



Sistema Inteligente de Monitoramento do Ambiente

Aprendizagem de Máquina em Sistemas Embarcados

Élisson França de Souza
José Renilson Almeida da Silva
Monique Moreira Moraes

Introdução



- Importância do monitoramento de ambientes;
- Objetivo do projeto: otimização das condições ambientais para proporcionar um ambiente saudável e confortável.

Descrição do Problema



- Tipo de Problema: classificação e controle de ambiente por meio de aprendizagem de máquina;
- Entradas: intensidade sonora, pressão (KPa), temperatura (°C), luminosidade (lux) e umidade percentual;
- Saída: classificação em relação à qualidade do ambiente (intolerável, tolerável, bom e ótimo);
- Base de Dados: Base de dados fornecida pelo [METEORED](#) + [CLIMATEDATE](#), que são dados fornecidos pelo INMET (Instituto Nacional de Meteorologia);
- Base de Dados: 212 instâncias x 6 atributos.

Descrição do Problema

1	intensidade_sonora	pressao_kpa	temperatura_c	luminosidade_lux	umidade_percentual	classificacao
2	0	101	20	1500	55	toleravel
3	1	99	24	1900	58	toleravel
4	8	104	30	3200	70	intoleravel
5	9	98	27	4500	67	intoleravel
6	19	102	29	3100	69	intoleravel
7	6	100	23	1800	57	otimo
8	23	103	28	3300	68	intoleravel
9	2	101	22	1600	56	otimo
10	5	99	26	2000	59	toleravel
11	9	104	32	3200	72	intoleravel
12	16	98	25	4600	67	intoleravel
13	12	110	32	1200	80	intoleravel
14	7	120	18	1500	60	bom
15	25	102	30	3000	69	intoleravel
16	9	100	21	1700	57	bom
17	11	103	28	3400	69	toleravel
18	6	101	24	1900	58	otimo
19	6	99	20	1300	54	bom
20	18	104	31	3200	71	intoleravel

Etapas da criação do modelo:



1. Importação e Tratamento de Dados:

- Importação de dados: utilizando as bibliotecas pandas e numpy, os dados são importados a partir de um arquivo CSV.
- Tratamento de dados: a classe alvo é convertida em valores numéricos usando codificação de fator. Além disso, as características são selecionadas e divididas em conjuntos de treinamento (70%) e teste (30%) usando o método `train_test_split` da biblioteca `scikit-learn`.

2. Modelo com Redes Neurais:

- Definição do modelo: uma rede neural é construída utilizando a API Keras com três camadas densas. A função de ativação "relu" é aplicada nas duas primeiras camadas, enquanto a função "softmax" é usada na camada de saída para classificação multiclases.
- Compilação e treinamento: o modelo é compilado com a função de perda de entropia cruzada categórica e a métrica de acurácia. Em seguida, é treinado com os dados de treinamento por um número especificado de épocas.

3. Testes do Modelo

- Avaliação do modelo: o modelo treinado é avaliado usando os dados de teste para calcular a perda e a acurácia do teste.
- Predições: os dados de teste são carregados e as previsões são feitas usando o modelo treinado. As previsões são arredondadas para as classes originais usando o mapeamento de fator e são exibidas.

Etapas da criação do modelo:

dense_3_input	input:	[(None, 5)]
InputLayer	output:	[(None, 5)]

dense_3	input:	(None, 5)
Dense	output:	(None, 32)

dense_4	input:	(None, 32)
Dense	output:	(None, 16)

dense_5	input:	(None, 16)
Dense	output:	(None, 4)

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('database.csv')
df.info()

factorized_mapping = dict(enumerate(df['classificacao'].unique()))
df['classificacao'] = pd.factorize(df['classificacao'])[0]
print(factorized_mapping)

features = df[['intensidade_sonora', 'pressao_kpa', 'temperatura_c', 'luminosidade_lux', 'umidade_percentual']]
labels = df['classificacao']

labels_one_hot = pd.get_dummies(labels).values
features_train, features_test, labels_train, labels_test = train_test_split(features, labels_one_hot, test_size=0.3, random_state=0)

model = keras.Sequential([
    keras.layers.Dense(units=32, activation='relu', input_shape=(5,)),
    keras.layers.Dense(units=16, activation='relu'),
    keras.layers.Dense(units=len(factorized_mapping), activation='softmax')
])

model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(features_train, labels_train, epochs=200, validation_data=(features_test, labels_test))
```

Tensor Flow Lite

- Cria um modelo de previsão do tempo usando a biblioteca TensorFlow;
- Converte o modelo em um formato TensorFlow Lite que irá possibilitar o uso em embarcados;
- Salva o modelo convertido em um arquivo `.h` que contém os metadados do modelos que serão compilados junto com código de captura dos sensores para prever novos casos;
- Verificação do tamanho e compatibilidade com a kit de desenvolvimento usado;
- Codifica o modelo em um arquivo de cabeçalho do Arduino.

```
import tensorflow as tf

# Convert the model to the TensorFlow Lite format without quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model to disk
open("weather_model.tflite", "wb").write(tflite_model)
```

5328

```
import os
basic_model_size = os.path.getsize("weather_model.tflite")
print("Model is %d bytes" % basic_model_size)

"""# Encode the Model in an Arduino Header File"""

!echo "const unsigned char model[] = {" > /content/model.h
!cat weather_model.tflite | xxd -i >> /content/model.h
!echo "};" >> /content/model.h
```

Model is 5328 bytes

Classificação via Arduino (teste inicial)

<code>int64_t entrada[18][5] = {</code>		<code>toleravel: 0.8925</code>	<code> </code>	<code>intoleravel: 0.0742</code>	<code> </code>	<code>otimo: 0.0333</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{10,101,25,1200,90},</code>	TOLERÁVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0010</code>	<code> </code>	<code>otimo: 0.1498</code>	<code> </code>	<code>bom: 0.8492</code>
<code>{12,101,32,1000,50},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0040</code>	<code> </code>	<code>otimo: 0.0003</code>	<code> </code>	<code>bom: 0.9957</code>
<code>{20,101,25,4000,50},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 1.0000</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{12,101,35,2000,20},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.9599</code>	<code> </code>	<code>otimo: 0.0004</code>	<code> </code>	<code>bom: 0.0397</code>
<code>{6,100,25,1300,40},</code>	BOM	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0080</code>	<code> </code>	<code>otimo: 0.0017</code>	<code> </code>	<code>bom: 0.9903</code>
<code>{5,101,24,1000,30},</code>	BOM	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0011</code>	<code> </code>	<code>otimo: 0.0001</code>	<code> </code>	<code>bom: 0.9988</code>
<code>{10,101,27,100,50},</code>	TOLERÁVEL	<code>toleravel: 1.0000</code>	<code> </code>	<code>intoleravel: 0.0000</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{6,108,26,1500,60},</code>	TOLERÁVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0031</code>	<code> </code>	<code>otimo: 0.0109</code>	<code> </code>	<code>bom: 0.9860</code>
<code>{8,98,26,1600,60},</code>	BOM	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0487</code>	<code> </code>	<code>otimo: 0.1153</code>	<code> </code>	<code>bom: 0.8360</code>
<code>{7,101,32,1000,10},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0035</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.9965</code>
<code>{12,102,29,3000,20},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 1.0000</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{9,103,27,3000,60},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 1.0000</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{13,102,30,3000,80},</code>	INTOLERAVEL	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 1.0000</code>	<code> </code>	<code>otimo: 0.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{6,101,20,1000,40},</code>	BOM	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0020</code>	<code> </code>	<code>otimo: 0.0005</code>	<code> </code>	<code>bom: 0.9976</code>
<code>{4,102,17,1000,20},</code>	BOM	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0058</code>	<code> </code>	<code>otimo: 0.0001</code>	<code> </code>	<code>bom: 0.9942</code>
<code>{1,98,16,400,100},</code>	ÓTIMO	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0000</code>	<code> </code>	<code>otimo: 1.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{2,100,22,300,90},</code>	ÓTIMO	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0000</code>	<code> </code>	<code>otimo: 1.0000</code>	<code> </code>	<code>bom: 0.0000</code>
<code>{0,95,21,250,80},</code>	ÓTIMO	<code>toleravel: 0.0000</code>	<code> </code>	<code>intoleravel: 0.0000</code>	<code> </code>	<code>otimo: 0.9999</code>	<code> </code>	<code>bom: 0.0001</code>

Classificação via Arduino (teste inicial)



- Testes:
 - 12 acertos;
 - 6 erros;
 - Acurácia:
 - Geral: 66%;
 - Boa: 80%;
 - Ótima: 100%;
 - Tolerável: 66%;
 - Intolerável: 38%.

Classificação via Arduino



- Obtenção de dados via sensores:
 - MP34DT05 (microfone):
 - Adequação de amostras (512 amostras) para intensidade sonora.
 - APDS9960 (luz):
 - Valor lido é próximo a lumens.
 - LPS22HB (barométrico):
 - Valor lido em kPa.
 - DHT11 (temperatura e umidade):
 - Temperatura em celsius e umidade em percentual.

Classificação via Arduino



```
Intensidade sonora = 8
Pressao = 100.26 kPa
Temperature = 23.00 °C
Light = 4097
Umidade = 19.00 %
toleravel: 0.0000   ||||   intoleravel: 1.0000   ||||   otimo: 0.0000   ||||   bom: 0.0000
Intensidade sonora = 5
Pressao = 100.26 kPa
Temperature = 22.00 °C
Light = 13
Umidade = 20.00 %
toleravel: 0.9989   ||||   intoleravel: 0.0011   ||||   otimo: 0.0000   ||||   bom: 0.0000
Intensidade sonora = 4
Pressao = 100.27 kPa
Temperature = 24.00 °C
Light = 213
Umidade = 19.00 %
toleravel: 0.0048   ||||   intoleravel: 0.0000   ||||   otimo: 0.0000   ||||   bom: 0.9951
```

Obrigado!

