UNIVERSITY OF BIRMINGHAM

Machine Learning (Extended)

Computer Based Test 2

Bayesian Classification

November 10, 2016

Thomas Brereton 1708846

Table of Content

1 Abstract **2**

2 Scope **2**

3 Overall Task **3**

4 Maximum Likelihood (ML) **4**
- 4.1 Introduction . . . 4
- 4.2 Results . . . 4
- 4.3 Conclusion . . . 4
- 4.4 Further Comments . . . 5

5 Maximum A Posteriori (MAP) **8**
- 5.1 Introduction . . . 8
- 5.2 Results . . . 8
- 5.3 Conclusion . . . 9
- 5.4 Further Comments . . . 9

6 Maximum Likelihood vs Maximum A Posteriori (ML VS MAP) **10**
- 6.1 Difference of ML and MAP . . . 10
- 6.2 Affects of Difference on this Data . . . 11

7 References **15**

List of Figures

1 Training Data for the 4 methods outlined in this report. Note the imbalanced amount of healthy to diseased patients. . . . 3
2 Maximum Likelihood classification of new data . . . 5
3 Maximum Likelihood without Naive assumption classification of new data . . . 6
4 Comparison of class conditional density contours for each class and with Naive and without Naive assumption . . . 6
5 Comparison of probability contours for each class and with Naive and without Naive assumption . . . 7
6 Maximum A Posteriori classification of new data . . . 9
7 Maximum A Posteriori without Naive assumption classification of new data . . . 10
8 Comparison of training data and classification of new data by MAP and MAPWON . . . 11
9 Comparison of density contours for each class and with Naive and without Naive assumption 12
10 Comparison of probability contours for each class and with Naive and without Naive assumption . . . 13
11 Comparison of classification of new data by ML, MLWON, MAP, and MAPWON . . . 14

List of Tables

1 Classification Differences between Methods . . . 8
2 Classification Differences between Methods . . . 10

1 Abstract

In this computer based test we are asked to perform Bayesian classification on a given data set. The classification methods used are the following:

- Maximum Likelihood (ML)
- Maximum Likelihood without Naive assumption (MLWON)
- Maximum A Posteriori (MAP)
- Maximum A Posteriori without Naive assumption(MAPWON)

We are given a data set of healthy and diseased patients to train the 4 classifiers on. Once trained, we use the classifiers to assess 2000 unclassified data and label each patient (data point / example) as diseased or healthy.

We are also asked to compare and discuss the results of the different methods. In short...

- ML showed general trend of increase in concentration, increased likelihood of diseased patient. If both chemicals went to extreme values, would likely become healthy again. Flaw in ML - MLWON showed similar trend but was thinner and showed a downward trend due to the dependence between attributes. Definitely showed healthy patients again, unlikely.

ML and MAP were very similar. Difference of only 62 and 15 for Naive and without Naive, respectively. Likely because prior has no affect on this amount of data.

The analysis was performed in Matlab, and the code listings can be found in Appendices A, B, C, and D.

2 Scope

The report consists of the following structure.

1. Overall Task

   Details of task set out in the "Computer Based Test 2."

2. Maximum Likelihood (ML)

   Discussion on training the ML with and without the Naive assumption

   Interpretation of results

3. Maximum A Posteriori (MAP)

   Discussion on training the MAP with and without the Naive assumption

   Interpretation of results

4. Maximum Likelihood VS Maximum A Posteriori

   Difference of ML and MAP

   Affects of Difference on this data

3 Overall Task

We are asked to determine if a patient is diseased or healthy from a given data set. A new test is in development and measures the concentration of 2 chemicals in urine samples. 500 patients provide urine samples and also undergo a blood tests to classify if they are diseased or healthy. Thus, we have a 500 by 2 data set which is plotted in Figure 1.

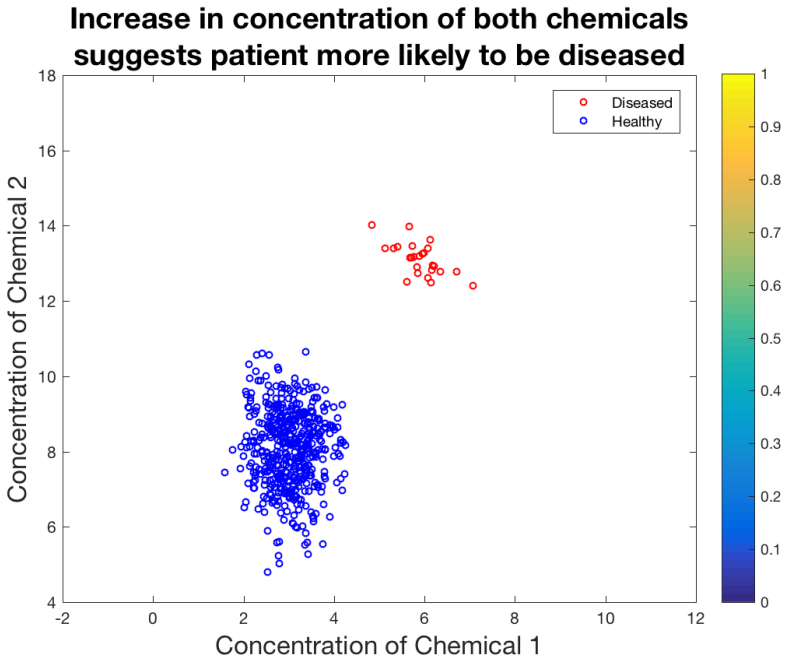


Figure 1: Training Data for the 4 methods outlined in this report. Note the imbalanced amount of healthy to diseased patients.

We use this data set to train the 4 following methods.

- Maximum Likelihood (ML)
- Maximum Likelihood without Naive assumption (MLWON)
- Maximum A Posteriori (MAP)
- Maximum A Posteriori without Naive assumption(MAPWON)

Once trained, we are tasked with classifying 2000 new urine samples. We use the 4 methods outlined previously to determine whether a 'patient' is diseased or healthy.

It is helpful to state a hypothesis for assessment. Considering the data in Figure 1 the hypothesis is, an increase in concentration of chemical 1 and 2 increases the likelihood of a patient being diagnosed as diseased.

4 Maximum Likelihood (ML)

4.1 Introduction

In Task 1, we are asked to "train the Bayesian classifier, with maximum likelihood estimate (ML), on the training data with and without the Naive assumption." The data set is the one shown in 1. Once trained we classify 2000 new data points (patients).

To train ML with the Naive assumption first compute the mean and variance of the data for each class and attribute. As it is Naive, we assume the attributes (chemical 1 and 2) are independent. Armed with the mean, variance, and the new data set we use the Gaussian expression to compute the likelihood of each data point as either diseased or healthy. In this case, we are looking at the maximum likelihood and thus, only use class-conditional likelihood to classify.

To train ML without the Naive assumption is similar to before. We compute the mean as normal but this method requires the covariance not variance of the classes and attributes. This assumes a dependence between the attributes which is useful to look at if there are not many attributes. We then compute the maximum likelihood as normal but accounting for the 3 dimensional matrix due to the covariance.

4.2 Results

Figure 2 shows the likelihood of being diseased or healthy depending on the concentration of the 2 chemicals. It shows there is some relationship with the concentration of the 2 chemicals and if a patient is healthy or diseased. It shows that an increase in chemical concentration insreases likelihood of being diseased to a point. After which the likelihood starts decreasing. This seems like a flaw in the classifier or an error in the measurements but we cannot be certain. To clarify this issue we require further testing over a greater range of chemical concentration.

If the data were to extend further with high concentration in both directions, we would likely see an ellipse or sphere of diseased patients. This spherical shape is due to the naive assumption which means the attributes (concentrations) are independent of each other.

The trend of this data is investigated further in following sections via density and probability contour plots.

In Figure 3, the trend becomes even more prominent. We can see a downward trend in the diseased patients which is due to modelling it without the Naive assumption. This means the attributes can now have dependence (i.e. relationship), which is shown here as the downward trend. In this case, if the data were to extend further with high concentration in both directions, we would likely see an ellipse or sphere with a tilt.

Interestingly, Figure 3 shows that healthy patients appear above the diseased when chemical 2 concentration reaches approximately 17. This is an odd trend and in reality would be a cause for additional testing for clarification. It can be explained by the small initial cluster of diseased patients in 1. This cluster will have low covariance (or variance), meaning likelihood decreases rapidly from the centre of this cluster. This trend will be explained further in following sections via density and probability contour plots.

4.3 Conclusion

The Naive assumption has a significant influence on this data set indicating there is a relationship between the attributes. However, this does not mean it is the best method for classification. In reality,

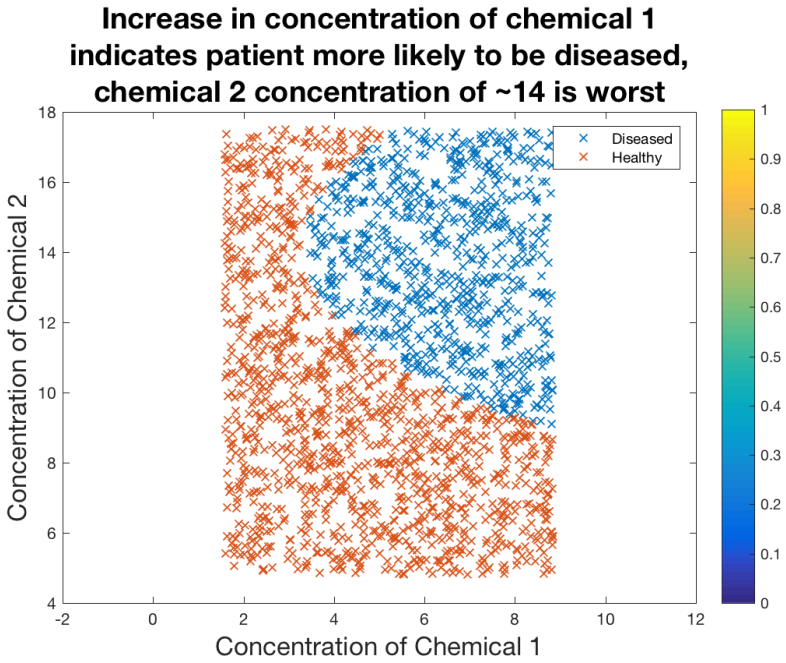


Figure 2: Maximum Likelihood classification of new data

it is unlikely that as the concentration of these chemicals the patients become diseased, then healthy again, unless there is some optimum level of chemical concentration. But one would assume that if you had a concentration greater than **x** amount, you would be classified as diseased.

Furthermore, as we are classifying if patients are diseased or healthy, it is safer to classify as diseased. This is because we can test again to confirm the results but if classify as healthy we would likely not test again.

In summary, maximum likelihood with the Naive assumption is the preferred model as it 'safer' in disease classification. In other words, it avoids the awkward trend that is a result of the low covariance (or variance) of the diseased data.

4.4 Further Comments

Further investigation was undertaken to determine the trend of the data. We plot the density contours of the maximum likelihood for each class in 4. Figures 4a and 4c show the difference between the Naive assumption. The spherical density plot in Figure 4a indicates independence of the attributes and the downward tilted ellipse in Figure 4c. This tilted ellipse partly explains why we see healthy patients above those that are diseased.

We also discover the rate at which the maximum likelihood decreases for diseased patients compared to healthy. Looking at Figures 4a and 4c, we see this by the close proximity of contours for the diseased compared to more spread of the healthy. This means, for an increase in chemical concentration, there is a point where the likelihood of being diseased becomes less than being healthy. This can be explained by imaging two lines with different slopes and y intercepts, where the y value is the likelihood of the class and slope represents the proximity of contour lines. If we imagine line 1 has a larger y intercept (larger initial likelihood) but greater negative slope (closer contour lines) than line 2, then there will be a point where y value of line 1 becomes less than line 2 due to the greater negative slope.

The probability contours in Figure **??** reinforce what we deduced in the density contour plots. This is

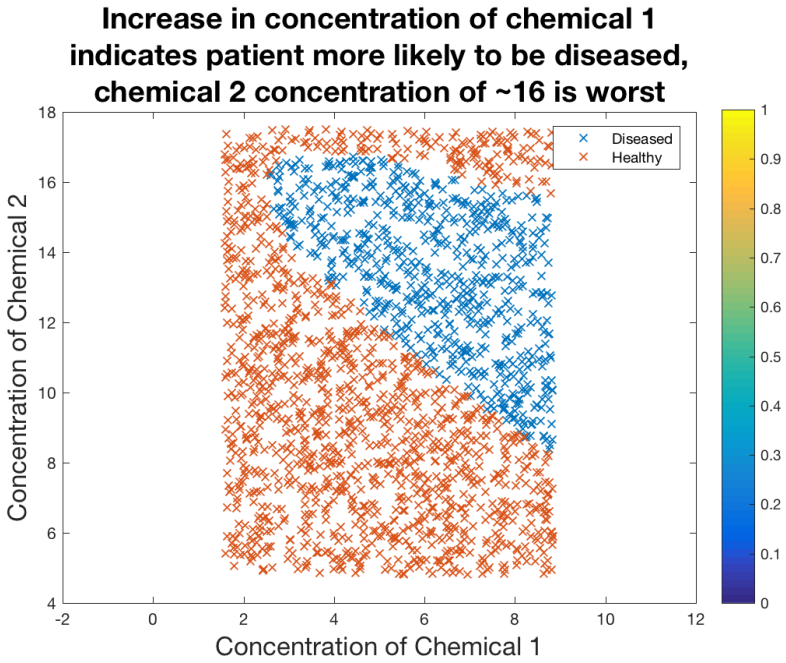
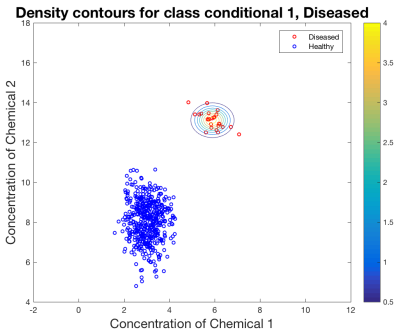
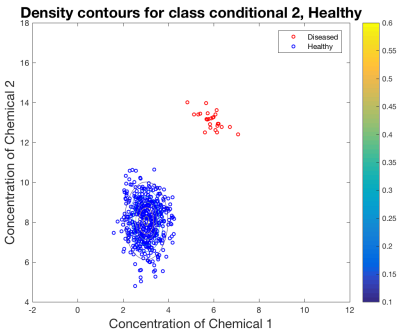


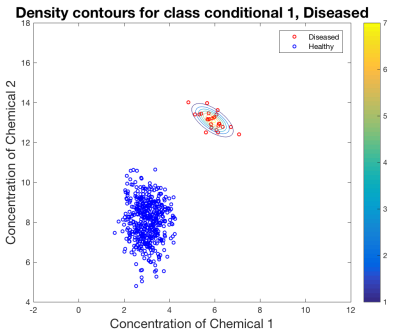
Figure 3: Maximum Likelihood without Naive assumption classification of new data



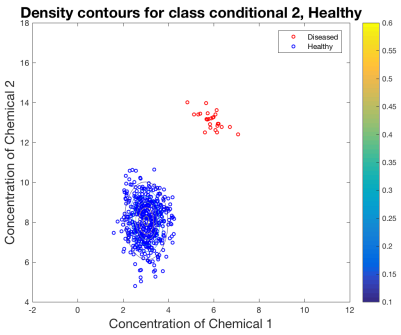
(a) ML class conditional 1 density contours



(b) ML class conditional 2 density contours



(c) MLWON class conditional 1 density contours



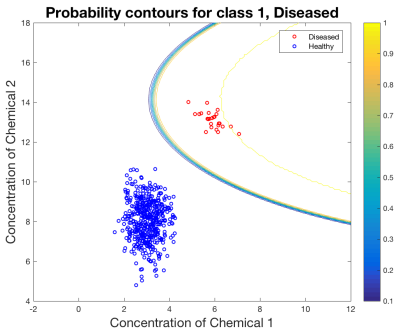
(d) MLWON class conditional 2 density contours

Figure 4: Comparison of class conditional density contours for each class and with Naive and without Naive assumption

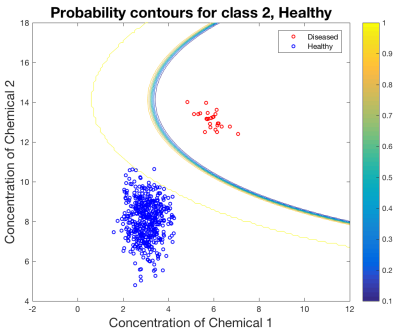
because they are very much dependent on each other. To generate the probability contours we normalise

the maximum likelihood by dividing it with the sum of the ML for each class. Therefore, the probability contour plots are the normalised versions of the maximum likelihood density plots. These are useful as the contours represent actual probabilities rather than some scaled unit.

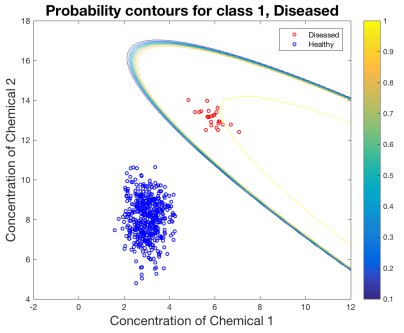
Interestingly, the probability contours are elliptical (or spherical) around the diseased class for both class 1 and 2. This is again due to the low covariance (or variance) of the diseased data points, which allows class 1 to dominate in terms of probability. This is exacerbated by the imbalanced data set. The healthy patients outnumber the diseased by 19 to 1 (475 vs 25). This means we get more varied values and thus a larger covariance (or variance). This imbalance also has other influences on the prior which is discussed in the following section.



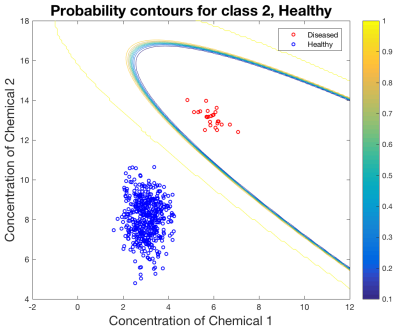
(a) ML probability contours for class 1



(b) ML probability contours for class 2



(c) MLWON probability contours for class 1



(d) MLWON probability contours for class 2

Figure 5: Comparison of probability contours for each class and with Naive and without Naive assumption

5 Maximum A Posteriori (MAP)

5.1 Introduction

In Task 1, we are asked to "train the Bayesian classifier, with maximum likelihood estimate (ML), on the training data with and without the Naive assumption." The data set is the one shown in 1. Once trained we classify 2000 new data points (patients).

To train ML with the Naive assumption first compute the mean and variance of the data for each class and attribute. As it is Naive, we assume the attributes (chemical 1 and 2) are independent. Armed with the mean, variance, and the new data set we use the Gaussian expression to compute the likelihood of each data point as either diseased or healthy. In this case, we are looking at the maximum likelihood and thus, only use class-conditional likelihood to classify.

To train ML without the Naive assumption is similar to before. We compute the mean as normal but this method requires the covariance not variance of the classes and attributes. This assumes a dependence between the attributes which is useful to look at if there are not many attributes. We then compute the maximum likelihood as normal but accounting for the 3 dimensional matrix due to the covariance.

5.2 Results

From Figure **??** and Table 1 it is shown that a polynomial function with order 4 is the best fit. However, in order to determine the "best" order, it must be given a concrete meaning for this task. "Best" is considered to be the minimum mean squared loss, where loss is the difference between predicted and observed labels. Ideally, both cross-validation (CV) and training loss are considered but that is not always the case. So, three cases are defined as the following.

1. Only CV loss is considered
2. Only Train loss is considered
3. Both CV and Train loss are considered

Table 1: Classification Differences between Methods

Order	CV Loss	Train Loss	Average Squared Loss
0	10.83	8.07	178.61
1	2.71	1.54	9.03
2	1.57	1.01	3.33
3	3.99	0.98	12.35
4	1.64	0.93	3.30

Consider item 1, Figure **??** and Table 1 show that a polynomial function with order 2 is the best fit. It is visualised clearly on Figure **??** as the minimum point on the line. Looking at Table 1, a minimum value of 1.57 also corresponds with order 2.

Consider item 2, Figure **??** and Table 1 show that a polynomial function with order 4 is the best fit. Figure **??** shows a downward trend to the right, indicating that an order of 4 is indeed the best fit. Looking at Table 1, the minimum value, 3.30, also corresponds with an order of 4.

Consider item 3, Figure **??** and Table 1 show that a polynomial function with order 4 is the best fit. It is difficult to see this via the visualisation of Figure **??**. However, looking at Table 1, the average squared loss (of CV and Train loss) is lowest when order equals 4.

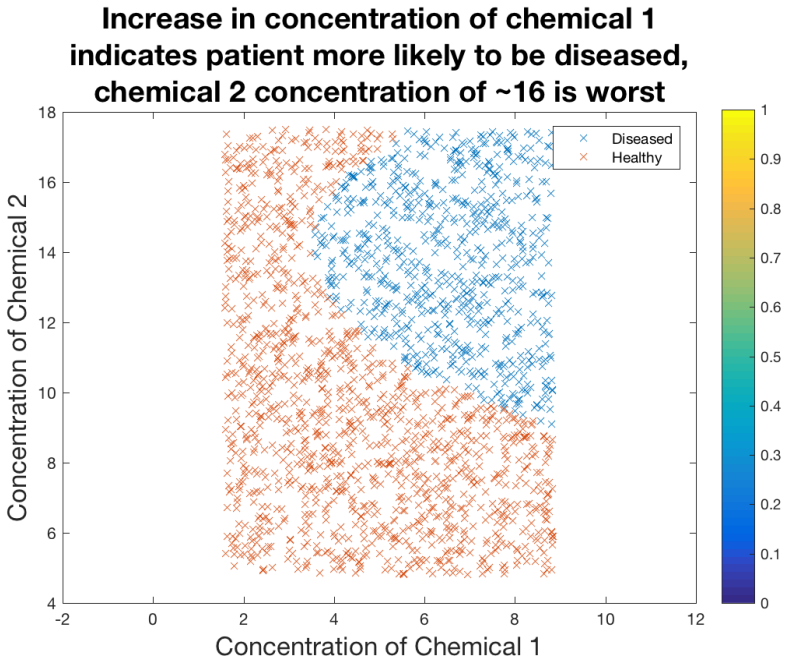


Figure 6: Maximum A Posteriori classification of new data

Figure 11c shows how well each of the models fit the data. Interestingly, Figure **??**, shows an order of 4 provides an accurate model and more realistic future predictions than an order of 2. In comparison, an order of 2 shows that future predictions would increase in time at an increasing rate. This is highly unlikely given the downward trend of the data.

The remaining models in Figure **??** clearly do not model the data well, where orders 1 and 3 show that a time of 0 will be achieved soon. This is not humanly possibly so the models can be discarded.

5.3 Conclusion

The problem for Task 1 is to find the "best model based on average cross-validation loss." This only considers point 1 from before, therefore, a polynomial function with order 2 best fits the model based on average cross-validation loss.

5.4 Further Comments

Further investigation found if the data was not standardised, a systematic error is produced in the model for orders greater than or equal 4. Figure **??** illustrates an order of 3 produces no error, but an order of 4 does. Reviewing literature showed this systematic error is common when dealing with high order polynomial function[**WhenIsIt**].Therefore, it is good practice to standardise data when the regression model contains polynomial terms.

Another point worth mentioning is the effect of permuting (randomising) the attributes. For this small dataset the permutations caused some instability in the model and different results were obtained as can be seen in Figure **??**. The results of the permuted data agree with the results from before.

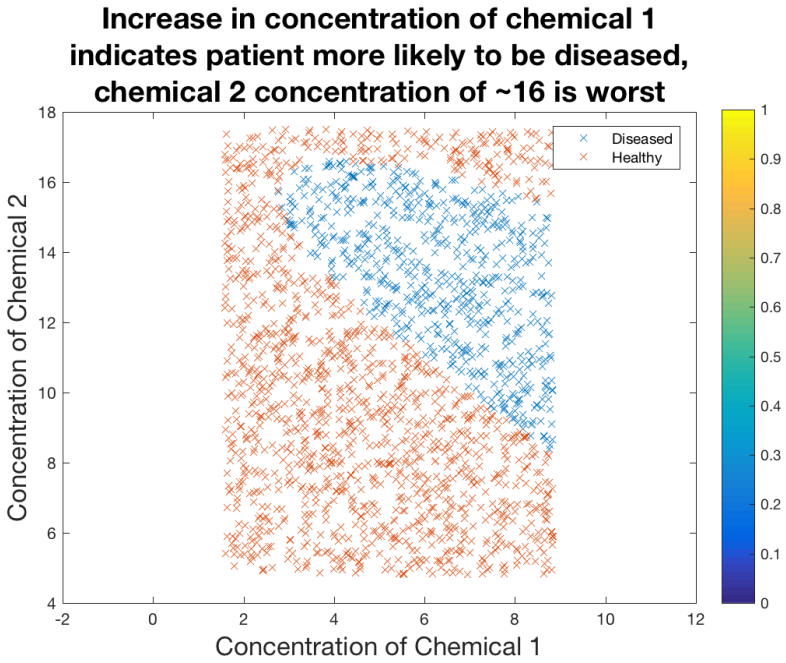


Figure 7: Maximum A Posteriori without Naive assumption classification of new data

6 Maximum Likelihood vs Maximum A Posteriori (ML VS MAP)

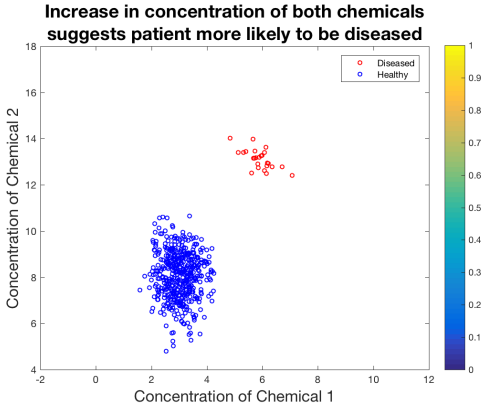
6.1 Difference of ML and MAP

In Task 1, we are asked to analyse the Olympics men's 400m data. We must find the polynomial function of order **n** which best fits this data, where **n** is 1 to 4, and use 10-fold cross-validation to choose the "best" value of **n**. Refer to Appendix A for the Matlab code used for analysis.

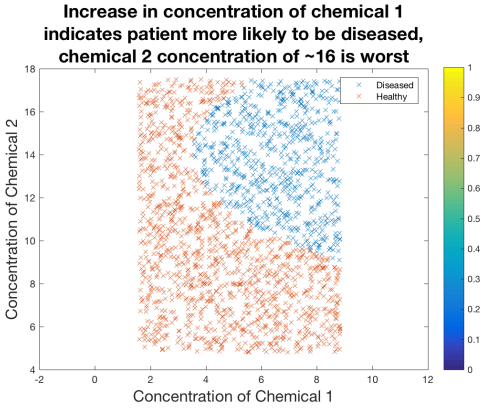
Table 2: Classification Differences between Methods

Method	Classified as Diseased	Classified as Healthy
ML	764	1236
MLWON	681	1319
MAP	826	1174
MAPWON	749	1254

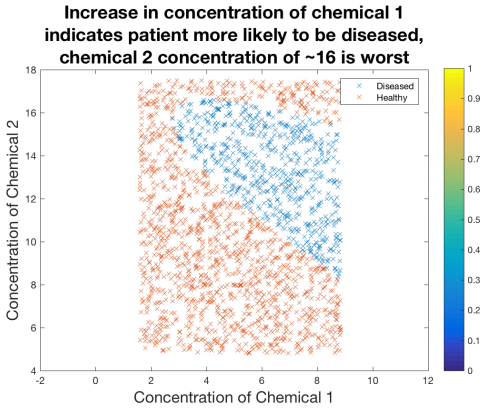
To compute the average cross validation loss, data (attributes and labels) are separated into 10 partitions (i.e. 10-fold), where 1 partition is reserved for testing the model. The 9 remaining partitions are used to learn the model with parameters, w_n. This models can use the attributes of the test partition to predict the labels, in this case the time ran for the men's 400m. The predicted values can then be compared to the actual values (labels) from the test partition. The cross validation (CV) loss is calculated by taking the mean squared difference (*msd*) of these two values. This is then repeated 9 more times by rotating the test partition through. The average of these 10 *msd* values is calculated and then plotted against the order to find the minimum value.



(a) Training data



(b) Maximum A Posteriori classification of new data



(c) Maximum A Posteriori without Naive assumption classification of new data

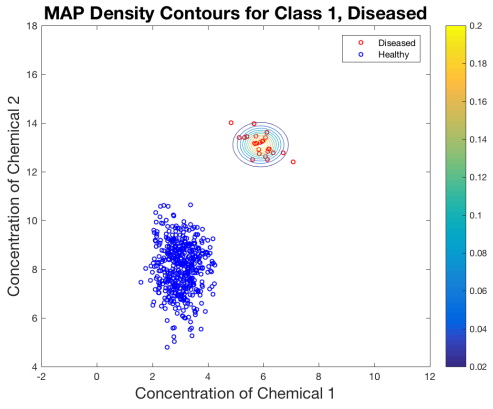
Figure 8: Comparison of training data and classification of new data by MAP and MAPWON

6.2 Affects of Difference on this Data

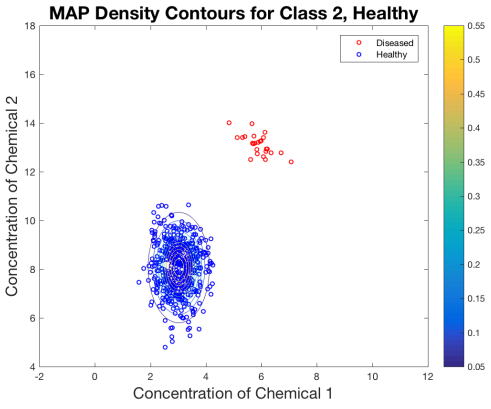
In Task 1, we are asked to analyse the Olympics men's 400m data. We must find the polynomial function of order **n** which best fits this data, where **n** is 1 to 4, and use 10-fold cross-validation to choose the "best" value of **n**. Refer to Appendix A for the Matlab code used for analysis.

To compute the average cross validation loss, data (attributes and labels) are separated into 10 partitions (i.e. 10-fold), where 1 partition is reserved for testing the model. The 9 remaining partitions are used to learn the model with parameters, w_n. This models can use the attributes of the test partition to predict the labels, in this case the time ran for the men's 400m. The predicted values can then be compared to the actual values (labels) from the test partition. The cross validation (CV) loss is calculated by taking the mean squared difference (*msd*) of these two values. This is then repeated 9 more times by rotating the test partition through. The average of these 10 *msd* values is calculated and then plotted against the order to find the minimum value.

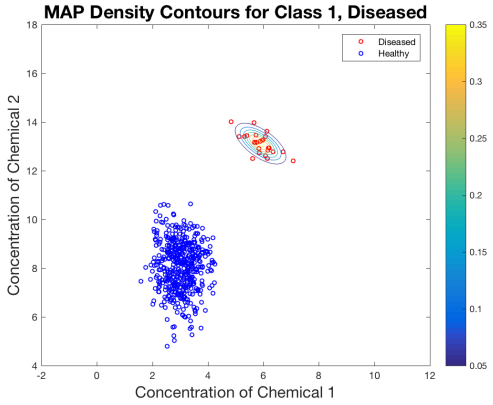
Insert Table showing difference!!!



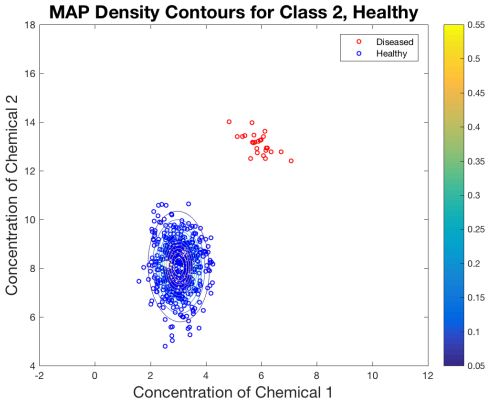
(a) MAP density contours for class 1



(b) MAP density contours for class 2

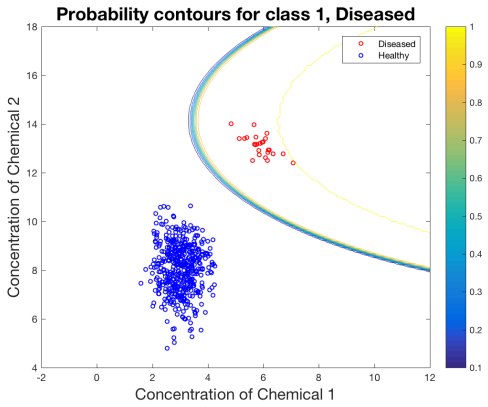


(c) MAPWON density contours for class 1

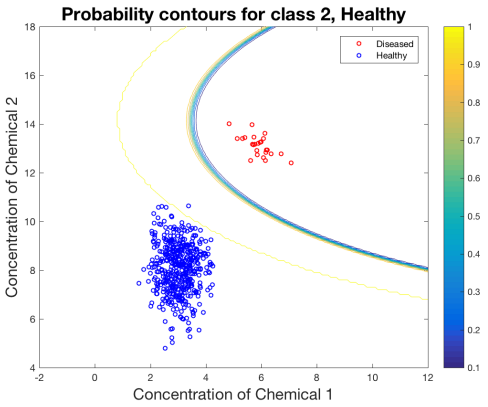


(d) MAPWON density contours for class 2

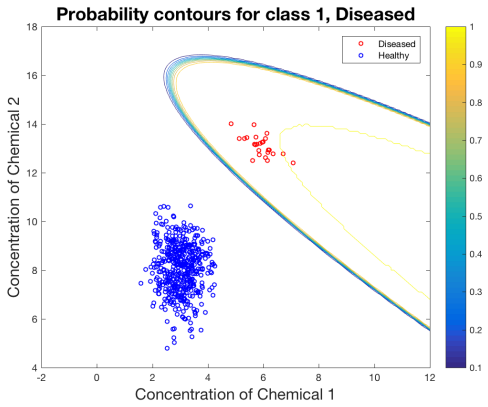
Figure 9: Comparison of density contours for each class and with Naive and without Naive assumption



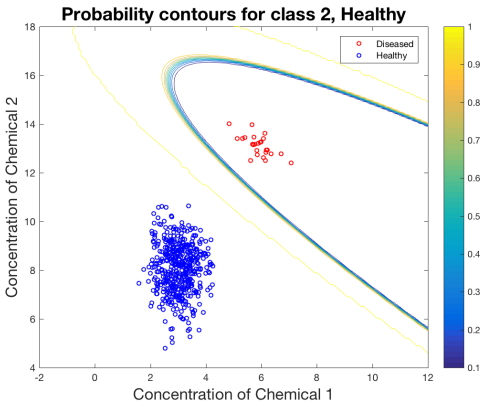
(a) MAP probability contours for class 1



(b) MAP probability contours for class 2

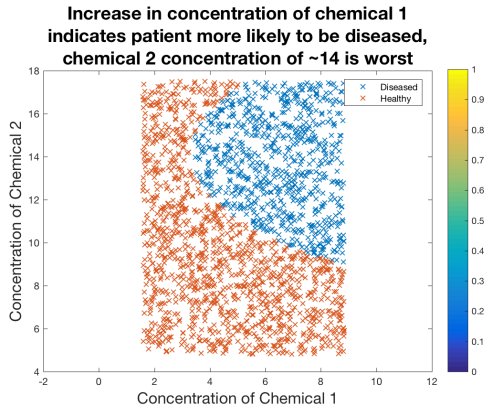


(c) MAPWON probability contours for class 1

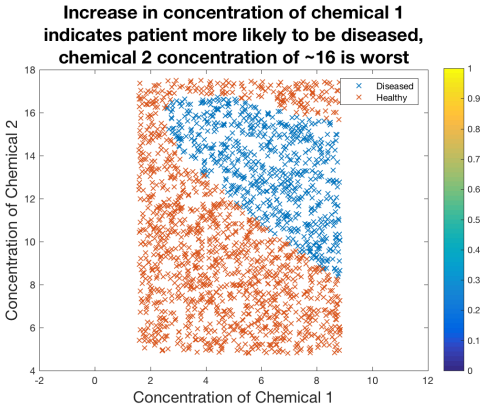


(d) MAPWON probability contours for class 2

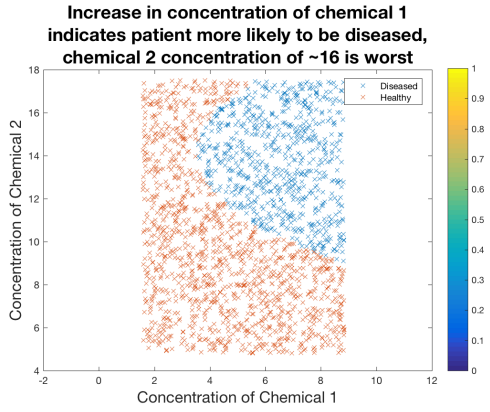
Figure 10: Comparison of probability contours for each class and with Naive and without Naive assumption



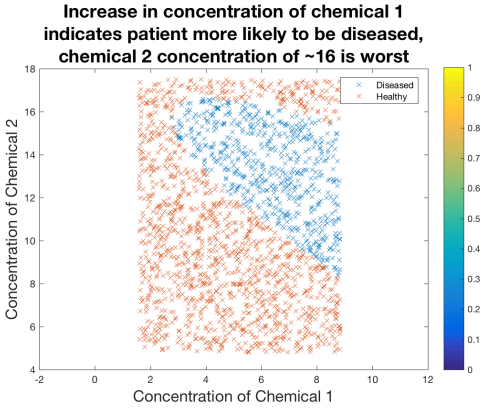
(a) Maximum Likelihood classification of new data



(b) Maximum Likelihood WON classification of new data



(c) Maximum A Posteriori classification of new data



(d) Maximum A Posteriori WON classification of new data

Figure 11: Comparison of classification of new data by ML, MLWON, MAP, and MAPWON

7 References

Appendix A: ML

```
%% This program trains a Bayesian classifier
% using the Maximum likelihood estimate.
% We train it on the helthy and diseased datasets
% provided in cbt2data.mat with and without
% the naive assumption
% Whilst all 4 methods (ML and MAP Naive, ML and MAP wo Naive)
% could be done in one program, it was found separate
% files were easier for experimenting and gaining insight of data

% Based on the following files
% Reference 1: bayesclass.m, A First Course in Machine Learning, Chapter 5.
% Reference 2: bayestraintest.m, Machine Learning (Extended)

% NOTE: do we want to compare the labels between ML, MAP, Naive, WONaive
% build separate program for this without plots?
%% ****************** ANALYSIS ******************
%  ***********************************************
%% We load the data
clear all; close all;
load cbt2data.mat
X_train = [diseased'; healthy']; % We put the training data into a single matrix
t_train = [ones(length(diseased'),1); ones(length(healthy'),1).*2]; % We give
    diseased a label of 1, healthy a label of 2
X_new = newpts'; % load new points
savePlots = 1; % 0 = don't save plots, 1 = save plots

%% Find the mean and variance
% Using the Naive (independence) assumption
cl = unique(t_train); % get total number of classes from test set

for c = 1:length(cl) % We loop over the number of classes (1,diseased and 2,healthy
    )
    pos = find(t_train==cl(c)); % We store position of class label in vector
    class_mean(c,:) = mean(X_train(pos,:)); % We compute the mean for both
    attributes and for each class
    class_var(c,:) = var(X_train(pos,:),1); % We compute the variance for both
    attributes and for each class
end

%% Maximum likelihood (ML) classification for new data
class_probs_new = [];

for c = 1:length(cl)
    sigma_naive = diag(class_var(c,:)); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-class_mean(c,1) X_new(:,2)-class_mean(c,2)]; % We
    compute difference between data point and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma_naive))); % we compute the
    constanst for the gaussian function
    class_probs_new = [class_probs_new const * exp(-1/2*diag(diff_train * inv(
```

```
    sigma_naive) * diff_train'))]; % We use the gaussian function to compute ML
end

%% Classify each example
% We classify each example by getting the column index which has the higher
% maximum likelihood (ML). Column 1 and column 2 give the ML of belonging to
% each class respectively. For example, a column 1 has a ML of 0.7 and
% column 2 is 0.3, therefore it returns the index (or column) 1.
[M,Class_new] = max(class_probs_new, [], 2);

%% Compute the predictive probabilities
% We calculate the proabilities for random data (mesh grid)
% so we can plot contours of probabilities and ML
[Xv,Yv] = meshgrid(-2:0.1:12,4:0.1:18); % We create mesh grid to compute contour
    plots
Probs = [];

for c = 1:length(cl) % loop over unique classes
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)]; % x1i - mean, x2i - mean
    tempc = diag(class_var(c,:)); %
    const = -log(2*pi) - log(det(tempc)); % We compute constant using log laws
    Probs(:,:,c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')),size(Xv));
    % We compute probability for each point
end

%% ******************* PLOTS *******************
%  *********************************************
% Plots kept seperate so we can easily
% adjust titles and labels

%% We plot the training data
train_colours = {'r','b'};
tag = {'o','o'};
figure(1);
hold off

for c = 1:length(cl)
    pos = find(t_train==cl(c)); % get position, or row, of each data point in class
    plot(X_train(pos,1),X_train(pos,2),tag{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',4,'linewidth',1,...
        'color',train_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of both chemicals';...
    'suggests patient more likely to be diseased'},'fontsize',18);
xlim([-2 12]), ylim([4 18]); % set x and y scales square
colorbar('eastoutside');
```

```
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MLtrainingData.png');
    saveas(gcf,filename);
end
```

```
%% We plot the new data
new_colours = {'r','b'};
tag_new = {'x','x'};
figure(2);
hold off

for c = 1:length(cl)
    pos = find(Class_new==cl(c)); % get position, or row, of each data point in
    class
    plot(X_new(pos,1),X_new(pos,2),tag_new{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',5,'linewidth',.1,...
        'markerfacecolor',new_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of chemical 1';...
    'indicates patient more likely to be diseased,';...
    'chemical 2 concentration of ~14 is worst'},'fontsize',18);
xlim([-2 12]), ylim([4 18]);
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MLnewData.png');
    saveas(gcf,filename);
end
```

```
%% Plot the density contours for class-conditional distributions
classLabel = {'Diseased';'Healthy'};

for i = 1:2
    figure(i+2);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
```

```
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Density contours for class conditional %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize',18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MLclassCondContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

```
%% Plot the contours of the classification probabilities
Probs = Probs./repmat(sum(Probs,3),[1,1,2]); % We normalise to get probability

for i = 1:2
    figure(i+4);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Probability contours for class %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize', 18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MLprobContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

Appendix B: ML WON

```
%% This program trains a Bayesian classifier
% using the Maximum likelihood estimate WITHOUT naive assumption.
% We train it on the helthy and diseased datasets
% provided in cbt2data.mat

% Whilst all 4 methods (ML and MAP Naive, ML and MAP wo Naive)
% could be done in one program, it was found separate
% files were easier for experimenting and gaining insight of data

% Based on the following files
% Reference 1: bayesclass.m, A First Course in Machine Learning, Chapter 5.
% Reference 2: bayestraintest.m, Machine Learning (Extended)
```

```
%% ****************** ANALYSIS ******************
%  **********************************************
```

```
%% We load the data
clear all; close all;
load cbt2data.mat
X_train = [diseased'; healthy']; % We put the training data into a single matrix
t_train = [ones(length(diseased'),1); ones(length(healthy'),1).*2]; % We give
    diseased a label of 1, healthy a label of 2
X_new = newpts'; % load new points
savePlots = 0; % 0 = don't save, 1 = save
```

```
%% Find the mean and variance
% Using the Naive (independence) assumption
cl = unique(t_train); % get total number of classes from test set

for c = 1:length(cl) % We loop over the number of classes (1,diseased and 2,healthy
    )
    pos = find(t_train==cl(c)); % We store position of class label in vector
    class_mean(c,:) = mean(X_train(pos,:)); % class-wise & attribute-wise mean
    class_var(:,:,c) = cov(X_train(pos,:),1); % class-wise & attribute-wise
    variance
end
```

```
%% Maximum likelihood (ML) classification for new data
% Without the naive assumption, thus we don't take the diag of variance
% (see sigma)
class_probs_new = [];

for c = 1:length(cl)
    sigma = class_var(:,:,c); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-class_mean(c,1) X_new(:,2)-class_mean(c,2)]; % We
    compute difference between data point and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma))); % we compute the constanst
    for the gaussian function
    class_probs_new = [class_probs_new const * exp(-1/2*diag(diff_train * inv(sigma
    ) * diff_train'))]; % We use the gaussian function to compute MLend
```

```
end

%% Classify each example
% We classify each example by getting the column index which has the higher
% maximum likelihood (ML). Column 1 and column 2 give the ML of belonging to
% each class respectively. For example, a column 1 has a ML of 0.7 and
% column 2 is 0.3, therefore it returns the index (or column) 1.
[M,Class_new] = max(class_probs_new, [], 2);

%% Compute the predictive probabilities
% We calculate the proabilities for random data (mesh grid)
% so we can plot contours of probabilities and ML
[Xv,Yv] = meshgrid(-2:0.1:12,4:0.1:18); % We create mesh grid to compute contour
    plots
Probs = [];

for c = 1:length(cl) % loop over unique classes
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)]; % xnew - mean, tnew -
    mean
    tempc = (class_var(:,:,c)); % get all rows and columns for each class
    const = -log(2*pi) - log(det(tempc)); % % We compute constant using log laws,
    is it actually easier with log?, why not -1/2 out front?
    Probs(:,:,c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')),size(Xv));
    % reshape exp(of function) into size of Xv
end

%% ******************* PLOTS *******************
%  *********************************************
% Plots kept seperate so we can easily
% adjust titles and labels

%% We plot the training data
train_colours = {'r','b'};
tag = {'o','o'};
figure(1);
hold off

for c = 1:length(cl)
    pos = find(t_train==cl(c)); % get position, or row, of each data point in class
    plot(X_train(pos,1),X_train(pos,2),tag{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',4,'linewidth',1,...
        'color',train_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of both chemicals';...
    'suggests patient more likely to be diseased'},'fontsize',18);
xlim([-2 12]), ylim([4 18]); % set x and y scales square
```

```
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MLWONtrainingData.png');
    saveas(gcf,filename);
end
```

```
%% We plot the new data
new_colours = {'r','b'};
tag_new = {'x','x'};
figure(2);
hold off

for c = 1:length(cl)
    pos = find(Class_new==cl(c)); % get position, or row, of each data point in
    class
    plot(X_new(pos,1),X_new(pos,2),tag_new{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',5,'linewidth',.1,...
        'markerfacecolor',new_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of chemical 1';...
    'indicates patient more likely to be diseased,';...
    'chemical 2 concentration of ~16 is worst'},'fontsize',18);
xlim([-2 12]), ylim([4 18]);
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MLWONnewData.png');
    saveas(gcf,filename);
end
```

```
%% Plot the density contours for class-conditional distributions
classLabel = {'Diseased';'Healthy'};

for i = 1:2
    figure(i+2);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
```

```
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Density contours for class conditional %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize',18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MLWONclassCondContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

```
%% Plot the contours of the classification probabilities
Probs = Probs./repmat(sum(Probs,3),[1,1,2]); % We normalise to get probability

for i = 1:2
    figure(i+4);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Probability contours for class %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize', 18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MLWONprobContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

Appendix C: MAP

```
%% This program trains a Bayesian classifier
% using the Maximum A Posteriori estimate WITH naive assumption.
% We train it on the helthy and diseased datasets
% provided in cbt2data.mat

% Whilst all 4 methods (ML and MAP Naive, ML and MAP wo Naive)
% could be done in one program, it was found separate
% files were easier for experimenting and gaining insight of data

% Based on the following files
% Reference 1: bayesclass.m, A First Course in Machine Learning, Chapter 5.
% Reference 2: bayestraintest.m, Machine Learning (Extended)

%% ****************** ANALYSIS ******************
%  **********************************************
%% We load the data
clear all; close all;
load cbt2data.mat
X_train = [diseased'; healthy']; % We put the training data into a single matrix
t_train = [ones(length(diseased'),1); ones(length(healthy'),1).*2]; % We give
    diseased a label of 1, healthy a label of 2
X_new = newpts'; % load new points
savePlots = 0; % 0 = don't save, 1 = save

%% We compute the prior
class1Total = sum(t_train==1); % We count the number of diseased patients
class2Total = sum(t_train==2); % We count the number of healthy patients
probClass1 = class1Total/(class1Total + class2Total); % We compute the dieased
    prior based on frequency
probClass2 = class2Total/(class1Total + class2Total); % We compute the healthy
    prior based on frequency
probPrior = [probClass1; probClass2]; % We store priors in vector, make matrix and
    put in other prior values?

% Uncomment to experiment with different priors:
% probPrior = [.99; 0.01]; % We give diseased class a strong prior
% probPrior = [0.5; 0.5]; % We compute prior based on 1/C

%% Find the mean and variance
% Using the Naive (independence) assumption
cl = unique(t_train); % get total number of classes from test set

for c = 1:length(cl) % We loop over the number of classes (1,diseased and 2,healthy
    )
    pos = find(t_train==cl(c)); % We store position of class label in vector
    class_mean(c,:) = mean(X_train(pos,:)); % class-wise & attribute-wise mean
    class_var(c,:) = var(X_train(pos,:),1); % class-wise & attribute-wise variance
end
```

```
%% Maximum A Posteriori (MAP) classification for new data
% with the naive assumption, thus we take the diag of variance
% (see sigma_naive)
class_probs_new = [];

for c = 1:length(cl)
    sigma_naive = diag(class_var(c,:)); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-class_mean(c,1) X_new(:,2)-class_mean(c,2)]; % We
    compute difference between data points and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma_naive))); % we compute the
    constanst for the gaussian function
    class_probs_new = [class_probs_new const * exp(-1/2*diag(diff_train * inv(
    sigma_naive) * diff_train'))* probPrior(c)]; % We use the gaussian function to
    compute MAP
end
```

```
%% Classify each example
% We classify each example by getting the column index which has the higher
% maximum likelihood (ML). Column 1 and column 2 give the ML of belonging to
% each class respectively. For example, a column 1 has a ML of 0.7 and
% column 2 is 0.3, therefore it returns the index (or column) 1.
[M,Class_new] = max(class_probs_new, [], 2);
```

```
%% Compute the predictive probabilities
% We calculate the proabilities for random data (mesh grid)
% so we can plot contours of probabilities and ML
[Xv,Yv] = meshgrid(-2:0.1:12,4:0.1:18); % We create mesh grid to compute contour
    plots
Probs = [];

for c = 1:length(cl) % loop over unique classes
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)]; % xnew - mean, tnew -
    mean
    tempc = diag(class_var(c,:)); % only diagonal, assuming naive? , (co)variance
    for class
    const = -log(2*pi) - log(det(tempc)); % is it actually easier with log?, why
    not -1/2 out front?
    Probs(:,:,c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')),size(Xv)) *
     probPrior(c); % reshape exp(of function) into size of Xv
end
```

```
%% ****************** PLOTS ******************
%  *******************************************
% Plots kept seperate so we can easily
% adjust titles and labels
```

```
%% We plot the training data
train_colours = {'r','b'};
tag = {'o','o'};
figure(1);
hold off
```

```
for c = 1:length(cl)
    pos = find(t_train==cl(c)); % get position, or row, of each data point in class
    plot(X_train(pos,1),X_train(pos,2),tag{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',4,'linewidth',1,...
        'color',train_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of both chemicals';...
    'suggests patient more likely to be diseased'},'fontsize',18);
xlim([-2 12]), ylim([4 18]); % set x and y scales square
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MAPtrainingData.png');
    saveas(gcf,filename);
end
```

```
%% We plot the new data
new_colours = {'r','b'};
tag_new = {'x','x'};
figure(2);
hold off

for c = 1:length(cl)
    pos = find(Class_new==cl(c)); % get position, or row, of each data point in
    class
    plot(X_new(pos,1),X_new(pos,2),tag_new{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',5,'linewidth',.1,...
        'markerfacecolor',new_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of chemical 1';...
    'indicates patient more likely to be diseased,';...
    'chemical 2 concentration of ~16 is worst'},'fontsize',18);
xlim([-2 12]), ylim([4 18]);
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
```

```
    filename = strcat('MAPnewData.png');
    saveas(gcf,filename);
end

%% Plot the density contours distributions
classLabel = {'Diseased';'Healthy'};

for i = 1:2
    figure(i+2);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('MAP Density Contours for Class %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize',18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MAPclassCondContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end

%% Plot the contours of the classification probabilities
Probs = Probs./repmat(sum(Probs,3),[1,1,2]); % We normalise to get probability

for i = 1:2
    figure(i+4);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Probability contours for class %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
```

```
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize', 18);
    xlim([−2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MAPprobContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

Appendix D: MAP WON

```
%% This program trains a Bayesian classifier
% using the Maximum A Posteriori estimate WITHOUT naive assumption.
% We train it on the helthy and diseased datasets
% provided in cbt2data.mat

% Whilst all 4 methods (ML and MAP Naive, ML and MAP wo Naive)
% could be done in one program, it was found separate
% files were easier for experimenting and gaining insight of data

% Based on the following files
% Reference 1: bayesclass.m, A First Course in Machine Learning, Chapter 5.
% Reference 2: bayestraintest.m, Machine Learning (Extended)

%% ****************** ANALYSIS ******************
%  **********************************************
%% We load the data
clear all; close all;
load cbt2data.mat
X_train = [diseased'; healthy']; % We put the training data into a single matrix
t_train = [ones(length(diseased'),1); ones(length(healthy'),1).*2]; % We give
    diseased a label of 1, healthy a label of 2
X_new = newpts'; % load new points
savePlots = 0; % 0 = don't save, 1 = save

%% We compute the prior
class1Total = sum(t_train==1); % We count the number of diseased patients
class2Total = sum(t_train==2); % We count the number of healthy patients
probClass1 = class1Total/(class1Total + class2Total); % We compute the dieased
    prior based on frequency
probClass2 = class2Total/(class1Total + class2Total); % We compute the healthy
    prior based on frequency
probPrior = [probClass1; probClass2]; % We store priors in vector, make matrix and
    put in other prior values?

% Uncomment to experiment with different priors:
% probPrior = [.99; 0.01]; % We give diseased class a strong prior
% probPrior = [0.5; 0.5]; % We compute prior based on 1/C
% probPrior = [0;1]; % We give diseased class a strong prior

%% Find the mean and variance
% Using the Naive (independence) assumption
cl = unique(t_train); % get total number of classes from test set

for c = 1:length(cl) % We loop over the number of classes (1,diseased and 2,healthy
    )
    pos = find(t_train==cl(c)); % We store position of class label in vector
    class_mean(c,:) = mean(X_train(pos,:)); % class-wise & attribute-wise mean
    class_var(:,:,c) = cov(X_train(pos,:),1); % class-wise & attribute-wise
    variance
```

```
end

%% Maximum A Posteriori (MAP) classification for new data
% Without the naive assumption, thus we don't take the diag of variance
% (see sigma)
class_probs_new = [];

for c = 1:length(cl)
    sigma = class_var(:,:,c); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-class_mean(c,1) X_new(:,2)-class_mean(c,2)]; % We
    compute difference between data point and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma))); % we compute the constanst
    for the gaussian function
    class_probs_new = [class_probs_new const * exp(-1/2*diag(diff_train * inv(sigma
    ) * diff_train'))* probPrior(c)]; % We use the gaussian function to compute
    MLend
end

%% Classify each example
% We classify each example by getting the column index which has the higher
% maximum likelihood (ML). Column 1 and column 2 give the ML of belonging to
% each class respectively. For example, a column 1 has a ML of 0.7 and
% column 2 is 0.3, therefore it returns the index (or column) 1.
[M,Class_new] = max(class_probs_new, [], 2);

%% Compute the predictive probabilities
% We calculate the proabilities for random data (mesh grid)
% so we can plot contours of probabilities and ML
[Xv,Yv] = meshgrid(-2:0.1:12,4:0.1:18); % We create mesh grid to compute contour
    plots
Probs = [];

for c = 1:length(cl) % loop over unique classes
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)]; % xnew - mean, tnew -
    mean
    tempc = (class_var(:,:,c)); % get all rows and columns for each class
    const = -log(2*pi) - log(det(tempc)); % % We compute constant using log laws,
    is it actually easier with log?, why not -1/2 out front?
    Probs(:,:,c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')),size(Xv)) *
     probPrior(c); % reshape exp(of function) into size of Xv
end

%% ******************* PLOTS *******************
%  *********************************************
% Plots kept seperate so we can easily
% adjust titles and labels

%% We plot the training data
train_colours = {'r','b'};
tag = {'o','o'};
figure(1);
```

```
hold off

for c = 1:length(cl)
    pos = find(t_train==cl(c)); % get position, or row, of each data point in class
    plot(X_train(pos,1),X_train(pos,2),tag{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',4,'linewidth',1,...
        'color',train_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of both chemicals';...
    'suggests patient more likely to be diseased'},'fontsize',18);
xlim([-2 12]), ylim([4 18]); % set x and y scales square
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);

if (savePlots == 1)
    filename = strcat('MAPWONtrainingData.png');
    saveas(gcf,filename);
end
```

```
%% We plot the new data
new_colours = {'r','b'};
tag_new = {'x','x'};
figure(2);
hold off

for c = 1:length(cl)
    pos = find(Class_new==cl(c)); % get position, or row, of each data point in
    class
    plot(X_new(pos,1),X_new(pos,2),tag_new{c},... % plot points of column 1 on x
    and column 2 on y
        'markersize',5,'linewidth',.1,...
        'markerfacecolor',new_colours{c});
    hold on
end

xlabel('Concentration of Chemical 1','fontsize',16);
ylabel('Concentration of Chemical 2','fontsize',16);
title({'Increase in concentration of chemical 1';...
    'indicates patient more likely to be diseased,';...
    'chemical 2 concentration of ~16 is worst'},'fontsize',18);
xlim([-2 12]), ylim([4 18]);
colorbar('eastoutside');
legend('Diseased', 'Healthy');
set(gca,'Color',[0.7 0.7 0.7]);
```

```
if (savePlots == 1)
    filename = strcat('MAPWONnewData.png');
    saveas(gcf,filename);
end
```

```
%% Plot the density contours for class-conditional distributions
classLabel = {'Diseased';'Healthy'};

for i = 1:2
    figure(i+2);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('MAP Density Contours for Class %g, %s',i, classLabel{i});
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize',18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MAPWONclassCondContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

```
%% Plot the contours of the classification probabilities
Probs = Probs./repmat(sum(Probs,3),[1,1,2]); % We normalise to get probability

for i = 1:2
    figure(i+4);hold off
    hold off
    for c = 1:length(cl) % plot the data points for each class
        pos = find(t_train==cl(c));
        plot(X_train(pos,1),X_train(pos,2),tag{c},...
            'markersize',4,'linewidth',1,...
            'color',train_colours{c});
        hold on
    end

    contour(Xv,Yv,Probs(:,:,i)); % plot contour line
    ti = sprintf('Probability contours for class %g, %s',i, classLabel{i});
```

```
    xlabel('Concentration of Chemical 1','fontsize',16);
    ylabel('Concentration of Chemical 2','fontsize',16);
    title(ti, 'fontsize', 18);
    xlim([-2 12]), ylim([4 18]);
    colorbar('eastoutside');
    legend('Diseased', 'Healthy');
    set(gca,'Color',[0.7 0.7 0.7]);

    if (savePlots == 1)
        filename = sprintf('MAPWONprobContours%s.png', classLabel{i});
        saveas(gcf,filename);
    end
end
```

Appendix E: Method Comparison

```
%% This program trains a Bayesian classifier
% using the different methods and compares the difference in classification

% We train it on the helthy and diseased datasets
% provided in cbt2data.mat with and without

% Whilst all 4 methods (ML and MAP Naive, ML and MAP wo Naive)

% Based on the following files
% Reference 1: bayesclass.m, A First Course in Machine Learning, Chapter 5.
% Reference 2: bayestraintest.m, Machine Learning (Extended)

% NOTE: do we want to compare the labels between ML, MAP, Naive, WONaive
% build separate program for this without plots?
%% ******************* ANALYSIS *******************
%  ************************************************
%% We load the data
clear all; close all;
load cbt2data.mat
X_train = [diseased'; healthy']; % We put the training data into a single matrix
t_train = [ones(length(diseased'),1); ones(length(healthy'),1).*2]; % We give
    diseased a label of 1, healthy a label of 2
X_new = newpts'; % load new points
savePlots = 0; % 0 = don't save plots, 1 = save plots

%% We compute the prior
class1Total = sum(t_train==1); % We count the number of diseased patients
class2Total = sum(t_train==2); % We count the number of healthy patients
probClass1 = class1Total/(class1Total + class2Total); % We compute the dieased
    prior based on frequency
probClass2 = class2Total/(class1Total + class2Total); % We compute the healthy
    prior based on frequency
probPrior = [probClass1; probClass2]; % We store priors in vector, make matrix and
    put in other prior values?

% Uncomment to experiment with different priors:
probPrior = [.99; 0.01]; % We give diseased class a strong prior
% probPrior = [0.5; 0.5]; % We compute prior based on 1/C
% probPrior = [1,0];

%% Find the mean and variance
% Using the Naive (independence) assumption
cl = unique(t_train); % get total number of classes from test set

for c = 1:length(cl) % We loop over the number of classes (1,diseased and 2,healthy
    )
    pos = find(t_train==cl(c)); % We store position of class label in vector
    Class_mean(c,:) = mean(X_train(pos,:)); % We compute the mean for both
    attributes and for each class
```

```
    Class_var(c,:) = var(X_train(pos,:),1); % We compute the variance for both
    attributes and for each class

    pos = find(t_train==cl(c)); % We store position of class label in vector
    WONclass_mean(c,:) = mean(X_train(pos,:)); % class-wise & attribute-wise mean
    WONclass_var(:,:,c) = cov(X_train(pos,:),1); % class-wise & attribute-wise
    variance
end
```

```
%% Maximum likelihood (ML) NAIVE classification for new data
MLclass_probs_new = [];
MAPclass_probs_new = [];

for c = 1:length(cl)
    sigma_naive = diag(Class_var(c,:)); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-Class_mean(c,1) X_new(:,2)-Class_mean(c,2)]; % We
    compute difference between data point and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma_naive))); % we compute the
    constanst for the gaussian function
    MLclass_probs_new = [MLclass_probs_new const*exp(-1/2*diag(diff_train * inv(
    sigma_naive) * diff_train'))]; % We use the gaussian function to compute ML
    MAPclass_probs_new = [MAPclass_probs_new const*exp(-1/2*diag(diff_train * inv(
    sigma_naive) * diff_train'))* probPrior(c)]; % We use the gaussian function to
    compute MAP
end
```

```
%% Maximum likelihood (ML) WITHOUT naive classification for new data
% Without the naive assumption, thus we don't take the diag of variance
% (see sigma)
MLWONclass_probs_new = [];
MAPWONclass_probs_new = [];

for c = 1:length(cl)
    sigma = WONclass_var(:,:,c); % we convert row to diagonal elements
    diff_train = [X_new(:,1)-WONclass_mean(c,1) X_new(:,2)-WONclass_mean(c,2)]; %
    We compute difference between data point and respective mean
    const = 1/sqrt((2*pi()^size(X_new,2)*det(sigma))); % we compute the constanst
    for the gaussian function
    MLWONclass_probs_new = [MLWONclass_probs_new const * exp(-1/2*diag(diff_train *
     inv(sigma) * diff_train'))]; % We use the gaussian function to compute MLend
    MAPWONclass_probs_new = [MAPWONclass_probs_new const * exp(-1/2*diag(diff_train
     * inv(sigma) * diff_train'))* probPrior(c)]; % We use the gaussian function to
     compute MLend
end
```

```
%% Classify each example
% We classify each example by getting the column index which has the higher
% maximum likelihood (ML). Column 1 and column 2 give the ML of belonging to
% each class respectively. For example, a column 1 has a ML of 0.7 and
% column 2 is 0.3, therefore it returns the index (or column) 1.
difference = [];
```

```
countDisease = [];
countHealthy = [];

[~,MLClass_new] = max(MLclass_probs_new, [], 2);
[~,MLWONClass_new] = max(MLWONclass_probs_new, [], 2);
[~,MAPClass_new] = max(MAPclass_probs_new, [], 2);
[~,MAPWONClass_new] = max(MAPWONclass_probs_new, [], 2);

countDisease = [sum(MLClass_new==1); sum(MLWONClass_new==1); sum(MAPClass_new==1);
    sum(MAPWONClass_new==1)];
countHealthy = [sum(MLClass_new==2); sum(MLWONClass_new==2); sum(MAPClass_new==2);
    sum(MAPWONClass_new==2)];
difference = [length(MLClass_new)-sum(MLClass_new==MLWONClass_new), length(
    MAPClass_new)-sum(MAPClass_new==MAPWONClass_new), length(MLClass_new)-sum(
    MLClass_new==MAPClass_new)];
```