یادگیری ماشین برای بیوانفورماتیک

تمرین اول، بخش تئوری

جواد راضی (۴۰۱۲۰۴۳۵۴)

۱ - رگرسیون خطی

۱.۱)

می‌دانیم:

$$W = (X^T X)^{-1} X^T Y$$

از آن‌جایی که در این سوال، یادگیری فقط با یک فیچر انجام شده‌است، ماتریس X یک ماتریس n x 1 است. پس داریم:

$$\omega_j = \left[\sum_{i=1}^{n} \left(x_j^{(i)}\right)^2\right]^{-1} x_j^T Y$$

$$\Rightarrow w_j = \frac{x_j^T y}{\sum_{i=1}^{n} \left(x_j^{(i)}\right)^2} = \frac{x_j^T y}{x_j^T x_j}$$

(لازم به ذکر است که حاصل ضرب ترانهاده ماتریس x_j در خود ماتریس یک اسکالر است، و معکوس آن، برابر با معکوس اسکالر است، چرا که تنها در این حالت است که ضرب دو ماتریس ماتریس مشخصه 1x1 را حاصل می‌شود.)

۱.۲)

ماتریس وزن‌ها از این رابطه بدست می‌آید:

$$W = (X^T X)^{-1} X^T Y$$

اگر ستون‌های X متعامد باشند، می‌دانیم:

$$Z_{i,j} = (x^T x)_{i,j} = \langle x_i^T, x_j \rangle = \begin{cases} 0 & i \neq j \\ \langle x_j^T, x_j \rangle & o.w \end{cases}$$

بنابراین ماتریس Z، یک ماتریس قطری خواهد بود که درایه jام قطر آن، برابر حاصل‌ضرب ترانهاده ستون j ماتریس x در خودش است.

می‌دانیم معکوس یک ماتریس قطری، یک ماتریس قطری است که درایه‌هایش معکوس ماتریس اصلی هستند. بنابراین، $Z_{i,j}^{-1}$، برابر است با $\frac{1}{\langle x_j^T, x_j \rangle}$

اکنون داریم:

$$w_j = z_j^{-1} x^T y = \frac{(x^T y)_j}{x_j^T x_j} = \frac{x_j^T y}{x_j^T x_j}$$

بنابراین، بردار وزن‌های هر فیچر jام، مستقلا از مقادیر ستون j ام ماتریس x حاصی می‌شوند و وزن بدست یافته در اینجا برای کل فیچرها، همانند قسمت قبل شده‌است که معادل بدست‌آوردن وزن هر فیچر به صورت جداگانه می‌باشد.

PCA -۲

(۲.۱

طبق گفته سوال:

$$\hat{x}_i = V_{1:k} z_i$$
$$\rightarrow ||\hat{x}_i - \hat{x}_j|| = ||V_{1:k} z_i - V_{1:k} z_j|| = ||V_{1:k}(z_i - z_j)||$$

چون ماتریس $V_{1:k}$ متعامد است، و می‌دانیم نرم حاصل‌ضرب یک ماتریس متعامد در یک وکتور، برابر نرم همان وکتور است داریم:

$$||V_{1:k}(z_i - z_j)|| = ||z_i - z_j||$$

(۲.۲

طبق گفته سوال:

$$\hat{x}_i = V_{1:k} z_i = V_{1:k} V_{1:K}^T x_i$$
$$\rightarrow \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2 = \sum_{i=1}^{n} \|x_i - V_{1:k} V_{1:K}^T x_i\|_2^2 = \sum_{i=1}^{n} \|(I - V_{1:k} V_{1:K}^T) x_i\|_2^2$$

از طرفی می‌دانیم که ماتریس V در PCA یک ماتریس orthogonal است. طبق تعریف ماتریس orthogonal داریم:

$$VV^T = I$$

اکنون فرض کنیم ماتریس $V_{1:k}$ شامل k بردار ویژه نخست و ماتریس $V_{k+1:p}$ شامل p-k بردار ویژه دیگر باشد. اکنون داریم:

$$VV^T = \left(V_{1:k} + V_{k+1:p}\right).\left(V_{1:k}^T + V_{k+1:p}^T\right)$$
$$= V_{1:k} V_{1:k}^T + V_{k+1:p} V_{k+1:p}^T + \cancel{V_{1:k} V_{k+1:p}^T} + \cancel{V_{k+1:p} V_{k+1:p}^T}$$
$$\rightarrow VV^T = I = V_{1:k} V_{1:k}^T + V_{k+1:p} V_{k+1:p}^T$$
$$\rightarrow I - V_{1:k} V_{1:k}^T = V_{k+1:p} V_{k+1:p}^T$$
$$\rightarrow \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2 = \sum_{i=1}^{n} \left\|V_{k+1:p} V_{k+1:p}^T x_i\right\|_2^2$$

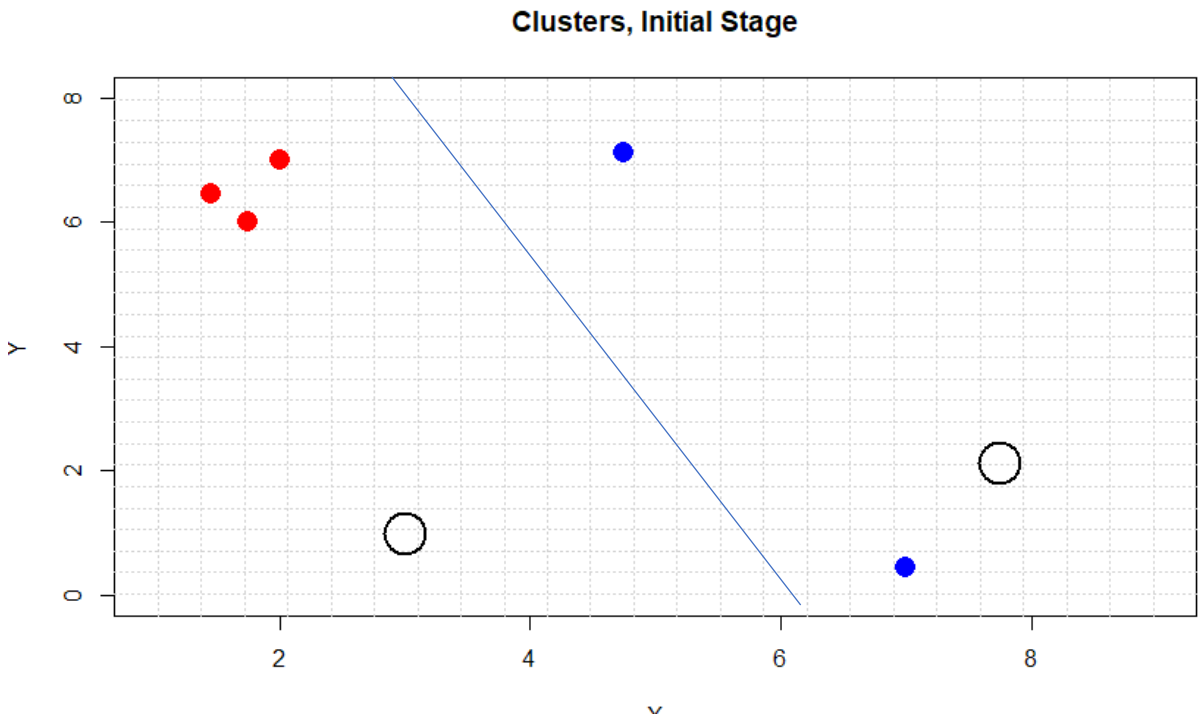
از طرفی می‌دانیم که $\|V\|_2^2 = \|VV^*\|_2 = \|VV^T\|_2$. بنابراین داریم:

$$\sum_{i=1}^{n} \left\|V_{k+1:p} V_{k+1:p}^T x_i\right\|_2^2 = \sum_{i=1}^{n} \left\|V_{k+1:p} V_{k+1:p}^T x_i x_i^T V_{k+1:p} V_{k+1:p}^T\right\|_2 = \sum_{i=1}^{n} V_{k+1:p}^T x_i x_i^T V_{k+1:p}$$
$$\sum_{i=1}^{n} V_{k+1:p}^T [(n-1) Cov(x_i)] V_{k+1:p} = (n-1) \sum_{j=k+1}^{p} \lambda_j$$

K-Means -۳

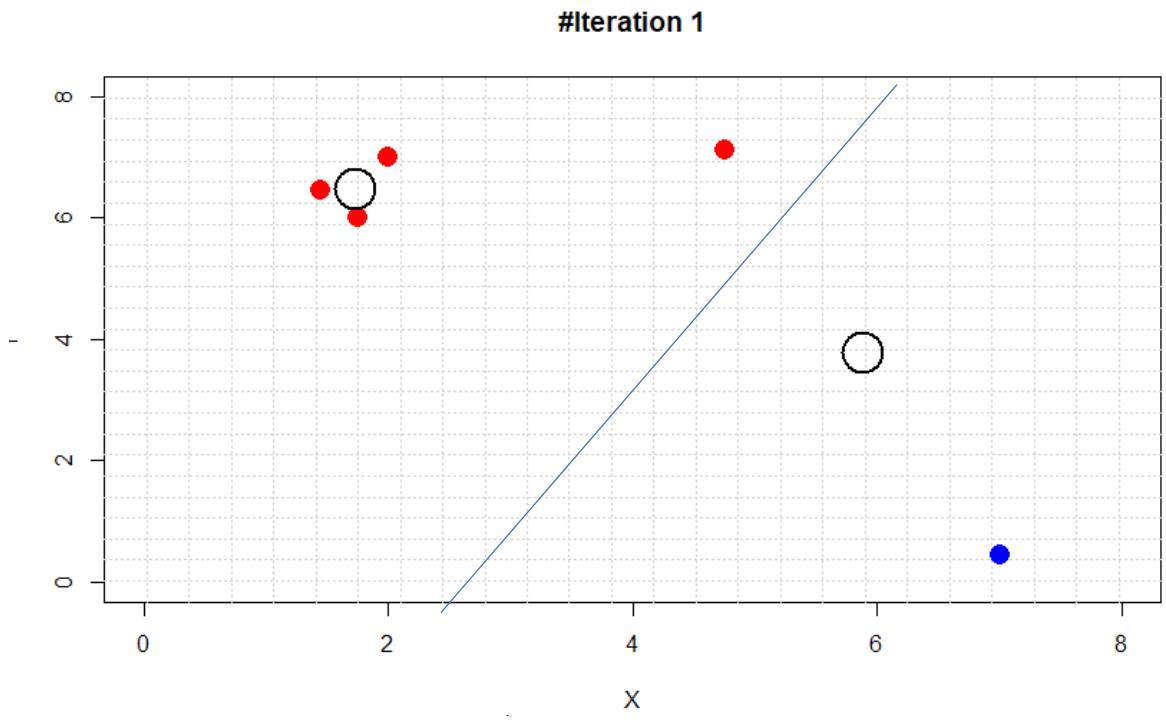
(۳.۱

ایتریشن ۰: وضعیت آغازین مراکز خوشه‌ها

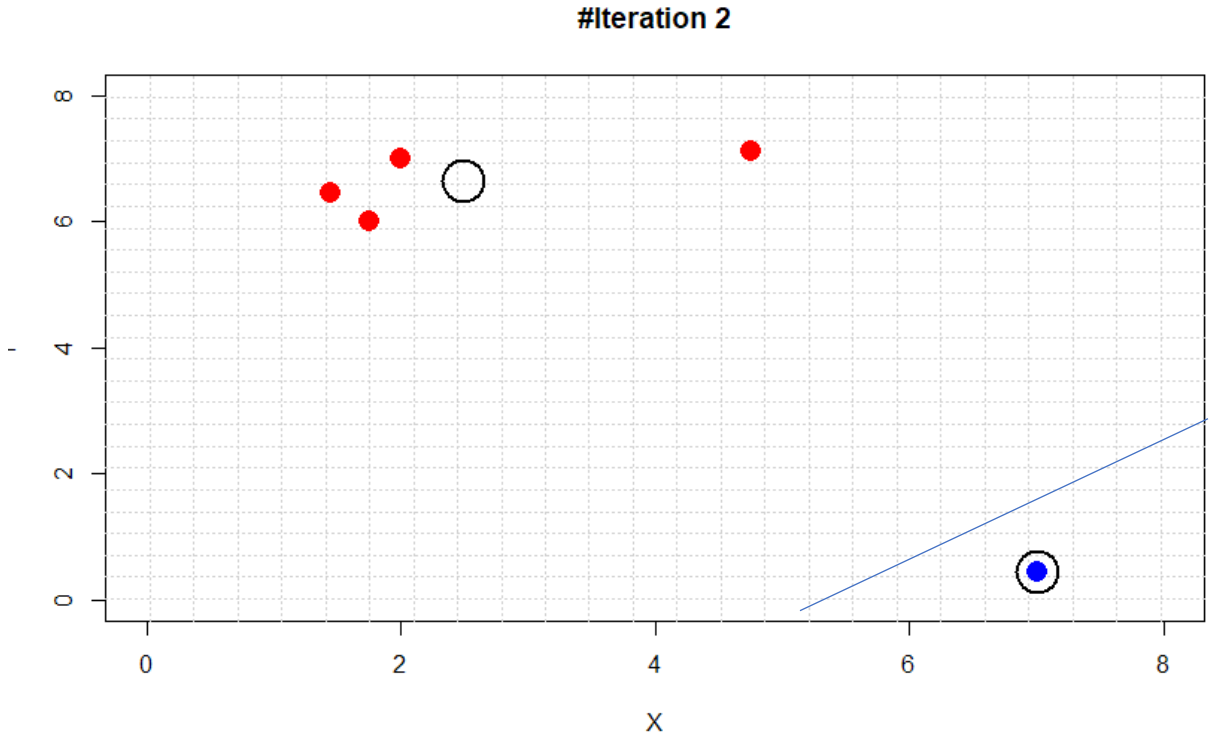


ایتریشن ۱: بروزرسانی مراکز خوشه‌ها و خوشه‌بندی جدید

موقعیت جدید مراکز خوشه‌ها، از میانگین گرفتن موقعیت نقطه‌های هر خوشه به‌دست می‌آید. پس از مشخص کردن مکان جدید، نقطه‌ها با توجه اینکه به کدام یک از دو مرکز آپدیت‌شده نزدیک‌ترند، مجددا خوشه‌بندی می‌شوند. در اینجا مشاهده می‌شود که نقطه شماره ۴ به خوشه مربع ۷ اساین شد.



ایتریشن دو: بروزرسانی و همگرایی نهایی موقعیت مراکز خوشه‌ها



(۳.۲

مزایای K-Means Clustering:

- پیچیدگی محاسباتی کم‌تری دارد. بنابراین برای دیتاست‌های عظیم می‌تواند گزینه مناسب‌تری باشد. در این الگوریتم، پیچیدگی زمانی در هر ایتریشن O(kN) است که N تعداد دیتاپوینت‌ها و k تعداد خوشه‌ها می‌باشد.
- مصرف حافظه این الگوریتم نیز پایین‌تر است. در خوشه‌بندی سلسله‌مراتبی، به ذخیره یک ماتریس nدرn نیاز است.
- در الگوریتم k-means، چون انتخاب مختصات مرکز خوشه بر اساس میانگین فاصله نقاط اختصاص‌یافته به خوشه، و اختصاص نقاط به خوشی بر اساس فاصله اقلیدسی از مرکز تعیین می‌شود، برای دیتاست‌هایی که الگوی داده در آن‌ها ساختاری کره‌مانند دارد مناسب‌تر است.
- همگرایی در الگوریتم k-means تضمین شده‌است.

مزایای Hierarchal Clustering:

- برخلاف k-means که باید تعداد خوشه‌ها از پیش مشخص باشد، در خوشه‌بندی سلسله‌مراتبی، می‌توان خوشه‌بندی را هنگامی که به تعداد دلخواه خوشه رسیدیم متوقف کنیم، یا با توجه به مناسب‌بودن خوشه‌بندی.
- برای خوشه‌بندی نوع داده‌هایی که به طور طبیعی دارای ساختاری سلسله‌مراتبی هستند مناسب‌تر است. برای مثال، در گونه‌بندی جانداران مختلف و مشخص‌کردن اجداد مشترک در فرایند تکامل این نوع خوشه‌بندی مناسب‌تر است.
- در خوشه‌بندی سلسله‌مراتبی، نتایج قابلیت بازتولید را دارند. این برخلاف k-means است که چون مراکز خوشه‌ها در ابتدا به طور رندم انتخاب می‌شود، نتایج ممکن است هر دفعه فرق کنند.

GMM -۴

(۴.۱

در این مثال خاص، بله. چرا که اولا دو خوشه داده با فاصله معقولی از هم جدا هستند و در هر خوشه نقاط با چگالی کمابیش یکسانی متمرکزند. به همین علت، هر دو الگوریتم k-means و GMM محتمل است که نتیجه یکسانی را برگردانند. البته لازم به ذکر است که خوشه‌بندی k-means، از نوع قطعی (hard-clustering) و خوشه‌بندی GMM از نوع احتمالاتی است. بنابراین، GMM برای هر خوشه، باز هم احتمالی را که دیتاپوینت‌های مجاور خوشه به آن تعلق داشته‌باشند در نظر می‌گیرد. اما به طور کل، خوشه‌بندی را در این مثال خاص می‌توان یکسان تلقی کرد.

(۴.۲

(۴.۲.۱

برای بروزرسانی میانگین توزیع‌های گاوسی، در مرحله M طبق فرمول زیر پیش می‌رویم:

$$u_j = \frac{\sum_{i=1}^{N} P\left(z_j^{(i)} = 1 \middle| x^{(i)}, u^{(i-1)}, \Sigma^{(i-1)}, \pi^{(i-1)}\right).\, x^{(i)}}{\sum_{i=1}^{N} P\left(z_j^{(i)} = 1 \middle| x^{(i)}, u^{(i-1)}, \Sigma^{(i-1)}, \pi^{(i-1)}\right)}$$

مطابق این فرمول، انتظار داریم که در مرحله Maximization، احتمال اینکه نقاط متعلق به توزیع آماری خوشه خود باشند بیشتر باشد. بنابراین، مطابق فرمول بالا، u_0 و u_1 جدید، میانگین وزن‌داری از نقاط روی صفحه هستند که وزن‌هایشان، با احتمال تعلق هر نقطه به توزیع مربوطه متناسب است تا احتمال اینکه نقطه‌ها، با پارامترهای تخمین‌زده‌شده مدل شوند بیشینه باشد. با این استدلال، انتظار می‌رود که در ایتریشن بعدی u_0 به سمت چپ و u_1 به سمت راست برود تا احتمال داده‌های کلاسترها با احتمال بالاتری به کلاستر خود تعلق یابند.

(۴.۲.۲

افزایش می‌یابد؛ چرا که هر ایتریشن از الگوریتم EM، احتمال (Likelihood) داده‌هایمان را یا بیشتر می‌کند، و یا در بدترین حالت، در مینیمم محلی گیر کرده‌ایم و likelihood افزایش نمی‌یابد. اما کاهشی را نخواهیم دید.