MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Fault Detection and Prognosis of Time Series Data with Random Projection Filter Bank

Pourazarm, Sepideh; Farahmand, Amir-massoud; Nikovski, Daniel N.

TR2017-142 October 02, 2017

Abstract

We introduce Random Projection Filter Bank (RPFB) as a general framework for feature extraction from time series data. RPFB is a set of randomly generated stable autoregressive filters that are convolved with the input time series. Filters in RPFB extract different aspects of the time series, and together they provide a reasonably good summary of the time series. These features can then be used by any conventional machine learning algorithm for solving tasks such as time series prediction, and fault detection and prognosis with time series data. RPFB is easy to implement, fast to compute, and parallelizable. Through a series of experiments we show that RPFB alongside conventional machine learning algorithms can be effective in solving data-driven fault detection and prognosis problems.

Prognostics and Health Management Society (PHM) 2017

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Fault Detection and Prognosis of Time Series Data with Random Projection Filter Bank

Sepideh Pourazarm, Amir-massoud Farahmand, and Daniel Nikovski

Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, 02139, USA
sepid@bu.edu, farahmand@merl.com, nikovski@merl.com

ABSTRACT

We introduce Random Projection Filter Bank (RPFB) as a general framework for feature extraction from time series data. RPFB is a set of randomly generated stable autoregressive filters that are convolved with the input time series. Filters in RPFB extract different aspects of the time series, and together they provide a reasonably good summary of the time series. These features can then be used by any conventional machine learning algorithm for solving tasks such as time series prediction, and fault detection and prognosis with time series data. RPFB is easy to implement, fast to compute, and parallelizable. Through a series of experiments we show that RPFB alongside conventional machine learning algorithms can be effective in solving data-driven fault detection and prognosis problems.

1. INTRODUCTION

The modular approach to data-driven fault detection and prognosis of time series data has two components: A feature extraction module that summarizes the time series into a feature vector, and a machine learning module that uses the extracted features to solve the decision problem, e.g., classify the fault. Ideally, the first module should compactly summarize all important aspects of the time series while having a small computational footprint. Several methods have been proposed for feature extraction from time series data. Traditionally, they can be categorized as (i) time domain methods, e.g., fixed-window history-based estimator and autoregressive modeling [Kakade et al., 2016; Ge et al., 2007], and (ii) frequency domain methods, e.g., short-time Fourier transform and envelope analysis [Nayak & Panigrahi, 2011]. Recently, numerous deep learning-based approaches have also been proposed for feature extraction of time series for various prognosis and health management problems, e.g., Deutsch & He [2016].

This paper proposes Random Projection Filter Bank (RPFB) as a generic module for extracting features from time series data. RPFB is a set of randomly generated stable autoregressive (AR) filters whose convolution with the input time series provide features to the second module, i.e., the machine learning module. The use of AR filters, which have an infinite impulse response, allows RPFB to capture information from the distant past of the input time series. This is in contrast with more conventional approaches such as considering only a fixed window of the past time steps, which may not capture all relevant information from the time series. Each filter in an RPFB extracts different aspects of the time series. Together, they provide a reasonably good summary of the time series. RPFB is easy to implement, is fast to compute, and can be parallelized.

RPFB is a general framework for feature extraction from time series data. As opposed to domain-specific feature extraction methods that have been developed for particular prognosis applications (e.g., for capacity estimation of Lithium-Ion batteries by Zhang & Guo [2015] or bearing prognosis by Kim et al. [2016]), RPFB can work on a range of applications and decision problems. We empirically show this by evaluating it on several synthetic and real-world datasets. We have also theoretically studied RPFB elsewhere [Farahmand et al., 2017]. We emphasize that the main goal of this paper is to introduce RPFB as a simple, yet powerful, method for feature extraction that can easily be used with any conventional machine learning approach such as (kernelized) ridge regression, Support Vector Machines, Random Forest, etc. [Hastie et al., 2001; Bishop, 2006] to solve various prediction, diagnosis, and prognosis problems with the time series data.

Section 2 defines the fault detection and prognosis with time series data and discusses the difficulty of using a fixed-window history-based approach. Section 3 proposes RPFB for feature extraction from time series. Section 4 compares RPFB with a fixed window-size approach for several synthetic and standard datasets in the Prognosis and Health Mon-

Part of this work was done when Sepideh Pourazarm was a postdoctoral researcher at MERL.

itoring (PHM) community. The synthetic problem is the next-step prediction task for an AutoRegressive Fractionally Integrated Moving Average (ARFIMA) process, which has a long memory. This is an example of a time series prediction problem. For the fault detection, we use the bearing vibration dataset provided by Machinery Failure Prevention Technology (MFPT) Society.[1] For the fault prognosis problem, we solve the Remaining Useful Life (RUL) problem using the Turbofan Engine Degradation Simulation dataset. Finally, the Li-Ion battery charge-discharge dataset is used as an example of the battery capacity predicting problem. These datasets are both provided by Prognostics Data Repository of NASA.[2]

2. FAULT DETECTION AND PROGNOSIS WITH TIME SERIES DATA

Consider a sequence $(X_1, Y_1), \ldots, (X_t, Y_t)$ of dependent random variables with $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. Depending on how we define X and Y, we can describe different learning problems.

To start, suppose that $Y_t = f^*(X_t) + \varepsilon_t$, in which f^* is an unknown function of the current value of X_t and ε_t is independent of the history $X_{1:t} = (X_1, \ldots, X_t)$ and has a zero expectation, i.e., $\mathbb{E}[\varepsilon_t] = 0$. Finding an estimate $\hat{f}$ of f^* using data is the regression (or classification) problem depending on whether $\mathcal{Y} \subset \mathbb{R}$ (regression) or $\mathcal{Y} = \{0, 1, \ldots, c-1\}$ (classification). The difference of this scenario with more conventional scenarios in the supervised learning and statistics is that here the input data does not satisfy the usual independence assumption anymore.

Problems in diagnosis or prognosis are closely related to the above-mentioned general setup. Suppose that X_t is the sensor observations from a device at time t, e.g., a vibration sensor in an elevator or voltage of a battery. Moreover, assume that there exists a set $D \subset \mathcal{X}$ such that whenever $x \in D$, the device is in its faulty mode. This can be formulated by choosing

$$f^*(x) = \mathbb{I}\{x \in D\} = \begin{cases} +1 & x \in D \\ 0 & x \notin D \end{cases} \tag{1}$$

Fault Detection

For the fault detection problem, the goal is to find a function $\hat{f}$ that approximates $f^*(x) = \mathbb{I}\{x \in D\}$. Since we do not know the set D a priori, we would like to estimate it using dataset in the form of

$$\mathcal{D}_m = \big\{(X_{i,1}, Y_{i,1} = 0), (X_{i,2}, Y_{i,2} = 0), \ldots, (X_{i,T_i-1}, Y_{i,T_i-1} = 0), (X_{i,T_i}, Y_{i,T_i} = +1)\big\}_{i=1}^{m}, \tag{2}$$

which is a dataset consisting of m independent sequences all ended up being in the faulty mode. This dataset is provided by an expert who has decided when the device entered in its faulty state. In this dataset, T_i is the stopping time, which is a random variable that indicates the first time the device becomes faulty. Here it is assumed that the recording of data stops as soon as a faulty state is reached, but we are not limited to this representation and a similar framework can be designed for more general cases too.

This problem can be seen as a classification problem. An estimator might be found by casting the problem as the regularized empirical risk minimization problem with the $0/1$ loss:

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{m} \sum_{t=1}^{T_i} \mathbb{I}\{f(X_{i,t}) \neq Y_{i,t}\} + \lambda J(f), \tag{3}$$

in which $\mathcal{F}$ is a function (hypothesis) space, $J(f)$ is the regularizer (or penalizer), and $\lambda > 0$ is the regularization coefficient. A flexible choice for $\mathcal{F}$ is the family of reproducing kernel Hilbert spaces (RKHS) [Shawe-Taylor & Cristianini, 2004; Steinwart & Christmann, 2008]. If we choose to work with an RKHS, a suitable regularizer is the squared norm of the RKHS, i.e., $J(f) = \|f\|_{\mathcal{F}}^2$. Other types of function spaces and estimators can also be used. For example, one may use a deep neural network to represent $\mathcal{F}$ [Goodfellow et al., 2016].

Notice that the $0/1$-loss function is non-convex, so one in practice uses convex surrogates such as the the hinge loss:

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{m} \sum_{t=1}^{T_i} (1 - f(X_{i,t}) Y_{i,t})_+ + \lambda J(f). \tag{4}$$

Prognosis

A common problem in prognosis is to estimate the RUL of a device. For estimating the RUL, we may use a dataset in the following form:

$$\mathcal{D}_m = \big\{(X_{i,1}, Y_{i,1} = T_i - 1), (X_{i,2}, Y_{i,2} = T_i - 2), \ldots, (X_{i,T_i-1}, Y_{i,T_i-1} = 1), (X_{i,T_i}, Y_{i,T_i} = 0)\big\}_{i=1}^{m}. \tag{5}$$

Finding an estimator that returns the RUL based on the current reading can be seen as a regression problem:

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{m} \sum_{t=1}^{T_i} |f(X_{i,t}) - Y_{i,t}|^2 + \lambda J(f) \tag{6}$$

The use of symmetric loss function such as the least squares loss implicitly assumes that the cost of over-estimating and under-estimation of the RUL is the same. For some applications, this might not be the case. Therefore, we can define the

[1] Available from `www.mfpt.org/faultdata/faultdata.htm`.
[2] Available from `ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository`.

loss of predicting y by $y' = f(x)$ as follows:

$$l(y', y) = \begin{cases} c_1(y' - y)^2 & y' \geq y \\ c_2(y' - y)^2 & y' < y \end{cases} \tag{7}$$

with $c_1, c_2 > 0$ as two constants, c_1 in general being different from c_2, that are determined by the relative cost of over- or under-estimation of the RUL. More generally, we may use any loss function $l : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, e.g., $l(y_1, y_2) = |y_1 - y_2|^2$, and have the estimator as the solution of the following (regularized) risk minimization problem:

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{m} \sum_{t=1}^{T_i} l(f(X_{i,t}), Y_{i,t}) + \lambda J(f). \tag{8}$$

2.1. History-Dependent Learning Problems

More generally, Y_t is a function of the history $X_{1:t} = (X_1, \ldots, X_t)$, possibly contaminated by an independent noise:

$$Y_t = f^*(X_{1:t}) + \varepsilon_t, \tag{9}$$

where ε_t is independent of $X_{1:t}$ and has a zero expectation, i.e., $\mathbb{E}[\varepsilon_t] = 0$. As in the previous case, f^* is an unknown function and we would like to estimate it. The special case of $f^*(X_{1:t}) = f^*(X_t)$ is the same as the previous setting.

To learn an estimator by directly using the history $X_{1:t}$ is challenging as it is a time-varying vector with an ever increasing dimension. There are several approaches to deal with this issue. A standard approach is to use a fixed-window history-based estimator, which shall be explained next (cf. Kakade et al. [2016] for some recent theoretical results). The RPFB is an alternative approach that we describe in Section 3.

In the fixed-window history-based approach, we only look at a fixed window of the immediate past values of $X_{1:t}$. That is, we use samples in the form of $Z_t \triangleq X_{t-H+1:t}$ with a finite integer H that determines the length of windows within the whole history. The regularized least-squares regression estimator would then be

$$\hat{f} \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{m} \sum_{t=1}^{T_i} |f(X_{i,t-H+1:t})) - Y_{i,t}|^2 + \lambda J(f), \tag{10}$$

which should be compared to (6). The formulation for classification is similar with the difference in the choice of loss function, e.g., $(1 - f(X_{i,t-H+1:t})Y_{i,t})_+$ for the hinge loss..

One problem with this approach is that for many stochastic processes, a fixed-sized window of length H is not enough to capture all information about the process. As an example, consider a simple MA(1) univariate random process (e.g., $\mathcal{X} = \mathbb{R}$):

$$X_t = U(t) + bU(t-1) = (1 + bz^{-1})U_t, \tag{11}$$

in which z^{-1} is the time-delay operator (cf. Z-transform, Oppenheim et al. 1999), i.e., $z^{-1}X_t = X_{t-1}$. Suppose that U_t $(t = 1, 2, \ldots)$ is an unobservable random process that drives X_t. For example, it might be an independent and identically distributed (i.i.d.) Gaussian noise, which we do not observe (so it is our latent variable).

If we want to predict $Y_t = X_{t+1}$ given the previous observations $X_{1:t}$, we write

$$U_t = \frac{X_t}{1 + bz^{-1}} = \sum_{k \geq 0} (-b)^k X_{t-k}, \tag{12}$$

so

$$X_{t+1} = U_{t+1} + bU_t = U_{t+1} + \frac{b}{1 + bz^{-1}} X_t. \tag{13}$$

This means that X_t is an autoregressive process AR(∞). The prediction of X_{t+1} requires the value of U_{t+1}, which is unavailable at time t, and all the past values $X_{1:t}$. Since U_{t+1} is unavailable, we cannot use it in our estimate, so this is the intrinsic difficulty of prediction. On the other hand, the values of $X_{1:t}$ are available to us and we can use them to predict X_{t+1}. But if we use a fixed window of the past values (i.e., only use $X_{t-H+1:t}$ for a finite $H \geq 1$), we would miss some information that could potentially be used. This simple example shows that even for a simple MA(1) process with unobserved latent variables, a fixed-horizon window is not a complete summary of the stochastic process.

More generally, suppose that we have a univariate linear ARMA process:

$$A(z^{-1})X_t = B(z^{-1})U_t, \tag{14}$$

with A and B both being polynomials in z^{-1}.[3] The random process U_t is not available to us, and we want to design a predictor (filter) for X_{t+1} based on the observed values $X_{1:t}$. Suppose that A and B are of degree more than 1, so we can write $A(z^{-1}) = 1 + z^{-1}A'(z^{-1})$ and $B(z^{-1}) = 1 + z^{-1}B'(z^{-1})$.[4]

Assuming that A and B are both invertible, we use (14) to get

$$U_t = B^{-1}(z^{-1})A(z^{-1})X_t. \tag{15}$$

Also we can write (14) as

$$\begin{aligned}(1 + z^{-1}A'(z^{-1}))X_{t+1} &= (1 + z^{-1}B'(z^{-1}))U_{t+1} \\ &= U_{t+1} + B'(z^{-1})U_t. \end{aligned} \tag{16}$$

[3] We assume that A and B both have roots within the unit circle, i.e., they are stable.

[4] The fact that both of these polynomials have a leading term of 1 does not matter in this argument.

By (16) and (15), we have

$$X_{t+1} = U_{t+1} + \left[\frac{B'(z^{-1})A(z^{-1})}{B(z^{-1})} - A'(z^{-1})\right] X_t$$
$$= U_{t+1} + \frac{B'(z^{-1}) - A'(z^{-1})}{B(z^{-1})} X_t. \quad (17)$$

When the unknown noise process U_t has a zero mean (i.e., $\mathbb{E}[U_t] = 0$), the estimator

$$\hat{X}_{t+1}(X_{1:t}) = \frac{B'(z^{-1}) - A'(z^{-1})}{B(z^{-1})} X_t, \quad (18)$$

is unbiased, i.e.,

$$\hat{X}_{t+1}(X_{1:t}) = \mathbb{E}[X_{t+1}|X_{1:t}]. \quad (19)$$

In words, the estimate $\hat{X}_{t+1}(X_{1:t})$, which is a function of all previous observations $X_{1:t}$, is an unbiased estimate for X_{t+1}. If we know the distribution of U_{t+1} (e.g., it is a zero-mean Gaussian), we can infer the distribution of X_{t+1} using $\hat{X}_{t+1}(X_{1:t})$ as its mean.

If we knew the model of the dynamical system (A and B), we could design the filter (18) to provide an unbiased prediction for the future values of X_{t+1}. If the learning problem is such that it requires us to know an estimate of the future observations of the dynamical system, this scheme would allow us to design such an estimator.

The challenge here is that we often do not know A and B (or similar for other types of dynamical systems). Estimating A and B for a general dynamical system is a difficult task. The use of maximum likelihood-based approaches is prone to local minimum since U is not known, and one has to use EM-like algorithms, cf. White et al. [2015] and references therein. Here we suggest a simple alternative based on the idea of projecting the signal onto the span of randomly generated dynamical systems. This would be RPFB, which we describe next.

3. RANDOM PROJECTION FILTER BANK

The idea of RPFB can be understood if we write

$$\frac{B'(z^{-1}) - A'(z^{-1})}{B(z^{-1})} = \frac{p(z^{-1})}{q(z^{-1})} \quad (20)$$

for two polynomials p and q, both in z^{-1}. Suppose that $\deg(q) = \deg(B) = m$ and $\deg(A) = \alpha$. So $\deg(p) = \max\{\alpha - 1, m - 1\} = n$. Assume that q has roots $z_1, \ldots, z_m \in \mathcal{C}$ without any multiplicity. This means that $q(z^{-1})$ can be written as

$$q(z^{-1}) = \prod_{i=1}^{m} (z^{-1} - z_i). \quad (21)$$

We have two cases of either $n < m$ or $n \geq m$. We focus on the first case and describe the RPFB, and the intuition behind it. Afterwards we will discuss the second case.

Case 1: Suppose that $n < m$, which implies that $\alpha - 1 < m$. We may write

$$\frac{p(z^{-1})}{q(z^{-1})} = \sum_{i=1}^{m} \frac{b_i}{1 - z_i z^{-1}}, \quad (22)$$

for some choice of b_is. This means that we can write (17) as

$$X_{t+1} = U_{t+1} + \frac{B'(z^{-1}) - A'(z^{-1})}{B(z^{-1})} X_t$$
$$= U_{t+1} + \sum_{i=1}^{m} \frac{b_i}{1 - z_i z^{-1}} X_t. \quad (23)$$

That is, if we knew the set of complex poles $Z_p = \{z_1, \ldots, z_m\}$ and their corresponding coefficients $B_p = \{b_1, \ldots, b_m\}$, we could provide an unbiased estimate of X_{t+1} based on $X_{1:t}$. From now on, we assume that the underlying unknown system is a stable one, i.e., $|z_i| \leq 1$.

The idea of random projection filter bank is based on randomly generating many simple dynamical systems, i.e., filters. We then approximate the true dynamical system as a linear combination of these randomly generated filters. If the number of filters is large enough, the approximation would be accurate.

To be more precise, we cover the set of $\{z \in \mathcal{C} : |z| \leq 1\}$ with $\mathcal{N}(\varepsilon)$ random points $N_\varepsilon = \{Z'_1, \ldots, Z'_{\mathcal{N}(\varepsilon)}\}$ such that for any $z_i \in Z_p$, there exists a $Z' \in N_\varepsilon$ with $|z_i - Z'(z_i)| < \varepsilon$. Roughly speaking, we require $\mathcal{N}(\varepsilon) = O(\varepsilon^{-2})$ random points to cover the unit circle.

We then define the RPFB as the following set of filters denoted by $\phi(z^{-1})$:

$$\phi(z^{-1}) : z^{-1} \mapsto \left(\frac{1}{1 - Z'_1 z^{-1}}, \ldots, \frac{1}{1 - Z'_{\mathcal{N}(\varepsilon)} z^{-1}}\right). \quad (24)$$

With a slight abuse of notation, we use $\phi(X)$ to refer to the (multivariate) time series generated after passing a signal $X = (X_1, \ldots, X_t)$ through the set of filters $\phi(z^{-1})$. We use $[\phi(X)]_t$ to refer to the t-th component of the signal.

The intuition of why this is a good construction is that whenever $|z_1 - z_2|$ is small, the behaviour of the filter

$$\frac{1}{1 - z_1 z^{-1}} \quad (25)$$

is similar to[5]

$$\frac{1}{1-z_2 z^{-1}}. \tag{26}$$

So whenever N_ε provides a good coverage of the unit circle, there exists a sequence (b'_j) such that the dynamical system

$$\frac{p'(z^{-1})}{q'(z^{-1})} = \phi(z^{-1})b' = \sum_{j=1}^{\mathcal{N}(\varepsilon)} \frac{b'_j}{1-Z'_j z^{-1}} \tag{27}$$

behaves similar to unknown $\frac{p}{q}$ (22).

Note that since this is a linear model, parameters b' can be estimated using ordinary least-squares regression, ridge regression, Lasso, etc. For example, the ridge regression estimator for b' is

$$\hat{b} \leftarrow \underset{b}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{t=1}^{T_i} \left|[\phi(X_i)]_t b - X_{i,t+1}\right|^2 + \lambda \left\|b\right\|_2^2. \tag{28}$$

Under proper conditions, we would have $\hat{b} \to b'$. After obtaining $\hat{b}$, we define

$$\tilde{X}(X_{1:t};\hat{b}) = \sum_{j=1}^{\mathcal{N}(\varepsilon)} \frac{\hat{b}_j}{1-Z'_j z^{-1}} X_{1:t},$$

which is an estimator of $\hat{X}(X_{1:t})$, i.e., $\hat{X}(X_{1:t}) \approx \tilde{X}(X_{1:t};\hat{b})$.

Case 2: Suppose that $n \geq m$, which implies that $\alpha - 1 \geq m$. We may write

$$\frac{p(z^{-1})}{q(z^{-1})} = R(z^{-1}) + \frac{\rho(z^{-1})}{q(z^{-1})}, \tag{29}$$

where ρ and R are obtained by Euclidean division of p by q, i.e., $p(z^{-1}) = R(z^{-1})q(z^{-1}) + \rho(z^{-1})$ and $\deg(R) \leq \alpha - 1 - m$ and $\deg(\rho) < m$. We can write

$$\frac{p(z^{-1})}{q(z^{-1})} = \sum_{j=0}^{\alpha-1-m} \nu_j z^{-j} + \sum_{i=1}^{m} \frac{b_i}{1-z_i z^{-1}}. \tag{30}$$

This is similar to Case 1, except that we have additional lag terms. As before, if we knew the set of complex poles and their corresponding coefficients, as well as the coefficients of the residual lag terms ν_j, we could provide an unbiased estimate of X_{t+1} based on $X_{1:t}$. But since we do not know them, we randomly generate random filters. In this case, the feature set would be

$$\phi(z^{-1}) : z^{-1} \mapsto \left(\left[1, z^{-1}, .., z^{-(\alpha-1-m)}\right], \frac{1}{1-Z'_1 z^{-1}}, .., \frac{1}{1-Z'_{\mathcal{N}(\varepsilon)} z^{-1}}\right), \tag{31}$$

which consists of a history window of length $\alpha-1-m$ and the random projection filters, cf. (24). In this case the regressor should estimate both b_is and ν_is in (30).

RPFB is not limited to time series prediction with linear combination of filtered signals. One may use the generated features as the input to any other estimator too. RPFB can be used for other problems such as classification with time series as well.

Algorithm 1 shows how RPFB can be used alongside a regularized empirical risk minimization algorithm. The inputs to the algorithm are the time series data, the loss function to be optimized, the function space $\mathcal{F}$ (e.g., linear, RKHS, etc.), and the number of filters N in the RPFB. The dataset $\mathcal{D}_m = \{(X_{i,1}, Y_{i,1}), \ldots, (X_{i,T_i}, Y_{i,T_i})\}_{i=1}^m$ consists of m time series, each of them possibly have a different length T_i. The labels $Y_{i,t}$ depends on the decision problem. For example, in the time series prediction, $Y_{i,t} = X_{i,t+1}$.

The first step is to create the RPFB by randomly selecting N stable autoregressive filters. In order to avoid dealing with complex numbers, whenever we choose a complex pole, we also include its complex conjugate too. So the pair defines a second-order AR filter with complex conjugate pairs, and it has a real-valued output. We then pass each time series in the dataset through the filter bank in order to create filtered features X'_i. The dimension of features is the same as the number of filters in RPFB, i.e., N. As illustrated in Figure 1, the filtered features are created by the convolu-

Algorithm 1 Random Projection Filter Bank

// $\mathcal{D}_m = \{(X_{i,1}, Y_{i,1}), \ldots, (X_{i,T_i}, Y_{i,T_i})\}_{i=1}^m$: Input data
// $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$: Loss function
// $\mathcal{F}$: Function space
// N: Number of filters in the random projection filter bank
Draw $Z'_1, \ldots, Z'_n$ uniformly within the unit circle
Define filters $\phi(z^{-1}) = \left(\frac{1}{1-Z'_1 z^{-1}}, \ldots, \frac{1}{1-Z'_N z^{-1}}\right)$
for $i = 1$ to m **do**
  Pass the i-th time series through all the random filters $\phi(z^{-1})$, i.e., $X'_{i,1:T_i} = \phi(z^{-1}) * X_{i,1:T_i}$.
end for
Solve the regularized empirical risk minimization

$$\hat{f} \leftarrow \underset{f\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{t=1}^{T_i} l(f(X'_{i,t}), Y_{i,t}) + \lambda J(f).$$

return $\hat{f}$

[5]This is shown rigorously in a theoretical paper submitted to the Neural Information Processing Systems (NIPS) conference [Farahmand et al., 2017].

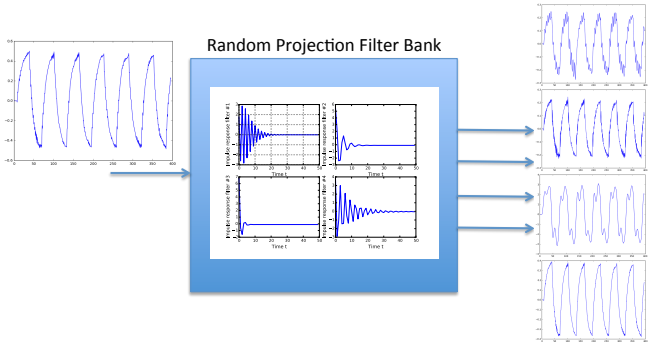

Figure 1. RPFB: Output signals are created by the convolution of the input time series with different filters' impulse response.

tion of input time series and filters' impulse responses, i.e., $X'_{i,1:T_i} = \phi(z^{-1}) * X_{i,1:T_i}$. Here $*$ is the convolution operator and $\phi(z^{-1})$ should be interpreted as the impulse response of the filters.[6]

Finally, taking into account the decision problem (e.g., regression, classification, etc.) and function space, we apply conventional machine learning algorithms to estimate $\hat{f}$. The use of the regularized empirical risk minimizer as the estimator is only one of the possible choices. One may use other estimators too with the features obtained by RPFB, e.g., K-Nearest Neighbourhood, Smoothing Kernel, Decision Trees, etc. [Hastie et al., 2001; Bishop, 2006].

Remark 1. It should be noted that when the number of filters in an RPFB, N, is not large, the unit circle is not covered with a high resolution, i.e., the resolution of the coverage is $\varepsilon = O(\frac{1}{\sqrt{N}})$ with a high probability. For small N, the performance depends on whether the randomly selected points happen to be near the actual poles or not. This causes a high variance in the performance of the RPFB-based estimator. As N increases, the coverage becomes denser, and the performance improves. We will observe these in our empirical studies in this paper. These arguments are precisely stated and proven by Farahmand et al. [2017].

As its name suggests, RPFB is related to the random projection method [Vempala, 2004; Baraniuk et al., 2010]. Random projection can be used to reduce the dimension of a vector space while preserving important properties of the input data. For example, it can be shown that random projection approximately preserves the distances between data points, under certain conditions, e.g., if they all belong to a low-dimensional manifold [Baraniuk & Wakin, 2009]. RPFB is also related to Random Kitchen Sink [Rahimi & Recht, 2009] for approximating potentially infinite-dimensional reproducing kernel Hilbert space (RKHS) with a finite set of randomly selected features. RPFB can be thought of as the dynamical system (or filter) extension of these methods. It is also related to the methods in the Reservoir Computing literature [Lukoševičius & Jaeger, 2009] such as Echo State Network and Liquid State Machine, in which a recurrent neural network with random weights provides a feature vector to a trainable output layer. The difference of RPFB with the methods in reservoir computing is that we are not considering a recurrent neural network as the underlying excitable dynamical system, but only a set of simple AR filters.

[6] In the description of the algorithm we have not determined the initial states of the AR filters in RPFB. In our experiments, we simply select the zero initial state for all filters, but other initializations are possible too.

4. EXPERIMENTS

4.1. Time Series Prediction: ARFIMA Time Series

To study the effectiveness of the RPFB approach, we start with the next-step time series prediction problem of an AutoRegressive Fractionally Integrated Moving Average (ARFIMA) process. ARFIMA is stochastic process with a long memory, i.e., it has a long-range of dependency [Hosking, 1981]. Such time series have been studied and appeared in different fields such as hydrology (river flow) and economy (currency exchange rates). The ARFIMA(p, d, q) process y_t is defined as

$$\Phi(z^{-1})X_t = \Theta(z^{-1})(1 - z^{-1})^{-d}U_t. \tag{32}$$

The operators $\Phi(z^{-1}) = 1 - \sum_{i=1}^{p} \phi_i z^{-i}$ and $\Theta(z^{-1}) = 1 + \sum_{i=1}^{q} \theta_i z^{-i}$ are the autoregressive and moving average operators, respectively. The fractional part of ARFIMA is due to the fractional differencing operator $(1 - z^{-1})^{-d}$ with $-0.5 < d < 0.5$, which has the following binomial expansion:

$$(1 - z^{-1})^{-d} = \sum_{j=0}^{\infty} \frac{\Gamma(j + d)}{\Gamma(j + 1)\Gamma(d)} z^{-j} = \sum_{j=0}^{\infty} \eta_j z^{-j}. \tag{33}$$

The output of the ARFIMA process is driven by a white noise sequence (U_t) with zero mean and variance σ^2.

We aim to find a hypothesis $\hat{f}$ to estimate X_{t+1} given values of the time series in the previous time steps, i.e., $X_{1:t}$. In our experiments, we compare two sets of features: fixed-window history features and features generated by passing the time series through the RPFB. We study the effect of the number of features on the prediction error, for both types of the features.

For the fixed-window history-based features, we apply a sliding window with length H on $X_{1:t}$, that is, we use the feature vector $Z_i = X_{i-H+1:i}$ for $i = H, \ldots, T$. We can obtain different number of features by changing H.

For RPFB, we first create a filter bank by randomly choosing N stable poles of degree 1 (real poles) or degree 2 (complex poles) in the general form of $Z'_j = re^{j\theta}$, cf. (24).[7] More specifically, we randomly choose r from a uniform distribu-

[7] When picking a complex pole, we pick its conjugate too to make the output of the filter real valued.

tion over $[0, 1]$ and θ from a uniform distribution over $[0, 2\pi]$.[8] Afterwards, the time series $X_{1:t}$ is passed through all such filters resulting in the desired feature set. Denoting the impulse response of filter corresponding to the pole Z'_j by $h_{Z'_j}(t)$, the output signal (filtered $X_{1:t}$) is $X'_{j,1:t} = h_{Z'_j}(t) * X_{1:t}$ where $*$ is the convolution operation. The RPFB feature vector at time i is

$$Z_i = \left(X'_{1,i}, \cdots X'_{N,i}\right) \qquad i = 1, \ldots, t. \tag{34}$$

For both feature types, we use ridge regression with the extracted features as its input to estimate the next-step value of the time series, i.e., X_{t+1}.

To qualitatively observe the behavior of these two estimators, we first generate a time series with the length of $T = 10000$ using the following ARFIMA process:

$$X_t = (1 - z^{-1})^{0.4} \frac{(1 + 0.99z^{-1})}{(1 - 0.6z^{-1})} U_t,\ U_t \sim N(0, 1). \tag{35}$$

We train the ridge regression estimator, for both fixed window-based and RPFB features, with the data from this time series. In both cases we use two filters, i.e., $H = 2$ for fixed-window and $N = 2$ for RPFB. Figure 2 shows the truth and predicted signals for both feature types for the same ARFIMA process (the prediction is on a test set that is not used for training).

One can see that the ridge regression applied to the RPFB features, for this particular randomly chosen random poles, performs the next-step prediction task reasonably well even with 2 filters. As discussed earlier, when the number of filters is small, there would be a high variance in performance with respect to the random choice of filters. So it is possible that for another random choice of filters, the performance would be worse or even better.

We now study the effect of the number of filters (N for RPFB and H for window-based) on the performance. Since the performance of RPFB is a random variable because of its dependence on the random choice of filters, we report the average performance over 20 random set of filters. More specifically, we generate 20 independent ARFIMA processes (with the same $T = 10000$ as before). For each time series, we independently generate 40 random filters. We then choose a subset of these random filters with the size of N ($1 \leq N \leq 40$) and train the ridge regression estimator using the extracted features using the filters defined by that subset. The subset of filters are selected in an increasing manner, that is, the set of filters with N_1 is included in the set of filters with N_2 when $N_1 < N_2$. This is to ensure that as we increase the number of features, we also increase the size of the filter space defined by RPFB. For the window-based approach, the randomness is only due to the randomness of the time series.

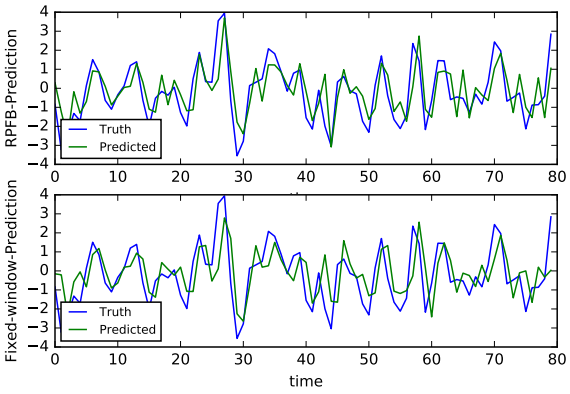

Figure 2. (ARFIMA synthetic time series) - Next-step prediction using ridge regression on RPFB and fixed window (window size=filter No = 2).

Figure 3 shows the average prediction error for 20 different ARFIMA processes as a function of number of features (we use the same axis for N and H). The error bars show one standard error around the empirical average. The performance of both method gradually improves as the number of features increases, and saturates slightly above the prediction error of 1, which is the minimum achievable error (note that in predicting X_{t+1}, we have a noise U_{t+1} that has a normal distribution with a variance of 1. As argued just after (13) in Section 2.1, this is the intrinsic difficulty of prediction).

For the small number of filters in RPFB ($N = 1, 3$), the error bars for the performance of RPFB are large, as expected. As the number of features N are increased, not only the average error of RPFB decreases, but also its variance, which is due to the random choice of filters, decreases too. RPFB outperforms the window-based approach for most values of the number of features. This synthetic example is reassuring as it shows that RPFB is effective in modeling time series data with an analytical model (ARFIMA) without actually trying to directly estimate the parameters of the model.

4.2. Fault Detection: Condition Monitoring for Bearings

Reliable operation of rotating equipments (e.g., turbines) depends on the condition of their bearings. Detecting when a bearing is faulty and requires maintenance is of crucial importance. We consider a bearing vibration dataset provided by Machinery Failure Prevention Technology (MFPT) Society in our experiments.[9]

The dataset consists of three time series including a baseline (good condition bearing/class 0), an outer race fault (class

[8] Note that this does not induce a uniform distribution over the whole unit circle, but induces a distribution that has a higher density closer to zero.

[9] Available from `www.mfpt.org/faultdata/faultdata.htm`.

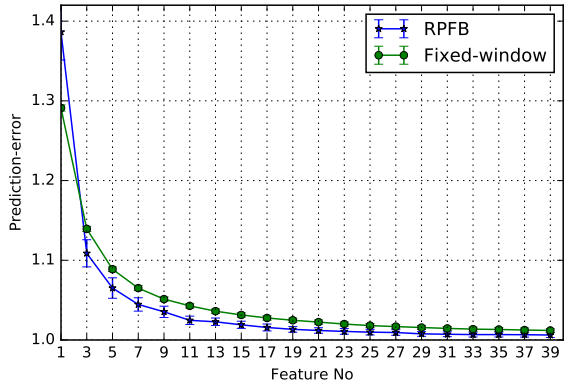

Figure 3. (ARFIMA time series) MSE prediction error using RPFB and fixed-window feature sets for different number of features.

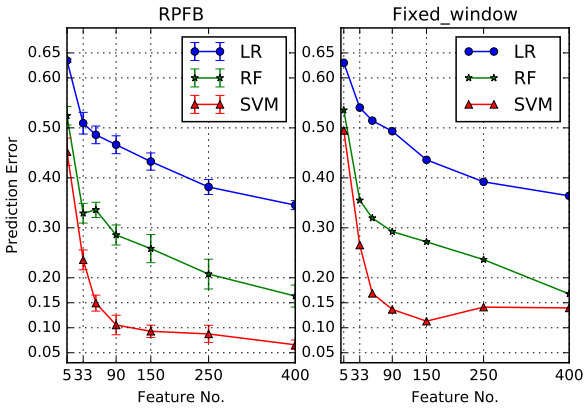

Figure 4. (Bearing Dataset) Classification error on the test dataset using RPFB and fixed-window feature sets.

1), and inner race fault with various loads (class 2). For this problem the goal is to find a function $\hat{f}$ that predicts the class label of a given vibration time series $X_{1:t} = (X_1, ..., X_t)$. We compare fixed-window history-based vs. RPFB feature extraction methods using several standard classification algorithms.

Figure 4 shows the classification error while applying Logistic Regression (LR), Support Vector Machine (SVM) with RBF kernel, and Random Forest (RF) on both feature sets, with varying feature size $N = \{5, 33, 55, 90, 147, 242, 399\}$. For the RPFB, the error bars show one standard error around the average classification error over 10 independent RPFB feature sets. For this dataset, SVM offers the best performance for both feature sets, with the error rate of 0.0658 ± 0.0304 for $N = 399$ for the RPFB and 0.113 for the fixed-window feature sets for $N = 147$. We also observe that SVM with RPFB features performs better than the window-based one for the whole range of features number. It is noticeable that the average classification error of SVM with RPFB features becomes 10.5% for $N = 90$. The window-based approach never reaches this low value of error, even with the much larger number of features $N = 399$.

4.3. Fault Prognosis: Predicting Remaining Useful Life for Turbofan Engine Degradation

Remaining Useful Life (RUL) is defined as the remaining time until a component will no longer operate properly at a particular time of operation. Generally RUL is a random quantity and its accurate prediction is essential for condition-based maintenance and prognosis. Here we consider the simulated dataset for sensor measurements of gas turbine engines provided by the Prognostics Center of Excellence at NASA Ames Research Center [Saxena & Goebel, 2008]. The dataset consists of time series of 24 different sensor measurements for 100 different simulated engines, until the failure criterion was reached. For this problem the goal is to find a function $\hat{f}$ that approximates the RUL of an engine using sensor measurements. For each engine $i = 1, \ldots, 100$, we have the End of Life (EoL) time T_i. Thus, as discussed in Section 2, we create the dataset

$$\mathcal{D}_{100} = \big\{(\mathbf{X_{i,1}}, Y_{i,1} = T_i - 1), (\mathbf{X_{i,2}}, Y_{i,2} = T_i - 2), \ldots, (\mathbf{X_{i,T_i-1}}, Y_{i,T_i-1} = 1), (\mathbf{X_{i,T_i}}, Y_{i,T_i} = 0)\big\}_{i=1}^{100}, \tag{36}$$

where $\mathbf{X_{i,j}}$ is the vector of sensor measurements for engine i at the time step j, i.e., $\mathbf{X_{i,j}} = [X_{i,j}^1, ..., X_{i,j}^{24}]$ in which $X_{i,j}^k$ denotes measurement of sensor k of engine i at the jth time step. The value $Y_{i,j}$ indicates the RUL of engine i at the time step j. This problem can be seen as a regression problem with RUL as the target values.

We investigate the efficiency of RPFB compared to the fixed-window features. For extracting features using RPFB, first a filter bank consisting of randomly chosen (stable) degree-one and degree-two AR filters is created. Then, for engine i, the time series of all sensor measurements are separately passed through the filter bank, i.e. $X'^k_{i,1:T_i} = h_{Z'}(t) * X^k_{i,1:T_i}$, $k = 1, \ldots, 24$ for all filters, identified by Z', in the filter bank. Then the feature vector at the specific time step j for engine i is

$$Z_{i,j} = \left(\mathbf{X'^1_{i,j}}, \cdots \mathbf{X'^N_{i,j}}\right), \tag{37}$$

in which $\mathbf{X'^k_{i,j}}$ includes all 24 filtered signals of engine i at time step j passing through the kth filter in the RPFB with N filters. Thus, the length of the RPFB feature vector for this example equals to $24 \times N$.

For the fixed window history approach, we use a sliding window with length H for all sensors, that is, for engine i the feature vector $Z_{i,j} = (\mathbf{X_{i,j-H+1}}, \cdots \mathbf{X_{i,j}})$ for $j = H, \ldots, T_i$.

Similarly, the length of the fixed-window feature vector is $24 \times H$.

We then apply several standard regression algorithms over both feature sets and investigate their performance. We consider randomly chosen 66 engines as the training dataset and the rest 34 of them as the test dataset. Since the performance using RPFB features depends on the randomly chosen poles for the simple AR filters, we apply multiple independently selected RPFBs and pass the whole dataset through them. We report the average prediction error on the test dataset for different number of features over all such RPFBs. We also report the standard errors.

To measure the prediction error, we compare the Root Mean Square (RMS) of the prediction error considering equal number of features using both fixed-window and RPFB feature sets in the test dataset, i.e.,

$$J_{emp} = \sqrt{\frac{\sum_{i=1}^{34} \sum_{j=H}^{T_i} (Y_{i,j} - \hat{Y}_{i,j})^2}{\sum_{i=1}^{34} \sum_{j=H}^{T_i} 1}}. \qquad (38)$$

Figure 5 shows the prediction error for the RUL of engines in the test dataset using different regression methods (random forest, ridge regression and kernel ridge regression with RBF kernel) applied to both sets of features. For the RPFB features, the average RMS error over 10 randomly chosen RPFBs is depicted. The error bar shows one standard error. It is observed that the performance depends on both the number of features and the type of estimator. The best prediction error is obtained by the kernel-based ridge regression with RPFB features, which achieves the prediction error of 21.45 ± 1.085 for $N = 90$.

It should be noted that when one uses a fixed-window-based feature extraction method, the RUL cannot be predicted from the start of the time series. If the window size is H, one has to wait at least for the first H time steps of the time series before being able to construct the window-based features to be used by the predictor. That is why the error is computed from time step H onwards in (38). This is, however, not a limitation for RPFB as the convolution of a filter is well-defined even from the beginning of a time series. Consequently, even if a fixed-window approach with a large H outperforms the RPFB, it suffers from this limitation, i.e., the larger the length of the window H, the later from the start of the time series one can start predicting.

4.4. Time Series Prediction: Predicting Battery Capacity

The performance of an electrochemical battery cell is heavily affected by ambient environmental condition as well as its discharge profile. There are two major phenomena that strongly influence a battery's behavior as well as its EoL: (i) battery losses some of its capacity with increasing load current, known as *rate capacity effect* [Doyle & Newman, 1997];

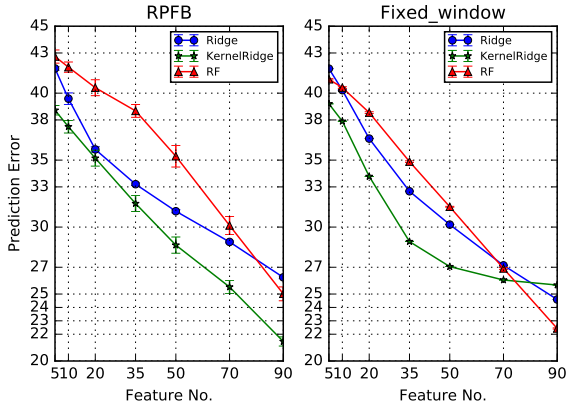

Figure 5. (Turbofan Engine dataset) Prediction error on the test dataset using RPFB feature sets and fixed-window feature sets.

(ii) battery regains portion of its capacity after some rest time, known as *recovery effect* [Martin, 1999]. The main indicator of the battery's State of Health (SOH) and RUL is its actual capacity value, which decreases over the working time of the battery. Due to this non-linearities in battery behavior, predicting the battery capacity is a challenging task. Here we consider a battery dataset provided by the Prognostics Center of Excellence at NASA Ames Research Center [Saha & Goebel, 2007]. The dataset consists of a set of 34 Li-ion batteries that were run through three different operational profiles (charge, discharge and impedance) at different ambient temperatures (4, 24, 44 deg. C). The charging of the batteries was carried out in a constant current mode at 1.5A until the battery voltage reached 4.2V. It then continued in a constant voltage mode until the charge current dropped to 20mA. The discharge of batteries was at different currents and modes (including fixed discharge loads of 1A, 2A and 4A, as well as 0.05Hz square wave loading profile of 4A amplitude and 50% duty cycle) until the battery voltage fell to a predefined value. In some cases the experiments were continued until batteries reached a 20% or 30% fade in their rated capacity, while for other cases it is not the stopping criteria.[10]

Our goal is to predict the battery capacity using the available measurements during charge and discharge profile. The measurements include battery temperature, load current and voltage, charger's current and voltage, battery's terminal voltage and output current during charge and discharge profiles. For each battery we consider the average voltage, current and temperature in charge and discharge profile along with the ambient temperature as the input vector. We use 32 batteries in the dataset and consider randomly selected 21 of them as the training dataset and the rest as the test dataset. For this

[10]These descriptions are quoted from the documents provided by Saha & Goebel [2007].

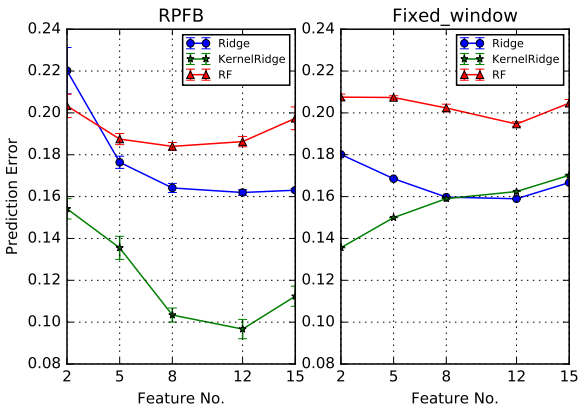

Figure 6. (Battery dataset) Prediction error on the test dataset using RPFB feature sets and fixed-window feature sets.

problem, we create the dataset as follows:

$$\mathcal{D}_{32} = \big\{(\mathbf{X_{i,1}}, Y_{i,1} = C_{i,1}), (\mathbf{X_{i,2}}, Y_{i,2} = C_{i,2}), \ldots, (\mathbf{X_{i,T_i}}, Y_{i,T_i} = C_{i,T_i})\big\}_{i=1}^{32}. \quad (39)$$

Where $\mathbf{X_{i,j}}$ denotes average (charger/load) voltage, current and temperature measurements for battery i during the jth charge-discharge cycle, $Y_{i,j}$ is the capacity of battery in Amp-hrs during the jth discharge cycle, and T_i is the number of cycles that experiments carried out for battery i. For different groups of batteries we have different T_is within the range of 24 to 196 cycles. We use various types of regression estimators to predict the capacity value, using either RPFB or fixed-window as the input features. As a performance metric, we compare the RMS of the predication error on the test dataset. For RPFB, we use 10 independently generated RPFB and report the average performance, as well as the standard error, over various RPFB sizes. The set of filters N are 2, 5, 8, 12, and 15. We use the same sizes for the window size H.

Figure 6 shows the average prediction error for batteries in the test dataset when we use different estimators (random forest, ridge regression and kernel ridge regression with RBF kernel). The error bars show one standard error over different RPFBs. We observe that the Kernel Ridge with RPFB feature results in the least prediction error of 0.0967 ± 0.0146 for $N = 12$.

5. CONCLUSION

This paper introduced Random Projection Filter Bank (RPFB) as a general framework for feature extraction of time series data. Compared to other commonly using methods for feature extraction from time series, such as Short-Time Fourier Transform and deep learning methods, RPFB is computationally cheaper and easier to implement. Through a set of experiments in Section 4, we have shown RPFB's capability for solving various time series prediction, fault detection, and prognosis problems. As a future work, we would like to consider how the physical properties of a particular problem can be used to generate more problem-specific random features.

REFERENCES

Baraniuk, R. G., Cevher, V., & Wakin, M. B. (2010). Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective. *Proceedings of the IEEE*, *98*(6), 959–971.

Baraniuk, R. G., & Wakin, M. B. (2009). Random projections of smooth manifolds. *Foundations of computational mathematics*, *9*(1), 51–77.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Deutsch, J., & He, D. (2016). Using deep learning based approaches for bearing remaining useful life prediction. In *Annual conference of the prognostics and health management society.*

Doyle, M., & Newman, J. (1997). Analysis of capacity-rate data for lithium batteries using simplified models of the discharge process. *Journal of Applied Electrochemistry*, *27*(7), 846–856.

Farahmand, A.-m., Pourazarm, S., & Nikovski, D. N. (2017). Random projection filter bank for time series data. In *Advances in neural information processing systems (NIPS) (under review).*

Ge, D.-F., Hou, B.-P., & Xiang, X.-J. (2007). Study of feature extraction based on autoregressive modeling in ECG automatic diagnosis. *Acta Automatica Sinica*, *33*(5), 462–466.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

Hosking, J. R. M. (1981). Fractional differencing. *Biometrika*, *68*, 165-176.

Kakade, S., Liang, P., Sharan, V., & Valiant, G. (2016). Prediction with a short memory. *arXiv:1612.02526*.

Kim, S., Park, S., Kim, J.-W., Han, J., An, d., Kim, N. H., & Choi, J.-H. (2016). A new prognostics approach for bearing based on entropy decrease and comparison with existing methods. In *Annual conference of the prognostics and health management society.*

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Martin, T. L. (1999). *Balancing batteries, power, and performance: System issues in CPU speed-setting for mobile*

computing (Unpublished doctoral dissertation). Carnegie Mellon University.

Nayak, M., & Panigrahi, B. S. (2011). Advanced signal processing techniques for feature extraction in data mining. *International Journal of Computer Applications*, *19*(9), 30–37.

Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1999). *Discrete-time signal processing* (Second ed.). Prentice Hall.

Rahimi, A., & Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems (NIPS - 21)* (pp. 1313–1320).

Saha, B., & Goebel, K. (2007). *Battery data set.* NASA Ames Prognostics Data Repository. Retrieved from `https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/`

Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set.* NASA Ames Prognostics Data Repository. Retrieved from `https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/`

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge, UK: Cambridge University Press.

Steinwart, I., & Christmann, A. (2008). *Support vector machines*. Springer.

Vempala, S. S. (2004). *The random projection method*. American Mathematical Society.

White, M., Wen, J., Bowling, M., & Schuurmans, D. (2015). Optimal estimation of multivariate ARMA models. In *Proceedings of the 29th AAAI conference on artificial intelligence (AAAI).*

Zhang, Y., & Guo, B. (2015). Online capacity estimation of lithium-ion batteries based on novel feature extraction and adaptive multi-kernel relevance vector machine. *Energies*, *8*, 12439–12457.