МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

ДОКЛАД

По теме «KAN»

по дисциплине «Спецсеминар»

Работу выполнил студент группы ПМ63 Брезицкий С.О.

Работу принял зав. каф. ИТ Подколзин В.В.

Краснодар-2025

Слайд 2. Эволюция архитектуры нейронных сетей уходит корнями в фундаментальные работы, заложенные в 1940-х годах Уорреном Маккаллохом и Уолтером Питсом, которые предложили концепцию искусственных нейронов и их взаимосвязь.

Однако значительные прорывы произошли только в 1980-х годах с разработкой алгоритмов обратного распространения ошибки: алгоритм Геоффри Хинтона и других – все это позволило создавать более глубокие нейронные сети и улучшить методы обучения.

В это время появились классические архитектуры, многослойные перцептроны (MLP), и сверточные нейронные сети (CNN), которые революционизировали различные области, включая компьютерное зрение, обработку естественного языка и распознавание образов – теперь мы говорим про своего рода инновационную архитектуру.

В основе этой новаторской концепции лежит теорема представления Колмогорова-Арнольда, математическая теория, разработанная Владимиром Арнольдом и Андреем Колмогоровым. Причем достаточно давно, вот только исследователи разработали архитектуру и небольшую библиотеку под работу недавно.

Теорема утверждает, что сложные многомерные функции могут быть разложены на более простые одномерные функции, полагая основу для уникальной структуры нейросети KAN.

Слайд 3. Многослойный перцептрон — частный случай перцептрона Розенблатта, в котором один алгоритм обратного распространения ошибки обучает все слои.

Многослойный перцептрон — нейронная сеть, состоящая из слоев, каждый из которых состоит из элементов — нейронов (точнее их моделей). Эти элементы бывают трех типов: сенсорные (входные, S), ассоциативные (обучаемые «скрытые» слои, A) и реагирующие (выходные, R). Многослойным этот тип перцептронов называется не потому, что состоит из нескольких слоев, ведь входной и выходной слои можно вообще не оформлять в коде, а потому, что содержит несколько (обычно, не более двух — трех) обучаемых (A) слоев. Модель нейрона (будем называть его просто нейрон) — это элемент сети, который имеет несколько входов, каждый из которых имеет вес. Нейрон, получая сигнал, помножает сигналы на веса и суммирует получившиеся величины, после чего передает результат к другому нейрону или на выход сети.

Несколько слоев, которые могут обучаться (точнее, подстраиваться) позволяют аппроксимировать очень сложные нелинейные функции, то есть их область применения шире, нежели однослойных.

Слайд 4. Входной слой состоит из нейронов, каждый из которых представляет один признак входных данных. Если у вас есть d признаков, то входной слой будет состоять из d нейронов. Эти нейроны просто передают входные значения в первый скрытый слой.

Каждый признак соответствует одному измерению входного вектора. Входной вектор передается в MLP, где каждый элемент вектора (признак) умножается на соответствующий вес и передается на следующий слой.

Проклятие размерности (curse of dimensionality) — это термин, введенный Ричардом Беллманом в 1957 году, описывающий различные феномены, которые возникают при анализе и организации данных в пространствах высокой размерности.

Проклятие размерности в нейронных сетях — это проблема, возникающая при работе с данными, у которых очень много признаков или параметров.

Представьте, что у вас есть таблица с данными, и каждая строка — это набор характеристик какого-то объекта. Чем больше таких характеристик, тем больше измерений у вашего пространства данных.

В низкоразмерных пространствах (например, с двумя или тремя признаками) данные располагаются близко к друг другу, поэтому их легко проанализировать. С увеличением числа измерений (размерностей) объём пространства данных растет экспоненциально. Очень быстро.

Представьте, что у нас есть MLP, который должен классифицировать объекты по двум признакам (например, рост и вес).

Это можно представить как двухмерное пространство, где каждый объект описывается точкой с координатами (рост, вес). Допустим, что мы хотим разделить это пространство на области, которые соответствуют различным классам объектов.

Теперь предположим, что у нас появляется третий признак, например возраст. Теперь каждый объект описывается точкой в трёхмерном пространстве (рост, вес, возраст). Модель должна разделить это трёхмерное пространство на области, соответствующие классам объектов.

Для того чтобы обучить MLP эффективно классифицировать объекты в этом высокоразмерном пространстве, нам нужно достаточное количество данных, чтобы покрыть это пространство.

В двухмерном случае для точного представления данных может потребоваться несколько десятков или сотен точек. В d-пространстве, чтобы иметь такую же плотность данных, требуется экспоненциально больше точек — примерно N^d, где N — количество точек данных на одно измерение.

Рассмотрим пример с простым MLP, у которого 100 входных признаков и 50 нейронов в скрытом слое.

Пусть у нас есть 1000 тренировочных примеров. В этом случае соотношение данных к параметрам составляет 1000/5000 = 0.2, что очень мало для эффективного обучения.

Теперь увеличим количество входных признаков до 1000. Количество весов станет 1000×50=50000

Для того чтобы сохранить соотношение данных к параметрам на приемлемом уровне, скажем 10:1, нам потребуется 500000 тренировочных примеров, что существенно больше и может быть трудно достижимо на практике.

Большинство точек данных находятся далеко друг от друга, и статистические закономерности, наблюдаемые в низкоразмерных пространствах, становятся менее значимыми.

Слайд 5. Представьте, что вы стараетесь найти иголку в стоге сена, но теперь сена в тысячу раз больше, чем раньше. Причем вы уже не различаете самые маленькие соломинки из-за их разреженности.

И это мы не говорим про переобучение. В высокоразмерных пространствах нейронные сети требуют большего количества параметров для эффективного обучения, что может привести к переобучению (overfitting). Хотя здесь подключаются “методы регуляризации”.

Это приводит к нескольким сложностям.

Во-первых, для того чтобы модель могла эффективно учиться на таких данных, ей нужно гораздо больше примеров, чтобы покрыть всё это огромное пространство.

Во-вторых, различия между данными становятся менее заметными, и модель может с трудом находить закономерности.

В-третьих, вычислительные ресурсы, необходимые для обработки и анализа этих данных, сильно увеличиваются – в разы.

Для преодоления этих проблем используются различные методы, такие как понижение размерности (например, метод главных компонент (PCA), t-SNE), архитектуры, которые лучше работают с высокоразмерными данными и

т.д. Но совсем недавно исследователи предложили новый способ работы. Радикально, но верно.

Поменять MLP на KAN. Сойти с нейронов к их связям.

Слайд 6. Теорема представления Колмогорова-Арнольда звучит примерно так: сложные функции многих переменных могут быть представлены в виде суперпозиции более простых одномерных функций.

Короче говоря, любую сложную функцию можно представить в виде простых.

Представьте, что у вас есть сложная функция, которая зависит от нескольких переменных, например, f(x,y). Это может быть что-то вроде f(x,y)=x^2+y^2+xy – квадратичная функция двух переменных.

Теперь, вместо того чтобы рассматривать эту сложную функцию как одно целое, теорема Колмогорова-Арнольда говорит нам, что мы можем разложить эту функцию на более простые части, которые зависят только от одной переменной. В общем виде для f(x,y):

Слайд 7. о хорошо, есть у нас разбитый полином на простую сумму функций, а зачем нам вообще это нужно? Чтобы при помощи сплайнов формировать гибкие функции активации, сократить число параметров и лучше контролировать процесс обучения.

Если мы можем найти все “мелкие функции” – нам больше не нужно переходить на высокие размерности – проблема curse of dimensionality решена.

Слайд 8. А как вообще выглядят эти функции?

KAN параметризирует функции активации, чтобы сделать их обучаемыми. В этом кроется весь смысл архитектуры: переход от активации нейронов – к ребрам нейросети.

А как управлять локально параметрами и переобучать функции активации? – B-сплайнами.

Сплайны — это математические функции, которые позволяют создавать плавные кривые, соединяя серию управляющих точек.

Кусочные полиномы, также известные как сплайны, являются мощным инструментом в математике и науке о данных для приближения и интерполяции функций.

Они представляют собой полиномы, которые определены на отрезках (или "кусках") и объединены таким образом, чтобы обеспечить гладкость и непрерывность на всем интервале.

1. f(x) - сложная нейронная функция, которую мы хотим аппроксимировать или представить.
2. n - количество базисных функций (B-сплайнов).
3. N_i(x), i-ая базисная функция (B-сплайн), зависящая от переменной (x).
4. w - весовой коэффициент, который определяет вклад каждого B-сплайна в общую функцию f(x).

Базисные функции обычно определяются рекурсивно, используя рекуррентные формулы, такие как формулы де Бура.

Эти функции могут быть локальными, что позволяет им вносить вклад только в определенные участки области определения функции f(x) – B-сплайны гибче и эффективнее для аппроксимации сложных функций.

Весовые коэффициенты wi – параметры модели, их можно настроить с использованием градиентного спуска или других алгоритмов оптимизации, чтобы минимизировать ошибку аппроксимации функции f(x).

Благодаря суммированию B-spline, перемноженных на весовые коэффициенты, мы непременно приближаемся к основной функции – закономерности, которую мы и пытаемся выбить на выходе из нейронки.

Одна из наиболее распространенных форм сплайнов — кубические сплайны, полиномы третьей степени (три слагаемых) на каждом сегменте между узловыми точками.

Эти полиномы выбираются таким образом, чтобы они были гладкими на границах сегментов, то есть чтобы первая и вторая производные были непрерывны.

Слайд 9. Для самых непосвящённых.

Производная отражает скорость изменения, как на картинке ниже. Если мы посчитаем производные на всем участке графика – мы получим новую функцию, которую также можно отразить визуально.

Если мы не видим в графике производной скачков, разрывов – она гладка.

Выбирают полиномы, в которых непрерывно растет или уменьшается скорость изменения их графика.

Это обеспечивает не только гладкость кривой, но и ее изменение, когда мы переходим с одного сегмента на другой.

Сплайны часто используются в восстановлении данных, когда нам нужно аппроксимировать или восстановить функцию по заданным точкам.

Слайд 10. Мы можем настраивать их форму и поведение, изменяя положение узловых точек или условия на границах сегментов. Это позволяет нам создавать кривые, которые наилучшим образом соответствуют нашим потребностям и требованиям при анализе данных или визуализации.

Чтобы создать сплайн, обычно начинают с набора управляющих точек, определяющих траекторию кривой. Затем кривая конструируется путем интерполяции или аппроксимации пути между этими управляющими точками с использованием базисных функций: B-сплайны или кривые Безье.

Слайд 11. Основное преимущество B-сплайнов – свойство локальности.

То есть изменения в одном сегменте сплайна оказывают непосредственное влияние только на этот сегмент, без существенного влияния на остальные части сплайна.

Короче говоря, сплайны можно изменять локально.

B-сплайны имеют структуру, основанную на узловых точках, которые определяются пользователем и влияют на форму сплайна.

Сразу хочется привести пример из программ Adobe, где мы можем поменять в настройках цвета кривыми интенсивность цвета на разных участках изображения или скорость движущегося объекта.

Каждый сегмент B-сплайна обычно определяется набором узловых точек и набором базисных функций, которые управляют формой сегмента. Их можно увидеть на гифке.

Эти базисные функции обладают свойством локальности и могут быть скомбинированы таким образом, чтобы обеспечить непрерывность и гладкость сплайна на всем интервале.

По итогу мы получаем контролируемый график, который можно локально изменять, двигая “ползунки” параметров. Если упрощенно. И соответственно мы можем управлять параметрами и функциями активации, но…

Слайд 12. KAN отличается от традиционных MLP тем, что заменяет фиксированные функции активации на обучаемые функции (B-сплайны) вдоль ребер сети.

Эта адаптивная архитектура позволяет KAN эффективно моделировать сложные функции, сохраняя интерпретируемость и сокращая количество параметров, необходимых для работы. Используя локальную гибкость B-сплайнов, эти функции придают KAN адаптивность.

Хорошо видно, как в зависимости от получаемых данных нейрон регулируют фактически B-spline на участках. Получается, что нейроны “динамически” отвечают на получаемые данные, а не пассивно передают активации.

Слайд 13. KAN демонстрирует неплохую масштабируемость по сравнению с MLP, особенно в сценариях с высокомерными данными. Ее способность декомпозировать сложные функции на более простые дает нам возможность работать с большими объемами данных.

Несмотря на использование меньшего количества параметров, KAN достигает более высокой точности и меньшего потери, чем традиционные MLP, на различных задачах.

Просто новая архитектура может гибко моделировать отношения в данных – у нас развязаны (частично руки).

Слайд 14. Структура KAN облегчает интерпретируемость, позволяя исследователям выводить символьные формулы, эффективно представляющие выученные закономерности.

В отличие от черных ящиков и системы “скрытых слоев” без контролируемого полноценно результата – KAN предлагают понимание того, как входные признаки трансформируются во всей сети, делая нейронки “прозрачнее”.

a) В классическом MLP у нас обучаемые веса, фиксированный функции активации на нейронах/нодах. Но вот на картинке b) в модели KAN – мы получаем обучаемые функции активации и суммирование операций над нейронами.

Слайд 15. Проблемы архитектуры KAN.

1. Фрактальные функции. В череде слагаемых “полинома Колмогорова — Арнольда” можно отыскать фрактальные функции, которые создают... фракталы.

Фракталы — это сложные геометрические фигуры, которые выглядят одинаково на любом уровне увеличения. Это значит, что если вы возьмете часть фрактала и увеличите её, она будет выглядеть так же, как и вся фигура.

Но вот только... такие функции достаточно сложные, чтобы работать с ними на вычислительных уровнях.

Не зря в машинном обучении функции активации линейно-кусочные. Короче говоря, простые по своей природе.

Эта проблема долго заставляла исследователей отвернуться от этой теоремы, как предвестника новой архитектуры нейронных сетей взамен полносвязных MLP.

2. Долгая обучаемость. Да, мы получаем 200 параметров в нейросети KAN вместо 3000 на архитектуре MLP. Мы получаем эффективность в 10 раз больше нежели у полносвязной модели. Но. Скорость обучения у такой нейронки – в те же 10 раз меньше…
3. Потенциальная неэффективность. Оба варианта архитектуры – оба аппроксиматора функции, поэтому эффективность KAN нужно будет еще показать на примере.