Image Interpretation – Assignment 4

This assignment on **Classification** covers the labs from the 24th and 31st October. The exercises on this sheet are graded by a maximum of 10 points. You will be asked to implement several functions and write a brief report.

In order to submit the results, send a ZIP file with your implemented code (functions prefixed with `ii_` and any "helper" functions you wrote) to

`<mikhail.usvyatsov@geod.baug.ethz.ch>`

with subject

`Image Interpretation 2019 Assignment 4`

no later than on the

`6th of November, 2019.`

Your functions should work when called by the provided test code (functions prefixed with `test_`) which **must not** be modified. When run, they should produce a plausible output, no warnings, and no unnecessary output (remember to end your statements with semicolons). Eventual example output is given in the `ref_`-images. The functions that you write take images as arguments, not image filenames (i.e. do not use `imread` inside the functions). Your functions should not generate figures/plots themselves, the plots are generated by the test scripts.

In addition to the functions, include a brief report (max. 4 pages PDF) explaining the structure of the code and the MATLAB functions used. This includes the reasons for choosing particular functions as well as a short justification of their parameter setting. For the more complicated tasks, the choice of the underlying data structures and algorithms should be explained too. We encourage you to add also diagrams, illustrations, and figures into the report when appropriate, but it is not necessary to copy the related theory from the lecture slides. The report should not contain any code snippets (but the code should contain comments if appropriate).

Team work is not allowed. Everybody implements his/her own code and writes his/her own report. Discussing issues with others is fine, sharing code and/or report with others is **not**. If you use any code fragments found on the Internet, make sure you reference them properly.

This exercise focuses on using MATLAB built-in classification methods, as implementing most classifiers takes more time than is reasonable in the scope of this assignment. The exception is the exercise about LDA, which is simple enough to be done manually. The rest of the programming exercises are very short.

As in the regression assignment, Y stands for a column vector of "target" values, that is the i-th row of Y contains the desired output for the i-th data point. Contrary to regression, the elements of Y are integer values. In most exercises we use binary classifiers, where one class has the label 1 and the other class has the label 2. X is a matrix containing the feature values, where the i-th row contains the values for the i-th data point.

The synthetic data are generated randomly for each run in the "toy example" test scripts, so you can run the scripts multiple times to get different outputs.

In your reports, comment on the performance of the individual methods when applied on the aerial photo of Graz. Describe also the difference between the results on data seen during training (the left quarter of the image) and data that are "new" (the rest of the image). Does a given method work equally well for every class? You can (and actually need to in order to generate the results for the report) change the "true" class in the test scripts that use binary classification by changing the `true_class` variable if you want to see outputs for different classes. A value of 1 represents streets, 2 buildings, and 3 vegetation. A reference to this is given in file `untitled.m`.

You can also use more features for classification (features from a lower scale) by employing function `ii_gaussian` which you implemented in the second exercise (see the commented lines with `create_features` in the test scripts).

Exercise 1. 1 P.

Nearest neighbour classification (especially its k-NN variant) is a simple but yet quite powerful classification method which does not make any assumptions about the data model.

MATLAB function `fitcknn` performing k-NN classification is demonstrated in scripts `demo_knn` (toy example) and `demo_knn2` (Graz aerial photo). In the toy example, one can see the classification of red and blue points into red and blue classes, respectively. Graz aerial photo example and the class values for the second test have been explained in the introduction already.

Run the scripts for different values of parameter K, *distance* and *rule* and comment the results in your report. Also report the best parameters with explanation why you think that they are the best.

Exercise 2. 4 P.

LDA classifier can be used to efficiently classify linearly separable data. Implement function `ii_train_lda` that trains an LDA classifier. The input arguments are the training data (one row per sample and one column per feature) and a column vector with the class labels of the samples (0 or 1). Return the normal θ of the decision boundary and the threshold t between classes (so that if $X\theta - t > 0$, class 1 is predicted). Since we use row vectors as features, the formulas are slightly different from those seen in the lecture (but θ is still a column vector).

m_1 and m_2 are the means of the input features over all rows where Y is 0 or 1 respectively. You can calculate mean and covariance for the classes with MATLAB functions `mean` and `cov`. In order to do operations on the rows of X where the corresponding row in Y has a certain value (e.g. 1), you can use the expression `X(Y==1,:)`. Instead of determining θ by matrix inversion, which is numerically unstable, use MATLAB linear equation solving capabilities:

```
normal = SW \ (m2-m1)';
```

where SW is the sum of within-class covariances. The threshold you output is the value of the projection at the point located in the middle between the class means (the (dot) product of the mean of m_1 and m_2 and the estimated normal).

The corresponding `ii_test_lda` is already provided. You can use scripts `test_lda` and `test_lda2` to test your implementation and generate results for the report.

Exercise 3. 3 P.

Logistic regression is a probabilistically principled classification method. You are provided with the breast cancer dataset. You task is to implement function `ii_train_logress` that uses MATLAB function `glmfit` to fit a logistic regression model to the data. It should take the same input arguments as `ii_train_lda` and return the trained model parameters (a vector). Also implement function `ii_test_logress` that takes the trained model parameters and input features (in the common format) and returns the output of the logistic regression model using `glmval`. What does `glmval` output? The setting of the parameters of `glmfit` and `glmval` can be a bit tricky, see the supplementary document `link.pdf` for explanation.

You can use scripts `test_logress` and `test_logress2` to test your implementation.

Implement the script `classify_breast_cancer_with_logress.m` which using the functions `ii_train_logress` and `ii_test_logress` can distinguish benign and malignant tumor. Use 75% of the dataset for training.

Note, that there are missing values in the dataset. You have to either:

1. remove such examples
2. interpolate missing values

in order to let matlab load the data for you with `csvread('materials/dataset/data.csv')`.

You can find description of the dataset in file `description.txt`. Report how well logistic regression works for the breast cancer prediction (accuracy).

Exercise 4. 2 P.

Classification using random trees and forests is currently very popular thanks to its efficiency and flexibility. Implement function `ii_train_tree` that uses MATLAB function `fitctree` to train a decision tree. The function should use the same input arguments as the previous training functions, but note that the tree classifier can handle more than two classes, so this time the input is not constrained to -1 or 1 (you do not need to handle this, this is done automatically by the classifier). The function returns the trained decision tree. Also implement function `ii_test_tree` that takes the trained decision tree and input features (in the common format) and returns the prediction made by the tree (search the documentation of `ClassificationTree` for method `predict`).

You can use scripts `test_tree` and `test_tree2` to test your implementation and generate results for the report.

Implement the script `classify_breast_cancer_with_tree.m` which using the functions `ii_train_tree` and `ii_test_tree` can distinguish benign and malignant tumor. Use 75% of the dataset for training.