Exercise for MA-INF 2213 Computer Vision SS16
06.05.2016
Submission on 25.05.2016
Boosting

Boosting is an effective way to combine several weak classifiers to a single strong classifier. In this exercise, you are going to implement a simple boosting algorithm and evaluate it for classification and tracking.
Write your code in C++. You can use the provided framework and fill in your code. If you prefer to write your own classes from scratch, provide a Makefile. Please **comment** your code.

1. **Viola & Jones face detector**:

   A popular example of boosting is the Viola & Jones face detector [1]. Use OpenCV to detect the faces in `img1.jpg`, `img2.jpg`, and `img3.jpg` using the model `face-model.xml`. Visualize your result by drawing a rectangle around each detected face. Output the detection results in a file 'result-faceDetector.txt'.

   (2 Points)

2. **Discrete AdaBoost**:

   (a) In the lecture, *Discrete AdaBoost* (see Appendix) has been presented. Implement the algorithm for a two-class problem. As weak classifier, use a decision tree stump, i.e. a one-level decision tree that splits the data only once, resulting in two leaves that represent a partition of the training set. Each of the leaves represents a class. Choose the split attribute, the split point, and the class label for each leaf (either $c_{\text{left}} = 0, c_{\text{right}} = 1$ or vice versa) of the stump such that the error rate on the (weighted) training examples is minimized. Use the "$<$"-relation for splitting. Evaluate the performance of your implementation. To this end, train a cascade of k weak classifiers, $k \in \{1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 50\}$, and plot the accuracy on the train set (`splice.train`) and test set (`splice.test`) as a function of k and store the values in the file 'result-adaBoost.txt'.

   (10 Points)

   (b) What effect do you observe?

   (1 Point)

3. **Tracking**:

   Boosting can be used to track an object in a video sequence. Your task is to track Nemo while he swims lonely in the deep blue ocean. Find a sequence of 32 frames in the directory `nemo/`. Nemo is annotated in the first ten frames. Annotations found in `frames.train` should be used for training. You can test your tracker on the remaining frames. For details on the file format, see the `README`.

Implement a tracker based on your implementation of Discrete AdaBoost. The training procedure should follow these steps:

- *Generate positive training examples:* use the function *loadTrainFrames* to load the frames defined in `frames.train` as gray-scale images and use a window of size 121×61 around each frame's reference point as positive example
- *Generate negative training examples:* create four negative examples (also patches of size 121×61) per frame (located on top left, top right, bottom left, and bottom right of the reference point)
- *Feature representation:* for each patch, compute a 256-dimensional gray-scale histogram (so you have 256-dimensional examples with label 0 (negative) or 1 (positive))
- *Training:* train Discrete AdaBoost on these examples

In order to track the object (Nemo), follow these steps:

- *Generate test samples:* use the function *loadTestFrames* to load the frames defined in `frames.test` as gray-scale images. For each frame, search for the object in a patch of size 61×61 around the position of the object in the previous frame (for the first frame, use the starting position given in the test file)
- *Compute result:* for each point within this window, compute the confidence (see Appendix) of finding the object at this location and take the point with the highest confidence as new object position

Display each frame after you hypothesized the object position and draw a rectangle of size 121×61 around the hypothesized object position. For each test frame, output the frame number and the detection result in a file 'result-tracking.txt'.

(7 Points)

References

[1] P. Viola, M. Jones., *Rapid object detection using a boosted cascade of simple features*. Conference on Computer Vision and Pattern Recognition, 2001.

Appendix

Algorithm: Discrete AdaBoost.

Input: sequence of N labeled examples $\{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$ with labels $c_i \in \{0, 1\}$, number of iterations T
Initialize the weight vector: $w_i^1 = 1/N$ (defines a distribution over the training examples)
Do for $t = 1, 2, \ldots, T$

1. train the weak learner h_t, providing it with the training data $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$ and distribution $\mathbf{w}^t$
2. calculate the error of h_t: $\epsilon_t = \sum_{i=1}^{N} w_i^t \mathbb{I}(h_t(\mathbf{x}_i) \neq c_i)$
3. set $\beta_t = \epsilon_t/(1 - \epsilon_t)$
4. update the weights: $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(\mathbf{x}_i)-c_i|}$
5. renormalize the weights, such that $\sum_{i=1}^{N} w_i^{t+1} = 1$

Classification: Given $\mathbf{x}$, select $c = \underset{k}{\operatorname{argmax}} \sum_{t=1}^{T} \left(\log(\frac{1}{\beta_t})\right) \mathbb{I}(h_t(\mathbf{x}) = k)$
Note:
The term $\sum_{t=1}^{T} \left(\log(\frac{1}{\beta_t})\right) \mathbb{I}(h_t(\mathbf{x}) = k)$ can be seen as a confidence. It is a weighted vote of each weak classifier for class 1.