Programming Practice II

Yiwen Liao
Institute of Signal Processing and System Theory

1 Overview

In this practice, you are asked to implement a neural network with one hidden layer for regression using NumPy only. In particular, you are expected to learn to

- implement simple activation functions;
- get familiar with gradient descent algorithms;
- get familiar with back-propagation algorithms in neural networks.

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 Sigmoid Function

Please write a function `sigmoid(x)` that returns the result of applying $\sigma(x) = \frac{1}{1+e^{-x}}$ to an x, where the parameter is `x` which can be scalar/vector/matrix/tensor. Hint:

- consider using `np.exp(x)`;
- $\sigma(\cdot)$ is element-wisely applied to a vector/matrix/tensor.

2.2 Define Derivative of $\sigma(\cdot)$

Please write a function `derivative_sigmoid(x)` that returns $\sigma'(x)$ of a given x, where the parameter `x` can be scalar/vector/matrix/tensor. Hint:

- consider first calculating $\sigma'(x)$ w.r.t. x by hand.

Manuscript updated on 16.03.2021

2.3 Plot Functions

Please plot the sigmoid function and its derivative for $x \in [-5, 5]$ in a 5×5 figure with necessary labels. Hint:

- refer to the plotting exercise in Programming Practice I.

2.4 Define Neural Network

Please define a neural network with one hidden layer as follows: $\hat{y}_i = \boldsymbol{w}_2^\top \cdot \sigma(\boldsymbol{w}_1 \cdot x_i + \boldsymbol{b}_1) + b_2$, where $\hat{y}_i$, x_i and b_2 are scalars while $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are column vectors. The network should be defined as a function `forward_pass(x, w1, b1, w2, b2)` that returns $\hat{y}_i$ of a given x_i. Hint:

- the parameter `x` can be a batch of dataset with the shape of `number_of_samples` $\times$ `number_of_dimensions`;
- you do not have to define the number of neurons at this stage.

2.5 Define Loss Function

Please define the mean squared error (MSE) loss as a function `mse(y_true, y_pred)` and its derivative w.r.t. `y_pred` with name of `derivative_mse(y_true, y_pred)`. Hint:

- `y_pred` refers to the output of a network and `y_true` refers to the ground truth;
- recall the definition of MSE;
- pay attention to the derivative of an MSE loss.

2.6 Define Gradient

Please write a function `cal_gradient(x, y, y_hat, w1, b1, w2, b2)` that returns the gradients w.r.t. the four learnable parameters: $\boldsymbol{w}_1$, $\boldsymbol{b}_1$, $\boldsymbol{w}_2$ and b_2. Hint:

- y refers to the ground truth and `y_hat` refers to the output of a neural network;
- recall the chain rule in calculating gradient;
- first calculate all derivatives by hand and transform your results into codes;
- the shape of weights and bias might be different with mathematical formulas due to the broadcasting property of NumPy.

2.7 Update Parameters

Please define a function `update_parameters(parameters, gradient, learning_rate)` that returns the updated parameters by applying the standard gradient descent once. Hint:

- recall the definition of standard gradient descent algorithm *without* momentum and decays.

2.8 Train Network

Please train your network using the functions implemented above using the 1D dataset from the Programming Practice I. The learning rate is 0.01, the number of hidden neurons is 500, and the number of training epochs are 5000. The initial weights are randomly sampled from a standard normal distribution with seed of 42, while the bias is initialized with zeros. During training, keep track with the loss for every epoch. Print the current epoch number and loss as well as keeping track of $\hat{y}_i$ every 500 epochs. Hint:

- calculate the gradient over the entire dataset for simplicity;
- pay attention to the broadcasting property of NumPy.

2.9 Plot Results

Please plot the predicted curves in comparison with the ground truth curves in a 5×5 figure with necessary legends. Please plot the losses in a separate 5×5 figure. Hint:

- are the ground truth curves perfectly fitted?
- what could be the reasons for the failures (if any)?