CS283: Assignment 3

Kerven Durdymyradov, KAUST ID: 187618

February 18, 2024

1 Problem 1: Recurrent Neural Networks (20 points)

a. (10 points)

The input of the Sigmoid function is any value, however, the output is the range between 0 and 1, which means the Sigmoid will squeeze any number to between 0 and 1. In LSTM, the forget gate is responsible for deciding how much of the previous cell state information should be passed on to the current time step. So, we are using the Sigmoid because it's good to decide to how much information should we forget, with respect to the closeness of output of the sigmoid to 0 and 1. On the other side if we use tanh the output will be the range -1 and 1, which will be more complex to decide compared to Sigmoid, also Sigmoid is more computationally cheaper than tanh.

b. (10 points)

1. In the context even the longer-range inputs also maybe more related. In LSTM if two inputs are longer-range from each other, less information will reach to the next input because of the distance. However in Transformers, because of these attention scores, even if the words are far from each other the attention scores might be high. As a result LSTM has difficulty on capturing long-range dependencies. It's designed to capture dependencies between inputs that are separated with small numbers of time steps not for the large, because it might lead to vanishing or exploiting gradients. But Transformers can catch even longer-range dependencies.

2. Because of LSTM's process is sequential, that means it's difficult to do the calculations in parallel time. And this makes it difficult to train the model on large datasets. However, on transformers they are parallelizable, it will be possible to process longer sequences and train larger models.

2 Problem 2: DCGAN (40 points)

3 Problem 3: Mode Collapse (40 points)

a. (5 points)

This is the problem when the generator produces a small number of varieties. For example, if there are 10 classes of images, it will produce images just for example 1-2 classes. It happens because the generator can fool the discriminator by generating close real images from these small number of varieties, instead of trying to cover the entire distribution.

b. (10 points)

The aim of UnRolled GAN to prevent the model keeps producing the same output. It allows the generator to predict several steps ahead to see how the discriminator might adjust its strategy. This helps the generator improve by considering the discriminator's future moves and then updating itself to avoid being easily predictable. This process helps avoid situations where the model gets stuck in a loop of repetitive outputs, leading to more diverse and stable results. The discriminator's training process remains the same as in traditional GANs. However, the generator forecasts up to 5 to 10 steps ahead to understand potential adjustments by the discriminator, which has been shown to enhance the model's performance significantly.

c. (10 points)

MAD-GAN changes the usual setup by having more than one generator to help solve a problem where the model keeps creating the same kind of output. In MAD-GAN, each generator aims to produce different types of data, and the discriminator learns to tell which generator made a specific piece of data. This setup helps because it makes each generator focus on making unique data types, leading to more variety in the outputs. If the generators just repeated the same few data types, the discriminator would easily guess which generator was which. So, to avoid being predictable, the generators try to cover more types of data, making the overall results more diverse and realistic. In short, by using several generators, MAD-GAN promotes more variety in what's created, tackling the issue where the model would otherwise keep producing similar results.

d. (15 points)

WGAN improves upon traditional GANs by fixing some common issues like unstable training and the problem where the model keeps producing the same types of outputs. The big change in WGAN is using the Wasserstein distance, or earth mover's distance, to measure how different the real and generated data distributions are. This distance measure is better than KL or JS divergence when comparing two distributions that don't overlap because it still provides a meaningful way to measure how far apart these distributions are. KL and JS divergence can't do this well because they end up infinite or undefined if the distributions don't overlap at all. For training, the WGAN changes the discriminator to give out a score that indicates how real or fake the data is, instead of just saying if it's real or fake. This score helps calculate the Wasserstein distance. The generator's job is then to make this score as low as possible, meaning it's trying to make the generated data look as real as possible by minimizing the Wasserstein distance between the real and generated data. This method encourages the generator to create a wide variety of realistic data, helping to solve the problem of the model only producing a limited range of outputs. In short, using the Wasserstein distance helps WGANs train more steadily and produce better and more diverse outputs.