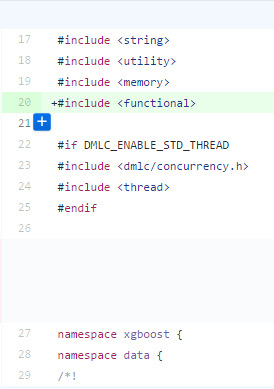
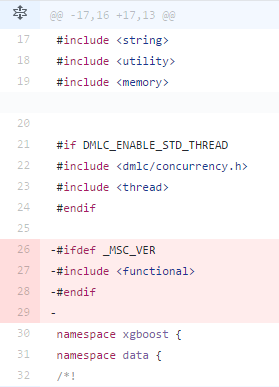
Instructions for setting up XGBoost for python on Windows 7 (should work on later versions as well, but not tested)

As of: 5/12/2017

Primarily taken from [here](https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing_XGBoost_For_Anaconda_on_Windows?lang=en), with the following modifications:

1. Keep track of your MINGW version and modify the path references to it accordingly
2. After cloning and updating submodules of the git repository, modify these header files (same changes to both files):
   1. src/data/sparse\_batch\_page.h
   2. include/xgboost/tree\_updater.h



#### Installing XGBoost For Anaconda on Windows

[*JeanFrancoisPuget*](javascript:;)*| Mar 29 2016 | Visits (65891)*

[XGBoost](http://dmlc.cs.washington.edu/xgboost.html) is a recent implementation of Boosted Trees.  It is a machine learning algorithm that yields great results on recent [Kaggle competitions](https://github.com/dmlc/xgboost/blob/master/demo/README.md" \l "machine-learning-challenge-winning-solutions).  I decided to install it on my computers to give it a try.  Installation on OSX was straightforward using these [instructions](https://xgboost.readthedocs.org/en/latest/build.html#building-on-osx) (as a matter of fact, reality is a bit more complex, see the update at the bottom of this post).  Installation on Windows was not as straightforward.  I am sharing what worked for me in case it might help others.  I describe how to install for the [Anaconda](https://www.google.fr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiz5paA2eXLAhWEvBoKHfeBApYQFggdMAA&url=https%3A%2F%2Fwww.continuum.io%2Fdownloads&usg=AFQjCNH5KKA7CTASoQKpNBeQAV2xSKKTrQ&sig2=9DpmLfw2JBfFYb4iJbhTBQ) Python distribution, but it might work as-is for other Python distributions.

In order to install and use XGBoost with Python you need three software on your windows machine:

* A Python installation such as [Anaconda.](https://www.google.fr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiz5paA2eXLAhWEvBoKHfeBApYQFggdMAA&url=https%3A%2F%2Fwww.continuum.io%2Fdownloads&usg=AFQjCNH5KKA7CTASoQKpNBeQAV2xSKKTrQ&sig2=9DpmLfw2JBfFYb4iJbhTBQ)
* Git
* MINGW

I assume you have Anaconda up and running.  I am using Anaconda for Python 3.5.

Git installation is quite easy.  There are several options, one is to use [Git for Windows](https://git-for-windows.github.io/).  Just download and save the installer file on your disk, then launch it by double clicking it.  You may need to authorize this operation.  Then follow the installer instructions.

Once the installation has completed look for a program called Git Bash in your start menu.  Launch it.  It starts a terminal running the Bash shell.  This is different from the regular Windows terminal, but it is more handy for what we need to do.  First, go to the directory where you want to save XGBoost code by typing the cd command in the bash terminal.  I used the following.

 $ cd /c/Users/IBM\_ADMIN/code/

Then download XGBoost by typing the following commands.

$ git clone --recursive <https://github.com/dmlc/xgboost>

$ cd xgboost

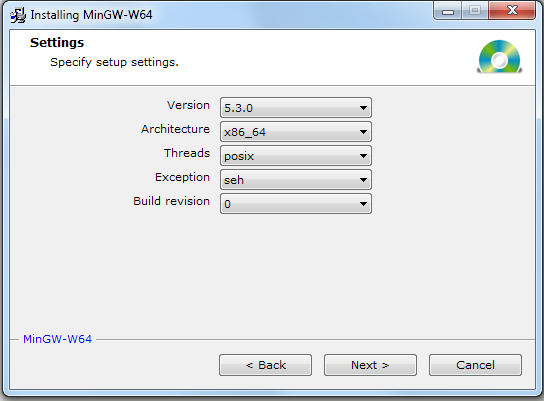
$ git submodule init

$ git submodule update

Next step is to build XGBoost on your machine, i.e. compile the code we just downloaded.  For this we need a full fledged 64 bits compiler provided with MinGW-W64.  I downloaded the installer from this [link](http://iweb.dl.sourceforge.net/project/mingw-w64/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe).  Save the file on your disk, then launch it by double clicking it.  You may need to authorize this operation.  Then click next on the first screen:

[](https://www.ibm.com/developerworks/community/blogs/jfp/resource/BLOGS_UPLOADED_IMAGES/mingw64-1.png)

Then select the x86\_64 item in the architecture menu.  Do not modify the other settings.

[](https://www.ibm.com/developerworks/community/blogs/jfp/resource/BLOGS_UPLOADED_IMAGES/mingw64-2.png)

Then click Next and follow the instructions.  On my machine, it installed the compiler in the C:\Program Files\mingw-w64\x86\_64-5.3.0-posix-seh-rt\_v4-rev0 directory.  The make command and the runtime libraries are in this directory (look for the directory that contains mingw32-make):

C:\Program Files\mingw-w64\x86\_64-5.3.0-posix-seh-rt\_v4-rev0\mingw64\bin

Use [these instructions](http://www.computerhope.com/issues/ch000549.htm), depending on your Windows version, to add the above to the Path system variable.

Then close the Git Bash terminal, and launch it again.  This will take into account the new Path variable.  To check you are fine, type the following

$ which mingw32-make

It should return something like:

/c/Program Files/mingw-w64/x86\_64-5.3.0-posix-seh-rt\_v4-rev0/mingw64/bin/mingw32-make

To make our life easier, let us alias it as follows:

$ alias make='mingw32-make'

We can now build XGBoost.  We first go back to the directory where we downloaded it:

 $ cd /c/Users/IBM\_ADMIN/code/xgboost

The command given in the [instructions](https://xgboost.readthedocs.org/en/latest/build.html#building-on-windows) does not work as I write this blog entry.  Until this is fixed, we need to compile each sub module explicitly with the following commands.  Wait until each make command is completed before typing the next command.

$ cd dmlc-core

$ make -j4

$ cd ../rabit

$ make lib/librabit\_empty.a -j4

$ cd ..

$ cp make/mingw64.mk config.mk

$ make -j4

Once the last command completes the build is done.

We can now install the Python module.  What follows depends on the Python distribution you are using.  For Anaconda, I will simply use the Anaconda prompt, and type the following in it (after the prompt, in my case [Anaconda3] C:\Users\IBM\_ADMIN>):

[Anaconda3] C:\Users\IBM\_ADMIN>cd code\xgboost\python-package

The point is to move to the python-package directory of XGBoost.  Then type:

[Anaconda3] C:\Users\IBM\_ADMIN\code\xgboost\python-package>python setup.py install

We are almost done.  Let's launch a notebook to test XGBoost.  Importing it directly causes an error.  In order to avoid it we must add the path to the g++ runtime libraries to the os environment path variable with:

**import** **os**

mingw\_path = 'C:**\\**Program Files**\\**mingw-w64**\\**x86\_64-5.3.0-posix-seh-rt\_v4-rev0**\\**mingw64**\\**bin'

os.environ['PATH'] = mingw\_path + ';' + os.environ['PATH']

We can then import xgboost and run a small example.

**import** **xgboost** **as** **xgb**

**import** **numpy** **as** **np**

data = np.random.rand(5,10) *# 5 entities, each contains 10 features*

label = np.random.randint(2, size=5) *# binary target*

dtrain = xgb.DMatrix( data, label=label)

dtest = dtrain

param = {'bst:max\_depth':2, 'bst:eta':1, 'silent':1, 'objective':'binary:logistic' }

param['nthread'] = 4

param['eval\_metric'] = 'auc'

evallist = [(dtest,'eval'), (dtrain,'train')]

num\_round = 10

bst = xgb.train( param, dtrain, num\_round, evallist )

bst.dump\_model('dump.raw.txt')

We are all set!