An *Introduction* to dmlc XGBoost

2016年秋季北京大学研究生算法课

两句话介绍 XGBoost

- 基于Tree Boosting 方法的高效、通用、准确的机器学习系统
- 机器学习比赛的**大！ 杀！ 器！**
  - *Machine Learning Challenge Winning Solutions*

XGBoost 小档案

- 作者：陈天奇
  - 上海交通大学毕业，华盛顿大学博士在读
  - 研究领域：大规模机器学习
- 项目主页
  - https://xgboost.readthedocs.io/en/latest/
- Github 地址
  - https://github.com/dmlc/xgboost
- 论文链接
  - xgboost: eXtreme Gradient Boosting

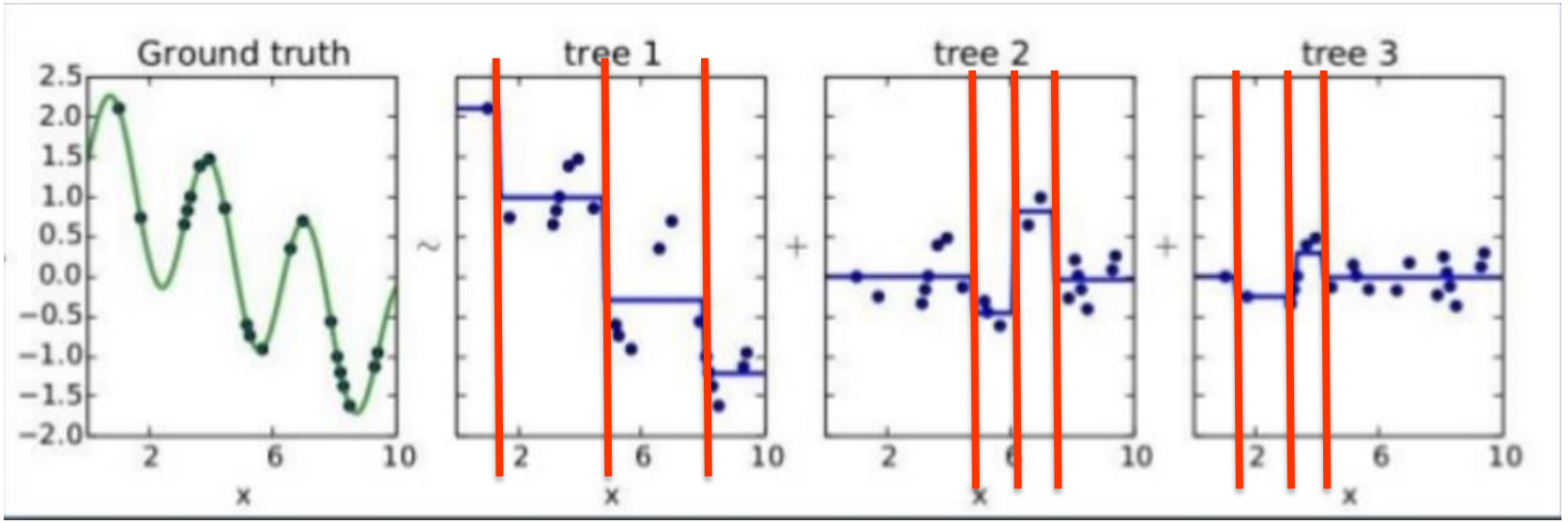
XGBoost 特性

- 面向多种任务场景
  - 分类任务
  - 回归任务
  - 排序任务
  - 用户自定义任务 - demo
- 支持单机/并行/分布式，可扩展性强
- 支持主流操作系统Linux/Mac/Windows
- 支持C/C++/Python/R/Java/Scala 等主流编程语言
- 支持Hadoop/Spark等分布式平台

XGBoost 原理 – Boosting

- Boosting 方法
  - “三个臭皮匠，顶个诸葛亮”
  - 将多个能力较弱的机器学习模型组合成为单个能力较强的机器学习模型

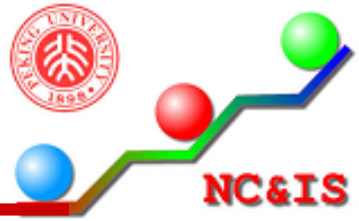
- Gradient Tree Boosting
  - Gradient boosting: $y_i^{(t)} = \sum_{i=1}^{t} f_k(x_i) = y_i^{(t-1)} + f_t(x_i)$
  - eg. Gradient boosting on CART



XGBoost 原理 – Loss Function

- Loss Function – 度量模型在给定数据集上的预测能力
  - 平方loss - $L(y,\hat{y}) = (y-\hat{y})^2$
  - 0-1互信息loss - $L(y,\hat{y}) = \hat{y}\log\frac{1}{y} + (1-\hat{y})\log\frac{1}{1-y}$
  - ……
- XGBoost 目标：
  - $\min_{f_1,f_2\cdots f_t}\{\sum_{i=1}^{n} L\left(\sum_{j=1}^{t} f_j(x_i), \hat{y_i}\right) + \sum_{j=1}^{t}\Omega(f_j)\}$
  - 训练决策树使Loss 最小

XGBoost 安装 - Python on Linux



- 环境要求
  - 编译器 - g++ (版本号 >= 4.6)
    - 其他支持c++11 & OpenMP的编译器亦可
  - 代码管理 - git
  - 语言支持 - Python
    - Ubuntu/Debian: `sudo apt-get install g++ python git`
- 安装操作
  - 下载源代码 `git clone --recursive https://github.com/dmlc/xgboost`

不能直接下载源码包

  - 编译共享库 `cd xgboost; make -j4`

4线程编译

  - 安装相关python包 `sudo apt-get install python-numpy python-scipy`
  - 安装到Python `cd python-package; sudo python setup.py install`

代码演示

- 0-1分类问题： 蘑菇毒性预测
- 多分类问题：皮肤病诊断问题
- 回归问题：CPU性能预测问题

0-1分类问题：蘑菇毒性预测

- 问题背景
  - mushroom dataset from UCI machine learning repository
  - 根据蘑菇的外观属性预测其是否可以食用
  - 数据规模：训练集6513，测试集1611
  - 外观特征（共22个）
    - 1.伞形：钟形= b，锥形= c，凸= x，平面= f，把手形= k，凹陷= s
    - 2.伞面：纤维= f，沟槽= g，鳞屑= y，平滑= s
    - 3.伞帽颜色：棕色= n，浅黄色= b，肉桂= c，灰色=g，绿= r，粉色= p，紫色= u，红色= e，白= w，黄= y
    - ……

0-1分类问题：蘑菇毒性预测

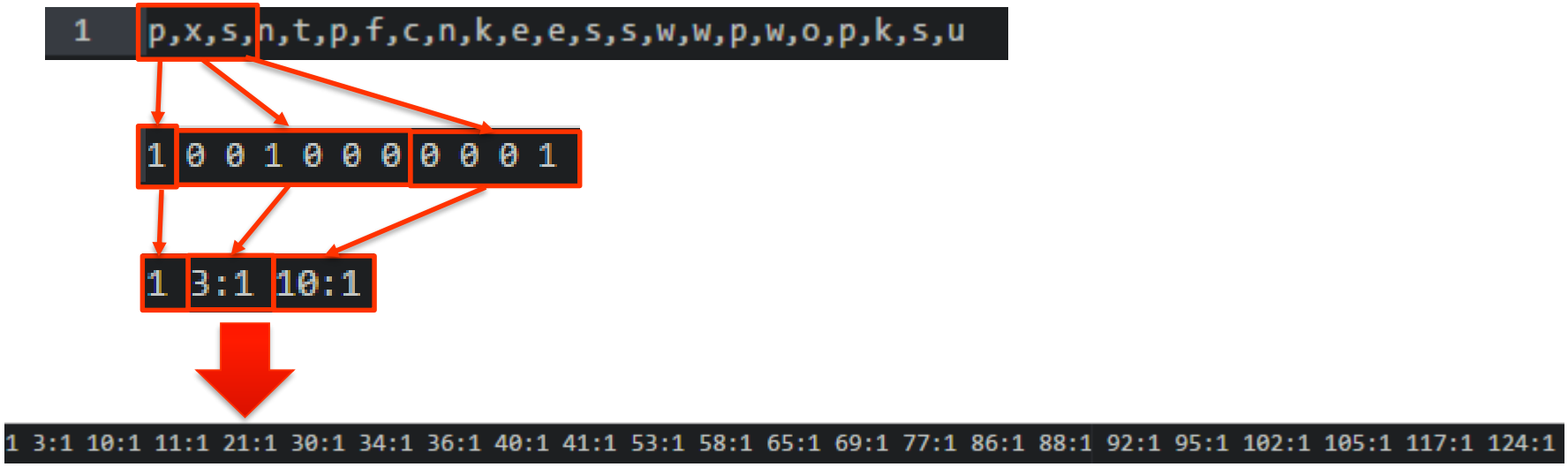
- 数据格式
  - 第1列 – 蘑菇毒性
    - p – 有毒的，e – 可食用的
  - 第2~23列 – 蘑菇外观属性

```
1   p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u
```

    - 毒性：有，伞形：凸，伞面：平滑，伞帽颜色：棕色

0-1分类问题：蘑菇毒性预测

- 数据预处理
  - One-hot Encoding
  - 输出为 XGBoost数据格式



0-1分类问题：蘑菇毒性预测

用XGBoost进行训练和预测

```
import xgboost as xgb
dtrain = xgb.DMatrix('agaricus.txt.train')
dtest = xgb.DMatrix('agaricus.txt.test')
param = {'max_depth': 3,
         'eta': 1.0,
         'gamma': 1.0,
         'min_child_weight': 1,
         'save_period': 0,
         'booster': 'gbtree',
         'objective': 'binary:logistic'}
num_round = 2
watchlist = [(dtest, 'eval'), (dtrain, 'train')]
bst = xgb.train(param, dtrain, num_round, watchlist)
preds = bst.predict(dtest)
write_pred('pred2.txt', preds)
bst.dump_model('dump2.nice.txt', 'featmap.txt')
```

导入模块 **xgboost**

DMatrix：xgboost 内部存储训练/测试数据的数据结构

单颗决策树的最大深度

学习步长/收缩因子，用来防止过拟合，取值范围(0, 1]

正则项，叶节点代价因子

单个叶节点最小允许的数据个数（权值和）

每一轮不保存中间结果

模型：gradient boosted tree

目标任务：0-1分类；Loss：0-1互信息Loss

训练轮数

每轮评测的训练集和测试集

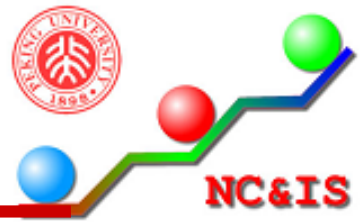
训练

预测

输出预测结果

输出模型

0-1分类问题：蘑菇毒性预测

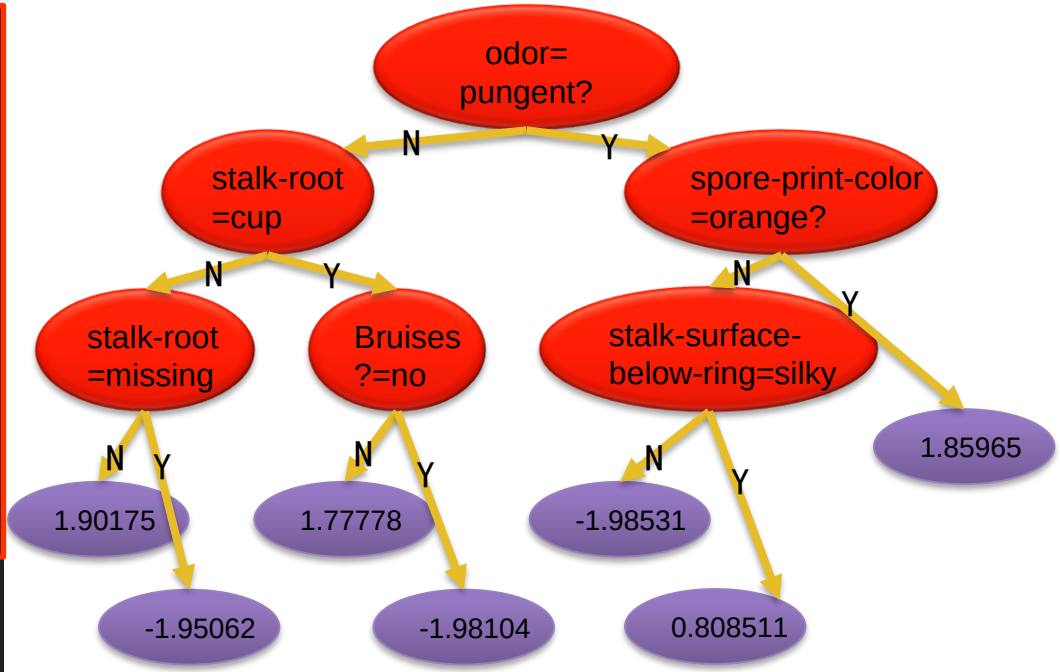



```
#!/bin/bash
# map feature using indicator encoding, also produce featmap.txt
python mapfeat.py
# split train and test
python mknfold.py agaricus.txt 1
# use xgboost to train, predict & dump model
python runxgb.py
```

```
leckie@SLOT3-10:~/git/xgboost/demo/binary_classification$ ./runpy.sh
[18:39:14] 6513x127 matrix with 143286 entries loaded from agaricus.txt.train
[18:39:14] 1611x127 matrix with 35442 entries loaded from agaricus.txt.test
[18:39:14] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[0]     eval-error:0.016139     train-error:0.014433
[18:39:14] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 0 pruned nodes, max_depth=3
[1]     eval-error:0    train-error:0.001228
```

0-1分类问题：蘑菇毒性预测

```
booster[0]:
0:[odor=pungent] yes=2,no=1
  1:[stalk-root=cup] yes=4,no=3
    3:[stalk-root=missing] yes=8,no=7
      7:leaf=1.90175
      8:leaf=-1.95062
    4:[bruises?=no] yes=10,no=9
      9:leaf=1.77778
      10:leaf=-1.98104
  2:[spore-print-color=orange] yes=6,no=5
    5:[stalk-surface-below-ring=silky] yes=12,no=11
      11:leaf=-1.98531
      12:leaf=0.808511
    6:leaf=1.85965
booster[1]:
0:[odor=pungent] yes=2,no=1
  1:[bruises?=no] yes=4,no=3
    3:leaf=1.1457
    4:[gill-spacing=crowded] yes=8,no=7
      7:leaf=-6.87558
      8:leaf=-0.127376
  2:[spore-print-color=orange] yes=6,no=5
    5:[gill-size=narrow] yes=10,no=9
      9:leaf=-0.0386054
      10:leaf=-1.15275
    6:leaf=0.994744
```



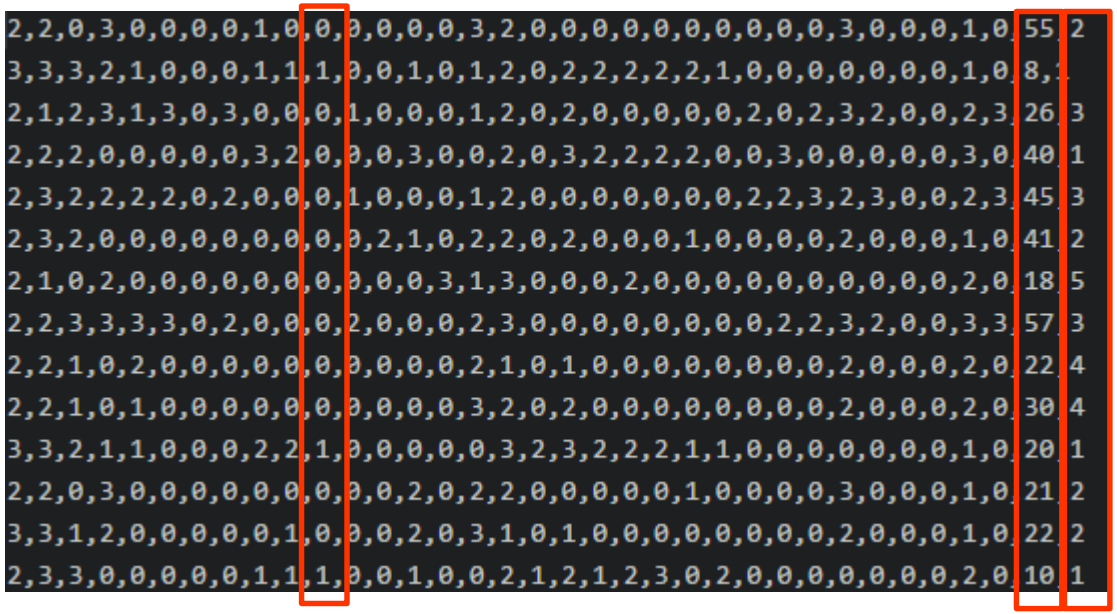

```
0   cap-shape=bell  i
1   cap-shape=conical   i
2   cap-shape=convex    i
3   cap-shape=flat  i
4   cap-shape=knobbed   i
5   cap-shape=sunken    i
6   cap-surface=fibrous i
```

多分类问题－皮肤病诊断问题

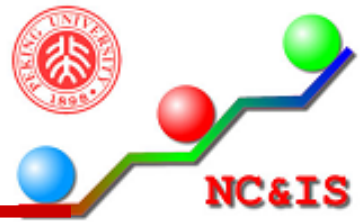
- 问题背景
  - UCI Dermatology dataset
  - 鳞状红斑狼疮类疾病(erythemato-squamous diseases, ESD)的诊断
    - 六种疾病：银屑病，脂溢性皮炎，扁平苔癣，玫瑰糠疹，慢性皮炎和毛发红糠疹（psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris）
    - 可以通过活体检查获得病人的外部特征，然而最终的确诊要靠显微镜下的组织病理学手段。
    - 任务：通过外部特征预测皮肤病种类
  - 数据规模：共365条数据

多分类问题－皮肤病诊断问题

- 数据格式
  - 按行存储数据，最后一列是疾病种类（1-6）
  - 特征：大多数为外观特征，以0（无），1，2，3（最显著）标记显著情况
    - 家族史（0 - 无， 1 - 有）
    - 年龄（整数）
    - 红斑
    - 多角形丘疹
    - 毛囊性丘疹
    - ……

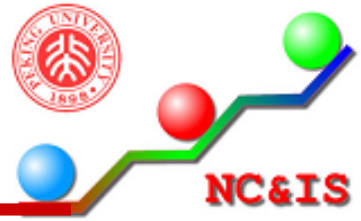


多分类问题 – 皮肤病诊断问题




```
#! /usr/bin/python
import numpy as np
import xgboost as xgb
# label need to be 0 to num_class -1
data = np.loadtxt(
    './dermatology.data',
    delimiter=',',
    converters={33: lambda x: int(x == '?'), 34: lambda x: int(x) - 1})
sz = data.shape
train = data[:int(sz[0] * 0.7), :]
test = data[int(sz[0] * 0.7):, :]
train_X = train[:, 0:33]
train_Y = train[:, 34]
test_X = test[:, 0:33]
test_Y = test[:, 34]
xg_train = xgb.DMatrix(train_X, label=train_Y)
xg_test = xgb.DMatrix(test_X, label=test_Y)
```

多分类问题－皮肤病诊断问题




```
# setup parameters for xgboost
param = {
    'objective': 'multi:softmax',
    'eta': 0.1,
    'max_depth': 6,
    'silent': 1,
    'nthread': 4,
    'num_class': 6}
watchlist = [(xg_train, 'train'), (xg_test, 'test')]
num_round = 5
bst = xgb.train(param, xg_train, num_round, watchlist)
# get prediction
pred = bst.predict(xg_test)
print ('predicting, classification error=%f' % (sum(int(pred[i]) != test_Y[i] for
i in range(len(test_Y))) / float(len(test_Y))))
```

多分类任务 – 互信息Loss

4线程训练

6种目标值

输出测试集的错误率

回归问题：CPU性能预测问题

- 问题背景
  - computer hardware dataset from UCI repository
  - 数据规模 – 训练集169，测试集40
  - 给出相关特征，预测CPU的性能分数(倒数第二列)
    - 制造商厂家
    - ~~CPU型号(未使用)~~
    - 主频周期(ns)
    - 最小支持内存(Kb)
    - 最大支持内存(Kb)
    - 缓存大小(Kb)
    - 最少通道数
    - 最大通道数
    - ~~用线性回归得到的预测值（未使用）~~

```
adviser,32/60,125,256,6000,256,16,128,198,199
amdahl,470v/7,29,8000,32000,32,8,32,269,253
amdahl,470v/7a,29,8000,32000,32,8,32,220,253
amdahl,470v/7b,29,8000,32000,32,8,32,172,253
amdahl,470v/7c,29,8000,16000,32,8,16,132,132
amdahl,470v/b,26,8000,32000,64,8,32,318,290
amdahl,580-5840,23,16000,32000,64,16,32,367,381
amdahl,580-5850,23,16000,32000,64,16,32,489,381
amdahl,580-5860,23,16000,64000,64,16,32,636,749
amdahl,580-5880,23,32000,64000,128,32,64,1144,1238
apollo,dn320,400,1000,3000,0,1,2,38,23
apollo,dn420,400,512,3500,4,1,6,40,24
basf,7/65,60,2000,8000,65,1,8,92,70
basf,7/68,50,4000,16000,65,1,8,138,117
```

回归问题：CPU性能预测问题

- 数据处理
  - 制造商厂家 – 离散型特征 – one-hot encoding
  - 其他特征 – 连续型特征 – 原封不动

```
adviser,32/60,125,256,6000,256,16,128,198,199
amdahl,470v/7,29,8000,32000,32,8,32,269,253
amdahl,470v/7a,29,8000,32000,32,8,32,220,253
amdahl,470v/7b,29,8000,32000,32,8,32,172,253
amdahl,470v/7c,29,8000,16000,32,8,16,132,132
amdahl,470v/b,26,8000,32000,64,8,32,318,290
amdahl,580-5840,23,16000,32000,64,16,32,367,381
amdahl,580-5850,23,16000,32000,64,16,32,489,381
amdahl,580-5860,23,16000,64000,64,16,32,636,749
amdahl,580-5880,23,32000,64000,128,32,64,1144,1238
apollo,dn320,400,1000,3000,0,1,2,38,23
apollo,dn420,400,512,3500,4,1,6,40,24
basf,7/65,60,2000,8000,65,1,8,92,70
```

```
198 0:125 1:256 2:6000 3:256 4:16 5:128 6:1
269 0:29 1:8000 2:32000 3:32 4:8 5:32 7:1
220 0:29 1:8000 2:32000 3:32 4:8 5:32 7:1
172 0:29 1:8000 2:32000 3:32 4:8 5:32 7:1
132 0:29 1:8000 2:16000 3:32 4:8 5:16 7:1
318 0:26 1:8000 2:32000 3:64 4:8 5:32 7:1
367 0:23 1:16000 2:32000 3:64 4:16 5:32 7:1
489 0:23 1:16000 2:32000 3:64 4:16 5:32 7:1
636 0:23 1:16000 2:64000 3:64 4:16 5:32 7:1
1144 0:23 1:32000 2:64000 3:128 4:32 5:64 7:1
38 0:400 1:1000 2:3000 3:0 4:1 5:2 8:1
40 0:400 1:512 2:3500 3:4 4:1 5:6 8:1
92 0:60 1:2000 2:8000 3:65 4:1 5:8 9:1
```

回归问题：CPU性能预测问题

XGBoost代码

```
import xgboost as xgb
dtrain = xgb.DMatrix('machine.txt.train')
dtest = xgb.DMatrix('machine.txt.test')
param = {'max_depth': 3,
         'eta': 1.0,
         'gamma': 1.0,
         'min_child_weight': 1,
         'save_period': 0,
         'booster': 'gbtree',
         'objective': 'reg:linear'}
num_round = 2
watchlist = [(dtest, 'eval'), (dtrain, 'train')]
bst = xgb.train(param, dtrain, num_round, watchlist)
preds = bst.predict(dtest)
write_pred('pred2.txt', preds)
bst.dump_model('dump2.nice.txt', 'featmap.txt')
```

回归任务 – 平方Loss

参考资料

- 来自项目主页/Github
  - 原理介绍 - Introduction to Boosted Trees
  - 安装教程 - XGBoost - Installation Guide
  - Demo页面 – Awesome XGBoost
  - 参数意义 - XGBoost Parameters
  - 输入数据格式 - Text Input Format of DMatrix
  - Python API - Python API Reference
- 其他的介绍XGBoost的资料
  - XGBoost - eXtreme Gradient Boosting – ppt
  - 原理介绍（中文版）
  - Complete Guide to Parameter Tuning in XGBoost – 参数调试教程