Applied Machine Learning
Homework 3

Daniel Gonzalez
Colton Piper
19th of September, 2018

1 Results

1.1 Table

We would first like to apologize for the lateness of our submission. We initially implemented the gradient descent using bare Python list comprehensions and for-loop iteration, which turned out to be unacceptably slow. We spent an inordinate amount of time trying to refactor that code before discovering NumPy, after which we completely rewrote our code to take advantage of NumPy's array/vector parallelization.

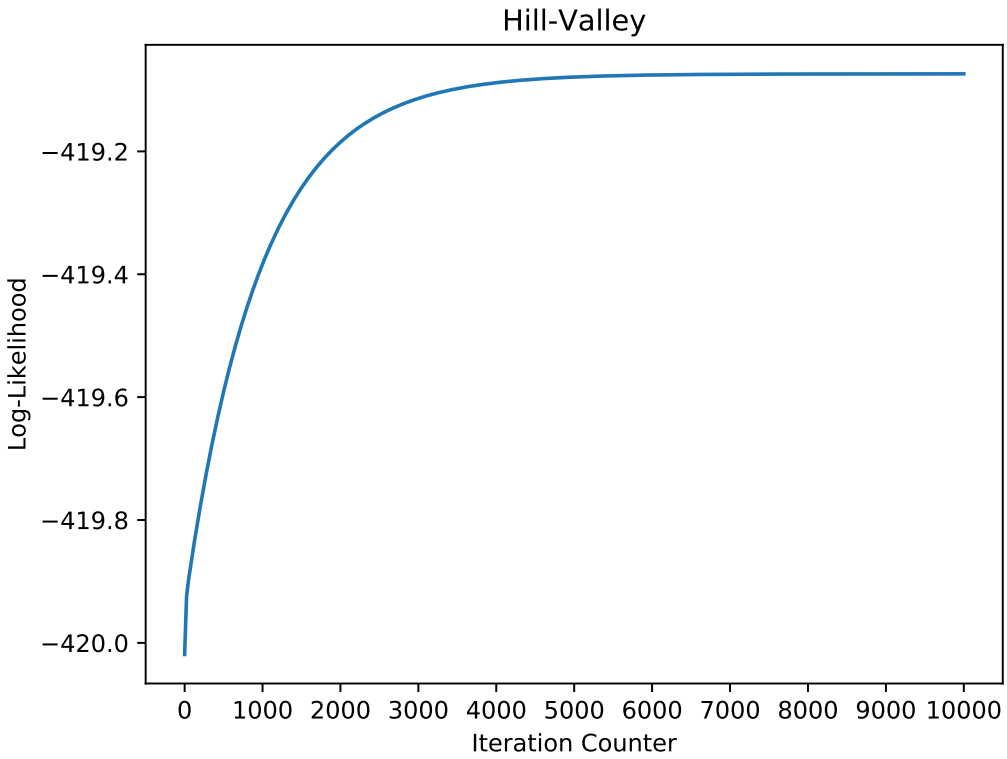
Now that we were able to quickly produce data, we tried a few different learning rates (starting with $\eta = 0.1$) before settling on a learning rate of $\eta = 1$, which produced convergent behaviour for the `Hill-Valley` data set in around 5000 iterations and in the `Gisette` data set in 500 iterations. With this learning rate, we observed convergence on the `Madelon` data set in around 4000 iterations. In the table below, we've compiled training and validation errors for classifiers trained on using varying numbers of iterations (some of the lower iteration values are hold-overs from when the code ran more inefficiently).

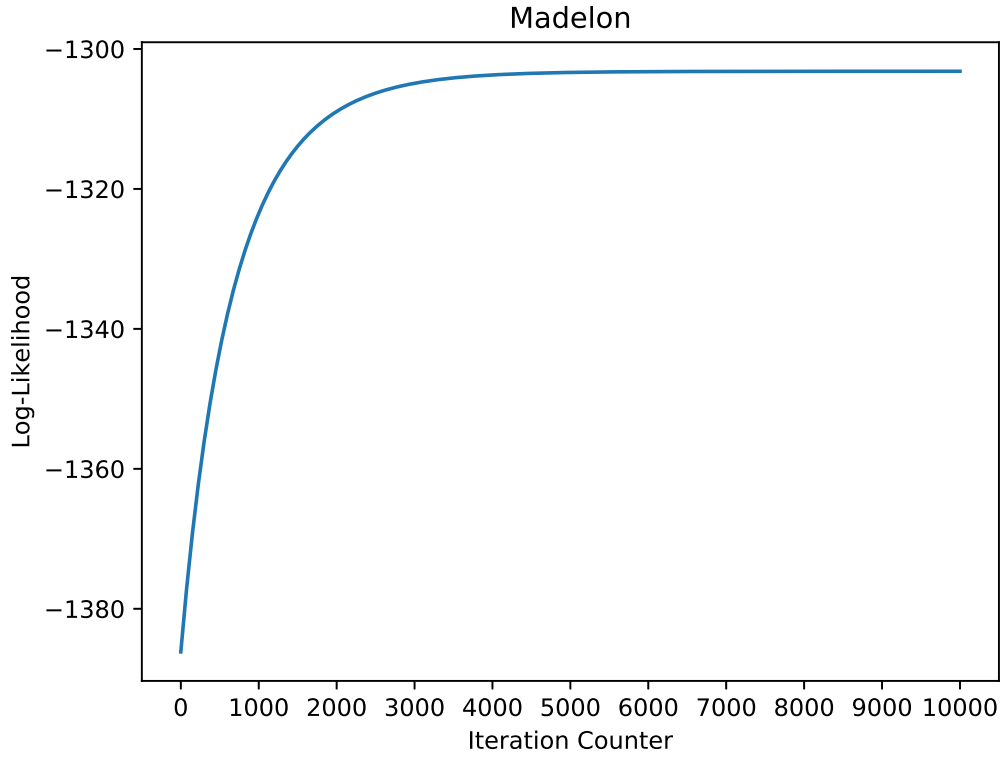
Note the increase in the test error for `Madelon` with 10000 iterations compared to 500 iterations despite the decrease in training error. This may be indicative of overfitting and could indicate that a classifier should be trained on less iterations than 10000 for this data set.

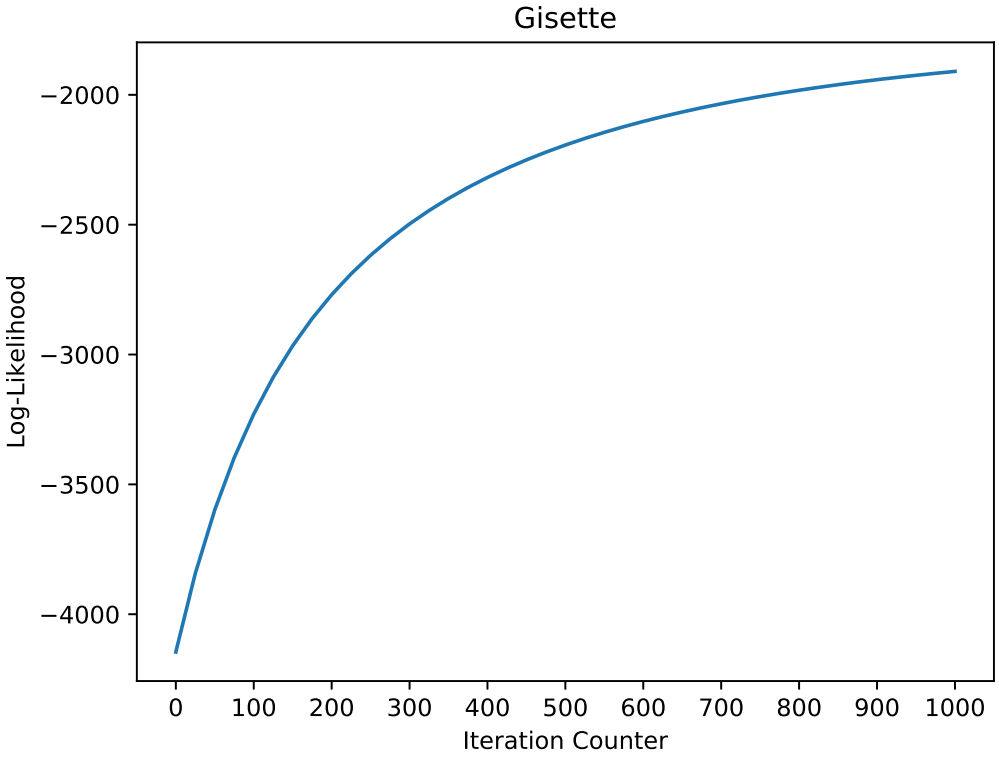
Table 1: Test Misclassification Errors

# Data Set	# Iterations	Training Error	Test Error
hill-valley	300	48.84%	49.83%
	10000	46.86%	47.03%
madelon	500	34.80%	41.50%
	10000	31.2%	42.33%
gisette	3	11.87%	11.00%
	15	11.85%	10.90%
	20	11.83%	10.90%
	1000	5.28%	5.6%

1.2 Figures


Hill-Valley
Log-Likelihood
−419.2
−419.4
−419.6
−419.8
−420.0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Iteration Counter


Madelon
Log-Likelihood
−1300
−1320
−1340
−1360
−1380
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Iteration Counter


Gisette
−2000
−2500
−3000
−3500
−4000
Log-Likelihood
0
100
200
300
400
500
600
700
800
900
1000
Iteration Counter

2 Appendix: Code

```
# Daniel Gonzalez, FSU Mathematics PhD
# Colton Piper, FSU Mathematics PhD
# Applied Machine Learning Assignment 3

from math import exp
from math import log
import numpy as np
import matplotlib.pyplot as plt

training_data = ["./data_norm/gisette/gisette_train.data"]
test_data = ["./data_norm/gisette/gisette_valid.data"]

training_labels = ["./data_norm/gisette/gisette_train.labels"]
test_labels = ["./data_norm/gisette/gisette_valid.labels"]

def gradient(X, Y, weights):
    N = X.shape[0]
    M = X[0].shape[0]

    grad = np.zeros(M+1)
    x = np.asarray([ np.append([1], x) for x in X])
    xT = np.transpose(x)
    W = weights.reshape(1, M+1)
    fraction = np.exp(np.inner(W, x)).transpose()
    fraction = (fraction / (1 + fraction))
    grad = np.dot(xT, Y - fraction)

    return grad

def gradient_ascent(X, Y, η=1, λ=0.001, num_itr=300):
    llh = np.array([])
    N = X.shape[0]
    M = X[0].shape[0]
    weights = np.zeros(M + 1)

    for itr in range(0, num_itr+1):
        print(itr)
        grad = gradient(X, Y, weights)[:,0]
        weights = ((1 - η * λ) * weights) + ((η / N) * grad)
        print(weights.shape)
        if itr % 25 == 0:
            llh = np.append(llh, log_likelihood(X, Y, weights))

    return (weights, llh)

def log_likelihood(X, Y, weights):
    llh = 0
    N = X.shape[0]
    M = X[0].shape[0]

    for j in range(0, N):
        x = np.append([1], X[j])
        product = np.inner(x, weights)
        llh = llh + Y[j, 0] * product - np.log(1 + np.exp(product))

    return np.array(llh)

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_data, open(f_test_data) as test_data, open(f_train_labels) as train_labels, open(f_test_labels) as test_labels:
        output_train_error = open("./output/train_error_" + f_train_data.split("/")[-2], "w")
        output_test_error = open("./output/test_error_" + f_train_data.split("/")[-2], "w")

        train_error = 0
        test_error = 0
        llh = np.array([])
        train_predictions = np.array([])
        test_predictions = np.array([])

        print("Processing: " + f_train_data.split("/")[-1])
        Xtrain = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in train_data])
        Ytrain = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in train_labels])
"assignment3.py" 111L, 4089C written                                                                                1,
```

```
        if itr % 25 == 0:
            llh = np.append(llh, log_likelihood(X, Y, weights))

    return (weights, llh)

def log_likelihood(X, Y, weights):
    llh = 0
    N = X.shape[0]
    M = X[0].shape[0]

    for j in range(0, N):
        x = np.append([1], X[j])
        product = np.inner(x, weights)
        llh = llh + Y[j, 0] * product - np.log(1 + np.exp(product))

    return np.array(llh)

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_data, open(f_test_data) as test_data, open(f_train_labels) as train_labels, open(f_test_labels) as test_labels:
        output_train_error = open("./output/train_error_" + f_train_data.split("/")[-2], "w")
        output_test_error = open("./output/test_error_" + f_train_data.split("/")[-2], "w")

        train_error = 0
        test_error = 0
        llh = np.array([])
        train_predictions = np.array([])
        test_predictions = np.array([])

        print("Processing: " + f_train_data.split("/")[-1])
        Xtrain = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in train_data])
        Ytrain = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in train_labels])
        Xtest = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in test_data])
        Ytest = np.asarray([[float(x) for x in line.strip().strip("\n").split(" ")] for line in test_labels])

        results = gradient_ascent(Xtrain, Ytrain)
        weights = results[0]
        llh = results[1]

        for x in Xtrain:
            if np.inner(np.append([1], x), weights) > 0:
                train_predictions = np.append(train_predictions, [1])
            else:
                train_predictions = np.append(train_predictions, [0])

        for x in Xtest:
            if np.inner(np.append([1], x), weights) > 0:
                test_predictions = np.append(test_predictions, [1])
            else:
                test_predictions = np.append(test_predictions, [0])

        for y, p in zip(Ytrain, train_predictions):
            if y != p:
                train_error = train_error + 1

        for y, p in zip(Ytest, test_predictions):
            if y != p:
                test_error = test_error + 1

        print(llh)

        output_train_error.write(str(train_error/(Ytrain.shape[0])) + "\n")
        output_test_error.write(str(test_error/(Ytest.shape[0])) + "\n")

        axes = [x for x in range(0, 301) if (x % 25 == 0)]
        fig, ax = plt.subplots()
        ax.plot(axes, llh)
        plt.title("Hill-Valley")
        plt.xticks([x for x in range(0, 301) if (x%25 == 0)], [x for x in range(0, 301) if (x%25 == 0)])
        plt.xlabel("Iteration Counter")
        plt.ylabel("Log-Likelihood")
        plt.savefig("./report/figures/hill-valley.eps", format="eps", dpi=1000, bbox_inches="tight")
```