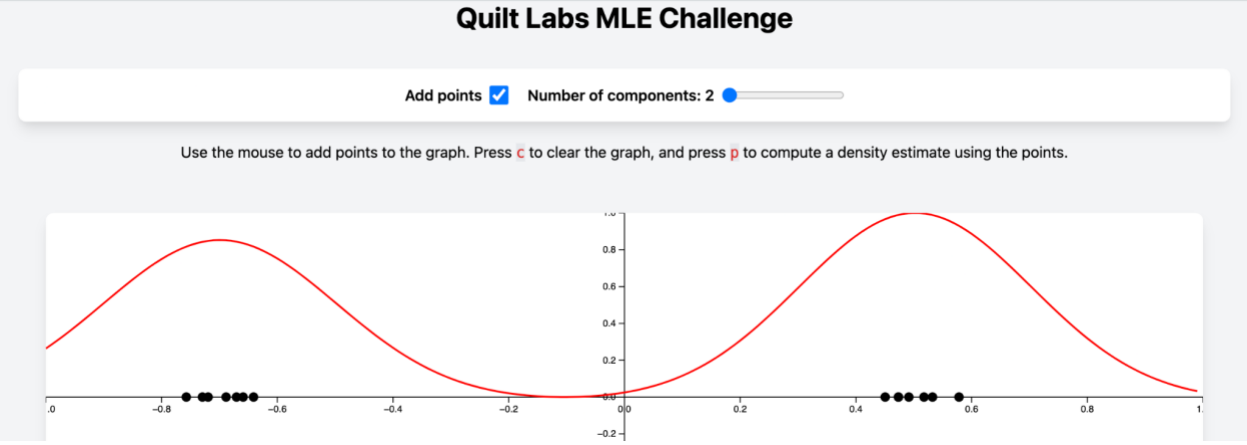
QUILTLABS

Challenge: Interactive ML model training July 2023

Goal

In this challenge, you will create an interactive human-in-the-loop training system using a simple machine learning model.



1. You are given parts of a web application and must implement the backend for this system
2. The frontend (shown above)
    a. Allows users to interactively add data points to a graph
    b. Sends data points to the backend
    c. Receives model parameters from the backend and plots a density over the points
3. The backend must
    a. Accept these data points and train the model
    b. Return the parameters of the model to the front-end

Detailed requirements

1. Read through the provided code
2. Implement a backend API service that hosts the model, trains it, and returns model parameters
3. When `p` is pressed, you need to retrieve the parameters of the model and pass them to the frontend for plotting
4. Implement a logging solution for the model so that you can keep track of:
    a. training loss
    b. model versions
    c. data used to train each model
5. Implement a solution to visualize each training result
6. Submit a README.txt containing details about the methods you implemented, problems you ran into, and how to improve your submission if you had more time

Provided code

1. Web application:
    a. **solution_skeleton.py**: web application. Root endpoint takes the user to the app shown above. You'll need to implement any other endpoints in this file

   b. **index.html**: static HTML page with layouts for visual elements of the app
   c. **static/script.js**: Contains plotting code for the web app
2. Model training code:
   a. **gmm.py**: contains Pytorch model class, function to train the model, function to extract parameters from the model and return them as lists of floats

Judgment criteria

1. Does your web app work? For your submission to be considered complete, it must run on our Ubuntu 20.04 VM. If you choose to include additional requirements, please be sure to mention them in your README.txt
2. How clean is your code? Do you use comments? Type annotations?
3. Do you discuss testing in README.txt? How would you think to test an application like this?
4. How did you implement logging and model versioning? How do you store metadata associated with a trained model?
5. How easy to understand is your visualization of training results?
6. How memory-efficient is your code? What is the runtime of a training run?

FAQs

- Which files can I edit? Anything in the folder can be edited
- Can I use other packages? Yes, just list them in your README.txt so we can install them before judging
- Do I need to worry about cleaning up visualizations? Not really, as long as they're easy to understand
- How should I submit the code? Please zip your submission and upload to Google Drive, then share a public link with us via email