Surgical Mask Detection - Report

Condurachi Corina

Description of the Machine learning approach:

I have tried various approaches of not only classification, but also extracting of features. I have used models such as Linear SVM, SVC, Random Forest classification, Naive Bayes and XGBClassifier. The ones that yields the best results are: SVC and XGBClassifier. The predictions made by the SVC classifier have been selected for the final evaluation.

1. SVM Method

‘ Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.
The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).’ [1]

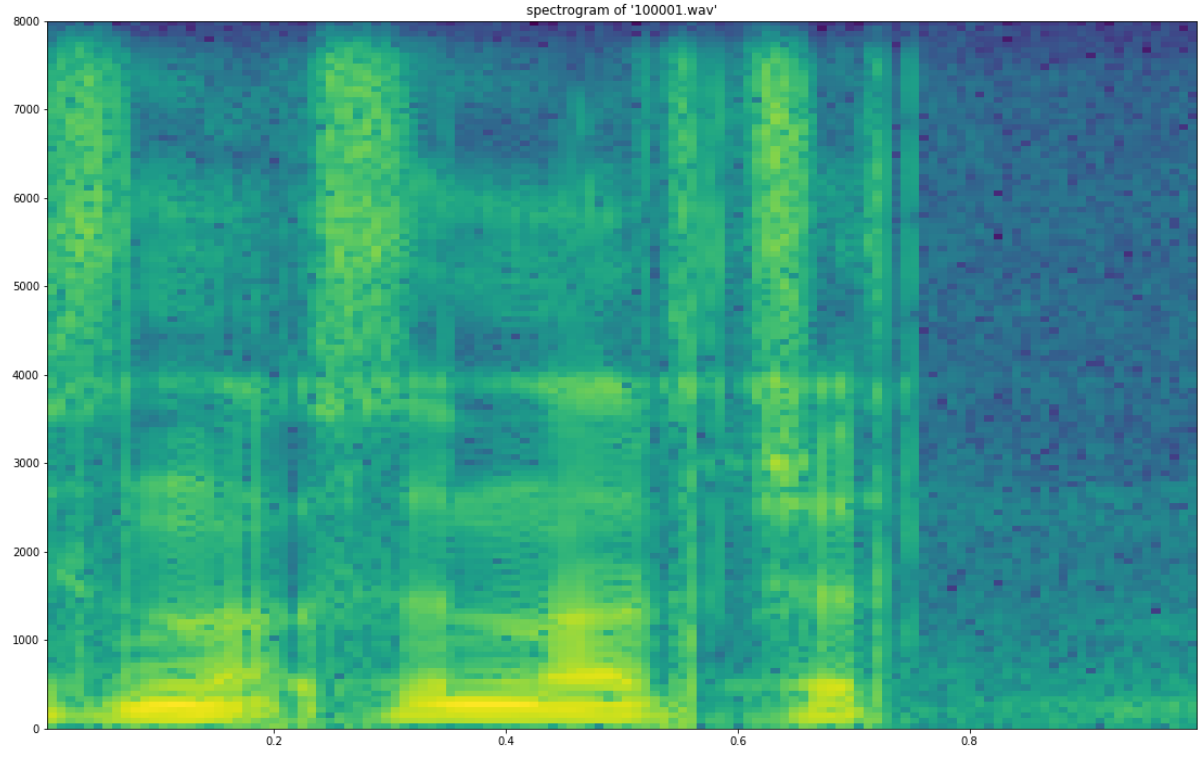
2. XGBClassifier

‘XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.’ [2]

Steps for my implementation of the SVC Classifier solution:

1. Function for extracting the features - reads every file using librosa.load (import librosa) and extracts the features for every file using mfcc and spectral_contrast (these two turned out to be the most efficient ones). It preprocesses the values (np.mean, np.std and skew) into arrays and stacks them horizontally (np.hstack) and finally returns a single array. With all those arrays, I build a list and transform it into an numpy.ndarray. (the format required in order to build the classifier). I have also tried fft method of extracting features, but turned out to yield poorer results.
2. I apply this function for every wav file in the training set. Transform the list into an numpy.ndarray in order to use for fitting. I have also tried to normalize the data using a StandardScaler, a MinMaxScaler, l1 and l2 normalisation, but turned out to yield not so good results.
3. Extracts the labels for every file and adds them to an array.
4. Builds an SVC classifier (from sklearn.svm import SVC) and fits the training_data and training_data_labels. I have also tried Naive Bayes Classifier, Random Forest Classifier, KNN, XGB Classifier but the one that proved to be the most efficient was the SVC model.
5. Extracts the features from the validation set of data, just the same as it does with the training set and then calculates the score (accuracy).
6. Extracts the features for test set and then predicts the test_labels. Every prediction will be written in a txt file, similar to the sample_submission one.

The **spectrogram** for the first audio file (100001.wav) can be observed below.



In order to test the model accuracy I have tried different values for parameters (gamma and C) and the accuracy grew as I tried bigger values for C. I realised this didn't yield true results and it was actually overfitting the validation set. In order to control overfitting, I have used crossed validation to test the true accuracy of the model. The results are shown in the following table.

C parameter	Accuracy for validation set	Cross Validation Results
1	0,658	0.60 ± 0.08
10	0,668	0.63 ± 0.08
100	0,725	0.66 ± 0.08
1000	0,779	0.66 ± 0.10
10000	0,798	0.65 ± 0.09

Alternatively, I will present the results I got applying a GridSearchCV for the XGBClassifier in the following table. I have chosen the most appropriate parameters to build my model (clf = XGBClassifier(max_depth = 10, min_child_weight = 2, gamma = 0). The cross validation Result was : Accuracy: 0.64 (+/- 0.09)

max_depth	min_child_weight	Accuracy
9	1	0.830251474878764
10	2	0.8316328825994272
8	2	0.8306775422346255

I have done a **classification report** so as to determine **precision**, **recall** and **f1-score** for my model (SVC classifier). Results are shown below.

```
: from sklearn.metrics import classification_report
print(classification_report(validation_labels, validation_predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.75      0.76       472
           1       0.78      0.80      0.79       528

    accuracy                           0.78      1000
   macro avg       0.78      0.78      0.78      1000
weighted avg       0.78      0.78      0.78      1000
```

Confusion matrix determines that out of 1000 predictions (on validation test), there were 725 right predictions and 275 wrong ones.

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(validation_labels, validation_predictions))
```

```
[[326 146]
 [129 399]]
```

The Python code for my approach

```
from scipy.stats import skew
import librosa
import numpy as np
from scipy.io import wavfile as wav
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report

# function for extracting the features from the audio files
def get_features(path,i):
    y, sample_rate = librosa.load(str(path) + str(i) + '.wav', sr = None)
    feature1 = librosa.feature.mfcc(y, sr = sample_rate)
    feature2 = librosa.feature.spectral_contrast(y)[0]
    feature1_preprocessed = np.hstack((np.mean(feature1, axis=1), np.std(feature1, axis=1), skew(feature1, axis = 1)))
    feature2_preprocessed = np.hstack((np.mean(feature2), np.std(feature2), skew(feature2)))
    return np.hstack((feature1_preprocessed, feature2_preprocessed))
```

```
# preparing the train set by extracting features from the audio files situated in train folder
path_train = '/Users/corinacondurachi/Downloads/ml-fmi-23-2020/train/train/'
train_data = []
# train set
for i in range (100001,108001):

    x = get_features(path_train, i)
    train_data.append(x)

train_data = np.asarray(train_data)
```

```
# extracting the labels for the coresponding audio files (train_data)
l = [None] * 8000
f = open("/Users/corinacondurachi/Downloads/ml-fmi-23-2020/train.txt", "r")
line = f.readline()
while line:
    l[int(line[0:6])-100001] = int(line[11])
    # read the next line
    line = f.readline()
f.close()
train_data_labels = np.array(l)
```

```
# building the model and applying fit
from sklearn.svm import SVC
clf = SVC(C = 100)
clf.fit(train_data, train_data_labels)
```

```
SVC(C=100, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
# preparing the validation set by extracting features from the audio files situated in validation folder
path_validation = '/Users/corinacondurachi/Downloads/ml-fmi-23-2020/validation/validation/'
valid_data = []
# validation set
for i in range (200001,201001):

    x = get_features(path_validation, i)
    valid_data.append(x)

validation_data = np.asarray(valid_data)
```

```
# extracting the labels for the coresponding audio files (validation_data)
l = [None] * 1000
f = open("/Users/corinacondurachi/Downloads/ml-fmi-23-2020/validation.txt", "r")
line = f.readline()
while line:
    l[int(line[0:6])-200001] = int(line[11])
    # read the next line
    line = f.readline()
f.close()
validation_labels = np.array(l)
```

```
# applying score for validation data
clf.score(validation_data, validation_labels)
```

```
0.725
```

```
# making predictions on the validation data
validation_predictions = []
validation_predictions = clf.predict(valid_data)
```

```
# preparing the test set by extracting features from the audio files situated in test folder
path_test = '/Users/corinacondurachi/Downloads/ml-fmi-23-2020/test/test/'
test_data = []
# validation set
for i in range (300001,303001):

    x = get_features(path_test,i)
    test_data.append(x)

test_data = np.asarray(test_data)
```

```
# making predictions on the test data
predictions = []
predictions = clf.predict(test_data)
```

```
# writing predictions for the test data in a txt file
f = open("/Users/corinacondurachi/Downloads/ml-fmi-23-2020/test15.txt", "w")
contor = 0
f.write('name,label\n')
for i in range (300001,303001):
    f.write(str(i) + '.wav,')
    f.write(str(predictions[contor]))
    f.write('\n')
    contor += 1
f.close()
```

```
# calculating the acores accuracy on validation data
scores = cross_val_score(clf, validation_data, validation_labels, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.66 (+/- 0.08)
```

```
# printing a classification report for the validation data
print(classification_report(validation_labels, validation_predictions))
```

```
              precision    recall  f1-score   support

           0       0.72      0.69      0.70       472
           1       0.73      0.76      0.74       528

    accuracy                           0.73      1000
   macro avg       0.72      0.72      0.72      1000
weighted avg       0.72      0.72      0.72      1000
```

```
# printing confusion matrix for the validation data
print(confusion_matrix(validation_labels, validation_predictions))
```

```
[[326 146]
 [129 399]]
```

References

1. https://scikit-learn.org/stable/modules/svm.html#svm-kernels
2. https://xgboost.readthedocs.io/en/latest/