Uncovering bias in the PlantVillage dataset_code

June 7, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import glob
import cv2
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from keras.datasets import mnist
```

[2]:
```
main_path = 'PlantVillage-Dataset-master/raw/color/'
```

[3]:
```
class_folders = []
class_names = []
for class_folder in glob.glob(main_path + '*'):
    class_folders.append(class_folder)
    class_names.append(class_folder.split('/')[-1])
```

[4]:
```
images = []
y = []
y_names = []
for i in range(len(class_names)):
    for im_path in glob.glob(class_folders[i] + '/*'):
        im = cv2.imread(im_path)
        if im is None:continue
        if im.shape != (256, 256, 3):continue
        images.append(im)
        y.append(i)
        y_names.append(class_folders[i].split('/')[-1])

images = np.array(images)
y_names = np.array(y_names)
y = np.array(y)
```

[5]:
```
images.shape, y.shape
```

[5]:
```
((54305, 256, 256, 3), (54305,))
```

[6]:
```
def feature_extractor(images):
    mid_point = int(images.shape[1]/2)
    tl = images[:, 0, 0, :]
    tr = images[:, 0, -1, :]
    bl = images[:, -1, 0, :]
    br = images[:, -1, -1, :]
    lm = images[:, mid_point, 0, :]
    tm = images[:, 0, mid_point, :]
    bm = images[:, -1, mid_point, :]
    rm = images[:, mid_point, -1, :]
    X = np.stack((tl, tr, bl, br, lm, tm, bm, rm), axis=1)
    X = X.reshape(X.shape[0], -1)
    return X
```

[7]:
```
X = feature_extractor(images)
```

[8]:
```
X.shape, y.shape
```

[8]:
```
((54305, 24), (54305,))
```

[9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
```

[10]:
```
clf = RandomForestClassifier(random_state=0)
clf.fit(X_train, y_train)
```

[10]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

[11]:
```
acc_result = clf.score(X_test, y_test)
np.round(acc_result*100, 1)
```

[11]:
```
49.0
```

[12]:
```
# random guess
np.round(100/38, 1)
```

[12]:
```
2.6
```

0.1 MNIST

[13]:
```
(X_train_mnist, y_train_mnist), (X_test_mnist, y_test_mnist) = mnist.load_data()
```

[14]:
```
X_train_mnist.shape, X_test_mnist.shape
```

[14]:
```
((60000, 28, 28), (10000, 28, 28))
```

[15]:
```
X_train_mnist_8px = feature_extractor(X_train_mnist.reshape(60000, 28, 28, 1))
X_test_mnist_8px = feature_extractor(X_test_mnist.reshape(10000, 28, 28, 1))
```

[16]:
```
X_train_mnist_8px.shape, X_test_mnist_8px.shape
```

[16]:
```
((60000, 8), (10000, 8))
```

[17]:
```
clf_mnist = RandomForestClassifier(random_state=0)
clf_mnist.fit(X_train_mnist_8px, y_train_mnist)
```

[17]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

[18]:
```
acc_result = clf_mnist.score(X_test_mnist_8px, y_test_mnist)
np.round(acc_result*100, 1)
```

[18]:
```
11.7
```

0.2 Blur

[19]:
```
y = []
X_im_sharpness = []
X_bg_sharpness = []
X_seg_sharpness = []
for i in range(len(class_names)):
    for im_path in glob.glob(class_folders[i] + '/*.JPG'):

        # path of the corresponding foreground image
        splitted = im_path.split('/')
        color_name = splitted[-1]
        segmented_name = color_name[:-4] + '_final_masked.jpg'
        splitted[2] = 'segmented'
```

```
        splitted[-1] = segmented_name
        seg_path = '/'.join(splitted)

        im = cv2.imread(im_path)
        seg = cv2.imread(seg_path)

        if im is None:continue
        if seg is None:continue

        if im.shape == (256,256,3):
            if seg.shape == (256,256,3):
                c1 = seg[:, :, 0]==0
                c2 = seg[:, :, 1]==0
                c3 = seg[:, :, 2]==0
                mask = c1*c2*c3
                bg = np.zeros_like(im)
                bg[mask] = im[mask]

                im_sharpness = cv2.Laplacian(im, cv2.CV_64F).var()
                seg_sharpness = cv2.Laplacian(seg, cv2.CV_64F).var()
                bg_sharpness = cv2.Laplacian(bg, cv2.CV_64F).var()

                X_im_sharpness.append(im_sharpness)
                X_seg_sharpness.append(seg_sharpness)
                X_bg_sharpness.append(bg_sharpness)

                y.append(i)
```

[20]:
```
y = np.array(y)
X_im_sharpness = np.array(X_im_sharpness)
X_bg_sharpness = np.array(X_bg_sharpness)
X_seg_sharpness = np.array(X_seg_sharpness)

print(y.shape)
print(X_im_sharpness.shape)
print(X_bg_sharpness.shape)
print(X_seg_sharpness.shape)
```

```
(51607,)
(51607,)
(51607,)
(51607,)
```

[21]:
```
results = []
for X in [X_im_sharpness, X_bg_sharpness, X_seg_sharpness]:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

```
    clf = RandomForestClassifier(random_state=0)
    clf.fit(X_train.reshape(-1,1), y_train)

    acc_result = clf.score(X_test.reshape(-1,1), y_test)
    results.append(acc_result)
```

[22]:
```
print('FG+BG:', np.round(results[0]*100, 1))
print('BG:', np.round(results[1]*100, 1))
print('FG:', np.round(results[2]*100, 1))
```

```
FG+BG: 11.7
BG: 10.8
FG: 10.0
```

[ ]:
```

```