DOCTORAL PROGRAM IN ENGINEERING SCIENCES AT ITESO

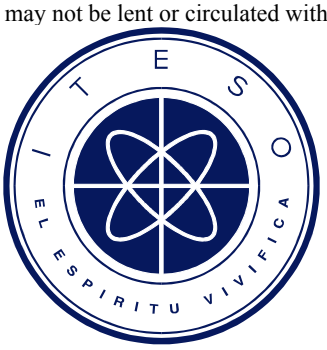
ADAPTATIVE DISCOVERING ALGORITHM BASED ON NEURAL NETWORKS

Edgar-D. Ramirez-de-L.
Maria-del-P. Pozos-P.
Ivan Villalon-Turrubiates

Algorithm 1 - Considerations.doc

May 15, 2018
Tlaquepaque, Mexico 45604

Doctoral Program in Engineering Sciences
ITESO (*Instituto Tecnológico y de Estudios Superiores de Occidente*)



ADAPTATIVE DISCOVERING ALGORITHM BASED ON NEURAL NETWORKS

Edgar-D. Ramirez-de-L.
Maria-del-P. Pozos-P.
Ivan Villalon-Turrubiates

May 15, 2018

Doctoral Program in Engineering Sciences
ITESO (*Instituto Tecnológico y de Estudios Superiores de Occidente*)

Tlaquepaque, Mexico 45604
Tel +52 33 3669 3598
E-mail: dci@iteso.mx

Keywords *neural networks, deep learning, machine learning, classifiers, python*

Abstract

We present the Adaptative Discovering Algorithm based on Neural networks (ADAN algorithm) and its main considerations.

Algorithm 1 ADAN

Require: $args \neq \varnothing \wedge readArg('dataSource', args) \neq \varnothing \wedge readArg('label', args) \neq \varnothing \wedge (readArg('hlan', args) \neq \varnothing \wedge readArg('hlan', args) \geq 1) \wedge (readArg('npla', args) \neq \varnothing \wedge size(readArg('npla', args)) = readArg('hlan', args))$

Ensure: $st = nste$

$tsac \leftarrow 0.0$
$st \leftarrow 0$
$tsta \leftarrow readArg('tsta', args) \vee 0.95$
$sptr \leftarrow readArg('sptr', args) \vee 0.75$
$stpt \leftarrow readArg('stpt', args) \vee 0.01$
$nste \leftarrow readArg('nste', args) \vee 10$
$hlan \leftarrow readArg('hlan', args)$
$npla \leftarrow readArg('npla', args)$
$ts \leftarrow readArg('ts', args) \vee 1000$
$bs \leftarrow readArg('bs', args) \vee 1000$
$x \leftarrow 0$
$hiddenUnits \leftarrow \varnothing$
$LABELS \leftarrow 'Setosa' \mid 'Versicolor' \mid 'Virginica'$
while $x < hlan$ **do**
  $hiddenUnits \leftarrow append(hiddenUnits, npla[x])$
  $x \leftarrow x + 1$
end while
while $tsac < tsta$ **do**
  $(dfTraining, trainY), (dfTesting, testY), dfPredict, expected \leftarrow readSource(args)$
  $features \leftarrow \varnothing$
  $x \leftarrow 0$
  while $x < size(dfTraining.keys)$ **do**
    $features \leftarrow append(features, dfTraining.keys[x])$
    $x \leftarrow x + 1$
  end while
  $classifier \leftarrow DNNClasifier(features, hiddenUnits, size(LABELS))$
  $classifier.train(trainInputFunction(dfTraining, trainY, bs), ts)$
  $results \leftarrow classifier.evaluate(evaluateInputFunction(dfTesting, testY, bs))$
  $tsac \leftarrow results.testAccuracy$
  $printEvaluateResults(results)$
  $predictions \leftarrow classifier.predict(evaluateInputFunction(dfPredict, \varnothing, bs))$
  $printPredictResults(predictions)$
  $sptr \leftarrow sptr + stpt$
  if $tsac \geq tsta$ **then**
    $st \leftarrow st + 1$
  end if
end while

Please take into consideration the follow about the Algorithm 1 (ADAN):

1. The function `readArg('argName', from)` will depend from the programming language, in our case, for Python, we use `argparse.ArgumentParser().parser.add_argument(...)`[1].

2. The function `append(var, value)` (lines 13 and 21) will return `var` with `value` added at its end.

3. `readSource(args)` from line 17, is a function that returns two tuples: the first is the data frame without the y axis (the label for each record/row) and the y axis, all these for the training phase, this is, `(dfTraining, trainY)`; the second tuple is analogous to the former, but for the testing phase, this is `(dfTesting, testY)`; and the last two values are the data frame without the y axis and its corresponding labels, these for the predictions phase. `readSource(args)` and its related functions: `chooseRandomFeatures(args, dataFrame)` and `chooseRandomData(args, randomizedDataFrame)` are defined in Algorithms 2, 3 and 4.

4. The function `size(var)` returns the size of `var` (list or array) and its implementation relies on the programming language and the `var` data type.

5. In Line 20, `dfTraining.keys` refers to each feature/column name in the data frame (`dfTraining`, in this case). This could be an array, list or another data structure, once more, this will depend on the programming language; for Python this is achieved through: `for key in df_training.keys()`.

6. `DNNClassifier` (line 24), like its name suggested, is a Deep Neural Network classifier, which, in this case, receives three arguments: the features' names, the number of units for each hidden layer, and the number of classes (`size(LABELS)`, three in this case). More details will be presented in the Subsection *The implementation*.

7. The function `train` from the `classifier` object, expects two arguments: the train input function, and the number of train steps. This in line 25. `trainInputFunction(data, labels, batchSize)` will be defined later, by the Algorithm 5.

8. The function `evaluate` from the `classifier` object, expects one argument: the evaluate function. This in line 26. `evaluateInputFunction(data, labels,`

[1] Argparse Tutorial, Python Software Foundation. May 4, 2018, https://docs.python.org/3/howto/argparse.html

`batchSize)` will be defined later, by the Algorithm 6.

9. In lines 28 and 30, `printEvaluateResults(arg)` and `printPredictResults(arg)` are simple functions that prints the test accuracy and the predictions results, respectively. Like many others, they depend from the programming language and the data structures returned by them. For Python, the returned result is contained in a dictionary[2] data structure.

10. The function `predict` (line 29) from the `classifier` object, expects one argument: the evaluate input function, same as above (point number 8), defined in Algorithm 6, but its passed arguments are different: for the data we pass the data frame to predict, and for the labels we pass empty/null (`None`, in Python) and the same batch size as the previous.

We believe that the rest of the Algorithm is self-explanatory, of course, considering also the Algorithms 2 to 6.

[2] Data Structures, Python Software Foundation. May 4, 2018, https://docs.python.org/2/tutorial/datastructures.html