NGUYỄN HOÀNG QUỐC ANH

TOÁN ỨNG DỤNG

2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Hoàng Quốc Anh

ỨNG DỤNG MẠNG NEURON TRONG VIỆC HỌC CÁC HỆ ĐỘNG LỰC

LUẬN VĂN THẠC SĨ TOÁN HỌC

Hà Nội - 2024

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Hoàng Quốc Anh

ỨNG DỤNG MẠNG NEURON TRONG VIỆC HỌC CÁC HỆ ĐỘNG LỰC

LUẬN VĂN THẠC SĨ TOÁN HỌC

Ngành: Toán ứng dụng
Mã số: 8 46 01 12

NGƯỜI HƯỚNG DẪN KHOA HỌC:
TS. Lưu Hoàng Đức

Hà Nội – 2024

Lời cam đoan

Tôi xin cam đoan đề tài nghiên cứu trong luận văn **Ứng dụng mạng neuron trong việc học các hệ động lực** này là công trình nghiên cứu của tôi dựa trên những tài liệu, số liệu do chính tôi tự tìm hiểu và nghiên cứu dưới sự hướng dẫn khoa học của TS. Lưu Hoàng Đức. Chính vì vậy, các kết quả nghiên cứu đảm bảo trung thực và khách quan nhất. Đồng thời, các kết quả này chưa từng xuất hiện trong bất kỳ nghiên cứu nào. Nếu sai phạm, tôi hoàn toàn chịu trách nhiệm trước phát luật.

Tác giả luận văn

Nguyễn Hoàng Quốc Anh

Lời cảm ơn

Dưới góc nhìn của mình, khi nhìn lại quãng thời gian tại Viện Toán học, từ lúc là sinh viên tham gia *Chương trình hướng dẫn nghiên cứu khoa học cho sinh viên Đại học tiềm năng* cho đến hôm nay, tác giả muốn bày tỏ lòng biết ơn sâu sắc tới Trung tâm đào tạo sau Đại học, Viện Toán học, Học viện Khoa học và Công nghệ, cùng với Viện Hàn lâm Khoa học và Công nghệ Việt Nam, cũng như tất cả các thầy cô giáo. Em xin gửi lời cảm ơn chân thành tới các thầy cô đã tận tình giảng dạy, hướng dẫn và hỗ trợ lớp chúng em.

Đặc biệt, tác giả xin dành lời tri ân sâu sắc nhất tới người hướng dẫn, **TS. Lưu Hoàng Đức**, Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam. Em vô cùng biết ơn thầy vì luôn tin tưởng, giúp đỡ và chỉ dạy em từng bước trên con đường học tập. Em xin cảm ơn thầy đã chia sẻ những góc nhìn và truyền cảm hứng cho em.

Tác giả cũng xin bày tỏ lòng biết ơn Quỹ Đổi mới sáng tạo Vingroup (VINIF) đã cấp học bổng *Chương trình học bổng đào tạo Thạc sĩ, Tiến sĩ trong nước*. Đây là sự hỗ trợ và động lực rất lớn, giúp tác giả có thể tập trung vào học tập, nghiên cứu và sáng tạo.

Vì thời gian hạn chế và kiến thức còn nhiều giới hạn, luận văn này không tránh khỏi những thiếu sót. Tác giả rất mong nhận được những ý kiến đóng

góp, nhận xét từ các thầy cô để luận văn có thể hoàn thiện hơn nữa. Tác giả xin chân thành cảm ơn.

Tác giả luận văn

Nguyễn Hoàng Quốc Anh

Mục lục

Danh mục các hình vẽ

Danh mục các bảng

Mở đầu

Các hệ động lực liên tục có tập hút phức tạp thường thể hiện các dáng điệu động lực học phi tuyến rất mạnh và khó dự đoán. Một trong những đặc điểm nổi bật của những hệ thống này là sự thay đổi không đồng đều và phi tuyến của trường vector trong không gian pha. Trường vector này đại diện cho tốc độ và hướng chuyển động của các trạng thái của hệ trong không gian pha. Khi hệ động lực di chuyển từ vùng này sang vùng khác, trường vector có thể thay đổi một cách đột ngột, dẫn đến những dáng điệu rất khác biệt, chẳng hạn như sự xuất hiện các chuyển động tuần hoàn hoặc chuyển động nhiễu loạn.

Việc học và dự đoán hệ động lực có tập hút phức tạp gặp nhiều khó khăn do sự thay đổi trường vector giữa các vùng không gian pha. Trong những vùng khác nhau của không gian pha, quỹ đạo của hệ có thể phản ứng rất khác nhau với các yếu tố đầu vào, điều này đòi hỏi mô hình học máy phải có khả năng thích ứng cao và phân biệt được từng loại hành vi khác nhau. Tuy nhiên, đa phần các mô hình mạng neuron gặp khó khăn trong việc nắm bắt và tổng quát hóa đầy đủ những thay đổi phức tạp này. Ngoài ra, các mô hình mạng neuron thường học dựa trên các mẫu thống kê và sự phụ thuộc vào phân bố của dữ liệu, điều này khiến chúng dễ gặp khó khăn khi phải học một trường vector thay đổi bất ngờ theo từng vùng của không gian pha.

Luận văn này sẽ kiểm tra tính hiệu quả của các phương pháp mạng neuron học hệ động lực tất định liên tục có tập hút phức tạp. Ngoài ra, luận văn cũng đề xuất một phương pháp mới cải thiện khả năng học của phương pháp NeuralODE đối với bộ dữ liệu gồm nhiều quỹ đạo. Luận văn có bố cục gồm 4 chương. Chương 1 trình bày một số kiến thức chuẩn bị về hệ động lực, mạng neuron nhân tạo và phương pháp NeuralODE. Chương 2 trình bày tổng quan về tình hình nghiên cứu trên thế giới và đóng góp của luận văn. Trong chương 3, luận văn kiểm thử các phương pháp mạng neuron MLP và phương pháp NeuralODE lên các bộ dữ liệu khác nhau. Chương 3 cũng đề xuất một phương pháp mới giúp cải thiện việc học hệ

động lực với bộ dữ liệu gồm nhiều quỹ đạo. Chương 4 là một số kết luận và kiến nghị về hướng nghiên cứu tương lai.

Chương 1

Kiến thức nền tảng

1.1 Giới thiệu hệ động lực

1.1.1. Một số khái niệm về hệ động lực

Định nghĩa 1.1. [1] Một hệ động lực là một bộ $(\phi, T, \mathcal{X})$ ở đó $T \in \{\mathbb{N}, \mathbb{Z}, \mathbb{R}_+, \mathbb{R}\}$ là tập hợp thời gian, $\mathcal{X} \neq \emptyset$ là không gian trạng thái và ánh xạ $\phi : T \times \mathcal{X} \to \mathcal{X}$ thoả mãn với mọi $x \in \mathcal{X}$:

$$\begin{aligned} &i, \quad \phi(0, x) = x, \\ &ii, \quad \phi(t, \phi(s, x)) = \phi(t + s, x), \quad \forall t, s \in T. \end{aligned}$$

Nếu $\phi : T \times \mathcal{X} \to \mathcal{X}$ là ánh xạ liên tục, ta gọi hệ động lực là liên tục. Theo quy ước, trong trường hợp $T = \mathbb{N}$ hoặc $\mathbb{Z}$ ta gọi là hệ động lực thời gian rời rạc, trong trường hợp $T = \mathbb{R}$ hoặc $\mathbb{R}_+$ ta gọi là hệ động lực thời gian liên tục.

Với một hệ động lực ϕ và cho trước một điểm ban đầu x_0, ta sẽ gọi $\phi(\cdot, x_0) : T \to \mathcal{X}$ là một **quỹ đạo** xuất phát từ điểm x_0. Ta xem xét một số ví dụ sau:

Phương trình vi phân thường (Ordinary Differential Equations - ODEs)

Cho phương trình vi phân (ODE) dạng vector:

$$\frac{dx}{dt} = f(x), \quad x(0) = x_0 \in \mathbb{R}^n. \tag{1.1}$$

Nghiệm của phương trình 1.1 (nếu tồn tại) là một hàm vector

$$x(t) = (x_1(t), x_2(t), \ldots, x_n(t)) \in \mathbb{R}^n$$

được tham số hóa bởi biến thực $t \in \mathbb{R}$, trong đó miền xác định chung của các hàm tọa độ là một khoảng con $I \subset \mathbb{R}$.

Dạng tổng quát cho các hệ ODE là:

$$\frac{dx}{dt} = f(x, t), \tag{1.2}$$

với biến độc lập t được thêm vào ở phía bên phải. Hàm f trong công thức 1.1 và công thức 1.2 còn được gọi là **trường vector (vector field)** của phương trình ODE. Một ODE (hoặc hệ phương trình) được gọi là **autonomous** nếu trường vector f không phụ thuộc vào t, và **non-autonomous** khi trường vector f phụ thuộc vào t. Ví dụ, phương trình $\frac{dx}{dt} = tx^2 = f(x, t)$ là non-autonomous, trong khi $\frac{dx}{dt} = \frac{x}{4} = f(x)$ là autonomous.

Với phương trình autonomous 1.1, ta có một bài toán giá trị ban đầu (Initial Value Problem - IVP). Trong trường hợp hàm f có tính chất liên tục Lipschitz địa phương và tăng trưởng tuyến tính một phía, có thể chứng minh rằng tồn tại duy nhất một nghiệm của phương trình xuất phát tại thời điểm $t = 0$ tại x_0. Ký hiệu nghiệm này là $\phi(t, x_0)$, khi ấy có thể chứng minh rằng $\phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ là một hệ động lực liên tục.

Phép ánh xạ (Maps)

Cho một tập hợp $\mathcal{X}$ bất kì và một ánh xạ $f : \mathcal{X} \to \mathcal{X}$, ta có thể tạo thành một hệ động lực bằng cách lặp đi lặp lại (từng bước) hàm f lên $\mathcal{X}$. Khi $\mathcal{X}$ là một không gian topo, $f : \mathcal{X} \to \mathcal{X}$ liên tục, f định nghĩa sự tiến hóa (theo cách đệ quy) bằng cách hợp thành với chính nó. Nếu $x \in \mathcal{X}$, định nghĩa $\phi(0, x) = x$ và $\phi(1, x) = f(x)$. Khi đó với mọi $n \in \mathbb{N}$ (các số tự nhiên), ta có:

$$\phi(n, x) = f \circ f \circ \ldots \circ f(x) = f^n(x).$$

Đây là một ví dụ của các hệ động lực rời rạc. Một ví dụ khác về hệ động lực rời rạc phổ biến là các giải thuật tính toán số của các phương trình vi phân được trình bày trong mục 1.1.2..

Ngoài ra, ta có một số định nghĩa sau.

Định nghĩa 1.2. Cho $(\mathcal{X}, \Sigma_1)$ và $(\mathcal{Y}, \Sigma_2)$ là các không gian đo được (measurable space). Khi đó, một ánh xạ $f : \mathcal{X} \to \mathcal{Y}$ được gọi là **đo được (measurable)** nếu với mọi $A \in \Sigma_2$, nghịch ảnh của A qua f thuộc Σ_1. Cụ thể, $\forall A \in \Sigma_2 : f^{-1}(A) = \{x \in \mathcal{X} \mid f(x) \in A\} \in \Sigma_1$.

Định nghĩa 1.3. Cho $(\mathcal{X}, \Sigma)$ là không gian đo được và $f : X \to \mathcal{X}$ là ánh xạ đo được đi từ $\mathcal{X}$ và chính nó. Một độ đo μ trên $(\mathcal{X}, \Sigma)$ được gọi là **bất biến với f (invariant under f)** nếu với mọi tập đo $A \in \Sigma, \mu\left(f^{-1}(A)\right) = \mu(A)$. Ta cũng có thể nói ánh xa f được **bảo toàn (preserve)** dưới độ đo μ.

Định nghĩa 1.4. Một **hệ động lực bảo toàn độ đo (measure-preserving dynamical system)** là một bộ $(\mathcal{X}, \Sigma, \mu, f)$ trong đó:

- $\mathcal{X}$ là một tập hợp,
- Σ là một $\sigma-$đại số của $\mathcal{X}$,
- $\mu : \Sigma \to [0, 1]$ là một độ đo xác suất với $\mu(\mathcal{X}) = 1$ và $\mu(\emptyset) = 0$,
- $f : \mathcal{X} \to \mathcal{X}$ là một ánh xạ đo được được bảo toàn bởi độ đọ μ. Cụ thể, $\forall A \in \Sigma : \mu\left(f^{-1}(A)\right) = \mu(A)$.

Định nghĩa 1.5. Cho $(\mathcal{X}, \Sigma)$ là một không gian đo được, $f : \mathcal{X} \to \mathcal{X}$ là một ánh xạ đo được. Tập đo được $A \in \Sigma$ được gọi là **bất biến (invariant)** khi và chỉ khi $f^{-1}(A) = A$.

Định nghĩa 1.6. Cho hệ động lực bảo toàn độ đo $(\mathcal{X}, \Sigma, \mu, f)$. Khi đó, f được gọi là **ergodic** nếu với mọi tập đo được $A \in \Sigma$ bất biến, thì hoặc $\mu(A) = 0$ hoặc $\mu(A) = 1$.

1.1.2. Giải số phương trình ODE

Xét phương trình ODE:

$$\frac{dx}{dt} = f(x), \quad x(0) = x_0 \tag{1.3}$$

Hầu hết các phương trình ODE không thể được giải hiển, tức là, không thể có một biểu thức tường minh cho $x(t, x_0)$. Thay vào đó, các phương trình vi phân ODE thường được giải số xấp xỉ.

Với bất kỳ giá trị nào của x, $f(x)$ cho biết x sẽ thay đổi như thế nào. Ví dụ, với điều kiện ban đầu $x_0 \equiv x(0)$, $f(x_0)$ cho biết hệ thống đang thay đổi theo hướng nào và nhanh như thế nào. Điều này gợi ý phương pháp giải số xấp xỉ sau đây:

$$x_{n+1} = x_n + \Delta_t \cdot f(x_n), \tag{1.4}$$

trong đó n chỉ mục cho tập hợp $\{x_0, x_1, \ldots\}$ và Δ_t là khoảng thời gian giữa hai lần lặp. Đây là cách giải phương trình vi phân một cách số học nguyên thủy nhất, được gọi là phương pháp Euler [2]. Đạo hàm $\frac{dx}{dt}$ cho biết x thay đổi như thế nào trong một khoảng thời gian vi phân Δ_t đủ nhỏ. Khi đó, ta có thể có được một xấp xỉ tốt cho $x(t)$. Mã giả của phương pháp giải số Euler được cho trong thuật toán 1. Chú ý rằng, kết quả trả về của phương pháp giải số Euler cho ta một hệ động lực rời rạc.

Algorithm 1 Phương pháp giải số Euler

1: **Tham số:** Hàm số f, Δ_t, số vòng lặp n, giá trị ban đầu x_0.
2: **for** i=1,2,...,n **do**
3: $\quad x_i = x_{i-1} + \Delta_t f(x_{t-1})$
4: **end for**
5: **Trả về:** $\{x_i\}_{i=0}^{n}$

1.1.3. Ví dụ một số hệ ODE autonomous

Hệ neuron Fitzhugh-Nagumo

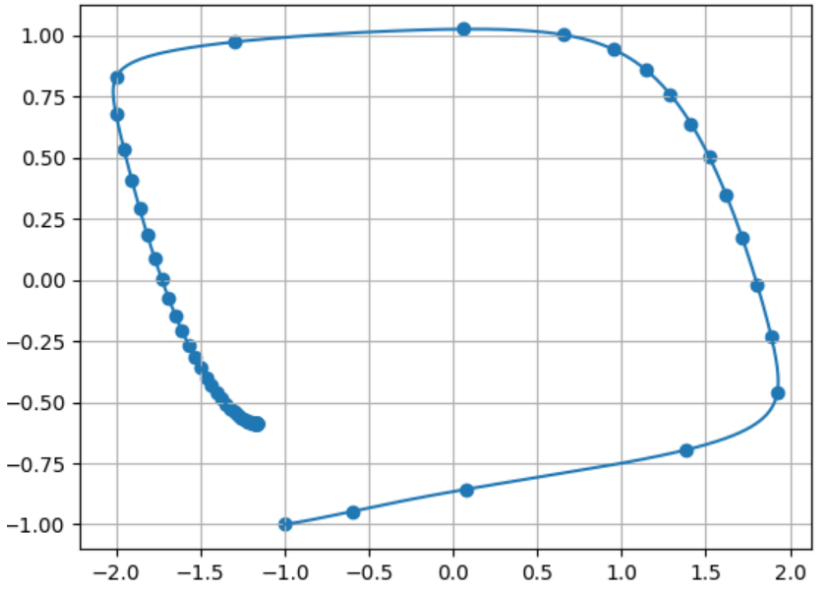
Mô hình FitzHugh–Nagumo (FHN) [3] mô tả một nguyên mẫu của hệ thống 2 chiều có tính kích thích (ví dụ, một neuron). Mô hình này sử dụng các phương trình vi phân để mô tả sự tương tác giữa điện thế màng và một biến trạng thái khác, giúp giải thích các hiện tượng như sự phát sinh xung điện và tính nhạy cảm của neuron.

Phương trình ODE biểu diễn hệ động lực FHN được cho dưới dạng:

$$\begin{aligned} \frac{dv}{dt} &= v - \frac{v^3}{3} - w + RI_{\text{ext}} \\ \tau\frac{dw}{dt} &= v + a - bw \end{aligned} \tag{1.5}$$

Phương trình (1.5) có trường vector f thoả mãn điều kiện Lipschitz địa phương và tăng trưởng tuyến tính một phía, do vậy có thể chứng minh định lý tồn tại và duy nhất nghiệm. Đặc điểm của mô hình FHN là chúng có tập hút (attractor) và có một cấu trúc động lực học khá phức tạp bao gồm các điểm cân bằng (hay điểm cố định) và các đa tạp xung quanh. **Tập hút** (attractor) là tập hợp các trạng thái mà hệ có thể tiến tới khi thời gian tiến tới vô cùng lớn. Tập hút của hệ FHN có các vùng đa tạp chậm (slow manifold) và đa tạp nhanh (fast manifold) do có sự khác biệt trong tốc độ động lực học của các biến trong hệ thống. Đa tạp chậm (Slow Manifold) là vùng trong không gian trạng thái nơi mà các biến hệ thống thay đổi từ từ, phản ánh sự ổn định của hệ. Ngược lại, đa tạp nhanh (Fast Manifold) là vùng các biến thay đổi nhanh chóng, thường liên quan đến các quá trình như phát sinh xung điện trong neuron. Từ đó, ta có sự phân chia giữa các vùng trong không gian trạng thái. Vùng đa tạp chậm tương ứng với các biến đang ổn định, chúng thay đổi chậm, và các trạng thái này thể hiện các điểm cân bằng của hệ thống. Trái lại, ở vùng đa tạp nhanh, các biến thay đổi nhanh trong quá trình phát sinh xung điện. Hệ thống có thể chuyển đổi giữa các trạng thái một cách nhanh chóng, tạo ra các quỹ đạo không ổn định. Ngay cả trường hợp hệ có 1 điểm cân bằng, các vùng ổn định/không ổn

định của hệ cũng nằm rất gần nhau và thậm chí rất gần với điểm cân bằng của hệ. Hình 1.1 cho thấy một quỹ đạo của hệ FHN với các tham số $a = 0.7; b = 0.8; \tau = 12.5; R = 0.1; I_{ext} = 0.5$. Có thể thấy, quỹ đạo di chuyển nhanh ở vùng $[-1, 2] \times [-1, -0.5]$ tương ứng với vùng đa tạp nhanh và di chuyển chậm ở vùng $[-1.5, -1] \times [-0.5, -0.25]$ tương ứng với vùng đa tạp chậm.



Hình 1.1: Quỹ đạo hệ động lực FHN

Động lực học của hệ FHN được xác định bởi mối quan hệ giữa **đường không bậc ba (cubic nullcline)** và **đường không tuyến tính (linear nullcline)**.

Đường không bậc ba được định nghĩa bởi

$$\dot{v} = 0 \leftrightarrow w = v - \frac{v^3}{3} + RI_{\text{ext}}. \tag{1.6}$$

Đường không tuyến tính được định nghĩa bởi

$$\dot{w} = 0 \leftrightarrow w = \frac{v + a}{b}. \tag{1.7}$$

Thông thường, hai đường không cắt nhau tại một hoặc ba điểm, mỗi điểm là một điểm cân bằng. Khi giá trị của $v^2 + w^2$ lớn, xa khỏi gốc tọa độ, dòng chảy xuôi theo chiều kim đồng hồ. Do đó khi có một điểm cân bằng, đó phải là điểm xoáy theo chiều kim đồng hồ hoặc là một nút. Khi có ba điểm cân bằng, chúng phải là hai điểm xoáy theo chiều kim đồng hồ và một điểm yên ngựa.

Khi đường không tuyến tính cắt đường không bậc ba tại ba điểm, hệ FHN có **separatrix**, là hai nhánh của đa tạp ổn định của điểm yên ngựa ở giữa:

- Nếu separatrix là một đường cong, thì các quỹ đạo bên trái của separatrix hội tụ về điểm chìm bên trái, và tương tự cho bên phải.
- Nếu separatrix là một chu kỳ xung quanh giao điểm bên trái, thì các quỹ đạo bên trong separatrix hội tụ về điểm xoáy bên trái. Các quỹ đạo bên ngoài separatrix hội tụ về điểm chìm bên phải. Separatrix chính là chu kỳ giới hạn của nhánh dưới của đa tạp ổn định của điểm yên ngựa ở giữa. Tương tự cho trường hợp separatrix là chu kỳ xung quanh giao điểm bên phải.

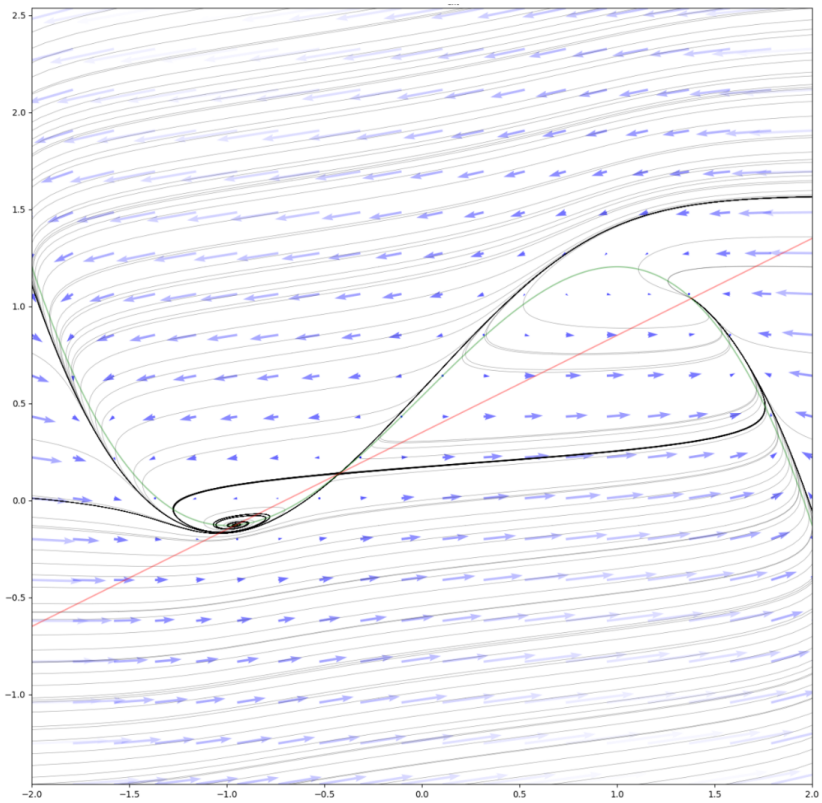
Hình 1.2 mô tả tính chất động lực phức tạp của hệ FHN khi có 3 điểm cân bằng với các tham số $a = 0.7; b = 2.0; \tau = 12.5; R = 0.1; I_{ext} = 5.393$. Ở đây đường màu xanh lá là đường không bậc ba, đường màu đỏ là đường không tuyến tính còn đường màu đen là separatrix.

Hệ động lực Lorenz

Hệ động lực Lorenz [5] là một trong những ví dụ nổi tiếng nhất về hệ thống phi tuyến tính trong lý thuyết động lực học và đã được sử dụng để mô tả hiện tượng thời tiết. Hệ này được phát triển bởi nhà khí tượng học Edward Lorenz vào những năm 1960, và nó là một trong những mô hình đầu tiên cho thấy sự nhạy cảm đối với điều kiện ban đầu, một đặc điểm nổi bật của các hệ thống hỗn loạn.

Hệ Lorenz được mô tả bởi ba phương trình vi phân sau:

$$\begin{aligned}
\frac{dx}{dt} &= \sigma(y - x), \\
\frac{dy}{dt} &= x(\rho - z) - y, \\
\frac{dz}{dt} &= xy - \beta z,
\end{aligned}$$



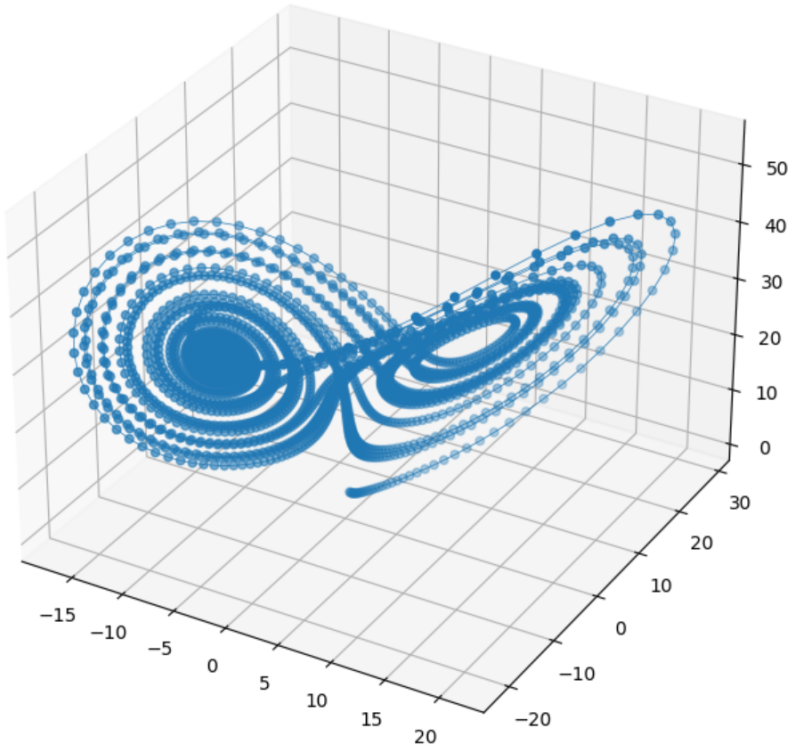
Hình 1.2: Hệ động lực FHN với ba điểm cân bằng (nguồn [4]). Đường màu đỏ là đường không tuyến tính; đường màu xanh lá là đường không bậc ba; đường màu đen là separatrix.

Trong đó:

- x, y, z là các biến mô tả trạng thái của hệ thống.
- σ là hệ số Prandtl, thể hiện sự khuếch tán động lực.
- ρ là một tham số liên quan đến độ ổn định của hệ thống.
- β là hệ số khuếch tán nhiệt.

Hệ Lorenz cho thấy tính chất nhạy cảm với điều kiện ban đầu, nghĩa là những thay đổi rất nhỏ trong điều kiện ban đầu có thể dẫn đến những khác biệt lớn trong hành vi của hệ thống theo thời gian. Đây là một đặc điểm quan trọng của các hệ thống hỗn loạn. Hệ Lorenz có một tập hút nổi tiếng được gọi là tập hút Lorenz, thể hiện một hình dạng phức tạp trong không gian ba chiều. Hai quỹ đạo $\phi(\cdot, x_0)$ và $\phi(\cdot, y_0)$ xuất phát tại các điểm rất gần nhau x_0, y_0 có thể nhanh chóng

tách ra xa nhau trong thời gian ngắn, và một quỹ đạo $\phi(\cdot, x_0)$ bất kì có thể di chuyển đến gần bất cứ một điểm trên tập hút Lorenz vào một thời điểm nào đó. Hình 1.3 cho thấy một quỹ đạo của hệ Lorenz với $\sigma = 10; \rho = 28; \beta = 2.667$.



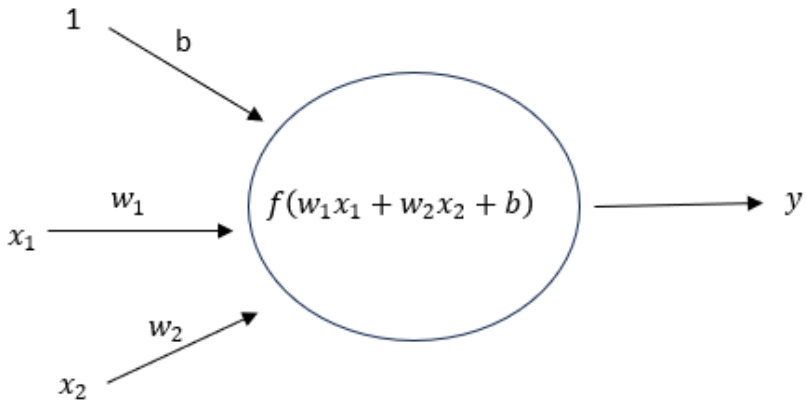
Hình 1.3: Quỹ đạo hệ động lực Lorenz

1.2 Giới thiệu mạng neuron nhân tạo

Mạng neuron nhân tạo (Artificial Neural Network-ANN) là một mô hình tính toán được lấy cảm hứng từ cách mà các mạng neuron sinh học trong não người xử lý thông tin. Các mô hình neuron đang ngày càng trở nên phổ biến trong lĩnh vực máy học. Sự hiệu quả của các mô hình mạng neuron là rất đáng kể và chúng được ứng dụng trên mọi lĩnh vực: xử lý hình ảnh, nhận diện giọng nói hay dự đoán chuỗi thời gian. Mục này sẽ tìm hiểu về một loại mạng neuron nhân tạo cụ thể gọi là Perceptron nhiều lớp (Multi Layer Perceptron).

1.2.1. Cấu trúc của một neuron

Đơn vị tính toán cơ bản trong một mạng neuron là **neuron**, thường được gọi là **nốt** (node). Một neuron sẽ nhận đầu vào từ một số nốt khác hoặc từ một nguồn bên ngoài và tính toán đầu ra. Mỗi đầu vào có một **trọng số** (w) liên quan, được gán dựa trên tầm quan trọng tương đối của nó so với các đầu vào khác. Một neuron có thể biểu diễn như hình 1.4. Mạng neuron trong hình 1.4



Hình 1.4: Cấu trúc một neuron

nhận đầu vào gồm các giá trị x_1, x_2 và sẽ gán trọng số w_1, w_2 tương ứng lên các đầu vào. Ngoài ra, mạng neuron trên cũng nhận thêm một đầu vào có giá trị bằng 1 và gán trọng số b. Trọng số này được gọi là **độ lệch** (bias). Từ hình 1.4, ta có thể thấy rằng đầu ra y của mạng neuron được tính toán thông qua một hàm số phi tuyến f khác. Hàm số f này được gọi **hàm kích hoạt** (Activation function). Mục đích của hàm kích hoạt là đưa tính phi tuyến vào đầu ra của neuron. Điều này quan trọng vì hầu hết dữ liệu trong thế giới thực đều là phi tuyến và ta muốn các neuron học được những biểu diễn phi tuyến này.

Mỗi hàm kích hoạt sẽ nhận đầu vào là một một số thực và thực hiện một phép toán học cố định trên nó. Một số hàm kích hoạt hay được sử dụng trong mạng neuron gồm có:

- **Hàm Sigmoid**: nhận giá trị là một số thực và trả ra giá trị nằm trong

khoảng $(0, 1)$

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1.8}$$

- **Hàm Tanh**: nhận giá trị là một số thực và trả giá trị nằm trong khoảng $[-1, 1]$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.9}$$

- **Hàm ReLu**: nhận giá trị là một số thực và thay thế các giá trị âm bằng 0

$$R(x) = \max(0, x) \tag{1.10}$$

- **Hàm ELU**: nhận giá trị là một số thực và trả về các giá trị lớn hơn 0.

$$\begin{aligned} f(x) &= x \text{ nếu } x > 0 \\ \alpha(\exp(x) - 1) &\text{ nếu } x \leq 0 \end{aligned} \tag{1.11}$$

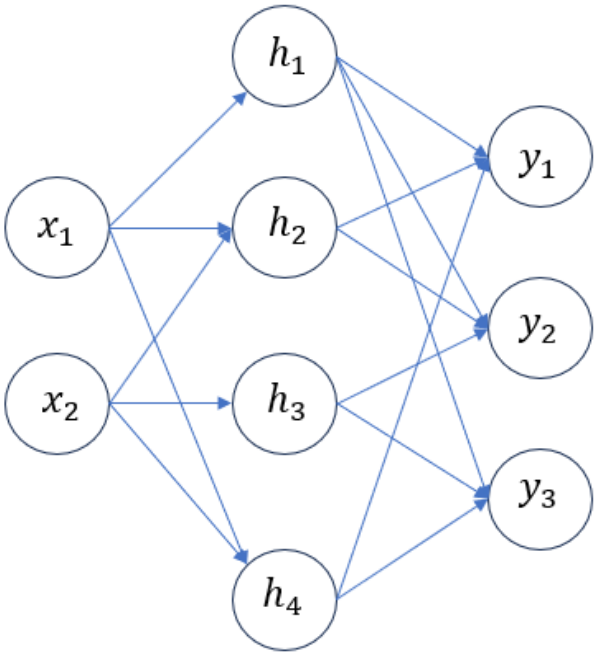
trong đó $\alpha > 0$.

1.2.2. Mạng Perceptron nhiều lớp

Trước khi đi đến khái niệm của mạng Perceptron nhiều lớp (Multilayer Perceptron - MLP), ta cần có khái niệm của mạng neuron truyền thẳng (Feedforward Neural Neutwork - FNN). Mạng neuron truyền thẳng là loại mạng neuron nhân tạo đầu tiên và đơn giản nhất được thiết kế. Mạng bao gồm nhiều neuron được sắp xếp theo các **lớp** (layer). Các nốt từ các lớp liền kề có các **kết nối** (connections) hoặc **cạnh** (edges) giữa chúng. Tất cả các kết nối này đều có các trọng số đi kèm biểu thị độ liên kết giữa các nốt. Ví dụ về mạng neuron truyền thẳng có thể thấy trong hình 1.5:

Một mạng neuron truyền thẳng có thể bao gồm ba loại nốt:

- **Nốt đầu vào** (Input Nodes): Các nốt đầu vào cung cấp thông tin từ bên ngoài vào mạng. Tập hợp các nốt đầu vào tạo thành **Lớp Đầu Vào** (Input Layer). Không có phép toán nào được thực hiện tại các nốt đầu vào. Thông



Hình 1.5: Cấu trúc một mạng neuron truyền thẳng

tin của nốt đầu vào sẽ truyền đến các nốt ẩn trong mạng. Trong hình 1.5, x_1, x_2 là hai nốt đầu vào mang thông tin từ bên ngoài.

- **Nốt đầu ra** (Output Nodes): Tập hợp các nốt đầu ra cấu tạo nên **Lớp Đầu Ra** (Output Layer). Các nốt đầu ra sẽ chứa thông tin được tính toán từ mạng neuron và truyền thông tin từ mạng ra thế giới bên ngoài. Trong hình 1.5, y_1, y_2, y_3 là ba nốt đầu ra của mạng neuron truyền thẳng.
- **Nốt ẩn** (Hidden Nodes): Các nốt ẩn sẽ không có kết nối trực tiếp với thông tin từ bên ngoài (do đó có tên gọi là "ẩn") cũng như không thể truyền trực tiếp thông tin ra bên ngoài. Các nốt ẩn thực hiện các phép toán và truyền thông tin từ các nốt đầu vào đến các nốt đầu ra. Một tập hợp các nốt ẩn tạo thành một **Lớp Ẩn** (Hidden Layer). Chú ý rằng, một mạng truyền thẳng chỉ có một lớp đầu vào và một lớp đầu ra nhưng có thể có không hoặc nhiều lớp ẩn. Trong hình 1.5, mạng neuron có một lớp ẩn với các nút ẩn tương ứng là h_1, h_2, h_3, h_4.

Trong một mạng truyền thẳng, thông tin chỉ di chuyển theo một hướng – tiến về phía trước – từ các nốt đầu vào, qua các nốt ẩn (nếu có) và đến các nốt đầu ra. Tuy nhiên một vài biến thể khác của mạng neuron cho phép thông tin có thể truyền lặp lại trong mạng, ví dụ mạng neuron hồi quy (Recurrent Neural Network).

Mạng Perceptron nhiều lớp (Multilayer Perceptron - MLP) là mạng neuron truyền thẳng có một hoặc nhiều lớp ẩn. Hình 1.5 là ví dụ của mạng Perceptron một lớp. Dựa vào cấu trúc của mạng như trong hình 1.5 ta có thể tính giá trị các nốt như sau:

$$\begin{aligned} h_1 &= \sigma(w_1 x_1 + b_1) \\ h_2 &= \sigma(w_2 x_1 + w_3 x_2 + b_2) \\ h_3 &= \sigma(w_4 x_2 + b_3) \\ h_4 &= \sigma(w_5 x_1 + w_6 x_2 + b_4) \\ y_1 &= \sigma(w_7 h_1 + w_8 h_2 + w_9 h_4 + b_5) \\ y_2 &= \sigma(w_{10} h_2 + w_{11} h_3 + b_6) \\ y_3 &= \sigma(w_{12} h_1 + w_{13} h_4 + b_7) \end{aligned} \tag{1.12}$$

trong đó $\sigma(.)$ là một hàm kích hoạt bất kì, $\{w_i\}_{i=1}^{12}$ là các trọng số, $\{b_i\}_{i=1}^{7}$ là các độ lệch. Chú ý rằng, ta có thể áp dụng các hàm kích hoạt khác nhau tại mỗi nốt của mạng neuron. Tổng quát lại, nếu kí hiệu đầu vào của lớp thứ i là $\mathbf{x_i} \in \mathbb{R}^{m_{i-1}}$ và đầu ra của lớp thứ i là $\mathbf{y_i} \in \mathbb{R}^{m_i}$, công thức cập nhật được biểu diễn như sau:

$$\mathbf{y_i} = \mathbf{f_i}(\mathbf{x_i}) = \sigma_\mathbf{i}(\mathbf{W_i x_i} + \mathbf{b_i}) \tag{1.13}$$

trong đó $\sigma_\mathbf{i} = \begin{bmatrix} \sigma_{i_1} \\ \vdots \\ \sigma_{i_{m_i}} \end{bmatrix}$ chứa các hàm kích hoạt từng phần tử của lớp thứ i, m_i là số nốt của lớp thứ i, $\mathbf{W_i} \in \mathbb{R}^{m_i \times m_{i-1}}$ là ma trận trọng số, $\mathbf{b_i} \in \mathbb{R}^{m_i}$ là vector độ lệch.

Ngoài ra ta cũng có thêm 2 định nghĩa sau:

- **Chiều rộng của mạng neuron**: là số nốt tối đa trong một lớp của mạng neuron. Kí hiệu chiều rộng của mạng neuron là W. Công thức tính sẽ là $W = \max_i m_i$ trong đó m_i là số nốt của lớp thứ i.
- **Chiều dài của mạng neuron**: là số lớp ẩn của mạng neuron. Kí hiệu chiều dài của mạng neuron là D.

1.2.3. Huấn luyện mạng neuron

Để huấn luyện mạng neuron nói chung hay mạng MLP nói riêng, ta cần phải có bộ dữ liệu huấn luyện $\mathcal{D}$. Tùy thuộc vào tính chất của bài toán mà bộ dữ liệu $\mathcal{D}$ có nhãn hoặc không có nhãn. Trong trường hợp bộ dữ liệu $\mathcal{D}$ có nhãn, bài toán đó được gọi là *Học có giám sát* (Supervised Learning). Ngược lại, trong trường hợp bộ dữ liệu không có nhãn, bài toán đó được gọi là *Học không giám sát* (Unsupervised Learning). Bộ dữ liệu huấn luyện $\mathcal{D}$ có nhãn sẽ bao gồm các điểm dữ liệu huấn luyện (x_i, y_i), trong đó y_i được gọi là nhãn của x_i. Nếu bộ dữ liệu $\mathcal{D}$ có N điểm dữ liệu, ta có thể viết $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. Thông thường, nhãn y_i là một số vô hướng $(y_i \in \mathbb{R})$. Mặt khác, bộ dữ liệu $\mathcal{D}$ không có nhãn sẽ chỉ bao gồm các điểm dữ liệu $\{x_i\}_{i=1}^N$. Trong mục này, giả thiết rằng bộ dữ liệu có nhãn và nhãn dữ liệu là một số vô hướng.

Trong kiến trúc mạng neuron, mỗi lớp nhận đầu ra từ lớp trước, xử lý thông tin đầu vào đó và truyền kết quả cho lớp tiếp theo. Cuối cùng, mạng neuron sẽ trả kết quả dự đoán $\hat{y}_i$ tại lớp đầu ra. Để đánh giá độ chính xác của kết quả dự đoán, dựa trên bộ dữ liệu $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, hàm **tổn thất** (Loss function) sẽ được sử dụng để định lượng sai số của phương pháp mạng neuron. Tùy thuộc vào từng bài toán mà hàm tổn thất sẽ được định nghĩa khác nhau. Một trong những hàm tổn thất phổ biến nhất hay được sử dụng là hàm trung bình bình phương sai số (Mean Square Error - MSE):

$$L = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{1.14}$$

trong đó N là số điểm dữ liệu ta có, $\hat{y}_i$ là giá trị dự đoán của mạng neuron và y_i là giá trị thực tế. Giá trị hàm tổn thất lớn đồng nghĩa với việc dự đoán của mạng neuron đang chưa chính xác. Do đó, mạng neuron cần phải được huấn luyện để tối thiểu hàm tổn thất, từ đó khiến cho mạng neuron dự đoán chính xác hơn. Cụ thể, việc huấn luyện mạng neuron sẽ đưa về bài toán tối ưu sau:

$$\text{argmin}_\theta \, L(\theta) \tag{1.15}$$

trong đó θ bao gồm các trọng số của mạng neuron. Ví dụ đối với mạng neuron được cập nhật trong công thức (1.12), các trọng số cần được học sẽ là $\theta = \{w_1, w_2, .., w_{13}, b_1, b_2, ..., b_7\}$. Do đó, ngoài việc phải phản ánh đúng ý nghĩa của bài toán, hàm tổn thất sẽ đồng thời ảnh hưởng trực tiếp đến quá trình huấn luyện mạng neuron. Việc thiết kế hàm tổn thất phù hợp là một chủ đề lớn và đang được nghiên cứu rộng rãi hiện nay [6].

1.2.4. Thuật toán Adam

Adam là một thuật toán tối ưu hóa hiệu quả và phổ biến, thường được sử dụng trong học sâu. Thuật toán này kết hợp các ưu điểm của **Momentum** (làm mượt gradient) và **RMSProp** (điều chỉnh tốc độ học theo từng tham số). Nhờ đó, Adam đảm bảo hội tụ nhanh và ổn định ngay cả khi gradient có độ lớn không đồng nhất.

Adam sử dụng hai moment động:

- **Moment bậc 1**: Trung bình động của gradient, đại diện cho tốc độ thay đổi tham số.
- **Moment bậc 2**: Trung bình động của bình phương gradient, đại diện cho độ lớn gradient.

Dựa vào hai moment động, thuật toán Adam được mô tả qua các bước sau:

1. **Khởi tạo**:

- Tham số ban đầu θ_0, moment bậc 1 $m_0 = 0$, moment bậc 2 $v_0 = 0$.
- Siêu tham số: tốc độ học α, hệ số giảm tốc β_1, β_2 (thường là 0.9 và 0.999), và ϵ để tránh chia cho 0.

2. **Cập nhật moment**:
   - Moment bậc 1: $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$.
   - Moment bậc 2: $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$.
3. **Hiệu chỉnh bias**: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$.
4. **Cập nhật tham số**:$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$.
5. **Lặp lại**: Tiếp tục các bước trên cho đến khi hội tụ.

Ưu điểm của phương pháp Adam là:

- Điều chỉnh tốc độ học theo từng tham số, phù hợp với gradient có độ lớn không đồng nhất.
- Hội tụ nhanh và ổn định, ngay cả khi dữ liệu nhiễu.
- Ít phụ thuộc vào việc điều chỉnh siêu tham số.

Adam được sử dụng rộng rãi trong học sâu, từ huấn luyện mạng nơ-ron tích chập (CNNs), mạng nơ-ron tuần tự (RNNs), đến các mô hình hiện đại như Transformer.

1.2.5. Định lý xấp xỉ toàn cục (Universal Approximation Theorem)

Định lý xấp xỉ toàn cục nghiên cứu về khả năng xấp xỉ hàm số của mạng MLP. Đã có rất nhiều bài báo nghiên cứu về định lý xấp xỉ toàn cục cho mạng MLP. Thông thường, sẽ có hai hướng tiếp cận cho định lý xấp xỉ toàn cục:

- **Nghiên cứu về chiều rộng của MLP**: hướng tiếp cận này nghiên cứu khả năng xấp xỉ của mạng neuron với chiều dài hạn chế.

- **Nghiên cứu về chiều dài của MLP**: hướng tiếp cận này nghiên cứu khả năng xấp xỉ của mạng neuron với chiều rộng hạn chế.

Mục này sẽ giới thiệu hai định lý xấp xỉ toàn cục cho hai hướng tiếp cận trên. Các định lý phát biểu khác có thể tham khảo thêm ở bài báo [7].

Nghiên cứu về chiều rộng của mạng neuron

Định lí 1.7. *[8] Cho $\mathcal{X}$ là tập compact bất kì thuộc $\mathbb{R}^n$ và S là hàm kích hoạt sigmoid. Xét tổng hữu hạn sau:*

$$f_{NN}(x) = \mathbf{W_2} S(\mathbf{W_1} x + \mathbf{b_1}) \tag{1.16}$$

trong đó $\mathbf{W_1} \in \mathbb{R}^{m_1 \times n}, \mathbf{W_2} \in \mathbb{R}^{1 \times m_1}, \mathbf{b_1} \in \mathbb{R}^{m_1}$. Khi đó, với bất kì hàm liên tục $f : \mathcal{X} \to \mathbb{R}$ và $\epsilon > 0$, tồn tại bộ chỉ số $\mathbf{W_2}, \mathbf{W_1}, \mathbf{b_1}$ thỏa mãn: $\mid f(\mathbf{x}) - f_{NN}(\mathbf{x}) \mid \leq \epsilon$ với mọi $x \in \mathcal{X}$.

Định lý 1.7 chỉ ra rằng mạng neuron với một lớp ẩn và hàm kích hoạt sigmoid có thể xấp xỉ bất kì hàm liên tục liên tục nào trong tập compact với sai số tùy ý.

Nghiên cứu về chiều dài của mạng neuron

Định lí 1.8. *[9] Với bất kì một hàm số khả tích Lebesgue $f : \mathbb{R}^n \to \mathbb{R}$ và $\epsilon > 0$, tồn tại một mạng neuron f_{NN} với chiều dài $W \leq n + 4$ và hàm kích hoạt ReLU thỏa mãn:*

$$\int \mid f(\mathbf{x}) - f_{NN}(\mathbf{x}) \mid d\mathbf{x} < \epsilon \tag{1.17}$$

Định lý 1.8 chỉ ra rằng mạng neuron với số lượng lớp ẩn bất kỳ và tối đa $n + 4$ neuron mỗi lớp có thể xấp xỉ bất kỳ hàm nào khả tích Lebesgue với độ chính xác đủ cao.

1.3 Giới thiệu NeuralODE

NeuralODE là một lớp các mô hình mạng neuron được lấy ý tưởng dựa trên phương trình vi phân. NeuralODE [10] được giới thiệu lần đầu vào năm 2018 và

đạt giải bài báo xuất sắc nhất của hội nghị NeuRIPS 2018. NeuralODE tỏ ra hiệu quả khi học các hệ động lực [11], các phép biến đổi vi phôi (diffeomorphism) [12] và đặc biệt có thể sử dụng một hệ thống lý thuyết đồ sộ từ phương trình vi phân để phân tích các đặc điểm, tính chất của phương pháp học máy [13]. Điểm khác biệt giữa NeuralODE và MLP là số lớp ẩn. Trong kiến trúc của MLP, số lớp ẩn là hữu hạn. Tuy nhiên, số lớp ẩn trong mô hình NeuralODE là vô hạn. Cụ thể, ta có thể biểu diễn sự truyền thông tin giữa các lớp ẩn dưới dạng phương trình ODE sau:

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t) \tag{1.18}$$

trong đó $\mathbf{h}(t) \in \mathbb{R}^D$ là trạng thái của lớp ẩn ở thời điểm t, $f_\theta(.,.)$ là một mạng neuron với trọng số θ. Chú ý rằng, trong NeuralODE, số chiều của lớp ẩn ở mọi thời điểm t là bằng nhau. Trong khi đó số chiều của lớp ẩn ở trong kiến trúc MLP có thể khác nhau. Đầu vào của NeuralODE sẽ được coi là trạng thái của lớp ẩn ở thời điểm ban đầu $\mathbf{h}(0)$ và đầu ra của NeuralODE là trạng thái của lớp ẩn ở thời điểm cuối $\mathbf{h}(T)$. Do vậy, có thể coi đầu ra của phương pháp NeuralODE là nghiệm của bài toán IVP với trường vector f_θ tại thời điểm T. Để tính giá trị $\mathbf{h}(T)$, ta có thể sử dụng các bộ giải số phương trình vi phân cấp 1.

Tuy nhiên, việc sử dụng bộ giải số để tính đầu ra cho NeuralODE gây khó cho quá trình huấn luyện mô hình bởi lẽ bộ giải số không cung cấp cách tính đạo hàm. Để vượt qua vấn đề này, Chen và các cộng sự [10] sử dụng phương pháp *adjoint sensitivity method* cho việc huấn luyện NeuralODE. Phương pháp adjoint tính đạo hàm bằng cách giải một phương trình vi phân khác. Cụ thể, giả sử ta có một hàm tổn thất $L()$ nhận đầu vào là đầu ra của của NeuralODE:

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f_\theta(\mathbf{z}(t), t)dt\right) = L\left(\text{ODESolve}\left(\mathbf{z}(t_0), f_\theta, t_0, t_1\right)\right) \tag{1.19}$$

trong đó ODESolve (z, f, t_0, t_1) kí hiệu cho bộ giải số phương trình vi phân từ thời điểm ban đầu t_0 tới thời điểm cuối t_1 với trường vector f_θ và trạng thái ở

thời gian ban đầu là z.

Để tối ưu L, ta sẽ phải tính đạo hàm của L theo trọng số θ. Trước hết, đạo hàm của L cho từng lớp ẩn $\mathbf{z}(t)$ sẽ được tính. Đại lượng này được gọi là *adjoint* và kí hiệu bởi $\mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{z}(t)}$. [10] đã chứng minh được rằng $\mathbf{a}(t)$ sẽ tuân theo một phương trình vi phân khác có dạng:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f_\theta\left(\mathbf{z}(t), t\right)}{\partial \mathbf{z}} \tag{1.20}$$

Dựa vào phương trình vi phân 1.20, có thể tính được $\frac{\partial L}{\partial \mathbf{z}(t_0)}$ bằng bộ giải số phương trình vi phân với giá trị ban đầu là $\frac{\partial L}{\partial \mathbf{z}(t_1)}$ và ta sẽ giải ngược từ t_1 về t_0. Để giải số được phương trình vi phân 1.20, ta sẽ cần phải biết giá trị của $\mathbf{z}(t)$ trên toàn bộ quỹ đạo từ t_0 đến t_1. Ta có thể giải ngược $\mathbf{z}(t)$ cùng với đại lượng adjoint $\mathbf{a}(t)$ cùng một lúc với giá trị ban đầu tương ứng là $\mathbf{z}(t_1)$ và $\frac{\partial Ll}{\partial \mathbf{z}(t_1)}$.

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\partial L / \partial \mathbf{z}(t_1)$
$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state
def aug_dynamics($[\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta$): ▷ Define dynamics on augmented state
  return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products
$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE
return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

Hình 1.6: Thuật toán tính đạo hàm cho mô hình NeuralODE

Sau khi tính được đại lượng adjoint, đạo hàm của L theo trọng số θ được tính bằng công thức:

$$\frac{\partial L}{\partial \theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^T \frac{\partial f_\theta\left(\mathbf{z}(t), t\right)}{\partial \theta} dt \tag{1.21}$$

Tất cả tích phân để giải $\mathbf{z}, \mathbf{a}$ và $\frac{\partial L}{\partial \theta}$ đều có thể được tính bởi một lần gọi bộ giải phương trình vi phân trong đó ta sẽ ghép trạng thái lớp ẩn, đại lượng adjoint và đạo hàm trọng số thành một vector trạng thái. Chen và các cộng sự [10] đề xuất thuật toán tính đạo hàm được thể hiện trong hình 1.6.

Chương 2

Tình hình nghiên cứu trên thế giới và đóng góp của luận văn

2.1 Học hệ động lực tất định

Sử dụng các mô hình học máy để học hệ động lực là một chủ đề lớn được các nhà khoa học nghiên cứu rộng rãi. Nổi tiếng nhất là công trình của Chen và các cộng sự [10] về phương pháp NeuralODE. Phương pháp NeuralODE có thể học tốt các hệ động lực nhờ mô hình hóa mạng neuron tuân theo phương trình ODE. Việc sử dụng bộ giải số ODE cho việc huấn luyện phương pháp NeuralODE là đóng góp lớn nhất của bài báo, làm tiền đề cho các kết quả khác sau này. Các bài nghiên cứu [11] [14] đề xuất một mô hình đảm bảo tính ổn định (stability) của hệ động lực. Cụ thể, các phương pháp đề xuất học đồng thời trường vector và hàm Lyapunov tương ứng. Trường vector sau đó sẽ được chiếu xuống vùng thỏa mãn điều kiện Lyapunov, đảm bảo điều kiện cân bằng của hệ động lực được học. Cũng học hệ động lực ổn định, tuy nhiên Wenxin cùng các cộng sự [15] đề xuất học trực tiếp hệ động lực lẫn hàm Lyapunov bằng quá trình Gaussian (Gaussian Process). Ưu điểm của phương pháp là chúng cho ta biết khoảng tin cậy của các điểm dự báo. Tuy nhiên, các phương pháp sử dụng quá trình Gaussian luôn đòi hỏi phải xác định một hàm nhân (kernel function) và độ hiệu quả phương pháp ảnh hưởng nhiều vào hàm nhân này. Beik-Mohammadi

cùng các cộng sự [16] đề xuất một phương pháp học hệ động lực có tính chất "contraction". Thay vì học trực tiếp trường vector, nhóm tác giả học ma trận Jacobi của trường vector đó. Mô hình mạng neuron được thiết kế sao cho ma trận Jacobi luôn là ma trận xác định âm. Khi đó, hệ động lực học được luôn thỏa mãn điều kiện "contraction". Dựa trên giả thiết hệ động lực tuyến tính dễ học hơn các hệ động lực khác, hai bài nghiên cứu của Fichera [17] và Rana [18] đề xuất xây dựng hệ động lực tuyến tính rồi sau đó biến đổi ngược trở lại hệ động lực gốc bằng một phép vi phôi. Thay vì học hệ động lực, các phương pháp này sẽ học phép biến đổi vi phôi sao cho phép ánh xạ ngược có sai số nhỏ nhất. Tuy vậy, nhược điểm của phương pháp là phải xác định trước hệ động lực tuyến tính. Các phương pháp kể trên đều áp dụng cho một lớp các hệ động lực cụ thể. Tuy nhiên, theo hiểu biết của chúng tôi, các phương pháp trên chưa xây dựng cho hệ động lực có tập hút phức tạp mà ở đó trường vector các điểm dữ liệu khác nhau lớn.

2.2 Học hệ động lực ngẫu nhiên

Mục này sẽ giới thiệu một số phương pháp học hệ động lực ngẫu nhiên, cụ thể cho phương trình vi phân ngẫu nhiên (Stochastic Differential Equation - SDE). SDE là một công cụ toán học có thể mô hình các hệ động lực có tính ngẫu nhiên. Công thức tổng quát của SDE có thể được viết dưới dạng:

$$\underbrace{\frac{dx}{dt} = f(t,x)}_{\text{ODE}} + \underbrace{g(t,x)Z(t)}_{\text{nhiễu ngẫu nhiên}}. \tag{2.1}$$

trong đó $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ thường được gọi là **hàm dịch chuyển (drift function)**; $g : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d \times m}$ thường được gọi là **hàm khuếch tán (diffusion function)** và $Z(t)$ là một quá trình ngẫu nhiên m chiều. Thông thường, $Z(t) = dW(t)$ với $W(t)$ là quá trình Weiner (Weiner process) m chiều. Ngoài ra, nhiều bài toán cũng giả thiết hàm g chỉ phụ thuộc vào biến thời gian t. Khi đó, phương

trình 2.1 được viết lại dưới dạng:

$$\frac{dx}{dt} = f(t, x) + g(t)dW(t) \tag{2.2}$$

Phương trình SDE 2.2 có nhiễu cộng tính (additive noise). Trong những năm gần đây, các mô hình SDE nhận được nhiều sự chú ý từ cộng đồng nghiên cứu học máy. SDE có thể được sử dụng để phân tích các đặc điểm hành vi của các mô hình học máy [19], hoặc là nền tảng để xây dựng các mô hình dự báo chuỗi thời gian [20], hoặc là nền tảng để xây dựng các mô hình sinh [21].

Mặt khác, một số ứng dụng thực tế yêu cầu xác định phương trình SDE dựa trên một tập dữ liệu, cụ thể xác định hàm dịch chuyển, hàm khuếch tán và thậm chí là xác định quá trình ngẫu nhiên. Ví dụ phổ biến nhất có lẽ là cho các mô hình tài chính khi đầu vào là các quan sát của thị trường tài chính (ví dụ giá) và đầu ra là mô hình tài chính phù hợp với quan sát đó. Tuy vậy, việc xác định chính xác hàm dịch chuyển, hàm khuếch tán chỉ dựa trên bộ dữ liệu quan sát là một điều thách thức. Đã có nhiều bài báo nghiên cứu về vấn đề này thông qua các phương pháp học máy. Kidger và các cộng sự của mình [22] đề xuất một thuật toán học SDE thông qua phương pháp GAN [23]. Phương pháp GANs sẽ có hai mô hình: mô hình sinh (generator) và mô hình phân biệt (discriminator). Trong bài báo của Kidger và các cộng sự [22], mô hình sinh được sử dụng để tìm hàm dịch chuyển, hàm khuếch tán và sinh ra một quỹ đạo ngẫu nhiên. Sau đó, mô hình phân biệt dự đoán đâu là quỹ đạo thực của hệ SDE gốc và đâu là quỹ đạo của mô hình sinh. Mục tiêu của phương pháp huấn luyện là khiến cho mô hình sinh tái tạo được quỹ đạo giống với hệ SDE gốc nhất có thể và mô hinh phân biệt phân loại được đâu là quỹ đạo gốc cao nhất có thể. Dietrich và các cộng sự [24] học hệ SDE thông qua hàm tổn thất hợp lí cực đại. Cụ thể, nếu sử dụng phương pháp giải số Euler-Maruyama cho SDE, mối liên hệ giữa trạng thái x_1 và x_0 cách nhau một khoảng thời gian $h > 0$ là: $x_1 = x_0 + hf(x_0) + g(x_0)dW_0$. Khi h đủ bé, ta có kết luận sau về xác suất của trạng thái x_1 : $\mathbb{P}(x_1 \mid x_0, h) \sim \mathcal{N}(x_0 + hf(x_0), hg(x_0)^2)$. Do đó, để tìm hàm hàm dịch chuyển và hàm khuếch tán, hàm hợp lí cực đại được cực đại hóa:

$\text{argmax}_\theta \mathbb{E}[\log \mathbb{P}_\theta(x_1 \mid x_0, h)]$ trong đó θ là trọng số của hàm dịch chuyển và hàm khuếch tán. Tuy nhiên, phương pháp này chỉ đúng khi h đủ bé. Về mặt lý thuyết, Wang và các cộng sự [25] chứng minh điều kiện cần và đủ để có thể học hàm dịch chuyển và hàm khuếch tán cho hệ SDE tuyến tính. Tuy vậy, lý thuyết tương tự cho trường hợp hệ SDE phi tuyến vẫn chưa có lời giải.

Trong nhiều bài toán thực tiễn, việc xác định chính xác mô hình SDE là không cần thiết. Thay vào đó, việc xác định hành vi và đặc điểm của mô hình SDE là phù hợp và đơn giản hơn. Ví dụ, trong dự báo giá cổ phiếu, người sử dụng muốn dự báo trong tương lai giá cổ phiếu sẽ có xác suất lớn ở vùng nào. Khi đó, thay vì xác định chính xác hàm dịch chuyển và hàm khuếch tán, hàm mật độ $p(t, x)$ sẽ được xác định. Hàm mật độ $p(t, x)$ đại diện cho phân phối xác suất của trạng thái tại thời điểm t. Giả sử có bộ dữ liệu $\mathcal{D}^{(sto)} = \left\{ \left\{ x_j^{(i)} \right\}_{j=0}^{m} \right\}_{i=0}^{n}$ trong đó $\left\{ x_j^{(i)} \right\}_{j=0}^{m}$ là một quỹ đạo của hệ SDE với các điểm cách đều nhau một khoảng thời gian Δ_t. Nếu giả thiết mô hình SDE có nhiễu cộng tính như trong công thức 2.2, dựa vào tính Markov của quá trình Weiner, $x(t)$ cũng là một quá trình Markov. Khi đó, ta có một phương pháp đơn giản để xấp xỉ hàm mật độ xác suất $p(t, x)$. Ta gọi đây là phương pháp $\mathcal{P}$. Đầu tiên, phương pháp $\mathcal{P}$ chia lưới không gian trạng thái $\mathcal{X}$ thành các hình hộp nhỏ $\mathcal{B}_i$ có kích thước $\underbrace{\Delta_x \times \Delta_x \times \ldots \times \Delta_x}_{\text{d lần}}$. Giả sử ta có N hình hộp $\{\mathcal{B}_i\}_{i=1}^{N}$. Coi mỗi hình hộp $\mathcal{B}_i$ là một trạng thái của chuỗi Markov thuần nhất theo thời gian (time-homogeneous Markov chains) $\mathcal{M}$ với dãy các biến ngẫu nhiên $X_1, X_2, ..., X_m$ trong đó:

$$\mathbb{P}(X_j = \mathcal{B}_i) = \text{ số điểm của tập hợp } \{x_j^{(i)}\}_{i=1}^{n} \text{ nằm trong hình hộp } \mathcal{B}_i.$$

Khi đó, ma trận xác suất chuyển $\mathbf{A} \in \mathbb{R}^{N \times N}$ của chỗi Markov $\mathcal{M}$ sẽ xấp xỉ hàm mật độ $p(t, x)$ của mô hình SDE. Để học ma trận $\mathbf{A}$, hàm tổn thất sau được sử dụng:

$$L_{Markov}(\mathbf{A}) = \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{A} s_{j-1} - s_j\|_2 \tag{2.3}$$

trong đó $s_j = \begin{bmatrix} \mathbb{P}(X_j = \mathcal{B}_1) \\ \mathbb{P}(X_j = \mathcal{B}_2) \\ \vdots \\ \mathbb{P}(X_j = \mathcal{B}_N) \end{bmatrix} \in \mathbb{R}^N$. Tuy nhiên, nhược điểm lớn nhất của phương pháp $\mathcal{P}$ là số tham số cần tối ưu rất lớn (số phần tử của ma trận $\mathbf{A}$). Để dự đoán chính xác hơn, phương pháp $\mathcal{P}$ cần phải chia nhỏ lưới thành các hình hộp. Giá trị Δ_x càng bé thì độ chính xác càng cao. Tuy vậy, khi Δ_x bé, số hình hộp $\mathcal{B}_i$ sẽ tăng. Khi đó, số trạng thái của quá trình Markov $\mathcal{M}$ cũng tăng và số phần tử của ma trận xác suất chuyển A tăng cao. Khi ấy việc tối ưu hàm tổn thất 2.3 sẽ thách thức và khó đạt được kết quả tốt. Trong một bài báo khác, Solin và các cộng sự [14] tiếp cận theo một hướng khác để xấp xỉ hàm mật độ $p(t, x)$. Tác giả xấp xỉ hàm mật độ $p(t, x)$ bằng mô hình Gaussian Mixture. Tuy nhiên, việc xác định số cụm phù hợp cho mô hình Gaussian Mixture là một điều không dễ. Số cụm lớn sẽ ảnh hưởng đến khả năng tính toán và huấn luyện của mô hình. Ngược lại, số cụm bé sẽ khiến mô hình không đủ phức tạp và kết quả huấn luyện sẽ không tốt.

2.3 Đóng góp của luận văn

Trong luận văn này, chúng tôi giải quyết bài toán học hệ động lực tất định có tập hút phức tạp ở đó trường vector có sự thay đổi giữa các vùng trong không gian trạng thái. Khi bộ dữ liệu huấn luyện được sinh theo phân phối đều trong không gian học, chúng tôi kết luận rằng các phương pháp học máy đều có thể học hệ động lực một cách hiệu quả. Trong trường hợp bộ dữ liệu huấn luyện được sinh bởi nhiều quỹ đạo, chúng tôi kết luận rằng các phương pháp mạng neuron đều gặp khó khăn do tính chất động lực học phức tạp của hệ. Thông thường các kết quả hiện nay đều áp dụng dưới giả thiết trường vector của hệ động lực liên tục là liên tục Lipschitz toàn cục, trong khi đó các hệ ví dụ xét trong luận án như FHN chỉ liên tục Lipschitz địa phương và tăng trưởng tuyến

tính một phía. Do đó, luận văn đề xuất một phương pháp có thể học hiệu quả bộ dữ liệu quỹ đạo thông qua học hệ động lực liên hợp tương ứng. Đóng góp của luận văn được cho như sau:

- Thử nghiệm phương pháp mạng neuron MLP và NeuralODE lên bộ dữ liệu đều trong không gian học.
- Thử nghiệm phương pháp mạng neuron MLP và NeuralODE lên bộ dữ liệu gồm nhiều quỹ đạo.
- Phân tích sai số của phương pháp MLP và NeuralODE khi áp dụng học bộ dữ liệu sinh bởi nhiều quỹ đạo.
- Đề xuất một phương pháp giải quyết bài toán học hệ động lực với bộ dữ liệu được sinh bởi nhiều quỹ đạo.
- Thực nghiệm chứng minh kết quả phương pháp đề xuất.

Chương 3

Áp dụng mạng neuron giải hệ động lực tất định

Chương này sẽ áp dụng mạng neuron nhân tạo cho việc học hệ động lực có tập hút hỗn loạn. Do tính chất động lực học phức tạp, nếu bộ dữ liệu huấn luyện được chọn từ nhiều quỹ đạo của hệ, phân bố của của bộ dữ liệu huấn luyện sẽ tập trung không đồng đều trên không gian học. Khi đó, cả hai phương pháp MLP và NeuralODE đều không học hiệu quả. Do đó, dựa trên ý tưởng của hệ động lực liên hợp, chúng tôi đề xuất biến đổi bộ dữ liệu huấn luyện về dạng dễ học hơn trước khi áp dụng phương pháp MLP hay NeuralODE lên bộ dữ liệu sau khi đã được biến đổi. Trong mục 2.1, ta sẽ đánh giá sự hiệu quả của phương pháp MLP và phương pháp NeuralODE khi học hệ động lực có tập hút hỗn loạn với bộ dữ liệu đều trên không gian học. Mục 2.2 sẽ áp dụng phương pháp MLP và phương pháp NeuralODE học hệ động lực có tập hút hỗn loạn với bộ dữ liệu được lấy từ các quỹ đạo của hệ động lực đó. Mục 2.3 sẽ đề xuất phương pháp học cải thiện trong trường hợp bộ dữ liệu bao gồm các quỹ đạo của hệ động lực. Phương pháp đề xuất có thể áp dụng cho mọi hệ động lực có tập hút hỗn loạn. Tuy nhiên trong khuôn khổ luận văn này, chúng tôi chỉ xây dựng phương pháp cho hệ ODE autonomous trên không gian pha 2 chiều, và trình bày áp dụng cho hệ FHN.

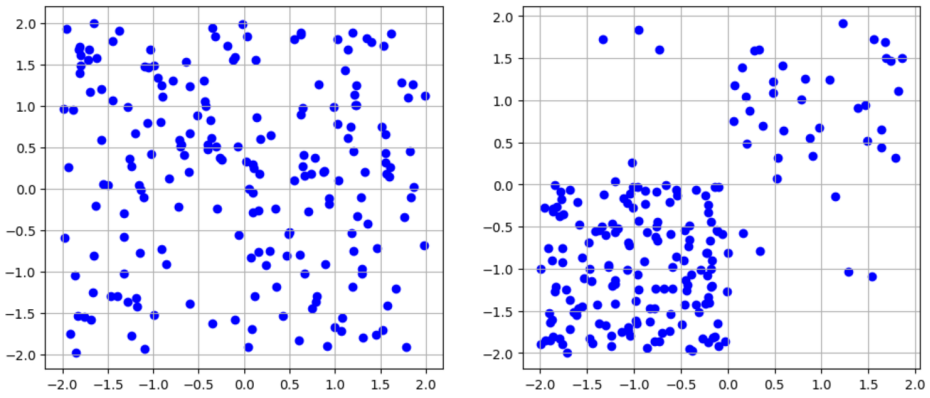
3.1 Học hệ động lực với bộ dữ liệu đều

3.1.1. Phương pháp sinh dữ liệu

Các phương pháp học máy là các phương pháp học tập trung vào dữ liệu (data-centric). Dựa trên dữ liệu huấn luyện, các phương pháp mạng neuron sẽ học các tính chất, đặc điểm của đối tượng cần được học trên không gian $\mathcal{X}$ mà bộ dữ liệu huấn luyện hỗ trợ tốt nhất.

Phân bố của dữ liệu huấn luyện đóng vai trò quan trọng đối với sự hiệu quả của mạng neuron [26]. Để mạng neuron có thể học chính xác hệ động lực trong không gian $\mathcal{X}$, bộ dữ liệu $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ cần phủ tốt không gian $\mathcal{X}$ hay $\{x_i\}_{i=1}^N$ cần tuân theo phân phối đều trong khoảng $\mathcal{X}$. Hình 3.1 sẽ giải thích rõ hơn về hiện tượng này. Ta xét $\mathcal{X} = [-2, 2]^2$. Hình 3.1 bên trái biểu diễn 200 điểm dữ liệu phân bố đều trong $\mathcal{X}$. Khi đó, có thể thấy $\mathcal{X}$ được phủ đều bởi các điểm dữ liệu và mạng neuron sẽ học được nhiều thông tin về hệ động lực hơn trong không gian $\mathcal{X}$. Ngược lại, hình 3.1 bên phải là một ví dụ khi phân bố của dữ liệu không phủ đều không gian $\mathcal{X}$. Cụ thể, dữ liệu tập trung chủ yếu ở vùng $[-2, 0]^2$ và thưa ở những vùng còn lại trong không gian $\mathcal{X}$. Khi đó, mạng neuron sẽ học rất tốt ở trong vùng $[-2, 0]^2$ do thông tin có được từ dữ liệu là đủ nhiều. Ngược lại, mạng neuron sẽ học kém hiệu quả trong những vùng còn lại do dữ liệu thưa, không mang đủ thông tin để mạng neuron có thể học hiệu quả. Vấn đề huấn luyện mô hình mạng neuron học hiệu quả trên những vùng không gian thưa là một chủ đề được nghiên cứu rộng rãi [27]. Trong chương này, ta sẽ không đi sâu vào vấn đề cải thiện độ hiệu quả của các phương pháp mạng neuron với trên vùng dữ liệu thưa mà sẽ tập trung phân tích sự hiệu quả của các phương pháp mạng neuron với bộ dữ liệu có phân phối đều. Thuật toán sinh dữ liệu đều được cho trong thuật toán 2.

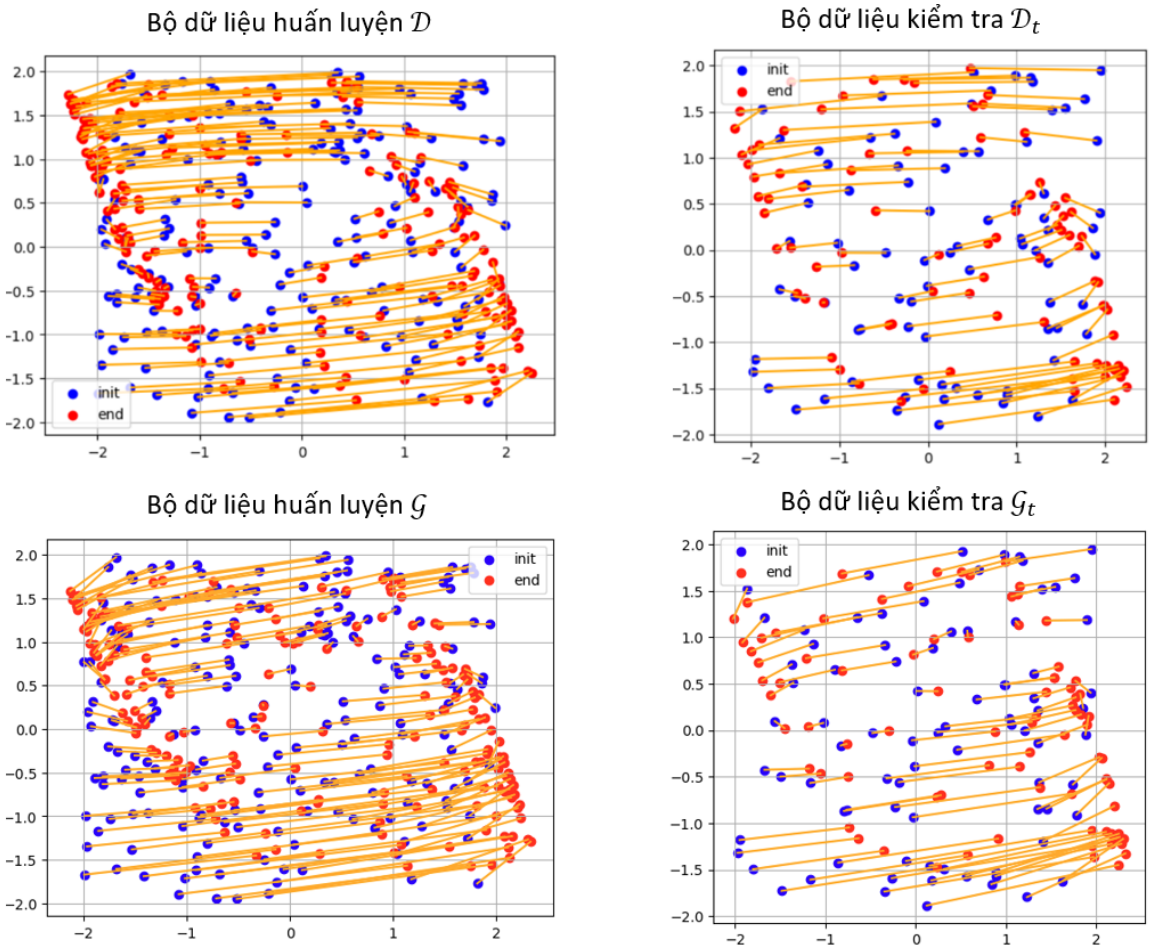
Trong mục này, ta sẽ áp dụng phương pháp mạng neuron MLP và phương pháp mạng NeuralODE học hệ động lực FHN có tập hút phức tạp. Bộ dữ liệu huấn luyện được sinh bởi thuật toán 2 với hệ động lực FHN và các tham số



Hình 3.1: Ví dụ về phân bố của dữ liệu. (**Bên trái**) 200 điểm dữ liệu phân bố đều trong khoảng $[-2, 2]^2$; (**Bên phải**) 200 điểm dữ liệu phân bố không đều trong khoảng $[-2, 2]^2$.

$\mathcal{X} = [-2, 2]^2; t_0 = 0; t_1 = 1; N = 200$. Ta sẽ kiểm thử phương pháp mạng neuron với hai trường hợp của hệ FHN: trường hợp có một điểm cân bằng và trường hợp có ba điểm cân bằng. Đối với trường hợp có một điểm cân bằng, ta xác định các tham số $a = 0.7; b = 0.8; \tau = 12.5; R = 0.1; I_{ext} = 0.5$. Khi đó, bộ dữ liệu huấn luyện thu được là $\mathcal{D} = \{x_{i,\mathcal{D}}; y_{i,\mathcal{D}}\}_{i=1}^{200}$. Trong trường hợp hệ FHN có ba điểm cân bằng, ta xác định các tham số $a = 0.7; b = 2.0; \tau = 12.5; R = 0.1; I_{ext} = 5.393$. Khi đó, bộ dữ liệu huấn luyện thu được là $\mathcal{G} = \{x_{i,\mathcal{G}}; y_{i,\mathcal{G}}\}_{i=1}^{200}$.

Thông thường để kiểm tra độ hiệu quả, mô hình mạng neuron với trọng số tối ưu sẽ được kiểm thử trên một tập dữ liệu kiểm tra (test set). Tập dữ liệu kiểm tra này sẽ không chứa các điểm nằm trong tập dữ liệu huấn luyện. Nếu mạng neuron được huấn luyện cho kết quả sai số thấp trên bộ dữ liệu kiểm tra, mô hình mạng neuron có tính tổng quát hóa (generalization) tốt. Để tạo bộ dữ liệu kiểm tra $\mathcal{D}_t$ (và bộ dữ liệu kiểm tra $\mathcal{G}_t$), phương pháp sinh dữ liệu trong thuật toán 2 được sử dụng với hệ động lực FHN và các tham số $\mathcal{X} = [-2, 2]^2; t_0 = 0.0; t_1 = 1.0$ và $N = 85$. Bộ dữ liệu huấn luyện $\mathcal{D}; \mathcal{G}$ và bộ dữ liệu kiểm tra $\mathcal{D}_t = \{\widetilde{x_{i,\mathcal{D}}}; \widetilde{y_{i,\mathcal{D}}}\}_{i=1}^{85}; \mathcal{G}_t = \{\widetilde{x_{i,\mathcal{G}}}; \widetilde{y_{i,G}}\}_{i=1}^{85}$ có thể thấy trong hình 3.2.



Hình 3.2: (**Trái trên**) Bộ dữ liệu huấn luyện $\mathcal{D}$ của hệ động lực FHN với một điểm cân bằng; (**Phải trên**) Bộ dữ liệu kiểm tra $\mathcal{D}_t$ của hệ động lực FHN với một điểm cân bằng; (**Trái dưới**) Bộ dữ liệu huấn luyện $\mathcal{G}$ của hệ động lực FHN với ba điểm cân bằng; (**Phải dưới**) Bộ dữ liệu kiểm tra $\mathcal{G}_t$ của hệ động lực FHN với ba điểm cân bằng. Màu xanh là các điểm $\{x_{i,\mathcal{M}}; \widetilde{x_{i,\mathcal{M}}}\}$, màu đỏ là các điểm $\{y_{i,\mathcal{M}}; \widetilde{y_{i,\mathcal{M}}}\}(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$, đoạn thẳng màu cam kết nối các điểm $x_{i,\mathcal{M}}, y_{i,\mathcal{M}}$ và $\widetilde{x_{i,\mathcal{M}}}, \widetilde{y_{i,\mathcal{M}}}(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$ tương ứng.

Algorithm 2 Thuật toán sinh dữ liệu đều

1: **Tham số:** Độ lớn của dữ liệu N, trường vector f của hệ động lực, không gian $\mathcal{X} \subset \mathbb{R}^2$, t_0, t_1.
2: **for** i=1,2,...,N **do**
3: $\quad x_i \sim \mathcal{U}(\mathcal{X})$
4: $\quad y_i = \text{ODESolve}\,(x_i, f, t_0, t_1)$
5: **end for**
6: **Trả về:** Bộ dữ liệu $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$.

3.1.2. Áp dụng phương pháp MLP học hệ động lực với bộ dữ liệu đều

Để học hệ động lực với bộ dữ liệu huấn luyện được cho trong hình 3.2, kiến trúc mạng neuron MLP được xây dựng với các tham số trong bảng 3.1.

Số lớp đầu vào	2
Số lớp ẩn	1
Số nốt lớp ẩn	500
Số nốt lớp đầu ra	2
Hàm kích hoạt	ELU

Bảng 3.1: Kiến trúc mạng neuron MLP

Kí hiệu đầu vào là $\mathbf{x} \in \mathbb{R}^2$, đầu ra $\widehat{\mathbf{y}} \in \mathbb{R}^2$ và mạng MLP $f_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ với trọng số cần học là θ. Khi đó, ta có mối liên hệ sau:

$$\widehat{\mathbf{y}} = f_\theta(\mathbf{x}) \tag{3.1}$$

Với bộ dữ liệu huấn luyện $\mathcal{D}$ (tương tự với bộ dữ liệu huấn luyện $\mathcal{G}$), hàm tổn thất MSE được sử dụng cho việc huấn luyện mạng MLP $f_{\theta,\mathcal{D}}$:

$$\theta^*_{\mathcal{D}} \in \operatorname{argmax}_\theta L(\theta) \tag{P_1}$$

trong đó

$$L(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_i;y_i)\in\mathcal{D}} ||f_\theta(x_i) - y_i||_2 \tag{3.2}$$

Để giải bài toán (P_1), ta sử dụng phương pháp huấn luyện Adam, một biến thể của phương pháp hướng giảm để huấn luyện các mạng neuron. Phương pháp Adam được sử dụng phổ biến hơn so với phương pháp hướng giảm trong việc huấn luyện các mạng neuron. Bài toán (P_1) được giải với 1500 vòng lặp của thuật toán Adam. Trọng số tối ưu của mạng neuron MLP sẽ được lưu tại vòng lặp đạt giá trị hàm tổn thất nhỏ nhất (chọn $\theta^*_{\mathcal{D}}$ tại vòng lặp có giá trị hàm tổn thất nhỏ nhất).

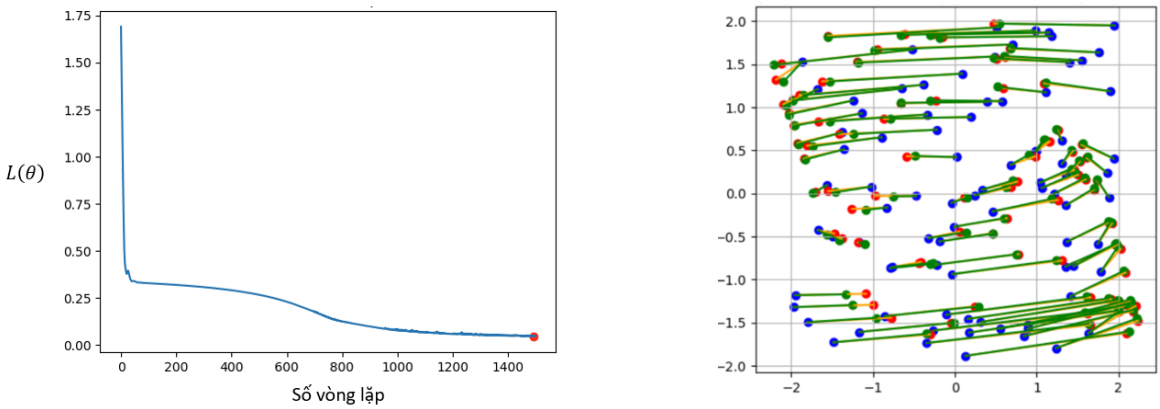
Kết quả huấn luyện với bộ dữ liệu huấn luyện $\mathcal{D}$ được cho trong hình 3.3 trái. Nhìn vào hình 3.3 trái, có thể thấy hàm tổn thất giảm dần theo từng vòng lặp và

đạt giá trị nhỏ nhất tại vòng lặp 1500. Kết quả của mạng neuron MLP tối ưu với bộ dữ liệu kiểm tra $\mathcal{D}_t$ được cho trong hình 3.3 phải. Nhìn vào hình 3.3 phải, có thể thấy mạng neuron MLP $f_{\theta^*_{\mathcal{D}}}$ dự đoán khá chính xác với bộ dữ liệu kiểm tra $\mathcal{D}_t$. Để đo lường độ hiệu quả, ta đi tính hàm tổn thất:

$$L(\theta^*_{\mathcal{D}}) = \frac{1}{\mid \mathcal{D}_t \mid} \sum_{(\widetilde{x}_i;\widetilde{y}_i)\in\mathcal{D}_t} \left|\left| f_{\theta^*_{\mathcal{D}}}(\widetilde{x}_i) - \widetilde{y}_i \right|\right|_2 \approx 0.0553 \tag{3.3}$$

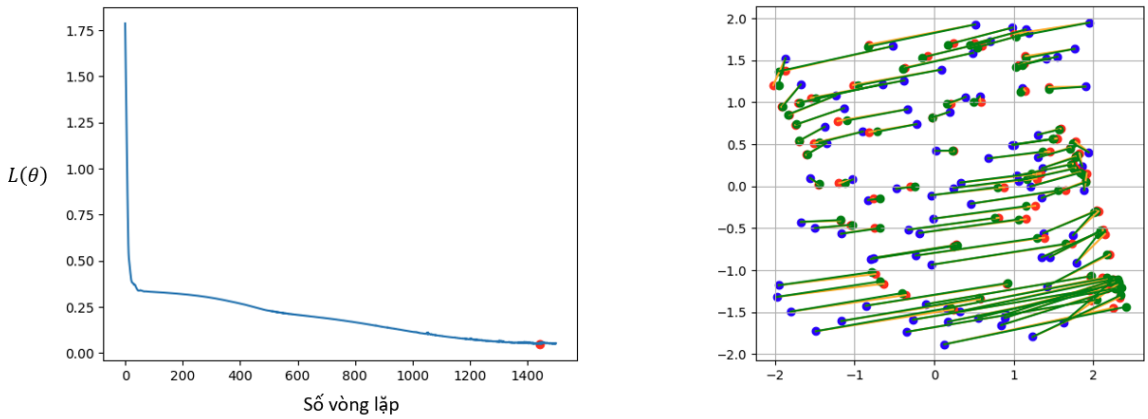
Tương tự, kết quả huấn luyện với bộ dữ liệu huấn luyện $\mathcal{G}$ được cho trong hình 3.4. Nhìn vào hình 3.4 phải, mạng neuron MLP $f_{\theta^*_{\mathcal{G}}}$ tối ưu dự đoán khá chính xác với bộ dữ liệu kiểm tra $\mathcal{G}_t$. Để đo lường độ hiệu quả, ta đi tính hàm tổn thất:

$$L(\theta^*_{\mathcal{G}}) = \frac{1}{\mid \mathcal{G}_t \mid} \sum_{(\widetilde{x}_i;\widetilde{y}_i)\in\mathcal{G}_t} \left|\left| f_{\theta^*_{\mathcal{G}}}(\widetilde{x}_i) - \widetilde{y}_i \right|\right|_2 \approx 0.0505 \tag{3.4}$$



Hình 3.3: (**Trái**) Quá trình huấn luyện mạng neuron MLP với bộ dữ liệu $\mathcal{D}$. Trục hoành biểu thị số vòng lặp của thuật toán Adam, trục tung biểu thị giá trị của hàm tổn thất $L(\theta)$, điểm màu đỏ biểu thị giá trị nhỏ nhất của hàm tổn thất; (**Phải**) Kết quả mạng neuron MLP trên bộ dữ liệu kiểm tra $\mathcal{D}_t$. Màu xanh da trời là các điểm $\widetilde{x}_i$, màu đỏ là các điểm $\widetilde{y}_i$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x}_i$ và $\widetilde{y}_i$ tương ứng. Màu xanh lá cây là các điểm dự đoán của mạng neuron MLP, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của mạng neuron MLP $\widetilde{x}_i$.

Tuy vậy, phương pháp mạng neuron MLP chỉ có thể dự đoán lời giải của bài toán IVP sau một khoảng thời gian $\Delta_t = t_1 - t_0$ cố định. Phương pháp sẽ tỏ ra kém hiệu khi dự đoán lời giải sau khoảng thời gian $\Delta'_t \neq \Delta_t$. Do đó, phương



Hình 3.4: (**Trái**) Quá trình huấn luyện mạng neuron MLP với bộ dữ liệu $\mathcal{G}$. Trục hoành biểu thị số vòng lặp của thuật toán Adam, trục tung biểu thị giá trị của hàm tổn thất $L(\theta)$, điểm màu đỏ biểu thị giá trị nhỏ nhất của hàm tổn thất; (**Phải**) Kết quả mạng neuron MLP trên bộ dữ liệu kiểm tra $\mathcal{G}_t$. Màu xanh da trời là các điểm $\widetilde{x}_i$, màu đỏ là các điểm $\widetilde{y}_i$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x}_i$ và $\widetilde{y}_i$ tương ứng. Màu xanh lá cây là các điểm dự đoán của mạng neuron MLP, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của mạng neuron MLP $\widetilde{x}_i$.

pháp neuron MLP không học được nhiều các tính chất động lực của hệ. Để có thể học được các tính chất động lực hiệu quả hơn, ta sẽ cần phải đi học trường vector $f(x)$ của hệ động lực. Phương pháp NeuralODE cung cấp cơ chế để có thể làm điều đó.

3.1.3. Áp dụng phương pháp NeuralODE học hệ động lực với bộ dữ liệu đều

Điểm khác biệt giữa phương pháp NeuralODE và phương pháp MLP khi học hệ động lực là phương pháp NeuralODE sẽ học trực tiếp trường vector. Cụ thể, NeuralODE sẽ mô hình hóa trường vector bằng một mạng neuron $f_\theta : \frac{dx}{dt} = f_\theta(x)$. Sau đó, phương pháp sẽ sử dụng thuật toán "Adjoint" như trong thuật toán 1.6 để huấn luyện.

Ta sẽ áp dụng phương pháp NeuralODE cho việc học hệ động lực FHN với bộ dữ liệu huấn luyện $\mathcal{D}$ và $\mathcal{G}$. Kiến trúc của trường vector $f_{\theta,\mathcal{M}}(\mathcal{M} \in$

$\{\mathcal{D}, \mathcal{G}\}$) được xây dựng tương tự như trong bảng 3.1. Để huấn luyện phương pháp NeuralODE, hàm tổn thất MSE được sử dụng và đi giải bài toán sau tối ưu:

$$\theta^*_{\mathcal{M}} \in \text{argmax}_\theta \, L_{ODE}(\theta) \tag{P_2}$$

trong đó

$$L_{ODE}(\theta) = \frac{1}{|\mathcal{M}|} \sum_{(x_i;y_i)\in\mathcal{M}} \|\text{ODESolve}\,(x_i; f_\theta; t_0; t_1) - y_i\|_2\,; \tag{3.5}$$

và $\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\}$.

Dựa vào công thức tính toán hàm tổn thất như trong công thức (3.5), ta cần xác định t_0 và t_1, trong đó t_0 là thời điểm ban đầu và t_1 là thời điểm cuối. Chen và các cộng sự [10] đề xuất một phương pháp có thể học đồng thời t_0 và t_1. Tuy nhiên, để thuận tiện trong việc tính toán, ta sẽ xét cố định $t_0 = 0$ và $t_1 = 1$.

Để giải bài toán (P_2), phương pháp "adjoint" trong thuật toán 1.6 được sử dụng để tính đạo hàm của hàm tổn thất với trọng số θ. Sau khi tính được đạo hàm $\frac{\partial L_{ODE}}{\partial \theta}$, phương pháp cập nhật trọng số Adam được sử dụng với 1500 vòng lặp. Trọng số của trường vector f_θ được lưu tại vòng lặp đạt giá trị hàm tổn thất nhỏ nhất.

Kết quả huấn luyện với bộ dữ liệu $\mathcal{D}$ được cho trong hình 3.5 trái. Kết quả của phương pháp NeuralODE với bộ trọng số tối ưu áp dụng lên bộ dữ liệu kiểm tra $\mathcal{D}_t$ được cho trong hình 3.5 phải. Độ hiệu quả của phương pháp NeuralODE trên bộ dữ liệu kiểm tra $\mathcal{D}_t$ được tính toán với hàm tổn thất sau:

$$L_{ODE}(\theta^*_{\mathcal{D}}) = \frac{1}{|\ \mathcal{D}_t\ |} \sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{D}_t} \|\text{ODESolve}\,(\widetilde{x}_i; f_{\theta^*}; t_0 = 0; t_1 = 1) - \widetilde{y}_i\|_2 \approx 0.0533$$

Kết quả huấn luyện với bộ dữ liệu $\mathcal{G}$ được cho trong hình 3.6. Kết quả của phương pháp NeuralODE với bộ trọng số tối ưu áp dụng lên bộ dữ liệu kiểm tra $\mathcal{G}_t$ được cho trong hình 3.6 phải. Độ hiệu quả của phương pháp NeuralODE trên bộ dữ liệu kiểm tra $\mathcal{G}_t$ được tính toán với hàm tổn thất sau:

$$L_{ODE}(\theta^*_{\mathcal{G}}) = \frac{1}{|\ \mathcal{G}_t\ |} \sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{G}_t} \|\text{ODESolve}\,(\widetilde{x}_i; f_{\theta^*}; t_0 = 0; t_1 = 1) - \widetilde{y}_i\|_2 \approx 0.0556$$

Kết quả so sánh phương pháp mạng neuron MLP và phương pháp NeuralODE được cho trong bảng 3.2. Nhìn vào bảng 3.2 , phương pháp NeuralODE cho kết quả tốt hơn một chút trên bộ dữ liệu kiểm tra $\mathcal{D}_t$ so với phương pháp mạng neuron MLP. Điều này có thể giải thích rằng khi sử dụng phương pháp NeuralODE, ta đã mô hình hóa hệ động lực cho quá trình học, ví dụ sử dụng bộ giải phương trình vi phần cho quá trình tính toán.

	MSE trên bộ $\mathcal{D}$	MSE trên bộ $\mathcal{D}_t$	MSE trên bộ $\mathcal{G}$	MSE trên bộ $\mathcal{G}_t$
MLP	0.0544	0.0553	0.0430	0.0505
NeuralODE	0.0254	0.0533	0.0248	0.0556

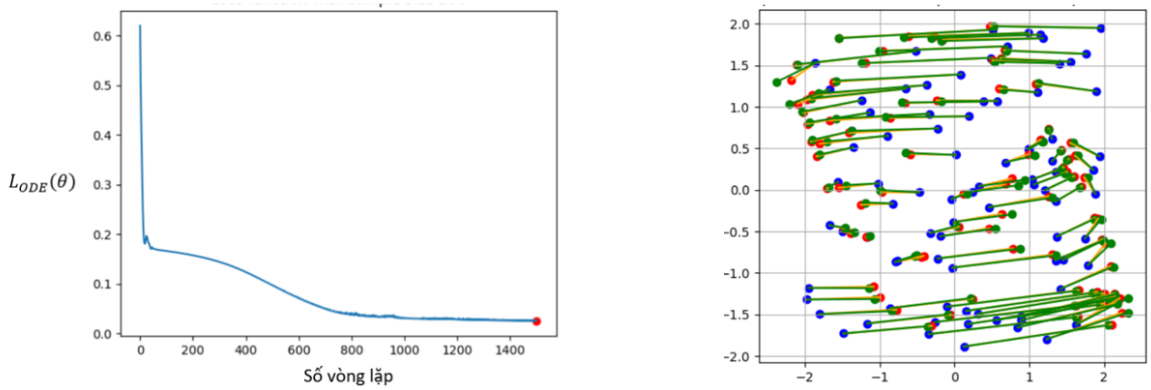
Bảng 3.2: Kết quả so sánh sai số của phương pháp MLP và phương pháp NeuralODE.

Sau khi đã học được trường vector, phương pháp NeuralODE có thể dự đoán tại bất kì thời điểm t_1 nào. Đây là ưu điểm của phương pháp NeuralODE so với phương pháp MLP. Ta sẽ kiểm thử phương pháp NeuralODE với bộ dữ liệu $\mathcal{D}_t^{'}, \mathcal{G}_t^{'}$ với $t_0 = 0; t_1 = 1.5$ và $N = 85$. Bộ dữ liệu $\mathcal{D}_t^{'}$ tương ứng với hệ FHN với một điểm cân bằng và bộ $\mathcal{G}_t^{'}$ tương ứng với hệ FHN với ba điểm cân bằng. Kết quả dự đoán của phương pháp NeuralODE được cho trong hình 3.7. Sai số của phương pháp trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{'}, \mathcal{G}_t^{'}$ là:

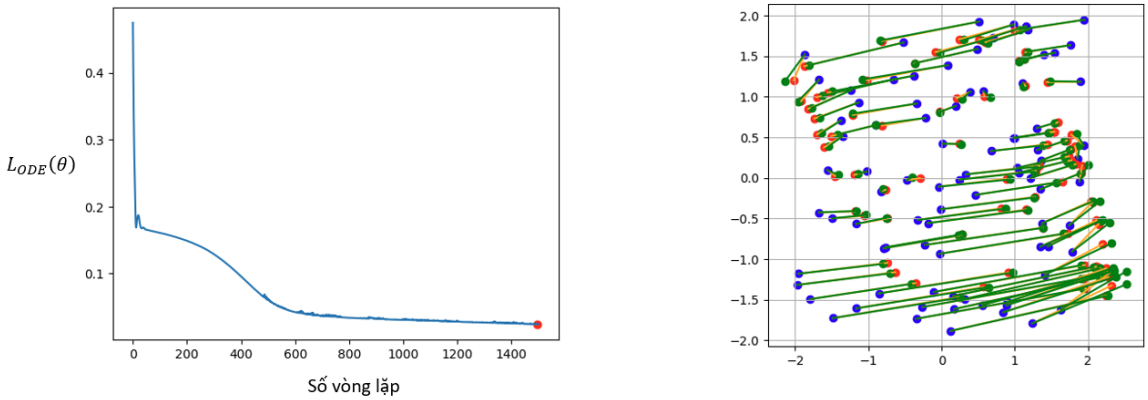
$$L_{ODE}(\theta^*_{\mathcal{D}}) = \frac{1}{\mid \mathcal{D}_t^{'} \mid} \sum_{(\widetilde{x}_i, \widetilde{y}_i) \in \mathcal{D}_t^{'}} ||\text{ODESolve}\left(\widetilde{x}_i; f_{\theta_{\mathcal{D}^*}}; t_0 = 0; t_1 = 1.5\right) - \widetilde{y}_i||_2 \approx 0.0681$$

$$L_{ODE}(\theta^*_{\mathcal{G}}) = \frac{1}{\mid \mathcal{G}_t^{'} \mid} \sum_{(\widetilde{x}_i, \widetilde{y}_i) \in \mathcal{G}_t^{'}} \left|\left|\text{ODESolve}\left(\widetilde{x}_i; f_{\theta^*_{\mathcal{G}}}; t_0 = 0; t_1 = 1.5\right) - \widetilde{y}_i\right|\right|_2 \approx 0.0721$$

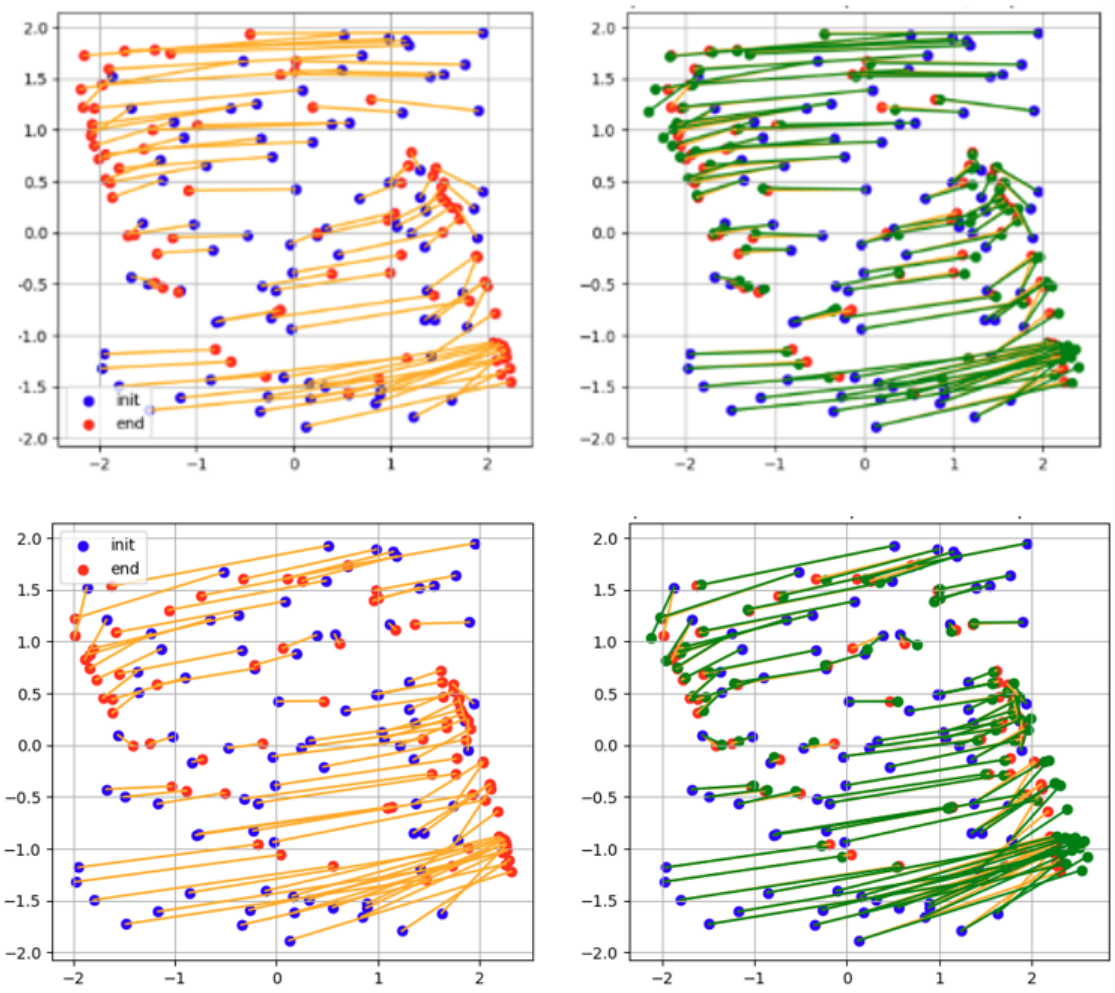
Có thể thấy rằng, phương pháp NeuralODE vẫn có thể dự đoán tốt với $t_1 = 1.5$ cho dù bộ dữ liệu huấn luyện được sinh ra với $t_1 = 1$.



Hình 3.5: (**Trái**) Quá trình huấn luyện NeuralODE với bộ dữ liệu đều $\mathcal{D}$. Trục hoành biểu thị số vòng lặp của thuật toán Adam, trục tung biểu thị giá trị của hàm tổn thất $L_{ODE}(\theta)$, điểm màu đỏ biểu thị giá trị nhỏ nhất của hàm tổn thất; (**Phải**) Kết quả phương pháp NeuralODE trên bộ dữ liệu kiểm tra $\mathcal{D}_t$. Màu xanh da trời là các điểm $\widetilde{x}_i$, màu đỏ là các điểm $\widetilde{y}_i$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x}_i$ và $\widetilde{y}_i$ tương ứng. Màu xanh lá cây là các điểm dự đoán của mạng neuron MLP, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của phương pháp NeuralODE $\widetilde{x}_i$.



Hình 3.6: (**Trái**) Quá trình huấn luyện NeuralODE với bộ dữ liệu đều $\mathcal{G}$. Trục hoành biểu thị số vòng lặp của thuật toán Adam, trục tung biểu thị giá trị của hàm tổn thất $L_{ODE}(\theta)$, điểm màu đỏ biểu thị giá trị nhỏ nhất của hàm tổn thất; (**Phải**) Kết quả phương pháp NeuralODE trên bộ dữ liệu kiểm tra $\mathcal{G}_t$. Màu xanh da trời là các điểm $\widetilde{x}_i$, màu đỏ là các điểm $\widetilde{y}_i$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x}_i$ và $\widetilde{y}_i$ tương ứng. Màu xanh lá cây là các điểm dự đoán của mạng neuron MLP, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của phương pháp NeuralODE $\widetilde{x}_i$.



Hình 3.7: Kết quả phương pháp NeuralODE trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{'}$ và $\mathcal{G}_t^{'}$ với $t_1 = 1.5$. (**Trái trên**) Bộ dữ liệu kiểm tra $\mathcal{D}_t^{'}$; (**Phải trên**) Kết quả dự đoán của phương pháp NeuralODE với $t_1 = 1.5$ trên bộ $\mathcal{D}_t^{'}$; (**Trái dưới**) Bộ dữ liệu kiểm tra $\mathcal{G}_t^{'}$; (**Phải dưới**) Kết quả dự đoán của phương pháp NeuralODE với $t_1 = 1.5$ trên bộ $\mathcal{G}_t^{'}$

3.2 Học hệ động lực với bộ dữ liệu gồm nhiều quỹ đạo

3.2.1. Phương pháp sinh dữ liệu

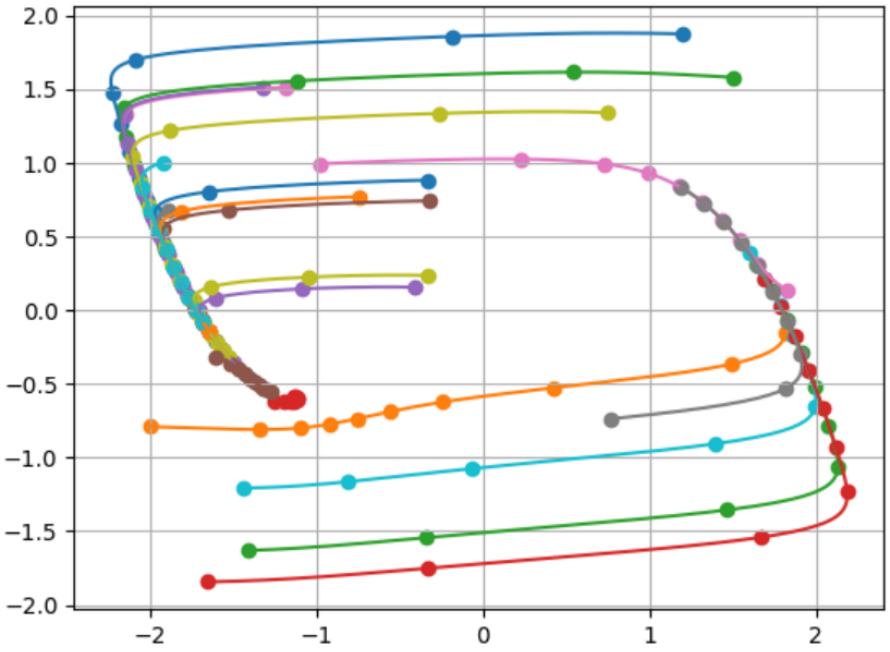
Trọng mục 3.1, có thể thấy rằng phương pháp MLP lẫn phương pháp NeuralODE đều có thể học hiệu quả hệ động lực khi bộ dữ liệu huấn luyện đều. Tuy nhiên, hạn chế của việc sinh dữ liệu đều là các điểm dữ liệu được sinh ra

bị giới hạn trong không gian học $\mathcal{X}$. Điều này khiến các phương pháp mạng neuron không dự đoán chính xác bên ngoài khoảng không gian $\mathcal{X}$. Do đó, việc xác định không gian $\mathcal{X}$ cho việc sinh dữ liệu đều đóng vai trò rất quan trọng đối với tính tổng quát của các phương pháp mạng neuron.

Đối với những hệ động lực có tập hút phức tạp, việc xác định không gian $\mathcal{X}$ là một vấn đề khó. Hình 3.8 cho thấy hình dạng của 20 quỹ đạo hệ động lực FHN có một điểm cân bằng với điểm ban đầu ngẫu nhiên, mỗi quỹ đạo có 10 điểm dữ liệu, các điểm dữ liệu cách đều nhau một khoảng thời gian là $\Delta_t = 1$. Kí hiệu quỹ đạo thứ i là $\{x_j^{(i)}\}_{j=0}^{9}$. Nhìn vào hình 3.8, cho dù giữa hai điểm bất kì đều cách nhau một khoảng thời gian $\Delta_t = 1$, dữ liệu có xu hướng bị hút vào những vùng nhất định. Cụ thể, dữ liệu tập trung nhiều ở trong vùng đa tạp chậm $[-2, -1] \times [-0.6, 0.5]$. Trong khi đó, dữ liệu bị thưa ở vùng đa tạp nhanh $[-2, 2] \times [-2, -1]$. Dựa trên cấu trúc phức tạp này, nếu không gian $\mathcal{X}$ chỉ chứa vùng đa tạp chậm, mô hình mạng neuron sẽ không dự đoán chính xác trên vùng đa tạp nhanh và ngược lại.

Do đó, để hạn chế bộ dữ liệu huấn luyện phụ thuộc vào không gian $\mathcal{X}$, dữ liệu sẽ được sinh bởi các quỹ đạo. Việc sinh dữ liệu dựa trên quỹ đạo không cần xác định không gian $\mathcal{X}$ cho toàn bộ dữ liệu huấn luyện mà chỉ cần xác định không gian sinh ngẫu nhiên cho các điểm ban đầu của các quỹ đạo $\{x_0^{(i)}\}_{i=1,2,\ldots}$. Ta kí hiệu không gian này là $\mathcal{X}_0$. Khi đó, độ phủ của dữ liệu sẽ được quyết định dựa trên chính tính chất động lực học của hệ. Thuật toán tạo bộ dữ liệu quỹ đạo được cho trong thuật toán 3.

Tương tự như mục 3.1.1., ta sinh bộ dữ liệu $\mathcal{D}^{(traj)}$ với hệ động lực FHN có một điểm cân bằng và bộ dữ liệu $\mathcal{G}^{(traj)}$ với hệ động lực FHN có ba điểm cân bằng với các tham số $n = 10, m = 21, t_0 = 0, t_1 = 1, \mathcal{X}_0 = [-2, 2]^2$. Chú ý rằng việc chọn $m = 21, n = 10$ để đảm bảo số điểm dữ liệu trong $\mathcal{D}^{(traj)}, \mathcal{G}^{(traj)}$ là 200, bằng số điểm dữ liệu huấn luyện đều $\mathcal{D}$ và $\mathcal{G}$ trong mục 3.1.2.. Dữ liệu huấn luyện quỹ đạo $\mathcal{D}^{(traj)} = \left\{ \left\{ x_{j,\mathcal{D}}^{(i)} \right\}_{j=0}^{20} \right\}_{i=1}^{10}$ và $\mathcal{G}^{(traj)} =$



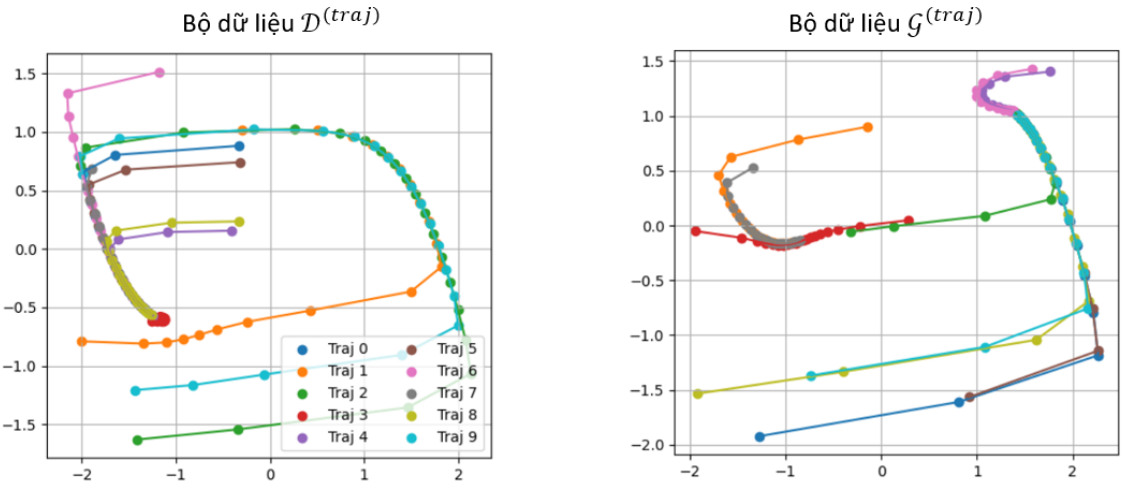
Hình 3.8: Hệ FHN với 20 quỹ đạo ngẫu nhiên. Mỗi quỹ đạo có một màu sắc khác nhau. Các điểm trên quỹ đạo được nối với nhau bởi đường cong cùng màu.

$\left\{\left\{x_{j,\mathcal{G}}^{(i)}\right\}_{j=0}^{20}\right\}_{i=1}^{10}$ được cho trong hình 3.9. Ngoài ra, để kiểm tra độ hiệu quả, các mô hình sẽ được kiểm thử trên bộ dữ liệu $\mathcal{D}_t$ và $\mathcal{G}_t$ trong mục 3.1.2..

Algorithm 3 Thuật toán sinh dữ liệu quỹ đạo

Tham số: trường vector hệ động lực f, số quỹ đạo n, số điểm trên quỹ đạo m, không gian $\mathcal{X}_0 \subset \mathbb{R}^2$, t_0, t_1.
for i=1,2,...,n **do**
    $x_0^{(i)} \sim \mathcal{U}(\mathcal{X}_0)$
    for j = 1, 2 ,..., m-1 **do**
        $x_j^{(i)} = \text{ODESolve}\left(x_{j-1}^{(i)}; FHN; t_0; t_1\right)$
    end for
end for
Trả về: $\{x_j^{(i)}\}_{j=0}^{m-1} \forall i = 1, 2, .., n.$

3.2.2. Áp dụng phương pháp MLP học hệ động lực với bộ dữ liệu quỹ đạo



Hình 3.9: (**Trái**) Bộ dữ liệu huấn luyện quỹ đạo $\mathcal{D}^{(traj)}$; (**Phải**) Bộ dữ liệu huấn luyện quỹ đạo $\mathcal{G}^{(traj)}$. Mỗi quỹ đạo và các điểm trên quỹ đạo tương ứng với một màu khác nhau.

Kiến trúc mạng neuron MLP được xây dựng tương tự như trong bảng 3.1 với kí hiệu là $f^{(traj)}_{\theta,\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}^2$ với $\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\}$ tương ứng với bộ dữ liệu huấn luyện $\mathcal{D}^{(traj)}$ và $\mathcal{G}^{(traj)}$. Với đầu vào là $x^{(i)}_{j,\mathcal{M}}$ ($j, i, \mathcal{M} \in \{\mathcal{D}, \mathcal{G}\}$ bất kì), mạng neuron MLP $f^{(traj)}_{\theta,\mathcal{M}}$ sẽ dự báo điểm tiếp theo trên quỹ đạo đó $x^{(i)}_{j+1,\mathcal{M}}$:

$$\widehat{x^{(i)}_{j+1,\mathcal{M}}} = f^{(traj)}_{\theta,\mathcal{M}}\left(x^{(i)}_{j,\mathcal{M}}\right) \tag{3.6}$$

hàm tổn thất có thể được xây dựng theo công thức sau:

$$L^{(traj)}_{1,\mathcal{M}}(\theta) = \frac{1}{n(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m-1} \left\| f^{(traj)}_{\theta,\mathcal{M}}\left(x^{(i)}_{j-1,\mathcal{M}}\right) - x^{(i)}_{j,\mathcal{M}} \right\|_2 \tag{3.7}$$

Chú ý rằng, do các điểm trên quỹ đạo cách đều nhau với một khoảng thời gian $\Delta_t = t_1 - t_0$, ta có thể sử dụng hàm hợp để dự đoán nhiều bước. Kí hiệu $F^{(traj)}_{\theta,k,\mathcal{M}}$ là hợp của k hàm $f^{(traj)}_{\theta,\mathcal{M}}$:

$$F^{(traj)}_{\theta,k,\mathcal{M}} = \bigcirc_k f^{(traj)}_{\theta,\mathcal{M}} \tag{3.8}$$

Khi đó, với điểm đầu vào $x^{(i)}_{j,\mathcal{M}}$, điểm $x^{(i)}_{j+k,\mathcal{M}}$ có thể được dự đoán với với công thức:

$$\widehat{x^{(i)}_{j+k,\mathcal{M}}} = F^{(traj)}_{\theta,k,\mathcal{M}}\left(x^{(i)}_{j,\mathcal{M}}\right) \tag{3.9}$$

Dựa vào công thức (3.9), hàm tổn thất cho quỹ đạo $\{x^{(i)}_{j,\mathcal{M}}\}^{m}_{j=0}$ có thể được viết dưới dạng:

$$L^{(i)}_{\mathcal{M}}(\theta) = \frac{1}{m-1}\sum_{j=1}^{m-1} || \ F^{(traj)}_{\theta,j,\mathcal{M}}\left(x^{(i)}_{0,\mathcal{M}}\right) - x^{(i)}_{j,\mathcal{M}} \ ||_2 \tag{3.10}$$

Khi đó, hàm tổn thất cho bộ dữ liệu huấn luyện quỹ đạo $\mathcal{M}^{(traj)}(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$:

$$L^{(traj)}_{2,\mathcal{M}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L^{(i)}_{\mathcal{M}}(\theta) \tag{3.11}$$

Mạng neuron $f^{(traj)}_{\theta,\mathcal{M}}$ sẽ được huấn luyện với cả hai hàm tổn thất (3.7) và (3.11). Đối với cả hai hàm tổn thất, mạng neuron MLP $f^{(traj)}_{\theta,\mathcal{M}}$ được huấn luyện theo phương pháp Adam với 1500 vòng lặp. Trọng số tối ưu của mạng neuron MLP với hàm tổn thất $L^{(traj)}_{1,\mathcal{M}}$ là $\theta^*_{1,\mathcal{M}}$, trong khi đó trọng số tối ưu của mạng neuron MLP với hàm tổn thất $L^{(traj)}_{2,\mathcal{M}}$ là $\theta^*_{2,\mathcal{M}}$. Kết quả huấn luyện được cho trong hình 3.10. Ta có thể thấy trong cả hai trường hợp, hàm tổn thất có xu hướng giảm và đạt giá trị nhỏ nhất tại vòng lặp 1500.

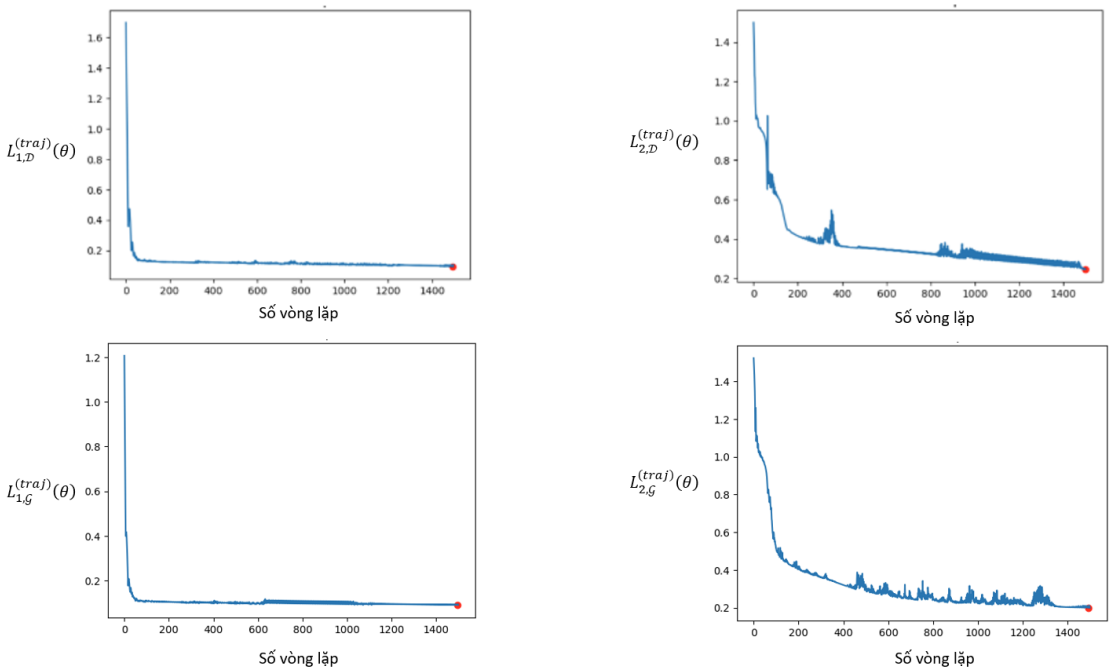
Để kiểm tra độ hiệu quả, $f^{(traj)}_{\theta,\mathcal{M}}$ sẽ được kiểm thử trên bộ dữ liệu $\mathcal{M}_t(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$ trong mục 3.1.2.. Kết quả được cho trong hình 3.11 với sai số lần lượt là:

$$L(\theta^*_{1,\mathcal{D}}) = \frac{1}{|\ \mathcal{D}_t\ |}\sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{D}_t} || \ f^{(traj)}_{\theta^*_{1,\mathcal{D}}}(\widetilde{x}_i) - \widetilde{y}_i \ ||_2 \approx 0.6179 \tag{3.12}$$

$$L(\theta^*_{2,\mathcal{D}}) = \frac{1}{|\ \mathcal{D}_t\ |}\sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{D}_t} || \ f^{(traj)}_{\theta^*_{2,\mathcal{D}}}(\widetilde{x}_i) - \widetilde{y}_i \ ||_2 \approx 0.3909 \tag{3.13}$$

$$L(\theta^*_{1,\mathcal{G}}) = \frac{1}{|\ \mathcal{G}_t\ |}\sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{G}_t} || \ f^{(traj)}_{\theta^*_{1,\mathcal{G}}}(\widetilde{x}_i) - \widetilde{y}_i \ ||_2 \approx 0.6677 \tag{3.14}$$

$$L(\theta^*_{2,\mathcal{G}}) = \frac{1}{|\ \mathcal{G}_t\ |}\sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{G}_t} || \ f^{(traj)}_{\theta^*_{2,\mathcal{G}}}(\widetilde{x}_i) - \widetilde{y}_i \ ||_2 \approx 0.3546 \tag{3.15}$$

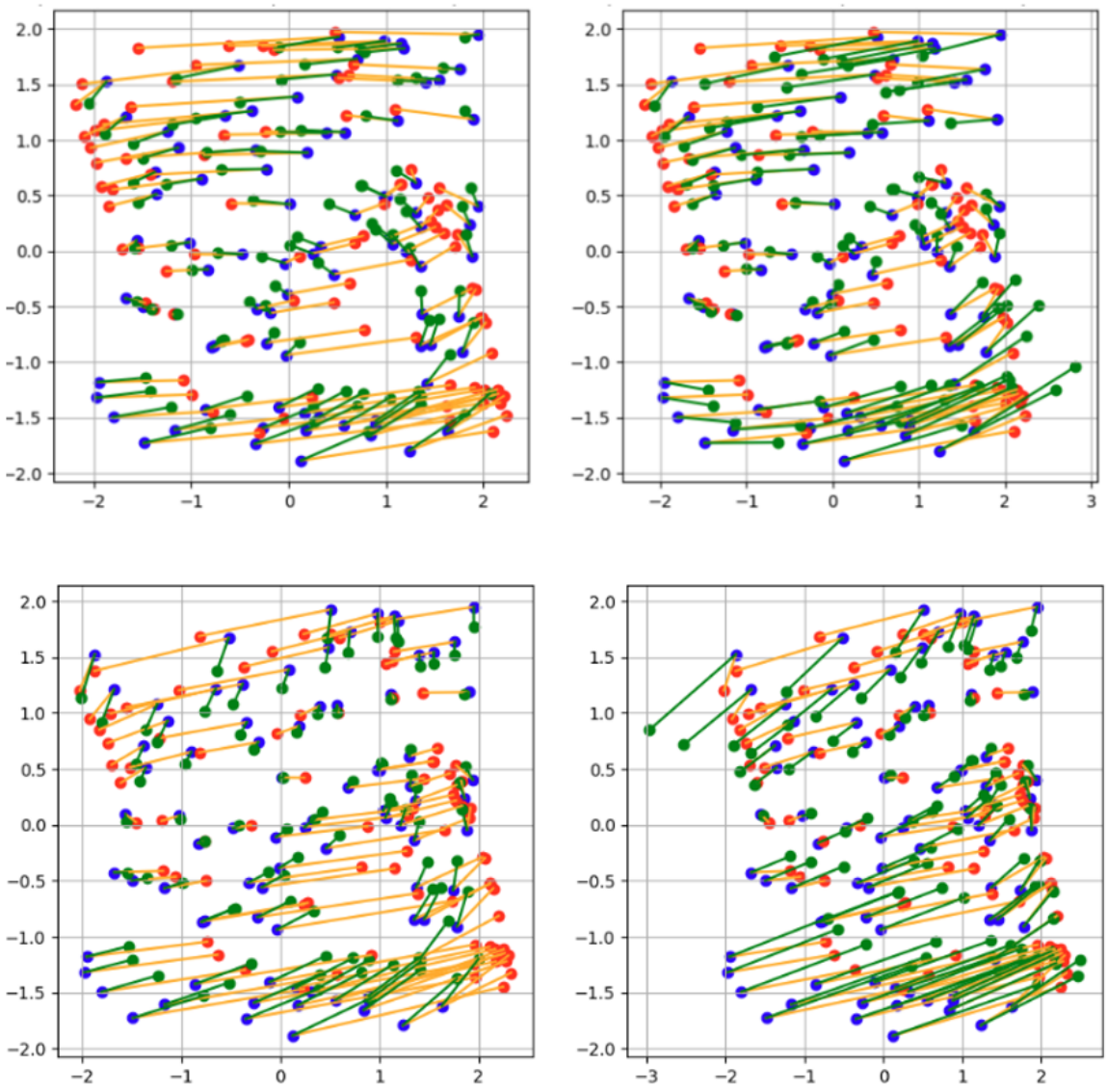


Hình 3.10: Quá trình huấn luyện mạng neuron MLP với bộ dữ liệu quỹ đạo. (**Trái trên**) Quá trình huấn luyện với hàm tổn thất $L_{1,\mathcal{D}}^{(traj)}(\theta)$; (**Phải trên**) Quá trình huấn luyện với hàm tổn thất $L_{2,\mathcal{D}}^{(traj)}(\theta)$; (**Trái dưới**) Quá trình huấn luyện với hàm tổn thất $L_{1,\mathcal{G}}^{(traj)}(\theta)$; (**Phải dưới**) Quá trình huấn luyện với hàm tổn thất $L_{2,\mathcal{G}}^{(traj)}(\theta)$

.

So sánh với sai số của phương pháp MLP với bộ dữ liệu đều trong công thức (3.3) và (3.4), phương pháp MLP với bộ dữ liệu quỹ đạo có kết quả kém hơn trong cả hai trường hợp. Điều này có thể được giải thích bởi bộ dữ liệu huấn luyện quỹ đạo $\mathcal{D}^{(traj)}$ và $\mathcal{G}^{(traj)}$ có phân bố phức tạp, dữ liệu tập trung nhiều ở vùng đa tạp chậm và thưa ở vùng đa tạp nhanh. Ta gọi đây là hiện tượng *bộ dữ liệu lệch chuẩn*. Do đó, khi tối ưu hàm tổn thất (3.7) hay (3.11), mạng neuron MLP sẽ tập trung dự đoán về các điểm dữ liệu ở vùng đa tạp chậm. Điều này vô tình khiến cho lớp đầu ra của mạng neuron MLP sẽ có xu hướng hút về vùng đa tạp chậm. Ta có thể dễ thấy trong hình 3.11 các điểm dự đoán màu xanh lá của mạng neuron MLP đều có xu hướng hút về vùng đa tạp chậm trong vùng $[-2,-1]\times[-0.5,0]$ và vùng $[1,2]\times[-0.5,0.5]$ đối với hệ FHN có một điểm cân bằng. Đối với hệ FHN có ba điểm cân bằng, các điểm dự đoán bị hút về

vùng $[-2, -1] \times [0, 0.5]$ và vùng $[1, 2] \times [0, 0.5]$.



Hình 3.11: (**Trái trên**) Kết quả mạng neuron MLP $f^{(traj)}_{\theta^*_{1,\mathcal{D}}}$ trên bộ dữ liệu kiểm tra $\mathcal{D}_t$; (**Phải trên**) Kết quả mạng neuron MLP $f^{(traj)}_{\theta^*_{2,\mathcal{D}}}$ trên bộ dữ liệu kiểm tra $\mathcal{D}_t$; (**Trái dưới**) Kết quả mạng neuron MLP $f^{(traj)}_{\theta^*_{1,\mathcal{G}}}$ trên bộ dữ liệu kiểm tra $\mathcal{G}_t$, (**Phải dưới**) Kết quả mạng neuron MLP $f^{(traj)}_{\theta^*_{2,\mathcal{G}}}$ trên bộ dữ liệu kiểm tra $\mathcal{G}_t$. Màu xanh da trời là các điểm $\widetilde{x_{i,\mathcal{M}}}$, màu đỏ là các điểm $\widetilde{y_{i,\mathcal{M}}}$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x_{i,\mathcal{M}}}$ và $\widetilde{y_{i,\mathcal{M}}}$ tương ứng. Màu xanh lá cây là các điểm dự đoán của mạng neuron MLP $f^{(traj)}_{\theta^*_{i,\mathcal{M}}}$, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của mạng neuron MLP $\widetilde{x_{i,\mathcal{M}}}$.

3.2.3. Áp dụng phương pháp NeuralODE học hệ động lực với bộ dữ liệu quỹ đạo

Hiện tượng lệch chuẩn của bộ dữ liệu huấn luyện quỹ đạo $\mathcal{D}^{(traj)}$ và $\mathcal{G}^{(traj)}$ khiến cho việc học hệ động lực của phương pháp mạng neuron MLP không hiệu quả. Câu hỏi đặt ra là liệu phương pháp NeuralODE có thể học tốt với bộ dữ liệu lệch chuẩn?

Ta sử dụng kiến trúc mạng neuron cho trường vector f_θ được cho trong bảng 3.1. Tương tự như phương pháp mạng neuron MLP, phương pháp NeuralODE cũng có hai hàm tổn thất để huấn luyện:

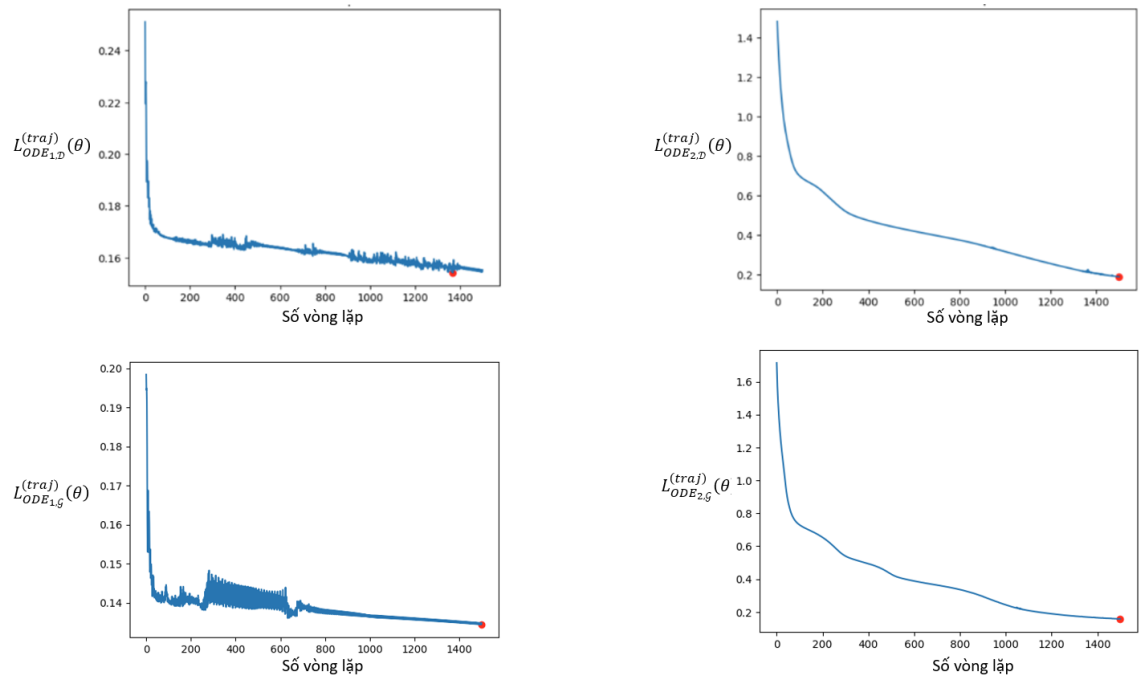
$$L^{(traj)}_{ODE_{1,\mathcal{M}}}(\theta) = \frac{1}{n(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m-1} \left\| \text{ODESolve}\left(x^{(i)}_{j,\mathcal{M}}; f_\theta; t_0 = 0; t_1 = 1\right) - x^{(i)}_{j,\mathcal{M}} \right\|_2 \tag{3.16}$$

$$L^{(traj)}_{ODE_{2,\mathcal{M}}}(\theta) = \frac{1}{n(m-1)} \sum_{i=1}^{n} \sum_{j=1}^{m-1} \left\| \text{ODESolve}\left(x^{(i)}_{0,\mathcal{M}}; f_\theta; t_0 = 0; t_1 = j\right) - x^{(i)}_{j,\mathcal{M}} \right\|_2 \tag{3.17}$$

trong đó hàm tổn thất $L^{(traj)}_{ODE_{1,\mathcal{M}}}$ (3.16) sử dụng NeuralODE dự báo một bước còn hàm tổn thất $L^{(traj)}_{ODE_{2,\mathcal{M}}}$ (3.17) sử dụng NeuralODE dự báo nhiều bước. Ta gọi phương pháp NeuralODE với hàm tổn thất (3.16) là *phương pháp NeuralODE một bước* và phương phóa NeuralODE với hàm tổn thất (3.17) là *phương pháp NeuralODE nhiều bước*. Phương pháp Adam được sử dụng để cập nhật trọng số với 1500 vòng lặp. Trọng số tối ưu với hàm tổn thất $L^{(traj)}_{ODE_{1,\mathcal{M}}}$ là $\theta^*_{1,\mathcal{M}}$ trong khi đó trọng số tối ưu với hàm tổn thất $L^{(traj)}_{ODE_{2,\mathcal{M}}}$ là $\theta^*_{2,\mathcal{M}}$. Quá trình huấn luyện được cho trong hình 3.12.

Hệ số tối ưu $\theta^*_{1,\mathcal{M}}$ của hàm tổn thất $L^{(traj)}_{ODE_{1,\mathcal{M}}}$ và hệ số tối ưu $\theta^*_{2,\mathcal{M}}$ của hàm tổn thất $L^{(traj)}_{ODE_{2,\mathcal{M}}}$ sẽ được kiểm thử với bộ dữ liệu $\mathcal{M}_t(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$ trong mục 3.1.2.. Kết quả được cho trong hình 3.13 với sai số lần lượt là:

$$L_{ODE}(\theta^*_{1,\mathcal{D}}) = \frac{1}{\mid \mathcal{D}_t \mid} \sum_{(\widetilde{x}_i, \widetilde{y}_i) \in \mathcal{D}_t} \left|\left| \text{ODESolve}\left(\widetilde{x}_i; f_{\theta^*_{1,\mathcal{D}}}; t_0 = 0; t_1 = 1\right) - \widetilde{y}_i \right|\right|_2$$



Hình 3.12: Quá trình huấn luyện mạng NeuralODE với bộ dữ liệu quỹ đạo. (**Trái**) Quá trình huấn luyện với hàm tổn thất $L^{(traj)}_{ODE_1}(\theta)$; (**Phải**) Quá trình huấn luyện với hàm tổn thất $L^{(traj)}_{ODE_2}(\theta)$

$$\approx 0.3849 \tag{3.18}$$

$$L_{ODE}(\theta^*_{2,\mathcal{D}}) = \frac{1}{\mid \mathcal{D}_t \mid} \sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{D}_t} \left|\left|\text{ODESolve}\left(\widetilde{x}_i; f_{\theta^*_{2,\mathcal{D}}}; t_0 = 0; t_1 = 1\right) - \widetilde{y}_i\right|\right|_2$$

$$\approx 0.2281 \tag{3.19}$$

$$L_{ODE}(\theta^*_{1,\mathcal{G}}) = \frac{1}{\mid \mathcal{G}_t \mid} \sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{G}_t} \left|\left|\text{ODESolve}\left(\widetilde{x}_i; f_{\theta^*_{1,\mathcal{G}}}; t_0 = 0; t_1 = 1\right) - \widetilde{y}_i\right|\right|_2$$

$$\approx 0.3811 \tag{3.20}$$

$$L_{ODE}(\theta^*_{2,\mathcal{G}}) = \frac{1}{\mid \mathcal{G}_t \mid} \sum_{(\widetilde{x}_i,\widetilde{y}_i)\in\mathcal{G}_t} \left|\left|\text{ODESolve}\left(\widetilde{x}_i; f_{\theta^*_{2,\mathcal{G}}}; t_0 = 0; t_1 = 1\right) - \widetilde{y}_i\right|\right|_2$$

$$\approx 0.2082 \tag{3.21}$$

Kết quả so sánh sai số MSE trên bộ dữ liệu $\mathcal{D}_t$ và $\mathcal{G}_t$ của tất cả phương pháp được cho trong bảng 3.3 và 3.4. Nhìn chung, phương pháp NeuralODE cho kết quả tốt hơn phương pháp MLP. Tuy nhiên, kết quả sai số MSE vẫn cao hơn nhiều khi so sánh với mô hình được huấn luyện với bộ dữ liệu huấn luyện đều

$\mathcal{D}, \mathcal{G}$. Hơn nữa, hình 3.13 cho thấy các điểm dự đoán màu xanh lá vẫn có xu hướng hút về vùng đa tạp chậm. Điều này cho thấy phương pháp NeuralODE cũng gặp khó khăn với những bộ dữ liệu lệch chuẩn.

	MSE trên bộ $\mathcal{D}_t$
MLP với bộ huấn luyện $\mathcal{D}$	0.0553
NeuralODE với bộ huấn luyện $\mathcal{D}$	0.0533
MLP với bộ huấn luyện $\mathcal{D}^{(traj)}$ **và hàm tổn thất** $L_{1,\mathcal{D}}^{(traj)}$	0.6179
MLP với bộ dữ liệu $\mathcal{D}^{(traj)}$ **và hàm tổn thất** $L_{2,\mathcal{D}}^{(traj)}$	0.3909
NeuralODE với bộ huấn luyện $\mathcal{D}^{(traj)}$ **và hàm tổn thất** $L_{ODE_{1,\mathcal{D}}}^{(traj)}$	0.3849
NeuralODE với bộ huấn luyện $\mathcal{D}^{(traj)}$ **và hàm tổn thất** $L_{ODE_{2,\mathcal{D}}}^{(traj)}$	0.2281

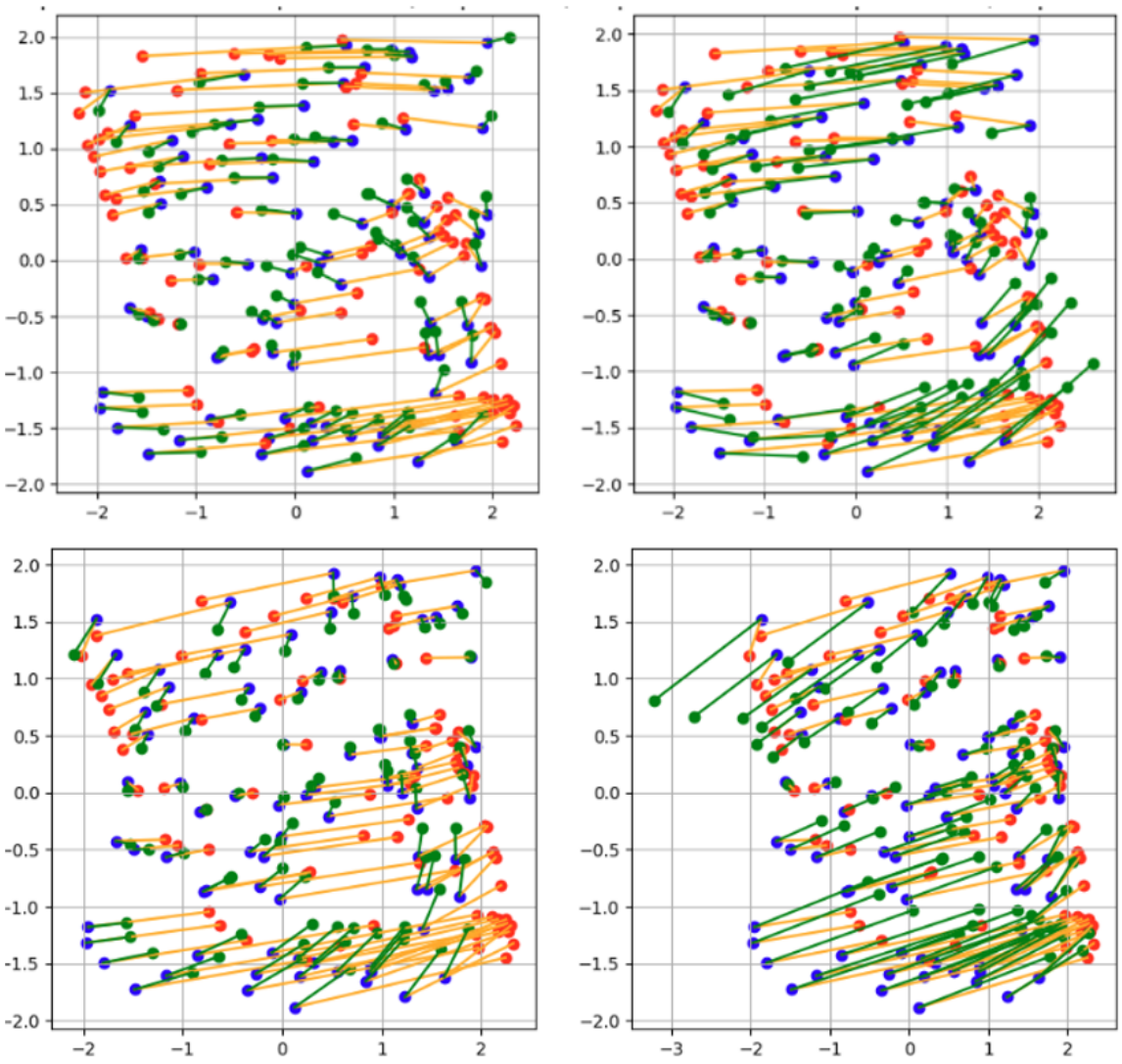
Bảng 3.3: Kết quả so sánh sai số của phương pháp MLP và phương pháp NeuralODE với bộ dữ liệu huấn luyện khác nhau trên bộ dữ liệu kiểm tra $\mathcal{D}_t$.

	MSE trên bộ $\mathcal{G}_t$
MLP với bộ huấn luyện $\mathcal{G}$	0.0505
NeuralODE với bộ huấn luyện $\mathcal{G}$	0.0556
MLP với bộ huấn luyện $\mathcal{G}^{(traj)}$ **và hàm tổn thất** $L_{1,\mathcal{G}}^{(traj)}$	0.6677
MLP với bộ dữ liệu $\mathcal{G}^{(traj)}$ **và hàm tổn thất** $L_{2,\mathcal{G}}^{(traj)}$	0.3546
NeuralODE với bộ huấn luyện $\mathcal{G}^{(traj)}$ **và hàm tổn thất** $L_{ODE_{1,\mathcal{G}}}^{(traj)}$	0.3811
NeuralODE với bộ huấn luyện $\mathcal{G}^{(traj)}$ **và hàm tổn thất** $L_{ODE_{2,\mathcal{G}}}^{(traj)}$	0.2082

Bảng 3.4: Kết quả so sánh sai số của phương pháp MLP và phương pháp NeuralODE với bộ dữ liệu huấn luyện khác nhau trên bộ dữ liệu kiểm tra $\mathcal{G}_t$.

Phân tích ưu, nhược điểm phương pháp NeuralODE

Phương pháp NeuralODE một bước và phương pháp NeuralODE nhiều bước có những ưu nhược điểm sau. Phương pháp NeuralODE một bước sẽ đảm bảo sai số thấp với từng điểm trên quỹ đạo huấn luyện. Do đó, phương pháp NeuralODE một bước sẽ xấp xỉ trường vector của từng điểm trên quỹ đạo huấn luyện. Tuy nhiên, phương pháp NeuralODE một bước không học được sự phụ



Hình 3.13: (**Trái trên**) Kết quả NeuralODE $f^{(traj)}_{\theta^*_{1,\mathcal{D}}}$ trên bộ dữ liệu kiểm tra $\mathcal{D}_t$, (**Phải trên**) Kết quả NeuralODE $f^{(traj)}_{\theta^*_{2,\mathcal{D}}}$ trên bộ dữ liệu kiểm tra $\mathcal{D}_t$, (**Trái dưới**) Kết quả NeuralODE $f^{(traj)}_{\theta^*_{1,\mathcal{G}}}$ trên bộ dữ liệu kiểm tra $\mathcal{G}_t$, (**Phải dưới**) Kết quả NeuralODE $f^{(traj)}_{\theta^*_{2,\mathcal{G}}}$ trên bộ dữ liệu kiểm tra $\mathcal{G}_t$. Màu xanh da trời là các điểm $\widetilde{x_i}$, màu đỏ là các điểm $\widetilde{y_i}$, đoạn thẳng màu cam kết nối các điểm $\widetilde{x_i}$ và $\widetilde{y_i}$ tương ứng. Màu xanh lá cây là các điểm dự đoán của NeuralODE $f^{(traj)}_{\theta^*_{i,\mathcal{M}}}$, đoạn thẳng màu xanh lá cây nối điểm dự đoán với dữ liệu đầu vào của NeuralODE $\widetilde{x_i}$.

thuộc giữa các điểm trên một quỹ đạo. Điều này có thể khiến việc dữ báo nhiều bước bị sai lớn do sai số của từng bước cộng dồn lại. Ngược lại, phương pháp NeuralODE nhiều bước sẽ học được sự phụ thuộc giữa các điểm trên một quỹ đạo. Tuy nhiên, trong nhiều trường hợp, phương pháp NeuralODE nhiều bước

sẽ không học được trường vector của các điểm trên quỹ đạo huấn luyện. Ví dụ trong hình 3.14 sẽ làm rõ về vấn đề này. Hình 3.14 bao gồm quỹ đạo màu xanh nước biển là quỹ đạo huấn luyện của hệ động lực FHN, quỹ đạo màu xanh lá là kết quả huấn luyện của phương pháp NeuralODE một bước khi dự đoán một bước, quỹ đạo màu cam là kết quả huấn luyện của phương pháp NeuralODE một bước khi khi dự đoán cho toàn bộ quỹ đạo, cuối cùng quỹ đạo màu đỏ là kết quả huấn luyện của phương pháp NeuralODE nhiều bước khi dự đoán cả quỹ đạo. Có thể thấy ngay cả khi kết thúc quá trình huấn luyện, phương pháp NeuralODE nhiều bước cũng không thể học chính xác toàn bộ quỹ đạo huấn luyện. Hơn thế nữa, hình 3.14 cũng cho thấy phương pháp NeuralODE nhiều bước dự đoán không chính xác điểm 1 của quỹ đạo huấn luyện. Do đó, trường vector tại vị trí điểm 1 của quỹ đạo huấn luyện sẽ không được học. Ngược lại, do đặc điểm của hàm tổn thất, phương pháp NeuralODE một bước có thể học được khá chính xác trường vector điểm 1 trong quỹ đạo huấn luyện. Điều này được phản ánh bởi quỹ đạo màu xanh lá khi quỹ đạo dự đoán điểm số 2 gần như trùng với điểm 2 trong quỹ đạo huấn luyện. Tuy nhiên, do không học được tính phụ thuộc của các điểm trên quỹ đạo, phương pháp NeuralODE một bước khi dự đoán nhiều bước (quỹ đạo màu cam) cho sai số cao hơn so với phương pháp NeuralODE nhiều bước (quỹ đạo màu đỏ).

3.2.4. Mối quan hệ với định lý Ergodic-Birkhoff

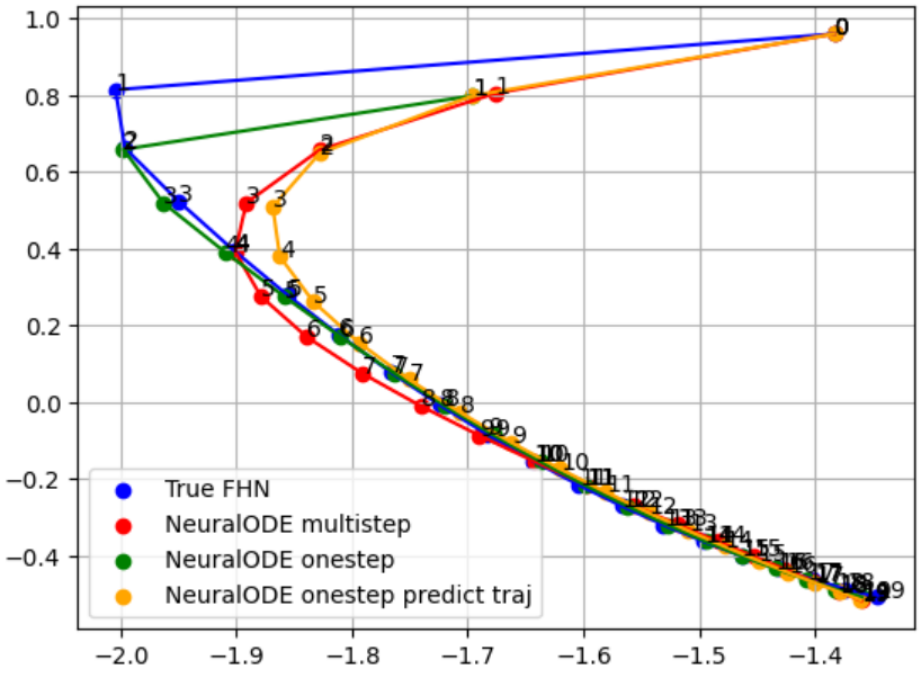
Định lý Ergodic-Birkhoff phát biểu như sau:

Định lí 3.1. ***(Định lý Ergodic-Birkhoff [28])****: Cho* $(\mathcal{X}, \Sigma, \mu, f)$ *là hệ động lực bảo toàn độ đo. Khi đó, với mọi ánh xạ* $\mu-$*khả tích* g $(g \in \mathcal{L}^1_\mu)$*, giới hạn*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} g\left(f^i(x)\right)$$

hội tụ hầu chắc chắn tới ánh xạ $f-$*bất biến* $\hat{g} \in \mathcal{L}^1_\mu$ $(\hat{g} \circ f = \hat{g})$ *và*

$$\int g d\mu = \int \hat{g} d\mu,$$



Hình 3.14: Kết quả huấn luyện phương pháp NeuralODE một bước và NeuralODE nhiều bước.

Hơn nữa, nếu ánh xạ f ergodic,

$$\int g d\mu = \hat{g}.$$

Từ định lý 3.1, nếu ta đặt

$$f(x) = \phi(t_1 - t_0, x) \tag{3.22}$$

$$g(x) = \|f(x) - f_\theta(x)\|_2, \tag{3.23}$$

trong đó $\phi : T \times \mathcal{X} \to \mathcal{X}$ là hệ động lực bất kì. Khi đó,

$$\begin{aligned} g\left(f^i(x)\right) = \left\|f\left(f^i(x)\right) - f_\theta\left(f^i(x)\right)\right\|_2 &= \left\|f^{i+1}(x) - f_\theta\left(f^i(x)\right)\right\|_2 \\ &= \left\|x_{i+1} - f_\theta\left(x_i\right)\right\|_2 \end{aligned}$$

với $x_i = \phi\left(i\left(t_1 - t_0\right), x\right)$. Nếu hệ động lực ϕ là ergodic, từ định lý 3.1, ta có kết quả sau:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \left\|x_{i+1} - f_\theta\left(x_i\right)\right\|_2 = \int \left\|f(x) - f_\theta(x)\right\|_2 d\mu \tag{3.24}$$

Vế phải của biểu thức (3.24) là kì vọng hàm tổn thất MSE trên toàn bộ không gian $\mathcal{X}$. Vế trái của biểu thức (3.24) là trung bình hàm tổn thất MSE trên quỹ đạo $\tau = \{x_0 = x, x_1, x_2, ...\}$ của hệ động lực ϕ với $x \in \mathcal{X}$ bất kì. Nếu n đủ lớn, tối ưu mạng neuron theo hàm mất mát với quỹ đạo τ (vế trái công thức 3.24) tương đương việc tối ưu mạng neuron trên toàn bộ không gian $\mathcal{X}$ (vế phải công thức 3.24), Khi đó, kết quả huấn luyện của mạng neuron sẽ đạt kết quả tốt trên toàn bộ không gian $\mathcal{X}$ cho dù bộ dữ liệu huấn luyện chỉ bao gồm một quỹ đạo τ.

Tuy nhiên, nếu hệ động lực ϕ không ergodic như hệ FHN, dựa trên định lý 3.1, ứng với từng điểm ban đầu $x_0^{(k)}$, hàm tổn thất MSE với quỹ đạo $\tau^{(k)} = \{x_i^{(k)}\}_{i=0,1,2,...}$ sẽ hội tụ về các hàm $\widehat{g^{(k)}}$ khác nhau:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \left\| x_{i+1}^{(k)} - f_\theta \left(x_i^{(k)} \right) \right\| = \widehat{g^{(k)}} \left(x_0^{(k)} \right)$$

Khi đó, cho dù độ dài quỹ đạo lớn hay nhỏ, việc tối ưu mạng neuron với hàm tổn thất MSE trên các quỹ đạo $\left\{\tau^k\right\}_{k=1,2,..}$ sẽ không đảm bảo tối ưu mạng neuron với hàm tổn thất MSE trên toàn bộ không gian $\mathcal{X}$. Khi ấy, việc học có thể không chính xác trên toàn bộ không gian.

Do đó, ta cần có một phương pháp khác hiệu quả hơn cho dữ liệu huấn luyện được sinh ra bởi các quỹ đạo từ hệ động lực phức tạp như hệ FHN. Mục 3.3 đề xuất một phương pháp học hiệu quả hơn hệ động lực có tập hút phức tạp như hệ FHN đối với bộ dữ liệu sinh ra từ các quỹ đạo. Do phương pháp NeuralODE tỏ ra hiệu quả hơn phương pháp MLP đối với bộ dữ liệu quỹ đạo, các phương pháp được giới thiệu trong mục 3.3 sẽ chỉ áp dụng cho phương pháp NeuralODE. Tuy nhiên, các phương pháp đề xuất hoàn toàn có thể mở rộng cho MLP. Các bài nghiên cứu tương lai sẽ đi sâu hơn về việc cải thiện phương pháp MLP.

3.3 Phương pháp đề xuất

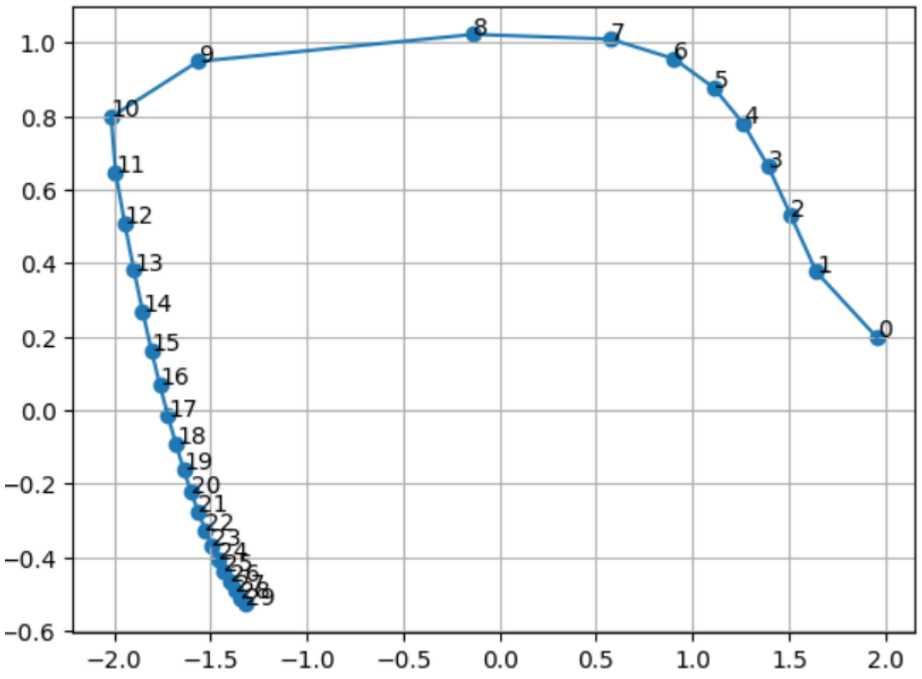
3.3.1. Phép chuẩn hóa liên hợp

Xét một quỹ đạo $\tau = \{x_j\}_{j=0}^{m}$ là nghiệm của một phương trình vi phân thường $\frac{dx}{dt} = f(x)$ trong đó $f : \mathbb{R}^d \to \mathbb{R}^d$ và các điểm kề nhau đều cách nhau một khoảng thời gian Δ_t. Phương pháp NeuralODE sẽ mô hình hóa trường vector $f(x)$ bằng một mạng neuron $f_\theta(x)$ và tối ưu hàm tổn thất sau:

$$L_{ODE}(\theta) = \frac{1}{m}\sum_{j=1}^{m} \|\text{ODESolve}\left(x_0; f_\theta; t_0 = 0; t_1 = j\Delta_t\right) - x_j\|_2 \qquad (3.25)$$

Chú ý ta lựa chọn phương pháp NeuralODE dự báo nhiều bước để xây dựng công thức hàm tổn thất (3.25) do độ hiệu quả của phương pháp này tốt hơn so với phương pháp NeuralODE dự báo một bước. Về mặt ý nghĩa, công thức hàm tổn thất (3.25) lấy trung bình sai số của phương pháp NeuralODE khi dự đoán các điểm dữ liệu x_j nằm trên quỹ đạo τ. Khi đó, mỗi điểm dữ liệu x_j sẽ có **độ quan trọng** như nhau (đều bằng $\frac{1}{m}$) khi huấn luyện mô hình NeuralODE. Tuy nhiên, trong nhiều trường hợp, độ quan trọng của từng điểm dữ liệu trên quỹ đạo không bằng nhau. Ví dụ xét một quỹ đạo của hệ động lực FHN như trong hình 3.15. Các điểm quỹ đạo thứ $7, 8, 9, 10$ nên có độ quan trọng cao hơn so với những điểm còn lại do độ phủ dữ liệu xung quanh các điểm đó thưa hơn so với những điểm dữ liệu còn lại. Nếu độ quan trọng của các điểm dữ liệu trong hình 3.15 đều bằng nhau, các mô hình mạng neuron nói chung sẽ ưu tiên học các điểm dữ liệu ở vùng trù mật hơn. Hiện tượng này đã được thấy rõ trong mục 3.2. Do đó, việc xác định độ quan trọng cho từng điểm dữ liệu huấn luyện sẽ giúp cho các phương pháp mạng neuron nói chung và phương pháp NeuralODE nói riêng tránh khỏi tình trạng học lệch [29]. Tuy nhiên, việc xác định độ quan trọng cho từng điểm dữ liệu là một điều không dễ và cũng là một chủ đề đang được các nhà khoa học nghiên cứu sâu [30].

Thay vì đi xác định độ quan trọng của điểm dữ liệu, chúng tôi đề xuất học phép chuẩn hóa lên quỹ đạo τ, biến đổi quỹ đạo τ thành quỹ đạo τ' "dễ học"



Hình 3.15: Ví dụ một quỹ đạo của hệ FHN

hơn. Khái niệm "dễ học" đối với một quỹ đạo trong hệ động lực là một khái niệm trừu tượng và theo hiểu biết của chúng tôi chưa được các bài báo khác nghiên cứu. Dựa vào các phân tích trong mục 3.2, chúng tôi đề xuất khái niệm một quỹ đạo "dễ học" đối với phương pháp mạng neuron thỏa mãn ba điều kiện sau:

1. Các điểm dữ liệu trên quỹ đạo không quá cách xa nhau. (C_1)
2. Các điểm dữ liệu trên quỹ đạo không quá gần nhau. (C_2)
3. Hướng trường vector tại các điểm trên quỹ đạo thay đổi ít. (C_3)

Điều kiện (C_1) chỉ ra rằng độ phủ của các điểm dữ liệu trên quỹ đạo không quá thưa. Điều kiện (C_2) chỉ ra rằng các điểm dữ liệu không được quá gần nhau khiến cho việc huấn luyện của các phương pháp mạng neuron sẽ tập trung học vào vùng trù mật đó. Điều kiện (C_3) đến từ quan sát những vùng khó học nhất trên quỹ đạo là những vùng dữ liệu mà hướng trường vector thay đổi nhiều. Ví dụ trong hình 3.15, điểm dữ liệu thứ 9 và 10 là vùng khó học khi hướng của trường vector thay đổi đột ngột. Khi điều kiện (C_3) được đảm bảo, quỹ đạo sẽ

trơn. Trong điều kiện lý tưởng khi hướng của trường vector không thay đổi, ta sẽ có một quỹ đạo tạo thành một đường thẳng.

Dựa trên ba điều kiện trên, chúng tôi đề xuất học một phép biến đổi vi phôi h_ϕ. Phép biến đổi vi phôi $h_\phi : \mathbb{R}^d \to \mathbb{R}^d$ biến đổi quỹ đạo $\tau = \{x_j\}_{j=0}^m$ thành quỹ đạo $\tau' = \{z_j\}_{j=0}^m$ $(h_\phi(x_j) = z_j \forall j = 1, 2, .., m)$ thỏa mãn tốt nhất ba điều kiện (C_1),(C_2),(C_3).

Nếu h là phép vi phôi biến đổi quỹ đạo τ thành quỹ đạo τ', ta có thể suy ra được quỹ đạo τ' là nghiệm của bài toán IVP $\frac{dz}{dt} = g(z)$ với điểm ban đầu $z(t_0) = z_0$ và:

$$f(x) = M^{-1}(x)g(h(x)) \tag{3.26}$$

trong đó $M(x) = \frac{\partial h(x)}{\partial x}$. Phương trình vi phân $\frac{dz}{dt} = g(z)$ còn được gọi là phương trình vi phân liên hợp đối với phương trình vi phân $\frac{dx}{dt} = f(x)$. Do đó, ta có thể gọi phép biến đổi h là **phép chuẩn hóa liên hợp**.

Để huấn luyện phép biến đổi vi phôi h_ϕ, ta cần xác định hàm tổn thất. Dựa trên điệu kiện (C_1), (C_2), (C_3), ta có các hàm tổn thất $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ tương ứng sau:

$$\mathcal{L}_1(\phi) = \max_{j=\overline{1,m}} \frac{q_j}{\sum_{j=1}^m q_j} \tag{3.27}$$

$$\mathcal{L}_2(\phi) = -\min_{j=\overline{1,m}} \frac{q_j}{\sum_{j=1}^m q_j} \tag{3.28}$$

$$\mathcal{L}_3(\phi) = -\min_{j=\overline{1,m-1}} \frac{d_j \cdot d_{j+1}}{q_j q_{j+1}} \tag{3.29}$$

trong đó $d_j = z_j - z_{j-1} \forall j = \overline{1, m}; q_j = ||d_j||_2 \forall j = \overline{1, m}$.

Tối thiểu hàm tổn thất (3.27) sẽ tối thiểu khoảng cách lớn nhất giữa hai điểm dữ liệu liên tiếp trên quỹ đạo τ'. Tối thiểu hàm tổn thất (3.28) sẽ cực đại hóa khoảng cách bé nhất giữa hai điểm dữ liệu liên tiếp trên quỹ đạo τ'. Cuối cùng, khi tối thiểu hàm tổn thất (3.29), ta muốn góc giữa hai vector $\angle d_j, d_{j+1}$ gần 0 nhất có thể.

Cuối cùng, hàm tổn thất để huấn luyện phép vi phôi h là:

$$\mathcal{L}(\phi) = \mathcal{L}_1(\phi) + \mathcal{L}_2(\phi) + \mathcal{L}_3(\phi) \tag{3.30}$$

Chú ý rằng, hàm tổn thất $\mathcal{L}(\phi)$ trong công thức (3.30) không có nhãn. Do đó, bài toán huấn luyện phép vi phôi h là bài toán học không giám sát.

3.3.2. NeuralODE với phép chuẩn hóa liên hợp học hệ động lực

Sau khi học được phép biến đổi vi phôi h, ta có bộ dữ liệu huấn luyện quỹ đạo dễ học hơn $\tau' = \{z_j\}_{j=1}^m$ tuân theo phương trình vi phân liên hợp $\frac{dz}{dt} = g(z)$. Trường vector $g(.)$ được xấp xỉ bằng mạng neuron MLP g_φ và sử dụng phương pháp NeuralODE để học bộ dữ liệu quỹ đạo τ' này. Khi đó, do đã chuẩn hóa liên hợp để các điểm dữ liệu có độ quan trọng tương đương nhau, ta có thể sử dụng hàm tổn thất sau:

$$\widehat{L_{ODE}(\varphi)} = \frac{1}{m}\sum_{j=1}^{m} \left\| \text{ODESolve}\left(z_0; g_\varphi; t_0 = 0; t_1 = j\Delta_t\right) - z_j \right\|_2 \tag{3.31}$$

Sau khi tối ưu hàm tổn thất $\widehat{L_{ODE}(\varphi)}$, ta được trọng số tối ưu φ^*. Do đã được chuẩn hóa liên hợp, g_{φ^*} sẽ xấp xỉ tốt được hệ động lực liên hợp $\frac{dz}{dt} = g(z)$ trên bộ dữ liệu quỹ đạo τ'. Tuy nhiên, mục tiêu là học hệ động lực ban đầu $f(.)$. Do đó, dựa vào mối liên hệ giữa h, g và f trong công thức (3.26), ta có thể xây dựng bộ dữ liệu huấn luyện $\mathcal{D}_f = \left\{ x_j, \left[\frac{\partial h_{\phi^*}}{\partial x}(x_j)\right]^{-1} g_{\varphi^*}(h_{\phi^*}(x_j)) \right\}_{j=0}^{m}$ và huấn luyện hàm f_θ với hàm tổn thất sau:

$$\widehat{L(\theta)} = \frac{1}{m+1}\sum_{j=0}^{m} \left\| f_\theta(x_j) - \left[\frac{\partial h_{\phi^*}}{\partial x}(x_j)\right]^{-1} g_{\varphi^*}(h_{\phi^*}(x_j)) \right\|_2 \tag{3.32}$$

Ngoài ra, khi có nhiều quỹ đạo $\left\{ \left\{ x_j^{(i)} \right\}_{j=0}^{m} \right\}_{i=1}^{n}$, ta có thể huấn luyện f_θ tương tự với trường hợp chỉ có 1 quỹ đạo. Ứng với từng quỹ đạo $\left\{ x_j^{(i)} \right\}_{j=0}^{m}$, ta sẽ huấn luyện một phép chuẩn hóa liên hợp h_{ϕ_i} tương ứng. Sau khi huấn luyện phép chuẩn hóa liên hợp h_{ϕ_i} thỏa mãn ba điều kiện (C_1), (C_2), (C_3), ta được các trọng số tối ưu ϕ_i^* cùng với quỹ đạo liên hợp tương ứng $\left\{ z_j^{(i)} \right\}_{j=0}^{m}$. Ứng với mỗi quỹ đạo liên hợp $\left\{ z_j^{(i)} \right\}_{j=0}^{m}$, ta sử dụng phương pháp NeuralODE để học trường

vector tối ưu $g_{\varphi_i^*}$. Sau khi học được các tham số tối ưu $\{\phi_i^*, \varphi_i^*\}_{i=1}^n$, ta xây dựng bộ dữ liệu $\mathcal{D}_f = \bigcup_{i=1}^n \left\{ x_j^{(i)}, \left[\frac{\partial h_{\phi_i^*}}{\partial x}(x_j^{(i)})\right]^{-1} g_{\varphi_i^*}(h_{\phi_i^*}(x_j^{(i)})) \right\}_{j=0}^m$ và huấn luyện f_θ với hàm tổn thất sau:

$$\widehat{L(\theta)} = \frac{1}{n(m+1)} \sum_{i=1}^{n} \sum_{j=0}^{m} \left\| f_\theta\left(x_j^{(i)}\right) - \left[\frac{\partial h_{\phi_i^*}}{\partial x}(x_j^{(i)})\right]^{-1} g_{\varphi_i^*}\left(h_{\phi_i^*}\left(x_j^{(i)}\right)\right) \right\|_2 \tag{3.33}$$

Mã giả thuật toán được viết ở trong thuật toán 4.

Algorithm 4 Thuật toán đề xuất

1: **Đầu vào:** Bộ dữ liệu quỹ đạo $\mathcal{D}^{(traj)} = \left\{ \left\{ x_j^{(i)} \right\}_{j=0}^m \right\}_{i=1}^n$, các mạng neuron $\{h_{\phi_i}\}_{i=1}^n, \{g_{\varphi_i}\}_{i=1}^n, f_\theta$.

2: # Học phép chuẩn hóa liên hợp

3: **for** i=1,2,...,n **do**

4: Tìm $\phi_i^* \in \min_{\phi_i} \mathcal{L}(\phi_i)$ trong đó $\mathcal{L}(\phi_i)$ được cho trong công thức số (3.30) với bộ dữ liệu quỹ đạo $\left\{ x_j^{(i)} \right\}_{j=0}^m$.

5: Áp dụng phép biến đổi $h_{\phi_i^*}$ lên $\left\{ x_j^{(i)} \right\}_{j=0}^m$ thu được quỹ đạo liên hợp $\left\{ z_j^{(i)} \right\}_{j=0}^m$

6: **end for**

7: # Học phương trình vi phân thường liên hợp

8: **for** i=1,2,...,n **do**

9: Tìm $\varphi_i^* \in \min_{\varphi_i} \widehat{L_{ODE}(\varphi_i)}$ trong đó $\widehat{L_{ODE}(\varphi_i)}$ được cho trong công thức số (3.31) với bộ dữ liệu quỹ đạo $\left\{ z_j^{(i)} \right\}_{j=0}^m$.

10: Xây dựng bộ dữ liệu $\mathcal{D}_f^{(i)} = \left\{ x_j^{(i)}, \left[\frac{\partial h_{\phi_i^*}}{\partial x}(x_j^{(i)})\right]^{-1} g_{\varphi_i^*}(h_{\phi_i^*}(x_j^{(i)})) \right\}_{i=0}^m$.

11: **end for**

12: Xây dựng bộ dữ liệu $\mathcal{D}_f = \bigcup_{i=1}^n \mathcal{D}_f^{(i)}$.

13: Tìm $\theta^* \in \min_\theta \widehat{L_\theta}$ trong đó $\widehat{L_\theta}$ được cho trong công thức số (3.33).

14: **Đầu ra:** Bộ trọng số tối ưu θ^*

3.3.3. Thảo luận

So sánh với phương pháp NeuralODE

Khác với phương pháp NeuralODE một bước, phương pháp đề xuất sử dụng hàm tổn thất của phương pháp NeuralODE nhiều bước để học quỹ đạo liên hợp. Do đó, phương pháp đề xuất học được tính phụ thuộc giữa các điểm trên một quỹ đạo, khắc phục được hạn chế của phương pháp NeuralODE một bước. Do quỹ đạo liên hợp dễ học hơn, trường vector của các điểm trên quỹ đạo liên hợp cũng được xấp xỉ tốt hơn so với trường vector các điểm trên quỹ đạo gốc. Ngoài ra, phương pháp đề xuất sử dụng hàm tổn thất trong công thức (3.33) để học ngược lại trường vector của hệ động lực gốc. Vì vậy, trường vector của các điểm trên quỹ đạo gốc sẽ được học, khắc phục điểm yếu của phương pháp NeuralODE nhiều bước. Do đó, ta có thể nói phương pháp đề xuất có thể khắc phục được những nhược điểm đồng thời tận dụng được những ưu điểm của cả hai phương pháp NeuralODE một bước và NeuralODE nhiều bước.

Ưu điểm của phép phân tách h và g

Bằng cách huấn luyện phép chuẩn hóa liên hợp h, hệ động lực g được huấn luyện trên bộ dữ liệu quỹ đạo dễ học hơn. Do h là phép vi phôi, ta hoàn toàn có thể biến đổi từ g về f mà không chịu thêm một sai số nào khác. Công thức (3.26) xác định mối liên hệ giữa hệ động lực gốc f, phép chuẩn hóa liên hợp h và hệ động lực liên hợp g. Khi đó, có thể hiểu rằng phương pháp đề xuất đã tách hàm f khó học thành những hàm thành phần khác dễ học hơn, phép chuẩn hóa liên hợp h và hệ động lực liên hợp g. Do vậy, thay vì sử dụng một kiến trúc mạng neuron phức tạp cho hệ động lực f, ta có thể sử dụng các kiến trúc mạng neuron đơn giản (ví dụ ít số lớp ẩn hơn hay ít số nốt trong lớp ẩn hơn) cho phép chuẩn hóa liên hợp h và hệ động lực liên hợp g. Điều này có thể làm cải thiện thời gian và chi phí huấn luyện của phương pháp đề xuất. Việc sử dụng nhiều mạng neuron đơn giản để cấu thành mạng neuron phức tạp là điều nên làm bởi

việc xác định các tham số cho mạng neuron đơn giản sẽ dễ dàng và thuận tiện hơn cho cả người sử dụng lẫn phương pháp huấn luyện.

Sử dụng một phép biến đổi liên hợp toàn cục

Trong thuật toán đề xuất 4, ứng với mỗi quỹ đạo ta cần phải huấn luyện một phép biến đổi liên hợp. Câu hỏi được đặt ra liệu có thể huấn luyện một phép biến đổi liên hợp duy nhất cho toàn bộ quỹ đạo không? Để trả lời câu hỏi đó, xét ví dụ hệ động lực sau:

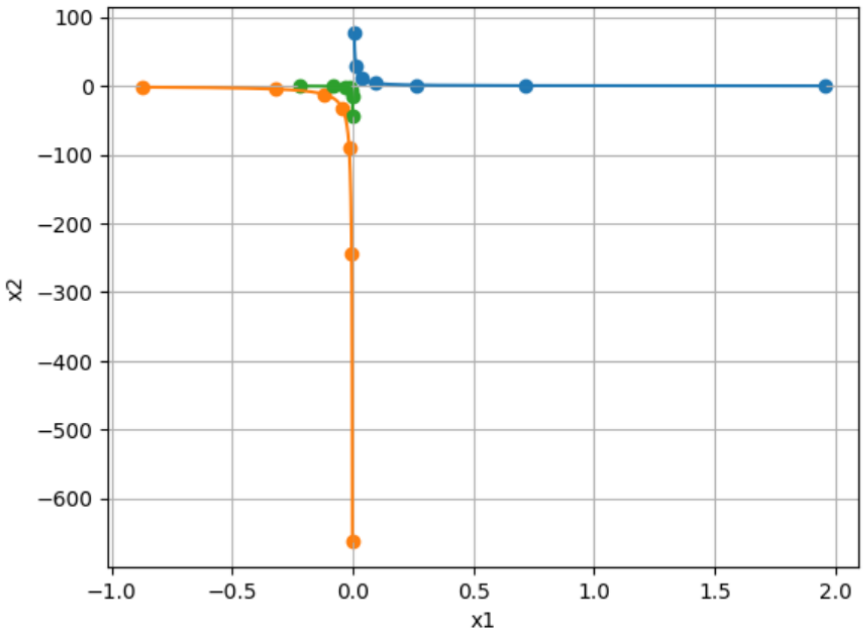
$$\begin{aligned} \frac{dx_1}{dt} &= -x_1 \\ \frac{dx_2}{dt} &= x_2 \end{aligned} \tag{3.34}$$

Hình 3.16 cho thấy 3 quỹ đạo tương ứng của hệ động lực (3.34). Nghiệm của hệ động lực (3.34) là:

$$x_1(t) = c_1 e^{-t} \tag{3.35}$$

$$x_2(t) = c_2 e^{t} \tag{3.36}$$

Nhìn vào hình 3.16 và dựa vào công thức nghiệm (3.35), (3.36) , các điểm dữ liệu sẽ bị hút về 0 ở trục x_1 và sẽ bị đẩy ra 0 ở trục x_2 khi t tiến đến dương vô cùng. Khi đó, dựa vào điều kiện (C_1) và (C_2), một phép biến đổi liên hợp h sẽ phải kéo giãn các điểm trên trục x_1 và co các điểm trên trục x_2 lại gần với nhau. Do đó, việc huấn luyện một phép chuẩn hóa liên hợp h lên toàn bộ quỹ đạo huấn luyện sẽ có độ khó tương đương khi học trực tiếp hệ động lực (3.34). Thay vào đó, nếu mỗi quỹ đạo huấn luyện có một phép chuẩn hóa liên hợp h_{ϕ_i} riêng, ta đã giảm bớt áp lực cho h_{ϕ_i} khi không phải chuẩn hóa các quỹ đạo huấn luyện khác. Từ những lí do trên, việc huấn luyện một phép chuẩn hóa liên hợp chung cho toàn bộ quỹ đạo là không dễ dàng, thậm chí có độ khó tương đương với việc học hệ động lực gốc. Vì vậy, chúng tôi đề xuất dùng các phép chuẩn hóa liên hợp khác nhau cho từng quỹ đạo.



Hình 3.16: Hệ động lực mẫu 3.34

Sự đánh đổi về mặt thời gian huấn luyện

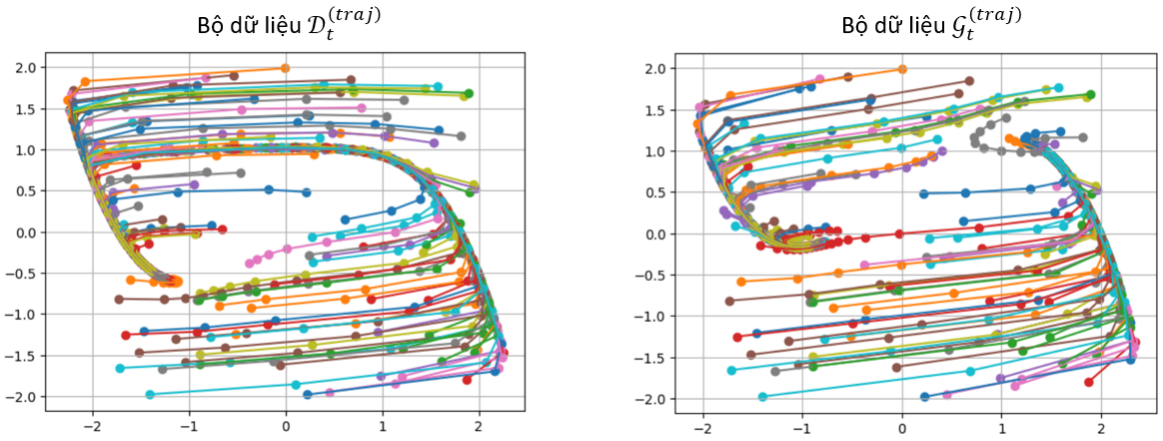
Trong thuật toán 4, để tìm bộ trọng số tối θ^*, ta phải thông qua 3 bước: bước 1 tìm phép chuẩn hóa liên hợp với từng quỹ đạo, bước 2 tìm hệ động lực tương ứng với từng quỹ đạo liên hợp, bước 3 tìm bộ trọng số tối ưu θ^* từ phép chuẩn hóa liên hợp tối ưu và hệ động lực liên hợp tối ưu. Mặt khác, phương pháp NeuralODE có thể áp dụng trực tiếp 1 bước để tìm bộ trọng số tối ưu θ^*. Do vậy, thuật toán đề xuất có thời gian huấn luyện lâu hơn so với các phương pháp thông thường. Tuy nhiên, việc đánh đổi giữa thời gian huấn luyện và độ hiệu quả là một điều chấp nhận được. Các công trình tương lai sẽ giải quyết vấn đề tối ưu thời gian huấn luyện của thuật toán đề xuất.

3.3.4. Kết quả thực nghiệm

Bộ dữ liệu huấn luyện và bộ dữ liệu kiểm tra

Ta sẽ sinh bộ dữ liệu huấn luyện với cả hai trường hợp hệ FHN có một điểm cân bằng và hệ FHN có ba điểm cân bằng như trong mục 3.2.2.. Khi đó,

bộ dữ liệu quỹ đạo huấn luyện cho trường hợp hệ FHN một điểm cân bằng là $\mathcal{D}^{(traj)} = \left\{ \left\{ x_{j,\mathcal{D}}^{(i)} \right\}_{j=0}^{20} \right\}_{i=1}^{10}$ và bộ dữ liệu quỹ đạo huấn luyện cho trường hợp hệ FHN ba điểm cân bằng là $\mathcal{G}^{(traj)} = \left\{ \left\{ x_{j,\mathcal{G}}^{(i)} \right\}_{j=0}^{20} \right\}_{i=1}^{10}$. Ngoài ra, để kiểm tra tính tổng quát hóa của mô hình, bộ dữ liệu kiểm tra được sinh với 100 quỹ đạo. Cụ thể, thuật toán 3 được sử dụng với các tham số $n = 100, m = 21, \mathcal{X}_0 = [-2, 2]^2, t_0 = 0, t_1 = 1$ để sinh bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)} = \left\{ \left\{ \overline{x_{j,\mathcal{D}}^{(i)}} \right\}_{j=0}^{20} \right\}_{i=1}^{100}$ cho trường hợp hệ FHN có một điểm cân bằng và $\mathcal{G}_t^{(traj)} = \left\{ \left\{ \overline{x_{j,\mathcal{G}}^{(i)}} \right\}_{j=0}^{20} \right\}_{i=1}^{100}$ cho trường hợp hệ FHN có ba điểm cân bằng. Hình 3.17 cho ta thấy bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}, \mathcal{G}_t^{(traj)}$.



Hình 3.17: **(Trái)** Bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}$; **(Phải)** Bộ dữ liệu kiểm tra $\mathcal{G}_t^{(traj)}$.

Thí nghiệm về phép chuẩn hóa liên hợp

Trong thuật toán 4, bước đầu tiên là học phép chuẩn hóa liên hợp thỏa mãn tốt nhất ba điều kiện (C_1), (C_2), (C_3). Mỗi quỹ đạo i ($i \in \{0, 1, 2\}$) có tương ứng một phép chuẩn hóa liên hợp h_{ϕ_i}. Chú ý rằng mạng neuron h_{ϕ_i} cần được xây dựng thỏa mãn điều kiện là một phép vi phôi $\mathbb{R}^2 \to \mathbb{R}^2$. Do đó, dựa trên ý

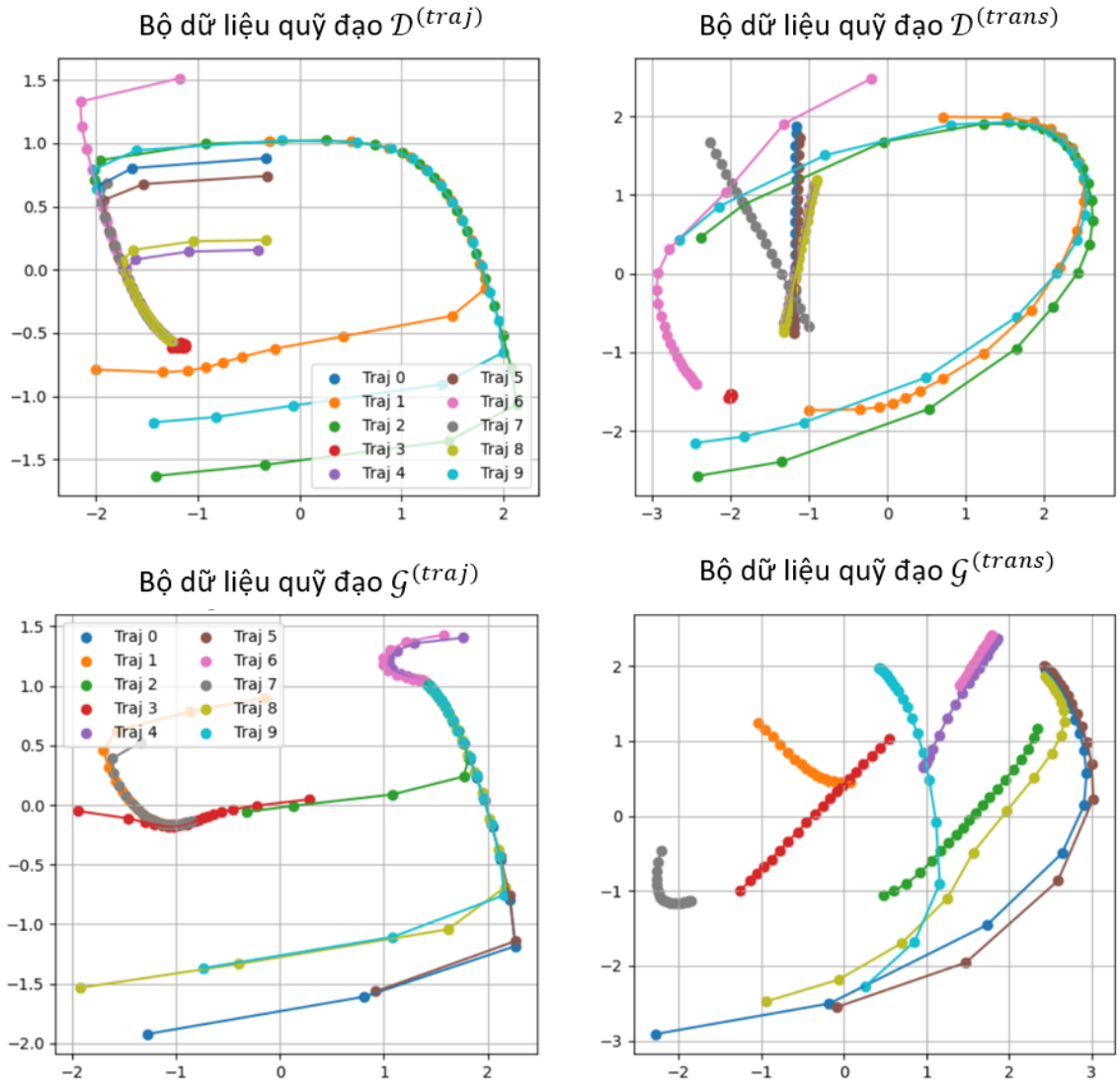
tưởng của [10], ta xây dựng h_{ϕ_i} dưới dạng phép biến đổi NeuralODE. Cụ thể,

$$h_{\phi_i}(x) = \text{ODESolve}(x; k_{\phi_i}; t_0; t_1) \tag{3.37}$$

trong đó k_{ϕ_i} là các mạng neuron $\mathbb{R}^2 \to \mathbb{R}^2$ cần được huấn luyện. Để xây dựng h_{ϕ_i} dựa trên công thức (3.37), ta chọn $t_0 = 0; t_1 = 1$ và kiến trúc mạng neuron k_{ϕ_i} được xây dựng tương tự như bảng 3.1. Ứng với mỗi quỹ đạo i, mạng h_{ϕ_i} được huấn luyện bằng phương pháp tối ưu Adam với 1500 vòng lặp và công thức hàm tổn thất được cho trong (3.30). Trọng số tối ưu của từng phép chuẩn hóa liên hợp cho từng quỹ đạo i là ϕ_i^*. Kí hiệu bộ dữ liệu quỹ đạo sau khi áp dụng phép chuẩn hóa liên hợp tối ưu đối với hệ FHN một điểm cân bằng là $\mathcal{D}^{(trans)}$ và đối với hệ FHN ba điểm cân bằng là $\mathcal{G}^{(trans)}$. Kết quả của phép chuẩn hóa liên hợp tối ưu cho từng quỹ đạo được cho trong hình 3.18. Từ hình 3.18 có thể thấy các quỹ đạo liên hợp trơn hơn và khoảng cách giữa các điểm trong quỹ đạo cũng cách đều nhau hơn. Điều này phản ánh đúng ba điều kiện (C_1), (C_2) và (C_3) về hệ động lực dễ học.

Thí nghiệm về việc học hệ động lực đơn giản

Mục này sẽ kiểm thử quỹ đạo liên hợp có dễ học hơn quỹ đạo gốc. Đầu tiên, phương pháp NeuralODE nhiều bước sẽ được áp dụng để học các quỹ đạo gốc. Cụ thể hơn, ta huấn luyện riêng một hệ động lực $f_{\theta_{i,\mathcal{M}}}$ ứng với từng quỹ đạo gốc $\left\{x_{j,\mathcal{M}}^{(i)}\right\}_{j=0}^{20}$ ($\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\}$). Phương pháp NeuralODE nhiều bước cũng sẽ được áp dụng để học quỹ đạo liên hợp. Kí hiệu bộ dữ liệu quỹ đạo liên hợp đối với hệ FHN một điểm cân bằng là $\mathcal{D}^{(trans)} = \left\{\left\{z_{j,\mathcal{D}}^{(i)}\right\}_{j=0}^{20}\right\}_{i=1}^{10}$ và với hệ FHN ba điểm cân bằng là $\mathcal{G}^{(trans)} = \left\{\left\{z_{j,\mathcal{G}}^{(i)}\right\}_{j=0}^{20}\right\}_{i=1}^{10}$. Ứng với từng quỹ đạo liên hợp $\left\{z_{j,\mathcal{M}}^{(i)}\right\}_{j=0}^{20}$ ($\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\}$), ta sẽ huấn luyện một phương trình vi phân liên hợp tương ứng với trường vector $g_{\varphi_{i,\mathcal{M}}}$ sử dụng phương pháp NeuralODE nhiều bước. Kiến trúc các mạng neuron $g_{\varphi_{i,\mathcal{M}}}(i = \overline{1,10}; \mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$ được xây dựng tương tự như trong bảng 3.1. Phương pháp tối ưu Adam được sử dụng



Hình 3.18: Kết quả huấn luyện phép chuẩn hóa liên hợp. (**Trái trên**) Bộ dữ liệu quỹ đạo gốc $\mathcal{D}^{(traj)}$; (**Phải trên**) Bộ dữ liệu quỹ đạo liên hợp $\mathcal{D}^{(trans)}$; (**Trái dưới**) Bộ dữ liệu quỹ đạo gốc $\mathcal{G}^{(traj)}$; (**Phải dưới**) Bộ dữ liệu quỹ đạo liên hợp $\mathcal{G}^{(trans)}$.

với 1500 vòng lặp để tối ưu hàm tổn thất trong công thức (3.25) cho việc huấn luyện. Kết quả huấn luyện được cho trong bảng 3.5 và bảng 3.6.

Có thể dễ dàng thấy được rằng, với cùng một kiến trúc mạng và thời gian huấn luyện như nhau, kết quả huấn luyện của phương pháp NeuralODE nhiều bước lên bộ dữ liệu liên hợp tốt hơn đáng kể so với bộ dữ liệu gốc. Điều này khẳng định bộ dữ liệu quỹ đạo liên hợp $\mathcal{D}^{(trans)}$, $\mathcal{G}^{(trans)}$ dễ học hơn so với bộ dữ liệu quỹ đạo gốc $\mathcal{D}^{(traj)}$, $\mathcal{G}^{(traj)}$.

Giá trị tối ưu hàm MSE	**Quỹ đạo gốc**	**Quỹ đạo liên hợp**
Quỹ đạo 0	0.0211	0.0143
Quỹ đạo 1	0.2065	0.1366
Quỹ đạo 2	0.1665	0.1434
Quỹ đạo 3	0.0292	0.0114
Quỹ đạo 4	0.0199	0.0109
Quỹ đạo 5	0.0185	0.0152
Quỹ đạo 6	0.0365	0.0508
Quỹ đạo 7	0.0106	0.0145
Quỹ đạo 8	0.0225	0.0121
Quỹ đạo 9	0.2205	0.2063

Bảng 3.5: Kết quả so sánh giá trị tối ưu hàm tổn thất trên bộ dữ liệu $\mathcal{D}^{(traj)}$ và $\mathcal{D}^{(trans)}$.

Thí nghiệm học hệ động lực gốc từ hệ động lực liên hợp

Sau khi tìm trọng số tối ưu cho hệ động lực liên hợp $g_{\varphi^*_{i,\mathcal{M}}}$, hệ động lực gốc $f_{\theta,\mathcal{M}}$ có thể được huấn luyện một cách dễ dàng. Ta huấn luyện $f_{\theta,\mathcal{M}}$ với hàm tổn thất được cho trong công thức (3.33). Một điều lưu ý rằng hàm tổn thất (3.33) dễ tính toán và tối ưu hơn nhiều so với hàm tổn thất huấn luyện phương pháp NeuralODE nhiều bước do ta không phải sử dụng bộ giải số phương trình vi phân. Vì vậy, phương pháp Adam sẽ được sử dụng để tối ưu hàm tổn thất (3.33) với 10000 vòng lặp. Tuy số vòng lặp lớn hơn nhiều so với với khi huấn luyện mạng NeuralODE nhiều bước (10000 » 1500), thời gian để tối ưu hàm tổn thất (3.33) chỉ bằng một nửa so với thời gian huấn luyện phương pháp NeuralODE. Để so sánh độ hiệu quả, phương pháp NeuralODE nhiều bước cũng được sử dụng để huấn luyện với bộ dữ liệu quỹ đạo gốc $\mathcal{D}^{(traj)}$ và $\mathcal{G}^{(traj)}$. Đối với cả hai phương pháp, kiến trúc mạng neuron trường vector được cho trong bảng 3.1. Phương pháp NeuralODE nhiều bước sẽ được huấn luyện bằng phương pháp Adam với 1500 vòng lặp. Kết quả huấn luyện được cho trong bảng 3.7, 3.8 và hình 3.19.

Giá trị tối ưu hàm MSE	**Quỹ đạo gốc**	**Quỹ đạo liên hợp**
Quỹ đạo 0	0.0451	0.0265
Quỹ đạo 1	0.0447	0.0092
Quỹ đạo 2	0.0604	0.0277
Quỹ đạo 3	0.0512	0.0211
Quỹ đạo 4	0.0090	0.0696
Quỹ đạo 5	0.0447	0.0538
Quỹ đạo 6	0.0843	0.0337
Quỹ đạo 7	0.0278	0.0077
Quỹ đạo 8	0.0678	0.0358
Quỹ đạo 9	0.0338	0.0263

Bảng 3.6: Kết quả so sánh giá trị tối ưu hàm tổn thất trên bộ dữ liệu $\mathcal{G}^{(traj)}$ và $\mathcal{G}^{(trans)}$.

Từ bảng 3.7 và bảng 3.8, sai số trên bộ dữ liệu huấn luyện $\mathcal{D}^{(traj)}$ và $\mathcal{G}^{(traj)}$ đều được tối ưu tương đối nhỏ đối với cả hai phương pháp NeuralODE nhiều bước và phương pháp đề xuất. Tuy vậy, hình 3.19 cho thấy rằng phương pháp NeuralODE nhiều bước gặp khó khi học các điểm nằm ở vùng đa tạp không ổn định. Ngược lại, phương pháp đề xuất có thể học những điểm ở vùng đa tạp không ổn định hiệu quả hơn.

Kiểm thử trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}$ và $\mathcal{G}_t^{(traj)}$

Mục này sẽ so sánh phương pháp đề xuất, phương pháp NeuralODE với bộ huấn luyện nhiều quỹ đạo (ký hiệu $\mathrm{NeuralODEtraj}$) và phương pháp NeuralODE với bộ huấn luyện đều (ký hiệu $\mathrm{NeuralODEuni}$) trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}$ và $\mathcal{G}_t^{(traj)}$. Kết quả được cho trong hình 3.20. Hình 3.20 cho thấy biểu đồ hộp sai số của các phương pháp trên bộ $\mathcal{M}_t^{(traj)}(\mathcal{M} \in \mathcal{D}, \mathcal{G})$. Nhìn chung, sai số của phương pháp đề xuất thấp hơn so với sai số của phương pháp $\mathrm{NeuralODEtraj}$ và ngang bằng phương pháp $\mathrm{NeuralODEuni}$ trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}$ và $\mathcal{G}_t^{(traj)}$. Ngoài ra, độ ổn định của phương pháp đề xuất cũng tốt hơn so với phương pháp $\mathrm{NeuralODEtraj}$ khi khoảng sai số bé và số lượng

Sai số MSE	**NeuralODE nhiều bước**	**Phương pháp đề xuất**
Quỹ đạo 0	0.0510	0.0607
Quỹ đạo 1	0.2483	0.1809
Quỹ đạo 2	0.3588	0.1803
Quỹ đạo 3	0.0116	0.0771
Quỹ đạo 4	0.0388	0.1750
Quỹ đạo 5	0.0467	0.0666
Quỹ đạo 6	0.0549	0.0604
Quỹ đạo 7	0.0233	0.0211
Quỹ đạo 8	0.0375	0.1931
Quỹ đạo 9	0.2677	0.2230

Bảng 3.7: Kết quả so sánh giá trị tối ưu hàm tổn thất trên bộ dữ liệu $\mathcal{D}^{(traj)}$.

các sai số lớn cũng ít hơn. Hình 3.20 cũng cho so sánh sai số của các phương pháp trên từng quỹ đạo trong bộ dữ liệu kiểm tra. Có thể thấy, số lượng các quỹ đạo phương pháp đề xuất cho sai số ít hơn phương pháp NeuralODEuni là chiếm phần lớn. Cụ thể, đối với bộ kiểm tra $\mathcal{D}_t^{(traj)}$, phương pháp đề xuất dự đoán chính xác hơn 65 trên tổng số 100 quỹ đạo so với phương pháp NeuralODE nhiều bước. Đối với bộ kiểm tra $\mathcal{G}_t^{(traj)}$, phương pháp đề xuất dự đoán chính xác hơn 83 trên tổng số 100 quỹ đạo so với phương pháp NeuralODE nhiều bước. Điều này phản ánh tính tổng quát hóa của phương pháp đề xuất tốt hơn so với phương pháp NeuralODE nhiều bước.

Tính ổn định của phương pháp đề xuất

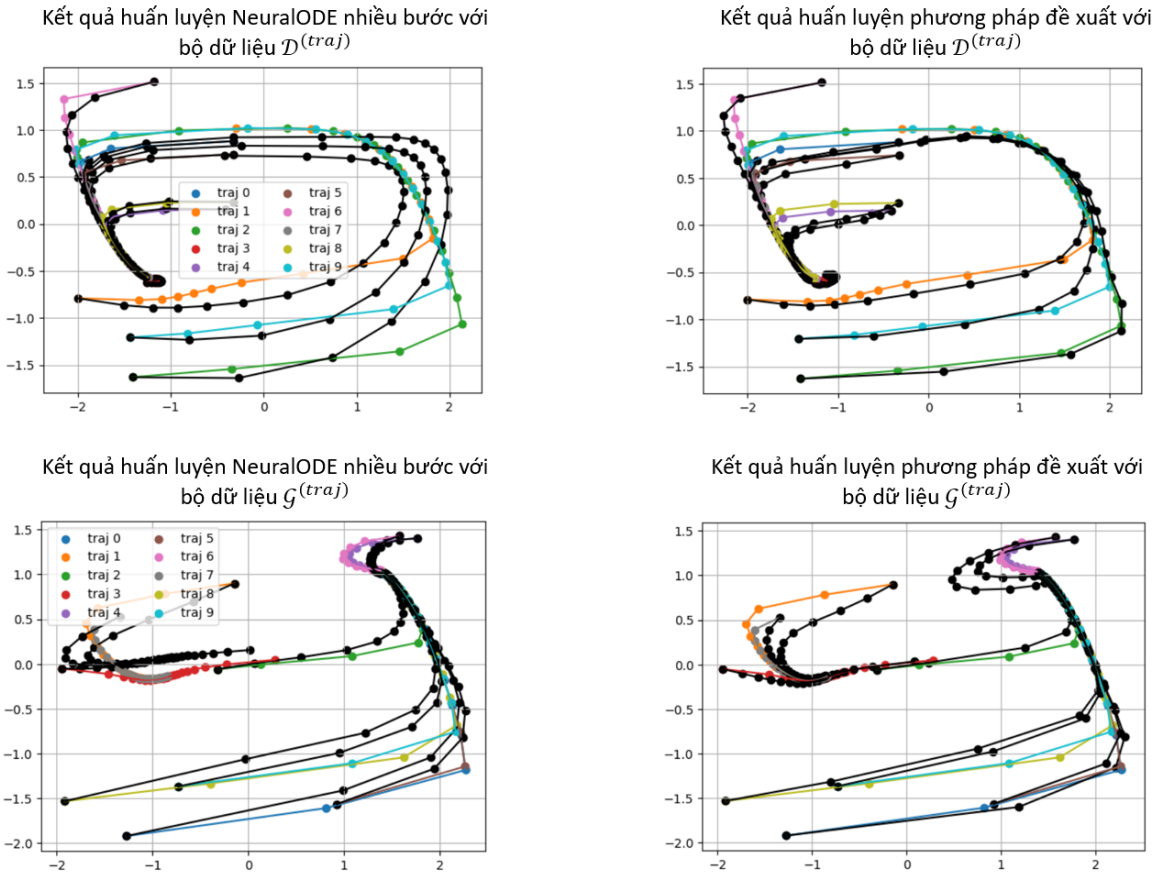
Mục tiêu cuối cùng của phương pháp đề xuất là học trường vector của hệ động lực. Trường vector của hệ động lực sẽ được xấp xỉ bởi một mạng neuron MLP với các tham số tối ưu được học thông qua bộ dữ liệu huấn luyện. Thiết kế kiến trúc mạng neuron MLP cho việc xấp xỉ trường vector là một điều không đơn giản. Hiện nay theo hiểu biết của chúng tôi chưa có phương pháp thống nhất nào cho việc thiết kế kiến trúc mạng neuron hiệu quả. Việc thiết kế mạng

Sai số MSE	**NeuralODE nhiều bước**	**Phương pháp đề xuất**
Quỹ đạo 0	0.0653	0.0462
Quỹ đạo 1	0.2050	0.1707
Quỹ đạo 2	0.1517	0.0584
Quỹ đạo 3	0.2078	0.1360
Quỹ đạo 4	0.0999	0.0994
Quỹ đạo 5	0.0641	0.0265
Quỹ đạo 6	0.0950	0.2048
Quỹ đạo 7	0.2298	0.0308
Quỹ đạo 8	0.1081	0.1127
Quỹ đạo 9	0.0564	0.0651

Bảng 3.8: Kết quả so sánh giá trị tối ưu hàm tổn thất trên bộ dữ liệu $\mathcal{G}^{(traj)}$.

neuron không phù hợp có thể ảnh hưởng xấu đến kết quả của toàn bộ mô hình. Trong mục này, ta sẽ xem xét tính ổn định của phương pháp đề xuất khi thay đổi cấu trúc của mạng neuron MLP $f_{\theta,\mathcal{M}}$ xấp xỉ trường vector hệ động lực.

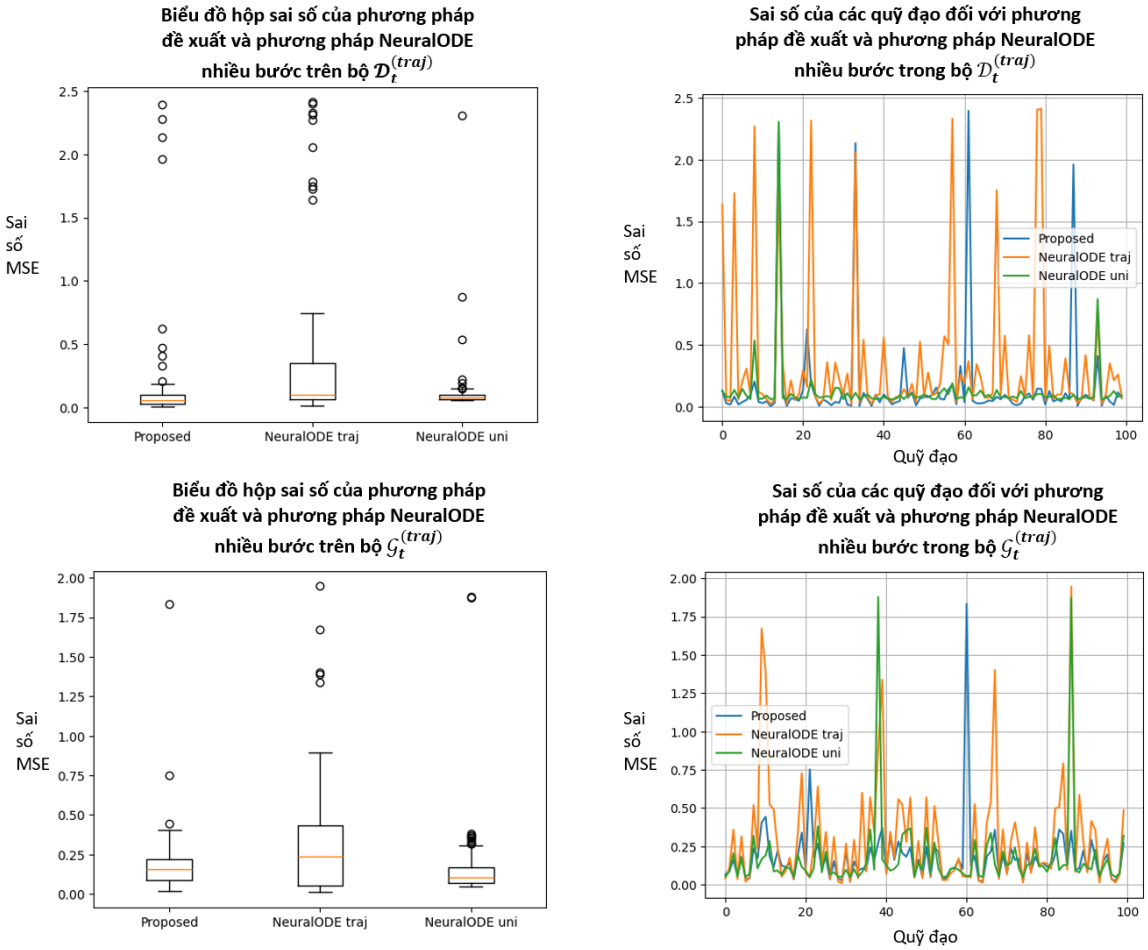
Thay đổi số nốt lớp ẩn Kiến trúc mạng neuron MLP sẽ được cho tương tự như trong bảng 3.1 ngoại trừ việc số nốt của lớp ẩn sẽ thay đổi. Số nốt của lớp ẩn sẽ thay đổi trong tập $[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$. Ta giữ nguyên bộ tham số tối ưu ϕ_i^* của phép chuẩn hóa liên hợp, bộ tham số tối ưu φ_i^* của hệ động lực liên hợp. Ứng với mỗi trường hợp của số nốt lớp ẩn, phương pháp Adam sẽ được sử dụng để tối ưu hàm tổn thất (3.33) với 10000 vòng lặp. Kết quả tối ưu cuối cùng được kiểm thử trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$. Kết quả được cho trong hình 3.21. Nhìn vào hình 3.21, kết quả của phương pháp đề xuất trên bộ dữ liệu $\mathcal{M}_t^{(traj)}$ khi thay đổi số nốt của lớp ẩn thay đổi không đáng kể và đều có sai số thấp hơn so với phương pháp NeuralODE nhiều bước. Sai số MSE có xu hướng giảm dần khi số nốt của lớp ẩn tăng. Điều này có thể lí giải rằng mạng neuron MLP phức tạp hơn sẽ xấp xỉ tốt hơn trường vector của hệ động lực. Tuy nhiên, cũng có thể dễ dàng thấy



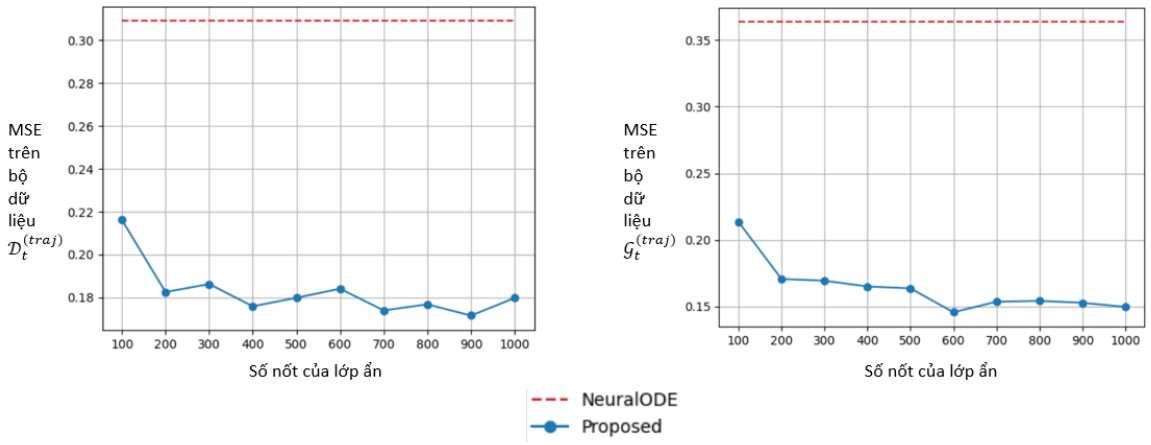
Hình 3.19: (**Trái trên**) Kết quả huấn luyện phương pháp NeuralODE trên bộ dữ liệu $\mathcal{D}^{(traj)}$; **(Phải trên)** Kết quả huấn luyện phương pháp đề xuất trên bộ dữ liệu $\mathcal{D}^{(traj)}$; (**Trái dưới**) Kết quả huấn luyện phương pháp NeuralODE trên bộ dữ liệu $\mathcal{G}^{(traj)}$; **(Phải dưới)** Kết quả huấn luyện phương pháp đề xuất trên bộ dữ liệu $\mathcal{G}^{(traj)}$. Trong tất cả các hình, quỹ đạo màu đen là quỹ đạo dự đoán.

sai số MSE không giảm mạnh khi số nốt của lớp ẩn cao (800,900,1000). Điều này có thể lí giải rằng khi kiến trúc mạng neuron MLP phức tạp, ta cần sử dụng nhiều vòng lặp của thuật toán tối ưu Adam hơn để tìm tham số tối ưu cho mô hình. Việc sử dụng cùng số vòng lặp sẽ khiến cho các mô hình mạng neuron phức tạp chưa tìm được nghiệm đủ tốt. Chung quy lại, phương phấp đề xuất không bị ảnh hưởng quá nhiều khi ta thay đổi số nốt lớp ẩn mạng neuron xấp xỉ trường vector.

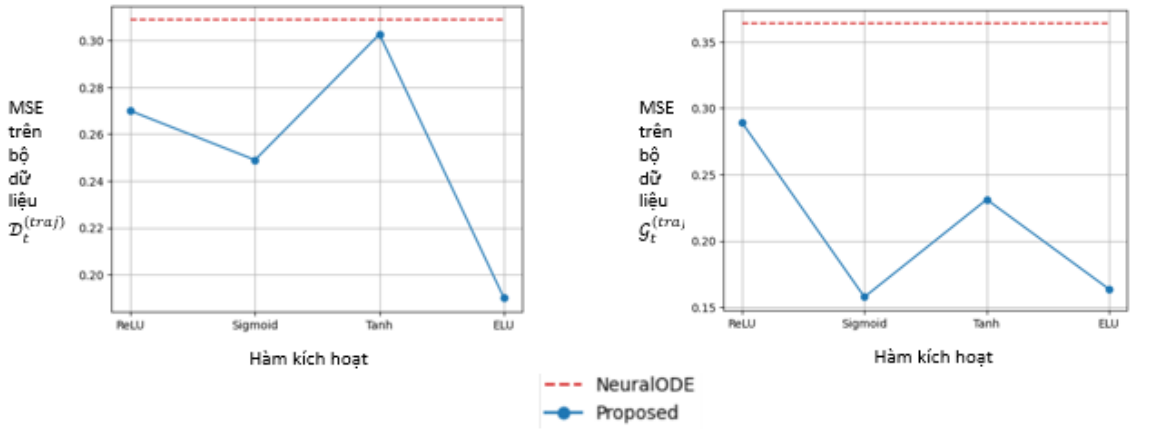
Thay đổi hàm kích hoạt Kiến trúc mạng neuron MLP sẽ được cho tương tự như trong bảng 3.1 ngoại trừ việc loại hàm kích hoạt sẽ bị thay đổi. Hàm kích hoạt



Hình 3.20: (**Trái trên**) Biểu đồ hộp sai số của phương pháp đề xuất, phương pháp NeuralODE với bộ huấn luyện quỹ đạo, và phương pháp NeuralODE với bộ huấn luyện dữ liệu đều trên bộ $\mathcal{D}_t^{(traj)}$; **(Phải trên)** Sai số của các quỹ đạo đối với phương pháp đề xuất, phương pháp NeuralODE với bộ huấn luyện dữ liệu quỹ đạo, và phương pháp NeuralODE với bộ huấn luyện dữ liệu đều trên bộ $\mathcal{D}^{(traj)}$; (**Trái dưới**) Biểu đồ hộp sai số của phương pháp đề xuất, phương pháp NeuralODE với bộ huấn luyện quỹ đạo, và phương pháp NeuralODE với bộ huấn luyện dữ liệu đều trên bộ $\mathcal{G}_t^{(traj)}$; **(Phải dưới)** Sai số của các quỹ đạo đối với phương pháp đề xuất, phương pháp NeuralODE với bộ huấn luyện dữ liệu quỹ đạo, và phương pháp NeuralODE với bộ huấn luyện dữ liệu đều trên bộ $\mathcal{G}^{(traj)}$.



Hình 3.21: Sai số MSE của phương pháp đề xuất khi thay đổi số nôt lớp ẩn mạng neuron trường vector $f_{\theta,\mathcal{M}}$. (**Trái**) Kết quả trên bộ kiểm tra $\mathcal{D}_t^{(traj)}$; (**Phải**) Kết quả trên bộ kiểm tra $\mathcal{G}_t^{(traj)}$. Đường màu đỏ thể hiện sai số MSE của phương pháp NeuralODE nhiều bước trên bộ dữ liệu $\mathcal{M}_t^{(traj)}$; đường màu xanh thể hiện sai số của MSE của phương pháp đề xuất khi thay đổi số nốt của lớp ẩn.



Hình 3.22: Sai số MSE của phương pháp đề xuất khi thay đổi hàm kích hoạt mạng neuron trường vector $f_{\theta,\mathcal{M}}$. (**Trái**) Kết quả trên bộ kiểm tra $\mathcal{D}_t^{(traj)}$; (**Phải**) Kết quả trên bộ kiểm tra $\mathcal{G}_t^{(traj)}$. Đường màu đỏ thể hiện sai số MSE của phương pháp NeuralODE nhiều bước trên bộ dữ liệu $\mathcal{M}_t^{(traj)}$; đường màu xanh thể hiện sai số của MSE của phương pháp đề xuất khi thay đổi số nốt của lớp ẩn.

được thay đổi về các dạng ELU, ReLU, Sigmoid và Tanh như được chỉ ra trong mục 1.2.1.. Ta giữ nguyên bộ tham số tối ưu ϕ_i^* của phép chuẩn hóa liên hợp, bộ tham số tối ưu φ_i^* của hệ động lực liên hợp. Ứng với mỗi trường hợp của số nốt lớp ẩn, phương pháp Adam sẽ được sử dụng để tối ưu hàm tổn thất (3.33) với 10000 vòng lặp. Kết quả tối ưu cuối cùng được kiểm thử trên bộ dữ liệu kiểm tra $\mathcal{D}_t^{(traj)}(\mathcal{M} \in \{\mathcal{D}, \mathcal{G}\})$. Kết quả được cho trong hình 3.22. Nhìn vào hình 3.22, kết quả của phương pháp đề xuất trên bộ dữ liệu $\mathcal{M}_t^{(traj)}$ khi thay đổi hàm kích hoạt thay đổi không đáng kể và đều có sai số thấp hơn so với phương pháp NeuralODE nhiều bước. Chung quy lại, phương phấp đề xuất không bị ảnh hưởng quá nhiều khi ta thay đổi hàm kích hoạt của mạng neuron xấp xỉ trường vector.

Chương 4

Kết luận và kiến nghị

Kết luận

Luận văn thử nghiệm phương pháp mạng neuron MLP và phương pháp NeuralODE cho việc học hệ động lực tất định có tập hút phức tạp, ở đó trường vector tại mỗi điểm có sự thay đổi giữa các vùng trong không gian trạng thái. Các kết quả tính toán cho thấy với bộ dữ liệu huấn luyện sinh đều ngẫu nhiên, cả hai phương pháp mạng neuron MLP và NeuralODE đều cho kết quả tốt. Tuy vậy, khi bộ dữ liệu được sinh từ các quỹ đạo của hệ động lực tất định, cả hai phương pháp đều không học hiệu quả do tính nhiễu của bộ dữ liệu. Từ đó, bài báo cáo đề xuất một phương pháp học hệ động lực với bộ dữ liệu sinh từ các quỹ đạo thông qua phép biến đổi liên hợp. Cụ thể, phép biến đổi liên hợp sẽ biến đổi các quỹ đạo gốc thành các quỹ đạo liên hợp dễ học hơn. Sau khi đã học quỹ đạo liên hợp, trường vector gốc sẽ được học dựa trên phép biến đổi liên hợp và hệ động lực liên hợp. Kết quả thử nghiệm cho thấy phương pháp đề xuất cho kết quả tốt hơn phương pháp NeuralODE.

Kiến nghị

Trong tương lai, tác giả sẽ thử nghiệm phương pháp đề xuất với nhiều hệ động lực có tính chất phức tạp hơn như hệ Lorenz. Nhược điểm của phương

pháp biến đổi liên hợp là nó có thể rất nhạy cảm với dữ liệu đầu vào, do đó khi dữ liệu có nhiễu sẽ có thể dẫn đến các vấn đề phức tạp của việc học autonomous. Do giới hạn về thời gian, luận văn cũng chưa tập trung nghiên cứu tính ổn định của phương pháp biến đổi liên hợp theo dữ liệu đầu vào. Chúng tôi dự kiến sẽ nghiên cứu mở rộng phương pháp này cho các phương trình vi phân ngẫu nhiên và phương pháp học hệ động lực ngẫu nhiên trong tương lai để xử lý những nhược điểm này. Ngoài ra, việc cải thiện tốc độ huấn luyện của phương pháp đề xuất cũng sẽ được xem xét và nghiên cứu trong các công trình tương lai.

Tài liệu tham khảo

[1] Richard Brown. *A Modern Introduction to Dynamical Systems*. 06 2018.

[2] David F. Griffiths and Desmond J. Higham. *Euler's Method*, pages 19–31. Springer London, London, 2010.

[3] William Erik Sherwood. *FitzHugh–Nagumo Model*, pages 1–11. Springer New York, New York, NY, 2013.

[4] Wikipedia contributors. Fitzhugh–nagumo model — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=FitzHugh%E2%80%93Nagumo_model&oldid=1224775541, 2024. [Online; accessed 13-October-2024].

[5] Tali Pinsky. Analytical study of the lorenz system: Existence of infinitely many periodic orbits and their topological characterization. *Proceedings of the National Academy of Sciences*, 120(31):e2205552120, 2023.

[6] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning. *CoRR*, abs/2301.05579, 2023.

[7] Midhun T. Augustine. A survey on universal approximation theorems. *CoRR*, abs/2407.12895, 2024.

[8] Ken-ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[9] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6231–6239, 2017.

[10] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583, 2018.

[11] J. Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11126–11134, 2019.

[12] Shuangshuang Chen, Sihao Ding, Yiannis Karayiannidis, and Mårten Björkman. Learning continuous normalizing flows for faster convergence to target distribution via ascent regularizations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[13] Ivan Dario Jimenez Rodriguez, Aaron D. Ames, and Yisong Yue. Lyanet: A lyapunov framework for training neural odes. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato,

editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 18687–18703. PMLR, 2022.

[14] Nathan P. Lawrence, Philip D. Loewen, Michael G. Forbes, Johan U. Backström, and R. Bhushan Gopaluni. Almost surely stable deep dynamics. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[15] Wenxin Xiao, Armin Lederer, and Sandra Hirche. Learning stable non-parametric dynamical systems with gaussian process regression. *CoRR*, abs/2006.07868, 2020.

[16] Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Nadia Figueroa, Gerhard Neumann, and Leonel Rozo. Neural contractive dynamical systems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[17] Bernardo Fichera and Aude Billard. Linearization and identification of multiple-attractor dynamical systems through laplacian eigenmaps. *J. Mach. Learn. Res.*, 23:294:1–294:35, 2022.

[18] Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan D. Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In Alexandre M. Bayen, Ali Jadbabaie, George J. Pappas, Pablo A. Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 2nd Annual Conference on Learning for Dynamics and Control, L4DC 2020, Online Event, Berkeley, CA, USA, 11-12 June 2020*, volume 120 of *Proceedings of Machine Learning Research*, pages 630–639. PMLR, 2020.

[19] Umut Simsekli, Levent Sagun, and Mert Gürbüzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. *ArXiv*, abs/1901.06053, 2019.

[20] Ruizhi Deng, Marcus A. Brubaker, Greg Mori, and Andreas M. Lehrmann. Continuous latent process flows. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 5162–5173, 2021.

[21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[22] Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Neural sdes as infinite-dimensional gans. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5453–5463. PMLR, 2021.

[23] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

[24] Felix Dietrich, Alexei Makeev, George A. Kevrekidis, Nikolaos Evangelou, Tom S. Bertalan, Sebastian Reich, and Ioannis G. Kevrekidis. Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning. *Chaos*, 33 2:023121, 2021.

[25] Yuanyuan Wang, Xi Geng, Wei Huang, Biwei Huang, and Mingming Gong. Generator identification for linear sdes with additive and multiplicative noise. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[26] Dang Nguyen, Paymon Haddad, Eric Gan, and Baharan Mirzasoleiman. Make the most of your data: Changing the training data distribution to improve in-distribution generalization performance. *CoRR*, abs/2404.17768, 2024.

[27] Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. Towards a theoretical framework of out-of-distribution generalization. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23519–23531, 2021.

[28] George D. Birkhoff. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.

[29] Dandan Guo, Zhuo Li, Meixi Zheng, He Zhao, Mingyuan Zhou, and Hongyuan Zha. Learning to re-weight examples with optimal transport for imbalanced classification. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[30] Yuqi Liu, Bin Cao, and Jing Fan. Improving the accuracy of learning example weights for imbalance classification. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.