مینا جمشیدی
۴۰۱۱۲۶۲۱۳۴
مبینا محمدی
۴۰۱۱۲۶۲۰۹۸

مستندات پروژه clustering

۱ فاز اول: استخراج ویژگی

۱.۱ مقدمه

در این فاز از پروژه، هدف پردازش تصاویر و استخراج ویژگی‌های مفید از آن‌ها می‌باشد. ویژگی‌های استخراج شده باید به گونه‌ای باشند که امکان تفکیک بهتر کلاس‌های مختلف را فراهم کنند.

۲.۱ ویژگی‌های استخراج شده

در این پروژه، شش ویژگی مختلف از تصاویر استخراج شده است:

- **ویژگی‌های رنگی**: میانگین و انحراف معیار کانال‌های R، G و B
- **ویژگی‌های آماری**: میانگین و واریانس تصویر خاکستری
- **تراکم لبه‌ها**: با استفاده از فیلتر سوبل
- **ویژگی‌های بافتی**: کنتراست و همگنی با استفاده از ماتریس هم‌رخساره خاکستری (GLCM)

۳.۱ توضیحات کد

۱.۳.۱ تابع extract_features

این تابع مسئول استخراج ویژگی‌ها از هر تصویر است:

```
def extract_features(image):
features = {}

#               128x128
image = cv2.resize(image, (128, 128))

#
for i, color in enumerate(["R", "G", "B"]):
```

```
features[f"mean_{color}"] = np.mean(image[:, :, i])
features[f"std_{color}"] = np.std(image[:, :, i])

#
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#
features["mean_gray"] = np.mean(gray)
features["var_gray"] = np.var(gray)

#
edges = sobel(gray)
features["edge_density"] = np.sum(edges) / (128 * 128)

#            GLCM
glcm = graycomatrix(gray, [1], [0], symmetric=True, normed=True)
features["contrast"] = graycoprops(glcm, "contrast")[0, 0]
features["homogeneity"] = graycoprops(glcm, "homogeneity")[0, 0]

return features
```

۲.۳.۱ بخش اصلی کد

این بخش تصاویر را از پوشه‌های مختلف خوانده و ویژگی‌های آن‌ها را استخراج می‌کند:

```
data = []
for category in os.listdir(DATASET_PATH):
category_path = os.path.join(DATASET_PATH, category)
if not os.path.isdir(category_path):
continue

for image_name in os.listdir(category_path):
image_path = os.path.join(category_path, image_name)
image = cv2.imread(image_path)
if image is None:
continue

features = extract_features(image)
```

```
features["label"] = category
data.append(features)

df = pd.DataFrame(data)
df.to_csv("features.csv", index=False)
```

۴.۱ خروجی

ویژگی‌های استخراج شده در یک فایل CSV با نام features.csv ذخیره می‌شوند. هر سطر این فایل مربوط به یک تصویر و شامل ویژگی‌های استخراج شده و برچسب کلاس آن می‌باشد.

۵.۱ نتیجه‌گیری

در این فاز، با موفقیت ویژگی‌های مختلفی از تصاویر استخراج شد که می‌تواند در مراحل بعدی برای طبقه‌بندی تصاویر مورد استفاده قرار گیرد. ویژگی‌های استخراج شده شامل ویژگی‌های رنگی، آماری، لبه‌ای و بافتی می‌باشند که همگی به صورت دستی پیاده‌سازی شده‌اند.

۲ فاز دوم: انتخاب ویژگی‌ها

۱.۲ اهداف اصلی

- انتخاب حداقل ۳ ویژگی بهینه از میان ویژگی‌های استخراج شده
- محاسبه ماتریس همبستگی بین ویژگی‌ها
- تعیین آستانه بهینه برای انتخاب ویژگی‌ها

۲.۲ بخش اصلی کد

```
# Calculate correlation matrix
def find_correlation():
# Compute correlations between features
correlation_matrix = np.zeros((n_features,n_features))
for i in range(n_features):
for j in range(i+1, n_features):
# Correlation calculation formula
corr = calculate_correlation(i,j)
correlation_matrix[i][j] = corr
```

```
correlation_matrix[j][i] = corr
return correlation_matrix

# Select final features
def select_features(corr_matrix, k=3):
# Calculate combined score of correlation and variance
scores = [sum(abs(row))/variance for row,variance in zip(corr_matrix,variances)]
return np.argsort(scores)[:k]
```

۳.۲ نکات کلیدی

- ماتریس همبستگی به صورت دستی و بدون استفاده از کتابخانه‌های آماده محاسبه می‌شود
- معیار انتخاب ویژگی‌ها ترکیبی از میزان همبستگی و واریانس است
- تعداد ویژگی‌های انتخابی قابل تنظیم است (پیش‌فرض=۳)

۴.۲ خروجی‌ها

- ماتریس همبستگی در فایل correlation_matrix.txt
- اندیس ویژگی‌های انتخابی در فایل selected_features.txt

۳ فاز سوم: خوشه‌بندی

۱.۳ اهداف اصلی

- پیاده‌سازی ۴ الگوریتم خوشه‌بندی مختلف
- تنظیم پارامترهای هر الگوریتم
- مقایسه عملکرد الگوریتم‌ها و انتخاب بهترین روش
- تحلیل ویژگی‌های متمایزکننده هر خوشه

۲.۳ بخش اصلی کد

```
# Initialize clustering algorithms
algorithms = {
 'KMeans': KMeans(n_clusters=k, random_state=42),
```

```
 'Agglomerative': AgglomerativeClustering(n_clusters=k),
 'DBSCAN': DBSCAN(eps=0.3, min_samples=5),
 'MeanShift': MeanShift(bandwidth=0.5)
}

# Evaluate and compare algorithms
results = []
for name, algorithm in algorithms.items():
labels = algorithm.fit_predict(X_scaled)
score = silhouette_score(X_scaled, labels)
results.append({
 'name': name,
 'labels': labels,
 'score': score,
 'n_clusters': len(np.unique(labels))
})

# Visualize cluster characteristics
cluster_means = df.groupby('cluster').mean()
sns.heatmap(cluster_means, annot=True, cmap="YlGnBu")
plt.savefig("cluster_heatmap.png")
```

۳.۳ نکات کلیدی

- از معیار Silhouette Score برای ارزیابی کیفیت خوشه‌بندی استفاده شده است
- داده‌ها قبل از خوشه‌بندی نرمال شده‌اند
- نتایج به صورت مصورسازی‌های مختلف ذخیره می‌شوند
- برای هر الگوریتم پارامترهای بهینه انتخاب شده‌اند

۴.۳ خروجی‌ها

- فایل clustered_results.csv: داده‌های خوشه‌بندی شده
- تصاویر all_clustering_results.png و best_clustering_result.png: نتایج خوشه‌بندی
- تصویر cluster_heatmap.png: ویژگی‌های متمایزکننده هر خوشه

۴ فاز چهارم: مصورسازی نتایج

۱.۴ اهداف اصلی

- کاهش ابعاد داده‌ها برای نمایش بهتر
- نمایش بصری نتایج خوشه‌بندی
- مقایسه روش‌های مختلف کاهش ابعاد

۲.۴ بخش اصلی کد

```
# PCA 2D Visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.scatter(X_pca[:,0], X_pca[:,1], c=labels, cmap='viridis')
plt.title('PCA 2D Projection')
plt.savefig('pca_2d.png')

# PCA 3D Visualization
pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca_3d[:,0], X_pca_3d[:,1], X_pca_3d[:,2], c=labels)
plt.savefig('pca_3d.png')

# t-SNE Visualization
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)
plt.scatter(X_tsne[:,0], X_tsne[:,1], c=labels, cmap='coolwarm')
plt.title('t-SNE Projection')
plt.savefig('tsne_2d.png')
```

۳.۴ نکات کلیدی

- از دو روش کاهش ابعاد PCA و t-SNE استفاده شده است
- نتایج هم در ۲ بعد و هم در ۳ بعد نمایش داده شده‌اند

- رنگ‌های مختلف نشان‌دهنده خوشه‌های مختلف هستند
- تصاویر با کیفیت بالا ذخیره می‌شوند

۴.۴ خروجی‌ها

- pca_2d.png: نمایش دو بعدی با PCA
- pca_3d.png: نمایش سه بعدی با PCA
- tsne_2d.png: نمایش دو بعدی با t-SNE

۵.۴ تحلیل نتایج

- PCA برای نمایش کلی ساختار داده مناسب است
- t-SNE برای نمایش روابط غیرخطی و حفظ فاصله‌های محلی بهتر عمل می‌کند
- نمایش سه بعدی می‌تواند بینش بهتری از توزیع داده‌ها ارائه دهد

۵ فاز پنجم: ارزیابی خوشه‌بندی

۱.۵ معیارهای ارزیابی

- **Score Silhouette**: ارزیابی کیفیت خوشه‌بندی بر اساس فاصله‌های درون‌خوشه‌ای و بین‌خوشه‌ای
- **Precision**: خلوص هر خوشه (نسبت نمونه‌های متعلق به کلاس غالب)
- **Recall**: میزان پوشش هر کلاس در خوشه‌ها
- **F۱-Score**: میانگین هارمونیک precision و recall

۲.۵ بخش اصلی کد

```
def calculate_silhouette(X, labels):
# Calculate intra-cluster and nearest-cluster distances
a_i = np.mean([distance within cluster])
b_i = np.min([distance to other clusters])
return (b_i - a_i) / max(a_i, b_i)

def calculate_precision_recall_f1(true_labels, pred_labels):
# Build confusion matrix
```

```
confusion_matrix = pd.crosstab(true_labels, pred_labels)

# Precision = max class in cluster / cluster size
precision = {cluster: matrix[cluster].max()/sum(matrix[cluster])
 for cluster in matrix.columns}

# Recall = max cluster for class / class size
recall = {class: matrix.loc[class].max()/sum(matrix.loc[class])
 for class in matrix.index}

# F1 = 2 * (precision * recall) / (precision + recall)
return {
 'precision': np.mean(precision.values()),
 'recall': np.mean(recall.values()),
 'f1_score': np.mean(f1_scores)
}
```

۳.۵ نکات کلیدی پیاده‌سازی

- محاسبه Silhouette Score به صورت دستی با فرمول اصلی
- محاسبه معیارهای precision، recall و F۱ بر اساس ماتریس درهم‌ریختگی
- ذخیره نتایج ارزیابی در فایل خروجی
- نمایش ماتریس درهم‌ریختگی برای تحلیل دقیق‌تر

۴.۵ خروجی‌ها

- evaluation_results.csv: شامل مقادیر عددی معیارهای ارزیابی

۵.۵ تحلیل نتایج

- مقدار Silhouette Score بین ۱- تا ۱ متغیر است که مقادیر بالاتر نشان‌دهنده کیفیت بهتر خوشه‌بندی است
- مقادیر precision و recall بالای ۸.۰ نشان‌دهنده تطابق خوب خوشه‌ها با کلاس‌های واقعی است
- F۱-Score معیار متوازنی از دقت و بازیابی ارائه می‌دهد

۶ فاز ششم: پیش‌بینی خوشه‌ها

۱.۶ اهداف اصلی

- پیش‌پردازش و نرمال‌سازی داده‌های تست
- پیش‌بینی خوشه‌های داده‌های تست با استفاده از مدل KMeans آموزش‌دیده
- مصورسازی نتایج برای ۱۰ نمونه تصادفی
- ذخیره نتایج در قالب فایل CSV

۲.۶ مراحل اجرا

۱.۲.۶ بارگذاری مدل و داده‌ها

```
# Load selected features and clustered results
selected_features = [int(line.strip()) for line in open("selected_features.txt")]
clustered_df = pd.read_csv("clustered_results.csv")

# Load and preprocess training data for scaler
train_df = pd.read_csv("features.csv")
X_train = train_df.iloc[:, selected_features].values
scaler = StandardScaler().fit(X_train)
```

۲.۲.۶ پیش‌بینی خوشه‌ها

```
# Process test images and extract features
features_dict = extract_features(image)
X_test_selected = X_test[:, selected_features]
X_test_scaled = scaler.transform(X_test_selected)

# Predict clusters
test_clusters = model.predict(X_test_scaled)
```

۳.۲.۶ ذخیره نتایج

```
results_df = pd.DataFrame({
 'image_path': test_image_paths,
```

```
 'true_class': test_classes,
 'cluster': test_clusters
})
results_df.to_csv("test_predictions.csv", index=False)
```

۳.۶ مصورسازی نتایج

- برای هر یک از ۱۰ نمونه تصادفی:
  - نمایش تصویر تست
  - نمایش ۵ نمونه از خوشه مربوطه
  - عنوان‌بندی شامل کلاس واقعی و شماره خوشه
- ذخیره خروجی در فایل test_samples_with_cluster_members.png

۴.۶ خروجی‌ها

- test_predictions.csv: شامل مسیر تصاویر، کلاس واقعی و خوشه پیش‌بینی شده
- test_samples_with_cluster_members.png: مصورسازی نتایج

۵.۶ نکات پیاده‌سازی

- استفاده از همان اسکیلر آموزش‌دیده در فازهای قبل
- حفظ ترتیب ویژگی‌ها مطابق با داده‌های آموزش
- نمایش نمونه‌های تصادفی برای بررسی کیفیت خوشه‌بندی
- قابلیت کار با انواع مدل‌های خوشه‌بندی