Project Title

Optimizing Multi-Layer Perceptron Hyperparameters Using Bayesian Optimization

Group Members

- Abhishek Satpathy
- Gaetano Scafidi
- Mark Do

Experimental Set-up

The goal of our experiment is to efficiently optimize the hyperparameters of a Multi-Layer Perceptron (MLP) neural network. Traditional grid search methods are computationally expensive and impractical for large search spaces. We hypothesize that Bayesian Optimization can significantly reduce the time and computational resources required to find optimal hyperparameters.

Methods

- **Datasets**:
  - We utilized a publicly available dataset to train and test our MLP model. The dataset was split into training and testing subsets using the `train_test_split` function from `scikit-learn` to ensure the model's performance could be accurately evaluated. We cleaned up and loaded the dataset using the `PIL` and `pandas` libraries.
- **Algorithms**:
  - **Multi-Layer Perceptron (MLP)**: We employed the `MLPClassifier` from `scikit-learn` to construct our neural network model. The MLP's architecture included two hidden layers, with the number of neurons and activation functions as hyperparameters to be optimized.
  - **Gaussian Process Model**: The core of Bayesian Optimization is the Gaussian Process (GP) model, implemented using `GaussianProcessRegressor` from `scikit-learn`. This model serves as a surrogate to approximate the objective function, which in this case is the accuracy of the MLP model.
  - **Bayesian Optimization Framework**: In terms of the actual optimization process, we built an algorithm from the ground up starting from creating our own acquisition functions, and building out a simple process that iteratively improves the estimations of the surrogate model around predicted optimums.
- **Implementation Details for Our Demo**:
  - **Hyperparameter Space**: We defined a search space that included learning rates, number of neurons per hidden layer, and activation functions. Specifically, we considered:
    - Learning rates: A range of values between 0.0001 and 0.1.
    - Neurons per hidden layer: Values between 10 and 500 for each layer.
    - Activation functions: Options including 'relu', 'tanh', 'logistic', and 'identity'.
  - **Optimization Process**:
    - The Gaussian Process was initialized with a small number of random samples from the hyperparameter space.
    - The acquisition function was used iteratively to select new hyperparameter sets that were likely to improve the model's performance.
    - Each selected set was evaluated by training the MLP and recording its accuracy.
    - The GP model was updated with new data points after each iteration, refining its prediction of the objective function.

Results

The experiment demonstrated that Bayesian Optimization significantly reduced the search time compared to traditional grid search methods. We approached a semi-optimal solution in computational time < 2 hours, whereas a grid search method would've taken up to three years on the same device.

Learning Objectives

From this project, we aim for the class to learn:

- The fundamental principles of Bayesian Optimization and its application in machine learning.
- How Bayesian Optimization can practically be used to efficiently navigate large hyperparameter spaces.
- The practical advantages of using probabilistic models like Gaussian Processes in optimizing complex functions.