

Neural ODE로 표면을 할 수 없는 (다르게 얘기하면 학습할 수 없는) 함수들이 존재한다. 그 이유는 "ODE trajectories cannot cross each other" 때문이다.

이를 이해하기 위해 이

오류를 그려보자.

$$g_{1d}: \mathbb{R} \rightarrow \mathbb{R}, g_{1d}(-1) = 1$$

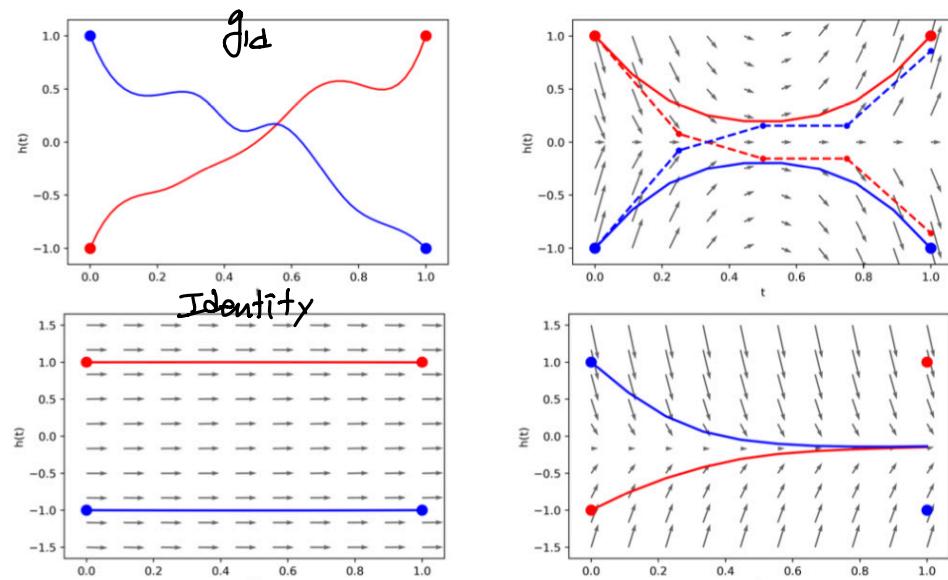
$$g_{1d}(1) = -1 \text{ 이다. 오른쪽}$$

위 그림을 보면 실현이

Neural ODE로 학습한 결과

이다. 학습이 안되는 것을

확인할 수 있다. 예상처럼



Neural ODE는 flow가 교차할 수 없기 때문이다. (이를 homeomorphism orbit 한다) 실제로는 Euler method로 (Resnet이라 보면 됨) 구현하는데 이는 학습이 된다. Resnet은 되는데 Neural ODE는 왜 안될까?

그 이유는 Resnet의 이산화 (discretization)에 이유가 있다.

trajectory가 discrete jump를 할 수 있기 때문에 교차가 가능한 것이다. 차운을 확대해도 중일하는 현상이 일어난다.

$\mathbb{R}^d \rightarrow \mathbb{R}$ を mapping 하는 $g(x)$ 를 생각해보자

$$\begin{cases} g(\mathbf{x}) = -1 & \text{if } \|\mathbf{x}\| \leq r_1 \\ g(\mathbf{x}) = 1 & \text{if } r_2 \leq \|\mathbf{x}\| \leq r_3, \end{cases}$$

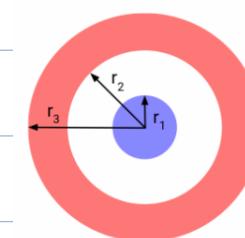


Figure 4: Diagram of $g(\mathbf{x})$ for $d = 2$.

마찬가지로 Neural ODE는 $g(x)$ 를 학습할 수

없다. 예상하면 디전후분이 빨간 부분에

들어있어서 trajectory가 교차해야하기 때문이다.

우리 예시의 경우 여기서 학습을 할 수 있는 하지만 flow가 지나치게 복잡해지는 문제가 발생한다 (NFE 값이 크다)

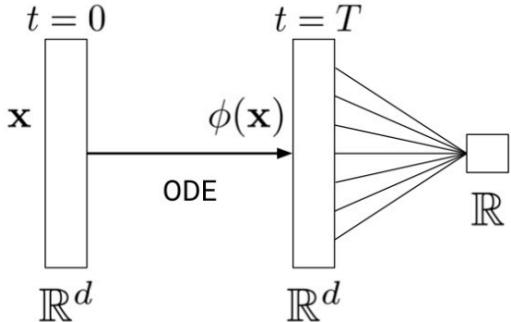
〈해결방법〉

이 솔루션에서 적용하는 방식은 너무 간단하다. 단순히 차원을 늘린다.

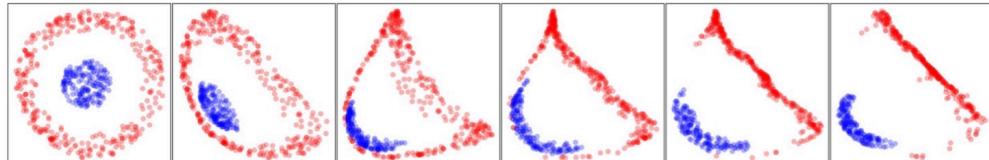
$$R^d \text{ 차원인 } \quad \frac{dh(t)}{dt} = f(h(t), t), \quad h(0) = x \quad \text{예시}$$

$$R^{d+p} \text{ 차원인 } \quad \frac{d}{dt} \begin{bmatrix} h(t) \\ a(t) \end{bmatrix} = f \left(\begin{bmatrix} h(t) \\ a(t) \end{bmatrix}, t \right), \quad \begin{bmatrix} h(0) \\ a(0) \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix} \quad \text{으로 바꾸는 것이다.}$$

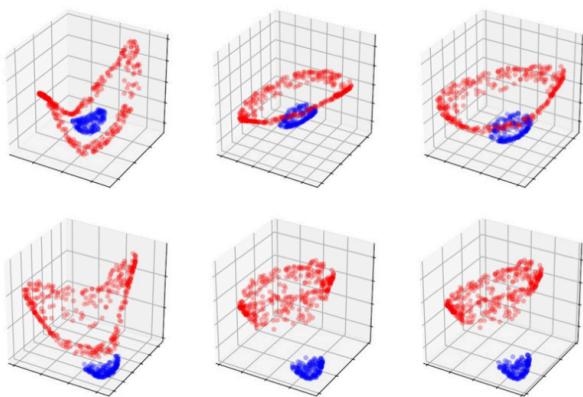
$a(t)$ 는 R^p 차원이다.



왼쪽의 주조에서 R^d 가 R^{d+p} 로 변경된 것을 말고는 동일하다. 이렇게 차원을 augment 해주면 문제가 해결된다.



$f(x)$ 가 위 그림처럼 예외로 학습되던게 (학습방법을)



차원을 추가하게 되면 왼쪽과 같이 간단하게 학습이 되는 것을 확인할 수 있다.

〈장점〉

① Parameter efficiency

차원을 추가해서 학습률 parameter가 높아졌다. augmented 방법을 사용하면 기존 Neural ODE보다 같은 NFE에 같은 loss를 보여 준다. 즉 parameter를 ~~한정적으로~~ 사용한다.

② NFEs and weight decay

Weight decay?

weight decay는 weight들의 값이 증가하는 것을 제한함으로써, 모델의 복잡도를 감소시킴으로써 제한하는 기법, 즉 weight를 decay(부식시킨다라는 의미를 조금 감량시키는 의미로 생각하면 될 것 같습니다.)시켜서 Overfitting을 방지하는 기법으로 소개됩니다.

수학적 생각하면 weight들이 너무 비약적으로 커지는 것을 방지하기 위한 방법 (Lasso or Ridge 같은 ...)

Weight decay를 사용하지 않은 ANODE가 사용한 NODE보다 성능이 뛰어나다. 물론 weight decay를 사용한 ANODE가 성능이 제일 좋다.

③ Accuracy

낮은 cost로 높은 정확도를 보인다.

④ Generalization for images

이미지 데이터셋에서도 잘 된다.

⑤ Stability

NODE에서 timestamp를 너무 줄일 때 발생하는 underflow 가 발생하지 않는다.

⑥ Scaling

class 수를 늘려도 (classification task 경우) 잘 학습한다.