دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده ی ریاضی و علوم کامپیوتر

پایان‌نامه کارشناسی‌ارشد

گرایش علوم کامپیوتر

سرچ محلی

گزارش سوم

نگارش

پویاپارسا

استاد راهنما

دکتر قطعی

فروردین ۱۴۰۰


بسم الله الرحمن الرحيم

فصل اول

تعریف مسئله

۱-۱ مقدمه

پیرو گزارش های قبلی و اهمیت سیستم های پیشنهاد دهنده که در گزارش ۲ نیز به آن ها اشاره شد، در این گزارش به توصیف سرچ محلی ای می پردازم که در آن عامل هوشمند سعی بر خوشه بندی کاربران دارد.

۱-۲ صورت مسئله

برای پیشنهاد به یک کاربر ، ابتدا باید آن کاربر را شناخت ، خصوصیات رفتاری وی را دانست ، به علایق فرد آگاه بود اما چگونه می توان کاربران را به خوشه هایی تقسیم بندی کرد که مشابه به هم هستند؟ در پاسخ به این سوال باید گفت که روش های متفاوتی و متنوعی وجود دارد که عموما بر دو دسته ی کلی تعامل محور و اطلاعات محور تقسیم بندی می شوند ([۱])

سعی در خوشه بندی کاربران بر حسب اطلاعات ایشان که تا کنون در دیتابیس جمع آوری شده دارم این اطلاعات می تواند سن - جنسیت - تعداد دقایق گذرانده در سرویس و ...

توجه : لازم به ذکر است که برای دیداری سازی در جای جای این گزارش از تصاویر در دو بعد استفاده شده است این در صورتی است که اطلاعات کاربر یک بردار n بعدی که گاها n تا هزاران بعد هم می رسد است.

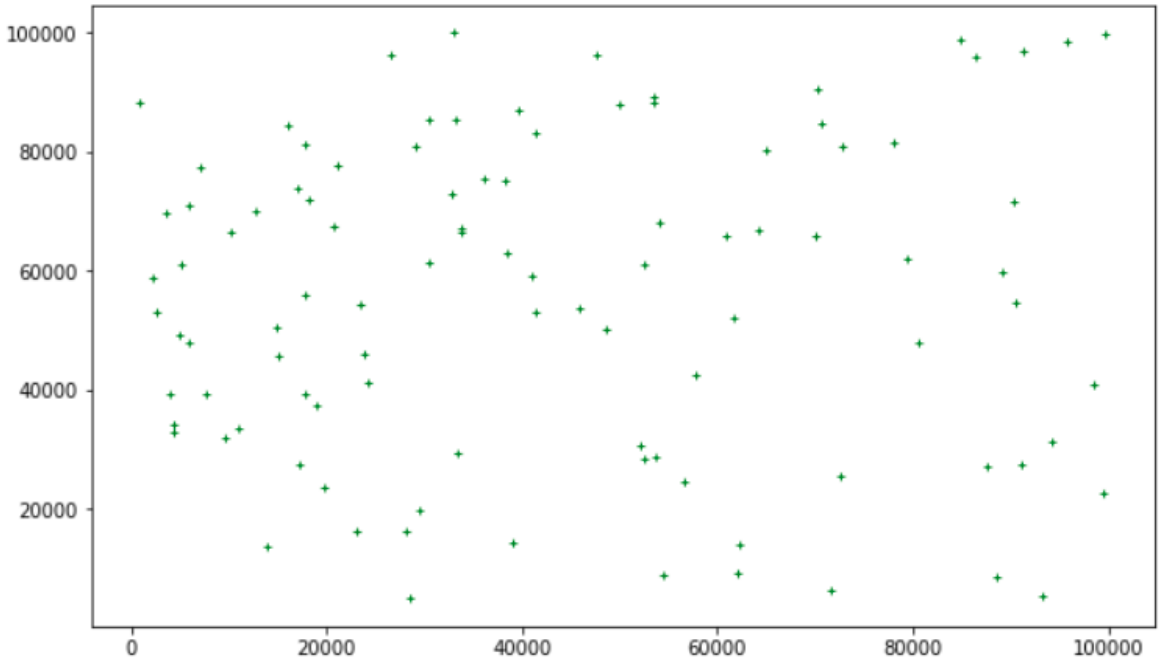
فصل دوم

مدل سازی

۲-۱ رویکرد

خوشه بندی را می توان در قالب دو سوال کلی خلاصه کرد
الف) کاربران باید چند دسته شوند ؟
ب) مراکز این دسته ها کجاست ؟

دقت کنید که هدف از این گزارش بهبود خوشه بندی و ارزیابی آن نیست بلکه ارائه ی عامل هوشمندی است که با استفاده از سرچ محلی بتواند این خوشه بندی را انجام دهد به همین دلیل فرض کردیم تعداد دسته ها دو می باشد و بر روی چگونه پیدا کردن مراکز تمرکز کرده ایم.



شکل ۲-۱: فضای کاربران با توجه به دو ویژگی آن ها

۲-۲ ساخت گراف

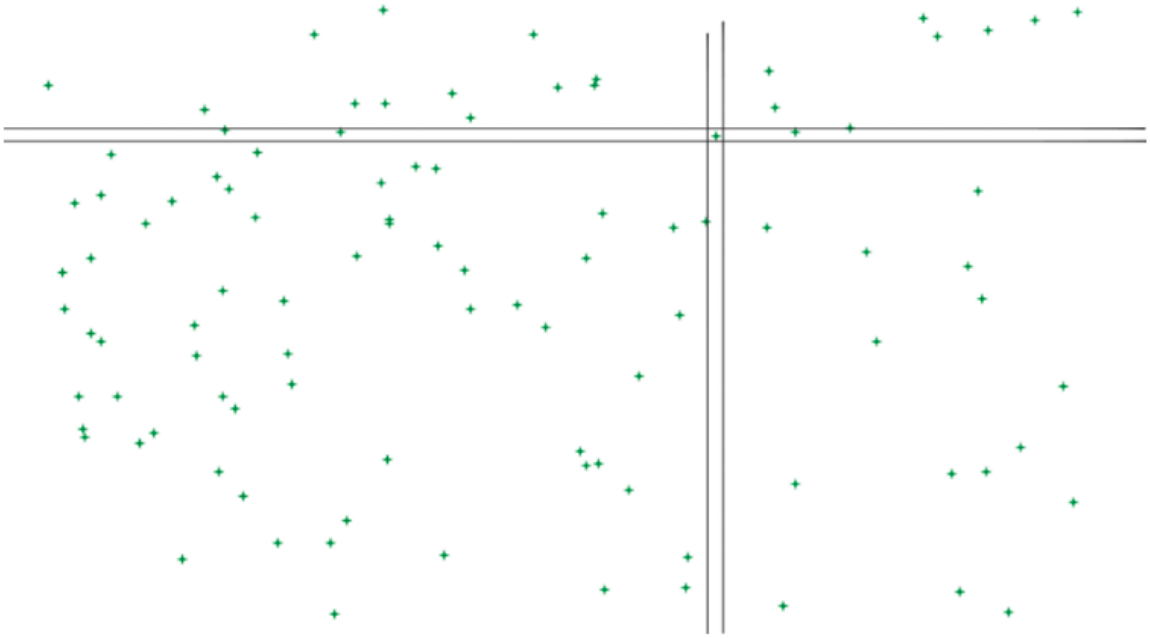
یکی از چالش های اصلی در ساخت گراف ، تعریف همسایگی است زیرا اگر درجه ی هر راس عدد بزرگی باشد یک مرحله پیمایش در گراف بسیار زمان بر و از لحاظ محاسباتی سنگین خواهد بود اما از طرفی زیاد بودن همسایه ها باعث بهبود بیشتر جواب در هر مرحله خواهد شد. در واقع با نوعی trade-off در اینجا رو به رو هستیم، در نتیجه اتصال هر کاربر به همه ی کاربران نتیجه ی جالبی به همراه نخواهد داشت لذا باید اتصالات را به نحوی هوشمندانه تعریف کنیم که هم درجه هر راس پایین و هم امکان بهبود در همسایه معقول باشد.

۱. محاسبه ی ۴ کاربر نزدیک به هر کاربر از $O(n^2)$ زمان می برد. (naive solution)

۲. با جا به جایی از یک راس به همسایه های آن بهبود چندانی حاصل نمی شود.

۲-۳ فاصله در تک بعد به جای فاصله اقلیدسی

اما با یک راه حل ساده می توان هر دو مشکل فوق را هم زمان حل کرد به جای نزدیک ترین کاربر، نزدیک ترین کاربر در هر بعد و از هر سمت را پیدا کنیم در واقع یک بار کاربر با نزدیک ترین f1 از سمت راست به کاربر مورد نظر و بار دیگر از سمت چپ و همین روند برای نزدیک ترین f2 برای دیداری سازی بهتر ، در واقع کاربرانی که به چهار خط کشیده شده نزدیک ترین هستند انتخاب می شوند.



شکل ۲-۲: چهار نقطه نزدیک در هر بعد از راست و چپ

در شکل می توان مشاهده کرد که علی رغم اشتراک معیار اقلیدسی و معیار تعریف شده ، مزایای زیر وجود دارند :

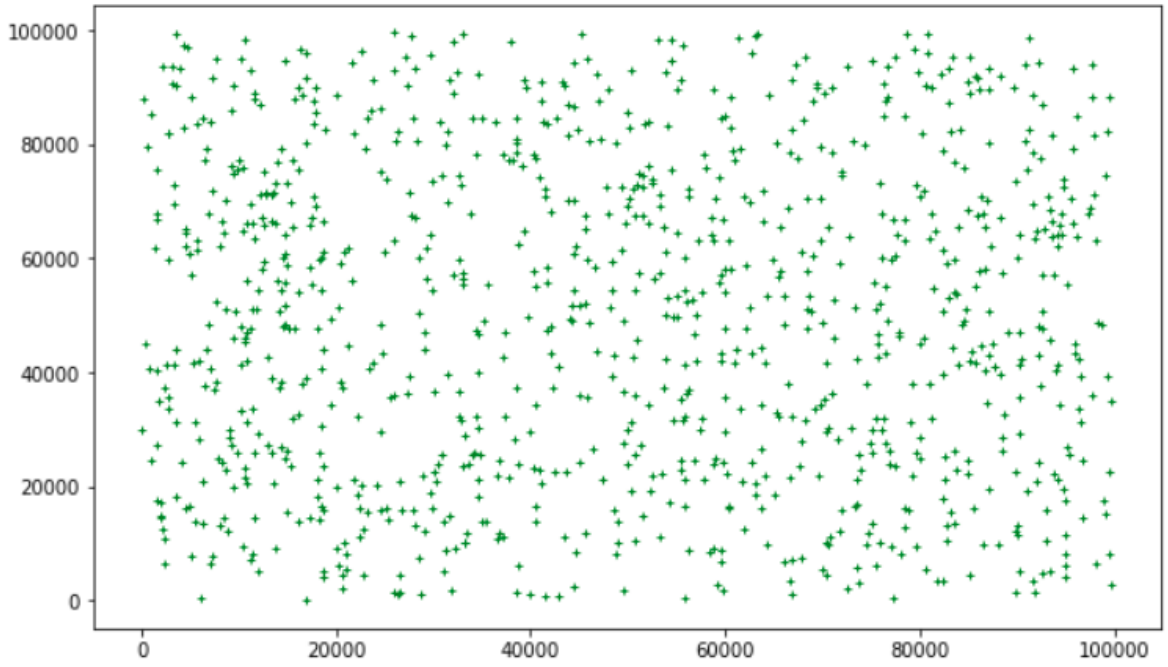
۱. محاسبه ی چهار کاربر نزدیک از $O(n \log n)$ امکان پذیر شده است.

۲. همسایه های تعریف شده پراکندگی بیشتری دارند و لذا پتانسیل بهبود بیشتری دارند لازم به ذکر است که اگر همسایه ای نتیجه ی بدتر به ما بدهد ما به آن همسایه ورود نخواهیم کرد.

۲-۴ یک خلاقیت ساده

یکی از مشکلاتی که در داده های جزیره ای متراکم وجود داشت (شکل ۲-۴) این بود که ۴ همسایه نزدیک با استفاده از روش دوم تقریبا همان ۴ همسایه نزدیک اقلیدسی بودند به همین خاطر به جای دو

همسایه نزدیک ، دو همسایه رندوم از گراف را اضافه کردم تا احتمال بهبود های چشم گیر افزایش پیدا کند همین خلاقیت ساده موجب همگرایی ۲۵ درصد سریع تر در تعدادی نمونه ی تصادفی بود. که در بخش تحلیل حساسیت به آن ها پرداخته شده است.



شکل ۲-۳: وجود داده های متراکم موجب کندی همگرایی می شود.

فصل سوم

پیاده سازی و جستجو

۳-۱ مقدمه

در جستجو ی محلی ، تابع H بر روی هر نود تخمین ما از رسیدن به هدف را مشخص می کند ، هدف در این الگوریتم پیدا کردن مراکزی است که به داده ها نزدیک تر هستند. در این بخش به تعریف و پیاده سازی H و همچنین نحوه ی جریان مراکز ها در داده ها می پردازم. کد استفاده شده در این گزارش از طریق این لینک در دسترس است .

۳-۲ پیمایش در گراف

همانند تمام الگوریتم های سرچ محلی ، با شروع از هر گره ، گره های مجاور رو رویت کرده و در صورت بهبود در تخمین مان به آن ها پیمایش می کنیم.

```python
# traverse gets a center and check its neighbours for better H
# returning index of center as result

def traverse(center, centers, points):

  centers_ = centers.copy()

  current_H = H(points, centers_)
  current_center = center
  centers_.remove(current_center)



  for node in G.neighbors(center):


    centers_.append(node)

    if H(points, centers_) < current_H:
      current_H = H(points, centers_)
      current_center = node

    centers_.remove(node)

  return current_center
```

شکل ۳-۱: تخمین مدل از بهبود در آینده.

۳-۳ نحوه ی محاسبه ی H

در تابع H مراکز به عنوان ورودی گرفته خواهند شد مجموع فاصله ی نقاط از این مراکز به دست خواهد آمد ، وقتی که تعداد مراکز ثابت باشد؛ هر چه این عدد کوچکتر باشد نشان دهنده ی بهتر بودن آن مراکز است.

```
1  def H(points, centers):
2      H_value = 0
3      for point in points:
4          H_value += nearest_center(point, centers)[1]
5
6      return H_value
```

شکل ۳-۲: تخمین مدل از بهبود در آینده.

۳-۴ نحوه ی چرخش مرکز ها و اعلام همگرایی :

در هر مرحله (iteration) هر مرکز به همسایه ی خود که مرکز بهتری است منتقل می شود هنگامی که هیچ یک از مرکز ها همسایه ی بهتری نداشته باشند الگوریتم متوقف شده و اعلام همگرایی می کند.

```
5  centers = [0, 1]
6  centers_ = [0, 0]
7
8  while True:
9
10     if centers_ == centers:
11         break
12
13     centers_ = centers
14
15     for i in range(len(centers)):
16
17
18         # looking for better centers in neighbours
19         node = traverse(centers[i], centers, points)
20
21         centers[i] = node
```

شکل ۳-۳: همگرایی در مدل.

فصل چهارم

تحلیل حساسیت

در این بخش به مقایسه زمان اجرای سه الگوریتم مختلف با هم پرداخته ایم. NS همان سرچ معمولی با اتصال به ۴ کاربر نزدیک در هر بعد و SS سرچ با استفاده از گرافی است که بعضی از یال های آن تصادفی هستند و KMeans هم یکی از الگوریتم های معروف خوشه بندی است. همچنین زمان اجرای هر الگوریتم در مقایسه با سایز ورودی آن می توانید مشاهده کنید.

تعداد کل کاربران	SS	NS	KMeans
۱۰۰	۰.۰۱۳	۰.۰۰۸	۰.۰۳۱
۱۰۰۰	۰.۰۷۶	۰.۱۲۱	۰.۰۵۸
۱۰۰۰۰	۰.۷۱۹	۰.۹۴۲	۰.۱۳۹

جدول ۴-۱: زمان اجرا با سایز ورودی متفاوت

هرچند که در داده های با تعداد زیاد الگوریتم KMeans عملکرد بسیار بهتری را به نمایش گذاشته است ولی در تعداد بسیار کم الگوریتم NS عملکرد بهتری داشته است.
تمایز بین افزودن یال های رندوم به گراف و حالت معمولی نیز قابل توجه است.

منابع و مراجع

[1] Kordík, Pavel. Machine learning for recommender systems — part 1 (algorithms, evaluation and cold start)). https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-sta