CSE641 - Deep Learning
Assignment 2 - Part 2
ReadMe

Submitted By:
Chirag Chawla(MT19089)
Kashish Narang(MT19026)
Saumya Jain(MT19098)

Preprocessing :
For both the questions in this part, the preprocessing steps followed in part 1 are automatically applied. Also, the best setting of defined parameters and architecture found in Part 1 for this problem has been used.

Question 1 Part 2:

In this part, we have to implement the two weight initialization techniques :

1. He
2. Xavier

Methodology :

For this question, we have implemented the following methods.

- **initializeWeights() :** It is a helper function which calls the respective initialization technique based function based on the parameter provided.

- **he() :** In this function, random initialization of weights is equipped with the following term in order to approximate the variance in all layers output close to 1. This is particularly useful with activation functions where activation is non-differentiable at x=0 (Relu).

$$\text{He initialization:} \quad w = rand()\sqrt{\frac{2}{n}}$$

- **xavier() :** Similar to the above method, this function implements the xavier initialization which is nothing but multiplication of random initialization with sqrt of 1/n in order to get output variance of 1.

$$\text{Xavier initialization:} \quad w = rand()\frac{1}{\sqrt{n}}$$

Question 2 Part 2:

In this part, we have to implement the following regularization techniques :

1. L1 Regularization
2. L2 Regularization
3. Dropouts

Methodology :

For this question, we have implemented the following methods.

- **regularize() :** A helper function which calls the respective regularization technique as defined in the toolkit constructor.

- **l1loss() :** L1 or lasso regularization technique adds “absolute value of weights" as penalty term in the loss function. The amount of regularization is controlled by the ‘lambda’ hyperparameter.

$$Loss(W) = \frac{1}{N}\sum_{i}^{N} div(y_i, d_i) \quad + \quad \frac{1}{2}\lambda ||W||$$

- **l2loss() :** L2 or ridge regularization technique adds “squared value of weights" as penalty term in the loss function. The amount of regularization is controlled by the ‘lambda’ hyperparameter.

$$Loss(W) = \frac{1}{N}\sum_{i}^{N} div(y_i, d_i) \quad + \quad \frac{1}{2}\lambda ||W||_2$$

- **drop_out_forward() :** In this function, we implemented the drop out technique for regularization. The first step was to create one hot coded vector representing the on/off state of each neuron based on the dropout threshold provided. In the next step, the outputs of the first layer are thus multiplied by the above vector to pass outputs of allowed neurons to the next layer.

$$r_i^{(1)} = 0, \quad \text{rand}() < d_p^{(1)};$$
$$= 1, \quad \text{rand}() \geq d_p^{(1)};$$

$$\hat{y}_i^{(1)} = r \,.\, y_i^{(1)}$$

$$z_1^{(2)} = \hat{y}_1^{(1)} \,.\, w_{11}^{(2)} + \hat{y}_2^{(1)} \,.\, w_{21}^{(2)}$$

- **drop_out_forward_test() :** For testing in dropout method, it is simply the multiplication of output of the layer with the dropout value.

$$\hat{y}_i^{(1)} = y_i^{(1)} * d_p^{(1)}$$

- **Other function Changes :** In order to implement L2 regularization, we need to consider changing the gradient function where weights are added to the gradient of W1 and W2.

$$\frac{\partial Loss}{\partial W} = \frac{\partial div(y_i, d_i)}{\partial W} \quad + \quad \lambda W$$

In order to implement L1 regularization, we need to consider changing the gradient function where sign of the weights is used in the gradient of W1 and W2.

$$\frac{\partial Loss}{\partial W} = \frac{\partial div(y_i, d_i)}{\partial W} \quad + \quad \lambda$$