Data Analysis and Machine Learning: Introduction and Representing data

Morten Hjorth-Jensen[1,2]

Department of Physics, University of Oslo[1]

Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University[2]

Nov 27, 2017

What is Machine Learning?

Machine learning is the science of giving computers the ability to learn without being explicitly programmed. The idea is that there exist generic algorithms which can be used to find patterns in a broad class of data sets without having to write code specifically for each problem. The algorithm will build its own logic based on the data.

Machine learning is a subfield of computer science, and is closely related to computational statistics. It evolved from the study of pattern recognition in artificial intelligence (AI) research, and has made contributions to AI tasks like computer vision, natural language processing and speech recognition. It has also, especially in later years, found applications in a wide variety of other areas, including bioinformatics, economy, physics, finance and marketing.

Types of Machine Learning

The approaches to machine learning are many, but are often split into two main categories. In *supervised learning* we know the answer to a problem, and let the computer deduce the logic behind it. On the other hand, *unsupervised learning* is a method for finding patterns and relationship in data sets without any prior knowledge of the system. Some authours also operate with a third category, namely *reinforcement learning*. This is a paradigm of learning inspired by behavioural psychology, where learning is achieved by trial-and-error, solely from rewards and punishment.

Another way to categorize machine learning tasks is to consider the desired output of a system. Some of the most common tasks are:

- Classification: Outputs are divided into two or more classes. The goal is to produce a model that assigns inputs into one of these classes. An example is to identify digits based on pictures of hand-written ones. Classification is typically supervised learning.
- Regression: Finding a functional relationship between an input

Different algorithms

In this course we will build our machine learning approach on a statistical foundation, with elements from data analysis, stochastic processes etc before we proceed with the following machine learning algorithms

1. Linear regression and its variants
2. Decision tree algorithms, from simpler to more complex ones
3. Nearest neighbors models
4. Bayesian statistics
5. Support vector machines and finally various variants of
6. Artifical neural networks

Before we proceed however, there are several practicalities with data analysis and software tools we would like to present. These tools will help us in our understanding of various machine learning algorithms.

Our emphasis here is on understanding the mathematical aspects of different algorithms, however, where possible we will emphasize the importance of using available software

Software and needed installations

We will make intensive use of python as programming language and the myriad of available libraries. Furthermore, you will find IPython/Jupyter notebooks invaluable in your work. You can run **R** codes in the Jupyter/IPython notebooks, with the immediate benefit of visualizing your data.

If you have Python installed (we recommend Python3) and you feel pretty familiar with installing different packages, we recommend that you install the following Python packages via **pip** as

1. pip install numpy scipy matplotlib ipython scikit-learn mglearn sympy pandas pillow

For Python3, replace **pip** with **pip3**.

For OSX user we recommend also, after having installed Xcode, to install **brew**. Brew allows for a seamless installation of additional software via for example

1. brew install python3

Python installers

If you don't want to perform these operations separately, we recommend two widely used distrubutions which set up all relevant dependencies for Python, namely

1. anaconda
2. Enthought canopy

Installing R and C++

You will also find it convenient to utilize R. Say more about R. Jupyter/Ipython notebook allows you run **R** code interactively in your browser. The software library **R** is tuned to statistically analysis and allows for an easy usage of the tools we will discuss in these texts.

For the C++ affecianodas, Jupyter/IPython notebook allows you also to install C++ and run codes written in this language interactively in the browser. Since we will emphasize writing many of the algorithms yourself, you can thus opt for either Python or C++ as programming languages.

To add more entropy, **cython** can also be used when running your notebooks. It means that Python with the Jupyter/IPython notebook setup allows you to integrate widely popular softwares and tools for scientific computing. With its versatility, including symbolic operations, Python offers a unique computational environment. Your Jupyter/IPython notebook can easily be converted into a nicely rendered **PDF** file or a Latex file for further processing

Representing data, overarching aims

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import sparse
import pandas as pd
from IPython.display import display
eye = np.eye(4)
print(eye)
sparse_mtx = sparse.csr_matrix(eye)
print(sparse_mtx)
x = np.linspace(-10,10,100)
y = np.sin(x)
plt.plot(x,y,marker='x')
plt.show()
data = {'Name': ["John", "Anna", "Peter", "Linda"], 'Location': ["Roma
data_pandas = pd.DataFrame(data)
display(data_pandas)
```

Representing data, more examples

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import sparse
import pandas as pd
from IPython.display import display
import mglearn
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
x, y = mglearn.datasets.make_wave(n_samples=100)
line = np.linspace(-3,3,1000,endpoint=False).reshape(-1,1)
reg = DecisionTreeRegressor(min_samples_split=3).fit(x,y)
plt.plot(line, reg.predict(line), label="decision tree")
regline = LinearRegression().fit(x,y)
plt.plot(line, regline.predict(line), label= "Linear Rgression")
plt.show()
```