Large Scale Machine Learning and Deep Learning
Review Questions 5

1. Is it OK to initialize all the weights to the same value as long as that value is selected randomly using He initialization? Is it okay to initialize the bias terms to 0?

   Answer: 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 symmetries: if all the weights have the same initial value, even if that value is not zero, then symmetry is not broken (i.e., all neurons in a given layer are equivalent), 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. Its like having just one neuron per layer, and much slower. It is virtually impossible for such a configuration to converge to a good solution. It is perfectly fine to initialize the bias terms to zero. Some people like to initialize them just like weights, and thats okay too; it does not make much difference.

---

2. In which cases would you want to use each of the following activation functions: ELU, leaky ReLU (and its variants), ReLU, tanh, logistic, and softmax?

   Answer: the ELU 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). The simplicity of the ReLU activation function makes it many people's preferred option, despite the fact that they are generally outperformed by the ELU and leaky ReLU. However, the ReLU activation function's capability of outputting precisely zero can be useful in some cases. The 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. The logistic activation function is also useful in the output layer when you need to estimate a probability (e.g., for binary classification), but it is also rarely used in hidden layers (there are exceptions for example, for the coding layer of variational autoencoders). Finally, the softmax activation function is useful in the output layer to output probabilities for mutually exclusive classes, but other than that it is rarely (if ever) used in hidden layers.

---

3. What is batch normalization and why does it work?

   Answer: training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. The idea is then to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is done for each individual mini-batch at each layer, i.e., compute the mean and variance of that mini-batch alone, then normalize. This is analogous to how the inputs to networks are standardized. How does this help? We know that normalizing the inputs to a network helps it learn. But a network is just a series of layers, where the output of one layer becomes the input to the next. That means we can think of any layer in a neural network as the first layer of a smaller subsequent network. Thought of as a series of neural networks feeding into each other, we normalize

the output of one layer before applying the activation function, and then feed it into the following layer (sub-network).

---

4. Does dropout slow down training? Does it slow down inference (i.e., making predictions on new instances)?

   Answer: yes, dropout does slow down training, in general roughly by a factor of two. However, it has no impact on inference since it is only turned on during training.

---

5. What may happen if you set the `momentum` hyperparameter too close to 1 (e.g., 0.99999) when using a `MomentumOptimizer`?

   Answer: if you set the momentum hyperparameter too close to 1 (e.g., 0.99999) when using a `MomentumOptimizer`, then the algorithm will likely pick up a lot of speed, hopefully roughly toward the global minimum, but then it will shoot right past the minimum, due to its momentum. 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.