

Autoencoder e Sistemi Dinamici

Pietro Sillano

December 21, 2021

1 Introduzione/Abstract

We present a method for the simultaneous discovery of sparse dynamical models and coordinates that enable these simple representations.

L'idea del paper considerato consiste nello sviluppare un approccio che contemporaneamente individui un modello per un sistema dinamico e delle coordinate rappresentative del sistema. Il primo obiettivo viene raggiunto attraverso una sparse regression in una libreria di funzioni candidate mentre il secondo viene raggiunto tramite l'utilizzo di un autoencoder tramite una riduzione di dimensionalità del sistema.

2 Obiettivi/Problem,question,task

Risulta opportuna l'unione di questi due metodi in quanto non sempre i dati che vengono raccolti rappresentano la dinamica del sistema nella migliore e nella più semplice rappresentazione possibile. Mentre gli autoencoder possono essere allenati individualmente per effettuare riduzioni di dimensionalità non c'è garanzia che le coordinate intrinseche (nel latent space) individuate avranno dei modelli dinamici associati semplici e sparsi.

3 Metodi

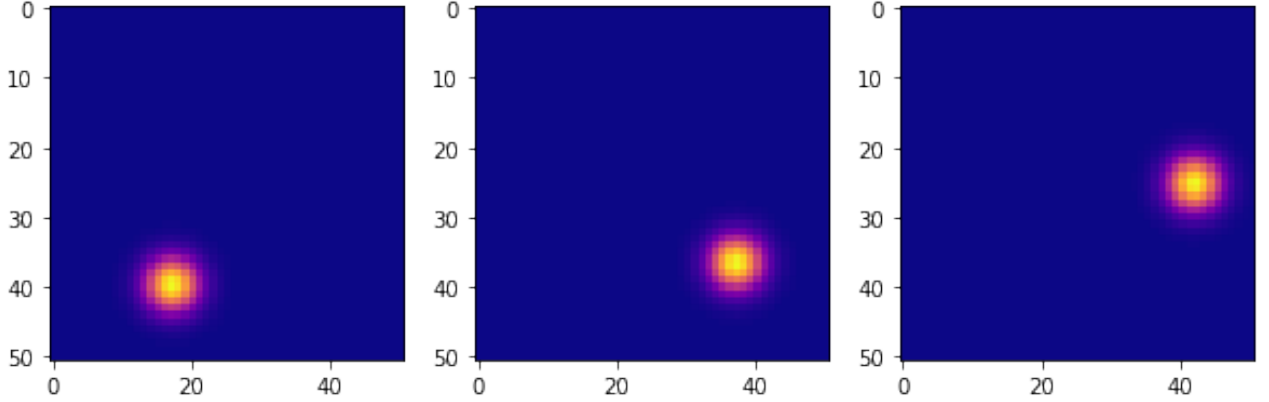
3.1 Dataset

Un punto a favore di questo approccio è la grande quantità di samples che è possibile generare sinteticamente e la libertà con cui è possibile manipolare e condizionare il dataset; per esempio il numero di condizioni iniziali del sistema dinamico che stiamo considerando, la possibilità di poter studiare segnali temporali con un noise inserito ad hoc o l'intervallo temporale con cui integriamo le equazioni del moto.

3.1.1 Lorentz

3.1.2 Pendolo

In questo caso i samples del dataset considerato sono snapshot(istantanee) di un video del moto di un pendolo. Snapshots of the high-dimensional system are images representing the position of the pendulum in time.



Ricordando l'equazione del moto di un pendolo:

$$\ddot{\theta} = -\sin(\theta)$$

che posso riscrivere come un sistema di due ODE del primo ordine, ossia:

$$\begin{cases} \dot{\theta} = \omega \\ \dot{\omega} = -\sin(\theta) \end{cases}$$

con condizioni iniziali θ_0 e ω_0 scelte arbitrariamente.

Ho integrato numericamente questo sistema con un integratore RK4 (Runge-Kutta del 4° ordine) ottenendo l'andamento di θ e ω nel tempo.

Successivamente per generare il video del pendolo ho generato una funzione 2d della forma di una gaussiana dipendente da $\theta(t)$:

$$G(x, y, \theta(t)) = e^{-A[(x - \cos \theta(t))^2 + (y - \sin \theta(t))^2]}$$

dove A é una costante che determina l'ampiezza della gaussiana, mentre x e y sono punti di una griglia $NX \times NY$ in cui ho valutato la funzione in ogni punto ottenendo un immagine di questo tipo:

In questo modo é possibile generare molti samples. É importante però variare le condizioni iniziali in modo che la rete possa avere samples del sistema in ogni punto della dinamica.

Per generare il dataset sono state usate 100 condizioni iniziali per θ_0 e 100 per ω_0 combinandole assieme e escludendo le condizioni iniziali per cui il pendolo non ha un attrattore "chiuso" o con un ciclo limite. Complessivamente quindi rimangono circa 5000 condizioni iniziali differenti.

3.2 Architettura della rete

L'autoencoder fa our method seeks a set of reduced coordinates $z(t) = \phi(x(t)) \in \mathbb{R}^d$

The coordinate transformation is achieved using an autoencoder network architecture. The autoencoder is a feedforward neural network with a hidden layer that represents the intrinsic coordinates. the autoencoder learns a nonlinear embedding into a reduced latent space. Our network takes measurement data $x(t) \in \mathbb{R}^n$ from a dynamical system as input and learns intrinsic coordinates $z(t) \in \mathbb{R}^d$, where $d \times n$ is chosen as a hyperparameter prior to training the network. The autoencoder network consists of a series of fully-connected layers. é semplice sono 3/4 layer fully connected

L'architettura presentata nel paper é stata riscritta in PyTorch.

3.3 Sindy

La libreria delle funzioni scelte é usata per formulare un problema di regressione per risolvere in modo approssimato il sistema lineare di equazioni sovradeterminato(abbiamo piú equazioni che incognite).

$$\dot{X} = \Theta(X)\Xi$$

dove X é una matrice $(t \times n)$ dove t é il numero di samples temporali e n é il numero di variabili del sistema dinamico contenente gli stati del sistema, \dot{X} di uguale dimensione a X ma contenente i time derivatives degli stati del sistema. La matrice Θ é nota e costituisce la nostra libreria di funzioni con cui vogliamo fittare il sistema dinamico. é $(t \times \text{num di funzioni})$

La matrice incognita $\Xi = (\xi_1, \xi_2, \dots, \xi_n)$ é $(\text{num funzioni} \times \text{numero di variabili del modello})$ contiene i coefficienti che determinano i termini attivi della matrice theta. Noi vorremmo che questa matrice sia sparsa in modo da avere pochi termini selezionati dalla theta.

3.4 Loss

Un punto principale di questa architettura é come vengono definite e usate le funzioni di costo della rete. Infatti la loss non é semplicemente la MSE loss (mean squared loss) tra l'input x dell' autoencoder e la ricostruzione \tilde{x} in output. Vengono aggiunti alcuni termini che garantiscono oltre alla ricostruzione dell'input, anche la ricostruzione delle derivate temporali \dot{x} e \dot{z} e infine é presente anche un termine di regolarizzazione L1 sui coefficienti Ξ in modo da promuovere la sparsity di quest'ultimi.

Per un dataset con m samples di input, ogni loss é esplicitamente definita come segue:

- $L_{recon} = \frac{1}{m} \sum_{i=1}^m \|x_i - \tilde{x}_i\|_2^2$ questa misura quanto bene l'autoencoder riesce a ricostruire l'input x .
- $L_{dz} = \frac{1}{m} \sum_{i=1}^m \|\dot{z}_i - \Theta(z_i)\Xi\|_2^2$ Questo termine misura quanto il modello predica correttamente le derivate temporale delle variabili ridotte z
- $L_{dx} = \frac{1}{m} \sum_{i=1}^m \|\dot{x}_i - \hat{\dot{x}}_i\|_2^2$ misura quanto le predizioni di Sindy ossia $\hat{\dot{x}}$ possano essere usate per ricostruire l'input originale \dot{x}

- $L_{reg} = \frac{1}{m} \sum_{i=1}^m \|\Xi\|_1$ é una regolarizzazione L1 che promuove la sparsity dei coefficienti Ξ che coincide con i nostri obiettivi perché noi vorremmo scoprire modelli parsimoniosi ossia con il minor numero di parametri possibile.

Per cui combinando insieme i quattro termini di loss insieme ai relativi iperparametri la loss finale é la seguente:

$$L_{tot} = L_{recon} + \lambda_1 L_{dx} + \lambda_2 L_{dz} + \lambda_3 L_{reg}$$

In queste loss compaiono delle derivate temporali di alcune grandezze: \dot{x} assumiamo di poterla calcolare a partire dai dati di input mentre \dot{z} e $\dot{\hat{x}}$ non possiamo calcolarle tramite backpropagation della rete. Dobbiamo quindi trovare un metodo per esprimere tali grandezze in funzione delle grandezze note.

x = input, y = output della rete, $z = w^\top x + b$, $f(z)$ funzione di attivazione differenziabile, $a_l(z) = f(z_l)$ attivazione del layer l , l = layer l della rete e m = numero di layer nella rete

$$\frac{dy}{dt} = \frac{dy}{dx} \frac{dx}{dt} = \frac{dx}{dt} \prod_{l=0}^{m-1} \frac{da_l}{dx} = \frac{dx}{dt} \prod_{l=0}^{m-1} \frac{df(z_l)}{dz_l} = \dot{x} \prod_{l=0}^{m-1} \frac{df(z_l)}{dz_l} \frac{dz_l}{dx} = \dot{x} \prod_{l=0}^{m-1} \frac{df(z_l)}{dz_l} w_l^\top$$

Applicando questa regola con $y = \dot{z}$ é possibile calcolarla a partire da \dot{x} e allo stesso modo se $y = \dot{\hat{x}}$ a partire da $\dot{\hat{z}}$ quest ultima ricavata con Sindy $\dot{\hat{z}} = \theta(z)\xi$

3.5 Dettagli Training

Batch size = 1024 Funzione di attivazione f = ReLU o Sigmoide optimizer = adam o sgd weight initializations (sottolineare quanto sia importante per un training ragionevole)

4 Risultati

plot con le loss standard

plot con le traiettorie predette a confronto con quelle originali

parlare della difficoltà della model selection in quanto non é una semplice misura o valutazione della loss ma una considerazione della parsimoniosità e della correttezza dei coefficienti del modello dinamico trovato

5 Conclusioni e possibili sviluppi futuri

Estendere ad altri sistemi dinamici, modificare e rendere piu deep l architettura, introdurre un sistema di time derivatives al secondo ordine, quindi provare a ricostruire \ddot{x} da x .

References

- [1] PyTorch documentation. <https://pytorch.org/docs/stable/index.html>.
- [2] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. *Data-driven discovery of coordinates and governing equations*. 2019.

6 Codice

Implementazione del codice su <https://github.com/pietro-sillano/SindyPendulum>.