The best of general-purpose prediction with R

Zach Kurtz

2018-04-24

Zach's little world of R

Data stores: csv, mongolite, RSQLite

Exploratory analysis: data.table, base graphics

Prediction: **lightgbm**

Model tuning: **mlrMBO**

Presenting: https://github.com/zkurtz/useR_meetup_2018_04

Simulate a trivial dataset

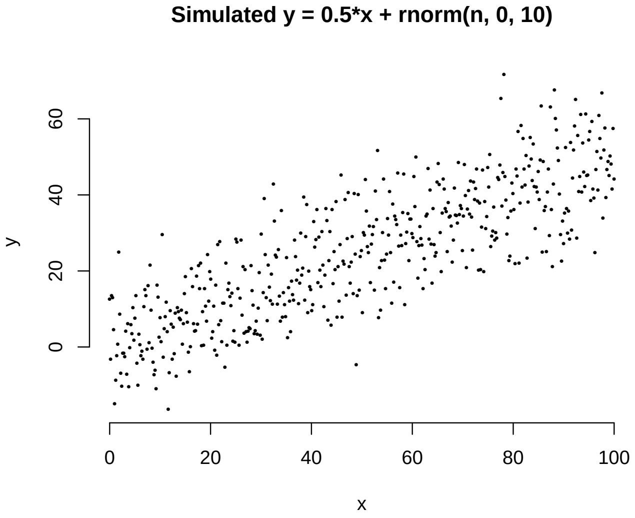

Figure 1: Data generated from a linear model

Introduction to lightgbm

Goal: Predict outcome y from predictor(s) x

Basic building block: a decision tree

The first decision tree will have errors/residuals

The second tree reduces the errors of the first

Successive trees keep ‘chipping away’ at the errors

Running lightgbm

```r
# matrix X of features
# vector y of labels
library(lightgbm)
bst = lightgbm(
    data = X,
    label = y,
    num_leaves = 4,
    min_data_in_leaf = 1,
    learning_rate = 1,
    nrounds = 1,
    objective = "regression")
yhat = predict(bst, data = X)
```

Fitted LightGBM

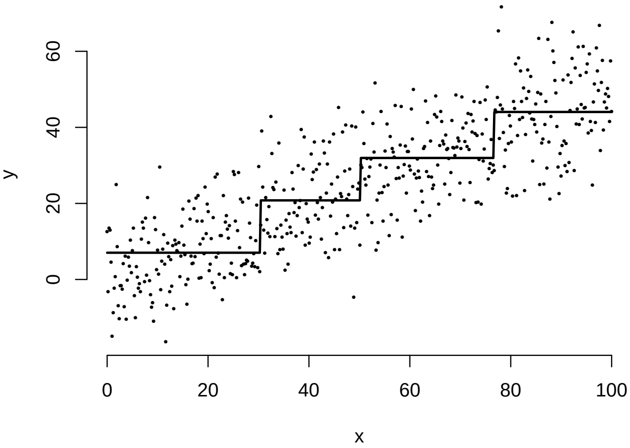

Figure 2: Fitted values for LightGBM with 4 leaves

Fitted LightGBM

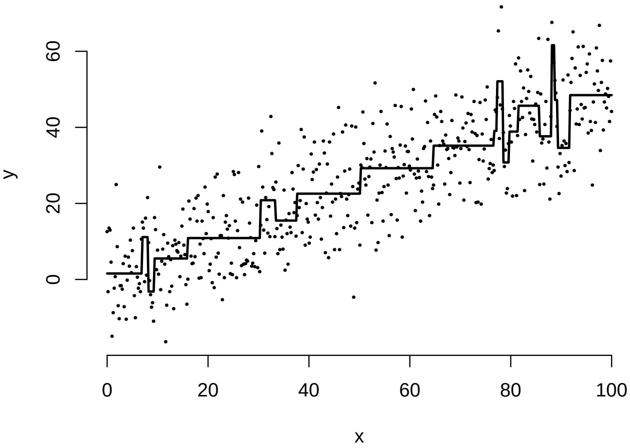

Figure 3: Fitted values for LightGBM with 20 leaves

Running lightgbm

```
bst = lightgbm(
    data = X,
    label = y,
    num_leaves = 20,
    min_data_in_leaf = 1,
    learning_rate = 0.3,
    nrounds = 1,
    objective = "regression")
```

Fitted LightGBM

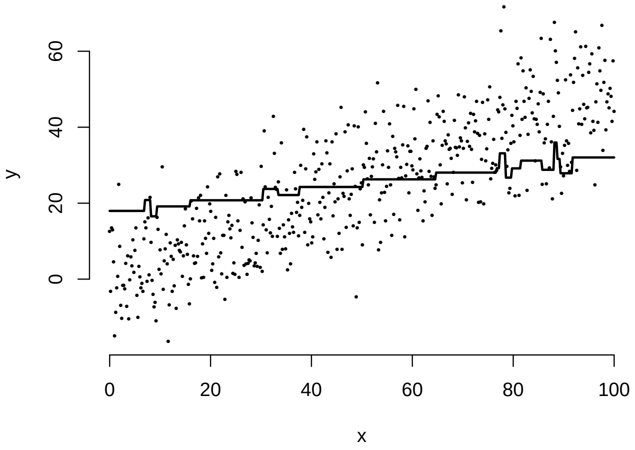

Figure 4: Decrease learning rate from 1 to 0.3

Fitted LightGBM

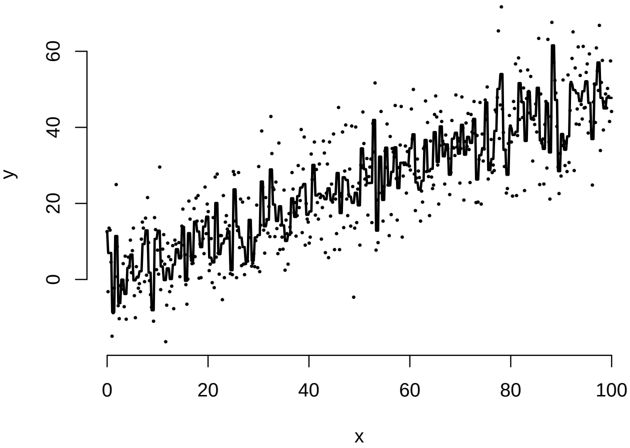

Figure 5: Increase nrounds from 1 to 100

Automatic hyperparameter tuning with mlrMBO

mlr: Machine learning in R – “a generic, object-oriented, and extensible framework” that provides a standardized interface to 160+ learners

mlrMBO: Model-based optimization

Tuning outline:

- Decide which parameters to tune
- Decide what range of values to consider for each parameter
- Define a loss function
- Choose a search strategy to minimize the loss

Automatic hyperparameter tuning with mlrMBO

```r
search_space = makeParamSet(
    makeIntegerParam("num_leaves", 2, 30),
    makeIntegerParam("min_data_in_leaf", 1, 50),
    makeNumericParam("learning_rate", 0.001, 0.4),
    makeIntegerParam("nrounds", 20, 200)
)
loss = function(h){
    cv = lightgbm::lgb.cv(
        data = X, label = y,
        num_leaves = h[1],
        min_data_in_leaf = h[2],
        learning_rate = h[3],
        nrounds = h[4],
        nfold = 5, verbose = -1, objective = "regression")
    return(tail(cv$record_evals$valid$l2$eval, 1)[[1]])
}
```

Automatic hyperparameter tuning with mlrMBO

Package the loss and the search space into one objective:

```
tuning_objective = makeSingleObjectiveFunction(
    name = "wrap_lightgbm",
    fn = objective,
    par.set = search_space,
    noisy = TRUE,
    minimize = TRUE
)
```

Automatic hyperparameter tuning with mlrMBO

```
# Decide what tuner to use and how long to run it
ctrl = makeMBOControl()
ctrl = setMBOControlTermination(ctrl, time.budget = 3600)
# Start tuning!
res = mbo(tuning_objective, control = ctrl)
```

Best 6 out of 487 iterations:

loss	num_leaves	min_data	learn_rate	nrounds	iter
98.6	3	30	0.094	167	300
98.9	3	32	0.252	41	233
99.1	3	30	0.092	148	340
99.1	3	30	0.098	140	312
99.4	3	31	0.093	149	334
99.6	3	31	0.092	122	343

Fitted LightGBM

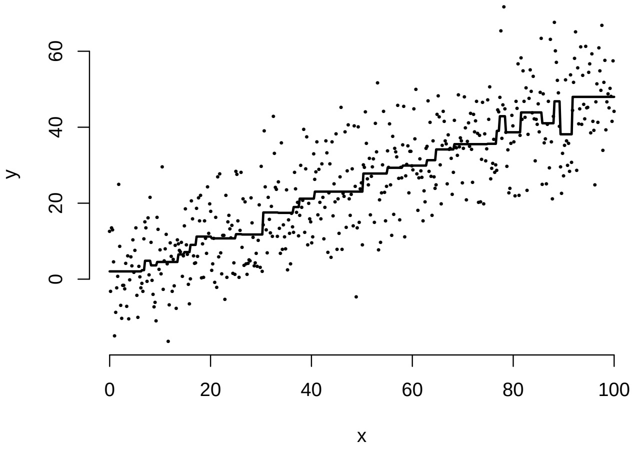

Figure 6: Fitted values for LightGBM after tuning

Some proposals for future work

GBMs:

- Try an adaptive learning rate (popular in deep learning)
- Use a variety of weak learners; not only trees

Hyperparameter tuning:

- Grow a battery of test cases to evaluate tuners
- Make hyperparameter transfer learning methods more accessible
- Invent the first (?) hyperparameter regularization method

Thank you!

Contact: `zkurtz` at `gmail`