18-794
Face Detection Competition

Spencer Barton (sebarton)

December 12, 2014

1 Overview

I decided to implement Viola-Jones. The actual algorithm that I implemented is described in *algorithm.txt*.

2 Modifications from Viola-Jones

I made a number of modifications to the basic Viola-Jones.

One of the main things was changing window size. From the provided data I found the average bounding box had a height to width ratio of about $3 : 2$. In the paper they use a $24x24$ pixel window so I used a $24x16$ window.

The major limitation that I ran into throughout the project was memory on my machine (I did not have success with the Andrew Machines or Red-Dragon as the ensemble learning features are only on Matlab 2014), so I greatly reduced the number of features. In the paper they use $160,000$ features while I use about $5,800$. This enabled me to run many more trials but I was never able to reach the error rates achieved in the paper.

A second major difference was my weak learners in the Adaboost cascade. In the paper they used decision trees of depth 1. In the Matlab implementation boosted trees become computationally unfeasible with a huge number of dimensions so I used learners that could reduce the dimensionality. In the end linear discriminators are very close to the paper's learners since Viola-Jones stuck to using single depth trees (or stumps as they call them).

3 Trials and Tribulations

Once I had a testing framework up and running I was stumped for some time on really bad results. My detector initially classified nearly everything a face on the validation examples. I was mean-centering my data but that wasn't enough. On a closer read of the paper I realized that they also variance normalized the windows. I did the same which helped. The computation actually did not being too cumbersome because the variance could be computed from the integral image and the integral of the image squared.

3.1 Tuning the Weak Learners

I then spent the majority of my time trying to tune the classifier. I started with a single boosted learner instead of a cascade. The default settings were disappointing. I had three main parameters to play with

- **Discrimination Type** The choices here were linear or quadratic (with some additional sub-varieties of each). I began with the linear discriminator but was only able to achieve about .6 positive detection and .2 false detection on my validation data. These two numbers were my metrics of use throughout the project so to be concise I'll use the notation [.6, .2] from here on.

  The quadratic was far superior (about [.8, .05] was the best I got using just a quadratic) but suffered when the data was singular. I eventually discovered the *psuedoQuadratic* which utilizes a psuedo inverse instead of normal inverse which makes it much more robust.

- **Number of Features to Use Per Weak Learner** This was probably the most important perimeter. I spent the most time getting this one down. I had best results with a quadratic classifier with about 30 features. I used 35 features for my final submission. I found that the quadratic discrimination couldn't do any better than the linear until over 20 features were used. Above about 50 there was major over-fitting - zero training error and almost 100% testing error.

- **Number of Weak Learners** This final parameter I never determined an optimal number. I found that at least 2 weak learners were needed to reduce the randomness but beyond that each weak learner had random starting parameters so there was not much error stabilization as weak learners were added. For computation reasons I limited myself to 10 weak learners.

3.2 Tuning the Cascade

- running out of memory - running at 12x8 - quadratic vs linear - varying the number of weak learners - varying the number of parameters - cascade vs single boosted learner - over-fitting - increasing negative examples - PCA failure