参考 XGBoost编译以及C接口调用 完成源码下载和编译，生成xgboost.sln工程文件。
之后我们走的路线跟上一篇不同，上一篇是利用C接口，这一篇是不用xgboost的任何接口，直接调试源码。

# 一、Visual Studio环境配置

## 1）调试用debug版本

需要修改的地方：
项目-->属性-->配置管理器-->活动解决方案配置-->新建-->Debug
项目-->属性-->配置管理器-->活动解决方案平台-->X64
项目 --> 属性 --> C/C++ --> 常规 --> 调试信息格式 --> 程序数据库 (/Zi) 。
项目 --> 属性 --> C/C++ --> 优化 --> 已禁用 (/Od)。

**注意：**是每个项目都要右键点击属性修改（如果有C/C++这个选项的话，因为有的项目点击属性后没有C++或者连接器，那就不用改了，具体原理我也没细究）。

## 2）找到码源的...\xgboost\demo\CLI\binary_classification目录

### 1.cmd下运行下面命令生成训练、测试数据

```
python mapfeat.py
python mknfold.py agaricus.txt 1
```

该文件夹下文件需要说明一下：
mushroom.conf是要作为cli_main.cc的主函数参数传入的，在cli_main.cc中会对其进行解析
agaricus.txt是原始数据
agaricus.txt.test是测试数据，同时也可以作为验证数据
agaricus.txt.train是训练数据
featmap.txt是特征映射编号

### 2.将agaricus.txt, agaricus.txt.test, agaricus.txt.train复制到.sln工程文件目录下。

### 3.打开vs2013工程右键runxgboost项目->属性->配置属性->调试->命令参数：添加mushroom.conf的绝对路径

## 3）程序中打断点，debug运行xgboost项目

至此就可以单步运行C++码源了。

## 二、可能遇到的错误：

### ❶fatal error C1083: 无法打开包括文件: “xgboost/data.h”。 这种错误就是找不到data.h文件，一般是inclue路径没设置对。

**解决方法：**（所有的项目凡是右键点击属性后有C/C++选项的都给它改过来）

项目 --> 属性 --> C/C++ --> 附加包含目录。

增加了三项（根据自己对应路径）：

D:\demo\xgboost\include

D:\demo\xgboost\rabit\include

D:\demo\xgboost\dmlc-core\include

### ❷LINK : fatal error LNK1561: 必须定义入口点。一个.exe文件没有main()函数，那么编译时，编译器就会报错。

**解决办法：**

项目 --> 属性 -->常规-->配置类型。

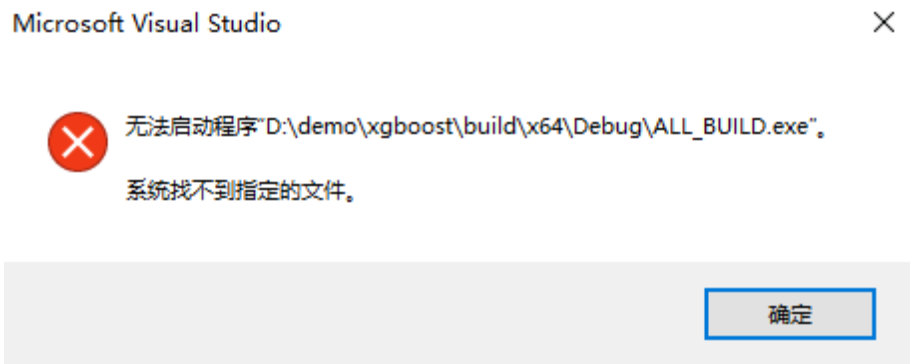
只有包含cli_main的runxgboost为.exe，其余都为.lib

### ❸fatal error LNK1104: 无法打开文件“..... \xgboost\build\objxgboost.dir\Debug\c_api.obj”

**解决方法：**（根据自己对应路径）：

将D:\demo\xgboost\build\src\x64下的Debug复制到D:\demo\xgboost\build\src\objxgboost.dir下

❹




**解决办法：**右键包含cli_main的runxgboost，将其设为启动项目。

## 三、调试结果：

cli_main的输入都在mushroom.conf文件下，用txt格式打开，可以修改相应的参数，例如num_round ,save_period等等。

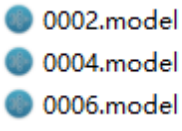
```
# the number of round to do boosting
num_round = 6
# 0 means do not save any model except the final round model
save_period = 2
```

还有一部分参数没有包含也可以自行添加进去，否则的话就参照cli_main的100行左右的声明参数中的默认参数解析了。

```
// declare parameters
DMLC_DECLARE_PARAMETER(CLIParam) {
  // NOTE: declare everything except eval_data_paths.
  DMLC_DECLARE_FIELD(task).set_default(kDumpModel)
      .add_enum("train", kTrain)
      .add_enum("dump", kDumpModel)
      .add_enum("pred", kPredict)
```

训练模式（kTrain）：

```
[13:53:17] [0]  test-logloss:0.162646   train-logloss:0.161646
[13:53:27] [1]  test-logloss:0.061633   train-logloss:0.064984
[13:53:28] [2]  test-logloss:0.024160   train-logloss:0.028140
[13:53:28] [3]  test-logloss:0.010586   train-logloss:0.013239
[13:53:28] [4]  test-logloss:0.007233   train-logloss:0.007400
[13:53:28] [5]  test-logloss:0.003974   train-logloss:0.004458
```



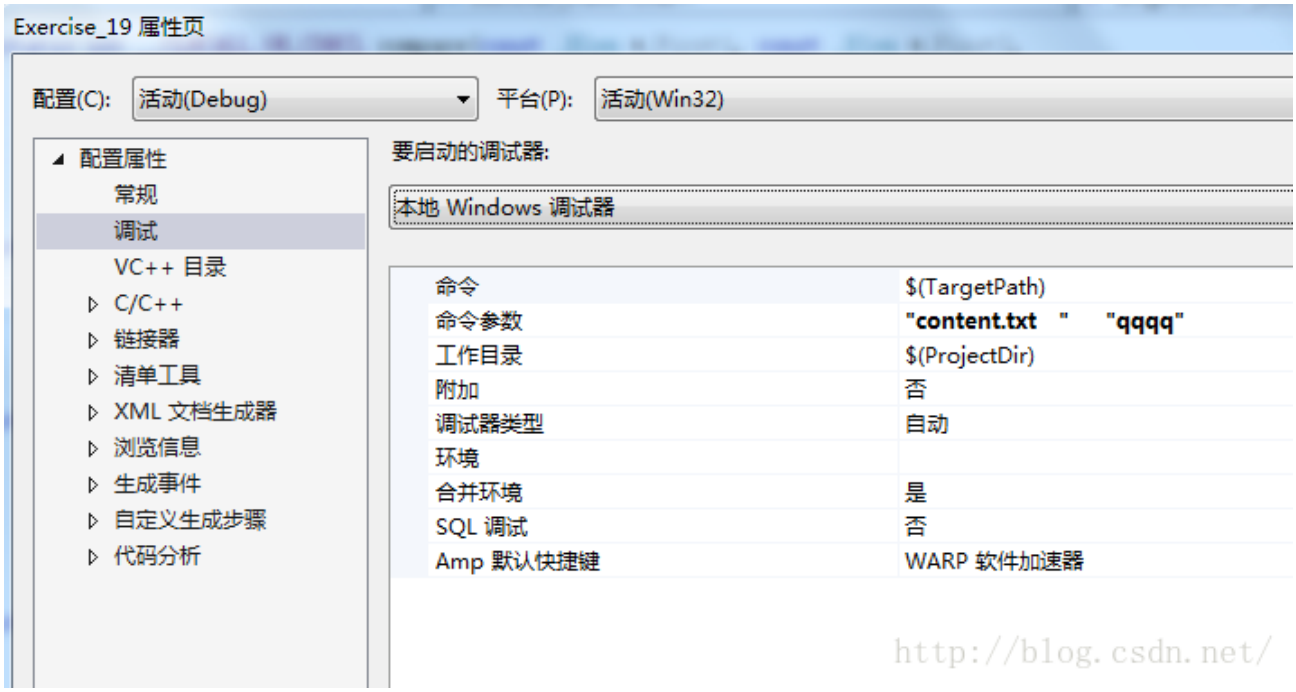

输出模型模式（kDumpModel）：输出TXT可视化模型

```
booster[0]:
0:[f28<-9.53674316e-07] yes=1,no=2,missing=1
        1:[f55<-9.53674316e-07] yes=3,no=4,missing=3
                3:[f59<-9.53674316e-07] yes=7,no=8,missing=7
                        7:leaf=1.89899647
                        8:leaf=-1.94736838
                4:[f20<-9.53674316e-07] yes=9,no=10,missing=9
                        9:leaf=1.78378379
                        10:leaf=-1.98135197
        2:[f108<-9.53674316e-07] yes=5,no=6,missing=5
                5:[f66<-9.53674316e-07] yes=11,no=12,missing=11
                        11:leaf=-1.9854598
                        12:leaf=0.938775539
                6:leaf=1.87096775
```

预测模式（kPredict）：即预测回归值。

## 四、另外的技巧

命令行参数添加：右键要添加命令行参数的工程->属性->配置属性->调试，右边有命令行参数输入框，输入即可。argv[0]是程序名，从argv[1]开始是输入的参数。个参数之间用空格进行分隔，当参数中含有空格时，要将参数用双引号括起来，否则空格不会被添加到参数中。




调试技巧:

1、启动调试。

可以通过VS的调试（Debug）菜单启动调试。点击调试菜单下的“启动调试”或者按F5键启动。如果你已经在代码中加入了断点，那么执行会自动开始。

注：退出调试快捷键shift+F5。

2、断点（Breakpoints）。

断点用于通知调试器何时何处暂停程序的执行。通过点击左边栏或者按F9键在当前行添加断点。在加断点之前，你需要知道你的代码将会出现什么错误，在什么地方停止执行。当调试器执行到断点处时，你可以使用其他的调试工具核对代码何处出现错误。

3、逐过程（Step Over）

快捷键F10，逐过程不会进入函数内部。

4、逐语句（Step Into）

快捷键是F11，会进入函数内部。

5、跳出（Step Out）

当你在一个方法内部调试时会用到它。如果你在当前方法内按Shift - F11，调试器会完成此方法的执行，之后在调用此方法的语句的下一条语句处暂停。

6、继续（Continue）它像是重新执行你的程序。它会继续程序的执行直到遇到下一个断点。快捷键是”F5“。