Análisis Matemático para Inteligencia Artificial

Martín Errázquin (merrazquin@fi.uba.ar)

Especialización en Inteligencia Artificial

Diferenciación automática

Regla de la Cadena en forma matricial

Sea $f(x_1(s,t), x_2(s,t))$

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}$$

Y luego

$$\frac{df}{d(s,t)} = \frac{df}{dx}\frac{dx}{d(s,t)} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix} \begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix}$$

Nota: Observar que si bien $\frac{df}{d(s,t)} \in \mathbb{R}^2$, hay que pasar por $\frac{dx}{d(s,t)} \in \mathbb{R}^{2\times 2}$. Podría interesarnos *compilar* el cálculo analítico para ser más eficientes en cómputo.

Derivadas y código

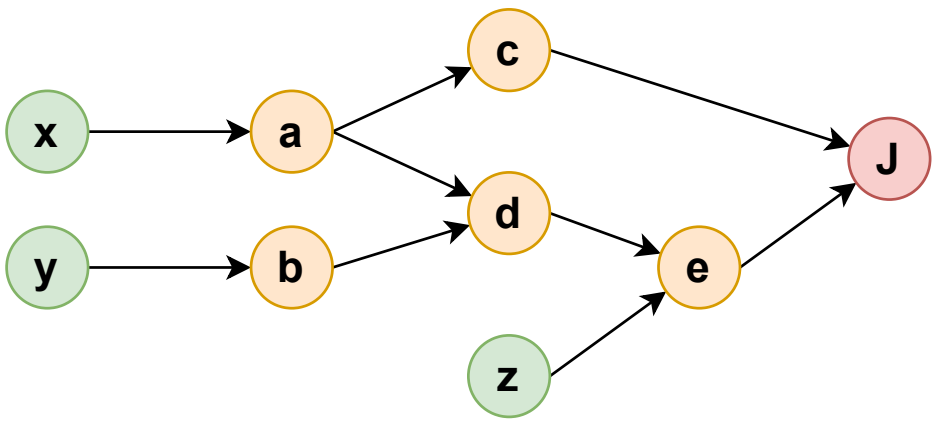
Situación: parte de mi código involucra, para un cierto valor de entrada x, calcular una función `f(x)` y su derivada `df(x)` en ese punto.

- Cálculo 100 % analítico
  - (+) máxima eficiencia de cómputo
  - (−) máxima inflexibilidad de código

- Cálculo 100 % numérico
  - (−) máxima ineficiencia de cómputo
  - (+) máxima flexibilidad de código

Grafo de cómputo

- Uso específico: composición de funciones de diversas entradas y me importa la derivada de la salida "final" respecto de cada entrada.
- Punto medio entre eficiencia y flexibilidad a través de *bloques diferenciables*.
- Cada bloque debe poder calcular la derivada de su salida respecto de todas sus entradas (en gral. analítica).
- Un "orquestador" se ocupa de ir aplicando regla de la cadena.
- ¡Cada bloque podría ser (adentro) una función compuesta!



Ejemplo (simplificado) de código

```
class Product(BaseOperation):
  def f(self, x, y):
      return x * y

  def df(self, x, y):
      return (y, x)


class Cosine(BaseOperation):
  def f(self, x):
      return np.cos(x)

  def df(self, x):
      return -np.sin(x)
```

Ejemplo

Sea $z(x, y) = x^2 \cdot e^{x+cos(y)}$, quiero $\nabla_z(2, \frac{\pi}{2})$.