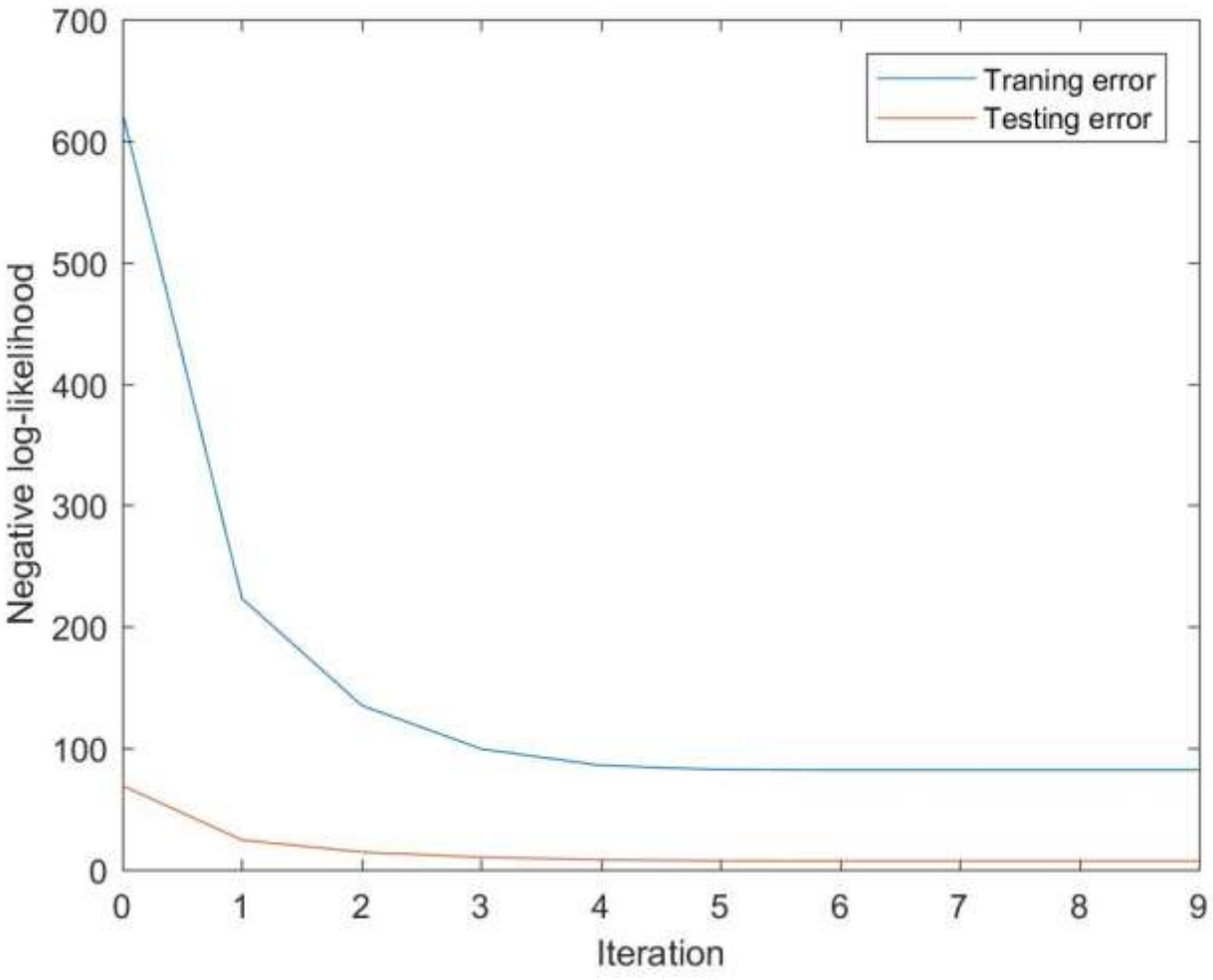
1.(c)

```
clear;clc;
filename = 'train_features.dat';
train_X = importdata(filename);
train_X = [ones(length(train_X),1) train_X];
filename = 'train_labels.dat';
train_Y = importdata(filename);
filename = 'test_features.dat';
test_X = importdata(filename);
test_X = [ones(length(test_X),1) test_X];
filename = 'test_labels.dat';
test_Y = importdata(filename);

eps = 1e-8;
w_update = zeros(3,1);
w = zeros(3,1);
diff = 1;
iter = 0;
training_error = [log_likelihood(train_Y,train_X,w)];
testing_error = [log_likelihood(test_Y,test_X,w)];
while diff >= eps
    iter = iter + 1;
    g = Gradient(train_Y,train_X,w);
    h_inv = inv(Hessian(train_X,w));
    w_update = w - h_inv*g;
    error1 = log_likelihood(train_Y,train_X,w_update);
    error2 = log_likelihood(test_Y,test_X,w_update);
    training_error = [training_error error1];
    testing_error = [testing_error error2];
    diff = abs(error1 - training_error(end-1));
    w = w_update;
end
x = 0:1:iter;
plot(x,training_error,x,testing_error);
xlabel('Iteration');
ylabel('Negative log-likelihood');
legend('Traning error','Testing error');

function l_w = log_likelihood(y,x,w)
    % y: n by 1, x: n by m+1, w: m+1 by 1
    l_w = -transpose(y)*x*w + sum(log(exp(x*w)+1));
end
```

```
function g = Gradient(y,x,w)
    linear_comb = -x*w;
    a = y - 1./(1+exp(linear_comb));
    g = -transpose(x)*a;
end
function H = Hessian(x,w)
    linear_comb = -x*w;
    a = exp(linear_comb);
    b = 1./(1 + exp(linear_comb)).^2;
    c = a.*b;
    xx = x.*repmat(c,1,3);
    H = transpose(x)*xx;
end
```



W = [-4.73878262951508, 4.40214932791691, -1.51521664732469]

Iteration = 9

5.(d)

```
clear;clc;
filename = 'diabetes_scale.csv';
data = csvread(filename);

train_X = data(1:500,2:end);
train_Y = data(1:500,1);
test_X = data(501:end,2:end);
test_Y = data(501:end,1);
C = linspace(0.1, 2, 20);
idx = crossvalind('Kfold', 500, 5);
rng(42);
%Soft-Margin
ce = zeros(20,1);
for i = 1:20
    SM_mdl = fitcsvm(train_X,train_Y,'KFold',5,'BoxConstraint',C(i));
    ce(i) = kfoldLoss(SM_mdl);
end
[~,I] = min(ce);
C_best = C(I);
SM_mdl = fitcsvm(train_X,train_Y,'BoxConstraint',C_best);
SM_label = predict(SM_mdl,test_X);
SM_accuracy = sum(SM_label==test_Y)/length(test_Y);

%Hard-Margin
HM_mdl = fitcsvm(train_X,train_Y,'BoxConstraint',1e6);
HM_label = predict(HM_mdl,test_X);
HM_accuracy = sum(HM_label==test_Y)/length(test_Y);
```

(i) Best C = 1.4, accuracy = 0.787313432835821

(ii) Accuracy = 0.317164179104478.

The C parameter means how much we want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. It makes the cost of misclassification high, thus forcing the algorithm to explain the input data stricter and potentially overfit. Therefore, it would cause a lower accuracy.

On the contrary, a small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. It makes the cost of misclassification low, allowing more of them due to a wider margin.