KOAD IMPLEMENTED AUTOMATIC ICU SIGNALLING AND ERROR MINIMIZATION

Nazifa Mubashshera Shemonti
ID: 12210010

Arnoba Chakraborty
ID: 12210014

Aniqa Afra Ibnat
ID: 12210016

Department of Electrical and Electronics Engineering
August 2016

BRAC University, Dhaka, Bangladesh

DECLARATION

We hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Dr. Tarem Ahmed

Signature of Supervisor

Nazifa Mubashshera Shemonti

Arnoba Chakraborty

Aniqa Afra Ibnat

ABSTRACT

In this paper, a previously developed algorithm based on self-implementation and kernel mapping is executed in medical field to develop an automated, ICU signalling. The method is taken into consideration because it is adaptive, portable and has lower complexity in comparison to contemporary approaches. The proposed algorithm takes in set of different medical parameters to monitor the conditions of individual patients in an ICU. Application of real data and the learning mechanism of underlying patterns of the algorithm results in instantaneous detection and alarming in response to critical circumstances. In addition, the system yields recognitions with minimum false alarms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Intensive Care Unit (ICU) is the application of intensive care medicine, a branch of medicine centred to treat patients who require external assistance to maintain normal bodily functions. Severely ill or injured patients experience difficulty in breathing, sustaining stable blood pressure, performing renal functions, etc. In ICU, such patients are equipped with mechanical ventilators for respiration, cardiac monitors to display blood pressure, dialysis equipment for renal failure and so on. The complex health condition of ICU patients insists on highly trained physicians and nurses in a greater staff-to-patient ratio than regularhospital wards. Each nurse observes and records vital statistics of patients periodically in order to notify the physician when the readings demonstrate irregularity from usual pattern. However, such manual approach is deemed impractical since the possibility of emergency events in the period in-between going unnoticed increases.

ICU patients,in recent times,are attached to numerous probes of multiple devices which constantly display the vital statistics to their corresponding monitors. Each device has a simplistic, automated signalling installed to alert the staff when some reading deviate from a pre-set threshold. The devices nonetheless, cannot establish an underlying correlation contained by the set of vital statistics. This poses a significant drawback for timely detection since certain indicators are reliant on others.For example,blood pressure levels are

determined by oxygenation and perfusion of tissues where ample supply of oxygen is ensured by proper respiration [1].

In developed countries, the recommended ratio for two patients to one nurse in ICUs is conserveduniformly. Moreover, patients on mechanical ventilator with anaesthetics, sedation and analgesics require 1:1 ratio basis intensive care. Such well-established, speciality ICUs are described as Closed ICU, staffed by intensivists. Intensivist is a physician proficient in treating the range of conditions commonly observed in critically ill patients alongside administering the technical procedures and devices in ICU.

The inadequate financial and technological resources allotted to healthcare in developing countries prove to be a challenge to ensure proper ICU facilities. In Bangladesh, the doctor to nurse to patient ratio recommended by World Health Organization of 1:3:5 falls short by less than one doctor per 3000 population. Moreover, there is an alarming shortage of adequately skilled doctors to provide services let alone operate complex machines [2]. The Kernel-based Online Anomaly Detection (KOAD) algorithm proposed in [3] is a real-time application of automated signalling systems. The algorithm constantly monitors the vital statistics provided by the devices connected to an ICU patient. The adaptive property of KOAD learns the pattern of underlying correlation from the input in frequent intervals. The lightweight nature in terms of calculation and memory space renders the algorithm feasible to use with any kind of device.

The objective of this paper is to show KOAD algorithm can be suitable for ICU implementation considering low false alarm rates in detecting emergency events. We provide the course of trial and error procedure to achieve such result by resolving several parameter values. Our experimentations indicate that cumulative dataset inputs improve the performance of the learning algorithm.

1.1 Related Work

Our work builds most closely on the series of works by Ahmed, T. et al. in [3] and [4]. They demonstrate machine learning principles in pattern matching algorithms to hospital monitoring, automated visual surveillance and network flows in high-speed backbones. Other anomaly detection systemsinclude Principal Component Analysis (PCA), One Class Neighbouring Machine (OCNM) and Kernel PCA. PCA is successful for variables with no inherent correlation. The block- based OCNM method maintains no dictionary since it takes the single closest neighbour in consideration. Unlike KOAD, Kernel PCA executes only for large number of inputs [4]. Kernel methods are used in biology and medicine to predict the formation of disulphide bridges in proteins [5] and to build a Cerebellar Model Articulation Controller (CMAC) neural network to model the part of the brain responsible for fine muscle control in animals [6].

1.2 Outline of Paper

The rest of the paper is organized as follows. Section II provides background on kernel machines and minimum volume sets. Section III presents the KOAD algorithm, discusses the choice of algorithm parameters and analyses computational and memory complexity. Section IV details the performance of KOAD algorithm on data of patients obtained from ICU. Section V concludes and provides future avenues of work on the research.

2. PRELIMINARIES

2.1 Kernel Function

Kernel mapping functions, used by kernel machines, when applied to a pair of input data vectors (feature vectors), maps the input data onto a feature space of much higher dimension [8]. The points exhibiting similar behaviour are expected to cluster in the feature space. The inner product of input vectors each mapped onto the feature space can be used to evaluate the kernel function:

$$\kappa(x_i, x_j) = \prec \phi(x_i), \phi(x_j) \succ \tag{1.1}$$

where x_i, x_j denote the input vectors and Φ represents the mapping onto the feature space.

The KOAD algorithm for ICU emergency signalling uses Gaussian kernel with variance σ^2:

$$\kappa(x_1, x_2) = e^{-\frac{\|x_1 - x_2\|}{2\sigma^2}} \tag{1.2}$$

2.2Kernel Recursive Least Squares Algorithm

The Kernel Recursive Least Squares (KRLS) algorithm combines the principles of kernel machines and the popular Recursive Least Squares (RLS) algorithm [8] and performs non-linear, non-parametric, efficient approach for performing online data mining. In KRLS, the dimension of the space spanned by $\{\phi(x_i)\}_{i=1}^{t}$ can increase without bound.

At each timestep, the dimension will increase unless $x_t$satisfies

$$\phi(x_t) = \sum_{i=1}^{t-1} a_i \cdot \phi(x_i) \tag{1.3}$$

Feature vector $\phi(x_t)$ is *approximately* linearly dependent on $\{\phi(x_i)\}_{i=1}^{t}$ with approximation threshold ν, if the projection error δ_t satisfies the following criterion:

$$\delta_t = min_a \| \sum_{i=1}^{t-1} a_i \cdot \phi(x_i) - \phi(x_t) \|^2 < \nu \tag{1.4}$$

This error measurement δ_t is then compared to two thresholds ν_1 andν_2, where$\nu_1 < \nu_2$.

If$\delta_t < \nu_1$, $x_t$is sufficiently linearly dependent on the dictionary depicting normal behaviour. Hence, “Green” alarm is raised.

If$\delta_t > \nu_2$, $x_t$is far away from the normality region and immediately raise a “Red1” alarm to signal an anomaly.

If $\nu_1 < \delta_t < \nu_2$, x_t is considered sufficiently linearly independent on the dictionary and considered to be an unusual event. An “Orange” signal is raised since it may represent an expansion or migration of the normality region. KOAD implements the “usefulness test” to distinguish between arrivals that is an anomaly:

$$[\sum_{i=t+1}^{t+\ell} \mathbb{I}(\kappa(x_t, x_i) > \ d)\ ] > \ \epsilon\ell \tag{1.5}$$

where$\mathbb{I}$ is the indicator function, d is kernel threshold and $\epsilon \in (0,1)$ is a selected constant. If the test evaluates false, it elevates the “Orange” alarm to “Red2” alarm.

KOAD also deletes obsolete elements from the dictionary to maintain a sparse and current dictionary. Additionally, it incorporates exponential forgetting so that the impact of past observations is gradually reduced.

3. EXPERIMENTS

3.1 Data Collection

Real data was collected from a renowned private hospital consisting of 120 measurement vectors of 11 vital statistics of 3 patients recorded at 1-minute intervals. The list of vital statistics is as follows. Pulse rate, systolic blood pressure, diastolic blood pressure, mean blood pressure, mean pulmonary arterial pressure, peripheral temperature, oxyhaemoglobin saturation, fractional concentration of inspired oxygen, partial pressure of oxygen, partial pressure of carbon dioxide and serum bicarbonate. The appointed cardiac surgeon of the hospital manually identified the values where statistics reached critical level. The labelled instances are the desired values that need automatic detection.

3.2 Results

In order to detect the incidence of the anomalous situations, predetermined by the physician, KOAD wasrun on a set of values for thresholds ν_1 and ν_2. Reduction of false alerts was an additional objective of theexperiment. Figure 3.1 portraits the changes in probability of false alarm in accordance to the probability of detection. As the normal points lie in the feature space defined by the chosen kernel mapping, it isevident from figure 3.1, that the detection of phenomena with the lowest rate of false alerts isachievable.The values for the thresholds ν_1 and ν_2 for the optimum detection to false alert ratio

may be determinedover a training period using a supervised, cross-validation approach from trial and error methods.Furthermore, Figure 3.2 presents a plot of the KOAD detection statistics for a sample experiment. Here KOAD was run with the lower threshold (ν_1) set to a value of 0.06,repeated for set of the upper threshold (ν_2) that included value of 0.001 to 0.9 with an interval of 0.01. Thetimesteps corresponding to the identified anomalies are shown as red, filled stems which yield highvalues for the detection statistic.

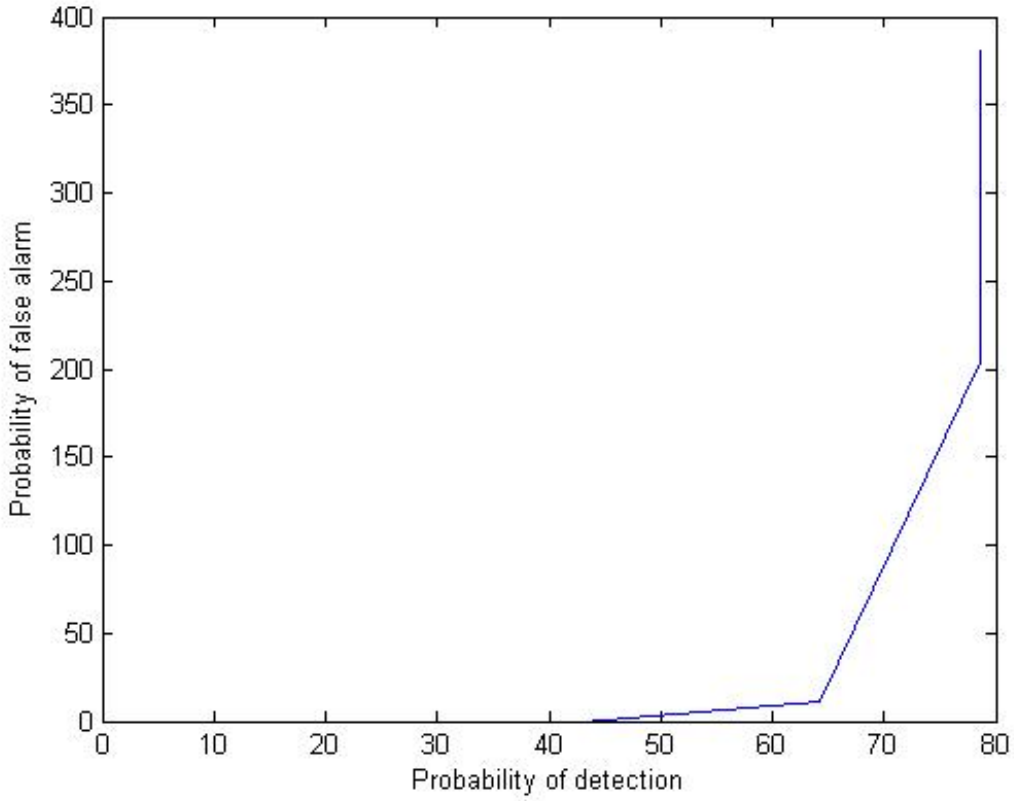

Fig. 3.1 Detection vs. False Alarm

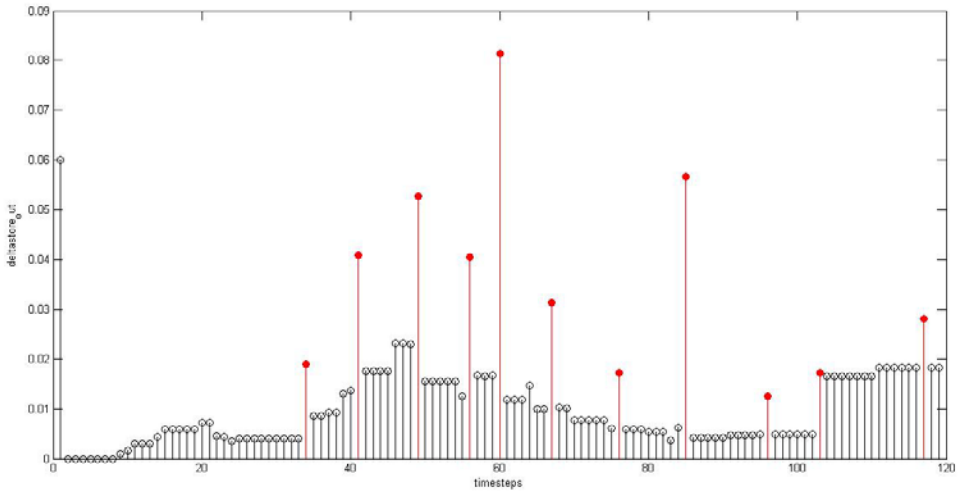

Fig. 3.2 Progression of Detection Statistics, δ_t across 120 timesteps

Remaining two sets of data including detection statistics can be found in appendix.

3.3 Complexity Analysis

Principal concerns regarding the real-time applications are storage and complexity issues. For a dictionary size of *m,* the KOAD creates and eventually stores an *m* X *m* matrix. This results in a computational complexity of $O(m^2)$ in terms of storage requirements and makes the algorithm independent of time and thus suitable for online use.

4. Conclusion and Future Work

In this paper, we have implemented KOAD algorithm for performing automatic, instantaneous detection of emergencies. The algorithm is able to detect anomalous events in real time occurring in a hospital ICU which results in higher detection accuracy with low false alarm rates. The pattern learning algorithm processes a range of vital statistics which are continuously monitored for any particular patient and it has a faster response in detecting anomalies and lower computational complexity compared to similar block-based approaches. Moreover, the system implementation does not require any expensive or sophisticated components which make it easier for developing countries to overcome the financial constraints and establish the usage of the system.

Our future work will expand the testing on investigating the use of additional algorithms to automatically determine the KOAD thresholds. It is also mentionable that the discussed algorithm can be further developed if integrated with medical devices along with supporting interfaces for effective integration and interpretation from multiple sources.

References

[1] E. Kipnis *et al.*, "Monitoring in the intensive care," *Critical Care Research and Practice*, vol. 2012, pp. 1–20, 2012.

[2] *Bangladesh Health System Review*. Philippines: WHO Regional Office for the Western Pacific, pp. 154–162, 2015.

[3] T. Ahmed, S. Ahmed, S. S. Ahmed, and M. Motiwala, "Real-time intruder detection in surveillance networks using Adaptive kernel methods," *2010 IEEE International Conference on Communications*, pp. 1–5, May 2010.

[4] T. Ahmed, M. Coates, and A. Lakhina, “Multivariate online anomaly detection using Kernel Recursive Least Squares,” in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*, Anchorage, AK, USA, May 2007.

[5] J. Cheng, H. Saigo, and P. Baldi, “Large-scale prediction of disulphide bridges using kernel methods, two-dimensional recursive neural networks, and weighted graph matching,” *PROTEINS: Structure, Function, and Bioinformatics*, vol. 62, no. 3, pp. 617–629, Mar. 2006.

[6] C. Laufer and G. Coghill, “Kernel recursive least squares for the CMAC neural network,” *Int. Journal Computer Theory and Engineering*, vol. 5, no. 3, pp. 454–459, Jun. 2013.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein,*Introduction to algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.

[8] B. Scholkopf, A. J. Smola, and er J. Smola, *Learning with kernels: Support vector machines, Regularization, optimization, and beyond*. Cambridge, MA, United States: MIT Press, 2002.

APPENDIX

KOAD MATLAB Code

```
clc; clear all; close all;

p = xlsread('Dataset_p3.xls');
X = double(xlsread('Dataset_p3.xls'));

[T f]=size(X);

gamma = 1; % Forgetting factor
r = 1;  % Parameters for resetting P
R = 10000;
el = 10;  % Parameters for resolving orange alarm
epsilon = 0.2;
L = 40;  % Parameters for dropping obsolete elements
d = 0.9;
kernelChoice = 2;
sigma = 0.1;

X = X./repmat(sqrt(sum(X.*X,2)+eps),1,size(X,2)); % normalize to unit circle
(i.e. divide by norm)
Y = sum(X, 2); %Add after normalizing

sumdec = 0;

nu1 = 0.06;
for nu2 = 0.001:0.01:0.9

```

```
sumdec = sumdec + 1;
flagint = zeros(1, T);

    Red1 = []; Red2 = []; % Clear alarms
    Orange = []; x_Orange = []; % Store x in timesteps when Orange alarm is raised

    % Initialize %
    t = 1;
    x = X(t,:)';
    y = Y(t);
    k11 = kernel(x, x, kernelChoice, sigma);
K_tilde = [k11];
K_tilde_inv = [1/k11];
    Dictionary = [x];
index_m = [t]; % Keeps track of timesteps when elements are added (+2), deleted (-1) or no change to D (0); for debugging only
    Orange = [Orange t];
x_Orange = [x_Orange x];
drop_index = [0];
    P = [1]; % P = inv(A'A)
    m = 1;
m_t(t) = m; % Keep track of m, for debugging only
index_m(t) = 2; % index_m(t)=2 implies x(t) is being added to Dictionary
    alpha = y(t)/k11;
deltaStore(1) = nu1 + eps;  % For debugging

    % % Evaluate y_hat %
```

```
y_hat = zeros(T,1);
    Error = zeros(1,T);
    for j=1:m
y_hat(t) = y_hat(t) + alpha(j)*kernel(Dictionary(:,j),x,kernelChoice, sigma);
    end %for j=1:m
    Error(t) = (Y(t)-y_hat(t))/Y(t)*100;

    clear Lambda lambdadotProd;
    Lambda = kernel(Dictionary(:,j), x, kernelChoice, sigma);

    %Keep track of all dot product (kernel) values; for debugging only
    for j=1:m
dotProd(t,j) = kernel(Dictionary(:, j), x, kernelChoice, sigma);
    end %for j=1:m

    for t=2:T
        %t=t+1;
        x = X(t,:)';
        y = Y(t);

        % Evaluate current measurement %
k_tilde = zeros(m, 1);
        for j = 1:m
k_tilde(j) = kernel(Dictionary(:, j), x, kernelChoice, sigma); % Computing
k_tilde{t-1}
        end % for j = 1:m
        a = K_tilde_inv*k_tilde;
        delta = kernel(x, x, kernelChoice, sigma) - k_tilde'*a;
```

```
        % deltaCheck = a'*K_tilde*a - 2*a'*k_tilde + kernel(x, x); % Verify delta; not part of algorithm
deltaStore(t) = delta;  % Keep track of delta, for debugging only
        if t > L
            Lambda = [Lambda(2:end, 1:end) ; ceil(k_tilde'-repmat(d, 1, m))]; % Append with 1 or 0
        else
            Lambda = [Lambda ; ceil(k_tilde'-repmat(d,1,m))]; % Append with 1 or 0
        end % if t > L

        if (delta >= nu1 & delta < nu2) % Orange alarm, add to Dictionary
x_Orange = [x_Orange x];
            Orange = [Orange t];
            Dictionary = [Dictionary x];
drop_index = [drop_index 0];
a_tilde = a;
K_tilde_inv = [ (delta*K_tilde_inv+a_tilde*a_tilde') (-1*a_tilde) ; (-1*a_tilde') (1) ] / delta;
K_tilde = [ K_tildek_tilde ; k_tilde' kernel(x, x, kernelChoice, sigma) ];
            if t > L
                lambda = [zeros(L-1, 1) ; 1];
            else
                lambda = [zeros(t-1, 1) ; 1];
            end % if t > L
            Lambda = [Lambda lambda];

            a = [zeros(m-1, 1) ; 1];
            P = [ P zeros(m, 1) ; zeros(m, 1)' gamma ]/gamma;
            alpha = [ (gamma^(-0.5)*alpha - a_tilde*(y-gamma^(-
```

```
0.5)*k_tilde'*alpha)/delta)  ;  ((y-gamma^(-0.5)*k_tilde'*alpha)/delta) ];
            m = m+1;
m_t(t) = m;
index_m(t) = 2; % Element added to D in this timestep
        else % delta < nu1 or delta >= nu2, Dictionary unchanged
            if delta > nu2 % Red1 alarm
                Red1 = [Red1 t];
            end % if delta > nu2
            % K_tilde = K_tilde;
            % K_tilde_inv = K_tilde_inv;
            q = (P*a) / (gamma+a'*P*a);
            P = (1/gamma)*[P - q*a'*P];
            alpha = alpha + K_tilde_inv*q*(y-k_tilde'*alpha);
m_t(t) = m;
index_m(t) = 0; % No change to D in this timestep
        end % delta >= nu1 & delta < nu2

        %Keep track of all dot product (kernel) values; for debugging only
        for j = 1:m
dotProd(t, j) = kernel(Dictionary(:, j), x, kernelChoice, sigma);
        end % for j = 1:m

        % Process previous orange alarm %
        if t > el & sum(Orange == t-el) == 1 % means orange alarm at timestep t-
el
            % Identify Dictionary element j corr. to the orange alarm at timestep t-el
            for j = 1:m
                if x_Orange(:, Orange == t-el) == Dictionary(:, j)
```

```
                break;
            end % if x_Orange(:, Orange == t-el) == Dictionary(:, j)
        end % for j = 1:m

        if sum(Lambda(end-el+1:end, j)) <= epsilon*el
            % Orange turns Red
            Red2 = [Red2  Orange(Orange == t-el)]; % Red2 alarm
x_Orange(:, Orange == t-el) = [];
            Orange(Orange == t-el) = [];
drop_index = [zeros(1, j-1) 1 zeros(1, m-j) ];
        else
            % Orange turns green
x_Orange(:, Orange==t-el) = [];
            Orange(Orange==t-el) = [];
        end % if size(find(Lambda(end-el+1:end, j) < d), 1) >= 0.80*25
    end % if t > el & sum(Orange == t-el) == 1

    % Remove obsolete elements %
    for j = 1:m
        %Dropping condition: kernel exists for past L timesteps, and is always
< d
        if  ( t > L & sum(Lambda(1:end, j)) == 0 )
drop_index(j) = 1;
        end % if  ( t>L &gt(Lambda(:, j), 0) &lt(Lambda(:, j), d) )
    end % for j = 1:m

    % DropElement(p) %
    if ( find(drop_index == 1) & m > 1 & t > r )
```

```
        t;
        p = min(find(drop_index == 1)); % Drop Dictionary element # p
        % Reorganize K_tilde_p and K_tilde_inv_p, with p'th row/col moved to the end
K_tilde = [ K_tilde(1:p-1, 1:p-1) K_tilde(1:p-1, p+1:m) K_tilde(1:p-1, p)  ; K_tilde(p+1:m, 1:p-1) K_tilde(p+1:m, p+1:m) K_tilde(p+1:m, p)  ;  K_tilde(p, 1:p-1) K_tilde(p, p+1:m) K_tilde(p, p) ];
K_tilde_inv = [ K_tilde_inv(1:p-1, 1:p-1) K_tilde_inv(1:p-1, p+1:m) K_tilde_inv(1:p-1, p)  ;  K_tilde_inv(p+1:m, 1:p-1) K_tilde_inv(p+1:m, p+1:m) K_tilde_inv(p+1:m, p)  ;  K_tilde_inv(p, 1:p-1) K_tilde_inv(p, p+1:m) K_tilde_inv(p, p) ];
delta_p = 1/(K_tilde_inv(m, m));
a_tilde_p = -delta_p*[K_tilde_inv(1:m-1, m)];
K_tilde_inv = K_tilde_inv(1:m-1, 1:m-1)-a_tilde_p*a_tilde_p'/delta_p;
        alpha = alpha - (1/delta_p)*[a_tilde_p*a_tilde_p'  -a_tilde_p  ;  -a_tilde_p'  1] * K_tilde*alpha;
        alpha = alpha(1:m-1);
K_tilde = K_tilde(1:m-1, 1:m-1);
        Dictionary(:, p) = [];
drop_index(p) = [];
        Lambda(:, p) = [];
dotProd(:, p) = [];
        m = m-1;
m_t(t) = m;
index_m(t) = -1; % Element deleted from D in this timestep

        % Reset P %
        P = R*eye(m);
        for i_r = 1:r
k_tilde = zeros(m, 1);
```

```
            for j = 1:m
k_tilde(j) = kernel(Dictionary(:, j), X(t-i_r, :)', kernelChoice, sigma); % Computing k_tilde{t-1}
            end % for j = 1:m
            a = K_tilde_inv*k_tilde;
            q = (P*a) / (gamma+a'*P*a);
            P = (1/gamma)*[P - q*a'*P];
            alpha = alpha + K_tilde_inv*q*(Y(t-i_r)-k_tilde'*alpha);
        end % for i_r = 1:r-1
    end % if ( find(drop_index ==1) & m>1 & t>r )

    % Evaluate y_hat %
    for j = 1:m
y_hat(t) = y_hat(t) + alpha(j)*kernel(Dictionary(:, j), x);
    end %for j =1:m
    Error(t) = (Y(t)-y_hat(t))/Y(t)*100;

end %for t=2:T

  Red1_out = Red1; Red2_out = Red2; deltaStore_out = deltaStore; Error_out = Error;

trueint = zeros(1, T);
  act = [34 41 49 56 60 67 76 85 96 103 117]; %Dataset_p1
trueint(act)=1;
flagint(Red1) = 1;
flagint(Red2) = 1;
flagint(1) = 0;
```

```
    detected = bitand(flagint, trueint);
    false = bitxor(flagint, trueint);
falsem = false;
    false(act) = 0;
    missed = bitxor(falsem, false);
mis(sumdec)= sum(missed);
dec(sumdec)=(sum(detected)/14)*100;
fal(sumdec)=(sum(false)/(40-14))*100;

    figure(1)
    scatter(sort(dec),sort(fal)), xlabel('Probability of detection'),
ylabel('Probability of false alarm');

End

figure(2)
g = stem(deltaStore_out, 'k');
set(g, 'LineWidth', 1);
hold on;
g = stem(act,deltaStore_out(act), 'r', 'filled');
xlabel('timesteps')
ylabel('deltastore_out')

find(deltaStore_out>0.3);
```

Table A

List of Parameters

Column	Parameter	Full	Range	Unit
A	PR	Pulse Rate	60 – 100	Beats per minute (beats/min)
B	SBP	Systolic Blood Pressure	90 – 140	millimeter of mercury (mm of Hg)
C	DBP	Diastolic Blood Pressure	60 – 89	millimeter of mercury (mm of Hg)
D	MBP	Mean Blood Pressure	80 – 110	millimeter of mercury (mm of Hg)
E	MPAP	Mean Pulmonary Arterial Pressure	10 – 25	millimeter of mercury (mm of Hg)
F	$Temp_P$	Peripheral Temperature	98.6 – 100.4	Fahrenheit
G	SpO_2	Oxyhaemoglobin Saturation	90 – 100	%
H	FiO_2	Fractional Concentration Of Inspired Oxygen	0.21 – 1.0	(N/A – it being a ratio)
I	PaO_2	Partial Pressure of Oxygen	80 – 100	millimeter of mercury (mm of Hg)
J	$PaCO_2$	Partial Pressure of Carbon dioxide	36 – 44	millimeter of mercury (mm of Hg)
K	HCO_3^-	Serum bicarbonate	22 – 26	milli-equivalent perliter (mEq/L)

Table B

EXCEL Dataset of Detection Statistics of Figure 3.2

PR	SBP	DBP	MBP	MPAP	Temp	SPO2	FiO2	PaO2	PaCO2	HCO3
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	142	66	104	12.1	98	99.8	100	87	42	25
92	143	66	104.5	12.1	98	99.8	100	87	42	25
92	143	66	104.5	12.1	98	99.8	100	88	42	25
92	143	66	104.5	12.1	98	99.8	100	88	41	25
92	143	66	104.5	12.1	98	99.8	100	88	41	25
92	143	66	104.5	12.1	98	99.8	100	88	41	25
92	143	66	104.5	12.1	98	99.8	99.8	88	41	26
92	143	65	104	12.1	98	99.8	99.8	88	41	26
92	143	65	104	12.3	98	99.8	99.8	88	41	26
92	143	65	104	12.3	98	99.8	99.8	88	41	26
92	143	65	104	12.3	98	99.8	99.8	88	41	26
92	143	65	104	12.3	98	99.8	99.8	88	41	26
91	143	65	104	12.3	98	99.8	99.8	88	41	26
91	143	65	104	12.3	98	99.8	99.8	88	41	26
91	142	65	103.5	12.3	98	99.9	99.8	87	41	26
91	142	65	103.5	12.3	98	99.8	99.8	87	41	26
91	142	65	103.5	12.3	98	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26

EXCEL Dataset of Detection Statistics of Figure 3.2 (cont'd)

91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
91	142	65	103.5	12.3	97	99.8	99.8	87	42	26
95	143	64	103.5	12.3	97	99.8	99.8	87	42	26
92	143	64	103.5	12.3	97	99.8	99.9	87	42	26
92	143	64	103.5	12.3	97	99.8	99.9	87	42	26
92	143	64	103.5	12.3	97	99.8	99.9	86	42	26
92	143	64	103.5	12.3	97	99.8	99.9	86	42	26
92	143	64	103.5	12.3	97	99.8	99.9	86	42	27
91	143	64	103.5	12.3	97	99.8	99.9	86	42	27
91	143	64	103.5	12.2	97	99.8	99.9	86	37	27
91	143	64	103.5	12.2	97	99.8	99.9	86	40	27
91	143	64	103.5	12.2	97	99.8	99.9	86	40	27
91	143	64	103.5	12.2	97	99.8	99.9	86	40	27
91	143	64	103.5	12.2	97	99.8	99.9	86	40	27
91	143	63	103	12.2	97	99.9	99.9	86	40	27
91	143	63	103	12.2	97	99.9	99.9	86	40	27
91	143	63	103	12.2	97	99.9	99.9	87	40	27
91	142	63	102.5	12.2	103	99.9	99.9	87	40	27
92	142	63	102.5	12.2	98	99.9	99.9	87	40	25
92	142	63	102.5	12.2	98	99.9	99.9	87	40	25
92	142	63	102.5	12.2	98	99.9	99.9	87	40	25
92	142	63	102.5	12.2	98	99.9	99.9	87	40	25
92	142	63	102.5	12.2	98	99.9	99.9	87	40	25
92	142	63	102.5	12.2	98	99.9	99.9	87	41	25
92	142	63	102.5	12.2	98	99.9	99.9	87	41	20
92	142	63	102.5	12.2	98	99.9	99.9	87	41	23
91	142	63	102.5	12.2	98	99.9	99.9	87	41	23
91	142	63	102.5	12.2	98	99.9	100	87	41	23

EXCEL Dataset of Detection Statistics of Figure 3.2 (cont'd)

91	150	63	106.5	12.2	98	99.9	100	87	41	23
91	142	63	102.5	12.2	98	99.9	100	87	41	23
91	142	63	102.5	12.2	98	99.9	100	87	41	23
91	142	63	102.5	12.2	98	99.9	100	87	41	23
91	142	63	102.5	12.2	98	99.9	100	88	41	23
91	142	64	103	12.1	98	99.8	100	88	41	23
91	142	64	103	12.1	98	99.8	100	88	41	23
96	142	64	103	12.1	98	99.8	100	88	41	23
92	142	64	103	12.1	98	99.8	100	88	41	23
92	142	64	103	12.1	97	99.8	100	88	41	23
92	142	64	103	12.1	97	99.8	100	88	41	24
92	142	64	103	12.1	97	99.8	100	88	41	24
92	142	64	103	12.1	97	99.8	100	88	41	24
92	142	64	103	12.1	97	99.8	100	88	41	24
92	142	64	103	12.1	97	99.8	100	88	41	24
92	143	64	103.5	12.1	97	99.8	99.9	88	41	24
92	143	64	103.5	12.1	97	96	99.9	88	41	24
92	143	64	103.5	12.1	97	99.7	99.9	88	41	24
92	143	64	103.5	12.1	97	99.7	99.9	88	41	24
92	143	64	103.5	12.1	97	99.7	99.9	88	41	24
91	143	64	103.5	12	97	99.7	99.9	88	41	24
91	143	64	103.5	12	97	99.7	99.9	88	42	24
91	143	64	103.5	12	97	99.7	99.9	88	42	24
91	143	65	104	12	97	99.7	99.9	88	42	24
91	143	65	104	12	96	99.7	99.9	88	42	24
91	143	72	107.5	12	96	99.7	99.9	87	42	25
91	143	65	104	12	96	99.7	99.9	87	42	25
91	143	65	104	12	96	99.7	99.9	87	42	25
91	143	65	104	12	96	99.7	99.9	87	42	25
91	143	65	104	12	96	99.7	99.9	87	42	25

EXCEL Dataset of Detection Statistics of Figure 3.2 (cont'd)

91	143	65	104	12	96	99.7	99.9	87	42	25
91	144	65	104.5	12	96	99.7	99.9	87	42	25
91	144	65	104.5	12	96	99.7	99.9	87	42	25
91	144	65	104.5	12	96	99.7	99.9	87	42	25
91	144	65	104.5	12	96	99.7	99.9	87	42	25
92	144	65	104.5	12	96	99.7	99.9	87	42	25
92	144	65	104.5	12	96	99.7	97	87	42	25
92	144	65	104.5	12	96	99.7	99.6	87	42	25
92	144	65	104.5	12	96	99.6	99.6	87	42	25
92	144	65	104.5	11.9	96	99.6	99.6	87	42	25
92	144	65	104.5	11.9	96	99.6	99.6	87	42	25
92	144	65	104.5	11.9	96	99.6	99.6	87	42	25
92	144	65	104.5	11.9	96	99.6	99.6	87	42	25
92	144	64	104	11.9	97	99.6	99.6	83	42	25
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	11.9	97	99.6	99.6	89	42	27
91	143	64	103.5	12	97	99.6	99.6	89	43	27
91	143	64	103.5	12	97	99.6	99.6	89	43	27
91	143	64	103.5	12	97	99.7	99.6	89	43	27
91	143	64	103.5	12	97	99.7	99.6	89	43	27
91	143	64	103.5	12	97	99.7	99.7	89	43	27
91	143	64	103.5	12	97	99.7	99.7	89	43	27
91	143	64	103.5	15	97	99.7	99.7	89	43	27
91	143	64	103.5	12	97	99.7	99.7	89	43	27
91	143	64	103.5	12	97	99.7	99.7	89	43	27

O

Figure A

Detection Statistics of Patient 2

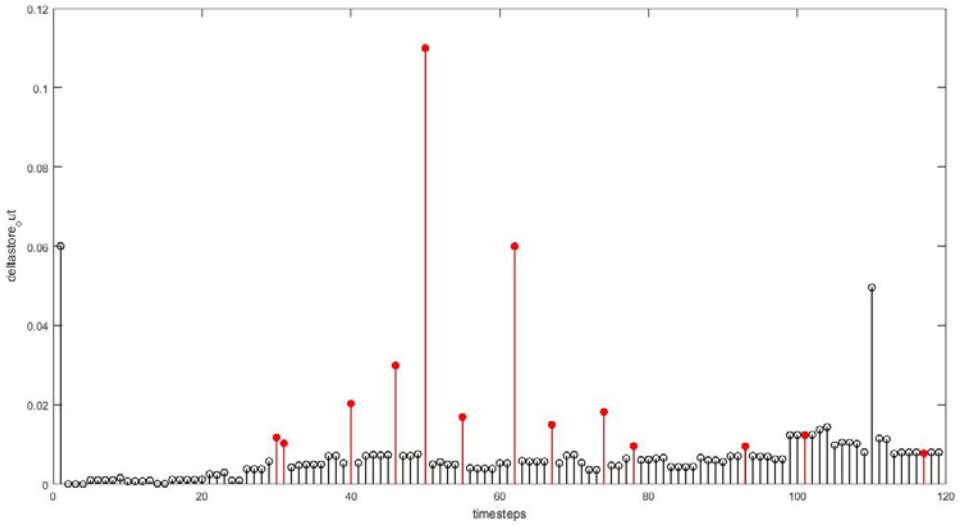

Figure B

Detection Statistics of Patient 3