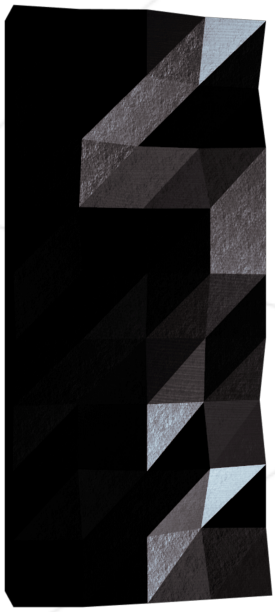


ft_sommelier

Elementary Machine Learning Project

Nicholas Mei nmei@student.42.us.org
42 Staff staff@42.fr

Summary: This project is the gateway to the machine learning branch. It will also help you become a master sommelier.

Contents

Chapter I

Foreword

Vinho Verde is a Portuguese wine that originated in the historic Minho province in the far north of the country. The modern-day 'Vinho Verde' region, originally designated in 1908, includes the old Minho province plus adjacent areas to the south. In 1976, the old province was dissolved.

Vinho Verde is not a grape variety, it is a DOC (denominação de origem controlada) for the production of wine. The name literally means "green wine," but translates as "young wine". They may be red, white, or rosé and they are usually consumed soon after bottling.

A Vinho Verde can be a sparkling, a Late Harvest, or even Brandy. In its early years of production, the slight effervesce of the wine came from malolactic fermentation taking place in the bottle. In winemaking this is usually considered a wine fault but Vinho Verde producers found that consumers liked the slightly fizzy nature. However, the wines had to be packaged in opaque bottles in order to hide the unseemly turbidity and sediment that the "in-bottle MLF" produced. Today, most Vinho Verde producers no longer follow this practice with the slight sparkle being added by artificial carbonation.

Knowing about Vinho Verde will make this project much easier.

Chapter II

Introduction

After attending a recent town hall at `42 Silicon Valley`, you came to the disturbing realization that you never learned how to properly handle alcoholic beverages. Ashamed at your inability to even distinguish between high and low quality wines, you've decided to learn how to properly appraise and handle your drinks (assuming you're over the legal age limit of course!).

Not even knowing where to begin, you decide to enlist the help of Marvin to tackle this difficult problem! With some prodding, Marvin grudgingly provides a detailed chemical analysis and quality score for a number of wines. However, he remarks, "It's pointless, even if you used all the neurons in that tiny brain of yours. But you won't listen will you? No one ever does..."

Wanting to prove Marvin wrong, you decide to show him that you can learn how to distinguish high quality wines from bad ones with just one neuron! Try as you might though, you can't seem to focus the attention of one of your neurons on the task. You recently just started studying machine learning though and vaguely recall that a perceptron might be a reasonable substitute for your clearly non-cooperative neurons!

Your goal for this project is to use perceptrons to distinguish between high and low quality wines and prove Marvin wrong!

Chapter III

Objectives

Marvin has grudgingly provided a detailed chemical analysis on a number of red and white wines (`winequality-red.csv` and `winequality-white.csv`). At the top of each dataset, he's listed the parameters he's analyzed. These parameters include:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol

Marvin has also included a "Quality" parameter which is his simulation of the score that three trained wine tasters would give the wine. Thus, each row in the dataset describes the chemical characteristics and quality score for a particular wine. Each parameter is separated by a semicolon (;).

Overwhelmed by the sheer amount of data **you decide to focus on red wines only.** Your overall goal? **Given only chemical attributes of a red wine, be able to classify that wine as a 'good wine' or a 'bad wine'.**

Chapter IV

General instructions

- You get to use Python for this project!
- An optional dockerfile has also been provided to help make the python environment setup process less painful.
- You are allowed to import and use the matplotlib, pandas, and the standard python libraries for *this* project.
    - You are **NOT** allowed to use pandas DataFrame math/matrix methods (i.e. '.T', '.transpose', '.dot', '.add', '.divide', '.mean', '.std', etc...)
    - It's nice to know they exist though isn't it?
- You are **NOT** allowed to directly import and use libraries like: numpy, scipy, scikit-learn, tensorflow, etc...
- You can download the red and white wine-quality datasets from the intra resources. You can also find this dataset online: here
- Data on Pan-Galactic Gargle Blaster can only be found at 42 though :)

```
Don't know anything about matplotlib?  learn more
Don't know anything about pandas?  learn more
Want to learn more about pandas?  learn more
Really need to learn about pandas?  learn more
```

Chapter V

Mandatory part

For this project, you will be writing a series of simple machine learning algorithms based on single biological neurons. Along the way, you'll also be writing some useful data exploration plotting functions and dabble a bit in validation/cross-validation techniques.

- All your work must be turned in as an IPython/Jupyter notebook (*.ipynb).
- For each section (V.1, V.2, etc...) you must create a "Markdown" header cell in your notebook.
- Exercises you will need to address will be in the form of alphabetical list elements: "a), b), c), d), etc..."
- Exercises with open ended questions should be answered in a "Markdown" cell.

Example of section Markdown header:

V.2: Learning to perceptron

a) & b) Perceptron implementation in 'C'ython

In [14]:
```
%%cython
from libc.stdlib cimport malloc, free, rand, srand, RAND_MAX
from cpython cimport array
import array
```

V.1 Exploring the green reds

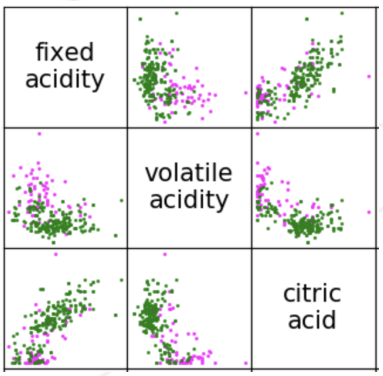
As a beginner in machine learning, you recall that one of the best things to do before breaking out the machine learning tools is to explore and look at your data.

Technically the best thing to do first would be to clean the data and check for anomalies, errors, and outliers! But since, Marvin gave us this data, we'll trust him. This time...

a) Write a **function** that will **plot** a **scatterplot matrix** of your red wine data. Your function should plot wines with quality over some "good_threshold" as one color, and wines below some "bad_threshold" as another. Your plotting function should have the option to save a .png of your plots. Here's an example function definition:

```
def plot_scatter_matrix(wine_data, good_threshold, bad_threshold, save_plot=False):
```

Here is a partial example (first 3 rows and columns) of what your function might output if you were to set a "good_threshold" score of 6 and a "bad_threshold" score of 5:



You don't have to use the colors shown in the examples, they're just optimized so colorblind folks can distinguish them too. You can check if your palettes are colorblind safe too: here.

Don't use the scatter_matrix() method in pandas.plotting! Learning how to modify plots with just matplotlib will be extremely helpful when creating custom plots in the future.

b) Now that you've had a chance to analyze different chemical factors in red wines and their relationship with high scores (8 or higher) and low scores (3 or lower), which factors do you think will be most useful to your perceptron for distinguishing high quality vs. low quality wines? Why?

V.2 Learning to perceptron

Now that you've explored the red wines, it's time to put together a perceptron that can learn to classify the wines as "good" or "bad" based on only chemical attributes!

a) Implement a perceptron that:

- Has randomly initialized weights and bias
- Uses the Rosenblatt perceptron learning rule (with changeable learning rate)
- Utilizes the heaviside step activation function (discrete version)

`Hopefully you've realized that your particular perceptron will only return '0' or '1' values. You'll need to either modify or add a column to your red wine data. Try to do this programmatically without modifying the .csv file!!`

b) You'll need to implement a function to train your perceptron. Your training function should take in your red wine data as a parameter and should:

- Have a way to specify number of training epochs
- Train your perceptron until it makes no errors, if training epochs is set to 0,
- Have a way to specify learning rate.
- Return a list of python tuples containing (performance):

[(current_epoch, num_errors_at_epoch_end, [array_of_weights], bias), ...]

What should your 'training' data be? For now, focus on training your perceptron on 2 chemical factors (for example, alcohol and pH) and only use wines with a score of 8 or higher and wines with a score of 3 or lower.

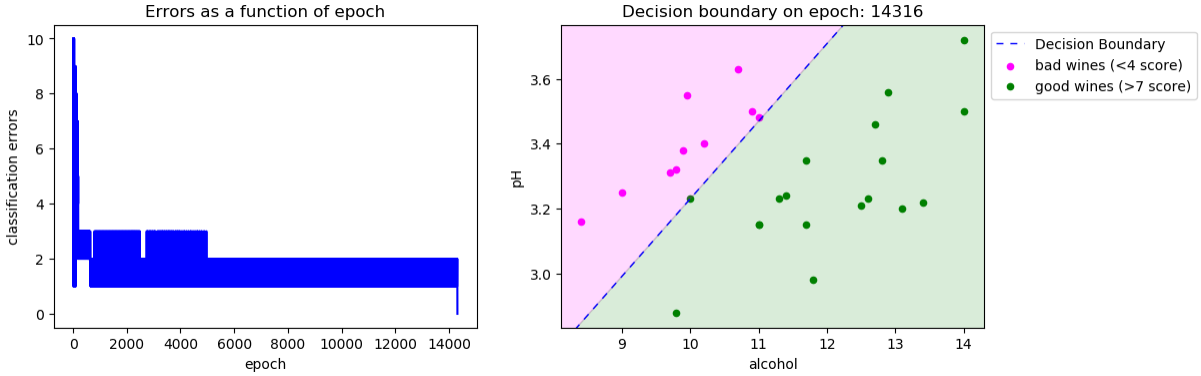
c) Now that you have a perceptron, a way to train your perceptron, and some data, you need a way to confirm that your perceptron actually works. Write a **function** that will take the output of your perceptron training function and your red wine data and **generate two plots in one figure:**

- The first plot should **plot the number of errors your perceptron made as a function of epoch.** Be careful with how you calculate errors!
- The second plot should **plot the decision boundary of your perceptron and also show 'good' and 'bad' wine data points on the final training epoch. This second plot should also shade 'good' and 'bad' areas!**
- Your function should allow the user to specify a specific epoch and see what the decision boundary of the perceptron was on that epoch. If a negative epoch is given, cause the plots to show the last epoch.

Here is an example function definition:

```
def plot_performance(performance, wine_data, good_thresh, bad_thresh, epoch=-1, save_plot=False):
    """
    Plot the performance of our perceptron or adaline.
    This function will produce a two plot figure:
    1) Classification Errors vs. Epochs
    2) Decision boundary for two factors
    """
```

Your figure with two plots should look similar to:



d) Your perceptron appears to work... but why is it taking so many epochs to train? Maybe you can modify the red wine data to help the perceptron learn more quickly? **Use the function you just created in part c) to plot and verify that your perceptron is learning more efficiently!**

Feature Scaling

It will be useful to set a specific random seed. This way, you'll know that the perceptron is learning more efficiently due to changes in the input and not due to the perceptron just getting lucky with its starting weights and bias.

V.3 My fair ADALINE

Encouraged by the results of your perceptron, you decide to see if you can use it to **distinguish between wines with a score of 4 and lower, and wines with a score of 7 and higher.** Feeling confident, you decide to let the perceptron train until it can find the best decision boundary.

a) Marvin notices and chides you for torturing your perceptron. Why?

You decide to upgrade your perceptron to handle this harder classification task. You've heard a bit about the gradient descent technique and also about ADALINE, maybe these will do the trick?

`Gradient descent, know it, love it!`

b) Implement an ADALINE that:

- Has randomly initialized weights and bias
- Uses a linear activation function and some kind of quantizer
- Uses the Widrow-Hoff learning rule

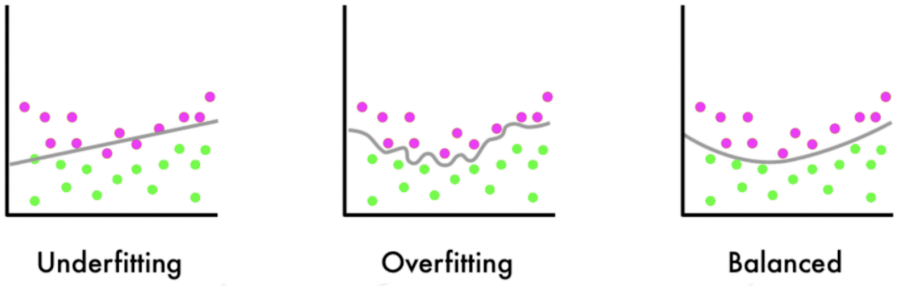
c) You'll need to implement a function to train your ADALINE. Your training function should:

- Take in your red wine data as a parameter
- Have a way to specify number of training epochs
- If training epochs is set to 0, your ADALINE should train until it converges on a good set of weights.
- Have a way to specify learning rate.
- Have an option to perform either online learning or batch learning

d) Find a good learning rate for your ADALINE and plot number of classification errors vs. epoch and the decision boundary of your ADALINE. What settings tend to give the smallest number of classification errors? Support your reasoning with example plots!

V.4 Advanced wine sampling and resampling

So far, you've been using all your data to train your ADALINE. Training your ADALINE many times on a dataset improves its performance for that dataset. But, too much training can lead to overfitting. This can lead to poor performance when the ADALINE encounters new data it has never seen before. Using a validation set and/or cross-validation methodologies reduces (but doesn't completely eliminate!) the possibility of overfitting!



a) Write a **function** that uses the holdout method to partition the red wine data into a training and a validation set. The function should take a parameter to adjust the proportion of training to validation data. It should return a tuple containing:

(training_pandas_dataframe, validation_pandas_dataframe)

If you have very few training data points, the holdout method can be expensive in terms of data usage and it may be better to use something like k-fold cross-validation.

b) Write a **function** that generates a k-fold cross-validation dataset from the red wine data. The function should allow k to be arbitrarily adjusted and also have an optional setting to shuffle data before creating k-folds. The function should return a list of 'k' tuples with **each** tuple containing:

(training_pandas_dataframe, cross_validation_pandas_dataframe)

You might want to take a look at how an established machine learning package does k-fold splitting. Scikit-learn would be a good start.

c) What effects does changing learning rate and number of training epochs have on the ADALINE when evaluated via k-fold cross-validation? To address this question, you should write (or modify) a function that will train and assess the ADALINE on each training and cross-validation fold produced by your k-fold function.

V.5 Adventures in the Nth dimension

Up until now, you've used only two factors in the wine data to train your perceptron and ADALINE. You wonder if it is possible to train your perceptron or ADALINE with even more chemical factors as inputs...

a) Try training your perceptron/ADALINE with different numbers and types of chemical factors. Under what circumstances can your perceptron/ADALINE successfully train?

b) You know what the decision boundary for 2 wine chemical factors looks like, but what does the decision boundary for 3 factors look like? What about if you use 7 factors? How about if you use all 11 wine chemical factors?

This resource may be useful for part b)

V.6 Marvin's rebuttal

Although you've managed to do a pretty decent job at predicting red wine quality given chemical characteristics, Marvin is not impressed. He gives you one more dataset for Pan-Galactic Gargle Blaster and challenges you to solve it with your **single** perceptron or ADALINE.

a) While not a wine... find a way to successfully classify the Pan-Galactic Gargle Blaster dataset. Show that your perceptron or ADALINE successfully classifies the Pan-Galactic Gargle Blaster data set by plotting the decision boundary and also show 'good' and 'bad' Gargle Blaster data points.

Problems similar to the one Marvin gave you were one of many factors that led to the AI Winter. You have the advantage of many more decades of research though!!

Pan Galactic Gargle Blaster? Drink...but...very carefully...

Chapter VI

Bonus part

VI.1 Gotta go fast!

Write (or maybe rewrite!) your perceptron and ADALINE algorithm functions (Initialization, Training, Learning, and Classification/Prediction functions) as Cython functions!

- You are allowed and encouraged to use "Cython magic" with the Jupyter notebook
- You must free all mallocs!
- No segfaults/double frees/bus errors!
- No norme though :)

Your imports should look like the following:

```
from libc.stdlib cimport malloc, free, rand, srand, RAND_MAX
from cpython cimport array
import array
```

Your perceptron struct typedef might look something like:

```
cdef struct s_perceptron:
    float *weights;
    float bias;
    int n_weights;
ctypedef s_perceptron t_perceptron
```

An example heaviside activation function definition:

```
cdef int heaviside(float activity_sum):
    if (activity_sum >= 0):
        return (1);
    else:
        return (0);
```

These resources should be useful!
Cython Tutorial
Cython Language Basics
Cython Memory Allocation

VI.2 Do perceptrons dream of electric sheep?

Peer into the decision making process of your perceptrons and ADALINEs!

Create animated plots of errors as a function of epoch **and** the changing decision boundary of your perceptron and/or ADALINE as a function of epoch. Each time step in your animation should be equivalent to advancing the training epoch by 1!

You'll probably want to do some googling on matplotlib animations in jupyter...The dockerized jupyter environment comes with everything you need to get animations working though! They're "libav-tools" and "ffmpeg" if you're curious...

VI.3 Dimensional traveler

Create a function that can plot the decision boundary when you train your perceptron or ADALINE on 3 different factors! Your plot will of course need to plot the actual wine data points in 3-factor space as well!

Did you know that matplotlib can create 3D plots?

Chapter VII

Turn-in and peer-evaluation

Submit your code to your `Git` repository as usual. Only the work in the repository will be considered for the evaluation. Any extraneous files will count against you unless justified.