清 华 大 学

综 合 论 文 训 练

题目：基于深度学习的非线性滤波方法研究

系　　别：电子工程系

专　　业：数学与应用数学（第二学位）

姓　　名：许文杰

指导教师：丘成栋

2021 年 5 月 1 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

滤波是指利用一个带噪声系统的观测历史来对系统当前状态进行估计，非线性滤波则是指系统演化或者观测所服从的关于系统状态的方程是非线性的情形下的滤波。滤波广泛地应用在信号处理和金融等领域。而深度学习作为一种基于神经网络的机器学习方法[1]，已经在计算机视觉，自然语言处理和机器人等领域获得了引人注目的成果。本文对神经网络的表示能力进行了理论分析。本文利用深度学习的方法来进行非线性滤波，成功得将该方法应用到了稳定，不稳定，一维和多维的情形。最后本文分析了深度学习的神经网络结构中神经元数量与问题维数和滤波效果的关系。

本文的创新点主要有：

- 把深度学习的方法引入了非线性滤波
- 尝试分析了神经网络结构中神经元个数与问题维数和滤波效果的关系

关键词：非线性滤波，深度学习

ABSTRACT

Filtering refers to estimating the state of a noisy dynamic system using the observation history of the system state. And nonlinear filtering refers to the case where the system equation or the observation equation is nonlinear with respect to the real state of the system. Filtering has been widely applied in domains such as signal processing and finance. On the other hand, deep learning, as a neural network based machine learning methods, has been successfully applied and achieved remarkable success in the fields such as computer vision, natural language processing and robotics.[1]This paper conducted theoretical analysis of the representation ability of neural networks. And this paper introduced a deep learning based nonlinear filtering method and successfully applied the method to stable, unstable, one-dimensional and multi-dimensional nonlinear filtering cases. And this paper also characterizes the relationship of the number of neurons, filtering performance and the dimensions of the filtering problem. This paper's main innovative points include:

- Successfully applying the deep learning based method to nonlinear filtering.
- Characterizing the relationship among filtering performance, the dimensions and the number of neurons.

Keywords: Nonlinear filtering; deep learning

目 录

主要符号对照表

x_t	t 时刻的系统状态
y_t	t 时刻的系统观测
f	系统演化过程中的漂移率
G	系统演化过程中噪声前的矩阵
h	系统观测过程中时间微分项前的函数
v	在系统演化过程中引入的布朗运动
w	在系统观测过程中引入的布朗运动

第 1 章　引言

1.1　研究背景

滤波是指利用一个系统的观测历史对当前系统状态进行估计。滤波问题广泛地存在于军事、金融等实际领域中。实际系统本身在演化过程中可能会存在噪声，观测本身也可能会存在噪声，因此无法通过观测来完全准确地确定系统当前状态。同时现实中存在的系统有大量是非线性的，也就是观测或者系统演化本身关于系统真实状态所满足的关系是非线性的。对于系统演化表现为 Gauss-Markov 过程而观测过程具有 Markov 形式的线性滤波问题，Kalman 等人已经设计出了最优的 Kalman 滤波器 Kalman Bucy 滤波器[2] [3]。而非线性滤波问题则仍然需要更多的相关研究。

1.2　研究现状

一般而言，滤波问题在数学上可以建模成以下随机微分方程组。[4]

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + G(x_t, t)\,\mathrm{d}v_t \tag{1-1}$$

$$\mathrm{d}y_t = h(x_t, t)\,\mathrm{d}t + \mathrm{d}w_t, \tag{1-2}$$

系统状态演化方程 (1-1) 中，x_t 是一个 n 维的系统状态随机向量，其表示了系统当前的状态。$G(x_t, t) \in \mathrm{R}^{n\times r}$ 是一个依赖于系统状态随机向量 x_t 和时间 t 的随机矩阵，演化过程的噪声会首先被这个随机矩阵作用。而 f 是一个关于系统状态随机向量 x_t 和时间 t 的映到 n 维空间的向量值函数，用于描述无噪声条件下系统演化的速率和方向。而在观测方程 (1-2) 中，h 是关于系统状态随机向量 x_t 和时间 t 的映到 m 维空间的向量值函数，刻画了观测与系统状态之间的局部微分关系。系统的噪声来源为 v_t 和 w_t，v_t 是协方差矩阵满足 $\mathrm{E}[\mathrm{d}v_t\,\mathrm{d}v_t^T] = Q(t)\,\mathrm{d}t$ 的 r 维布朗运动过程，而 w_t 是协方差矩阵满足 $\mathrm{E}[\mathrm{d}w_t\,\mathrm{d}w_t^T] = S(t)\,\mathrm{d}t$ 的 m 维布朗运动过程。

对于带高斯噪声的线性系统，卡尔曼等人已经给出了基于预测和线性更正

的最优滤波器，也即 Kalman 滤波器[2]。然而现实系统可能是非线性的，为了处理非线性的滤波问题，人们又提出了利用一阶线性化的思想设计得到的扩展卡尔曼滤波器，也即 Extended Kalman Filter（EKF)[5]。然而，EKF 的一个严重缺陷是其存在不稳定的问题[6]，在一些问题中会出现完全失效的结果。除了这些从滤波的数学理论出发进行解析分析得到滤波算法的工作外，James Ting-Ho Lo 也提出了一种利用循环多层感知机来进行滤波的方法[7]。

另一方面，作为一种最近提出的基于神经网络的机器学习方法，深度学习以其强大的非线性函数拟合能力，已经在诸多的技术领域获得了广泛应用，并取得了令人瞩目的成果。[1]

1.3 课题目标

本课题试图利用深度学习的技术来解决滤波问题，并试图在理论上借鉴滤波理论分析深度学习在滤波中的作用。

第 2 章　理论分析

2.1　数学模型的建立

在本课题中，仅仅考虑时不变系统，也即系统演化和观测模型中的 f, h, G 均不依赖于时间，噪声的相关性质也是时不变的。这里，为了方便讨论，记号 $Y^t = \{y_s | 0 \leqslant s \leqslant t\}$。本文首先给出滤波的一个形式化定义。

定义 1：　对于一个系统演化模型和系统观测模型，滤波可以定义为一个可能依赖于时间的算子 $\mathcal{F}^t : Y^t \mapsto \mathrm{R}^n$，以将 $\mathcal{F}^{\sqcup}(Y^t)$ 作为对当前状态的估计。

在给定 Y^t 的条件下，未归一化的条件概率密度函数 $\sigma(x,t)$ 将满足著名的 DMZ 方程[8]。DMZ 方程是一个随机偏微分方程，难以直接求解[4]。通过对时间进行采样和离散化，并进行指数变换的操作[4]，将可以得到一个一个时间间隔内未归一化的条件概率密度函数 u 所满足的如下的 Kolmogorov Forward Equation（KFE）[4]。

$$\frac{\partial u}{\partial t}(x,t) = Fu(x,t), \tag{2-1}$$

其中 F 表示一个空间算子，其可以定义为 $F = K - \frac{1}{2}h^T S^{-1} h$，其中 K 是另外一个空间算子，如下定义[4]。

$$K(*) \coloneqq \frac{1}{2}\sum_{k,l=1}^{n} \frac{\partial^2}{\partial x_k \partial x_l}[(GQG^T)_{kl} *] - \sum_{k=1}^{n} \frac{\partial (f_k *)}{\partial x_k}, \tag{2-2}$$

2.2　人工神经网络的基本数学原理

人工神经网络是对人类神经网络的模拟，它是由一个一个的人工神经元通过模拟连接和模拟信号传递获得的。[9] 单个神经元的功能为对输入进行线性加权之后通过一个非线性函数产生输出。人工神经网络的拓扑结构则表达了神经元之间信号传递的方向。通常，在进行人工神经网络的设计时人们会考虑分层的结构。如图2.1所示，为一个简单的单个隐含层的神经网络。假设单个神经元所采用的激活函数（也即一个非线性函数）为 ϕ，同时输入到该神经元的 n 个输入

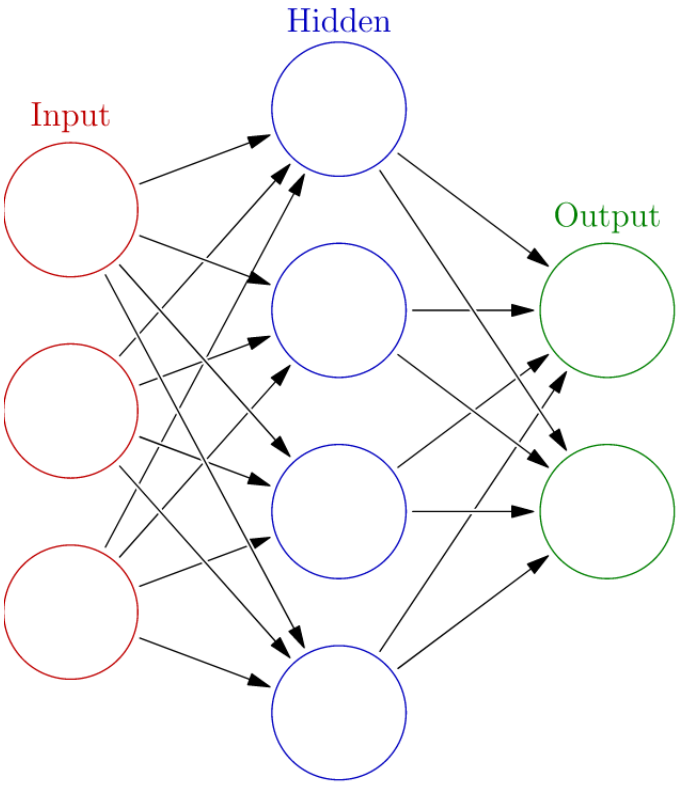


图 2.1　单隐含层的神经网络[9]

分别为 $x_1, x_2, \cdots, x_n$，每个输入所对应的权重分别为 $w_1, w_2, \cdots, w_n$，则该神经元产生的输出 a 将满足下列方程。

$$a = \phi(\sum_{i=1}^{n} w_i x_i + b) \tag{2-3}$$

这其中 b 是一个偏置。进而容易推出，对于一个多输入，拥有单隐含层，单输出的人工神经网络，其所表示的是如下这样一个函数。

$$f(x_1, x_2, \cdots, x_n) = \sum_{j=1}^{m} v_j \phi(\sum_{i=1}^{n} w_{ij} x_i + b_j) \tag{2-4}$$

由于引入了大量的线性加权项，偏置项以及一个非线性函数单元，直觉上，神经网络拥有极强的对非线性函数的拟合能力。

2.3 神经网络表示能力的分析

神经网络以及基于神经网络的深度学习在工程实践中的巨大成功的一个根源在于神经网络的表示能力。本节将简单分析神经网络的表示能力并给出 Kurt Hornik 的关于神经网络极限性质的一个定理[10]。本节仅考虑单隐含层的神经网络，为了方便讨论，沿用[10] 的分析思路，做如下定义。

定义 2: 定义 $\mathrm{SHLNN}_k^n := \{f : \mathrm{R}^k \mapsto \mathrm{R} | f(x) = \sum_{i=1}^n v_i \phi(< w_i, x > + b_i), v_i, b_i \in \mathrm{R}, w_i, x \in \mathrm{R}^k\}$，其中 SHLNN 表示 Single Hidder Layer Neural Network。

通过简单的分析可知，SHLNN_k^n 对于若干的函数操作是封闭的。

命题 1: 若 $f \in \mathrm{SHLNN}_k^n$，则 $f(Q^T x), f(\Lambda^{-1} x), f(x+t), \alpha f \in \mathrm{SHLNN}_k^n$，其中 Q 是一个单位正交阵，Λ 是一个对角元非负的对角阵，$t \in \mathrm{R}^k$，$\alpha \in \mathrm{R}$，也即 SHLNN_k^n 对函数的平移、旋转和伸缩是封闭的。

证明 对于 $\forall f(x) \in \mathrm{SHLNN}_k^n$，有 $f(Q^T x) = \sum_{i=1}^n v_i \phi(< w_i, Q^T x > + b_i) = \sum_{i=1}^n v_i \phi(< Q w_i, x > + b_i) \in \mathrm{SHLNN}_k^n$，$f(\Lambda^{-1} x) = \sum_{i=1}^n v_i \phi(< w_i, \Lambda^{-1} x > + b_i) = \sum_{i=1}^n v_i \phi(< \Lambda^{-1} w_i, x > + b_i) \in \mathrm{SHLNN}_k^n$，$f(x+t) = \sum_{i=1}^n v_i \phi(< w_i, x > + < w_i, t > + b_i) \in \mathrm{SHLNN}_k^n$,$\alpha f(x) = \sum_{i=1}^n \alpha v_i \phi(< w_i, x > + b_i) \in \mathrm{SHLNN}_k^n$。 □

为了研究神经网络的极限性质，定义 $\mathrm{SHLNN}_k^\infty = \cup_{n=1}^\infty \mathrm{SHLNN}_k^n$。Kurt Hornik 也给出了如下关于神经网络的极限表示能力的定理。

定理 2.1: [10] 若非线性激活函数 ϕ 是非 0 有界的，则相应的 SHLNN_k^∞ 在 $L^p(\sigma)$ 中稠密，其中 σ 是 R^k 中的任意有限测度。

2.4 滤波模型的离散化

考虑到在实际的工程实践中，观测信号往往是经过采样的，同时为了计算机仿真和处理的方便，本文考虑对连续滤波模型进行离散化，并且仅考虑噪声相互独立的情形。经过离散化之后的滤波模型如下所示:

$$\Delta x_k = f(x_k)\Delta t + G(x_k)\sqrt{\Delta t}\epsilon_{1k} \tag{2-5}$$

$$\Delta y_k = h(x_k)\Delta t + \sqrt{\Delta t}\epsilon_{2k}, \tag{2-6}$$

其中 Δt 为仿真或者实际工程中的采样时间间隔，x_k 表示第 k 个采样时刻对应的系统真实状态，Δx_k 和 Δy_k 则是第 k 个采样时刻到第 $k+1$ 个采样时刻系统状态和观测获得的增量，ϵ_{1k} 和 ϵ_{2k} 分别服从 $\mathcal{N}(0, I_{n\times n})$ 和 $\mathcal{N}(0, I_{m\times m})$ 的分布，且两者相互独立。

第 3 章　算法设计

为了更加直观的表达滤波模型，可以做出其所对应的系统框图，如图3.1。从

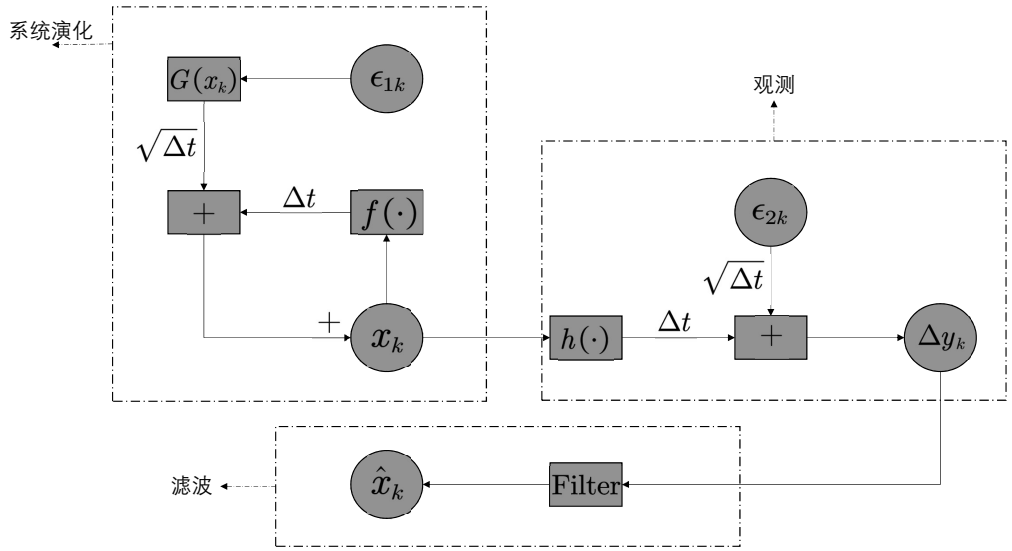


图 3.1　滤波系统模型框图

系统框图可以看出来，为了估计当前系统的真实状态，不仅当前的观测结果是有价值，过往的所有历史观测都可能是有价值的，因此在滤波模块中有必要引入记忆机制，也就是说滤波的系统（某种意义上可以视为由系统真实状态到系统观测的系统的逆系统）应当是一个有记忆系统。而引入记忆的一种方法就是引入反馈。

在深度学习中，有一类神经网络，也即循环神经网络 (Recurrent Neural Network, RNN) 可以引入所谓的记忆机制。这一类神经网络已经成功地被应用到了需要记忆机制的语音识别等场景中。[1]。

3.1 基于 RNN 的网络结构设计

基于以上考虑，本课题计划使用 RNN 来设计相应的滤波算法。设计得到的 RNN 的网络模型如图3.2所示。

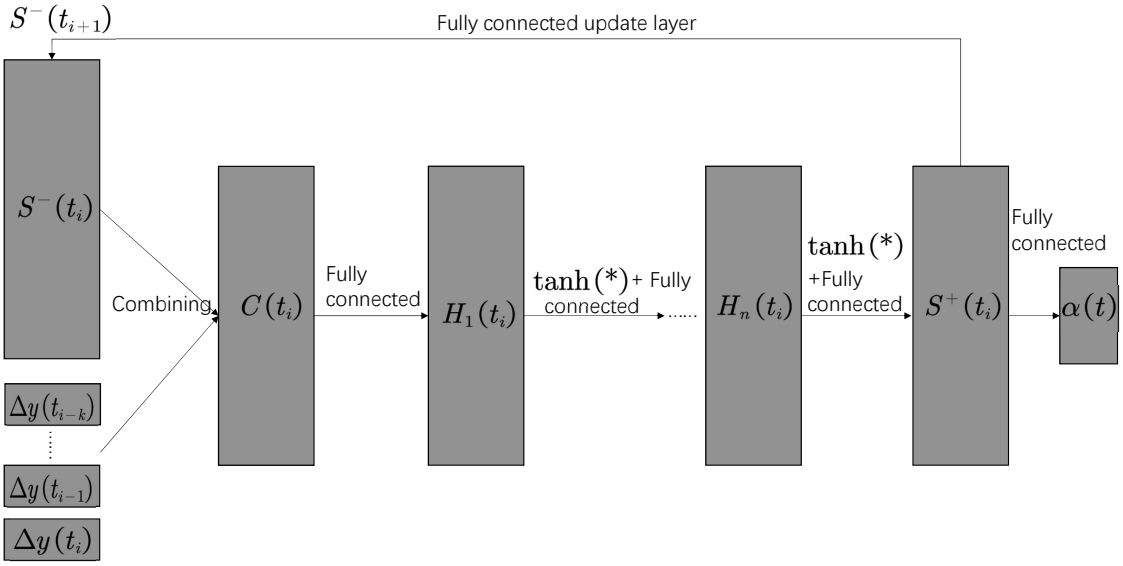


图 3.2　基于 RNN 的用于滤波的神经网络架构

其中神经网络的输入为当前时刻到紧邻之前共 k 个采样时刻得到的 Δy，而神经网络的输出为对当前系统状态的估计 $\hat{x}$（图中用 α 表示）。

直观地，可以这样来理解以上神经网络的结构：从输入和 S^- 到 S^+ 可以直观地理解为从先验混合分布的一个“表示” S^- 结合当前和历史观测信息，变换为一个后验混合分布的“表示” S^+，而“表示”的方式是通过“学习”获得的。而从 S^+ 到 S^- 的部分则可以直观地理解为对 KFE 的模拟。而从 S^- 到最终输出的部分，则可以直观地理解为混合分布的“表示”系数对表示“基”相应的加权平均。在这样一个模型中，之前所提及的记忆单元即体现在 S 向量上。

3.2 模型参数的优化

在设计好网络模型之后，需要对网络参数进行优化，本课题采用了一种数据驱动的“训练”方法来对网络参数进行优化。优化的算法如算法1。这样的仿

设定采样路径数 N，训练的时间步数 M，时间步长 Δt，训练过程参数每过 l 个时间步数进行一次优化，设定学习率（也即参数优化过程中梯度项前的系数）为 λ
for $1 \leqslant i \leqslant N$ **do**
    独立同分布地从初始分布 $\sigma_0(x)$ 中采样得到 $x_0(\omega_i)$，并设定 $y_0(\omega_i) = 0$
end for
for $0 \leqslant m \leqslant \frac{M}{l} - 1$ **do**
    $\text{loss}(\theta) \leftarrow 0$
    for $ml \leqslant k < (m+1)l$ **do**
        for $1 \leqslant i \leqslant N$ **do**
            $\Delta x_k(\omega_i) \leftarrow f(x_k(\omega_i))\Delta t + G(x_k(\omega_i))\sqrt{\Delta t}\epsilon_{1k}(\omega_i)$
            $\Delta y_k(\omega_i) \leftarrow h(x_k(\omega_i))\Delta t + \sqrt{\Delta t}\epsilon_{2k}(\omega_i)$,
        end for
        for $1 \leqslant i \leqslant N$ **do**
            将 $\Delta y_k(\omega_i)$ 输入到设计好的神经网络中，得到相应的输出 $\hat{x_k}(\omega_i)$
            $\text{loss}(\theta) \leftarrow \text{loss}(\theta) + |\hat{x}_k(\omega_i) - x_k(\omega_i)|$
        end for
    end for
    $\theta \leftarrow \theta - \lambda\nabla\text{loss}(\theta)$
end for

Algorithm 1: 网络参数优化算法

真和优化过程可以连续进行多个 epochs（世代），所谓优化一个世代，便是指上述优化过程进行了一遍。

3.3 模型滤波性能的验证

假设优化了第 n 个世代之后的模型参数为 $\theta^{(n)}$，为了验证当前模型的滤波性能，可以调用以下验证模块。为了更好地刻画模型滤波的性能，做如下两个定

给定当前模型的参数 θ，验证过程中采样的轨道数目 N'，时间步长 M
for $1 \leqslant i \leqslant N'$ **do**
    独立同分布地从初始分布 $\sigma_0(x)$ 中采样得到 $x_0(\omega_i)$，并设定 $y_0(\omega_i) = 0$
end for
for $0 \leqslant k < M$ **do**
    for $1 \leqslant i \leqslant N'$ **do**
        $\Delta x_k(\omega_i) \leftarrow f(x_k(\omega_i))\Delta t + G(x_k(\omega_i))\sqrt{\Delta t}\epsilon_{1k}(\omega_i)$
        $\Delta y_k(\omega_i) \leftarrow h(x_k(\omega_i))\Delta t + \sqrt{\Delta t}\epsilon_{2k}(\omega_i)$,
        将 $\Delta y_k(\omega_i)$ 输入到参数为 θ 的神经网络中，可以得到对当前系统状态的估计 $\hat{x}_k(\omega_i)$
        计算可得估计的 $\mathrm{SE}_k(\omega_i) \leftarrow |x_k(\omega_i) - \hat{x}_k(\omega_i)|^2$（其中 SE 表示平方误差，也即 Square Error）
    end for
end for

Algorithm 2: 当前模型滤波性能的验证

义。

定义 3：定义 PMSE_k 为对不同路径进行滤波时的对 x_k 估计的平均的误差，也即 $\mathrm{PMSE}_k := \frac{1}{N}\sum_{i=1}^{N}\mathrm{SE}_k(\omega_i)$。

定义 4：定义 TPMSE 为对不同时刻的 PMSE_k 再取平均之后得到的结果，也即 $\mathrm{TPMSE} := \frac{1}{M}\sum_{k=1}^{M}\mathrm{PMSE}_k$。

本文接下来将使用这两个指标来评价设计得到的算法的滤波性能。

第 4 章　实验结果

4.1　算法实现与实验平台

本项目使用 PyTorch 作为深度学习的框架来实现深度学习算法，并且在一台 Intel(R) Xeon(R) CPU E5-2603 v2 @ 1.80GHz 的机器上进行了实验。

4.2　一个简单的一维 Cubic Sensor 算例

首先利用上述算法框架和优化策略，在一个简单的一维 Cubic Sensor 算例上进行实验。一维 Cubic Sensor 算例的演化和观测方程如下[4]。

$$\mathrm{d}x_t = \mathrm{d}w_t \tag{4-1}$$

$$\mathrm{d}y_t = x_t^3\,\mathrm{d}t + \mathrm{d}v_t \tag{4-2}$$

设定各个参数如表4.1。

表 4.1　一维 Cubic Sensor 算例的参数设定

参数名	参数值
采样时间间隔 Δt	0.01
采样的时间步数 M	5000
采样的路径数目 N	1000
训练的世代数目	100
从输入和 S^- 到 S^+ 的 FC 层（全连接层）的数目	2

设定 $\dim(S^-) = \dim(S^+) = 14$，得到的滤波效果如图4.1。

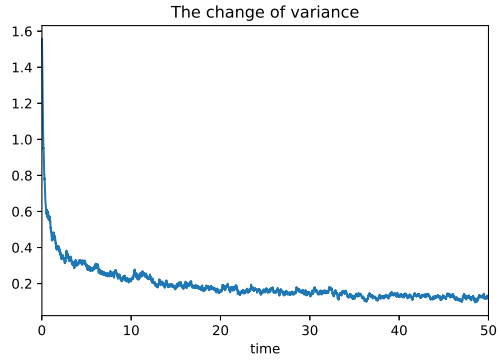


图 4.2　在一维的 Cubic 算例中 PMSE_k 随着时间的演变

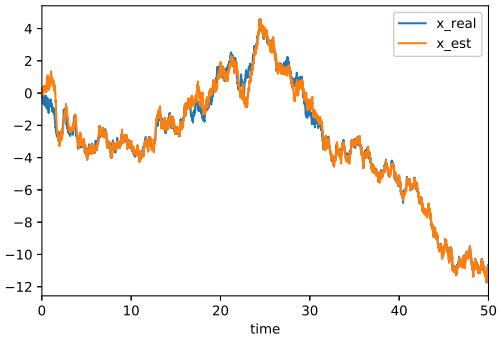


图 4.1　深度学习算法在一维的 Cubic Sensor 算例上的滤波效果示例

可以看到，除了在滤波起始的一段时间内，滤波算法给出的估计值和真实值产生了较大偏差外，在滤波的后半部分，滤波效果越来越好。进一步地，可以计算 PMSE_k 随着滤波时间的变化，如图4.2所示为随着滤波时间的推移，PMSE_k 的变化。进一步地，可以对比谱方法和深度学习方法在这个算例上的滤波性能和时间效率，如表4.2。

方法	TPMSE	单个时间步的更新时间
基于深度学习的方法	0.18	7×10^{-4} 秒
谱方法	> 0.4	10^{-4} 秒

表 4.2　基于深度学习的方法和谱方法在一维 Cubic Sensor 的算例上

4.3 二维 Cubic Sensor 的算例

将 Cubic Sensor 的算例拓展到二维的情形。考察以下二维的 Cubic Sensor 滤波模型。

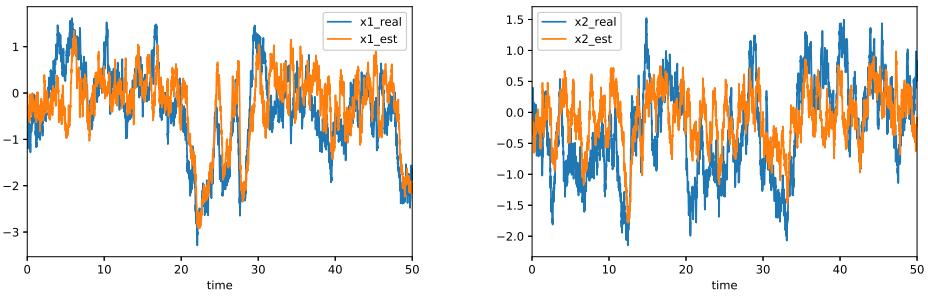
$$\mathrm{d}x_1(t) = (-0.4x_1(t) + 0.1x_2(t))\,\mathrm{d}t + \mathrm{d}v_1(t) \tag{4-3}$$

$$\mathrm{d}x_2(t) = -0.6x_2(t)\,\mathrm{d}t + \mathrm{d}v_2(t) \tag{4-4}$$

$$\mathrm{d}y_1(t) = x_1^3(t)\,\mathrm{d}t + \mathrm{d}w_1(t) \tag{4-5}$$

$$\mathrm{d}y_2(t) = x_2^3(t)\,\mathrm{d}t + \mathrm{d}w_2(t) \tag{4-6}$$

由于矩阵 $[-0.4, 0.1; 0, -0.6]$ 的两个特征值均为负数，因而该二维系统是一个稳定的系统。最终得到的对 x_1 和 x_2 的滤波效果分别如图4.3(a) 和图4.3(b)。



(a) 二维 Cubic Sensor 算例中对 x_1 估计的效果样例 (b) 二维 Cubic Sensor 算例中对 x_2 估计的效果样例

图 4.3 对系统状态的估计结果样例

在本算例中，再次将得到的结果和用谱方法得到的结果进行比较，得到的结果如表4.3。可以看到，在相当的时间消耗下，基于深度学习的方法获得了和

表 4.3 基于深度学习的方法和谱方法在二维 Cubic Sensor 算例中滤波性能和时间效率的比较

滤波方法	x_1 的 TPMSE	估计 x_2 的 TPMSE	单个时间步的时间
基于深度学习的方法	0.457	0.456	8×10^{-4} 秒
谱方法 (每一维的基的个数为 15)	0.418	0.531	2.8×10^{-4} 秒

谱方法滤波性能相当的效果。

4.4 不稳定的 Lorenz 系统算例

4.2和4.3均考虑了稳定的系统，下面本文将考察一个不稳定的系统，即Lorenz系统。本算例考虑的三维 Lorenz 系统的对应的方程如下[11]。

$$\mathrm{d}x_1(t) = \alpha(x_2(t) - x_1(t))\,\mathrm{d}t + \mathrm{d}v_1(t) \tag{4-7}$$

$$\mathrm{d}x_2(t) = (\gamma x_1(t) - x_2(t) - x_1(t)x_3(t))\,\mathrm{d}t + \mathrm{d}v_2(t) \tag{4-8}$$

$$\mathrm{d}x_3(t) = (x_1(t)x_2(t) - \zeta x_3(t))\,\mathrm{d}t + \mathrm{d}v_3(t) \tag{4-9}$$

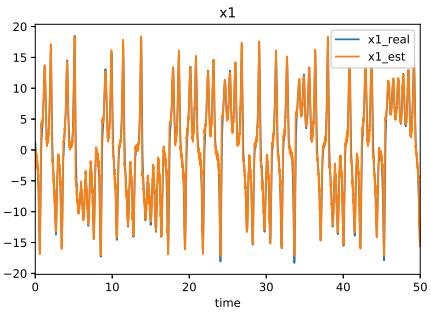
$$y(t_k) = [x_1(t_k)x_2(t_k) + x_1(t_k)x_3(t_k) + x_2(t_k)x_3(t_k)]/100 + w(t_k) \tag{4-10}$$

这其中参数取值为 $\alpha = 10, \zeta = 8/3, \gamma = 28$，初始状态向量取为 $x_1 = 1.508870, x_2 = -1.531271, x_3 = 25.46091$。由于 Lorenz 系统是一个较为复杂和高度动态的系统，为了能够跟踪这样一个系统，需要设计的神经网路来做出迅捷的响应，因而在此例中，设计得到的深度学习的相关超参数（非神经网络内部边权值）和一些其他相关参数如表4.4。该场景下使用的神经网络相对于一个稳定系统而言需要提供更强的表示能力，因而神经网络的层数，状态表示向量 S^+ 和 S^- 的维数和神经网络的直接记忆深度相较前两个算例中的稳定系统都要更大一些。如

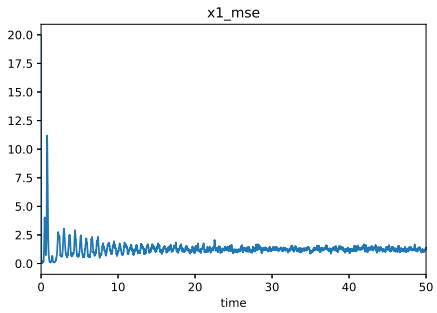
参数名	参数值
采样时间间隔 Δt	0.005
采样的时间步数 M	10000
采样的路径数目 N	1000
训练的世代数目	1000
从输入和 S^- 到 S^+ 的 FC 层（全连接层）的数目	5
从 S^+ 到产生对当前状态的估计 $\hat{x}$ 的 FC 层的数目	2
从 S^+ 到下一个时刻的 S^- 的全连接层 (FC）层的数目	2
状态表示向量 S^- 和 S^+ 的维数	100

表 4.4　三维 Lorenz 系统算例中的参数

图4.4(a)和4.4(b)分别为该 Lorenz 系统中对 x_1 估计的一个样例和估计的 PMSE_k 随着时间的变化。可以看到除了在初始的一段时间内，基于深度学习的方法对系统状态向量估计地有些不稳定外，在往后的时间内，该方法可以取得较为稳定和良好的系统状态向量的估计结果。



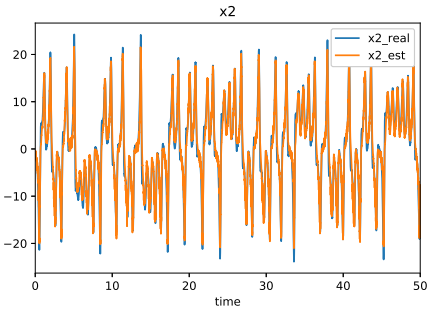
(a) 对 x_1 估计结果的样例



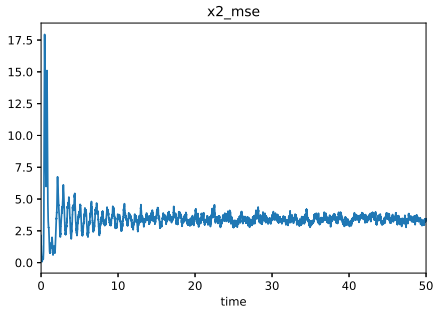
(b) 对 x_1 估计的 PMSE_k 随着时间的变化

图 4.4　Lorenz 系统算例下基于深度学习的方法对 x_1 的估计情况

如图4.5(a)和4.5(b)分别表示基于深度学习的方法对 x_2 的估计样例和对 x_2 估计的 PMSE_k 随着时间的变化，可以看到其结果与 x_1 的情形相类似。



(a) 对 x_2 估计结果的样例



(b) 对 x_2 估计的 PMSE_k 随着时间的变化

图 4.5　基于深度学习的方法在 Lorenz 算例中的滤波效果

第 5 章　滤波效果和神经元的数量之间的关系刻画

根据 James Ting-Ho Lo 在 1994 年给出的关于神经滤波器极限性质的一个主定理[7]，单个隐含层的层内全连接的神经滤波器的滤波结果可以收敛到 $\mathbb{E}(x_t|Y^t)$，也即可以收敛到最小 PMSE_k 的滤波器。本文接下来探讨在本文所给出的基于深度学习的滤波器结构和滤波问题下，滤波器中的神经元个数是如何影响到滤波性能的。

5.1　一维 Cubic Sensor 中的神经元数量对滤波性能的影响

为了能够刻画基于深度学习的方法得到的滤波算法的滤波性能和神经网络中的神经元的个数的相互关系，固定神经网络的的结构（从输入到 S^+ 共两个全连接层，从 S^+ 都 S^- 有一个全连接的 FC 层，从 S^+ 到最后对当前状态的估计有一个全连接的 FC 层）。考虑到神经元的数量大致与状态表示向量的维数成正比。本文将使用状态表示向量的维数来表示神经网络所使用的神经元的数量，实验的结果如图5.1。从图中可以有下面几个观察：

图 5.1　一维 Cubic Sensor 算例中不同状态表示向量维数下 TPMSE 随着训练世代的变化

- 在完全无记忆（$\dim(S^-) = 0$）的情况下，神经网络只依赖当前观测对当前系统状态进行估计，得到的效果较差
- 引入反馈记忆机制后，随着训练世代的增加，滤波性能逐渐收敛
- 表示维数增加到 2 维之后，再增加表示维数将不能改善收敛结果
- 表示维数的增加，将可以加速神经网络滤波性能的收敛

5.2 二维 Cubic Sensor 算例中的神经元数量对滤波性能的影响

类似于一维 Cubic Sensor 算例的情形，同样固定用于二维 Cubic Sensor 算例的神经网络结构（从输入到 S^+ 共四个全连接的 FC 层，从 S^+ 到 S^- 一共三个全连接的 FC 层，从 S^+ 到输出一共一个全连接的 FC 层）。考虑到二维情形较为复杂，增加了深度为 10 的直接记忆单元，也就是在网络的输入中是最近 10 个采样时刻对系统的观测。最终得到的不同表示向量的维数下，对 x_1 的估计效果随着训练世代的变化如图5.2，对 x_2 的估计效果随着训练世代数的变化如图5.3。

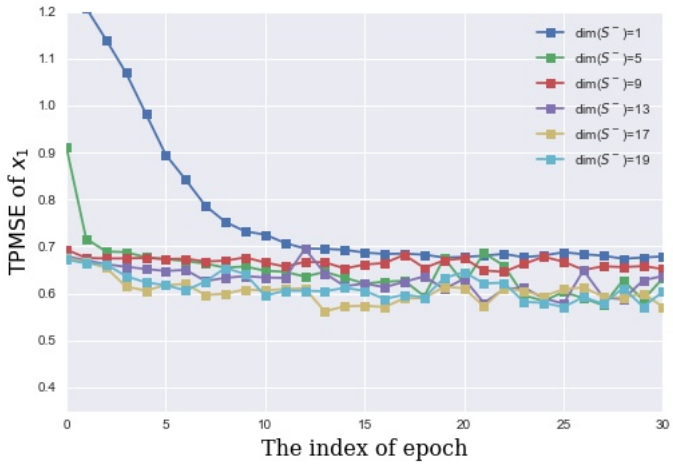


图 5.2　二维 Cubic Sensor 的算例中随着训练世代数的增加对 x_1 估计效果的变化

可以发现和一维情形类似的现象如下：

1. 随着神经元数量的增加，滤波效果收敛地越快
2. 随着神经元数量的增加，滤波效果将收敛到一个稳定的值

值得注意的一点是，在该算例中，状态向量 S^- 表示维数上升到 17 左右时，滤波性能就已经收敛了，这甚至比一维情形还好，事实上这是因为二维情形下增加

图 5.3　二维 Cubic Sensor 的算例中随着训练世代的增加对 x_2 估计效果的变化

了 10 个直接记忆单元，并且网络也更深了，因而最终到达收敛时所需要的状态向量 S^- 的表示维数可能会较小。另外一个值得注意的地方是当 $\dim(S^-) = 1$ 时，当训练世代数为 2 到 10 时，对 x_2 的估计 TPMSE 随着训练世代的增加，反而呈现上升的趋势，这是因为在对神经网络参数进行优化时，优化目标选择的是在所有维数上的估计 TPMSE，事实上，x_2 估计 TPMSE 上升的过程，恰好是对 x_1 估计 TPMSE 急剧下降的过程。

5.3　三维情形下 Cubic Sensor 算例中神经元数量对滤波性能的影响

进一步地，将 Cubic Sensor 的问题维数扩充到三维，考虑以下三维 Cubic Sensor 的滤波模型。

$$\mathrm{d}x_1(t) = (-0.4x_1(t) + 0.1x_2(t) + 0.3x_3(t))\,\mathrm{d}t + \mathrm{d}v_1(t) \tag{5-1}$$

$$\mathrm{d}x_2(t) = (0.2x_1(t) - 0.6x_2(t) + 0.3x_3(t))\,\mathrm{d}t + \mathrm{d}v_2(t) \tag{5-2}$$

$$\mathrm{d}x_3(t) = (0.3x_1(t) + 0.1x_2(t) - 0.9x_3(t))\,\mathrm{d}t + \mathrm{d}v_3(t) \tag{5-3}$$

$$\mathrm{d}y_1(t) = x_1^3(t)\,\mathrm{d}t + \mathrm{d}w_1(t) \tag{5-4}$$

$$\mathrm{d}y_2(t) = x_2^3(t)\,\mathrm{d}t + \mathrm{d}w_2(t) \tag{5-5}$$

$$\mathrm{d}y_3(t) = x_3^3(t)\,\mathrm{d}t + \mathrm{d}w_3(t) \tag{5-6}$$

如图5.4所示为三维情形下滤波过程中对 x_1 估计的 TPMSE 随着训练世代的变化。可以看到结果和二维的情形高度类似，也当 $dim(S^-) = 10$ 左右估计的 TPMSE 基本收敛，再增加表示维数不再能进一步显著降低估计的 TPMSE。

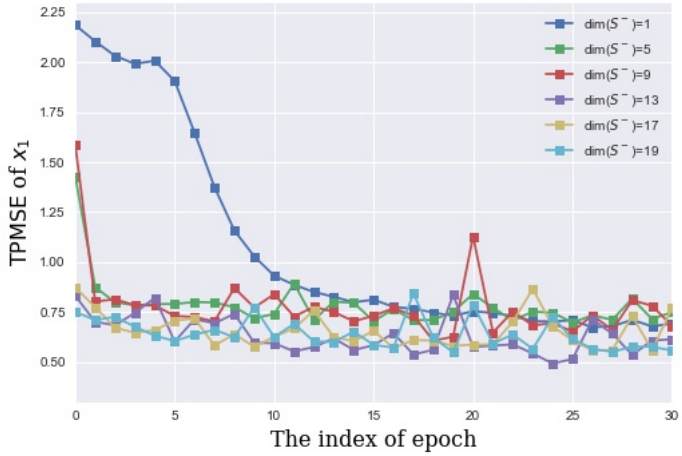


图 5.4　三维 Cubic Sensor 算例中对 x_1 估计的 TPMSE 随着训练世代的变化

如图5.5所示为三维情形下滤波过程中对 x_2 估计的 TPMSE 随着训练世代的变化。

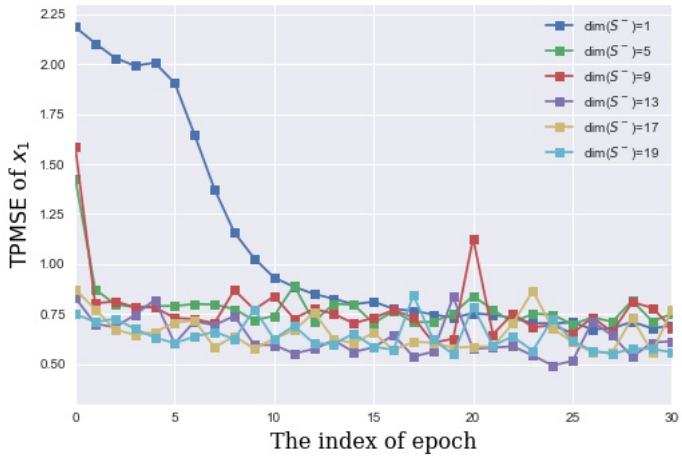


图 5.5　三维 Cubic Sensor 算例中对 x_2 估计的 TPMSE 随着训练世代的变化

5.4 四维情形下 Cubic Sensor 算例中神经元数量对滤波性能的影响

更进一步地，将 Cubic Sensor 的算例推广到四维的情形，考虑如下的滤波模型：

$$\mathrm{d}x_1(t) = (-0.4x_1(t) + 0.1x_2(t) + 0.3x_3(t) + 0.5x_4(t))\,\mathrm{d}t + \mathrm{d}v_1(t) \tag{5-7}$$

$$\mathrm{d}x_2(t) = (0.2x_1(t) - 0.6x_2(t) + 0.3x_3(t) + 0.5x_4(t))\,\mathrm{d}t + \mathrm{d}v_2(t) \tag{5-8}$$

$$\mathrm{d}x_3(t) = (0.3x_1(t) + 0.1x_2(t) - 0.9x_3(t) + 0.3x_4(t))\,\mathrm{d}t + \mathrm{d}v_3(t) \tag{5-9}$$

$$\mathrm{d}x_4(t) = (0.4x_1(t) - 0.5x_2(t) + 0.6x_3(t) - 1.1x_4(t))\,\mathrm{d}t + \mathrm{d}v_4(t) \tag{5-10}$$

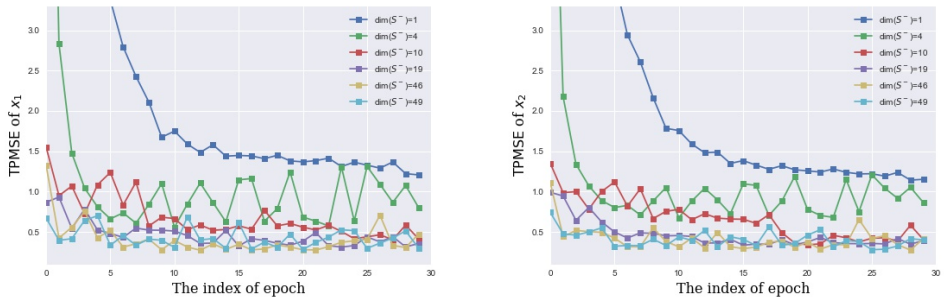
$$\mathrm{d}y_1(t) = x_1^3(t)\,\mathrm{d}t + \mathrm{d}w_1(t) \tag{5-11}$$

$$\mathrm{d}y_2(t) = x_2^3(t)\,\mathrm{d}t + \mathrm{d}w_2(t) \tag{5-12}$$

$$\mathrm{d}y_3(t) = x_3^3(t)\,\mathrm{d}t + \mathrm{d}w_3(t) \tag{5-13}$$

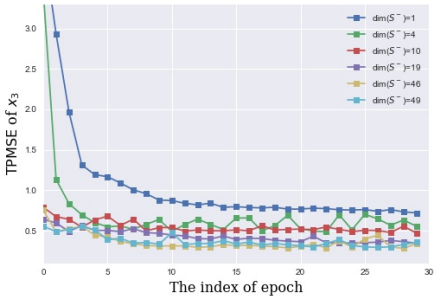
$$\mathrm{d}y_4(t) = x_4^3(t)\,\mathrm{d}t + \mathrm{d}w_4(t) \tag{5-14}$$

如图5.6(a)所示为在上述四维 Cubic Sensor 算例中对 x_1 估计的 TPMSE 随着训练世代的变化，如图5.6(b)所示为相应的对 x_2 的估计的 TPMSE 随着训练世代的变化，如图5.7(a)所示为相应的对 x_3 的估计随着训练世代的变化，如图5.7(b)所示为相应的对 x_4 的估计的 TPMSE 随着训练世代的变化。同样可以看到当神经网络结构中 S^- 的维数上升到大致十几维再上升时，滤波效果不再有较为明显的改善。

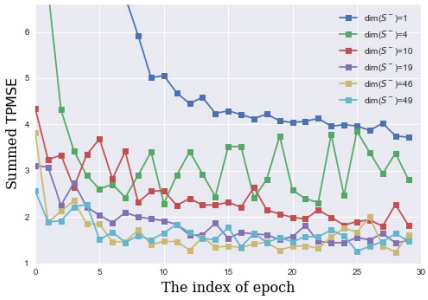


(a) 对 x_1 的估计　　(b) 对 x_2 的估计

图 5.6　四维 Cubic Sensor 算例中对各维变量估计的 TPMSE 随着训练世代的变化



(a) 对 x_3 的估计



(b) 对 x_4 的估计

图 5.7　四维 Cubic Sensor 算例中对各维变量估计的 TPMSE 随着训练世代的变化 (续)

第 6 章　基于深度学习的方法对于问题维数的敏感性分析

基于谱方法的滤波算法的复杂度会随着问题的维数增长而呈现指数上升的趋势，从而出现所谓 “维数诅咒”[12]。下面将要看到基于深度学习的算法相对于传统的谱方法而言在一类特殊的问题，也即 Cubic Sensor 的问题上表现出了对问题维数的不敏感性。

首先考虑到谱方法中计算的时间和存储复杂度和谱方法中采用的基的个数基本成正比，考虑在本文4.3中使用的谱方法的例子，在每一维上使用的基的个数为 15 个，则在 d 维条件下，需要的基的总个数为 15^d，基的总个数随着问题维数的变化如图6.1。因而可以发现谱方法在滤波问题中表现出的对于问题维数的高度敏感性。

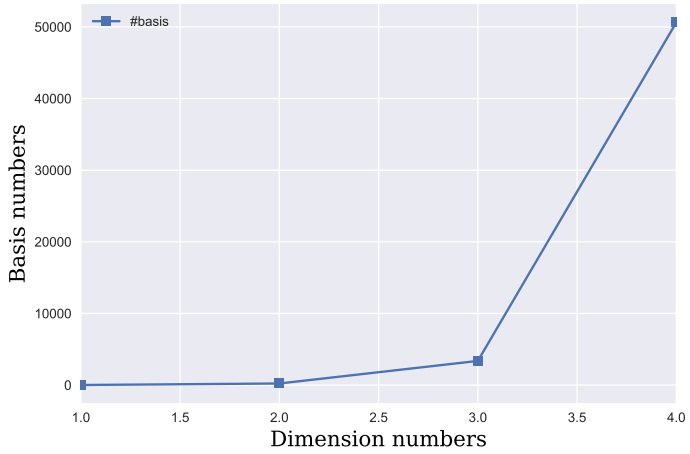


图 6.1　Cubic Sensor 算例中基的总个数随着问题维数的变化

本文接下来考察基于深度学习的滤波方法对于问题维数的敏感性，首先本文根据之前做的实验的结果，可以得到如表6.1所示的不同问题维数下的滤波性能大致收敛时状态表示向量的维数。

同时结合各自的网络结构，可以得到不同维数下滤波性能大致收敛时神经网络结构中神经元的数量以及网络中的边的数量的变化如表6.2。

可以很明显地看到，无论是神经元数量还是网络中边的数量，基于深度学

问题维数	1	2	3	4
大致收敛时的 dim(S^-)	2	9	13	19

表 6.1　滤波性能大致收敛时状态表示向量的维数与问题维数的关系

问题维数	1	2	3	4
大致收敛时的神经元数量	3	64	66	96
大致收敛时网络中边的数量①	21	1487	1290	2721

表 6.2　神经元数量和神经元之间边的数量随问题维数的变化

习的方法在 Cubic Sensor 这类算例中都没有出现谱方法会有的 “维数诅咒”[12] 的问题，滤波性能大致收敛时神经元的数量和网络中边的数量对问题维数并不非常敏感，事实上 2,3 和 4 维算例所需要的神经元和边的数量是同一个数量级的。

第 7 章　结论

本文将深度学习方法引入到非线性滤波问题中。一方面，在理论上探讨了神经网络的表示能力，另一方面在一系列滤波算例，包括一维和二维 Cubic Sensor 算例以及不稳定的 Lorenz 算例中实现了基于深度学习的滤波方法。本文指出，在上述算例中，基于深度学习的方法在滤波性能和滤波算法复杂度方面都获得了和经典的谱方法更优或近似的效果。

本文接着重点研究了不同维度下基于深度学习的非线性滤波算法的滤波性能大致收敛时需要的神经元和网络中边的数量（近似为参数的数量）对问题维度的敏感性，并且在 Cubic Sensor 这一类算例中指出基于深度学习的非线性滤波算法并不具有与谱方法中“维数诅咒”[12] 相似的高度敏感性。

插图索引

表格索引

公式索引

参考文献

[1] Lecun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436.

[2] Kalman R E. A new approach to linear filtering and prediction problems[J]. Journal of Basic Engineering Transactions, 1960, 82: 35–45.

[3] Kalman R E, Bucy R S. New results in linear filtering and prediction theory[C]//Trans. ASME, Ser. D, J. Basic Eng. [S.l.: s.n.], 1961: 109.

[4] Luo X, Yau S S T. Complete real time solution of the general nonlinear filtering problem without memory[J]. IEEE Transactions on Automatic Control, 2013, 58(10): 2563–2578.

[5] Jazwinski, AndrewH. Stochastic processes and filtering theory[M]. [S.l.]: Academic Press, 1970: 1730–1730

[6] Ljung L. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems[J]. Automatic Control IEEE Transactions on, 1979, 24(1): 36–50.

[7] Ting-Ho L J. Synthetic approach to optimal filtering[J]. IEEE Transactions on Neural Networks, 2002, 5(5): 803–811.

[8] Duncan T E. Probability densities for diffusion processes with applications to nonlinear filtering theory and detection theory[R]. [S.l.]: STANFORD UNIV CA STANFORD ELECTRONICS LABS, 1967.

[9] Wikipedia contributors. Artificial neural network — Wikipedia, the free encyclopedia [EB/OL]. 2018. https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=840504565.

[10] Hornik K. Approximation capabilities of multilayer feedforward networks[J]. Neural Networks, 1991, 4(2): 251–257.

[11] Evensen G. Sequential data assimilation for nonlinear dynamics: The ensemble kalman filter [M]. [S.l.: s.n.], 2002.

[12] Han J, Jentzen A, Weinan E. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning[M]. [S.l.: s.n.], 2017.

致　谢

非常感谢指导教师丘成栋教授对本人的勉励和悉心的指导，没有他的指导，我无法完成这个毕业论文的写作。

感谢丘成栋教授控制组的陈秀琼学姐，和她的几次讨论给了我很大的帮助。感谢董文慧学姐和时骥学长的热情帮助。感谢罗雪师姐的热心指导。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料的调研阅读报告或书面翻译

Deep learning based nonlinear filtering survey report for final year project
（综合论文训练开题报告）

A.1　Background

Filtering is a very important estimation technique utilized in many fields such as signal processing, control, astronomy and finance. Generally, the filtering problem can be formulated mathematically as follows:

$$dx_t = f(x_t, t)dt + G(x_t, t)dv_t$$
$$dy_t = h(x_t, t)dt + dw_t,$$

where x_t is $n-$dimensional system state vector, f is a $n-$vector valued function of x_t and time t, $G(x_t, t) \in R^{n\times r}$ is a matrix dependent on x_t and time t, $h(x_t, t)$ is an $m-$vector dependent on x_t and time t, v_t is an $r-$dimensional Brownian motion process with $E[dv_t dv_t^T] = Q(t)dt$ and w_t is an $m-$dimensional Brownian motion process with $E[dw_t dw_t^T] = S(t)dt$.

A.2　Existing work

As stated in ,optimal filtering method actually dates back to the least-square algorithm developed by Gauss and Legendre. Over the past several decades, there have been tremendous work on solving filtering problem. And in the 1940s, Kolmogorov and Wiener developed the so-called linear-minimum-variance filters and paved the way for the celebrated Kalman filter. However, Kalman filter needs the linear assumption and Gaussian assumption which means the equation of signal and measurment should be linear and the noise should be Gaussian noise. However, the applications are nonlinear. So linearization at the prediction value is applied to continue using Kalman filter

and that is called extended Kalman filter or EKF in short. The question is that EKF sometimes can very unstable. So much work has be done to find a better nonlinear filtering method.

A.2.1 Kalman filter and its extensions

A.2.1.1 Kalman filter

Kalman filter, with another name linear quadratic estimation(LQE), uses a stream of observation data to update the state variable's estimate. Kalman filter has been widely used in technology including fields like signal processing and robotics. Kalman filtering has two steps, one is prediction step, the other is update step. As the formulation in , the Kalman filter tried to estimate the state $x \in \mathbb{R}^n$, where x is a controlled stochastic process which satisfies a linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}, \tag{A-1}$$

where u_k is an optional control term input to the system. A and B are matrices with compatible dimensions. To estimate the hidden state x_k at discrete time slot k, we also have an observation process observing x as shown in equation (A-2).

$$z_k = Hx_k + v_k \tag{A-2}$$

Kalman filter uses a linear combination of the priori estimate of the state, $\hat{x}_k^-$ and a weighted difference between the real mesurement and the predicted measurement $H\hat{x}_k^-$. Thus we get the Kalman filter estimate equation (A-3).

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-), \tag{A-3}$$

where K is a linear weighting term and can be computed by:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}, \tag{A-4}$$

where P_k^- is the priori covariance matrix of the estimation and R is the observation noise's covariance matrix.

A.2.1.2 Extended Kalman Filter

Kalman filter is very powerful in reality. Unluckily, the state $x \in \mathbb{R}^n$ often satisfied nonlinear stochastic difference equation as following.

$$x_k = f(x_{k-1}, u_k, w_{k-1}), \tag{A-5}$$

where x is the state of the system to be estimated, u is the control imposed on the system and w is the system noise. And the observation model can also be nonlinear as the following equation.

$$z_k = h(x_k, v_k) \tag{A-6}$$

In nonlinear case, to continue using the techniques of Kalman filter, we need to do local linearization. First, we define

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0)$$
$$\tilde{z}_k = h(\tilde{x}_k, 0)$$

We do Taylor expansion for equation (A-5) and equation (A-6) at the point $(\hat{x}_{k-1}, u_k, 0)$ and $(\tilde{x}_k, 0)$ respectively. Then we can get equation (A.2.1.2).

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}$$
$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k,$$

where A, W, H, V are corresponding Jacobian matrices. After linearization, we can apply the standard Kalman filter technique.

A.2.2 Partial Differential Equation Approach

Let's consider a continuous time filtering problem:

$$\begin{aligned} dx(t) &= f(x(t))dt + g(x(t))dv(t), & x(0) &= x_0 \\ dy(t) &= h(x(t))dt + dw(t) & y(0) &= 0, \end{aligned}$$

where $x(t), y(t), v(t), w(t)$ are $\mathcal{R}^n, \mathcal{R}^m, \mathcal{R}^p, \mathcal{R}^m$ valued processes, respectively and $v(t), w(t)$ are independent, standard Brownian processes. The unnormalized density function $\sigma(x,t)$ of x_t conditioned on the observation history $Y_t = \{y_s : 0 \leqslant s \leqslant t\}$ satisfies the DMZ equation (A.2.2),

$$\begin{aligned} d\sigma(x,t) &= L\sigma(x,t)dt + \sigma(x,t)h^T(x,t)S^{-1}(t)dy_t \\ \sigma(x,0) &= \sigma_0(x), \end{aligned}$$

where $\sigma_0(x)$ is the probability density of the initial state x_0, and

$$L(*) := \frac{1}{2}\sum_{k,l=1}^{n} \frac{\partial^2}{\partial x_k \partial x_l}[(GQG^T)_{kl}*] - \sum_{k=1}^{n} \frac{\partial (f_k *)}{\partial x_k} \tag{A-7}$$

DMZ equation is a stochastic partial differential equation which is difficult to solve directly. Thus we first make an exponential transformation,

$$\sigma(x,t) = \exp[h^T(x,t)S^{-1}(x,t)y_t]\rho(x,t) \tag{A-8}$$

After exponential transformation, we can get a deterministic partial differential equation(PDE) with stochastic coefficients. We refer it as "pathwise-robust" DMZ equation.

$$\begin{aligned} &\frac{\partial \rho}{\partial t}(x,t) + \frac{\partial}{\partial t}(h^T S^{-1})^T y_t \rho(x,t) \\ &= \exp(-h^T S^{-1} y_t)[L - \frac{1}{2}h^T S^{-1} h] \cdot [\exp(h^T S^{-1} y_t)\rho(x,t)] \\ &\rho(x,0) = \sigma_0(x) \end{aligned}$$

Or we can write the above equation in a more compact form as follows.

$$\frac{\partial \rho}{\partial t}(x,t) = \frac{1}{2} D_w^2 \rho(x,t) + F(x,t) \cdot \nabla \rho(x,t) + J(x,t)\rho(x,t)$$
$$\rho(x,0) = \sigma_0(x),$$

where

$$\begin{aligned}
D_w^2 =& \sum_{i,j=1}^{n} (GQG^T)_{ij} \frac{\partial^2}{\partial x_i \partial x_j}, \\
F(x,t) =& [\sum_{j=1}^{n} \frac{\partial}{\partial x_j}(GQG^T)_{ij} + \sum_{j=1}^{n} (GQG^T)_{ij} \frac{\partial K}{\partial x_j} - f_i]_{i=1}^{n}, \\
J(x,t) =& -\frac{\partial}{\partial t}(h^T S^{-1})^T y_t + \frac{1}{2} \sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}(GQG^T)_{ij} \\
&+ \sum_{i,j=1}^{n} \frac{\partial}{\partial x_i}(GQG^T)_{ij} \frac{\partial K}{\partial x_j} \\
&+ \frac{1}{2} \sum_{i,j=1}^{n} (GQG^T)_{ij} [\frac{\partial^2 K}{\partial x_i \partial x_j} + \frac{\partial K}{\partial x_i} \frac{\partial K}{\partial x_j}] \\
&- \sum_{i=1}^{n} \frac{\partial f_i}{\partial x_i} - \sum_{i=1}^{n} f_i \frac{\partial K}{\partial x_i} - \frac{1}{2}(h^T S^{-1} h),
\end{aligned}$$

in which

$$K(x,t) = h^T(x,t) S^{-1}(t) y_t \tag{A-9}$$

Suppose we sample the observations at time sequence $0 = \tau_0 < \tau_1 < \ldots < \tau_k = T$.We divide the time interval $[0, T]$ into k small intervals.Let ρ_i be the solution of the robust DMZ equation with $y_t = y_{\tau_{i-1}}$ on the interval $\tau_{i-1} \leqslant t \leqslant \tau_i$, then ρ_i satisfies the following equation.

$$\frac{\partial \rho_i}{\partial t}(x,t) + \frac{\partial}{\partial t}(h^T S^{-1})^T y_{\tau_{i-1}} \rho_i(x,t)$$

$$= \exp(-h^T S^{-1} y_{\tau_{i-1}})[L - \frac{1}{2} h^T S^{-1} h] \cdot [\exp(h^T S^{-1} y_{\tau_{i-1}})\rho_i(x,t)]$$

$\rho_1(x,0) = \sigma_0(x)$,

or

$\rho_i(x,\tau_{i-1}) = \rho_{i-1}(x,\tau_{i-1})$, for $i = 2,3,\cdots,k$.

As given in , we have the following proposition.

Proposition 1 : For each $\tau_{i-1} \leqslant t < \tau_i, i = 1,2,\cdots,k, \rho_i(x,t)$ satisfies (A.2.2) if and only if

$$u_i(x,t) = \exp[h^T(x,t)S^{-1}(t)y_{\tau_{i-1}}], \tag{A-10}$$

satisfies the Kolmogorov forward equation(KFE)

$$\frac{\partial u_i}{\partial t}(x,t) = (L - \frac{1}{2} h^T S^{-1} h)u_i(x,t), \tag{A-11}$$

where L is defined in (A-7).

Recursively, the initial condition of (A-11) for $\tau_{i-1} \leqslant t \leqslant \tau_i$ is

$$u_i(x,\tau_{i-1}) = \exp[h^T(x,\tau_{i-1})S^{-1}(\tau_{i-1})(y_{\tau_{i-1}} - y_{\tau_{i-2}})] \cdot u_{i-1}(x,\tau_{i-1}) \tag{A-12}$$

for $i = 2,3,\cdots,k$, where $u_{i-1}(x,\tau_{i-1}) = \sum_{l=1}^{\infty} \hat{u}_{i-2,l}[U(\tau_{i-1},\tau_{i-2})\phi_l(x)]$. Here, $U(\tau_{i-1},\tau_{i-2}) : L^2(R^n) \mapsto L^2(R^n)$ is an operator defined by the Kolmogrov equation (A-11) that transforms an initial condition $u_i(x,\tau_{i-1})$ to the condition $u_i(x,t)$ at time t.

A.2.3 Deep Learning a PDE

Jiequn Han, etc. recently uses the techniques of deep learning to solve the high dimensional partial differential equation. They consider semilinear parabolic PDEs.

These PDEs can be represented as in equation

$$\frac{\partial u}{\partial t}(t,x)+\frac{1}{2}\mathrm{Tr}(\sigma\sigma^T(t,x)(\mathrm{Hess}_x u)(t,x))+\nabla u(t,x)\cdot\mu(t,x)+f(t,x,u(t,x),\sigma^T(t,x)\nabla u(t,x))=0 \tag{A-13}$$

with some terminal condition $u(T,x)=g(x)$. We want to solve u at $t=0, x=\xi$ for some vector $\xi\in\mathbb{R}^d$. As stated in , if we let $\{W_t\}_{t\in[0,T]}$ be a d-dimensional Brownian motion and $\{X_t\}_{t\in[0,T]}$ be a d-dimensional stochastic process which satisfies

$$X_t=\xi+\int_0^t\mu(s,X_s)ds+\int_0^t\sigma(s,X_s)dW_s \tag{A-14}$$

Then equation (A-13)'s solution satisfies the BSDE(backward stochastic differential equation):

$$u(t,X_t)=u(0,X_0)-\int_0^t f(s,X_s,u(s,X_s),\sigma^T(s,X_s)\nabla u(s,X_s))ds+\int_0^t[\nabla u(s,X_s)]^T\sigma(s,X_s)dW_s \tag{A-15}$$

After dicretizing the time interval $[0,T]$ to $0=t_0<t_1<\cdots<t_N=T$, the discretized case is derived:

$$X_{t_{n+1}}\approx X_{t_n}+\mu(t_n,X_{t_n})(t_{n+1}-t_n)+\sigma(t_n,X_{t_n})(W_{t_{n+1}}-W_{t_n}) \tag{A-16}$$

and

$$\begin{aligned}u(t_{n+1},X_{t_{n+1}})\approx& u(t_n,X_{t_n})-f(t_n,X_{t_n},u(t_n,X_{t_n}),\sigma^T(t_n,X_{t_n})\nabla u(t_n,X_{t_n}))(t_{n+1}-t_n)\\&+[\nabla u(t_n,X_{t_n})]^T\sigma(t_n,X_{t_n})(W_{t_{n+1}}-W_{t_n})\end{aligned}$$

With the continuous time discretized, next step is to use a multilayer feedforward network to approximate the function $x\mapsto\sigma^T(t,x)\nabla u(t,x)$ at each time step $t=t_n$.

$$\sigma^T(t_n,X_{t_n})\nabla u(t_n,X_{t_n})=(\sigma^T\nabla u)(t_n,X_{t_n})\cong(\sigma^T\nabla u)(t_n,X_{t_n}|\theta_n), \tag{A-17}$$

where θ_n denotes parameters of the neural network. After stacking all the neural network, we can input the sampled data $\{X_{t_n}\}_{0\leqslant n\leqslant N}$ and $\{W_{t_n}\}_{0\leqslant n\leqslant N}$ and get the output

$\hat{u}$ to estimate $u(t_n, X_)$. And we can use stochastic gradient descent-type(SGD) algorithm to train the parameters to minimize the loss function $l(\theta) = \mathbb{E}[|g(X_{t_N}) - \hat{u}|^2]$.

A.2.4 Neural network synthetic approach

The above methods all need mathematical equations to do filtering. However, James Ting-Ho Lo uses recurrent multilayer perceptron(RMLP) to do optimal filtering. James Ting-Ho Lo considers the following model:

$$x(t+1) = f(x(t)) + G(x(t))w(t)$$
$$y(t) = h(x(t)) + v(t)$$

James Ting-Ho Lo etc. gave RMLP to synthesize a filter and used data generated by computer simulation or experiment to train the RMLP's parameters and use the RMLP to do filtering. As shown in Figure A-1, the RMLP neural filter has input $y_i(t)$, output $\alpha_i(t)$, input to neuron i $\eta_i(t)$ and activation level $\beta_i(t)$. And let w^1_{ji} be the weight from the ith input to neuron j, w^2_{ji} be the weight from the ith neuron to the jth output, w^r_{ji} be the weight for the lagged feedback from ith neuron to jth neuron. Those parameters are all to be optimized. And the update formulas are:

$$\begin{aligned}
\beta_j(t) &= a(\eta_j(t)) \\
\eta_j(t) &= w^1_{j0} + \sum_{i=1}^{m} w^1_{ji} y_i(t) + \sum_{i=1}^{q} w^r_{ji} \beta_i(t-1) \\
\alpha_i(t) &= w^2_{i0} + \sum_{j=1}^{q} w^2_{ij} \beta_j(t),
\end{aligned}$$

where $a(x)$ is an activation function like $\tanh(x)$ and $\max\{0, x\}$.

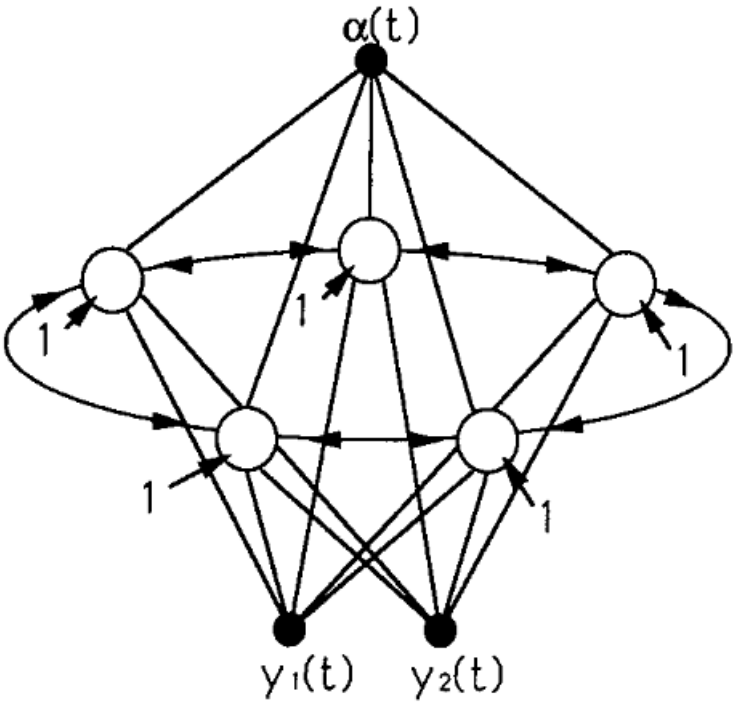


图 A-1　Neural Filter example

A.3　Research motivation

Deep learning has made a very impressive success in many practical fields such as computer vision and natural language processing, etc. However, there lacks a sound theoretical foundations for deep learning. On the other hand, a large amount of work has been done concerning the theory of nonlinear filtering from a mathematical perspective of view but due to the curse of dimensionality, it is difficult to design a real time filter in high-dimensional filtering case. So there are two questions to be answered in this project:

1. Can we utilize the power of deep learning to overcome the curse of dimensionality in high dimensional filtering problem?
2. Can we interpret deep learning through filtering theory?

A.4 Aims and methods

In this final year project, the aim is to utilize the power of deep learning to overcome the curse of dimensionality in high dimensional filtering problem and interpret deep learning through filtering theory. The potential methods or tools include:

1. Recurrent neural network(RNN) for learning a filter.
2. PyTorch as a deep learning framework.
3. Filtering theory based on stochastic differential equation and partial differential equation for interpretation of deep learning.

Actually, in low dimensional case, a cubic sensor filter based on deep learning has been implemented. The cubic sensor model is as follows:

$$dx_t = dv_t$$
$$dy_t = x_t^3 dt + dw_t$$

Set the sampling time interval to be 0.01s. The filtering result is shown in figure A-2. And the mean square is approximately 0.28 and the update time is approximately

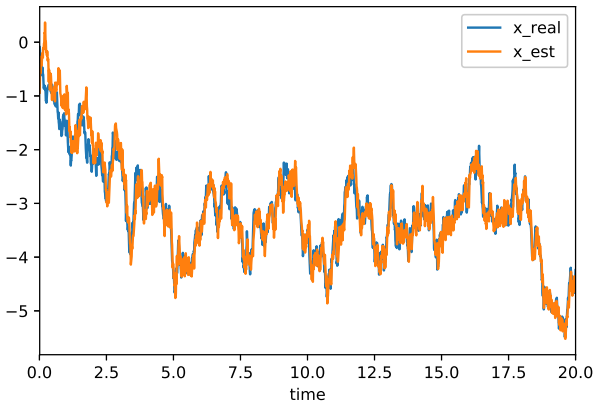


图 A-2 Cubic Sensor Tracking Using Deep Learning Based Filter

3×10^{-6} seconds.

A.5 Significance and future work plan

The potential result of this final year project can provide real time efficient high dimensional nonlinear filter. Also the potential result can provide theoretical insight for understanding deep learning.

The future work plan is as follows:

1. March-May: Design and implement the deep learning based nonlinear filtering algorithm both in low dimensional case and high dimensional case.
2. May: Theoretically justify the deep learning based nonlinear filtering.
3. May-June: Write the paper.

References

[1] X. Luo and S. S.-T. Yau, "Complete real time solution of the general non- linear filtering problem without memory," IEEE Transactions on Automatic Control, vol. 58, no. 10, pp. 2563–2578, 2013.

[2] J. T.-H. Lo, "Neural Filtering," Scholarpedia, vol. 4, no. 8, p. 7868, 2009, revision #91563.

[3] G. Welch and G. Bishop, An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, 2001.

[4] T. E. Duncan, "Probability densities for diffusion processes with applications to nonlinear filtering theory and detection theory," STANFORD UNIV CA STANFORD ELECTRONICS LABS, Tech. Rep., 1967.

[5] J. Han, A. Jentzen, and E. Weinan, "Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning," 2017. 8

[6] L. J. Ting-Ho, "Synthetic approach to optimal filtering," IEEE Transactions on Neural Networks, vol. 5, no. 5, pp. 803–811, 2002.