Look Further into Logistic Regression and Machine Learning

Qin Haocheng 11911317

May 2, 2022

Abstract

In this paper, we concluded three given paper to make a accuracy comparison for Logistic Regression(LR) and traditional machine learning(ML) models like Naive Bayes(NB) and Support Vector Machine(SVM), and we additionally made a simulation for runtime comparison, then we concluded the dataset they each is suitable for. Based on this, we make a further illusion for the relationship between LR and NB, SVM, also Neural Network(NN). With respect to conclusions above, we then proposed two way to improve existing ML models, one for Boost method and another one for NN.

1 Research Design

Our task is to train a model to classify whether persons on the given data sets are bald. It's a trivial problem, so we further realized the implement to group photos and also videos. To achieve this, we divide the task in to four steps: Bald classification, Face detection, Implement to group photo, Implement to video. To be more detailed, our work is assigned as below:

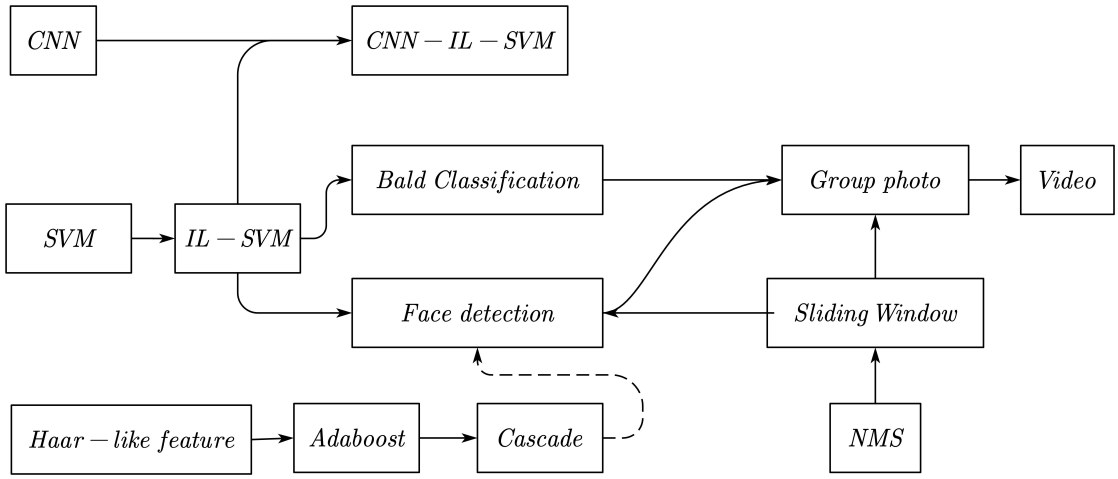


Figure 1: Our work

Almost all these works are based on R, except two things: Implement of Haar-Adaboost model and importing video. The second is because R can't import a video, one should cut a video into frames the deal them with R, and finally write the results frame by frame into a result video.

2 Research Details

2.1 Bald classification

2.1.1 Histogram of Oriented Gradient

HOG is a method for image reduction and feature extraction by statistically calculating the gradient histogram between pixels. It usually consist of following the follows five steps.(1)

Graying picture. When a picture was read by the function readJPEG(), there is a three dimension big RGB matrix. With graying, combine color message into the grey value α which is usually calculated by

$$\alpha = 0.3R + 0.59G + 0.11B$$

After graying picture, the image matrix is reduced from three dimensions to one.

Gamma correction. Gamma correction is often used to reduce the effects of light, by the Theorem $z = \alpha^{1/\gamma}$. when $\gamma > 1$, The overall content of the image looks bright because it removes some of the effects of light, while the $\gamma < 1$ does the opposite.

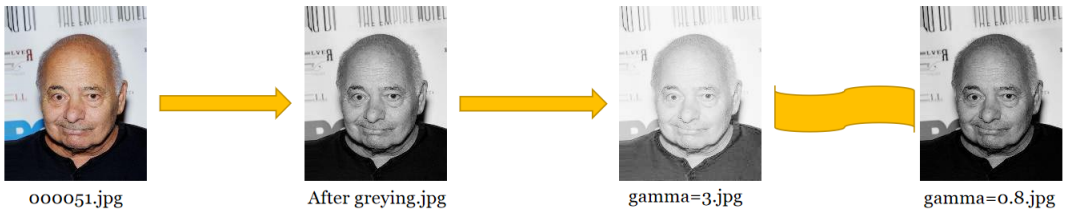


Figure 2: HOG steps

Calculate gradient. For any pixel (x,y) in the picture, its gradient g will relates to horizontal gradients g_x and vertical gradients g_y. Both of two gradients can be calculated with numerical differentiation. So the size and direction of the pixel gradient can be expressed by g_x and g_y.

$$|g| = \sqrt{g_x^2 + g_y^2} \qquad \theta = arctan(\frac{g_x}{g_y + \epsilon})$$

The ϵ here is added to keep the denominator from being 0.

Statistics cells HOG. A cell is a small pieces of the picture, it is usually consist of about 100 pixels. Focus on one cell, we can draw a histogram of oriented gradient by the every cell's gradient's size and direction. For example, We firstly divide 180 degrees into 9 parts, and statistic the value of every angle. Like the pixel (x,y), $|g| = 0.5$, $\theta = 90$, then the 80 and 100 would add $\frac{(90-80)\times 0.5}{(100-80)}$ and $\frac{(100-90)\times 0.5}{(100-80)}$.

Merge cells into blocks. By steps before, all the message in the cells had gathered on the 18 variables in the histogram(HOG's nine sizes and directions). Merge some cells into a block, in the end, the HOG feature can be obtained by merging the information of the blocks.

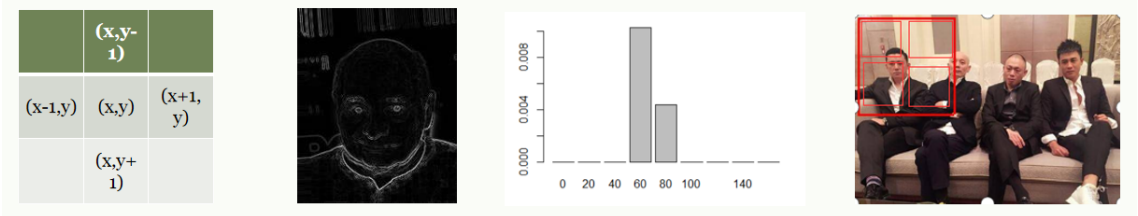
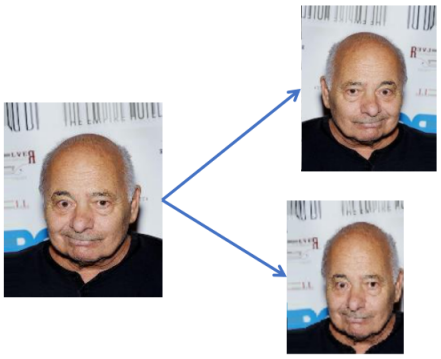


Figure 3: HOG steps

2.1.2 Improvement of Support Vector Machine

We firstly use the HOG features to train a normal C-SVM with package e1071, here C means we apply a punishment coefficient to the wrong classified samples, which is an important idea we will use later. We used some normal ways to improve our SVM model, including tune search, data enhancement and ROC analysis. The first two are trivial, we will just give a explain to the third method.



(a) Data enhancement

```
> tuned$best.parameters
   gamma cost
42 0.015 1.25
> tuned$best.performance
[1] 0.1297519
>
```

(b) Tune search

Figure 4: Trivial improvements

See the two graphs, we want false negative rate + false positive rate to be minimized, just want to find a threshold which have:

$$Threshold = argmin_x FN + FP = argmin_x 1 - TP + FP = argmax_x FP - TP$$

For in the ROC x to be FP, y to be TP, we just need to find a slope 1 line tangent to the ROC curve. Draw the ROC and got the best threshold by package pROC, then we got a better SVM as below.

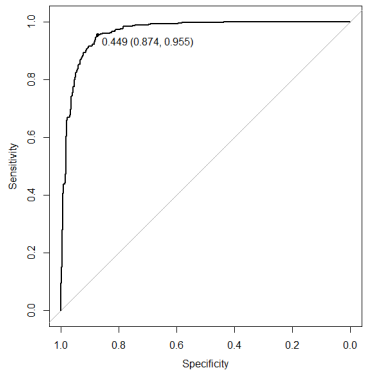


Figure 5: ROC analysis

```
> confusionMatrix(svm_table)
Confusion Matrix and Statistics

          test_y
svm_pred FALSE TRUE
   FALSE   418   26
   TRUE     52  444

               Accuracy : 0.917
                 95% CI : (0.8975, 0.9339)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.834

 Mcnemar's Test P-Value : 0.004645

            Sensitivity : 0.8894
            Specificity : 0.9447
         Pos Pred Value : 0.9414
         Neg Pred Value : 0.8952
             Prevalence : 0.5000
         Detection Rate : 0.4447
   Detection Prevalence : 0.4723
      Balanced Accuracy : 0.9170
```

Figure 6: Initial result

2.1.3 Increment Learing of SVM

We want to add a function of online learning to our model to achieve better performance, thus we introduce a simple ideal of SVM increment learning, I-SVM. The core idea is, since SVM is only depend on the support vectors, we could just train a model with support vectors and the given

new datas. This simple trick makes the training controlable, because at most cases, the number of support vectors will not change a lot, which promised a bound of space and time cost for each time of increment learning.(2)

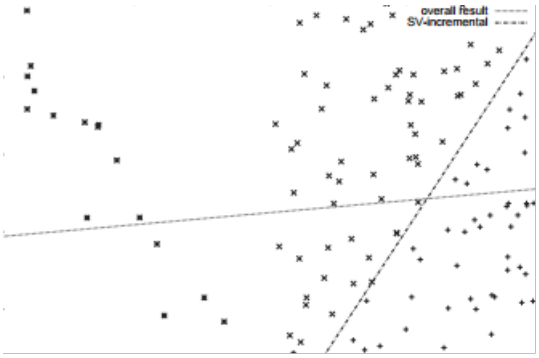


Figure 7: Icement learning

While the problem is, if new data is too much, the model will be dominate only by the new samples. To solve this, we just need to change our loss function a bit, as:

$$\underset{w}{argMin}\ Loss_{SVM}\left(w\right)\ + C\left(\sum \xi_{new(i)} + L\sum \xi_{old(j)}\right), L > 1.$$

to give more importance to the old observations. Actually, we have trouble changing the code of e1071 package, thus we approximate L by copy the support vectors for [L] times. Here we tried the increment model, the accuracy just keep still for the new data set is too small. We will further show the influence of increment learning in section 3.

2.1.4 Initial result by CNN

CNN(Convolution Neural Network) is powerful at image recognition(3). We will just briefly introduce it here. A normal CNN has typically 3 kinds of layers: convolution layers, pooling layers and fully connected layers. For the first one, we train a kernel to do convolution cell by cell over the image to derive features we are interested in; Pooling layer doesn't need to train, it gives the max feature from each sell; Fully connected cell maps the outcome of former layers (which has been flatten) to different labels, which is equivalent to a linear map.

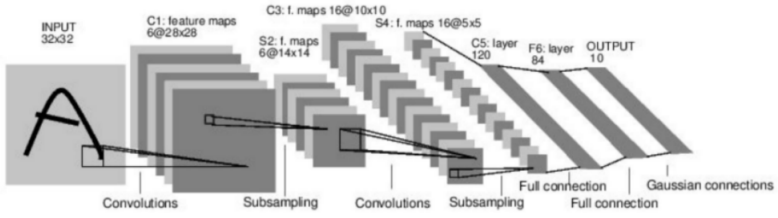


Figure 8: Structure of CNN

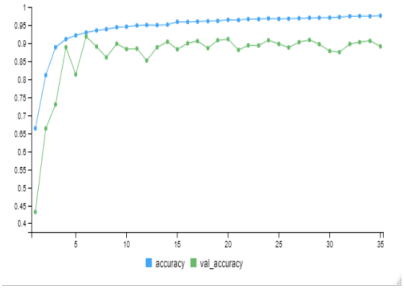
Here gives our CNN with 2 convolution layers, 2 pooling layers, 2 fully connected layers, and 2 dropout layers, which randomly kill some neural for each epoch, by keras package and TensorFlow frame. By cross validation, we find that over fitting may occur when it's over 30 epoches. Set epoch to be 30, we get 0.948 ACC.

2.1.5 CNN-IL-SVM

Notice that, the structure last two fully connected layers is just equivalent to a logic regression model, and in small size problems, SVM alway does better than LR, which indicates us that, if we

```
> model
Model
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape
=================================================================
 conv2d_2 (Conv2D)            (None, 159, 159, 16)

 max_pooling2d_3 (MaxPooling2D)  (None, 53, 53, 16)

 conv2d_1 (Conv2D)            (None, 44, 44, 8)

 max_pooling2d_2 (MaxPooling2D)  (None, 22, 22, 8)

 dropout_3 (Dropout)          (None, 22, 22, 8)

 flatten_1 (Flatten)          (None, 3872)

 cnn_feature (Dense)          (None, 128)

 dropout_2 (Dropout)          (None, 128)

 dense_1 (Dense)              (None, 2)

=================================================================
Total params: 515,226
Trainable params: 515,226
Non-trainable params: 0
_________________________________________________________________
```

(a) Our CNN



(b) Cross validation

Figure 9: Training of CNN

replace the final layer by a SVM, or we can say, make the out put of the first fully connected layer to be the input feature of SVM, we will possibly get a better outcome.(4) We do this by K.function in Keras package, which can derive the middle layer outcomes of a CNN. And we finally get a model with more than 0.95 ACC the result is as below:

```
> test_model(cnn_svm_model,cnn_svm_testset,test_y )
Confusion Matrix and Statistics

          test_y
svm_pred FALSE TRUE
   FALSE   455   32
   TRUE     15  438

               Accuracy : 0.95
                 95% CI : (0.9341, 0.963)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9

 Mcnemar's Test P-Value : 0.0196

            Sensitivity : 0.9681
            Specificity : 0.9319
         Pos Pred Value : 0.9343
         Neg Pred Value : 0.9669
             Prevalence : 0.5000
         Detection Rate : 0.4840
   Detection Prevalence : 0.5181
      Balanced Accuracy : 0.9500

       'Positive' Class : FALSE
```

Figure 10: Result for CNN-IL-SVM

We can then do all the improvement strategy above to make ACC over 0.95 (Actually 0.96 in one of the trials). Not to repeat here.

2.1.6 Model Test

Both of the testing samples are passed. "True" to be bald.

(a) Notbald

```
FALSE
attr(,"probabilities")
      TRUE      FALSE
1 0.04591608 0.9540839
Levels: FALSE TRUE
```

(b) Result

(c) Bald

```
TRUE
attr(,"probabilities")
     TRUE     FALSE
1 0.7185733 0.2814267
Levels: FALSE TRUE
```

(d) Result

Figure 11: two figures

2.2 Face detection

2.2.1 Haar-like feature and integration map

Haar-like future is more often be used for face detection than HOG, for it's easy to compute, and has a high ability to derive texture information from an image.(5) There isn't an R package for Haar-like feature, we realized it on our own, so I need to make some explain for my codes. Haar-like feature is computed just as the convolution layers of CNN, but the kernels are given(6) as the following graph, black element to be 1, white to be -1. In practice, we just take 1(a) and 1(b).

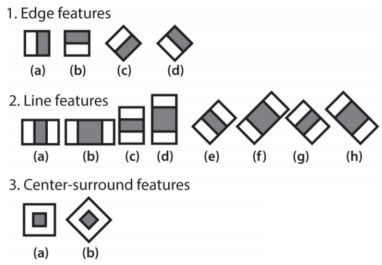


Figure 12: Haar kernels

We replace each pixel by the sum of all pixels up or left to it, then the compute of Haar features becomes simple plus-minus problems, which makes it more faster. Algorithm is trivial so I will not make any remarks. Here gives our outcome of one of the training samples

Figure 13: A man

Figure 14: Haar 1(a)

Figure 15: Haar 1(b)

We can see the information remains well, and the running time is mot more than 1s for one image, almost 1 over 10 of HOG costs.

2.2.2 Adaboost and cascade

It's obvious that, SVM is not suitable for such a high dimensional feature, alternatively, we utilize Adaboost, and then form a cascade classification model to accelerate our algorithm.(7) Adaboost is a kind of boosting with weight to be trained for each base classifier. Here we trained 4 Adaboost model with high recall to form a cascade, numbers of base classifier to be 1,2,4,8, we call the first and the second to be weak classifier, others to be strong ones.

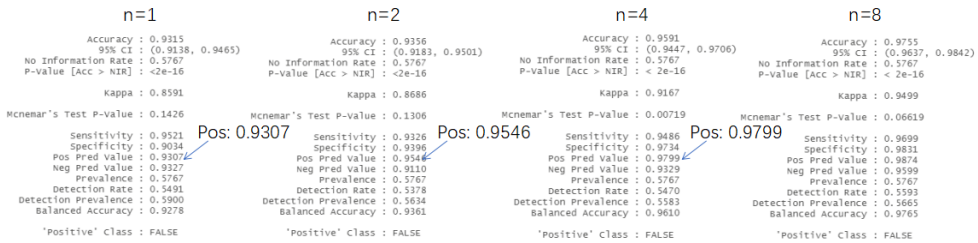


Figure 16: Trained Adaboosts

Cascade is a structure as below, which reject most of samples if the weak classifier claimed it to be negative, thus save us a lot of time from running a strong classifier. For such a structure, we don't want the weak classifiers to assign positive samples to negative, while we have relatively high tolerance for false negative prediction, that's why we prefer higher recall than accuracy when training Adaboost models.

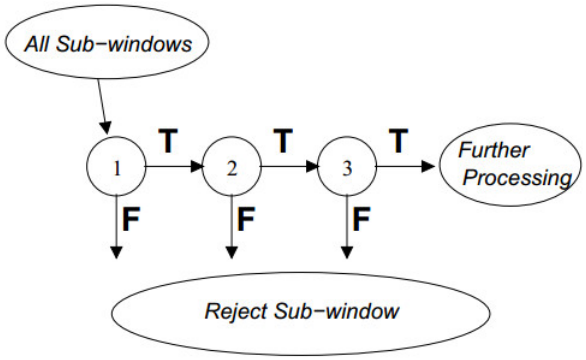


Figure 17: Structure of Cascade

That's all sounds great, while here comes a problem: it's too slow for R to allocate high dimensional data, and we can't just solve this by choosing significant features of a sample, for R doesn't accept this. This made our algorithm even much more slower than SVM. But we just remain this part, in case you readers have some good idea to deal with that.

2.2.3 Sliding window and NMS

The main idea: For the problem we argued above, Adaboost is not a good choice for face detection in R. As a result, SVM is now responsible for detecting the face, Window Sliding is responsible for detecting all positions of the picture, and NMS is responsible for de-weighting the sliding window results.

SVM. We trained a classifier using SVM with the new data set to recognize whether human faces in the area or not. For the new data set, the human set is consist of 2000 bald and 2000 not-bald, while the not-human set is about picture of flowers. Due to the data set, our classifier has a very high recognition accuracy for big-head photo identification, and the accuracy of the classifier in the confusion matrix reaches 97%. Also due to the limitations of the data set, our model here assumes that the face in the photo is facing straight into the lens, with clear facial features and no obvious shielding.

Sliding Window. Choose a window which size is than the picture, let the window Slide from the left up point on the picture to the down right point. If the picture size is $X \times Y$, the window size is $a \times b$, and the slide step is c, then when the window slide the picture overall, finally there will be $[\frac{X-a}{c}] \times [\frac{Y-b}{c}]$ different window cutting pictures. We pull all these pictures into the SVM model trained before, and it would return the results that the picture contains human face with predicted probability. Based on the results, we select the window cutting pictures that are most likely to contain faces by predicted probability. Usually we would get some windows with high overlap.

NMS. It means non-maximum-suppression, which is one method which could be used to Local extreme value search.(8) The idea to remove windows with high coincidence rate with the NMS is

as follows:

Step1 :*Define a overlapping ratio between picture A and B* AB_{iou} :
$AB_{iou} = area(A \cap B) \ / \ area(A \cup B)$, *choose a* iou_{max} IOU;

Step2 :*Order the pictures according to the predicted probability as the set* :
$\{C_1, C_2, C_3 \dots C_n\}$ *where* $pro(C_1) > pro(C_2) > \cdots > pro(C_n)$;

Step3 :*Add the maximum* C_1 *to another set* F, *and repeat the step*2 *for the rest picture until in* F :
$F = \{F_1, F_2, F_3 \dots F_K\}$,$\forall\ i, j < k$, $F_i F_{j_{iou}} < IOU$

figure:Sliding Window

figure:The predicted prob 99.99% windows

Figure 18: Sliding Windows + NMS steps

3 Research Results

3.1 Implement to group photo

With baldness recognizer and face position recognizer, it is very convenient to calculate baldness rate in the group photo. We return the slide-window picture containing the face to the personal baldness recognizer through face position detection. If the personal baldness recognizer recognizes TRUE, a red box will be drawn on the picture, and if FALSE, a green box will be drawn on the picture.

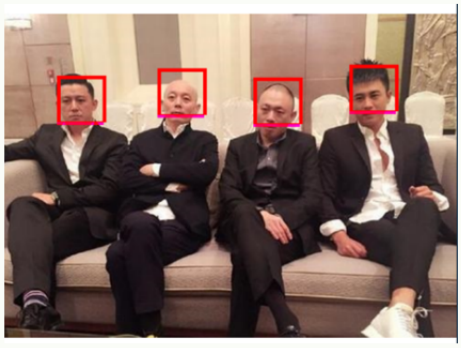


figure: face dection result

figure: group photo application

Figure 19: Group photo results

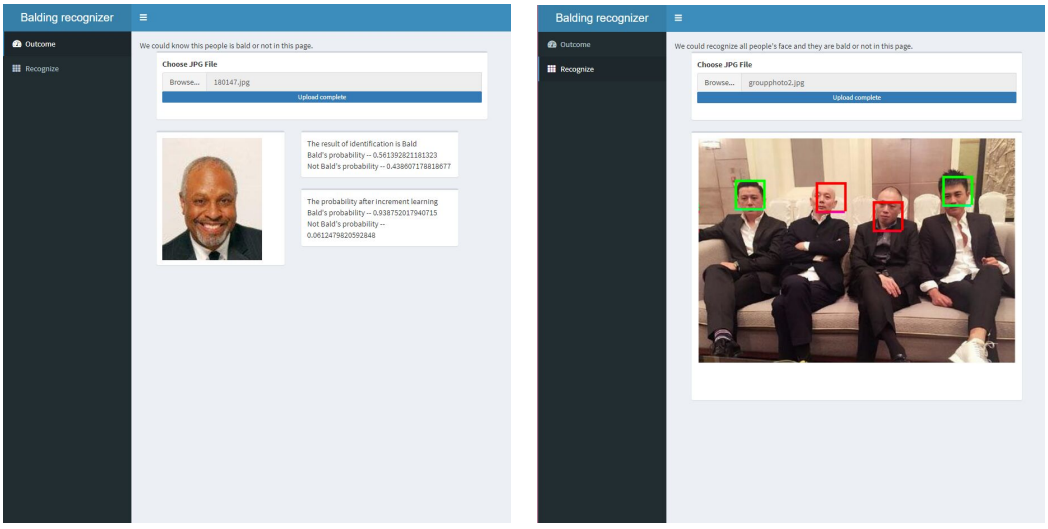
3.2 Implement to video

Actually video is a set of the group photo. Images are extracted through python frame processing, and the extracted images are tested for group baldness rate such as 3.1, and then the results of each image are returned and combined together to obtain baldness detection of the video

Figure 20: video results

3.3 Shiny UI Presentation

We also construct an Shiny UI to let it work more convenient.Here are two examples.In our shiny, you should first set the work path in pictures' file, and you should select 178*218 JPG picture in the first page, but in the second page you can choose any JPG but make sure that your work path is same as the pictures' file. The multiple bald-classification(the second page's implement) use the Increment method to evaluate the picture you choose.



(a) Shiny UI Example1

(b) Shiny UI Example2

Figure 21: Shiny UI example

4 Research Analysis

4.1 The limitation of SVM

In theory,increment learning is a good learning method and if we choose the good support vector as the representative of all the vector we will get a good result because support vector have some good nature.In fact,it is hard to choose a good enough vector that can substitute all the vector.

In Figure20 we can see that at the beginning the blue point is a support vector, but when red X

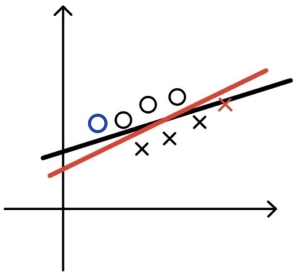


Figure 22: The deletion of support vector

has been inserted,the red line will be new dividing line and the blue point will not be support vector anymore because he is too far from the dividing line.It will cause some important information loss which will resulting in decreased accuracy.

In Figure21 we can see that after the update of dividing line,the green point and black point

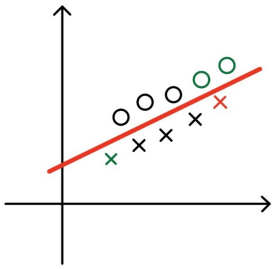


Figure 23: The loss of potential support vector

together make a new support vector set.But in the first-round-selection,the green point will be deleted because they are not support vector at all.So we will lose their information after deletion and it will resulting in decreased accuracy.

4.2 Improvement

To improve this algorithm,we choose shell vector instead of support vector.Shell vectors contains the support vectors we have chosen and the vectors that are potential to be new support vectors.Shell vector can avoiding the missing of new support vectors.

To speed up this algorithm,we can use KKT condition to restrict the vector newly added.KKT

condition is:

$We\ want\ to\ get\ the\ minima\ of\ f(x)\ under\ the\ restriction\ that$
$h_j(x) = 0, j = 1, 2, ..., p$
$g_k(x) \leq 0, k = 1, 2, ..., q$
$L(X, \lambda, \mu) = f(X) + \Sigma_{j=1}^{p} \lambda_j h_j(X) + \Sigma_{k=1}^{q} \mu_k g_k(X)$
$In\ order\ to\ solve\ the\ above\ optimization\ problem, the\ following\ conditions\ must\ be\ met$
$\frac{\partial L}{\partial X}|_{X=X^*} = 0\ (L\ is\ Lagrange\ function)$
$\lambda_j \neq 0$
$u_k \geq 0$
$\mu_k g_k(X^*) = 0$
$h_j(X^*) = 0, j = 1, 2, ..., p$
$g_k(X^*) \leq 0, k = 1, 2, ..., q$

They are KKT condition.In fact,KKT condition is to get the minimun value subject to some restriction.In SVM model,the vectors that satisfy KKT condition are support vectors.So we only need to choose the shell vectors in new data.After the selection,we put two shell vector together and we can get a new training data to do increment learning.

4.3 The goodness of model

We use ROC curve to evaluation our model.ROC curve implies the goodness of the prediction by our model.The point in the left top corner means the model is good, using different threshold value will get different point and we can use a curve to symbol them.So the larger the area under the curve,the better our model are.The left ROC is initial model and right one is increment-learning-model.They are similar with each other because our initial model is very good,but if you look closely,there are still some differences between them.

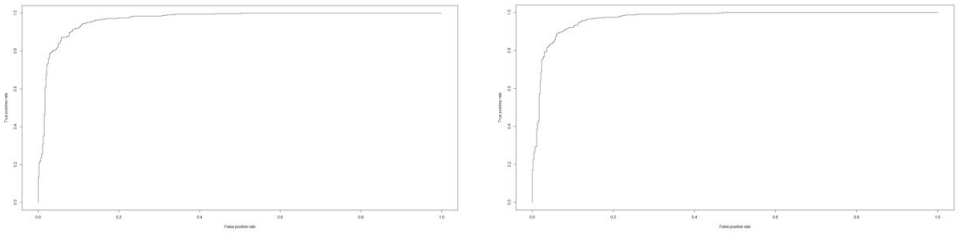


Figure 24: ROC curve of two model

4.4 Conclusion

After many tests and checks, we can evaluate the model as follows:

4.4.1 Advantages

The algorithm of SVM + Windows Sliding + NMS is simple and easy to understand and explain. It has relevantly high real meanings than deep learning based model, and has almost the same accuracy. The final implementation effect is good when detecting and classifying front-towards faces, it can be applied to practical problems.

Besides the final front-end containing model, we have some more powerful models like CNN-IL-SVM and Haar-Adaboost-Cascade. Thought it's hard to utilize in R (at least for us for now), they have proved to be efficient.

4.4.2 Drawbacks and reflection

Since the data set and model power limited, the model can only test faces facing the camera, but it is not good for people with a mask, a hat or dark skin. Haar and HOG are all weak at deal with side faces, maybe we should try LBP features, for example.

R is not a that good choice for complex machine learning models, though it has quite many packages, but R itselt is weak at allocate high dimensional data. This problem was revealed the most when utilizing Adaboost models, maybe we can disign a better predict function for boosting models, which only need to read significant features of the input samples.

And sliding window is a complicated process with high computation power and long running time, for a 530×330 picture, it takes about 3 miniutes to finish the recognition process. Also since the NMS is sensitive to parameters, the model needs to adjust parameters many times.

References

[1] B. T. Navneet Dalal, "Histograms of oriented gradients for human detection," *CVPR2005*, 2005.

[2] K. K. S. Nadeem Ahmed Syed, Huan Liu, "Incremental learning with support vector machines," 1999. www.comp.nus.edu.sg.

[3] Y. Y.Lecun, L.Bottou and P.Haffner, "Gradient-based learning applied to document recognition,," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[4] A. F. M. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," 2019. https://arxiv.org/abs/1712.03541.

[5] J. M. J. Viola P, "Robust real-time face detection.," *International journal of computer vision*, 2004.

[6] J. M. J. Viola P, "Rapid object detection using a boosted cascade of simple features," *ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 2001.

[7] P. V.]Lienhart R, Kuranov A, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection[m] pattern recognition.," *Springer Berlin Heidelberg*, 2003.

[8] V. G. Neubeck, "Efficient non-maximum suppression," *18th International Conference on Pattern Recognition (ICPR'06)*, 2006.