Sreehari Sankar (sxs2284@case.edu)
Dec 2019,
Machine Learning Final Project,
Case Western Reserve University

Kernel functions for Graph-based Semi-Supervised Learning.

Graph based Semi-supervised learning is one of the many different types of semi-supervised learning. Here, labeled and unlabeled data is represented in a graph, where the edges weights reflect the “similarity” between the data in a manner of speaking.
Of course, different implementations have their own modifications of the above specified detail and is another way the structure of semi-supervised learning differs.

For my final project, I implement multiple Kernels functions and analyze them and their properties.
Kernel function are extremely critical for graph based semi-supervised learning (As we shall see later on) and the choice of a kernel has a direct and rather large impact on the given classifier’s performance.

Here, since we are analyzing Kernel functions, the classifiers are irrelevant (Although I do analyze how different algorithms mix with different kernel functions later on), I have obtained permission from the professor to use library functions for ease of implementing the classifier on which I will test the Kernel functions, which is the main focus of my final project.

Kernel functions map the “closeness” or “similarity” and allow the weights of the graph to be set. There are many families of Kernel functions, the most common being a lower order Gaussian.

I started this project based on the Section 6.1.7 from the seed paper given which is SSL_literature survey (X.Zhu, 2008), but have ended up analyzing more than just those family of kernel functions mentioned in that section, which is from the spectrum of laplacian:

The first paper is on a family of kernel functions called diffusion Kernels:
Diffusion Kernels on Graphs and Other input spaces, Kondor, Lafferty, 2002.

The second paper, from which I have taken and Implemented the mentioned Kernel functions are:
Kernels and Regularization on Graphs, Smola, Kondor.
This contains several Kernel functions and a brief analysis of them.

Part 1: Regularization and Laplacian Kernels

!!! Paper[1]

Starting with paper: Kernels and Regularization on Graphs, Smola, Kondor.

Most of this paper focuses on defining Laplacian operators on graphs and their properties.

Here, the analysis important to Kernel functions begins in Section 3 (Regularization), given the fact that L(Laplacian) induces a semi-norm on f(features), then penalizes the changes between adjacent features, and this property is exploited to design regularization operators, which will serve as Kernel functions.

According to the paper:

Kernels are obtained by solving the self-consistency condition [Smola et al., 1998]

$$\langle k(x,\cdot), Pk(x',\cdot)\rangle = k(x,x') \, .$$

Where Pf is given by:

$$\langle f, Pf\rangle := \int |\tilde{f}(\omega)|^2 \, r(\|\omega\|^2)\, d\omega = \langle f, r(\Delta) f\rangle \, .$$

Here, $r(\|\omega\|^2)$ gives the function penalizing the frequency components $|\tilde{f}(\omega)|$ of f, which typically increases for $\|\omega\|^2$.

Using these results given in the paper, it goes on to prove that $k(x,x') = \kappa(x-x')$ where K is the inverse Fourier Transform of $r^{-1}(\|\omega\|^2)$, which is also outlined to be one of the properties of Laplacian Kernel functions.

Fundamentally, functions r that give solutions are acceptable to build Kernel functions. The two most common such functions are:

	$r(\|\omega\|^2)$	$k(x,x')$
Gaussian RBF	$\exp\left(\frac{\sigma^2}{2}\|\omega\|^2\right)$	$\exp\left(-\frac{1}{2\sigma^2}\|x-x'\|^2\right)$
Laplacian RBF	$1+\sigma^2\|\omega\|^2$	$\exp\left(-\frac{1}{\sigma}\|x-x'\|\right)$

Where RBF is the Radial Basis Function.

These are two major and very important function and is seen throughout semi-supervised, being used as kernels most frequently in both major algorithms of LabelPropogation and LabelSpreading.

Now, the paper defines a set of functions for r() and their associated Kernel functions given by:

$$
\begin{aligned}
r(\lambda) &= 1 + \sigma^2 \lambda && \text{(Regularized Laplacian)} \\
r(\lambda) &= \exp\left(\sigma^2 / 2\lambda\right) && \text{(Diffusion Process)} \\
r(\lambda) &= (aI - \lambda)^{-1} \text{ with } a \geq 2 && \text{(One-Step Random Walk)} \\
r(\lambda) &= (aI - \lambda)^{-p} \text{ with } a \geq 2 && \text{(p-Step Random Walk)} \\
r(\lambda) &= (\cos \lambda \pi / 4)^{-1} && \text{(Inverse Cosine)}
\end{aligned}
$$

And

$$
\begin{aligned}
K &= (I + \sigma^2 \tilde{L})^{-1} && \text{(Regularized Laplacian)} \\
K &= \exp(-\sigma^2 / 2\tilde{L}) && \text{(Diffusion Process)} \\
K &= (aI - \tilde{L})^p \text{ with } a \geq 2 && \text{(p-Step Random Walk)} \\
K &= \cos \tilde{L} \pi / 4 && \text{(Inverse Cosine)}
\end{aligned}
$$

//Note:cos() is cosine distance function. (To resolve any ambiguity with the trig function cosine)

I have, in my project implemented these Kernel functions and measured differences in accuracy across different datasets. (As seen in later sections)

Now, this paper goes on to prove that the canonical family of kernels on graphs are of the form of power series of the graphs in Laplacian and that such kernels can be characterized as real-valued functions of the eigenvalues of the Laplacian. Since these proofs are largely mathematical and extensive and owing to the fact that we are interested in the performance of these kernel functions in the context of semi-supervised learning, I do not repeat it here and instead will be giving the results of the use of these multiple different types of Kernel functions for SSL.

Part 2: Diffusion Kernels and related topics

!!! Paper[2]

Diffusion Kernels on Graphs and Other Discrete Input Spaces. (Kondor, Lafferty)

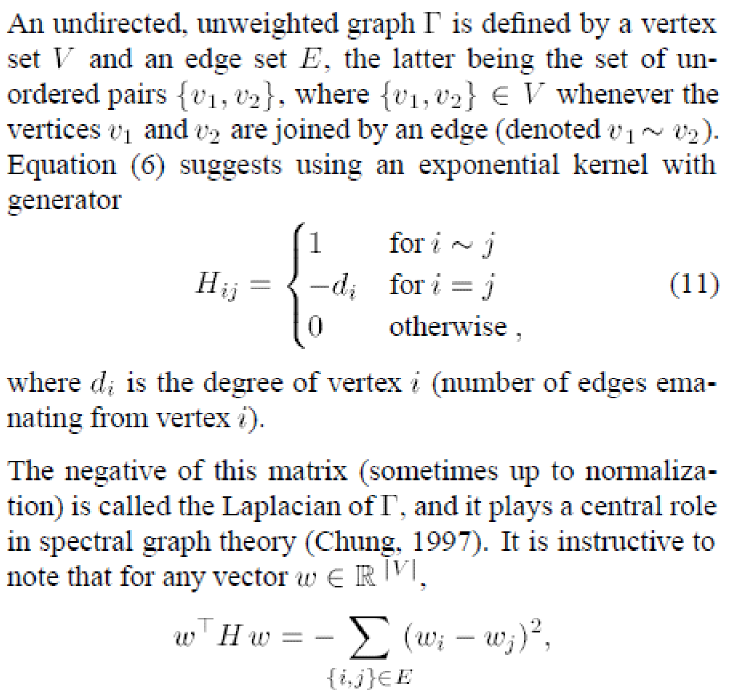
One of the types of Laplacian Kernels that is of particular interest in the Diffusion Kernel, which is specifically considered next, taking "Diffusion Kernels on Graphs and Other Discrete Input Spaces, Kondor, Lafferty." as the paper behind it.

The key ideas discussed here are the positive semi-definiteness of kernel functions, given by

$$\int_{\mathcal{X}} \int_{\mathcal{X}} f(x)\, f(x')\, K(x, x')\, dx\, dx' \quad \geq \quad 0$$

and then the given the critical differential

equation: $\frac{d}{d\beta} K_\beta = H K_\beta,$, from which the kernel function called the class of "Diffusion Kernels" or "Heat Equation Kernels" are derived.
Following is a direct extract from the paper, showing the relationship between these family of matrices and the Laplacian kernel functions.



An undirected, unweighted graph Γ is defined by a vertex set V and an edge set E, the latter being the set of unordered pairs $\{v_1, v_2\}$, where $\{v_1, v_2\} \in V$ whenever the vertices v_1 and v_2 are joined by an edge (denoted $v_1 \sim v_2$). Equation (6) suggests using an exponential kernel with generator

$$H_{ij} = \begin{cases} 1 & \text{for } i \sim j \\ -d_i & \text{for } i = j \\ 0 & \text{otherwise ,} \end{cases} \tag{11}$$

where d_i is the degree of vertex i (number of edges emanating from vertex i).

The negative of this matrix (sometimes up to normalization) is called the Laplacian of Γ, and it plays a central role in spectral graph theory (Chung, 1997). It is instructive to note that for any vector $w \in \mathbb{R}^{|V|}$,

$$w^\top H\, w = - \sum_{\{i,j\} \in E} (w_i - w_j)^2,$$

// EXTRACTED FROM PAPER[2]

!! This is important as I have also implemented a rudimentary version of LabelPropogation, upto the point where it builds the graph, which is considered to be the most important step in any graph-based SSL algorithm. Although Kernel functions are the main topic of this project, this is for extra credits, if applicable. !!

Also, this provides context on how these family of kernel functions relate to the graph that is built for SSL.

GAUSSIAN KERNEL AND THE DIFFUSION KERNEL:
The Gaussian Kernel, is the most widely used Kernel function and is probably recognized as the "Default" Kernel function if not specified otherwise.

The following has been extracted from paper[2], and it gives a conclusive idea about diffusion kernels and their relationship to the family of Gaussian Kernels.

$$\frac{d}{d\beta} K_\beta = H K_\beta$$

Here we have the famous Heat equation.

Now, using $k_x(x') = K(x, x')$ we have:

$$\frac{d}{d\beta}\, k_x(x') = \int H(x, x'')\, k_{x''}(x')\, dx'' \,.$$

Since the Laplacian is a local operator in the sense that $\Delta f(x)$ is only affected by the behavior of f in the neighborhood of x, as long as $k_x(x')$ is continuous in x, the above can be rewritten as simply

$$\frac{d}{d\beta}\, k_x(x') = \Delta\, k_x(x') \,.$$

It is easy to verify that the solution of this equation with Dirac spike initial conditions $k_x(x') = \delta(x - x')$ is just the Gaussian

$$k_x(x') = \frac{1}{\sqrt{4\pi\beta}}\, e^{-|x-x'|^2/(4\beta)},$$

showing that similarity to any given point x', as expressed by the kernel, really does behave as some substance diffusing in space, and also that the familiar Gaussian kernel on $\mathbb{R}^m$,

$$K(x, x') = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-|x-x'|^2/(2\sigma^2)}$$

is just a diffusion kernel with $\beta = \sigma^2/2$. In this sense, diffusion kernels can be regarded as a generalization of Gaussian kernels to graphs.

//EXTRACTED FROM PAPER[2]

The above function is implemented as regular_gaussian(x,y,gamma) in code and results have been collected and condensed.

We see that the so called “Diffusion Kernel” should behave more or less like the regular Gaussian kernel. This similarity shows the universality of the Gaussian kernel.
Laplacian Kernels are one of the few families of kernels that are fundamentally different.
This also goes to show how many and most kernels designs reduces to the regular Gaussian.

Further, Laplacian Kernels have associated degrees, given by:

$$K^n(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i=1}^{n} K(x_i, x_i') \,,$$

or, using the tensor product notation, $K^n = \bigotimes_{i=1}^{n} K$.

In part three, an analysis of the effect of degree on runtime v/s accuracy has been done, which reflects the scalability of such kernel functions for SSL purposes.

Lastly, before going to part three, which is the analysis, I would also like to identify a set of kernel functions that are not from any of the previously considered families of functions.

These will be used to compare results and do analysis. Briefly, they are:
//Sorry about not being able to put down an image regarding these, took the equations from LaTeX code//

Haversine Kernel : D(x,y)=2arcsin[sqrt{sin^2((x1 - y1) / 2 + cos(x1)cos(y1)sin^2((x2 - y2) / 2)}]
Chi Square Kernel: e^(-gamma*Sum[(x - y)^2 / (x + y)])
Euclidean Kernel : D(x,y)=sqrt(sum((x-y)^2))
Hamming Kernel : D(x,y)=Hamming_Dist(x,y)
Jaccard Kernel: D(x,y)=Jaccard_Dist(x,y)

// All these have been implemented in the code, as they have been defined in the papers to compare.

Part 3: Implementation of the Papers in code (Kernels), Analysis, Results, Conclusion

Here, all the above mentioned Kernel Functions have been implemented (See Code) from the chosen papers and analyzed.

From Paper[1] We have the set of Laplacian Kernels:

H_O_Laplacian is the Higher order Laplacian Kernel of order 3.
These are compared against the Cosine Correlation and Hamming Kernels for reference.
Also a comparison is done in paper[1].

Accuracy	Reg_Laplacian	Inverse_Cosine	Gen_Diffusion_process	CC_Kernel	Hamming	H_O_Laplacian
Isolet	0.81701	0.80577	0.91974	0.8266	0.5	0.90288
Haberman	0.8474	0.83050	0.78	0.8477	0.7796	0.8474

Notes:
Higher_Orders of the Laplacian Kernels take exponentially large amounts of time to run.
I wanted to plot this, but it took so long, I could not realistically do that.

As we can see, the hamming distance kernel performs the worst among them, and all Laplacian kernel functions are performing much better, with the Diffusion Kernel from the spectrum of Laplacian doing best.

From Paper[2] we have:
Diffusion Kernel : From Paper[2], Given earlier (Not the same as Laplacian variant)
Regular Gaussian Kernel: Given above, taken from paper[2].
Higher order Gaussian Kernel: Given in Paper[2], with degree set to 3.

Here, I also compare across Gaussian and Laplacian Kernels, to compare directly.

Accuracy	Regular_Gaussian	Reg_Laplacian	Diffusion	Hamming	H_O_Laplacian	RBF_Gaussian
Isolet	0.906	0.845	0.92	0.5	0.902	0.64
Haberman	0.8305	0.845	0.72	0.7796	0.89	0.71

Here, we can conclusively see that Kernels from the Spectrum of Laplacian outperform almost all other types of Kernel functions.
These are :
Diffusion Kernels,
Regular and Higher Order Laplacian Kernels.

Since it is proved in paper[2] that Laplacian Kernels can be reduced to Gaussian, using appropriately selected functions, we can understand why it outperforms Gaussians.
Laplacian Kernels can be specifically modified for use on tree data structures and other discrete structures also. (Refer paper[2]).

All this ensure that the Laplacian family of Kernels can be an ideal choice for any graph-based SSL algorithm.

Going by the absolute values of accuracy, we can see that of all the functions I have implemented and tested, Regular Laplacian from the Spectrum of Laplacian is giving the Highest accuracy. Hence, I am selecting that for further testing.
Although Diffusion Kernels give good results, it is highly dependent on the input data and hence we can conclude that it is highly unstable. For this reason, I will not be selecting that for further analysis.

Further Experiments and analysis on Laplacian family of Kernel functions:
Conducted on isolet dataset.
Regular Laplacian Kernel from the Spectrum of Laplacian has been chosen.

Degree	1	2	3	4
Time	119s	372s	501s	1121s

NOTE: TIMES ARE COMPUTER-SCALED, ARCHITECTURE DEPENDANT

Here, we can clearly see the non-linear growth of time with respect to degree of the Laplacian Kernel.

Next, I use haberman dataset,

Scalability of accuracy:

Percent UL	20	40	60	80	90
Acc	0.846	0.74	0.71	0.7	0.69

Since this scalability is giving such results, I found it extremely important and continued with it.

Percent UL	90	95	99	99.99	99.999
Acc	0.69	0.69	0.4	0.26	0.2

Although this is a function of the number of elements in the dataset, we can see that it scaled very very well indeed., Only going below 50% after around 95-99 percent was unlabeled. Again, I understand that this is also dependent on the number of samples vs the dimension, but the fact that the Kernel function scaled so well indicated definite scalability and reflection of accurate data.

The hyper parameters chosen for these experiments are 'neighbors' and 'tol'. Neighbors are the number of neighbors a graph vertex can have. This value has an inverse distribution over accuracy and the inverse of that over time required. This is set such after much experimentation with the runtimes and accuracies manually. Tol is simply the learning rate. Set so that it is low enough but not so low that it does not converge.

Conclusion:

Kernels for SSL, from the Spectrum of Laplacian

In implementing and analyzing the two papers, we have understood the impact Kernel functions have on graph-based SSL algorithms. We have also seen the families of Kernel functions and their general behaviors. We have seen how the Laplacian family of Kernel functions scale and their general adaptability. We have seen how they outperform almost all other Kernel functions. Although they have drawbacks like the non-linear increase in time required for computation, Laplacian Kernels gives extremely good results in terms if accuracy and scaling.
As said in Paper[2], Laplacian Kernels are to be the preferred choice for SSL algorithms, since they can also very easily be modified for specific purposes and retain their general characteristics.

Additionally, I have attempted a rudimentary label-propogation algorithm that works as a regressor taken from paper "Learning from Labeled and Unlabeled Data with Label Propagation, X.Zhu". Due to the general complexity of LabelPropogation, I could not completely finish it as a classifier. This was an attempt for extra credits. It does work and give regression results though. So, it works only for regression dataset. Unfortunately due to the general complexity of SSL and its associated algorithms, I could not proceed any further since it would take a large amount of time and effort and given the general conceptual complexity of Kernel functions, I could not proceed any further can create codified results for that Additional part.

References, Citations

Paper[1] is Referring to "Kernels and Regularization on Graphs, Smola, Kondor."

Paper[2] is Referring to "Diffusion Kernels on Graphs and Other Discrete Input Spaces. (Kondor, Lafferty)"

Additional Papers :" Learning from Labeled and Unlabeled Data with Label Propagation, X.Zhu"

```
!!IMPORTANT: Since these Kernel functions are independent of
classifiers, I had previously requested clearance for using
Library functions for implementing these classifiers.
They are:
import numpy as np
from scipy.spatial.distance import cdist, pdist, squareform
import math
import pandas as pd
import random
import sklearn.semi_supervised as lp
import copy
```