In maximum likelihood decoding, the probability of error (p(error)) is computed by comparing a given observation to a threshold. In the first analysis of this homework, 10,000 random data points were generated from two Gaussian random variables ($GRV(\mu,\sigma^2)$) and the probability of error was computed using a simple Matlab script. These results were compared to the analytical probability of error and are listed in Tables I and II respectively for $GRV(1,1)$ and $GRV(-1,1)$.

**Table I. Probability of Error for GRV(1,1)**

| GRV(1,1) | | |
|---|---|---|
| Threshold | p(error)$_{sim}$ | p(error)$_{analytical}$ |
| 0 | 0.157 | 0.1586 |
| -0.5 | 0.066 | 0.0668 |
| 0.5 | 0.31 | 0.3085 |

**Table II. Probability of Error for GRV(-1,1)**

| GRV(-1,1) | | |
|---|---|---|
| Threshold | p(error)$_{sim}$ | p(error)$_{analytical}$ |
| 0 | 0.157 | 0.1586 |
| -0.5 | 0.31 | 0.3085 |
| 0.5 | 0.066 | 0.0668 |

Given that the means of GRV(1,1) and GRV(-1,1) are equidistant from the first threshold of 0, it makes sense that their p(error) are identical (See Tables I and II). Similarly because of symmetry, the p(error) for GRV(1,1) and GRV(-1,1) with threshold of -0.5 and 0.5 have opposite values for p(error) with respect to each other. These results are confirmed when they are compared to the analytical results for the p(error). Note that the analytical p(error) was computed numerically in Matlab by integrating the standard equation for a normal distribution. The limits of integration for each error was between the threshold and 3 standard deviations. Ideally we would integrate to infinity (or from infinity depending on the distribution under consideration), but since 99% of the area under the distribution is within 3 standard deviations, we can achieve an accurate approximation by integrating to 3 standard deviations (or from 3 standard deviations).

The next question that arises is what happens to the error rate as the variance of the normal distribution is changed. By considering the distributions GRV(1,1) and GRV(-1,1) with a threshold of 0, Intuitevly, we would expect the error to be directly proportional to the variance because as the variance decresease, the distribution curve gets more narrow. Thus the amount of area that exceeds the threshold is reduced thereby reducing the p(error). Figure 1, confirms this intution by plotting the p(error) for a normal distribution with a mean of 1 and a threshold of 0 as a function of the variance. 100,000 data points were genereated and compared to the threshold to produce a smooth curve. The variance was varied from 0 to 5 with interval spacings of 0.1.

From Figure 5, we observe that the p(error) takes a sharp increase when the variance starts to exceeds 0.3. The reason is justified by recgonizing that the distance between the mean of the normal distribution and the threshold is 1 (mean = 1, threshold = 0). As was mentioned before, 99% of the area is within three standard deviations. if we take the distance between the mean and threshold and divide by three, we find that the maximum standard deviation before crossing the threshold is 1/3. Squaring this results in a variance of 1/9 = 0.1111. Thus when the variance exceeds 0.1111, the p(error) starts increasing in accordance with the exponetional nature of the normal distribution.

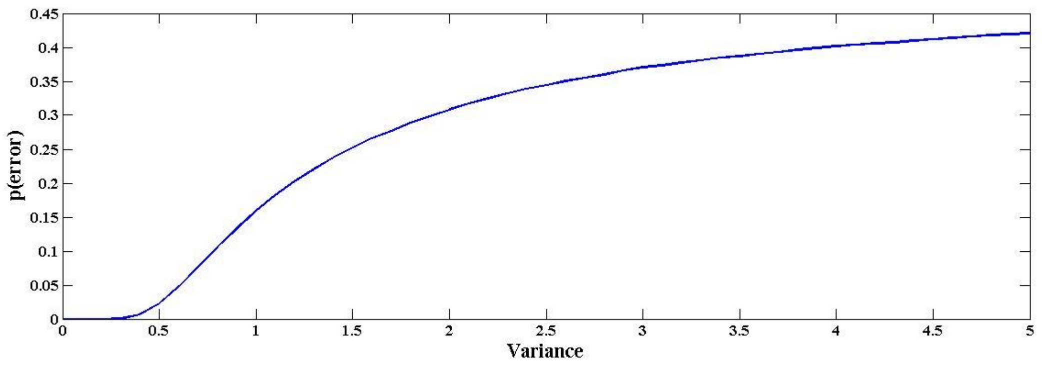

**Figure 1. p(error) as a function of variance for GRV(1,1).**

Now we consider a two dimensional (2D) normal distribution. We will present the same analysis for the 2D case as we did for the 1D case excepts that the tools used for analysis differ to account for the added dimension. Adding a second dimension is like adding another classifier for decision pruposes. An example of classifiers could be the length and lightness used for deciding whether a fish is a sea bass and salmon. Figure 1 depicts two 2D normal distrubtions with means of [-1 1] for GRV1 and [1 1] for GRV2. Both distributions have an identity covariance matrix.

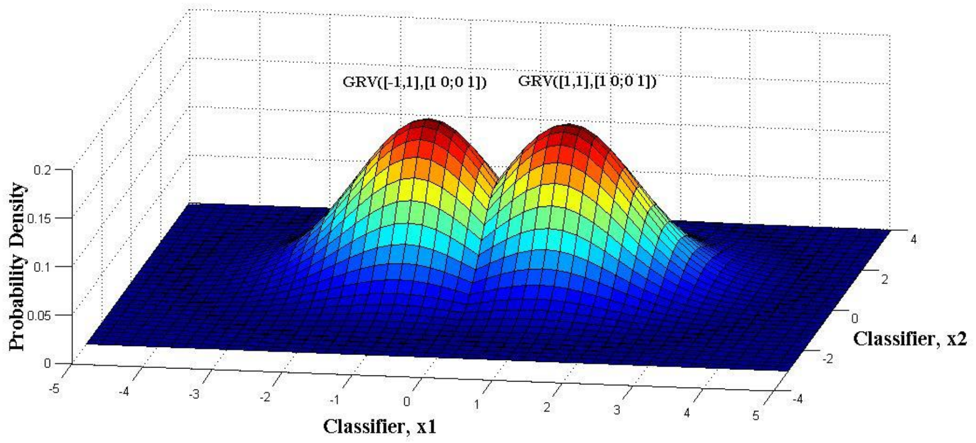

**Figure 2. 2D normal distributions with identity covariance matrices and means of [-1 1] and [1 1] respectivly.**

Again we computed the p(error) for each 2D distribution when the threshold was -0.5, 0, and 0.5. For this analysis, 1000 data points were generated from each distribution. The distribution of data points are illustrated in Figure 3.

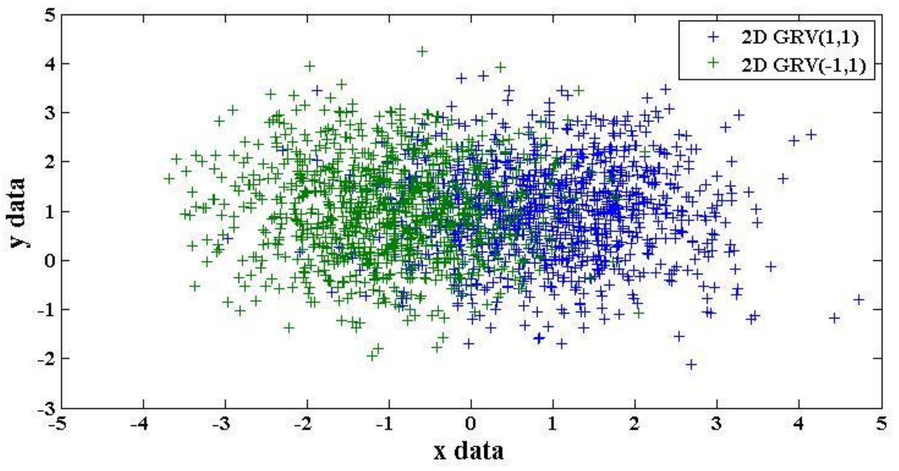

**Figure 3. 1,000 normally distributed data points for each of the 2D GRV**

Interestingly enough, when the p(error) for a multivariate normal random variable is computed, the results are similar to the univariate case in the 1D analysis. Comparing Tables III to I and IV to II, we find almost identical results.

Table III

| 2D GRV([1 1], [1 0; 0 1]) | |
|---|---|
| Threshold | p(error)$_{sim}$ |
| 0 | 0.154 |
| -0.5 | 0.062 |
| 0.5 | 0.307 |

Table IV.

| 2D GRV([-1 1], [1 0; 0 1]) | |
|---|---|
| Threshold | p(error)$_{sim}$ |
| 0 | 0.159 |
| -0.5 | 0.31 |
| 0.5 | 0.063 |

When the decision threshold is zero along the x-axis, the p(error) for each distribution is approximately the same and when the decision thresholds are -0.5 and 0.5, the distributions p(error) are opposite each other's. Again this is the result of symmetry amongst the means and covariance's.

In our final analysis, we varied the covariance matrix of one normal distribution and compared it to the other to see how it effects the shape of the support region. In Matlab, we can generate a support region by plotting a contour plot of the distribution. Each contour line represents a constant probability.

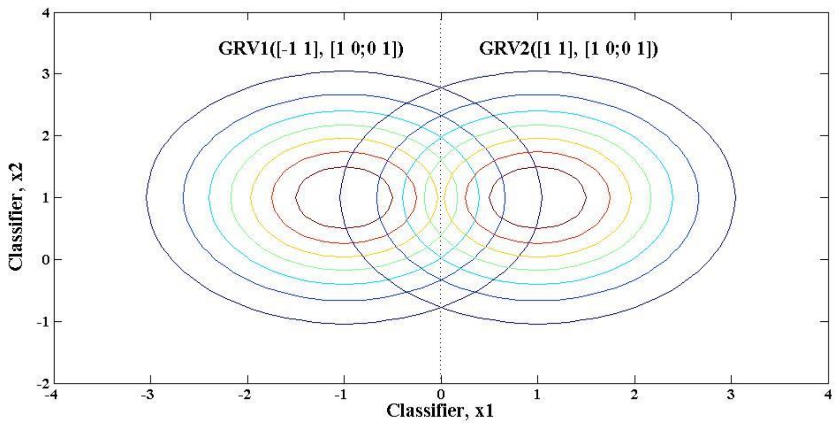

**Figure 4. Support regions for two normal distributions with identity covariance matrices.**

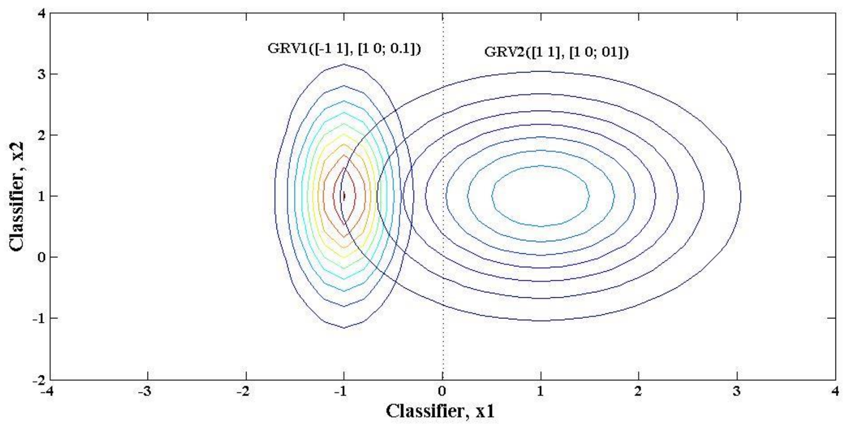

**Figure 5. Support regions for two normal distributions with GRV1 having a covariance matrix of [.1 0; 0 1] and GRV2 having and identity covariance matrix**

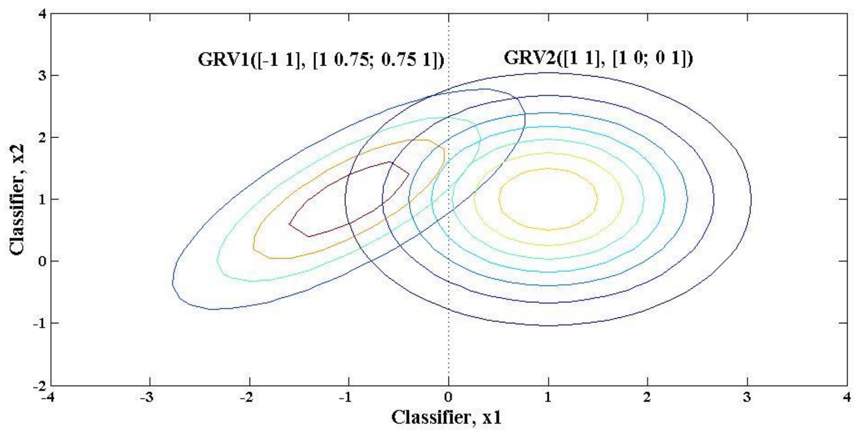

**Figure 6. Support regions for two normal distributions with GRV1 having a covariance matrix of [1 0.75; 0.75 1] and GRV2 having and identity covariance matrix**

Figures 4-6 illustrate how changing the covariance matrix results in a support region with a different shape. Mathematically, the covariance matrix is represented as follows:

$$COVAR = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix} \tag{1}$$

Thus from equation (1) we can adjust how wide or narrow the support region is by adjusting $\sigma_x^2$ or $\sigma_y^2$ or we can rotate the support by adjusting $\sigma_{yx}$ and $\sigma_{xy}$. However, it's important to note that the covariance matrix must be a square, symmetric positive definite matrix.

In Figure 4, the covariance matrices for each distribution are an identity matrix. The result is a 2D normal distribution with equal variance in all direction form the mean. In Figure 5 though, the covariance matrix of GRV1 is [.1 0; 0 1]. The variance of the classifier, x1 was reduced from 1 to 0.1 resulting in a narrower distribution along the x-axis and a reduced p(error). Finally, Figure 5, illustrates a rotated support region. This is accomplished by adjusting the off diagonal elements of equation (1). The covariance matrix in Figure 5 was [1 0.75;0.75 1]. The larger the off diagonal elements, the larger the rotation.

---

```
clear;
clc;
clf;
%
% Brian Thibodeau
% ECE 8527:  Machine Learning
% HW NO. 1:  Gaussian Distributions and
%            Maximum likelihood Decoding
%
%
% Q1
x1 = normrnd(1,1,1,10e3);
x2 = normrnd(-1,1,1,10e3);

tr = 0;

x1_count_c1 = 0;
x1_count_c2 = 0;
x2_count_c1 = 0;
x2_count_c2 = 0;

for j = 1:length(x1)
    if (x1(j) > tr)
        x1_count_c1 = x1_count_c1 + 1;
    end
    if (x1(j) < tr)
        x1_count_c2 = x1_count_c2 + 1;
    end

    if (x2(j) > tr)
        x2_count_c1 = x2_count_c1 + 1;
    end
    if (x2(j) < tr)
        x2_count_c2 = x2_count_c2 + 1;
    end
```

```
end

perr_x1 = x1_count_c2/length(x1);
perr_x2 = x2_count_c1/length(x2);

syms x;
mu1 = -1;
sigma = 1;
rv = 1/sqrt(2*pi*sigma)*exp(-1/2*((x-mu1)^2/sigma));
double(int(rv,x,tr,3*sigma))
%% Q2
clear;
clc;
clf;

var = 0:0.1:5;
mu1 = 1;
tr = 0;
for i = 1:length(var)
    x1 = normrnd(mu1,var(i),1,1e6);
    errCnt = 0;
    for j = 1:length(x1)
        if (x1(j) < tr)
            errCnt = errCnt+ 1;
        end
    end
    perr(i) = errCnt/length(x1);
end
plot(var,perr);
xlabel('Variance');
ylabel('p(error)');
%% Q3
clear;
clc;
clf;

tr = 0.5;

mu1 = [1 1];
mu2 = [-1 1];
sigma1 = [1 0;0 1];
sigma2 = [1 0;0 1];
r1 = mvnrnd(mu1,sigma1,1e3);
r2 = mvnrnd(mu2,sigma2,1e3);
plot(r1(:,1),r1(:,2),'+',r2(:,1),r2(:,2),'+');
xlabel('x data');
ylabel('y data');
legend('2D GRV(1,1)','2D GRV(-1,1)');
r1numErr = 0;
r2numErr = 0;
for i = 1:length(r1(:,1))
    if (r1(i,1) < tr)
        r1numErr = r1numErr + 1;
    end
    if (r2(i,1) > tr)
        r2numErr = r2numErr + 1;
    end
end

r1perr = r1numErr/1e3
r2perr = r2numErr/1e3


%% Q4
clear;
clc;
clf;

mu1 = [-1 1];
mu2 = [1 1];

sigmax = .1;
sigmay = 1;
covar1 = [sigmax 0;0 sigmay];
covar2 = [1 0;0 1];
x1 = -5:.2:5; x2 = -3:.2:4;
[X1,X2] = meshgrid(x1,x2);

GRV1 = mvnpdf([X1(:) X2(:)],mu1,covar1);
GRV1 = reshape(GRV1,length(x2),length(x1));

GRV2 = mvnpdf([X1(:) X2(:)],mu2,covar2);
GRV2 = reshape(GRV2,length(x2),length(x1));

% figure(1);
% surf(x1,x2,GRV1);
% hold on;
% surf(x1,x2,GRV2);

figure(2);
contour(x1,x2,GRV1);
hold on;
contour(x1,x2,GRV2);
```