Machine Learning

CMPT 726

Mo Chen
SFU School of Computing Science
2021-11-01

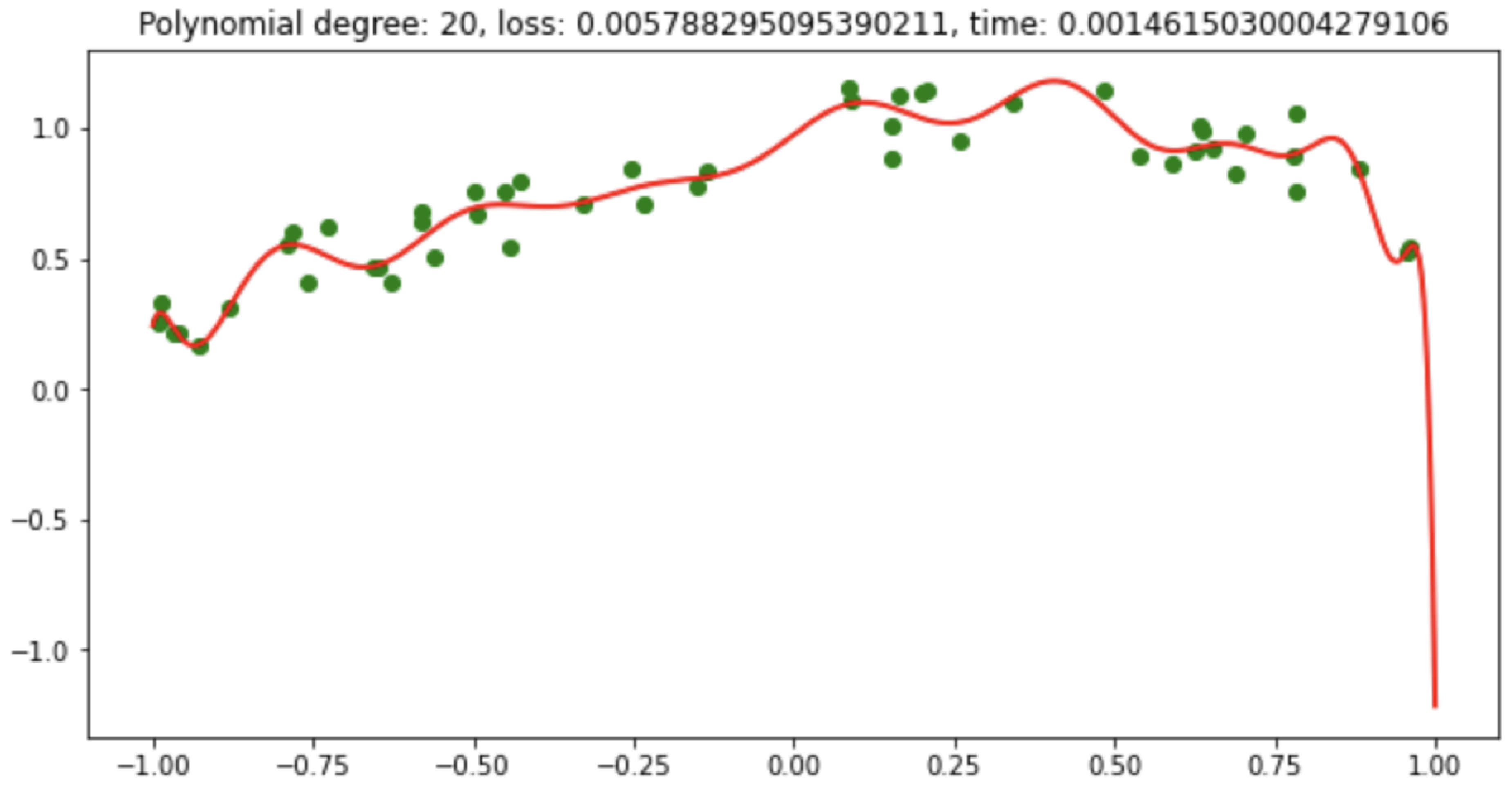
Bias-Variance Tradeoff

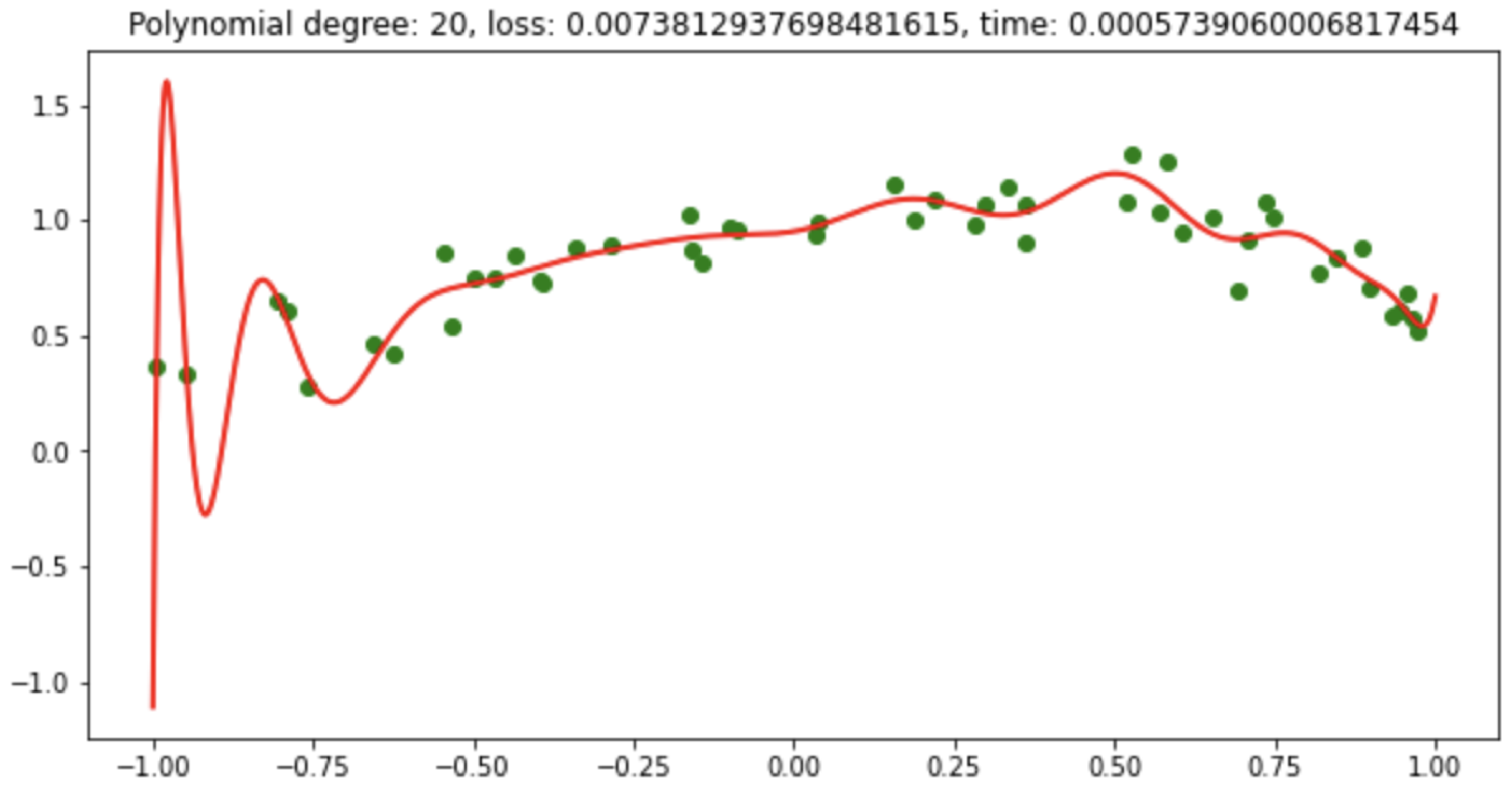
Understanding Overfitting

Recall: Overfitting happens when the model is fitting to the noise in the training data.

So, when overfitting happens, different training datasets can result in very different optimal parameter values, which often result in wrong predictions for some inputs. (See the [Colab notebook]() in Lecture 8)

Ideally, we would like to use a family of models that would typically produce accurate predictions on unseen testing data, regardless of the particular training dataset.





Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= \left(E_{\mathcal{D}}\left[f\left(\vec{x}; \theta^*(\mathcal{D}, L)\right)\middle|\vec{x}\right] - E_y[y|\vec{x}]\right)^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\underbrace{\vec{x}}_{\text{Unseen Testing Example}}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, \underbrace{f}_{\text{Model}}, L) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, \underbrace{L}_{\text{Loss Function}}) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\underbrace{\mathcal{D}}_{\text{Training Dataset}}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \underbrace{\theta^*(\mathcal{D}, L)}_{\text{Optimal parameters on training data}}) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(\underbrace{f(\vec{x}; \theta^*(\mathcal{D}, L))}_{\text{Prediction of trained model on new testing example}} - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - \underbrace{y}_{\text{(Possibly noisy) label corresponding to testing example}})^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[\underbrace{(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2}_{\text{Mean squared error (MSE) on testing example}}|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = \underbrace{E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]}_{\text{Testing error averaged over training datasets compared to noisy versions of the label}}$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (\underbrace{E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}]}_{\text{Prediction of the trained model on testing example, averaged over training datasets}} - E_y[y | \vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - \underbrace{E_y[y|\vec{x}]}_{\text{Average of noisy versions of the label for the testing example}})^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= \underbrace{(E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2}_{\text{Squared difference between average prediction and average label}}$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared

$$+\underbrace{\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})}$$ Variance

Variance of the prediction on the testing example over different training datasets

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error

Bias-Variance Decomposition

$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance

$$+\underbrace{\mathrm{Var}(y | \vec{x})}_{\text{Variance of different noisy versions of the label of the testing example}}$$ Irreducible Error

Bias-Variance Tradeoff

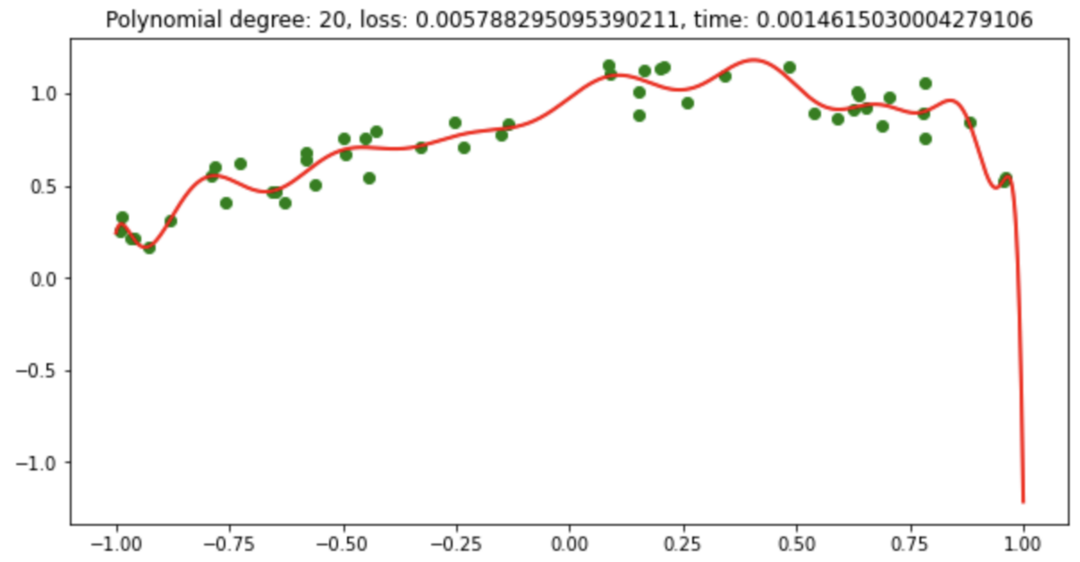
When overfitting:

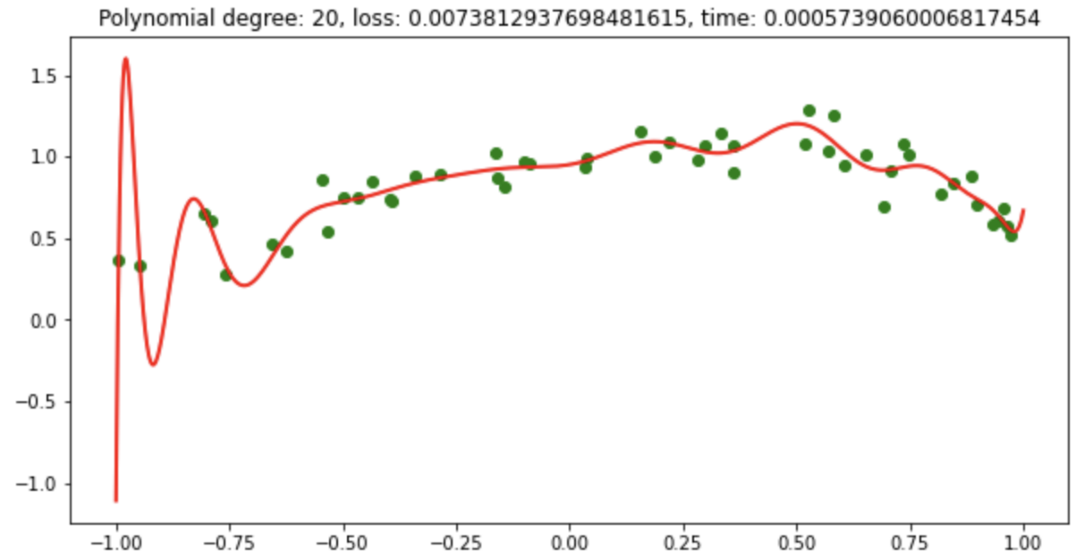
$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x};\theta^*(\mathcal{D},L)) - y)^2|\vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x}] - E_y[y|\vec{x}])^2$$ Bias Squared: Small

$$+\mathrm{Var}(f(\vec{x};\theta^*(\mathcal{D},L))|\vec{x})$$ Variance: Large

$$+\mathrm{Var}(y|\vec{x})$$ Irreducible Error





Bias-Variance Tradeoff

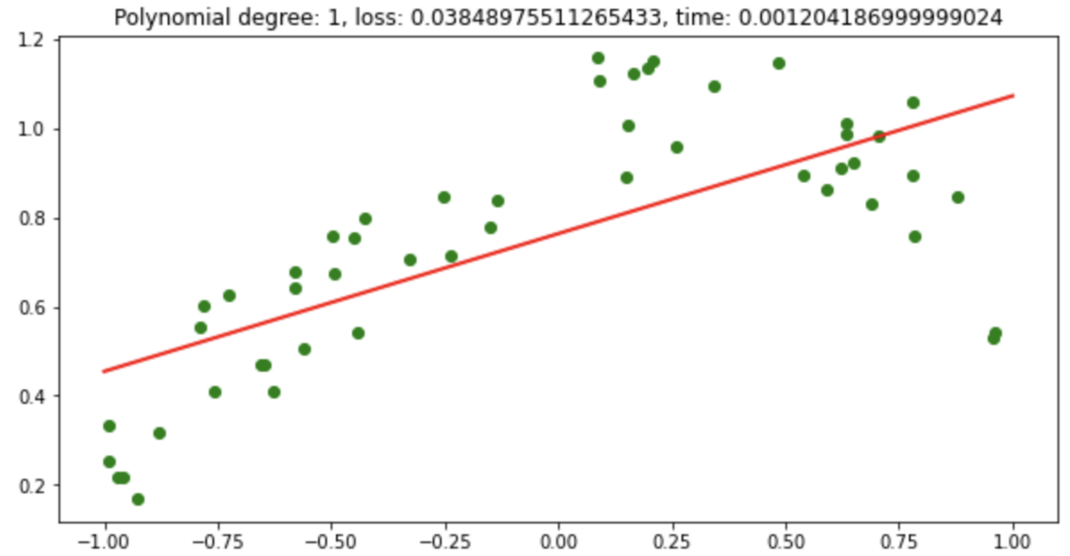
When underfitting:

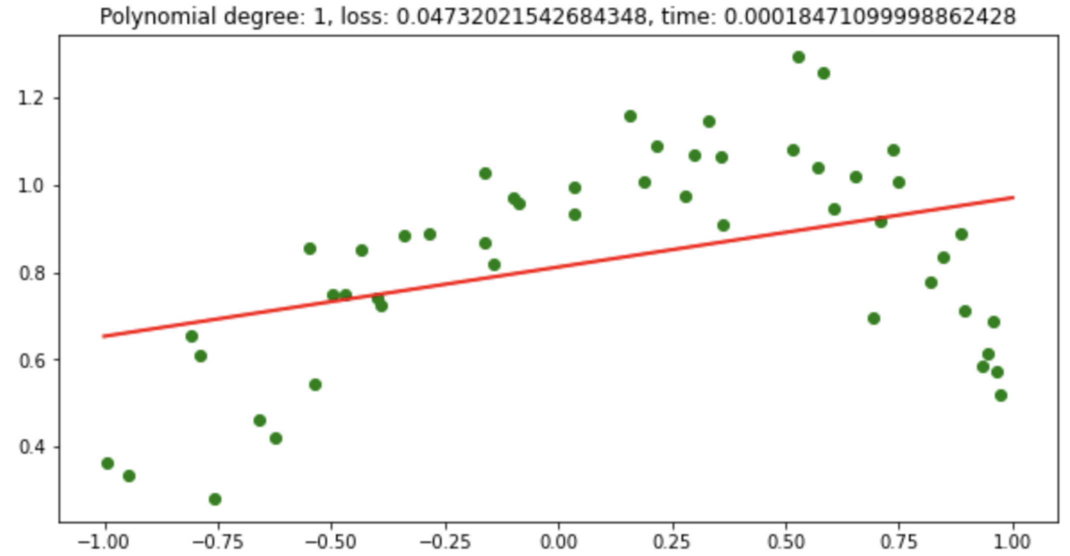
$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared: Large

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance: Small

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error





Bias-Variance Tradeoff

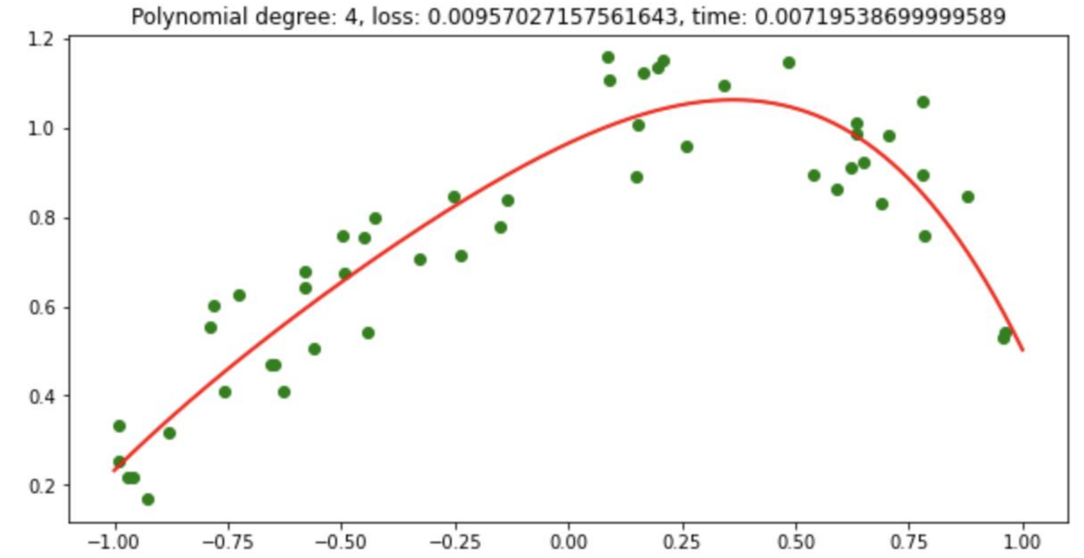
When fitted just right:

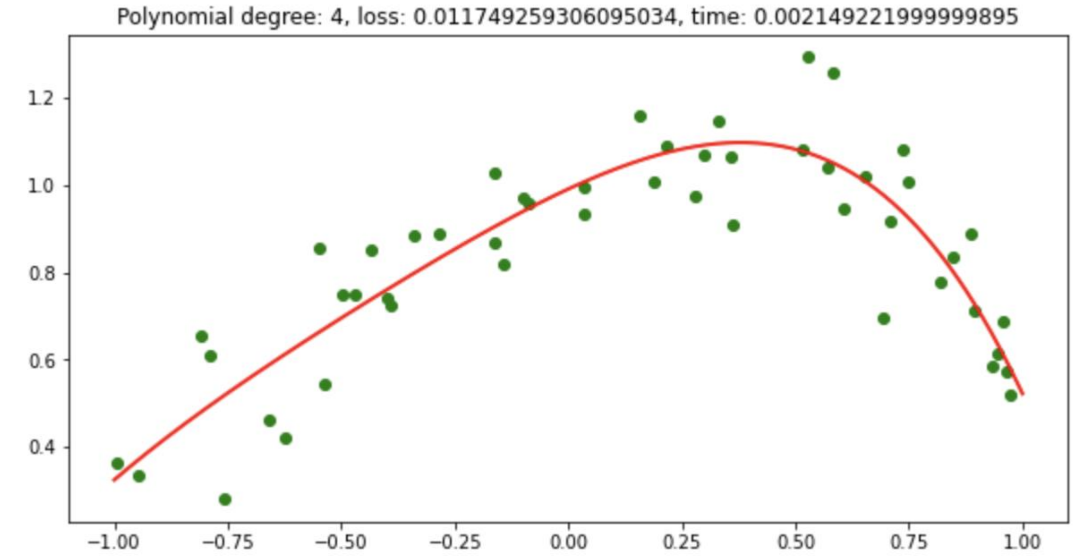
$$\epsilon(\vec{x}, f, L) = E_{y,\mathcal{D}}[(f(\vec{x}; \theta^*(\mathcal{D}, L)) - y)^2 | \vec{x}]$$ Expected Error

$$= (E_{\mathcal{D}}[f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x}] - E_y[y | \vec{x}])^2$$ Bias Squared: Small

$$+\mathrm{Var}(f(\vec{x}; \theta^*(\mathcal{D}, L)) | \vec{x})$$ Variance: Small

$$+\mathrm{Var}(y | \vec{x})$$ Irreducible Error





Takeaways

Expected Error on Testing Example = Bias^2 + Variance + Irreducible Error

Bias: Difference between average prediction and average label of the testing example

Variance: Variance of the prediction on the testing example over different training datasets

Both bias and variance must be low in order for the expected error to be low

When overfitting, bias is low and variance is high

When underfitting, bias is high and variance is low

Simply achieving low bias (which happens when the model is very expressive) isn't enough!

Nonlinear Regression and Optimization

Nonlinear Least Squares

Recall: By replacing raw data with features, we can make the prediction depend *non-linearly* on the raw data.
$\hat{y} = \vec{w}^\top \phi(\vec{x})$

However, the prediction $\hat{y}$ still depends linearly on the parameters $\vec{w}$.

Can we make the prediction depend *non-linearly* on the parameters?

E.g.: $\hat{y} = (\vec{w}^\top \vec{x})^2$

Nonlinear Least Squares

Suppose we have the following data generating process:

$y = f(\vec{x}; \vec{w}) + \sigma\epsilon$,where $\epsilon \sim \mathcal{N}(0,1)$

So, $y | \vec{x}, \vec{w}, \sigma \sim \mathcal{N}(f(\vec{x}; \vec{w}), \sigma^2)$

Hence the likelihood function is:

$$\mathcal{L}(\vec{w}, \sigma; \mathcal{D}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(\vec{x}_i; \vec{w}))^2}{2\sigma^2}\right)$$

$$\widehat{\vec{w}}_{\mathrm{MLE}} = \arg\max_{\vec{w}} \log \mathcal{L}(\vec{w}, \sigma; \mathcal{D}) = \arg\min_{\vec{w}} \sum_{i=1}^{N} \left(y_i - f(\vec{x}_i; \vec{w})\right)^2$$

How do we find the optimal parameters $\widehat{\vec{w}}_{\mathrm{MLE}}$?

Nonlinear Least Squares

To summarize, nonlinear least squares requires solving the following:

$$\widehat{\vec{w}}_{\mathrm{MLE}} = \arg\max_{\vec{w}} \log \mathcal{L}(\vec{w}, \sigma; \mathcal{D}) = \arg\min_{\vec{w}} \sum_{i=1}^{N} \left(y_i - f(\vec{x}_i; \vec{w})\right)^2$$

This can be viewed as solving an instance of the following more general problem:

$$\vec{\theta}^* = \arg\min_{\vec{\theta}} L\left(\vec{\theta}\right)$$

L is known as the "objective function"

$L\left(\vec{\theta}\right)$ is known as the "objective value"

In this case, $\vec{\theta} = \vec{w}$ and $L(\vec{\theta}) = -\log\mathcal{L}(\vec{\theta}, \sigma; \mathcal{D})$.

Such a problem is known as an **optimization problem**, or more specifically, an unconstrained minimization problem.

Nonlinear Least Squares

Goal: Find $\vec{\theta}^* = \arg\min_{\vec{\theta}} L(\vec{\theta})$

The first step to finding a global minimum is to find a critical point. We can try to find the critical points by setting the gradient to zero:

$$\frac{\partial}{\partial \vec{w}}\left(\sum_{i=1}^{N}\left(y_i - f(\vec{x}_i;\vec{w})\right)^2\right) = \sum_{i=1}^{N}\frac{\partial}{\partial \vec{w}}\left(y_i - f(\vec{x}_i;\vec{w})\right)^2$$

$$= -\sum_{i=1}^{N} 2\left(y_i - f\left(\vec{x}_i;\vec{w}\right)\right)\frac{\partial f}{\partial \vec{w}}(\vec{x}_i;\vec{w}) = \vec{0}$$

Cannot solve this in closed form in general because the form of $\frac{\partial f}{\partial \vec{w}}$ could be arbitrary.

Iterative Optimization Algorithms

Goal: Find $\vec{\theta}^* = \arg\min_{\vec{\theta}} L(\vec{\theta})$

Because we cannot in general solve for the critical point analytically, we find it numerically using an iterative optimization algorithm.

The simplest of these algorithms is **gradient descent**.

$\vec{\theta}^{(0)} \leftarrow \text{random vector}$
for $t = 1,2,3, \ldots$
  $\vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} - \gamma_t \frac{\partial L}{\partial \vec{\theta}}\left(\vec{\theta}^{(t-1)}\right)$
  if algorithm has converged
    return $\vec{\theta}^{(t)}$

γ_t is a hyperparameter and is known as the "step size" or "learning rate"

Iterative Optimization Algorithms

Gradient Descent:

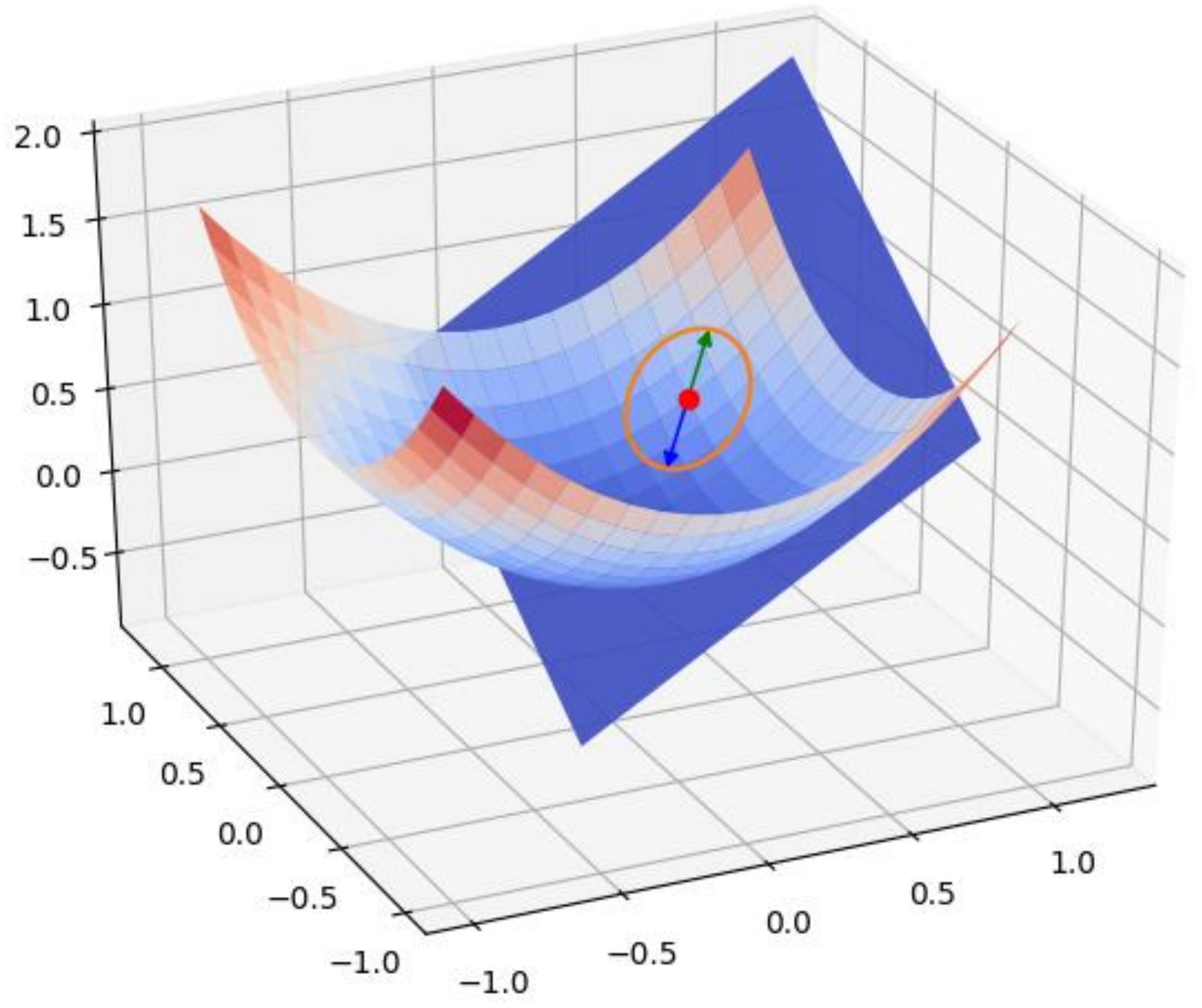
$\overrightarrow{\theta}^{(0)} \leftarrow$ random vector

for $t = 1,2,3, \ldots$

$\quad \vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} - \gamma_t \frac{\partial L}{\partial \vec{\theta}}(\vec{\theta}^{(t-1)})$

$\quad$ if algorithm has converged

$\quad\quad$ return $\overrightarrow{\theta}^{(t)}$



Credit: Pierre Vigier

Iterative Optimization Algorithms

Gradient Descent:

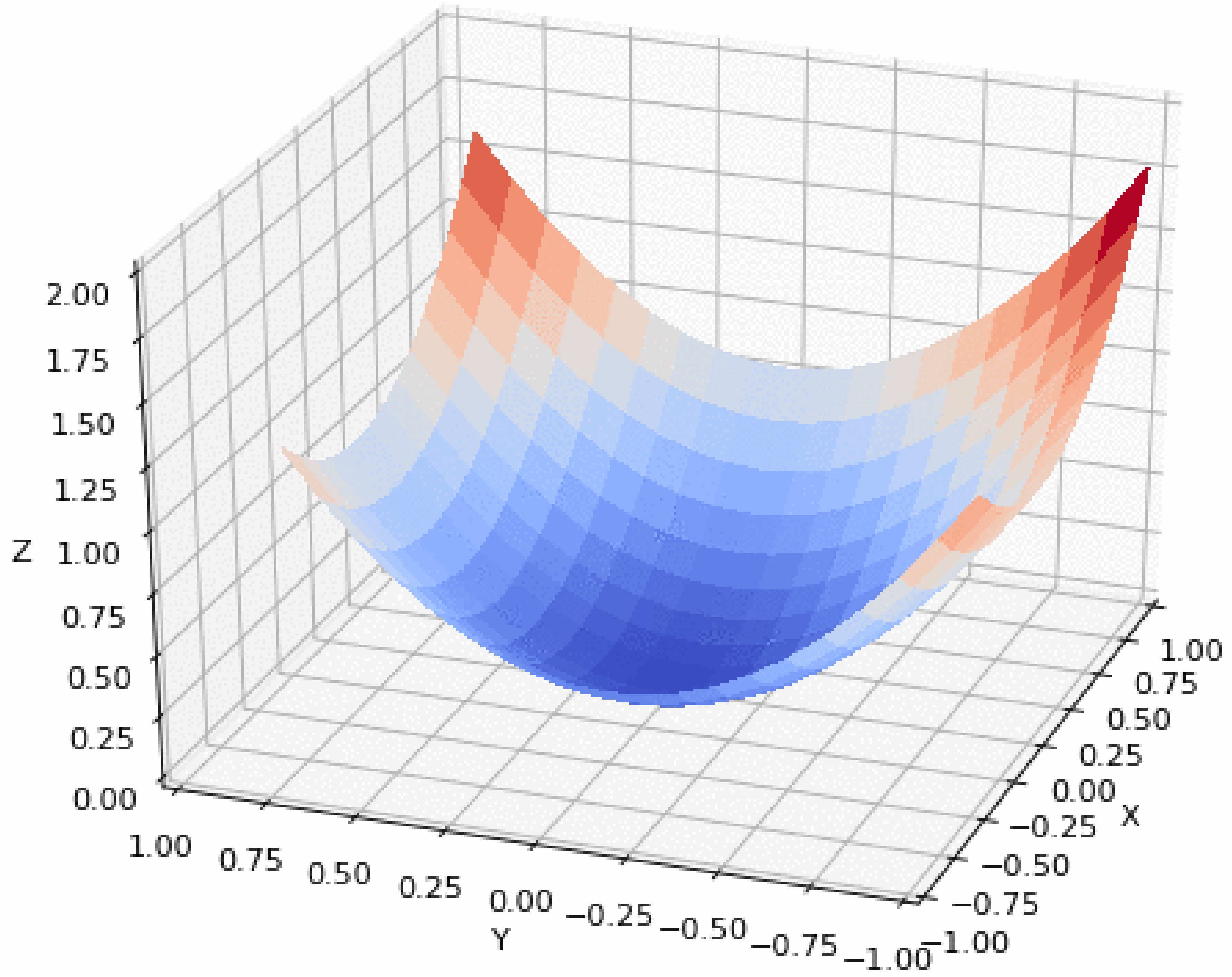
$\overrightarrow{\theta}^{(0)} \leftarrow$ random vector

for $t = 1,2,3, \ldots$

$\vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} - \gamma_t \frac{\partial L}{\partial \vec{\theta}} \left(\vec{\theta}^{(t-1)} \right)$

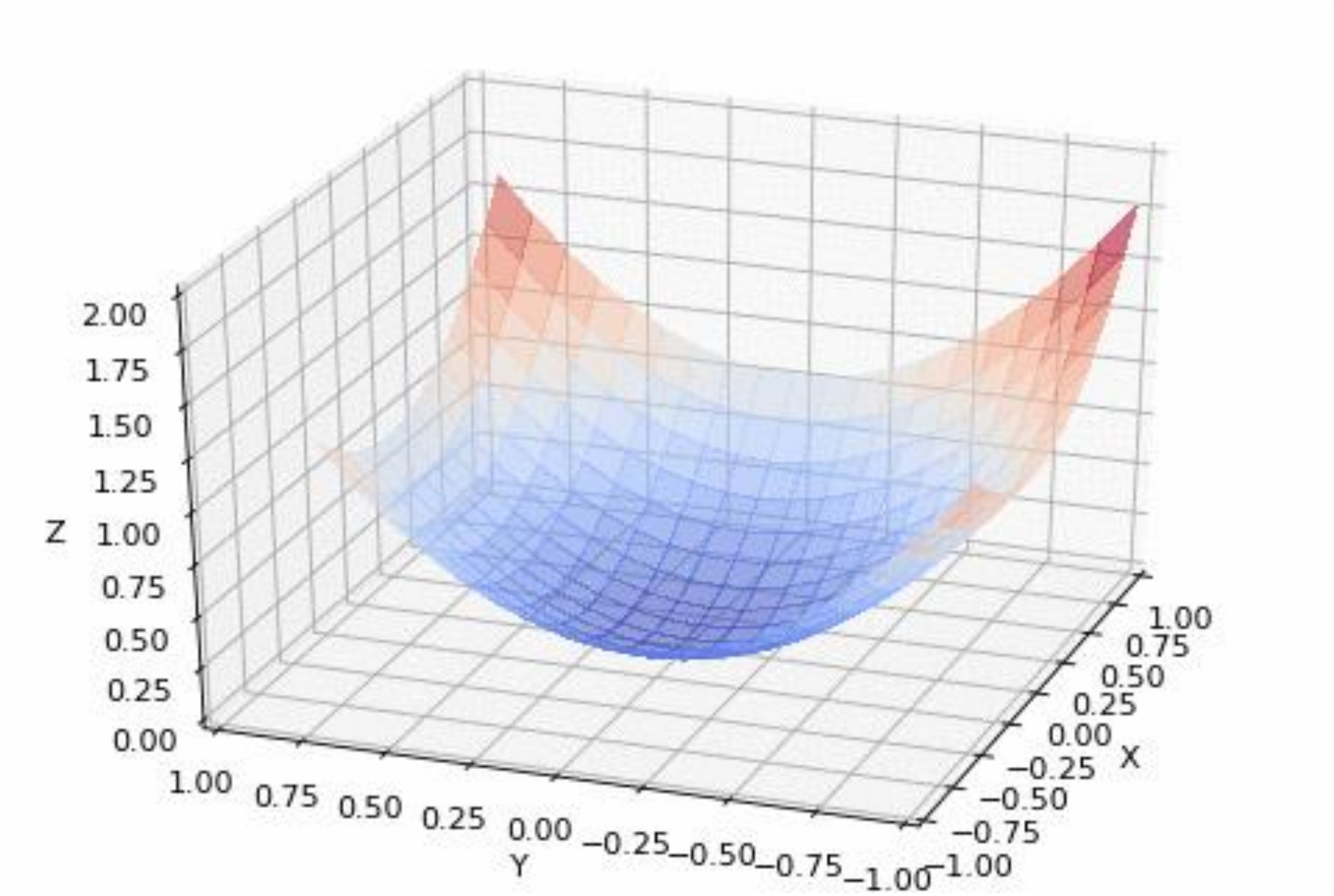
if algorithm has converged

return $\overrightarrow{\theta}^{(t)}$

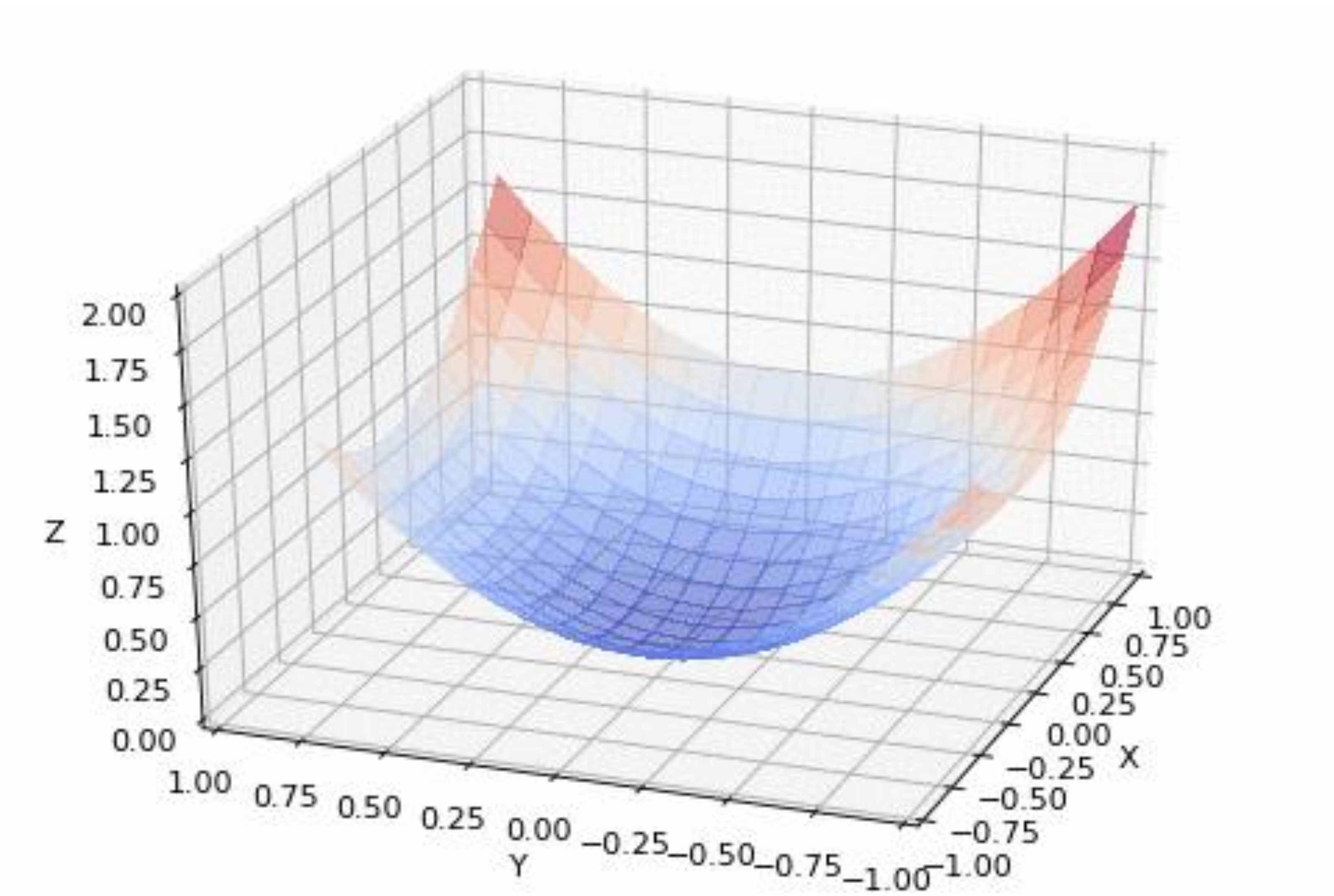


Credit: Pierre Vigier

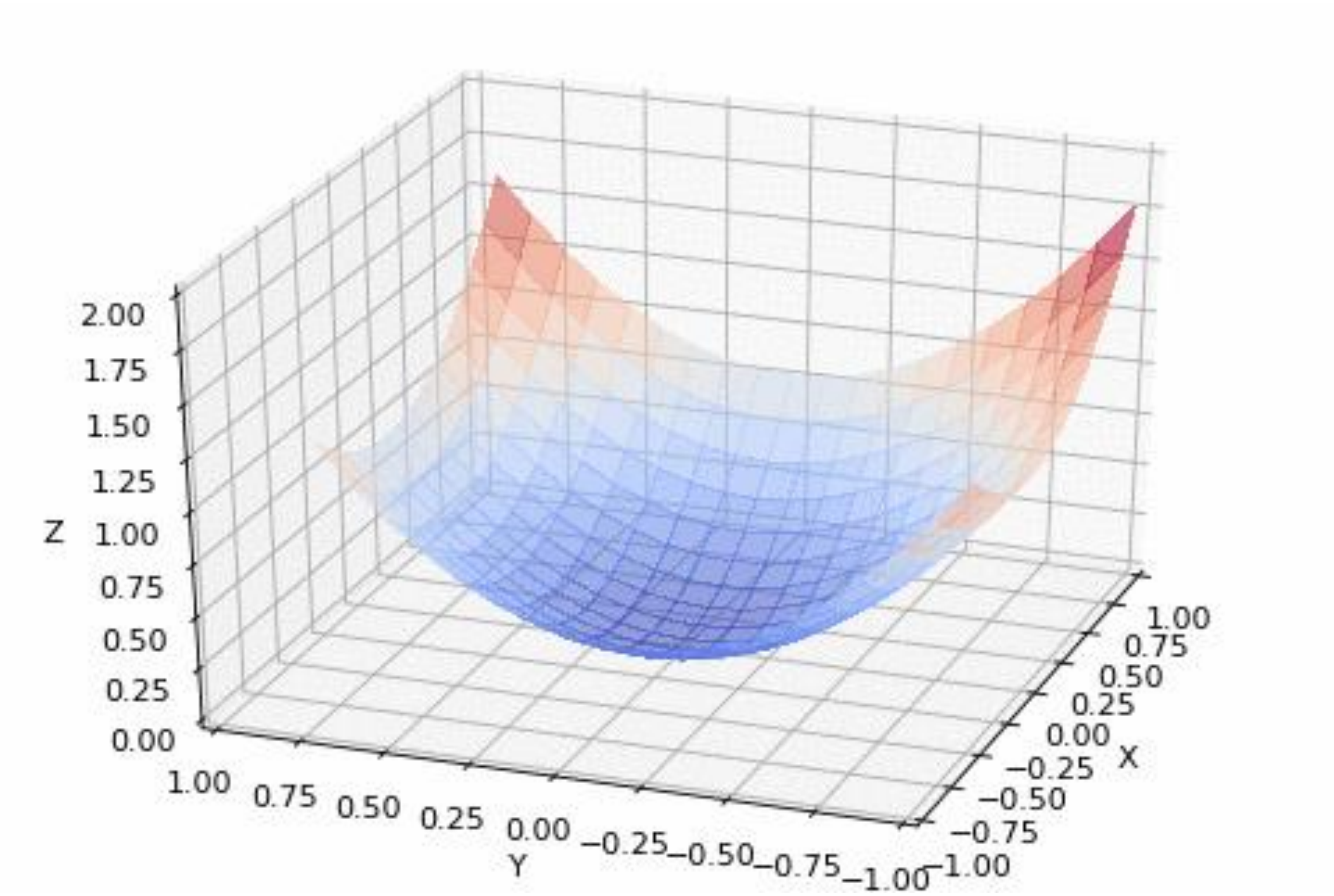
Effect of Step Size



Step size too large



Step size just right



Step size too small

Convergence

What does it mean for the algorithm to converge?

Two different notions of convergence:

- Convergence to a target objective value: as the iteration number increases, the objective value gets closer to the target objective value (e.g.: the minimum objective value)

- Convergence to a target parameter vector: as the iteration number increases, the parameter vector gets closer to the target parameter vector (e.g.: a global minimum, a local minimum or a critical point)

Typically, we don’t know the target parameter vector or objective value.

- To detect convergence (which can be used to determine when to stop the optimization algorithm), we can check if the change in parameters or objective value from the previous iteration is less than a threshold.

Convergence of Gradient Descent

Roughly speaking, if the objective function is **convex** and ***c*-Lipschitz**, gradient descent with a sufficiently small step size is guaranteed to converge to the minimal objective value.

The gap between the minimal objective value and the objective value at iteration t is at most $\Theta\left(\frac{1}{\sqrt{t}}\right)$.

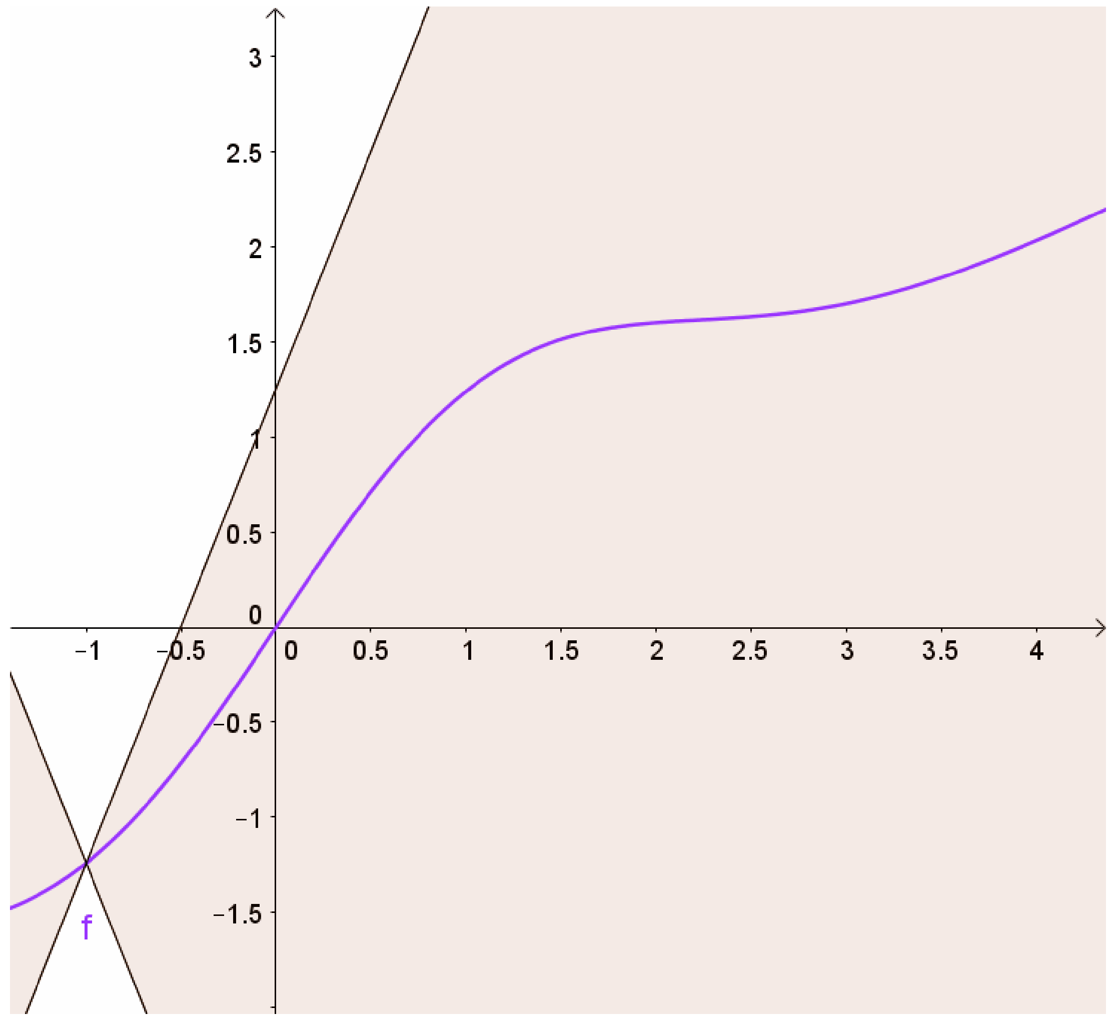
"Convergence Rate"

Equivalently, to get to a parameter vector whose objective value that is ϵ larger than the **minimal objective value**, need $\Theta\left(\frac{1}{\epsilon^2}\right)$ iterations.

Recall: A function $L: \mathbb{R}^n \to \mathbb{R}$ is c-Lipschitz if for all $\vec{x}_1, \vec{x}_2$,

$$|L(\vec{x}_1) - L(\vec{x}_2)| \leq c||\vec{x}_1 - \vec{x}_2||_2$$

An everywhere differentiable function is c-Lipschitz if and only if $||\nabla L(\vec{x})||_2 \leq c$



Convergence of Gradient Descent

Roughly speaking, if the objective function is **convex** and ***c*-Lipschitz**, gradient descent with a sufficiently small step size is guaranteed to converge to the minimal objective value.

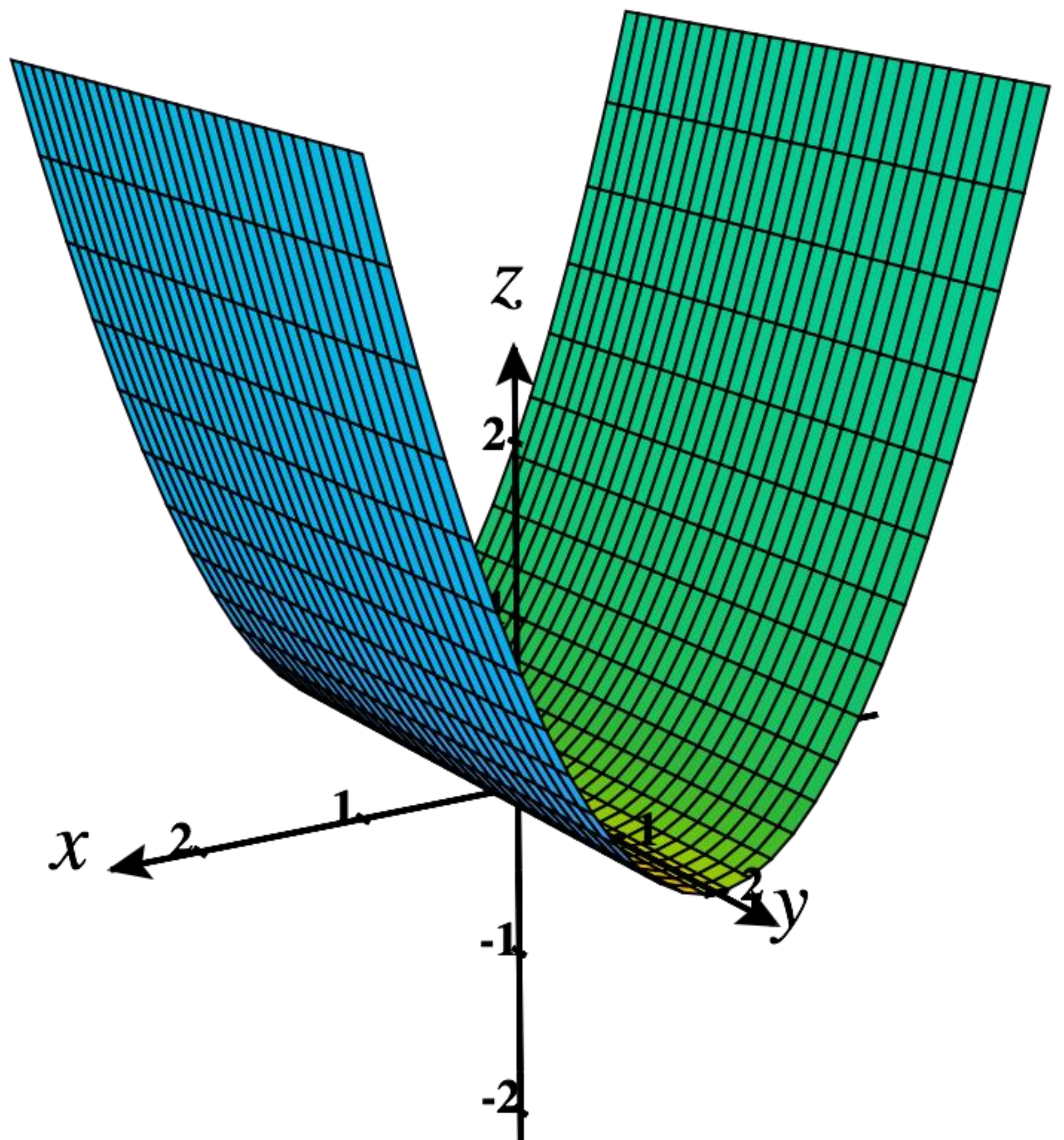
The gap between the minimal objective value and the objective value at iteration t is at most $\Theta\left(\frac{1}{\sqrt{t}}\right)$.

“Convergence Rate”

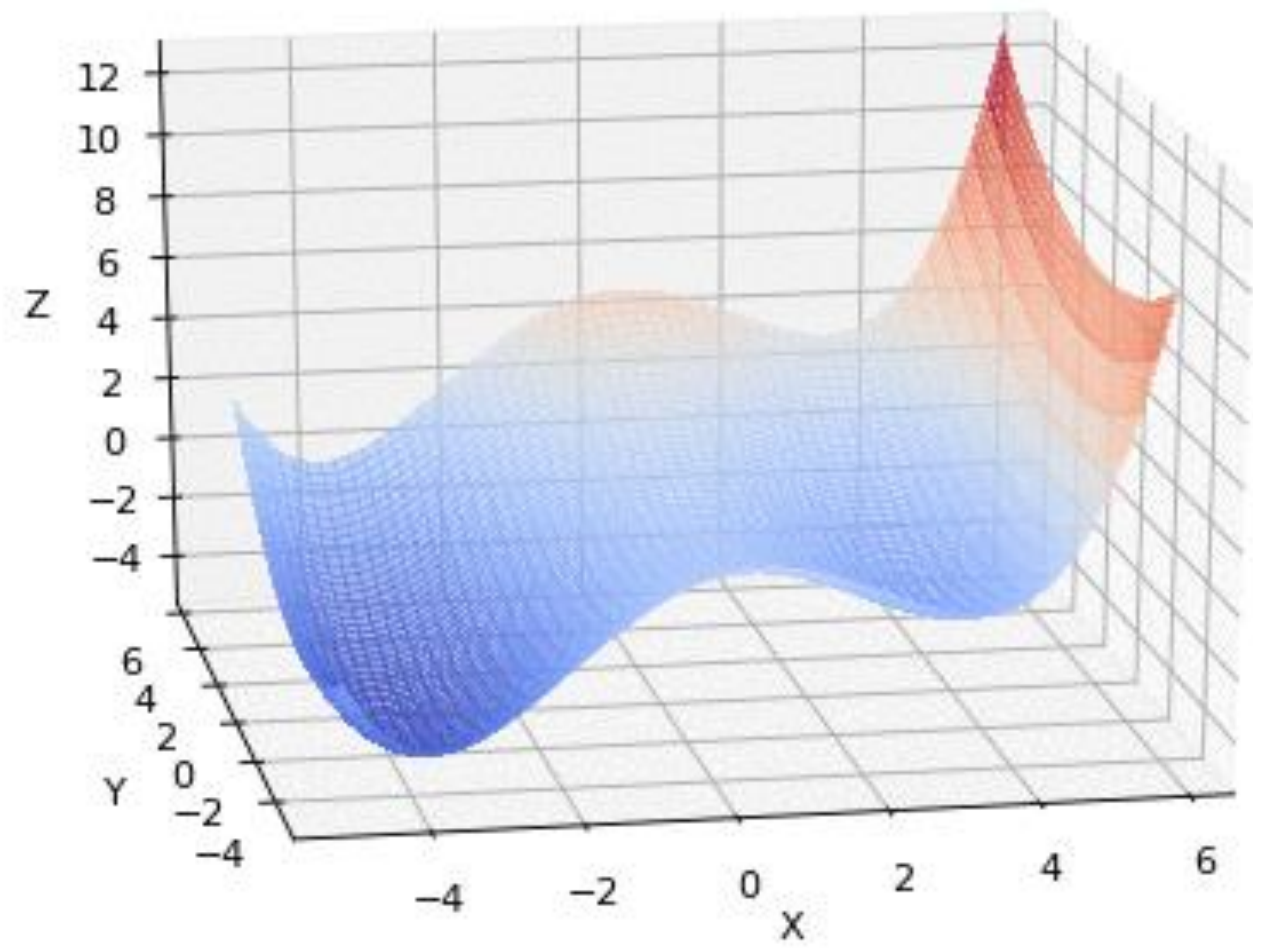
Equivalently, to get to a parameter vector whose objective value that is ϵ larger than the **minimal objective value**, need $\Theta\left(\frac{1}{\epsilon^2}\right)$ iterations.

Recall: A function $L: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if a line segment between any two points on the surface lies on or above the surface.

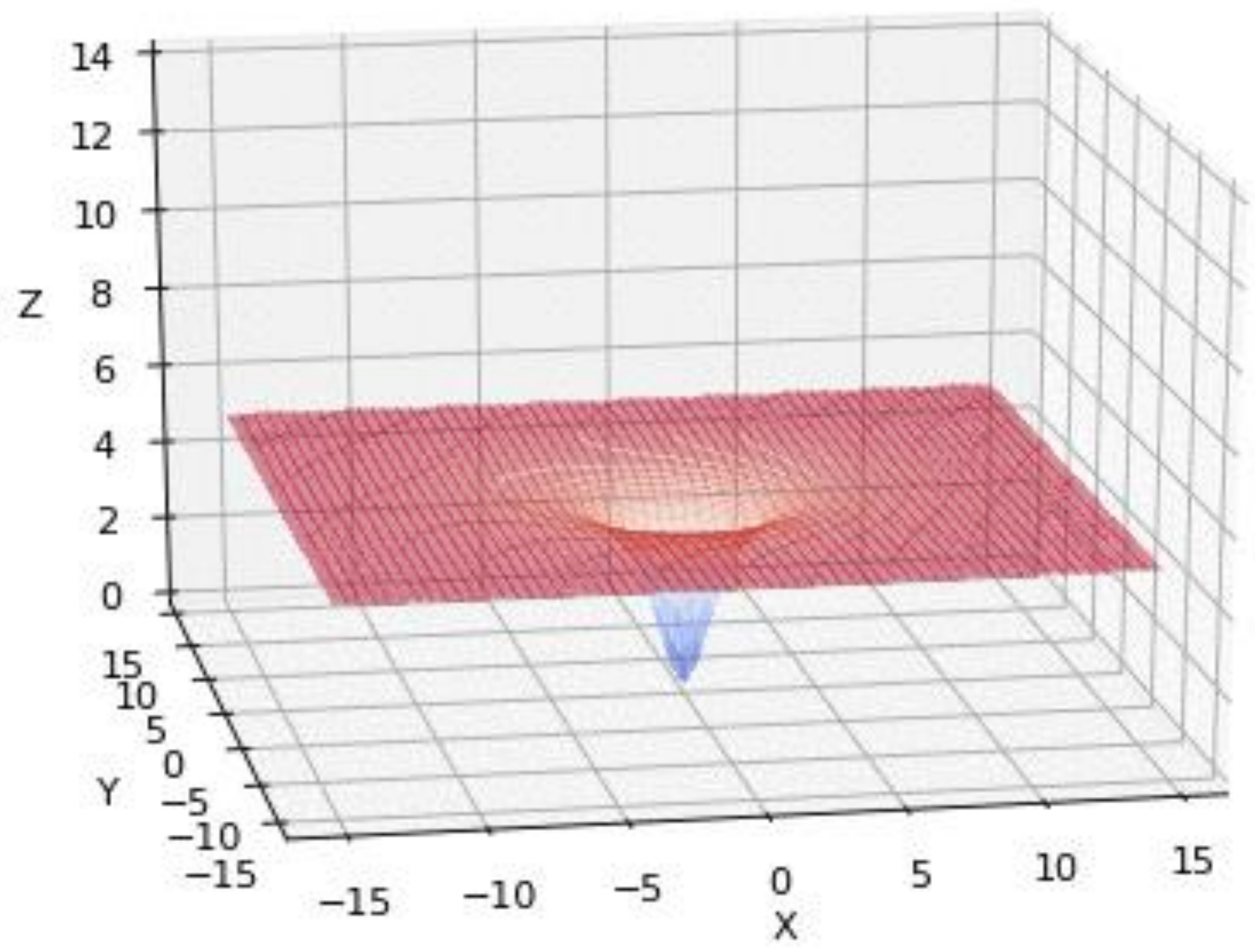
An everywhere twice differentiable function is convex if and only if the Hessian is positive semi-definite.



What Happens on Non-Convex Functions?



Gets stuck in local minimum

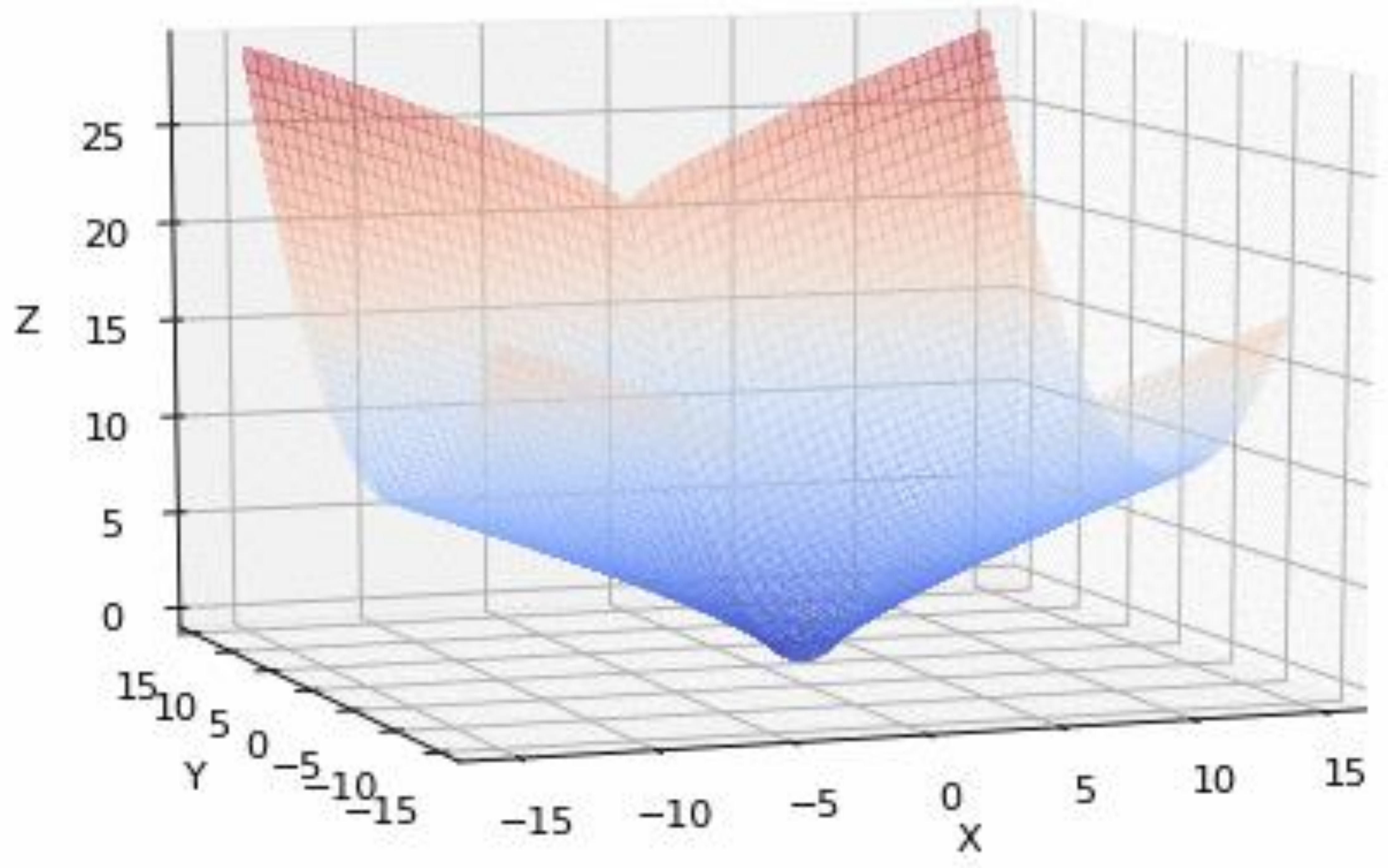


Very slow convergence due to vanishing gradients

What Happens on Non-Lipschitz Functions?

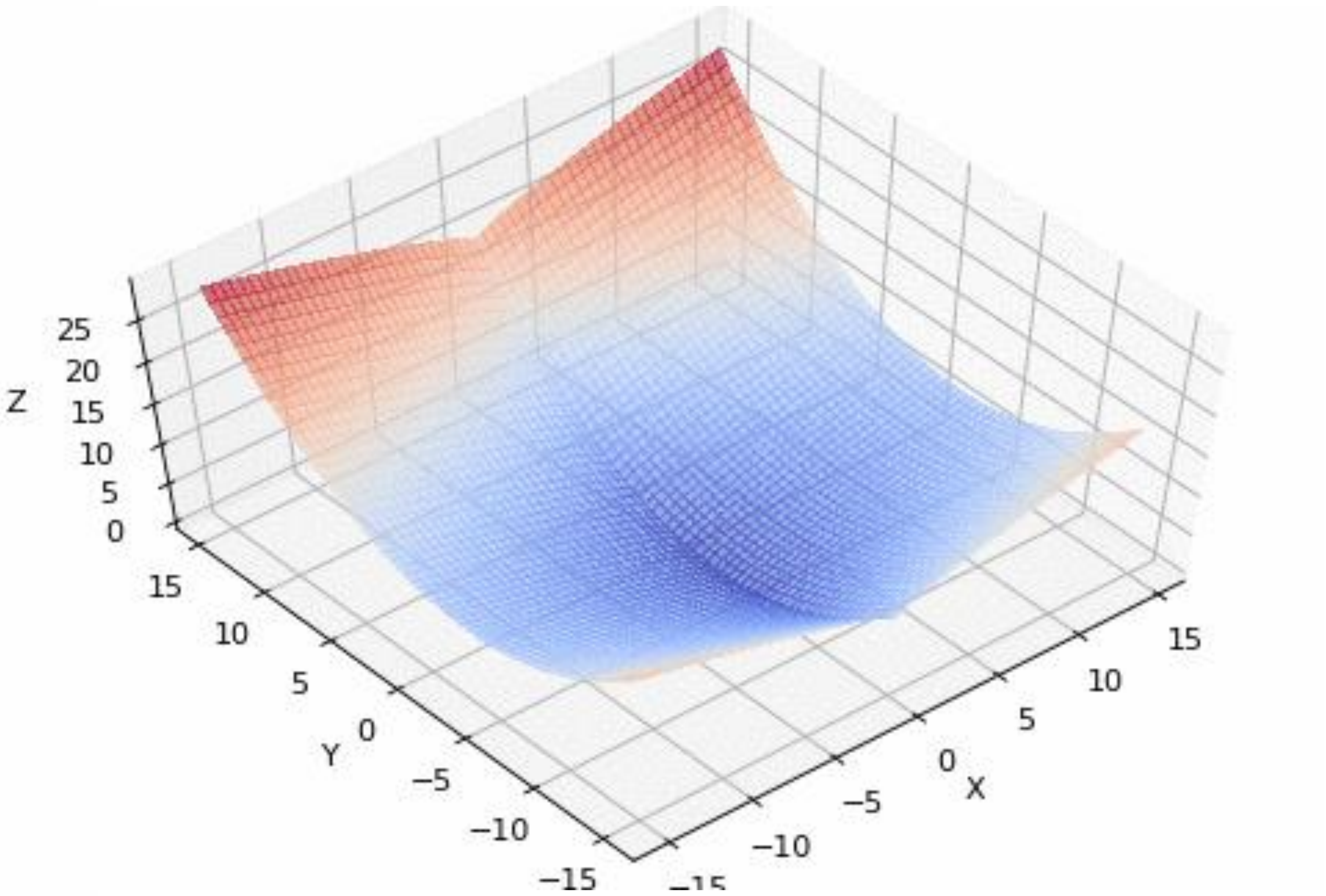
$f(x,y) = |x|^{0.8} + \frac{(y+3)^2}{15}$, so $\frac{\partial f}{\partial x}(x,y) = \mathrm{sign}(x)\,\frac{0.8}{|x|^{0.2}}$

As $x \to 0$, $\left|\frac{\partial f}{\partial x}(x,y)\right| \to \infty$



Front view

Divergence due to exploding gradients



Top view

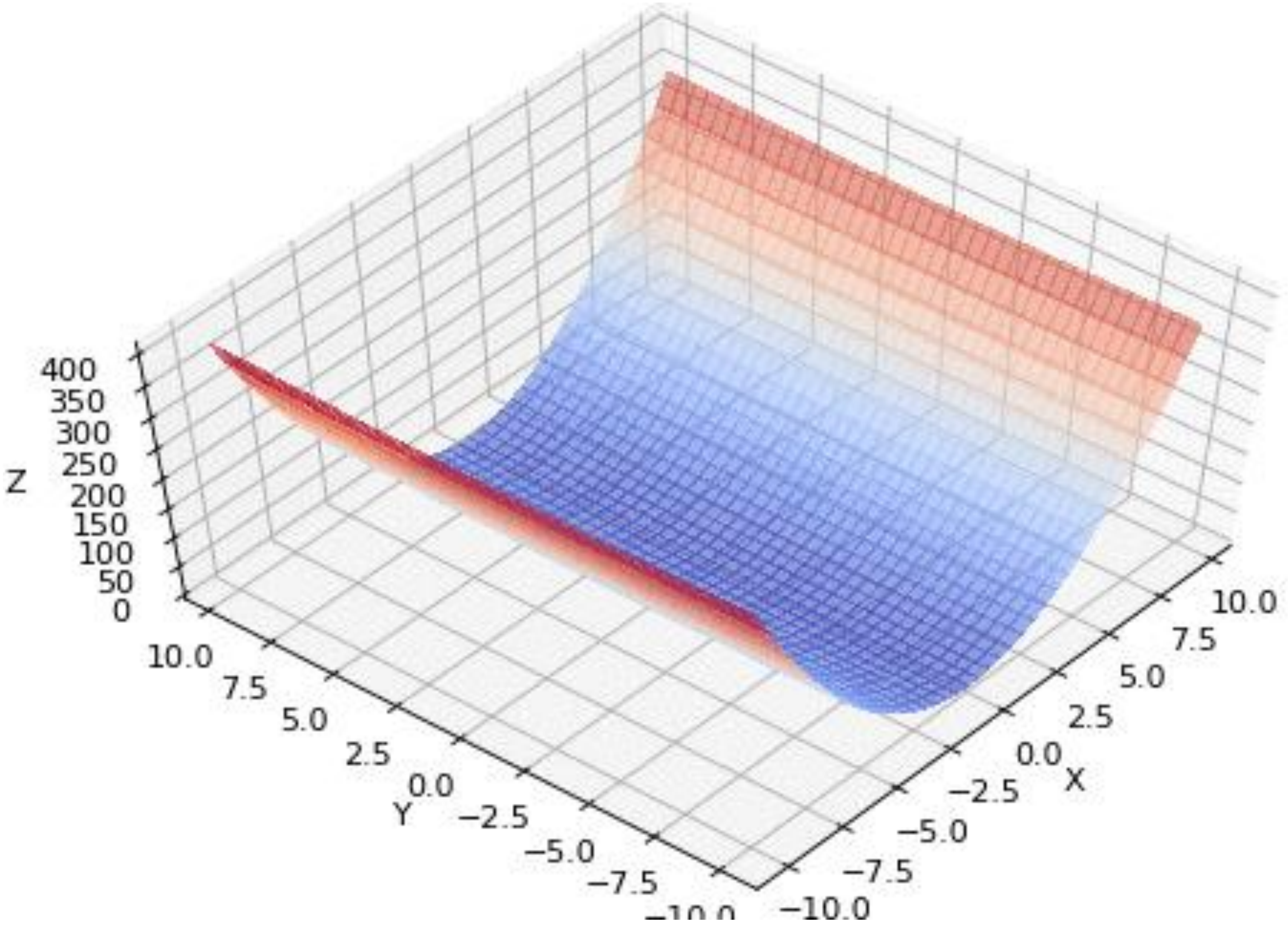
Slow Convergence to Optimal Parameter

Consider $f(x,y) = 4x^2 + \frac{(y+3)^2}{15}$, which is strictly convex.

$$\mathrm{H}f(x,y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}(x,y) & \frac{\partial^2 f}{\partial x \partial y}(x,y) \\ \frac{\partial^2 f}{\partial y \partial x}(x,y) & \frac{\partial^2 f}{\partial y^2}(x,y) \end{pmatrix} = \begin{pmatrix} 8 & 0 \\ 0 & \frac{2}{15} \end{pmatrix} \succ 0$$

As shown, gradient descent converges to the optimal parameter vector $\binom{0}{-3}$ very slowly.

(Though it does converge to the minimal objective value fairly quickly.)



Quiz Practice

Q1: Which of the following statements about gradient descent is NOT true?

(A) Gradient descent may not find the global minimum on non-convex functions

(B) Gradient descent may diverge on non-Lipschitz functions

(C) With a sufficiently small step size, gradient descent always converges at a rate of $\Theta(1/\epsilon^2)$ to the minimal objective value on convex Lipschitz functions with a sufficiently small step size

(D) With a sufficiently small step size, gradient descent always converges at a rate of $\Theta(1/\epsilon^2)$ to the minimal objective value on functions whose every local minimum is a global minimum

(E) With a sufficiently small step size, gradient descent always converges at a rate of $\Theta(1/\epsilon^2)$ to the optimal parameters when the objective function is both convex and Lipschitz

(F) All are true

(G) Don’t know

Gradient Descent with Momentum

Often known as just “momentum” or “Polyak’s heavy ball method”.

Gradient Descent with Momentum:

$\vec{\theta}^{(0)} \leftarrow$ random vector

Akin to position of a particle

$\Delta\vec{\theta} = \vec{0}$

Akin to velocity of the particle

for $t = 1,2,3,\ldots$

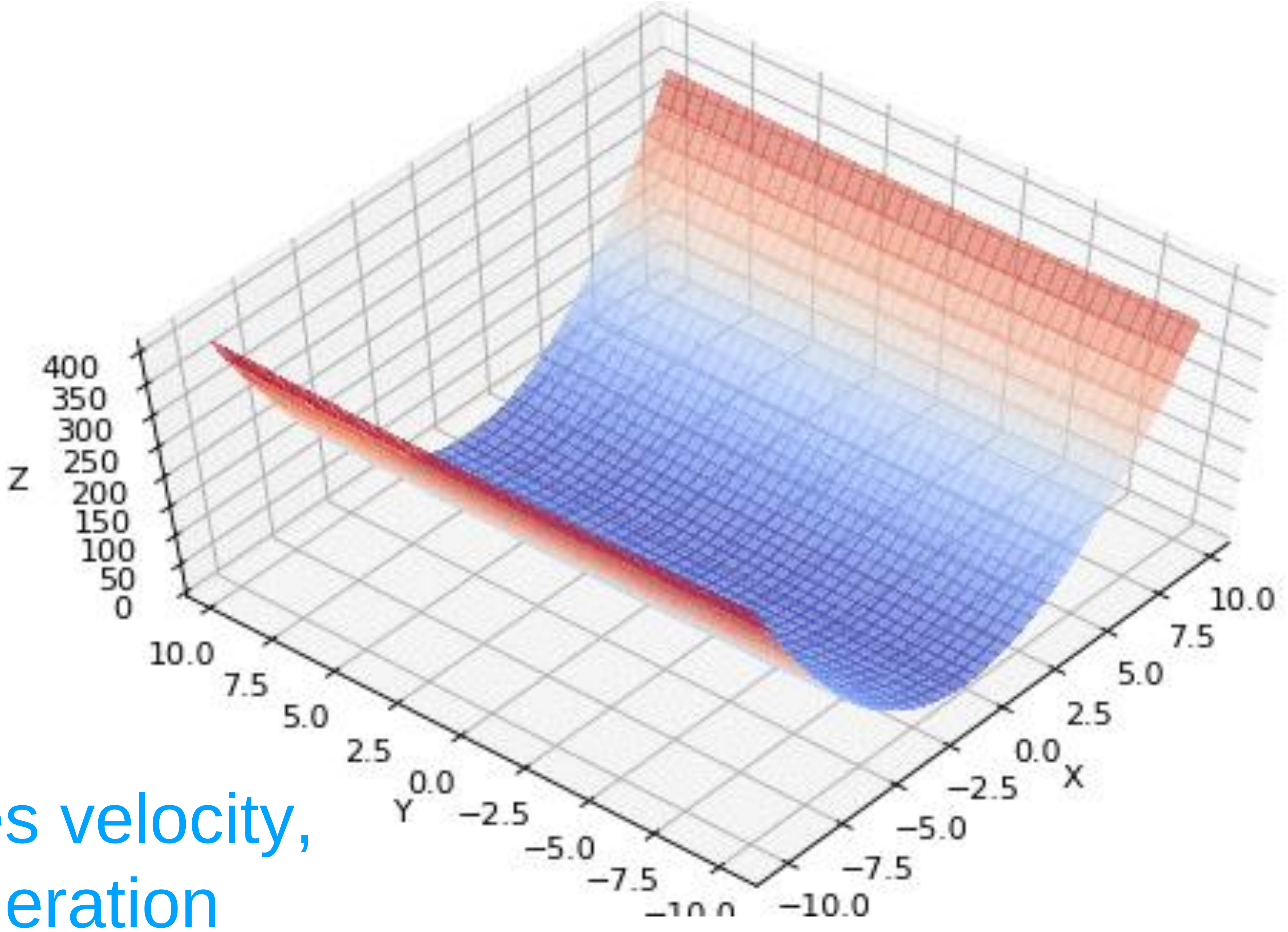
$$\Delta\vec{\theta} \leftarrow \alpha\Delta\vec{\theta} - \gamma_t \frac{\partial L}{\partial \vec{\theta}}\left(\vec{\theta}^{(t-1)}\right)$$

$$\vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} + \Delta\vec{\theta}$$

Gradient changes velocity, causing acceleration

if algorithm has converged

return $\vec{\theta}^{(t)}$



Gradient Descent with Momentum

Often known as just “momentum” or “Polyak’s heavy ball method”.

Gradient Descent with Momentum:

$\vec{\theta}^{(0)} \leftarrow \text{random vector}$

$\Delta\vec{\theta} = \vec{0}$

for $t = 1,2,3, \ldots$

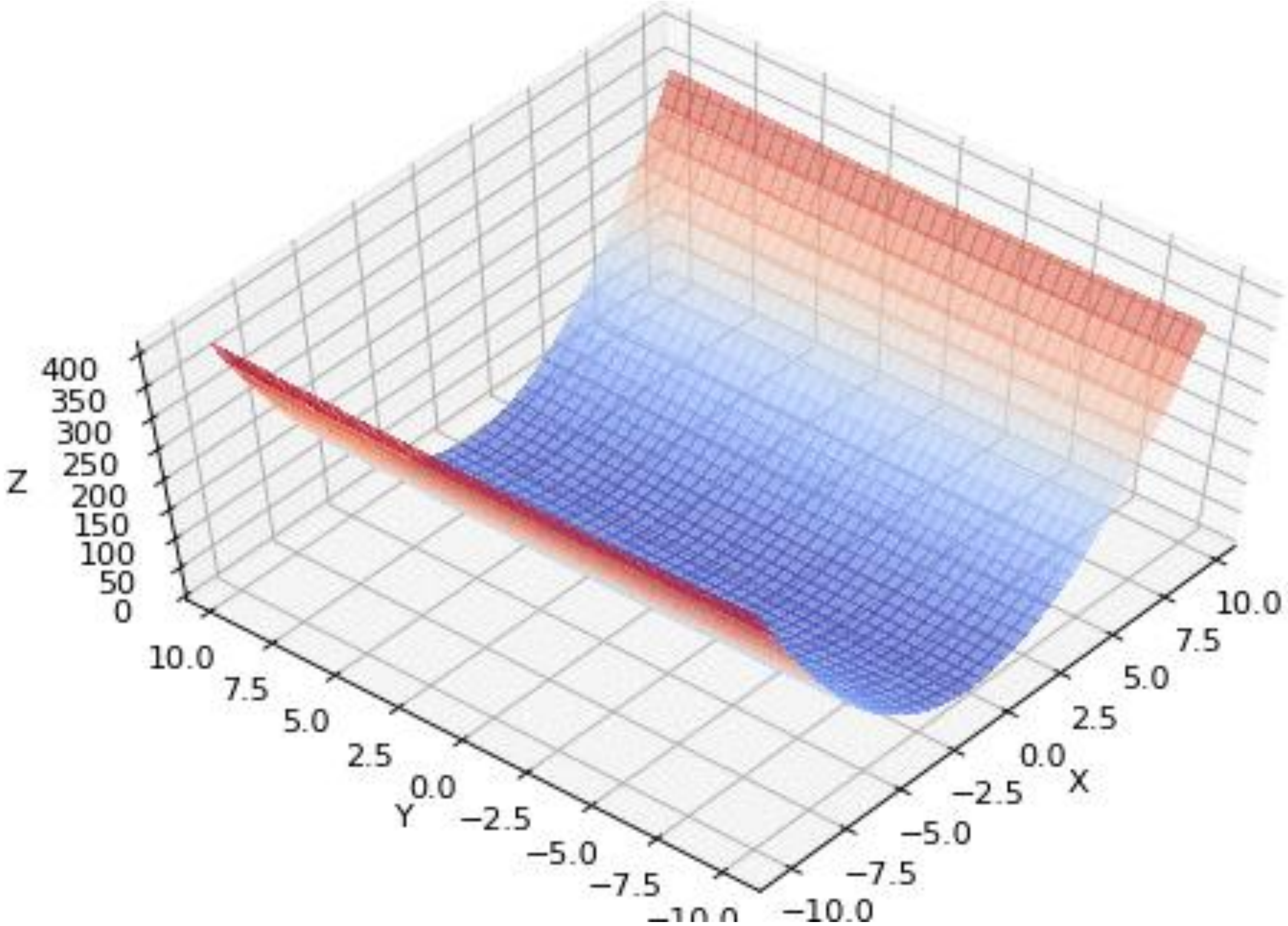
$\quad \Delta\vec{\theta} \leftarrow \alpha\Delta\vec{\theta} - \gamma_t \frac{\partial L}{\partial\vec{\theta}}\left(\vec{\theta}^{(t-1)}\right)$

$\quad \vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} + \Delta\vec{\theta}$

$\quad$ if algorithm has converged

$\quad\quad$ return $\vec{\theta}^{(t)}$

$\alpha \in [0,1)$ is a hyperparameter and is known as the “momentum parameter”



Momentum vs. Gradient Descent

Often known as just “momentum” or “Polyak’s heavy ball method”.

Gradient Descent with Momentum:

$\vec{\theta}^{(0)} \leftarrow \text{random vector}$

$\Delta\vec{\theta} = \vec{0}$

for $t = 1,2,3,\ldots$

Gradient changes velocity

$\quad \Delta\vec{\theta} \leftarrow \alpha\Delta\vec{\theta} - \gamma_t \frac{\partial L}{\partial \vec{\theta}}\left(\vec{\theta}^{(t-1)}\right)$

$\quad \vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} + \Delta\vec{\theta}$

$\quad$ if algorithm has converged

$\quad\quad$ return $\vec{\theta}^{(t)}$

Gradient Descent:

$\vec{\theta}^{(0)} \leftarrow \text{random vector}$

for $t = 1,2,3,\ldots$

Gradient changes position

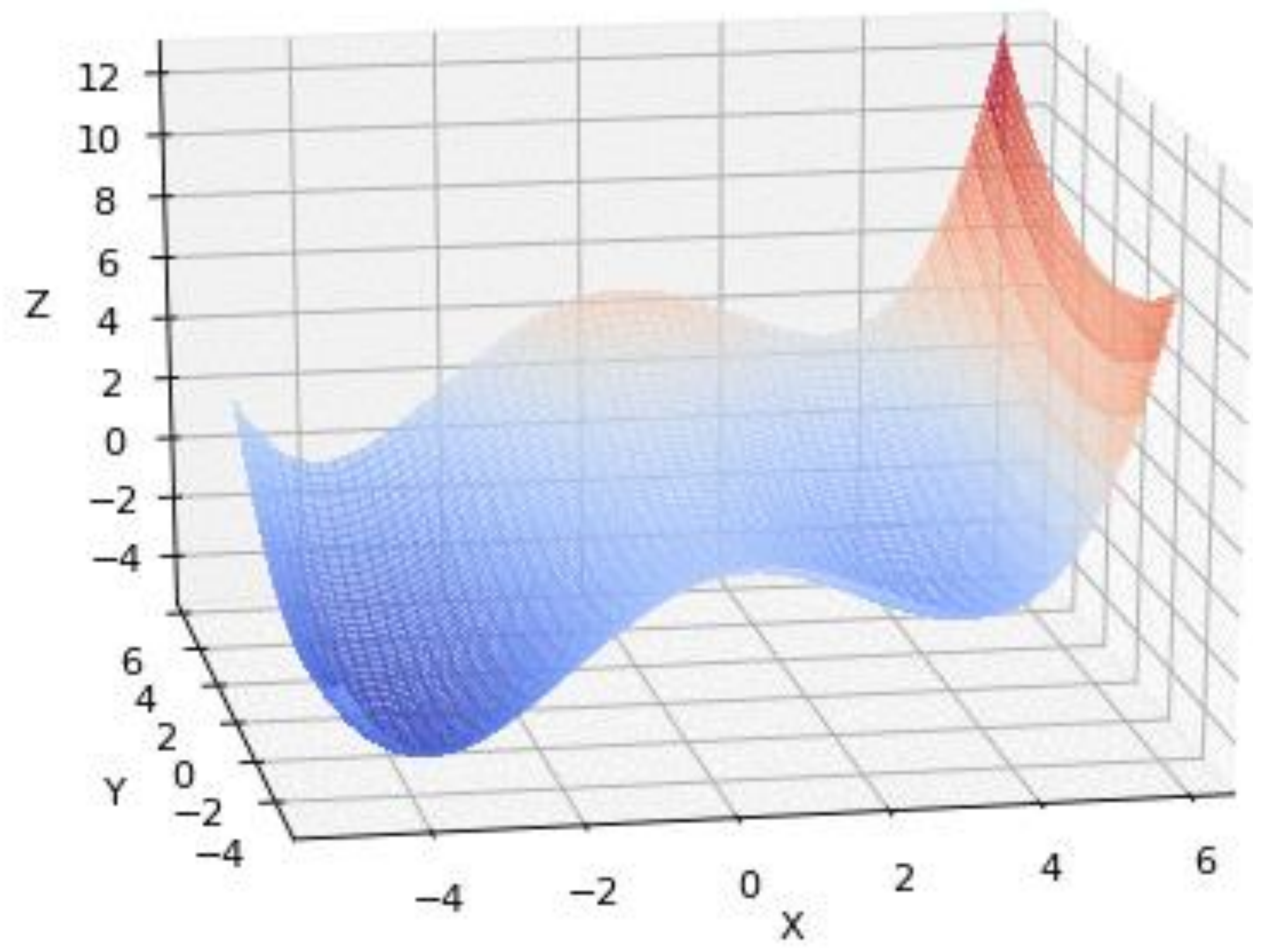
Equivalent to setting $\alpha = 0$

$\quad \vec{\theta}^{(t)} \leftarrow \vec{\theta}^{(t-1)} - \gamma_t \frac{\partial L}{\partial \vec{\theta}}\left(\vec{\theta}^{(t-1)}\right)$

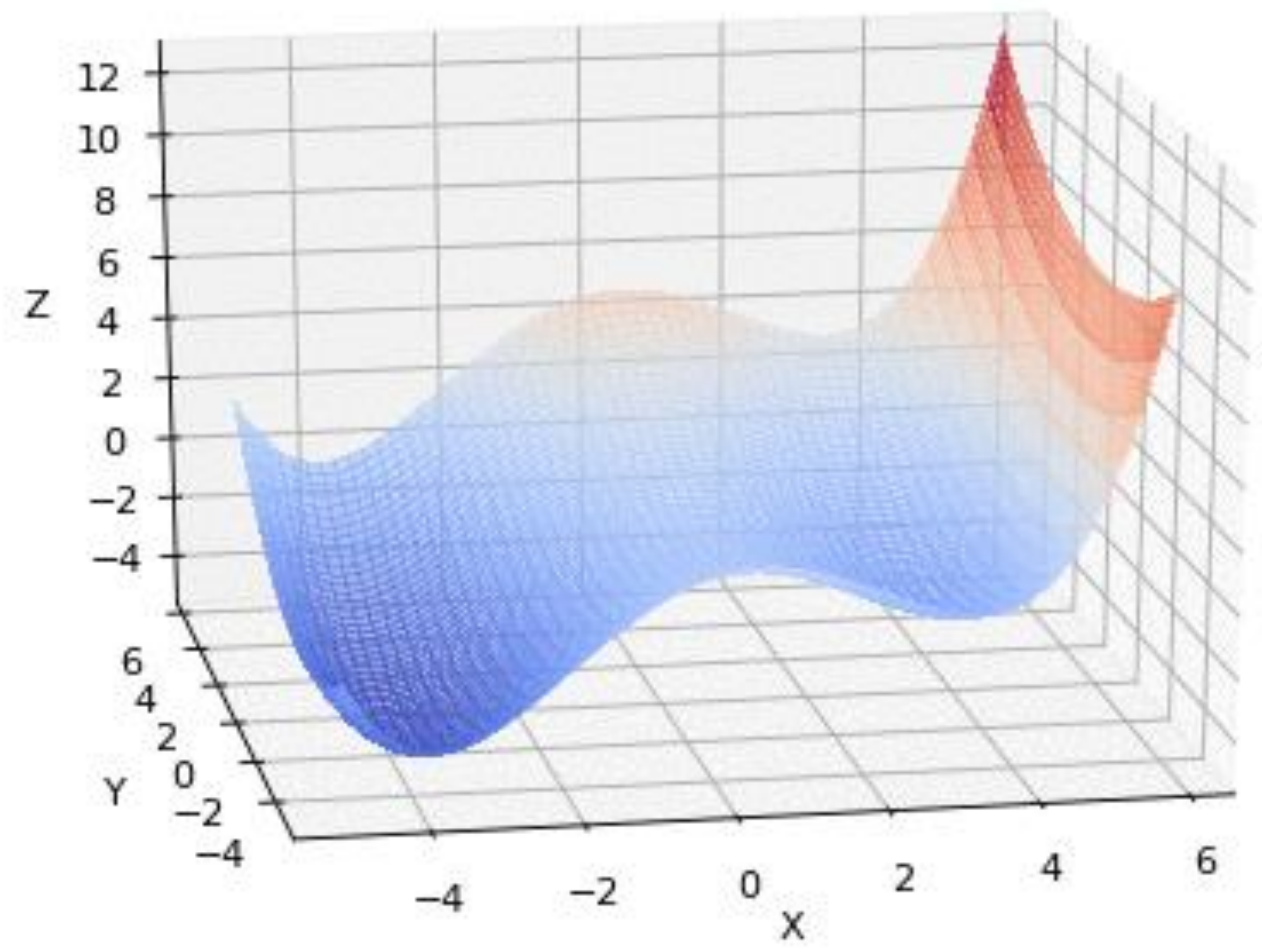
$\quad$ if algorithm has converged

$\quad\quad$ return $\vec{\theta}^{(t)}$

What Happens on Non-Convex Functions?

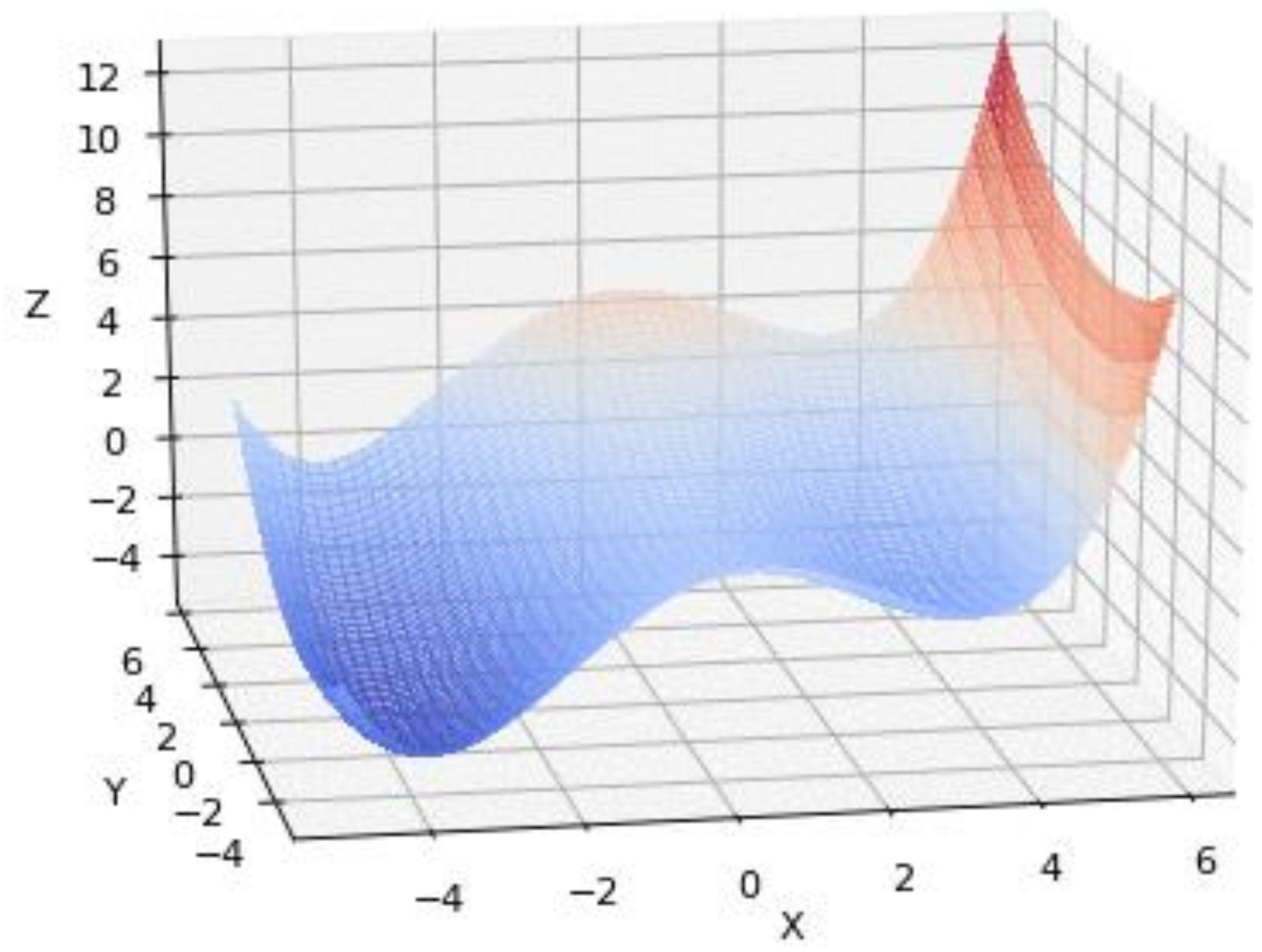


Gradient Descent
Gets stuck in local minimum

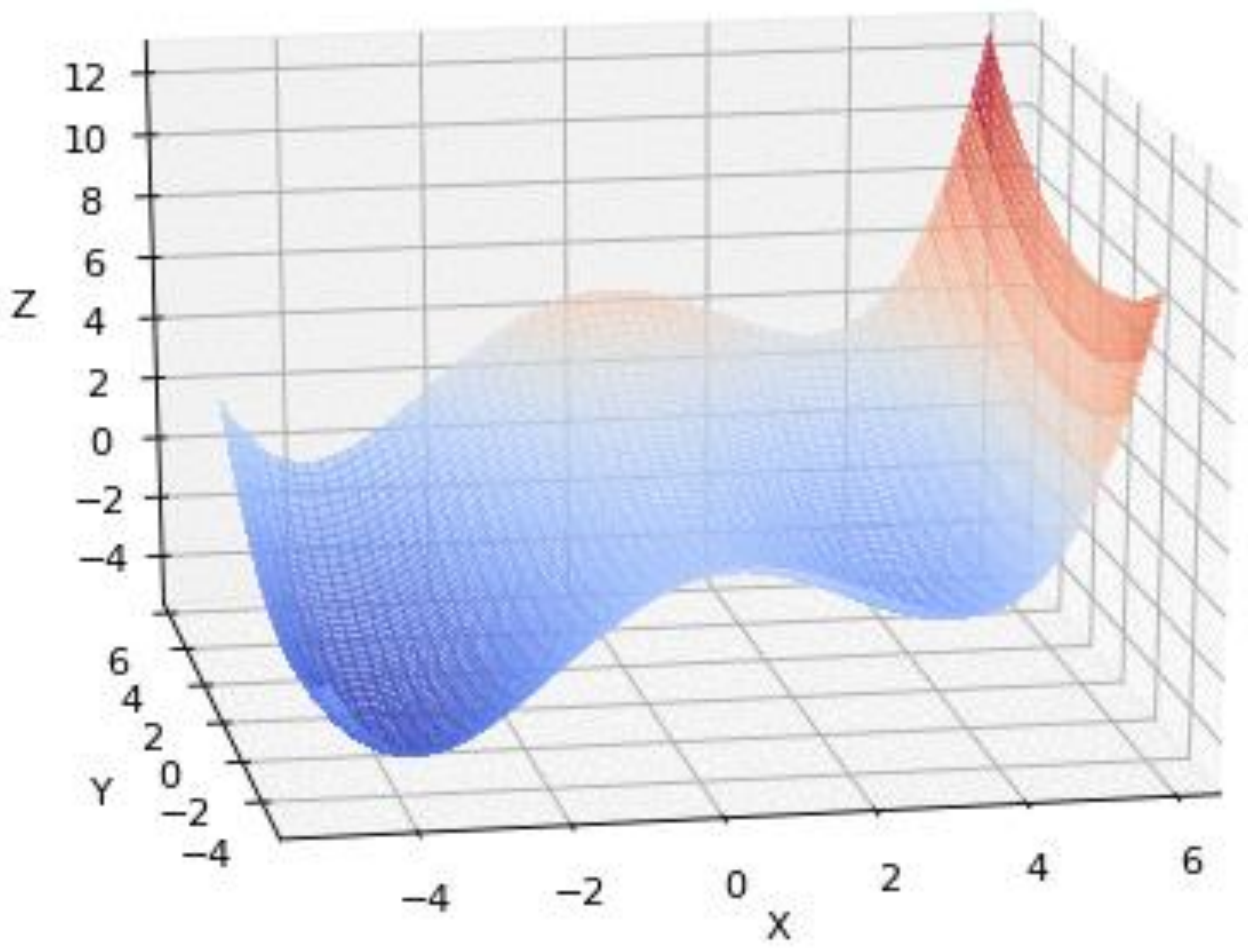


Momentum
Sometimes gets past local minimum

What Happens on Non-Convex Functions?

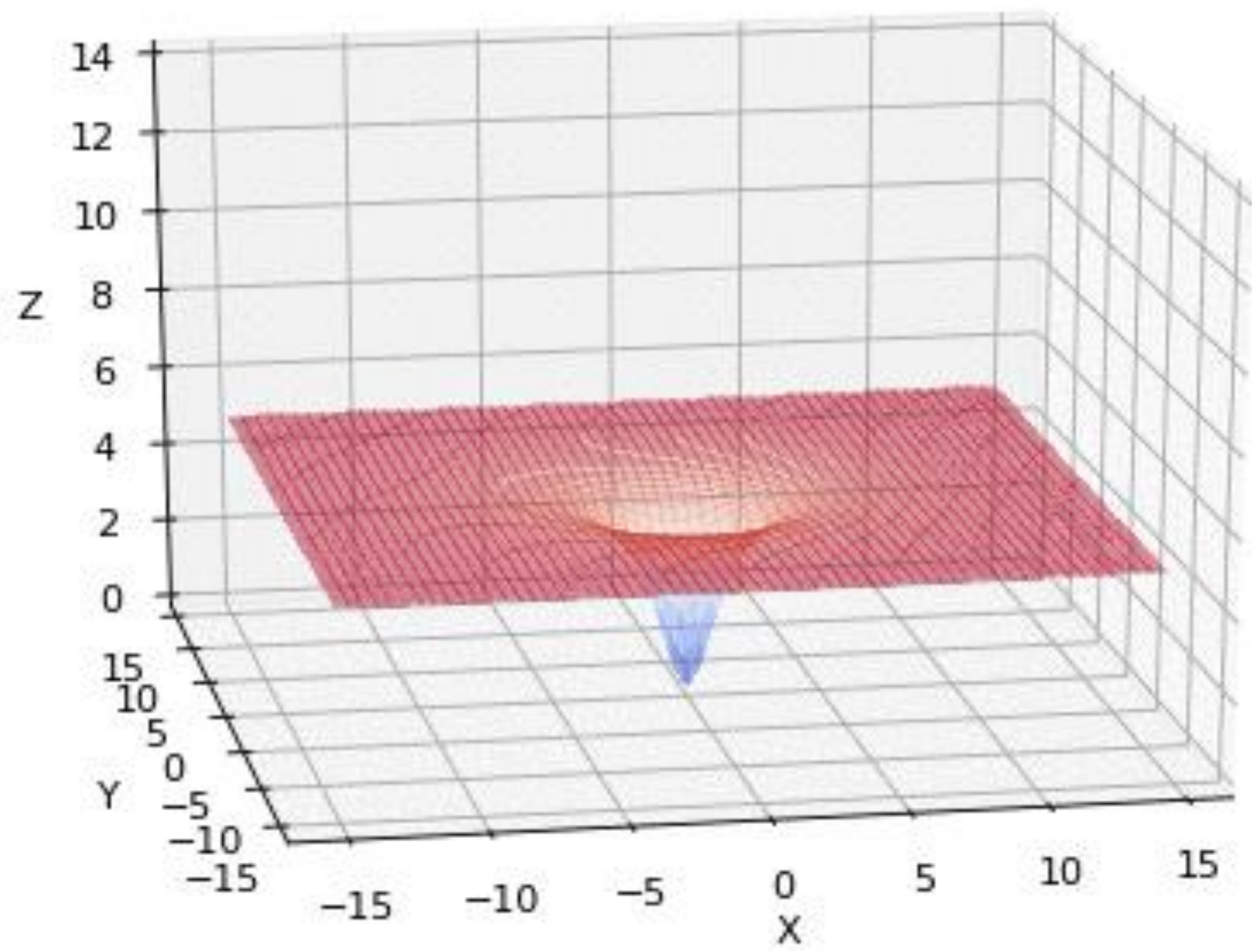


Gradient Descent
Gets stuck in local minimum

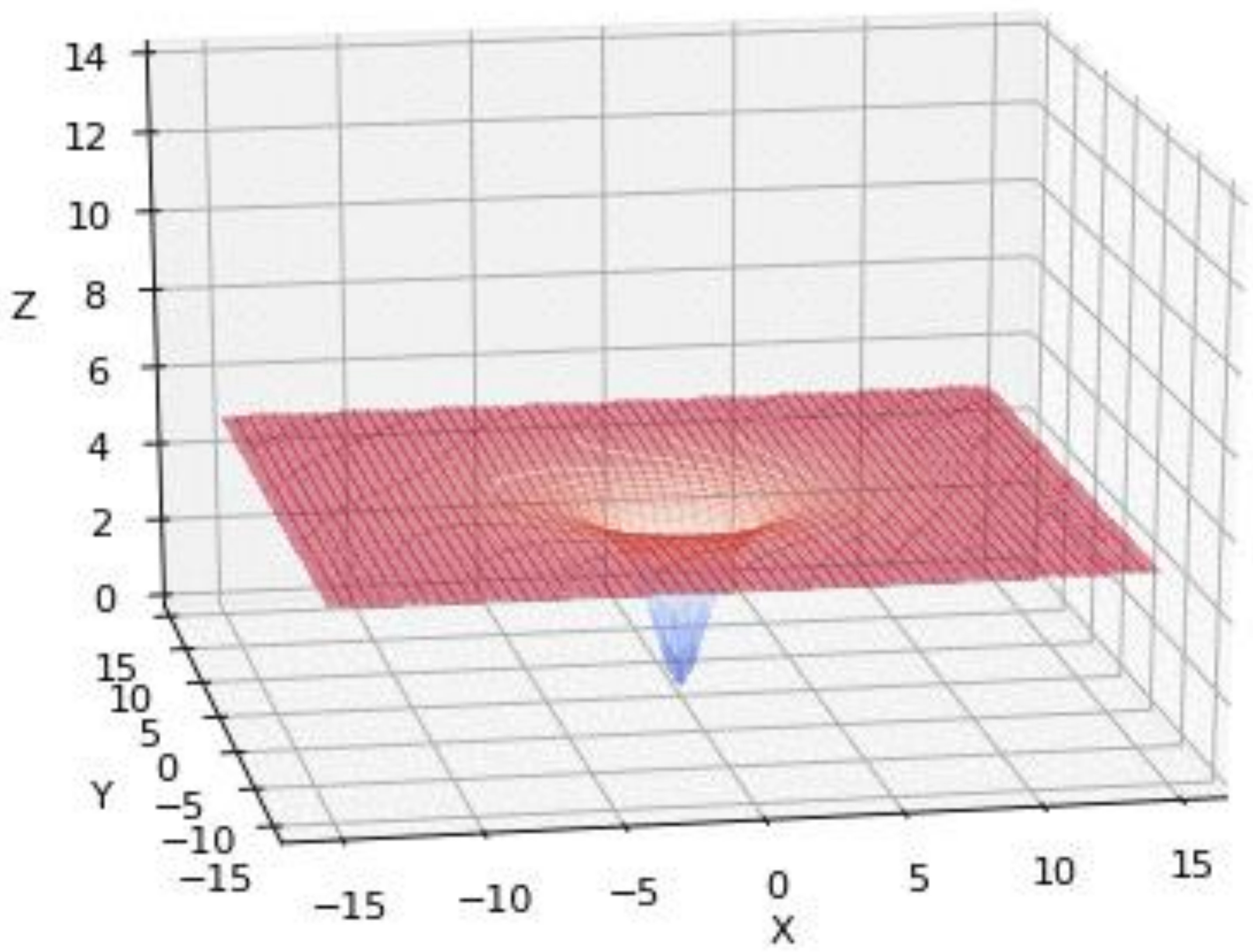


Momentum
With smaller step size, returns to local minimum

What Happens on Non-Convex Functions?

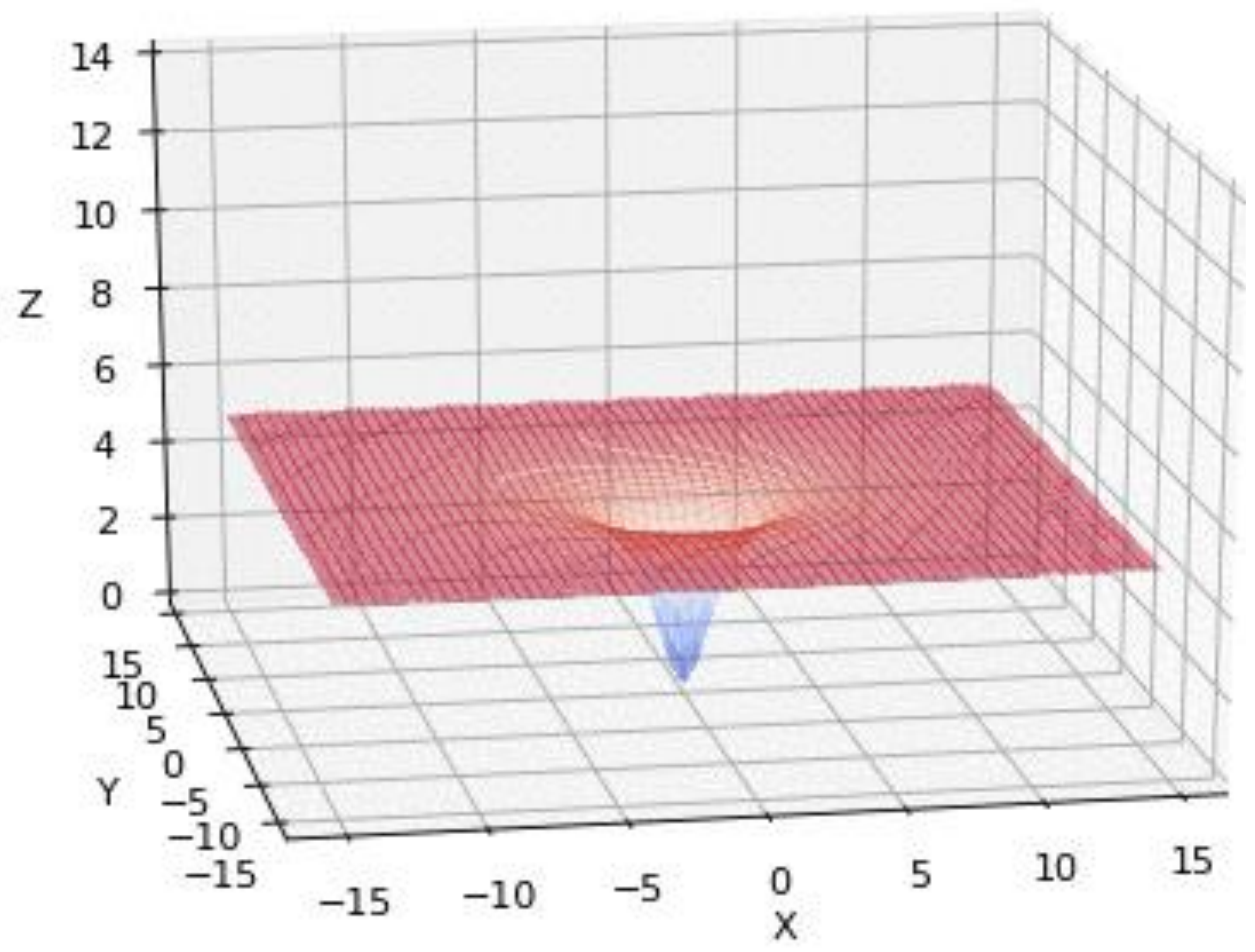


Gradient Descent
Very slow convergence due to vanishing gradients

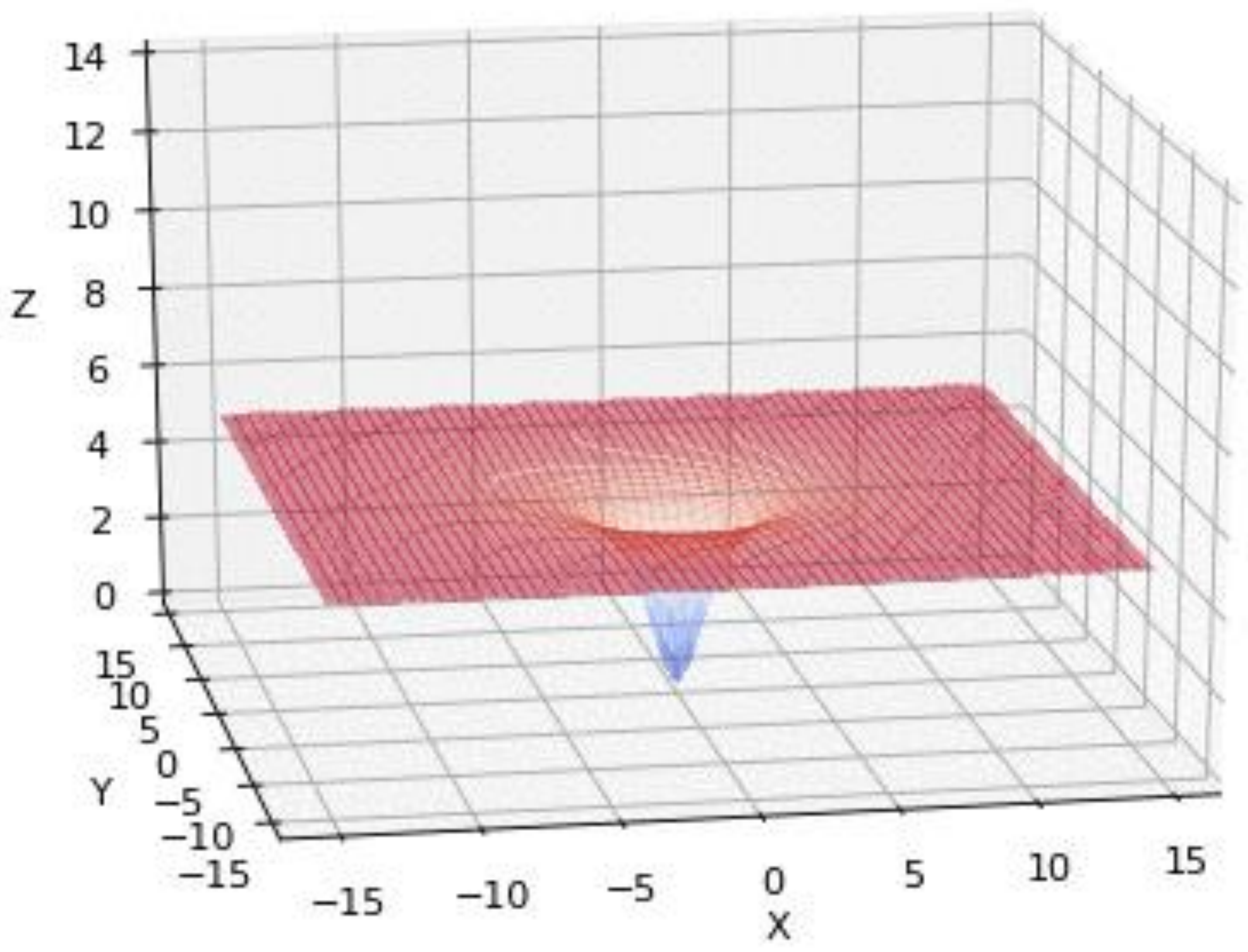


Momentum
Sometimes converges faster despite vanishing gradients

What Happens on Non-Convex Functions?

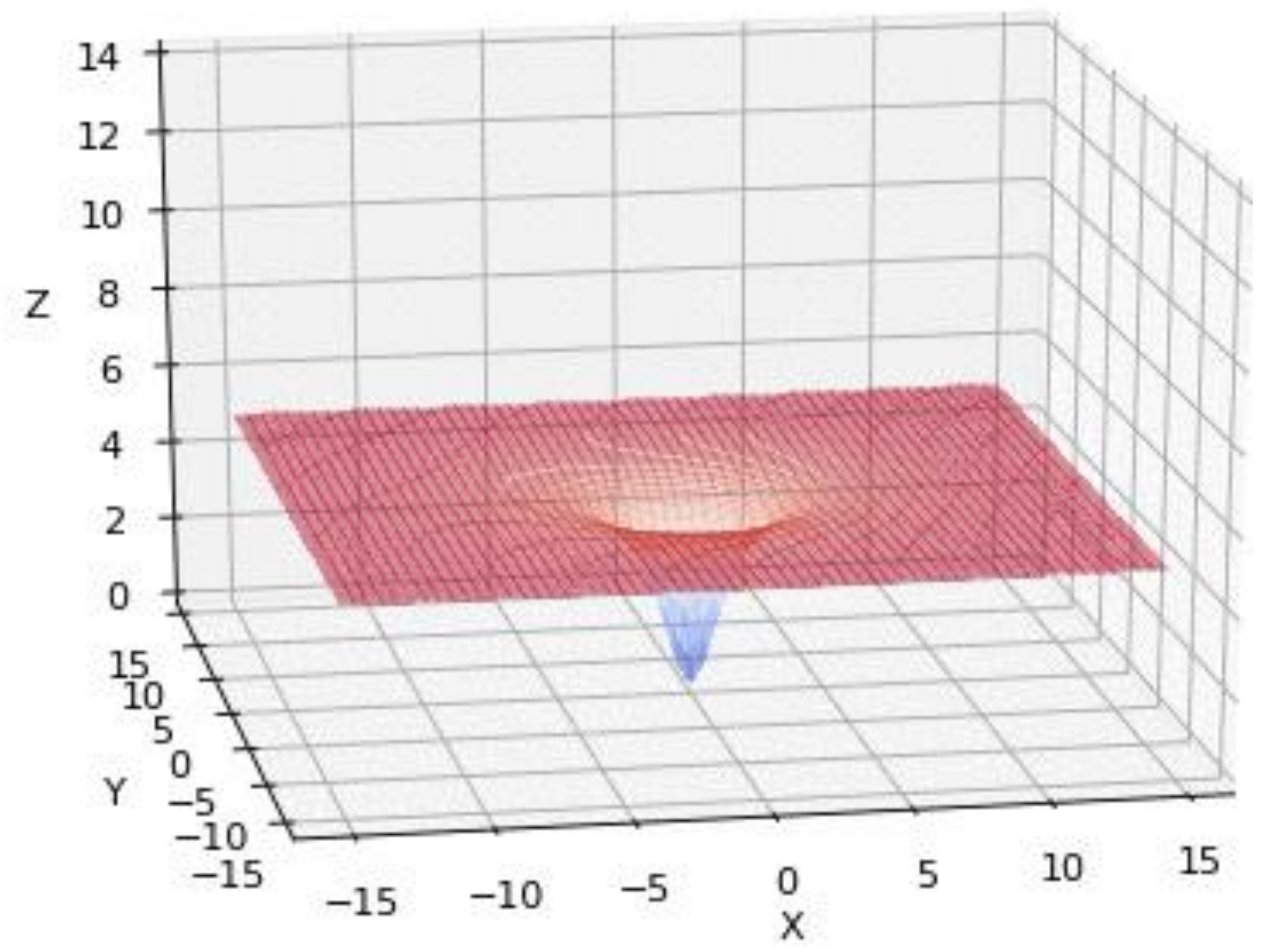


Gradient Descent
Very slow convergence due to vanishing gradients

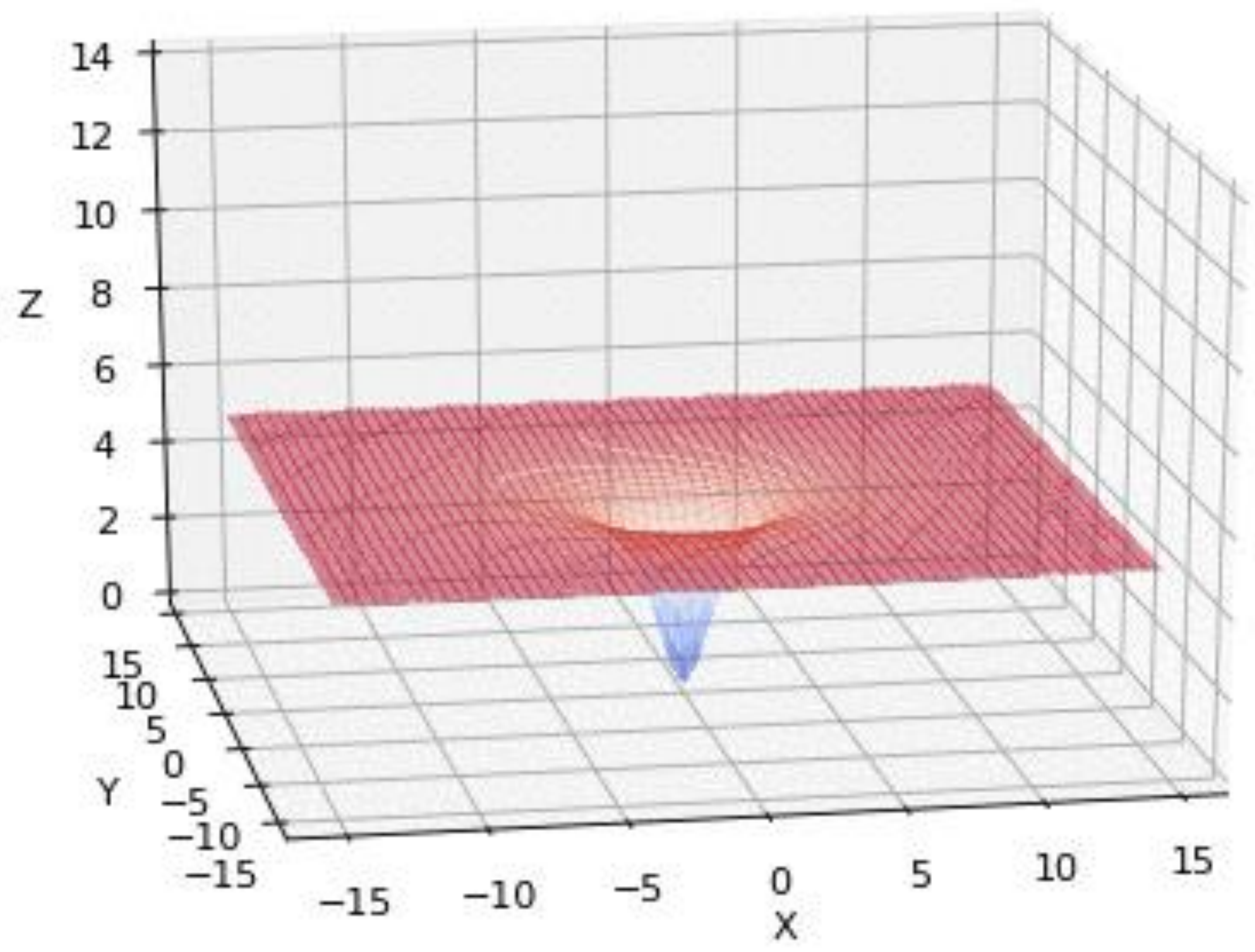


Momentum
With smaller step size, still converges slowly

What Happens on Non-Convex Functions?



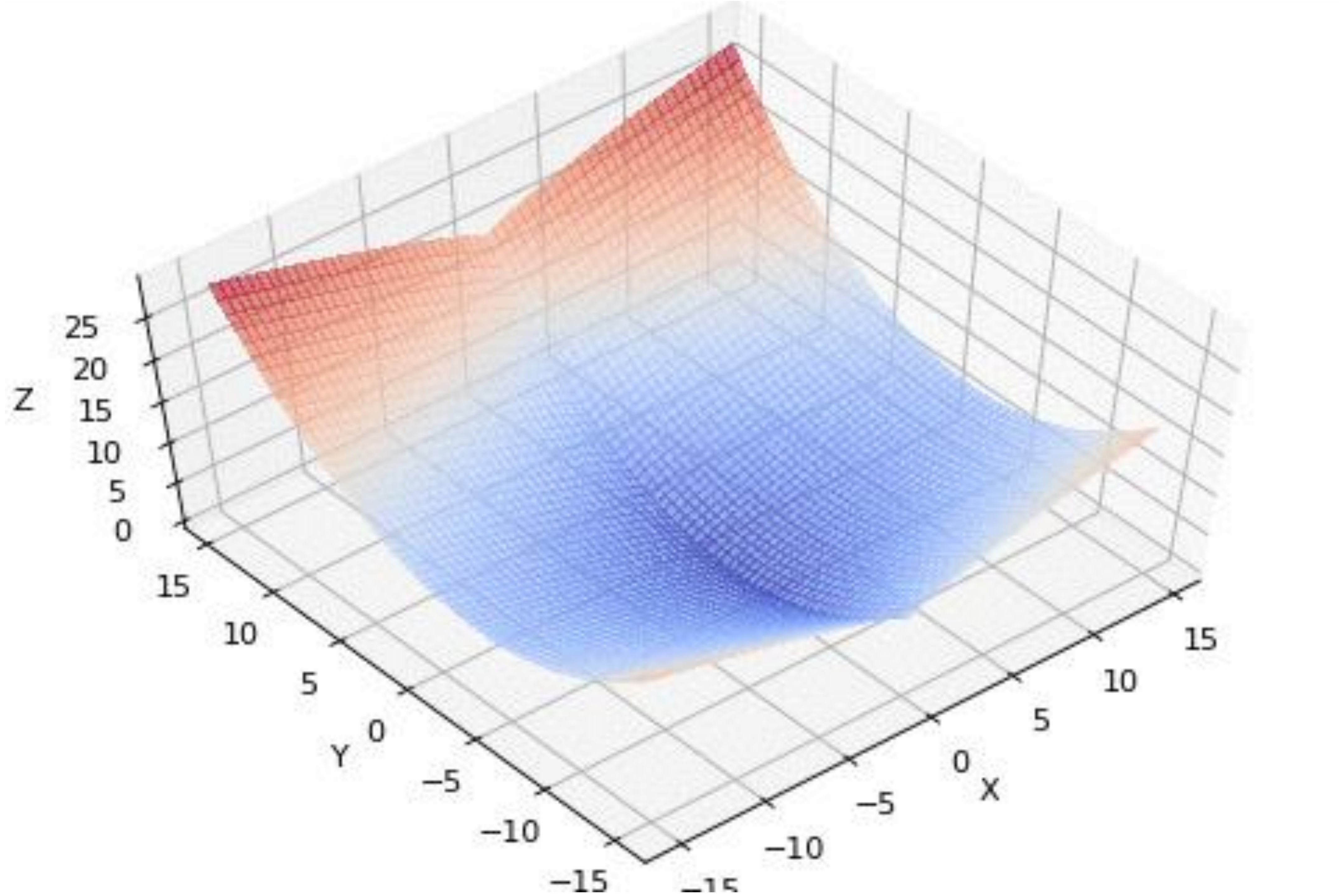
Gradient Descent
Very slow convergence due to vanishing gradients



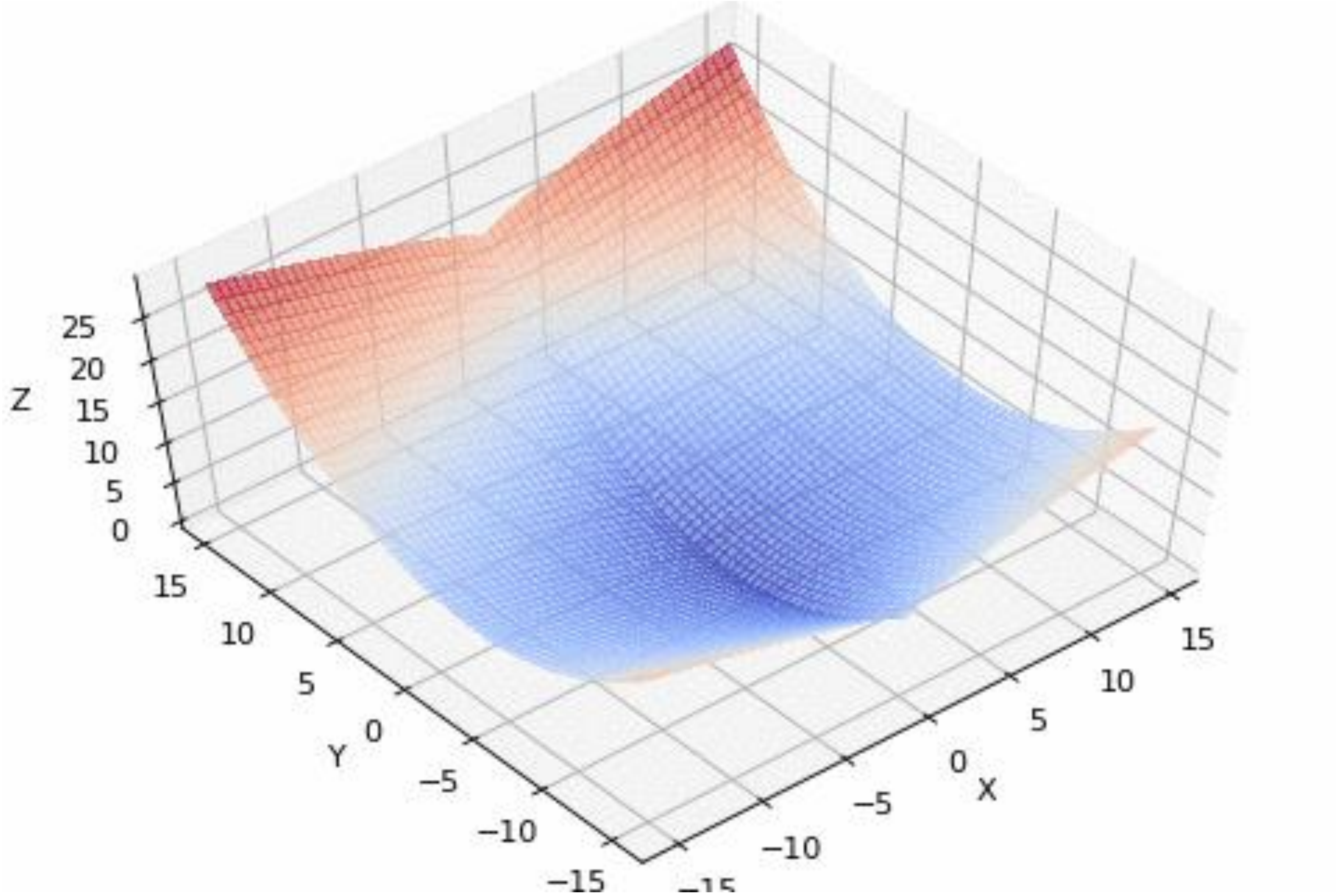
Momentum
With larger step size, can overshoot

What Happens on Non-Lipschitz Functions?

$f(x, y) = |x|^{0.8} + \frac{(y+3)^2}{15}$, so $\frac{\partial f}{\partial x}(x, y) = \text{sign}(x)\frac{0.8}{|x|^{0.2}}$ As $x \to 0$, $\left|\frac{\partial f}{\partial x}(x, y)\right| \to \infty$



Gradient Descent
Diverges



Momentum
Diverges also, albeit less severely

Ways Around Non-Lipschitzness

Apply a strictly increasing transformation to the objective function

- E.g.: The function $x \mapsto \sqrt{x}$ is non-Lipschitz. So instead of solving $\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2 = \min_{\vec{w}} \sqrt{(\vec{y} - X\vec{w})^\top (\vec{y} - X\vec{w})}$, we apply the transformation $y \mapsto y^2$) which is strictly increasing for $y \geq 0$) to the objective and solve$\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2$.

Gradient clipping

- No theoretical justification for this, but sometimes works well in practice
- $\frac{\partial L}{\partial \theta_i}\left(\vec{\theta}^{(t)}\right) \leftarrow \min\left(\left|\frac{\partial L}{\partial \theta_i}\left(\vec{\theta}^{(t)}\right)\right|, 10^3\right) \operatorname{sign}\left(\frac{\partial L}{\partial \theta_i}\left(\vec{\theta}^{(t)}\right)\right) \forall i \in \{1, \ldots, n\}$