Nnet-ExperimentalResultsAndAnalysis

May 29, 2017

In [1]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

from sklearn import preprocessing
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from collections import Counter
from IPython.display import Image
import pprint as pp
import random
import time
import sys
import os
try:
   import cPickle as cPickle
except:
   import pickle as cPickle
```

0.0.1 Data Analysis

Listing of short codes

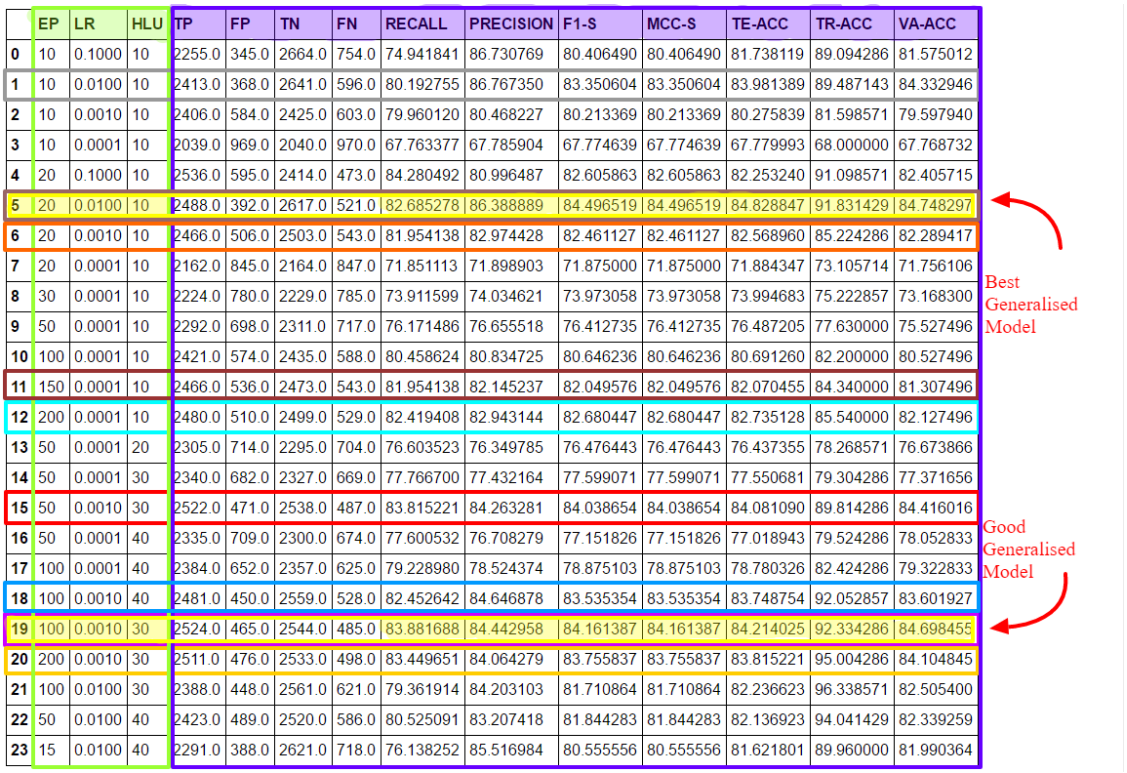
- EP: No. of Epochs

- LR: Learning Rate
- HLU: No. of Hiddel Layer Units
- TP: True Positive
- FP - False Positive
- TN: - True Negative
- FN: - False Negative
- F1-S : - F1-Score
- MCC-S: - Matthews Correlation Coefficient
- TE-ACC: Testing Accuracy for last Epoch
- TR-ACC: Training Accuracy for last Epoch
- VA-ACC: Validation Accuracy for last Epoch

Tabulation of training results for different hyper-parameters.

```
In [2]: Image(filename='training/tables/results_final_end.png')
```

Out[2]:



	EP	LR	HLU	TP	FP	TN	FN	RECALL	PRECISION	F1-S	MCC-S	TE-ACC	TR-ACC	VA-ACC
0	10	0.1000	10	2255.0	345.0	2664.0	754.0	74.941841	86.730769	80.406490	80.406490	81.738119	89.094286	81.575012
1	10	0.0100	10	2413.0	368.0	2641.0	596.0	80.192755	86.767350	83.350604	83.350604	83.981389	89.487143	84.332946
2	10	0.0010	10	2406.0	584.0	2425.0	603.0	79.960120	80.468227	80.213369	80.213369	80.275839	81.598571	79.597940
3	10	0.0001	10	2039.0	969.0	2040.0	970.0	67.763377	67.785904	67.774639	67.774639	67.779993	68.000000	67.768732
4	20	0.1000	10	2536.0	595.0	2414.0	473.0	84.280492	80.996487	82.605863	82.605863	82.253240	91.098571	82.405715
5	20	0.0100	10	2488.0	392.0	2617.0	521.0	82.685278	86.388889	84.496519	84.496519	84.828847	91.831429	84.748297
6	20	0.0010	10	2466.0	506.0	2503.0	543.0	81.954138	82.974428	82.461127	82.461127	82.568960	85.224286	82.289417
7	20	0.0001	10	2162.0	845.0	2164.0	847.0	71.851113	71.898903	71.875000	71.875000	71.884347	73.105714	71.756106
8	30	0.0001	10	2224.0	780.0	2229.0	785.0	73.911599	74.034621	73.973058	73.973058	73.994683	75.222857	73.168300
9	50	0.0001	10	2292.0	698.0	2311.0	717.0	76.171486	76.655518	76.412735	76.412735	76.487205	77.630000	75.527496
10	100	0.0001	10	2421.0	574.0	2435.0	588.0	80.458624	80.834725	80.646236	80.646236	80.691260	82.200000	80.527496
11	150	0.0001	10	2466.0	536.0	2473.0	543.0	81.954138	82.145237	82.049576	82.049576	82.070455	84.340000	81.307496
12	200	0.0001	10	2480.0	510.0	2499.0	529.0	82.419408	82.943144	82.680447	82.680447	82.735128	85.540000	82.127496
13	50	0.0001	20	2305.0	714.0	2295.0	704.0	76.603523	76.349785	76.476443	76.476443	76.437355	78.268571	76.673866
14	50	0.0001	30	2340.0	682.0	2327.0	669.0	77.766700	77.432164	77.599071	77.599071	77.550681	79.304286	77.371656
15	50	0.0010	30	2522.0	471.0	2538.0	487.0	83.815221	84.263281	84.038654	84.038654	84.081090	89.814286	84.416016
16	50	0.0001	40	2335.0	709.0	2300.0	674.0	77.600532	76.708279	77.151826	77.151826	77.018943	79.524286	78.052833
17	100	0.0001	40	2384.0	652.0	2357.0	625.0	79.228980	78.524374	78.875103	78.875103	78.780326	82.424286	79.322833
18	100	0.0010	40	2481.0	450.0	2559.0	528.0	82.452642	84.646878	83.535354	83.535354	83.748754	92.052857	83.601927
19	100	0.0010	30	2524.0	465.0	2544.0	485.0	83.881688	84.442958	84.161387	84.161387	84.214025	92.334286	84.698455
20	200	0.0010	30	2511.0	476.0	2533.0	498.0	83.449651	84.064279	83.755837	83.755837	83.815221	95.004286	84.104845
21	100	0.0100	30	2388.0	448.0	2561.0	621.0	79.361914	84.203103	81.710864	81.710864	82.236623	96.338571	82.505400
22	50	0.0100	40	2423.0	489.0	2520.0	586.0	80.525091	83.207418	81.844283	81.844283	82.136923	94.041429	82.339259
23	15	0.0100	40	2291.0	388.0	2621.0	718.0	76.138252	85.516984	80.555556	80.555556	81.621801	89.960000	81.990364

A good model should have high but very close values for PRECISION, RECALL, F1-S, MCC-S, TE-ACC,TR-ACC and VA-ACC. The one that fits this criteia most is the 5th and 19th model in the table. Therefore, its hyperparameters are the most optimum values for the sentiment neural network that we have trained. Those are: - **LR** = 0.01, 0.001 - **EP** = 20, 100 - **HLU** = 10, 30

0.0.2 Load Trained Model's with above Parameters

LR=0.01, **EP**=20, **HLU**=10

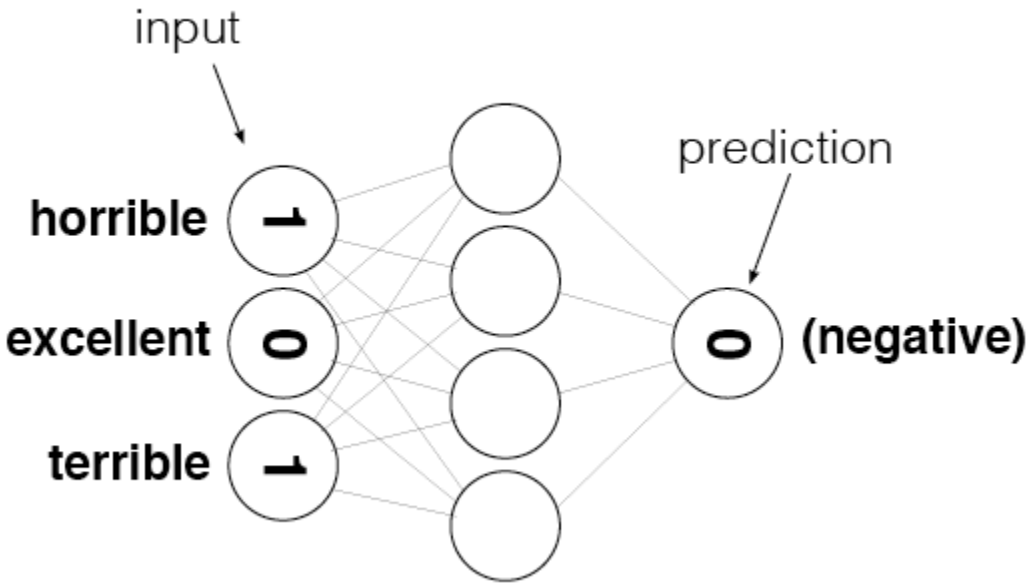
```
In [3]: %load toolkit/EfficientLowNoiseSentimentalNeuralNetwork.py
```

Predictions Criteria

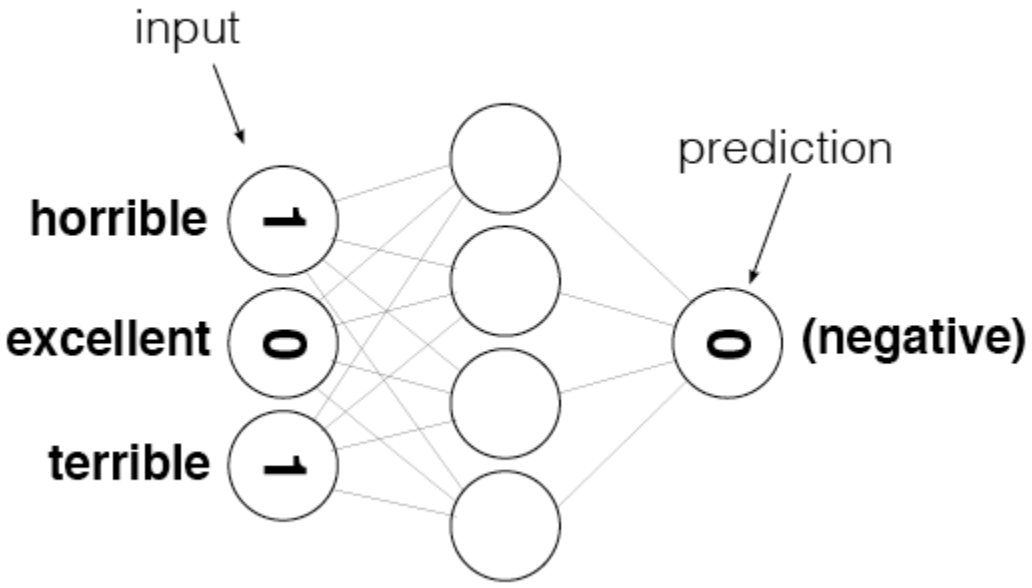
```
python if(prediction > 0.5):     Sentiment = POSITIVE if(prediction
< 0.5):     Sentiment = NEGATIVE
```

```
In [4]: # load the model from disk
        model_trained_filepath = 'models/se_model_lr0.01epoch_20hlu_10.sav'
        loaded_model = cPickle.load(open(model_trained_filepath, 'rb'))

In [5]: loaded_model.predict("This product sucks!! I bought it and it proved disgus
```

```
Out[5]: array([ 0.00281025])

In [6]: loaded_model.predict("I hate how this product was badly packaged. It was de

Out[6]: array([ 0.28054857])

In [7]: Image(filename='illustration/binary_negative.png')

Out[7]:
```




```
In [8]: loaded_model.predict("I enjoyed it. It is a very awesome product")

Out[8]: array([ 0.95822964])

In [9]: loaded_model.predict("I enjoyed using this product do much and would defini

Out[9]: array([ 0.65593228])

In [10]: Image(filename='illustration/binary_negative.png')

Out[10]:
```


input
prediction
horrible
1
excellent
0
terrible
1
0
(negative)