Symbol Recognition.

Suppose we have an array of bits:

01011001010010001

And our goal is to teach the machine to recognize some pattern, for example this one:

0101

Given that our set of bits has only finite number of rearrangements that is **pow(2,size\_of\_array)** we can say that we also have a finite set of arrays where we can face such a pattern.

So that means that a BMP image of only black and white pixels has only limited set of rearrangements, such that we can face the symbol we want to recognize.

Then our goal is to supply the program with the training data, so that it will be able to recognize similar patterns based on this data.

We can say that data we provided is 100% is the case of needed symbol but we want put multiple cases and designate them with one name for example 0 or 1.

It is resembling polynomials where we can get one result e.g., 0 from different parameters (images with our symbol).

So, the more data we have the bigger degree of the polynomial. You can consider a function this way:

**(x-learning\_data1)\* (x-learning\_data2)\* (x-learning\_data3)\* (x-learning\_data4)…**

But how can we represent an image as single number? We can do it in a similar way as binary numbers do.

We can say that out first set of bits **01011001010010001** can be represented as 45713.

We can differently identify every single bitmap image, but numbers we will get will be horribly big.

For example, black square of 10x10 will be pow (2,100) = 1,267,650,600,228,229,401,496,703,205,376

But what if we want to use bigger images?

Then we can create out own integers of every size we want. We just need to create an array of bools with corresponding arithmetic operations.

Our polynomials use subtraction and multiplication, under which integers are closed, so we don't need a floating point at all.