Kolmogorov-Arnold Networks application to NLP

Course: Natural Processing Language
Prof. Da San Martino

Roccatello Mattia
ID: 2131065

Milan Lisa
ID: 2103674

Ozen Aysenur
ID: 2107501

Marzola Devis
ID: 2097100

Gugole Mattia
ID: 2130343

June 5, 2024

Contents

1 Introduction

A new machine-learning algorithm, Kolmogorov-Arnold Networks (KANs), promises to be an alternative to Multi-Layer Perceptron Networks (MLPs). This new approach comes from a paper that appeared on arXiv in April 2024, titled: "KAN: Kolmogorov-Arnold Networks."

Like MLPs, KANs have fully-connected structures. However, while MLPs place fixed activation functions on nodes (neurons), KANs place learnable activation functions on edges (weights). As a result, KANs have no linear weight matrices at all; instead, each weight parameter is replaced by a learnable one-dimensional function parametrized as a spline. KANs nodes simply sum incoming signals without applying any non-linearities. As reported in the original paper, no experiments have been conducted on the application of KANs on NLP related tasks. Our aim is to dive into this new topic and to perform comparisons on the two models.

2 Kolmogorov-Arnold Networks

2.1 Kolmogorov-Arnold Representation Theorem

The "*nouvelle*" model is based on the Kolmogorov-Arnold Representation Theorem, which states that any multivariate continuous function on a bounded domain can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. More specifically, for a smooth $f : [0,1]^n \to \mathbb{R}$:

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

where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$.

At the same way MLPs are inspired by the universal approximation theorem, KANs are inspired by the Kolmogorov-Arnold representation theorem. This theorem shows, in a sense, that the only true multivariate function is addition, since every other function can be written using univariate functions and sum. In simpler terms, it's like saying that even when dealing with a complicated equation or a machine learning task where the outcome depends on many factors, we can break it down into smaller, easier-to-handle parts. We focus on each factor individually and then put everything together to solve the bigger problem.

Since all functions to be learned are univariate functions, we can parametrize each 1D function as a B-spline curve, with learnable coefficients of local B-spline basis functions, which allow us to maintain smoothness and to facilitate differentiability.

2.2 KANs architecture

The Kolmogorov-Arnold representations correspond to two-layer KANs, with n input units, $2n+1$ units in the hidden layer and 1 output unit. In order to have deeper KANs we can stack more layers, following the same reasoning behind MLPs. A general KAN network is a composition of L layers: given an input vector $\mathbf{x}_0 \in \mathbb{R}^{n_0}$ the output is

$$\text{KAN}(\mathbf{x}_0) = (\mathbf{\Phi}_{L-1} \circ \mathbf{\Phi}_{L-2} \circ \ldots \circ \mathbf{\Phi}_1 \circ \mathbf{\Phi}_0)\, \mathbf{x}_0$$

where $\mathbf{\Phi}_\ell$ is the function matrix corresponding to the ℓ-th KAN layer.

$$\mathbf{x}_{\ell+1} = \mathbf{\Phi}_\ell \mathbf{x}_\ell = \begin{pmatrix} \phi_{\ell,1,1}(\cdot) & \cdots & \phi_{\ell,1,n_\ell}(\cdot) \\ \vdots & \ddots & \vdots \\ \phi_{\ell,n_{\ell+1},1}(\cdot) & \cdots & \phi_{\ell,n_{\ell+1},n_\ell}(\cdot) \end{pmatrix} \begin{pmatrix} x_{\ell,1} \\ \vdots \\ x_{\ell,n_\ell} \end{pmatrix} \quad \ell = 0, \ldots, L-1$$

where with $\phi_{\ell,j,i}$ we denote the activation function, which has trainable parameters, that connects the i-th unit in the ℓ-th layer with j-th unit in the $(\ell+1)$-th layer. Between layer ℓ and layer $\ell+1$, there are $n_\ell n_{\ell+1}$ activation functions in total.

2.3 Training KANs

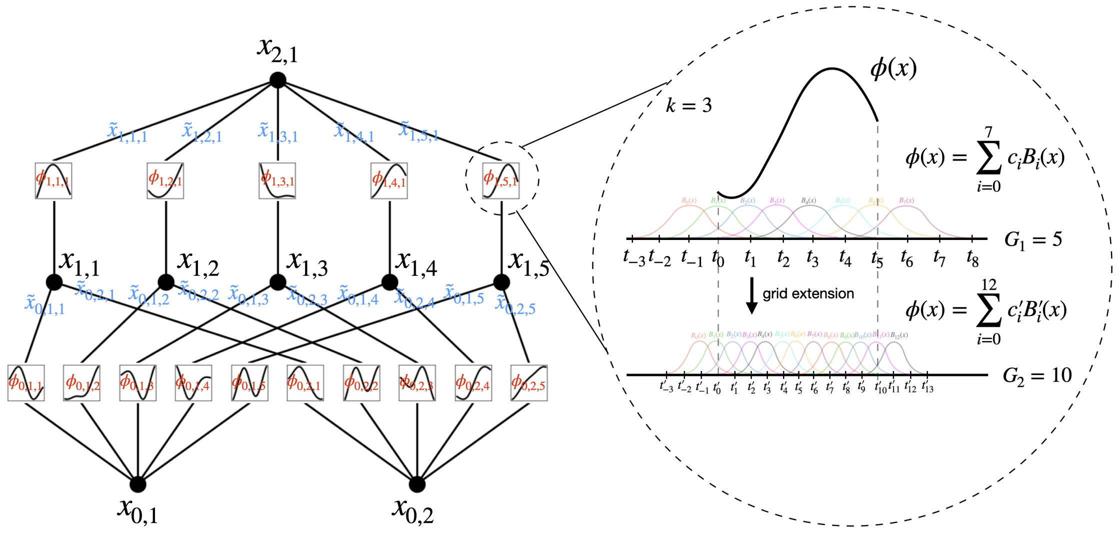
Notice that all the operations involved in the KAN architecture are differentiable, allowing us to train KANs using backpropagation. However, a key challenge in training KANs is ensuring stability and convergence during optimization. To address this, researchers employ several techniques, including:

- **Regularization**: Methods such as dropout and weight decay help prevent overfitting.
- **Optimization Algorithms**: Careful selection of optimization algorithms and learning rates is crucial.
- **Normalization Techniques**: Batch normalization and layer normalization help stabilize training and accelerate convergence.

Despite these advancements, the main drawback of KANs is their slow training speed, which is approximately 10 times slower than MLPs with the same number of parameters. However, research has not yet extensively focused on optimizing KANs' efficiency, suggesting that there is still significant room for improvement.

In summary:

- If you need fast training, MLPs are the better choice.
- If you prioritize interpretability and accuracy and can tolerate slower training speeds, KANs are worth considering.



$x_{2,1}$
$\tilde{x}_{1,1,1}$ $\tilde{x}_{1,2,1}$ $\tilde{x}_{1,3,1}$ $\tilde{x}_{1,4,1}$ $\tilde{x}_{1,5,1}$
$\phi_{1,1,1}$ $\phi_{1,2,1}$ $\phi_{1,3,1}$ $\phi_{1,4,1}$ $\phi_{1,5,1}$
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$
$\tilde{x}_{0,1,1}$ $\tilde{x}_{0,2,1}$ $\tilde{x}_{0,1,2}$ $\tilde{x}_{0,2,2}$ $\tilde{x}_{0,1,3}$ $\tilde{x}_{0,2,3}$ $\tilde{x}_{0,1,4}$ $\tilde{x}_{0,2,4}$ $\tilde{x}_{0,1,5}$ $\tilde{x}_{0,2,5}$
$\phi_{0,1,1}$ $\phi_{0,1,2}$ $\phi_{0,1,3}$ $\phi_{0,1,4}$ $\phi_{0,1,5}$ $\phi_{0,2,1}$ $\phi_{0,2,2}$ $\phi_{0,2,3}$ $\phi_{0,2,4}$ $\phi_{0,2,5}$
$x_{0,1}$ $x_{0,2}$
$k = 3$
$\phi(x)$
$\phi(x) = \sum_{i=0}^{7} c_i B_i(x)$
$G_1 = 5$
t_{-3} t_{-2} t_{-1} t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8
grid extension
$\phi(x) = \sum_{i=0}^{12} c_i' B_i'(x)$
$G_2 = 10$

KAN architecture representation

2.4 Advantages of KANs over MLPs

Now we will focus on the advantages of KANs over traditional MLPs. Through a comparative analysis, we'll elucidate how KANs offer better performance, efficiency, and interpretability, revolutionizing the landscape of deep learning architectures.

1. **Enhanced Accuracy**: KANs have showcased remarkable accuracy in various tasks compared to MLPs. By leveraging the Kolmogorov-Arnold Representation Theorem, KANs can represent complex multivariate functions more effectively, leading to more accurate predictions.

2. **Flexibility and Generalization**: KANs offer greater flexibility and generalization capabilities compared to traditional MLPs. Their ability to adaptively learn activation functions allows them to capture nonlinear relationships in the data more effectively, leading to improved generalization performance.

3. **Robustness to Noisy Data**:KANs exhibit enhanced robustness to noisy data and adversarial attacks compared to MLPs. Their ability to learn more robust representations of data through adaptive activation functions makes them less susceptible to perturbations.

3 KANs for Natural Language Processing Tasks

In the field of natural language processing, KANs offer a novel approach to language modeling, semantic analysis, and text generation. By learning representations of linguistic data in a more interpretable and structured manner, KANs can facilitate tasks such as sentiment analysis, language translation, and document summarization. We will discuss how KANs are advancing the state-of-the-art in NLP and enabling more sophisticated language understanding systems.

4 E.D.A. and Data Preprocessing: used techniques

In this section are summarized the most relevant NLP related techniques used to analyze and prepare the datasets, in order to detect patological features inside the data (missing values, duplicates leading to lack of coherence among sentence and label), and to enhance the quality and usability of the text features inside the models performing analysis.
For data cleaning purposes, the following have been perfomed : non-word and stop-word removal, multiple spaces substitution with single space, to lowercase conversion, word tokenization and lemmatization.
For the Exploratory Data Analysis (EDA) of the first dataset - suited for a binary classification of people's opinion on movies - has been studied the correlation between the distribution of the labels among the dataset (0: not-liked film, 1: liked film) and the polarity of each sentence. The polarity computation is the assessment to each phrase of a real value between -1 and 1, with 1 meaning positive phrase (related to sentiment detection), -1 is negative and 0 stands for neutral. Then, N-gram frequency counting has been performed, throughout the usage of the CountVectorizer tool.
In order to prepare the dataset for model training, the Tokenizer SpaCy has been used, including sentence segmentation, parsing and named entity recognition. The last tool is the TF-IDF vectorizer, to transform text into numerical vectors through the recognition of the imortance of each word inside the document. The TF (Term Frequency, to find the frequency of a word in a document) formula is combined with the IDF (Inverse Document Frequency) to diminish the weight of frequently occurring words across all documents. Then, in order to turn the number of features found in a feasible amount that can be used inside a tradition model of MLP or in a KAN, the data have gone through a dimensionality reduction with an SVD.

5 Comparison of performances between MLPs and KANs

The aim of this project is to compare the results of the application of MLP and KAN on the same datasets in order to evaluate their applicability to NLP tasks, which usually involve a great number of features. The two tasks here described comprehend a binary classification problem and a multilabel classification problem, in order to evaluate the performance of KANs also by varying the complexity of the problem to face.

5.1 IMDB dataset: binary classification

The first dataset used is available at this link. It is composed by 2000 movie reviews, that can be either positive or negative. The positive ones have label 1, while the negative have label 0. The aim of this analysis is to predict the appreciation of a movie, given a sentence describing one's opinion on it.
The exploratory data analysis and the features conversion can be found at the following link. Here are resumed the passages performed up to the MLP usage:

- **Preprocessing:** special and single characters removal, convertion to lowercase, tokenization and lemmatization;
- **E.D.A.:** words appearances counting, study of correlation between polarity of the sentences and their label, study of n-grams frequencies among the reviews;
- **Sets preparation:** vectorization through TF-IDF vectorizer, features' dimensionality reduction using an SVD from 5000 to 500 features, labels' one-hot encoding, train-test-validation splitting.

The Multi Layer Perceptron model defined is composed by 2 hidden layers (of 256 an 192 neurons each), and an output layer with 2 neurons, each predicting the probability of belonging to one of the two classes. The loss function chosen is the Cross Entropy function, 100 epochs are set. The results, provided within a few seconds (time varying from device to device) are collected in the table 1.

KAN usage for IMDB dataset: Finding a KAN model suited for every task is not straightforward, because of the lack of documentation available at the moment. Hence, in order to find the best possible hyperparameters that provides good results, a grid search on the number of hidden neurons, the grid number and the regularization factor has been performed. The metrics adopted to evaluate the best model are the same of the MLP. At the end, the best model found is the one reported here:

```
n = 5
grid = 4
lamb = 0.005

model = KAN(width=[500, n, 2], grid = grid, k = 3, seed = 2024)
results = model.train(dataset, opt="LBFGS", steps=20, metrics =(train_acc, val_acc),
loss_fn=torch.nn.CrossEntropyLoss(), lamb = lamb)
```

executed in at most 23 minutes. The results are reported on the table 1. In this phase, the high sensitivity of those models to the regularization hyperparameter has been noticed. About them, better values could have been found with more computational capacity. In fact, a greater number of neuron per hidden layer lead to an Error flagged beacause of the lack of memory available.

5.2 Clinical Reviews dataset: multiclass classification

The second analysis has been perfomed on a dataset that collects several labels to be predicted: each sentence is referred to a particular environment or office inside a clinic: the task is to predict the correct addressee given an utterance. The reference and the dataset can be found at this link. The preprocessing part follows almost the same steps as above, passing through cleaning of missing values, stop words removal, tokenization and padding and one hot encoding of labels. There were 20 labels originally present in the dataset, therefore, for reducing the computational burden found as a great issue

	train set	test set	train set	test set
accuracy	1.0	0.86	1.0	0.81
precision	1.0	0.88	1.0	0.75
recall	1.0	0.87	1.0	0.88
f1 score	1.0	0.87	1.0	0.81
AUC	1.0	0.94	1.0	0.82

Table 1: Comparison between MLP (left) and KAN (right): results obtained on the IMDB dataset.

	train set	test set	train set	test set
accuracy	1.0	0.78	0.28	0.28
precision	1.0	0.77	0.26	0.28
recall	1.0	0.77	0.24	0.24
f1 score	1.0	0.77	0.19	0.19
AUC	0.94	0.98	0.50	0.50

Table 2: Comparison between MLP (left) and KAN (right): results obtained on the Clinical Reviews dataset.

to deal with when dealing with KANs, the number of label has been reduced to the 5 most frequent ones. An MLP model has been defined withan Embedding layer to reduce dimensionality from 500 to 50, 2 hidden layers and 5 output values, one for each label. The results are shown in table 2.

KAN usage for Clinical Reviews dataset The KAN model defined has an input layer of 500 entries, an hidden layer of 10 neurons and an ouput layer of 5 neurons. Unfortunately, KAN networks don't have the possibility to be used with an Embedding layer that can reduce dimensionality of features. SVD reduction and Autoencoders has been tried before the actual training phase on the features, but it has not been possible to find a transformation that could preserve enough informations to ensure a sufficient performance. Due to the great amount of features and the number of functions to train (1 for each label), the time needed to complete the training is of at least 1 hour and a half. Nevertheless, the results are not good as expected, even acting on the regularization term and on the number of hidden neurons. The results are stored in the table 2.

6 Discussion, conclusions and further developments suggestion

The results obtained are not successful as expected, given the premises of the KAN networks. In both of the analyzed datasets, despite a deep research for the best hyperparameters pattern, KANs performed worse than the MLP model used on the same set of features; almost all the metrics used to conduct the comparisons shows a lower value for predictions provided by KAN model, especially in the test set. Furthermore, the resources needed to support the training (memory and time) have not been sufficient to proceed with a more complex model, using more hidden layer or an higher complexity for the splines. Hence, the most common tasks related to Natural Language Processing scope are not suited for a feasible usage of KANs. Nevertheless, these new type of neural networks can be exploited for their most remarkable feature, that is the high interpretability, reachable with the trained analytical function mapping the feature space into the real numbers field.
A fine understanding of how a Large Language Model works is a task belonging to cutting-edge researches that aims to comprehend what is the behavior inside the black box of the generative language models. An example of this kind of research has been developed to understand the hidden processes that allow the generative model Claude 3 Sonnet to answer a prompt given from a user in a very short time. In order to map the activation of neurons interposed into the model architecture, a lot of resources of memory and time must be used. Researchers made use of sparse AutoEncoders to

simplify the relationships among neurons and map their activations triggered by a particular sentence, or manually paused to detect shortcuts. Hence, the interpretability is the key word for this purpose. Kolmogorov-Arnold Networks can represent a viable, alternative solution to this problem, when the urge of resources is not a problem anymore. Henceforth our suggestion is to use KAN networks for research porposes, in order to exploit their capacity to provide functions mapping the feature space.

7 Bibliography and website

- Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet
- KAN: Kolmogorov-Arnold Networks
- KANs tutorials and introduction
- IMDB dataset
- ClinicalReviews dataset

8 Task division

Marzola Devis: study, comprehension, and summary of the nature of this type of network, its functionality, the mathematical foundations on which it is based, and its advantages and limitations. Special attention is given to the topic of the "interpretability" of these networks. 30 hours
Ozen Aysenur: research of algorithms in the field of Natural Language Processing (NLP) that utilize multi-layer perceptrons, applied to datasets that have already been analyzed or are suitable for their use. 30 hours
Roccatello Mattia, Milan Lisa, Gugole Mattia: installation and initial experimentation on the functionality of these KANs; analysis of the provided datasets, comparison of training time, accuracy, and other metrics between the two algorithms used. Analysis of the interpretability of the results. 30 hours each