CSC C11 – Fall 2017
Assignment 2 – How to tell where things belong

Assignment due date: Friday, Nov. 3 - 10am

Hand-in to be submitted at the start of lecture,
code to be submitted on the mathlab server by the above due date

This assignment can be completed individually, or in a team of 2 students

Student Names (last, first)

Student #1:
Student #2:

Student numbers:

Student #1:
Student #2:

Student UtorIDs:

Student #1:
Student #2:

We hereby affirm that all the solutions we provide, both in writing and in code, for this assignment are our own. We have properly cited and noted any reference material we used to arrive at our solution, and we understand both the written solutions and code submitted with our assignment.

_____ _____

Student #1 signature ***Student #2 signature***

(note: 3 marks penalty if any of the above information is missing)

CSC C11 – Fall 2017
Assignment 2 – How to tell where things belong

In this assignment you will explore the problem of clustering data into sets of similar items. As we have discussed in lecture, clustering is an essential tool when we are faced with data that has not been labeled or categorized by human observers, but at the same time, want to apply any of the data modeling techniques we have discussed thus far to items in our dataset. You will implement two general clustering methods, and explore issues related to the type and complexity of your data vectors, distance measures, data pre-processing, and how all of these steps affect the results of the clustering process.

Learning Objectives – after completing this assignment you should be able to:

Set up a clustering process that takes as input a set of data vectors of any length, which are assumed to represent samples of data from a few distinct categories, and produces a set of clusters that group input data by similarity so as to attempt to discover what categories are present in the dataset.

Explain and use Gaussian Mixture Models to perform clustering.

Implement the Expectation-Maximization algorithm for training GMMs, after which you will be able to use E-M to solve similar optimization problems.

Think of data in terms of what makes data vectors similar or different, and use careful Pre-processing to facilitate comparison of data items and enhance clustering results.

Implement document clustering on a real-world dataset. Though simple, this clustering method will already be capable of usefully clustering unknown documents from their word-frequency vector.

Understand and explain issues related to the dimensionality of the input data, and how it affects clustering ability.

Skills Developed:

Implementing and using k-means and GMMs for clustering multi-dimensional data.

Implementing data pre-preprocessing to transform input data vectors into a representation more amenable to clustering.

Handling high-dimensionality, sparse data, typical of real-world applications.

Reference material:

The lecture notes on regression.

Comments included in the starter code.

Acknowledgements:

Document training data provided by: http://mlg.ucd.ie/datasets/bbc.html

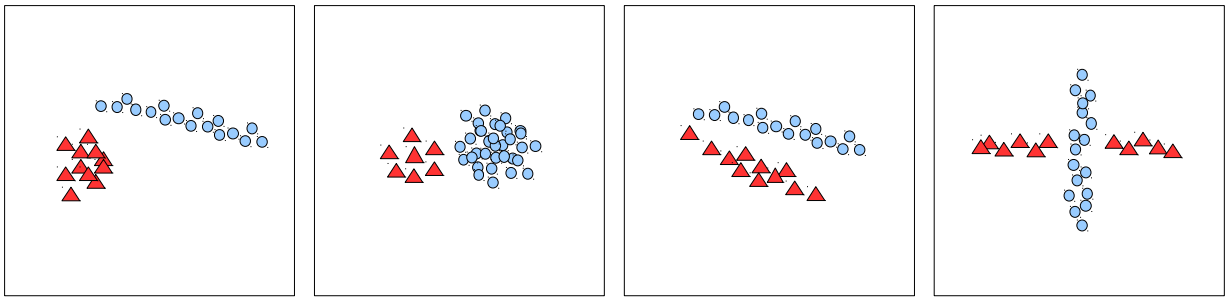
CSC C11 – Fall 2017
Assignment 2 – How to tell where things belong

Part 1 – Written work. Be sure to provide clean, legible derivations and do not omit any steps your TA may need to understand your work. Use diagrams wherever appropriate.

Clustering points – figuring out where data belongs

1) *[2 marks each]* In the datasets below, k-means may incorrectly label points that would be correctly labeled by a GMM, or vice-versa. Look at each dataset with care, consider what you know about how k-means and GMM cluster data, and answer ***for each case***:

   a) Which method will give the better clustering results (if any)
   b) Why this method produces better results (equivalently, why the opposite method mis-labels data)
   c) Circle data points that would be mislabeled by the 'losing' method.
   (assume both methods got a lucky break and their initial cluster centers were picked to be the closest data point to the true cluster's center)



2) *[1 marks each]* For each of the datasets above, indicate whether both methods should be able to find the true clusters ***given a lucky guess for the method's parameters***.
If one or the other of the clustering methods is unable to find the true clusters even with a lucky initial guess, indicate which method(s) fail(s) and why.

3) *[3 marks]* As we know, each component of a Gaussian Mixture Model has the form

$$\frac{w}{\sqrt{2\pi^D|\Sigma|}}\ e^{-\frac{(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})}{2}}$$

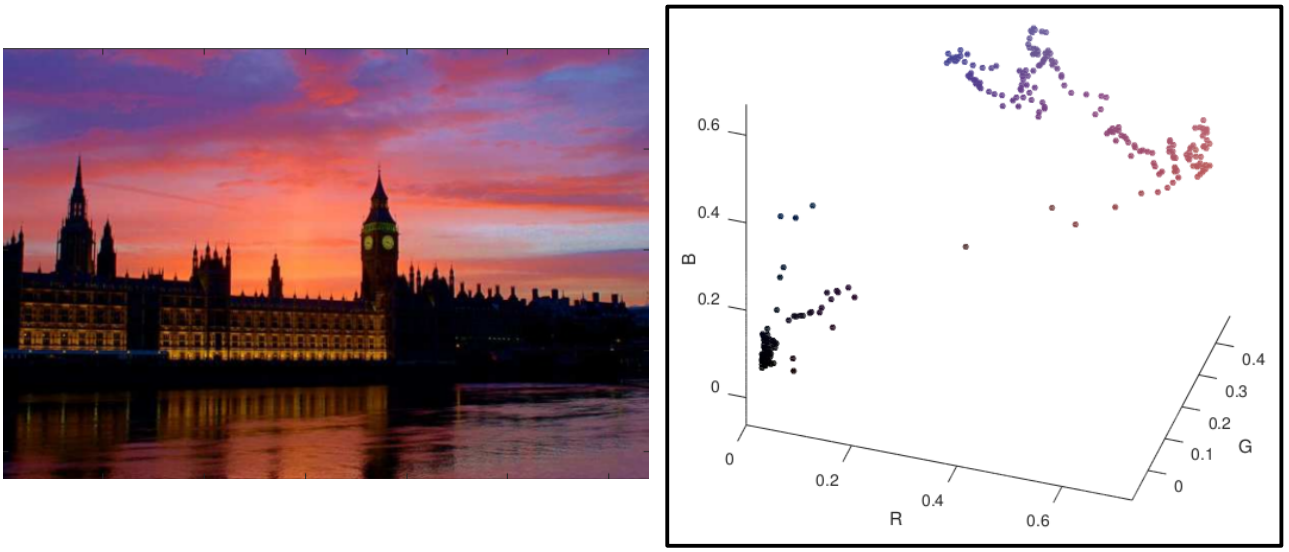
ignoring for the moment the constant, have a look at the exponent. There is a distance computation there, and it's worth understanding what it does.
Take the exponent in the above equation and expand it for a 2D Gaussian, assume the covariance matrix is diagonal.

Show what the distance computation expands to as a single equation in terms of the components of the two vectors and covariance matrix.

4) *[2 marks]* What does this distance measure reduce to when the covariance matrix is identity?

CSC C11 – Fall 2017
Assignment 2 – How to tell where things belong

Part 2 – Clustering data – k-Nearest-Neighbours and GMMs



Input image, and input pixels as points in RGB colourspace

Clustering is an extremely useful operation in a variety of domains of application for machine learning. In this part of the assignment, you will implement two of the most extensively used algorithms for clustering: ***k-means*** and ***Gaussian Mixture Models***. You will apply these methods to the problem of clustering pixels in colour images so that pixels in regions of similar colour end up in the same cluster.

The general task you are attempting to solve is called ***image segmentation***, and is a fundamental problem in computer vision, since it deals with determining regions of an image that belong to a single object – so that these regions can then be processed by more advanced vision modules dealing with object recognition or categorization, motion tracking, and so on.

Having implemented these two clustering methods, you will then spend some time exploring the difficulties of clustering data in high-dimensional spaces, and learning about how careful understanding of your data, and suitable pre-processing are both necessary to achieve a good clustering result.

Unlike assignment 1, this assignment asks a number of questions. You are expected to write short answers (no more than 2 or 3 sentences) for each of them in the file called 'questions.txt' which will be submitted with your solution.

Part 2 – Clustering data – k-Nearest-Neighbours and GMMs

Your first task is to implement the scripts that perform k-means and Gaussian Mixture Model clustering. The k-means implementation is straightforward. The GMM fitting is a bit more involved, but it basically involves carrying out the two steps of the E-M algorithm as described in lecture until the parameters for all model components are found.

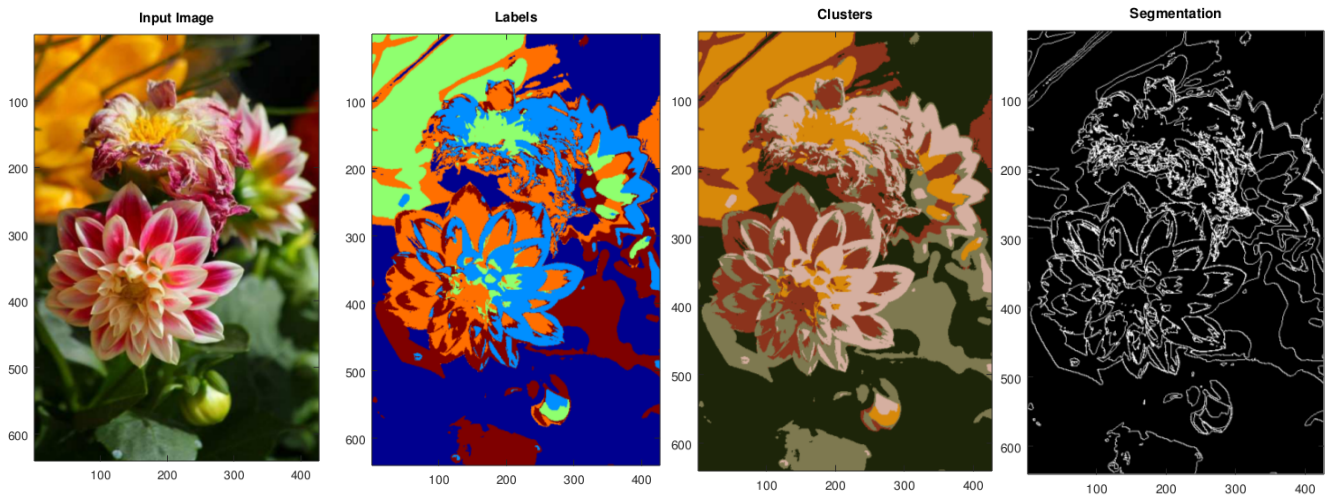
The amount of code you have to write is tiny, but it requires careful understanding of the algorithm for both clustering methods.

Step 1 – Download and uncompress the starter code into a suitable location in your home directory.
This is Matlab code, you can use Matlab at the computer lab, or you can use Octave (the free, open-source alternative) either at the lab or on your own machine. However, note the code is tested under Linux, your solution must work on Linux for grading.

The first thing you have to do is go over the scripts included with the starter code. Pay particular attention to the examples provided that show how each of these functions would be called, and what the expected output would be.

****note that the script for document clustering contains a save() statement to cache slow-to-compute data, you need to comment/uncomment the appropriate line depending on whether you are using Octave or Matlab, this is an unfortunate effect of Octave not saving Matlab-compatible files by default.***

Step 2 – Implement ***k-means*** and test it using the `RGB_clustering.m` script. Two test images are included. Try both, using different numbers of clusters, and make sure your algorithm produces a reasonable result for various clustering settings (also make sure it behaves properly given that the initialization of cluster centers is random).



Input image and k-means results: ***labels, clusters*** (colour is equal to final cluster center colour), and ***region boundaries***

Part 2 – Clustering data – k-Nearest-Neighbours and GMMs

Step 2 (cont.)

Now implement the Gaussian Mixture Models. Test it again, with various settings for number of clusters and on both test images. Make sure your code gives reasonable results.

Consider the following questions

- Which algorithm gives more 'natural' results? (i.e. which produces clusters that more closely approximate what you believe are distinct image regions?)

- Which algorithm gives better clustering results in terms of data similarity? (that means, purely in terms of RGB similarity, not your subjective perception of what makes an image region)

- Which method would you expect to work best in general for RGB clustering, given plots such as the one at the start of this section (the image of London and the corresponding points in RGB space)

- What kinds of pre-processing can you think of that would have helped the clustering process (and why would they have helped)?

Once you're satisfied your implementation works, run the inpainting driver script to clean up the provided test image. Choose appropriate parameters to obtain a clean image.

Step 3 – Let's now turn to a more crunchy problem. Read carefully the `document_clustering.m` script. It loads a dataset from the BBC which contains ***word frequency vectors*** for a couple thousand documents on ***five different topics***.

Our goal is to cluster these documents so as to discover ***what those topics may be***. this general problem is called ***topic modeling*** and it is an important and active area of research in machine learning. Here we will only explore some of the issues involved In figuring out what the possible topics in this dataset may be via clustering and careful data pre-processing.

Before you start (and in general, before you start applying machine learning to a new dataset)

- Have a look at the data. Load the BBCdata and plot a few vectors, inspect some of the words in the terms array, print the word contents (and frequencies) for a few of the document vectors in the dataset, and get a good idea of how many different words you can expect on average in a document. You should also look into how sparse these vectors are (i.e. on average, how many of the entries in each vector are zero).

- Do a little bit of footwork. Find what are the 10 most common terms, Find what are the 10 least common terms. What is the average value for word-frequencies? (only counting vector entries that are non-zero).

Part 2 – Clustering data – k-Nearest-Neighbours and GMMs

Step 3.1

Run the document clustering script with `K=5, norm_flag=0, diffuse=0`. This will call your k-means clustering code to get to work on the original input vectors.

Inspect the resulting clusters and labels and record:

- What is the size of each cluster?
- Does the clustering make any sense? For this, have a look at the text files produced by the clustering algorithm, they will contain the words that are distinct for each of the detected clusters. These files are called 'centers_[1-5].txt' and will be in the same directory where you have the code.
- What would you say is the topic for each cluster?

If you want a quick visualization, run the script 'mkWordClouds.sh' which will plot a word cloud for each cluster ***(Linux ONLY – you have to install WordCloud if you are running this on your own machine)***.

- Run the clustering script several times. Results should be similar.

At this point it is fair to wonder if we can actually cluster these documents.

- What are the factors that make clustering difficult?
- Should we expect much better results if we get a lucky guess at cluster centers?

Step 3.2

Run the document clustering step with `K=5, norm_flag=1, diffuse=0`. This will run clustering after pre-processing the input document vectors so they have ***unit mass***. In effect, we are now treating these vectors as probability distributions over terms.

- What problem we had in 3.1 does this solve?
- What is the size of each cluster?
- Does the clustering make sense?
- What would you say are the topics for clusters?
- Would you consider this result better, or worse than 3.1? why?

Step 3.3

Run the clustering with `K=5, norm_flag=1, diffuse=2`. This will pre-process each document by doing 2-steps of random-walk diffusion based on word co-occurrence probabilities (estimated from the entire data-set). That is, the Pre-processed vectors will not only have the original terms, but also contain words that are strongly associated with those originally in the document.

Part 2 – Clustering data – k-Nearest-Neighbours and GMMs

Step 3.3 (cont.)

Run the clustering a few times, checking the output results, and pick a good one.

Answer these questions:

- What is the size of each cluster?
- Does the clustering make sense?
- What would you say are the topics for clusters?
- Why is the clustering suddenly better?
- What would you say is the general lesson to be learned from trying to cluster high-dimensional sparse data?

Step 3.4

Plot the word-clouds for each of the output clusters from 3.3 and include them with your submitted code.

One of the output clusters for step 3.3 according to our solution for this assignment – your clusters ***will not be identical to this result*** (don't forget that random initialization step!)

(5 mark penalty for not following these instructions carefully):
Step 4 – Package your solution and submit it.

Fill-in '***autotester_id.txt***'. Compress your code, the text files 'centers_[1-5].txt', and the Word-clouds you captured in 3.4 into a single ***.tgz*** file named

Clusters_studentNo1_studentNo2.tgz **(e.g. Clusters_11223344_55667788.tgz)**

The *tar* command syntax is:

```
>tar  -cvzf  name_of_your_compressed_tar_file.tgz  *.m  *.txt *.png
```

Then submit the compressed file ***(don't copy/paste, type the command)***

```
>submit -c cscc11f17 -a A2 -f name_of_your_compressed_tar_file.tgz
```

Double check that your compressed file uncompresses properly, and that it contains all the filed noted above.

Marking Scheme	
Written problems	17 marks
Working code + answers	33 marks (auto-tested)
In-class quiz	50 marks (individual)

Optional Crunchy Bonus (up to 25 marks)

For bonus marks – come up with a different, crunchy application of clustering and write a working demo script.

It doesn't have to be fancy – but it should demonstrate that it addresses a real problem that is suitably handled by regression.

If you have an idea but would like my opinion on it, drop by and talk with me! If you are looking for datasets, try the CORGIS machine learning repository:

https://think.cs.vt.edu/corgis/

Pick a dataset or problem you care about, have some fun applying what you learn in this course, and earn bonus marks! ***(you can work on this with your team partner).***