ECE763 Computer Vision: Models, Learning and Inference (including Deep Learning)

Homework 03 and Project 02

Due (see moodle)

(Grades: 100 points in total)

Depart. of ECE, NC State University

Instructor: Tianfu (Matt) Wu

NC STATE UNIVERSITY Electrical and Computer Engineering

0

Grading Policy

- **Homework**: 7% x 5 = 35%
- **Projects**: 60% + [5% bonus]
  - Project 1 (10%) & 2 (20%): **work individually [not required if you work on the thrust 1 and 2 in the next slide)**
  - Project 3 (course project): 30%, **a group (<= 3 members) is allowed [for all thrusts]**
    - You can propose your own projects and work on it once agreed by both of you and me.
    - Requirement: final write-up (15%) and self-contained reproducible code (15%)
    - Bonus points (5%): for top-3 winner of the final project
- **Attendance**: 5% (via in-class moodle quiz, 10min before and 15min after class, from the 2nd week)
- **Late policy**
  - 5 free late days – use them in your ways (counted using 0.5 unit, <=6 hours as 0.5 late day, otherwise 1 later day)
  - Afterwards, 25% off per day late
  - Not accepted after 3 late days per HW and Project 1 & 2
  - Does not apply to Project 3 (no late assignments allowed).
  - We will **NOT** accept any replacement of submission after deadline, even if you can show the time stamp of the replacement is earlier than the deadline. So, please double-check if you submit correct files.
- **Other policies**: please see details in the syllabus.

1

1

Discussions on Projects

- Thrust I: If applicable, identify problem(s) in your own field that have images/videos generated and to be analyzed, and collect dataset(s) (e.g., publicly available ones, or from your own research)
  - Formulate the problem(s): e.g., input-output mapping, target platforms (edge device or workstation/server), evaluation metrics
  - Find a few references in top-tier conferences/journals
  - Our goal: to have a submission ready by the end of this class, through customized project 1-3 design
- Thrust II: If no specific domains of interest,
  - Check https://paperswithcode.com/
  - Find problem(s) that interest you
  - Survey SOTA methods therein
  - Our goal: to have a submission to top-tier CV conference ready by the end of this class, through customized project 1-3 design
- Thrust III: If both do not work for you, **you can work on this one!**
  - We will have standard project assignments.

2

2

Discussions on Projects

- If you decided to work on the first two Thrusts,
  - Prepare your proposal in slides, and discuss with me (during office hours or other appointments), we need both verify the scope and milestones for different sub-projects.
    - Criteria: the workload >= the standard project
  - You can work in a team with no more than 3 members in total.
    - Every member should have clear contributions and the individual workload >= the standard project
  - You can start with project 1 if you can catch it; if not, if not, you can work on the standard assignment, and then you can still work on the first Thrusts from project 2 or 3.
    - The deadline is universal, no matter which thrust you will work on.

3

3

Instructions and Notes

- *How to submit your solutions*: put your report (word or pdf) and results images (.png) if had in a folder named [your_unityid]_xxx (e.g., twu19_ proj02), and then compress it as a zip file (e.g., twu19_ proj02.zip). Submit the zip file through **moodle**.

- *If you miss the deadline and still have unused late days, please send your zip file to TAs and me.*

- **Important Note**: We will **NOT** accept any replacement of submission after deadline ([+late days you use]), even if you can show the time stamp of the replacement is earlier than the deadline. So, **please double-check whether you submit correct files**.

Adaptive Boosting (AdaBoost) for Face Detection

- HW 03 and Project 02 are coupled together. The former focuses on theoretical aspects, while the latter focuses on implementation. Refs:
  - P.Viola, M.Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *CVPR* 2001.
  - Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154. https://www.merl.com/publications/docs/TR2004-043.pdf DOI:10.1109/CVPR.2001.990517
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html
  - https://scikit-image.org/docs/dev/auto_examples/applications/plot_haar_extraction_selection_classification.html
- 2-Class AdaBoost
  - Input
    - Data: A set of labeled training data from 2 classes (e.g., the positives, face and the negatives, non-face image patches used in Project 1)

$$D_{train} = \{ (x_i, y_i): x_i \in R^d, y_i \in \{+1, -1\}, i = 1, 2, \cdots, N \},$$

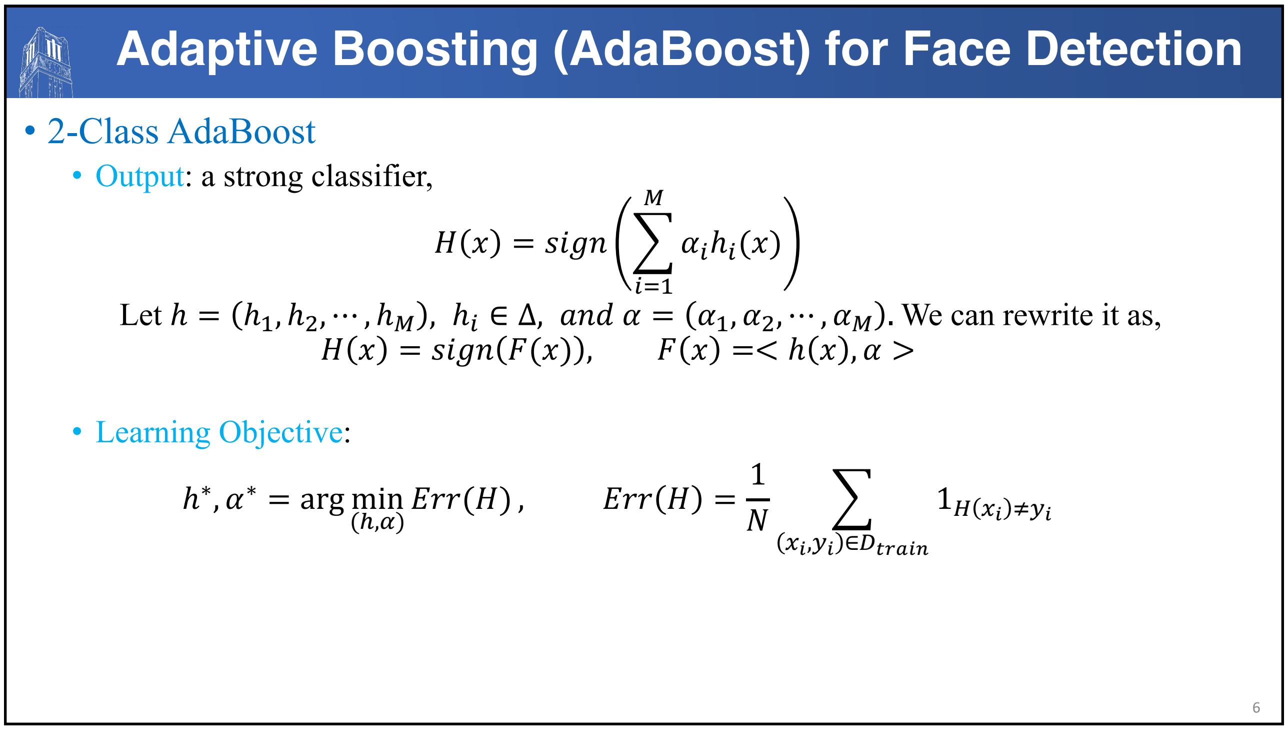
where x_i represents, e.g., 20×20 gray images for face and non-face respectively (e.g., those in Project 01).

    - Weak Classifiers: A set of weak classifiers (see an example in slide 8),

$$\Delta = \{h_t(x): t = 1, 2, \cdots, T\},$$

where $h_t(x)$: $\mathrm{R}^d \rightarrow \{+1, -1\}$, which is said to be a weak classifier because its error rate on D_{train} is slightly better than random guess (0.5 for a 2-class task),

$$\epsilon(h_t) = \frac{1}{N} \sum_{(x_i, y_i) \in D_{train}} 1_{h_t(x_i) \neq y_i} < \frac{1}{2}, \text{ where the indicator function } 1_z = \begin{cases} 1, if\ z\ is\ true \\ 0, otherwise \end{cases}$$



Adaptive Boosting (AdaBoost) for Face Detection

- 2-Class AdaBoost
  - Output: a strong classifier,

$$H(x) = sign\left(\sum_{i=1}^{M} \alpha_i h_i(x)\right)$$

Let $h = (h_1, h_2, \cdots, h_M),\ h_i \in \Delta,\ and\ \alpha = (\alpha_1, \alpha_2, \cdots, \alpha_M)$. We can rewrite it as,

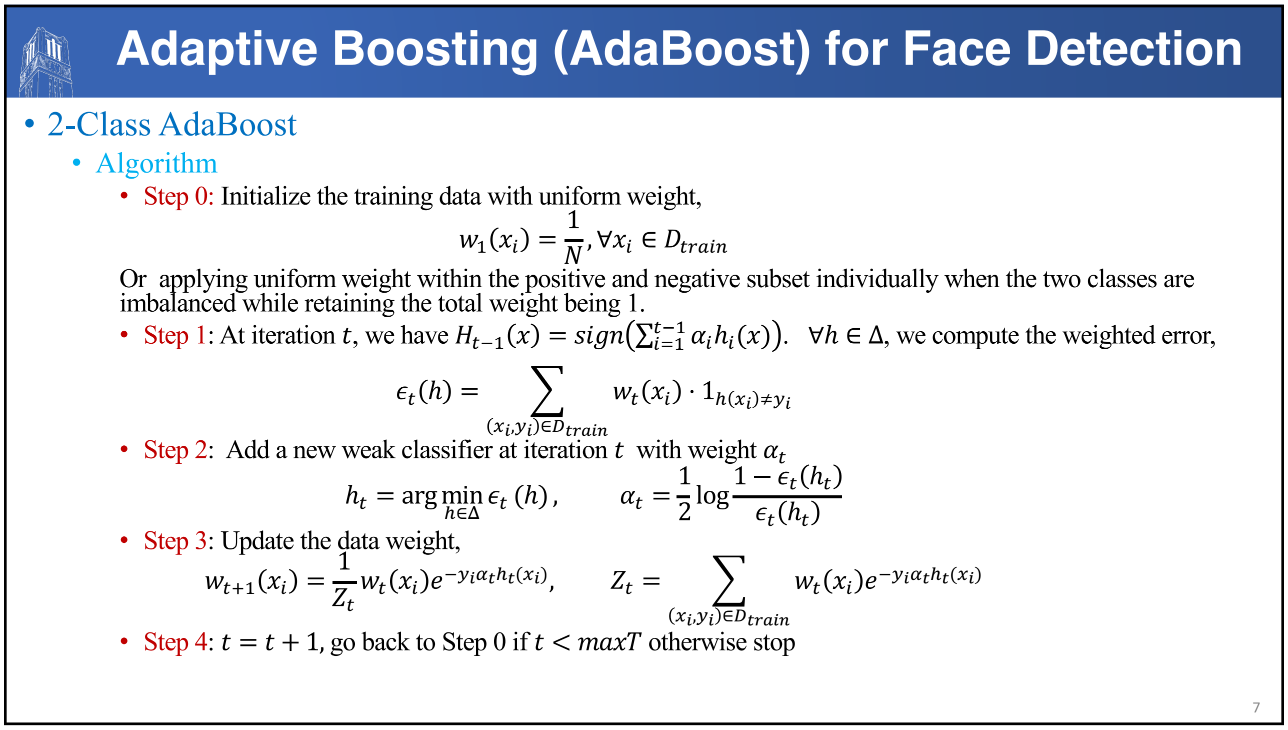
$$H(x) = sign(F(x)), \qquad F(x) =< h(x), \alpha >$$

  - Learning Objective:

$$h^*, \alpha^* = \arg\min_{(h,\alpha)} Err(H), \qquad Err(H) = \frac{1}{N} \sum_{(x_i,y_i)\in D_{train}} 1_{H(x_i)\neq y_i}$$

6

6



Adaptive Boosting (AdaBoost) for Face Detection

- 2-Class AdaBoost
  - Algorithm
    - Step 0: Initialize the training data with uniform weight,

$$w_1(x_i) = \frac{1}{N}, \forall x_i \in D_{train}$$

Or applying uniform weight within the positive and negative subset individually when the two classes are imbalanced while retaining the total weight being 1.

    - Step 1: At iteration t, we have $H_{t-1}(x) = sign\left(\sum_{i=1}^{t-1} \alpha_i h_i(x)\right)$. $\forall h \in \Delta$, we compute the weighted error,

$$\epsilon_t(h) = \sum_{(x_i,y_i)\in D_{train}} w_t(x_i) \cdot 1_{h(x_i)\neq y_i}$$

    - Step 2: Add a new weak classifier at iteration t with weight α_t

$$h_t = \arg\min_{h\in\Delta} \epsilon_t(h), \qquad \alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}$$

    - Step 3: Update the data weight,

$$w_{t+1}(x_i) = \frac{1}{Z_t} w_t(x_i) e^{-y_i \alpha_t h_t(x_i)}, \qquad Z_t = \sum_{(x_i,y_i)\in D_{train}} w_t(x_i) e^{-y_i \alpha_t h_t(x_i)}$$

    - Step 4: $t = t + 1$, go back to Step 0 if $t < maxT$ otherwise stop

7

7

- Step 3:
  - $w_{t+1}(x_i) = \frac{1}{Z_t} w_t(x_i) e^{-y_i \alpha_t h_t(x_i)}$, $Z_t = \sum_{(x_i, y_i) \in D_{train}} w_t(x_i) e^{-y_i \alpha_t h_t(x_i)}$
- From the recursion, we know
  - $w_t(x_i) = \frac{1}{Z_{t-1}} w_{t-1}(x_i) e^{-y_i \alpha_{t-1} h_{t-1}(x_i)}$, we can use it to rewrite $w_{t+1}(x_i)$ and have,
  - $w_{t+1}(x_i) = \frac{1}{Z_t} \left[\frac{1}{Z_{t-1}} w_{t-1}(x_i) e^{-y_i \alpha_{t-1} h_{t-1}(x_i)} \right] e^{-y_i \alpha_t h_t(x_i)}$

$$= \frac{1}{Z_t \cdot Z_{t-1}} w_{t-1}(x_i) e^{-y_i [\alpha_t h_t(x_i) + \alpha_{t-1} h_{t-1}(x_i)]}$$

  Then, we can recursively apply the recursion, and we will get

$$w_{t+1}(x_i) = \frac{1}{Z_t \cdot Z_{t-1} \cdots Z_1} w_1(x_i) e^{-y_i \sum_{k=1}^{t} \alpha_k h_k(x_i)} = \frac{1}{Z} \frac{1}{N} e^{-y_i \sum_{k=1}^{t} \alpha_k h_k(x_i)}$$

- We also know $\sum_{i=1}^{N} w_{t+1}(x_i) = 1$, from which we can derive the Z

8

8

Adaptive Boosting (AdaBoost) for Face Detection

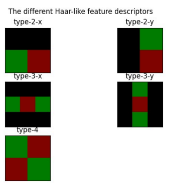
- HW03:
  - Problem 1 [30 points]: Let $z = z_t \cdot z_{t-1} \cdot \cdots \cdot z_1$. Please derive the formula of z.
  - Problem 2 [30 points]: For $H_t(x) = sign\left(\sum_{i=1}^{t} \alpha_i h_i(x)\right)$ learned via the aforementioned algorithm, Please show $Err(H_t) \leq Z$ (derived in problem 1). So, the magic of the AdaBoost algorithm is to minimize the error upper bound (since minimizing $Err(H_t)$ is a combinatorial problem).
  - Problem 3 [40 points]: Please show why $\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)}$ (step 2 in the AdaBoost Algorithm).

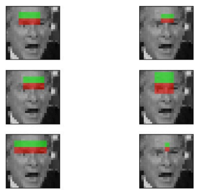
9

9

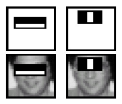
Adaptive Boosting (AdaBoost) for Face Detection

- Project 2.1 (90 points):
  - Data: Reuse the face /non-face data in Project 1. Recommended resolution will be 20 by 20 (gray images). You may want to extract a much large number of positive and negative images.
  - Feature vector: Use Harr feature.
    - Consider the 5 types of Harr features





- The value of the descriptor is equal to the difference between the sum of the intensity values in the green rectangle and the red one. The red area is subtracted to the sum of the pixel intensities of the green.
- As the figure on the right shows, in practice, the Haar-like features will be placed in all possible location of an image and a feature value will be computed for each of these locations.
  - For each of the 5 types, define the minimum size and the maximum size of the rectangle in instantiation, e.g., for a type-2-x Haar, the minimum size could be 1x2 pixels and the maximum size will HxW (the entire image patch such as 20x20). Each instantiation can be placed at all location inside the image patch (i.e. sliding).
  - You can define the sizes accordingly so the total number of features are not too large, and the selected features can cover potentially discriminative regions of faces, eg.,



10

10

Adaptive Boosting (AdaBoost) for Face Detection

- Project 2.1 (90 points):
  - Weak classifiers: Based on the extracted Harr features.
    - Compute the value of each instantiation of each type of Harr features for each sample. *Each feature corresponds to a weak learner.* Denote by $f_t(x_i)$ a feature t on an image x_i. A weak learner is thus defined by,

$$h_t(x_i) = sign(f_t(x_i) - \tau_t) \text{ -- see figure 1}$$
$$\text{Or } h_t(x_i) = sign(\tau_t - f_t(x_i)) \text{ -- see figure 2}$$

  - Where the threshold τ_t is learned using face samples and non-face samples for the weak learner.

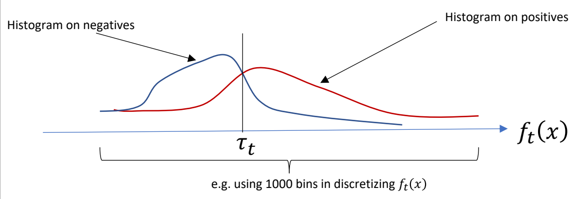


Figure 1

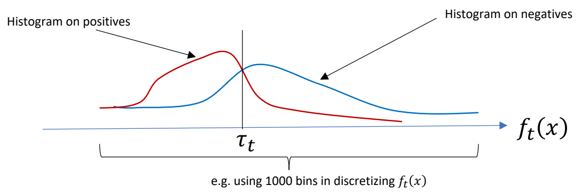


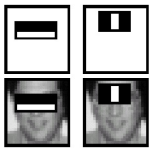
Figure 2

11

11

Adaptive Boosting (AdaBoost) for Face Detection

- Project 2.1 (90 points):
  - Task 1 (10 points): Calculate the classification error for each weak learner and draw the best ten features (before boosting) as Figure 3 in Ref [1].



  - Task 2 (70 points): Implementing the Adaboost Algorithm (also see Table 1 in [1]) and test it in the test dataset you created. Plot the ROC based on thresholding $H(x)$.
  - Requirement:
    - You can use off-the-shelf code to extract and visualize Haar features, and to plot the ROC.
    - You need to implement Task 1 and 2 from scratch.

[1] P.Viola, M.Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *CVPR* 2001.

Adaptive Boosting (AdaBoost) for Face Detection

- Project 2.1 (90 points):
  - [Optional] Task 3 (10 bonus points): Implement the cascade version as in
    - Viola, Paul, and Michael J. Jones. “Robust real-time face detection.” International journal of computer vision 57.2 (2004): 137-154. https://www.merl.com/publications/docs/TR2004-043.pdf DOI:10.1109/CVPR.2001.990517
  - [Optional] Task 4 (10 bonus points): Implement a hard-negative mining module to boost the performance.
    - If the negative image (non-face) patches that you sampled are not “broad” enough, the learned classifier may not be strong enough too.
    - A hard-negative mining process is to evaluate your current classification in a set of negative images (big images, not small patches) in a sliding window fashion, and use the detected false positives as the hard negatives.
    - Then you retrain your classifier using the hard-negative augmented negative samples.
    - You can repeat this procedure for some predefined number of runs, or until you can not find any false positives in the set of negative images.

Adaptive Boosting (AdaBoost) for Face Detection

- Project 2.1 (90 points):
- Hint:
  - *Start small.* Write a program using only a small number of weak classifiers – like the two illustrated above. Train and test the program on a small set of face and non-face images. At every stage, check to make sure that your results make intuitive sense. For example, see if your program selects the same first two features that Viola found (see above).
  - Be very careful about how your program scales with the number of weak classifiers and the number of face/non-face images. (This will depend on the details of how you write the code and the type of computer you are using). If your program scales badly, then only use a limited number of weak classifiers and a small number of faces/non-face images.
  - The classic mistake for this type of problem is to write a large amount of code and then try testing it with 50,000 weak classifiers on all the face and non-face images. Check the code first with only a few classifiers – so that you have a good guess of what classifier should be selected. Then try scaling up – and stop if the program scales badly or ceases to give intuitive results.

Refactoring Codes of Project 1

- Project 2.2 (10 points):
  - Go through the provided reference code framework of Project 1 (to be discussed in class too)
  - Refactor your own code in the reference code framework.
  - Provide a summary of your observations of the refactoring process in terms of coding.