Training Setup

In the training setup, various hyperparameters were considered to train and evaluate these neural network model. These hyperparameters include:

Number of Hidden Layers: 1 or 2 hidden layers.

Number of Neurons: 10 or 30 neurons in the hidden layers.

Learning Rates: 0.1, 0.01, 0.001 or 0.0001.

Activation Functions: ReLU, Tanh or sigmoid.

Epochs: 50.

Batch Sizes: 32.

Training Procedure

A multi-layer perceptron (MLP) was created using PyTorch, using hyperparameters, such as the number of hidden layers and neurons in each layer. The model was optimized using Adam optimizer also the loss function chosen was Cross-Entropy. As instructed the training and validation datasets were combined, creating a larger, dataset for training, enhancing the model's learning capability and generalization. Early stopping mechanism was used to mitigate the overfitting and for testing the model a separate test dataset was used to assess how well the model generalized to new, unseen data.

Accuracy Performance Results

Best Hyperparameters: number of hidden layers: 1, num of neurons: 30, learning rate': 0.01, activation: tanh, epochs: 50, batch size: 32.

Test Accuracy Confidence Interval: 0.12 ± 89.27

Questions

1. What type of measure or measures have you considered to prevent overfitting?

   Early stopping mechanism was used to mitigate the overfitting.

2. How could one understand that a model being trained starts to overfit?

   By checking the validation performance over a certain number of epochs

3. Could we get rid of the search over the number of iterations (epochs) hyperparameter by setting it to a relatively high value and doing some additional work? What may this additional work be?

   Yes we can but we will have to use early stopping so the model doesn’t overfit mechanism and other techniques too

4. Is there a ”best” learning rate value that outperforms the other tested learning values in all hyperparameter configurations?

   Best learning rate = 0.01

5. Is there a ”best” activation function that outperforms the other tested activation functions in all hyperparameter configurations?

   There isn’t any that outperformed the others. Tanh and ReLU were competing for best place.

6. What are the advantages and disadvantages of using a small learning rate?

   Small learning can get stuck in local minima. Training with a small learning rate takes more time to converge.

   Small learning rates can improve the model ability to generalize to new data.

7. What are the advantages and disadvantages of using a big learning rate?

   Big learning rate converges faster so saving time. Less prone to get stuck in local minima.

   Big learning rate may make it hard for model to achieve convergence. Models trained with large learning rates are prone to overfitting and can also miss the global minimum.

8. Is it a good idea to use stochastic gradient descent learning with a very large dataset? What kind of problem or problems do you think could emerge?

   It is a good idea to use stochastic gradient descent for very large dataset. However, tuning hyperparameters for SGD such as learning rate and mini-batch size will be more challenging.

9. In the given source code, the instance features are divided by 255 (Please recall that in a gray scale-image pixel values range between 0 and 255). Why may such an operation be necessary? What would happen if we did not perform this operation?

   We divide it by 255 to normalize the input features specifically for activation functions whose gradient might go to zero on very large data values. This problem is called vanishing gradient problem.