

# 非线性动力学课程报告

陈长 3120104099

## Abstract

本文的主题为神经网络动力学与人工智能。作者首先研究了 Hopfield 神经网络的动力学行为，随后对于蕴含着复杂动力学行为的实时电价给出了使用循环神经网络的预测方法。

*Keywords:* 神经网络, 动力学, 混沌, RNN, GRU

## 1. Hopfield 神经网络动力学研究

### 1.1. 神经网络介绍

人工神经网络是由大量简单的基本单元（即神经元）相互连接而成的系统。神经元在输入信号作用下产生输出信号的规律由神经元功能函数  $f$  给出，这是神经元模型的外特性。它包括了从输入信号到净输入、再到激活值、最终产生输出信号的过程。通常  $f$  函数可划分为 3 种类型：简单的映射关系、动态系统、概率统计模型。对于简单的映射关系模型，各神经元构成的输出向量  $Y$  与输入向量  $X$  符合某种映射关系，不考虑神经元的时间滞后效应。例如：

$$Y = f(WX - \theta) \quad (1)$$

其中  $W$  是连接权值矩阵， $\theta$  是阈值向量， $f$  是非线性激活函数。因此：人工神经网络是一个高度互联的复杂的非线性系统，其中每个神经元的输入可以与许多其他神经元相连，但只有一个输出。普通神经网络没有惯性环节，因此是一个纯代数系统，其数学模型采用非线性代数方程描述。而如下介绍的 Hopfield 神经网络具有惯性环节，因此它的数学模型是一个非线性动力学系统。

### 1.2. Hopfield 神经网络模型的建立

Hopfield 神经网络的电路模型如图1所示，第  $i$  个放大器输入端并联的电容  $C_i$  与电阻  $R_i$  模拟神经元的时间常数，在输出与输入之间产生延迟作用，构成神经元的动态特性。通过推导可以得出该电路的数学模型如下：

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_j \omega_{ij} V_j + I_i \quad (2)$$

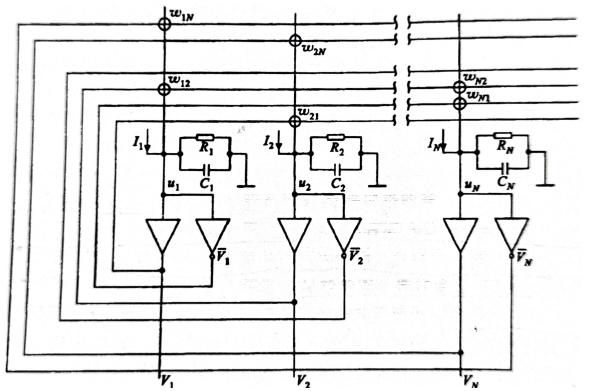


图 1: Hopfield 神经网络的电路模型

$$V_i = f_i(u_i) \quad (3)$$

其中  $i = 1, 2, \dots, N$ 。非线性函数  $f$  采用具有正、反相输出的放大器进行模拟。表达式为：

$$V_i = f_i(u_i) = \frac{1}{2}[1 + \tanh(\frac{u_i}{u_o})] \quad (4)$$

### 1.3. 三阶非线性动力学模型研究

为研究 Hopfield 神经网络，可以首先以非线性动力学为例展开研究：

$$\frac{d\vec{x}}{dt} = A\vec{x} + Bf(\vec{x}) + I \quad (5)$$

其中  $A$  为 3 阶对角矩阵， $B$  为  $3 \times 3$  系数矩阵， $I$  为  $3 \times 1$  系数向量。其中参数按如下给出：

$$A = diag[-1, -1, -1] \quad (6)$$

$$B = \begin{bmatrix} 3.8 & -1.9 & 0.7 \\ 2.5 & 0.06 & 1 \\ -6.6 & 1.3 & 0.07 \end{bmatrix} \quad (7)$$

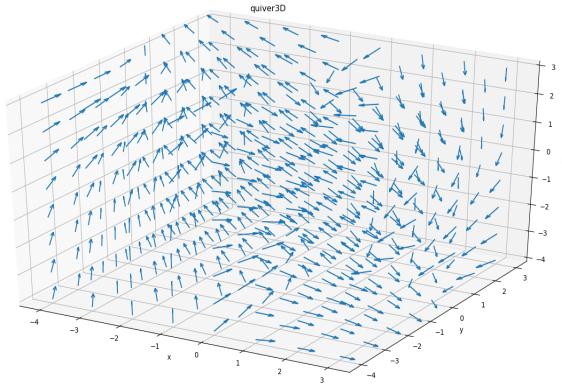


图 2: 三维速度向量场

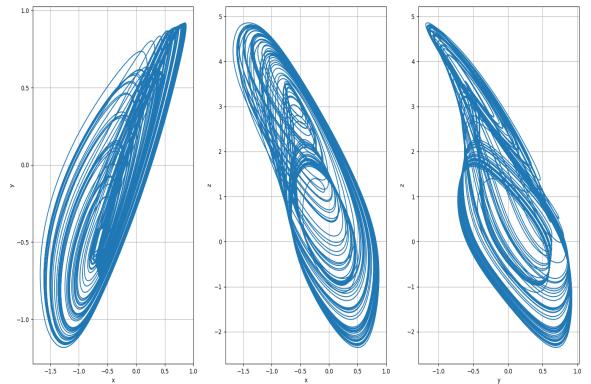


图 4: 相轨迹的投影图

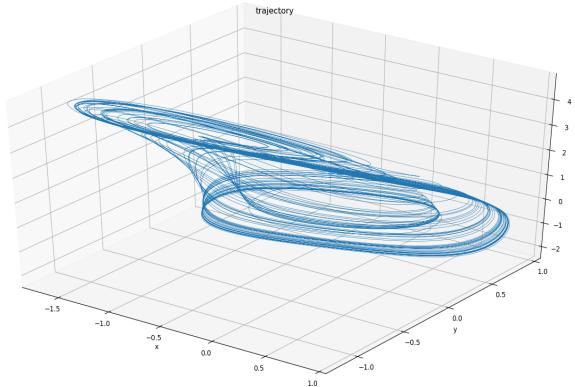


图 3: 三维相轨迹图

$$I = [0, 0, 0]^T \quad (8)$$

$$f(x) = \tanh(x) \quad (9)$$

### 1.3.1. 平衡点

平衡点即令  $\frac{dx}{dt} = 0$  的点  $\vec{x}$ 。求解平衡点的问题本质是求解一个非线性代数方程组，可以采用牛顿法等方法加以求解。编写程序加以求解。其中非线性方程组求解器使用 `scipy.optimize` 第三方库中的 `root` 函数。在给定一些随机的初始点之后，经计算可以得到两个收敛解： $\vec{x}_1 = [0.6593352, 0.47394677, -3.31029359]$ ， $\vec{x}_2 = [-0.6593352 - 0.47394677, 3.31029359]$ 。另外，显然  $\vec{x}_0 = [0, 0, 0]$  是一个平凡解。

### 1.3.2. 速度向量场

通过在三维坐标图里每一点处取它的速度矢量，有助于观察相空间的动力学特征。编写程序绘制三维速度向量场如图2。

### 1.3.3. 相轨迹图

通过给定一个初始点  $[5, -2, 5]$ ，在  $t \in (0, 700)$  区间内，积分步长为 0.001，积分器选择 `scipy.integrate` 库中的 `solve_ivp` 函数。选取后 80% 的点作为稳态点绘制空间相轨迹图。编写程序得到结果如图3。

作者同样为该混沌吸引子绘制了动态形成图（见附件），以便观察轨迹走向。其中在三个坐标平面的投影图如图4。从结果可知：它们具有典型的奇异吸引子的特征。

### 1.3.4. 时域波形图

通过在 Simulink 中搭建仿真模型得到波形图5，其中仿真时间为 7000s，最终绘图使用的时间段取最后 200s。对于该仿真模型，取同样参数编写程序得到波形图6，其中微分方程求解器的选择与绘制相轨迹时一致。从两者对比的角度来看，均呈现典型的混沌的时域波形图，没有一个明确的周期，两者波形大致相同。考虑到混沌对初始值极为敏感和不同的求解器带来的数值计算误差，两个波形很难完全一致。

### 1.3.5. 频谱图

对前面得到的时域波形图中的  $x = x(t)$  做离散傅里叶变换，得到它对应的频谱图。点数为 200000，采样频率为 1kHz。编写程序得到图7。其中快速傅里叶变换算法采用 `scipy.fft` 库中的 `fft` 函数实现。得到的单边幅频特性中只显示了幅值较大的低频部分，观察得到频率分量最大的约在 0.15Hz 处，但是幅频特性在低频区域并没有显著分离，这也是较为典型的混沌信号的频谱图。

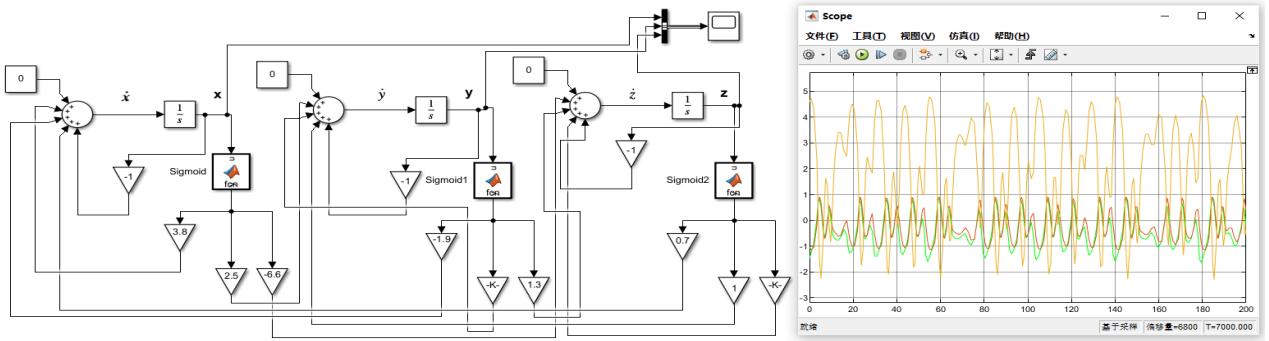


图 5: Simulink 仿真模型及其波形图

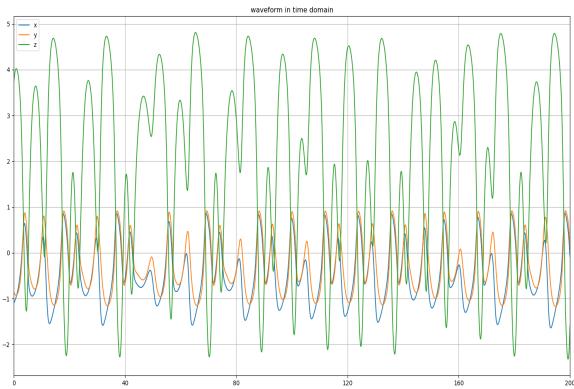


图 6: 使用编程计算得到的波形图

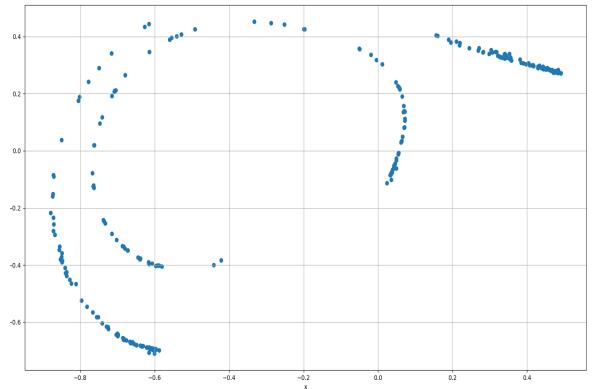


图 8: 双边 Poincare 映射图

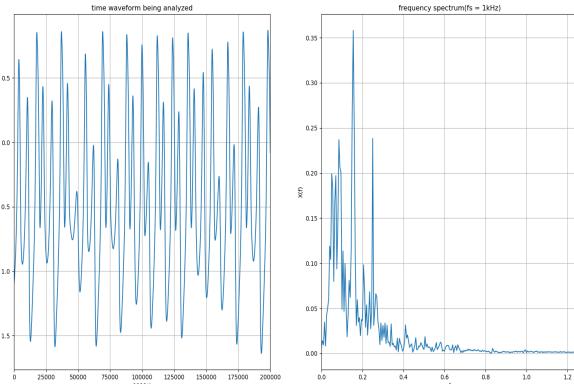


图 7: 时域和其对应的频域波形图

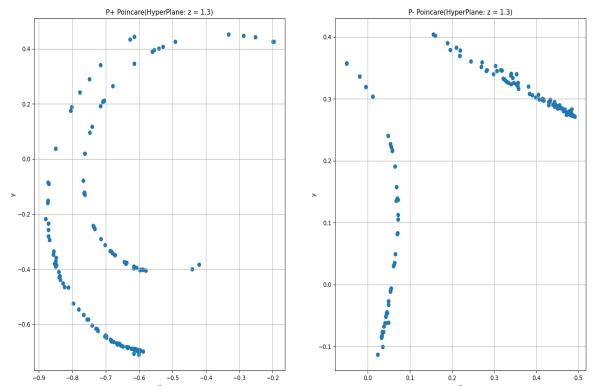


图 9: 单边 Poincare 映射图

### 1.3.6. Poincare 映射 $P_{\pm}$

庞加莱映射是一种能够将  $n$  阶连续时间系统的流转换为  $(n-1)$  维的离散时间系统的序列的变换。自治系统的庞加莱映射是将取一个  $n-1$  维的超平面  $\Sigma$  截取  $n$  维的相轨迹，得到离散点进行分析。其中  $P_{\pm}$  是指不论穿越  $\Sigma$  的方向，一致认为交点都属于离散序列。编写程序绘制 ( $P_{\pm}$ ) 图，其中  $\Sigma$  选取 2 维平面  $z = 1.3$ ，结果共有 273 个点，如图 8。

### 1.3.7. Poincare 映射 $P_+/P_-$

如果考虑轨迹穿越  $\Sigma$  的方向，即可以编写程序得到  $P_+/P_-$  庞加莱映射如图 9。显然  $P_+$  和  $P_-$  是  $P_{\pm}$  的两个组成部分，这是由定义决定的。这些散点图属于典型的混沌系统的离散序列图。从结论中不难发现， $P_+$  和  $P_-$  的序列在相空间中并不重叠。

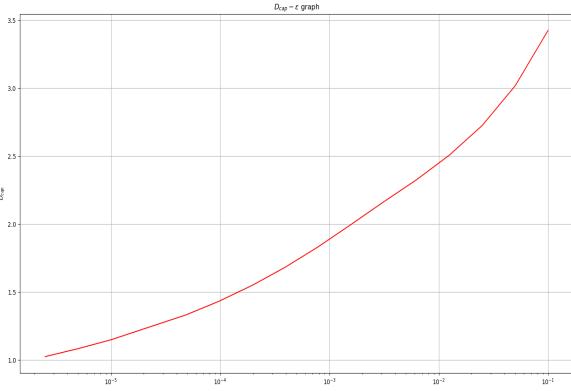


图 10: 分形维度计算过程图

表 1: 分形维度计算过程

$\epsilon$	$D_{cap}$
0.1	3.424554976606713
0.05	3.016921777277306
0.025	2.725691763975478
0.0125	2.5072957970153618
0.00625	2.3261459624292953
0.003125	2.163497109261765
0.0015625	1.995470160247151
0.00078125	1.831411660683802
0.000390625	1.6816519925967714
0.0001953125	1.5490158457696181
9.765625e-05	1.433260750195553
4.8828125e-05	1.333273824415717
1e-05	1.14963760540124
5e-06	1.0843530468056162
2.5e-06	1.0260846966837993

### 1.3.8. 奇异吸引子分形计算

分维最简单的定义是'capacity'[1]: 使用直径为  $\epsilon$  体积元覆盖吸引子 A，而需要的体积元数量记为  $N(\epsilon)$ 。'capacity' 分维定义为：

$$D_{cap} = \lim_{\epsilon \rightarrow 0} \frac{\ln N(\epsilon)}{\ln(1/\epsilon)} \quad (10)$$

因此求解分维的关键在于求  $N(\epsilon)$ 。一种可行的做法是将奇异吸引子所在的多维空间按照  $\epsilon$  为尺度进行有限元划分。通过判断某个单位元内是否有轨迹穿过来决定该单位元是否计入  $N(\epsilon)$ 。按照该算法1编写程序得到的计算结果如表1。

---

### Algorithm 1 'Capacity' Fractal Calculation

---

**Input:** time sequence, dynamical model, diameter, EPS

- 1: solve the Initial Value Problem and get the solution.
- 2: xyzrange = (xmin, xmax, ymin, ymax, zmin, zmax)
- 3: initialize an empty list L
- 4: **while** True **do**
- 5:     divide the space of xyzrange based on diameter.
- 6:     **for all** i that ( $x[i]$ ,  $y[i]$ ,  $z[i]$ ) is stable **do**
- 7:         calculate the coordinate of the point i
- 8:     **end for**
- 9:     count  $N(\epsilon)$
- 10:    L.append( $\frac{\ln N(\epsilon)}{\ln(1/\epsilon)}$ )
- 11:    diameter /= 2
- 12:    reset counter
- 13:    **if** diameter < EPS **then**
- 14:         Break
- 15:    **end if**
- 16: **end while**

**Output:** L: dim = dim(diameter)

---

因为使用更小的尺度计算将显著增加计算机运行时间，因此运行到  $\epsilon = 2.5e-6$  时即停止计算。为了观察  $D_{cap}$  的变化趋势，可以将上表绘制成曲线图10。由于计算过程中是采用的  $\epsilon$  是指数衰减的，因此绘图时将横坐标采用对数格式，可以预测：当  $\epsilon \rightarrow 0$  时， $D_{cap}$  将会收敛。

## 2. 基于循环神经网络的实时电价预测研究

### 2.1. 研究背景

需求侧管理 (DSM) 是 20 世纪 80 年代由美国提出的一种在用户有效参与下充分利用电力资源的系统工程，是电力公司为了影响用户的电力消费、使其产生公司希望的负荷形状而计划和实施的措施。峰谷分时电价作为 DSM 的一种经济手段，根据不同时段确定不同的销售电价，可以鼓励用户调整用电负荷、削峰填谷，有助于提高电能经济效率、降低全社会电价水平、提高社会效益。峰谷电价受多种因素的影响和制约，尤其是用户影响因素。该因素不仅难以量化，而且受负荷特性和各种其他复杂因素影响，加之其延迟特性使得峰谷分时电价效果明显滞后，从而造成供电部门出现先期盈

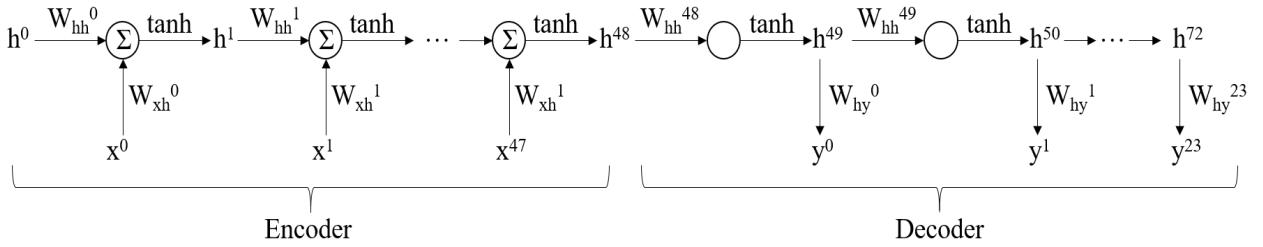


图 11: RNN 模型示意图

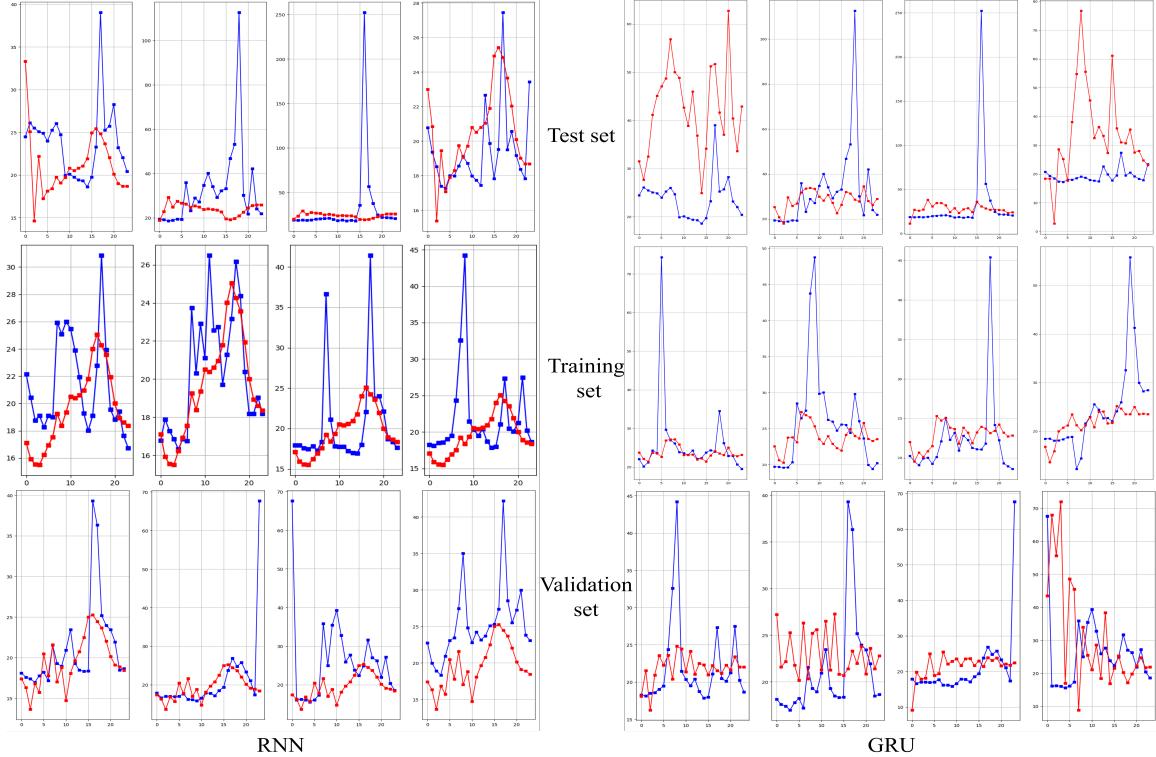


图 12: 选取 4 天的电价预测结果 (红色为预测值, 蓝色为实际值)

利后期亏损的现象 [2]。基于峰谷分时电价体系的时滞性、反馈性、非线性、动态性等特点，峰谷分时电价体系与非线性动力学系统紧密相关。

电价预测就是指: 在考虑市场供求关系, 市场参与者的市场力, 电力成本, 以及电力市场体制结构、社会经济形势等重要因素影响的条件下, 通过利用数学工具对历史数据进行分析和研究, 探索事物之间的内在联系和发展变化规律, 在满足一定精度和速度的情况下, 对未来电力市场中的电力交易价格进行预测。电价是反映电力市场运营状况, 评价市场竞争效率的核心指标, 是电力市场决策的基础。合理预测电价, 对于用户来说可以在电价的低峰期多用电, 高高峰期少用电, 进而达到最小化成本的目的。由于小时实时电价是一种特殊的时间序

列, 并且具有一定的时间因果规律。在人工神经网络领域中, 擅长处理这一类序列的是循环神经网络 (RNN)。比较常见的循环神经网络有简单 RNN、LSTM、GRU、Bi-RNN 等, 本文选择简单 RNN 与 GRU 对实时电价序列进行分析。

## 2.2. 基于 RNN 模型的实时电价预测

### 2.2.1. 原理

使用简单 RNN 用作时间序列预测的逻辑框图为图11. 其中前 48 个 RNN 单元为编码器部分, 陆续接受 48 个时间序列信号, 但是并不产生输出信号, 后 24 个 RNN 单元为解码器部分, 并不接受输入信号, 而是依据已经存在 RNN 中的状态生成相应的时间序列并输

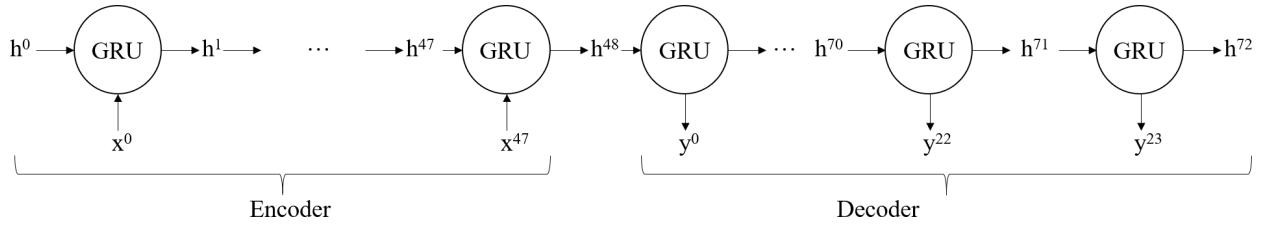


图 13: GRU 模型示意图

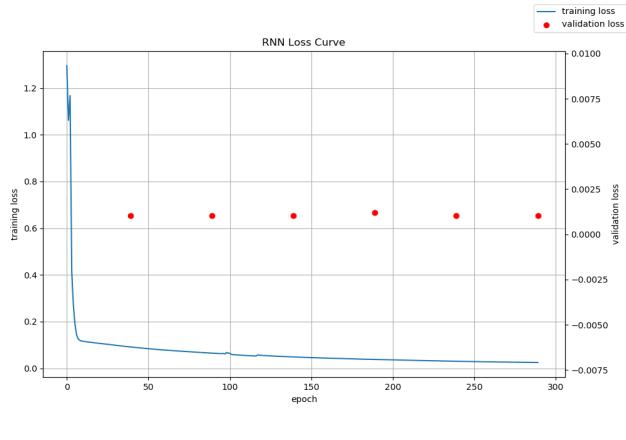


图 14: RNN 模型训练过程的损失函数曲线

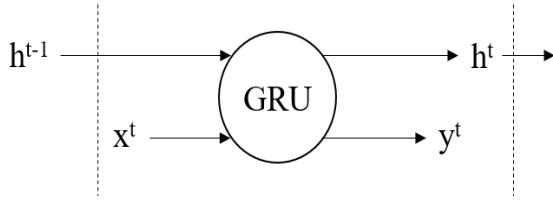


图 15: GRU 单元示意图

出。训练每个样本的流程如下.

正向传播:

Encoder 部分为:

$$h^{t+1} = \tanh(W_{hh}^t h^t + W_{xh}^t x^t), t = 0, 1, \dots, 47 \quad (11)$$

Decoder 部分为:

$$h^{t+1} = \tanh(W_{hh}^t h^t), t = 48, 49, \dots, 71 \quad (12)$$

$$y^t = W_{hy}^t h^{t+49}, t = 0, 1, \dots, 23 \quad (13)$$

求 MSE 损失函数部分为:

$$l = \frac{1}{24} \sum_{i=0}^{23} (y^t - y_{label}^t)^2 l = \frac{1}{24} \sum_{i=0}^{23} (y^t - y_{label}^t)^2 \quad (14)$$

反向传播:

以下公式中:  $h$  是本级的状态向量,  $h'$  是未经过  $\tanh$  的状态,  $h''$  是上一级的状态向量。

Decoder 部分为:

$$\frac{\partial l}{\partial y} = \frac{2}{24} (y - y_{label}) \quad (15)$$

$$\frac{\partial l}{\partial W_{hy}} = \frac{\partial l}{\partial y} h^T \quad (16)$$

$$\frac{\partial l}{\partial h} = \frac{\partial l}{\partial y} W_{hy}^T + \frac{\partial l}{\partial h}|_{previous} \quad (17)$$

$$\frac{\partial h}{\partial h'} = 1 - h^2 \quad (18)$$

$$\frac{\partial l}{\partial W_{hh}} = (\frac{\partial l}{\partial h} \cdot \frac{\partial h}{\partial h'}) @ (h'')^T \quad (19)$$

$$\frac{\partial l}{\partial h''} = W_{hh}^T @ (\frac{\partial l}{\partial h} \cdot \frac{\partial h}{\partial h'}) \quad (20)$$

Encoder 部分为:

$$\frac{\partial l}{\partial h'} = \frac{\partial l}{\partial h} (1 - h^2) \quad (21)$$

$$\frac{\partial l}{\partial W_{xh}} = \frac{\partial l}{\partial h'} @ x^T \quad (22)$$

$$\frac{\partial l}{\partial W_{hh}} = \frac{\partial l}{\partial h'} @ (h'')^T \quad (23)$$

$$\frac{\partial l}{\partial h''} = W_{hh}^T @ \frac{\partial l}{\partial h'} \quad (24)$$

网络参数校正采用梯度下降法:

$$W = W - lr \frac{\partial l}{\partial W} \quad (25)$$

其中  $W$  是神经网络中的参数, 包括  $W_{xh}, W_{hh}, W_{hy}$ 。

整个流程展示为算法2。

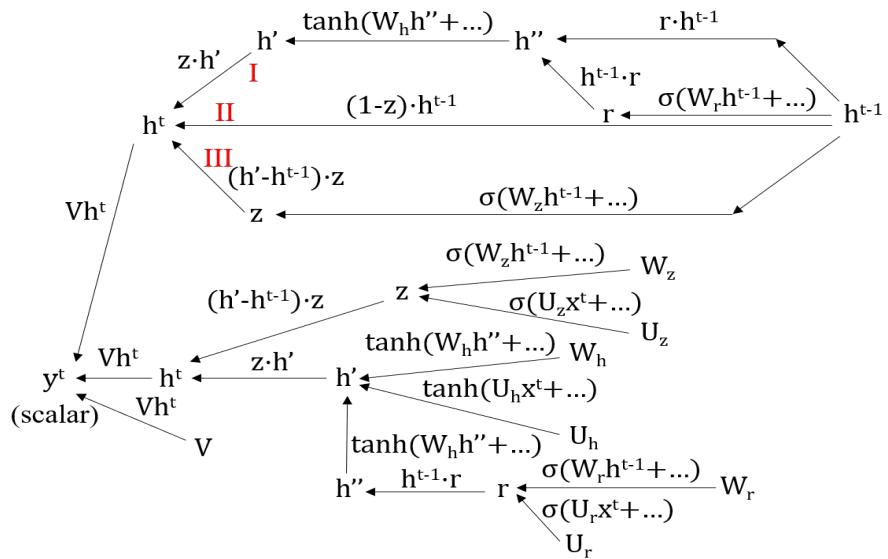


图 16: GRU 模型计算图

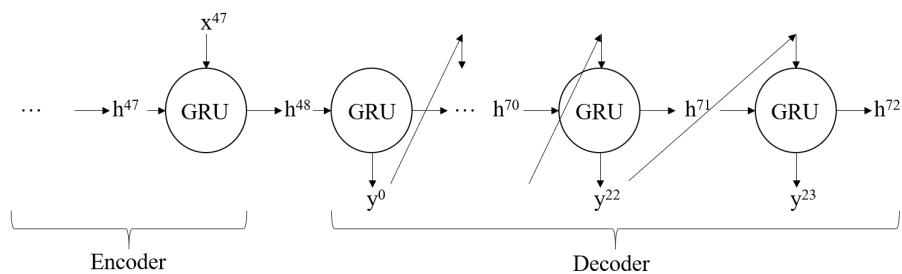


图 17: 带有输出反馈的 GRU 模型示意图

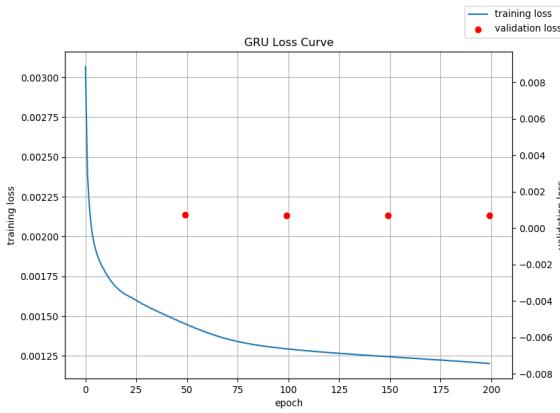


图 18: GRU 模型训练过程的损失函数曲线

### 2.2.2. 实验

本实验采用 2019、2020 年共 730 日的 ATHE-NIA 地点的小时实时电价作为源数据 [[https://dataminer2.pjm.com/feed/rt\\_hrl\\_lmps](https://dataminer2.pjm.com/feed/rt_hrl_lmps)]。假设在连续的三日之内，能够通过前两日的小时电价数据对第三日的小时电价数据做出预测，其中每日小时电价的时间为 5:00AM 至次日 4:00AM。因此本问题是一个时间序列拟合的问题，对该问题宜采用 RNN 求解。其中输入部分为序列长度为 48 的标量小时电价数据，输出部分为序列长度为 24 的标量电价数据，状态向量选用维度为 48 的列向量，共有 73 个。

数据预处理：

730 日中，前两日不可预测，因此从原始数据中可以构建 728 个样本 ( $x, y\_label$ )。其中  $x$  是前两日的电价数据 ( $shape = (48,)$ )， $y\_label$  是被预测日的真实电价数据 ( $shape = (24,)$ )。对这 728 个样本从前到后分别划分为三个集合：前 700 个为训练集 (1 个 epoch)，后 24 个为验证集，最后 4 个为测试集。实时电价数据的最小值在-137.28 左右，最大值在 781.28 左右，极差很大，大部分的每日电价在 0~30 左右波动。在训练开始时，对输入  $x$  做中心化处理：

$$x \leftarrow \frac{x - \mu}{\sigma^2} \quad (26)$$

采用梯度下降法对 RNN 进行训练，学习率设置为 0.0005，每个 epoch 后统计当前模型在训练集上的 loss\_tr，在验证集上的 loss\_va。如果两者都呈下降趋势，则继续训练，如果 loss\_va 不再下降而是回升，说明出现了过拟合现象，此时不再更新网络参数。大约在 300 个 epoch 后验证集上误差不再下降，此时在训

---

### Algorithm 2 RNN Model Training

**Input:** dataset, encodeUnits, decodeUnits, hlen, lr, EPO

- 1: data preprocessing
- 2: get training set, validation set and test set
- 3: initialize network params  $\mathbf{W}$
- 4: initialize state vector  $\mathbf{h}$
- 5: **for**  $e$  in range(EPO) **do**
- 6:      $lossEpo = 0$
- 7:     **for**  $i$  in sizeof(training set) **do**
- 8:         Forward/Backward Propagation Calculation
- 9:          $lossEpo += loss(i)$
- 10:     **end for**
- 11:      $lossEpo /= sizeof(training set)$
- 12:     print('epoch:',  $e$ , 'loss:',  $lossEpo$ )
- 13:      $\mathbf{W} -= lr * d\mathbf{W}$
- 14: **end for**

**Output:** well-trained network params  $\mathbf{W}$

---

练集上的误差为 0.0247414839508201，在验证集上的误差为 0.0010249270541722736，结束训练。

测试集上的表现：

对最后四日的电价数据做出预测如图12。由于所选取的前三日的最大电价均较高，因此预测效果并不理想，最后一日的预测效果相对较好。

训练、验证集上的表现：

由于在测试集上的表现并不理想，因此为了体现网络拟合电价的能力，在训练集上随机抽取 4 日观察预测结果。从图12中可以看出：在这四日上，RNN 能够在一定程度上对电价趋势做出预测，但是精度并不高。

### 2.2.3. 分析

本实验提出的 RNN 具有一定的时间序列拟合能力，但是能力并不足以拟合波动较为剧烈、随机性较强的实时小时电价数据，仅仅是在变化趋势上具有一定的指导作用。

可能提高拟合能力的办法有：

1，在不改变现有模型的情况下，可以增加状态向量的维度。

2，采用更为复杂的 RNN 模型，例如深度 RNN，或者采用对于更长时间序列更为流行的 LSTM, GRU 模型。

但是上述办法会增加模型的复杂性，需要的计算资源也必然增加。

### 2.3. 基于 GRU 模型的实时电价预测

#### 2.3.1. 原理

使用 GRU 用作时间序列预测的逻辑框图为图13。

其中每个 GRU 单元 (图15) 的输入-输出方程为：

$$r = \sigma(W_r h^{t-1} + U_r x^t) \quad (27)$$

$$z = \sigma(W_z h^{t-1} + U_z x^t) \quad (28)$$

$$h'' = r \cdot h^{t-1} \quad (29)$$

$$h' = \tanh(W_h h'' + U_h x^t) \quad (30)$$

$$h^t = (1 - z) \cdot h^{t-1} + z \cdot h' = h^{t-1} + (h' - h^{t-1}) \cdot z \quad (31)$$

$$y^t = Vh^t \quad (32)$$

对所有 GRU 单元进行前向计算就得到了 GRU 的正向传播过程，其中误差的计算与 RNN 一致。由于 GRU 单元的网络参数与变量数比简单 RNN 大幅增加，为了便于分析，绘制计算图16。并根据该计算图给出反向传播公式：

$$\frac{\partial l}{\partial y^t} = \frac{2}{24}(y^t - y_{label}^t) \frac{\partial l}{\partial y^t} = \frac{2}{24}(y^t - y_{label}^t) \quad (33)$$

$$\frac{\partial l}{\partial h^t} = \frac{\partial l}{\partial y^t} V^T + \frac{\partial l}{\partial h^{t+1}} \quad (34)$$

$$\left. \frac{\partial l}{\partial h^{t-1}} \right|_I = v \cdot r + W_r^T @ (v \cdot h^{t-1} \cdot r \cdot (1 - r)) \quad (35)$$

$$v = W_h^T @ (\frac{\partial l}{\partial h^t} \cdot z \cdot (1 - h'^2)) \quad (36)$$

$$\left. \frac{\partial l}{\partial h^{t-1}} \right|_{II} = \frac{\partial l}{\partial h^t} (1 - z) \quad (37)$$

$$\left. \frac{\partial l}{\partial h^{t-1}} \right|_{III} = W_z^T @ (\frac{\partial l}{\partial h^t} \cdot (h' - h^{t-1}) \cdot z \cdot (1 - z)) \quad (38)$$

$$\left. \frac{\partial l}{\partial h^{t-1}} \right|_{III} = \left. \frac{\partial l}{\partial h^{t-1}} \right|_I + \left. \frac{\partial l}{\partial h^{t-1}} \right|_{II} + \left. \frac{\partial l}{\partial h^{t-1}} \right|_{III} \quad (39)$$

$$\frac{\partial l}{\partial W_r} = (W_h^T @ (\frac{\partial l}{\partial h^t} \cdot z \cdot (1 - h'^2)) \cdot h^{t-1} \cdot r \cdot (1 - r)) @ (h^{t-1})^T \quad (40)$$

$$\frac{\partial l}{\partial U_r} = (W_h^T @ (\frac{\partial l}{\partial h^t} \cdot z \cdot (1 - h'^2)) \cdot h^{t-1} \cdot r \cdot (1 - r)) @ (x^t)^T \quad (41)$$

$$\frac{\partial l}{\partial W_h} = (\frac{\partial l}{\partial h^t} \cdot z \cdot (1 - h'^2)) @ (h'')^T \quad (42)$$

$$\frac{\partial l}{\partial U_h} = (\frac{\partial l}{\partial h^t} \cdot z \cdot (1 - h'^2)) @ (x^t)^T \quad (43)$$

$$\frac{\partial l}{\partial W_z} = (\frac{\partial l}{\partial h^t} \cdot (h' - h^{t-1}) \cdot z \cdot (1 - z)) @ (h^{t-1})^T \quad (44)$$

$$\frac{\partial l}{\partial U_z} = (\frac{\partial l}{\partial h^t} \cdot (h' - h^{t-1}) \cdot z \cdot (1 - z)) @ (x^t)^T \quad (45)$$

$$\frac{\partial l}{\partial V} = \frac{\partial l}{\partial y^t} (h^t)^T \quad (46)$$

网络参数校正采用梯度下降法：

$$W = W - lr \frac{\partial l}{\partial W} \quad (47)$$

其中  $W$  是神经网络中的参数，包括  $W_r, W_h, W_z, U_r, U_h, U_z, V$ 。

#### 2.3.2. 实验

本实验采用的数据集与 RNN 实验完全一致，GRU 单元的状态向量维度设置为 24。优化方法为梯度下降法，学习率设置为 0.25。

作出训练误差图18。

大约在 100 个 epoch 后验证集上误差不再下降，此时在训练集上的误差为 0.0012930269502544，在验证集上的误差为 0.000674166563471665。

在测试、训练、验证集上的表现为图12。

#### 2.3.3. 分析

本实验提出的 GRU 相比前文提及的 RNN 的学习能力显著提升，主要体现在误差收敛速度上，这是因为 GRU 使用了门控信号对每个单元的输入、状态和输出进行控制，使得它具有更好的记忆与遗忘特性。但是由于 GRU 单元更为复杂，每个 epoch 的训练时间和 RNN 相比有所增加。GRU 仅使用了 24 维的状态向量，取得的训练效果比使用了 48 维的状态向量的 RNN 还要好。

思考：

1，本实验采用的 GRU 的解码器部分中，如果将之前的输出  $y^{t-1}$  作为下一时间的输入  $x^t$ ，如图17所示，是否会得到更好效果？

答：因为  $y^{t-1}$  是直接使用状态向量  $h^{t-1}$  做线性变换得到的，而  $h^{t-1}$  是可以直接保留到下一个 GRU 单元的，因此这一举措没有必要，反而会增加网络的复杂程度。经过实验证后也的确没有明显的性能提升。

2，对于本电价预测实验，在不改变数据处理的方式下，仅仅更换 Bi-RNN 模型能否对实验效果有显著的改善？

答：在普通的序列处理中，单边 RNN 仅能从前到后提取信息，在输出  $y^t$  时没有考虑  $x^{t+1}$  以及之后的信号，因此采用 Bi-RNN 能够显著提升训练效果。但是本文所提出的时间序列处理模型在输出预测序列之前是先用了一系列 RNN 单元对所有输入信号进行了读取，已经对所有的输入信号做出了综合分析，因此可以预见，即使换用 Bi-RNN 模型，提升效果也并不显著。

### 3. 总结

本文的第一部分以一个三阶的动力学系统为例展开研究，先后给出了该系统的速度场、相轨迹图、时域波形图和频谱图，利用 Simulink 仿真和 Python 编程两种工具从各个侧面描绘了该系统的行为。Poincare 映射是一种将连续时间系统降阶为离散时间系统的手段，本文给出了常见的三种 Poincare 映射的散点图，并且发现了  $P_+$  和  $P_-$ ，即  $P_\pm$  两个的组成成分在空间中彼此相互分开的规律。随后使用有限元方法给出了混沌吸引子的‘capacity’ 分形维度计算，通过观察其收敛特性曲线图可以猜测：在划分区域足够小的时候，存在的极限值就是分形维度。

由于电力系统需求侧的电价波动是由诸多复杂因素组合而成的非线性动力学系统，而电价预测是电力市场的热门研究领域，本文的第二部分以循环神经网络 RNN 以及 GRU 为例对实时电价给出了分析、处理及预测。从实验结果中可以得出结论：GRU 比普通 RNN 的学习能力更强，训练效率更高。两个模型都在一定程度上具有对随机的电价的预测能力，但是由于电价波动受到诸多因素的影响，当天的电价数据不仅仅取决于昨日和前日 48 小时的电价数据，因此若要对电价做出更加精确的预测，可以增加诸如被预测日所处季节，是否为周末，是否处于节假日，以及气温的高低等因素综合加以处理。

通过本课程的学习，我不仅获得了非线性动力学与混沌理论的知识，还探索了神经网络尤其是循环神经网络方面相关的理论，在动力学系统的各种角度的描述方面，我锻炼了自己的编程能力，在神经网络反向传播过程的推导方面，我锻炼了自己的数学推理能力，最后大作业的完成锻炼了自己对办公软件的熟练操作能力。总之这是一门令人收获很大的选修课，感谢任课老师和同学！

### 附录 A. 损失函数求偏导数的分析

本文的反向传播过程中涉及到的所有偏导数大致可以依据以下几种情形推出（其中标量的损失函数用  $l$  表示，行向量用  $r$  表示，列向量用  $c$  表示，矩阵用  $A$  表示， $\cdot$  表示 Hadamard 积， $@$  表示矩阵乘法）。

- $l = rc$

$$\frac{\partial l}{\partial c} = r^T, \quad \frac{\partial l}{\partial r} = c^T$$

- $l = rAc$

$\frac{\partial l}{\partial A} = r^T @ c^T$ 。如果是对  $r$  或  $c$  求导数，将剩余两项按照乘法结合律先合并，接下来和的操作与上一项一致。

- $l = r(\tanh(c))$

$$\frac{\partial l}{\partial c} = r^T \cdot (1 - \tanh(c)^2)$$

- $l = r(\tanh(Ac))$

$$\frac{\partial l}{\partial A} = r^T \cdot (1 - \tanh(Ac)^2) @ c^T$$

$$\frac{\partial l}{\partial c} = ((r^T \cdot (1 - \tanh(Ac)^2))^T @ A)^T$$

$$\frac{\partial l}{\partial c} = A^T @ (r^T \cdot (1 - \tanh(Ac)^2))$$

### References

- [1] Thomas S. Parker and Leon O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer, 1989.
- [2] 黄健柏, 黄向宇, 邵留国, and 扶缚龙. 基于系统动力学的峰谷分时电价模型与仿真(一) 模型的建立. 电力系统自动化, (11):22–27, 2006.
- [3] Steven Strogatz. *Nonlinear Dynamics and Chaos*. Westview, 1998.
- [4] J.E. Marsden, L. Sirovich, and M. Golubitsky. *Differential Equations and Dynamical Systems*. Springer, 2001.