Northeastern University
CS6220 – Data Mining Techniques
Fall 2017, Derbinsky

Homework 2, Part 3

This is the third part of Homework 2. Submission requirements for each of the three problems are specified below.

1 Sampling

This part of the assignment focuses on how to produce samples from various distributions. You will do all the work inside the `starter-2.3.1.ipynb` Jupyter notebook file – include this in your final submission ZIP. You have been provided an output PDF (`starter-2.3.1.pdf`) so you have a sense of what correct implementations produce.

This problem has you implement several operations from first principles. While these are often not the most practical/efficient methods, it is important you understand the basic building blocks of real code libraries you will use. In this spirit, you are **NOT** allowed to change the rather constrained imports in the notebook. Furthermore, you will be using pre-filled lists of random values (each in [0,1]) rather than making calls to various `random` module functions.

1.1 Basic Stats

In this part, you will implement `mean` and `scovariance` (sample covariance) functions. You have been provided with a few sample input-outputs in the notebook to ensure these are working properly.

1.2 Uniform Sampling

We won't be covering how to generate random numbers (a whole research area in and of itself!). So this section creates three lists (`sample1`, `sample2`, `sample3`) that each contain 10,000 values uniformly distributed in [0,1]. You will use these later on in the notebook. For now, answer the question asking whether these appear to be reasonable samples.

1.3 Inverse Transform Sampling

Inverse Transform Sampling[1] is a method for sampling values from *any* distribution, assuming you have (i) the distribution's inverse cumulative distribution $F^{-1}(x)$ and (ii) random values uniformly distributed in [0,1]. The main idea is that since the CDF is within [0,1], we can invert the function to produce coordinates that are sampled with respect to the underlying PDF.

In this section, first re-implement the CDF of a piece-wise PDF function (that you saw in the Self-Test!). Then, implement the inverse CDF. Assuming you implement these correctly, existing plotting code will produce (i) an overlaid PDF/CDF graph and (ii) overlaid samplings from this distribution using the uniform samples in the last section.

[1] `https://en.wikipedia.org/wiki/Inverse_transform_sampling`

1.4 1D Gaussian Sampling

While the method in the last section often works, it is not feasible for the Gaussian distribution (the inverse CDF does not have a clean analytical form). The simplest method (though not ideal in other respects) to sample from a Gaussian is the Box-Muller Transform[2] – it takes two uniformly distributed values in [0,1] and outputs two values distributed in $\mathcal{N}(0,1)$:

$\sqrt{-2lnU_1}cos(2\pi U_2)$
$\sqrt{-2lnU_1}sin(2\pi U_2)$

You are to first implement this function and then implement shifting/scaling of an input $\mathcal{N}(0,1)$ to an arbitrary $\mathcal{N}(\mu,\sigma^2)$. Provided code will compare this output to what Python's Gaussian sampling function produces.

1.5 1D Mixture Model Sampling

In this section you are given a mixture model with two 1D Gaussian distributions. Your task is to sample this mixture given an input list of uniformly distributed [0,1] values (used to select from the two Gaussians, given a mixing parameter), and then to comment on how the mixing parameter affects the output distribution.

1.6 2D Gaussian Sampling

Finally, you will implement 2D Gaussian sampling given two means, a covariance matrix, and two sources of uniformly distributed [0,1] values. Provided code will compare this output to what `numpy`'s Multivariate Normal sampling function produces.

[2] `https://en.wikipedia.org/wiki/Box-Muller_transform`

2 K-Means

This part of the assignment focuses on implementing K-Means, as well as parameters including K and distance metric. You will do all the work inside the `starter-2.3.2.ipynb` Jupyter notebook file – include this in your final submission ZIP. You have been provided an output PDF (`starter-2.3.2.pdf`) so you have a sense of what correct implementations produce.

2.1 Exploratory Analysis

You have been provided code to load a small dataset. You must compute and output a few statistics, as well as histograms of the values of individual features.

2.2 Distance Metrics

You are to implement functions that compute SSE and the Minkowski Distance, for arbitrary dimensions. You have been provided a couple sample inputs/outputs.

2.3 Algorithm

You must implement K-Means, given initial centroid positions, data, and a distance metric. To help modularize, you will also implement a helper function to choose the best centroid for a point (feel free to add other helpers as you see fit). There is provided code to plot the clusters – after K-Means is run with three distance metrics, there is a question to which you must respond.

2.4 Choosing K

You will use the Elbow method in order to find a good value for K. For each value of K, run with different initial centroid positions (use a sample of data points) and record SSE. Then plot SSE vs K and respond as to your choice of K (with WHY?).

3 Evaluation

This part of the assignment focuses on evaluation metrics. You are to submit a single PDF (`p3.pdf`) with the result of this section.

3.1 Cophenetic Correlation Coefficient

a. Examine Table 8.7 in the TSK text. Explain how the following cells of the table were computed: P3/P6, P2/P5, P3/P5, P2/P6.

b. Based upon Tables 8.4 and 8.7, show all work to compute the Cophenetic Correlation Coefficient. (Note that individual additions/subtractions are not necessary, but results of sums and products are – make sure to demonstrate the process.)

3.2 Purity

While not robust to increasing K, purity is a simple measure accounting for the extent to which clusters contain a single class.

a. Examine Table 8.9 in the TSK text. Show each step to compute the values in the Purity column of the table.

b. Based upon the purity metric, is this a good clustering? Which of the clusters is particularly good via this metric – provide a reasonable explanation as to why this might be true.