Midterm Homework

DS-GA 1008 Deep Learning

Fall 2021

Midterm Homework covers Geometric Deep Learning topics.

For theory questions, you should put all your answers in a PDF file and we will not accept any scanned hand-written answers. It is recommended to use LaTeX.

For the coding question, you need to program in Python. We recommend you to use **PyTorch** and **NumPy** for the experiments and **Matplotlib** to draw the plot. You need to submit your python files and describe the results with plots in the pdf.

If you only submit your python files without relevant description and plots in the pdf, you will get no points for the questions.

The midterm homework has a total of 100 points, divided into 50 points for theory and 50 for coding. **You need to pick two out of the three theory questions to answer**. If you choose to answer all three, the two with highest scores will be kept.

The due date of midterm homework is `23:55 EST` of `11/9`. Submit the following files in a zip file `your_net_id.zip` through NYU Brightspace:

- `theory.pdf`
- `GNN.py`

The following behaviors will result in penalty of your final score:

1. 5% penalty for submitting your files without using the correct format. (including naming the zip file, PDF file or python file wrong, or adding extra files in the zip folder, like the testing scripts from coding questions).
2. 20% penalty for late submission within the first 24 hours. We will not accept any late submission after the first 24 hours.
3. 20% penalty for code submission that cannot be executed using the steps we mentioned in coding questions. So please test your code before submit it.

1 Short Questions (Pick two out of three)

This section asks questions that admit a short and direct answer, but you need to provide mathematical justification thoughout.

1.1 GDL Blueprint and Compositional Architectures (25 pt)

The GDL blueprint considers a composition of feature transformations of the form $\varphi_k : \mathcal{X}(\Omega_k, \mathcal{C}_k) \to \mathcal{X}(\Omega_{k+1}, \mathcal{C}_{k+1})$. Here, $\mathcal{X}(\Omega, \mathcal{C})$ denotes the space of signals defined over the domain Ω and having features in $\mathcal{C}$ in each point of the domain. In other words, $x \in \mathcal{X}(\Omega, \mathcal{C})$ if $x(u) \in \mathcal{C}$ for each $u \in \Omega$. The previous mapping φ_k maps from a domain Ω_k to a coarser domain Ω_{k+1}, possibly by modifying the number of feature maps $\mathcal{C}_k$ to $\mathcal{C}_{k+1}$. Additionally, we constrain each φ to be *local*, that is, the output $\varphi_k(x)(\tilde{u})$, where $\tilde{u}$ is a point in Ω_{k+1}, only depends on $\{x(v); \|v - 2u\|_\infty \leqslant \delta\}$.

In this exercise, we will explore *why* this compositional non-linear structure is necessary, by studying limitations of architectures that replace any of these components by simple linear coarsening.

Assume a representation using two layers as above, ie $\Phi(x) = \varphi_2 \circ \varphi_1(x)$, with Ω_1 the original 2-d grid of an image, and Ω_2 a coarsened version (say by a factor of 2 along each horizontal and vertical dimensions), and $\Omega_3 = \{u\}$ a singleton; in other words, $\Phi(x) \cong \mathcal{C}_3$ is a delocalised feature vector (think of a global pooling). Moreover, for simplicity assume a coarsening step with stride 2, and set $\delta = 1$.

1. (5pt) Draw the diagram of the representation Φ in terms of its factors φ_1 and φ_2.
2. (10pt) Suppose now that φ_1 is a *linear* map, given by an average pooling. Construct an example of a two-class classification problem where the resulting representation Φ won't be able to separate the two classes, irrespective of φ_2.
3. (10pt) Suppose now that φ_2 is a *linear* map, given by an average pooling. Construct an example of a two-class classification problem where the resulting representation Φ won't be able to separate the two classes, irrespective of φ_1.

1.2 Invariant Polynomials (25 pt)

Consider a geometric domain given by a 1-d periodic grid $\Omega = \{1, \dots d\}$, and a signal domain $\mathcal{X}(\Omega, \mathbb{R}) \cong \mathbb{R}^d$. We consider two symmetry groups defined in the domain, the *cyclic* group $\mathcal{C}_d$ given by cyclic shifts of Ω, ie $g.(x_1, \dots, x_d) = (x_k, x_{k+1}, \dots, x_d, x_1, \dots x_{k-1})$, and the *symmetric* group $\mathcal{S}_d$ given by all permutations $\sigma : \Omega \to \Omega$. The study and design of generic invariant function approximations starts with a good understanding of the set of invariant polynomials (since any continuous function can be well approximated by a polynomial of a sufficiently large degree).

1. (5pt) Give an example of a polynomial p in $(x_1, \dots, x_d)$, ie

$$p(x_1, \dots, x_d) = \sum_{k=0}^{K} \sum_{1 \leqslant i_1 \leqslant i_2 \leqslant \dots \leqslant i_k \leqslant d} \alpha_{i_1, \dots, i_k} x_{i_1} \dots x_{i_k}$$

   such that it is *not* invariant to neither of these groups.

2. (10pt) Give an example of a polynomial that it is invariant with respect to $\mathcal{C}_d$ but *not* invariant with respect to $\mathcal{S}_d$.

3. (10pt) Can you give an example of the opposite, ie a polymonial that is invariant to $\mathcal{S}_d$ but not invariant to $\mathcal{C}_d$? Justify your answer.

1.3 Graph Isomorphism and Graph Convolutional Networks (25 pt)

Let $G = (V, E)$ be an unattributed undirected graph, with nodes V and edges E. Given G, let $A \in \{0,1\}^{|V|\times|V|}$ be its adjacency matrix, where $A_{ij} = 1$ iff $(i,j) \in E$. Consider a simplified GCN architcture that takes as input the binary adjacency matrix of a graph, constructs initial node features $x_i^{(0)}$ given by node degrees $d_i := \sum_j A_{ij}$, $i \in V$, and propagates node features according to the diffusion $x_i^{(k)} = \sigma\left(\alpha_k x_i^{(k-1)} + \beta_k \sum_j A_{i,j} x_j^{(k-1)}\right)$, where $k \leqslant K$ is the layer index (ie, the number of diffusion steps), and σ an activation function such as the ReLU. The output of the GCN is defined as $\Phi(G) = \frac{1}{|V|}\sum_{i\in V} x_i^{(K)}$, ie the average pooling of the last layer. This GCN is therefore parametrised by $\Theta = (\{\alpha_k, \beta_k\})_{k\leqslant K} \in \mathbb{R}^{2K}$.

Despite being one of the most popular GNN architectures, the GCN is not a universal approximator. In this exercise, you will construct some example tasks that, while being permutation invariant, cannot be solved by GCNs.

1. (10pt) Draw two non-isomorphic graphs G, G' for which the outputs $\Phi(G), \Phi(G')$ will be always the same, irrespective of Θ. [*Hint*: Recall the 1-Weisfeller-Lehman test.]

2. (10pt) Fix $K = 1$ and give an example of a graph prediction task $f : G \to \{0,1\}$ that Φ is provably unable to solve. [*Hint*: Use the previous question]

3. (5pt) Give an example of a graph prediction task $\tilde{f} : G \to \{0,1\}$ that Φ is provably unable to solve when $K = 1$ but able to solve when $K = 2$.

2 Coding Question: GNNs for the Stochastic Block Model (50 pt)

In this exercise, you will train a Graph Neural Network to perform community detection on the Stochastic Block Model (SBM). You can check Chen et al. (2017) for further details. The SBM is a random graph model with a planted community structure, and has been thoroughly studied in the fields of statistical physics, high-dimensional probability and theoretical computer science. We will focus on the two-class model. Given a collection V of n elements (assume n is even), we partition them randomly in two equally sized subsets of $n/2$ elements each, giving the hidden variable $y_i = 1$ or $y_i = -1$ accordingly. We then draw an edge between nodes i and j with probability p if both i and j belong to the same set (ie $y_i = y_j$), and with probability q if $y_i \neq y_j$. Edges are drawn independently from each other. The resulting random graph is $G = (V, E)$. The SBM model is therefore parametrised by two numbers $p, q \in [0,1]$. Assume from now on that $p > q$.

The goal of community detection is to infer the hidden community structure y_i, $i \in V$, given G. Several well-known methods exist in the literature to perform this inference, from spectral methods, to semi-definite programs, to belief-propagation. Intuitively, these algorithms are trying to perform a clustering (or *cut*) of the graph into components more connected than average.

In this exercise, we will consider Graph Neural Networks $\Phi(G,\Theta) = \{\Phi_i(G,\Theta); i \in V\}$ parametrised by Θ. The output of the GNN is still localised in the graph, and is used to perform node-wise binary classification. For that purpose, we will first construct a training set $\{(G^{(k)}, y^{(k)}\}_{k \leqslant K}$ of iid instances of the SBM, together with their labels.

1. (7pt) Let $\ell(z', z) = \log(1 + \exp(-z'z))$ be the logistic loss. Given an input graph G, let $\hat{y}_i = \Phi_i(G,\Theta)$, $i \in V$ be the output of the prediction model. Why cannot we use as loss function

$$\tilde{L}(\Theta) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|V^{(k)}|} \sum_{i \in V^{(k)}} \ell(\Phi_i(G^{(k)}, \Theta), y_i^{(k)}) \ ?$$

   Justify your answer.

2. (7pt) Instead, we will use

$$L(\Theta) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|V^{(k)}|} \min \left\{ \sum_{i \in V^{(k)}} \ell(\Phi_i(G^{(k)}, \Theta), y_i^{(k)}), \sum_{i \in V^{(k)}} \ell(\Phi_i(G^{(k)}, \Theta), -y_i^{(k)}) \right\}$$

   Explain what are the differences between L and $\tilde{L}$. We will stick with L from now on.

3. (7pt) Explain why $p \neq q$ is necessary in order to detect the communities.

4. (7pt) For each $s \in \{1, \ldots, 5\}$, generate a training set of SBM instances, choosing $n = 1000$, $p = a_s/n$, $q = b_s/n$, $K = 100$, with $a_s = 12 + s$, $b_s = 8 - s$. Visualize one sample for each s to show the community. [*Hint:* Less connections between community and more connections in the community when s is increasing]

5. (7pt) Using a test set of the same size and characteristics as the train set above, measure the average *overlap* between predicted and true labels. Again, if $\hat{y}_i$ denotes the predicted label of the model for node i, the overlap over a single SBM instance is defined as

$$\text{Overlap} = 2 \left(\frac{1}{|V|} \max \left\{ \sum_i \mathbf{1}[\text{sign}(\hat{y}_i) = y_i], \sum_i \mathbf{1}[\text{sign}(\hat{y}_i) = -y_i] \right\} - \frac{1}{2} \right) .$$

   Verify that $0 \leqslant \text{Overlap} \leqslant 1$. What does it convey? [*Hint:* Think about the expected overlap for a purely random guess, when is the overlap equal to 0 and when is it equal to 1]

6. (8pt) Consider a GCN architecture such as the one from 1.3 (except for the final global pooling step). Choose σ as ReLU. Use instance normalisation (or spatial batch normalisation assuming a batchsize of 1), and use the node degrees $x_i = \sum_j A_{i,j}$ as input features. You can play with the depth of the network (K in our previous notation) as well as the choice of activation. Plot the results as a function of the *Signal-to-Noise* ratio, defined as $\text{SNR} = (a-b)^2/(2(a+b))$. In other words, average overlap on the test set in the y-axis vs SNR in the x-axis.

7. (7pt) Compare this model against a *spectral clustering* baseline. Given the adjacency matrix of a graph A, the spectral clustering computes v_2 as the eigenvector with the second-largest eigenvalue:

$$v_1 = \arg\max_{\|v\|=1} \|Av\| \ , \ v_2 = \arg \max_{\|v\|=1, v^\top v_1 = 0} \|Av\| \ ,$$

   and defines the community estimator as $\hat{y} = \text{sign}(v_2)$. This is also called the *Fiedler* vector. Analyse the results of your GNN model in light of this baseline.

References

Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.