

# Optimization Algorithms

11.1: Optimization and Deep Learning

11.2: Convexity

Lily Whitler

February 28, 2022

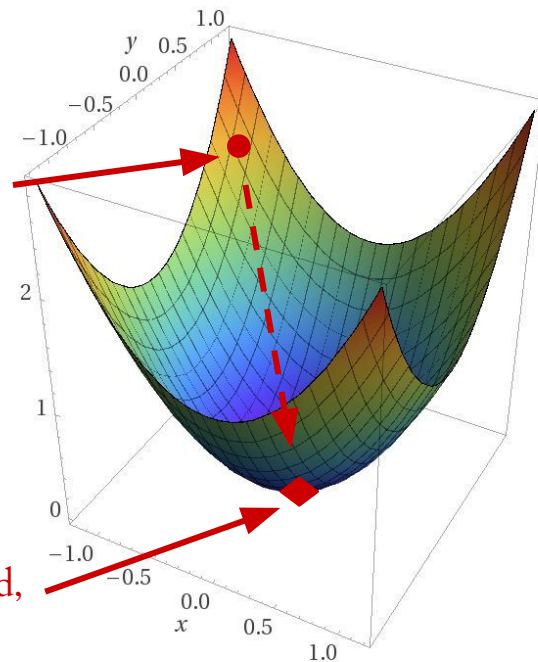
# Why care about optimization algorithms in deep learning?

“optimization algorithms are the tools that allow us to continue updating model parameters and to minimize the value of the loss function, *as evaluated on the training set*” (emphasis added)

- Model performance is quantified by the “objective function” (= loss function)
- Lower loss is better  $\Rightarrow$  need a way to find a place where the loss is low

Loss is large, not  
the best model

Loss is minimized,  
better model

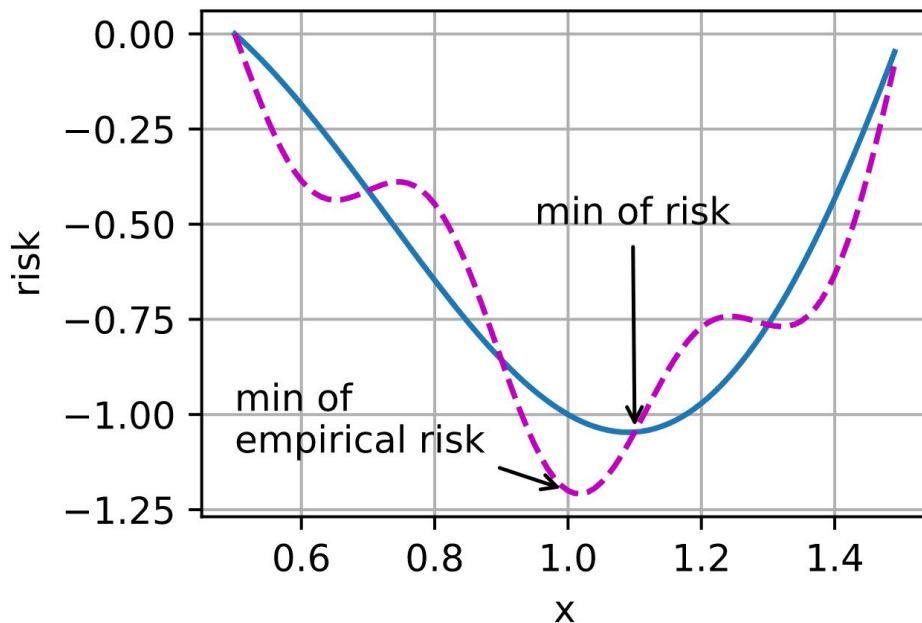


Loss function for a network with two  
parameters (credit: O'Reilly Media)

# Why care about optimization algorithms in deep learning?

“optimization algorithms are the tools that allow us to continue updating model parameters and to minimize the value of the loss function, *as evaluated on the training set*” (emphasis added)

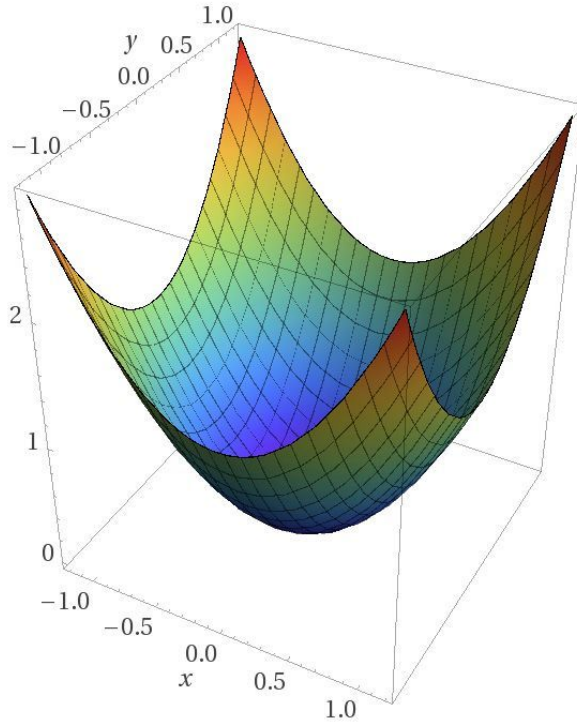
- Model performance is quantified by the “objective function” (= loss function)
- Lower loss is better  $\Rightarrow$  need a way to find a place where the loss is low
- **Note:** optimization  $\neq$  deep learning – goal of optimization is to minimize or maximize a function; goal of deep learning is to find the best model
  - Training vs. generalization error



# Loss functions are not usually very nice

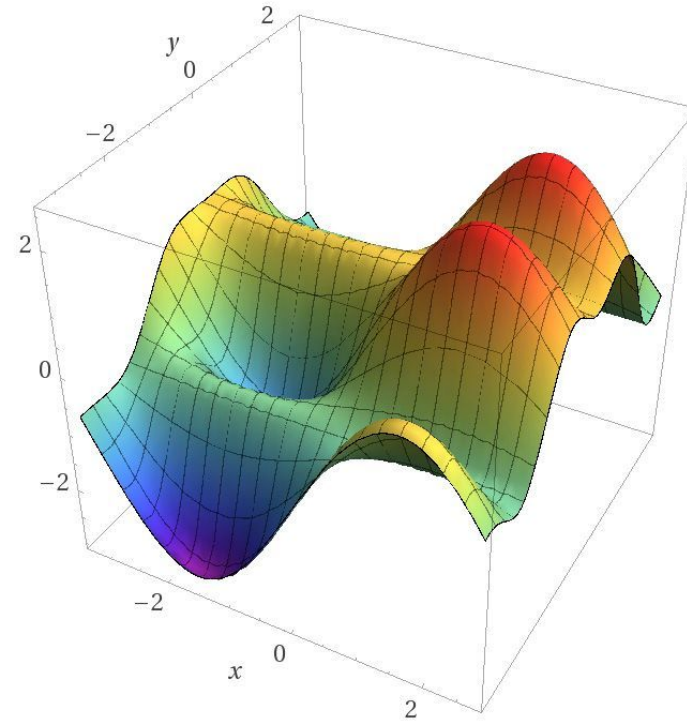
Loss functions for a network with two parameters (credit: O'Reilly Media)

convex function



Computed by Wolfram|Alpha

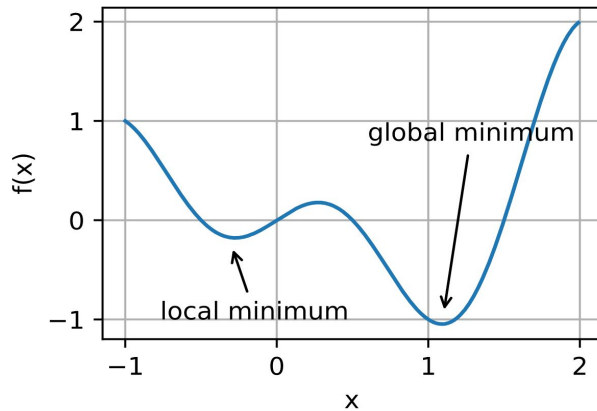
non-convex function



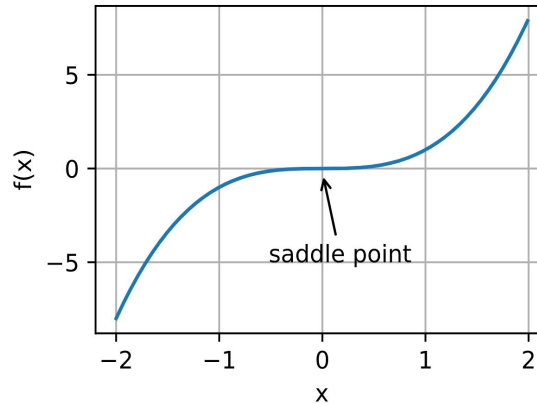
Computed by Wolfram|Alpha

Optimization algorithms slow down or stop entirely when the gradient is vanishingly small

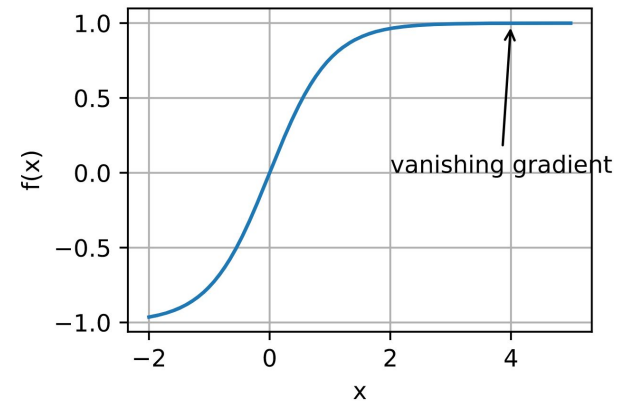
*local minima*



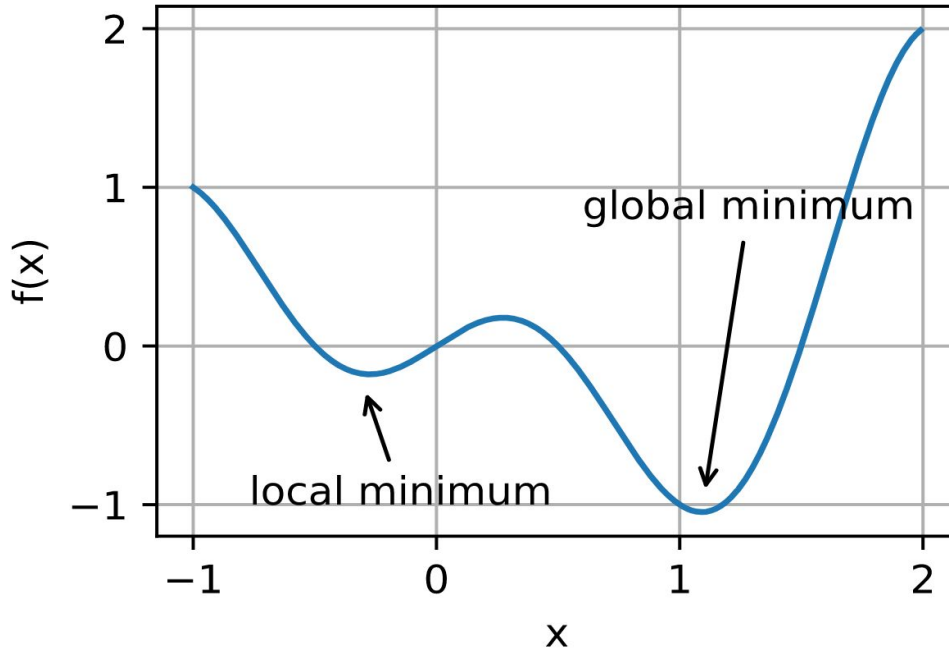
*saddle points*



*vanishing gradient*

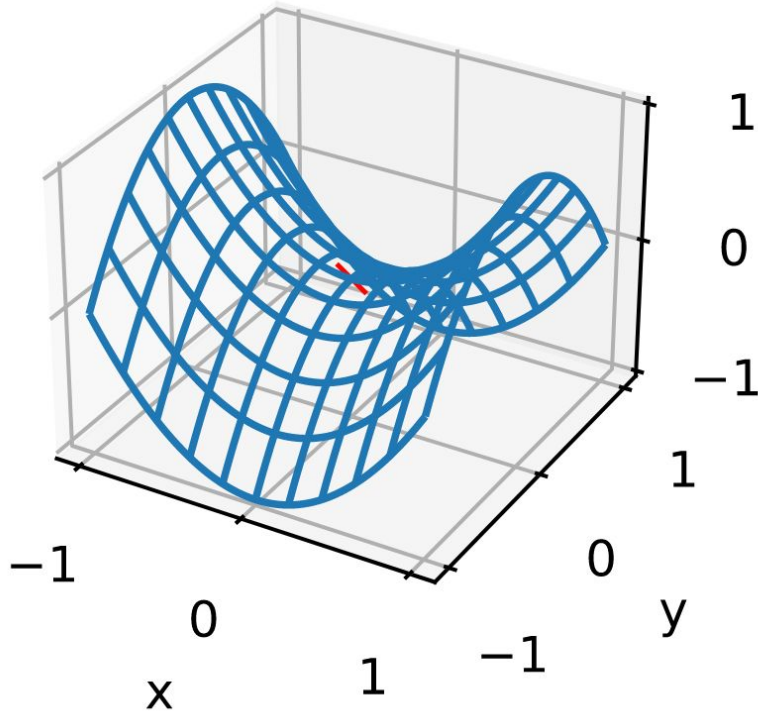


# Loss functions in deep learning usually have multiple minima



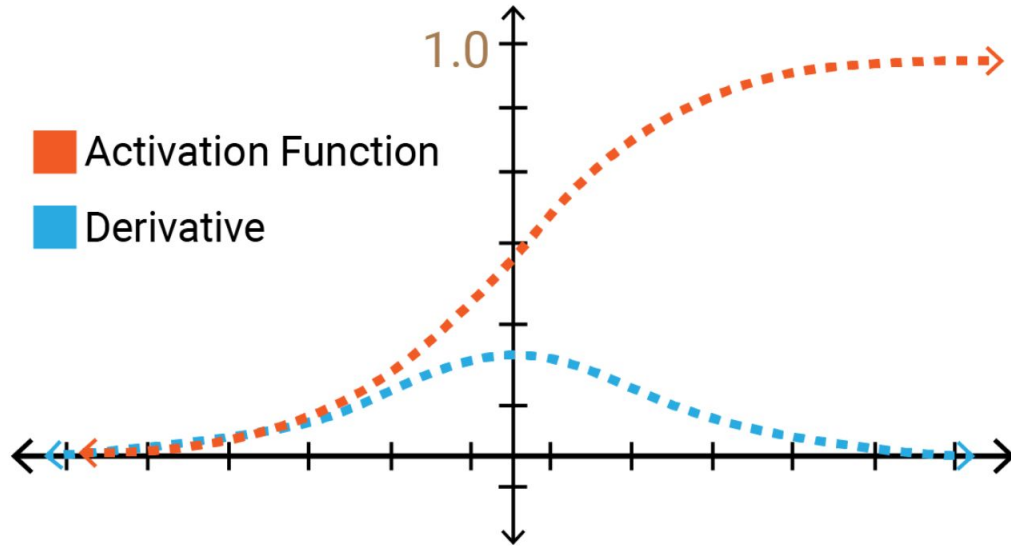
- The goal of deep learning is to find an acceptable model with low loss, not necessarily the best model with the lowest loss possible
- Stopping at a local minimum can be an acceptable outcome

# Saddle points are more common than local minima



- Hessian matrix: matrix of second derivatives (captures information about concavity)
- At a local minimum, all eigenvalues of the Hessian matrix must be positive
- At a saddle point, eigenvalues of the Hessian matrix are a mix of positive and negative
- In many-dimensional space, a mix of positive and negative eigenvalues is much more likely

# Vanishing gradients slow the training process down



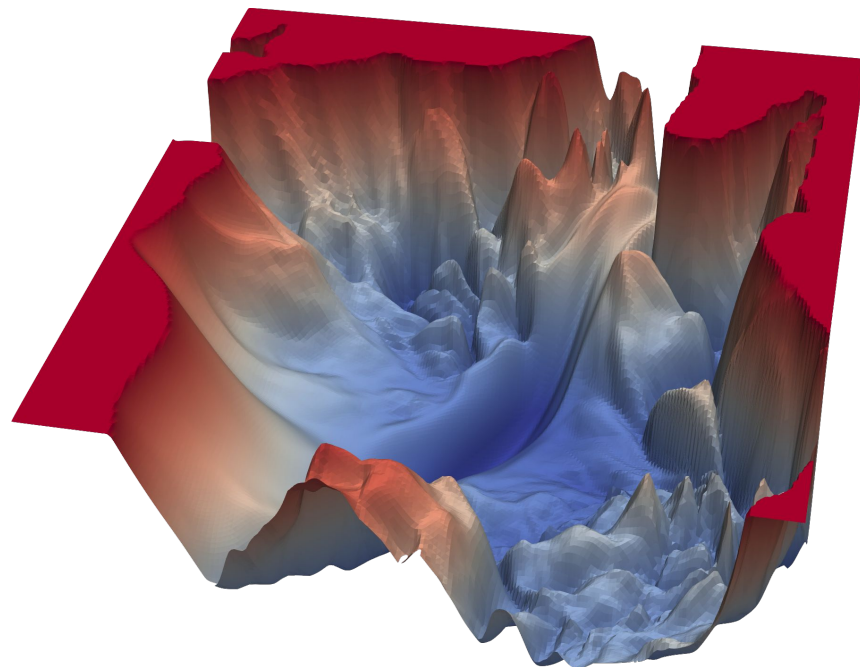
- The derivative of the activation function for each layer is used in back propagation
- In a deep network with many layers, we might be multiplying a *lot* of very small numbers together
  - Gradients can be driven to zero
- Little to no information about which direction to move parameters to decrease the loss  $\Rightarrow$  very inefficient training

<https://codeodysseys.com/posts/gradient-calculation/>

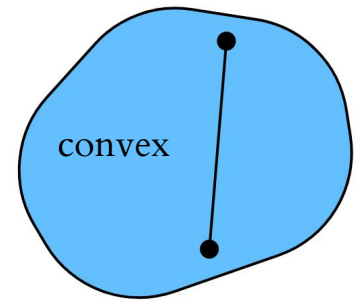
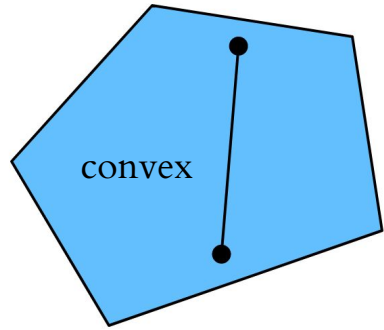
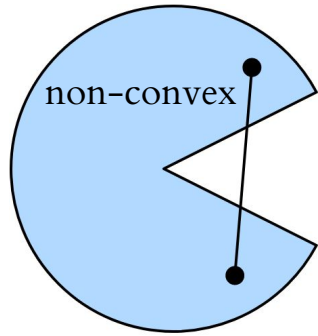


# How does convexity fit into this?

- Most deep learning optimization problems are not convex, but convex problems are still insightful
- If an optimization algorithm doesn't perform well in a convex situation, it definitely won't do well with a non-convex problem
- Non-convex loss functions are locally convex



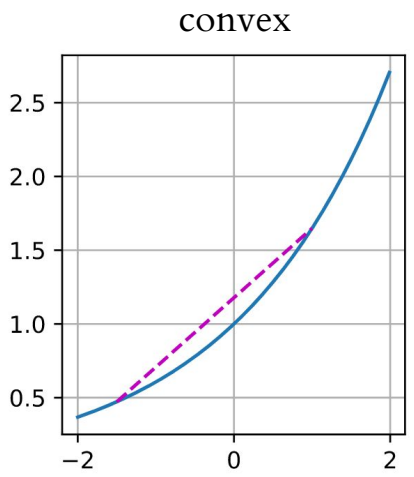
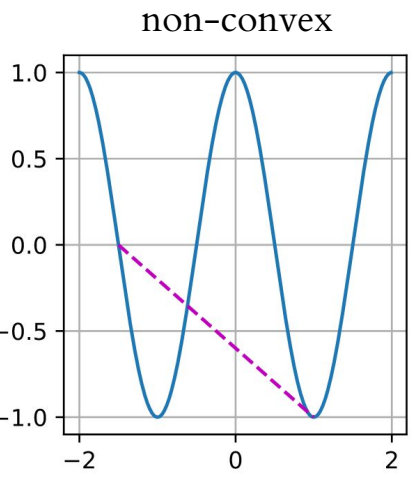
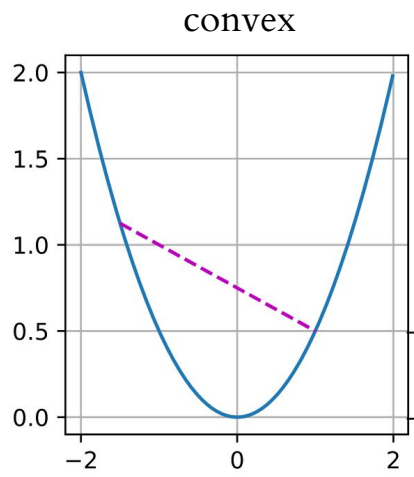
# Definitions: convex sets and convex functions



For any two points within the set, all points on the line connecting them are also fully within the set

$$\forall \lambda \in [0, 1], \lambda a + (1 - \lambda)b \in \mathcal{X} \text{ if } a, b \in \mathcal{X}$$

If the points on and above the function are a convex set, the function is convex

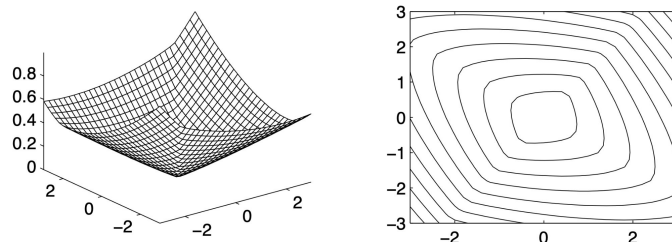


# Convex functions have some nice properties

The local minimum is the global minimum

**Note:** does not guarantee only one global minimum, or that it even exists

Below sets of convex functions are convex sets




**Figure 6.1** Left: Convex Function in two variables. Right: the corresponding convex level sets  $\{x | f(x) \leq c\}$ , for different values of  $c$ .

<http://agbs.kyb.tuebingen.mpg.de/lwk/>

A function is convex if and only if its Hessian matrix is positive semidefinite (has all non-negative eigenvalues)

And make constrained optimization problems easier

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} \ f(\mathbf{x}) \quad \leftarrow \text{convex function} \\ & \text{subject to } \mathbf{x} \in C \\ & C = \{\mathbf{x} \mid g_1(\mathbf{x}) \leq 0, \dots, g_n(\mathbf{x}) \leq 0, h_1(\mathbf{x}) = 0, h_m(\mathbf{x}) = 0\} \end{aligned}$$

  
convex sets                      affine functions  
linear transformation + translation

Convex constraints can be added via the Lagrangian, which makes optimization much easier; to make the Lagrangian, just add the constraints to the objective/loss function with a penalty

$$f(\mathbf{x}) + \sum_{i=1}^n \lambda_i g_i(\mathbf{x})$$

We've seen this in Section 4.5 with  $L_1$  and  $L_2$  regularization; we added the  $L_1/L_2$  norm of the network's weight parameters as a penalty for more complex models

# Summary

Optimization plays a critical role in deep learning by allowing us to intelligently update model parameters

We can only optimize the training loss, not the general case

This is generally fine, the Central Limit Theorem argues that with enough training data, the training error will approach the generalization error.

Loss functions can have complex behavior

Optimization algorithms tend to “get stuck”  
where the gradient is vanishingly small

Convex problems have useful properties to leverage in deep learning applications and they make optimization easier