Weekly report

FANGYI GUO

Weight bug

Reason: in Yu's code, he normalized $\Sigma w = 1$. So if you have a large sum of weight, then weight number would be very small.

Code: in def importFromFileList()

w=[1*i/sumOfWeight for i in w]

Change it to

w=[1e5*i/sumOfWeight for i in w]

Learning score after debug:

- Xgb: 0.2882
- BDTG: 0.2874
- Ada Boost: 0.2082

Parameter tuning

Input dataset:

- Signal: VBF Higgs, H->yy
- Background: di-photon continuum background.
- 6 variables, input training tree and test tree separately (BDTtrain, BDTtest).

Available method:

- Learning curve
- Grid Search

Criteria: learning score($R^2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$)

My code: /publicfs/atlas/atlasnew/higgs/hgg/guofy/ML_training/WorkDir/python

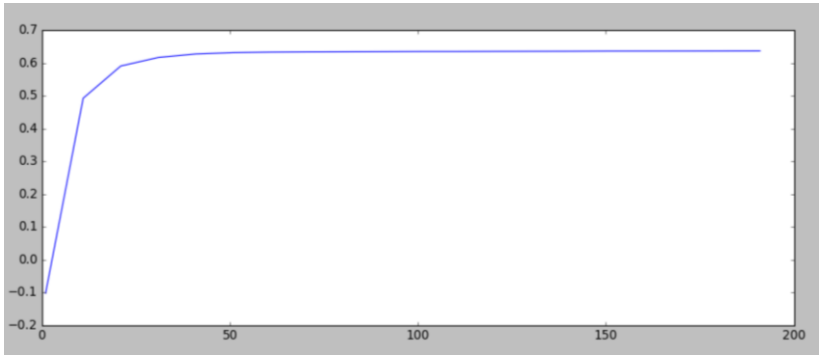
Parameter tuning

Learning curve: first step is always tuning n_estimator (NTree)

- For each n_estimator, build model and get score. Draw the curve of n_estimator vs. score

  code: Xgboost.py

```
#_______Learning curve_____________
scorel = []
for i in range(0,500,50):
    reg_lc = XGBR(n_estimators=i+1, n_jobs=-1, random_state=42)
    score = CVS(reg_lc, X_test, y_test, cv=3).mean()
    scorel.append(score)
print(max(scorel), (scorel.index(max(scorel))*10)+1)
plt.figure(figsize=[20,5])
plt.plot(range(1,501,50),scorel)
plt.show()
```



Parameter tuning

Grid search:

- Try the combination of different parameters within sklearn

  Code: DrawPlots.py

```
#_____________________________________________________________
def doGridSearchCV(clf,X,y,method,outputDir):
    logging.info("do grid search of"+method)
    param_range={"n_estimators":[100,500, 800],
                 "learning_rate":[.2,.5,.8]}
    if method=="BDTG":
        param_range["max_depth"]=[2,3,4]
    if method=="XGBoost":
        param_range["max_depth"]=[2,3,4]
        param_range["gamma"]=[0,.2,.4,.6,.8,1.]
        param_range["lambda"]=[.5,.8,1.,1.5,1.8]
    grid_search = GridSearchCV(clf, param_grid=param_range,verbose=2,cv=3)
    grid_search.fit(X,y)
    logging.info("grid search result:")
    logging.info(grid_search.cv_results_)
    logging.info("Best estimator:")
    logging.info(grid_search.best_estimator_)

doGridSearchCV(reg,X_train,y_train,"XGBoost",outputDir)
doGridSearchCV(bdtg, X_train, y_train, "BDTG", outputDir)
```

Application

import joblib

Save your training model (Xgboost.py):

```
reg = XGBR(n_estimators=100)
reg.fit(X_train, y_train,sample_weight=wi_train)
joblib.dump(reg,outputDir+"/model/XGBoost.model")
```

- TMVA: VBF_vs_yy_Cfg2_ITK_BDTG.weights.xml

Load the model(Xgboost_appli.py):

reg_xg = joblib.load(outputDir+ "/model/XGBoost.model")

- TMVA:

```
TMVA::Reader *reader = new TMVA::Reader();
 reader->AddVariable("pTt_yy",&pTt_yy);
 reader->BookMVA("BDTG","VBF_vs_yy_Cfg2_ITK_BDTG.weights.xml");
```

Application

Write into ROOT file

```
BDTResponse = np.empty((1), dtype="float32")
pTt_yy = np.empty((1), dtype="float32")
f=ROOT.TFile("BDTresponse_VBF_xgb.root","recreate")
t=ROOT.TTree("Upgrade","Upgrade")
t.Branch("pTt_yy", pTt_yy, 'pTt_yy/F’)
t.Branch("BDTResponse",BDTResponse, 'BDTResponse/F’)
for i in range(len(signal_use_dataset)):
    BDTResponse[0] = reg_xg.predict(signal_use_dataset)[i]
    pTt_yy[0] = signal_use_dataset[i,0]
    t.fill()
t.Write()
f.Write()
```

Application

Output: BDTresponse_VBF_xgb.root

```
TFile**         BDTresponse_VBF_xgb.root
 TFile*         BDTresponse_VBF_xgb.root
  KEY: TTree    Upgrade;2       Upgrade
  KEY: TTree    Upgrade;1       Upgrade
root [2] Upgrade->Show(1)
======> EVENT:1
 pTt_yy          = 64170.4
 m_jj            = 837796
 DeltaEta_jj     = 3.90912
 DeltaPhi_yy_jj  = 3.03128
 Zepp            = 0.343662
 minDeltaR_y_j   = 2.15921
 weight          = 4.04973
 BDTResponse     = 0.910029
```