Online Denoising Solutions for Forecasting Applications

Pejman Khadivi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Naren Ramakrishnan, Chair
Madhav Marathe
Ravi Tandon
Danfeng Yao
B. Aditya Prakash

August 11, 2016
Blacksburg, Virginia

Keywords: Data analytics, Denoising, Information theory, Time series, Forecasting.

Online Denoising Solutions for Forecasting Applications

Pejman Khadivi

(ABSTRACT)

Dealing with noisy time series is a crucial task in many data-driven real-time applications. Due to the inaccuracies in data acquisition, time series suffer from noise and instability which leads to inaccurate forecasting results. Therefore, in order to improve the performance of time series forecasting, an important pre-processing step is the denoising of data before performing any action. In this research, we will propose various approaches to tackle the noisy time series in forecasting applications. For this purpose, we use different machine learning methods and information theoretical approaches to develop online denoising algorithms. In this dissertation, we propose four categories of time series denoising methods that can be used in different situations, depending on the noise and time series properties. In the first category, a seasonal regression technique is proposed for the denoising of time series with seasonal behavior. In the second category, multiple discrete universal denoisers are developed that can be used for the online denoising of discrete value time series. In the third category, we develop a noisy channel reversal model based on the similarities between time series forecasting and data communication and use that model to deploy an out-of-band noise filtering in forecasting applications. The last category of the proposed methods is deep-learning based denoisers. We use information theoretic concepts to analyze a general feed-forward deep neural network and to prove theoretical bounds for deep neural networks behavior. Furthermore, we propose a denoising deep neural network method for the online denoising of time series. Real-world and synthetic time series are used for numerical experiments and performance evaluations. Experimental results show that the proposed methods can efficiently denoise the time series and improve their quality.

Dedication

To my lovely wife, Marjan.

Acknowledgments

I would like to thank my wonderful advisor, Prof. Naren Ramakrishnan for his patience, guidance, and support during this research. He has been supportive since the first days that I joined Virginia Tech community. Indeed this research would not have been possible without his efforts.

I would like to thank my thesis committee members, Prof. Madhav Marathe, Dr. Ravi Tandon, Dr. Danfeng Yao, and Dr. B. Aditya Prakash for their advice, comments, and time. Special thanks to Dr. Tandon for all his efforts, comments, and support during this research. I would also like to thank my fellow lab-mates and administrative staffs at Virginia Tech for their helps and suggestions. Special thanks to Sharon Kinder-Potter from Computer Science Department and Juanita Victoria from Discovery Analytics Center for their administrative support.

Finally, I would like to thank my wonderful family. In particular, I thank my parents, Hossein and Rezvan, for their endless support and love over the years. They always encouraged me to pursue higher educations and I was so lucky to have them by my side. Indeed, without their support, I would not be able to make it this far. I would also like to thank my lovely wife, Dr. Marjan Momtazpour, for her love, support, helps, and suggestions during these years. I learned so much from her and she was always by my side through the good and bad times. She encouraged me to get my second PhD and without her love, patience, and encouragement, I would not be able to survive in graduate school.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time series analytics is considered as a critical component in various domains including operations research, forecasting, and cyber-physical systems [5, 14, 48]. With recent advancements in sensor technologies, social networks, and telemetry, a gigantic number of time series are generated everyday and with the help of various machine learning techniques, policy makers and scientists are now able to deploy these time series in important applications such as epidemiological forecasting, tourism industry, anomaly detection, and financial forecasting [14, 37, 49, 58].

In most of the real-world applications, due to the inaccuracies in data acquisition, time series are naturally unstable and noisy. This noise and instability may be resulted due to various reasons such as incorrect measurements, faulty sensors, or imperfect data collection. On the other hand, machine learning algorithms are sensitive to outliers. Therefore, any noise and instability can be considered as a source of estimation error and performing any further analysis using the noisy time series will subsequently result in inaccurate reasoning and forecasting. In addition to noise and instability, time series may also suffer from heterogeneous quality as the level of noise may be changed over time which leads to higher level of estimation error.

In real-time data-driven applications, such as closed-loop control systems or forecasting a critical event, immediate usability of the noisy time series is of critical importance. As an example, when we are dealing with influenza count time series, policy makers and health care providers need to know the forecasting outcomes as soon as possible and hence, one cannot wait for the time series to be stabilized. As another example, a data-driven closed loop control system that is used to control the temperature of a data center is illustrated in Fig. 1.1[1]. In this control system, sensors measure the temperature and an AI engine performs required data analysis before issuing the control commands. In this example, faulty sensors, or error in data communication may result in inaccurate datasets and time series that may

[1] `http://www.vigilent.com/products-and-services/dynamic-control/`

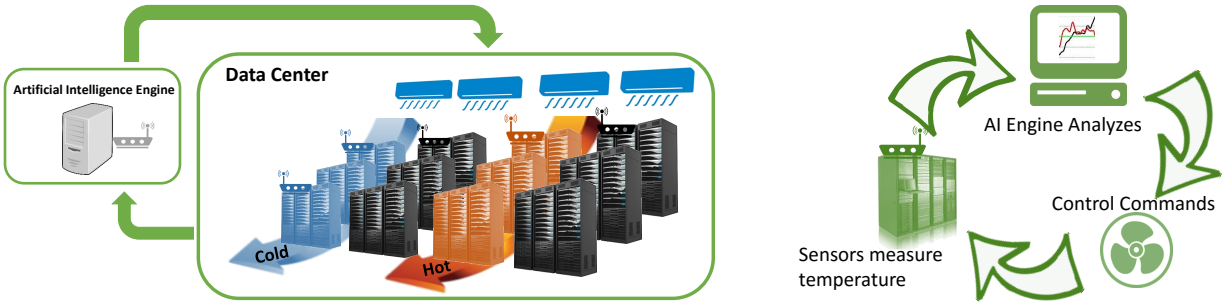


Figure 1.1: An example of a data-driven closed loop temperature control system of a data center.

lead to a major system failure. Accurate time series analysis with noisy time series in real-time applications leads us to perform *denoising* tasks in an online manner.

In this dissertation, the problem of online denoising of time series for forecasting applications is explored. We propose various solutions for this purpose and using theoretical models and data-driven experiments we study the performance of the proposed solutions. In this research, information and communication theoretic methods are deployed to build a theoretical foundation for the problem of online denoising and with the help of signal processing and deep learning techniques we will propose algorithms for the denoising purposes. Real-world and synthetic time series are used for the evaluation of the proposed solutions.

1.1 Goals of the Dissertation

The ultimate goal of this dissertation is to study the problem of time series denoising and propose effective solutions for that. We aim to explore the problem of time series online denoising for forecasting applications using communication and information theoretic methods. In this section, after describing the thesis motivations, we summarize our contributions.

1.1.1 Thesis Motivation

With recent advancements in machine learning and data analytics, time series are deployed widely in various applications such as anomaly detection, disease outbreak forecasting, and tourism industry [14, 37, 49]. These time series are generated by different sources and using various techniques and technologies. While a subset of time series are generated manually by human agents, nowadays, automatically generated time series are becoming more popular which is due to the considerable advancements in the sensor technology.

Despite the techniques which are used in time series generation, a typical real-world time series is susceptible to noise and instability. Human mistakes, unreliable data collection

techniques, incorrect measurements, or faulty sensors are just a few examples that may result in generation of a noisy time series. As we mentioned earlier, deployment of noisy time series in data-driven applications not only results in inaccurate outcomes but also may cause catastrophic conditions such as major system failure. As a matter of fact, machine learning algorithms are sensitive to outliers and noisy data. Hence, deployment of noisy time series will subsequently result in inaccurate reasoning and forecasting.

Time series can suffer from various kinds of distortion. The simplest form of distortion in a time series is known as *missing value*. Figure 1.2(a) shows a base temperature time series from Ozone Level Detection dataset[2], where 7.4% of the values are missing. Various techniques based on Expectation Maximization, Bayesian Networks, and Maximum Likelihood Estimation have been reported in the literature for working with time series that have missing values [33, 56, 44, 57].

Measurement (and estimation) noise is another type of distortion that can be observed in time series. As an example, five time series are shown in Fig. 1.2(b) illustrating the average deviation of earth temperature with respect to a baseline (i.e. average temperature between 1980 and 2000). These temperatures have been measured by different techniques and have been reported in different data sources[3] including NASA's GISS (Goddard Institute for Space Studies), HadCRU (Hadley Centre/Climate Research Unit in the U.K.), NCDC (National Climate Data Center), RSS (Remote Sensing Systems), and UAH (Univ. of Alabama at Huntsville). It is obvious from this figure that while these time series represent the same phenomenon, their values are different which is due to the measurement (or estimation) error. Even if we are not able to reduce the level of noise in a time series, having information about the noise statistics is valuable as we can estimate the reliability of our analysis [34].

Another example is illustrated in Fig. 1.2(c) where the number of people with influenza like illnesses (ILI) are shown for Argentina [14]. This data is collected from Pan American Health Organization (PAHO) website[4]. Reported values in PAHO website are not stable and change over time. Hence, ILI count forecasting based on this dataset is unreliable. In Fig. 1.2(c), ILI counts reported for each week are illustrated based on the reports that have been generated on week 28 and week 41 of 2013. It can be observed from this figure that earlier reports are not reliable and may take some time to be stabilized.

1.1.2 Noise in Communication and Information Theory

Noise is a well-studied concept in information theory and communication systems [18, 53]. It is a widely accepted fact that noise is an inevitable part of an electrical system [63]. Generally speaking, *electrical noise* is used to name any unwanted signal that disturbs the message signals [63, 30]. Noise and distortion can have different effects on the behavior of a

[2] `https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection`

[3] `https://tamino.wordpress.com/2010/12/16/comparing-temperature-data-sets/`

[4] `http://ais.paho.org/phip/viz/ed_flu.asp`

communication system which depends on the noise and signals properties as well as the data transmission model. As an example, three different channel models for a binary transmission system are illustrated in Fig. 1.3. In this figure, for each channel model a transition matrix is provided which shows the transition probabilities from transmitted symbols to received ones. In a binary symmetric channel (BSC), input and output alphabets of the transmission channel are binary while noise has the same effect on all the input values. In Z-Channel, input and output alphabets are binary. However, one of the symbols (e.g. 0) is not affected by the noise. In a binary deletion channel, transmitted 0 and 1 symbols can be deleted due the noise, which is analogous to the missing values in a dataset.

In a communication system, the main job of a receiver is to accurately estimate the transmitted information and the presence of noise limits the capability of receiver in this estimation. Therefore, in the past decades, considerable amount of theoretical methods and engineering techniques have been developed to address the noise in an electrical communication system. As an example, Shannon-Hartley theorem precisely determines the capacity of a transmission channel in presence of additive white Gaussian noise [18, 63]. The theorem states that the capacity, C, of a transmission channel with bandwidth B in presence of an additive white Gaussian noise with power of N is

$$C = Blog_2\left(1 + \frac{S}{N}\right)$$

where S is the (message) signal power. One can easily observe that if noise power is increased, channel capacity, i.e. the maximum amount of information that can be accurately passed through the channel, is decreased. This capacity reduction is due to the fact that higher level of noise results in lower level of accuracy in message estimation at the receiver. Preliminary experiments show that noise can have similar effects on time series analysis. An example is illustrated in Fig. 1.4. In this example, Gaussian noise with different levels of standard deviation is added to synthetic linear and non-linear time series. Then, a simple autoregressive model, AR(1), is used to estimate the future values of each time series. It can be observed that regression error increases with higher level of noise.

Due to the similarities that exist between the effect of noise in time series applications and effect of noise on an electrical communication system, we may be able to use theories and techniques that have been developed for communication systems in time series analysis in order to address the issues of noise and instability in time series.

1.1.3 Related Works

Smoothing and trend filtering methods are typically used in time series analysis to remove useless information and to observe meaningful patterns and trends in the time series more efficiently. Various smoothing and trend filtering methods have been studied in the literature [46, 2, 1, 40, 7]. Moving average smoothing methods are well-known classic techniques that

due to their simplicity are very popular methods for time series smoothing [11]. By definition, trend of a time series is considered as the main additive part of data that demonstrate the global behavior of time series [2]. Kernel Kalman filtering [55], non-linear adaptive filtering [22], and wavelet-based filtering [22, 29] methods have been also proposed in the literature for time series denoising and smoothing.

Trend extraction and smoothing are useful for long term forecasting applications [46, 1]. As an example, when a scientist wants to forecast the earth temperature in the next 10 or 20 years, he is not interested in daily small fluctuations in temperature or seasonal patterns, but he is mainly interested in the temperature trend. Hence, trend extraction methods can be efficiently used here to have a better understanding of the underlying behavior of the time series. However, in daily weather forecasting, minor variations in temperature and seasonal patterns are really important. While for long term forecasting these daily fluctuations are interpreted as noise, in short term forecasting applications they are used as valuable data. As an example, let us consider the time series of Fig. 1.2(b). As we mentioned before, this figure illustrates five time series that each of them shows the deviation in the average temperature of earth with respect to a reference point. Let us assume that the average of all the five time series is used as the base line. Then, at each particular year, the difference between the value of an individual time series (e.g. CRU time series) and the calculated baseline can be interpreted as noise. However, to observe the global behavior of time series, smoothing techniques may be used to extract the time series trend. This is illustrated in Fig. 1.5. Figure 1.5(a) illustrates how the baseline temperature and CRU time series are changing over time. The smoothed versions of these time series are illustrated in Fig. 1.5(b). To determine the smoothed values, simple and weighted moving average smoothing methods are deployed. It is obvious from this figure that the extracted trends show the global warming trend.

Denoising have been also studied in other fields such as image processing. In [73] a universal denoising algorithm for datasets with discrete value alphabets have been proposed. This method has been customized for the denoising of grayscale images in [51]. Various deep learning based methods have been also proposed in the literature for image denoising applications [43, 75, 72]. Surveys on image denoising techniques have been provided in [52] and [12].

1.1.4 Proposed Denoising Methods

In this thesis we deploy information and communication theoretic methods to explore the problem of noise and instability in time series and to propose new methods for time series denoising. In this regard, four categories of solutions are proposed and evaluated.

The first proposed solution is to use regression algorithms for time series denoising. The proposed method deploys the seasonality feature of real-world time series to adjust the noisy samples of a given time series based on the previous noisy and noiseless observations. We

perform extensive experiments with real-world datasets to study the performance of the proposed approach. Results show that the proposed model can improve the forecasting performance in terms of accuracy.

In the second category of the proposed solutions we extend discrete universal denoiser (DUDE) [73] to address the tradeoff between time-sensitivity and denoising accuracy in online denoising applications. DUDE is a statistical denoising method that has been proposed for denoising of datasets with finite discrete alphabet. Based on DUDE, we propose three algorithms for online denoising of time series. Furthermore, we propose a discrete universal denoising algorithm for datasets with very large alphabet. Numerical experiments with synthetic and real-world time series show the efficiency of the proposed algorithms.

The third category is a framework based on the similarities of time series regression and communication systems, to address the aforementioned issues by making explicit assumptions about the noise and behavior of time series. We will show that noise and deviations from the predetermined behavior of time series is similar to the noise of a communication channel and thus, use communication-theoretic methods to address this deviation to some level. We model the noisy communication channel between the future and past values of time series as an additive Gaussian noise channel. Channel parameters are estimated from the observed data together with an out-of-band filtering algorithm to boost the signal-to-noise (SNR) ratio. The estimates of this noisy channel process are subsequently used for forecasting the future future values of time series via regression. We perform extensive experiments with real-world and synthetic datasets to study the performance of the proposed approach. Results show that the proposed model can efficiently improve the forecasting performance in terms of accuracy.

In the forth category we propose a framework to deploy deep learning in the denoising of time series. Furthermore, we use information theory concepts to study the behavior of a feed-forward deep neural network. Real-world time series with different noise and distortion models are used to evaluate the performance of the proposed method. Experimental results show that efficiency of the proposed neural network based denoising method.

It should be noted that there is no unique denoising strategy that can be applied to all noisy time series and depending on the properties of noise and time series we need to adjust our denoising method. Furthermore, our prior knowledge regarding the noise model plays an important role in the selection of appropriate denoising technique.

In a time series forecasting application, noise can be categorized into two major groups: *data noise* and *forecasting noise*. Data noise is the noise that accompanies the time series and may have various sources such as existence of a faulty sensor, wrong estimation, human mistakes, or externally injected noise due to data transmission. Data noise is observable in the time series before using any time series analytics method. Forecasting noise is the noise that is generated in the forecasting results due to the use of an inaccurate forecasting method. In fact, to develop a forecasting method, we typically assume that the time series follows a particular behavior. If the actual behavior of time series deviates from our prior

Table 1.1: Appropriate situation that each proposed denoising method can be deployed.

Denoising Method	Prior Knowledge about Noise	Discrete or Continuous	Forecasting or Data Noise	Time Series Seasonality
Seasonal Regression Denoising	Not Required	Both	Data	Required
Discrete Universal Denoising	Required	Discrete	Data	Not Required
Noisy Channel Reversal Model and Out-of-band Filtering	Required	Both	Both	Not Required
Deep Learning based Denoising	Not Required	Both	Data	Not Required

assumptions, some noise will be injected in the forecasting results. While the nature of data and forecasting noises are different, they have a similar effect: both of them make the forecasting results inaccurate.

Another important question is that how much of information we have about the noise. Noise is considered as a random process. However, through the use of an appropriate training dataset, one may be able to acquire significant knowledge about the noise model. While having prior knowledge about the noise model can dramatically improves the denoising performance, it is not always possible to obtain enough reliable information about noise. This may be due to the lack of training data or to the time varying behavior of noise.

In addition to the various properties of noise, time series may also have different properties. As an example, while various time series, such as tourism demand [37], are highly seasonal, other time series do not demonstrate any seasonal behavior. Seasonality can be considered as an important feature in the denoising process. As an example, when tourism demand time series have a seasonal behavior with a period of 12-month, last year data of the same month can be efficiently used to improve the quality of the currently generated data samples. Another important feature of a time series is that if the time series is constructed by discrete or continuous values. As an example, number of people that have influenza symptoms in a geographical region, tourism demand in a particular destination, or number of retweets of a Twitter message can be considered as discrete value time series. However, time series about the humidity in a specific city, or temperature of a data center are typically considered as continuous value time series.

Depending on the properties of the noise and time series as well as our prior knowledge about the noise model, appropriate denoising technique may be selected. This is illustrated in Table 1.1. As an example, discrete universal denoising techniques can be deployed for discrete valued time series and we need to have prior knowledge about the noise model. However, seasonality of the time series is not required for this category of denoisers.

1.1.5 Thesis Contributions

In this dissertation, we use information and communication theoretical concepts and techniques for the purpose of time series denoising. We develop multiple denoising algorithms that can be used under different conditions. The contributions of this thesis are as follows:

- We explore the locality and seasonality features of time series and their applications in denoising process. Based on the seasonality features, a seasonal regression based denoising approach is developed and studied in different applications.
- We extend the discrete universal denoising algorithms to address the time sensitivity of online forecasting application. Furthermore, we address the large alphabet issue of this class of denoisers and propose a hierarchical quantization based method to solve the large alphabet issue.
- A noisy channel reversal model is proposed based on the similarities between a data transmission system and a time series forecasting model. The proposed noisy channel reversal model can be used to estimate the noise statistics.
- Using the proposed noisy channel reversal model, we propose the use of out-of-band noise filtering to reduce the side effects of data and forecasting noises.
- We propose a deep learning based framework for the denoising of time series. The proposed framework is explored under different noise models.
- From an information theoretical perspective, we analyze the flow of information in a feed-forward deep neural network and prove theoretical bounds both for a typical feed-forward neural network and for the denoising neural networks.

1.2 Organization of Dissertation

The rest of this report is organized as follows:

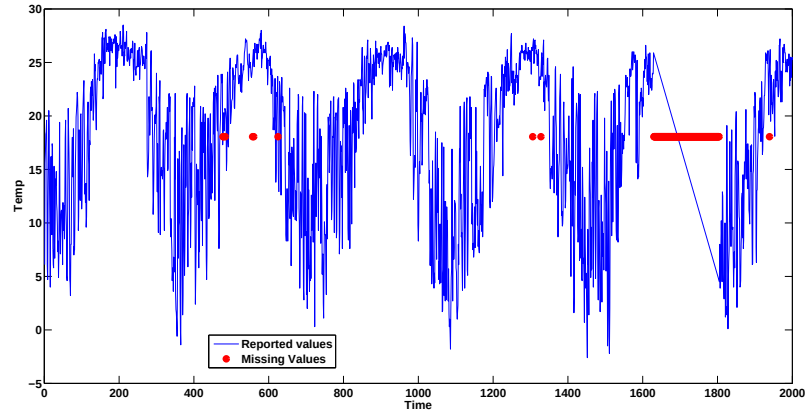
Chapter 2) Regression based Denoising for Seasonal Time Series: In this chapter, we first explore the seasonality and locality features of real-world time series. Then, through multiple case studies, we propose regression based solutions for the denoising of seasonal time series. Furthermore, we explore the problem of time series instability for forecasting applications. Part of this chapter is excerpted from the paper published in the Proceedings of the SIAM International Conference on Data Mining (SDM 2014) [14].

Chapter 3) Online Denoising of Discrete Noisy Data: In this chapter, the problem of online discrete denoising is addressed from the information theoretic perspective. Three online denoising algorithms are proposed which address the tradeoff between delay and accuracy of denoising. We will show that the proposed online algorithms asymptotically converge to a class of optimal offline block denoisers. Furthermore, we propose a hierarchical quantization based algorithm to address the problem of large alphabets in discrete universal denoisers. Part of this chapter has been published in the Proceedings of the IEEE International Symposium on Information Theory (ISIT'15) [38].

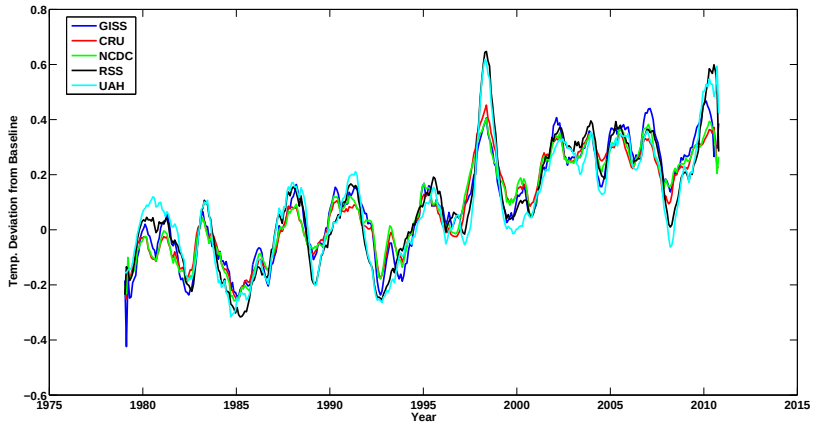
Chapter 4) Time Series Forecasting via Noisy Channel Reversal: In this chapter, we introduce a new communication-theoretic framework for modeling and forecasting time series. We use a data-driven probabilistic approach to estimate the unknown parameters of the system. In this chapter, we also show how out-of-band noise filtering may efficiently used for noise reduction in time series. The model is generalized to include the heterogeneity in the quality of time series. This chapter has been published in the Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2015) [36].

Chapter 5) Deep Learning based Denoising: In this chapter, we propose a deep learning based framework for online denoising of time series. In this framework, using the previously observed noisy and noiseless time series values, we train a feed-forward multi-layer neural network and deploy it to adjust the future noisy observations. If we do not have enough noisy observations, noise injection is used to synthetically generate the required training dataset. Furthermore, we theoretically study the flow of information in a feed-forward deep neural network and prove some theoretical bounds.

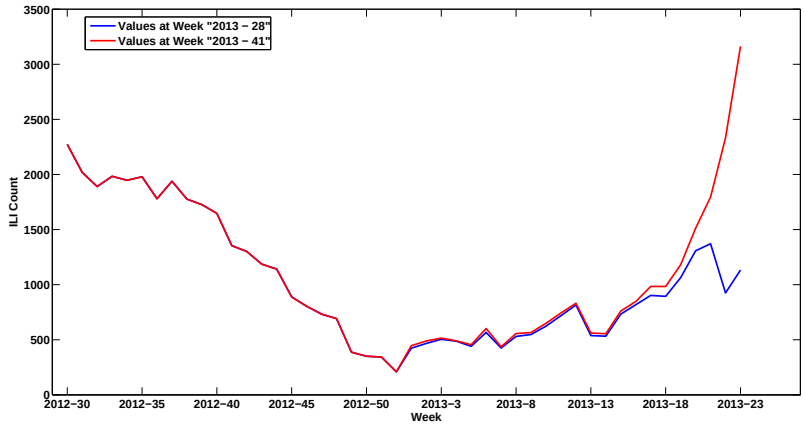
Chapter 6) Conclusion and Future Work: This chapter is dedicated to some concluding remarks and future work.



(a)



(b)



(c)

Figure 1.2: Examples of noisy time series: (a) A base temperature time series from Ozone Level Detection dataset with missing values. (b) Deviation in earth temperature measured by different techniques. (c) Instability in ILI Counts for Argentina from PAHO dataset.

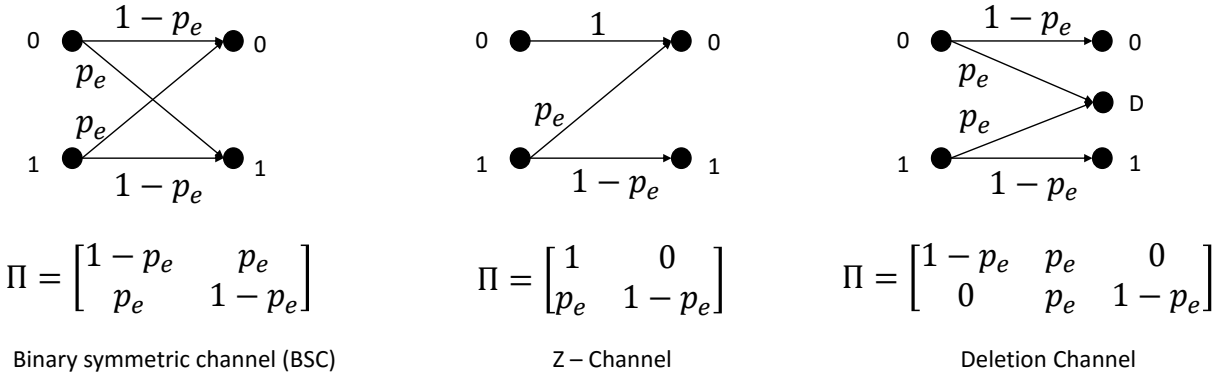


Figure 1.3: Three example channel models for a binary data transmission system.

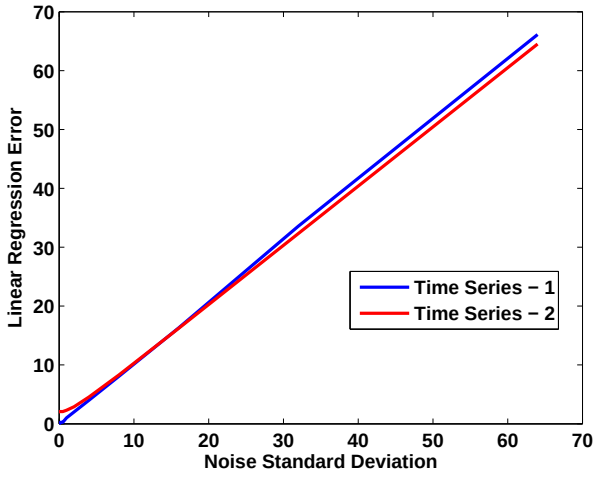
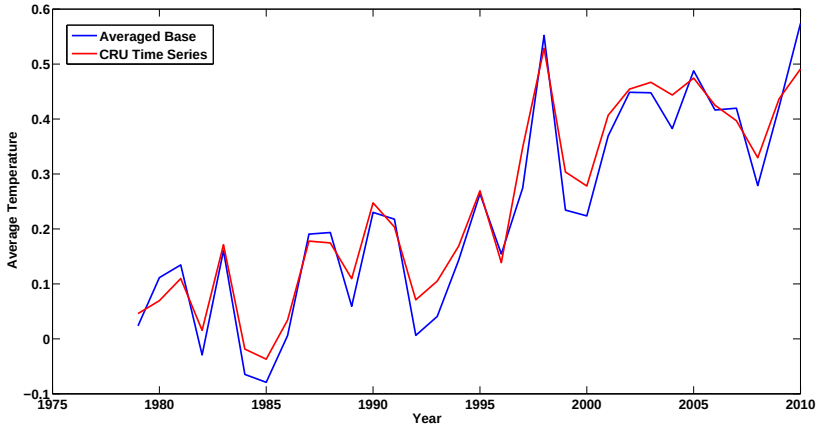
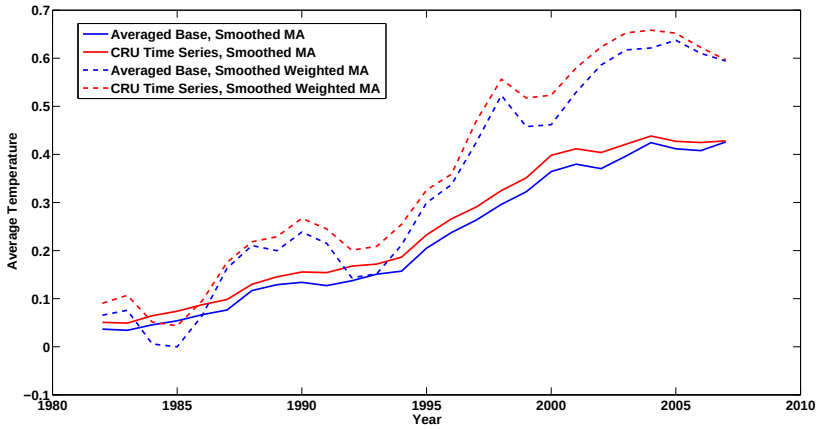


Figure 1.4: Effect of noise on regression accuracy. *Time Series - 1* is a linear time series where its value at time t is $x(t) = 3t + 2$. *Time Series - 2* is a non-linear time series where its value at time t is $x(t) = t^2 - t^{1.5} + t + 1$.



(a)



(b)

Figure 1.5: Trend extraction and noise in time series of Fig. 1.2(b): (a) Difference between a *noisy* time series and the base line (b) Smoothed time series and extracted trends.

Chapter 2

Regression based Denoising for Seasonal Time Series

Real-world time series have natural accompanying properties that can be deployed in denoising process. As an example, almost all the real-world time series have locality properties which means that neighboring values are typically closer together, compared to the farther ones. As another example, tourism and influenza outbreak generally demonstrate seasonal behavior. These accompanying properties may be efficiently deployed in the denoising of time series. In this chapter, we aim to use the seasonal property of time series and propose regression-based denoising methods for this class of time series.

2.1 Local and Seasonal Behavior of Time Series

Locality is a natural property of real-world time series. Observations show that in real-world time series and on average, neighboring samples have closer values than the farther ones. Let us assume that we have observed a time series, $Y = y_1, y_2, \cdots, y_t$. One metric that may be used to measure the similarity of values in a dataset is standard deviation (SD). As a matter of fact, smaller values of SD means that the values are closer to the expected value. To measure the locality property of a particular time series we locate a sliding window with the length of w over the time series and move it over the samples. Then, we measure the SD of samples inside the sliding window and compare it to the SD of the whole time series. We name the SD of data samples inside a sliding window as the local SD which is represented by σ_L. The local SD of data samples from y_i to y_{i+w-1} is represented by $\sigma_L(i)$ and we have:

$$\bar{\sigma}_L = \frac{1}{t-w+1} \sum_{i=1}^{t-w+1} \sigma_L(i) \tag{2.1}$$

where $\bar{\sigma}_L$ is the average of local SDs over all values. In order to compare the local SDs with the SD of the whole time series, i.e. *global SD*, we define the *relative local SD of time series* as follows:

$$R_\sigma = \frac{\bar{\sigma}_L}{\sigma} \tag{2.2}$$

where σ is the global SD and R_σ is the relative local SD and can be used as a measure of locality in the time series. Smaller values of R_σ means that the time series illustrates higher locality.

To provide an illustrative example we performed numerical experiments with 64 real-world time series. These time series are illustrated in Table 2.1 and have been collected from the following sources:

- UCI: Real-world time series from UCI Repository.
- NIST: Time series from the NIST StRD website[1] .
- STOCK: Closing stock price of different companies from Yahoo Finance.
- GST: Weekly Google Search Trends for different topics.
- PAHO: Influenza Like Illnesses statistics of various countries from the Pan American Health Organization website[2].
- WUT: Wikipedia Usage Trends for different topics.

Results of this experiment are illustrated in Fig. 2.1. Figure 2.1(a) depicts the value of R_σ for five selected time series, when w changes from 3 to 45. The average value of R_σ for all the 64 time series when w changes from 3 to 45 is illustrated in Fig. 2.1(b). It can be observed from this figure that no matter which time series we are using, R_σ is smaller for smaller values of w. However, depending on the time series the value of R_σ changes with different rate when we change the value of w. Furthermore, some time series demonstrate more locality (i.e. smaller R_σ's) than the other ones.

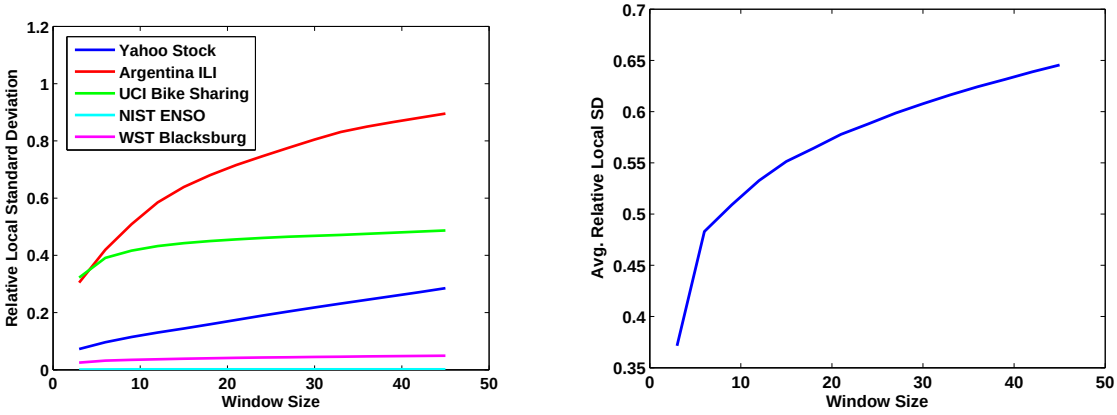
In addition to the locality property of time series, some time series demonstrate seasonal behavior. As an example, tourism demand in a particular area generally results in seasonal time series, demonstrating periodic patterns of high seasons, which are triggered by external causes such as holidays or weather conditions. As another example, for some regions, seasonality property can be observed in the number of people with Influenza symptoms. The reason is that climate and environmental conditions is correlated with Influenza epidemics [69]. Autocorrelation coefficient is a well-known metric that may be used to measure the seasonal behavior of a time series. Tourism demand time series for Hawaii [37] together with their autocorrelation coefficients are depicted in Fig. 2.2. It can be observed from this

[1] `http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`

[2] `http://ais.paho.org/phip/viz/ed_flu.asp`

Table 2.1: Time series used in locality test.

Data Source	Time Series	Data Source	Time Series
STOCK	Yahoo	UCI	Bike Sharing
STOCK	Ford	UCI	CalIt2 Flow
STOCK	Microsoft	UCI	Dodgers Avg. Daily
STOCK	GE	GST	Storm
STOCK	HP	GST	Influenza
STOCK	AEP	GST	Melanoma
STOCK	IBM	GST	Schizophrenia
PAHO	Argentina	GST	Diabetes
PAHO	Bolivia	GST	Hepatitis
PAHO	Chile	GST	Basketball
PAHO	Colombia	GST	Football
PAHO	Costa Rica	GST	Volleyball
PAHO	Ecuador	GST	Wrestling
PAHO	El Salvador	GST	Swimming
PAHO	Guatemala	GST	Yahoo
PAHO	Honduras	GST	Microsoft
PAHO	Mexico	GST	Google
PAHO	Nicaragua	GST	Facebook
PAHO	Panama	GST	Twitter
PAHO	Paraguay	GST	Youtube
PAHO	Peru	GST	Hotmail
WUT	Blacksburg	GST	Craigslist
WUT	Christiansburg	GST	Amazon
WUT	Radford	GST	eBay
WUT	Roanoke	GST	Dollar
WUT	Salem	GST	Euro
WUT	Organic Chemistry	GST	Yen
WUT	Organic Component	GST	Pound
WUT	Organic Farming	GST	Earthquake
WUT	Organic Food	GST	Typhoon
WUT	Organic Matter	GST	Hurricane
NIST	ENSO	GST	Tsunami



(a) (b)

Figure 2.1: Relative local standard deviation of time series. (a) R_σ of five selected time series. (b) Average of R_σ over all the time series of Table 2.1.

figure that every 12 months the autocorrelation coefficient reaches a local maximum which indicates that tourism demand is a seasonal time series and the number of tourists in each month of the year has a high correlation with the number of tourists in the same month of the past (and future) years.

Locality and seasonality properties of time series are important characteristics that can be efficiently deployed in the denoising process of a noisy time series. In the following sections, we first provide an illustrative simple example to highlight the importance of these properties in the process of denoising. Then, we provide real-world case studies to show how seasonality property may be deployed in the denoising process.

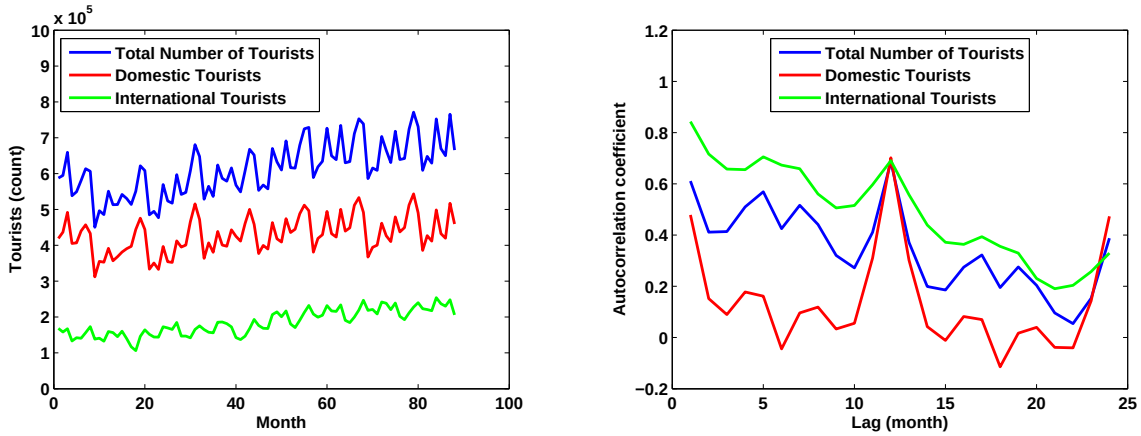
2.2 Sawtooth Wave Time Series

In this section, we present an illustrative example with a simple synthetic time series to indicate the importance of locality and seasonality properties in the denoising process. For this purpose, we use a *sawtooth wave* time series which is depicted in Fig. 2.3. Figure 2.3 depicts a typical sawtooth wave that changes between 0 and 99. The noisy versions of this time series with normal and uniform noise models are illustrated in Fig. 2.3 (b) and (c), respectively.

In Fig. 2.3(b), the normal noise has been applied to the original time series as follows:

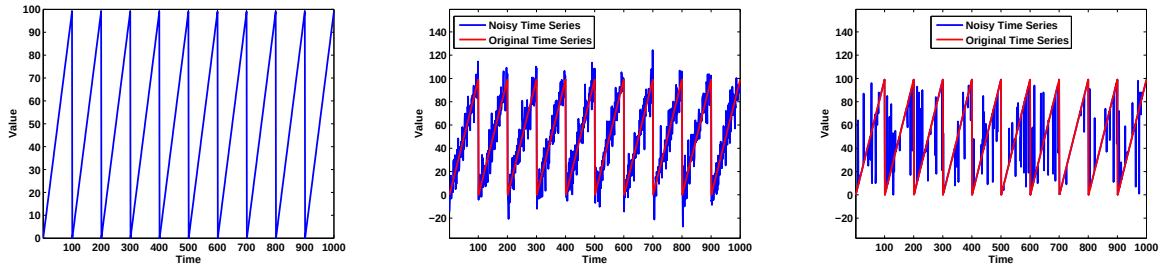
$$x_t^N = x_t + n_t \tag{2.3}$$

where x_t is the noiseless sample at time t, x_t^N is the noisy sample, and n_t is the i.i.d. normal noise, $n_t \sim N(0, \sigma_N^2)$. In this noise model, σ_N^2 is the noise power.



(a) (b)

Figure 2.2: Three time series together with their autocorrelation coefficients: (a) Total, domestic, and international tourism demand for Hawaii, and (b) the autocorrelation coefficients of the time series.



(a) (b) (c)

Figure 2.3: (a) A typical sawtooth wave with its noisy versions when noise is (b) normal and (c) uniform.

In Fig. 2.3(c), the uniform noise has been applied to the noiseless time series based on the following model:

$$x_t^U = \begin{cases} u_t & , \quad with\ probability\ of\ \epsilon \\ x_t & , \quad with\ probability\ of\ 1-\epsilon \end{cases} \tag{2.4}$$

where, ϵ is the error probability and u_t is uniformly selected from the range of the values in the time series.

Let us evaluate the example of Fig. 2.3(a). If at a particular time we observe $0, 1, 10, 3, 4$, as the five consecutive samples of time series, there are three possibilities:

1. The noiseless values are $0, 1, 2, 3, 4$ and 2 has been changed to 10.

2. The noiseless values are $8, 9, 10, 11, 12$ and four errors have been occurred.

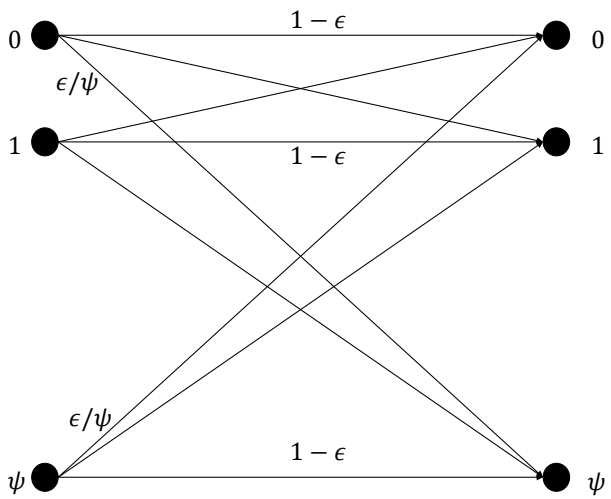


Figure 2.4: Uniform noise model of Eq. 2.4.

3. The noiseless values are $a, a+1, a+2, a+3, a+4$ where $a \notin \{0, 8\}$ and five errors have been occurred.

Note that in the third case, it is not possible to have less than five errors. Considering the noise model of Eq. 2.4 for uniform noise, the error probability at each sample of the time series is ϵ. Hence, probability of each of the above cases are as follows:

1. Probability of one noisy sample and four noiseless ones: $\epsilon(1-\epsilon)^4$.
2. Probability of one noiseless sample and four noisy ones: $\epsilon^4(1-\epsilon)$.
3. Probability of five noisy samples: ϵ^5.

It is easy to show that when $\epsilon < \frac{1}{2}$ the probability of the first case is highest. It can be shown that when the error probability is less than 0.5 nearest neighbor decoding results in the lowest expected loss.

To generalize the above example, let us assume that the sawtooth time series varies between 0 and ψ and the noise model is similar to Eq. 2.4. Moreover, we assume that the time series is constructed by integer values. This noise model is illustrated in Fig. 2.4. Also, we assume that we have observed a sequence of w consecutive samples, $x_i^U, \cdots, x_{i+w-1}^U$. Then, the probability of having k errors in specific locations is

$$\pi_k = \epsilon^k (1-\epsilon)^{w-k}. \tag{2.5}$$

When $\epsilon < \frac{1}{2}$ it is easy to show that the probability of having fewer errors is higher. In other words,

$$k_1 < k_2 \Rightarrow \pi_{k_1} > \pi_{k_2}. \tag{2.6}$$

Algorithm 1 Local Sawtooth Wave Time Series Denoising Algorithm

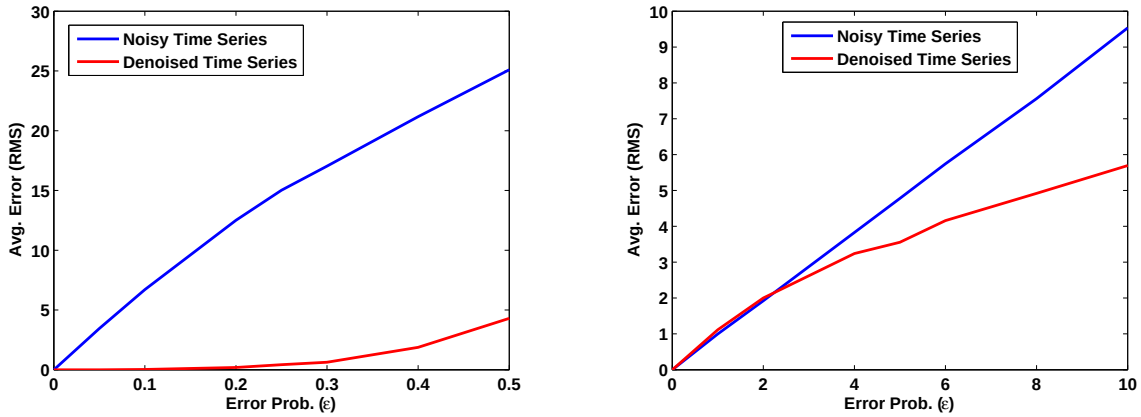
function SAWDENOISER($X^{noisy}_{i:i+w-1}$,w)
  $minimumDistance = \infty$
  $minA = -1$
  for $a = 0$ to ψ **do**
    if $d_H(X^{noisy}_{i:i+w-1}, \Psi_{a:a+w-1}) < minimumDistance$ **then**
      $minimumDistance = d_H(X^{noisy}_{i:i+w-1}, \Psi_{a:a+w-1})$
      $minA = a$
    else
      if $d_H(X^{noisy}_{i:i+w-1}, \Psi_{a:a+w-1}) = minimumDistance$ **then**
        if $d_E(X^{noisy}_{i:i+w-1}, \Psi_{a:a+w-1}) < d_E(X^{noisy}_{i:i+w-1}, \Psi_{minA:minA+w-1})$ **then**
          $minimumDistance = d_H(X^{noisy}_{i:i+w-1}, \Psi_{a:a+w-1})$
          $minA = a$
  return $\Psi_{minA:minA+w-1}$

Let us represent the w consecutive samples of the noisy time series starting at the i^{th} sample by $X^U_{i:i+w-1} = \left[x^U_i, \cdots, x^U_{i+w-1}\right]$. We also represent a noiseless sequence of w consecutive samples from the sawtooth wave time series by $\Psi_{a:a+w-1} = [a, \cdots, (a+w-1) \mod (\psi+1)]$. As an example, when $\psi = 99$ and $w = 3$, $[0,1,2]$,$[25,26,27]$, and $[98,99,0]$ are all valid sequences of the sawtooth wave time series.

The Hamming distance between two *vectors*, each with w samples from an alphabet $\mathcal{A}$, is defined as the number of locations that the two vectors are different. Then, when $\epsilon < \frac{1}{2}$ and based on Eq. 2.6, it can be shown that *nearest neighbor decoding* (NND) results in the most probable noiseless sequence. In other words, a noisy sequence of w samples from a sawtooth wave time series, $X^U_{i:i+w-1}$, should be decoded to $\Psi_{a:a+w-1}$ where the Hamming distance between $X^U_{i:i+w-1}$ and $\Psi_{a:a+w-1}$ is minimized for all possible values of a. The whole denoising algorithm is illustrated in Algorithm 1. In this algorithm, $d_H(\cdot,\cdot)$ measures the Hamming distance between two vector and $d_E(\cdot,\cdot)$ is used to measure the Euclidean distance. In fact, if the Hamming distance between $X^{noisy}_{i:i+w-1}$ and two valid sawtooth wave sequences, $\Psi_{a:a+w-1}$ and $\Psi_{b:b+w-1}$ is equal, the algorithm chooses the sequence that has the minimum Euclidean distance. The time complexity of this algorithm is $O(w\psi)$.

Simulation results for $\psi = 99$ and $w = 5$ are illustrated in Fig. 2.5. Average root mean square (RMS) error of the noisy and denoised time series for uniform noise model are compared in Figure 2.5(a). Figure 2.5(b) depicts the average RMS errors for the normal noise model. Because we assumed that the time series only takes integer values, the normal noise model rounds the real values to the nearest integer value. It can be observed from this figure that the proposed denoising algorithm can efficiently denoise the sawtooth wave time series.

It is worth mentioning that Algorithm 1 works based on the locality features of the sawtooth wave time series. In fact, a sliding window of the size of w is moved over the noisy time



(a) (b)

Figure 2.5: Effect of simple locality based denoiser of a sawtooth wave time series with (a) uniform and (b) normal noise models. In this experiment, $\psi = 99$ and $w = 5$.

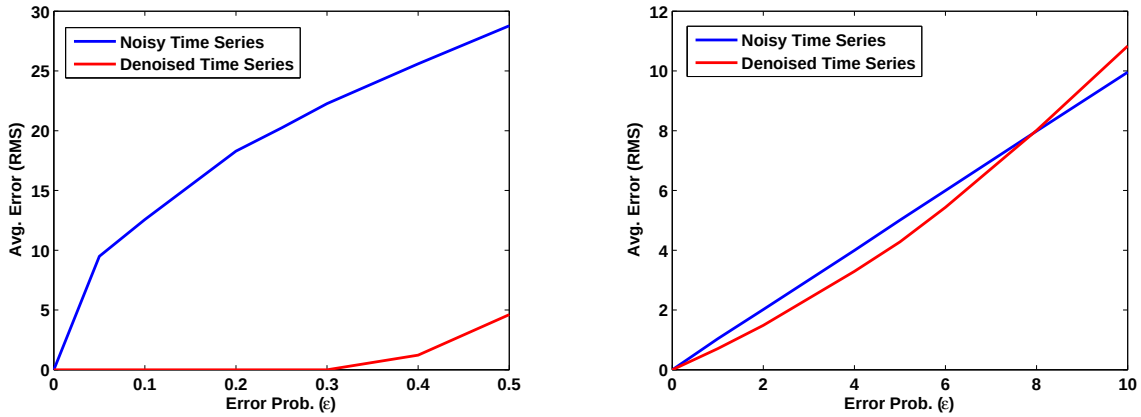
Algorithm 2 Seasonal Sawtooth Wave Time Series Denoising Algorithm

function SAWSEASONALDENOISER($X_{1:T}^{noisy}$)
  for $t = 1$ to $\psi + 1$ **do**
    $w = \lceil \frac{T-t+1}{\psi+1} \rceil$
    $V = [X_t^{noisy}, X_{t+(\psi+1)}^{noisy}, \cdots, X_{t+(w-1)(\psi+1)}^{noisy}]$
    for $i = 1$ to w **do**
      $\hat{X}_{t+(i-1)(\psi+1)} = \textbf{Majority}(V)$
  return $\hat{X}_{1:T}$

series and each block of w consecutive samples are denoised based on this fact that valid w-sample sequences should be in the form of $\Psi_{a:a+w-1}$ for some value of $a \in [0\,,\,\psi]$.

Sawtooth wave time series has the seasonality feature too and one can develop a denoising algorithm based on the seasonality features of the time series. A simple algorithm based on majority vote is illustrated in Algorithm 2. In this algorithm, all the samples that in the noiseless time series should be equal are put in a vector, i.e. V, and then the majority vote of the values in the vector is considered as the denoised value. The time complexity of this algorithm is $O(\psi w^2)$. Simulation results are represented in Fig. 2.6. In this figure, RMS error of the noisy and denoised time series compared to the noiseless one are illustrated for uniform and normal noise models.

While the proposed denoising algorithms can efficiently denoise the sawtooth wave time series based on its locality and seasonality features, for real-world time series more sophisticated denoising methods are required. As a matter of fact, sawtooth wave time series can be considered as a perfect time series for the denoising task which exhibits locality and seasonality features pretty well. The developed denoising algorithms in this section, work



(a) (b)

Figure 2.6: Effect of simple seasonality based denoiser of a sawtooth wave time series with (a) uniform and (b) normal noise models. In this experiment $\psi = 99$.

based on our prior knowledge about the time series and its features. In real situations, we do not have enough prior knowledge about the noiseless time series. . Furthermore, time series behavior may change over time and hence, sophisticated machine learning algorithms should be deployed for the purpose of denoising.

2.3 Case Study: Tourism Time Series

Tourism is considered as one of the most profitable industries around the world and, hence, tourism demand forecasting is important in various domains such as business planning and assessing economic activity in a region [37].

Various prediction techniques have been used in the literature for tourism demand forecasting [64, 37, 6, 76]. In [65], authors used vector autoregressive (VAR) model to forecast tourist arrivals to Macau. In addition to a tourist arrival time series, they also incorporated the costs of living in Macau and the originating country. Various forecasting models to predict international tourism demand in Paris are compared in [28]. In [4], multivariate exponential smoothing methods are deployed for the forecasting of tourism demand for Australia and New Zealand. In [64] a detailed survey is provided on various forecasting techniques.

A number of tourism demand forecasting approaches are based on usage trend data that are supplied by search engines. In [6], GST data about *hotels* and *flights* search terms for popular tourist destinations in the Caribbean are used for forecasting tourism demand. In [76], authors use GST and Baidu trends to predict tourism demand in the Hainan Province of China. In [37], Khadivi and Ramakrishnan proposed a forecasting method that deploys the Wikipedia Usage Trends (WUT) for the Wiki-pages related to Hawaii. It has been

shown through various experiments that WUT improves the demand forecasting in terms of accuracy.

In Section 2.1 we showed that tourism demand time series have seasonality property. In this section, we propose a seasonal regression method for the denoising of tourism demand time series. We represent the tourism demand time series by $Y = y_1, \cdots, y_t$ where y_t is the total number of monthly visitors at time t. In a similar way, Y^D and Y^I are the time series that represent the total number of domestic and international visitors, respectively.

Due to the seasonality of tourism demand, there is a high correlation between the number of visitors to a particular destination at months t and $t \pm \tau$ where τ is the period of season. As an example, for the tourism demand time series of Hawaii, illustrated in Fig. 2.2, τ is 12 and high correlation exists between the value of Y at each month of a year and the corresponding months in the past and future years. When the i^{th} sample of the time series is noisy and we have no information about the noise model, one approach is to replace the noisy value with the predicted value which has been generated based on the previously observed value at $i - \tau$. In other words,

$$\hat{Y}_i = \beta + \alpha Y_{i-\tau} \tag{2.7}$$

where α and β are regression coefficients and can be determined through the learning process.

In order to evaluate this method, we setup an experiment with Hawaii tourism demand time series. Monthly tourism demand time series for Hawaii are available through the website of the *Department of Business, Economic Development and Tourism* of the state of Hawaii[3]. We used 88 month of this dataset starting at January 2008 [37]. Furthermore, the first two-third of the time series used for the training purpose, to estimate the regression coefficients used in Eq. 2.7. For each month of the year we determined a set of regression coefficients. Rest of the time series samples are used for testing the method. We injected normal noise to the actually reported values based on the following model:

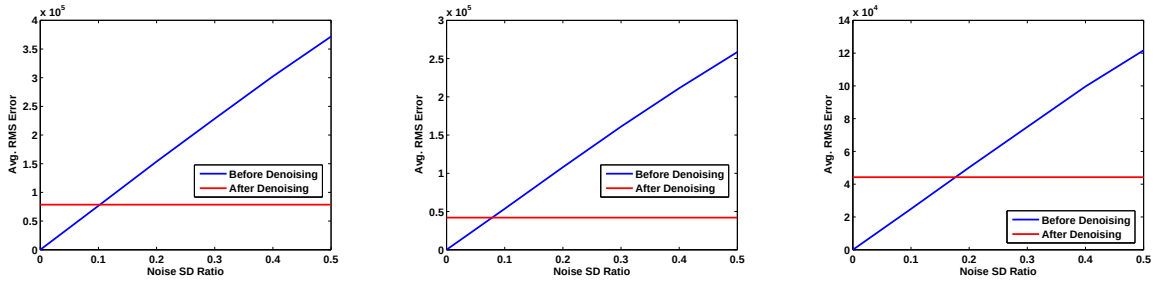
$$y_t^N = \begin{cases} y_t + \lfloor n_t \rceil & ; \quad y_t + \lfloor n_t \rceil \geq 0 \\ 0 & ; \quad y_t + \lfloor n_t \rceil < 0 \end{cases} \tag{2.8}$$

where n_t is a zero mean normal random variable with standard deviation of σ_N, i.e. $n_t \sim N(0, \sigma_N^2)$, and $\lfloor \cdot \rceil$ rounds a real value to the nearest integer value. Value of σ_N is selected based on the maximum value of time series. In other words, we have

$$\sigma_N = \sigma_R \max_t Y_t \tag{2.9}$$

where σ_R is named as the noise standard deviation ratio. This is necessary as the actual number of tourists per month is changing between 100 and 800 thousands and a small amount of noise is negligible. In other words, noise power should be selected with respect to the actual values of time series.

[3] dbedt.hawaii.gov/visitor/tourism/



(a) (b) (c)

Figure 2.7: Seasonal linear autoregression in order to denoise tourism demand time series of Hawaii with normal noise injection (a) Total number of tourists (b) Domestic tourists and (c) International tourists.

Results of this experiment are illustrated in Fig. 2.7. In Fig. 2.7 RMS error of the noisy part of time series Y is illustrated before and after denoising. The same result of Y^D and Y^I are illustrated in Fig. 2.7(b) and (c), respectively. It can be observe that for small values of σ_R the proposed regression method is not effective and cannot efficiently correct the data. However, for larger values of noise SD ratio, the proposed method efficiently reduces the average RMS error.

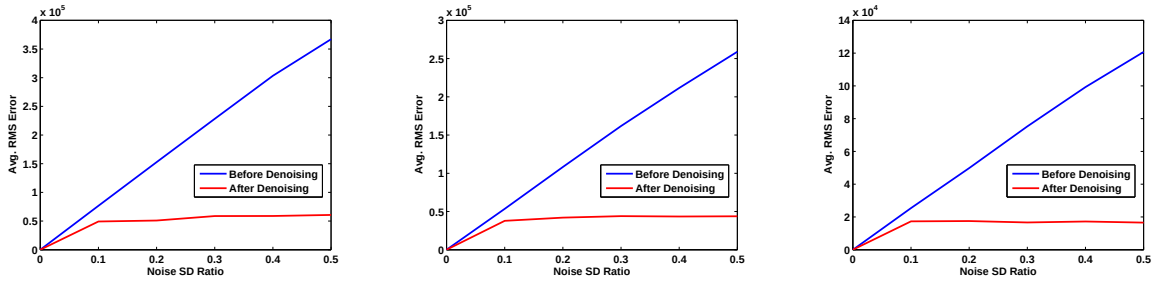
The denoising model of Eq. 2.7 is designed for the situations that we do not have any knowledge about the noise model. If we have information about the noise model we can denoise the time series in a more efficient manner.

Let us assume that we have knowledge about the noise model. This knowledge can be acquired based on the previous observations. Then, the seasonal regression method for correcting the noisy sample at time t can be formulated as follows:

$$\hat{Y}_t = \beta + \alpha Y_{t-\tau} + \gamma Y_t^{noisy} \tag{2.10}$$

where Y_t^{noisy} is the recently observed noisy sample. Moreover, α, β, and γ are regression coefficients and can be determined through the training process. In the training phase, based on the noise model, we inject error into the previously noiseless observations to construct the appropriate training data. Note that if we have access to the previously recorded noisy data, those data samples can be used in the training phase.

To evaluate the regression model of Eq. 2.10, we performed the same set of experiments as described above with Hawaii tourism demand time series. Results of this set of experiments are illustrated in Fig. 2.8. It can be observed that the proposed denoiser improves the RMS error of the time series, with respect to the noiseless data, even for small values of σ_R which is due to the extra knowledge that we have about the noise model.



(a) (b) (c)

Figure 2.8: Seasonal linear regression with known noise model in order to denoise tourism demand time series of Hawaii with normal noise injection (a) Total number of tourists (b) Domestic tourists and (c) International tourists.

2.4 Case Study: Epidemiological Time Series

Disease epidemics are one of the main concerns of public health officials, due to their serious social and economic effects on the societies [3]. Therefore, epidemiological forecasting has been considered as one of the important forecasting applications in the past decades [77, 25, 19, 41, 14]. The previously proposed forecasting methods for influenza outbreak and influenza like illness (ILI) counts use various datasets such as historical data on ILI, weather information [14, 69, 61, 62], the volume of search engine queries [25], Twitter data [35, 17], and Wikipedia usage trends [31].

Domestic and international health organizations (such as World Health Organization[4] (WHO), Centers for Disease Control and Prevention[5] (CDC) in United States, and Pan American Health Organization[6] (PAHO) in Latin America) provide detailed statistics about the number of people with ILI symptoms on a periodic manner. While these reports and statistics play an important role in ILI count forecasting, observations show that these reports are generally based on inaccurate estimations and imperfect data collection mechanisms that result in noisy and unstable time series. Moreover, these statistics are typically announced with a considerable delay. Therefore, denoising of the announced ILI counts is an inevitable task for early consumption of these reports. In this section, we propose a seasonal regression based solution to denoising the ILI time series based on the time series that have been published on PAHO website.

The forecasting model that is used in this section is illustrated in Fig. 2.9. This model uses six different data sources and use an ensemble method to forecast the ILI counts for a particular country [14]. Let us assume that the ILI count time series is $X = x_1, \cdots, x_t$, where x_t is the total number of people with ILI symptoms in a particular week, t. Various

[4]http://www.who.int/en/
[5]http://www.cdc.gov/
[6]http://www.paho.org/hq/

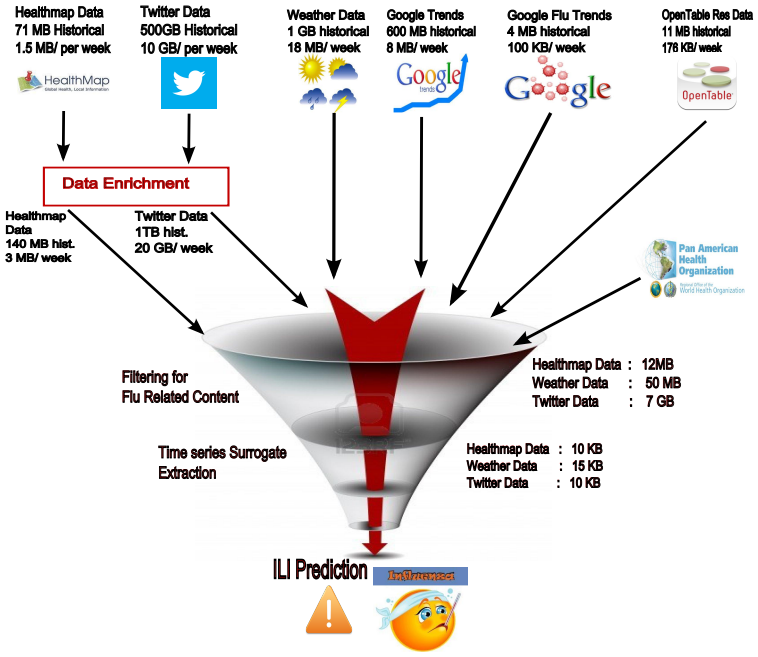


Figure 2.9: The ILI count forecasting model of [14] with six data sources.

regression methods, such as autoregressive model and KNN-based regression, are deployed in the model of Fig. 2.9 to generate forecasts based on each observed data source and then, the outcomes are combined. The whole forecasting method has been introduced in [14].

Preliminary analysis of the data published on PAHO website reveals that the published time series have two important issues. The first issue is that the published values are announced on PAHO website with a considerable delay. This means that when we are at week $t-1$, in order to forecast the value of ILI count on week t, we only have access to the ILI counts up and until week $t-k$ where $k \geq 1$. The second issue of the PAHO ILI count time series is that for some countries, the initial announcement of data for a particular week of the year is inaccurate. These values are then revised multiple times before they are stabilized. Analysis of the data for 14 Latin American countries shows that from stability point of view, ILI time series for different countries have different stabilization behavior. Table 2.2 illustrates the average number of weeks that ILI counts of each country needs to be stabilized. This study has been conducted for the ILI time series published in the season of 2012-2013. In Table 2.2, countries are divided into two groups: slow stabilizing countries and fast stabilizing ones. Figure 2.10(a) compares the relative error of the noisy time series for Argentina, Ecuador, Colombia, and Chile. Relative error has been calculated as follows:

$$e_t^\tau = \frac{x_t^S - x_t^\tau}{x_t^S} \tag{2.11}$$

where x_t^S is the stabilized (i.e. noiseless) value of time series at time t and x_t^τ is the updated value of time series at time t which has been generated $\tau \geq 0$ weeks after t. We can observe

Table 2.2: Stabilization of ILI count time series in different Latin America countries based on PAHO dataset. Average required time for data stabilization are shown in parenthesis.

Slow stabilizing countries	Fast stabilizing countries
Argentina (16 weeks)	Chile (3 weeks)
Colombia (14 weeks)	Peru (3 weeks)
Paraguay (12 weeks)	El Salvador (4 weeks)
Honduras (10 weeks)	Nicaragua (4 weeks)
Mexico (9 weeks)	Costa Rica (5 weeks)
Guatemala (8 weeks)	Panam (5 weeks)
Ecuador (7 weeks)	Bolivia (6 weeks)

from Fig. 2.10(a) that Chile is a fast stabilizing country as the value of e_t^τ converges to zero in a short time while the other three countries are slow stabilizing ones.

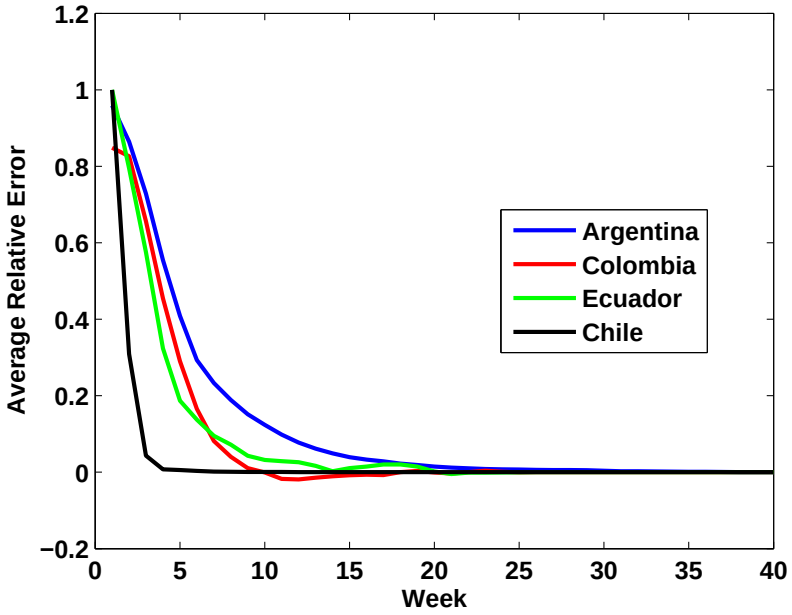
Preliminary studies show that the stabilization of PAHO ILI counts depends on the time of the year. An example is illustrated in Fig. 2.10(b) for Argentina, where weeks of the year are categorized into low, mid, and high seasons depending on the value of ILI count in that particular week and the overall distribution of ILI counts.

In addition to the ILI counts, PAHO dataset also includes the size of the population which has been used to estimate the value of ILI count. This population size is an example of side information and due to its correlation with ILI count time series, can be used to improve the denoising performance. Therefore, based on the seasonality behavior of ILI time series and the available data, for each country we train three regression models: for low, mid, and high season weeks of the year. Each of these regression models works based on the following equation:

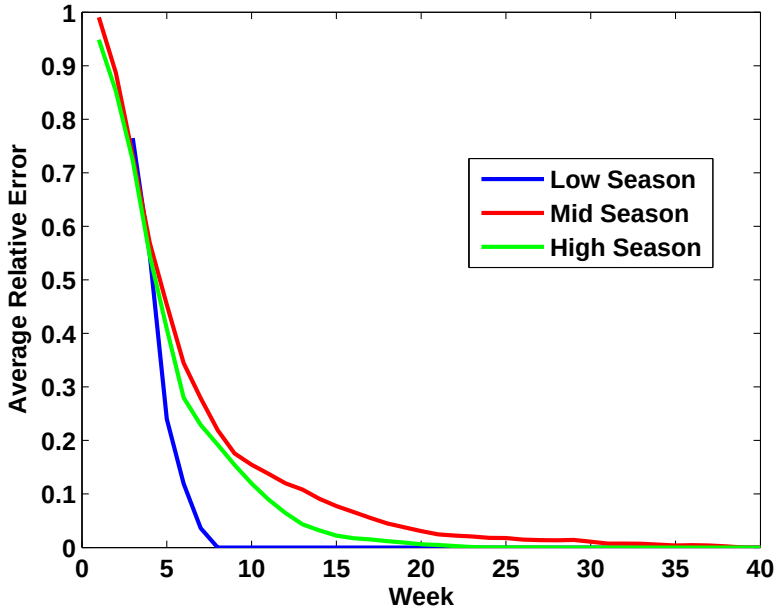
$$\hat{x}_t^\tau = \beta + \alpha\tau + \gamma x_t^\tau + \eta N_t^\tau \tag{2.12}$$

where α, β, γ, and η are the regression coefficients and are determined through the learning process. In this model, $\hat{x}_t^\tau$ is the denoised value of x_t based on its published value τ weeks after t. Moreover, x_t^τ is the unstable (i.e. noisy) value of time series and N_t^τ indicates the size of the population announced on PAHO website. Note that other variations of the denoising model can also be deployed. As an example, depending on the data availability, one may drop the N_t^τ term and perform the denoising task only based on τ and x_t^τ.

In order to evaluate the performance of the proposed denoising method we performed various experiments with PAHO time series. Results are illustrated in Fig. 2.11 and 2.12. Figure 2.11 illustrates the average relative error of values with respect to the stabilized noiseless data before and after denoising. It can be observed from this figure that for most of the countries, the quality of time series have been improved due to denoising. However, for some countries there was no change in the relative error or the relative error has been increased. Inaccurate parameter setting and training due to the lack of sufficient training dataset can be considered as the culprit.



(a)



(b)

Figure 2.10: Average relative error of ILI count values with respect to stable values. (a) Comparison between Argentina, Colombia, Chile, and Ecuador (b) Comparison between different seasons for Argentina.

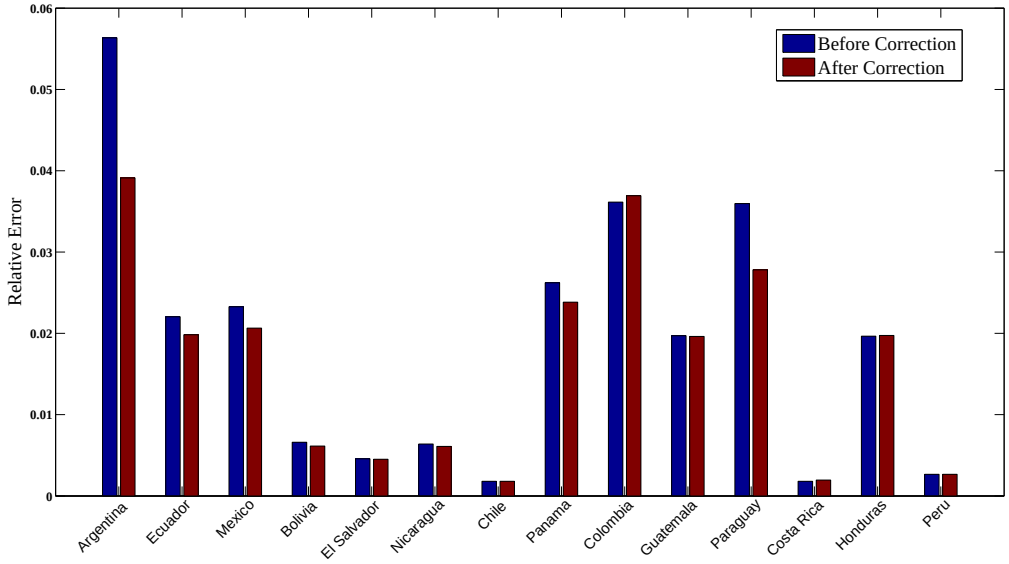


Figure 2.11: Average relative error of ILI count time series before and after denoising for different countries.

Figure 2.12 shows the accuracy of forecasts generated with and without denoising. In this figure, similar to [14], we used the following equation to measure the accuracy:

$$\mathcal{S} = \frac{4}{t_e - t_s + 1} \sum_{t=t_s}^{t_e} \frac{|x_t - \hat{x}_t|}{max(x_t, \hat{x}_t, 10)} \tag{2.13}$$

where $\mathcal{S}$ is a score between 0 and 4. Higher values of $\mathcal{S}$ means that the forecasting results are more accurate. Moreover, t_s and t_e indicate the start and the end of the duration which has been used for the test experiment. Details of this experiment can be found in [14]. Results show that the proposed seasonal regression based denoising method can improve the quality of time series and the accuracy of forecasts.

2.5 Discussion

In this chapter we addressed the locality and seasonality features of time series and based on these features, we proposed basic denoising algorithms for a synthetic sawtooth wave time series. Experimental results indicate that local and seasonal properties of a temporal dataset can be efficiently used in the denoising process. Furthermore, we presented two case studies involving seasonal time series. The first case study considered the tourism demand time series of Hawaii. As we mentioned, tourism demand is a highly seasonal data and hence, a seasonal regression algorithm can be efficiently used for the denoising purposes. In the second

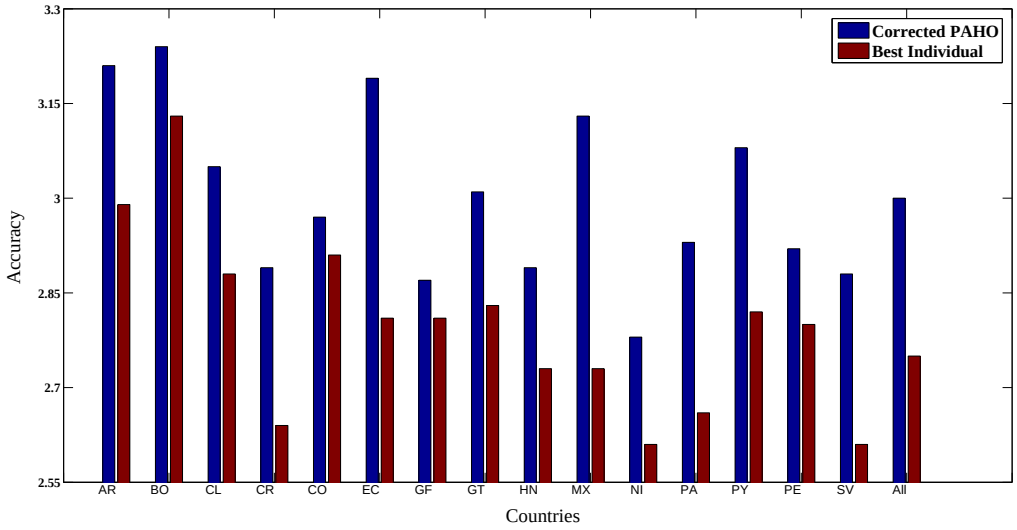


Figure 2.12: Accuracy score, $\mathcal{S}$, of forecasts for each country before and after denoising.

case study, we explored the problem of instability in ILI counts of Latin American countries collected from PAHO dataset. Due to the seasonal behaviors of ILI counts, we presented a seasonal regression method to stabilize these time series. Our results demonstrate that there are significant opportunities to improve forecasting performance using the denoised time series. In the following chapters, various techniques are proposed for time series denoising that are based on the locality features of time series.

Chapter 3

Online Denoising of Discrete Noisy Data

Real-time data-driven systems often utilize discrete valued time series data and their functionality is highly dependent on the accuracy of such data. In order to improve the performance of these systems, an important pre-processing step is the denoising of data before performing any action (e.g. forecasting or control activities). Existing algorithms have primarily focused on the offline denoising problem, which requires the entire data to be collected before the denoising process. In this chapter, the problem of online discrete denoising is considered. The online denoising problem is motivated by real-time applications, where the data must be utilizable soon after it is collected. Three online denoising algorithms are proposed which can strike a tradeoff between delay and accuracy of denoising. It is also shown that the proposed online algorithms asymptotically converge to a class of optimal offline block denoisers. Furthermore, we extend the proposed methods for the denoising of time series with large alphabet. Experimental results with synthetic and real time series support the theoretical results of this chapter.

3.1 Introduction

With emerging data-driven applications, such as data-driven system design, control systems, and data mining, noise removal from data sources is becoming increasingly important [14, 13, 59]. In such applications, we typically encounter data in the form of discrete valued time series, which could either be generated automatically at the output of sensors, or reported manually (such as in collection of disease counts reported by health care personnel [14]). However, such data is often noisy in nature, where the noise could be an artifact of incorrect measurements, faulty sensors, or imperfect data collection mechanisms. In time-sensitive applications, (such as forecasting or control activities), immediate usability of such noisy

data is of critical importance. This leads to the problem of *online denoising*, in which data must be denoised immediately after being collected.

Prior work in denoising literature (such as [73, 51, 54, 12]) has primarily focused on the problem of *offline denoising*, which assumes the availability of the entire data. Weissman et al. in [73] proposed the discrete universal denoiser (DUDE) algorithm for *offline denoising* of x^n from its noisy version z^n, with the assumption of an i.i.d noise generating mechanism (modeled through a noisy memoryless channel $p(z|x)$). DUDE assumes the statistical knowledge of the noisy mechanism, i.e., $p(z|x)$, but makes no assumptions on the distribution of the underlying data x^n. It is shown in [73], that this algorithm converges asymptotically to the optimum Bayes offline denoiser.

Several other offline denoising algorithms have been subsequently developed that are inspired by DUDE, such as [51], [12], and [50]. In [51], original DUDE algorithm is extended for denoising of grayscale images. To this end, an extension for the case of large alphabets is presented. [50] proposes another version of this algorithm named as S-DUDE, which attempts to address the non-stationarity of data. To handle correlation in the noise generating mechanism, an extension for the case of noisy channels with memory is presented in [78] and [26]. Furthermore, [23] addresses the issue of knowledge uncertainty in the statistics of the noise generating mechanism. While the above extensions are indeed valuable, they are primarily targeted at making the denoising mechanisms more robust and do not address the timeliness aspect of denoising, which is of importance in many applications, when data must be denoised on the fly.

In this chapter, we focus on online discrete denoising problem to address the issue of data correction in time-sensitive applications. For this purpose, we precisely formulate the online denoising problem and propose three algorithms for online universal denoising. These algorithms can strike a tradeoff between time-sensitivity and denoising accuracy.

• *Repeatable Online Denoising (ROD)* algorithm, which upon the collection of a new data point, denoises the entire past and current data points. This algorithm sacrifices delay for accuracy and is suitable for scenarios in which the past data could also be useful.

• *One-time Online Denoising (OOD)* algorithm, which only denoises the currently observed data point. This algorithm sacrifices accuracy for delay and is suitable for time-sensitive applications, where the data must be used immediately for real-time forecasting or real-time control commands.

• *Hybrid Online Denoising (HOD)* algorithm, which combines the timeliness of OOD with the accuracy of ROD. In this algorithm, denoising starts by initially applying the ROD algorithm. After a specific period of time, when data statistics become sufficiently reliable, denoising is substituted with OOD to speed up the denoising process.

The proposed algorithms belong to the class of block denoisers and are applicable for denoising the data streams when characteristics of the noise generation mechanism are known and the data points are from a finite alphabet. Universal denoising of an online sequence of

symbols from a finite alphabet, also known as universal filtering, has been addressed before in [74]. In [74] authors study the association between the filtering problem and the problem of prediction of individual sequences and provide a general formulation for the construction of filters. They also show that how their general universal filtering solution may result in an online version of DUDE. However, the filtering solution that has been provided in [74] does not address the issue of tradeoff between time-sensitivity and denoising accuracy. Furthermore, [74] does not provide a detailed algorithm for online filtering.

In this chapter, we construct a universal online denoiser based on DUDE that supports unbalanced contexts. We provide three algorithms to address the tradeoff between time-sensitivity and denoising accuracy and prove that the proposed algorithms asymptotically converge to the optimum block denoiser. We also present numerical results which support the theoretical aspects of this work. Furthermore, we provide theoretical justifications and experimental results that show the original DUDE algorithm and its online offspring are not able to be used for time series that span over a wide range of values (i.e. time series with large alphabet). We extend the proposed methods to address the issue of large alphabet in time series denoising and through various numerical experiments show that the proposed methods can dramatically improve the performance of the denoising process.

3.2 Problem Formulation

We consider an arbitrary discrete-valued n length sequence $x^n = (x_1, \ldots, x_n)$, which is sequentially observed through a noisy mechanism. In particular, at time t, a noisy version of x_t, which is denoted by z_t is observed, where $x_t, z_t \in \mathcal{A} = \{\alpha_1, \cdots, \alpha_M\}$. The noise generating mechanism is assumed to be i.i.d. over time and described through the transition matrix $\Pi_{M\times M}$, where $\Pi(a, b)$ is the probability of observing b if a is the underlying true data. The i^{th} column of Π is illustrated by π_i. We denote by $z^t = (z_1, \ldots, z_t)$ (resp., $x^t = (x_1, \ldots, x_t)$) as the noisy sub-sequence (resp., underlying data sub-sequence) available up and until time t. To measure denoising accuracy, we define a loss function through the matrix, $\Lambda_{M\times M}$, where $\Lambda(a, b)$ is the loss incurred by estimating a by b. The i^{th} column of Λ is illustrated by λ_i. Also, the maximum possible loss is defined as $\Lambda_{max} = \max_{a,b} \Lambda(a, b)$. The goal of online denoising is to sequentially produce a denoised version $\hat{x}^t$ using z^t for each time t.

Definition 1. *Online discrete denoising is the process of reproducing a new sequence of symbols, $\hat{x}^t$, at each time t, based on a received noisy sequence, z^t, such that:*

1. *The total loss in $\hat{x}^t$ is less than the total loss in z^t, i.e.,*

$$\sum_{i=1}^{t} \Lambda(x_i, \hat{x}_i) \leq \sum_{i=1}^{t} \Lambda(x_i, z_i) \tag{3.1}$$

2. *Reproduced symbol, $\hat{x}_t$, is generated with an acceptable amount of delay, i.e., $\delta << n$, after receiving z_t, where n is the length of the entire sequence. Note that under no delay constraint, this reduces to offline denoising.*

We first review the concept of offline denoising and describe DUDE as a universal block denoiser [73]. Let us assume that we have received the *entire noisy sequence* z^n and we want to denoise the symbol at time t, i.e. z_t. The optimum Bayes denoiser is the one that minimizes the expected loss of estimating x_t. In other words, considering the posterior distribution of the *entire* noiseless data, x^n, we have

$$\hat{X}^{opt}\left(z^n\right)[t] = \arg\min_{\hat{x}\in\mathcal{A}} \sum_{\alpha\in\mathcal{A}} \Lambda(\alpha,\hat{x}) \Pr(X_t = \alpha | z^n), \tag{3.2}$$

where the summation is the expected loss of denoising z_t to $\hat{x}$, knowing the posterior distribution of x_t, i.e., $\Pr(X_t = \alpha|z^n)$. In vector notation, (3.2) can be shown as follows

$$\hat{X}^{opt}\left(z^n\right)[t] = \arg\min_{x\in\mathcal{A}} \lambda_{\hat{x}}^T \mathbf{P}_{X_t|z^n} = \arg\min_{x\in\mathcal{A}} \lambda_{\hat{x}}^T \mathbf{P}_{X_t,z^n}. \tag{3.3}$$

where $\mathbf{P}_{X_t|z^n} = [\Pr(X_t = \alpha_1|z^n) \dots \Pr(X_t = \alpha_M|z^n)]^T$.

In [73], it has been shown that this optimum denoiser can also be formulated as follows:

$$\hat{X}^{opt}\left(z^n\right)[t] = \arg\min_{\hat{x}\in\mathcal{A}} \left[\mathbf{P}_{Z_t,z^{n\backslash t}}\right]^T \Pi^{-1} \left[\lambda_{\hat{x}} \odot \pi_{z_t}\right] \tag{3.4}$$

where, $\mathbf{P}_{Z_t,z^{n\backslash t}} = [\Pr(Z_t = \alpha_1, Z^{n\backslash t} = z^{n\backslash t}) \dots \Pr(Z_t = \alpha_M, Z^{n\backslash t} = z^{n\backslash t})]^T$, and $\odot$ is a pair-wise vector multiplication for vectors $\mathbf{u}$ and $\mathbf{v}$ defined as follows:

$$(\mathbf{u} \odot \mathbf{v})[i] = u_i v_i. \tag{3.5}$$

Based on the formulation in (3.4), the DUDE algorithm [73] develops an empirical estimation procedure for $\mathbf{P}_{Z_t,z^{n\backslash t}}$. In particular, it considers a window of size $2k+1$, symmetrically wrapped around time t, i.e. $(z_{t-k}, \cdots, z_{t+k})$. Then, the algorithm takes a pass through the whole sequence and counts all the occurrences of $z_{t-k}, \cdots, z_{t-1}, \beta, z_{t+1}, \cdots, z_{t+k}$ for all the possible values of β. This counting process results in an empirical distribution of symbols that are located at the center of the window. The resulting empirical distribution, which is shown to be an approximation of $\mathbf{P}_{Z_t,z^{n\backslash t}}$, is then used for denoising of z_t. In the next section, we propose online universal denoising algorithms for the case of known channel and discrete finite alphabets. We will prove later that the online algorithms converge to the optimum denoiser.

3.3 Online Universal Denoiser

In this section we introduce the online universal denoiser. We provide the general denoising rule and then discuss various versions of the algorithm that may be considered as the online

version of DUDE [73]. The optimum Bayes denoiser, defined by (3.4), can be appropriately modified for the online problem such that each new noisy symbol, is corrected with a small appropriate delay. Let us assume that we can tolerate the delay of δ symbols and we have received symbols up to time $t+\delta$. Then, the online version of (3.4) can be written as follows

$$\hat{X}^{opt}\left(z^{t+\delta}\right)[t] = \arg\min_{\hat{x}\in\mathcal{A}} \left[\mathbf{P}_{Z_t, z^{(t+\delta)\setminus t}}\right]^T \Pi^{-1}\left[\lambda_{\hat{x}} \odot \pi_{z_t}\right] \tag{3.6}$$

Based on (3.6), with each new received symbol, a new symbol, which is δ steps back of the latest received one, can be optimally denoised. However, it should be noted that with any new received symbol, we have more information about the whole sequence, which in turn, may improve the accuracy of the denoising process. Due to the importance of delay in online denoising, it is desirable to start denoising of a newly received symbol as soon as possible. Therefore, instead of the symmetric double-sided context of [73], we use unbalanced context windows to estimate the conditional distribution $\mathbf{P}_{Z_t, z^{(t+\delta)}}$. Hence, we define the following vector:

$$\mathbf{C}(z^{(t+\delta)}, b^k, c^\delta)[\beta] = \left|\left\{i : k+1 \le i \le t, z_{i-k}^{i+\delta} = b^k \beta c^\delta\right\}\right| \tag{3.7}$$

with the assumption that $\delta \le k$. In fact, $\mathbf{C}(z^{(t+\delta)}, b^k, c^\delta)[\beta]$ is the number of times that we have observed β, wrapped in the context of $(b^k \cdot c^\delta)$ in sequence $z^{(t+\delta)}$. Following the terminology of [73], b^k is the left context, c^δ is the right context, and $(b^k \cdot c^\delta)$ is the double-sided context. Obviously, when $\delta < k$, the double-sided context is unbalanced and when $\delta = 0$, the vector $\mathbf{C}$ is defined only based on the left-context. Using (3.7), online denoising of symbol at time t is performed as follows

$$\hat{X}^{k,\delta}(z^{t+\delta})[t] = \arg\min_{\hat{x}\in\mathcal{A}} \mathbf{C}^T(z^{t+\delta}, z_{t-k}^{t-1}, z_{t+1}^{t+\delta})\Pi^{-1}\left[\lambda_{\hat{x}} \odot \pi_{z_t}\right]. \tag{3.8}$$

Thus, the core aspect of the algorithm is related to counting the number of times that an *unbalanced context* appears inside the received sequence up and until time $t+\delta$. However, as t grows, and more data is collected, it is a time consuming process to start counting the contexts from the beginning of $z^{t+\delta}$. For real-time denoising applications, a more appropriate implementation is to keep the counts up to time $t+\delta$ in memory, and update the counts with a newly received symbol (we denote this memory by $\mathcal{C}$ and the counts by $\mathbf{C}$). It should be noted that with an alphabet of size M and a double-sided context of size $k+\delta$, we need to save $M^{k+\delta}$ vectors, each of size M. In other words, the total amount of memory which is required to keep the current counts in the memory is $O\left(M^{k+\delta+1}\right)$. Therefore, there is a trade-off between the time and memory that depends on the values of M, k, and δ. Depending on the time-sensitivity of the desired denoising process, we next present three online denoising algorithms.

3.3.1 One-time Online Denoiser

In time-sensitive applications, when there is a hard deadline to use the time series, deadline satisfaction is the most important constraint of the system. Therefore, it may not be possible to trade-off time-sensitivity with denoising accuracy. In the first version of the online denoising algorithm, which we name it as One-time Online Denoiser (OOD), the denoiser does not re-process a previously denoised symbol, because it has already been utilized. Hence, it is intuitive to expect higher values of loss for smaller values of t. However, as time grows, we will have more reliable information about the data statistics (in the form of $\mathbf{C}$), and hence, it is to be expected that denoising via OOD improves with time. The pseudo-code of OOD is illustrated in Algorithm 3. In this algorithm, $\mathcal{C}$ represents the memory that keeps the updated counts of $\mathbf{C}$ for all possible unbalanced contexts.

The OOD algorithm starts by retrieving the previous counts for the context vector $(z_{t-k}^{t-1} \cdot z_{t+1}^{t+\delta})$ around the new observed symbol z_t. We denote this count vector by $\mathbf{C}$ and increment one of its elements, i.e., $\mathbf{C}[z_t]$ by 1. Using the updated count vector $\mathbf{C}$, OOD denoises the t^{th} symbol and returns $\hat{x}_t$. Finally, dictionary of all counts ($\mathcal{C}$) is updated by new count vector $\mathbf{C}$. The time complexity of this algorithm, to denoise a single symbol at time t, is $O(M^3)$ where M is the size of alphabet.

3.3.2 Repeatable Online Denoiser

In some applications, when we are not facing with hard deadlines, it is possible to trade-off the denoising accuracy with time-sensitivity. In the second version of denoising algorithms, which we name it as Repeatable Online Denoising (ROD), the denoiser is able to go back and reprocess the whole sequence. Comparing with OOD, ROD is a slower denoising algorithm, while it has higher accuracy and converges faster to the optimum denoiser. With a new received symbol, when we move from $t+\delta$ to $t+\delta+1$, the ROD algorithm reconsiders all the similar contexts in the past and using (3.8), denoises the corresponding symbols again. The pseudo-code of ROD is illustrated in Algorithm 4. The time complexity of this algorithm, to denoise (and re-denoise) all the symbols uptil time t, is $O(tM^3)$ where M is the size of alphabet.

3.3.3 Hybrid Online Denoiser

To overcome the loss in accuracy of OOD and complexity/delay of ROD, a third solution is to use a hybrid algorithm. In this algorithm, which we name it as Hybrid Online Denoising (HOD), ROD algorithm is used for denoising the symbols for the first η symbols, where η is large enough that lets the unbalanced context counts in $\mathbf{C}$ to be stabilized. After receiving the first η symbols, OOD is subsequently used for denoising. Pseudo-code of HOD is presented in Algorithm 5.

Algorithm 3 One-time Online Denoising (OOD) Algorithm

function OOD(t,$z_{t-k}^{t+\delta}$,$\mathcal{C}$,k,δ,Π,Λ)
  $\mathbf{C}(z_{t-k}^{t-1}, z_{t+1}^{t+\delta}) \leftarrow \mathcal{C}[z_{t-k}^{t-1} z_{t+1}^{t+\delta}]$ ▷ (retrieve context counts around z_t)
  $\mathbf{C}(z_{t-k}^{t-1}, z_{t+1}^{t+\delta})[z_t] \leftarrow \mathbf{C}(z_{t-k}^{t-1}, z_{t+1}^{t+\delta})[z_t] + 1$ ▷ (update context count)
  $\hat{x}_t = \arg\min_{\hat{x}\in\mathcal{A}} \mathbf{C}^T \Pi^{-1} [\lambda_{\hat{x}} \odot \pi_{z_t}]$ ▷ (denoise)
  Update $\mathcal{C}[z_{t-k}^{t-1} z_{t+1}^{t+\delta}]$ By $\mathbf{C}$ ▷ (update context dictionary)
  return $\hat{x}_t$

Algorithm 4 Repeatable Online Denoising (ROD) Algorithm

function ROD(t,$z^{t+\delta}$,$\hat{x}^{t+\delta}$,$\mathcal{C}$,k,δ,Π,Λ)
  $\mathbf{C} \leftarrow \mathcal{C}[z_{t-k}^{t-1} z_{t+1}^{t+\delta}]$
  $\mathbf{C}[z_t] \leftarrow \mathbf{C}[z_t] + 1$
  for $i = k+1$ to t **do**
    if $z_{i-k}^{i-1} = z_{t-k}^{t-1}$ and $z_{i+1}^{i+\delta} = z_{t+1}^{t+\delta}$ **then**
      $\hat{x}_i = \arg\min_{\hat{x}\in\mathcal{A}} \mathbf{C}^T \Pi^{-1} [\lambda_{\hat{x}} \odot \pi_{z_i}]$
  Update $\mathcal{C}[z_{t-k}^{t-1} z_{t+1}^{t+\delta}]$ By $\mathbf{C}$
  return $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_t$

Algorithm 5 Hybrid Online Denoising (HOD) Algorithm

function HOD(t,$z_{t-k}^{t+\delta}$,$\mathcal{C}$,k,δ,Π,Λ,η)
  if $t \leq \eta$ **then**
    return ROD(t,$z^{t+\delta}$,$\hat{x}^{t+\delta}$,$\mathcal{C}$,k,δ,Π,Λ)
  return OOD(t,$z_{t-k}^{t+\delta}$,$\mathcal{C}$,k,δ,Π,Λ)

3.4 Properties of Proposed Online Denoisers

In this section, we address the convergence properties of the proposed methods. Consider denoising $z^{t+\delta}$ to the noiseless $x^{t+\delta}$ using an online denoiser and for this purpose we use an unbalanced block denoiser to denoise z_t to x_t. If we use a denoising function, such as $f_t(\cdot)$, that makes decision based on the unbalanced context $z_{t-k}^{t+\delta}$ around the received symbol z_t, then the instantaneous loss occurred by denoising the t^{th} symbol is $\Lambda\left(x_t, \hat{x}_{f_t\left(z_{t-k}^{t+\delta}\right)}\right)$. The total loss incurred by denoising function $f_t(\cdot)$ is defined in Definition 2.

Definition 2. *Total cumulative loss (uptil time t) incurred by denoising function $f_t(\cdot)$ used*

to denoise z^t (when the noiseless sequence is x^t) is $\mathcal{L}_{f_t}$ and is defined as

$$\mathcal{L}_{f_t} = \sum_{t'=k+1}^{t} \Lambda\left(x_{t'}, \hat{x}_{f_t\left(z_{t'-k}^{t'+\delta}\right)}\right). \tag{3.9}$$

We next define a normalized measure of average loss:

Definition 3. *Relative Average Loss (RAL) incurred by denoising function $f_t(\cdot)$ is $\overline{\mathcal{L}}_{f_t}$ and is defined as follows*

$$\overline{\mathcal{L}}_{f_t} = \frac{\mathcal{L}_{f_t}}{(t-k)\Lambda_{max}}. \tag{3.10}$$

It is clear from the above definition that the relative average loss (RAL) for any denoising algorithm satisfies $\overline{\mathcal{L}}_{f_t} \leq 1$, for all t. The behavior and performance of the online denoiser depends on the particular choice of the denoising function f_t. For instance, the denoising function $f_{t'}(z_{t'-k}^{t'+\delta})$ works only based on the available data up to time $t'+\delta$. It should be mentioned that in OOD, at time t', $f_{t'}(z_{t'-k}^{t'+\delta})$ is only used to denoise $z_{t'}$ while in ROD, $f_{t'}(z_{t'-k}^{t'+\delta})$ is also used to re-denoise *all* the previously denoised symbols from $t = k+1$ to $t = t'-1$. It is readily seen that ROD is actually an online version of an unbalanced offline block denoiser that behaves similar to DUDE [73]. In fact, when we want to denoise the symbol at time t and data up to $t+\delta$ is available, ROD denoises the whole data $z^{t+\delta}$. Subsequently, when we receive a new symbol, $z_{t+\delta+1}$, ROD denoises the whole data $z^{t+\delta+1}$. This is similar to using an offline block denoiser repeatedly and hence, similar to [73], it can be shown that ROD asymptotically converges to the optimum Bayes denoiser.

We next focus on the properties of the OOD algorithm. In OOD algorithm, at each time step, we only denoise one symbol. In other words, when we receive $z_{t+\delta}$, OOD denoises z_t. The following lemma shows that the asymptotic behavior of OOD is close to unbalance offline block denoiser that uses (3.8) when all the data, z^n is available.

Lemma 1. *Let*

$$\varphi(p) = \begin{cases} \frac{1}{1-2p}\log\frac{1-p}{p} & 0 \leq p < \frac{1}{2} \\ \frac{1}{2p(1-p)} & \frac{1}{2} \leq p \leq 1 \end{cases}$$

Also, define the following vectors at time t:

$$\mathbf{C}_t^{OFF} = \mathbf{C}(z^n, z_{t-k}^{t-1}, z_{t+1}^{t+\delta}), \;\; \mathbf{C}_t^{OOD} = \mathbf{C}(z^{t+\delta}, z_{t-k}^{t-1}, z_{t+1}^{t+\delta})$$

Then, for all $z^n \in \mathcal{A}^n$ we have

$$\begin{aligned} &\Pr\left(\left\|\frac{\mathbf{C}_t^{OFF}}{n} - \frac{\mathbf{C}_t^{OOD}}{t+\delta}\right\|_1 \geq \epsilon\right) \\ &\qquad \leq (2^M - 2)e^{-t(\min_{A\subseteq\mathcal{A}}\varphi(P(A)))\frac{\epsilon^2}{4}}, \end{aligned} \tag{3.11}$$

where $P(A) = \sum_{\alpha\in A}\frac{\mathbf{C}_t^{OFF}[\alpha]}{n}$.

Proof. *It is shown in [73] (Proposition 1) that for a probability distribution vector* $\mathbf{P}$ *with length of* M *and its empirical estimation* $\hat{\mathbf{P}}$*, that has been estimated using* t *observations, we have*

$$\Pr\left(\left\|\mathbf{P}-\hat{\mathbf{P}}\right\|_1 \geq \epsilon\right) \leq (2^M-2)e^{-t(\min_{A\subseteq\mathcal{A}}\varphi(P(A)))\frac{\epsilon^2}{4}} \tag{3.12}$$

The proof of the Lemma follows directly by substituting $\mathbf{P}$ *by* $\frac{\mathbf{C}_t^{OFF}}{n}$ *and* $\hat{\mathbf{P}}$ *by* $\frac{\mathbf{C}_t^{OOD}}{t+\delta}$.

The above lemma (i.e., (3.11)) shows that the empirical pmf (obtained by the counts in the observed vector $z^{t+\delta}$ for a context) of OOD converges in probability to the empirical pmf of the unbalanced offline block denoiser. Similar to (3.8), let us define the offline unbalanced denoiser (denoted by OFF) as:

$$\hat{X}^{k,\delta}(z^n)[t] = \arg\min_{\hat{x}\in\mathcal{A}} \mathbf{C}^T(z^{t+\delta}, z_{t-k}^{t-1}, z_{t+1}^{t+\delta})\Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right]. \tag{3.13}$$

Lemma 1 shows that when we estimate the empirical distribution of z_t wrapped in the context of $(z_{t-k}^{t-1}\cdot z_{t+1}^{t+\delta})$ using OOD, this distribution asymptotically converges to the empirical distribution of z_t wrapped in the context of $(z_{t-k}^{t-1}\cdot z_{t+1}^{t+\delta})$ using the offline block denoiser. Now we show that convergence property of Lemma 1 results in the convergence of the final estimation of $\hat{x}_t^{OOD}$ to $\hat{x}_t^{OFF}$ where $\hat{x}_t^{OFF}$ is the result of denoising by the offline block denoiser. It is easy to observe that denoising rules of (3.8) and (3.13) for OOD and offline denoisers can be written as follows:

$$\hat{x}^{OOD}[t] = \arg\min_{\hat{x}\in\mathcal{A}} \left(\frac{\mathbf{C}_t^{OOD}}{t+\delta}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right], \tag{3.14}$$

and

$$\hat{x}^{OFF}[t] = \arg\min_{\hat{x}\in\mathcal{A}} \left(\frac{\mathbf{C}_t^{OFF}}{n}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right]. \tag{3.15}$$

Then, we can write the OOD denoiser output at time t as

$$\begin{aligned}
&\hat{x}^{OOD}[t] \\
&= \arg\min_{\hat{x}\in\mathcal{A}} \left(\frac{\mathbf{C}_t^{OOD}}{t+\delta} + \frac{\mathbf{C}_t^{OFF}}{n} - \frac{\mathbf{C}_t^{OFF}}{n}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right] \\
&= \arg\min_{\hat{x}\in\mathcal{A}} \left(\frac{\mathbf{C}_t^{OFF}}{n}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right] + \\
&\qquad \left(\underbrace{\frac{\mathbf{C}_t^{OOD}}{t+\delta} - \frac{\mathbf{C}_t^{OFF}}{n}}_{\to 0 \text{ in probability}}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}\odot\pi_{z_t}\right].
\end{aligned} \tag{3.16}$$

However, we proved in Lemma 1 that $\frac{\mathbf{C}_t^{OOD}}{t+\delta}$ asymptotically converges to $\frac{\mathbf{C}_t^{OFF}}{n}$. Hence, as t grows and more data is collected, it is expected that $\hat{x}^{OOD}[t]$ converges to $\hat{x}^{OFF}[t]$. We have showed that OOD asymptotically converges to the offline block denoiser with unbalanced context.

Using an approach similar to [73], it can be shown that the offline block denoiser with unbalanced context that denoises based on (3.13) asymptotically converges to the optimum Bayes denoiser.

Proposition 1. *The offline unbalanced block denoiser that denoises based on (3.13) asymptotically converges to the Bayes optimal denoiser as defined in (3.4).*

Proof. *Let us define the following M-vector, which is a modified version of vector $\mathbf{q}$ defined in [73]:*

$$\mathbf{q}(a^n, b^n, c^{k+\delta+1})[\alpha] = |\{i : k+1 \leq i \leq n-\delta, b_i = \alpha\}| \tag{3.17}$$

In this vector, the α^{th} position is the number of times that $c^{k+\delta+1}$ is observed in a^n while the i^{th} symbol in b^n, corresponding to the $k+1^{th}$ symbol in $c^{k+\delta+1}$ is α. It is easy to observe that

$$\sum_{\alpha} \mathbf{q}\left(a^n, b^n, c^{k+\delta+1}\right)[\alpha] = \mathbf{C}\left(a^n, c_1^k, c_{k+2}^{k+\delta+1}\right)[c_{k+1}] \tag{3.18}$$

It should be noted that in the offline block denoiser, the total loss can be determined using the following equation:

$$\mathcal{L}_f = \sum_{u_{-k}^{\delta} \in \mathcal{A}^{k+\delta+1}} \lambda_{f\left(u_{-k}^{\delta}\right)}^T \mathbf{q}\left(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}\right) \tag{3.19}$$

where, instead of time-dependent $f_t(\cdot)$, we used the offline block denoiser function $f(\cdot)$.

For the optimum offline block denoiser with right-context of size k and left-context of size δ, the total loss can be determined by (3.19). Then, knowing the true distribution of $\mathbf{q}\left(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}\right)$, the optimum block denoising function that minimizes the loss is as follows:

$$f(u_{-k}^{\delta}) = \arg\min_{\hat{x} \in \mathcal{A}} \lambda_{\hat{x}}^T \mathbf{q}\left(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}\right) \tag{3.20}$$

Similar to [73] (Lemma 3 and using Proposition 1), we can prove that the following estimation is asymptotically correct:

$$\begin{aligned} &\mathbf{q}\left(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}\right)[a] \\ &\approx \Pi(a, u_0) \sum_{b \in \mathcal{A}} \mathbf{q}\left(z^{t+\delta}, x^{t+\delta}, u_{-k}^{-1} b u_1^{\delta}\right)[a] \end{aligned} \tag{3.21}$$

In vector notation, (3.21) can be written as

$$\mathbf{q}(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}) \approx \pi_{u_0} \odot \sum_{b \in \mathcal{A}} \mathbf{q}(z^{t+\delta}, x^{t+\delta}, u_{-k}^{-1} b u_1^{\delta}) \tag{3.22}$$

Summing both sides of (3.22), using Eq.3.18, and by iterating over all possible values of u_0 we will have:

$$\mathbf{C}\left(z^{t+\delta}, u_{-k}^{-1}, u_1^{\delta}\right) \approx \Pi^T \left[\sum_{b\in\mathcal{A}} \mathbf{q}(z^{t+\delta}, x^{t+\delta}, u_{-k}^{-1} b u_1^{\delta})\right] \tag{3.23}$$

From (3.23) we have:

$$\sum_{b\in\mathcal{A}} \mathbf{q}(z^{t+\delta}, x^{t+\delta}, u_{-k}^{-1} b u_1^{\delta}) \approx \Pi^{-T}\mathbf{C}\left(z^{t+\delta}, u_{-k}^{-1}, u_1^{\delta}\right) \tag{3.24}$$

and using (3.22) we have:

$$\mathbf{q}(z^{t+\delta}, x^{t+\delta}, u_{-k}^{\delta}) \approx \pi_{u_0} \odot \left[\Pi^{-T}\mathbf{C}\left(z^{t+\delta}, u_{-k}^{-1}, u_1^{\delta}\right)\right] \tag{3.25}$$

By combining (3.25) with (3.20) we can observe that the optimum denoising function is

$$f(u_{-k}^{\delta}) = \arg\min_{\hat{x}\in\mathcal{A}} \lambda_{\hat{x}}^T \pi_{u_0} \odot \left[\Pi^{-T}\mathbf{C}\left(z^{t+\delta}, u_{-k}^{-1}, u_1^{\delta}\right)\right] \tag{3.26}$$

It is easy to see that $\odot$ operator has the following property:

$$(\mathbf{u}\odot\mathbf{v})^T.\mathbf{w} = \mathbf{u}^T.(\mathbf{v}\odot\mathbf{w}) \tag{3.27}$$

Using (3.27), we can show:

$$f(u_{-k}^{\delta}) = \arg\min_{\hat{x}\in\mathcal{A}} \mathbf{C}\left(z^{t+\delta}, u_{-k}^{-1}, u_1^{\delta}\right)^T \Pi^{-1}\left[\lambda_{\hat{x}}^T \odot \pi_{u_0}\right] \tag{3.28}$$

which is similar to (3.13). This equivalency shows that the offline unbalanced block denoiser, defined by (3.13), converges to the optimum Bayes denoiser.

3.5 Numerical Results

To investigate the performance of the proposed algorithms, we performed various experiments. In this section, we present some numerical results for the case of binary alphabet, i.e., $\mathcal{A} = \{0, 1\}$, and the noisy mechanism is modeled through a binary symmetric channel (BSC) with crossover probability μ. We run the denoising algorithm for 100 randomly generated texts and report the average RAL over all cases. Each of the random texts contained binary data of length $n = 1000$. Before generating the i^{th} random text, we first generate a pattern dictionary which contains 12 patterns, $\mathcal{P}_i^{\mathcal{S}} = \{\mathcal{P}_{i,1}, \cdots, \mathcal{P}_{i,12}\}$, where $\mathcal{P}_{i,j}$, for $j = 1$ to 10 are randomly generated patterns with length 10 bits, generated using uniform distribution of 0s and 1s. Also, $\mathcal{P}_{i,11}$ and $\mathcal{P}_{i,12}$ are 0 and 1, respectively. The i^{th} text is the random sequence of patterns in $\mathcal{P}_i^{\mathcal{S}}$, where each pattern is selected with equal probability. We assumed that the crossover probability of the BSC is $\mu = 0.2$ and the loss matrix is

$\Lambda(0,0) = \Lambda(1,1) = 0$, and $\Lambda(0,1) = \Lambda(1,0) = 1$. Furthermore, we assumed that k is 3 and we show results for $\delta = 0$ and $\delta = 3$.

The RAL of various approaches for $\delta = 0$ are compared in Figure 3.1. In this figure, the proposed OOD, ROD, and HOD algorithms are compared with DUDE and Unbalanced DUDE, where by unbalanced, we mean that left context has the size of k and right context has the size of δ. Relative average loss of the noisy uncorrected sequence is also illustrated in this figure. It can be observed in Fig. 3.1 that as we proved in the previous section, online algorithms converge to the offline algorithm. It is also observable that ROD (red curve) shows lower RAL than OOD and HOD which shows that ROD convergence is faster than other ones. However, the faster convergence of ROD comes at a higher computational cost since ROD re-denoises all the past symbols upon receiving a new noisy symbol.

The effect of δ on the RAL of the proposed algorithms is shown in Fig. 3.2. It should be noted that when k is 3, $\delta = 3$ means that online denoisers use balanced double-sided contexts as is used in DUDE [73]. From Fig. 2 it is obvious that for this specific test case, the average performance of online denoisers is better when δ is 0 which shows that *balanced denoisers are not necessarily better than unbalanced denoisers*. Results of Fig. 3.2 confirms that ROD results in better RAL compared to the other online denoisers.

Running times of the online denoising algorithms are compared in Fig. 3.3. This figure illustrates that how long does it take to denoise each symbol of the time series for the first time. It can be observed that ROD has the worst running time while OOD demonstrates the smallest running time. This is consistent with the behavior of the algorithms. The average total denoising times required to denoise the whole time series for OOD, ROD, and HOD are 0.0128, 7.2072, and 0.2488 seconds, respectively. These times are measured with the implementation of the algorithms in MATLAB R2014a over an Intel Core i7 computer with 8 GBytes of RAM and 64-bit Windows 10 Home edition operating system.

3.6 Denoising of Time Series with Large Alphabet

The drawback of the original DUDE method, and hence its online offspring, is that they demonstrate an unacceptable level of accuracy when the source alphabet is large. In fact, the accuracy of these algorithms greatly depends on the empirical distribution of symbols that is estimated based on the observed samples. In Lemma 1, we proved that

$$\begin{aligned} &\Pr\left(\left\|\frac{\mathbf{C}_t^{OFF}}{n} - \frac{\mathbf{C}_t^{OOD}}{t+\delta}\right\|_1 \geq \epsilon\right) \\ &\qquad \leq (2^M - 2)e^{-t(\min_{A\subseteq\mathcal{A}}\varphi(P(A)))\frac{\epsilon^2}{4}}, \end{aligned} \tag{3.29}$$

where $\mathbf{C}_t^{OFF}$ and $\mathbf{C}_t^{OOD}$ are the empirical distributions of the observed contexts in offline DUDE and online denoising algorithm (i.e. OOD), respectively. In this equation, t and n

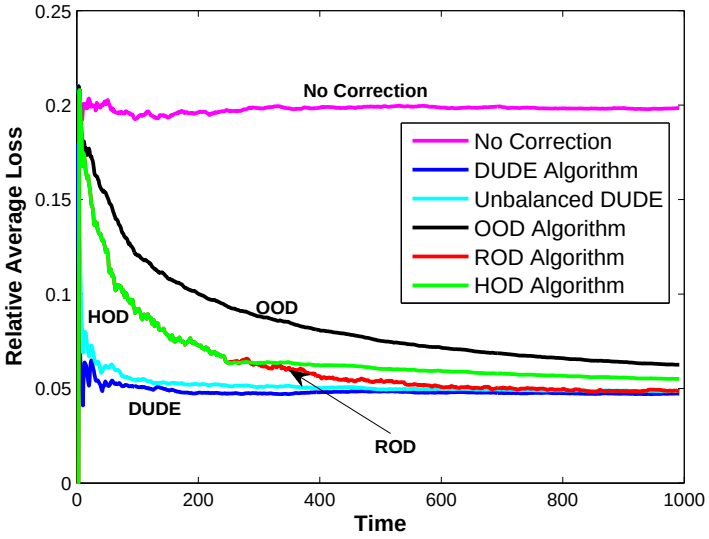


Figure 3.1: Relative average loss of BSC example.

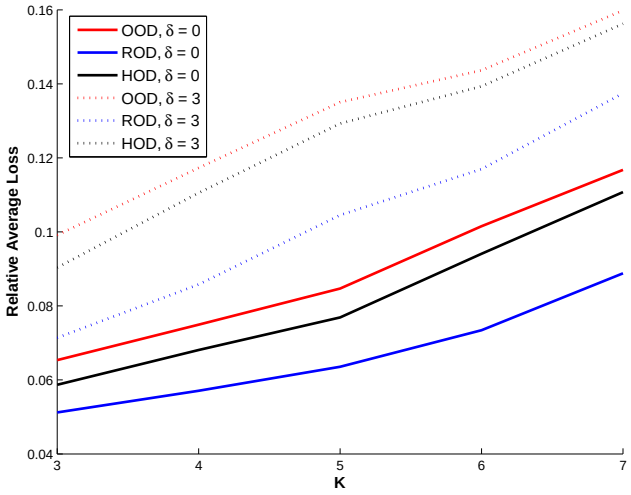


Figure 3.2: Effect of (k, δ) on ratio of errors in BSC example.

represent the number of observed symbols in online and offline algorithms, and M is the size of alphabet. The upper bound in Eq. 3.29 exponentially grows with the size of alphabet, M, and in fact depends on the difference between the size of alphabet, M, and number of observed samples, t. Therefore, with large alphabets we expect that the denoising algorithm fails to achieve an acceptable level of accuracy.

When we perform forecasting with real-world time series, dealing with large alphabets is

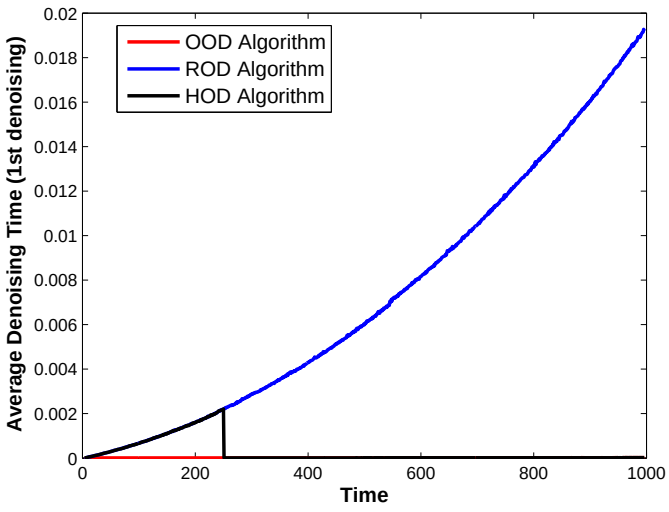


Figure 3.3: Running time of the online denoising algorithms for the first time that each symbol is denoised.

inevitable. As an example, in the case of Influenza forecasting for Argentina in 2012-2013, ILI counts change from 209 to 5136. This means that for this forecasting problem we are facing with an alphabet that has $M = 4928$ symbols. On the other hand, during the same period of time, we only observe 58 samples. If we assume that in Eq. 3.29, $\min_{A\subseteq\mathcal{A}} \varphi(P(A)) = 2$ (note that $\varphi(P(A)) \geq 2$), then we will have

$$(2^M - 2)e^{-t(\min_{A\subseteq\mathcal{A}} \varphi(P(A)))\frac{\epsilon^2}{4}} \geq e^{3465.7-29\epsilon^2}. \tag{3.30}$$

The above example illustrates that DUDE-based denoisers may have serious issues with time series that span over a wide range of values. In the rest of this chapter, we explore online and offline denoising of time series with large alphabet.

3.6.1 Quantization-based DUDE

Let us assume that the alphabet is $\mathcal{A} = \{0, 1, 2, \cdots, M\}$ and we have a time series $X = x_1, \cdots, x_t$ such that $x_j \in \mathcal{A}$. Note that in this case, the size of alphabet is $M + 1$. We can represent each sample of time series, x_j, by the following values:

$$\begin{aligned} x_j^q &= \left\lfloor \frac{x_j}{Q} \right\rfloor \\ x_j^r &= x_j - Q \left\lfloor \frac{x_j}{Q} \right\rfloor \end{aligned} \tag{3.31}$$

where, Q is a quantization factor and $\lfloor\cdot\rfloor$ is the floor operator. In other words:

$$x_j = Qx_j^q + x_j^r. \tag{3.32}$$

Then, we are representing the time series, X, with two new time series $X^q = x_1^q, \cdots, x_t^q$ and $X^r = x_1^r, \cdots, x_t^r$. We name this representation as *double quantization* and it has the following properties:

1. As we have shown in Eq. 3.32, the original time series can be reconstructed from the quantized time series.
2. The size of alphabet in X^q and X^r is smaller than $M+1$. Size of alphabet in X^q is $\left\lfloor \frac{M}{Q} \right\rfloor + 1$ and size of alphabet in X^r is Q.
3. The quantization is transparent to noise. In other words, if sample x_j is noisy, at least one of the X^q and X^r samples are noisy too.

Due to the transparency of double quantization to noise (the third property), any error in the original noisy time series, Z, is transmitted to the new ones, Z^q and Z^r. These time series can then be denoised with any appropriate algorithm and then, using the first property of double quantization, we can combine the denoised versions of Z^q and Z^r to generate the denoised version of X. Due to the second property of double quantization, we can use DUDE and its online offspring more efficiently. This means that we can apply DUDE to Z^q and Z^r to generate the denoised time series $\hat{X}^q$ and $\hat{X}^r$, and combine $\hat{X}^q$ and $\hat{X}^r$ using Eq. 3.32 to generate the denoised version of Z, i.e. $\hat{X}$. We name this approach as *Double Quantized DUDE*, or DQ-DUDE.

Algorithm 6 Double Quantized DUDE (DQ-DUDE) Algorithm

function DQ-DUDE(t,$Z = z_1^n$,k,Q,Π,Λ)
  for $i = 1$ to n **do**
    $z_i^q = \left\lfloor \frac{z_i}{Q} \right\rfloor$
    $z_i^r = z_i - Q z_i^q$
  Construct Π^q and Λ^q matrices for Z^q
  Construct Π^r and Λ^r matrices for Z^r
  $\hat{x}_i^q$ = DUDE(t,Z^q,k,Π^q,Λ^q)
  $\hat{x}_i^r$ = DUDE(t,Z^r,k,Π^r,Λ^r)
  return $\hat{x}_i^r + Q\hat{x}_i^q$

The algorithm for DQ-DUDE is illustrated in Algorithm 6. Note that after converting the original time series to its quantized equivalents, we also need to modify the loss, Λ, and transition, Π, matrices as well to accommodate new time series. In this algorithm, Π^q and Λ^q represent the corresponding matrices for Z^q while Π^r and Λ^r are the transition and loss matrices for Z^r. The time complexity of this algorithm to denoise a time series of length n is $O(nM^3)$ where M is the size of alphabet.

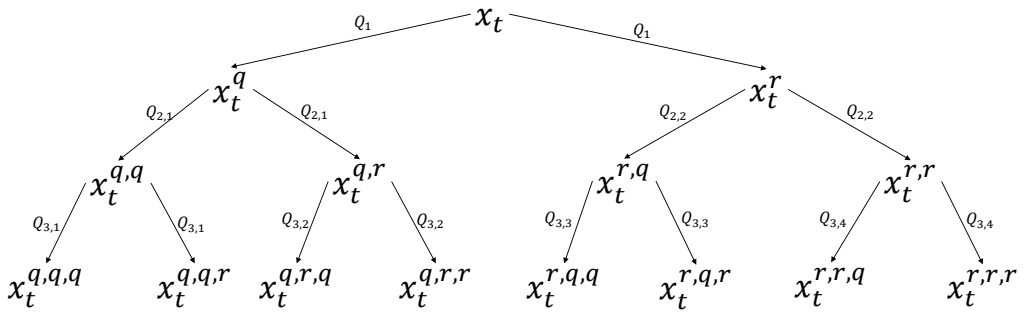


Figure 3.4: Hierarchical model of DQ-DUDE.

3.6.2 Hierarchical DQ-DUDE

The proposed DQ-DUDE algorithm can be extended in a hierarchical manner. In fact, if we are working with a very large alphabet, even one level of quantization may not suffice. Note that using a small value for the quantization factor, Q, results in a large alphabet for X^q while a large Q results in a large alphabet for X^r. Hence, more levels of double quantization are required in a hierarchical manner. This hierarchical model is illustrated in Fig. 3.4. In this figure, Q_i's are the quantization factors, and to distinguish different time series appropriate superscripts are used. As an example, $x_t^{q,r,q}$ belongs to the time series that is generated after three levels of quantization: x_t^q in the first level, $x_t^{q,r}$ in the second level, and $x_t^{q,r,q}$ in the third level. It is easy to observe that hierarchical double quantization is transparent to noise. Furthermore, the original time series can be reconstructed using the quantized values. As an example, in Fig. 3.4 we have:

$$\begin{aligned} x_t =& x_t^{r,r,r} + Q_{3,4} x_t^{r,r,q} + Q_{2,2}\left(x_t^{r,q,r} + Q_{3,3} x_t^{r,q,q}\right) + \\ & Q_1\left(x_t^{q,r,r} + Q_{3,2} x_t^{q,r,q} + Q_{2,1}\left(x_t^{q,q,r} + Q_{3,1} x_t^{q,q,q}\right)\right) \end{aligned} \tag{3.33}$$

The hierarchical approach of Fig. 3.4 is a complete three level of quantization. However, depending on the deployed quantization factors, it is not necessary to perform quantization in some directions. As an example, if we use $Q = 10$ as the quantization factor in all levels, after the first quantization X^r varies between 0 and 9 which typically does not need any further quantization. Moreover, as depicted in Fig. 3.4, different quantization factors may be used in different levels of the hierarchy.

3.6.3 Online Hierarchical Double Quantization Denoising

Online versions of the hierarchical DQ-DUDE algorithm can be developed in a similar way that we developed the online universal denoising algorithms in Section 3.3. An example is illustrated in Algorithm 7. In this algorithm, similar to Algorithm 6, we first perform the

Algorithm 7 Double Quantized OOD (DQ-OOD) Algorithm

function DQ-OOD(t,$z_{t-k}^{t+\delta}$,$\mathcal{C}^q$,$\mathcal{C}^r$,k,δ,Π^q,Λ^q,Π^r,Λ^r)
  for $i = t-k$ to $t+\delta$ **do**
    $z_i^q = \left\lfloor \frac{z_i}{Q} \right\rfloor$
    $z_i^r = z_i - Q z_i^q$
  $\hat{x}_i^q = \text{OOD}(t,(z_{t-k}^{t+\delta})^q,\mathcal{C}^q,k,\delta,\Pi^q,\Lambda^q)$
  $\hat{x}_i^r = \text{OOD}(t,(z_{t-k}^{t+\delta})^r,\mathcal{C}^r,k,\delta,\Pi^r,\Lambda^r)$
  return $\hat{x}_i^r + Q\hat{x}_i^q$

double quantization on the noisy data and then, OOD algorithm of Algorithm 3 is called on each of the new time series. Note that $\mathcal{C}^q$ and $\mathcal{C}^r$ are the memory spaces that store the context counts for the corresponding time series. Furthermore, loss and transition matrices are calculated offline to speedup the online algorithm. The time complexity of this algorithm, to denoise a time series of length n, is $O(nM^3)$ where M is the size of alphabet.

Another benefit of double quantization, in addition to its importance for using the discrete universal denoisers for large alphabet sources, is that it helps us to use less memory space and run the denoising process in a shorter time. In Section 3.3 we mentioned that the total required memory to keep the context counts is $O(M^{k+\delta+1})$. With one level of double quantization, we are working with two time series with alphabet sizes Q and $\left\lfloor \frac{M}{Q} \right\rfloor + 1$. Hence, the total memory space which is required to store the context counts is $O\left(Q^{k+\delta+1} + \left\{\left\lfloor \frac{M}{Q} \right\rfloor + 1\right\}^{k+\delta+1}\right)$. Considering the fact that

$$\begin{aligned} 0 < Q < M \\ 0 < \left\lfloor \frac{M}{Q} \right\rfloor + 1 < M \\ Q + \left\lfloor \frac{M}{Q} \right\rfloor + 1 < M \end{aligned}$$

it is easy to show that the following inequality holds

$$Q^{k+\delta+1} + \left\{\left\lfloor \frac{M}{Q} \right\rfloor + 1\right\}^{k+\delta+1} < M^{k+\delta+1}$$

and hence, total memory space is lower in DQ-OOD algorithm. Furthermore, due to the smaller size of alphabets in DQ-OOD, all the vectors and matrices are smaller and hence matrix multiplications in Eq. 3.8 can be performed faster which results in shorter running time of the algorithm.

3.7 Experimental Results

In order to evaluate the proposed discrete universal denoising algorithms we performed an extensive set of experiments with real-world and synthetic time series. In these experiments we aimed to answer the following questions:

- What is the relationship between alphabet size, time series length, and the required level of quantization?
- What is the performance of the proposed DQ-DUDE algorithm?
- What is the performance of the proposed double quantization online denoisers?
- What are the corresponding running times and which algorithms are faster?

In this section, we present the experimental results.

3.7.1 Synthetic Time Series

Similar to the experiments we performed in Section 2.2, in this section we use sawtooth wave time series to perform preliminary simulations and study the relationships between the size of alphabet and required level of quantization. In this set of experiments, value of the time series changes between 0 and $M-1$ where M exponentially varies between 2 and 2048. We use hierarchical double quantization with one, two, and three levels of quantization. Also, the quantization factor in all the experiments is $Q = 10$ and quantization is performed on the original time series as well as the quantized time series with larger alphabet, i.e. Z^q, and $Z^{q,q}$. Uniform noise is injected to each time series for 10 times and the average loss for each time series and denoising method is measured. For each value of time series, loss is defined as the absolute difference between the noiseless value and the noisy or denoised values. The uniform noise model is similar to Eq. 2.4 with $\epsilon = 0.3$. Results are represented in Table 3.1 and Fig. 3.5. In this set of experiments the size of the double-sided context window is 3, i.e. $k = 1$.

It can be observe from the simulation results of Fig. 3.5 and Table 3.1 that with the fixed size of data, as the size of alphabet grows, higher levels of double quantization is required. Furthermore, results show that a logarithmic relationship exists between the alphabet size and the level of quantization hierarchy. This is consistent with the previous observations (e.g. in [73]) that the maximum supportable alphabet size is of the logarithmic order of the data size.

Table 3.1: Effect of alphabet size on performance of the denoisers.

Alphabet Size	Loss Noisy Time Series	Loss Original DUDE	Loss DQ-DUDE 1-Level	Loss DQ-DUDE 2-Level	Loss DQ-DUDE 3-Level
2	763.5	307.3	-	-	-
4	1791.0	815.0	-	-	-
8	3826.9	1789.6	-	-	-
16	7953.8	4396.5	5986.3	-	-
32	15968.7	12286.0	9576.8	-	-
64	32461.2	32461.2	17855.2	-	-
128	65813.2	65813.2	40068.3	39913.0	-
256	129433.1	129433.1	103980.2	63785.4	-
512	259928.8	259928.8	260169.5	126222.9	-
1024	525012.1	525012.1	525320.0	267966.3	267930.8
2048	4200672.2	4200672.2	4202513.0	2234917.2	2000449.8

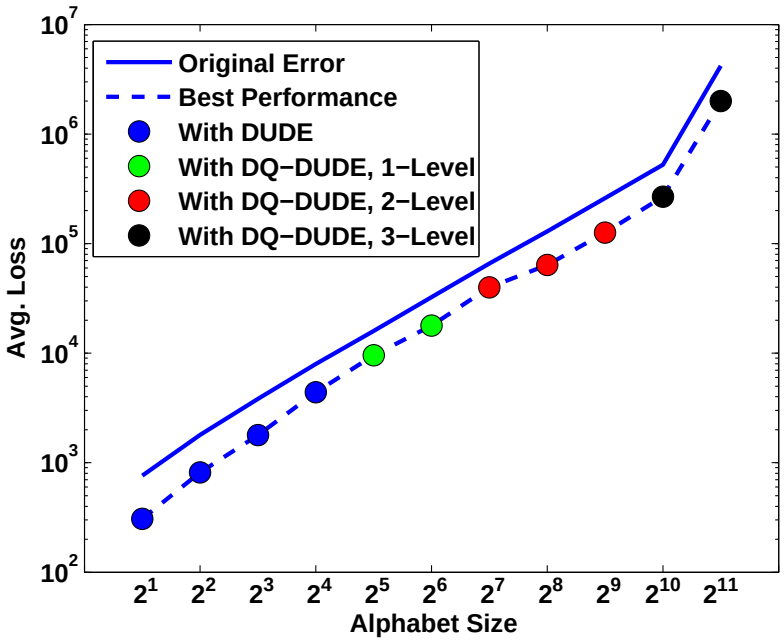


Figure 3.5: Performance of various denoising methods with respect to alphabet size. Results are for sawtooth wave time series.

Table 3.2: Average loss for the offline denoisers.

Time Series Source	Time Series Name	Alphabet Size	Time Series Size	Loss Noisy Time Series	Loss Original DUDE	Loss DQ-DUDE 1-Level	Loss DQ-DUDE 2-Level	Loss DQ-DUDE 3-Level
WUT	Blacksburg	4839	2769	577838.7	577838.7	577838.7	399075.1	96827.9
WUT	Christiansburg	3879	2769	513182.7	513182.7	513182.7	123846.9	54274.5
WUT	Radford	3927	2769	498883.7	498883.7	498883.7	308227.7	66836.5
WUT	Salem	3837	2769	521292.3	521292.3	521292.3	236470.7	57717.5
WUT	Roanoke	4200	2769	446678.2	446678.2	446678.2	445588.2	107066.3
WUT	Organic Food	5329	2769	441120.2	441120.2	441120.2	441120.2	253319.6
WUT	Organic Chemistry	7371	2769	656893.6	656893.6	656893.6	656893.6	333768.2
WUT	Organic Compound	5343	2769	422196.	422196.	422196.	422196.	224040.2
WUT	Organic Farming	11917	2769	1302458.5	1302458.5	1302458.5	1302458.5	1301458.5
WUT	Organic Matter	3095	2769	320662.2	320662.2	320662.2	306116.6	74688.5
PAHO	Argentina	1816	172	15776.3	15776.3	15776.3	15776.3	12874.3
PAHO	Bolivia	504	172	3798.5	3798.5	3798.5	1565.9	-
PAHO	Guatemala	96	172	598.9	598.9	488.5	-	-
PAHO	Honduras	50	172	354.7	354.7	260.5	-	-
PAHO	Panama	60	172	362.3	362.3	312.3	-	-

3.7.2 Real-World Time Series

In this section, we perform experiments with 15 real-world time series. These time series have been collected from the following two different sources:

- PAHO: Influenza Like Illnesses statistics of various countries from the Pan American Health Organization website[1]. We use five time series from this dataset.
- WUT: Wikipedia Usage Trend time series from Wikipedia Trends[2] websites. We use 10 time series from this dataset.

These time series have different alphabet sizes varying between 50 and 11917. In this set of experiments, Q is 10 and quantization is performed on the original time series as well as the quantized time series with larger alphabet. Uniform noise is injected to each time series for 10 times and the average loss for each time series and denoising method is measured. The uniform noise is injected based on Eq. 2.4 with $\epsilon = 0.1$. For the denoising purpose we use both of the offline and online denoisers. For the symmetric offline denoiser we use $k = 1$ while for the asymmetric offline denoiser and online denoisers, k is equal to 1 and δ is set to 0.

Average values of loss for the offline denoisers are shown in Table 3.2. The best achievable performance with respect to alphabet size is illustrated in Fig. 3.6. Furthermore, this figure shows how many levels of the quantization hierarchy (between 1 and 3) was required for each time series to achieve the best improvement in terms of average loss. It can be observed from Fig. 3.6 that there is a logarithmic relationship between the required levels of quantization and the alphabet size. This result is consistent with our previous observations in Section 3.7.1.

[1] `http://ais.paho.org/phip/viz/ed_flu.asp`

[2] www.wikipediatrends.com

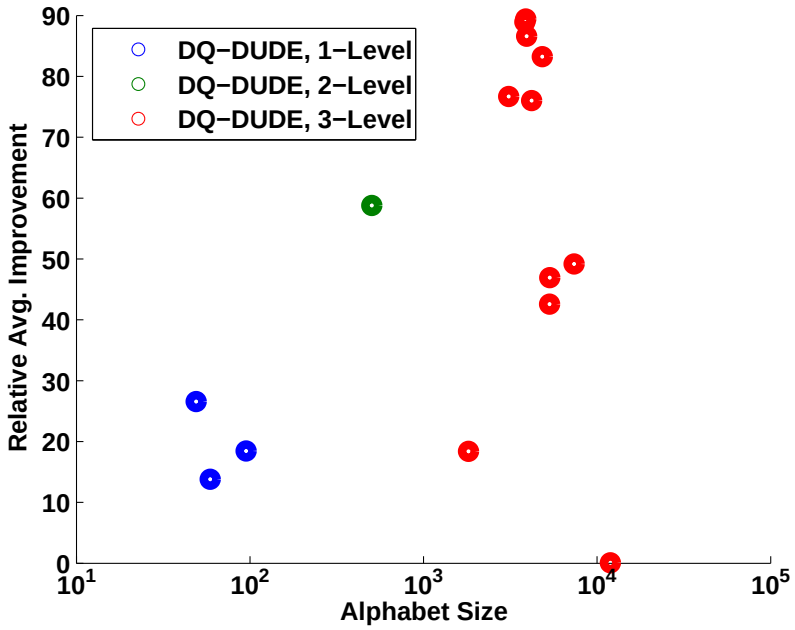


Figure 3.6: Performance of various denoising methods with respect to alphabet size. Results are for the time series in Table 3.2.

Experimental results of the online denoiser, DQ-OOD with three levels of quantization are illustrated in Fig. 3.7. In this figure, average loss of the noisy time series is compared with the loss of the denoised time series when DQ-OOD, symmetric DQ-DUDE, and asymmetric DQ-DUDE are used. This figure shows how he performance of the denoisers change when error probability, ϵ, changes between 0.025 and 0.3. As we can observe, significant improvements can be observed in terms of average loss through the denoising process.

Table 3.3: Running time of various denoising algorithms.

Denoising Method	Running Time
Offline Symmetric DUDE	283.1
Offline Symmetric DQ-DUDE, 1-Level	41.3
Offline Symmetric DQ-DUDE, 2-Level	56.7
Offline Symmetric DQ-DUDE, 3-Level	80.1
Offline Assymetric DUDE	279.1
Offline Assymetric DQ-DUDE, 3-Levels	78.0
Online DQ-OOD, 3-Levels	40.2

Running times of various denoising algorithms for WUT-Blacksburg time series (Table 3.2) are illustrated in Table 3.3. These times are measured with the implementation of the algorithms in MATLAB R2014a over an Intel Core i7 computer with 8 GBytes of RAM and 64-bit

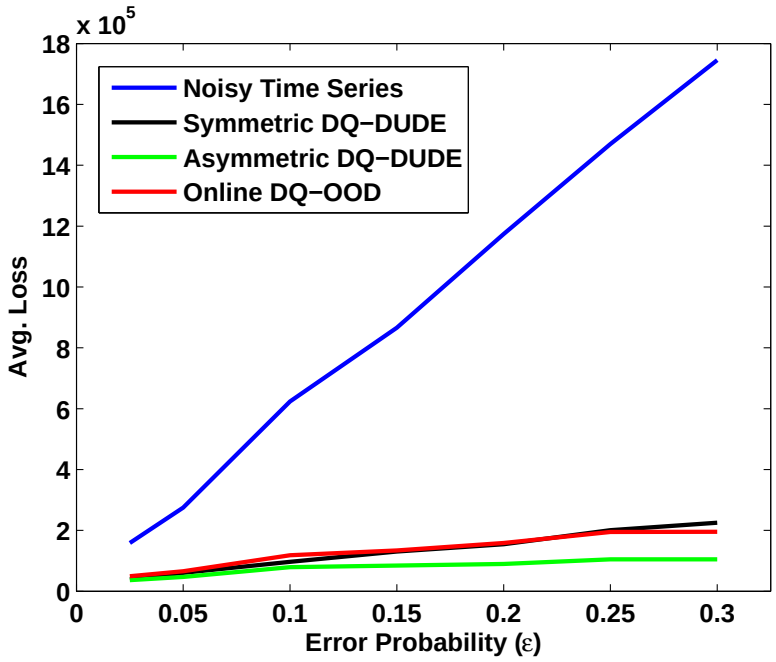


Figure 3.7: Performance of offline and online three-level double quantization denoising methods with respect to error probability. Results are for WUT-Blacksburg time series (Table 3.2).

Windows 10 Home edition. As we expected, running time of the online algorithm DQ-OOD with three levels of quantization is smaller compared to the other methods. Furthermore, quantization helps us to run the denoising process in a shorter time.

3.8 Discussion

In this chapter, we studied the problem of online discrete denoising which is applicable in various real-time data driven applications. First we presented three algorithms for different situations to strike the trade-off between the time-sensitivity and denoising accuracy. We proved that the proposed algorithms asymptotically converge to the optimum offline block denoisers. Furthermore, we provided numerical results for the case of binary data source and BSC channel, which support the theoretical justifications. Then, we studied the problem of discrete universal denoising for time series with large alphabet and proposed the double quantization approach. Various experimental results with synthetic and real-world time series showed that the proposed methods can efficiently improve the time series quality in terms of the average loss. Moreover, we showed that the proposed solutions can perform the denoising task faster and with less memory requirements.

Chapter 4

Time Series Forecasting via Noisy Channel Reversal

Developing a precise understanding of the dynamic behavior of time series is crucial for the success of forecasting techniques. In this chapter, we introduce a novel communication-theoretic framework for modeling and forecasting time series. In particular, the observed time series is modeled as the output of a noisy communication system with the input as the future values of time series. We use a data-driven probabilistic approach to estimate the unknown parameters of the system which in turn is used for forecasting. We also develop an extension of the proposed framework together with a filtering algorithm to account for the noise and heterogeneity in the quality of time series. Experimental results demonstrate the effectiveness of this approach.

4.1 Introduction

With recent advances in data analytics and machine learning, scientists and policy makers are now able to use time-series analysis in important applications such as epidemiological [14] and financial forecasting [58]. Classical time-series methods such as the autoregressive model (AR), ARMA, and ARIMA are widely popular [10]. When the modeled process is highly nonlinear, methods such as Gaussian and Dirichlet processes are more popular [24].

In traditional time series analysis, it is assumed that the behavior of time series in the future will follow the same dynamics as in the past [27]. However, in real-world time series noise and the unfolding dynamic behavior of data can have a significant effect on the accuracy of predictions.

In this chapter, we propose a framework based on the similarities of time series regression and communication systems, to address the aforementioned issues by making explicit assumptions

about the noise and behavior of time series. We will show that noise and deviations from the predetermined behavior of time series is similar to the noise of a communication channel and thus, use communication-theoretic methods to address this deviation to some level. We model the noisy communication channel between the future and past values of time series as an additive Gaussian noise channel. Channel parameters are estimated from the observed data together with an out-of-band filtering algorithm to boost the signal-to-noise (SNR) ratio. The estimates of this noisy channel process are subsequently used for forecasting the future time series data via regression. We perform extensive experiments with real-world and synthetic datasets to study the performance of the proposed approach. Results show that the proposed model can efficiently improve the forecasting performance in terms of accuracy. Our contributions are:

- Developing a framework to directly address the noise, instabilities, and deviations of time series behavior from its known behavior using communication theory concepts.
- Improving the accuracy of regression through a frequency domain out-of-band noise filtering.
- Developing unsupervised analytic models to estimate the amount of noise in a specific time series.

4.2 Problem Formulation

Let us assume that we have a time series $y_1, y_2, \cdots$, which in the rest of this chapter, we denote by Y. We denote the sequence of samples between time points t_1 and t_2 by $\mathbf{Y}_{t_1}^{t_2}$, i.e. $\mathbf{Y}_{t_1}^{t_2} = [y_{t_1} \cdots y_{t_2}]^T$. If we have observed $\mathbf{Y}_1^{t-k}$ and we desire to forecast the value of Y at time t, y_t, we have:

$$\hat{y}_t = \varphi\left(\mathbf{Y}_1^{t-k}\right) \tag{4.1}$$

where $\varphi(\cdot)$ is the estimation function and $\hat{y}_t$ is the estimated value of y_t. As an example, in the classic AR(m) approach, y_t is forecasted based on the following equation:

$$\hat{y}_t = \beta + \sum_{j=0}^{m-1} \alpha_j y_{t-k-j} = \beta + \mathbf{A}^T \mathbf{Y}_{t-k-(m-1)}^{t-k} \tag{4.2}$$

where m is the order of autoregressive model, α_j's and β are the regression weights, and $\mathbf{A} = [\alpha_{m-1} \cdots \alpha_0]^T$.

In many practical applications, due to inaccuracies in data acquisition, forecasts should be made using unstable and noisy time series. Most of the times, this is more serious for the newer values since older samples may be corrected and stabilized over time. This non-uniform imperfection of the data causes heterogeneity in the quality of time series. Regression

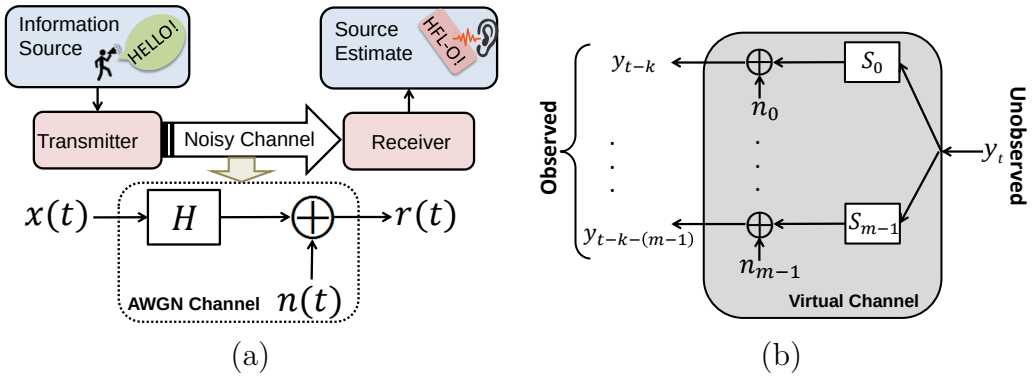


Figure 4.1: (a) General model of a communication channel, (b) Multiple-output virtual channel model.

is highly sensitive to outliers [66] and hence, this noise is considered as a source of estimation error. To model heterogeneity, we use $\dot{y}_j$ to denote a stable sample at time j and $\dot{\mathbf{Y}}_{t_1}^{t_2}$ to denote all the stable samples between t_1 and t_2. Each sample of time series Y can be modeled as follows:

$$y_j = \dot{y}_j + \nu_j \tag{4.3}$$

where $\dot{y}_j$ is the stable part of the sample and ν_j is a Gaussian random variable with zero mean and variance ς_j^2, $\nu_j \sim \mathcal{N}(0, \varsigma_j^2)$. Since newer values suffer from more instability, we can assume that if $i < j$ then $\varsigma_i^2 \leq \varsigma_j^2$. Note that stable data samples are modeled as in Eq. 4.3 with $\varsigma_j = 0$.

Similar to time series analysis, communication systems suffer from imperfect noisy channels and hence, the received message may be different from the sent one. The general model of a communication system is illustrated in Fig. 4.1(a). In this model, a transmitter sends a signal containing some information which is unknown to the receiver. On the other side of the channel, a receiver receives a signal and is aimed to determine the transmitted information. One of the most studied channel models in communication theory is the additive white Gaussian noise (AWGN) channel which is illustrated in Fig. 4.1(a). In this model, the transmitted signal $x(t)$ is corrupted by an additive noise, $n(t)$, which is white and comes from a Gaussian random process with zero mean, $n \sim N(0, \sigma_0^2)$. In Fig. 4.1(a), H is the impulse response of the channel and in its simplest form can be modeled as an attenuation. Generally, the transmitter and receiver are situated in different locations and transmission occurs before the reception.

Let us assume that a transmitter is located in a *future* time, t_F, and is sending us a message, M_{t_F}. We, as the receivers of the message, are located at the current time, $t_C < t_F$, and receive a corrupted and noisy signal, S_{t_C}. Then, the problem of communication is to estimate the message through the received signal as $\hat{M}_{t_F} = g(S_{t_C})$, where $\hat{M}_{t_F}$ is the estimated message and $g(\cdot)$ is the estimation function. Here, since M_{t_F} is generated in the *future* and S_{t_C} is

received at the present time, estimating the unknown message is a prediction problem. By mapping the t_F to t, t_C to $t-k$, M_{t_F} to y_t, and S_{t_C} to $\mathbf{Y}_1^{t-k}$, we can observe that the above communication system is a regression problem, defined in Eq. 4.1. More precisely, we can assume that a transmitter is *located* at time t and sends us an unknown message y_t that we received it in the form of $\mathbf{Y}_1^{t-k}$. Then, the regression problem is to estimate the transmitted message, y_t, based on the received signal. We define the noisy channel forecasting model as follows:

Definition 4. ***Noisy Channel Forecasting (NCF) Model*** *is a representation of a time series forecasting problem as a noisy communication system that assumes that the future values of the time series are unknown messages that have been sent back in time through a noisy channel reversal and received in the form of our observations.*

The NCF model aids in a better understanding of the underling dynamic behavior of time series by leveraging tools and techniques from the communication domain. It is reminiscent of the work of Chen et al. [15] which addresses the application of communication theory and information theory through the use of equalizers to optimize information transfer and analyze causal relationships. These authors also proposed an information coupling approach for dimensionality reduction.

Definition 5. ***NCF model with additive Gaussian noise (AGN)*** *is an NCF that models the channel as an additive Gaussian noise as* $\mathbf{Y}_1^{t-k} = \mathbf{h}(y_t) + \mathbf{N}$*, where* $\mathbf{h}(\cdot)$ *is a vector function, and* $\mathbf{N}$ *is a multivariate Gaussian noise.*

The NCF model can be used as a wrapper for any time series analysis approach. While various channel models can be used in NCF, in this chapter we use NCF with additive Gaussian noise channels. Let us assume that we use the general forecasting function of Eq. 4.1 and the NCF model with AGN. Then, we have the following two equations that relates observed data with future values:

$$\begin{cases} \hat{y}_t = \varphi(\mathbf{Y}_1^{t-k}) \\ \mathbf{Y}_1^{t-k} = \mathbf{h}(y_t) + \mathbf{N} \end{cases} \tag{4.4}$$

Using the training dataset, optimum model parameters including noise, channel, and estimation function coefficients, can be estimated through an optimization problem that minimizes some sort of risk function. For instance, with a mean square error (MSE) risk function, the following optimization problem can be solved to find the system parameters:

$$\min \left(y_t - \hat{y}_t\right)^2 = \min \left(y_t - \varphi\left(\mathbf{h}(y_t) + \mathbf{N}\right)\right)^2. \tag{4.5}$$

Here, we model linear regression problems using NCF models with AGN in order to tackle with noisy time series and reduce the effect of noise in the forecasting process to improve the accuracy of predictions. The provided framework can be also extended to nonlinear regression algorithms.

4.3 Noisy Channel Reversal Regression

In this section, we first model an autoregression problem using NCF with AGN. Then, we estimate noise and channel statistics simultaneously. Results of this estimation can be used to estimate the amount of noise in a specific time series forecasting problem.

4.3.1 Modeling time series as a Noisy Channel

Based on Definitions 4 and 5, in this section we model the AR(m) of Eq. 4.2 using NCF. In the general case, as depicted in Fig. 4.1(b), the system is modeled as a multiple-output virtual communication channel. Hence, based on Definition 5, we can formulate this channel as follows:

$$\mathbf{Y}_{t-k-(m-1)}^{t-k} = y_t\mathbf{S} + \mathbf{N} \tag{4.6}$$

where $\mathbf{S} = [s_{m-1} \cdots s_0]^T$, $s_j \geq 0$, represents the channel effect (attenuation) and $\mathbf{N} = [n_{m-1} \cdots n_0]^T$. Similar to Eq. 4.5 and using Eq. 4.2, the coefficients α_i's and β, can be obtained by solving the following optimization problem:

$$\begin{aligned} \mathbf{A}^*, \beta^* &= \arg\min_{\mathbf{A},\beta} E\left[(y_t - \hat{y}_t)^2\right] \\ &= \arg\min_{\mathbf{A},\beta} E\left[\left(\beta + \mathbf{A}^T\mathbf{N} + y_t\left(\mathbf{A}^T\mathbf{S} - 1\right)\right)^2\right] \end{aligned} \tag{4.7}$$

By the simplifying assumption that Y is a set of i.i.d. random variables, we can show that Eq. 4.7 depends on the mean, μ_Y, and variance, σ_Y^2, of the time series Y. In other words,

$$\begin{aligned} \mathbf{A}^*, \beta^* = \arg\min_{\mathbf{A},\beta} \left(\beta + \mathbf{A}^T\mathbf{N}\right)^2 + \\ \left(\mathbf{A}^T\mathbf{S} - 1\right)^2\left(\sigma_Y^2 + \mu_Y^2\right) + 2\left(\beta + \mathbf{A}^T\mathbf{N}\right)\left(\mathbf{A}^T\mathbf{S} - 1\right)\mu_Y \end{aligned} \tag{4.8}$$

In the special case when data is available uptil y_{t-1} and we model AR(1) with NCF, the optimum weights are as follows:

$$\alpha_0^* = \frac{s_0\sigma^2}{s_0^2\sigma^2 + \sigma_0^2}\,;\ \beta^* = \frac{\mu\sigma_0^2 - s_0\mu_0\sigma^2}{s_0^2\sigma^2 + \sigma_0^2} \tag{4.9}$$

where $s_0 \geq 0$ is channel attenuation and n_0 is an additive Gaussian noise, $n_0 \sim N(\mu_0, \sigma_0^2)$. Furthermore, μ and σ^2 are the expectation and variance of the time series, respectively. The following lemma provides a lower bound for noise and signal parameters in the case of AR(1) with NCF model.

Lemma 2. *Considering the noise in the NCF model of AR(1) improves the accuracy if* $\frac{\mathrm{R}^2}{1-\mathrm{R}^2} < \frac{s_0^2\sigma^2}{\sigma_0^2}$, *where* $\mathrm{R} \in [-1, +1]$ *is the Pearson's product-moment coefficient of the time series.*

Proof *It can be shown that the expected Mean-Square error of the standard AR(1) is* $e_{MSE} = \sigma^2(1-\mathrm{R}^2)$. *Also, for NCF-based AR(1), error can be estimated using* $e^{NCF}_{MSE} = \frac{\sigma^2\sigma_0^2}{s_0^2\sigma^2+\sigma_0^2}$. *We can show that* $e^{NCF}_{MSE} < e_{MSE}$ *holds, if we have* $\sigma_0^2 < s_0^2\sigma^2\frac{1-\mathrm{R}^2}{\mathrm{R}^2}$ *and with minor manipulations, this leads to* $\frac{\mathrm{R}^2}{1-\mathrm{R}^2} < \frac{s_0^2\sigma^2}{\sigma_0^2}$ *thus proving the lemma.*

Based on Lemma 2, when $\mathrm{R} = 0$, there is no linear relationship between y_t and y_{t-1} and for any value of s_0, σ^2, and σ_0^2, the proposed method works better than the classic approach. Also, with perfect linear relationship between y_t and y_{t-1}, $\mathrm{R}^2 = 1$, it is not possible to achieve lower error using the proposed method.

In order to address instability, based on Eq. 4.3, we can reformulate Eq. 4.6 as follows:

$$\begin{aligned}\mathbf{Y}^{t-k}_{t-k-(m-1)} &= \dot{\mathbf{Y}}^{t-k}_{t-k-(m-1)} + \boldsymbol{\nu}^{t-k}_{t-k-(m-1)} \\ &= y_t\mathbf{S} + \mathbf{N} + \boldsymbol{\nu}^{t-k}_{t-k-(m-1)}\end{aligned} \tag{4.10}$$

where $\boldsymbol{\nu}^{t-k}_{t-k-(m-1)} = [\nu_{t-k-(m-1)} \cdots \nu_{t-k}]^T$. Hence, we have two noise components and by independence assumption and assuming that ν_j is wide sense stationary, these noise components can be aggregated into a single noise factor, $\dot{n}_j$, which is also Gaussian. Then, Eq. 4.7 can be used to determine the regression coefficients if we substitute n_j's with $\dot{n}_j$'s.

4.3.2 Simultaneous Estimation of Noise and Channel

Before calculating α_i's and β, we have to first estimate the channel properties, $\mathbf{S}$ and $\mathbf{N}$. There are various channel estimation methods in the literature [21]. In this research, we develop an integrated maximum likelihood method to simultaneously estimate the noise and channel characteristics.

Let us assume that we have already observed the time series Y till time $t-k$ and an m-output noisy channel (Fig. 4.1(b)) is used in the NCF model. Then, using the maximum likelihood approach, we want to choose our parameters to maximize the probability of observation. In other words,

$$\mathbf{S}^*, \mathbf{M}^*, \boldsymbol{\Sigma}^* = \arg\max_{\mathbf{S},\mathbf{M},\boldsymbol{\Sigma}} \Pr\left[\mathbf{Y}^{t-k}_{m+k} | \mathbf{S}, \mathbf{M}, \boldsymbol{\Sigma}\right] \tag{4.11}$$

where $\mathbf{S} = [s_{m-1} \cdots s_0]^T$ represents channel attenuation, $\mathbf{M} = [\mu_{m-1} \cdots \mu_0]^T$ is a vector that contains noise expectations, and $\boldsymbol{\Sigma}$ is the $m \times m$ covariance matrix of noise. Note that based on the proposed model, in order to learn the system parameters, we need to relate each observed data sample, y_j to the m samples $\mathbf{Y}^{j-k}_{j-k-(m-1)}$. Therefore, in Eq. 4.11, the index of y begins from $m+k$. Assuming that y_j's are i.i.d. and this assumption that noise has an m-dimensional Gaussian distribution, $\mathbf{N} \sim \mathcal{N}(\mathbf{M}, \boldsymbol{\Sigma})$, one can easily observe that Eq. 4.11 is equivalent to maximization of the following equation:

$$L = \prod_{j=m+k}^{t-k} \frac{e^{-\frac{1}{2}(\mathbf{Y}^{j-k}_{j-k-(m-1)} - \mathbf{M}_j)^T \Sigma^{-1} (\mathbf{Y}^{j-k}_{j-k-(m-1)} - \mathbf{M}_j)}}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}|}}$$

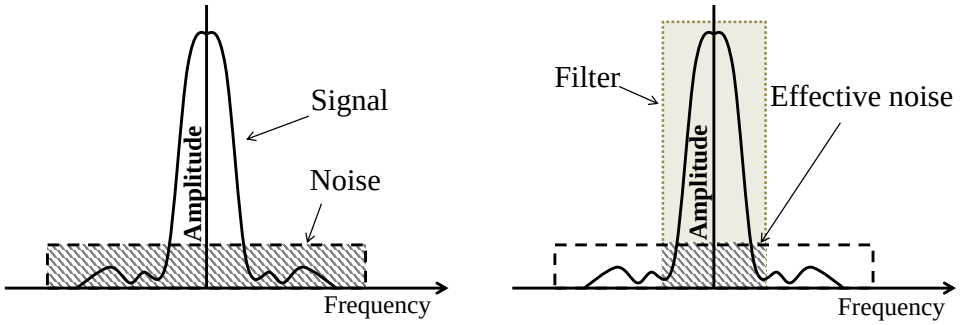


Figure 4.2: (a) Example of a signal and noise in frequency domain. (b) Effect of filtering on signal and noise.

where $\mathbf{M}_j = y_j\mathbf{S} + \mathbf{M}$. Using the log-likelihood, maximization of L is equivalent to the following optimization problem:

$$\mathbf{S}^*, \mathbf{M}^*, \mathbf{\Sigma}^* = \arg\min_{\mathbf{S},\mathbf{M},\mathbf{\Sigma}} (t-m-2k+1)\ln(|\mathbf{\Sigma}|) + \sum_{j=m+k}^{t-k} (\mathbf{Y}_{j-k-(m-1)}^{j-k} - \mathbf{M}_j)^T \mathbf{\Sigma}^{-1} (\mathbf{Y}_{j-k-(m-1)}^{j-k} - \mathbf{M}_j) \tag{4.12}$$

For the single receiver case ($m = 1$) and when data is available uptil y_{t-1} one can show that these parameters can be determined using the following set of equations:

$$\begin{cases} \sigma_0^2 = \frac{1}{t-2}\sum_{j=1}^{t-2}(y_j - s_0 y_{j+1} - \mu_0)^2 \\ s_0 \sum_{j=1}^{t-2} y_{j+1}^2 + \mu_0 \sum_{j=1}^{t-2} y_{j+1} = \sum_{j=1}^{t-2} y_{j+1} y_j \\ s_0 \sum_{j=1}^{t-2} y_{j+1} + (t-2)\mu_0 = \sum_{j=1}^{t-2} y_j \end{cases} \tag{4.13}$$

4.4 Forecasting with Noise Filtering

Usually, real-world time series and their accompanying noise have different representations in the frequency domain. The difference between the noise and signal power distribution in the frequency domain is the key intuition behind using out-of-band noise filtering in order to improve the SNR and increase the prediction accuracy. An illustrative example is shown in Fig. 4.2. In this figure, noise is uniformly distributed over the channel bandwidth while signal power is concentrated around zero. As it is shown in Fig. 4.2(b), by using a low-pass filter with a sufficient bandwidth, a significant power of noise is reduced while the effective part of signal remains unchanged.

The first step in filter design is recognition of important and non-important frequency components of the signal. For this purpose, we use FFT to find the representation of the signal in frequency domain. We define the γ-effective bandwidth of a time series as follows:

Definition 6. ***γ-effective bandwidth:*** *For an arbitrary time series y_t and $\gamma \in [0,1]$, the γ-effective bandwidth is the set of frequency-domain components, symmetrically located around the peak frequency, that carry γP_y of the time series power, where P_y is the total power of time series. Size of the γ-effective bandwidth of $y(t)$ is shown by B_y^γ.*

After representing the time series in frequency domain, we aim to find the γ-effective bandwidth of the signal. Therefore, we need to identify those frequency components that carry the most part of the signal power and to choose an appropriate value for γ. Using a very small value for γ results in an unacceptable distortion in the signal while using larger values results in higher amount of noise. This step can be carried out using cross-validation. Those frequency components that carry γ portion of P_y are in the γ-effective bandwidth of the signal and the remaining ones are considered as out-of-band components. Out-of-band frequency components are forced to zero and then, the filtered version of Y, Y^f, is determined using an inverse-FFT transformation.

While filtering improves the accuracy of regression for *noisy* time series, it can be harmful for *noiseless* ones. In fact, the total energy of a noiseless time series is the signal energy and filtering any number of the frequency components results in unnecessary signal distortions which may reduce the accuracy of the regression.

Lemma 3. *When noise power is uniformly distributed in frequency domain, out-of-band noise filtering with γ-effective bandwidth filter improves time series SNR if $\gamma > B_y^\gamma / B_y^1$.*

Proof *Let us assume that the total power of the time series is $P_{S,N} = P_S + P_N$, where P_S is the signal power and P_N is the power of noise. Obviously, SNR of the original time series is $SNR^{Original} = P_S/P_N$. Similar to Fig. 4.2, the noise power is uniformly distributed over all frequency components. The total bandwidth of the signal contains B_y^1 components in the frequency domain while the filtered time series contains B_y^γ components. Then, after filtering, we will have:*

$$P_N^{Filtered} = \frac{B_y^\gamma}{B_y^1} P_N \quad , \quad P_S^{Filtered} = \gamma P_{S,N} - \frac{B_y^\gamma}{B_y^1} P_N$$

$$SNR^{Filtered} = \frac{\gamma(P_S + P_N) - \frac{B_y^\gamma}{B_y^1} P_N}{\frac{B_y^\gamma}{B_y^1} P_N}.$$

If we aim to improve the SNR we should have $SNR^{Filtered} > SNR^{Original}$. This means that we should have $\gamma > B_y^\gamma / B_y^1$.

4.4.1 Forecasting Framework

The overall view of the proposed framework is illustrated in Fig. 4.3. This framework has three major building blocks:

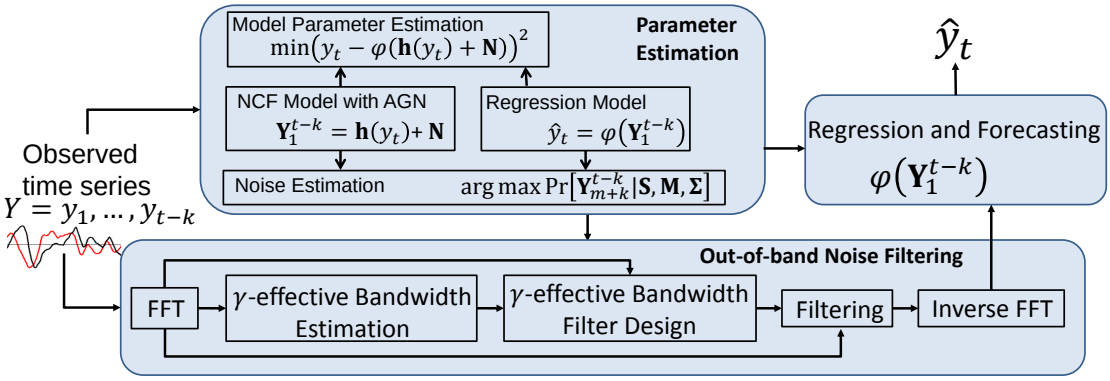


Figure 4.3: Overall proposed forecasting framework.

- *Parameter estimation:* where various parameters including regression coefficients and time series noise are estimated.
- *Out-of-band noise filtering:* where a filter is used to remove the out-of-band noise of the time series.
- *Regression and forecasting:* where using the filtered time series forecasting is performed.

Note that filtering should be performed if time series is noisy. Otherwise, the procedure is likely to result in lower accuracy. The main goal of the parameter estimation step is to estimate the amount of noise in the time series in terms of noise statistics and SNR to be used in the filtering block. After out-of-band noise filtering, one can use the filtered time series for forecasting.

4.5 Case Study: Synthetic Dataset

In order to study the behavior of regression algorithms under out-of-band noise filtering and to determine the accuracy of the estimations performed in previous sections, we performed various experiments on synthetic time series. In this section we provide results and discuss the observations.

To construct the synthetic dataset, we first generate a time series based on $x(t) = 3t + 2$, where t changes between 0 and 100 in steps of 0.1. Then, we construct various time series with added Gaussian noise with zero mean and different variances. For this purpose, we set $y^1 = x$ and

$$y^j = x + n^j \, , \; n^j \sim \mathcal{N}(0, \sigma_j^2) \, , \; \sigma_j = 2^{j-2} \, , \; j = 2, \cdots, 9$$

From each of these time series, we generated 100 instances and for each instance, we first estimate the properties of the noise based on Eq. 4.13. Then, we use AR(1) with and without

out-of-band noise filtering. For AR(1) with filtering, we first performed cross validation to choose the best value of γ. In addition to the regression accuracy, we also measured the signal and noise properties, before and after filtering.

Results of the synthetic time series are shown in Fig. 4.4. Figure 4.4(a) shows the best selected values for γ for each of the time series. As we expected, for smaller values of σ_j (when noise power is low), $\gamma = 1$ is the best. When noise power is low, filtering results in unnecessary signal distortions and this may be harmful for regression accuracy. However, while σ_j increases, the best value of γ decreases. Fig. 4.4(b) compares the original values of σ_j (the ones used in data generation) with the estimated ones, calculated based on Eq. 4.13. Results show that the estimated values are close to real ones. Fig. 4.4(c) shows σ_j of the time series before and after filtering. As we expected, filtering reduces the noise effect dramatically. The effect of filtering on SNR is illustrated in Fig. 4.4(d). Again, results show that filtering improves the SNR dramatically.

Regression accuracy, before and after filtering, are compared in Fig. 4.4(e) and (f). In these figures, RMS error is used as a measure of accuracy. Fig. 4.4(e) shows the regression error when we compare the estimated value with the actual value of the noisy signal, $RMSE_{noisy}$. For each of these time series, we used the last 100 samples for test. Generally, when we aim to test the accuracy of a regression algorithm, we use $RMSE_{noisy}$. However, when the observed time series is noisy and we are using it to predict a noiseless value (such as prediction of the number of people that will have Influenza based on an observed noisy data), we need to compare our predictions with the noiseless time series. We show this error by $RMSE_{noiseless}$ and results are illustrated in Fig. 4.4(f).

Figure 4.4(g) compares the value of γ and the ratio $\frac{B^\gamma}{B^1}$. In Lemma 3, we showed that for an effective filtering, γ should be greater than $\frac{B^\gamma}{B^1}$. Results of Fig. 4.4(g) shows that this condition is always true. To illustrate how the amount of noise in the time series is reduced by filtering, Fig. 4.4(h) shows an example. This figure compares the original and filtered synthetic time series with $\sigma_6 = 16$. It is obvious that filtering has reduced the amplitude of noise in the time series dramatically which results in the improvements in regression accuracy.

4.6 Experimental Results

In this section, to have a better understanding of the performance of the proposed framework, we perform experiments with 40 time series from the following datasets:

- UCI: Real-world time series from UCI Repository.
- NIST: Time series from the NIST StRD website[1] .

[1] `http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`

Table 4.1: Effect of filtering on regression algorithms and time series. Green color shows the improvement in accuracy.

Time series	AR(1)	AR(Opt)	R-AR(1)	R-AR(Opt)	R-ARMA(1,1)	R-ARMA(Opt)
PAHO - Argentina	1.50	4.86	0.00	0.00	5.60	1.22
PAHO - Chile	0.00	2.37	9.19	0.00	15.56	0.25
PAHO - Colombia	0.00	3.83	1.04	5.18	5.21	0.00
PAHO - Costa Rica	8.88	12.00	4.15	4.36	5.76	0.35
PAHO - Ecuador	44.50	37.42	8.83	9.86	12.99	6.03
PAHO - El Salvador	2.35	3.46	8.43	1.89	7.32	2.12
PAHO - Guatemala	5.07	5.07	7.12	1.05	3.36	0.00
PAHO - Honduras	11.22	11.22	11.81	6.20	6.02	0.00
PAHO - Mexico	18.41	21.57	0.00	0.00	0.00	0.00
PAHO - Nicaragua	10.75	10.75	8.03	0.00	2.67	0.00
PAHO - Panama	1.36	1.36	1.10	0.82	7.24	0.27
PAHO - Paraguay	15.88	13.92	0.00	0.00	0.00	0.00
GST - Melanoma	4.17	0.00	8.00	6.00	2.08	2.08
GST - Volleyball	16.67	16.67	3.33	3.33	13.33	0.00
GST - Wrestling	37.04	26.09	5.26	5.26	19.05	10.53
GST - Swimming	9.30	22.50	0.00	0.00	0.00	0.00
GST - Yahoo	35.14	12.50	0.00	0.00	4.55	4.55
GST - Microsoft	66.67	100.00	0.00	0.00	0.00	0.00
GST - Facebook	14.29	15.00	10.53	5.56	5.56	0.00
GST - Youtube	14.29	10.00	10.00	5.26	5.26	0.00
GST - Hotmail	15.79	13.89	3.23	0.00	0.00	0.00
GST - Ebay	18.00	8.89	2.38	2.38	2.38	2.38
GST - Dollar	15.09	6.25	2.17	2.17	8.16	0.00
GST - Yen	26.67	24.14	11.54	4.17	26.67	0.00
GST - Pound	18.60	10.53	0.00	0.00	10.26	15.79
GST - Earthquake	21.19	11.19	4.96	0.00	6.45	0.00
GST - Hurricane	35.17	35.17	36.23	7.78	23.58	0.00
GST - Tsunami	21.92	21.92	18.46	3.51	24.66	0.00
GST - Storm	1.49	1.49	17.65	0.00	0.00	0.00
STOCK - Yahoo	5.13	5.13	5.13	5.13	7.69	7.69
STOCK - Ford	12.50	12.50	12.50	12.50	12.50	12.50
STOCK - GE	14.29	14.29	14.29	14.29	14.29	14.29
STOCK - AEP	11.11	5.88	11.11	5.88	11.11	5.88
NIST StRD - ENSO	3.21	3.31	0.00	0.00	0.00	1.62
NIST StRD - Gauss1	13.03	0.00	15.30	0.00	5.41	0.00
NIST StRD - Gauss2	22.41	10.94	25.87	12.50	10.83	5.96
NIST StRD - Gauss3	21.80	7.91	23.78	9.31	6.36	2.18
UCI - Bike Sharing (Daily)	0.57	0.00	4.40	0.00	1.70	0.00
UCI - CalIT2 In (Daily sum)	2.01	0.00	1.99	0.00	4.84	0.00
UCI - Dodgers (Daily Avg)	0.00	0.00	3.60	0.00	7.64	0.00

- STOCK: Closing stock price of different companies from Yahoo Finance.
- GST: Weekly Google Search Trends from different topics.
- PAHO: Influenza Like Illnesses statistics of various countries from the Pan American Health Organization website[2].

In addition to AR(m), we also perform experiments with recursive-AR(m) (R-AR) and recursive-ARMA(m,q) (R-ARMA). We report results for basic (i.e. $m = q = 1$) and optimal

[2] `http://ais.paho.org/phip/viz/ed_flu.asp`

settings for each algorithm, where optimal settings were determined through cross validation. We also performed experiments with and without filtering. We measured the accuracy using the following relative error (RE):

$$e_{RE} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{|\hat{y}_t - y_t|}{max(y_t, \hat{y})}$$

where $\mathcal{T}$ is the test set (in this experiment, last 15% samples).

Table 4.1 shows the effect of filtering on various algorithms and time series. Numbers depict the achieved improvements in accuracy due to filtering and the green color shows cases where filtering has improved the accuracy. It is obvious from the table that in most of the cases filtering has improved the accuracy of regression. Results also show that on average, AR(1) shows the highest improvement. This is due to the nature of AR(1), as other methods are more complex and less vulnerable to noise. The average error of algorithms before and after filtering over 40 time series are compared in Fig. 4.6.

The effect of filtering with $\gamma = 0.9$ on two example time series is illustrated in Fig. 4.5. The first row of the figure shows the FFT of the time series before and after filtering. This figure also shows that how filtering affects on the quality of signal and amplitude of noise. It is obvious from this figure that the noise amplitude has been reduced dramatically after filtering. The results are consistent with Table 4.1.

4.7 Discussion

In this chapter, we proposed the NCF model to extend forecasting methods and a new framework for time series regression, using a communication theoretic method. In this model, noise plays the role of a hidden factor in the system and we showed that under certain conditions, the proposed model performs better than the classic methods. Experimental results showed that the proposed solution results in higher accuracy than the classic approaches. Future work includes extending the NCF model to nonlinear regressions and expanding it to include other communication-theoretic techniques such as equalizers.

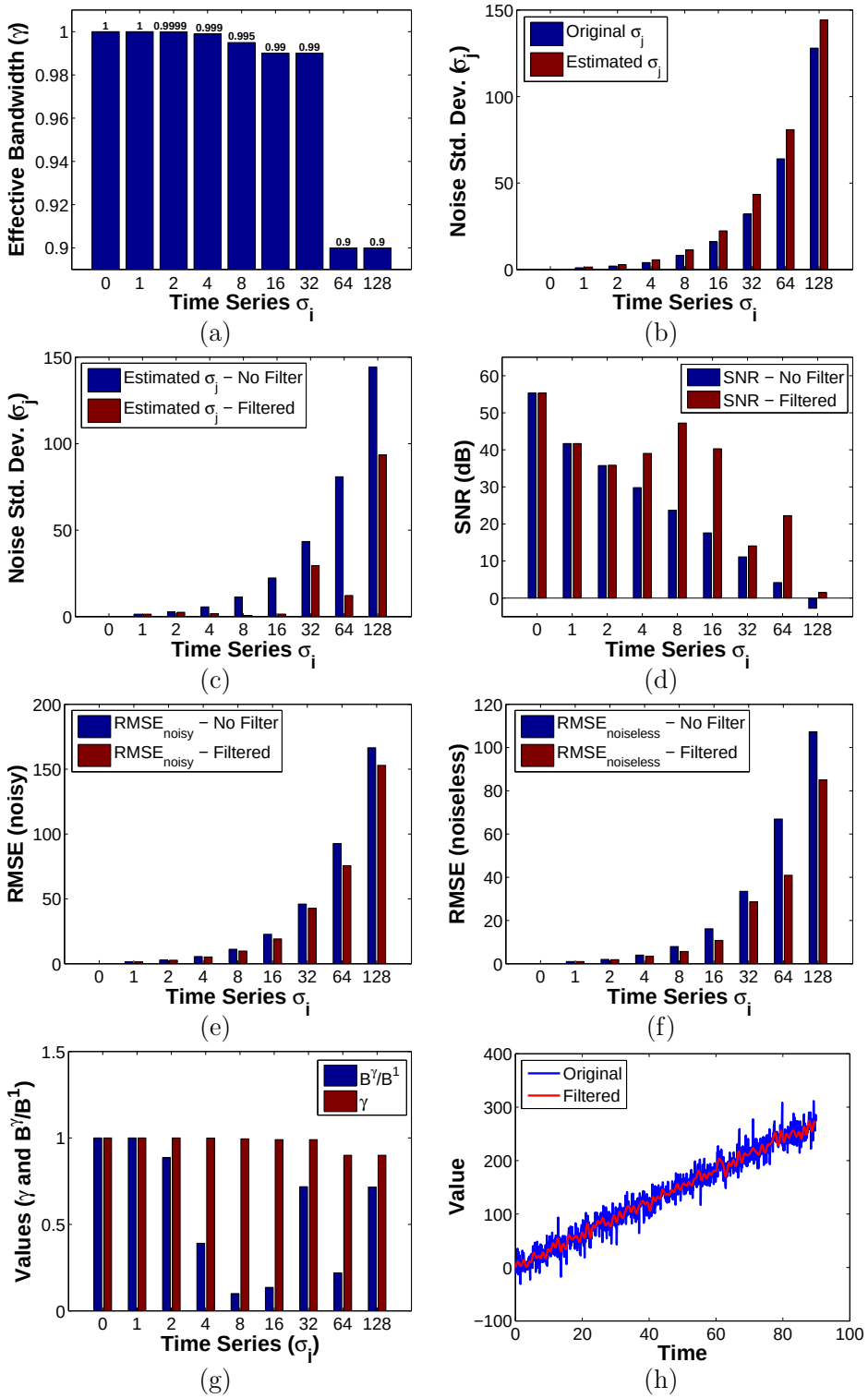


Figure 4.4: Results of synthetic data: (a) selected values for γ, (b) estimated and actual values of σ_j, (c) effect of filtering on σ_j, (d) SNR of time series in dB, (e) RMS error w.r.t noisy data, (f) RMS error w.r.t noiseless data, (g) comparison of γ with $\frac{n_\gamma}{n_F}$ and (h) original and filtered time series with $\sigma_j = 16$.

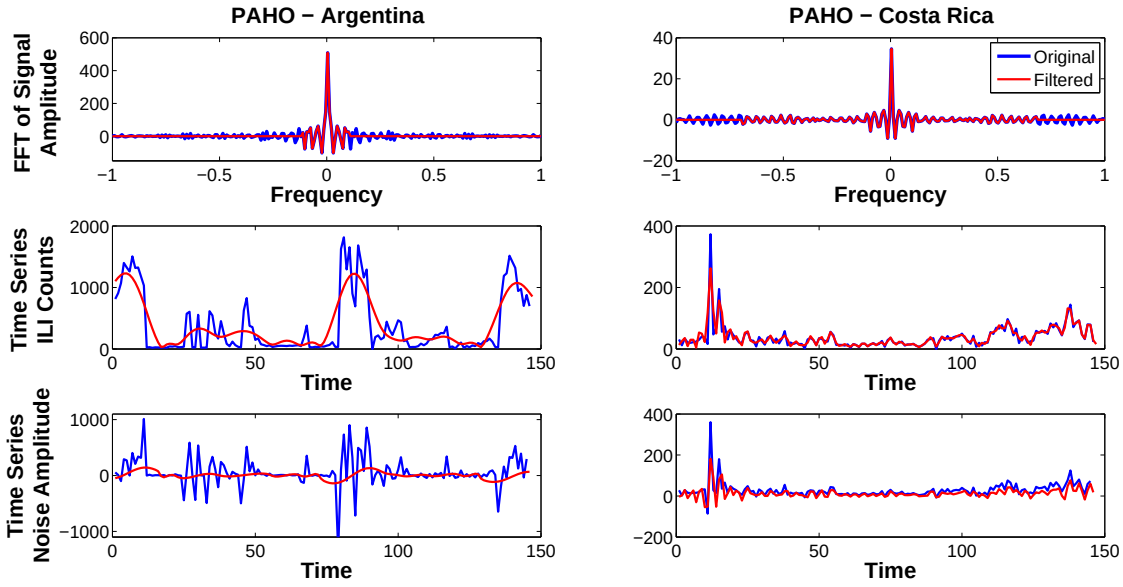


Figure 4.5: The effect of filtering on two time series ($\gamma = 0.9$). The top row shows time series in frequency domain. The middle row shows the time series in time domain. The bottom row shows the estimated effective noise.

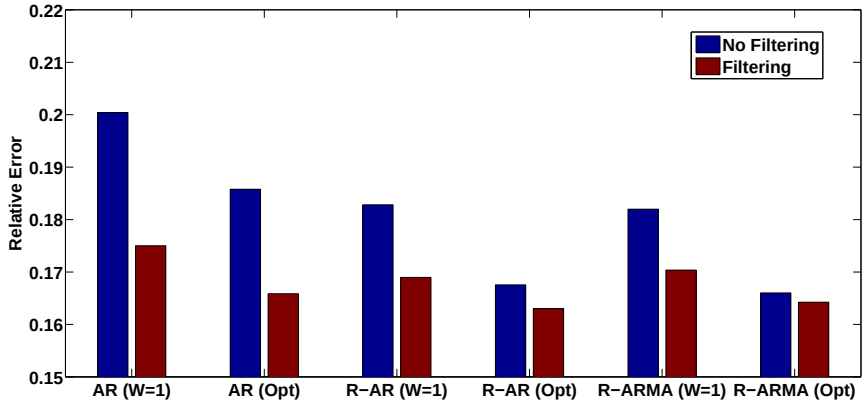


Figure 4.6: Comparison between average error of different regression algorithms before and after filtering over 40 time series.

Chapter 5

Deep Learning based Denoising

Feed-forward deep neural networks have been used extensively in various machine learning applications. In this chapter, we develop a framework for the denoising of time series using feed-forward deep neural networks. For this purpose, we deploy the locality features of time series by using a delayed line. This is similar to the concept of context window that we used in Chapter 3. Neighboring values of the time series are then fed into a feed-forward neural network and through a supervised learning process we determine the appropriate weights of the network for the denoising purpose. The proposed framework is evaluated through multiple experiments using real-world time series.

Developing a precise understanding of the underling behavior of neural networks is crucial for their efficient deployment. To have a better understanding about the dynamic behavior of a denoising feed-forward neural network, we study the behavior of these networks using information theory concepts. For this purpose, we first use an information theoretic approach to study the flow of information in a general feed-forward neural network and to determine how entropy of information changes between consecutive layers. Then, using the Information Bottleneck principle, we develop a constrained optimization problem that addresses the training requirements of a feed-forward neural network. Results of this analysis are used to prove a theoretical bound for the training phase of a denoising deep neural network.

In the rest of this chapter, after a brief literature review, we study the flow of information in a deep neural network from an information theoretic perspective. Then we propose a frame work for deep learning based denoising using feed-forward neural networks and we prove theoretical bounds for denoising neural networks. Experimental results with different time series and noise models are used to evaluate the performance of the developed denoisers and to support the theoretical results of this chapter.

5.1 Background

With the increasing demand for data analytics, Big Data, and artificial intelligence, efficient machine learning algorithms are required now more than anytime before [16]. Deep learning and deep neural networks (DNNs) have been shown to be among the most efficient machine learning paradigms, specifically for supervised learning tasks. Due to their fascinating performance, different deep learning structures have been deployed in various applications in the past decades [16, 67, 42]. However, despite of their great performance, more theoretical effort is required to understand the dynamic behavior of DNNs both from learning and design perspectives.

Deep neural networks are considered as multi-layer structures, constructed by simple processing units known as neurons that process the input information to generate a desired output [47, 60]. These structures have been used previously in a variety of applications, such as dimensionality reduction [32], face representation [67], robotic grasps detection [42], and object detection [68].

Autoencoders and deep neural networks have been previously used for image denoising applications [43, 75, 72]. Traditionally, autoencoders have been used for feature selection and dimensionality reduction purposes. In [72], the authors proposed an autoencoder based deep learning solution for denoising of data for classification tasks. Unlike the previously proposed solutions in Chapters 3 and 4, the method of [72] works based on a supervised learning approach and needs to have access to the noiseless data samples. The general architecture of the fundamental approach proposed in [72] is illustrated in Figure 5.1. In this architecture, noise is injected to the noiseless data samples, $\mathbf{x}$, to construct a noisy dataset, $\tilde{\mathbf{x}}$. Mapping functions f_θ and $g_{\theta'}$ are used to perform autoencoding and autodecoding tasks which result in reconstructed data samples, $\mathbf{z}$. The ultimate goal is to minimize a loss function, $L_H(\mathbf{x}, \mathbf{z})$. In this model, mapping functions, f_θ and $g_{\theta'}$, are implemented using deep neural networks. The appropriate loss function, $L_H(\mathbf{x}, \mathbf{z})$, may be selected based on the application and the noise model. In [72], cross-entropy, $\mathbf{H}(X|Z)$, and squared error, $||\mathbf{x} - \mathbf{z}||^2$, loss functions are used for different cases. Note that cross-entropy loss function is minimized when the mutual information between $\mathbf{x}$ and $\mathbf{z}$ is maximized as we have

$$\mathbf{I}(X||Z) = \mathbf{H}(X) - \mathbf{H}(X|Z).$$

5.2 Information Flow in DNNs

While DNNs have shown their capability in solving machine learning problems, they have been traditionally deployed in a heuristic manner [71, 8]. However, to be able to use these structures more efficiently, we need to have a deeper understanding of their underling dynamic behavior [8].

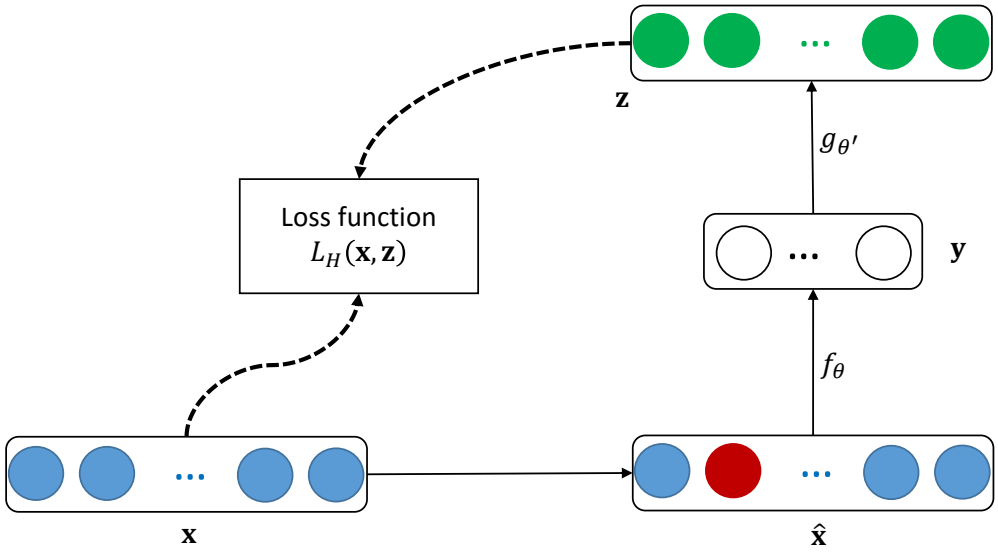


Figure 5.1: The denoising autoencoder architecture of [72].

Mehta and Schwab showed in [45] that deep learning is related to renormalization groups in theoretical physics and provide a mapping between deep learning methods and variational renormalization groups using Restricted Boltzmann Machines. In [71], Tishby and Zaslavsky proposed a theoretical framework, based on the principle of Information Bottleneck [70], to analyze the DNNs where, the ultimate goal of deep learning has been formulated as a trade-off between compression and prediction. In [71], the authors claim that an optimal point exists on the compression-prediction plane that can efficiently address that trade-off. Moreover, they suggest that an Information Bottleneck based learning algorithm may achieve the optimal information representation.

In this section, we analyze the flow of information in a deep neural network using an information theoretic approach. While different structures have been developed for DNNs, we consider the multi-layer feed-forward structure and we assume that the network is used in a supervised setting. We determine an upper bound on the total compression rate in a neural network that can be achieved with an acceptable level of distortion. Furthermore, using the fundamental concepts of Information Bottleneck and based on the approach of Tishby and Zaslavsky in [71], we develop an optimization problem that can be used in the learning process of DNNs.

5.2.1 Flow of Entropy in Deep Neural Networks

The typical structure of a feed-forward DNN is illustrated in Fig 5.2. In this figure, input layer is represented by X and the representation of data in the i^{th} hidden layer is shown by X_i. We assume that the network has ν layers and the output layer, i.e. X_ν, should

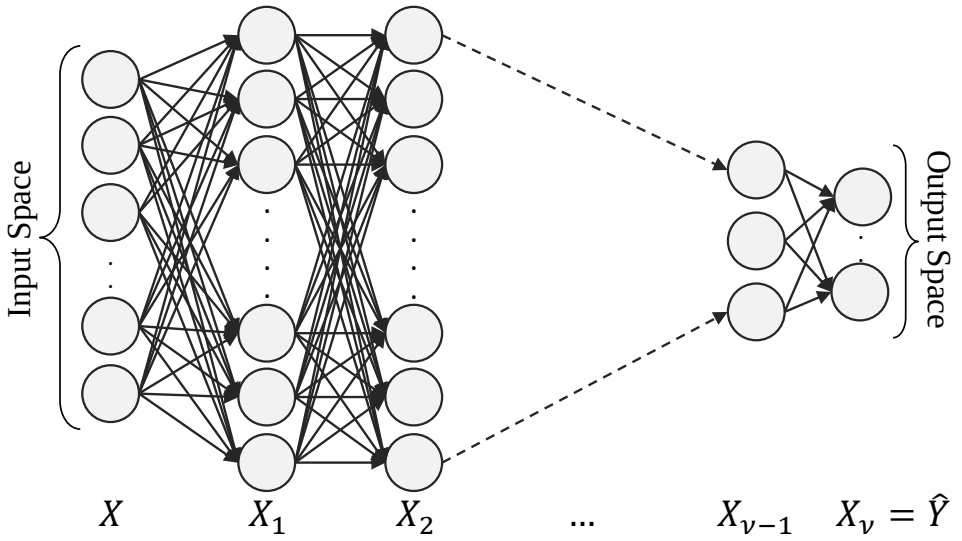


Figure 5.2: General structure of a feed-forward deep neural network.

estimate a desired output, Y. Each layer is constructed by multiple neurons that process the information in parallel and the output of the j^{th} neuron in layer n is calculated as follows:

$$x_{n,j} = g_j^n(x_{n-1}) = f^n\left(\sum_{i=1}^{m_{n-1}} w_{i,j}^n x_{n-1,i} + b_j^n\right) \tag{5.1}$$

where m_{n-1} is the number of neurons in layer $n-1$ and $w_{i,j}^n$ is a weight that connects the output of the i^{th} neuron in layer $n-1$ to the input of the j^{th} neuron in layer n. Also, b_j^n is the bias of the j^{th} neuron in layer n and $f^n(\cdot)$ is the output function of the neurons in this layer. The general structure of a single neuron is illustrated in Fig. 5.3. To illustrate (5.1) in vector notation we say:

$$\mathbf{x}_n = G^n(\mathbf{x}_{n-1}) \tag{5.2}$$

where $\mathbf{x}_n$ is a combination of neuron outputs in layer n, i.e. $\mathbf{x^n} = [x_{n,1}, \cdots, x_{n,m_n}]$. The total number of possible output combinations in the n^{th} layer is illustrated by k_n which depends on the output functions and the input space. As an example, with binary output functions, $k_n = 2^{m_n}$. At the n^{th} layer, $\mathcal{X}_n = \{\mathbf{x_1^n}, \cdots, \mathbf{x_{k_n}^n}\}$ is the set of all possible output combinations. It should be noted that a single neuron has a very limited processing capability. As a matter of fact, an individual neuron can only implement a hyper-plane in its input space and hence, not all the mappings from its input to its output are feasible. This limitation is one of the main reasons that neural networks are constructed by multiple layers of interacting neurons.

The neural computation that occurs in each layer performs a mapping between the outputs of the consecutive layers. In other words, various output combinations in layer $n-1$ are mapped to certain output combinations in layer n. Depending on the weights, bias, and the output function in layer n, there are two possibilities:

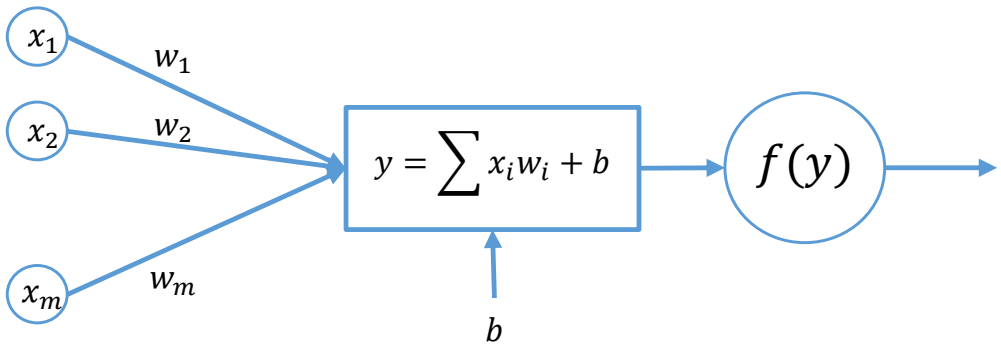


Figure 5.3: General structure of a single neuron.

- Each unique combination of neuron outputs in layer $n-1$ is uniquely mapped to a combination of neuron outputs in layer n. In other words:

$$\forall \mathbf{x_1}, \mathbf{x_2} \in \mathcal{X}_{n-1} \quad ; \quad G^n(\mathbf{x_1}) \neq G^n(\mathbf{x_2})$$

  In this case, regardless of the number of neurons in each layer, we have $k_n = k_{n-1}$.

- Multiple (at least two) combinations of neuron outputs in layer $n-1$ are mapped to a single combination of neuron outputs in layer n. In other words:

$$\exists \mathbf{x_1}, \mathbf{x_2} \in \mathcal{X}_{n-1} \quad ; \quad G^n(\mathbf{x_1}) = G^n(\mathbf{x_1})$$

  and in this case, we have $k_n < k_{n-1}$.

It is worth mentioning that the mapping in each layer of DNN is also a partitioning of the layer's input space and hence, what a DNN does is multiple consecutive of partitioning and mapping processes with the goal of estimating a desired output.

Definition 7. *In the n^{th} layer of a feed-forward DNN, layer partition is the partitioning of the n^{th} layer's input space which occurs due to the neural computations performed in layer n. We illustrate this partitioning by $\mathcal{S}_n = \{S_1^n, \cdots, S_{k_n}^n\}$ where, S_j^n is the set of all the output combinations in layer $n-1$ that are mapped to the j^{th} output combination in layer n, i.e. $\mathbf{x_j^n}$. In other words,*

$$S_j^n = \{\, \mathbf{x_i^{n-1}} \in \mathcal{X}_{n-1} \mid \quad \mathbf{x_j^n} = G^n(\mathbf{x_i^{n-1}}) \,\}$$

Note that we have:

$$i \neq j \Rightarrow S_i^n \cap S_j^n = \emptyset \quad \textit{and} \quad \bigcup_{j=1}^{k_n} S_j^n = \mathcal{X}_{n-1}.$$

Let us assume that $P_n(\mathbf{x_j^n})$ is the probability of $\mathbf{x_j^n}$. Then, it can be observed that

$$P_n(\mathbf{x_j^n}) = \sum_{\mathbf{x'} \in S_j^n} P_{n-1}(\mathbf{x'}) \tag{5.3}$$

Furthermore, considering this fact that $\mathcal{S}_n$ is a partitioning of $\mathcal{X}_{n-1}$, one can easily show that $P_n(\mathbf{x_j^n})$ is the probability of partition S_j^n. It can be shown that

$$\mathbf{\Pi}_j^n = \left\{ \forall \mathbf{x} \in S_j^n \mid \frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}' \in S_j^n} P_{n-1}(\mathbf{x}')} \right\} \tag{5.4}$$

is the probability distribution of all the combinations in S_j^n.

Definition 8. *In a feed-forward DNN, the entropy of layer n, $H(X_n)$, is the entropy of the neuron outputs at this layer and we have:*

$$H(X_n) = -\sum_{i=1}^{k_n} P_n(\mathbf{x_i^n}) \log P_n(\mathbf{x_i^n}).$$

Definition 9. *The entropy of partition S_j^n, $H(S_j^n)$, is the entropy of the output combinations that belong to S_j^n. In other words:*

$$H(S_j^n) = -\sum_{\mathbf{x} \in S_j^n} \frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}' \in S_j^n} P_{n-1}(\mathbf{x}')} \log \frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}' \in S_j^n} P_{n-1}(\mathbf{x}')}$$

In the following lemma we show how entropy of information changes in a feed-forward neural network.

Lemma 4. *In a feed-forward DNN, the entropy of information that flows from layer $n-1$ to layer n is decreased by the expected entropy of the partitions on the possible output combinations in layer $n-1$. The amount of this reduction is shown by Δ_n and we have*

$$\Delta_n = H(X_{n-1}) - H(X_n) = \sum_{j=1}^{k_n} P_n(\mathbf{x_j^n}) H(S_j^n). \tag{5.5}$$

Proof. Let us assume that the information has been processed up to layer X_{n-1}. Using Definition 8, the entropy of layer $n-1$ is

$$H(X_{n-1}) = -\sum_{i=1}^{k_{n-1}} P_{n-1}(\mathbf{x_i^{n-1}}) \log P_{n-1}(\mathbf{x_i^{n-1}}) \tag{5.6}$$

To determine $H(X_n)$, we can say

$$H(X_n) = -\left\{\sum_{i=1}^{k_{n-1}} P_{n-1}(\mathbf{x_i^{n-1}}) \log P_{n-1}(\mathbf{x_i^{n-1}}) + \sum_{j=1}^{k_n}\left[\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right)\log\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right) - \left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\log P_{n-1}(\mathbf{x})\right)\right]\right\} \tag{5.7}$$

In other words, we started from the entropy of X_{n-1} and substituted the individual terms of all the output combinations that belong to S_j^n with their new equivalent term, i.e. $\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right)\log\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right)$. On the other hand, we have:

$$\begin{aligned}&\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right)\log\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\right)\\ &-\left(\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\log P_{n-1}(\mathbf{x})\right)\\ &= -\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\log\frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}'\in S_j^n} P_{n-1}(\mathbf{x}')}\end{aligned}$$

then, considering (5.6), we can rewrite (5.7) as follows:

$$H(X_n) = H(X_{n-1}) + \sum_{j=1}^{k_n}\sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x})\log\frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}'\in S_j^n} P_{n-1}(\mathbf{x}')} \tag{5.8}$$

Equivalently, we have

$$H(X_n) = H(X_{n-1}) + \sum_{j=1}^{k_n}\left(\sum_{\mathbf{x}'\in S_j^n} P_{n-1}(\mathbf{x}')\right)\left(\sum_{\mathbf{x}\in S_j^n}\frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}'\in S_j^n} P_{n-1}(\mathbf{x}')}\log\frac{P_{n-1}(\mathbf{x})}{\sum_{\mathbf{x}'\in S_j^n} P_{n-1}(\mathbf{x}')}\right) \tag{5.9}$$

Then, based on Definition 9 we can say:

$$H(X_n) = H(X_{n-1}) - \sum_{j=1}^{k_n} P_n(\mathbf{x_j^n}) H(S_j^n) \tag{5.10}$$

and from here, we can observe that $H(X_n) \leq H(X_{n-1})$. Furthermore, the difference between the entropy in layers $n-1$ and n is $\sum_{j=1}^{k_n} P_n(\mathbf{x_j^n}) H(S_j^n)$ which is the expected entropy of the partitions on $\mathcal{X}_{n-1}$. This proves the lemma. □

The following lemma proves a similar result for the flow of conditional entropy in a deep neural network.

Lemma 5. *In a feed-forward DNN, the conditional entropy of each layer, conditioned on the desired outputs, Y, i.e. $H(X_n|Y)$, is a non-increasing function of n and we have:*

$$\begin{aligned} \Delta'_n &= H(X_{n-1}|Y) - H(X_n|Y) \\ &= \sum_{y\in\mathcal{Y}} \sum_{j=1}^{k_n} P_Y(y) P_n(\mathbf{x_j^n}|Y=y) H(S_j^n|y) \end{aligned} \tag{5.11}$$

where

$$P_n(\mathbf{x_j^n}|Y=y) = \sum_{\mathbf{x}\in S_j^n} P_{n-1}(\mathbf{x}|Y=y)$$

and

$$H(S_j^n|y) = -\sum_{\mathbf{x}\in S_j^n} \frac{P_{n-1}(\mathbf{x}|Y=y)}{P_n(\mathbf{x_j^n}|Y=y)} \log \frac{P_{n-1}(\mathbf{x}|Y=y)}{P_n(\mathbf{x_j^n}|Y=y)}.$$

Proof. The proof of Lemma 5 follows on similar lines as Lemma 4 by conditioning on the random variable Y and is therefore omitted. □

5.2.2 Optimal Mapping and Information Bottleneck

Extraction of relevant information from an input data with respect to a desired output is one of the central issues in supervised learning [20]. In information theoretic terms, assuming that X and Y are the input and output random variables, $I(X;Y)$ is the relevant information between X and Y. In order to improve the efficiency of a machine learning task, we generally need to extract the minimal sufficient statistics of X with respect to Y. Hence, as the Information Bottleneck principle [70] indicates, we need to find a maximally compressed representation of X by extracting the relevant information with respect to Y [71]. In this section, we follow the approach of [71] to define an optimization problem that can be used in the learning process of a DNN. We also prove an upper bound on the achievable compression rate of the input in DNNs.

Consider a DNN with ν layers. Let us assume that $\hat{X} = [X_1, \cdots, X_\nu]$ denote the output of these layers. To find the maximally compressed representation of X, we need to minimize the mutual information between the input, i.e. X, and the representation of data by the

DNN, $\hat{X}$. This can be formulated as minimizing the mutual information between the input and the representation of data in each layer, i.e. $X_i, i = 1, \cdots, \nu$ which can be modeled as

$$\min \sum_{i=1}^{\nu} I(X; X_i). \tag{5.12}$$

To measure the fidelity of the training process of a feed-forward DNN, we focus on the Logarithmic-loss distortion function. Let $p(y|x)$ denote the conditional distribution of Y given X (i.e., the original input data) and similarly, let $p(y|\hat{x})$ denote the conditional distribution of Y given $\hat{X}$ (i.e., given the outputs of all the layers). Then, one measure of fidelity is the KL-divergence between these distributions:

$$d_{IB}(x, \hat{x}) = \sum_{y} P(y|x) \log \frac{P(y|x)}{P(y|\hat{x})} \tag{5.13}$$

and taking the expectation of $d_{IB}(x, \hat{x})$ we get:

$$D_{IB} = E\left[d_{IB}(x, \hat{x})\right] = I(X; Y|\hat{X}) \tag{5.14}$$

Using the Markov chain properties of the feed-forward network, one can show that

$$I(X; Y|\hat{X}) = I(X; Y \mid [X_1, \cdots, X_\nu]) = I(X; Y|X_\nu) \tag{5.15}$$

and hence, the overall distortion of the DNN will be $D_{IB} = I(X; Y|X_\nu)$. Therefore, using the Information Bottleneck principle, the training criteria for a feed-forward DNN can be formulated as follows

$$\begin{aligned} &\min \sum_{i=1}^{\nu} I(X; X_i) \\ &s.t. \quad I(X; Y|X_\nu) \leq \epsilon \end{aligned} \tag{5.16}$$

As we have mentioned in Section 5.2.1, in the i^{th} layer, each input combination is mapped to a specific output combination. Therefore, it can be easily shown that $H(X_i|X) = 0$ and hence, $I(X; X_i) = H(X_i)$. Then, using Lemma 4 we have:

$$I(X; X_i) = H(X) - \sum_{j=1}^{i} \Delta_j. \tag{5.17}$$

Note that in the above equation, $H(X)$ is constant, i.e. it does not depend on the neural network settings.

Regarding the constraint of (5.16), it can be shown that

$$I(X; Y|X_\nu) = I(X; Y) - I(X_\nu; Y) \tag{5.18}$$

On the other hand, we know that $I(X_\nu;Y) = H(X_\nu) - H(X_\nu|Y)$ and using Lemma 4 and Lemma 5, we have

$$\begin{cases} H(X_\nu) = H(X) - \sum_{i=1}^{\nu} \Delta_i \\ H(X_\nu|Y) = H(X|Y) - \sum_{i=1}^{\nu} \Delta'_i \end{cases}$$

which results in the following equation:

$$I(X;Y|X_\nu) = \sum_{i=1}^{\nu} (\Delta_i - \Delta'_i)\,. \tag{5.19}$$

Using (5.17) and (5.19) and by minor manipulation, the optimization problem of (5.16) can be rewritten as follows:

$$\begin{aligned} &\max \sum_{i=0}^{\nu-1} (\nu - i)\Delta_{i+1} \\ s.t. \quad &\sum_{i=1}^{\nu} (\Delta_i - \Delta'_i) \le \epsilon \end{aligned} \tag{5.20}$$

This is a convex optimization problem w.r.t. Δ_i's and Δ'_i's and due to its complexity, it is generally difficult to find an analytic solution for that. However, numerical solutions (such as algorithms based on Blahut-Arimoto [9]) may be deployed here to solve (5.20).

In the optimization problem of (5.20), Δ_i is the amount of entropy reduction in layer i which can be interpreted as the amount of data compression that has been occurred at this layer. Moreover, we can observe that the total data compression (i.e. reduction in entropy) that occurs in DNN is $\sum \Delta_i$ and is defined in the following definition:

Definition 10. *The total compression in a feed forward DNN with ν layers is illustrated by C_ν and we have:*

$$\mathsf{C}_\nu = \sum_{i=1}^{\nu} \Delta_i$$

The following lemma shows an upper bound on C_ν:

Lemma 6. *In a multi-layer neural network with input X, output layer X_ν, and the desired output Y, the maximum possible entropy reduction from the input space to the output space that satisfies the distortion constraint is $H(X|Y) - H(X_\nu|Y)$.*

Proof. From the constraint of (5.20) we have:

$$\sum_{i=1}^{\nu} \Delta_i \le \epsilon + \sum_{i=1}^{\nu} \Delta'_i \tag{5.21}$$

However, we know that $\Delta'_i = H(X_{i-1}|Y) - H(X_i|Y)$. Hence, we have

$$\sum_{i=1}^{\nu} \Delta'_i = H(X|Y) - H(X_1|Y) + H(X_1|Y) - H(X_2|Y) \\ + H(X_2|Y) - \cdots - H(X_\nu|Y) = H(X|Y) - H(X_\nu|Y)$$

Therefore, using Definition 10 and (5.21) and when $\epsilon \to 0$ we have:

$$\mathsf{C}_\nu \leq H(X|Y) - H(X_\nu|Y) \tag{5.22}$$

This proves the lemma. □

The first consequence of Lemma 6 is that, regardless of the number of layers, C_ν cannot be greater than $H(X|Y)$. In fact, considering Lemma 5, $H(X_i|Y)$ is a non-increasing function of i, and hence we have:

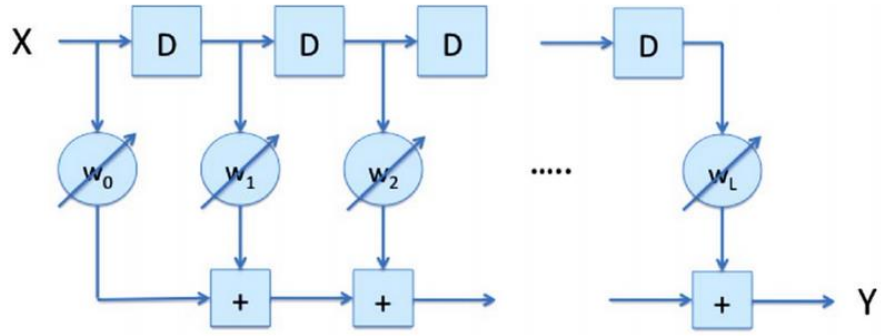
$$\mathsf{C}_\nu \leq H(X|Y) \tag{5.23}$$

However, it should be noted that higher number of layers may result in a more compressed representation of data. In other words, based on Lemma 5, for $\nu_1 > \nu_2$ we have $H(X_{\nu_1}|Y) \geq H(X_{\nu_2}|Y)$ and hence, $\mathsf{C}_{\nu_1} \leq \mathsf{C}_{\nu_2}$.

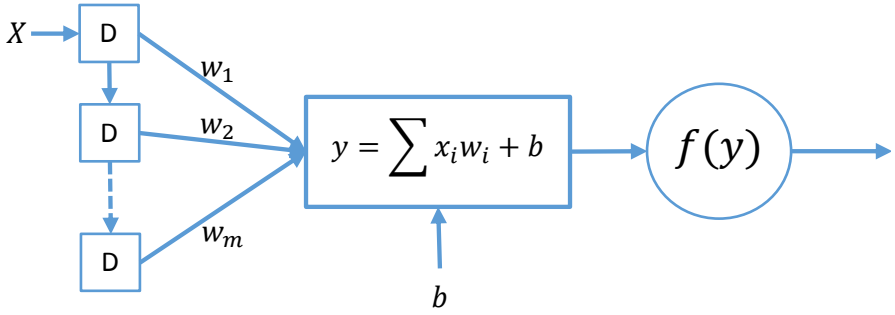
5.3 Online Denoising with Deep Networks

In this section, we explore the application of deep neural networks for time series online denoising. A straightforward solution for this purpose is to use noiseless and stabilized samples of time series to train a neural network and then use the trained network for denoising purposes. Due to the locality properties of real-world time series, one may use a number of consecutive samples of time series as the input to the denoising model. Addressing the locality in a time series can be handled in a similar way that we used *context windows* in Chapter 3. One approach to apply linear functions on local contexts of a time series is to use *tapped delay lines* (TDLs). The general model of a TDL is illustrated in Fig. 5.4(a). TDL is a signal processing model that is widely used for various applications such as signal correlation, equalization, and finite-impulse response (FIR) filtering [39, 53]. Combining the idea of time series locality with a single neuron structure (Fig. 5.3) results in the structure shown in Fig. 5.4(b). In this figure, X is the noisy time series and m consecutive samples of it are fed into the neuron structure to generate the final (denoised) result. The basic structure of Fig. 5.4(b) can be generalized to a feed-forward deep neural network that has been shown to be a powerful structure for nonlinear regression and classification tasks. The structure of a feed-forward DNN with delay line is illustrated in Fig. 5.4(c).

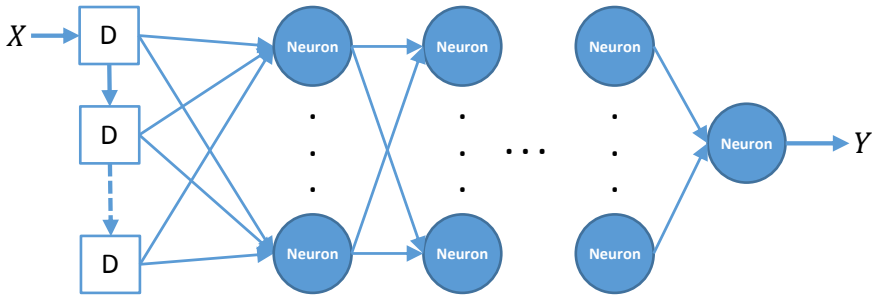
The feed-forward DNN model for denoising purpose is illustrated in Fig. 5.5. The training phase of the DNN is represented in Fig. 5.5(a). In this model, noiseless samples of time



(a)

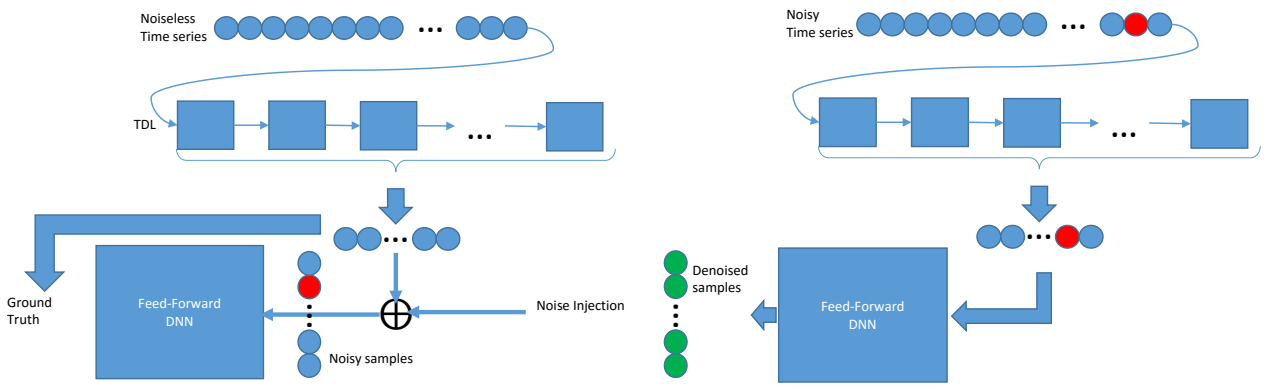


(b)



(c)

Figure 5.4: (a) Structure of a TDL [15]. (b) Single neuron with TDL. (c) Feed-forward DNN with delay line where each of the neurons has the structure shown in Fig. 5.3.



(a) (b)

Figure 5.5: Feed-forward DNN model for denoising purpose in (a) training phase and (b) working condition. Blue dots are noiseless samples, red dots are noisy ones and green dots represent the denoised samples.s

series are manipulated by noise injection. The synthetically generated noisy data is given to the neural network together with its noiseless equivalent. However, noise is a random effect and hence, for each noiseless data sample, multiple noisy training samples should be fed into the neural network. After training, the neural network is ready for the denoising process which is illustrated in Fig. 5.5(b).

Let us assume that the noiseless time series is $X = x_1, \cdots, x_t$ and the noisy observation is $Z = z_1, \cdots, z_t$. Moreover, assume that a ν layer feed-forward neural network is used for the denoising and the denoised output of the network is $\hat{X} = \hat{x}_1, \cdots, \hat{x}_t$. If at each denoising step we fed a k-sample subset of the time series to the neural network, at a particular time, t, the input to the network is $Z^t_{t-k+1} = z_{t-k+1}, \cdots, z_t$, its noiseless equivalent is $X^t_{t-k+1} = x_{t-k+1}, \cdots, x_t$, and the denoised data is $Z_\nu = \hat{X}^t_{t-k+1} = \hat{x}_{t-k+1}, \cdots, \hat{x}_t$. This is illustrated in Fig. 5.6.

From information theory perspective, it is important to maximize the mutual information between X and $\hat{X}$. Based on the definition of mutual information we have:

$$I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) = H(\hat{X}^t_{t-k+1}) - H(\hat{X}^t_{t-k+1}|X) \tag{5.24}$$

using Lemma 4 and 5 we can rewrite Eq. 5.24 as follows:

$$\begin{aligned} I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) = H(Z^t_{t-k+1}) - \sum_{i=1}^{\nu} \Delta_i \\ - \left\{ H(Z^t_{t-k+1}|X^t_{t-k+1}) - \sum_{i=1}^{\nu} \Delta'_i \right\} \end{aligned} \tag{5.25}$$

On the other hand, based on Definition 10, $\sum_{i=1}^{\nu} \Delta_i$ is the total compression of information,

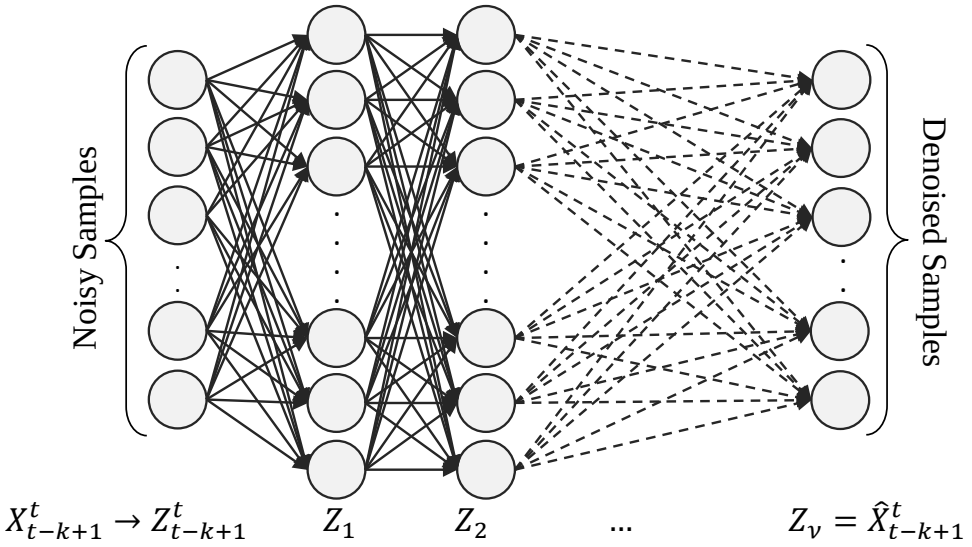


Figure 5.6: Model of a feed-forward DNN used in a denoising process.

C_ν, performed by the neural network. Hence, we have:

$$I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) = H(Z^t_{t-k+1}) - \mathsf{C}_\nu - \left\{ H(Z^t_{t-k+1}|X^t_{t-k+1}) - \sum_{i=1}^{\nu} \Delta'_i \right\} \tag{5.26}$$

Based on Lemma 6 we know that $\mathsf{C}_\nu \leq H(Z^t_{t-k+1}|X^t_{t-k+1})$. Therefore,

$$H(Z^t_{t-k+1}) - I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) - \left\{ H(Z^t_{t-k+1}|X^t_{t-k+1}) - \sum_{i=1}^{\nu} \Delta'_i \right\} \leq H(Z^t_{t-k+1}|X^t_{t-k+1}) \tag{5.27}$$

It is worth mentioning that Z^t_{t-k+1} is the noisy version of X^t_{t-k+1}. Then, if N_k represents the noise random variable for k consecutive samples of time series, it can be easily shown that $H(Z^t_{t-k+1}|X^t_{t-k+1}) = H(N_k)$. Hence, we have:

$$H(Z^t_{t-k+1}) - I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) - H(N_k) + \sum_{i=1}^{\nu} \Delta'_i \leq H(N_k) \tag{5.28}$$

or equivalently

$$H(Z^t_{t-k+1}) - 2H(N_k) + \sum_{i=1}^{\nu} \Delta'_i \leq I(X^t_{t-k+1}; \hat{X}^t_{t-k+1}) \tag{5.29}$$

The importance of Eq. 5.29 is that it includes some features of the deployed DNN. Note that we typically have no control over the noise, N_k, and the denoising DNN is the only part of the whole system that we can manipulate. In Eq. 5.29, $\sum_{i=1}^{\nu} \Delta'_i$ depends on two factors:

number of layers, ν, and amount of conditional entropy reduction between consecutive layers, Δ'_i, which depends on the consecutive partitioning/mappings performed in the network. Equation 5.29 is summarized in the following lemma.

Lemma 7. *In an appropriately trained denoising feed-forward DNN where X_{t-k+1}^t is the k consecutive samples of the noiseless data, Z_{t-k+1}^t is the k-sample noisy observations, $\hat{X}_{t-k+1}^t$ is the corresponding denoised sequence, and N_k is the noise random variable, we have*

$$H(Z_{t-k+1}^t) - 2H(N_k) + \sum_{i=1}^{\nu} \Delta'_i \leq I(X_{t-k+1}^t; \hat{X}_{t-k+1}^t)$$

Based on Lemma 7, when Eq. 5.29 is not satisfied it means that the training phase of the denoising DNN has not been completed (i.e. did not satisfy the optimization problems of 5.16 and 5.20). Experimental results support this justification.

5.4 Experimental Results

In order to evaluate the performance of the proposed DNN-based denoiser we performed a large variety of experiments with real-world time series. In this section we present some representative results which demonstrate the relative performance of the proposed algorithm.

5.4.1 Performance under Uniform Noise

In this section, we present results of the experiments performed under uniform noise injection. In this set of experiments, 15 time series introduced in Table 5.1 are used. Uniform noise is injected to each time series for 50 times and the average RMS error of the noisy and denoised time series with respect to the noiseless data are measured and reported in Table 5.1, Table 5.2, and Fig. 5.9 and Fig. 5.8. The uniform noise is injected based on Eq. 2.4 with ϵ value varying between 0.0 and 0.55. In Table 5.1 settings of the denoising DNN for each time series are also illustrated. For each case, 75% of the data is used for the training purpose and the remaining 25% is used during the denoising phase.

Training and denoising times are illustrated in Table 5.1. It can be observed from this table that the denoising time is very short while the training time greatly depends on the alphabet size, length of the time series, and properties of the denoising DNN. These times are measured for the implementation of the system in MATLAB R2014a over an Intel Core i7 computer with 8 GBytes of RAM and 64-bit Windows 10 Home edition operating system.

Performance of the denoising DNN for various error probabilities (i.e. ϵ in Eq. 2.4) is illustrated in Fig. 5.7. In this figure, RMS error of the noisy and denoised time series with respect to the noiseless data are depicted. We can observe that while for smaller values of ϵ

Table 5.1: Features of the time series and the corresponding setting of the DNN together with the training and denoising running times.

Time Series Source	Time Series Name	Alphabet Size	Time Series Length	Neurons in Hidden Layers	Window Size (k)	Training Time	Denoising Time
WUT	Blacksburg	4838	2769	1	9	3.87	0.010
WUT	Christiansburg	3878	2769	3	2	2.85	0.010
WUT	Radford	3926	2769	9	1	1.69	0.010
WUT	Salem	3836	2769	1	1	0.99	0.009
WUT	Roanoke	4199	2769	9	2	1.68	0.012
WUT	Organic Food	5328	2769	8	7	34.28	0.011
WUT	Organic Chemistry	7370	2769	5	6	20.24	0.010
WUT	Organic Compound	5342	2769	7	9	55.27	0.011
WUT	Organic Farming	11916	2769	4	9	56.55	0.012
WUT	Organic Matter	3049	2769	9	1	2.29	0.011
PAHO	Argentina	1815	172	3	2	0.23	0.011
PAHO	Bolivia	503	172	4	2	0.27	0.012
PAHO	Guatemala	95	172	8	2	0.30	0.009
PAHO	Honduras	49	172	3	1	0.16	0.009
PAHO	Panama	59	172	4	2	0.29	0.010

the proposed denoising method is not effective, for larger values this approach has reduced the RMS error dramatically. RMS errors for the noisy and denoised time series for different datasets are compared in Table 5.2. In this table, for uniform noise model (the 3^{rd} and 4^{th} columns), error probability is 0.35. We can observe that for most of the time series, the RMS error has been reduced significantly.

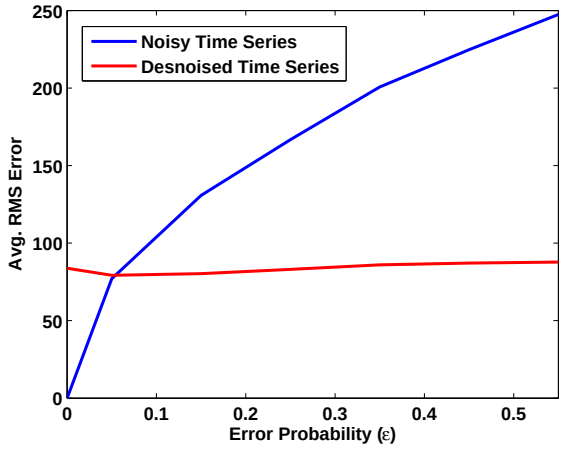


Figure 5.7: Performance of the proposed denoising DNN under uniform noise for WUT of Blacksburg time series.

The theoretical bound of Lemma 7 is evaluated in Fig. 5.8. In this figure, the blue curve illustrates how the left side of inequality 5.29 changes versus ϵ while the red curves shows the variations in the mutual information between the noiseless data and the denoised one. Results show that for most of the data points, this inequality is satisfied. The only exception

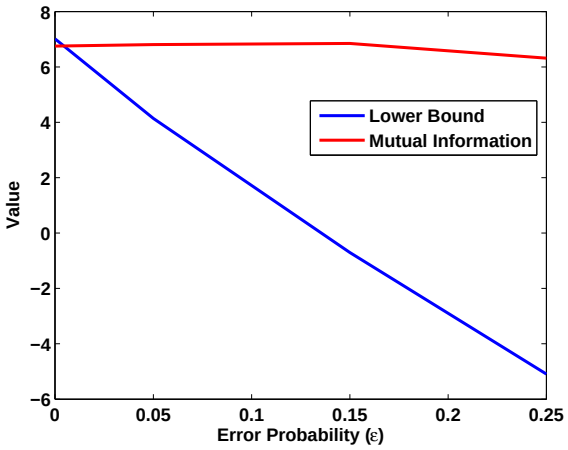


Figure 5.8: Left and right sides of the inequality of Lemma 7. Results are for PAHO-Argentina time series.

is when ϵ is 0. Further investigations revealed that for this case, the training algorithm of the DNN did not converge and the training phase has not been completed. This means that the training process did not satisfy the optimization problems of 5.16 and 5.20 and hence, Lemma 7 conditions should not be satisfied.

5.4.2 Performance under Normal Noise

In this section, we present results of the experiments performed under normal noise injection. Similar to the previous subsection, we used the time series of Table 5.1. Normal noise is injected to each time series for 50 times and the average RMS error of the noisy and denoised time series with respect to the noiseless data are measured and reported in Table 5.2 and Fig. 5.9. The noise is injected based on Eq. 2.8 and Eq. 2.9 with σ_R value varying between 0.0 and 0.3.

Performance of the denoising DNN for various time series is illustrated in Table 5.2. In this table, RMS errors for the noisy and denoised time series are compared (the last two columns of the table). In this set of experiments the value of σ_R is 0.2. We can observe that for most of the time series, the RMS error has been reduced significantly. In Fig. 5.9, RMS error of the noisy and denoised time series with respect to the noiseless data are depicted for WUT-Blacksburg time series with normal noise injection. This figure, illustrates how RMS error changes when σ_R changes from 0.0 to 0.3. We can observe that while for smaller values of σ_R the proposed denoising method is not effective, for larger values this approach has reduced the RMS error dramatically.

Table 5.2: Performance of the proposed denoising DNN for different time series and noise models. For uniform noise model ϵ is 0.35 and for normal noise model σ_R is 0.2.

Time Series Source	Time Series Name	RMS Error Noisy Uniform Noise	RMS Error Denoised Uniform Noise	RMS Error Noisy Normal Noise	RMS Error Denoised Normal Noise
WUT	Blacksburg	200.68	85.98	960.8	89.81
WUT	Christiansburg	23.88	15.86	779.64	44.82
WUT	Radford	67.00	60.56	778.46	66.74
WUT	Salem	72.92	25.96	765.27	30.00
WUT	Roanoke	386.01	135.49	841.35	153.45
WUT	Organic Food	1064.27	500.30	1057.47	525.80
WUT	Organic Chemistry	1817.86	735.43	1461.02	736.94
WUT	Organic Compound	1209.21	413.49	1065.18	446.12
WUT	Organic Farming	3404.96	614.07	2374.77	556.35
WUT	Organic Matter	485.43	118.42	662.37	145.28
PAHO	Argentina	396.73	364.08	355.03	281.46
PAHO	Bolivia	161.90	83.44	98.57	80.96
PAHO	Guatemala	9.26	7.58	18.73	9.69
PAHO	Honduras	10.16	11.43	9.53	9.60
PAHO	Panama	14.13	11.60	11.69	9.77

5.4.3 Denoising of Time Series with Heterogeneous Quality

Non-uniform data collection mechanisms and inaccurate estimation techniques result in time series with heterogeneous quality. As an example, as we described in Chapter 2, influenza count estimations for a particular week of a year may vary over time [14]. In fact, these estimates are typically generated with a significant delay (e.g. a few weeks) and even after initial announcement, they are revised multiple times. Therefore, influenza counts for a particular week have a varying quality over time which makes the denoising process more complicated. In this section we propose and evaluate a modified denoising DNN for time series with heterogeneous quality.

Due to the varying quality of time series over time, in addition to the local values of the time series, we need another input to the network which represents the time span between the actual time that the value belongs to and the recent update. Let us assume that at time τ we have observed $Z^\tau = z_1^\tau, \cdots, z_t^\tau$ and we have $\tau \geq t$. Then, in order to denoise the value of z_t^τ with a context window of size k the following vector is used as the input to the denoising DNN:

$$V_{t,\tau}^{input} = \left[\tau - t\,,\, z_{t-k+1}^\tau\,,\, z_t^\tau\right] \tag{5.30}$$

The corresponding neural network is illustrated in Fig. 5.10. In this model, $V_{t,\tau}^{input}$ is fed into the network to generate the denoised version of Z_t^τ, $\hat{X}_t^\tau$.

In order to evaluate the performance of the denoising DNN of Fig. 5.10, we performed some experiments with influenza counts of Argentina. In this set of experiments we used the context window of $k = 4$. Experimental results are presented in Fig. 5.11. It can be observed from this figure that for earlier updates (when $\tau - t$ is smaller than 8 weeks) the proposed model can efficiently denoised the time series and generate a time series with higher quality.

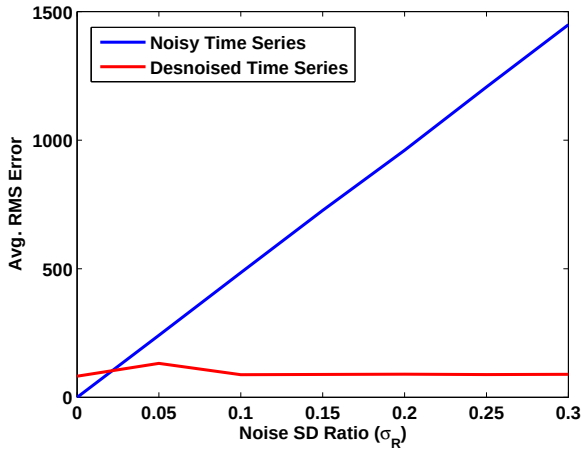


Figure 5.9: Performance of the proposed denoising DNN under normal noise for WUT of Blacksburg time series.

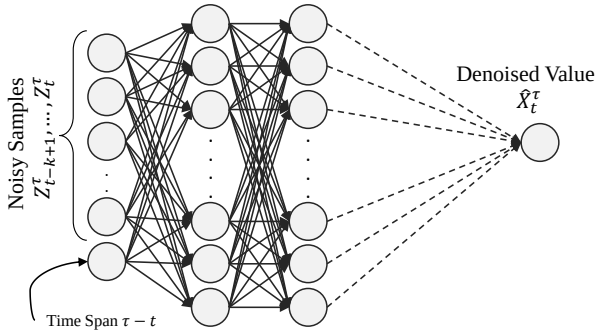


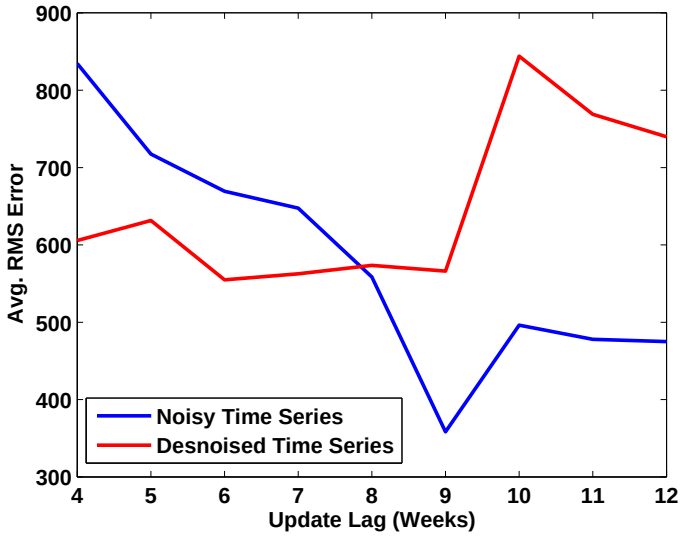
Figure 5.10: Neural network model used in the denoising of time series with heterogeneous quality.

However, when $\tau - t$ is greater than 8, the quality of the noisy time series is higher. This is consistent with our previous observations. In fact, when $\tau - t$ is increased the quality of the observed time series is improved (i.e. level of noise is reduced) and as we observed in previous subsections, with lower levels of noise, the noisy time series have a better quality than its denoised equivalent.

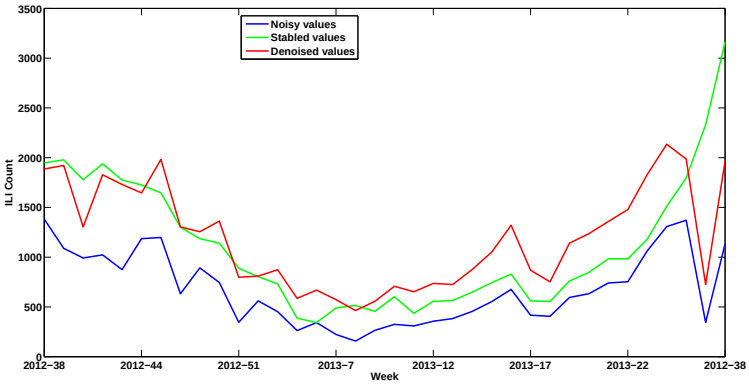
5.5 Discussions

In this chapter we studied the application of feed-forward DNNs in the denoising of time series. We proposed a framework for this purpose which works based on the locality features of the time series. Furthermore, we theoretically studied some information theoretic features of a DNN. In this regard, we used information theory methods to study the flow of informa-

tion in DNNs. We determined how entropy and conditional entropy of information changes between consecutive layers and using the Information Bottleneck principle we modeled the learning process of a feed-forward neural network as a constrained optimization problem. Furthermore, we proved an upper bound for the total compression rate of information that can be achieved in a neural network while the overall distortion in the output layer with respect to a desired output is in an acceptable range. Moreover, for the denoising DNN we proved a lower bound on the mutual information between the noiseless and denoised time series which should be satisfied when the training process has been appropriately completed. Experimental results with various distortions show that the proposed denoising model can efficiently denoise the corrupted time series. In this chapter, using the specific features of a denoising feed-forward DNN, we extended our theoretical analysis of feed-forward neural networks to the denoising DNNs. A similar approach may be used to extend the theoretical analysis to other applications (e.g. pattern recognition or dimensionality reduction) based on their specific characteristics.



(a)



(b)

Figure 5.11: (a) Average RMS error of the denoising DNN of Fig. 5.10 for denoised and noisy time series of Argentina versus time span between the data update and the actual data (i.e. $\tau - t$) (b) Time series of Argentina ILI count. Unstable time series are compared with stabilized and denoised ones for the updates 6 weeks after the actual time.

Chapter 6

Conclusion

A huge amount of the data that is generated and collected every second around the world is temporal data and is represented in the form of time series. In the past decades, the demand for time series analysis has been grown dramatically, both in the private and public sectors. However, due to the imperfect measurement and data collection mechanisms, real-world time series are distorted by various types of noise and instability. Therefore, working with noisy time series is an inevitable part of any time series mining task and must be addressed precisely. In this dissertation, we proposed multiple solutions for the problem of forecasting with noisy and unstable time series.

In Chapter 2, after addressing the locality and seasonality features of real-world time series, we presented a simple case study to address the denoising of a synthetically generated sawtooth wave time series. Then, we proposed seasonal regression algorithms, as a data correction mechanism, to deal with noise and instabilities in real-world seasonal time series. In this regard, we presented two case studies. The first case study was for tourism demand time series of Hawaii and the second case study addressed the instability issues in the ILI count time series of Latin American countries. Experimental results showed that using an appropriate denoising mechanism we can significantly improve the forecasting performance.

In Chapter 3, we developed online denoising algorithms for discrete value time series based on the DUDE algorithm. Here we proposed three denoising algorithms to strike the tradeoff between time sensitivity and denoising accuracy which is important for real-time applications. We proved that the online algorithm asymptotically converges to the optimum offline denoiser and we provided numerical results to support our claims. Furthermore, we extended the offline and online denoisers to address the large alphabet issue of original DUDE algorithm and its online offspring. In this regard, we proposed a hierarchical quantization based universal denoising frame work. Experimental results with synthetic and real-world time series show that the proposed algorithms can efficiently denoise discrete value time series in appropriate time.

In Chapter 4, we proposed a novel communication theoretic approach to model the problem of noise and instability in time series forecasting. Using the proposed NCF model, we provided required formulations to estimate the level of noise in a time series under an additive Gaussian noise setting. We also showed that under certain circumstances the proposed approach improves the forecasting results. Furthermore, we proposed an out-of-band noise filtering approach to reduce the noise level in the time series. Numerical results demonstrated that the proposed solution results in higher forecasting accuracy compared with the traditional methods.

In Chapter 5, we proposed a deep learning based solution for the problem of time series online denoising. In this chapter we first studied the problem of information flow in a deep neural network. We proved that the entropy of information is on a non-increasing trend. We also determined the amount of entropy change between consecutive layers of a feed-forward DNN. Moreover, an upper bound is determined in this chapter for the total compression rate of information that can be achieved in a neural network. Using the information theoretic features of DNNs and the specific properties of a denoising DNN, we determined a theoretical bound for the mutual information between the noiseless and denoised time series. This bound should be satisfied when the neural network has been trained successfully. Various experiments with different time series and noise models showed that the proposed DNN-based denoising framework can efficiently perform the denoising task.

The proposed denoising methods in this thesis have different characteristics and hence, to efficiently denoise time series, each of these methods can be used under specific conditions. In the following section, we compare these methods with respect to their specific features.

6.1 Comparison and Summary of Results

As we mentioned in Chapter 1, different solutions are required for the denoising of time series based on the level of knowledge that we have regarding the time series properties and the noise model. Furthermore, we need to choose appropriate denoising techniques based on the properties of the noise and time series. From this perspective, each of the solutions that have been presented in this thesis are appropriate for a particular setting. To summarize it, we have:

- Depending on the seasonality features of the time series, one may use the seasonal regression methods presented in Chapter 2. As we showed in this chapter, for highly seasonal time series such as tourism demand, we may perform the denoising task only based on the seasonal information. However, when time series has no seasonal behavior, seasonal regression denoising method is not effective and we have to rely on the locality features.

- For discrete valued time series we are able to use the class of discrete universal denoisers,

Table 6.1: Features of the proposed denoising methods.

Features	Seasonal Regression	Discrete Universal	Out-of-Band Filtering	Deep Learning based
Prior Knowledge about Noise	Not Required	Required	Required	Not Required
Noiseless Training Data	Required	Not Required	Not Required	Required
Supervised Learning	Yes	No	No	Yes
Discrete Alphabet	Yes	Yes	Yes	Yes
Continuous (real-value) Alphabet	Yes	No	Yes	Yes
Time Series Seasonality	Required	Not Required	Not Required	Not Required

studied in Chapter 3. In addition to the discrete-value property, the performance of these solutions greatly depend on our knowledge about the noise model. Hence, for real-value time series or when our knowledge about the noise model is not complete, discrete universal denoisers of Chapter 3 cannot be used efficiently.

- The noisy channel reversal model of Chapter 4 and the proposed out-of-band noise filtering have been developed based on the assumption that the time series forecasting model suffers from additive Gaussian noise. The proposed solution in Chapter 4 can efficiently used for all time series and not only reduces the amount of noise and improves the forecasting accuracy, but also can be used to estimate the noise features such as noise standard deviation. Moreover, the proposed noisy channel reversal model can be used to address the noise which is generated due to the use of inaccurate regression models.

- When we do not have enough knowledge about the noise properties, regression-based solutions of Chapter 2 and DNN-based solutions of Chapter 5 can be deployed for the denoising purposes. While the solutions of Chapter 2 have been proposed for seasonal time series, similar regression-based methods can be developed for non-seasonal time series. However, the proposed regression-based solutions are based on linear regression. It has been shown that DNNs can implement complex linear and non-linear regression algorithms and hence, the DNN-based denoising solutions can be used in more complicated denoising applications.

Different proposed solutions are compared in Table 6.1. In this table, denoising techniques are compared with respect to their requirements, functionality, and features. As indicated in Table 6.1, seasonal regression based denoisers (Chapter 2) and deep learning based denoisers (Chapter 5) are supervised techniques. As a matter of fact, in a regression based denoiser we need to train a regression model to determine its coefficients. Moreover, in a deep learning based denoiser, supervised learning is required to determine the neural network's weights. Due to this requirement, in regression based and deep learning based denoisers we need to have access to a training dataset which includes noiseless data samples and their noisy equivalent.

In online denoising of time series there is a tradeoff between the accuracy and time-sensitivity. In fact, depending on the time-sensitivity of the applications and the rate of data generation,

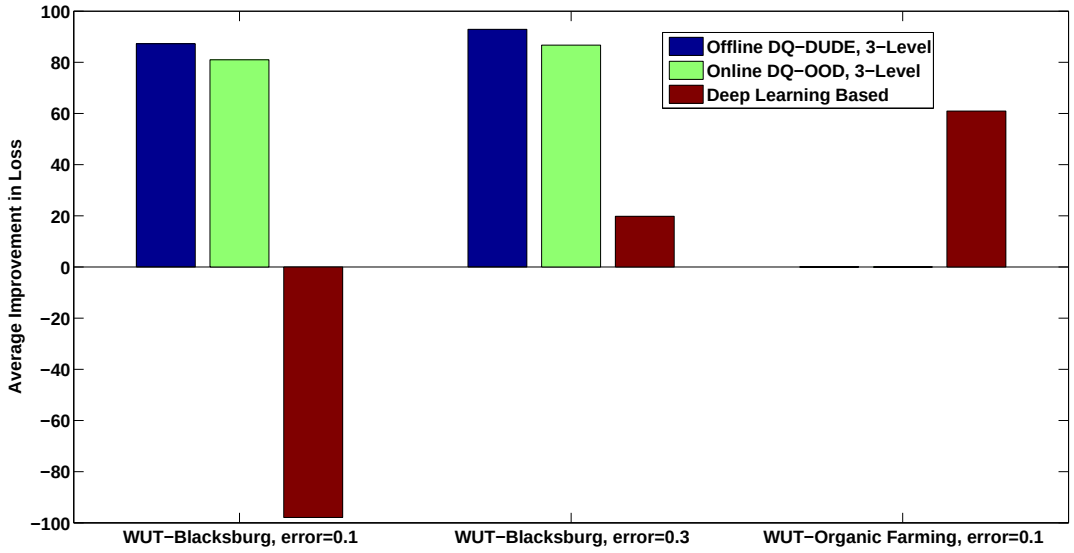


Figure 6.1: Effect of offline and online universal denoisers with three levels of quantization and deep learning based denoiser on the averaged loss improvement of two noisy time series. This experiment has been conducted under uniform noise model with $\epsilon = 0.1$ and 0.25.

one may need to sacrifice accuracy in order to keep the overall system working in an appropriate manner. In Chapter 3, we proposed various online denoising algorithms to address the above mentioned tradeoff. Furthermore, for different denoising algorithms we measured the running time of the algorithm. It is worth mentioning that the running time not only depends on the algorithm but also it depends on the alphabet size and the length of the time series. For DNN-based denoisers, alphabet size and time series length mainly affect on the duration of training phase while the denoising phase can be performed in a significantly shorter time.

Multiple factors have effect on the performance of denoising process. Accuracy of denoising not only depends on the denoising method and the time series properties but also on the noise statistics. It is worth mentioning that the most efficient denoising method for a noisy time series should be selected based on the time series and noise features. As an example, two time series with two different levels of noise are compared in Fig. 6.1. In this figure, Wikipedia Usage Trends of Blacksburg and Organic Farming Wikipedia pages with uniform noise injection are compared. Offline and online universal denoisers with three levels of quantization as well as deep learning based denoisers are used for the denoising purpose. Furthermore, the error probability for Organic Farming is 0.1 while for Blacksburg, two error probabilities, 0.1 and 0.25, have been used. Figure 6.1 illustrates how average loss is improved when a particular denoising technique is applied to the noisy time series. It is obvious that for the noisy Blacksburg WUT time series, offline and online universal denoisers are effective and

can improve the average loss of the time series by about 80%. For this time series and when error probability is 0.1, deep learning based denoising technique is not effective and results in higher values of loss while for error probability of 0.25 the average loss is improved by about 20%. On the other hand, when error probability is 0.1, universal denoisers with three levels of quantization are not able to denoise the Organic Farming WUT time series while a deep learning based denoiser was able to improve the quality of time series by 60%. This example shows that how the performance of a particular denoiser changes under different conditions. In order to achieve the best denoising performance, preliminary experiments may be used to select appropriate denoising algorithm and settings. Preliminary experiments should be performed on a validation dataset which includes training and test data. Furthermore, based on the validation dataset, one may be able to determine locality and seasonality features of the time series.

6.2 Future Work

The theoretical and experimental results presented in this dissertation establish a foundation for future research in denoising, forecasting, and deep learning:

- In Chapter 5 we used information theoretic concepts to study the behavior of DNNs. In this chapter, we only studied feed-forward DNNs and we theoretically studied the flow of information in this class of neural networks. It should be noted that to understand the dynamic behavior of DNNs, more theoretical effort is needed. Moreover, we need to extend the proposed framework to other types of neural networks. Furthermore, the application of other types of DNNs in the denoising process of time series should be studied properly. This should be supported by experimental results performed with real-world datasets. In this chapter, using the specific features of a denoising feed-forward DNN, we proved some theoretical bounds for the denoising neural networks. Theoretical bounds for the use of neural networks in other applications can be determined using the same approach.

- In Chapter 4, we proposed a framework for noisy channel reversal forecasting and we applied the framework to the autoregressive model. The proposed solution in this chapter can be extended to include other regression algorithms and noise models, such as ARIMA. As a future work, we aim to apply the noisy channel forecasting framework to other linear and non-linear regression algorithms. Furthermore, we will extend the framework to include non-Gaussian noise and distortions.

- While in this dissertation, denoising solutions are proposed as individual techniques, we may deploy them in serial and parallel ensembles as depicted in Fig. 6.2. In the parallel ensemble model of Fig. 6.2(a), various denoising methods are applied to the noisy time series and then, an appropriate ensemble method is used to combine the

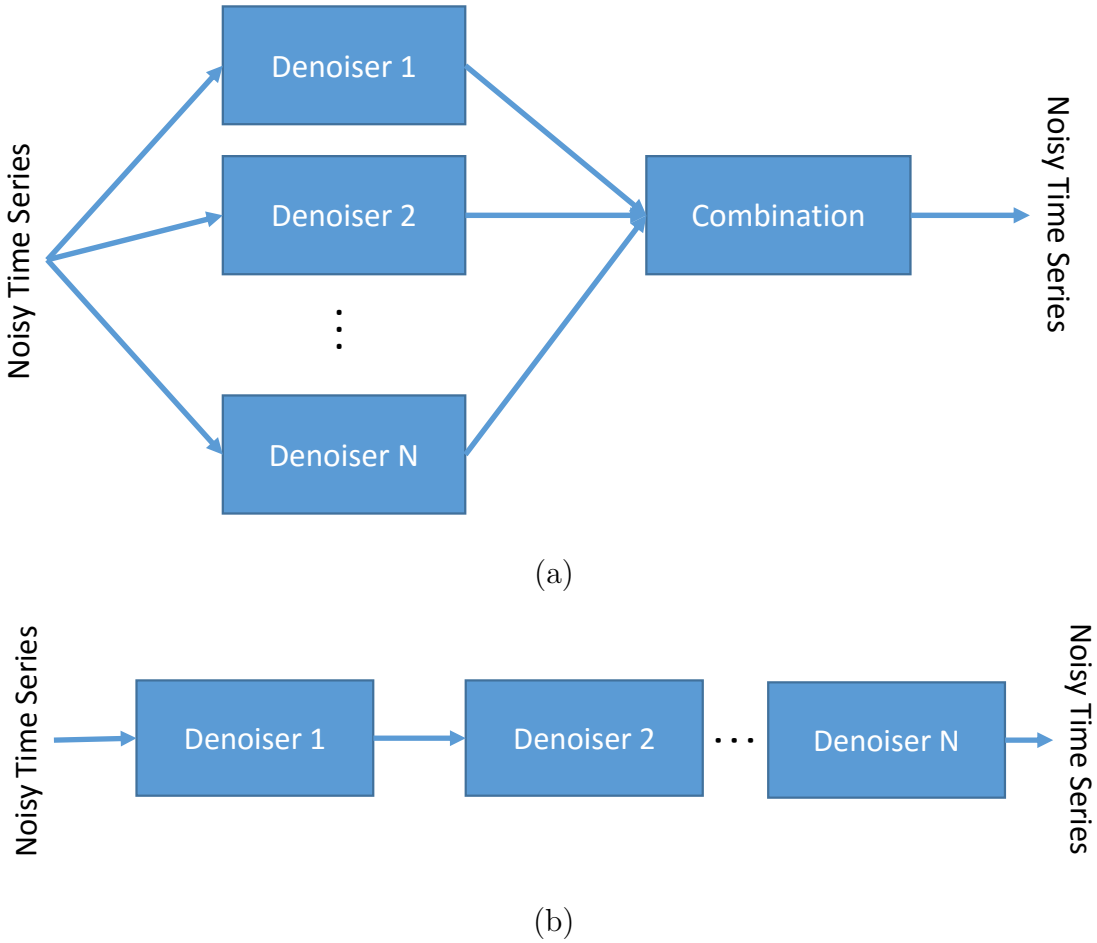


Figure 6.2: (a) Parallel ensemble of denoisers (b) Serial ensemble of denoisers.

generated denoised values. In the serial ensemble model of Fig. 6.2(b), each denoiser is applied to the output of another denoiser. As an example, the out-of-band noise filtering method, presented in Chapter 4, can be used as a pre-processing step and its output may be used as the input to any other denoising algorithm. As a future work, different ensemble methods can be explored.

- In this dissertation, we addressed the denoising problem of uni-variate time series with no side information. However, many time series used in data mining problems and forecasting tasks are multi-variate ones. As an example, different tourism demand forecasting algorithms use multiple time series, such as demand history, tourists income, and living expenses at a particular destination [37]. Furthermore, in some situations, we may have useful side information about the time series which may be deployed in the denoising process. As a future work, the denoising problem of time series can be extended to multi-variate time series and to incorporate side information in the denoising process.

- In this dissertation, we proposed various time series denoising algorithms for forecasting applications. Denoising algorithms may also be deployed for other applications such as anomaly detection. It can be easily shown that there is a close relationship between the noise and anomaly in time series. Therefore, we expect that denoising algorithms can be used as anomaly detection tools. As an example, after denoising a time series, we may compare that with the original observation and if we observe a significant difference between the original time series and its denoised version, it can be interpreted as the anomalous behavior of the original data. This application can be explored as a future work.

- An important open problem that should be addressed as a future work is the capacity of forecasting algorithms when time series are noisy. The main question is that from an information theoretical perspective, what is the maximum achievable performance (in terms of accuracy) for any particular forecasting method when observed time series suffer from noise and instability. We believe that this is an important theoretical problem as its result may be used to determine the robustness of various forecasting methods.

Bibliography

[1] T. Alexandrov. A method of trend extraction using singular spectrum analysis. *Statistical Journal*, 7(1), 2009.

[2] T. Alexandrov et al. *Smoothing, Forecasting and Prediction of Discrete Time Series*. U.S. Census Bureau Research Report Series, 2008.

[3] A. Apolloni, V. Kumar, M. Marathe, and S. Swarup. Computational epidemiology in a connected world. *Computer*, 42(12):83–86, 2009.

[4] G. Athanasopoulos and A. de Silva. Multivariate exponential smoothing for forecasting tourist arrivals. *Journal of Travel Research*, 51(5):640652, 2012.

[5] Y. Aviv. A time-series framework for supply-chain inventory management. *Operations Research*, 51(2):210–227, 2003.

[6] P. Bangwayo-Skeete and R. Skeete. Can google data improve the forecasting performance of tourist arrivals? mixed-data sampling approach. *Tourism Management*, 46:454–464, 2015.

[7] S. A. Billings and K. L. Lee. A smoothing algorithm for nonlinear time series. *International Journal of Bifurcation and Chaos*, 14(3):1037–1051, 2004.

[8] C. M. Bishop. Theoretical foundations of neural networks. In *Proceedings of Physics Computing*, pages 500–507, 1996.

[9] R. Blahut. Computation of channel capacity and rate-distortion functions. *Information Theory, IEEE Trans. on*, 18(4):460–473, April 1972.

[10] G. Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[11] R. G. Brown. *Smoothing, Forecasting and Prediction of Discrete Time Series*. Dover Publications, 2004.

[12] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms with a new one. *SIAM Journal on Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

[13] S. Buadhachain and G. Provan. A model-based control method for decentralized calibration of wireless sensor networks. In *American Control Conference*, pages 6571–6576, 2013.

[14] P. Chakraborty, P. Khadivi, et al. Forecasting a moving target: Ensemble models for ili case count predictions. In *Proceedings of the SIAM International Conference on Data Mining*, pages 262–270, 2014.

[15] K.-C. Chen, S.-L. Huang, L. Zheng, and H. V. Poor. Communication theoretic data analytics. *IEEE Journal on Selected Areas in Communications*, 33(4):663–675, April 2015x.

[16] X. Chen and X.Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2(2):514–525, 1991.

[17] C. Chew and G. Eysenbach. Pandemics in the age of twitter: Content analysis of tweets during the 2009 h1n1 outbreak. *PlosOne*, 5(11):e14118, 2013.

[18] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 2006.

[19] K. Denecke, P. Dolog, and P. Smrz. Making use of social media data in public health. In *Proceedings of WWW '12*, pages 243–246, 2012.

[20] T. Dietterich. *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, chapter Machine Learning for Sequential Data: A Review, pages 15–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[21] R. Ganesh and J. Kumari. A survey on channel estimation techniques in mimo-ofdm mobile communication systems. *IJSER*, 4(5):1851–1855, 2013.

[22] J. Gao, H. Sultan, J. Hu, and W.-W. Tung. Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: A comparison. *IEEE Signal Processing Letters*, 17(3):237–240, 2010.

[23] G. Gemelos, S. Sigurjonsson, and T. Weissman. Algorithms for discrete denoising under channel uncertainty. *IEEE Tran. Signal Processing*, 54(6):22632276, 2006.

[24] S. Gershman and D. Blei. A tutorial on bayesian nonparametric models. *J. of Math. Phys.*, 56(1):1–12, 2012.

[25] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant1. Influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[26] C. Giurcaneanu and B.Yu. Efficient algorithms for discrete universal denoising for channels with memory. In *Proc. of the IEEE ISIT*, 2005.

[27] J. Guerard and E. Schwartz. Regression analysis and forecasting models. In *Quantitative Corporate Finance*, pages 277–301. Springer, 2007.

[28] U. Gunter and I. Onder. Forecasting international city tourism demand for paris: Accuracy of uni- and multivariate models employing monthly data. *Tourism Management*, 46:123–135, 2015.

[29] M. Han, Y. H. Liu, J. H. Xi, and W. Guo. Noise smoothing for non-linear time series using wavelet soft threshold. *IEEE Signal Processing Letters*, 14:62–65, 2007.

[30] S. Haykin and M.Moher. *Introduction to Analog and Digital Communications*. John Wiley and Sons, 2007.

[31] K. Hickmann et al. Forecasting the 20132014 influenza season using wikipedia. *PLoS Comput Biol*, 11(5), May 2015.

[32] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[33] J. Honaker and G. King. What to do about missing values in time-series cross-section data. *American Journal of Political Science*, 54(2):561–581, 2010.

[34] J. Hu, J. Gao, and K. White. Estimating measurement noise in a time series by exploiting nonstationarity. *Chaos, Solitons and Fractals Journal*, 22:807–819, 2004.

[35] N. Kanhabua and W. Nejdl. Understanding the diversity of tweets in the time of outbreaks. In *Proceedings of WWW '13*, pages 1335–1342, 2013.

[36] P. Khadivi, P. Chakraborty, R. Tandon, and N. Ramakrishnan. Time series forecasting via noisy channel reversal. In *Proceeding of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2015)*, 2015.

[37] P. Khadivi and N. Ramakrishnan. Wikipedia in tourism industry: Forecasting and usage behavior. In *Proceedings of Innovative Applications of Artificial Intelligence (IAAI)*, 2016.

[38] P. Khadivi, R. Tandon, and N. Ramakrishnan. Online denoising of discrete noisy data. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT'15)*, 2015.

[39] S. Khaleghi et al. High-speed correlation and equalization using a continuously tunable all-optical tapped delay line. *IEEE Photonics Journal*, 4(4):1220–1235, 2012.

[40] S. Kim, K. Koh, S. Boyd, and D. Gorinevsky. l-1 trend filtering. *International Journal of Bifurcation and Chaos*, 51(2):339360, 2009.

[41] P. Kostkova. A roadmap to integrated digital public health surveillance: the vision and the challenges. In *Proceedings of WWW '13*, pages 687–694, 2013.

[42] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5), April 2015.

[43] H. Li. Deep learning for image denoising. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(3):171180, 2014.

[44] L. Li, J. McCann, N. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2009.

[45] P. Mehta and D. J. Schwab. An exact mapping between the Variational Renormalization Group and Deep Learning. *ArXiv e-prints*, Oct. 2014.

[46] F. Mhamdi, J. Poggi, and M. Jaidane. Trend extraction for seasonal time series using ensemble emprical mode decomposition. *Advances in Adaptive Data Analysis*, 3(3):363383, 2011.

[47] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[48] M. Momtazpour, R. Sharma, and N. Ramakrishnan. An integrated data mining framework for analysis and prediction of battery characteristics. In *Proceedings of IEEE Innovative Smart Grid Technologies Conference*, 2014.

[49] M. Momtazpour, J. Zhang, S. Rahman, R. Sharma, and N. Ramakrishnan. Analyzing invariants in cyber-physical systems using latent factor regression. In *Proceedings of 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2015.

[50] T. Moon and T. Weissman. Discrete denoising with shifts. *IEEE Tran. on Information Theory*, 55(11):5284–5301, 2009.

[51] G. Motta, E. Ordentlich, et al. The idude framework for grayscale image denoising. *IEEE Tran. on Image Processing*, 20(1):1–21, 2011.

[52] M. Motwani, M. Gadiya, R. Motwani, and F. C. Harris. A survey of image denoising techniques. In *in Proceedings of GSPx 2004*, 2004.

[53] J. G. Proakis. *Digital Communications.* McGraw-Hill, 2000.

[54] S. Pyatykh and J. Hesser. Salt and pepper noise removal in binary images using image block prior probabilities. *Journal of Visual Communication and Image Representation*, 25(5):748754, 2014.

[55] L. Ralaivola and F. d'Alche Buc. Time series filtering, smoothing and learning using the kernel kalman filter. In *in Proceedings of IEEE International Joint Conference on Neural Networks*, 2005.

[56] V. Romero and A. Salmeron. Multivariate imputation of qualitative missing data using bayesian networks. In *Soft methodology and random information systems*, 2004.

[57] M. Saar-Tsechansky and F. Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1625–1657, 2007.

[58] N. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *Computational Intelligence, IEEE*, 4(2):24–38, 2009.

[59] S. Sarkar, X. Jin, and A. Ray. Data-driven fault detection in aircraft engines with noisy sensor measurements. *Journal of Engineering for Gas Turbines and Power*, 13, August 2011.

[60] J. Schmidhuber. Deep learning in neural networks: An overview. Technical Report IDSIA-03-14, The Swiss AI Lab IDSIA, University of Lugano & SUPSI, Swiss, 2014.

[61] J. Shaman, E. Goldstein, and M. Lipsitch. Absolute Humidity and Pandemic Versus Epidemic Influenza. *American journal of epidemiology*, 173(2):127–135, 2010.

[62] J. Shaman, V. E. Pitzer, C. Viboud, B. T. Grenfell, and M. Lipsitch. Absolute humidity and the seasonal onset of influenza in the continental United States. *PLoS biology*, 8(2):e1000316, 2010.

[63] K. S. Shanmugam. *Digital and Analog Communication Systems*. John Wiley and Sons, 1979.

[64] H. Song and G. Li. Tourism demand modeling and forecasting-a review of recent research. *Tourism Management*, 29:203–220, 2008.

[65] H. Song and S. F. Witt. Forecasting international tourist flows to macau. *Tourism Management*, 27:214224, 2006.

[66] J. Stevens. Outliers and influential data points in regression analysis. *Psychological Bulletin*, 95:334–344, 1984.

[67] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proc. of the CVPR'14*, pages 1891–1898, 2014.

[68] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Proc. of NIPS'13*, pages 2553–2561, 2013.

[69] J. D. Tamerius, J. Shaman, W. J. Alonso, K. Bloom-Feshbach, C. K. Uejio, A. Comrie, and C. Viboud. Environmental predictors of seasonal influenza epidemics across temperate and tropical climates. *PLoSPathog*, 9(3):68–72, 2013.

[70] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of 37th Annual Allerton Conference on Communication, Control and Computing*, 1999.

[71] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *Proc. of ITW'15*, 2015.

[72] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[73] T. Weissman, E. Ordentlich, et al. Universal discrete denoising: Known channel. *IEEE Tran. on Information Theory*, 51(1):5–28, 2005.

[74] T. Weissman, E. Ordentlich, et al. Universal filtering via prediction. *IEEE Tran. on Information Theory*, 53(4):1253–1264, 2007.

[75] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems (NIPS'12)*, 2012.

[76] X. Yang et al. Forecasting chinese tourist volume with serach engine data. *Tourism Management*, 46:386–397, 2015.

[77] Q. Yuan, E. O. Nsoesie, B. Lv, G. Peng, R. Chunara, and J. S. Brownstein. Monitoring influenza epidemics in china with search query from baidu. *PlosOne*, 8(5):e64323, 2013.

[78] R. Zhang and T. Weissman. Discrete denoising for channels with memory. *Communications in Information and Systems*, 5(2):257–288, 2005.