

COMP 448/548 – Medical Image Analysis

Homework #2

Due: 23:59, May 10, 2022

In this homework, you will design an algorithm for classifying colon tissue images. This algorithm relies on representing a tissue image with a set of handcrafted texture features and classifying the image based on these features using a support vector machine (SVM) classifier.

The steps you need to follow are detailed below. Except for the SVM classifier and the operations explained in Part 4 and, you are required coding your own implementations. In other words, **you are NOT allowed to use any built-in library functions to implement those listed in Part 1.** (Of course, you may use built-in library functions for mathematical operations such as log.)

What to submit: Submit the following via Blackboard.

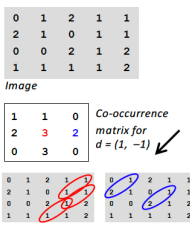
- Your source codes. Put all of your source codes in a zip file called **Lastname_FirstName_HW2.zip**
- Report (follow the instructions separately given for Part 2, Part 3, and Part 4). You are expected to write your report neatly and properly. The format, structure, and writing style of your report as well as the quality of the tables will be a part of your grade. Use reasonable font sizes, spacing, margin sizes, etc. You may submit either a one-column or a double-column document. The filename of your report should be **Lastname_FirstName_HW2.pdf**

PART 1: Implementations

In this homework, you will use co-occurrence matrices to define and extract texture features. Thus, please be sure that you will have revised the material from the slides (7_TextureAnalysis.pdf) before starting the homework. The necessary slides are also copied below. The rest of the homework will be explained using the notations given in these slides.

Gray-level co-occurrence matrix

- Second-order statistical features
- A gray-level co-occurrence matrix P is an NxN array, where N is the number of gray levels in the image
- $P(i, j)$ gives how many times gray-levels i and j co-occur at a given distance $d = (di, dj)$



5

Gray-level co-occurrence matrix

- Common to use normalized co-occurrence matrix $N(i, j) = \frac{P(i, j)}{\sum_u \sum_v P(u, v)}$
- Sometimes useful to group gray-levels into bins
 - E.g., Four bins: [0, 63], [64, 127], [128, 191], [192, 255]
- Sometimes useful to accumulate co-occurrence matrices calculated for different distances
 - For example, for a rotation invariant image, it is common to calculate $P(i, j)$ at $d1 = (di, dj)$, $d2 = (-di, dj)$, $d3 = (di, -dj)$, and $d4 = (-di, -dj)$ and take their summation
- Statistical features are computed from the co-occurrence matrix to represent the texture more compactly (*Haralick features*)

6

Haralick features

Angular second moment = $\sum_i \sum_j N(i, j)^2$

Maximum probability = $\max N(i, j)$

Inverse difference moment = $\sum_i \sum_j \frac{N(i, j)}{1 + (i - j)^2}$

Contrast = $\sum_i \sum_j (i - j)^2 N(i, j)$

Entropy = $-\sum_i \sum_j N(i, j) \log N(i, j)$

Correlation = $\frac{\sum_i \sum_j i j N(i, j) - \mu_i \mu_j}{\sigma_i \sigma_j}$

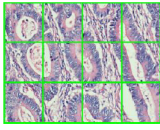
μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of $N_i = \sum_j N(i, j)$ and $N_j = \sum_i N(i, j)$, respectively.

Haralick et al., "Textural features for image classification," IEEE Transactions on Systems, Man, and Cybernetics, 1973.

7

Image classification using Haralick features

- Calculate a co-occurrence matrix on an entire image
- Calculate the Haralick features (or a subset of them) of this matrix
- Classify the image based on these features
- **Grid-based approach**
 - Divide an image into grids
 - Calculate a co-occurrence matrix on each grid
 - Calculate the Haralick features of each calculated matrix
 - Average the feature values
 - Classify the image based on these average features



8

First implement the following functions using a programming language you will prefer. You will use these functions in reporting the results for Part 2, Part 3, and Part 4.

1. **`M = calculateCooccurrenceMatrix(grayImg, binNumber, di, dj)`**

This function takes a grayscale image **`grayImg`** as an input, groups its pixels into **`binNumber`** bins, and returns a co-occurrence matrix **`M`** calculated on these grouped pixels for a given distance **`(di, dj)`**. This function may take the grayscale of an entire image (for Part 2) or the grayscale of a cropped subimage (for Part 3) as its input.

2. **`accM = calculateAccumulatedCooccurrenceMatrix(grayImg, binNumber, d)`**

This function accumulates co-occurrence matrices calculated for different distances specified by the parameter **`d`** and returns the accumulated matrix **`accM`**. To do so, it calculates eight different co-occurrence matrices for the following distances

`(d, 0), (d, d), (0, d), (-d, d), (-d, 0), (-d, -d), (0, -d), (d, -d)`

and returns the sum of the calculated matrices. Likewise, this function may take the grayscale of an entire image (for Part 2) or the grayscale of a cropped subimage (for Part 3) as its input.

3. **`feat = calculateCooccurrenceFeatures(accM)`**

This function takes an accumulated co-occurrence matrix **`accM`** as an input and calculates six texture features on the normalized matrix (i.e., it should first normalize the matrix and then calculate the features on this normalized matrix). These six features are; the angular second moment, maximum probability, inverse difference moment, contrast, entropy, and correlation (please see the slides for their definitions).

PART 2: Classification

For each image, calculate the accumulated co-occurrence matrix inputting the entire image to the **`calculateAccumulatedCooccurrenceMatrix`** function and extract the six texture features from this matrix using the **`calculateCooccurrenceFeatures`** function. For the parameters, use the following values: **`binNumber = 8`** and **`d = 10`**.

Then, train an SVM classifier on the features of the training images. You may use any available SVM classifier implementation. Below is one of such implementations.

<https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

When you train the SVM classifier, please do not forget the following:

- SVMs were originally developed for binary classifications and then extended for multi-class classifications. Thus, please be sure that you will have used the multi-class version of these SVM classifiers.
- You need to select a kernel for the SVM. In this homework, first use a linear kernel due to the simplicity of its parameter selection. In this case, there is only one parameter, called *C*, to be selected. You need to try different values of *C* from a wide range. For example, you may try 0.1, 1, 5, 10, 50, 100, 250, 500, 1000, 5000, ... (This is not the final list of values. As a part of this homework, please determine your own list.) You will report the training and test set results obtained when the linear kernel is used.
- Then, train another SVM, this time with an RBF kernel. In this case, in addition to the parameter *C*, there is another parameter called *gamma*, which also needs to be selected. Try different values of *gamma* and *C* (like a grid search) for this parameter selection. You will also report the training and test set results obtained when the RBF kernel is used.
- There is a class-imbalance problem in the training set. Take proper actions, if necessary.
- Remember our discussion about normalizing the features. Take proper actions, if necessary.

After training these two SVM classifiers (one with the linear kernel and the other with the RBF kernel), obtain accuracies for the training and test set images. Here obtain the overall accuracies as well as the class-based accuracies. Report these accuracy results complying with the format given below. At the end, also apply the Mc Nemar's test with $\alpha = 0.05$, to see whether or not the accuracies obtained with the linear kernel are statistically significantly different than those obtained with the RBF kernel.

What to submit:

- Report (a maximum of 1 page for this part): Use the following format to report your findings.

Implementation details									
Did you take an extra action for the class-imbalance problem? If so, explain.									
Did you take an extra action for normalization? If so, explain.									
Specify the library from which you use the SVM. Alternatively, you may give a link for its implementation.									
Additional comments, if you have any.									
	Selected parameters	Training set accuracies				Test set accuracies			
		Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall
Linear kernel	C = ?								
RBF kernel	C = ? gamma = ?								
Statistically different?		yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no

PART 3: Grid-based approach

For each image, follow a grid-based approach to extract its texture features. For that, first divide an image into $N \times N$ grids, find the accumulated co-occurrence matrix on each grid using the `calculateAccumulatedCooccurrenceMatrix` function and calculate the aforementioned six features for each accumulated co-occurrence matrix using the `calculateCooccurrenceFeatures` function. At the end, take the average of these six features over all grids to obtain the feature set of the given image. Note that here you need to crop a subimage that corresponds to a grid entry from the entire image and use it as the input of the `calculateAccumulatedCooccurrenceMatrix` function. For the parameters, use the following values: **N = 4**, **binNumber = 8** and **d = 10**.

Likewise, train SVM classifiers on this new set of features of the training images. Do not forget to consider the aforementioned issues related to the SVM training.

After training the two SVM classifiers (one with the linear kernel and the other with the RBF kernel), obtain accuracies for the training and test set images. Likewise obtain the overall accuracies as well as the class-based accuracies. Report these accuracy results complying with the format given below. At the end, also apply the Mc Nemar's test with $\alpha = 0.05$ to see whether or not there exists a statistically significant difference between the following:

- The first approach and the grid-based approach when SVMs with linear kernels are used
- The first approach and the grid-based approach when SVMs with RBF kernels are used

What to submit:

- Report (a maximum of 1 page for this part): Use the following format to report your findings.

Implementation details									
Did you take an extra action for the class-imbalance problem? If so, explain. If you have already explained it in Part 2 and if you give the same explanation, just write <i>see Part 2</i> .									
Did you take an extra action for normalization? If so, explain. If you have already explained it in Part 2 and if you give the same explanation, just write <i>see Part 2</i> .									
Any additional comments:									
	Selected parameters	Training set accuracies				Test set accuracies			
		Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall
Linear kernel (grid-based)	C = ?								
Linear kernel (entire image)	C = ?								
<i>Copy and paste from the previous table</i>									
Statistically different?		yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no
RBF kernel (grid-based)	C = ? gamma = ?								
RBF kernel (entire image)	C = ? gamma = ?								
<i>Copy and paste from the previous table</i>									
Statistically different?		yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no	yes or no

PART 4: Extension

Select and work on one of the following extensions. Of course, you may work on both of them. However, you can earn credits for only one of them.

Your report should be of a maximum of 1 page for each extension.

Extension 1: Instead of cropping subimages from a grid, find the key points by the SIFT algorithm and crop subimages around these key points (see Pages 5-6 of 8_FeatureExtraction.pdf). You may use third-party codes for the SIFT algorithm. Repeat all experiments explained in Part 3 and provide a table similar to the one that you will give for Part 3.

Extension 2: Instead of extracting texture features using co-occurrence matrices, use a filter-based approach (see Pages 14-17 of 7_TextureAnalysis.pdf). You may use built-in functions (e.g., `imfilter` in Matlab) for the filtering operation. Repeat all experiments explained in Part 2 except that now you have a new set of parameters. Provide a table similar to the one that you will give for Part 2. In this table, also briefly discuss how you extract your features and the parameters associated with this feature extraction.