Homework 10

Name:

Student ID:

- Reasoning and work must be shown to gain partial/full credit
- Please include the cover-page on your homework PDF with your name and student ID. Failure of doing so is considered bad citizenship.

1. (6 points) **Graph Signal Processing**: In this problem, you will use a existing package "pygsp[1]" to finish some simple tasks regarding to graph signal processing.

   (a) (0.5 points) Generate ErdosRenyi graph with $N = 50$, and $p = 0.1$ and ring graph $N = 50$ by pygsp, e.g. *GR = graphs.Ring(N=50)*. Plot the two graphs.

   (b) (0.5 points) Load the citation network dataset, Cora and plot this graph. The Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words. (hint: *from torch_geometric.datasets import Planetoid* and *dataset = Planetoid(root='/tmp/Cora', name='Cora')*)

   (c) (1 points) We consider here only undirected graphs, such that the Laplacian matrix is real symmetric, thus diagonalizable in an orthonormal eigenbasis (i.e., eigen-decomposition)

   $$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top,$$

   where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_N) \in \mathbb{R}^{N\times N}$ is the matrix of orthonormal eigenvectors and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is the diagonal matrix of associated sorted eigenvalues:

   $$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N.$$

   Plot the graph frequencies of the ErdosRenyi, pygsp and Cora graphs from the smallest to largest. Plot $\mathbf{u}_2$, $\mathbf{u}_3$ and $\mathbf{u}_6$ over graph, as shown in Slide 39, lecture 22 (hint: `https://networkx.org/documentation/stable/auto_examples/drawing/plot_node_colormap.html`).

   (d) (1 point) The spectral content of a signal indicates if the signal is low-pass, band-pass, or high-pass. Again, intuition transfers from classical Fourier analysis.

   1. Build a graph by *G = graphs.Sensor(N=100, seed=1)*.
   2. Compute the graph fourier basis by *G.compute_fourier_basis()*.
   3. Create two kinds of graph signals as follows.
      (a) Generate random signals by $\boldsymbol{x}_0$ = *np.random.RandomState(10).normal(size=G.N)*.
      (b) Define two filters *g = pygsp.filters.Heat(G, tau)* by setting $tau = 0$ and $tau = 10$, respectively.
      (c) Use the two filters to process the above random signals $\boldsymbol{x}_0$ by $\boldsymbol{x} = g.filter(\boldsymbol{x}_0)$, and then obtain two kinds of graph signals, where one is low-pass and one is high-pass.
   4. Calculate the graph Fourier transform for the above low-pass and one is high-pass graph signals to obtain the graph signal in Graph Fourier domain $\hat{\boldsymbol{x}}$ = *G.gft(**x**)*.
   5. Plot the graph spectral content of $\hat{\boldsymbol{x}}$, i.e. $|\hat{\boldsymbol{x}}|$.
   6. Evaluate the smoothness of the two graph signals, i.e. $\boldsymbol{x}^\top\mathbf{L}\boldsymbol{x}$
   7. Is the graph signal generated by $tau = 0$ a high-pass graph signal or low-pass graph signals?

   (e) (1 point) Consider the feature data of Cora. Compute the covariance matrix, and utilize the TSNE to obtain two principal components for clustering, i.e., *n_components=2*.

[1] `https://pygsp.readthedocs.io/en/stable/`

(f) (2 points) Let's define a low-pass filter

$$\tilde{h}(\lambda) = \frac{1}{1+\alpha\lambda}$$

Given a noisy version of a smooth signal $\boldsymbol{x}_{\text{noisy}}$, one can denoise it with the low-pass filter g:

$$x_{\text{denoised}} = \mathbf{U}\tilde{h}(\mathbf{\Lambda})\mathbf{U}^{\top}\boldsymbol{x}_{\text{noisy}}$$

1. Consider the ring graph *GR = graphs.Ring(N=50)*
2. Consider the following graph signals $\boldsymbol{x} = [\sin(i) + \cos(2i) + 0.5\cos(30i)]$, where $i = np.linspace(-np.pi + 2 * np.pi * j/N, np.pi + 2 * np.pi * j/N)$, $j = 0, \cdots, N-1$.
3. Add white noise to x by $np.random.uniform(-0.4, 0.4, size = N)$ to graph signals $\boldsymbol{x}$.
4. Plot the noisy graph signals.
5. Compute the Graph Fourier domain $\hat{\boldsymbol{x}}$, and plot the graph spectral content of $\hat{\boldsymbol{x}}$, i.e. $|\hat{\boldsymbol{x}}|$
6. Define the low-pass filter
$$\tilde{h}(\lambda) = \frac{1}{1+\alpha\lambda}$$
by setting $\alpha = 2$ by $g = pygsp.filters.Filter(G, g)$.
7. Utilize g to process the noisy graph signals $\boldsymbol{x}$ with method $x\_denoised = g.filter(x, method =' exact')$.
8. Plot the denoisy graph graph signals, and its graph spectral content.

2. (4 points) **Graph Neural Networks**: In this problem, you will use an existing package "pyg[2]" to build your own graph neural networks to finish some simple classification tasks. There is a tutorial of graph neural network implementation [3].

   (a) (2 points) Build a two-layer Chebyshev Graph Neural Network class by *from torch_geometric.nn import ChebConv* in Pytorch. For the first layer, the input channel is the dataset feature number, and output channel is 40, the order of GSO $K = 5$. For the second layer, the input channel is 40 and the output channel is dataset label classes. In the forward function, you should use the *relu* as the first layer activation function, and the output activation is *log_softmax* to obtain the predicted labels.

   (b) (1 point) Train your graph neural network by utilizing *torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)*. You can change the learning rate *lr* and *weight_decay* according to your own model. And loss function is suggested to use *torch.nn.functional.nll_loss* or any other loss functions that are suitable for your GCN model. Plot your training loss with the epoch as the x-axis.

   (c) (1 point) Test your graph neural network to report the accuracy. Visualize the graph with node labels for both the ground truth and the predicted ones.

[2] `https://github.com/pyg-team/pytorch_geometric`

[3] `https://pytorch-geometric.readthedocs.io/en/latest/notes/introduction.html`