DEEP LEARNING MINI PROJECT - THEORY

Dody Harianto

1. What may happen if you set the momentum too close to 1? (Example: 0.99999)

Momentum that approaches 1 will lead to overshooting (far away from optima), which decreases the convergence quality because there will be more oscillations around the optima.

2. Does drop out slows down training? What about MC dropouts?

Yes. Based on my experiment in the notebook, adding dropout slows down the training process by around 4 seconds per epoch. My assumption is because the model requires more time to drop some fraction of the neurons in each epoch of training.

3. State 3 advantages of SELU over RELU!

- SELU enables self-normalization so we do not have to include batch normalization explicitly. Meanwhile RELU does not support self-normalization since it cannot output anything if the input is negative.

- SELU reduces the possibility to get the vanishing gradient problem in RELU.
- SELU's convergence speed is better over RELU regardless the choice of hyperparameters.

4. Specify each cases when to use SELU, Leaky RELU and variants, RELU, Tanh, Logistic, and Softmax!

- SELU

SELU is used to train a deep neural network because it can gain a better test accuracy.

- RELU

RELU is one of the common practices when we build a deep neural network. We use RELU activation function when computational efficiency (fast convergence) is needed.

- Leaky RELU

We usually use Leaky RELU to prevent the common problem when using RELU, which is dying RELU (the situation where the gradient is zero if input is negative), so it

- Tanh

Tanh activation function is commonly used for binary classification. Tanh also has better convergence speed rather than sigmoid. This is one of the best activation function for Recurrent Neural Network (RNN).

- Softmax

This activation function is used when we want to build a multiclass classification model and get the probability distributions. Softmax is often used in the output layer.

- Logistic / Sigmoid

We use sigmoid activation function if we want to get the probability of a predicted value. This function takes a predicted value and converts it into probabilistic value that ranges from 0 to 1. However, sigmoid activation function performs poor if the network is too deep.

5. Is it ok to initialize the bias terms to 0?

By default, Keras also initialize the bias terms to zero. We can also pass a parameter **bias_initializer** and set it to **'zeros'** before training our model.

(it will look something like this)

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.], dtype=float32),
```

So, it is ok to set the bias to zero. Because when we build a neural network and our model is training, the weights and biases of all neurons will be updated during training.

```
array([-6.5095727e-03, -3.1065221e-03, -3.2026023e-03, -1.2197275e-02,
       -1.2458657e-02, -1.0188344e-02, -2.5508574e-03, -1.1791829e-03,
       -6.3047023e-03, -3.5102661e-03, -2.2031055e-03, -2.3150812e-03,
       -5.0387904e-03, -2.3149459e-03, -4.0483675e-03, -2.3150784e-03,
       -2.3726353e-03, -2.6454062e-03, -1.7673399e-02,  2.9252157e+00],
      dtype=float32),
```

The full experiment can be seen in **CIFAR10_Image_ Classification.ipynb** notebook, in the bias initialization experiment section.

6. Is it ok to initialize all weights to the same value as long as that value is randomly selected using He initialization?

No. If all weights are initialized to the same value, all neurons in that layer will output the same value, which means every unit will learn the same feature from input (generalization won't be achieved). During backpropagation, the updated weights will be same as well.

In order to make every unit learns different feature, we need to initialize weights randomly.