Training Predictive Models

Dr. Charles Determan Jr. PhD[*]

May 17, 2016

1 Introduction

This vignette is designed to simply get a user 'up and running' with model development. The statistical justifications are not included here but are intended to be included in a subsequent vignette. If the user is curious, feel free to search the extensive machine learning literature on the value of cross-validation.

2 Installation

This package is not an official R package and exists solely in the authors github repository. There are some additional dependencies that must also be installed.

```
# install devtools and bigmemory backends
install.packages(c("devtools","bigmemory", "bigalgebra", "biganalytics"))

# the other dependencies should be installed automatically
devtools::install_github("cdeterman/HGTools")
```

[*]cdeterman@healthgrades.com

3 Grid Searching

If you are completely unfamiliar with the model you wish to tune this package provides the function `denovo.grid`. For most methods, this function only requires you to specify the method and 'resolution' for the grid. You can see all available models with the `modelInfo` function.

```
denovo.grid(method = "neuralnet", res = 3)
```

However, some methods require the dataset for estimating appropriate parameters. This includes random forest (denoted 'rf').

```
# Note the dvs denotes the dependent variables to omit from the estimation
# This can be omitted but you will receive a warning
denovo.grid("neuralnet", res=5, data=trainingData, dvs=c("my_dv"))
```

If you know what you are doing, you can also create the grid manually using the `expand.grid` function. Please note that the hyperparameters you specify must match the desired model and be prefixed with a 'period'

```
expand.grid(.hidden = seq(2,5), .threshold = c(5, 1))
```

You can see the available parameters for each model by running `modelInfo`.

4 Demo Data

Now you are essentially ready to begin training your model. You only need to have some data to work with. The following is an example using the adhd dataset included in this package for testing purposes. You can load the training and testing datasets with the following commands:

```
data("adhd_train")
data("adhd_test")
```

5 Model Training

The function call below will fit a neuralnet model. The primary arguments include:

- X - character vector of independent variable names
- Y - character vector of dependent variable names
- data - the dataset to be trained from
- testData - the dataset used to evaluate the final model
- k - the number of fold used in cross-validation (default: 10)
- metric - performance metric to evaluate models (default: AUC)

```r
# To save space I am indexing the names
cnames <- colnames(training)
ivs <- cnames[3:ncol(training)]
dvs <- cnames(training)[1]

f <- as.formula(paste(dvs, " ~ ", paste(ivs, collapse= "+")))

fit_nn <- train(formula = f,
                data = training,
                testData = testing,
                method = "neuralnet",
                grid = grid,
                k = 5,
                metric = "AUC"
)
```

6 Results

Once the `train` call has completed, you will be returned a list with contains the following elements:

1. finalModel - the final model generated from the 'best' hyperparameters, this is your production model from those evaluated
2. performance - the performance statistics for the final model against the test dataset
3. cvPerformanceMatrix - a list containing the means and variances of each hyperparameter iteration
4. bestParams - a data.frame containing the 'best' hyperparameters

7 Parallelization

This task can clearly take advantage of parallelization. To do so, you must set up the parallel backend in R. To do so depends upon your operating system. The cross-platform backed to use the package *doParallel*. You simply need to pass the number of cores you wish to parallize over with `registerDoParallel`.

```r
# register 8 cores
cl <- makeCluster(8)
registerDoParallel(8)

# make sure to stop cluster when completed
stopCluster(cl)
```

The linux specific version is *doMC* with the similar function `registerDoMC`. You can essentially use whatever backed you wish so long as it is compatible with your system.

Once the backend is setup you can call the `train` the same as before but change the 'allowParallel' argument to TRUE.

```r
fit_nn <- train(formula = f,
                data = training,
                testData = testing,
                method = "neuralnet",
                grid = grid,
                k = 5,
                metric = "AUC",
                allowParallel = TRUE
)
```

8 Saving All Models

It may be the user's wish to save every model that is created during cross-validation. This option is also provided with the `train` function with the argument `save_models`. Simply set the `save_models` to TRUE and all models generated with be saved with the format 'model_hyperparam_hyperparam_iter_cv_model.rda' with varying numbers of hyperparameters for the specific model.