Pre-Final Exam, Math Modeling 2021

I'm going to outline the final exam problems below, except that I'm not going to give you the actual data. You should work out how you're going to solve each problem, even possibly going as far as creating your own "toy data" to work with. **On this part of the exam, I am encouraging everyone to work in groups.** If you don't have a group partner in mind, tell me and we'll try to form at least one partner for each person- you can work in groups of up to three each.

On Friday, I will ask you to **stop all group conversations**. I will upload the data for you to download, then I'll ask you to finish the questions on your own with no more help. You'll then upload a write up of your conclusions (much like the last labs we did).

Timeline in Summary:

- Mon: Publish the problems.
- Mon-Thur: You may work in groups and work out templates and examples- Do whatever you need to do!
- Friday: Stop all group work. The data will be loaded into Canvas for you to download. You'll be given until Sunday evening to finish the problems.

  NOTE: If you prep well, then the Friday session should be really short- You'll just run the new data and summarize your conclusions!
- Sunday evening: Deadline to turn in your work.

Continued on Next Page...

Final Exam Questions

1. Problem 1: Build a k-Nearest Neighbor classifier on some data.

   The data X will be about 4,000 points in $\mathbb{R}^{22}$, and the targets will be two classes, so it will be in $\mathbb{R}^2$. You'll try two different k values- One equal to 2 and one equal to 10, and you'll compare the confusion matrices.

2. Problem 2: Build an RBF classifier.

   Same data as Problem 1, but you'll use $k-$means to set a given number of centers, then construct the RBF equations and solve it. Finally, create the confusion matrix and present the results.

   Compare this with results if you choose the centers at random.

   Remember the issue that you might have seen in the homework- it had to do with scaling the RBFs- The "correct" scaling can be very important!

3. Problem 3: Neural Differential Equations

   Suppose $\mathbf{x}'(t) = \mathbf{F}(\mathbf{x})$.

   If we don't have a formula for F, we can replace the ODE with $\mathbf{x}'(t) = \texttt{NeuralNet}(x)$, then run the ODE solver using `NeuralNet` instead of $\mathbf{F}$.

   I'll give you data that represents data from the unknown vector field F (input: 3-d, output 3-d), and will have you train a $3-20-3$ neural network (not using anything built-in, but using our script file) on this data. The neural network thus provides the derivative for the ODE solver.

   Once trained, run the built-in ODE solver `ode45` (in Matlab) or `rk45` (in Python) to get a numerical solution to the ODE for $0 \le t \le 20$, then plot the solution curve in 3-d. To prepare for this, be sure you can call the ODE solver on a 3-d ODE (make one up to check).

   A bit of a wrinkle to think about: We'll need to build a function file for our neural net; meaning we need to tell the function what the weights and biases are. If you're good with programming, feel free to use global variables. If you're not, another option is to "hard code" the weights and biases into the function by having Matlab or Python print out the weights and biases, then copy/paste the numerical values directly into a function file.