Reconocimiento de patrones

Diferenciación de matrices jacobianas de orden fraccionario y su aplicación en neuronas artificiales Redes

--Borrador del manuscrito--

Número de manuscrito:	PR-D-25-03298
Tipo de artículo:	Artículo completo
Sección/Categoría:	Aplicaciones innovadoras del reconocimiento de patrones
Palabras clave:	Métodos de optimización de orden fraccionario; Diferenciación de matrices de orden fraccionario; Matriz jacobiana de orden fraccionario; Método de descenso de gradiente de orden fraccionario; Redes neuronales artificiales
Abstracto:	La diferenciación de orden fraccionario presenta muchas características diferentes a la diferenciación de orden entero. Estas características pueden aplicarse a los algoritmos de optimización de redes neuronales artificiales para obtener mejores resultados. Sin embargo, debido a la insuficiente investigación teórica, actualmente no existe un método de diferenciación matricial de orden fraccionario que sea perfectamente compatible con la tecnología de diferenciación automática (Autograd). Por lo tanto, proponemos un método de cálculo de diferenciación matricial de orden fraccionario. Este método se introduce mediante la definición de la matriz jacobiana de orden entero. La denotamos como diferenciación matricial jacobiana de orden fraccionario. A través de la diferenciación matricial jacobiana de orden fraccionario, podemos llevar a cabo la regla de la cadena de orden fraccionario basada en matrices. Con base en el módulo Linear y la diferenciación de orden fraccionario, diseñamos la tecnología Autograd de orden fraccionario para permitir el uso de la diferenciación de orden fraccionario en capas ocultas, mejorando así la practicidad de la diferenciación de orden fraccionario en el aprendizaje profundo. En el experimento, según el framework PyTorch, diseñamos un método lineal de orden fraccional (FLinear) y reemplazamos nn.Linear en el perceptrón multicapa con FLinear. Mediante el análisis cualitativo de la pérdida de los conjuntos de entrenamiento y validación, el análisis cuantitativo de los indicadores del conjunto de prueba y el análisis del consumo de tiempo y de memoria de la GPU durante el entrenamiento del modelo, verificamos el rendimiento superior de la diferenciación de matrices jacobianas de orden fraccional y demostramos que es un excelente método de descenso de gradiente de orden fraccional en el ámbito del aprendizaje profundo.

Xiaojun Zhou

Facultad de Ciencias de la Información e Ingeniería, Universidad de Yunnan, ciudad de Kunming,

China, 650091

zxjssldqm@163.com (Xiaojun Zhou) y zhaochunna@ynu.edu.cn (Chunna Zhao)

30 de abril de 2025

Presentación del manuscrito: "Diferenciación de matrices jacobianas de orden fraccionario y su

Aplicación en redes neuronales artificiales

Estimado Editor,

Me complace presentar nuestro artículo de investigación, “Matriz jacobiana de orden fraccionario

Diferenciación y su aplicación en redes neuronales artificiales”, para su consideración en

Reconocimiento de patrones.

Este trabajo introduce un nuevo método de diferenciación de matrices de orden fraccionario, derivado

de la diferenciación jacobiana de orden entero, que llamamos jacobiana de orden fraccionario

diferenciación matricial (J). apalancamiento Y_o , ideamos un sistema automático de orden fraccionario

Técnicas de diferenciación (Autograd), que permiten el cálculo eficiente de fracciones.

Descenso de gradiente de orden (FGD) y sus variantes en redes neuronales artificiales (RNA).

El avance clave es la capacidad de nuestro método para manipular directamente gráficos computacionales.

nodos, resolviendo el desafío de larga data de la aplicación de FGD a las capas ocultas de ANN.

Nuestra solución cierra esta brecha al proponer un método compatible con Autograd,

Marcando una contribución significativa tanto al cálculo fraccional como a las redes neuronales.

mejoramiento.

Creemos que la novedad de este trabajo, su rigor técnico y su relevancia para el reconocimiento de patrones

El alcance justifica su inclusión en su diario.

Gracias por considerar nuestro manuscrito. Esperamos tener la oportunidad de...

contribuir a su estimada publicación.

Atentamente,

Xiaojun Zhou

Reflejos

Diferenciación de matrices jacobianas de orden fraccionario y su aplicación en redes neuronales artificiales

Xiaojun Zhou, Chunna Zhao, Yaqun Huang, Chengli Zhou, Junjie Ye, Ke-meng Xiang

- Propuesta de diferenciación de matriz jacobiana de orden fraccionario (J α) basada en la diferenciación de orden entero, revelando su estructura de matriz de bloques y su idoneidad para aplicaciones ANN.
- La diferenciación de matrices de orden fraccionario implícita mejora la eficiencia computacional de FGD al evitar operaciones explícitas con elementos de matriz.
- Explora FGD en capas ocultas de ANN, se integra con Autograd y propone Autograd de orden fraccional.
- Redefine las reglas de aplicación de FGD en ANN, eliminando la necesidad de pruebas de convergencia.

Diferenciación de matrices jacobianas de orden fraccionario y sus Aplicación en redes neuronales artificiales

Xiaojun Zhoua,b, Chunna Zhaoa,1, Yaqun Huanga , Chengli Zhoua , Junjie Sí , Kemeng Xianga

a Facultad de Ciencias de la Información e Ingeniería, Universidad de Yunnan, Kunming, 650500, China b Universidad Normal de Lijiang, Lijiang, 674199, China

Abstracto

La diferenciación de orden fraccionario tiene muchas características diferentes a la de orden entero. diferenciación. Estas características se pueden aplicar a los algoritmos de optimización de Redes neuronales artificiales para obtener mejores resultados. Sin embargo, debido a la insuficiente teoría... En la investigación científica, en la actualidad no existe un método de diferenciación de matrices de orden fraccionario que Es perfectamente compatible con la tecnología de diferenciación automática (Autograd). Por lo tanto, proponemos un método de cálculo de diferenciación de matrices de orden fraccionario. Este El método se introduce mediante la definición de la matriz jacobiana de orden entero. Denotamos Se la denomina diferenciación de matrices jacobianas de orden fraccionario (J α). Mediante J α， podemos llevar la regla de la cadena de orden fraccionario basada en matrices. Basada en el módulo lineal y la Diferenciación de orden fraccional: diseñamos la tecnología Autograd de orden fraccional para permitir el uso de diferenciación de orden fraccionario en capas ocultas, mejorando así La viabilidad de la diferenciación de orden fraccionario en el aprendizaje profundo. En el experimento, De acuerdo con el marco de PyTorch, diseñamos algoritmos lineales de orden fraccionario (FLinear) y Reemplazar nn.Linear en el perceptrón multicapa por FLinear. A través de la cualitativa Análisis del conjunto de entrenamiento y del conjunto de validación Pérdida, el análisis cuantitativo de la prueba establecer indicadores y el análisis del consumo de tiempo y el uso de memoria de la GPU durante Entrenamiento del modelo, verificamos el rendimiento superior de J α y demostramos que es un excelente

Esta investigación fue financiada por el Proyecto del Fondo de Innovación en Investigación de Posgrado de la Universidad de Yunnan.

Direcciones de correo electrónico: zxjssldqm@163.com (Xiaojun Zhou), zhaochunna@ynu.edu.cn (Chunna Zhao), huangyq@ynu.edu.cn (Yaqun Huang), chenglizhou@mail.ynu.edu.cn (Chengli Zhou), yejunjie_cdx@163.com (Junjie Ye), Kemengxiang6@gamil.com (Kemeng Xiang)

1Autor correspondiente

Se aplicó el método de descenso de gradiente de orden fraccionario en el campo del aprendizaje profundo. Los datos...
y el código están disponibles en Github.

Palabras clave:
Métodos de optimización de orden fraccionario, diferenciación de matrices de orden fraccionario,
Matriz jacobiana de orden fraccionario, método de descenso de gradiente de orden fraccionario, matriz artificial
redes neuronales

---

1. Introducción

La diferenciación de matrices de orden entero es una base teórica importante para el gradiente
Cálculo en redes neuronales artificiales (RNA). Con base en ello, los investigadores han propuesto
Muchos optimizadores de aprendizaje profundo de alto rendimiento, que han promovido el desarrollo
desarrollo de la tecnología ANN. Los algoritmos de optimización de aprendizaje profundo siempre han sido...
Mejorado desde la perspectiva de las diferenciaciones de primer y segundo orden. Cómo-
Sin embargo, con el desarrollo de la diferenciación de orden fraccionario, algunos académicos comenzaron a
Estudiar algoritmos de optimización de aprendizaje profundo desde la perspectiva del orden fraccionario.
diferenciación. Estos métodos pueden denominarse colectivamente como gradientes de orden fraccionario.
métodos de descendencia focal (FGD).

En teoría, los métodos FGD tienen características diferentes de los métodos de orden entero.
Estas características incluyen tener estrategias de programación de ritmo de aprendizaje avanzado incorporadas.
gías y tener menos probabilidades de quedarse atascado en los puntos de la silla de montar (ver Apéndice A), así como
con capacidades de regularización integradas (prueba en el Apéndice B). Debido a estas ventajas
A lo largo de los años, los académicos han intentado introducir el cálculo fraccionario en el aprendizaje profundo y
han logrado resultados prometedores [1, 2, 3]. Sin embargo, en las ANN, los datos se representan en
forma tensorial. En consecuencia, cuando la diferenciación fraccionaria se extiende de escalares a
tensores, la diferenciación de la matriz fraccionaria dentro de las capas ocultas de las ANN no puede
se puede calcular de la misma manera que la diferenciación de orden entero, haciendo que la matriz-
La regla de la cadena de forma no es aplicable. Como resultado, la diferenciación de orden fraccionario muestra un rendimiento deficiente.
compatibilidad con marcos de diferenciación automática de orden entero (Autograd).

Actualmente, existen tres enfoques para aplicar métodos FGD en ANN: (1) Iniciar
Desde la perspectiva de la diferenciación simbólica para estudiar los métodos de FGD. Para evitar la cal-

Cálculo de la diferenciación de matrices de orden fraccionario y los cálculos en cadena subsiguientes
En las ANN, solo utilizan la diferenciación matricial de orden fraccionario en el cuadrado medio.
Función de pérdida de error (MSELoss). Es decir, utilizan la función de potencia de orden fraccionario.
Fórmula de diferenciación para resolver la diferenciación de nodos secundarios en el nodo raíz, luego
Realizar retropropagación según la regla de la cadena y participar en el cálculo.
de gradientes de nodos intermedios y foliares. De hecho, las diferenciaciones de los nodos intermedios
Los nodos ate y hoja se obtienen mediante la tecnología Autograd de orden entero. No importa
cómo se transforma el MSELoss, en esencia, estos son métodos que solo se pueden utilizar
diferenciación de orden fraccionario en la capa de salida [4, 5, 6, 7, 8]. (2) Transformar la
Fórmula de Grunwald-Letnikov (GL) y uso de Autograd para crear la solución de gradiente.
Se trata de la acumulación no lineal de gradientes en diferentes nodos temporales. En esencia,...
Son métodos de optimización de orden entero con un impulso especial y un aprendizaje especial.
tasas [9, 10, 11]. (3) Abandone la regla de la cadena y utilice una diferenciación simbólica específica
Método para resolver gradientes en ANN específicas basado en la regla de Leibniz [12].

Los métodos anteriores concluyen que la diferenciación de orden fraccionario es superior a
Diferenciación de orden entero mediante análisis teórico y comparación experimental.
hijo. Sin embargo, en escenarios de aplicación práctica, los defectos de estos métodos también son
obvio. Por un lado, los académicos han ignorado la combinación de métodos relevantes.
Los sistemas con marcos de aprendizaje profundo convencionales no han diseñado sistemas de orden fraccionario.
Optimizadores de aprendizaje profundo desde la perspectiva de la tecnología Autograd. Esto puede afectar
Su posterior promoción en el campo de los métodos de optimización del aprendizaje profundo. Por otro lado,
Por otro lado, muchos gradientes de matrices de parámetros de orden fraccionario no se derivan del jacobiano.
Diferenciación de matrices, como las matrices de orden entero. Muchas de ellas son computacionalmente...
complejo durante el proceso de solución, y cuando se transforman las fórmulas, la base
de la diferenciación de matrices de orden fraccionario se debilita en diversos grados [13]. En resumen,
Los optimizadores de aprendizaje profundo de orden fraccionario propuestos con base en métodos FGD existentes
No son suficientes para superar por completo los numerosos aprendizajes profundos de orden entero.
optimizadores.

Además, los académicos han realizado análisis de convergencia sobre el FGD propuesto.
métodos, y la mayoría de ellos enfatizan una tasa de convergencia más rápida en el conjunto de entrenamiento que
los métodos de orden entero correspondientes. Sin embargo, los métodos FGD existentes y

Sus variantes son mejoras basadas en algoritmos de optimización de primer orden existentes,
y su rendimiento de convergencia es el mismo que el del entero correspondiente.
algoritmos de optimización de pedidos.

Dado que existe un mapeo lineal en la matriz de gradiente entre algunos valores existentes
Métodos FGD y métodos de optimización de orden entero o sus variantes, ajustando sus
En última instancia, se puede considerar que los órdenes de orden fraccionario ajustan la tasa de aprendizaje. Esto
Afecta aún más la aplicación en ingeniería de los métodos de desulfuración de gases de combustión (FGD). Por lo tanto, para...
Los métodos FGD tienen un desarrollo y aplicación a largo plazo en el aprendizaje profundo.
Es necesario reexaminar los métodos FGD y redefinirlos de acuerdo con las ideas de investigación.
de la tecnología Autograd y la diferenciación de matrices de orden entero. En este proceso, la
El mayor problema al que se enfrentan los métodos FGD es que la diferenciación de orden fraccionario no puede
se puede aplicar a las capas ocultas de las ANN [14, 15, 16, 17, 18, 19, 20].

Por tanto, este artículo propone una diferenciación de matriz jacobiana de orden fraccionario J α basada
sobre la definición de matrices jacobianas de orden entero y diferencias simbólicas de orden fraccionario
Diferenciación. En el experimento, combinado con el marco general de aprendizaje profundo...
En el trabajo de PyTorch, el módulo lineal se reconstruye en un módulo lineal de orden fraccionario.
ule. De hecho, este artículo propone preliminarmente la tecnología Autograd de orden fraccionario.
Al reconstruir de manera similar otros módulos en torch.nn, también podemos establecer un
ecosistema de tecnología Autograd de orden fraccional. La mayor diferencia entre
El trabajo en este documento y los métodos FGD anteriores es que a través del cálculo de la diferencia-
tiaciones de cada nodo en el gráfico de cálculo, el gradiente de orden fraccionario es finalmente
obtenido. A diferencia de los métodos FGD existentes que realizan modificaciones en Optimizer.step().
De hecho, este artículo realiza un cálculo de gradiente de orden fraccionario en Loss.backward()
(ver Figura 1).

En detalles específicos, extraemos la matriz de diferenciación de orden fraccionario requerida para
Resolviendo la matriz de gradiente de J α · Su tamaño es el mismo que el de la diferencia de orden entero.
Matriz de referenciación. Por el contrario, para casos de orden entero, solo múltiples completamente identificados
Las matrices de diferenciación tórica se pueden definir a partir de la definición de J α . Cuando α = 1.0,
La matriz de bloques obtenida es la misma que la matriz de diferenciación obtenida en el modelo tradicional.
casos de orden entero nacionales, es decir, solo se puede obtener una matriz de diferenciación.
Por el contrario, los casos de orden fraccionario pueden definir múltiples matrices de diferenciación diferentes.

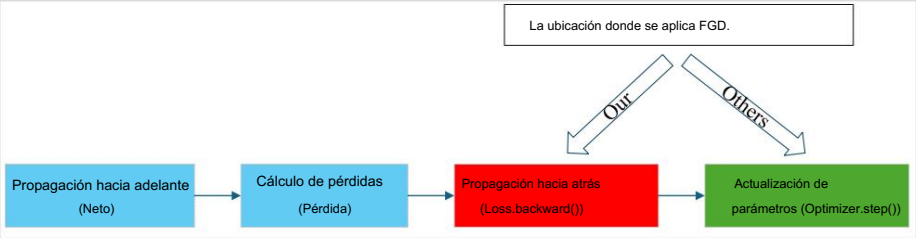


Figura 1: Diferencias de ubicación en el alcance operativo entre el método propuesto y los enfoques existentes en PyTorch.

ces. Dado que los cálculos basados en escalares no son prácticos en las ANN, necesitamos convertir el cálculos de cada matriz de diferenciación en J α en cálculos basados en matrices. El El tamaño de la matriz de diferenciación definida por J α está relacionado con el tamaño de la columna de la pa- Matriz de parámetros (generalmente la matriz de pesos en este documento) y cada matriz de parámetros tiene un tamaño diferente. Por lo tanto, cada matriz de parámetros puede seleccionar un número diferente de Matrices de diferenciación para el cálculo de gradientes. En el diseño de algoritmos basados en matrices, para Para reducir la complejidad del problema, en este artículo solo seleccionamos la matriz del primer bloque en J α (es decir, la primera matriz de bloque en la diagonal) como la diferenciación de orden fraccionario Matriz para participar en el cálculo del gradiente de la matriz de pesos en el nodo hoja.

En comparación con investigaciones y aplicaciones anteriores, el gradiente de peso de orden fraccionario La matriz ent en este artículo no es una aplicación lineal de la matriz de gradiente de orden entero, ni ¿Es equivalente a una variante del método de descenso de gradiente (GD)? Por el contrario, este El método del artículo utiliza plenamente la estrategia de programación de la tasa de aprendizaje y la regularización. Características de la optimización de orden fraccionario.

En resumen, en este artículo: (1) Exponemos el método teórico de J α . (2) Presentar una fórmula de diferenciación de funciones lineales de orden fraccionario adecuada para el campo del aprendizaje profundo. (3) En un perceptrón multicapa (MLP) dado, siguiendo la idea de Tecnología Autograd, combinamos la diferenciación de funciones lineales de orden fraccionario fórmula con J α y presentar un algoritmo de solución de gradiente basado en el cálculo matricial ciones. (4) Realizamos un experimento en una tarea de regresión basada en MLP. En el experimento iment, utilizando el marco de aprendizaje profundo PyTorch y combinando la tecnología de Autograd

ogy, reconstruimos el módulo lineal, agregando una función inversa de orden fraccionario
y un solucionador de diferenciación de matrices de orden fraccionario. De hecho, diseñamos un...
Ordenar el módulo lineal (FLinear) en PyTorch. FLinear es un módulo que contiene
Lineal. En el experimento, reemplazamos la capa Lineal en la RNA con la capa FLinear.
capa, lo que permite que el modelo utilice la regla de la cadena de orden fraccionario basada en matriz durante
Retropropagación. Mediante análisis de rendimiento de diferentes órdenes (cuando α = 1,0,
FLinear es lo mismo que Lineal), J α muestra superioridad.

En el aprendizaje profundo, el método propuesto en este artículo permite la matriz de orden fraccionario.
La diferenciación se aplica dentro de las RNA, no solo en la capa de salida. Esto permite
El método FGD ya no depende de la tecnología Autograd de orden entero y lo convierte en un
método que incluye la diferenciación de orden entero. Por lo tanto, las contribuciones de este
El artículo se puede resumir de la siguiente manera:

- Con base en la diferenciación de matrices jacobianas de orden entero, se propone J α y su
  Se estudian las características de la matriz de bloques. Las características del primer bloque...
  Se comparan y analizan matrices de orden fraccionario y de orden entero.
  experimentalmente.

- El método de diferenciación de matrices de orden fraccionario propuesto puede evitar errores explícitos.
  operaciones sobre elementos de la matriz, mejorando así la eficiencia computacional de
  el método FGD.

- Este artículo se centra en la aplicación del método FGD en las capas ocultas de las ANN.
  y explica y analiza experimentalmente el método FGD y sus variantes desde
  Una perspectiva de teoría matemática. Integrada estrechamente con la tecnología Autograd,
  Su capacidad de aplicación es generalizada. Esto hace que el método FGD ya no sea un
  método que sólo utiliza diferenciación de orden fraccionario en el nodo raíz.

- Dado que este artículo estudia el método FGD basado en la idea de la tecnología Autograd,
  Ogy, de hecho, hemos encontrado una aplicación innovadora para la automatización de orden fraccionario.
  Tecnología de diferenciación automática (FAutograd). Esto hace que el método FGD y
  sus variantes ya no son una transformación lineal o no lineal simple del entero-
  gradiente de la matriz de pesos de orden en Optimizer.Step(). En cambio, es una operación en

el gráfico de cálculo en Loss.backward().

- A través de una amplia investigación sobre la aplicación del método FGD en ANN, este El documento reafirma y define las reglas de aplicación del método FGD y aclara la no necesidad de demostrar la convergencia del método FGD.

Las partes restantes de este documento están organizadas de la siguiente manera: La Sección 2 presenta Estrategias de optimización del aprendizaje profundo y el estado actual de la investigación del método FGD. así como el método de aplicación de la diferenciación de orden fraccionario en MLP. Sección 3 propone el método en este artículo, incluyendo J^{α}, la diferenciación de orden fraccionario Fórmula en las ANN y el cálculo de la matriz de gradiente de orden fraccionario. Sección La sección 4 trata sobre los experimentos. La sección 5 es el resumen.

2. Trabajo relacionado y motivación

En esta sección, presentaremos las estrategias de aprendizaje profundo existentes y los métodos FGD. y explicar su relación. Sobre esta base, presentaremos en detalle los Aplicación de métodos FGD en ANN. Finalmente, explicaremos la motivación para la Trabajo realizado en este documento.

2.1. Estrategias de optimización del aprendizaje profundo y descenso de gradiente de orden fraccionario

En los métodos de optimización de parámetros de redes neuronales, el aprendizaje profundo de orden entero... Los algoritmos de optimización se dividen en optimización de primer orden y de segundo orden. métodos. El método FGD analizado en este artículo es una extensión basada en el primer Métodos de optimización de orden. Por lo tanto, esta sección revisa primero la optimización de primer orden. Métodos de ción. Luego, explora las estrategias de programación de la tasa de aprendizaje en el aprendizaje profundo. métodos de optimización. Finalmente, reexamina el estado actual de la investigación de los métodos existentes. Métodos FGD y los resume.

Métodos de optimización de aprendizaje profundo de orden entero: en el aprendizaje profundo de orden entero Marco de aprendizaje PyTorch, el optimizador de descenso de gradiente estocástico (SGD) es un Solución integrada. Incluye el método SGD [21], el método Momentum [22], y el método de gradiente acelerado de Nesterov (NAG) [23]. Estas tres optimizaciones Los métodos se logran ajustando los parámetros del optimizador. Entre ellos, el

El método SGD es el método de optimización más básico. Debido a sus problemas inherentes [24],
Los académicos propusieron los métodos Momentum y NAG para mejorar el optimizador SGD.
El método Momentum utiliza información de gradiente pasado, mientras que el método NAG utiliza
Información de gradiente futuro para optimizar los parámetros iterados. En la práctica...
En los escenarios de aplicación, el optimizador SGD suele combinarse con ellos para acelerar
convergencia del modelo y ayuda al modelo a escapar de óptimos locales. Otra categoría importante
Uno de los métodos de optimización de primer orden son los optimizadores de tasa de aprendizaje adaptativo. Estos optimizadores...
Los modelos incluyen Adagrad [25], Adadelta [26], RMSprop [27] y Adam [28], etc. Estos
Los optimizadores de tasa de aprendizaje adaptativo utilizan la información de primer o segundo orden.
momentos del gradiente, o ambos, y lograr actualizaciones iterativas del parámetro ma-
Trix mediante una transformación no lineal de la matriz de gradiente. Son equivalentes a
mejorando el tamaño del paso de la matriz de gradiente en cada iteración.

De hecho, los optimizadores de la tasa de aprendizaje adaptativo son equivalentes a dar a cada elemento de
La matriz de gradiente es una tasa de aprendizaje independiente, que es una transformación no lineal de la
matriz.

Estrategias de programación de la tasa de aprendizaje: además de utilizar las estrategias de programación de primer orden y
momentos de segundo orden del gradiente, otro método importante en el entrenamiento de
Las RNA son la estrategia de programación de la tasa de aprendizaje. Durante el proceso de iteración, estas...
Las estrategias ajustan dinámicamente la tasa de aprendizaje para aumentar el tamaño del paso de cada iteración.
Estas técnicas incluyen la caída de la tasa de aprendizaje, el recocido de coseno [29] y el calentamiento [30].
De hecho, son multiplicaciones escalares de la matriz, que son transformaciones lineales.
de la matriz.

Es decir, existe una diferencia importante entre el aprendizaje profundo de orden entero.
ing métodos de optimización y estrategias de programación de la tasa de aprendizaje a nivel de matriz.
En términos de la transformación de la matriz de gradiente, las operaciones de aprendizaje profundo de orden entero
Los métodos de temporización son transformaciones no lineales. Por el contrario, la tasa de aprendizaje...
Las estrategias de ajuste son transformaciones lineales. Sin embargo, a partir del proceso longitudinal
de iteración, la mayoría de los métodos de disminución de la tasa de aprendizaje, recocido de coseno y calentamiento se pueden
consideradas como transformaciones no lineales en la matriz de gradiente. Con base en estos descubrimientos
En las discusiones, al introducir los métodos FGD existentes, podemos encontrar sus características inherentes.
características y determinar si las matrices de gradiente son transformaciones lineales o

transformaciones no lineales en Optimizer.step().

Métodos FGD existentes: En la investigación de ANN, los métodos FGD utilizan fracciones
Técnicas de diferenciación de órdenes y optimización de orden entero para desarrollar nuevas
Aprendizaje de técnicas de optimización. Algunos de estos métodos utilizan gradientes de orden entero.
para resolver. Las referencias [9, 10, 11] son ejemplos. Al transformar la fórmula GL
Y combinándolo con la tecnología de diferenciación automática, finalmente obtienen el
gradiente de orden fraccionario. En esencia, se pueden clasificar como una programación de tasa de aprendizaje.
estrategia de inversión o un método de impulso con una ventana deslizante. De hecho, también pueden ser
Se considera un método de momento de orden entero especial. Además, se basan en
diferenciación numérica y no son convenientes para la compatibilidad con la tecnología Autograd.
tecnología. De manera similar, las referencias [31, 32] también utilizan la diferenciación numérica. Aunque
Se utiliza la definición de Caputo y se aplica específicamente la diferenciación de orden fraccionario.
Para MSELoss en el nodo raíz, desde la perspectiva de las ANN, esto es equivalente a usar-
diferenciación de orden fraccionario en la capa de salida mientras se mantiene el orden entero
Diferenciación en las capas ocultas. Otra categoría de métodos de FGD se basa en
Diferenciación simbólica para resolver problemas. En particular, la diferenciación de orden fraccionario
La fórmula de las funciones cuadráticas se utiliza para resolver, es decir, la diferenciación de orden fraccionario.
Se aplica la función a MSELoss [5]. La referencia [4] transforma la función cuadrática antes
Al aplicar la diferenciación de orden fraccionario para aliviar el problema de explosión de gradiente,
Causada por la pérdida de la dirección del gradiente debido al símbolo de valor absoluto. Referencia
[12] También utiliza la diferenciación simbólica, pero abandona la regla de la cadena y utiliza la ley de Leibniz.
Regla de Niz en RNA específicas. De hecho, al transformar la fórmula de diferenciación, el uso
Se evita el uso de valores absolutos. Sin embargo, incluyendo la referencia [7], estos métodos de FGD
El uso de la diferenciación simbólica puede tener una aplicación lineal con optimización de orden entero.
métodos de ción. Este mapeo lineal se puede lograr ajustando la tasa de aprendizaje en
Optimizadores de aprendizaje profundo de orden entero. Incluso si la matriz de gradiente de algunos de estos...
Los métodos ya no son una relación de mapeo lineal después de la transformación de fórmula, en esencia
En esencia, operan sobre la base de la matriz de gradiente de orden entero, en lugar de hacerlo directamente.
Resolviendo la matriz de gradiente de orden fraccionario de los nodos de hojas en el gráfico de cálculo.
La ventaja de los métodos FGD basados en la diferenciación simbólica es que pueden utilizar
La idea de la tecnología Autograd. Sin embargo, en términos generales, estos métodos pueden

tienen dos problemas: (1) La fórmula de diferenciación de orden fraccionario está muy modificada
ificado para evitar la diferenciación de matrices de orden fraccionario complejas. (2) Al realizar la
Regla de la cadena en forma matricial, para resolver el desajuste fila-columna entre dos diferenciaciones
Matrices en la multiplicación de matrices, la multiplicación de matrices se cambia al método de Hadamard.
producto.

2.2. Cuatro casos de resolución de pesos con diferenciación de orden fraccionario en multicapa
Perceptrones

Para profundizar más en el estado actual de la investigación del método FGD en ANN,
Utilizamos una MLP con dos funciones lineales y una función cuadrática como ANN, y
Presentamos cuatro casos de uso del método FGD. Esta RNA se puede expresar como la ecuación (1).
y el gráfico de cálculo correspondiente se muestra en la Figura 2. Específicamente, para simplificar
La descripción del problema de ejemplo y cumplir los requisitos de los hiperparámetros.
Para los experimentos posteriores, primero asumimos que las variables y parámetros en la ecuación (1)
son todos escalares y no hay función de activación después de cada función lineal.

$$\begin{aligned} y1 &= xw1 + b1 \\ y2 &= y1w2 + b2 \\ L &= (y2 - \text{yetiqueta})^2 \end{aligned} \tag{1}$$

donde x representa los datos de entrada, {w1,w2} son los pesos, {b1, b2} son los sesgos, ylabel
es la etiqueta y L es el valor de la función de pérdida (Pérdida).

En la Figura 2, a través de la regla de la cadena y la diferenciación de orden fraccionario, se presentan cuatro métodos:
Los métodos para calcular los gradientes de los nodos hoja se pueden resumir y definir.
w1 como ejemplo, las definiciones de sus cuatro métodos de cálculo de gradiente son las siguientes:
Caso 1:

$$\frac{(1,1,\alpha)}{w1} L(w1; x, b1, y1, y2) \qquad \frac{\partial L}{\partial y2}\frac{\partial y2}{\partial y1}\frac{\partial^{\alpha} y1}{\partial (w1)^{alfa}} \tag{2}$$

Aquí, $\frac{(1,\alpha,\alpha)}{w1} L(w1; x, b1, y1, y2)$ significa que tanto el orden entero como el orden fraccionario son
Se utiliza al calcular el gradiente de w1. $\frac{\partial^{\alpha} y1}{\partial (w1)}\alpha$ representa la derivada parcial de orden α-ésimo
tivo de y1 con respecto a w1.
Caso 2:

$$\frac{(1,\alpha,\alpha)}{w1} L(w1; x, b1, y1, y2) \qquad \frac{\partial L}{y2}\frac{\partial^{\alpha}}{\partial y2 \partial (y1)^{alfa}}\frac{\partial^{\alpha}}{y1\,\partial (w1)^{alfa}} \tag{3}$$

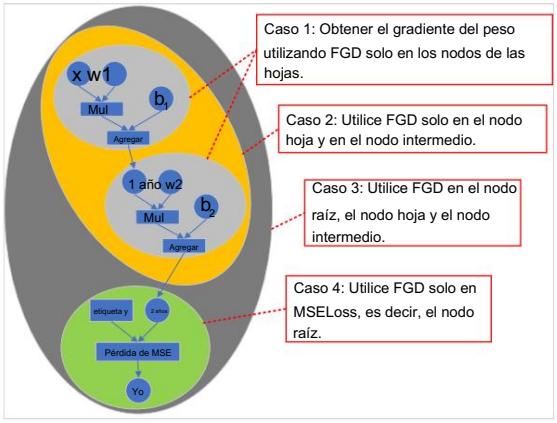


Figura 2: Gráfico computacional de ANN y 4 casos de FGD en retropropagación.

Caso 3:

$$\overset{alfa}{w1}L(w1; x, b1, y1, y2) \qquad \frac{\partial \alpha L}{\partial (y2)^{alfa}} \frac{\partial \alpha_{y2}}{\partial (y1)^{alfa}} \frac{\partial \alpha_{y1}}{\partial (w1)^{alfa}} \tag{4}$$

Aquí, $\alpha_{w1} L(w1; x, b1, y1, y2)$ representa el gradiente de orden fraccionario de L con respecto a w1.

Caso 4:

$$\overset{(\alpha,1,1)}{w1} L(w1; x, b1, y1, y2) \qquad \frac{\partial \alpha L}{\partial (y2)^{alfa}} \frac{\partial y2}{\partial y1} \frac{\partial y1}{\partial w1} \tag{5}$$

Entre las cuatro fórmulas de cálculo de gradiente, el Caso 4 corresponde a algunas carac-
Características de los métodos FGD existentes [4, 5, 6, 7, 8]. En retropropagación, el caso 4 es un método utilizado solo en la capa de salida. Por el contrario, los casos 1, 2 y 3 implican el Problema de la diferenciación de matrices de orden fraccionario en el cálculo interno de las neuronas. redes. Es decir, de acuerdo con la aplicación de los métodos FGD existentes en las Secciones 1 y 2.1, en las capas ocultas de las ANN, el Caso 4 es un método que debe basarse en números enteros. Diferenciación de órdenes para obtener gradientes de parámetros. No es independiente de los enteros. orden, y su llamado orden fraccionario no incorpora verdaderamente el orden entero.

2.3. Motivación

¿Por qué no se pueden aplicar los métodos FGD existentes a las capas ocultas de las ANN? Se asumen muchos estudios sobre diferenciación de orden fraccionario, variables y parámetros.

ser escalares. Sin embargo, en aplicaciones de ingeniería reales, los datos, pesos y bi-
Los ases que se entrenan son tensores [13, 33, 34, 35]. En la retropropagación, la multiplicación
entre tensores deben seguir las reglas de multiplicación de matrices. Es decir, cuando dos dife-
Las matrices de referenciación se multiplican, sus dimensiones de fila y columna deben ser iguales.
En la diferenciación de matrices de orden entero, siempre que las matrices estén en propagación hacia adelante
satisfacen las reglas de operación de la multiplicación de matrices, las matrices de diferenciación obtenidas
a través de las reglas de solución de diferenciación de matrices de orden entero en retropropagación también
satisfacen las reglas de multiplicación de matrices. Pero cuando introducimos el cálculo de orden entero
Idea de laminación (como la tecnología Autograd del marco de aprendizaje profundo PyTorch)
en orden fraccionario y usar la diferenciación de orden fraccionario para resolver los gradientes de
funciones matriciales, no podemos obtener una matriz de diferenciación que satisfaga la matriz
Reglas de multiplicación. En los optimizadores de aprendizaje profundo, esta matriz de diferenciación debe...
Participar en soluciones basadas en cadenas para obtener la matriz de gradiente. Por lo tanto, necesitamos
Estudiar un tipo de método de diferenciación de matrices de orden fraccionario de modo que la matriz obtenida
La matriz de diferenciación satisface la regla de la cadena basada en matrices.

3. Método

En esta sección se introduce sistemáticamente J^{α}. Mientras tanto, con base en estudios previos, una
Fórmula de diferenciación de orden fraccionario adecuada para el enfoque tecnológico de Autograd
Se presenta. Sobre esta base, el método de cálculo implícito del orden fraccionario
Se explica la matriz de gradiente.

3.1. Diferenciación de matrices jacobianas de orden fraccionario

Con base en la discusión de la motivación en la Sección 2.3, cuando los datos y parámetros
Los éteres cambian de escalares a tensores durante la retropropagación, los defectos de los existentes
Se pueden conocer los métodos de FGD. Para resolver este problema, necesitamos determinar los tamaños de los
Las matrices de diferenciación de orden fraccionario y entero son las mismas. Al mismo tiempo, cuando
el orden fraccionario $\alpha = 1.0$, la matriz de diferenciación obtenida debe ser igual a la
obtenido por el método de orden entero. Para ello, con base en la definición de
Matriz jacobiana de orden entero, utilizamos diferenciación simbólica de orden fraccionario en la

Función matricial para obtener la matriz jacobiana de orden fraccionario. Finalmente, una matriz de bloques que satisface las reglas de multiplicación de matrices durante la retropropagación se extrae de La matriz jacobiana de orden fraccionario. Los detalles son los siguientes:

Definición 1. Sea la función matricial $F(X) \quad R^{p\times q}$, y la matriz $X \quad R^{m\times n}$. Sus vectorizaciones se definen como

$$\mathrm{vec}(F(X)) \quad f11(X), \cdots, fp1(X), \cdots, f1q(X), \cdots, fpq(X)^{T} \quad R^{pq} \tag{6}$$

$$\mathrm{vec}(X) \ [x11, \cdots \quad , xm1, \cdots, x1n, \cdots \quad , xmn]^{T} \quad R^{\text{Minnesota}} \tag{7}$$

Entonces la matriz jacobiana de orden fraccionario de la función matricial F(X) se define como

$$Yo^{alfa} = {}^{alfa}(vecX)^{T} F(X) \quad \frac{\partial\, \alpha\, vecF(X)}{\partial((vecX)^{T})^{alfa}} \quad R^{pq\times mn} \tag{8}$$

Su representación específica es la ecuación (9).

$$Yo_{alfa} = \begin{matrix} \frac{\partial^{alfa} f11}{\partial((vecX)\,T)^{alfa}} \\ \vdots \\ \frac{\partial^{alfa} fp1}{\partial((vecX)\,T)^{alfa}} \\ \vdots \\ \frac{\partial^{alfa} f1q}{\partial((vecX)\,T)^{alfa}} \\ \vdots \\ \frac{\partial^{\alpha} fpq}{\partial((vecX)\,T)^{alfa}} \end{matrix} = \begin{matrix} \frac{\partial^{\alpha} f11}{\partial(x11)^{alfa}} & \cdots & \frac{\partial^{\alpha} f11}{\partial(xm1)^{alfa}} & \cdots & \frac{\partial^{\alpha} f11}{\partial(x1n)^{alfa}} & \cdots & \frac{\partial^{\alpha} f11}{\partial(xmn)^{alfa}} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \frac{\partial^{alfa} fp1}{\partial(x11)^{alfa}} & \cdots & \frac{\partial^{alfa} fp1}{\partial(xm1)^{alfa}} & \cdots & \frac{\partial^{alfa} fp1}{\partial(x1n)^{alfa}} & \cdots & \frac{\partial^{alfa} fp1}{\partial(xmn)^{alfa}} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \frac{\partial^{alfa} f1q}{\partial(x11)^{alfa}} & \cdots & \frac{\partial^{alfa} f1q}{\partial(xm1)^{alfa}} & \cdots & \frac{\partial^{alfa} f1q}{\partial(x1n)^{alfa}} & \cdots & \frac{\partial^{alfa} f1q}{\partial(xmn)^{alfa}} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \frac{\partial^{\alpha} fpq}{\partial(x11)^{alfa}} & \cdots & \frac{\partial^{\alpha} fpq}{\partial(xm1)^{alfa}} & \cdots & \frac{\partial^{\alpha} fpq}{\partial(x1n)^{alfa}} & \cdots & \frac{\partial^{\alpha} fpq}{\partial(xmn)^{alfa}} \end{matrix} \tag{9}$$

En la ecuación (9), cuando α = 1,0, J α = J, que es la matriz jacobiana de orden entero.

Ahora, sea la función matricial $F(X) = A^{p\times m} X^{m\times n} \quad R^{p\times n}$ en la Definición 1, y la matriz de coeficientes $A = \begin{matrix} a11 \cdots a1m \\ \vdots \quad \vdots \\ ap1 \cdots apm \end{matrix}$. Según la ecuación (9), para el orden entero diferenciación de matrices, tenemos la siguiente fórmula

$$J = \begin{bmatrix} A1 & O & \cdots & O \\ O & A2 & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & Un \end{bmatrix} \tag{10}$$

donde O es una matriz cero p × m, y A1 = A2 = · · · = An = A. Y $A^{T} \in R^{m\times p}$, cual es la matriz de diferenciación de orden entero. Cuando consideramos $F(X) = A_{p\times m} X_{m\times n}$ como un capa lineal de una ANN sin sesgo, A^{T} se puede utilizar como un nodo intermedio en el gráfico de cálculo y participar en el cálculo de las matrices de gradiente de cada nodo en el gráfico de cálculo.

De manera similar, la fórmula de cálculo para la diferenciación de matrices de orden fraccionario es la siguiente: Sigue

$$Yo^{alfa} = \begin{bmatrix} A11 & A12 & \cdots & A1n \\ A21 & A22 & \cdots & A2n \\ \vdots & \vdots & & \vdots \\ An1 & An2 & \cdots & Ana \end{bmatrix} \tag{11}$$

donde {A11, · · · , A1n, · · · , An1, · · · , Ann} $\in R^{p\times m}$.

Dado que la derivada de una constante no es cero en la diferenciación de orden fraccionario, en En la ecuación (11), las matrices de bloques en la no diagonal no son matrices cero, y el bloque n Las matrices en la diagonal también son diferentes. Es decir, J α tiene n^2 matrices de diferenciación, y cada una de estas matrices de bloques se puede utilizar como una matriz de diferenciación para participar En la derivada basada en cadena de la diferenciación de matrices de orden fraccionario. En particular, cuando α = 1,0 o n = 1, J α tiene solo una matriz de diferenciación, que es la matriz entera-matriz de diferenciación de orden.

3.2. Fórmulas de diferenciación de orden fraccionario en redes neuronales

En muchos métodos que aplican el método FGD a las ANN, debido a problemas como el límites superior e inferior de la diferenciación de orden fraccionario, tamaño del paso, cálculo de polinomios lación e integración, suelen transformar las fórmulas de GL, Riemann-Liouville (RL) o Caputo para cumplir con los requisitos de eficiencia en aplicaciones de ingeniería.

Sin embargo, en este artículo, basándonos en la idea técnica de Autograd, queremos estudiar
La tecnología Autograd de orden fraccional y el uso de la diferenciación de orden fraccional en el
red neuronal dentro de la RNA, es decir, en el grafo de computación. Por lo tanto,
Aplicar la diferenciación simbólica de orden fraccionario a la diferenciación matricial de orden fraccionario
en lugar de utilizar diferenciaciones numéricas como GL, RL y Caputo.

Con base en las discusiones de las secciones anteriores y las dos razones siguientes,
Utilice funciones lineales como ejemplo de discusión.

(1) Aunque los modelos de redes neuronales se han vuelto muy complejos y muchos clásicos
Han surgido modelos, la capa lineal basada en funciones lineales sigue siendo una importante
módulo en muchas ANN.

(2) En la parte experimental de este artículo, se realizó una tarea de predicción de series temporales basada en
Se lleva a cabo MLP y su modelo de red está compuesto de muchas capas lineales.

Así, basándonos en funciones lineales, damos una definición de la diferencia de orden fraccionario.
La fórmula de titulación propuesta en este artículo [36] es la siguiente:

Definición 2. Sea y = xw + b, entonces

$$\frac{\partial^{\alpha} y}{\partial w^{\alpha}} = {}_{0}^{RL}D_{o}^{\alpha} y = \frac{1}{\Gamma(2-\alpha)} |w|^{1-\alpha} + \text{signo}(w) |w| \frac{b}{\Gamma(1-\alpha)} {}^{-\alpha} \quad (12)$$

donde el orden α (0, 2). En este artículo, para simplificar la discusión del problema y
Centrándonos en los métodos de primer orden y de orden fraccionario, dejamos α (0, 1]. sign(•) es el signo
función, cuyo objetivo es garantizar que la dirección del gradiente del término correspondiente sea
No se pierde. En el aprendizaje profundo, x son los datos de entrada, w es el peso y b es el sesgo.

Dado que la regla de la cadena en la tecnología Autograd se basa en la diferenciación simbólica,
La ecuación (12) como fórmula de diferenciación simbólica cumple con los requisitos de la tecnología Autograd.
Tecnología para fórmulas de diferenciación. Al mismo tiempo, cabe señalar que en la Fórmula
12, dado que el estudio del límite inferior no es el foco de este trabajo, su límite inferior es
Se establece en 0. Este es el límite inferior predeterminado. De hecho, la fórmula de diferenciación de Definición
2 se obtiene por inducción estadística, es decir, el límite inferior es 0. Esto es consistente
con la fórmula de diferenciación de la función potencia obtenida después de usar la derivación RL. En
En aplicaciones de ingeniería, se seleccionan diferentes métodos de inicialización para los parámetros.
según diferentes modelos. Por lo tanto, cabe señalar que el límite inferior de
La diferenciación de orden fraccionario aquí es diferente de los valores de los parámetros iniciales. En

Al mismo tiempo, se añade un signo (•) antes del segundo término en la ecuación (12). Esto es lo mismo como la mayoría de los métodos que aplican la diferenciación de orden fraccionario en ingeniería, que consiste en evitar que se obtengan números complejos al resolver potencias fraccionarias. Como En su conjunto, este artículo también combinará el problema de la dirección del gradiente y explicará el Razón para añadir la función de signo en el segundo término desde una nueva perspectiva.

Tomando como ejemplo la ANN que se muestra en la Figura 2, está compuesta por múltiples Lincapas de oído y una pérdida MSE. Desde la perspectiva escalar, en orden entero, el gradiente La dirección de cualquier nodo hoja no se ve afectada por el valor positivo o negativo del nodo En la capa lineal al calcular el gradiente. La dirección del gradiente solo es relevante. afectado por la MSELoss, es decir, la influencia del valor retropropagado cuando el El nodo raíz está diferenciado. De hecho, si ignoramos la influencia de los nodos intermedios En el valor final del gradiente, podemos pensar que en orden entero, la dirección del gradiente está determinada por la función de pérdida. Esta se determina por la convexidad o no convexidad. naturaleza de la función. En el orden entero, por lo general, solo encontramos los problemas de Diferenciar funciones de primera y segunda potencia. Sin embargo, en orden fraccionario, Dado que la diferenciación simbólica es un operador lineal, también existe una diferencia de potencia cero. Tiación. La potencia cero también es una potencia par. Por lo tanto, para asegurar la unidad de orden fraccionario Autograd y Autograd de orden entero, agregamos una función de signo antes de la diferencia de potencia cero. diferenciación. Por el contrario, en las funciones lineales, no se añade ninguna función de signo para la primera potencia. diferenciación de orden fraccionario.

Ahora, sea la función matricial $F(X) = A_{p\times m} X_{m\times n} + b_{1\times n}$ (En PyTorch, el sesgo es se expande automáticamente a lo largo de las filas hasta b p×n debido a la transmisión), luego de acuerdo con Con las fórmulas (9) y (11), podemos obtener cualquier matriz de diferenciación de orden fraccionario en $\frac{\partial F(X)}{\partial X}$. Tomando el cálculo de la primera matriz de diferenciación A11 en la ecuación (11) como Por ejemplo, se puede expresar como ecuación (13).

$$A11 = \begin{bmatrix} \frac{\partial^{\alpha} f11}{\partial (x11)^{\alpha}} & \cdots & \frac{\partial^{\alpha} f11}{\partial (xm1)^{\alpha}} \\ \vdots & & \vdots \\ \frac{\partial^{\alpha} fp1}{\partial (x11)^{\alpha}} & \cdots & \frac{\partial^{\alpha} fp1}{\partial (xm1)^{\alpha}} \end{bmatrix} \tag{13}$$

Se puede saber que en la ecuación (14),

$$\begin{aligned} f11 &= a11\,x11 + a12\,x21 + \cdots + a1mxm1 + b1 \\ f21 &= a21\,x11 + a22\,x21 + \cdots + a2mxm1 + b1 \\ &\vdots \\ fp1 &= ap1\,x11 + ap2\,x21 + \cdots + apmxm1 + b1 \end{aligned} \tag{14}$$

De acuerdo con la ecuación (12), para $\frac{\partial^{\alpha} f11}{\partial(x11)^{\alpha}}, \cdots, \frac{\partial^{\alpha} fp1}{\partial(x11)^{\alpha}}, \cdots, \frac{\partial^{\alpha} f11}{\partial(xm1)^{\alpha}}, \cdots, \frac{\partial^{\alpha} fp1}{\partial(xm1)^{\alpha}}$, Tenemos la ecuación (15).

$$\begin{aligned} \frac{\partial^{\alpha} f11}{\partial(x11)^{\alpha}} &= \frac{a11}{|x11| + \text{signo}(x11)\,\Gamma(2-\alpha)} \frac{1-\alpha}{\Gamma(1-\alpha)} \left| \sum_{i=1}^{m} a1i\,xi1 - a11\,x11 + b1 \right|^{-\alpha} \\ &\vdots \\ \frac{\partial^{\alpha} fp1}{\partial(x11)^{\alpha}} &= \frac{ap1}{\Gamma(2-\alpha)} |x11|^{1-\alpha} + \text{signo}(x11) \frac{1-\alpha}{\Gamma(1-\alpha)} \left| \sum_{i=1}^{m} api\,xi1 - ap1\,x11 + b1 \right|^{-\alpha} \\ &\vdots \\ \frac{\partial^{\alpha} f11}{\partial(xm1)^{\alpha}} &= \frac{a11}{1-\alpha} |xm1|^{1-\alpha} \Gamma(2-\alpha) + \text{signo}(xm1) \frac{1-\alpha}{\Gamma(1-\alpha)} \left| \sum_{i=1}^{m} a1i\,xi1 - a11\,xm1 + b1 \right|^{-\alpha} \\ &\vdots \\ \frac{\partial^{\alpha} fp1}{\partial(xm1)^{\alpha}} &= \frac{ap1}{\Gamma(2-\alpha)} |xm1|^{1-\alpha} + \text{signo}(xm1) \frac{1-\alpha}{\Gamma(1-\alpha)} \left| \sum_{i=1}^{m} api\,xi1 - ap1\,xm1 + b1 \right|^{-\alpha} \end{aligned} \tag{15}$$

La ecuación (15) es el cálculo explícito de la diferenciación de matrices de orden fraccionario. En computadoras, al convertir el cálculo explícito de la diferenciación de matrices en implícito cálculo, podemos obtener de manera eficiente la matriz de diferenciación del nodo correspondiente y almacenarlo en el gráfico de cálculo correspondiente, y finalmente proporcionarlo a cada hoja nodo para calcular la matriz de gradiente correspondiente.

3.3. Cálculo de la matriz de gradiente de orden fraccionario

El problema central de este artículo es utilizar J α para realizar la aplicación del FGD método en capas ocultas. Como se describe en la Sección 2.2, los casos 1, 2 y 3 son esenciales. esencialmente el mismo tipo de problema. Por lo tanto, para simplificar el problema, los algoritmos y los experimentos en este artículo solo toman el Caso 1 como ejemplo. Y la RNA en el La parte experimental también se basa en el modelo de red de la ecuación (1) para realizar análisis cualitativos. y análisis cuantitativos sobre conjuntos de datos de series temporales del mundo real.

Como se puede ver en la ecuación (15), en la sección 2.2, la diferenciación de matrices de orden fraccionario se obtiene mediante cálculo explícito. Sin embargo, en escenarios de aplicación, esto es muy requiere mucho tiempo. Por lo tanto, en esta sección, primero necesitamos convertir el orden fraccionario La diferenciación de matrices de la Sección 2.2 se convierte en un cálculo implícito, es decir, todos los cálculos son basado en matrices.

De acuerdo con la ecuación (1) y la figura 2, en el ejemplo de modelo de red neuronal en este
En el artículo, la forma tensorial de cualquier capa lineal se puede expresar como la ecuación (16). Nótese que en
PyTorch, el sesgo se expande automáticamente a lo largo de rowa a b p×n debido a la transmisión.

$$Y_{p\times n} = X_{p\times m} W_{m\times n} + b^{1\times n} \tag{16}$$

Al resolver los gradientes de cada nodo en la ecuación (16), sea la retropropagación requerida
matriz sea G, $G = \begin{matrix} g11 \cdots g1n \\ \vdots \quad \vdots \\ gp1 \cdots gpn \end{matrix} \in R^{p\times n}$, entonces, según el Caso 1, el gradiente
Las matrices
de la matriz de pesos W, el nodo intermedio X y el sesgo b se pueden expresar
presionado como Eq. (17).

$$\begin{aligned} \alpha WY &= Xi\, jT \bullet G \quad XY \\ &= G \bullet WT \\ bY &= \sum_{pk=1} Griego \end{aligned} \tag{17}$$

donde • representa la multiplicación de matrices. Xi j representa la diferencia de orden fraccionario.
Matriz de titación para el bloque (i, j)-ésimo J $\alpha \cdot \sum_{pk=1} Gkn$ representa la suma de los elementos de
cada columna de G para obtener un vector fila de tamaño 1 × n, es decir, el gradiente de b. En el
Capa lineal, que también es un nodo hoja, la matriz de peso suele tener una longitud mucho mayor.
El impacto en el resultado de salida de la ANN es mayor que el del sesgo. Por lo tanto, para mayor comodidad
De la discusión cualitativa en la parte experimental de la Sección 4, en la Ecuación (17), sólo la
La matriz de gradiente de W se obtiene mediante la diferenciación de matrices de orden fraccionario, y
La matriz de gradiente de b se obtiene mediante la diferenciación de matrices de orden entero.

Ahora, si tomamos la primera matriz de bloque en la matriz jacobiana de orden fraccionario como la
Matriz de diferenciación. En la ecuación (17), es x11. Entonces, el proceso de cálculo implícito de
α WY se puede mostrar como Algoritmo 1.

En el algoritmo 1, se calculan principalmente el primer y segundo término de la ecuación (12). De acuerdo con
Según la ecuación (12), en forma tensorial, el primer término es en realidad una aplicación lineal en números enteros.
orden, y su cálculo es relativamente sencillo, correspondiendo a la línea 3 del Algoritmo 1.
En el caso 4, muchas diferenciaciones de orden fraccionario en funciones de pérdida también son aplicaciones lineales.
pings en orden entero, y son equivalentes a calcular solo la línea 3. Desde la línea
4 a 10, es el cálculo del segundo término de la ecuación (12). Definimos el resultado obtenido.

Algoritmo 1 Cálculos del gradiente de la matriz de pesos basados en X11

Requerir: W R $^{m\times n}$, X R p×m, b R $^{1\times n}$, G R p×n , alfa

Asegurarse: Wgrad Wgrad denota α WY.

0].view(1, −1) R 1×m Para obtener A11, la primera columna de W 1: F = W[:,

Se extrae y luego se transpone.

2: Calcula la primera parte de W, que es el primer término de la ecuación (12):

3: Wprincipal = X |F| $^{\frac{1-\alpha}{\Gamma(2-\alpha)}}$ R p×m

4: Calcule la segunda parte de W, que es el segundo término en la ecuación (12). El cálculo

La ecuación implica Wpartial1, Wpartial2, Wpartial3 y Wpartial4. Consulte la ecuación (15) para detalles:

5: Wpartial1 = (X • F 6: ^{T}).view(−1, 1) R p×1

Wpartial2 = (Wpartial1).expand(−1, m) R p×m

7: bias = torch. f ull((p, m), b[0].item()) R p×m Especifique b[0] como una matriz de tamaño

p × m

8: Wpartial3=(Wpartial2 − X F + sesgo) R p×m

9: Wpartial4=signo(F) |F| $^{\frac{}{\text{gamma}(1-\alpha)^{-\alpha}}}$ R p×m

10: Wparcial = Wparcial3 Wparcial4 R p×m Obtener la segunda parte de W. Ver

Ecuaciones (12) y (15).

11: Wdiferenciación = (Wprincipal + Wparcial) ^{T} R m×p Obtener la diferenciación de W.

12: Wgrad = Wdiferenciación • G R $^{m\times n}$ Obtener la pendiente de W.

matriz de diferenciación como la matriz de diferenciación de términos de orden fraccionario. Aunque la

El cálculo de la matriz de diferenciación de términos de orden fraccionario es relativamente complejo.

asegura la diferencia esencial entre la matriz de diferenciación final obtenida y

que en orden entero. Es decir, no se puede obtener simplemente ajustando el aprendizaje.

tasa o utilizando una estrategia de programación de tasa de aprendizaje especial.

En el algoritmo 1 se realizan 10 pasos de operaciones para obtener el gradiente de la

matriz de pesos W. Por el contrario, en la diferenciación de orden entero, WY =X ^{T} • G. Com-

En comparación con la diferenciación de matrices de orden entero, el método de este artículo aumenta en

complejidad computacional. Sin embargo, dado que ambos se basan en el cálculo implícito, esto

El aumento es lineal y controlable. Y cuando α = 1,0, usando el algoritmo 1 se puede obtener

$\frac{1}{WY} = X^{T} \cdot G$, que es el mismo que el resultado obtenido por diferenciación de matrices de orden entero.
Entiación.

El algoritmo 1 finalmente obtiene el gradiente del nodo hoja. Cuando reemplaza el gra-
Los participantes en el método GD y sus métodos variantes participan en la actualización iterativa.
de la matriz de parámetros, formará la optimización de orden fraccionario correspondiente
método, como el optimizador SGD de orden fraccional (FSGD) y el optimizador de orden fraccional
Optimizador de Adam (FAdam). Aquí, tomando el optimizador FSGD definido en este artículo como
Como ejemplo, analizamos su complejidad temporal y su convergencia.

Análisis de complejidad temporal: aunque el optimizador FSGD necesita realizar
múltiples pasos al resolver la matriz de gradiente de orden fraccionario e implica una diferencia
Matriz de referenciación con una asignación lineal a un término de orden entero y un término de orden fraccionario
Matriz de diferenciación. Sin embargo, al igual que los optimizadores SGD y Adam, su gradiente
Las matrices se basan en el cálculo implícito, es decir, la complejidad temporal del FSGD
El optimizador es el mismo que el de su método de optimización de orden entero correspondiente.
Aquí, suponiendo que dado el modelo de red y el lote de muestras de entrenamiento, podemos
Establezca la complejidad temporal del entrenamiento de un lote como O(d). Si el número total de iteraciones...
la época es n, entonces la complejidad temporal total del optimizador FSGD es O(n d), que es
Lo mismo que el de los optimizadores SGD, Adam y FAdam.

Análisis de convergencia: como se muestra en la Figura 1, los métodos de orden fraccionario existentes
mejorar en Optimizer.step(), que es una operación sobre la matriz de gradiente de orden entero
del grafo de cálculo. Cuando la operación implica una transformación no lineal de
la matriz, sus características de convergencia serán diferentes y la convergencia será separada.
Se requiere un análisis. Por el contrario, el método propuesto en este trabajo es un método operativo.
ción para resolver la matriz de gradiente de orden fraccionario del nodo hoja en Loss.backward().
Cuando asumimos que el límite de la función objetivo está acotado y la función fraccionaria-
El gradiente de orden está acotado, sus características de convergencia son las mismas que las de
sus correspondientes optimizadores de orden entero. El cambio del orden fraccionario tiene un
impacto lineal en la velocidad de convergencia de la pérdida [37]. Por lo tanto, no es necesario enumerar
fórmulas separadas para la demostración. Es decir, cuando se dan los mismos hiperparámetros, la conversión
Las velocidades de respuesta de ambos son las mismas. En muchos métodos de FGD aplicados al entrenamiento de ANN,
Aunque tienen una velocidad de convergencia más rápida que el orden entero, esta diferencia en

La velocidad de convergencia es en realidad equivalente al impacto de diferentes tasas de aprendizaje en la Trayectoria de convergencia en optimizadores de orden entero. Además, en aplicaciones de ingeniería... cationes, ya que el entrenamiento a menudo utiliza un mecanismo de detención temprana y conjuntos de datos del mundo real suelen estar sesgados. Por lo tanto, el óptimo global en el conjunto de entrenamiento no es necesariamente... el óptimo en el conjunto de prueba. En conclusión, este artículo presta más atención al rendimiento Rendimiento del modelo de red tras el entrenamiento con el método FGD. Por lo tanto, la información relevante La discusión se lleva a cabo con más detalle en la parte experimental de la Sección 4.

4. Experiment

Esta sección diseña una tarea de entrenamiento de ANN simple y verifica el método en dos tipos de conjuntos de datos del mundo real. Realiza análisis cuantitativos y cualitativos sobre los rendimiento del método.

4.1. Estructura general de la red neuronal

La red neuronal utilizada en el experimento se muestra en la ecuación (1) y el cálculo gráfico en la Figura 2. Cuando se utiliza FLinear para reemplazar Linear, su estructura general durante El entrenamiento se puede resumir en la Figura 3.

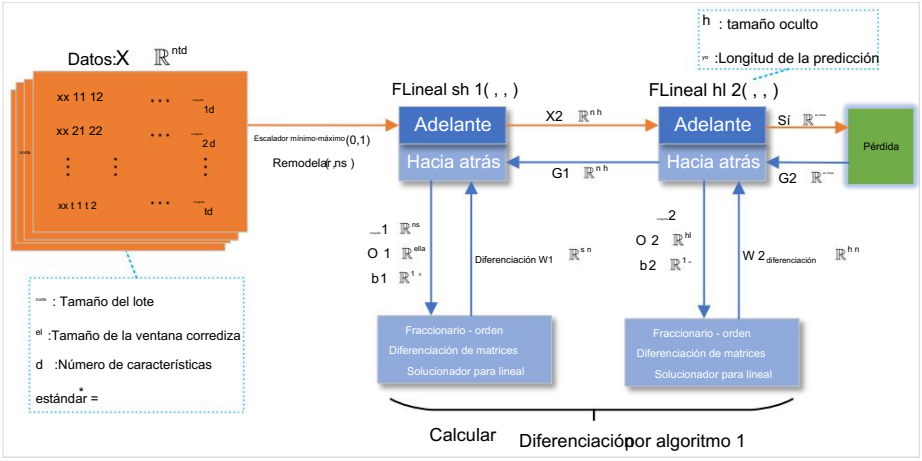


Figura 3: Estructura general del artículo.

La Figura 3 no es un modelo SOTA. Su propósito es facilitar la cuantificación y Análisis cualitativos del método FGD en este artículo. Tiene dos capas lineales y Implica el problema de la diferenciación de matrices de orden fraccionario en la retropropagación. de ANN, satisfaciendo los requisitos de la investigación sobre ANN.

En la Figura 3, FLinear solo agrega un parámetro entrante α. En Forward, FLinear es el Igual que el lineal. Sus diferencias radican en la retroactividad. Comparado con el lineal, el FLinear Backward tiene un solucionador de diferenciación de matrices de orden fraccionario diseñado por separado. El papel del solucionador es calcular las matrices de diferenciación de orden fraccionario de cada nodo, y el método de cálculo específico corresponde al Algoritmo 1. Como se describe En la Sección 3 anterior, el solucionador es la implementación de ingeniería específica del método en este artículo.

Al igual que en Autograd de orden entero, en Backward, la diferenciación de orden fraccionario La matriz se multiplica por la matriz retropropagada para obtener la matriz de gradiente de la Nodo. Luego, las matrices de gradiente de cada nodo hoja se guardan en el gráfico de cálculo. para que el optimizador correspondiente llame y participe en la actualización iterativa de la matriz de parámetros.

4.2. Preparación para el experimento

Configuración del entorno experimental: Dado que la Sección 4.3 implica el análisis Análisis del tamaño de la memoria y el consumo de tiempo, el entorno experimental específico Aquí se presenta la configuración. La CPU es Intel® Xeon® E5-2699 v4 @ 2,20 GHz. La GPU es NVIDIA GeForce RTX 2080 Ti. El IDE es Jupyter 7.2.2. El El lenguaje de programación es Python 3.12.7. El framework de aprendizaje profundo es PyTorch. 2.6.0 + CUDA 12.6.

Conjuntos de datos y preprocesamiento: en el experimento, se preprocesaron dos series de tiempo del mundo real. Se utilizan conjuntos de datos de dicción, a saber, el Promedio Industrial Dow Jones (DJI) y el público conjunto de datos ETTh1. DJI es un conjunto de datos con picos, fuertes fluctuaciones y fuerte ruido, Mientras que ETTh1 es lo opuesto. Esto se hace para analizar el rendimiento del método FGD. sobre diferentes tipos de conjuntos de datos. Su información detallada se muestra en la Tabla 1.

En la Tabla 1, las cinco características de DJI son Apertura, Alto, Bajo, Volumen y Cierre. El valor de cierre se predice conjuntamente mediante las cinco dimensiones de características. Las siete características

Tabla 1: Estadísticas de ETTh1 y DJI

Conjuntos de datos	Características	Pasos de tiempo	Granularidad	Intervalo de tiempo	Capacitación: Validación: Prueba	Etiquetas
DJI	5	5.969	1 día	3 de enero de 2000 - 22 de septiembre de 2023	7:2:1	Cerca
ETTh1	7	17.420	1 hora	1 de julio de 2016 - 26 de junio de 2018	7:2:1	[illegible]

de ETTh1 son HUFL, HULL, MUFL, MULL, LUFL, LULL y OT. El valor de OT
se predice conjuntamente por las siete dimensiones de características. No hay valores atípicos en ninguna de ellas.
conjunto de datos, por lo que no se requiere limpieza de datos. Como se muestra en la Figura 3, para el preprocesamiento de datos,
Sólo se llevan a cabo normalización Min - Max y mini-batching simple.

Configuración de hiperparámetros: para mostrar de forma justa las trayectorias de los valores de pérdida y las cantidades
Valores métricos estimativos de diferentes órdenes, algunas selecciones y configuraciones de hiperparámetros
se realizan en el experimento:

(1) Establezca valores de semilla aleatorios para el procesamiento por lotes y la inicialización de la matriz de ponderación. Al iterar
En diferentes órdenes, asegúrese de que los resultados de la mezcla del procesamiento por lotes se puedan reproducir
y las matrices de peso inicializadas son las mismas. Esto puede eliminar la influencia
de aleatoriedad en los resultados.

(2) Durante el experimento, establezca la tasa de aprendizaje de manera uniforme como lr = 1e − 4. En muchos casos
experimentos sobre la investigación del método FGD, la influencia de la tasa de aprendizaje en la
La trayectoria de convergencia es obvia. Especialmente para algunos métodos FGD, el ob-
La matriz de gradiente contenida puede en realidad ser equivalente a proporcionar un escalar externo.
La matriz de gradiente de orden entero. Este escalar cambia la magnitud del aprendizaje.
tasa de ing. Es decir, debido al cambio de orden, la matriz de gradiente de orden entero se vuelve
una tasa de aprendizaje mayor o menor para la multiplicación de matrices escalares. Por lo tanto,
Establecer una tasa de aprendizaje adecuada puede garantizar que el rendimiento se muestre correctamente.
El método FGD no se ve afectado por la tasa de aprendizaje.

(3) Como se muestra en la Figura 1, el método en este documento es resolver el problema de orden fraccionario.
Matrices de gradiente de cada nodo hoja en el grafo de cálculo. Durante la parametrización

Tras la actualización, el optimizador se selecciona según los requisitos de la tarea. En
Durante todo el proceso experimental de este artículo, se selecciona el optimizador SGD, que
es decir, el optimizador FSGD. Comparado con otros optimizadores (como el popular
Adam), el optimizador SGD no cambiará las características del parámetro
matriz. Otros optimizadores realizarán varias operaciones no lineales en la gradación.
Matriz ent después de obtener la matriz de gradiente del nodo hoja, destruyendo la aditividad
y la multiplicidad de la matriz de gradiente, cambiando así las características de
La matriz de gradiente de orden fraccionario. Finalmente, afectará nuestra cualitativa y
Análisis cuantitativo del método FGD. De igual manera, para la configuración de los parámetros
del optimizador SGD, establecemos la caída de peso_= 0 y el momento = 0. Porque
Estos dos hiperparámetros también destruirán la aditividad del orden fraccionario.
matriz de gradiente.

(4) De manera similar, la ANN utilizada en el experimento no establece una función de activación.
Porque la función de activación también destruirá la aditividad y la multiplicación.
idad de la matriz de gradiente de orden fraccionario, que afecta nuestros resultados cualitativos y cuantitativos.
Análisis estadístico de los resultados.

(5) De acuerdo con la Figura 3, establezca estos hiperparámetros de manera uniforme: la ventana deslizante
El tamaño es 36, el tamaño del lote es 256, la longitud de predicción es 48 y la función de pérdida es
La función utiliza MSELoss. Según las características de DJI y ETTh1, para apostar...
Para mostrar las trayectorias de convergencia de diferentes órdenes, al entrenar DJI, configure
tamaño oculto = 256. Al entrenar ETTh1, establezca el tamaño oculto = 128._

4.3. Análisis del rendimiento de los resultados experimentales

Entrenamos respectivamente los conjuntos de entrenamiento de dos conjuntos de datos y mostramos su convergencia.
trayectorias en 1500 iteraciones. Mientras tanto, después de cada iteración, el modelo se utiliza para
evaluación cuantitativa sobre el conjunto de validación, siendo la métrica de evaluación MSE, es decir,
Pérdida de MSE. Se realizan experimentos con diferentes órdenes y los resultados de los órdenes
Se recopilan α = {0,7, 0,8, 0,9, 1,0} y se muestran en la Figura 4.

En general, la Figura 4 muestra las características del optimizador FSGD que son diferentes
de los del optimizador SGD, incluida la programación de tasa de aprendizaje autónoma

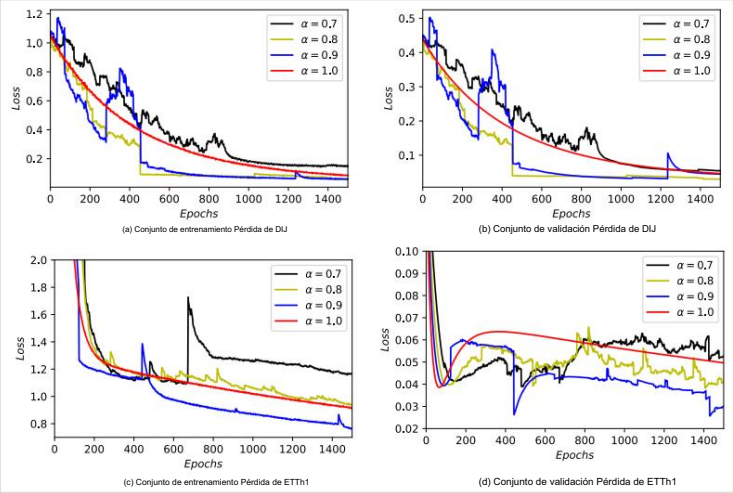


Figura 4: Las curvas de convergencia de DJI y ETTh1 en los conjuntos de entrenamiento y validación.

Estrategia y regularización. Los detalles son los siguientes:

(1) Aunque existen diversos factores que interfieren, todavía se puede observar que, como máximo,
órdenes fraccionarias, debido a la influencia de la estrategia de programación de la tasa de aprendizaje de
El optimizador FSGD, la pérdida de los dos conjuntos de datos a menudo disminuye más rápido que eso
del orden entero en la etapa inicial de las iteraciones. Por supuesto, esta diferencia en
La velocidad de convergencia está en un nivel lineal y no ha alcanzado el nivel de cambio desde
El optimizador SGD al optimizador Adam.

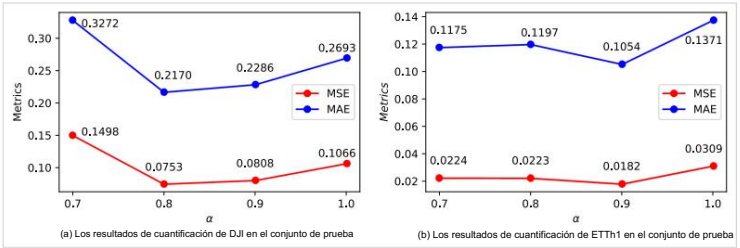
(2) También se puede ver en la Figura 4(d) que, durante la optimización de orden fraccionario
proceso, debido a la característica especial de regularización del optimizador FSGD,
Incluso en caso de sobreajuste, pueden salirse de la solución óptima local.
y buscar la solución óptima global. Por el contrario, una vez que el orden entero
se sobreajusta y cae en una solución óptima local, no puede saltar fuera de ella.

(3) En general, en comparación con el optimizador SGD, la trayectoria de convergencia del
El optimizador FSGD no es fluido. Esto se debe al término de orden fraccionario en
Ecuación (12). Al realizar la diferenciación de matrices de orden fraccionario, aunque
Este método solo extrae una matriz de bloques de la diferenciación de la matriz jacobiana,
La matriz de diferenciación de orden fraccionario de la matriz de peso se ve afectada por la

Matriz de entrada y el vector de polarización. En las ANN, debido al mecanismo de detención temprana...
anismo, normalmente solo guardamos el modelo con la menor pérdida en la validación
conjunto. Esta característica no suave en la trayectoria de convergencia no tendrá
un gran impacto en el rendimiento.

(4) Dado que finalmente obtenemos el modelo en el momento en que se produce la pérdida en la validación
El conjunto es el más pequeño, al comparar las trayectorias de convergencia de diferentes órdenes
En la Figura 4(b) y (d), se puede encontrar que el rendimiento de α = 0,7 es generalmente
en realidad peor que la del orden entero, y α = 0,8 es el segundo. Analizando
el término de diferenciación de orden fraccionario en la ecuación (12), se puede ver que cuando
A medida que el valor de α se acerca cada vez más a 0, el valor de la diferenciación será:
se vuelven cada vez más extremos. Desde la perspectiva del optimizador FSGD,
Significa explosión de gradiente. El entrenamiento del modelo ANN casi no tiene oportunidad.
oportunidad de aprovechar al máximo las ventajas únicas del optimizador FSGD y
empeoran cada vez más debido a la explosión del gradiente. Esto también verifica los resultados.
de algunos estudios previos [4, 7, 14], es decir, la selección óptima de orden fraccionario
El valor del método FGD suele ser 0,9 < α < 1,0.

En la Figura 4(b) y (d), la trayectoria de convergencia de Pérdida ya se puede mostrar claramente el rendimiento de cada pedido. Sin embargo, para el análisis cuantitativo, utilizamos el valor óptimo. modelos para realizar evaluaciones cuantitativas en el conjunto de prueba. Además del MSE, la media El error absoluto (MAE) también se utiliza como métrica de evaluación. Se utilizan comúnmente métricas de evaluación en tareas de regresión, y un valor más pequeño indica un mejor rendimiento. Los resultados cuantitativos detallados se muestran en la Figura 5.



(a) Los resultados de cuantificación de DJI en el conjunto de prueba (b) Los resultados de cuantificación de ETTh1 en el conjunto de prueba

Figura 5: Métricas del conjunto de prueba en diferentes α.

En general, en la Figura 5, el rendimiento de los órdenes de orden fraccionario α = {0,8, 0,9}
es mejor. Esto verifica los resultados y las especulaciones del conjunto de validación. Además,
En la Figura 5(a), el cambio de orden es más sensible a los resultados. Como se mencionó anteriormente,
El DJI tiene mucho ruido. Esto significa que necesita más regularización para evitar el sobreajuste.
El optimizador FSGD tiene regularización autónoma y funcionará mejor para
Conjuntos de datos de tipo DJI. Por lo tanto, después de seleccionar un orden adecuado, el rendimiento mejora...
La mejora será obvia. De igual manera, los resultados de la Figura 5(b) son una verificación inversa.
Dado que ETTh1 es un conjunto de datos estacionario con poco ruido, el cambio en los resultados cuantificados
No es tan grande como la de DJI cuando cambia el orden. Finalmente, podemos ver que en DJI,
Los resultados de α = 0,7 se deterioran rápidamente. Es el único orden fraccionario con peor rendimiento.
rendimiento que el orden entero. Esto verifica el análisis del cuarto punto de la Figura.
4.

Análisis del consumo de tiempo y el uso de memoria de la GPU: Para analizar el tiempo
consumo y uso de memoria de la GPU del optimizador FSGD (en este documento, se refiere a
SGD + FLinear) y el optimizador de orden entero, registramos el tiempo de ejecución y la GPU
memoria en diferentes órdenes para la primera época del conjunto de entrenamiento de ETTh1. Su detalle
La información se muestra en la Tabla 2.

Tabla 2: Comparación del consumo de tiempo y el uso de memoria de la GPU en diferentes órdenes.

Método	alfa	Tiempo(s)	Memoria (MB)
FSGD	0.9	0.7821	38.4521
FSGD	1.0	0.5873	38.4521
Dólar estadounidense	#	0.3041	38.4521

En la Tabla 2, se puede encontrar que ya sea utilizando directamente el optimizador SGD o utilizando
órdenes enteras o órdenes fraccionarias en el optimizador FSGD, su uso de memoria de GPU es
Lo mismo. Como se sabe de las Figuras 1 y 3, el optimizador FSGD (incluido el FAdam)
El optimizador en realidad cambia el método de cálculo de la matriz de gradiente de cada nodo.
En el gráfico de cálculo. Dados el modelo y los datos, el tamaño del gráfico de cálculo...
es fijo y no aumentará la memoria de la GPU debido a la adición de un orden fraccionario
solucionador de diferenciación de matrices en Loss.Backward(retain graph = True).

En la Tabla 2 nos centramos en analizar el consumo de tiempo en tres condiciones.

En primer lugar, aunque los consumos de tiempo de los tres son diferentes, todavía cambian a un ritmo
nivel lineal. Esto verifica el análisis de complejidad temporal del Algoritmo 1. En segundo lugar, el
El optimizador FSGD tarda más que el optimizador SGD. Esto se debe a que el cálculo
La diferenciación de matrices de orden fraccionario requiere más procesos intermedios y una
Se requiere cierto tiempo para estos procesos. En tercer lugar, en el FSGD
optimizador, el consumo de tiempo de $\alpha = 0,9$ es mayor que el de $\alpha = 1,0$. A través de la
De la discusión de la fórmula anterior 12 y el Algoritmo 1, se puede saber que el entrenamiento
El resultado del optimizador FSGD con $\alpha = 1.0$ es igual al del optimizador SGD, pero
En comparación con $\alpha = 0,9$, $\alpha = 1,0$ puede eliminar el cálculo del orden fraccionario.
término de diferenciación, ahorrando así la mayoría de los cálculos en la matriz de orden fraccionario
solucionador de diferenciación y reducción del consumo de tiempo.

A través de la visualización y análisis de las Figuras 4 y 5 y la Tabla 2, podemos verificar la
rendimiento del optimizador FSGD y luego verificar la superioridad del fraccional-
teoría de diferenciación de matrices de orden, así como la viabilidad de J^{α} en aplicaciones de ingeniería.
cationes.

5. Resumen

Este artículo estudia el método de diferenciación de matrices de orden fraccionario en las ANN.
Calculando la matriz de diferenciación de orden fraccionario utilizando el método J^{α}, se propone
Una solución novedosa para la investigación del método FGD, resolviendo el problema que presenta el FGD
El método no se puede aplicar dentro de las ANN. La viabilidad del método se verifica a partir de
tanto perspectivas teóricas como experimentales, sentando las bases para futuras investigaciones
y mejora de la tecnología FAutograd. En aplicaciones prácticas de ingeniería,
Al reemplazar Linear por FLinear, el optimizador se convierte en un optimizador de orden fraccionario.
Por ejemplo, en los experimentos, cuando el optimizador durante el entrenamiento se cambia de
De SGD a Adam, el optimizador FSGD se convierte en el optimizador FAdam. Sin embargo, como de-
Descrito en la configuración de hiperparámetros (3), los optimizadores de la tasa de aprendizaje adaptativo cambiarán
las características originales de la matriz de gradiente de orden fraccionario, que afectan nuestra cali-
Investigación cuantitativa y representativa sobre J^{α}. Por lo tanto, no se selecciona el optimizador FAdam.
en la parte experimental.

En trabajos futuros, estudiaremos y mejoraremos FLinear. Además,
Reconstruir módulos como CNN, RNN y Transformer en sus correspondientes
Módulos de orden fraccionario, a saber, FCNN, FRNN y FTransformer. Finalmente,
Completar la construcción del marco FAutograd para permitir la aplicación de
Diferenciación de matrices de orden fraccionario en más tipos de ANN.

De la discusión en la parte experimental, podemos encontrar que en la ingeniería práctica,
En aplicaciones de ingeniería, el método FGD es más propenso a la explosión de gradiente que el GD.
método. Esta situación es más evidente cuando el orden fraccionario es de configuración extrema.
En algunos casos, el método FGD experimentará una degradación del rendimiento debido a la
Problema de explosión de gradiente. Sabemos que esto se debe a la diferencia de orden fraccionario.
término de iniciación. Por lo tanto, en las aplicaciones prácticas de ingeniería de las ANN,
comience con la fórmula de diferenciación simbólica de orden fraccionario y optimícela aún más.
Ésta es nuestra futura dirección de investigación.

Apéndice A Trayectorias de convergencia y direcciones de gradiente

Sea la función escalar $y = (x1 + 2)2 + (x2 + 3)2$. Se puede saber que el extremo
El punto de la función es $(x1, x2) = (−2, −3)$. Según la ecuación (12), las fórmulas de actualización de los dos parámetros son las siguientes:

$$x_1^{(k+1)} = x_1^{(k)} - \eta \cdot \text{signo}(x_1^{(k)} + 2) \left| x_1^{(k)} + 2 \right|^{2-\alpha} \frac{2}{\Gamma(3-\alpha)} + \left| x_2^{(k)} + 2 \right|^{2-\alpha} \frac{(x_2^{(k)} + 3)2}{\Gamma(1-\alpha)} \tag{A.1}$$

$$x_2^{(k+1)} = x_2^{(k)} - \eta \cdot \text{signo}(x_2^{(k)} + 3) \left| x_1^{(k)} + 2 \right|^{2-\alpha} \frac{(x_1^{(k)} + 2)2}{\Gamma(1-\alpha)} + \left| x_2^{(k)} + 3 \right|^{2-\alpha} \frac{2}{\Gamma(3-\alpha)} \tag{A.2}$$

donde η es la tasa de aprendizaje y k = 1, 2, ..., N.

A partir de las fórmulas (A.1) y (A.2), las trayectorias de convergencia de los parámetros
Durante las iteraciones se puede obtener. Cuando α = {0,5, 1,0}, establecemos aleatoriamente (x_1^0, x_2^0) = (−3,5, −4,7) (que también son los límites inferiores de sus respectivas diferencias de orden fraccionario)
ferenciaciones) y registrar los resultados de las primeras 20 iteraciones. Véase la Figura A.1.

En la Figura A.1, se puede ver que, en comparación con la diferenciación de orden entero, la
La trayectoria de convergencia de la diferenciación de orden fraccionario muestra la característica de ser-
Al principio es rápido y luego lento. Por lo general, en los métodos de optimización de aprendizaje profundo, este tipo de...
El efecto solo se puede lograr llamando a un programador de tasa de aprendizaje. Por lo tanto, fraccionario-

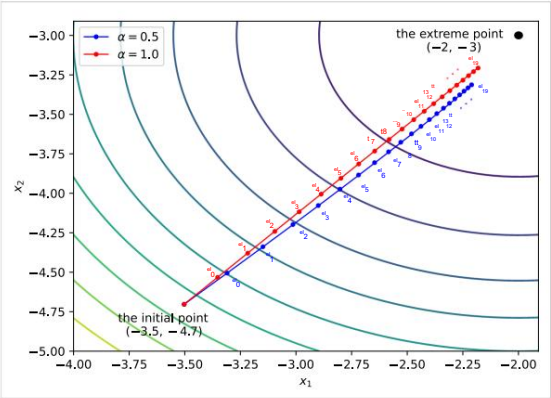


Figura A.1: Las trayectorias de convergencia de los parámetros en el orden fraccionario y el orden entero.

La diferenciación de órdenes es en realidad equivalente a tener una programación de tasa de aprendizaje especial. estrategia.

Respecto a las características del orden fraccionario en los puntos de silla, dado un ternario función z = x ^{2} ^{2} – y , Dibuja su gráfica tridimensional. Cuando los parámetros están en En el punto de silla, resolvemos los gradientes respectivamente usando diferenciación de orden entero y diferenciación de orden fraccionario, es decir, utilizando el método GD y el método FGD respectivamente. Véase la Figura A.2.

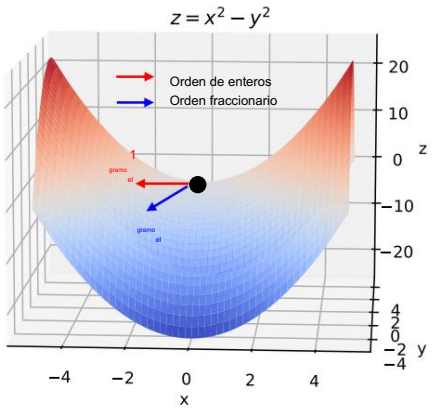


Figura A.2: Las direcciones de gradiente en los puntos de silla para el orden fraccionario y el orden entero.

Como es bien sabido, la diferenciación de orden entero hará que los parámetros oscilen.

izquierda y derecha en el punto de asiento. Por el contrario, según la Fórmula 12 y la Figura
A.2, debido a la influencia del término de diferenciación de orden fraccionario, cuando α 1.0, la
El vector de gradiente no debe ser paralelo al plano xz. Por lo tanto, el método FGD...
no quedarse atascado en el punto de silla de montar.

Apéndice B Prueba de regularización de diferenciación de orden fraccionario en sistemas artificiales Redes neuronales sociales

Demostración. Sea J(w) la función objetivo. Para el método GD, tenemos una regularización.
método con una función de penalización. Es decir,

$$J\tilde{}(w) = J(w) + p(w) \tag{B.1}$$

donde J˜(w) es la función objetivo total y p(w) es la función de penalización. Específicamente,
L ÿ1L son funciones de penalización cuando p(w) toma formas específicas.

Al utilizar el orden entero para calcular la derivada parcial de J˜(w), tenemos

$$J\tilde{}'(w) = J''(w) + p''(w) = J''(y)y''(x) + p''(w) \tag{B.2}$$

Por el contrario, para una función objetivo sin una función de penalización, utilizando fracciones
Ordenando la diferenciación y considerando y = wx + b, tenemos

$$\begin{aligned} Y_o''(w) = J'' \, {}_y\frac{\partial^\alpha}{(y)\,\partial w^\alpha} &= J''(y)\Big(\frac{}{|w|\,\Gamma(2-\alpha)}\Big)^{1-\alpha} + \text{signo}(w)\,\frac{b}{|w|\,\Gamma(1-\alpha)}\Big)^{-\alpha}\Big) \\ &= J''(y)\,\frac{}{|w|\,\Gamma(2-\alpha)}\Big)^{1-\alpha} + J''(y)\text{signo}(w)\,\frac{b}{|w|\,\Gamma(1-\alpha)}\Big)^{-\alpha} \end{aligned} \tag{B.3}$$

En la ecuación (B.3), sea $J_1''(w) = J''(y)\,|w|\,F(2-\alpha)\Big)^{1-\alpha}$ y $p_1''(w) = J''(y)\text{signo}(w)\,|w|\,F(1-\alpha)\Big)^{-\alpha}$.
Entonces, de acuerdo con la ecuación (B.2), tenemos

$$Y_o''(w) = J_1''(w) + p_1''(w) \tag{B.4}$$

De la ecuación (B.4), se puede ver que cuando se utiliza la diferenciación de orden fraccionario,
La función objetivo equivale a tener una función de penalización incorporada. Por lo tanto,
La diferenciación de orden fraccionario tiene la característica de ser equivalente a tener una construcción
en regularización. Además, este tipo de regularización no es una aplicación lineal con L_2 y L_1. □

Referencias

[1] Hongying Yang, Shuren Qi, Jialin Tian, Panpan Niu y Xiangyang Wang. ro-
Busto y representación de imágenes discriminativas: modelo Jacobi-Fourier de orden fraccionario
mentos. Reconocimiento de patrones, 115:107898, 2021.

[2] Sangheeta Roy, Palaiahnakote Shivakumara, Hamid A Jalab, Rabha W Ibrahim,
Umapada Pal y Tong Lu. Modelo de mejora de Poisson fraccional para la de-
Protección y reconocimiento en fotogramas de vídeo. Reconocimiento de patrones, 52:433–447, 2016.

[3] Tengfei Yang, Zhiquan Liu, Jingjing Guo, Yong Yu, Fang Ren y Teng Wang.
Análisis de imágenes mediante momentos de Bessel-Fourier esféricos ponderados por orden fraccionario.
Reconocimiento de patrones, 157:110872, 2025.

[4] Yong Wang, Yuli He y Zhiguang Zhu. Estudio sobre el orden fraccionario de alta velocidad.
Método de descenso de gradiente y su aplicación en redes neuronales. Neurocomputación.
489:366–376, 2022.

[5] Zeshan Aslam Khan, Naveed Ishtiaq Chaudhary y Syed Zubair. Fraccionario
Descenso de gradiente estocástico para sistemas de recomendación. Mercados Electrónicos, 29:
275–285, 2019.

[6] Yuquan Chen, Qing Gao, Yiheng Wei y Yong Wang. Estudio sobre orden fraccionario.
Métodos de gradiente. Matemáticas Aplicadas y Computación, 314:310–321, 2017.

[7] Xingwen Zhou, Zhenghao You, Weiguo Sun, Dongdong Zhao y Shi Yan.
Método de descenso de gradiente estocástico de orden fraccionario con momento y energía
Para redes neuronales profundas. Redes neuronales, 181:106810, 2025.

[8] Weipu Lou, Wei Gao, Xianwei Han y Yimin Zhang. Fraccional de orden variable
Método de descenso de gradiente y su aplicación en la optimización de redes neuronales. En
2022 34ª Conferencia de Control y Decisión de China (CCDC), páginas 109–114.
IEEE, 2022.

[9] Xiaojun Zhou, Chunna Zhao y Yaqun Huang. Un optimizador de aprendizaje profundo basado
sobre la definición de orden fraccionario de grunwald-letnikov. Matemáticas, 11(2):316, 2023.

[10] ZhongLiang Yu, Guanghui Sun y Jianfeng Lv. Un impulso de orden fraccionario
Enfoque de optimización de redes neuronales profundas. Computación neuronal y aplicaciones.
ciones, 34(9):7091–7111, 2022.

[11] Tao Kan, Zhe Gao, Chuang Yang y Jing Jian. Redes neuronales convolucionales
Basado en el momento de orden fraccionario para el entrenamiento de parámetros. Neurocomputación,
449:85–99, 2021.

[12] Xuetao Xie, Yi-Fei Pu y Jian Wang. Un algoritmo de descenso de gradiente fraccionario.
Robusto a los pesos iniciales del perceptrón multicapa. Redes neuronales, 158:154–
170, 2023.

[13] Sroor M Elnady, Mohamed El-Beltagy, Ahmed G Radwan y Mohammed E
Fouda. Un estudio exhaustivo de los métodos de descenso de gradiente fraccional y sus
Análisis de convergencia. Caos, solitones y fractales, 194:116-154, 2025.

[14] Xiaojun Zhou, Chunna Zhao, Yaqun Huang, Chengli Zhou y Junjie Ye. Soy-
Método probado de descenso de gradiente de orden fraccional basado en un perceptrón multicapa.
Redes neuronales, 183:106970, 2025.

[15] Min-Rong Chen, Bi-Peng Chen, Guo-Qiang Zeng, Kang-Di Lu y Ping Chu.
Una red neuronal bp de orden fraccionario adaptativa basada en optimización extremal
Para el reconocimiento de dígitos manuscritos. Neurocomputing, 391:260–272, 2020.

[16] Hongqiu Zhu, Zhiliang Wu, Chunhua Yang, Tao Peng, Zhiwen Chen y Xi-
Aoyue Yang. Método de ascenso más empinado fraccional para la detección de fallas en TCU. IFAC-
PapersOnLine, 51(24):1336–1342, 2018.

[17] Bi-Peng Chen, Yun Chen, Guo-Qiang Zeng y Qingshan She. orden fraccionario
Redes neuronales convolucionales con optimización extrema de la población. Neurocom-
Puting, 477:36–45, 2022.

[18] Abdul Wahab, Shujaat Khan, Imran Naseem y Jong Chul Ye. Actuación
Análisis de algoritmos de aprendizaje fraccional. Transacciones IEEE sobre Procesamiento de Señales.
ing, 70:5164–5177, 2022.

[19] E Viera-Martin, JF Gomez-Aguilar, JE Sol ´ıs-Perez, JA Hern andez-P erez, and
RF Escobar-Jiménez. Redes neuronales artificiales: una revisión práctica de aplicaciones
ciones que involucran cálculo fraccional. El artículo especial de la Revista Europea de Física...
ics, 231(10):2059–2095, 2022.

[20] Manisha Joshi, Savita Bhosale y Vishwesh A Vyawahare. Un estudio de fracturación
Aplicaciones del cálculo racional en redes neuronales artificiales. Inteligencia Artificial
Reseña, páginas 1–54, 2023.

[21] Sebastian Ruder. Una visión general de los algoritmos de optimización por descenso de gradiente. arXiv
preimpresión arXiv:1609.04747, 2016.

[22] Ning Qian. Sobre el término de momento en algoritmos de aprendizaje por descenso de gradiente. Neu-
Redes regionales, 12(1):145–151, 1999.

[23] Yu. Nesterov. Un método para resolver un problema de programación convexa con conversión
Tasa de dependencia de (1/k²). En Matemáticas Soviéticas Doklady, 1983.

[24] Richard S Sutton. Dos problemas con la retropropagación y otros métodos de descenso más pronunciado.
procedimientos de aprendizaje para redes. En Actas de la Reunión Anual de la
Sociedad de Ciencias Cognitivas, volumen 8, 1986.

[25] John Duchi, Elad Hazan y Yoram Singer. Métodos de subgradiente adaptativos para
Aprendizaje en línea y optimización estocástica. Revista de aprendizaje automático.
búsqueda, 12(7), 2011.

[26] Matthew D. Zeiler. Adadelta: un método de tasa de aprendizaje adaptativo. Preimpresión de arXiv.
arXiv:1212.5701, 2012.

[27] Tijmen Tieleman, Geoffrey Hinton y otros. Conferencia 6.5-rmsprop: Divide la gradiente
ent por un promedio móvil de su magnitud reciente. COURSERA: Redes neuronales
para el aprendizaje automático, 4(2):26–31, 2012.

[28] Diederik P Kingma y Jimmy Ba. Adam: Un método para la optimización estocástica.
Preimpresión de arXiv arXiv:1412.6980, 2014.

[29] Ilya Loshchilov y Frank Hutter. Sgdr: Descenso de gradiente estocástico con temperaturas cálidas.
reinicia. preimpresión de arXiv arXiv:1608.03983, 2016.

[30] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong y Richard Socher.
Una mirada más cercana a las heurísticas de aprendizaje profundo: reinicios de la tasa de aprendizaje, calentamiento y
destilación. preimpresión de arXiv arXiv:1810.13243, 2018.

[31] Yeonjong Shin, Jer´ome Darbon y George Em Karniadakis. Aceleración de gra-
Descendencia de dient y Adán mediante gradientes fraccionales. Redes neuronales, 161:185–201.
2023.

[32] Jian Wang, Yanqing Wen, Yida Gou, Zhenyun Ye y Hua Chen. Fraccionario-
Aprendizaje por descenso de gradiente de orden de redes neuronales bp con derivada de caputo.
Redes neuronales, 89:19–30, 2017.

[33] Yuan Cao y Shuai Su. Algoritmos de descenso de gradiente fraccional para sistemas con
Valores atípicos: Una derivada fraccionaria de una matriz o una derivada fraccionaria escalar. Caos,
Solitones y fractales, 174:113881, 2023.

[34] Haixin Wu, Yaqian Mao, Jiacheng Weng, Yue Yu y Jianhong Wang. frac-
Modelo de aprendizaje conjunto de máquinas de refuerzo de gradiente de luz: un modelo no causal
Enfoque de descenso por diferencias fraccionarias. Information Fusion, 118:102947, 2025.

[35] Shuli Yan, Qi Su, Zaiwu Gong, Xiangyan Zeng y Enrique Herrera-Viedma.
Predicción de la opinión pública en línea basada en el modelo gris fraccional móvil con nuevo
Prioridad de la información. Information Fusion, 91:277–298, 2023.

[36] Igor Podlubny. Ecuaciones diferenciales fraccionarias: una introducción a las ecuaciones fraccionarias.
derivadas, ecuaciones diferenciales fraccionarias, a métodos de su solución y
Algunas de sus aplicaciones. Elsevier, 1998.

[37] Priyanka Harjule, Rinki Sharma y Rajesh Kumar. gradiente de orden fraccionario
Enfoque para optimizar redes neuronales: un análisis teórico y empírico.
Caos, solitones y fractales, 192:116009, 2025.

Declaración de intereses

Los autores declaran que no tienen ningún interés financiero en competencia conocido ni relaciones personales que pudieran haber parecido influir en el trabajo presentado en este artículo.

Los autores declaran los siguientes intereses financieros/relaciones personales que pueden considerarse como posibles intereses en conflicto: