مینا جمشیدی
۴۰۱۱۲۶۲۱۳۴
مبینا محمدی
۴۰۱۱۲۶۲۰۹۸

مستندات پروژه clustering

۱ فاز اول: استخراج ویژگی‌ها

اهداف اصلی

هدف از این فاز، پردازش تصاویر طبیعت و استخراج ویژگی‌های کلیدی برای خوشه‌بندی بود. ویژگی‌های استخراج شده باید بتوانند تفاوت بین انواع مختلف تصاویر طبیعی را به خوبی نشان دهند.

چالش‌ها و راهکارها

یکسان‌سازی اندازه تصاویر:
تصاویر ورودی با اندازه‌های مختلف وارد می‌شدند. برای حل این مشکل، همه تصاویر به اندازه استاندارد ۱۲۸×۱۲۸ پیکسل تغییر اندازه داده شدند. این اندازه به دلیلی انتخاب شد که هم جزئیات کافی حفظ شود و هم محاسبات بهینه باشد.

انتخاب ویژگی‌های معنادار:
ویژگی‌های انتخاب شده باید تفاوت بین کلاس‌ها را به خوبی نشان می‌دادند. برای این منظور از ترکیب چند نوع ویژگی استفاده شد.

محاسبه کارآمد ویژگی‌ها:
برای پردازش ۳۶۰۰ تصویر، از توابع بهینه‌شده کتابخانه‌های OpenCV و scikit-image استفاده شد تا محاسبات سریعتر انجام شود.

ویژگی‌های استخراج شده

ویژگی‌های رنگی:
- میانگین کانال‌های R، G، B
- انحراف معیار کانال‌های رنگی
این ویژگی‌ها تفاوت رنگ بین مناطق مختلف مثل دریا و جنگل را نشان می‌دهند.

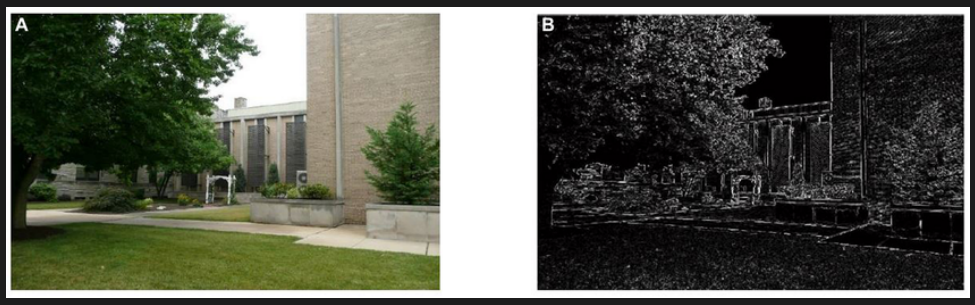
ویژگی‌های آماری:

- میانگین سطح خاکستری
- واریانس سطح خاکستری
این موارد روشنایی و کنتراست کلی تصویر را اندازه می‌گیرند.

ویژگی‌های لبه‌ای:
- تراکم لبه‌ها با فیلتر سوبل
برای تشخیص مرزهای واضح مثل خط ساحلی مفید است.

ویژگی‌های بافتی:
- کنتراست GLCM
- همگنی GLCM
این ویژگی‌ها تفاوت بین مناطق یکنواخت و پرجزئیات را نشان می‌دهند.

مثال



شکل ۱: (A) تصویر نمونه، (B) تراکم لبه ها

نتایج و خروجی‌ها

تمامی ویژگی‌های استخراج شده در فایل features.csv ذخیره شدند.
- ۱۱ ویژگی عددی برای هر تصویر
- برچسب کلاس مربوطه
- امکان استفاده در مراحل بعدی

جمع‌بندی

ویژگی‌های انتخاب شده:
- تفاوت بین کلاس‌های مختلف طبیعت را نشان دهند
- نسبت به تغییرات جزئی در تصاویر حساس نباشند

- خوب برای مراحل بعدی خوشه‌بندی
این ویژگی‌ها امکان تفکیک صحیح تصاویر طبیعی را فراهم می‌کنند.

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

اهداف اصلی

در این فاز، عملکرد الگوریتم‌های خوشه‌بندی با معیارهای مختلف ارزیابی شد. هدف اصلی سنجش کیفیت خوشه‌بندی انجام شده و تحلیل نتایج بود.

نتایج اجرای برنامه

خروجی کنسول پس از اجرای کد:

```
Clustering Evaluation Results:
Average Precision: 0.5878
Average Recall: 0.6033
Average F1-Score: 0.5964
Silhouette Score: 0.1472

Confusion Matrix:
Predicted 0 1 2 3 4 5
Actual
beach 21 18 82 0 291 188
dense_residential 0 318 124 132 24 2
desert 509 1 8 1 80 1
forest 0 56 542 0 2 0
intersection 0 283 47 67 168 35
sea_ice 5 47 222 1 229 96

Results saved to evaluation_results.csv
```

معیارهای ارزیابی

Silhouette:
این معیار با مقدار ۱۴۷۲.۰ نشان می‌دهد که ساختار خوشه‌بندی تا حدی مناسب است، ولی فضای بهینه‌ای بین خوشه‌ها وجود ندارد. این مقدار نشان می‌دهد که برخی نمونه‌ها نزدیک به مرز خوشه‌ها

قرار گرفته‌اند.
Precision:
با مقدار متوسط ۵۸۷۸.۰ نشان می‌دهد که به طور متوسط حدود ۵۸% از نمونه‌های هر خوشه متعلق به کلاس غالب هستند. این نشان می‌دهد خوشه‌ها تا حدی خالص هستند.
Recall:
مقدار ۶۰۳۳.۰ نشان می‌دهد که حدود ۶۰% از نمونه‌های هر کلاس در خوشه مربوطه قرار گرفته‌اند.
F1-Score:
با مقدار ۵۹۶۴.۰ نشان می‌دهد که توازن نسبتاً خوبی بین دقت و فراخوانی وجود دارد.

تحلیل ماتریس درهم‌ریختگی

- کلاس desert بهترین عملکرد را با ۵۰۹ نمونه در خوشه صحیح دارد
- کلاس beach و ice sea بیشترین اختلاط را نشان می‌دهند
- برخی خوشه‌ها (مانند خوشه ۳) نمونه‌های کمی دارند

تفاوت معیارها

تفاوت بین مقادیر معیارها به دلایل زیر است:
- Silhouette بر اساس فاصله‌های هندسی است
- Precision/Recall بر اساس تطابق با برچسب‌های واقعی است
- برخی مناطق طبیعی (مثل ساحل و یخ دریا) ویژگی‌های مشابهی دارند

جمع‌بندی

نتایج الگوریتم خوشه‌بندی:
- برای برخی کلاس‌ها (مانند بیابان) عملکرد خوبی دارد
- برای کلاس‌های با ویژگی‌های مشابه نیاز به بهبود دارد
- به طور کلی ساختار معقولی ایجاد کرده اما جای پیشرفت وجود دارد

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