CSci 5525 Homework 4

Luis Guzman

Thursday December 10, 2020

1. For this problem, I implemented the Adaboost algorithm using 100 decision stumps as my weak learners and applied it to the Wisconsin Breast Cancer dataset. This particular dataset has some (16 total) missing values, so I chose to replace those instances with the mode of that feature's data. The decision stumps were implemented from scratch using binary splitting on the feature that maximizes information gain:

$$IG(S,a) = H(S) - \sum_{v \in vals(a)} \frac{|S_v|}{|S|} H(S_v)$$

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

where IG is the information gain and H is the information entropy. The Adaboost algorithm starts by initializing the weight vector $w_1(i)$ to $1/n$ where n is the number of data samples. At each iteration, I trained a new decision stump G_t, and computed that weak learner's error ϵ_t as

$$\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbb{1}(y_i \neq G_t(x_i))$$

Each weak learner then gets a weight assiciated with it, which indicates how well it classifies the dataset:

$$\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$$

I then update the sample weights so that the next classifier puts more emphasis on the misclassified data

$$w_{t+1} = \frac{w_t(i) \exp(-\alpha_t y_i G_t(x_i))}{z_i}$$

where $z_i = \sum_{i=1}^{n} w_{t+1}(i)$ is the normalization factor. Rather than modifying the information gain equations to account for the Adaboost weights, I chose to resample the dataset with replacement at each iteration, using the weights as the sampling probability distribution.

```
Xtrain = X[np.random.choice(X.shape[0], X.shape[0], replace=True, p=w)]
```

This resampling method ensures that subsequent weak learners respond more strongly to misclassified datapoints. The final prediction of the ensemble of weak learners is then the weighted average of their outputs

$$g(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t G_t(x))$$

I evaluated my model by computing the test error at each step as I add more weak learners to the ensemble. My first weak learner classified the test dataset with around 9% accuracy, and this value decreased to around 4% after adding 10 weak learners. As I added more weak learners, the error actually increased because the later learners were not contributing useful data to the ensemble. My error rates are shown in figure 1

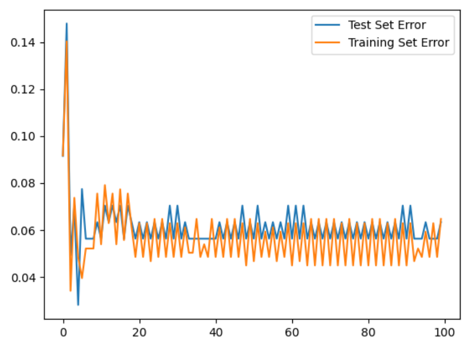

Figure 1: The testing error rate plotted against the number of weak learners in the ensemble

2. In the next problem, I implemented the Random Forest algorithm on the same dataset as above. I again used information gain to determine the best feature to split on, but to increase the randomness of my random forest, I limited each decision stump to consider only m random features of the dataset. In part (i), I chose m=3 and plotted the testing error vs. the number of weak learner in the forest. In part (ii), I vary m from 1 to 9 (I only had 9 features since I threw away the patient ID) and plotted the testing error for each value of m. My results are given below:

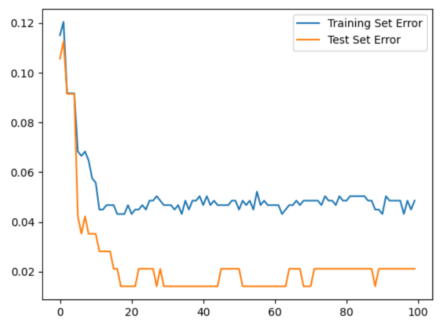

Figure 2: The testing and training error rates plotted against the number of weak learners in the random forest

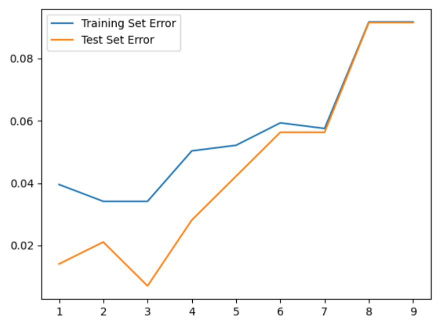

Figure 3: The testing and training error rate plotted against the number of random features (m) for each weak learner

The results for varying m are expected because lower values of m increase the randomness and thus lead to a more robust ensemble classifier.

3. In the last problem on this homework, I implemented the kmeans algorithm and used it to segment images based on pixel color. The algorithm was pretty straightforward, so I won't overview that here, but I chose to evaluate the algorithm using the distortion measure:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||$$

where r_{nk} is the indicator of $x_n \in C_k$ and μ_k is the mean of C_k. I plotted the loss per iteration and show the resulting images below:

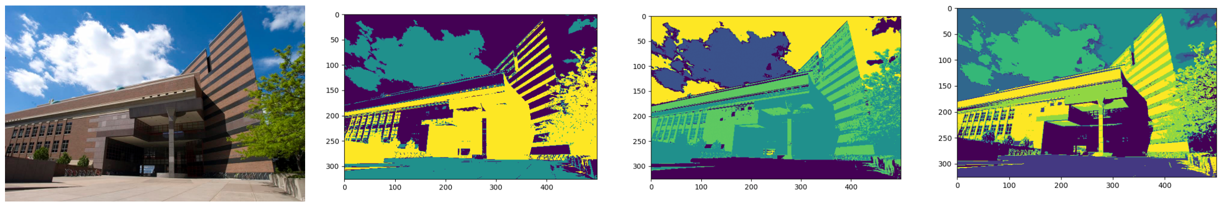

Figure 4: The original image and resulting images for kmeans segmentation for k=3,5,7

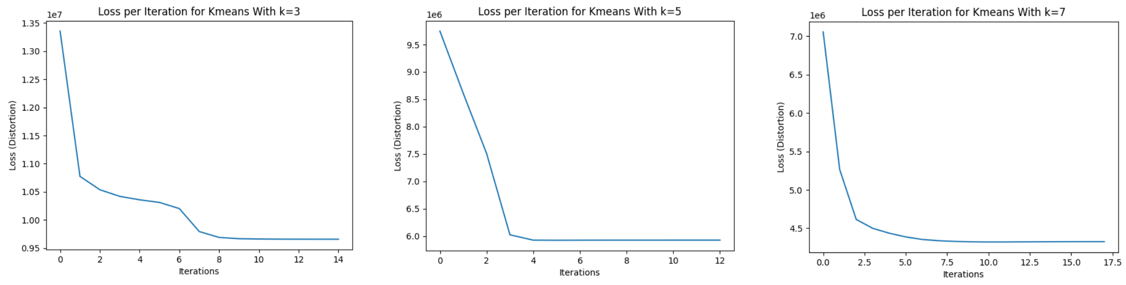

Figure 5: The loss (distortion measure) per iteration of kmeans