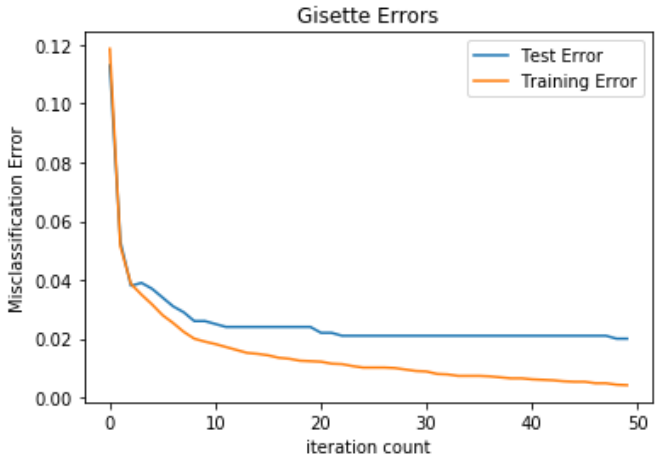

We perform Logistic Regression on four datasets: Gisette, Arcene, Madelon, Hill/Valley.
As always our code can be found at https://github.com/johansent/ML_Fall2017/tree/master/hw3

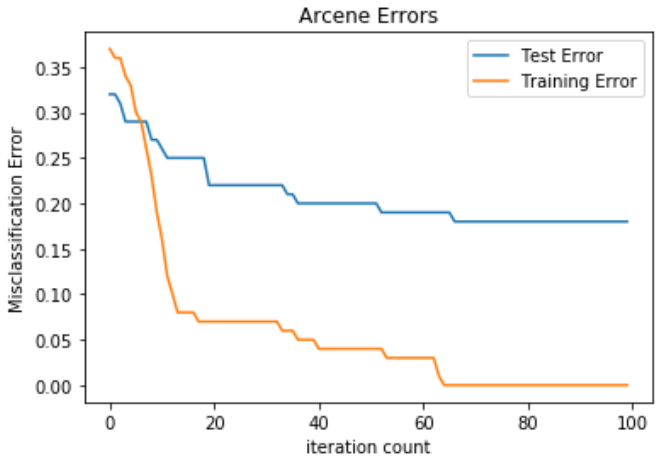
(Gissette Data)

For the Gissette Data set we found that a quick learning rate for this data was around .1. With this learning rate we converged to 98 percent accurate within about 20 iterations. With a learning rate of .001 we get a convergence somewhere after 300 iterations.



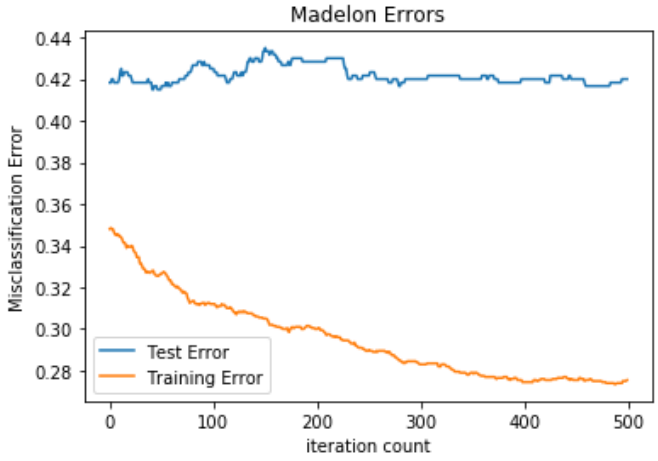
(Arcene)

For the Arcene Data set we did not need anywhere near even 300 iterations. We are able to quickly overfit the data regardless of learning rate and get 0 error on the training data, the weights that do this give us the best error of .2 on the data. This graph has a learning rate of .001.



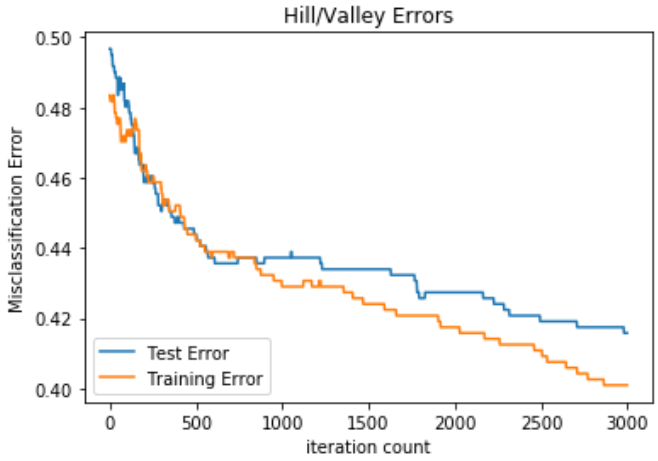
(Madelon)

This Data set just couldn't do very well. We can eventually get the training data down to about 70



(Hill/Valley)

This Dataset needs a slightly smaller learning rate than the other sets. A learning rate of .1 causes major oscilations. I found that a learning rate of .01 seemed to work best though we still didn't get much better than 60 percent accuracy.



(Results)

Data name	Test Error	Training Error
Gisette	0.020000	0.004167
Arcene	0.180000	0.000000
Madelon	0.415000	0.273500
Hill/Valley	0.415842	0.400990

(Most of our Code)

```
from import_data import import_data
from sklearn import preprocessing
import pandas
import matplotlib.pyplot as plt
import sklearn
import numpy as np

def updateWeights(X,X1,Y,w, Ncol,Nrow, learningRate, lam):
    tmp = w[1:Ncol]
    product = np.dot(X,tmp)
    shiftedValue = w[0] + product
    expValue = np.exp(shiftedValue)
    ratio = expValue / (1 + expValue)
    error = Y - ratio
    dLnew = np.dot(np.transpose(X1),error)
    w = w + learningRate * (-lam*w + dLnew/Nrow)
    return w

def Test(w, X, Y):
    #nomralize data
    X = preprocessing.scale(X)
    #Add column of 1s
    X = np.array(np.c_[np.ones((len(X), 1)), np.matrix(X)])
    Y = [0 if y <= 0 else 1 for y in np.array(Y)]
    wx = 1/(1+np.exp(-1 * np.dot(X, w)))
    Ypredict = [0 if x < .5 else 1 for x in wx]
    results = np.array(Y) - np.array(Ypredict)
    return sum(abs(results))/len(Y)

def Plot(x,y1,y2,title,legendLoc = 1):
    plt.title(title)
    plt.plot(x,y1,label = 'Test Error')
    plt.plot(x,y2,label = 'Training Error')
    plt.legend(loc = legendLoc)
    plt.xlabel('iteration count')
    plt.ylabel('Misclassification Error')
    plt.show()

def TrainWeights(X,Y,Xtest,Ytest,k, learnRate = .01):
    X = preprocessing.scale(X)
    Y = [0 if y <= 0 else 1 for y in np.array(Y)]
    X1 = np.array(np.c_[np.ones((len(X), 1)), np.matrix(X)])
    #initialize variables
    Nrow = len(X)
    Ncol = len(X1[0])

    learningRate = learnRate
    lam = .001

    w = np.array([0]*(Ncol))

    y1 = []
    y2 = []
    for i in range(k):
        w = updateWeights(X,X1,Y,w, Ncol,Nrow, learningRate, lam)
        y1.append(Test(w,Xtest, Ytest))
        y2.append(Test(w,X,Y))

    return y1,y2
```

```
### Script
min_table = {}

#Gisette Data

#Read in the Data
X, Y, Xtest, Ytest = import_data('gisette', 'gisette_train.data', 'gisette_valid.data','gisette_train.labe
X.drop(X.columns[len(X.columns)-1], axis=1, inplace=True)
Xtest.drop(Xtest.columns[len(Xtest.columns)-1], axis=1, inplace=True)  #conner.xyz  at https://stackoverfl

niter = 50
y1, y2 = TrainWeights(X,Y,Xtest,Ytest,niter,.1)

Plot(range(niter), y1, y2, 'Gisette Errors')
min_table['Gisette'] = [min(y1), min(y2)]
```

Bibliography

1. Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37
2. John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55
3. Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)
4. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)
5. conner.xyz at https://stackoverflow.com/questions/20517650/how-to-delete-the-last-column-of-data-of-a-pandas-dataframe