Gradient Descent

Enbao Cao & Ayaan Dhuka

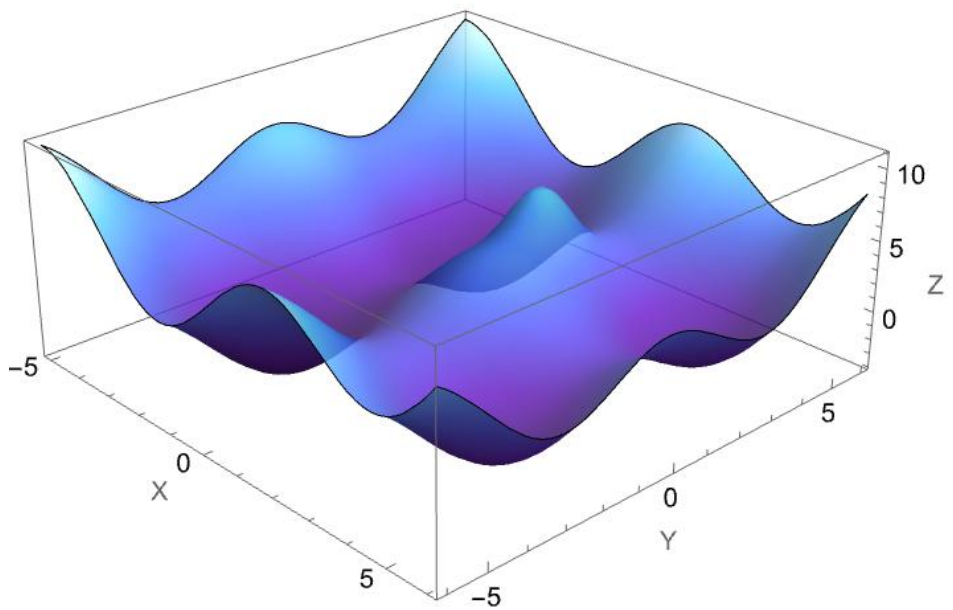
Multivariable Calculus

4/26/2024

Gradient Descent

goal: find $\min_{x,y} f(x, y)$ and the corresponding optimal $\mathbf{r}^* =< x, y >$.

moving in the direction of steepest descent, which is the negative of the gradient.



Formulation

iterative step:

$$\mathbf{r}_{n+1} = \mathbf{r}_n - \gamma \frac{\nabla F(\mathbf{r}_n)}{\left\|\nabla F(\mathbf{r}_n)\right\|}$$

$\mathbf{r}_n$: point on the n^{th} iteration, expressed as a vector.

γ: learning rate. analogous to "step size" for Euler method.

∇F: gradient of function F

Features

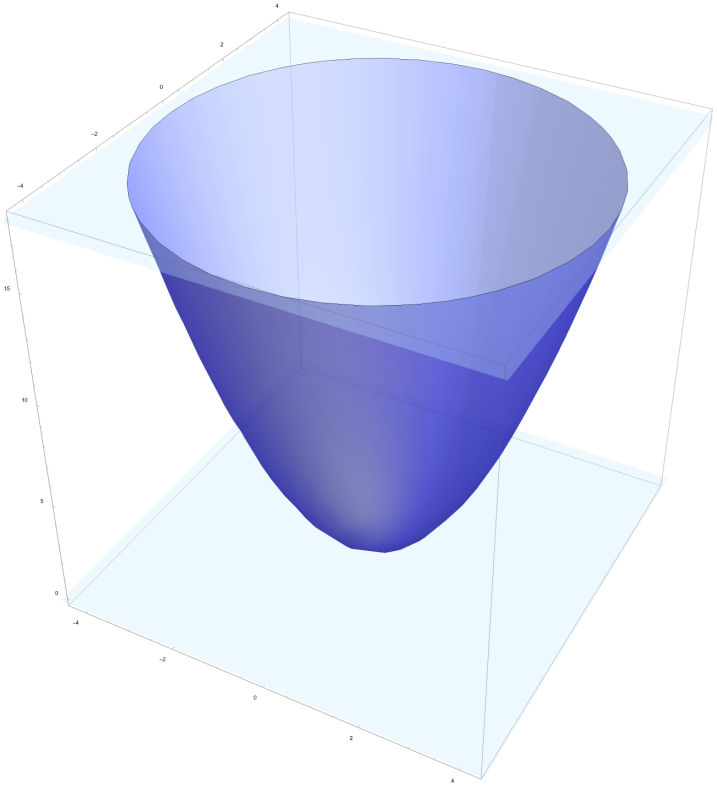
- computationally cheap iterations
- fast for convex problems (convex: only one minimum, so global guaranteed)

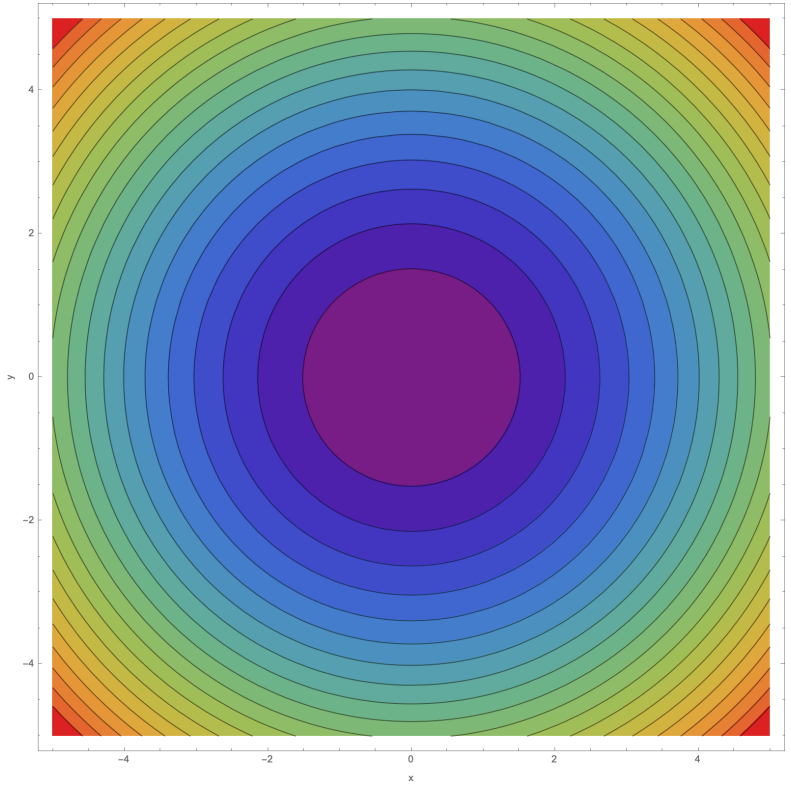
Problems

- cannot handle non-differentiable functions
- cannot guarantee a global minimum

Easy Application: Paraboloid

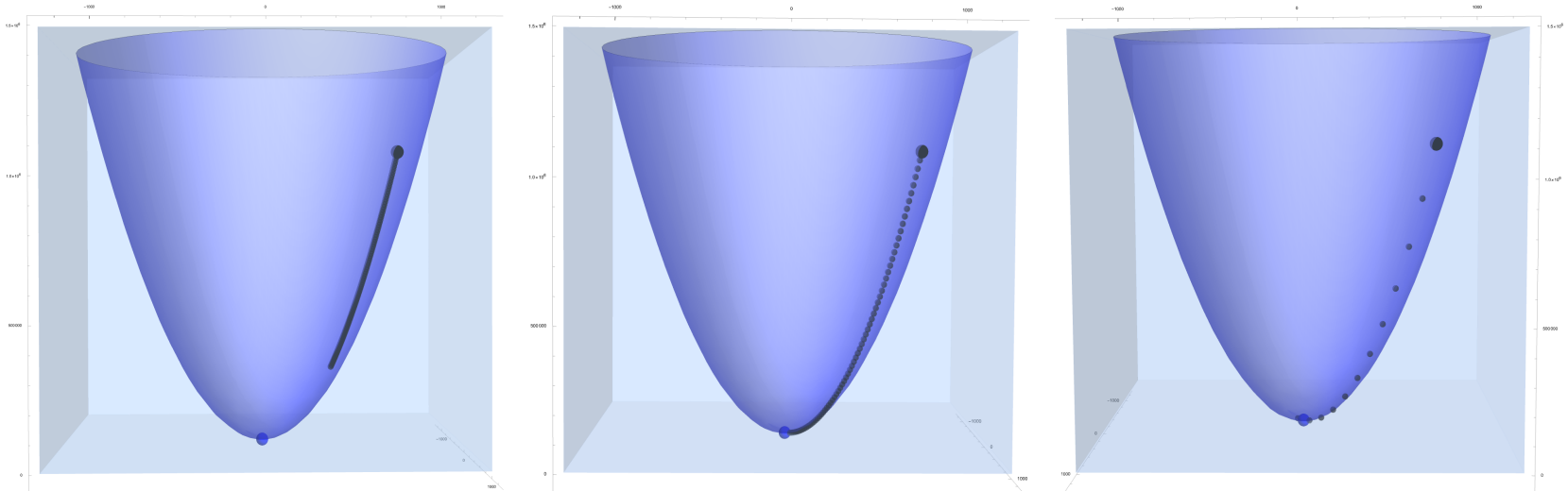
$$f(x, y) = x^2 + y^2$$





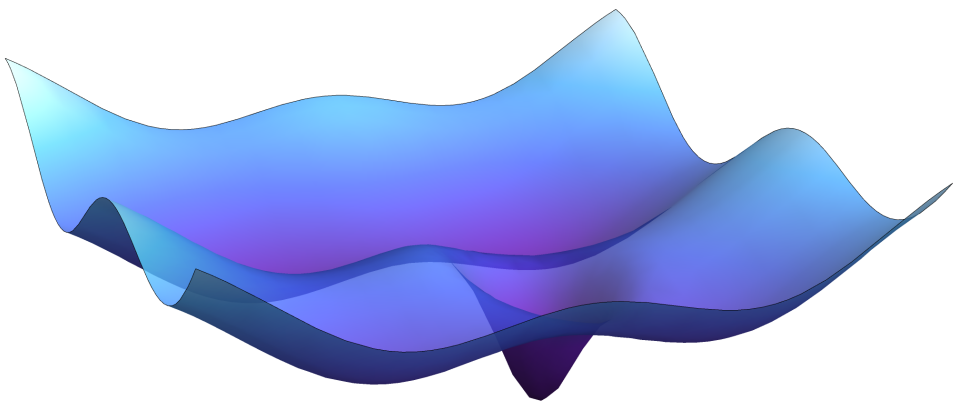
Easy Application: Learning Rates

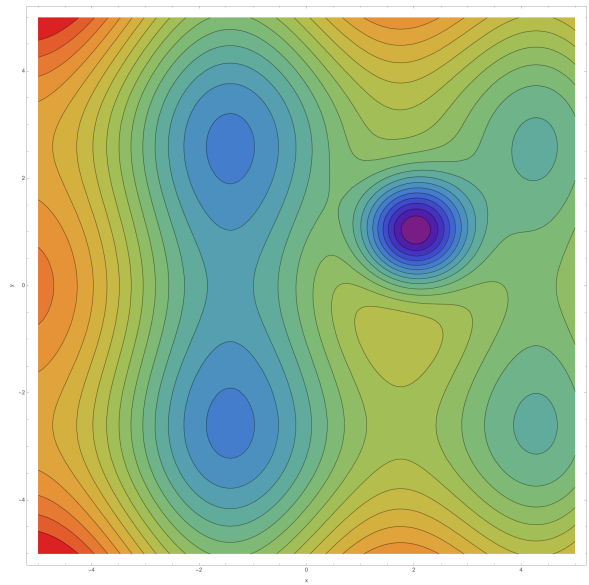
100 iterations of gradient descent were simulated.
$\gamma = 5, 25, 100$ respectively for the figures below.



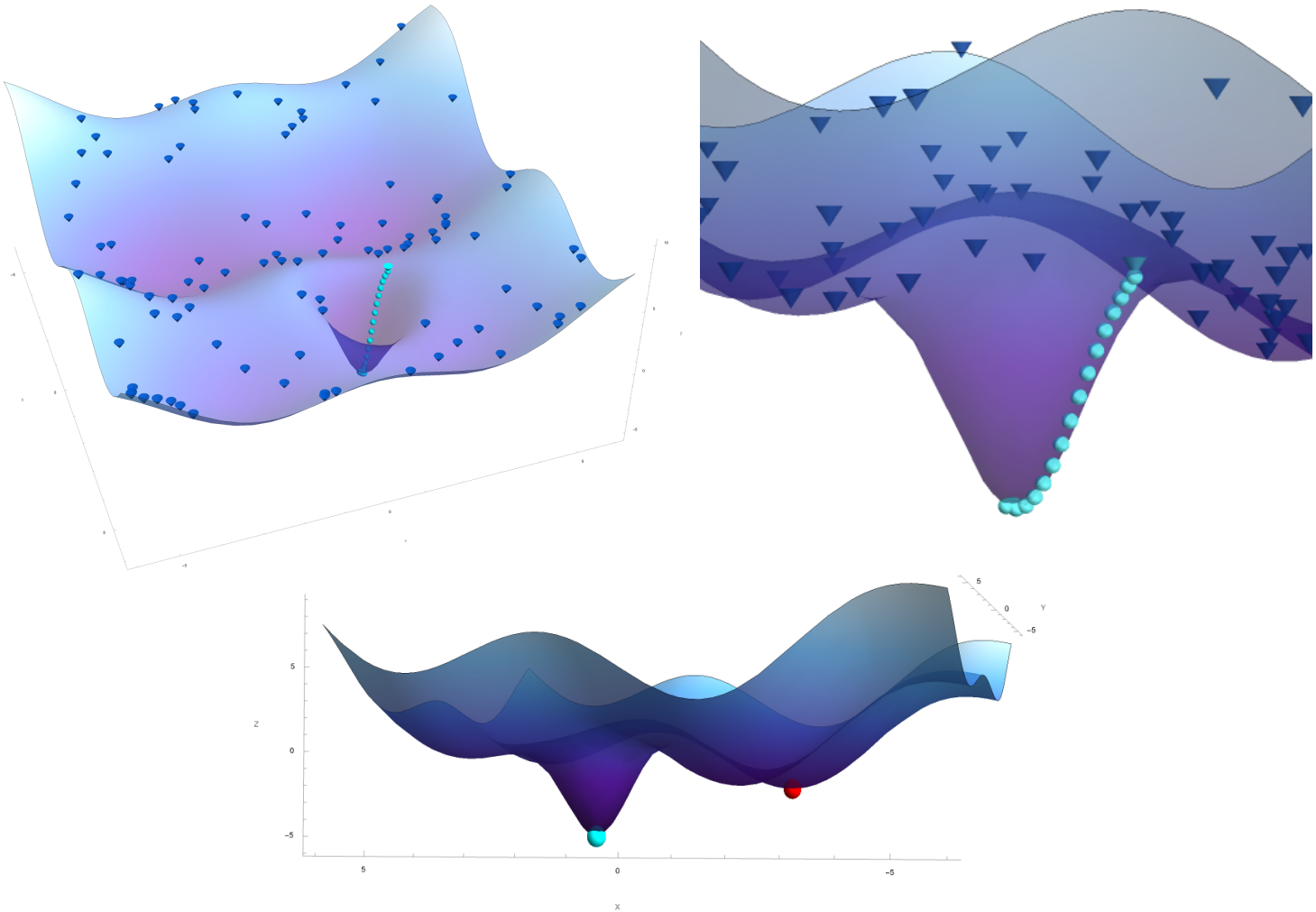
Non-Trivial Application: More Realistic Surface

$$f(x,y) = 2\sin(x) + \cos(y) - 8e^{-(x-2)^2-(y-1)^2} + 0.1(x^2+y^2)$$





Non-Trivial Application: Random Sampling



Extensions

Interactive Demo (Cut to Ayaan's App)

Wrap-up

- lots of parameters for real-world uses
- picking the right algorithm and tools

Credit

Visual aids were generated in Mathematica.

Algorithms and demo were implemented in Python.

The notebook and source code can be found on the repository.

Enbao: Presentation Visualization, Presentation Format
Ayaan: Application 1 Code, Application 2 Code, Simulation Code
Together: Presentation Content