بسمه تعالی

پاسخ سری هشتم تمرینات درس یادگیری ماشین

امیرحسین رمضانی بناب (۹۹۲۱۰۲۹۴)

۱ سوال ۱

۱.۱ الف

ابتدا بدون کاسته‌شدن از کلیت، فرض می‌کنیم میانگین تمام ویژگی‌ها صفر باشد. اگر این ویژگی برقرار نبود می‌توانیم با کم کردن میانگین هر ستون از داده‌های آن ستون، میانگین را به صفر تبدیل کنیم. حال ماتریس کوایانس به شکل زیر تشکیل می‌شود:

$$C_{n\times n} = \frac{1}{m-1}\sum_{i=1}^{m}(x_i-\mu)(x_i-\mu)^T = \frac{1}{m-1}\sum_{i=1}^{m}(x_i)(x_i)^T = \frac{1}{m-1}X^TX$$

ماتریس به دست آمده متقارن است زیرا:

$$C^T = (\frac{1}{m-1}X^TX)^T = \frac{1}{m-1}X^TX = C$$

طبق قضیه‌ی *finite-dimensional spectral theorem* می‌توانیم هر ماتریس متقارنی را قطری‌سازی کنیم. پس می‌توان C را قطری‌سازی کرد. داریم:

$$C = V\Lambda V^T$$

که V ماتریسی است که ستون‌های آن، بردارهای ویژه‌ی C هستند و Λ ماتریسی قطری است که در قطر اصلی، مقادیر ویژه‌ی C قرار دارند. اعمال PCA روی X معادل این است که ماتریس X را در V ضرب کنیم. یعنی

$$X_P = XV$$

حال اگر SVD را روی ماتریس اعمال کنیم خواهیم داشت:

$$X = USV^T$$

که U یک ماتریس یکانی بوده و S یک ماتریس قطری از مقادیر تکین X می‌باشد. حال اگر C را مجددا محاسبه کنیم داریم:

$$\begin{aligned} C = \frac{1}{m-1}X^TX &= \frac{1}{m-1}(USV^T)^T(USV^T) \\ &= \frac{1}{m-1}VS^TU^TUSV^T \\ (U \textit{ is unitary}) \quad &= \frac{1}{m-1}VS^TSV^T \\ &= V\ \frac{S^TS}{m-1}\ V^T \end{aligned}$$

پس $\frac{S^TS}{m-1}$ تشکیل دهنده‌ی مقادیر ویژه‌ی ماتریس کواریانس C است و به عبارتی اگر عناصر قطر اصلی Λ را λ_i و S را s_i بنامیم داریم:

$$\lambda_i = \frac{s_i^2}{m-1}$$

حال اگر بخواهیم PCA را توسط SVD به دست آوریم با توجه به عمود بودن بردارهای ویژه (ستون‌های V) بر یکدیگر داریم:

$$X_P = XV = (USV^T)V = (US)(V^TV) = (US)(I) = US$$

همچنین اگر بخواهیم از SVD برای کاهش بعد از n به d استفاده کنیم. کافی است d ستون سمت چپ U و d سطر و ستون بالا و چپ S را در هم ضرب نماییم تا همان کاهش بعدی که در PCA نیاز داشتیم، از طریق SVD به دست آوریم.

۲.۱ ب

استفاده از SVD

اگر تعداد داده‌ها از تعداد ابعاد کمتر باشد $(m < n)$ ، در آن‌صورت برای محاسبه‌ی PCA از طریق بردارهای ویژه‌ی ماتریس کواریانس ابتدا باید خود ماتریس کواریانس را تشکیل دهیم. ابعاد این ماتریس $n \times n$ است. اگر ماتریسی $D \times D$ باشد، مرتبه‌ی زمانی محاسبه‌ی بردارهای ویژه‌ی آن، $O(D^3)$ خواهد بود. پس در اینجا مرتبه‌ی زمانی $O(n^3)$ است و به دلیل نیاز به دخیره‌ی کل ماتریس کواریانس در حافظه، مقدار بسیار زیادی حافظه مصرف خواهد کرد. از طرف دیگر مرتبه‌ی زمانی SVD طبق این مقاله برابر $O(mn\min(n,m))$ می‌باشد که وقتی $m \ll n$ برابر $O(m^2n)$ می‌شود. که به مراتب بهتر از روش قبلی است.

نکته : همانطور که در سوال دوم امتحان پایان‌ترم دیدیم، محاسبه‌ی PCA توسط بردار ویژه های ویژه‌ی ماتریس کواریانس می‌تواند از طریق بردارهای ویژه‌ی ماتریس XX^T نیز محاسبه شود که در این صورت مرتبه‌ی زمانی این روش به $O(m^3)$ کاهش خواهد یافت که وقتی $m \ll n$ به مراتب بهتر از محاسبه‌ی PCA از طریق SVD خواهد بود. ولی در حالت کلی همانطور که اشاره شد SVD نتیجه بهتری دارد.

۲ سوال ۲

۱.۲ آ

اگر تعداد داده‌ها N و تعداد خوشه‌ها K باشد، تعداد حالات تخصیص داده‌ها به خوشه‌ها برابر K^N است. از آن‌جایی که هم N و هم K متناهی هستند، پس تعداد این حالت‌ها محدود است. از آنجایی که مقادیر μ در هر مرحله به مرکز داده‌های هر خوشه تغییر پیدا می‌کنند، پس تعداد مقادیر مختلف بردار μ نیز متناهی است. الگوریتم K-Means در هر مرحله در صورتی مرکز خوشه‌ها را تغییر می‌دهد که تابع هزینه کاهش یابد. یعنی در طول الگوریتم تابع هزینه همواره نزولی است.
حال این فضای حالت را در نظر بگیرید. یعنی یک گراف که هر راس آن یک خوشه‌بندی را نمایش می‌دهد و تابع هزینه متناظر هر راس در اختیار ما است. پس K-Means متناظر با یک مسیر روی این گراف است. چون تابع هزینه نزولی است پس امکان ندارد در طی این مسیر تابع هزینه افزایش یابد و این یعنی دوری در مسیر نخواهیم داشت. پس چون در این مسیر دور نداریم و تعداد رئوس آن نیز متناهی است، در نهایت در یک راس متوقف خواهیم شد و این یعنی K-Means در نهایت همگرا می‌شود.

۲.۲ ب

ناحیه‌ای که داده‌ها چگالی کمتری دارند، A و ناحیه‌ای که داده‌ها چگالی بیشتری دارند را B می‌نامیم.
فرض کنیم در ابتدای امر مراکز خوشه‌ها به شکل تصادفی روی صفحه انتخاب می‌شوند. از آنجایی که نصف داده‌ها در ناحیه‌ی A و نصف دیگر در ناحیه‌ی B قرار دارند، پس کم‌بودن چگالی داده‌ها در ناحیه‌ی A به معنای بزرگ‌تر بودن ناحیه‌ی A می‌باشد. در نتیجه نقاطی که به عنوان مرکز خوشه در نظر گرفته می‌شوند با احتمال بزرگتری نزدیک و یا درون ناحیه‌ی A خواهند بود. بدین ترتیب با احتمال زیادی، داده‌های ناحیه A به تعداد بیشتری خوشه تعلق می‌گیرند و داده‌های ناحیه‌ی B به خوشه‌های کمتری تعلق خواهند یافت. با پیش رفتن الگوریتم kmeans همین روند حفظ می‌شود و مراکز خوشه ها به ناحیه‌ی A نزدیک‌تر می‌شوند. در نتیجه در انتهای کار، ناحیه‌ی A میزبان تعداد بیشتری خوشه خواهد بود و چگالی مراکز در A بیشتر از B می‌باشد.

۳.۲ پ

یکی از ایرادات بزرگ kmeans آن است که نسبت به نقاط اولیه‌ی مراکز خوشه‌ها حساس می‌باشد. اگر مراکز خوشه ها به خوبی انتخاب نشده باشند، ممکن است نتیجه الگوریتم خوب نبوده و در یک بهینه‌ی محلی نه‌چندان‌خوب گیر کنیم. روشی که ارائه شده است، سعی می‌کند این مشکل را برطرف سازد. در روش ارائه شده اگر یک نقطه، فاصله‌ی بیشتری از نزدیک‌ترین مرکز داشته باشد، شانس بیشتری را برای انتخاب شدن به عنوان مرکز جدید خواهد داشت. به این ترتیب، مراکز تا حد ممکن دور از یکدیگر انتخاب می‌شوند و احتمال اینکه مراکز اولیه در خوشه‌های مجزایی باشند، افزایش می‌یابد. البته دور بودن مراکز از همدیگر، احتمال انتخاب داده‌ی outlier به عنوان مرکز خوشه را افزایش می‌دهد که می‌تواند مشکل‌زا باشد. پس برای جمع‌بندی، اگر تعداد داده‌های outlier زیاد نباشد الگوریتم جدید به دلیل انتخاب نقاط دور از هم که احتمال بیشتری دارد که در حوشه‌های جداازهم باشند، مراکز اولیه را به خوبی مقداردهی می‌کند که باعث می‌شود الگوریتم kmeans در مراحل بعدی به خوشه‌های مناسبی همگرا شود.
از لحاظ همگرایی، از آنجایی که مراکز انتخاب شده با احتمال بیشتری در خوشه‌های مجزایی قرار می‌گیرند، فاصله‌ی که هر مرکز تا مکان خود در نقطه‌ی همگرایی خواهد داشت کاهش می‌یابد. یعنی مراکز، فاصله‌ی کمتری را از شروع تا همگرایی الگوریتم طی می‌کنند. بدین ترتیب الگوریتم سریعتر همگرا می شود.
البته همانطور که گفته شد، مطالبی که در بالا آمده است، در حالت کلی است و اگر تعداد زیادی داده‌ی outlier داشته باشیم، ممکن است عملکرد الگوریتم ارائه شده، تحت تاثیر قرار گیرد.

۳ سوال ۳

۱.۳ آ

فرض کنیم از Q-Learning استفاده می‌کنیم. همچنین مجموعه‌ی اعمال را $\{U, D, L, R\}$ در نظر می‌گیریم. در اینصورت بروزرسانی Q-Value ها به شکل زیر خواهد بود.

$$Q(s,a) = Q(s,a) + \alpha(r(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

پس در اپیزود اول خواهیم داشت:

$$\begin{aligned} Q(s_{11},R) &= Q(s_{11},R) + \alpha(r(s_{11},R,s_{12}) + \gamma \max_{a'} Q(s_{12},a') - Q(s_{11},R)) \\ &= 0.5 + \alpha(0 + 0.5\gamma - 0.5) \\ &= 0.5 + 0.5\alpha\gamma - 0.5\alpha \end{aligned}$$

و سپس

$$\begin{aligned} Q(s_{12},R) &= Q(s_{12},R) + \alpha(r(s_{12},R,s_{13}) + \gamma \max_{a'} Q(s_{13},a') - Q(s_{12},R)) \\ &= 0.5 + \alpha(-1 + 0 - 0.5) \\ &= 0.5 - 1.5\alpha \end{aligned}$$

و در اپیزود بعدی:

$$\begin{aligned} Q(s_{11},R) &= Q(s_{11},R) + \alpha(r(s_{11},R,s_{12}) + \gamma \max_{a'} Q(s_{12},a') - Q(s_{11},R)) \\ &= (0.5 + 0.5\alpha\gamma - 0.5\alpha) + \alpha(0 + 0.5\gamma - (0.5 + 0.5\alpha\gamma - 0.5\alpha)) \\ &= 0.5 + \alpha\gamma - \alpha - 0.5\alpha^2\gamma + 0.5\alpha^2 \end{aligned}$$

و سپس

$$\begin{aligned} Q(s_{12},D) &= Q(s_{12},D) + \alpha(r(s_{12},D,s_{22}) + \gamma \max_{a'} Q(s_{22},a') - Q(s_{12},D)) \\ &= 0.5 + \alpha(0 + 0.5\gamma - 0.5) \\ &= 0.5 + 0.5\alpha\gamma - 0.5\alpha \end{aligned}$$

و سپس

$$\begin{aligned} Q(s_{22},D) &= Q(s_{22},D) + \alpha(r(s_{22},D,s_{32}) + \gamma \max_{a'} Q(s_{32},a') - Q(s_{22},D)) \\ &= 0.5 + \alpha(0 + 0.5\gamma - 0.5) \\ &= 0.5 + 0.5\alpha\gamma - 0.5\alpha \end{aligned}$$

و

$$\begin{aligned} Q(s_{32},R) &= Q(s_{32},R) + \alpha(r(s_{32},R,s_{33}) + \gamma \max_{a'} Q(s_{33},a') - Q(s_{32},R)) \\ &= 0.5 + \alpha(1 + 0 - 0.5) \\ &= 0.5 + 0.5\alpha \end{aligned}$$

۲.۳ ب

می‌دانیم الگوریتم Q-Learning اگر به تعداد مراحل بسیار زیادی اجرا شود، پاسخ به دست‌آمده از آن به پاسخ بهینه همگرا می‌شود. اگر پاسخ بهینه را در نظر بگیریم، با توجه به معادلات بهینه‌ی Bellman داریم:

$$Q^*(s,a) = \sum_{s' \in S} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')]$$

حال به دلیل اینکه اقدامات عامل، قطعی بوده و نتیجه هر عمل از قبل مشخص است، داریم

$$Q^*(s,a) = r(s,a,s') + \gamma \max_{a'} Q^*(s',a')$$

اولا توجه شود که دور زدن در مسیر، در حالت بهینه چیزی به عامل اضافه نمی‌کند. همچنین فرض می‌کنیم که عامل پس از رسیدن به هدف حرکت خود را ادامه نمی‌دهد. در این شرایط می‌توانیم از معادله بالا استفاده کنیم و جدول Q-value را شکل دهیم.

State	*U*	*D*	*L*	*R*
s_{11}	$\times$	γ^3	$\times$	γ^3
s_{12}	$\times$	γ^2	$-$	-1
s_{13}	$\times$	0	0	$\times$
s_{21}	0	-1	$\times$	γ^2
s_{22}	$-$	γ	$-$	γ
s_{23}	-1	$+1$	$-$	$\times$
s_{31}	0	$\times$	$\times$	0
s_{32}	$-$	$\times$	-1	$+1$
s_{33}	0	$\times$	0	$\times$

پس در حالت بهینه :

$$Q(s_{11}, right) = \gamma^3$$

پس

$$Q_1(s_{11}, right) = \gamma_1^3$$

و

$$Q_2(s_{11}, right) = \gamma_2^3$$

در نتیجه

$$\frac{\gamma_1^3}{\gamma_2^3} = \frac{9}{7} \Rightarrow \frac{\gamma_1}{\gamma_2} = 1.08$$

پس حتی اگر به حالت بهینه نیز همگرا نشده باشیم و در یک بهینه محلی باشیم رابطه‌ی $\gamma_1 > \gamma_2$ برقرار است.

۳.۳ پ

در حالت model-free که عامل از تجربیات خود استفاده می‌کند، تعداد پارامترهای موجود در هر خانه‌ی ماتریس برابر تعداد اعمال مجاز در آن خانه است. به شکل زیر:

2	3	0
3	4	3
0	3	0

پس در مجموع ۱۸ پارامتر برای یادگیری داریم.
در حالت model-based به تابع انتقال و پاداش دسترسی نداریم . عامل درون محیط می‌رود و سعی می‌کند توزیع احتمال $p(s'|s,a)$ و تابع پاداش $r(s,a,s')$ را یاد بگیرد. فرض کنیم اگر عامل وارد خانه‌ی هدف شد، همانجا متوقف می‌شود. همچنین از آنجایی که حرفی از دیوار به میان نیامده است، فرض می‌کنیم عمل‌هایی که منجر به برخورد با دیوار می‌شوند، جزو مجموعه‌ی عمل‌های مجاز عامل نباشند. به دلیل وجود ابهام، دو حالت را در نظر می‌گیریم:

- از آنجایی که اعمال قطعی هستند، فرض می‌کنیم اگر در یک خانه، یک عمل انجام دهیم فقط به یک خانه‌ی دیگر می‌رویم و احتمال رفتن به سایر خانه‌ها لحاظ نمی‌شود. در این‌صورت (عامل قطعی بودن محیط را بداند)، تعداد پارامترهای روش model-based دو برابر حالت model-free است. زیرا به ازای هر انتقال، یک پاداش نیز داریم. پس در کل ۳۶ پارامتر خواهیم داشت.

- در این حالت فرض می‌کنیم، عامل نمی‌داند که محیط قطعی است. در اینصورت باید تمام $p(s,a,s')$ های ممکن را مدل کند. البته به دلیل اینکه برخی از $p(s,a,s')$ها احتمال صفر می‌گیرند، $r(s,a,s')$ برایشان وجود نخواهد داشت. در این حالت، تعداد پارامترهای مربوط به p که باید یادگرفته شوند، برابر است با:

$$\sum_s |A(s)| * |H(s)| = 2*2+3*3+3*3+3*3+3*3+4*4 = 56$$

که $A(s)$ مجموعه‌ی اعمال مجاز در s و $H(s)$ مجموعه‌ی s' هایی است که با انجام یک عمل از s به آن‌ها می‌رسیم. همچنین تعداد $r(s,a,s')$ها در این حالت مثل قبل برابر ۱۸ است. پس در کل ۷۴ پارامتر خواهیم داشت.

توجه شود که تمام مطالب گفته شده با فرض‌هایی انجام شد که آمده‌اند. به عنوان مثال می‌توانستیم فرض کنیم انتقال‌هایی که موجب خروج از مربع می‌شوند و اصطلاحا به دیوار برخورد می‌کنند در تابع انتقال لحاظ می‌شوند ولی احتمال آن‌ها صفر در نظر گرفته می‌شود. یا اینکه در مورد تابع پاداش، شرط ساده‌سازی داشته باشیم و ... . از آنجایی که در صورت سوال در مورد چنین فرض هایی صحبت نشده است، دو فرض در نظر گرفته شد و مقادیر خواسته شده محاسبه شد.

همچنین منظور طراح سوال از بهینه‌بودن روش مشخص نمی‌باشد. اگر تعداد پارامترهای مدل مدنظر باشد، Q-Learning بهینه‌تر می‌باشد. توجه شود که Q-Learning نیاز به تعداد قدم‌های بسیار زیادی دارد تا مقدار Q-Value ها به مقدار بهینه همگرا شوند . اما الگوریتم های Model-free به دلیل ماهیت قطعی محیط می‌توانند در محیط حرکت کرده و توزیع‌های مربوطه را بیابند و سپس با استفاده از Planning به پاسخ بهینه‌ی سراسری برسند.