Classificação de posição de uma pessoa a partir de uma imagem

Introdução

Este experimento foi desenvolvido por Vilson Rodrigues Câmara Neto. Nele, foi tentado classificar se uma pessoa está em pé ou sentada a partir de uma base de dados. Está base de dados foi gerada a partir de uma vídeo onde nele a pessoa se move, variando entre ficar em pé e sentada. Foi utilizada a *Pose Estimation*, um algoritmo de classificação da biblioteca *TensorFlow*, para gerar os pontos a partir do vídeo. Depois foi usada a *Support Vector Machine - SVM* para classificar se ela está em pé ou sentada. O algoritmo de Machine Learning *SVM*, tem como princípio classificar, aplicando uma transformação linear num hiperespaço e traçando um hiperplano que separe o conjunto de pontos. Foi ajustado os parâmetros *c* e *gamma* de modo que não gerasse *Overffiting* ou *Underffiing***.** O *c* diz o quão de curvas a *SVM* pode realizar, já o *gamma* diz o quão ajustado o modelo pode ser.

Metodologia

Usando um celular, foi filmado uma pessoa variando entre em pé e sentada. Depois foi rodado um código usando o *Pose Estimation* para gerar os pontos, em seguida o *.CSV* foi gerado.

Os seguintes atributos foram utilizados para poder classificar:

- nose_x, nose_y: Posição do nariz
- leftEye_x, leftEye_y: Posição do olho esquerdo.
- rightEye_x ,rightEye_y: Posição do olho direito.
- leftEar_x, leftEar_y: Posição da orelha esquerda.
- rightEar_x, rightEar_y: Posição da orelha direita.
- leftShoulder_x, leftShoulder_y: Posição do ombro esquerdo.
- rightShoulder_x, rightShoulder_y: Posição do ombro direito.
- leftElbow_x, leftElbow_y: Posição do cotovelo esquerdo.
- rightElbow_x, rightElbow_y: Posição do cotovelo direito.
- leftWrist_x, leftWrist_y: Posição do pulso esquerdo.
- rightWrist_x, rightWrist_y: Posição do pulso direito.
- leftHip_x, leftHip_y: Posição do quadril esquerdo.
- rightHip_x, rightHip_y: Posição do quadril direito.
- leftKnee_x, leftKnee_y: Posição do joelho esquerdo.

- rightKnee_x, rightKnee_y: Posição do joelho direito.
- leftAnkle_x, leftAnkle_y: Posição do tornozelo esquerdo.
- rightAnkle_x, rightAnkle_y: Posição do tornozelo direito.

Foi separado o conjunto de dados em 2, treinamento e teste, numa divisão de 75% de treino e 25% de teste

Códigos

Arquivo gerador do *.CSV*

```
import requests as req
import numpy as np

gitraw = "https://raw.githubusercontent.com/"
user = "Lucasgsr14/"
repository = "Topicos-Avancados-de-Informatica-I/"
branch = "master/"
folder = "unidade%2001/Pose%20Estimation/Imagens/img"
ext = ".jpg"
arrImages = []
arrPos = []

for i in range(1,107):
  arrImages.append(gitraw+user+repository+branch+folder+str(i)+ext)

r = req.get("http://poseestimation.herokuapp.com/header")
header = r.text.replace('[', '').replace(']', '')+",estado"

for i in range(106):
  r = req.get("http://poseestimation.herokuapp.com/estimate?url="+arrImages[i])
  arrPos.append(str(i+1)+','+r.text.replace('[', '').replace(']', ''))

for i in range(106):
  if i<47:
    arrPos[i]+=",levantado"
  else:
    arrPos[i]+=",sentado"

print(np.transpose(arrPos))
```

Separação dos conjuntos de treino e teste:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

SVM e seus parâmetros:

```
# Fitting SVM to the Training set
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline

#classifier = SVC(kernel = 'linear', random_state = 0)
classifier = Pipeline((
("scaler", StandardScaler()),
("svm_clf", SVC(kernel="rbf", gamma=0.5, C=0.5))
))



classifier.fit(X_train, y_train)
```

Foi utilizado o parâmetro *gamma* igual a 0.5 e o *C* também igual a 0.5, objetivo foi treinar o modelo sem gerar *overfitting* ou *underfitting*.

Testando o modelo:

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)

print(y_test[0:35])
print(y_pred[0:35])
```

Experimentos

Com uma acurácia de 84%, podemos dizer que o experimento teve um êxito muito bom. Os parâmetros *C* e *Gamma* tiveram um ajuste que impediu o *Overfitting* e o *Underfitting*, o que gerou um bom resultado com os valores preditos.

Referências

GOOGLE, ***Pose Estimation*.** Disponível em:<https://www.tensorflow.org/lite/models/pose_estimation/overview>. Acesso em: 11 set. 2019.

GÉRON, Aurélien. ***Hands on: Machine Learnig with Sckit-Learn and Tensor Flow***. Ed: 1. Alta Books.