Neural Ordinary Differential Equations

Esta fuente explica las **Neural ODEs (NODEs)**, una arquitectura avanzada de aprendizaje automático diseñada para modelar sistemas dinámicos mediante **ecuaciones diferenciales ordinarias**. A diferencia de las redes residuales tradicionales que operan en pasos de tiempo discretos, las NODEs representan el sistema de forma **continua**, permitiendo el uso de integradores numéricos sofisticados para obtener mayor precisión. Esta flexibilidad facilita el procesamiento de datos **espaciados irregularmente** y la incorporación de principios físicos, como la conservación de energía, mediante estructuras hamiltonianas o lagrangianas. El autor destaca que la red aprende el **campo vectorial** subyacente, optimizando sus parámetros mediante una técnica matemática llamada **método adjunto** que aprovecha la diferenciación automática. Finalmente, se presenta a las NODEs como una herramienta potente que mejora la robustez frente al ruido y puede combinarse con otros métodos para generar modelos más **interpretables**

- Vídeo explicativo de introducción - Neural ODEs (NODEs) [Physics Informed Machine Learning] by Steve Brunton
- Artículo - Otra introducción con ejemplo en Python

Vídeo explicativo de introducción - Neural ODEs (NODEs) [Physics Informed Machine Learning] by Steve Brunton

https://www.youtube.com/embed/nJphsM4obOk?si=CH9byD8ThHdf1rwX

Resumen del video: Neural ODEs (NODEs)

El video introduce el concepto de **Neural Ordinary Differential Equations (Neural ODEs)**, una arquitectura moderna de *machine learning* propuesta en un artículo muy influyente de 2018. Su objetivo principal es **modelar sistemas dinámicos descritos por ecuaciones diferenciales ordinarias (EDOs)** usando redes neuronales.

1. Idea central de las Neural ODEs

Muchísimos sistemas físicos y dinámicos (péndulos, fluidos, robots, etc.) se describen mediante ecuaciones del tipo:

$$\dot{x} = f(x)$$

La idea clave de las Neural ODEs es **usar una red neuronal para aprender directamente la función $f(x)$**, es decir, el **campo vectorial** que gobierna la dinámica del sistema.

Una vez aprendida esa dinámica continua, se utilizan **métodos numéricos de integración** para simular la evolución del sistema en el tiempo.

---

2. Relación con las ResNets

Las Neural ODEs están fuertemente inspiradas en las **Residual Networks (ResNets)**:

- Una ResNet modela la transición entre dos estados discretos:
  $$x_{k+1} = x_k + f(x_k)$$
- Esto es matemáticamente equivalente a un **paso del método de Euler**, uno de los integradores más simples (y menos precisos) para EDOs.

La interpretación importante es que:

- **Una ResNet equivale a integrar una ecuación diferencial con Euler y un paso de tiempo grande**.
- Esto funciona, pero es numéricamente inestable y poco preciso para sistemas dinámicos complejos.

---

3. Mejora clave de las Neural ODEs

En lugar de aprender directamente el paso discreto ($x_k \to x_{k+1}$), las Neural ODEs:

- Aprenden la **dinámica continua** $ \dot{x} = f(x) $.
- Usan **integradores numéricos más avanzados**, como Runge–Kutta de segundo o cuarto orden.
- Permiten incluso integradores especiales que conservan propiedades físicas (energía, simetrías, etc.).

Esto hace que el modelo sea:

- Más preciso
- Más estable
- Más fiel a la física subyacente

---

4. Ventajas frente a ResNets

Las Neural ODEs tienen varias ventajas importantes:

- **Datos no uniformes en el tiempo**: pueden entrenarse con datos muestreados de forma irregular.
- **Predicciones más suaves y robustas al ruido**.
- **Mejor extrapolación al futuro**, al aprender el campo vectorial real.
- **Separación clara entre modelo dinámico e integrador**, lo que permite intercambiar métodos numéricos fácilmente.

---

5. Entrenamiento y el método adjunto

Entrenar una Neural ODE es más complejo que una red estándar porque:

- Entre dos observaciones existe un **estado continuo oculto**.
- Para optimizar los parámetros de la red se usa un método llamado **adjoint method** (método adjunto).

Este método:

- Introduce una variable auxiliar (multiplicador de Lagrange).
- Permite calcular gradientes **integrando una ecuación diferencial hacia atrás en el tiempo**.
- Se implementa eficientemente gracias a la **autodiferenciación** de las redes neuronales modernas.

Este enfoque conecta las Neural ODEs con ideas clásicas de:

- Teoría de control
- Optimización
- Sistemas dinámicos

---

6. Extensiones físicas: redes con estructura

Una de las grandes fortalezas de las Neural ODEs es que permiten **incorporar estructura física**:

- **Hamiltonian Neural Networks**: conservan la energía total del sistema.
- **Lagrangian Neural Networks**: respetan las ecuaciones de Euler–Lagrange.
- Integradores **simpliciales o variacionales**, ideales para sistemas físicos reales.

Esto las hace especialmente atractivas para *physics-informed machine learning*.

---

7. Interpretabilidad y combinación con otros métodos

Aunque las Neural ODEs son potentes, **no son muy interpretables** porque $f(x)$ es una red neuronal compleja.

Una estrategia práctica es:

1. Entrenar una Neural ODE con datos irregulares.
2. Generar datos regularmente espaciados mediante integración.
3. Aplicar métodos interpretables como:
   - SINDy
   - Regresión simbólica
   - Dynamic Mode Decomposition (DMD)

---

8. Conclusión

Las Neural ODEs:

- Generalizan las ResNets a tiempo continuo.
- Son ideales para modelar sistemas dinámicos reales.
- Permiten usar integradores avanzados y respetar leyes físicas.
- Funcionan bien con datos irregulares y ruidosos.

En resumen, son una **arquitectura muy poderosa para modelar ecuaciones diferenciales con machine learning**, especialmente en problemas físicos y de sistemas dinámicos.

Artículo - Otra introducción con ejemplo en Python

Capítulo 1 de Scientific ML: Desmitificando las ecuaciones diferenciales ordinarias neuronales: un tutorial sencillo