UNIVERSITY OF VICTORIA

Department of Electrical and Computer Engineering

ECE 503 Optimization for Machine Learning

LABORATORY REPORT

Experiment No: *Lab 2*

Title: *One-Versus-All Multi-Class Classification*

Date of Experiment: *Oct.22nd, 2025*

Report Submitted on: *Oct.22nd, 2025*

To: *Ruilin Wang*

Names: *Da Zhang (V01062902)*

1. Objective

The goal of this lab experiment is to build and employ an one-versus-all multi-class classification model predicting and classifying 3 types, namely Setosa, Vesicolor and Virginica, of Iris plants in the Fisher's 3-class dataset, where each sample Iris plants is represented by 4 features, namely lengths and widths of the sepal and petal of the flower.

2. Introduction

The method we employed in this experiment is an one-vs-all multi-class classification model, where the data is split into a training set with the first 35 samples and a training set with the last 15 samples per class, the optimal augmented weight parameters of each of the 3 binary classification is then computed based on the training set to acquire the optimal classifiers, $\widetilde{y_j} = \text{sign}\left(w_j^{*T} x + b_j^*\right) = \text{sign}\left(\widehat{w_j^*}^T \hat{x}\right)$*, by running Newton's Method algorism with Back Tracking Line Search algorism of finding the step sizes. After that, the optimal augmented parameters are normalized by scaling a factor* $\alpha = \frac{1}{|w_j|_2}$*, and a new data input's class label is obtained by computing* $w_j^{\text{nr}\,T} x + b_j^{\text{norm}}$*, in which the class j that maximize the "signed distance" is the predicted class.*$j^* = \arg\max_{j=1,\ldots,K} w_j^{\text{norm}\,T} \hat{x}$*. Lastly, the prediction model is run separately on the test and training sets with the number of Newton iterations set to K=5, 1, and 3, and the model performance is therefore evaluated in terms of confusion matrix and classification accuracy.*

3. Implementation and Results

3.1. Implementation

1. Load recourses: bring X_iris.mat into workspace and add the LCBR helper folder to the MATLAB path.
2. Partition Iris data into class-specific training/testing splits (35 training, 15 testing per class) following the Section 3 guidelines.
3. Build one-vs-all training matrices and shared label vector, then train three logistic regressors with Newton's method (K=5 iterations) using LRBC_newton.mat from the supplied library, in which the step sizes are calculated by using bt_lsearch2019.mat.
4. Normalize each weight vector to unit length to ensure comparable decision scores prior to multi-class assignment.
5. Classify samples by computing dot products with the normalized classifiers and selecting the argmax.

6. Convert feature matrices to homogeneous coordinates (append bias row of ones) and obtain predicted labels for all training samples.
7. Generate a confusion matrix without toolbox and then calculate overall training accuracy.
8. Repeat the training -> classification -> evaluation loop for alternative Newton ideration counts (K = 1 and K=3) to study convergence effects on the training set.
9. Mirror the test-set evaluation on the same model trained by the training data for K=5, K=3 and K=1 and compute confusion matrix and accuracy.

3.2. MATLAB code

```
%% 4.1 From the course website download X_iris.mat and add the LRBC folder
to the Matlab path

    load /Users/adamcheung/Documents/Uvic/Code/ECE503/Lab_2/X_iris.mat

    addpath('/Users/adamcheung/Documents/Uvic/Code/ECE503/Lab_2/LRBC');



    %% 4.2 Follow Section 3 to prepare training and test data

    X1 = X_iris (:, 1:50);

    Xtr1 = X1 (:, 1:35);

    Xte1 = X1 (:, 36:50);

    X2 = X_iris (:, 51:100);

    Xtr2 = X2 (:, 1:35);

    Xte2 = X2 (:, 36:50);

    X3 = X_iris (:, 101:150);

    Xtr3 = X3 (:, 1:35);

    Xte3 = X3 (:, 36:50);




    %% 4.3 Prepare code that applies Step 1 and 2 of the One-Versus-All 3-
class algo

    % Input: Prepare training data input for One-Versus_All 3-class Algo

    D_setosa_tr = Xtr1;
```

```
D_not_setosa_tr = [Xtr2 Xtr3];


D_versicolor_tr = Xtr2;
D_not_versicolor_tr = [Xtr1 Xtr3];


D_virginica_tr = Xtr3;
D_not_virginica_tr = [Xtr1 Xtr2];


X_setosa_tr = [D_setosa_tr D_not_setosa_tr];
X_versicolor_tr = [D_versicolor_tr D_not_versicolor_tr];
X_viginica_tr = [D_virginica_tr D_not_virginica_tr];


y = [ones(1,35) -ones(1,70)];


% Step 1: Apply binary classification to each One-Versus-All pair
[ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,5);
[ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,5);
[ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,5);


ws1 = ws_setosa(1:end-1);
ws2 = ws_versicolor(1:end-1);
ws3 = ws_virginica(1:end-1);


% Step 2: Normalize the 3 paris of parameters obatined from Step 1
ws_setosa_norm = ws_setosa/norm(ws1);
ws_versicolor_norm = ws_versicolor/norm(ws2);
ws_virginica_norm = ws_virginica/norm(ws3);
```

```
    % Step 3: Assigning the sample to the class with the largest decision
score

    function    predicted_labels    =    classifyByDot(X,    ws_setosa_norm,
ws_versicolor_norm, ws_virginica_norm)

        % Compute the decision scores for each column of X against the three
classifiers

        scores = [ws_setosa_norm' * X;

                  ws_versicolor_norm' * X;

                  ws_virginica_norm' * X];



        % Select the classifier with the largest score for each column
(1=setosa, 2=versicolor, 3=virginica)

        [~, predicted_labels] = max(scores, [], 1);



        % Return the labels as a 15x1 column vector

        predicted_labels = predicted_labels.';

    end




    %% 4.4 Apply the classifier to training set for K = 5 Newton iteration
and report confusion matrix and classification accuracy

    Xtr1_h = [Xtr1; ones(1,35)];

    Xtr2_h = [Xtr2; ones(1,35)];

    Xtr3_h = [Xtr3; ones(1,35)];



    y_tr_pred_1  =  classifyByDot(Xtr1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 1

    y_tr_pred_2  =  classifyByDot(Xtr2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 2
```

```
    y_tr_pred_3  =  classifyByDot(Xtr3_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 3



    y_tr_pred  =  [y_tr_pred_1'  y_tr_pred_2'  y_tr_pred_3'];  %%  Combine  all
predicted labels

    y_tr_true = [ones(1,35), 2*ones(1,35), 3*ones(1,35)]; %% Combine all true
labels



    % Compute confusion matrix

    C3_tr_k5 = zeros(3,3);

    for i = 1:length(y_tr_true)

        GroundTruth = y_tr_true(i);

        PredClass = y_tr_pred(i);

        C3_tr_k5(PredClass,      GroundTruth)      =      C3_tr_k5(PredClass,
GroundTruth)+1;

    end

    disp(C3_tr_k5);



    % Compute overall classification accuracy

    Accuracy_tr_k5 = trace(C3_tr_k5) / sum(C3_tr_k5(:));

    disp(Accuracy_tr_k5);




    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Replicate results for K = 1 Newton iteration on training data

    [ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,1);

    [ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,1);

    [ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,1);
```

```
    ws1 = ws_setosa(1:end-1);

    ws2 = ws_versicolor(1:end-1);

    ws3 = ws_virginica(1:end-1);


    ws_setosa_norm = ws_setosa/norm(ws1);

    ws_versicolor_norm = ws_versicolor/norm(ws2);

    ws_virginica_norm = ws_virginica/norm(ws3);


    Xtr1_h = [Xtr1; ones(1,35)];

    Xtr2_h = [Xtr2; ones(1,35)];

    Xtr3_h = [Xtr3; ones(1,35)];


    y_tr_pred_1  =  classifyByDot(Xtr1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 1

    y_tr_pred_2  =  classifyByDot(Xtr2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 2

    y_tr_pred_3  =  classifyByDot(Xtr3_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 3


    y_tr_pred = [y_tr_pred_1' y_tr_pred_2' y_tr_pred_3'];

    y_tr_true = [ones(1,35), 2*ones(1,35), 3*ones(1,35)];


    % Confusion Matrix

    C3_tr_k1 = zeros(3,3);

    for i = 1:length(y_tr_true)

        GroundTruth = y_tr_true(i);

        PredClass = y_tr_pred(i);
```

```
        C3_tr_k1(PredClass,      GroundTruth)      =      C3_tr_k1(PredClass,
GroundTruth)+1;

    end

    disp(C3_tr_k1);


    % Accuracy

    Accuracy_tr_k1 = trace(C3_tr_k1) / sum(C3_tr_k1(:));

    disp(Accuracy_tr_k1);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Replicate results for K = 3 Newton iteration on training data

    [ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,3);

    [ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,3);

    [ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,3);


    ws1 = ws_setosa(1:end-1);

    ws2 = ws_versicolor(1:end-1);

    ws3 = ws_virginica(1:end-1);


    ws_setosa_norm = ws_setosa/norm(ws1);

    ws_versicolor_norm = ws_versicolor/norm(ws2);

    ws_virginica_norm = ws_virginica/norm(ws3);


    y_tr_pred_1  =  classifyByDot(Xtr1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 1

    y_tr_pred_2  =  classifyByDot(Xtr2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 2
```

```
    y_tr_pred_3 = classifyByDot(Xtr3_h,ws_setosa_norm, ws_versicolor_norm,
ws_virginica_norm); %% Classify training samples from class 3


    y_tr_pred = [y_tr_pred_1' y_tr_pred_2' y_tr_pred_3'];

    y_tr_true = [ones(1,35), 2*ones(1,35), 3*ones(1,35)];


    % Confusion Matrix

    C3_tr_k3 = zeros(3,3);

    for i = 1:length(y_tr_true)

        GroundTruth = y_tr_true(i);

        PredClass = y_tr_pred(i);

        C3_tr_k3(PredClass,     GroundTruth)     =     C3_tr_k3(PredClass,
GroundTruth)+1;

    end

    disp(C3_tr_k3);


    % Accuracy

    Accuracy_tr_k3 = trace(C3_tr_k3) / sum(C3_tr_k3(:));

    disp(Accuracy_tr_k3);




    %% 4.4 Apply the classifier to testing set for K = 5 Newton iteration and
report confusion matrix and classification accuracy

    [ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,5);

    [ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,5);

    [ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,5);


    ws1 = ws_setosa(1:end-1);
```

```
    ws2 = ws_versicolor(1:end-1);

    ws3 = ws_virginica(1:end-1);



    ws_setosa_norm = ws_setosa/norm(ws1);

    ws_versicolor_norm = ws_versicolor/norm(ws2);

    ws_virginica_norm = ws_virginica/norm(ws3);



    Xte1_h = [Xte1; ones(1,15)];

    Xte2_h = [Xte2; ones(1,15)];

    Xte3_h = [Xte3; ones(1,15)];



    y_te_pred_1  =  classifyByDot(Xte1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 1

    y_te_pred_2  =  classifyByDot(Xte2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 2

    y_te_pred_3  =  classifyByDot(Xte3_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 3



    y_te_pred = [y_te_pred_1' y_te_pred_2' y_te_pred_3'];

    y_te_true = [ones(1,15), 2*ones(1,15), 3*ones(1,15)];



    % Confusion Matrix

    C3_te_k5 = zeros(3,3);

    for i = 1:length(y_te_true)

        GroundTruth = y_te_true(i);

        PredClass = y_te_pred(i);

        C3_te_k5(PredClass,      GroundTruth)      =      C3_te_k5(PredClass,
GroundTruth)+1;

    end
```

```
    disp(C3_te_k5);


    % Accuracy

    Accuracy_te_k5 = trace(C3_te_k5) / sum(C3_te_k5(:));

    disp(Accuracy_te_k5);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Replicate results for K = 1 Newton iteration on testing data

    [ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,1);

    [ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,1);

    [ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,1);


    ws1 = ws_setosa(1:end-1);

    ws2 = ws_versicolor(1:end-1);

    ws3 = ws_virginica(1:end-1);


    ws_setosa_norm = ws_setosa/norm(ws1);

    ws_versicolor_norm = ws_versicolor/norm(ws2);

    ws_virginica_norm = ws_virginica/norm(ws3);


    y_te_pred_1  =  classifyByDot(Xte1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 1

    y_te_pred_2  =  classifyByDot(Xte2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 2

    y_te_pred_3  =  classifyByDot(Xte3_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 3


    y_te_pred = [y_te_pred_1' y_te_pred_2' y_te_pred_3'];
```

```
    y_te_true = [ones(1,15), 2*ones(1,15), 3*ones(1,15)];


    % Confusion Matrix

    C3_te_k1 = zeros(3,3);

    for i = 1:length(y_te_true)

        GroundTruth = y_te_true(i);

        PredClass = y_te_pred(i);

        C3_te_k1(PredClass,      GroundTruth)      =      C3_te_k1(PredClass,
GroundTruth)+1;

    end

    disp(C3_te_k1);


    % Accuracy

    Accuracy_te_k1 = trace(C3_te_k1) / sum(C3_te_k1(:));

    disp(Accuracy_te_k1);




    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Replicate results for K = 3 Newton iteration on testing data

    [ws_setosa,C2_setosa] = LRBC_newton(X_setosa_tr,y,3);

    [ws_versicolor,C2_versicolor] = LRBC_newton(X_versicolor_tr,y,3);

    [ws_virginica,C2_virginica] = LRBC_newton(X_viginica_tr,y,3);


    ws1 = ws_setosa(1:end-1);

    ws2 = ws_versicolor(1:end-1);

    ws3 = ws_virginica(1:end-1);
```

```
    ws_setosa_norm = ws_setosa/norm(ws1);

    ws_versicolor_norm = ws_versicolor/norm(ws2);

    ws_virginica_norm = ws_virginica/norm(ws3);



    y_te_pred_1  =  classifyByDot(Xte1_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 1

    y_te_pred_2  =  classifyByDot(Xte2_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 2

    y_te_pred_3  =  classifyByDot(Xte3_h,ws_setosa_norm,  ws_versicolor_norm,
ws_virginica_norm); %% Classify testing samples from class 3



    y_te_pred = [y_te_pred_1' y_te_pred_2' y_te_pred_3'];

    y_te_true = [ones(1,15), 2*ones(1,15), 3*ones(1,15)];



    % Confusion Matrix

    C3_te_k3 = zeros(3,3);

    for i = 1:length(y_te_true)

        GroundTruth = y_te_true(i);

        PredClass = y_te_pred(i);

        C3_te_k3(PredClass,      GroundTruth)      =      C3_te_k3(PredClass,
GroundTruth)+1;

    end

    disp(C3_te_k3);



    % Accuracy

    Accuracy_te_k3 = trace(C3_te_k3) / sum(C3_te_k3(:));

    disp(Accuracy_te_k3);
```

3.3 Results

Results for 4.4. Report numerical results for classifier applied on training data

```
>> C3_tr_k5, Accuracy_tr_k5, C3_tr_k3, Accuracy_tr_k3, C3_tr_k1, Accuracy_tr_k1

C3_tr_k5 =

    35     0     0
     0    33     2
     0     2    33


Accuracy_tr_k5 =

    0.9619


C3_tr_k3 =

    35     0     0
     0    28     5
     0     7    30


Accuracy_tr_k3 =

    0.8857


C3_tr_k1 =

    35     0     0
     0    27     6
     0     8    29


Accuracy_tr_k1 =

    0.8667
```

Fig 1. Confusion Matrix and Classification Accuracy for classifier applied on training data with K = 5, 3, and 1 Newton Iterations

Results for 4.5. Report numerical results for classifier applied on testing data

```
>> C3_te_k5, Accuracy_te_k5, C3_te_k3, Accuracy_te_k3, C3_te_k1, Accuracy_te_k1

C3_te_k5 =

    15     0     0
     0    15     1
     0     0    14


Accuracy_te_k5 =

    0.9778


C3_te_k3 =

    15     0     0
     0    15     1
     0     0    14


Accuracy_te_k3 =

    0.9778


C3_te_k1 =

    15     0     0
     0    11     3
     0     4    12


Accuracy_te_k1 =

    0.8444
```

Fig 2. Confusion Matrix and Classification Accuracy for classifier applied on testing data with K = 5, 3, and 1 Newton Iterations

4. Discussion

In the following discussion, Setosa, Versicolor and Virginica are represented by class 1, 2 and 3 respectively.

With five newton steps, the one-vs-all logistic classifier almost perfectly separates the Iris classes. Training confusion C3_tr_k5 mislabels only four sample (two class-2 and two class-3), giving 0.9619 accuracy; the test matrix keeps just a single class-3 classified as class-2, with an accuracy of 0.9778. This indicates the optimization has essentially converged and generalizes cleanly. Reducing to three Newton updates weakens the fit on the training split, which has an accuracy of 0.8857. The confusion matrices show every error is still restricted to the class 2 and 3 boundary (5 class-2 predicted as class 3 and seven class 3 predicted as class 2). Despite the looser decision boundary, the test set remains strong accuracy, 0.9778, because only one of those borderline samples appears in the held-out data. The comparable test accuracy suggests mild underfitting on the training data rather than overfitting. With a single Newton step the model struggles to separate the two non-Setosa classes on both sets. Training accuracy drops to 0.8667, and eight class-2 and six class-3 points are swapped; test accuracy falls to 0.8444 with similar confusions. This is consistent with Newton's method needing more than one iteration to reach the optimum, as we are effectively observing an early-stopped solution where the weight vectors are still close to their initialization.

Across all settings the class-1 remains perfectly predicted. Because its samples are linearly separable from the rest, even under-converged models align well with the discriminant; the performance gap stems entirely from the tighter class overlap between class-2 and 3

5. Conclusion

In this experiment, we successfully implemented a one-vs-all multi-class classification model to classify Iris flower species using Newton's method with backtracking line search. The experiment illustrates the importance of allowing the Newton optimizer to converge: a small number of iterations (greater than or equal to 3 in this lab) is enough to capture the class structure, while too few leave noticeable confusion between the classes that are tougher to separate (Versicolor and Virginica in this lab). Overall, this experiment illustrates that giving Newton's method enough iterations is essential; once the optimizer fully settles, the linear decision boundaries capture the class structure reasonably well without overfitting.

6. References

None.