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) since both are making a single threshold decision.

3 Trials and Tribulations

Once I had a testing framework up and running I was stumped for some time with 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 gave disappointing results so I had to play with the parameters. 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

After playing around with the single classifier I decided to implement the cascaded classifiers found in Viola-Jones. The key idea behind a cascaded classifier is that each stage can filter out a decent number of false positives so that further computation can be avoided.

The cascade classifier had the same parameters as the single classifier but had a few more considerations.

The main challenge with the cascade was to add more weak learners to an individual stage or to make the whole cascade longer. Additionally I needed to consider the errors to tolerate at each stage.

One of the first things I found with the cascade was that it could over-fit very easily. Since the number of false detection training examples decreased with each stage, after the first stage classifiers began to dangerously over-fit. For my first few runs I was getting zero training error for false detection but performing worse on the validation than the single classifier.

I additionally found that the cascade computation did not always terminate. At each level of the cascade a false detection and positive detection target are set. The number of weak learners and cost of false detection are varied until the two targets are met. I quickly found that I was nowhere near matching the targets in Viola-Jones. In fact when I attempted to achieve a false detect rate of .3 with positive detection of .99 as they did in the paper I was not terminating. Basically given my training data I could not converge on these targets.

To mitigate the termination and over-fitting problems I decided to set maximums for the number of weak learners and overall number of classifiers in the cascade. I was never able to meet the error rates from Viola-Jones of $[.9, .00001]$ (or even come close to the false detection rate - the best I got was $[0.76705, 0.020921$ in terms of getting a low false detection rate). As discussed above have a large number of weak learners did not help particularly so I kept that to about 5 per classifier. Initially I also kept the cascade length down to about 3 classifiers because I was consistently getting zero training error with more classifiers or getting a few false detections that could not be classified.

Initially I was running out of memory with $24x16$ windows with about 73,000 features so I sized down to $12x8$. This worked well but I found in the end that $24x16$ was better. Since I couldn't get all of the features, I reduced the filters by imposing some size constraints. I had found from earlier that the filters that ended up being used were usually on the larger side, aka more than two pixels wide and tall. Also the xor filter was less popular so I made fewer of those. These observations were consistent with the filters that Viola-Jones found to be the best.

3.3 Future Improvements

Overall I found that the algorithm did better as I added more training data. In particular since the cascade eliminates so many negative examples between stages it helped to have more negative examples. If I were to improve this algorithm in the future I would take more time to optimize functions and run on as much data as Viola-Jones. I had 532 positive examples and 6370 negative examples in my final classifier. The paper used 5000 of each.

3.4 Other Attempts

In addition to Viola Jones I attempted a PCA, SVM solution. I utilized the same features as before. Interestingly I found that out of the thousands of features 30 features was enough to have a reconstruction rate of 99%. I quickly realized that this solution was not going to be efficient enough as computing the thousands of features was a computational headache. Since PCA reduced the full set of features I still had to compute every feature. An SVM would likely be a good solution given a smaller feature set.

3.5 Example Cascade Generation

This is an example print out from training a single classifier and then a cascade. The single classifier uses fewer features (350) but all of those filter need to be computed as opposed to the cascade where fewer features are needed at individual stages. The cascade generation shows the common problem of adding weak classifiers with no real impact. D is the detection rate on the training data and F is the false detection rate. At the end the validation shows `D :0.76705 F: 0.020921`. This is also a good example of over-fiting as the training false detection error goes to zero. More training data might help with this problem as would earlier termination of the cascade.

```
Loading non-faces
Generating negative example features
Num pos: 532 Num  neg: 6370
======================
Num of filters used: 350
Ens     D: 0.99432 F: 0.9477
======================
Training cascade f:0.3 d:0.95 Ftarget:0.001
Chose cost:0.8 Di:1 Fi:0.86531
Chose cost:0.8 Di:1 Fi:0.86578
Chose cost:0.8 Di:1 Fi:0.86405
Chose cost:0.8 Di:1 Fi:0.86264
Chose cost:0.8 Di:1 Fi:0.86562
Weak learner added i:1 n:10 D:1 F:0.86562
Chose cost:1.9 Di:0.91541 Fi:0.11697
Weak learner added i:2 n:6 D:0.91541 F:0.10126
Chose cost:1.6 Di:0.87406 Fi:0.18295
Chose cost:1.6 Di:0.87406 Fi:0.17209
Chose cost:1.6 Di:0.8703 Fi:0.17674
Chose cost:1.6 Di:0.87594 Fi:0.15349
Chose cost:1.4 Di:0.8703 Fi:0.13643
Weak learner added i:3 n:10 D:0.8703 F:0.013815
Chose cost:0.8 Di:0.8703 Fi:0.022727
Chose cost:0.8 Di:0.86842 Fi:0.034091
Chose cost:0.8 Di:0.8703 Fi:0.034091
Chose cost:0.8 Di:0.8703 Fi:0.022727
Chose cost:0.8 Di:0.8703 Fi:0
Weak learner added i:4 n:10 D:0.8703 F:0
======================
Num of filters used: 1260
Cascade D :0.76705 F: 0.020921
======================
Training complete
```