UNIVERSITY OF VICTORIA

Department of Electrical and Computer Engineering

ECE 403 Optimization for Machine Learning

LABORATORY REPORT

Experiment No: 3

Title: Breast Cancer Diagnosis via Logistic Regression

Date of Experiment: July 7, 2020

Report Submitted on: July 14, 2020

To: Lei Zhao

Names: Ashlynn Steeves (V00850631)

Table of Tables

Table of Figures

1. Objective

The objective of this experiment is to gain experience with logistic regression by implementing a system to diagnose patients with breast cancer based on 30 medical features. This experiment also highlights important role normalization can play in the preprocessing of data.

2. Introduction

This experiment uses a logistic regression based binary classifier to classify breast tumors as malignant or benign. For the dataset at hand it is important to first normalize the data as the feature scales vary significantly. Without correction this variance can negatively impact classification. The normalized dataset is constructed by first computing the mean and variance of each feature across all samples. Then, the computed mean is subtracted form each sample and this value is then divided by the computed variance to obtain the normalized value. This process can be denoted mathematically as:

$$\widetilde{\boldsymbol{u}}_i = \frac{1}{\sigma_i}\left[u_1^i - m_i \;\; u_2^i - m_i \;\; \ldots \;\; u_P^i - m_i \;\right] \; for \; i = 1,2,\ldots,N \quad (1)$$

Where P denotes the number of samples in a dataset, N denotes the number of features, σ_i is the variance across feature i, m_i is the mean of feature i. An important note to make regarding normalization of features for classification is that the testing dataset represents unseen data. Therefore, the normalization applied to all data should be derived solely from the training data.

The logistic regression based binary classifier used in this experiment classifies tumors as malignant if the returned label is 1 and classifies tumors as benign if the returned label is -1. This classifier classifies an unseen input **x** using the function:

$$\tilde{y} = sign(\boldsymbol{w}^{*T}\boldsymbol{x} + b^*) = sign(\widehat{\boldsymbol{w}}^{*T}\widehat{\boldsymbol{x}}) \quad (2)$$

Where $\widehat{\boldsymbol{w}}^* = \begin{bmatrix}\boldsymbol{w}^* \\ b^*\end{bmatrix}$ is obtained by minimizing a variant of the softmax cost function which includes a regularization term:

$$E_L(\widehat{\boldsymbol{w}}) = \frac{1}{P}\sum_{p=1}^{P}\log\left(1 + e^{-y_p\widehat{\boldsymbol{x}}_p^T\widehat{\boldsymbol{w}}}\right) + \frac{\mu}{2}\|\widehat{\boldsymbol{w}}\|_2^2 \quad (3)$$

The addition of the regularization term to the softmax function ensures that the magnitude of $\widehat{\boldsymbol{w}}$ stays within a reasonable range in order to avoid overfitting of the training data. As the value of mu in the regularization term increases it begins to dominate the softmax cost function consequentially pulling $\widehat{\boldsymbol{w}}$ away from a single point near its minimizer towards a sufficiently sparse $\widehat{\boldsymbol{w}}$.

The results of this classifier can be numerically evaluated in a variety of ways including a confusion matrix, which details the classification and misclassifications of each sample. The elements along the diagonal of a confusion matrix represent the samples that were classified correctly and the off diagonal represents misclassifications, where the column denotes the true class of the sample and the row denotes the predicted class of the sample. The confusion matrix in our case will look like:

Table 1 – Confusion Matrix Example

	Malignant	**Benign**
Malignant	C(1,1) Correctly classified as Malignant	C(1,2) Benign misclassified as Malignant
Benign	C(2,1) Malignant misclassified as Benign	C(2,2) Correctly classified as Benign

Another method of numerical evaluation is classification accuracy. This value is simply a percentage that denotes the number of samples that were classified correctly and can be calculated as:

$$Classification\ Accuracy = 100 * \frac{\sum_{i=1}^{3} C(i,i)}{\sum_{i=1}^{3}\sum_{j=1}^{3} C(i,j)}\% \quad (4)$$

3. Implementation and Results

3.1. Implementation

This experiment was implemented using a MATLAB script that followed these steps:

1. Load the provided breast cancer diagnosis dataset of 569 patient samples
2. Split the dataset into a training set (285 samples) and a testing set (284 samples)
3. Normalize of all of the data in the training set
4. Normalize of all of the data in the testing set using data from the training set
5. Use the training set to obtain parameters $\{\boldsymbol{w}^*, b^*\}$ by means of minimizing the regularized softmax function via gradient descent.
    a. This step was repeated four times with varying parameters for mu and K
6. Apply the 4 classifiers generated in step 5 to the testing set and evaluate their performance using confusion matrices
7. Bonus Question
    a. Create linearly spaced array of mu values with 100 entries between 0 and 1 and again between 1 and 100
    b. Loop through the arrays and obtain parameters $\{\boldsymbol{w}^*, b^*\}$ for each value of mu
    c. Apply the generated classifier at each value of mu to the testing dataset and record the classification accuracy
    d. Plot the classification accuracy as a function of mu and return the optimal value of mu from each array.

3.2. MATLAB code

The MATLAB code used for this experiment is shown below. Please note that this script was generated as a MATLAB live script, which allows plain text to be included in a file. Furthermore, this script calls functions f_wdbc(), g_wdbc(), GD_Lab3 and bt_lsearch2019() which were provided and can be found at: https://studentweb.uvic.ca/~leizhao/code.html

This script in its entirety can be found at: https://github.com/ashlynns/ECE403/tree/master/Lab3

4.1 From the course website download data matrix D_wdbc.mat.

```
load('D_wdbc.mat')
```

4.2 Follow Sec. 3.1 above to generate normalized train and test data sets and their labels.

```
Dtr = D_wdbc(:,1:285); % traning data
Dte = D_wdbc(:,286:569); % testing data

% Normalization of the tranning set
Xtr = zeros(30,285);
m = zeros(1,30);
v = zeros(1,30);
for i = 1:30
 xi = Dtr(i,:);
 m(i) = mean(xi);
 v(i) = sqrt(var(xi));
 Xtr(i,:) = (xi - m(i))/v(i);
end

% Normalizing the testing set with data from the tranning set
Xte = zeros(30,284);
for i = 1:30
 xi = Dte(i,:);
 Xte(i,:) = (xi - m(i))/v(i);
end

% Data labels
ytr = Dtr(31,:);
yte = Dte(31,:);
```

4.3 Based on (E4.2) and (E4.3), prepare two MATLAB functions for evaluating the regularized cost function and its gradient, respectively.

Functions provided. Included at the bottom of this script

4.4 Prepare MATLAB code to minimize ELR(w) in (E4.2) using the GD algorithm (see Sec. 2.3 of the course notes).

Functions provided. Included at the bottom of this script

4.5 Set initial point w= **0** and run the code prepared in Item 4.4 to obtain parameters {w,b} for the following settings.

(i) mu= 0 and K=10;

(ii) mu= 0.1 and K= 10;

(iii) mu= 0 and K= 30;

(iv) mu=0.075 and K= 30 .

```
w = zeros(31,1);
fname = 'f_wdbc';
gname = 'g_wdbc';

%i
w_i = GD_Lab3(fname,gname, w, [Xtr; ytr], 0, 10);
%ii
w_ii = GD_Lab3(fname,gname, w, [Xtr; ytr], 0.1, 10);
%iii
w_iii = GD_Lab3(fname,gname, w, [Xtr; ytr], 0, 30);
%iv
w_iv = GD_Lab3(fname,gname, w, [Xtr; ytr], 0.075, 30);
```

4.6 Use the four solutions obtained in 4.5 above to specify the classifier in (E4.4) and apply it to the 284 test samples prepared in Item 4.2 above. Report the confusion matrix for each case, and comment on the results obtained.

```
%i
yp_i = sign((w_i(1:30,:)'*Xte)+w_i(31));
confusion_i = confusionmat(yte, yp_i)

%i
yp_ii = sign((w_ii(1:30,:)'*Xte)+w_ii(31));
confusion_ii = confusionmat(yte, yp_ii)

%i
yp_iii = sign((w_iii(1:30,:)'*Xte)+w_iii(31));
confusion_iii = confusionmat(yte, yp_iii)

%i
yp_iv = sign((w_iv(1:30,:)'*Xte)+w_iv(31));
confusion_iv = confusionmat(yte, yp_iv)
```

Bonus question

```
mu_vec_1 = linspace(0, 1, 100);
accuracy_1 = zeros(size(mu_vec_1));
mu_vec_2 = linspace(1, 100, 100);
accuracy_2 = zeros(size(mu_vec_2));

for i = 1:length(mu_vec_1)
    mu1 = mu_vec_1(i);
    mu2 = mu_vec_2(i);

    w_1 = GD_Lab3(fname,gname, w, [Xtr; ytr], mu1, 30);
    w_2 = GD_Lab3(fname,gname, w, [Xtr; ytr], mu2, 30);

    yp_1 = sign((w_1(1:30,:)'*Xte)+w_1(31));
    confusion_1 = confusionmat(yte, yp_1);
    accuracy_1(i) = (sum(diag(confusion_1))/284)*100;

    yp_2 = sign((w_2(1:30,:)'*Xte)+w_2(31));
    confusion_2 = confusionmat(yte, yp_2);
    accuracy_2(i) = (sum(diag(confusion_2))/284)*100;
end

[~,highest_acc1_idx] = max(accuracy_1);
best_mu1 = mu_vec_1(highest_acc1_idx)

[~,highest_acc2_idx] = max(accuracy_2);
best_mu2 = mu_vec_2(highest_acc2_idx)

plot(mu_vec_1, accuracy_1)
title('Bonus Question 1')
xlabel('mu')
ylabel('Classification Accuracy')

plot(mu_vec_2, accuracy_2)
title('Bonus Question 2')
xlabel('mu')
ylabel('Classification Accuracy')
```

```
function f = f_wdbc(w,D,mu)
X = D(1:30,:);
y = D(31,:);
P = length(y);
f = 0;
for i = 1:P
    xi = [X(:,i); 1];
    fi = log(1+exp(-y(i)*(w'*xi)));
    f = f + fi;
end
f = f/P + 0.5*mu*norm(w)^2;
end

function g = g_wdbc(w,D,mu)
X = D(1:30,:);
y = D(31,:);
P = length(y);
g = zeros(31,1);
for i = 1:P
    xi = [X(:,i); 1];
    ei = exp(y(i)*(w'*xi));
    gi = -y(i)*xi/(1+ei);
    g = g + gi;
end
g = g/P + mu*w;
end

function xs = GD_Lab3(fname,gname,x0,D, mu, K)
format compact
format long
k = 1;
xk = x0;
gk = feval(gname, xk, D, mu);
dk = -gk;
ak = bt_lsearch2019(xk,dk, fname,gname,D, mu);
adk = ak*dk;
while k <= K
  xk = xk + adk;
  gk = feval(gname,xk, D, mu);
  dk = -gk;
  ak = bt_lsearch2019(xk,dk, fname,gname, D, mu);
  adk = ak*dk;
  k = k + 1;
end
xs = xk + adk;
end

function a = bt_lsearch2019(x,d,fname,gname,p1,p2)
rho = 0.1;
gma = 0.5;
x = x(:);
d = d(:);
a = 1;
xw = x + a*d;
parameterstring ='';
if nargin == 5
   if ischar(p1)
      eval([p1 ';']);
   else
      parameterstring = ',p1';
   end
end
if nargin == 6
   if ischar(p1)
      eval([p1 ';']);
```

```
   else
      parameterstring = ',p1';
   end
   if ischar(p2)
      eval([p2 ';']);
   else
      parameterstring = ',p1,p2';
   end
end

eval(['f0 = ' fname '(x' parameterstring ');']);
eval(['g0 = ' gname '(x' parameterstring ');']);
eval(['f1 = ' fname '(xw' parameterstring ');']);

t0 = g0'*d;
f2 = f0 + rho*a*t0;
er = f1 - f2;
while er > 0
     a = gma*a;
     xw = x + a*d;
     eval(['f1 = ' fname '(xw' parameterstring ');']);
     f2 = f0 + rho*a*t0;
     er = f1 - f2;
end
if a < 1e-5
   a = min([1e-5, 0.1/norm(d)]);
end
end
```

3.3. Results

Results for 4.5 – Application of the classifier to the testing data where parameters $\{\boldsymbol{w}^*, b^*\}$ were generated under a variety of conditions. Evaluated by confusion matrices and classification accuracy.

(i) $\mu = 0, K = 10$

Table 2 – Confusion Matrix for Case i

	Malignant	**Benign**
Malignant	167	4
Benign	3	110

Classification Accuracy = 97.5%

(ii) $\mu = 0.1, K = 10$

Table 3 – Confusion Matrix for Case ii

	Malignant	**Benign**
Malignant	170	1
Benign	4	109

Classification Accuracy = 98.2%

(iii) $\mu = 0, K = 30$

Table 4 – Confusion Matrix for Case iii

	Malignant	**Benign**
Malignant	169	2
Benign	5	108

Classification Accuracy = 97.5%

(iv) $\mu = 0.075, K = 30$

Table 5 – Confusion Matrix for Case iv

	Malignant	**Benign**
Malignant	171	0
Benign	4	109

Classification Accuracy = 98.6%

Results for Bonus Question 1 – The impact of mu on classification accuracy over a range of [0,1]

The mu in the range [0,1] that resulted in the highest accuracy was 0.0707…

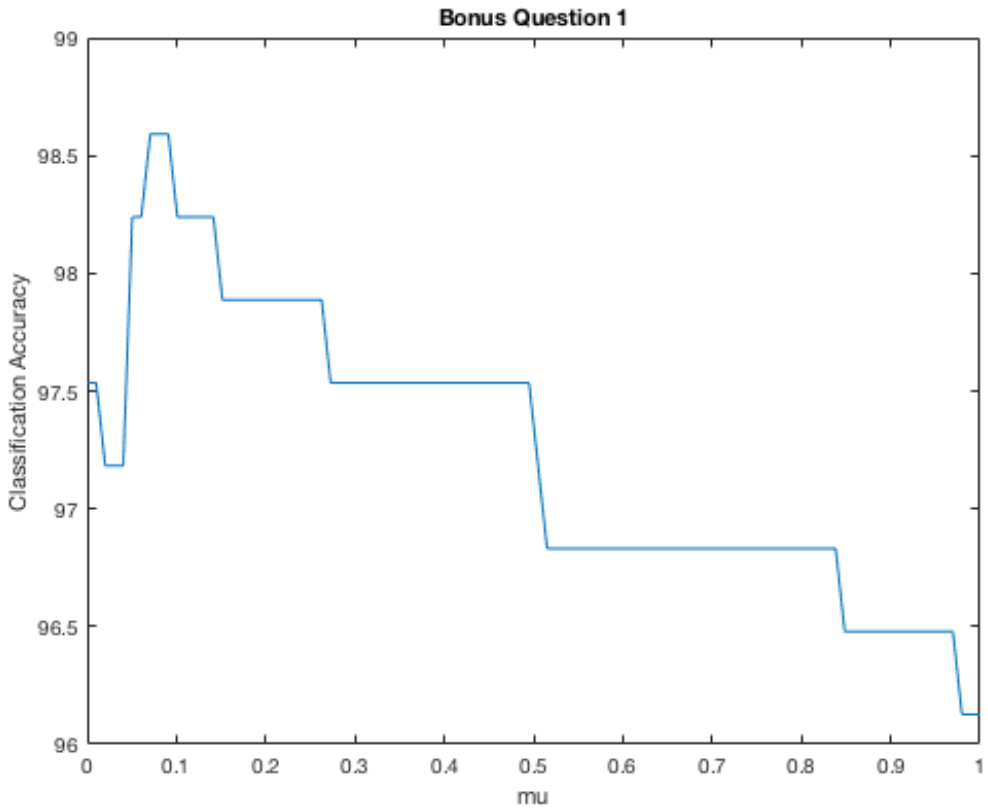

Figure 1 – Bonus Question 1

Results for Bonus Question 2 – The impact of mu on classification accuracy over a range of [1,100]

The mu in the range [1,100] that resulted in the highest accuracy was 1

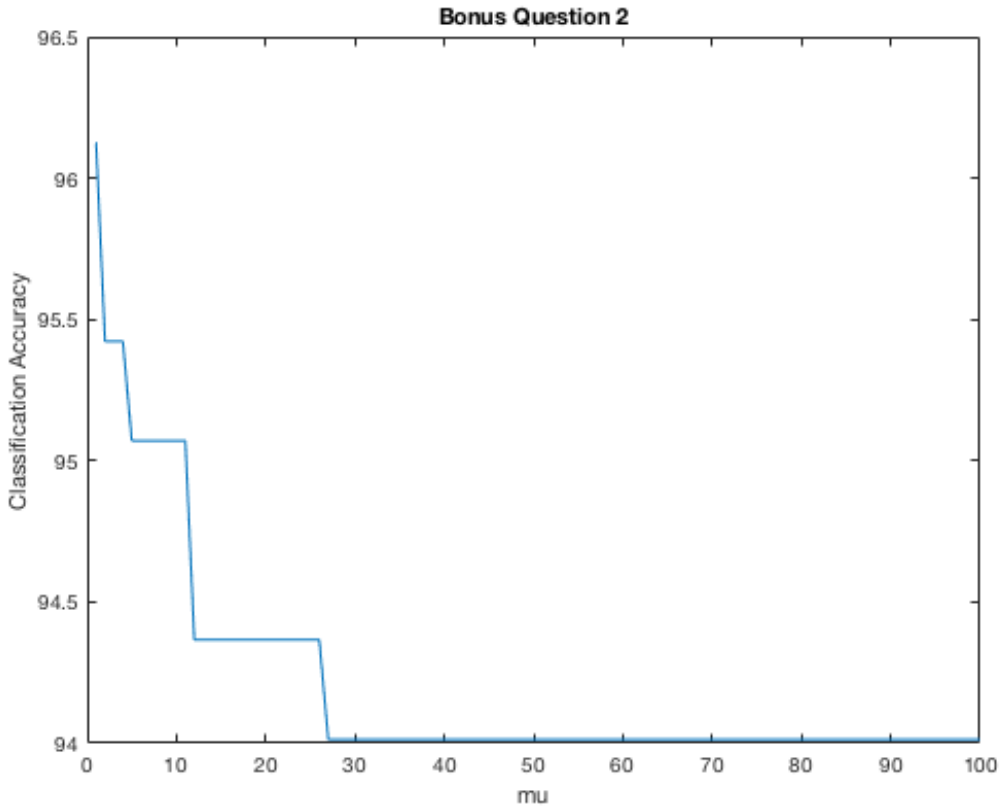

Figure 2 – Bonus Question 2

4. Discussion

The results obtained in this experiment are as expected. Furthermore, the results obtained in section 4.5 and in the bonus question validate one another. From the results of the bonus question we know that the optimal value of mu is at 0.0707, and case iv, which used a mu value of 0.075 returned the highest classification accuracy of 98.6%. Case ii, which used a mu value of 1, had a slightly lower classification accuracy of 98.2% and cases i and iii which used mu values of zero had classification accuracies of 97.5%.

5. Conclusion

In this experiment a logistic regression based binary classifier was used to classify breast tumors as malignant or benign. This was executed in a MATLAB script using a dataset containing 569 samples each including 30 medical features and a label representing their diagnosis. The dataset was normalized to account for the variance across the feature scales. The optimal weight and bias were then calculated from the normalized training data for a variety of cases by minimizing a variant of the softmax cost function, which includes a regularization term. The case that used a mu value of 0.075 and 30 iterations performed best as it yielded classification accuracy of 98.6%. This result was as expected and confirmed by the bonus question which evaluated the of classification accuracies across different ranges of mu.

6. References

[1] Lu, W., 2020. *Laboratory Manual - OPTIMIZATION For MACHINE LEARNING*. [PDF] Victoria. Available at: <https://www.ece.uvic.ca/~wslu/403/403pass/Trans/Exp1-2020.pdf> [Accessed 2 June 2020].