Programming Practice III

Yiwen Liao
Institute of Signal Processing and System Theory

1 Overview

In this practice, you study the common hyperparameters of the neural network implemented in Programming Practice II. In particular, you are expected to

- compare the training procedure and the performance of a neural network with different initialization methods, learning rates, capacities and regularization decays;
- learn to implement the techniques mentioned above from scratch with simple examples.

2 Tasks

In this section, you can start the programming practice task by task. Please pay attention to the hints as well as the API documents available online.

2.1 Define Neural Network

Please convert the neural network implemented in the Programming Practice II into a Python class `NeuralNetwork`. The `NeuralNetwork` class has an initialization method with three parameters: `parameters`, `learning_rate` and `step`. `parameters` is a container consisting of `w1, b1, w2, b2` as the weights and bias for this one-hidden-layer network. The parameter `step` denotes how often you print the training loss during the entire training procedure.

In addition, the `NeuralNetwork` class should have an attribute `losses` to store the training loss for each epoch. `NeuralNetwork` has a function `train()` that takes `x`, `y` and `epochs` as the parameters. Finally, this class has a `predict()` function that takes `x` as a parameter to obtain predictions.

In this practice, we use ReLU as the activation function. Therefore, you should also implement the ReLU function with its derivative as `relu(x)` and `derivative_relu(x)`, respectively. Hint:

- the major part of `train()` is similar to the implemented network in the last practice;
- refer to the last practice when you implement `relu(x)` and `derivative_relu(x)`;

Manuscript updated on 16.03.2021

2.2 Parameter Initialization

Please write a function `initialize_parameters(method, num_neurons)` that returns the initialized network parameters `w1, b1, w2, b2`. The parameter `method` can take the string values of either `"normal"` or `"uniform"`. The parameter `num_neurons` denotes the number of the hidden neurons (capacity). Bias should be initialized with zeros for simplicity. In this task, the normal distribution is defined as $\mathcal{N}(0, 0.1^2)$ and the uniform distribution is defined as $U[-0.1, 0.1]$.

Please build and train your neural network with the following setups: the number of hidden neurons equals 500, learning rate is 0.01 and the number of training epochs is 5000. You can define your own `step` for tracking the training procedure. With the same settings, please train networks with normal initialization and uniform initialization respectively.

Please plot the two resulting predicted curves in one figure. In addition, please plot the training losses of both methods in another figure, where the y-axis should be limited to $[0, 0.1]$. Hint:

- consider using `plt.ylim()` to limit the range of axis in figures;
- you may also try normal/uniform distributions with different parameters;
- how does the initialization affect the performance and the training procedure?

2.3 Learning rate

Please train the neural network with the following learning rates: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. The other hyperparameters should maintain same: the number of hidden neurons equals 500, initialization method is set to `"normal"`, the number of training epochs is 5000.

Please plot the resulting five training losses in one figure with necessary labels, where the legend should be located on the right of the figure. Furthermore, please plot the ground truth and the predicted curves of different learning rates in one separate figure. Hint:

- consider using `bbox_to_anchor` and `loc` in `plt.legend()` to customize the desired location of the legends;
- how does the learning rate affect the training procedure?
- you may also increase the training epochs to study when the loss will converge.

2.4 Network Capacity

Please train the neural networks with the following capacities (the number of hidden neurons): 5, 50, 100, 500, 1000. The other hyperparameters should maintain same: the learning rate is 0.01, the number of training epochs is 5000 and the initialization method is `"normal"`.

Please plot the five training losses in one figure with necessary labels. In addition, please plot the ground truth and the predicted curves of different learning rates in one figure. Hint:

- how does the model capacity affect the performance and the training procedure?

2.5 Regularization

In this task, we only consider ℓ_2-regularization for simplicity. In order to minimize the modification cost on the existing codes, you can simply add a new parameter `decay` to the function `train()` of the class `NeuralNetwork`. Subsequently, you can slightly modify the gradient calculation within the `train()` function to enable ℓ_2-regularization.

Please train neural networks with the following decays: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. The other hyperparameters should maintain same: the number of hidden neurons equals 500, the initialization method is set to `"normal"`, the number of epochs is 5000 and the learning rate is 0.01.

After training, please plot the five training losses in one figure with necessary labels. In addition, please plot the ground truth and the predicted curves of different learning rates in one figure. Hint:

- how is ℓ_2-regularization defined?
- what is the gradient of the regularization term?
- how does the regularization affect the performance of the network?