Deciding the Best Artificial Neural Network Architecture

John Smutny

4/15/2022

HW5

ECE-5984: Applications of Machine Learning

Abstract

This report details the methodology and results to decide what the best architecture is to produce the best Artificial Neural Network (ANN). The two performance criteria used to judge the best ANN architecture in this report are 1) the best MisClassification Rate (MCR) model and 2) the best Area Under the Receiver Operating Characteristic (AUROC) curve model. To find the best MCR model and the best AUROC model; a python script utilizing scikit-learn's neural_network module trained and evaluated every combination of a range of metrics. A total of 4440 models were trained by varying the model's number of hidden layers, number of nodes, and the activation function used at each node. Normalization was the only data preparation on the used data, because of the processed Data Quality Report.

After hours of iterating through each model architecture combination, the best performing models were determined. The best MCR model architecture had 3 Hidden Layers with {2, 8, 6} nodes in each respective layer and a tanh activation function.The best AUROC model architecture had 2 Hidden Layers with {1, 8, 0} nodes in each respective layer and an identity activation function.

These best model architectures were decided given that the model Learning Rate was 0.001, Tolerance was 0.001, and Max Iterations (epochs) available were set to 10000. Three combinations of Learning Rate and Tolerance were tested, all had similar results.

Methodology

The quantities used to define a single ANN model are: the activation function used at each node, the number of hidden layers, the number of nodes in each hidden layer, the activation function used at each node, the learning rate for each training epoch, the tolerance value to stop the model's iterative refinement of weights, and the maximum number of iterations (epochs). Normalization was the only data preparation step since the DataQualityReport showed valid data qualities. The range of architect quantities tested are described below in Table1.

ANN Architecture Quality	**Constant or Value Range []**
Activation Function	[relu, logistic, identity, tanh]
# of Hidden Layers	[1 - 3]
# of Nodes per Hidden Layer	[1 - 10]
Learning Rate	0.001
Tolerance	0.001
Max Iterations (epochs)	10000

Table 1: ANN architecture qualities used to determine the best models for each metric.

The script used to train and evaluate each model iterated through each possible combination of characteristics by using nested *for* loops and a base-10 counter. Once the activation function and structural number of hidden layers a model should use, the counter goes from 1 to the mathematical number of node combinations available based on the number of hidden layers and max nodes in each layer. This loop structure is shown below.

```
For (each possible activationFunction)
      For (each possible amount of Hidden Layers)
            # Now the base structure is created. All that is needed is
            # to create individual models with numeric combinations of
            # nodes.
            For (the numeric combinations of nodes in each layer)
                  Assign #nodes in hl1
                  Assign #nodes in hl2 if necessary
                  Assign #nodes in hl3 if necessary
                  …
                  Train model
                  Determine MCR & AUROC
```

Results

The architectures that had the best (lowest) and worst (highest) AUROC and MCR scores are shown below in their respective tables. The tables show the variable architecture quantities that yielded those performance metrics. In the appendix, the Top-10 Best and Worst model architectures for both performance metrics are defined (see Table A2-A5).

AUROC

Given the model combinations as defined in Table1 above. The best AUROC model architecture had 2 Hidden Layers with {1, 8, 0} nodes in each respective layer and an identity activation function. The worst AUROC model architecture had 3 Hidden Layers with {2, 8, 6} nodes in each respective layer and a tanh activation function.

Rank	**Activation Function**	**# of Hidden Layers**	**# of Nodes (HL1)**	**# of Nodes (HL2)**	**# of Nodes (HL3)**	**AUROC**	**MCR**
BEST	identity	2	1	8	0	0.460774	0.509927
WORST	tanh	3	2	8	6	0.958724	0.041275

Table 2: The best and worst ANN models evaluated for AUROC.

MCR

Given the model combinations as defined in Table1 above.The best MCR model architecture had 3 Hidden Layers with {2, 8, 6} nodes in each respective layer and a tanh activation function. The worst MCR model architecture had 2 Hidden Layers with {2, 1, 0} nodes in each respective layer and also a tanh activation function.

Rank	**Activation Function**	**# of Hidden Layers**	**# of Nodes (HL1)**	**# of Nodes (HL2)**	**# of Nodes (HL3)**	**AUROC**	**MCR**
BEST	tanh	3	2	8	6	0.958724	0.041275
WORST	tanh	2	2	1	0	0.5	0.53187

Table 3: The best and worst ANN models evaluated for MCR.

Discussion & Observations

After several hours of training each model, it became clear that the performance criteria used to judge each model was the biggest factor in what model was considered the “best”. By looking at the Top-10 best and worst models for AUROC and MCR, the architectures appeared to be exact opposites of each other based on the top model’s activation function used and how many nodes overall to use. The best AUROC model is related in node count to the worst MCR, and the best MCR is the worst AUROC model.

If AUROC curve is the desired performance metric, then the best models had a variety (but majority had the logistic) activation functions with a variable node structure. There was a trend to have a similar number of nodes in each layer, but it was not consistent among the best models. Having two (not three) hidden layers was not present in the MCR top models. On the other side of the AUROC performance curve; the worst AUROC models all used the tanh or identity activation function. Most notably, the worst AUROC model was the best MCR model.

If MCR was the desired performance metric, then the best models mostly used the tanh or identity activation function, mostly with three layers, and a variety of node structures like the AUROC models. The worst models most commonly used the logistic or relu activation function and tended to have the least amount of nodes in the third layer.

One other strange trend in both performance metrics and both best/worst performing models was a discrete nature of the resulting AUROC and MCR values. There are a variety of magnitudes for both with 6 significant figures, but it is surprising that numerous Top-10 models for both criteria had identical values for their performance metric. It would be expected that after training 4000+ models on a real dataset that the outcomes would be more varied. This would imply that there is eventually a bound that these criteria could reach.

Appendix

Dataset Used

1. ccpp.xlsx
    a. Provider: Dr Creed Jones, Virginia Tech

Data Quality Report

stat	**ID**	**AT**	**V**	**AP**	**RH**	**TG**
count	9568	9568	9568	9568	9568	9568
cardinality	9568	2612	623	2348	4034	2
mean	4784.5	19.58345	54.25371	1013.349	73.39545	0.524561
median	4784.5	20.32	52.065	1013.06	75.15	1
n_at_median	N/A	7	N/A	8	3	5019
mode	1	25.21	41.17	1013.88	100.09	1
n_at_mode	1	16	60	19	22	5019
stddev	2762.188	7.444781	12.67577	5.924366	14.5123	0.499422
min	1	1.81	25.36	993.31	25.89	0
max	9568	37.11	81.56	1033.3	100.16	1
n_zero	N/A	N/A	N/A	N/A	N/A	4549
n_missing	0	0	0	0	0	0

Table A1: Data Quality Report of the used ‘ccpp.xlsx’ dataset.

Top-10 Model Architectures (Best & Worst)

10 Best AUROC Architectures

Rank	Activation Function	# of Hidden Layers	# of Nodes (HL1)	# of Nodes (HL2)	# of Nodes (HL3)	AUROC	MCR
BEST	identity	2	1	8	0	0.460774	0.509927
2	logistic	3	4	8	10	0.496829	0.471787
3	tanh	3	4	10	1	0.499509	0.468652
4	relu	3	1	3	6	0.499509	0.468652
5	logistic	3	6	4	4	0.5	0.46813
6	logistic	3	5	4	4	0.5	0.46813
7	logistic	3	4	4	4	0.5	0.46813
8	logistic	3	3	4	4	0.5	0.46813
9	logistic	3	7	3	4	0.5	0.46813
10	logistic	3	2	4	4	0.5	0.46813

Table A2: Top-10 model architectures based on AUROC value.

10 Worst AUROC Architectures

Rank	Activation Function	# of Hidden Layers	# of Nodes (HL1)	# of Nodes (HL2)	# of Nodes (HL3)	AUROC	MCR
1	tanh	3	8	7	9	0.957451	0.042842
2	identity	3	8	8	2	0.957518	0.042842
3	identity	3	8	2	5	0.957541	0.04232
4	tanh	3	7	4	8	0.957608	0.04232
5	tanh	3	4	6	10	0.957652	0.042842
6	tanh	3	5	1	2	0.957675	0.04232
7	identity	2	2	3	0	0.957675	0.04232
8	identity	3	7	6	8	0.957852	0.042842
9	identity	3	1	10	8	0.957875	0.04232
WORST	tanh	3	2	8	6	0.958724	0.041275

Table A3: Top-10 worst model architectures based on AUROC value.

10 Best MCR Architectures

Rank	Activation Function	# of Hidden Layers	# of Nodes (HL1)	# of Nodes (HL2)	# of Nodes (HL3)	AUROC	MCR
BEST	tanh	3	2	8	6	0.958724	0.041275
2	identity	3	1	10	8	0.957875	0.04232
3	identity	2	2	3	0	0.957675	0.04232
4	tanh	3	5	1	2	0.957675	0.04232
5	tanh	3	7	4	8	0.957608	0.04232
6	identity	3	8	2	5	0.957541	0.04232
7	tanh	3	8	8	9	0.956782	0.042842
8	tanh	3	5	9	4	0.95705	0.042842
9	relu	3	7	2	8	0.95705	0.042842
10	tanh	3	10	7	10	0.95705	0.042842

Table A4: Top-10 model architectures based on MCR value.

10 Worst MCR Architectures

Rank	Activation Function	# of Hidden Layers	# of Nodes (HL1)	# of Nodes (HL2)	# of Nodes (HL3)	AUROC	MCR
1	logistic	3	3	3	1	0.5	0.53187
2	logistic	3	9	2	1	0.5	0.53187
3	relu	3	2	7	1	0.5	0.53187
4	logistic	3	3	6	1	0.5	0.53187
5	logistic	3	4	6	1	0.5	0.53187
6	logistic	3	2	9	3	0.5	0.53187
7	logistic	3	6	9	1	0.5	0.53187
8	relu	3	7	6	1	0.5	0.53187
9	relu	3	8	3	1	0.5	0.53187
WORST	tanh	2	2	1	0	0.5	0.53187

Table A5: Top-10 worst model architectures based on MCR value.

Python Script

```
'''
@package    vt_ApplicationsOfML
@fileName   hw5_FindBestNeuralnetwork.py
@author     John Smutny
@date       04/13/2022
@info       Create a script that will train various Artificial Neural
           Networks and determine which architecture performs the best. The
           ANN models are judged based on AUROC and Misclassification Rate,
           with the performance of each architecture being outputted to an
           external xlsx file.

           Configurations
           1. Number of hidden layers: 1-3
           2. Number of Nodes possible in a hidden layer: 1-10
           3. Internal activation function: relu, logistic, identity and tanh
'''

from sklearn import neural_network as ann
from sklearn import metrics
from sklearn import preprocessing as preproc
import sklearn.model_selection as modelsel
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Local libraries
from vt_ApplicationsOfML.Libraries.DataExploration.DataQualityReport import \
   DataQualityReport

## Program settings
##############################################################################
##

TRAIN_DATA = 0.8
TEST_DATA = 0.2
VALID_DATA_FROM_TRAIN = 0.25
RANDOM_SEED = 23

DEBUG = False
OUTPUT_FILES = True
DATA_USE_RATIO = 1

# Neural Network Qualities to be tested.
RANGE_HL = range(1, 4)  # Range: 1-3
RANGE_NODES = range(1, 11)  # Range: 1-10
RANGE_ActFcts = ['relu', 'logistic', 'identity', 'tanh']
```

```
# Universal ANN training settings for all model variations
SOLVER = 'adam'
MAX_ITER = 10000
LEARNING_RATE = 0.0001 *10
TOLERANCE = 0.0001 *10
EARLY_STOPPING = True

# Values used for debugging only. Overwrite if Debugging.
if DEBUG:
   DATA_USE_RATIO = 0.1

   # Neural Network Qualities to be tested.
   RANGE_HL = range(1, 4)
   RANGE_NODES = range(1, 4)
   RANGE_ActFcts = ['relu']

   MAX_ITER = 10


## Control flags and constants
##############################################################################
##

INPUT_FILE = '../data/ccpp.xlsx'
INPUT_SHEET = 'allBin'
OUTPUT_DQR = '../artifacts/ccpp_DQR.xlsx'
OUTPUT_BEST_AUROC = '../artifacts/bestANNs_AUROC.xlsx'
OUTPUT_WORST_AUROC = '../artifacts/worstANNs_AUROC.xlsx'
OUTPUT_BEST_MCR = '../artifacts/bestANNs_MisClassRate.xlsx'
OUTPUT_WORST_MCR = '../artifacts/worstANNs_MisClassRate.xlsx'
OUTPUT_IMAGE = '../artifacts/linearRegressionResults-seed{}.png'.format(
   RANDOM_SEED)


ID_FEATURE = 'ID'
FEATURES = ['AT', 'V', 'AP', 'RH']
TARGET_FEATURE = 'TG'


## Helper Functions
##############################################################################
##
def prepareData(df: pd.DataFrame) -> [pd.DataFrame, pd.DataFrame]:
   # Reduce the number of samples used to speed up testing
   df = df.sample(frac=DATA_USE_RATIO)

   # isolate the Independent variables
   X = df.drop([ID_FEATURE, TARGET_FEATURE], axis=1).to_numpy()
   scalerX = preproc.MinMaxScaler()
```

```
    scalerX.fit(X)

    # Normalize the Independent variable
    X = scalerX.transform(X)

    # isolate the Target variable
    Y = df[TARGET_FEATURE].to_numpy()

    return [X, Y]


def trainAndLogANN(hl, activationFct, trainXY, testXY):
    # Train a single MLP given the user settings and hl/activationFct
    # combination.
    clf = ann.MLPClassifier(hidden_layer_sizes=hl,
                            activation=activationFct,
                            solver=SOLVER,
                            alpha=LEARNING_RATE,
                            early_stopping=EARLY_STOPPING,
                            max_iter=MAX_ITER,
                            validation_fraction=VALID_DATA_FROM_TRAIN,
                            tol=TOLERANCE)

    # Record the performance of the model trained.
    clf.fit(trainXY[0], trainXY[1])
    predictY = clf.predict_proba(testXY[0])
    predictY = np.array(predictY)
    predictY = np.insert(predictY, 2, [0]*predictY.shape[0],
                         axis=1)

    # Process the predictions based on the defined threshold.
    THRESHOLD = 0.5
    for i in range(predictY.shape[0]):
        if predictY[i][1] > THRESHOLD:
            predictY[i][2] = 1
        else:
            predictY[i][2] = 0

    # Factor 1: Area under ROC curve
    auroc = metrics.roc_auc_score(testXY[1], predictY[:, 2])

    # Factor 2: Misclassification Rate
    c_matrix = metrics.confusion_matrix(testXY[1], predictY[:, 2])
    totalWrong = c_matrix[1][0] + c_matrix[0][1]
    totalCases = c_matrix[0][0] + c_matrix[1][1] + totalWrong
    misClassRate = totalWrong / totalCases

    return [auroc, misClassRate]
```

```
def chooseBestANN(x: pd.DataFrame, y: pd.DataFrame, hls, nodes, actFcts) -> [
    pd.DataFrame]:

    # Set up dataframe to collect statistics
    ERROR_HEADER = ['ID', 'ActivationFunction', 'numHiddenLayers',
                    'numNodes_Layer1', 'numNodes_Layer2', 'numNodes_Layer3',
                    'AUROC', 'MisClassificationRate']
    df_error = pd.DataFrame(columns=ERROR_HEADER)

    ################################################
    # Iterate through all defined Neural Network qualities as defined in the
    #   Settings section and record the model's performance.
    #   1) Activation Function
    #   2) # of Hidden Layers
    #   3) # of Nodes per Hidden Layer

    # Set up train, validation, and test data
    trainX, testX, trainY, testY = modelsel.train_test_split(x, y,
                                                test_size=TEST_DATA,
                                                random_state=RANDOM_SEED)
    trainXY = [trainX, trainY]
    testXY = [testX, testY]

    # Define an array of empty hidden layers based on the max number of
    # layers tested.
    hl = [0] * max(hls)

    # Define the max number of nodes per HL are in this iteration
    maxNumNodes = len(nodes)
    NUMBER_OF_MODELS = 0
    for x in range(1, len(hls)+1):
        NUMBER_OF_MODELS = NUMBER_OF_MODELS + (maxNumNodes**x * len(actFcts))

    print("Number of models: {}".format(NUMBER_OF_MODELS))
    id = 0

    # Begin for loop iteration
    print("-- Begin Model Evaluation - Sit back and relax -- ")
    for fct in actFcts:

        # Define how many hidden layers are in this iteration
        for numHLs in hls:

            # Generate a counter to set the number of nodes in each HL
            for i in range(((maxNumNodes)**numHLs)):
                hl[0] = i % (maxNumNodes) + 1

                # Loop through indices of each Hidden Layer.
```

```
                for j in range(1, numHLs):
                    hl[j] = int(i / maxNumNodes**j) % (maxNumNodes) + 1

                # Train the model and record the performance metrics
                [auroc, misClassRate] = trainAndLogANN(hl[0:numHLs], fct,
                                                       trainXY, testXY)

                # TODO - Make this insertion dynamic with size of hls.
                entry = [id, fct, numHLs, hl[0], hl[1], hl[2],
                         auroc, misClassRate]
                df_error.loc[df_error.shape[0]] = entry

                # Increment the count of models trained&Tested
                id = id + 1

            print("-- Evaluating: {}% complete".format(
                                    int(id/NUMBER_OF_MODELS*100)))

    return [df_error]


def outputBestANNs(df: pd.DataFrame):
    id_min_mcr = df[['MisClassificationRate']].idxmin()
    id_min_auroc = df[['AUROC']].idxmin()
    print("Best Model (MisClassificationRate): {}".format(
        df.loc[id_min_mcr, :].values.tolist()))
    print("Best Model (AUROC): {}".format(
        df.loc[id_min_auroc, :].values.tolist()))

    # Gather top 10 AUROC model architectures
    if OUTPUT_FILES:
        df = df.sort_values(by='AUROC')
        df.head(10).to_excel(OUTPUT_BEST_AUROC)
        df.tail(10).to_excel(OUTPUT_WORST_AUROC)
        df = df.sort_values(by='MisClassificationRate')
        df.head(10).to_excel(OUTPUT_BEST_MCR)
        df.tail(10).to_excel(OUTPUT_WORST_MCR)

## Data Handling Functions
#############################################################################
##

# Load and analyze
df_raw = pd.read_excel(INPUT_FILE, sheet_name=INPUT_SHEET)
report = DataQualityReport()
report.quickDQR(df_raw, list(df_raw.columns))

if OUTPUT_FILES:
    report.to_excel(OUTPUT_DQR)
```

```
# Prepare data
[x, y] = prepareData(df_raw)

# Determine the best artificial Neural Network
[df_ErrorData] = chooseBestANN(x, y, RANGE_HL, RANGE_NODES, RANGE_ActFcts)

# Output the 10 best MCR and AUROC models
outputBestANNs(df_ErrorData)
```