مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

مسأله

در مسأله ی مسافرت شخص دوره گرد ، یک دوره گرد باید کوتاه ترین مسیر را که یک مجموعه از شهرها را طی می کند ، پیدا نماید . فرض کنید که ما فاصله ی میان هر شهر را می دانیم . این مسأله ای مشکل می باشد ، زیرا تعداد مسیرهای ممکن برابر N! می باشد ، که N ، برابر با تعداد شهرها می باشد . الگوریتم ساده ای که در این مورد بهترین جواب را به سرعت ارایه نماید وجود ندارد . یک کد کننده ی کروموزوم ، یک عملکرد جهش و یک تابع Crossover را برای مسأله ی مسافرت شخص دوره گرد پیدا می نماید . فرض کنید تعداد شهرها (N) برابر ۱۰ می باشد . بعد از همه ی عملیّات ، کروموزوم های تولید شده باید همیشه مسافرت های ممکن مجاز را ارایه نمایند (هر شهر فقط باید یک بار ملاقات شود) . برای این مسأله ، یک راه حل وجود ندارد و تعداد زیادی الگوهای مختلف ، قبلاً استفاده شده اند .

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

فصل بیستم

سیستم های خبره [۱]

[۱] Expert systems

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

سیستم های خبره ، برنامه هایی طرّاحی شده برای مدل سازی و استدلال در مورد دانش انسان می باشند و معمولاً حول یک دامنه مثل ، تشخیص بیماری ، پرواز فضایی ، لجستیک (اقدامات مربوط به تهیه و توزیع) و عیب یابی نرم افزار تمرکز می کنند . در این سیستم ها ، برنامه نویسان ، *واقعیّات* و *قوانین* را اضافه می کنند و سیستم ، پردازه ی استنتاج را به صورت خودکار انجام می دهد و می توانند یا از زنجیره ی مستقیم (سیستم های تولید) یا از زنجیره ی معکوس (ثابت کننده های قضیّه) استفاده نمایند .

یک سیستم خبره چیست ؟

خبره – در صورتی که ما بگوییم که کسی در رشته ای ، یک خبره می باشد منظور ما این است که ، در آن تخصص محدود ، شایستگی بالایی را به نمایش می گذارد . اگر ما یک عامل تنها را بسازیم و آن را یک سیستم خبره بنامیم ، سیستم همیشه اطّلاعات کلّی تر را به نمایش نمی گذارد و در یک محدوده ی نسبی و با تخصّص نسبی کار می کند . امّا در آن زمینه عملکردش در سطح خبره می باشد . این سیستم ها ممکن است برای طبقه بندی ، تشخیص عیب ، پیداکردن خرابی ، تفسیر اطّلاعات ، طرّاحی ، پیکربندی یا پیش بینی به کار روند . آن ها شاید برای آموزش دادن ، نمایش دادن ، تحلیل ، مشاوره ، تجدید نظر یا کنترل به کار روند . استفاده ی تجاری سیستم های خبره شامل این موارد می باشند : یک سیستم استفاده شده برای ارایه ی

هوش مصنوعی

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

نصیحت در مورد وام خانه ؛ یک سیستم استفاده شده توسط یک تولید کننده ی کامپیوتر برای بررسی اجرای کامل دستورات مشتریان ؛ یک سیستم استفاده شده در بیمارستان برای تفسیر اندازه گیری های ریوی برای مشاهده ی علایم ناخوشی ریه ؛ یک سیستم استفاده شده توسط شیمیدان ها برای تفسیر توده ی داده های طیف سنج برای کمک به پی بردن به ساختار مولکولی ترکیبات آلی (زیستی) [1] ؛ و یک سیستم برای کمک به زمین شناسان برای ارزیابی مکان های معدنی برای ذخایر .

چرا یک سیستم خبره را می سازیم ؟

برای تکثیر خبره ی نادر و پرهزینه ؛ برای فرمول بندی دانش خبره و برای جمع آوری منابع دانش متمایز یک سیستم خبره را می سازیم . دلایل ممکن دیگری هم وجود دارند . برخی از افراد دلیل می آورند که سیستم های خبره در مقایسه با انسان ها کم تر خطا می کنند و مناسب تر هستند و رک تر هستند ، یا در مقایسه با خبره های بشری ، بی طرف تر می باشند . البتّه ، سیستم های خبره معایب زیادی هم دارند ، که باید استفاده از آن ها را به دامنه های مشخّص ، محدود نماییم . سیستم های خبره دارای بینش ، دلسوزی ، فهم انگیزه ی بشری ، توانایی حدس ، توانایی یادگیری (معمولاً) ، دانش قضاوت صحیح (عقل سلیم) کمی می باشند . در این مورد ، کاربر می تواند دلسوزی و قضاوت صحیح را اضافه نماید .

خصوصیّات خوب یک سیستم خبره

در صورتی که سیستم خبره ، به صورت محاوره ای باشد ، یک رابط کاربری خوب ، بسیار ضروری می باشد . توجه کنید که مکالمه ای که سیستم خبره با کاربر انجام می دهد باید به صورت "طبیعی" توسط کاربر مطرح شده باشد . که شامل مواردی نظیر این موارد می باشد : شیوه ی سؤالاتی که سیستم می پرسد

[1] ماده ای که مولکول های آن شامل یک یا بیش تر اتم های کربن است ، به استثنای کربنات ها ، سیانیدها ، کربیدها ، و تعدادی موارد دیگر (دایره المعارف بریتانیکا) ، در ضمن در لغت نامه ی مهندسی محیط هم ، چنین آمده است : گروه بزرگی از ترکیبات شیمیایی که معمولاً شامل کربن ، هیدروژن ، نیتروژن و اکسیژن می باشند . تمام موجودات زنده از این ترکیبات آلی تشکیل شده اند .

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

باید طبیعی باشد . سؤالات احمقانه نباید وجود داشته باشند (کسانی که به سیستم جواب می دهند باید با دلیل ، تدبیر کرده باشند) . سیستم باید قادر باشد که توضیح دهد که چرا یک سؤال را می پرسد و هر نتیجه ای که به آن می رسد را توجیه نماید و باید به کاربر در مورد سیستم ، اطمینان دهد .

در استدلال ، باید یک سیستم خبره ، قادر به انجام این موارد باشد : باید استنتاج هایش باورکردنی و نه الزاماً صحیح باشند ؛ به عنوان مثال ، یک سیستم خبره که یک وضعیت بهداشتی را از یک مجموعه از علایم ، استنتاج می کند ؛ بعید است که وضعیّت ، رشته ای منطقی از علایم (نشانه ها) باشد . باید قادر به کار با دامنه ی دانش ناقص و مواردی که اطّلاعات ناقص هستند ، باشد . اغلب ، دانش و / یا مورد اطّلاعات ، ناکامل می باشند و دانش و / یا داده ها ممکن است قابل اعتماد نباشند (مثلاً ، ممکن است دارای خطا باشند) و شاید دانش و / یا داده ها به صورت نادرست بیان شده باشند (به عنوان مثال ، " اگر دارای دندان بلندی باشد ، آن گاه این خطرناک می باشد . ") . همچنین باید سیستم ، رقابت همزمان مفروضات را در نظر داشته باشد .

سیستم های خبره باید با نگهداشت پذیری (ویژگی مربوط به جداسازی و تعمیر یک خرابی) [۱] طرّاحی شده باشند . باید بتوانند به سادگی اطّلاعات جدید را یکی نمایند (یا از طریق مهندسی دانش بیش تر یا با استفاده از آموزش ماشینی) .

سیستم های خبره ی قانون گرا [۲]

روش های زیادی می توانند برای ساخت سیستم های خبره استفاده شوند . امّا ، اکثر سیستم های خبره که تا کنون ساخته شده اند سیستم های قانون گرا می باشند ؛ یعنی ، پایگاه دانش ، شامل واقعیّت ها و قانون ها می باشد .

پایگاه دانش

---

[۱] maintainability

[۲] rule-based

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

پایگاه دانش ، شامل واقعیّت ها و قوانین می باشد . تصوّر کنید کـه مـا در حـال سـاخت یـک سیسـتم خبره ی قانون گرا برای تشخیص پزشکی بودیم . قانون ها ارتباط های میان علایـم و عبـارت هـای پزشـکی را کد خواهند کرد . واقعیّت ها دانش در مورد بیماری جاری را کد خواهند کرد . در سیسـتم هـای قـانون گـرا ، اغلب به صورت اگر ... آن گاه ... نوشته می شوند ، اما به صورت برابر می توانند بـه یکـی از شـکل هـایی کـه برای موارد زیر استفاده می شوند نوشته شوند :

if p_1 AND p_2 … AND p_n then q

$(p_1 \wedge \ldots \wedge p_n) \Rightarrow q$

$q \vee \neg p_1 \vee \ldots \vee \neg p_n$

برای سیستم های قانون گرا ، استفاده از منطق گزاره ای ، زمانی که گزاره هـا دارای هـیچ آرگومـانی نمی باشند ، کاملاً معمولی است . بنابراین متغیّرها و سمبل های تابعی در واقعیّت ها و قانون ها وجـود ندارنـد . ما در مثال های زیر خودمان را با این وضعیّت ، محدود نموده ایم . نتیجه ی یک قانون شاید یـک قلـم تجزیـه ناپذیر باشد که در قانون قبلی آمده است . ما می تـوانیم ایـن زنجیـره را بـه شـکل گرافیکـی بـه صـورت یـک گراف AND-OR نمایش دهیم . برای مثال ، برای قانون های زیر :

if p AND q **then** r

if s AND t **then** r

if u **then** p

if v **then** p

if w AND x **then** s

if y **then** s

if z **then** t

که گراف زیر را در مورد قانون های بالا داریم :

در زیر مثالی از پایگاه دانشی برای شناسایی حیوانات را می بینید :

if *animalGivesMilk* **then** *animalIsMammal*[1]

if *animalHasHair* **then** *animalIsMammal*

if *animalIsMammal* AND *animalChews*[2]*Cud*[3] **then** *animalIsUngulate*[4]

if *animalIsUngulate* AND *animalHasLongNeck*[5] **then** *animalIsGiraffe*[6]

if *animalIsUngulate* AND *animalIsStriped*[7] **then** *animalIsZebra*[8]

ما از این مثال ، بعداً استفاده خواهیم کرد .

---

[۱] پستاندار
[۲] می جَوَد
[۳] نُشخوار می کند
[۴] سُم دار
[۵] گردن
[۶] زرّافه
[۷] راه راه یا خط دار
[۸] گورِ خر

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

موتور استنتاج

موتور استنتاج ، استنتاج ها را با استفاده از دانش موجود در پایگاه دانش ، استخراج می کند و با استفاده از استنتاج قانون گرا [۱] انجام می دهد . از یک نقطه نظر منطقی ، اساساً از تحلیل استفاده می کند . یک چشم انداز (دورنمای سه بعدی) استنتاج (استدلال) قانون گرا این است که در حال انجام یک جستجوی گرافی AND-OR می باشد . به طور مؤثّر ، ما به دنبال یک مسیر که ریشه و برگ ها را به هم متّصل می کند و این موارد را اجرا می کند می گردیم : در صورتی که گره ای ، یک گره ی OR می باشد ، برای نمایش این که یک مسیر برای فقط یکی از نسل ها وجود دارد ، مناسب می باشد ؛ اگر گره ای ، یک گره ی AND باشد ، برای نمایش این که برای هر یک از نسل ها مسیری وجود دارد ، لازم می باشد ؛ در صورتی که یک گره ، یک برگ باشد ، گره یک واقعیّت را که باید درست باشد را نمایش می دهد .

در سیستم های خبره ی محاوره ای ، ما فرض نمی کنیم که همه ی واقعیّات شناخته شده اند و قبلاً در پایگاه دانش وجود داشته اند . بنابراین ، در زمانی که ما در تلاش برای این هستیم که ببینیم که آیا یک گره ی برگ (واقعیّت) درست است یا نه ، ما پایگاه دانش را برّرسی خواهیم نمود ، اما اگر در پایگاه دانش نباشد ، در این صورت ما از کاربر پرسش می کنیم که آیا واقعیّت درست است یا نه . جواب کاربر می تواند به پایگاه دانش اضافه شود . (این محاوره ، چیزی است که استدلال قانون گرا را از تحلیل SLD متمایز می کند .)

زنجیره ی معکوس – بیش تر استدلال قانون گرا به روشی انجام می شود که شبیه تحلیل SLD می باشد (همان طوری که در پاراگراف قبل گفته شد ، دلایلی وجود دارد که به ندرت با تحلیل SLD برابر می باشند) . امّا ، عبارت های زنجیره ی معکوس ، استدلال مشتق شده از هدف می باشند و استدلال مشتق شده از فرض به صورت عمومی تری در جامعه ی سیستم های خبره مورد استفاده قرار می گیرند . در عبارت

---

[۱] Rule-Based Reasoning (RBR)

های گراف AND-OR ، این نوع از استدلال در ریشه ی گراف شروع می شود و برای پیداکردن مسیری از ریشه به برگ ها تلاش می نماید .

زنجیره ی مستقیم – این روش ، استدلال مشتق شده از داده ها هم نامیده می شود . در عبارات گراف AND-OR ، این نوع از استدلال از برگ ها شروع می کند و تلاش می کند که مسیری را از برگ ها به طرف ریشه پیدا نماید .

زنجیره ی معکوس و مستقیم برگ برگ شده [۱] – برخی از سیستم ها منحصراً از یکی از این روش ها یا هر دوی این روش ها استفاده می کنند . امّا تعداد زیادی از سیستم های برگ برگی از هر دو روش استفاده می کنند که این کار خیلی طبیعی می تواند باشد . به یک مشاوره با یک دکتر بشری توجّه نمایید . بیمار برخی از علایم را تشریح می کند . نتایج با استفاده از زنجیره ی مستقیم به دست می آید (شاید فقط به طور آزمایشی) . دکتر یک فرض را انتخاب می کند و با استفاده از زنجیره ی معکوس به پرسشی می رسد که برای بیمار ارایه شده است . بیمار ، دوباره صحبت می کند ، و شاید به پرسشی دیگر جواب دهد و یا از او اطّلاعات دیگری خواسته شود . و این کار (پروسه) باز هم تکرار شود .

توضیح های استدلال

ما قبلاً اهمّیّت داشتن توضیح روان را در یک سیستم خبره بیان کردیم . سیستم های قانون گرا معمولاً دو امکان را برای توضیحات ارایه می کنند . ممکن است یا بپرسند چرا و یا بپرسند چگونه . سیستم ، سؤالات را با نمایش برخی از قوانین مربوط ، جواب می دهد . برای مثال ، تصوّر نمایید که سیستم خبره ی شناسایی حیوانات از کاربر این پرسش را می پرسد که آیا حیوان ، نشخوار می کند . در این مورد ، کاربر می تواند به جای جواب دادن به این سؤال بپرسد که : چرا شما این سؤال را می پرسید ؟ برای این کار ، سیستم خبره با نمایش یک گراف AND-OR پاسخ می دهد . برای مثال ، ممکن است سیستم پاسخ دهد : *من از شما پرسیدم که آیا حیوان نشخوار می کند ، زیرا این در تشخیص این که حیوان ، جانور سم دار است کمک می*

[۱] Interleaved Backwards- and Forwards-Chaining

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

کند . قبلاً تشخیص داده شده که جانور ، پستاندار است . بنابراین اگر حیوان نشخوار می کند آن گاه حیوان ، سم دار می باشد . این در تشخیص این که آیا حیوان زرّافه است کمک می کند . در صورتی که حیوان سم دار باشد و دارای گردن بلند باشد ، زرّافه می باشد . به بیان دیگر ، تصوّر نمایید که یک سیستم خبره تشخیص داده که برخی از گره ها درست هستند . در گفتن نتیجه به کاربر ، کاربر می تواند چگونگی توضیح را بپرسد ؛ یعنی این که بپرسد : چگونه به این نتیجه رسیدی ؟ برای این کار ، سیستم خبره با نمایش قسمت های موفّق گراف AND-OR ، جواب می دهد . ایده این است که برای تصدیق (توجیه کردن) یک استنتاج ، سیستم باید نشان دهد که کدام قانون ها در رسیدن به آن نتیجه اجرا می شوند . برای مثال ، تصوّر کنید در موردی که سیستم تشخیص می دهد که حیوان یک جانور سم دار می باشد ، کاربر ممکن است درخواست کند که چگونه به این نتیجه رسیدی و این کار ممکن است به صورت زیر باشد : این قانون برای تشخیص این که حیوان ، سم دار است استفاده شده : اگر حیوان ، پستاندار است و نشخوار می کند ، سم دار می باشد . این قانون برای تشخیص این که آیا جانور پستاندار است استفاده شده است : حیوان دارای مو می باشد ، پس پستاندار می باشد . شما به من گفتید که حیوان دارای مو می باشد . شما به من گفتید که حیوان نشخوار می کند .

بیش تر تحقیقات انقلابی در سیستم های قانون گرا در مورد سیستمی به نام MYCIN انجام شده است . ایده های درون MYCIN دارای برتری بر همه ی سیستم های خبره ی قانون گرا هستند . MYCIN برای استفاده ی یک پزشک برای تشخیص عفونت های باکتریایی خون طراحی شد . در حدود ۴۵۰ قانون دارد و تقریبا به طور کامل با استفاده از زنجیره ی معکوس کار می کند . بازده MYCIN و طبیعی بودن مکالمه ی آن با مشارکت قطعه های خیلی کوچک دانش که شاید در مدّت زنجیره ی معکوس به کار گرفته شوند ، بهبود یافت . یکی از پرمعنی ترین این موارد ، استفاده از یک مجموعه از فرا قانون ها می باشد که برای کمک به تصمیم در مورد این که کدام قانون های نرمال به صورت بعدی باید استفاده شوند ، طراحی شدند . یکی دیگر از سیستم های قانون گرای خبره ی مشهور ، PROSPECTOR (برای ارزیابی مکان های معدنی دارای استعداد برای استخراج) و R1 (a.k.a. XCON) برای برّرسی ، کامل کردن و پیکربندی تقاضاهای تجهیزات کامپیوتری مشتری ، می باشند . بعد از برخی از آزمایش ها با ساخت چند

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸

سیستم خبره ی قانون گرا ، محقّقان هوش مصنوعی دریافتند که فقط پایگاه دانش ، وابسته به دامنه مـی باشـد . موتور استنتاج و رابط کاربر (نسبتاً) مستقل از دامنه بودند . به طور کلّی ، برای سـاخت یـک سیسـتم خبـره ی جدید برای یک دامنه ی مختلف به یک پروسه ی مهندسی دانش برای دریافت کردن قانون ها نیاز داریم .

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸

۱

فصل بیست و یکم

پردازش های تصمیم گیری مارکوف [۲]

[۱] تصویر بالا متعلّق به آندرِی اِی . مارکوف ((۱۹۲۲ – ۱۸۵۶) Andrei A. Markov) ، دانشمند و ریّاضیدان معروف روسی است .

[۲] Markov Decision Processes(MDP)

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

پردازش های تصمیم گیری مارکوفی

یک مدل گسسته برای تصمیم گیری تحت نـامعلومی مـی باشـد . چهـار جـزء ایـن مـدل عبارتنـد از : *وضعیّت ها* ؛ که جهان به وضعیّت هایی تقسیم شده است . *عملکردها* ؛ هر وضعیّت دارای یک تعـداد محـدود از عملکردها می باشد . *تابع انتقال* ؛ ارتباطی احتمالی میان وضعیّت ها و عملکردهای قابـل دسترسـی بـرای هـر وضعیّت و *تابع پاداش* ؛ پاداش مورد انتظار انجام یک کار ، تحت وضعیّت S می باشد . [۱]

به وجود آوردن تصمیم گیری های ترتیبی

قبلاً ما در مورد این موارد صحبت کردیم : به وجود آوردن تصـمیم گیـری هـای بـه یکبـاره در یـک محیط قطعی ؛ به وجود آوردن تصمیم گیری های ترتیبی در یک محیط قطعـی ، مثـل : جسـتجو – اسـتنتاج – برنامه ریزی ؛ به وجود آوردن تصمیم گیری های به یکباره در یک محیط اتّفـاقی ، مثـل : شـبکه هـای بـاور و

[۱] http://www.sci.brooklyn.cuny.edu/~parsons/courses/790-spring-2004/notes/incremental.ppt

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

احتمال - پیش بینی سودمندی . حال این سؤال مطرح است که : تصمیم گیری های ترتیبی در یک محیط اتّفاقی ، چگونه می باشند ؟ .

سودمندیِ مورد انتظار

به یاد بیاورید که سودمندی مورد انتظار یک عملکرد ، سودمندی هر نتیجه ی ممکن است و توسّط احتمالی که نتیجه اتّفاق بیفتد ارزیابی (وزن) می شود ؛ به طور صریح تر ، از وضعیّت s ، یک عامل ممکن است عملکردهای $a_1 ، a_2 ، ... ، a_n$ را بگیرد . هر عملکرد a_i می تواند منجر به وضعیّت های $s_{i1}, s_{i2}, ..., s_{im}$ با احتمال $p_{i1}, p_{i2}, ..., p_{im}$ شود ؛ در نتیجه ، سودمندی مورد انتظار برای عملکردها برابر است با : $EU(a_i) = \sum p_{ij} s_{ij}$. ما مجموعه ای از احتمال ها و وضعیّت های وابسته را *مدل انتقال وضعیّت* [1] می نامیم . عامل باید وضعیّت 'a ای که سودمندی مورد انتظار را بیشینه می کند ، انتخاب نماید .

محیط های مارکوفی

ما می توانیم این ایده را برای محیط های ترتیبی بسط دهیم . در این مورد ، مسأله این است که چگونه احتمال وضعیّت ها را تشخیص دهیم ؟ ؛ احتمال رسیدن به وضعیّت s ارایه شده توسط عملکرد a ممکن است وابسته به عملکردهای قبلی ای که انجام شده اند باشد . فرض مارکوف می گوید که احتمال تغییر وضعیّت ها فقط وابسته به یک تعداد محدودی از والدها می باشد . ساده ترین ، پردازه ی مرتبه ی اوّل مارکوف در این صورت است که احتمال تغییر وضعیّت ها فقط وابسته به وضعیّت قبلی باشد . ما نیز روی پردازش مرتبه ی اوّل مارکوف تمرکز می نماییم .

توزیع های ثابت [2]

---

[1] state transition model

[2] Stationary distributions

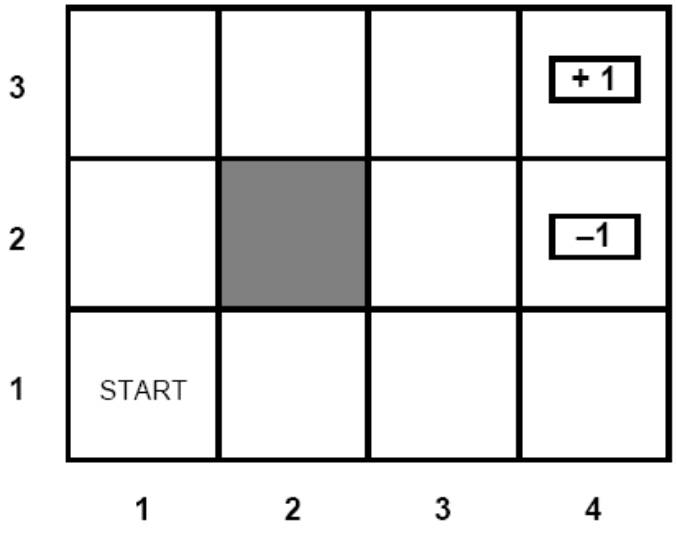
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

ما فرض می کنیم که توزیعمان ثابت باشد ، *که این به این معنی است که احتمال رسیدن به یک وضعیّت s' ارائه شده توسط عملکرد a از وضعیّت s با تاریخچه ی H تغییر نمی یابد* ؛ تاریخچه های مختلف ، ممکن است احتمال های مختلف را به وجود بیاورند . داشتن تاریخچه های یکسان ، تغییر وضعیّت های یکسان را سبب خواهد شد . ما همچنین فرض خواهیم کرد که سودمندی وضعیّت ، تغییری را در روش مسأله به وجود نمی آورد .

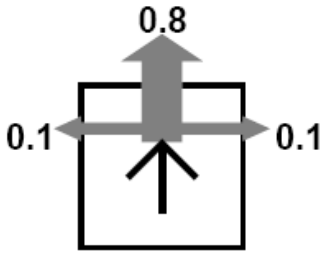
حل مسایل ترتیبی

در این مورد ، سودمندی ، وابسته به یک رشته از وضعیّت های $s_1, s_2, \ldots, s_n$ می باشد و برای هر وضعیّت ، یک پاداش $R(s_i)$ را در نظر می گیریم . عامل می خواهد مجموع پاداش ها را بیشینه نماید . *ما این فرمول بندی را یک پردازش تصمیم گیری مارکوفی می نامیم* . به طور صریح ، این موارد را در یک پردازه ی تصمیم گیری مارکوفی داریم : یک وضعیت اوّلیه ی s_0 ؛ یک مجموعه از وضعیّت ها و عملکردهای گسسته ؛ یک مدل انتقال : $T(s,a,s')$ که احتمال رسیدن به وضعیّت s' از s وقتی که عمل a انجام می شود را نشان می دهد و یک تابع پاداش $R(s)$.

مثال : مسأله ی شبکه



(a)



(b)

در این مثال ، عامل در جهت خواسته شده با احتمال ۰.۸ حرکت می کند و در یک جهت راست یا چپ با احتمال ۰.۲ حرکت می کند ؛ در هر وضعیّت ، یک عامل باید چه کاری انجام دهد تا پاداش را بیشینه نماید ؟

راه حل های پردازه ی تصمیم گیری مارکوفی – از آنجایی که محیط ، تصادفی یا اتّفاقی می باشد ، یک راه حل به صورت رشته ای از عملکرد نمی باشد . در عوض ، ما باید معیّن نماییم که یک عامل باید چه کاری در هر وضعیّت ممکن انجام دهد . ما این خصوصیّت را یک *روش*[۱] می نامیم : " اگر شما زیر هدف می باشید بالا بیایید و اگر شما در ستون سمت چپ می باشید ، به راست حرکت نمایید . " ما یک روش را با π نشان می دهیم و $\pi(s)$ روش را برای وضعیّت s ، مشخّص می نماید .

مقایسه ی روش ها – ما می توانیم روش ها را با توجه به سودمندی مورد انتظار تاریخچه هایی که آن ها تولید می کنند مقایسه نماییم . روشی که دارای بالاترین سودمندی مورد انتظار می باشد *روش بهینه*[۲]

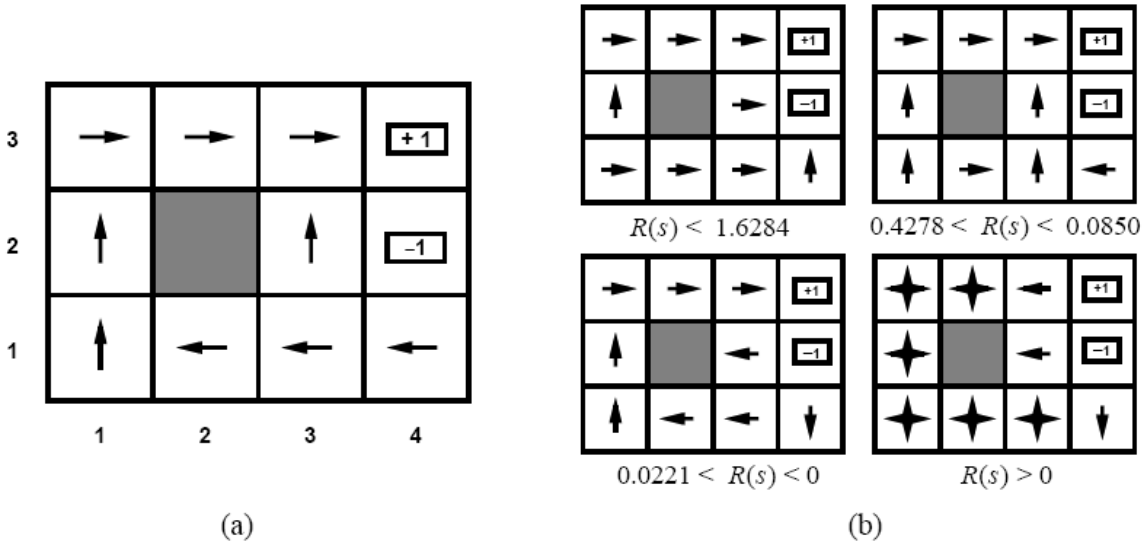
---

[۱] policy

[۲] optimal policy

نام دارد . اوّلین باری که یک روش پیدا می شود ، عامل فقط می تواند بهترین عملکـرد را بـرای هـر وضـعیّت پیدا نماید .

مثال : مسأله ی شبکه



توجّه کنید که در شکل سمت چپ ، $R(s)$ برابر است بـا ۰۴ . ۰ − و در شـکل ، تمـام پـاداش هـای غیر صفر ، باید منفی باشند . از آنجایی که هزینه های تغییر در وضعیّت های غیرپایانی تغییر می یابـد ، بنـابراین ، روش بهینه را انجام می دهد . وقتی که هزینه خیلی بالا باشد ، عامل تلاش می کند که بلافاصله خارج شود ؛ در زمینه ی میانی ، عامل تلاش می نماید که از خروج بد جلـوگیری نمایـد و در پـاداش مثبـت ، عامـل تـلاش نمی کند که خارج شود .

مطالب بیش تری در مورد توابع پاداش

در حل یک پردازه ی تصمیم گیری مارکوفی ،یک عامل بایـد بـه مقـدار عملکردهـای آینـده توجّـه نماید . انواع مختلفی از مسایل که باید به آن ها توجّه نماید ، وجود دارد :

- وسعت – آیا جهان برای همیشه ادامه می یابد ؟

o وسعت محدود : بعد از N عملیّات ، جهان متوقّف می شود و هیچ پاداشی دریافت نمی شود .

o وسعت نامحدود : جهان به صورت نامحدود ادامه می یابد یا این که ما نمی دانیم چه هنگامی متوقّف می شود . در وسعت نامحدود روش ها در طول زمان تغییر نمی کنند .

- ما همچنین نیاز داریم که در مورد چگونگی مقدار پاداش در آینده فکر کنیم ، صد دلار در یک روز دارای ارزش بیش تری نسبت به صد دلار در یک سال است . ما با *تخفیف* [۱] در پاداش های آینده این مورد را مدل سازی می نماییم .

اگر γ ، فاکتور تخفیف باشد داریم :

$$U(s_0, s_1, s_2, s_3, ...) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + ..., \gamma \in [0,1]$$

در صورتی که γ بزرگ باشد ، ما وضعیّت های آینده را مقدار دهی می نماییم . در صورتی که γ کوچک باشد ما روی پاداش نزدیک تمرکز می نماییم . در موارد پولی ، یک فاکتور تخفیف γ برابر است با یک بهره با نرخ $(1/\gamma) - 1$. تخفیف به ما اجازه می دهد که مقدار محسوسی ، مسایل با وسعت نامحدود را داشته باشیم . در غیر این صورت ، همه ی سودمندی های مورد انتظار باید نزدیک به مقدار نامحدود باشند . سودمندی های مورد انتظار در صورتی که پاداش ها محدود و در محدوده باشند و $\gamma<1$ باشد ، محدود خواهند شد . ما حالا می توانیم روش بهینه ی π^* را به این صورت بیان نماییم :

$$\pi^* = \arg\max_{\pi} EU\left(\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right)$$

---

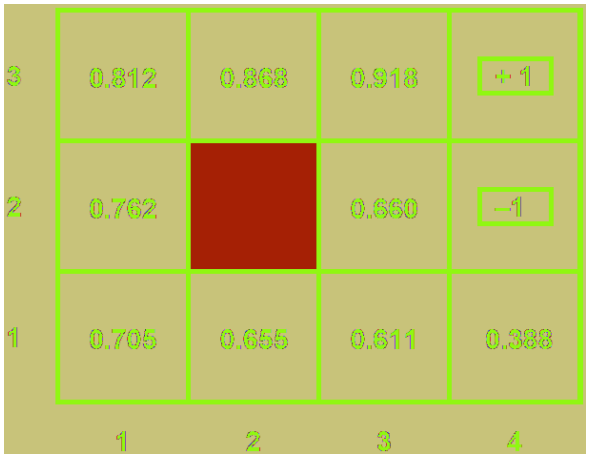
[۱] discounting

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

تکرار مقدار

چگونه یک روش بهینه را پیدا نماییم ؟ ما با محاسبه ی سودمندی مورد انتظار هر وضعیّت و سپس انتخاب عملکردهایی که سودمندی مورد انتظار را بیشینه می کنند شروع می نماییم . در یک مسأله ی ترتیبی ، سودمندی یک وضعیّت ، سودمندی مورد انتظار همه ی رشته های وضعیّتی که از آن دنبال می شوند ، می باشد . این به روشی که π اجرا می شود وابسته می باشد . در اصل ، $U(s)$ سودمندی مورد انتظار یک روش بهینه از وضعیّت s می باشد .

سودمندی وضعیّت ها



توجّه کنید که سودمندی ها دارای بیش ترین مقدار برای وضعیّت های نزدیک به خروجی ۱+ هستند . سودمندی یک وضعیّت ، پاداش فوری برای آن وضعیّت به اضافه ی سود تخفیف داده شده ی مورد انتظار وضعیّت بعدی می باشد و فرض براین است که عامل ، عملکرد بهینه را انتخاب می نماید .

معادله ی بلمن :

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$$

مثال :

$$U(1,1)=-0.04+\gamma\max(0.8U(1,2)+0.1U(2,1)+0.1U(1,1)$$
$$0.9U(1,1)+0.1U(1,2),$$
$$0.9U(1,1)+0.1U(2,1),$$
$$0.8U(2,1)+0.1U(1,2)+0.1U(1,1))$$

برنامه نویسی پویا

معادله ی بلمن ، براساس برنامه نویسی پویا می باشد . در یک گراف انتقال بدون حلقه ، شما می توانید این ها را به صورت بازگشتی توسط معکوس کارکردن از وضعیّت نهایی به وضعیّت های اولیّه حل نمایید . *شما این کار را برای گراف های انتقال با حلقه ، به صورت مستقیم نمی توانید انجام دهید .*

تکرار مقدار

از آنجایی که سودمندی وضعیّت ها وابسته به سودمندی سایر وضعیّت ها می باشد ، چگونه یک راه حل نزدیک را پیدا نماییم ؟ ما می توانیم از یک نگرش تکراری به صورت زیر استفاده نماییم :

- هر وضعیّت را به صورت تصادفی مقداردهی اولیّه نمایید .
- برای یک وضعیّت ، طرف سمت چپ را با توجّه به مقدار های همسایگانش محاسبه نمایید .
- این کار را برای به روز رسانی طرف سمت راست دیگر وضعیّت ها ادامه دهید .
- قانون به روز رسانی به صورت زیر است :

$$U_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$$

الگوریتم تکرار مقدار

انجام دهید [۱]

برای S های موجود در وضعیّت ها

$$U(s) = R(s) + gamma * max\ T(s,a,s')\ U(s')$$

تا هنگامی که [۲]

تمام سودمندی ها با مقدار کم تر از دلتا تغییر نمایند (که دلتا برابر است با : $\delta = error * (1-\gamma)/\gamma$) .

مثال :

1 0.1	2 -0.1	3 0.05	**+1**
4 -0.02		5 0.15	**-1**
6 0.0	7 0.1	8 -0.1	9 0.15

[۱] do

[۲] until

با نسبت دادن سودمندی های تصادفی به هر وضعیّت ، شروع می کنیم ؛ فرض می کنیم γ برابر با ۰.۸ باشد ؛ هزینه ی زمان برابر با ۰.۰۴ - باشد ($R(s) = -0.04$) و خطا برابر با ۰.۰۱ باشد ($\delta = 0.0025$)

بعد از یک بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 0.03	2 -0.02	3 0.62	**+1**
4 0.02		5 0.05	**-1**
6 0.02	7 0.02	8 0.08	9 0.06

بعد از دو بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 -0.02	2 0.35	3 0.65	**+1**
4 -0.02		5 0.28	**-1**
6 -0.02	7 0.01	8 0.02	9 0.01

بعد از سه بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 0.19	2 0.43	3 0.69	**+1**
4 -0.06		5 0.32	**-1**
6 -0.04	7 -0.03	8 0.14	9 -0.03

بعد از چهار بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 0.25	2 0.47	3 0.68	**+1**
4 0.07		5 0.34	**-1**
6 -0.07	7 0.04	8 0.16	9 -0.03

بعد از پنج بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 0.27	2 0.47	3 0.68	**+1**
4 0.13		5 0.34	**-1**
6 0.0	7 0.07	8 0.18	9 -0.02

بعد از شش بار تکرار ، مقادیر تخمین زده شده در اینجا آورده شده است :

1 0.29	2 0.47	3 0.68	**+1**
4 0.15		5 0.34	**-1**
6 0.04	7 0.08	8 0.18	9 -0.01

و در این نقطه ، ما به یک نزدیکی (همگرایی) رسیده ایم .

تشریح مطلب

نقاط قوّت روش تکرار مقدار : برای نزدیک شدن (همگرایی) به راه حل صحیح ، ضمانت شده است و الگوریتم تکرار ، ساده می باشد .

نقاط ضعف روش تکرار مقدار : نزدیک شدن (همگرایی) می تواند به صورت کند باشد ، ما در واقع به تمام این اطّلاعات نیاز نداریم و فقط به این نیاز داریم که بدانیم در هر وضعیّت چه کاری را انجام دهیم .

تکرار روش [۱] (بهبود روش تکرار مقدار)

[۱] Policy iteration

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

به طور مستقیم برای روش های بهینه جستجو می کنیم ؛ به بیان دیگر ، روش ها را با تکـرار بـرای هـر وضعیّت ، به روز می نماییم . این کار دارای دو مرحله می باشد : ارایه ی یک روش ، محاسبه ی سودمندی ها برای هر وضعیّت . به دست آوردن یک روش جدید براساس این سودمندی های جدید .

الگوریتم

سودمندی همه ی وضعیّت ها را برابر با صفر قرار بده

رشته ی روش های تصادفی شاخص گذاری شده توسط وضعیّت را در Pi قرار بده

انجام دهید [۱]

سودمندی هر وضعیّت را برای Pi ارزیابی نمایید و آن را در U قرار دهید .

برای s موجود در وضعیّت ها

عملکردی که سودمندی مورد انتظار را برای آن وضـعیّت ، بیشـینه مـی نمایـد پیـدا نمایید و در a قرار دهید

a را در Pi(s) بریزید

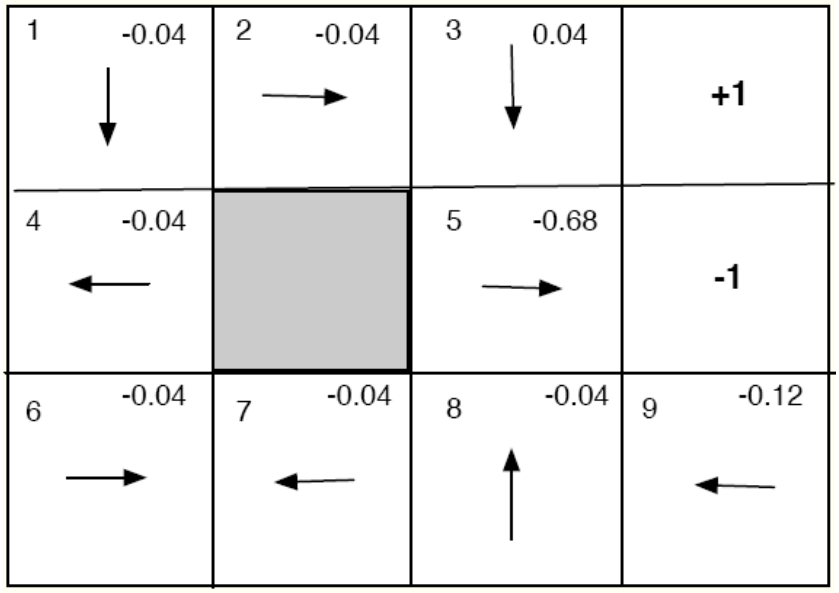
مادامی [۲] که برخی از عملکردها تغییر کردند

مثال :

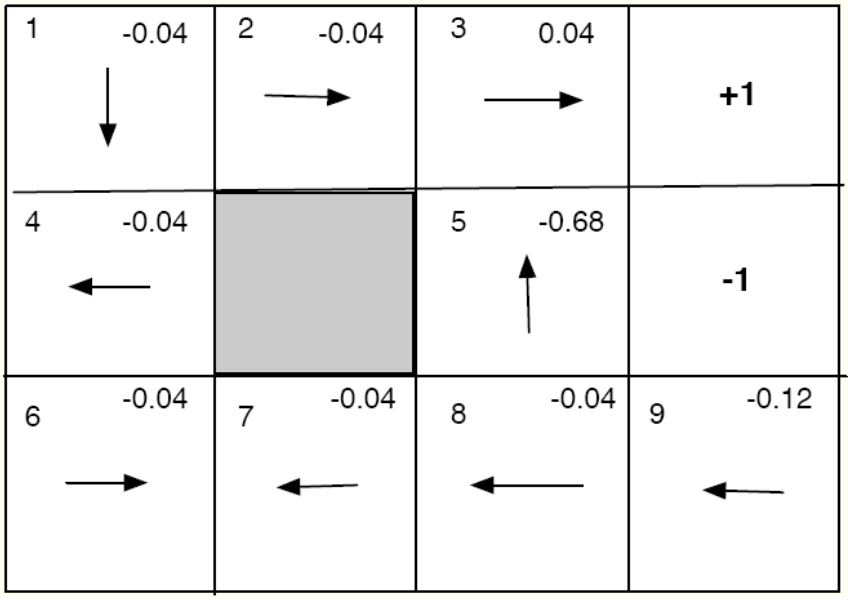
---

[۱] do

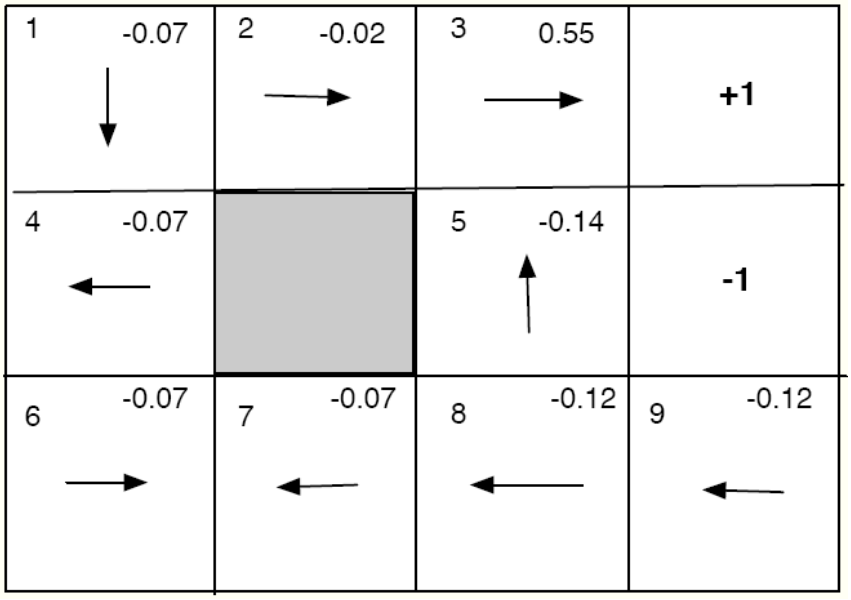
[۲] while



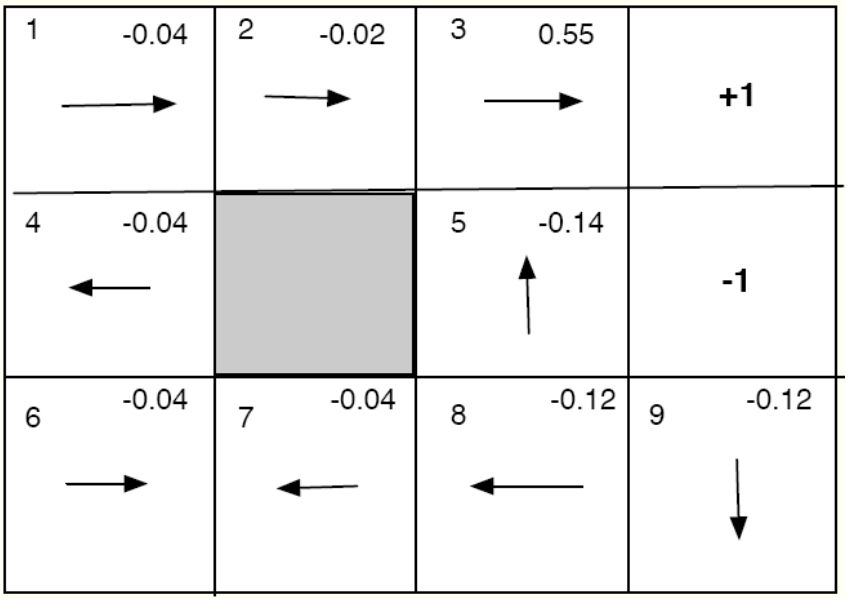
روش های تصادفی را بـه وجـود آوریـد ، سـودمندی وضـعیّت هـا را براسـاس ایـن روش هـا ، مـورد ارزیابی قرار دهید .



روش های بهینه ای که این سودمندی ها را ارایه می نمایند ، انتخاب نمایید .



براساس روش های جدید به دست آمده ، سودمندی های جدید را برای هر وضعیّت ، تخمین بزنید .



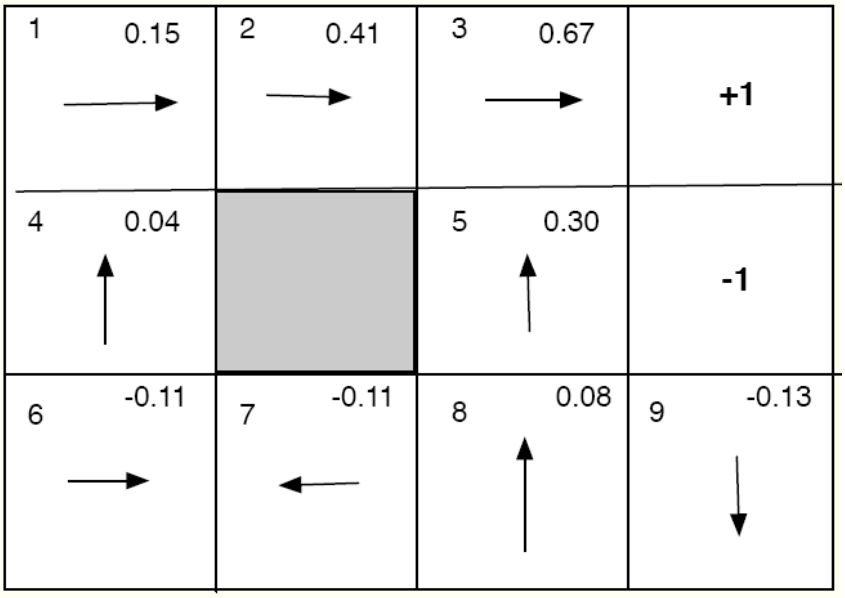
براساس این سودمندی های جدید ، روش های بهینه را انتخاب نمایید .

1 -0.06 →	2 0.31 →	3 0.63 →	+1
4 -0.09 ←		5 0.22 ↑	-1
6 -0.09 →	7 -0.09 ←	8 -0.10 ←	9 -0.12 ↓

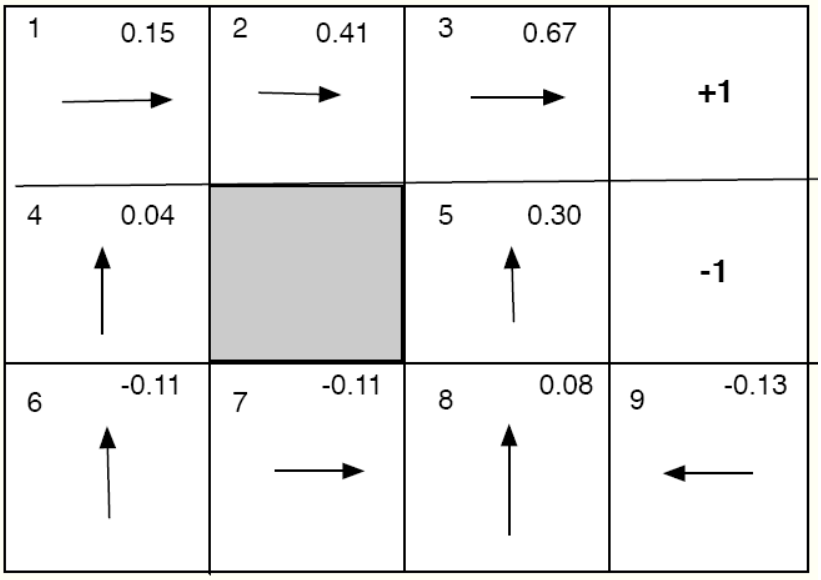
از این روش های جدید برای تخمین مجدّد سودمندی ها استفاده نمایید .

1 -0.06 →	2 0.31 →	3 0.63 →	+1
4 -0.04 ↓		5 0.22 ↑	-1
6 -0.04 →	7 -0.04 ←	8 -0.08 ↑	9 -0.12 ↓

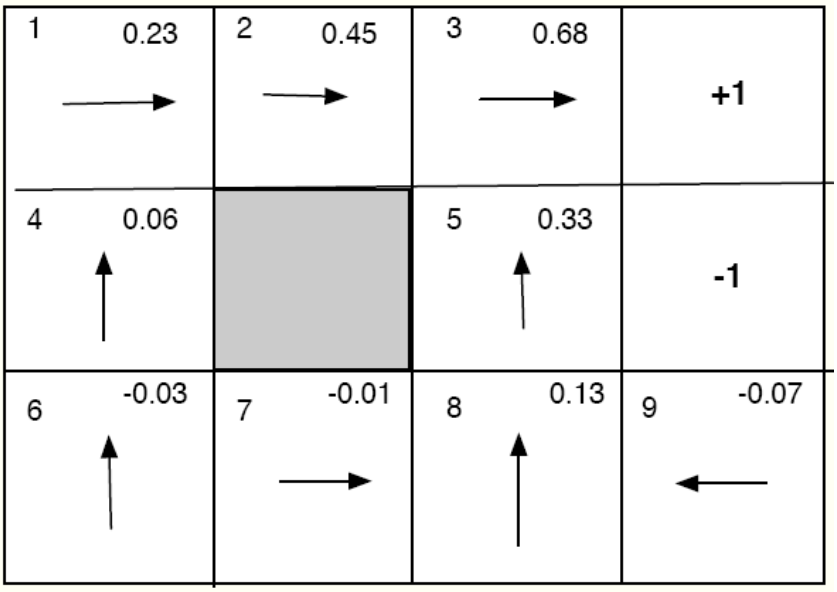
و از این تخمین های سودمندی جدید برای به دست آوردن روش های بهینه استفاده نمایید .



مجدّداً ، از روش های به دست آمده برای تخمین مجدّد سودمندی ها استفاده نمایید .



و سپس از سودمندی ها برای به روز کردن روش های بهینه استفاده نمایید .



و دوباره سودمندی های تخمین ها را براساس روش جدید ، به روز نمایید . اگر ما روش را براساس تخمین های جدید به روز کردیم و دیدیم که تغییری به وجود نمی آید ، در این صورت ، کار به پایان رسیده است . حُسن تکرار روش در این است که نزدیک شدن یا همگرایی سریع تر می باشد .

مترجم: سهراب جلوه گر

ویرایش دوّم ، بهار ۱۳۸۸

فصل بیست و دوّم

سیستم های طبقه بندی کننده [۱]

[۱] Classifier Systems

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

سیستم های طبقه بندی کننده

الگوریتم های ژنتیکی برای مسایل بهینه سازی ، خیلی خوب کار می کنند و داریم $f(x_1, x_2, x_3, ..., x_n) = R$. در مورد مسایلی که کم تر به صورت خوب تعریف شده اند چطور ؟ آیا ما می توانیم از این ایده ها برای ساختن یک عامل جاروبرقی استفاده نماییم ؟ بله ، ولی ما نیاز داریم که کمی بادقّت تر باشیم . *طبقه بندی* ، پردازه ی انتساب یک ورودی به یکی از کلاس های چندگانه می باشد .

سیستم های تولید [۱]

یک سیستم تولید ، از یک مجموعه از تولیدات (قوانین) ، حافظه ای که در آن واقعیّات قرار می گیرند و یک الگوریتم که با استفاده از روش زنجیره ی مستقیم اجرا می شود و واقعیّات جدید را با استفاده از قبلی ها به وجود می آورد تشکیل شده است . یک قانون ، زمانی اجرا می شود که مجموعه ای از شرایط ، که در حافظه هستند ، برقرار باشند . [۲] سیستم های تولید ، یک روش رایج در هوش مصنوعی هستند . و در آن ها

---

[۱] Production system

[۲] Babylon / FOLDOC

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

از اگر – آن گاه یا قانون های شرط – عملکرد استفاده می شود . مثلاً در دنیای جاروبرقی داریم : در صورتی که خانه ی [۱و۱] جارو نشده باشد و در خانه ی مجاور یا همسایه باشد ، آن گاه به [۱،۱] برو . یک سیستم تولید ، از تعداد زیادی قانون تشکیل شده است . برخی از قوانین ممکن است باعث عمل کردن قانون های دیگر شوند . در سیستم های تجاری ، مسأله این است که وقتی برای یک مورد ، بیش از یک قانون داریم ، چه کار کنیم [و کدام را اجرا نماییم] ، ما بعداً به این مورد خواهیم پرداخت .

طبقه بندی کننده ها

الگوریتم های ژنتیکی ، به صورت بالقوّه ، به ما یک روش برای استنتاج و انتخاب قانون ها را ارایه می دهند و می گویند ، قانونی که دارای بیش ترین شایستگی می باشد را انتخاب نمایید . توسّط ۰،۱ و * قانون ها را به صورت یک رشته ی بیتی کد نمایید . بیت های وضعیّت ، همان بیت های عملکرد می باشند . مشکلاتی که در مورد وجود دارند عبارتند از اینکه : چگونه شایستگی را به یک قانون انتساب دهیم ؟ و چگونه عملکرد برخط بالا را نگهداری نماییم ؟

اجزای یک سیستم طبقه بندی

یک سیستم طبقه بندی دارای سه جزء می باشد : یک قانون و سیستم پیام ، یک سیستم انتساب اعتبار و یک الگوریتم ژنتیکی برای تولید قانون های جدید .

قانون و سیستم پیام - حسگرهای عامل ، اطّلاعات را که به صورت یک رشته ی بیتی کد شده اند دریافت می نمایند ؛ این اطّلاعات ، پیامی از محیط می باشد . این پیام طبقه بندی کننده ها (قانون ها) را با تطبیق شرط ها فعّال می نماید . طبقه بندی کننده ها پیام هایشان را به لیست پیام ارسال می نمایند ؛ این پیام ها ممکن است دیگر طبقه بندی کننده ها یا عمل کننده های عامل را فعّال نمایند .

مثال – فرض کنید سیستم ما دارای طبقه بندی کننده های زیر می باشد :

1. 01## : 0000
2. 00#0 : 1100
3. 11## : 1000
4. ##00 : 0001

اگر پیام ۰۱۱۱ از محیط دریافت می شود . در ابتدا ، قانون یک اجرا می شود ، ۰۰۰۰ در لیست پیام قرار می گیرد . سپس ، قانون های دو و چهار اجرا می شود ، ۱۱۰۰ و ۰۰۰۱ در لیست پیام قرار می گیرند . بعد ، قانون سه اجرا می شود ، ۱۰۰۰ در لیست پیام قرار می گیرد ، که با قانون شماره ی ۴ مطابقت دارد که پیامش در لیست پیام ها قرار دارد . حالا چند پیام در لیست پیام قرار دارد – کدام به عمل کننده ها فرستاده می شود ؟

دسته سطل [۱] – الگوریتم دسته سطل به قانون ها برای پیشنهاد اجرا براساس عملکرد قبلی اجازه می دهد . وقتی که یک قانون برقرار می شود ، در یک " مزایده [۲] " شرکت می نماید . هر قانون براساس عملکرد قبلی ، دارای یک قوّت می باشد ؛ یک قانون یک نسبت از قدرت را داراست . بالاترین قانون های شرکت کننده برنده می شوند و این پیشنهاد به طبقه بندی کننده (ها) ای که آن را فعّال کرده (اند) فرستاده می شود .

مثال : اگر به صورت اوّلیّه (t=0) داشته باشیم : M = ۰۱۱۱

1) 01## : 0000 S=200 B=20
2) 00#0 : 1100 S=200
3) 11## : 1000 S=200
4) ##00 : 0001 S=200

---

[۱] Bucket Brigade : زنجیره ای از افراد که با دست به دست کردن سطل هایی از آب سعی می کنند آتش را خاموش نمایند (Babylon / Merriam – Webster Collegiate ® Dictionary) .

[۲] auction

محیط S=0

در t = 1:

1. 01## : 0000 S=180 0000
2. 00#0 : 1100 S=200 B=20
3. 11## : 1000 S=200
4. ##00 : 0001 S=200 B=20

محیط S=20

در t = 2:

1. 01## : 0000 S=220 1100
2. 00#0 : 1100 S=180 0001
3. 11## : 1000 S=200 B=20
4. ##00 : 0001 S=180 B=18

محیط S=20

در t = 3:

1. 01## : 0000 S=220 1000
2. 00#0 : 1100 S=218 0001
3. 11## : 1000 S=180
4. ##00 : 0001 S=162 B=16