به نام خدا

پروژه دوم: دریاچه یخی

درس: مبانی هوش مصنوعی

اعضا:

علی پورقیصری

طاها داوری

محمد امین مولوی زاده

توضیحات:

این کد پیاده سازی **(Q-Learning)** برای حل این مسئله است. در ابتدا کتابخانه های مورد نیاز را اضافه می کنیم. حال از کتابخانه **gymnasium** یک محیط تصادفی انتخاب می کنیم که با استفاده از یک سری پارامتر ها تعریف می شود. در حین تعریف محیط تصادفی، لغزندگی را در نظر میگیریم که باعث احتمال انتخاب یک سوم از خانه های رو به رو چپ یا راست عامل می شود. اگر پارامتر لغزندگی را در نظر نگیریم می بینیم که تصمیمات عامل همان طوری که می خواهد اتفاق می افتد. پس از پیاده سازی محیط، پارامتر های مربوطه را هم تنظیم می کنیم که به شرح زیر است.

تعداد مراحل بازی(**max_iter_number**)، ضریب یادگیری (**alpha**)، ضریب تخفیف (**gamma**)، اپسیلون (**epsilon**)، پاداش چاله(**hole_reward**)، پاداش هدف(**goal_reward**)، پاداش خانه های خالی(**state_reward**).

سپس ماتریس **QTable** را با استفاده از **numpy** و با مقادیر پیشفرض صفر پیاده سازی می کنیم که تعداد سطر های آن، تعداد خانه های محیط و تعداد ستون های آن به تعداد کنش های موجود در آن خانه (که در اینجا ۴ است) می باشد. در ادامه ارایه تک بعدی از نوع **numpy** تعریف می کنیم که برای ذخیره ارزش هر خانه است. بلافاصله بعد از تعریف کردن این ماتریس، ارزش های اولیه هر خانه را به دست می آوریم.

پروژه دو فاز دارد. فاز اول که فاز یادگیری عامل است و فاز دوم که فاز تست عامل می باشد.

فاز اول(فاز یادگیری):

در این فاز بازی به تعداد مراحل بازی(**max_iter_number**) اجرا می شود. از الگوریتم **Q-learning** در این بخش استفاده شده که شامل توابع زیر است:

تابع **can_move_to_next_state** :

این تابع محیط را ببرسی می کند و حالت هایی که امکان حرکت با کنش انتخاب شده را دارد از حالت هایی که امکان حرکت با کنش انتخاب شده را ندارد، متمایز می سازد. این تابع برای بهره وری بیشتر کد و سرعت در انتخاب کنش موفق پیاده سازی شده و بدون این تابع هم امکان انتخاب کنش وجود دارد.

تابع **choose_action**:

این تابع برای انتخاب کنش سودمند پیاده سازی شده به این صورت که ابتدا یک لیست خالی برای اندیس کنش ها در نظر گرفته شده سپس تمام کنش ها (در اینجا ۴ کنش) به لیست اضافه شده و به صورت نزولی مرتب می

شود. سپس ارزش ها گروه بندی می شوند و کنش هایی که ارزش یکسان دارند با هم جا به جا شوند که این امر باعث می شود هر بار تنها کنشی که در ابتدای لیست هست انتخاب نشود و کنش هایی با ارزش یکسان هم احتمال انتخاب شدن داشته باشند. پس از این گروه بندی و جا به جا شدن اعداد هم ارزش، با استفاده از نحوه انتخاب **epsilon-greedy** ارزش بیشینه انتخاب می شود. و سپس بررسی می شود که آیا می توان این کنش را انتخاب کرد یا خیر. اگر نه، عضو بعدی لیست ارزش ها انتخاب می شود و در آخر این کنش برمی گردد.

تابع **Q_learning**:

الگوریتم اصلی این سوال در این تابع نوشته شده است. به این صورت که کنش برگردانده شده از تابع انتخاب کنش (**choose_action**) به عنوان کنش به محیط داده می شود. سپس حالتی که سر انجام عامل با انتخاب کنش در آن قرار می گیرد به ما با نام **new_observation** برگردانده می شود. حال یک لیست به عنوان کنش های این حالت جدید می سازیم. اگر لیست خالی بود یعنی عامل هیچ کنشی نمی تواند انجام دهد و در چاله افتاده است. سپس پاداش چاله را کمتر می کنیم که باعث می شود عامل در دفعات بعد چاله را انتخاب نکند.

سپس با استفاده از فرمول **Q-learning** ارزش کنش انتخاب شده و همچنین ارزش خانه فعلی را به روز رسانی می کنیم. بعد از اتمام یک دور بازی دور بعد شروع می شود.

فاز دوم(فاز ارزیابی):

در این فاز که فقط یک دور بازی می شود با توجه به ارزش کنش های هر خانه، کنش با بیشترین ارزش انتخاب شده و کنش انجام می شود و عامل به هدف می رسد.

توجه شود که درصورت عدم تفهیم درست به کد مراجعه شود. کد به صورت کامل کامنت گذاری شده است.