

Classificação com dados de áudio

Projeto

Universidade Federal do ABC

Descrição

Neste projeto você desenvolverá um classificador capaz de distinguir gravações de diferentes caracteres. Especificamente, você receberá um arquivo **.wav** contendo uma sequência de 4 caracteres. Você precisará segmentar essa sequência, *i.e.* separar o áudio referente a cada caractere. Na sequência seu classificador deverá identificar cada caractere. Para cada arquivo wav recebido a predição é a combinação das predições individuais para cada caractere. Para simplificar o problema, consideraremos apenas os seguintes caracteres: a,b,c,d,h,m,n,x,6,7.

Você também participará da geração da base de dados. Para facilitar e padronizar tal geração foi disponibilizado o arquivo **gravacao.zip**, contendo o arquivo **gravacao.html** e alguns arquivos JavaScript da biblioteca P5JS¹. Abra o arquivo html com seu navegador, libere o acesso do microfone e siga as instruções na tela para a gravação. O script escolhe aleatoriamente os caracteres que devem ser gravados. O intuito é evitar a repetição exagerada dos caracteres, o que poderia reduzir a variação nos dados. Cada gravação será de 2 segundos. Você deverá gravar **pelo menos** 10 repetições de cada caractere. Note que após a gravação o seu navegador reproduzirá o áudio gravado e baixará o arquivo correspondente. Quanto maior a quantidade (com qualidade) das gravações, mais fácil tende a ser o trabalho do seu classificador, então faça com cuidado! :)

Todos os participantes devem realizar as gravações e enviar link para download via TIDIA até o dia **02/07**.

Algumas amostras de dados já gravados e unidos estão no arquivo **amostras.zip**, dessa forma você já pode explorar ideias de segmentação e classificação para o problema.

Avaliação

Este projeto terá 2 fases de avaliação. A primeira inicia dia **9/7** e finaliza dia **22/7**. Enquanto que a segunda inicia dia **23/7** e finaliza dia **22/8**. Ao término de cada fase de avaliação você deverá submeter dois arquivos. Um pdf contendo um breve relatório (modelo no TIDIA) e um script que execute o experimento e gere quaisquer tabelas/gráficos do relatório. Este script deve ser preferencialmente R ou Python, outras linguagens serão aceitas com as seguintes ressalvas: (i) deve ser possível executar o script sem adquirir nenhum software, *i.e.*, linguagens e bibliotecas gratuitas; (ii) no relatório deve ter uma descrição do procedimento necessário para executar o script. Ambos os arquivos devem ser submetidos no TIDIA na atividade correspondente.

A avaliação deste projeto será da seguinte forma:

- eficácia da abordagem de classificação: 40%
- metodologia experimental: 30%
- reprodutibilidade: 20%
- qualidade do script de experimentos: 10%

A eficácia da abordagem de classificação será mensurada em relação à três modelos desenvolvidos como **benchmark**. Denotando estes modelos como B1, B2 e B3. Grupos que tiverem eficácia pior que a obtida pelo B1 terão nota zero neste quesito; entre B1 e B2 terão nota 2,5, entre B2 e B3 nota 7,5 e acima de B3 nota 10. A reprodutibilidade será avaliada considerando o que consta no relatório e o que foi obtido ao executar o script de experimento. Na metodologia experimental serão considerados todos os cuidados

¹<http://p5js.org>

mencionados durante as aulas. Por fim, a qualidade do script é um prêmio para boas práticas de programação e organização!

Kit de iniciante

Caso você nunca tenha trabalhado com manipulação de arquivos de áudio e/ou aprendizado de máquina, nesta seção são apresentados alguns códigos em R e Python para servir de exemplo para algumas operações que são úteis no projeto.

Segmentação de áudio com pedaços de tamanho fixo

- Exemplo em Python (bibliotecas librosa e pandas)
 - <http://pandas.pydata.org/pandas-docs/stable/10min.html>
 - <http://librosa.github.io/librosa/0.4.2/tutorial.html>

```
1 import librosa
2 import pandas as pd
3
4
5 data, fs = librosa.load('6.wav', None)
6 duracao_total = data.shape[0]/fs
7 intervalo = 1
8 dados_p_seg = {}
9 for i, ini in enumerate(range(0, data.shape[0], fs*intervalo)):
10     dados_p_seg[i] = pd.Series(data[ini:(ini+fs*intervalo)])
11
12
13 dados_p_seg[0].plot()
14 dados_p_seg[1].plot()
```

- Exemplo em R (biblioteca tuneR)
 - <https://cran.r-project.org/web/packages/tuneR/tuneR.pdf>

```
1 library(tuneR)
2 audio <- readWave('6.wav')
3
4 str(audio)
5 m_audio <- mono(audio)
6
7 duracao_total <- length(m)/m@samp.rate
8 intervalo <- 1
9 quebras <- seq.int(0, length(m), by = m@samp.rate * intervalo)
10 dados_p_seg <- lapply(seq_along(quebras),
11                       function(i){
12                           q_pos <- if(i == length(quebras)) length(m) else quebras[i+1]
13                           m[quebras[i]:q_pos]
14                       })
15
16
17 plot(dados_p_seg[[1]])
18 plot(dados_p_seg[[2]])
```

Classificação com Análise de Discriminante Linear

- Exemplo em Python (biblioteca scikit-learn)
 - http://scikit-learn.org/stable/user_guide.html

```
1 import sklearn
2 from sklearn.datasets import load_iris
3 from sklearn.model_selection import train_test_split
4 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
5
6 iris = load_iris()
7 X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
8                                                    test_size=0.4, random_state=0)
9
10 clf = LinearDiscriminantAnalysis()
11 y_pred = clf.fit(X_train, y_train).predict(X_test)
12 print(sum(y_pred == y_test)/len(y_pred))
```

- Exemplo em R (biblioteca mlr)
 - <https://mlr-org.github.io/mlr-tutorial/release/html/index.html>

```
1 library(mlr)
2
3 task <- makeClassifTask(data = iris, target = "Species")
4 lrn <- makeLearner("classif.lda")
5 rdsc <- makeResampleDesc("Holdout", split = 0.6)
6
7 mod <- resample(lrn, task, rdsc)
8 mod
```