Technische Universität München
Fakultät für Informatik
Dr. Slobodan Ilic

WS 13/14
Exercise Sheet 6
November 28, 2013

Exercises in Tracking & Detection

Exercise 1 AdaBoost

In this exercise we will implement the AdaBoost classifier for synthetic datasets.

- Download the synthetic datasets (**data1.mat**, **data2.mat**, **data3.mat**) from the website of the course. Each sample (1 positive and -1 negative) has two features (x and y position).
- Implement the Adaboost algorithm. A class based implementation is preferred (Figure 1).
- The weak classifiers should be based on simple thresholding, applied in an axis (decision stump classifiers). They are required to perform slightly better than random.
- For every dataset, do the following:
  - Visualize the data samples before the classification (use different colors for the positive and the negative samples).
  - Train N weak classifiers and then test on the dataset.
  - Visualize the data samples after the classification (use again different colors for each class).
  - Visualize the classification error in comparison to the number of the week classifiers.

Exercise 2 Face detector

In this exercise we will implement a Face Detector which is based on the Adaboost algorithm and the Haar-like features (Paul Viola, and Michael Jones, *Robust Real-Time Face Detection.* In International Journal of Computer Vision, 2004). At first, read carefully the publication and try to understand how the features are extracted and the training is done. It is not required to implement the training and the cascade part.

- Implement the feature extraction part of the Viola-Jones face detector.
  - Write a class implementing Haar-like features. The constructor initializes the features from the file **Classifiers.mat**. A method must be created to extract the features from an image patch.
  - Write a function for constructing the integral image from a gray scale image.
  - The feature extraction function should be able to extract different kind of features (two,three and four-rectangle features is enough).
- Load the weak classifers from the file **Classifiers.mat**. The file contains N information about a pre-trained Adaboost classifier.

- Each entry describes one weak classifier having the following attributes :
  INFORMATION ABOUT THE HAAR-FEATURE USED BY THE CLASSIFIER
  - upper left corner row (r): (row position of the Haar-like feature upper left corner)
  - upper left corner column (c): (column position of the Haar-like feature upper left corner)
  - Haar-like feature width (winWidth)
  - Haar-like feature height (winHeight)
  - Haar-feature type number: 1 - 5

  INFORMATION ABOUT THE WEAK CLASSIFIER
  - mean of positive responses for the employed feature (mean)
  - standard devation of positive responses for the employed feature
  - max positive value of the feature response (maxPos)
  - min positive value of the feature response (minPos)
  - classifier's number (R out of n rounds)
  - alpha that should be multiplied choosen by Adaboost
  - error of this classifier
  - False negative error
  - False positive error

- There are 5 different types of weak classifiers, which are based on Haar-like responses. They have the following form:
  - Classifier 1
    - Rectangle 1: $[r \quad c \quad ((winWidth/2)-1) \quad (winHeight-1)]$
    - Rectangle 2: $[r \quad (c+winWidth/2) \quad ((winWidth/2)-1) \quad (winHeight-1)]$
    - Feature response: $Rectangle1 + Rectangle2$
  - Classifier 2
    - Rectangle 1: $[r \quad c \quad (winWidth-1) \quad ((winHeight/2)-1)]$
    - Rectangle 2: $[(r+winHeight/2) \quad c \quad (winWidth-1) \quad ((winHeight/2)-1)]$
    - Feature response: $Rectangle1 + Rectangle2$
  - Classifier 3
    - Rectangle 1: $[r \quad c \quad ((winWidth/3)-1) \quad (winHeight-1)]$
    - Rectangle 2: $[r \quad (c+(winWidth/3)) \quad ((winWidth/3)-1) \quad (winHeight-1)]$
    - Rectangle 3: $[r \quad (c+((2*winWidth)/3)) \quad ((winWidth/3)-1) \quad (winHeight-1)]$
    - Feature response: $Rectangle1 - Rectangle2 + Rectangle3$
  - Classifier 4
    - Rectangle 1: $[r \quad c \quad (winWidth-1) \quad ((winHeight/3)-1)]$
    - Rectangle 2: $[(r+(winHeight/3)) \quad c \quad (winWidth-1) \quad ((winHeight/3)-1)]$
    - Rectangle 3: $[(r+((2*winHeight)/3)) \quad c \quad (winWidth-1) \quad ((winHeight/3)-1)]$
    - Feature response: $Rectangle1 - Rectangle2 + Rectangle3$
  - Classifier 5
    - Rectangle 1: $[r \quad c \quad (winWidth/2-1) \quad (winHeight/2-1)]$
    - Rectangle 2: $[r \quad (c+winWidth/2) \quad (winWidth/2-1) \quad (winHeight/2-1)]$
    - Rectangle 3: $[(r+winHeight/2) \quad c \quad (winWidth/2-1) \quad (winHeight/2-1)]$

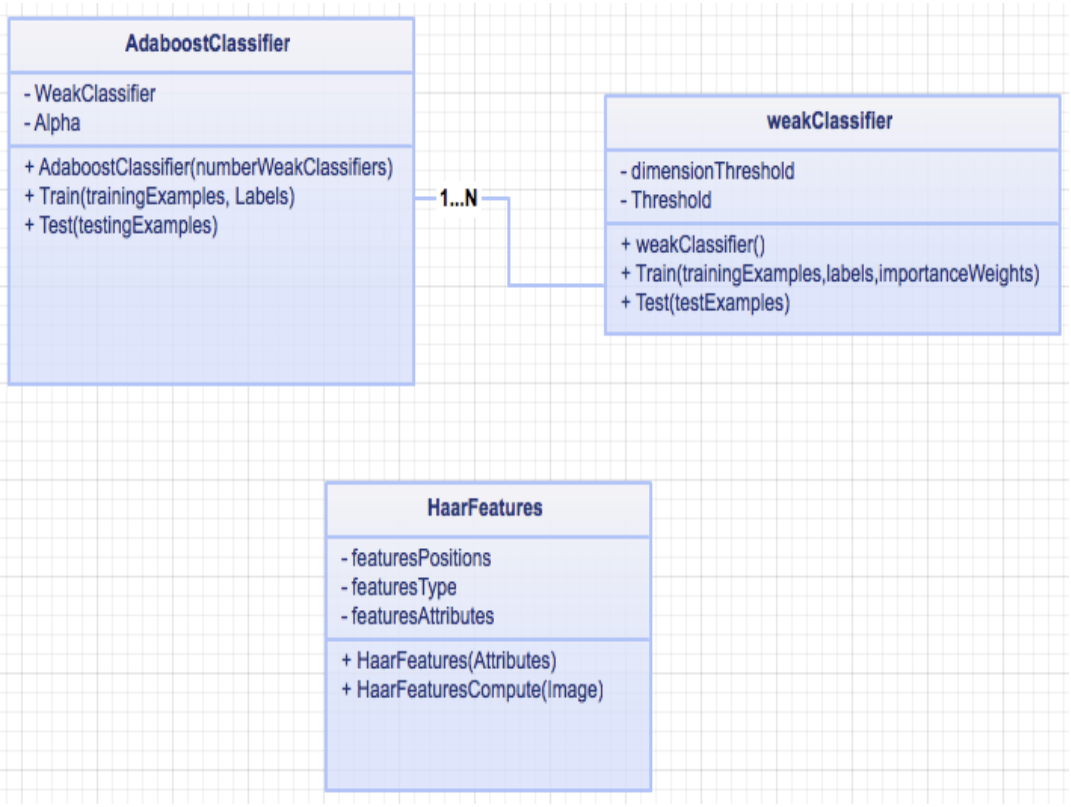

Figure 1: Suggested class diagram.

  - Rectangle 4: $[(r+winHeight/2) \quad (c+winWidth/2) \quad (winWidth/2-1) \quad (winHeight/2-1)]$
  - Feature response: $Rectangle1 - Rectangle2 + Rectangle3 - Rectangle4$

  The attribute *classifier's number* of each weak classifier indicates which is being used.

- Write a function which uses the loaded weak classifiers for detecting faces. Use the sliding window technique and detect faces in the test images. The classification output is based on thresholding in this exercise too.

  – The default detector search window is 19×19 for current dataset.

  – The rounds of the boosting algorithm during the training were 50.

  – The threshold for a positive output t from a classifier is:

    - $(mean - abs(mean - minPos) * (R-5)/50) <= t <= (mean + abs(maxPos - mean) * (R-5)/50)$.

  – Use the pyramid model for detecting faces in different scales.

```
classdef test
    %TEST Summary of this class goes here
    %   Detailed explanation goes here

    properties
        attribute1;
        attribute2; %this is a vector
        attribute3; %this is a matrix
    end

    methods
        function obj=test(init)
            obj.attribute1=init(1);
            obj.attribute2=init*100;
            obj.attribute3=zeros(30,30);
        end

        function obj=train(obj,traindata,label)
            %blabla
            obj.attribute1=1000;
        end

        function obj=test(obj,testdata)
            %blabla
        end
    end

end
```

Figure 2: Simple class implementation example in Matlab(TM).