École Nationale de la Statistique et de l'Administration Économique

# Implementation of Neural Network using C++

Nicolas BETIN Julien CHANCEREUL Yannick DERIPPE

2013-2014

Projet de C++

## WARNING

We were not able to link the two parts of our project. The goal was to call the VS project from the Qt interface and we faced a ultimate bug with startDetached in Qt. It seems to us that the problem comes from the fact that out VS solution WinesClassification.exe couldn't write on the disk. We debugged the code and the trained neural network model couldn't be written on disk. Nevertheless, our programs compile and work and we joined them separately in the "Compiled solution" folder.

## 1 Introduction

The aim of this project is to predict the quality of different types of wine. The artificial neural network is a machine learning technique that uses explicative (input) variables to predict the value of another (output) variable via series of neurons. The learning principle is the minimization of the model's error, through its retropropagation from the output layer to the input layer.

The program has to be trained on a dataset. The values of the explicative variables and that of the predicted variable are known in the training dataset. After the training, the neural network can predict the value of the output variable when the values of the input variables of an observation are given.

## 2 Dataset and library

Our dataset is extracted from the UCI Machine Learning Repository website and contains 4898 observations of white wines.

There are 11 numerical input variables : fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates and alcohol. The output variable is a score between 0 and 10 that represents the quality of the wine. Our major goal was to successfully build a usable program that could come in useful to any kind of user. In order to achieve that, we used an already existing machine learning library called OpenCV that manages Neural Network implementation.

The programming of the neural network in C++ required 3 major steps.

## 3 Importation of data

The first one consisted in importing the data using C++. We first imported the .csv file, that contains the 4898 observations, via C++ and transformed it into a matrix of dimension (12,4898). We normalized each of the 11 first columns, the columns of the input variables, for the data to be contained between 0 and 1. The normalization function used is defined by : $x = (x - x_{min})/(x_{max} - x_{min})$.

We then exported the transformed matrix into 3 separate .csv files : one aims at training the neural network, and contains the majority of the data, the second one is used to test the accuracy of the neural network by computing the percentage of good predictions, and the last one is reserved for pure prediction, so that other people can try and use the program. In order to have unbiased results, we randomly shuffled the lines of the matrix before the separation. The first two matrices are then each separated into two, one containing the input variables, the other containing the output variable, so that they can be used by OpenCV.

# 4 Construction of the model

The second step is the construction of the model using the OpenCV library. We arbitrarily chose 3 layers of neurons : the first one contains one neuron for each input variable, the hidden one contains 9 neurons, and the last one contains one neuron for each output modality. The activation function chosen, the function used to transform the data that goes through a neuron, is the sigmoid function. Its expression is the following : $S(t) = 1/(1 + exp(-t))$ . We then wrote a method that creates the trained model, a method that predicts the output variables for non-training data, and another one that computes the accuracy of the model.

# 5 User interface

The final step of our work is the programming of a graphical user interface, using the application framework Qt. The graphical interface calls the C++ program in order to :

1. Train the neural network model
2. Store the model in an xml file
3. Use this model to predict the class label of user's data

Our goal was to allow the user to manually enter the values of each of the 11 input variables, and then the interface would return the predicted quality computed by the neural network. Nevertheless, we were not able to achieve that and the user actually has to enter his data diretly in the command prompt. We also included a button that returns an error message when one of the values of the input variables is not included between the minimum and maximum values of the training dataset.

# 6 OpenCV Integration

In order to implement a neural network in C++, we used the OpenCV library. This library is useful to handle images and to do machine learning with C++.

## 6.1 Installation

1. Download and install the basic OpenCV 2.4.8 package on the Opencv website
2. To integrate the library in Visual Studio 2013, we modified the properties of our VS Project in the properties manager
3. You can find our configuration files in the "Utilities" folder in our github project

**NOTA :** We used 64-bit computers to run our programs and we installed the 64-bit version of OpenCV. The installation process take time and we have encounter a lot of problems during the VS implementation.

### 6.2 Use

We used the opencv_core248.dll and opencvml_248.dll during the project. The two major functions of interest for us were the data importation and the multi layers perceptrons model. You can find these in the DataHandler and NNModel classes.

## 7 How to compile the code

You need to install the OpenCV library to run the purely C++ code and Qt Creator 5.2.1 to run the graphical interface code.

## 8 How to get the .exe files

The VS project returns us .exe files (Debug and Release). In order to run these without VS 2013, you have to :

1. Put your training data in the ../../folder
2. Create a Results/trained_models in the ../../folder
3. Copy the opencv_core248(d).dll and opencv_ml248(d).dll in your Release (Debug) folder

That done, we were able to launch the .exe files without VS and the neural network was successfully created.

## 9 Problems

During the project, we encountered several problems :

- Use the different **vector**, **std** and **cv : :Mat** classes to handle arrays
- Correctly handle the data
- Compatibility problems between our computers and software
- Make the juncture between Visual Studio and Qt Creator

We couldn't create a single program using Qt because the VS integration seemed very bad and the final program calls WinesClassification.exe from the graphical interface. The user can then enter his data in the command prompt and control the likelihood of his data using the graphical interface.