« به نام او »

آموزش شبیه سازی دو بعدی فوتبال

نویسندگان : محمدعلی میرزایی - علی یعقوبی

قسمت ششم

مرجع روبوکاپ ایران

www.iranrcss.com

مباحث این جلسه :

- ادامه آموزش بیس UVA-Trilearn، ساختار و کدنویسی
- آموزش هوش مصنوعی Artificial Intelligence

مقدمه :

هم اکنون شما با مطالعه جزوات، از آشنایی خوبی راجع به شبیه سازی فوتبال دوبعدی برخوردار شده اید. از این پس، آموزش ها بصورت تخصصی بهمراه بخش هوش مصنوعی ارائه میشوند. تنها چیزی که قابل ذکر است، دانش پژوه نباید خود را وابسته به این جزوات کرده و تحقیقات دیگری انجام ندهد. به شما توصیه میشود همزمان با مطالعه جزوات، کتاب های آموزش برنامه نویسی تخصصی و همچنین کتاب های مرجع و معتبر هوش مصنوعی نظیر A modern Approach راسل استوارت و همچنین کتب مربوط به سیستم های چند عامل ه (Multi-Agent systems) را تهیه و مطالعه نمایید.

ادامه آموزش بیس یو وی ای :

در جلسه قبل تا کد زیر پیش رفتیم :

```
۱ - else if( WM->getFastestInSetTo( OBJECT_SET_TEAMMATES, OBJECT_BALL, &iTmp )
۲ -              == WM->getAgentObjectType()  && !WM->isDeadBallThem() )
۳ -   {                                                  // if fastest to ball
۴ -     Log.log( ۱۰۰, "I am fastest to ball; can get there in %d cycles", iTmp
);
۵ -     soc = intercept( false );                        // intercept the ball
۶ -
۷ -     if( soc.commandType == CMD_DASH &&               // if stamina low
۸ -         WM->getAgentStamina().getStamina() <
۹ -            SS->getRecoverDecThr()*SS->getStaminaMax()+۲۰۰ )
۱۰-      {
۱۱-        soc.dPower = ۳۰.۰ * WM->getAgentStamina().getRecovery(); // dash slow
۱۲-        ACT->putCommandInQueue( soc );
۱۳-        ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۱۴-      }
۱۵-      else                                             // if stamina high
۱۶-      {
۱۷-        ACT->putCommandInQueue( soc );                 // dash as intended
۱۸-        ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۱۹-      }
```

```
۲۰-     }
۲۱-     else if( posAgent.getDistanceTo(WM->getStrategicPosition()) >
۲۲-                   ۱.۵ + fabs(posAgent.getX()-posBall.getX())/۱۰.۰)
۲۳-                                              // if not near strategic pos
۲۴-     {
۲۵-       if( WM->getAgentStamina().getStamina() >     // if stamina high
۲۶-                              SS->getRecoverDecThr()*SS->getStaminaMax()+۸۰۰ )
۲۷-       {
۲۸-         soc = moveToPos(WM->getStrategicPosition(),
۲۹-                           PS->getPlayerWhenToTurnAngle());
۳۰-         ACT->putCommandInQueue( soc );            // move to strategic pos
۳۱-         ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۳۲-       }
۳۳-       else                                        // else watch ball
۳۴-       {
۳۵-         ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT_BALL ) );
۳۶-         ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۳۷-       }
۳۸-     }
۳۹-  else if( fabs( WM->getRelativeAngle( OBJECT_BALL ) ) > ۱.۰ ) // watch ball
۴۰-     {
۴۱-       ACT->putCommandInQueue( soc = turnBodyToObject( OBJECT_BALL ) );
۴۲-       ACT->putCommandInQueue( turnNeckToObject( OBJECT_BALL, soc ) );
۴۳-     }
۴۴-     else                                          // nothing to do
۴۵-       ACT->putCommandInQueue( SoccerCommand(CMD_TURNNECK,۰.۰) );
۴۶-   }
```

حال توضیحاتی در مورد توابع بکار رفته در این چند خط خواهیم داد .

در خط شماره ۱ تابعی به نام `getFastestInSetTo` به کار رفته است . همانطور که می دانید در زبان c++ توابع این توانایی را دارند که هم نام باشند اما در شرایطی که آرگومان هایشان با هم متفاوت باشد . این تابع نیز دارای همین خاصیت است ، یعنی در این کلاس ۲ تابع با این نام وجود دارد . این تابع برای یافتن سریعترین بازیکن به یک ObjectT است . ما درباره ی آرگومان های این تابع صحبت می کنیم و بررسی توابع هم اسم دیگر را به خودتان واگذار میکنیم . دراولین آرگومان این تابع یک ObjectSetT می خواهد .

ObjectSetT یک enum با ۷ عضو می باشد . برای این تابع ۴ عضو OBJECT_SET_TEAMMATES ، OBJECT_SET_OPPONENTS ، OBJECT_SET_PLAYERS ، OBJECT_SET_TEAMMATES_NO_GOALIE به کار می آید . این اعضا یک مجموعه از بازیکنان می باشند که اعضای آنها با نام این اعضا مشخص می شود یعنی بطور مثال OBJECT_SET_OPPONENTS مجموعه ای از تمامی بازیکنان تیم حریف می باشد . ما با مشخص کردن این مجموعه به تابع بیان می کنیم که سریعترین بازیکن را از کدام بازیکنان انتخاب کند . آرگومان بعدی یک ObjectT می باشد . تعریف ObjectT - درفایل SoccerTypes.h می باشد - را اینجا قرار می دهم . توضیحات خود بیس گویای همه چیز است :

```
/*! ObjectT is an enumeration of all possible objects that are part of the
    RoboCup soccer simulation. The class SoccerTypes contains different methods
    to easily work with these objects and convert them to text strings and text
    strings to ObjectT. */
enum ObjectT { // don't change order
  OBJECT_BALL,              /*!< Ball                      */
  OBJECT_GOAL_L,            /*!< Left goal                 */    // ۲ goals
  OBJECT_GOAL_R,            /*!< Right goal                */
  OBJECT_GOAL_UNKNOWN,      /*!< Unknown goal              */
  OBJECT_LINE_L,            /*!< Left line                 */    // ۴ lines
  OBJECT_LINE_R,            /*!< Right line                */
  OBJECT_LINE_B,            /*!< Bottom line               */
  OBJECT_LINE_T,            /*!< Top line                  */
  OBJECT_FLAG_L_T,          /*!< Flag left top             */   // ۵۳ flags
  OBJECT_FLAG_T_L_۵۰,       /*!< Flag top left ۵۰          */
  OBJECT_FLAG_T_L_۴۰,       /*!< Flag top left ۴۰          */
  OBJECT_FLAG_T_L_۳۰,       /*!< Flag top left ۳۰          */
  OBJECT_FLAG_T_L_۲۰,       /*!< Flag top left ۲۰          */
  OBJECT_FLAG_T_L_۱۰,       /*!< Flag top left ۱۰          */
  OBJECT_FLAG_T_۰,          /*!< Flag top left ۰           */
  OBJECT_FLAG_C_T,          /*!< Flag top center           */
```

```
OBJECT_FLAG_T_R_١٠,      /*!< Flag top right ١٠        */
OBJECT_FLAG_T_R_٢٠,      /*!< Flag top right ٢٠        */
OBJECT_FLAG_T_R_٣٠,      /*!< Flag top right ٣٠        */
OBJECT_FLAG_T_R_٤٠,      /*!< Flag top right ٤٠        */
OBJECT_FLAG_T_R_٥٠,      /*!< Flag top right ٥٠        */
OBJECT_FLAG_R_T,         /*!< Flag right top           */
OBJECT_FLAG_R_T_٣٠,      /*!< Flag right top ٣٠        */
OBJECT_FLAG_R_T_٢٠,      /*!< Flag right top ٢٠        */
OBJECT_FLAG_R_T_١٠,      /*!< Flag right top ١٠        */
OBJECT_FLAG_G_R_T,       /*!< Flag goal right top      */
OBJECT_FLAG_R_٠,         /*!< Flag right ٠             */
OBJECT_FLAG_G_R_B,       /*!< Flag goal right bottom   */
OBJECT_FLAG_R_B_١٠,      /*!< Flag right bottom ١٠     */
OBJECT_FLAG_R_B_٢٠,      /*!< Flag right bottom ٢٠     */
OBJECT_FLAG_R_B_٣٠,      /*!< Flag right bottom ٣٠     */
OBJECT_FLAG_R_B,         /*!< Flag right bottom        */
OBJECT_FLAG_B_R_٥٠,      /*!< Flag bottom right ٥٠     */
OBJECT_FLAG_B_R_٤٠,      /*!< Flag bottom right ٤٠     */
OBJECT_FLAG_B_R_٣٠,      /*!< Flag bottom right ٣٠     */
OBJECT_FLAG_B_R_٢٠,      /*!< Flag bottom right ٢٠     */
OBJECT_FLAG_B_R_١٠,      /*!< Flag bottom right ١٠     */
OBJECT_FLAG_C_B,         /*!< Flag center bottom       */
OBJECT_FLAG_B_٠,         /*!< Flag bottom ٠            */
OBJECT_FLAG_B_L_١٠,      /*!< Flag bottom left ١٠      */
OBJECT_FLAG_B_L_٢٠,      /*!< Flag bottom left ٢٠      */
OBJECT_FLAG_B_L_٣٠,      /*!< Flag bottom left ٣٠      */
OBJECT_FLAG_B_L_٤٠,      /*!< Flag bottom left ٤٠      */
OBJECT_FLAG_B_L_٥٠,      /*!< Flag bottom left ٥٠      */
OBJECT_FLAG_L_B,         /*!< Flag left bottom         */
OBJECT_FLAG_L_B_٣٠,      /*!< Flag left bottom ٣٠      */
OBJECT_FLAG_L_B_٢٠,      /*!< Flag left bottom ٢٠      */
OBJECT_FLAG_L_B_١٠,      /*!< Flag left bottom ١٠      */
```

```
OBJECT_FLAG_G_L_B,       /*!< Flag goal left bottom    */
OBJECT_FLAG_L_٠,         /*!< Flag left ٠              */
OBJECT_FLAG_G_L_T,       /*!< Flag goal left top       */
OBJECT_FLAG_L_T_١٠,      /*!< Flag left bottom ١٠      */
OBJECT_FLAG_L_T_٢٠,      /*!< Flag left bottom ٢٠      */
OBJECT_FLAG_L_T_٣٠,      /*!< Flag left bottom ٣٠      */
OBJECT_FLAG_P_L_T,       /*!< Flag penaly left top     */
OBJECT_FLAG_P_L_C,       /*!< Flag penaly left center  */
OBJECT_FLAG_P_L_B,       /*!< Flag penaly left bottom  */
OBJECT_FLAG_P_R_T,       /*!< Flag penaly right top    */
OBJECT_FLAG_P_R_C,       /*!< Flag penaly right center */
OBJECT_FLAG_P_R_B,       /*!< Flag penaly right bottom */
OBJECT_FLAG_C,           /*!< Flag center field        */
OBJECT_TEAMMATE_١,       /*!< Teammate nr ١            */    // teammates ۶١
OBJECT_TEAMMATE_٢,       /*!< Teammate nr ٢            */
OBJECT_TEAMMATE_٣,       /*!< Teammate nr ٣            */
OBJECT_TEAMMATE_۴,       /*!< Teammate nr ۴            */
OBJECT_TEAMMATE_۵,       /*!< Teammate nr ۵            */
OBJECT_TEAMMATE_۶,       /*!< Teammate nr ۶            */
OBJECT_TEAMMATE_٧,       /*!< Teammate nr ٧            */
OBJECT_TEAMMATE_٨,       /*!< Teammate nr ٨            */
OBJECT_TEAMMATE_٩,       /*!< Teammate nr ٩            */
OBJECT_TEAMMATE_١٠,      /*!< Teammate nr ١٠           */
OBJECT_TEAMMATE_١١,      /*!< Teammate nr ١١           */
OBJECT_TEAMMATE_UNKNOWN, /*!< Teammate nr unkown       */
OBJECT_OPPONENT_١,       /*!< Opponent nr ١            */    // opponents ٧٣
OBJECT_OPPONENT_٢,       /*!< Opponent nr ٢            */
OBJECT_OPPONENT_٣,       /*!< Opponent nr ٣            */
OBJECT_OPPONENT_۴,       /*!< Opponent nr ۴            */
OBJECT_OPPONENT_۵,       /*!< Opponent nr ۵            */
OBJECT_OPPONENT_۶,       /*!< Opponent nr ۶            */
OBJECT_OPPONENT_٧,       /*!< Opponent nr ٧            */
OBJECT_OPPONENT_٨,       /*!< Opponent nr ٨            */
```

```
  OBJECT_OPPONENT_۹,       /*!< Opponent nr ۹            */
  OBJECT_OPPONENT_۱۰,      /*!< Opponent nr ۱۰           */
  OBJECT_OPPONENT_۱۱,      /*!< Opponent nr ۱۱           */
  OBJECT_OPPONENT_UNKNOWN, /*!< Opponent nr unknown      */    // ۸۴
  OBJECT_PLAYER_UNKNOWN,   /*!< Unknown player           */
  OBJECT_UNKNOWN,          /*!< Unknown object           */
  OBJECT_TEAMMATE_GOALIE,  /*!< Goalie of your side      */
  OBJECT_OPPONENT_GOALIE,  /*!< Goalie of opponent side  */
  OBJECT_ILLEGAL,          /*!< illegal object           */
  OBJECT_MAX_OBJECTS       /*!< maximum nr of objects    */ // ۹۰
} ;
```

ما با دادن این آرگومان به تابع می فهمانیم که هدف چیست ؛ به طور مثال با دادن OBJECT_OPPONENT_۱۱ به عنوان آرگومان ، به تابع می گوییم سریعترین بازیکن (از مجموعه ای که در آرگومان اول به تابع دادیم) به بازیکن شماره ۱۱ حریف کیست . خروجی این تابع یک objectT می باشد .

در خط اول شرطی قرار دارد که در صورتی درست است که سریعترین بازیکن به توپ خودمان (به نکته شماره ۱ توجه فرمایید) باشیم ؛ همچنین صاحب توپ بازیکنان حریف باشند (به نکته شماره ۲ توجه فرمایید) .

نکته ۱ : همانطور که قبلا گفته بودیم بیس در هر سایکل برای هر بازیکن بصورت جداگانه اجرا می شود؛ بنابراین منظور از بیان خودمان این است که سریعترین بازیکن به توپ بازیکنی باشد که در حال حاضر تابع برای او اجرا می شود .

نکته ۲ : تابع `isDeadBallThem()` یک تابع از کلاس WorldModel می باشد که بررسی می کند آیا بازیکنان حریف توپ را از دست داده اند یا نه . خروجی این تابع یک بولین است . البته بدیهی است دقیقا مشابه همین تابع برای بازیکنان خودی وجود دارد .

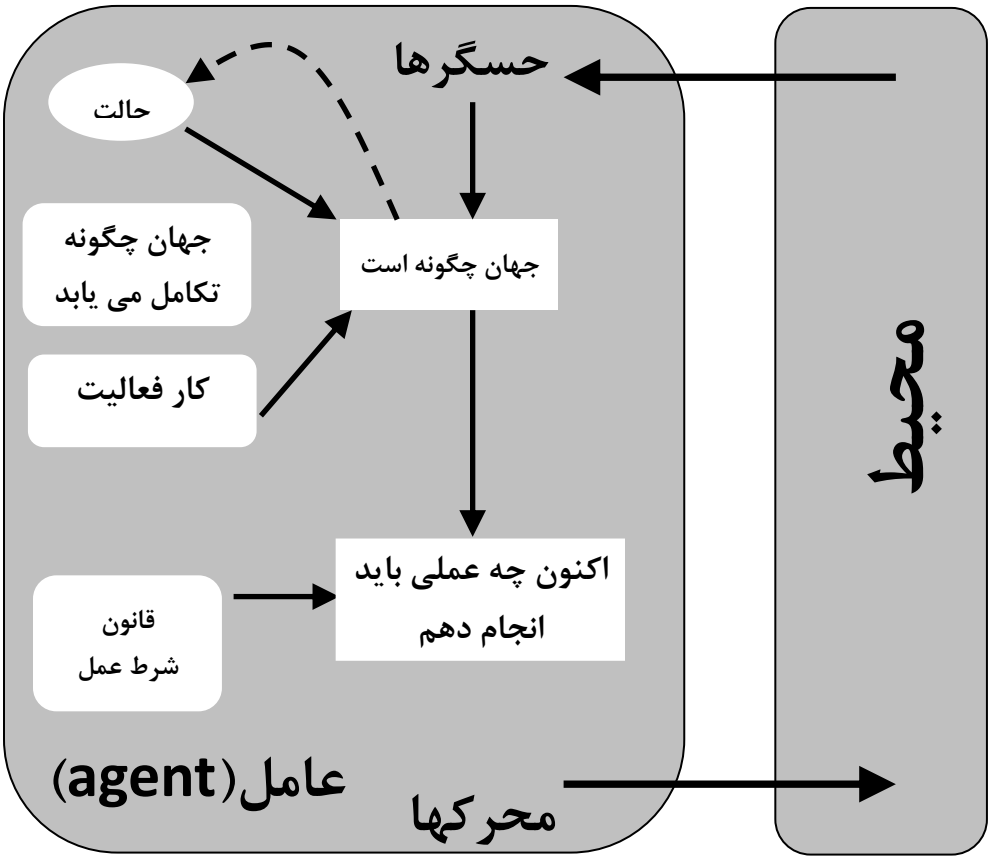
در جلسه ی بعد به ادامه ی این بحث خواهیم پرداخت .

هوش مصنوعی :

عاملهای هوشمند

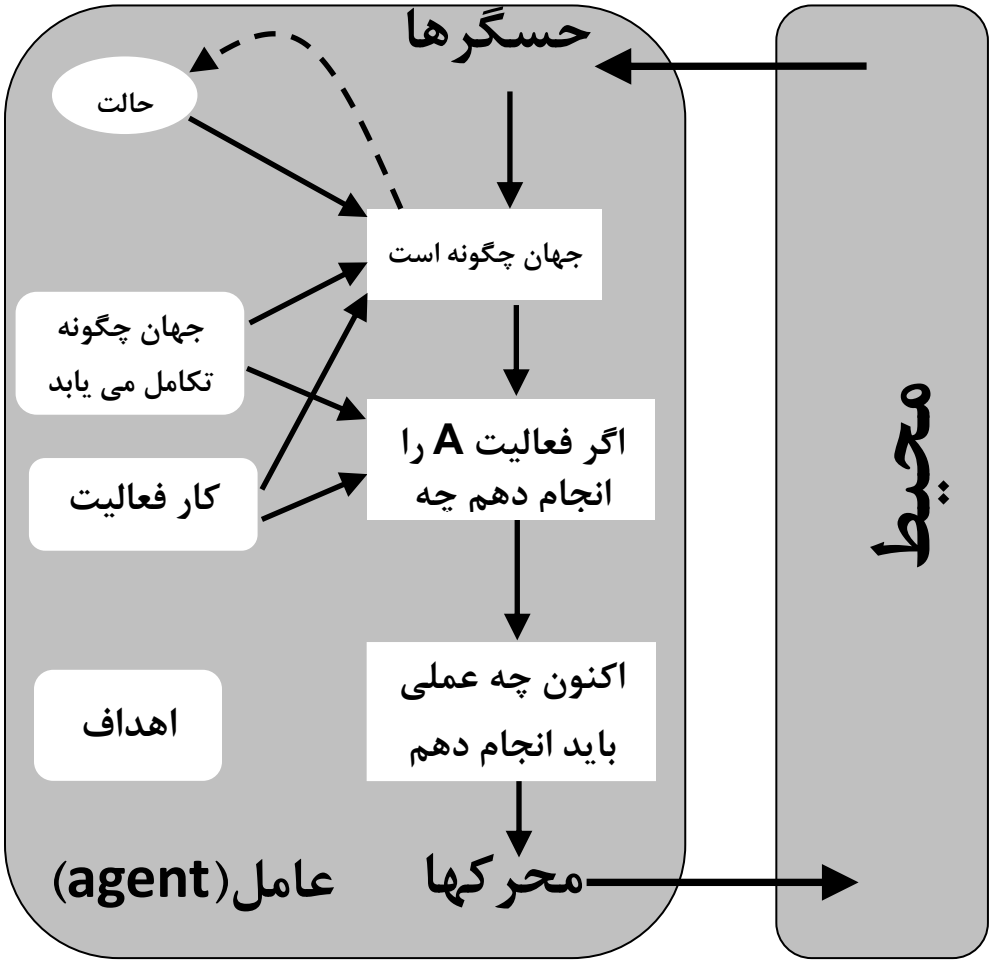
عاملهای واکنشی مدل گرا

- ✓ استفاده از دانش "چگونگی عملکرد جهان" که مدل نام دارد.
- ✓ عامل بخشی از دنیایی را که فعلا میبیند ردیابی میکند.
- ✓ عامل باید حالت داخلی را ذخیره کند که به سابقه ادراک بستگی دارد.
- ✓ در هر وضعیت, عامل میتواند توصیف جدیدی از جهان را کسب کند.



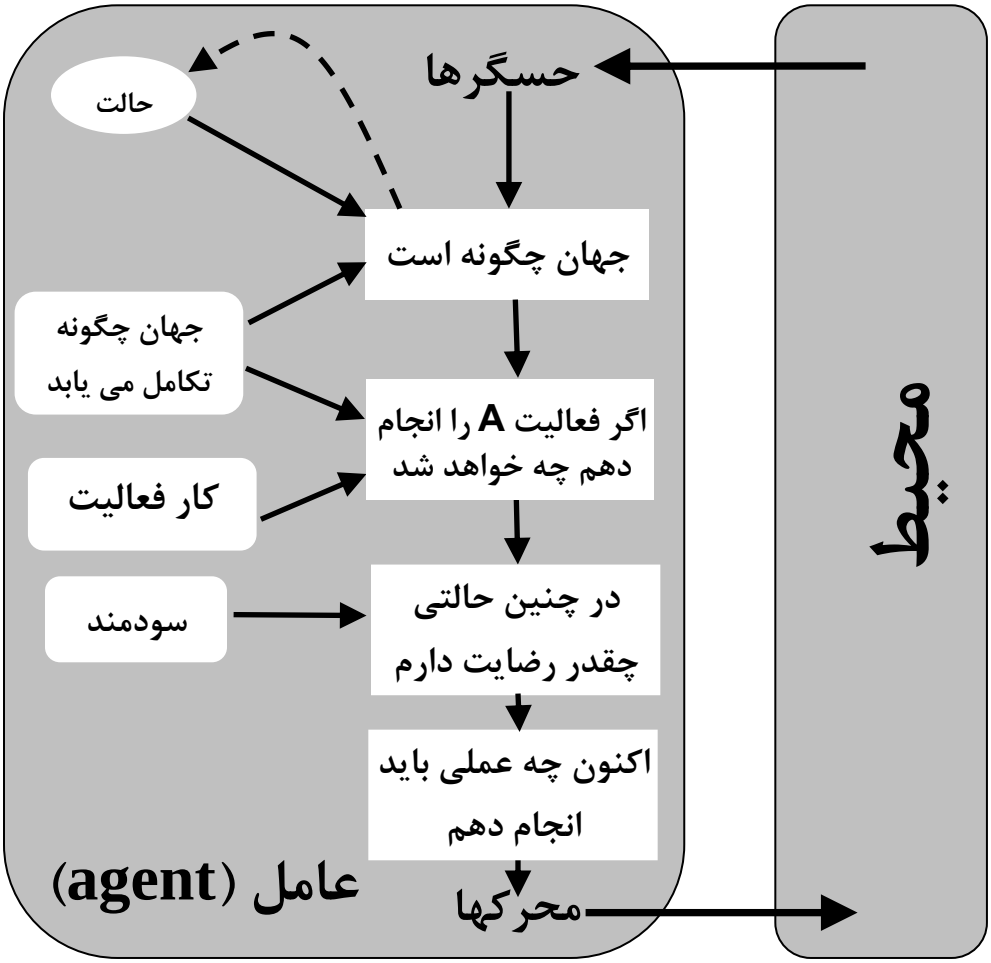
عاملهای هدف گرا

- ✓ این عامل علاوه بر توصیف حالت فعلی، برای انتخاب **موقعیت مطلوب** نیازمند اطلاعات **هدف** نیز میباشد.
- ✓ **جست و جو و برنامه ریزی**، دنباله ای از فعالیتها را برای رسیدن عامل به هدف، پیدا میکند.
- ✓ این نوع تصمیم گیری همواره **آینده** را در نظر دارد و با قوانین شرط عمل تفاوت دارد.
- ✓ این نوع عامل کارایی چندانی ندارد، اما **قابلیت انعطاف** بیشتری دارد.



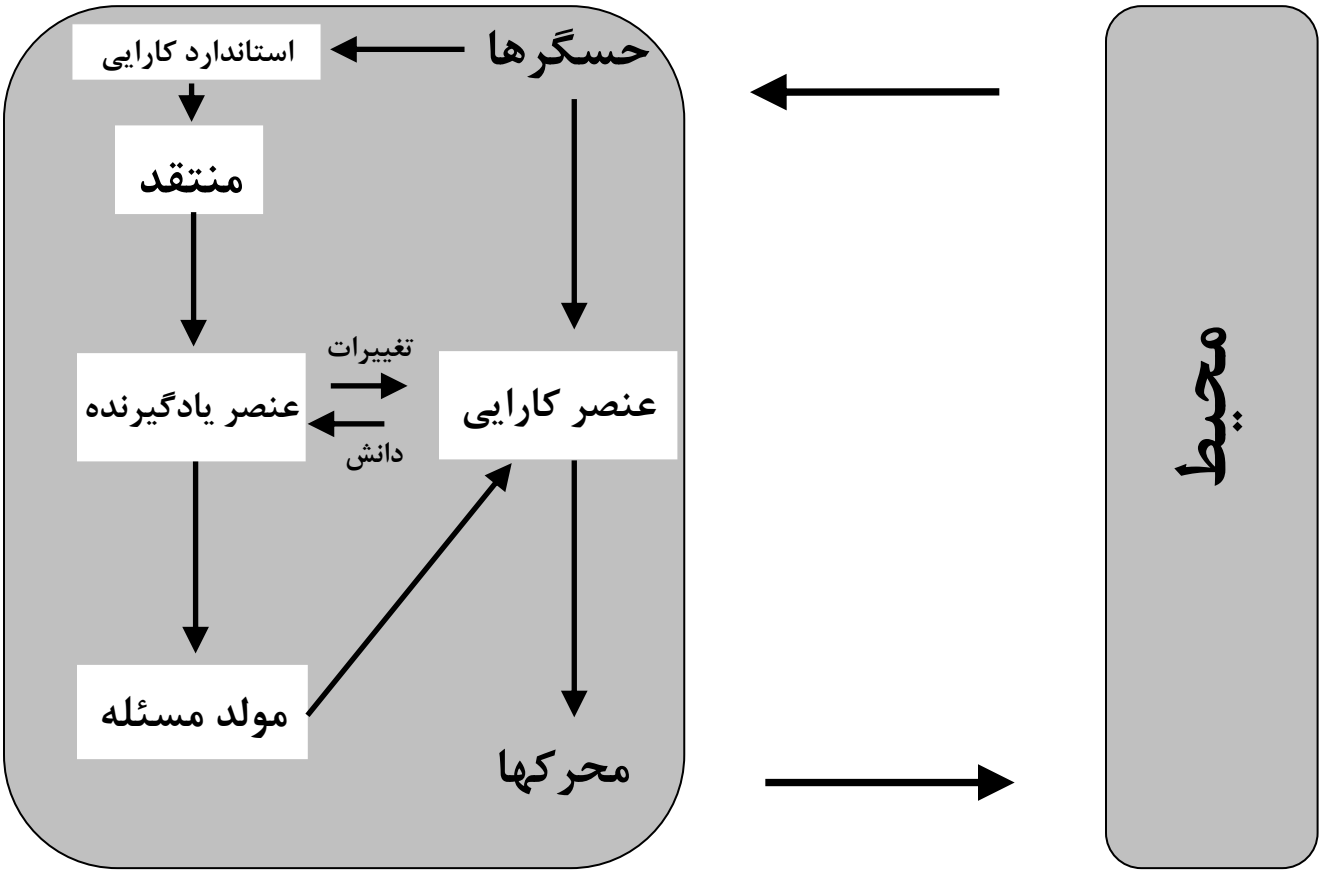
عاملهای سودمند

- ✓ این عامل برای **اهداف مشخص، راه های مختلفی** دارد، که راه حل بهتر برای عامل **سودمندتر** است.
- ✓ **تابع سودمندی**، حالت یا دنباله ای از حالتها را به یک عدد **حقیقی** نگاشت میکند که **درجه رضایت** را توصیف مِیکند.
- ✓ وقتی **اهداف متضاد** باشند، **بعضی** از آنها برآورده میشوند
- ✓ اگر **هیچیک از اهداف** به طور قطعی قابل حصول نباشند، احتمال موفقیت با **اهمیت هدف** مقایسه میشود.



عاملهای یادگیرنده

- ✓ عنصرِ**یادگیرنده** مسئول ایجاد بهبودها
- ✓ عنصر **کارایی** مسئول انتخاب فعالیتهای خارجی
- ✓ **منتقد** مشخص میکند که یادگیرنده با توجه به استانداردهای کارایی چگونه عمل میکند.
- ✓ **مولد مسئله** مسئول پیشنهاد فعالیتهایی است که منجر به تجربیات آموزنده جدیدی میشود.



روش حل مسئله با جستجو در هوش مصنوعی

فهرست:

چهار گام اساسی برای حل مسائل

- **فرموله کردن هدف: وضعیتهای مطلوب نهایی کدامند؟**
- **فرموله کردن مسئله: چه فعالیتها و وضعیتهایی برای رسیدن به هدف موجود است؟**
- **جستجو: انتخاب بهترین دنباله از فعالیتهایی که منجر به حالاتی با مقدار شناخته شده میشود.**
- **اجرا: وقتی دنباله فعالیت مطلوب پیدا شد، فعالیتهای پیشنهادی آن میتواند اجرا شود.**

در این قسمت، انواع عامل های هوشنمد را شناختید و با یکی از روش های حل مساله یعنی جستجو آشنا شدید. در قسمت بعدی، همین روند در مبحث حل مساله به روش جست و جو با مثال های کاربردی و اطلاعات تکمیلی دنبال خواهد شد.

پایان قسمت ششم