EE7207 Assignment1

ZHANG ZHENGHANG

E-mail: ZZHANG062@e.ntu.edu.sg

1.The process of building a RBF model:

1. The suitable number of hidden layer neurons

   center_num=10

2. Neuron center determination

   I use the K-means clustering to select prototypes of the training samples.

3. the radial basis function

   Gaussian function: $g(d) = \exp(-\frac{d^2}{2\sigma^2})$ .d means the distance of sample and the neuron centre.

   σ determines the size of receptive field, which is equal to $d_{\max} / \sqrt{2m}$.

4. weight estimation

   $W = (\phi^T \phi)^{-1} \phi^T d$

Data preprocessing:

Totally, there are 330 samples in the TRAIN set. I divided the set into two parts. The 80% samples used as train samples and the other 20% samples used to judge the performance of the model.

Python code:

```
class RBF:
    def __init__(self, input_dim, num_centers, out_dim):  # 输入变量数 中间层个数 输出变量数目
        self.input_dim = input_dim
        self.num_center = num_centers
        self.out_dim = out_dim
        self.beta = 0.1  # 标准差初始化定义
        self.centers = [np.random.uniform(-1, 1, input_dim) for i in range(num_centers)]  # 中心点初始赋值
        """
        中心点个数和number_centers个数相同，坐标数量和input_dim对应
        """
        self.W = np.random.random((self.num_center, self.out_dim))  # num_centers*out_dim矩阵形式

    def _basisfunc(self, c, d):  # 基函数计算
        return np.exp(-(self.num_center * (norm(c - d) ** 2)) / self.beta ** 2)  #高斯内核##

    def _calcAct(self, x):   # 定义一个激活矩阵
        G = np.zeros((x.shape[0], self.num_center), dtype=np.float)  # 样本数*中间层数##
        for ci, c in enumerate(self.centers):
            for x1, xx in enumerate(x):
                G[x1, ci] = self._basisfunc(c, xx)
        return G

    def _calcbeat(self):  # 找到选取中心点最大值——及求解σ的值
        bate_temp = np.zeros((self.num_center, self.num_center))  # 定义一个矩阵 隐藏层中心值确定的
        for iindex, ivalue in enumerate(self.centers):
            for jindex, jvalue in enumerate(self.centers):
                bate_temp[iindex, jindex] = norm(ivalue - jvalue)  # 依次求解各中心层的值
        return np.max(bate_temp)  # 返回两点间距离最大值的索引值
```

```
def train(self, x, y):
    """
    :param x: 样本数*输入变量数
    :param y: 样本数*输出变量数
    :return:无
    """
    rnd_idx = np.random.permutation(x.shape[0])[:self.num_center]  # 随机我们的输入样本
    self.centers = [x[i, :] for i in rnd_idx]  # 根据随机值找打对应样本中心
    self.beta = self._calcbeat()/(sqrt(2*self.num_center))  # 4410   #    #self._calcbeat()   #3400 # 根据样本中心计算σ值
    print(self.beta)
    G = self._calcAct(x)  # 返回G值
    self.W = np.dot(pinv(G), y)  # 求解W值
    # print(self.W)
    pass

def test(self, x):
    G = self._calcAct(x)
    y = np.dot(G, self.W)
    return y
```

```
if __name__ == '__main__':
    x = loadmat("data_train.mat")["data_train"]  # 330*33
    y = loadmat("label_train.mat")["label_train"]  # 330*1
    x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, random_state=1, test_size=0.2)
    #x_test=loadmat("data_test.mat")["data_test"]   #21*33

#x_train
    # rbf regression
    rbf = RBF(264, 30, 264)
    rbf.train(x_train, y_train)
 ###训练

    y_hat= rbf.test(x_test)
    ture=0  #正确数量
    false=0
    for i in range(66):
        if (y_test[i]>0 and y_hat[i]>0) or (y_test[i]<0 and y_hat[i]<0):
            ture+=1
        else:
            false+=1
    test_acc=ture/(ture+false)
    print("测试精准度：")
    print(test_acc)
```

2.The process of building a SVM model:

C means penalty parameter. The larger C is, it is equivalent to penalizing the slack variable. It is hoped that the slack variable will be close to 0, that is, the penalty for misclassification will increase, which tends to be the case of fully splitting the training set, so that the accuracy of the training set is very high, but generalization Ability is weak. The C value is small, the penalty for misclassification is reduced, fault tolerance is allowed, they are regarded as noise points, and the generalization ability is strong.
Therefore, I assume it equal to 0.8.

Gamma is a parameter of the Gaussian Kernel Function used in SVM. In python, it is usually set as ‘auto’. Through a series test on the validation set, I found it is better to set Gamma equal to 0.4.

Data preprocessing:

The same as RBF part.

Python code:

```
from scipy.io import loadmat
from sklearn import svm
from sklearn import model_selection

x = loadmat("data_train.mat")["data_train"] #330*33
y = loadmat("label_train.mat")["label_train"]  #330*1
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,random_state=1,test_size=0.2)

clf = svm.SVC(C=0.8, kernel='rbf', gamma=0.4, decision_function_shape='ovr')
clf.fit(x_train, y_train.ravel())

def show_accuracy(y_hat,y_train,str):
    pass

print("SVM-输出训练集的准确率为：",clf.score(x_train,y_train))
y_hat=clf.predict(x_train)
show_accuracy(y_hat,y_train,'训练集')

y_hat=clf.predict(x_test)
ture=0
false=0
for i in range(66):
    if(y_hat[i]==y_test[i]): ture+=1
    else: false+=1
acc=ture/(ture+false)
print("SVM-输出验证集的准确率为：")
print(acc)
```

3.Result and analysis:

A.RBF

```
D:\python\python.exe "C:/Users/86198/Desktop/NN ass/RBF2.py"
训练集精准度：
[0.9280303]
```

```
D:\python\python.exe "C:/Users/86198/Desktop/NN ass/RBF2.py"
验证集精准度：
0.9090909090909091
```

B.SVM

```
D:\python\python.exe "C:/Users/86198/Desktop/NN ass/SVM.py"
SVM-输出训练集的准确率为：  0.9924242424242424
SVM-输出验证集的准确率为：
0.9696969696969697
```

accuracy	RBF	SVM
Train data	**92.80%**	**99.24%**
Validation data	**90.90%**	**96.96%**

SVM has a better performance no matter on train data or validation data, which means the SVM has high prediction accuracy and good generalization ability in the mean time.

Because the underlying theory of SVM avoids the need for heuristics often used in the conventional RBF.

And in Gaussian kernel SVM, the number of the neurons and their centre vectors are determined automatically.

4.Predict:

The are 21 samples without labels given in another file. By the above two models, the prediction is shown below.

A.RBF

```
D:\python\python.exe "C:/Users/86198/Desktop/NN ass/RBF2.py"
Label of testing data are:
[1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 1, 1, -1, 1, -1, 1, -1, 1]
```

B.SVM

```
D:\python\python.exe "C:/Users/86198/Desktop/NN ass/SVM1.py"
[ 1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1]
```