Spectral clustering, semi-supervised learning

Instructions : You can work as a binome. You will provide a report that includes your answers to this practical session as well as the Python implementations.

Your report will take the form of a **UNIQUE** linear document *IPython Notebook* `.ipynb`.

It is mandatory to name it as `Name1_Name2_TP2_group_k.ipynb` (where $k \in \{1, 2\}$ is your group number). You will submit your report before 11 :59pm, 2016-11-28, on the Moodle of the Data Science Master (`http://datascience-x-master-paris-saclay.fr/`), in the section associated to the course Machine Learning : from theory to practise. The maximal score for the TP is 5.
Malus : **1** pt for each 12h period -(except a good reason validated by administration) ; score divided by p when p answers are identical in reports.

- Spectral Clustering -

Questions

Spectral Clustering on toy datasets

The goal of this first exercise is (i), to address clustering problems when data belong to some manifold, (ii), to measure the impact of noise on spectral clustering and (iii), measure its stability especially when the graph construction parameter varies.

1. Generate a few datasets of the kind 'noisymoon' using :

   `noisy_moons = datasets.make_moons(n_samples=n_samples, noise=.05)` for various levels of noise.

   Plot the different datasets.
2. For each dataset, build several graphs by varying the hyperparameter k in the k-nearest neighbor construction method. Visualize the graph (or easier, the adjacency matrix) in each case.
3. Recall the optimization problem leading to the Normalized-cut Spectral Clustering [3] (NB : in Scikit-Learn the normalized cut is implemented).
4. Run the two-ways spectral clustering algorithm of Scikit-Learn on each graph and visualize the results in 2D. For each level of noise, what is the effect of modifying k ?
5. Stability[4] is a statistical criterion to measure the relevance of an algorithm. We'll reuse the stability measure defined in the paragraph 3.1 of [1]. A Stability measure can be built as a distance between the partition given by a clustering algorithm on a complete sample and the partitions obtained with a clustering algorithm applied on "bootstrap" samples of the original one. In this case, stability is measured my running B times the clustering algorithm on "bootstrap" samples without replacement containing a large percentage of the initial dataset (like 80%) and comparing the obtained clusters with the one computed on the complete sample. The matrix representation of a partition will be given by $C_{ij} = 1$ if element i and j belong to the same cluster and 0 otherwise. The normalized dot product between 2 matrix representations of a partition can then be seen as a cosine similarity distance and will be used as our similarity measure between the original partition and a bootstrap based one (see [1]). Implement a general stability method that will apply on a clustering algorithm, a dataset, a number B and a number C of clusters.
6. Some algorithms are well known to be relatively instable like hierarchical clustering, on the contrary, others like spectral clustering exhibit an excellent stability. Apply this method to measure the stability of spectral clustering on a non-noisy moon dataset and on a noisy moon dataset. Given $C = 2$, does the stability criterion help to choose k, the hyperparameter of the graph construction ?
7. Now compute the stability of hierarchical clustering (you will choose the most appropriate distance between clusters) on the same datasets.

Spectral Clustering on MNIST dataset

Now, we will test the spectral clustering on a real dataset containing many classes. For that purpose, use a stratified subsample of the MNIST dataset with 10 digits. The goal of the exercise is to run the full approach and to study the relevance of hyperparameter k and C. We will check if the stability property of Spectral Clustering still holds in this case and is of any help to choose C. You will compare this procedure to the eigengap one to choose C.

- Semi-supervised Learning -

8. We now assume we have a training dataset that contains a few labeled data and a large number of unlabeled data. We want to learn a prediction function using a semi-supervised learning method. Choose a dataset among "sklearn.dataset" (http://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets) and build your semi-supervised dataset. We advise you to choose a binary classification data set or to transform a multiclass classification problems into one-versus all binary classification problems. In the following, we will avoid to do cross-validation and will study the impact of hyperparameters but we will not choose them. Then you need two datasets, one for training and one for testing. We will vary the number of unlabeled sample in our observations.
9. Implement the self-training algorithm and apply it with k-nearest neighbors on a toy dataset (can be the two-moons) to check your code and on the chosen real dataset.
10. Implement LapRLS, seen during the lecture, and introduced by [2] with ℓ : number of labeled training, u : number of unlabeled training data, $n = \ell + u$ size of the training set, $\mathbf{f}$: vector composed of the values $f(x_1), \ldots, f(x_n)$. $\mathcal{H}_\mathcal{K}$ is the RKHS associated to kernel $\mathcal{K}$. L is the Laplacian of the data graph defined as follows : it is a $n \times n$ matrix $L = D - W$, where W is the adjacency matrix of a graph built from the k-nearest neighbor method applied on the distance matrix of the training data. In semi-supervised learning, we often simply take : $w_{ij} = \exp(-\gamma \|x_i - x_j\|^2)$.
    - Solve $\min_{f \in \mathcal{H}_\mathcal{K}} \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - f(x_i))^2 + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} + \frac{\lambda_u}{u+\ell}^2 \mathbf{f}^T L \mathbf{f}$. Hint : after application of the representer theorem for this convex loss, the solution admits an expansion in the following terms :

      $$f^*(\cdot) = \sum_{i=1}^{\ell+u} \mathcal{K}(\cdot, x_i)\alpha_i^*$$

      In practice, we take a constant function added to the function of the RKHS : $g_{\alpha^*,b}(x) = f_{\alpha^*}(x) + b$. Then, solve the minimization problem in α^* and b and give the closed-form expression of vector α.
11. Provide a first implementation of LapRLS using the closed-form. Test it on the same toy and real previous problems using for instance the Gaussian kernel. Fix the value of λ and play with λ_u. What do you observe ?
12. What are the limitations of a closed-from solution implementation ?
13. Provide a second implementation of LapRLS using the stochastic gradient descent algorithm. Test it on the same dataset as before. Again play with the hyperparameters and plot the errors on the test sample.
14. Now let us work on LapSVM [2] which minimizes :

    $$\min_{f \in \mathcal{H}_\mathcal{K}} \frac{1}{\ell} \sum_{i=1}^{\ell} (1 - y_i f(x_i))_+ + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} + \frac{\lambda_u}{u+\ell}^2 \mathbf{f}^T L \mathbf{f}.$$

    Use the representer theorem to express f^* the minimizer as :

    $$f^*(\cdot) = \sum_{i=1}^{\ell+u} \mathcal{K}(\cdot, x_i)\alpha_i^*$$

    Again, we take $g_{\alpha^*,b}(x) = f^*(x) + b$. α^* (and b) can be obtained by (1) writing the semi-supervised problem in the primal space, (2) applying Lagrangian duality, and (3) eventually exhibiting a reduced Lagrangian that can be solved. The goal of this question is to implement LapSVM using this reduced Lagrangian formulation that boils down to use QP-SVM appropriately. Hint : you can find help in the paper [2]. Note that this implementation is expensive in computation time as well as in memory. More efficient implementations requires to deal with non-smooth penalties such as subgradient algorithms or proximal algorithms.

15. Compare Self-training, LapRLS and LapSVM on a real dataset for various percentages of labeled data. For the self training algorithm, you can vary the base learner.
16. Finally propose a proper protocole to select automatically the hyperparameters for a semi-supervised task.

- Useful links -

— Spectral clustering in scikitlearn : http://scikit-learn.org/stable/modules/clustering.html

— Stability : an overview, U. Von Luxburg. https://arxiv.org/pdf/1007.1075v1.pdf

— Manifold regularization, M. Belkin, P. Nyogi, V. Sindwani, JMLR 2016.http://web.cse.ohio-state.edu/~mbelkin/papers/MR_JMLR_06.pdf

Références

[1] Isabelle Guyon Asa Ben-Hur. Detecting stable clusters using principal component analysis. *Functional Genomics : Methods and Protocols*, 2003. 1

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization : A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7 :2399–2434, 2006. 2, 3

[3] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4) :395–416, December 2007. 1

[4] Ulrike von Luxburg. Clustering stability : An overview. *Found. Trends Mach. Learn.*, 2(3) :235–274, March 2010. 1