CSDN
不 止 于 代 码

3.6 XGBoost使用指南 (一)

CSDN学院

2017年11月

XGBoost的使用

- 直接调用XGBoost
  - import xgboost as xgb
- 与scikit-learn一起使用
  - from xgboost import XGBClassifier

独立调用XGBoost

- 1. 数据→Dmatrix
- 2. 设置参数
- 3. 模型训练：train/cv
  - 校验集
  - 交叉验证
  - 过早停止
  - 特征重要性/模型选择
- 4. 模型测试

XGBoost的数据接口

- XGBoost加载的数据存储在对象Dmatrix中，做了存储效率和运行速度的优化
- 支持三种接口
  - libsvm txt格式文件
  - 常规矩阵（ numpy 2D array ）
  - xgboost binary buffer file

Dmatrix

class `xgboost.` `DMatrix` **(*data*, *label=None*, *missing=None*, *weight=None*, *silent=False*, *feature_names=None*, *feature_types=None*, *nthread=None*)**

Bases: `object`

Data Matrix used in XGBoost.

DMatrix is a internal data structure that used by XGBoost which is optimized for both memory efficiency and training speed. You can construct DMatrix from numpy.arrays

`feature_names`

Get feature names (column labels).

Returns: **feature_names**
Return type: list or None

`feature_types` ¶

Get feature types (column types).

Returns: **feature_types**
Return type: list or None

1. 读取数据

- *将数据从文件中读出，并为XGBoost训练准备好*

- # 设置数据路径，数据在*xgbboost*安装的路径下的demo目录
- *# read in data，训练集和测试集*
- dpath = ‘./data/’
- dtrain = xgb.DMatrix(dpath +'agaricus.txt.train')
- dtest = xgb.DMatrix(dpath +'agaricus.txt.test')

- 该数据为libsvm格式的文本数据，libsvm的文件格式（稀疏特征）
  - 每一行为一个样本：1 101:1.2 102:0.03 0 1:2.1 10001:300 10002:400 ...
    - 开头的“1”是样本的标签。“101”和“102”为特征索引，'1.2'和'0.03' 为特征的值。
    - 在两类分类中，用“1”表示正样本，用“0” 表示负样本。也支持[0,1]表示概率用来做标签，表示为正样本的概率
- XGBoost加载的数据存储在对象Dmatrix中

2. 设置训练参数

- *# specify parameters via map*
- param = {'max_depth':2, 'eta':1, 'silent':0, 'objective':'binary:logistic' }

- max_depth： 树的最大深度。缺省值为6，取值范围为：[1,∞]
- eta：学习率/收缩因子。缺省值为0.3，取值范围为：[0,1]

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}_i) + \eta\phi_m(\mathbf{x}_i)$$

- silent: 0表示打印出运行时信息，取1时表示以缄默方式运行，不打印运行时信息。缺省值为0
- objective： 定义学习任务及相应的学习目标， “binary:logistic” 表示二分类的逻辑回归问题，输出为概率。

更多参数说明请见参数调优部分

3. 模型训练

- # 设置*boosting*迭代计算次数
- num_round = 2
- bst = xgb.train(param, dtrain, num_round)

xgboost.train(*params*, *dtrain*, *num_boost_round=10*, *evals=()*, *obj=None*, *feval=None*, *maximize=False*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *xgb_model=None*, *callbacks=None*, *learning_rates=None*)

Train a booster with given parameters.

Parameters:

- **params** (*dict*) – Booster params.
- **dtrain** (*DMatrix*) – Data to be trained.
- **num_boost_round** (*int*) – Number of boosting iterations.
- **evals** (*list of pairs (DMatrix, string)*) – List of items to be evaluated during training, this allows user to watch performance on the validation set.
- **obj** (*function*) – Customized objective function.
- **feval** (*function*) – Customized evaluation function.
- **maximize** (*bool*) – Whether to maximize feval.
- **early_stopping_rounds** (*int*) – Activates early stopping. Validation error needs to decrease at least every <early_stopping_rounds> round(s) to continue training. Requires at least one item in evals. If there's more than one, will use the last. Returns the model from the last iteration (not the best one). If early stopping occurs, the model will have three additional fields: bst.best_score, bst.best_iteration and bst.best_ntree_limit. (Use bst.best_ntree_limit to get the correct value if num_parallel_tree and/or num_class appears in the parameters)
- **evals_result** (*dict*) –
  This dictionary stores the evaluation results of all the items in watchlist. Example: with a watchlist containing [(dtest,'eval'), (dtrain,'train')] and a parameter containing ('eval_metric': 'logloss') Returns: {'train': {'logloss': ['0.48253', '0.35953']},

  —

  > 'eval': {'logloss': ['0.480385', '0.357756']}}

`xgboost.train` (*params*, *dtrain*, *num_boost_round=10*, *evals=()*, *obj=None*, *feval=None*, *maximize=False*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *xgb_model=None*, *callbacks=None*, *learning_rates=None*)

- **verbose_eval** (*bool or int*) – Requires at least one item in evals. If *verbose_eval* is True then the evaluation metric on the validation set is printed at each boosting stage. If *verbose_eval* is an integer then the evaluation metric on the validation set is printed at every given *verbose_eval* boosting stage. The last boosting stage / the boosting stage found by using *early_stopping_rounds* is also printed. Example: with verbose_eval=4 and at least one item in evals, an evaluation metric is printed every 4 boosting stages, instead of every boosting stage.
- **learning_rates** (*list or function (deprecated - use callback API instead)*) – List of learning rate for each boosting round or a customized function that calculates eta in terms of current number of round and the total number of boosting round (e.g. yields learning rate decay)
- **xgb_model** (*file name of stored xgb model or 'Booster' instance*) – Xgb model to be loaded before training (allows training continuation).
- **callbacks** (*list of callback functions*) – List of callback functions that are applied at end of each iteration. It is possible to use predefined callbacks by using xgb.callback module. Example: [xgb.callback.reset_learning_rate(custom_rates)]

Returns: **booster**

Return a trained booster model

cv参数

xgboost.cv(*params*, *dtrain*, *num_boost_round=10*, *nfold=3*, *stratified=False*, *folds=None*, *metrics=()*, *obj=None*, *feval=None*, *maximize=False*, *early_stopping_rounds=None*, *fpreproc=None*, *as_pandas=True*, *verbose_eval=None*, *show_stdv=True*, *seed=0*, *callbacks=None*, *shuffle=True*)

Cross-validation with given parameters.

Parameters:

- **params** (*dict*) – Booster params.
- **dtrain** (*DMatrix*) – Data to be trained.
- **num_boost_round** (*int*) – Number of boosting iterations.
- **nfold** (*int*) – Number of folds in CV.
- **stratified** (*bool*) – Perform stratified sampling.
- **folds** (*a KFold or StratifiedKFold instance*) – Sklearn KFolds or StratifiedKFolds.
- **metrics** (*string or list of strings*) – Evaluation metrics to be watched in CV.
- **obj** (*function*) – Custom objective function.
- **feval** (*function*) – Custom evaluation function.
- **maximize** (*bool*) – Whether to maximize feval.
- **early_stopping_rounds** (*int*) – Activates early stopping. CV error needs to decrease at least every <early_stopping_rounds> round(s) to continue. Last entry in evaluation history is the one from best iteration.
- **fpreproc** (*function*) – Preprocessing function that takes (dtrain, dtest, param) and returns transformed versions of those.
- **as_pandas** (*bool, default True*) – Return pd.DataFrame when pandas is installed. If False or pandas is not installed, return np.ndarray

cv参数

xgboost. cv (***params*, *dtrain*, *num_boost_round=10*, *nfold=3*, *stratified=False*, *folds=None*, *metrics=()*, *obj=None*, *feval=None*, *maximize=False*, *early_stopping_rounds=None*, *fpreproc=None*, *as_pandas=True*, *verbose_eval=None*, *show_stdv=True*, *seed=0*, *callbacks=None*, *shuffle=True***)

- **verbose_eval** (*bool, int, or None, default None*) – Whether to display the progress. If None, progress will be displayed when np.ndarray is returned. If True, progress will be displayed at boosting stage. If an integer is given, progress will be displayed at every given *verbose_eval* boosting stage.
- **show_stdv** (*bool, default True*) – Whether to display the standard deviation in progress. Results are not affected, and always contains std.
- **seed** (*int*) – Seed used to generate the folds (passed to numpy.random.seed).
- **callbacks** (*list of callback functions*) –
  List of callback functions that are applied at end of each iteration.

  It is possible to use predefined callbacks by using xgb.callback module. Example: [xgb.callback.reset_learning_rate(custom_rates)]

  shuffle : *bool*

  Shuffle data before creating folds.

Returns: **evaluation history**

Return list(string)

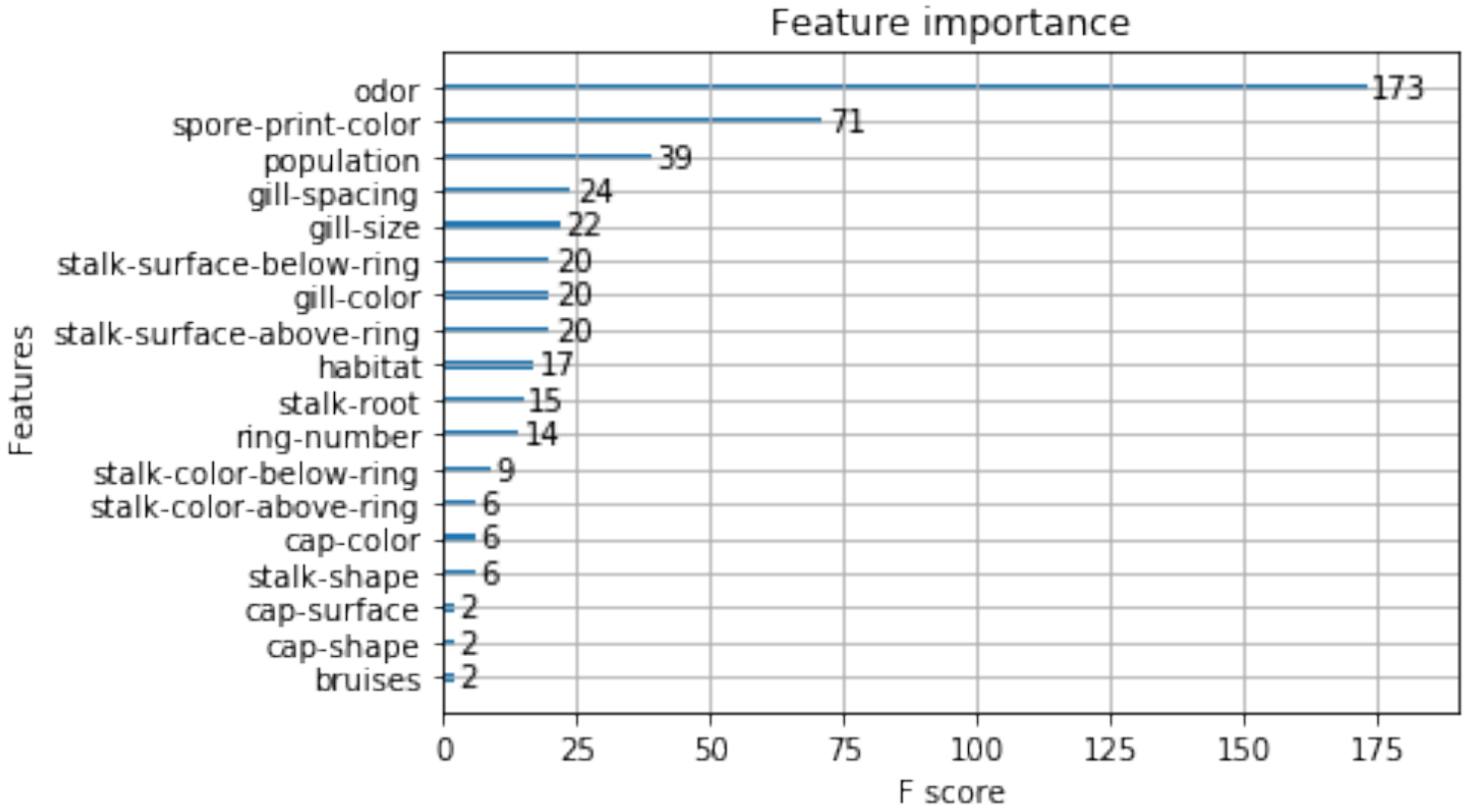
特征重要性

- 采用树集成学习（如Gradient Boosting）的好处之一是可以从训练好的预测模型得到特征的重要性
- 单棵树中的特征重要性：每个特征分裂点对性能的提升量，并用每个结点的样本数加权。
  - 性能测量可以是用于选择分裂点的指标（纯度）或其他更特别的误差函数
- 整个模型中的特征重要性：模型中每棵树的平均

特征重要性(cont.)

- 在XGBoost中已经自动算好，存放在feature_importances_
  - from matplotlib import pyplotpyplot.bar(range(len(model_XGB.feature_importances_)), model_XGB.feature_importances_)
  - pyplot.show()
  - 按特征顺序打印
- 还可以使用XGBoost内嵌的函数，按特征重要性排序
  - from xgboost import plot_importance
  - plot_importance(model_XGB)
  - pyplot.show()

► 例：蘑菇数据集的特征重要性



CART_RF_Mushroom.ipynb

特征选择

- 可以根据特征重要性进行特征选择
  - from sklearn.feature_selection import SelectFromModel

- 输入一个（在全部数据集上）训练好的模型
- 给定阈值，重要性大于阈值的特征被选中
  - selection = SelectFromModel(model, threshold=thresh, prefit=True)
    select_X_train = selection.transform(X_train)

例：蘑菇数据集上的特征选择

- Thresh=0.000, n=22, Accuracy: 100.00%
- Thresh=0.000, n=22, Accuracy: 100.00%
- Thresh=0.000, n=22, Accuracy: 100.00%
- Thresh=0.000, n=22, Accuracy: 100.00%
- Thresh=0.004, n=18, Accuracy: 100.00%
- Thresh=0.004, n=18, Accuracy: 100.00%
- Thresh=0.004, n=18, Accuracy: 100.00%
- Thresh=0.013, n=15, Accuracy: 100.00%
- Thresh=0.013, n=15, Accuracy: 100.00%
- Thresh=0.013, n=15, Accuracy: 100.00%
- Thresh=0.019, n=12, Accuracy: 100.00%
- Thresh=0.030, n=11, Accuracy: 100.00%
- Thresh=0.032, n=10, Accuracy: 100.00%
- Thresh=0.036, n=9, Accuracy: 100.00%
- Thresh=0.043, n=8, Accuracy: 100.00%
- Thresh=0.043, n=8, Accuracy: 100.00%
- Thresh=0.043, n=8, Accuracy: 100.00%
- **Thresh=0.047, n=5, Accuracy: 100.00%**
- Thresh=0.051, n=4, Accuracy: 99.51%
- Thresh=0.083, n=3, Accuracy: 99.45%
- Thresh=0.152, n=2, Accuracy: 99.45%
- Thresh=0.370, n=1, Accuracy: 98.58%

5个特征就足够好了：
odor
spore-print-color
population
gill-spacing
gill-size

4. 预测

- 模型训练好后，可以进行预测
  - XGBoost预测的输出是概率，输出值是样本为第一类的概率 → 将概率值转换为0或1

```
preds = bst.predict(dtest)
y_pred = [round(value) for value in preds]
y_test = dtest.get_label()
test_accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy: %.2f%%" % (test_accuracy * 100.0))
```

XGBoost资源

- XGBoost官方文档：https://xgboost.readthedocs.io/en/latest/
  - Python API：http://xgboost.readthedocs.io/en/latest/python/python_api.html
  - 参数说明： http://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters
- Github： https://github.com/dmlc/xgboost
  - 很多有用的资源：https://github.com/dmlc/xgboost/blob/master/demo/README.md
  - GPU加速：https://github.com/dmlc/xgboost/blob/master/plugin/updater_gpu/README.md
  - 示例代码： https://github.com/dmlc/xgboost/tree/master/demo/guide-python
- XGBoost原理：XGBoost: A Scalable Tree Boosting System
  - https://arxiv.org/abs/1603.02754

其他资源

- XGBoost参数调优：
  - https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
  - 中文版：http://blog.csdn.net/u010657489/article/details/51952785
  - http://blog.csdn.net/han_xiaoyang/article/details/52665396

- Owen Zhang, Winning Data Science Competitions
  - https://www.slideshare.net/OwenZhang2/tips-for-data-science-competitions?from_action=save
- XGBoost User Group：
  - https://groups.google.com/forum/#!forum/xgboost-user/

CSDN
不 止 于 代 码

THANK YOU

AI100