SGN-16006, AUDITORY SCENE RECOGNITION

Jonas Nikula, Vili Saura

240497, 240264
jonas.nikula@student.tut.fi, vili.saura@student.tut.fi

1. INTRODUCTION

In this laboratory assignment we implement a k-nearest neighbor classifier and try to classify audio data collected in different environments. First we preprocess the data and extract the relevant features, and then we run them through our classifier. We've used Python 3.6 in this assignment.

2. METHODS

2.1. Feature extraction

We load the audio signals data using the Python module soundfile. This command gives us the actual data and the sample rate used. The sample rate in all the data was 8000Hz.

The features we want are the relative energy ratios of 4 frequency bands: $0.0-0.5kHz, 0.5-1.0kHz, 1.0-2.0kHz$ and $2.0-4.0kHz$. We get this by first dividing the audio signal into frames with a length of 30ms, and an overlap of 15ms. With the sampling rate used this means that one frame has 240 samples.

These frames are then multiplied, or windowed using the Hanning window function. In figure 1 there's the waveform of the 50th windowed frame.

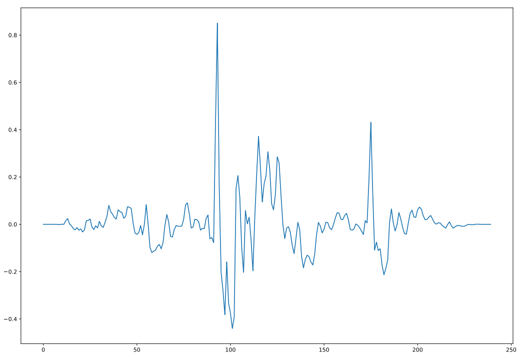

Fig. 1: 50th Hanning-windowed frame

Then we get the discrete fourier transform of the windowed frames, with a bin size of 1024. In figure 2 there's the amplitude spectrum of the 50th frame.

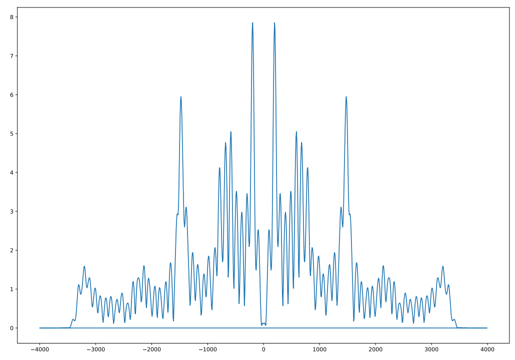

Fig. 2: Amplitude spectrum of the 50th frame

Out of the frame's DFT we extract the values corresponding to the 4 frequency bands. For example, the indices of the frequency band $1-2kHz$ are 128 to 256. Then we calculate the energy ratio of each band. The energy ratio is calculated by dividing the energy of a subband by the total energy of that frame. In figure 3 there's a visualization of the 50th frames extracted features. Finally, we combine all the frames of a sample by taking the average of all the frames energy ratios.

2.2. k-Nearest neighbor classification

When we have extracted the features, we implement a k-nn classifier as a Python object. Basically, in the training phase we just copy the data and labels. The real work is done in the prediction function, where we calculate all the distances between the sample whose label we're predicting, and all the training samples. When we have the distances, we take the k-lowest (ie. nearest), and out of those samples we take the most common.

The data set we use in the classification is the *Environmental Noise Data Set (Series 2)* recorded by the University

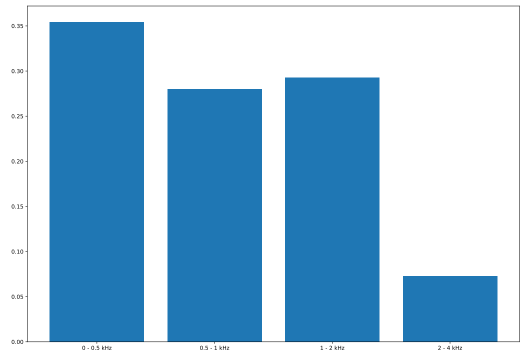

Fig. 3: Features of the 50th frame

Table 1: Class names and the corresponding numbers

Class name	Class number
Buildingsite	1
Bus	2
Car	3
Car highway	4
Laundrette	5
Office	6
Presentation	7
Shopping centre	8
Street people	9
Street traffic	10
Supermarket	11
Train	12

of East Anglia. In the test code and results we refer classes by their index number, but in table 1 we've compiled the class names and numbers together.

First we classify using only one neighbor. The overall accuracy was reported by scikit-learn as 72.1%. Table 2 has the prediction accuracies of the individual classes.

In figure 4 we can see the color plot of the normalized confusion matrix.

Then we run the classifier with 5 neighbors. This time the accuracy was 73.5%. Table 3 has the prediction accuracies of the individual classes.

Figure 5 has the color plot of the normalized confusion matrix of that classifier.

3. RESULTS AND CONCLUSIONS

3.1. Results

In this assigment, the overall classification accuracy didn't improve significantly by increasing the k value. The accu-

Table 2: The prediction accuracies of the NN classifier when k=1

Class	Classification accuracy (%)
1	98.46
2	50.00
3	77.78
4	79.69
5	74.24
6	100.00
7	73.13
8	52.46
9	50.94
10	77.78
11	46.03
12	78.26

Table 3: The prediction accuracies of the NN classifier when k=5

Class	Classification accuracy (%)
1	100.00
2	53.85
3	88.89
4	84.38
5	80.30
6	100.00
7	67.16
8	54.10
9	54.72
10	74.07
11	39.68
12	84.06

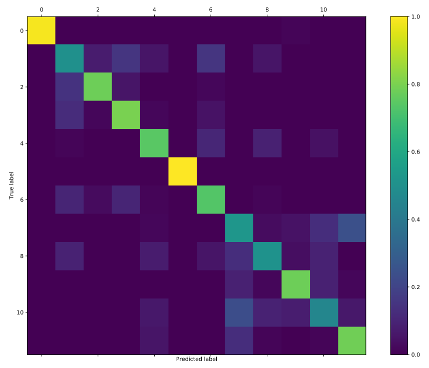

Fig. 4: Confusion matrix of k=1 classifier

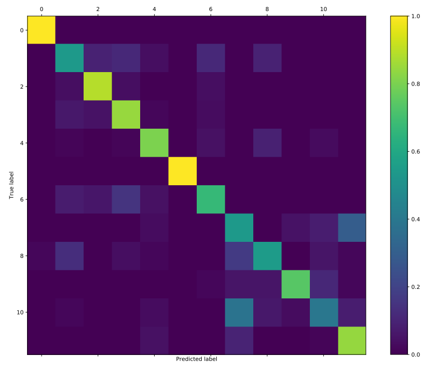

Fig. 5: Confusion matrix of k=5 classifier

racy with k=1 was 72.1%, and with k=5 73.5% which is only a 1.4 percentage point increase. In many test runs the accuracies were within $< 0.1\%$ of each other.

If we look at the individual classification accuracies, we see that some classes are classified very accurately, and some very poorly. The classifier performs worst on classes 2, 8, 9 and 11. In most cases, increasing k improves the accuracy of classes that were already accurate with k=1, except for classes 7 and 10. The classifier works best on classes 1 and 6, which have a 100% accuracy with the k=5 classifier, although this is unlikely to be the case in realistic use cases.

Looking at the confusion matrices, one interesting thing we see is that with k=5 there's less confusion overall, except for classes 11, 8 and to a lesser extent 6, which are often mislabeled as classes 8, 12 and 4 respectively. The rate of correct labels is improved with k=5, most noticeably with classes 3 and 12.

3.2. Conclusions

If we look at the worst and best classified classes and their names, we can take some guesses as to why the classifier performs as it does.

The classifier performed worst on the following scenes: Bus, Shopping centre, Street [with] people and Supermarket. What these all seem to have in common is that they have a lot of people and probably also other noises. They probably have a lot of energy on all the frequency bands. On the other hand, the classifier performed best on the scenes Buildingsite and Office. Although buildingsites are very noisy, their noises have a certain pattern to them, and there's not a lot of different noises. The Office on the other hand is just quiet, with some talking maybe.

The classifier seemed to also confuse the Supermarket and Shopping centre with each other, which isn't really suprising. Most people would take a while to differentiate them.

For such a simple method and implementation, our program performs surprisingly well. We can detect some scenes with very high accuracies. However, in realistic situations nearest neighbor isn't the greatest classifier because a) there are more accurate classifiers and b) it's slow at classifying. It's pretty much the slowest classifier at classifying, because it has to go through the whole training data each time it classifies something. Contrast it with other classifiers like random forests, which can take a while to train but whose classification is pretty much instantaneous.

3.3. Feedback

The instructions were easy enough to follow. Framing and nearest neighbor classifiers are things that are though on the basic courses, so most of the things in this assigment were not new. However it was interesting doing the whole classification procedure from start to finish. It took us about 4–5 hours to do the whole assignment and an additional 2 hours for the corrections.