3.2 Discriminant function:

$$g(x) = 1 \circ \frac{P(C_1 \mid x)}{P(C_2 \mid x)}$$

Choose $C_1$ if $g(x) > 0$, else $C_2$

3.3 $R(\alpha_1 \mid x) = 10 \ast P(C_2 \mid x)$;

$$R(\alpha_2 \mid x) = P(C_1 \mid x);$$

Choose $C_1$ if $P(C_1 \mid x) > 10/11$; otherwise $C_2$

4.2 We have the maximization problem as below:

$$\log P(x \mid p_1, \ldots, p_k) = \log \prod_{i=1}^{N} \prod_{k=1}^{K} p_i^{x_i} = \sum_{i=1}^{N} \sum_{k=1}^{K} x_i \log p_i;$$

With restriction $\sum_{i=1}^{k} p_i = 1$

So we need to use Lagrange Multipliers to maximize the following:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} x_i \log p_i + \lambda (1 - \sum_{i=1}^{k} p_i)$$

In order to have MLE, we have:

$$\frac{\partial \log P(x \mid p_1, \ldots, p_k)}{\partial p_i} = \sum_{i=1}^{N} x_i / p_i - \lambda = 0;$$

So we have: $p_i = \sum_{i=1}^{N} x_i / \lambda$

And with the constriction: $\sum_{i=1}^{k} p_i = \sum_{i=1}^{K} \sum_{k=1}^{N} x_i / \lambda = N / \lambda = 1$; we have $\lambda = N$

So $p_i = \sum_{i=1}^{N} x_i / N$
4.3 We can use Box-Muller transform to generate normal sample:

```c
void GenerateNormalSamples()
{
    double samples[N];
    for (int i=0; i<N; i++)
    {
        double u = rand(); // u is a uniformly distributed random variable in the interval (0, 1);
        double v = rand(); // v is a uniformly distributed random variable in the interval (0, 1);
        double temp = sqrt(-2*ln(u))*cos(2*PI*v); //temp is a standard normal variable
        temp = temp/σ + μ; // now we have a normal sample with μ and σ
        samples[i] = temp;
    }
}
```

We can use Mersenne Twister method to generate uniform distributed random variable

```c
int MT[624]
int index = 0

// Initialize the generator from a seed
initializeGenerator(int seed) {
    MT[0] = seed
    for (i = 1; i<624; i++) { // loop over each other element
        MT[i] := last 32 bits of(1812433253 * (MT[i-1] xor (right shift by 30 bits(MT[i-1]))) + i)
    }
}

// Extract a tempered pseudorandom number based on the index-th value,
// calling generateNumbers() every 624 numbers
double extractNumber() {
    if index == 0 {
        generateNumbers()
    }
    double x
    int y = MT[index]
    y = y xor (right shift by 11 bits(y))
    y = y xor (left shift by 7 bits(y) and (2636928640)) // 0x9d2c5680
    y = y xor (left shift by 15 bits(y) and (4022730752)) // 0xefc60000
    y = y xor (right shift by 18 bits(y))
\[
x = \frac{y}{(2^{32})-1}
\]

index := (index + 1) mod 624
return x

// Generate an array of 624 untempered numbers
function generateNumbers() {
    for i from 0 to 623 {
        int y := 32nd bit of(MT[i]) + last 31 bits of(MT[(i+1) mod 624])
        MT[i] := MT[(i + 397) mod 624] xor (right shift by 1 bit(y))
        if (y mod 2) == 1 { // y is odd
            MT[i] := MT[i] xor (2567483615) // 0x9908b0df
        }
    }
}

The following is the code to calculate m and s
void CalculateMeanAndVar(double *samples, int N) {
    double mean = 0;
    double var = 0;
    for (int i=0; i<N; i++)
    {
        mean = mean + samples[i];
    }
    mean = mean/N;
    for (int i=0; i<N; i++)
    {
        var = var + (samples[i] – mean)* (samples[i] – mean);
    }
    var = var/N;
}

If we assume that \( \mu \) has a normal distribution with mean m1 and variance s1, then we can estimate \( \mu \) and s as follows:
void EstimateMean (double *samples, int N, double m1, double s1) {
    double mean = 0;
    double var = 0;
// First, calculate sample mean and variance
for (int i=0; i<N; i++)
{
    mean = mean + samples[i];
}
mean = mean/N;

for (int i=0; i<N; i++)
{
    var = var + (samples[i] - mean) * (samples[i] - mean);
}
var = var/N;

// Then calculate expected mean for \( \mu \)
mean = (N/(var*var)/(N/(var*var) + 1/(s1*s1))*mean + (1/(s1*s1)/(N/(var*var) + 1/(s1*s1))*m1)

4.4 To get the discriminant point, we have:

\[
- \log \sigma_1 - (x - \mu_1)^2 / 2\sigma_1^2 + \log P(C_1) = -\log \sigma_2 - (x - \mu_2)^2 / 2\sigma_2^2 + \log P(C_2);
\]

\[
- \log(\sigma_1 / \sigma_2) + \log(P(C_1) / P(C_2)) = (x - \mu_1)^2 / 2\sigma_1^2 - (x - \mu_2)^2 / 2\sigma_2^2;
\]

\[
2\sigma_1^2\sigma_2^2 \log[\sigma_2 P(C_1)]/\sigma_1 P(C_2)] = (\sigma_2^2 - \sigma_1^2)x^2 - 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)x + (\mu_1^2\sigma_2^2 - \mu_2^2\sigma_1^2)
\]

Let

\[ A = (\sigma_2^2 - \sigma_1^2); B = -2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2); \]

\[ C = \mu_1^2\sigma_2^2 - \mu_2^2\sigma_1^2 - 2\sigma_1^2\sigma_2^2 \log[\sigma_2 P(C_1)]/\sigma_1 P(C_2)]; \]

We have \( x = (B \pm \sqrt{B^2 - 4AC}) / 2A; \)

4.6 I use the above method to generate classes normal samples. One class has mean 0 and variance \( \sigma_1^2 = 1 \). The other class has mean 0 and variance \( \sigma_2^2 = 16 \). Each class has 100 samples. I got a sample variance 0.76 for class1 and for sample variance 14.75 for class2, so the estimated

The computational discriminant values are:

\[
\log \sigma_1 + x^2 / 2\sigma_1^2 = \log \sigma_2 + x^2 / 2\sigma_2^2;
\]
The estimated discriminant values are $\pm 1.53$
The theoretical discriminant values are $\pm 1.72$

7.4 Suppose we have $K$ classes, $N$ samples, and each sample has $D$ features, that is, it is a $D$ dimension vector. The parameters of this problem is $P(C_i)$ (probability of each class) and $P_{id}$ (feature probability of class $i$)
First, we assign each $P(C_i) = 1/K$ and $P_{id} = 1/2$ as initial random value;
Repeat until convergence:

**E-step:** Calculate hidden variable $h_i^t$

$$h_i^t = \frac{\prod_{d=1}^{D} P_{id}^{x_d^t} (1 - P_{id})^{1-x_d^t}}{\sum_{i=1}^{K} \left( \prod_{d=1}^{D} P_{id}^{x_d^t} (1 - P_{id})^{1-x_d^t} \right) P(C_i)};$$

**M step:** Update $P(C_i)$ and $P_{id}$:

$$P(C_i) = \frac{1}{N} \sum_{t=1}^{N} h_i^t ;$$

$$\frac{\partial}{\partial P_{id}} \sum_{t=1}^{N} \sum_{i=1}^{K} \log \prod_{d=1}^{D} p_{id}^{x_d^t} (1 - p_{id})^{1-x_d^t} = \frac{\sum_{i=1}^{N} h_i^t x_d^t}{P_{id}} - \frac{\sum_{i=1}^{N} h_i^t (1 - x_d^t)}{1 - P_{id}} = 0;$$

$$p_{id} = \frac{1}{\sum_{t=1}^{N} h_i^t} \sum_{t=1}^{N} h_i^t x_d^t$$

Project: Distinguish the digit in a handwritten image

Input:
Handwritten digits stretched in a rectangular box 16x16 in a gray scale of 256. Then each pixel of each image was scaled into a boolean (1/0) value using a fixed threshold. Since each image is 16x16, so it has 256 attributes whose value is either 1 or 0.

I use 200 samples as triaing data set and got the result as follows:
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: digits
Instances: 200
Attributes: 257

[attributes list omitted]

Test mode: evaluate on training data

Weka Result:

Correctly Classified Instances 188 94%
Incorrectly Classified Instances 12 6%
Kappa statistic 0.9333
Mean absolute error 0.0129
Root mean squared error 0.1103
Relative absolute error 7.1872%
Root relative squared error 36.7806%
Total Number of Instances 200

=== Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.952</td>
<td>0.006</td>
<td>0.952</td>
<td>0.952</td>
<td>0.952</td>
<td>0.999</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
<td>0.028</td>
<td>0.783</td>
<td>0.9</td>
<td>0.837</td>
<td>0.984</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.95</td>
<td>0.006</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.999</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.95</td>
<td>0</td>
<td>1</td>
<td>0.95</td>
<td>0.974</td>
<td>0.989</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0.95</td>
<td>0.006</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.999</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.95</td>
<td>0</td>
<td>1</td>
<td>0.95</td>
<td>0.974</td>
<td>0.999</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0.9</td>
<td>0.017</td>
<td>0.87</td>
<td>0.9</td>
<td>0.947</td>
<td>0.999</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.017</td>
<td>0.87</td>
<td>1</td>
<td>0.93</td>
<td>0.997</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>0.895</td>
<td>0.006</td>
<td>0.944</td>
<td>0.895</td>
<td>0.919</td>
<td>0.994</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0.95</td>
<td>0</td>
<td>1</td>
<td>0.95</td>
<td>0.974</td>
<td>0.999</td>
<td>9</td>
</tr>
</tbody>
</table>

Weighted Avg. 0.94 0.007 0.945 0.94 0.941 0.996

The first two columns are the TP Rate (True Positive Rate) and the FP Rate (False Positive Rate). TP Rate is the ratio of predicted correctly cases to the total of positive cases. The FP Rate is then the ratio incorrectly predicted cases to the total of cases.

The next two columns are terms related to information retrieval theory. When one is conducting a search for relevant documents, it is often not possible to get to the relevant documents easily or directly. In many cases, a search will yield lots results many of which will be irrelevant. Under these circumstances, it is often impractical to get all results at once but only a portion of them at a time. In such cases, the terms recall and precision...
are important to consider. Recall is the ratio of relevant documents found in the search result to the total of all relevant documents. Thus, higher recall values imply that relevant documents are returned more quickly. A recall of 30% at 10% means that 30% of the relevant documents were found with only 10% of the results examined. Precision is the proportion of relevant documents in the results returned. Thus a precision of 0.75 means that 75% of the returned documents were relevant. Lastly, the F-measure is a way of combining recall and precision scores into a single measure of performance. The formula for it is:

\[
\frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}
\]

Then I try a test set with 10 samples and result is as follows:

```plaintext
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances          10              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               10

=== Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>9</td>
</tr>
</tbody>
</table>

Weighted Avg. 1 0 1 1 1 0
```