دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

پروژه درس هوش مصنوعی | فاز اول

مروری بر روش‌های مبتنی بر HMM

اساتید درس: دکتر مهدیه سلیمانی - دکتر محمدحسین رهبان

تهیه شده توسط: محمد مشتاقی‌فر

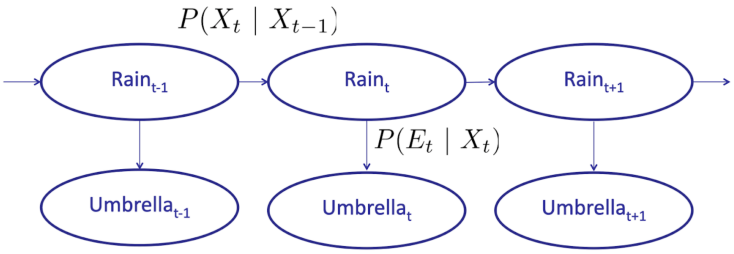
مسئول پروژه: امیرحسین رازلیقی

طراحان فاز اول پروژه: امیرحسین رازلیقی - حمیدرضا یعقوبی - علیرضا حیدری - رضا وحیدی مجد - پرهام رضایی

یادآوری HMM

در درس با شبکه‌های بیزی برای مدل‌سازی وابستگی بین تعدادی اتفاق و احتمال‌های شرطی وقوع آن‌ها با استفاده از یک گراف جهت‌دار، آشنا شدید. Hidden Markov Model s یا به اختصار HMM حالت خاصی از شبکه‌های بیزی هستند که برای وقایعی به کار می‌روند که به نمونه از برخی متغیرهای تصادفی مدل‌سازی شده دسترسی نداریم. در واقع به این معنا است که اطلاعاتی از یکسری رئوس مدلمان در دسترس نیست و صرفا درباره ارتباطشان (همان جدول احتمالات) اطلاعات داریم.

برای مثال فرض کنید ما در اتاقی بدون پنجره زندگی می‌کنیم و توانایی مشاهده محیط بیرون از اتاق را نداریم. حال می‌خواهیم در هر روز احتمال بارش باران در آن روز و روزهای بعد را بدست بیاوریم. تنها داده‌ای که ما داریم احتمال باریدن باران در روز iام به شرط بارش یا عدم بارش باران در روز $i-1$ام است. طبیعتا تا اینجا این کار غیرممکن به نظر می‌رسد. اما فرض کنید که شخصی هر روز از جلوی اتاق ما رد می‌شود و ما می‌توانیم او را ببینیم. با هربار مشاهده، می‌توانیم از روی چتر داشتن یا نداشتن این مرد بدانیم که چقدر احتمال دارد باران در حال باریدن باشد. ما می‌توانیم این اتفاق را با استفاده از شبکه‌های بیزی مدل کنیم:



همچنین برای این مدل دو جدول نیز نیاز داریم تا احتمال‌های شرطی را برای ما مشخص کنند. برای مثال این دو جدول را در نظر بگیرید:

R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

به مدل طراحی شده در بالا یک HMM می‌گوییم. علت این قضیه هم این است که ما اطلاعات قطعی‌ای از راس‌های Rain نداریم و فقط می‌توانیم با استفاده از evidenceهایی که از طریق راس‌های Umbrella کسب می‌کنیم، احتمال بارش باران را بدست بیاوریم. اکنون که با نحوه مدل‌سازی در HMM ها آشنا شدیم، مانده است نحوه محاسبه مقادیر دلخواهمان. همانطور که در مثال بالا دیدید یکی از چیزهایی که علاقه داشتیم بدست بیاوریم، احتمال باریدن باران در روز tام به شرط اینکه در روزهای $1, 2, \ldots, t$ بدانیم که مرد همسایه ما چتر داشته یا خیر. به همین منظور تابع زیر را تعریف می‌کنیم:

$$B(X_t) = P(X_t \mid e_{1:t})$$

که دقیقا به معنای همان چیزی است که در بالا تعریف کردیم. احتمال وقوع اتفاق مورد نظرمان در لحظه tام به شرط دیدن شواهد تا آن لحظه را برابر با تابع $B(X_t)$ تعریف کردیم. روش‌هایی برای بدست آوردن مقدار این تابع وجود دارد. یکی از ابتدایی‌ترین آن‌ها، استفاده از Forward Algorithm است.

در این الگوریتم با استفاده از یک رابطه بازگشتی، می‌توانیم مقدار تابع را در لحظه t محاسبه کنم. رابطه مورد نظر را می‌توانید در اینجا مشاهده کنید:

$$\begin{aligned} P(x_t \mid e_{1:t}) &\propto_X P(x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t \mid x_{t-1}) P(e_t \mid x_t) \\ &= P(e_t \mid x_t \sum_{t-1} P(x_{t-1}, e_{1:t-1}) P(x_t \mid x_{t-1}) \end{aligned}$$

با این روش می‌توانیم احتمالات مورد نظرمان را بدست بیاوریم. اما مشکل کجاست؟
واقعیت این است که در دنیای واقعی مثال‌های زیادی از استفاده HMM داریم و اکثر آن‌ها با الگوریتم بالا کنار نمی‌آیند. دلیل اصلی این اتفاق، حجم بالای محاسبات در این الگوریتم است. اینجا چند مورد از کاربردهای HMM را مشاهده می‌کنید. در هر کدام از آن‌ها استدلال کنید که چرا استفاده از روش بالا انتخاب خوبی نیست:

- تشخیص صدا
- ترجمه خودکار متن
- Robot Tracking
- Localization

در ادامه قصد داریم یکی از این کاربردها را با یک مثال، دقیق‌تر بررسی کنیم.

بررسی مسئله Localization

ابتدا این مسئله را توصیف می‌کنیم. فرض کنید رباتی در صفحه در حال حرکت است و هدف ما پیدا کردن این ربات است. شاید بگویید خب با نگاه کردن به صفحه آن را به راحتی پیدا می‌کنیم. اما نه! حالتی را فرض کنید که ربات نامرئی باشد. در این حالت ما برای اینکه بتوانیم ربات را پیدا کنیم، در برخی خانه‌ها سنسور تشخیص حرکتی قرار می‌دهیم که در هر لحظه به ما یک احتمال برمی‌گردانند. این عدد برابر با احتمال حضور ربات در شعاع کمتر از r از آن سنسور است.

حال ما می‌خواهیم با استفاده از دیتاهایی که از سنسورها دریافت می‌کنیم، محل ربات را پیدا کنیم. کمی به مدل‌سازی این مسئله با استفاده از HMM فکر کنید. همانطور که قبلا دیده بودید، در HMM دو نوع State داشتیم (Hidden State, Evidence). برای اینکه ببینیم هرکدام

از این‌ها متناظر با چه دیتایی در مسئله ما می‌شوند، باید ببینیم کدام دیتاها را مسئله به ما می‌دهد)(Evidence و چه چیزی از ما می‌خواهد که دیتای دقیقی از آن در دسترس نیست (Hidden State). یک مدل‌سازی سازگار با شرایط بالا برای این مسئله به این صورت است که Hidden State در لحظه t را برابر با موقعیت دقیق ربات در لحظه t بنامیم و Evidence در لحظه t را هم برابر با احتمال‌هایی که رادارها در لحظه t خروجی می‌دهند بگیریم. بنابر این مدل‌سازی مسئله از ما $P(x_t \mid s_{1:t})$ را می‌خواهد که $s_{1:t}$ برابر با احتمال‌های خروجی سنسورها تا لحظه tام و x_t برابر با مکان ربات در لحظه tام است. توجه کنید که گره‌های Evidence ممکن است اطلاعات نویزی درمورد گره‌های مخفی به ما بدهند.

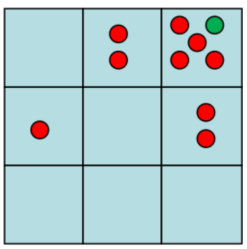
در درس سه روش برای محاسبه این احتمال دیدیم. یکی روش استفاده از رابطه بازگشتی برای محاسبه B_t ه دقیقا احتمال بالا است و نحوه محاسبه آن را در مثال باران و چتر دیدید. یکی استفاده از Particle Filtering برای محاسبه تقریبی احتمال خواسته شده با استفاده از نمونه گیری (particle) و حرکت دادن آن‌ها. آخری هم استفاده از viterbi که یک روش مبتنی بر Dynamic Programming برای محاسبه محتمل‌ترین مسیر حرکت ربات ما است.

روش اول را در این مسئله تا زمانی می‌توانیم انجام دهیم که تعداد سنسورها و اندازه نقشه کوچک باشد و اگر از حدی بزرگتر شود این روش به علت حجم محاسبات بسیار زیاد نمی‌تواند عملی باشد. از آنجا که عموما در مسائلی که از HMM در آن‌ها استفاده می‌شود این مشکل وجود دارد، خوب است به سراغ دو روش دیگر برویم.

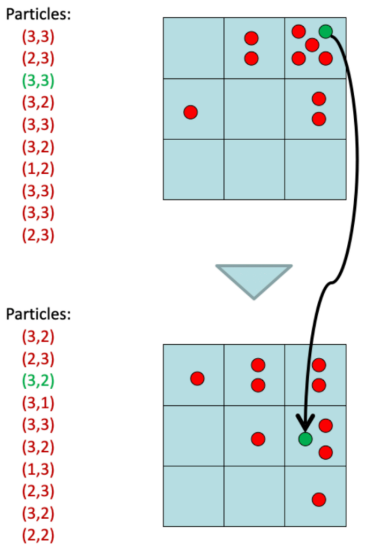
استفاده از الگوریتم Particle Filtering

اکنون که روش اول را رد کردیم، به سراغ روش Particle Filtering می‌رویم تا ببینیم این روش چه گلی بر سر ما می‌زند. در این روش تنها نیاز ما احتمال وجود ربات در خانه (i, j) در لحظه t است. این احتمال را نیاز داریم تا بتوانیم particleهایمان را نمونه‌گیری کنیم و به احتمال $P(x_t \mid x_{t+1})$ هم نیاز داریم تا بتوانیم particleهایمان را در مرحله Resample حرکت دهیم. این احتمال را می‌توانیم با یک توزیع uniform تخمین بزنیم. یعنی اگر خانه x_{t-1} در نقشه ما 3 همسایه دارد، احتمال رفتن ربات به همسایه‌های آن را $\frac{1}{3}$ و احتمال رفتن ربات به خانه‌های دیگر را 0 بگیریم. در دنیای HMM و کلا هوش مصنوعی، همواره با تخمین زدن اتفاق‌ها با توزیع‌های احتمالاتی مختلف روبرو هستیم. بلکه شما هم نیاز داشته باشید احتمالات زیادی را با توزیع‌های مختلف تخمین بزنید. در بخش بعدی نحوه اجرای این الگوریتم را می‌توانید ببینید.

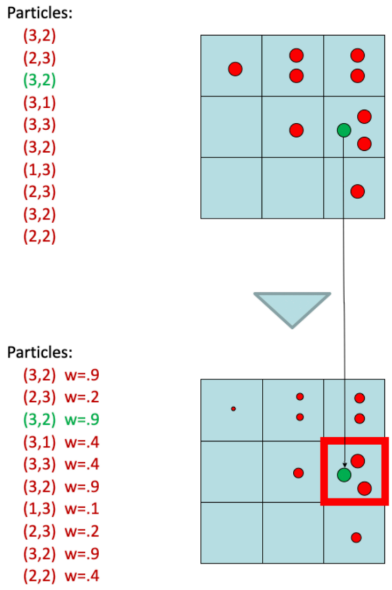
در این روش ما می‌خواهیم با استفاده از نمونه‌گیری (مانند کاری که در شبکه‌های بیزی انجام می‌دادیم.) احتمالاتی که می‌خواهیم را محاسبه کنیم. با ذکر یک مثال به سراغ این روش می‌رویم. فرض کنید مسئله Localization را برای یک ربات در یک جدول 3×3 می‌خواهیم انجام دهیم. همانطور که در بالا اشاره شد، در هر لحظه نمی‌توانیم با اطمینان مکان ربات را پیدا کنیم و صرفا داده‌های سنسورها را در اختیار داریم. فرض کنید در مرحله اول با استفاده از احتمالاتی که سنسورها در اختیارمان گذاشته‌اند یک توزیع احتمالاتی برای مکان اولیه ربات بدست بیاوریم و از این توزیع تعدادی نمونه بگیریم. شکل زیر یک مثال از نمونه‌های گرفته شده است:



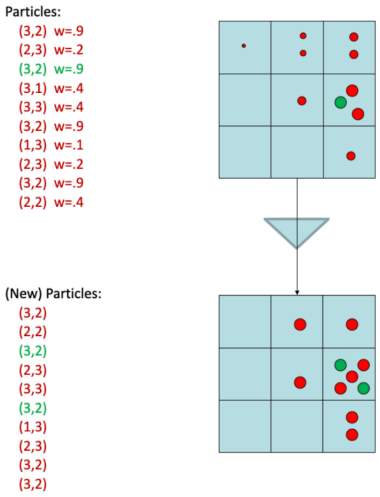
حال می‌خواهیم این نمونه‌ها را، که هرکدام را یک particle می‌نامیم، حرکت دهیم. علت اینکار این است که این particleهای را نماینده ربات فرض می‌کنیم و می‌خواهیم با تعداد زیادی حرکت بتوانیم نمونه‌هایمان را به یک مکان همگرا کنیم. حرکت دادن این نمونه‌ها هم با استفاده از جدول‌های احتمالات شرطی‌ای که داریم قابل انجام است؛ به این مرحله از اجرای الگوریتم Elapse Time گفته می‌شود:



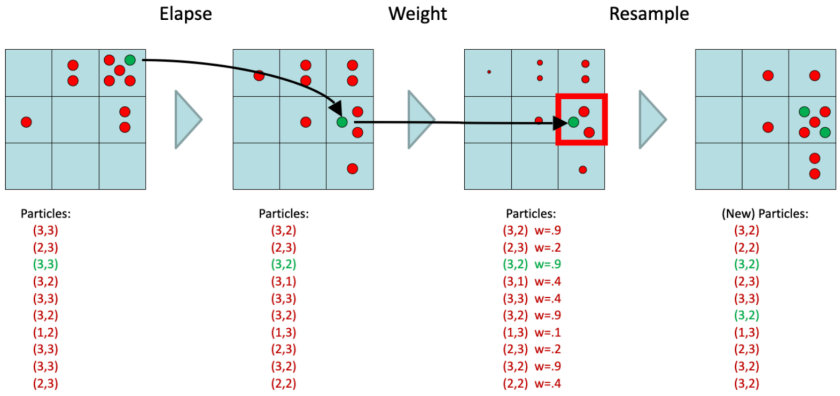
اما تا اینجا ما داده‌هایی که از سنسور در لحظه $t = 2$ دریافت کرده‌ایم را اعمال نکردیم. برای انجام این کار در این مرحله به هر داده وزنی متناسب با Evidence موجود در آن لحظه نسبت می‌دهیم. در مثال ما وزن هر particle برابر با ضرب احتمالاتی است که سنسورها به خانه‌ای که particle در آن قرار دارد می‌دهند.



در نهایت باید اثر این وزن‌ها را از بین ببریم. زیرا اگر وزن‌ها باقی بمانند، چون اعدادی بین $0, 1$ هستند و در هر مرحله از انجام الگوریتم در هم ضرب می‌شوند، در نهایت اعداد بسیار کوچکی ساخته می‌شود که مطلوب ما نیست. برای همین تاثیر وزن‌ها را می‌خواهیم در همینجا با استفاده از Resampling از بین ببریم. بدین صورت که از particleها با توزیع وزن‌هایی که در مرحله قبل برایشان بدست آوردیم نمونه‌گیری می‌کنیم:



و اینجا پایان یک مرحله از اجرای Particle Filtering است. برای ادامه particle های خروجی را دوباره وارد مرحله وزن‌دهی کرده و Resample می‌کنیم. اینکار را به اندازه‌ای که Evidence داریم انجام می‌دهیم و در نهایت از روی particle های نهایی، می‌توانیم احتمال حضور ربات در هر خانه را حساب کنیم.



استفاده از الگوریتم Viterbi

اما اگر Viterbi را به عنوان روش حل انتخاب کنیم هدفمان به چه صورت می‌شود؟ در این الگوریتم احتمالی که قصد داریم محاسبه کنیم با دو الگوریتم قبل فرق می‌کند. در دو الگوریتم قبل، هدفمان محاسبه $P(x_t \mid s_{1:t})$ بود تا بتوانیم با توجه به Evidenceهایی که در این t لحظه جمع کردیم، محتمل‌ترین مختصات برای ربات را پیدا کنیم. اما در روش Viterbi هدف حساب کردن تابعی به نام $m_t[x_t]$ است که به صورت زیر تعریف می‌شود:

$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, s_{1:t}) = P(s_t \mid x_t) \max_{x_{t-1}} P(x_t \mid x_{t-1}) m_{t-1}[x_{t-1}]$$

شهود این رابطه از این می‌آید که فرض می‌کنیم تابع $m_t[x_t]$ برابر محتمل‌ترین مسیری است که ربات طی کرده تا به خانه x_t رسیده باشد. حال می‌خواهیم این تابع را بر اساس مقدارش در x_{t-1} بدست بیاوریم. اگر رابطه بالا را بازبینی کنید، متوجه می‌شوید که این دقیقا چیزی است که

سعی دارد حساب کند. برای استفاده از این الگوریتم علاوه بر تخمینی که در Particle Filtering برای احتمال $P(x_t \mid x_{t-1})$ زدیم، به یک تخمین دیگر نیز نیاز داریم. ما برای استفاده از رابطه بالا، باید بتوانیم احتمال $P(s_t \mid x_t)$ را محاسبه کنیم و چون آن را به صورت دقیق در اختیار نداریم، نیاز به تخمین مناسبی از توزیع آن داریم. به توزیع‌هایی که برای تخمین این احتمال مناسب هستند فکر کنید. به نظر شما لفظ "**مناسب بودن**" یک توزیع به چه معناست؟

برای تخمین این احتمال در ابتدا باید ببینیم که چه نیازمندی‌هایی دارد. برای مثال می‌دانیم که احتمالا هرچقدر سنسورها از x_t دورتر باشند، این احتمال هم باید کوچکتر شود. پس خوب است جایی در تخمینمان، مقدار فاصله x_t تا سنسورها را وارد کنیم. برای نکته دوم می‌توانیم فرض کنیم احتمال‌هایی که سنسورها خروجی می‌دهند، مستقل از یکدیگر است. این فرض را برای این اضافه می‌کنیم تا بتوانیم $P(s_t \mid x_t)$ را به صورت زیر بنویسیم:

$$P(s_t \mid x_t) = \prod_i P(s_{i,t} \mid x_t)$$

که $s_{i,t}$ برابر با داده‌های سنسور iام در لحظه t است. اکنون که توانستیم سنسورها را از هم جدا کنیم، می‌توانیم برای هر سنسور یک توزیع نمایی به صورت زیر در نظر بگیریم:

$$P(d_{i,t}) = \frac{1}{\beta} e^{\frac{-d_{i,t}}{\beta}}$$

که در آن $d_{i,t}$ برابر با فاصله سنسور iام تا نقطه x_t است. بررسی کنید که آیا این توزیع خواسته گفته شده در بند قبل را دارد؟