exp 2-分类聚类

聚类算法实现

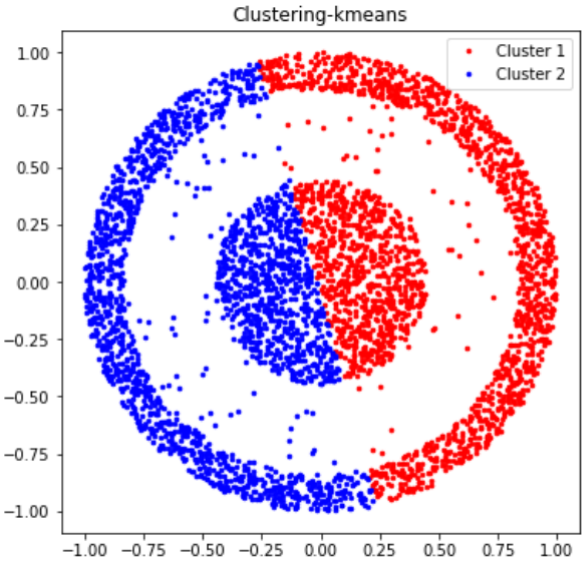
k-means算法

实现代码：

```
def kmeans(X, k):
    N, P = X.shape
    idx = np.zeros(N)
    # YOUR CODE HERE
    # ----------------
    centers = {}
    for i in range(k):
        centers[i] = X[i]
    # update points
    for i in range(1000):
        subsets = {}
        for j in range(k):
            subsets[j] = []
        for point in X:
            dist = []
            for center in centers:
                dist.append(np.linalg.norm(point-centers[center]))
    # update centers
            subsets[dist.index(min(dist))].append(point)
        pre_centers = dict(centers)
        for c in subsets:
            centers[c] = np.average(subsets[c], axis=0)
    # calculate diff
        flag = True
        for c in centers:
            pre_c = pre_centers[c]
            cur_c = centers[c]
            if np.sum((cur_c - pre_c)/pre_c*100) > 0.01:
                flag = False
        if flag:
            break

    for i in range(N):
        dist = []
        for center in centers:
            dist.append(np.linalg.norm(X[i]-centers[center]))
        idx[i] = dist.index(min(dist))
    # ----------------
    return idx
```

效果图：



其中起始中心点为文件中从头开始的k个点。

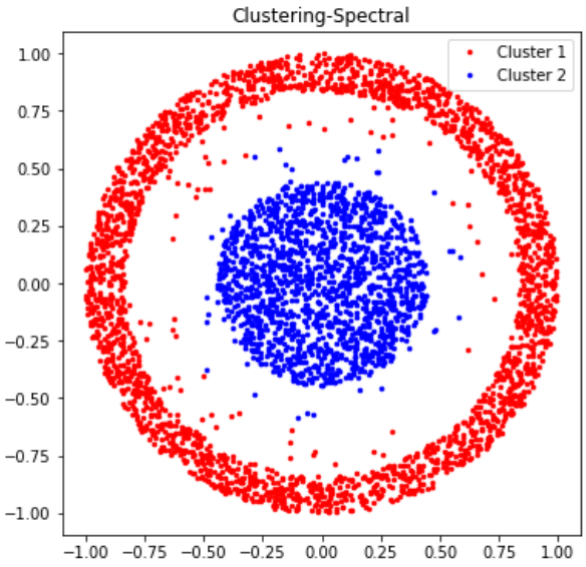
谱聚类算法

实现代码：

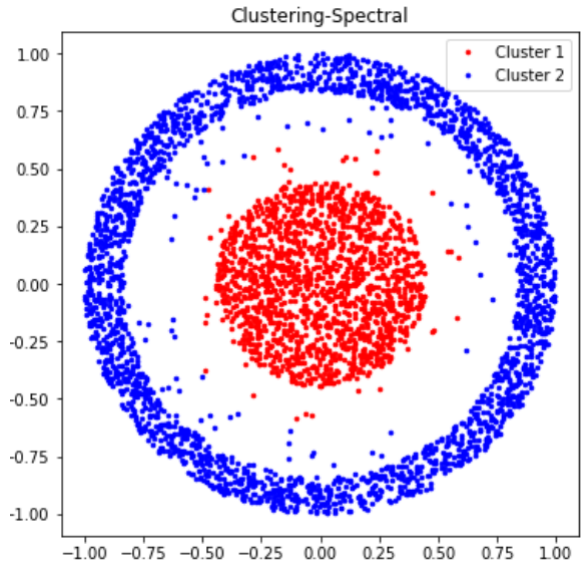
```
def spectral(W, k):
    N = W.shape[0]
    idx = np.zeros((N, 1))
    # YOUR CODE HERE
    # ----------------
    # get degree matrix
    size_ = len(W)
    D = np.diag(np.zeros(size_))
    for i in range(size_):
        D[i][i] = sum(W[i])
    # get L matrix
    L = D-W
    # get eig matrix
    eigval, eigvec = np.linalg.eig(L)
    dim = len(eigval)
    dictEigval = dict(zip(eigval,range(0,dim)))
    kEig = np.sort(eigval)[0:k]
    ix = [dictEigval[k] for k in kEig]
    X = eigvec[:,ix]
    # ----------------
    X = X.astype(float)  # keep real part, discard imaginary part
    idx = kmeans(X, k)
    return idx
```

效果图：

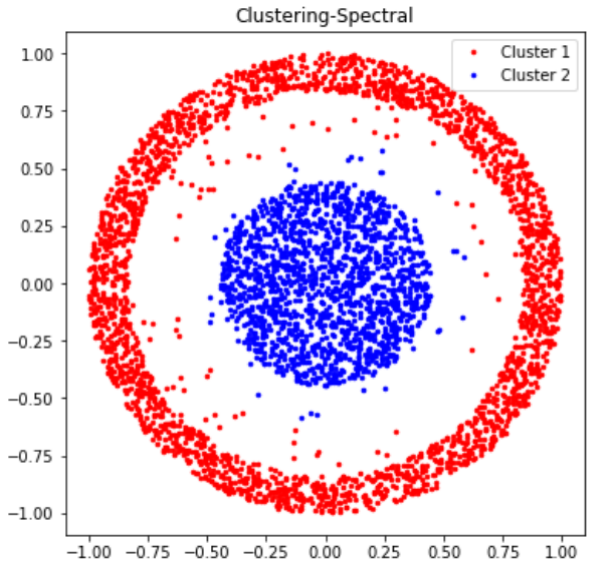
knn参数-k=20, threshold=1.45



knn参数-k=30,threshold=1.45



knn参数-k=10,threshold=1.45



参数threshold在[1,5]区间图像没有明显区别。

两种聚类算法的聚类结果区别

k-means聚类算法只能进行凸函数的聚类，面对像环状的数据集不能很好地区分，谱聚类算法可以对非凸函数进行比较好的聚类。（k-means比较快速，谱聚类耗时较久）

分类算法实现

LR模型

首先，选择划分训练集为0.8*源数据集大小。测试集为剩下的0.2。

先对给定的X进行扩充1使其成为$[1, x, x^2]$的矩阵：

```
P, N = X.shape
X_t = np.zeros((P+1, N))
X_t[0] = 1
X_t[1:] = X_train
```

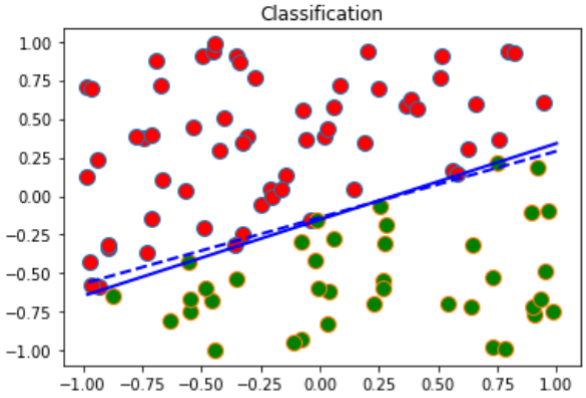
然后使用LR模型公式对权重矩阵进行计算：

```
w = np.dot(np.dot(np.linalg.inv(np.dot(X_t,X_t.transpose())),X_t),y_train.transpose())
```

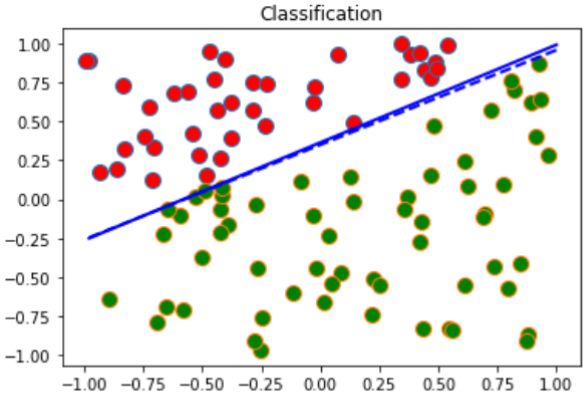
对训练集进行训练，并对测试集进行测试，记录错误数据：

```
train_err = 0
test_err = 0
P, N = X_train.shape
X_t = np.zeros((P+1, N))
X_t[0] = 1
X_t[1:] = X_train
trainy = np.dot(w_l.transpose(),X_t)
P, N = X_test.shape
X_t = np.zeros((P+1, N))
X_t[0] = 1
X_t[1:] = X_test
testy = np.dot(w_l.transpose(),X_t)
for i in range(no_train):
    if trainy[0][i] * y_train[0][i] < 0:
        train_err += 1
for i in range(no_test):
    if testy[0][i] * y_test[0][i] < 0:
        test_err += 1
```

效果如下：




```
Training error: 2
Testing error: 2
```




```
Training error: 1
Testing error: 0
```

感知机模型

首先，选择划分训练集为0.8*源数据集大小。测试集为剩下的0.2。

先对给定的X进行扩充1使其成为$[1, x, x^2]$的矩阵。

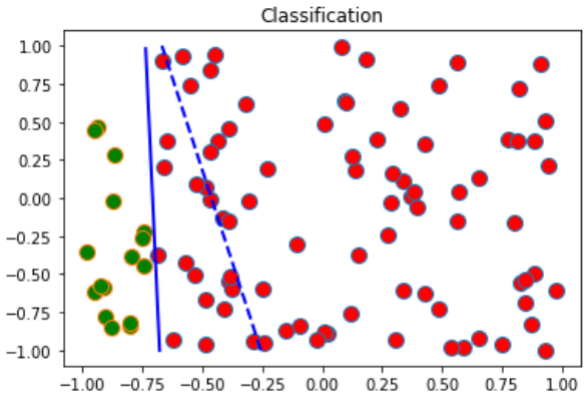
然后对数据集进行训练。这里使用一个神经元，设置学习率为0.75，循环n次，每次对每个输入输出进行预测和调整：

```
for i in range(n):
        for i in range(N):
            if np.dot(X_t[i], w) * y[i] <= 0:
                z = learning_rate * X_t[i] * (y[i]-np.dot(X_t[i], w))
                for j in range(P+1):
                    w[j] += z[j]
```

对训练集进行训练，并对测试集进行测试，记录错误数据。

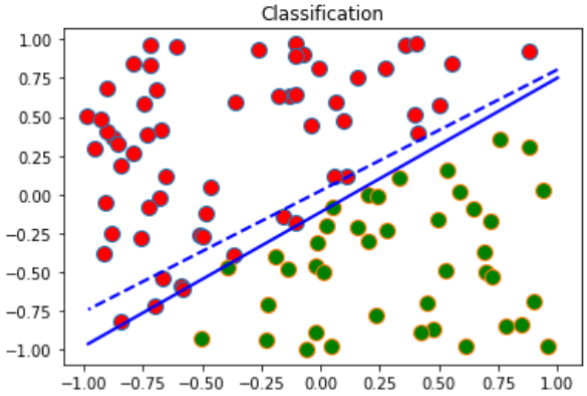
效果如下：

n=1



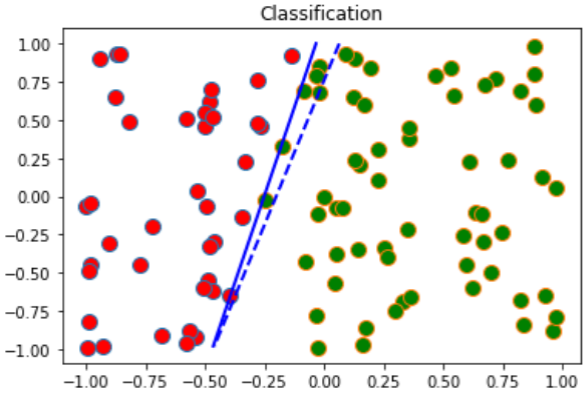

```
Training error: 13
Testing error: 4
```

n=10



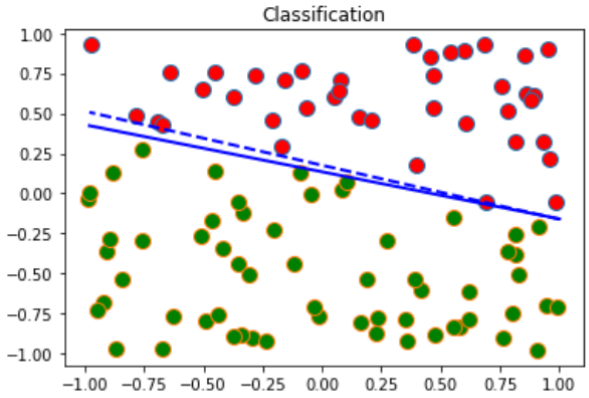

```
Training error: 7
Testing error: 1
```

n=20




```
Training error: 3
Testing error: 2
```

n=30




```
Training error: 0
Testing error: 0
```

异同分析

LR模型可以通过计算公式得到，所以很快，而感知机在迭代次数少时效果较差，所以需要较多的迭代次数，耗时较长。但是LR模型的准确率没有多次迭代的感知机高

> 注：代码中默认使用func函数，即LR模型，如果要使用感知机模型，请import func2, 并将func函数名称改为fun2.