EECS 445 – Machine Learning - Homework #4

Zhao Fu & Valli — Due: Monday, 11/13/2016 @ 5pm
Edited by: — Last Updated: November 3, 2016 10:19pm

Homework Policy: Working in groups is fine, but each member must submit their own writeup. Please write the members of your group on your solutions. There is no strict limit to the size of the group but we may find it a bit suspicious if there are more than 4 to a team. **For coding problems, please include your code and report your results (values, plots, etc.) unless we say otherwise within the problem.** in your PDF submission. You will lose points if your experimental results are only accessible through rerunning your code. Homework will be submitted via Gradescope (https://gradescope.com/). We accept late submissions but your score will be rescaled to $Score_{rescaled} = Score_{origin} * \max(0, 1 - 0.05 * late\_hour)$.

Additional Note: You *may not* use `scikit-learn` or `scipy`'s information theory libraries for the programming exercises *unless* mentioned in specific problems, although you may use them to check if your implementation is correct.

1. Random Forest(20 pts)

In this question, we will implement the following functions in `q1_starter.py`: `plot_error`, `random_forest` and `bagging_ensemble`.

Problem 1. First, we will study decision trees on the Iris flower dataset, which consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Each sample is described by 4 features: the length and the width of the sepals and the length and the width of the petals. We will use only 2 features sepal length and sepal width.

Implement decision tree classifiers using `sklearn`. Let `criterion='entropy'`, so that our selection of which node to split on depends on the information gain. In addition, we will use `max_depth`, the maximum allowed depth of the decision tree, to avoid over-complex trees that overfit the data.

Implement `plot_error(X, y)` and generate a plot of 5-fold cross-validation training and test error vs. depth of the trees by varying `max_depth` from 1 to 20.

(Split the data and labels in 5-folds using `sklearn.cross_validation.StratifiedKFold` and train decision trees on 4 folds.) For which value of `max_depth`, does the classifier perform the best?

Problem 2. Now, we will study ensemble approaches, bagging and random forest on a handwritten digit dataset. We will use a subset of 720 examples from 4 classes.

Implement `bagging_ensemble(X_train, y_train, X_test, y_test, n_clf = 10)`. A bagging ensemble classifier consists of `n_clf` decision trees where each decision tree is trained independently on a bootstrap sample of the training data. Here, the final prediction of the bagging classifier is determined by a majority vote of these `n_clf` decision trees.

Implement `random_forest(X_train, y_train, X_test, y_test, n_clf = 10)`. Like bagging, random forest also consists of `n_clf` decision trees where each decision tree is trained independently on a bootstrap

sample of the training data. However, for each node we randomly select m features as candidates for splitting on (see parameter `max_features` of `sklearn.tree.DecisionTressClassifier`). Again, here the final output is determined by majority vote.

Now, compare the performance of these ensemble classifiers using 100 random splits of the digits dataset into training and test sets, where the test set contains roughly 20% of the data, i.e. you need to randomly generate training set and test set for 100 times. Run both algorithms on these data and obtain 100 accuracy values for each algorithm.

How does the average test performance of the two methods compare as we vary m? Choose a setting for m based on your observations and plot the result as two histograms (we've provided you with a function for plotting the histograms).

2. Clustering (20 points)

Download the image `mandrill.png` from Canvas. In this problem you will apply the k-means algorithm to image compression. In this context it is also known as the Lloyd-Max algorithm.

Problem 3. First, partition the image into blocks of size $M \times M$ and reshape each block into a vector of length $3M^2$ (see `q2_starter.py`). The 3 comes from the fact that this is a color image, and so there are three intensities for each pixel. Assume that M, like the image dimensions, is a power of 2.

Next, write a program that will cluster the vectors from **Problem 3** using the k-means algorithm. **You should implement the k-means algorithm yourself.** Please initialize the cluster means to be randomly selected data points, sampled without replacement.

Finally, reconstruct a quantized version of the original image by replacing each block in the original image by the nearest centroid. Test your code using $M = 2$ and $k = 64$.

Deliverables:

- A plot of the k-means objective function value versus iteration number.
- A description of how the compressed image looks compared to the original. What regions are best preserved, and which are not?
- A picture of the difference of the two images. You should add a neutral gray $(128, 128, 128)$ to the difference before generating the image.
- The compression ratio (use your forumla from **problem 4**).
- The relative mean absolute error of the compressed image, defined as

$$\frac{\frac{1}{3N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{3}|\tilde{I}(i,j,r) - I(i,j,r)|}{255}$$

  where $\tilde{I}$ and I are the compressed and original images, respectively, viewed as 3-D arrays. This quantity can be viewed as the average error in pixel intensity relative to the range of pixel intensities.
- Please submit you code, as usual.

Problem 4. The original uncompressed image uses 24 bits per pixel (bpp), 8 bits for each color. Assuming an image of size $N \times N$, where N is a power of 2, what is the number of bits per pixel, as a function of M, N, and k, needed to store the compressed image? What is the compression ratio, which is defined as the ratio of bpp in the compressed image relative to the original uncompressed image? *Hint: To calculate the bpp of the compressed image, imagine you need to transmit the compressed image to a friend who knows the compression strategy as well as the values of M, N, and k.*

Problem 5. (Optional, ungraded) Play around with M and k.

3. Kaggle Challenge(10 points)

Problem 6. You can use any algorithm and design any features to do this Kaggle challenge. Please refer to `https://inclass.kaggle.com/c/mdst-flint` for details. This problem will be graded separately based on your performance on the public leaderboard. To get extra credit of 2 points, you have to perform better than 0.68. We'll set a lower benchmark later for full points without extra credit. **When submitting the homework, make a screen shot of your rank on the board with your uniqname displayed**. So make sure your uniqname show up on the leaderboard(you may need to edit profile to achieve this). For this part, **you must submit for yourself individually instead of making a group submission**, though you can have discussions among your group.