UNLP. Facultad de Informática

Temario

- Inteligencia artificial no-simbólica: Algoritmos de aprendizaje automático (Machine learning), Redes neuronales artificiales. Algoritmos de entrenamiento de redes neuronales artificiales. Backpropagation

Introducción

Michigrad es un motor de cálculo automático de gradientes para valores escalares diseñado con fines pedagógicos. Funciona con los mismos principios matemáticos que PyTorch o Tensorflow, construyendo el grafo de operaciones de la función de costo y permitiendo calcular el gradiente parcial para cada nodo del grafo, lo que permite aplicar el algoritmo de Backpropagation.

Cada nodo del grafo es un elemento de la clase Value definida en el archivo `michigrad/engine.py/`. Las operaciones sobre Values que están implementadas permiten sumar (`__add__`), multiplicar (`__mul__`) , exponenciar (`exp`) , y calcular la potencia a un número entero (`__pow__`). Además está implementada la función de activación `relu`.

Cada operación sabe como se calcula el gradiente local (en la función local `_backward`), calculando la derivada de la operación. Para la suma, la derivada es constante con lo que solamente se propaga el gradiente. Para la multiplicación, se propaga el gradiente multiplicado por la derivada parcial respecto a cada una de las variables y viceversa. Si tengo -5 * 6, la derivada con respecto a 6 es -5 y con respecto a -5 es 6.

Además, la clase Value implementa la función `backward()` que permite calcular todos los gradientes desde el nodo raíz (generalmente el resultado de la función de pérdida) usando un sort topológico del grafo de operaciones.

Descripción del trabajo

Hacer un fork y clonar el repositorio de código de michigrad visto en clase.
https://github.com/Purrfect-AI/michigrad

Parte 1

Usando la librería MyNN, implementar una red neuronal que prediga la salida de la función XOR. Usar una única capa con dos neuronas lineales (sin ReLU). Graficar el grafo de operaciones de una secuencia de entrenamiento, luego de la etapa de forward y al finalizar la primera pasada de backpropagation.

Iterar varias veces el bucle de entrenamiento:

1. Forward pass
2. Loss
3. Zero grad
4. Backward pass
5. Update

Compruebe que la función de pérdida disminuye en cada iteración, verificando que el modelo está aprendiendo.

Realizar una pequeña explicación de entre 100 y 200 palabras sobre porqué cree que el modelo es incapaz de representar la función correctamente. Si bien esta respuesta es muy personal, puede consultar con sus compañeros antes de elaborar la respuesta. Evite usar un modelo de lenguajes, salvo para la edición.

Parte 2

Extender la clase Value para que soporte otras funciones de activación además de ReLU. Implementar la función Tanh y la función Logística (sigmoide). Aquí la dificultad radica en calcular las derivadas parciales de las funciones en cómo propagarlas a los nodos superiores. Para Tanh por ejemplo, la función está definida como:

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

y su derivada es

$$tanh'(x) = 1 - tanh(x)^2 = 1 - (\frac{e^{2x} - 1}{e^{2x} + 1})^2$$

La función Logística está definida como

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

y su derivada es

$$sigmoid'(x) = \frac{e^{x}}{(1 + e^{-x})^2}$$

Modificar `nn.py` para que ReLU, Tanh y Sigmoid se comporten como capas en lugar de ser un parámetro de `Neuron`.

Parte 3

Rehaga el ejercicio de la función XOR usando ahora usando una capa de dos neuronas con alguna función de activación (ReLU, Tanh o Sigmoid).

Realizar una pequeña explicación de entre 100 y 200 palabras donde interpreta los resultados del modelo entrenado. Como en el caso anterior, si bien esta respuesta es muy personal, puede consultar con sus compañeros antes de elaborar la respuesta. Evite usar un modelo de lenguajes, salvo para la edición.

Referencias

Videos del curso Nociones de Deep Learning para Inteligencia Artificial Generativa de Texto

- Como programar un Autograd desde cero - Clase 1
- Como programar un Autograd desde cero - Clase 2
- Haciendo Backpropagation a mano - Clase 3
- Extendiendo la clase Value - Clase 4
- MyNN, implementando tu propia librería de Redes Neuronales - Clase 1

Videos de Andrej Karpathy sobre Deep Learning

- Neural Networks: Zero to Hero by Andrej Karpathy
  - video 1 - building micrograd
  - Video 5 - makemore part4, becoming backprop ninja

Understanding Deep Learning by Simon J. D. Prince

- Understanding Deep Learning
  - Chapter 3 - Shallow Neural Networks
  - Chepter 4 - Deep Neural Networks
  - Chapter 5 - Loss Functions
  - Chapter 6 - Fitting Models
  - Chapter 7 - Gradient and Initialization (incluye la explicación del algoritmo de Backpropagation).
- Activation Functions Explained