Home Assignment 1

Introduction to Machine Learning.   Due: December 19, 2018

General Instructions:

The solution should be formatted as a report and running code should be included in a digital form. The solution can be done in pairs independently to other groups. Identical (or very similar solutions) are not allowed!

1. Skin Detection

This assignment performs skin detection in images based on skin color using MAP classifier.

Your interaction with all the data and everything else is totally encapsulated in the `corresponding matlab` scripts. At a very high level, once you complete all the required implementations, you will only need to run these scripts to obtain the results. The scrips will each run a fairly large number of tests, generate some graphs, print things to the screen, and create files.

Data and Supplied Code: download file SkinDetection_distr.zip from the moodle. The archive contains the data (images and the skin masks) in **`ibtd`** directory.

The code involves two steps:

1. Training: learn likelihood for pixel color, given skin and non-skin pixels (a database of images with the segmented skin pixels is provided) and assuming that both distributions are 2D Gaussians.
2. Testing: classify a new image into skin/non-skin regions.

You are provided with a partial implementation of the system. There are comments in all functions that you should implement explaining the data structures that you should use and the parameters. You should find and learn about ROC curve and AUC (area under ROC curve) in the literature or Web.

The main file to run is `runSkinClassification.m` It takes the first 370 labeled images, 2/3 of them it uses for training the models of the skin and non-skin pixels (aka training set), and 1/3 it uses as a validation set to find the best threshold on the likelihood ratio. We use only 2 out of three channels (of RGB) to decrease the computational load.

Once the parameters of the models and the threshold are found it runs the classification on k images (starting from 0371). You should pass k as a parameter to `runSkinClassification` routine (for example `runSkinClassification(5)` will test 5 images). The classification script computes the ROC curve and the AUC statistic for each image and displays the predicted and the true skin pixels for each image. Note that the ROC computation takes time, so be patient or replace the provided implementation with your own, or try using the matlab function for it. The `runSkinClassification` outputs the AUC averaged over k test images.

Your task is to implement the training as explained above in script `learnSkinModel.m` (search for places saying **`%add you code here`**), specifically you should find the parameters of Gaussian distribution for the skin and non-skin pixels. The code, provided in the script, constructs the training and validation sets for you. You should find the parameters using the training sets and you should find the best threshold on the likelihood ratio using the validation set.

You next task is to implement the classification procedure in the script `classifyImg.m`. Again, the code that deals with feature creation is implemented. You task is to add **your** implementation of classification, using the estimated parameters and the threshold. This script shows the ROC curve, the predicted mask (the skin pixels are shown in red and the background pixels in blue) and the true skin pixels from labeling.
Warning: Note that the predicted mask should be similar to the true mask to some extent, but they will not be identical for any threshold.

2. Text Classification Using Naïve Bayes Algorithm

Implement in Matlab text classification using Naïve Bayes Algorithm:

a. Write a function **`[Pw, P]=learn_NB_text`** that computes probabilities (**`Pw`** –a matrix of class-conditional probabilities, **`P-`** a vector of class priors).

b. Write a function **`suc=ClassifyNB_text(Pw, P)`** that classifies all documents from the test set and computes the success rate (**`suc`**) as a number of correctly classified documents divided by the number of all documents in the test set.

c. Report the classification rate.

Note: Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow. Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities. Class with highest final un-normalized log probability score is still the most probable.

Data and Supplied Code: Download `textClassif.zip` from moodle. The archive contains the following files.

`r8-test-stemmed.txt` -- test set for 8 categories.
`r8-train-stemmed.txt` -- train set for 8 categories.

`readTrainData.m` -- a matlab function that reads all documents from the train set creates the following data structures and saves them in a mat file "corpus_train.mat":

`texAll` - cell array of documents; each entry corresponds to a document which is a cell array of words.

`lbAll` - cell array of documents' labels.

`Voc` - cell array of all distinct words in the train set.

`cat` - cell array of document categories.

`readText.m` a matlab function used in `readTrainData`.m

Instructions for using the code:

1. Unzip the data and the code in the same directory.
2. Run `readTrainData('r8-train-stemmed.txt')`. It should create `corpus_train.mat` file in the same directory.
3. Write the required functions and run them. You can use the code in `readTrainData`.m as an example for reading the test data.
4. Report the success rate on the test set.

Good Luck!