# Tracking the tracker:

## Time Series Analysis in Python From First Principles




Kenneth Emeka Odoh


**PyCon APAC**

@**National University of Singapore**
Computing 1 (COM1) / Level 2
13 Computing Drive Singapore 117417

May **31st**, 2018 - June **2nd**, 2018

# Talk Plan

- Bio
- Motivations & Goals
- Theory
- Application: Forecasting
- Application: Anomaly Detection
- Deployment
- Lessons
- Conclusion

# Bio

## Education

- Masters in Computer Science (University of Regina) 2013 - 2016
- A number of MOOCs in statistics, algorithm, and machine learning

## Work

- Research assistant in the field of Visual Analytics 2013 - 2016
  - ( https://scholar.google.com/citations?user=3G2ncgwAAAAJ&hl=en )
- Software Engineer in Vancouver, Canada 2017 -
- Regular speaker at a number of Vancouver AI meetups, organizer of distributed systems meetup, and Haskell study group and organizer of Applied Cryptography study group 2017 -

This talk focuses on two applications in time series analysis:

- **Forecasting**
  - Can I use the past to predict the future?

- **Anomaly Detection**
  - Is a given point, normal or abnormal?

"Simplicity is the ultimate sophistication" --- Leonardo da Vinci

# Motivation

Figure 1: Pole vault showing time series of position
[http://vaultermagazine.com/middle-school-womens-pole-vault-rankings-from-athletic-net-5/]

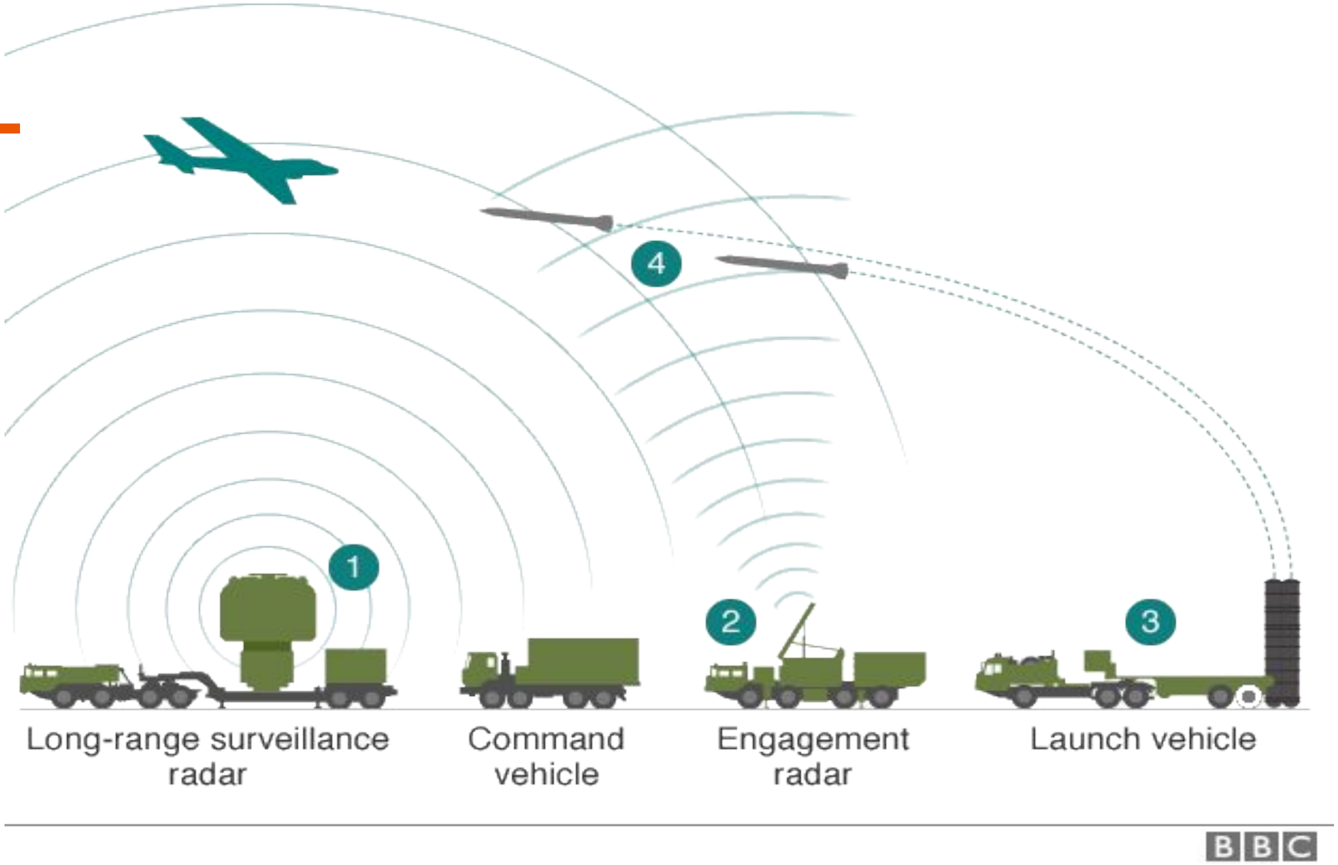



Figure 2: Missile Defence System
[http://www.bbc.com/news/world-middle-east-43730068]


Kenneth Emeka Odoh

Figure 3: Bitcoin time series [yahoo finance]

# Goals

- A real-time model for time series analysis.
- A less computational intensive model thereby resulting in scalability.
- A support for multivariate / univariate time series data.

introducing

**pySmooth**

( https://github.com/kenluck2001/pySmooth)

and

**Anomaly**

( https://github.com/kenluck2001/anomaly)

# Application: Forecasting

# Let’s dive into the maths for some minutes

[http://www.boom-studios.com/2017/12/01/20th-century-fox-buys-girl-detective-adventure-movie-goldie-vance/]

## Theory

- **Time Series Data**
  - $Y_t$ where $t$=1,2,3, ..., $n$ where t is index and n is number of points.

- **Time Series Decomposition**

$$Y_t = V_t + S_t + H_t + \xi_t, \quad t = 1, ..., n$$

Sometimes the relation may be in multiplicative form

$$Y_t = V_t * S_t * H_t * \xi_t, \quad t = 1, ..., n$$

Where $Y_t$ is predicted time series, $V_t$ is trend, $S_t$ is cyclic component, $H_t$ is holiday component, and $\xi_t$ is residual.

- **pth-Order Difference Equation**, Autoregressive, AR(p)

$$AR(p) = \mathbf{\Theta}_1 Y_{t-1} + \mathbf{\Theta}_2 Y_{t-2} + \ldots + \mathbf{\Theta}_p Y_{t-p} + \boldsymbol{\varepsilon}_t \text{, for } i = 1, \ldots, p$$

Where $\mathbf{\Theta}$ is parameter, and $\boldsymbol{\varepsilon}_t$ is noise respectively. See **EWMA** for anomaly detection as an example of AR.

- **Moving Average Process**, MA(q)

$$MA(q) = \mathbf{\Phi}_1 \boldsymbol{\varepsilon}_{t-1} + \mathbf{\Phi}_2 \boldsymbol{\varepsilon}_{t-2} + \ldots + \mathbf{\Phi}_q \boldsymbol{\varepsilon}_{t-q} \text{, for } i = 1, \ldots, q$$

Where $\mathbf{\Phi}$ is parameter, and $\boldsymbol{\varepsilon}_t$ is noise respectively.

- **Autoregressive Moving Average**, ARMA

$$ARMA(p,q) = \Theta_1 Y_{t-1} + \Theta_2 Y_{t-2} + \ldots + \Theta_p Y_{t-p} + \varepsilon_t + \Phi_1 \varepsilon_{t-1} + \Phi_2 \varepsilon_{t-2} + \ldots + \Phi_q \varepsilon_{t-q}$$

Where p, q, $\Theta$, and $\Phi$ are parameters respectively.

- **Autoregressive Integrated Moving Average**, ARIMA

ARIMA(p, d, q) = ARMA(p+d,q)

See Box-Jenkins methodology for choosing good parameters.

- ARCH
- GARCH ( related to exponential weighted moving average ). See section on anomaly detection.
- NGARCH
- IGARCH
- EGARCH

...
For more **information,**
see link ( https://en.wikipedia.org/wiki/Autoregressive_conditional_heteroskedasticity )

For more **practical demonstration,** see
(https://www.kaggle.com/jagangupta/time-series-basics-exploring-traditional-ts)

# Why Kalman Filters?

- Allows for using **measurement** signals to augment the prediction of your next **states**. **State** is the target variable.
- Allows for correction using the correct state from the **recent past** step.
- Allows for **online** prediction in real time.

It is good to note the following:

- **How to identify the appropriate measurement signals?**
- **Measurement** captures the **trend** of the **state**.
- Modeling can include **domain knowledge** e.g **physics**.

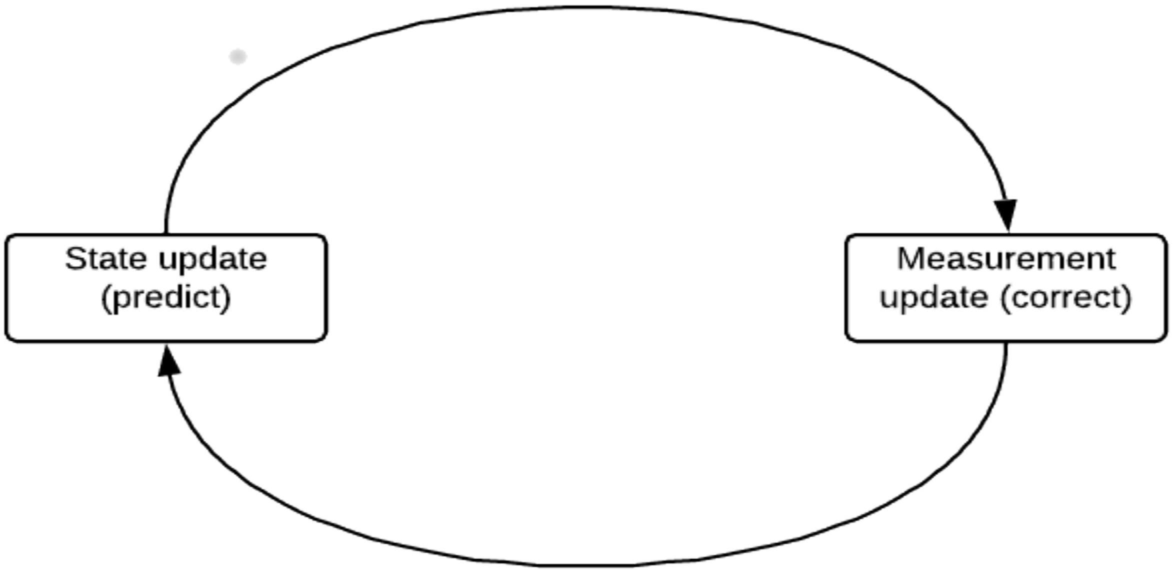



Figure 4: How kalman filter works

- **Kalman filter**

$$\mathbf{x}_k = F(\mathbf{x}_{k-1}, \mathbf{w}) + B \cdot v_{k-1}$$

$$\begin{bmatrix} x_k \\ x_{k-1} \\ \vdots \\ x_{k-M+1} \end{bmatrix} = \begin{bmatrix} f(x_{k-1}, \ldots, x_{k-M}, \mathbf{w}) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{k-1} \\ \vdots \\ x_{k-M} \end{bmatrix} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdot v_{k-1}$$

$$y_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \cdot \mathbf{x}_k + n_k$$

Where x, y, F,n, v are states, measurement, function, measurement noise, and state noise respectively.

Kalman formulation allows for

- Handling missing data.
- Data fusion

It is a **state space** model.

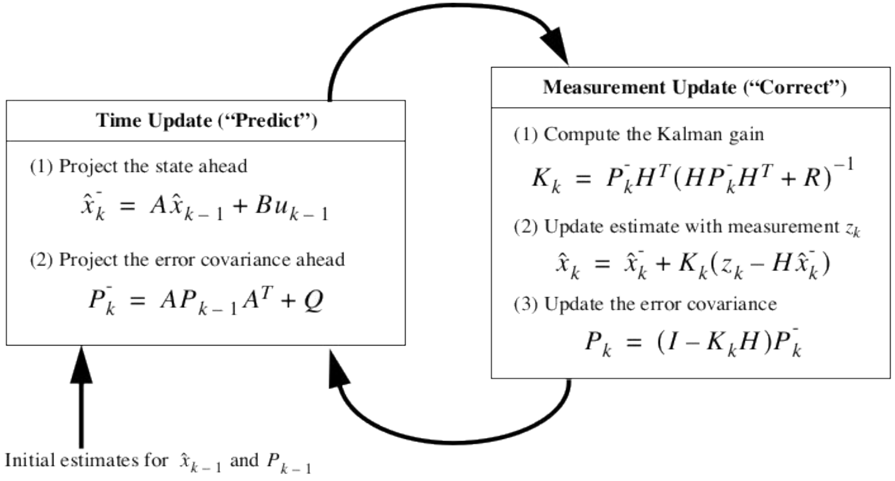
**Continuous** form of the forward algorithm of an **HMM** [Andrew Fraser, 2008].

[Julier, 2002]

# Discrete Kalman Filter (DKF)

- Capture linear relationship in the data.
- Fluctuates around the means and covariance.
- Eventual convergence as the data increases, thereby improving filter performance ( **bayesian** ).

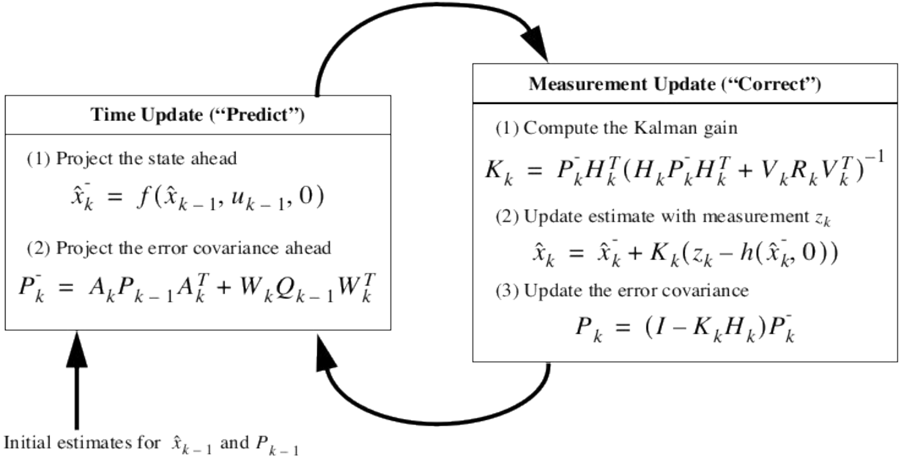
In the presence of nonlinear relationship . The increased **error** in the posterior estimates , thereby leading to suboptimal filter performance ( **underfitting** ).




[Welch & Bishop, 2006]

# Extended Kalman Filter (EKF)

- Makes use of Jacobians and hessians.
  - Increase computation cost.
  - Need for **differentiable** non-linear function
- Linearizing nonlinear equation using taylor series to **1st order**.
- Same computational complexity as Unscented kalman filter.
- Captures nonlinear relationship in the data.




The limited order of taylor series can lead to **error** in the posterior estimates , thereby leading to suboptimal filter performance.

[Welch & Bishop, 2006]

# Unscented Kalman Filter (UKF)

- Better sampling to capture more **representative** characteristics of the model.
  - By using **sigma** points.
  - **Sigma** point are extra data points that are chosen within the region surrounding the original data. This helps to account for variability by capturing the likely position that data could be given some **perturbation**. Sampling these points provides richer information of the distribution of the data. It is more **ergodic**.

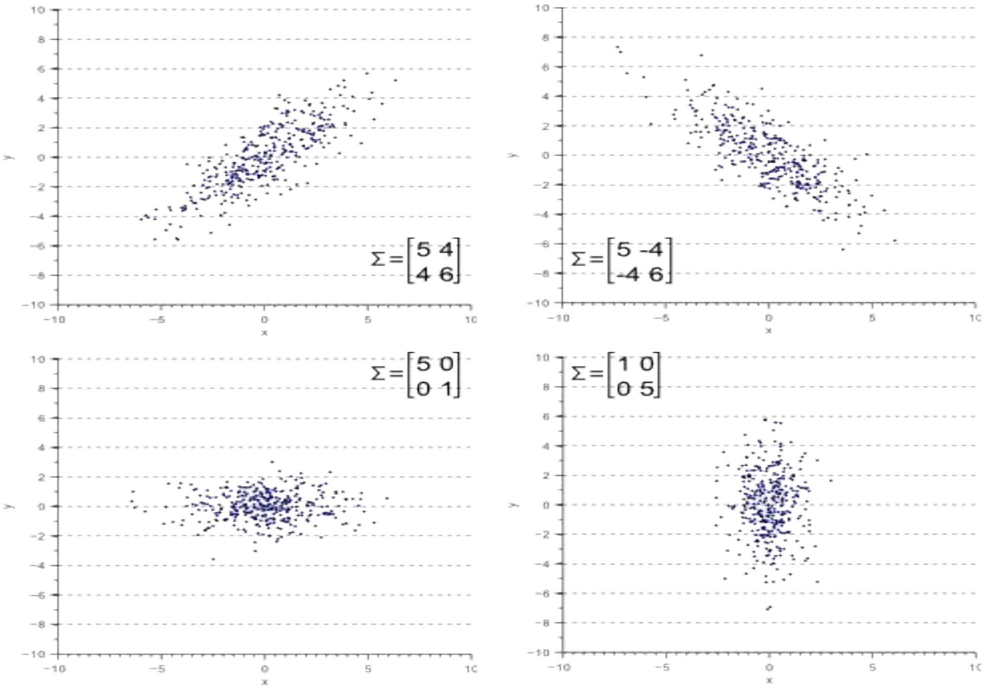
- Avoid the use of **jacobians** and **hessians**.
  - Possible to use any form of **nonlinear** function ( not only differentiable function ).

- Linearizing nonlinear function using taylor series to **3rd order**.

- Same computational efficiency as the **EKF**.



[Julier, 2002]




For more info,
[http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/]



Figure 5: Geometric interpretation of covariance matrix

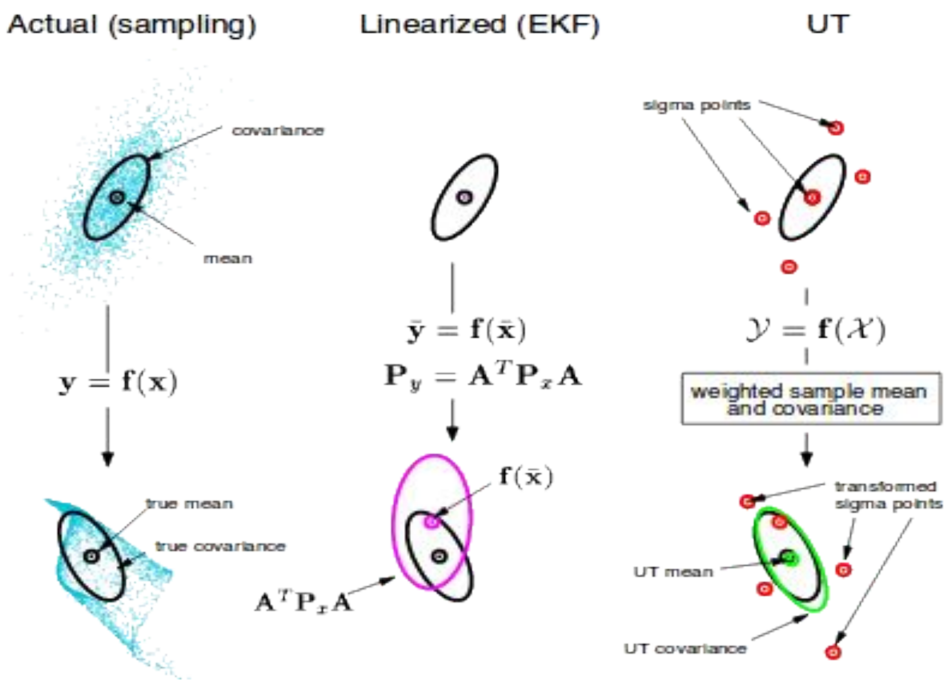



Figure 6: Actual sampling vs extended kalman filter, and unscented kalman filters

[Julier, 2002]

# Another kind of Kalman filters

- Particle filtering

See more tweaks that can be made in the Kalman filter formulation

{ https://users.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf}

## ● **RLS** (Recursive Linear Regression)

Initial model at time, t with an update as new data arrives at time t+1.

$y = \theta(t)^* x_t$

At time t+1, we have data, $x_{t+1}$ and $y_{t+1}$ and estimate $\theta$(t+1) in incremental manner

**Matrix Inversion Lemma RLS, version 1**

At time step $t+1$

1. Form x(t + 1) using the new data.
2. Form $\varepsilon(t+1) = y(t+1) - x^T(t+1)\hat{\theta}(t)$
3. Form $P(t+1) = P(t)\left[ I_m - \dfrac{x(t+1)x^T(t+1)P(t)}{1 + x^T(t+1)P(t)x(t+1)} \right]$
4. Update $\hat{\theta}(t+1) = \hat{\theta}(t) + P(t+1)x(t+1)\varepsilon(t+1)$
5. Loop back to step (1).

[Sherman and Morrison formula]
https://en.wikipedia.org/wiki/Sherman%E2%80%93Morrison_formula

[See closed form of $\theta = (X^T X)^{-1} X^T Y$ ]

Kenneth Emeka Odoh

[Wellstead & Karl, 1991]

## Online ARIMA

# RLS

# +

# ‘Vanilla’ ARIMA



[Kenneth Odoh]

# Box-Jenkins Methodology

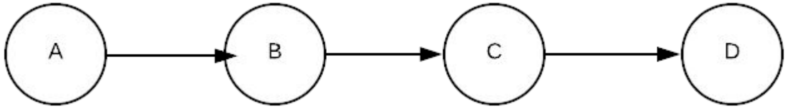
- Transform data so that a form of stationarity is attained.
  - E.g taking logarithm of the data, or other forms of scaling.
- Guess values p, d, and q respectively for ARIMA (p, d, q).
- Estimate the parameters needed for future prediction.
- Perform diagnostics using AIC, BIC, or REC (Regression Error Characteristic Curves) [Jinbo & Kristin, 2003].

Probably, a generalization of Occam's Razor


[https://www.tass.gov.uk/about/facts-figures/]

# Stationarity

- Constant statistical properties over time
  - Joint probability distribution is shift-invariant
  - Can occur in different moments (mean, variance, skewness, and kurtosis)
- **Why is it useful?**
- Mean reverting.
- Markovian assumption
  - Future is independent of the past given the present




With states (A, B, C, D), then a sequence of states occurring in a sequence

$$P(A, B, C, D) = \mathbf{P(D \mid A, B, C)} * \mathbf{P(C \mid A, B)} * P(B \mid A) * P(A)$$

$$= \mathbf{P(D \mid C)} * \mathbf{P(C \mid B)} * P(B \mid A) * P(A)$$

- n: number of states, m: number of observations
- S: set of states, Y: set of measurements
    - $S = \{s_1, s_2, \ldots, s_n\}, Y = \{y_1, y_2, \ldots, y_m\}$
    - $P_{S(1)} = [p_1, p_2, \ldots, p_n]$ -- prior probability
    - $T = P ( s(t+1) \mid s(t) )$ --- transition probability
    - $P_{S(t+1)} = P_{S(t)} * T$ --- stationarity
- Mapping states to measurement, $P ( y|s )$
- Makes it easy to compose more relationship using bayes theorem in Kalman filtering formulation (continuous states and measurements).

- There are methods for converting data from non-stationary to stationary
  ( **implicit** in algorithm formulation or **explicit** in data preprocessing )
  - Detrending
  - Differencing

- Avoid using algorithm based on stationarity assumption with non-stationary data.

- Testing for stationarity

  - Formally, use unit root test [https://faculty.washington.edu/ezivot/econ584/notes/unitroot.pdf]
  - **Informally,** compare means or other statistic measures between different disjoint batches.

For a fuller treatment on stationarity, see [http://www.phdeconomics.sssup.it/documents/Lesson4.pdf]

# Why stocks are hard to predict

- If **efficient market hypothesis** holds, then the stock times series, $X_t$ is i.i.d. This means that the past **cannot** be used to predict the future.

$$P(X_t \mid X_{t-1}) = P(X_t)$$

- Efficient market hypothesis
  - Stocks are always at the right price or value
  - Cannot profit by purchasing undervalued or overvalued asset.
  - Stock selection is not guaranteed to increase returns.
  - Profit can only be achieved by seeking out riskier investments.

In summary, it is "impossible to beat the market". **Very Controversial**

- "Essentially, all models are wrong, but some are useful." --- George Box
  - As such, there is still **hope** with prediction, albeit a faint one.
  - Avoid transient **spurious correlation**. (**Domain knowledge, causal inference, & statistical models** ).

# Other Things to Consider in your Model

- Outliers
- Collinearity
- Heteroscedasticity
- Underfitting vs overfitting

# API

- Time difference model
- Online ARIMA
- Discrete kalman filter
- Extended kalman filter
- Unscented kalman filter

# API: Online ARIMA

```
import numpy as np
from RecursiveARIMA import RecursiveARIMA

X = np.random.rand(10,5)

recArimaObj = RecursiveARIMA(p=6, d=0, q=6)

recArimaObj.init( X )

x = np.random.rand(1,5)

recArimaObj.update ( x )

print "Next state"
print recArimaObj.predict( )
```

# API: Discrete Kalman Filter

```
import numpy as np
from DiscreteKalmanFilter import DiscreteKalmanFilter

X = np.random.rand(2,2)
Z = np.random.rand(2,2)

dkf = DiscreteKalmanFilter()
dkf.init( X, Z ) #training phase

dkf.update( )

x = np.random.rand(1,2)
z = np.random.rand(1,2)

print "Next state"
print dkf.predict( x, z )
```

# API: Extended Kalman Filter

```
import numpy as np
from ExtendedKalmanFilter import ExtendedKalmanFilter

X = np.random.rand(2,2)
Z = np.random.rand(2,15)

dkf = ExtendedKalmanFilter()
dkf.init( X, Z ) #training phase

dkf.update( )

x = np.random.rand(1,2)
z = np.random.rand(1,15)

print "Next state"
print dkf.predict( x, z )
```

# API: Unscented Kalman filter

```
import numpy as np
from UnscentedKalmanFilter import UnscentedKalmanFilter

X = np.random.rand(2,2)
Z = np.random.rand(2,15)

dkf = UnscentedKalmanFilter()

dkf.init( X, Z ) #training phase
dkf.update( )

x = np.random.rand(1,2)
z = np.random.rand(1,15)

print "Next state"
print dkf.predict( x, z )
```

# Other Approaches

- LSTM / RNN
- HMM
- Traditional ML methods

**Popular libraries**: number of R packages, Facebook's prophet

# Application: Anomaly Detection

# Anomaly Detection

## It is ideal for unbalanced data set

- small number of negative samples and large number of positive samples or vice versa.

**What is a normal data?**

- This is the **holy grail** of a number of **anomaly detection methods**.

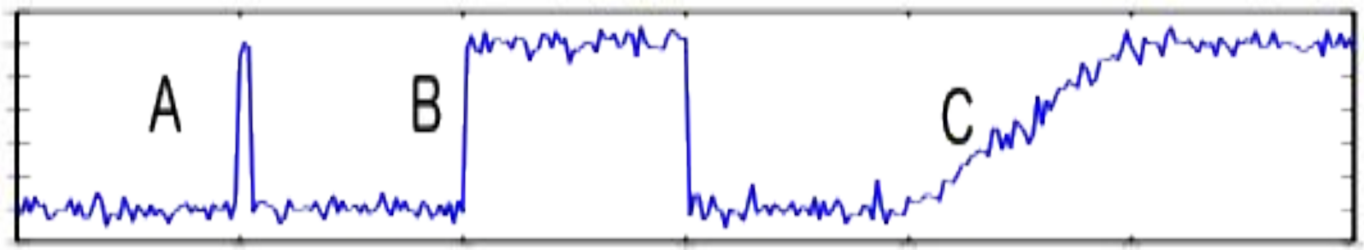



Figure 7: Types of signal changes: abrupt transient shift (**A**), abrupt distributional shift (**B**), and gradual distributional shift (**C**) [kelvin & william, 2012].

## Data Stream

- Time and space constraints.
- Online algorithms
    - Detecting concept drift.
    - Forgetting unnecessary history.
    - Revision of model after significant change in distribution.
- Time delay to prediction.

## Handling Data Stream

There are a number of window techniques :

- Fixed window
- Adaptive window (ADWIN)
- Landmark window
- Damped window

[Zhu & Shasha, 2002]

# Basic Primer on Statistics

Expected value, **E[X]** of X is the weighted average of the domain that X can take with each value weighted according to the probability of the event occuring. This is the long-run mean.

Variance , **Var[X]** is the squared of the standard deviation. This is the expectation of the squared deviation of X from its mean. This is a spread of the data.

$$Var[X] = E[X^2] - E^2[X]$$

Standard deviation , $\sigma$ ,is a measure of dispersion.

**Require:** $X_t, \hat{X}_t, \hat{\sigma}_t, T, t$
**Ensure:** $P_t, \hat{X}_{t+1}, \hat{\sigma}_{t+1}$

$Z_t \leftarrow \frac{X_t - \hat{X}_t}{\hat{\sigma}_t}$ # Z-score

$P_t \leftarrow \frac{1}{\sqrt{2\pi}} \exp(-\frac{Z_t^2}{2})$ # normal distribution pdf

**if** $t \leq T$ **then**

$\alpha_t \leftarrow 1 - 1/t$ # weighing factor

**else**

$\alpha_t \leftarrow (1 - \beta P_t)\alpha$ # weighing factor

**end if**

$s_1 \leftarrow \alpha_t s_1 + (1 - \alpha_t)X_t$ # Weighted Moving average

$s_2 \leftarrow \alpha_t s_2 + (1 - \alpha_t)X_t^2$ # Weighted Moving $E(X^2)$

$\hat{X}_{t+1} \leftarrow s_1$

$\hat{\sigma}_{t+1} \leftarrow \sqrt{s_2 - s_1^2}$ # Moving standard deviation

Algorithm 1: Probabilistic Exponential Moving Average [kelvin & william, 2012].

Kenneth Emeka Odoh

# Exponentially Weighted Moving Average (EWMA)

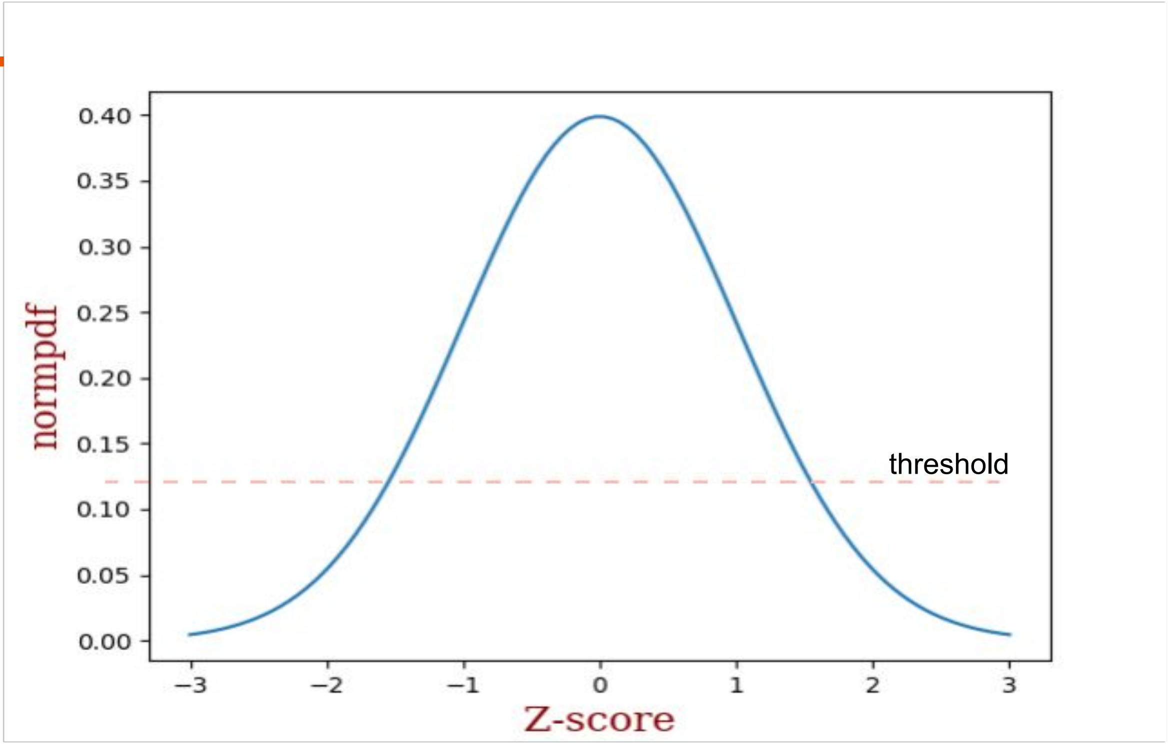
- Add a forgetting parameter to weight recent items more or less.
- Volatile to abrupt transient change and bad with distributional shifts.

$$\mu_t = \alpha \mu_{t-1} + (1 - \alpha) X_t$$

# Probabilistic Exponentially Weighted Moving Average (PEWMA)

- Retains the forgetting parameter of **EWMA** with an added extra parameter to include probability of data.
- Works with every kind of shift.

$$\mu_t = \alpha(1 - \beta P_t)\mu_{t-1} + (1 - \alpha(1 - \beta P_t))X_t$$




Kenneth Emeka Odoh

Figure 8: Normal Distribution Curve.

# Anomaly: A machine learning library for Anomaly detection

- A new library in Python named **anomaly**
  - ( https://github.com/kenluck2001/anomaly )
  - Written by Kenneth Emeka Odoh
- This make use of  a redis store to ensure persistence of summary parameters as new data arrives.
- This is capable of handling multiple data streams with time series data.
- Current implementation does not support **multivariate** time series.

# Exercise [ Group Activity ]

- Modify **Algorithm 1** to work for the multivariate  case?
- Customize threshold (Symmetric vs One-sided outlier)

# Deployment

This aims to show a streaming architecture for serving content.

The goal is that the deployment should not be **AWS specific** to prevent getting locked down in a proprietary vendor.

- Flask
- Redis
- Gevent
- Scikit-learn / keras / numpy
- PostgreSQL

# Machine learning as a service

- Web Server
  - Nginx (web server), Flask (lightweight web framework), and Gunicon (load balancer), Gevent (non-blocking I/O)
- Caching Server
  - Redis (as a transportation medium and caching)
- ML Server
  - Custom algorithm and third party libraries
- Database
  - PostgreSQL

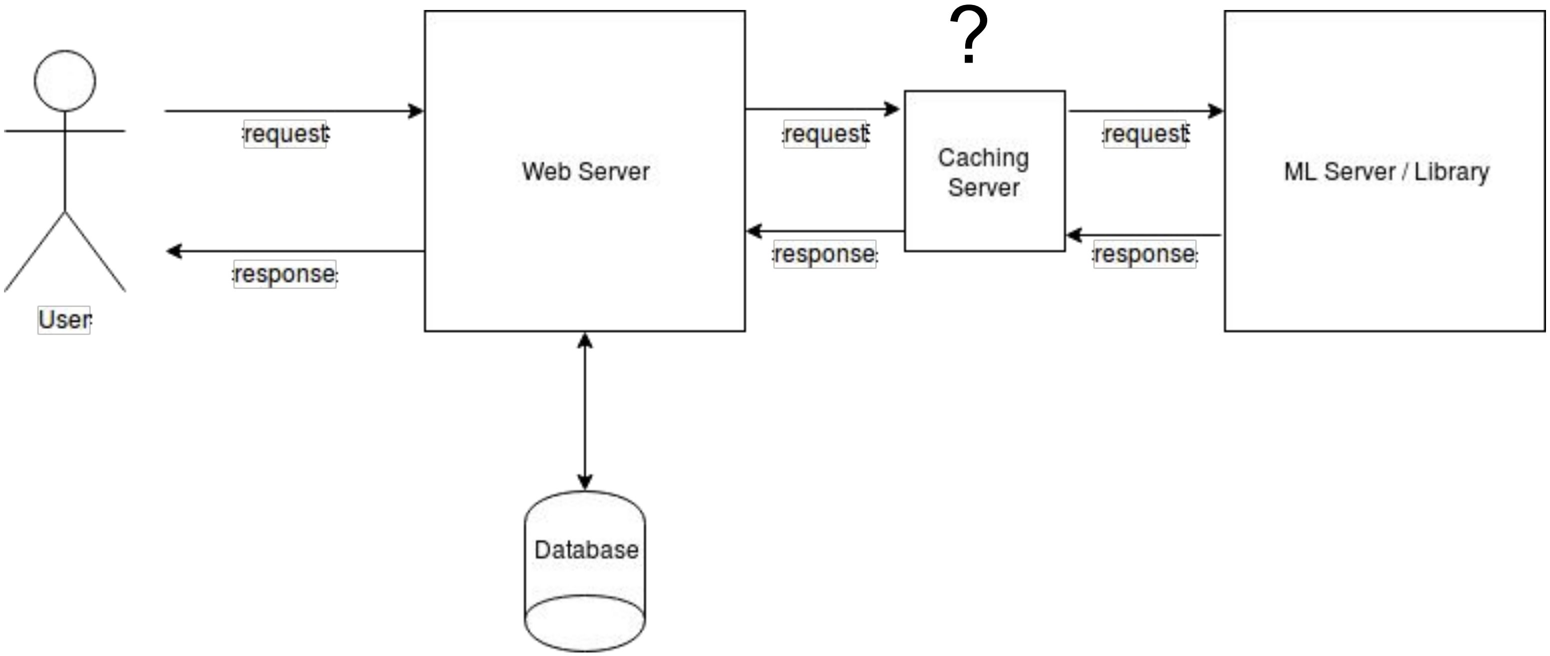





Figure 9: An example of a deployment architecture.

# Database

- Authentication
- metadata
- User management
    - Subscription
    - Managing limiting rates

# Other Considerations

- Rather than serve json in your rest API, think of using Google protobuf.
- Use streaming frameworks like Apache Kafka for building the pipeline.
- Flask is an easy to use web framework, think about using in **MVP.**
- Load balancer and replicated database.
- Make a Version control of your model to enhance reproducibility.
  - If necessary, map each prediction interval to a model.

# Lessons

- "Prediction is very difficult, especially if it's about the future." -- **Niels Bohr.**
- "If you make a great number of predictions, the ones that were wrong will soon be forgotten, and the ones that turn out to be true will make you famous." -- **Malcolm Gladwell.**
- "I have seen the future and it is very much like the present, only longer." -- **Kehlog Albran.**

# Conclusion

- Recursive ARIMA
    - Not ideal for chaotic or irregularly spaced time series.
- Higher orders and moment are necessary in the model, making unscented Kalman filter ideal for chaotic time series.
- Time difference model tends to overfit.
- MLE based method tend to under-estimates the variance.
- ARIMA requires domain knowledge to choose ideal parameters.
- Regularizers can be used in the RLS to reduce overfitting.
- **Kalman gain ⇔ damped window**

# Thanks for listening

https://github.com/kenluck2001?tab=repositories

@kenluck2001

https://www.linkedin.com/in/kenluck2001/

kenneth.odoh@gmail.com

# References

1. Castanon, D., & Karl, C. W. SC505: Stochastic processes. Retrieved 06/15, 2017, from http://www.mit.edu/people/hmsallum/GradSchool/sc505notes.pdf

2. Wellstead, P. E. & Karl, C. W. (1991). Self-tuning Systems: Control and Signal Processing. Chichester, United Kingdom: John Wiley & Sons Ltd.

3. Hamilton, J. D. (1994). Time series analysis. Chichester, United Kingdom: Princeton University Press.

4. Julier, S. J. The scaled unscented transformation. Retrieved 06/15, 2017, from https://www.cs.unc.edu/~welch/kalman/media/pdf/ACC02-IEEE1357.PDF

5. Terejanu, G. A. Extended kalman filter tutorial. Retrieved 06/15, 2017, from https://www.cse.sc.edu/~terejanu/files/tutorialEKF.pdf

6. Wan , E. A., & Merwe, R. (2000). The unscented kalman filters for nonlinear estimation. IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium, pp. 153-158.

7.Kevin M. Carter & William W. Streilein (2012). Probabilistic reasoning for streaming anomaly detection. In Proceedings of the Statistical Signal Processing Workshop, pp. 377–380.

8. Y.Zhu & D. Shasha (2002). Statstream: Statistical monitoring of thousands of data streams in real time. In Proceedings of the 28th international conference on Very Large Data Bases, 358–369. VLDB Endowment.

9. Welch, G., & Bishop, G. An introduction to the kalman filter. Retrieved 06/15, 2017, from https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

10. Andrew Fraser (2008). Hidden markov models and Dynamical System. Retrieved 06/15, 2017, from http://www.siam.org/books/ot107/

11. Jinbo Bi & Kristin P. Bennett (2003). Regression Error Characteristic Curves. Retrieved 06/15, 2017, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.566.4289&rep=rep1&type=pdf