HA1 Implementation

31-March-2020

Arshad Nowsath

1. SigmaEllipse2D

```
function [ xy ] = sigmaEllipse2D( mu, Sigma, level, npoints )
%SIGMAELLIPSE2D generates x,y-points which lie on the ellipse describing
% a sigma level in the Gaussian density defined by mean and covariance.
%Input:
%   MU          [2 x 1] Mean of the Gaussian density
%   SIGMA       [2 x 2] Covariance matrix of the Gaussian density
%   LEVEL       Which sigma level curve to plot. Can take any positive value,
%               but common choices are 1, 2 or 3. Default = 3.
%   NPOINTS     Number of points on the ellipse to generate. Default = 32.
%Output:
%   XY          [2 x npoints] matrix. First row holds x-coordinates, second
%               row holds the y-coordinates. First and last columns should
%               be the same point, to create a closed curve.

%Setting default values, in case only mu and Sigma are specified.
if nargin < 3
    level = 3;
end
if nargin < 4
    npoints = 32;
end

%Your code here
xy= zeros(2,npoints);  % xy should be [2*32]
phi=linspace(0,2*pi,npoints); % in order to give equal distance between o to 2*pi
for i=1:npoints
    xy(:,i)= mu + level * sqrtm(Sigma) * [cos(phi(i)), sin(phi(i))]';
end
end
```

2. affine Gaussian Transform

```
function [mu_y, Sigma_y] = affineGaussianTransform(mu_x, Sigma_x, A, b)
%affineTransformGauss calculates the mean and covariance of y, the
%transformed variable, exactly when the function, f, is defined as
%y = f(x) = Ax + b, where A is a matrix, b is a vector of the same
%dimensions as y, and x is a Gaussian random variable.
%
%Input
%   MU_X        [n x 1] Expected value of x.
%   SIGMA_X     [n x n] Covariance of x.
%   A           [m x n] Linear transform matrix.
%   B           [m x 1] Constant part of the affine transformation.
%
%Output
%   MU_Y        [m x 1] Expected value of y.
%   SIGMA_Y     [m x m] Covariance of y.

%Your code here
mu_y=A*mu_x+b;          % Normal mean formula from Lecture notes
Sigma_y=A*Sigma_x*A';   % Normal Covariance formula from Lecture notes
end
```

3. Approx Gaussian Transform

```
function [mu_y, Sigma_y, y_s] = approxGaussianTransform(mu_x, Sigma_x, f, N)
%approxGaussianTransform takes a Gaussian density and a transformation
%function and calculates the mean and covariance of the transformed density.
%Inputs
%   MU_X        [m x 1] Expected value of x.
%   SIGMA_X     [m x m] Covariance of x.
%   F           [Function handle] Function which maps a [m x 1] dimensional
%               vector into another vector of size [n x 1].
%   N           Number of samples to draw. Default = 5000.
%Output
%   MU_Y        [n x 1] Approximated mean of y.
%   SIGMA_Y     [n x n] Approximated covariance of y.
%   ys          [n x N] Samples propagated through f
```

```
if nargin < 4
    N = 5000;
end


%Your code here
x_s=mvnrnd(mu_x,Sigma_x,N);     % general form of function mvnrnd (μ,σ)
y_s=f(x_s');                    % Samples propagated through f
mu_y=mean(y_s,2);               % Approximated mean of y
Sigma_y = 1/(N-1)*(y_s-mu_y)*(y_s-mu_y)';   % Approximated covariance of y.
end
```

4. joint Gaussion

```
function [mu, Sigma] = jointGaussian(mu_x, sigma2_x, sigma2_r)
%jointGaussian calculates the joint Gaussian density as defined
%in problem 1.3a.
%
%Input
%   MU_X        Expected value of x
%   SIGMA2_X    Covariance of x
%   SIGMA2_R    Covariance of the noise r
%
%Output
%   MU          Mean of joint density
%   SIGMA       Covariance of joint density

%Your code here
A=[1 0;1 1]; b=[0;0];        % Assumed A and b matrix
mu=A*[mu_x;0]+b;             % Mean of joint density
Sigma=A*blkdiag(sigma2_x,sigma2_r)*A';    %Covariance of joint density

End
```

5. Posterior Gaussian

```
function [mu, sigma2] = posteriorGaussian(mu_x, sigma2_x, y, sigma2_r)
%posteriorGaussian performs a single scalar measurement update with a
%measurement model which is simply "y = x + noise".
%
%Input
%   MU_P            The mean of the (Gaussian) prior density.
%   SIGMA2_P        The variance of the (Gaussian) prior density.
%   SIGMA2_R        The variance of the measurement noise.
%   Y               The given measurement.
%
%Output
%   MU              The mean of the (Gaussian) posterior distribution
%   SIGMA2          The variance of the (Gaussian) posterior distribution
%Your code here
mu=(mu_x*sigma2_r + sigma2_x*y)/(sigma2_r+sigma2_x); %The mean of the (Gaussian)
posterior distribution
sigma2=inv(1/sigma2_r+1/sigma2_x)   %The variance of the (Gaussian) posterior
distribution
end
```

6. Gauss Mix MMSEEst

```
function [ xHat ] = gaussMixMMSEEst( w, mu, sigma2 )
%GAUSSMIXMMSEEST calculates the MMSE estimate from a Gaussian mixture
%density with multiple components.
%
%Input
%   W           Vector of all the weights
%   MU          Vector containing the means of all components
%   SIGMA2      Vector containing the variances of all components
%w'*mu
%Output
%   xHat        MMSE estimate


%YOUR CODE HERE
xHat=w'*mu;      %MMSE estimate
end
```