1)is it okay to initialize all the weight to the same value as long as that value is selected randomly using he Initialization ?

Ans :

No, all weights should be sampled independently; they should not all have the same initial value.

-One important goal of sampling weights randomly is to break symmetry: if all the weights have the same initial value, even if that value is not zero, then symmetry is not broken and backpropagation will be unable to break it. Concretely, this means that all the neurons in any given layer will always have the same weights. It’s like having just one neuron per layer, and much slower.

-It is virtually impossible for such a configuration to converge to a good solution.

2)is it okay to Initialize the bias terms to 0?

Ans : It is perfectly fine to initialize the bias terms to zero. Some people like to initialize them just like weights, and that’s okay too,it does not make much difference.

3)name three advantage of the ELU Activation Function over ReLU .

Ans : It can take on negative values, so the average output of the neurons in any given layer is typically closer to zero than when using the ReLU activation function. This helps alleviate the vanishing gradients problem.

-It always has a nonzero derivative, which avoids the dying units issue that can affect ReLU units.

-When the conditions are right then the SELU activation function ensures the model is self-normalized, which solves the exploding/vanishing gradients problems.

4)in which case would you want to use each of the following activation functions:ELU,leaky ReLU,ReLU,tanh,Logistic and softmax.?

Ans : -he SELU activation function is a good default.

-If you need the neural network to be as fast as possible, you can use one of the leaky ReLU variants instead (e.g., a simple leaky ReLU using the default hyperparameter value).

-simplicity of the ReLU activation function makes it many people’s preferred option, despite the fact that it is generally outperformed by SELU and leaky ReLU. However, the ReLU activation function’s ability to output precisely zero can be useful in some cases it can sometimes benefit from optimized implementation as well as from hardware acceleration.

-hyperbolic tangent (tanh) can be useful in the output layer if you need to output a number between –1 and 1, but nowadays it is not used much in hidden layers.

-logistic activation function is also useful in the output layer when you need to estimate a probability,rare in hidden layer.

5)what may happen if you set the momentum hyperparameter too close 1(e.g.0.99999)when using a momentum optimizer?

Ans : If you set the momentum hyperparameter too close to 1 (e.g., 0.99999) when using an SGD optimizer, then the algorithm will likely pick up a lot of speed, hopefully moving roughly toward the global minimum, but its momentum will carry it right past the minimum. Then it will slow down and come back, accelerate again, overshoot again, and so on. It may oscillate this way many times before converging, so overall it will take much longer to converge than with a smaller momentum value.

6)name three ways you can produce sparse model ?

Ans : One way to produce a sparse model (i.e., with most weights equal to zero) is to train the model normally, then zero out tiny weights.

-For more sparsity, you can apply ℓ1 regularization during training, which pushes the optimizer toward sparsity.

-A third option is to use the TensorFlow Model Optimization Toolkit.

7)does dropout slow down training?does it slow down Inference?

Ans : Yes, dropout does slow down training, in general roughly by a factor of two. However, it has no impact on inference speed since it is only turned on during training. MC Dropout is exactly like dropout during training, but it is still active during inference, so each inference is slowed down slightly. More importantly, when using MC Dropout you generally want to run inference 10 times or more to get better predictions. This means that making predictions is slowed down by a factor of 10 or more.