**5 algorithms to train a neural network**

<https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network>

The procedure used to carry out the learning process in a [neural network](https://www.neuraldesigner.com/learning/tutorials/neural-network) is called the[optimization algorithm](https://www.neuraldesigner.com/learning/tutorials/training-strategy#OptimizationAlgorithm)(or optimizer).

There are many different optimization algorithms. All have different characteristics and performance in terms of memory requirements, processing speed, and numerical precision.

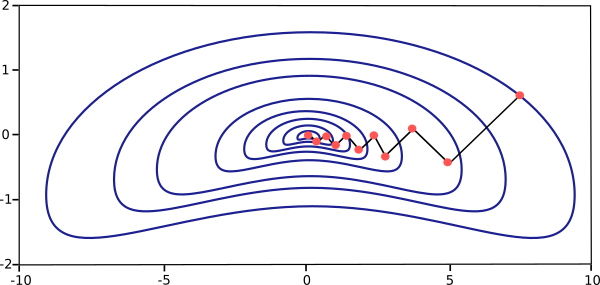
In this post, we formulate the learning problem for neural networks. Then, some important [optimization algorithms](https://www.neuraldesigner.com/learning/tutorials/training-strategy#OptimizationAlgorithm) are described. Finally, the memory, speed, and precision of those algorithms are compared.

* [Learning problem](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#LearningProblem).
* [1. Gradient descent](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#GradientDescent).
* [2. Newton method](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#NewtonMethod).
* [3. Conjugate gradient](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#ConjugateGradient).
* [4. Quasi-Newton method](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#Quasi-Newton).
* [5. Levenberg-Marquardt algorithm](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#Levenberg-Marquardt).
* [Performance comparison](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#PerformanceComparison).
* [Conclusions](https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network#Conclusions).

## 1. Gradient descent

[Gradient descent](https://www.neuraldesigner.com/learning/tutorials/training-strategy#GradientDescent), also known as steepest descent, is the most straightforward training algorithm. It requires information from the gradient vector, and hence it is a first-order method.

The [gradient descent](https://www.neuraldesigner.com/learning/tutorials/training-strategy#GradientDescent) training algorithm has the severe drawback of requiring many iterations for functions which have long, narrow valley structures. Indeed, the downhill gradient is the direction in which the [loss](https://www.neuraldesigner.com/learning/tutorials/training-strategy#LossIndex) function decreases the most rapidly, but this does not necessarily produce the fastest convergence. The following picture illustrates this issue.



[Gradient descent](https://www.neuraldesigner.com/learning/tutorials/training-strategy#GradientDescent) is the recommended algorithm when we have massive neural networks, with many thousand parameters. The reason is that this method only stores the gradient vector (size nn).