1. Project description, instructions to compile and run.

1.1 Project description

This project uses the OpenCV-4.3.0 library, specifically the Support Vector Machine method to recognize hand-written digits, which is called Mnist dataset, and it's from http://yann.lecun.com/exdb/mnist/. The reason to choose such project is that, in addition to practice C++ language, I would like to take this chance to gain some initial experience with the widely used OpenCV library, and some machine learning methods, which is my future career interest.

1.2 Usage of OpenCV library

I first successfully tested on the Windows Visual Studio 2017 environment, for which you can simply download and use the pre-built OpenCV library from https://sourceforge.net/projects/opencvlibrary/files/4.3.0/.

In the Linux environment of Udacity workspace, since there is already an OpenCV installed under the Non-GPU mode in the directory /opt/conda, with header files in directory "include" and library files in directory "lib", I simply use it.

1.3 Compile and run on Udacity workspace Non-GPU mode

My project is pretty simple, a "data" directory contains the Mnist data, and a "src" directory contains 3 source files. In order to compile, in my case, I first set the library paths in .student_bashrc file for both compile time and run time:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/conda/lib
export LIBRARY_PATH=$LIBRARY_PATH:/opt/conda/lib
```

and then activate it by command:

```
# source .student_bashrc
```

When you encounter compile errors like, "undefined reference to `dpotrf_'", try to run the above "source" command. It works for me.

Then in directory "src", you compile the program with following g++ command:

```
# g++ mnistdb.cpp main.cpp -o a.exe -I/opt/conda/include -lopencv_core -lopencv_ml
```

And then run by following command:

```
# ./a.exe
```

1.4 Expected output

The program will output some basic information of the Mnist dataset, like the numbers of training example (60000), the test example (10000), and the digit image size (28x28). After training and predicting, it will output the result of 10 test examples, and the overall correct ratio of the prediction, and the trained SVM model result to an xml file, which can be loaded to OpenCV later.

1.5 File and class structure

It only contains 3 source files, main.cpp, mnistdb.h and mnistdb.cpp, and there is only 1 class called Mnistdb which is defined in the latter two files. This class has the data members to hold the Mnist dataset, and the member functions to operate on them, e.g. get/set/read data from files, and most importantly an SVM classifier function to do the digit recognition.

2. Rubric points addressed in this project

In addition to the REQUIRED rubric points for “README” and “Compiling and Testing”, the project also addresses the following points.

2.1 The project demonstrates an understanding of C++ functions and control structures.
In file “mnistdb.cpp” there are several functions, and they use “if” and “for” statements, particularly in function “void Mnistdb::SVM_Classifier()”, at lines 162 ~ 170, it uses both “for” and “if” statements.

2.2 The project reads data from a file and process the data, or the program writes data to a file.
In file “mnistdb.cpp” there are four functions that read mnist dataset, e.g. the function “int Mnistdb::ReadTrainImg(const string &fn)” reads the training dataset from file “train-images.idx3-ubyte” using ifstream object. In function “void Mnistdb::SVM_Classifier()” it writes out the trained SVM model to file “SVM_MNIST.xml” using OpenCV’s “save()” method.

2.3 The project uses Object Oriented Programming techniques.
In files mnistdb.h and mnistdb.cpp, it implemented a class called Mnistdb, which holds the relevant data and functions to operate on the data.

2.4 Classes use appropriate access specifiers for class members.
In file mnistdb.h, all of the Mnistdb class data members are explicitly specified as either public or private.

2.5 Classes abstract implementation details from their interfaces.
In file mnistdb.h, all of the member functions indicate their effects through function names.

2.6 Classes encapsulate behavior
In file mnistdb.h, relevant data and functions are grouped into one class of Mnistdb, all of the data members are set to be private, and they are hidden from the user, and are accessed via member functions.

2.7 The project makes use of references in function declarations.
In the Mnistdb class definition from file mnistdb.h, lines from 27 to 44, most of the functions use references either as function parameters (e.g. int ReadTrainImg(const string &fn);) or return types (e.g. Mat &TrainImg() { return m_TrainImg; }) .

2.8 The project uses smart pointers instead of raw pointers.
In the function definition of void Mnistdb::SVM_Classifier(), from file mnistdb.cpp, at lines 141-142,
Ptr<SVM> svm;
svm = SVM::create();
The definition of Ptr is an OpenCV template class for smart pointers with shared ownership, so it’s actually an implementation of shared_ptr. See reference at
https://docs.opencv.org/3.4/d0/de7/structcv_1_1Ptr.html