APM120, Homework #8

Applied Linear Algebra and Big Data

Last updated: Thursday 23rd March, 2023, 20:58

Assigned March 28, due Friday, April 7, 11:59 pm, via gradescope, **in pdf $\leq$ 20Mb, $\leq$ 30 pages.**

Unsupervised learning: cluster analysis, part II

Show all steps in all calculations explicitly. Attach code used, well documented, and relevant plots and Matlab/python output, attaching code and figures *immediately following* the relevant question solution. A code printout is not a substitute for complete solutions, your solution should stand alone without the Matlab/python code or output. See needed python preliminaries at end of this HW[1]. It is fine to use Matlab/python unless a hand calculation (using only a simple calculator) is required *in orange*. For all questions, make sure you show all steps as if you are doing the problem by hand, and do not use library functions unless explicitly allowed in the question. Make sure you can do all calculations using no more than a hand-held calculator. See the end-note for guidelines and examples of hand-calculations.[2]

1. **Self Organizing maps:**

   (a) Consider the following data set of two dimensional points,

   ```
   X=[ 4,  9,  8,  7,  1,  1,  1,  2,  1,  5,  4,  10,  9,  8,  2,  4;
       5,  8,  8,  2,  9,  10,  1,  1,  2,  5,  6,  8,  7,  2,  9,  4];
   ```

   Use an $n \times 1$ SOM with an appropriate n to cluster the data: to get the initial representative vectors, start with the data vectors $\#6, 7, \ldots, 6+n-1$, and add to each the vector $[1, -1]$. The corresponding grid space ordering is $1, 2, \ldots, n$ in a 1d configuration. Let the neighborhood kernel be 0.8 for adjusting the nearest representative point and 0.1 for adjusting its two neighboring points in grid space. Make the learning rate constant, $\eta = 0.1$. Perform one iteration over all data points (one epoch) to find the location of the representative points at the end of the epoch. Iterate through at least 2 data points, showing and explaining all steps, after which you may write a Matlab/python script to complete the computation. Use a scatter plot

   ```
   % Matlab
   scatter(X(1,:),X(2,:),'r')
   ```

   ```
   # python:
   plt.scatter(X[0,:],X[1,:]); plt.show()
   ```

   to display the original data and superimpose the final representative vectors, using different colors for the data and the final solution. See `self_organizing_maps_hand_calculation_example.(m/py)` for guidance.

   (b) ***Optional extra credit challenge problems**: Create a data set using:

   ```
   % Matlab
   rng(2021); N=250; X=zeros(2,4*N);
   for i=1:N;       X(:,i)=[5,5]' +(1.5*(randn(2,1)-1)); end;
   for i=N+1:2*N;   X(:,i)=[0,-5]'+(2.0*(randn(2,1)-1));end;
   for i=2*N+1:3*N; X(:,i)=[5,20]'+(3.0*(randn(2,1)-1));end;
   for i=3*N+1:4*N; X(:,i)=[20,5]'+(2.0*(randn(2,1)-1));end;
   p=randperm(4*N); X=X(:,p);figure(1); clf; scatter(X(1,:),X(2,:),'r')
   ```

```
# python:
np.random.seed(2021); N=250
X=np.vstack([np.array([[5,5]]) +1.0*np.random.normal(size=(N,2)),
             np.array([[0,-5]])+2.0*np.random.normal(size=(N,2)),
             np.array([[5,20]])+3.0*np.random.normal(size=(N,2)),
             np.array([[20,5]])+2.0*np.random.normal(size=(N,2))]);
X=X.T; RN=np.random.permutation(4*N); X=X[:,RN];
plt.scatter(X[0,:],X[1,:],s=0.1,marker='o',c='b');
```

modify the SOM script self_organizing_maps_library_function_example.(m/py) on the course web page which uses the library SOM functions to cluster and plot these data using a 3×2 SOM.

2. **Spectral clustering:** divide the following 2d data set into clusters using spectral clustering:

```
X=[0.5,1.5,0.5,1,9,7.5,7,9.5,1.5,1,1,8.5,9.5,6.5,8,0.5,1,9,9.5,7,10,1.5,8.5,9.5;
   9.5,8.5,0.5,1,0.5,5.5,4.5,9.5,9.5,0.5,1.5,0.5,1,5.5,4.5,9,9,9.5,9,5,9.5,0.5,1,8.5];
```

Define the similarity matrix $\mathsf{W} = \{w_{ij}\}$ based on the L_2 distance d_{ij}, such that $w_{ij} = \exp(-d_{ij}^2/d^2)$ where d is the std of the elements d_{ij}. Write explicitly W, D and L. Calculate and plot all eigenvalues of the Laplacian matrix and — on different axes — of the normalized Laplacian matrix. Based on the eigenvalues, how many clusters do you think should be calculated? Calculate the appropriate number of eigenvectors of the normalized Laplacian matrix. Plot these eigenvectors, and explain why we may expect them to be helpful in clustering the data. Perform the spectral clustering by applying k-means to the appropriate eigenvectors. You may want to start from the example spectral_clustering_example.(m/py) on the course web page

3. **CURE: (a)** plot the data in `HW08_CURE.mat`, and cluster it using hierarchical clustering using first the 'single' and then the 'centroid' linkages. Plot the results using a different color for each cluster. Explain your results. **(b)** ***Optional extra credit challenge problems**: Use the CURE algorithm to cluster the data in a way that would work for a much larger problem for which Hierarchical clustering cannot be used. That is, use hierarchical clustering on a small subset of the data to obtain appropriate representatives, and then CURE for the full data set. Plot the smaller subset of data using red $\times$ marks, and the full data using dots. Use a different color for each cluster.

* [What's the point of ***optional extra credit challenge problems: apart from the fun of doing them, they may bring the total score of this HW assignment up to 110%, making up for problems you may have missed in this or other HW assignments...]

Python preliminaries & notes

1 python commands within the HW assume you have first used the followings: `import numpy as np; from numpy import linalg; import scipy as scipy; from scipy import linalg; import matplotlib.pyplot as plt; import matplotlib;`
Input a matrix A, column vector **b** and row vector **c** into python in the form
`A=np.array([[a11,a12,a13],[a21,a22,a23],[a31,a32,a33]]; b=np.array([[b1],[b2],[b3]]); c=np.array([[c1,c2,c3]]);` or convert Matlab arrays given in HW directly to python arrays using, e.g., `A=np.array(np.matrixlib.defmatrix.matrix('[1 2 3; 4 5 6]'));`

2 **Hand calculations**, in which you are asked to use only a simple hand-held calculator, are required only when we want to make sure you understand exactly what each step of an algorithm is doing. These also prepare you for the quizzes that involve similar hand calculations. **How much work to show?** Just don't use scratch paper, show all steps that you actually use for the hand calculations, but no more, carrying out calculations to **three significant digits**. Trust the graders to be reasonable, they were students in the course last year. **Examples: (i)** If you are asked, *not* in orange, to calculate the LU decomposition of a matrix, you may use Matlab/python as in `A(2,:)=A(2,:)-A(1,:)*(A(2,1)/A(1,1))` etc, but you may not use a library routine as in `[L,U,P]=lu(A)` except for checking your results. **(ii)** If required to calculate by hand the element $C_{2,3}$ of a matrix product $\mathsf{C} = \mathsf{AB}$, you need to explicitly multiply using a hand calculator the second row of A with the third column of B. You may not use Matlab/python to calculate `C=A*B` and then take the needed element from that product, nor to calculate `C23=A(2,:)*B(:,3)`.