Deep networks for system identification: a survey

UPPSALA UNIVERSITET

Gianluigi Pillonetto[♣], Aleksandr Aravkin[♢], **Daniel Gedon**[♡], Lennart Ljung[♠], Antônio H. Ribeiro[♡], Thomas B. Schön[♡]

[♣]University of Padova, Italy
[♢]University of Washington, USA
[♡]Uppsala University, Sweden
[♠]Linköping University, Sweden

ERNSI Workshop 2023
Stockholm, September 26, 2023

arXiv link:

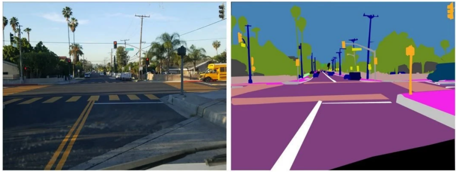
Introduction

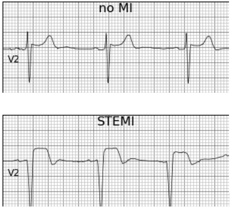
UPPSALA UNIVERSITET

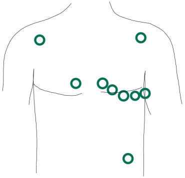
System identification with long history

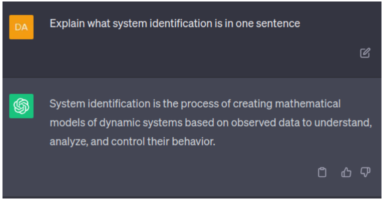
Introduction

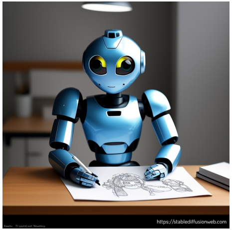
Deep neural networks with recent success











→ Innovate system identification with power of deep neural networks

Contents

Modeling of dynamical systems

Three main players:

1. Family of parameterized models

$$Z = \{x(t), y(t)\}_{t=1}^{\#train}$$
$$g_\theta : Z(t) \mapsto \hat{y}(t+1), \qquad \theta \in D_\theta$$

2. Parameter estimation method

$$\hat{\theta} = \arg\min_{\theta \in D_\theta} \mathcal{L}_N(\theta, Z_e)$$

3. Validation process
    - residual analysis
    - cross-validation

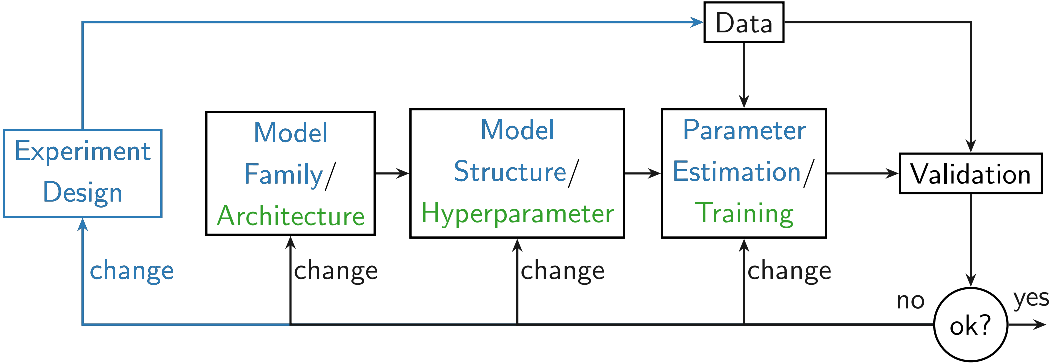
$\#features = \dim \theta$

$\mathcal{L}_{\text{emp}} = \mathcal{L}(\hat{\theta}, Z_e)$

overfitting $\mathcal{L}_{\text{emp}} = 0$ typically for $\#features = \#train$.

Modeling procedure:

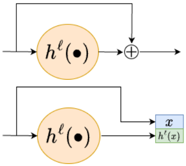
System identification vs deep learning

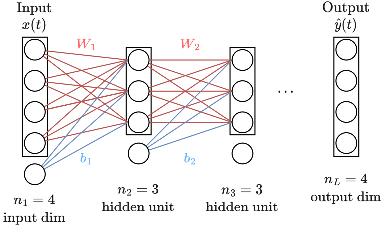


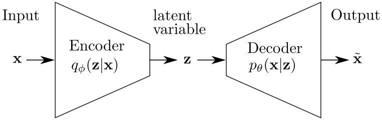
Contents

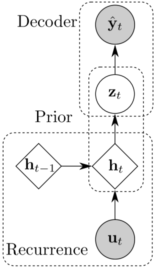
DNN architectures

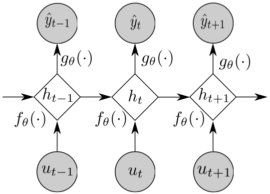
- Fully-connected networks
- Skip and direct connections
- **Convolutional networks**
- Recurrent neural networks
- Latent variable models
  - Autoencoder
  - Variational autoencoder
  - Deep state-space models
- **Energy-based models**









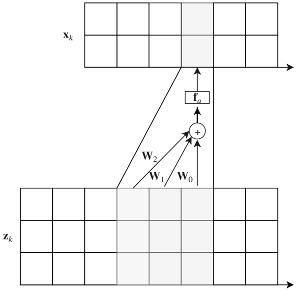


UPPSALA UNIVERSITET

Convolutional networks

Basic building block: convolutional layer

$$w(t) * z(t) = \sum_{j=0}^{k-1} w(j)^\top z(t-j)$$



Not just one filter but many: $W = \{w^1, \ldots, w^b\}$.

Then, ith output: $\quad x^i(t) = w^i(t) * z(t) \quad \text{for} \quad i = 1, \ldots, b$

Formulating regression problems Find predictive distribution $p(y(t)|x(t))$.

Example: NARX model

$$y(t) = f_\theta(x(t)) + e(t), \quad \text{with} \quad e(t) \sim \mathcal{N}(0, \sigma^2)$$

$\rightarrow$ Implicit assumption: $p(y(t)|x(t))$ is Gaussian $\rightarrow$ neural network models the mean.

Energy-based models

$$p_\theta(y(t) \mid x(t)) = \frac{e^{g_\theta(y(t),x(t))}}{Z_\theta(x(t))} \quad \text{with} \quad Z_\theta(x(t)) = \int e^{g_\theta(z,x(t))} dz$$

- Neural network mapping $g_\theta : (y(t), x(t)) \mapsto \mathbb{R}$
- Generalize implicit Gaussian assumption

$\rightarrow$ asymmetric, heavy-tailed, multimodal, ... distributions possible

Optimization

System identification:

$$\min_{\theta} \sum_{t=1}^{\#train} \mathcal{L}\Big(y(t), f_\theta\big(z(t)\big)\Big)$$

Deep learning:

$$\min_{\theta_1,\dots,\theta_L} \sum_{t=1}^{\#train} \mathcal{L}\Big(y(t), f^L_{\theta_L} \circ f^{L-1}_{\theta_{L-1}} \circ \cdots \circ f^1_{\theta_1}\big(z(t)\big)\Big)$$

Optimization: Newton's method $\mathcal{O}(\#train\#param^2 + \#param^3)$ $\lightning$

$\rightarrow$ first-order methods

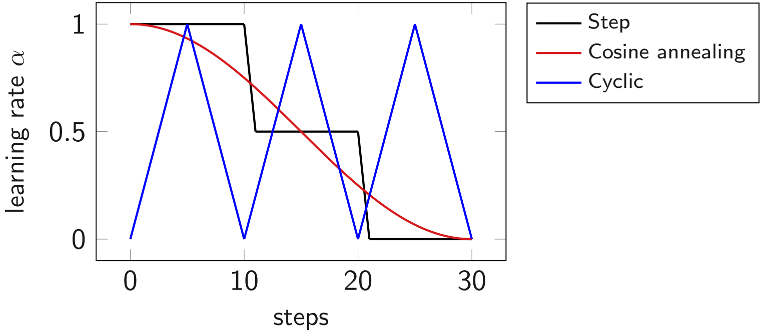
- Large dim(θ), nested structure $\rightarrow$ gradient w.r.t. each layer + chain rule
  $\rightarrow$ Backpropagation
- Large datasets $\rightarrow$ stochastic methods

Gradient decent optimization:

$$\theta^{i+1} = \theta^i - \alpha \nabla V(\theta^i) \qquad \text{with } \alpha \text{ as learning rate}$$

Stochastic gradient descent with fixed α does not converge $\lightning$

Solution: Learning rate scheduler $\rightarrow$ reduce α to zero



Contents

UPPSALA UNIVERSITET

Kernels for modeling dynamical systems

- Linear kernel

$$K(x_i, x_j) = x_i^\top P x_j \qquad \text{with positive semidefinite } P$$

  induces linear functions $f(x) = \theta^\top x$ $\rightarrow$ FIR models

- Linear kernel with $P_{ij} = \varphi^{\max(i,j)}$ with $0 \leq \varphi < 1$ $\rightarrow$ stable spline/TC kernel
- Gaussian kernel $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\rho}\right)$ with $\rho > 0$ $\rightarrow$ NFIR models

Choice of kernel $\rightarrow$ encode high level assumptions

Deep kernel-based learning

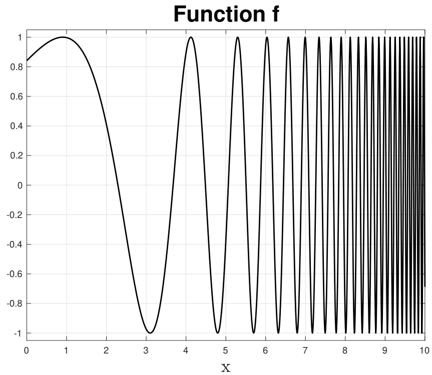
UPPSALA UNIVERSITET

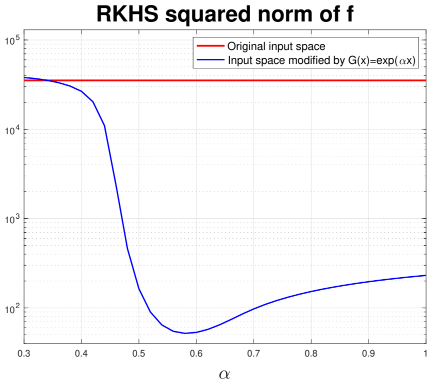
Example: $f = sin(e^{x/2})$ $\rightarrow$ complicated frequency content

- Gaussian kernel: high RKHS norm $\rightarrow$ biased estimator
- Idea: transform data $f = \tilde{f} \circ G$

$$x(t) \longrightarrow \boxed{G = e^{x/2}} \rightarrow \boxed{\quad \tilde{f} \quad} \longrightarrow y(t)$$

Choose $G = e^{x/2}$ $\rightarrow$ $\tilde{f} = sin(x)$ with single frequency





Deep kernel-based learning

Consider idea: $f = \tilde{f} \circ G$

$$x(t) \longrightarrow \boxed{\text{Neural Network}} \longrightarrow \boxed{\tilde{f}} \longrightarrow y(t)$$

$\rightarrow$ manifold Gaussian process with

$$K(x_i, x_j) := \tilde{K}(\tilde{x}_i, \tilde{x}_j) = \tilde{K}(G(x_i), G(x_j))$$

Previously: Gaussian kernel K with one scale parameter $\rho > 0$

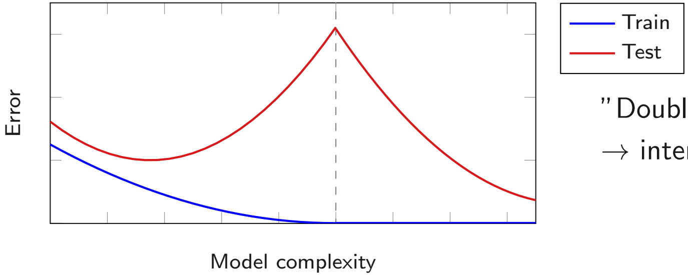
Now: Manifold Gaussian kernel K with many parameters $\eta = [\rho, \theta]$

$\rightarrow$ Optimize by marginal likelihood of joint density $p(Y, f|\eta)$

UPPSALA UNIVERSITET

Why are deep models so successful?

- 2-layer ConvNet on MNIST: 1.2m parameters vs 60k data points
- AlexNet on ImageNet: 62.3m parameters vs 1.2m data points
- ...



"Double descent"

$\rightarrow$ interpolate and generalize

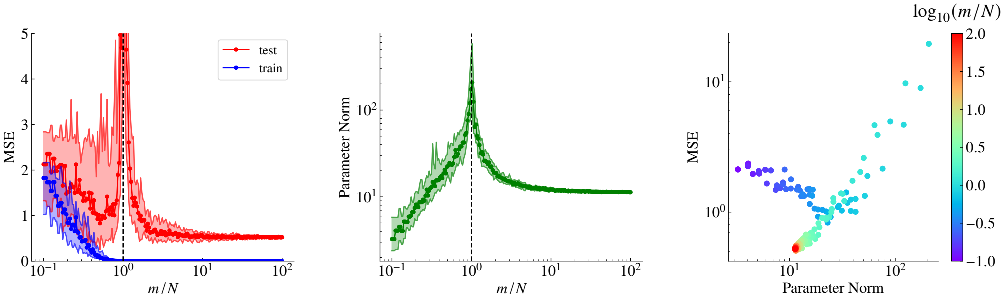
Theoretical development:

1. **interplay of overparameterization and generalization**
2. simplification of non-convex optimization problem

UPPSALA UNIVERSITET

System identification example:

- NARX model: $\hat{y}(t) = \sum_{i=1}^{\#features} \theta_i \phi_i(x(t))$
- Data from: $y(t) = f(x(t)) + v(t)$
- $\#train = 100$ samples
- 1-step ahead prediction



Theoretical development

- Nonlinear transformation $\phi(x)$, input to feature space

$$\phi : \mathbb{R}^{\#inputs} \mapsto \mathbb{R}^{\#features}$$

- Linear model:

$$\hat{y} = \hat{\theta}^\top \phi(x)$$

- Estimation procedure:

$$\min_{\theta} \sum_{i=1}^{\#train} (y_i - \hat{\theta}^\top \phi(x_i))^2$$

- Optimization procedure: Gradient descent starting from zero

$$\theta^{i+1} = \theta^i - \alpha \nabla V(\theta^i)$$

Theoretical development

Solutions of a linear system

$$X\theta = y$$

Three scenarios:

1. no solution if $\#features < \#train$
2. one unique solution if $\#features = \#train$
3. multiple solution if $\#features > \#train$

Gradient descent:

$$\min_{\theta} \|\theta\|_2 \quad \text{subject to} \quad X\theta = y$$

converges to the minimum-norm solution

$\rightarrow$ Implicit regularization of gradient descent

UPPSALA UNIVERSITET

Implicit Regularization

Gradient descent step: $\theta^{i+1} = \theta^i - \alpha \nabla V(\theta^i)$

$\rightarrow$ does not follow continuous gradient flow

Gradient descent follows more closely

$$\dot{\theta} = -\nabla \widetilde{V}(\theta)$$

with modified cost

$$\widetilde{V}(\theta) = V(\theta) + \lambda R(\theta)$$

$$\lambda = \frac{\alpha \ \#features}{4}, \quad R(\theta) = \frac{1}{\#features} \sum_{j=1}^{\#features} (\nabla_j V(\theta))^2$$

$\rightarrow$ gradient descent penalizes directions j with large cost $V(\theta)$

UPPSALA UNIVERSITET

2. Simplification of non-convex optimization problem

Setup:

- wide neural network with large $\theta \in \mathbb{R}^{\#features}$
- each update changes θ just by small amour

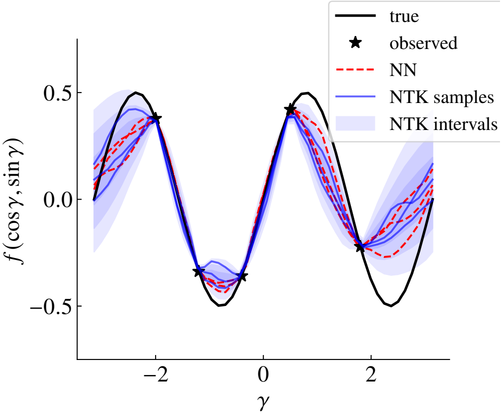
$\rightarrow$ linearize model around θ_0

$$f_\theta(x) \approx f_{\theta_0}(x) + \nabla f_{\theta_0}(x)^\top(\theta - \theta_0)$$

Neural tangent kernel

$$K(x, z; \theta_0) = \nabla f_{\theta_0}(x)^\top \nabla f_{\theta_0}(z)$$

$\rightarrow$ convex optimization problem



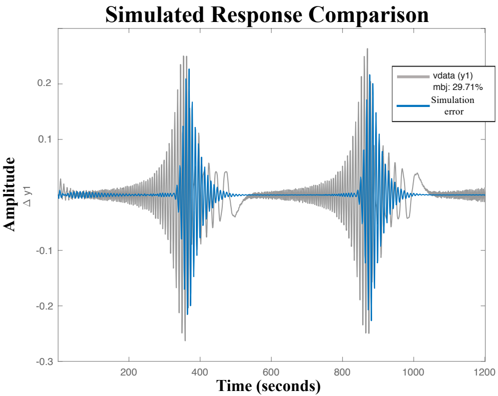
Contents

Applications

`Matlab` example: forced duffing oscillator (silverbox benchmark)

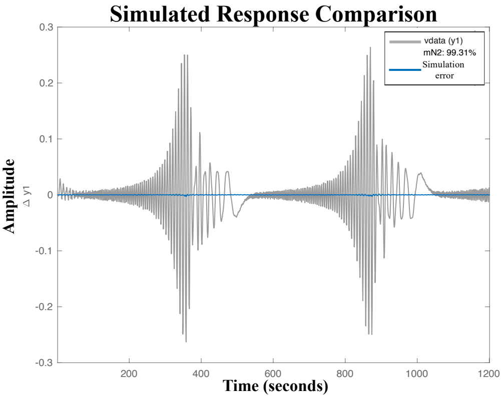
Linear Box-Jenkins type model

$\rightarrow$ Fit is 29.7%



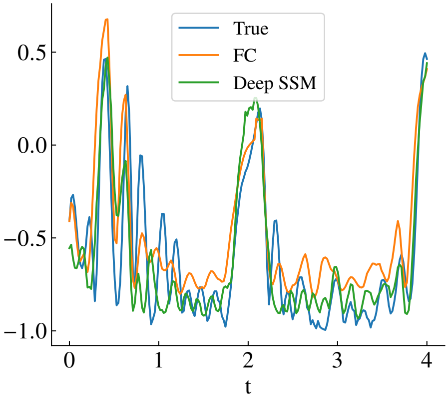
Cascaded feedforward network

$\rightarrow$ Fit is 99.2%



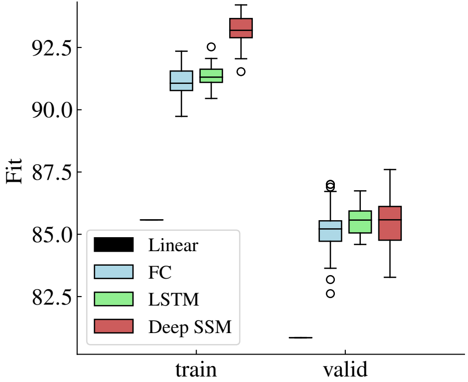
`Pytorch` example: Coupled electronic drives benchmark

- Basline: linear ARX model
- Feedforwad model
- LSTM
- Deep state-space model



Good fit of deep models despite $\#train = 300$

- $\dim(\theta_{FF}) = 184,200$
- $\dim(\theta_{LSTM}) = 169,801$
- $\dim(\theta_{DSSM}) = 111,902$



Conclusion

Essential for using neural networks:

- many parameters $\rightarrow$ overparameterization
- many layers $\rightarrow$ deep architectures

Open problems:

- Successful architectures:
  - Attention models and transformers
  - Flow-based models
  - Generative adversarial models (GANs) and diffusion models
  - Graph neural networks
- Robustness issues
- Theoretical development
- ...

Contact

Thank you!

Daniel Gedon, Uppsala University

E-mail: `daniel.gedon@it.uu.se`
Web: `dgedon.github.io`
Twitter: `@danigedon`

arXiv link: