Assignment 2 – Deep Learning

Amit Binenfeld and Omri Attal

Question 4

Qualitative: Model Advantages/Disadvantages:

- MLP/ANN:
  1. Advantages
     - Ability to work with incomplete knowledge: After we train MLP, the network can produce results even with incomplete information.
     - Parallel processing: MLPs have numerical strength that can perform more than one job at the same time, which makes them computationally efficient and can easily be parallelized. For example, computations for each node are generally independent of other nodes.
  2. Disadvantages
     - Unexplained behavior of the network: When the MLP produces wrong results, it doesn't give a clue as to how this had happened.
     - Determination of network structure: There is "winning" rule for determining the structure of MLPs. Proper network structure is achieved through experience and trial and error.
- RNN:
  1. Advantages
     - Model sequential data: Captures the sequential information present in the input data. Enabling the option for dependencies between inputs and their calculations while making predictions.
     - Computational cost: RNNs share the parameters across different time steps, which results in fewer parameters to train and decreases the computational cost.
  2. Disadvantages
     - Exploding/Vanishing gradients: Deep RNNs (RNNs with a large number of time steps) suffer from the exploding/vanishing gradient problem. Because RNNs remember all past computations, backpropagation will result in a very large or tiny derivatives in the early layers, which potentially causes forgetting important information.
     - Harder to train: Training an RNN is a very difficult task because of the exploding/vanishing gradients mentioned above.
- LSTM:
  1. Advantages

- Combats short term memory problem: LSTMs can learn which data in a sequence is important to keep or to throw away. By doing that, LSTM passes only the relevant information down the long chain of sequences to make predictions. This solves the problem of exploding/vanishing gradients.
- Model sequential data: Like RNNs, LSTMs are built to process sequential data, as mentioned above.

2. Disadvantages
   - Computational heavy: Each step in an LSTM requires many computations. For example, each layer has 4 gates which include multiplication and addition of matrices. This results in extending the time it takes to process the inputs in each time step as opposed to RNNs.

Qualitative: Time and Space Complexity

For simplicity we'll assume that calculating sigmoid, relu and tanh functions take $O(1)$ time.

1. MLP:
   - Space: The layer stores a weight matrix with $O(m \cdot n)$ for m is the size of the output features and n is the size of the input features.
   - Time: let's denote the input parameter's shape as $d \times n$. The forward function performs a matrix multiplication of the weights and the input. If we use the naïve algorithm for matrix multiplication, we get $O(dnm)$. We assumed calculating Sigmoid take $O(1)$, we'll get $O(dm)$ time to calculate the matrix of size $d \times m$ resulted. So, overall $O(dmn + dm) = O(dnm)$
2. RNN:
   - Space: The layer stores 2 weight matrices in size $n \times m \; and \; m \times m$, for n and m described above. So, overall $O(nm + m^2)$.
   - Time: let's denote the input parameter shape as before ($d \times n$). Therefore, the hidden parameter shape $d \times m$. Here we have two matrix multiplications, the first takes $O(dnm)$ while the second $O(dm^2)$ if using naïve algorithm resulting in two $d \times m$ matrices. The functions tanh and relu take $O(1)$ time and matrices addition also take $O(dm)$, overall we get $O(dnm + dm^2 + dm) = O(dnm + dm^2)$
3. LSTM:
   - Space: Our layer stores 8 matrices. 4 of them are of size $n \times m$ and 4 are $m \times m$. So overall $O(4nm + 4m^2) = O(nm + m^2)$
   - Time: We'll denote input shape and hidden shape as in RNN since the input is the same. Calculating input gate, forget gate, the $\hat{c}$ (g in our code) and output gate take: $O(4dnm + 4dm^2 + dm)$ as described in the RNN

layer, we get that these procedures take $O(dnm + dm^2)$. We need to specify time complexity calculating the resulting cell state and hidden state. Calculating c1 take $O(2dm) = O(dm)$ since we use Hadamard product and addition of two $d \times m$ matrices. Finally, calculating h1 takes also $O(dm)$ because tanh takes $O(1)$ time. Overall, the forward procedures take $O(dnm + dm^2)$

Quantitative: Training Time

We ran all networks with the same input including forward + gradient calculating.

Since we use random weights with each run, the time it takes to calculate the gradient and updates the weight matrices differ for each run. The inputs and the labels are present in the code. All the networks include the same "update" function and the same input.

1. MLP: 7.56 ms, 5.33 ms, 4.59, 3.6 ms, 7.56 ms, AVG: 5.728 ms

2. RNN: 3.71 ms, 3.24 ms, 4.34, 3.97 ms, 5.88 ms, AVG: 4.228 ms

3. LSTM: 11.2 ms, 5.54 ms, 7.42 ms, 9.4 ms, 5.19 ms, AVG: 7.75 ms