CS 383 - Machine Learning

Assignment 2 - Clustering

Introduction

In this assignment you'll work on clustering data.

You may not use any functions from machine learning library in your code, however you may use statistical functions. For example, if available you **MAY NOT** use functions like

- kmeans

However you **MAY** use basic statistical functions like:

- std
- mean
- cov
- svd
- In addition, since the spirit of this assignment is not (necessarily) about feature reduction, you may use functions like **pca**.

Grading

Although all assignments will be weighed equally in computing your homework grade, below is the grading rubric we will use for this assignment:

Part 1 (Theory)	20pts
Part 2 (K-Means Clustering)	70pts
Report	10pts
TOTAL	100pts

Table 1: Grading Rubric

DataSets

Pima Indians Diabetes Data Set In this dataset of 768 instances of testing Pima Indians for diabetes each row has the following information (1 class label, 8 features).

0. Class Label (-1=negative,+1=positive)
1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thinkness (mm)
5. Insulin (mu U/ml)
6. Body mass index (kg/m^2)
7. Diabetes pedigree function
8. Age (yrs)

Data obtained from: https://archive.ics.uci.edu/ml/support/diabetes

1 Theory

1. (10pts) Given an average intracluster distance for clustering level j, W_j, what is the **third** derivative at W_j, W_j'''?.

2. (10pts) Given the output of your clustering algorithm as $C_1 = \{1, 2, 3, 4\}, C_2 = \{5, 6, 7, 8\}$, and a hand labeled clustering of $C_1 = \{3, 4\}, C_2 = \{1, 2, 5, 6, 7, 8\}$, what is the weighed average purity of the clusters created by the clustering algorithm?

2 Clustering

Let's implement our own version of k-means to cluster data!

Write a script that:

1. Reads in the data. For demonstrative purposes, you will be using the dataset mentioned in Datasets section (i.e. the diabetes dataset). However, given another dataset's observable data matrix, X, your code should still work.
2. Separates the class label from the observable data to form matrices Y and X, respectively.
3. Standardizes the features.
4. Passes the observable data X and the class labels Y to a (user-created) function called *myK-Means* that takes three parameters:
   (a) The data to cluster, X
   (b) The target clustering, Y (we'll use this for measuring purity as wel go)
   (c) The number of clusters, k

The myKMeans function :
Your *myKMeans* function should run k-means on the supplied data and value of k. Since we'll visualize the clustering process, a few caveats:

- Although theoretically your code should work for any value of $1 <= k <= N$, we'll constrain k to have a maximum value of 7, so that you will only need to define up to 7 colors.
- Although your code should be written in such a way to be able to cluster in any dimensional space, for visualization purposes, you may assume dimesonality $D > 1$, and if your data's dimenionality is greater than 3, first reduce its dimensionality to 3 using PCA. You may use the code you developed in HW1 to do this, or a PCA function from your linear algebra library.

In order to visualize the cluster process, you'll generate a video. Choose a frame rate that is natural for observing the changes in the association. Each frame of the video will be a plot showing the current clustering. If X has only 2 features, this will be a 2D plot. If it has 3 features (either natively, or after PCA reduction), then this will be a 3D plot. If possible (that is, if the graphics library you are using allows it), try to have any 3D plots rotated so that it is easier to see what's going on.

For each frame/plot you should:

1. Display the current reference vectors as solid circles and the observations as x's.
2. The reference vectors and associated observations should have the same colors. For instance, if you use blue to represent cluster 1, then its reference vector should be a solid blue circle, and all observations associated with it should be blue x's.
3. At the top of your graph you should state the iteration number as well as the current weighed average purity of the clusters, using the vector Y as the hand-labeled cluster assignments.

Figures 1-2 show the initial and terminal clusterings when $k = 2$.

Implementation Details

1. Seed your random number generator with zero (do this right before running your k-means).
2. Randomly select k observations and use them for the initial reference vectors. I suggest you use randomize the indices and use the first k randomized indices to select the observations.
3. Use the L2 distance (Euclidean) to measure the distance between observations and reference vectors.
4. Terminate the training process when the sum of magnitude of change of the cluster centers (from the previous iteration to the current one) is less than $\epsilon = 2^{-23}$. That is, when $\sum_{i=1}^{k} d(a_i(t-1), a_i(t)) < \epsilon$ where k is the number of clusters, $a_i(t)$ is the reference vector for cluster i at time t and $d(x, y)$ is the L1 distance (Manhattan) between vectors x and y (as defined in the *Similarity and Distance Functions* link on BBlearn).
5. **Python Note:** If you are working in python, you may want to consider using opencv's Python module for generating your video. To install this locally on tux you can run the command *pip3 install –user opencv-python*.

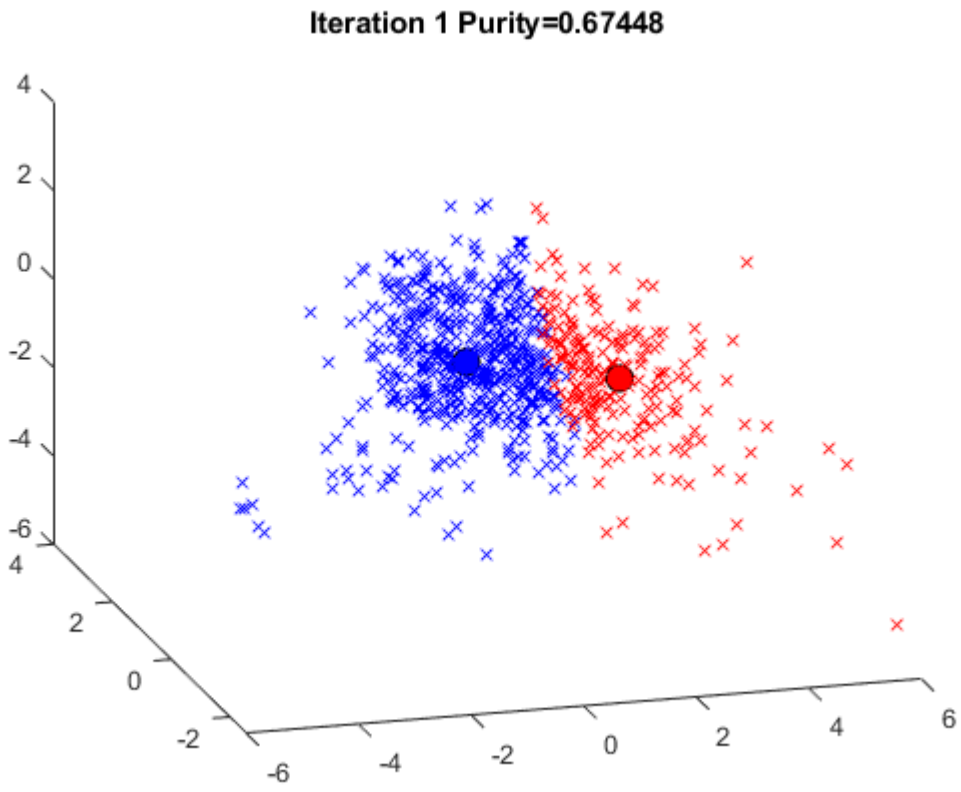

Figure 1: Initial Clustering

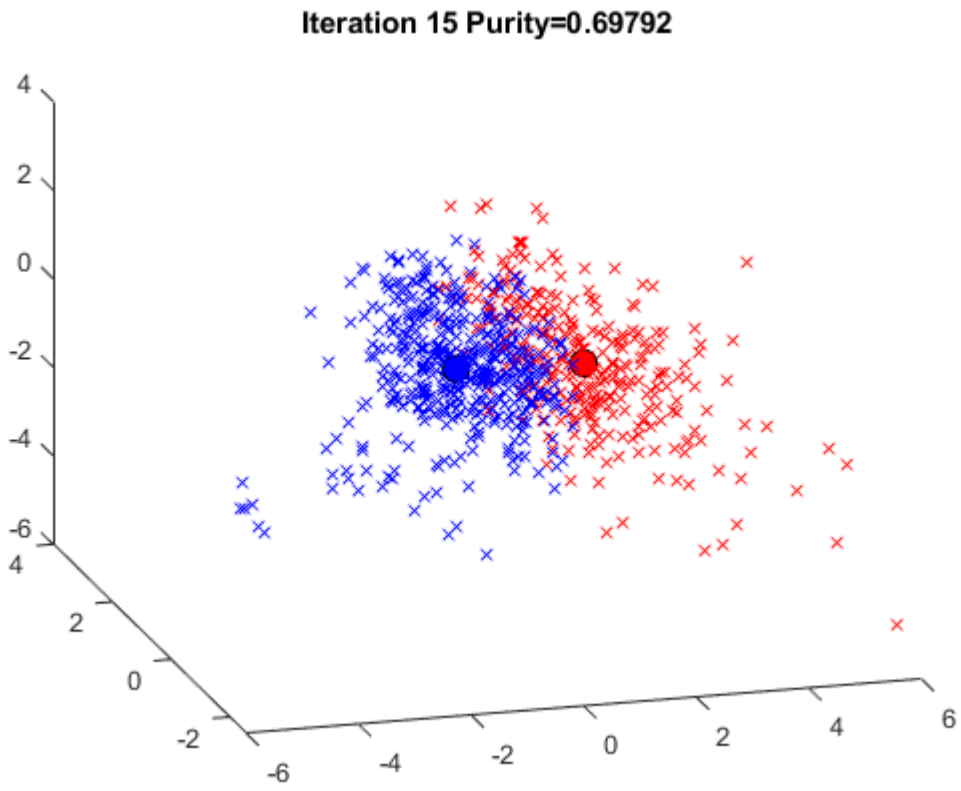

Figure 2: Final Clustering

Submission

For your submission, upload to Blackboard a single zip file containing:

1. Your solutions to the theory questions, as a typeset PDF.
2. Source Code - Including any necessary makefiles, etc..
3. readme.txt file - The readme.txt file should contain information on how to run your code to reproduce your results.
4. Sample videos of at least three different runs where you vary k and/or the features used. At least one of these must use more than two features. If you were able to get the purity part working, then these videos will include the purity in text within each frame. The video filenames should be in the format

   $$K\_[k]\_F\_[f].[ext]$$

   where:

   (a) [k] is the value of k passed to your function.

   (b) [f] is an underscore-separated list of the features passed in to your kmeans function. If you used all the features, just state *all*.

   (d) [ext] is the file extension.

   For example, in the example in Part 2, the file name would be

   $$K\_2\_F\_all.avi$$

Do not include spaces or special characters (other than the underscore character) in your file and directory names. Doing so may break our grading scripts.