Python Machine-Learning Book

Python libraries:

NumPy, Matplotlib, scikit-learn, Pandas

**1 Giving Computers the ability to learn from data.**

**Machine Learning:**

* Supervised Learning
  + Learn a model from **labeled training data** . The desired output signals (labels) are already known.
    - Discrete. Class labels. Outcome signal is a classification task.
    - Continuous. Regression. Outcome signal is a continuous value.
  + Classification for prediction class labels. [Discrete]. Assign categorical, unordered labels to instances. Ex:
    - Binary classification: spam-nonspam model
      * A 2-dimensional dataset means that each sample has two values associated with it: x1 and x2.
    - Multi-class classification: Handwritten character recognition.
  + Regression for predicting continuous outcomes. [Continuous]. We are given a number of predictor (explanatory variables) X and a continuous response variable (outcome) Y and we try to find a relationship between those variables.Ex:
    - Given X and Y we fit a straight line to this data that minimizes the distance (optimal problem) - most commonly the average squared distance.
* Reinforcement learning
  + The goal is to develop a system (agent) that improves its performance based on interactions with the environment. *The state of the Agent affects the Environment and this returns to the agent with an Action and a Reward.* The agent will try to learn a series of actions that maximizes this reward via an exploratory trial-and-error approach or deliberative planning. Ex:
    - Chess game. The agent decides upon a series of moves depending on the state of the board (environment) and the reward can be defined as win or lose at the end of the game. *Here the reward signal is delibered a while after each action or move.*
* Unsupervised Learning. Discovering hidden structures.
  + We are dealing with unlabeled data or data of unknown structure. We will explore the structure of our data to extract meaningful information without the guidance of a known structure (supervised learning) or reward function (reinforced learning).
  + Finding subgroups with clustering (aka unsupervised classification). To organize data into meaningful subgroups withot any prior knowledge of their group memberships.
  + Dimensionality Reduction for data compression. Often dealing with high dimensionality data (each observation comes with a high number of measurements). *DR is used for preprocessing: to remove noise from data, compressing the data onto a smaller dimensional subspace while retaining most of the relevant information*. *Also useful for visualizing data (high dimensional data can be reduced into 3D or 2D scatterplots or histograms.*

**Terminology**:

ML uses vector-matrix notation.

Sample (rows in the matrix-superscript); Features (columns in the matrix-subscript); Measurement (value on each position).

**A Roadmap for building machine learning**

*(Important parts of a machine learning system accompanying the learning algorithm.)*

Typical workflow diagram for predictive modeling:

Machine generated alternative text:
Feature Extraction and 
Feature Select-ion 
Dimensionality Reduction 
Sampling 
Training Dataset 
Test Dataset 
Preprocessing 
Learning 
AlgMithrn 
Learning 
Final Model 
Evaluation 
Prediction 
Model Selection 
Cross-V i 
Performance Metrics 
Hyperparameter Optimization 

* Preprocessing. Crucial stage.
  + Many machine learning algorithms require that the selected features are on the same scale for optimal performance, often achieved by transforming the features in the range [0,1] or a standard normal distribution with zero mean and unit variance.
  + Some selected features may be highly correlated and therefore redundant -> Dimensionality reduction (less storage space) much faster learning algorithm.
  + To determine the performance of our algorithm the dataset can be randomly divided into training (to train and optimize out machine learning model) and test set (to evaluate the final model).

**Training and selecting a predictive model**

Many machine learning algorithms have been developed to solve different problem tasks.

Hyperparameter optimization techniques.

There are not universal models that can be applied to every scenario. One should be very careful when picking a model. Also, the model default parameters may be not optimal for our data, therefore we need to fine tune them (knobs analogy).

**Evaluating models and predicting unseen data instances**

The parameters for the previously mentioned procedures—such as feature scaling and dimensionality reduction—are solely obtained from the training dataset, and the same parameters are later re-applied to transform the test dataset, as well as any new data samples—the performance measured on the test data may be overoptimistic otherwise.

**Python for machine learning**

Although the performance of interpreted languages, such as python, for computation-intensive tasks is inferior to lower-level programming languages, extension libraries such as ***NumPy*** and ***SciPy*** have been developed that build upon lower layer Fortran and C implementations for fast and vectorized operations on multidimensional arrays. ***Scikit-learn*** Open source machine learning library.

**2 Training Machine Learning Algorithms for Classification**

One of the first algorithmically described machine learning algorithms for classification:

* Perceptron
* Adaptive linear neurons

Points covered in the chapter:

* Building an intuition for machine learning algorithms
* Using pandas,Numpy and matplotlib to read in , process and visualize data
* Implementing linear classification algorithms in Python

**Artificial neurons- early story of machine learning**

Neurons are interconnected nerve cells : processing and transmitting signals.

Perceptron learning: Rosenblatt proposed an algorithm that would automatically learn the optimal weight coefficients that are then multiplied with the input features in order to make the decision of whether a neuron fires or not.

Supervised learning+Perceptron algorithm= predict if a sample belongs to one class or the other.

Perceptron: creates a threshold to make the differentiation.

This *thresholded* perceptron model uses a reductionist approach to mimic how a single neuron in the brain works: it either fires or it doesn’t.

z=w1x1+w2x2+…wmxm

Sigma(z)={ 1 if z>= theta (threshold); -1 otherwise

*Perceptron steps:*

1. Initialize the weights to 0 or small random numbers
2. For each training sample perform the following steps
   1. Compute the output value y
   2. Update the weights

Updating weights:

w\_j:=w\_j+Delta w\_j

Perceptron learning rule: Delta w\_j= \eta (y^I - \hat{y}^i)\*x\_j^I

\eta is the learning rate 0.0-1.0

\*All the weights are updated simultaneously

Machine generated alternative text:
AW2 
output( ) 
output 

The "push" is directly proportional to the value of the measure.

THE CONVERGENCE OF THE PERCEPTRON is only guaranteed if the two classes are linearly separable and the learning rate is sufficiently small. In the opposite case we can set a maximum number of passes over the training dataset (epochs) and/or a threshold for the number of tolerated misclassifications- otherwise the perceptron would never stop updating the weights.

Machine generated alternative text:
Now, before we jump into the implementation in the next section, let us summarize 
what we just learned in a simple figure that illustrates the general concept of 
the perceptron: 
Weight update 
Error 
Net input 
function 
Output 
1 
Activatio n 
function 

The perceptron algorithm is not only used for binary cases but also on multiclass problems : One-vs-All / One-vs-Rest.