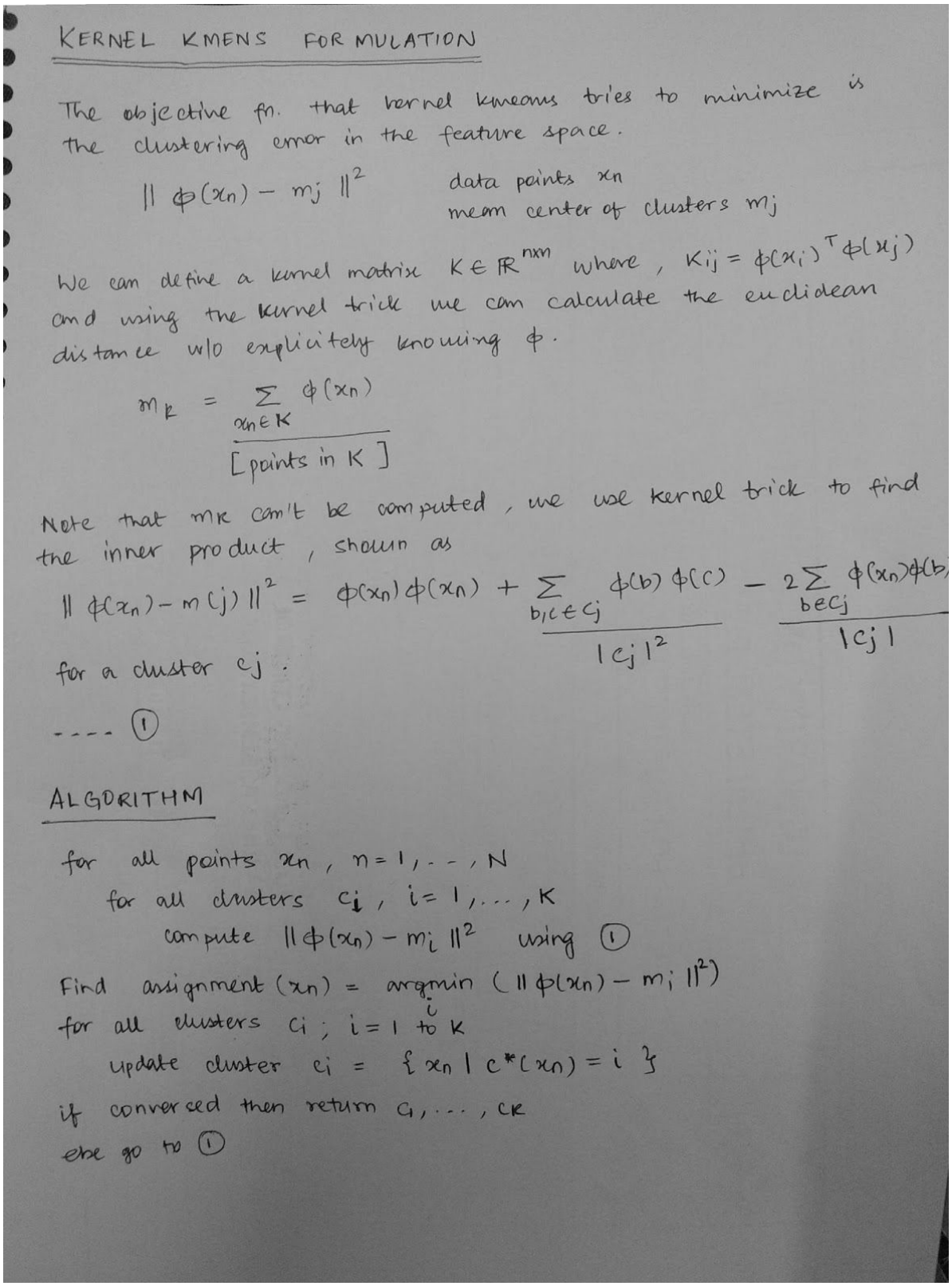
SMAI Assignment 4 Report (201401074)

Problem 1



KERNEL KMENS FORMULATION

The objective fn. that kernel kmeans tries to minimize is the clustering error in the feature space.

$\| \phi(x_n) - m_j \|^2$ data points x_n, mean center of clusters m_j

We can define a kernel matrix $K \in \mathbb{R}^{n \times n}$ where, $K_{ij} = \phi(x_i)^T \phi(x_j)$ and using the kernel trick we can calculate the euclidean distance w/o explicitely knowing ϕ.

$$m_k = \frac{\sum_{x_n \in K} \phi(x_n)}{[\text{points in } K]}$$

Note that m_K can't be computed, we use kernel trick to find the inner product, shown as

$$\| \phi(x_n) - m(j) \|^2 = \phi(x_n)\phi(x_n) + \frac{\sum_{b,c \in c_j} \phi(b)\phi(c)}{|c_j|^2} - \frac{2\sum_{b \in c_j} \phi(x_n)\phi(b)}{|c_j|}$$

for a cluster c_j.

---- (1)

ALGORITHM

for all points x_n, $n = 1, \ldots, N$
  for all clusters c_i, $i = 1, \ldots, K$
    compute $\| \phi(x_n) - m_i \|^2$ using (1)
Find assignment $(x_n) = \underset{i}{\text{argmin}} (\| \phi(x_n) - m_i \|^2)$
for all clusters c_i; $i = 1$ to K
  update cluster $c_i = \{ x_n \mid c^*(x_n) = i \}$
if converged then return $c_1, \ldots, c_K$
else go to (1)

Problem 2

Code : Agglomerative Clustering

```
import numpy as np
from numpy import linalg as la
import matplotlib.pyplot as plt
import sys
from numpy import genfromtxt
my_data = genfromtxt('data1.txt', delimiter=',')

def similarity(a, b):

      return np.nan_to_num(np.exp(-la.norm(a-b)))

def merge(dsu, x, y):

      px = find(dsu, x)
      py = find(dsu, y)
      if px == py:
      return
      dsu[py-1] = px

def find(dsu, x):

      if dsu[x-1] == x:
      return x
      dsu[x-1] = find(dsu, dsu[x-1])
      return dsu[x-1]

def agglomerativeClustering(data, numClusters):

      inf = -1
      N = data.shape[0]
      C = np.zeros((N, N))
      dsu = range(1, N+1)
      for n in range(N):
      for i in range(N):
            C[n][i] = similarity(data[n,:], data[i,:])
      C = C + inf * np.identity(N)
```

```
        for k in range(N - numClusters):
        tmp =  np.where(C == np.max(C))
        i = tmp[0][0]
        m = tmp[1][0]
        C[i, m] = 0
        C[m, i] = 0
        merge(dsu, i+1, m+1)
        for j in range(N):
              C[i, j] = max(C[i, j], C[m, j])
              C[j, i] = C[i, j]
        C[m, :] = C[m, :] * 0
        C[:, m] = C[:, m] * 0
        for i in range(N):
        dsu[i] = find(dsu, i+1)
        return dsu

def main():

        raw = np.loadtxt('data.txt')
        #raw = genfromtxt('data2.txt', delimiter=',')
        data = raw[:,:-1]
        labels = raw[:,-1].astype(int)
        N = data.shape[0]
        plabels = agglomerativeClustering(data, 2)
        plabels = np.array(plabels)
        total = 0
        for plabel in np.unique(plabels):
        idx = np.where(plabels == plabel)[0]
        vec = labels[idx]
        cnt = np.bincount(vec)
        total += float(np.amax(cnt))
        print "Purity",total/N
        plabels1 = np.where(plabels==1)
        plabels2 = np.where(plabels==26)
        for idx in plabels1[0]:
        plt.plot(data[idx, 0], data[idx, 1], 'ro')
        for idx in plabels2[0]:
        plt.plot(data[idx, 0], data[idx, 1], 'bo')
        plt.show()
```

```
if __name__ == '__main__':
      main()
```

Code : Spectral Clustering

```
import numpy as np
from numpy import linalg as la
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform
from sklearn.cluster import KMeans
import sys
from numpy import genfromtxt


def getW(data, numNeighbours):

      pairwise_dists = squareform(pdist(data, 'sqeuclidean'))
      return np.nan_to_num(np.exp(-pairwise_dists))

def findEigenGap(vec):

      diff = np.ediff1d(vec)
      return np.argmin(diff)

def spectralClustering(data, numNeighbours, numClusters):

      N = data.shape[0]
      W = getW(data, numNeighbours)
      print W
      D = W.sum(axis = 0) * np.identity(N)
      print W[:,0].sum(axis = 0)
      L = D - W
      eigenValues,eigenVectors = la.eig(L)
      idx = eigenValues.argsort()
      kcrit = findEigenGap(eigenValues)
      k = 5
      idx = idx[:k] # ????
      eigenValues = eigenValues[idx]
      eigenVectors = eigenVectors[:, idx].reshape(-1,k)
```

```
        print eigenVectors
        kmeans = KMeans(n_clusters=2, random_state=0).fit(eigenVectors)
        label = kmeans.labels_
        return label

def main():

        raw = np.loadtxt('data.txt')
        #raw = genfromtxt('data2.txt', delimiter=',')
        data = raw[:,:-1]
        labels = raw[:,-1].astype(int)
        plabels = spectralClustering(data, 10, 2)
        plabels = np.array(plabels)
        print plabels
        N = data.shape[0]
        total = 0
        for plabel in np.unique(plabels):
        idx = np.where(plabels == plabel)[0]
        vec = labels[idx]
        cnt = np.bincount(vec)
        total += float(np.amax(cnt))
        print "Purity",total/N
        plabels1 = np.where(plabels==1)
        plabels2 = np.where(plabels==0)
        for idx in plabels1[0]:
        plt.plot(data[idx, 0], data[idx, 1], 'ro')
        for idx in plabels2[0]:
        plt.plot(data[idx, 0], data[idx, 1], 'bo')
        plt.show()

if __name__ == '__main__':
        main()
```

Code : Kernel Kmeans

```
import numpy as np
from numpy import linalg as la
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist, squareform
import sys
import random
from numpy import genfromtxt
my_data = genfromtxt('data1.txt', delimiter=',')

def linearKernel(data):
      kernel = data.T.dot(data)
      return kernel

def polynomialKernel(data, p):
      mat = data.T.dot(data)
      mat = np.add(1, mat)
      kernel = np.power(mat, p)
      return kernel

def gaussianKernel(data, sigma):
      pairwise_dists = squareform(pdist(np.transpose(data),
'sqeuclidean'))
      mat = np.divide(pairwise_dists, -2*sigma*sigma)
      kernel = np.exp(mat)
      return kernel


def kernelClustering(data, numClusters):
      N = data.shape[0]
      K = linearKernel(data.T)
      #K = polynomialKernel(data.T, 5)
      #K = gaussianKernel(data.T, 0.1)
      A = np.zeros((N, numClusters))
      f = np.zeros((1, N))[0]
      for i in range(N):
      f[i] = random.randint(0, numClusters-1)
      f.astype(int)
      for i in range(N):
      A[i, f[i]] = 1
```

```
      print A.shape
      change = 1
      while change == 1:
      change = 0
      E = A.dot(np.diag(np.divide(1.0, np.sum(A, axis=0))))
      Z =
np.ones((N,1)).dot(np.diagonal(E.T.dot(K.dot(E))).reshape(1,-1)) -
2*K.dot(E)
      ff = np.argmin(Z, axis=1)
      for i in range(numClusters):
            if f[i] != ff[i]:
                  A[i, ff[i]] = 1
                  A[i, f[i]] = 0
                  change = 1
      f = ff
      return f

def main():
      for haah in range(10):
      raw = np.loadtxt('data.txt')
      #raw = genfromtxt('data2.txt', delimiter=',')
      data = raw[:,:-1]
      labels = raw[:,-1].astype(int)
      N = data.shape[0]
      plabels = kernelClustering(data, 2)
      plabels = np.array(plabels)
      total = 0
      for plabel in np.unique(plabels):
      idx = np.where(plabels == plabel)[0]
      vec = labels[idx]
      cnt = np.bincount(vec)
      total += float(np.amax(cnt))
      print "Purity",total/N
      plabels1 = np.where(plabels==1)
      plabels2 = np.where(plabels==0)
      for idx in plabels1[0]:
      plt.plot(data[idx, 0], data[idx, 1], 'ro')
      for idx in plabels2[0]:
      plt.plot(data[idx, 0], data[idx, 1], 'bo')
```

```
        plt.show()

if __name__ == '__main__':
        main()
```

Results : Average purity of clusters

Breast Cancer Dataset

- **Agglomerative Clustering - Inverse Exponential Similarity : 0.658**
- **Agglomerative Clustering - Inverse Euclidean Similarity : 0.658**
- **Spectral Clustering - First 2 Eigenvectors : 0.665**
- **Spectral Clustering - First 3 Eigenvectors : 0.662**
- **Spectral Clustering - First 4 Eigenvectors : 0.662**
- **Spectral Clustering - First 5 Eigenvectors : 0.660**
- **Kernel K-Means : 0.655 (same for linear kernel, all polynomial kernels, all gaussian kernels)**

2 Moons Dataset

- **Agglomerative Clustering - Inverse Exponential Similarity : 0.809**
- **Agglomerative Clustering - Inverse Euclidean Similarity : 0.767**
- **Spectral Clustering - First 1 Eigenvector : 0.809**
- **Spectral Clustering - First 2 Eigenvectors : 1.000**
- **Spectral Clustering - First 3 Eigenvectors : 0.930**
- **Spectral Clustering - First 4 Eigenvectors : 0.966**
- **Spectral Clustering - First 5 Eigenvectors : 0.966**
- **Kernel K-Means - Linear Kernel : 0.771**
- **Kernel K-Means - Polynomial (1) : 0.778**
- **Kernel K-Means - Polynomial (3) : 0.743**
- **Kernel K-Means - Polynomial (5) : 0.739**
- **Kernel K-Means - Gaussian (0.001/0.01/0.1) : 0.739**
- **Kernel K-Means - Gaussian (1) : 0.739**
- **Kernel K-Means - Gaussian (10) : 0.762**
- **Kernel K-Means - Gaussian (100) : 0.778**
- **Kernel K-Means - Gaussian (1000) : 0.770**