Supervised self-initiative work

Stochastic gradient descent

Benjamin LOISON (MPSI 1)

2018-2019

1 Intuitive introduction

slide

The **objective** of my TIPE is to obtain an algorithm capable, when provided with an image containing fish, of counting them and identifying their species. This artificial intelligence would be based on a **neural network**. An intermediate tool is the **gradient descent**
The algorithm of the stochastic gradient is a **training of the parameters** on a database. After training we can use these parameters to make a **prediction**. I will first describe how we proceed with the training.

slide

Let's start by making an **intuition** of what gradient descent is. In a pictorial way, by considering the axes (Ox) and (Oy) and a curve representing a function, the method of gradient descent consists then in **calculating the minimum of this function**.
The technique of the gradient descent consists by its initial conditions fixed by ourselves, to place a **ball undergoing in a certain way the gravity on the curve**, for example in M_0.
We work here for example with the polynomial function: $x \longrightarrow x^4 - 3x^3$. We also define in the initial conditions, the norm of the **speed** of the ball what we will call later a **learning rate** α. The gradient algorithm then consists of the **iteration** of the falling process, so we obtain the points M_1 and M_2.
We can easily predict and see that the point M_3 will not be closer than M_2 to the minimum we are seeking.

slide

The point M_4 is again on the right branch of the curve because the gradient descent is "going down the curve", i.e. going in the direction where the gradient vector is smallest. Knowing that the gradient vector is infinitesimally the smallest vector in norm allowing to maximize the growth of the curve.
The sequence of points starting from M_3 does not seem interesting because it does not seem to converge towards the minimum.

This **problem** comes from the learning rate that we would like to decrease little by little by approaching the minimum, the problem is that the **we don't know where the minimum is**.

slide

One technique is then to choose a learning rate that for bad predictions will tend to decrease by successively taking the values of a strictly decreasing sequence, such as: $\left(\frac{1}{n+1}\right)_{n\in\mathbb{N}}$

slide

2 Principle of the algorithm

I first tried to train on the standard database **MNIST** before considering a real case with pictures of fishes even if I will show that comparing pictures of numbers or fishes in automatic learning is almost the same thing. Indeed this database is made of **images** of 28 by 28 pixels **hand written numbers** in black on white with a range of white and black colors from 0 to 1 in steps of 0.01 knowing that 0 is white and 1 is black.

So I first tried to make a **binary classification** algorithm, i.e. an algorithm capable of saying whether or not a given image corresponds to a 6 for example. The **output of the algorithm being in practice a real** between 0 and 1. **The closer the real is to 1** and the more we are sure that the considered image corresponds in my example to a 6.
After coding this binary classification, it is easy to **do 10 times the previous algorithm** of binary classification in order to successively know if the input image is a 0 or not, then if it is a 1 or not and so on. We can then predict in relation to an image what is the number written by taking the number associated with an output closest to 1.

slide

The gradient descent can be interpreted as the **convergence of the line passing at the most points in space**, where the space is here of dimension 28^2 and the coordinates of the points successively represent the values between 0 and 1 of each pixel.
We work on a given **digit**, noted n.
- Remember that α is the learning rate and it **may depend on the iteration considered**.
- We put m the number of photos studied knowing that the **MNIST training database contains 60,000 photos with the associated number**.
Let $j \in [\![1; 28^2]\!]$,θ_j **is the j-th parameter** defining the line passing closest to the points.
$h_\theta(x^{(i)})$ is the **prediction** of our algorithm for the i-th picture $x^{(i)}$.
$y^{(i)}$ is **0 or 1**, 1 if the photo corresponds to a n and 0 otherwise.
We have $h_\theta(x^{(i)}) = \sigma(\theta_1 + \theta_2 x_2 + ... + \theta_{28^2} x_{28^2})$, with σ the bijective **sigmoid** function from $\mathbb{R}$ to [0; 1], defined by: $\sigma(x) = \frac{1}{1+e^{-x}}$
- We clearly have $\sigma(0) = 0.5$ and $\sigma(x) \geq 0.5$ if $x \in \mathbb{R}^+$ and $\sigma(x) \leq 0.5$ if $x \in \mathbb{R}^-$
The gradient algorithm consists in **minimizing the cost function J using the method of least squares** that is the error of our algorithm, with $J(\theta_1, \theta_2, ..., \theta_{28^2}) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$.

slide

Read derivation if have enough time...

slide

3 Algorithm, results and critique

Briefly explain the code if have enough time.

slide

18.3 % of error may seem to be important, but remember not to compare with an intuitive **esperation** of 50 % when answering randomly. Indeed the real hope by answering in a random way is 10%. In other words, my algorithm is almost 8 times more efficient than an algorithm that answers randomly.
We can therefore speak of artificial intelligence and more precisely of supervised machine learning algorithms.
Moreover the important information regarding this 18.3% error is that this error rate is reached for a **test database**, i.e. on data that the algorithm has never encountered before, each entry can then be considered as a question that will be asked to the algorithm and it answers about 4 times out of 5 the right number.

Considered a line in a 28^2 dimension space may first seem like a good idea to approach a relationship to determine which number is written on the picture but the affine model may not go through all the points even with the best θ parameters. For example, a **polynomial model** may be more appropriate.
One can also notice that I did not use **adaptive learning rate** which can improve the efficiency of my algorithm.

In the algorithm we scan the θ and then for each θ we scan the images. One can wonder if by **reversing the order of the loops** for this would not improve or if it is equivalent.

4 Time optimization

slide

With the **numpy** library in Python, we could have computed h_θ with a product of vectors.

slide

In **C++ with threads**, this lower level language equipped with parallel calculation for the 10 digits separately, we go from an execution time on the **training database of 2 hours in Python without threads to 60 seconds in C++ with threads**! We could have considered using the power of a **graphics card** to perform all these small calculations with a language such as OpenCL.

5 Conclusion

Now that the gradient descent is familiar we can look at the **initial fish problem**. It can be reduced to a gradient descent provided that **enumerate and isolate graphically** with for example the OpenCV library the areas containing fishes and apply the gradient descent on these images. Indeed the gradient descent does not in itself make **no distinction between pictures of fishes or pictures of numbers** since one can assign a number to each species of fish. It would then be necessary to consider **arrays of three times the size** in order to consider the colors red, green and blue. It would also be necessary to **redimension the images** with a suitable interpolation algorithm the photos of different sizes of fish.