Face Detection using Adaboost and Classifier Cascades

Connor Scott

Christian Guardiola

Samuel Sidelinger

CS4379C Computer Vision

Dr. Metsis

11/30/2021

Introduction

Abstract:

In this project we attempt to create an architecture that can detect faces utilizing AdaBoost, skin detection, bootstrapping, and classifier cascades. Adaboost tweaks a set of weak classifiers until they converge into a strong classifier. Skin detection is used on an image to remove non-skin pixels which increases the accuracy of our face detector. Bootstrapping helps facilitate training by identifying false positive subwindows from larger images in our training set and adding them into the training set. Classifier cascades are used to improve true face detection accuracy while retaining efficiency by creating multiple layers of classifiers where each one is stronger than the previous. These weak classifiers are also known as Haar filters and when given an offset, we can extract features and determine how good of a classifier it was by using the integral of the face.

Challenges:

Of the four key components that make up the face detector, classifier cascades was the most difficult to understand conceptually due to the complexity of the cascade architecture. However, we were able to understand the cascade architecture fully by analyzing its pseudo code deeply, until we were able to implement it in code. Also, adding a bootstrapping component within classifier cascades posed another challenge for us codewise, as there was no straightforward guide on how to concatenate the two components together. So, we had to get creative with how we completed this task by fully understanding bootstrapping.

Assumptions Made:

Assumptions made in the project are that when training images, all face images are centered in the frame and that the training images are 100x100 pixels. Also, when testing images, it is assumed that the faces are within a certain scale of the face window we chose. For example, if a face is 200 pixels wide, we should adjust our detection scale manually to match our classifier face size, which is 63x57 pixels. We also assume that rotation is not needed, meaning the person’s face is relatively aligned with the camera.

Data preparation/preprocessing

The training dataset given to is what we used to train our classifiers. This data set included 3047 cropped face images and 130 non face images of varying sizes, a total of 3177 images. The cropped faces were of size 100x100, grayscale, and in bmp file format. The non face images were of various sizes, color, and of the jpg format.

When we started training our classifiers, we chose to leave the training images as they were, which was a 100x100 grayscale image. This gave us problems later, meaning we had highly inaccurate results of about 2% accuracy. After getting bad results when creating simple classifiers, we decided we needed to crop the faces to remove the less important features such as hair or even background. The cropped face dimensions are 63x57, which gave us more accurate results with basic boosted classifiers. Because we chose 63x57 as our face size window, we must also send this size when detecting the location of the subface.

We started with 50 faces and 100 non faces, which were chosen because it is a decent number of face images to build classifiers from and build more from when bootstrapping. Also, this is a 1:2 ratio for total faces to non-faces, which was chosen as a face to non face ratio as recommended by Dr. Metsis. Because the non face training images were of varying sizes, it was decided that random subwindows in the image would be chosen. After teasing, we decided on using 5 random subwindows for each non face image. So, with 50 face images, there are 20 non face images with 5 subwindows each image. Also, we did not train our classifiers over colors so we read in the non face images as grayscale images even though they were provided in color.

Face Detection Architecture Components:

As stated before, our face detection architecture consisted of 4 main components. These components are AdaBoost, Bootstrapping, Classifier Cascades, and skin detection. AdaBoosting consists of creating soft/boosted classifiers from a set of weak classifiers. Bootstrapping helped create more accurate boosted classifiers by adding outlier training images into the training set. Classifier Cascades uses layers of classifiers that start with simple feature classifiers which have a high detection rate and false positive rate classifier that will remove obvious non faces, and any detected subwindow will pass through to complex classifiers that handle more features and are

slower but more accurate. Adding skin detection is meant to help remove any non-skin pixels within a sample image, which increases the accuracy of our face detector.

AdaBoost

Training Classifiers for AdaBoost

Before we can run the AdaBoost algorithm, a set of weak classifiers must be made. To do this, a random rectangle filter is created with a random offset on the image. A rectangle filter is used to extract information about the feature, which helps us classify a face later. Once a set of weak classifiers is created, we can then call the AdaBoost function.

At first, prior to implementing classifier cascades we chose to create 1000 weak classifiers, but ended up with few good strong classifiers later. We found that 10000 weak classifiers was not time consuming but gave many possible weak classifiers that collected face features well.

AdaBoost Overview

AdaBoost takes in a set of weak classifiers and finds the classifiers that extract the best data between all the image integral's. As stated before we chose 50 face images and 100 non face images to train our classifiers. AdaBoost takes in this data and returns a set of boosted classifiers, also known as soft classifiers. At first we chose 15 boosted classifiers because after testing we found this number was both more accurate compared to 5 boosted classifiers and also did not take much time to detect faces as well as generating these boosted classifiers. This changed later when we implemented a more complex architecture.

The thresholds assigned by AdaBoost represent the threshold value of the rectangle filter over the that gave the best result over multiple images. The alpha value assigned to a classifier from AdaBoost represents whether or not that classifier is detecting a face or not depending on if it is positive or negative. The larger the alpha value on a strong classifier, the better it is at detecting if an image is a face or non face.

Results

Using the 15 boosted classifiers from AdaBoost and several training and testing runs, the most accurate results achieved were a 9.73% false negative and a 7.21% false positive, meaning there was a 90.2% face classification accuracy and a 92.8% non face

classification accuracy with an overall classification accuracy of 91.5%. It took only 0.0036 seconds to classify one test face image. To classify all 770 test face images, it took 0.0731 seconds. To classify all images, which includes the 770 test faces and the 36 test non faces it took only 0.0898 seconds.

While AdaBoost alone does partially detect a face, this is not accurate

Bootstrapping

Bootstrapping within AdaBoost

To implement bootstrapping, we tested the boosted classifiers from AdaBoost on all training faces. If any of the training faces were wrongly classified, that face image was added to the faces matrix that is used to create weak classifiers. We performed bootstrapping in train.m where we look at the results of the boosted classifiers and add false negatives into the matrix of faces we create weak classifiers from.

In addition to images being added, the top 5 boosted classifiers were kept and added back into the weak classifiers cell array. The top 5 number of boosted classifiers increased by 5 each time AdaBoost was looped through with bootstrapping. This was useful because it helped attain much higher accuracy, but took a very long time to compute. It took a long time because bootstrapping was done until a certain level of accuracy was attained or until the accuracy stopped improving.

Results

Using bootstrapping increased accuracy at the cost of time when training. Some training sessions took over 30 minutes. When still using 15 boosted classifiers from AdaBoost but with bootstrapping, we got a 7.3% false negative rate and a 4.8% false positive rate. This means there was a 92.7% face classification rate and a 95.2% non face classification rate. Compared to previous results from AdaBoosting alone, this is more accurate at both face classification and non face classification.

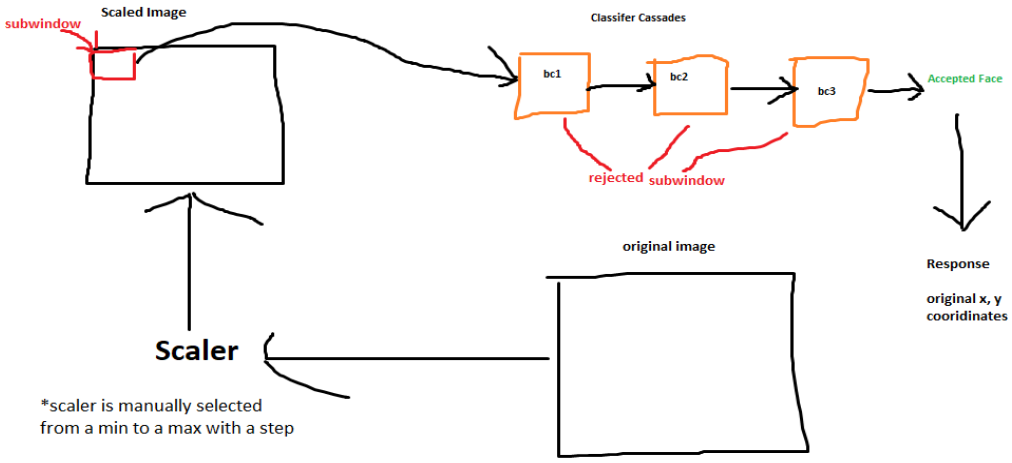
Classifier Cascades

Classifier Cascades using AdaBoost

Classifier cascades are used to improve true face detection accuracy while retaining efficiency by creating multiple layers of classifiers where each one is stronger than the previous. The level of classification was dependent on how many strong classifiers we trained until our false positive rate was below or equal to our target false positive rate. Each boosted classifier has at least weak classifiers, or 2 features. The maximum acceptable false positive per layer initialized at 0.5, the minimum acceptable detection rate per layer is set to 0.5, the false positive target is 0.01 and the threshhold_decrease value is set to -1000. These values and rates were chosen to create a stable classifier cascade.

To accommodate classifier cascades we do not create a defined number of weak classifiers, but rather this number is based on the amount of training steps that we take for each layer of the classifier until a layer has reached its target false positive rate. It's important to note that it is possible for the classifier cascades to reach its target false positive rate with few classifiers. If a loew testing accuracy is met, training with more classifiers solves this problem.

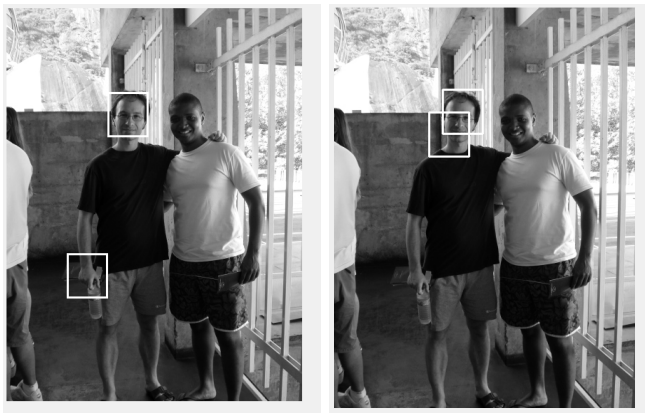
When a boosted classifier has been created from Adaboost, the boosted classifier is evaluated with a set of faces and nonfaces and we receive two values, false positive rate and detection rate, which we use to continuously train the current classifier until the target false positive rate has been reached.



Drawn image used to understand how Classifier Cascades will work

Results

With classifier Cascades, our results were much more accurate than before. We had a false negative rate of 8.2% and a false positive rate of 0%. This means there was a 91.8% face classification rate and a 100% non face classification rate. Compared to results from bootstrapping this is much more accurate at classifying non faces, but can be improved on when classifying faces. It is possible this can be more accurate with a different set of classifiers.



Detection using classifier cascades

Skin Detection

Testing Faces with Skin Detection

We implemented skin detection when testing the images to make it more accurate and faster. By using skin detection, we limit the possible area where we can use our boosted classifiers. To do this, a skin detector receives a grayscale image and utilizes skin histograms to detect the skin pixels of that image and returns an image with logical values that are 1 where skin pixels were located and 0 where they were not. We multiply the original grayscale image with the logical valued image and this keeps the pixels with their original intensities where logical values were 1 within the logical valued image.

Results

By adding the skin detection component to our face detection algorithm, our classification accuracy is 5.3% false negative and 0% false positive, this means we had a classification accuracy of 94.7% for faces and 100% for non faces. This did not change the runtime for classification because it did not affect the areas in which we apply the classifiers. Compared to previous results from classifier cascades without skin detection, this is more accurate at both face classification and non face classification.

<table>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>Skin detection and Classifier Cascades with 2 and 3 results respectively.</td>
<td>Skin detection and Classifier Cascades with 2 results, different set of classifiers</td>
</tr>
</table>

Important Functions

Overview of train.m:

We load in 50 faces and 20 nonfaces so that when we split the nonface images into 100 subwindows, the ratio between faces and nonface subwindows is 1:2.

We use several constants for classifier cascades based on the cascade architecture, which include false positive rate, detection rates, number of classifiers per layer, index for the current layer, the maximum acceptable false positive rate per layer, the minimum acceptable detection rate per layer, target overall false positive rate, threshold decreaser, and a cell to store all classifiers from each layer. The classifier cascades returns a cell which stores all classifiers from each layer, and then saves variables into files which are used for testing.

Overview of test.m:

To test the given data sets, we read in all the images into respective data structures. This includes 20 color photos that include a varying number of faces, 770 cropped face photos, and 36 non face images. The 20 color photos with faces are used when detecting the location of a face while the other 806 faces are used when classifying whether the image is a face or not.

When testing the color images and detecting the location of the face, we first start changing the color image into a grayscale one and use skin detection to keep the skin pixels along with their intensities values. Then we will break up our image into multiple subwindows with face size at every pixel in the full image unless that pixel creates a subwindow that goes out of bounds from our full image.

When simply classifying whether an image contains a face or not, each image is sent through the boosted predict and we check to see if the returned value is greater than or less than zero. We check if it's greater or less than because a positive number means the image was classified as a face and a negative number means the image was classified as a non face.

Conclusion: Summarize your observations.

All 4 components to our face detection architecture are important. Using AdaBoost alone was fast, but was not very accurate since the boosted classifiers were random and might not have accurate classifiers. Implementing Bootstrapping fixed this issue, but increased runtime. Classifier Cascades allowed for better classifiers, but also took some time. Skin detection also increased the testing classification accuracies.

We resulted with a classification accuracy of 5.3% false negative and 0% false positive. However, it is important to note that the classification accuracy can still increase with a different set of classifiers if the training function is run again. But, with our current set, we had a classification accuracy of 94.7% for faces and 100% for non faces, which is an average overall classification accuracy of 97.4% accuracy. It can be argued that these results are accurate, but could be improved on if we implemented other styles of detection that have been previously learned in this class. If we wanted this to be more accurate, we could extend the skin detection architecture to limit the subwindows that are sent through the boosted predict function as well as when detecting the area the face is in. This would not only make it faster, but more accurate as well, but we would likely have to change the way we train our classifiers if we do this.