Walk-through-examples

July 8, 2020

[35]:
```
import sys
import numpy
import random
import pandas as pd
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from IPython.display import Image
sys.path.append('/Users/zhengxiangyu/Documents/Reasearch/RecursivePartition/5.
 ↪Code_Examples/CODE_python_3.7/')
import SLRT_ag1_simple_v1 as lrt_simple
import SLRT_ag1_testing_v1 as lrt_testing
import SLRF_ag1_simple_v1 as lrf
import plot_tree as tw
import plot_tree_beta as tw_model
import math
from sklearn.datasets import load_boston
def num2str(x):
    dict_x4={1:'a',2:'b',3:'c'}
    return dict_x4[x]
# require x and y are numpy.array
def rmse(x, y):
    return numpy.sqrt(((x-y)**2).mean())
dir_save = '/Users/zhengxiangyu/Documents/Reasearch/RecursivePartition/5.
 ↪Code_Examples/EXAMPLE_python_3.7/eg1/res_independent/'
```

1 Simulated Data

1.1 generate data

$$m_1(X) = 3X_1\mathbb{I}\{X_2 > 15\} - 3X_1\mathbb{I}\{X_2 \leq 15\} - 3X_2\mathbb{I}\{X_2 > 10\} - 5X_2\mathbb{I}\{X_2 \leq 10\} + X_3\mathbb{I}\{X_1 > 10\} - X_3\mathbb{I}\{X_1 \leq 10\} + X_3\mathbb{I}\{X_4 \in \{\text{a, b}\}\} - 3X_3\mathbb{I}\{X_4 \in \{\text{c}\}\}. \quad (1)$$

[31]:
```
# 1.1 generate training data
numpy.random.seed(123)
x1 = numpy.random.random(1500)*20
```

```
x2 = numpy.random.random(1500)*20+5
x3 = numpy.random.random(1500)*10
x4 = numpy.random.randint(1,4,1500)

data_train=pd.DataFrame()
data_train['x1']=x1
data_train['x2']=x2
data_train['x3']=x3
data_train['x4']=[num2str(x) for x in x4]

y_1 =␣
 ↪(3*x1)*(x2>15)+(-3)*x1*(x2<=15)+(-3*x2)*(x2>=10)-5*x2*(x2<10)+(x3)*(x1>10)\
    -x3*(x1<10)
y_2 = x3.copy()
y_2[numpy.where(x4==3)] = -3*x3[numpy.where(x4==3)]
y = y_1+y_2+numpy.random.randn(1500)

data_train['y']=y
#1.2   generate testing data
numpy.random.seed(456)
x1 = numpy.random.random(500)*20
x2 = numpy.random.random(500)*20+5
x3 = numpy.random.random(500)*10
x4 = numpy.random.randint(1,4,500)

data_test=pd.DataFrame()
data_test['x1']=x1
data_test['x2']=x2
data_test['x3']=x3
data_test['x4']=[num2str(x) for x in x4]

y_1 =␣
 ↪(3*x1)*(x2>15)+(-3)*x1*(x2<=15)+(-3*x2)*(x2>=10)-5*x2*(x2<10)+(x3)*(x1>10)\
    -x3*(x1<10)
y_2 = x3.copy()
y_2[numpy.where(x4==3)] = -3*x3[numpy.where(x4==3)]
y = y_1+y_2+numpy.random.randn(500)

data_test['y']=y
# variables setting
var_all=numpy.array(['x1', 'x2', 'x3', 'x4', 'y'])
var_res=['y']
var_split=['x1', 'x2', 'x3', 'x4']
var_reg=['x1', 'x2', 'x3']
id_res = numpy.where(numpy.isin(var_all,var_res))[0]
id_cand_all = numpy.where(numpy.isin(var_all,var_split))[0]
id_value_all = numpy.where(numpy.isin(var_all,var_reg))[0]
```

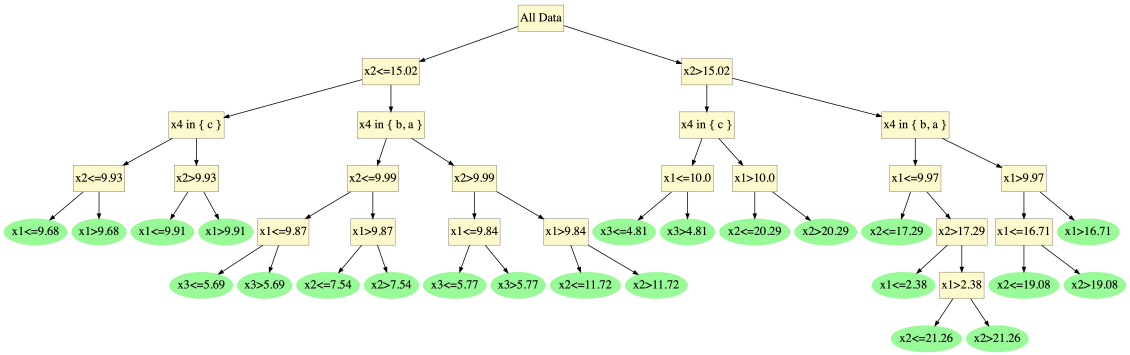
```
id_dis = numpy.where(numpy.isin(var_all,['x4']))[0]
```

1.2 build model and predict

[32]:
```
# ============================================================================
# SLRT with different stopping rules
# ============================================================================
# with simple stopping rules
tree_slrt = lrt_simple.grow_tree(data_train, id_value_all, id_cand_all, id_dis,␣
 ↪id_res, index = [1],\
                          labels = var_all, max_depth = 10, Nmin = 50, start =␣
 ↪True, sNsplit = 500)
pred_slrt = tree_slrt.predict_all(data_test[['x1','x2','x3','x4']])
print('the test error of Linear Rgression Tree: %.
 ↪3f'%rmse(pred_slrt,data_test['y']))
# plot the tree
filename = dir_save + 'eg1_simple_stop.jpg'
writer1 = tw.treeWriter(tree_slrt)
writer1.write(filename)
Image(filename = filename, width=500, height=300)
```

```
the test error of Linear Rgression Tree: 2.281
```

[32]:

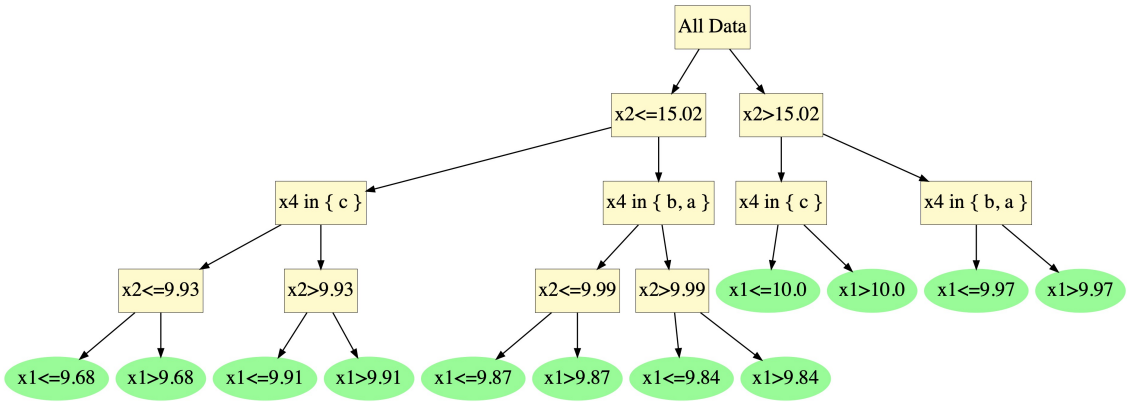


[34]:
```
# with hypothesis based stopping
tree_slrt_wise_stop = lrt_testing.grow_tree(data_train, id_value_all,␣
 ↪id_cand_all, id_dis, id_res, index = [1],\
                          labels = var_all, Nmin = 50, start = True, sNsplit =␣
 ↪500)
pred_slrt_wise_stop = tree_slrt.predict_all(data_test[['x1','x2','x3','x4']])
print('the test error of Linear Rgression Tree with testing-based stopping: %.
 ↪3f'%rmse(pred_slrt_wise_stop,data_test['y']))# plot the tree
filename = dir_save + 'eg1_testing_stop.jpg'
writer1 = tw.treeWriter(tree_lrt_wise_stop)
```

```
writer1.write(filename)
Image(filename = filename, width=500, height=300)
```

```
stoped by hypothesis testing, pvalue is 0.09918680010311506
stoped by hypothesis testing, pvalue is 0.09942678504412557
stoped by hypothesis testing, pvalue is 0.012978352432891837
stoped by hypothesis testing, pvalue is 0.012081213446230915
stoped by hypothesis testing, pvalue is 0.07873635544787812
stoped by hypothesis testing, pvalue is 0.048978268923673324
stoped by hypothesis testing, pvalue is 0.015259972035232282
stoped by hypothesis testing, pvalue is 0.09839751855658478
the test error of Linear Rgression Tree with testing-based stopping: 2.281
```

[34]:



[42]:

```
# ================================================================================
# CART: piecewise-linear regression tree
# ================================================================================
data_train_onehot = pd.get_dummies(data_train[['x1','x2','x3','x4']],␣
 ↪drop_first=True)
tree_cart = tree.DecisionTreeRegressor(max_depth=10, min_samples_leaf=50)
tree_cart.fit(data_train_onehot, data_train['y'])
data_test_onehot = pd.get_dummies(data_test[['x1','x2','x3','x4']],␣
 ↪drop_first=True)
pred_cart=tree_cart.predict(data_test_onehot)
print('the test error of CART: %.3f'%rmse(pred_cart, data_test['y']))
#tree.plot_tree(tree_cart)
```

```
the test error of CART: 9.752
```

2 Real Data: Boston house pricing

2.1 load data

[2]:
```
# load Data
X, y = load_boston(return_X_y=True)
data_boston = pd.concat([pd.DataFrame(X), pd.DataFrame(y)], axis=1,␣
↪ignore_index=True)
data_boston.columns = numpy.array(['crim', 'zn', 'indus', 'chas', 'nox', 'rm',␣
↪'age',
       'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'])
data_boston.loc[:,'log_medv'] = data_boston['medv'].apply(math.log)
data_boston.drop(labels = 'log_medv', axis=1)
# data_boston.info()
# create training data: 9/10 testing 1/10
random.seed(1)
random.random()
ind_train = random.sample(range(len(data_boston)),math.floor(len(data_boston)*0.
↪9))
ind_test = list(set(range(len(data_boston))) - set(ind_train))
data_train_boston = data_boston.iloc[ind_train,]
data_test_boston = data_boston.iloc[ind_test,]
data_boston.sort_values('log_medv',inplace = True)
data0=pd.read_csv('/Users/zhengxiangyu/Documents/Reasearch/RecursivePartition/1.
↪example/eg_pub_data/1.Boston/Boston.csv')
data1=data0.dropna()
data1.loc[:,'log_medv']=data1['medv'].apply(math.log)
# variables setting
var_all=numpy.array(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age',
       'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat', 'log_medv'])
var_res=['log_medv']
var_x=var_all[0:13]
var_split=var_x
var_reg=var_x
id_res = numpy.where(numpy.isin(var_all,var_res))[0]
id_cand_all = numpy.where(numpy.isin(var_all,var_split))[0]
id_value_all = numpy.where(numpy.isin(var_all,var_reg))[0]
id_dis = numpy.where(numpy.isin(var_all,['chas']))[0]
```

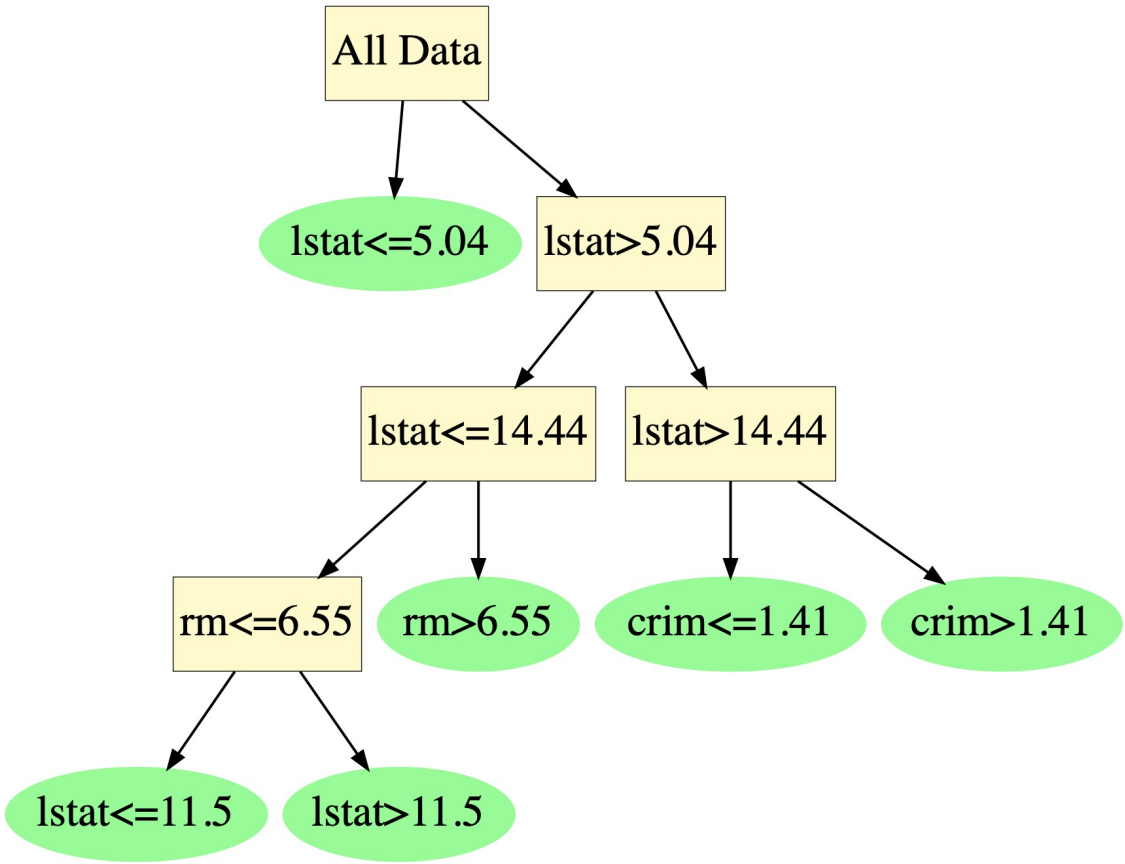
2.2 build model and predict

[4]:
```
# ============================================================================
# SLRT
# ============================================================================
# with simple stopping rules
```

```
tree_slrt = lrt_simple.grow_tree(data_train_boston, id_value_all, id_cand_all,␣
 ↪id_dis, id_res, index = [1],\
                          labels = var_all, Nmin = 60, start = True, sNsplit =␣
 ↪100)
pred_slrt = tree_slrt.predict_all(data_test_boston[var_x])
print('the test error of Linear Rgression Tree: %.3f'%rmse(pred_slrt,␣
 ↪data_test_boston[var_res[0]]))
# plot the tree
filename = dir_save + 'boston_slrt.jpg'
writer1 = tw.treeWriter(tree_slrt)
writer1.write(filename)
Image(filename = filename, width=300, height=300)
```

```
the test error of Linear Rgression Tree: 0.179
```

[4]:

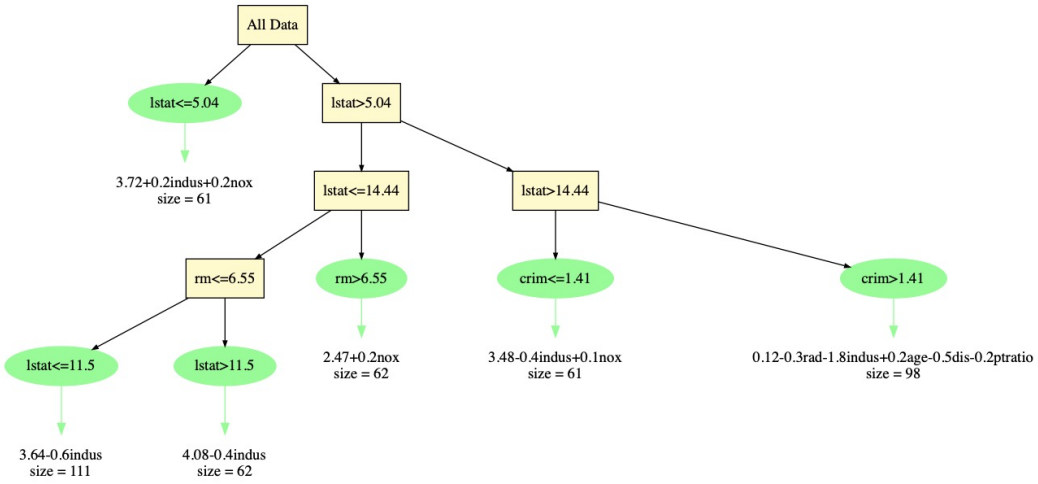


[5]:

```
# plot the tree with linear models on leaves
# only show parameters with |beta|>0.01
filename = dir_save + 'boston_slrt_beta.jpg'
writer1 = tw_model.treeWriter(tree_slrt, labels = var_reg)
writer1.write(filename)
```

```
Image(filename = filename, width=1000, height=700)
```

[5]:



[17]:
```
# ================================================================================
# CART
# ================================================================================
tree_cart_boston = tree.DecisionTreeRegressor(min_samples_leaf=60)
tree_cart_boston.fit(data_train_boston[var_x], data_train_boston[var_res])
pred_cart=tree_cart_boston.predict(data_test_boston[var_x])
print('the test error of Regression Tree (CART): %.3f'%rmse(pred_cart,␣
 ↪data_test_boston[var_res[0]]))
```

```
the test error of Regression Tree (CART): 0.216
```

[18]:
```
# ================================================================================
# Random Forest based on CART
# ================================================================================
rf = RandomForestRegressor(n_estimators = 50, min_samples_leaf = 60, \
                           max_features = int(13**(0.75)), random_state = 0)
rf.fit(data_train_boston[var_x], data_train_boston[var_res[0]])
pred_rf = rf.predict(data_test_boston[var_x])
print('the test error of RandomForest: %.3f'%rmse(pred_rf,␣
 ↪data_test_boston[var_res[0]]))
```

```
the test error of RandomForest: 0.222
```

[19]:
```
# ================================================================================
# Random Forest based on SLRT
# ================================================================================
slrf = lrf.make_forest(data_train_boston, 50, id_value_all, id_cand_all,␣
 ↪id_dis, id_res, \
```

```
                          labels = var_all, Nmin = 60, sNsplit = 100, mi=0.75)
pred_slrf = slrf.predict_all(data_test_boston)
print('the test error of LinearRandomForest: %.3f'%rmse(pred_slrf,␣
 ↪data_test_boston[var_res[0]]))
```

```
the test error of LinearRandomForest: 0.150
```