A survey of Time Series Methods

Bassem Monla
basmonla@ca.ibm.com

May 28th 2020 – Finance Montreal

Agenda

I. Types & characteristics of Time Series

II. Tasks for Time Series

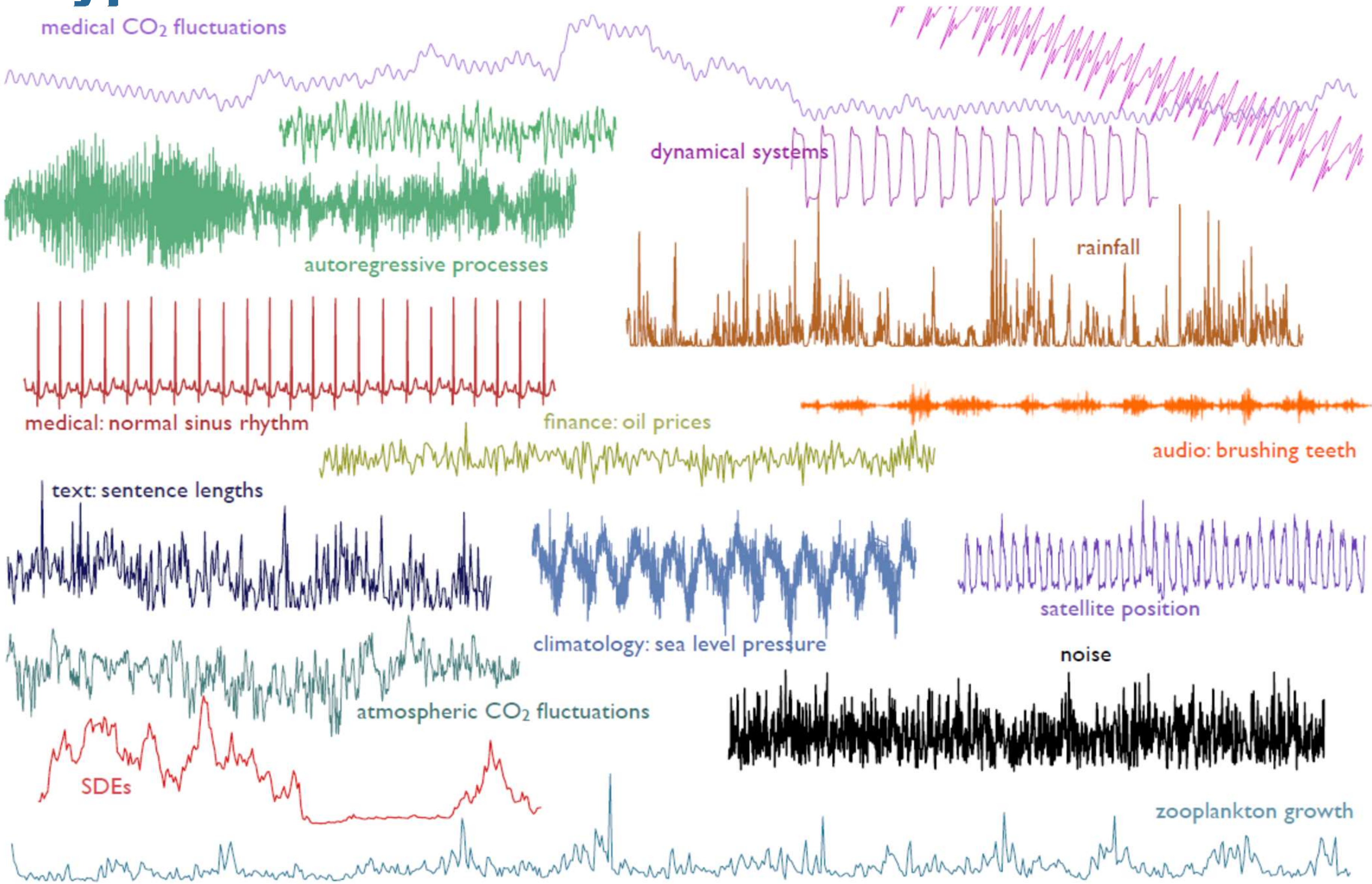
III. Survey of Time Series Techniques

   a) Stochastic Process Methods

   b) State Space Methods

   c) Neural Network Methods

Types of Time Series



Championing Artificial Intelligence (AI) practices into your Business & IT

Tasks for time series

1. Classify time series
2. Cluster a time series
3. Regime change, change point detection
4. Segmentation or summarization
5. Decompose a time series into sums or products of time series, curve fitting and function approximation
6. Detect anomalie
7. Forecast next value (Filtering)
8. Forecast next values after n times in the future
9. Find missing value from the past (smoothing)
10. Detect causalities between time series
11. Find and fit best model
12. Generate new similar time series
13. Simulate time series based on inputs
14. Query by content
15. Denoising
16. Survival Analysis to detect when an event would happen
17. Optimise decisions based on forecastings

Time line of time series methods

1) 1960 box and Jenkins ARIMA methods
2) 1980 GARCH Models
3) 1990 State Space Models
4) 1990 Linear Gaussion State Space Models and Kalman filtering
5) 2000 extension of Kalman Filtering
6) 2005 Regression methods : Dimensionality reduction with SVD, USVT, Matrix Methods, and SVM, RF, Boosting, Random Forests, XGBoosts, etc …
7) 2010 Particle filtering methods
8) 2015 Recurrent Neural Network & Convolutional Networks models
9) 2017 Attention Models
10) 2019 Transformers Networks

AutoRegressive Integrated Moving Average (ARIMA) model

ARIMA models are widely used in econometrics.
These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values.

The MA part indicates that the regression error is actually a linear combination of error terms.

The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values

ARIMA models are generally denoted ARIMA(p,d,q) where parameters p, d, and q are non-negative integers, p is the order (number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average mode.

Autoregressive integrated moving average (ARIMA) model (and many extensions exist)

Given a time series data Xt where t is an integer index and the Xt are real numbers, an ARMA(p',q) model is given by

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_{p'} X_{t-p'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

1. ARIMA models borrow much of its properties from LTS (Linear Time Invariant Systems) theory. A very well known and understood theory applied in signal processing.
2. An autoregressive model can thus be viewed as the output of an all-pole infinite impulse response filter whose input is white noise.
3. These are stochastic process

4. Some implementations include: the "statsmodels" package in Python and Automatic ARIMA with SPSS.
5. These are Linear Models , the integration is supposed to make them stationary, they have a short memory, they are black boxes models.
6. Data transformation might be required to make linear model fit.

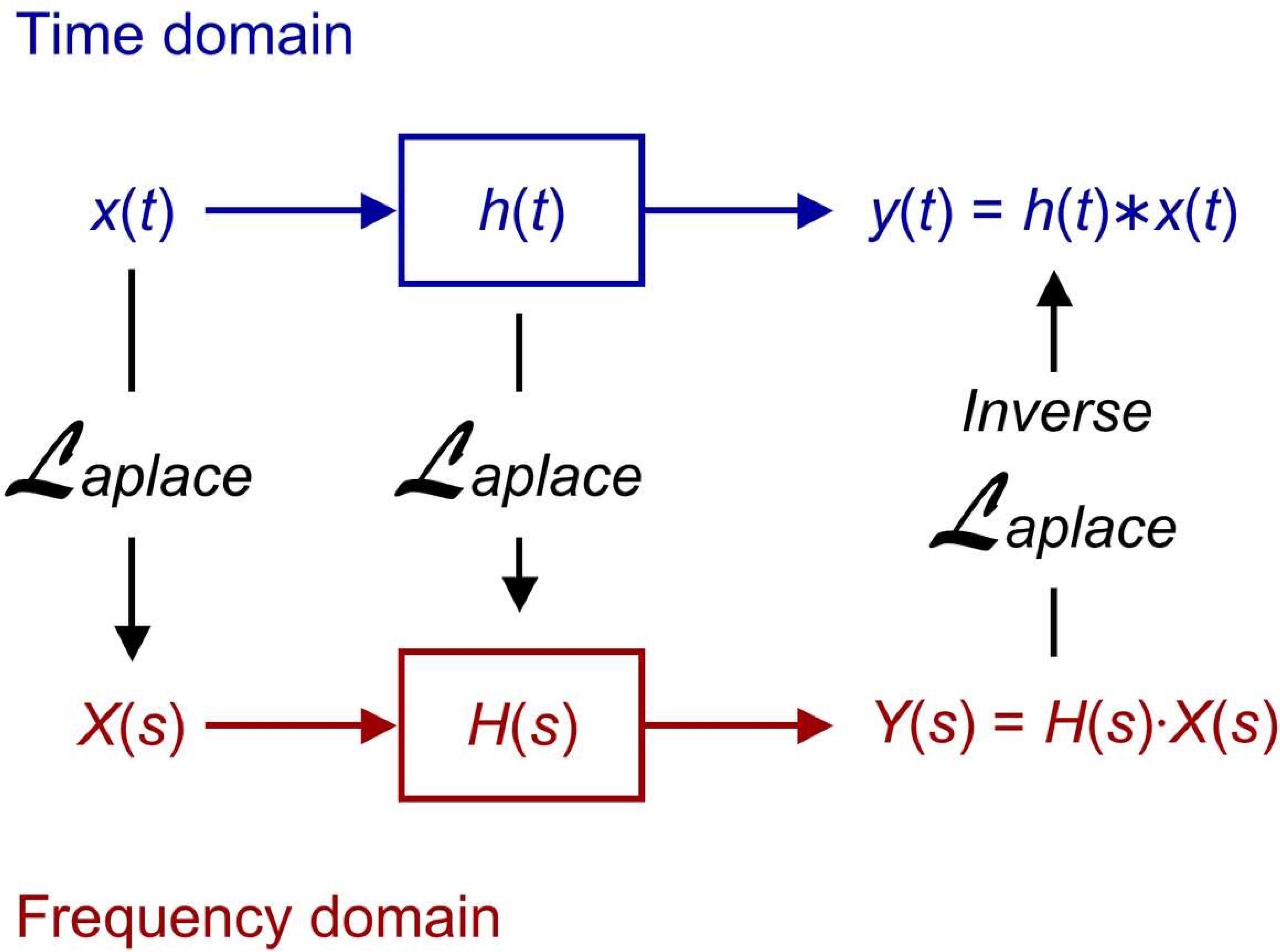
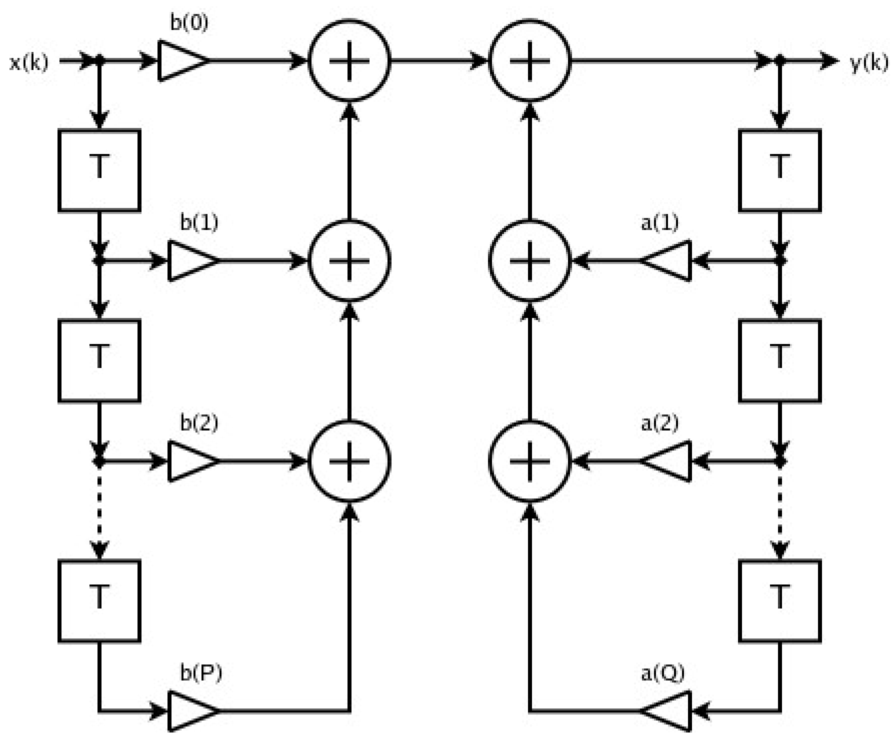
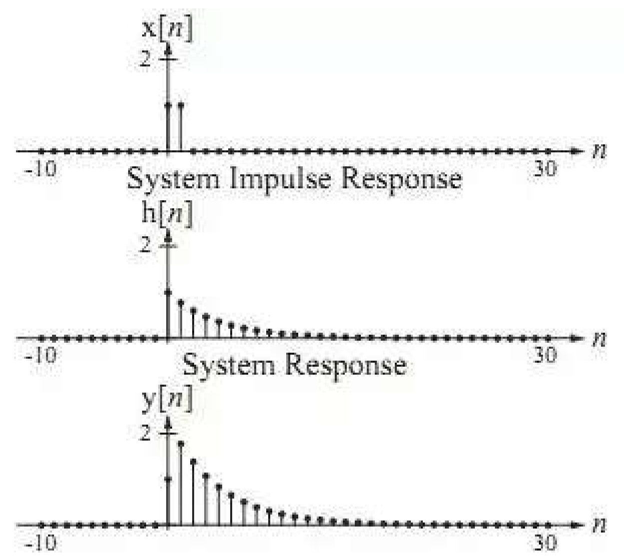
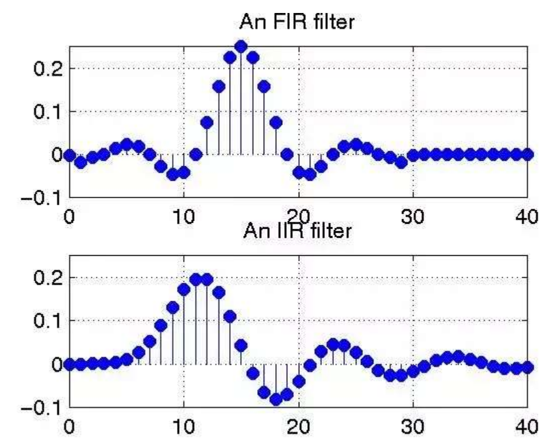

Time domain
x(t)
h(t)
y(t) = h(t)∗x(t)
Laplace
Laplace
Inverse
Laplace
X(s)
H(s)
Y(s) = H(s)·X(s)
Frequency domain

Illustration of Infinite and Finite response filters





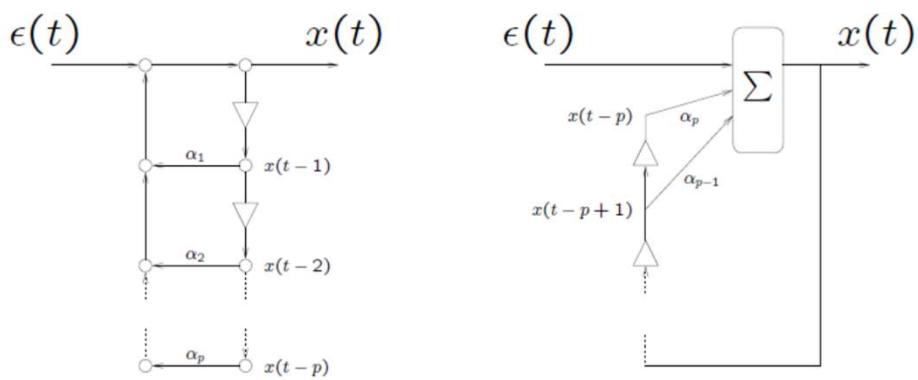


Relation between LTI Systems (FIR & IIR) , ARIMA & Neural Network theories

Linear DSP Models as Linear NNs

DSP Filter	DSP Model	NN Connections
FIR	MA[q]	feedforward
IIR	AR[p]	recurrent

An AR[p] model is equivalent to:



Exponential Smoothing

Exponential smoothing is a technique for smoothing time series data using the exponential window function.

$$s_0 = x_0$$
$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}, \; t > 0$$

where α is the *smoothing factor*, and $0 < \alpha < 1$.

Exponential smoothing is equivalent to a first-order Infinite Impulse Response or IIR filter applied to the raw data.

The AutoRegressive Conditional Heteroscedasticity (ARCH) model

The autoregressive conditional heteroscedasticity (ARCH) model is a statistical model for time series data that describes the variance of the current error term or innovation as a function of the actual sizes of the previous time periods' error terms;

Often the variance is related to the squares of the previous innovations. The ARCH model is appropriate when the error variance in a time series follows an autoregressive (AR) model;

If an autoregressive moving average (ARMA) model is assumed for the error variance, the model is a generalized autoregressive conditional heteroskedasticity (GARCH) model

The AutoRegressive Conditional Heteroscedasticity (ARCH) model

The ARCH models are commonly employed in modeling financial time series that exhibit time-varying volatility and volatility clustering, i.e. periods of swings interspersed with periods of relative calm.

To model a time series using an ARCH process, let ϵ_t denote the error terms (return residuals, with respect to a mean process), i.e. the series terms. These ϵ_t are split into a stochastic piece z_t and a time-dependent standard deviation σ_t characterizing the typical size of the terms so that

$$\epsilon_t = \sigma_t z_t$$

The random variable z_t is a strong white noise process. The series σ_t^2 is modeled by

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2,$$

If an autoregressive moving average model (ARMA) model is assumed for the error variance, the model is a generalized autoregressive conditional heteroskedasticity (GARCH) model

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$$

State Space representation and modeling

State space modelling provides a unified methodology for treating a wide range of problems in time series analysis.

In this approach it is assumed that the development over time of the system under study is determined by an unobserved series of vectors with which are associated a series of observations

The main purpose of state space analysis is to infer the relevant properties of the unobserved series of vectors from a knowledge of the series of observations.

Structural time series models (models in which the observations are made up of trend seasonal, cycle and regression components plus error) can be put into state space form.

Computational algorithms in state space analyses are mainly based on recursions, that is, formulae in which we calculate the value at time $t+1$ from earlier values for $t, t-1, \ldots, 1$.

Linear Gaussian State Space models

ARIMA models can be put into state space form. They are special cases of state space models.

Exponential smoothing relates to simple forms of state space and ARIMA models.

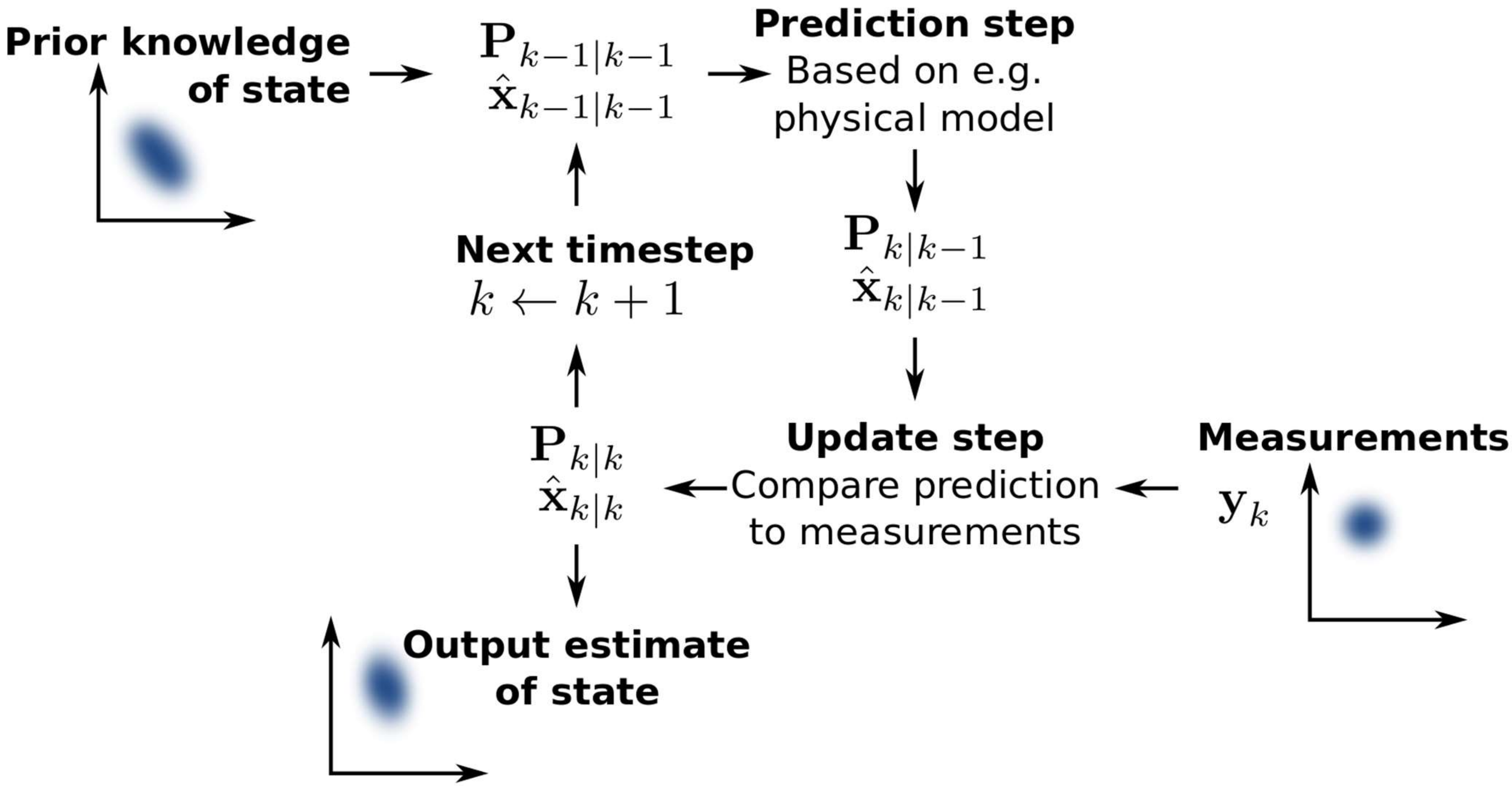
The general linear Gaussian state space model can be written in the following form:

$$\begin{aligned} y_t &= Z_t \alpha_t + \varepsilon_t, & \varepsilon_t &\sim \mathrm{N}(0, H_t), & \\ \alpha_{t+1} &= T_t \alpha_t + R_t \eta_t, & \eta_t &\sim \mathrm{N}(0, Q_t), & t = 1, \ldots, n, \end{aligned}$$

a very wide class of problems could be encapsulated in a simple linear model, essentially the state space model.

The key advantage of the state space approach is that it is based on a structural analysis of the problem. The different components that make up the series, such as trend, seasonal, cycle and calendar variations, together with the effects of explanatory variables and interventions, are modelled separately before being put together in the state space model

Advanced time series: Kalman filtering



The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

Linear Gaussian State Space models: Kalman filtering

Additional remarks:

1. Limited to Linear Gaussian Models

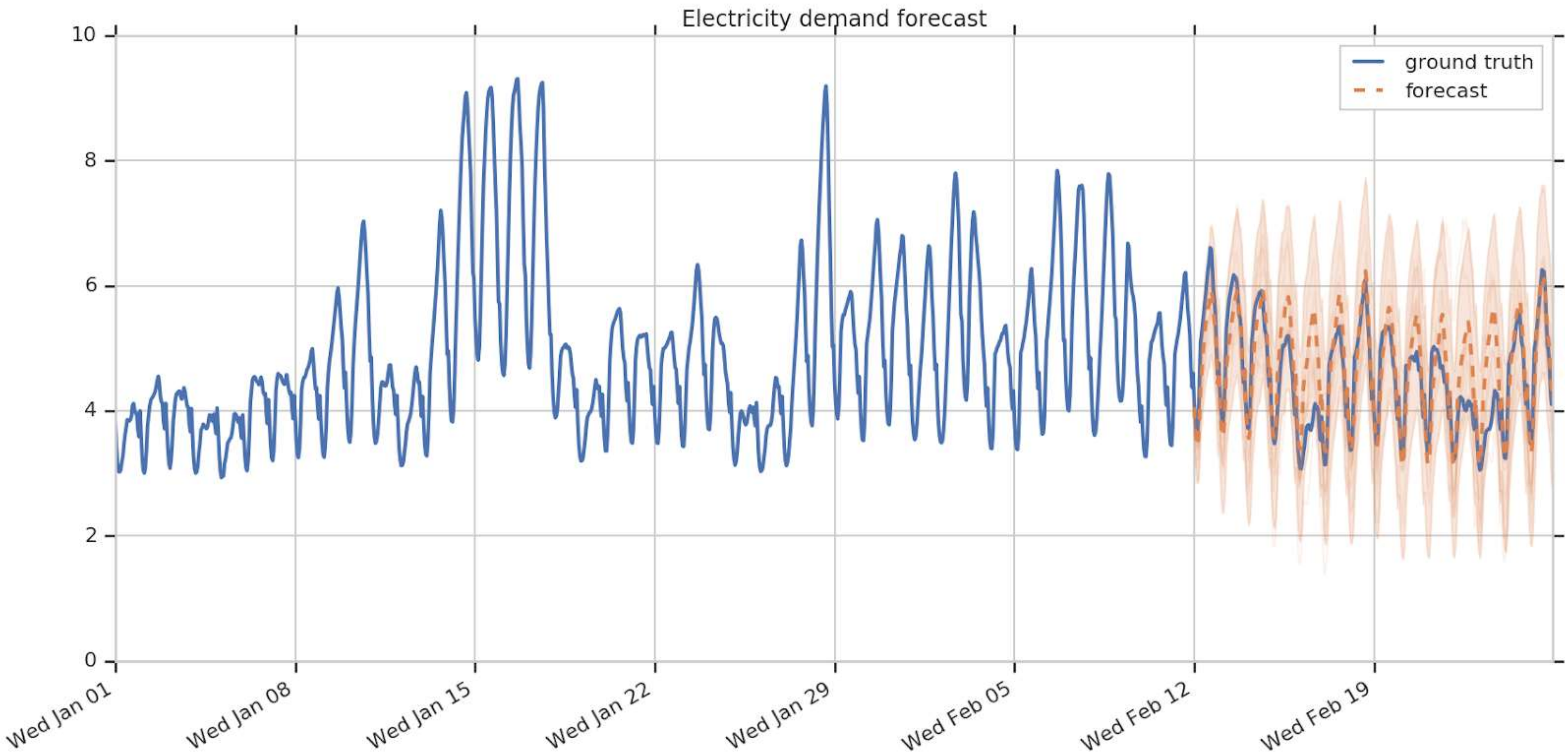
2. The algorithm is recursive.

3. Learns recursively and adapts its parameters constantly, so it is not time independent as in ARIMA

4. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

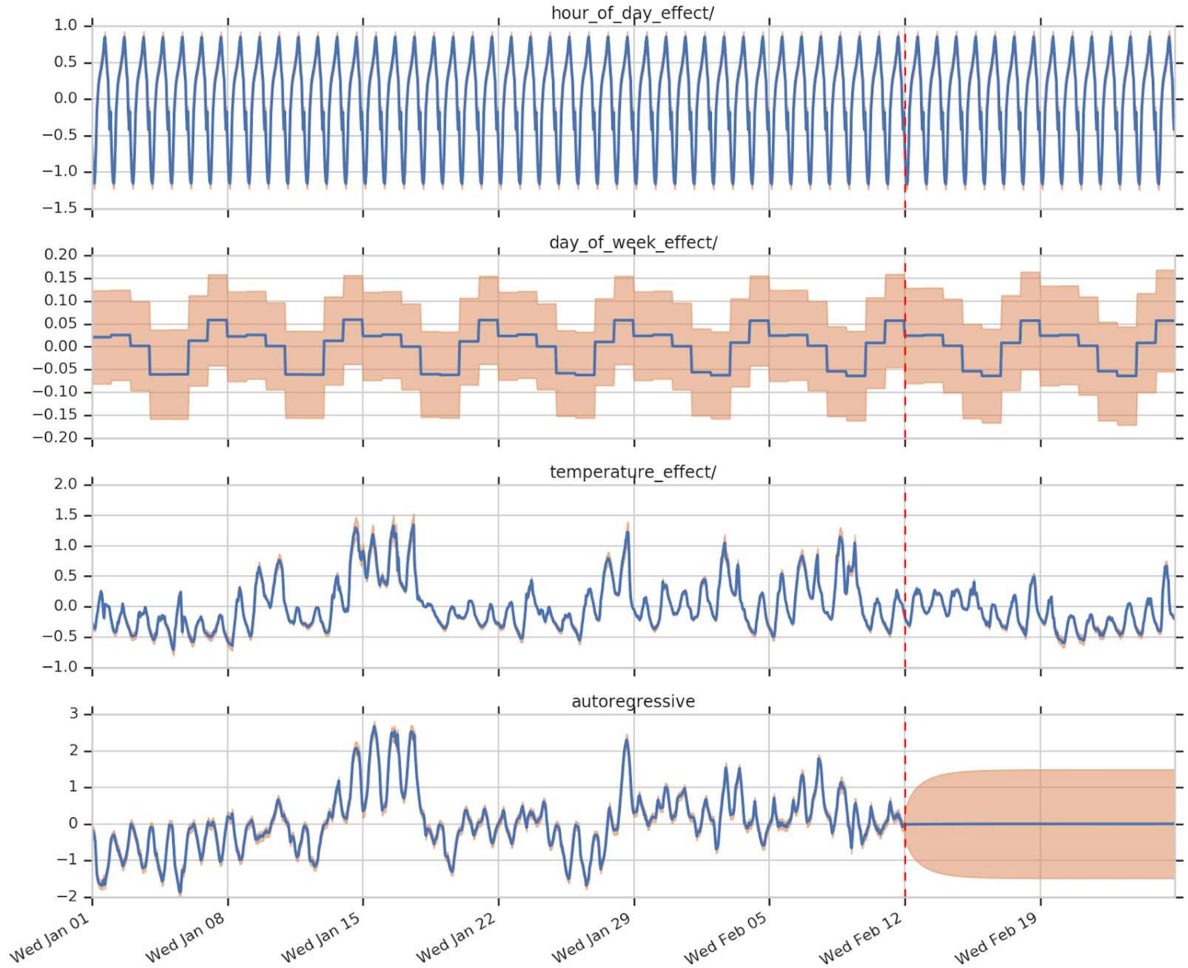
5. Extensions to Kalman filtering (Non Linear & Non Gaussian) are a process of linearization of the functions around the estimation point.

Structural Time Series modeling in TensorFlow Probability



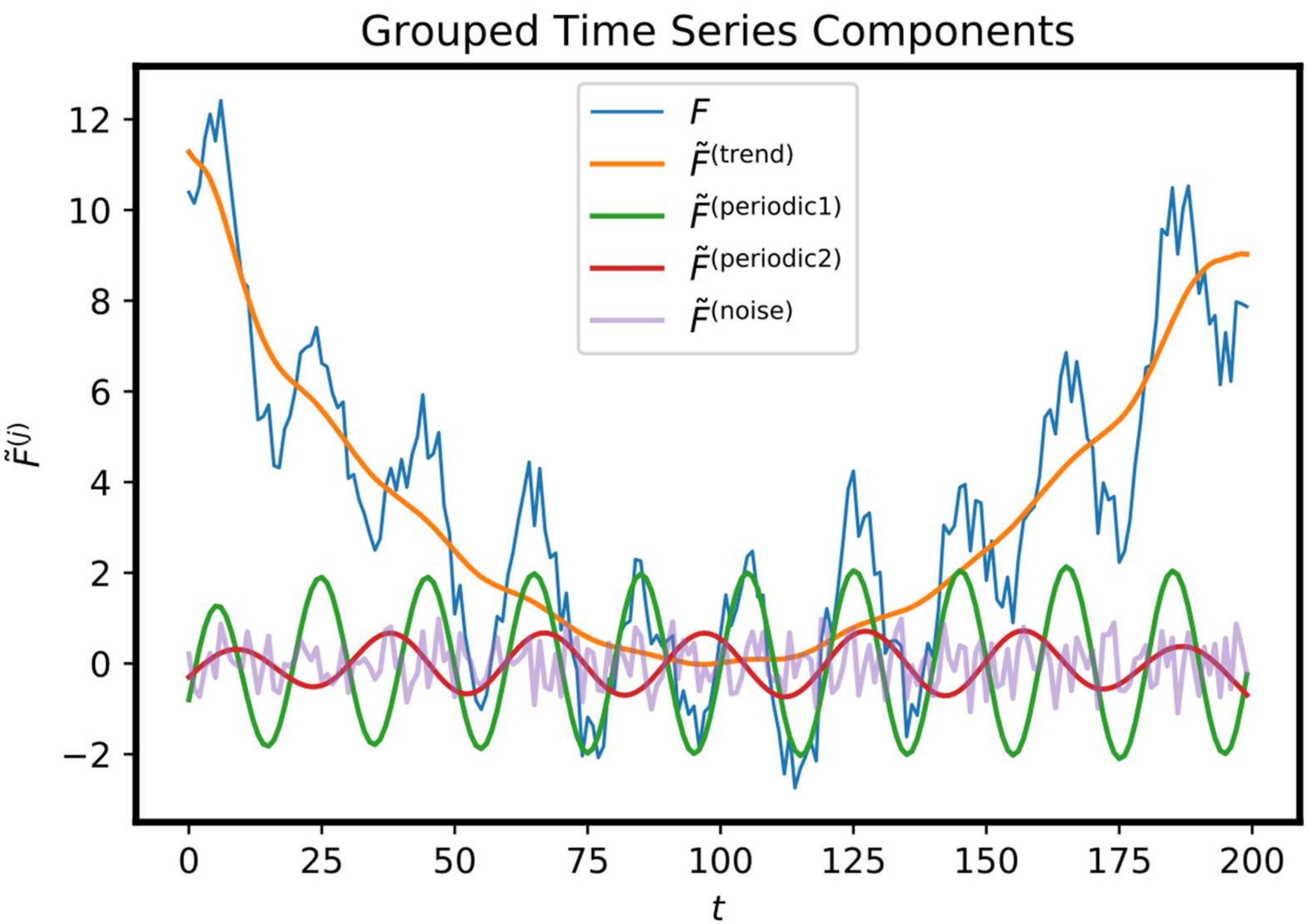
https://blog.tensorflow.org/2019/03/structural-time-series-modeling-in.html

Structural Time Series modeling in TensorFlow Probability



https://blog.tensorflow.org/2019/03/structural-time-series-modeling-in.html

Singular Spectrum Analysis



Singular spectrum analysis applied to a time-series *F*, with reconstructed components grouped into trend, oscillations, and noise

Transform a time series model to a classical Machine learning Model with State Space

Time Series Decomposition and Feature Engineering to discover the hidden State Space Variables by multiple techniques:

1) STL decomposition of time series data (Seasonal and Trend decomposition using Loess).
2) Singular Spectrum Analysis for time series decomposition
3) Matrix Methods : Model Agnostic Time Series Analysis via Matrix Estimation with Universal Singular Value Thresholding

The way to proceed:

1) Time variable meaning and embeddings
2) Combine with other time series
3) Lags on the current time series and lags other time series with additional transformations on all these variables
4) Apply classical Machine Learning methods on the tabular data

Advanced time series: Recurrent Neural Network

1) Requires lot of training data

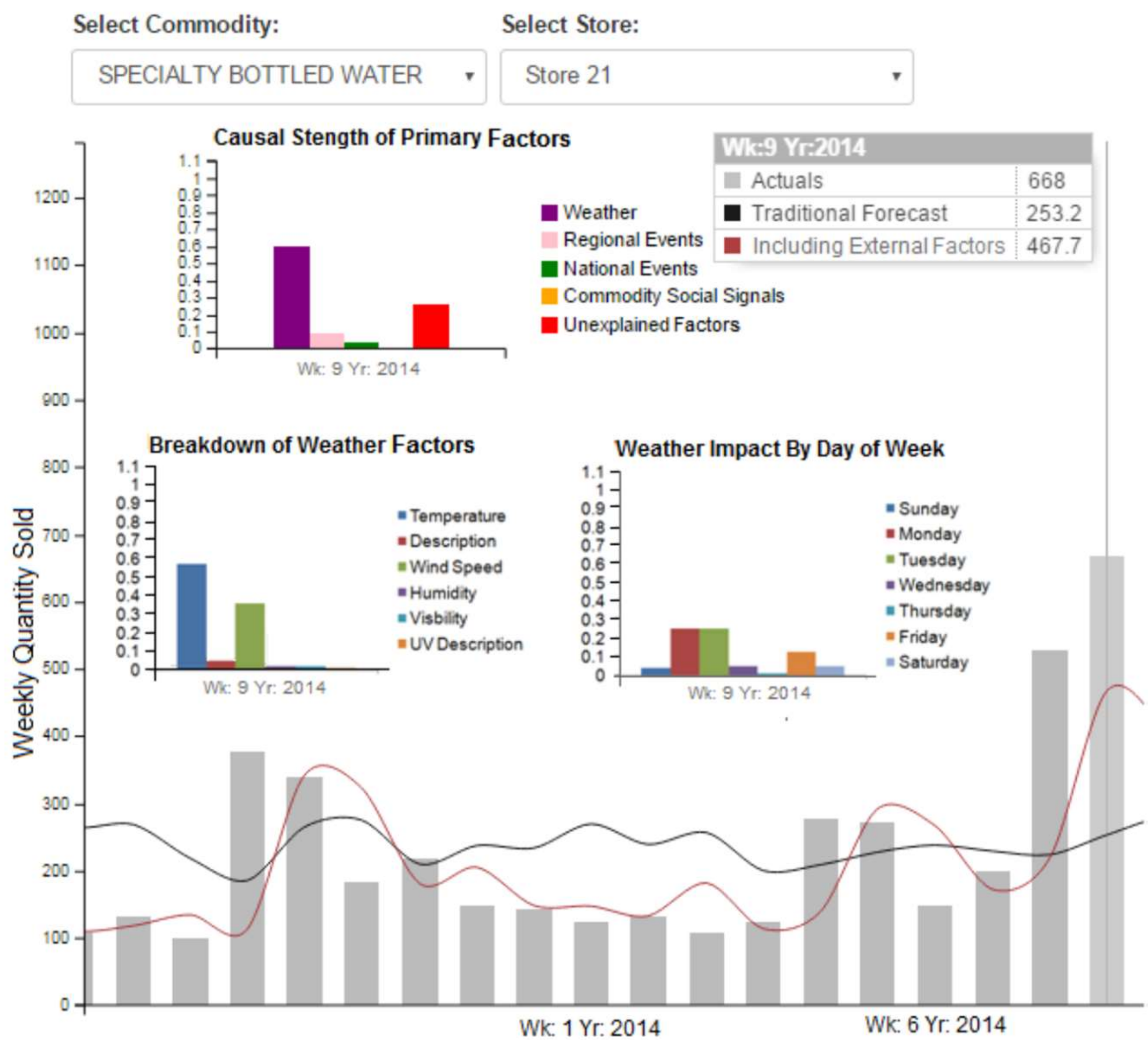
2) Models are extremely rich but are very difficult to train

3) One major advantage is that they extract knowledge from other time series

4) Memory Recurrent Neural Networks like GRU and LSTM still fail to capture long term dependency

5) Transformer Architecture with Attention Mechanisms are extremely promising but they are still in research phase.

Advanced time series: Attention Mechanism



Championing Artificial Intelligence (AI) practices into your Business & IT

Advanced time series: Attention Mechanism