Reconhecimento de Comandos de Voz

Luigi G. Marchetti[1], **Ari Elias da S. Júnior**[2]

[1] Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

lgmarchett@furb.br, ari@furb.br

Resumo. *Este trabalho investiga o uso de técnicas de aprendizado de máquina para reconhecimento de comandos de voz voltado ao controle de robôs móveis. Avaliamos e comparamos diversos modelos, incluindo Máquinas de Vetores de Suporte (SVM), K-Vizinhos Mais Próximos (KNN) e uma rede neural. A extração de características baseia-se em coeficientes cepstrais na escala de Mel (MFCC), com adição de atributos espectrais e otimizações nos modelos autorais. Os resultados mostram que SVM e KNN alcançam alta acurácia mesmo com áudios reais, evidenciando uma boa generalização.*

1. Introdução

Com o avanço das tecnologias de automação e inteligência artificial, os robôs móveis têm sido amplamente empregados em tarefas como patrulhamento, inspeção e assistência a humanos em diversos contextos.

O reconhecimento de voz utilizando da inteligência artificial, especificamente de aprendizado de máquina para aprender certos comandos é peça fundamental em determinadas indústrias.

Operadores de máquinas usam comandos de voz para controlar braços robóticos, acionar esteiras ou verificar status de sensores sem precisar tocar em telas ou teclados [Fraunhofer 2023].

O estudo de Rendyansyah [Rendyansyah 2022] demonstra a implementação de um robô móvel controlado por voz, utilizando os métodos Mel-Frequency Cepstral Coefficients (MFCC) para extração de características e Support Vector Machine (SVM) para reconhecimento de padrões de comandos como ”forward”, ”backward”, ”left”, ”right”e ”stop”, que significam em português respectivamente ”avançar”, ”recuar”, ”esquerda”, ”direita”e ”parar”. Essa abordagem alcançou uma taxa de sucesso de 96% em experimentos, evidenciando a viabilidade de sistemas baseados em voz para navegação robótica somente com a voz.

Encontramos as mesmas palavras em grande quantidade (cerca de 4.000 áudios para cada palavra) no conjunto de dados do chamado de Google Speech Commands V1 [Dogra 2023].

A aplicação tem a necessidade de processar sinais de voz de forma robusta e eficiente, utilizando técnicas como MFCC para extração de características e modelos de aprendizado de máquina para classificação. Este artigo busca explorar essas tecnologias, aprendendo com os avanços apresentados em [Rendyansyah 2022], com o objetivo de propor um algoritmo de aprendizado de máquina com precisão do reconhecimento de comandos de voz para robôs móveis, para ser capaz de ter uma aplicação eficiente que reconhece comandos de voz em tempo real.

2. Materiais e Métodos:

Vale ressaltar que em todos algoritmos abaixo utilizamos NumPy para operações matemáticas e Matplotlib para visualização gráfica dos dados e resultados.

Utilizamos a validação cruzada 5-fold na nossa base de testes em todos algoritmos (menos no item 3.2). Ele consiste em dividir os dados em cinco partes (ou "folds"). Em cada rodada, uma parte é usada para teste e as outras quatro para treino. Isso é repetido cinco vezes, trocando a parte de teste a cada rodada, e a média dos resultados é usada para avaliar o desempenho do modelo.

2.1. Dataset

Iniciamos com a definição do nosso conjunto de dados [Dogra 2023] e sua análise. O Google Speech Commands Dataset v0.02 é uma coleção curada de gravações curtas de áudio (aproximadamente um segundo) contendo palavras isoladas, desenvolvida especificamente para treinamento e avaliação de sistemas de detecção de palavras-chave (KWS). O KWS e o reconhecimento de comandos de voz são tarefas intimamente relacionadas no campo do processamento de sinais de áudio. Ambas envolvem a identificação de padrões específicos em dados de voz, e para nosso objetivo os dados se encaixam perfeitamente.

Cada gravação representa um único comando de voz pronunciado por diferentes falantes, o que torna o conjunto altamente útil para o desenvolvimento de aplicações robustas de controle por voz em ambientes reais. Todos os áudios são em inglês e decidimos por utilizar apenas as palavras do artigo [Rendyansyah 2022] para restringir nosso escopo, elas são: "forward", "backward", "left", "right"e "stop", que simulam um controle de um robô.

2.2. Implementação SVM baseada em artigo existente

Então seguimos para a reaplicação do artigo de Rendyansyah [Rendyansyah 2022], com o objetivo de verificar o desempenho em nossa base de dados. Utilizamos da linguagem de programação Python (versão 3.12) e de algumas de suas bibliotecas específicas para nosso cenário.

Uma das bibliotecas mais relevantes foi a Librosa, utilizada para o pré-processamento dos áudios e extração de características dos áudios por meio da técnica Mel-Frequency Cepstral Coefficients (MFCC).

Os coeficientes MFCC extraídos com os seguintes parâmetros: 20 coeficientes, tamanho da janela FFT de 256 pontos, e deslocamento de 128 amostras. Para representar cada comando de voz, calculou-se a média dos coeficientes ao longo do tempo, resultando em um vetor de 20 dimensões por amostra. A taxa de amostragem foi fixada em 11 kHz, consistente com o artigo.

A segmentação dos dados e a divisão em subconjuntos de treino e teste foram realizadas com a Scikit-learn, como não é especificado no artigo, utilizando 20% do conjunto de total como teste.

A biblioteca Scikit-learn também foi usada para o treinamento do classificador Support Vector Machine (SVM). Utilizamos os parâmetros fixos conforme o artigo original com o (*kernel* RBF, *gamma*=1000, $C = 1$).

2.3. Implementação de algoritmo autoral com SVM

Com o algoritmo anterior não conseguirmos resultados tão eficazes, avançamos para tentar algo diferente com base no que fizemos até esse ponto, ajustando para nosso contexto e otimizando conforme abaixo.

Enquanto no script anterior do artigo que utilizamos, era baseado em 20 coeficientes MFCC médios, fizemos uma estratégia mais robusta que incorpora: 13 MFCCs com derivadas temporais (delta e delta-delta), características espectrais complementares (*centróides, roll-off e ZCR*), além de estatísticas adicionais (média, desvio padrão, máximos e mínimos). O pré-processamento foi reforçado com normalização, remoção de silêncio e padronização de duração.

Ademais, nosso algoritmo introduz o uso de *GridSearch* para otimização de hiperparâmetros do SVM (como *kernel*, C e *gamma*), diferente do original que deixamos fixos os valores que o pesquisador Rendyansyah encontrou como sendo o melhor. Assim, permitindo uma busca pelos melhores parâmetros, onde testamos todas alternativas misturadas em vez de depender de valores fixos como no primeiro script.

Outra diferença é o pré-processamento mais sofisticado, que inclui padronização da duração do áudio para 1 segundo, garantindo consistência nos dados e a remoção de silêncio. Essas melhorias visam aumentar a robustez e a generalização do modelo, potencialmente melhorando a acurácia em comparação com a abordagem mais simples primeiramente realizada.

2.4. Implementação de algoritmo autoral com Rede Neural

Com o objetivo de explorar arquiteturas mais avançadas para reconhecimento de comandos de voz implementamos uma nova rede neural buscando ser simples e eficiente.

Utilizando a biblioteca PyTorch, conforme descrito no código apresentado. Essa abordagem utiliza 13 coeficientes MFCC extraídos com a biblioteca Librosa, normalizados e achatados como entrada para uma rede com quatro camadas.

A rede utiliza o classificador MLP (Perceptron Multicamadas). É aplicada uma técnica básica de normalização, para tentar contribuir a uma melhor generalização do modelo. Além de que, continuamos a utilizar a extração de características dos áudios com MFCC e os áudios são padronizados também para 1 segundo com taxa de amostragem de 16 kHz.

Na arquitetura da rede, a primeira camada mapeia a entrada para 256 neurônios, seguida por camadas que reduzem para 128 e 64 neurônios, e uma camada de saída com 5 neurônios (um por comando). Cada camada oculta utiliza ativação ReLU para introduzir não-linearidade, normalização em lote (*batch normalization*) para estabilizar o treinamento, e *dropout* (50% nas duas primeiras camadas ocultas e 30% na terceira) para prevenir *overfitting* (sobreajuste). O treinamento foi conduzido com o otimizador Adam (taxa de aprendizado de 0.001), função de perda de entropia cruzada, um *batch size* de 32 e um total de 50 épocas, salvando o modelo com a melhor acurácia no conjunto de validação.

2.5. Implementação de algoritmo autoral com KNN

Com Scikit-learn, nossa última implementação feita foi o K-Nearest Neighbors (KNN).

Diferente da rede neural, que possui um custo computacional maior, o KNN é leve e ideal pra situações com menos recursos. Comparado aos algoritmos descritos acima, o KNN é mais simples, mas ainda aplicável a esse cenário e o escolhemos pois ele lida bem com padrões não lineares dos áudios sem precisar de uma arquitetura complicada.

Para o treinamento, usamos *GridSearch* com validação cruzada de 5 *folds* já mencionada, testando várias combinações de parâmetros: número de vizinhos (3, 5, 7, 9, 11, 15, 21, 25), pesos (*uniform* ou *distance*), métricas de distância (*euclidean*, *manhattan*, *minkowski*) e o parâmetro *p* (1 ou 2). Montamos um *pipeline* com *StandardScaler* pra normalizar as características (crucial pro KNN) e o classificador. O *GridSearch* escolheu os melhores parâmetros com base na acurácia da validação cruzada.

3. Resultados

Para avaliar a eficácia dos modelos de classificação utilizados neste trabalho, realizamos testes cruzados dos algoritmos especificados acima.

E para evitar overfitting, realizamos uma coleta adicional de dados. Gravamos dois áudios (com aproximadamente 2 segundos cada) para cada uma das palavras do conjunto de comandos utilizados: *['right', 'left', 'forward', 'backward', 'stop']*. Esses dados foram utilizados exclusivamente na etapa de testes.

Tabela 1. Acurácia dos modelos testados na classificação de comandos de voz com base original e dados caseiros

Modelo	Base original (%)	Dados caseiros (%)
Modelo do Artigo	86	20
SVM	99	100
KNN	98	100
Rede Neural	98	40

Além da acurácia apresentada na Tabela 1, é importante analisar outras métricas como *precision*, *recall* e *f1-score*. Essas métricas são úteis para entender como o modelo lida com falsos positivos e falsos negativos.

Tabela 2. Médias macro de precision, recall e f1-score dos modelos avaliados com k-fold

Modelo	Precision	Recall	F1-score
Modelo do Artigo	0.93	0.85	0.87
SVM	0.99	0.99	0.99
KNN	0.98	0.98	0.98
Rede Neural	0.96	0.95	0.96

3.1. Análise dos Resultados

Os resultados mostram uma diferença significativa no desempenho dos modelos quando comparamos os testes realizados com a base original e com os dados caseiros.

Na coluna 'Base original' da Tabela 1, todos os modelos apresentaram bom desempenho com a base original, com acurácias superiores a 98% para os modelos clássicos (SVM, KNN e rede neural) e 86% para o modelo do artigo.

Já na Tabela 1, que apresenta os resultados usando nossos próprios áudios, o desempenho caiu bastante para alguns modelos. O modelo do artigo teve uma acurácia de apenas 20%, e a rede neural ficou com 40%. Isso indica que esses modelos estavam muito ajustados aos dados de treino originais e tiveram dificuldade para generalizar para novos dados, seria um indicativo de overfitting.

Por outro lado, os modelos SVM e KNN mantiveram desempenho excelente, atingindo 100% de acurácia, o que demonstra maior robustez diante de variações na voz.

Esses testes reforçam a importância de avaliar os modelos com dados diferentes daqueles usados no treino. Só assim dá pra saber se o modelo realmente aprendeu algo útil ou só decorou os exemplos que viu.

4. Conclusão

Este trabalho demonstrou a viabilidade da aplicação de técnicas clássicas de aprendizado de máquina no reconhecimento de comandos de voz para controle de robôs móveis. Através da comparação de diferentes abordagens, incluindo SVM, KNN e uma rede neural, foi possível verificar que modelos como SVM e KNN apresentaram excelente desempenho na base original e com dados caseiros, evidenciando maior capacidade de generalização. Em contrapartida, o modelo de rede neural demonstrou sinais de sobreajuste, com desempenho significativamente inferior em cenários fora do conjunto de treino.

Entre os principais aprendizados, destacamos a importância do pré-processamento adequado dos dados, da extração de características mais ricas (como MFCCs combinados com atributos espectrais) e da otimização dos hiperparâmetros para o sucesso do modelo.

Apesar dos avanços, algumas limitações foram encontradas. A rede neural utilizada é simples e não explorou todo o potencial de arquiteturas mais modernas e robustas. Além disso, o conjunto de dados caseiro é restrito, com apenas 10 palavras.

Como trabalhos futuros, propomos:

- Utilizar um conjunto de dados mais amplo, com maior diversidade de palavras e falantes, para melhorar a generalização;
- Investigar modelos mais avançados de redes neurais, como redes convolucionais ou recorrentes, que são mais apropriadas para o processamento de sinais de áudio;
- Realizar treinamento e testes com ruído;

Essas direções visam consolidar um sistema de reconhecimento de comandos de voz mais robusto, preciso e eficiente para aplicações no mundo real.

Referências

Dogra, Y. (2023). Speech commands. Dataset.

Fraunhofer (2023). Fraunhofer machine control via speech recognition. Acessado em: 18 jun. 2025.

Rendyansyah (2022). Implementation of MFCC and SVM for voice command recognition as control on mobile robot. *Journal Ecotipe (Electronics, Control, Telecommunication, Information, and Power Engineering)*, 9(2):192–200.