**Basics of Mathematical Notation for Machine Learning**

by [**Jason Brownlee**](https://machinelearningmastery.com/author/jasonb/) on March 26, 2018  in [**Linear Algebra**](https://machinelearningmastery.com/category/linear-algebra/)

Tweet   **Share** **Share**

You cannot avoid mathematical notation when reading the descriptions of machine learning methods.

Often, all it takes is one term or one fragment of notation in an equation to completely derail your understanding of the entire procedure. This can be extremely frustrating, especially for machine learning beginners coming from the world of development.

You can make great progress if you know a few basic areas of mathematical notation and some tricks for working through the description of machine learning methods in papers and books.

In this tutorial, you will discover the basics of mathematical notation that you may come across when reading descriptions of techniques in machine learning.

After completing this tutorial, you will know:

* Notation for arithmetic, including variations of multiplication, exponents, roots, and logarithms.
* Notation for sequences and sets including indexing, summation, and set membership.
* 5 Techniques you can use to get help if you are struggling with mathematical notation.

Let’s get started.

* **Update May/2018**: Added images for some notations to make the explanations clearer.



Basics of Mathematical Notation for Machine Learning  
Photo by [Christian Collins](https://www.flickr.com/photos/collins_family/33746908612/), some rights reserved.

**Tutorial Overview**

This tutorial is divided into 7 parts; they are:

1. The Frustration with Math Notation
2. Arithmetic Notation
3. Greek Alphabet
4. Sequence Notation
5. Set Notation
6. Other Notation
7. Getting More Help

Are there other areas of basic math notation required for machine learning that you think I missed?  
Let me know in the comments below.

**Need help with Linear Algebra for Machine Learning?**

Take my free 7-day email crash course now (with sample code).

Click to sign-up and also get a free PDF Ebook version of the course.

[**Download Your FREE Mini-Course**](https://machinelearningmastery.lpages.co/leadbox/143e9e9f3f72a2%3A164f8be4f346dc/5667039158992896/)

**The Frustration with Math Notation**

You will encounter mathematical notation when reading about machine learning algorithms.

For example, notation may be used to:

* Describe an algorithm.
* Describe data preparation.
* Describe results.
* Describe a test harness.
* Describe implications.

These descriptions may be in research papers, textbooks, blog posts, and elsewhere.

Often the terms are well defined, but there are also mathematical notation norms that you may not be familiar with.

All it takes is one term or one equation that you do not understand and your understanding of the entire method will be lost. I’ve suffered this problem myself many times, and it is incredibly frustrating!

In this tutorial, we will review some basic mathematical notation that will help you when reading descriptions of machine learning methods.

**Arithmetic Notation**

In this section, we will go over some less obvious notations for basic arithmetic as well as a few concepts you may have forgotten since school.

**Simple Arithmetic**

The notation for basic arithmetic is as you would write it. For example:

* Addition: 1 + 1 = 2
* Subtraction: 2 – 1 = 1
* Multiplication: 2 x 2 = 4
* Division: 2 / 2 = 1

Most mathematical operations have a sister operation that performs the inverse operation; for example, subtraction is the inverse of addition and division is the inverse of multiplication.

**Algebra**

We often want to describe operations abstractly to separate them from specific data or specific implementations.

For this reason we see heavy use of algebra: that is uppercase and/or lowercase letters or words to represents terms or concepts in mathematical notation. It is also common to use letters from the Greek alphabet.

Each sub-field of math may have reserved letters: that is terms or letters that always mean the same thing. Nevertheless, algebraic terms should be defined as part of the description and if they are not, it may just be a poor description, not your fault.

**Multiplication Notation**

Multiplication is a common notation and has a few short hands.

Often a little “x” or an asterisk “\*” is used to represent multiplication:

|  |  |
| --- | --- |
| 1  2 | c = a x b  c = a \* b |

You may see a dot notation used; for example:

|  |  |
| --- | --- |
| 1 | c = a . b |

Which is the same as:

|  |  |
| --- | --- |
| 1 | c = a \* b |

Alternately, you may see no operation and no white space separation between previously defined terms; for example:

|  |  |
| --- | --- |
| 1 | c = ab |

Which again is the same thing.

**Exponents and Square Roots**

An exponent is a number raised to a power.

The notation is written as the original number, or the base, with a second number, or the exponent, shown as a superscript; for example:

|  |  |
| --- | --- |
| 1 | 2^3 |

Which would be calculated as 2 multiplied by itself 3 times, or cubing:

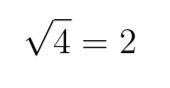
|  |  |
| --- | --- |
| 1 | 2 x 2 x 2 = 8 |

A number raised to the power 2 to is said to be its square.

|  |  |
| --- | --- |
| 1 | 2^2 = 2 x 2 = 4 |

The square of a number can be inverted by calculating the square root. This is shown using the notation of a number and with a tick above, I will use the “sqrt()” function here for simplicity.

|  |  |
| --- | --- |
| 1 | sqrt(4) = 2 |

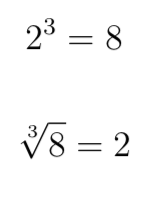


Here, we know the result and the exponent and we wish to find the base.

In fact, the root operation can be used to inverse any exponent, it just so happens that the default square root assumes an exponent of 2, represented by a subscript 2 in front of the square root tick.

For example, we can invert the cubing of a number by taking the cube root (note, the 3 is not a multiplication here, it is notation before the tick of the root sign):

|  |  |
| --- | --- |
| 1  2 | 2^3 = 8  3 sqrt(8) = 2 |

****

**Logarithms and e**

When we raise 10 to an integer exponent, we often call this an order of magnitude.

|  |  |
| --- | --- |
| 1 | 10^2 = 10 x 10 or 100 |

Another way to reverse this operation is by calculating the logarithm of the result 100 assuming a base of 10; in notation this is written as log10().

|  |  |
| --- | --- |
| 1 | log10(100) = 2 |

Here, we know the result and the base and wish to find the exponent.

This allows us to move up and down orders of magnitude very easily. Taking the logarithm assuming the base of 2 is also commonly used, given the use of binary arithmetic used in computers. For example:

|  |  |
| --- | --- |
| 1  2 | 2^6 = 64  log2(64) = 6 |

Another popular logarithm is to assume the natural base called e. The e is reserved and is a special number or a constant called Euler’s number (pronounced “*oy-ler*“) that refers to a value with practically infinite precision.

|  |  |
| --- | --- |
| 1 | e = 2.71828... |

Raising e to a power is called a natural exponential function:

|  |  |
| --- | --- |
| 1 | e^2 = 7.38905... |

It can be inverted using the natural logarithm, which is denoted as ln():

|  |  |
| --- | --- |
| 1 | ln(7.38905...) = 2 |

Without going into detail, the natural exponent and natural logarithm prove useful throughout mathematics to abstractly describe the continuous growth of some systems, e.g. systems that grow exponentially such as compound interest.

**Greek Alphabet**

Greek letters are used throughout mathematical notation for variables, constants, functions, and more.

For example, in statistics we talk about the mean using the lowercase Greek letter mu, and the standard deviation as the lowercase Greek letter sigma. In linear regression, we talk about the coefficients as the lowercase letter beta. And so on.

It is useful to know all of the uppercase and lowercase Greek letters and how to pronounce them.

When I was a grad student, I printed the Greek alphabet and stuck it on my computer monitor so that I could memorize it. A useful trick!

Below is the full Greek alphabet.



Greek Alphabet, [Taken from Wikipedia](https://en.wikipedia.org/wiki/Greek_letters_used_in_mathematics,_science,_and_engineering)

The Wikipedia page titled “[Greek letters used in mathematics, science, and engineering](https://en.wikipedia.org/wiki/Greek_letters_used_in_mathematics,_science,_and_engineering)” is also a useful guide as it lists common uses for each Greek letter in different sub-fields of math and science.

**Sequence Notation**

Machine learning notation often describes an operation on a sequence.

A sequence may be an array of data or a list of terms.

**Indexing**

A key to reading notation for sequences is the notation of indexing elements in the sequence.

Often the notation will specify the beginning and end of the sequence, such as 1 to n, where n will be the extent or length of the sequence.

Items in the sequence are index by a variable such as i, j, k as a subscript. This is just like array notation.

For example, a\_i is the i^th element of the sequence a.

If the sequence is two dimensional, two indices may be used; for example:

b\_{i,j} is the i,j^th element of the sequence b.

**Sequence Operations**

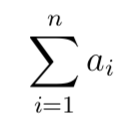
Mathematical operations can be performed over a sequence.

Two operations are performed on sequences so often that they have their own shorthand: the sum and the multiplication.

**Sequence Summation**

The sum over a sequence is denoted as the uppercase Greek letter sigma. It is specified with the variable and start of the sequence summation below the sigma (e.g. i = 1) and the index of the end of the summation above the sigma (e.g. n).

|  |  |
| --- | --- |
| 1 | Sigma i = 1, n a\_i |

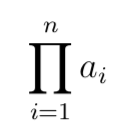


This is the sum of the sequence a starting at element 1 to element n.

**Sequence Multiplication**

The multiplication over a sequence is denoted as the uppercase Greek letter pi. It is specified in the same way as the sequence summation with the beginning and end of the operation below and above the letter respectively.

|  |  |
| --- | --- |
| 1 | Pi i = 1, n a\_i |



This is the product of the sequence a starting at element 1 to element n.

**Set Notation**

A set is a group of unique items.

We may see set notation used when defining terms in machine learning.

**Set of Numbers**

A common set you may see is a set of numbers, such as a term defined as being within the set of integers or the set of real numbers.

Some common sets of numbers you may see include:

* Set of all natural numbers: N
* Set of all integers: Z
* Set of all real numbers: R

There are other sets; see [Special sets on Wikipedia](https://en.wikipedia.org/wiki/Set_(mathematics)#Special_sets).

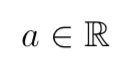
We often talk about real values or real numbers when defining terms rather than floating point values, which are really discrete creations for operations in computers.

**Set Membership**

It is common to see set membership in definitions of terms.

Set membership is denoted as a symbol that looks like an uppercase “E”.

|  |  |
| --- | --- |
| 1 | a E R |



Which means a is defined as being a member of the set R or the set of real numbers.

There is also a host of set operations; two common set operations include:

* Union, or aggregation: A U B
* Intersection, or overlap: A ^ B

Learn more about [sets on Wikipedia](https://en.wikipedia.org/wiki/Set_(mathematics)).

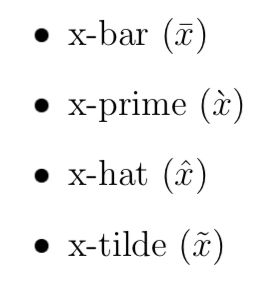
**Other Notation**

There is other notation that you may come across.

I try to lay some of it out in this section.

It is common to define a method in the abstract and then define it again as a specific implementation with separate notation.

For example, if we are estimating a variable x, we may represent it using a notation that modifies the x; for example:



The same notation may have a different meaning in a different context, such as use on different objects or sub-fields of mathematics. For example, a common point of confusion is |x|, which, depending on context, can mean:

* |x|: The absolute or positive value of x.
* |x|: The length of the vector x.
* |x|: The cardinality of the set x.

This tutorial only covered the basics of mathematical notation. There are some subfields of mathematics that are more relevant to machine learning and should be reviewed in more detail. They are:

* Linear Algebra.
* Statistics.
* Probability.
* Calculus.

And perhaps a little bit of multivariate analysis and information theory.

Are there areas of mathematical notation that you think are missing from this post?  
Let me know in the comments below.

**5 Tips for Getting Help with Math Notation**

This section lists some tips that you can use when you are struggling with mathematical notation in machine learning.

**Think About the Author**

People wrote the paper or book you are reading.

People that can make mistakes, make omissions, and even make things confusing because they don’t fully understand what they are writing.

Relax the constraints of the notation you are reading slightly and think about the intent of the author. What are they trying to get across?

Perhaps you can even contact the author via email, Twitter, Facebook, LinkedIn, etc., and seek clarification. Remember that academics want other people to understand and use their work (mostly).

**Check Wikipedia**

Wikipedia has lists of notation which can help narrow down on the meaning or intent of the notation you are reading.

Two places I recommend you start are:

* [List of mathematical symbols on Wikipedia](https://en.wikipedia.org/wiki/List_of_mathematical_symbols)
* [Greek letters used in mathematics, science, and engineering on Wikipedia](https://en.wikipedia.org/wiki/Greek_letters_used_in_mathematics,_science,_and_engineering)

**Sketch in Code**

Mathematical operations are just functions on data.

Map everything you’re reading to pseudocode with variables, for-loops, and more.

You might want to use a scripting language as you go, along with small arrays of contrived data or even an Excel spreadsheet.

As your reading and understanding of the technique improves, your code-sketch of the technique will make more sense, and at the end you will have a mini prototype to play with.

I never used to take much stock in this approach until I saw an academic sketch out a very complex paper in a few lines of matlab with some contrived data. It knocked my socks off because I believed the system had to be coded completely and run with a “real” dataset and that the only option was to get the original code and data. I was very wrong. Also, looking back, the guy was gifted.

I now use this method all the time and sketch techniques in Python.

**Seek Alternatives**

There is a trick I use when I’m trying to understand a new technique.

I find and read all the papers that reference the paper I’m reading with the new technique.

Reading other academics interpretation and re-explanation of the technique can often clarify my misunderstandings in the original description.

Not always though. Sometimes it can muddy the waters and introduce misleading explanations or new notation. But more often than not, it helps. After circling back to the original paper and re-reading it, I can often find cases where subsequent papers have actually made errors and misinterpretations of the original method.

**Post a Question**

There are places online where people love to explain math to others. Seriously!

Consider taking a screenshot of the notation you are struggling with, write out the full reference or link to it, and post it and your area of misunderstanding to a question-and-answer site.

Two great places to start are:

* [Mathematics Stack Exchange](https://math.stackexchange.com/)
* [Cross Validated](https://stats.stackexchange.com/)

What are your tricks for working through mathematical notation?  
Let me know in the comments below?