به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

تمرین سری چهارم داده کاوی – بخش پیاده سازی

توضیحات:

- پاسخ به تمرین‌ها باید به صورت انفرادی صورت گیرد و در صورت مشاهده هرگونه تقلب نمره صفر برای کل تمرین منظور خواهد شد.
- تمیزی و خوانایی گزارش تمرین از اهمیت بالایی برخوردار است.
- گزارش تمرین خود را در قالب یک فایل PDF با نام **«HW4_StudentNumber.pdf»** به همراه کد های بخش پیاده سازی (فایل های ipynb. یا py.) در فایلی به نام **«HW4_StudentNumber.zip»** قرار داده و در سایت درس در مهلت معین بارگذاری نمایید.
- توجه داشته باشید که به سوالات پیاده سازی بدون گزارش نمره ای تعلق نمی گیرد.
- در صورت داشتن اشکال می‌توانید از طریق ایمیل **datamining.fall2020@gmail.com** با تدریس‌یاران درس در ارتباط باشید.
- همچنین لازم بذکر است که اگر مواردی در کلاس تدریس نشده انتظار می رود که خود دانشجویان جستجو کنند و انجام دهند.
- پیاده سازی این تمرین نسبت به تمارین قبلی حجم بیشتری دارد و نمره بیشتری هم خواهد داشت و تحویل این تمرین به صورت مجازی خواهد بود که زمان آن هم اطلاع رسانی می شود.

نیم‌سال اول ۰۰-۹۹

سوال ۱- در این بخش قصد داریم الگوریتم خوشه بندی **K-Means** را برای مجموعه داده های **2D**پیاده سازی کنیم. سپس عملکرد آن را ارزیابی کرده و در انتها محدودیت **K-Means** در خوشه بندی را با اعمال آن روی یک مجموعه داده بررسی میکنیم.

همانطور که می دانید الگوریتم **K-Means** با انتخاب تعدادی نقطه اولیه به عنوان مراکز خوشه و منصوب کردن نقاط داده به نزدیک ترین مرکز خوشه آغاز می شود. سپس در هر دور تکرار این مراکز خوشه ها به روز شده و انتصاب داده ها دوبار انجام می شود. این مراحل تا همگرایی الگوریتم تکرار شده و در انتها نقاط داده به تعدادی خوشه تقسیم خواهند شد. شبه کد این الگوریتم در شکل ۱ آورده شده است.

```
Input:
    D= {t1, t2, …. Tn }  // Set of elements
    K                    // Number of desired clusters
Output:
    K                    // Set of clusters
K-Means algorithm:
   Assign initial values for m1, m2,…. mk
   repeat
     assign each item ti    to the clusters which has the closest mean;
     calculate new mean for each cluster;
   until convergence criteria is met;
```

شکل ۱- شبه کد الگوریتم **K-Means**

نکته- شرط اتمام الگوریتم را می توان به یکی از ۴ حالت زیر تعریف کرد:

۱- محل مراکز خوشه بین دور **n** و **n+1** تغییر نکند. در نتیجه الگوریتم در مرحله **n** همگرا شده است و در مرحله **n+1** متوقف میشود.

۲- یک آستانه (**threshold**) برای تغییر محل مراکز خوشه در نظر گرفته شود، و اگر تغییر محل مراکز از آن **threshold** کمتر باشد الگوریتم متوقف بشود.

۳- یک آستانه برای تعداد نقاطی که به خوشه جدیدی منصوب می شوند گذاشته بشود و اگر تعداد نقاط مذکور کمتر از آستانه مورد نظر باشد الگوریتم متوقف بشود.

۴- محدودیت روی تعداد دور های اجرای الگوریتم گذاشته شود.

الف) ابتدا مجموعه داده **dataset1** و **dataset2** را مصور سازی کنید.

الف) پس از پیاده سازی الگوریتم، می خواهیم آن را روی **dataset1** امتحان کنیم. تعداد دور ها را حداقل ۱۵ گذاشته و الگوریتم را با مقادیر **k=2, 3, 4** اجرا کنید. پس از هر اجرا نقاط داده خوشه بندی شده را با رنگ های مختلف برای هر خوشه نمایش دهید. (می توانید از **matplotlib** برای این مصور سازی استفاده کنید.)

ب) پس از اتمام خوشه بندی، برای هر خوشه، میانگین فاصله مرکز خوشه با نقاط موجود در آن خوشه را به دست آورید و گزارش دهید. این مقدار خطای خوشه خوانده می شود.

ج) میانگین خطاهای به دست آمده در قسمت قبل را محاسبه کنید و گزارش دهید. این مقدار خطای خوشه بندی خوانده می شود.

د) الگوریتم را برای **0<k<15** اجرا کرده و برای هر کدام از **k** ها خطای خوشه بندی را محاسبه کنید و این خطا ها را مصور سازی کنید.

ه) با استفاده از روش **elbow**، مقدار **k** بهینه را با استفاده از شکل به دست آمده در قسمت (د) گزارش دهید.

ی) الگوریتم پیاده سازی شده را این بار بر روی **dataset**۲ اجرا کرده و آن را مصور سازی کنید. دلیل مناسب نبودن این الگوریتم برای خوشه بندی این مجموعه داده را ذکر کنید.

سوال۲- کاهش رنگ با استفاده از الگوریتم **K-means**

کاهش حجم تصاویر و در عین حال حفظ کیفیت همیشه یکی از موضوعات مورد توجه در حوزه تصویر بوده است. از آنجایی که ما برای دخیره رنگ(RGB) یک پیکسل از ۲۴ بیت استفاده می کنیم(۸ بیت برای قرمز، ۸ بیت برای سبز و ۸ بیت برای آبی)، رنگ های موجود در یک تصویر می تواند شامل تنوع بسیار زیادی باشد(۲۵۶x۲۵۶x۲۵۶). می توان با استفاده از الگوریتم K-means رنگ های شبیه به هم در یک خوشه قرار داد و آن رنگ ها را با رنگ مرکز خوشه جایگزین کرد. مثلا فرض کنید ما تصویر را به ۳۲ رنگ کاهش می دهیم، در این صورت برای ذخیره رنگ هر پیکسل ۵ بیت کافی است(که نسبت به ۲۴ بیت بسیار کمتر است).

در این سوال قرار است شما تصویر sample_img1.png را کاهش رنگ دهید و به ازای مقادیر K (تعداد رنگ های نهایی تصویر) برابر با ۲، ۴، ۱۶، ۳۲ و ۶۴ خروجی خود را نمایش دهید.

عکس مورد نظر را میتوانید با استفاده از کتاب خانه image در matplotlib لود کنید. دقت کنید که با این روش عکس به صورت یک آرایه ۳ بعدی numpy در می آید که ۲ بعد اول مکان هر پیکسل را نمایش میدهد و بعد سوم مقادیر RGB را نگهداری میکند.

توجه- باید از الگوریتم K-Means ای که در بخش اول پیاده سازی کردید استفاده کنید. از آنجایی که الگوریتمی که پیاده سازی کردید برای فضای 2D میباشد، باید ماتریس عکس مورد نظر را به یک ماتریس ۲ بعدی تغییر شکل دهید.

به عکس های زیر بعنوان نمونه توجه کنید:

سوال ۳-

در این بخش هدف آشنایی بیشتر با الگوریتم DBSCAN و کاربرد آن در تحلیل داده میباشد. این الگوریتم برای مواقعی که خوشه ها به خوبی از هم جدا نیستند و داده ها شامل نویز می‌باشد بسیار کارآمد خواهد بود. به علت پیچیدگی پیاده سازی این الگوریتم، برای این بخش میتوانید از کتابخانه DBSCAN موجود در scikit-learn استفاده کنید. برای پیاده سازی این بخش حتما از jupyter notebook استفاده کنید تا در مصور سازی ها به مشکل نخورید.

در این بخش با مجموعه داده covid.csv که شامل توزیع جغرافیایی بیماران covid۱۹ در ایران میباشد کار خواهیم کرد. (داده ها مربوط به اوایل شیوع کرونا میباشد)

شکل زیر نقاط موجود در مجموعه داده را روی نقشه نشان میدهد.

هر سطر از این مجموعه داده نمایانگر یک بیمار میباشد، ستون اول نمایانگر عرض جغرافیایی و ستون دوم نمایانگر طول جغرافیایی محل آن بیمار است. برای مصورسازی بهتر این مجموعه داده جغرافیایی از کتابخانه folium در پایتون استفاده خواهیم کرد. میتوانید این کتابخانه را با دستور زیر نصب کنید.

pip install folium -

تکه کد زیر نمونه ای از نمایش یک موقعیت جغرافیایی با استفاده از folium میباشد:

```
import folium

# set iran as map starting point
m = folium.Map(location=[32.427910, 53.688046], zoom_start=5)

# mark an example location
loc = [35.703136,51.409126]
folium.Marker(location=loc).add_to(m)
```

همانطور که میدانید، DBSCAN دارای ۲ پارامتر اصلی minPoints و eps میباشد. eps حداکثر فاصله مجاز بین دو نقطه داده در یک خوشه و minPoints حداقل تعداد نقاط داده برای تشکیل یک خوشه میباشد. با تغییر این دو پارامتر میتوان تراکم نقاط داده موجود در خوشه ها و تعداد خوشه ها را کنترل کرد. هدف از این بخش پیدا کردن خوشه های چگال (دارای تراکم بالای بیمار) در ایران میباشد.

الف) ابتدا مجموعه داده را لود کرده و با استفاده از folium آن را روی نقشه نشان دهید. از تکه کد پایین برای این کار کمک بگیرید.

```
folium.Circle(location=loc,
              radius=1,
              color="red",
              fill=True).add_to(m)
```

این تکه کد یک نقطه را روی نقشه اضافه میکند. با اجرای یک حلقه روی مجموعه نقاط میتوانید همه ی نقاط را روی نقشه نمایش دهید.

ب) الگوریتم DBSCAN را برای خوشه بندی نقاط روی نقشه استفاده کنید. (ابتدا یک بار با مقادیر دلخواه الگوریتم را اجرا کرده و نتیجه را گزارش کنید.)

ج) در این بخش میخواهیم مکان های پر تراکم را شناسایی کنیم. با تغییر مقادیر eps و minPoints، مقادیر مناسب برای یافتن مکان های پرتراکم را پیدا کنید. در واقع تعداد خوشه ها باید نمایانگر مکان های پر تراکم باشد. (حداقل ۳ و حداکثر ۶ شهر پرتراکم باید شناسایی شود)

راهنمایی: تهران و قم جزو شهر های پر تراکم میباشند.

د) هر خوشه (شهر پر تراکم) را با یک رنگ خاص نمایش دهید. داده های پرت (outlier) ها را (نقاطی که جزو خوشه های ما و در نتیجه جزو مکان های پرتراکم نیستند) با رنگ سیاه نمایش دهید.

توجه- مجموعه دادی اصلی با نام covid.csv شامل تعداد حدود ۱۸۰۰۰ نمونه میباشد. اگر حجم آن برای شما سنگین است و در مصورسازی به سیستم شما فشار می‌آید، از مجموعه داده covid-sample.csv که یک نمونه ۲۰۰۰ تایی از فایل اولیه است استفاده کنید.

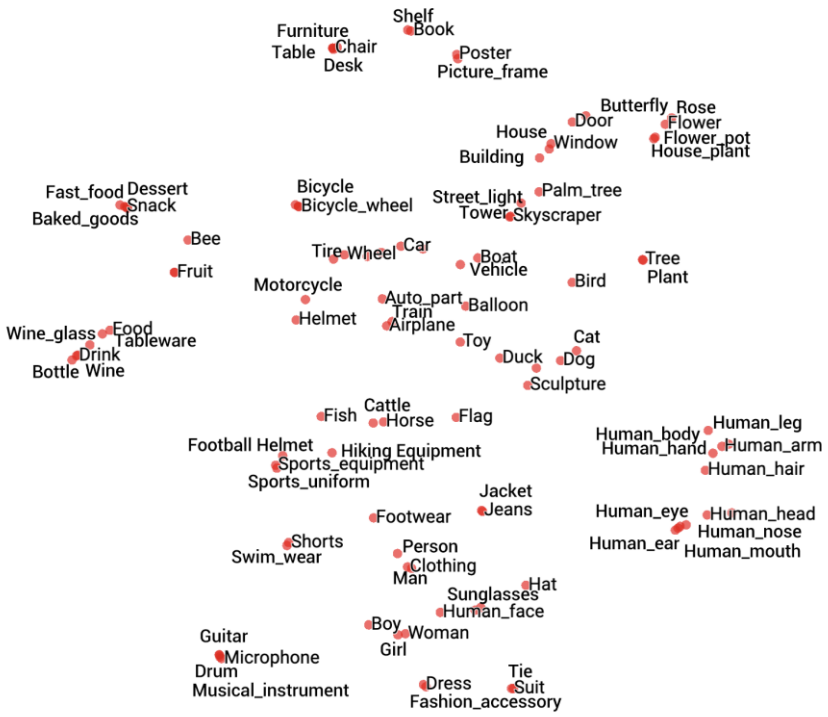
سوال ۴- بردارهای بازنمایی[1] در بحث پردازش زبان طبیعی یکی از پرکاربردترین روش‌ها برای نگاشت معانی و مفاهیم کلمات به یک نمایش عددی در کامپیوتر هستند. تعدادی از این مجموعه‌ها به صورت آماده وجود دارند که بر روی حجم زیادی از متن آموزش دیده‌اند و هر کلمه در این مجموعه‌ها یک بردار بازنمایی خاص دارد. در حال حاظر این مجموعه‌ها برای انجام کارهای متفاوت در حوزه پردازش متن مورد استفاده قرار می‌گیرند. یکی از ویژگی‌های بردارهای بازنمایی در این مجموعه‌های آماده این است که کلماتی که معانی مشابهی دارند (یا کلماتی که در یک حوزه خاص هستند مانند فوتبال و والیبال در ورزش) دارای بردارهای بازنمایی مشابهی می‌باشند اما به دلیل اینکه ابعاد بالای هر بردار بازنمایی امکان تشخیص بصری این موضوع ممکن نیست. هدف از این تمرین این است که با استفاده از الگوریتم PCA ابعاد یک مجموعه محدود از بردارهای بازنمایی کلمات موجود در مجموعه از پیش‌آموزش‌دیده GLoVE[2] را کاهش داده تا بتوان تشابه کلمات را بر روی یک نمودار دو

---

[1] embedding

[2] https://nlp.stanford.edu/projects/glove/

بعدی مشاهده کرد. مجموعه GLoVE یکی از معروف‌ترین مجموعه‌هایی است که برای حدود ۴۰۰ هزار کلمه در زبان انگلیسی بردار بازنمایی دارد.



توجه:

برای قسمت استفاده از PCA به دلخواه می‌تواند یک زیرمجموعه از کلمات در GLoVE را انتخاب کنید و برای آن‌ها بردار فشرده را بدست آورید. توجه داشته باشید که کلمات را به گونه‌ای انتخاب کنید که در صورت نمایش دادن بردارهای فشرده آنان بتوان متوجه تشابه و تفاوت‌های معنایی آن‌ها شد.

برای دانلود مجموعه GLoVE می‌توانید از سرویس google colab استفاده کنید. دانلود مجموعه با دستور زیر قابل انجام است:

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

بعد از دانلودکردن فایل GLoVE می‌توانید آن را با دستور زیر در Drive خود ذخیره کنید:

```
!cp "glove.6B.zip" "/content/drive/My Drive/MyDirectory"
```