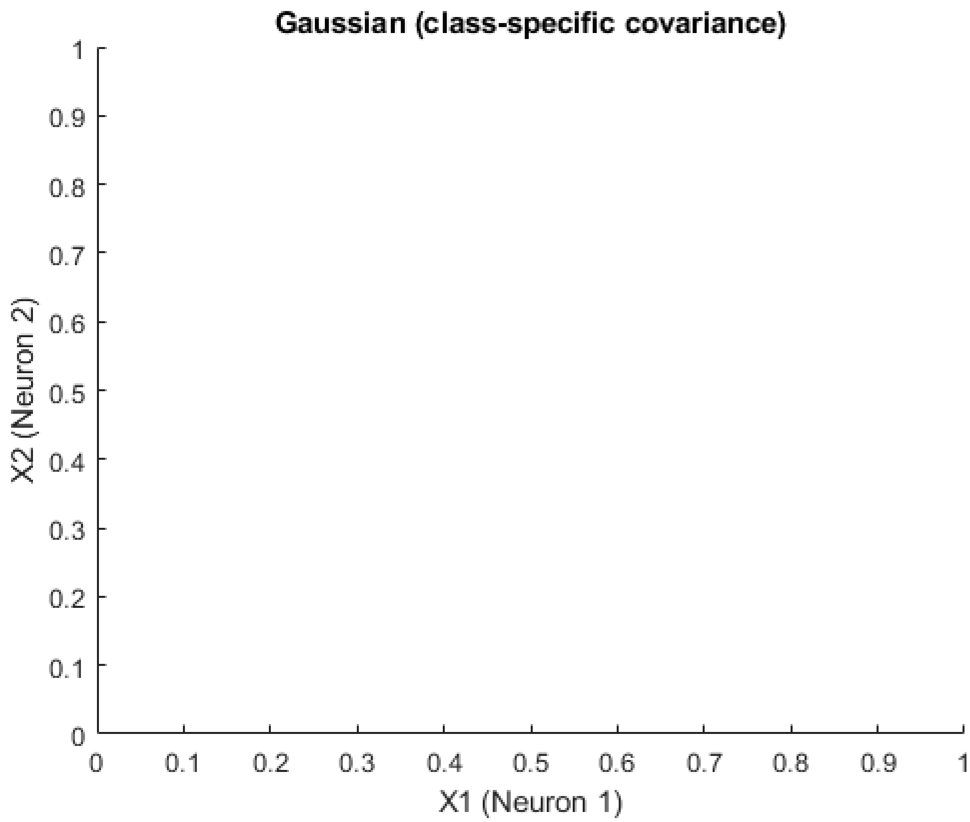
# Table of Contents

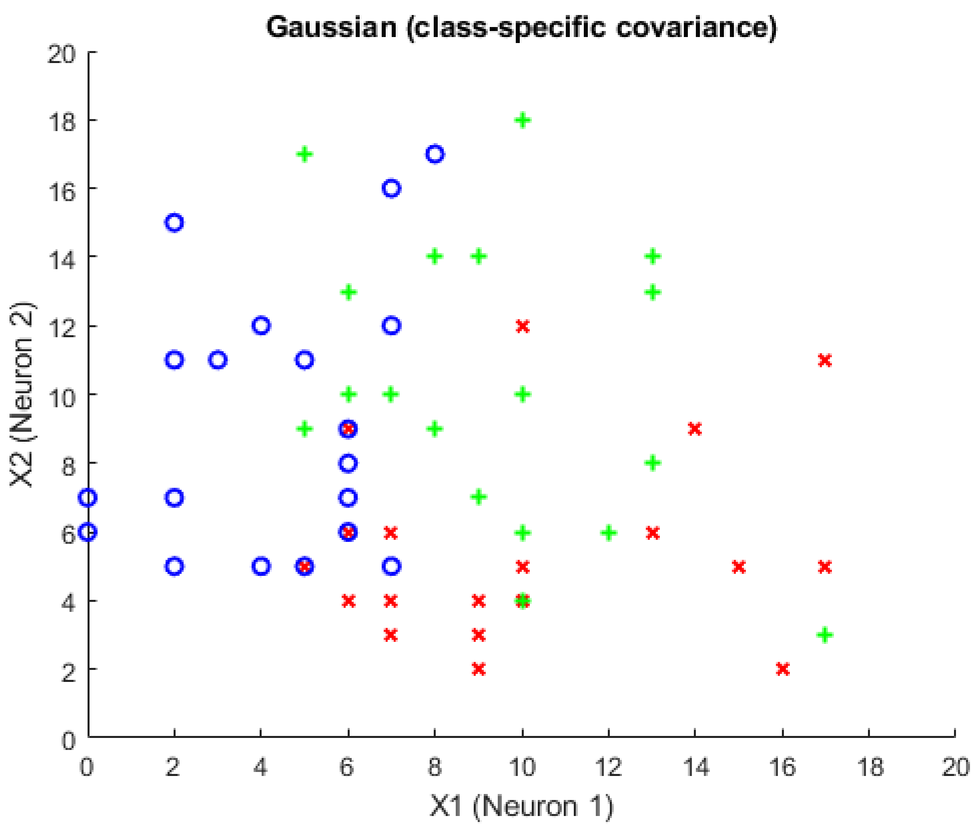


```
load('ps3_simdata.mat')
figure
xlabel('X1 (Neuron 1)')
ylabel('X2 (Neuron 2)')
title('Gaussian (class-specific covariance)')
hold on
```



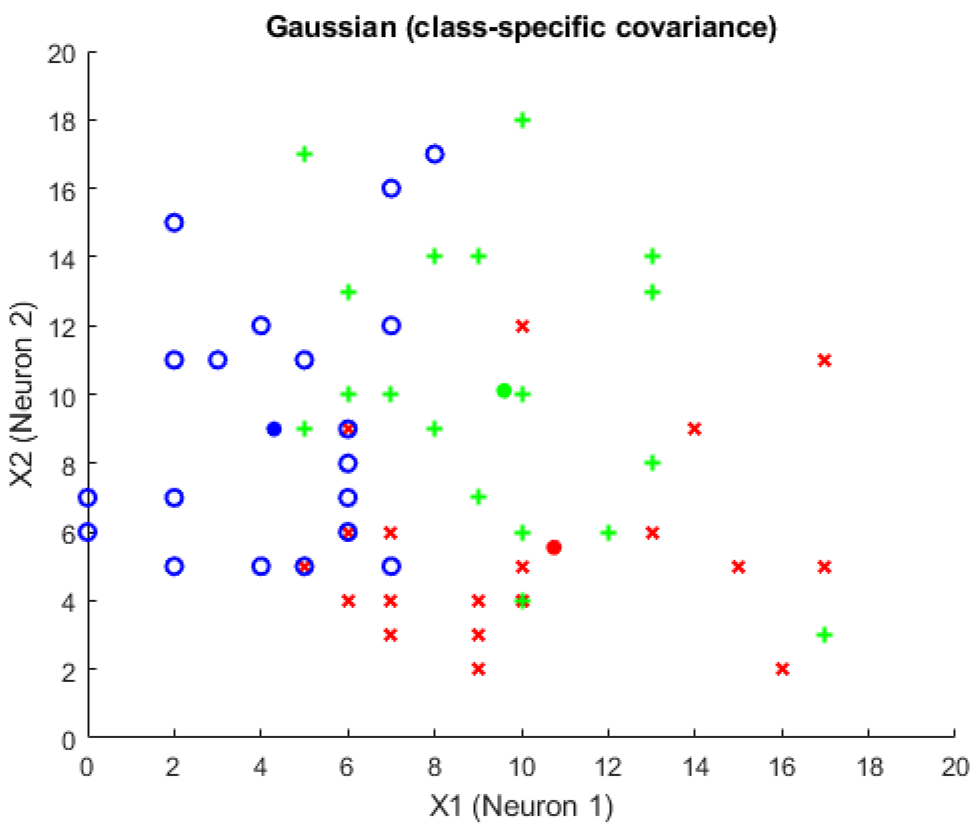
## Plot the data points in a two-dimensional space. For classes k = 1, 2, 3, uses a red ×, green +, and blue o for each data point, respectively.

```
for k = 1:3
[X1, X2] = getValues(trial, k);
if k == 1; marker = 'rx';
elseif k == 2; marker = 'g+';
else; marker = 'bo';
end
plot(X1, X2, marker, 'LineWidth', 1.5)
end
axis([0 20 0 20])
```



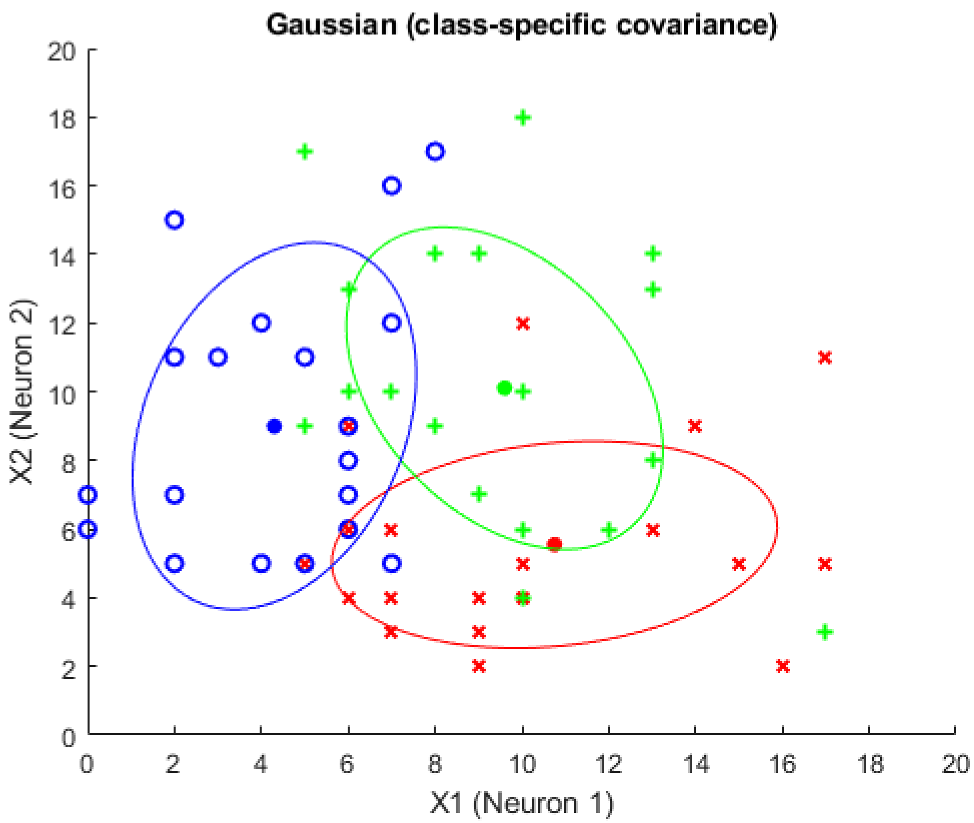
# Find the maximum likelihood (ML) parameters for a Gaussian generative model with specific covariance for each class, and plots the ML mean as a solid dot.

```
pi = 20/60; %Nk/N

[mu1] = getMu(trial, 1);
plot(mu1(1), mu1(2), 'r.', 'MarkerSize', 20)
[mu2] = getMu(trial, 2);
plot(mu2(1), mu2(2), 'g.', 'MarkerSize', 20)
[mu3] = getMu(trial, 3);
plot(mu3(1), mu3(2), 'b.', 'MarkerSize', 20)

[sigma1] = getSigma(trial, 1);
[sigma2] = getSigma(trial, 2);
[sigma3] = getSigma(trial, 3);
```



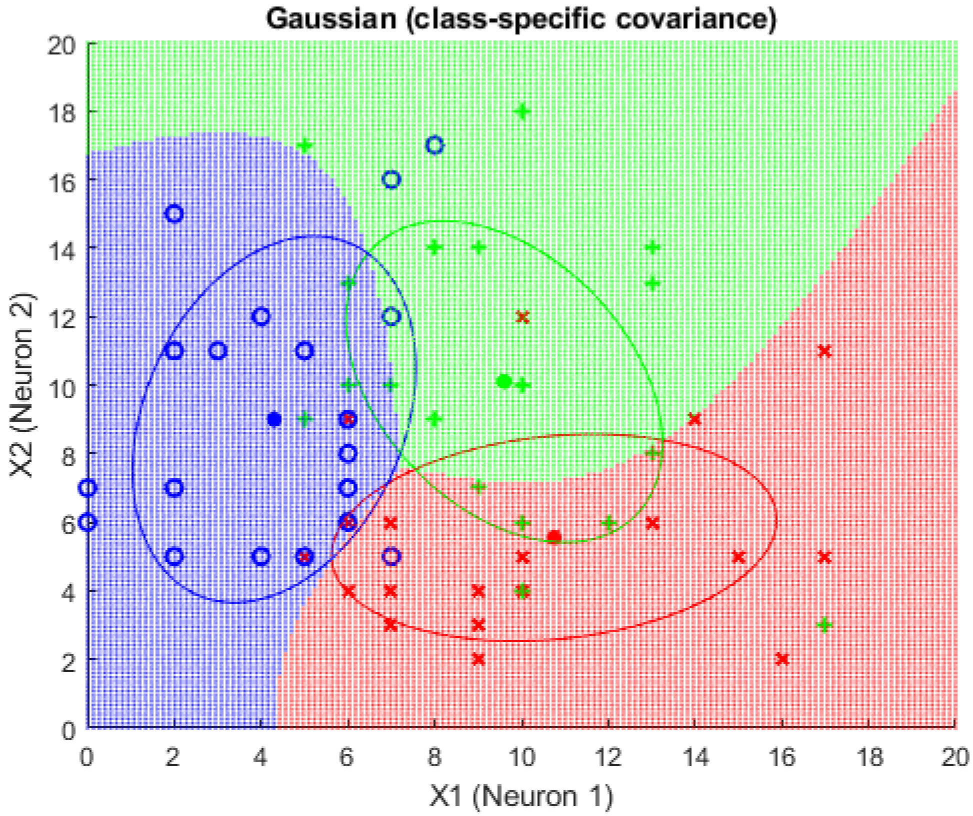
## For each class, plots the ML covariance using an ellipse of the appropriate color.

```
X1 = linspace(0,20);
X2 = linspace(0,20);
[X,Y] = meshgrid(X1(:),X2(:));
nX = [X(:) Y(:)];
Z = mvnpdf(nX, mu1', sigma1);
Z = reshape(Z,[100 100]);
contour(X, Y, Z, [0.007, 0.007], 'r');
Z = mvnpdf(nX, mu2', sigma2);
Z = reshape(Z,[100 100]);
contour(X, Y, Z, [0.007, 0.007], 'g');
Z = mvnpdf(nX, mu3', sigma3);
Z = reshape(Z,[100 100]);
contour(X, Y, Z, [0.007, 0.007], 'b');
```




## Plot multi-class decision boundaries corresponding to the decision rule: ˆk = argmax_k P(Ck | x)

```
plotDecisionSpecificSigma(mu1, mu2, mu3, sigma1, sigma2, sigma3);
```




# FUNCTIONS

```
function [X1, X2] = getValues(trial, k)
%Returns spike counts from neuron 1 and neuron 2 as an array given
%trial number and class k
X1 = zeros(20, 1);
X2 = zeros(20, 1);
for n = 1:20
    X1(n) = trial(n, k).x(1);
    X2(n) = trial(n, k).x(2);
end
end

function [mu] = getMu(trial, k)
%Returns means spike count for a given class for neuron 1 and neuron 2
 as an array
[X1, X2] = getValues(trial, k);
mu = [sum(X1)/20; sum(X2)/20];
end

function [sigma] = getSigma(trial, k)
%Returns covariance for a given class as a 2x2 matrix
[mu] = getMu(trial, k);
sigma = zeros(2, 2);
for n = 1:20
```

```
    x = trial(n, k).x;
    s_x = ((x - mu) * (x - mu)');
    sigma = sigma + s_x;
end
sigma = sigma / 20;
end

function [] = plotDecisionSpecificSigma(mu1, mu2, mu3, sigma1, sigma2,
 sigma3)
%Plots a decision boundary for a Gaussian model with different
 covariance
%for each class
X1 = linspace(0,20,150);
X2 = linspace(0,20,150);
count = 0;
for i = 1:length(X2)
    for j = 1:length(X1)
        vx = [X1(i); X2(j)];
        c1 = log(1/3)-log(2*pi)-((1/2)*log(det(sigma1)))-((1/2)*(vx-
mu1)'*inv(sigma1)*(vx-mu1));
        c2 = log(1/3)-log(2*pi)-((1/2)*log(det(sigma2)))-((1/2)*(vx-
mu2)'*inv(sigma2)*(vx-mu2));
        c3 = log(1/3)-log(2*pi)-((1/2)*log(det(sigma3)))-((1/2)*(vx-
mu3)'*inv(sigma3)*(vx-mu3));
        k = max([c1, c2, c3]);

        if c3 >= c1; count = count + 1;
        elseif c3 >= c2; count = count + 1;
        end
        if k == c1; marker = '.r';
        elseif k == c2; marker = '.g';
        elseif k == c3; marker = '.b';
        end
        plot(X1(i),X2(j),marker, 'MarkerSize', 1);
    end
end
end
```

*Published with MATLAB® R2019b*