Machine Learning Methods in Sciences

Mendoza-Cortes Group

January 18, 2019

Abstract

The following guide is intended for non CS majors that would like to use Machine Learning on their field. It contains applications to Art, Engineering, Physics, Medicine and Chemistry.

Chapter 1

Introduction and Background

Which machine learning algorithms should you use for your research or project? There are already several automatized programs that take your data and evaluate machine learning algorithms on it. By design, those methods don't tune the algorithm parameters or don't use the recent technology. In this notes we aim to give several examples of algorithms and applications to help you identify the method most suitable for your problem while understanding how to use the more capabilities of the algorithm. This will be an important difference if you are interested in research, or to optimize a solution to a problem. In case that all you want is a quick solution without worrying if it is optimal or not, you may prefer to use automatized methods instead, we will mention a couple of them in the corresponding section.

Our goal is to make this notes accessible for non CS majors with plenty of examples that will help you identify if your particular problem can be solve with the method.

Software required: This guide will require a basic knowledge of python 3, we recommend to install it with Anaconda. We also have notebooks that serve as an introduction to Python. Half of the lectures use Sklearn [1], the section on Neural Networks contains a lecture on Keras[2]. For the advanced parts we have lectures on Pytorch[3] and Tensorflow[4] where we introduce those tools.

1.0.1 Fairness

For motivational purposes assume that you have measured data, and for each value you have a label. Perhaps you have a list of characteristics of breast cancer samples and the label is 1 if the cancer is malignant or 0 if it is benign as in Breast cancer Wisconsin (diagnostic) dataset, another example is the data of police misconduct in Chicago and the labels are the officer's race as in Citizens Police Data Project. The main task for us is to find if we can we make reasonable predictions for the labels of new, unseen, data values.

The algorithm will return a class based on the algorithm's design, how we trained the algorithm and what we do with the output are choices made by humans and this will have consequences. We recommend to take the 70 min Fairness training to prevent different types of bias on your project.

1.0.2 Preprocessing Data

A big part of the performance of a machine learning algorithm relies on preprocessing the data. If we don't have enough data, or if it is biased, the algorithms will reflect those deficiencies in the output. For example, we worked with 3D-scanners until we realized they failed with people of color. The community on twitter suggested that it was an illumination issue, we claim that deciding that the algorithm is ready after testing it only on white people while not caring about the results on

people of color is a choice that makes the algorithm biased. After all, that scanner was used on toys and the toy producer was not expecting kids to have special illumination while using the toys.

An important lesson here is that data quality is a priority. You should avoid working on data that was collected for a different experiment. Since data is collected by humans, you should always look for mistakes by cleaning the data. The notebook " First Machine Learning Notebook" is an excellent guide on data analysis followed by "Higher Dimensions" which explains the curse of dimensionality and gives an introduction to higher dimensional geometry.

1.1 Machine Learning algorithms covered

1.1.1 SVC

Imagine that you are designing a new material. First you create it in vitro, and then you have a team that will materialize those designs. The task is to determine when something goes wrong before investing in the realization of the material; perhaps the atomic configuration is unstable or the material is below quality. It is relatively easy to gather training data of successful attempts. But on the other side, collection example data of a faulty system can be expensive, or just impossible. There is no way to describe a priory all the possible defects on the material. SVM can be use to attack this kind of problems as explained in the notebooks "Support Vector Machine".

The general idea is explained as follows:

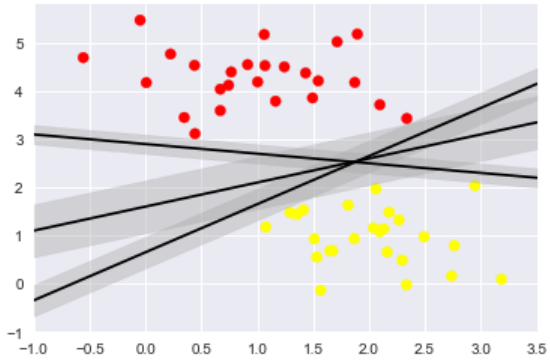

Figure 1.1

Assume we are dealing with classification of labeled 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 $z = .6$ that classifies our data. SVM performs 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,

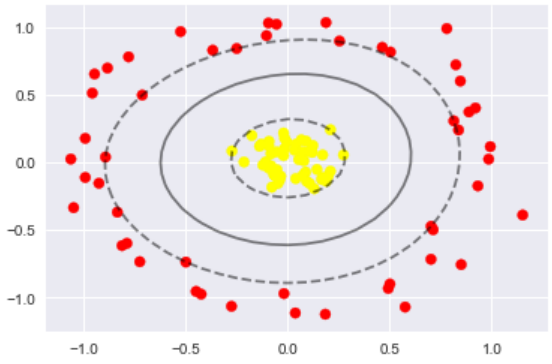

Figure 1.2

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".

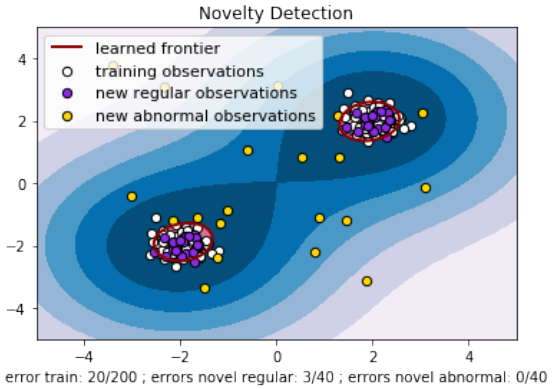

Figure 1.3

1.1.2 Neural Networks

We saw that SVM uses hyperplanes to separate the data, and that sometimes a transformation of the data is needed. In feed forward neural networks we use iterations of non linear functions followed by matrix transformations that transforms the data to create a classification.

A neural network is a composition of functions of the form: $x \to \sigma(W \cdot x + b)$ where x, b, are vectors, W is a matrix, and σ stands for a non linear function applied entry wise to the coordinates of the vector $W \cdot x + b$. It is standard notation to represent matrices W as arrows from the domain of x to the domain of $w \cdot x + b$ as in 1.4. For details see the notebook 'Neural Network'. Finding the coefficients of those matrices requires multivariable calculus and the chain rule. In "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 application of Neural Networks to chemistry is contained in the note "Poterntial Energy Surface ANN" where we describe the work of The Roitberg group in University of Florida, a transferable deep learning potential: ANAKIN-ME (Accurate NeurAl networK engINe for Molecular Energies) or ANI for short.

Supposed that you are given input in different steps, like if the input are letters of a name

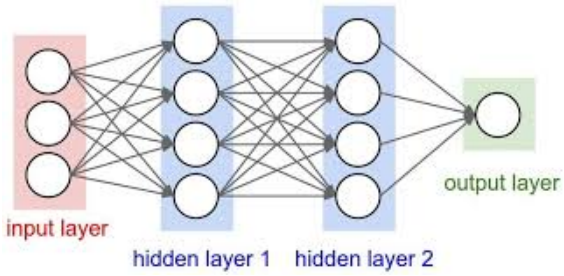

Figure 1.4

and you want to predict the nationality of the name. To deal with this kind of data, we need to modify our algorithms. Using time just as another variable will result in classification based on when did the event/measurement happened. But perhaps we want to find a classification based on classifications previous made on the partial data already registered. The lecture "Recurrent Neural Networks" deals with networks designed for modeling this sequential data.

This can be use to make numerical simulations of partial differential equations and ordinary differential equations, and we include an example with pytorch of Character-Level classification of names in "RNN II".

1.1.3 CNN

Given a picture of an Warehouse Shelving, a neural network can immediately identify the objects and restock the items. Chinese government used face recognition to identify fugitives in a music concert.

In order to work with images, we need to solve a problem, images are represented by pixels and a small image has too many pixels. Go to your favorite browser and look for images with exactly 500×500 pixels. That is 250000×3 numbers and Feed Forward Neural Networks will require you to find multiples of that many coefficients. Besides that, we are not working any more on the case $R^b \to_{M_{a,b}} R^a$ as the pixels are related with the pixels in their neighborhood.

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. The good thing of image processing is that we have visual help to understand the process. We are going to learn smaller matrices called filters, they may learn geometric concepts as a curve, or a square, see fig 1.5.

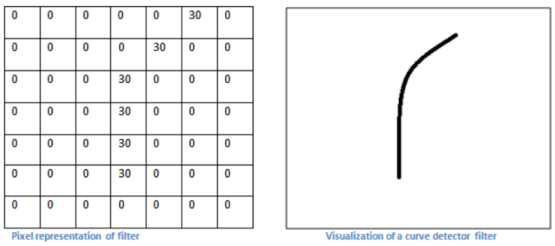

Figure 1.5

In the guest post "CNN" we explain in more detail how CNN work. In lecture "Deep Dreams" we ask the neural network if it recognizes a pattern and then we modify images to show that pattern. In " Style transfer" we learn images styles and impose them to other images, for example

in 1.6 we used a picture of our team and we added the style of a paint of Botero.

Figure 1.6

"Generative Adversarial Networks" describes a generative method. Suppose that we have a set of examples and we want to generate more images which look like the previous examples. For example, we might have the data set of fashion-mnist, a set of clothes' pictures and we want to generate artificial images as in this ghost wardrobe. While it is hard to train a GAN, it produces interesting results as in this GAN-paint or this password cracking tool.

We recommend to read Why does deep and cheap learning work so well?. The authors use physics to train to explain the way neural networks are capable to make classifications.

1.1.4 Bayesian methods

In "Frequentist vs Bayes" we introduce Bayesian methods and show how do they differ from the frequetist perspective.

Frequently we want to quantify the accuracy of our predictions. In the "Markov" lectures we find 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. We also introduce Simulated Annealing and Uncertainty Quantification.

1.1.5 SOM

Self Organizing Maps is a method that allow us to do dimensional reduction. The main idea is to place randomly points and let those points get attracted to the data points, in such a way that clusters will pull the points harder. We expect the points to end up placed in the center of the clusters. We will make this precise on the lecture "SOM". In the following picture we can see an image generated using Self Organizing Maps.

Figure 1.7: SOM image generated.

1.1.6 Genetic Algorithms

A genetic algorithm is a search heuristic inspired by Charles Darwins theory of natural evolution. They reflect the process of natural selection where the fittest individuals are selected for reproduction in order to produce the next generation.

Genetic Algorithms are most commonly used in optimization problems wherein we have to maximize or minimize a given objective function value under a given set of constraints. GAs are also used to characterize various economic models like the cobweb model, game theory equilibrium resolution, asset pricing, etc. They also have been used to plan the path which a robot arm takes by moving from one point to another. We can find more applications in multimodal optimization in which we have to find multiple optimum solutions.

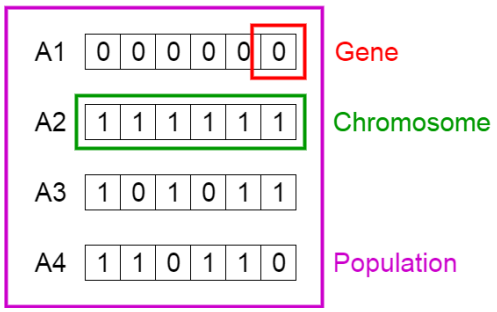

Figure 1.8

Given a problem we define a fitness function that measures how good is a proposed solution. The higher the fitness the closer to an ideal solution of my problem. The process of natural selection starts with the selection of fittest individuals from a random population, we also select a percentage

of the remaining population. This new smaller population produce offspring, mixing genes that seem useful to solve our problem. To explore new genes, we mutate a small percentage of the offspring and add foreigners to obtain the new population.

By doing this process several times, the maximum value obtained by the fitness function on the population cannot decrease as every new generation has the fittest individuals of the previous generation, while the mutations and foreigners help us avoid fixed points that are not minimums.

For more details see the notebook "Genetic Algorithm" and the notebook "GARFfield" where an application to reactive force fields is explained.

1.1.7 Decision Trees

Look at the following example:

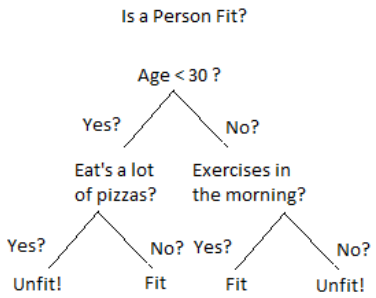

Figure 1.9

we asked several questions to determine an algorithm that will tell us if a person is fit or not based on their replies. The questions are not necessarily the optimal ones, but this illustrates the general methods: we split data points into subsets based on the values on certain features. The order in which we choose the features give us different algorithms. You can find an algorithm in the notebook "Decision Trees" where we aim to find the best features to consider in the decision.

1.1.8 KNN

In K-Nearest-Neighbors, we make a prediction or classify an element by only analyzing a neighborhood of the element an assigning to it a function of this neighborhood values, for example given a new point x and the closest k points to x, you can assign to x the average of x's neighbors values. K stands for how many points nearby do you consider to make your choice. The results depend on a good selection of the parameter K. If the boundary is a concern, Kernel Regression assigns different weights to the nearby points so that points that are closer matter most. Locally weighted polynomial regression is a local version of polynomial regression where we find the best line or curve that approximate the local nbh. The possibility to write a model that deals with the local picture is important in applications, in fig:1.10 we see examples locally weighted polynomial regression.

This should remind us of typical diagrams in phase transition as the Mexican hat 1.11.

1.1.9 Non Negative Tensor Factorization

We consider data in form of a matrix. For example, given a vocabulary with m words, and n documents, V_{ik} is the number of times the i -word in the dictionary appears on the k document. A

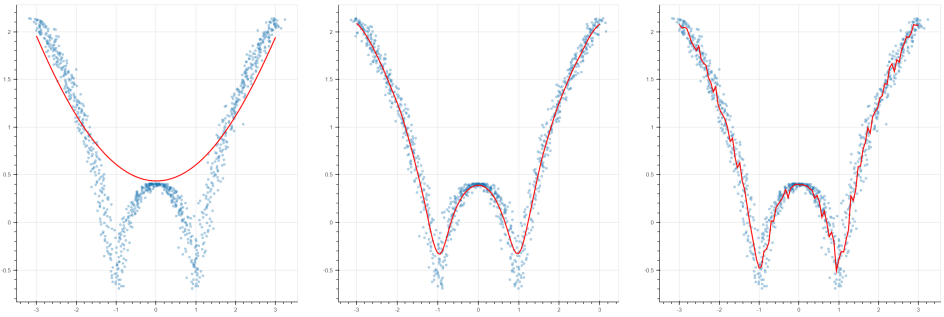

Figure 1.10

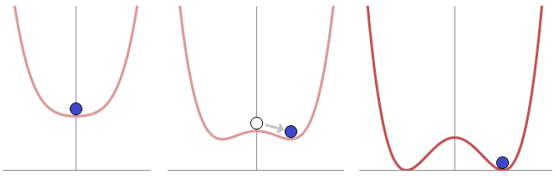

Figure 1.11: The 3d mexican hat potential.

database of gray-scale images of a fixed size can also be described by a matrix V. We relabel the pixels in an image form 1 to m, and assume the database has n images. The value of the V_{ik} is the intensity of the i-pixel in the k image.

Non negative matrix factorization is a process to find two matrices with non negative entries W, H so that $V \sim W \times H$. It turns out that the columns of H can be understood as a basis whose linear combination approximates the columns of V. W give us the coefficients so that a juxtaposition of the columns of H can recover the initial data. Columns of H have been recognized as eigen images [6].

Imagine that for every patient you have a matrix of diagnosis and medications. We can put the column of medications and the rows of diagnosis. So the entries will be mostly zeroes, except when the patient was diagnosed with the i-sickness and received the k-medication. If we consider a third dimension given by the patient, then we have a 3-dimensional matrix or a vector. There is a higher dimensional version of tensor factorization where we substitute matrices by tensors. The main yoga was that the non negativity constrain allow us to recover the initial object from juxtaposition of it's parts. When we replaced matrices by tensors we substitute the factorization into matrix multiplication by tensor decomposition into rank one tensors.

In Medicine, Tensor decomposition into rank one tensor shows concurring diagnosis and medications, this process is known as phenotyping. The study of tensors (phenotypes) using machine learning lead to the discovering of 3 distinct groups of Hearth Failure with preserved Ejection Fractions (HFpEF), those groups 'differed markedly in clinical characteristics, cardiac structure / function, invasive hemodynamics, and outcomes' [7].

1.2 Comparisons among methods

HERE WE CAN ADD A DIAGRAM THAT EXPLAINS WHEN TO USE WHICH METHOD WITH WHICH DATA

Bibliography

[1] Fabian Pedregosa and G Varoquaux. Scikit-learn: Machine learning in Python. ... *of Machine Learning* ..., 12:2825–2830, 2011.

[2] François Chollet et al. Keras. https://keras.io, 2015.

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Henry W. Lin, Max Tegmark, and David Rolnick. Why Does Deep and Cheap Learning Work So Well? *Journal of Statistical Physics*, 2017.

[6] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[7] Sanjiv J. Shah, Daniel H. Katz, Senthil Selvaraj, Michael A. Burke, Clyde W. Yancy, Mihai Gheorghiade, Robert O. Bonow, Chiang Ching Huang, and Rahul C. Deo. Phenomapping for novel classification of heart failure with preserved ejection fraction. *Circulation*, 131(3):269–279, 2015.