Comparison of the advantages and disadvantages of common optimization algorithms

20195732 李御霖

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.01 | 0.001 | 0.0001 | 0.00001 |
| sgd | 0.8687 | 0.1135 | 0.1135 | 0.1135 | 0.1135 |
| sgd+momentum | 0.9868 | 0.2406 | 0.1135 | 0.1135 | 0.1135 |
| adam | 0.0982 | 0.988 | 0.987 | 0.9256 | 0.2295 |

Look at the conclusion first, the data in the table is the prediction accuracy of the test data set, and the momentum is 0.9

background:

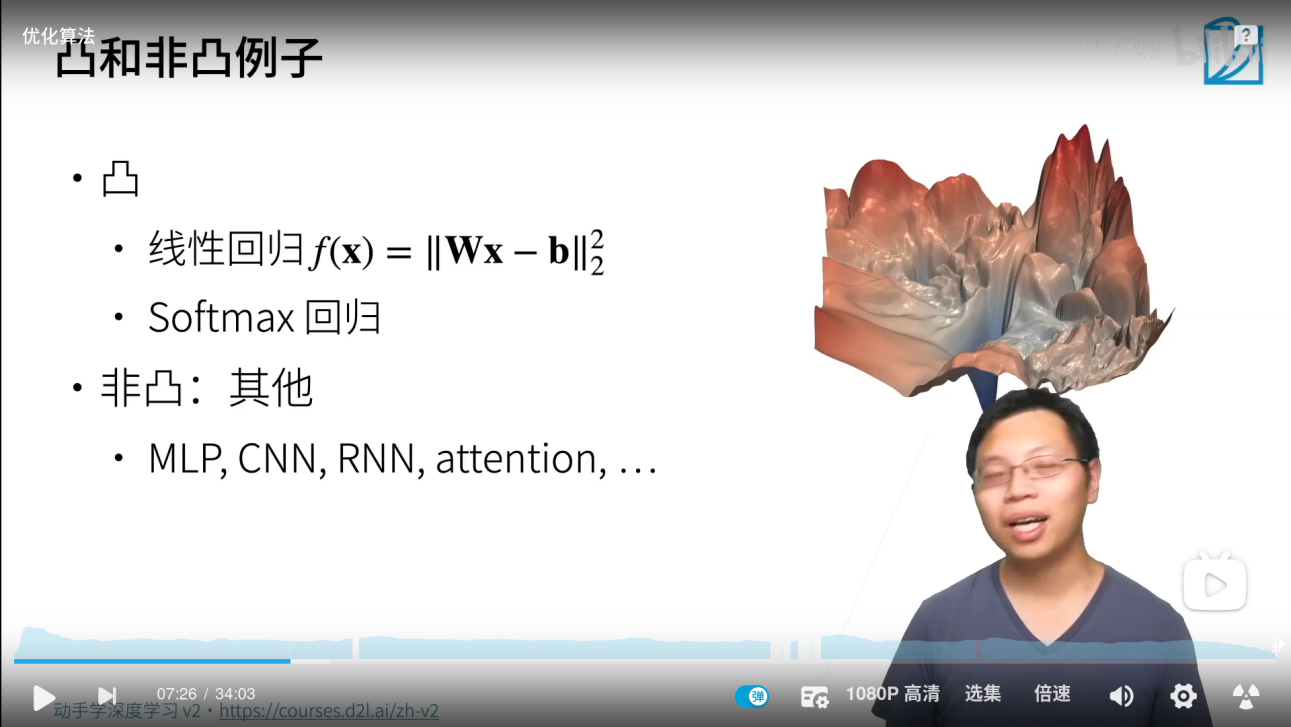
Optimization problem: Find the minimum value of the function f(x), where x∈C (C is a prescribed set, for example, all x needs to be greater than 0).

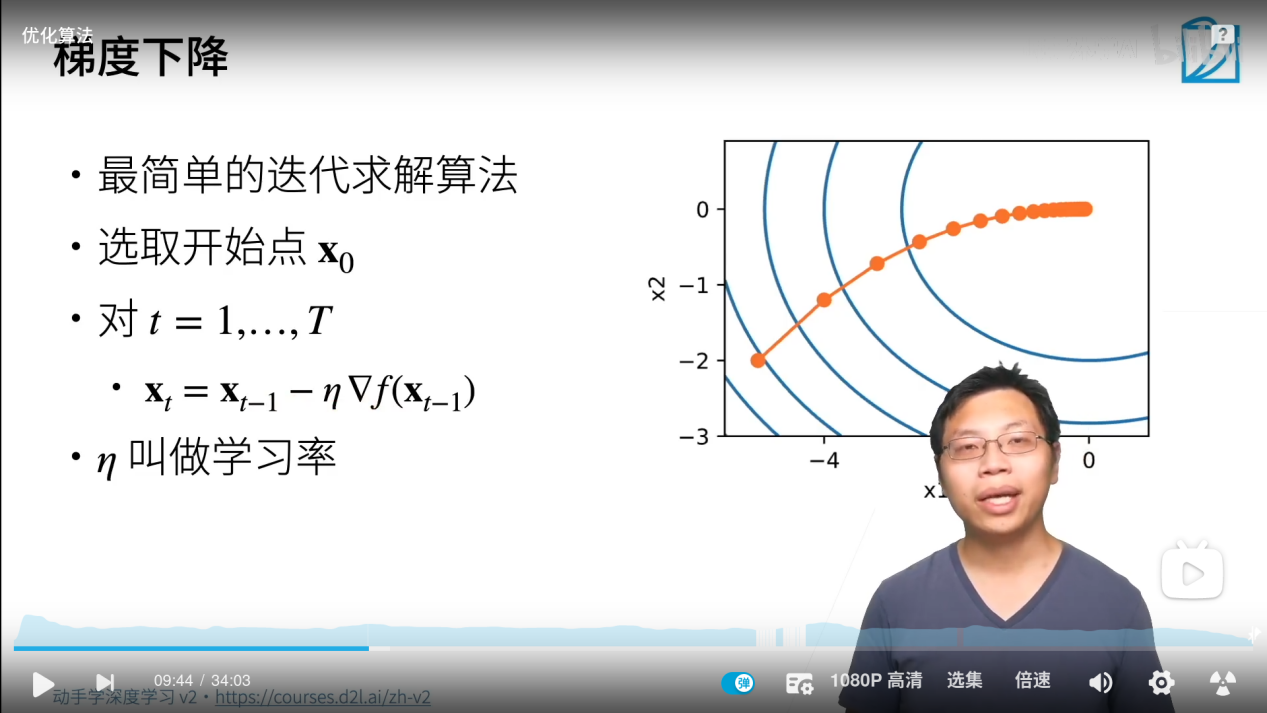
Convex function: The function f is convex if and only if: f(ax+(1-a)y)<=af(x)+(1-a)f(y), a belongs to (0,1), x!= y, and is strictly convex at this time. To put it bluntly, it is the connection between any two points (x, y) on the function, and the function value between x and y is less than f(x), f(y)

At this time, there is often only one maximum point.

What is the ‘‘f(x)’’ in the neural network: it is actually the entire forward inference process, namely loss(net(x)), where x is the input value. The task of optimization is to minimize the loss function. So as to achieve the purpose of training.

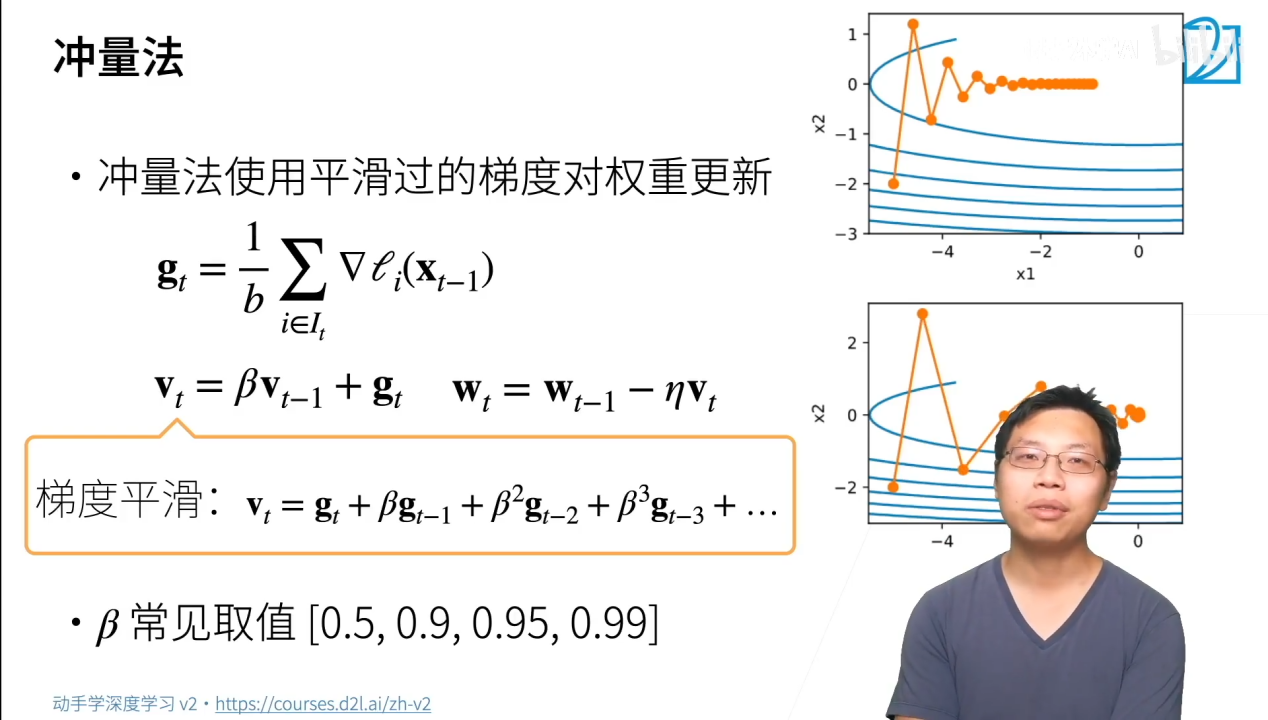
Unfortunately, most of the networks in neural networks are non-convex. There are only two convex functions that we often touch: linear regression and sotfmax regression. Other multi-layer perceptrons, convolutions, etc. are already non-convex due to various activation functions added to the function. It can be imagined as a bumpy valley, and our task is to descend the mountain.

(Visualization of complex functions)

**Algorithm principle:SGD：**

Simply put, after each calculation, the average gradient is recorded and then backpropagated. Simple and rude.

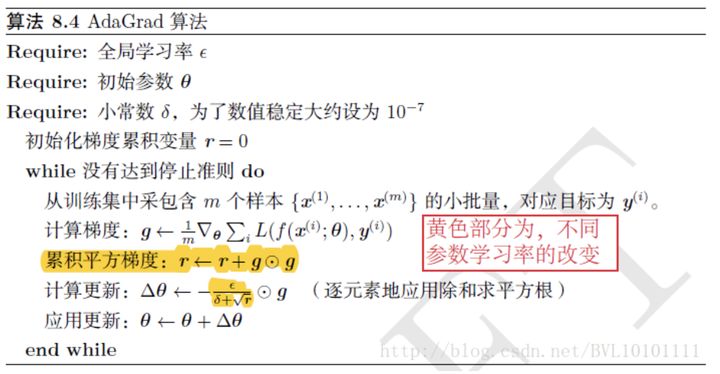
**Momentum：**



Simply put, it uses historical gradient information to ensure that the gradient change rate will not be too large, reducing the possibility of the point staying in the local optimal solution.

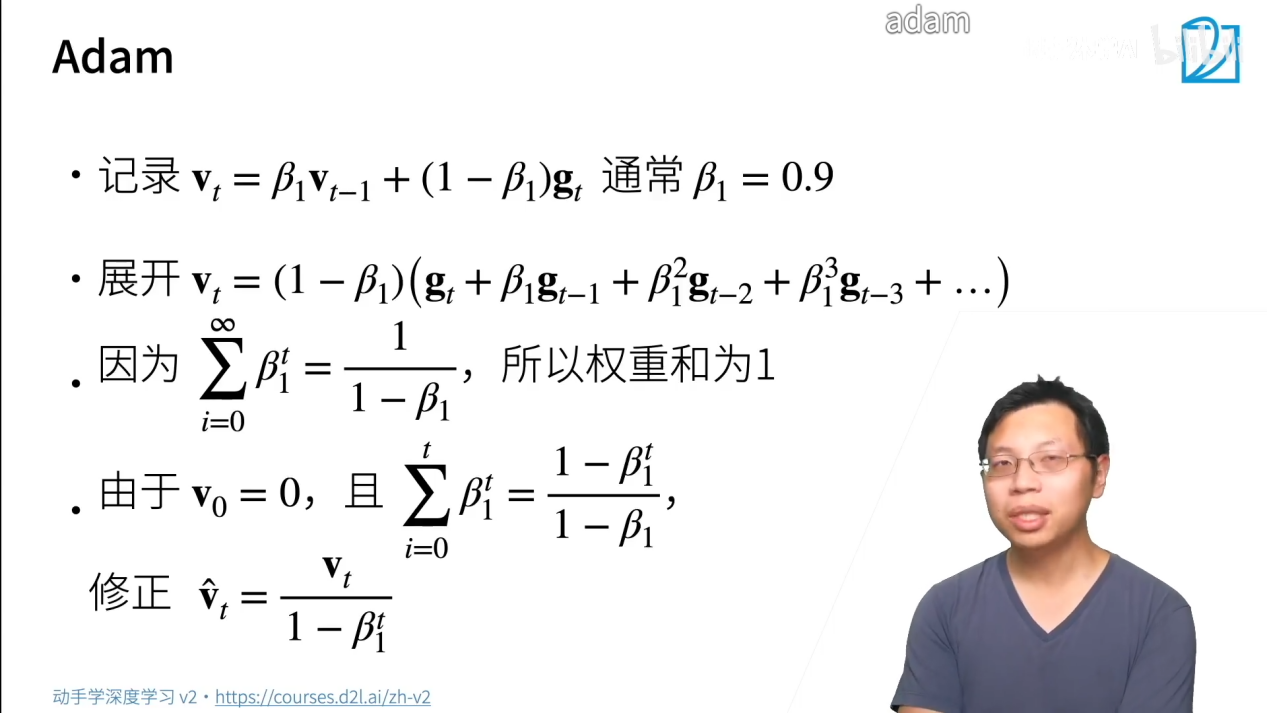
The object of filtering here is the gradient.

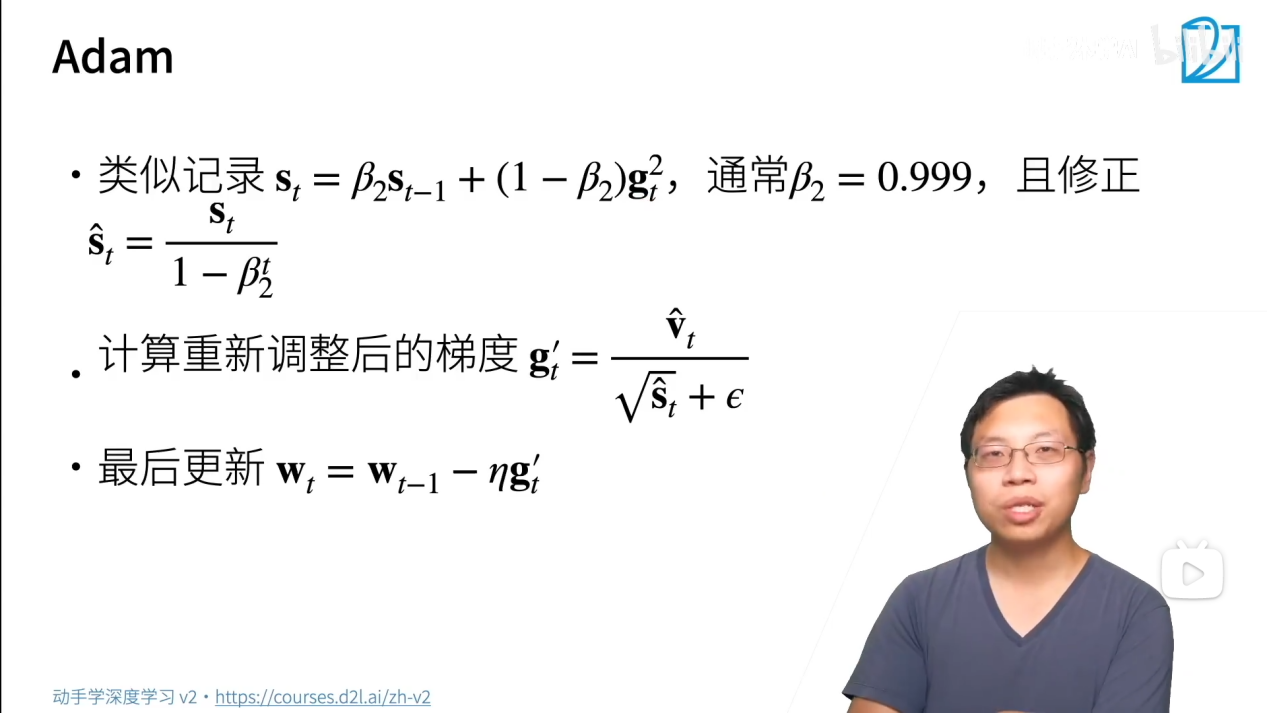
AdaGrade：



Simply put, after setting the global learning rate, for each pass, the global learning rate is divided by the square root of the square sum of the historical gradients parameter by parameter, so that the learning rate of each parameter is different. However, because the gradient has been accumulated here, the final change rate will often be close to 0, so it is not recommended to use it in practice.

**Adam:**

****

****

It can be seen from the complete formula that in the final expression, v hat t is part of the momentum method, and the harmonic average is used to record and smooth the historical gradient. In the lower half, the s is to record the square of each element, and then smooth it. The complex formulas are also incomprehensible. The conclusion is that the parameters and gradients are smoothed at the same time, and they are not sensitive to the learning rate. In other words, you can adjust the parameters less.

**实际经验和优劣：**

**SGD：**

Advantages: Although it looks rough, the core question is, will the piles of smoothing make the function miss a better solution? Empirically, it is true that only meticulous adjustments and sgd can make the accuracy increase continuously, so the momentum method or adam is often used to iterate the cosine learning rate first. When you can’t continue to improve, go to sgd to adjust the parameters. Maybe you get the optimal solution if you are lucky?

Disadvantages: does not converge, and it is easy to get stuck in the local optimum. If you only use the sgd method. It is easy to encounter non-convergence, explosion or disappearance of the gradient, and the most local solution of the card. It is often necessary to change the learning rate and recalculate after saving the parameters, which is very troublesome

Momentum：

Advantages: easy to call, often bound with sgd, adding a few parameters can get good results.

Disadvantages: The parameters are not smoothed, and the same problems as sgd are still prone to occur. (For example, it gets stuck when it is locally optimal)

AdaGrad：

Because there is no experience in using it, it can only be guessed based on the formula.

Advantages: It should be similar to adam and converge quickly

Disadvantages: But if you iterate for too long, you will definitely not find a good solution, and loss will fluctuate in a bunch of values

Adam:

Advantages: fast convergence, less need to adjust parameters. It's so cool to use in practice. A good solution can always be found.

Disadvantages: Because it is not sensitive to the learning rate, you will actually find that Adam will eventually float in a range. At this time, he still has to rely on sgd to stabilize him.

Look at the actual experiment again：

The experimental table is as follows:

Here, the test data set after 20 epochs is used as the accuracy:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0.1 | 0.01 | 0.001 | 0.0001 | 0.00001 |
| sgd | 0.8687 | 0.1135 | 0.1135 | 0.1135 | 0.1135 |
| sgd+momentum | 0.9868 | 0.2406 | 0.1135 | 0.1135 | 0.1135 |
| adam | 0.0982 | 0.988 | 0.987 | 0.9256 | 0.2295 |

The experimental conclusion is consistent with the previous theoretical analysis. The only problem is that Adam does not converge at 0.1, or the result is strange?

Due to time constraints, I personally think that the learning rate is too large, the value keeps changing close to random, and the accuracy rate is close to 0.1.