Assignment: Homework 5

How to Hand It In

1. Put all your solutions in one folder. Compress this folder and name it <firstname>_<lastname>_HW5.zip. For example, “Zhiyao_Duan_HW5.zip”.
2. Make sure you have a report to show graphs and explain your answers, in addition to source code.
3. Submit to the corresponding entry on Blackboard.

When to Hand It In

It is due at 11:59 PM on the date specified on the course calendar. **Late assignments will receive a 20% deduction of the full grade each day.**

Problems (10 points in total)

1. (6 points) Implement a multi-layer perceptron (MLP) classifier and the backpropagation algorithm to train it on the "Maternal Health Risk Data" dataset. This dataset was collected from medical institutions to identify health conditions that pose the greatest risks during pregnancy. The dataset can be accessed at https://www.kaggle.com/datasets/drmbsharma/maternal-health-risk-data-set.
   Note: Skeleton code is provided which includes data preprocessing and the basic structure of the code. You need to fill the TODOs in the skeleton code.
   a. (1 point) Implement the forward pass of a two-layer MLP (i.e., with a single hidden layer) classifier that uses sigmoid activation for both the hidden layer and output layer. The output layer has a single node. - Randomly initialize weights and biases in the range of [-1, 1).
   b. (2 points) Implement the backpropagation algorithm using squared error loss.
   c. (1.5 points) Implement a training function. Train your model on the training set with default hyperparameters provided in the skeleton code. Use the entire training set as a batch, i.e., computing the gradient on the entire dataset in each iteration. Plot the error versus the number of epochs and report classification accuracies on subsets.
   Note: When we have a very large dataset, it would be computationally expensive to compute the gradient on the entire dataset in each iteration, and people often compute a stochastic gradient using a small subset (called a batch) in each iteration. You are welcome to try it, but there is no extra credit for this exploration.
   d. (1.5 points) Experiment with hyperparameters such as number of iterations, learning rate, and hidden layer size to evaluate how accuracy is affected. Use validation data to identify the best set of hyperparameters. If you are not confident about your own implementation, you may use sklearn.neural_network.MLPClassifier instead for this

question. Record the changes made, the reasoning behind the changes, and the results in your report.

2. (3.5 points) Train a two-layer Hopfield network with equilibrium propagation on the same "Maternal Health Risk Data" dataset.
   a. (1 point) Implement the Equilibrium Propagation (EP) Update Rule. The continuous network is defined by the following energy function:

$$\mathrm{E(s)} \;=\; 1/2\;\textstyle\sum_{\mathrm{i}} s_i^2 \;\; - \;\; \sum_{\mathrm{i>j}} W_{ij}\,\rho(s_i)\rho\big(s_j\big) \;\; - \;\; \sum_{\mathrm{i}} b_i\,\rho(s_i),$$

   where s denotes the state vector, W is the symmetric weight matrix, b is the bias vector, and $\rho(\cdot)$ is the activation function.

   In Equilibrium Propagation, parameters are updated using the contrast between the nudged equilibrium (β) and the free equilibrium (0). The gradient-based parameter updates take the form

$$W_{ij} \leftarrow W_{ij} + (\eta/\beta)\left(\rho\left(s_i^{\beta}\right)\rho\left(s_j^{\beta}\right) - \rho(s_i^{0})\rho\left(s_j^{0}\right)\right),$$

$$b_i \leftarrow b_i + (\eta/\beta)\left(\rho\left(s_i^{\beta}\right) - \rho(s_i^{0})\right),$$

   where η is the learning rate and β is the nudging strength.

   Complete the TODOs in update() to implement the EP learning procedure by running the free and nudged phases, computing the contrast between equilibria, and updating all model parameters consistently across layers according to the expressions above.

   b. (0.5 points) Implement the Prediction Function. Complete the TODO in predict() so that the model runs in the free phase and returns class predictions for given inputs.
   c. (1 point) Implement the Training Procedure. Complete the TODO in fit() to train the model over multiple epochs using mini-batches. The procedure should iteratively update model parameters and track loss and accuracy throughout training. Include a plot showing how training loss and accuracy evolve across epochs.
   d. (1 point) Experiment with hyperparameters such as number of epochs, learning rate, beta, steps, dt and hidden layer size to evaluate how accuracy is affected. Use validation data to identify the best set of hyperparameters.

3. (0.5 points) Compare the results of the two approaches you implemented. Try to explain the performance differences. Discuss the pros and cons of each approach, on accuracy, training efficiency, inference efficiency, power consumption, hardware implementation, etc.