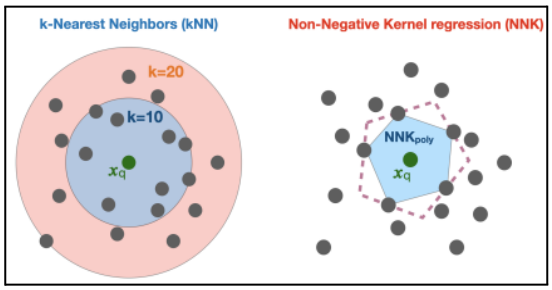
Project Report: Non-Negative Kernel Graphs

Ansh Khandelwal

Introduction:

Machine learning and signal processing applications frequently employ data-driven neighbourhood definitions and graph constructions. Even though they frequently need a somewhat ad hoc selection of their parameters, i.e., k, k-nearest neighbour (kNN) is the approach that is most frequently employed among them. This is because of their computational simplicity. The non-negative kernel regression (NNK) algorithm, as put out by [1], has been described in this project. Utilising this novel approach, one may find neighbourhoods that result in improved sparse representation. In a graph construction or graph learning problem, we are given N data points with feature vectors {x1, x2 . . . xN } ∈ Rd, and the goal is to obtain an efficient graph representation, for example, one where the number of edges is of the same order as the number of nodes. Such sparse graph constructions lead to faster downstream tasks in image processing etc. Local neighbourhood definition or construction is often the starting point for graph-based algorithms in scenarios where no graph is given a priori, and a graph has to be constructed to fit the data. Even though techniques for neighbourhood selection based on the choice of a single optimal value for k exist, they can fail in non-uniformly distributed datasets where it might be desirable to adapt the number of neighbours (k) to the local characteristics of the data. The NNK approach considers a novel interpretation of neighbourhood definition as a non-negative sparse approximation. So the problem is all about signal representation.



We can clearly see the geometric difference in neighbourhood definitions using k-nearest neighbours (kNN) and the proposed NNK approach with a Gaussian kernel. kNN selects neighbours based on the choice of k and can lead to the selection of neighbours that are geometrically redundant, i.e., the vector from the query to neighbours is almost collinear with other vectors joining the query and neighbours. In contrast, the proposed NNK definition selects only neighbours corresponding to non-overlapping directions.

NNK approach:

Algorithms previously proposed for solving graph learning can be broadly classified into two main families: similarity-based and locality-inducing. The similarity-based approaches like KNN compute a similarity metric between every pair of points and associate this value to the edge weight between the corresponding nodes. Methods in this class sparsify the fully connected graph to reduce complexity while seeking to maintain the properties of the original graph. Locality-inducing approaches compute edge weights around a given data point so as to reflect local characteristics of data better points. The optimisation problem for LLE(local linear embedding) is as follows:

$$\boldsymbol{\theta}_S = \arg\min_{\boldsymbol{\theta}} \; ||\boldsymbol{x}_q - \boldsymbol{X}_S\boldsymbol{\theta}||_2^2$$

where Xs is the matrix containing the k-nearest neighbours of xq (query), whose indices are denoted by set S. A solution to can produce both positive and negative values for the entries of θ, which would be the weights associated with the elements in the set S.

The NNK approach is to view graph construction as a signal representation problem. So the problem statement is that Given N data points {x1, x2, . . . , xN }, the neighbourhood definition for a query xq is the problem of obtaining a weighted subset of the given N data points that best represents the query. NNK can be viewed as a sparse representation approach where the coefficients θ for the representation are such that the error in representation is orthogonal to the space spanned by the selected atoms. We note that this reformulation results in neighbourhoods that are adaptive to the local geometry of the data and robust to parameters such as k. For a query xq, the NNK weights θ are found by solving the following: $\boldsymbol{\theta}_S = \arg\min_{\boldsymbol{\theta}:\, \boldsymbol{\theta}\geq\boldsymbol{0}} \; ||\boldsymbol{\phi}_q - \boldsymbol{\Phi}_S\boldsymbol{\theta}||_2^2$. Where ΦS corresponds to the kernel space representations of an approximate set of data point candidates S for the neighbourhood, such as the k nearest neighbours. Here, a positive definite kernel $k(x_i, x_j)$, corresponds to a transformation of data points in Rd to points in a Hilbert space, such that similarities can be interpreted as dot products in this space, i.e., $k(x_i, xj) = \varphi_i{}^T\varphi_j$, where φ is a mapping from data space to transformed space and φi represents the transformed observation xi. Now, employing the kernel trick i.e., $k(x_i, xj) = \varphi_i{}^T\varphi_j$, the objective in can be rewritten as the minimization of

$$\mathcal{J}_q(\boldsymbol{\theta}) \; = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{K}_{S,S}\boldsymbol{\theta} - \boldsymbol{K}_{S,q}^\top\boldsymbol{\theta} + \frac{1}{2}\boldsymbol{K}_{q,q}$$

.

Consequently, the NNK neighbourhood is obtained as the solution to the optimization problem, $\boldsymbol{\theta}_S = \underset{\boldsymbol{\theta}:\,\boldsymbol{\theta}\geq\mathbf{0}}{\arg\min}\ \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{K}_{S,S}\boldsymbol{\theta} - \boldsymbol{K}_{S,q}^\top\boldsymbol{\theta}$ with non-zero elements of the solution θS corresponding to the selected neighbours, and weights are given by the corresponding reconstruction coefficients. For graph construction, the i-th row of the adjacency matrix W is given by $W_{i,S} = \theta_s$ and $W_{i,S^c} = 0$. To get an undirected graph, an edge conflict occurs, say between nodes i and j , due to the varying influence of one node over the other node's representation. In such a scenario, the approximation error corresponding to nodes i and j, namely Ji (θ) and Jj (θ) from the above equation, is considered, and the weight corresponding to the smaller of the objective is kept. The KRI (kernel ratio interval) states that for any positive definite kernel with a range in [0, 1] (e.g. bilateral kernel (1)), the necessary and sufficient condition for two nodes j and k to be both connected to node i in an NNK graph is : $\boldsymbol{K}_{j,k} < \frac{\boldsymbol{K}_{i,j}}{\boldsymbol{K}_{i,k}} < \frac{1}{\boldsymbol{K}_{j,k}}$.

NNK graph construction algorithm:

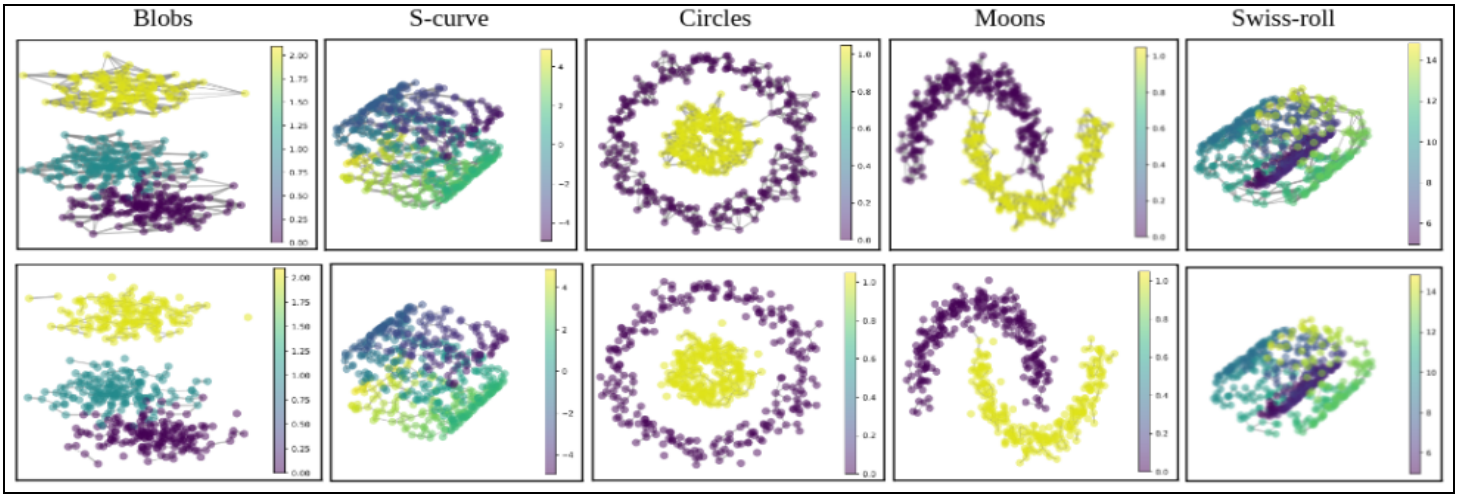
Input : Data $\mathcal{D} = \{\boldsymbol{x}_1 \cdots \boldsymbol{x}_N\}$, Kernel function κ,
No. of neighbors k
for $i = 1, 2, \ldots N$ **do**
$S = \{$ k nearest neighbors of $\boldsymbol{x}_i$ in $\mathcal{D}_{-i}\}$
$\boldsymbol{\theta}_S = \underset{\boldsymbol{\theta}\geq\mathbf{0}}{\arg\min}\ \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{K}_{S,S}\boldsymbol{\theta} - \boldsymbol{K}_{S,i}^\top\boldsymbol{\theta}$
$\mathcal{J}_i = \frac{1}{2}\boldsymbol{\theta}_S^\top \boldsymbol{K}_{S,S}\boldsymbol{\theta}_S - \boldsymbol{K}_{S,i}^\top\boldsymbol{\theta}_S + \frac{1}{2}\boldsymbol{K}_{i,i}$
$\boldsymbol{W}_{i,S} = \boldsymbol{\theta}_S, \quad \boldsymbol{W}_{i,S^c} = \mathbf{0}$
$\boldsymbol{E}_{i,S} = \mathcal{J}_i\,\mathbf{1}, \quad \boldsymbol{E}_{i,S^c} = \mathbf{0}$
end
$\boldsymbol{W} = \mathbb{I}(\boldsymbol{E} \leq \boldsymbol{E}^\top) \odot \boldsymbol{W} + \mathbb{I}(\boldsymbol{E} > \boldsymbol{E}^\top) \odot \boldsymbol{W}^\top$
Output: Graph Adjacency $\boldsymbol{W}$, Local error $\boldsymbol{E}$

Experiment:

I have implemented the NNK and KNN algorithm for graph construction. The idea is to test both these algorithms for different types of synthetically generated datasets and determine whether the NNK algorithm generates a sparser graph. I will compare the number of edges these algorithms make during graph learning. The below comparison has been done for k=5 neighbours. The datasets used are synthetically generated using the popular sklearn library of Python. The datasets used are – make_moons, make_blobs, make_circles, make_s_curve, and make_swiss_roll.

Results and observations:

We see from the plots (the ones on the top row are made from the KNN approach, and the bottom ones belong to NNK) that the number of edges made by NNK is significantly lesser. The plots are made from 500 nodes for different synthetic datasets. It can be observed that the NNK algorithm is superior in terms of geometric representation (sparsity) from the KNN algorithm for all types of datasets. Also, the experiments have been done for varying nodes from 50 to 1000, and the results have been tabulated. We can observe that edge to node ratio for KNN is approximately three times for the make_blobs dataset. The edge-to-node ratio is always higher for the KNN algorithm. We can also observe that edge to node ratio mostly remains consistent with an increase in the number of nodes for the KNN and NNK algorithms. Since we have been looking at the graph construction from a sparsity angle, the closer the edge-to-node ratio is to one, the better graph. Therefore, these experiments show that NNK is superior to the KNN algorithm for making sparser graphs.



Number of nodes	Datasets and the corresponding number of edges formed									
	make_blobs		make_s_curve		make_circles		make_moons		make_swiss_roll	
	NNK	KNN	NNK	KNN	NNK	KNN	NNK	KNN	NNK	KNN
50	50	163	62	149	56	159	55	146	75	148
100	112	325	120	306	123	290	116	292	122	312
500	610	1563	631	1499	606	1558	571	1591	624	1531
1000	1253	3067	1238	2998	1203	3121	1202	3127	1271	2997

Downstream task: NNK for image representation

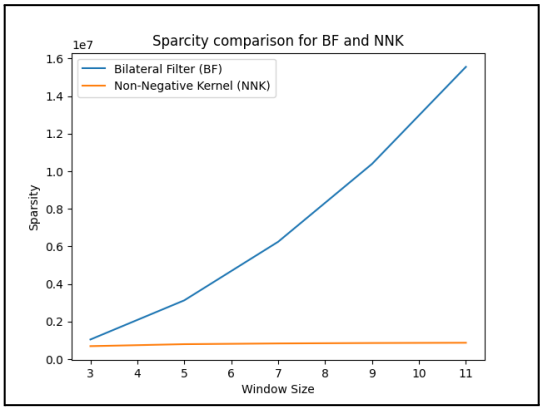
A recent trend in image processing has been to move from simple non-adaptive filters to image-dependent filters such as the bilateral filter. One shortcoming of these adaptive filters is that they cannot be efficiently described using traditional image domain Fourier techniques since these models are highly non-linear. Conventional analysis based on fourier and grid-based transforms does not apply to non-linear filters. Therefore, graph-based spectral analysis can be used. One can represent images as graphs and pixel intensity as graph signal. The eigenspectrum provides a frequency basis structure similar to fourier. So in terms of graph representation, each pixel is a node and edge weight is referred to as the similarity between pixel intensities. One approach to do this is to use window-based filters like bilateral filters, where each centre pixel is connected to its neighbouring pixels. The bilateral filter kernel is as follows:

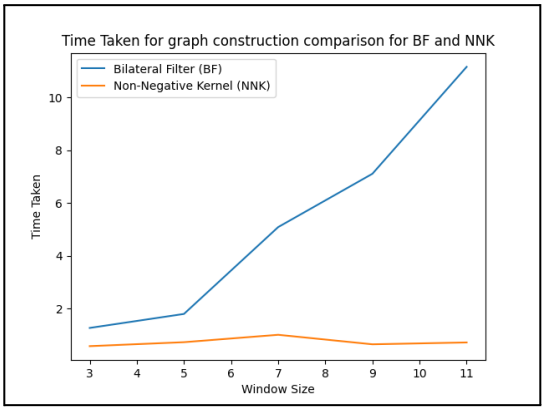
$$\boldsymbol{K}_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_d^2}\right)\exp\left(-\frac{\|\boldsymbol{f}_i - \boldsymbol{f}_j\|^2}{2\sigma_f^2}\right) \qquad \begin{array}{ll}\boldsymbol{x}_i: & \text{pixel position}\\ \boldsymbol{f}_i: & \text{pixel intensity}\end{array}$$

However, the issue with this method is that the sparsity of the image graph is only dependent on window size w. Therefore, NNk graphs are used as these are robust to hyperparameters, and edges are defined by the local geometry of the data. The optimization can be written only regarding similarity information, as mentioned in the above equations for NNK. The NNK for images optimizes the KRI condition as follows: $(\boldsymbol{f}_j - \boldsymbol{f}_k)^\top(\boldsymbol{f}_j - \boldsymbol{f}_i) < \left(\frac{\sigma_f}{\sigma_d}\right)^2 (\boldsymbol{x}_k - \boldsymbol{x}_j)^\top(\boldsymbol{x}_j - \boldsymbol{x}_i)$. Unlike in the general case, where all data dimensions are irregular and unknown, in image graphs, pixel locations are regularly spaced and are known beforehand. This observation allows reducing the KRI condition to simple intensity thresholding rules for removing neighbours in images. This helps in efficiently using the NNK algorithm for image representation. The right term in the above equation depends only on pixel locations and kernel parameters and can be determined for a given window size w and saved beforehand. Thus, in contrast to KNN graphs, even when window size w increases, the number of connected nodes does not necessarily grow, so NNK graphs tend to reflect the actual data topology better (due to the KRI theorem).

Experiments and results:

To check how the change in hyperparameter w (window size) changes the graph construction via bilateral filters and NNK graphs, I tested both algorithms on an image by varying the filter size. We can clearly observe that the sparsity value increases (almost linearly) with an increase in window size for bilateral filters. However, for the NNK algorithm, sparsity is mostly consistent. The low value of sparsity helps with several downstream tasks, as it helps in efficiently representing the images in space. We also see an increase in time for graph construction with an increase in window size. But the time taken remains the same for the NNK algorithm for graph (sgwt) construction due to its dependence on pixel intensity and geometric variability of pixels.





(Top: Bilateral Filter Graph vs Bottom: Proposed NNK Graph Construction)
The experiment: the window size is increased from left to right (from 3X3 to 11X11) for both approaches. The idea is to evaluate image representation by calculating the variance of the wavelet signals as an indicator of information content corresponding to the frequency band of the wavelet. The more the variance, the more the energy of the image captured by the approaches will be. We can observe that the NNK approach captures more energy for all the window sizes than the BF method. We can also see the variance decreases with an increase in window size. This is intuitive as more neighbouring nodes connected will affect the image representation and make it more distorted.

References:

1. **S. Shekkizhar and A. Ortega, "Graph construction from data by non-negative kernel regression," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 3892–3896.**
2. **Shekkizhar, S., & Ortega, A. (2020). Efficient graph construction for image representation. ArXiv. /abs/2002.06662**
3. **H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," IEEE Transactions on Image Processing, vol. 16, no. 2, pp. 349–366, 2007.**
4. **C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in Sixth International Conference on Computer Vision. IEEE, 1998, pp. 839–846**