دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

هوش مصنوعی

بهار ۱۴۰۰

استاد: محمدحسین رهبان
گردآورندگان: کورش شریعت، علیرضا توکلی

تمرین سوم — **شبکه‌های بیزی و زنجیره مارکوف** — مهلت ارسال: ۲۱ اردیبهشت

- مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روز مشخص‌شده است.
- هم‌کاری و هم‌فکری شما در انجام تمرین مانعی ندارد اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد.
- در صورت هم‌فکری و یا استفاده از هر منابع خارج درسی، نام هم‌فکران و آدرس منابع مورد استفاده‌برای حل سوال مورد نظر را ذکرکنید.
- لطفا تصویری واضح از پاسخ سوالات نظری بارگذاری کنید. در غیر این صورت پاسخ شما تصحیح نخواهد شد.

سوالات نظری (۱۰۰ نمره)

۱. (۵۰ نمره) به جوپیتر رجوع کنید.

۲. (۵۰ نمره) یکی از چالش‌های هوش مصنوعی در سال‌های اخیر مسئله اتوموبیل‌های خودران بوده‌است. در این سوال قصد داریم با استفاده از مدل‌های پنهان مارکوف (HMM) موقعیت اتوموبیل‌ها را پیش‌بینی کرده و با استفاده از آن از برخورد جلوگیری کنیم.

تمامی فایل‌های پروژه در یک پوشه زیپ با عنوان `car.zip` در اختیار شما قرار گرفته است. لطفا توجه داشته باشید که توابعی که شما باید تغییر دهید همگی در فایل `submission.py` قرار دارند و نیازی به تغییر ساختار پروژه نیست.

در ابتدا برای آشنایی با محیط مسئله می‌توانید با استفاده از **پایتون ۲** فایل `drive.py` را به صورت زیر اجرا کرده و خودتان اتوموبیل را دستی با کلیدهای $wasd$ حرکت دهید:

```
python drive.py -l lombard -i none
```

همانطور که احتمالا متوجه شدید، بدون دانستن موقعیت دیگر اتوموبیل‌ها رساندن اتوموبیل خود به مقصد بسیار دشوار است. بنابراین با استفاده از HMM سعی در پیش‌بینی موقعیت دیگر اتوموبیل‌ها خواهیم داشت. در این مسئله فرض می‌شود عامل (agent) ما مجهز به یک حسگر سونار است که در لحظه t یک فاصله تقریبی از موقعیت خود (a_t) تا اتوموبیل (یا اتوموبیل‌های) مقابل (c_t) گزارش می‌کند. فاصله گزارش شده (d_t) از یک توزیع نرمال با میانگین برابر با فاصله واقعی پیروی می‌کند:

$$D_t \sim \mathcal{N}\left(\|a_t - C_t\|, \sigma^2\right)$$

که در آن انحراف معیار (σ) نشان‌دهنده خطای حسگر است و مقدار آن در فایل تمرین در متغیر `Const.SONAR_STD` ذخیره‌شده است. فضای مسئله و موقعیت‌ها نیز همگی دو بعدی هستند ($c, a \in \mathbb{R}^2$). بنابراین وظیفه شما یافتن توزیع $\mathbb{P}(C_t | D_1 = d_1, \cdots, D_t = d_t)$ و پیش‌بینی $\mathbb{P}(C_{t+1} | D_1 = d_1, \cdots, D_t = d_t)$ است. برای ساده‌سازی مسئله، محیط را با تقسیم به تعدادی ستون و ردیف گسسته کرده و احتمالات را برای کدام از این خانه‌ها محاسبه می‌کنیم.

(آ) در بخش اول فرض می‌کنیم که اتوموبیل‌های مقابل ساکنند ($c_t = c_{t-1}$). شما می‌بایست تابع `observe` متعلق به کلاس `ExactInference` را طوری کامل کنید که با ورودی گرفتن موقعیت کنونی agent و فاصله تقریبی، احتمال حضور اتوموبیل را به‌روز رسانی کند. به عبارتی:

$$\overbrace{\mathbb{P}\left(C_t | D_1 = d_1, \cdots, D_t = d_t\right)}^{\text{posterior}} \propto \overbrace{\mathbb{P}\left(C_t | D_1 = d_1, \cdots, D_t = d_{t-1}\right)}^{\text{prior}} p\left(d_t | c_t\right)$$

دقت کنید که اطلاعات توزیع در کلاس `Belief` وجود دارند. به طور مثال برای دسترسی به prior می‌توانید از دستور `self.belief.getProb(i, j)` استفاده کنید که در آن `i` و `j` شماره ردیف و ستون هستند. تعداد کل ردیف‌ها و ستون‌ها نیز در همین کلاس موجود است. سپس پس از تعیین posterior می‌توانید با تابع `self.belief.setProb(i, j, p)` آن را در کلاس ذخیره کنید.
مطالعه فایل `util.py` که شامل این کلاس و دیگر متدهای کاربردی است، توصیه می‌شود. در پیاده‌سازی این سوال به‌ازای هر اتوموبیل مقابل یک instance از کلاس `ExactInference` ایجاد شده‌است. بنابراین در تابع این بخش و بخش بعدی فرض کنید ورودی توابع تنها مربوط به فاصله agent با یکی از اتوموبیل‌های مقابل است.
در نهایت می‌توانید عملکرد عامل خود را با اجرای فایل `driver.py` با استفاده از **پایتون ۲** به صورت زیر ملاحظه کنید:

```
python drive.py -a -p -d -k 1 -i exactInference
```

برای آشنایی بیشتر با فلگ‌های استفاده‌شده و تغییر آنها توضیحات فایل `index.html` را مطالعه کنید.

(ب) در این بخش حرکت اتوموبیل‌های مقابل نیز در نظر گرفته می‌شود. این حرکت دارای یک trainsition probability $p(c_{t+1}|c_t)$ است که از آن مطلع هستیم. شما باید تابع `elapseTime` را کامل کنید که ورودی نمی‌گیرد و احتمالات کلاس `self.belief` را با استفاده از trainsition probability به صورت زیر به‌روز رسانی می‌کند:

$$\mathbb{P}\left(C_{t+1} = c_{t+1} | D_1 = d_1, \cdots, D_t = d_t\right) \propto \sum_{c_t} \mathbb{P}\left(C_t = c_t | D_1 = d_1, \cdots, D_t = d_t\right) p(c_{t+1}|c_t)$$

این trainsition probability در قالب یک دیکشنری پایتون به نام `self.transProb` ذخیره شده‌است که کلیدهای آن یک تاپل متشکل از شماره ردیف و ستون جدید و قدیم است. به طور مثال `self.transProb[((0, 0), (0 ,1))]` (در صورت وجود) احتمال انتقال از ردیف صفر و ستون صفر به ردیف صفر و ستون یک را نشان می‌دهد. در نهایت می‌توانید عملکرد عامل خود را با دستور زیر مشاهده کنید:

```
python drive.py -a -d -k 1 -i exactInference
```

در صورت علاقه می‌توانید با تغییر فلگ‌های ورودی کد، شرایط مختلفی مانند تعداد بیشتر ماشین یا نقشه‌ی متفاوت را امتحان کنید.

در نهایت برای نمره‌دهی فایل `grader.py` با استفاده از **پایتون ۳** اجرا می‌شود. از آنجایی که سوال بخش‌های دیگری نیز دارد، نمره‌ی خروجی `grader` نمره کامل نخواهد بود و نمره شما پس از تصحیح تعیین می‌شود. **برای این سوال تنها آپلود فایل `submission.py` کافیست.**