Applied Machine Learning
Homework 4

Daniel Gonzalez
Colton Piper

26th of September, 2018

1 Results

After speaking with Prof. Barbu, we decided to use w_0 in our computations, along with a column of 1s in our data matrix, since Prof. Barbu made the argument that ignoring w_0 removes one degree of freedom from our model and would give worse generalization for the classifier. However, the important thing to note is that w_0 is does not undergo thresholding via Θ, since we are not treating w_0 like a feature for selection. When comparing our results to results we'd previously computed ignoring w_0, our misclassification errors were slightly lower, but not low enough to be noticeable on a graph or to be worth tabulating.

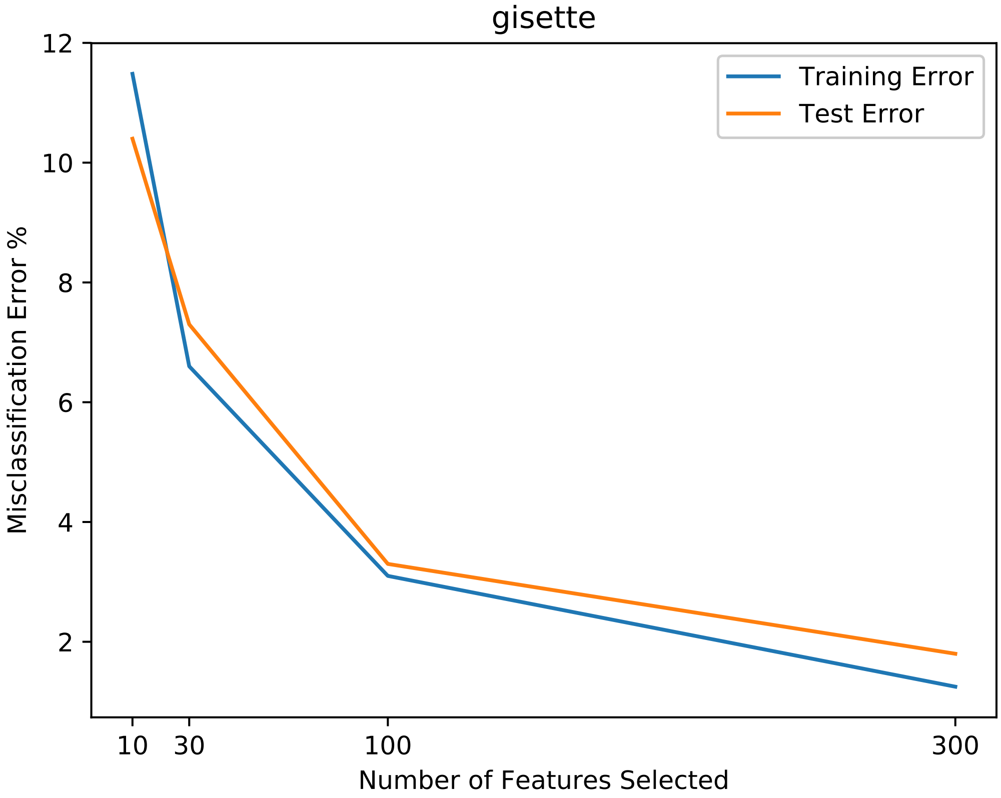
For this entire assignment, we used $\eta = {}^1\!/\!{}_N$ where N is the number of samples in the training data.

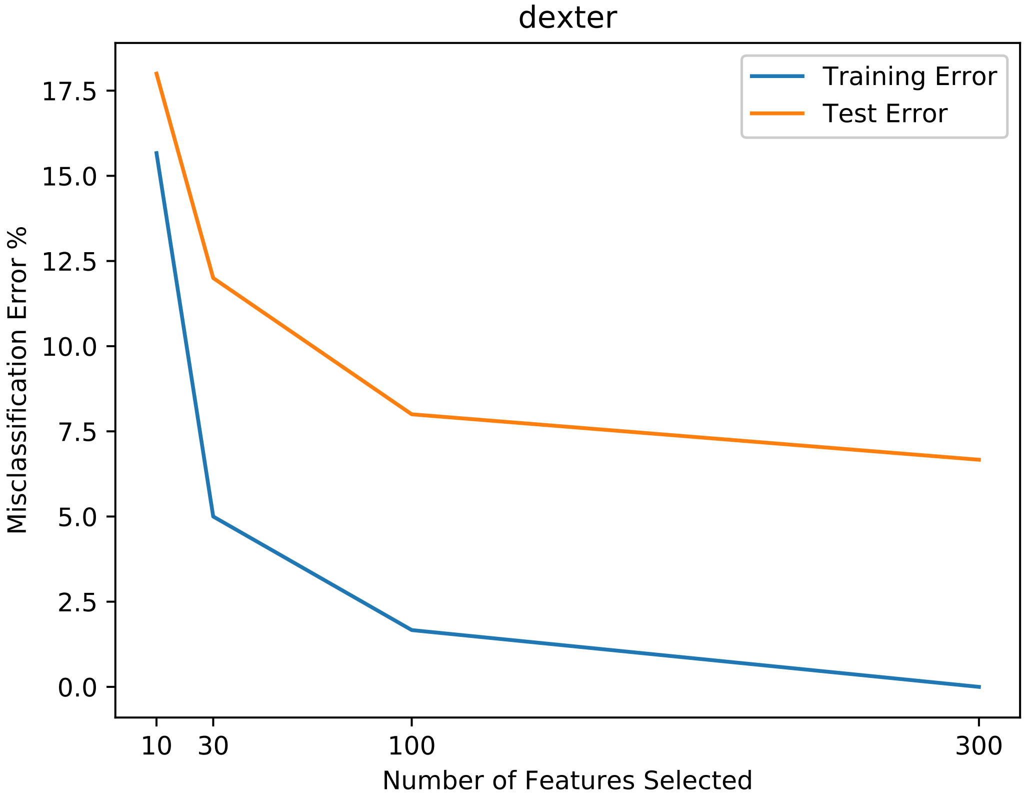
1.1 Table

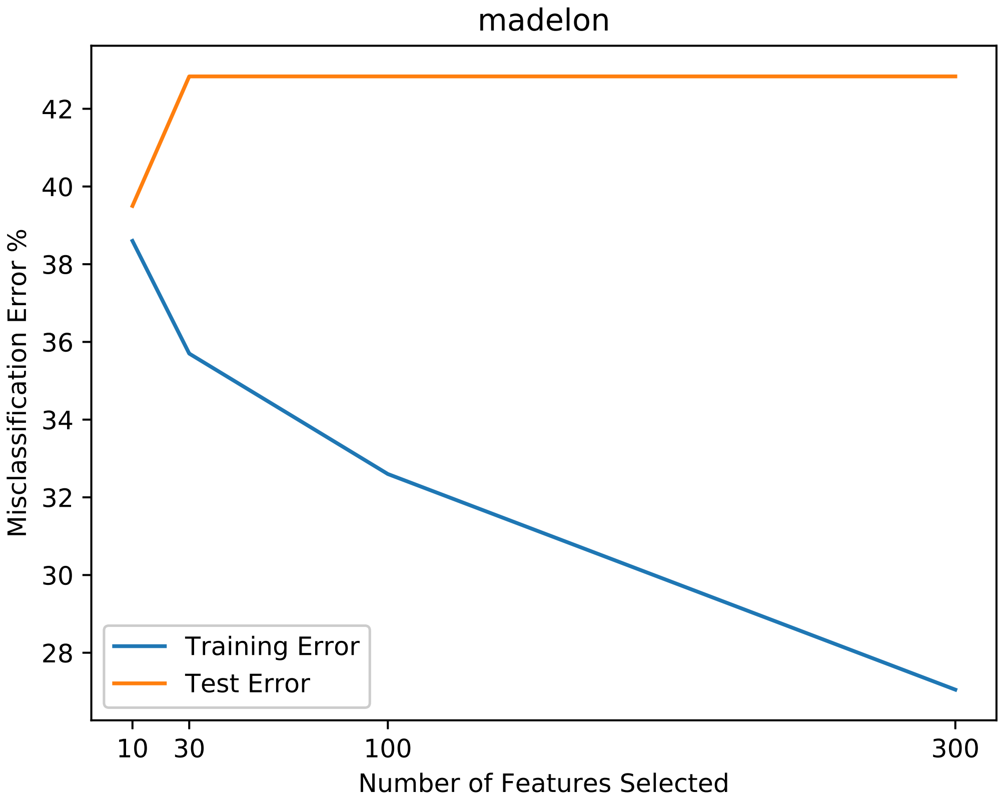
Table 1: Summary of Results

# Data Set	λ	Features Selected	Training Error	Test Error
gisette	0.197	10	11.4833 . . . %	10.4%
	0.134	30	6.6%	7.3%
	0.08745	100	3.1%	3.3%
	0.053	300	1.25%	1.8%
dexter	0.15	10	15.6666 . . . %	18.0%
	0.0983663	31	5.0%	12.0%
	0.0713	100	1.6666 . . . %	8.0%
	0.05269	293	0.0%	6.6666 . . . %
madelon	0.029799	8	38.6%	39.5%
	0.02435	30	35.7%	42.8333 . . . %
	0.01695	100	32.6%	42.8333 . . . %
	0.0074	299	27.05%	42.8333 . . . %

1.2 Figures

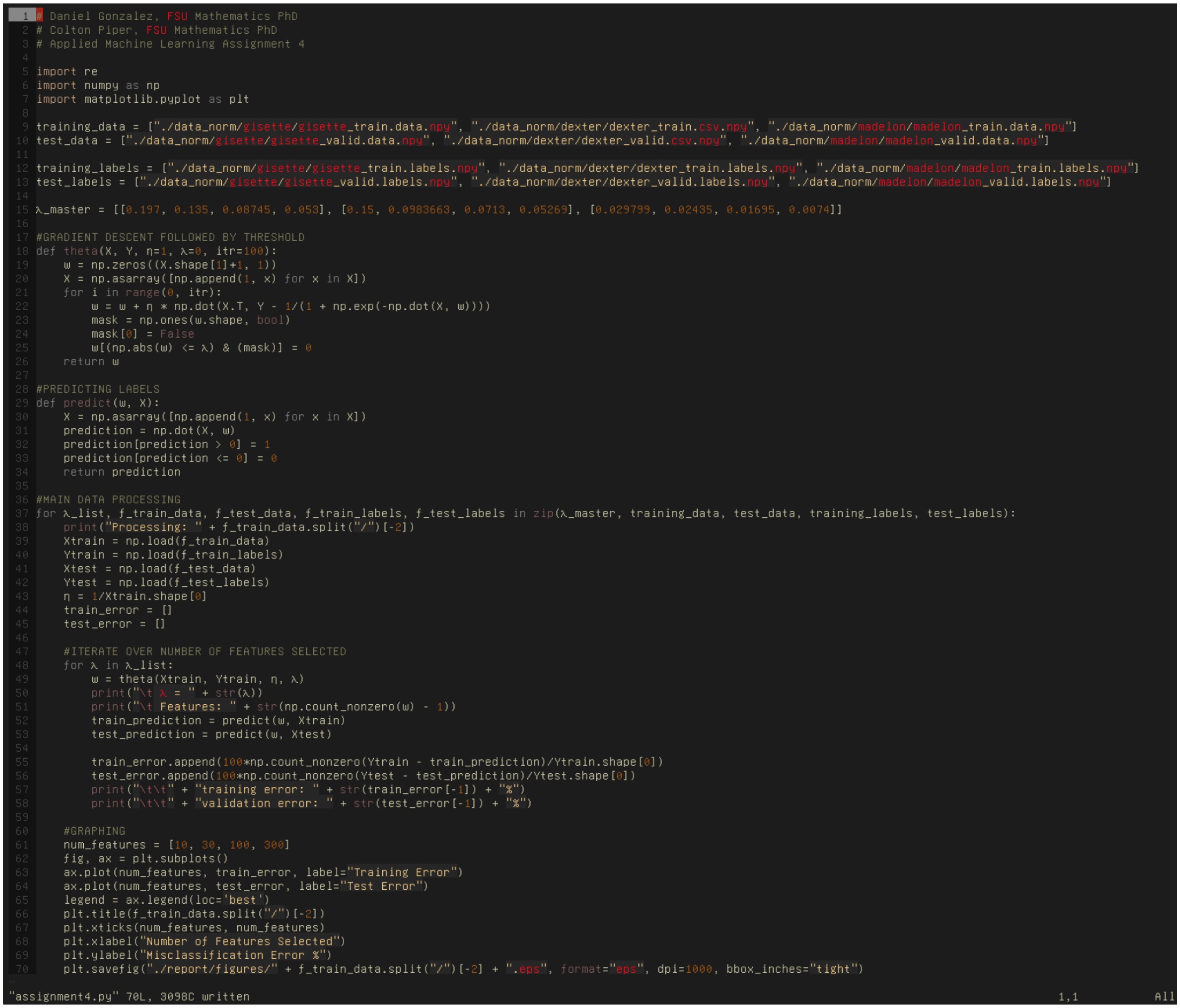

gisette
Training Error
Test Error
Misclassification Error %
12
10
8
6
4
2
10
30
100
300
Number of Features Selected


dexter
Training Error
Test Error
Misclassification Error %
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
10
30
100
300
Number of Features Selected


madelon
Misclassification Error %
42
40
38
36
34
32
30
28
Training Error
Test Error
10
30
100
300
Number of Features Selected

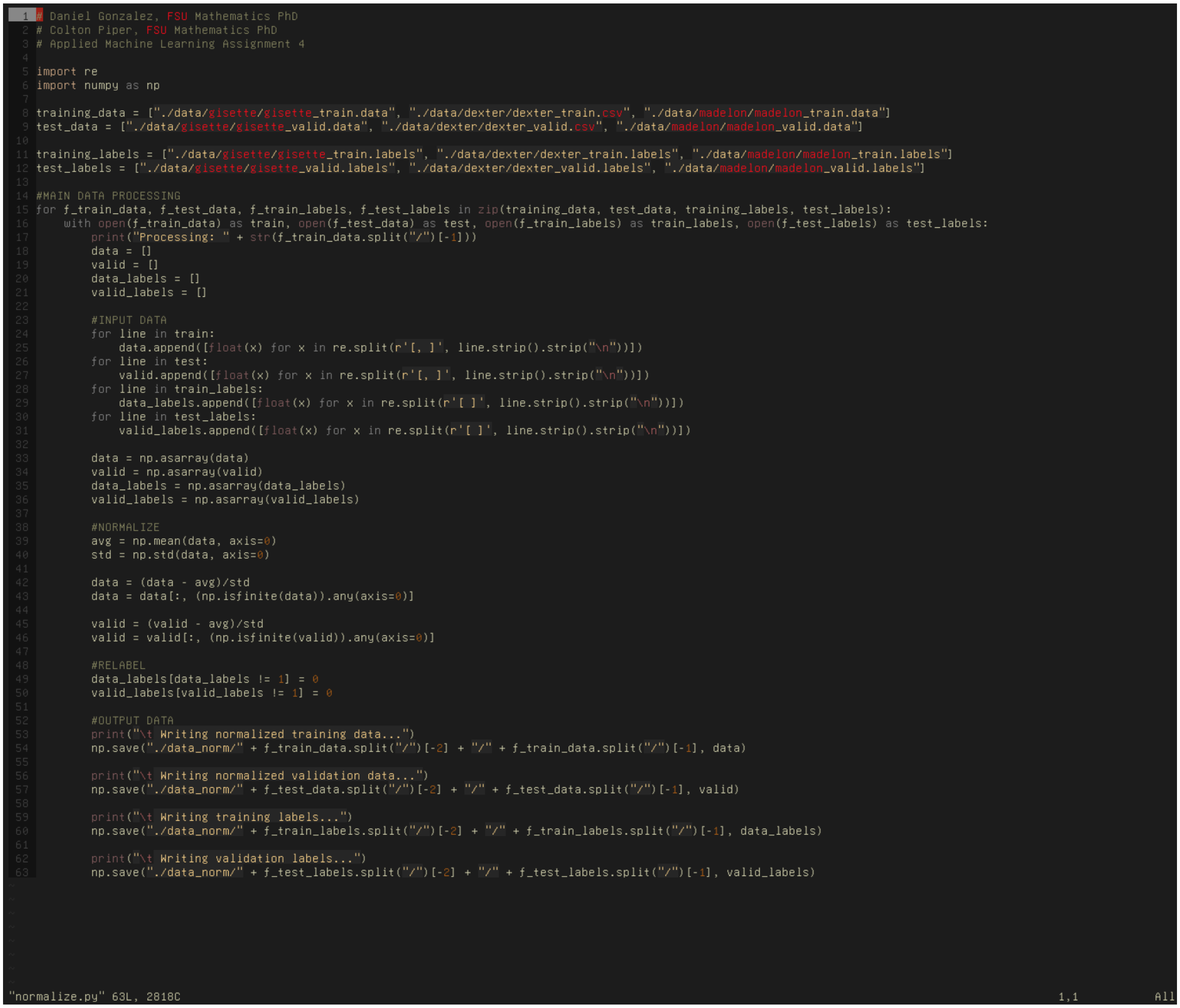
2 Appendix: Code

If the code looks too small, please zoom in on the pdf. The screenshots are `.png` images, so you should be able to zoom in and read at whatever is a comfortable size for you.



```python
# Daniel Gonzalez, FSU Mathematics PhD
# Colton Piper, FSU Mathematics PhD
# Applied Machine Learning Assignment 4

import re
import numpy as np
import matplotlib.pyplot as plt

training_data = ["./data_norm/gisette/gisette_train.data.npy", "./data_norm/dexter/dexter_train.csv.npy", "./data_norm/madelon/madelon_train.data.npy"]
test_data = ["./data_norm/gisette/gisette_valid.data.npy", "./data_norm/dexter/dexter_valid.csv.npy", "./data_norm/madelon/madelon_valid.data.npy"]

training_labels = ["./data_norm/gisette/gisette_train.labels.npy", "./data_norm/dexter/dexter_train.labels.npy", "./data_norm/madelon/madelon_train.labels.npy"]
test_labels = ["./data_norm/gisette/gisette_valid.labels.npy", "./data_norm/dexter/dexter_valid.labels.npy", "./data_norm/madelon/madelon_valid.labels.npy"]

λ_master = [[0.197, 0.135, 0.08745, 0.053], [0.15, 0.0983663, 0.0713, 0.05269], [0.029799, 0.02435, 0.01695, 0.0074]]

#GRADIENT DESCENT FOLLOWED BY THRESHOLD
def theta(X, Y, η=1, λ=0, itr=100):
    w = np.zeros((X.shape[1]+1, 1))
    X = np.asarray([np.append(1, x) for x in X])
    for i in range(0, itr):
        w = w + η * np.dot(X.T, Y - 1/(1 + np.exp(-np.dot(X, w))))
        mask = np.ones(w.shape, bool)
        mask[0] = False
        w[(np.abs(w) <= λ) & (mask)] = 0
    return w

#PREDICTING LABELS
def predict(w, X):
    X = np.asarray([np.append(1, x) for x in X])
    prediction = np.dot(X, w)
    prediction[prediction > 0] = 1
    prediction[prediction <= 0] = 0
    return prediction

#MAIN DATA PROCESSING
for λ_list, f_train_data, f_test_data, f_train_labels, f_test_labels in zip(λ_master, training_data, test_data, training_labels, test_labels):
    print("Processing: " + f_train_data.split("/")[-2])
    Xtrain = np.load(f_train_data)
    Ytrain = np.load(f_train_labels)
    Xtest = np.load(f_test_data)
    Ytest = np.load(f_test_labels)
    η = 1/Xtrain.shape[0]
    train_error = []
    test_error = []

    #ITERATE OVER NUMBER OF FEATURES SELECTED
    for λ in λ_list:
        w = theta(Xtrain, Ytrain, η, λ)
        print("\t λ = " + str(λ))
        print("\t Features: " + str(np.count_nonzero(w) - 1))
        train_prediction = predict(w, Xtrain)
        test_prediction = predict(w, Xtest)

        train_error.append(100*np.count_nonzero(Ytrain - train_prediction)/Ytrain.shape[0])
        test_error.append(100*np.count_nonzero(Ytest - test_prediction)/Ytest.shape[0])
        print("\t\t" + "training error: " + str(train_error[-1]) + "%")
        print("\t\t" + "validation error: " + str(test_error[-1]) + "%")

    #GRAPHING
    num_features = [10, 30, 100, 300]
    fig, ax = plt.subplots()
    ax.plot(num_features, train_error, label="Training Error")
    ax.plot(num_features, test_error, label="Test Error")
    legend = ax.legend(loc='best')
    plt.title(f_train_data.split("/")[-2])
    plt.xticks(num_features, num_features)
    plt.xlabel("Number of Features Selected")
    plt.ylabel("Misclassification Error %")
    plt.savefig("./report/figures/" + f_train_data.split("/")[-2] + ".eps", format="eps", dpi=1000, bbox_inches="tight")
```

```
"assignment4.py" 70L, 3098C written                                                    1,1           All
```



```python
# Daniel Gonzalez, FSU Mathematics PhD
# Colton Piper, FSU Mathematics PhD
# Applied Machine Learning Assignment 4

import re
import numpy as np

training_data = ["./data/gisette/gisette_train.data", "./data/dexter/dexter_train.csv", "./data/madelon/madelon_train.data"]
test_data = ["./data/gisette/gisette_valid.data", "./data/dexter/dexter_valid.csv", "./data/madelon/madelon_valid.data"]

training_labels = ["./data/gisette/gisette_train.labels", "./data/dexter/dexter_train.labels", "./data/madelon/madelon_train.labels"]
test_labels = ["./data/gisette/gisette_valid.labels", "./data/dexter/dexter_valid.labels", "./data/madelon/madelon_valid.labels"]

#MAIN DATA PROCESSING
for f_train_data, f_test_data, f_train_labels, f_test_labels in zip(training_data, test_data, training_labels, test_labels):
    with open(f_train_data) as train, open(f_test_data) as test, open(f_train_labels) as train_labels, open(f_test_labels) as test_labels:
        print("Processing: " + str(f_train_data.split("/")[-1]))
        data = []
        valid = []
        data_labels = []
        valid_labels = []

        #INPUT DATA
        for line in train:
            data.append([float(x) for x in re.split(r'[, ]', line.strip().strip("\n"))])
        for line in test:
            valid.append([float(x) for x in re.split(r'[, ]', line.strip().strip("\n"))])
        for line in train_labels:
            data_labels.append([float(x) for x in re.split(r'[ ]', line.strip().strip("\n"))])
        for line in test_labels:
            valid_labels.append([float(x) for x in re.split(r'[ ]', line.strip().strip("\n"))])

        data = np.asarray(data)
        valid = np.asarray(valid)
        data_labels = np.asarray(data_labels)
        valid_labels = np.asarray(valid_labels)

        #NORMALIZE
        avg = np.mean(data, axis=0)
        std = np.std(data, axis=0)

        data = (data - avg)/std
        data = data[:, (np.isfinite(data)).any(axis=0)]

        valid = (valid - avg)/std
        valid = valid[:, (np.isfinite(valid)).any(axis=0)]

        #RELABEL
        data_labels[data_labels != 1] = 0
        valid_labels[valid_labels != 1] = 0

        #OUTPUT DATA
        print("\t Writing normalized training data...")
        np.save("./data_norm/" + f_train_data.split("/")[-2] + "/" + f_train_data.split("/")[-1], data)

        print("\t Writing normalized validation data...")
        np.save("./data_norm/" + f_test_data.split("/")[-2] + "/" + f_test_data.split("/")[-1], valid)

        print("\t Writing training labels...")
        np.save("./data_norm/" + f_train_labels.split("/")[-2] + "/" + f_train_labels.split("/")[-1], data_labels)

        print("\t Writing validation labels...")
        np.save("./data_norm/" + f_test_labels.split("/")[-2] + "/" + f_test_labels.split("/")[-1], valid_labels)
```

```
"normalize.py" 63L, 2818C                                                      1,1           All
```