Ramansh Sharma Written Qualifying Exam

Read the questions carefully. Follow the indicated page lengths. Feel free to cite references. You have one week.

1. Let $0 < m < d_1, d_2 < \infty$ be integers. Let $(X_1, Y_1), \ldots, (X_N, Y_N)$ be data points in $\mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$.

   We want to derive a kernelized autoencoder. Given a kernel K on $\mathbb{R}^{d_1}$ with RKHS $\mathcal{H}_K$ and a kernel Γ on $\mathbb{R}^m$ with RKHS $\mathcal{H}_\Gamma$, we seek

   $$g = (g_1, \ldots, g_m), \quad g_i \in \mathcal{H}_K, \qquad f = (f_1, \ldots, f_{d_2}), \quad f_j \in \mathcal{H}_\Gamma,$$

   that minimize

   $$\min_{g_1,\ldots,g_m \in \mathcal{H}_K,\, f_1,\ldots,f_{d_2} \in \mathcal{H}_\Gamma} \sum_{i=1}^{m} \|g_i\|_K^2 + \sum_{j=1}^{d_2} \|f_j\|_\Gamma^2 + \lambda \left\| f \circ g(X) - Y \right\|_2^2, \tag{1}$$

   where $X = (X_1, \ldots, X_N)$, $Y = (Y_1, \ldots, Y_N)$, and $f \circ g(X)$ denotes the vector $(f(g(X_1)), \ldots, f(g(X_N)))$. Reduce (1) to a finite-dimensional optimization problem.
   Recommended answer length: less than a page. AI/LLM not allowed.

   Answer:

   In order to reduce (1) to a finite-dimensional problem, we have to rewrite the functions $(g_1, \ldots, g_m)$ and $(f_1, \ldots, f_{d_2})$ in terms of their respective reproducing kernels. We assume kernels K and Γ are real valued kernels. By definition [1, Chapter 2.3], the real valued functions from the RKHS $\mathcal{H}_K$ and $\mathcal{H}_\Gamma$ can respectively be written as,

   $$g_i(\cdot) = \sum_{p=1}^{N} c_p^i K(\cdot, \mathbf{x}_p), \tag{2}$$

   $$f_j(\cdot) = \sum_{k=1}^{m} d_k^j \Gamma(\cdot, \mathbf{y}_k), \tag{3}$$

   where i and j denote the indices of the functions from the respective RKHS and $\mathbf{x} \in \mathbb{R}^{d_1}, \mathbf{y} \in \mathbb{R}^m$ are arbitrary points in the respective domains of the kernels. Using the

properties of symmetry and positive definiteness of reproducing kernels [1, Chapter 2.3], and the forms of functions that belong to RKHS described above, we have the following,

$$\|g_i\|_K^2 = \langle g_i, g_i \rangle_{\mathcal{H}_K} = \left\langle \sum_{p=1}^{N} c_p^i K(\cdot, \mathbf{x}_p), \sum_{b=1}^{N} c_b^i K(\cdot, \mathbf{x}_b) \right\rangle = (\mathbf{c}^i)^\top \mathbf{K} \mathbf{c}^i, \tag{4}$$

$$\|f_j\|_\Gamma^2 = \langle f_j, f_j \rangle_{\mathcal{H}_\Gamma} = \left\langle \sum_{k=1}^{m} d_k^j \Gamma(\cdot, \mathbf{y}_k), \sum_{t=1}^{m} d_t^j \Gamma(\cdot, \mathbf{y}_t) \right\rangle = (\mathbf{d}^j)^\top \mathbf{\Gamma} \mathbf{d}^j, \tag{5}$$

where, $\mathbf{K}$ and $\mathbf{\Gamma}$ are the full $(N \times N)$ and $(m \times m)$ kernel matrices respectively. The goal now is to rewrite (1) as an optimization problem in terms of the expansion coefficients $\mathbf{c}$ and $\mathbf{d}$. We reformulate (1) below,

$$\min_{\mathbf{c}^1,\ldots,\mathbf{c}^m,\, \mathbf{d}^1,\ldots,\mathbf{d}^{d_2}} \sum_{i=1}^{m} (\mathbf{c}^i)^\top \mathbf{K} \mathbf{c}^i + \sum_{j=1}^{d_2} (\mathbf{d}^j)^\top \mathbf{\Gamma} \mathbf{d}^j + \lambda \sum_{n=1}^{N} \left(\begin{bmatrix} f_1(g(X_n)) \\ f_2(g(X_n)) \\ \cdots \\ f_{d_2}(g(X_n)) \end{bmatrix} - Y_n \right)^2, \tag{6}$$

$$\min_{\mathbf{c}^1,\ldots,\mathbf{c}^m,\, \mathbf{d}^1,\ldots,\mathbf{d}^{d_2}} \sum_{i=1}^{m} (\mathbf{c}^i)^\top \mathbf{K} \mathbf{c}^i + \sum_{j=1}^{d_2} (\mathbf{d}^j)^\top \mathbf{\Gamma} \mathbf{d}^j + \lambda \sum_{n=1}^{N} \sum_{j=1}^{d_2} \left(f_j(g(X_n)) - Y_n^j \right)^2. \tag{7}$$

For brevity, we use $\mathbf{g}_n$ to denote $g(X_n) = (\mathbf{K}(X_n, X)\mathbf{c}^1, \ldots, \mathbf{K}(X_n, X)\mathbf{c}^m)$, where $\mathbf{K}(X_n, X)$ is a column vector. Then, (7) can be written as,

$$\min_{\mathbf{c}^1,\ldots,\mathbf{c}^m,\, \mathbf{d}^1,\ldots,\mathbf{d}^{d_2}} \sum_{i=1}^{m} (\mathbf{c}^i)^\top \mathbf{K} \mathbf{c}^i + \sum_{j=1}^{d_2} (\mathbf{d}^j)^\top \mathbf{\Gamma} \mathbf{d}^j + \lambda \sum_{n=1}^{N} \sum_{j=1}^{d_2} \left(\mathbf{\Gamma}(\mathbf{g}_n, \mathbf{g}_j) \mathbf{d}_j - Y_n^j \right)^2. \tag{8}$$

In the final form shown in (8), the optimization problem is with respect to the finite-dimensional expansion coefficients $(\mathbf{c}^1, \ldots, \mathbf{c}^m)$ and $(\mathbf{d}^1, \ldots, \mathbf{d}^{d_2})$.

2. Consider the setting of operator learning, with an oracle/exact operator G:

$$A \ni a \overset{G}{\mapsto} u \in U, \quad G \in B(A; U), \;\; A = A(D_a; \mathbb{R}^a), \;\; U = U(D_u; \mathbb{R}^u),$$

where A and U are function spaces (i.e., appropriate measure spaces, such as Hilbert or Banach spaces), with $a, u \in \mathcal{N}$. E.g., A is a space of functions mapping some domain D_a to $\mathbb{R}^a$, where D_a is a subset of a Euclidean space. We likewise assume B is some measurable space of mappings (operators) that map elements in A to elements in U. We consider the *supervised* learning setting, where G is approximated through (possibly noisy) input-output pairs.

Informally, a *neural operator* is a model class mapping class A to U that is constructed through iterative applications of affine global operators with componentwise/local nonlinear ("activation") operators. Neural operator model classes with finite encodings are

of particular interest, as they represent a feasible space of computable maps. The concept of *universal approximation* is popular to investigate in neural operators, with the goal to establish that, e.g., given an arbitrary G and tolerance ϵ, there is a finite-encoding neural operator that is ϵ-close to G.

Summarize the current state of universal approximation results/theorems in the literature, paying particular attention to pros and cons of these results (either individually or collectively). In particular, describe how (or if) one can use these universal approximation statements to guide computational construction of architectures. Propose a strategy to use some tools from existing methods for approximating functions over finite-dimensional spaces that might be used to provide more constructive and quantitatively rigorous methods for constructing neural operators.

In your response, consider the following aspects:

1. Provide a reasonable high-level survey of current universal approximation results. Theoretical precision is appreciated, but the main goal is to summarize overall strengths and weaknesses of approximability statements. It's not important to cite every single result on universal approximation for neural operators, but provide enough breadth of narrative to cover "much" of the literature.
2. Identify the practical utility of universal approximation results in the computational construction of neural operators. I.e., how would you use these results to actually construct computational architectures? (Are these results useful for that purpose?)
3. Describe or propose how one might leverage existing quantitative results on approximation of functions to augment the current neural operator approximation theory. (Here, "quantitative" refers to actual, fairly precise rates of approximability or convergence for finite-dimensional approximation of functions.) In particular, how might one attempt to port quantitative results from function approximation to operators? What challenges require new investigations to address? It's perhaps most useful to narrow this discussion to leveraging a particular class/type of function approximation results, rather than attempting to broadly consider existing results on function approximation.

Recommended answer length: 3-5 pages. AI/LLM not allowed.

3. **I will ask two questions both related to your work. Answer one in detail (1 page) and one in short (1/2 page) (your choice).**
   (a) One of the challenges as I understand in SciML applications is that your error requirements are much more stringent than those in ML applications. For certain kinds of ML models, there are so-called "scaling laws" (e.g., `https://arxiv.org/pdf/2001.08361`). Go over the paper above, summarizing the main results. Would you expect analogous results for SciML applications (e.g., learning very simple PDEs' solutions), and if so, do you expect to drive the error to an arbitrarily small

quantity? This is an open ended question, so please include references to existing work.

(b) Study one of the early papers on "in context" learning, specifically: `https://arxiv.org/pdf/2208.01066`. While SciML papers do not think about what they do as ICL, expecting a model to learn a solution to a "new" PDE is quite similar in spirit. Go over the paper above and summarize the main results. There have been many follow up works to the paper above claiming that the distributions are important, etc., but one interesting paper is: `https://arxiv.org/pdf/2306.09927`. *Skim* the main results, but especially look at Section 4.2. Are you aware of analogous results in the SciML area? Does the distribution over boundary conditions, etc. matter for learnability?

4. (a) Please introduce and summarize existing operator learning methods. Please give some categories, and summarize pros and cons for each category.
Recommendation: no less than 2 pages.

(b) What do you think about the future of operator learning? You can explain whatever thoughts you have, positive, negative, and future development direction, etc.
Recommendation: no less than 1 page.

5. Consider partition of unity (PoU) kernel methods in the context of divergence-free approximation. These methods allow one to scale global interpolation to very large numbers of points without a loss of computational efficiency. However, there are some specific challenges to applying PoU methods to divergence-free approximations. **Note that we are not talking about operator learning, but just function approximation/interpolation**.
No AI/LLM allowed.

(a) It is quite straightforward to use a global divergence-free kernel for interpolation. However, in the PoU context, this is not straightforward at all. *Show why* mathematically.
Recommendation: No more than 1/2 a page.

(b) Drake, Fuselier, and Wright present an approach in `https://arxiv.org/abs/2010.15898` that works for 2D approximations. Summarize their approach. What is the *primary mathematical* difficulty in applying their technique to 3D problems? Hint:It relates to the nature of the div-free potentials in 3D, Eq 2.10.
Recommendation: No more than 1 page.

(c) How would you use machine learning (ML) to overcome these difficulties? Derive a new ML-based PoU technique to do so that is also divergence-free by construction; describe the ML architecture, point out how it overcomes the issue with the Drake-Fuselier-Wright method, discuss training procedures and difficulties. Briefly connect it to your answer in part 1. Assume you already have a div-free approximant/interpolant on each patch.
Recommendation: 1-2 pages.

References

[1] Gregory E Fasshauer and Michael J McCourt. *Kernel-based approximation methods using Matlab*, volume 19. World Scientific Publishing Company, 2015.