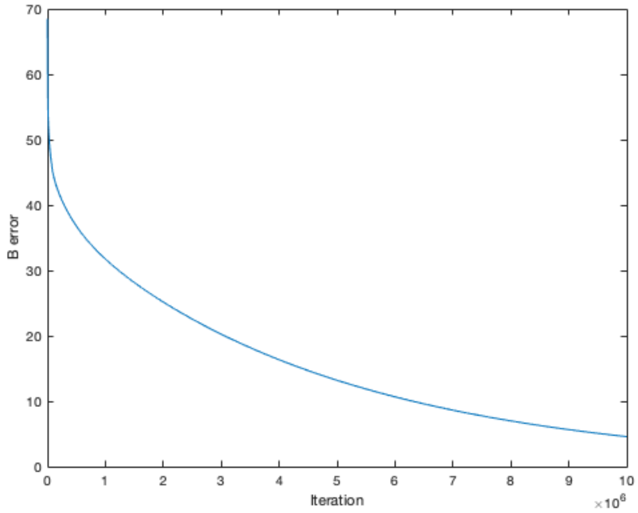
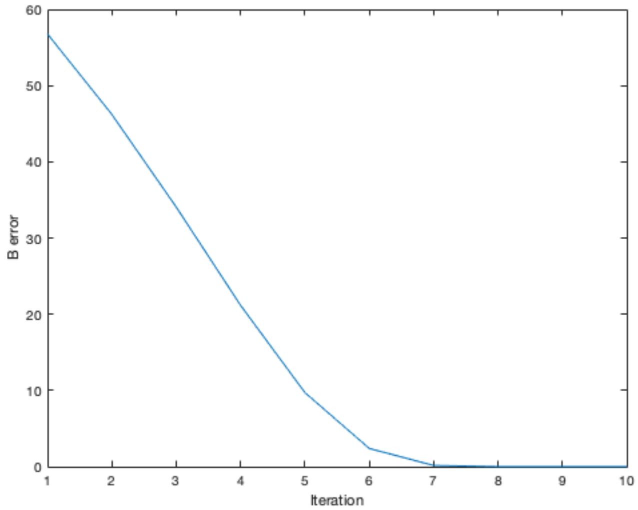
```
% Homework 3 - Question 1
clear all;
disp("1a) glmfit for Wisconsin Breast Cancer Dataset")
data = readtable('bc_wisc.csv');
data = data.Variables;
y = data(:,2);
X = data(:,[10, 23, 24, 30, 31]);

B_glm = glmfit(X,y,'binomial');
disp("Coeffecients of B");
disp(B_glm);
X_test = [ones(size(y)), X];
y_pred = X_test*B_glm>=0;
num_correct_predictions = sum(y_pred==y);
fprintf("Proportion of correction predictions = %f \n",num_correct_predictions/length(y));
disp(" ")

%1b

fprintf("1b) Logistic Regression using Gradient Descent iterates\n");
X = [ones(size(X,1),1), X];
B_old = zeros(6,1);

step_size = 0.00012;
num_iterations = 10000000;
err = zeros(num_iterations,1);

for i=1:num_iterations
    u = 1./(exp(X*B_old)+1);
    delta = (1-y).*X - u.*X;
    grad = sum(delta,1)';
    B_new = B_old - step_size*grad;
    err(i) = norm(B_new-B_glm);
    B_old = B_new;
end
fprintf("The following is the plot for the error in iterates of beta as a function of iterations.\nThis method converges takes a very long time tc

figure(1)
plot(err);
xlabel('Iteration')
ylabel('B error')
snapnow
disp("")
% 1c
fprintf("1c) Logistic Regression using Newton-Raphson iterates\n");
B_old = zeros(6,1);
num_iterations = 10;
err = zeros(num_iterations,1);
for i=1:num_iterations
    %calculate gradient
    u = 1./(exp(X*B_old)+1);
    delta = (1-y).*X - u.*X;
    grad = sum(delta,1)';
    %calcualte hessian matrix
    N = size(X,1);
    W = zeros(N);
    for j=1:N
        W(j,j) = exp(X(j,:)*B_old)/((exp(X(j,:)*B_old)+1).^2);
    end
    B_new = B_old - inv(X'*W*X) * grad;
    err(i) = norm(B_new-B_glm);
    B_old = B_new;
end
fprintf("The following is the plot for the error in iterates of beta as a function of iterations.\nThis method converges in just 8 iterations. \n"
figure(2)
plot(err);
xlabel('Iteration')
ylabel('B error')
snapnow;
disp("")
```

```
1a) glmfit for Wisconsin Breast Cancer Dataset
Coeffecients of B
  -34.2597
   53.8172
    1.0915
    0.2912
   22.8282
    9.8843

Proportion of correction predictions = 0.976786

1b) Logistic Regression using Gradient Descent iterates
The following is the plot for the error in iterates of beta as a function of iterations.
This method converges takes a very long time to converge.
```



1c) Logistic Regression using Newton-Raphson iterates
The following is the plot for the error in iterates of beta as a function of iterations.
This method converges in just 8 iterations.



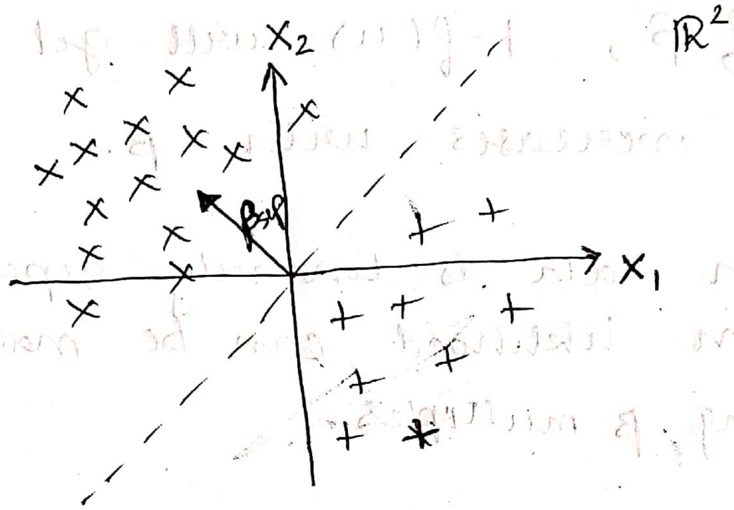
Published with MATLAB® R2018a

2.

a) $$L_\beta(X,y) = \prod_{i=1}^{n} L_\beta(x_i, y_i)$$

$$= \prod_{i=1}^{n} \left(\frac{1}{1+\exp(-x_i^T\beta)}\right)^{y_i} \left(\frac{\exp(-x_i^T\beta)}{1+\exp(-x_i^T\beta)}\right)^{1-y_i} \quad -①$$

b)



c) Given the property $f(u) = \frac{1}{1+\exp(-u)}$ being strictly increasing, lets take a look at the first part of eq ①

$\therefore$ $u = x_i^T\beta$

When the data is linearly seperable, $x_i^T\beta$ will be positive for all training points with $y_i = 1$.

So this part can always be made larger by increasing β (by multiples) - (deduced from that fact that $f(u)$ is strictly increasing

The second term in ①, is

$$\left(\frac{\exp(-x_i^T\beta)}{1+\exp(-x_i^T\beta)} \right)^{1-y_i}$$

which is equal to $1-f(u)$. Since the data is linearly seperable, $x_i^T\beta < 0$ for all training items with $y_i = 0$, on increasing multiples of β, $1-f(u)$ will get larger.

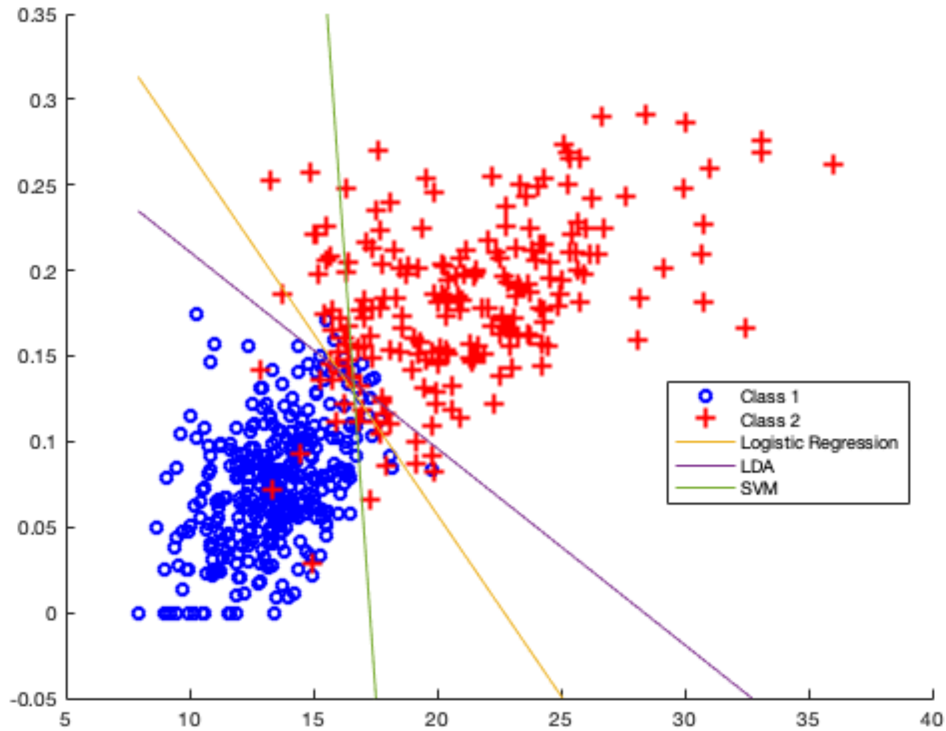
This part also increases with β.

Hence, when data is linearly seperable, the Maximum Likelihood can be made larger by increasing β multiples.

```
% Homework 3 - Question 3
clear all;
data = readtable('bc_wisc.csv');
data = data.Variables;


y = data(:,2);
X = data(:,3:end);
methods = ["logistic" ;"lda"; "svm"];
disp("3a) Below is the average accuracy for each fold in 5-fold cross validation ");

average_accuracy=perform_cross_validation(X, y, 5, methods);
disp("Logistic Reg    LDA     SVM ");
disp(average_accuracy);

disp("Average accuracy of LDA is highest in the all the folds, so we should select LDA");
disp(" ");
%3b
fprintf("3b) Visualisation of data.\n");
X = data(:,[23,30]);
y = data(:,2);
hold on
gscatter(X(:,1),X(:,2),y, 'br','o+')

B_glm = glmfit(X,y,'binomial');
x_axis = [min(X(:,1)), max(X(:,1))];
y_axis_1 = -(B_glm(1)+B_glm(2) * x_axis)/B_glm(3);

[class,err,POSTERIOR,logp,B_lda] = classify(X, X, y);
y_axis_2 = -(B_lda(1,2).const+B_lda(1,2).linear(1) * x_axis)/B_lda(1,2).linear(2);

svm_mdl = fitcsvm(X,y);
y_axis_3 = -(svm_mdl.Bias+svm_mdl.Beta(1) * x_axis)/svm_mdl.Beta(2);

fprintf("Below is the scatter plot of data considering two featues: 21 and 28 \n");
plot(x_axis,y_axis_1, x_axis,y_axis_2, x_axis,y_axis_3);
ylim([-0.05 0.35]);
legend('Class 1', 'Class 2', 'Logistic Regression', 'LDA', 'SVM');

function average_accuracy=perform_cross_validation(X,Y,k, methods)
    average_accuracy = zeros(k,size(methods,1));
    for j = 1:length(methods)
        chunk_size = size(Y,1)/k;
        for i=1:k
            index = (i * chunk_size) - chunk_size;
            X_test = X(index+1:index+chunk_size, :);
            Y_test = Y(index+1:index+chunk_size, :);
            X_train = [X(1:index, :);X(index+chunk_size+1:end, :)];
            Y_train = [Y(1:index);Y(index+chunk_size+1:end)];
            if methods(j) == "logistic"
                B_glm = glmfit(X_train,Y_train,'binomial');
                X_test = [ones(size(Y_test)), X_test];
                y_pred = X_test*B_glm>=0;
                average_accuracy(i,j) = sum(y_pred==Y_test)/length(y_pred);
            elseif methods(j) == "lda"
                y_pred = classify(X_test, X_train, Y_train);
                average_accuracy(i,j) =  sum(y_pred==Y_test)/length(y_pred);
            else
                mdl = fitcsvm(X_train,Y_train);
                average_accuracy(i,j) = sum(Y_test==predict(mdl, X_test))/length(y_pred);
            end

        end
```

```
    end
end
```

```
3a) Below is the average accuracy for each fold in 5-fold cross validation
Logistic Reg     LDA      SVM
    0.9375     0.9732     0.9554
    0.9196     0.9286     0.9196
    0.9732     0.9732     0.9732
    0.9286     0.9554     0.9464
    0.9464     0.9643     0.9554

Average accuracy of LDA is highest in the all the folds, so we should select LDA

3b) Visualisation of data.
Below is the scatter plot of data considering two featues: 21 and 28
```



Published with MATLAB® R2018a

```
data = readtable('wine.csv');
data = data.Variables;
y = data(1:end-50,1);
X = data(1:end-50,2:end);

y_val = data(end-49:end,1);
X_val = data(end-49:end,2:end);

%a One-vs-One Classification
fprintf("4a) One-vs-One Classification \n")
exclude_class = [1 ,2, 3];
y_pred = [];

for i=1:length(exclude_class)
    y_train = y(y~=exclude_class(i));
    X_train = X(y~=exclude_class(i),:);
    [class,err,POSTERIOR,logp,B_lda] = classify(X_val, X_train, y_train);
    y_pred = [y_pred class];
end

predictions = zeros(size(y_val));
ambiguous_classification = 0;correct_classification = 0;incorrect_classification = 0;
for i=1:length(X_val)
    if y_pred(i,1) ~= y_pred(i,2) && y_pred(i,1) ~= y_pred(i,3) && y_pred(i,2) ~= y_pred(i,3)
        ambiguous_classification = ambiguous_classification + 1;
    else
        predictions(i) = mode(y_pred(i,:));
    end
end

acc = sum(predictions==y_val)/length(y_val);
fprintf("Ambigous classification = %f \n", ambiguous_classification/length(X_val));
fprintf("Correct classification = %f \n", acc);
fprintf("Incorrect classification = %f \n", 1 - acc -ambiguous_classification/length(X_val) );
disp(" ")

%a One-vs-All Classification
fprintf("4b) One-vs-All Classification \n")
y = data(1:end-50,1);
X = data(1:end-50,2:end);

y_val = data(end-49:end,1);
X_val = data(end-49:end,2:end);

y_train(y==1)=1;
y_train(y~=1)=0;
[class1,~,~,~,coeff] = classify(X_val, X, y_train);
w1 = [coeff(1,2).const ; coeff(1,2).linear];

y_train(y==2)=1;
y_train(y~=2)=0;
[class2,~,~,~,~] = classify(X_val, X, y_train);

y_pred = [class1 class2];
predictions = zeros(size(y_val));
for i=1:length(X_val)
    if y_pred(i,1) == 1 &&  y_pred(i,2) == 1
        ambiguous_classification = ambiguous_classification + 1;
    elseif y_pred(i,1) == 1
        predictions(i) = 1;
    elseif y_pred(i,2) == 1
        predictions(i) = 2;
    else
        predictions(i) = 3;
    end
end
acc = sum(predictions==y_val)/length(predictions);

fprintf("Ambigous classification = %f \n", ambiguous_classification/length(y_val));
fprintf("Correct classification = %f \n", acc);
fprintf("Incorrect classification = %f \n", 1-acc-ambiguous_classification/length(y_val));
%}
disp(" ")
%c mnrfit
clear all;
```

```
fprintf("4c) Classification using mnrfit\n")
data = readtable('wine.csv');
data = data.Variables;
y = data(1:end-50,1);
X = data(1:end-50,2:end);

y_val = data(end-49:end,1);
X_val = data(end-49:end,2:end);

B = mnrfit(X,y);
y_pred = mnrval(B,X_val);
[~,I]=max(y_pred, [], 2);
acc = sum(y_val ==I);
acc = acc/length(y_pred);
fprintf("Correct classification = %f \n", acc);
fprintf("Incorrect classification = %f \n", 1-acc);
disp(" ")

%1d
fprintf("4d) Based on the validation accuracy, I would choose One-vs-All or One-vs-One Classification Method \n");
```

```
4a) One-vs-One Classification
Ambigous classification = 0.000000
Correct classification = 0.980000
Incorrect classification = 0.020000

4b) One-vs-All Classification
Ambigous classification = 0.000000
Correct classification = 0.980000
Incorrect classification = 0.020000

4c) Classification using mnrfit
Correct classification = 0.860000
Incorrect classification = 0.140000

4d) Based on the validation accuracy, I would choose One-vs-All or One-vs-One Classification Method
```

Published with MATLAB® R2018a