HW 4

Machine Learning in Healthcare

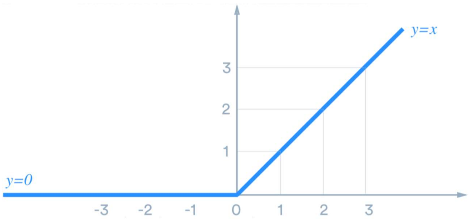
Naama Rivlin 311576599

Daniel Sapir 308412113

PART 1: Fully connected layers

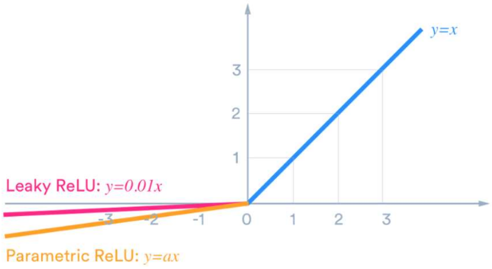
2. Activation functions and how they can affect the model:

ReLU (Rectified Linear Unit) is defined as $y = \max(0, x)$ –



The fact that ReLU is zero for all negative inputs leads to lower computational complexity and faster convergence, which results in smaller running time. However, it also leads to sparse activation, which is not always an advantage. On the one hand, sparsity can be good because it causes only neurons that are important for action to be activated. This can contribute to shorter running time and prevent overfitting. On the other hand, neurons can be not activated at all, causing a problem called "dying ReLU" – The neuron gets a negative value and therefore its output is 0, and is likely to remain stuck that way, so that in fact the neuron does not function at all and. Eventually a network is formed of which a significant part of it is disabled. The "dying ReLU" problem can occur when there is a significant negative bias or when the learning rate is too high.

ReLU is defined as $y = \mathrm{x}$ for $x > 0$ and $y = \alpha x$ for $x < 0$.



This solves the problem of "dying ReLU" because thus the neurons that receive negative values can "recover" and return to function.

Credit:

- https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
- https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

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

There is no significant difference between 25 epochs and 40 epochs in terms of performance. For both options, an accuracy of ~0.65 and a loss of ~0.82 are obtained. As we have learned, in each epoch the weights are updated to minimize the error. We hypothesize that there is no improvement because after 25 epochs there is a convergence of the model to a local minima. Performing additional epochs does not cause significant improvement. In addition, performance is low because the data is not very good.

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

Neural network training is done using a gradient descent algorithm: Perform a prediction, compare the prediction to the expected values, use the difference to find the error and update the weights, and repeat the process until an optimal result is reached. The more training examples used in the estimate, the more accurate this estimate will be. using fewer examples results in a less accurate estimate. In SGD, the "batch Size" is 1 sample – the algorithm uses only one example at a time. The result is a model that performs fast learning but gives noisy and uneven results. On the other hand, a model that uses all the examples at a time (BGD) will have a very long runtime. The compromise between the 2 options is Mini-Batch Gradient Descent, in which the batch size is bigger than 1 sample but smaller than the size of the training set (usually 32, 64, or 128 samples), so the computational cost is still low, but the prediction is more accurate. A good balance is struck when the minibatch size is small enough to avoid some of the poor local minima, but large enough that it does not avoid the global minima or better-performing local minima.

Credit:

- https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/
- https://stats.stackexchange.com/questions/49528/batch-gradient-descent-versus-stochastic-gradient-descent

add batch normalization layers. How does it impact your results?

Adding the batch normalization layers caused a very small improvement (accuracy increased by about 2%). Batch normalization is used to prevent a situation in which one weight gains an unusually high value which causes instability in the network. We add batch normalization between the layers - after each hidden layer, the weights are standardized. This process happens separately for each batch, and should give better results. As already mentioned, the data is not very good so adding batch normalization layers is not helpful.

PART 2: Convolutional Neural Network (CNN)

1. 2D CNN model-

- **How many layers does it have?**
- **How many filters in each layer?**
- **Would the number of parameters be similar to a fully connected NN?**
- **Is this specific NN performing regularization?**

Layers and filters:

- Conv2D: There is 5 layers.
    - 1st- 64 filters.
    - 2nd – 128 filters.
    - 3rd – 128 filters.
    - 4th – 256 filters.
    - 5th -256 filters.
- Dense: 3 layers.

Sum of the layers is 8.

Would the number of parameters be similar to a fully connected NN?

As we saw in the tutorial when we use convolution, we reduce the parameter's number. So as a result, the number of the parameters in this part are smaller than the number of the parameters in fully connected NN. This is an advantage of CNN over fully connected NN. We can use CNN when we applied images like in this case. Is this scenario the use is in one filter for a variety of images.

Is this specific NN performing regularization?

Yes. The Conv2D layer includes regularization. The regularization is by L2 norm and a value of rate of 1e-2.

In addition, Dropout is another technique for regularization. This technique includes removing of nodes randomly from the calculation of errors.

2. train the model with the number of filters reduced by half. What were the results?

Results:

```
test loss, test acc: [8.332893371582031, 0.2971428632736206]
```

By reduced:

```
test loss, test acc: [4.478744029998779, 0.35428571701049805]
```

we can see that reducing the filters increase the ACC value. But anyway, in both cases the results are not good. In addition, there is sometimes increase in acc and sometimes decrease.

We can explain that because the data are not of high quality.