Machine Learning in Sciences

Mendoza-Cortes Group

September 20, 2018

Contents

List of Tables

Page

List of Figures

Page

Chapter 1

Introduction and Background

Abstract

The following guide is intended for non CS majors that wonder if there will be any use of Machine Learning on their field. It contains applications to Art, Engineering, Physics, Medicine and Chemistry.

The problem we are going to focus on, is the following: supposed that you have measured data, for each point you have a label. Can we make reasonable predictions for the labels of new data values?

A big part of the quality of a machine learning algorithm relies on the data. If we don't have enough data, or if it is biased, the algorithms will reflect that in their classification. For example, we worked with certain 3D-scanners until we realized they don't work with people of color. The community on twitter suggested that it was an illumination issue but deciding that the algorithm is ready after it performs well on white people while not caring about the results on people of color is a choice that makes the algorithm biased. In principle you should never work on data that was collected for a different experiment. Since data is collected by humans, you should always look for mistakes by cleaning and preprocessing the data. The notebook 'L2 First Machine Learning Notebook' is an excellent guide on data analysis followed by 'L4 The Cocktail Party Problem' which requires more math and familiarity with programming.

1.0.1 SVC

Imagine a factory environment; heavy machinery under constant surveillance of some advanced system. The task of the controlling system is to determine when something goes wrong; the products are below quality, the machine produces strange vibrations or something like a temperature that rises. It is relatively easy to gather training data of situations that are OK; it is just the normal production situation. But on the other side, collection example data of a faulty system state can be rather expensive, or just impossible. If a faulty system state could be simulated, there is no way to guarantee that all the faulty states are simulated and thus recognized in a traditional two-class problem. We will se how SVM can solve this problem.

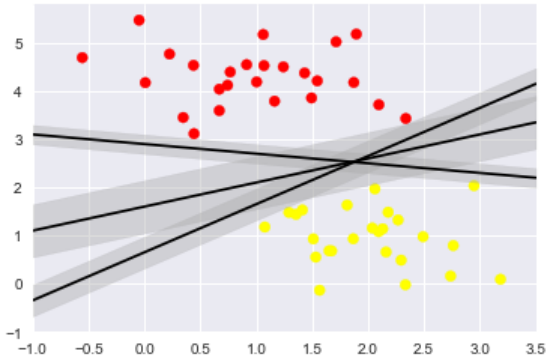

Figure 1.1

Let's return to the problem of classification of data. We have data with information about in which class do they belong. We want an algorithm to classify new unseen data. In good cases it is possible to use lines to separate classes and then the question becomes, among all possible lines, which one is best to use? as in fig 1.1. Once we have decided how to evaluate quality, we will be able to select a line. A natural requirement is that the algorithm should be robust to new data and minimize the number of miss classifications. In the notebook 'SVM' you can see how the algorithm SVC looks for the coefficients of such a line.

Most real life cases are not linearly separable, but sometimes we can transform the features hoping that the new features can be separated with hyper planes. On the image 1.2 we add a third coordinate to the data, the distance to the origin. Then in R^3 it is easy to find a plane $z = .6$

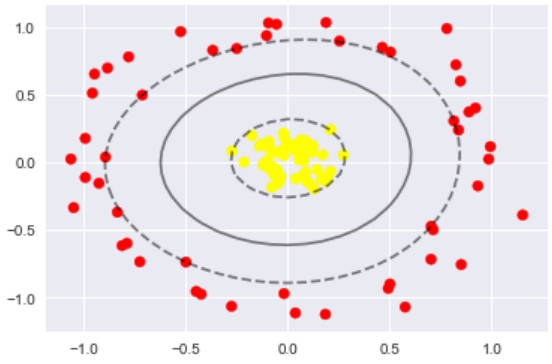

Figure 1.2

that classifies our data. SVM perform better if we pre-process the data following the rules on the notebook 'Cleaning data'.

How can we use this algorithm to find anomalies in new products? we can assume that good products have label 1 and products that are outliers have label 0. By training the SVC this way, it will learn to classify products are regular or outliers. It will recognize when a new product is far from the standard as in fig 1.3. This idea is explained in the notebook 'SVC II'.

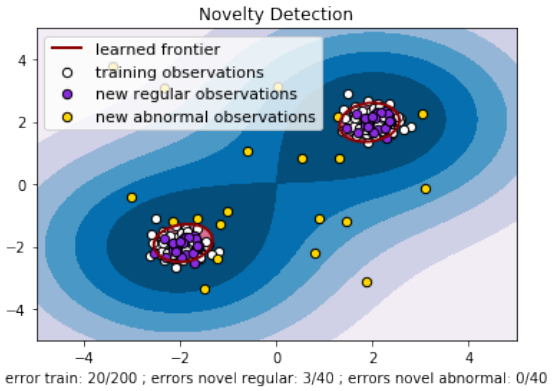

Figure 1.3

1.0.2 Neural Networks

What if instead of using lines or hyperplanes, we use non linear functions to classify our data? A neural network is an improvement over SVM, in the sense that the last component (layer) of a Neural Network behaves like a SVM.

A neural network is a combination of functions of the form: $x \to \sigma(W_{a,b}x)$ where $x \in R^b, W_{a,b}$ is a matriz transformation and σ stands for a non linear function applied entry wise to the a coordinates of the vector $W_{a,b}x$. It is standard notation to represent matrices $W_{a,b}$ as arrows from a column with b neurons to a column with a neurons as in 1.4. Note that in principle some activations could be the identity. If all activations are the identity we can use composition of matrix and reduce the algorithm to SVM. Finding the coefficients of those matrices requires multivariable calculus and relabeling the chain rule. In 'L5 Neural Networks' you can find an introduction to Keras, which we consider the most friendly language to use Neural Networks, and a deeper explanation of how

Neural Networks work and can be trained.

An important research question is: What determines the architecture of a Neural Network. The paper "Why does deep and cheap learning work so well?" targets that question from the Physics point of view.

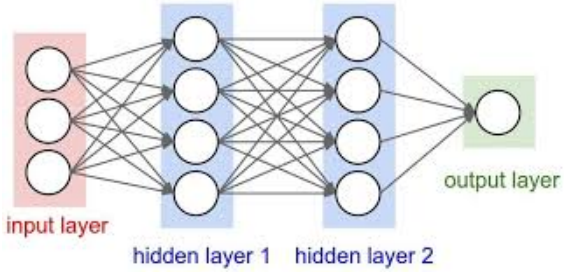

Figure 1.4

1.0.3 CNN

Given a picture of an Warehouse Shelving, a neural network can immediately identify the objects and add them to the inventory. Chinese government used face recognition to identify fugitives in a music concert. It is possible to train a special Neural Network called Convolutional Neural Network, to transfer the style of a paint into a different paint as we will see in 'L 19 Style Transfer' and 'L 25 Pix2Pix', see fig 1.5.

Figure 1.5

The main problem with images, is that they are represented as vectors that are too large. Go to your favorite browser and look for images with exactly 500×500 pixels. That is 250000×3 numbers. Normal Neural Networks will require you to find too many matrix coefficients. Besides that, we are not working any more on the case $R^b \rightarrow_{M_{a,b}} R^a$ as the pixels are related with the pixels in their neighborhood, including pixels above and below, so we want to use this information, which may be missing if we just think of an image as a very long vector.

We need to change the vectors input to matrices input, and the vectors output to matrix output. You can think of a matrix as a vector with vector entries. So now the matrices are not only operations but the elements we work on. Instead of learning a coefficient of a matrix, we learn a filter. The good thing of image processing is that we have visual help to understand the process. A filter may be geometric information that is present in several parts of the image, as the concept of a curve, or a square, see fig 1.6 .

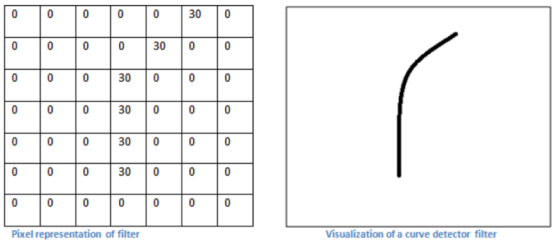

Figure 1.6

In 'L6 CNN' we explain in more detail how CNN work. In lecture "L 18" we experiment with Deep Dreams, where we ask the neural network if it recognizes a pattern and then we modify images to show that pattern. In L 19 we work with Style transfer, as in 1.5 where we used a picture of our team and we added the style of a paint of Botero. Lecture "L 25" is TBD.

Frequently we want to quantify the accuracy of our predictions. A Bayesian approach–one where we consider the values of the parameters as random variables is a popular method.

Recall Bayes' theorem for training parameters

$$P(\theta|X_{train}, T_{train}) = \frac{P(T_{train}, \theta|X_{train})}{\int P(T_{train}, \theta|X_{train})\, \mathrm{d}\theta} \tag{1.1}$$

The integral on the bottom is generally not analytic. And for higher dimensional problems, infeasible to calculate. By using Monte Carlo Markov Chain algorithm, it is possible to make estimations. This is also useful to study the Ising model, which was created to describe Magnetic and ferromagnetic materials. After this notebook, it is possible to read 'An exact mapping between the Variational Renormalization Group and Deep Learning' where they explain an important relationship between the Renormalization Group and deep learning architectures based on Restricted Boltzmann Machines.

Lecture L9 explains Genetic Algorithms. This algorithms are motivated by evolution, and work very well when it is not reasonable to make a simulation of all possible parameters. Instead we use rules of evolution to search for best ones. We describe the use of Genetic Algorithms to Reactive Force Fields.

L10 Introduces Decision Trees,

Software required: This guide will require a basic knowledge of python 3, we recommend to install it with Anaconda. We also have an introduction to Python. We will use Sklearn and Keras for most of the introductory part and Tensorflor, Pytorch for the advanced parts.

Chapter 2

Results

Chapter 3

Conclusions

Here we will put the most significant results over time

Appendix A

Scripts

Appendix B

Extra plots

Appendix C

Inputs

Appendix D

Things to be careful with

- Remember to update the path and home in constants when trying to run the create_MOF script from a folder outside of scratch.
- Remember to change the time step in the in.lammps file from 2 to 1.
- Some of the nets in the database are just named according to shape and not by shape and symmetry.
- Make sure there is at least one SBU of the correct symmetry for a framework to be created from a net.

Bibliography