Statistical and Machine Learning (01.113)
Homework 4

Due on 16 APR, 3 PM

Problem 1 (2 points)

Let $p = (p_1, p_2)$ and $q = (q_1, q_2)$ be distributions over the set $\{1, 2\}$ and assume that p_1, p_2, q_1 and q_2 are all not zero. Find values for p and q such that $D_{KL}(p \,|\, q) \neq D_{KL}(q \,|\, p)$. Use base 2 for the logarithm and show all working.

Problem 2 (2 points)

Let $p(x, z)$ be a joint distribution on $\mathbb{R}^2$, with $p(x)$ and $p(z|x)$ denoting the corresponding marginal and conditional distributions respectively. Let q(z) be any distribution on $\mathbb{R}$, and prove that

$$\log p(x) = \int_{\mathbb{R}} q(z) \log \frac{p(x, z)}{q(z)} \, \mathrm{d}z + D_{KL} \left[q(z) \,|\, p(z \,|\, x) \right].$$

Problem 3 (2 points)

In this problem, we will perform principal component analysis (PCA) on sklearn's diabetes dataset. Start by importing the required packages and load the dataset.

```
import numpy as np
from sklearn import decomposition
from sklearn import datasets

X = datasets.load_diabetes().data
```

You can find out more on how to use sklearn's PCA module from:
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

(a) Write code to print the matrix V that will be used to transform the dataset, and print all the singular values.

(b) Now perform PCA on the dataset and print out the 3 most important components for the first 10 data-points.

As this problem is short, print out your script and results, and include it with your hardcopy submission. Do not upload the answer into your Dropbox folder.

Problem 4 (4 points)

In this problem, we will implement the EM algorithm for clustering. Start by importing the required packages and preparing the dataset.

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

from numpy import linalg as LA
from matplotlib.patches import Ellipse
from sklearn.datasets.samples_generator import make_blobs
from scipy.stats import multivariate_normal

K = 3
NUM_DATAPTS = 150

X, y = make_blobs(n_samples=NUM_DATAPTS, centers=K, shuffle=False,
                                        random_state=0, cluster_std=0.6)
g1 = np.asarray([[2.0, 0], [-0.9, 1]])
g2 = np.asarray([[1.4, 0], [0.5, 0.7]])
mean1 = np.mean(X[:int(NUM_DATAPTS/K)])
mean2 = np.mean(X[int(NUM_DATAPTS/K):2*int(NUM_DATAPTS/K)])
X[:int(NUM_DATAPTS/K)] = np.einsum('nj,ij->ni',
                X[:int(NUM_DATAPTS/K)] - mean1, g1) + mean1
X[int(NUM_DATAPTS/K):2*int(NUM_DATAPTS/K)] = np.einsum('nj,ij->ni',
                X[int(NUM_DATAPTS/K):2*int(NUM_DATAPTS/K)] - mean2, g2) + mean2
X[:,1] -= 4
```

(a) Randomly initialize a numpy array *mu* of shape (K, 2) to represent the mean of the clusters, and initialize an array *cov* of shape (K, 2, 2) such that *cov*[*k*] is the identity matrix for each k. *cov* will be used to represent the covariance matrices of the clusters. Finally, set π to be the uniform distribution at the start of the program.

(b) Write a function to perform the E-step:

```
def E_step():
        gamma = np.zeros((NUM_DATAPTS, K))
        ...
        ...
        return gamma
```

(c) Write a function to perform the M-step:

```
def M_step(gamma):
        ...
        ...
```

(d) Now write a loop that iterates through the E and M steps, and terminates after the change in log-likelihood is below some threshold. At each iteration, print out the log-likelihood, and use the following function to plot the progress of the algorithm:

```
def plot_result(gamma=None):
        ax = plt.subplot(111, aspect='equal')
        ax.set_xlim([-5, 5])
        ax.set_ylim([-5, 5])
        ax.scatter(X[:, 0], X[:, 1], c=gamma, s=50, cmap=None)

        for k in range(K):
                l, v = LA.eig(cov[k])
                theta = np.arctan(v[1,0]/v[0,0])

                e = Ellipse((mu[k, 0], mu[k, 1]), 6*l[0], 6*l[1],
                        theta * 180 / np.pi)
                e.set_alpha(0.5)
                ax.add_artist(e)

        plt.show()
```

Show ALL results, upload the final script in your Dropbox folder and name it as "**HW4_4.py**'.