Optimization_HNFG_IMDB_V2

April 27, 2021

1 Computational Intelligence Project: Sentiment Analysis on IMDB dataset Using HNGFS

Musab - 19030008

In this Notebook, I have done implementing Hybrid Neuro Genetic Fuzzy System. In this second approach, an optimization is applied to Neuro-fuzzy inference system using genetic algorithm. Neuro-fuzzy is also called ANFIS. Genetic Algorithm is used to optimize the hybrid model using different weights of the network.

[ ]:

```
from keras.layers import Input, Dense, Dropout
from keras.models import Model
from keras.datasets import mnist, imdb
import numpy as np
from keras import regularizers
import matplotlib.pyplot as plt
from FuzzyLayer import FuzzyLayer
from tensorflow.python.client import device_lib
from keras.utils import to_categorical
import re
import keras
from keras.models import Sequential
from keras.models import Model
from keras.models import load_model
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, confusion_matrix
```

[ ]:

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=5000)
```

```
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested
sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with
different lengths or shapes) is deprecated. If you meant to do this, you must
specify 'dtype=object' when creating the ndarray
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/datasets/imdb.py:159:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
```

```
'dtype=object' when creating the ndarray
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/datasets/imdb.py:160:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
```

[ ]:

```
print("train_data ", x_train.shape)
print("train_labels ", y_train.shape)
print("_"*100)
print("test_data ", x_test.shape)
print("test_labels ", y_test.shape)
print("_"*100)
print("Maximum value of a word index ")
print(max([max(sequence) for sequence in x_train]))
print("Maximum length num words of review in train ")
print(max([len(sequence) for sequence in x_train]))
```

```
train_data  (25000,)
train_labels  (25000,)
--------------------------------------------------------------------------------
--------------------
test_data  (25000,)
test_labels  (25000,)
--------------------------------------------------------------------------------
--------------------
Maximum value of a word index
4999
Maximum length num words of review in train
2494
```

[ ]:

```
x_train = x_train[:10000]
y_train = y_train[:10000]
print(x_train.shape)

x_test = x_test[:10000]
y_test = y_test[:10000]
print(x_test.shape)
```

```
(10000,)
(10000,)
```

[ ]:

```
def vectorize_sequences(sequences, dimension=5000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results
```

[ ]:

```
x_train = vectorize_sequences(x_train)
x_test = vectorize_sequences(x_test)

print("x_train ", x_train.shape)
print("x_test ", x_test.shape)
```

```
x_train  (10000, 5000)
x_test  (10000, 5000)
```

[ ]:

```
y_train = np.asarray(y_train).astype('float32')
y_test = np.asarray(y_test).astype('float32')
print("y_train ", y_train.shape)
print("y_test ", y_test.shape)
```

```
y_train  (10000,)
y_test  (10000,)
```

[ ]:

```
x_val = x_train[:2000]
partial_x_train = x_train[2000:]
y_val = y_train[:2000]
partial_y_train = y_train[2000:]

print("x_val ", x_val.shape)
print("partial_x_train ", partial_x_train.shape)
print("y_val ", y_val.shape)
print("partial_y_train ", partial_y_train.shape)
```

```
x_val  (2000, 5000)
partial_x_train  (8000, 5000)
y_val  (2000,)
partial_y_train  (8000,)
```

1.0.1 ANFIS Network Ensemble with Genetic Algo

[ ]:

```
class FuzzyLayer(Layer):

    def __init__(self,
                 output_dim,
                 initializer_centers=None,
                 initializer_sigmas=None,
                 **kwargs):
```

```
        if 'input_shape' not in kwargs and 'input_dim' in kwargs:
            kwargs['input_shape'] = (kwargs.pop('input_dim'),)
        self.output_dim = output_dim
        self.initializer_centers = initializer_centers
        self.initializer_sigmas = initializer_sigmas
        super(FuzzyLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        self.input_dimensions = list(input_shape)[:-1:-1]
        self.c = self.add_weight(name='c',
                                 shape=(input_shape[-1], self.output_dim),
                                 initializer= self.initializer_centers if self.
↪initializer_centers is not None else 'uniform',
                                 trainable=True)
        self.a = self.add_weight(name='a',
                                 shape=(input_shape[-1], self.output_dim),
                                 initializer=self.initializer_sigmas if self.
↪initializer_sigmas is not None else 'ones',
                                 trainable=True)
        super(FuzzyLayer, self).build(input_shape)

    def call(self, x):

        aligned_x = K.repeat_elements(K.expand_dims(x, axis = -1), self.
↪output_dim, -1)
        aligned_c = self.c
        aligned_a = self.a
        for dim in self.input_dimensions:
            aligned_c = K.repeat_elements(K.expand_dims(aligned_c, 0), dim, 0)
            aligned_a = K.repeat_elements(K.expand_dims(aligned_a, 0), dim, 0)

        xc = K.exp(-K.sum(K.square((aligned_x - aligned_c) / (2 * aligned_a)),␣
↪axis=-2, keepdims=False))
        #sums = K.sum(xc,axis=-1,keepdims=True)
        #less = K.ones_like(sums) * K.epsilon()
        return xc# xc / K.maximum(sums, less)

    def compute_output_shape(self, input_shape):
        return tuple(input_shape[:-1]) + (self.output_dim,)
```

[ ]:

```
class Network:
    def __init__(self):

        input_img = Input(shape=(5000,))

        model = Dense(256, kernel_regularizer=regularizers.l1(0.0001),␣
↪activation='relu')(input_img)
```

```
        model = Dense(2,activation='relu')(model)
        f_layer = FuzzyLayer(100)
        model = f_layer(model)
        model = Dense(1, activation='linear')(model)
        model = Model(input_img, model)

        model.compile(optimizer='sgd', loss='mse',metrics=['acc'])
        self.model = model
        self.acc_history = []

    def return_acc_history(self):
        return self.acc_history

    def get_layer_weight(self,i):
        return self.model.layers[i].get_weights()

    def set_layer_weight(self,i,weight):
        self.model.layers[i].set_weights(weight)

    def train(self):
        # self.model.fit(X_train,y_train, batch_size = 32, epochs = 1, verbose = 1,shuffle = True) #, validation_data =(X_test, y_test)
        self.model.fit(x_train, y_train,
                epochs=1,
                verbose = 1,
                batch_size=32,
                shuffle=True,
                validation_data=(x_val, y_val))

    def test(self):
        loss, acc = self.model.evaluate(x_test,y_test)
        self.acc_history.append(acc)

        prediction = self.model.predict(x_test)
        y_pred = (prediction > 0.5)
        print('F1-score: {0}'.format(f1_score(y_pred, y_test)))
        print('Confusion matrix:', confusion_matrix(y_pred, y_test))
        return acc

    def load_layer_weights(self,weights):
        self.model.set_weights(weights)

    def give_weights(self):
        return self.model.get_weights()
    def weight_len(self):
        i = 0
        for j in self.model.layers:
            i+=1
        return i
    def architecture(self):
        self.model.summary()
```

[ ]:

```
class GeneticAlgorithm:
    def __init__(self, population_size, mutation_rate, generations = 50):
        self.population_size = population_size
        self.mutation_rate = mutation_rate
        self.generations = generations
        self.population = None
        self.children_population_weights = []
        self.acces = []
        self.norm_acces = []

    def create_population(self):
        self.population = [Network() for i in range(self.population_size)]

    def train_generation(self):
        for member in self.population:
                member.train()

    def predict(self):
        for member in self.population:
                acc = member.test()
                self.acc.append(acc)

    def normalize(self):
        sum_ = sum(self.acc)
        self.norm_acc = [i/sum_ for i in self.acc]
        print("\nNormalization sum: ",sum(self.norm_acc))
        #assert sum(self.norm_acc) == 1

    def show_weights(self):
        for i in parent_weights:
            print(i)
    def clear_losses(self):
        self.norm_acc = []
        self.acc = []

    def mutate(self):
        for member in self.population:
            for i in range(member.weight_len()):
                if np.random.random() < self.mutation_rate:
                    print("\Mutation at ", self.mutation_rate)
                    old_weight = member.get_layer_weight(i)
                    new_weight = [np.random.uniform(low=-1, high=1,␣
↪size=old_weight[i].shape) for i in range(len(old_weight))]
                    member.set_layer_weight(i, new_weight)

    def reproduction(self):
        """
        Reproduction through midpoint crossover method
        """
        population_idx = [i for i in range(len(self.population))]
        for i in range(len(self.population)):
        #selects two parents probabilistic accroding to the fitness
            # print("Crossover | Choosing Best Parent on the bases of fitness␣
↪probbaility\n")
            if sum(self.norm_acc) != 0:
                parent1 = np.random.choice(population_idx, p = self.norm_acc)
                parent2 = np.random.choice(population_idx, p = self.norm_acc)
            else:
              # if there are no "best" parents choose randomly
                parent1 = np.random.choice(population_idx)
                parent2 = np.random.choice(population_idx)

            # picking random midpoint for crossing over name/DNA
            parent1_weights = self.population[parent1].give_weights()
            parent2_weights = self.population[parent2].give_weights()


            mid_point = np.random.choice([i for i in␣
↪range(len(parent1_weights))])
            # adding DNA-Sequences of the parents to final DNA
            self.children_population_weights.append(parent1_weights[:mid_point]␣
↪+ parent2_weights[mid_point:])
        # old population gets the new and proper weights
        for i in range(len(self.population)):
            for j in range(len(self.children_population_weights)):
                self.population[i].load_layer_weights(self.
↪children_population_weights[j])

    def run_evolution(self):
        for episode in range(self.generations):
            self.clear_losses()
            self.train_generation()
            self.predict()
            if episode != self.generations -1:
                self.normalize()
                self.reproduction()
                self.mutate()
            else:
                pass

        # plotting history:
        for a in range(self.generations):
            for member in self.population:
                plt.plot(member.acc_history)
        plt.xlabel("No. of Generations")
        plt.ylabel("Model Accuracy")
        plt.show()
```

[ ]:
```
GA = GeneticAlgorithm(population_size = 4,mutation_rate = 0.05, generations = 20)
GA.create_population()
GA.run_evolution()
```

```
313/313 [==============================] - 5s 7ms/step - loss: 2.6481 - acc:
0.5253 - val_loss: 2.3353 - val_acc: 0.6595
313/313 [==============================] - 3s 6ms/step - loss: 2.4499 - acc:
0.5040 - val_loss: 2.3771 - val_acc: 0.5620
313/313 [==============================] - 2s 6ms/step - loss: 2.4554 - acc:
0.5146 - val_loss: 2.3440 - val_acc: 0.6485
313/313 [==============================] - 3s 6ms/step - loss: 2.4519 - acc:
0.5264 - val_loss: 2.3498 - val_acc: 0.6295
313/313 [==============================] - 1s 3ms/step - loss: 2.3343 - acc:
0.6682
F1-score: 0.6894421564956945
Confusion matrix: [[2999 1290]
 [2028 3683]]
313/313 [==============================] - 1s 3ms/step - loss: 2.3789 - acc:
0.5538
F1-score: 0.6708468574800827
Confusion matrix: [[ 991  426]
 [4036 4547]]
313/313 [==============================] - 1s 3ms/step - loss: 2.3434 - acc:
0.6468
F1-score: 0.6909345467273363
Confusion matrix: [[2520 1025]
 [2507 3948]]
313/313 [==============================] - 1s 3ms/step - loss: 2.3484 - acc:
0.6286
F1-score: 0.5314155942467828
Confusion matrix: [[4180 2867]
 [ 847 2106]]

Normalization sum:  1.0
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 2.4328 - acc:
```

```
0.4987 - val_loss: 2.3316 - val_acc: 0.4900
313/313 [==============================] - 2s 6ms/step - loss: 2.2838 - acc:
0.7303 - val_loss: 2.2241 - val_acc: 0.8165
313/313 [==============================] - 2s 6ms/step - loss: 2.2848 - acc:
0.7329 - val_loss: 2.2244 - val_acc: 0.8120
313/313 [==============================] - 2s 6ms/step - loss: 2.2829 - acc:
0.7332 - val_loss: 2.2525 - val_acc: 0.7335
313/313 [==============================] - 1s 3ms/step - loss: 2.3333 - acc:
0.5045
F1-score: 0.02976307029567261
Confusion matrix: [[4969 4897]
 [  58   76]]
313/313 [==============================] - 1s 3ms/step - loss: 2.2345 - acc:
0.7868
F1-score: 0.7849939491730538
Confusion matrix: [[3976 1081]
 [1051 3892]]
313/313 [==============================] - 1s 3ms/step - loss: 2.2353 - acc:
0.7826
F1-score: 0.7908811081185071
Confusion matrix: [[3715  862]
 [1312 4111]]
313/313 [==============================] - 1s 3ms/step - loss: 2.2590 - acc:
0.7254
F1-score: 0.654851684263449
Confusion matrix: [[4649 2368]
 [ 378 2605]]

Normalization sum:  1.0
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 2.1909 - acc:
0.8246 - val_loss: 2.1437 - val_acc: 0.8675
313/313 [==============================] - 2s 6ms/step - loss: 2.1987 - acc:
0.8143 - val_loss: 2.1602 - val_acc: 0.8525
313/313 [==============================] - 2s 6ms/step - loss: 2.1964 - acc:
0.8188 - val_loss: 2.1536 - val_acc: 0.8670
313/313 [==============================] - 2s 6ms/step - loss: 2.1973 - acc:
0.8171 - val_loss: 2.1544 - val_acc: 0.8610
313/313 [==============================] - 1s 3ms/step - loss: 2.1725 - acc:
0.8161
F1-score: 0.8052113123609788
Confusion matrix: [[4360 1172]
 [ 667 3801]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1811 - acc:
0.8103
F1-score: 0.8234856238950404
Confusion matrix: [[3678  548]
```

```
 [1349 4425]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1735 - acc:
0.8221
F1-score: 0.8236343808862892
Confusion matrix: [[4067  819]
 [ 960 4154]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1742 - acc:
0.8176
F1-score: 0.8214565387627252
Confusion matrix: [[3980  777]
 [1047 4196]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 2.1386 - acc:
0.8590 - val_loss: 2.1005 - val_acc: 0.8950
313/313 [==============================] - 2s 6ms/step - loss: 2.1409 - acc:
0.8550 - val_loss: 2.1012 - val_acc: 0.8935
313/313 [==============================] - 2s 6ms/step - loss: 2.2727 - acc:
0.5893 - val_loss: 2.1432 - val_acc: 0.8165
313/313 [==============================] - 2s 6ms/step - loss: 2.1403 - acc:
0.8589 - val_loss: 2.1570 - val_acc: 0.7635
313/313 [==============================] - 1s 3ms/step - loss: 2.1286 - acc:
0.8366
F1-score: 0.8306384742951907
Confusion matrix: [[4359  966]
 [ 668 4007]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1296 - acc:
0.8344
F1-score: 0.8265605362379557
Confusion matrix: [[4398 1027]
 [ 629 3946]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1598 - acc:
0.7887
F1-score: 0.797740978271274
Confusion matrix: [[3720  806]
 [1307 4167]]
313/313 [==============================] - 1s 3ms/step - loss: 2.1823 - acc:
0.7406
F1-score: 0.661446097624641
Confusion matrix: [[4872 2439]
 [ 155 2534]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 2.0871 - acc:
0.8806 - val_loss: 2.0530 - val_acc: 0.9130
313/313 [==============================] - 2s 6ms/step - loss: 2.1481 - acc:
```

```
0.7730 - val_loss: 2.0517 - val_acc: 0.8890
313/313 [==============================] - 2s 6ms/step - loss: 2.0873 - acc:
0.8796 - val_loss: 2.0528 - val_acc: 0.9120
313/313 [==============================] - 2s 6ms/step - loss: 2.0898 - acc:
0.8784 - val_loss: 2.0593 - val_acc: 0.8980
313/313 [==============================] - 1s 3ms/step - loss: 2.0873 - acc:
0.8437
F1-score: 0.8365575656174841
Confusion matrix: [[4437  973]
 [ 590 4000]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0880 - acc:
0.8264
F1-score: 0.8187891440501044
Confusion matrix: [[4342 1051]
 [ 685 3922]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0868 - acc:
0.8425
F1-score: 0.8363636363636363
Confusion matrix: [[4400  948]
 [ 627 4025]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0947 - acc:
0.8309
F1-score: 0.8442766368910581
Confusion matrix: [[3725  389]
 [1302 4584]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 2.0410 - acc:
0.8972 - val_loss: 2.0140 - val_acc: 0.9105
313/313 [==============================] - 2s 6ms/step - loss: 2.0404 - acc:
0.8965 - val_loss: 2.0110 - val_acc: 0.9185
313/313 [==============================] - 2s 6ms/step - loss: 2.0331 - acc:
0.8929 - val_loss: 2.0174 - val_acc: 0.8830
313/313 [==============================] - 2s 6ms/step - loss: 2.0416 - acc:
0.8946 - val_loss: 2.0055 - val_acc: 0.9280
313/313 [==============================] - 1s 3ms/step - loss: 2.0542 - acc:
0.8365
F1-score: 0.8492392807745505
Confusion matrix: [[3760  368]
 [1267 4605]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0492 - acc:
0.8432
F1-score: 0.8331204767986377
Confusion matrix: [[4518 1059]
 [ 509 3914]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0667 - acc:
0.8184
```

```
F1-score: 0.8391211906449326
Confusion matrix: [[3448  237]
 [1579 4736]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0445 - acc:
0.8491
F1-score: 0.848295968633759
Confusion matrix: [[4272  754]
 [ 755 4219]]

Normalization sum:  0.9999999999999999
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 64.2902 - acc:
0.5040 - val_loss: 64.2351 - val_acc: 0.5115
313/313 [==============================] - 2s 6ms/step - loss: 1.9955 - acc:
0.9089 - val_loss: 2.0019 - val_acc: 0.8625
313/313 [==============================] - 2s 6ms/step - loss: 1.9960 - acc:
0.9069 - val_loss: 1.9758 - val_acc: 0.9170
313/313 [==============================] - 2s 6ms/step - loss: 1.9955 - acc:
0.9086 - val_loss: 1.9613 - val_acc: 0.9400
313/313 [==============================] - 1s 3ms/step - loss: 64.2353 - acc:
0.4973
F1-score: 0.6642623388766447
Confusion matrix: [[   0    0]
 [5027 4973]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0435 - acc:
0.7885
F1-score: 0.7443490873927233
Confusion matrix: [[4806 1894]
 [ 221 3079]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0181 - acc:
0.8314
F1-score: 0.8141944015869517
Confusion matrix: [[4620 1279]
 [ 407 3694]]
313/313 [==============================] - 1s 3ms/step - loss: 2.0050 - acc:
0.8531
F1-score: 0.8548849155388718
Confusion matrix: [[4204  646]
 [ 823 4327]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.9511 - acc:
0.9208 - val_loss: 1.9223 - val_acc: 0.9425
313/313 [==============================] - 2s 6ms/step - loss: 1.9520 - acc:
0.9195 - val_loss: 1.9252 - val_acc: 0.9365
313/313 [==============================] - 2s 6ms/step - loss: 1.9504 - acc:
```

```
0.9193 - val_loss: 1.9379 - val_acc: 0.9100
313/313 [==============================] - 2s 6ms/step - loss: 1.9833 - acc:
0.8639 - val_loss: 1.9363 - val_acc: 0.9070
313/313 [==============================] - 1s 3ms/step - loss: 1.9703 - acc:
0.8488
F1-score: 0.8413764162819973
Confusion matrix: [[4478  963]
 [ 549 4010]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9726 - acc:
0.8458
F1-score: 0.835817717206133
Confusion matrix: [[4533 1048]
 [ 494 3925]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9866 - acc:
0.8301
F1-score: 0.8471159902816522
Confusion matrix: [[3594  266]
 [1433 4707]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9850 - acc:
0.8181
F1-score: 0.7936940002268346
Confusion matrix: [[4682 1474]
 [ 345 3499]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.9106 - acc:
0.9277 - val_loss: 1.8822 - val_acc: 0.9425
313/313 [==============================] - 2s 6ms/step - loss: 1.9097 - acc:
0.9278 - val_loss: 1.8780 - val_acc: 0.9565
313/313 [==============================] - 2s 6ms/step - loss: 1.9121 - acc:
0.9231 - val_loss: 1.8914 - val_acc: 0.9260
313/313 [==============================] - 2s 6ms/step - loss: 1.9128 - acc:
0.9247 - val_loss: 1.8821 - val_acc: 0.9470
313/313 [==============================] - 1s 3ms/step - loss: 1.9336 - acc:
0.8534
F1-score: 0.8598470363288718
Confusion matrix: [[4037  476]
 [ 990 4497]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9291 - acc:
0.8547
F1-score: 0.8547435769269219
Confusion matrix: [[4272  698]
 [ 755 4275]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9430 - acc:
0.8396
F1-score: 0.8527900146842878
Confusion matrix: [[3750  327]
 [1277 4646]]
```

```
313/313 [==============================] - 1s 3ms/step - loss: 1.9327 - acc:
0.8505
F1-score: 0.8431105047748977
Confusion matrix: [[4488  956]
 [ 539 4017]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.8687 - acc:
0.9339 - val_loss: 1.8487 - val_acc: 0.9350
313/313 [==============================] - 2s 6ms/step - loss: 1.8696 - acc:
0.9350 - val_loss: 1.8375 - val_acc: 0.9620
313/313 [==============================] - 2s 6ms/step - loss: 1.8693 - acc:
0.9349 - val_loss: 1.8749 - val_acc: 0.8975
313/313 [==============================] - 2s 6ms/step - loss: 1.8684 - acc:
0.9345 - val_loss: 1.8418 - val_acc: 0.9485
313/313 [==============================] - 1s 3ms/step - loss: 1.9032 - acc:
0.8406
F1-score: 0.8526802218114603
Confusion matrix: [[3793  360]
 [1234 4613]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8925 - acc:
0.8574
F1-score: 0.8575424575424576
Confusion matrix: [[4282  681]
 [ 745 4292]]
313/313 [==============================] - 1s 3ms/step - loss: 1.9275 - acc:
0.8006
F1-score: 0.763407688656858
Confusion matrix: [[4789 1756]
 [ 238 3217]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8971 - acc:
0.8522
F1-score: 0.8589963747376455
Confusion matrix: [[4020  471]
 [1007 4502]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.8278 - acc:
0.9418 - val_loss: 1.8301 - val_acc: 0.9095
313/313 [==============================] - 2s 6ms/step - loss: 1.8278 - acc:
0.9423 - val_loss: 1.8061 - val_acc: 0.9500
313/313 [==============================] - 2s 6ms/step - loss: 1.8279 - acc:
0.9433 - val_loss: 1.7994 - val_acc: 0.9620
313/313 [==============================] - 2s 6ms/step - loss: 1.8277 - acc:
0.9431 - val_loss: 1.8041 - val_acc: 0.9550
313/313 [==============================] - 1s 3ms/step - loss: 1.8880 - acc:
0.8071
F1-score: 0.7739894551845342
```

```
Confusion matrix: [[4768 1670]
 [ 259 3303]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8642 - acc:
0.8440
F1-score: 0.8324022346368715
Confusion matrix: [[4566 1099]
 [ 461 3874]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8584 - acc:
0.8550
F1-score: 0.8598763045999227
Confusion matrix: [[4101  524]
 [ 926 4449]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8624 - acc:
0.8481
F1-score: 0.8387302261386558
Confusion matrix: [[4531 1023]
 [ 496 3950]]

Normalization sum:  0.9999999999999999
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.7879 - acc:
0.9496 - val_loss: 1.7602 - val_acc: 0.9660
313/313 [==============================] - 2s 6ms/step - loss: 1.7883 - acc:
0.9475 - val_loss: 1.7662 - val_acc: 0.9585
313/313 [==============================] - 2s 6ms/step - loss: 1.7886 - acc:
0.9497 - val_loss: 1.7588 - val_acc: 0.9690
313/313 [==============================] - 2s 6ms/step - loss: 1.8100 - acc:
0.9328 - val_loss: 1.7801 - val_acc: 0.9320
313/313 [==============================] - 1s 3ms/step - loss: 1.8225 - acc:
0.8573
F1-score: 0.8621921776919363
Confusion matrix: [[4109  509]
 [ 918 4464]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8281 - acc:
0.8459
F1-score: 0.8343188904418881
Confusion matrix: [[4579 1093]
 [ 448 3880]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8211 - acc:
0.8577
F1-score: 0.8601199252924406
Confusion matrix: [[4202  598]
 [ 825 4375]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8444 - acc:
0.8148
F1-score: 0.7885844748858448
Confusion matrix: [[4694 1519]
```

```
 [ 333 3454]]

Normalization sum:  0.9999999999999999
\Mutation at  0.05
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.7605 - acc:
0.9385 - val_loss: 1.7286 - val_acc: 0.9645
313/313 [==============================] - 2s 6ms/step - loss: 1.8281 - acc:
0.8836 - val_loss: 1.7219 - val_acc: 0.9610
313/313 [==============================] - 2s 6ms/step - loss: 1.7598 - acc:
0.9418 - val_loss: 1.7456 - val_acc: 0.9360
313/313 [==============================] - 2s 6ms/step - loss: 64.2296 - acc:
0.5034 - val_loss: 64.1817 - val_acc: 0.4950
313/313 [==============================] - 1s 3ms/step - loss: 1.7948 - acc:
0.8425
F1-score: 0.8303715670436188
Confusion matrix: [[4570 1118]
 [ 457 3855]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7844 - acc:
0.8505
F1-score: 0.8441571979568435
Confusion matrix: [[4456  924]
 [ 571 4049]]
313/313 [==============================] - 1s 3ms/step - loss: 1.8120 - acc:
0.8138
F1-score: 0.7858785648574056
Confusion matrix: [[4721 1556]
 [ 306 3417]]
313/313 [==============================] - 1s 3ms/step - loss: 64.1835 - acc:
0.5064
F1-score: 0.16677920324105333
Confusion matrix: [[4570 4479]
 [ 457  494]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.7077 - acc:
0.9491 - val_loss: 1.6839 - val_acc: 0.9580
313/313 [==============================] - 2s 6ms/step - loss: 1.7087 - acc:
0.9488 - val_loss: 1.6775 - val_acc: 0.9720
313/313 [==============================] - 2s 6ms/step - loss: 64.2403 - acc:
0.4950 - val_loss: 64.2155 - val_acc: 0.5115
313/313 [==============================] - 2s 6ms/step - loss: 1.7088 - acc:
0.9482 - val_loss: 1.6828 - val_acc: 0.9600
313/313 [==============================] - 1s 3ms/step - loss: 1.7529 - acc:
0.8528
F1-score: 0.8608695652173912
```

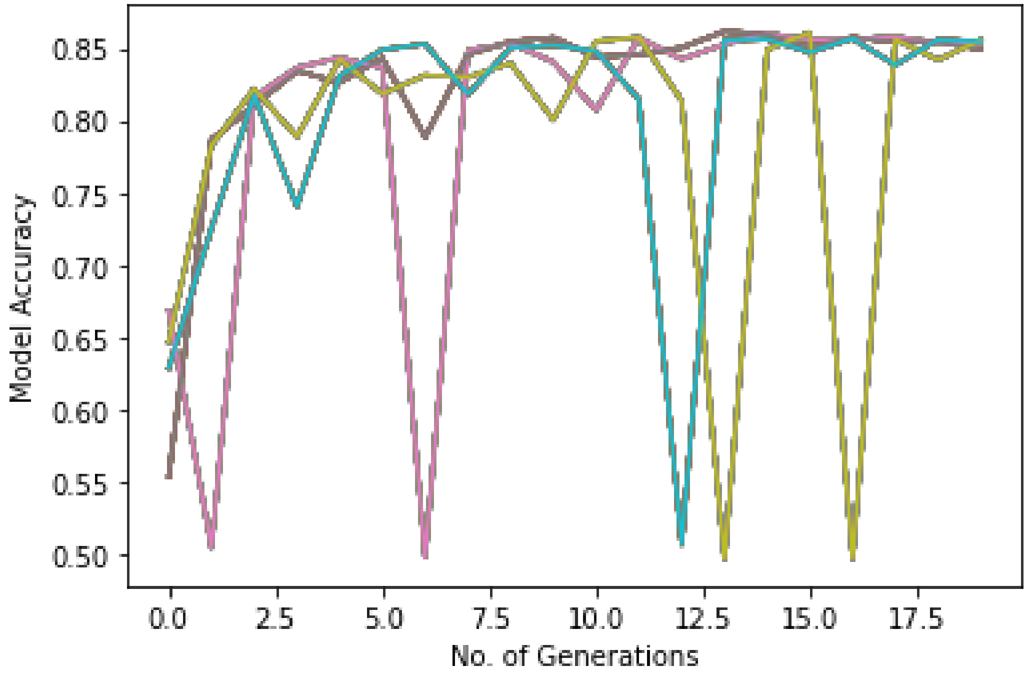
```
Confusion matrix: [[3974  419]
 [1053 4554]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7453 - acc:
0.8618
F1-score: 0.8637886851961365
Confusion matrix: [[4236  591]
 [ 791 4382]]
313/313 [==============================] - 1s 3ms/step - loss: 64.2160 - acc:
0.4962
F1-score: 0.6625586068318822
Confusion matrix: [[  16   27]
 [5011 4946]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7504 - acc:
0.8558
F1-score: 0.8622731614135626
Confusion matrix: [[4044  459]
 [ 983 4514]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.6669 - acc:
0.9591 - val_loss: 1.6376 - val_acc: 0.9760
313/313 [==============================] - 2s 6ms/step - loss: 1.6669 - acc:
0.9590 - val_loss: 1.6385 - val_acc: 0.9760
313/313 [==============================] - 2s 6ms/step - loss: 1.6663 - acc:
0.9583 - val_loss: 1.6479 - val_acc: 0.9515
313/313 [==============================] - 2s 6ms/step - loss: 1.6664 - acc:
0.9600 - val_loss: 1.6403 - val_acc: 0.9755
313/313 [==============================] - 1s 3ms/step - loss: 1.7103 - acc:
0.8603
F1-score: 0.8600060126265158
Confusion matrix: [[4312  682]
 [ 715 4291]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7103 - acc:
0.8597
F1-score: 0.8611578426521523
Confusion matrix: [[4246  622]
 [ 781 4351]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7201 - acc:
0.8491
F1-score: 0.8592744567751562
Confusion matrix: [[3884  366]
 [1143 4607]]
313/313 [==============================] - 1s 3ms/step - loss: 1.7137 - acc:
0.8563
F1-score: 0.8509799854816965
Confusion matrix: [[4460  870]
 [ 567 4103]]
```

```
Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.6276 - acc:
0.9679 - val_loss: 1.6065 - val_acc: 0.9690
313/313 [==============================] - 2s 6ms/step - loss: 1.6273 - acc:
0.9668 - val_loss: 1.6049 - val_acc: 0.9775
313/313 [==============================] - 2s 6ms/step - loss: 1.6280 - acc:
0.9658 - val_loss: 1.6002 - val_acc: 0.9800
313/313 [==============================] - 2s 6ms/step - loss: 1.6275 - acc:
0.9673 - val_loss: 1.6086 - val_acc: 0.9735
313/313 [==============================] - 1s 3ms/step - loss: 1.6826 - acc:
0.8561
F1-score: 0.8631999239471434
Confusion matrix: [[4021  433]
 [1006 4540]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6819 - acc:
0.8511
F1-score: 0.8426503223079362
Confusion matrix: [[4524  986]
 [ 503 3987]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6766 - acc:
0.8610
F1-score: 0.860413737698333
Confusion matrix: [[4326  689]
 [ 701 4284]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6864 - acc:
0.8473
F1-score: 0.8364220674879486
Confusion matrix: [[4569 1069]
 [ 458 3904]]

Normalization sum:  1.0
\Mutation at  0.05
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.6158 - acc:
0.9631 - val_loss: 1.5792 - val_acc: 0.9835
313/313 [==============================] - 2s 6ms/step - loss: 1.5892 - acc:
0.9734 - val_loss: 1.5641 - val_acc: 0.9795
313/313 [==============================] - 2s 6ms/step - loss: 64.2501 - acc:
0.4956 - val_loss: 64.2258 - val_acc: 0.5110
313/313 [==============================] - 2s 6ms/step - loss: 1.5897 - acc:
0.9721 - val_loss: 1.5653 - val_acc: 0.9825
313/313 [==============================] - 1s 3ms/step - loss: 1.6493 - acc:
0.8564
F1-score: 0.8553294378400161
Confusion matrix: [[4319  728]
 [ 708 4245]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6438 - acc:
0.8577
```

```
F1-score: 0.8608040692556002
Confusion matrix: [[4177  573]
 [ 850 4400]]
313/313 [==============================] - 1s 3ms/step - loss: 64.2260 - acc:
0.4967
F1-score: 0.6633669988629523
Confusion matrix: [[   8   14]
 [5019 4959]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6454 - acc:
0.8567
F1-score: 0.8512096355518638
Confusion matrix: [[4468  874]
 [ 559 4099]]

Normalization sum:  1.0
\Mutation at  0.05
313/313 [==============================] - 2s 6ms/step - loss: 1.5554 - acc:
0.9738 - val_loss: 1.5290 - val_acc: 0.9870
313/313 [==============================] - 2s 6ms/step - loss: 1.5558 - acc:
0.9742 - val_loss: 1.5335 - val_acc: 0.9785
313/313 [==============================] - 2s 6ms/step - loss: 1.5555 - acc:
0.9738 - val_loss: 1.5320 - val_acc: 0.9820
313/313 [==============================] - 2s 6ms/step - loss: 1.5550 - acc:
0.9739 - val_loss: 1.5439 - val_acc: 0.9745
313/313 [==============================] - 1s 3ms/step - loss: 1.6096 - acc:
0.8581
F1-score: 0.8573726002613328
Confusion matrix: [[4316  708]
 [ 711 4265]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6145 - acc:
0.8544
F1-score: 0.8610952108376263
Confusion matrix: [[4031  460]
 [ 996 4513]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6138 - acc:
0.8561
F1-score: 0.8621515470830539
Confusion matrix: [[4061  473]
 [ 966 4500]]
313/313 [==============================] - 1s 3ms/step - loss: 1.6277 - acc:
0.8385
F1-score: 0.8225859606723057
Confusion matrix: [[4641 1229]
 [ 386 3744]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.5171 - acc:
0.9816 - val_loss: 1.4929 - val_acc: 0.9885
```

```
313/313 [==============================] - 2s 6ms/step - loss: 1.5172 - acc:
0.9800 - val_loss: 1.4953 - val_acc: 0.9865
313/313 [==============================] - 2s 6ms/step - loss: 1.5173 - acc:
0.9797 - val_loss: 1.5055 - val_acc: 0.9775
313/313 [==============================] - 2s 6ms/step - loss: 1.5171 - acc:
0.9809 - val_loss: 1.4934 - val_acc: 0.9900
313/313 [==============================] - 1s 3ms/step - loss: 1.5780 - acc:
0.8539
F1-score: 0.8508727161375932
Confusion matrix: [[4371  805]
 [ 656 4168]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5808 - acc:
0.8549
F1-score: 0.848522810314229
Confusion matrix: [[4485  909]
 [ 542 4064]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5923 - acc:
0.8420
F1-score: 0.8276614310645725
Confusion matrix: [[4626 1179]
 [ 401 3794]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5780 - acc:
0.8561
F1-score: 0.8529982633568289
Confusion matrix: [[4386  798]
 [ 641 4175]]

Normalization sum:  1.0
313/313 [==============================] - 2s 6ms/step - loss: 1.4814 - acc:
0.9846 - val_loss: 1.4657 - val_acc: 0.9835
313/313 [==============================] - 2s 6ms/step - loss: 1.4816 - acc:
0.9832 - val_loss: 1.4690 - val_acc: 0.9795
313/313 [==============================] - 2s 6ms/step - loss: 1.4814 - acc:
0.9835 - val_loss: 1.4586 - val_acc: 0.9920
313/313 [==============================] - 2s 6ms/step - loss: 1.4813 - acc:
0.9845 - val_loss: 1.4585 - val_acc: 0.9900
313/313 [==============================] - 1s 3ms/step - loss: 1.5542 - acc:
0.8535
F1-score: 0.8612032212221695
Confusion matrix: [[3990  428]
 [1037 4545]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5565 - acc:
0.8491
F1-score: 0.8594317652538426
Confusion matrix: [[3878  360]
 [1149 4613]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5459 - acc:
0.8558
```

```
F1-score: 0.8581267217630855
Confusion matrix: [[4197  612]
 [ 830 4361]]
313/313 [==============================] - 1s 3ms/step - loss: 1.5466 - acc:
0.8547
F1-score: 0.8513859056970441
Confusion matrix: [[4385  811]
 [ 642 4162]]
```



[1]:
```
import os
os.chdir('/content/drive/My Drive/')
!pwd
```

```
/content/drive/My Drive
```

[2]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-generic-recommended
```

```
Processing triggers for fontconfig (2.12.6-0ubuntu2) ...
Processing triggers for tex-common (6.09) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
76.)
```

```
debconf: falling back to frontend: Readline
Running updmap-sys. This may take some time... done.
Running mktexlsr /var/lib/texmf ... done.
Building format(s) --all.
        This may take some time... done.
```

[8]:
```
!jupyter nbconvert --to pdf GA_2_HNFG_IMDB.ipynb
```

```
[NbConvertApp] WARNING | pattern u'GA_2_HNFG_IMDB.ipynb' matched no files
This application is used to convert notebook files (*.ipynb) to various other
formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
-------

Arguments that take values are actually convenience aliases to full
Configurables, whose aliases are listed on the help line. For more information
on full configurables, see '--help-all'.

--execute
    Execute the notebook prior to export.
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
--no-input
    Exclude input cells and output prompts from converted document.
    This mode is ideal for generating code-free reports.
--stdout
    Write notebook output to stdout instead of files.
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
    relevant when converting to notebook format)
-y
    Answer yes to any questions instead of prompting.
--clear-output
    Clear output of current file and save in place,
    overwriting the existing notebook.
--debug
    set log level to logging.DEBUG (maximize logging output)
--no-prompt
    Exclude input and output prompts from converted document.
```

```
--generate-config
    generate default config file
--nbformat=<Enum> (NotebookExporter.nbformat_version)
    Default: 4
    Choices: [1, 2, 3, 4]
    The nbformat version to write. Use this to downgrade notebooks.
--output-dir=<Unicode> (FilesWriter.build_directory)
    Default: ''
    Directory to write output(s) to. Defaults to output to the directory of each
    notebook. To recover previous default behaviour (outputting to the current
    working directory) use . as the flag value.
--writer=<DottedObjectName> (NbConvertApp.writer_class)
    Default: 'FilesWriter'
    Writer class used to write the  results of the conversion
--log-level=<Enum> (Application.log_level)
    Default: 30
    Choices: (0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL')
    Set the log level by value or name.
--reveal-prefix=<Unicode> (SlidesExporter.reveal_url_prefix)
    Default: u''
    The URL prefix for reveal.js (version 3.x). This defaults to the reveal CDN,
    but can be any url pointing to a copy  of reveal.js.
    For speaker notes to work, this must be a relative path to a local  copy of
    reveal.js: e.g., "reveal.js".
    If a relative path is given, it must be a subdirectory of the current
    directory (from which the server is run).
    See the usage documentation
    (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-
    slideshow) for more details.
--to=<Unicode> (NbConvertApp.export_format)
    Default: 'html'
    The export format to be used, either one of the built-in formats
    ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf',
    'python', 'rst', 'script', 'slides'] or a dotted object name that represents
    the import path for an `Exporter` class
--template=<Unicode> (TemplateExporter.template_file)
    Default: u''
    Name of the template file to use
--output=<Unicode> (NbConvertApp.output_base)
    Default: ''
    overwrite base name use for output files. can only be used when converting
    one notebook at a time.
--post=<DottedOrNone> (NbConvertApp.postprocessor_class)
    Default: u''
    PostProcessor class used to write the results of the conversion
--config=<Unicode> (JupyterApp.config_file)
    Default: u''
    Full path of a config file.

To see all available configurables, use `--help-all`

Examples
--------

    The simplest way to use nbconvert is

    > jupyter nbconvert mynotebook.ipynb

    which will convert mynotebook.ipynb to the default format (probably HTML).

    You can specify the export format with `--to`.
    Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

    > jupyter nbconvert --to latex mynotebook.ipynb

    Both HTML and LaTeX support multiple output templates. LaTeX includes
    'base', 'article' and 'report'.  HTML includes 'basic' and 'full'. You
    can specify the flavor of the format used.

    > jupyter nbconvert --to html --template basic mynotebook.ipynb

    You can also pipe the output to stdout, rather than a file

    > jupyter nbconvert mynotebook.ipynb --stdout

    PDF is generated via latex

    > jupyter nbconvert mynotebook.ipynb --to pdf

    You can get (and serve) a Reveal.js-powered slideshow

    > jupyter nbconvert myslides.ipynb --to slides --post serve

    Multiple notebooks can be given at the command line in a couple of
    different ways:

    > jupyter nbconvert notebook*.ipynb
    > jupyter nbconvert notebook1.ipynb notebook2.ipynb

    or you can specify the notebooks list in a config file, containing::

        c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

    > jupyter nbconvert --config mycfg.py
```

[ ]:

```

```