EGR 598: Machine Learning and Artificial Intelligence (Fall 2023)

Instructor: Shenghan Guo

Assignment 2

This assignment is worth 160 points. It will be due on Thursday, 10/31/2023, and submitted through Canvas. Code should be written in Python. Other programming language can be used upon instructor's approval. A written report should be submitted as a separate document along with the code through Canvas. If you use Jupyter Notebook, then you may submit a single ".ipynb" file through Canvas.

Data: A dataset, "iris.data", and an image, "Assignment 2_image.jpg", are provided to you. Data description for "iris.data" can be found in "iris.names". Please read through the document and then use the data to do the following tasks.

Note: You may consider the samples as independent and identically distributed (iid).

Part 1 (60 pts)

Use "Assignment 2_image.jpg" for this part.

1. Image compression with k-means: k-means clustering algorithm can be used for image compression. The image is divided into nonoverlapping $c \times c$ windows and these c^2-dimensional vectors make up the sample. For a given k, we do k-means clustering. The reference vectors and the index for each window is sent over the communication line. At the receiving end, the image is then reconstructed by reading from the table of reference vectors using the indices. Write the computer program that does this for $k \in \{4, 8, 16\}$ and $c \in \{20, 40, 60\}$. For each pair of (k, c), (1) save the compressed images as ".jpg" or ".png", and (2) calculate the reconstruction error (Eq. (6.12) in textbook).

Each pair of (k, c) will have a reconstruction error and a compression rate, so you will have 9 pairs of rates. Use Python library "scikit-learn.clsuter". The image may be RGB-colored, so you can flatten each window into a vector of length $c \times c \times 3$ using "ndarray.flatten('F')" in Python and then convert it back to a $c \times c \times 3$ matrix when reconstructing the compressed image. (Hint: Use "for" or "while" loop to exhaust all combinations of (k, c). To visualize your compressed images without issues, keep in mind to use integer data type, i.e., "int" in Python, for pixel values.) (30 pts)

2. Based on the reconstruction errors in 1, identify the best combination of k and c. (5 pts)

3. **Image compression with hierarchical clustering**: Repeat the image compression task in 1 using *agglomerative clustering* algorithm with *complete link*. Again, $k \in \{4, 8, 16\}$ and $c \in \{20, 40, 60\}$. For each pair of (k, c), (1) save the compressed images as ".jpg" or ".png", and (2) calculate the reconstruction error. (Hint: Change the clustering function to "AgglomerativeClustering" in Python library "scikit-learn".) (20 pts)

4. For the results of 3, identify the best value for k and c based on the reconstruction error and compression rate. (5 pts)

Part 2 (100 pts)

Use "iris.data" for this part. You may find the variable names in "iris.names". The final column of "iris.data" is the ground-truth class.

1. For class "setosa", visualize their *kernel estimate* using a *Gaussian kernel* for bin length $h \in \{1, 0.5, 0.25\}$. You will use multivariate Gaussian density. This is the high-dimensional version of Eq. (8.6) and (8.7) in textbook. (Hint: To get the horizontal axis, generate data sequence for the range of each variable and then use them together to get one set of Gaussian kernel estimate. Then, visualize the Gaussian estimate against each variable, respectively, to get 4 kernel estimate plots for each h value.) (15 pts)

2. Split the iris data into training set (take 50% of the data points in each class) and testing set (the rest 50% of the data points in each class). This type of data split is called "stratified sampling". (10 pts)

3. Perform classification analysis with *k-nearest neighbors*. $k \in \{3, 5, 10, 20\}$. Specifically, you will fix a k value and train k-nearest neighbors classifier with the training set in 2 and then use the classifier on testing set to do prediction. Calculate the *confusion matrix* and prediction accuracy for each k. Identify the k value giving the best performance and briefly comment the result. (Hint: Use "neighbors" in Python "scikit learn" library.) (25 pts)

4. Repeat the problem in 2 using *classification tree* with *Gini index* as the impurity measure. Still, you will report the confusion matrix and prediction accuracy. (Hint: Use "tree" in Python "scikit learn" library.) (20 pts)

5. Repeat the problem in 2 using *logistic regression*. Still, you will report the confusion matrix and prediction accuracy. (Hint: Use "linear_model" in Python "scikit learn" library.) (20 pts)

6. Among k-nearest neighbors, classification tree, and logistic regression, which achieves the best classification accuracy? Please explain why the classification method performs better than the others. (Hint: you can make reasonable conjecture.) (5 pts)

7. In terms of the class densities, what is the biggest difference between logistic regression and the other two classification methods (k-nearest neighbors and decision tree)? (5 pts)

***Bonus: Part 3 (40 pts)**

Use "Assignment 2_image 1.jpg" for this part.

1. **Object recognition with neural nets:** Convolutional Neural Nets (CNNs) are powerful tools in computer vision. There are multiple model variants based on CNNs that recognize objects in an image. An example is *You-Only-Look-Once (YOLO)* (https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9). There is pretrained YOLO model accessible through Python library. For the given image, please use the pretrained YOLO model to recognize objects. Comment on the correctly and incorrectly recognized objects. (35 pts)

2. Briefly explain the connection and difference between CNNs and multilayer perceptron. (5 pts)