KAN Read notes

October 16, 2024

Background

Comparison between KAN and MLPs(should pull back)

1. 1. MLPs, also known as fully connected feedforward neural networks (**Have a fixed activation function on the node**).
   Inspiring by universal approximation theorem, MLPs can approximate any function.
   MLP (Multilayer Perceptron) is a type of deep neural network (DNN), consisting of multiple layers: input, hidden, and output.

   - **Input Layer**: Receives sample data, where each neuron represents a feature of the input data.
   - **Hidden Layers**: One or more layers with neurons, performing nonlinear transformations via activation functions to capture features and patterns in the input data.
   - **Output Layer**: Produces the network's output, designed based on the task—classification typically uses a softmax activation, while regression may use linear or no activation.

   In Transformer architectures, after each self-attention layer, there is an MLP block, also known as a Feed Forward Network (FFN), which adds computational power and expressive capacity to the model, enabling it to handle various tasks such as natural language processing, image recognition, and time series analysis.

2. 2. KANs : parametrize each 1D function as a B-spline curve, pervisouly research most stuck owith the 2-layer KAN with shape [n, 2n+1], and did not use the other techniques like back proprogation.

   This paper expand the KAN to multiple layers (any depth and width)

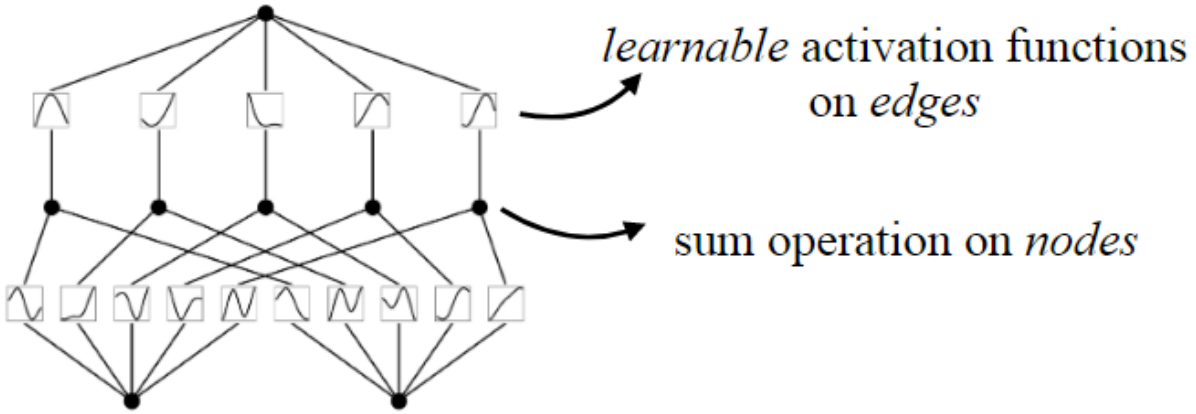
learnable activation functions
on *edges*
sum operation on *nodes*

Figure 1: KAN

Technical background

1. : Kolmogorov-Arnold THM: **Any continuous function can be represented as a sum of continuous functions of a single variable.** To some extent, they showed that the only true multivariate function is addition, since every other function can be written using univariate functions and sum.
   Given the function:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

   where:

   - $\phi_{q,p}$ are continuous one-dimensional functions mapping each variable to the real number line.
   - Φ_q are continuous one-dimensional functions, and the final result is the sum after processing.
   - The minimal number of functions is used to ensure that any continuous function can be represented in this form.

   For the example function:

$$f(x, y, z) = x^2 + 2y^3 + \sin z$$

   we can find a set of one-dimensional functions such that:

$$f(x, y, z) = \sum_{q=1}^{7} \Phi_q \left(\sum_{p=1}^{3} \phi_{q,p}(x_p) \right)$$

   where $x_1 = x$, $x_2 = y$, $x_3 = z$. We can choose the following function forms:

$$\begin{aligned} &\phi_{1,1}(x) = x^2, \quad \phi_{1,2}(y) = 0, \quad \phi_{1,3}(z) = 0, \\ &\phi_{2,1}(x) = 0, \quad \phi_{2,2}(y) = y^3, \quad \phi_{2,3}(z) = 0, \\ &\phi_{3,1}(x) = 0, \quad \phi_{3,2}(y) = 0, \quad \phi_{3,3}(z) = \sin z, \end{aligned}$$

   and the remaining functions can be defined as zero since they do not affect the final sum.

   Accordingly, the functions Φ_q can be selected as:

$$\Phi_1(u) = u, \quad \Phi_2(u) = 2u, \quad \Phi_3(u) = u$$

2. Different spline: A B-spline function (a type of bezier) is a combination of flexible bands that is controlled by a number of points that are called control points, creating smooth curves

   Recall our CS371 or numerical analysis: Spline functions are typically defined by a series of low-degree polynomials over different segments, with these polynomials being continuous at the connection points (called knots) and having continuous derivatives. The degree of these polynomials determines the smoothness of the spline. Common examples include linear splines (degree 1), quadratic splines (degree 2), and cubic splines (degree 3). Cubic splines are the most widely used because they strike a good balance between smoothness of the curve and computational complexity, providing an effective fit.

Getting into KANs

1. Structure: **Insert and Explain formula from 2.2 - 2.8, example 1.1**
   A nice image illustrate the difference between MLP and KANs(need some adjustment)

   - **MLP**: Linear combination followed by nonlinear activation.
   - **KAN**: Nonlinear activation for each input followed by linear combination.

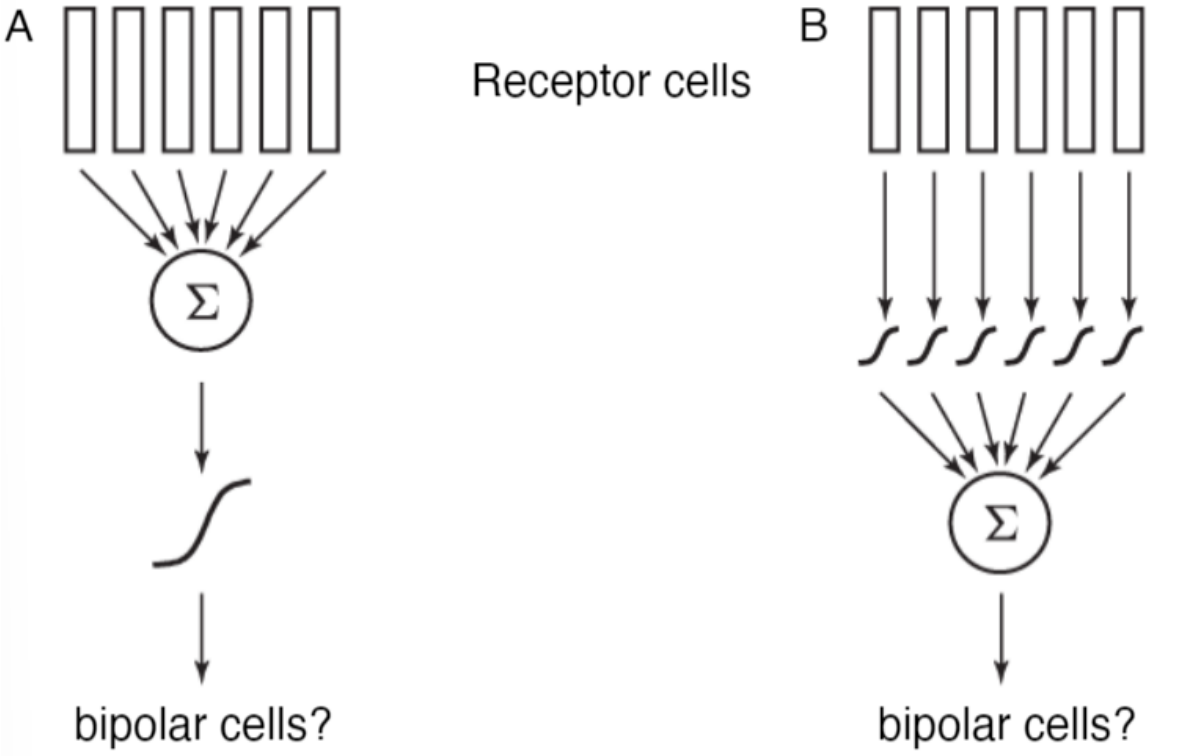

Figure 9: Two models of integrating single photon signals at the first synapse in the retina [57]. (A) Receptor cell signals are summed, and the target (bipolar) cell applies a nonlinearity, as in conventional neural network models. (B) Each receptor cell signal is passed through a nonlinearity, as suggested by Eq (20), and the resulting estimates of photon count are summed.

Figure 2: Not actually but its same thing https://x.com/ChongwenWang/status/1786900136163160308

2. My own assumption, or guessing: KANs = splines (accurate in low-dimensional functions) + MLPs (capable of learning composite structures)

   In essence, KANs are a combination of splines and MLPs, leveraging their respective strengths:

   - **Splines** are accurate for low-dimensional functions, easy to adjust locally, and can switch between different resolutions. However, splines struggle with high dimensions and cannot exploit composite structures.
   - **MLPs**, on the other hand, are less affected by dimensionality issues (thanks to feature learning) but are less accurate than splines in low dimensions and unable to optimize univariate functions.

   KANs benefit from:

- **Internal splines**: Splines have internal degrees of freedom (between knots) but lack external flexibility. KANs optimize learned features to a high level of accuracy, similar to splines in approximating univariate functions.
- **External MLPs**: MLPs have external degrees of freedom but no internal flexibility. KANs not only learn features (similar to MLPs externally) but also learn the composite structure of multiple variables.

Thus, KANs combine feature learning from MLPs and precise univariate approximation from splines, offering the best of both worlds.

3. COD (Curse of Dimensionality): Although MLPs were initially designed to handle high-dimensional data, this design can be inefficient for low-dimensional tasks, leading to resource waste. For instance, a complex MLP intended for handling hundreds of dimensions may be overpowered and computationally inefficient when working with tasks having only a few input features.

4. Grid externsion: for accuracy:

0.1 Fitting a Fine-Grained Spline onto a Coarse-Grained Spline

In principle, splines can be made sufficiently accurate to approximate the target function by refining the grid. This advantage is inherited by KANs.

0.1.1 MLPs Lack a "Refinement" Concept

While increasing the width and depth of MLPs can improve performance (as suggested by the neural scaling laws), these improvements are slow and expensive, requiring separate training of models with different sizes.

For KANs, however, one can first train a KAN with fewer parameters and then expand it into a larger KAN by simply refining the spline grid without needing to retrain a larger model from scratch.

0.1.2 Grid Expansion

Next, we describe how to perform grid expansion, the proceduce is describe in th Figure 2.2:, add those explanation on 2.16

5. Simplification: technique

Properties of Kans

1. Interpretable: examples of Kans on calculations:
2. Continual Learning: Avoid catastrophic forgetting: Potiental for LLM
3. accuracy: example on PDE,

Ans: Solution: A3.