오픈소스 프로젝트 데이터 분석

데이터 수집 및 활용, 분석

openpose를 활용한 이미지 처리

과목명	오픈소스 전문프로젝트
담당교수	박수창 교수님
팀원	3조 2016038025 박상협 2016038028 배석훈 2016038037 김성훈 2016038039 최건희


충북대학교
CHUNGBUK NATIONAL UNIVERSITY

목차

데이터 활용

운동 자세(스쿼트, 런지 등)의 정확도를 측정하기 위해 먼저 올바른 자세와 올바르지 못한 자세의 여러 장의 사진을 찍었습니다. 사진 촬영 이후에 올바른 자세와 올바르지 못한 자세로 나눈 뒤, 사진에서 신체 points(neck, wrist, knee, ankle 등)을 추출하여 csv 파일로 변환하고, 각 사진에서 일정한 방법으로 각각 point들의 연결선의 각도로 변환하였습니다. 그리고 분류기(퍼셉트론)를 사용하여 올바른 자세인지 아닌지를 확인할 수 있도록 구성하였습니다.

데이터 추출

사용 프로그래밍 언어는 Python 3 이상의 버전을 지원하는 Anaconda3 Jupyter Notebook을 사용하였습니다. 사용한 라이브러리는 os, cv2, pandas를 사용하였습니다.

os	이미지 파일 이름을 불러오기 위한 용도
cv2(opencv library)	계산을 위한 이미지 전처리
pandas	데이터 시각화 및 파일로 저장

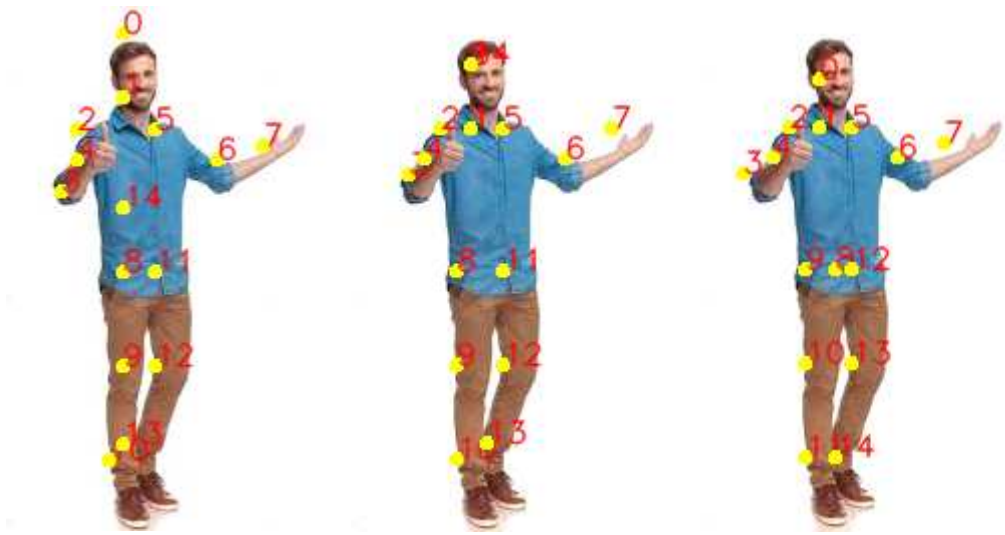
올바른 동작인지 확인하기 위해서 openpose의 이미 학습된 모델을 사용하였습니다.
openpose는 OpenCV의 human pose estimation에 CNN(Convolution Neural Network)을 사용해서 보다 정확한 pose estimate를 하기 위한 Computer vision 분야의 기술 중 하나입니다.

ANN(Artificial), DNN(Deep), CNN(Convolution)의 차이점
ANN, 인공신경망은 사람의 신경망 구조를 모방하여 기계 학습을 구현하는 방법입니다. ANN에서 사용자는 입력 대비 출력을 알 수 있지만, 그 내부는 어떻게 작동하는지 알 수 없습니다. 중간 과정에는 특수한 알고리즘이 아닌 어떠한 능력만이 포함되어 있을 뿐입니다. 이 중간 과정을 여러 개로 늘린 모델을 DNN, 심층신경망이라고 합니다. ANN은 데이터에서 지식을 추출하여 결과를 추출하였지만, 데이터의 패턴을 추출하여 특징들의 패턴을 파악하는 구조도 있습니다. 이를 CNN, 합성곱신경망이라고 합니다.

각 모델은 다른 points를 사용하는 동시에 비슷한 지점(point) 측정에서도 다른 결과를

도출해냅니다. 각 모델은 다른 weights를 사용하며, 더 큰 weight를 사용하는 모델에서는 정확한 측정을 할 수 있지만, 그만큼 시간 소요도 커지게 됩니다. 저희는 사진 한 장의 모델을 측정하여 오래 걸리지 않는다고 판단하여 weight 584,000의 body_25 모델을 사용하였습니다.

MPI(weight 160000), COCO(weight 440000), BODY_25(weight 584000) 실행 결과



mpi(160,000) coco(440,000) body_25(584,000)

point 추출 과정

1. OpenCV 라이브러리를 사용하여 이미지 로드

2. 이미지에서 point를 추출하기 위한 전처리

이미지 크기에 맞춰 blob을 구성.

blob은 인접한 화소들을 비슷한 화소끼리 묶은 여러 개의 객체.

3. 이미 학습된 모델로 신경망 구성

MPI, COCO, BODY_25중에 알맞은 모델과 weight 선택.

OpenCV의 readNetFromCaffe 메서드를 이용하여 net(신경망) 구성.

4. 전처리된 이미지를 신경망에 입력함

5. 신경망 출력

net.setInput과 net.forward로 입력 및 출력.

6. 출력값에서 원본 사진의 point 좌표(x,y) 추출

출력 map에서 원본 사진의 크기만큼 확장하고,

point 들에도 확장을 적용하면 원본 사진에서의 x와 y값을 추출해낼 수 있다.

```
# fashion_pose.py : MPII를 사용한 신체부위 검출
import pandas as pd
import cv2
import os

# 각 파일 path
protoFile ="./body_25/pose_deploy.prototxt"
weightsFile ="./body_25/pose_iter_584000.caffemodel"
# 위의 path에 있는 network 불러오기
net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

path ="../squat_x"# 사진 파일 주소
file_list = os.listdir(path) # 사진 파일명 리스트
num_list = [1,5,12,13,14] # 스쿼트
columns_name = ['1x','1y','5x','5y','12x','12y','13x','13y','14x','14y'] # 스쿼트
df = pd.DataFrame(columns=columns_name);
for file in file_list:
    # 이미지 읽어오기
    print(len(df), end=' ')
    print(file)
    image = cv2.imread(path +'/'+file)
    # frame.shape = 불러온 이미지에서 height, width, color 받아옴
    imageHeight, imageWidth, _ = image.shape
    # network에 넣기위해 전처리
    inpBlob = cv2.dnn.blobFromImage(image, 1.0 /255, (imageWidth,
imageHeight), (0, 0, 0), swapRB=False, crop=False)
    # network에 넣어주기
    net.setInput(inpBlob)
    # 결과 받아오기
    output = net.forward()
    # output.shape[0] = 이미지 ID, [1] = 출력 맵의 높이, [2] = 너비
    H = output.shape[2]
    W = output.shape[3]
    #print("이미지 ID : ", len(output[0]), ", H : ", output.shape[2], ", W :
",output.shape[3]) # 이미지 ID
    # 키포인트 검출시 이미지에 그려줌
    row = []
    for i in num_list:
        # 해당 신체부위 신뢰도 얻음.
        probMap = output[0, i, :, :]
        # global 최대값 찾기
        minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)
        # 원래 이미지에 맞게 점 위치 변경
        x = (imageWidth * point[0]) / W
        y = (imageHeight * point[1]) / H
        #print(i, " ", x, " ", y);

        row.append(x);
        row.append(y);
        # point를 dataframe으로 변경
    df = df.append(pd.Series(row, index=df.columns), ignore_index=True)
    df.to_csv('squat_x.csv', mode='w')
```

데이터 전처리

분류기에 입력값에 넣기 위한 데이터를 전처리하기 위해 Anaconda3 Jupyter Notebook을 사용하였습니다. 라이브러리는 Math, train_test_split, StandardScaler를 사용하였습니다.

StandardScaler	Dataset을 정규화하기 위한 라이브러리
Math	삼각함수 math.atan2를 사용하기 위한 라이브러리
train_test_split	Data를 Traindata, Testdata, Trainlable, Testlable로 일정 비율을 정하여 분리하는 라이브러리

데이터 추출에서 받아온 Keypoint의 좌표 값(x,y)을 통해서 두 점 사이의 각도를 삼각함수 arctan을 사용하여 두 점 사이의 각도를 X축을 기준으로 math.atan2() * 180 / math.pi를 통해서 구할 수 있습니다. 또한, rain_test_split을 통하여 Dataset을 Traindata, Testdata, Trainlable, Testlable로 나누어 훈련용, 테스트용을 구분합니다.

** 각 자세마다 관절의 각도가 어느 정도 일정하기에 Keypoint들의 각도를 입력값으로 받아 M-L-P 모델을 생성합니다.

모델 선택 및 학습

모델은 층이 하나이고 크기가 100개인 MLPClassifier Model을 사용합니다. 생성할 모델의 층과 크기를 정한 기준은 사이킷런(Scikit-learn)의 MLPClassifier를 통해서 Mini Dataset으로 훈련한 결과 가장 학습률이 잘 나오는 크기와 깊이를 선택한 결과입니다. 모델의 학습은 전처리한 데이터를 입력값으로 받아 오차 함수를 통해 반복적으로 Update를 합니다. Update는 경사 하강법을 사용합니다.

0	-56.053959	180.000000	-72.917668	1
1	-53.983212	-177.016129	-75.243343	1
2	-55.526432	-174.048356	-69.838440	1
3	-52.163547	-177.299865	-72.189296	1
4	-54.427257	-175.953537	-80.040317	1
...	...	...	...	...
95	-39.288102	158.388815	-65.564428	0
96	-39.288102	160.881913	-90.000000	0
97	-37.778677	161.730486	-64.438999	0
98	-35.050521	161.730486	-64.438999	0
99	-43.326424	160.733053	-75.832944	0

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
                                                    random_state=1)
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.fit_transform(X_test)
mlp = MLPClassifier(hidden_layer_sizes=(100),solver='lbfgs', random_state=0).fit(X_train_std, y_train)



mlp.score(X_test_std,y_test)
```

```
C:₩Users₩82107₩Anaconda3₩lib₩site-packages₩sklearn₩neural_network₩multilayer_perceptron.py:921: DataConversion
Warning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

0.94

** Mini Dataset으론 정답 100개와 오답 100개를 train_test_split을 사용하여 무작위로 섞어 구성하였고 Traindata 70%, Testdata 30%로 분리하였습니다. 그 후 StandardScaler를 통하여 Mini Dataset을 정규화한 후 성능을 추출하였습니다. 각도 값으로는 스쿼트의 경우 간단하게 어깨, 골반과 골반, 무릎과 무릎, 발목의 3개의 각도를 사용하여 테스트를 진행하였으며 모든 데이터를 처리하고 분석이 완료되면 여러 각도를 통해서 성능이 좋은 입력값을 사용합니다.

모델 검증 및 선택

데이터를 각 운동에 대하여 1000장을 수집하였지만, 데이터가 분류기의 학습에 충분하게 많다고 할 수 없다. 따라서 모델이 일반화가 어느 정도 잘 되었는지 확인할 필요가 있기 때문에 K-fold Cross Validation을 사용하여 Dataset를 최대한 활용하여 교차 검증을 실시하여 가장 성능이 좋은 모델을 Validation set을 사용하여 선택한다. 이 경우 특정 모델에서 발생할 수 있는 Overfitting을 방지할 수 있으며 통계적으로 모델의 성능을 검증할 수 있다.

데이터 처리 코드

```
# 1. Data 각도값 추출 및, 입력 data와 정답 lable로 분리

import pandas as pd
import math
df  = pd.read_csv('squat_o.csv')
X = df.iloc[:,1:]
X = X.values.tolist()  ## DataFrame을 List값으로 추출
array = []
for i in range(0,100): ## 좌표값을 math.atan2(y,x) * 180 / math.pi로 두 점 사이의 각도값 추출
    array.append([
                math.atan2(X[i][3]-X[i][5],X[i][4]-X[i][2]) *180 / math.pi,
                math.atan2(X[i][5]-X[i][7],X[i][6]-X[i][4]) *180 / math.pi,
                math.atan2(X[i][7]-X[i][9],X[i][8]-X[i][6]) *180 / math.pi
                 ,1]
                )
result_o = pd.DataFrame(array) ## List를 DataFrame으로 변환
df = pd.read_csv('squat_x.csv')
X = df.iloc[:,1:]
X = X.values.tolist()
array = []
for i in range(0,100): ## 각도값 추출
    array.append([
                math.atan2(X[i][3]-X[i][5],X[i][4]-X[i][2]) *180 / math.pi,
                math.atan2(X[i][5]-X[i][7],X[i][6]-X[i][4]) *180 / math.pi,
                math.atan2(X[i][7]-X[i][9],X[i][8]-X[i][6]) *180 / math.pi
                 ,0]
                )

result_x = pd.DataFrame(array) ## List를 DataFrame으로 변환
result_ox  =  result_o.append(result_x)  ##  DataFrame  두개를  합쳐서  새로운 DataFrame 정의
X = result_ox.iloc[0:,:3].values ## 입력 Data 추출
y = result_ox.iloc[0:,[3]]  ## 정답 Lable 추출
```

```
# 2. trainset, testset 분리 및 정규화 후 모델 학습 및 성능 확인

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, ## Dataset을 무작위로 섞어 분리
                                                    random_state=1)
stdsc = StandardScaler() ## 정규화 객체 생성
X_train_std = stdsc.fit_transform(X_train) ## 정규화 진행
X_test_std = stdsc.fit_transform(X_test) ## 정규화 진행
mlp             =             MLPClassifier(hidden_layer_sizes=(100),solver='lbfgs', random_state=0).fit(X_train_std, y_train) ## 모델 학습
mlp.score(X_test_std,y_test) ## Testdata에 대하여 분류기의 정답률을 추출
```

```
# 3. 오답데이터 46개를 추가적으로 test_data로 사용해 성능 확인

df = pd.read_csv('squat_x.csv')
X = df.iloc[:,1:]
X = X.values.tolist()
array = []
for i in range(100,146): ## 추가 오답 데이터 46개의 각도값을 추출하여 모델 일반화 확인
    array.append([
                  math.atan2(X[i][3]-X[i][5],X[i][4]-X[i][2]) *180 / math.pi,
                  math.atan2(X[i][5]-X[i][7],X[i][6]-X[i][4]) *180 / math.pi,
                  math.atan2(X[i][7]-X[i][9],X[i][8]-X[i][6]) *180 / math.pi
                   ,0]
                  )
result_xxx = pd.DataFrame(array)
result_xxx
A = result_xxx.iloc[:,0:3].values
b = result_xxx.iloc[:,[3]].values
A_std = stdsc.fit_transform(A)
mlp.score(A_std,b)
```

비고 및 고찰

1. 이미지에서 point 추출을 위해 가장 weight가 높은 모델을 사용한 이유.

사진을 찍을 때, 가장 사용자에게 얻기 좋은 데이터가 90도 돌아서 옆면이 보이는 사진을 생각했다. 정면 또는 옆면이 보이도록 찍도록 요구하는 것이 사용자로부터 가장 비슷한 데이터들을 수집할 수 있다고 생각했기에 그 중 이미지에서 얻을 수 있는 2D 자료가 더 유의미한 옆면 자료를 수집하기로 정했다. 하지만, 우리가 가진 모델로 옆면의 points를 추출해본 결과 생각보다 올바른 위치에 point가 추출되지 않았고, 옆면의 데이터를 사용할 수 있는 방법을 모색하다가 더 정확한 자세를 측정할 수 있는 모델을 찾았다. 사용자에게는 사진 한 장만 분석하면 되는 비교적 길지 않은 시간만 필요하기에 시간 소요가 늘어나더라도 더 잘되는 모델을 사용하였다.

2. 각도 값으로 변환한 이유.

좌푯값이 아닌 각도 값을 입력값으로 사용하는 이유는 좌푯값은 자세가 일정해도 사진이 찍힌 쪽이 중앙에서 왼쪽인지 오른쪽인지에 따라 달라질 수 있지만, 각도 값은 각각 자세마다 관절의 각도가 자세가 일치한다면 일정하기에 Keypoint들의 각도를 입력값으로 받아 M-L-P 모델을 생성한다.

3. 각도 값을 정할 때 각도가 역전되었었던 문제

수학에서의 좌표는 y값이 보통 위쪽이 크고, 아래쪽이 작다. 하지만 Keypoint의 좌표(x,y)는 사진에 밑에 있는 점일수록 y값이 커지기 때문에 math.atan2()를 사용할 때 좌표 값이 반전이 되지 않게 고려해줘야 한다.

3. 데이터 정규화를 하지 않아서 생겼던 문제

데이터를 정규화하지 않았을 때 값들이 너무 크고, 모두 양수이기 때문에 정규화를 통해서 입력값을 줄여 가중치 Update가 Overshootiong 되지 않게 하며 입력값의 분포를 평면상 중앙으로 모아 학습이 잘되도록 한다.

4. 가장 결과가 잘 나오는 조건

MLP모델의 층과 크기를 고려할 때 적당한 값의 여러 케이스를 시도해본 결과 데이터의 값을 고려해서 층이 한 개이고 크기가 100개인 M-L-P모델이 Mini dataset에서 가장 좋은 성능을 보였다.

5. scikit-learn을 사용한 이유.

위의 검증 사항을 위해서 지금 당장 직접 프로그래밍한 MLP 모델이 없기 때문에, 사이킷런의 MLPClassifier를 사용하여 성능 및 모델의 세부 사항을 고려하였다.