Taylor Series and Applications in ML

Hongtao Li

Karina Bakhtiyarova

Outline

1. Introduction to Taylor Series
2. Theoretical Foundations of Taylor Series
3. Applications in Machine Learning
4. Applications in Deep Learning
5. Implementation in Python

Introduction to Taylor Series

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

A Taylor series is a method to express a function using an endless series centered around a specific point. It is named after mathematician Brook Taylor, who developed it. The formula for a Taylor series of a function f(x) centered at a is a versatile form that can be utilized to estimate the function.

Historical context of Taylor Series

- ***Ancient Roots***
- ***Madhava and Kerala School***
- ***Renaissance Europe***
- ***Brook Taylor***

- The Taylor series has since become a cornerstone of mathematical analysis, providing a powerful tool for approximating and studying functions in various mathematical contexts.

Importance and applications

- ***Function Approximation***
- ***Error Analysis***
- ***Derivatives and Integrals***
- ***Physics and Engineering***
- ***Probability and Statistics***
- ***Machine Learning and Optimization***
- ***Signal Processing***

Theoretical Foundations of Taylor Series

A POLYNOMIAL APPROXIMATION IS SIMPLY AN ESTIMATE OF A CURVE USING A POLYNOMIAL FUNCTION.

Basic concepts: Polynomial approximation

The Taylor series is a mathematical tool that allows us to approximate functions using polynomials. Specifically, it represents a function $f(x)$ as an infinite sum of terms involving powers of $(x - a)$, where a is a point around which we want to approximate the function.

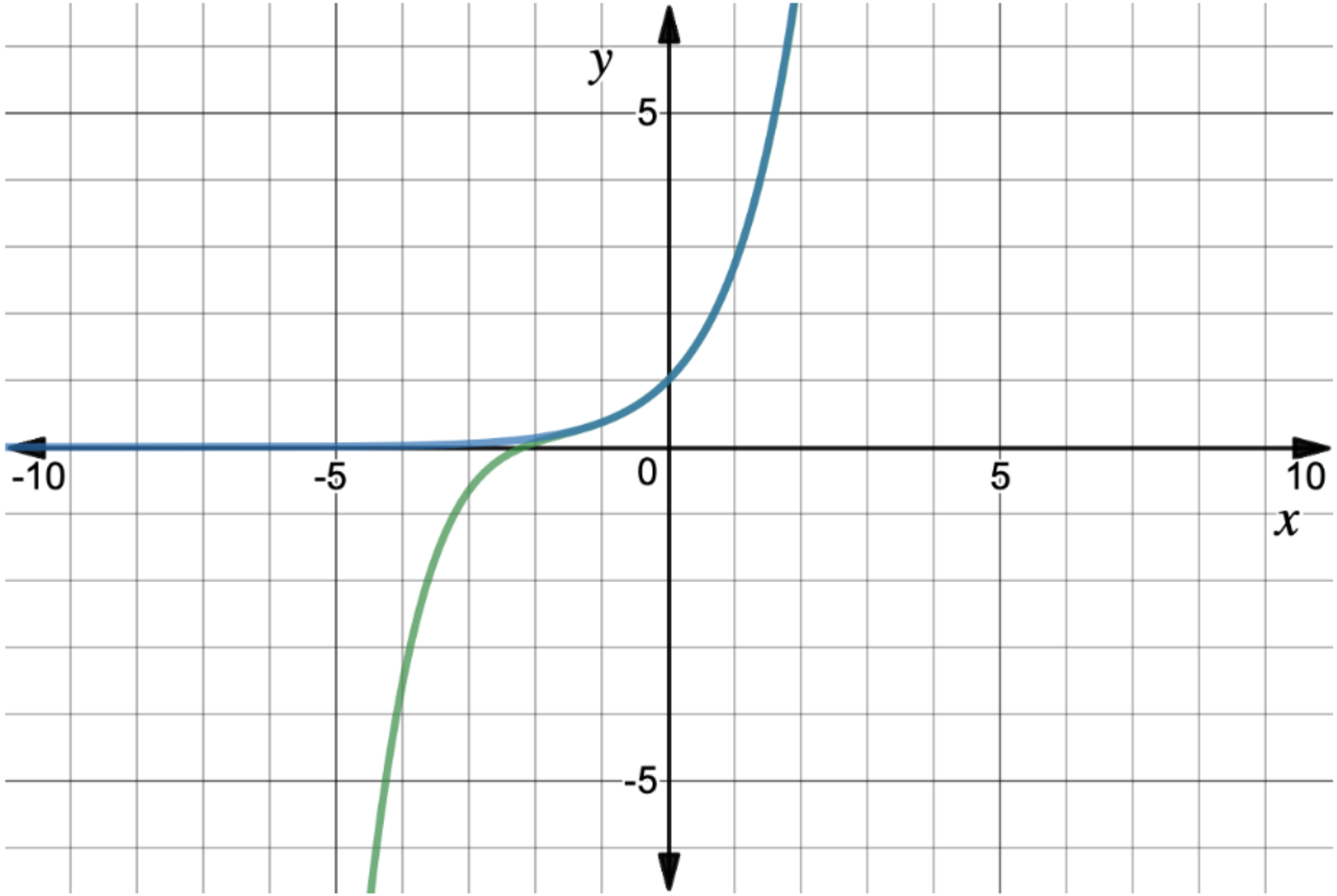
The basic idea is to start with a known function $f(x)$ and its derivatives at the point $x = a$. These derivatives give us information about how the function changes at that point. The Taylor series then constructs a polynomial that matches the function and its derivatives up to a certain order near $x = a$.

The general form of the Taylor series for a function $f(x)$ around $x = a$ is:

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \cdots$$

Here, $f'(a)$, $f''(a)$, $f'''(a)$, and so on, represent the derivatives of $f(x)$ evaluated at $x = a$. The terms $(x - a)^n$ are powers of the difference between x and a, which capture how the function changes as x moves away from a.

We have the curve $f(x) = e^x$ in blue, and a Polynomial Approximation with equation $g(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$ in green.



Derivation of Taylor Series

- **Maclaurin Series:** Start with a function $f(x)$ and assume it has derivatives of all orders at $x=0$. The Maclaurin series of $f(x)$ is given by:

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots$$

- **General Taylor Series:** To derive the Taylor series for a function f(x) around a point a, we use a similar approach but with x−a as the variable. Let h=x−a, so x=a+h. Rewrite the Maclaurin series using h:

$$f(a+h) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \frac{f'''(a)}{3!}h^3 + \cdots$$

This series represents $f(x)$ in terms of h (which is $x - a$) around $x = a$. It's called the Taylor series of $f(x)$ about $x = a$.

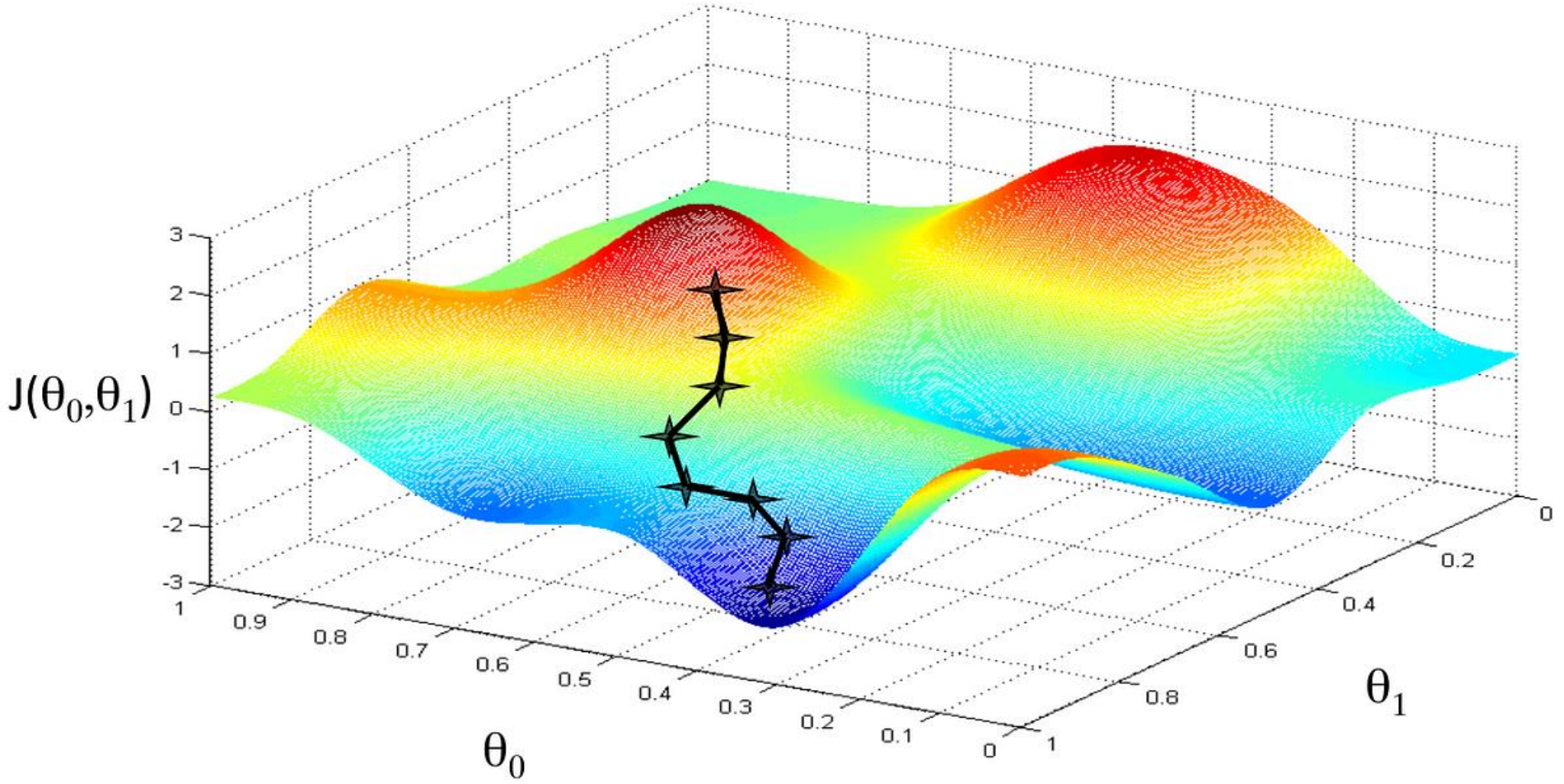
- **Simplification:** Often, we rewrite the Taylor series using sigma notation for brevity:

$$f(a+h) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}h^n$$

Here, $f^{(n)}(a)$ denotes the n-th derivative of $f(x)$ evaluated at $x = a$.

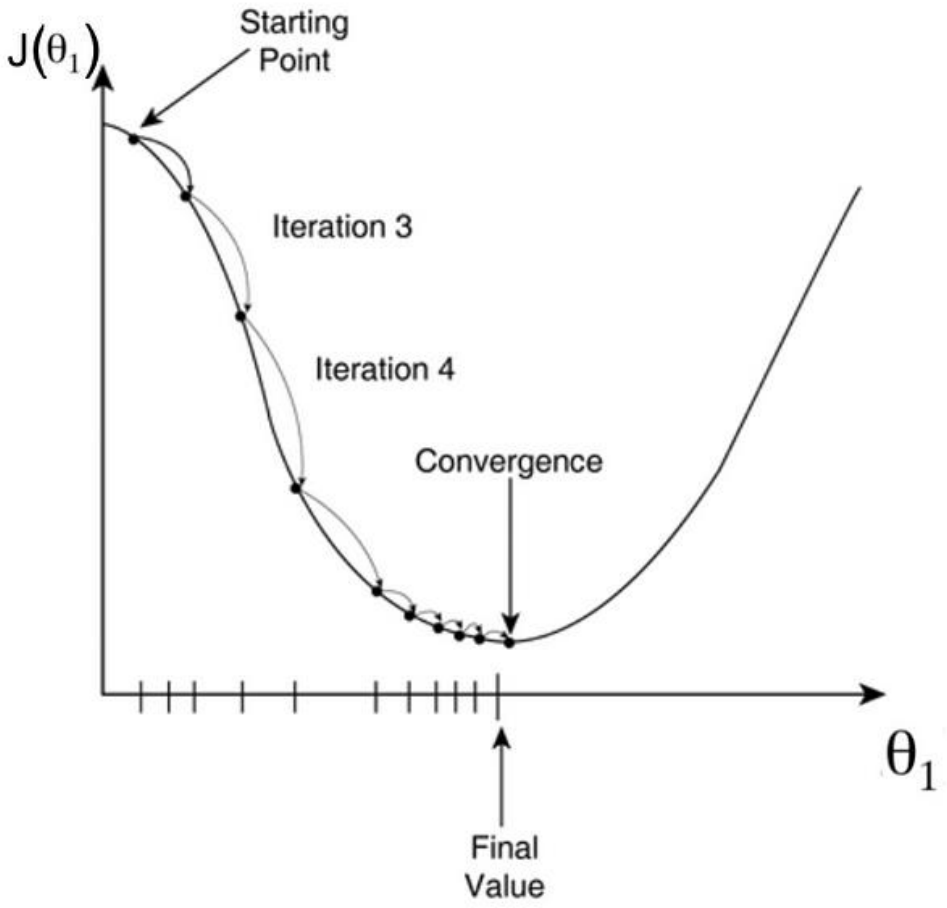
Applications in Machine Learning

Gradient Descent Optimization



Jaleel Adejumo (2023)

Gradient Descent Optimization



Cost Function – "One Half Mean Squared Error":

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Derivatives:

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$

Taylor Series Imitation Learning

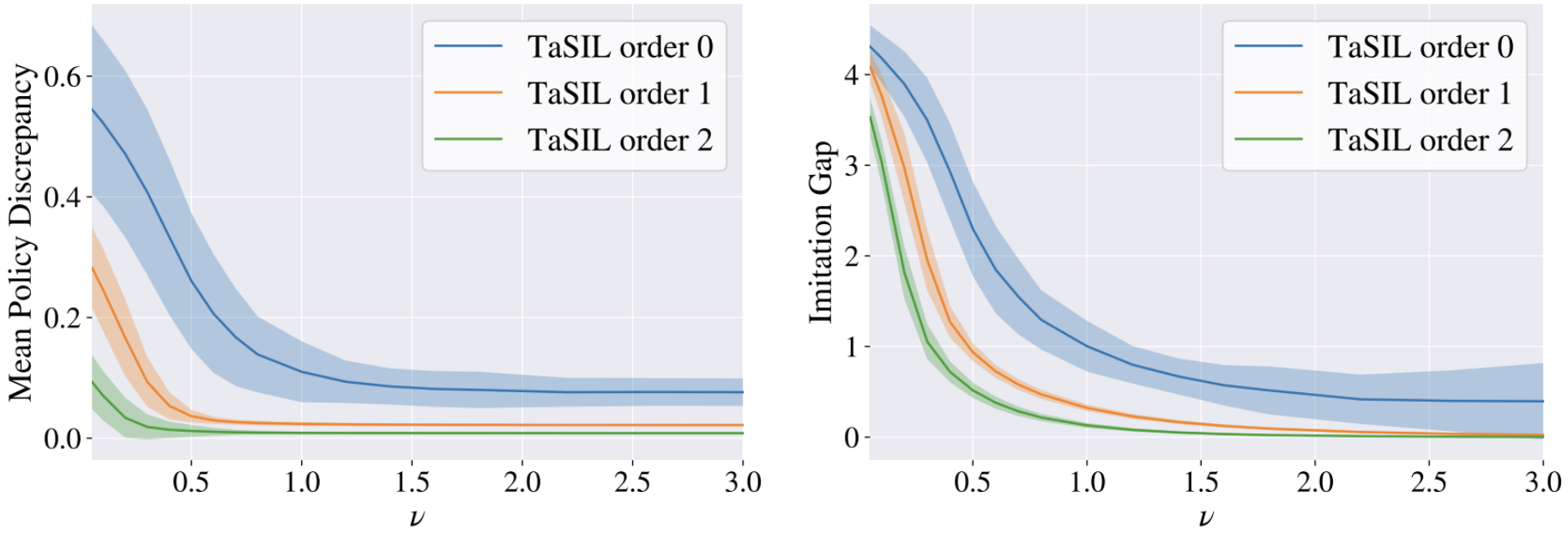


Figure 1: Left: The average Euclidean norm difference between the expert and learned policies on trajectories rolled out under the learned policy. **Right:** The maximum deviation between an expert and learned policy trajectory starting from identical initial conditions. All statistics are averaged across 50 test trajectories and we plot the mean and standard deviation for 10 random seeds.

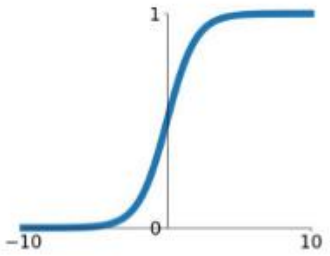
Pfrommer, D., Zhang, T., Tu, S. & Matni, N. (2023)

Applications in Deep Learning

Activation Functions Optimization

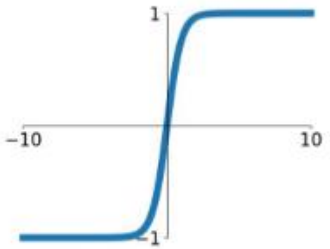
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



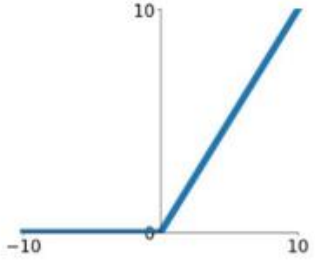
tanh

$$\tanh(x)$$



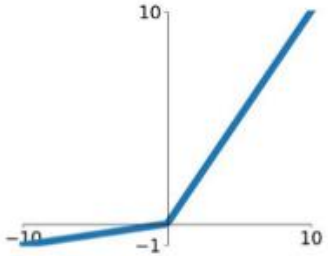
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

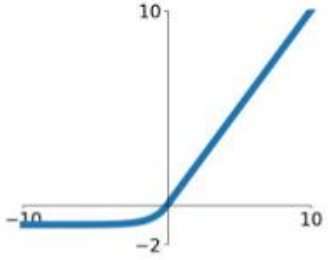


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

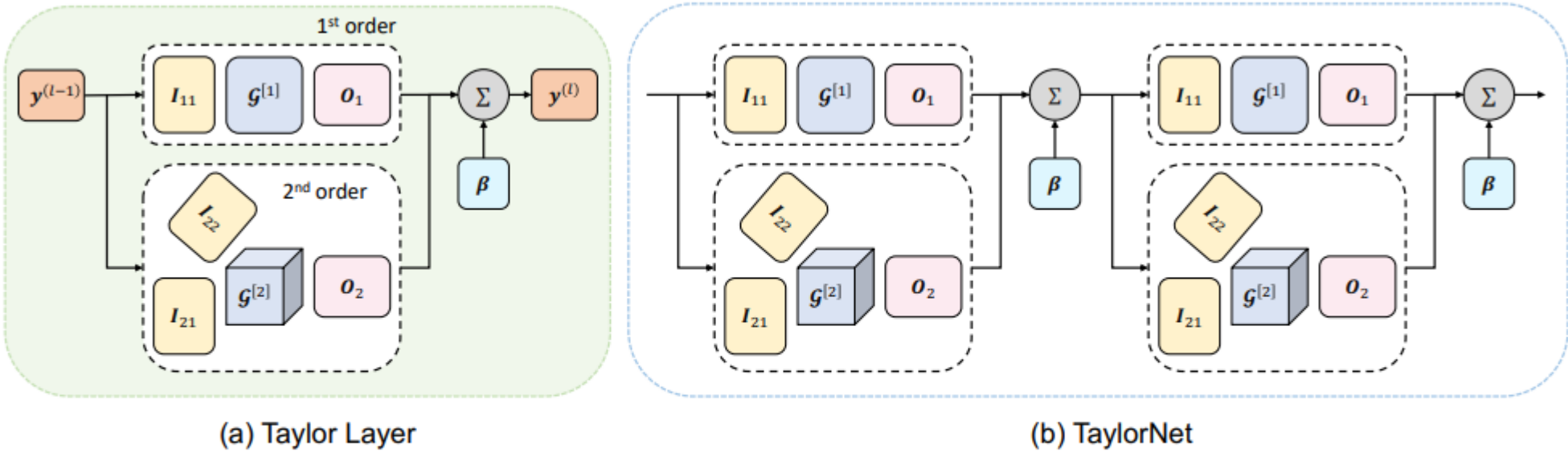
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Shruti Jadon (2018)

Taylor Neural Network (TaylorNet)



(a) Taylor Layer

(b) TaylorNet

Figure 1: (a) Architecture of Taylor Layer of order 2 using Tucker decomposition; (b) TaylorNet consists of N Taylor layers of order 2.

Zhao H. et, al. (2023)

Implementation in Python