دانشگاه تهران

دانشکده علوم مهندسی

الگوریتم‌ها و محاسبات

تکتم سمیعی

۸۱۰۸۹۶۰۵۴

یادگیری ماشین – دکتر سایه میرزایی

تمرین چهارم

بهار ۰۰

سوال اول

۱ . خوشه بندی K- میانگین :

الگوریتم means-K یکی از روش های خوشه بندی ساده و سریع است. این الگوریتم دارای یک پارامتر به نام k است که تعداد خوشه هایی که باید به دست آید را مشخص می کند. الگوریتم means-K پایه به صورت زیر است:

K داده را به عنوان مرکز خوشه انتخاب می کنیم، سپس فواصل بقیه داده‌ها با مرکز خوشه‌ها را تعیین می‌کنیم و داده‌هایی که به مرکز هر خوشه نزدیکتر هستند را در آن خوشه قرار می‌دهیم. میانگین هر خوشه را به عنوان مرکز جدید خوشه انتخاب می‌کنیم این مراحل را تا زمانی ادامه می‌دهیم که خوشه‌ها بدون تغییر باقی بمانند. در ادامه مراجل الگوریتم means-K بیان شده است.

- ورودی: خصوصیات n داده و k تعداد دسته ها
- خروجی: k دسته که داده های هر دسته از نظر شباهت به هم نزدیک و از دسته های دیگر دورند.

1. k داده را به عنوان مرکز خوشه انتخاب می کنیم.
2. مرحله سوم تا پنجم را تا رسیدن به عدم تغییر در خوشه ها تکرار می کنیم.
3. فواصل بقیه داده ها با مرکز خوشه ها را تعیین می کنیم.
4. داده هایی که به مرکز هر خوشه نزدیکترند در آن خوشه قرار می گیرند.
5. میانگین هر خوشه را به عنوان مرکز جدید خوشه در نظر می گیریم.

به طور معمول، مرکزخوشه‌های اولیه به صورت تصادفی از میان نمونه‌های اولیه گزینش می شوند. به همین دلیل، مرکز خوشه های اولیه در دو خوشه‌بندی مستقل means-K می توانند متفاوت باشند. این موضوع موجب می شود که خوشه های به جا مانده از دو اجرای مختلف means-K با هم متفاوت باشند. بنابراین همواره به بهینه ی سراسری نمی رسد اما ممکن است به بهینه ی محلی برسد. در الگوریتم means-K می توان از معیار های فاصله ی گوناگون بهره گرفت و خوبی یا بدی به کارگیری آن معیار بستگی به نوع داده هایی دارد که باید خوشه بندی شوند.

اگر یک ماتریس را به عنوان ورودی به means-K بدهیم، means-K این ماتریس را داده با خصوصیت در نظر می گیرد و اگر k را دو در نظر بگیریم داده ها را به دو دسته تقسیم می کند با این شرط که اعضایی در یک دسته قرار

می گیرند که خصوصیات آنها به یکدیگر نزدیک تر باشد یعنی سطر متناظر با آنها در ماتریس ورودی شباهت بیشتری به یکدیگر داشته باشد.

خوشه بندی K-Means قصد دارد پارامتر n اشیا را به خوشه های k که هر شیء آن را با نزدیکترین میانگین متعلق به خوشه قرار دهد. این روش دقیقا خوشه های مختلفی از بزرگترین تفاوت ممکن را تولید می کند . بهترین تعداد خوشه های k که منجر به بزرگترین جدایی (فاصله) می شود به عنوان یک پیش فرض شناخته نمی شود و باید از داده ها محاسبه شود.هدف از خوشه بندی K-Means این است که به حداقل رساندن واریانس کل درونی خوشه یا تابع خطای مربع:

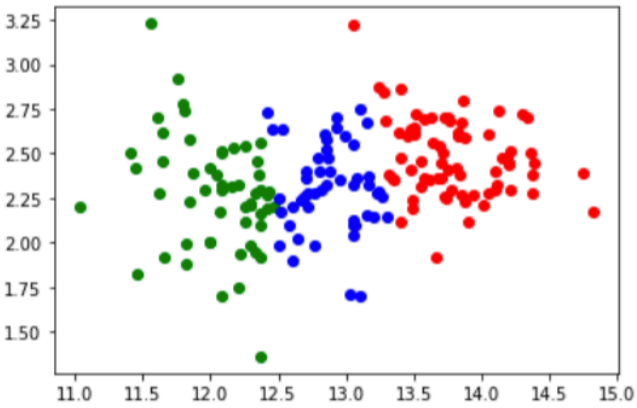
در روش خوشه بندی k-means، ما باید تعداد خوشه ها را قبل از عمل خوشه بندی مشخص کنیم و نتایج نهایی به مقدار k حساس هستند و اغلب در بهترین نقطه محلی متوقف می شوند . متاسفانه روش نظری جهانی برای پیدا کردن تعداد مطلوب خوشه ها وجود ندارد. یک رویکرد عملی، مقایسه نتایج چندین اجرا با k است و بهترین بر اساس یک معیار از پیش تعریف شده را انتخاب کنید.

۲ . الگوریتم k-means :

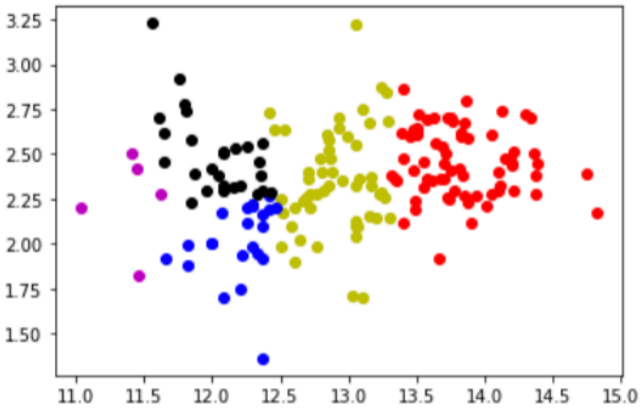
```
def K_means_Clustring (K,data):
    mu = [0]*K
    for i in range(K):
        mu[i]  = data.iloc[random.randint(0, 176)]
#     C = [0]*len(data)
    for counter in range(100):
        ##############
        ## Step one ##
        ##############
        for i in range(len(data)):
            temp = [0]*K
            for k in range(K):
                temp[k] = np.linalg.norm(data.iloc[i]-mu[k])
            data['C'][i] = temp.index(min(temp))
        ##############
        ## Step two ##
        ##############
        for k in range(K):
            mu[k] = data[ data['C'] == k ].mean()
    return data
```

نتایج به صورت زیر است :با توجه به نتایج به دست آمده به نظر میرسید که k=3 تعداد خوشه های مناسب تری برای clustring است .

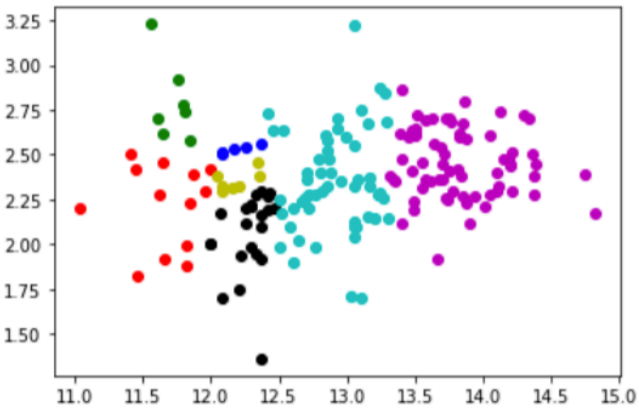
$k = 3$



$k = 5$



$k = 7$



۳- معیارهای شباهت درونی و بیرونی :

از آنجایی که برخلاف عملیات Classification ، Clustering یک فرآیند بدون نظارت است، بررسی صحت نتیجه خوشه‌بندی به راحتی امکان‌پذیر نیست. بنابراین احتیاج به معیارهای مناسب، هم برای بررسی کارایی یک روش خوشه‌بندی در بازیابی خوشه‌ها و هم برای مقایسه عملکرد روش‌های مختلف خوشه‌بندی، ضروری به نظر می‌رسد.از آنجایی که در این روش متغیر k را به عنوان ورودی میگیریم و از روی دیتاها آن را آموزش نمیبینم ، نمیتونیم مقدار صحیحی برای k از پیش در نظر بگیریم . در این میان دو گونه معیار ارزیابی نتایج خوشه‌بندی وجود دارد. معیارهای

ارزیابی درونی (Internal Criteria Index) و معیارهای ارزیابی بیرونی (External Criteria index). روشی که بیشترین شباهت درون خوشه یا بیشترین تمایز بین خوشه‌ها را ایجاد کند، روش مناسبی در نظر گرفته می‌شود.

معیارهای درونی

اگر داده‌های X که p بعدی هستند را با استفاده از روش خوشه‌بندی k-میانگین به k خوشه تفکیک کرده باشیم، می‌توان میزان پراکندگی کل را با T نشان داد و به صورت زیر محاسبه کرد:

$$T = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \overline{x})'(x_{ij} - \overline{x})$$

که در این رابطه ni تعداد داده های در خوشه ی i م و x_{ij} ، مشاهده ی jم از خوشه ی i م است . همچنین k نیز تعداد خوشه ها است . به این ترتیب اگر W میزان پراکندگی درون و B میزان پراکندگی بین خوشه‌ها باشد، روابط زیر برقرارند:

$$W = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \overline{x})'(x_{ij} - \overline{x})$$

$$B = \sum_{i=1}^{k} n_i (\overline{x}_i - \overline{x})'(\overline{x}_i - \overline{x})$$

می‌توان به راحتی نشان داد که بین این سه جزء رابطه W+B=T همیشه وجود دارد.در این میان، خوشه‌بندی که پراکندگی درون خوشه‌هایش (W) کمینه یا پراکندگی بین خوشه‌هایش (B) بیشینه باشد، مناسبترین روش خوشه‌بندی است. در ادامه به معرفی چند شاخص ارزیابی درونی می‌پردازیم که در «خوشه‌بندی تفکیکی» (Partition Clustering) کاربرد بیشتری دارند.

Elbow Method

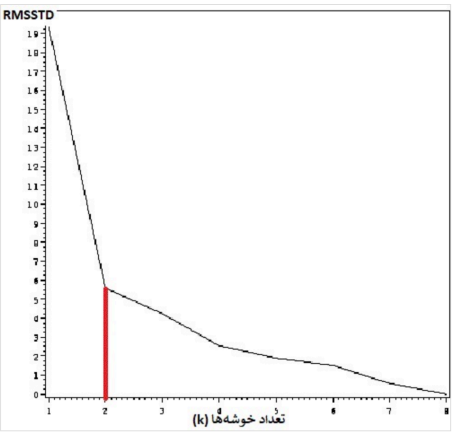
برای مشخص کردن تعداد خوشه‌ها در خوشه‌بندی تفکیکی می‌توان از شاخص ریشه میانگین مربعات انحراف استاندارد (square standard deviation-mean-Root) استفاده کرد. این شاخص میزان شباهت مقدارهای درون خوشه‌ها را اندازه‌گیری می‌کند . ابتدا علامت های زیر را تعیین میکنیم :

- SS_W : مجموع مربعات درون گروه ها
- SS_b :مجموع مربعات بین گروه ها
- SS_t : مجموع مربعات مقادیر

اگر k تعداد خوشه ها و p نیز بُعد داده ها باشد، آنگاه شاخص RMSSTD به صورت زیر محاسبه می‌شود:

$$v_{RMSSTD} = \sqrt{\frac{SS_w}{p(n-k)}}$$

برای تعیین بهترین تعداد خوشه ، برای مثال در روش خوشه‌بندی k-میانگین، کافی است RMSSTD را به ازای مقدارهای مختلف k محاسبه و در یک نمودار به صورت زیر ترسیم کنیم. در نقطه‌ای از محور افقی که شیب منحنی دارای تغییر محسوسی باشد، مناسب‌ترین خوشه‌بندی صورت گرفته است زیرا بعد از این شکستگی، با افزایش تعداد خوشه‌ها، تغییر زیادی در شاخصRMSSTD رخ نمی‌دهد. این نمودار را با نام (Elbow) می‌شناسند زیرا خمیدگی آن شبیه آرنج دست است.



با توجه به نمودار ، تعداد خوشه های k=2 بهترین مقدار برای خوشه بندی است زیرا پس از این مقدار تغییر محسوسی در کاهش معیار RMSSTD مشاهده نمیکنیم .

Bouldin-Davies

ابتدا به معرفی دو معیار زیر میپردازیم :

اندازه ی پراکندگی یا dispersion measure

اگر Si میزان پراکندگی مربوط به خوشه ی Ci و d نیز یک تابع فاصله باشد ، میزان پراکندگی به صورت زیر تعریف میشود :

$$S_i = [\frac{1}{C_i}\Sigma d^r(x, c_i)]^{\frac{1}{r}}$$

عدم شباهت بین خوشه ها یا Cluster dissimilarity

فاصله ی بین دو خوشه بر اساس فاصله ی بین دو نقطه ی مرکزی آن سنجیده میشود . فرض کنید Vi و Vj دو نقطه ی مرکزی دو دو خوشه ی i و j باشند ، در این صورت فاصله ی بین دو خوشه را به صورت زیر مینویسیم :

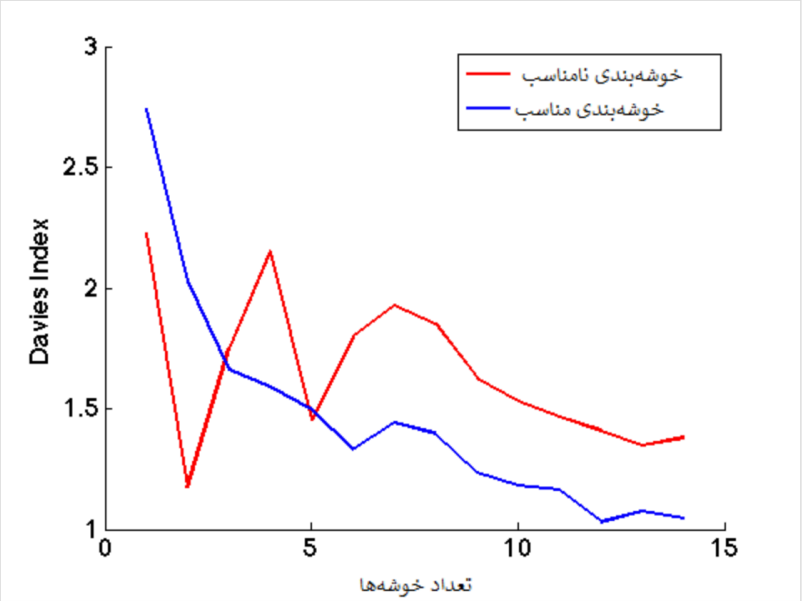
$$D_{ij} = [\Sigma d(V_i, V_j)^t]^{\frac{1}{t}}$$

و حالا فاصله ی بین دو خوشه را به صورت زیر تعریف میکنیم :

$$R_{ij} = \frac{S_i + S_j}{D_{ij}}$$

با توجه به تعاریف فوق ، هرچه مقدار R بزرگ تر باشد ، میزان پراکندگی هر خوشه بیشتر است و یا فاصله ی بین دو خوشه کمتر است . پس باید برای ارزیابی مدل به دست آمده فاصله ی هرخوشه تا خوشه های دیگر را به دست

آوریم و ماکزیمم آن را حساب کنیم و سپس برای همه ی خوشه ها این مقادیر به دست آمده را جمع کنیم و بر تعداد k تقسیم کنیم . این شاخص را V_{DB} نمایش میدهند . هرچه این مقدار کمتر باشد ، مدل بهتری داریم .



Silhouette

در این روش سه پارامتر ai و bi و si را برای هر داده محاسبه میکنیم ، سپس میانگین متغیر si را برای دادگان درون یک خوشه محاسبه میکنیم و در نهایت میانگین کل si را برای ارزیابی مدل محاسبه میکنیم :

$$a(i) = \frac{1}{n}\Sigma d(xi, xj)$$

این متغیر میانگین فاصله ی داده ی xi با تمام داده های دیگر در یک کلاس را محاسبه میکند .

$$b(i) = min\frac{1}{n_l}\Sigma d(xi, xj)$$

این متغیر میانگین فاصله ی داده ی xi با تمام داده های کلاس های دیگر را محاسبه میکند و min آن را برمیگرداند ، به این معنا که به دنبال نزدیک ترین خوشه به خوشه ی فعلی است .

و در نهایت :

$$s(i) = \frac{b(i) - a(i)}{max(b(i), a(i))}$$

اگر si منفی باشد ، به این معنا است که فاصله ی خوشه ی همسایه از خوشه ی فعلی به داده ی مورد نظر کمتر است و خوشه بندی به خوبی صورت نگرفته و اگر مقدار آن مثبت باشد یعنی خوشه بندی به درستی صورت گرفته است و میانگین فاصله از داده های خوشه خود ، از میانگین فاصله ی داده های خوشه همسایه کمتر است .

معیار های بیرونی

در روش ارزیابی با معیارهای بیرونی، برای همه نقاط یک برچسب واقعی وجود دارد که نشان دهنده تعلق نقاط به دسته‌ها است.از طرفی برچسب‌های خوشه‌بندی نیز کد مربوط به خوشه‌ای است که یک نقطه درون آن قرار دارد.
دسته هارا با S و خوشه هارا با C نشان میدهیم .

Purity Index

در این روش برچسب هر خوشه با برچسب واقعی دسته‌ای که بیشترین اشتراک را دارد مطابقت پیدا کرده و تعداد نقاطی از خوشه که در دسته صحیح طبقه‌بندی شده‌اند ، شمارش می‌شوند. نسبت این تعداد به تعداد کل نقاط شاخص خلوص را می‌سازد که به صورت زیر است :

$$Purity(S, C) = \frac{\sum_m \max_n |S_m \cap C_n|}{N}$$

در واقع در این روش ، تعداد نقاط مشترک در خوشه ی Cn با دسته ی Sm محاسبه میشود و بر تعداد کل تقسیم میشود . حداکثر مقدار Purity برابر ۱ است وحداقل مقدار آن برابر ۰ است . طبیعتاهرچه این مقدار بیشتر باشد ، بهتر است .

F - score

ابتدا دو معیار precision و recall را به صورت زیر براساس true positive ، true negative ، false و positive و false negetive تعریف می کنیم :

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

در این صورت F-score با تعریف ضریب β که در واقع برای تغییر وزن دو پارامتر فوق به صورت زیر تعریف میشود :

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 Precision + Recall}$$

۴ - معیار ارزیابی درونی : Silhouette

Internal Index : Silhouette

```
def Silhouette(df_grouped,K):
    Si = [0]*K
    for i in range(len(df_grouped)):
        ## for cluster i
        cluster = df_grouped.get_group(i)
        cluster = cluster.set_axis(np.arange(0, len(cluster) , 1), axis='index')
        s_i_cluster = 0
        for j in range(len(cluster)):
            ## for each data xi in this cluster
            a_i = ai(cluster.loc[j].at["Alcohol"],cluster.loc[j].at["Ash"],cluster)
            counter = 0
            for m in range(len(df_grouped)):
                temp = [0]* (K-1)
                if m != i and counter < K:
                    neighbor_cluster = df_grouped.get_group(m)
                    temp[counter] = ai(cluster.loc[j].at["Alcohol"],cluster.loc[j].at["Ash"],neighbor_cluster)
                    counter +=1
            b_i = min(temp)
            s_i_cluster += (b_i - a_i)/max(b_i , a_i)
        Si[i] = s_i_cluster/len(cluster)
        total_mean_si = mean(Si)
    return Si , total_mean_si
```

```
def distance(xi,yi,xj,yj):
    return np.sqrt((xi-xj)**2 + (yi-yj)**2)
```

```
def ai(xi,yi,cluster):
    Sum = 0
    for i in range(len(cluster)):
        Sum += cluster.apply(lambda row : distance(xi,yi,row['Alcohol'],row['Ash']), axis = 1)
#         print("Summ in ai",Sum)
    return Sum.sum()/len(cluster)
```

```
Si_3 , total_mean_si_3 = Silhouette(df3,3)
Si_5 , total_mean_si_5 = Silhouette(gk5,5)
Si_7 , total_mean_si_7 = Silhouette(gk7,7)
```

در این روش همانند آنچه در توضیحات بالا گفته شد روش silhouette پیاده سازی شد وبرای ۳ حالت مختلف k

بررسی شد .

۵ - به طور کلی ، بهترین مقدار k زمانی به دست می آید که شیب نمودار خطا بر حسب مقادیر مخلف k ، مقدار زیادی کاهش یابد و پس از آن ، تغییر محسوسی در اندازه ی خطا مشاهده نشود و به عبارتی نمودار تقریبا flat شود .برای این کار میتوانیم از روش های elbow and silhouette و نمودار های حاصل از آن ها همان طور که پیشتر توضیح دادم ، استفاده کرد .

hierarchical clustering - سوال دوم

معیارهای Simple matching coefficient و Jaccard coefficient برای هر جفت داده محاسبه کرده و جدول های زیر حاصل شد :

SMC

	x1	x2	x3	x4	x5	x6
x1	1	0.5	0.66	0.25	0.5	0.25
x2	0.5	1	0.25	0.66	0.2	0.25
x3	0.66	0.25	1	0.33	0.25	0.33
x4	0.25	0.66	0.33	1	0	0.33
x5	0.5	0.2	0.25	0	1	0.25
x6	0.25	0.25	0.33	0.33	0.25	1

JC

	x1	x2	x3	x4	x5	X6
x1	1	0.6	0.8	0.4	0.6	0.4
x2	0.6	1	0.4	0.8	0.2	0.4
x3	0.8	0.4	1	0.6	0.4	0.6
x4	0.4	0.8	0.6	1	0	0.6
x5	0.6	0.2	0.4	0	1	0.4
x6	0.4	0.4	0.6	0.6	0.4	1

برای خوشه بندی سلسله مراتبی single linkage با استفاده از معیار SMC به صورت زیر عمل میکنیم :

کوچک ترین داده را از بین داده های جدول پیدا میکنیم ، سطر و ستون مربوط به آن داده را حذف میکنیم و چون جدول متقارن است ، در مجموع ۴ سطو و ستون حذف میشوند به انتهای جدول یه سطر و ستون اضافه میکنیم با نام جدید . پس در هر مرحله ابعاد جدول یک شماره کاهش می یابد .

سپس برای پرکردن سط و ستون جدید ، مینیم مقدار دو ستون حذف شده در متغیر مربوطه را جاگذاری میکنیم :

در ابتدا جدول به صورت زیر است :

	x1	x2	x3	x4	x5	X6
x1	1	0.6	0.8	0.4	0.6	0.4
x2	0.6	1	0.4	0.8	0.2	0.4
x3	0.8	0.4	1	0.6	0.4	0.6
x4	0.4	0.8	0.6	1	0	0.6
x5	0.6	0.2	0.4	0	1	0.4
x6	0.4	0.4	0.6	0.6	0.4	1

کمترین cell مقدار 0 را دارد که فاصله ی داده ی x4 و x5 است . سطر و ستون مربوط به این ویژگی هارا حذف کرده و سطر و ستون جدیدی به نام y1 اضافه میکنیم :

	x1	x2	x3	X6	Y1
x1	1	0.6	0.8	0.4	0.4
x2	0.6	1	0.4	0.4	0.2
x3	0.8	0.4	1	0.6	0.4
x6	0.4	0.4	0.6	1	0.4
Y1	0.4	0.2	0.4	0.4	1

مقدار cell در y1 و x1 ، مینیم مقدار x1 در دو ستون حذف شده ی x4 و x5 است که برابر است با 0.4 .

به همین ترتیب برای بقیه ی خانه ها مقادیر را پیدا میکنیم .

سپس در این مرحله ، چون برابر 0.2 است ، سطر و ستون مربوط به خوشه های y1 و x2 را حذف میکنیم و کانند قبل ، جدول را کامل میکنیم :

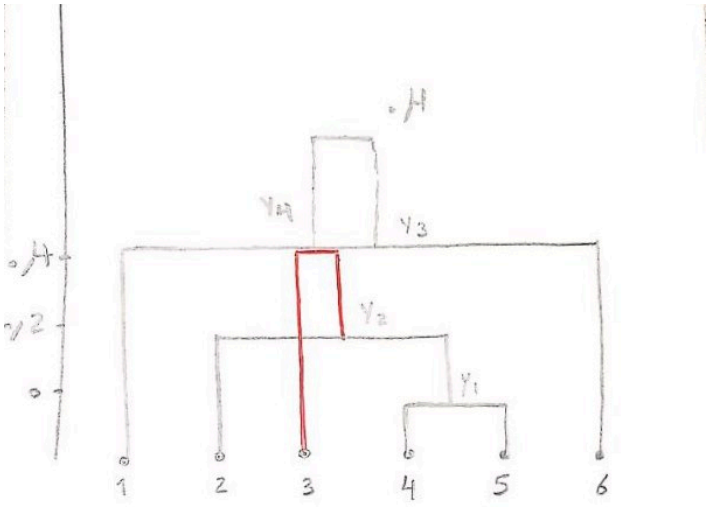
	x1	x3	X6	y2
x1	1	0.8	0.4	0.4
x3	0.8	1	0.6	0.4
x6	0.4	0.6	1	0.4
y2	0.4	0.4	0.4	1

در این جدول ، کوچک ترین مقدار برابر 0.4 است و تعدادی از cell های جدول این مقدار را دارند ، پس به دلخواه یک cell را در نظر میگیریم . سطر و ستون مربوط به x1 و x6 را در نظر میگیریم و آن هارا حذف میکنیم :

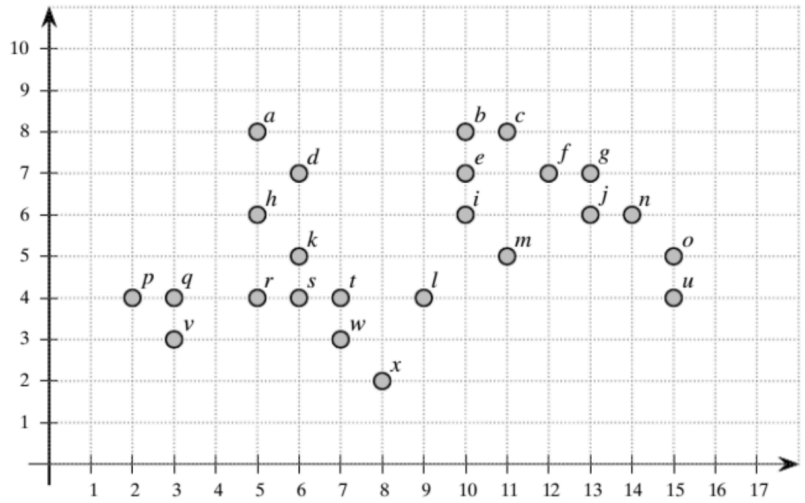
	x3	y2	y3
x3	1	0.4	0.6
y2	0.4	1	0.4
y3	0.6	0.4	1

	y3	y4
y3	1	0.4
y4	0.4	1

در نهایت مقدار 0.4 از جدول بالا حاصل میشود .



سوال سوم - DBSCAN



ابتدا از نقطه ی q شروع میکنیم و فاصله ی نقاط اطراف آن را محاسبه میکنیم . این کار را از نزدیک ترین نقاط نقطه ی q شرو میکنیم . جدول زیر فاصله ی نقاط مختلف از نقطه ی q است :

مختصات نقطه ی q	مختصات نقطات اطراف	فاصله
(3,4)	p=(2,4)	1
(3,4)	v=(3,3)	1
(3,4)	r=(5,4)	2
(3,4)	h=(5,6)	$2\sqrt{2}$
(3,4)	a=(5,8)	$2\sqrt{5}$

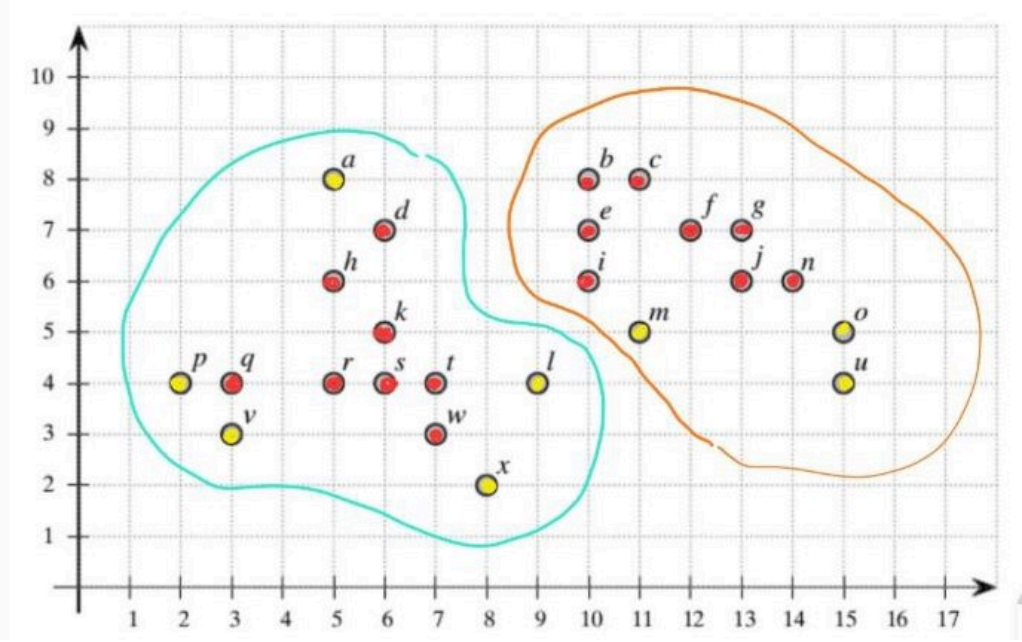
با توجه به این که $\epsilon = 2$ ، فقط نقاط p , r و v در این شعاع هستند و بقیه نقاط خارج از این شعاع هستند و به محاسبه فاصله ی آن ها از نقطه ی q نپرداختیم . نقاط p , r , q و v چون به تعداد minpts میرسند ، نقطه ی q نقطه ی مرکزی میشود و نقاط p و v ، r نقاط border هستند .

سپس نقطه ی q را که بررسی نمودیم ، به ترتیب به محاسبه ی نقاط اضافه شده به این cluster میپردازیم . ابتدا نقطه ی p. را بررسی میکنیم . تنها نقاط v , q در شعاع اپسیلون نقطه ی p قرار دارند ، پس این نقطه core نیست و یک border point است . سپس به بررسی نقطه ی v میپردازیم . این نقطه نیز مانند نقطه ی p ، فقط نقاط p , q را در شعاع اپسیلون خود دارد و border point است . سپس به بررسی نقطه ی r میپردازیم .

مختصات نقطه ی r	مختصات نقطات اطراف	فاصله
(5,4)	p=(2,4)	1
(5,4)	v=(3,3)	1
(5,4)	h=(5,6)	2
(5,4)	s=(6,4)	1
(5,4)	k=(6,5)	$\sqrt{2}$

همانطور که مشاهده میکنید ، نقاطی که در بازه ی اپسیلون نقطه ی r قرار دارند بیشتر از minpts هستند و این نقطه ، یک نقطه ی مرکزی است . به همین ترتیب برای بقیه نقاط این الگوریتم را چک میکنیم و در نهایت دو خوشه برای این دیتاست حاصل میشود که به صورت زیر از هم جدا میشوند :

نقاط مرکزی را با قرمز و نقاط border را با زرد مشخص کردم . این مجموعه دارای outlier نیست .



سوال چهارم - PCA

PCA

ابتدا داده های Iris را لود میکنم و به هر ستون آن لیبل میدهم و ستون پنجم آن را حذف میکنم .

توسط تابع normalize ، داده هارا نرمال میکنم . در تابع PCA ، ابتدا تابع covariance داده های نرمال را محاسبه میکنم ، سپس از تابع svd استفاده میکنم و بردار ویژه های این ماتریس را به دست می آورم . بردار ویژه های داده شده توسط تابع svd بر اساس اهمیت آن ها ، یعنی مقدار مقدار ویژه ی مربوطه مرتب میشوند . چون قصد داریم داده هارا به ۲ بعد کاهش دهیم ، ۲ بردار ویژه ی اول را انتخاب میکنم .

سپس با ضرب ماتریسی دیتافریم داده ها (Iris)با ماتریس بردار ویژه های کاهش داده شده(U_reduced) ، ماتریس Z را که دارای بعد m*2 است را به دست می آوریم و آن را رسم میکنیم .

```
Iris = pd.read_csv("iris.data")
```

```
Iris.columns = ['X1','X2','X3','X4','X5']
Iris.drop('X5', inplace=True, axis=1)
```

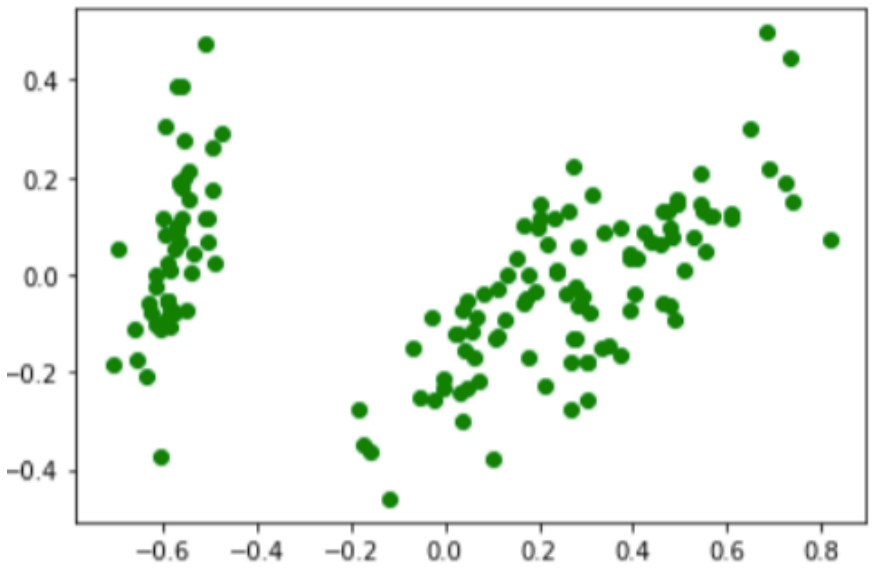
```
def normalize(df):
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        mean_value = df[feature_name].mean()
        df[feature_name] = (df[feature_name] - mean_value) / (max_value - min_value)
    return result
```

```
normalized_Iris = normalize(Iris)
```

```
def PCA(normalized Iris,k):
    covariance = normalized Iris.cov()
    u, s, vh = np.linalg.svd(covariance)
    U_reduced = -u[:,:k]
    Z = np.matmul(Iris,U reduced)
    return Z , U reduced
```

```
Z , U_reduced = PCA(normalized_Iris,2)
```

```
plt.plot(Z.iloc[:,0], Z.iloc[:,1], 'go')
```



Power Method

در این روش ، با ضرب متوالی ماتریس داده در بردار اولیه و دلخواه x ، و نرمال کردن بردار x در هر مرحله به وسیله ی تقسیم آن بر ماکزیمم مقدار آن ، این ماکزیمم مقدار همان بزرگترین لاندا را تشکیل میدهد ، بردار x را آپدیت میکینم . این کار را تا جایی انجام میدهیم که بردار x تغییرات اندکی داشته باشد . به این ترتیب بزرگتریم مقدار ویژه و بردار ویژه ی مروبط به آن را پیدا میکنیم .

```
def powerMethod_landa_1(A,iteration):
    x = np.array([1, 1, 0, 0])
    for i in range(iteration):
        x = np.dot(A, x)
        landa_1 = abs(x).max()
        x = x / x.max()
    return landa_1 , x
```

برای یافتن دومین بردار ویژه ، به جای ضرب ماتریس داده ها به صورت متوالی در ماتریس اولیه و دلخواه x ، ماتریس $A - \lambda_1 I$ را در آن ضرب میکنیم و به این صورت دومین بردار ویژه هم به دست می آید . در نظر داشته باشید که بردار های ویژه های به دست آمده باید نرمال شوند .

```
def powerMethod_landa_2(A,landa_1,iteration):
    shifted_A = A - landa_1 * np.identity(len(A))
    Lan , x_2 = powerMethod_landa_1(shifted_A,iteration)
    landa_2 = Lan + landa_1
    return landa_2 , x_2
```

سپس مانند قبل بردار های ویژه ی را concatenate کرده و با ضرب در داده های نرمال اولیه ،داده های کاهش بعد یافته را به دست می آوریم :

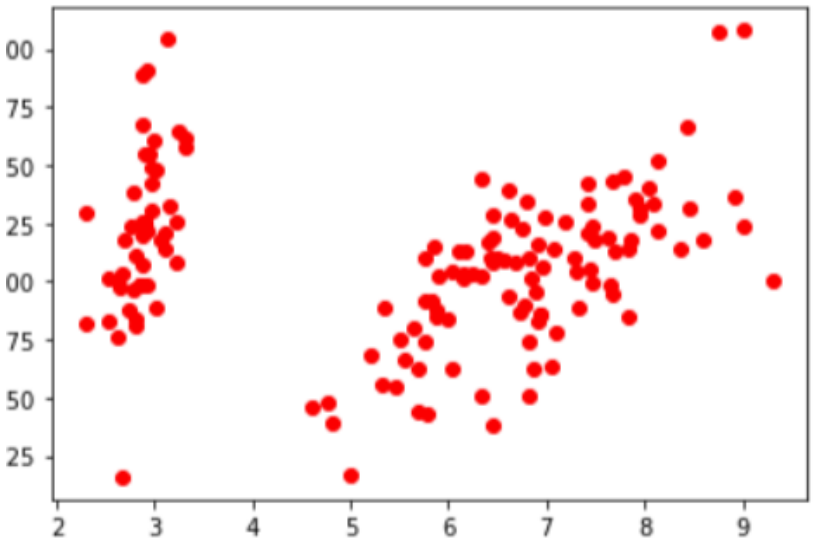
```
covariance = normalized_Iris.cov()
landa_1 , x_1 = powerMethod_landa_1(covariance,100)
landa_2 , x_2 = powerMethod_landa_2(covariance,landa_1,100)
```

```
X_1 = (x_1/np.linalg.norm(x_1)).reshape(4,1)
X_2 = (x_2/np.linalg.norm(x_2)).reshape(4,1)
U = np.concatenate((X_1,X_2), axis=1)
U
```

```
array([[ 0.36263433, -0.03218843],
       [-0.08122848,  0.90930602],
       [ 0.85629752,  0.24126574],
       [ 0.35868209, -0.33751638]])
```

```
Z_PM = np.matmul(normalized_Iris,U)
```

```
plt.plot(Z_PM.iloc[:,0], Z_PM.iloc[:,1], 'ro')
```



Kernel PCA

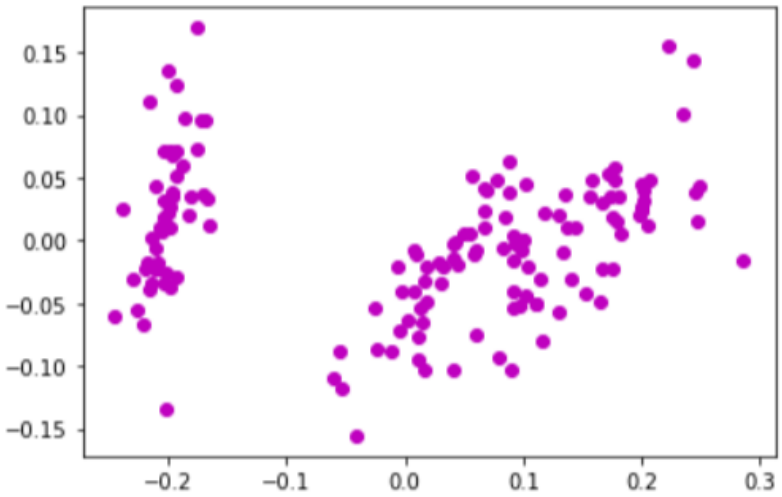
از توابع آماده ی kernelPCA از کتابخانه ی sklearn استفاده کردم . نتایج زیر حاصل شد :

```
from sklearn.decomposition import KernelPCA
```

```
transformer = KernelPCA(n_components=2, kernel='sigmoid')
Iris_transformed = transformer.fit_transform(Iris)
Iris_transformed.shape
```

```
(149, 2)
```

```
plt.plot(Iris_transformed[:,0], Iris_transformed[:,1], 'mo')
```



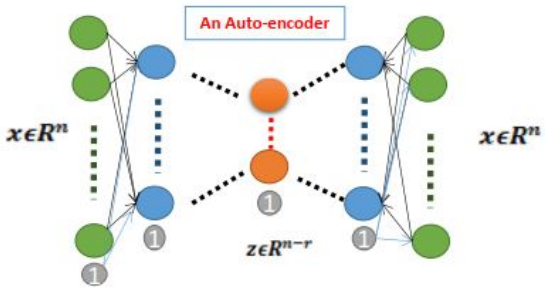
AutoEncoder

این روش که مبتنی بر شبکه عصبی کار می کند ، به این شکل عمل می کند که ابتدا یک شبکه Auto-Encoder را با استفاده از مجموعه داده ای که می خواهیم روی آن کاهش دهیم آموزش می دهیم و سپس قسمت Encoder آن را جدا می کنیم و به عنوان یک شبکه کاهش دهنده ابعاد قبل از مدل اصلی استفاده می کنیم.

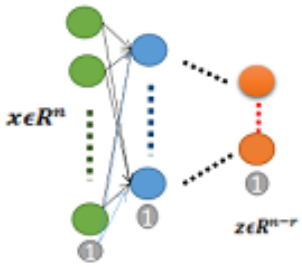
معماری Encoder-Auto به این صورت است که تعداد نورون ورودی و خروجی آن برابر با تعداد ویژگی های تصویر یا داده ورودی است ، در لایه میانی Encoder-Auto لایه ای وجود دارد که قسمت Encoder و

Decoder را متمایز می کند ، اگر تعداد ویژگی داده های ورودی را برابر با N در نظر بگیریم ، تعداد ویژگی داده های خروجی نیز برابر N می شود و اگر R تا از این ویژگی ها کاهش یابند ، تعداد نورون لایه میانی برابر با N - R خواهد بود!

به طور کلی در بخش Encoder ابعاد کاهش و در بخش Decoder ابعاد افزایش پیدا می کند. برای آموزش شبکه به این صورت عمل می شود که داده ها به ورودی Auto- Encoder داده می شوند و انتظار می رود در خروجی Auto- Encoder همان داده ها دیده شود ، چندین Epoch این کار برای همه داده ها تکرار می شود تا زمانی که تا حد ممکن نتایج خروجی برابر با ورودی ها شود!



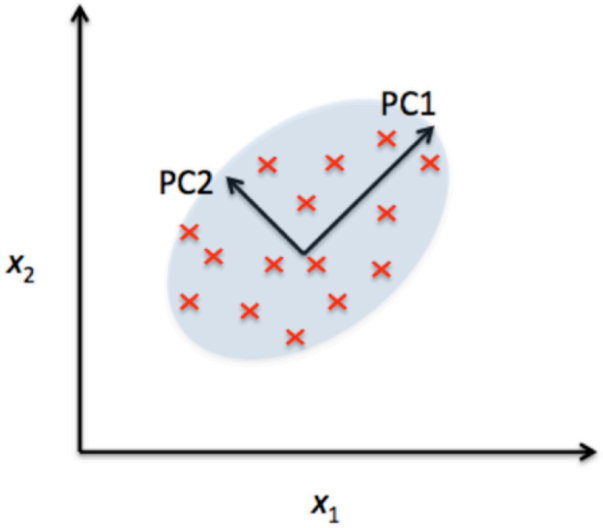
در نهایت پس از پایان آموزش Encoder-Auto ، قسمت Encoder جدا می شود و قبل از شبکه اصلی برای کاهش بعد داده های ورودی استفاده می شود!



برای افزایش دقت میتوان تعداد لایه های درونی را افزایش داد .

LDA vs PCA

هردو الگوریتم ، تکنیک های تبدیل خطی هستند برای کاهش بعد ، در حالیکه PCA بدون نظارت (unsupervised) و LDA با نظارت (supervised) است . الگوریتم PCA لیبل داده هارا نادیده میگیرد و تلاش میکند جهت های حداکثر واریانس را پیدا کند .



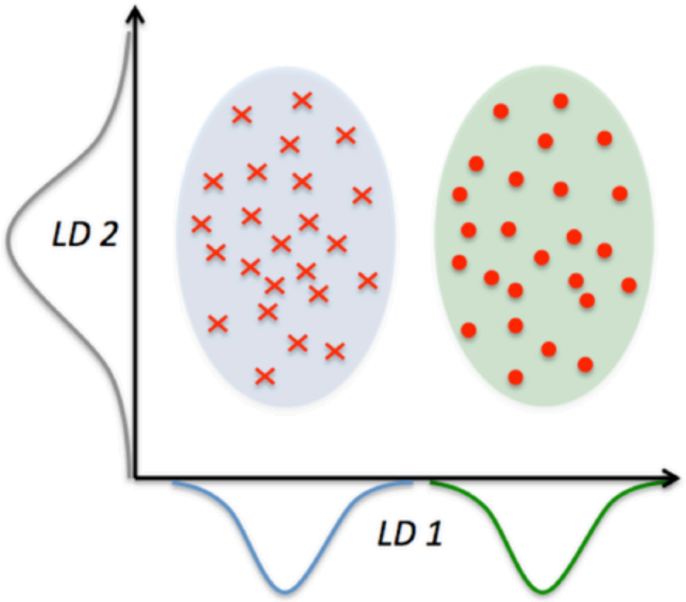
از PCA برای دو منظور استفاده میکنیم :

- کاهش بعد
- تصویر سازی کلاس ها

PCA یک متد feature selection نیست یلکه متد feature combination است .

LDA روشی است که برای جداسازی دو کلاس به کار میرود . ایده ی کلی LDA این است که تفکیک پذیری دو کلاس را ماکزیمم کنیم تا بتوانیم بهترین تصمیم را برای classify دو کلاس بگیریم .

در واقع هردو روش از جهت کاهش ابعاد شبیه هم هستند اما LDA به وسیله ی ایجاد محور خطی جدید و project کردن نقاط روی این محور جدید ،سعی در ماکزیمم کردن جدایی پذیری میان کتگوری های معلوم دارد .


LD 2
LD 1