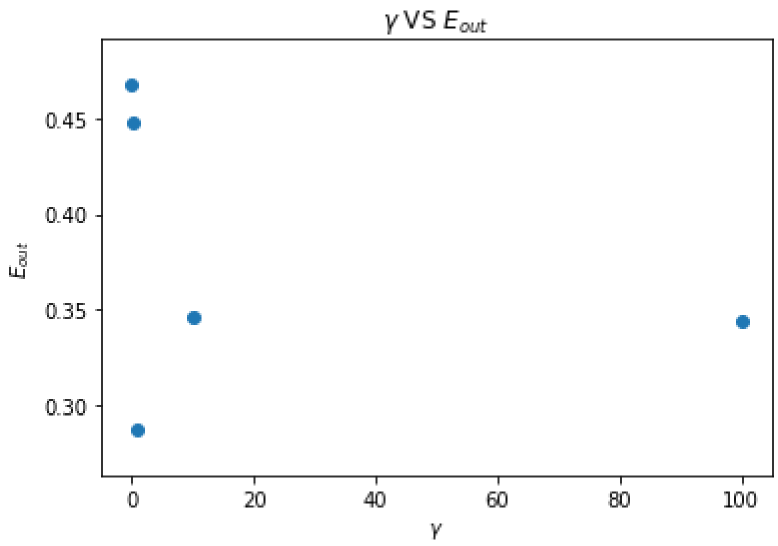
```
class RBFNetworkClassifier():
    def __init__(self, gamma):
        self.gamma = gamma
        self.beta = None

    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, X):
        #计算距离矩阵
        d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
        d2 = np.sum(self.X ** 2, axis=1).reshape(1, -1)
        dist = d1 + d2 - 2 * X.dot(self.X.T)

        #计算exp(-gamma*dist)
        d = np.exp(-self.gamma * dist)

        #计算预测结果
        y = np.sign(np.sum(d * self.y, axis=1))

        return y

# Q15-16
Gamma = [0.001, 0.1, 1, 10, 100]
```

```
Ein_rbf = []
for gamma in Gamma:
    #训练模型
    knn = RBFNetworkClassifier(gamma=gamma)
    knn.fit(X_train, y_train)
    #预测结果
    y = knn.predict(X_train)
    ein = np.mean(y != y_train)
    Ein_rbf.append(ein)

plt.scatter(Gamma, Ein_rbf)
plt.title("$\gamma$ VS $E_{in}$")
plt.xlabel("$\gamma$")
plt.ylabel("$E_{in}$")
plt.show()
```



Problem 17-18

```
# Q17-18
Gamma = [0.001, 0.1, 1, 10, 100]
Eout_rbf = []
for gamma in Gamma:
    #训练模型
    knn = RBFNetworkClassifier(gamma=gamma)
    knn.fit(X_train, y_train)
    #预测结果
    y = knn.predict(X_test)
    eout = np.mean(y != y_test)
    Eout_rbf.append(eout)

plt.scatter(Gamma, Eout_rbf)
plt.title("$\gamma$ VS $E_{out}$")
plt.xlabel("$\gamma$")
plt.ylabel("$E_{out}$")
plt.show()
```



Problem 19-20

均值聚类算法如下：

- 1.随机初始化几个聚类中心$\mu_1,\ldots,\mu_k\in\mathbb{R}^n$
- 2.重复如下操作直到收敛：{
  - 对每个i，令

$$c^{(i)} := \arg\min_j ||x^{(i)}-\mu_j||^2$$

  - 对每个j，令

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)}=j\}}$$

}

算法的内层循环重复两个步骤：

1. 把每个训练样本$x^{(i)}$"分配"给距离最近的聚类中心。
2. 将聚类中心μ_j移动到所分配的样本点的中心。

上述算法中的距离是欧式距离，实际中别的距离也可以使用，但欧式距离应用的较多。

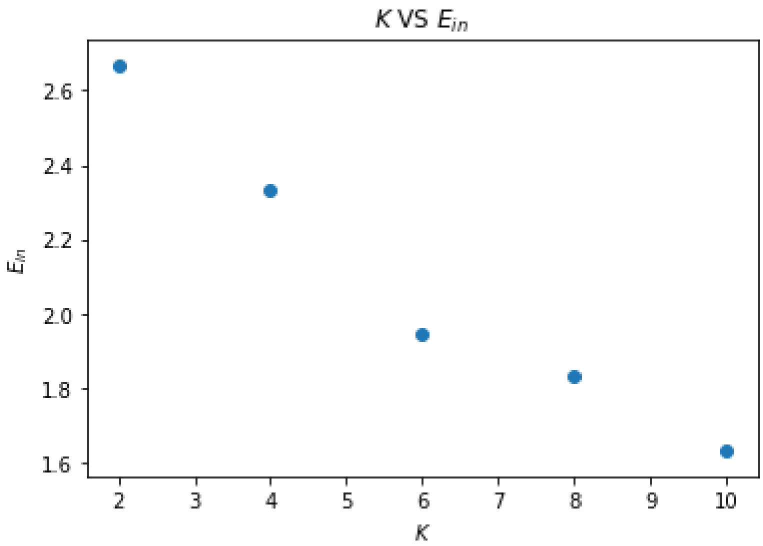
（备注，k是人为选择的参数。）

这里使用之前介绍的向量化方法，完整代码如下：

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 19 16:24:00 2019

@author: qinzhen
"""
```

```python
import numpy as np
import matplotlib.pyplot as plt
X = np.genfromtxt("hw8_nolabel_train.dat")

class KMeans_():
    def __init__(self, k, D=1e-5):
        #聚类数量
        self.k = k
        #聚类中心
        self.cluster_centers_ = []
        #聚类结果
        self.labels_ = []
        #设置阈值
        self.D = D

    def fit(self, X):
        #数据维度
        n, d = X.shape
        #聚类标签
        labels = np.zeros(n, dtype=int)
        #初始中心点
        index = np.random.randint(0, n, self.k)
        cluster_centers = X[index]
        #记录上一轮迭代的聚类中心
        cluster_centers_pre = np.copy(cluster_centers)

        while True:
            #计算距离矩阵
            d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
            d2 = np.sum(cluster_centers ** 2, axis=1).reshape(1, -1)
            dist = d1 + d2 - 2 * X.dot(cluster_centers.T)

            #STEP1:找到最近的中心
            labels = np.argmin(dist, axis=1)
            #STEP2:重新计算中心
            for i in range(self.k):
                #第i类的索引
                index = (labels==i)
                #第i类的数据
                x = X[index]
                #判断是否有点和某聚类中心在一类
                if len(x) != 0:
                    cluster_centers[i] = np.mean(x, axis=0)

            #计算误差
            delta = np.linalg.norm(cluster_centers - cluster_centers_pre)

            if delta < self.D:
                break

            cluster_centers_pre = np.copy(cluster_centers)
```

```
        self.cluster_centers_ = np.copy(cluster_centers)
        self.labels_ = labels


    def predict(self, X):
        #计算距离矩阵
        d1 = np.sum(X ** 2, axis=1).reshape(-1, 1)
        d2 = np.sum(self.cluster_centers_ ** 2, axis=1).reshape(1, -1)
        dist = d1 + d2 - 2 * X.dot(self.cluster_centers_.T)

        #找到最近的中心
        self.cluster_centers_ = np.argmin(dist, axis=1)

        return self.cluster_centers_

n = X.shape[0]
K = [2, 4, 6, 8, 10]
Ein = []
for k in K:
    #训练模型
    kmeans = KMeans_(k)
    kmeans.fit(X)
    #获得标签
    label = kmeans.labels_
    #获得聚类中心
    center = kmeans.cluster_centers_
    #计算Ein
    ein = 0
    for i in range(k):
        #计算每一类的误差
        ein += np.sum((X[label==i] - center[i]) ** 2)
    #计算均值
    ein /= n
    Ein.append(ein)

plt.scatter(K, Ein)
plt.title("$K$ vs $E_{in}$")
plt.xlabel("$K$")
plt.ylabel("$E_{in}$")
plt.show()
```



Problem 21

这题一开始没思路，做了下面一题反推了一些思路，但是我证明不出来原命题，只能证明如下更弱的命题：

$$\Delta \ge 2, \text{如果} N \ge 3\Delta \log_2 \Delta, \text{那么} N^{\Delta} + 1 \le 2^N, \\ \text{其中} \Delta, N \text{均为整数}$$

证明：

我们先证明如下结论：

$$\text{当} N = 3\Delta \log_2 \Delta \text{时，} N^{\Delta} < 2^N$$

事实上：

$$\begin{aligned} N^{\Delta} &< 2^N \Leftrightarrow \\ N^{\Delta} &< 2^{3\Delta \log_2 \Delta} \Leftrightarrow \\ N^{\Delta} &< \Delta^{3\Delta} \Leftrightarrow \\ N &< \Delta^3 \Leftrightarrow \\ 3\Delta \log_2 \Delta &< \Delta^3 \Leftrightarrow \\ 3 \log_2 \Delta &< \Delta^2 \end{aligned}$$

当$\Delta \ge 2$时，上式成立。

接着证明

$$\text{当} N \ge 3\Delta \log_2 \Delta \text{时，} N^{\Delta} < 2^N$$

上式等价于取对数之后的情形

$$\Delta \ln N < N \ln 2$$

设$f(x) = \Delta \ln x - x \ln 2$，那么

$$f(3\Delta\log_2\Delta)<0 \tag{1}$$

以及

$$f'(x)=\frac{\Delta}{x}-\ln 2=0\Rightarrow x=\frac{\Delta}{\ln 2}$$
$$\text{所以当}x\ge\frac{\Delta}{\ln 2},f'(x)\le 0;\ \text{当}x<\frac{\Delta}{\ln 2},f'(x)>0$$

因此

$$f(x)\text{在}\left[\frac{\Delta}{\ln 2},\infty\right)\downarrow,\text{在}\left(-\infty,\frac{\Delta}{\ln 2}\right]\uparrow \tag{2}$$

注意

$$3\Delta\log_2\Delta=\frac{3\Delta\ln\Delta}{\ln 2},\Delta\ge 2$$

所以

$$3\Delta\log_2\Delta=\frac{3\Delta\ln\Delta}{\ln 2}\ge\frac{\Delta}{\ln 2}$$

因此由(1)(2)可得当$x\ge 3\Delta\log_2\Delta$时：

$$f(x)\le f(3\Delta\log_2\Delta)<0,$$

即

$$N\ge 3\Delta\log_2\Delta\text{时，}N^{\Delta}<2^N$$

又因为这里假定Δ,N都为整数，所以

$$N^{\Delta}+1\le 2^N$$

Problem 22

因为第1层到第2层的变换是固定的，所以至少考虑第1层能产生多少个$0,1$的组合就行。

回顾机器学习基石第6讲的函数$B(N,k)$

$$B(N,k)\text{表示 break point为}k\text{时，}N\text{个点能表示的最多组合数量。}$$

我们知道d维感知机（不包括偏置项）的break point为$d+1$，所以如果有N组数据，那么最多能够表示的组合数量小于等于

$$B(N,d+1)$$

对于$d-3-1$架构的神经网络，第0层到第1层的每个节点相当于感知机，表示的组合数量最多为$B(N,d+1)$，所以3个节点能够表示的组合数量小于等于

$$B(N,d+1)^3$$

回顾机器学习基石第6讲，有如下不等式估计

$$B(N,k) \le \sum_{i=0}^{k-1} \binom{N}{i}$$

Learning from data第二章Problem 2.5（可以参考我的解答，传送门）给出如下结论

$$\sum_{i=0}^{D} \binom{N}{i} \le N^D + 1$$

所以

$$B(N,d+1) \le \sum_{i=0}^{d} \binom{N}{i} \le N^d + 1 \le N^{d+1} + 1$$

从而3个节点能够表示的组合数量小于等于

$$(N^{d+1}+1)^3 = N^{3(d+1)} + 3N^{2(d+1)} + 3N^{d+1} + 1$$

令$\Delta = 3(d+1)+1 \ge 4$，取$N \ge \Delta \ge 4$，那么

$$\begin{aligned}(N^{d+1}+1)^3 &= N^{3(d+1)} + 3N^{2(d+1)} + 3N^{d+1} + 1 \\ &< N^{3(d+1)} + N^{3(d+1)} + N^{3(d+1)} + 1 \\ &= 3N^{3(d+1)} + 1 \\ &< N^{3(d+1)+1} + 1\end{aligned}$$

如果$N \ge 3\Delta \log_2 \Delta$，那么满足Problem 21的条件，所以

$$(N^{d+1}+1)^3 < N^{3(d+1)+1} + 1 \le 2^N$$

从而无法shatter大于等于$3\Delta \log_2 \Delta$个点，所以

$$VC < 3\Delta \log_2 \Delta = 3(3(d+1)+1)\log_2(3(d+1)+1)$$

所以命题成立。