Differential equations in neural networks

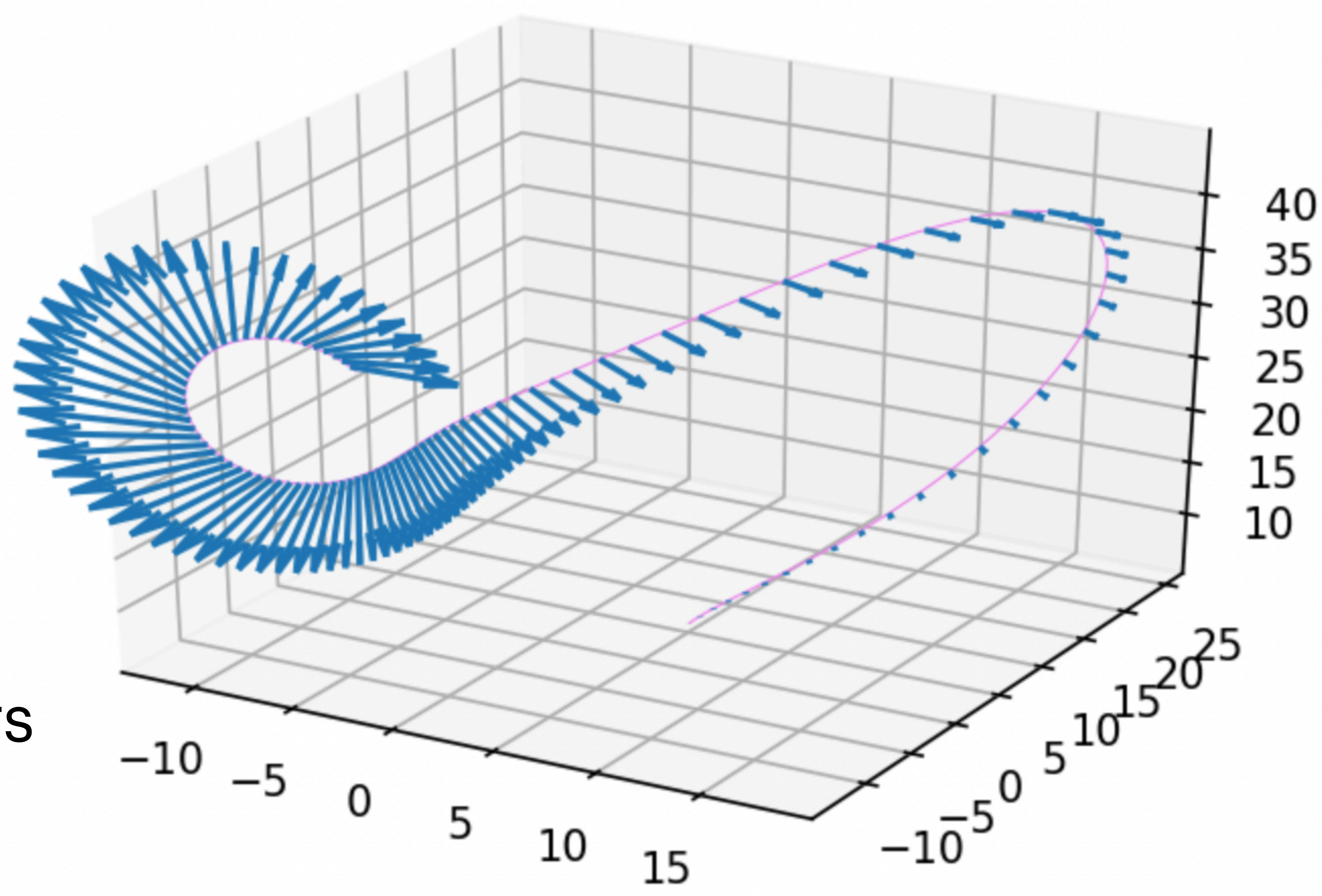
Deep Learning II - uvald2c.github.io

Efstratios Gavves - University of Amsterdam

Overview

Introduction to implicit layers
Forward propagation with implicit layers
Automatic differentiation with implicit layers
Neural ordinary differential equations

http://implicit-layers-tutorial.org/



Credit: http://implicit-layers-tutorial.org/implicit_functions/

Neural network modules as explicit layers

- Neural networks are cascades of module functions h_l: $h_L \circ h_{L-1} \circ \ldots \circ h_1(x, \theta)$
- Each of these module functions must be explicitly defined, that is, there is an explicit line of code describing it
- For instance, the self-attention layer

$$z = \text{softmax}\left(\frac{KQ^T}{\sqrt{n}}\right)V$$

```python
import numpy as np

def self_attention(K,Q,V):
    A = np.exp(K @ Q.T) / np.sqrt(K.shape[1])
    return (A / np.sum(A,1)) @ V

K, Q, V = np.random.randn(3, 5, 4)
print(self_attention(K, Q, V))
```

Implicit layers

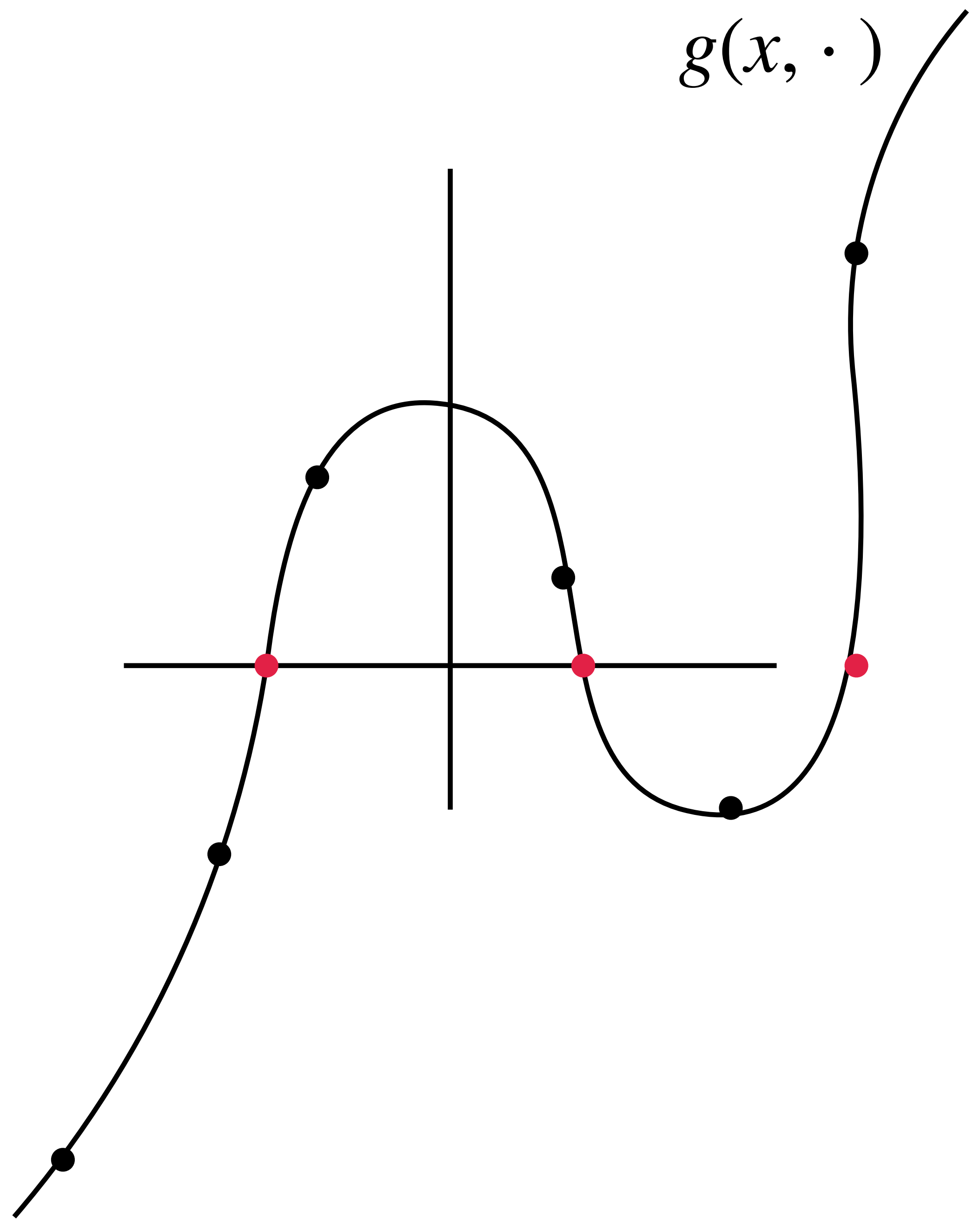
- *Explicit layers* defining what a layer is $\rightarrow$ *Implicit layers* define what a layer does

$$z = f(x)$$

-Explicit layers-

$$\text{return } z : g(x, z) = 0$$

-Finding the root of equation $g(\cdot)$-



- Of course, implicit layers require a root computing algorithm

Properties of implicit layers

- Defining what a layer does → give structure to layer and outputs
- For instance, we can put constraints on types of outputs we obtain
- Any root computing algorithm works: adaptive solvers, quantifying errors, etc
- Differentiable solvers → auto-differentiation and backprop works
- Still, backprop chains might be too long ← *implicit function theorem*

Applications of implicit layers

- Solving arbitrary structured complex problems differentiable
- Solving smooth relaxations of combinatorial optimization problems
- Integrating differential equations as neural network layers
- Architectures for efficient representation of smooth densities