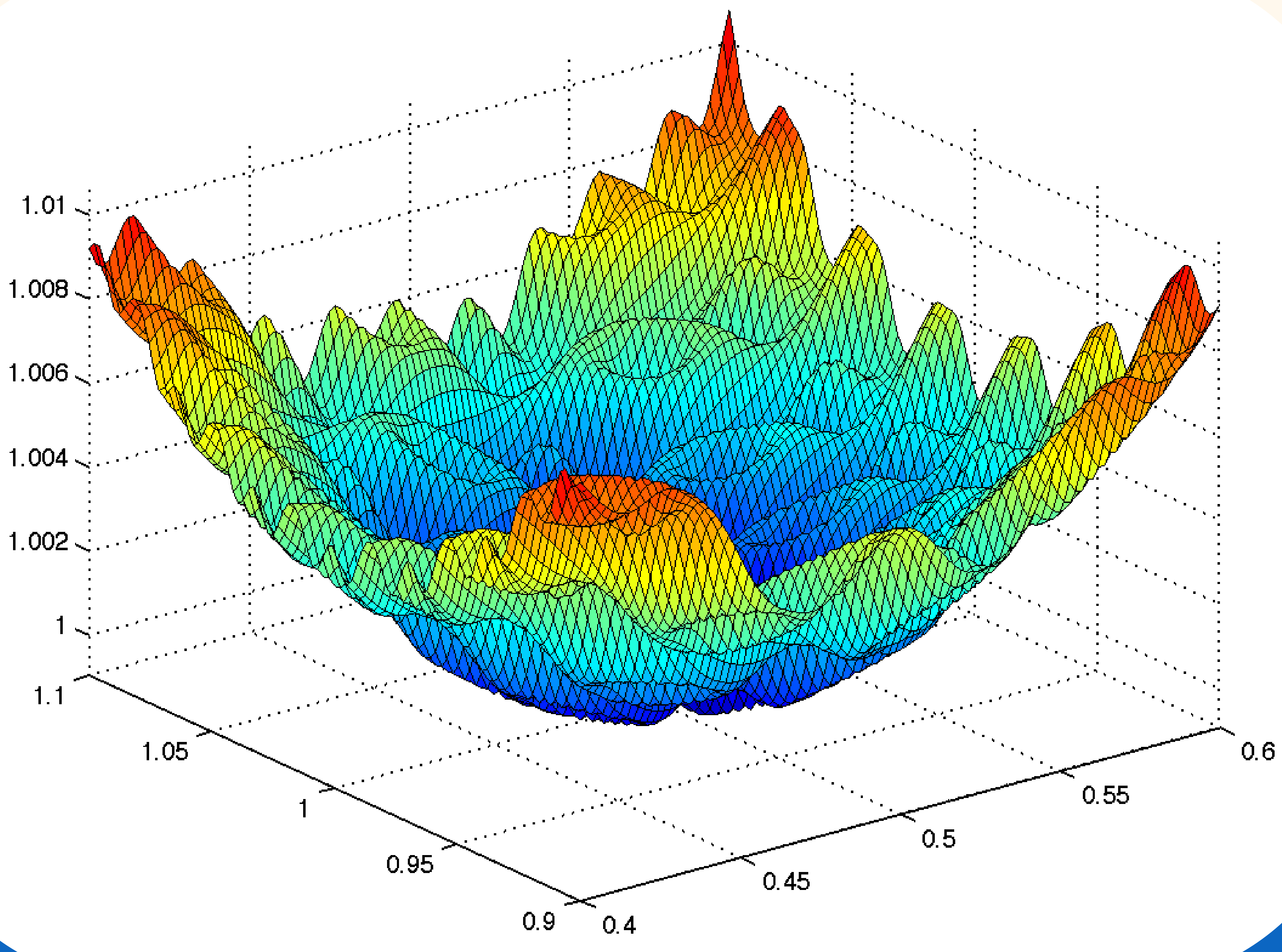
9 Most Popular Optimization Algorithms in Deep Learning

✓Pros, ✗Cons, 💡Use Cases



1

Stochastic Gradient Descent (SGD):

- **Pros:** It is simple to implement and computationally efficient. It can be used for a wide range of problems, including linear regression and neural networks.
- **Cons:** It can be sensitive to the choice of learning rate and may require manual tuning. It can also get stuck in local minima.
- **Use case:** It is commonly used in large-scale machine learning problems, where the dataset is too large to be processed in one pass.

2

Mini-batch Gradient Descent:

- **Pros:** It is more computationally efficient than SGD because it processes multiple examples at once. It can also be less sensitive to the choice of learning rate.
- **Cons:** It can still get stuck in local minima and may require manual tuning.
- **Use case:** Mini-batch gradient descent is a common optimization algorithm for training deep neural networks.

3

Nesterov Accelerated Gradient (NAG):

- **Pros:** It can help the optimization to converge faster and escape from local minima by providing a better approximation of the gradient.

- **Cons:** It can overshoot the global minimum if the momentum term is set too high.

- **Use case:** It is commonly used in deep learning, especially in training large neural networks.

4

Adagrad:

- **Pros:** It adapts the learning rate for each parameter, allowing for efficient optimization of non-uniformly scaled data.

- **Cons:** It can have a monotonically decreasing learning rate, which can cause the optimization to converge too slowly.

- **Use case:** It is particularly well-suited for problems with sparse data, such as natural language processing.

5

Adadelta:

- **Pros:** It adapts the learning rate based on the historical gradient information, which can be more robust to the choice of initial learning rate.
- **Cons:** It can be sensitive to the choice of the hyper-parameter.
- **Use case:** Itis a good choice for problems where the data is noisy or the optimization is prone to getting stuck in local minima.

6

RMSprop:

- **Pros:** It adapts the learning rate based on the historical gradient information, which can help the optimization converge more quickly.
- **Cons:** Like Adagrad and Adadelta, it can be sensitive to the choice of the hyper-parameter.
- **Use case:** It is a good choice for problems where the data is noisy or the optimization is prone to getting stuck in local minima.

7

Adam (Adaptive Moment Estimation):

- **Pros:** It combines the advantages of both momentum and RMSprop by adapting the learning rate for each parameter and building up a momentum in the direction of the steepest descent.
- **Cons:** It can be sensitive to the choice of the hyper-parameters.
- **Use case:** It is a popular optimization algorithm for training deep neural networks.

8

Adamax:

- **Pros:** It is similar to Adam but uses the infinity-norm of the gradient instead of the second-order moment, which can be more robust to the choice of the hyper-parameters.

- **Cons:** It can be sensitive to the choice of the hyper-parameters.

- **Use case:** It is particularly well-suited for problems where the data is sparse or the optimization is prone to getting stuck in local minima, similar to Adagrad.

9

Nadam (Nesterov-accelerated Adaptive Moment Estimation):

- **Pros:** It combines the advantages of NAG and Adam by providing a better approximation of the gradient and adapting the learning rate for each parameter.
- **Cons:** It can be sensitive to the choice of the hyper-parameters.
- **Use case:** It is a good choice for training deep neural networks, especially in cases where the optimization is prone to getting stuck in local minima.

DONT FORGET TO LIKE, SHARE AND SAVE IF YOU LIKE THIS POST

FOLLOW

Salman Khaliq

SALMANKHALIQ22

@DATASCHOLAR