I'm working on things related to norm a lot lately and it is time to talk about it. In this post we are going to discuss about a whole family of norm.

What is a norm?

Mathematically a norm is a total size or length of all vectors in a vector space or matrices. For simplicity, we can say that the higher the norm is, the bigger the (value in) matrix or vector is. Norm may come in many forms and many names, including these popular name: Euclidean distance, Mean-squared Error, etc.

Most of the time you will see the norm appears in a equation like this:

\[ \| \mathbf{x} \| \]

where \( \mathbf{x} \) can be a vector or a matrix.

For example, a Euclidean norm of a vector \( \mathbf{a} = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix} \) is \( \| \mathbf{a} \|_2 = \sqrt{3^2 + (-2)^2 + 1^2} = 3.742 \) which is the size of vector \( \mathbf{a} \).

The above example shows how to compute a Euclidean norm, or formally called an \( l_2 \)-norm. There are many other types of norm that beyond our explanation here, actually for every single real number, there is a norm correspond to it (Notice the emphasised word real number, that means it not limited to only integer.)

Formally the \( l_p \)-norm of \( \mathbf{x} \) is defined as:

\[ \| \mathbf{x} \|_p = \left( \sum_{i} |x_i|^p \right)^{1/p} \quad \text{where} \quad p \in \mathbb{R} \]

That’s it! A \( p \)-th-root of a summation of all elements to the \( p \)-th power is what we call a norm.

The interesting point is even though every \( l_p \)-norm is all look very similar to each other, their mathematical properties are very different and thus their application are dramatically different too. Hereby we are going to look into some of these norms in details.

\( l_0 \)-norm

The first norm we are going to discuss is a \( l_0 \)-norm. By definition, \( l_0 \)-norm of \( \mathbf{x} \) is

\[ \| \mathbf{x} \|_0 = \sqrt{\sum_{i} |x_i|^0} \]

Strictly speaking, \( l_0 \)-norm is not actually a norm. It is a cardinality function which has its definition in the form of \( l_0 \)-norm, though many people call it a norm. It is a bit tricky to work with because there is a presence of zeroth-power and zeroth-root in it. Obviously any \( x_i > 0 \) will become one, but the problems of the definition of zeroth-power and especially zeroth-root is messing things around here. So in reality, most mathematicians and engineers use this definition of \( l_0 \)-norm instead:

\[ \| \mathbf{x} \|_0 = \# \{i | x_i \neq 0 \} \]

that is a total number of non-zero elements in a vector.

Because it is a number of non-zero element, there is so many applications that use \( l_0 \)-norm. Lately it is even more in focus because of the rise of the Compressive Sensing scheme, which is try to find the sparsest solution of the under-determined linear system. The sparsest solution means the solution which has fewest non-zero entries, i.e. the lowest \( l_0 \)-norm. This problem is usually regarding as a optimisation problem of \( l_0 \)-norm or \( l_0 \)-optimisation.

\( l_0 \)-optimisation

Many application, including Compressive Sensing, try to minimise the \( l_0 \)-norm of a vector corresponding to some constraints, hence called “\( l_0 \)-minimisation”. A standard minimisation problem is formulated as:
However, doing so is not an easy task. Because the lack of $l_0$-norm’s mathematical representation, $l_0$-minimisation is regarded by computer scientist as an NP-hard problem, simply says that it’s too complex and almost impossible to solve. In many case, $l_0$-minimisation problem is relaxed to be higher-order norm problem such as $l_1$-minimisation and $l_2$-minimisation.

### $l_1$-norm

Following the definition of norm, $l_1$-norm of $x$ is defined as

$$ \|x\|_1 = \sum |x_i| $$

This norm is quite common among the norm family. It has many name and many forms among various fields, namely *Manhattan norm* is its nickname. If the $l_1$-norm is computed for a difference between two vectors or matrices, that is

$$ SAD(x_1, x_2) = \|x_1 - x_2\|_1 = \sum |x_{1i} - x_{2i}| $$

it is called *Sum of Absolute Difference (SAD)* among computer vision scientists.

In more general case of signal difference measurement, it may be scaled to a unit vector by:

$$ MAE(x_1, x_2) = \frac{1}{n} \|x_1 - x_2\|_1 = \frac{1}{n} \sum |x_{1i} - x_{2i}| $$

where $n$ is a size of $x$.

which is known as *Mean-Absolute Error (MAE)*.

### $l_2$-norm

The most popular of all norm is the $l_2$-norm. It is used in almost every field of engineering and science as a whole. Following the basic definition, $l_2$-norm is defined as

$$ \|x\|_2 = \sqrt{\sum x_i^2} $$

$l_2$-norm is well known as a *Euclidean norm*, which is used as a standard quantity for measuring a vector difference. As in $l_1$-norm, if the Euclidean norm is computed for a vector difference, it is known as a *Euclidean distance*:

$$ \|x_1 - x_2\|_2 = \sqrt{\sum (x_{1i} - x_{2i})^2} $$

or in its squared form, known as a *Sum of Squared Difference (SSD)* among Computer Vision scientists:

$$ SSD(x_1, x_2) = \|x_1 - x_2\|_2^2 = \sum (x_{1i} - x_{2i})^2 $$

It’s most well known application in the signal processing field is the *Mean-Squared Error (MSE)* measurement, which is used to compute a similarity, a quality, or a correlation between two signals. MSE is

$$ MSE(x_1, x_2) = \frac{1}{n} \|x_1 - x_2\|_2^2 = \frac{1}{n} \sum (x_{1i} - x_{2i})^2 $$

As previously discussed in $l_0$-optimisation section, because of many issues from both a computational view and a mathematical view, many $l_0$-optimisation problems relax themselves to become $l_1$- and $l_2$-optimisation instead. Because of this, we will now discuss about the optimisation of $l_2$-optimisation.

### $l_2$-optimisation

As in $l_0$-optimisation case, the problem of minimising $l_2$-norm is formulated by

$$ \min \|x\|_2 \text{ subject to } Ax = b $$

Assume that the constraint matrix $A$ has full rank, this problem is now a underdetermined system which has infinite solutions. The goal in this case is to draw out the best solution, i.e. has lowest $l_2$-norm, from these infinitely many solutions. This could be a very tedious work if it was to be computed directly. Luckily it is a mathematical trick that can help us a lot in this work.

By using a trick of Lagrange multipliers, we can then define a Lagrangian

$$ \mathcal{L}(x) = \|x\|_2^2 + \lambda^T (Ax - b) $$

where $\lambda$ is the introduced Lagrange multipliers. Take derivative of this equation equal to zero to find an optimal solution and get

$$ \dot{x}_{opt} = -\frac{1}{2} A^T \lambda $$

plug this solution into the constraint to get

$$ A \dot{x}_{opt} = -\frac{1}{2} AA^T \lambda = b $$

$$ \lambda = -2 (AA^T)^{-1} b $$

and finally
\[ x_{opt} = A^T(AA^T)^{-1}b = A^+b \]

By using this equation, we can now instantly compute an optimal solution of the \( l_2 \)-optimisation problem. This equation is well known as the Moore-Penrose Pseudoinverse and the problem itself is usually known as Least Square problem, Least Square regression, or Least Square optimisation.

However, even though the solution of Least Square method is easy to compute, it’s not necessary be the best solution. Because of the smooth nature of \( l_2 \)-norm itself, it is hard to find a single, best solution for the problem.

![l2.png](https://rorasa.files.wordpress.com/2012/05/l2.png)

In contrary, the \( l_2 \)-optimisation can provide much better result than this solution.

**\( l_1 \)-optimisation**

As usual, the \( l_1 \)-minimisation problem is formulated as

\[
\min \|x\|_1 \text{ subject to } Ax = b
\]

Because the nature of \( l_1 \)-norm is not smooth as in the \( l_2 \)-norm case, the solution of this problem is much better and more unique than the \( l_2 \)-optimisation.

![l1.png](https://rorasa.files.wordpress.com/2012/05/l1.png)

However, even though the problem of \( l_1 \)-minimisation has almost the same form as the \( l_2 \)-minimisation, it’s much harder to solve. Because this problem doesn’t have a smooth function, the trick we used to solve \( l_2 \)-problem is no longer valid. The only way left to find its solution is to search for it directly. Searching for the solution means that we have to compute every single possible solution to find the best one from the pool of “infinitely many” possible solutions.

Since there is no easy way to find the solution for this problem mathematically, the usefulness of \( l_1 \)-optimisation is very limited for decades. Until recently, the advancement of computer with high computational power allows us to “sweep” through all the solutions. By using many helpful algorithms, namely the Convex Optimisation algorithm such as linear programming, or non-linear programming, etc. it’s now possible to find the best solution to this question. Many applications that rely on \( l_1 \)-optimisation, including the Compressive Sensing, are now possible.

There are many toolboxes for \( l_1 \)-optimisation available nowadays. These toolboxes usually use different approaches and/or algorithms to solve the same question. The example of these toolboxes are l1-magic (http://users.ece.gatech.edu/~justin/l1magic/), SparseLab (http://sparselab.stanford.edu/), ISAL1 (http://imo.rz.hu-bs.de/mo/spear/).

Now that we have discussed many members of norm family, starting from \( l_1 \)-norm, \( l_2 \)-norm, and \( l_\infty \)-norm. It’s time to move on to the next one. As we discussed in the very beginning that there can be any \( l \)-whatever norm following the same basic definition of norm, it’s going to take a lot of time to talk about all of them. Fortunately, apart from \( l_1 \), \( l_2 \), and \( l_\infty \)-norm, the rest of them usually uncommon and therefore don’t have so many interesting