Author: Jinjin Qin, Liu Yang

Preparation

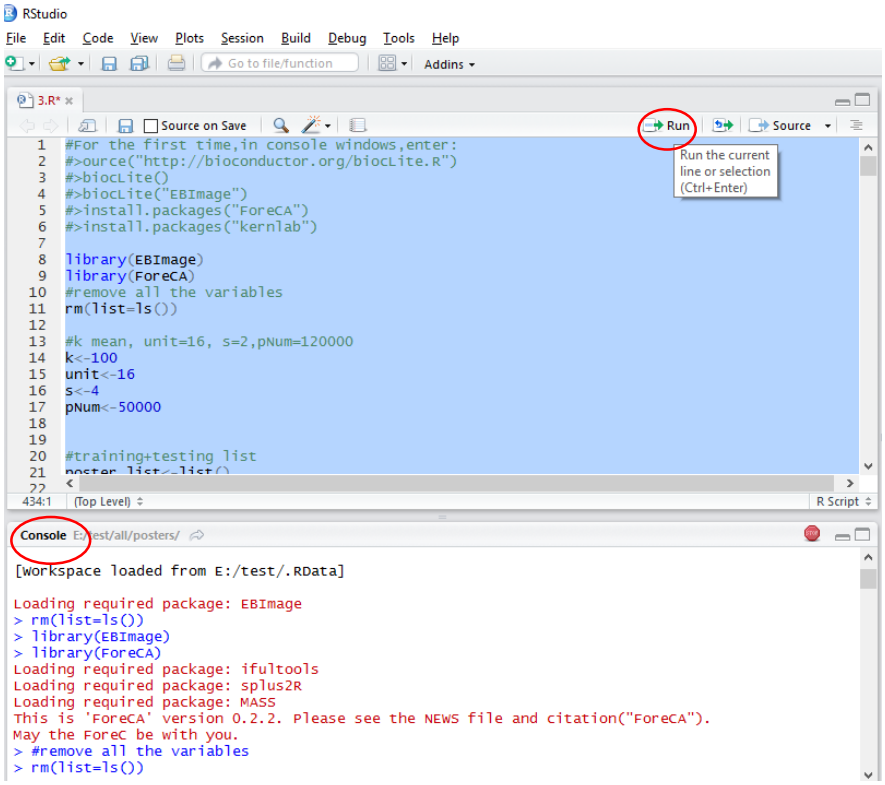
1. Install RStudio on server/computer
https://www.rstudio.com/products/rstudio/download/
2. Open file 3.R with RStudio
3. **For the first time to run the program**, you need to enter commands to install some packages in the console window:

>ource("http://bioconductor.org/biocLite.R")
>biocLite()
>biocLite("EBImage")
>install.packages("ifultools")
>install.packages("splus2R")
>install.packages("MASS")
>install.packages("ForeCA")
>install.packages("kernlab")

4. Change the path in the program:

Line 38: setwd("E:/test/all/posters") #change the path to labeled posters
Line 73:setwd("E:/test/all/originals") #change the path to labeled originals
Line 403: setwd("E:/poster/") #change the path to new posters you want to test
Line 419: setwd("E:/origin/") #change the path to new originals you want to test

5.select the whole program, click **RUN**



Author: Jinjin Qin, Liu Yang

Explanation:
The whole program is divided to 9 parts:
1. Read in all the unlabeled posters and originals (training data) from the file, and do grayscale+(resize to 96*96 matrix), then store them in different lists.
Line 20~Line 86.

2. Divide the list to 2 parts: 80% as training data, 20% as test data.
Line 91 ~ Line 107 in the program

3. Randomly extract pNum(80000) patches(size: 16*16) from images **in the training data**(resized to 96*96 in the 1st step), convert every patch to 256-dimensional vector, combine patch vectors to be a 256*80000 matrix. Normalize each X_i with its col_mean and col_variance, do ZCA whitening to the whole matrix.
Line 117 ~ Line 182

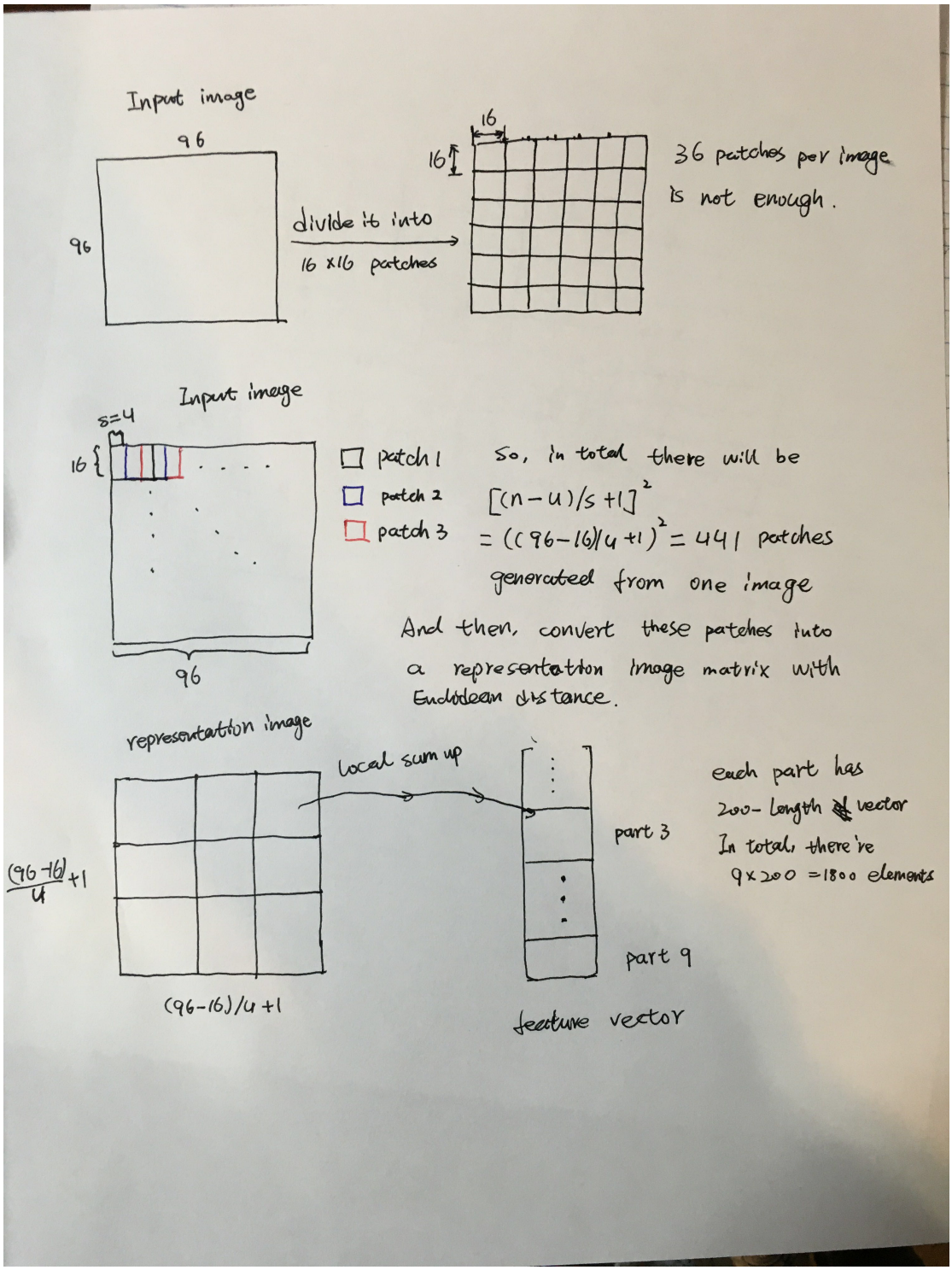
4. Use k-means clustering to find k (set to be 200) centers for the matrix(256*80000), and get a kcentroids(256*200)
Line 190 ~ Line 192

5. Uniformly select 441 patches (16*16) from each image in the training data set and the testing data set, store them in different matrix.
Each patch(16*16) is converted to 256-dimensional vector, one image is represented with a 256*441 matrix named X(need to do locally sum up later, so insert the patch into the matrix in a different way).
Compute the euclidean distance between the matrix and the kcentroids(256*200), then we get a 441*200 matrix G, G_{ij} is the distance between the ith vector in matrix X(X_i) and the jth vector in kcentroids.
Compute the rowMean to get the average distance of each X_i to k centroids.
Compute max(0, rowMean - G) #k-means(triangle)

> 3. **K-means clustering:** We apply K-means clustering to learn K centroids $c^{(k)}$ from the input data. Given the learned centroids $c^{(k)}$, we consider two choices for the feature mapping f. The first is the standard 1-of-K, hard-assignment coding scheme:
>
> $$f_k(x) = \begin{cases} 1 & \text{if } k = \arg\min_j ||c^{(j)} - x||_2^2 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$
>
> The second is a non-linear mapping that attempts to be "softer" than the above encoding, but also yield sparse outputs through simple competition:
>
> $$f_k(x) = \max\{0, \mu(z) - z_k\} \quad (3)$$
>
> where $z_k = ||x - c^{(k)}||_2$ and $\mu(z)$ is the mean of the elements of z.
> We refer to these as `K-means (hard)` and `K-means (triangle)` respectively.

So far, the 441 patches(256*441) are represented in G(441*200).
Divide the whole image to 9 parts, each part contains 49 overlapped patches, add up every 49 adjacent patches and get an average 200-dimensional vector.
The whole image can be represented by 9 200-dimensional vectors, convert to 1 vector--1800-dimensional vector.
Line 204 ~ Line 281
The following picture illustrates these steps.

Author: Jinjin Qin, Liu Yang



6. Add labels.
Line 286~ Line 292

7. Shuffle the training data set, do 5-fold cross validation with SVM model.
Line 305~Line 334
8. Use the model to test the test data.
Line 342 ~ Line 349

9. Test new data whose label is unknown with the model.
Line 400 ~ Line 433

Notice:
The model works well on the given dataset(accuracy 82%~88% on validation and test data set).

It doesn't work well on the new images we downloaded from the internet.(85% ~ 90% for originals, but 65%~70% for posters)
For this problem we tried the following ways, but still didn't have a optimal solution:

1. Increase K to 800, 1000 or 1200; increase pNum to 100,000 or 120,000; reduce s(stride) from 4 to 2(This almost reached our computers' limitation). This results to 10,000 features for one image(9*k), and since we only have 1000 images in total. The accuracy on validation and testing dataset increases, but worked worse on new images downloaded from internet.
2. Due to this, it may suffer from overfitting problem, we reduced k to 200, pNum to 60,000 or 80,000, and add 250 new images to the training data set. The accuracy on validation and test data reduced, but worked better on new images from internet.

Besides, about 15 images in the data set we are given can't be recognized by the R's readImage function, and that will cause crash of the program(It insert NA into the list), so we deleted them by hand.

For the above problems, if possible, we want to increase the size of training dataset and test the program on the server to see if it can perform better.