Classificação com dados de aúdio

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 P5JS[1]. 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 **20/10**.

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 **21/10** e finaliza dia **17/11**. Enquanto que a segunda inicia dia **18/11** e finaliza dia **8/12**. 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 mencionados durante as aulas. Por fim, a qualidade do script é um prêmio para boas práticas de programação e organização!

[1] http://p5js.org

Kit de iniciante

Caso você nunca tenha trabalhado com manipulação de arquivos de aúdio 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

```
import librosa
import pandas as pd


data, fs = librosa.load('6.wav', None)
duracao_total = data.shape[0]/fs
intervalo = 1
dados_p_seg = {}
for i,ini in enumerate(range(0, data.shape[0], fs*intervalo)):
  dados_p_seg[i] = pd.Series(data[ini:(ini+fs*intervalo)])


dados_p_seg[0].plot()
dados_p_seg[1].plot()
```

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

```
library(tuneR)
audio <- readWave('6.wav')

str(audio)
m_audio <- mono(audio)

duracao_total <- length(m)/m@samp.rate
intervalo <- 1
quebras <- seq.int(0, length(m), by = m@samp.rate * intervalo)
dados_p_seg <- lapply(seq_along(quebras),
                      function(i){
                        q_pos <- if(i == length(quebras)) length(m) else quebras[i+1]
                        m[quebras[i]:q_pos]
})


plot(dados_p_seg[[1]])
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

```
import sklearn
from sklearn.datasets import load_iris
```

```
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
                test_size=0.4, random_state=0)

clf = LinearDiscriminantAnalysis()
y_pred = clf.fit(X_train, y_train).predict(X_test)
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

```
library(mlr)

task <- makeClassifTask(data = iris, target = "Species")
lrn <- makeLearner("classif.lda")
rdesc <- makeResampleDesc("Holdout", split = 0.6)

mod <- resample(lrn, task, rdesc)
mod
```