CS6220 Data Mining Techniques – Spring 2017

Assignment 4

Understanding clustering algorithms

1. Consider density-based clustering algorithm DBSCAN with parameters $\epsilon = \sqrt{2}, MinPts = 3$, and Euclidean distance measures. Given the following points:

$$([0,1],[5,2],[2,3],[6,1],[10,2],[0,6],[3,4],[6,3],[0,7],[7,2],[1,2])$$

   (a) List the clusters in terms of their points.

   (b) What are the density-connected points?

   (c) What points (if any) does DBSCAN consider as noise?

2. Given two clusters

$$C_1 = \{(5,6),(8,7),(7,3)\} \qquad C_2 = \{(6,5),(4,5),(9,2),(3,5),(8,4)\}$$

   compute the values in (a) - (f). Use the definition for scattering criteria presented in class. Note that tr in the scattering criterion is referring to the trace of the matrix.

   (a) The mean vectors m_1 and m_2

   (b) The total mean vector m

   (c) The scatter matrices S_1 and S_2

   (d) The within-cluster scatter matrix S_W

   (e) The between-cluster scatter matrix S_B

   (f) The scatter criterion $\frac{tr(S_B)}{tr(S_W)}$

3. Present three graphs (drawn by-hand is fine) illustrating cases where agglomerative hierarchical clustering would produce different results, depending on the distance metrics used. In particular consider the following distances: Minimum, Maximum, and Average.

Implementing and Comparing Clustering Algorithms

In this section, you will implement 3 algorithms: K-means, Gaussian Mixture models, and DBSCAN. You will then use these algorithms to compare clustering on 3 datasets, `dataset1.txt`, `dataset2.txt`, and `dataset3.txt`. Each dataset contains two columns of features and a third column with labels.

To evaluate your clustering results, consider 3 different metrics:

- The normalized mutual information (NMI)

$$\mathrm{NMI}(Y;Z) = \frac{\mathrm{I}(Y;Z)}{\sqrt{H(Y)H(Y)}}$$

  scikit-learn: `scklearn.metrics.normalized_mutual_info_score`

- The Calinski-Harabaz (CH) index $\mathrm{tr}(S_B)/\mathrm{tr}(S_W)$.

  scikit-learn: `sklearn.metrics.calinski_harabaz_score`

- The Silhouette coefficient (SC) (*see attached note*).

  scikit-learn: `sklearn.metrics.calinski_harabaz_score`

4. Implement the DBSCAN algorithm. Run the algorithm with the following *Eps* and $MinPts$ values

   `dataset1.txt`: $Eps \in \{0.2, 0.3, 0.4\}$, $MinPts \in \{2, 3, 4\}$
   `dataset2.txt`: $Eps \in \{0.8, 0.85, 0.9\}$, $MinPts \in \{6, 7, 8\}$
   `dataset3.txt`: $Eps \in \{0.2, 0.3, 0.4\}$, $MinPts \in \{5, 6, 7\}$

   Report the following information

   - Make tables for the NMI, CH index and SC values as a function of *Eps* and $MinPts$.
   - For the best result (based on NMI) make a scatter plot in which different shapes are used to show the true label, and different colors represent the learned cluster assignments. Noise points should be given a special color.

5. Implement K-means. Your implementation should perform multiple random initializations and keep the best result. Make sure that your code does not crash when one of your initializations results in an empty cluster. Finally, make sure that your implementation accepts the initial value of means as an input. We will check your implementation against a set of predetermined initializations.

For each dataset, plot the sum of squared errors (SSE), the CH index, the SC, and the NMI for the best restart (based on SSE) as a function of the number of clusters $K = 1, 2, 3, 4, 5$. For each K value, again include a scatter plot, in which different symbols are used to signify each label, and different colors represent the learned cluster assignments.

6. Now implement expectation maximization (EM) for Gaussian mixture models. As a simplifying assumption, you may take the covariance matrix for each cluster to be diagonal. Once again, your implementation should perform multiple restarts and accept initial values for the mean and variance as inputs. Initialize the mean for each of your clusters by sampling from a Gaussian distribution centered on a random point in your data. Initialize the variance along each dimension to a random fraction of the total variance in the data.

   For each dataset, plot the log likelihood, the CH index, the SC, and the NMI as a function of $K = 1, 2, 3, 4, 5$. Also include the scatter plots that you made for k-means, using $z_n = \arg\max_k \gamma_{nk}$ to determine the cluster assignments.

   Hint: When implementing your E-step, you need to be careful about numerical underflow and overflow when calculating

   $$\gamma_{nk} = p(x_n, z_n = k \,|\, \theta) / \sum_l p(x_n, z_n = l \,|\, \theta).$$

   A common trick is to first calculate the log probabilities

   $$\omega_{nk} = \log p(x_n, z_n = k \,|\, \theta), \qquad \omega_n^* = \max_k \omega_{nk}.$$

   Before exponentiation you can now first subtract ω_n^*, which is equivalent to dividing both the numerator and denominator by $\exp \omega_n^*$

   $$\gamma_{nk} = \exp(\omega_{nk} - \omega_n^*) / \sum_l \exp(\omega_{nl} - \omega_n^*).$$

   Explain in your code comments why this helps prevent underflow/overflow.

7. From above implementations, you may see different models perform differently on these three datasets:

   - Which model performs best on `dataset1`? Can you explain why?
   - Which model performs better on `dataset2` and `dataset3`, GMM or K-means? Can explain the reason for this?