Exercise for MA-INF 2213 Computer Vision II SS18
04.05.2018
Submission on 16.05.2018
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 on a small subset of the `splice` dataset. Eventually, you are going to use your boosting algorithm for tracking.
Complete the provided C++ template, it should compile with the provided 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`. You do not need to train the detector. Instead, use the model `face-model.xml`. Visualize your result by drawing a rectangle around each detected face.

   (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.
   (9 Points)

   (b) 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. What effect do you observe?
   (2 Points)

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/`. The first ten frames are annotated and can be used for training. You can test your tracker on the remaining frames. For details on the file format, see the `README`.

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

- load the frames defined in `frames.train` as gray-scale images
- use a window of size 121×61 around each frame's reference point as positive example
- 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)
- for each patch, compute a 256-dimensional gray-scale histogram (so you have 256-dimensional examples with label 0 (negative) or 1 (positive))
- train Discrete AdaBoost on these examples

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

- 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)
- for each point within this window, compute the confidence (see Appendix) of finding the object at this location
- 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.

(6 Points)

(b) Create the negative examples so that they do not overlap with the window of the corresponding positive example. Train a cascade of 50 weak classifiers and have a look at the performance of the tracker on the test frames. Now, create the negative examples so that they strongly overlap with the window of the corresponding positive example (e.g. 50% overlap). Again, train a cascade of 50 weak classifiers and compare the performance of the resulting tracker to the previous case. Which performs better? Why?

(1 Point)

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.