CME 216, ME 343 - Spring 2020

Eric Darve, ICME

Stanford University

Let's talk about convergence and why it may slow down.

In gradient methods, there are basically two scenarios that could slow down convergence

1. Presence of local minima; the algorithm cannot distinguish between a local or a global minimum and may therefore get stuck near a local minimum.
2. Presence of saddle points.

A point where the gradient is zero is a critical point.

A saddle point is a critical point, but it is not a local extremum.

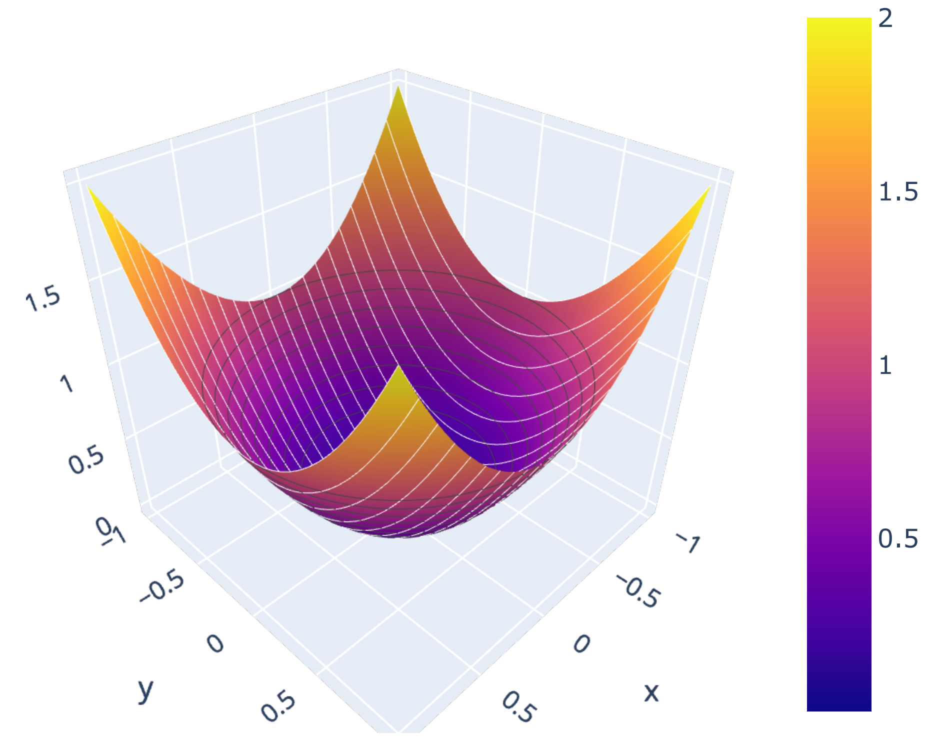
Mathematically, a critical point may also be a point where the function is not differentiable.

But we will assume that L is smooth everywhere for this discussion.

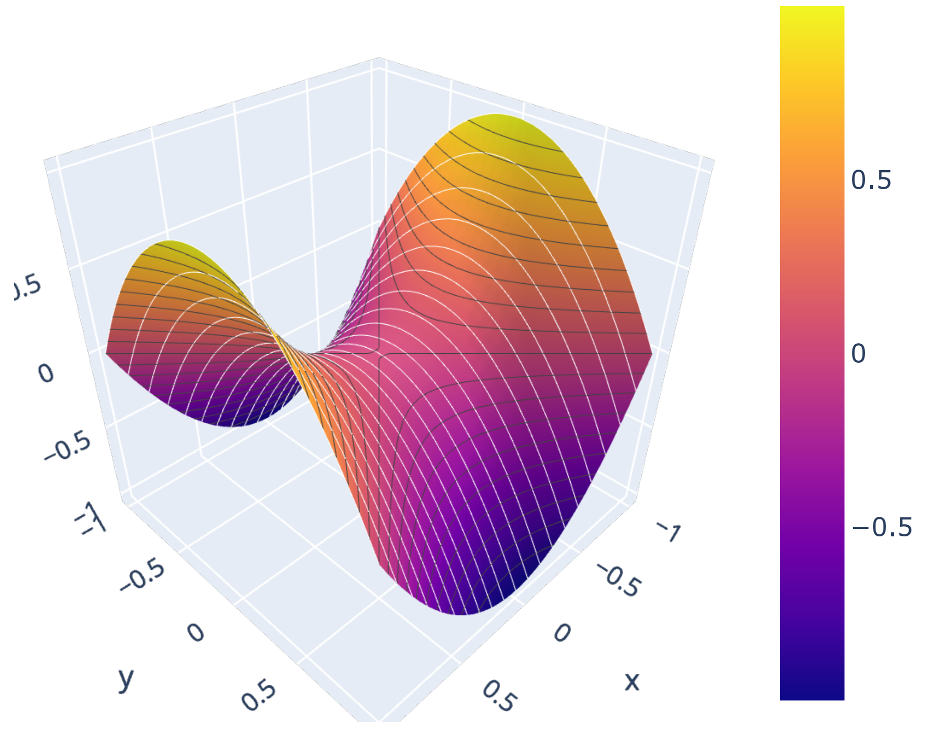
At a critical point, we can consider the Hessian of L and its eigenvalues.

- All the eigenvalues are positive: local minimum.
- All the eigenvalues are negative: local maximum.
- If some of the signs are positive and others negative: saddle point.

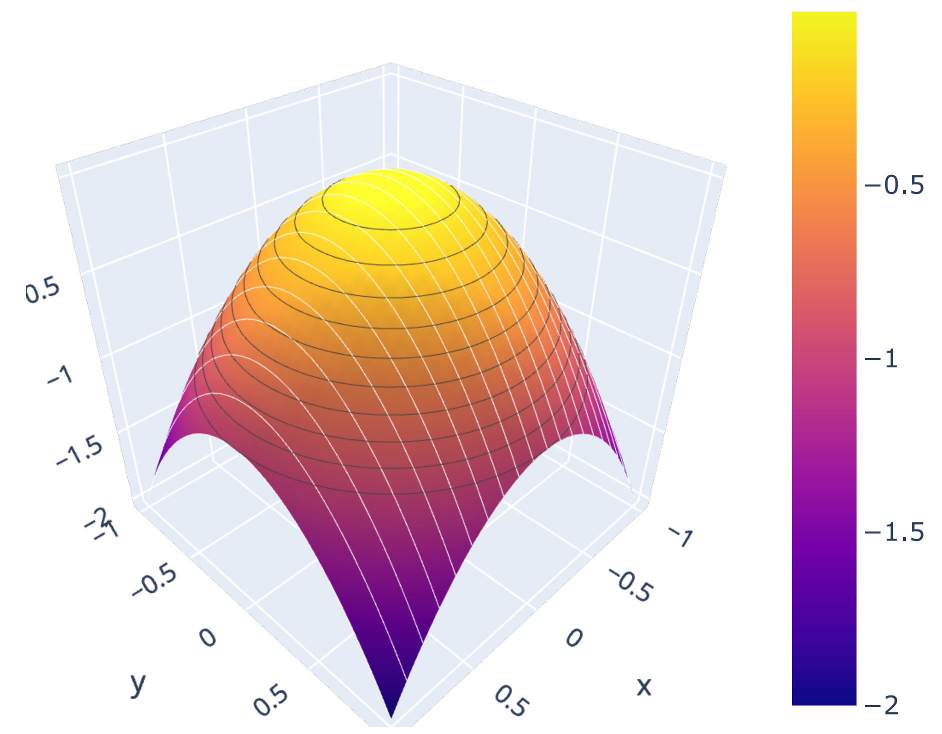
Minimum



Saddle point



Maximum



In general it is hard to say whether one may encounter more frequently local minima or saddle points.

However, in DNN, we have many weights $W_{ij}^{(l)}$ so L is a very high dimensional function.

Assume we have n weights to train.

The Hessian has n eigenvalues.

Let's assume for the sake of the argument that the sign of the eigenvalues are random.

The probability that all signs are positive is only $1/2^n$.

So in fact local minima are quite rare.

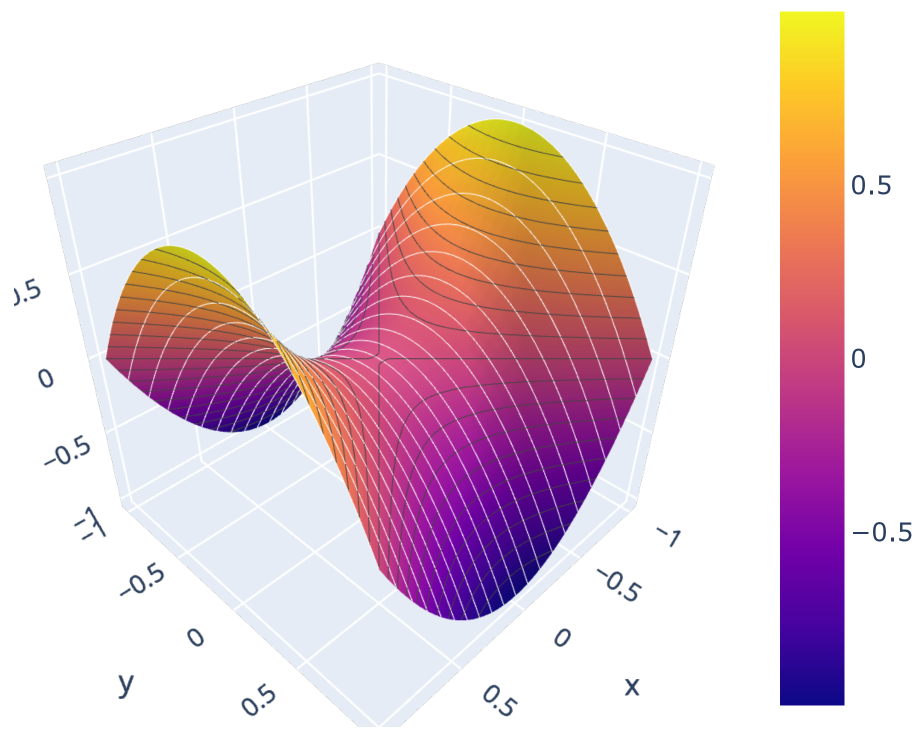
Of course, there is at least one, the global minimum.

But roughly we should be mostly concerned about saddle points.

In principle, in gradient based methods, saddle points are not really a problem.

Unless we approach the saddle point strictly along a line that follows the positive eigenvalue directions, we will eventually fall down away from the critical point.

Saddle point



However, in practice, this is very bad.

In particular, near the saddle point the gradient becomes very small.

This means that near a saddle point we tend to take very small steps and more very slowly.

This goes on until we are sufficiently far from the saddle point that convergence resumes.

This is probably one of the main reasons the loss converges slowly sometimes.

This is in addition to the problem of stiffness which leads to very small steps.

This is a distinct issue however. The problem is not that a small learning rate α is required.

The problem is that the gradient itself becomes small.

SGD is a good method in part because its stochastic component allows moving away from the saddle point and resume convergence in an area where the gradient is large again.

```python
def sgd(W, lr, batch_size):
    W -= lr * W.grad / batch_size
```