دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

گزارش ششم

الگوریتم ژنتیک

علیرضا داودی _ 4013011

دکتر مهدی قطعی _ دکتر بهنام یوسفی مهر

بهار هزار و چهارصد و سه

مقدمه

الگوریتم‌های ژنتیک یکی از روش‌های بهینه‌سازی تکاملی هستند که بر اساس اصول انتخاب طبیعی و ژنتیک عمل می‌کنند. این الگوریتم‌ها برای حل مسائل پیچیده بهینه‌سازی، جستجو، و یادگیری ماشین مورد استفاده قرار می‌گیرند. هدف از این پروژه، پیاده‌سازی الگوریتم ژنتیک برای دسته‌بندی داده‌های Dry Bean Datasetاست.

فهرست مطالب صفحه

چکیده

این پروژه برای خوشه‌بندی داده‌های مجموعه "Dry Bean Dataset" از الگوریتم ژنتیک (یک الگوریتم تکاملی) و الگوریتم K-means (یک الگوریتم مرکز خوشه‌ای) استفاده می‌کند. در این پروژه، ابتدا داده‌ها از فایل Excel خوانده شده و پس از پیش‌پردازش‌های لازم مانند حذف داده‌های ناموجود و تبدیل برچسب‌ها به اعداد، داده‌ها مقیاس‌بندی و آماده شده‌اند.

سپس الگوریتم ژنتیک پیاده‌سازی شده است که شامل مراحل ایجاد جمعیت اولیه، ارزیابی تناسب، انتخاب، کراس‌اوور و جهش است. الگوریتم ژنتیک برای ایجاد خوشه‌بندی داده‌ها استفاده می‌شود.

علاوه بر این، خوشه‌بندی با استفاده از الگوریتم K-means نیز انجام شده است که به طور معمول از روش‌های متداول در خوشه‌بندی است.

نهایتاً، نتایج به دست آمده از هر دو الگوریتم مورد بررسی قرار می‌گیرند و بهترین خوشه‌بندی بر اساس هر الگوریتم مشخص می‌شود. این پروژه نشان می‌دهد که الگوریتم‌های تکاملی می‌توانند جایگزین مناسبی برای الگوریتم‌های مرسوم مانند K-means در مسائل خوشه‌بندی باشند، به ویژه زمانی که داده‌ها ساختارهای پیچیده‌تری دارند.

توضیح کامل الگوریتم ژنتیک

الگوریتم ژنتیک (Genetic Algorithm) یک روش جستجو و بهینه‌سازی مبتنی بر اصول انتخاب طبیعی و ژنتیک است. این الگوریتم به طور گسترده‌ای برای حل مسائل پیچیده بهینه‌سازی مورد استفاده قرار می‌گیرد. در اینجا، ما به توضیح مفصل هر مرحله از الگوریتم ژنتیک و نحوه پیاده‌سازی آن در کد می‌پردازیم.

مراحل الگوریتم ژنتیک

1.ایجاد جمعیت اولیه(Initialization)

- اولین مرحله در الگوریتم ژنتیک، ایجاد یک جمعیت اولیه از کروموزوم‌ها (راه‌حل‌های ممکن) به صورت تصادفی است.

- در کد ما، جمعیت اولیه با استفاده از `np.random.rand` ایجاد شده است که مقادیر تصادفی بین 0 و 1 تولید می‌کند.

```
pop_Size = 100
population = np.random.rand(pop_Size, X_scaled.shape[1])
```

2.ارزیابی تناسب(Fitness Evaluation)

- هر کروموزوم باید ارزیابی شود تا تناسب (fitness) آن تعیین شود. تناسب نشان‌دهنده کیفیت یک راه‌حل است.

- در کد ما، از `silhouette_score` برای ارزیابی کیفیت دسته‌بندی استفاده شده است. تابع `fitness_func`فاصله‌ها را محاسبه کرده و بر اساس آن دسته‌بندی می‌کند.

```python
from sklearn.metrics import silhouette_score

def fitness_func(chromosome, x):
    distances = np.linalg.norm(x[:, np.newaxis] - chromosome, axis=2)
    labels = np.argmin(distances, axis=1)
    if len(np.unique(labels)) > 1:
        fitness = silhouette_score(x, labels)
    else:
        fitness = 0
    return fitness
```

3.انتخاب(Selection)

- در این مرحله، کروموزوم‌هایی که دارای تناسب بیشتری هستند برای تولید نسل بعدی انتخاب می‌شوند.

- انتخاب به گونه‌ای انجام می‌شود که کروموزوم‌های بهتر شانس بیشتری برای انتخاب داشته باشند. در کد ما، کروموزوم‌ها بر اساس نمرات تناسب مرتب شده و نیمی از بهترین کروموزوم‌ها انتخاب می‌شوند.

```python
def selection(pop, fitness):
    selected_chromosomes = np.argsort(fitness)[-len(pop)//2:]
    return pop[selected_chromosomes]
```

4.کراس‌اور(Crossover)

- در مرحله کراس‌اور، دو کروموزوم انتخاب شده و بخش‌هایی از آن‌ها با یکدیگر ترکیب می‌شوند تا دو کروموزوم جدید (فرزند) ایجاد شود.

- نقطه کراس‌اور به صورت تصادفی انتخاب می‌شود و بخش‌های مختلف والدین با یکدیگر ترکیب می‌شوند.

```
def crossover(parent_1, parent_2):
    crossover_point = np.random.randint(1, len(parent_1))
    child1 = np.hstack((parent_1[:crossover_point], parent_2[crossover_point:]))
    child2 = np.hstack((parent_2[:crossover_point], parent_1[crossover_point:]))
    return child1, child2
```

5.جهش(Mutation)

- در این مرحله، برخی از ژن‌های کروموزوم‌های جدید به صورت تصادفی تغییر می‌کنند تا تنوع در جمعیت حفظ شود و از همگرایی زودرس جلوگیری شود.
- در کد ما، با استفاده از یک ماسک تصادفی جهش انجام می‌شود.

```
def mutation(child, mut):
    random_values = np.random.rand(len(child))
    mutation_mask = random_values < mut
    child[mutation_mask] = np.random.rand(np.sum(mutation_mask))
    return child
```

6.تکرار(Iteration)

- مراحل انتخاب، کراس‌اور و جهش به تعداد معینی نسل تکرار می‌شوند تا به یک راه‌حل بهینه برسیم.
- در هر نسل، جمعیت جدید جایگزین جمعیت قبلی می‌شود و الگوریتم به بهبود راه‌حل‌ها ادامه می‌دهد.

```
generations = 200
mutation_rate = 0.1

for generation in range(generations):
    fitness_values = np.array([fitness_func(chrom, X_scaled) for chrom in population])
    selected_pop = selection(population, fitness_values)
    next_pop = []

    while len(next_pop) < pop_size:
        parents = np.random.choice(range(len(selected_pop)), size=2, replace=False)
        parent1, parent2 = selected_pop[parents[0]], selected_pop[parents[1]]
        c1, c2 = crossover(parent1, parent2)
        c1 = mutation(c1, mutation_rate)
        c2 = mutation(c2, mutation_rate)
        next_pop.extend([c1, c2])

    population = np.array(next_pop)
```

7.خروجی نهایی(Final Output)

- پس از تکرار تعداد معینی از نسل‌ها، بهترین کروموزوم از جمعیت نهایی انتخاب و خروجی داده می‌شود.
- بهترین کروموزوم نمایانگر بهترین راه‌حل به دست آمده توسط الگوریتم ژنتیک است.

```
best_chromosome = population[np.argmax([fitness_func(chrom, X_scaled) for chrom in pop
for i in range(len(best_chromosome)):
    print(f"{i} : {best_chromosome[i]}")
```

الگوریتم K-means

- روش عملکرد: الگوریتم K-means یک الگوریتم خوشه‌بندی مبتنی بر مرکز است که داده‌ها را به K خوشه بر اساس میانگین فاصله بین نقاط داده و مراکز خوشه تقسیم می‌کند.
- مزایا:
- ساده و آسان در پیاده‌سازی است.
- برای مجموعه‌داده‌های بزرگ کارآمد است.
- معایب:
- حساس به موقعیت اولیه مراکز است.
- ممکن است به loca optima همگرا شود.

خوشه‌بندی با الگوریتم‌های تکاملی

- روش عملکرد: الگوریتم‌های تکاملی، مانند الگوریتم‌های ژنتیک، از اصول انتخاب طبیعی، کراس‌اوور و جهش برای تکامل جمعیتی از راه‌حل‌ها به سوی خوشه‌های بهینه استفاده می‌کنند.
- مزایا:
- قادر به کنترل ساختارهای پیچیده و غیرخطی داده‌ها هستند.
- کمتر حساس به شرایط اولیه هستند.
- معایب:
- هزینه محاسباتی بالا، به ویژه برای مجموعه‌داده‌های بزرگ.
- نیاز به تنظیم پارامترهای الگوریتم برای عملکرد بهینه دارد.

به طور خلاصه، در حالی که الگوریتم K-means ساده‌تر و کارآمد محاسباتی است، خوشه‌بندی با الگوریتم‌های تکاملی انعطاف پذیرتر و مقاومت بیشتری را ارائه می‌دهد، به خصوص برای مجموعه‌داده‌ها یا مسائلی که با روابط غیرخطی مواجه هستند. با این حال، این مزیت با هزینه پیچیدگی محاسباتی و نیاز به تنظیم پارامترها همراه است. انتخاب بین این دو به نیازها و ویژگی‌های خاص مجموعه‌داده و مسئله بستگی دارد.

نتیجه

الگوریتم ژنتیک با شبیه‌سازی فرآیندهای طبیعی انتخاب، ترکیب و جهش، به یک راه‌حل بهینه برای مسئله‌ی دسته‌بندی داده‌ها دست می‌یابد. این الگوریتم با تولید و بهبود مستمر جمعیت‌های کروموزومی، می‌تواند به نتایجی دست یابد که دیگر روش‌های بهینه‌سازی ممکن است نتوانند به سرعت یا به همان دقت برسند.

منابع

Abstract

The genetic algorithm, by simulating natural processes of selection, crossover, and mutation, converges to an optimal solution for the problem of data classification. By continuously generating and improving populations of chromosomes, this algorithm can achieve results that other optimization methods may not reach as quickly or as accurately.

Amirkabir University of Technoligy

(Tehran Polytechnic)

Facolty MCS

Fourth Project

Genetic algorithm

4013011 _ Alieza Davoudi

Dr.Yousefi Mehr _ Dr.Ghatee

Spring, 2024