Machine learning with H2O

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

What is H2O?

```
library(h2o)
h2o.init()
```

```
H2O is not running yet, starting it now...

java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)

Starting H2O JVM and connecting: ... Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         2 seconds 124 milliseconds
    H2O cluster version:        3.20.0.8
    H2O cluster total nodes:    1
    H2O cluster total memory:   3.56 GB
    H2O cluster total cores:    8
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.5.1 (2018-07-02)
```

New dataset: seeds data

```
glimpse(seeds_data)
```

```
Observations: 150
Variables: 8
$ area          <dbl> 15.26, 14.88, 14.29, 13.84 ...
$ perimeter     <dbl> 14.84, 14.57, 14.09, 13.94 ...
$ compactness   <dbl> 0.8710, 0.8811, 0.9050 ...
$ kernel_length <dbl> 5.763, 5.554, 5.291, 5.324 ...
$ kernel_width  <dbl> 3.312, 3.333, 3.337, 3.379 ...
$ asymmetry     <dbl> 2.2210, 1.0180, 2.6990 ...
$ kernel_groove <dbl> 5.220, 4.956, 4.825, 4.805 ...
$ seed_type     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
```

```
seeds_data %>%
  count(seed_type)
```

```
# A tibble: 3 x 2
  seed_type     n
      <int> <int>
1         1    50
2         2    50
3         3    50
```

Preparing the data for modeling with H2O

- Data as **H2O frame**

```
seeds_data_hf <- as.h2o(seeds_data)
```

- Define **features and target** variable

```
y <- "seed_type"
x <- setdiff(colnames(seeds_data_hf), y)
```

- For classification target should be a **factor**

```
seeds_data_hf[, y] <- as.factor(seeds_data_hf[, y])
```

Training, validation and test sets

```
sframe <- h2o.splitFrame(data = seeds_data_hf,
                         ratios = c(0.7, 0.15),
                         seed = 42)
train <- sframe[[1]]
valid <- sframe[[2]]
test <- sframe[[3]]
```

```
summary(train$seed_type, exact_quantiles = TRUE)
```

```
seed_type
 1:36
 2:36
 3:35
```

```
summary(test$seed_type, exact_quantiles = TRUE)
```

```
 seed_type
 1:8
 2:8
 3:5
```

Model training with H2O

- Gradient boosted models with `h2o.gbm()` & `h2o.xgboost()`
- Generalized linear models with `h2o.glm()`
- Random forest models with `h2o.randomForest()`
- Neural networks with `h2o.deeplearning()`

Model training with H2O

```
gbm_model <- h2o.gbm(x = x, y = y,
                     training_frame = train,
                     validation_frame = valid)
```

```
Model Details:
==============

H2OMultinomialModel: gbm
Model ID:  GBM_model_R_1540736041817_1
Model Summary:
number_of_trees number_of_internal_trees model_size_in_bytes min_depth
             50                      150                24877         2
max_depth mean_depth min_leaves max_leaves mean_leaves
        5    4.72000          3         10     8.26667
```

- Model **performance**

```
perf <- h2o.performance(gbm_model, test)
h2o.confusionMatrix(perf)
```

```
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
       1 2 3  Error     Rate
1      7 0 1 0.1250 =  1 / 8
2      0 8 0 0.0000 =  0 / 8
3      0 0 5 0.0000 =  0 / 5
Totals 7 8 6 0.0476 = 1 / 21
```

```
h2o.logloss(perf)
```

```
0.2351779
```

- **Predict** new data

```
h2o.predict(gbm_model, test)
```

Let's practice!

HYPERPARAMETER TUNING IN R

Grid and random search with H2O

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

datacamp

Hyperparameters in H2O models

- Hyperparameters for **Gradient Boosting**:

```
?h2o.gbm
```

- `ntrees` : Number of trees. Defaults to 50.
- `max_depth` : Maximum tree depth. Defaults to 5.
- `min_rows` : Fewest allowed (weighted) observations in a leaf. Defaults to 10.
- `learn_rate` : Learning rate (from 0.0 to 1.0) Defaults to 0.1.
- `learn_rate_annealing` : Scale the learning rate by this factor after each tree (e.g., 0.99 or 0.999) Defaults to 1.

Preparing our data for modeling with H2O

- Convert to **H2O frame**

```
seeds_data_hf <- as.h2o(seeds_data)
```

- Identify **features and target**

```
y <- "seed_type"
x <- setdiff(colnames(seeds_data_hf), y)
```

- **Split** data into train, test & validation set

```
sframe <- h2o.splitFrame(data = seeds_data_hf, ratios = c(0.7, 0.15), seed = 42)
train <- sframe[[1]]
valid <- sframe[[2]]
test <- sframe[[3]]
```

Defining a hyperparameter grid

- GBM hyperparamters

```
gbm_params <- list(ntrees = c(100, 150, 200), max_depth = c(3, 5, 7), learn_rate = c(0.001, 0.01, 0.1))
```

- `h2o.grid` function

```
gbm_grid <- h2o.grid("gbm",
                     grid_id = "gbm_grid",
                     x = x,
                     y = y,
                     training_frame = train,
                     validation_frame = valid,
                     seed = 42,
                     hyper_params = gbm_params)
```

- Examine results with `h2o.getGrid`

Examining a grid object

- **Examine results** for our model `gbm_grid` with `h2o.getGrid` function.
- Get the grid results sorted by validation accuracy

```
gbm_gridperf <- h2o.getGrid(grid_id = "gbm_grid", sort_by = "accuracy", decreasing = TRUE)
```

```
Grid ID: gbm_grid
Used hyper parameters:
  -  learn_rate
  -  max_depth
  -  ntrees
Number of models: 27
Number of failed models: 0

Hyper-Parameter Search Summary: ordered by decreasing accuracy
```

Extracting the best model from a grid

- Top GBM model chosen by **validation accuracy** has id position 1

```
best_gbm <- h2o.getModel(gbm_gridperf@model_ids[[1]])
```

- These are the **hyperparameters** for the best model:

```
print(best_gbm@model[["model_summary"]])
```

```
Model Summary:
 number_of_trees number_of_internal_trees model_size_in_bytes min_depth
             200                       600              100961         2
 max_depth mean_depth min_leaves max_leaves mean_leaves
         7    5.22667          3         10     8.38833
```

Extracting the best model from a grid

- `best_gbm` is a **regular H2O model** object and can be treated as such!

```
h2o.performance(best_gbm, test)
```

```
MSE: (Extract with `h2o.mse`) 0.04761904
RMSE: (Extract with `h2o.rmse`) 0.2182179
Logloss: (Extract with `h2o.loglos
```

Random search with H2O

- In addition to hyperparameter grid, add **search criteria**:

```
gbm_params <- list(ntrees = c(100, 150, 200),
                   max_depth = c(3, 5, 7),
                   learn_rate = c(0.001, 0.01, 0.1))
search_criteria <- list(strategy = "RandomDiscrete",
                        max_runtime_secs = 60,
                        seed = 42)
gbm_grid <- h2o.grid("gbm",
                     grid_id = "gbm_grid",
                     x = x, y = y,
                     training_frame = train,
                     validation_frame = valid,
                     seed = 42,
                     hyper_params = gbm_params,
                     search_criteria = search_criteria)
```

```
search_criteria <- list(strategy = "RandomDiscrete",
                        stopping_metric = "mean_per_class_error",
                        stopping_tolerance = 0.0001,
                        stopping_rounds = 6)
gbm_grid <- h2o.grid("gbm", x = x, y = y, training_frame = train, validation_frame = valid,
                     seed = 42, hyper_params = gbm_params, search_criteria = search_criteria)
```

```
H2O Grid Details
================
Grid ID: gbm_grid
Used hyper parameters:
  -  learn_rate
  -  max_depth
  -  ntrees
Number of models: 30
Number of failed models: 0
```

Time to practice!

HYPERPARAMETER TUNING IN R

Automatic machine learning with H2O

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

Automatic Machine Learning (AutoML)

- **Automatic tuning of algorithms**, in addition to hyperparameters

- AutoML makes model tuning and optimization much **faster and easier**

- AutoML only needs a **dataset**, a **target** variable and a **time or model number limit** for training

AutoML in H2O

AutoML compares

- Generalized **Linear Model** (GLM)
- (Distributed) **Random Forest** (DRF)
- Extremely **Randomized Trees** (XRT)
- Extreme **Gradient Boosting** (XGBoost)
- **Gradient Boosting** Machines (GBM)
- **Deep Learning** (fully-connected multi-layer artificial neural network)
- Stacked **Ensembles** (of all models & of best of family)

GBM Hyperparameters

- `histogram_type`
- `ntrees`
- `max_depth`
- `min_rows`
- `learn_rate`
- `sample_rate`
- `col_sample_rate`
- `col_sample_rate_per_tree`
- `min_split_improvement`

Deep Learning Hyperparameters

- `epochs`
- `adaptivate_rate`
- `activation`
- `rho`
- `epsilon`
- `input_dropout_ratio`
- `hidden`
- `hidden_dropout_ratios`

```
# Using h2o.automl function
automl_model <- h2o.automl(x = x, y = y,
                           training_frame = train,
                           validation_frame = valid,
                           max_runtime_secs = 60,
                           sort_metric = "logloss",
                           seed = 42)
```

- returns a **leaderboard** of all models, **ranked** by the chosen metric (here "logloss")

```
Slot "leader":
Model Details:
==============

H2OMultinomialModel: gbm
Model Summary:
 number_of_trees number_of_internal_trees model_size_in_bytes min_depth
             189                       567               65728         1
 max_depth mean_depth min_leaves max_leaves mean_leaves
         5    2.96649          2          6     4.20988
```

Viewing the AutoML leaderboard

```
lb <- automl_model@leaderboard
```

```
                                        model_id mean_per_class_error
1  GBM_grid_0_AutoML_20181029_144443_model_6           0.01851852
2 GBM_grid_0_AutoML_20181029_144443_model_30           0.02777778
3 GBM_grid_0_AutoML_20181029_144443_model_18           0.02777778
4  GBM_grid_0_AutoML_20181029_144443_model_9           0.03703704
```

- Per default, the leaderboard is calculated on 5-fold cross-validation.

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

Extracting models from AutoML leaderboard

```
# List all models by model id
model_ids <- as.data.frame(lb)$model_id
```

```
 [1] "GBM_grid_0_AutoML_20181029_144443_model_6"
 [3] "GBM_grid_0_AutoML_20181029_144443_model_18"
[19] "XRT_0_AutoML_20181029_144443"
[20] "DRF_0_AutoML_20181029_144443"
[24] "DeepLearning_0_AutoML_20181029_144443"
[41] "StackedEnsemble_BestOfFamily_0_AutoML_20181029_144443"
[42] "StackedEnsemble_AllModels_0_AutoML_20181029_144443"
```

```
# Get the best model
aml_leader <- automl_model@leader
```

- `aml_leader` is again a regular **H2O model** object and can be treated as such!

Get ready for your last round of exercises!

HYPERPARAMETER TUNING IN R

Congratulations!

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

datacamp

What you've learned in this course

- What **hyperparameters** are
- How they are different from **model parameters**
- And **why** to tune them
- **How** tuning works in three R packages:
- `caret`
- `mlr`
- `h2o`

Terms you can understand and apply

- Cartesian Grid Search
- Random Search
- Adaptive Resampling
- Automatic Machine Learning
- Evaluating tuning results with performance metrics
- Stopping criteria

How you can use this knowledge

- Find best hyperparameter set for your models
- Compare and contrast R packages => **favorite**

Where to go from here?

- Package manuals & vignettes
- Try it out!
- **UC Irvine Machine Learning Repository**
- **Kaggle**

Thank you and have fun!

HYPERPARAMETER TUNING IN R