UNIVERSITY OF
BIRMINGHAM

Machine Learning (Extended)

Computer Based Test 2

Bayesian Classification

November 11, 2016

Thomas Brereton 1708846

Table of Content

List of Figures

List of Tables

1 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.

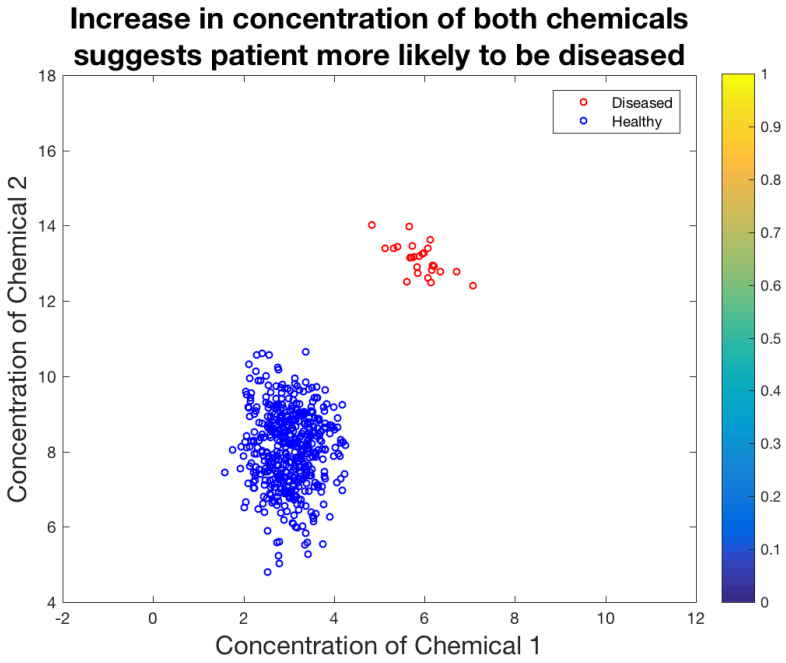


Figure 1: Training Data for the 4 methods outlined in this report.

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

- Maximum Likelihood (ML) - Appendix A
- Maximum Likelihood without Naive assumption (MLWON) - Appendix B
- Maximum a posteriori (MAP) - Appendix C
- Maximum a posteriori without Naive assumption(MAPWON) - Appendix D

The analysis was done in Matlab and the code listings for each method are found in the Appendices as described above. Appendix E contains a code listing for a program comparing the classification of each method.

Note the code is separated into 'Analysis' and 'Plots.' The 'Analysis' part is the algorithm for each respective classification method. The 'Plots' part can be disregarded unless the reader is interested in how the plots were generated.

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.

2 Task 1 - Maximum Likelihood

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

To train ML with the Naive assumption we 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 being diseased or healthy. In this case, we are looking at the Maximum Likelihood and thus, only use class-conditional likelihood to classify. Refer Figure 2 for classification without the Naive assumption.

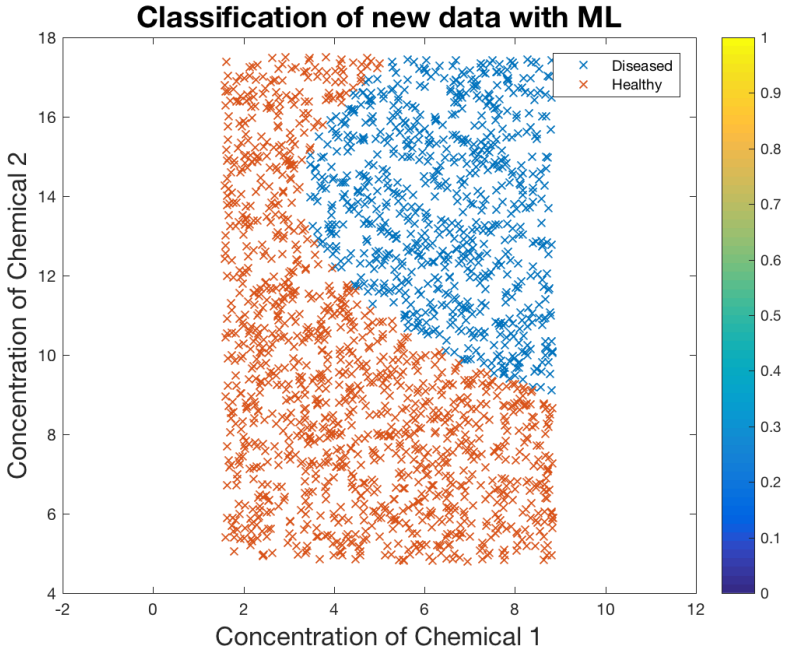


Figure 2: Maximum Likelihood classification of new data

Training 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 attributes for each class. This allows us to model any dependence between the attributes. We then compute the maximum likelihood as normal but accounting for the 3 dimensional matrix due to the covariance of each class. Refer Figure 3 for classification with the Naive assumption.

Refer Figure 4 for plots of the class-conditional density contours for the 2 ML methods. These plots show the mean of the data by the centre of the contours. The spread of the data is represented by the proximity of contours lines i.e. closer contour lines equals smaller spread. Dependence between attributes is shown by tilted contours i.e. downward tilt is a negative relationship.

The mean of the training data is shown in Table 1. The mean of the healthy patients is represented by (5.90, 13,11) and for diseased it is (3.02, 8.06).

Tables 2 and 3, show the covariance for the diseased and healthy patients respectively. Note, this is with the Naive assumption, therefore, the off diagonal elements are zero (independence).

Tables 4 and 5, show the covariance for the diseased and healthy patients respectively. Note, this is

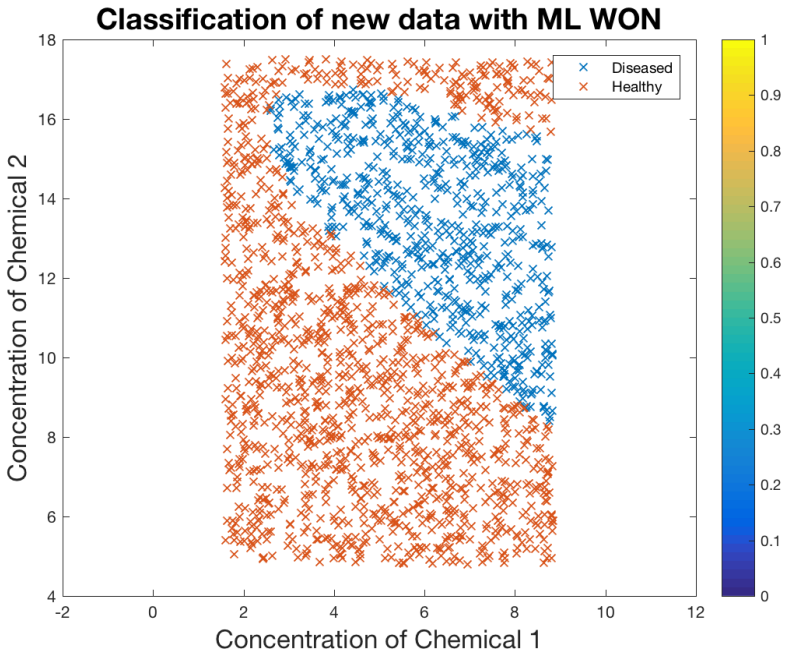
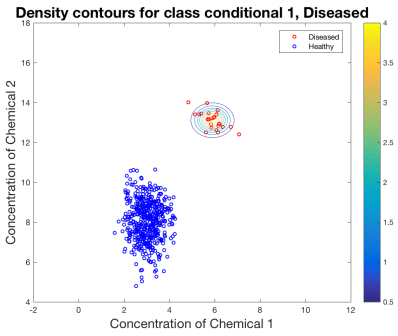
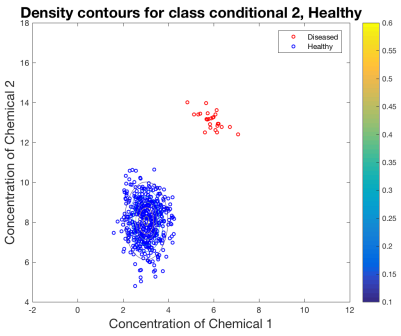


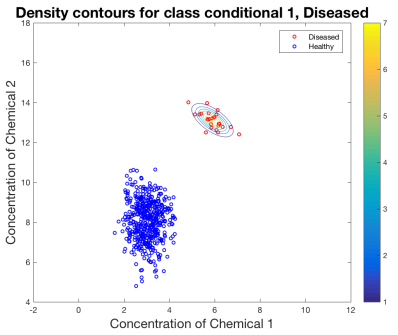
Figure 3: Maximum Likelihood without Naive assumption classification of new data



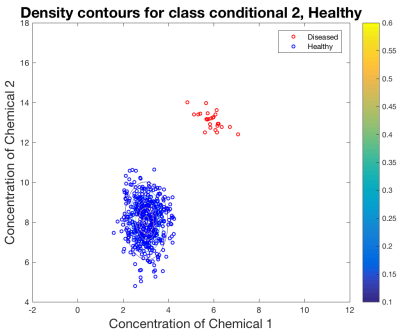
(a) ML class conditional 1 density contours



(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 ML class conditional density contours for each class and with Naive and without Naive assumption

without the Naive assumption, therefore, the off diagonal elements are not zero (dependence).

Table 1: Mean of Diseased and Healthy Class for Training Data

Class	Chemical 1	Chemical 2
Diseased	5.90	13.113
Healthy	3.02	8.06

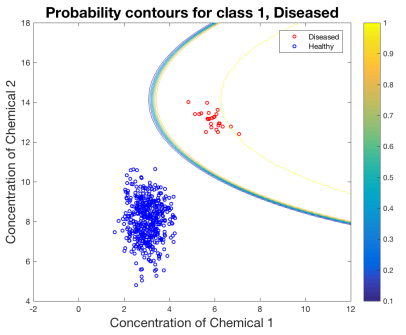
Table 2: Class 1 (Diseased) variance of Chemical Concentrations for ML (or MAP)

	Chemical 1	Chemical 2
Chemical 1	0.210	0.0
Chemical 2	0.0	0.176

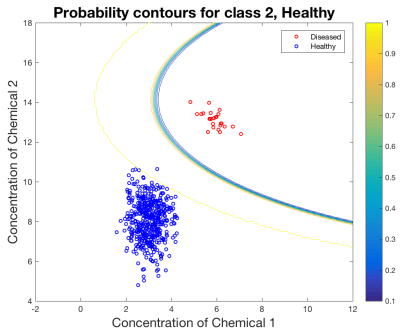
Table 3: Class 2 (Healthy) variance of Chemical Concentrations for ML (or MAP)

	Chemical 1	Chemical 2
Chemical 1	0.247	0.0
Chemical 2	0.0	1.045

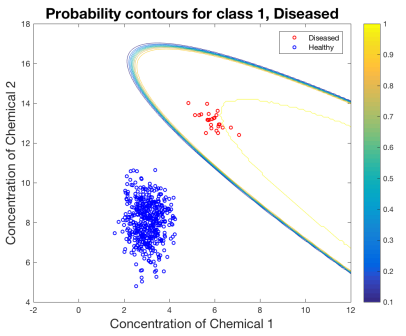
Refer to Figure 5 for the probability contours of the Maximum Likelihood. These plots show the likelihood of classification for any value of the 2 attributes. For example, in the top-right of Figure 5a, there is high probability of diseased classification as it is within the yellow contour line (probability $>$ 0.9).



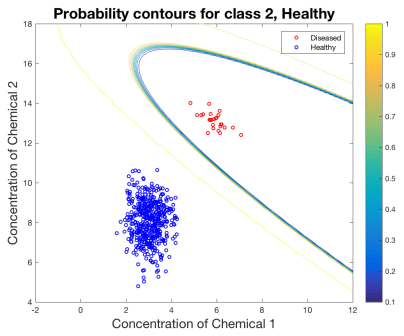
(a) ML probability contours for class 1



(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 ML probability contours for each class and with Naive and without Naive assumption

Table 4: Class 1 (Diseased) Covariance of Chemical Concentrations for ML (or MAP) WON

	Chemical 1	**Chemical 2**
Chemical 1	0.210	-0.124
Chemical 2	-0.124	0.176

Table 5: Class 2 (Healthy) Covariance of Chemical Concentrations for ML (or MAP) WON

	Chemical 1	**Chemical 2**
Chemical 1	0.247	-0.023
Chemical 2	-0.023	1.045

3 Task 2- Maximum a posteriori

In Task 2, we are asked to "train the Bayesian classifier, with Maximum a Posteriori (ML), on the training data (Refer Figure 1) with and without the Naive assumption." Once trained, we classify 2000 new data points (patients).

To train MAP with the Naive assumption is the same as ML, except we multiply the ML by the prior. Where the prior is the probability of each class. The affects of the prior is discussed further in Section 5. Refer Figure 6 for classification with the Naive assumption.

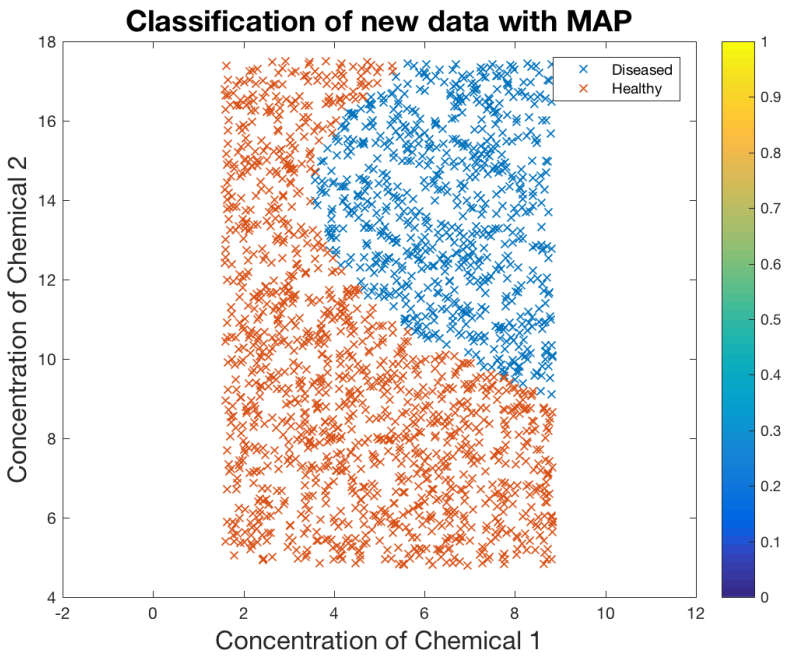


Figure 6: Maximum a posteriori classification of new data

Training MAP without the Naive assumption is similar to ML without the Naive assumption except we multiply by the prior. Refer Figure 7 for classification without the Naive assumption.

Refer Figure 8 for plots of the density contours for the 2 MAP methods. These plots show the mean of the data by the centre of the contours. The spread of this data is represented by the proximity of the contours i.e close contour lines indicate small spread. Dependence between the attributes is shown by the tilted contours i.e. downward tilt is a negative relationship.

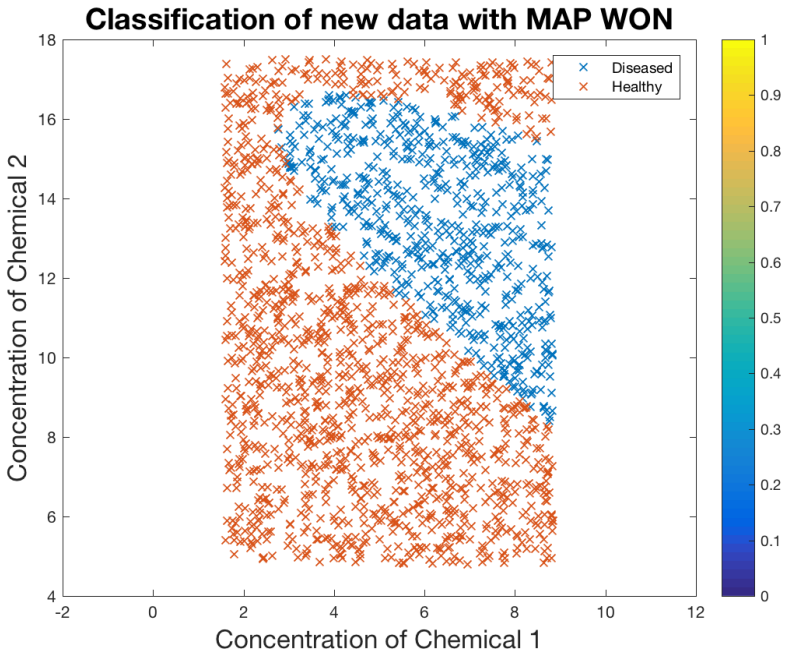


Figure 7: Maximum a posteriori without Naive assumption classification of new data

The mean is the same as before and is shown in Table 1. To reiterate, the healthy patients have mean (5.90, 13,11) and diseased have (3.02, 8.06).

The covariance is the same as outlined in the Maximum Likelihood method. Refer to the previous section for the explanation of interpreting the Tables 2, 3, 4, and 5.

Refer to Figure 9 for the probability contours of the Maximum a posteriori. These plots show the likelihood of classification for any value of the 2 attributes. For example, in the top-right of Figure 9a, there is high probability of diseased classification as it is within the yellow contour line (probability $>$ 0.9).

4 Comparison of Naive vs Without Naive

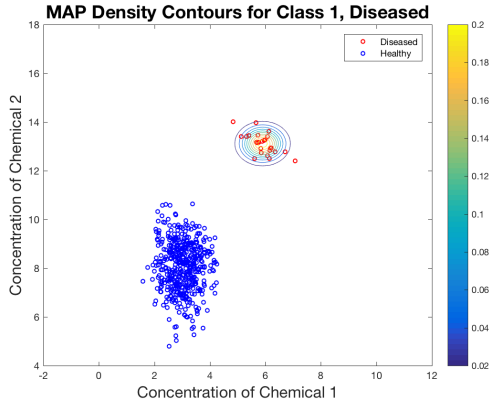
4.1 Introduction

For simplicity we refer only to Maximum Likelihood (ML) when comparing the Naive vs without Naive. For this case, MAP and ML are similar, so what is discussed here can also be applied to MAP. The difference between ML and Maximum a posteriori will be discussed later in the report.

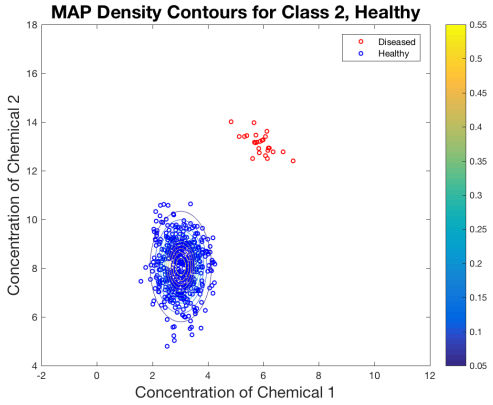
4.2 The means

The mean is the average of an attribute. If we consider a normal distribution for 1 attribute, it is the value at which the distribution is centred.

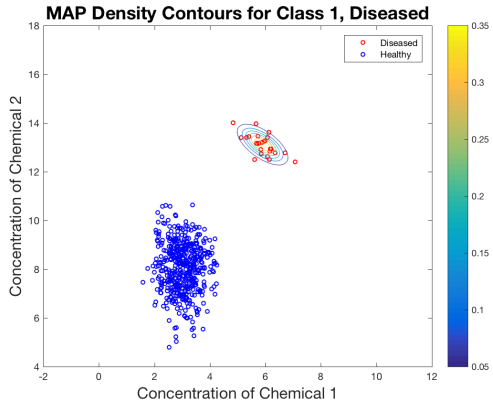
Consider Figure 1 and Table 1. We can see the means of each class are far apart, with the diseased mean at point (5.90, 13.13) and the healthy mean at (3.02, 8.06). Relative to the scale of this data, the means are indeed far apart. This has affects on the prior which are discussed in Section 5.



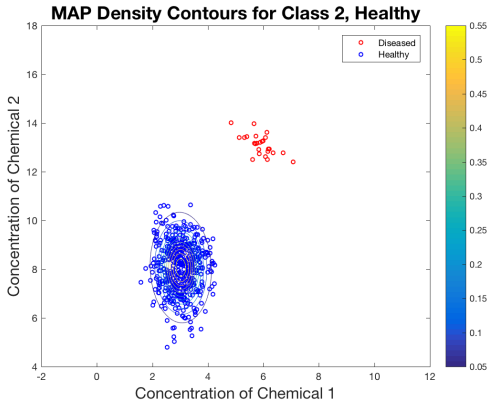
(a) MAP density contours for class 1



(b) MAP density contours for class 2



(c) MAPWON density contours for class 1



(d) MAPWON density contours for class 2

Figure 8: Comparison of MAP density contours for each class and with Naive and without Naive assumption

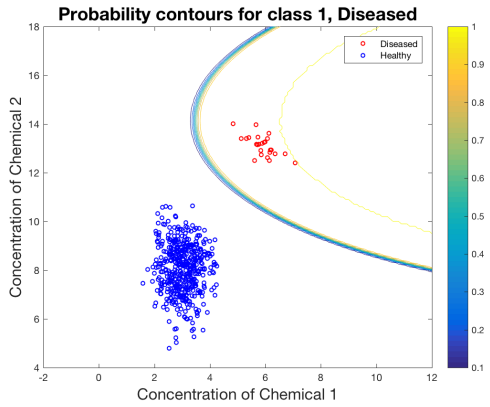
It should be noted the means remain the same for with and without the Naive assumption, also for ML vs MAP. This is because we calculate the mean on the training data, thus, new data or the method of analysis has no affect.

4.3 The Covariance

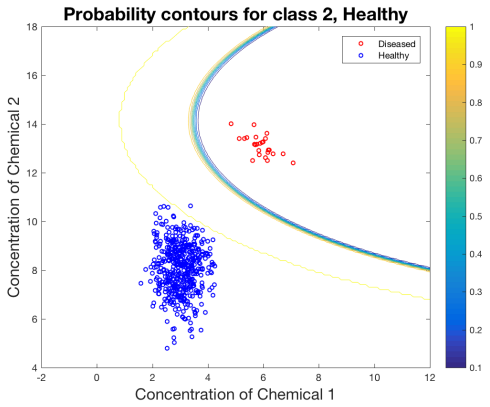
The covariance defines the shape and spread of the data. If we consider the normal distribution for 1 attribute again, a large covariance (or variance in this case for 1 attribute), indicates it will have a large spread from the centre.

Looking at Figures 4a and 4b, the contour plots of each class-conditional describe the covariance by their shape. For the naive assumption, the healthy and diseased contour plots are more or less circular. This is because we assume independence between the 2 attributes. The circular lines indicate that varying attribute 1 does not cause attribute 2 to vary. Tables 2 and 3 reinforce this observation and shows the covariances for diseased and healthy patients respectively. Notice the off diagonal elements are 0, meaning there is no relationship between attribute 1 and 2 i.e. a circular density contour.

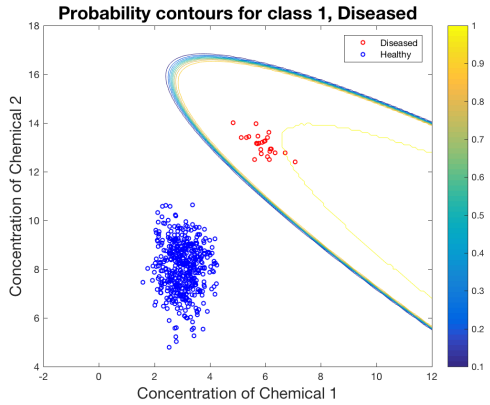
Moreover, the variance (covariance with 0 off diagonal elements) for healthy is relatively larger when compared to diseased, namely the chemical 2. This means chemical 2 for healthy patients can vary significantly along the vertical axis. We can see the affects of this in Figure 2, looking within the range



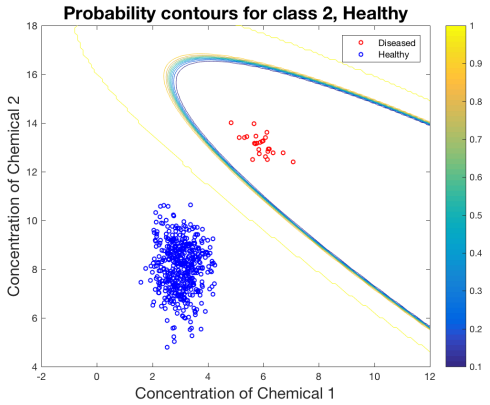
(a) MAP probability contours for class 1



(b) MAP probability contours for class 2



(c) MAPWON probability contours for class 1



(d) MAPWON probability contours for class 2

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

where chemical 1 = 4 to 5, healthy patients are classified above diseased again. This is because healthy has a chemical 2 variance of 1.045 vs 0.176 for diseased. This difference is an order of magnitude and means there is a high chance of healthy classification further away from its mean.

Without the Naive assumption, we see the dependence between the attributes on the healthy and diseased density contours in Figure 4c and 4d. However, even though we allow for dependence in this method, there is little relationship between the attributes for the healthy class. This is illustrated by the circular density contours.

The dependence of the attributes for diseased classification is shown by the tilted ellipse (contour lines). This downward trend indicates a negative relationship between the attributes for diseased patients. The shape of the healthy and diseased contour plots is further explained by the covariance in Table 4 and 5. Table 4 is the covariance for diseased patients and confirms this negative relationship by negative off diagonal elements. This means if one of the chemicals has high concentration, the patient is likely diseased, but If both are high, we start to see healthy patients again. This is shown in the top-right corner of Figure 7. The small values on the diagonal also explains the small spread of diseased patients, and why healthy patients start to dominate again. Small diagonal elements indicate that the MAP for diseased decreases rapidly as we travel from the mean.

Table 5 also has negative off diagonal elements but are an order of magnitude smaller. This means the attributes have little affect on one another and confirms our assumption that they are independent.

5 Maximum Likelihood vs Maximum a posteriori (ML vs MAP)

The main difference of ML and MAP is the prior in MAP. The prior allows us to specify belief about the data before we see it. For example, we can be biased against rare data, so we only classify it as rare data when the likelihood is high. Conversely, we might want to increase the chance of detecting the rare class. We can then set the prior to a high value for the rare class. This could be useful here as diseased classification is rare and that it is safer for patients to be misclassified as diseased and then use further tests to confirm diagnoses. The consequences of re testing is minor compared to misclassification of healthy when a patient is diseased. This does mean more healthy patients will be misclassified as diseased but as discussed before, we can test the patient again to confirm.

Several priors were trialled (Nc/N, 1/C, 0.99 vs 0.01), but varying the prior had little affect. This is due to the significant difference in means and small variances. In other words, the clusters of data points do not overlap so it is clear when a patient is diseased or not.

If the clusters overlapped by some amount, ambiguity of classification is introduced. This means the prior will have more affect. The likelihood will be less 'certain' in classifying overlapped points, therefore, the prior would provide the bias needed to classify the preferred, or more likely, class.

Looking at Table 6, we can see the difference in classification between the 4 methods. Looking at ML vs MAP, we can see the diseased has a difference of (826 - 764) 62. This is quite small and is expected for a data set with far apart means and small covariance.

Table 6: Classification Differences between Methods

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

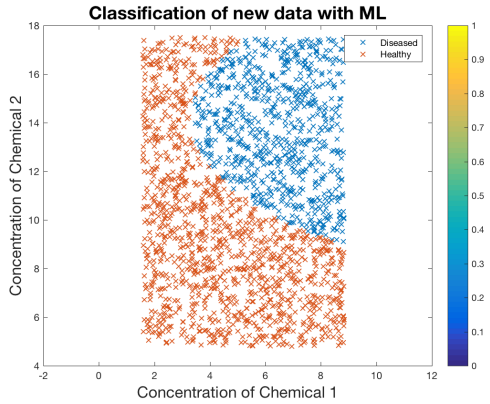
Looking at Figure 10, it is hard to notice any difference in classification. This is because of the small difference as discussed and show in Table 6.

6 Conclusion

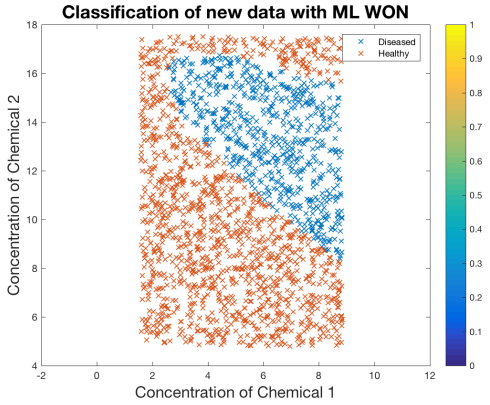
The Naive assumption has a significant influence on this data set indicating there is a relationship between the attributes.

ML and MAP had little difference in this case due to limited affect of the prior. The prior had little affect because of the large difference in means and the small covariances.

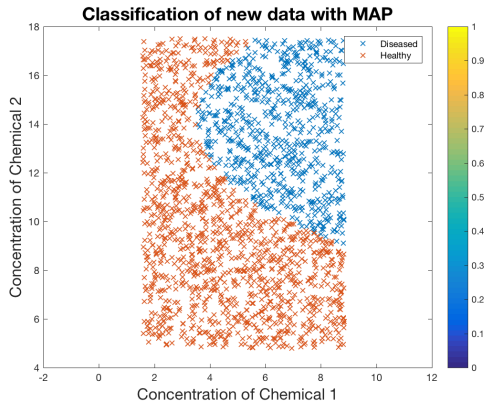
Considering the hypothesis of the training data - an increase in concentration of chemical 1 and 2 increases the likelihood of a patient being diagnosed as diseased. With the Naive assumption, it almost agrees with our hypothesis. However, we notice that healthy patients appear again above the diseased, therefore, we reject the hypothesis. Without the Naive assumption, there is a negative relationship between the attributes for diseased classification. In other words, if the concentration increases for one chemical but not the other, a patient is more likely diseased. Therefore, we reject the hypothesis for this case also.



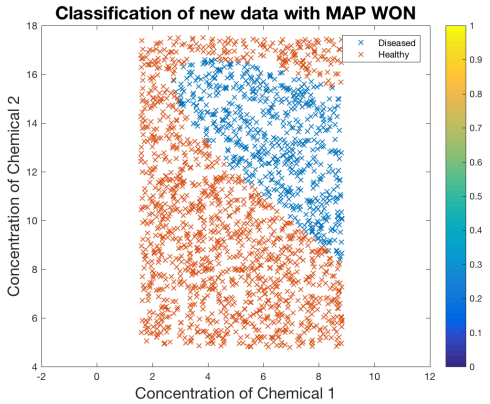
(a) Maximum Likelihood classification of new data



(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 10: Comparison of classification of new data by ML, MLWON, MAP, and MAPWON

7 Further Comments

It is safer to classify patients as diseased and re test for confirmation than classify for healthy when a patient is diseased. In other words, a false negative is more costly than a false positive. This is reinforced by the speed and reduced cost of the urine test. Considering this, it may be better to use the naive assumption to classify. But when diagnosing it would be best to look at both and make an educated decision.

We also discover the rate at which the ML (or MAP) 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 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 is a point where y value of line 1 becomes less than line 2 due to the greater negative slope. This point is when line 2 has greater y value i.e. when there is greater likelihood of class 2.

The probability contours in Figures 5 and 9 reinforce what we deduced in the density contour plots. This is 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 (or MAP) 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 and also indicate the decision boundary for the 2 classes.

8 References

[1] Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. 1st. Chapman & Hall/CRC, 2011. isbn: 1439824142, 9781439824146.

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
% Refe0ence 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

%% 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({'Classification of new data with ML'},'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({'Classification of new data with ML WON'},'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({'Classification of new data with MAP'},'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({'Classification of new data with MAP WON'},'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({'Classification of new data with MAP WON'},'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)];
```