Detection Theory and Pattern Classification

Jongoh (Andy) Jeong

ECE302 Stochastic Processes
Cooper Union, New York, NY

***Abstract*—This exercise explores applications of detection theory, including MPE and MAP decision rules, provided a set of conditions and hypotheses, as well as data samples. Deriving decision rules for scalar Gaussian radar detection system, one can determine the threshold at which the decision of presence of the target is made. For pattern classification of the given set of data sample points with classes, maximum a posteriori probability density functions are determined, and the probability of error from the derived confusion matrix is determined, after normalization due to multiple iterations.**

Keywords—MAP, MPE, Confusion Matrix, Error

I. Introduction

A. *MAP Decision Rule Derivation*

In general making a maximum a posteriori decision in a binary hypotheses scenario involves minimizing the associated expected cost of the decision. The cost for each decision is computed the following way, with each cost C_{ij} defined as cost of declaring decision i given j true. For decision for target absence (H_0) and presence (H_1), one can define the expected cost for each decision as:

For decision $\mathrm{D(y)} = H_0$,

$$\mathrm{E}\left[C_{D(y)=H_0} \middle| y\right] = C_{00}P_{H|Y}(H_0|y) + C_{01}P_{H|Y}(H_1|y)$$

For decision $\mathrm{D(y)} = H_1$,

$$\mathrm{E}\left[C_{D(y)=H_1} \middle| y\right] = C_{10}P_{H|Y}(H_0|y) + C_{11}P_{H|Y}(H_1|y)$$

Since the optimal decision derives from the smaller of the two above, represent this decision as inequalities as follows:

$$C_{00}P_{H|Y}(H_0|y) + C_{01}P_{H|Y}(H_1|y) \gtreqless C_{10}P_{H|Y}(H_0|y) + C_{11}P_{H|Y}(H_1|y),$$

$(\text{if} >, \text{then choose } H_1, \quad else\ if <, choose\ H_0)$ * denoted as $\frac{H_1}{H_0}$

Using the Bayes Rule,

$$P_{H|Y}(H_i|y) = \frac{P_{Y|H}(y\,|H_i)P_H(H_i)}{P_Y(y)}$$

and assuming that errors are more costly than correct decisions, such that

$$(C_{01} - C_{11}) > 0, \quad (C_{10} - C_{00}) > 0,$$

Bayes Risk decision rule D(y), or a likelihood ratio test, is determined as

$$Likelihood\ Ratio\ L(y) = \left[\frac{P_{Y|H}(y\,|H_1)}{P_{Y|H}(y\,|H_0)}\right] \gtreqless \frac{(C_{10} - C_{00})P_H(H_0)}{(C_{01} - C_{11})P_H(H_1)} \equiv \eta, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

Using the cost assignment

$$C_{ij} = 1 - \delta_{ij}, \quad \delta_{ij} = \begin{cases} 1, & if\ i = j \\ 0, & if\ i \neq j \end{cases},$$

the expected cost for the decision made, D(y), is simply the probability of error:

$$\begin{aligned}\mathrm{E}[C_D(y)] = {} & C_{00}\,\Pr[Choose\ H_0, H_0\ true] \\ & + C_{01}\,\Pr[Choose\ H_0, H_1\ true] \\ & + C_{10}\,\Pr[Choose\ H_1, H_0\ true] \\ & + C_{11}\,\Pr[Choose\ H_1, H_1\ true] \\ = {} & \Pr[Choose\ H_0, H_1\ true] + \Pr[Choose\ \mathrm{H}_1, \mathrm{H}_0\ true] \\ = {} & \Pr[Error]\end{aligned}$$

The optimal detector would try to minimize this probability of error, which is simply the minimum probability of error (MPE) decision rule:

$$\frac{P_{Y|H}(y\,|H_1)}{P_{Y|H}(y\,|H_0)} \gtreqless \frac{P(H_0)}{P(H_1)} \equiv \eta, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

Since $P_{Y|H}(y\,|H_i)P_H(H_i) = P_{H|Y}(H_i|y)P_Y(y)$, the maximum a posteriori decision probability decision rule (MAP rule) is derived to be

$$P_{H|Y}(H_1|y) \gtreqless P_{H|Y}(H_0|y), \qquad \left(for\ \frac{H_1}{H_0}\right)$$

II. Exercises

A. *Radar Detection*

- Conditions (a)

Two hypotheses – H_1 and H_0 – for the presence of target are proposed, with H_0 (target absent) having probability of 0.8 and noise level defined by a zero-mean Gaussian distribution with arbitrary variance $\sigma_x{}^2$, and H_1 (target present) having a constant symbol A in addition to the same noise level as H_0.

$$\begin{aligned}& H_1: \text{Target present}; \mathrm{Y = A + X} \\ & H_0: \text{Target not present}; \mathrm{Y = X}, \qquad \text{where } \mathrm{X} \sim \mathrm{N}(0, \sigma_x{}^2) \\ & \mathrm{P}\,[H_0] = 0.8, \qquad \mathrm{P}\,[H_1] = 0.2 \\ & \text{Let } \sigma_x{}^2 = \alpha, \text{and thus SNR} = \frac{A}{\sigma_x{}^2} = \beta\end{aligned}$$

- Hypotheses for Target Presence

Under H_1 and H_0, the class conditional densities are Gaussian distributions with mean A, variance $\sigma_x{}^2$, and zero mean, variance $\sigma_x{}^2$, respectively, as the resulting signal Y is either the Gaussian distribution X itself, or the sum of X and a constant symbol.

$$H_1: p_{Y|H_1}(y \mid H_1) = \mathrm{N}(y;\, m_{1,}\sigma_1)$$

$$= \frac{1}{\sqrt{2\pi\sigma_1{}^2}} - \exp\left(\frac{(y - m_1)^2}{2\,\sigma_1{}^2}\right) = \frac{1}{\sqrt{2\pi\sigma_x{}^2}} - \exp\left(\frac{(y - A)^2}{2\sigma_x{}^2}\right)$$

$$H_0: p_{Y|H_0}(y \mid H_0) = \mathrm{N}(y;\, m_{0,}\sigma_0)$$

$$= \frac{1}{\sqrt{2\pi\sigma_0{}^2}} - \exp\left(\frac{(y - m_0)^2}{2\sigma_0{}^2}\right) = \frac{1}{\sqrt{2\pi\sigma_x{}^2}} - \exp\left(\frac{y^2}{2\sigma_x{}^2}\right)$$

- MAP decision rule for target detection

This is a case for same variances, different means for scalar Gaussian Detection, such that:

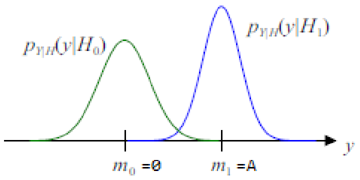


Fig. 1. Same $\sigma_0{}^2$, Different Means for 2-hypotheses scalar Gaussian detection[3]

Using the MPE rule, given the conditions for prior probabilities, one can determine η, the threshold, to be 4:

$$\frac{P_X(X=y-A)}{P_X(X=y)} \gtrless \frac{0.8}{0.2} = 4, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

Determine Bayes Risk (LRT):

$$\mathrm{L(y)} = \left[\frac{\frac{1}{\sqrt{2\pi\sigma_1{}^2}}\exp\left(\frac{(y-m_1)^2}{2\sigma_1{}^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_0{}^2}}\exp\left(\frac{(y-m_0)^2}{2\sigma_0{}^2}\right)}\right] \gtrless \eta$$

$$\left(for\ \frac{H1}{H0}, for\ some\ threshold\ \eta\right)$$

Taking natural logarithmic to eliminate exponentials,

$$-\frac{(y-m_1)^2}{2\sigma_1{}^2} + \frac{(y-m_0)^2}{2\sigma_0{}^2} \gtrless \ln\left(\frac{\sigma_1}{\sigma_0}\eta\right), \qquad \left(for\ \frac{H_1}{H_0}\right)$$

For $\sigma_0{}^2 = \sigma_1{}^2 = \sigma_x{}^2\ and\ m_0 = 0, m_1 = A,$

$$\mathrm{y} \gtrless \frac{m_0+m_1}{2} + \frac{\sigma_x{}^2\ln(\eta)}{m_1-m_0} = \frac{A}{2} + \frac{\sigma_x{}^2 ln(\eta)}{A} \equiv \Gamma \quad \left(for\ \frac{H_1}{H_0}\right)$$

(if y is larger than Γ, choose H_1, and if smaller than Γ, choose H_0)

- Probability of Error

Terminology:

$$P_F \equiv [Choose\ H_1 \mid H_0\, true], false\ alarm$$
$$P_D \equiv [Choose\ H_1 \mid H_1\, true], detection$$
$$P_M \equiv [Choose\ H_0 \mid H_1\, true], miss$$

As explained for the expected cost of the decision, E[$C_{D(y)}$], the probability of error is simply the sum of probabilities of a "miss" and a "false alarm" based on the cost structure as follows:

$$\begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{aligned} \Pr[Error] &= \Pr[Choose\ H_0, H_1\, true] \\ &\quad + \Pr[Choose\ \mathrm{H}_1, \mathrm{H}_0\, true] \\ &= P_M P_1 + P_F P_0\ (= (1-P_D)P_1 + P_F P_0) \\ &= \Pr[y < \Gamma | H_1\ true]\Pr[H_1] \\ &\quad + \Pr[y > \Gamma | H_0 true]\Pr[H_1] \end{aligned}$$

- ROC for SNR values (b)

In order to compare theoretical and simulation results for the probability of error as derived, arbitrary experimental values for variance of X, α, and SNR, β, are chosen.

α (VarianceX) = 3, β (SNR) = 1.0

Initially, the constant symbol A is determined by multiplying SNR by the variance of X. With known prior probabilities for each hypothesis, the detector is simulated a thousand times with different random variables from Gaussian distributions with specified mean and variance for each hypothesis. These random variables are then fit to distributions for each hypothesis, for the purpose of "declaring" as that hypothesis. For each case, the number of error is computed by subtracting the distribution obtained from mismatching declaration-hypothesis from the one obtained from matching pair. This allows for detecting an error when the difference is negative. The experimental error is then calculated by counting the total number of errors and diving by the number of iterations.

The theoretical probability of error is computed from the threshold determined earlier from the MPE (MAP) decision rule. Since error is found by

$$\mathrm{P_{error} = (1\text{-}P_{detection})*P_{H1} + P_{FalseAlarm}*P_{H0}},$$

probability density functions (pdf) for the specified MAP threshold, mean, and variance for each hypothesis are drawn from normal cumulative distribution curves, from which probabilities of detection, miss, and false alarm are determined. Detection ranges from the threshold to infinity for H_1, miss is simply complement of detection, and false alarm is drawn from the threshold to infinity for H_0.

After a thousand iterations of random sampling, the probability of error is determined to be 0.0075, or 0.75%, while the theoretical error is 0.1330, or 13.3%. The experimental result is quite close to theoretically calculated value, with the deviation of 0.132, or (0.1330-0.1320)/0.1330*100% = 0.75% away from the theoretical value.

- Receiver Operating Characteristics (b)

For the identical arbitrary values for the variance of X and SNR, a set of various SNR values yield different behaviors in the receiver operating characteristic curves (ROC). Probabilities of detection and false alarm are computed with the MAP decision rule threshold as the criteria for deciding one hypothesis or the other, which are marked on the graph as black star *, and the theoretically calculated P_D and P_F are marked as magenta star *. Theoretical calculation of the probabilities are performed as follows:

$$P_F = Q\left(\frac{\Gamma}{\sigma}\right), P_D = Q\left(\frac{\Gamma}{\sigma} - d\right)$$

where $$\mathrm{d2} = \frac{[\,\mathrm{E}[\,\Pr[\mathrm{y} \mid H_1]\,] - \mathrm{E}[\,\Pr[\mathrm{y} \mid H_0]\,]^2}{\sqrt{Var(y|H_1)Var(y|H_0)}} = \frac{\mathrm{E}^2}{\sigma^2}.$$

In general for a meaningful detection system model, the ROC tends to be convex down, lying above $P_D = P_{FA}$ identity line. For a set of SNR values {2, 1.5, 1, 0.75}, the curve exhibits a more ideal behavior as the ratio increases. This intuitively makes sense as there is relatively less noise than the signal, and thus the probability of choosing H_1, the target present, increases.

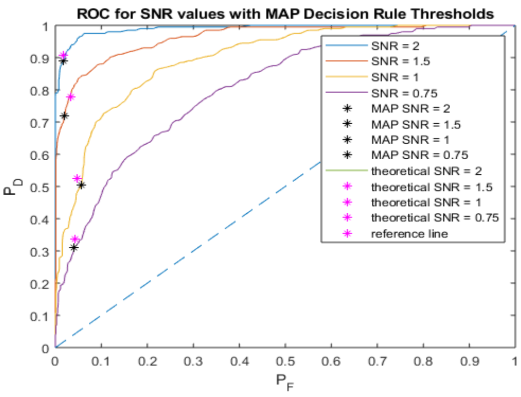


Fig. 2. ROC for various SNR values

- Modification of the Cost Assignment (c)

The threshold in the decision rule takes into account two factors – cost structure and the prior probabilities. In this modification, the cost structure is varied to have the form as follows.

*Cost for "miss" = 10 * Cost for "false alarm" , or*

$$C_{01} = 10\,C_{10},\ such\ that\ \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ 1 & 0 \end{bmatrix}$$

Given a new condition, the threshold for the MAP decision rule decreases by a factor of 10.

Determine the threshold η:

$$\frac{(C_{01}-C_{11})P_{Y|H}(y\,|H_1)}{(C_{10}-C_{00})P_{Y|H}(y\,|H_0)} \gtrless \frac{P(H_0)}{P(H_1)} \equiv \eta, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

$$\frac{10\,P_X(X=y-A)}{P_X(X=y)} \gtrless \frac{0.8}{0.2} = 4, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

$$\frac{P_X(X=y-A)}{P_X(X=y)} \gtrless \frac{4}{10} = 0.4, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

$$\therefore \mathrm{y} \gtrless \frac{A}{2} + \frac{\sigma_x{}^2 \ln(0.4)}{A} \equiv \Gamma, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

Performing the same procedure as in (b) but conditioned on the new threshold, the new ROC seems to follow similar slope variations as the one with initial cost structure of 1. However, there is a slight decrease in detection because there is more cost associated with the decision, and as the value increases, closer to the new MAP threshold, the detection level increases, as expected.

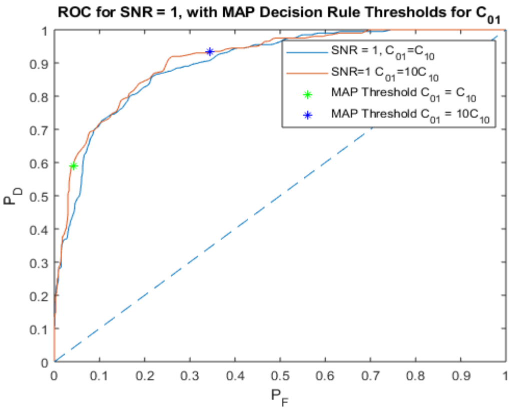


Fig. 3. ROC for SNR = 1

- Expected Cost for A-Priori Probability Range (d)

Given prior probabilities Pr[H$_0$] = p, Pr[H$_1$] = 1- p, and the new cost structure from (c), expected cost for the decision is plotted by sweeping through a range of prior probability, p. The expected cost E[CD(y)] is now the sum of the second term and 10 times the first from the previous derivation:

$$\begin{aligned} \mathrm{E}[C_D(y)] = {} & C_{00}\,\mathrm{Pr}[Choose\ H_0, H_0\ true] \\ & + C_{01}\,\mathrm{Pr}[Choose\ H_0, H_1\ true] \\ & + C_{10}\,\mathrm{Pr}[Choose\ H_1, H_0\ true] \\ & + C_{11}\,\mathrm{Pr}[Choose\ H_1, H_1\ true] \\ = {} & 10\mathrm{Pr}[Choose\ H_0, H_1\ true] + \mathrm{Pr}[Choose\ \mathrm{H}_1, \mathrm{H}_0\ true] \end{aligned}$$

$$\mathrm{y}(\mathrm{P}(H_0)) \gtrless \frac{A}{2} + \frac{\sigma_x{}^2 \ln(\eta)}{A} \equiv \Gamma, \qquad \left(for\ \frac{H_1}{H_0}\right)$$

$$\text{where } \eta = \frac{\mathrm{P}(H_0)}{\mathrm{P}(H_1)} = \frac{p}{10(1-p)}.$$

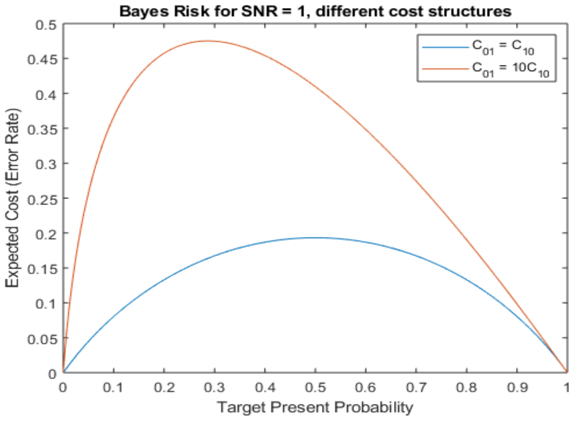


Fig. 4. Expected Cost per Prior Probability

Since when $\Gamma \to -\infty, (\eta \to -\infty)$, or $\mathrm{p} \to 0$, the target will be more likely to be (always) present and thus choose H1, and when $\Gamma \to \infty, (\eta \to \infty)$, or $\mathrm{p} \to 1$, the target will be less present (as ln(1/0) diverges), and thus choose H0, regardless of the value of received signal y.

In general in an ROC plot, the endpoints at (0,0) and (1,1) represent trivial classifiers. The origin (0,0) implies classification of observations as always negative (H$_0$), and (1,1) represents classifying as always positive (H$_1$). The cost curves for these classifiers are the diagonal lines from (0, 0) to (1, 1) (the always-negative classifier) and from (0, 1) to (1, 0) (the always-positive classifier). The operating range of a classifier is the set of operating points (target present probabilities) for which the classifier outperforms both trivial classifiers[2].

A') Modified Hypotheses for Different Noise Models (e)

- Conditions

The two hypotheses – H$_1$ and H$_0$ are modified to have different variances, keeping the same constant mean of A. The variances for the received signal in both hypotheses are assumed to derive from a zero-mean Gaussian distribution, and the prior probabilities are kept the same. Instead of explicitly stating the variance for the negative case (H$_0$), the ratio of variances are specified, namely ZXR – Z to X variance ratio, similar to SNR.

H_1: Target present; $Y = A + X$
H_0: Target not present; $Y = A + Z$,
where $X \sim N(0, \sigma_x{}^2), Z \sim N(0, \sigma_z{}^2), such\ that\ \sigma_z{}^2 > \sigma_x{}^2$
$P\,[H_0] = 0.8, \qquad P\,[H_1] = 0.2$
Let $\sigma_x{}^2 = \alpha$, and thus $\text{SNR} = \frac{A}{\sigma_x{}^2} = \beta$,
and $\text{ZXR} = \frac{\sigma_z{}^2}{\sigma_x{}^2} = \gamma$

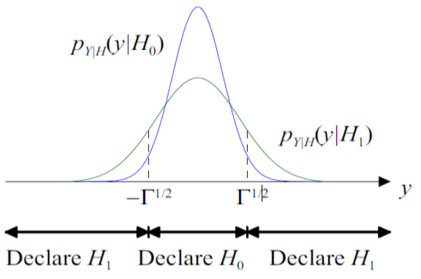
- Hypotheses

With the modified model, define the hypotheses H_1 and H_0 to have mean of A and each variance of X and Z, respectively:

$$H_1: p_{Y|H_1}(y \mid H_1) = N(y;\ m_1, \sigma_1) = N(y;\ A, \sigma_x)$$
$$= \frac{1}{\sqrt{2\pi\sigma_1{}^2}} - \exp\left(\frac{(y-m_1)^2}{2\,\sigma_1{}^2}\right) = \frac{1}{\sqrt{2\pi\sigma_x{}^2}} - \exp\left(\frac{(y-A)^2}{2\sigma_x{}^2}\right)$$
$$H_0: p_{Y|H_0}(y \mid H_0) = N(y;\ m_0, \sigma_0) = N(y;\ A, \sigma_z)$$
$$= \frac{1}{\sqrt{2\pi\sigma_0{}^2}} - \exp(\frac{(y-m_0)^2}{2\sigma_0{}^2}) = \frac{1}{\sqrt{2\pi\sigma_z{}^2}} - \exp(\frac{(y-A)^2}{2\sigma_z{}^2})$$

- MAP decision rule for target detection

This is a case for different variances, same means for scalar Gaussian Detection, such that:



* Detection Theory[3]

Fig. 5. Different $\sigma_0{}^2$, Same Means for 2-hypotheses scalar Gaussian

Determine Bayes Risk (LRT):

$$L(y) = \left[\frac{\frac{1}{\sqrt{2\pi\sigma_x{}^2}} - \exp(\frac{(y-A)^2}{2\sigma_x{}^2})}{\frac{1}{\sqrt{2\pi\sigma_z{}^2}} - \exp(\frac{(y-A)^2}{2\sigma_z{}^2})}\right] \gtrless \eta \left(for\ \frac{H1}{H0}, for\ some\ threshold\ \eta\right)$$

Taking natural logarithmic to eliminate exponentials,

$$-\frac{(y-m_1)^2}{2\sigma_1{}^2} + \frac{(y-m_0)^2}{2\sigma_0{}^2} \gtrless \ln\left(\frac{\sigma_1}{\sigma_0}\eta\right), \qquad (for\ \frac{H_1}{H_0})$$

For $\sigma_0{}^2 = \sigma_z{}^2 > \sigma_x{}^2 = \sigma_1{}^2\ and\ m_0 = m_1 = A$,

$$(y-A)^2 \gtrless 2\left(\frac{\sigma_x{}^2\sigma_z{}^2}{\sigma_x{}^2 - \sigma_z{}^2}\right)\ln\left(\frac{\sigma_x}{\sigma_z}\eta\right) \equiv \Gamma', \qquad (for\ \frac{H_1}{H_0})$$

where $\eta = \frac{0.8}{0.2} = 4$, with $|\Gamma'^{1/2}| = \sqrt{2\left(\frac{\sigma_x{}^2\sigma_z{}^2}{\sigma_x{}^2-\sigma_z{}^2}\right) ln\left(\frac{\eta}{\left(\frac{\sigma_z}{\sigma_x}\right)}\right)}$

* Note that the decision regions are no longer possible to be determined from simple linear interpolation, and the decision rule is a nonlinear function of the observation of the received signal y.

Similar to (a),
Probability of error is the sum of probabilities of a "miss" and a "false alarm":

$$\Pr[Error] = \Pr[Choose\ H_0, H_1\ true]$$
$$+ \Pr[Choose\ \mathrm{H}_1, \mathrm{H}_0\ true]$$
$$= P_M P_1 + P_F P_0\ (= (1 - P_D)P_1 + P_F P_0)$$
$$= \Pr[y < \Gamma | H_1\ true]\Pr[H_1] + \Pr[y > \Gamma | H_0\ true]\Pr[H_1]$$

The similar procedure as in (a) is performed – random variable from a Gaussian distribution of each hypothesis case is used to generate a pdf of the "declared" distribution, from which the error is calculated. The theoretical value is determined by computing "miss," by subtracting the cumulative probabilities for decision H1 from total probability of 1, and "false alarm," by integrating under the curve between +/- $\Gamma^{1/2}$ bounds.

Setting ZXR = 30, SNR = 1, and variance of X = 3, the probabilities of experimentally determined error and the theoretical exhibit a relatively minor difference as well, simil ar to (a). The deviation is 0.0211, which is (0.2408-0.2)/0.2408*100% = 16.9%.

It is interesting to also note that the ZXR ratio below 4 yields a constant value of 0.2 error, as the decision rule always results in choosing H0, which is trivial case. This is because as the ratio falls below η, the threshold, logarithm term increases to a large value, which increases the decision for H0 which lies in between the negative and positive square root of the decision value $\Gamma'^{\frac{1}{2}}$. On the other hand, if the ratio is greater than the threshold, either decision is likely, and thus becomes nontrivial. The larger value the ratio reaches, the more likely it will choose the decision H1, and thus the ROC will tend more towards the true positive ($P_{Detection}$) faster at the given threshold, η. As Figure 6 shows, ZXR of 16 or below falls below the lower bound line ($P_D = P_F$) for lower values, due to constant error of 0.2, whereas for higher values of ZXR yields better probability of detection.

Experimental Error
For $\text{ZXR} = 40 > 16, \Pr[Error] = 0.2$
Theoretical Error
For $\text{ZXR} = 40 > 16, \Pr[Error] = 0.2408$

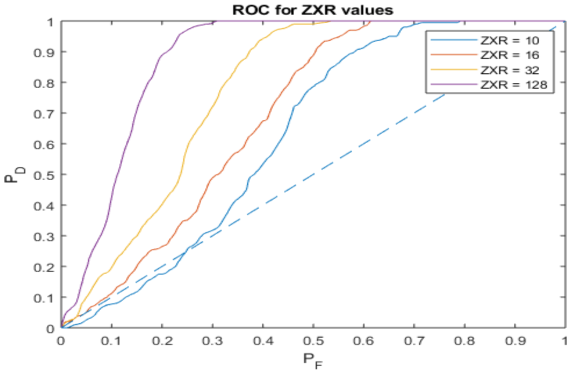


Fig. 6. ROC for various ZXR values

B. *Pattern Classification*

- Conditions and Procedure

The data set for Iris were retrieved from UCI Machine Learning Repository[1], in which the following attribute data are collected:

TABLE I. IRIS DATA BY SPECIES

Type	*Iris.mat data*		
	Data Type	*Rows*	*Columns*
Features	- sepal length [cm]	150	1
	- sepal width [cm]		2
	- petal length [cm]		3
	- petal width [cm]		4
Corresp. Labels	- 3 class ('labels'): Iris Setosa (label 1) Iris Versicolour (label 2) Iris Virginica (label 3)	150	1

The implemented classification (Bayes classifier) is based on the assumptions that the observations follow multivariate distributions given class membership, and the features composing the observations are independent of each other. This allows for use of mvnpdf(data, mean, std_dev) function in MATLAB, which returns a probability density function of the multivariate Gaussian distribution for the input data points with the covariance determined from the input standard deviation.

While the assumption of independence among the observations could yield respectable performance in predictions, it imposes some difficulty upon estimating the likelihood probabilities (class-conditional densities) for higher dimensional data due to the curse of dimensionality, where a large number of training instances are necessary for a reliable estimate of the corresponding multidimensional probability density function assuming that features could be statistically dependent on each other. For the purpose of initial exploration with pattern classification rather than using a neural network to estimate with a lower error probability, the species were classified simply with MAP estimates using a multivariate normal probability density function.

- Confusion matrix

A confusion matrix, initially developed by Kohavi and Provost in 1998, is a matrix that holds information about the actual and predicted classifications performed by a classifier model. As Table II demonstrates, the data, typically classes into which data are classified, are represented and compared accordingly, where actual and predicted results are of same class is noted as "true," and else if different. The terminology for denoting each case is as follows.

(i) True positive: correctly predicted event

(ii) False positive: incorrectly predicted event

(iii) True negative: correctly predicted no-event

(iv) False negative: incorrectly predicted no-event

This chart informs types of errors the classifier has made in the predictions.

TABLE II. CONFUSION MATRIX SAMPLE

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

- Procedure

Upon loading the data set, it is separated into a train set (1/3) and a test set (2/3), and the test set is kept in place for application of the predicted probability model at the end. On the train data set, the prior probabilities were calculated for each species class by marking the labels vector with 1, 2, and 3 and counting the frequencies for each. Treating the labels data set with 3 different species classes each as a multivariate Gaussian (normal) distribution, the mean and variance for each features data set (corresponding to the index of the labels data set) are used to formulate each multivariate normal pdf, which is then multiplied by the previously determined prior probability, as described by

$$\Pr[y; H_i] = \Pr[H_i] * \Pr[y|H_i]$$

Then for the superimposed 3 pdf's, the row and column indices of each row-wise maximum probability are stored; the column index in this case would be the "class" of the pdf (1, 2, or 3), which correspond to the labels data set values. This set of labels data values taken at the maximum of the three generated pdf's (predicted) is sorted by ascending order the index at which maximum values of each row of the 3 generated pdf's occur. Then for each label in the test set, it is compared against the MAP predicted label, and added to the confusion matrix. The probability of error is found from the confusion matrix by determining how many data points of the predicted do not match the actual class.

- Results

After repeating the procedure for a number of iterations and normalizing the confusion matrix, the resulting matrix is as shown below. The MAP classifier predicts species class 1 correctly for all iterations, while it incorrectly predicts 2 as 3, and 3 as 2 for some times. The overall error for the entire iterations is found to be 0.1768, or approximately 17%, which is reasonable, but may be a little high for a practical classifier. As this is a simple Bayes classifier implemented for independent observations specifically, the performance of this model was not expected to be as high. However, an advantage of this classifier would be that despite somewhat high error rate, the complexity in computation is relatively low.

TABLE III. RESULTING CONFUSION MATRIX

	Predicted		
Actual	1	0	0
	0	0.8801	0.05692
	0	0.1199	0.9431

- Approaches for improvement

For improved pattern classification applications, a neural network with complex algorithmic approach, or assumptions of distributions other than Gaussian could be employed. Besides implementation of other machine learning networks, a simpler measure would be increasing then number of training data points for better predictions.

References

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Drummond, Chris, and Robert C. Holte. “Cost Curves: An Improved Method for Visualizing Classifier Performance.” Machine Learning, vol. 65, no. 1, 2006, pp. 109–110., doi:10.1007/s10994-006-8199-5.

[3] Karl, W. Clem. “Detection Theory.” STOCHASTIC PROCESSES Class Notes, edited by D. Castanon, Boston University ECE Dept., 2011, pp. 123–154.

Contents

```
% ECE302 - Stochastic Processes
% Andy Jeong / Spring 2019
% Project 3: Detection Theory
clear all; close all; clc;
```

1 - Radar Detection

1 - (a)

```
% (1) Set arbitrary value for variance of x and SNR
% (2) Generate Gaussian random variable (number) for H0 and H1,
%     given target is not present 80% of the time
% (3) Determine PDFs for Pr( declare H0 | H1 true ) and Pr(H1 | H1) for target present cas
e(i),
%     and Pr(H0 | H0) and Pr(H1 | H0) for target not present case(ii)
% (4) Determine errors for miss (case i) and false alarm (case ii)
iterations = 1000;
varianceX = 3;
stdX = sqrt(varianceX);
SNR = 1;
A = SNR*varianceX;
priorH0 = 0.8;
priorH1 = 1 - priorH0;

% Generate random variables for H0, H1 true cases
% to input for generating Gaussian distributions below
randVarH0 = normrnd(0, stdX, [int32(priorH0 * iterations), 1]);
randVarH1 = normrnd(A, stdX, [int32(priorH1 * iterations), 1]);

% Note: for assumed "true" value (=randVar) determined from a Gaussian
%       distribution, declare it to be H1 or H0 by fitting into Gaussian pdf

% For true H1 (target present) (case i)
% detection (target present) - true positive
P_declareH1_H1true = normpdf(randVarH1, A, stdX);
% miss - true negative
P_declareH0_H1true = normpdf(randVarH1, 0, stdX);
% P(errorH1) = P(tp) - P(tn)
Perror_declareH1 = P_declareH1_H1true * priorH1 - P_declareH0_H1true * priorH0;

% For true H0 (target NOT present) (case ii)
% detection (target NOT present) - false positive
P_declareH0_H0true = normpdf(randVarH0, 0, stdX);
% false alarm - false negative
P_declareH1_H0true = normpdf(randVarH0, A, stdX);
```

```
% P(errorH0) = P(fp) - P(fn)
Perror_declareH0 = P_declareH0_H0true * priorH0 - P_declareH1_H0true * priorH1;

% for declaring H0, H1 cases (i), (ii),
% if Pr(H1|H1), Pr(H0|H0) < 0, there is error present for that case
% (function) sign(X):: 1 if X > 0 / 0 if X == 0 / -1 if X < 0
s = sign([Perror_declareH0; Perror_declareH1]);
% calculate total Pr(error)
% by counting number of errors and dividing by # of iterations
Perror = sum(s(:) == -1)/iterations

% Theoretical Error calculation
% MAP (=MPE) decision rule
eta = priorH0/priorH1; % 0.8/0.2 = 4
% MAP decision rule threshold
MAP = A/2 + varianceX * log(eta)/A;
% probability of y | H1 (detection) from gamma to infinity
P_detection = 1 - normcdf(MAP, A, stdX);
% probability of y | H1 (miss) from -infinity to gamma
P_miss = 1 - P_detection;
% probability of y | H0 (false alarm) from gamma to infinity
P_falseAlarm = 1 - normcdf(MAP, 0, stdX);
P_error_theoretical = P_miss * priorH1 + P_falseAlarm * priorH0
```

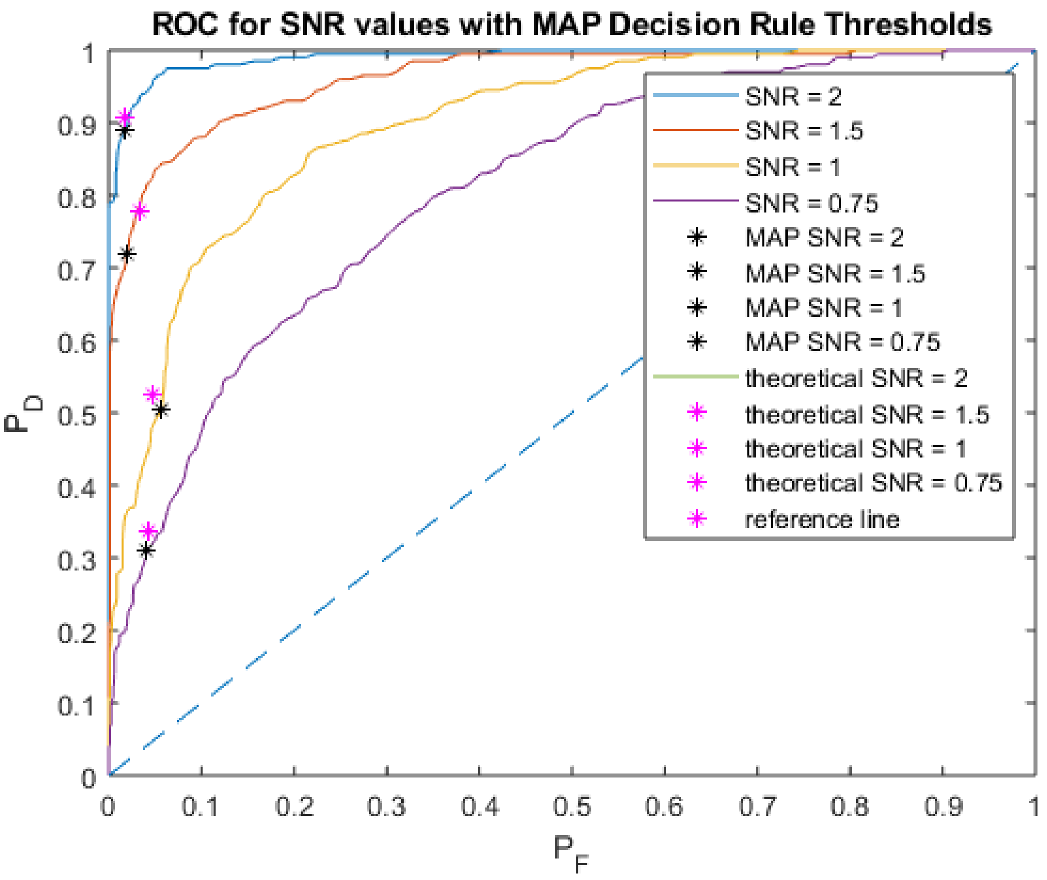
```
Perror =

    0.1320


P_error_theoretical =

    0.1330
```

1 - (b)

Receiver Operating Characteristic (curve) ("ROC") for various SNRs

```
iterations = 1000;
varianceX = 3;
priorH0 = 0.8;
priorH1 = 1 - priorH0;

% Set of SNR values, whose ROC will be plotted
SNR1 = 2;
SNR2 = 1.5;
SNR3 = 1;
SNR4 = 0.75;
% plot points for MAP decision rule threshold
A = [SNR1 SNR2 SNR3 SNR4] .* varianceX;
eta = priorH0/priorH1;
MAP = A/2 + varianceX .* log(eta)./A;
% obtain P_{detection}, P_{falsealarm} for various MAP threshold values
% for each SNR
[Pd0, Pfa0, idx0, idxMAP0] = getROC(varianceX, SNR1, priorH0, priorH1, iterations, MAP(1))
;
[Pd1, Pfa1, idx1, idxMAP1] = getROC(varianceX, SNR2, priorH0, priorH1, iterations, MAP(2))
;
```

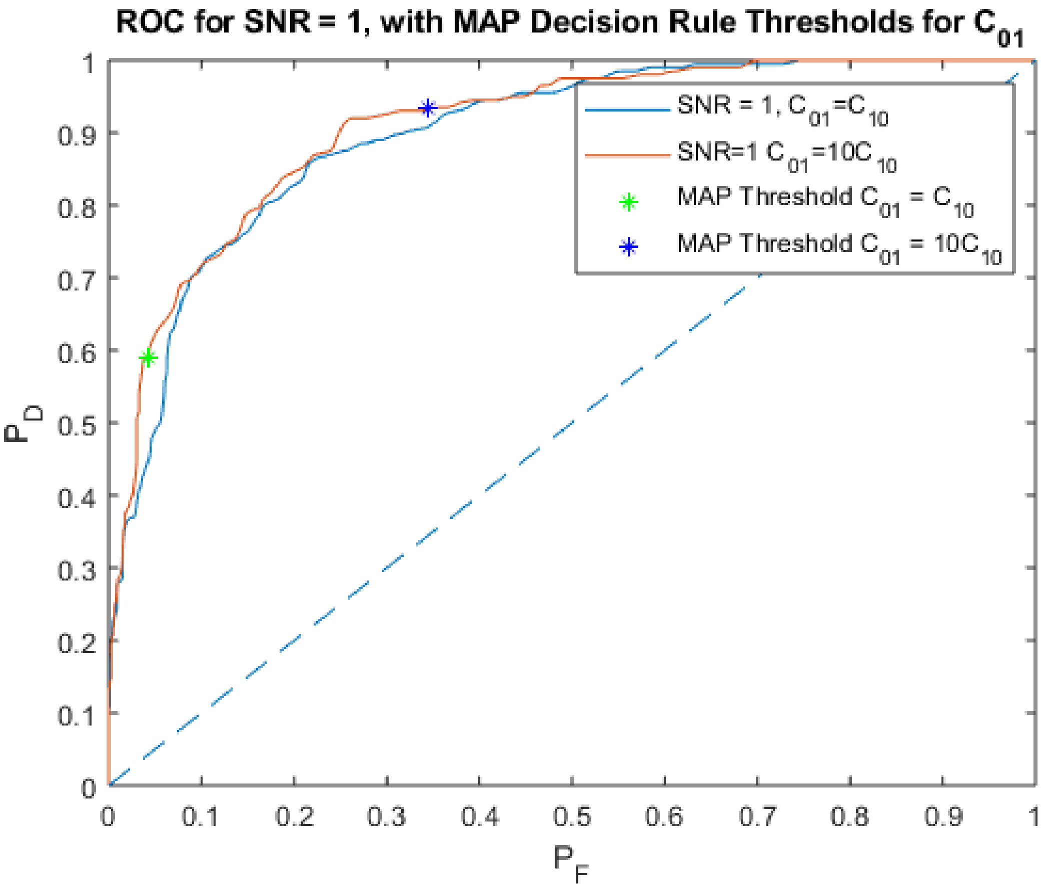
```
[Pd2, Pfa2, idx2, idxMAP2] = getROC(varianceX, SNR3, priorH0, priorH1, iterations, MAP(3))
;
[Pd3, Pfa3, idx3, idxMAP3] = getROC(varianceX, SNR4, priorH0, priorH1, iterations, MAP(4))
;

% plot P_{detection} vs. P_{false alarm} for the specified 4 SNR values
plot(smooth(Pfa0),smooth(Pd0), smooth(Pfa1), smooth(Pd1), ...
    smooth(Pfa2),smooth(Pd2), smooth(Pfa3), smooth(Pd3));
hold on;
plot(Pfa0(idxMAP0), Pd0(idxMAP0),'k*',Pfa1(idxMAP1), Pd1(idxMAP1),'k*', ...
    Pfa2(idxMAP2), Pd2(idxMAP2),'k*',Pfa3(idxMAP3), Pd3(idxMAP3),'k*');

xlim([0,1]); ylim([0,1]);
xlabel('P_F')
ylabel('P_D')
title('ROC for SNR values with MAP Decision Rule Thresholds')

% determine theoretical ROC
% P_D = Q(gamma/stddev), P_FA = Q(gamma/stddev - d),
% where d is a measure of the relative separation of the means under each
% hypothesis, d^2 =(mean1-mean0)^2 / variance
d = A/stdX;
q_distribution = 1 - normcdf([0:0.01:1],0,1);
plot(q_distribution);
Pfa0_theoretical = qfunc(MAP(1)/stdX);
Pd0_theoretical = qfunc(MAP(1)/stdX - d(1));
Pfa1_theoretical = qfunc(MAP(2)/stdX);
Pd1_theoretical = qfunc(MAP(2)/stdX - d(2));
Pfa2_theoretical = qfunc(MAP(3)/stdX);
Pd2_theoretical = qfunc(MAP(3)/stdX - d(3));
Pfa3_theoretical = qfunc(MAP(4)/stdX);
Pd3_theoretical = qfunc(MAP(4)/stdX - d(4));
plot(Pfa0_theoretical, Pd0_theoretical, 'm*', ...
    Pfa1_theoretical, Pd1_theoretical, 'm*', ...
    Pfa2_theoretical, Pd2_theoretical, 'm*', ...
    Pfa3_theoretical, Pd3_theoretical, 'm*');
% reference line y = x; this is the lower bound for the curve
ref=refline(1,0);
ref.LineStyle='--';

legend('SNR = 2', 'SNR = 1.5', 'SNR = 1', 'SNR = 0.75', ...
    'MAP SNR = 2', 'MAP SNR = 1.5', 'MAP SNR = 1', 'MAP SNR = 0.75', ...
    'theoretical SNR = 2', 'theoretical SNR = 1.5', 'theoretical SNR = 1', ...
    'theoretical SNR = 0.75', 'reference line');
```



1 - (c)

ROC with different cost structures

```
SNR = 1; % plot ROC for SNR = 1
A = SNR * varianceX;
eta = priorH0/priorH1/10; % threshold
newMAP = A/2 + varianceX .* log(eta)./A;
% obtain P_{detection} and P_{false alarm}
% with new MAP threhsold condition as input
[Pd4, Pfa4, idx4, idxMAP4] = getROC(varianceX, SNR, priorH0, priorH1, iterations, newMAP);

figure;
plot(smooth(Pfa2), smooth(Pd2), smooth(Pfa4), smooth(Pd4)); hold on;
% idx4: from newly set MAP decision rule from the newly set cost structure
% idxMAP4: from original MAP decision rule
plot(Pfa4(idxMAP4), Pd4(idxMAP4),'g*', Pfa4(idx4), Pd4(idx4),'b*');
ref=refline(1,0);
ref.LineStyle='--';
xlim([0,1]); ylim([0,1]);
xlabel('P_F')
ylabel('P_D')
title('ROC for SNR = 1, with MAP Decision Rule Thresholds for C_{01}')
legend('SNR = 1, C_{01}=C_{10}','SNR=1 C_{01}=10C_{10}', 'MAP Threshold C_{01} = C_{10}',
'MAP Threshold C_{01} = 10C_{10}');
```

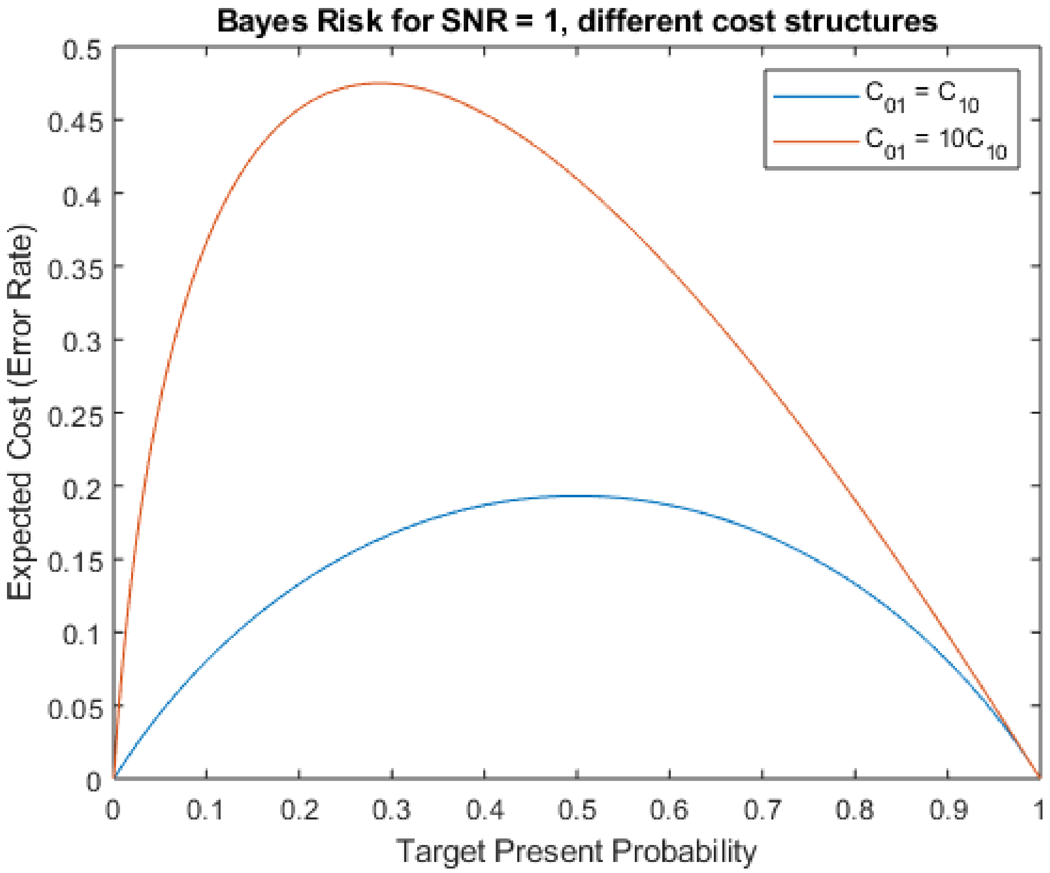


1 - (d)

Select an SNR value and plot the value of the expected cost for a range of a priori target present probabilities from 0 to 1

```
% range of prior probability values to iterate through
priorH1 = linspace(0,1,1000);
priorH0 = 1 - priorH1;
cost1 = [0 1; 1 0]; % previous (original) cost structure
C00 = 0; C01 = 10; C10 = 1; C11 = 0; % C = 1 - Kronecker delta(i,j)
cost2 = [C00 C01; C10 C11]; % newly defined cost structure

SNR = 1;
stdX = sqrt(varianceX);
A = SNR*varianceX;

% compute expected cost curve for the original threshold
eta1 = priorH0./priorH1;
MAP1 = A/2 + varianceX .* log(eta1)./A;
% obtain Pd, Pfa from normcdf since the custom function doesn't accept
% vector from of MAP values
Pd1 = 1 - normcdf(MAP1, A, stdX);
Pfa1 = 1 - normcdf(MAP1, 0, stdX);
% equation for expected cost of making a certain decision
% refer to reference notes for details
expected_cost1 = cost1(1,1).*priorH0 + cost1(1,2).*priorH1 + ...
    (cost1(2,1)-cost1(1,1)).*priorH0.*Pfa1 - (cost1(1,2)-cost1(2,2)).*priorH1.*Pd1;

% compute expected cost curve for the new threshold (from new costs)
eta2 = priorH0./priorH1./10;
MAP2 = A/2 + varianceX .* log(eta2)./A;
Pd2 = 1 - normcdf(MAP2, A, stdX); % H1 conditional density
Pfa2 = 1 - normcdf(MAP2, 0, stdX); % H0 conditional density
```

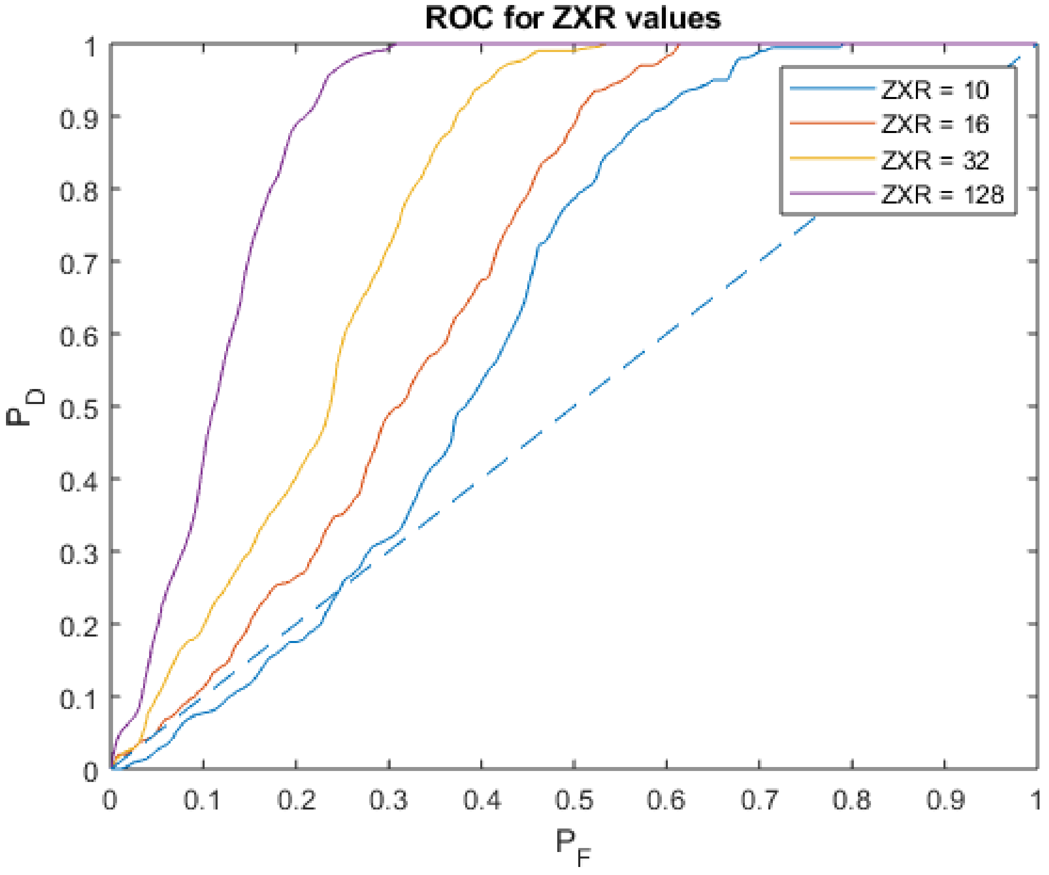
```
expected_cost2 = cost2(1,1)*priorH0 + cost2(1,2).*priorH1 + ...
    (cost2(2,1)-cost2(1,1)).*priorH0.*Pfa2 - (cost2(1,2)-cost2(2,2)).*priorH1.*Pd2;

figure;
plot(priorH1, expected_cost1, priorH1, expected_cost2);
xlabel('Target Present Probability');
ylabel('Expected Cost (Error Rate)');
title('Bayes Risk for SNR = 1, different cost structures');
legend('C_{01} = C_{10}','C_{01} = 10C_{10}');
```



1 - (e)

```
% similar to procedure in (a)
iterations = 1000;
% ZXR: ratio of variances of Z to X
ZXR = 30;
varianceX = 3;
varianceZ = ZXR * varianceX;
stdX = sqrt(varianceX);
stdZ = sqrt(varianceZ);
SNR = 1;
A = SNR*varianceX;
priorH0 = 0.8;
priorH1 = 1 - priorH0;

% Generate random variables for H0, H1 true cases
% to input for generating Gaussian distributions below
randVarH0 = normrnd(0, stdZ, [int32(priorH0 * iterations), 1]);
randVarH1 = normrnd(A, stdX, [int32(priorH1 * iterations), 1]);

% Note: for assumed "true" value (=randVar) determined from a Gaussian
%       distribution, declare it to be H1 or H0 by fitting into Gaussian pdf
% For true H1 (target present) (case i)
% detection (target present) - true positive
P_declareH1_H1true = normpdf(randVarH1, A, stdX);
% miss - true negative
P_declareH0_H1true = normpdf(randVarH1, A, stdX);
% P(errorH1) = P(tp) - P(tn)
Perror_declareH1 = P_declareH1_H1true * priorH1 - P_declareH0_H1true * priorH0;

% For true H0 (target NOT present) (case ii)
% detection (target NOT present) - false positive
P_declareH0_H0true = normpdf(randVarH0, A, stdZ);
% false alarm - false negative
P_declareH1_H0true = normpdf(randVarH0, A, stdZ);
% P(errorH0) = P(fp) - P(fn)
Perror_declareH0 = P_declareH0_H0true * priorH0 - P_declareH1_H0true * priorH1;

% for declaring H0, H1 cases (i), (ii),
% if Pr(H1|H1), Pr(H0|H0) < 0, there is error present for that case
% (function) sign(X):: 1 if X > 0 / 0 if X == 0 / -1 if X < 0
s = sign([Perror_declareH0; Perror_declareH1]);
% calculate total Pr(error)
% by counting number of errors and dividing by # of iterations
Perror = sum(s(:) == -1)/iterations

% theoretical
% MAP (MPE) decision rule
eta = priorH0/priorH1; % 0.8/0.2 = 4
% MAP decision rule threshold (sqrt(gamma))
MAP = sqrt(2*varianceX*varianceZ/(varianceX-varianceZ)*log(eta*varianceX/varianceZ));
% probability of abs(y-A) < MAP | H1 (miss) - choosing H1
P_miss = 1 - (normcdf(MAP, 0, stdX) - normcdf(-MAP, 0, stdX));
% probability of abs(y-A) < MAP | H0 (false alarm) - choosing H0
P_falseAlarm = normcdf(MAP, 0, stdZ) - normcdf(-MAP, 0, stdZ);
P_error_theoretical = P_miss * priorH1 + P_falseAlarm * priorH0

% (b)
% range of ZXR ratio values whose ROCs will be plotted
ZXR1 = 10;
ZXR2 = 16;
ZXR3 = 32;
ZXR4 = 128;
priorH0 = 0.8;
priorH1 = 1 - priorH0;

% obtain P_{detection} and P_{false alarm}
% by inputting different ZXR values
[Pd1, Pfa1] = getROC_diffVar(varianceX, ZXR1, SNR, priorH0, priorH1, iterations);
[Pd2, Pfa2] = getROC_diffVar(varianceX, ZXR2, SNR, priorH0, priorH1, iterations);
[Pd3, Pfa3] = getROC_diffVar(varianceX, ZXR3, SNR, priorH0, priorH1, iterations);
[Pd4, Pfa4] = getROC_diffVar(varianceX, ZXR4, SNR, priorH0, priorH1, iterations);
plot(smooth(Pfa1), smooth(Pd1), smooth(Pfa2), smooth(Pd2), ...
    smooth(Pfa3), smooth(Pd3), smooth(Pfa4), smooth(Pd4));
% reference line to show lower bound for ROC curves
ref = refline(1,0);
ref.LineStyle='--';
xlabel('P_{F}');
ylabel('P_{D}');
title('ROC for ZXR values');
legend('ZXR = 10','ZXR = 16','ZXR = 32','ZXR = 128');
```

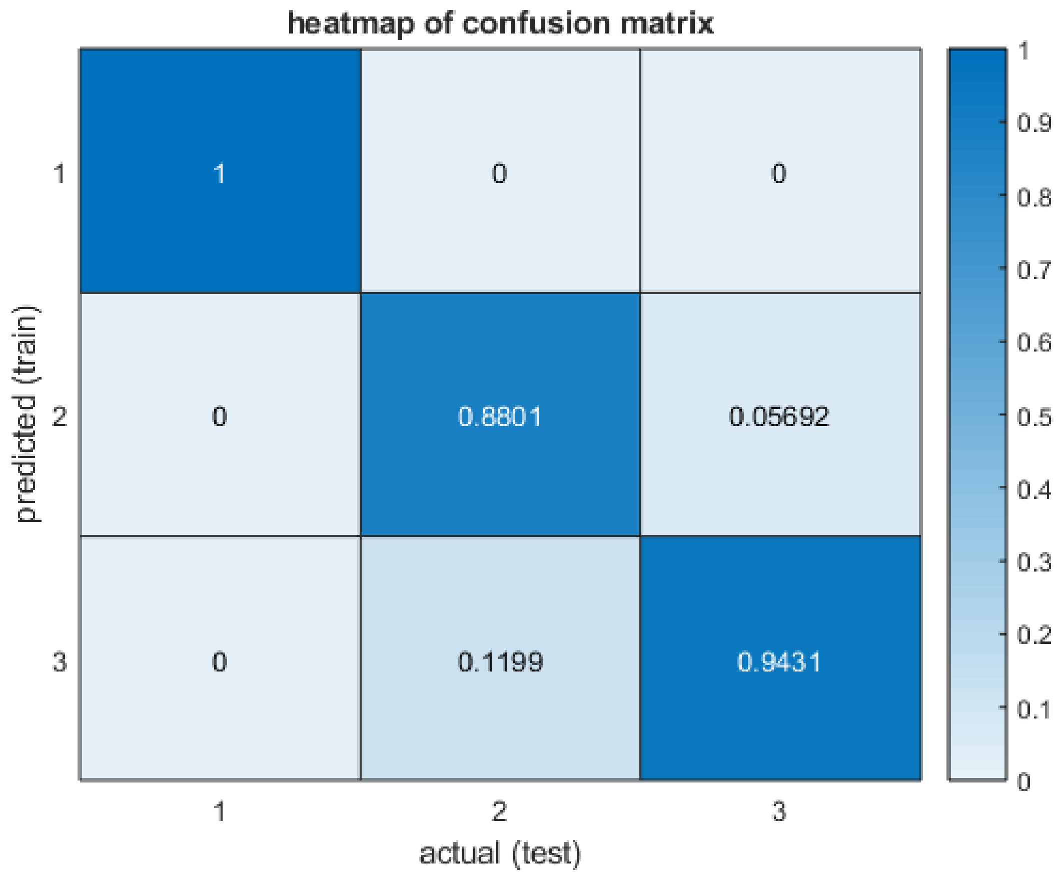
```
Perror =

    0.2000


P_error_theoretical =

    0.2408
```



2 - Pattern Classification

```
% load the data set
% initialize confusion matrix 3x3 because there are three labels
% row: actual, column: predicted labels
iris = load('Iris.mat');
confusion_matrix = zeros(3);

% iterate 1000 times
for iter = 1:1000
    % (1) randomly retrieve data from iris from randomly permuted indices
    % iris.features: data points per label range from [0.1, 7.9]
    % iris.labels: [1], [2], [3] classes
    rand_indices = randperm(length(iris.features));
    features = iris.features(rand_indices,:);
    labels = iris.labels(rand_indices);

    % (2) split into train and test data (split into 1/3 and 2/3)
    test_features = features(1:floor(end/3),:);
    test_labels = labels(1:floor(end/3),:);
    train_features = features(floor(end/3)+1:end,:);
    train_labels = labels(floor(end/3)+1:end,:);
    % (3) find indices where the labels are marked as 1, 2, or 3 for the
    %     training set
    index1_train = find(train_labels==1);
    index2_train = find(train_labels==2);
    index3_train = find(train_labels==3);

    % compute the length of each label (class)
    length_labels_train = [length(index1_train), ...
        length(index2_train), length(index3_train)];
    % (4) the number of frequency of appearance of each label
    % over the total frequency gives prior probabilities for each label
    priors = length_labels_train/sum(length_labels_train);

    % (5) sort features into labels (classes) by the found indices
    features1_train = train_features(index1_train,:);
    features2_train = train_features(index2_train,:);
    features3_train = train_features(index3_train,:);

    % mvnpdf: multivariate Gaussian pdf, given data vector, and
    %         mean and variance of the prior set
    % (6) pdf_train gives joint probability density of the distribution and prior
    pdf1_train = priors(1) * mvnpdf(test_features, mean(features1_train), sqrt(var(feature
s1_train)));
    pdf2_train = priors(2) * mvnpdf(test_features, mean(features2_train), sqrt(var(feature
s2_train)));
    pdf3_train = priors(3) * mvnpdf(test_features, mean(features3_train), sqrt(var(feature
s3_train)));

    % (7) Find the location at which the maximum feature of the three pdf's
    % occur; returns (row, column)
    [max_row_train, max_col_train] = find([pdf1_train, pdf2_train, pdf3_train]==max([pdf1_
train, pdf2_train, pdf3_train], [], 2));

    % (8) sort (ascending order) only by max_row_train, which is the first column
    max_by_label_train = sortrows([max_row_train, max_col_train]);
    max_labels_train = max_by_label_train(:,2);

    % (9) increment at each (predicted, actual) position of the confusion matrix
    % for each map predicted label (class)
    % i(row): actual test data, j(col): MAP predicted label
    % length(test_labels) == length(max_labels_train)
    for i = 1:1:length(test_labels)
        idx = [test_labels(i), max_labels_train(i)];
        confusion_matrix(idx(1), idx(2)) = confusion_matrix(idx(1), idx(2)) + 1;
    end
end

% (10) normalized by each column sum (predicted)
normalized = [confusion_matrix(:,1)/sum(confusion_matrix(:,1)), ...
    confusion_matrix(:,2)/sum(confusion_matrix(:,2)), ...
    confusion_matrix(:,3)/sum(confusion_matrix(:,3))]

figure; heatmap(normalized);
title('heatmap of confusion matrix');
xlabel('actual (test)');
ylabel('predicted (train)');

% calculate total probability of error
diagonal = normalized.*eye(3);
error = sum(sum(normalized - diagonal))
```

```
normalized =

    1.0000         0         0
         0    0.8801    0.0569
         0    0.1199    0.9431


error =

    0.1768
```



Published with MATLAB® R2018a