Course name: Machine Learning in Healthcare
HW: HW4

Machine Learning in Healthcare

Winter 2020-2021

HW4 -Theoretical Part

Mor Ventura: 313177412
Omri Magen: 302301718

Date: 01.02.2021

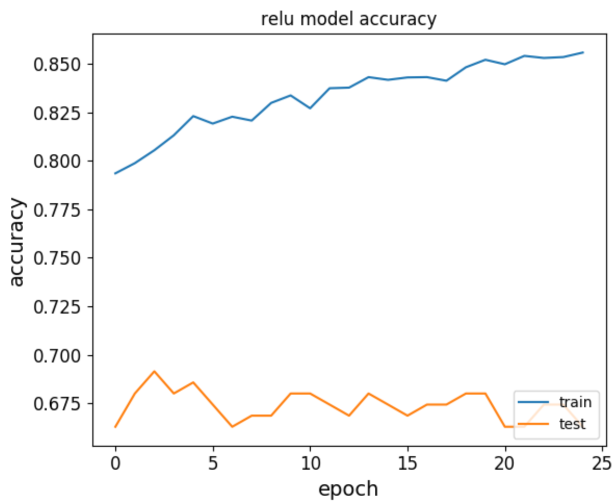
Course name: Machine Learning in Healthcare
HW: HW4

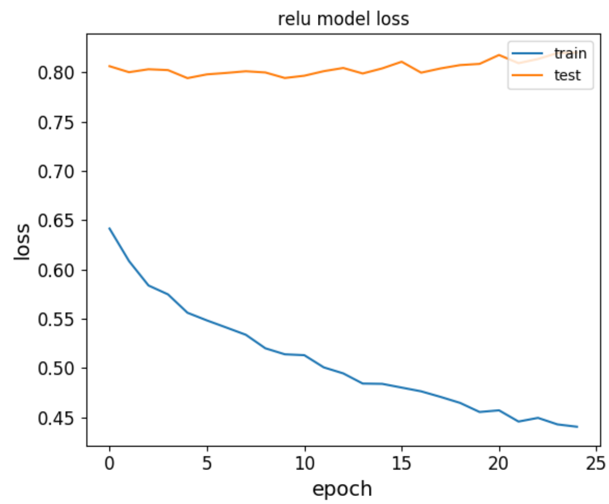
MLH- HW4

** using the faculty server, user: stu4

1. **Task 1:**

the accuracy and loss over the testing set:





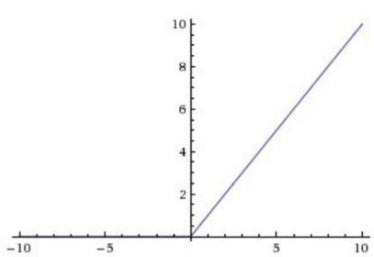
the evaluation results are:

```
Test Loss is 0.82
Test Accuracy is 66.29 %
```

2. **Task 2:** Activation functions

Generally speaking, changing the activation function is actually changing the output of the nodes, i.e., it is the decision whether to fire or not (as known as action potential). Practically, changing or defining the activation function should be correlated to an approximated known characteristics or should be leading to faster training.

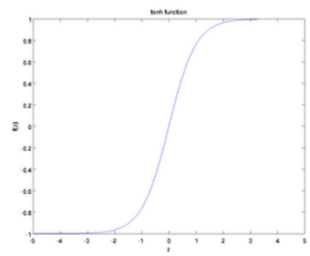
We used in the first model the 'ReLU' activation function:



which gives the maximum of [0, input].

Course name: Machine Learning in Healthcare
HW: HW4

Now, we changed it to tanh activation function (which it is a scaled sigmoid):

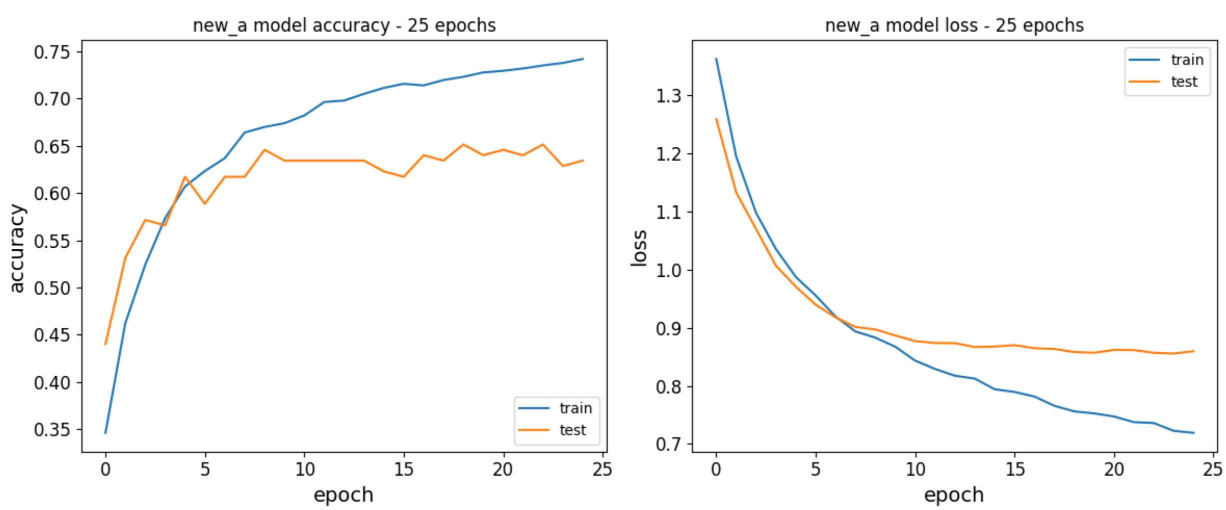


$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

this function maps to [-1,1] range, when negative input to a strongly negative and near zero inputs to near zero.
ReLU gives more sparsed outputs and more easy to compute with the gradients (0 or 1).

3. **Task 3**: Number of epochs

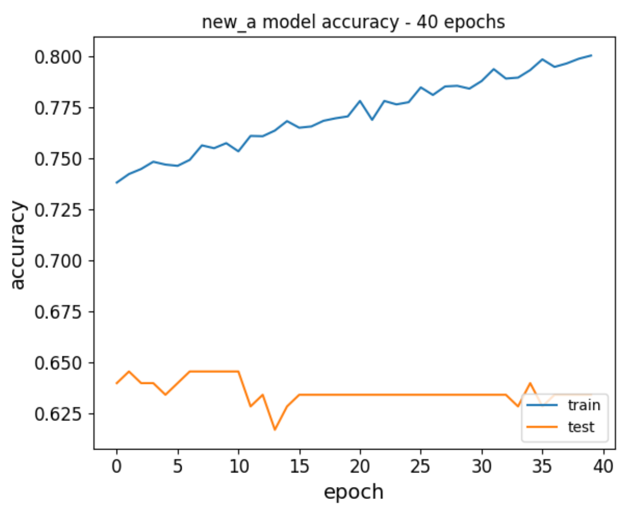
using the tanh activation function, over 25 epochs gives us the following results:

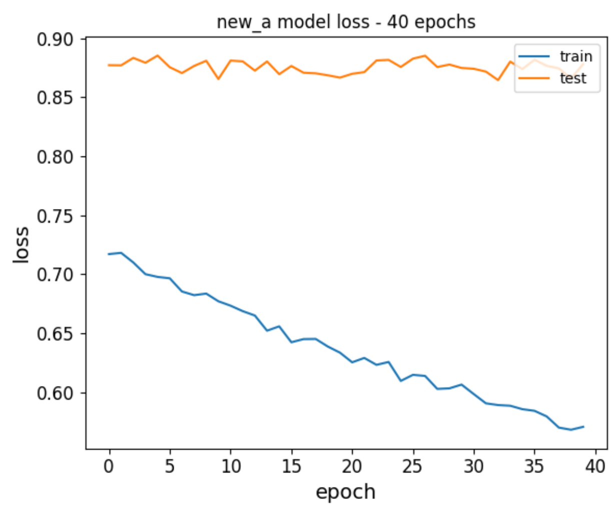


the evaluation results are:

```
Test Loss is 0.88
Test Accuracy is 65.14 %
```

<u>using the tanh activation function, over 40 epochs gives us the following results:</u>






```
Test Loss is 0.88
Test Accuracy is 63.43 %
```

We would expect to see higher accuracy and the lower loss rate as the number of epochs increased, of course taking a risk of over-fitting. We are not sure why the results are not exactly fit to our expectations.

Course name: Machine Learning in Healthcare
HW: HW4

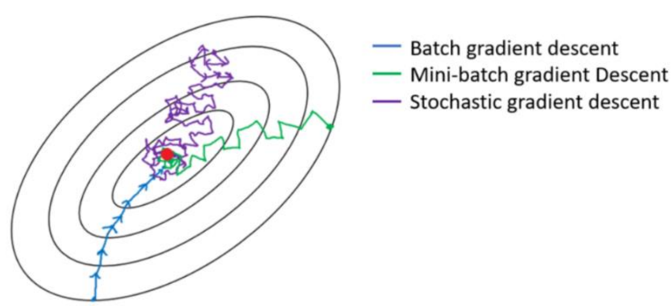
4. **Task 4:** mini-batches and batch normalization

mini-batches

The batch size is the number of training examples in one iteration or pass, before the model is updated. Batch size controls the accuracy of the estimate of the error gradient when we tarin NN. There is a tension, tradeoff, between batch size and speed and stability of the learning process. Larger batch size makes larger gradient steps than smaller batch sizes for the same number of samples seen.

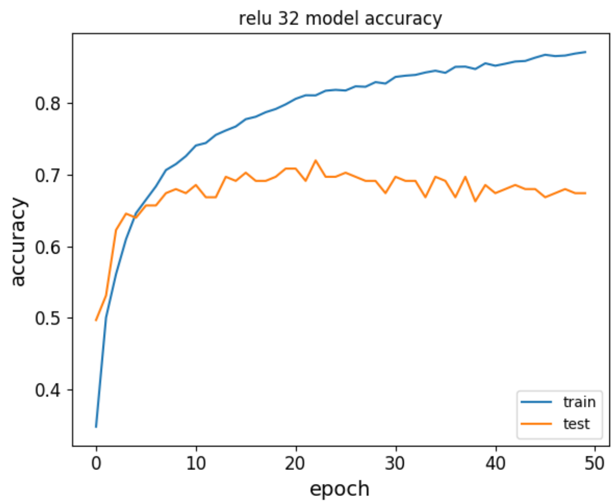
Batch, SGD (Stochastic) and mini-batch gradient descent are 3 main flavors. In SGD, we consider just one example at a time to take a single step.

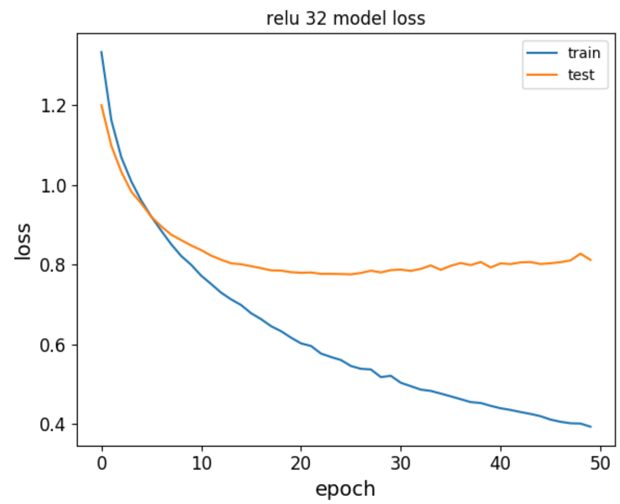
In mini-batches we use a batch of fixed number of training examples, and we update the weights by the mean gradient of it. In both, the average cost over the epochs fluctuates.



The advantages of the mini-batches vs. SGD are: noise reduction (averaging), lower variance, efficient – using vectorization.

<u>using the ReLU activation function, over 50 epochs, with 32 batch size, gives us the following results</u>:






```
Test Loss is 0.81
Test Accuracy is 67.43 %
```

There is a small improvement in the results, compared to the ReLU 64 over 25 epochs.

Comparing the 32 ReLU model with the same number of epochs (25):

```
Test Loss is 0.85
Test Accuracy is 65.14 %
```

- not so good as before.

(And specifically in the 25th epoch ,compared to the ReLU 64 model:

ReLU 32, epoch 25:

train accuracy: 0.8185048

test accuracy: 0.69714284

ReLU 64, epoch 25:

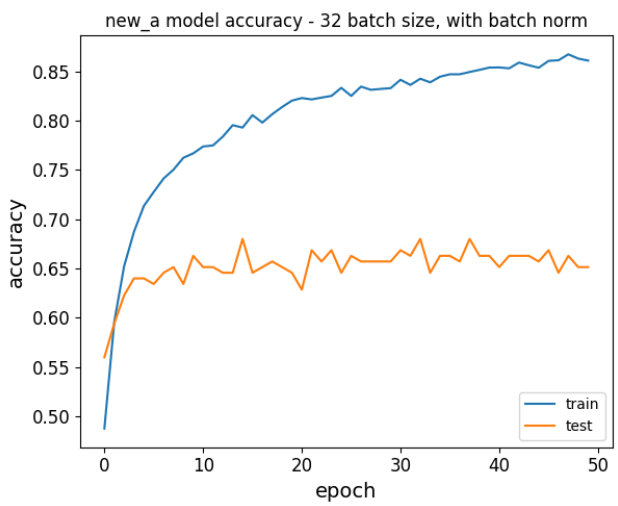
train accuracy: 0.78081554

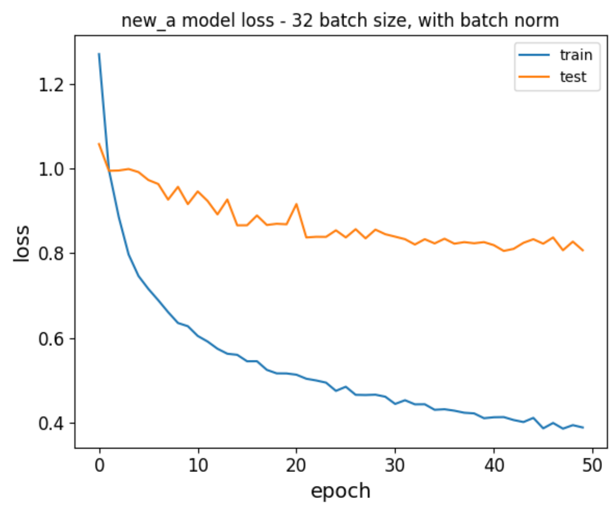
test accuracy: 0.6685714)

batch normalization

Batch normalization is a technique for standardize the input to a layer for each mini-batch. It leads to the effect of stabilizing the learning process and reducing the number of training epochs required.

Adding the batch normalization layers for the 'new_a_model' with the 32 batch size, over 50 epochs, gives us the following results:






```
Test Loss is 0.81
Test Accuracy is 65.14 %
```

In practice, we do see an improvement in the loss rate and in the accuracy (comparing to the results in task 3).

Part 2: CNN

1. **Task 1:** 2D CNN

- The number of layers:
  - 5 2D convolution layers
  - 3 dense (fully connected) layers
  - 5 batch normalization layers
  - 3 max pooling layers
  - 6 optional dropout layers
  - 1 flatten layer
  - 1 permutation layer

  So, in total there are 24 layers or 18, depend on the dropout condition (if set).
  scheme of the net sizes:

$$\underbrace{\mathbf{32x32x1}}_{input} \underbrace{\rightarrow}_{\substack{Conv\ 3x3x1 \\ x64}} \mathbf{32x32x64} \underbrace{\rightarrow}_{\substack{max-pool \\ 2x2}} \mathbf{16x16x64} \underbrace{\rightarrow}_{\substack{Conv\ 3x3x64 \\ x128}} \mathbf{16x16x128} \underbrace{\rightarrow}_{\substack{Conv\ 3x3x128 \\ x128}}$$

$$\mathbf{16x16x128} \underbrace{\rightarrow}_{\substack{max-pool \\ 2x2}} \mathbf{8x8x128} \underbrace{\rightarrow}_{\substack{Conv\ 3x3x128 \\ x256}} \mathbf{8x8x256} \underbrace{\rightarrow}_{\substack{Conv\ 3x3x256 \\ x256}} \mathbf{8x8x256} \underbrace{\rightarrow}_{\substack{max-pool \\ 2x2}} \mathbf{4x4x256}$$

$$\underbrace{\rightarrow}_{flatten} \mathbf{4096x1} \underbrace{\rightarrow}_{dense\ 512x4096} \mathbf{512x1} \underbrace{\rightarrow}_{dense\ 128x512} \mathbf{128x1} \underbrace{\rightarrow}_{dense\ 4x128} \underbrace{\mathbf{4x1}}_{output}$$

- The number of filters in each layer:

  [Initial_num_filters (64),
  Initial_num_filters*2 (128),
  Initial_num_filters*2 (128),
  Initial_num_filters*4 (256),
  Initial_num_filters*4 (256)]

- <u>The number of parameters compares to fully connected NN:</u>
  The number of parameters would not be similar, because the CNN's filters and max-pooling layers decrease the number of the parameters. Of course, the NN tail, the last dense layers, increase the number of parameters significantly.
- <u>Performing regularization:</u>
  The specific NN performs regularization. There is kernel regularization (l2 norm) in every convolution layer, which applies penalty on the loss.

2. **Task 2**: number of filters

For the original CNN (with those filters' sizes - [64,128,128,256,256]), we got:
test loss, test acc: [7.86983107975551, 0.3257143]

For the new CNN (with half-size filters – [32,64,64,128,128]), we got:
test loss, test acc: [4.6907029042925155, 0.30857143]

When we are increasing the number of filters, it is as if we were increasing the number of features' detectors, so we were generating more features' maps -> i.e. the network learns more and better.
So, when we decrease the number of filters – we will get less accurate model.
Specifically, we can see here a reduction in the loss and in the accuracy.

On a personal note, thank you for great guidance and teaching through the semester.

Figure: Adele, the pioneer prophet of NN's success, with the song "**Rolling in the deep**":)