دانشگاه صنعتي اميركبير
(پلی تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

گزارش۶: clustring دیتاست Iris

نگارش
احسان دهشیری پاریزی

استاد اول
دکتر مهدی قطعی

استاد دوم
دکتر بهنام یوسفی مهر

آبان ۱۴۰۲

چکیده

این گزارش به بررسی و پیاده‌سازی یک الگوریتم خوشه‌بندی مبتنی بر الگوریتم ژنتیک برای مجموعه داده Iris می‌پردازد. این الگوریتم از ایده‌های ژنتیک برای بهبود عملکرد خوشه‌بندی استفاده می‌کند. در این گزارش، مراحل گام به گام الگوریتم تشریح شده، کد پایتون ارائه شده و نتایج نهایی برای خوشه‌بندی به دست آمده است. همچنین، مقایسه با الگوریتم k-means انجام شده و نتایج و عملکرد الگوریتم ژنتیک مورد بحث قرار گرفته است. این گزارش به دنبال درک عمیق‌تر از عملکرد الگوریتم، ارزیابی کیفیت خوشه‌بندی و ارائه پیشنهادات برای بهبود عملکرد می‌باشد.

کلمات کلیدی:

الگوریتم ژنتیک - خوشه بندی - میوتیشن - کراس اور - خوشه بندی

مقدمه

در دهه‌های اخیر، با پیشرفت روزافزون در حوزه هوش مصنوعی، الگوریتم‌های تکاملی به‌عنوان ابزارهای مؤثری برای حل مسائل مختلف معرفی شده‌اند. یکی از حوزه‌هایی که از این الگوریتم‌ها بهره‌مند شده است، خوشه‌بندی داده‌ها می‌باشد. در این زمینه، الگوریتم ژنتیک به‌عنوان یک رویکرد ابتکاری در بهینه‌سازی و خوشه‌بندی اطلاعات معرفی شده است.

در این گزارش، ما تلاش کرده‌ایم تا الگوریتم ژنتیک را برای مسئله خوشه‌بندی داده‌های مشهور Iris به‌کار گیریم. این مسئله متناسب با مفهوم ژنتیک و ایده‌های تکاملی، در جستجوی یک تقسیم‌بندی بهینه از داده‌ها جهت تشکیل خوشه‌های یکسان می‌باشد.

در ادامه، مراحل این الگوریتم گام به گام تشریح شده است. این گزارش نیز شامل یک مقایسه با یک الگوریتم خوشه‌بندی معروف، یعنی k-means است، تا بتوانیم عملکرد الگوریتم ژنتیک را با یک روش سنتی مقایسه نماییم.

با مطالعه این گزارش، می‌توانید یک نگاه کامل به مراحل اجرای الگوریتم، ارزیابی عملکرد، و نتایج نهایی برای مسئله خوشه‌بندی Iris داشته باشید.

بخش اول

الگوریتم ژنتیک:

الگوریتم‌های ژنتیک (GA) یک دسته از الگوریتم‌های بهینه‌سازی هستند که الهام گرفته از فرآیند انتخاب طبیعی و اصول ژنتیک می‌باشند. معرفی شده توسط جان هالند در دهه ۱۹۶۰، الگوریتم‌های ژنتیک جزء حوزه

گسترده محاسبات تکاملی هستند. این الگوریتم‌ها از مکانیک انقلاب طبیعی برای بهبود گام به گام راه‌حل‌ها به مسائل بهینه‌سازی و جستجوی مسائل مختلف استفاده می‌کنند.

مفاهیم کلیدی:

1. جمعیت (population):

- یک جمعیت از یک مجموعه از راه‌حل‌های پتانسیل، به نام افراد یا کروموزوم‌ها تشکیل شده است. هر کروموزوم یک راه‌حل ممکن به مسئله است.

2. نمایش ژنتیک:

- راه‌حل‌ها به عنوان رشته‌های نمادها کدگذاری می‌شوند، معمولاً به عنوان ژن‌ها شناخته می‌شوند. ترکیب ژن‌ها در یک کروموزوم، یک راه‌حل ممکن به مسئله را نمایان می‌کند.

3. تابع تلفیق (Fitness Function):

- یک تابع تلفیق ارزیابی می‌کند که چقدر یک راه‌حل خاص مسئله را حل می‌کند. هدف، بیشینه کردن یا کمینه کردن مقدار تابع تلفیق، بسته به ماهیت مسئله بهینه‌سازی است.

4. انتخاب:

- افراد بر اساس توانایی ایشان انتخاب می‌شوند. راه‌حل‌هایی با ارزش تر برای تولید فرزندان انتخاب می‌شوند.

5. ترکیب (ترکیب مجدد) (Croosover):

- ترکیب مجدد شامل ترکیب اطلاعات ژنتیک از دو افراد والدین برای ایجاد فرزندان است. این به تقلید فرآیند بازترکیب ژنتیک در ارگانیسم‌های زنده مشابه است.

6. میوتیشن (Mutation):

- میوتیشن تغییرات تصادفی به افراد در جمعیت اضافه می‌کند. این تغییرات تنوع را افزایش می‌دهد و از گیر افتادن الگوریتم در اپتیمم محلی جلوگیری می‌کند.

7. شرایط پایان:

- الگوریتم زمانی که یک شرط پایان مشخص شده برقرار می‌شود متوقف می‌شود. این ممکن است تعدادی ثابت از نسل‌ها، رسیدن به یک آستانه خاص تابع تلفیق یا شرایط متوقف کننده دیگر باشد.

نحوه عملکرد:

1. مقدمات:

- یک جمعیت از راه‌حل‌های پتانسیل به صورت تصادفی ایجاد می‌شود.

2. ارزیابی:

- توانایی هر فرد با استفاده از تابع تلفیق ارزیابی می‌شود.

3. انتخاب:

- افراد بر اساس توانایی ایشان انتخاب می‌شوند. روش‌های انتخاب ممکن است شامل انتخاب چرخشی، انتخاب تورنومنت یا استراتژی‌های دیگر باشد.

4. ترکیب:

- جفت والدین جهت ترکیب انتخاب می‌شوند و فرزندان با اطلاعات ژنتیک از هر دو والدین ایجاد می‌شوند.

5. میوتیشن:

- تغییرات تصادفی به فرزندان اضافه می‌شود، که تنوع جمعیت را افزایش می‌دهد.

6. جایگزینی:

- جمعیت جدید ساخته می‌شود و جایگزین جمعیت قبلی می‌شود.

7. پایان:

- فرآیند ادامه می‌یابد تا تعدادی ثابت از نسل‌ها یا تا زمانی که شرایط پایان تعیین شده برقرار شوند.

کاربردها:

- الگوریتم‌های ژنتیک در حوزه‌های مختلفی از جمله مسائل بهینه‌سازی، یادگیری ماشین، زمان‌بندی، مالی و طراحی مهندسی استفاده می‌شوند.

- آن‌ها به ویژه در فضاهای جستجوی پیچیده که تکنیک‌های بهینه‌سازی سنتی ممکن است با مشکل روبرو شوند، موثر هستند.

بخش دوم

پیاده سازی:

- **وارد کردن کتابخانه‌ها:**

numpy به عنوان np برای عملیات عددی وارد شده است.

load_iris از sklearn.datasets برای بارگذاری مجموعه داده Iris استفاده می‌شود.

pairwise_distances_argmin_min از sklearn.metrics برای محاسبه فاصله زوجی استفاده می‌شود.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.metrics import pairwise_distances_argmin_min
```

- **بارگذاری مجموعه داده Iris:**

با استفاده از تابع load_iris مجموعه داده Iris بارگذاری می‌شود.

data حاوی مقادیر ویژگی‌های مجموعه داده است.

num_data_points و num_features تعداد نقاط داده و تعداد ویژگی‌ها را نگهداری می‌کنند.

```
iris = load_iris()
data = iris.data
num_data_points = data.shape[0]
num_features = data.shape[1]
```

- **پارامترهای الگوریتم ژنتیک:**

پارامترهایی مانند num_clusters، num_generations، population_size و mutation_rate برای کنترل الگوریتم ژنتیک تنظیم شده‌اند.

```
population_size = 50
num_generations = 100
num_clusters = 3
mutation_rate = 0.1
```

- **مقدماتی کردن جمعیت:**

تابع initialize_population یک جمعیت اولیه از کروموزوم‌ها با تخصیص تصادفی خوشه برای هر نقطه داده ایجاد می‌کند.

```
def initialize_population():
    return np.random.randint(0, num_clusters, size=(population_size, num_data_points))
```

- **محاسبه تابع تلفیق:**

تابع calculate_fitness توانایی هر کروموزوم در جمعیت را بر اساس مجموع فواصل بین نقاط داده و مراکز خوشه‌های اختصاص یافته به آنها ارزیابی می‌کند.

محاسبه فاصله:

ابتدا با استفاده از pairwise_distances_argmin_min، مراکز خوشه‌های تولید شده توسط کروموزوم‌ها با نقاط داده محاسبه می‌شود.

این تابع pairwise_distances_argmin_min برای هر نقطه داده، نزدیک‌ترین مرکز خوشه را محاسبه می‌کند و همچنین مقدار فاصله این نقطه تا مرکز محاسبه شده را ارائه می‌دهد.

محاسبه ارزش تابع تلفیق:

سپس با جمع کردن فواصل به دست آمده برای تمام نقاط داده، یک ارزش (عدد) به عنوان ارزش تابع تلفیق برای کل کروموزوم به دست می‌آید.

این ارزش تلفیق نشان‌دهنده کیفیت یا مطابقت کروموزوم با خوشه‌بندی مطلوب است. هر چه مقدار این ارزش کمتر باشد، کیفیت خوب‌تری نمایان می‌شود.

توجه داشته باشید که در این تابع، هدف الگوریتم ژنتیک کمینه کردن ارزش تابع تلفیق است؛ یعنی به دنبال یافتن کروموزومی است که خوشه‌بندی بهتری داشته باشد. این تابع تلفیق میزان مطابقت هر کروموزوم با الگوی خوشه‌بندی مطلوب را ارزیابی می‌کند.

```
def calculate_fitness(population):
    distances = pairwise_distances_argmin_min(data, data[population], axis=1)[1]
    fitness = np.sum(distances)
    return fitness
```

- ## ترکیب (ترکیب مجدد):

تابع crossover عملیات ترکیب (ترکیب مجدد) را با ترکیب اطلاعات ژنتیک از دو والدین برای ایجاد دو فرزند انجام می‌دهد.

انتخاب نقطه ترکیب (Crossover Point):

ابتدا یک نقطه تصادفی برای انجام عمل ترکیب انتخاب می‌شود. این نقطه تصادفی محل ترکیب کروموزوم‌ها را مشخص می‌کند. این نقطه تعیین کننده است که از کدام بخش از کروموزوم والد 1 و کدام بخش از کروموزوم والد 2 برای ایجاد فرزند استفاده شود.

ترکیب بخش‌های کروموزوم‌ها:

با استفاده از نقطه ترکیب، دو کروموزوم والد به دو نیمه تقسیم می‌شوند.

سپس نیمه اول کروموزوم والد 1 با نیمه دوم کروموزوم والد 2 ترکیب می‌شود تا فرزند اول ایجاد شود (child1).

نیمه اول کروموزوم والد 2 با نیمه دوم کروموزوم والد 1 ترکیب می‌شود تا فرزند دوم ایجاد شود (child2).

در نهایت، دو فرزند به عنوان نتیجه ترکیب برگردانده می‌شوند (child1 و child2).

این عملیات ترکیب به کروموزوم‌های فرزند ویژگی‌های هر دو والد را به اشتراک می‌گذارد و تنوع جمعیت را افزایش می‌دهد.

```
def crossover(parent1, parent2):
    crossover_point = np.random.randint(1, num_data_points)
    child1 = np.concatenate([parent1[:crossover_point], parent2[crossover_point:]])
    child2 = np.concatenate([parent2[:crossover_point], parent1[crossover_point:]])
    return child1, child2
```

- **جهش:**

تابع mutate تغییرات تصادفی (جهش) را به کروموزوم اعمال می‌کند که امکان کاوش تخصصی خوشه‌های مختلف را فراهم می‌کند.

جابه‌جایی لیبل (label_swap):

ابتدا، یک نقطه داده انتخاب می‌شود.

سپس با احتمال mutation_rate انجام می‌شود.

در صورت اجرای عمل جابه‌جایی لیبل (label_swap)، یک لیبل تصادفی از بین لیبل‌های ممکن (0 تا num_clusters - 1) به نقطه داده انتخاب شده اختصاص داده می‌شود.

جابه‌جایی با لیبل نزدیک‌ترین خوشه مرکزی (label_random):

اگر عمل جابه‌جایی با لیبل نزدیک‌ترین خوشه مرکزی (label_random) انجام شود:

ابتدا یک نقطه داده انتخاب می‌شود.

با احتمال mutation_rate انجام می‌شود.

لیبل نزدیک‌ترین خوشه مرکزی به نقطه داده انتخاب شده به نقطه داده اختصاص داده می‌شود.

به این ترتیب، با انجام عملیات جهش با احتمال مشخص، تنوع در جمعیت ایجاد می‌شود که می‌تواند به کاوش در فضای حالت‌های مختلف خوشه‌بندی منجر شود.

```
def mutate(chromosome):
    mutated_chromosome = chromosome.copy()
    for i in range(num_data_points):
        if np.random.rand() < mutation_rate:
            mutation_operation = np.random.choice(['label_swap', 'label_random'])
            if mutation_operation == 'label_swap':
                mutated_chromosome[i] = np.random.randint(num_clusters)
            elif mutation_operation == 'label_random':
                mutated_chromosome[i] = np.random.choice(np.setdiff1d(np.arange(num_clusters), mutated_chromosome[i]))
    return mutated_chromosome
```

- **الگوریتم ژنتیک:**

تابع genetic_algorithm الگوریتم ژنتیک را اجرا می‌کند. این تابع یک جمعیت را مقدماتی می‌کند، به صورت تکراری توانایی، والدین منتخب، ترکیب، جهش و به‌روزرسانی جمعیت را برای تعداد مشخصی از نسل‌ها انجام می‌دهد.

معرفی جمعیت:

ابتدا یک جمعیت اولیه از کروموزوم‌ها با استفاده از تابع initialize_population ایجاد می‌شود.

تکامل جمعیت:

سپس، برای تعداد دفعات مشخص شده در num_generations تکامل جمعیت انجام می‌شود.

محاسبه ارزش تابع تلفیق:

برای هر کروموزوم در جمعیت، ارزش تابع تلفیق با استفاده از calculate_fitness محاسبه می‌شود.

انتخاب نمونه‌های برتر:

نمونه‌هایی که ارزش تابع تلفیق کمتر دارند، به دلیل مینیمم کردن مسئله، انتخاب می‌شوند. این انتخاب با استفاده از argsort انجام می‌شود.

تولید نسل بعد:

برای تولید نسل بعد، از نمونه‌های برتر انتخاب شده (selected_population) برای تولید والدین استفاده می‌شود.

هر زوج والد با استفاده از تابع crossover ترکیب می‌شود و دو فرزند ایجاد می‌شوند.

برای افزودن تنوع، روی هر فرزند عملیات mutate اجرا می‌شود.

تحول جمعیت:

جمعیت جدید با افزودن فرزندان تولید شده به جمعیت اولیه تشکیل می‌شود.

بهترین کروموزوم:

در پایان تکامل، بهترین کروموزوم بر اساس مقدار کمینه‌ی تابع تلفیق انتخاب می‌شود و به عنوان جواب نهایی تابع برگردانده می‌شود.

این تابع به ازای تعداد نسل‌های مشخص شده (num_generations) الگوریتم ژنتیک را اجرا می‌کند و بهترین حالت برای مسئله خوشه‌بندی را برمی‌گرداند.

```
def genetic_algorithm():
    population = initialize_population()

    for generation in range(num_generations):
        fitness_values = np.array([calculate_fitness(chromosome) for chromosome in population])
        selected_indices = np.argsort(fitness_values)[:population_size // 2]
        selected_population = population[selected_indices]

        new_population = []

        for _ in range(population_size // 2):
            parent1, parent2 = selected_population[np.random.choice(selected_population.shape[0], 2, replace=False)]
            child1, child2 = crossover(parent1, parent2)
            new_population.extend([mutate(child1), mutate(child2)])

        population = np.array(new_population)

    best_chromosome = population[np.argmin([calculate_fitness(chromosome) for chromosome in population])]
    return best_chromosome
```

- **اجرای الگوریتم ژنتیک:**

الگوریتم ژنتیک اجرا می‌شود و بهترین راه‌حل خوشه‌بندی در نهایت چاپ می‌شود.

```
best_solution = genetic_algorithm()

print("Final Clustering Result:")
print(best_solution)
```

بخش سوم

جمع بندی و نتیجه گیری:

در این پروژه، یک روش خوشه‌بندی مبتنی بر الگوریتم ژنتیک برای دیتاست Iris پیاده‌سازی شد. این الگوریتم با استفاده از مبانی الگوریتم ژنتیک، اعمال جهش، ترکیب و انتخاب کروموزوم‌ها، به بهینه‌سازی یک تابع تلفیق برای مسئله خوشه‌بندی پرداخت.

نتایج نهایی حاکی از این است که الگوریتم ژنتیک توانسته است یک خوشه‌بندی بهینه انجام دهد که با توجه به تابع تلفیق، فواصل نقاط داده درون هر خوشه به حداقل رسیده است.

مزایای این روش شامل امکان کاوش در فضای گستردهٔ حل‌ها، امکان مدیریت تنوع جمعیت و ارائهٔ جواب بهینه بر اساس مسئله‌ی خوشه‌بندی می‌باشد.

الگوریتم ژنتیک، با توجه به ماهیت مسئله خوشه‌بندی، عملکرد خوبی از خود نشان داده است. با این حال، انتخاب پارامترهای مهم مانند اندازه جمعیت، تعداد نسل‌ها و احتمالات جهش و ترکیب نیازمند آزمایش و بهینه‌سازی است. همچنین، مقایسه با الگوریتم‌های دیگر خوشه‌بندی نیز می‌تواند به تحلیل عملکرد الگوریتم ژنتیک در مسائل خوشه‌بندی کمک کند.

منابع

https://github.com/EhsunD/AI_Report6_ClusteringWhitGeneticAlgorithm