باسمه تعالی

تمرین های هوش مصنوعی

محمد امیر نوروزی

دکتر عصایی معمم

تمرین اول : peas ربات فوتبالیست

معیارکارایی:

برد بازی ، گل زدن بیشتر از تیم حریف ، بیرون نرفتن از چهارچوب بازی ، خطا نکردن رو ی بازیکن حریف ، سرعت جابجایی قابل قبول برای پشت سر گذاشتن بازیکن تیم حریف ، توانایی

شناسایی و تشخیص عوامل محیطی و فیزیکی بازی ، استفاده نکردن دست بجز دروازه بان

محیط:

زمین چمن، فوتسال ، ساحل

عملگرها:

شوت ، چیپ ، سانتر کردن

سنسور:

سنسور تشخیص توپ ، سرعت توپ و جهت آن ، سنسور تشخیص فاصله ، سنسور آنالیز بازیکنان حریف ، سنسور خطوط سنسور تشخیص بازیکن حریف یا بازیکن خودی ، دروازه ها ، سنسور اب و هوا

تمرین دوم : با مسائل غیر قطعی چگونه رفتار میکنیم؟

راه حل مسائل غیر قطعی در هوش مصنوعی مرتبط با مدیریت و تصمیم گیری در شرایطی که دارای عدم قطعیت هستند می باشد. برای حل

اینگونه مسائل، میتوان از رویکردها و تکنیک های زیر استفاده کرد:

1)احتماالت و آمار:

استفاده از مفاهیم احتماالت و آمار برای مدل ساز ی و پیش بینی وقوع رویدادها در شرایط عدم قطعیت

2)مدل سازی بیزی :

استفاده از مدل های بیزی برای نمایش علاقه مندی ها و توزیع های احتمالی در مسائل غیر قطعی

3)تئور ی تصمیم گیر ی:

اعمال تکنیک های تصمیم گیری چون مدل های مارکوف تصمیم گیری و فرآیندهای تصمیم گیری مارکوف برای تعیین تصمیم های بهینه در شرایطی که دارای عدم قطعیت هستند

4)اطلاعات فازی :

استفاده از اطلاعات فازی برای مدل سازی عدم قطعیت و عدم دقت در داده ها و تصمیم گیری ها

5)تکنیک های ترکیبی:

ترکیب اطلاعات احتمالی و داده های مشاهده شده با دانش پیشین و تجربی به منظور بهبود تصمیم گیری در شرایط عدم قطعیت

6)الگوریتم های بهینه سازی:

استفاده از الگوریتم های بهینه سازی برای یافتن راه حل های بهینه در مسائل غیرقطعی

7)تکنیک های تحلیل حساسیت:

تجزیه و تحلیل حساسیت برای درک تأثیر پارامترها و عوامل مختلف بر نتایج تصمیم گیری در شرایط عدم قطعیت

8)شبکه های عصبی:

استفاده از شبکه های عصبی برای مدل سازی و پیش بینی در شرایط عدم قطعیت

ترکیبی از این رویکردها و تکنیکها بسته به مسئله مورد نظر و میزان عدم قطعیت می تواند به راه حل های موثری در مسائل غیر قطعی در هوش مصنوعی منجر شود

تمرین سوم : کد ۸ وزیر رو پیدا کنید و درمورد آن بررسی کنید

چک کردن آیا می توان وزیری را در سلول قرار داد یا خیر:

```
def is_safe(board, row, col, n):
```

چک کردن رد ی فافقی (سمت چپ):

```
for i in range(col):
    if board[row][i] == 1:
        return False
```

چک کردن قطر بالا به چپ:

```
for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
    if board[i][j] == 1:
        return False
```

چک کردن قطر پایین به چپ:

```
for i, j in zip(range(row, n, 1), range(col, -1, -1)):
    if board[i][j] == 1:
        return False
```

حالت پایه : اگر تمام وزیرها قرار گرفته باشند:

```
def solve_n_queens_util(board, col, n):
    if col >= n:
        return True
```

برای هر سلول در ستون فعلی:

```
    for i in range(n):
```

چک کردن آیا می توان وزیر را در این سلول قرار داد:

```
if is_safe(board, i, col, n):
```

قرار دادن وزیر در این سلول:

```
board[i][col] = 1
```

ادامه به جستجوی ستون بعدی:

```
if solve_n_queens_util(board, col + 1, n):
    return True
```

اگر قرار گرفتن وزیر در این سلول به حل مسئله منجر نشود آن را از صفحه حذف میکنیم:

```
board[i][col] = 0
```

اگر هیچ یک از سلول ها منجر به حل مسئله نشود:

```
def solve_n_queens(n):
    return False
```

ایجاد صفحه شطرنج خالی:

```
board = [[0 for _ in range(n)] for _ in range(n)]
```

حل مسئله با فراخوانی اولیه از ستون اول:

```
if not solve_n_queens_util(board, 0, n):
        print("هیچ راه حلی وجود ندارد")
            return False
```

نمایش جواب:

```
for i in range(n):
        for j in range(n):
            print(board[i][j], end=" ")
        print()
return True
```

برای حل مسئله 8 وزیر n = 8 تابع را فراخوانی میکنیم با:

solve_n_queens(8)

تمرین چهارم : به نظر شما پیچیدگی زمانی دستوپاگیر تره یا پیچیدگی حافظه؟

بستگی به نوع مسئله و الگوریتمی که برای حل آن استفاده می شود دارد که پیچیدگی حافظه یا زمانی کدام یک بیشتراست. در برخی موارد، مسائلی وجود دارند که پیچیدگی حافظه آنها بسیار بالاست و نیاز به استفاده از منابع حافظه بالا دارند. برای مثال، الگوریتم های که برای پردازش تصویر و صدا استفاده می شوند،به دلیل بزرگی حجم دادهایورودی نیز به استفاده از حافظه بالا دارند

درمقابل، در بسیاری از مسائل، پیچیدگی زمانی بیشتراز پیچیدگی حافظه است. به عنوان مثال الگوریتم هایی که برای مرتب سازی اعداد استفاده می شوند، نیاز به حافظه کمتری دارند ولی زمان بیشتری برای اجرای انها الزام است

بنابراین برای انتخاب بهترین الگوریتم برای حل یک مسعله باید به دو پیچیدگی حافظه و زمانی توجه کرد و الگوریتمی انتخاب کرد که برای ان مسعله پیچیدگی کمتری داشته باشد

تمرین پنجم : تحلیل WAMPUS:

بازی "Hunt the Wumpus" نشان‌دهنده مفاهیمی همچون جستجو در گراف، الگوریتم‌های DFS (جستجوی عمق اول)، BFS (جستجوی سطح اول) و الگوریتم‌های هوش مصنوعی مانند A* است. این الگوریتم‌ها در جستجوی هدف، کاوش محیط، و یافتن بهترین مسیر به کار می‌روند

در بازی "Hunt the Wumpus"، بازیکن نیازمند تصمیم‌گیری در مواجهه با شرایط ناقص و محیط پیچیده است. این شامل تصمیم‌گیری در مورد حرکت به یک اتاق خاص، اکتشاف گنجینه یا جلوگیری از ورود به اتاق حاوی Wumpus یا گودال است

برنامه‌ریزی حرکت در بازی نشان‌دهنده مفهومی اساسی در هوش مصنوعی است. بازیکن باید مسیر حرکت خود را در محیط مشخص کند تا به سرعت به هدف برسد و از خطرات جلوگیری کند.

بازی "Hunt the Wumpus" یک محیط آموزشی جذاب برای آشنایی با مفاهیم مهم هوش مصنوعی مانند جستجو، تصمیم‌گیری، ریسک و پاداش است. این بازی به بازیکنان کمک می‌کند تا مفاهیم اصلی هوش مصنوعی را در یک سیاق تعاملی فهمیده و به کار بگیرند.

بازیکن در مواجهه با محیط پویا و ناقص با مشکلاتی مانند نقاط کور، اطلاعات ناقص در محیط، و شوک‌های غیرمنتظره

روبرو می‌شود. این نشان می‌دهد چگونه هوش مصنوعی باید با شرایط ناپایدار و متغیر مواجه شود.

این بازی به بازیکنان نشان می‌دهد که چگونه تصمیم‌گیری در مواجه با ریسک‌های مختلف و انتخاب مسیرهایی که به شکار هدف می‌انجامد، بسیار مهم است.

به طور کلی، "Hunt the Wumpus" یک بازی آموزشی جذاب است که مفاهیم کلیدی هوش مصنوعی را به صورت عملی و تعاملی آموزش می‌دهد. این بازی به مسائل مرتبط با جستجو، تصمیم‌گیری هوشمندانه، و مدیریت ریسک در محیط‌های پویا می‌پردازد و می‌تواند در فهم بهتر دانشجویان در درس هوش مصنوعی موثر باشد