Machine Learning in Healthcare- 336546
Homework 4- due 7/3/2021

Adi Alfassi, Lior Drasinover, Ori Rajchert

Part 1:

Task 2: Explain how changing the activation function from ReLu to LeakyRelu or tanh or sigmoid Can affect the model:

The ReLu activation function is a non-linear function, that has an output of 0 if the input is less than 0, and an output that is equal to the input if the input is greater than 0.

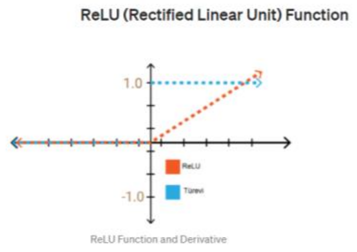

Figure 1 ReLu activation function[1].

Advantages:

- Activation is sparse and efficient.
- ReLu is less computationally expensive.
- A lighter network design - due to the nature of the function, when normalized input is given, about half of the weights would result 0 - which can lead to faster calculation.
- Higher preference for multi-layer networks.

Disadvantages:

- The dying ReLu problem - for negative value, gradient will be zero, which means those neurons which go into that state will stop responding to variations in error. This doesn't give us any learning in the region.
- Mainly meant to be used within Hidden layers of a Neural Network Model.
- Constant derivative - which makes the learning process less effective.
- When normalized input is given, about half of the weights would result 0 may cause ignoring part of the data[2].

The tanh activation function is a non-linear function, ranges between -1 to 1. The function gradient is larger for input values that are closer to zero, and becomes smaller for input values that are closer to 1 or -1 [3].

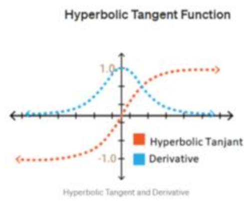

Figure 2 tanh activatione function[1].

Advantages:

- Its derivative is steeper in the center region, which means it can get more values. This means that it will be more efficient because it has a wider range for faster learning and grading.
- Negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero [4].

Disadvantages:

- Vanishing gradient problem - It is relatively flat except for a very narrow range (that range being about -1 to 1). The derivative of the function is very small unless the input is within the narrow range. It has a flat derivative which makes it difficult to improve the weights through gradient descent. This problem gets worse as the model has more layers. This issue is called the vanishing gradient problem and may cause the learning process to be less efficient.
- Activation is dense and therefore costly.
- Can be mainly used for classification between two classes.
- We had a major difficulty building models with many layers when using the tanh function.

Comparing the two activation functions, ReLu activation function is faster than the tanh function and prevents backpropagation errors unlike the tanh function.
On the other hand, even though tanh gets close to flat, it isn't completely flat anywhere. So its output always reflects changes in its input.

Using a tanh will cause almost all neurons to fire in an analog way. That means almost all activations will be processed to describe the output of a network. In other words the activation is dense. This is costly. We would ideally want a few neurons in the network to not activate and thereby making the activations sparse and efficient.
ReLu give us the benefit of sparse and efficient activation. This means a fewer neurons are firing (sparse activation) and the network design and implementation is lighter.
In addition, ReLU is not sensitive to vanishing gradients, whereas the other method of tanh is, causing a slowdown learning in the network.
Furthermore, ReLu is less computationally expensive than tanh because it involves simpler mathematical operations. That is an important point to consider when we are designing deep neural nets. As we notice, ReLU is the most widely used activation function today [5].

Task 3: Train the new model using 25 and 40 epochs. What difference does it makes in term of performance?

When we train the model with more epochs we let our model to get to know the data better.
We assumed that 40 epochs will give us a better results. However, it might give us over-fitted results.
On the whole, we don't see a significant changes between the model that runs with 25 epochs vs. the model that runs with 40 epochs.

For 25 epochs: test loss, test acc: [0.8439764046669006, 0.62857145]
For 40 epochs: test loss, test acc: [0.833736378805978, 0.62857145]

Task 4: What are the advantages of the mini batch vs. SGD?

In **Stochastic Gradient Descent** (SGD), we consider just one example at a time to take a single step. While activating SGD, in every epoch weights are updated by all examples. Meaning that all the examples are going through the model separately per each epoch. For example- if our training set has 30 examples, the weight will be updated 30 times in each epoch.

While activating **mini batch**, in every epoch, the weights are updated after every batch of examples. This process goes through the model separately in each epoch. For each batch the weights are updated according to the average gradient of the current batch. In mini batch, we use a batch of a fixed number of training examples which is less than the actual dataset.

For example, if our training set has 30 examples and a batch size of 3, the weight will be updated 10 times per each epoch. With each weight being updated according to the average gradient of the 3 examples in each batch.

Since we are considering just one example at a time in SGD, the cost will fluctuate over the training examples and it will not necessarily decrease. But in the long run, we will see the cost decreasing with fluctuations. As the cost is so fluctuating, it will never reach the minima but it will keep dancing around it. SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.

The mini batch algorithm is more efficient and less complex compared to the SGD algorithm, as it carries less updates per epoch while using an average gradient to determine the gradient of the examples in the batch.

How does batch normalization impacts the results:

Batch normalization is a technique for improving the speed, performance, and stability of a neural network. This technique normalizes the inputs of each layer in such a way that, they have a mean activation output zero and a unit standard deviation.

Batch normalization impacts the results in several ways:

1. Speeds up training: While using batch normalization, less information is needed for classification, which causes reduce in training and calculation time. In addition, the batch normalized model is more robust.
2. Regularizes the model: Batch normalization regularizes gradient from distraction to outliers and flows towards the common goal (by normalizing them) within a range of the mini batch. Resulting in the acceleration of the learning process.
3. Decreases importance of initial weights: Batch normalization uses the similar number of steps in order to "reach the minimal point" for every initial weights, in oppose to non-normalized model, in which the number of steps differs according to the initial weights[6].

Part 2

- How many layers does it have?
  This model contains 8 layers: 5 convolutional layers, 3 fully connected layers.

- How many filters in each layer?
  Filters are relevant for convolution layers only:
  1st convolution layer - 64 filters.
  2nd convolution layer - 128 filters.

3rd convolution layer - 128 filters.
4th convolution layer - 256 filters.
5th convolution layer - 256 filters.

- Would the number of parameters be similar to a fully connected NN?
  In Convolutional NN, output will be the number of filters times the size of the filters.
  In a fully connected NN, each neuron connects to every other neuron, which causes a higher number of parameters than the other types of layers.

- Is this specific NN performing regularization?
  This NN performs regularization using kernel regularization using two methods:
  1. **L2** - In order to apply a penalty on each convolution layer's kernel, we use L2 class. This penalty helps us to reduce overfitting.
  2. **Dropout** - A dropout is an approach to regularization in neural networks which helps to reduce interdependent learning amongst the neurons. A single model can be used for simulation of a large number of different network architectures by randomly dropping out nodes during training. Dropout offers a very computationally cheap and effective regularization method to reduce overfitting and improve generalization error.

<u>Task 2, results:</u>
Original filters: test loss, test acc: [8.455934965951101, 0.2742857]
Filter reduction: test loss, test acc: [5.044688846043178, 0.25142857]

References

[1] “Comparison of Activation Functions for Deep Neural Networks _ by Ayyüce Kızrak _ Towards Data Science.pdf.”

[2] S. S. May, “ReLU — Most popular Activation Function for Deep Neural Networks,” p. 5.

[3] “Understanding Activation Functions in Neural Networks _ by Avinash Sharma V _ The Theory Of Everything _ Medium.pdf.”

[4] “Rectified Linear Units (ReLU) in Deep Learning _ Kaggle.pdf.”

[5] H. S. Jan, “Activation Functions: Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning,” p. 12.

[6] “Regularization_ Batch-normalization and Drop out _ by aditi kothiya _ Analytics Vidhya _ Medium.pdf.”