Noisy Recurrent Neural Networks

Soon Hoe Lim
KTH

Ben Erichson
U Pittsburgh

Liam Hodgkinson
UC Berkeley

Michael Mahoney
UC Berkeley

35th Conference on Neural Information Processing Systems (NeurIPS 2021)

Recurrent Neural Networks (RNNs)

- Networks of neurons with *feedback connections* designed to deal with sequential data

Recurrent Neural Networks (RNNs)

Networks of neurons with feedback connectionsdesigned to deal with sequential data

Can use their hidden state (memory) to process variable length sequences of inputs

Recurrent Neural Networks (RNNs)

Networks of neurons with feedback connectionsdesigned to deal with sequential data

Can use their hidden state (memory) to process variable length sequences of inputs

Universal approximators of dynamical systems

Recurrent Neural Networks (RNNs)

- Networks of neurons with feedback connectionsdesigned to deal with sequential data
- Can use their hidden state (memory) to process variable length sequences of inputs
- Universal approximators of dynamical systems

A Supervised Learning Framework for RNNs

Data: $\{(x^{(i)}, c^{(i)})\}_{i=1,\dots,N}$

$x^{(i)} := (x_t^{(i)})_{t=0,1,\dots,T-1}$ = input sequence, $c^{(i)}$ = class label

A Supervised Learning Framework for RNNs

Data: $\{(x^{(i)}, c^{(i)})\}_{i=1,\dots,N}$

$x^{(i)} := (x_t^{(i)})_{t=0,1,\dots,T-1}$ = input sequence, $c^{(i)}$ = class label

Model (RNN): parametric non-autonomous discrete-time dynamical system

$$h_{t+1}^{(i)} = a(h_t^{(i)}, x_t^{(i)}), \quad t = 0, \dots, T-1, \tag{1}$$

$$y_T^{(i)} = g(h_T^{(i)}) \tag{2}$$

h = hidden state, y = output variable

A Supervised Learning Framework for RNNs

Data: $\{(x^{(i)}, c^{(i)})\}_{i=1,\dots,N}$

$x^{(i)} := (x_t^{(i)})_{t=0,1,\dots,T-1}$ = input sequence, $c^{(i)}$ = class label

Model (RNN): parametric non-autonomous discrete-time dynamical system

$$h_{t+1}^{(i)} = a(h_t^{(i)}, x_t^{(i)}), \quad t = 0, \dots, T-1, \tag{1}$$

$$y_T^{(i)} = g(h_T^{(i)}) \tag{2}$$

h = hidden state, y = output variable

– Example (vanilla):

$$a(h, x) = \tanh(W_h h + W_x x + b), \quad g(h) = W_y h$$

$:= (W_h, W_x, b, W_y)$ = learnable parameters

A Supervised Learning Framework for RNNs

Data: $\{(x^{(i)}, c^{(i)})\}_{i=1,\dots,N}$

$x^{(i)} := (x_t^{(i)})_{t=0,1,\dots,T-1}$ = input sequence, $c^{(i)}$ = class label

Model (RNN): parametric non-autonomous discrete-time dynamical system

$$h_{t+1}^{(i)} = a(h_t^{(i)}, x_t^{(i)}), \quad t = 0, \dots, T-1, \tag{1}$$

$$y_T^{(i)} = g(h_T^{(i)}) \tag{2}$$

h = hidden state, y = output variable

– Example (vanilla):

$$a(h, x) = \tanh(W_h h + W_x x + b), \quad g(h) = W_y h$$

$\theta := (W_h, W_x, b, W_y)$ = learnable parameters

Loss: $L(\theta) = \frac{1}{N} \sum_{i=1}^{N} l(y_T^{(i)}, c^{(i)})$

A Supervised Learning Framework for RNNs

Data: $\{(x^{(i)}, c^{(i)})\}_{i=1,\dots,N}$

$x^{(i)} := (x_t^{(i)})_{t=0,1,\dots,T-1}$ = input sequence, $c^{(i)}$ = class label

Model (RNN): parametric non-autonomous discrete-time dynamical system

$$h_{t+1}^{(i)} = a(h_t^{(i)}, x_t^{(i)}), \quad t = 0, \dots, T-1, \tag{1}$$

$$y_T^{(i)} = g(h_T^{(i)}) \tag{2}$$

h = hidden state, y = output variable

– Example (vanilla):

$$a(h, x) = \tanh(W_h h + W_x x + b), \quad g(h) = W_y h$$

$\theta := (W_h, W_x, b, W_y)$ = learnable parameters

Loss: $L(\theta) = \frac{1}{N} \sum_{i=1}^{N} l(y_T^{(i)}, c^{(i)})$

Optimization: $\theta^* = \arg\min_\theta L(\theta)$

From Good Old RNNs to SDEs

Two steps:

From Good Old RNNs to SDEs

Two steps:

(1) Adding leaky integrator:

$$h_{t+1} = h_t + a(h_t; x_t) \tag{3}$$

From Good Old RNNs to SDEs

Two steps:

(1) Adding leaky integrator:

$$h_{t+1} = h_t + a(h_t; x_t) \tag{3}$$

(2) Injecting noise:

$$h_{t+1} = h_t + a(h_t; x_t) + {}_t; \quad ; ; > 0; \tag{4}$$

where the $_t$ are i.i.d. random vectors (e.g., zero mean Gaussian)

From Good Old RNNs to SDEs

Two steps:

(1) Adding leaky integrator:

$$h_{t+1} = \quad h_t + \quad a(h_t; x_t) \tag{3}$$

(2) Injecting noise:

$$h_{t+1} = \quad h_t + \quad a(h_t; x_t) + \quad {}_t; \quad ; ; > \quad 0; \tag{4}$$

where the ${}_t$ are i.i.d. random vectors (e.g., zero mean Gaussian)

SDE intepretation. Setting $= 1 \quad t, \quad = \quad t, \quad = \sqrt{\quad t}$ and ${}_t$ = i.i.d. standard Gaussian, we see that the resulting eq. (4) is the Euler-Mayurama approximation of the following SDE:

$$\boxed{dh_t = \quad h_t\,dt + a(h_t; x_t)dt + \quad dB_t; \quad t \; 2 \; [0; T];} \tag{5}$$

where $(B_t)_{t \quad 0}$ is a Brownian motion (continuous-time process with independent Gaussian increments)

Noisy Recurrent Neural Networks (NRNNs)

More generally, we consider the following (Itô) SDE model for RNN.

Let $x \in C([0, T]; \mathbb{R}^{d_x})$ be an input signal.

Continuous-Time NRNNs

$$dh_t = f(h_t, x_t)dt + \sigma(h_t, x_t)dB_t, \qquad y_t = Vh_t, \tag{6}$$

where $\sigma : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h \times r}$ and $(B_t)_{t \geq 0}$ is an r-dimensional Brownian motion.

Noisy Recurrent Neural Networks (NRNNs)

More generally, we consider the following (Itô) SDE model for RNN.

Let $x \in C([0,T];\mathbb{R}^{d_x})$ be an input signal.

Continuous-Time NRNNs

$$dh_t = f(h_t, x_t)dt + \sigma(h_t, x_t)dB_t, \qquad y_t = Vh_t, \tag{6}$$

where $\sigma : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h \times r}$ and $(B_t)_{t \geq 0}$ is an r-dimensional Brownian motion.

The functions f and σ are referred to as the *drift* and *diffusion coefficients*, respectively.

Noisy Recurrent Neural Networks (NRNNs)

More generally, we consider the following (Itô) SDE model for RNN.

Let $x \; 2 \; C([0;T];R^{d_x})$ be an input signal.

Continuous-Time NRNNs

$$dh_t = f(h_t;x_t)dt + \quad (h_t;x_t)dB_t; \qquad y_t = Vh_t; \tag{6}$$

where $\; : R^{d_h} \quad R^{d_x} \; ! \quad R^{d_h} \quad ^{r}$ and $(B_t)_{t \quad 0}$ is an r-dimensional Brownian motion.

The functions f and are referred to as the drift and di usion coe cients, respectively.

Intuitively, (6) amounts to a noisy perturbation of the corresponding deterministic CT-RNN

Noisy Recurrent Neural Networks (NRNNs)

More generally, we consider the following (Itô) SDE model for RNN.

Let $x \in C([0,T];\mathbb{R}^{d_x})$ be an input signal.

Continuous-Time NRNNs

$$dh_t = f(h_t, x_t)dt + \sigma(h_t, x_t)dB_t, \quad y_t = Vh_t, \tag{6}$$

where $\sigma : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h \times r}$ and $(B_t)_{t \geq 0}$ is an r-dimensional Brownian motion.

The functions f and σ are referred to as the *drift* and *diffusion* coefficients, respectively.

- Intuitively, (6) amounts to a noisy perturbation of the corresponding deterministic CT-RNN
- To guarantee the existence of a unique solution to (6), in the sequel, we assume that $\{f(\cdot, x_t)\}_{t \in [0,T]}$ and $\{\sigma(\cdot, x_t)\}_{t \in [0,T]}$ are uniformly Lipschitz continuous, and $t \mapsto f(h, x_t)$, $t \mapsto \sigma(h, x_t)$ are bounded in $t \in [0,T]$ for each fixed $h \in \mathbb{R}^{d_h}$

Bene ts of Continuous-Time Formulation

(Design) Sampling from these RNNs gives us discrete-time RNNs $\Rightarrow$ guided principle and exibility in designing RNN architectures

Benefits of Continuous-Time Formulation

(Design) Sampling from these RNNs gives us discrete-time RNNs $\Rightarrow$ guided principle and flexibility in designing RNN architectures

(Modeling) In situations where the input data are generated by continuous-time dynamical systems, it is desirable to consider learning models which are also continuous in time

Bene ts of Continuous-Time Formulation

(Design) Sampling from these RNNs gives us discrete-time RNNs $\Rightarrow$ guided principle and exibility in designing RNN architectures

(Modeling) In situations where the input data are generated by continuous-time dynamical systems, it is desirable to consider learning models which are also continuous in time

(Analysis) A rich set of tools and techniques from the continuous-time theory can be borrowed to simplify analysis and to gain useful insights

Choice of Drift and Diffusion Coefficient

$$f(h, x) = Ah + a(Wh + Ux + b), \tag{7}$$

where $a : \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous scalar activation function extended to act on vectors pointwise, $A, W \in \mathbb{R}^{d_h \times d_h}$, $U \in \mathbb{R}^{d_h \times d_x}$ and $b \in \mathbb{R}^{d_h}$

Drift = a linear component + a Lipschitz nonlinearity

Choice of Drift and Di usion Coe cient

$$f(h; x) = Ah + a(Wh + Ux + b); \tag{7}$$

where $a : \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous scalar activation function extended to act on vectors pointwise, $A; W \in \mathbb{R}^{d_h \times d_h}$, $U \in \mathbb{R}^{d_h \times d_x}$ and $b \in \mathbb{R}^{d_h}$

$\boxed{\text{Drift = a linear component + a Lipschitz nonlinearity}}$

$$(h; x) = ({}_1 I + {}_2 \mathrm{diag}(f(h; x))); \tag{8}$$

where > 0 is small, and ${}_1 \geq 0$ and ${}_2 \geq 0$ are tunable parameters

$\boxed{\text{Di usion = additive + a multiplicative noise}}$

Choice of Drift and Di usion Coe cient

$$f(h;x) = Ah + a(Wh + Ux + b); \tag{7}$$

where $a : R \to R$ is a Lipschitz continuous scalar activation function extended to act on vectors pointwise, $A; W \in R^{d_h \times d_h}$, $U \in R^{d_h \times d_x}$ and $b \in R^{d_h}$

$\boxed{\text{Drift = a linear component + a Lipschitz nonlinearity}}$

$$(h;x) = ({}_1 I + {}_2 \mathrm{diag}(f(h;x))); \tag{8}$$

where > 0 is small, and ${}_1 \quad 0$ and ${}_2 \quad 0$ are tunable parameters

$\boxed{\text{Di usion = additive + a multiplicative noise}}$

One can set $= 0$ at inference time

$\implies$ noise injections in NRNNs can be viewed as a stochastic learning strategy

From Continuous-Time to Discrete-Time NRNNs

We consider explicit Euler-Maruyama (E-M) integrators, which are the stochastic analogues of Euler-type integration schemes for ODEs.

From Continuous-Time to Discrete-Time NRNNs

We consider explicit Euler-Maruyama (E-M) integrators, which are the stochastic analogues of Euler-type integration schemes for ODEs.

Let $0 = t_0 < t_1 < \quad < t_M = T$ be a partition of the interval $[0 ; T]$. Denote $\ {}_m := t_{m+1} \quad t_m$ for each $m = 0 ; 1; : : : ; M \quad 1$, and $\ := (\ {}_m)$

The E-M scheme provides a family (parametrized by) of approximations to the solution of the SDE in (6):

Discrete-Time NRNNs

$$h_{m+1} = h_m + f(h_m ; \hat{x}_m) \ {}_m + \ (h_m ; \hat{x}_m) \sqrt{\ {}_m} \ {}_m ; \tag{9}$$

for $m = 0 ; 1; : : : ; M \quad 1$, where $(\hat{x}_m)_{m=0 ;:::; M \quad 1}$ is a given sequential data, the $\ {}_m \quad N \ (0; I)$ are independent r-dimensional standard normal random vectors, and $h_0 = h_0$

Main Results (Theory)

We study Noisy RNNs via the lens of implicit regularization and derive an explicit regularizer induced by the noise injection through a perturbation analysis in the small noise regime

Main Results (Theory)

We study Noisy RNNs via the lens of implicit regularization and derive an explicit regularizer induced by the noise injection through a perturbation analysis in the small noise regime

It turns out that this regularizer reduces the state-to-state Jacobians and Hessian of the loss function according to the noise level, thereby promoting atter minima and biasing towards models with more stable dynamics

Main Results (Theory)

- We study Noisy RNNs via the lens of implicit regularization and derive an explicit regularizer induced by the noise injection through a perturbation analysis in the small noise regime
- It turns out that this regularizer reduces the state-to-state Jacobians and Hessian of the loss function according to the noise level, thereby promoting atter minima and biasing towards models with more stable dynamics

Figure: Hessian loss landscapes for deterministic (left) and noisy (right) model

Main Results (Theory)

We study Noisy RNNs via the lens of implicit regularization and derive an explicit regularizer induced by the noise injection through a perturbation analysis in the small noise regime

It turns out that this regularizer reduces the state-to-state Jacobians and Hessian of the loss function according to the noise level, thereby promoting atter minima and biasing towards models with more stable dynamics

Figure: Hessian loss landscapes for deterministic (left) and noisy (right) model

We show that, in this small noise regime, NRNNs promote classi ers with large classi cation margin, an attribute linked to improved model robustness

Main Results (Theory)

We study Noisy RNNs via the lens of implicit regularization and derive an explicit regularizer induced by the noise injection through a perturbation analysis in the small noise regime

It turns out that this regularizer reduces the state-to-state Jacobians and Hessian of the loss function according to the noise level, thereby promoting atter minima and biasing towards models with more stable dynamics

Figure: Hessian loss landscapes for deterministic (left) and noisy (right) model

We show that, in this small noise regime, NRNNs promote classi ers with large classi cation margin, an attribute linked to improved model robustness

We also provide su cient conditions for stability of the SDE, showing that noise injection can improve stability during training

Main Results (Experiments)

We demonstrate on benchmark data sets that NRNN classi ers are more robust to data perturbations when compared to other recurrent models, while retaining SOTA performance for clean data

Main Results (Experiments)

We demonstrate on benchmark data sets that NRNN classi ers are more robust to data perturbations when compared to other recurrent models, while retaining SOTA performance for clean data

Table: Robustness w.r.t. white noise ( ) and S&P ( ) perturbations on the ordered MNIST task.

Name	clean	= 0 :1	= 0 :2	= 0 :3	= 0 :03	= 0 :05	= 0 :1
Antisymmetric RNN (Chang et. al., 2019)	97.5%	45.7%	22.3%	17.0%	77.1%	63.9%	42.6%
CoRNN (Rusch et. al., 2021)	99.1%	96.6%	61.9%	32.1%	95.6%	88.1%	58.9%
Exponential RNN (Lezcano et. al., 2019)	96.7%	86.7%	58.1%	33.3%	83.6%	70.7%	43.4%
Lipschitz RNN (Erichson et. al., 2020)	99.2%	98.4%	78.9%	47.1%	97.6%	93.4%	73.5%
NRNN (mult. noise: 0.02 / add. noise: 0.02)	99.1%	98.9%	88.4%	62.9%	98.3%	95.6%	78.7%
NRNN (mult. noise: 0.02 / add. noise: 0.05)	99.1%	98.9%	92.2%	73.5%	98.5%	97.1%	85.5%

Main Results (Experiments)

We demonstrate on benchmark data sets that NRNN classi ers are more robust to data perturbations when compared to other recurrent models, while retaining SOTA performance for clean data

Table: Robustness w.r.t. white noise ( ) and S&P ( ) perturbations on the ordered MNIST task.

Name	clean	= 0 :1	= 0 :2	= 0 :3	= 0 :03	= 0 :05	= 0 :1
Antisymmetric RNN (Chang et. al., 2019)	97.5%	45.7%	22.3%	17.0%	77.1%	63.9%	42.6%
CoRNN (Rusch et. al., 2021)	99.1%	96.6%	61.9%	32.1%	95.6%	88.1%	58.9%
Exponential RNN (Lezcano et. al., 2019)	96.7%	86.7%	58.1%	33.3%	83.6%	70.7%	43.4%
Lipschitz RNN (Erichson et. al., 2020)	99.2%	98.4%	78.9%	47.1%	97.6%	93.4%	73.5%
NRNN (mult. noise: 0.02 / add. noise: 0.02)	99.1%	98.9%	88.4%	62.9%	98.3%	95.6%	78.7%
NRNN (mult. noise: 0.02 / add. noise: 0.05)	99.1%	98.9%	92.2%	73.5%	98.5%	97.1%	85.5%

test accuracy

amount of noise

(a) White noise perturbations.

amount of noise

(b) Salt and pepper perturbations.

Figure: Test accuracy for the ordered MNIST task as a function of the strength of input perturbations.

Conclusion

Noise injection can be viewed as a stochastic learning strategy used to improve robustness of learning models against data perturbations

Conclusion

- Noise injection can be viewed as a stochastic learning strategy used to improve robustness of learning models against data perturbations
- Within the framework of SDEs, we study RNNs trained by injecting noise into the hidden states and the implicit regularization effects of general noise injection schemes, showing that noise injection promotes classifiers with large classification margin

Conclusion

- Noise injection can be viewed as a stochastic learning strategy used to improve robustness of learning models against data perturbations
- Within the framework of SDEs, we study RNNs trained by injecting noise into the hidden states and the implicit regularization effects of general noise injection schemes, showing that noise injection promotes classifiers with large classification margin
- Our empirical results are in agreement with our theory and its implications, finding that NRNN classifiers achieve superior robustness to input perturbations

Conclusion

- Noise injection can be viewed as a stochastic learning strategy used to improve robustness of learning models against data perturbations
- Within the framework of SDEs, we study RNNs trained by injecting noise into the hidden states and the implicit regularization effects of general noise injection schemes, showing that noise injection promotes classifiers with large classification margin
- Our empirical results are in agreement with our theory and its implications, finding that NRNN classifiers achieve superior robustness to input perturbations

References

Paper: arXiv:2102.04877
Code: `https://github.com/erichson/NoisyRNN`