plain_svm_analysis

December 13, 2024

[1]:
```
# Install basic modules and make sure they are available with the latest pip␣
↪version.
# Always updating PIP could be either good or bad, you just have to choose one␣
↪base on the situation around.

# I use --quiet and --no-warn-script-location to hide the output of my␣
↪directory paths
import sys
import os

!{sys.executable} -m pip install --upgrade pip matplotlib numpy␣
↪tensorflow-macos tensorflow-metal scikit-learn --quiet␣
↪--no-warn-script-location
```

[2]:
```
import os
import glob
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, classification_report
```

[6]:
```
# Define dataset path
dataset_path = os.path.join(os.getcwd().replace("investigation",␣
↪"kaggledataset"), 'garbage_classification')

# Load all images and labels
image_data = []
labels = []
class_names = sorted(os.listdir(dataset_path))
print(f"Classes: {class_names}")

for class_idx, class_name in enumerate(class_names):
    class_folder = os.path.join(dataset_path, class_name)
    if os.path.isdir(class_folder):
```

```
        for img_file in glob.glob(os.path.join(class_folder, "*.jpg")):
            try:
                # Open the image, resize, and flatten
                img = Image.open(img_file).convert("RGB").resize((256, 256))
                flattened_img = np.array(img).flatten()  # Ensure 1D array
                image_data.append(flattened_img)
                labels.append(class_idx)
            except Exception as e:
                print(f"Error loading image {img_file}: {e}")

# Convert to NumPy arrays
image_data = np.array(image_data, dtype="float32") / 255.0  # Normalize to␣
 ↪range [0, 1]
labels = np.array(labels)

# Confirm the shape of image_data
print(f"Shape of image_data: {image_data.shape}")  # Should be (num_samples,␣
 ↪256*256*3)
```

```
Classes: ['battery', 'biological', 'brown-glass', 'cardboard', 'clothes',
'green-glass', 'metal', 'paper', 'plastic', 'shoes', 'trash', 'white-glass']
Shape of image_data: (15515, 196608)
```

[7]:
```
# Split data into train/test sets
X_train, X_test, y_train, y_test = train_test_split(
    image_data, labels, test_size=0.2, random_state=42, stratify=labels
)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Now X_train is 2D
X_test = scaler.transform(X_test)

# Apply PCA for dimensionality reduction (optional)
pca = PCA(n_components=100)  # Reduce to 100 features
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

[8]:
```
# Train an SVM
svm_model = SVC(kernel='linear', class_weight='balanced', probability=True)
svm_model.fit(X_train, y_train)
```

[8]:
```
SVC(class_weight='balanced', kernel='linear', probability=True)
```

[9]:
```
# Evaluate
y_pred = svm_model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred, target_names=class_names))
```

```
Accuracy: 0.5514018691588785
              precision    recall  f1-score   support

     battery       0.42      0.60      0.49       189
  biological       0.39      0.52      0.45       197
 brown-glass       0.36      0.53      0.43       122
   cardboard       0.51      0.59      0.55       178
     clothes       0.84      0.75      0.79      1065
 green-glass       0.71      0.80      0.75       126
       metal       0.25      0.27      0.26       154
       paper       0.45      0.47      0.46       210
     plastic       0.36      0.29      0.32       173
       shoes       0.42      0.28      0.34       395
       trash       0.47      0.58      0.52       139
 white-glass       0.37      0.30      0.33       155

    accuracy                           0.55      3103
   macro avg       0.46      0.50      0.47      3103
weighted avg       0.57      0.55      0.55      3103
```

[ ]: