README

Original Date: March 23, 2019

From: Kevin M. Johnson, M.D.

Re: Machine Learning Analysis of CCTA data in MATLAB

Version: original

This describes practical aspects of data analysis for the original research paper

“Using Machine Learning to Optimize Vessel Scoring on Coronary CT Angiograms” by Johnson KM et al, under review by the journal Radiology, manuscript RAD-18-2061.R2. It appears as an Addendum in the Supplemental Materials for that paper.

Keep in mind that randomization plays several roles within these algorithms so that your results will very likley not exactly match the results in the paper, but should be very close.

The topics are as follows:

1. Purpose of the research

2. Documentation for models

3. How cross-validation was handled: nested cross-validation

4. Description of the input data file

5. Description of the MATLAB scripts and functions

6. Link to Github repository

7. MATLAB Classification Learner App

Purpose of the Research

Coronary artery disease severity is depicted on CT angiograms; certain vessel features such as stenoses, atherosclerotic plaque amount, etc. can be described by the observer. The question of interest is how to combine these features together to best predict the risk for a future event such as death or myocardial infarction. The dataset contains the vessel imaging features and outcomes for 6892 patients followed for a median of 9 years. Multiple machine learning models were tested by the authors for their discriminatory value. The data and MATLAB scripts and functions have been uploaded to Github were they can be downloaded and explored by any user. MATLAB Statistics and Machine Learning Toolbox and the Neural Network Toolbox are required; the Parallel Computing Toolbox is recommended.

Documentation for models

The model types employed included logistic regression, nearest neighbor classifiers, bagged trees, and classification neural networks. Each model type is embodied by a “classifier creation algorithm (cca)” which does a hyperparameter selection search using cross-validation to estimate error. The set of hyperparameters that gives the highest AUC is found.

The classifier creation algorithms are:

Model1\_LogisticRegression.m

Model2\_NearestNeighbors.m

Model3\_BaggedTrees.m

Model4\_ClassificationNeuralNet.m

The hyperparameters considered in the model creation process were as follows:

|  |  |  |
| --- | --- | --- |
| **Supplemental Table 2.** Hyperparameter Values Explored by Classifier Creation Algorithms | | |
| Model type | Choices explored | Final choices |
| Logistic regression | none | none |
| Nearest Neighbor Classifiers | Distance={'cosine' 'euclidean' 'cityblock'}  NumNeighbors={100 500 1000 2000}  DistanceWeight={'equal'}  Standardize={‘on’ ‘off’} | Distance={'cosine'}  NumNeighbors={1000}  DistanceWeight={'equal'}  Standardize={‘on’} |
| Bagged  trees | NumLearningCycles={200 1000 4000};MaxNumSplits={1 10 100}  MinLeafSize={1 10 100}  NumVariablesToSample={1 2 4 16} | NumLearningCycles={200};  MaxNumSplits={100}  MinLeafSize={10}  NumVariablesToSample={1} |
| Classification  Neural  Network  (patternnet) | trainFcn = {'trainscg' 'trainlm'}  performFcn={'crossentropy' 'mse' ‘msereg’}  hiddenLayerSize = {5 10 15 20} | trainFcn = {'trainscg'}  performFcn={'crossentropy'}  hiddenLayerSize = {15} |

The following link provides documentation of each model type:

<https://www.mathworks.com/help/stats/classification.html>

How cross-validation was handled: nested cross-validation

Cross-validation needs to be carefully done when used for model selection. The prediction error estimate can be artefactually small if testing is not done on data previously unseen by the model. Nested cross-validation is a method to avoid bias in the estimate [1,2].

Briefly, the data are divided into 5 folds, and a cross-validation loop is created; this is the “outer” loop. Within this loop, a classifier creation algorithm (cca) is called, for example Model3\_BaggedTrees.m. This algorithm receives data from the outer loop, representing 4/5 of the total data; the cca explores the space of hyperparameters to find the set that gives the smallest prediction error on this subset, as judged within the cca by an internal CV loop, entirely separate from the outer loop CV. The cca uses 3 fold CV. Of note, because of their stochastic nature, both inner and outer loops are repeated multiple times and the results averaged.

Once the best classifier is returned by the cca to the outer loop, it is used to score the remaining 1/5 of the data, and the scores are recorded for that fold. This process is repeated for all of the folds in the outer loop. After all 5 iterations we have one score per each set of original observations. This paper uses area under the curve (AUC) as a measure of model accuracy rather than using misclassification rate. Therefore, the scores and known outcomes undergo ROC analysis; the AUC serves as an estimate of likely performance when applied to new data. This this estimate tends to be less biased than the estimates found within the cca CV routines.

For the separate hold out validation procedure, nested cross-validation is done as above but only using 2/3 of the original data. Then the final classifier is used to predict outcomes on the held out data.

Schematic code for the nested cross-validation method (full code on Github, link below):

% "Outer" cross-validation routine

for fold = 1:5

% get training data

xtrain = 4/5 of predictors

ytrain = 4/5 of responses

% call a classifier creation algorithm

which returns the best classifier as determined by an “inner” cross-validation routine.

Hyperparameters are selected within this routine.

trainedClassifier=cca(xtrain,ytrain);

%apply this classifier to test set

xtest = other 1/5 of predictors

ytest = other 1/5 of responses [predicted\_responses,predicted\_scores]=…

trainedClassifier.predictFcn(xtest);

%for this fold, find area under the receiver operating characteristic curve (AUC) as a measure of accuracy

AUC\_fold(fold)=perfcurve(ytest,xtest);

end

% We now have AUC\_fold, a distribution of AUC values. The mean is estimate

% of how well this suite of classifiers is likely to generalize.

% Make the final classifier

%get all the data

xall = all\_predictors

yall = all\_responses

%get the final classifier by retraining on all the data

final\_trainedClassifier=cca(xall,yall)

In practice, the cross-validation steps are each repeated multiple times and their results averaged.

The following schematic code is used to test any new data:

% Get new data

xnew = new\_predictors

% Apply final trained classifier [predicted\_responses,predicted\_scores]=…

final\_trainedClassifier.predictFcn(xnew)

% Find AUC as a measure of accuracy (if new responses are known)

ynew = new\_responses

AUC\_new=perfcurve(ynew,xnew)

1. Varma S, Simon R. Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics 2006, 7:91-98, doi:10.1186/1471-2105-7-91.

2. <https://www.mathworks.com/discovery/cross-validation.html>

Description of the input data file

CCTAdataMarch2019.mat is a 6892 x 72 table; the first 64 columns are the coronary artery *features*, columns 65 through 69 are *conventional scores*, and columns 70 through 72 are *outcomes*.

*Features*. The coronary arterial tree is considered to be comprised of 16 segments; for example, the proximal third of the right coronary is one segment, the distal third of the left anterior descending coronary is another, etc. For each segment, 4 features are defined: degree of stenosis, amount of plaque, amount of calcification, and presence of remodeling (focal external diameter dilatation). Thus there are 64 imaging features per patient. Each feature is assigned a value depending on the degree of abnormality. The overall score for a given patient is the sum of these values.

*Conventional scores* are the reference standard against which the machine learning results are to be compared. There are 5 such scores: CAD-RADS, LeS, SPS, SSS, and SIS. See the Methods section of the paper for details on how these are constructed.

*Outcomes* are the events that occur in the years of follow-up after the CT scan has been performed. Three events have been defined: all deaths, coronary artery deaths, and the sum of coronary artery deaths and myocardial infarctions. Median follow-up was 9.0 years. In general, variable numbering in the MATLAB code refer to these as 1 through 3 respectively.

Description of the MATLAB scripts and functions

Note: The load and save commands in the following routines must be edited to reflect the correct storage locations in your file system.

A\_load\_data.m - This is the first script to run. It loads the data as a table “data\_table\_all” with the first 64 columns representing vessel features, columns 65 through 69 representing conventional scores, and columns 70 through 72 representing the three types of outcomes. Predictors with zero variance are eliminated. The row order is randomized because the original matrix is in chronological order.

B1\_call\_nested\_CV.m - This is the second script to run. It loads “data\_table\_all”. The first step is to remove predictors that we do not wish to include; for example, if only the vessel features are desired as inputs, the 5 conventional score columns must be set to empty. Next, if a separate hold out validation step is planned after nested CV, the input dataset is split into a training set (2/3) for nested cross-validation and a naïve test set (1/3). Otherwise, all of the data will be used for nested cross-validation.

“Redundancy” refers to the number of folds and also to the number of repeats of the outer and inner CV routines. We average those results because the partition process is random each time. The outer loop is repeated 30 times, giving us that many AUC values as our final result. The inner loop is repeated 10 times when multiple possible vlaues for hyperparameters have been designated within a given classifier creation routine. To adjust this, go to the code for the cca and uncomment the relevant lines at the top of the code to activate either lists of hyperparameter values or else a single value for each parameter. The single values shown are those used by the final classifiers found in the paper. If hyperparameters are not being selected, the inner loop does not need to be repeated (but redundancy.numrepeats\_inner still needs to be set to 1).

You are asked to designate the classifier creation algorithm type. The nested CV routine B2\_nested\_CV.m is called three times, once for each outcome type. Results are summarized and displayed.

B2\_nested\_CV.m - This is the nested CV function described above under How cross-validation was handled. It calls the classifier creation algorithm (cca) for the model under consideration, once for each of 5 folds. The output is a final classifier and a distribution of AUC values, an estimate of how well the classifier will generalize to unseen data.

D1\_call\_predict\_newdata.m - This is used when the separate hold out validation procedure has been selected. The independent test data is loaded, and D2\_predict\_newdata is called for each of the three outcome types. Results are summarized and displayed.

D2\_predict\_newdata.m - This is called by D1\_call\_predict\_newdata. The final classifier found by B2\_nested\_CV.m is applied to the new set of predictors. When responses are known (as in this case), ROC analysis is done to find the “hold out” AUC as a measure of performance.

Link to Github repository

Our data, scripts and functions can be found at:

[https://github.com/kevinjohnson40/CCTA\_ml\_prognosis/](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fkevinjohnson40%2FCCTA_ml_prognosis%2F&data=02%7C01%7Ckevin.johnson%40yale.edu%7C8b94da85ffe443e9efba08d6733c3f0a%7Cdd8cbebb21394df8b4114e3e87abeb5c%7C0%7C0%7C636823099933596414&sdata=b31xVGQ70nK17BPg2BF2Hofj9kw0WX7%2BEzoZlm66qmM%3D&reserved=0)

MATLAB Classification Learner App overview

The MATLAB Classification Learner App is a feature of the Statistics and Machine Learning Toolbox. A graphical user interface provides the means to analyze data using many different model types, and to set parameters as desired for each model. The App can generate code for adaptation by the user to perform even more complex analyses. The data tables named CCTAtable1, etc. constructed by B1\_call\_nested\_CV.m can be used as inputs to the App.

<https://www.mathworks.com/products/statistics/classification-learner.html>

*Note* - The classification neural network is implemented via the Neural Network GUI (nprtool) rather than in the Classification Learner App.