Hyper-parameter tuning with Bayesian techniques

xgboost use case with mlrMBO package (R)

Author: Laurent Querella, PhD, MSc
Senior Data Scientist (freelance) at TotalEnergies

Version: May 2021

Table of Contents

1 Introduction

Avertissement

Session informelle plus pratique que théorique

Position du problème

Xgboost est un algorithme qui est très performant

Mais présentant un nombre de paramètres conséquent

http://xgboost.readthedocs.io/en/latest/parameter.html

Medium

Synced Follow
AI Technology & Industry Review @www.syncedreview.com | www.jiqizhixin.com
Oct 23, 2017 · 15 min read

Tree Boosting With XGBoost—Why Does XGBoost Win "Every" Machine Learning Competition?

"Parameter tuning is a dark art in machine learning"

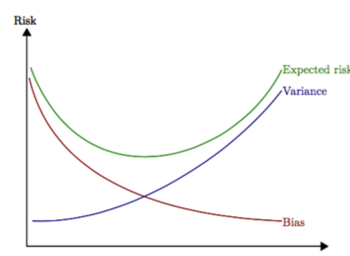
Généralités sur xgboost

http://xgboost.readthedocs.io/en/latest/model.html

- Supervised learning
  - Classification – regression – ranking
  - We use the training data (with multiple features) x_i to predict a target variable y_i
- Model and parameters
  - Comment la cible est-elle reliée aux prédicteurs?
    - Exemple linéaire: $\hat{y}_i = \sum_j \theta_j x_{ij}$
  - Les paramètres sont les inconnues que l'on apprend à partir des données (θ)
- Model selection
  - Objective function measures the performance of the model given a certain set of parameters

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta)$$

    - L: Loss function (performance sur le training) (eg. MSE)
    - Omega: Regularization term (contrôle la complexité – évite overfitting)



2 XGBOOST

Paramètres de xgboost
http://xgboost.readthedocs.io/en/latest/parameter.html

- Xgboost = (extreme gradient) boosted trees (addition d'arbres)
  - Random Forest = bagging (many models after resampling – weak learners (decision trees) -> strong one)
  - Xgboost = boosting + bagging (boosting = training new model by including error of previous one)
- Choix du booster
  - gbtree ou gblinear
- Learning objective
  - "reg:linear" –linear regression
  - "reg:logistic" –logistic regression
  - …
- Evaluation metric(s)
  - (…) MAE, Logloss, auc, …
- Tree booster
  - Eta – learning rate
  - Gamma – paramètre de régularisation (complexité des arbres)
  - Max_depth – profondeur maximale des arbres
  - …

3 Parameter Tuning

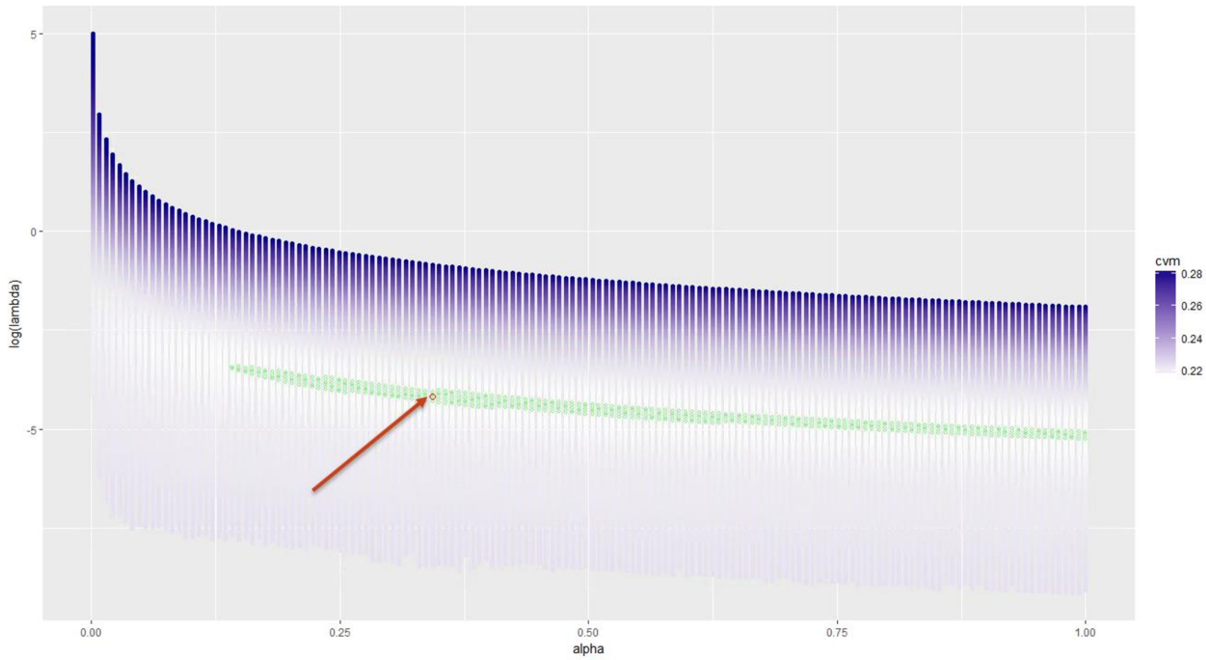
3.1 Method

Parameter Tuning

- Parameter space
  - Can be of very dimension
    - N parameters and 10 values each ➔ 10^N combinations
  - Simple case: glmnet: alpha and lambda => can be represented in 2D
    - Demo (**glmnetUtils** + **glmnet** R packages) – cf. next slide
- Approaches
  - Manual selection of parameters (règle du doigt mouillé – j'ai réussi à la placer!)
  - Random exploration (grid search)
    - Ex. matrix 10x10 and decide to iterate only 20 times
    - **xgboost** in R (equivalent in Python with scikit-learn)
      - xgb.train or xgb.cv with a loop or with **caret** grid search
  - Need more efficient method
    - Trade-off between exploring the parameter space and budget (execution time)
    - ➔ Bayes optimization!

3.2 Online demo

DEMO: OPTIMAL PARAMETERS WITH GLMNET



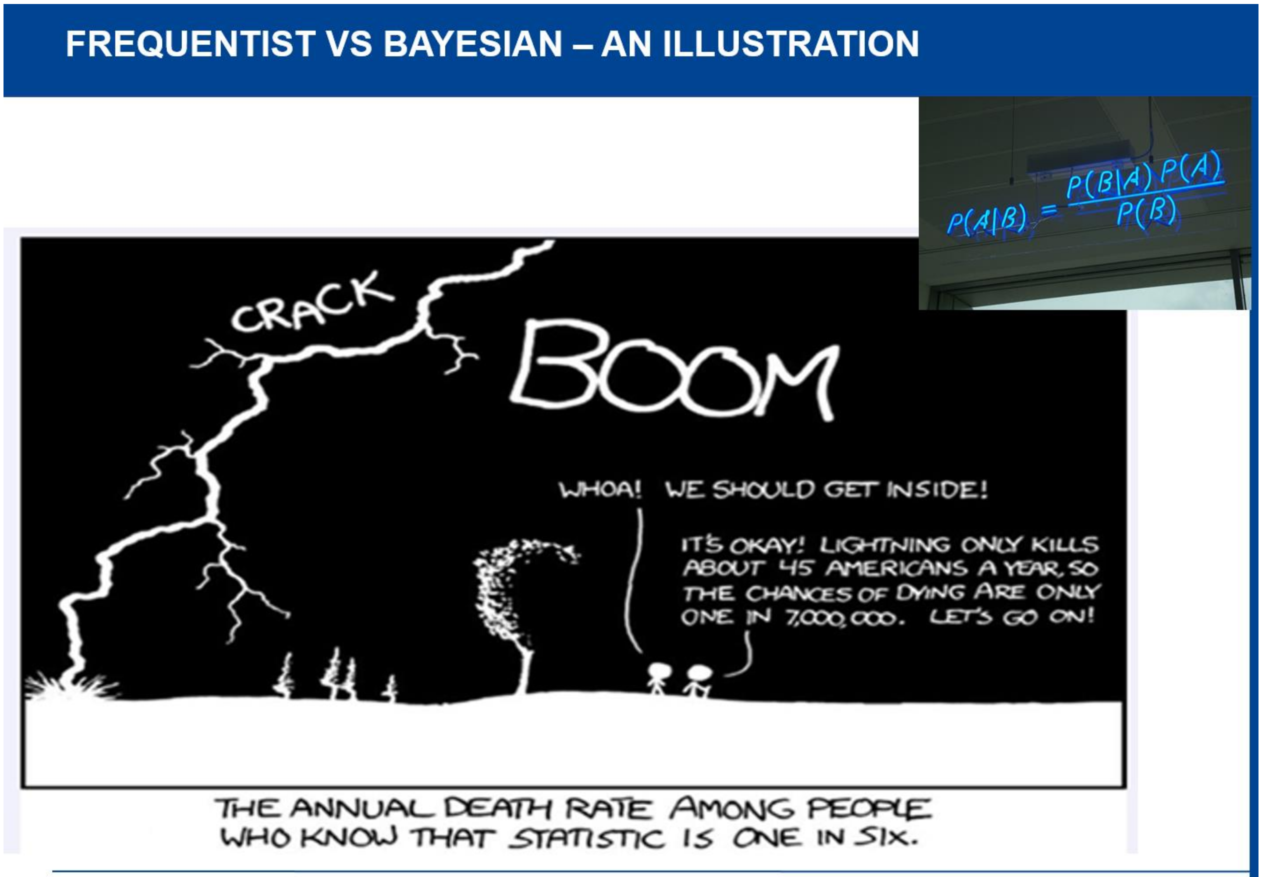
In red: best couple of alpha/lambda – minimizing CV error
In green: surface of best CV errors with 0.1% tolerance

3.3 Bayesian statistics

Bayesian statistics

- Inference statistics
  - Definition …
  - 1. estimation des paramètres qui déterminent les propriétes de la distribution des données
    - Ex. normale(mu,sigma)
  - 2. data prediction de données supplémentaires étant donné connaissance des paramètres
  - 3. Comparaison de modèles: pick the best one (celui qui fit le mieux les données)
    - Ex. régression simple ou + interactions terms ou polynomial terms
- Def of probabilities
  - Long-term frequencies ➔ **frequentist** approach
    - Only repeatable events
    - Parameters = unknown **fixed** constants
  - Degrees of belief + logical support ➔ **Bayesian** approach
    - Any event (even if rare or unique)
    - Parameters = unknown **random** variables

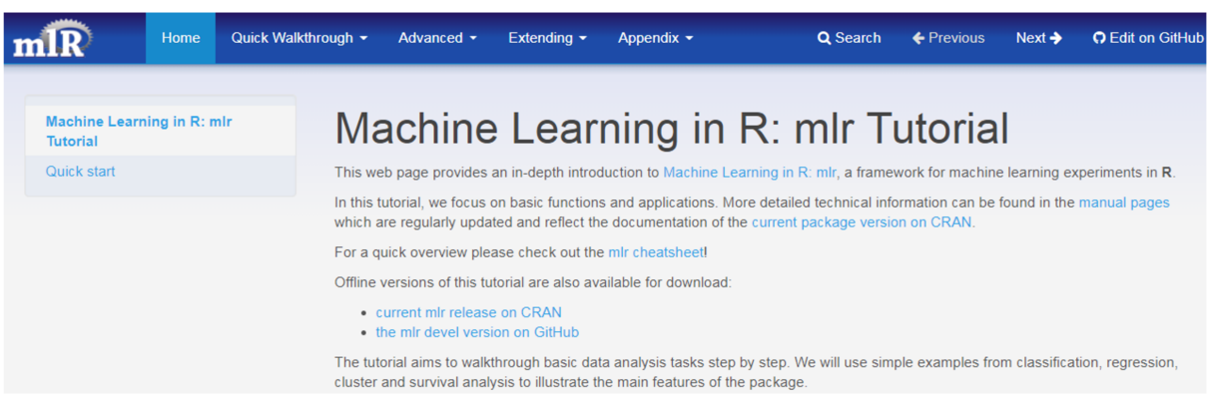
- Exemple
  - On veut estimer la moyenne de la taille des femmes d'une population donnée
  - Hypothèses:
    - On suppose que cette variable (y) suit une distribution normale (ou gaussienne) avec moyenne μ et variance σ^2
    - On suppose que la variance est connue
  - Approche fréquentiste
    - μ est inconnue mais on postule qu'elle prend une valeur fixe
    - La seule option est de collecter des données (échantillon) et d'estimer la moyenne sur cet échantillon (MLE – maximum likelihood estimation)
  - Approche bayesienne
    - μ est inconnue et présente une incertitude dans sa valeur (suit une distribution de probabilité)
    - Les données collectées au fur et à mesure vont affiner la connaissance de cette distribution
    - Clé = **Théorème de Bayes**
      - Probabilités conditionnelles – (ceci mériterait une session séparée – je me limite à l'illustration suivante)



3.4 R packages

R packages

- MLR
  - https://mlr-org.github.io/mlr-tutorial/release/html/



- mlrMBO
  - https://arxiv.org/abs/1703.03373

Cornell University Library

arXiv.org > stat > arXiv:1703.03373

Statistics > Machine Learning

mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions

Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, Michel Lang

(Submitted on 9 Mar 2017 (v1), last revised 15 Mar 2017 (this version, v2))

We present mlrMBO, a flexible and comprehensive R toolbox for model-based optimization (MBO), also known as Bayesian optimization, which addresses the problem of ex through a surrogate regression model. It is designed for both single- and multi-objective optimization with mixed continuous, categorical and conditional parameters. Addition logging and error-handling. mlrMBO is implemented in a modular fashion, such that single components can be easily replaced or adapted by the user for specific use cases, used, and infill criteria and infill optimizers are easily exchangeable. We empirically demonstrate that mlrMBO provides state-of-the-art performance by comparing it on differe DiceOptim, rBayesianOptimization, SPOT, SMAC, Spearmint, and Hyperopt.

Comments: 23 pages, 5 figures
Subjects: **Machine Learning (stat.ML)**
Cite as: arXiv:1703.03373 **[stat.ML]**
(or arXiv:1703.03373v2 **[stat.ML]** for this version)

Example xgboost with mlrMBO tuning

- Demo

1. Creating a learning task (here regression)

   regr.task.train <- makeRegrTask(data = df_train, target = targetname)

2. Building a learner (algo)

   regr.lrn = makeLearner("regr.xgboost", predict.type = "response",
   par.vals = par_list)

3. Setting the parameter space

   makeParamSet(…)

```
ps.mbo <- makeParamSet(
  makeDiscreteParam("booster", values = c(booster)),
  makeDiscreteParam("objective", values = c(obj)),
  makeDiscreteParam("eval_metric", values = c(eval_metric)),
  makeDiscreteParam("nrounds", values = c(25L, 50L, 100L, 150L)),
  makeDiscreteParam("watch_list", values = c(watchlist.evaltest)),
  makeIntegerParam("max_depth",lower = 4L, upper = 6L),
  makeNumericParam("eta",lower = 2.0, upper = 10.0, trafo = function(x) (x/241L)),
  makeNumericParam("gamma",lower = 1.0, upper = 20.0, trafo = function(x) log(x)),
  makeDiscreteParam("min_child_weight", values = c(1.0,2.0)),
  # makeNumericParam("min_child_weight",lower = 1.0, upper = 10.0),
  makeDiscreteParam("subsample", values = c(0.6,0.7,0.8,0.9)),
  # makeNumericParam("subsample",lower = 1.0,upper = 2.0, trafo = function(x) x/2),
  makeDiscreteParam("colsample_bytree", values = c(0.6,0.7,0.8,0.9)),
  # makeNumericParam("colsample_bytree",lower = 1.0,upper = 2.0, trafo = function(x) x/2),
  makeDiscreteParam("colsample_bylevel", values = c(0.6,0.7,0.8,0.9)),
  # makeNumericParam("colsample_bylevel",lower = 1.0,upper = 2.0, trafo = function(x) x/2),
  makeNumericParam("alpha",lower = 1.0, upper = 10.0, trafo = function(x) log(x)),
  makeNumericParam("lambda",lower = 0.0,upper = 1.0, trafo = function(x) exp(x)))
```

Example xgboost with mlrMBO tuning

- Demo

4. Make a design matrix
   = first set of parameters evaluated with xgboost as starting point
   generateDesign()

```r
# # Make a design matrix
# design.dim = 4L * length(ps.mbo$pars) #default
design.dim = 8L * length(ps.mbo$pars) #initial budget greater before 10.04

# design.mat = generateRandomDesign(n = (4L * length(ps.mbo$pars)), par.set = ps.mbo)
## Option: Latin Hypercube design
# generateDesign(n = 10L, par.set, fun, fun.args = list(), trafo = FALSE,
#                 augment = 20L)
design.lhs = generateDesign(n = design.dim, par.set = ps.mbo, fun = randomLHS)
```

5. Define the tuning strategy
   makeTuneControl**MBO**()

```r
ctrl = makeMBOControl(propose.points = ncpus)
ctrl = setMBOControlTermination(ctrl, iters = iters) #, max.evals = 25L)
ctrl = setMBOControlInfill(ctrl, crit = crit.ei) #expected improvement
ctrl = setMBOControlMultiPoint(ctrl, method = "cl", cl.lie = min)

rdesc = makeResampleDesc(method = "CV", iters = 3, predict = "both")

tune.ctrl = makeTuneControlMBO(mbo.control = ctrl, mbo.design = design.lhs)
```

6. Perform the tuning (in parallel mode)
   best parameters = res$x
   CV performance = res$y

```r
# parallelStartMulticore(cpus = ncpus) # not on windows
parallelStartSocket(ncpus)

res = tuneParams(learner =  makeLearner("regr.xgboost"),
                 task = regr.task.train,
                 resampling = rdesc,
                 measures = list(mae, setAggregation(mae, test.sd)),
                 par.set = ps.mbo,
                 control = tune.ctrl,
                 show.info = FALSE)
parallelStop()

print(res)
```