Artificial Intelligence and Automation

Runge-Kutta Methods, Gradient Descent, and Finite Difference Approximations

Ph.D. Gerardo Marx Chávez-Campos

Instituto Tecnológico de Morelia: Ing. Mecatrónica

Introduction

- Runge-Kutta methods are iterative techniques for solving Ordinary Differential Equations (ODEs).
- Gradient Descent is an optimization algorithm for minimizing functions by following the negative gradient.
- Finite Difference Approximations provide numerical methods for computing derivatives.

Finite Difference Approximations

- Used for numerical differentiation when an explicit derivative is not available.
- Three main types:
  1. Forward Difference:
  $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$
  2. Central Difference (more accurate):
  $$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$
  3. Backward Difference:
  $$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

Example of Finite Difference Approximation

Function: $f(x) = x^2$

- Compute the derivative at $x = 2$ with $h = 10^{-5}$:
  - Forward Difference:
    $$\frac{(2 + 10^{-5})^2 - 2^2}{10^{-5}}$$
  - Central Difference:
    $$\frac{(2 + 10^{-5})^2 - (2 - 10^{-5})^2}{2 \times 10^{-5}}$$
- Exact Derivative: $f'(x) = 2x$, so $f'(2) = 4$.

Runge-Kutta Method (RK4)

Equations:

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\
k_4 &= f(t_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

How Gradient Descent Computes Derivatives

- Gradient Descent minimizes a function by iteratively updating parameters in the direction of the negative gradient.
- The gradient is computed as:

$$\nabla J(\theta) = \left(\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \ldots, \frac{\partial J}{\partial \theta_n} \right)$$

- Methods to compute gradients:
  1. Analytical differentiation (using calculus).
  2. Finite difference approximation:

$$\frac{\partial J}{\partial \theta} \approx \frac{J(\theta + h) - J(\theta)}{h}$$

  3. Automatic differentiation (used in deep learning frameworks like TensorFlow and PyTorch).

Gradient Descent Update Rule

- Once the gradient $\nabla J(\theta)$ is computed, parameters are updated as:

$$\theta := \theta - \alpha \nabla J(\theta)$$

- Where:
  - α is the learning rate.
  - $\nabla J(\theta)$ is the gradient of the loss function.

Conclusion

- Runge-Kutta methods are widely used for solving ODEs numerically.
- Gradient Descent is crucial for optimization in machine learning and deep learning.
- Finite Difference Approximations are useful for numerical differentiation.
- These numerical methods are applied in physics, engineering, and artificial intelligence.

Referencias

- Numerical Analysis by Burden & Faires.
- Numerical Recipes in C by Press et al.
- Deep Learning by Goodfellow, Bengio, and Courville.