zl9901 Homework 3 Report

Problem 1 The logistic regression classifier

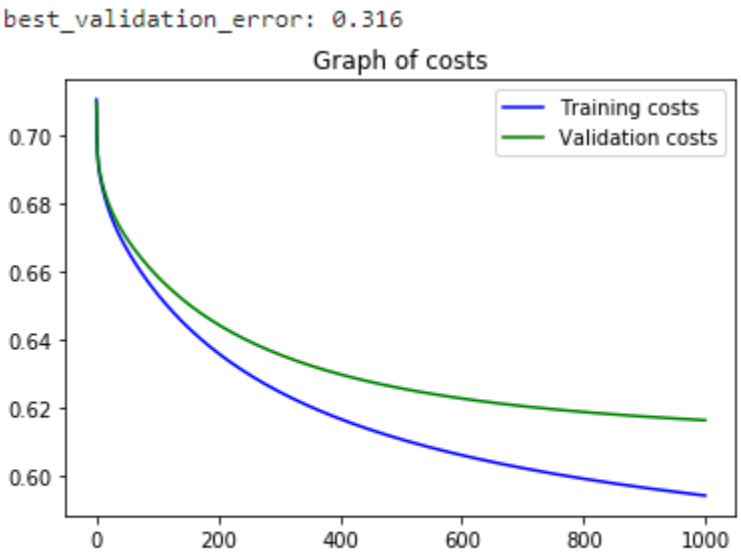

Figure 1. The logistic regression classifier

vi As the figure shown above, the best error value on the validation data is 0.316.

vii The plot of my training and validation costs is shown as Figure 1 above.

viii Accuracy of test set is 0.7096666666666667.

Problem 2 The neural network classifier

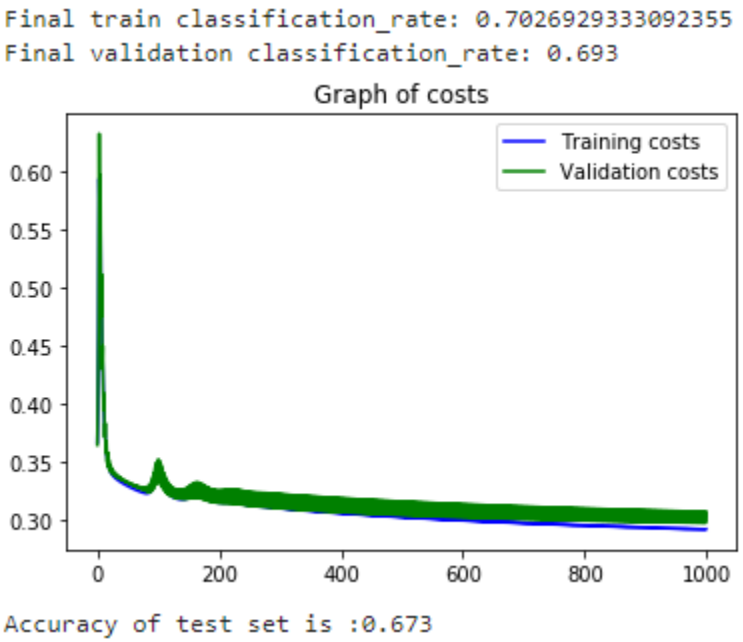

Figure 2. The neural network classifier

vii As the figure shown above, the training classification rate is 0.702692933309235 and the validation classification rate is 0.693.

viii The plot of my training and validation errors which shows the errors change with time can be viewed in Figure 2.

ix The accuracy for predicting new dataset from my trained network is 0.673.

Problem 3 Adding regularizers

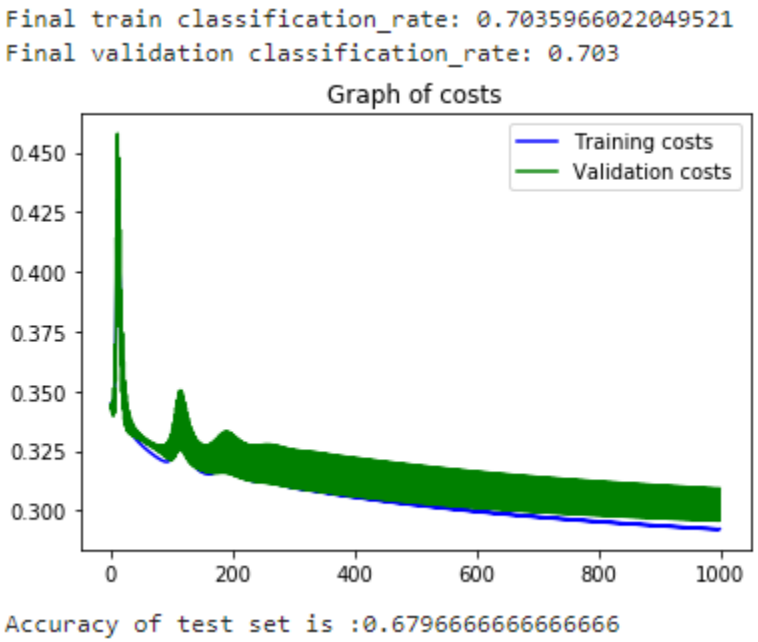

Figure 3. The neural network classifier adding regularizers

```
self.W2-=step_size*(np.dot(Ztrain.T,(Ytrain-Ytrain_ind))+self.W2)
self.b2-=step_size*(Ytrain-Ytrain_ind).sum(axis=0)


#(v) Then we propagate the errors we got from the previous layer W2 update W1 and b1
#ADD CODE HERE...
dZ=np.dot(Ytrain-Ytrain_ind,self.W2.T)*(1-Ztrain*Ztrain)
self.W1-=step_size*(np.dot(X.T,dZ)+self.W1)
self.b1-=step_size*dZ.sum(axis=0)
```

I choose to use λ=1.0 here.

a I choose to use L2 regularizer for my homework.

b New classification rate of training datasets is 0.7035966022049521 and new classification rate of validation datasets is 0.703.

c Because regularization penalizes very large weight, we can accomplish this by adding penalty function to our initial cost function.
We can conclude from statistic result that the accuracy of test set with regularization is higher than the model without regularizers.

Problem 4 Training with SVM

```
linear kernel ---------
[[5033 1007]
 [1230 4796]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.82      6040
           1       0.83      0.80      0.81      6026

    accuracy                           0.81     12066
   macro avg       0.82      0.81      0.81     12066
weighted avg       0.82      0.81      0.81     12066

rbf kernel ---------
[[5445  595]
 [ 843 5183]]
              precision    recall  f1-score   support

           0       0.87      0.90      0.88      6040
           1       0.90      0.86      0.88      6026

    accuracy                           0.88     12066
   macro avg       0.88      0.88      0.88     12066
weighted avg       0.88      0.88      0.88     12066
```

```
polynomial kernel degree=3---------
[[4906 1134]
 [3330 2696]]
              precision    recall  f1-score   support

           0       0.60      0.81      0.69      6040
           1       0.70      0.45      0.55      6026

    accuracy                           0.63     12066
   macro avg       0.65      0.63      0.62     12066
weighted avg       0.65      0.63      0.62     12066

polynomial kernel degree=5---------
[[5447  593]
 [4779 1247]]
              precision    recall  f1-score   support

           0       0.53      0.90      0.67      6040
           1       0.68      0.21      0.32      6026

    accuracy                           0.55     12066
   macro avg       0.61      0.55      0.49     12066
weighted avg       0.61      0.55      0.49     12066

polynomial kernel degree=7---------
[[5800  240]
 [5547  479]]
              precision    recall  f1-score   support

           0       0.51      0.96      0.67      6040
           1       0.67      0.08      0.14      6026

    accuracy                           0.52     12066
   macro avg       0.59      0.52      0.40     12066
weighted avg       0.59      0.52      0.40     12066
```

Figure 4. Training with SVM

We can conclude from data shown above radial-basis-function (RBF) kernel is the best choice for classification and polynomial kernel is the worst one. The linear kernel is in the middle.
Because the precision of RBF is highest and RBF has the highest f1-score.
Precision of polynomial kernel is lowest and the f1-score is also very low compared with another two.
As for the polynomial kernel, the degree of 3 would be the best choice, because it has the highest f1-score compared with degree of 5 and degree of 7 and has highest accuracy rate.

Problem 5 The neural network classifier using ReLU (Bonus)

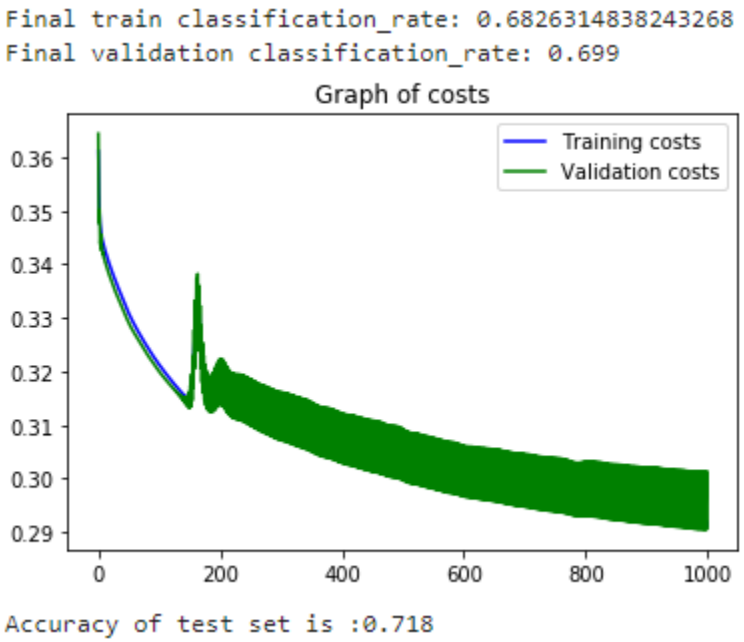

Figure 5. The neural network classifier using ReLU

Instead of using tanh for neural network, I choose to use ReLU as the activation function. The derivative of ReLU is:

$$\mathrm{ReLUDerivative} = \begin{cases} 0 & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x >= 0 \end{cases}$$

```
def reluDerivative(self,x):
    x[x<=0] = 0
    x[x>0] = 1
    return x
```

The function shown above which is in my code corresponds to the derivative of ReLU.

Problem 6 The neural network classifier using sigmoid (Bonus)

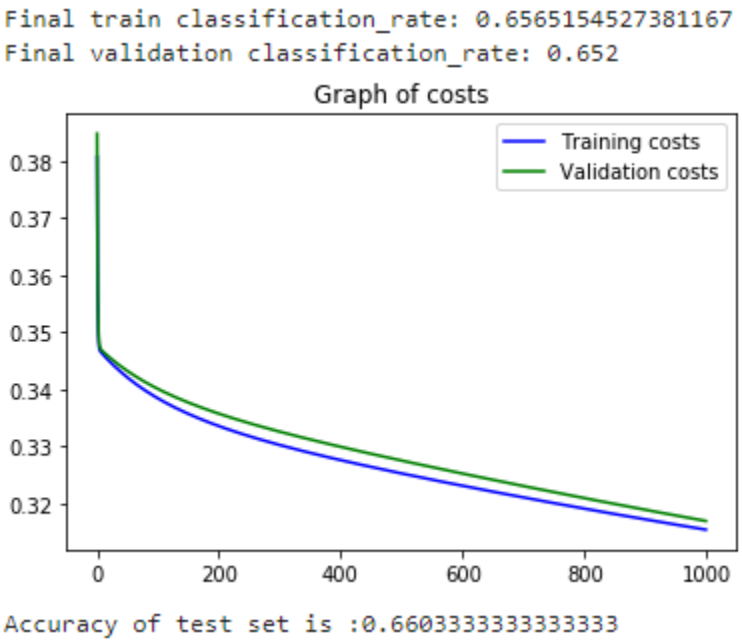

Figure 6. The neural network classifier using sigmoid

Instead of using ReLU for neural network, this time I choose to use sigmoid as the activation function. The derivative of sigmoid is:

$$sigmoidDerivative = \frac{dy}{dx} = (1+e^{-x})^{-1} - (1+e^{-x})^{-2} = y(1-y)$$

```
#(v) Then we propagate the errors we got from the previous layer W2 update W1 and b1
#ADD CODE HERE...
dZ=np.dot(Ytrain-Ytrain_ind,self.W2.T)*Ztrain*(1-Ztrain)
self.W1-=step_size*np.dot(X.T,dZ)
self.b1-=step_size*dZ.sum(axis=0)
```

The function Ztrain*(1-Ztrain) which is in my code corresponds to the derivative of sigmoid.