HW7_edgarsp2

November 25, 2019

1 STAT 542 / CS 598: Homework 7

Fall 2019, by Edgar Pino (edgarsp2)
Due: Monday, Nov 25 by 11:59 PM Pacific Time

[62]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
```

2 Question 1 [100 Points] AdaBoost with stump model

2.1 Generate Data

[63]:
```
def plot_data(x, y):
    unique = np.unique(y)
    y_plot = y + 0.1*np.random.uniform(-1, 1, len(x))
    for li in range(len(unique)):
        _x = x[y == unique[li]]
        _y = y_plot[y == unique[li]]
        label = 'Positive' if unique[li] == 1 else 'Negative'
        plt.scatter(_x, _y, c = COLOR_LABELS[li], label=label, s=8)

#     lines(sort(x), py(x)[order(x)] - 0.5)
#     plt.plot(x, py(y)-0.5, c='blue')
    plt.legend()
    plt.show()
```

[64]:
```
np.random.seed(1)
```

[65]:
```
COLOR_LABELS = ["deepskyblue","darkorange"]
```

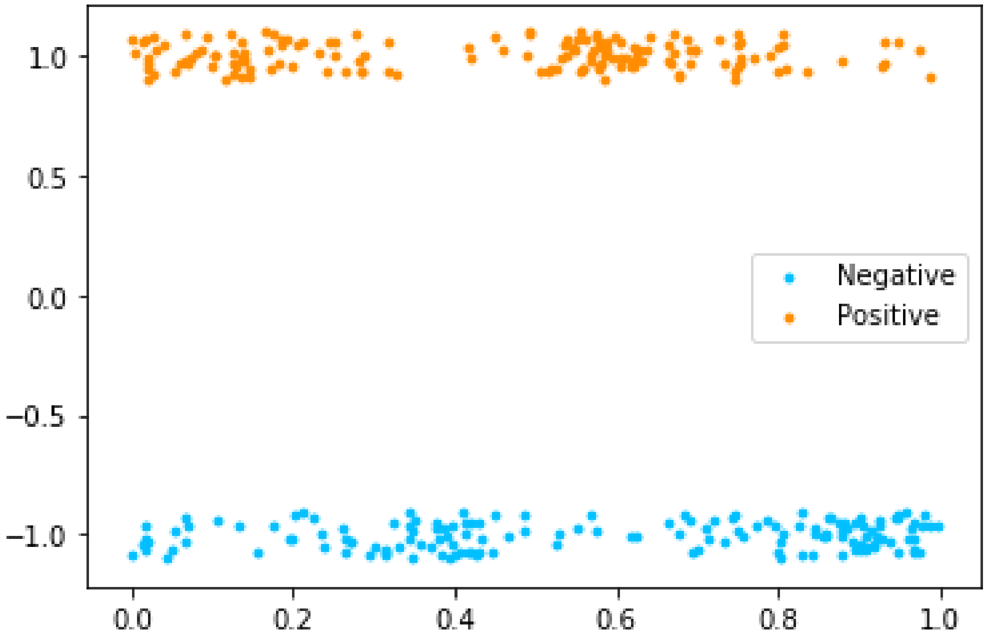
[66]:
```
n = 300
```

[67]:
```
x = np.random.uniform(size=n)
```

[68]:
```
py = lambda x: np.sin(4*np.pi*x)/3 + 0.5
```

[69]:
```
y = (np.random.binomial(1, py(x), n)-0.5)*2
```

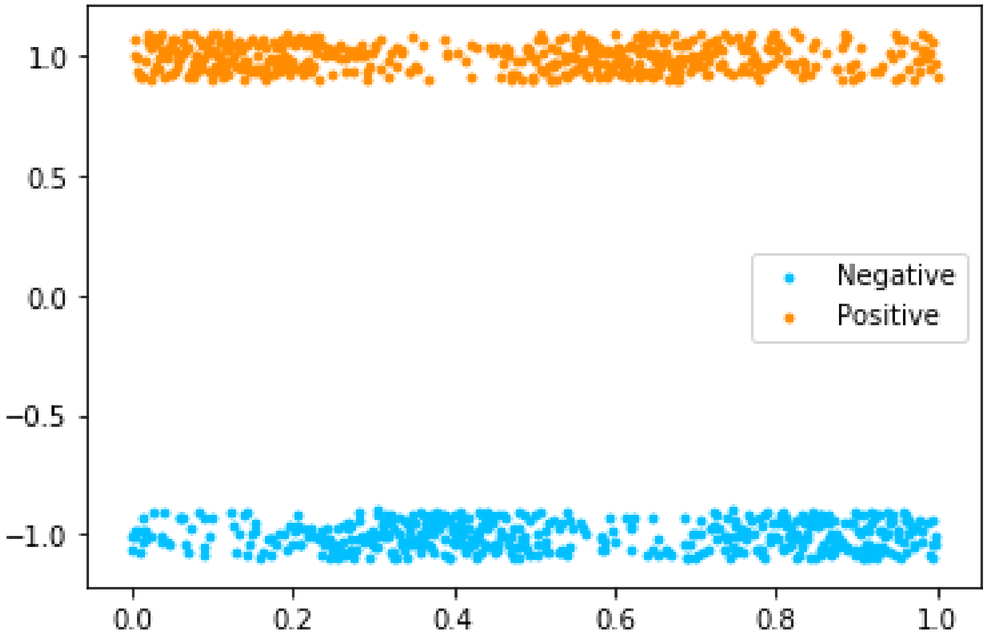
[70]:
```
plot_data(x,y)
```



[71]:
```
x_test = np.linspace(0, 1, 1000)
```

[72]:
```
y_test = (np.random.binomial(1, py(x_test), 1000)-0.5)*2
```

[73]:
```
plot_data(x_test,y_test)
```



2.2 Create model

[74]:

```
def exponentialLoss(y, ypred):
    return np.mean(np.exp(-(2. * y - 1.) * ypred))
```

[75]:

```
class Stump():
    def __init__(self):
        self.label = 1
        self.threshold = None
        self.loss = None
        self.alpha = None
```

[76]:

```
class MyAdaboost():
    def __init__(self,shrinkage=.5, num_classifiers=10):
        self.num_classifiers = num_classifiers
        self.shrinkage = shrinkage
        self.weights = None

    def fit(self,x,y):
        n_samples = len(x)
        weights = np.full(n_samples, (1 / n_samples))

        self.classifiers = []

        for _ in range(self.num_classifiers):
            classifier = Stump()
            min_error = float('inf')

            feature_values = np.expand_dims(x, axis=1)
            unique_values = np.unique(feature_values)

            for threshold in unique_values:
                p = 1
                prediction = np.ones(np.shape(y))
                prediction[x < threshold] = -1
                error = sum(weights[y != prediction])

                if error > 0.5:
                    error = 1 - error
                    p = -1

                if error < min_error:
                    classifier.label = p
                    classifier.threshold = threshold
                    min_error = error
                    classifier.loss = exponentialLoss(y, prediction)
            classifier.alpha = self.shrinkage * np.log((1.0 - min_error) / (min_error + 1e-10))
            predictions = np.ones(np.shape(y))
            negative_idx = (classifier.label * x < classifier.label * classifier.threshold)
            predictions[negative_idx] = -1
            weights *= np.exp(-classifier.alpha * y * predictions)
            weights /= np.sum(weights)

            self.classifiers.append(classifier)

        self.weights = weights

    def predict(self, x):
        n_samples = len(x)
        y_pred = np.zeros((n_samples, 1))

        for classifier in self.classifiers:
            predictions = np.ones(np.shape(y_pred))
            negative_idx = (classifier.label * x < classifier.label * classifier.threshold)
            predictions[negative_idx] = -1

            y_pred += classifier.alpha * predictions
        return np.sign(y_pred).flatten()
```

[77]:

```
number_of_classifiers = 150
```

[78]:

```
model = MyAdaboost(shrinkage=.5, num_classifiers=number_of_classifiers)
```

[79]:

```
model.fit(x, y)
```

[80]:

```
y_pred = model.predict(x_test)
```

[81]:

```
f"Model accuracy {accuracy_score(y_test, y_pred) * 100}%"
```

[81]:

```
'Model accuracy 67.30000000000001%'
```

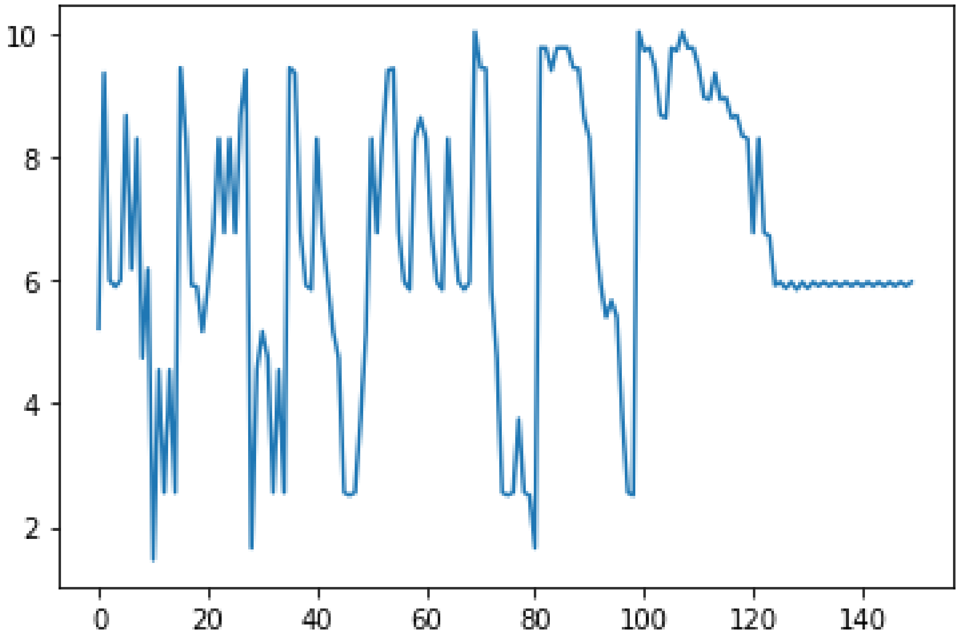
2.3 Plot the exponential loss

[82]:

```
loss_values = [clf.loss for clf in model.classifiers]
```

[83]:

```
plt.plot(loss_values)
plt.show()
```



It appears that the best number of classifiers is about 120. After that, the model loss seems to stay the same.

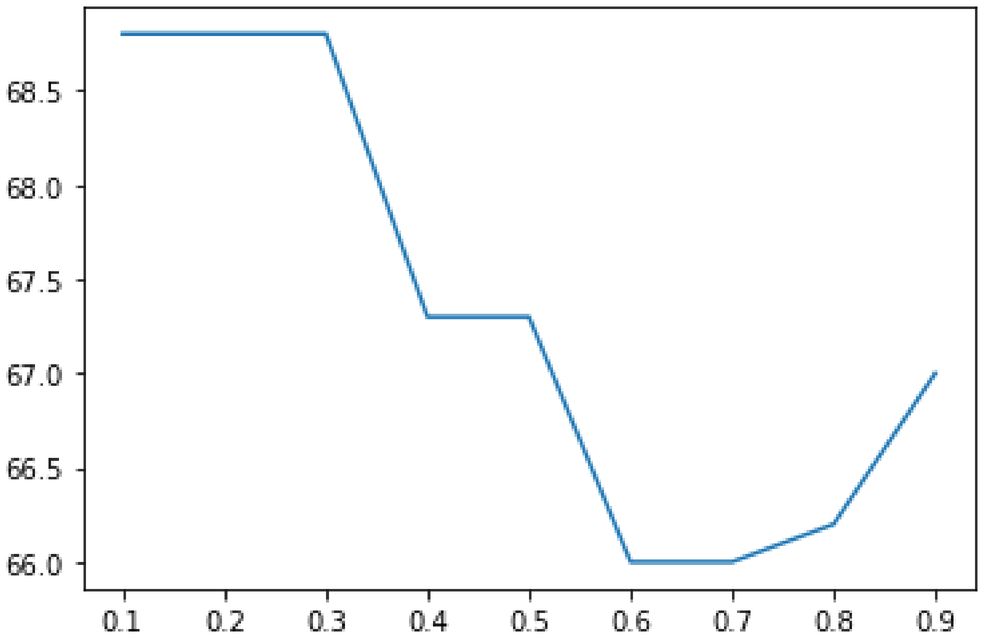
2.4 Try a few different shrinkage factors and comment on your findings

[84]:
```
shrinkage_values = np.arange(.1,1,.1)
```

[85]:
```
scores = []
```

[86]:
```
for val in shrinkage_values:
    s_model = MyAdaboost(shrinkage=val, num_classifiers=number_of_classifiers)
    s_model.fit(x, y)
    y_pred = s_model.predict(x_test)
    score = accuracy_score(y_test, y_pred) * 100
    scores.append(score)
```

[87]:
```
plt.plot(shrinkage_values, scores)
plt.show()
```



Based on this chart, the best shrinkage value is between .1 and .3. After that, the model accuracy score drops.

2.5 Plot the final model

[88]:
```
best_num_classifiers=120
```

[89]:
```
best_shrinkage=.1
```

[90]:
```
final_model = MyAdaboost(shrinkage=best_shrinkage,␣
 ↪num_classifiers=best_num_classifiers)
```

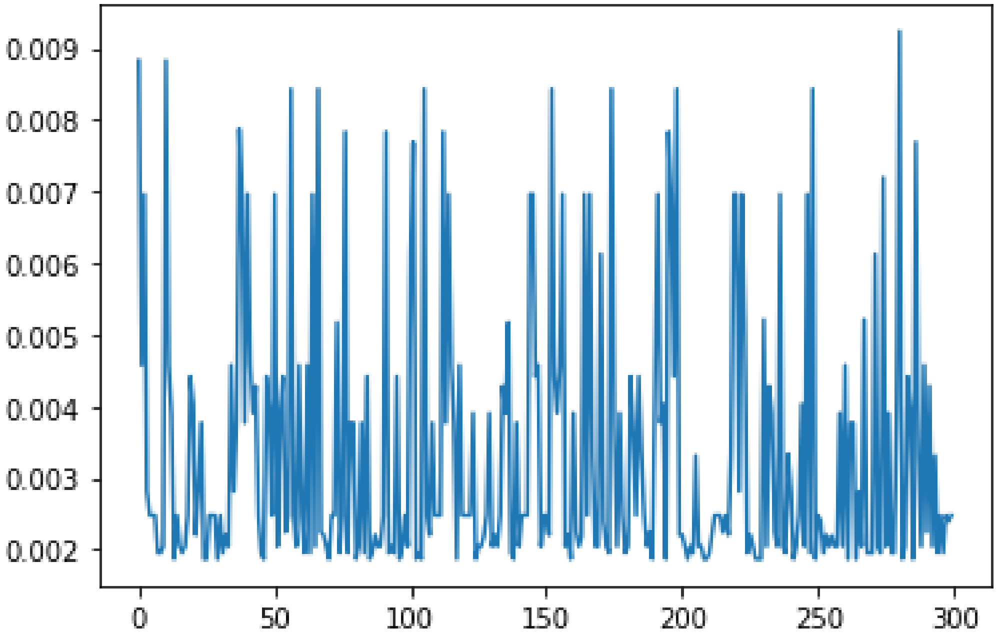
[91]:
```
final_model.fit(x, y)
```

[92]:
```
y_pred = final_model.predict(x_test)
```

[93]:
```
f"Final Model accuracy {accuracy_score(y_test, y_pred) * 100}"
```

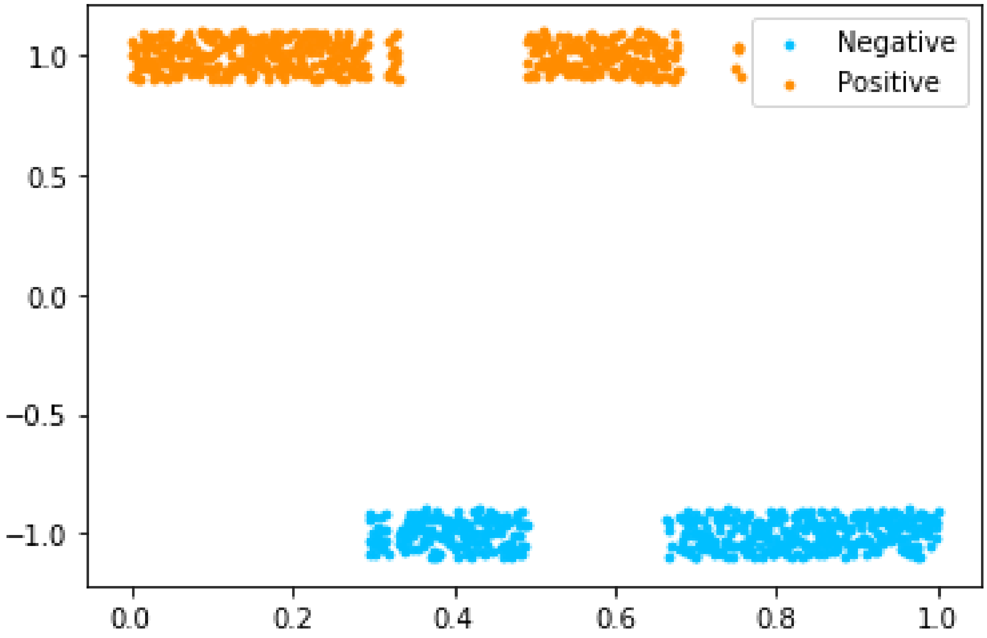
[93]:
```
'Final Model accuracy 68.8'
```

[94]:
```
plt.plot(final_model.weights)
plt.show()
```



[ ]:
```
plt.plot(final_model.weights)
plt.show()
```

[61]:
```
plot_data(x_test,y_pred)
```



[ ]: