Relatório do Projeto — Mineração de Dados

Gustavo Zanfelice Dib

16/11/2019

1 Introdução

Este projeto visa a classificação de arquivos de áudio gravados possuindo 4 letras pronunciadas espaçadamente. As 'classes' compreendidas são a,b,c,d,h,m,n,x,6,7. O retorno do programa é a combinação das 4 letras presentes no áudio.
Para isto foram utilizados 356 arquivos de áudio como treinamento de um modelo classificador. Resultando na distribuição a baixo:

Treinamento									
a	b	c	d	h	m	n	x	6	7
114	123	125	118	114	127	120	114	122	121

Para validação foram utilizados 267 arquivos com a distribuição:

Validação									
a	b	c	d	h	m	n	x	6	7
97	106	107	110	122	104	113	95	100	114

Os testes serão realizados como método avaliativo, sendo assim, não temos acesso à distribuição destes.
Para o projeto, todo o programa foi escrito na linguagem Python 3, através das bibliotecas:

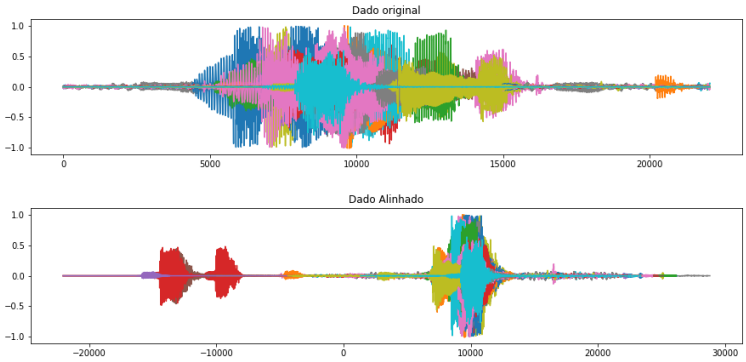
- Pandas: Uma biblioteca que auxilia a manipulação de dados em formato de tabelas implementando funções de processamento e de visualização
- Numpy: Uma biblioteca matemática que implementa diversas operações, como média, mediana, operações vetoriais e afins.
- Librosa: Uma biblioteca de leitura, processamento e gravação de arquivos de áudio.
- SciKit-learn: Uma biblioteca de aprendizado de máquina, implementando modelos de aprendizado, assim como métricas de avaliação e seleção de hiper parâmetros.

- MatplotLib: Biblioteca de visualização de gráficos e análises visuais.
- NoiseReduce: Esta biblioteca criada por Tim Sainburg que reduz ruídos de áudio.

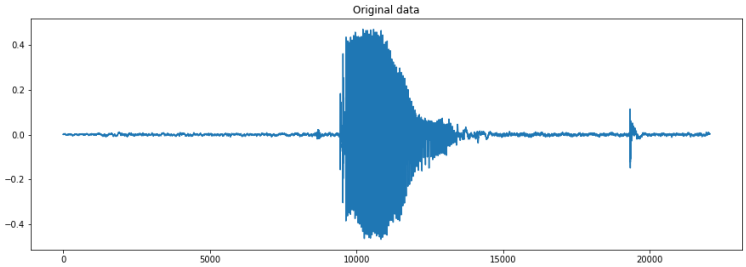
2 Análise Exploratória

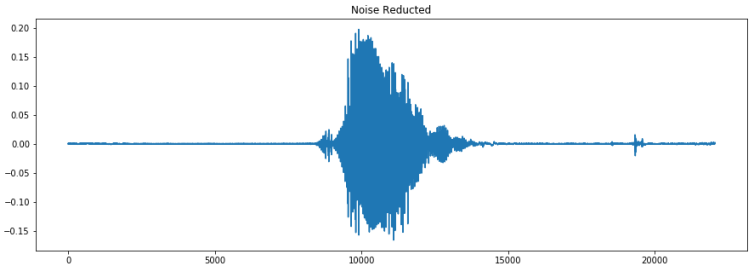
2.1 Transformações nos dados

Primeiramente, a tarefa inicial era isolar os quatro caracteres de cada áudio, para ter uma melhor compreensão da similaridade entre áudios de mesmo carácter e afins. Assim, os áudios foram segmentados e isolados em pastas de seu respectivo carácter. O exemplo dos dados da letra 'b' seguem abaixo (para melhor visualização, foi criada a visão alinhada dos sinais).

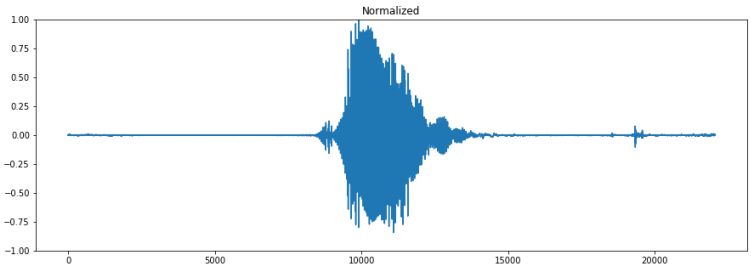


Um problema comum ao se trabalhar com áudio é o ruido sonoro em torno do áudio. Para diminuir este ruído, foi utilizada a biblioteca *NoiseReduce* que realiza a suavização de um ruído a partir de uma amostra deste. Um exemplo de suavização segue a baixo.





Em seguida, é notável que por se tratar de arquivos de áudio gravados em diferentes dispositivos por diferentes pessoas, temos uma distância de escala entre os áudios, onde até mesmo o volume em que a mensagem foi dita pode causar esta diferenciação. Para neutralizarmos essa discrepância, foi utilizado o método de normalização da biblioteca librosa. Aplicando essa normalização ao mesmo exemplo exibido acima, temos a mudança:



2.2 Extração de Features

Para o processo de aprendizado, como estamos lidando com uma série temporal, iremos extrair *features* desta série e armazena-las em um arquivo csv que será processado posteriormente. Para a seleção de quais features extrair, foram analisados os métodos da biblioteca librosa.features e vistos em alguns estudos similares. Dentre todas as features extraidas, a mais notável é o *Mel-frequency cepstrum*, pois este é robusto a ruídos e por aplicar logaritmo, atenua picos existentes. Estas *features* são:

- *Constant-Q chromagram*: Média do cronograma da transformação Constant-Q, que transfere o dado para o domínio da frequência.
- *Short-time Fourier transform chromagram*: Similar à *feature* anterior, também foi utilizada a média da transformada de Fourier do áudio, que também transfere o dado para o domínio da frequência.

- Centroide do Espectro: Média dos centroides dos frames (normalizados) do áudio
- Largura de Banda: Média das larguras de banda de cada frame do áudio
- Taxa de Cruzamentos no Zero: Número de vezes que o áudio cruzou o índice 0 no eixo y.
- Frequência de Roll-Off: Definida para cada quadro como a frequência central de modo que pelo menos 0,85 da energia do espectro nessa estrutura esteja contido nessa bandeja e nos compartimentos abaixo.
- Mel-frequency cepstrum: representação do espectro de potência de curto prazo de um som, com base em uma transformação de cosseno linear de um espectro de potência de log.

3 Metodologia

Com um arquivo csv de *Features* extraídas para todas as amostras que serão utilizadas, temos uma tabela no formato: Com isto, foram realizados dois pré-

	filename	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	...	mfcc17	mfcc18	mfcc19	mfcc20	label
0	0_2.wav	0.675671	0.647213	0.032813	4821.223004	...	0.258107	1.074864	-4.096735	0.139661	0.766718
1	100_0.wav	0.641985	0.500162	0.029933	4067.835660	...	3.159953	1.224929	3.162031	4.524626	7.771187
2	101_0.wav	0.602044	0.538624	0.055689	3305.766121	...	3.121379	1.829411	0.203210	-1.256773	2.162389
3	101_2.wav	0.688363	0.725112	0.043488	4474.397443	...	-0.924525	-0.984750	-3.872282	-2.954205	-5.688318
4	102_0.wav	0.724626	0.774140	0.033320	4744.066601	...	-0.674110	-3.580620	0.614482	-2.337616	-3.857900

processamentos nesta base: Os caracteres alvos foram tokenizados para o modelo e todas as features (por serem numéricas e suscetíveis) foram redimensionadas através da operação (x - média)/(desvio padrão).
Com os dados prontos, foi decidido que o modelo implementado é o Support Vector Machine (SVM) pois, como temos diversos atributos diferentemente distribuídos, é provável que os dados sejam linearmente separáveis ou separaveis através da aplicação de um kernel. Sendo assim, foi testado duas implementações: O SVC linear para o teste da possível divisão linear e o SVC com a função de kernel *Radial basis function kernel* (RBF).
Para a seleção de parâmetros do Kernel RBF foi realizado um Grid Search variando dois parâmetros: A penalidade por erro 'C' entre 0.5 e 100 e o coeficiente do kernel 'gamma' entre os valores [0.001, 0.01, 0.1, 1].
Para a seleção do melhor modelo, o critério de avaliação foi Acurácia, pois como se trata de um problema de classificação multi-classe, o modelo não classifica visando positivos e negativos. A aplicação do paradigma *One-Vs-All* poderia abrir visões para outros métodos de avaliação
Após a realização da busca em largura, foi obtido que os melhores parâmetros são:

- C=3

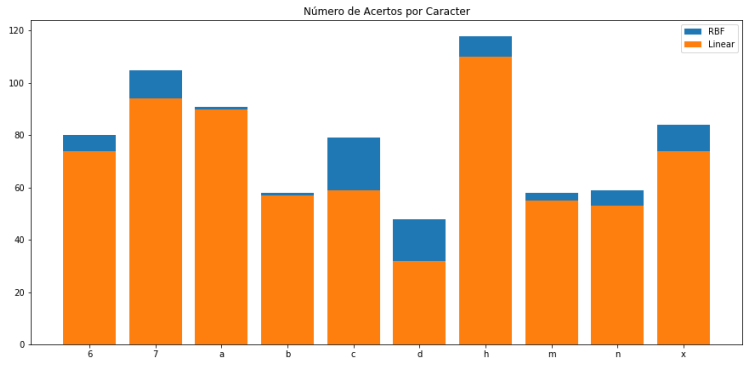
- gamma=0.1

4 Resultados

Após o treino e avaliação dos modelos, obtive que o modelo melhor ajustado é o SVM com kernel rbf, apresentando 73.0% acurácia contra 65.3% para o kernel linear. O detalhamento dos acertos e erros para o Kernel rbf é exibido abaixo:

	6	7	a	b	c	d	h	m	n	x	Total	Total(%)
Acerto_rbf	80	105	91	58	79	48	118	58	59	84	780	0.730337
Erro_rbf	20	9	6	48	28	62	4	46	54	11	288	0.269663
Acerto_linear	74	94	90	57	59	32	110	55	53	74	698	0.653558
Erro_linear	26	20	7	49	48	78	12	49	60	21	370	0.346442

A comparação entre os dois kernels pode ser vista no gráfico que resume a quantidade de acerto por carácter tratado: Sendo assim o modelo de kernel



rbf apresentou melhor performance para todos os padrões analisados, compreendendo melhor os dados apresentados.

Para o teste do modelo, será inseridos novos arquivos de mesmo formato e então realizada a classificação dos novos arquivos recebidos.

5 Comentários Finais

Trabalhar com arquivos de áudio requer uma boa compreensão de ondas para a melhor extração de features úteis e melhor normalização. Possivelmente existem melhores atributos para serem analisados, assim como a analise em cima

do espectrograma literal de cada onda.
Outra abordagem bem interessante para a proposta seria utilizar Deep Learning e redes neurais para a modelagem, já que redes convolucionais poderiam identificar melhor os padrões exibidos.