

# Machine Learning Algorithms Illustrated in Python

Cameron P. Dart

March 6, 2016

## Abstract

In order to gain a deeper understanding of artificial intelligence algorithms, I write on commonly used *supervised* and *unsupervised* learning algorithms. While researching these algorithms, I will create my own implementations of each in Python, describe and prove the mathematical reasoning, and analyze the space-time complexities of my own to open source implementations of these same algorithms in Big  $\mathcal{O}$ ,  $\theta$ , and  $\Omega$  notation.

## 1 Introduction

In this paper, I will strictly define machine learning and give mathematical descriptions and intuition on one *supervised* and one *unsupervised* learning algorithm. Additionally, I will provide an explanation of the differences between  $\mathcal{O}(n)$ ,  $\Theta(n)$ , and  $\Omega(n)$  in regards run time and space complexity. Lastly, I will combine the mathematical descriptions of algorithms and space/time complexity to analyze the differences between my own and open source implementations of the same algorithms. Machine Learning algorithms are repetitive in their computations so before writing any code, it would be most useful to write the algorithm by hand with a small sample set. This will be beneficial to understanding where inefficiencies in code could occur, or what mathematical concepts I can exploit using the computational power of a computer. Once I have the code written, analysis, I will move forward to writing my own algorithm in Python.

## 2 Method of Analysis

In order to get a better understanding of the algorithms used, I will use Python modules such as *memory* and *os* [2] to gather data on programs, functions, and processes run time and memory usage. The module *memory* will be used as a way to keep track of how much memory a Python process is consuming. While *os* will be used to test the runtime a certain function. I will provide line by line theoretical analysis with run times of functions and operations provided by Python documentation so I can further see the short comings and advantages of each implementation. Additionally, I will prove which runs more quickly and uses less space through test cases. I will run test cases with controlled input size and examine the results to confirm my hypothesizes. With all of the

theoretical and experimental data at my hand, I will be able to decide which implementation the most efficient.

### 3 Space-Complexity Analysis

Let there exist positive constants  $c_o$ ,  $c_1$ ,  $c_2$ , and  $n_0$ .  
As well as functions  $f$  and  $g$

#### 3.1 Big $\mathcal{O}(n)$

**Formal Definition**  $f(n) \leq Cg(n)$  whenever  $n > n_0$   
 $f(n)$  is  $\mathcal{O}(g(n)) \equiv (\exists C) (\exists n_0) (\forall n) (n > n_0 \rightarrow f(n) \leq Cg(n))$  [1]

**Informal Definition** A function  $f$  is considered to be  $(O)g$  if it is never larger than some constant multiplied by  $g$ .

#### 3.2 Big $\Omega(n)$

**Formal Definition** Let  $C$  and  $n_0$  be positive constants  
 $f(n)$  is  $\Omega(g(n)) \equiv (\exists C) (\exists n_0) (\forall n) (n > n_0 \rightarrow f(n) \geq Cg(n))$  [1]

**Informal Definition** A function  $f$  is considered to be  $\Omega(n)$  if it is always larger than some constant multiplied by  $g$

#### 3.3 Big $\Theta(n)$

**Formal Definition** the function  $f$  is considered to be  $\Theta(g(n)) \iff (c_1g(n)) \leq f(n) \leq (c_2g(n))$  [1]

**Informal Definition** A function  $f$  is considered to be  $\Theta(g(n))$  if it is bounded by two separate functions.

## 4 Machine Learning

The field of study that gives computers the ability to learn without being explicitly programmed. In general, Machine Learning algorithms produce patterns based on given input and use the patterns to predict future cases.

## 5 Supervised Learning Algorithms

### 5.1 Definition

*Supervised Learning* is the task of deducing a function from a given *training set*. [4] A *training set* is a vector used for the initial discovery of relationships between variables; ie to fit the weights of the classifying function. In order to prevent *overfitting*, deducing conclusions when none exist, a *validation set* is used in case any classification parameter needs to be adjusted. Further on, a *test set* is used to gauge the efficiency of a given model. One classic example of a *supervised learning* algorithm is a *linear regression*.

## 5.2 Example: Linear Regression

### 5.2.1 Definition

A *linear regression* is an approach for modeling the relationship between a scalar dependent variable  $y$  and one or more explanatory variables (or independent variables) denoted  $X$ . Specifically the case of one explanatory variable is called a *linear regression*. A sample data set for a linear regression would be pairs  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  where the regression models  $y_i$  as a function of  $x_i$

### 5.2.2 Mathematical Description

The *regression line* can be determined by finding *sum of squares* method. A *fitted linear regression line* can be described as

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 * x \quad [3] \quad (1)$$

$\hat{\beta}_0$  is defined as

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (2)$$

Given  $n$  is the number of data points in the *training set*

$$\hat{\beta}_1 = \left( \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2} \right) \quad (3)$$

An easier way to express  $\hat{\beta}_1$  can be defined as

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \quad (4)$$

Where  $S_{xy}$  and  $S_{xx}$  are

$$\begin{aligned} S_{xy} &= \sum (x_i - \bar{x})(y_i - \bar{y}) \\ &= \sum x_i y_i - \frac{(\sum x_i)(\sum y_i)}{n} \end{aligned}$$

$$\begin{aligned} S_{xx} &= \sum (x_i - \bar{x})^2 \\ &= \sum x_i^2 - \frac{(\sum x_i)^2}{n} \end{aligned}$$

Given  $\bar{y} \equiv \frac{1}{n} \sum y_i$  and  $\bar{x} \equiv \frac{1}{n} \sum x_i$

### 5.2.3 Example Use Cases

A doctor is given You are given a vector  $V$  of inputs such that  $V = \{(x_1, y_1), \dots, (x_n, y_n)\}$  where  $x_i$  is the height and  $y_i$  is an weight for the  $i^{th}$  individual. You want to predict the weight of an individual

## References

- [1] <https://www.cs.auckland.ac.nz/courses/compsci220s1t/lectures/lecturenotes/GG-lectures/220handout-lecture03.pdf>
- [2] <https://wiki.python.org/moin/TimeComplexity>
- [3] <http://www2.isye.gatech.edu/sman/courses/6739/SimpleLinearRegression.pdf>
- [4] <http://cs229.stanford.edu/notes/cs229-notes1.pdf>