CP: 99

Dario Lopez Falcon

Marzo, 2024

Ejercicio#1 y 2

ver en respuestas.py

Ejercicio#3

En la diferenciación automática, los modos hacia adelante y hacia atrás son dos enfoques para calcular derivadas de funciones de manera eficiente.

- El modo hacia adelante calcula la derivada de una función evaluando la función y sus derivadas en un camino específico a través de un grafo, desde las entradas hasta la salida. Esto implica seguir el camino de las operaciones desde el inicio hasta el final, calculando las derivadas parciales a medida que avanzamos.

- El modo hacia atrás, también conocido como "backpropagation", calcula las derivadas a través de la cadena de reglas de derivación utilizando la regla de la cadena. Comienza desde el final del grafo y trabaja hacia atrás, propagando las derivadas hacia atrás a través de las operaciones realizadas en el camino hacia la salida. Esto es especialmente útil en redes neuronales, donde se necesita calcular gradientes para ajustar los pesos de las conexiones.

Ejercicio# 4

a) Para demostrar que calcular el gradiente usando el método hacia adelante tiene una complejidad $O(nm)$, donde n es la cantidad de variables y m es la cantidad de operaciones necesarias para evaluar la función, podemos considerar el siguiente razonamiento:

En el método hacia adelante, para calcular el gradiente, necesitamos evaluar la función $f(x)$ y sus derivadas parciales con respecto a cada variable en

un camino específico a través de un grafo computacional. Este proceso implica calcular la función y sus derivadas parciales en n puntos diferentes (uno para cada variable), y cada evaluación requiere $O(m)$ operaciones, donde m es la cantidad de operaciones necesarias para evaluar la función $f(x)$.

Entonces, en total, el costo computacional para calcular el gradiente usando el método hacia adelante es $O(nm)$.

b) Para demostrar que calcular el gradiente usando el método hacia atrás también tiene una complejidad $O(nm)$, podemos seguir un razonamiento similar:

En el método hacia atrás, utilizamos el algoritmo de backpropagation para calcular las derivadas parciales de la función $f(x)$ con respecto a cada variable. Este algoritmo también implica evaluar la función y sus derivadas parciales en cada variable, pero en lugar de hacerlo en n puntos diferentes, lo hacemos en un solo punto (el punto de evaluación x). Esto implica n operaciones para calcular las derivadas parciales con respecto a cada variable, y cada operación requiere $O(m)$ operaciones, donde m es la cantidad de operaciones necesarias para evaluar la función $f(x)$.

Por lo tanto, el costo computacional para calcular el gradiente usando el método hacia atrás también es $O(nm)$.

Ejercicio#5

Además de Autograd, algunas otras bibliotecas de diferenciación automática actualmente utilizadas en Python son:

1. TensorFlow: TensorFlow es una plataforma de código abierto para machine learning desarrollada por Google. Ofrece capacidades de diferenciación automática a través de su módulo tf.GradientTape.

2. PyTorch: PyTorch es otra popular biblioteca de machine learning de código abierto desarrollada por Facebook. Proporciona diferenciación automática a través de su módulo torch.autograd.

3. JAX: JAX es una biblioteca de aceleración de álgebra lineal y diferenciación automática desarrollada por Google. Ofrece una sintaxis similar a NumPy y proporciona capacidades de diferenciación automática mediante su módulo jax.grad.

4. MXNet: MXNet es una biblioteca de aprendizaje profundo escalable y flexible desarrollada por Apache Software Foundation. Proporciona diferenciación automática a través de su módulo autograd.

Estas bibliotecas son ampliamente utilizadas en proyectos actuales de machine learning e investigación, y cuentan con una sólida base de usuarios y soporte activo por parte de sus respectivas comunidades.

Ejercicio#6

Aquí tienes tres bibliotecas de diferenciación automática en lenguajes que no son Python:

1. Julia: En el lenguaje de programación Julia, una biblioteca popular para diferenciación automática es ForwardDiff.jl. Proporciona herramientas para calcular derivadas de funciones de manera eficiente utilizando el método hacia adelante.

2. C++: En C++, una biblioteca conocida para diferenciación automática es CppAD (C++ Automatic Differentiation). Es una biblioteca de código abierto que ofrece capacidades tanto para el método hacia adelante como para el método hacia atrás.

3. R: Para el lenguaje de programación R, una biblioteca para diferenciación automática es ADOL-C (Automatic Differentiation by Overloading in C++). Aunque la implementación principal está en C++, se puede utilizar desde R a través de paquetes específicos como adolfr.

Ejercicio#7

La diferenciación automática es fundamental para el entrenamiento y la optimización de las redes neuronales. En el contexto de las redes neuronales, el objetivo principal es ajustar los pesos de las conexiones entre las neuronas para que la red pueda aprender a realizar tareas específicas, como reconocer patrones en datos o realizar predicciones.

La diferenciación automática permite calcular los gradientes de las funciones de error respecto a los pesos de la red de manera eficiente. Estos gradientes indican cómo deben ajustarse los pesos para minimizar el error de la red durante el entrenamiento. Al conocer la dirección y la magnitud en la que deben ajustarse los pesos para mejorar el rendimiento de la red, se puede utilizar un algoritmo de optimización, como el descenso del gradiente, para actualizar iterativamente los pesos de la red y mejorar su desempeño.

En resumen, la diferenciación automática facilita el proceso de entrenamiento de las redes neuronales al proporcionar una forma eficiente de calcular los gradientes necesarios para ajustar los pesos de la red y mejorar su capacidad para realizar tareas específicas.