Face Mask Detection

Andrea Marchini Anthony Palmieri Ian Solagna

January 11, 2023

1 Introduction

Face masks have become an important tool in the fight against the spread of COVID-19. As more and more people are required to wear face masks in public, there is a growing need for automated systems to ensure compliance with mask-wearing regulations. In this project, we present a machine learning approach for detecting whether a person is wearing a face mask in an image.
At first, we introduce some noise in our data, to simulate corrupted images and test different denoising techniques, while the second part of the report deals with building predictive models, trained over the original dataset.

2 Dataset

The original dataset can be downloaded from Kaggle at this link. 7553 RGB images are split in 2 categories called *with mask* and *without mask*, which respectively contain 3725 and 3828 pictures. For our analysis, we preferred to focus over black-and-white images, with dimensions 150x150 pixels. Data are loaded as matrices, whose values are between 0 and 1, according to the colour represented.
The images feature people of different genders and ethnicities, and most of the pictures are close-up photos.
For the predictive models' part, data are split between a train set and a test set, which respectively contain the 85% and the 15% of the pictures.

3 Image Denoising

In the following section, we will analyze some denoising tools to apply to images. In fact, our images may come from security or public cameras, thus, it is possible that they contain some noise.
Our dataset, however, contains perfectly clear pictures. Because of this, in order to test our models we inject some white noise to each pixel.

Figure 1: Images with noise injected.

3.1 PCA

The first technique used is **PCA**, which is not designed as a denoising tool but, sometimes, selecting a reduced number of components gets rid of nonrelevant features of images, i.e. noise. The number of components we picked is 20 and it is fixed for all images, which is not optimal but on average this explains more than 95% of the variability.

Figure 2: PCA as image denoiser.

3.2 Fused Lasso

Fused Lasso is a technique that when applied without any predictor, works as a signal approximator making features close to each other as similar as possible. For this reason, it can be applied to images to remove noise.

3.2.1 Fused Lasso 1D

At first we tried the 1D version of the method. This forces the pixels next to each other (in a vertical way) to be identical. The parameters for the function were chosen by performing a grid search. The denoised images were compared to their original version using the *Frobenius Norm* of the difference in their pixels.

Figure 3: Fused Lasso 1D as image denoiser.

3.2.2 Fused Lasso 2D

This model is identical to the previous one but instead of focusing on one specific direction, forces pixels to be similar to those that are on their right/left and on their top/bottom. To choose the parameters we used the same method.

Figure 4: Fused Lasso 2D as image denoiser.

4 Predictive Models

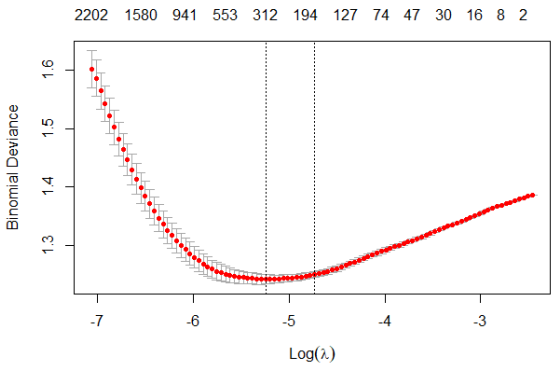
In this last section we finally focus on the actual goal of this project, that is face mask detection.
As we have already mentioned, this is an image classification problem, where, given an image, we want to be able to distinguish whether or not the subject portrayed in it is wearing a face mask. It is therefore a binary classification problem.
Unfortunately, due to the high dimensionality of the dataset, many of the algorithms analyzed during the course proved to be inapplicable, among these: Group Lasso, Sparse Additive Models and Fused Lasso, all of which resulted in excessive execution times.
Nonetheless, we were able to apply several other algorithms, specifically:

- Logistic Regression Lasso-penalized;
- Adaptive Lasso;
- Sparse Support Vector Machine;
- Decision Tree;
- Random Forest;
- Convolutional Neural Networks.

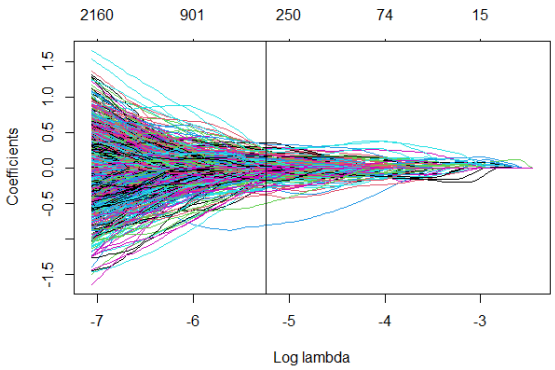
In the remaining part of this section we will discuss and analyze the first three of the above-mentioned models. As for the last three, we decided to simply report the accuracies achieved, without going into the detailed implementations.
To select the best value for each λ in the lasso models we used a 5-fold cross validation strategy.

4.1 Lasso

The first algorithm we implemented is a logistic regression with a Lasso penalty term. Recall that the Lasso objective function depends on an hyperparameter λ which controls the level of shrinkage.



(a) log(λ) via Cross-Validation

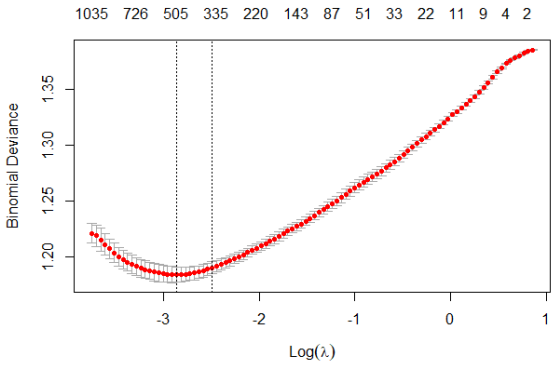


(b) Coefficients for the best-found λ

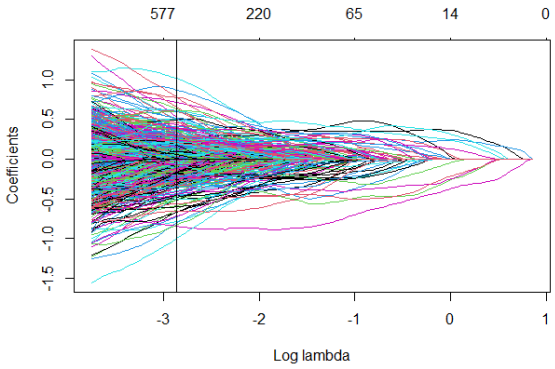
As we can see, the Lasso regularization results in a strong feature selection, decreasing the number of relevant predictors from 22500 down to just 312. The model achieves an accuracy of 68% on Test set.

4.2 Adaptive Lasso

The next model we tested is the Adaptive Lasso, a generalization of the Lasso aimed at reducing the bias of the classifier by limiting the shrinkage towards zero.
The idea behind the Adaptive Lasso is to introduce weights into the penalty term, one for each coefficient. In this sense, coefficients with lower weights will be penalized less whereas coefficients with higher weights will receive a stronger penalization.
To initialize the weights we used the optimal coefficients as found by Ridge Regression.



(a) log(λ) via Cross-Validation

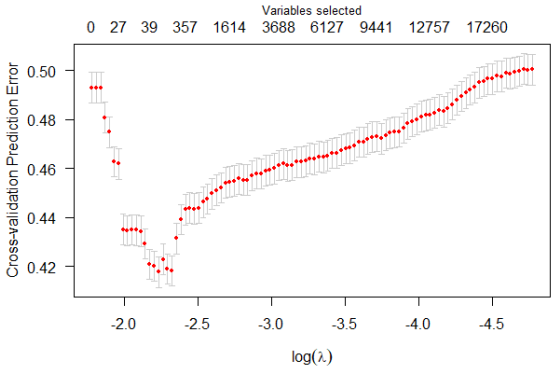


(b) Coefficients for the best-found λ

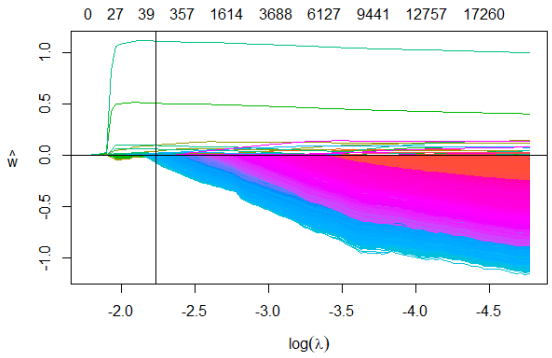
As expected, the number of selected features is now slightly larger (505 non-zero coefficients) even though the accuracy achieved on test set is still around 66%.

4.3 Sparse SVM

Finally, the last model we are going to discuss is a Lasso-regularized Support Vector Machine. Recall that the classical objective function for SVM can be rewritten using the Hinge-loss function, allowing us to introduce a Lasso regularization.



(a) log(λ) via Cross-Validation

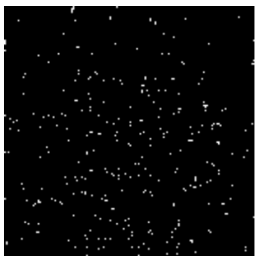


(b) Coefficients for the best-found λ

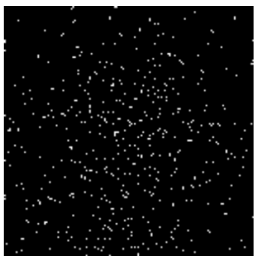
As can be seen from the plots above, feature selection is now even stronger, leading to only 88 nonzero coefficients. However, we could say that the regularization is now perhaps too strong, as the accuracy obtained on the test set is now only 61%.
In fact, this is the worst among the models we tested.

4.4 Selected Features

To conclude this section let's also take a look at the feature selection carried out across the three different models. The following pictures represent 150×150 images where the white pixels are the only selected predictors.



(a) Lasso



(b) Adaptive Lasso

(c) Sparse SVM

Contrary to what we hoped to see, the relevant pixels do not tend to concentrate at the bottom of the image, where the face mask detection activity should take place.
Looking at the dataset the reason is clear though: not all images are just frontal photographs, instead many depict the subject from different and unfamiliar perspectives, so the face mask is not always at the bottom of the picture.
Regardless, ignoring Sparse SVM which performs rather poorly, both Lasso and Adaptive Lasso tend to select pixels away from the edges and closer to the center, which actually makes sense.

5 Conclusions

As it is reported through this paper, we started focusing over the denoising part, applying PCA and two different Fused Lasso algorithms, with the latter behaving way better.
In the second part, the analysis deals with predicitve models, in order to classify the presence of a mask in the image.
Classic Lasso, Adaptive Lasso and Sparse SVM are outperformed by random forests, decision tree and CNN, even if they do not reach very high accuracy values.
The following table resumes the performances obtained by the different methods.

Model	Accuracy
Lasso	68%
Adaptive Lasso	66%
Sparse SVM	61%
Decision Tree	76%
Random Forest	86%
CNN	83%

Table 1: Performances of different models on the test set