Geometry of Information - Final Project: attempt to improve SpectralNet with local kernel method

Elky Sandor and Nachman Keren

19th February 2022

1 SpectralNet algorithm: motivation and methods

In our work, we tried to improve a neural network algorithm named SpectralNet. In this section, we are willing to review [1]. This article offers a comprehensive survey about SpectralNet.

The article opens with a review of Spectral Clustering algorithm (see [2] for details). This algorithm includes methods for low-dimension embedding and clustering by finding eigenvalues of a Leplesian matrix induced from a kernel matrix of the data. The authors of [1] point out two problems with this method:

1. **Time complexity:** Finding eigenvalues is indeed a problem that can be solved in polynomial running time ($\approx O\left(n^3\right)$)[1], but given high dimensional data, even such running time is problematic and can be unrealistic.
2. **Generalization problem**: Spectral Clustering algorithm provides a new representation of the information but does not provide a trivial tool for applying the dimensionality reduction method that was deduced by a given data for a new unseen data. Given new information, the algorithm has to be run again and in particular eigenvalue decomposition is needed and this is a problem that requires computational resources.

The authors of [1] refer to several studies that have been done on the subject and which offer solutions to these problems. We will briefly review two of these articles here:

1. In [4], there is a suggestion to solve the time complexity problem by using Neistrom approximation. In this context, let us note that (as we have seen in the second homework exercise in the course) sometimes this approximation does not provide good results as it does not take into account large parts of the data.
2. In [5], there is a suggestion to solve the generalization problem by using the eigenvalues and eigenvectors calculated using the original data. Assuming that the new data has the same distribution, instead of recalculating the spectral decomposition, it is proposed to produce an approximate embedding function that uses information from the initial decomposition. This approach has a clear disadvantage: the initial model cannot be improved by new information without starting again from scratch.

To solve the problems mentioned above, the authors of [1] propose a new course of action. The direct calculation of the eigenvalue decomposition of a graph Laplacian matrix L induced from the kernel matrix W is being replaced by a natural network. The network, once trained, computes an embedding function $F_\theta : \mathbb{R}^d \to \mathbb{R}^{\mathrm{k}}$ (usually $k \ll d$) by minimizing the following loss function:

$$L_{\text{SpectralNet}}(\theta) = \frac{2}{m^2}\text{trace}\left(Y^T (D - W) Y\right) = \frac{1}{m^2}\sum_{i,j} ||y_i - y_j||^2 \mathsf{W}_{\mathsf{i,j}}$$

[1] The complexity of approximate solutions to the eigenvalue problem is a little lower. see [3].

where $F_\theta(x_i) = y_i$, and D is the degree matrix of the graph (recall that $L = D - W$). while the real minimum of $L_{\text{SpectralNet}}(\theta)$ is achieved by plugging the eigenvectors of L corresponding to the lowest eigenvalues, iterated minimalization of $L_{\text{SpectralNet}}(\theta)$ (by using stochastic gradient descent[2]) should give a good approximation of these eigenvalues. After the low-dimentional embedding is found, the point in $\mathbb{R}^k$ is clustered using k-means algorithm. This method has several benefits that meet the problems mentioned above:

1. In terms of computational power, neural network training (which is actually done by random sampling of $n < m$ data points at each iterative stage) is easier than finding spectral decomposition of very large matrices. In contrast to the use of Neistrom approximation as proposed by [4], in which a subset of information is randomly sampled once - the network training is done by multiple random samples of subsets at each stage of iteration. Therefore, in practice, we do not ignore parts of information, but creating optimization based on the entire information.

2. Regarding the generalization problem: once the network is trained, we will have an embedding function to which we can enter new information which the network has not been trained on without running the algorithm again. Here, compared to the method proposed by [5], if we wish - we could continue to train the network given new information.

Let us note that if we want an approximation of the low eigenvectors of the normalized Laplacian, we can replace the loss function with the following function:

$$L_{\text{SpectralNet}}(\theta) = \frac{1}{m^2}\sum_{i,j}\left|\left|\frac{y_i}{d_i} - \frac{y_j}{d_j}\right|\right|^2 W_{ij}$$

Another proposed improvement in [1] is using neural network in order to determine the values of the matrix W. Instead of using the Gaussian kernel, the authors of the article used the "Siamese network" that creates a map $x_i \mapsto z_i$ with the feature that if x_i is similar to x_j, $||z_i - z_j||^2$ will be small and vice versa. The values of W is then calculated by $e^{-\frac{||z_i - z_j||^2}{\sigma^2}}$. According to [1] this method improves the model capabilities in comparison to the use of Euclidean distance.

2 Self-Tuning Spectral Clustering: suggestion for an improved kernel matrix

One can easily spot that SpectralNet loss function is defined merely by the graph weights matrix W. Hence, the crucial importance of this matrix for the proper operation of the algorithm is clear. A usual way to produce such a matrix given data is by calculating a kernel matrix[3].

The suggetion in [1] is using the popular Gaussian kernel calculated by $[W]_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$ for some constant $\sigma > 0$. The authors of the article suggest improving this method by using a natural network for finding an embedding of the original data before calculating the kernel. According to the article, the use of this method significantly improves the performance of the model, however unfortunately the training of this particular network requires tens of hours of computer calculation (of our poor little laptops) so we were not able to implement it.[4].

[2]For further details regarding stochastic gradient descent see section 5.9 in [6].

[3]Kernel matrices have the following special property: the matrix entries are defined by an inner product of an embedding of the data in some functions space (to be exact - Hilbert space). The exact definition and the theoretical justification for using such matrices will not be discussed here. Chapter 12 in [7] includes a theoretical discussion of the subject, examples, and bibliographic details.

[4]It should be noted that the large resources required for this method are not our private problem but a significant and general weakness of the model. The use of Siamese neural network is suggested by [1] in this context; there may be another "lighter" network (or method) that will provide similar results.

In light of the above, it is safe to assume that an improvement of the kernel matrix construction will cause improvement in the algorithm. A natural candidate for such improvement is proposed in [8]. In this article, a new way of choosing the bandwidth parameter σ^2 is suggested.

Accroding to [8], the scaling parameter is some measure of when two points are considered similar. thus, Instead of selecting a single scaling parameter σ, it is proposed to calculate a local scaling parameter σ_i for each data point x_i. The distance from x_i to x_j as 'seen' by x_i is $\frac{||x_i - x_j||_2}{\sigma_i}$ while the converse is $\frac{||x_i - x_j||_2}{\sigma_j}$. Therefore the squared distance in the Gaussian kernel may be generalized as $\frac{||x_i - x_j||_2^2}{\sigma_i \sigma_j}$ The affinity between a pair of points can thus be written as $\exp\left(-\frac{||x_i - x_j||^2}{\sigma_i \sigma_j}\right)$.

Moreover, [8] suggests an implicit way for choosing the local scaling parameter: $\sigma_i = ||x_i, x_K||_2$,where x_K is the K'th neighbour of x_i. This method is called Self-Tuning Spectral Clustering and according to [8] this local scaling leads to better clustering preformances.

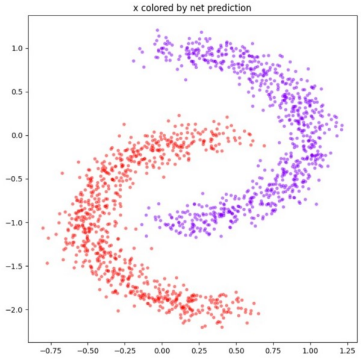
It is now natural to suggest replacing W that is being used in [1] with the local bandwidth parameter of [8].

3 Training SpectralNet with Self-Tuning kernel: experimental results

To test our proposal, we used two data sets. We used the code attached to [1][5] (written in Tensorflow). We The details and the results follow:

1. **Horizontal moons**: The first and simpler dataset on which we tried our new method is noisy two moons divided into two clusters

   Using universal bandwidth parameter[6], we succeeded in achieving an almost perfect separation between the two clusters. The evaluation metrics[7] for this following experiments were ACC $= 0.999$, NMI $= 0.993$:

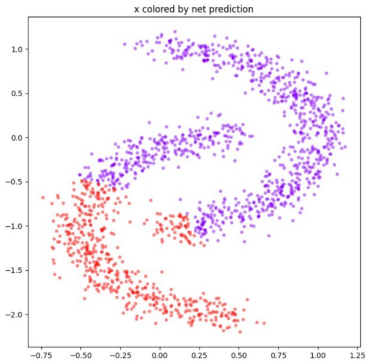


   Surprisingly, the Self-Tuning method achieved worse results with evaluation metrics of ACC $= 0.773$, NMI $= 0.263$:

[5] https://github.com/KlugerLab/SpectralNet

[6] The scale σ was set to be the median distance between a point to its 3rd neighbor.

[7] For the exact definition of ACI (unsupervised clustering accuracy) and NMI (normalized mutual information) see section 5.1 in [1].



2. **MNIST**: we tried both methods on MNIST data sets and achieved similar results:

	Univesal σ	Self-Tuning σ
ACC	0.793	0.792
NMI	0.813	0.812

   Although, to achieve these results, the net performed 120 learning epochs for the universal σ as opposed to 132 epochs for the Self-Tuning σ.

We tried to use different choices of the parameter K in the Self-Tuning Spectral Clustering algorithm,but with no help. In contrary to our hypothesis, we see that the use of Self-Tuning σ is a disadvantage for the model. We will try to find out the reason for this result in the following section.

3.1 Is Self-Tuning Spectral Clustering a problematic method? experimental and theoretical remarks

After these allegedly surprising results which contradict the claims of [8], we tried to dive deeply into the Self-Tuning Spectral Clustering algorithm and find out its weakness. We found out that even with the classic Spectral Clustering method (eigenvalue decomposition of a graph Laplacian) the promising results are somewhat disappointing in reality.

For example, we take the horizontal moons dataset we saw above. No choice of K was able to successfully separate the two clusters with the Self-Tuning Spectral Clustering algorithm. But only after a few trials and errors we found σ that succeed to separate the clusters perfectly. It seems that this method is highly sensitive to noisy data. In addition we found that in some cases this algorithm is highly sensitive to the choice of the K parameter. Due to this insight, we can deduce that algorithm is replacing the σ-choosing problem with an equivalent one - choosing K.

Farther deepening is required to understand when this algorithm fails and why. We noticed[8] that the transition to local σ_i instead of universal σ for all the data can easily produce matrices that are not semi-positive defined i.e. this method is not kernel method.

4 Summary

In this project we dived deep into the world lies in between deep learning through neural networks and learning based on spectral graph theory. We tried to apply a method we learned in the classical context of kernel-based

[8]Both with artificially made counterexamples and with real data.

learning in the world of deep learning but without success. We believe that there is a need for further examination of methods for creating a local kernel in the context of data fed into a neural network in general and in the context of SpectralNet in particular. It is posible that this method will work better with datasets with clusters are not equally distributed.

Much of the difficulty we faced in working on the project stemmed from the complexity of the topic of deep learning. This subject, which is new to us, required of us many self-learning abilities. In the theoretical aspect we faced concepts and a different way of thinking than the ones we have encountered so far in the contexts of machine learning. And in the practical aspect our work required learning the basics of the TensorFlow programming language, which is fundamentally different from libraries such as NumPy and Scikit-learn. We are grateful for the opportunity given to us to get to know a rich topic like deep learning and access it from places that are familiar to us and we are confident that we will continue to engage in the subject in the future as well.

References

[1] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks, 2018.

[2] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[3] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *STOC '99*, 1999.

[4] Chung Fowlkes, Belongie and Malik. Spectral grouping using the nystrom method. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26(2), pages 214–225, 2004.

[5] Yoshua Bengio, Jean-Francois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. *Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering.* MIT Press, advances in neural information processing systems 16 edition, January 2004.

[6] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, Cambridge, MA, USA, 2016.

[7] Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

[8] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.