Dlib Library的编译、安装与使用

Dlib概述

Dlib是一个机器学习库，使用C++语言写成。在安装Dlib库之前，请首先安装Boost库。

安装过程

初始化环境

```
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libx11-dev libatlas-base-dev
sudo apt-get install libgtk-3-dev libboost-python-dev
sudo apt-get install python-dev python-pip python3-dev python3-pip
sudo -H pip2 install -U pip numpy
sudo -H pip3 install -U pip numpy
```

编译C++环境下的Dlib Library

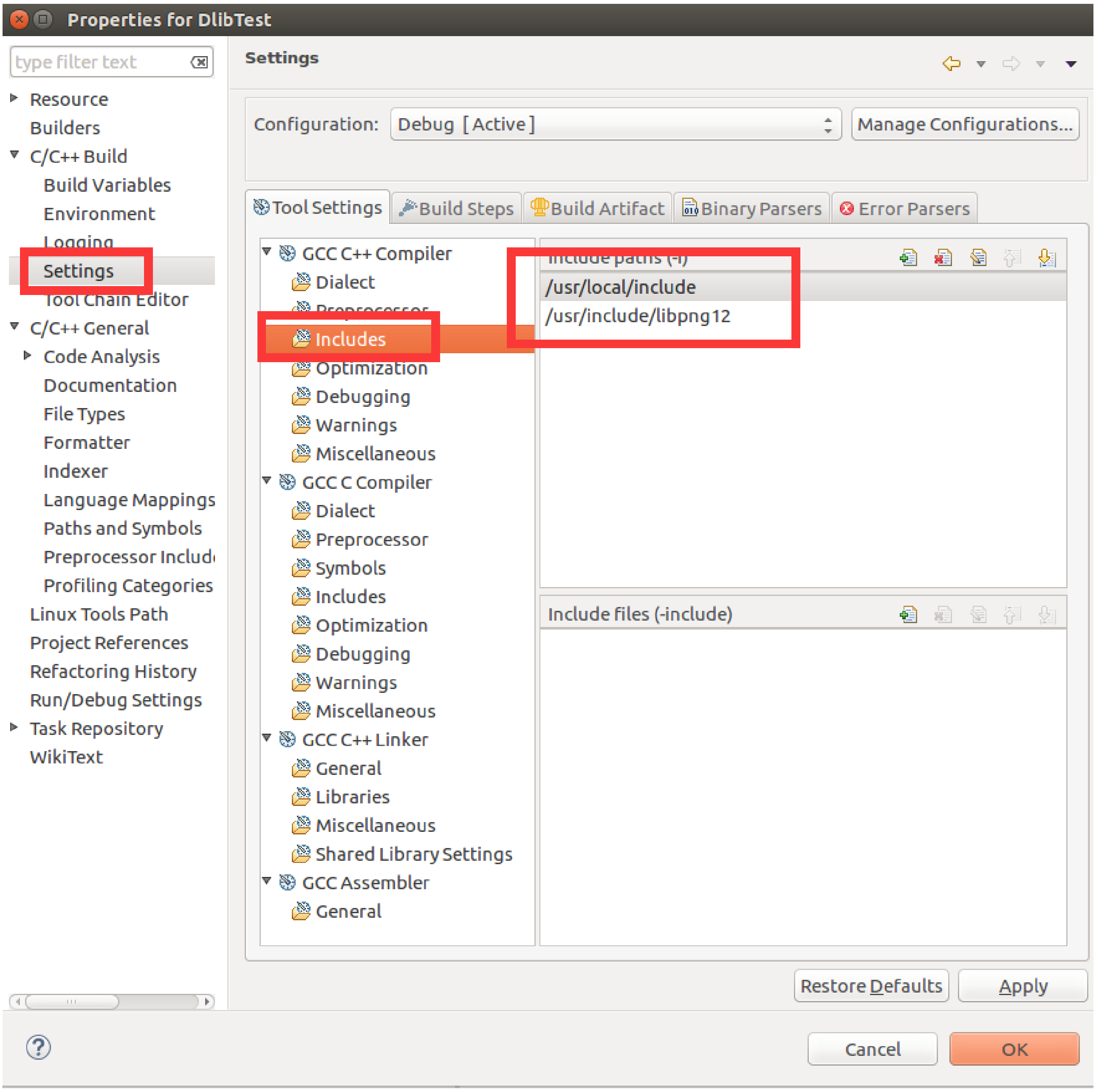
```
wget http://dlib.net/files/dlib-19.6.tar.bz2
tar xvf dlib-19.6.tar.bz2
cd dlib-19.6/
mkdir build
cd build
cmake ..
cmake --build . --config Release
sudo make install
sudo ldconfig
```

```
cd ..
pkg-config --libs --cflags dlib-1  #查看dlib-1需要的-I -L参数
```

编译代码语句：

```
g++ -std=c++11 svm_rank_ex.cpp -I/usr/local/include -I/usr/include/libpng12 -L/usr/local/lib -llapack -lblas -ldlib -lpng12
```

其中一定要注意，在编译时**-std=c++11**选项必须要设置，否则会出现编译错误



Properties for DlibTest

Settings

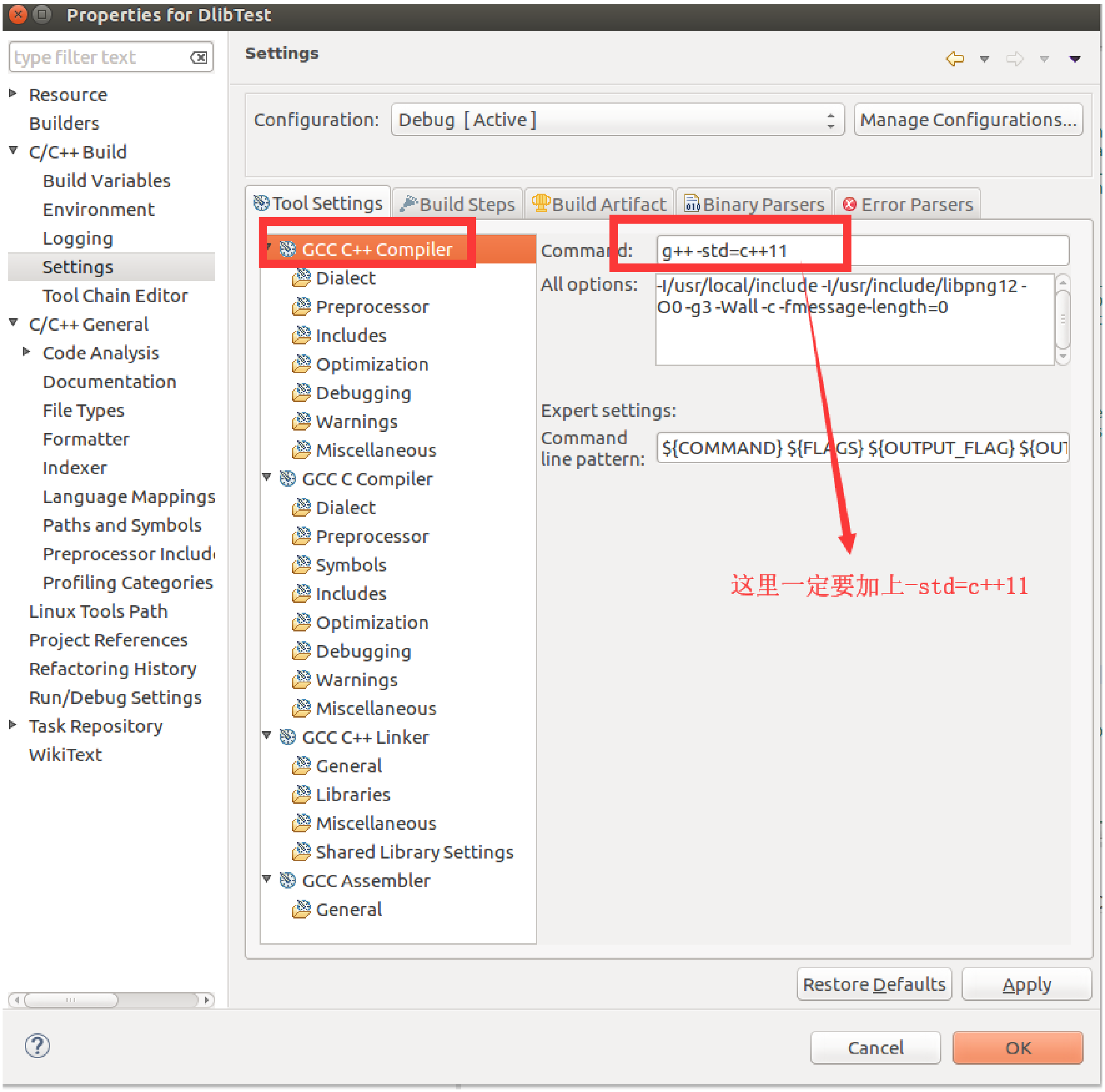
Configuration: Debug [Active]

Include paths (-I)

/usr/local/include

/usr/include/libpng12

Include files (-include)



Configuration: Debug [Active]

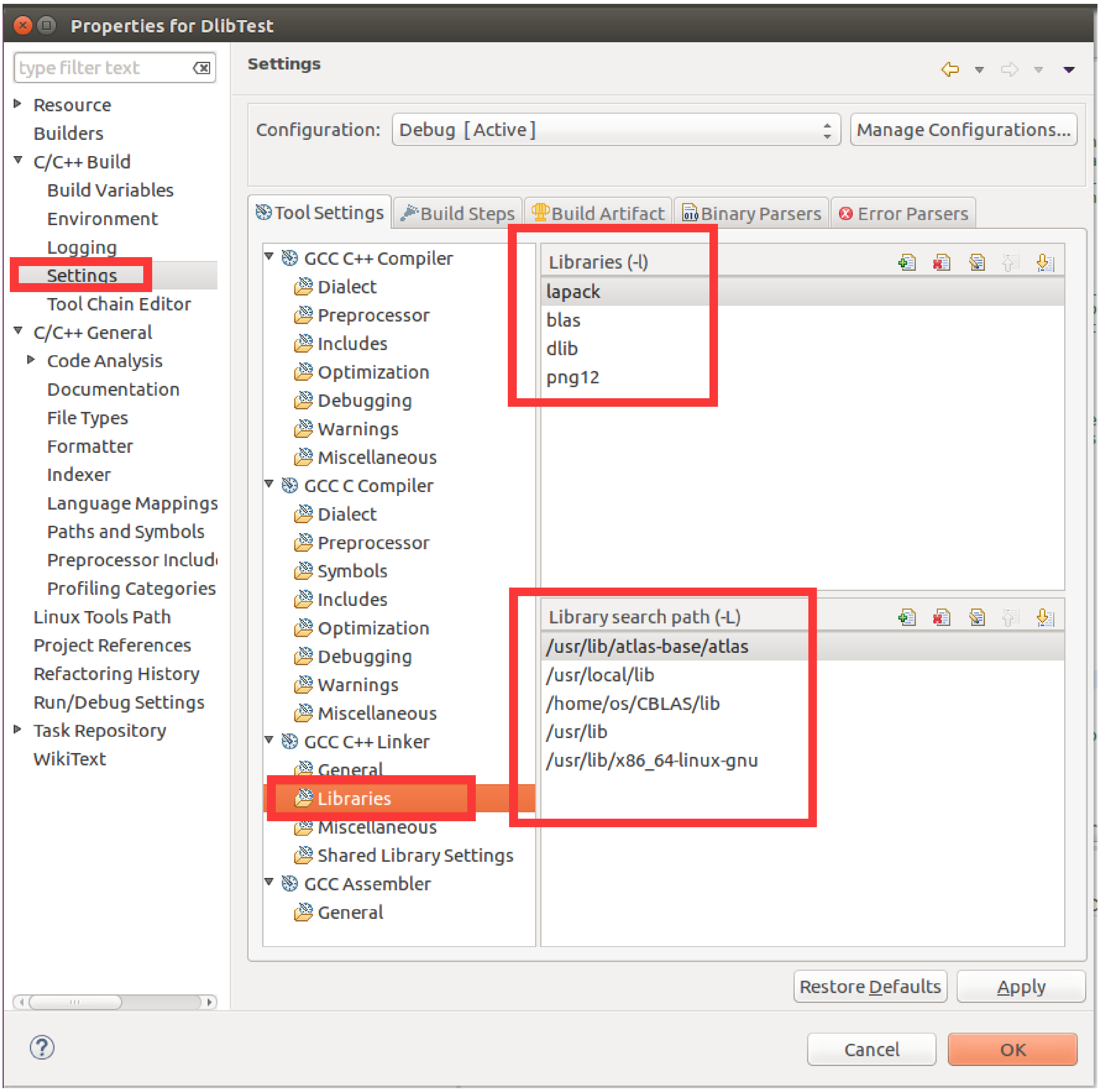
Command: g++ -std=c++11

All options: -I/usr/local/include -I/usr/include/libpng12 -O0 -g3 -Wall -c -fmessage-length=0

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUT

这里一定要加上-std=c++11



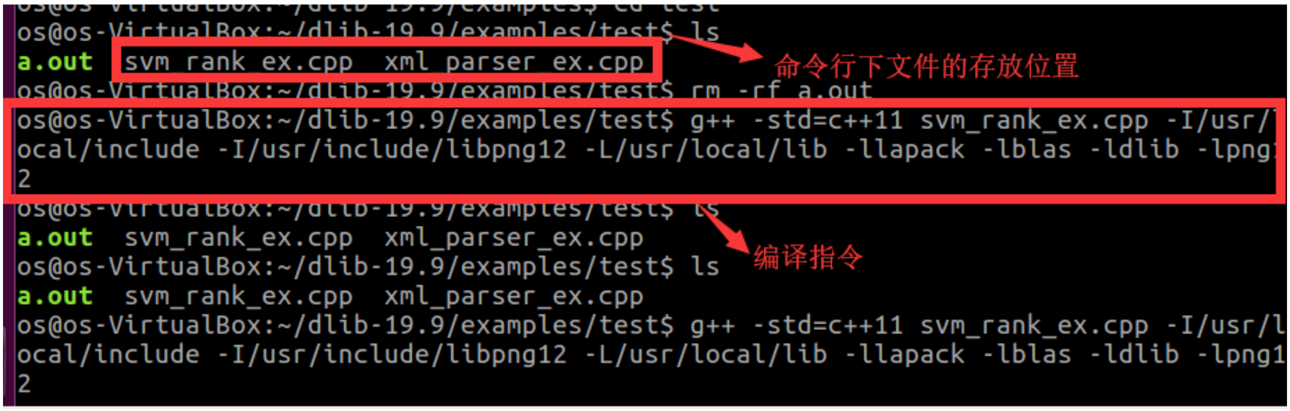
Configuration: Debug [Active]

Libraries (-l)

- lapack
- blas
- dlib
- png12

Library search path (-L)

- /usr/lib/atlas-base/atlas
- /usr/local/lib
- /home/os/CBLAS/lib
- /usr/lib
- /usr/lib/x86_64-linux-gnu



```
os@os-VirtualBox:~/dlib-19.9/examples$ cd test
os@os-VirtualBox:~/dlib-19.9/examples/test$ ls
a.out  svm_rank_ex.cpp  xml_parser_ex.cpp
os@os-VirtualBox:~/dlib-19.9/examples/test$ rm -rf a.out
os@os-VirtualBox:~/dlib-19.9/examples/test$ g++ -std=c++11 svm_rank_ex.cpp -I/usr/l
ocal/include -I/usr/include/libpng12 -L/usr/local/lib -llapack -lblas -ldlib -lpng1
2
os@os-VirtualBox:~/dlib-19.9/examples/test$ ls
a.out  svm_rank_ex.cpp  xml_parser_ex.cpp
os@os-VirtualBox:~/dlib-19.9/examples/test$ ls
a.out  svm_rank_ex.cpp  xml_parser_ex.cpp
os@os-VirtualBox:~/dlib-19.9/examples/test$ g++ -std=c++11 svm_rank_ex.cpp -I/usr/l
ocal/include -I/usr/include/libpng12 -L/usr/local/lib -llapack -lblas -ldlib -lpng1
2
```

命令行下文件的存放位置

编译指令

编译Python模块

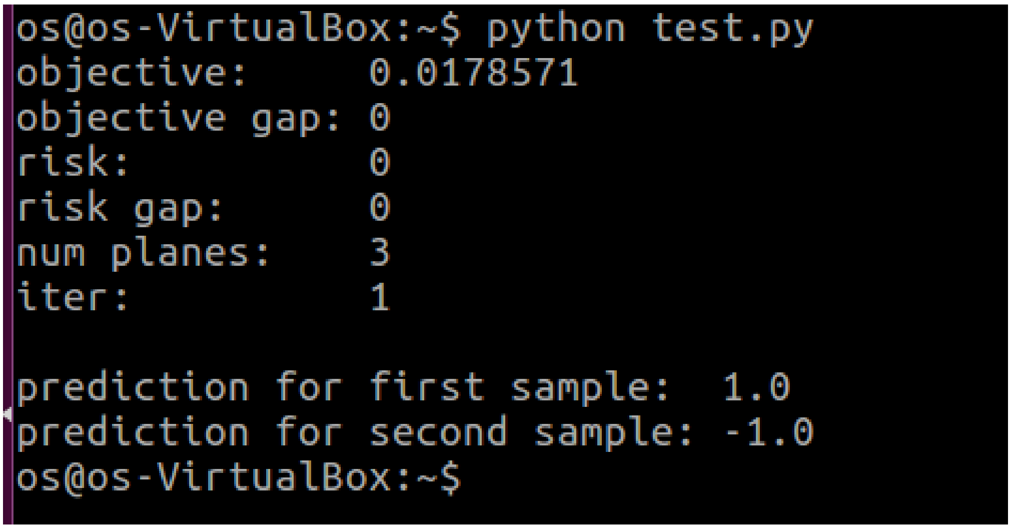
```
cd dlib-19.6/
python setup.py install
rm -rf dist
rm -rf tools/python/build
rm python_examples/dlib.so
```

测试上述python环境是否可用，测试代码：

```
# The contents of this file are in the public domain. See LICENSE_FOR_EXAMPLE_PROGRAMS.txt
#
#
# This is an example illustrating the use of a binary SVM classifier tool from
# the dlib C++ Library.  In this example, we will create a simple test dataset
# and show how to learn a classifier from it.
#
#
# COMPILING/INSTALLING THE DLIB PYTHON INTERFACE
#   You can install dlib using the command:
#       pip install dlib
#
#   Alternatively, if you want to compile dlib yourself then go into the dlib
#   root folder and run:
#       python setup.py install
#
#   Compiling dlib should work on any operating system so long as you have
#   CMake installed.  On Ubuntu, this can be done easily by running the
#   command:
#       sudo apt-get install cmake
#

import dlib
try:
    import cPickle as pickle
except ImportError:
    import pickle

x = dlib.vectors()
y = dlib.array()
```

```python

# Make a training dataset.  Here we have just two training examples.  Normally
# you would use a much larger training dataset, but for the purpose of example
# this is plenty.  For binary classification, the y labels should all be either +1 or -1.
x.append(dlib.vector([1, 2, 3, -1, -2, -3]))
y.append(+1)

x.append(dlib.vector([-1, -2, -3, 1, 2, 3]))
y.append(-1)

# Now make a training object.  This object is responsible for turning a
# training dataset into a prediction model.  This one here is a SVM trainer
# that uses a linear kernel.  If you wanted to use a RBF kernel or histogram
# intersection kernel you could change it to one of these lines:
#  svm = dlib.svm_c_trainer_histogram_intersection()
#  svm = dlib.svm_c_trainer_radial_basis()
svm = dlib.svm_c_trainer_linear()
svm.be_verbose()
svm.set_c(10)

# Now train the model.  The return value is the trained model capable of making predictions.
classifier = svm.train(x, y)

# Now run the model on our data and look at the results.
print("prediction for first sample:  {}".format(classifier(x[0])))
print("prediction for second sample: {}".format(classifier(x[1])))

# classifier models can also be pickled in the same was as any other python object.
with open('saved_model.pickle', 'wb') as handle:
    pickle.dump(classifier, handle, 2)
```

将上述代码保存为test.py，在命令行下输入pyhton test.py，如果出现以下结果说明安装成功：



```
os@os-VirtualBox:~$ python test.py
objective:      0.0178571
objective gap: 0
risk:           0
risk gap:       0
num planes:     3
iter:           1

prediction for first sample:  1.0
prediction for second sample: -1.0
os@os-VirtualBox:~$
```