Advances in Continuous and Discrete Models

RESEARCH Open Access

Path classification by stochastic linear recurrent neural networks

Youness Boutaib[1*], Wiebke Bartolomaeus[1], Sandra Nestler[2,3] and Holger Rauhut[1]

*Correspondence:
boutaib@mathc.rwth-aachen.de
[1]Chair for Mathematics of Information Processing, RWTH Aachen University, Pontdriesch 10, 52062, Aachen, Germany
Full list of author information is available at the end of the article

Abstract

We investigate the functioning of a classifying biological neural network from the perspective of statistical learning theory, modelled, in a simplified setting, as a continuous-time stochastic recurrent neural network (RNN) with the identity activation function. In the purely stochastic (robust) regime, we give a generalisation error bound that holds with high probability, thus showing that the empirical risk minimiser is the best-in-class hypothesis. We show that RNNs retain a partial signature of the paths they are fed as the unique information exploited for training and classification tasks. We argue that these RNNs are easy to train and robust and support these observations with numerical experiments on both synthetic and real data. We also show a trade-off phenomenon between accuracy and robustness.

MSC: Primary 68T07; 68Q32; secondary 92B20; 60L10

Keywords: Recurrent neural networks; Risk bounds; Agnostic PAC learnability; Empirical risk minimisation; Rademacher complexity; Signatures

1 Introduction

Recurrent neural networks (RNNs) constitute the simplest machine learning paradigm that is able to handle variable-length data sequences while tracking long-term dependencies and taking into account the temporal order of the received information. These data streams appear naturally in many fields such as (audio or video) signal processing or financial data. The RNN architecture is inspired from biological neural networks where both recurrent connectivity and stochasticity in the temporal dynamics are ubiquitous. Despite the empirical success of RNNs and their many variants (long short-term memory networks (LSTMs), gated recurrent units (GRUs), etc.), several fundamental mathematical questions related to the functioning of these networks remain open:

- What is the exact type of information that an RNN learns from the input sequences?
- Training artificial RNNs with classical methods like gradient descent suffers from fundamental problems such as instability, non-convergence, exploding gradient errors [10] and plateauing [27]. On the other hand, biological networks seem to be robust and easy to train. How does stochasticity contribute in regard to this?
- What is the amount of data needed for such a network to achieve a small estimation error with high probability?

In the current paper, we set out to answer these questions by modelling a biological neural network as a continuous-time (stochastic) RNN with a randomly chosen connectivity matrix and an identity activation function in view of classifying data streams (in this case, time-dependent paths). Let us say a few words about each of our three working assumptions:

- The continuous-time dynamics are a generalisation of the classical discrete-time dynamics frequently encountered in the literature [14]. The latter can be seen as an Euler discretisation of the former as the data stream is sampled at shorter time intervals. Working with continuous-time dynamics provides us with a richer mathematical toolbox, while still being applicable to the discrete-time case and keeping key features and issues of such systems such as the dependence on the whole data sequence and its order.
- Randomly generating the connectivity matrix of an RNN is the cornerstone of reservoir computing [20, 31]. This paradigm is based on the idea that universal approximation properties can be achieved for several dynamical systems without the need to optimise all parameters and has shown exceptional performances in a variety of tasks. This working assumption also has the benefit of simplifying the training process (as will be clear from the formulas in this paper, optimising over this matrix is computationally heavy, even in the linear case). This will consist in our case in finding a pre-processing projection vector and the parameters of a read-out map. This simplicity can be practically exploited, for instance to deploy the same network to deal with several tasks (i.e. multi-tasking) without the need for heavy retraining or storing a large number of parameters, in a fashion that is reminiscent of biological networks. Compared to the existing literature (e.g. [9]), we included the pre-processing map (input projection vector) as a tunable parameter in order to increase the performance compared to a classical reservoir computer.
- We choose to work with identity activation functions in order to build the intuition as to the answer to the questions above. In this case, we obtain precise formulas. We aim to generalise the results of this study to the non-linear case in a later study.

Before setting out our roadmap, let us note for the sake of completeness that there exists a number of ways in which one may avoid altogether recurrent architectures in order to handle data streams and use instead a feed-forward network, which is a more studied and understood paradigm. These are usually based on the transformation of paths into fixed-length vectors that can then be fed to the feed-forward structure. In particular, we cite the Independent Component Analysis [17, 19, 34], the signature methods [15, 21, 26] and the PCA-type dimension reduction introduced in [3]. However, these methods work best when the whole signal is processed (which may be computationally heavy) before being fed to the network, while RNNs are able to work with these signals in a continuous manner as they come, rendering them more suitable to real-time situations. As to the approximation properties of these recurrent architectures, there are several studies that suggest that such properties may also hold for the path-classification problem treated in this paper, although a precise statement in the stochastic case, which is of interest to us, is still missing. For example, rigorous results providing the approximation properties of (discrete-time) RNNs with a randomly generated connectivity matrix can be found in [13]. In [12], it is shown that every continuous path can be approximated (in the uniform convergence norm) as the outcome of an RNN with a suitable activation function, while,

more recently in [27], the authors show that an RNN with the identity activation function can approximate any functional on a path space provided it is continuous, linear, regular and time-homogeneous.

We will approach the problem of the binary classification of continuous-time paths with RNNs in the presence of noise from the point of view of statistical learning theory. After introducing the necessary mathematical notation and the learning setup (model, loss function, etc.) in Sect. 2, we will give a generalisation error bound that holds with high probability in Sect. 3. The uniform bound that we derive controls the difference between the risk of an hypothesis and its empirical counterpart and answers practical questions concerning the size of the sample and the bounds on the pre-processing and read-out maps needed to achieve a certain accuracy. Consequently, minimising the empirical risk achieves agnostic PAC learnability and gives guarantees on the ability of the empirical risk minimiser to generalise to unseen data. Section 4 looks in more detail at the empirical risk-minimisation (ERM) procedure:

- we compare its output to that of the popular Support Vector Machine (SVM) considered for example in [33];
- argue heuristically that noise, which is a natural assumption in modelling biological neural networks, provides stability and robustness against different types of perturbations to the dataset;
- show rigorously that in the linear case, the RNN retains a "partial signature" of the time-lifted input signal as global information about said signal. The empirical risk is a function of the tunable parameters of the model and the partial signatures of the training data.

Finally, we look into the numerical minimisation of the empirical risk using gradient descent in Sect. 5. With the explicit formulas we obtain, the global effect through time of the tunable parameters on the loss function is taken into account and we do not need some sort of unrolling of the network to apply back-propagation through time. The experiments are performed using the Japanese vowels dataset and classes of trigonometric polynomials.

2 The learning setup

2.1 The recurrent network input–output map

Let r, n and d be integers. The input and the hidden state of the RNN are modelled, respectively, as r- and n-dimensional time-dependent continuous paths x and y. Given a filtered probability space $(\Omega, \mathcal{A}, (\mathcal{F}_t)_{0\le t\le T}, \mathbb{P})$, a simple continuous-time model describing the time evolution of input–output dynamics is given by the following stochastic differential equation (SDE) (which can be seen as a stochastic version of the one in [37])

$$\mathrm{d}y(t) = \big(-y(t) + W\phi\big(y(t)\big) + u\big(x(t)\big)\big)\,\mathrm{d}t + \Sigma\,\mathrm{d}B(t), \quad t \le T, \tag{1}$$

with initial condition taken to be $y(0) = 0$. Here, $u \in \mathcal{L}(\mathbb{R}^r, \mathbb{R}^n)$ is a linear pre-processing map (identified as a matrix in $\mathbb{R}^{n\times r}$), $W \in \mathbb{R}^{n\times n}$ is the network matrix that models the connection strength between neurons, ϕ is an activation function (applied element-wise) and $\Sigma \in \mathbb{R}^{n\times d}$ is a matrix (which we will call the noise matrix) describing the random effect of a d-dimensional Brownian noise B. In this paper, we will consider the linear case when ϕ is the identity function. The more interesting case where ϕ is non-linear will be the subject of a future study.

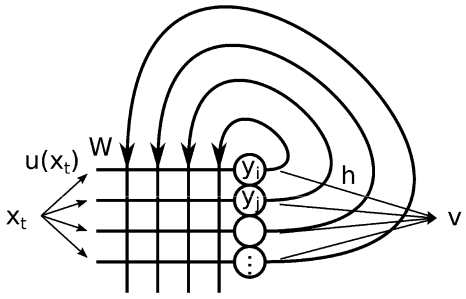

Figure 1 A sketch of the recurrent learning architecture

Given a path x, we will denote by $y(T,x)$ (or $y_u(T,x)$ to emphasise the dependence on the pre-processing map u) the terminal value (i.e. at time T) of the solution to equation (1). A readout map h is then combined with the final hidden state of the neural network to produce a prediction $v = h(y(T,x))$ (Fig. 1). In our case, h will be a labelling function, and more specifically, a hyperplane classifier.

Given a training set of labelled inputs, we aim to train this network by changing the values of the parameters in order to increase the accuracy of future predictions (in a sense to be made clear in the following subsections). In the classical framework of reservoir computing, the network's tunable parameter is the hyperplane h, while the connectivity matrix W and the pre-processing map u are generated randomly. In our case, we aim to increase the performance by considering u to be a tunable parameter to be optimised according to the learning task at hand.

2.2 Hypothesis class and read-out maps

As we have alluded to above, our global hypothesis class $\mathcal{H}$ comprises of maps that can be written as the composition of the reservoir-solution map $y_u(T,\cdot)$ and a read-out map h chosen from a read-out hypothesis class $\mathcal{H}^*$. As the read-out map h will be applied to the random vector $y_u(T,x)$, we can think of the hypothesis class $\mathcal{H}$ as a class of random learners

$$\begin{aligned} H : \mathcal{X} &\longrightarrow \mathcal{V}^{\Omega}, \\ x &\longmapsto h\big(y_u(T,x)\big) = v(x), \end{aligned}$$

where $\mathcal{X}$ denotes the space of input paths and $\mathcal{V}$ the target set of outputs; for example the labels $\{-1,+1\}$ as will be in our case. If one identifies random variables and their probability distributions, then one may think of the result of the learners applied to an input x as probability distributions instead of a single label. In our very simple setting, this translates into thinking of the hypothesis H as a regression function $x \longmapsto \mathbb{P}(H(x) = 1) \in [0,1]$. This discussion fits into the framework of probabilistic binary classification, or in the wider one of probabilistic supervised learning, as introduced in [16]. However, let us emphasise the fundamental difference that in our case we label inputs based on one realisation of the hypothesis rather than produce a probability distribution on the space of labels.

In the current work, and in line with most common practices, we will take the read-out hypothesis class $\mathcal{H}^*$ to be the class of hyperplane classifiers. We will adopt the following notations:

Notation 2.1

(1) For $\omega \in \mathbb{R}^n$ and $b \in \mathbb{R}$, we denote by $h_{\omega,b}$ the hyperplane classifier with normal direction $\omega \in \mathbb{R}^n$ and shift b

$$h_{\omega,b}(y) = \operatorname{sign}\big(\langle y, \omega\rangle + b\big) \in \{-1, +1\}, \quad \text{for all } y \in \mathbb{R}^n,$$

with the convention $\operatorname{sign}(0) = 1$.

(2) If $u \in \mathbb{R}^{n\times r}$, $H_{u,\omega,b}$ denotes the global classifier of the stochastic RNN with pre-processing map u

$$H_{u,\omega,b}(x) = h_{\omega,b}\big(y_u(T,x)\big), \quad \text{for all } x \in \mathcal{X}.$$

2.3 Loss functions and associated risk

The goal of supervised learning is to maximise the ability, with high probability and measured against a loss function, of the predicted outputs to generalise to unseen data. As our learner is generating a random variable instead of a deterministic label, we need to consider loss functions of the type $l : \mathcal{V}^\Omega \times \mathcal{V} \to \mathbb{R}_+$. Given a classical loss function $\tilde{l} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_+$ (e.g. the square loss or the binary loss), we may construct loss functions suitable to our framework in two ways, amongst others. The first way is by defining the loss function l to be a statistic of the random variable $\omega \mapsto \tilde{l}(\tilde{\nu}(\omega), \nu)$, for example

$$\begin{aligned} l : \mathcal{V}^\Omega \times \mathcal{V} &\longrightarrow \mathbb{R}_+, \\ (\tilde{\nu}, \nu) &\longmapsto \mathbb{E}\tilde{l}(\tilde{\nu}, \nu). \end{aligned}$$

For a fixed input and label, the sole source of randomness in our example is the d-dimensional Brownian motion B with respect to which we will take the expectation. An explicit example would be the binary loss function $\tilde{l}(\tilde{\nu}, \nu) = \mathbf{1}_{\tilde{\nu}\neq\nu}$ to which we associate the loss $l(\tilde{\nu}, \nu) = \mathbb{P}(\tilde{\nu} \neq \nu)$. This is the loss function that we will consider in our case. We choose this loss function as it is simpler to analyse than other popular types of losses (square loss, hinge loss, etc.) while involving similar key quantities (the Gaussian cumulative distribution function, as will be seen later) in the classification process and risk minimisation.

A second type of loss function can be obtained by defining a statistical functional $\Psi : \mathcal{V}^\Omega \to \mathcal{V}$ (here $\mathcal{V}$ can be understood in a broader sense, for example $\mathbb{R}$, instead of the labels $\{-1, +1\}$), then define $l(\tilde{\nu}, \nu) = \tilde{l}(\Psi(\tilde{\nu}), \nu)$. This type of loss function depends on the distribution produced by the hypothesis rather than on its single realisations and defines therefore a probabilistic loss function in the sense of [16]. An instance of such type would be to take $\Psi = \mathbb{E}$ and the square loss $\tilde{l}(\tilde{\nu}, \nu) = |\tilde{\nu} - \nu|^2$ to obtain $l(\tilde{\nu}, \nu) = |\mathbb{E}\tilde{\nu} - \nu|^2$.

The generalisation error (also called risk) of an hypothesis $H \in \mathcal{H}$ associated to a loss function l is then classically given by $R(H) = \mathbb{E}_{(x,\nu)} l(H(x), \nu)$, where the expectation is taken with respect to the (unknown) distribution according to which an input and its label (x, ν) are generated. One usually interprets the generalisation error as the ability of the learned hypothesis to generalise well for unseen data, assuming it is sampled according to the same unknown distribution that generated the training sample. As no assumptions are made in regard to the distribution of the data, the generalisation error remains unknown and thus its minimisation, as such, impossible. Instead, one classically aims to minimise

its empirical counterpart based on a training sample $(X, V) = \{(x_i, v_i)\}_{i=1}^m$

$$\widehat{R}_{(X,V)}(H) = \frac{1}{m}\sum_{i=1}^{m} l\big(H(x_i), v_i\big).$$

Obviously, one also needs to provide theoretical arguments justifying the use of the empirical error instead of the generalised one. These often come in the form of a concentration inequality. We will recall such an argument using the notion of Rademacher complexity in a subsequent subsection.

If we denote by $\overline{\mathcal{H}}$ the set of all measurable hypotheses, we can then decompose the difference between the true risk of an hypothesis $H \in \mathcal{H}$ and the smallest possible risk in the following manner:

$$R(H) - \inf_{H_0\in\overline{\mathcal{H}}} R(H_0) = \underbrace{R(H) - \inf_{H_0\in\mathcal{H}} R(H_0)}_{E_{\text{est}}} + \underbrace{\inf_{H_0\in\mathcal{H}} R(H_0) - \inf_{H_0\in\overline{\mathcal{H}}} R(H_0)}_{E_{\text{app}}}.$$

On the one hand, the estimation error E_{est} depends on the ability to solve the problem of minimising the risk over the chosen hypotheses class $\mathcal{H}$. Intuitively, this problem becomes more difficult with larger classes of hypotheses. In many situations where the concentration inequalities mentioned above apply, one obtains quantitative guarantees on how small the estimation error E_{est} can be made in the case of an empirical risk minimiser

$$H^{\text{ERM}} \in \operatorname*{argmin}_{H\in\mathcal{H}} \widehat{R}_{(X,V)}(H). \tag{2}$$

On the other hand, the approximation error E_{app} depends on how accurately the hypotheses in $\mathcal{H}$ can approximate any measurable hypothesis.

3 Generalisation error bounds

3.1 Solution to the SDE

In our simplified case of the identity function as an activation function, and similarly to an Ornstein–Uhlenbeck process, the solution to the SDE (1) is explicitly given by

$$y(t) = \int_0^t e^{(W-I)(t-s)} u\big(x(s)\big)\,\mathrm{d}s + \int_0^t e^{(W-I)(t-s)}\Sigma\,\mathrm{d}B(s), \quad t \le T.$$

Notation 3.1 For convenience, we will write $W_0 = W - I$.

The final hidden state $y(T, x)$ is then a Gaussian random vector with mean and covariance matrix given by

$$y(T,x) \sim \mathcal{N}\left(\int_0^T e^{W_0(T-s)} u\big(x(s)\big)\,\mathrm{d}s, \int_0^T e^{W_0(T-s)}\Sigma\Sigma^\top e^{W_0^\top(T-s)}\,\mathrm{d}s\right).$$

Notation 3.2 For the rest of this paper, we will denote

$$\nu_{x,u} = \int_0^T e^{W_0(T-s)} u\big(x(s)\big)\,\mathrm{d}s \in \mathbb{R}^n \quad \text{and} \quad A = \int_0^T e^{W_0(T-s)}\Sigma\Sigma^\top e^{W_0^\top(T-s)}\,\mathrm{d}s \in \mathbb{R}^{n\times n}.$$

Remark 3.3 Note that the covariance matrix A is independent of the tunable parameters u, ω and b and the input signals. As will be seen later, this is a key property of this algorithm.

It is also interesting to note that in this linear case, the hidden states of different inputs have similar distributions (Gaussian) differing only by their means. Loosely speaking, the role of the parameter u will then be to separate as much as possible the data x_i according to their labels ν_is through their processed weighted averages $\nu_{x_i,u}$, while the covariance matrix A quantifies by how much the hidden states are likely to deviate from their means. As in [33], one may be tempted to apply traditional separating algorithms such as soft SVM as a classification technique. However, we follow here a strategy dictated by the risk minimisation and the learning guarantees given by the generalisation error bounds that we will obtain. We will compare these two techniques in a subsequent subsection.

3.2 Empirical risk for the binary loss

As stated in Sect. 2.3, we consider the case of the binary loss function. We will denote by $\tilde{l}$ the classical binary loss function $(\tilde{\nu}, \nu) \mapsto \mathbf{1}_{\tilde{\nu} \neq \nu}$, while l denotes its counterpart associated with random labels (assuming measurability)

$$l : (\tilde{\nu}, \nu) \longmapsto \mathbb{E}\tilde{l}(\tilde{\nu}, \nu) = \mathbb{P}(\tilde{\nu} \neq \nu).$$

We will give first the exact formula of the loss in the pure stochastic regime, which shows the smoothing effect of the noise on the binary loss function. For a lighter notation, we will sometimes avoid making the dependence on other variables explicit. For example, we may write y instead of $y_u(T, x)$.

Proposition 3.4 *Let* $(x, \nu) \in \mathcal{X} \times \{-1, 1\}$, $u \in \mathbb{R}^{n \times r}$, $\omega \in \mathbb{R}^n$ *and* $b \in \mathbb{R}$. *If* $\omega \notin \operatorname{Ker}(A)$, *then*

$$l\big(H_{u,\omega,b}(x), \nu\big) = \Phi\left(-\nu \cdot \frac{\langle \nu_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}\right), \tag{3}$$

where Φ *denotes the standard Gaussian cumulative distribution function* (*CDF*).

Proof First, note that

$$\tilde{l}\big(H_{u,\omega,b}(x), \nu\big) = \mathbf{1}_{\operatorname{sign}(\langle y, \omega \rangle + b) \neq \nu} \leq \mathbf{1}_{\nu(\langle y, \omega \rangle + b) \leq 0}$$

and that this inequality is not an identity if and only if $\nu = 1$ and $\langle y, \omega \rangle + b = 0$ (because of the convention on the sign of 0). Next, recall that $\langle y, \omega \rangle + b$ is a Gaussian random variable

$$\langle y, \omega \rangle + b \sim \mathcal{N}\big(\langle \nu_{x,u}, \omega \rangle + b, \omega^\top A \omega\big).$$

We assume that $\omega \notin \operatorname{Ker}(A)$. Then, $\langle y, \omega \rangle + b$ is a non-trivial Gaussian random variable (and $\mathbb{P}(\langle y, \omega \rangle + b = 0) = 0$). Consequently

$$l\big(H_{u,\omega,b}(x), \nu\big) = \mathbb{E}_B \tilde{l}\big(H_{u,\omega,b}(x), \nu\big) = \mathbb{P}\big(\nu\big(\langle y, \omega \rangle + b\big) \leq 0\big) = \Phi\left(-\nu \cdot \frac{\langle \nu_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}\right),$$

which completes the proof. □

Remark 3.5 Note that if $\omega \notin \mathrm{Ker}(A)$, then one cannot obtain a null empirical risk because of the Gaussian CDF. The quantity

$$\frac{\langle \nu_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}$$

resembles a distance from a hyperplane (more details will be provided in the next section). Similar in spirit to the soft-SVM problem, the loss function (3) penalises both misclassification and close proximity to said hyperplane.

Remark 3.6 If $\omega \in \mathrm{Ker}(A)$, then $\langle y, \omega \rangle = \langle \nu_{x,u}, \omega \rangle$. We are then in a non-stochastic regime where full accuracy on the training set can be achieved if the averages $\nu_{x_i,u}$ are separable, since then

$$l\big(H_{u,\omega,b}(x), \nu\big) = \mathbf{1}_{\mathrm{sign}(\langle \nu_{x,u}, \omega \rangle + b) \neq \nu}.$$

In this regime, only misclassification is penalised.

3.3 Main result

Classically, quantitative guarantees for the minimisation of the estimation error within a chosen hypothesis class (agnostic PAC-learnability) is obtained by first showing that the hypothesis set satisfies the uniform convergence property (cf. [36]), i.e. by obtaining probabilistic bounds for the worst-in-class difference between the generalisation error and the empirical error

$$\mathbb{P}\Big(\sup_{H \in \mathcal{H}} \big| R(H) - \widehat{R}_{(X,V)}(H) \big| > \varepsilon \Big).$$

In turn, a way to achieve this is through controlling the Rademacher complexity (or the growth function or the VC dimension) of the hypothesis class. Given a state space $\mathcal{Z}$, a sample $Z = \{z_i\}_{i=1}^m$ of points in $\mathcal{Z}$ and a class $\mathcal{G}$ of real-valued maps defined on $\mathcal{Z}$, the (empirical) Rademacher complexity of $\mathcal{G}$ with respect to the sample Z is defined by (cf. [32])

$$\mathcal{R}_Z(\mathcal{G}) = \frac{1}{m} \mathbb{E}_\varepsilon \left(\sup_{g \in \mathcal{G}} \sum_{i=1}^m \varepsilon_i g(z_i) \right),$$

where the ε_is are independent Rademacher (i.e. symmetric Bernoulli) random variables and $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_m)$.[1] The Rademacher complexity can be seen as a measure of the richness of the class of functions $\mathcal{G}$ and its ability to provide a variety of labels $\{g(z_i)\}_{i=1}^m$ for the sample S. If we assume that the sample Z is drawn in an i.i.d. manner according to a distribution $\mathcal{D}$, we obtain the following concentration inequality:

Theorem 3.7 ([2, 23, 32]) *Let $\mathcal{G}$ be a family of real-valued maps defined over a sample space $\mathcal{Z}$ with values in the interval* $[0, 1]$. *Let $\mathcal{D}$ be a distribution over $\mathcal{Z}$ and $Z = \{z_i\}_{i=1}^m \sim$*

[1] We assume that the random quantity inside the expectation is indeed measurable. This happens to be the case for most machine-learning applications, including the one discussed in this paper.

$\mathcal{D}^m$ be a random sample. Denote by z a random variable distributed according to $\mathcal{D}$. Let $\delta > 0$. Then, the following holds with probability at least $1 - \delta$

$$\sup_{g\in\mathcal{G}}\left|\mathbb{E}g(z) - \frac{1}{m}\sum_{i=1}^{m} g(z_i)\right| \leq 4\mathcal{R}_Z(\mathcal{G}) + \frac{2 + 5\sqrt{\log(2/\delta)/2}}{\sqrt{m}}.$$

We will estimate the Rademacher complexity in our setting then apply the above theorem to quantify the error in estimating the true risk by its empirical counterpart. This will yield our main theoretical result below. For matrices, $\|\cdot\|$ denotes the spectral norm.

Theorem 3.8 *Let Θ and Λ be two positive real numbers. We consider the family of hypotheses given by*

$$\mathcal{H}_{\Theta,\Lambda} = \left\{H_{u,\omega,b} : \|\omega\|_2 = 1, |b| \leq \Theta, \|u\| \leq \Lambda\right\}.$$

Let $\delta > 0, R > 0$ and $m \in \mathbb{N}^*$. We assume that the input signals lie almost surely in the L^2-ball of radius R

$$\mathcal{B}_R = \left\{x \in \mathcal{C}\left([0,T],\mathbb{R}^r\right) : \left(\int_0^T \|x_s\|_2^2\,\mathrm{d}s\right)^{1/2} \leq R\right\},$$

and that the covariance matrix A is positive-definite. We denote by $\lambda_{\min}(A)$ its smallest eigenvalue. Let $\mathcal{D}$ be a distribution over $\mathcal{B}_R \times \{-1,1\}$ and $(X,V) = \{(x_i,v_i)\}_{i=1}^m \sim \mathcal{D}^m$ be a random sample. Then, with probability at least $1 - \delta$

$$\begin{aligned}&\sup_{H\in\mathcal{H}_{\Theta,\Lambda}} \left|R(H) - \widehat{R}_{(X,V)}(H)\right| \\ &\quad\leq \frac{4}{\sqrt{2\pi m\lambda_{\min}(A)}}\left(\Theta + \Lambda R\left(\int_0^t \left\|e^{W_0(t-s)}\right\|^2 \mathrm{d}s\right)^{1/2}\right) + \frac{2 + 5\sqrt{\log(2/\delta)/2}}{\sqrt{m}}.\end{aligned}$$

Proof By applying Theorem 3.7 for the set of functions

$$(x,v) \longmapsto l\big(H(x),v\big), \quad H \in \mathcal{H}_{\Theta,\Lambda},$$

together with the formula obtained in Proposition 3.4 (using the assumption on A being positive-definite), we obtain that the following holds with probability at least $1 - \delta$,

$$\sup_{H\in\mathcal{H}}\left|R(H) - \widehat{R}_{(X,V)}(H)\right| \leq \frac{4}{m}\mathbb{E}_\varepsilon \sup_{u,\omega,b}\sum_{i=1}^{m}\varepsilon_i\Phi\left(-v_i\cdot\frac{\langle v_{x_i,u},\omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right) + \frac{2 + 5\sqrt{\log(2/\delta)/2}}{\sqrt{m}},$$

with the supremum taken over the set

$$\left\{(u,\omega,b) : \|\omega\|_2 = 1, |b| \leq \Theta, \|u\| \leq \Lambda\right\}.$$

As Φ is Lipschitz with constant $\frac{1}{\sqrt{2\pi}}$, by Talagrand's inequality [25, 32]

$$\begin{aligned}\mathbb{E}_\varepsilon \sup_{u,\omega,b} \sum_{i=1}^{m} \varepsilon_i \Phi\left(-\nu_i \cdot \frac{\langle \nu_{x_i,u}, \omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right) &\le \frac{1}{\sqrt{2\pi}} \mathbb{E}_\varepsilon \sup_{\omega,b,u} \sum_{i=1}^{m}\left(-\varepsilon_i \nu_i \cdot \frac{\langle \nu_{x_i,u}, \omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right) \\ &= \frac{1}{\sqrt{2\pi}} \mathbb{E}_\varepsilon \sup_{\omega,b,u} \sum_{i=1}^{m}\left(\varepsilon_i \cdot \frac{\langle \nu_{x_i,u}, \omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right),\end{aligned}$$

where we used the fact that $(\varepsilon_i) \sim (\varepsilon_i \nu_i)$ in the last line. On the one hand,

$$\mathbb{E}_\varepsilon \sup_{u,\omega,b} \sum_{i=1}^{m} \frac{\varepsilon_i b}{\sqrt{\omega^\top A\omega}} \le \frac{\Theta}{\sqrt{\lambda_{\min}(A)}} \mathbb{E}_\varepsilon \left|\sum_{i=1}^{m} \varepsilon_i\right| \le \frac{\Theta}{\sqrt{\lambda_{\min}(A)}} \sqrt{\mathbb{E}_\varepsilon \left(\sum_{i=1}^{m} \varepsilon_i\right)^2} = \frac{\Theta\sqrt{m}}{\sqrt{\lambda_{\min}(A)}},$$

and on the other hand,

$$\mathbb{E}_\varepsilon \sup_{u,\omega,b} \frac{\langle \sum_1^m \varepsilon_i \nu_{x_i,u}, \omega\rangle}{\sqrt{\omega^\top A\omega}} \le \frac{1}{\sqrt{\lambda_{\min}(A)}} \mathbb{E}_\varepsilon \sup_{\|u\|\le\Lambda} \left\|\sum_{i=1}^{m} \varepsilon_i \nu_{x_i,u}\right\|_2.$$

Recall that

$$\sum_{i=1}^{m} \varepsilon_i \nu_{x_i,u} = \int_0^T e^{W_0(T-s)} \sum_{i=1}^{m} \varepsilon_i u\big(x_i(s)\big)\,\mathrm{d}s.$$

We now use the following inequality for matrix-valued maps $f : [0,T] \to \mathbb{R}^{n\times n}$ and $g : [0,T] \to \mathbb{R}^n$

$$\left\|\int_0^T f(s)g(s)\,\mathrm{d}s\right\|_2 \le \left(\int_0^T \|f(s)\|^2\,\mathrm{d}s\right)^{1/2} \left(\int_0^T \|g(s)\|_2^2\,\mathrm{d}s\right)^{1/2},$$

together with the linearity of u and $\|u\| \le \Lambda$, ensuring $\|\sum_1^m \varepsilon_i u(x_i(s))\| \le \Lambda \|\sum_1^m \varepsilon_i x_i(s)\|$, to obtain

$$\begin{aligned}\mathbb{E}_\varepsilon \sup_{\|u\|\le\Lambda} \left\|\sum_1^m \varepsilon_i \nu_{x_i,u}\right\|_2 &\le \left(\int_0^T \left\|e^{W_0(T-s)}\right\|^2 \mathrm{d}s\right)^{1/2} \mathbb{E}_\varepsilon \sup_{\|u\|\le\Lambda} \left(\int_0^T \left\|\sum_1^m \varepsilon_i u\big(x_i(s)\big)\right\|_2^2 \mathrm{d}s\right)^{1/2} \\ &\le \Lambda \left(\int_0^T \left\|e^{W_0(T-s)}\right\|^2 \mathrm{d}s\right)^{1/2} \mathbb{E}_\varepsilon \left(\int_0^T \left\|\sum_1^m \varepsilon_i x_i(s)\right\|_2^2 \mathrm{d}s\right)^{1/2}.\end{aligned}$$

Using Jensen's and Fubini's inequalities, we obtain the bound

$$\begin{aligned}\mathbb{E}_\varepsilon \left(\int_0^T \left\|\sum_{i=1}^m \varepsilon_i x_i(s)\right\|_2^2 \mathrm{d}s\right)^{1/2} &\le \left(\int_0^T \mathbb{E}_\varepsilon \left\|\sum_{i=1}^m \varepsilon_i x_i(s)\right\|_2^2 \mathrm{d}s\right)^{1/2} \\ &= \left(\sum_{i=1}^m \int_0^T \|x_i(s)\|_2^2\,\mathrm{d}s\right)^{1/2}.\end{aligned}$$

Using the fact that the input paths live in the set $\mathcal{B}_R$, we conclude that:

$$\mathbb{E}_\varepsilon \sup_{u,\omega,b} \sum_{i=1}^{m} \varepsilon_i \Phi\left(-\nu_i \cdot \frac{\mu_{x_i,\omega} + b}{\sqrt{\omega^\top A \omega}}\right) \leq \sqrt{\frac{m}{2\pi \lambda_{\min}(A)}} \left(\Theta + \Lambda R \left(\int_0^T \left\| e^{W_0(T-s)} \right\|^2 \mathrm{d}s\right)^{1/2}\right),$$

which gives the desired inequality. □

Remark 3.9 The use of the Rademacher complexity allows us to obtain a generalisation error bound that decays as $\frac{1}{\sqrt{m}}$, which is the common rate of decay encountered in the classical supervised learning framework (with i.i.d. entries). This comes, however, at the cost of the technical assumption of the inputs being uniformly bounded in the L^2-norm (which is arguably a realistic assumption). Similarly to the error bounds for the SVM algorithm, it may be possible to relax this assumption at the cost of a much slower rate of decay by using the notion of VC dimension. For example, given m labelled points $(X, V) = \{(x_i, v_i)\}_{i=1}^m$ in $\mathbb{R}^n$, one obtains with probability at least $1 - \delta$ for all hyperplane classifiers (cf. [32])

$$R(H) \leq \widehat{R}_{(X,V)}(H) + \sqrt{\frac{2(n+1)\log(\frac{em}{n+1})}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}}. \tag{4}$$

In the discrete case, such bounds on the VC dimension of RNNs (as a function of the number of weights in the network and the length of the sequence—which would correspond here to the number of steps one uses to discretise an input path) have been obtained, for example, by Koiran and Sontag in [22].

The previous theorem allows us to quantitatively control the estimation error:

Corollary 3.10 *Let Θ, Λ and R be positive real numbers. We assume that the input signals lie almost surely in the L^2-ball of radius R and that the covariance matrix A is definite. Then, the class $\mathcal{H}_{\Theta,\Lambda}$ is agnostically PAC learnable through its empirical risk-minimiser hypothesis H^{ERM} (defined in (2)), i.e. there exists a function $\tilde{m} : (0,1)^2 \to \mathbb{N}$ such that for all $\varepsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, 1\}$ and every sample $(X, V) \sim \mathcal{D}^m$ of size $m \geq \tilde{m}(\varepsilon, \delta)$, we have with probability at least $1 - \delta$*

$$R\big(H^{\mathrm{ERM}}\big) \leq \inf_{H \in \mathcal{H}_{\Theta,\Lambda}} R(H) + \varepsilon.$$

More explicitly, one may take $\tilde{m}(\varepsilon, \delta)$ to be an integer m (ideally the smallest one) such that

$$m \geq \frac{4}{\varepsilon^2} \left(\frac{4(\Theta + \Lambda R(\int_0^T \|e^{W_0(T-s)}\|^2 \,\mathrm{d}s)^{1/2})}{\sqrt{2\pi \lambda_{\min}(A)}} + 2 + 5\sqrt{\log(2/\delta)} \right)^2.$$

Proof Let $\varepsilon, \delta \in (0,1)$. Let $\tilde{m}(\varepsilon, \delta)$ be the smallest integer m such that

$$\frac{4}{\sqrt{2\pi m \lambda_{\min}(A)}} \left(\Theta + \Lambda R \left(\int_0^T \left\| e^{W_0(T-s)} \right\|^2 \mathrm{d}s\right)^{1/2}\right) + \frac{2 + 5\sqrt{\log(2/\delta)}}{\sqrt{m}} \leq \frac{\varepsilon}{2}.$$

Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{-1, 1\}$ and $(X, V) = \{(x_i, v_i)\}_{i=1}^m \sim \mathcal{D}^m$ be a random sample of size $m \geq \tilde{m}(\varepsilon, \delta)$. Note that if

$$\sup_{H \in \mathcal{H}_{\Theta,\Lambda}} \left| R(H) - \widehat{R}_{(X,V)}(H) \right| \leq \frac{\varepsilon}{2},$$

then for any hypothesis $H \in \mathcal{H}_{\Theta,\Lambda}$

$$\begin{aligned} R\big(H^{\mathrm{ERM}}\big) - R(H) &= R\big(H^{\mathrm{ERM}}\big) - \widehat{R}_{(X,V)}\big(H^{\mathrm{ERM}}\big) + \widehat{R}_{(X,V)}\big(H^{\mathrm{ERM}}\big) - \widehat{R}_{(X,V)}(H) \\ &\quad + \widehat{R}_{(X,V)}(H) - R(H) \\ &\leq \varepsilon. \end{aligned}$$

Hence, $R(H^{\mathrm{ERM}}) \leq \inf_{H\in\mathcal{H}_{\Theta,\Lambda}} R(H) + \varepsilon$ and

$$\mathbb{P}\Big(R\big(H^{\mathrm{ERM}}\big) \leq \inf_{H\in\mathcal{H}_{\Theta,\Lambda}} R(H) + \varepsilon\Big) \geq \mathbb{P}\Big(\sup_{H\in\mathcal{H}_{\Theta,\Lambda}} \big|R(H) - \widehat{R}_{(X,V)}(H)\big| \leq \frac{\varepsilon}{2}\Big).$$

Finally, note that by Theorem 3.8 and the definition of $\tilde{m}(\varepsilon,\delta)$ we have

$$\mathbb{P}\Big(\sup_{H\in\mathcal{H}_{\Theta,\Lambda}} \big|R(H) - \widehat{R}_{(X,V)}(H)\big| \leq \frac{\varepsilon}{2}\Big) \geq 1-\delta.$$

This completes the proof. □

Remark 3.11 Corollary 3.10 shows an additional major advantage of the bound obtained in Theorem 3.8: it (quantitatively) guarantees that the empirical risk minimiser is the best-in-class hypothesis with high probability. The bounds discussed, for example, in Remark 3.9 aim to directly lower the risk by choosing parameters for the hypothesis that minimises the upper bound of the risk (the right-hand side term in (4)). The efficiency of such a technique is thus very dependent on said bound being tight.

4 A study of the empirical risk

In this section, we aim to provide a better understanding of the empirical risk (in view of our future work on the non-linear case). We recall that in the pure robust stochastic case (i.e. where the covariance matrix A is positive-definite) and given a sample $(X,V) = \{(x_i,v_i)\}_{i=1}^m$, the empirical risk of an hypothesis $H_{u,\omega,b}$ is given by the formula

$$\widehat{R}_{(X,V)}(H_{u,\omega,b}) = \frac{1}{m}\sum_{i=1}^{m} \Phi\Big(-v_i \cdot \frac{\langle v_{x_i,u},\omega\rangle + b}{\sqrt{\omega^\top A\omega}}\Big).$$

In the subsequent subsections, we will compare and draw parallels between the empirical risk minimisation (ERM) and the support vector machine (SVM) approaches and show that stochastic (linear) RNNs keep a "partial signature" of the input path as a summary of the information about the path.

4.1 An interpretation via margins

By making the change of variable $\alpha = A^{1/2}\omega$, one can rewrite

$$\frac{\langle v_{x,u},\omega\rangle + b}{\sqrt{\omega^\top A\omega}} = \frac{\langle A^{-1/2}v_{x,u},\alpha\rangle + b}{\|\alpha\|_2}.$$

Note that the absolute value of the quantity above is the distance of the "transformed mean" $A^{-1/2}v_{x,u}$ from the hyperplane $H(\alpha,b)$. Given such a hyperplane, let I_0 be the set of

indices of input paths whose transformed means are correctly classified and $m_0 := |I_0|$ be its cardinal. Define the corresponding margin ρ_0 as the distance between the hyperplane $H(\alpha, b)$ and the closest correctly classified transformed average $A^{-1/2}\nu_{x_i,u}$,

$$\rho_0 := \min_{i \in I_0} \mathrm{d}\big(A^{-1/2}\nu_{x_i,u}, H(\alpha, b)\big).$$

Then, for all $i \in I_0$, one has

$$\Phi\left(-\nu_i \cdot \frac{\langle A^{-1/2}\nu_{x_i,u}, \alpha\rangle + b}{\|\alpha\|_2}\right) \leq \Phi(-\rho_0).$$

Similarly, let I_1 be the complement of I_0 and ρ_1 the distance of the furthest misclassified average $A^{-1/2}\nu_{x_i,u}$ from the hyperplane $H(\alpha, b)$

$$\rho_1 := \max_{i \in I_1} \mathrm{d}\big(A^{-1/2}\nu_{x_i,u}, H(\alpha, b)\big).$$

Then, for all $i \in I_1$, one has

$$\Phi\left(-\nu_i \cdot \frac{\langle A^{-1/2}\nu_{x_i,u}, \alpha\rangle + b}{\|\alpha\|_2}\right) \leq \Phi(\rho_1).$$

Using these inequalities, one can bound the empirical risk as follows:

$$\widehat{R}_{(X,V)}(H) \leq \frac{m_0\Phi(-\rho_0) + (m - m_0)\Phi(\rho_1)}{m}.$$

Intuitively, decreasing the upper bound above requires a balance between, on the one hand, correctly classifying the averages $A^{-1/2}\nu_{x_i,u}$ with a large margin ρ_0 and, on the other hand, misclassifying as few averages as possible with the smallest worst "misclassification margin" ρ_1. This is reminiscent of the soft SVM algorithm (see for example [7, 32]), which is the approach adopted in [33] to classify the data according to the statistics of their classes. Let us recall that the soft SVM algorithm consists of solving the following (constrained convex) optimisation problem (the parameter $\lambda \geq 0$ is to be freely chosen depending on the desired properties of the final classifier)

$$\begin{aligned} &\min_{\alpha,b,\xi} \frac{1}{2}\|\alpha\|_2^2 + \lambda\|\xi\|_1 \quad \text{subject to} \quad \nu_i\big(\langle \alpha, A^{-1/2}\nu_{x_i,u}\rangle + b\big) \geq 1 - \xi_i; \\ &\xi_i \geq 0; \forall 1 \leq i \leq m. \end{aligned} \tag{5}$$

Classically, this is equivalent to the dual (quadratic) problem

$$\max_{\theta} \sum_{1 \leq i \leq m} \theta_i - \frac{1}{2}\left\| \sum_{1 \leq i \leq m} \theta_i \nu_i A^{-1/2}\nu_{x_i,u} \right\|_2^2 \quad \text{subject to} \quad \langle \theta, \nu\rangle = 0; \quad \forall i : 0 \leq \theta_i \leq \lambda.$$

Given its minimiser θ^*, the solution of the primal SVM problem (5) can be computed as the direction $\alpha = \sum_{i \leq m} \theta_i^* \nu_i A^{-1/2}\nu_{x_i,u}$ and, given $i \in [\![1, m]\!]$ such that $0 < \theta_i^* < \lambda$, $b = \nu_i - \langle \alpha, A^{-1/2}\nu_{x_i,u}\rangle$.

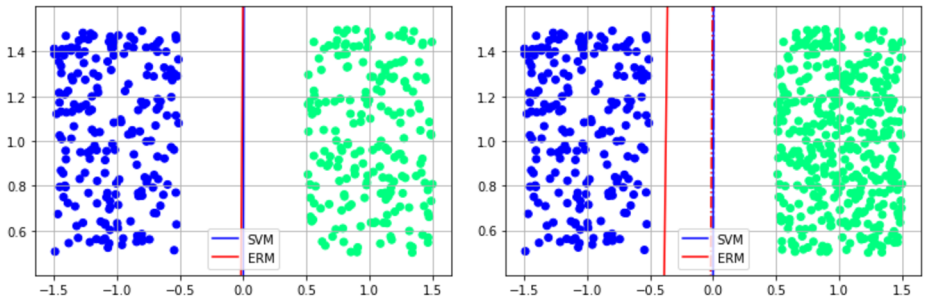

Figure 2 ERM (obtained via a gradient-descent algorithm) and SVM may produce similar results in generic situations (left). However, ERM is more sensitive to the relative number of inputs in each class (right). Dotted lines on the right plot recall the position of the classifiers from the left plot

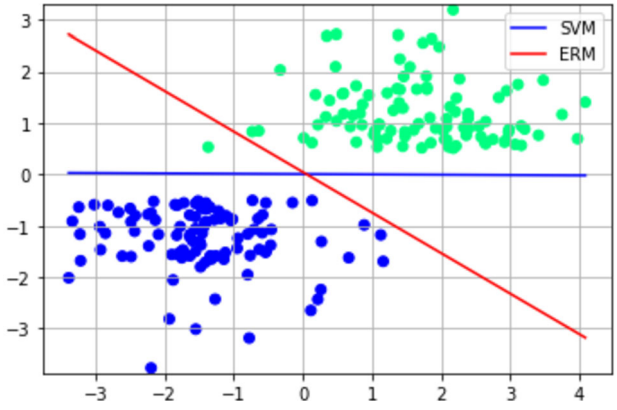

Figure 3 ERM and SVM seek two different geometric objectives that may result in very different outputs

First, note that this optimisation problem is no longer convex if we include the optimisation over the pre-processing map u. Secondly, while the solutions (for a fixed u) of the ERM and soft SVM may be similar in some *generic* situations, there are some significant differences in behaviour and interpretation in the results of the two algorithms:[2]

- The solution to the ERM algorithm is sensitive to the number of inputs in each class (thus, in some way, "learning" the data-generating distribution) while that of the SVM only depends on the support vectors (Fig. 2).
- Another key difference between these two algorithms lies in their respective objectives: the ERM attempts to find a hyperplane where most of the data is (correctly classified and) far away from said hyperplane, while the soft SVM attempts the same but only with respect to the support vectors. The results of these procedures can lead sometimes to very different outputs (Fig. 3).
- The two algorithms have different sensitivities to outliers and mislabelled training data (which we will discuss in the next subsection).
- Given the preprocessing map u, if one denotes the hyperplane parameters returned by the SVM algorithm by (ω_u, b_u), then the smoothness of the map $u \mapsto (\omega_u, b_u)$ (and

[2] In the following arguments, we show these differences for classification tasks for points in the plane. In our setting, this is equivalent to taking the dimensions $r = n = 2$, the pre-processing map $u = \mathrm{Id}$ and the input paths as constant paths.

therefore that of $u \mapsto \widehat{R}_{(X,V)}(H_{u,\omega_u,b_u})$) is much less trivial to prove or even define, thus rendering the use of classical gradient-descent algorithms to optimise over u unjustified (indeed, attempting to do so in several numerical experiments, the algorithm failed to converge).

Remark 4.1 In [33], the SVM algorithm is successfully used to separate classes that can be separated by their statistics (for example, the realisations of two Gaussian processes). A "good" pre-processing map u (based on a mathematical formula) is chosen beforehand; thus avoiding the use of gradient descent to optimise over this parameter.

4.2 Robustness and further analysis of the ERM

Making again the change of variable $\alpha = A^{1/2}\omega$, we saw from the above that, informally, an ERM algorithm has the task to find parameters (u, ω, b) such that all the transformed averages $A^{-1/2}v_{x_i,u}$ are correctly classified, i.e.

$$\forall i \in [\![1, m]\!] : \quad \nu_i\big(\big\langle A^{-1/2}v_{x_i,u}, \alpha\big\rangle + b\big) \geq 0,$$

and as distant from the hyperplane $H(\alpha, b)$ as possible (thus prompting $\Phi(-\nu \cdot \frac{\langle v_{x,u},\omega\rangle + b}{\sqrt{\omega^\top A\omega}})$ to be as small as possible) while working under the constraint that the overall sum of the losses associated to each training input (i.e. the empirical risk) has to be as small as possible. The combination of the latter with the loss function involving the Gaussian CDF is why this ERM algorithm differs from a simple classification of the transformed averages $A^{-1/2}v_{x_i,u}$. For example, if we assume that the two clouds

$$C_+ = \big\{A^{-1/2}v_{x_i,u} : \nu_i = 1\big\} \quad \text{and} \quad C_- = \big\{A^{-1/2}v_{x_i,u} : \nu_i = -1\big\}$$

are concentrated and well separated and then introduce an additional training input x_* with label $\nu_* = 1$ but such that $A^{-1/2}v_{x_*,u}$ is much closer to C_- than to C_+, it is then very possible for the ERM to choose to misclassify x_* rather than to correctly classify it (doing the latter might mean an increase in the empirical risk as the cloud C_- moves much closer to the new hyperplane $H(\alpha, b)$.) In practice, the label ν_* given to the path x_* could be a wrong one (i.e. mislabelled training data). However, this does not necessarily alter the result of this ERM algorithm in a drastic way (which would be the case for the hard SVM algorithm or a soft SVM algorithm with a poor choice of the regularising parameter λ). Figure 4[3] shows the dependence of the ERM and the SVM algorithms on mislabelled data.

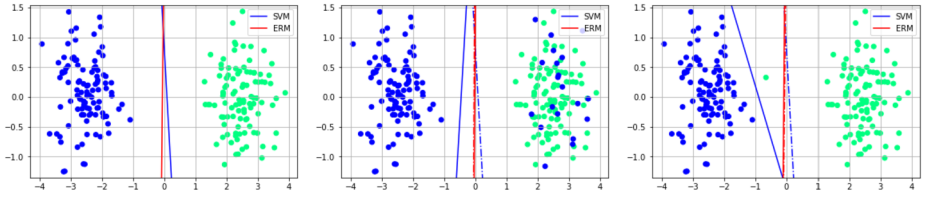

Figure 4 The resulting hyperplanes obtained by ERM and SVM by introducing mislabelled training data (the dotted lines represent the hyperplanes obtained in the first experiment)

[3]Here again, we illustrate our point with the classification task of planar points.

We analyse now the bounds $\|u\| \leq \Lambda$ and $|b| \leq \Theta$ in Theorem 3.8. While these were necessary to obtain the bounds in said theorem, they are also necessary for the ERM to converge in general. Let us first start with the bound $\|u\| \leq \Lambda$. As far as the separation of the means $\nu_{x_i,u}$ of the processed inputs is concerned, $\frac{u}{\|u\|}$ is responsible for geometrically separating these means as well as possible, while $\|u\|$ is an amplifying factor that could be exploited for adjusting to noise and randomness in the evaluation maps once a perfect separation is achieved. For example, if (u, ω, b) are parameters of the algorithm such that all the averages $\nu_{x_i,u}$ are correctly classified in the sense that

$$\min_{1\leq i\leq m} \nu_i\big(\langle \nu_{x_i,u}, \omega\rangle + b\big) \geq 0 \quad \text{and} \quad \max_{1\leq i\leq m} \nu_i\big(\langle \nu_{x_i,u}, \omega\rangle + b\big) > 0,$$

then

$$\lim_{\alpha\to+\infty} \widehat{R}_{(X,V)}(H_{\alpha u,\omega,\alpha b}) = 0.$$

In other words, once a training algorithm finds a pre-processing map u that enables a correct hyperplane separation of (the weighted means $\nu_{x,u}$ of) the data, it will tend to linearly scale the norm of u to infinity with two consequences: first, further distancing the means $\nu_{x_i,u}$ (or more precisely $A^{-1/2}\nu_{x_i,u}$, as seen earlier) from the classifying hyperplane and second, distancing these averages among themselves so that, with high probability, the random vectors $y(T, x_i)$s take their values in similar non-intersecting elliptic domains (becoming larger with α) centred at the $\nu_{x_i,u}$s (since the Gaussian random vectors $y(T, x_i)$s have the same covariance matrix A). Hence, if there are no bounds on the norm of u and if perfect classification is possible, the (non-converging) algorithm will try to transition from a robust classification to an accurate one (more details below). The bound $|b| \leq \Theta$ prevents in some particular scenarios the choice of hyperplanes whose distance from the origin tends to infinity. If we take, for example, the case where all the data belong to the same class (less extreme examples can also be stated), say "+1", and if (u, ω, b_0) are parameters of the algorithm such that all the averages $\nu_{x_i,u}$ are correctly classified, then

$$\lim_{b\to+\infty} \widehat{R}_{(X,V)}(H_{u,\omega,b_0+b}) = 0.$$

Note also that the bounds on the norm of u (and b) are interchangeable with a rescaling of the covariance matrix A as the classifier with parameters $(\alpha u, \omega, \alpha b)$ and covariance matrix A generates the same risk as the classifier with parameters (u, ω, b) and covariance matrix A/α^2. Hence, we choose the bound $\Lambda = 1$ in the numerical experiments and control the covariance matrix through a noise scale to be chosen carefully (see Sect. 5).

Finally, we discuss the condition of definiteness of the covariance matrix A and its role in the robustness of the classification algorithm. When A is positive-definite, we have seen that the goal of the ERM is solving the minimisation problem

$$\min_{\|u\|\leq\Lambda,|b|\leq\Theta,\omega} \frac{1}{m}\sum_{i=1}^{m} \Phi\left(-\nu_i \cdot \frac{\langle \nu_{x_i,u}, \omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right). \tag{6}$$

On the one hand, if the empirical risk is interpreted as a measure of the precision of the classification task on the training set, then we see that in this regime (A positive-definite),

a full precision (null risk) is not achievable (due to the randomness of the evaluation map). The same applies to the classification of the data on a validation set as the label is a random variable whose value will depend on the current simulation (of the Brownian motion and the solution to the S.D.E). On the other hand, and as we have previously seen, note that even for a fixed pre-processing map u, it is clear that the choice of the optimal parameters depends explicitly on all the training data, providing in this way a robustness against mislabelled data in the training set. For these reasons, and the ones detailed above, we call this the robust or the stochastic regime.

If we now take $A = 0$, the objective of an ERM becomes the solution of the minimisation problem

$$\min_{\|u\|\leq\Lambda,|b|\leq\Theta,\omega} \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\operatorname{sign}(\langle v_{x_i,u},\omega\rangle+b)\neq v_i}.$$

The task at hand is a mere classification of the averages $v_{x_i,u}$ of the processed data. If there exists a pre-processing map u such that these averages are separable, then an ERM algorithm will achieve a full precision on the training data set (and in general, a unique solution does not exist unless we introduce an additional constraint like maximising the margin of the classifying hyperplane, as in SVM). However, this classification will generally depend only on some support vectors $v_{x_i,u}$ (i.e. the closest averages to the classifying hyperplane) while ignoring the information provided by the rest of the training data, hence potentially becoming sensitive and vulnerable to mislabelling and the data-generating distribution. We call such a regime the accurate regime.

In the general case (A not positive-definite but not necessarily null), the ERM may choose either the robust or the accurate regime depending on how well the data can be separated. If the averages $v_{x_i,u}$ are separable, the accurate regime is preferred as it leads to a null empirical risk, otherwise the ERM may choose a robust solution. We refer, for example, to [11, 35, 38] for more information on the general topic of the trade-off between accuracy and robustness.

4.3 Information retained by stochastic RNNs

In this subsection, we highlight that the ERM is equivalent to a minimisation of a functional of the signatures of the augmented input paths $(t, \int x_t\,\mathrm{d}t)_{t\leq T}$. We recall that the signature $S(z) = (Z^n)_{n\in\mathbb{N}}$ of a path z defined over an interval $[0, T]$ with values in a Banach space E is the sequence of its iterated integrals, i.e. for all $s \leq t$,

$$\begin{cases} Z^0(s,t) = 1, \\ Z^1(s,t) = \int_{s\leq u\leq t} \mathrm{d}z(u) = z(t) - z(s) \in E, \\ Z^{n+1}(s,t) = \int_{s\leq u\leq t} Z^n(s,u)\otimes \mathrm{d}z(u) \in E^{\otimes(n+1)} \quad \text{for all } n\in\mathbb{N}. \end{cases}$$

The role of the signature in RNNs should not come as a surprise for two main reasons:

- The signature of a path is the only information needed from a path to solve a differential equation. This is the cornerstone of the theory of rough paths (cf. [29, 30]).
- Every path is uniquely characterised (up to what is called a tree-like equivalence) by its signature [18]. This fact is the basis for much research in the machine learning of data streams. These streams are mapped via their (truncated) signatures to vectors in the

tensor algebra, thus allowing one to use many of the classical machine-learning techniques (such as feed-forward neural networks) for data streams (e.g. [15, 21, 26]). For more information, we refer the reader to the original work of Chen [5], the modern formulation in [30] and the introduction to signature methods in machine learning in [6].

We now reformulate the ERM problem in a way that fully separates the tunable parameters of the network from the information provided by the paths in the training dataset.

Theorem 4.2 *Define* $\mathbb{H}$ *as the Hilbert space*

$$\mathbb{H} := \left\{ \big(a, (b_k)_{k\geq 0}\big) \in \mathbb{R} \oplus \bigoplus_{k\geq 0} \mathbb{R}^r : \sum_{k\geq 0} k! \|b_k\|_2^2 < \infty \right\},$$

with inner product given by

$$\big\langle \big(a, (b_k)_{k\geq 0}\big), \big(c, (d_k)_{k\geq 0}\big) \big\rangle_{\mathbb{H}} = ac + \sum_{k\geq 0} k! b_k^\top d_k \quad \textit{for all } \big(a, (b_k)_{k\geq 0}\big), \big(c, (d_k)_{k\geq 0}\big) \in \mathbb{H}.$$

Then, the ERM is equivalent to the following optimisation problem

$$\min_{(\beta,(\theta_k)_{k\geq 0})\in \mathcal{C}_A} \sum_{i=1}^m \Phi\left(-\nu_i \left\langle \big(\beta, (\theta_k)_{k\geq 0}\big), \left(1, \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right)_{k\geq 0} \right) \right\rangle_{\mathbb{H}} \right),$$

where $\mathcal{C}_A$ *is defined as*

$$\begin{aligned} \mathcal{C}_A &:= \mathcal{C}_{A,\Lambda,\Theta} \\ &:= \left\{ \big(\beta, (\theta_k)_{k\geq 0}\big) \in \mathbb{H} : \beta \in \mathbb{R}, \theta_k = u^\top \frac{(W_0^\top)^k}{k!} \alpha, u \in \mathbb{R}^{n\times r}, \right. \\ &\qquad \left. \|u\| \leq \Lambda, |\beta| \leq \frac{\Theta}{\sqrt{\lambda_{\min}(A)}}, \alpha \in \mathbb{R}^n, \alpha^\top A \alpha = 1 \right\}. \end{aligned}$$

Proof We write the problem at hand

$$\min_{\|u\|\leq\Lambda, |b|\leq\Theta, \omega} \sum_{i=1}^m \Phi\left(-\nu_i \cdot \frac{\langle \omega, \nu_{x_i,u} \rangle + b}{\sqrt{\omega^\top A \omega}} \right) \quad \text{s.t. } \omega \neq 0,$$

in the simpler equivalent form

$$\min_{\|u\|\leq\Lambda, |\beta|\leq \frac{\Theta}{\sqrt{\lambda_{\min}(A)}}, \omega} \sum_{i=1}^m \Phi\big(-\nu_i\big(\langle \alpha, \nu_{x_i,u} \rangle + \beta\big)\big) \quad \text{s.t. } \alpha^\top A \alpha = 1.$$

Note now that we can expand the mean $\nu_{x,u}$ into a series

$$\nu_{x,u} = \int_0^T e^{W_0(T-s)} u\big(x(s)\big)\, \mathrm{d}s = \sum_{k=0}^\infty k! \frac{W_0^k}{k!} u\left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right),$$

so that one can separate, in the inner product $\langle \alpha, \nu_{x,u} \rangle + \beta$, the tunable parameters from a functional of the input signals

$$\begin{aligned}\langle \alpha, \nu_{x,u} \rangle + \beta &= \sum_{k=0}^{\infty} k! \alpha^{\top} \frac{W_0^k}{k!} u\left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right) + \beta \\ &= \left\langle \left(\beta, \left(u^{\top} \frac{(W_0^{\top})^k}{k!} \alpha \right)_{k\geq 0} \right), \left(1, \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right)_{k\geq 0} \right) \right\rangle_{\mathbb{H}}.\end{aligned}$$

This concludes the proof. □

Remark 4.3 Note that

$$\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s = \int_{0<s<u_1<\cdots<u_k<T} \mathrm{d}X(s)\, \mathrm{d}u_1 \cdots \mathrm{d}u_k,$$

where $X_\cdot := \int_0^\cdot x(s)\, \mathrm{d}s$ denotes a primitive of x. Hence, the sequence $(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s)_{k\geq 0}$ can be obtained from the signature of the path $(t, X_t)_{t\leq T}$.

Theorem 4.2 highlights the information retained by a stochastic RNN (with architecture and loss function chosen as described in this paper). More explicitly, we have the following theorem.

Theorem 4.4 *Both during training* (*solving the ERM problem*) *and classification* (*generating labels for unseen data*), *the continuous-time stochastic RNN* (*with linear activation function*) *is uniquely determined by the partial signature* $\widehat{S}$ *of the input path defined by*

$$\widehat{S}(x) := \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right)_{k\geq 0}. \tag{7}$$

Proof We have already shown in Theorem 4.2 that the ERM problem can be written as a minimisation problem of a functional of the partial signatures of the training paths (and that no other information about said paths is required). If we denote by (u, ω, b) some possible parameters of an ERM classifier H^{ERM} and given an input path x, then the RNN generates the label $\nu = \operatorname{sign}(\langle y_u(T,x), \omega \rangle + b)$ with $y_u(T,x) \sim \mathcal{N}(\nu_{x,u}, A)$. Following the proof of Theorem 4.2, $\nu_{x,u}$ can also be expressed as a functional of $\widehat{S}(x)$ only

$$\nu_{x,u} = \sum_{k=0}^{\infty} W_0^k u\left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right).$$

This completes the proof of the claims. □

We believe that a similar result to Theorem 4.4 holds for generic RNNs, i.e. that continuous-time RNNs (even with non-linear activation functions) can be viewed as kernel machines involving the (full) signatures of the input paths. We refer, for example, to [28] (and the references therein) for the first results in this direction.

Even though the RNN uses only the partial signature (7) of the time-lifted input paths (instead of the full signature), it turns out that it is still a faithful representation of continuous paths.

Theorem 4.5 *The partial signature map $\widehat{S}$ defined over the space of continuous paths defined over an interval $[0, T]$ and with values in a finite-dimensional space is injective.*

Proof Without loss of generality, we will consider the case of real-valued paths. By a simple change of variable and a reparametrisation of the path, it is equivalent to show that the map

$$\widetilde{S}(x) : x \longmapsto \left(\int_0^T s^k x(s)\, \mathrm{d}s \right)_{k \geq 0}$$

is injective. As $\widetilde{S}$ is linear it is also equivalent to show that $\widetilde{S}(x) = 0$ if and only if $x = 0$. Let then x be a continuous real-valued path such that $\widetilde{S}(x) = 0$. Then, for every polynomial function P, one has $\int_0^T P(s)x(s)\, \mathrm{d}s = 0$. Let $\varepsilon > 0$ be arbitrary and let P be a polynomial function such that $\|x - P\|_{\infty,[0,T]} \leq \varepsilon$. Then, one has

$$\left| \int_0^T x^2(s)\, \mathrm{d}s \right| = \left| \int_0^T x(s)\big(x(s) - P(s)\big)\, \mathrm{d}s \right| \leq \varepsilon \|x\|_{\infty,[0,T]} T.$$

Therefore, $\int_0^T x^2(s)\, \mathrm{d}s = 0$, from which we conclude that indeed $x = 0$. □

Remark 4.6 Despite the partial signature transform being injective, it is still possible for the linear RNN not to be able to distinguish two paths x and $\widetilde{x}$ if the inner product of their partial signatures with the matrices $(W_0^k u)_{k \geq 0}$ are the same, i.e.

$$v_{x,u} = \sum_{k=0}^{\infty} W_0^k u \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right) = \sum_{k=0}^{\infty} W_0^k u \left(\int_0^T \frac{(T-s)^k}{k!} \widetilde{x}(s)\, \mathrm{d}s \right) = v_{\widetilde{x},u}.$$

Replacing the sequence $(W_0^k u)_{k \geq 0}$ by another whose entries can be independent is the cornerstone and the reason behind the power of signature techniques. However, these techniques are confronted with computational issues and therefore the signature has to be restricted to low orders. RNNs are able, however, to surmount this obstacle by increasing the dimension n and thus allowing for more degrees of freedom (while signatures are computed implicitly, as shown above through the recursive architecture).

Using the factorial decay of the elements of the partial signature sequence (which is trivial and explicit in our case), we can then obtain a good approximation for the ERM by truncating the signature. To show such a result, we first prove the following technical lemma.

Lemma 4.7 *Let $N \in \mathbb{N}^*$, $u \in \mathbb{R}^{n \times r}$, $\omega \in \mathbb{R}^n$ and $x \in L^1([0, T], (\mathbb{R}^r, \|\cdot\|_2))$. Then,*

$$\left| \langle \omega, v_{x,u} \rangle - \sum_{k=0}^{N} \omega^\top W_0^k u \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right) \right| \leq \|\omega\|_2 \|u\| e^{\|W_0\| T} \frac{(\|W_0\| T)^{N+1}}{(N+1)!} \int_0^T \big\|x(s)\big\|_2\, \mathrm{d}s.$$

Proof Expanding the difference that we want to bound we obtain

$$\langle \omega, v_{x,u} \rangle - \sum_{k=0}^{N} \omega^\top W_0^k u \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right) = \sum_{k=N+1}^{\infty} \omega^\top W_0^k u \left(\int_0^T \frac{(T-s)^k}{k!} x(s)\, \mathrm{d}s \right).$$

For every $k \in \mathbb{N}$, one trivially has

$$\left|\omega^\top W_0^k u\left(\int_0^T \frac{(T-s)^k}{k!} x(s)\,\mathrm{d}s\right)\right| \leq \|\omega\|_2 \|W_0\|^k \|u\| \frac{T^k}{k!} \int_0^T \left\|x(s)\right\|_2 \mathrm{d}s,$$

while for every $a \geq 0$

$$\left|\sum_{k=N+1}^{\infty} \frac{a^k}{k!}\right| \leq e^a \frac{a^{N+1}}{(N+1)!}.$$

The combination of the three arguments above then gives the desired result. □

We can now show how one may exploit the signature-based expansion of the empirical risk in order to obtain an approximate solution to the ERM.

Theorem 4.8 *Let Θ, Λ and R be positive real numbers and $N \in \mathbb{N}^*$. We consider the set of parameters*

$$\mathcal{P}_{\Theta,\Lambda} = \left\{(u,\omega,b) : \|\omega\|_2 = 1, |b| \leq \Theta, \|u\| \leq \Lambda\right\}.$$

Assume that all input paths take their values in the L^1-ball of radius R

$$\mathcal{B}_R = \left\{x \in \mathcal{C}\left([0,T],\mathbb{R}^r\right) : \int_0^T \left\|x(s)\right\|_2 \mathrm{d}s \leq R\right\}.$$

Given a sample $(X,V) = \{(x_i,v_i)\}_{i=1}^m$, let (u_0,ω_0,b_0) be a solution to the ERM problem

$$\min_{(u,\omega,b)\in\mathcal{P}_{\Theta,\Lambda}} \frac{1}{m}\sum_{i=1}^m \Phi\left(-v_i \cdot \frac{\langle v_{x_i,u},\omega\rangle + b}{\sqrt{\omega^\top A\omega}}\right),$$

and $(\bar{u},\bar{\omega},\bar{b})$ be a solution to the "truncated" ERM problem

$$\min_{(u,\omega,b)\in\mathcal{P}_{\Theta,\Lambda}} \frac{1}{m}\sum_{i=1}^m \Phi\left(-v_i \cdot \frac{\sum_{k=0}^N \omega^\top W_0^k u(\int_0^T \frac{(T-s)^k}{k!} x(s)\,\mathrm{d}s) + b}{\sqrt{\omega^\top A\omega}}\right).$$

Then,

$$0 \leq \widehat{R}_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) - \widehat{R}_{(X,V)}(H_{u_0,\omega_0,b_0}) \leq \Lambda R e^{\|W_0\|T} \sqrt{\frac{2}{\lambda_{\min}(A)\pi}} \frac{(\|W_0\|T)^{N+1}}{(N+1)!}.$$

Proof For lighter expressions, we will introduce the following notation

$$\widehat{R}^N_{(X,V)}(H_{u,\omega,b}) = \frac{1}{m}\sum_{i=1}^m \Phi\left(-v_i \cdot \frac{\sum_{k=0}^N \omega^\top W_0^k u(\int_0^T \frac{(T-s)^k}{k!} x(s)\,\mathrm{d}s) + b}{\sqrt{\omega^\top A\omega}}\right).$$

The inequality $\widehat{R}_{(X,V)}(H_{u_0,\omega_0,b_0}) \leq \widehat{R}_{(X,V)}(\bar{u},\bar{\omega},\bar{b})$ is a direct consequence of the definition of (u_0,ω_0,b_0). We decompose the difference of these two terms in the following way

$$\begin{aligned}\widehat{R}_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) - \widehat{R}_{(X,V)}(H_{u_0,\omega_0,b_0}) &= \widehat{R}_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) - \widehat{R}^N_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) \\ &\quad + \widehat{R}^N_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) - \widehat{R}^N_{(X,V)}(H_{u_0,\omega_0,b_0}) \\ &\quad + \widehat{R}^N_{(X,V)}(H_{u_0,\omega_0,b_0}) - \widehat{R}_{(X,V)}(H_{u_0,\omega_0,b_0}).\end{aligned}$$

By the definition of $(\bar{u},\bar{\omega},\bar{b})$, the second difference is non-positive,

$$\widehat{R}^N_{(X,V)}(H_{\bar{u},\bar{\omega},\bar{b}}) - \widehat{R}^N_{(X,V)}(H_{u_0,\omega_0,b_0}) \leq 0.$$

Let $(u,\omega,b) \in \mathcal{P}_{\Theta,\Lambda}$. As Φ is $\frac{1}{\sqrt{2\pi}}$-Lipschitz, we have

$$\begin{aligned}&\left|\widehat{R}_{(X,V)}(H_{u,\omega,b}) - \widehat{R}^N_{(X,V)}(H_{u,\omega,b})\right| \\ &\quad \leq \frac{1}{\sqrt{2\pi}\,m}\sum_{i=1}^{m}\left|\frac{\langle v_{x_i,u},\omega\rangle - \sum_{k=0}^{N}\omega^\top W_0^k u\left(\int_0^T \frac{(T-s)^k}{k!}x(s)\,\mathrm{d}s\right)}{\sqrt{\omega^\top A\omega}}\right|.\end{aligned}$$

For each $i \in [\![1,m]\!]$, the following holds by Lemma 4.7 and the assumptions on the parameters ω and u and the path x_i

$$\left|\langle v_{x_i,u},\omega\rangle - \sum_{k=0}^{N}\omega^\top W_0^k u\left(\int_0^T \frac{(T-s)^k}{k!}x(s)\,\mathrm{d}s\right)\right| \leq \Lambda R e^{\|W_0\|T}\frac{(\|W_0\|T)^{N+1}}{(N+1)!}.$$

The result is then directly obtained by applying the above bounds to the vectors $(\bar{u},\bar{\omega},\bar{b})$ and (u_0,ω_0,b_0). □

Theorem 4.8 demonstrates then the possible power of the application of the signature-based decomposition of the empirical risk. In our setting, this technique avoids, for example, the expensive computation of $v_{x,u}$ (as integrals of time-dependent matrix exponentials) for each path in the training set and replaces it with the computation and storage of the powers $(W_0^k)_{k\leq N}$. However, in this still simple setting, we will not base our optimisation techniques on this method and will reserve its application for the non-linear case where exact formulae are not available.

5 Numerical results

In this subsection, we present the results of some numerical experiments run on real and synthetic data. The focus will be on the effect of noise on the accuracy and the robustness of the RNN and on the verification of the theoretical bound obtained in Theorem 3.8.

The application to real-world data will be demonstrated on the Japanese Vowels dataset.[4] This dataset contains speech recordings of nine male subjects pronouncing a combination of Japanese vowels. The recordings are in the form of 12-dimensional ($r = 12$) discrete time series with varying lengths. To create a binary classification problem of

[4] https://archive.ics.uci.edu/ml/datasets/Japanese+Vowels

continuous-time paths, we restrict ourselves to the recordings of the first two subjects and we reparametrise the time series to the unit interval.

For the synthetic data, we consider 5-dimensional trigonometric polynomials ($r = 5$)

$$x(t) = \sum_{k=0}^{N} a_k \cos(kt) + \sum_{k=1}^{N} b_k \sin(kt)$$

defined over the interval $[0, 2\pi]$. Here, we take $N = 6$ and the parameters a_k and b_k are chosen (uniformly randomly) in the intervals I^r and J^r, respectively, where, for one class $I = [-0.2, 1]$ and $J = [-1, 0.2]$ and for the other $I = [-1, 0.2]$ and $J = [-0.2, 1]$. We generate 70 samples for each class for our dataset.

We take the dimension of the network to be $n = 50$ (and the dimension d of the Brownian motion is taken to be equal to n). This is quite large for our simple synthetic dataset and appropriate for the Japanese Vowels dataset but still small compared to typical reservoirs, where n can be larger than 500 to make the reservoir self-averaging. The entries of the connectivity matrix W are drawn from independent Gaussian distributions $\mathcal{N}(0, \frac{0.9}{\sqrt{n}})$ (the choice of the value 0.9 is to insure the numerical stability of the system). To generate the noise matrix Σ, we first draw positive values $\lambda_1, \ldots, \lambda_n$ uniformly over $(0, 1)$, then uniformly a random orthogonal matrix U to output the matrix

$$\Sigma = \delta \cdot U^\top \operatorname{diag}(\lambda_1, \ldots, \lambda_n) U. \tag{8}$$

The parameter δ is a noise scale that has to be chosen carefully. A large value for δ makes the noise dominant and the RNN unable to reach an acceptable accuracy, while too small a value makes the system less robust.

The ERM problem will be solved numerically using a gradient-descent (GD) algorithm. Recall that, given a training sample $(X, V) = \{(x_i, v_i)\}_{i=1}^m$, we aim to minimise the quantity

$$\frac{1}{m} \sum_{i=1}^{m} \Phi\left(-v_i \cdot \frac{\langle v_{x_i,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}} \right),$$

as a function of the parameters u, ω and b. As discussed in Sect. 4.2, we impose the bound $\|u\| \leq 1$. To achieve a superior classification performance, we do not impose an a priori bound on b. Let us first say a few words about some simplifications in the implementation of the gradient-descent (GD) algorithm in this simple linear case. If we consider the loss function associated with a single observation (x, v) as a function of u, ω and b

$$L(u, \omega, b) = \Phi\left(-v \frac{\langle v_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}} \right),$$

then we may explicitly compute the gradient of L (for $i \in [\![1, n]\!]$ and $j \in [\![1, r]\!]$)

$$\begin{cases} \frac{\partial L}{\partial u_{i,j}}(u, \omega, b) = -\frac{v}{\sqrt{2\pi}\sqrt{\omega^\top A \omega}} \exp\left(-\frac{1}{2}\left(\frac{\langle v_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}\right)^2\right) \langle \omega, v_{x,e_{i,j}} \rangle, \\ \frac{\partial L}{\partial \omega_i}(u, \omega, b) = -\frac{v}{\sqrt{2\pi}\sqrt{\omega^\top A \omega}} \exp\left(-\frac{1}{2}\left(\frac{\langle v_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}\right)^2\right) \left((v_{x,u})_i - \frac{(A\omega)_i (\langle \omega, v_{x,u} \rangle + b)}{\omega^\top A \omega} \right), \\ \frac{\partial L}{\partial b}(u, \omega, b) = -\frac{v}{\sqrt{2\pi}\sqrt{\omega^\top A \omega}} \exp\left(-\frac{1}{2}\left(\frac{\langle v_{x,u}, \omega \rangle + b}{\sqrt{\omega^\top A \omega}}\right)^2\right), \end{cases}$$

where $e_{i,j}$ stands for the canonical $n \times r$ basis matrix $(\delta_{ik}\delta_{jl})_{k\leq n,l\leq r}$ and z_i stands for the ith coordinate of the vector z. Apart from its simple expression, one can see that the averages $(\nu_{x,e_{i,j}})_{i\leq n,j\leq r}$ need to be computed for each example x in the training set in order to run the algorithm. One can then exploit these to compute the mean $\nu_{x,u}$ at each iteration of the algorithm as a linear combination of the means $(\nu_{x,e_{i,j}})_{i\leq n,j\leq r}$ instead of a full new computation that can be heavy due to the presence of matrix exponentials. In our implementation, we used Scipy's minimise function with nonlinear constraints[5] that is a gradient-descent algorithm with suitably adjusted step sizes (based on the Sequential Least Squares Programming algorithm detailed in [24]). Due to the non-convexity of the problem, the algorithm is run a few times (typically 5 times) with random initialisations of the parameters in order to avoid poor local minima. The solution with the smallest loss function is then chosen.

The experiments consist of three parts:

(1) First, we check the accuracy achieved on a fixed-size testing set (30% of the total available data) while increasing the size of the training set. Recall that the computed accuracies are random realisations depending on the results of the simulations (of the solution to the SDE (1)).

(2) We verify the validity of the theoretical bound obtained in Theorem 3.8 as a function of the size of the training set. More precisely, having computed some values for the parameters u, ω and b, and thus the corresponding hypothesis H, we compare the difference $|R(H) - \widehat{R}_{(X,V)}(H)|$ with the theoretical bound (which is valid with high probability)

$$\frac{4}{\sqrt{2\pi m\lambda_{\min}(A)}}\left(\Theta + R\left(\int_0^t \left\|e^{W_0(t-s)}\right\|^2 \mathrm{d}s\right)^{1/2}\right) + \frac{2+5\sqrt{\log(2/\delta)/2}}{\sqrt{m}}.$$

In the above, the true risk $R(H)$ is computed as an empirical risk over the testing set. The value R is computed as the maximum of the L^2-norm of the paths in the whole dataset, while Θ is computed as the largest value obtained for $|b|$ in the accuracy experiment detailed above. We take the value $\delta = 0.01$ so that the bound holds with probability 0.99.

(3) We verify the robustness of the learning in the following way. We train the network with a corrupt dataset where a portion of the labels of the training inputs has been changed, then compute the resulting accuracy based on the test set. This robustness test is quite simple and does not show the full potential of the architecture, but we suspect that such robustness also holds when the input paths are themselves being perturbed. For other measures of robustness, we refer the reader to the previously cited references and the additional references [1, 4, 8] that investigate the robustness of training with corrupted datasets.

We start by the experiments on the Japanese vowels dataset with three choices of noise scales $\delta \in \{1, 1.5, 2.5\}$ in (8). The resulting plots are shown in Fig. 5. In the accuracy plots:

- the solid lines show the accuracy based on classifying the averages $\nu_{x,u}$ of the paths in the test dataset with the obtained hyperplane $h_{\omega,b}$. In other words, the generated label

[5] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

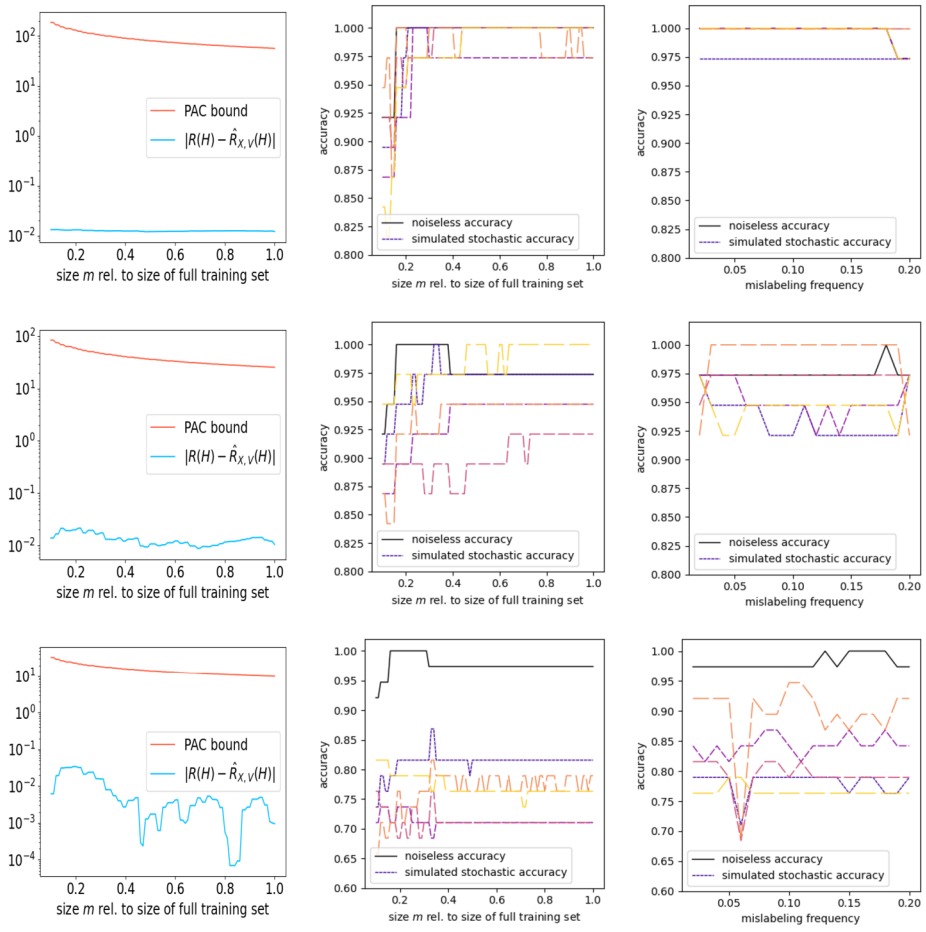

Figure 5 Verification of the theoretical PAC-bound (Theorem 3.8) and accuracy of the network for the Japanese vowels dataset. Each row of figures represents the results for different noise scales (as defined in Equation (8)); 1, 1.5 and 2.5, respectively. The middle column shows the evolution of the accuracy of the network when increasing the size of the training dataset. The right column shows how an increasing portion (described as mislabelling frequency) of corrupt labels in the training dataset affects the accuracy on the test set. Solid lines show the accuracy of classification by a noiseless RNN (trained as a stochastic RNN). Dotted lines show the results of simulations of the realised labels by the stochastic RNN

for a path x is $v = \text{sign}(\langle v_{x,u}, \omega \rangle + b)$. As $v_{x,u}$ is the hidden state of a noiseless RNN,[6] we refer to this scenario as noiseless accuracy or NRNN.

- The dotted lines show the accuracy of 5 simulated labels produced by the stochastic RNN. Here, the generated label for a path x is $v = \text{sign}(\langle y_u(T,x), \omega \rangle + b)$, where $y_u(T,x)$ is simulated as $y_u(T,x) \sim \mathcal{N}(v_{x,u}, A)$. We refer to this scenario as simulated stochastic accuracy.

Note that because of the discontinuity of the hyperplane classifying function, the former is not expected to be an average of the latter. For readability, we report some numerical values for 10 simulations in Table 1:

[6]Note, however, that a noiseless RNN will train differently and consequently produce different parameters u, ω and b in general.

Table 1 Numerical values when working the Japanese vowels dataset. The first column shows the noise scale (see Equation (8)), the next four columns give, respectively, the accuracy of classification by a noiseless RNN (trained as a stochastic RNN), then the lowest, the largest and the average accuracy given by the (stochastic) RNN over 10 simulations. The following four columns show the same numbers after training the RNN with 10% of mislabelled data. The last column shows the ratio of the average accuracy in the robustness test (after training with some corrupted labels) over the normal average accuracy (after training with the correct labels)

Noise scale	Accuracy				Robustness test accuracies				Ratio
	NRNN	min	max	avg.	NRNN	min	max	avg.	
1	100%	97.37%	100%	99.21%	100%	94.74%	100%	99.21%	100%
1.5	97.37%	86.84%	100%	95.00%	97.37%	92.11%	100%	95.79%	100.83%
2.5	97.37%	65.79%	86.84%	75.26%	97.37%	76.32%	94.74%	83.95%	111.54%

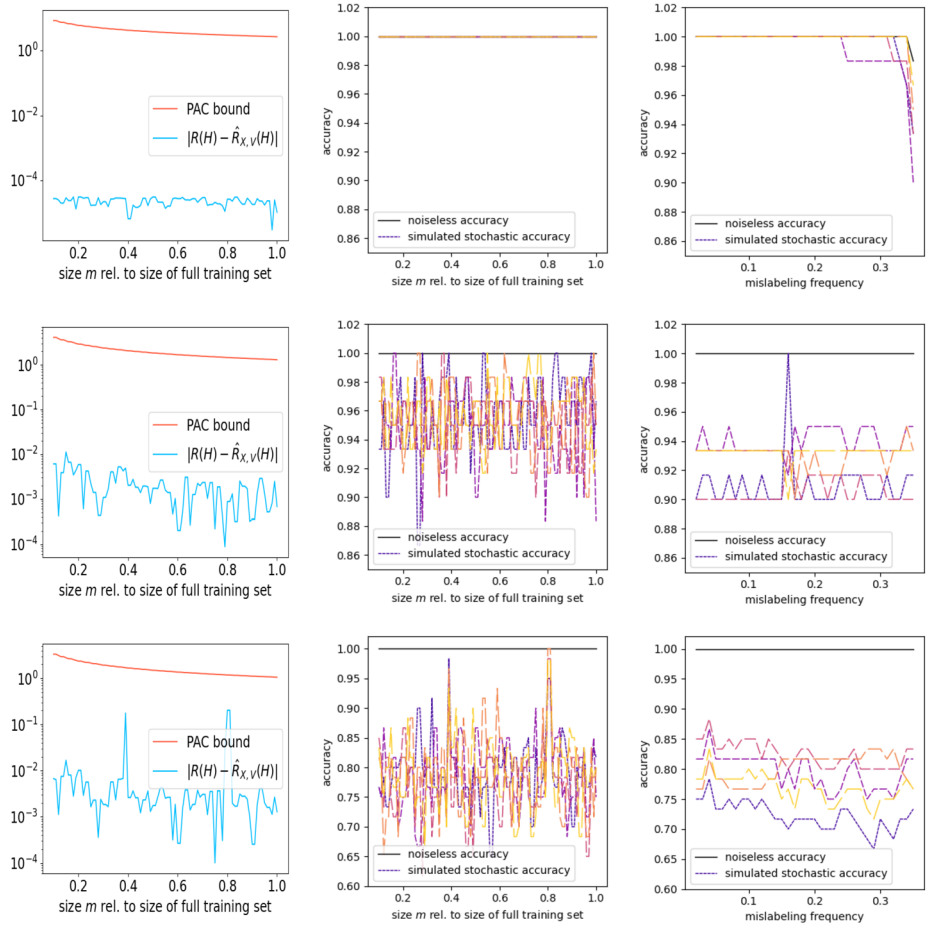

Figure 6 Verification of the theoretical PAC-bound (Theorem 3.8) and accuracy of the network for the synthetic dataset. Each row of figures represents the results for different noise scales (as defined in Equation (8)); 2, 4 and 6, respectively. The middle column shows the evolution of the accuracy of the network on increasing the size of the training dataset. The right column shows how an increasing portion (described as mislabelling frequency) of corrupt labels in the training dataset affects the accuracy on the test set. Solid lines show the accuracy of classification by a noiseless RNN (trained as a stochastic RNN). Dotted lines show the results of simulations of the realised labels by the stochastic RNN

- the first column refers to the value of the noise scale δ in (8);
- the group of columns "Accuracy" reports the results of the experiment after training with the entire training set, while the group of columns "Robustness test accuracies"

Table 2 Numerical values when working the synthetic dataset. The first column shows the noise scale (see Equation (8)), the next four columns give, respectively, the accuracy of classification by a noiseless RNN (trained as a stochastic RNN), then the lowest, the largest and the average accuracy given by the (stochastic) RNN over 10 simulations. The following four columns show the same numbers after training the RNN with 10% of mislabelled data. The last column shows the ratio of the average accuracy in the robustness test (after training with some corrupted labels) over the normal average accuracy (after training with the correct labels)

Noise scale	Accuracy				Robustness test accuracies				Ratio
	NRNN	min	max	avg.	NRNN	min	max	avg.	
2	100%	100%	100%	100%	100%	100%	100%	100%	100.00%
4	100%	88.33%	98.33%	94.83%	100%	90.00%	100.00%	92.83%	97.89%
6	100%	73.33%	90.00%	79.83%	100%	71.67%	81.67%	76.67%	96.03%

reports the accuracy of the network after training with the entire training set with 10% of mislabelled data;

- the column NRNN gives the accuracy of classification of the averages $v_{x,u}$ with the hyperplane $h_{\omega,b}$ (these labels are given by $\nu = \operatorname{sign}(\langle v_{x,u}, \omega\rangle + b)$);
- the columns "min", "max" and "avg." give, respectively, the smallest, the largest and the average observed accuracy over the 10 simulations (the labels are simulated as $\nu \sim \operatorname{sign}(\langle \mathcal{N}(v_{x,u}, A), \omega\rangle + b)$).

As usually observed when working with PAC-bounds based on the Rademacher complexity, the theoretical bound obtained in Theorem 3.8 can be very loose. We note, however, in Fig. 5 that the difference becomes smaller with larger noise scales. In Table 1, we note an almost perfect noiseless accuracy even in the presence of noise. A striking fact, however, is that, on average and in the presence of noise, there is no marked loss in the performance of the RNN when training with some mislabelled data.

Training with synthetic data showed a higher tolerance to the noise scales. In Fig. 6, we show the results of the same experiments as before with the noise scales $\delta \in \{2, 4, 6\}$. Numerical values are reported in Table 2 (with 15% for mislabelled data in the robustness test). We observe again the validity of the theoretical PAC-bound obtained in Theorem 3.8 (with the difference becoming smaller with higher noise scales) and note the same persistence in the quality of the labels generated by the RNN when training with mislabelled data.

Acknowledgements
YB, SN and HR are grateful for the financial support of the Excellence Initiative of the German federal and state governments through its grant G:(DE-82)EXS-SFneurolC002: "Recurrence and stochasticity for neuro-inspired computation".

Funding
The research of YB, SN and HR was supported by the Excellence Initiative of the German federal and state governments (G:(DE-82)EXS-SFneurolC002: "Recurrence and stochasticity for neuro-inspired computation"). Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials
The datasets analysed during the current study are available in the UCI Machine Learning Repository.

Declarations

Competing interests
The authors declare that they have no competing interests.

Authors' contributions

HR acquired the funding, initiated the idea of the project and helped in directing the research and inspecting the final version of the manuscript. YB proved the theoretical results of this paper, drafted the manuscript and designed the numerical experiments. WB carried out the numerical experiments of Sects. 4.1 and 4.2 and prepared the corresponding figures. SN carried out the numerical experiments of Sect. 5 and prepared the corresponding figures along with the sketch of Fig. 1. All the authors have read and agreed to the published version of the manuscript.

Author details

[1]Chair for Mathematics of Information Processing, RWTH Aachen University, Pontdriesch 10, 52062, Aachen, Germany. [2]Institute of Neuroscience and Medicine (INM-6) and Institute for Advanced Simulation (IAS-6) and JARA-Institute "Brain Structure-Function Relationships" (INM-10), Jülich Research Centre, Jülich, Germany. [3]RWTH Aachen University, Aachen, Germany.

Publisher's Note

Received: 21 September 2021 Accepted: 16 January 2022 Published online: 02 February 2022

References

1. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 16–25 (2006)
2. Bartlett, P.L., Boucheron, S., Lugosi, G.: Model selection and error estimation. Mach. Learn. **48**(1), 85–113 (2002)
3. Bhattacharya, K., Hosseini, B., Kovachki, N.B., Stuart, A.M.: Model reduction and neural networks for parametric PDEs. SMAI J. Comput. Math. **7**, 121–157 (2020)
4. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines (2012)
5. Chen, K.-T.: Integration of paths, geometric invariants and a generalized Baker–Hausdorff formula. Ann. Math. (2) **65**, 163–178 (1957)
6. Chevyrev, I., Kormilitzin, A.: A primer on the signature method in machine learning. arXiv preprint (2016). arXiv:1603.03788
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
8. Dalvi, N., Domingos, P., Sanghai, S., Verma, D.: Adversarial classification. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 99–108 (2004)
9. DePasquale, B., Cueva, C.J., Rajan, K., Escola, G.S., Abbott, L.: full-FORCE: a target-based method for training recurrent networks. PLoS ONE **13**(2), e0191527 (2018)
10. Doya, K.: Bifurcations in the learning of recurrent neural networks 3. Learn. RTRL **3**, 17 (1992)
11. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. Mach. Learn. **107**(3), 481–508 (2018)
12. Funahashi, K.-I., Nakamura, Y.: Approximation of dynamical systems by continuous time recurrent neural networks. Neural Netw. **6**(6), 801–806 (1993)
13. Gonon, L., Grigoryeva, L., Ortega, J.-P.: Approximation bounds for random neural networks and reservoir systems. arXiv preprint (2020). arXiv:2002.05933
14. Gonon, L., Grigoryeva, L., Ortega, J.-P.: Risk bounds for reservoir computing. J. Mach. Learn. Res. **21**, Paper No. 240, 61 pp. (2020)
15. Graham, B.: Sparse arrays of signatures for online character recognition. arXiv preprint (2013). arXiv:1308.0371
16. Gressmann, F., Király, F.J., Mateen, B., Oberhauser, H.: Probabilistic supervised learning. arXiv preprint (2018). arXiv:1801.00753
17. Hälvä, H., Hyvarinen, A.: Hidden Markov nonlinear ICA: unsupervised learning from nonstationary time series. In: Conference on Uncertainty in Artificial Intelligence. PMLR, pp. 939–948 (2020)
18. Hambly, B., Lyons, T.: Uniqueness for the signature of a path of bounded variation and the reduced path group. Ann. Math. (2) **171**(1), 109–167 (2010)
19. Hyvarinen, A., Morioka, H.: Nonlinear ICA of temporally dependent stationary sources. In: Artificial Intelligence and Statistics. PMLR, pp. 460–469 (2017)
20. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
21. Király, F.J., Oberhauser, H.: Kernels for sequentially ordered data. J. Mach. Learn. Res. **20**, Paper No. 31, 45 pp. (2019)
22. Koiran, P., Sontag, E.D.: Vapnik–Chervonenkis dimension of recurrent neural networks. Discrete Appl. Math. **86**(1), 63–79 (1998)
23. Koltchinskii, V., Panchenko, D.: Rademacher processes and bounding the risk of function learning. In: High Dimensional Probability, II (Seattle, WA, 1999). Progr. Probab., vol. 47, pp. 443–457. Birkhäuser, Boston (2000)
24. Kraft, D., et al.: A software package for sequential quadratic programming
25. Ledoux, M., Talagrand, M.: Probability in Banach Spaces. Ergebnisse der Mathematik und ihrer Grenzgebiete (3), vol. 23. Springer, Berlin (1991)
26. Levin, D., Lyons, T., Ni, H.: Learning from the past, predicting the statistics for the future, learning an evolving system. arXiv preprint (2013). arXiv:1309.0260
27. Li, Z., Han, J., E, W., Li, Q.: On the curse of memory in recurrent neural networks: approximation and optimization analysis. In: International Conference on Learning Representations (2021)
28. Lim, S.H.: Understanding recurrent neural networks using nonequilibrium response theory. J. Mach. Learn. Res. **22**, 1–48 (2021)
29. Lyons, T.J.: Differential equations driven by rough signals. Rev. Mat. Iberoam. **14**(2), 215–310 (1998)
30. Lyons, T.J., Caruana, M., Lévy, T.: Differential equations driven by rough paths. In: Lectures from the 34th Summer School on Probability Theory Held in Saint-Flour, July 6–24, 2004. Lecture Notes in Mathematics, vol. 1908, pp. 6–24. Springer, Berlin (2007). With an introduction concerning the Summer School by Jean Picard

31. Maass, W., Joshi, P., Sontag, E.D.: Computational aspects of feedback in neural circuits. PLoS Comput. Biol. **3**(1), e165 (2007)
32. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge (2012)
33. Nestler, S., Keup, C., Dahmen, D., Gilson, M., Rauhut, H., Helias, M.: Unfolding recurrence by Green's functions for optimized reservoir computing. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 17380–17390. Curran Associates, Red Hook (2020)
34. Oberhauser, H., Schell, A.: Nonlinear independent component analysis for continuous-time signals. arXiv preprint (2021). arXiv:2102.02876
35. Raghunathan, A., Xie, S.M., Yang, F., Duchi, J., Liang, P.: Understanding and mitigating the tradeoff between robustness and accuracy. arXiv preprint (2020). arXiv:2002.10716
36. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
37. Sompolinsky, H., Crisanti, A., Sommers, H.-J.: Chaos in random neural networks. Phys. Rev. Lett. **61**(3), 259–262 (1988)
38. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: International Conference on Learning Representations (2019)