Linnaeus University

Introduction to Machine Learning, 2DV516

Rafael M. Martins

`rafael.martins@lnu.se`

Assignment 4: Unsupervised Learning

Introduction

In this assignment you will implement two unsupervised learning methods, then you will test them with data sets of your choice. You can deliver the results in either (a) a jupyter notebook, combining the code, text, and images in a nice readable sequence; or (b) send the source code and the text separately, with the full report (with text and images) in a PDF file.

Exercise 1: Clustering

Goal: Implement the clustering algorithm called *Bisecting k-Means*.

Bisecting k-Means [1] is a clustering algorithm that combines hierarchical clustering with k-Means. However, differently than the hierarchical clustering we saw in the lecture, it uses a *divisive*, top-down approach (instead of the *agglomerative*, bottom-up that we are used to). It consists on the steps described below:

1. Start with a single cluster including all the observations in the data set.
2. [*Bisecting*] Divide the largest cluster into two smaller sub-clusters using k-Means.
3. Redo the *bisecting* step `iter` times and choose the *best* solution according to the *Sum of Squared Errors* (SSE).
4. Repeat from Step 2 until you have `k` clusters.

Implement the Bisecting k-Means algorithm in a function called `bkmeans`. It should take as **input**: (a) the data `X` to cluster, as a $n \times p$ matrix (n observations by p features); (b) the number `k` of clusters; and (c) the number `iter` of iterations for step 3. It should generate as **output** a $n \times 1$ vector with the cluster indices for each of the n observations.

Notes:

1. You can use Python or Matlab; the requirements are exactly the same. In case of Python, make sure that `X` is a Numpy array of shape `(n, p)`.
2. You do not need to re-implement K-Means; you can use an existing implementation (e.g. from `sklearn` or a Matlab built-in).
3. The requirements are **strict**. I will use standard test cases in order to test your solution by calling the function `bkmeans` as described.

Exercise 2: Non-linear Dimensionality Reduction

Goal: Implement the non-linear dimensionality reduction algorithm known as *Sammon Mapping*.

Sammon Mapping [2] is one of the first non-linear dimensionality reduction algorithms, and it is still used today as a benchmark due to its flexibility and good results. What differentiated Sammon Mapping from other MDS algorithms proposed at the time was the use of non-linear scaling, as opposed to most MDS techniques which scaled all distances by the same value (see the lecture slides or the original paper for details).

The algorithm can be implemented with gradient descent using the following steps:

1. Start with a random two-dimensional layout Y of points (Y is a $n \times 2$ matrix).
2. Compute the stress E of Y. See slide 47 of Lecture 10 for the formula.
3. If $E < \epsilon$, or if the maximum number of iterations `iter` has been reached, stop.
4. For each y_i of Y, find the next vector $y_i(t+1)$ based on the current $y_i(t)$. See slide 48 of lecture 10.
5. Go to Step 2.

Implement Sammon Mapping in a function called `sammon`. It should take as **input**: (a) the data `X` to reduce, as an $n \times p$ matrix (n observations by p features); (b) the maximum number `iter` of iterations for step 3; (c) the error threshold ϵ for step 3; and (d) the learning rate α for step 4. It should generate as **output** a $n \times 2$ vector with the final two-dimensional layout.

Notes:

- Be careful with division by zero when computing the gradient descent. If the denominator of a division is too small, almost zero, the gradient descent will behave strangely. Treat it by limiting how small the denominator can be; replace anything below an acceptable threshold with a fixed, small constant.
- The notes from Exercise 1 also apply here.

Exercise 3: Visualization of Results

In this exercise you will visualize and explore the results of the previous two exercises in a simple manner, using scatterplots. This will be a relatively open-ended task; you will choose three data sets and explore them with the new toolset you built for yourself. These could be data sets you already used in previous assignments, or you could download some new data. The only restriction is that the data sets must be multidimensional (i.e., more than 4 features) and must have labels.

These are some examples of interesting places to obtain new data sets:

- `http://archive.ics.uci.edu/ml/index.php`
- `https://www.openml.org/search?type=data`
- `https://www.kaggle.com/datasets`

Be careful, however, with the size of the data set you choose. Matlab or Python can get quite slow with too much data, and the scatterplots will also be very crowded, so go for smaller data sets this time (I'd say, around 1000 observations).

3.1. Comparison of DR Techniques

Generate a scatterplot matrix comparing the results of your implementation of Sammon Mapping with PCA and t-SNE for each data set. The resulting visualization should be a 3×3 matrix where each cell is a scatterplot of a DR technique applied to a data set. Color the points by their target variables (i.e., class/labels) using a qualitative colormap.

Then answer this shortly (in a couple of paragraphs): In your opinion, which technique performed the best for each data set, regarding the separation of the classes? How are the classes in the data sets separated? Are some classes easier to separate than others?

3.2. Comparison of Clustering Techniques

Choose one of the DR techniques from the previous exercise and generate a similar scatterplot matrix to compare the results of Bisecting k-Means with classic k-Means and hierarchical clustering for each data set. The resulting visualization should be a 3×3 matrix where each cell is a scatterplot of the chosen DR technique applied to a data set, with the colors of the points showing the clusters using a qualitative colormap (see, e.g., https://matplotlib.org/tutorials/colors/colormaps.html).

Then answer this shortly (in a couple of paragraphs): In your opinion, which clustering technique performed the best for each data set? How are the clusters in the data sets separated? Are some clusters easier to separate than others?

References

[1] M. Steinbach, G. Karypis, V. Kumar *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, no. 1. Boston, 2000, pp. 525–526. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/fetch/papers/docclusterKDDTMW00.pdf

[2] J. W. Sammon, "A Nonlinear Mapping for Data Structure Analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, 1969. [Online]. Available: https://pdfs.semanticscholar.org/154f/8a9906bcc99fca9b17aa521649b1c3734093.pdf