



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ویرایش دوم

بهار ۱۳۸۸

هوش مصنوعی

مترجم: سهراب جلوه گر

(کارشناس نرم افزار رایانه)

با استفاده از متون انگلیسی موجود در اینترنت قابل استفاده برای: دانشجویان دوره‌ی کارشناسی نرم افزار رایانه و کارشناسی ارشد گرایش‌های هوش مصنوعی و رباتیک و کلیه‌ی علاقه مندان

پست الکترونیکی مترجم: sohjel@yahoo.com
وبلاگ مترجم: <http://sohjel.blogfa.com>

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Second Edition Spring 2009

Persian Artificial Intelligence Ebook

Source: English subjects exist in internet

Translated by:
Sohrab Jelvehgar, computer software engineer

Email: sohjel@yahoo.com
Weblog: <http://sohjel.blogfa.com>

۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برای آگاهی یافتن از انتشار ویرایش های جدیدتر کتاب هوش مصنوعی
اینجانب به وبلاگ های زیر مراجعه نمایید:

<http://sohrabejelvehgar.blogspot.com>

<http://sohrab-e-jelvehgar.blogspot.com>

<http://sohjel.blogfa.com>

جویای کار هستم

در انتظار سرمایه گذار یا ناشر سرمایه گذار برای چاپ
ویرایش های بعدی این کتاب هستم.

صاحبان محترم مشاغل ، علاقه مندان به سرمایه گذاری و ناشرین گرامی ،
می توانند پیشنهادهای خود را از طریق پست الکترونیکی
برای اینجانب ارسال نمایند . sohrabejelvehgar@yahoo.com

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نظرات ، پیشنهادات و انتقادات

لطفاً جهت بهتر شدن این کتاب؛ نظرات ، پیشنهادات و انتقادات سازنده‌ی خود را از طریق ایمیل‌های زیر برای اینجانب ارسال نمایید.

sohrabejelvehgar@hotmail.com

sohrabejelvehgar@yahoo.com

sohjel@yahoo.com

با احترام ، سهراب جلوه گر ، کارشناس نرم افزار رایانه

شیراز - ایران - بهار ۱۳۸۸

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فهرست

فصل اول - هوش مصنوعی

فصل دوم - عامل های هوشمند

فصل سوم - حل مسئله و جستجو

فصل چهارم - جستجوی آگاهانه (مکاشفه ای)

فصل پنجم - الگوریتم های جستجوی محلی

فصل ششم - مسایل اراضی محدودیت

فصل هفتم - مسایل اراضی محدودیت و برنامه نویسی اراضی محدودیت

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل هشتم - تئوری بازی ها

فصل نهم - عامل های منطقی

فصل دهم - منطق مرتبه ای اوّل به بیان ساده

فصل یازدهم - منطق مرتبه ای اوّل

فصل دوازدهم - استنتاج در منطق مرتبه ای اوّل

فصل سیزدهم - نامعلومی (عدم قطعیت)

فصل چهاردهم - شبکه های بیزی

فصل پانزدهم - استنتاج در شبکه های بیزی

فصل شانزدهم - شناخت سخن یا سخن شناسی (به طور خلاصه)

فصل هفدهم - تصمیم گیری های عاقلانه (تیوری تصمیم گیری)

فصل هیجدهم - شبکه های عصبی

فصل نوزدهم - الگوریتم های ژنتیکی

فصل بیستم - سیستم های خبره

فصل بیست و یکم - پردازش های تصمیم گیری مارکوف

فصل بیست و دوم - سیستم های طبقه بندی کننده

فصل بیست و سوم - درخت های تصمیم گیری

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فصل بیست و چهارم - یادگیری Q ای

فصل بیست و پنجم - برنامه ریزی

فهرست برخی از منابع و مآخذ



Λ

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل اول

هوش مصنوعی

^۱- تصویر، متعلق به جان مکارتی، ملقب به پدر هوش مصنوعی است.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مقدمه

هوش مصنوعی^۱: ساخت ماشین هایی که کارهایی را انجام می دهند که آن کارها معمولاً با استفاده از هوش انسان انجام می شوند؛ مثل ، ترجمه‌ی یک زبان به زبان دیگر .^۲ به عنوان تعریفی دیگر ، دانش و مهندسی ساخت ماشین های هوشمند و مخصوصاً برنامه های کامپیوترا هوشمند می باشد . هوش مصنوعی ، وابسته به کامپیوتراها مورد استفاده برای فهم هوش انسانی می باشد ، ولی لازم نیست که خودش را به روش هایی که به صورت زیستی^۳ قابل مشاهده اند محدود نماید .

Artificial Intelligence (AI)^۱

Babylon / Concise Oxford English Dictionary^۴

biologically^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هوش ، توانایی به دست آوردن و به کار گرفتن دانش و مهارت ها می باشد .^۱ انواع و درجه های هوش در افراد و حیوانات و برخی از ماشین ها متنوع است . هنوز یک تعریف جامع از هوش که به ارتباط آن با هوش بشری وابسته نباشد وجود ندارد ؛ مسئله این است که ما هنوز نمی دانیم که در حالت کلی چه انواعی از روال های محاسباتی را برای فراخوانی هوش می خواهیم استفاده نماییم . ما برخی از مکانیزم های هوش را می دانیم و نه بیش تر . کسی نمی تواند پرسد که "آیا این ماشین ، هوشمند است یا نه ؟" زیرا هوش شامل طرز کارها می باشد و هوش مصنوعی به چگونگی ساخت کامپیوترها و نه چیزهایی بیش تر پی برده است . اگر انجام یک کار فقط نیازمند مکانیزم هایی که امروزه به سادگی قابل فهم هستند می باشد و برنامه های کامپیوتری کارهای خیلی موثر را در این موارد می توانند انجام دهند این جور برنامه ها باید " تاحدودی هوشمند " نامیده شوند .

بعضی اوقات ، اما نه همیشه هوش مصنوعی در مورد شبیه سازی هوش بشری می باشد . از یک طرف ما می توانیم چیزهایی را در مورد چگونگی ساخت ماشین های حل مسایل با مشاهده ی دیگر افراد و یا فقط با مشاهده ی رفتارهای خودمان یاد بگیریم . از طرف دیگر ، بیش تر کارها در هوش مصنوعی شامل مطالعه ی مسایل جهان که برای هوش ارایه می شود می باشد و بیش تر از مطالعه ی افراد و یا حیوانات می باشد . محققان هوش مصنوعی برای استفاده از متدهایی (رفتارهایی) که در افراد مشاهده نمی شوند یا شامل محاسبه کننده هایی که به مراتب بیش تر از آنچه که افراد می توانند انجام دهند ، آزاد هستند .

بهره ی هوشی ^۲ ، براساس میزان یا نرخی است که هوش در بچه ها توسعه پیدا می نماید . برای بزرگسالان از روش مناسب دیگری استفاده می نماییم . بهره ی هوشی به خوبی با اندازه های متفاوت موقیت یا عدم موقیت در زندگی ارتباط برقرار می نماید ، اما به طور کمی ، ساختن کامپیوترهایی که بتوانند امتیاز بالا را در تست های بهره ی هوشی انجام دهند به مفید بودن آن ها بستگی دارد .

Babylon / Concise Oxford English Dictionary^۱
Intelligence Quotient^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



آرتور آر. جنسن^۱، یک محقق خبره (ماهر) در هوش انسانی " یک فرضیه‌ی ابتکاری " که همه‌ی انسان‌ها دارای مکانیزم‌های عقلانی شبیه هستند و تفاوت‌ها در هوش وابسته به مکانیزم‌های کمی و وضعیت‌های فیزیولوژیکی هستند را پیشنهاد می‌کند " . که ما آن‌ها را به صورت سرعت^۲، حافظه‌ی کوتاه مدت^۳، توانایی به صورت صحیح و حافظه‌های بازیافتی طولانی مدت^۴ تقسیم بندی می‌کنیم . نظر جنسن در مورد هوش انسانی درست است ولی در موقعیت فعلی این نظر در مورد هوش مصنوعی ، معکوس می‌باشد . برنامه‌های کامپیوتری دارای سرعت و حافظه‌ی زیادی می‌باشند اما توانایی عقلانی آن‌ها با توانایی عقلانی طراحان برنامه‌ها بستگی دارد . به احتمال قوی ، سازماندهی مکانیزم‌های عقلانی برای هوش مصنوعی می‌تواند با سازماندهی مکانیزم‌های عقلانی برای افراد ، متفاوت باشد . هرگاه که افراد برخی از کارها را بهتر از کامپیوترها انجام می‌دهند و یا کامپیوترها تعداد زیادی محاسبه را برای انجام کار به خوبی انسان‌ها انجام می‌دهند ، این نشان می‌دهد که طراحان برنامه فهم مکانیزم‌های عقلانی لازم برای انجام کار را به طور مناسب نداشته‌اند .



بعد از جنگ جهانی دوم تعدادی از افراد به صورت مستقل کار بر روی ماشین‌های هوشمند را شروع کردند . ریاضیدان انگلیسی ، آلن تورینگ^۵ شاید اوّلین آن‌ها باشد . وی یک سخنرانی را در مورد هوش مصنوعی در سال ۱۹۴۷ میلادی ارایه نمود . همچنین وی شاید اوّلین کسی باشد که گفت برنامه نویسی کامپیوترها برای هوش مصنوعی نسبت به ساخت ماشین‌ها بهتر می‌باشد . تا اواخر سال ۱۹۵۰ ، تعداد زیادی محقق در مورد هوش مصنوعی

Arthur R. Jensen^۱

speed^۲

short term memory^۳

retrieveable long term memory^۴

Alan Turing^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

وجود داشت و بیش تر آن ها اساس کار خود را بر مبنای برنامه نویسی کامپیوترها گذاشته بودند.

برخی از محققان گفته اند که می خواهند افکار انسان را با استفاده از هوش مصنوعی در کامپیوتر پیاده سازی نمایند؛ فکر بشری دارای موارد زیادی می باشد و هر کسی به طور کامل نمی تواند همه ای آن ها را تقلید نماید.

اهداف هوش مصنوعی در سطح انسانی است؛ نهایت تلاش این است که برنامه های کامپیوتری که می توانند مسایل را حل کنند و به اهداف دسترسی پیدا کنند را در جهان، به خوبی انسان ها بسازند. تعداد کمی از افراد فکر می کنند که سطح هوش بشری با نوشتمن برنامه های طولانی که هم اکنون در حال نوشتمن و سرهم بندی پایگاه های دانش^۱ واقعیات، با استفاده از زبان هایی که اکنون برای بیان دانش استفاده می شوند، قابل دسترسی می باشند. به هر حال، بیش تر محققان هوش مصنوعی تصوّر می کنند که ایده های بنیادین جدید مورد نیازند، و بنابراین قابل پیش بینی نیست که سطح هوش انسانی قابل دسترسی باشد.

کامپیوترها ابزار مناسبی برای پیاده سازی هوش مصنوعی هستند؛ کامپیوترها برای شبیه سازی هر نوع از ماشین ها می توانند برنامه ریزی شوند. برخی از افراد فکر می کنند که کامپیوترهای به مراتب سریع تر برای ایده های جدید لازم هستند. نظر شخصی جان مک کارتی^۲ این است که کامپیوترهای سی سال پیش هم در صورتی که ما بدانیم آن ها را چگونه برنامه ریزی نماییم به اندازه ای کافی سریع می باشند. البته به طور مجزا از بلند پروازی های محققان هوش مصنوعی، کامپیوترها روند سریع تر شدن را دنبال می کنند.

knowledge base^۱

^۲ یکی از استادان دانشگاه استنفورد ایالات متحده ای آمریکا

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فرزند ماشینی^۱ می تواند با خواندن و آموزش و از راه تجربه یاد بگیرد و شروع بحث آن از سال ۱۹۴۰ میلادی بوده است. اما برنامه های هوش مصنوعی هنوز به آن حد نرسیده اند که قادر باشند به اندازه ای که یک بچه از تجربه‌ی محیط یاد می گیرد یاد بگیرند.

یک سیستم هوش مصنوعی می تواند خودش را به سطح های بالاتر از هوش برساند. اما ، ما هم اکنون در مرحله‌ای از هوش مصنوعی نمی باشیم که این کار بتواند انجام شود.

توضیحی در مورد چند بازی

بازی شطرنج^۲ - بازی ای است دو نفره که در آن هر نفر شانزده مهره را با استفاده از قوانینی ثابت در یک صفحه‌ی شطرنجی^۳ حرکت می دهد و تلاش می کند که مهره‌ی شاه حریف را کیش و مات^۴ نماید^۵. برنامه های بازی شطرنج در حال حاضر در سطح استادی اجرا می شوند ، اما آن ها این کار را با مکانیزم های محدود شده نسبت به انسان و با انجام تعداد زیادی محاسبات برای فهم راه ، انجام می دهند. ما این مکانیزم ها را بهتر می فهمیم و می توانیم برنامه های شطرنج را در سطح انسانی بسازیم طوری که محاسبه های کم تری را نسبت به برنامه های فعلی انجام دهند. نمونه ای از صفحه‌ی این بازی را به همراه مهره های آن در شکل زیر می توانید بینید.

child machine^۱

chess^۲

checkerboard^۳

checkmate^۴

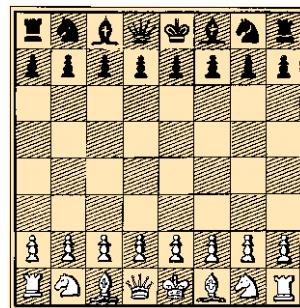
Babylon / Merriam-Webster^۵

مترجم: سهراب جلوه گر

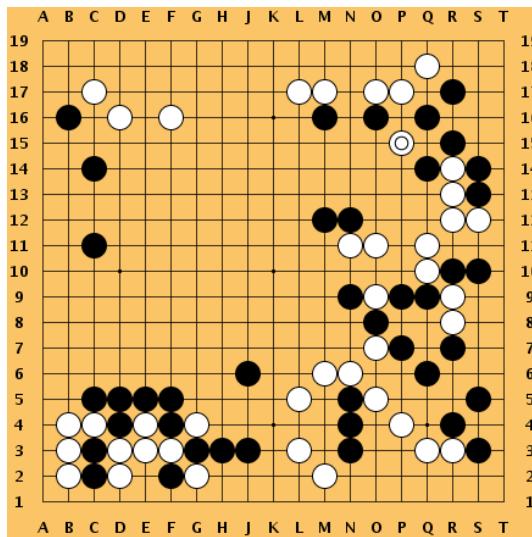
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



بازی Go – بازی دو نفره‌ی چینی و ژاپنی Go، بازی ای روی مقوایی دارای نوزده خط افقی و نوزده خط عمودی است که در آن معمولاً یکی از بازی کنندگان دارای مهره‌های سفید رنگ و یکی دیگر از بازی کنندگان دارای مهره‌های سیاه رنگ می‌باشند. در زیر تصویری از صفحه‌ی این بازی به همراه مهره‌های آن را مشاهده می‌نمایید.



برخی از افراد می‌گویند که هوش مصنوعی ایده‌ی بدی می‌باشد؛ فیلسوفی به نام جان سیرل^۱ می‌گوید که ایده‌ی ماشینی غیر بیولوژیکی که می‌تواند هوشمند شود، ایده‌ای نقض کننده می‌باشد. وی

John Searle (1980)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

استدلال اطاق چینی^۱ را پیشنهاد می کند^۲. فیلسوفی به نام هیوبرت دریفس^۳ می گوید که هوش مصنوعی غیرممکن می باشد . دانشمند علوم کامپیوترا بی نام جZF ویزناوم^۴ می گوید که ایده ی هوش مصنوعی ، ضد بشری و غیراخلاقی می باشد . برخی دیگر از افراد می گویند که چون که هوش مصنوعی تا کنون به اهداف خود نرسیده است پس غیر ممکن می باشد . سایر افراد از این که می بینند کمپانی های سرمایه گذاری کننده و رشکست می شوند مأیوس می شوند .

استدلال اطاق چینی : یک گمان آزمایشی طراحی شده توسط جان سرل برای کم ارزش کردن ادعاهای مطرح شده توسط هوش مصنوعی قوى^۵ و همچنین کارکردگرایي^۶ می باشد .^۷ از این استدلال همچنین برای نشان دادن این که مغز ، یک کامپیوتر نمی باشد و آزمایش تورینگ^۸ ناقص است استفاده شده است .^۹

هوش مصنوعی ضعیف^۹ : اوّلین هدف هوش مصنوعی (هوش مصنوعی ضعیف) ساختن چیزها (موجودیت ها) ی هوشمند می باشد .

هوش مصنوعی قوى : فهمیدن چیزها (موجودیت ها) ی هوشمند و شاید حتی فهمیدن و مهندسی هوش انسان می باشد .^۱ دیدی است که می گوید مغز بشر یک ابزار محاسباتی می باشد و

Chinese room argument^۱

<http://www-formal.stanford.edu/jmc/chinese.html>^۲

Hubert Dreyfus^۳

Joseph Weizenbaum^۴

Strong AI^۵

functionalism^۶

en.wikipedia.org/wiki/Chinese_room_argument^۷

Babylon / Dictionary of Philosophy of Mind^۸

weak artificial intelligence^۹

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



کامپیوترها به طور کلی قادرند که فکر کنند^۱. به عنوان تعریفی دیگر ، هوش مصنوعی قوی یک شکل فرضی از هوش مصنوعی است که می تواند به درستی استدلال نماید و مسایل را حل کند ؟ هوش مصنوعی قوی می تواند در ک نماید یا خودآگاه^۲ باشد اما ممکن است پردازش های فکری شبیه بشر را داشته باشد یا نداشته باشد.^۳

تیوری تجزیه پذیری^۴ و پیچیدگی محاسباتی^۵ روش هایی برای هوش مصنوعی نمی باشند [توجه کنید که افراد عامی و مبتدیان در علم کامپیوتر نمی توانند در این موارد اظهار نظر کنند ، این ها کاملاً شاخه های منطقی ریاضی و علم کامپیوتر می باشند و جواب این موارد باید تا حدودی تکنیکی باشد .] ؛ این تیوری ها مورد استفاده قرار می گیرند ولی مسایل اساسی هوش مصنوعی را جوابگو نمی باشند . در سال ۱۹۳۰ منطقدانان ریاضیات و مخصوصاً کورت گودل^۶ و آلن تورینگ ثابت کردند که الگوریتم هایی برای ضمانت این که همه ای مسایل را در دامنه های مهم ریاضی بتوانند حل کنند وجود ندارد . جمله ای از منطق مرتبه ای اوّل^۷ یک مثال می باشد و یک معادله ای چند متغیر مثال درست دیگری می باشد . انسان ها همه ای مسایل را همیشه در این زمینه ها حل کرده اند و آن را به صورت یک استدلال مطرح کرده اند (معمولاً با برخی اضافات) که کامپیوترها ذاتاً قادر به انجام کارهایی که افراد می توانند انجام دهند نمی باشند

^۱ پروفسور گرینوالد (Professor Greenwald) ، استاد دانشگاه برون (Brown) ایالات متحده ای آمریکا

^۲ www.ucd.ie/philosop/documents/2.%20definitions%20of%20some%20key%20terms.htm

^۳ self-aware

^۴ en.wikipedia.org/wiki/Strong_AI

^۵ computability theory – بحثی در علم نظری کامپیوتر می باشد که می گوید مسایل می توانند توسط هر کامپیوتری حل شوند . Babylon / FOLDOC

^۶ computational complexity – هزینه ای حل یک مسأله در محاسبات علمی گسترشده ، که با استفاده از تعداد عملیات لازم به علاوه ای مقدار حافظه ای استفاده شده برای مسأله و ترتیبی که مسأله حل می شود اندازه گیری می شود . (Britannica Concise Encyclopedia

^۷ Kurt Godel

^۸ First Order Logic(FOL)

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

، راجر پنروز^۱ این مطلب را ادعا می کند . به هر حال ، افراد حل مسایل دلخواه را در این موارد یا هر کدام از آن ها نمی توانند ضمانت کنند . در دهه ۱۹۶۰ دانشمندان علوم کامپیوتر و مخصوصاً استیو کوک^۲ و ریچارد کارپ^۳ تیوری مسایل با دامنه NP - **کامل**^۴ را توسعه دادند . مسایل در این دامنه ها قابل حل می باشند ، اما به نظر می رسد که زمان برای حل این مسایل با افزایش اندازه به صورت نمایی افزایش یابد . جملات حساب گزاره ای یک مثال پایه ای برای مسایل با دامنه NP - **کامل** می باشند . انسان ها اغلب مسایل را در مورد مسایل NP - **کامل** در زمان هایی به مراتب کوتاه تر از آن چه که توسط الگوریتم های عمومی ضمانت می شوند حل می کنند ، اما در حالت کلی نمی توانند این مسایل را به سرعت حل کنند . چه چیزی برای هوش مصنوعی مهم است که الگوریتم هایی به اندازه ای افراد توانمند ، برای حل مسایل باید داشته باشد . شناسایی برای این که کدام زیر دامنه ها برای الگوریتم ها وجود دارد مهم است ، اما تعداد زیادی از حل کنندگان مسایل هوش مصنوعی به آسانی با زیر دامنه ها سازگار نمی باشند . تیوری پیچیدگی کلاس های عمومی ، مسایل پیچیدگی محاسباتی نامیده می شود . تا کنون این تیوری با هوش مصنوعی به اندازه ای که انتظار می رفته است همکاری (تعامل) نداشته است . موققیت در مسایل حل شده توسط بشر و توسط برنامه های هوش مصنوعی به نظر می رسد که وابسته به خصوصیات مسایل و متدهای حل مسئله ای که نه توسط محققان و نه توسط اجتماع هوش مصنوعی به دقت قبل تعريف باشند وابسته باشد . تیوری پیچیدگی الگوریتمی که به نحوی توسط سولومونوف^۵ ، کولموگورو夫^۶ و چاپین^۷ (به طور مجزا از یکدیگر) توسعه داده شد نیز وابسته می باشند . پیچیدگی شیء سمبولیک را به وسیله ای کوتاه ترین برنامه ای که آن را تولید می کند تعريف کرد . اثبات این که برنامه ای کاندید ، کوتاه ترین یا نزدیک به کوتاه ترین

Roger Penrose^۱

Steve Cook^۲

Richard Karp^۳

NP-complete^۴

Solomonoff^۵

Kolmogorov^۶

Chaitin^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

است یک مسئله‌ی غیر قابل حل می‌باشد، حتی زمانی که شما نمی‌توانید ثابت کنید که برنامه کوتاه‌ترین می‌باشد باید اشیاء ارایه شده توسط برنامه‌های کوچک که آن‌ها تولید می‌کنند روشن شود.

مسایل NP-کامل – یک نوع از مسایل محاسباتی هستند که هیچ الگوریتم مناسبی برای حل آن‌ها پیدا نشده است. مسئله‌ی فروشنده‌ی دوره گرد^۱ مثالی از این نوع مسایل می‌باشد.^۲ در تیوری پیچیدگی، مسایل NP-کامل، سخت‌ترین مسایل در NP هستند، در صورتی که شما بتوانید یک راه برای حل یک مسئله‌ی NP-کامل به سرعت پیدا نمایید، آن‌گاه شما می‌توانید این الگوریتم را برای حل تمام مسایل NP به سرعت استفاده نمایید.

شاخه‌های هوش مصنوعی

در زیر یک لیست از شاخه‌های هوش مصنوعی آمده است، اماً به یقین برخی از شاخه‌ها وجود ندارند، زیرا هنوز کسی آن‌ها را نشناخته است. برخی از این‌ها ممکن است به صورت موضوعات یا افکاری بیش از شاخه‌های کامل ملاحظه شوند.

هوش مصنوعی منطقی^۳ – این که یک برنامه در حالت کلی، واقعیات وضعیت معینی که در آن باید عمل کند را می‌داند و اهدافی که توسط عبارات بعضاً با زبان منطقی ریاضی ارایه می‌شوند را می‌داند.

– یک مسئله‌ی بهینه‌سازی در تیوری گراف‌ها است که در آن گره‌ها (شهرها) یک گراف به وسیله‌ی خط‌های مستقیم (یال) به هم وصل شده است و طول هر خط به فاصله‌ی دو شهر از هم بستگی دارد به عبارت دیگر هر چقدر طول خط بیش تر باشد فاصله‌ی دو شهر بیش تر خواهد بود. حال مسئله‌این است که مسیری مناسب را پیدا کنیم که از هر شهر یکبار بگذرد و در ضمن کوتاه‌ترین مسیر هم باشد. (Babylon / Britannica.com)

Babylon / Britannica.com^۴

logical AI^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برنامه در مورد عملی که باید انجام دهد با استنتاج عملکردهای معین که برای به دست آوردن اهداف، مناسب هستند تصمیم گیری می نماید. اوّلین مقاله‌ی پیشنهاد شده در این مورد McC89 McC59 بود. McC96b برخی از افکار در برگیرنده‌ی هوش مصنوعی منطقی را لیست می کند. جدیدتر می باشد. Sha97 هم یک متن مهم است.

جستجو^۱ - برنامه‌های هوش مصنوعی معمولاً تعداد زیادی از احتمالات را بررسی می کنند، به عنوان مثال حرکت‌های درون یک بازی شطرنج.

الگو شناسی یا شناخت الگو^۲ - در علم کامپیوترا، تشخیص داده‌های ورودی مثل سخن، تصویرها و رشته‌های متنی با شناخت و تشریح خصوصیات و تشخیص ارتباط‌های میان آن‌ها است.^۳

نمایش^۴ - واقعیات یک محیط باید به طریقی نمایش داده شوند. در این مورد معمولاً از زبان‌های منطقی ریاضی استفاده می شود.

استنتاج^۵ - از برخی از واقعیات، دیگر واقعیات می تواند استنتاج شود. استنتاج منطقی ریاضی برای برخی از اهداف، کافی می باشد، اماً متدهای استنتاج جدید غیر یکنواخت از سال ۱۹۷۰ به منطق اضافه شده‌اند. ساده‌ترین نوع استدلال غیر یکنواخت^۶، استدلال پیش فرض است که در آن یک نتیجه گیری به صورت پیش فرض استنتاج می شود، اماً نتیجه گیری می تواند در صورتی که مدرک (دلیل) عوض شود، عوض بشود. برای مثال، زمانی که ما از یک پرنده حرف می زیم، ما نتیجه می گیریم که می تواند پرواز

search^۱

pattern recognition^۲

Babylon / Britannica Concise Encyclopedia^۳

representation^۴

inference^۵

non-monotonic^۶

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

کند، اما این استنتاج زمانی که ما در مورد پنگوین حرف می‌زنیم می‌تواند عوض شود. احتمال این وجود دارد که یک نتیجه گیری که خصوصیات غیر یکنواخت را از استدلال تولید می‌کند عوض شود. معمولاً استدلال منطقی در مجموعه‌ای از استنتاج‌ها که می‌توانند از یک مجموعه مقدمات گرفته شوند، یک تابع افزایشی یکنواخت از مقدمات می‌باشد.

دانش عقل سلیم و استدلال^۱ - محیطی است که در آن هوش مصنوعی از سطح انسانی خیلی دور می‌باشد، باوجود این واقعیت که از سال ۱۹۵۰ روی آن تحقیق شده است. به عنوان مثال، در توسعه‌ی سیستم‌های استدلال غیر یکنواخت و تیوری‌های عملکرد، هنوز ایده‌های بیشتری مورد نیاز می‌باشد. سیستم Cyc شامل مجموعه‌ای بزرگ از واقعیات عقل سلیم می‌باشد.

یادگیری از تجربه^۲ - برنامه‌ها این کار را انجام می‌دهند. براساس پیوند گرایی^۳ و شبکه‌های عصبی^۴ به هوش مصنوعی نزدیک می‌شوند و در هوش مصنوعی تخصصی می‌شوند. در یادگیری از تجربه، همچنین آموزش قوانین بیان شده در منطق وجود دارد. Mit97 یک متن آموزشی جامع در مورد فراگیری ماشین یا آموزش ماشینی^۵ می‌باشد. برنامه‌ها فقط می‌توانند واقعیات یا رفتاری را یاد بگیرند که بتوانند آن را نمایش دهند و متأسفانه سیستم‌های آموزشی تقریباً همه براساس توانایی‌های خیلی محدود شده برای ارایه اطلاعات می‌باشند.

^۱ common sense knowledge and reasoning

^۲ learning from experience

^۳ connectionism - یک تیوری شناختی که به مغز به صورت یک شبکه‌ی بزرگ که با هم کار می‌کنند (تعامل دارند)

با تعداد زیادی متصل کننده بین هر گره نگاه می‌کند. (Babylon / Concise Oxford English Dictionary)

^۴ neural nets - یک سیستم کامپیوتری از روی مغز انسان و سیستم عصبی ساخته شده است. (Babylon / Concise

Oxford English Dictionary)

^۵ machine learning - توانایی یک ماشین برای بهتر کردن عملکرد خود براساس نتیجه‌های قبلی. شبکه‌های عصبی یک

مثال از آن می‌باشند. (Babylon / FOLDOC)

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برنامه ریزی^۱ - برنامه ریزی تلاش می کند که عملیات را برای رسیدن به هدف ها مرتب نماید . کاربردهای برنامه ریزی شامل تدارکات^۲ ، زمانبندی ساخت^۳ و برنامه ریزی ساخت مراحلی برای تولید محصول مطلوب می باشد . با یک برنامه ریزی بهتر می توان مقادیر زیادی در هزینه صرفه جویی نمود . برنامه ریزی برنامه ها با [شناخت] واقعیت های عمومی در مورد جهان (مخصوصاً واقعیاتی در مورد اثرات عملکردها) شروع می شوند ، واقعیات ، در مورد وضعیت های مشخص و یک عبارت در مورد هدف می باشند . از آن ها یک استراتژی برای رسیدن به هدف به دست می آید . در بسیاری از موارد ، استراتژی فقط یک ترتیب از عملکردها می باشد .

شناخت شناسی^۴ - یک شاخه از فلسفه که در مورد منبع^۵ ، طبیعت^۶ ، روش ها و محدودیت های دانش بشری بحث می کند . به عبارت دیگر شناخت شناسی ، مطالعه در مورد این است که چه ما می دانیم و چگونه می دانیم .^۷ به بیان دیگر ، یک مطالعه بر روی انواع دانش که برای حل مسایل در محیط (جهان) لازم می شوند می باشد .

هستی شناسی^۸ - شاخه ای از متافیزیک است که در مورد طبیعت موجودات صحبت می کند .^۹ به عبارت دیگر ، هستی شناسی ، مطالعه ای انواع چیزهایی است که موجودند . در برنامه ها و عبارات به انواع

planning^۱

logistics^۲

manufacturing scheduling^۳

epistemology^۴

origin^۵

nature^۶

Babylon / Learning , Performance and Training Definitions^۷

ontology^۸

Babylon / Concise Oxford English Dictionary^۹

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مخالفی از اشیا توجه می شود و ما این انواع و خصوصیات اساسی آن ها را مطالعه می کنیم. تأکید بر هستی شناسی از سال ۱۹۹۰ شروع شد.

ابتکارها یا اکتشافات^۱ - برای افزایش احتمال حل برخی از مسئله ها به کار می روند.^۲ این واژه به صورت متنوع در هوش مصنوعی به کار می رود. از توابع مکاشفه ای^۳ در برخی از موقع در جستجو برای اندازه گیری این که برای یک گره در یک درخت جستجو، برای رسیدن به اهداف چه مسافتی طی می شود، استفاده می شود. مستندات اکتشاف^۴ دو گره را در یک درخت جستجو برای دیدن این که کدام بهتر از دیگری است مقایسه می کند.

برنامه نویسی ژنتیک^۵ - یک برنامه نویسی است که الگوریتم ژنتیکی^۶ را برای تمام برنامه های کامپیوتری توسعه می دهد. در برنامه نویسی ژنتیکی، جمعیت برنامه ها برای حل مسئله ها توسعه می یابد. از برنامه های ژنتیکی می توان برای حل مسایل طبقه بندی، کنترل، روبوتیک، بهینه سازی، تیوری بازی ها و الگو شناسی استفاده نمود.^۷ برنامه نویسی ژنتیک توسط گروه جان کوزا^۸ توسعه داده شد.

کاربردهای هوش مصنوعی

برخی از کاربردهای هوش مصنوعی در اینجا آمده است:

heuristics^۱

Babylon / WordNet 2.0^۲

heuristic functions^۳

heuristic predicates^۴

genetic programming (GP)^۵

Genetic Algorithm (GA)^۶

Babylon / FOLDOC^۷

John Koza^۸

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تیوری بازی ها^۱ - بازی ها مواردی خوب برای تحقیق می باشند زیرا بازی ها کوچک و جامع هستند. و بنابراین به آسانی برنامه ریزی می شوند. بازی ها مدل های خوبی از وضعیت های رقابتی می توانند باشند، بنابراین روش های طراحی شده برای تیوری بازی ها شاید بتوانند در مسایل عملی هم به کار گرفته شوند. با پرداخت چند دلار شما می توانید ماشین هایی را بخرید که می توانند به صورت حرفة ای بازی شترنج را پیاده سازی کنند. در آن ها برخی از موارد هوش مصنوعی وجود دارد، اما آن ها به خوبی افراد بازی می کنند و برخلاف افراد هستند که بیش تر با محاسبات جبری بی فکر بازی می کنند و به صدها هزار از وضعیت ها نگاه می کنند.

سخن شناسی^۲ - سخن شناسی، توانایی سیستم های کامپیوترا برای پذیرش ورودی سخن و کار بر روی آن یا تبدیل آن به صورت نوشتاری است.^۳ در دهه ۱۹۹۰، سخن شناسی کامپیوترا به سطحی کاربردی برای اهدافی محدود رسید. خطوط هوایی ایالات متحده ای آمریکا صفحه کلیدی درختی را که برای اطلاعات پرواز استفاده می شد را با سیستمی که از سخن شناسی استفاده می کرد، برای شماره های پرواز و نام شهرها جایگزین کردند؛ این روش کاملاً مناسب بود.

فهم زبان طبیعی^۴ - فقط دریافت (گرفتن) کلمات متوالی در یک کامپیوتر کافی نمی باشد. فقط تجزیه ای جملات هم کافی نمی باشد. کامپیوتر باید بفهمد که متن در چه موردی می باشد و این مورد به زودی برای دامنه های خیلی محدود امکان پذیر می باشد.

تصویر مجازی در کامپیوتر^۵ - موردی در روبوتیک است که در آن برنامه ها تلاش می کنند اشیایی که به صورت تصاویر دیجیتالی توسط دوربین های ویدیویی دریافت کرده اند را تشخیص دهند و در

game playing^۱

speech recognition^۲

Babylon / Britannica Concise Encyclopedia^۳

- زبان طبیعی متضاد کد کامپیوترا یا زبان مصنوعی می باشد.

understanding natural language^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نتیجه ربات ها بتوانند بینند. جهان از اشیایی سه بعدی تشكیل شده است اما ورودی ها برای چشم بشر و دوربین های تلویزیون کامپیوتری دو بعدی می باشند. برخی از برنامه ها فقط در حالت دو بعدی می توانند کار کنند، اما تصویر مجازی کامپیوتری کامل، اطلاعات سه بعدی ناتمام که فقط مجموعه ای از دیدهای دو بعدی نیستند را لازم دارد. در حال حاضر فقط راه های محدودی برای نمایش اطلاعات سه بعدی به صورت مستقیم وجود دارد و این راه ها از قرار معلوم به خوبی راه هایی که انسان ها استفاده می کنند نمی باشد.

سیستم های خبره^۱ - سیستم های خبره تلاش می کنند که دانش یک انسان خبره (ماهر) را بگیرند و آن را در یک سیستم کامپیوتری پیاده سازی نمایند. از سیستم های خبره انتظار می رود که بتوانند کارهایی که به یک فرد خبره نیاز دارند، مثل پزشکی، زمین شناسی و مشورت در سرمایه گذاری. سیستم های خبره برخی از موفق ترین کاربردهای هوش مصنوعی بوده اند چون که این برنامه ها باید در دنیای واقعی کار کنند و با برخی از مشکلات مهم موجود در هوش مصنوعی مواجه شده اند؛ مثل: کمی اطلاعات ورودی مناسب و استدلال بر پایه ای احتمال. یکی از اوّلین سیستم های خبره در سال ۱۹۷۴ به نام MYCIN بود، که اثرات باکتری های موجود در خون و معالجات آن را تشخیص می داد. این دستگاه این کار را بهتر از دانشجویان پزشکی یا دکترها انجام می داد. یعنی هستی شناسی آن شامل باکتری، نشان ها (علامت ها) و معالجات بود و کارها را به موقع انجام می داد. از زمانی که خبرگان با مهندسان همکاری کردند چیزهایی را در مورد بیماران، دکترها، سلامتی، بهبود و ... دانستند و واضح است که دانش مهندسان تحت تأثیر آن چه که خبرگان به آن ها می گفتند در یک چارچوب کاری معین قرار داشت. در وضعیت فعلی هوش مصنوعی، این مطلب باید درست باشد.

طبقه بندی اکتشافی^۲ - یکی از بیش ترین انواع عملی (امکان پذیر) سیستم های خبره ای ارایه شده توسط هوش مصنوعی برای قراردادن برخی از اطلاعات در یک مجموعه ای طبقه بندی شده ای معین با استفاده از چند منبع معین می باشد. مثالی در این مورد نصیحت کردن کسی برای دریافت کارت اعتباری

expert systems^۱

heuristic classification^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پیشنهاد شده، ثبت پرداخت او، همچنین چیزی که او می خرد و ثبت مؤسسه ای که او از آن خرید می نماید، می باشد.

ارتباط میان هوش مصنوعی و فلسفه: هوش مصنوعی دارای ارتباطات زیادی با فلسفه می باشد، مخصوصاً با فلسفه ای تحلیلی مدرن^۱ ارتباط زیادی دارد. هر دو فکر و عقل سلیم را مطالعه می کنند.

پیش نیازهای هوش مصنوعی: باید ریاضیات و مخصوصاً منطق ریاضیات^۲ را مطالعه کنیم. برای نزدیکی زیستی به هوش مصنوعی، روان شناسی و فیزیولوژی سیستم های عصبی را مطالعه می کنیم. تعدادی زبان برنامه نویسی و در کم ترین حالت، سی^۳، لیسپ^۴ و پرولوگ^۵ را باید بلد باشیم. همچنین فکر خوبی است که یکی از زبان های پایه ای ماشین را یاد بگیریم. کارها بیش تر با زبان هایی که مدد هستند انجام می شوند. در اواخر دهه ۹۰ این زبان ها شامل C++ و جاوا^۶ بود.

برخی از سازمان ها و موسساتی که در زمینه ی هوش مصنوعی فعالیت می کنند: انجمن آمریکایی هوش مصنوعی^۷، کمیته ی هماهنگ کننده ی اروپا برای هوش مصنوعی^۸ و جامعه ی

modern analytic philosophy^۹

mathematical logic^{۱۰}

C^{۱۱}

Lisp^{۱۲}

Prolog^{۱۳}

Java^{۱۴}

The American Association for Artificial Intelligence (AAAI)^{۱۵} با آدرس اینترنتی <http://www.aaai.org>

European Coordinating Committee for Artificial Intelligence (ECCAI)^{۱۶} به آدرس اینترنتی <http://eccai.org>

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هوش مصنوعی و شبیه سازی رفتار^۱ جوامع علمی علاقه مند به هوش مصنوعی می باشند . شرکت ماشین آلات محاسبه کننده^۲ یا SIGART یک گروه مهم در زمینه ای هوش مصنوعی می باشد . کنفرانس بین المللی هوش مصنوعی^۳ . تعاملات الکترونیک با هوش مصنوعی^۴ و هوش مصنوعی^۵ و روزنامه ای تحقیقات تحقیقات هوش مصنوعی^۶ و تعاملات IEEE با تحلیل الگو و ماشین های هوشمند^۷ چهار روزنامه ای مهم منتشر کننده ای مقالات هوش مصنوعی می باشند . در حال حاضر چیز دیگری که مناسب درج در این قسمت باشد را پیدا نکرده ایم .

Society for Artificial Intelligence and Simulation of Behavior (AISB)^۱ با آدرس اینترنتی

<http://www.cogs.susx.ac.uk/aisb>

http://www.acm.org/sigart به آدرس اینترنتی Association for Computing Machinery (ACM)^۲

http://www.ijcai.org به آدرس اینترنتی The International Joint Conference on AI (IJCAI)^۳

Electronic Transactions on Artificial Intelligence^۴ با آدرس اینترنتی

<http://www.ida.liu.se/ext/etai>

http://www.elsevier.nl/locate/artint به آدرس اینترنتی^۵

http://www.jair.org به آدرس اینترنتی Journal of Artificial Intelligence Research^۶

IEEE Transactions on Pattern Analysis and Machine Intelligence^۷ به آدرس اینترنتی

<http://computer.org/tpami>

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هوش مصنوعی

دیوس مطالب

هوش مصنوعی چیست؟

تاریخچه‌ی مختصر

چگونگی فن

چرا هوش مصنوعی را مطالعه می کنیم؟

کنجدکاوی؛ ساخت سیستم‌های هوشمند؛ انجام بعضی از کارها مثل بازی شطرنج که به نظر می‌رسد برای انجام آن‌ها می‌توانیم از هوش مصنوعی کمک بگیریم؛ انجام کارهایی مثل ختنی کردن مین

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یا تمیز کردن استخراج شنا که برای انسان خطرناک یا کسل کننده هستند، از دلیل هایی هستند که ما هوش مصنوعی را مطالعه می نماییم.

هوش مصنوعی چیست؟

در اینجا چهار تعریف زیر را ارایه می کنیم:

- ۱- سیستم هایی که شبیه انسان ها عمل می کنند.
- ۲- سیستم هایی که همانند انسان ها فکر می کنند.
- ۳- سیستم هایی که معقولانه ^۱ فکر می کنند.
- ۴- سیستم هایی که معقولانه عمل می کنند.

دو مین و سوییں مورد می توانند در طبقه بندی هوش مصنوعی قوى قرار گیرند و موارد اول و چهارم بیشتر در طبقه بندی هوش مصنوعی ضعیف قرار می گیرند. در زیر توضیحاتی در مورد تعریف هایی که ذکر شد آمده است:

عمل کردن مثل انسان: آزمایش تورینگ ^۲ - آلن تورینگ در سال ۱۹۵۰ در مقاله‌ی ماشین آلات محاسبه و هوش در مورد شرایطی برای ماشین هوشمند بحث کرده است. او گفته است که اگر ماشین بتواند کاملاً وانمود کند که مانند بشر می باشد آن گاه شما می توانید مطمین باشید که هوشمند است. این آزمایش توانست برخی از افراد و نه همه را راضی نماید. فرد پرسش کننده می تواند با ماشین کار کند و یک فرد (در طرف مقابل) با ماشین تایپ کند (برای جلوگیری از احتیاج ماشین به تقلید ظاهر یا صدای شخص) و فرد باید پرسش کننده را دارای این عقیده نماید که فرد می باشد و ماشین باید تلاش کند که فرد پرسش کننده را گمراه نماید [و خود را به جای فرد جا بزند]. آزمایش تورینگ یک جانبه (غیر منصفانه) می باشد.

rationally ^۱
Turing test ^۲

مترجم: سهراب جلوه گر

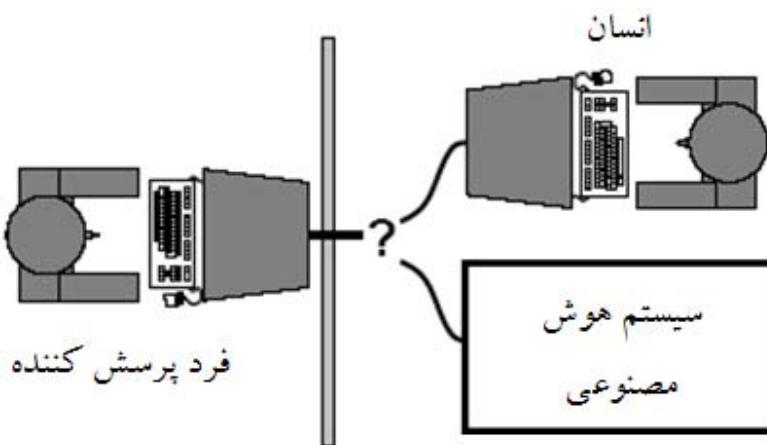
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک ماشین که آزمایش را می گذراند باید کاملاً هوشمند باشد ، ولی یک ماشین ، برای تقلید ، بدون دانستن به اندازه‌ی کافی در مورد افراد می تواند هوشمند باشد . کتاب مغز کودکان^۱ دانیل دنت^۲ دارای یک بحث عالی در مورد آزمایش تورینگ و جواب مختلف آزمایش تورینگ که به کار گرفته می شوند می باشد . برخی از افراد به سادگی تصوّر می کنند که یک برنامه‌ی نسبتاً گنج^۳ هوشمند می باشد .^۴ اجزای اصلی پیشنهاد شده برای هوش مصنوعی عبارتند از : دانش ، استدلال ، زبان و آموزش .

اشکالات آزمایش تورینگ این بود که : آزمایش تورینگ تجدید پذیر^۴ ، سودمند^۵ یا جوابگو^۶ برای تحلیل ریاضی نیست.



Brainchildren^۷

Daniel Dennett^۸

<http://www-formal.stanford.edu/jmc/whatisai/node1.html>^۹

reproducible^{۱۰}

constructive^{۱۱}

amenable^{۱۲}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هنوز هیچ برنامه ای از آزمایش تورینگ، سربلند بیرون نیامده است. یک برنامه در صورتی آزمایش تورینگ را با موفقیت می گذراند که قادر باشد زبان طبیعی را بفهمد؛ چیزهایی که می داند (دانش) را بتواند بیان (ارایه) کند؛ دانش داشته باشد و بتواند استدلال کند.

فکر کردن مانند انسان: علم شناخت^۱

علم شناخت برای به وجود آوردن تیوری هایی که چگونگی عملکرد مغز انسان را توضیح می دهد تلاش می کند. در این راه از مدل های کامپیوتری هوش مصنوعی و روش های تجربی روانشناسی استفاده می نماید. لازم است توجه کنیم که بیش تر روش های هوش مصنوعی به طور مستقیم براساس مدل های شناختی نمی باشند؛ اغلب دشوار است که روش های هوش مصنوعی به صورت برنامه های کامپیوتری ترجمه شوند. علم شناخت اساساً از هوش مصنوعی جدا می باشد.

در دهه ۱۹۶۰ انقلاب شناخت به وجود آمد. علم شناخت دارای اشتراک با هوش مصنوعی می باشد. دانشمندان علم شناخت، طبیعت هوش را با یک دید روانی مطالعه می کنند و بیش تر مدل های کامپیوتری که به توضیح رخدادهای درون مغز ما در مدت حل مسئله، به یادآوردن، درک و دیگر فعالیت های روانشناسی کمک می کنند را می سازند. یک اشتراک اساسی میان هوش مصنوعی و علم شناخت برای روانشناسی این بوده است که در مدل پردازش اطلاعات فکری انسان، تشییه "مغز به صورت کامپیوتر" به صورت لفظ به لفظ کاملاً انجام می شود.^۲

هربرت سیمون (۱۹۱۶-۲۰۰۱) در مورد علم شناخت می گوید: "هوش مصنوعی می تواند دو هدف داشته باشد. یکی استفاده از توانایی کامپیوترها برای تکمیل فکر بشری، همان طوری که ما از موتورها برای تکمیل توانایی بشر یا اسب استفاده می کنیم، علم روبوتیک و سیستم های خبره شاخه های اصلی این هدف اوّل می باشند. هدف دیگر، استفاده از هوش مصنوعی کامپیوتری برای فهمیدن چگونگی فکر بشر

cognitive science^۱

<http://www.aaai.org/AITopics/html/cogsci.html>^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

است. اگر شما برنامه ها را با چیزی که می توانند انجام دهند تست نکنید و بدون فهمیدن چگونگی انجام آن کارها تست کنید، آن گاه آن شماید که واقعا علم شناخت را انجام می دهید؛ شما در حال استفاده از هوش مصنوعی برای فهمیدن عملکرد بشری هستید. ”^۱

علم عصب شناسی^۲: شاخه ای از علم اعصاب است که اصول زیستی اتفاقات روانی را مطالعه می کند.^۳

علم شناخت و علم عصب شناسی در حال حاضر از هوش مصنوعی مجزا هستند. هر دو دارای این ویژگی مشترک با هوش مصنوعی هستند: **قیودی های در دسترس، هیچ چیزی را شبیه هوش انسان تولید نمی کنند.** بنابراین، هر سه زمینه در یک جهت اساسی مشترک هستند!

فکر منطقی (معقولانه) : قوانین فکر - قواعد اصولی^۴ بیش از تشریحی حاکم می باشند، چند مدرسه‌ی یونانی شکل‌های مختلف منطقی را توسعه دادند: نمادها^۵ و قوانین استنتاج^۶ برای تفکرات، برای فکر مکانیزاسیون ممکن است به کار روند یا ممکن است به کار نروند. مشکلات این روش یکی این است که همه‌ی رفتارهای هوش با بررسی های منطقی انجام نمی شوند. و دیگری این که میان حل یک مسئله در کل و حل آن در عمل تحت محدودیت منابع گوناگون نظیر زمان، محاسبه و دقت تفاوت وجود دارد.

عمل کردن به صورت منطقی (معقولانه)

^۱ همان منبع قبلی

cognitive neuroscience^۷

WordNet 2.0^۸

normative^۹

notation^{۱۰}

rules of derivation^{۱۱}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

رفتار منطقی^۱: کار درست را انجام می دهد. **کار درست**، کاری است که بیش ترین دستیابی به هدف را دارد. بدون فکر هم می توانیم کار درست را انجام دهیم؛ اما فکر کردن باید در انجام کار معقولانه وجود داشته باشد. مزایای عمل کردن به صورت منطقی یکی این است که کلی تراست و دیگری این است که هدفش به خوبی تعریف شده است. ارسطو^۲ می گوید: هر هنر، هر تحقیق و غیره، هر عمل و عکس العمل برای انجام صحیح، احتیاج به فکر دارد.

تاریخچهٔ هوش مصنوعی

در سال ۱۹۴۳، مک کلوچ و پیتر^۳: مدل مداری بولین مغز | در سال ۱۹۵۰، ماشین آلات محاسبه گر و هوش تورینگ | سال های ۱۹۵۲ تا ۱۹۶۹ نظر و دست به دست کردن | سال های دهه ۵۰، برنامه های هوش مصنوعی اویله، شامل برنامه‌ی چک کننده‌ی سامویل^۴، نظریه‌ی منطقی نوئل و سیمون^۵، ماشین هندسی گلرنتر^۶ | سال ۱۹۵۶ نشست دارتمند^۷: "هوش مصنوعی" پذیرفته شد، الگوریتم کامل راینسون^۸ برای استدلال منطقی | در سال های ۱۹۶۶ تا ۱۹۷۴، پیچیدگی محاسباتی پیدا می کند و پژوهش در مورد شبکه‌ی عصبی^۹ تقریباً ناپدید می شود | در سال های ۱۹۶۹ تا ۱۹۷۹ توسعه‌ی اویله‌ی سیستم های برمبنای دانش^{۱۰} | در سال های ۱۹۸۰-۸۸ توسعه‌ی عظیم سیستم های خبره‌ی صنعتی | در سال های ۱۹۹۸

rational behavior^۱

Aristotle^۲

McCulloch و Pitts^۳

samuel^۴

Newell & Simon^۵

Gelernter^۶

Dartmouth meeting^۷

Robinson^۸

neural network^۹

knowledge-based systems^{۱۰}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- ۹۳ سیستم های خبره‌ی صنعتی و رشکست می‌کند: "زمستان هوش مصنوعی" | در سال‌های ۱۹۸۵ تا ۱۹۹۵ شبکه‌های عصبی شهرت می‌یابند | سال ۱۹۸۸ تجدید حیات احتمال، افزایش اساسی در عمق تکنیک ALife, GAs : "Nouvelle AI" و محاسبه کننده‌ی انعطاف پذیر | سال ۱۹۹۵ - عامل‌ها و عامل‌ها و عامل‌ها و | سال ۲۰۰۳ به سطح انسانی هوش مصنوعی پرداخته می‌شود.

زندگی مصنوعی^۱ - دارای دورنمای بسیار خوبی در زیست‌شناسی و علم کامپیوتر می‌باشد. این شاخه از علم در مورد کشت مصنوعی، رفتارهای شبیه زندگی نظری رویش، سازگاری، تولید مثل، اجتماعی شدن، آموزش و حتی مرگ تحقیق می‌نماید. محدود به چیزهای ابتدایی نمی‌باشد، دانشمندان زندگی مصنوعی می‌توانند از ترکیبی از اجزا و برنامه‌های کامپیوتری که در زمان، صرفه‌جویی می‌کنند و دیگر فن آوری‌های شگفت‌انگیز، مانند آن‌هایی که برای تولید رفتارها و تجسمات شکل شناسی جدید^۲ جستجو می‌نمایند استفاده نمایند. منابع لیست شده در زیر شما را با چیزهای شگفت‌انگیز نظری شبکه‌های عصبی کامپیوتری که شبیه زندگی واقعی هستند آشنا می‌کند؛ ویروس‌های کامپیوتری که شبیه ویروس‌های زنده عمل می‌نمایند؛ الگوریتم‌های ژنتیکی و تکاملی که استراتژی‌های زندگی را تحت کنترل در می‌آورند و از آن‌ها برای حل مسئله استفاده می‌نمایند.^۳

وضعیت دانش

در حال حاضر کدامیک از موارد زیر می‌تواند انجام شود؟

اجرای یک بازی تنیس. (می‌شود) | رانندگی با اینمی در یک جاده‌ی مارپیچی کوهستانی. (می‌شود) | خرید مایحتاج هفتگی از فروشگاه‌های های وب (می‌شود) | اجرای یک بازی پل (می‌شود) | ترجمه

Artificial Life یا ALife^۱

neomorphic^۲

<http://www.aaai.org/AITopics/html/alife.html>^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

کردن بلا فاصله‌ی زبان گفتاری انگلیسی به زبان گفتاری سوندی (نمی شود) | مکالمه‌ی موفق با دیگر افراد برای مدت یک ساعت (نمی شود) | انجام دادن یک عمل جراحی پیچیده (نمی شود).

توضیحی در مورد بازی پل، بازی‌ای کارتی (با ورق) و چهار نفره می‌باشد. مانند سایر بازی‌های کارتی، یکی از خصوصیات اوّلیه‌ی این بازی این است که بازی‌ای با اطلاعات ناقص می‌باشد که در آن بازی کننده‌های مختلف دارای اطلاعات مختلفی در مورد وضعیت واقعی بازی می‌باشند. این یکی از خصوصیات این بازی می‌باشد که باعث می‌شود ماشین‌ها بازی کننده‌های شایسته‌ای برای بازی پل نباشند.
¹ در زیر تصویری از کارت‌های این بازی را مشاهده می‌کنید:



<http://www.cirl.uoregon.edu/research/bridge.html>¹

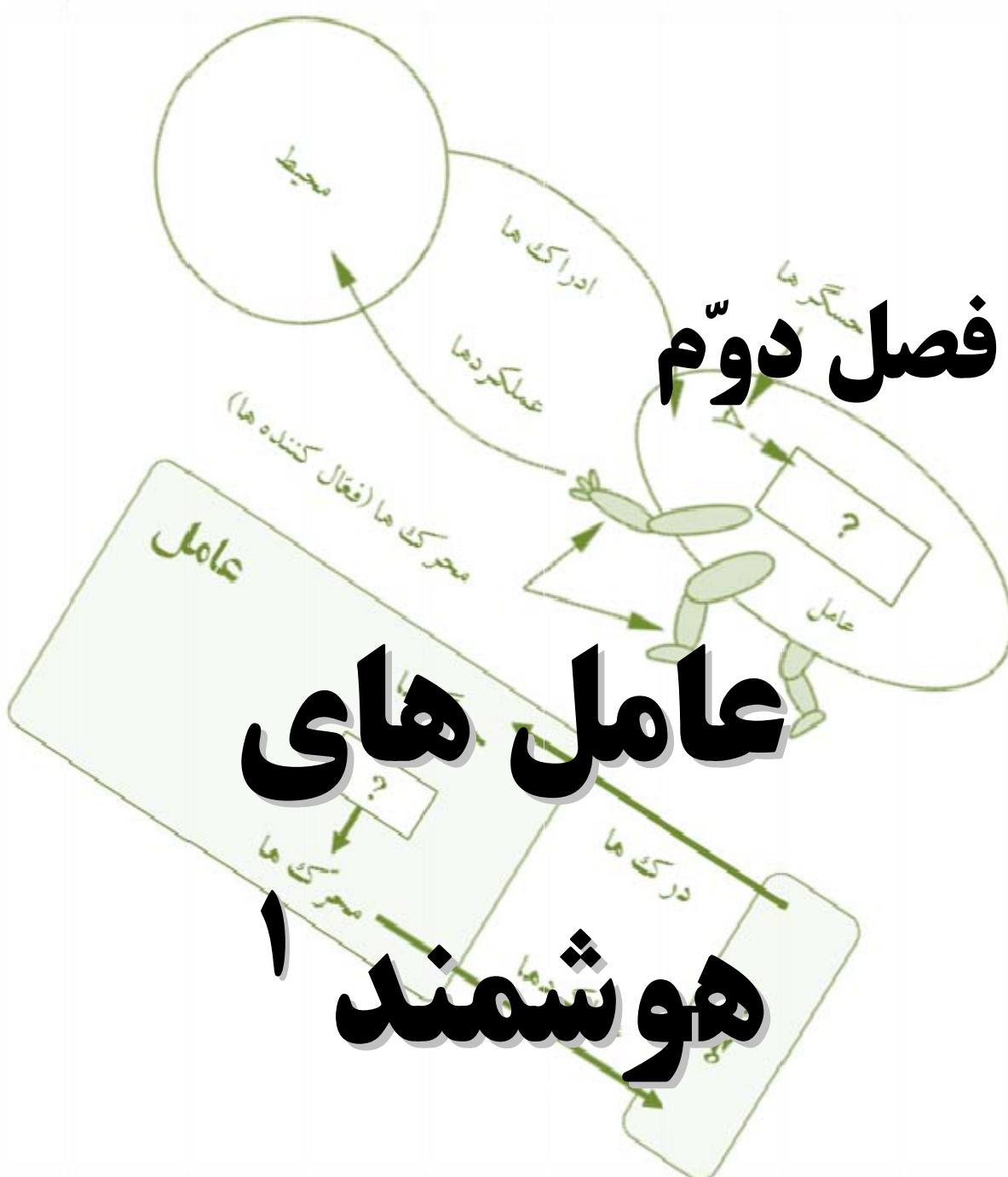
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل دوم

عامل های هوشمند



Intelligent agents¹

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- عامل ها و محیط ها
- عقلانیت^۱
- PEAS (ارزیابی عملکرد^۲، محیط^۳، عمل کننده ها^۴ و حسگرها^۵)
- انواع محیط
- انواع عامل ها

rationality^۱

Performance measure^۲ یا معیار کارایی

Environment^۳

Actuators^۴

Sensors^۵

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل ها

* یک عامل می تواند هر چیزی باشد که در یک **محیط** عمل می کند؛ [وضعیت] محیط خود را با استفاده از **حسگرها** دریافت می کند؛ روی محیط خود با استفاده از **عمل کننده ها** عمل می نماید و طوری عمل می کند که بیش ترین نزدیکی را به اهدافش داشته باشد.

تعريف های دیگری در مورد عامل:

- **تعريف راسل و نورویگ (R & N)**: هر چیزی که می تواند وضعیت محیط خود را با استفاده از حسگرها دریافت کند و توسط محرّک ها بر روی محیط عمل کند یک عامل است.
- **ولریج^۱**: یک عامل، یک سیستم کامپیوتری است که در یک محیط قرار داده می شود و قادر است به صورت خودمختار عمل نماید.
- * **ابزار ادراکی برای تحلیل سیستم ها**: روبوت ها^۲، سافتبوت ها^۳، چراغ های ترافیک، ترموموستات ها^۴ و غیره
- * عامل های هوشمند از این جهت برای ما اهمیّت دارند که به هوش مصنوعی کمک می کنند.

مثال هایی از عامل ها

Woolridge ^۱
robots ^۲
softbots ^۳
thermostats ^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل انسانی مثل: چشم ها، گوش ها، پوست، حس چشایی^۱ و ... برای حسگرها و دست ها، انگشتان، پاها، دهان^۲ و ... برای عمل کننده ها

برای روبوت: دوربین، دوربین مادون قرمز^۳، ضربه گیر یا سپر^۴ و ... برای حسگرها و گیرنده ها (ابزارهایی که چیزی را می گیرند)^۵، تایرها، نورها، بلند گوها و ... برای عمل کننده ها

برای عامل نرم افزاری یا سافتبوت، حسگرها به صورت توابع هستند و ورودی توابع، اطلاعات آماده شده به صورت رشته های بیتی یا نشان ها (سمبل ها) ای کد شده هستند. و توابع، که عمل کننده ها نیز هستند خروجی ها را اجرا می کنند. پس توابع هم به صورت حسگر هستند و اطلاعات را به صورت رشته های بیتی یا نشان ها (سمبل ها) ای کد شده می گیرند و هم به صورت عمل کننده هستند که بر روی محیط، عمل می کنند.

خودمختاری^۶ در عامل ها

عامل ها وظایف خود را تا حد زیادی به صورت مستقل انجام می دهند و به وسیله ی کاربران یا برنامه های دیگر برنامه ریزی می شوند و این برنامه ها عامل را برای انجام کارش راهنمایی می کند. مبنای سیستم های خودمختار، تجربه و دانش آن ها می باشد. عامل ها به دانش اوّلیه نیاز دارند و باید بتوانند یاد بگیرند و برای کارهای به مراتب پیچیده تر دارای انعطاف باشند.

مطلوب دیگری در مورد خودمختاری

taste buds ^۱
mouth ^۲
infrared ^۳
bumper ^۴
grippers ^۵
autonomy ^۶

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل ها اغلب دارای خود مختاری هستند ولی این خود مختاری ، اغلب به صورت کامل نمی باشد . حال مشکلی در این مورد وجود دارد و آن این است که عامل خود مختار چه هنگام باید درخواست کمک کند ؟ . یک عامل خود مختار قادر است با توجه به دریافت هایش تصمیم بگیرد .

چگونه با این دردسرها رفتار کنیم ؟ : عامل ها دوست دارند که کاری که انجام می دهند دارای پیش نیاز نباشد . مشخص کردن این که در هر موردی باید چه کاری انجام دهنده مشکل است . در مورد عامل ها بهتر است فرض را بر این بگذاریم که آن ها هوشمند هستند و سپس در مورد آن ها صحبت کنیم .

عامل ها و محیط ها

عامل ها شامل انسان ها ، ربات ها ، ترمومترات ها (تنظیم کننده های حرارت) و ... می باشند ؛ مولفان دارای عامل ها هستند ، ورزشکاران حرفه ای دارای عامل ها هستند ، ستاره های سینما دارای عامل ها هستند و شما نیز دارای عامل ها می باشید . زیرا عامل ، کسی یا چیزی دارای تخصص می باشد که عهده دار انجام کاری برای شما می شود . برنامه های کامپیوترا که به شما کمک می کنند که دانسته های محاسباتی خود را افزایش دهید " عامل ها " می باشند . تحقیقات دانشمندان هوش مصنوعی در حال افزایش توانایی ، استقلال و ... عامل ها می باشد .^۱ تابع عامل از تاریخچه های ادراکی ، به عملیات نگاشت می کند :

$$A \rightarrow f : p^* .$$

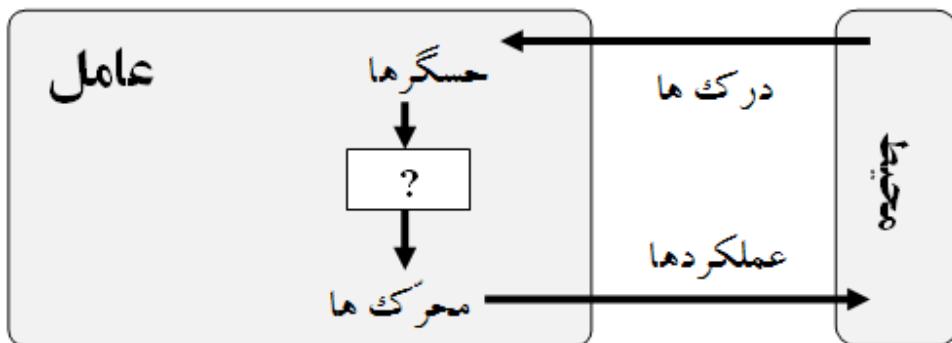
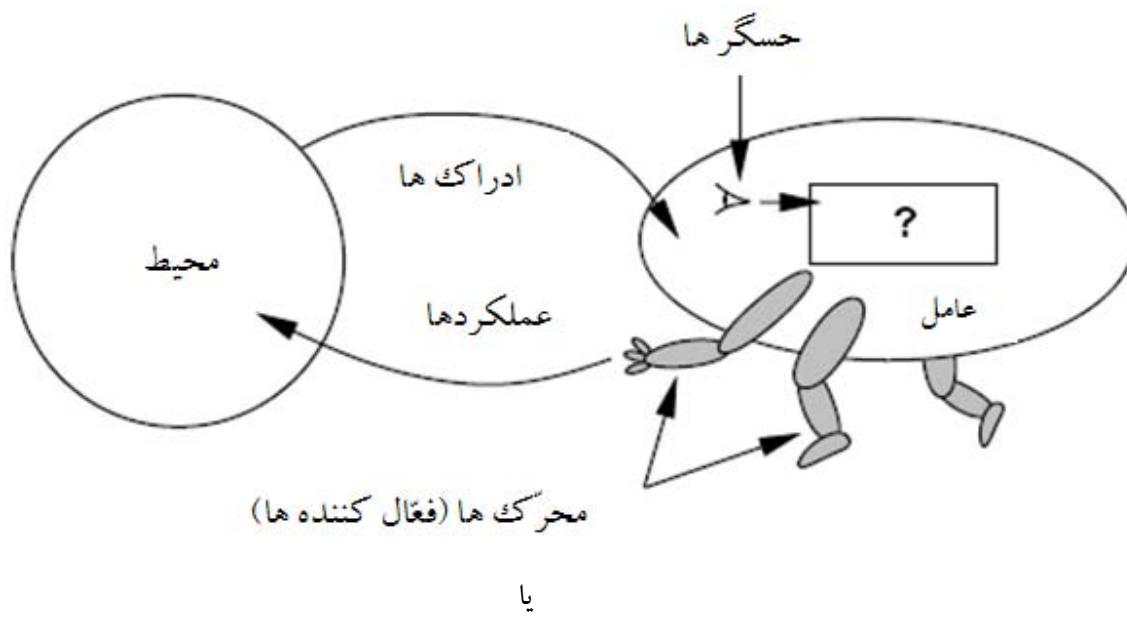
برنامه های عامل ، در معماری فیزیکی برای تولید f اجرا می شود .

^۱ <http://www.aaai.org/AITopics/html/agents.html>

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



ادراک‌ها (دريافت‌ها): اطلاعات فرستاده شده به حسگرهای یک عامل می‌باشد. مثل، نور، صدا، امواج الکترو مغناطیسی و علایم (سیگنال‌ها).

حسگرهای: روش‌هایی های یک عامل برای جمع آوری اطلاعات در مورد محیطش می‌باشد. مثل: چشم‌ها، گوش‌ها، سلول‌های فتوالکتریک و ...

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

حرکت ها: عامل ، به وسیله‌ی حرکت‌ها بر روی محیط ، عمل می‌نماید . تایرها ، بازوها ، رادیوها ، نورها و ... مثال‌هایی از حرکت‌ها هستند .

عملکرد: کار یا عملی است که بر روی محیط انجام می‌شود ؛ مثل : حرکت و غلطیدن .

ساختار عامل‌های هوشمند

عامل ، از **معماری و برنامه تشکیل شده** است . معماری ، ابزاری است که به وسیله‌ی آن می‌توان برنامه‌ی عامل را اجرا کرد . مثل : کامپیوتر همه منظوره ، ابزار تخصصی ، روبوت و ...

برنامه‌ی عامل‌ها

ما می‌توانیم رفتار عامل مان را با یک تابع F توصیف نماییم . داریم :

عملکرد = (تاریخچه‌ی ادراکی ، ادراک فعلی) F

به کار گیری این تابع ، کار یک برنامه‌ی عامل می‌باشد .

مثال : دنیای جاروبرقی

بیایید با یک مثال خیلی ساده شروع کنیم ؛ فرض کنید که دو فضای A و B وجود دارند که هر فضا یا می‌تواند تمیز یا کثیف باشد و این ، محیط عامل است . حسگر‌ها عبارتند از : حسگر کثیفی و تعیین محل . حرکت‌ها عبارتند از : جاروبرقی و چرخ‌ها . ادراک‌ها هم تمیز و پاک بودن محیط می‌باشد . عملکردها ؛ حرکت به چپ ، راست ، مکش و هیچ کار می‌باشند . در این مثال ساده ، ما می‌توانیم همه‌ی رشته‌های ادراکی ممکن و عملکردهای واپسی را لیست نماییم و این ، یک عامل برنایی جدول^۱ نام دارد .

table-based^۱

مترجم: سهراب جلوه گر

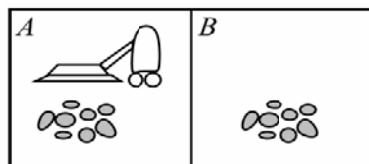
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عقلانیت^۱

به طور کلی ، عقلانیت به معنی " انجام چیز درست می باشد " در این مورد دقت بیش تری لازم می باشد ؛ " چیز درست " ، چیست ؟ ما به یک تعریف از موقعیت هم نیازمندیم . یک عملکرد عقلانی (عاقلانه یا از روی عقل) ، عملکردی است که معیار کارایی (ارزیابی عملکرد) یک عامل ، ادراک ها و عملکردهای آن را بیشینه می نماید . راسل و نورویگ در این باره می گویند : عامل های عقلانی ، برای هر رشته ای ادراکی ، باید عملکردی که انتظار می رود معیار کارایی و ملاک ارایه شده توسط رشته ای ادراکی و هر چه در دانش عامل وجود دارد را بیشینه می کند ، انتخاب نماید . ما نیازی نداریم که عامل قادر باشد که آینده را پیش بینی نماید یا رویدادهای بد را پیش گویی نماید . جمع آوری اطلاعات هم ممکن است یک عملکرد عقلانی باشد ؛ به عنوان مثال ، عبور بدون توجه از خیابان ، نامعقولانه می باشد . عامل های عقلانی باید دارای توانایی یادگیری (آموزش)^۲ باشند (به جز در موارد خیلی ساده و محیط هایی که به خوبی قابل فهم می باشند) . آموزش به معنی بهبود دادن عملکرد یک عامل است . آموزش ، ممکن است به معنی کاهش نامعلومی (احتمال) یا ارایه ای ملاحظاتی به حساب یا گزارش باشد .



عامل های یک جارو برقی

rationality^۱
learning^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تابع $\text{Reflex-Vacuum-Agent}([\text{location},\text{status}])$ یک عملکرد را برمی گرداند. در صورتی که وضعیت^۱ برابر با کشیف^۲ باشد مکش را برمی گرداند و در غیر این صورت، در صورتی که محل^۳ برابر با A باشد، راست را برمی گرداند و در صورتی که محل برابر با B بود، چپ را برمی گرداند.

عملکرد	رشته‌ی ادراکی
Right	[A,Clean]
Suck	[A,Dirty]
Left	[B,Clean]
Suck	[B,Dirty]
Right	[A,Clean],[A,Clean]
Suck	[A,Clean],[A,Dirty]
:	:

عامل عقلانی^۱ - یک عامل هوشمند هر کدام از اعمالی که مقدار معیار کارایی ارایه شده توسط رشته‌ی ادراکی مورد نظر را به بیش ترین مقدار افزایش می دهد انتخاب می نماید. عامل هوشمند، به همه

status^1

Dirty^1

location^1

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چیز آگاهی^۰ ندارد. دریافت ادراکی ممکن است همه‌ی اطلاعات را به وجود نیاورد. عامل هوشمند، نهان بین^۳ هم نمی‌باشد. عملکرد به وجود آمده ممکن است به صورتی که انتظار داشتیم، نباشد. در این صورت، عامل هوشمند، موقّیت آمیز نمی‌باشد. عامل هوشمند دارای ابتکار^۴، یادگیری^۵ و خودمختاری می‌باشد.

PEAS - برای طراحی یک عامل هوشمند، ما باید محیط عملیاتی^۶ را مشخص کنیم. مثلا در طراحی یک تاکسی خودکار، موارد زیر را باید مشخص نماییم: معیار کارایی^۷ | محیط^۸ | عمل کننده‌ها (محرك‌ها)^۹ | حسگرهای^{۱۰}

مثال، طراحی یک تاکسی خودکار، معیار کارایی^۷ اینستی^۷، مقصد^۸، مزایا^۹، رعایت قانون^{۱۱}، آسایش و راحتی^{۱۲} و | محیط^۸ خیابان‌های ایالات متحده‌ی آمریکا / آزاد راه‌های ایالات متحده‌ی آمریکا، ترافیک^{۱۳}، عابران پیاده^{۱۴}، هوا^{۱۵} و | عمل کننده‌ها^۹ فرمان^{۱۶}، گاز، ترمز^{۱۷}، بوق

rational agent ^۱
omniscient ^۲
clairvoyant ^۳
exploration ^۴
learning ^۵
task environment ^۶
safety ^۷
destination ^۸
profit ^۹
legality ^{۱۰}
comfort ^{۱۱}
traffic ^{۱۲}
pedestrains ^{۱۳}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بوق^۴، بلندگو^۵ یا نمایش دهنده^۶ و | حسگرها^۷ ویدئو^۹، شتاب سنج^۷، درجه ها^۸، حس گرهای موتور^۹، صفحه کلید^{۱۰}، GPS^{۱۱} و^{۱۲}

مثال : عامل های خرید ^{۱۲} اینترنتی ، معیار کارایی ^{۹۹} | محیط ^{۹۹} | عمل کننده ها ^{۹۹} | حسگرها ^{۹۹} را باید مشخص نماییم . معیار کارایی ^{۹۹} قیمت ^{۱۳}، کیفیت ^{۱۴}، تناسب ^{۱۵}، کارایی ^{۱۶} | محیط ^{۹۹} سایت های فعلی و آینده ^{۱۷}ی WWW ، فروشنده ها ^{۱۷}، حمل کننده ها ^۱ | عمل کننده ها ^{۹۹} نمایش دادن به کاربر ^۲، دنبال کردن URL ، پر کردن فرم | حسگرها ^{۹۹} صفحات HTML (متنی، گرافیکی، اسکریپتی)

weather^۱

steering^۲

brake^۳

horn^۴

speaker/display^۵

video^۶

accelerometer^۷

gauges^۸

engine sensors^۹

keyboard^{۱۰}

Babylon^{۱۱} مخفف Global Positioning System است و یک سیستم مکان یابی ماهواره ای می باشد .) / (Concise Oxford English Dictionary

shopping^{۱۲}

price^{۱۳}

quality^{۱۴}

appropriateness^{۱۵}

efficiency^{۱۶}

vendors^{۱۷}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

محیط ها

یک معیار برای وجود یک عامل ، وجود آن در یک محیط می باشد . لازم نیست که یک عامل به صورت فیزیکی در محیط باشد ؛ ممکن است محیط ، به صورت یک محیط نرم افزاری باشد . به نظر راسل و نورویگ ، محیط عملیاتی تشکیل شده از : معیار کارایی ، محیط و حرکت هایی که برای عامل قابل دسترسی می باشند .

ویژگی محیط ها

عبارتند از : قابل مشاهده بودن ^۳ ؛ قطعی / اتفاقی بودن ؛ دوره ای (اپیزودیک) و ترتیبی بودن ؛ ثابت (استاتیک) و پویا (دینامیک) بودن ؛ گستته و پیوسته بودن ؛ تک عاملی و چند عاملی بودن .

قابل مشاهده بودن : در صورتی که حسگرهای عامل همیشه اطلاعات کاملی در مورد اجزای مربوط محیط بدهنند ، محیط ، کاملاً قابل مشاهده است . به عنوان مثال ، محیط بازی شطرنج ، کاملاً قابل مشاهده می باشد . محیط جارو برقی ، به صورت جزیی قابل مشاهده می باشد (در صورتی که در فضای مجاور ، کثیفی وجود داشته باشد نمی تواند آن را بینند) .

قطعی / اتفاقی بودن : ما می توانیم فکر کنیم که جهان دارای انتقال میان وضعیت ها می باشد . در این مورد داریم ، وضعیت جاری * عملکردهای عامل ← (نتیجه می دهد) وضعیت جدید . در صورتی که انتقال وضعیت ، منحصر به فرد باشد ، جهان به صورت قطعی می باشد . به عنوان مثال ، بازی شطرنج ، بازی ای قطعی می باشد . دنیای جارو برقی ، قطعی است . رانندگی یک اتومبیل ، تصادفی می باشد .

shippers^۱

display to user^۲

observability^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نکته: در اینجا ما اجتناب می کنیم که سؤالات را با مکانیک کوانتوسی (ذره ای) موشکافی نماییم
– ممکن است که جهان، قطعی باشد، اما با توجه به پیچیدگی اش به نظر تصادفی (اتفاقی) برسد.

دوره ای و ترتیبی بودن: در حالت دوره ای، هر عملکرد به صورت مستقل می باشد و عامل، ادراک می کند، تصمیم می گیرد و عمل می نماید و دوباره [از نو] شروع می نماید و تصمیم گیری بعدی، وابسته به وضعیت های قبلی نمی باشد. به عنوان مثال، عامل فیلتر کننده ای اسپم^۱، دوره ای می باشد. در صورتی که عامل باید یک سری از عملیات را برای رسیدن به یک عمل یا رسیدن به یک هدف انجام دهد، محیط ترتیبی می باشد. در محیط ترتیبی، باید به تصمیم گیری های آینده توجه شود؛ به عنوان مثال، رانندگی یک اتومبیل، ترتیبی می باشد.

ثابت و پویا (دینامیک) بودن: یک محیط ثابت، مدامی که عامل در حال تصمیم گیری در مورد یک عملکرد می باشد، ثابت باقی می ماند و عامل برای رسیدن به یک تصمیم، تحت محدودیت یا فشار زمانی نمی باشد؛ محیط فیلتر کردن اسپم، ثابت می باشد؛ بازی شطرنج، ثابت می باشد. در یک محیط پویا، در زمانی که عامل در حال تصمیم گیری در مورد این می باشد که چه کاری را انجام بدهد، محیط تغییر پیدا می کند. عامل در این مورد باید "به اندازه ای کافی با سرعت" عمل نماید؛ به طور مثال، رانندگی یک اتومبیل، پویا می باشد. در محیط نیمه پویا، محیط تغییر نمی کند، اما معیار کارایی در طول زمان تغییر می کند. انجام یک آزمایش در زمان محدود و بازی شطرنج با یک ساعت مثال هایی از محیط های نیمه پویا می باشدند.

گسسته و پیوسته بودن: ما می توانیم در مورد گسسته و پیوسته بودن ادراک ها، عملکردهای عامل یا وضعیت های ممکن محیط، صحبت کنیم. در صورتی که مقادیر هر کدام از موارد گفته شده به صورت مجموعه ای گسسته باشد، آن گاه محیط گسسته می باشد. محیط گسسته، شبیه محیط محدود نمی

^۱ spam، پیام های نامرتب یا نامناسبی می باشند که در اینترنت برای تعداد زیادی از کاربران یا گروه های خبری فرستاده می شوند.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

باشد. یک محیط فیلتر کننده ای اسپم ، گستته می باشد ، حتی در میان تعداد (قابل شمارش) نامحدود از پیام های الکترونیکی . یک محیط پیوسته ، دارای متغیر های تغییر کننده به صورت پیوسته است . به طور مثال ، زاویه های فرمان در یک محیط رانندگی اتومبیل . در مورد حسگر خوان های با مقادیر واقعی (ما می توانیم در مورد ادراک ها در این مورد موشکافی نماییم ؛ در اینجا نکته این است که آیا یک تغییر قابل تشخیص میان دو مقدار وجود دارد یا نه) .

تک عاملی یا چند عاملی بودن : در محیط تک عاملی ، عامل ما در حال عمل بر روی خودش می باشد . به طور مثال ، محیط فیلتر کردن اسپم ، به صورت تک عاملی می باشد . در محیط چند عاملی ، عملکردها / اهداف / روش های عامل های دیگر باید به حساب آیند . مثل محیط بازی شترنج . گرچه یک جهان ممکن است دارای عامل های دیگری باشد ، اما ما ممکن است به خاطر پیچیدگی استنتاج ، با آن به صورت تک عاملی و اتفاقی رفتار نماییم . به عنوان مثال یک عامل کنترل کننده ای علایم ترافیکی .

برخی مثال ها عبارتند از : بازی شترنج ، ماشینی که در آن پول می ریزند و کالایی را دریافت می کنند^۱ ، روبوتی که در هارنی^۲ به من قهوه می دهد ، مدار مریخ^۳ ، عامل مکالمه ای و عامل تشخیص طبی .

مثال :

تخته نرد	فروشگاه اینترنتی	تاكسي	
بله	نه	نه	قابل مشاهده ؟؟

slot-machine^۱

Harney^۲: نام جایی در ایالات متحده ای آمریکا

Mars orbiter^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نه	تاریخی	نه	نتیجه‌ی فرآیند؟؟
نه	نه	نه	دوره‌ای؟؟
نیمی	نیمی	نه	ایستا؟؟
بله	بله	نه	محض؟؟
نه	بله (به جز حراج‌ها)	نه	تک مؤلفه‌ای؟؟

نوع محیط تا حد زیادی طرح عامل را تعیین می‌کند. البته دنیای واقعی اندکی قابل مشاهده، دوره‌ای، ترتیبی، پویا، دنباله دار و چند عاملی می‌باشد.

ارزیابی عملکرد (معیار کارایی)

در مورد عامل جاروبرقی، تعدادی از کاشی‌ها در یک زمان معین تمیز می‌شوند. انرژی، اغتشاش^۱، گم شدن قطعات مفید، اثاثیه‌ی خراب، خراش کف را هم در ارزیابی عملکرد در نظر می‌گیریم

عامل‌های نرم افزاری، معمولاً به صورت سافتبوت‌ها شناخته می‌شوند. در محیط‌های مصنوعی که براساس کامپیوترها و شبکه‌ها می‌باشند وجود دارند. می‌توانند خیلی پیچیده با نیازمندی‌های

noise^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

زیاد به عامل باشند و بجهانی^۱ مثالی از عامل های نرم افزاری می باشند. در این عامل ها، محیط های طبیعی و مصنوعی ممکن است با هم ادغام شده باشند.

عامل های متحرک^۲، در عامل های متحرک، برنامه ها می توانند از یک ماشین به ماشین دیگر بروند، در یک محیط اجرایی مستقل از پایگاه^۳ اجرا می شوند، به محیط اجرایی عامل نیازمندند، تغییر پذیری، لازم یا مناسب وضعیت برای عامل نمی باشد. این عامل ها، هزینه های ارتباط را کاهش می دهند. کاربرد آن ها در بازیابی اطلاعات توزیع شده و مسیریابی شبکه های مخابراتی می باشد.

عامل های اطلاعاتی^۴، رشد انفجاری اطلاعات را مدیریت می نمایند، اطلاعات چند منبع توزیع شده را به کار می گیرند یا مقایسه^۵ می نمایند. عامل های اطلاعاتی می توانند ثابت یا متحرک باشند مثل، اطلاعات درون وب یا یک سند واقعی (مثلا یک سند کاغذی)، برای صفحات تفسیر کننده های وب شناخته می شوند و منبع داده ای در داده های بی ساختارند و در آن ها پاسخگویی به سؤال با استفاده از دانش قوی متدهای آماری انجام می شود.

برنامه های محیط، شبیه سازی کننده های محیط برای آزمایش هایی که توسط عامل ها انجام می شود می باشند، یک ادراک را به یک عامل ارایه می نمایند، یک عملکرد را دریافت می نمایند و محیط را به روز می نمایند. اغلب به کلاس های محیط برای عملیات وابسته یا انواع عامل ها تقسیم می شوند و خیلی از اوقات مکانیزم هایی را برای اندازه گیری عملکرد عامل ها ارایه می نمایند.

انواع عامل ها

World Wide Web^۱

mobile agents^۲

independent-platform^۳

information agents^۴

collate^۵

مترجم: سهراب جلوه گر

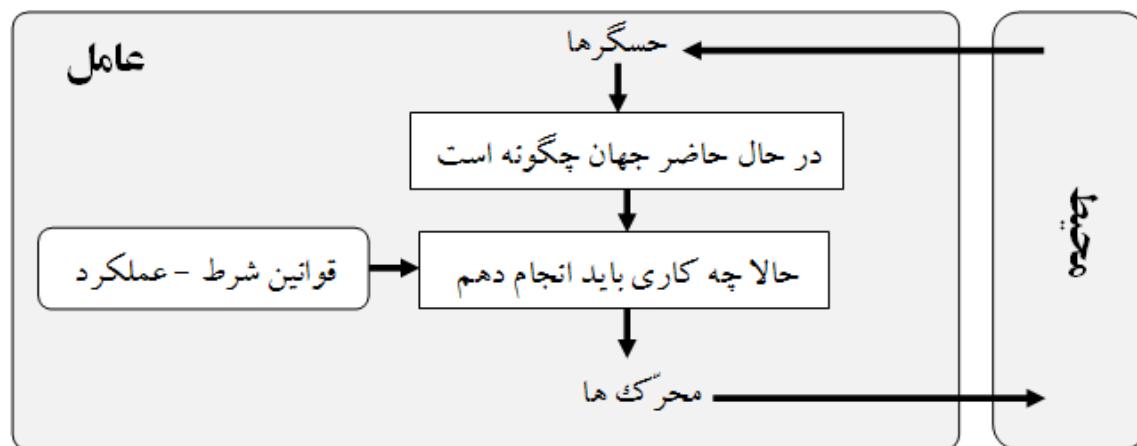
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چهار نوع اصلی وجود دارد: عامل های بازتابی ساده^۱ - عامل های بازتابی با حالت یا وضعیت یا عامل هایی که وضعیت جهان را حفظ می نمایند^۲ - عامل های براساس هدف یا هدف گرا^۳ و عامل های سودمند^۴. همه اینها می توانند در عامل های آموزشی به کار گرفته شوند.

عامل های بازتابی ساده: دارای جدول جستجوی ساده می باشند و ادراک ها را به عملیات نگاشت می نمایند (خیلی بزرگ و پرهزینه می باشند) و در آنها تعدادی از وضعیت ها می توانند توسط قانون های شرط - عملکرد خلاصه شوند. پیاده سازی این نوع عامل ها آسان می باشد ولی دارای کاربرد کمی می باشند .



مثال :تابع (Reflex-Vacuum-Agent([location,status]) یک عملکرد را برمی گرداند ، در صورتی که وضعیت برابر کثیف (Dirty) بود ، مکش (Suck) را برمی گرداند ، در غیر این صورت ،

simple reflex agents^۱
reflex agents with state^۲
goal-based agents^۳
utility-based agents^۴

مترجم: سهراب جلوه گر

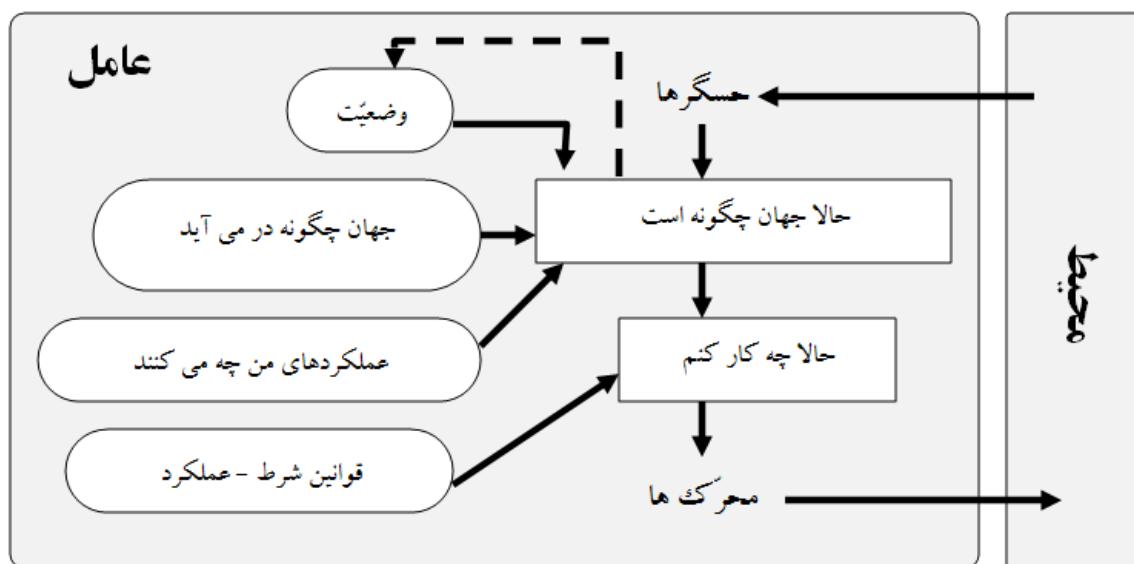
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اگر محل^۱ برابر A بود، راست (Right) را برمی گرداند، در غیر این صورت، اگر محل برابر B بود، چپ (Left) را برمی گرداند.

عامل هایی که وضعیت جهان را حفظ می کنند (عامل های بازتابی براساس مدل):
اطلاعات عامل به تنها یکی در مورد مشاهده پذیری جزئی کافی نمی باشند، لازم است که جریان تغییرات جهان را نگهداری نماییم و تکامل ها، به طور مستقل از عامل یا سبب شده توسط عملکرد عامل می باشند.



مثال : تابع (تابع (Reflex-Vacuum-Agent([location,status])) یک عملکرد را برمی گرداند

متغیرهای last_A, last_B, numbers : static دارای مقدار اولیه ∞

در صورتی که وضعیت (status = Dirty) باشد ...

location^۱

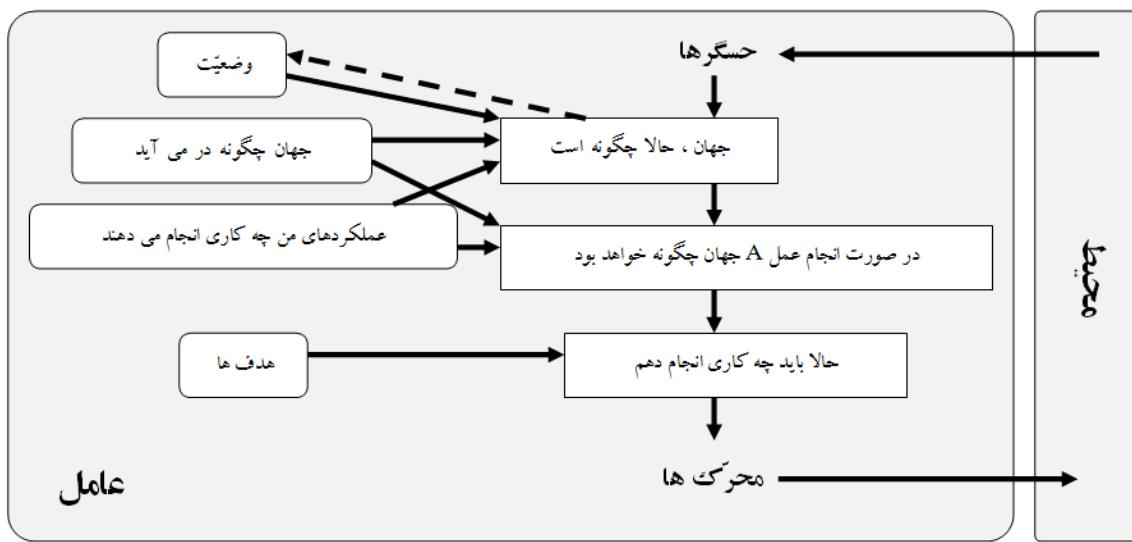
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل های هدف گرا: در این گونه عامل ها وضعیت و عملکردها نمی گویند که کجا برویم و به اهداف برای ساختن رشته ای از عملکردها (برنامه ریزی) نیازمندیم در هدف گرایی، از قوانین یکسان برای اهداف مختلف استفاده می نمایید و در بازتاب، به مجموعه ای کامل از قوانین برای هر هدف نیاز خواهیم داشت.



مطالبی در مورد عامل های هدف گرا: دانستن وضعیت فعلی محیط، همیشه کافی نمی باشد. عملکرد درست ممکن است همچنین وابسته به این که عامل در تلاش برای رسیدن به چه چیزی است باشد. عملکردهایی را که برای رسیدن به هدف ها کمک می کنند را انتخاب نمایید. جستجو و برنامه ریزی برای حل این مسئله مورد استفاده قرار می گیرند. استدلال هدف گرا برای محیط های ترتیبی خیلی مفید می باشد. مثل، بازی شطرنج، رانندگی تاکسی، خلبانی سفینه ای فضایی. عملکرد درست برای یک رشته ی ادراکی ارایه شده وابسته به دانش عامل و وضعیت جاری آن و چیزی که در حال حاضر در تلاش برای رسیدن به آن است می باشد.

عامل های سودمند: در عامل های سودمند، دانستن هدف ها ممکن است در محیط های با پیچیدگی بالا، کافی نباشند؛ چند رشته از عملکردها برای دسترسی به برخی اهداف لازمند (پردازش

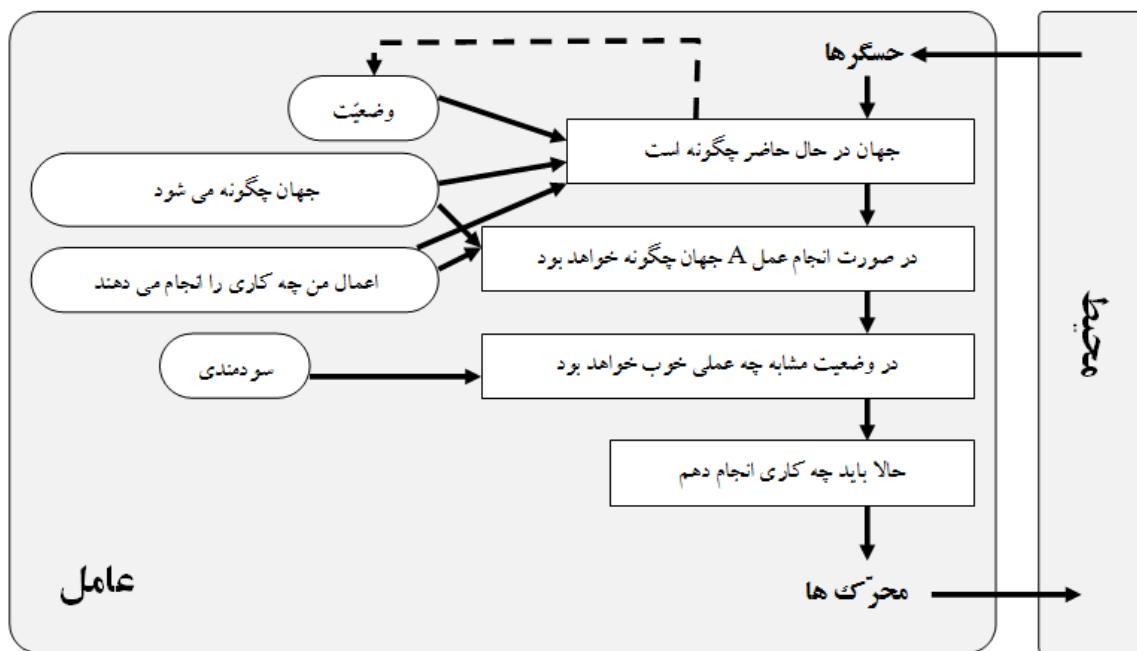
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دودویی)؛ ممکن است نیازمند انتخاب میان عملکردها و رشته ها می باشیم و سودمندی، وضعیت را به عددی حقیقی نگاشت می نماید و درجه ای ارضا را بیان می کند و سبک و سنجینی میان موارد متناقض را مشخص می نماید، از سودمندی برای مقایسه ای شرایط مطلوب وابسته به رشته های عملکرد استفاده می شود و می تواند تخمینی از هزینه، زمان، یا مقدار وابسته به نتایج مختلف باشد. سودمندی ها در محیط هایی که به صورت جزئی قابل مشاهده اند یا محیط های اتفاقی، خیلی مفید می باشند. سودمندی ها برخی اوقات یک چیز بحث انگیز در هوش مصنوعی می باشند. فرض می کنیم نتایج می توانند به صورت خطی در یک مقیاس یکسان، مرتب شده باشند. مثل: عامل رانندگی تاکسی. در این مورد، طراح باید نتایج را به طور صریح ارزیابی نماید (به صورت کمی و کیفی). سودمندی در دامنه های با احتمال، خیلی مفید می باشد، مثل: تجارت برخط، اکتشاف در محیط های دارای نامعلومی و قمار بازی^۱.



¹ gambling

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل های آموزشی: در این عامل ها عنصر آموزشی^۱، بهبودهایی را به وجود می آورد. عنصر کارایی^۲، انتخاب عملکردهای خارجی را ممکن می سازد. انتقاد^۳، جمع آوری بازخورد^۴ در مورد چگونگی عمل عامل می باشد. تولید کننده می مسئله^۵، عملکرد (آزمایش) های پیشنهادی (اکتشافی) را در نظر می گیرد. اغلب، یک عامل ممکن است نیاز به بروزرسانی برنامه ای عامل خود را داشته باشد. برنامه نویسان ممکن است فهم کاملی از محیط نداشته باشند. محیط، ممکن است در طول زمان، تغییر نماید در این مورد برنامه نویسی با دست ممکن است خسته کننده باشد. یک عامل آموزشی، عاملی است که عملکرد خود را با توجه به مجموعه ای از عملکردها در طول زمان، بهبود می بخشد. آموزش یا انطباق در محیط های پیچیده، ضروری می باشد. یک عامل آموزشی هم به عنصر عملکرد یا کارایی^۶ و هم به عنصر آموزشی^۷ نیازمند می باشد؛ عنصر کارایی، عملکرد یا عملکردهای جاری را انتخاب می نماید و عنصر آموزشی، درستی عنصر کارایی را ارزیابی می نماید.

learning element^۱

performance element^۲

critic^۳

feedback^۴

problem generator^۵

performance element^۶

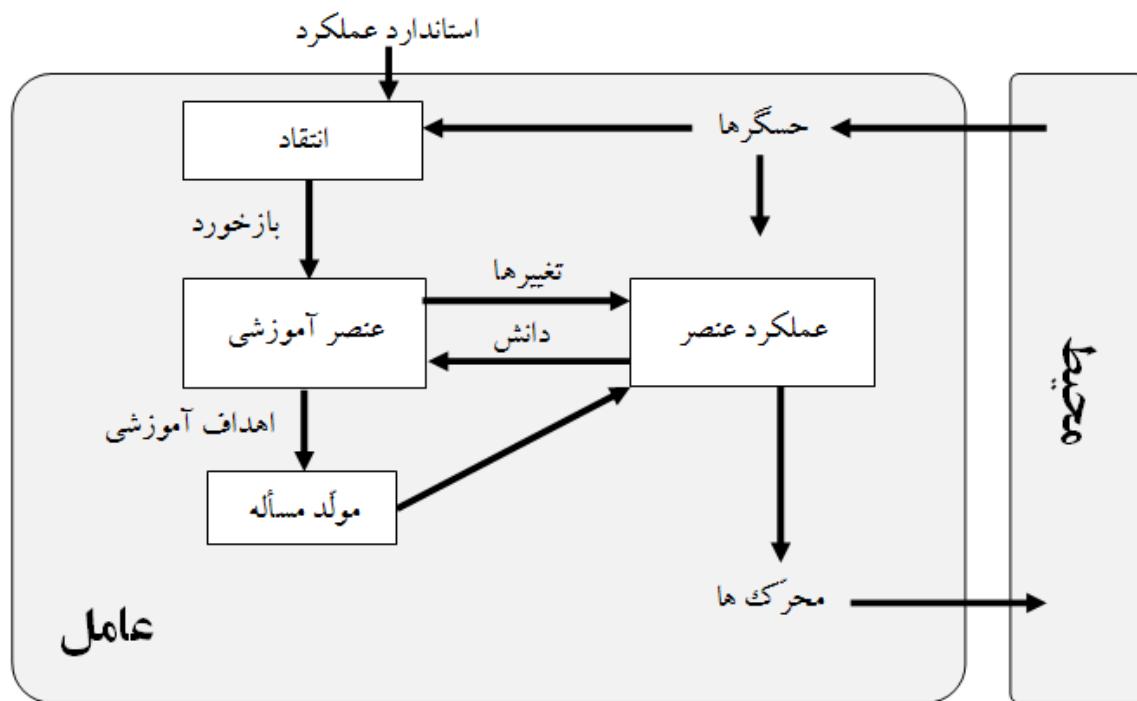
learning element^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



آموزش ، می تواند به صورت برخط^۱ یا برون از خط^۲ باشد ؛ آموزش ، ممکن است غیرفعال یا فعال باشد ؛ آموزش ممکن است نظارت شده یا نظارت نشده باشد . انتساب اعتبار^۳ در زمانی که آموزش در محیط های ترتیبی می باشد ، مسئله ای بزرگ می باشد .

نکته : اغلب ، آموزش در هوش مصنوعی به صورت یک موضوع مجزا ، عمل می کند ؛ ما برای جمع کردن آن با دیگر موضوعات ، تلاش می کنیم .

خلاصه

online^۴

offline^۵

credit assignment^۶

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل ها، روی محیط های وسیله ای محرک ها و حس گرها اثر می گذارند. تابع عامل چگونگی عملکرد عامل را در همه ای شرایط تشریح می کند. ارزیابی عملکرد (معیار کارایی) چگونگی محیط را ارزیابی می کند. یک عامل هوشمند بی عیب، عملکرد مورد انتظار را بیشینه می کند. برنامه های عامل برخی از توابع عامل را اجرا می کنند. توضیحات PEAS، محیط های کاری را تعریف می کند.

- محیط ها در چند بعد طبقه بندی می شوند: قابل مشاهده ؟ - قطعی ؟ - دوره ای ؟
پویا ؟ - گستره ؟ - تک مولفه ای ؟

چند معماری موجود عامل های پایه عبارتند از: عامل های بازتابی ساده - عامل هایی که وضعیت جهان را حفظ می نمایند - عامل های هدف گرا - عامل های سودمند

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل سوم

حل مسئله و جستجو

problem solving and search'

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

خلاصه‌ی ریوس مطالب

- عامل‌های حل‌کننده‌ی مسئله
- انواع مسئله
- فرمول بندی^۱ مسئله
- مثال‌هایی از مسایل
- الگوریتم‌های اساسی (پایه‌ای) جستجو

یک عامل هدف گرا می‌تواند بررسی کند که چه کاری را می‌تواند انجام دهد و عملکرد هایی که به نتیجه می‌رسند را انتخاب نماید. برنامه‌ی عامل از درک‌ها و هدف، به صورت ورودی استفاده می‌کند. ما به یک نوع عامل هدف گرا به نام عامل حل مسئله را بررسی می‌کنیم.

^۱ formulation

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

جستجو به عنوان روش حل مسئله

بسیاری از مسایل می توانند به صورت رسیدن به یک حالت هدف^۱ ، از یک نقطه‌ی شروع دیده شوند . فضای حالت^۲ ، مسئله و راه حل های ممکن آن را به یک صورت رسمی تری تعریف می نماید ؛ عموماً مسایل باید به صورت تجزیه شده^۳ باشند . عملکردهای عامل ، وضعیت را تغییر می دهد و فضای حالت را برای یک راه حل ، جستجو (ملاقات) می نماید . اطلاعات ، در مورد مسئله‌ی معین یا با دامنه‌ی عمومی می توانند برای بهبود جستجو استفاده شوند . این اطلاعات عبارتند از : تجربه از موارد قبلی مسئله ؛ روش‌های بیان شده به صورت اکتشافات ؛ نوع‌های ساده‌تر مسئله و محدودیت‌های وضعیت‌های مشخص (معین)^۴ مسئله .

چرا روش‌های جستجو را بررسی می نماییم ؟

روش‌های جستجو ، روش‌هایی مهم برای حل^۵ بسیاری از مسایل می باشند . استفاده از جستجو ، به یک فرمول بندی مجزای مسئله و مراحل ممکن برای به وجود آوردن راه حل ها نیازمند می باشد و الگوریتم های جستجو ، پایه‌ای برای بهینه سازی^۶ و روش‌های برنامه ریزی می باشند .

عامل‌های حل کننده‌ی مسئله

یک عامل حل کننده‌ی مسئله ، برای پیدا کردن یک رشته از عملکردها که به هدف می رساند تلاش می کند . به عنوان مثال ، چه رشته از حرکت‌ها یک مکعب را یک^۷ را حل می کنند ؟ . چگونه من از

goal state^۸

state space^۹

abstracted^{۱۰}

optimization^{۱۱}

مترجم: شهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دانشگاه فلوریدای جنوبی^۱ با اتومبیل به سمت لیورمور رانندگی نمایم؟ . چگونه من می توانم اجزای روی یک تراشه را بچینم؟ . کدام رشته از عملیات یک روبوت را در یک فضا حرکت می دهد؟ . در زیر تصویری از بازی مکعب رایک را مشاهده می نماید:



برای فرمول بندی مسئله این موارد را در نظر می گیریم: وضعیت های ممکن وابسته به دنیا برای حل مسئله کدامند؟ . چه اطلاعاتی برای عامل در دسترس می باشند؟ . رفتن عامل از یک وضعیت به وضعیت دیگر چگونه می تواند باشد؟ .

برای فرمول بندی هدف مسئله این موارد را در نظر می گیریم: وضعیت هدف چیست؟ . ویژگی های مهم وضعیت هدف چه می باشند؟ . چگونه عامل می فهمد که به هدف رسیده است؟ در این مورد بررسی می کند که آیا چند وضعیت پایانی ممکن وجود دارد؟ و آیا آن ها با هم برابرند یا برخی بهترند؟ .

جستجو

Rubik's cube^۱، یک پازل به شکل یک مکعب پلاستیکی که با مربع هایی با چند رنگ پوشانده شده است ، هر بازیگر تلاش می کند مربع ها را تغییر دهد تا این که همه مربع های هر وجهه دارای یک رنگ بشوند [انتشارات دانشگاه آکسفورد ، سال ۲۰۰۴ میلادی ، گرفته شده از لغتنامه ای Babylon University of South Florida = USF^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تعريف جستجو: پردازه‌ی رسیدگی کردن به صورت ترتیبی به عملکردها برای پیدا کردن رشته ای از عملیات که ما را از شروع به هدف می رسانند، جستجو نام دارد. یک الگوریتم جستجو، یک رشته از عملیات که برای عامل اجرا می شوند را برمی گرداند و جستجو معمولاً به صورت "برون خطی" انجام می شود. در این مورد نکته ای که باید در نظر بگیریم این است که، فرض بر این است که محیط به صورت استاتیک (ثابت) می باشد، همچنین فرض شده که محیط، گستره می باشد. محیط (ممولاً) به صورت قطعی می باشد.

برخی از مسایل جستجوی کلاسیک

مسایل سرگرمی: برای مطالعه به صورت مثال یا برای مقایسه‌ی الگوریتم‌ها، مفید می باشد؛ مثل : پازل ۸-تایی، دنیای جارو، مکعب راییک و N-وزیر.

مسایل دنیای واقعی: معمولاً کار زیادی می برند، اما جواب معمولاً جالب می باشد. مثل : مسیر یابی ، مسافرت شخص دوره گرد ، طرح VLSI و جستجو در اینترنت

وضعیت

ما معمولاً در مورد وضعیتی که یک عامل در آن می باشد صحبت می کنیم. وضعیت، به مقادیر متغیرهای مربوط توصیف کننده‌ی محیط و عامل، اشاره می کند. به عنوان مثال در دنیای جارو، وضعیت ($t=0$)، تمیز می باشد. در مسئله‌ی رومانی، در $t=0$ در بخارست هستیم و در مکعب راییک، وضعیت، نظم فعلی مکعب (چگونگی قرار گرفتن مریع‌های روی مکعب در حال حاضر) می باشد.

فرمول بندی مسئله

فرمول بندی مسئله شامل موارد زیر است :

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- ۱ **شناختن نوع مسئله:** این که عامل چه دانشی در مورد وضعیت جهان و نتیجه‌ی عملکرد خودش دارد؛ آیا اجرای عمل نیازمند به روز کردن اطلاعات دارد؟ و تعریف وضعیت‌های جهان.

- ۲ **توضیح رسمی برای عملکرد عامل:** توضیح هدف و عملگرها که همان عملکردهای عامل می‌باشد.

انتخاب وضعیت‌ها و عملکردها

مسایل برای حل باید **محضًا** باشند. وضعیت‌های قابل شناخت در مدت فرایند حل مسئله باید توضیح داده شوند. عملکردها (عملگرها) عامل را از وضعیتی به وضعیت دیگر می‌برند و وابسته به وضعیت‌ها، توانایی عامل و خصوصیات محیط می‌باشند و انتخاب وضعیت‌ها و عملگرها مناسب، می‌تواند باعث تفاوت میان این که یک مسئله می‌تواند حل شود یا نمی‌تواند حل شود بشود.

أنواع مسائل

مسایل تک حالته^۱: در صورتی که مسئله قطعی و کاملاً قابل مشاهده باشد، در نتیجه، مسئله، تک حالته خواهد بود؛ در این حالت، عامل دقیقاً می‌داند در کدام حالت قرار خواهد گرفت و راه حل به صورت ترتیبی است. در این حالت، جهان در دسترس می‌باشد و اثر عملکردها را می‌دانیم و عامل، وضعیتی که بعد از یک رشته از عملیات در آن خواهد بود را می‌داند.

مسایل چند حالته^۲: در صورتی که مسئله غیرقابل مشاهده باشد، در نتیجه مسئله، تطبیقی (ترکیبی^۳) خواهد بود. عامل، ممکن است تصمیمی در مورد کجا قرار گرفتن نداشته باشد؛ راه حل (در

single-state problems^۱

multiple-state problems^۲

conformant^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

صورت وجود) ترکیبی می باشد. در مسایل چند حالت، جهان فقط تا حدودی در دسترس می باشد و عامل به چند وضعیت ممکن توجه دارد.

مسایل احتمالی^۱: در صورتی که مسئله، احتمالی (غیرقطعی^۲) و یا اندکی قابل مشاهده^۳ باشد، در نتیجه، مسئله ای احتمال می باشد. در مسایل احتمالی، در طول اجرا به مشاهده و حس کردن نیازمندیم. همچنین به درخت عملیات هم نیاز داریم.

مسایل اکتشافی^۴: در صورتی که مسئله دارای فضای حالت ناشناخته باشد، در نتیجه، مسئله، اکتشافی می باشد. در مسایل اکتشافی، عامل نتایج عملکردش را نمی داند و آزمایش های عامل برای کشف وضعیت های جهان و اثرات عملکردها می باشد.

مسایل خوب تعریف شده

در مسایل خوب تعریف شده، وضعیت اوّلیه، نقطه ای شروع که عامل از آن شروع می کند می باشد، مثلا در مورد مسئله ای کشور رومانی، وضعیت اوّلیه، شهر آراد می باشد. فضای حالت، مجموعه ای از همه ای وضعیت هایی که از وضعیت اوّلیه توسط رشته ای از عملکردها قابل دسترسی می باشند است و مسیر، رشته ای از عملکردها که راهنمایی کننده از یک وضعیت به وضعیت دیگر می باشد هستند. عملکردها (عملگرها و توابع جانشین^۵) می گویند که عامل چه عملکردهایی را می تواند داشته باشد؟. عملکردها،

contingency problems^۱

nondeterministic^۲

partially observable^۳

exploration problems^۴

successor functions^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

توضیح مجموعه ای از عملکردهای ممکن می باشد ، مثلا در مورد جارو ، چپ ، راست ، بالا ، پایین ، مکش و هیچ کار می تواند باشد . آزمایش هدف ، تشخیص می دهد که آیا یک وضعیت ارایه شده ، یک وضعیت هدف می باشد . مسیری از وضعیت اوّلیه به یک وضعیت هدف می باشد .

تابع جانشین

تابع جانشین برای یک وضعیت داده شده ، یک مجموعه از جفت های عملکرد / وضعیت جدید را بر می گرداند و به ما می گوید که برای یک وضعیت داده شده ، چه کارهایی را می توانیم انجام دهیم و در چه جاهایی مقدم هستند . در یک جهان قطعی ، هر عملکرد با یک وضعیت منفرد ، جفت شده است . مثلا در دنیای جارو برقی :

$(In(0,0)) \rightarrow ('Left', In(0,0)), ('Right', In(0,0)), ('Suck', In(0,0)), 'Clean')$

در مسئله‌ی رومانی :

$In(Arad) \rightarrow ((Go(Timisoara), In(Timisoara)), (Go(Sibiu), In(Sibiu)), (Go(Zerind), In(Zerind)))$

در جهان های اتفاقی ، یک عملکرد ، شاید با تعدادی وضعیت ها ، جفت شده باشد .

آزمون هدف

آزمون هدف ، تشخیص می دهد که آیا یک وضعیت داده شده ، یک وضعیت هدف است . در این مورد باید توجه داشته باشیم که شاید یک وضعیت هدف منحصر به فرد وجود داشته باشد یا تعدادی . مثلا در دنیای جاروبرقی ، هر فضای تمیز در بازی شترنج ، مات و در مسئله‌ی رومانی ، در بخارست بودن وضعیت هدف می باشند .

هزینه‌ی مسیر

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هزینه‌ی مسیر، هزینه‌ی یک عامل برای رفتن از وضعیت اولیه به وضعیتی که به تازگی بررسی شده است می‌باشد و اغلب، مجموع هزینه برای هر عملکرد می‌باشد و هزینه‌ی مرحله^۱ نام دارد. ما فرض می‌کنیم که هزینه‌های مراحل، مثبت (غیرمنفی) هستند.

فضای حالت

ترکیب حالت‌های مسئله و توابع جانشین (راه‌های رسیدن به وضعیت‌ها) منجر به مفهوم فضای حالت می‌شود و گرافی است که همه‌ی وضعیت‌های ممکن جهان و انتقال‌های میان آن‌ها را ارایه می‌نماید. ما اغلب در مورد اندازه‌ی این فضاهای به صورت یک معیار سختی مسئله صحبت می‌کنیم. به عنوان مثال در پازل ۸-تایی فضای حالت برابر است با: $\frac{9!}{2} = 181,000$ حالت که فضای حالتی ساده است. در مورد پازل ۱۵-تایی، فضای حالت برابر است با: تقریباً 1.3×10^{24} حالت که تقریباً ساده است. در مورد پازل ۲۰-تایی: تقریباً 2.43×10^{18} وضعیت که خیلی پیچیده می‌باشد.

عملیات حل مسئله

هزینه‌ی مسیر، هزینه‌ی عامل را برای اجرای عملیات در یک مسیر تعیین می‌نماید و مجموع هزینه‌های عملیات فردی در یک مسیر می‌باشد. هزینه‌ی جستجو، زمان و حافظه‌ی لازم برای محاسبه‌ی یک راه حل است و داریم $\text{مجموع هزینه} = \text{هزینه مسیر} + \text{هزینه جستجو}$. برای مسایل چند حالتی ما از **مجموعه‌ای از وضعیت‌ها** استفاده می‌نماییم و هزینه‌ها و دیگر تعریف‌ها تغییر نمی‌کنند. شکل محدود شده‌ی کلی، دارای الگوریتم زیر می‌باشد:

step cost^۱

^۲ در انگلیسی برابر است با عدد ۱۸ صفر در جلوی آن قرار داده شود. (گرفته شده از Babylon)

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تابع (Simple-Problem-Solving-Agent(percept) یک عملکرد را بر می گرداند

: static متغیر

seq ، که یک ترتیب عملکرد است و در ابتدا دارای مقدار تهی می باشد

state ، توصیف وضعیت فعلی جهان

goal ، یک هدف که به صورت اوّلیه دارای مقدار NULL می باشد

problem ، یک فرمول بندهی مسئله

state \leftarrow Update-State(state,percept)

در صورتی که seq ، حالی است کارهای زیر را انجام بده

goal \leftarrow Formulate-Goal(state)

problem \leftarrow Formula-Problem(state,goal)

seq \leftarrow Search(problem)

(پایان شرط)

action \leftarrow Recommendation(seq,state)

seq \leftarrow Remainder(seq,state)

عمل (action) را برگردان

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نکته: این، راه حلی بروون خطی^۱ می باشد؛ حل به صورت "چشم بسته" اجرا می شود. راه حل های برخط^۲ شامل عمل کننده‌ی بدون دانش تمام می شوند.

استراتژی های جستجو - یک استراتژی با چیدن گره‌ها به صورت مرتب تعریف می شود.
استراتژی‌ها به وسیله‌ی موارد زیر ارزیابی می شوند: تمامیت^۳ - آیا همیشه، اگر راه حل موجود باشد آن را پیدا می کند؟ | پیچیدگی زمانی - تعداد گره‌های به وجود آمده / توسعه داده شده | پیچیدگی فضای - بیشینه‌ی تعداد گره‌های موجود در حافظه | بهینگی - آیا همیشه، کم هزینه ترین راه را پیدا می کند؟ | زمان و پیچیدگی فضای توسط پارامترهای زیر ارزیابی می شوند: b - ماکریم عوامل منشعب شده از درخت جستجو d - عمق کم هزینه ترین راه حل . m - ماکریم عمق فضای حالت (ممکن است بی نهایت باشد).

روش های ناآگاهانه^۴ - این روش‌ها فقط از داده‌های قابل دسترس از تعریف مسئله استفاده می کنند.

پیچیدگی

در مورد پیچیدگی الگوریتم‌ها نگران هستیم، زیرا یک مسئله ممکن است در حالت کلی قابل حل باشد اما در حالت عملی خیلی طولانی باشد. با استفاده از تحلیل مجانبی^۵، تخمین زمان (یا تعداد عملیات) لازم برای حل یک نمونه از اندازه‌ی n یک مسئله در زمانی که n به سمت بی نهایت می رود می توان پیچیدگی الگوریتم‌ها را ارزیابی نمود.

offline^۱

online^۲

completeness^۳

uninformed search^۴

asymptotic^۵

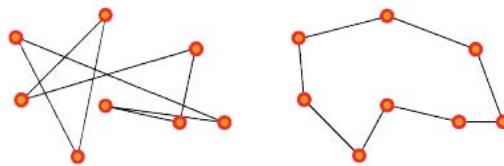
مترجم: شهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مثال : مسأله مسافت فروشنده ی دوره گرد^۱ - در این مسأله ، n شهر توسط جاده به هم متصل می باشند و مسأله ، پیدا کردن یک مسیر که از تمام شهرها عبور کند و حتی الامکان کوتاه ترین مسیر باشد است .



سختی ما در این مورد این است که الگوریتم های شناخته شده دارای پیچیدگی نمایی هستند و تعداد عملیات به صورت نمایی رشد می کند .

چرا پیچیدگی نمایی "مشکل" است ؟

این به آن معنی است که تعداد عملیات لازم برای محاسبه ی راه حل دقیق مسأله به صورت نمایی با اندازه ی مسأله رشد می کند (در زیر تعداد شهرها آورده شده است) .

- $\exp(1)=2.72$
- $\exp(10)=2.20 \times 10^4$
- $\exp(100)=2.69 \times 10^{43}$
- $\exp(500)=1.40 \times 10^{217}$
- $\exp(250,000)=10^{108,573}$

در حالی که در کامپیوترهای سریع ما 10^{12} عملیات در ثانیه را می داریم . بنابراین ، در حالت کلی ، مسایل با پیچیدگی نمایی در همه ی موارد قابل حل نمی باشند اما در اندازه های کوچک نمایی قابل حل می باشند !

Traveling Salesperson Problem (TSP)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پیچیدگی

مسایل با زمان چندجمله ای^۱ (P) : ما می توانیم الگوریتم هایی را پیدا کنیم که این مسایل را در یک زمان که به صورت چندجمله ای با اندازه ای ورودی رشد می کنند حل کنیم . برای مثال ، در مرتب کردن $n!$ عدد به صورت صعودی ، الگوریتم های ضعیف این کار را با پیچیدگی^۲ $n!$ انجام می دهند ، الگوریتم های بهتر این کار را در $\log(n)$ انجام می دهند . مسئله در اینجا این است که ما نمی توانیم وضعیت مرتبه ای چند جمله ای را مشخص کنیم ، شاید خیلی بزرگ باشد ! حال سؤال این است که آیا الگوریتم هایی که به بیش از زمان چندجمله ای نیاز داشته باشند هم وجود دارند ؟ . برای برخی از مسایل ، ما هیچ الگوریتم چندجمله ای نداریم که مسایل غیرقطعی چندجمله ای^۳ جزء این دسته هستند . برای مثال ، مسئله ای مسافت فروشنده ای دوره گرد این گونه است .

نکته ای در مورد مسایل NP-hard

یک مسئله به صورت چندجمله ای غیرقطعی است اگر یک راه حل (گمان) برای آن مسئله پیدا شده باشد . به عبارت دیگر ، الگوریتمی وجود داشته باشد که دارای زمان چندجمله ای باشد ؛ چه گمان درست باشد و چه نباشد . دشواری ما این جاست که شاید هیچ الگوریتم چندجمله ای که بتواند مسئله را حل کند وجود نداشته باشد و در صورت وجود هم ممکن است نمایش آسان نباشد و معمولاً قابل کاهش به صورت مسایل شناخته شده هم نمی باشد . در عمل ، با فرض این که $P \neq NP$ ، الگوریتم های زمان بیش تراز چندجمله ای برای حل مسئله دارند .

به دست آوردن پیچیدگی با استفاده تحلیل مجانبی

polynomial-time^۱
nondeterministic-polynomial-time(NP)^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اگر اندازه i ورودی مسئله برابر n باشد ، تعداد عملیات برای حل مسئله ، $f(n)$ باشد و $g(n)$ یک تابع ارایه شده باشد . برای برخی از مقادیر k ، تعریف $f(n) \leq k g(n)$ در صورتی که $f(n) \leq k g(n)$ باشد ، برابر $O(g(n))$ می باشد . به عنوان مثال ، $O(n^2)$ بدتر از $O(n)$ می باشد اما برای $n < 110$ $n^2 + 1 < 100n + 1000$.

یادآوری : مسایل خوب تعریف شده

در مسایل خوب تعریف شده ، وضعیت اوّلیه ، نقطه i شروعی که عامل از آن آغاز می نماید است .
فضای حالت ، مجموعه i همه i وضعیت های قابل دسترس از وضعیت اوّلیه توسط هر رشته از عملیات می باشد و مسیر ، رشته ای از عملیات مرتبط از یک وضعیت به وضعیت دیگر می باشد . عملیات (عملگرها ،
توابع جانشین) ، تشریح مجموعه ای از عملیات ممکن است . آزمون هدف ، این مطلب را که آیا وضعیت
ارایه شده یک وضعیت هدف است را تشخیص می دهد . راه حل ، مسیری از یک وضعیت اوّلیه به یک
وضعیت هدف می باشد .

یادآوری : جستجو برای راه حل ها

برای پیدا کردن راه حل ها ، فضای جستجو را بررسی می کنیم و از وضعیت اوّلیه به وضعیت نهایی (هدف) حرکت می کنیم که رشته ای مجاز از عملیات به صورت تعریف شده توسط توابع جانشین (عملگرها) می باشد .

مثال : رومانی^۱ - در یک تعطیلی در کشور رومانی ؛ در حال حاضر در شهر آراد^۲ هستیم و هواپیما فردا آراد را به مقصد شهر بخارست^۱ ترک می کند و ما می خواهیم در شهر بخارست باشیم.

Romania^۱

Arad^۲

مترجم: سه راب جلوه گر

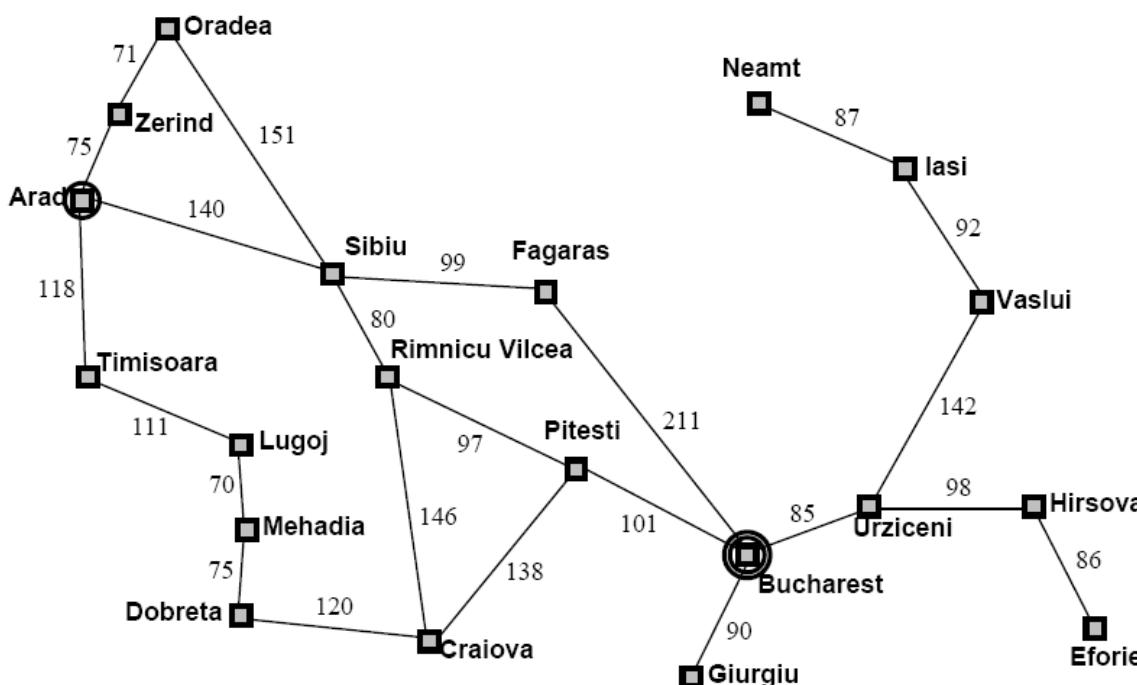
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فرمول بندی مسئله^۲: حالت ها، شهرهای مختلف هستند و عملکردها، حرکت بین دو شهر است.

پیدا کردن راه حل: ترتیب شهرها به صورت Arad , Sibiu , Fagaras , Bucharest و Timisoara باشد.



مثال : دنیای جاروبرقی - وضعیت های عامل جاروبرقی ، تمام ترکیب های دریافت ها (در که) هستند و محیط کاملاً قابل مشاهده می باشد . وضعیت های ۷ و ۸ وضعیت های هدف می باشند چون کثیف نمی باشند . فرض می کنیم برای هر عمل ، هزینه برابر ۱ باشد .

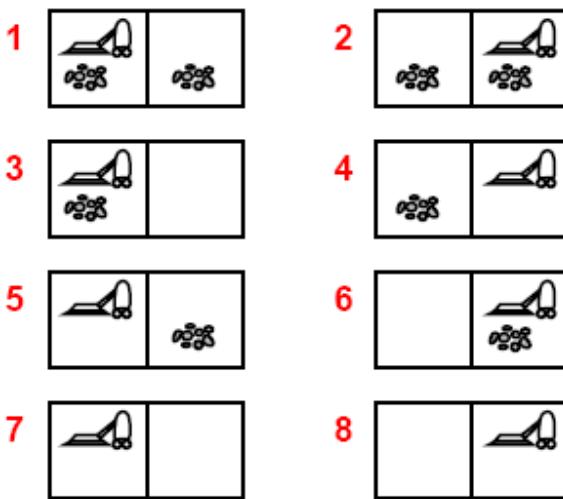
Bucharest^۱

Formulate problem^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



اگر در شماره ۱ پنج باشیم ، راه حل [Right,Suck] است . اگر در شماره های ۲ ، ۳ ، ۴ ، ۵ ، ۶ ، ۷ یا ۸ باشیم ، راه حل [Right,Suck,Left,Suck] می باشد . در اینجا باید به این نکته توجه کنیم که طبق قانون مرفری^۱ ، مکیدن (عمل مکش جاروبرقی) می تواند یک فرش تمیز را کثیف کند و حس کننده ی محلی^۲ کار تشخیص کثیف بودن و فقط تعیین محل را انجام می دهد . پس راه حل این است که [اگر کثیف بود مکش کن ، راست] یا [اگر کثیف بود مکش کن ، چپ] .

فرمول بندی مسئله ی تک حالته

یک مسئله توسط چهار عنصر زیر تعریف می شود :

حالت اولیه : در شهر Arad هستیم .

تابع جانشین : $S(x)$ =مجموعه ای از جفت حالات عملکرد ، به عنوان مثال ؛
 $S(Arad)=\{<Arad \rightarrow Zerind, Zerind>, \dots\}$

Murphy's Law^۱
local sensing^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

آزمایش هدف : می تواند ، صریح^۱ باشد مثلا ، " در بخارست " = x یا غیرصریح^۲ باشد ، مثلا ، $NoDirect(x)$ باشد .

هزینه هی مسیر : مجموعه ای از فاصله ها ، تعدادی از عملیات اجرا شده و $c(x,a,y)$ یک هزینه هی مرحله^۳ است ، طبق پیش فرض باید بزرگتر یا مساوی صفر باشد یا $c(x,a,y) >= 0$. یک راه حل ، ترکیبی از عملیات است . سیر آن ، از حالت اوّلیه به یک حالت هدف است .

انتخاب یک فضای حالت (وضعیت)

دنیای واقعی خیلی پیچیده است ، در نتیجه فضای حالت باید از راه حل مسئله جدا شود . حالت (مجرد) = مجموعه ای از حالت های واقعی است . عملکرد (مجزا) = مخلوطی پیچیده از عملکردهای واقعی ، مثلا ، "Arad → Zerind" و مجموعه ای پیچیده از مسیر های ممکن ، انحراف ها ، توقف ها و را ارایه می کند . برای تحقق پذیری ، هر حالت واقعی در شهر آراد باید برای بعضی حالت های واقعی در شهر زریند^۴ ضمانت شود . راه حل (مجزا) = مجموعه ای از مسیر های واقعی که راه حل هایی در دنیای واقعی هستند . هر عمل مجزا باید " آسان تر " از مسئله ای اصلی باشد .

جستجو در فضای حالت

بیش تر مسایل جستجو خیلی بزرگ تر از آن هستند که در حافظه جا گیرند . ما یک درخت جستجو را با شروع از وضعیت اوّلیه و به کارگیری مکرر تابع جانشین (مولد) ، تولید می نماییم . ایده ای اصلی این است که از یک وضعیت ، توجه کنید که چه چیزهایی می تواند انجام شود . سپس توجه کنید که از هر یک

explicit ^۱
implicit ^۲
step cost ^۳
zerind ^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

از وضعیت های آن ها چه چیزهایی می توانند انجام بگیرند. در اینجا سؤالاتی به وجود می آید که برخی از آن ها عبارتند از، آیا پیدا کردن راه حل بهینه، ضمانت شده است؟ تا کی، جستجو انجام خواهد شد؟ به چه مقدار از فضای نیاز خواهیم داشت؟

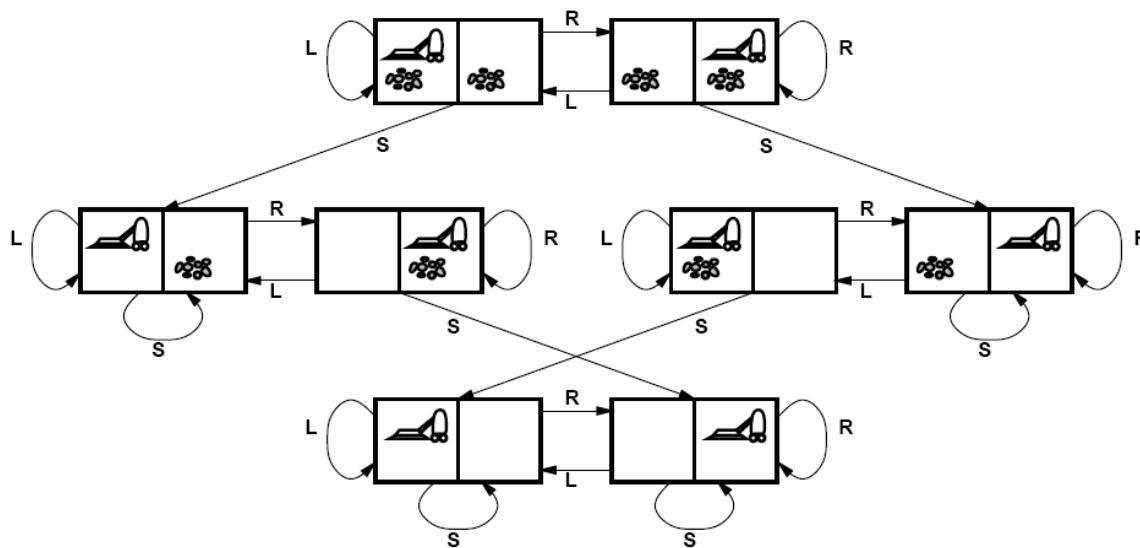
مثال: گراف فضای حالت جاروبرقی

وضعیت ها؟؟ کیفی واقعی و جای رو بوت (بدون توجه به میزان کیفی و)

عملکردها؟؟ چپ (Left)، راست (Right)، مکش (Suck)، بدون عملکرد (NoOp)

آزمون هدف؟؟ بدون کیفی بودن

هزینه های مسیر؟؟ در هر عملکرد یک بار (صفر برای بدون عملکرد (NoOp))



مثال: پازل هشت تایی

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

1	2	3
4	5	6
7	8	

وضعیت هدف

7	2	4
5		6
8	3	1

وضعیت شروع

حالت ها ؟؟ کاشی ها (بدون توجه به وضعیت های میانی)

عملکردها ؟؟ حرکت کردن در کاشی های خالی چپ ، راست ، بالا ، پایین

آزمایش هدف ؟؟ = وضعیت هدف (داده شده)

هزینه ای مسیر ؟؟ در هر حرکت یک بار

نکته : راه حل معمولی پازل n - تایی ، NP-hard می باشد. مسایل مسایلی هستند که در یک زمان چند جمله ای نمی توانند حل شوند . ^۱

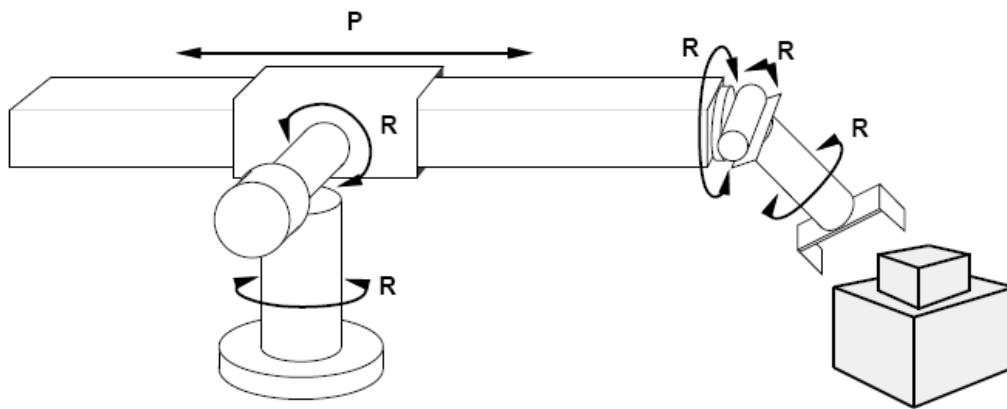
مثال : سرهم بندی روبوتیک

lorrie.cranor.org/pubs/diss/node1.html ^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



حالات ^۱؟؟ مختصات ^۲ مقدار دهی صحیح شده ^۳ تکه های متصل شده ^۴ روبوت برای سرهم

بندی

عملکردها ^۵؟؟ حرکات ^۳ پیوسته ^۴ اتصالات ^۵ روبوت

آزمایش هدف ^۶؟؟ کامل کردن سرهم بندی بدون این که ربوت را شامل شود!

ارزیابی مسیر ^۷؟؟ زمان اجرا کردن

تعریف: لیستی از گره ها که تولید شده اند اما هنوز توسعه داده نشده اند را

fringe ^۸ نامیم.

الگوریتم های جستجوی درختی

coordinates ^۱

real-valued ^۲

motions ^۳

continuous ^۴

joints ^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

الگوریتم :

تابع (TREE-SEARCH(problem , strategy) تابع (problem , strategy) یک راه حل یا عدم موفقیت را بر می گرداند .

در ابتدا درخت جستجو از حالت اوّلیه ای مسئله استفاده می کند .

در حلقه

اگر کاندید یا داوطلبی برای توسعه نبود عدم موفقیت را برگردان

یک گره ^۱ برگ ^۲ را با توجه به استراتژی یا روش انتخاب کن

اگر محتوای گره یک حالت هدف بود ، یک راه حل متناظر را برگردان

در غیراین صورت گره را توسعه بده و گره های منتج را به درخت جستجو اضافه کن

پایان حلقه

مثال جستجوی درختی

درخت جستجو؛ در گراف جستجو ، یک وضعیت می تواند از چند مسیر قابل دسترسی باشد و ریشه یک گره جستجو متناظر با وضعیت اوّلیه (آراد) می باشد

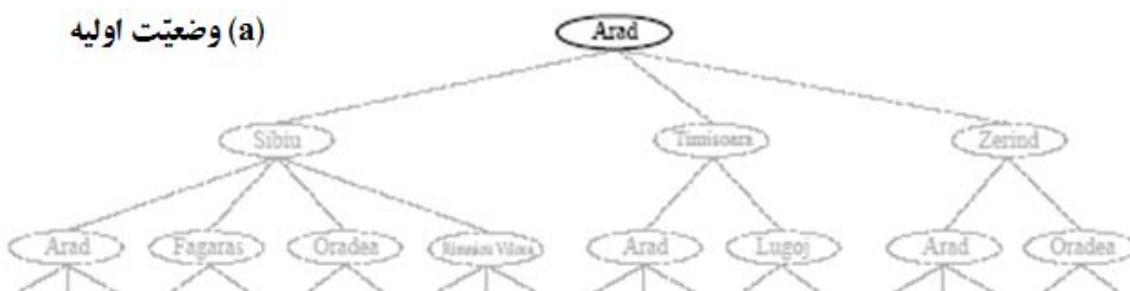
node^۱
leaf^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

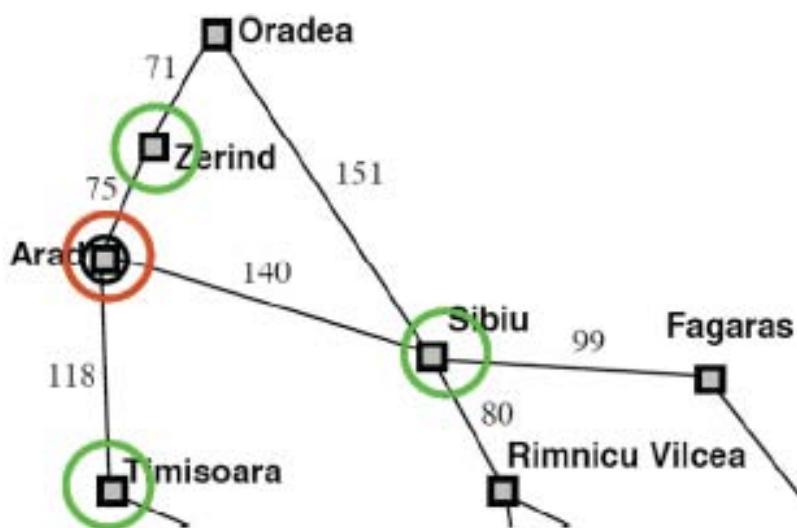


هوش مصنوعی

وضعیت اولیه (a)



شهرهای متصل شده به وضعیت اولیه عبارتند از: سیبیو^۱، تیمیسوارا^۲ و زریند و کاندیدهای حرکت بعدی در شکل زیر نشان داده شده اند:



Sibiu^۱

Timisoara^۲

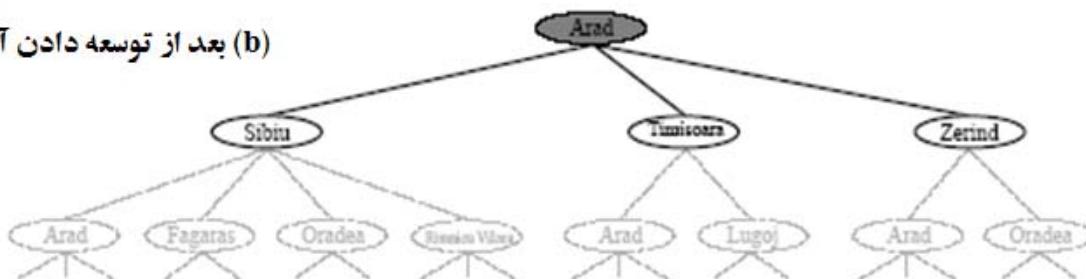
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

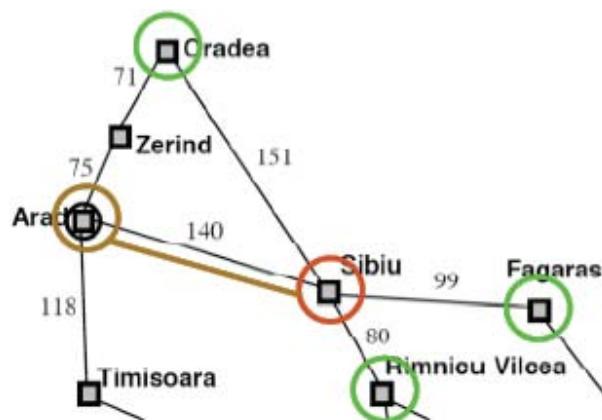


هوش مصنوعی

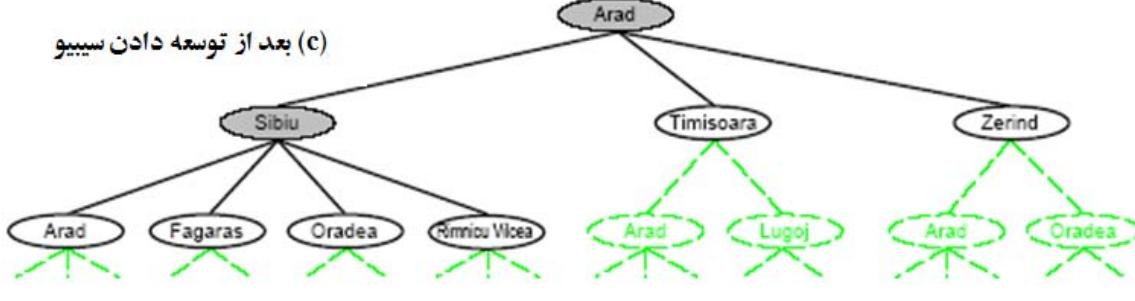
(b) بعد از توسعه دادن آراد



کاندیداهای حرکت بعدی عبارتند از: آراد ، فاگاراس^۱ ، اُرادیا^۲ و ریمنیکو ویلک^۳



(c) بعد از توسعه دادن سبیو



Fagaras^۱

Oradea^۲

Rimnicu Vilc^۳

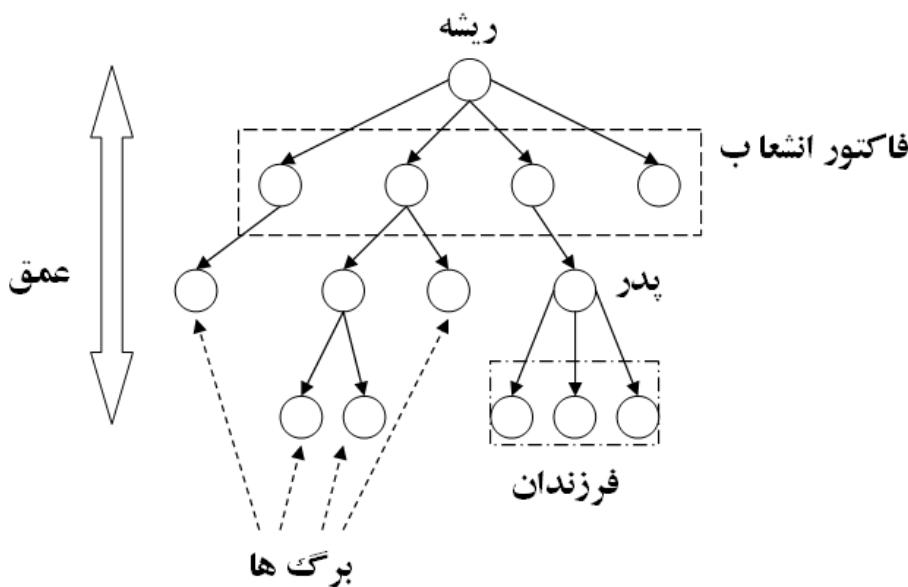
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

درخت ها

یک درخت، یک گراف^۱ بدون حلقه^۲ مستقیم^۳ می باشد. گراف مستقیم، مجموعه ای از گره های متصل شده توسط لبه ها (یال ها)^۴ می باشد. مسیر، رشته ای از لبه ها (که امکان دارد تکرار هم شده باشند) است. در درخت، حداقل یک مسیر متصل کننده^۵ هر جفت از گره ها وجود دارد (بدون حلقه است). درخت ها به دلیل این که رفتار الگوریتمی خوبی دارند به صورت گسترده ای در علم کامپیوتر استفاده می شوند و دارای قدرت زیاد برای تعریف الگوریتم های بازگشتی می باشند. در زیر تصویری از یک درخت را مشاهده می نمایید:



```

graph '
acyclic '
direct '
edges '

```

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

توجه کنید که گره ها حداکثر دارای یک پدر هستند و ریشه، گره ای متمایز است که دارای پدری نمی باشد در ضمن، عمق هم که برای خاتمه‌ی الگوریتم، مهم است می تواند نامحدود باشد.

الگوریتم جستجوی درختی

الگوریتم

تابع General-Search(problem,strategy) یک راه حل یا عدم موفقیت را به وجود می آورد

درخت جستجو را با استفاده از حالت اولیه‌ی مسئله مقداردهی اولیه می نماید

در حلقه کارهای زیر را انجام بده

در صورتی که داوطلبی برای توسعه وجود ندارد، عدم موفقیت را برگردان

یک گره برگ را برای توسعه با توجه به روش انتخاب نما

در صورتی که گره شامل یک وضعیت هدف می باشد راه حل متناظر را برگردان

در غیر این صورت گره را توسعه بده و گره های نتیجه شده را به درخت جستجو اضافه نما

پایان الگوریتم

روش: روش جستجو براساس ترتیبی که براساس آن گره ها توسعه داده می شوند مشخص می شود.

وضعیت ها و گره های درخت

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک حالت ، یک پیکربندی فیزیکی (نمایشی از یک پیکربندی فیزیکی) است . یک گره ، ساختمان داده ای است که جزء اصلی یک درخت جستجو ، شامل والد^۱ ، فرزندان^۲ ، عمق^۳ و ارزیابی مسیر (Xg) است . **حالت ها یا وضعیت ها** ، پدر (والد) ، فرزند ، عمق یا ارزیابی مسیر ندارند . از یک تابع مثل EXPAND برای ساختن گره های جدید می توانیم استفاده می کنیم و از SuccessorFn می توان برای ساختن حالت های مورد نظر در مسأله استفاده کرد .

الگوریتم جستجوی درخت عمومی

الگوریتم

تابع Tree-Search(problem,fringe) یک راه حل و یا اشکال را بر می گرداند .

$\text{fringe} \leftarrow \text{Insert}(\text{Make-Node}(\text{Initial-State}[problem]),\text{fringe})$

در حلقه کارهای زیر را انجام بده

در صورتی که fringe خالی است ، عدم موقتیت را برگردان

$\text{node} \leftarrow \text{Remove-Front}(\text{fringe})$

در صورتی که Goal-Test(problem,State(node)) ، گره را برگردان

$\text{fringe} \leftarrow \text{InsertAll}(\text{Expand}(\text{node},\text{problem}),\text{fringe})$

parent^۱
children^۲
depth^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تابع $\text{Expand}(\text{node}, \text{problem})$ مجموعه ای از گره ها را برمی گرداند

یک مجموعه ای خالی $\text{successors} \leftarrow$

برای هر عملکرد و نتیجه در $\text{Successor-Fn}(\text{problem}, \text{State}[\text{node}])$ کارهای زیر را انجام بده

یک گرهی (Node) جدید $\leftarrow s$

$\text{Parent-Node}[s] \leftarrow \text{node}$; $\text{Action}[s] \leftarrow \text{action}$; $\text{State}[s] \leftarrow \text{result}$

$\text{Path-Cost}[s] \leftarrow \text{Path-Cost}[\text{node}] + \text{Step-Cost}(\text{node}, \text{action}, s)$

$\text{Depth}[s] \leftarrow \text{Depth}[\text{node}] + 1$

S را به successors اضافه کن

Successors را برگردان

پیاده سازی الگوریتم های جستجو

تابع $\text{General-Search}(\text{problem}, \text{Queuing-Fn})$ یک راه حل یا عدم موفقیت را برمی گرداند

$\text{fringe} \leftarrow \text{make-queue}(\text{make-node}(\text{initial-state}[\text{problem}]))$

در حلقه کارهای زیر را انجام بده

اگر fringe خالی می باشد عدم موفقیت را برگردان

$\text{node} \leftarrow \text{Remove-Front}(\text{fringe})$

در صورتی که $\text{Goal-Test}[\text{problem}]$ به کار گرفته شده برای $\text{state}(\text{node})$ موفق

شد node را برگردان

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

`fringe ← Queuing-Fn(fringe,Expand(node,Operators[problem]))`

یک تابع صفت بندی است که مجموعه ای از عناصر را به Queueing-Fn(queue,elements) صف وارد می نماید و ترتیب توسعه ای عناصر را تشخیص می دهد . تنوعات تابع صفت بندی ، تنوعات الگوریتم جستجو را به وجود می آورد .

روش های جستجو

یک روش با انتخاب ترتیبی از توسعه ای گره تعریف می شود .

ارزیابی روش ها :

تمامیت : اگر راه حلی وجود داشته باشد همیشه روش ، آن را پیدا می نماید ؟

پیچیدگی زمانی : تعداد گره های تولید شده / توسعه داده شده

پیچیدگی فضایی : بیشینه ای تعداد گره های موجود در حافظه

بهینگی : آیا همیشه روش ، کم هزینه ترین راه حل را پیدا می کند ؟

پیچیدگی زمانی و فضایی با این موارد ارزیابی می شوند : بیشینه ای فاکتور انشعاب درخت جستجو ، عمق کم هزینه ترین راه حل ، بیشینه ای عمق درخت جستجو که شاید بی نهایت باشد .

جستجوی نا آگاهانه

ساده ترین الگوریتم های جستجو ، آن هایی هستند که هیچ اطلاعات اضافی که در شرح مسئله وجود دارد را استفاده نمی نمایند ، ما این روش ها را روش های جستجوی نا آگاهانه می نامیم . برخی اوقات

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

، این روش ها ، روش ضعیف هم نامیده می شوند . در صورتی که ما دارای اطلاعات اضافی در مورد چگونگی یک وضعیت غیر هدف باشیم ، در این صورت ما می توانیم جستجوی مکاشفه ای را انجام دهیم .

روش های جستجوی نآگاهانه- چنان که گفته شد ، فقط از اطلاعات قابل دسترسی از تعریف مسئله استفاده می نمایند . شامل جستجوی اوّل سطح^۱ | جستجوی با هزینه‌ی یکسان^۲ | جستجوی اوّل عمق^۳ | جستجوی با عمق محدود شده^۴ | جستجوی عمیق شونده‌ی تکراری^۵ می باشند .

جستجوی اوّل سطح

با توسعه دادن یک گره کار می کند ، سپس هر یک از فرزندانش را توسعه می دهد ، سپس هر یک از فرزندان آن ها را توسعه می دهد و در این روش ، همه ی گره های در عمق n ، قبل از یک گره در عمق $n+1$ ملاقات می شوند . ما می توانیم جستجوی اوّل سطح را با استفاده از یک صف ، پیاده سازی نماییم . به عبارت دیگر ، جستجوی اوّل سطح ، کم عمق ترین گره توسعه داده نشده را توسعه می دهد . در پیاده سازی ، ریشه^۶ یک صف FIFO^۷ می باشد و جانشینان جدید به انتهای می روند . مثال :

Breadth-first^۱

Uniform-cost^۲

Depth-first^۳

Depth-limited^۴

Iterative deepening^۵

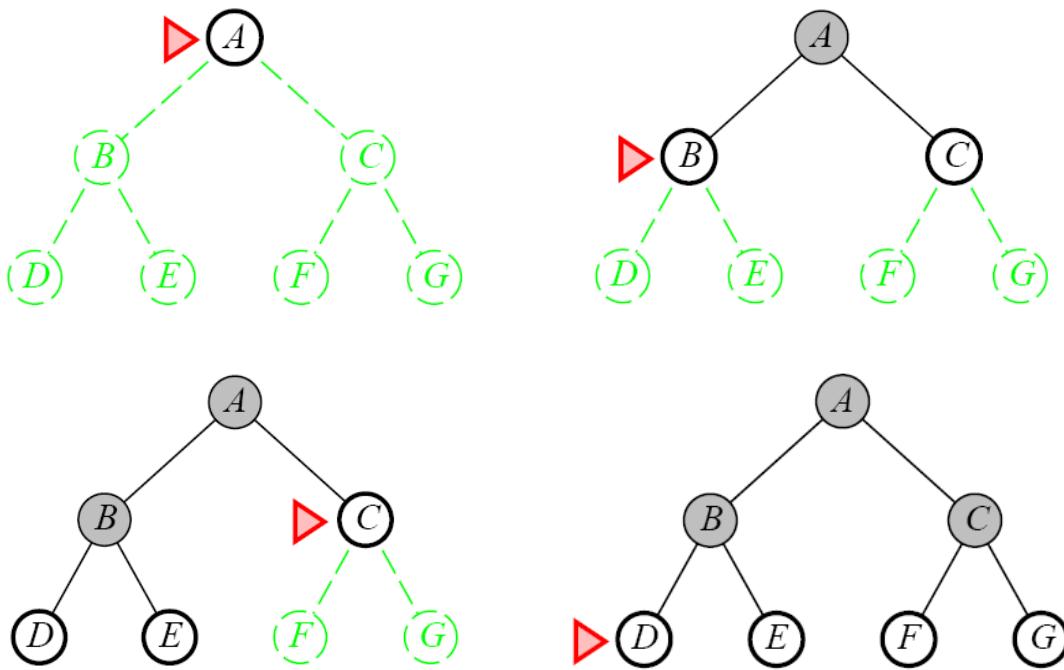
fringe^۶

First-In-First-Out^۷

مترجم: سهراب جلوه گر
ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی



مثال دیگر : آراد به بخارست

- آراد را از صف بردارید
- سیبو ، تیمیسووارا و زریند را به صف ، اضافه نمایید
- سیبو را از صف بردارید و تست نمایید
- ارادیا ، فاگاراس ، ریمنیکو ویلسیا را به صف ، اضافه نمایید
- تیمیسووارا را از صف بردارید و تست نمایید
- لوگاج را به صف ، اضافه نمایید

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

...

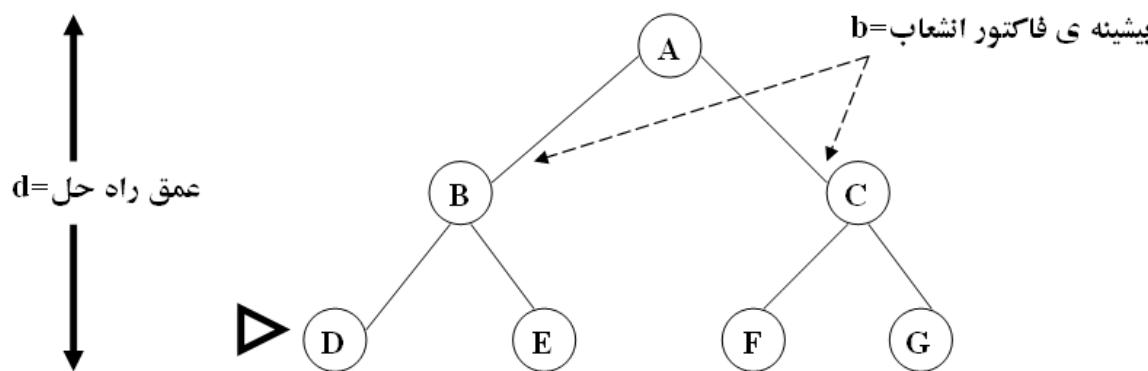
برخی نکات دیز

چگونه ما از ملاقات مجدد آراد جلوگیری نماییم؟ با استفاده از لیست closed، یک لیست از وضعیت های توسعه داده شده را نگهداری نماییم. توجه کنید که راه حل ما یک مسیر می باشد، نه یک شهر.

چگونه ما از وارد کردن دوباره ای ارادیا، اجتناب نماییم؟ open-list: یک لیست از وضعیت هایی که تولید شده اند اما، توسعه داده نشده اند.

چرا ما آزمون هدف را در زمانی که فرزندان را تولید کردیم، به کار نبردیم؟ در واقع، هیچ تفاوتی ندارد. گره ها ملاقات می شوند و به همان روش یا راه، تست می شوند.

خصوصیات جستجوی اول سطح



کامل ؟؟ آیا جستجوی اول سطح، راه حل را پیدا می کند؟ بله (در صورتی که b محدود باشد)

توضیح: تصوّر نمایید که راه حل در عمق n می باشد. از آنجایی که همه ی گره ها یا در عمق n یا در بالای n، قبل از هر چیزی در عمق n+1 ملاقات می شوند، یک راه حل، پیدا خواهد شد.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

زمان ?? جستجوی اوّل سطح ، به زمان اجرای $O(b^{d+1})$ نیاز دارد . که در آن b ، فاکتور انشعاب ، میانگین تعداد فرزندان می باشد . جستجوی اوّل سطح $(1+b+b^2+b^3+\dots+b^d+b(b^d-1))=O(b^{d+1})$ گره را ملاقات خواهد کرد (به d و چگونگی آن در فرمول توجه نمایید)

فضا ?? $(O(b^{d+1}))$ (هر گره را در حافظه نگهداری می کند)

توضیح : جستجوی اوّل سطح ، باید همه ی درخت جستجو را در حافظه نگهداری نماید (زیرا ما می خواهیم رشته ی عملکردها را برای رسیدن به هدف ، بدانیم) .

بهینگی ?? اگر چند راه حل وجود داشته باشد ، آیا جستجوی اوّل سطح ، بهترین راه حل را پیدا می کند ؟ بله

توضیح : جستجوی اوّل سطح ، کم عمق ترین راه حل در درخت جستجو را پیدا می نماید . در صورتی که هزینه ی مراحل ، یکسان باشند ، این روش بهینه خواهد بود ، در غیر این صورت ، لزوماً بهینه نخواهد بود .

• آراد ← سیبیو ← فاگاراس ← بخارست در ابتدا پیدا خواهند شد (فاصله = ۴۵۰)

• آراد ← سیبیو ← ریمنیکو ویلسیا ← پیتسنی ← بخارست ، کوتاه تر می باشد .

(فاصله = ۴۱۸)

فضا ، مسئله ای بزرگ است ، چون که باید گره ها در حافظه نگهداری شوند ؛ فضای زیادی اشغال می کند . می تواند گره ها را با سرعت ۱۰۰ مگابایت در ثانیه ^۱ تولید کند بنابراین می تواند در بیست و چهار (۲۴) ساعت هشت هزار و ششصد و چهل گیگابایت (۸۶۴۰) تولید نماید . در کل ، فضای مورد نیاز جستجوی اوّل سطح ، مسئله ای بزرگ تر از زمان مورد نیاز می باشد .

100MB/sec^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

جستجوی با هزینه‌ی یکسان

به یاد بیاورید که جستجوی اوّل سطح در زمانی که هزینه‌ی مراحل، غیر یکسان باشد، غیربهینه می‌باشد. ما می‌توانیم این اشکال را رفع نماییم برای این کار اوّل کوتاه‌ترین مسیرها را توسعه می‌دهیم و یک هزینه‌ی مسیر را به گره‌های توسعه داده شده، اضافه می‌کنیم. از یک صفت اولویّت هم برای منظم کردن آن‌ها به صورت افزایشی بر اساس هزینه‌ی مسیر، استفاده می‌نماییم. این روش، کوتاه‌ترین مسیر را پیدا خواهد کرد. در صورتی که هزینه‌ها، یکسان باشند، این روش با جستجوی اوّل سطح، برابر می‌باشد. پس، جستجوی با هزینه‌ی یکسان، کم هزینه‌ترین گره توسعه داده نشده را توسعه می‌دهد. در پیاده‌سازی، ریشه، صفتی با ارزیابی مسیر است که کم‌ترین در اوّل صفت قرار دارد.

خصوصیات جستجوی با هزینه‌ی یکسان

کامل؟؟ بله، اگر $\text{cost} \geq \epsilon$

زمان؟؟ تعداد گره‌ها در صورتی که g کوچک‌تر یا مساوی هزینه‌ی راه حل مطلوب باشد و $O(b^{\lceil C^*/\epsilon \rceil})$ در جایی که C^* راه حل مطلوب باشد خوب است.

فضا؟؟ تعداد گره‌ها در صورتی که مقدار g کوچک‌تر یا مساوی هزینه (بها)‌ی راه حل مطلوب باشد و $O(b^{\lceil C^*/\epsilon \rceil})$

بهینگی؟؟ بله - گره‌ها را با افزایش مقدار $(n)g$ توسعه می‌دهد. عمیق‌ترین گره‌ی توسعه داده نشده را توسعه می‌دهد.

برای پیاده‌سازی، ریشه، صفتی LIFO است و جانشین‌ها را [به ترتیب] در جلو قرار می‌دهد.

جستجوی اوّل عمق

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دارای روش متفاوتی نسبت به جستجوی اول سطح می باشد و همیشه ، عمیق ترین گره را توسعه می دهد . ابتدا یک فرزند را توسعه می دهد ، سپس ، سمت چپ ترین فرزند آن را توسعه می دهد و به همین ترتیب . با استفاده از یک پشته هم می توانیم روش جستجوی اول عمق را پیاده سازی نماییم . شبه برنامه ای این روش را در زیر مشاهده می نمایید :

```

push(initialState)
do
    node = pop( )
    if goalTest(node)
        return node
    else
        children = successor-fn(node)
        for child in children
            push(child)

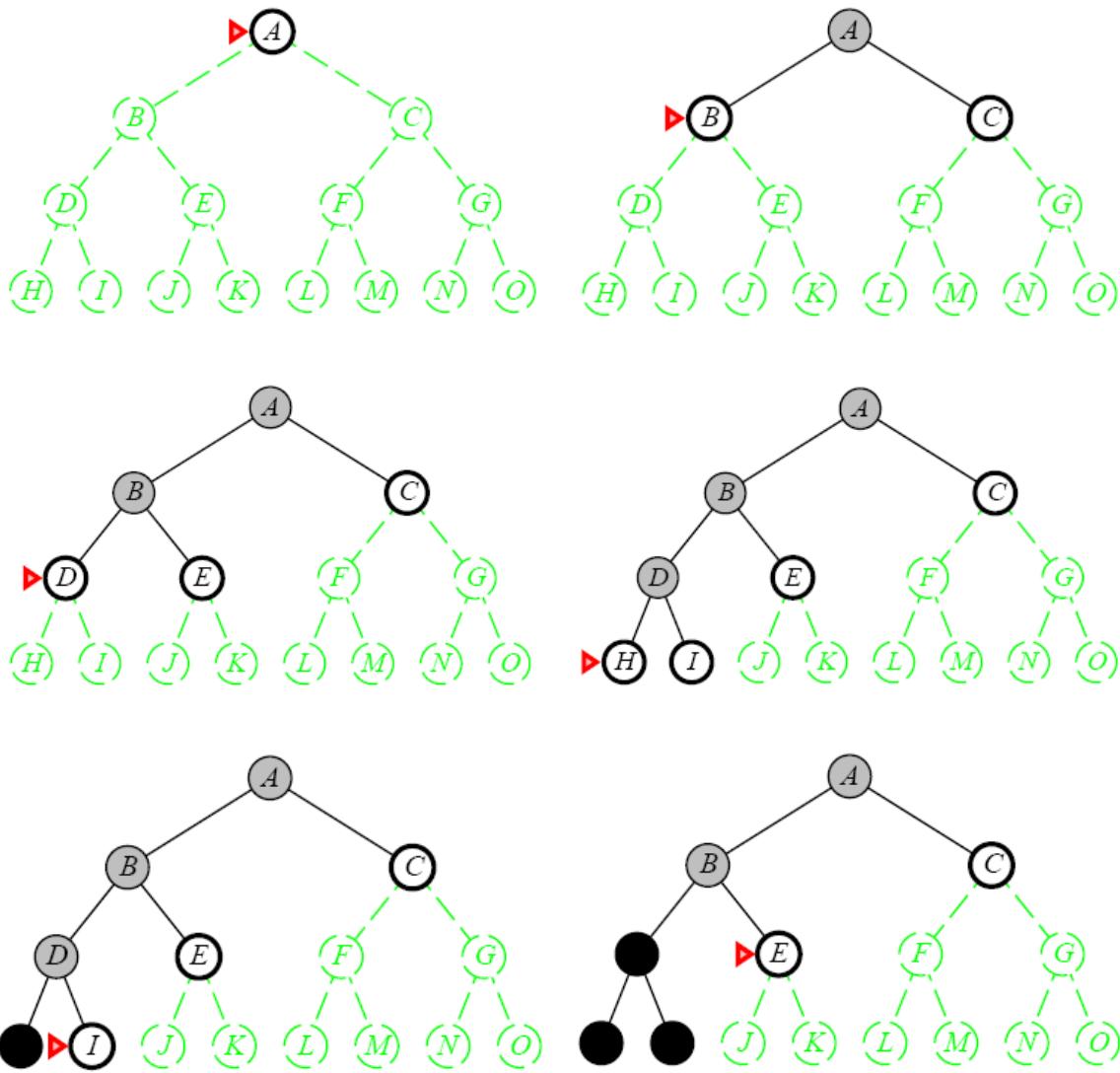
```

پس ، جستجوی اول عمق ، عمیق ترین گره ای توسعه داده نشده را توسعه می دهد . **پیاده سازی دیگر** : ریشه ، صفحی LIFO است و جانشین ها را [به ترتیب] در جلو قرار می دهد .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



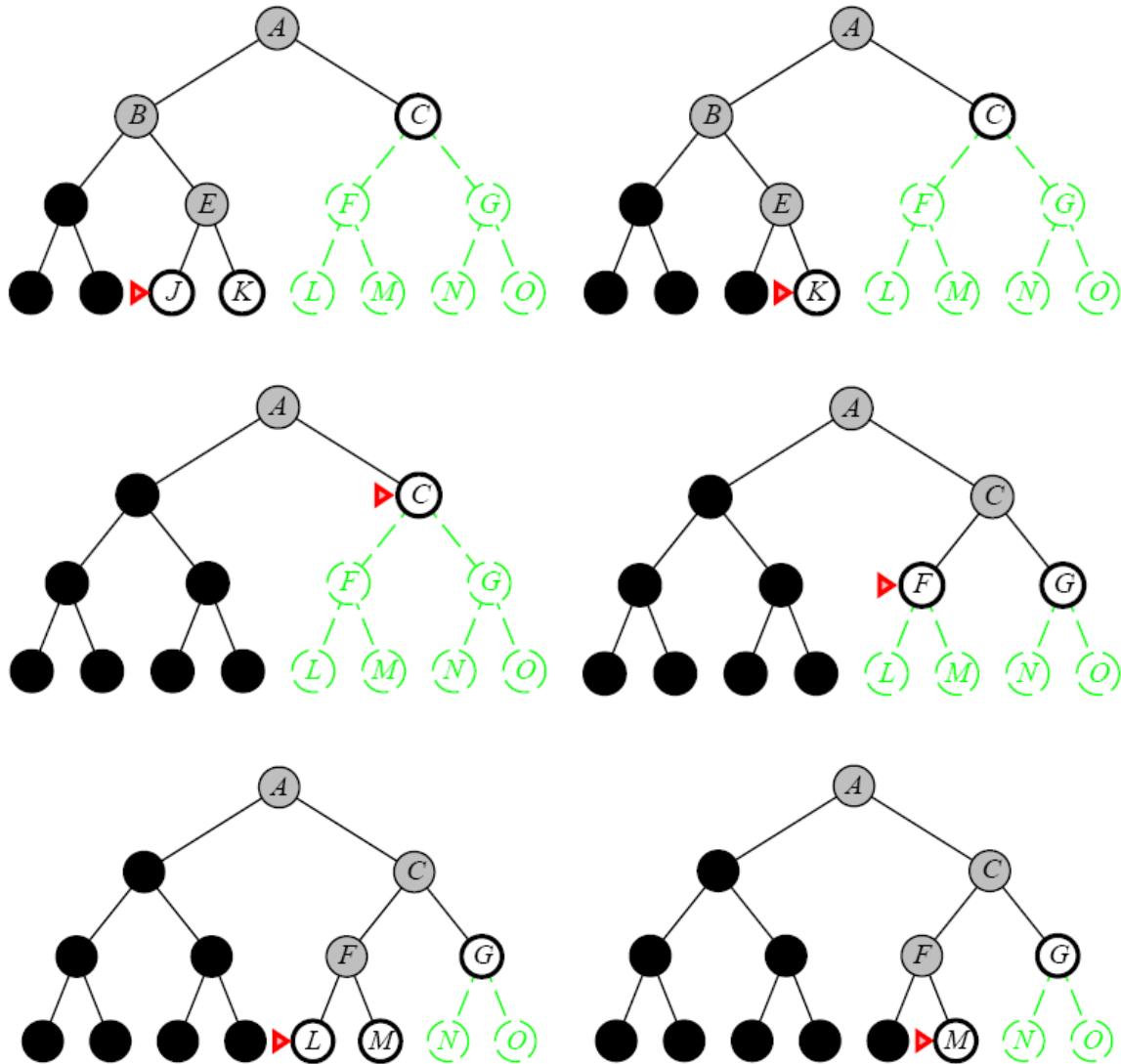
هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



مثال دیگر: آراد به بخارست (به صورت پشته)

- آراد را از پشته بردارید (pop)

- سیبیو، تیمیسوارا و زریند را به پشته اضافه نمایید (push)

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- سیبیو را از پشته بردارید و تست نمایید
- ارادیا ، فاگاراس و ریمنیکو ویلسیا را به پشته اضافه نمایید
- ارادیا را از پشته بردارید و تست نمایید
- فاگاراس را از پشته بردارید و تست نمایید
- بخارست را به پشته اضافه نمایید
- ...

خصوصیات جستجوی اول عمق

کامل بودن ؟؟ نه : در مواردی که عمق نامحدود است و در مورد حلقه ها درست کار نمی کند . بازنگری ، برای جلوگیری کردن از حالت های تکرار شده در طول مسیر منجر به کامل شدن در حالت های محدود می شود . به عبارت دیگر ، ما می توانیم در یک مسیر طولانی ، به صورت نامحدود ، سرگردان باشیم بدون این که به یک راه حل برسیم .

زمان ؟؟ $O(b^m)$ در صورتی که m به مراتب بزرگتر از d باشد ، بسیار بد است . ولی اگر راه حل ها پیچیده باشند ، این روش به مراتب سریع تر از جستجوی اول سطح می باشد . توجه کنید که m ، بیشینه عمق درخت می باشد . همچنین در برخی از موارد ، m ممکن است نامحدود باشد .

فضا ؟؟ $O(bm)$ ، که فضای خطی می باشد . ما فقط نیاز به نگهداشتن شاخه ای که به تازگی جستجو شده است داریم و این حسن بزرگ جستجوی اول عمق می باشد .

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بهینگی ?? نه . ما ممکن است یک راه حل را در عمق n ، تحت یک فرزند بدون دیدن یک راه حل کوتاه تر ، تحت فرزند دیگر پیدا نماییم . (در مثال قبل که با استفاده از پشته آن را حل کردیم ، اگر اول ریمنیکو ویلسیا را از پشته برمی داشتیم چه اتفاقی می افتاد؟)

اول ، یک سؤال

چرا ما به الگوریتم هایی که یک جستجوی جامع را انجام می دهنند توجه می کنیم ؟ آیا چیزی سریع تر وجود ندارد ؟ بسیاری از مسایلی که ما به آن ها توجه می کنیم ، NP - کامل می باشند و در مورد آن ها هیچ الگوریتم با زمان چندجمله ای شناخته شده ای وجود ندارد و بدتر از آن ، تعدادی هم غیرقابل تخمین می باشند .

جستجوی اول سطح

اگر جستجوی اول سطح را با استفاده از یک صفت پیاده سازی کنیم ، تمام گره های در سطح n ، قبل از هر گره در سطح $n+1$ ارزیابی می شوند و این دارای این حسن است که ، کامل می باشد و در زمانی که هزینه ای مرحله به صورت یکسان می باشد ، راه حل بهینه را پیدا خواهد نمود و دارای این ضعف است که به فضایی به صورت نمایی نیازمندیم .

جستجوی اول عمق

اگر جستجوی اول سطح را با استفاده از پشته پیاده سازی نماییم ؛ ابتدا یک گره توسعه داده می شود ، سپس فرزندانش توسعه داده می شوند و ... و در هر لحظه ، فقط یک شاخه از درخت جستجو در حافظه نگهداری می شود . حسن این روش این است که به فضایی به صورت خطی نیازمندیم و عیب آن این است که نا کامل و غیربهینه می باشد .

جلوگیری از جستجوی نامحدود

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چند روش برای جلوگیری از جستجوی اول عمق نامحدود وجود دارد، یکی استفاده از- closed-list است که ممکن است همیشه به ما کمک نکند. در این صورت ما باید تعداد زیادی گره را به صورت نمایی در حافظه نگهداری نماییم. دو روش دیگر جستجوی با عمق محدود شده و جستجوی عمیق کننده‌ی تکراری جستجوی اول عمق می‌باشد.

جستجوی با عمق محدود شده

با محدود نکردن عمق 1 با جستجوی اول عمق برابر است و گره‌های در عمق 1 هیچ جانشینی ندارند. در واقع، جستجوی با عمق محدود شده، همان جستجوی اول عمق است که در آن عمق جستجو محدود شده است (تا یک عمقی در درخت پایین می‌رویم و بیش تراز آن پایین نمی‌رویم) و عملیات جستجو در این عمق، متوقف می‌شود. در اینجا مشکل دیگری به وجود می‌آید و آن این است که اگر راه حل در عمقی بیش تراز 1 باشد، چطور؟ چگونه ما یک 1 خوب را به دست آوریم؟ در مسئله‌ی رومانی، ما می‌دانیم که بیست (۲۰) شهر وجود دارد، بنابراین $1=19$ ، یک انتخاب خوب می‌باشد. در مورد پازل ۸-تایی چه طور؟

شبه کد پیاده سازی روش جستجوی با عمق محدود شده با استفاده از پشته:

```

push(initialState)
do
    node = pop( )
    if goalTest(node)
        return node
    else
        if depth(node) < limit

```

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

children = successor-fn(node)

for child in children

push(child)

پیاده سازی بازگشتی :

تابع cutoff مقدار fail ، soln function Depth-Limited-Search(problem,limit) را
برمی گرداند

Recursive-DLS(Make-Node(Initial-State[problem]),problem,limit)

تابع Recursive-DLS(node,problem,limit) را fail ، soln function مقادیر
برمی گرداند

cutoff-occurred? \leftarrow false

در صورتی که Goal-Test(problem,State[node]) صحیح بود گره (node) را برگردان . (در
صورتی که مقدار تابع درست بود node را برگردان ;)

در غیر این صورت ، اگر Depth[node]=limit بود ، مقدار cutoff را برگردان .

در غیر این صورت ، برای هر successor (جانشین) در Expand(node,problem) کارهای زیر را
انجام بده :

result \leftarrow Recursive-DLS(successor,problem,limit)

اگر result=cutoff بود سپس مقدار true را در cutoff-occurred بروز .

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در غیر این صورت، اگر `result` را برگردان (`< >`) علامت نامساوی می باشد).

در صورتی که `cutoff-occurred` صحیح بود `cutoff` را برگردان و در غیر این صورت `failure` را برگردان.

پایان الگوریتم

جستجوی عمیق شونده ی تکراری

بر مبنای ایده ی جستجوی با عمق محدود شده، توسعه می باشد. جستجوی با عمق محدود شده را ابتدا با عمق $l=1$ ، سپس با عمق $l=2$... انجام دهید. سرانجام، $l=d$ می شود، این به این معنی می باشد که جستجوی عمیق کننده ی تکراری، کامل می باشد. اشکال این روش این است که برخی از گره ها تولید شده اند و چند بار، توسعه داده شده اند. در سطح یک؛ تعداد b^d گره، تعداد d مرتبه تولید می شوند. در سطح دو؛ تعداد b^2 گره، $d-1$ بار تولید می شوند و ... در نتیجه در سطح d ، تعداد b^d بار تولید می شوند. در این روش، مجموع زمان اجرا برابر $O(b^d)$ می باشد. که اندکی کم تر از تعداد گره های تولید شده در جستجوی اوّل سطح می باشد و هنوز به حافظه ای به صورت خطی نیازداریم. پس، عمق محدود شده را با افزایش محدودیت ها به کار می گیرد. الگوریتم این روش به صورت زیر است:

تابع `function Iterative-Deepening-Search(problem)` یک راه حل را بر می گرداند.

ورودی ها `problem` که یک مسأله است، می باشد

برای `depth` مساوی با صفر تا بی نهایت (∞) کارهای زیر را انجام بد

`result ← Depth-Limited-Search(problem, depth)`

در صورتی که `result` را برگردان `< >cutoff` بود

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پایان حلقه

پایان الگوریتم

شبه کدی به صورت دیگر برای جستجوی عمیق شوندهٔ تکراری:

$d = 0$

while (1)

result = depth-limited-search(d)

if result == goal

return result

else

$d = d + 1$

: مثال

جستجوی عمیق شوندهٔ تکراری با $l=0$

Limit = 0



جستجوی عمیق شوندهٔ تکراری با $l=1$

Limit = 1



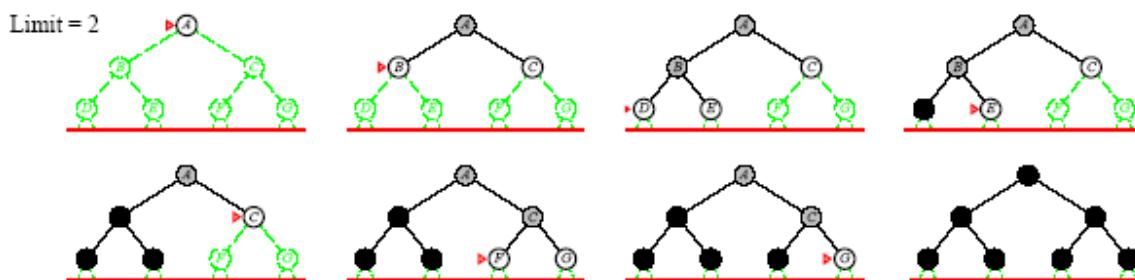
جستجوی عمیق شوندهٔ تکراری با $l=2$

مترجم: سهراب جلوه گر

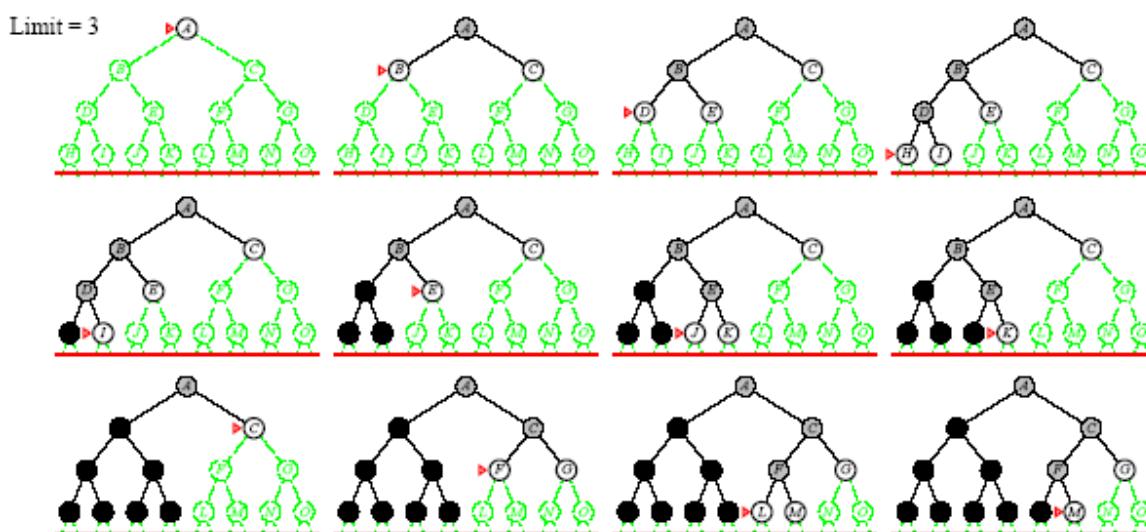
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



جستجوی عمیق شونده‌ی تکراری با $l=3$



جستجوی عمیق شونده‌ی تکراری، جستجوی اول سطح و اول عمق را با هم ترکیب می‌کند و لایه‌ها را شبیه جستجوی اول سطح به وجود می‌آورد، اما در هر اجرا یک اول عمق را انجام می‌دهد که باعث صرفه جویی در فضای شود. در این روش، وضعیت‌ها چند بار توسعه داده می‌شوند.

در واقع، جستجوی عمیق شونده‌ی تکراری، شبیه جستجوی اول سطح می‌باشد که در آن همه‌ی گره‌های در عمق n قبل از هر گره در عمق $n+1$ بررسی می‌شوند. نظریه جستجوی اول سطح، ما می‌توانیم بهینگی را در جهان‌های با هزینه‌ی غیریکسان با توسعه دادن با توجه به هزینه‌ی مسیر و ترجیحات با استفاده از

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عمق، به دست آوریم. این روش، جستجوی دراز کننده‌ی طول^۱ نام دارد. همه‌ی مسیرها را در هزینه‌ی کم تر از p جستجو می‌کند. p را با δ افزایش می‌دهد. در جهان‌های پیوسته، δ چه باید باشد؟

جريان معکوس^۲

در زمانی که جستجوی اوّل عمق و همتایانش به وضعیت عدم موقّیت می‌رسند، چه اتفاقی می‌افتد؟ به بالای گره‌ی پدر (والد) می‌روند و برادر بعدی را امتحان می‌کنند. و فرض را براي اين قرار می‌دهند که جدیدترین عملکرد انتخاب شده، عملکردی است که باعث عدم موقّیت شده است. این روش، جريان معکوس به ترتیب وقوع^۳ نام دارد و جدیدترین کاری را که انجام داده اید، بی اثر نمایید. مسأله‌ای که وجود دارد این است که عدم موقّیت، شاید یک نتیجه از یک تصمیم گیری قبلی باشد؛ مثال، ۴- وزیر. محدودیت‌ها می‌توانند به شما برای محدود کردن اندازه‌ی فضای جستجو کمک نمایند. جريان معکوس هوشمند^۴ برای تحلیل دلیلی برای عدم موقّیت و غیرفعال کردن جستجو برای آن نقطه تلاش می‌نماید و می‌تواند جدیدترین متغیر ناسازگار را غیرفعال نماید (جريان معکوس)، همچنین می‌تواند بررسی مستقیم^۵ را انجام دهد؛ آیا یک انتساب ممکن مقدارها برای متغیرهایی در این نقطه وجود دارد؟

خصوصیات جستجوی عمیق شونده‌ی تکراری

کامل بودن \Leftrightarrow بله

$$(d+1)b^0 + db^1 + (d-1)b^2 + \dots + b^d = O(b^d)$$

iterative lengthening search^۱

backtracking^۲

chronological backtracking^۳

intelligent backtracking^۴

forward checking^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فضا ?? O(bd)

بهینگی؟ بله اگر گام cost برابر یک باشد . می تواند برای به وجود آوردن درخت uniform-
بازنگری شود . به عنوان مثال ، مقایسه ای عددی برای $b=5$ و $d=10$ ، برای دورترین برگ سمت
راست :

$$N(IDS) = 50 + 400 + 3000 + 20000 + 100000 = 123450$$

$$N(BFS) = 10 + 100 + 1000 + 10000 + 100000 + 999990 = 1111100$$

IDS (جستجوی عمیق شونده ای تکراری) بهتر عمل می کند ، چون که گره های دیگر درون عمق d توسعه داده نمی شوند . BFS (جستجوی اوّل سطح) می تواند برای مدت زمانی که یک گره تولید می شود بازنگری شود.

جستجوی دو طرفه ^۱

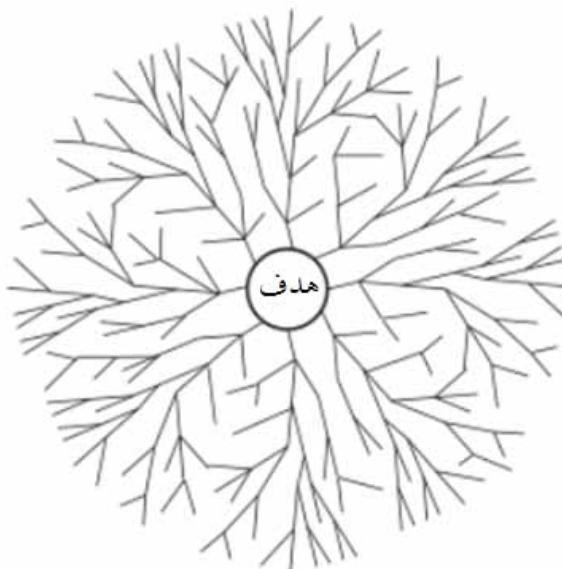
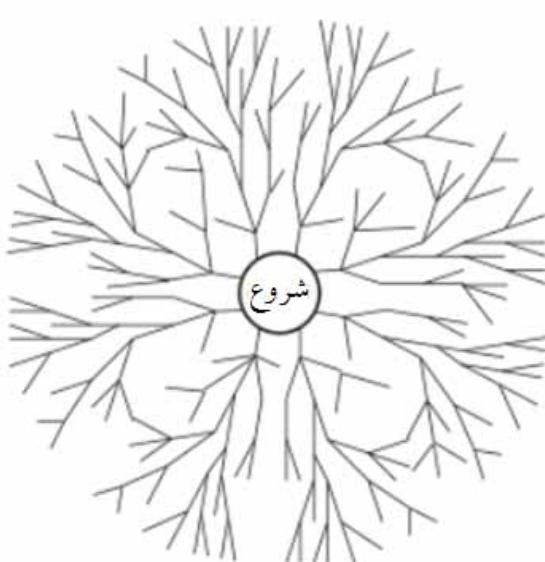
دو جستجوی شبیه را اجرا می نماید ، یکی در جهت وضعیت اوّلیه و دیگری در خلاف جهت و از طرف وضعیت هدف و در زمانی که دو جستجو به هم می رستند جستجو خاتمه می یابد .

bidirectional search ^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



خصوصیات جستجوی دو طرفه

- در صورتی که هر دو جستجو اوّل سطح باشند کامل است.

- زمان: $O(b^{d/2})$

- فضا: $O(b^{d/2})$

- در صورتی که هر دو روش اوّل سطح باشند بھینه می باشد

اشکالات^۱ این روش: تولید قبلی ها (معکوس کردن عملگرها) و هماهنگی میان دو جستجو و این که جستجو باید گره ها در حافظه نگهداری نماید (برای بررسی این که آیا گره ها قبلاً ملاقات شده اند یا نه)

drawbacks^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

خلاصه ای الگوریتم ها

عمق شونده ای تکراری	عمق محدود شده	اول عمق	با هزینه ای یکسان	اول سطح	مقیاس ^۱
بله	بله ، در صورتی که $ l \leq d$ باشد	نه	* بله	* بله	کامل بودن ??
b^d	b^l	b^m	$b^{\lceil C^*/\epsilon \rceil}$	$b^d + 1$	زمان ??
bd	bl	bm	$b^{\lceil C^*/\epsilon \rceil}$	$b^d + 1$	فضا ??
* بله	نه	نه	بله	* بله	بهینگی ??

در جدول بالا ، b : فاکتور انشعاب ، d : عمق کم ترین راه حل ، m : بیشینه ای عمق درخت جستجو و l : محدودیت عمق می باشند .

حالات (وضعیت) های تکرار شده - عدم موّقیت در پیدا کردن حالت های تکرار شده می تواند یک مسئله ای خطی را تبدیل به یک مسئله ای نمایی کند .

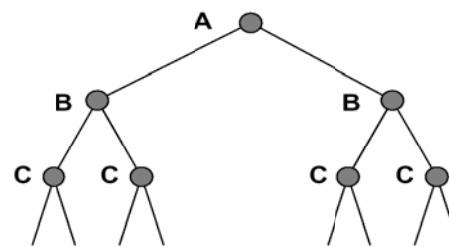
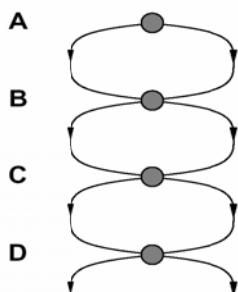
Criterion ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



راه حل این مشکل ، نگهداری تاریخچه است ؛ گره های ملاقات شده (توسعه داده شده) در closed list نگهداری می شوند و گره جاری اگر به تازگی در closed list بوده ، حذف می شود . مشکل این روش این است که بهینگی ممکن است از بین برود چون همه ی گره ها در حافظه نگهداری می شوند .

الگوریتم جستجوی گراف

تابع (function Graoh-Search(problem,fringe) یک راه حل یا عدم موقّیت را بر می گرداند .

یک مجموعه خالی را به closed list تخصیص بده .

$\text{fringe} \leftarrow \text{Insert}(\text{Make-Node}(\text{Initial-State}[problem]),\text{fringe})$

در حلقه کارهای زیر را انجام بده

در صورتی که fringe خالی باشد عدم موقّیت را بر گردان

$\text{node} \leftarrow \text{Remove-Front}(\text{fringe})$

در صورتی که Goal-Test(problem,State[node]) آمیز بود ، گره (node) را بر گردان

در صورتی که State[node] در closed list نبود

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

را به closed اضافه نما State[node]

بریز fringe را در InsertAll(Expand(node,problem),fringe)

پایان الگوریتم

خلاصه - فرمول بندی مسئله ، معمولاً به در کنار هم قرار دادن جزئیات دنیای واقعی برای تعریف یک فضای حالت که بتواند به طور عملی به وجود آید نیاز دارد . تنوع روش ها یا استراتژی های جستجوی ناآگاهانه : جستجوهای عمیق شونده ی تکراری^۱ فقط به صورت فضای خطی استفاده می شوند و زمان به مراتب بیش تری نسبت به سایر الگوریتم های ناآگاهانه ندارند . جستجوی گرافی می تواند به مراتب موثرتر از جستجوی درختی باشد . فرمول بندی یک مسئله ی جستجو با استفاده از وضعیت اوّلیه ، آزمون هدف ، عملیاتی که باید انجام شوند ، تابع جانشین (مولد) و هزینه ی مسیر ، منجر به جستجو در یک فضای حالت با استفاده از یک درخت جستجو می شود . الگوریتم ها عبارتند از : جستجوی اوّل سطح ، جستجوی اوّل عمق ، جستجوی با هزینه ی یکسان ، جستجوی با عمق محدود شده و جستجوی عمیق کننده ی تکراری

iterative deeping search^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی

فصل چهارم

جستجوی آکاها نه

(مکافتفه ای)

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

خلاصه‌ی ریوس مطالب

- جستجوی اول - بهترین^۱
- جستجوی^{*} A*
- ابتکارات^۲ (اکتشافات)

آگاهانه کردن جستجو

الگوریتم های قبلی، می توانستند راه حل ها را پیدا کنند، اما ناکارامد بودند؛ گره ها به صورت نمایی به وجود می آمدند. با استفاده از آگاهی هایی که در مورد ساختار مسئله داریم می توانیم کارایی را بهتر کنیم. البته باید توجه کنیم که این دانش را باید از جایی دیگر بگیریم و این دانش باید درست باشد.

چرا از دانش در یک مسئله استفاده می نماییم؟

Best-first^۱

Heuristics^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

روش های ناآگاهانه فقط از اطلاعات قابل دسترس از تعریف مسئله استفاده می نمایند. بدترین حالت پیچیدگی زمانی وقتی است که پیچیدگی زمانی به صورت نمایی باشد؛ در اکثر موارد روش های ناآگاهانه ناکارآمد هستند. ممکن در صورتی که چیزهایی در مورد مسئله بدانیم بهتر عمل کنیم. ایده اصلی این است که برای به وجود آوردن گره ها از یک تابع ارزیابی^۱ استفاده نماییم.

تابع ارزیابی

یک تابع ارزیابی $f(n)$ ، شایستگی یک وضعیت هدف را تخمین می زند؛ یک وضعیت با هر گره درخت جستجو پیوند برقرار می کند و بنابراین، تابع می تواند برای گره های درخت جستجو استفاده شود. گره ای که دارای کم ترین مقدار ارزیابی است در میان گره های کاندید انتخاب می شود؛ توجه کنید که ما به دنبال ارزان ترین راه حل هستیم.

در پیاده سازی این روش، از fringe که یک صف مرتب شده توسط تابع ارزیابی است استفاده می نماییم. جستجوی با هزینه‌ی یکسان را به یاد بیاورید در این روش گره ها براساس مجموع هزینه‌ی مسیر، توسعه داده می شوند و از یک صف اولویّت، برای پیاده سازی جستجوی با هزینه‌ی یکسان استفاده می شود؛ هزینه‌ی مسیر، مثالی از یک تابع ارزیابی می باشد.

روش جستجوی اوّل - بهترین^۲

اوّل، بهترین گره را برای رسیدن به هدف توسعه دهید. در ادامه نمی دانیم که این گره هنوز بهترین است یا نه؛ اگر ما این مطلب را بدانیم، آن گاه ما نیازی به عمل جستجو نداریم (در صورتی که شما راه خود را بدانید شما بر روی یک طرح کار نخواهید کرد) در ادامه گره ای را که به نظر می رسد بهترین باشد را توسعه دهید.

¹ evaluation function
² best-first search

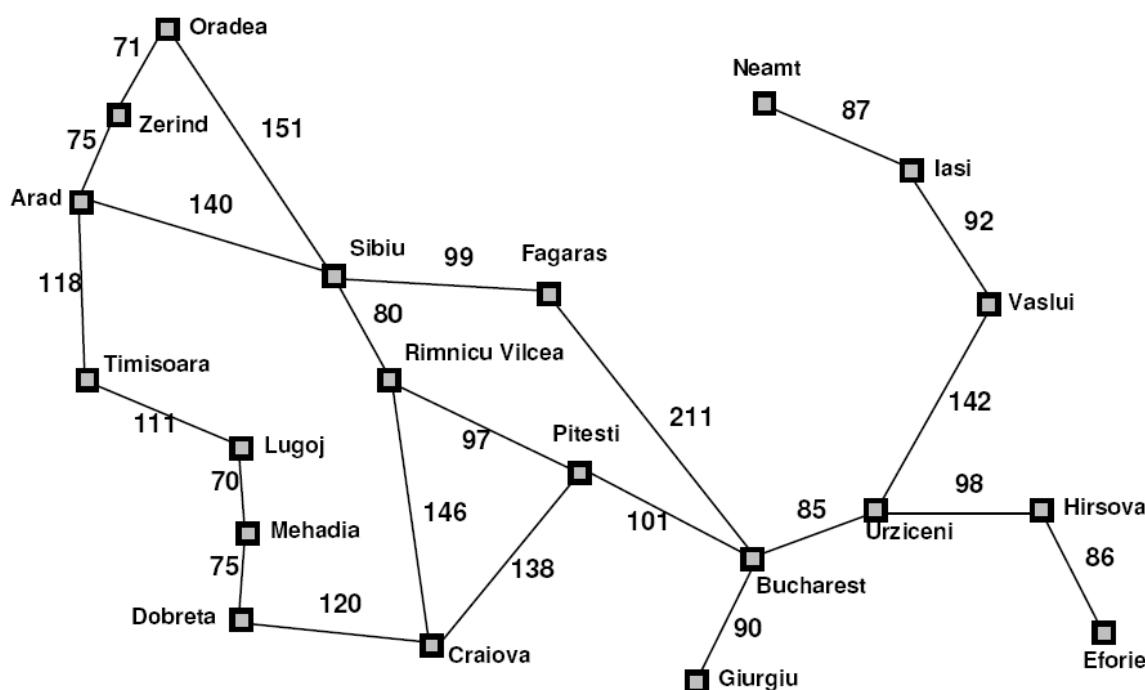
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پس از ایده‌ی جستجوی اول - بهترین استفاده از یک تابع ارزیابی می‌باشد.

مسئله‌ی کشور رومانی با ارزیابی مراحل به صورت کیلومتر



آراد	۳۶۶	فاگاراس	۱۷۸	مهدیا ^۱	۲۴۱	سیبیو	۲۵۳
------	-----	---------	-----	--------------------	-----	-------	-----

Mehadia^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بخارست	۰	گیور گیو ^۳	۷۷	نمت ^۲	۲۳۴	تیمیسوارا ^۱	۳۲۹
کرایوا ^۷	۱۶۰	هیرسوا ^۶	۱۵۱	ارادیا ^۵	۳۸۰	اردزیسنی ^۴	۸۰
دوبرتا ^{۱۰}	۲۴۲	یاسی	۲۲۶	پیستی ^۹	۹۸	والسلوی ^۸	۱۹۹
افری ^{۱۲}	۱۶۱	لو گوج ^{۱۱}	۲۴۴	ریمنیکو ویلسیا	۱۹۳	زریند	۳۷۴

توجّه کنید که اکتشاف (که به صورت جدولی در بالا داده شده است) جزیی از تعریف اولیه‌ی مسئله نمی‌باشد و در این مورد به صورت یک جدول خارجی داده شده است.

Timisoara^۱

Neamt^۲

Giurgiu^۳

Urziceni^۴

Oradea^۵

Hirsova^۶

Craiova^۷

Vaslui^۸

Pitesti^۹

Dobreta^{۱۰}

Lugoj^{۱۱}

Eforie^{۱۲}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



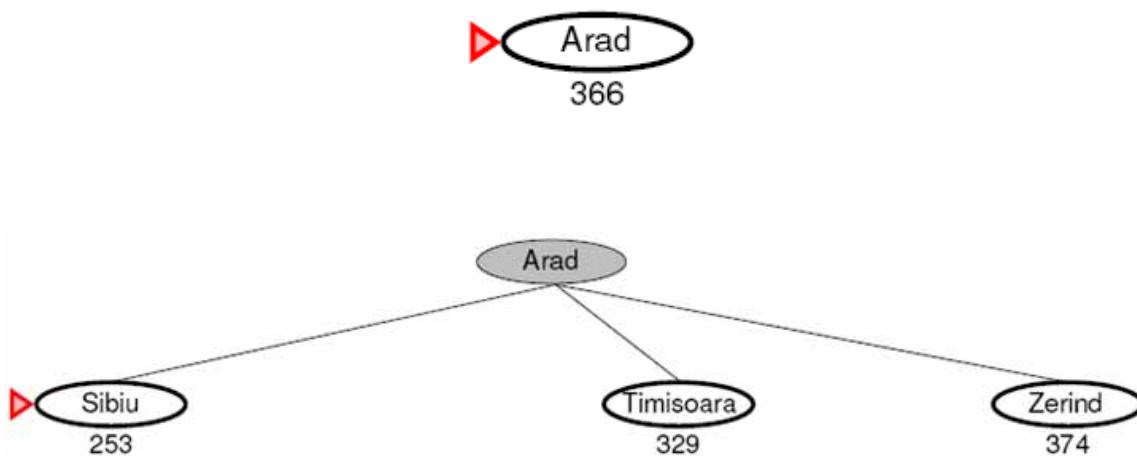
هوش مصنوعی

مسافت خطی مستقیم به شهر بخارست

جستجوی حریصانه^۱

در این روش ، در ارزیابی ، فقط از مکاشفه استفاده می شود ($f(n)=h(n)$) و الگوریتم ، گرهی را که به نظر می رسد به هدف نزدیک تر است را توسعه می دهد . وتابع ارزیابی ($h(n)$) (ابتکاری) برابر است با تخمین ارزش از n تا نزدیک ترین هدف و $h_{SLD}(n)$ برابر است با فاصله ای خطی مستقیم از n تا شهر بخارست .

مثالی از جستجوی Greedy



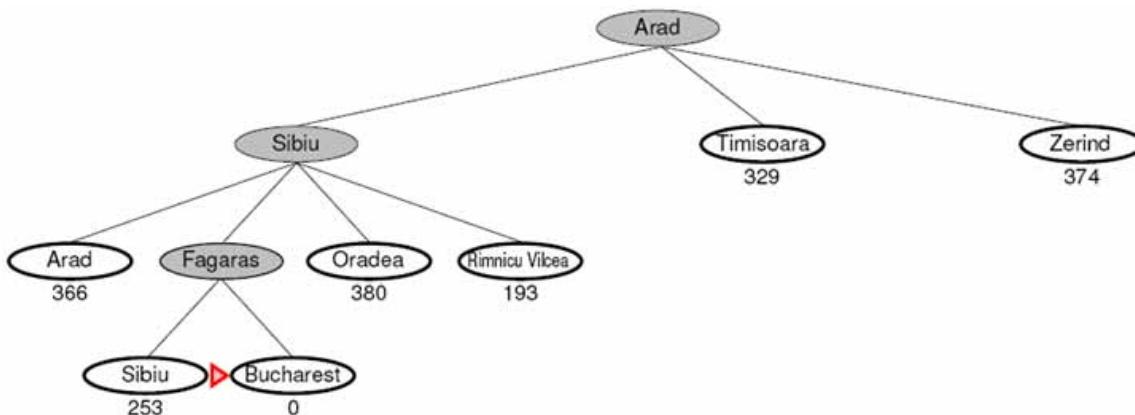
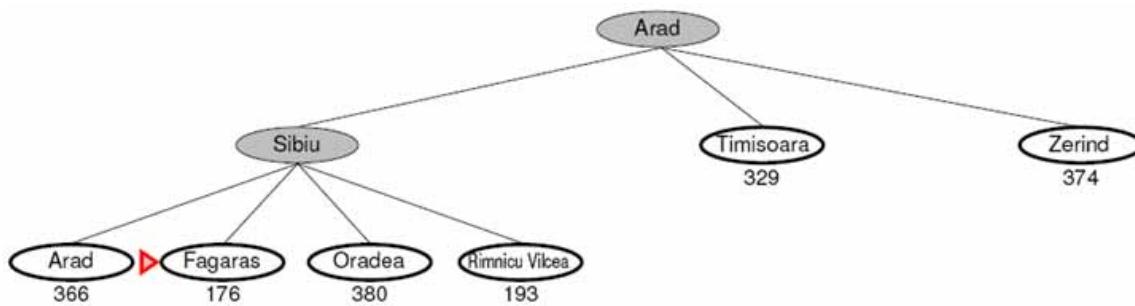
Greedy search^۱

مترجم: سهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی



خصوصیات جستجوی حریصانه

کامل بودن؟؟ نه ممکن است در حلقه ها گیر کند ، مثلا ، اگر مقصد ما شهر Oradea باشد ، داریم

:

lasi → Neamt → lasi → Neamt →

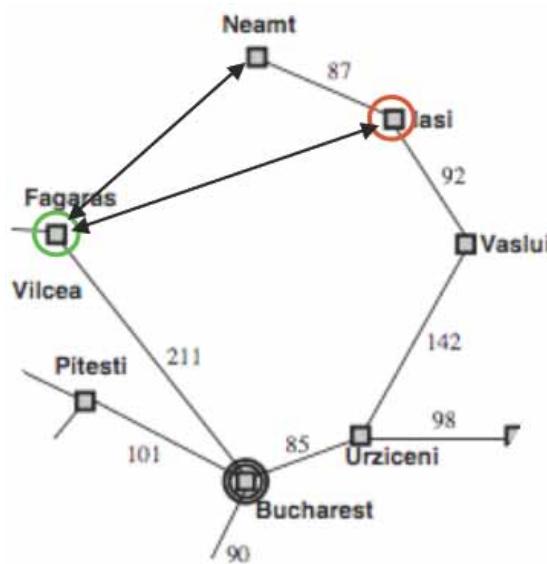
به عنوان مثالی دیگر ، اگر بخواهیم از یاسی^۱ به فاگاراس برویم ؛ میان یاسی و نمت^۱ حلقه به وجود می آید که برای رفع این مشکل باید از وضعیت های ملاقات شده خودداری نماییم .

Iasi^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



این روش در فضاهای جستجوی محدود با تست وضعیت تکرار شده، **کامل** است.

زمان : $O(b^m)$ ، اما با یک ابتکار صحیح می‌توان بهبودی قابل توجهی به دست آورد.

فضا : $O(b^m)$ ، همه ی گره‌ها در حافظه نگه می‌دارد. که b ، فاکتور انشعاب و m عمق

درخت جستجو می‌باشد.

بهینگی ؟؟ نه، یک مسیر را به سوی هدف دنبال می‌نماید؛ شبیه جستجوی اوّل - عمق می‌باشد.

A* جستجوی

مسئله‌ای که در مورد روش‌های جستجوی اوّل - بهترین و حریصانه وجود دارد، این است که هزینه‌ای که مسیر تا کنون داشته است مورد توجه قرار نگرفته است. روش جستجوی A^* از به وجود آوردن (توسعه دادن) مسیرهایی که پر هزینه هستند اجتناب می‌کند.

Neamt¹

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



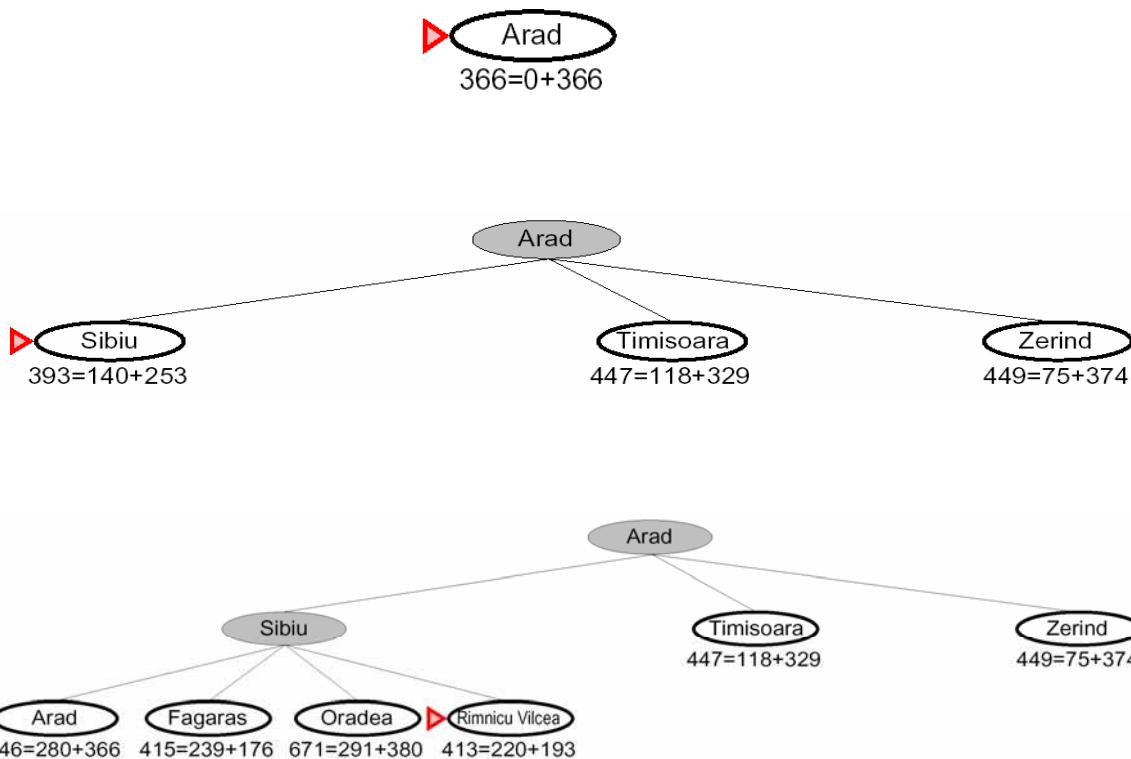
هوش مصنوعی

در این روش ، تابع ارزیابی $f(n) = g(n) + h(n)$ می باشد . که $g(n) =$ هزینه ای که تا کنون برای رسیدن به n صرف شده است . $h(n) =$ هزینه ای تخمین زده شده برای n و $f(n) =$ کل هزینه ای تخمین زده شده ای مسیر از n تا هدف .

روش جستجوی A^* از یک روش ابتکاری مجاز برای جستجو استفاده می کند . در واقع ، جستجوی A^* ترکیب جستجوی با هزینه ی یکسان و جستجوی حریصانه می باشد .

* قضیه : جستجوی A^* ، جستجویی بهینه یا مطلوب است .

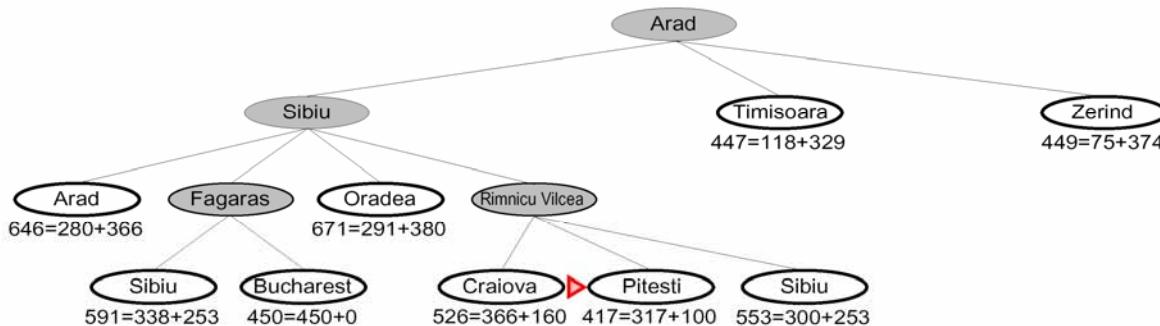
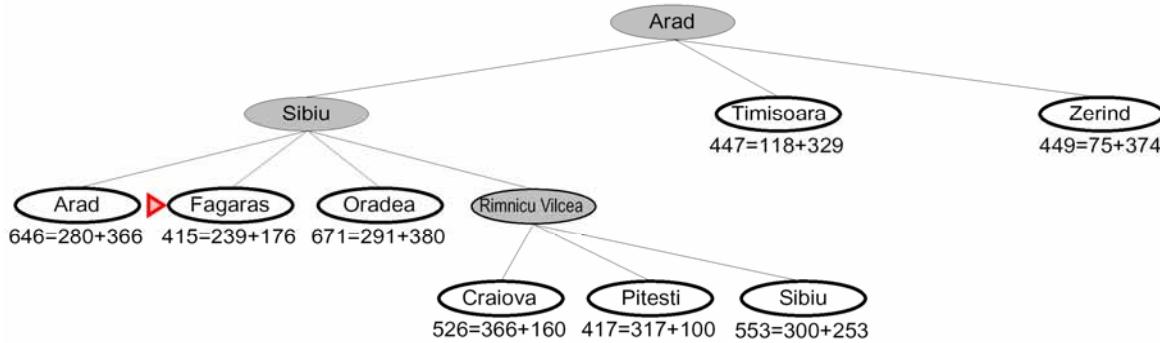
مثالی برای جستجوی A^*



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

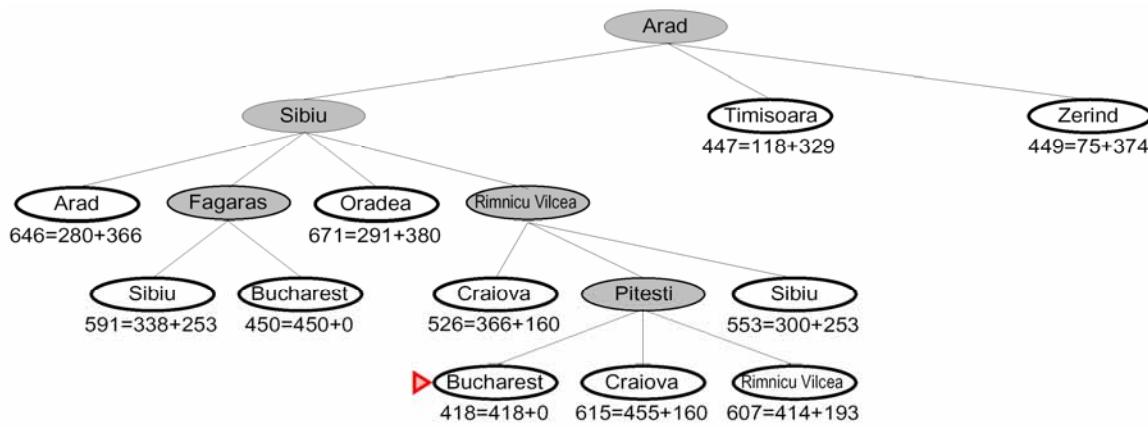


مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



بهینگی جستجوی A^*

برای تضمین بهینگی مکاشفه نباید هزینه‌ی رسیدن به هدف را زیاد برآورد نماییم در مکاشفه‌ی قابل قبول داریم؛ $h^*(n) \leq h^*(n)$ که $h^*(n)$ هزینه‌ی واقعی رسیدن به هدف می‌باشد. در مسیریابی، فاصله‌ی خطی مستقیم قابل قبول می‌باشد. این روش قابلیت قبول بهینگی را فقط برای الگوریتم جستجوی کلی تضمین می‌کند.

الگوریتم

تابع $\text{General-Search}(\text{problem}, \text{Queuing-Fn})$ یک راه حل یا عدم موفقیت را برمی‌گرداند

$\text{fringe} \leftarrow \text{make-queue}(\text{make-node}(\text{initial-state}[\text{problem}]))$

در حلقه کارهای زیر را انجام بد

در صورتی که fringe خالی می‌باشد عدم موفقیت را برگردان

$\text{node} \leftarrow \text{Remove-Front}(\text{fringe})$

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در صورتی که کار گرفته شده برای وضعیت (گره) موفق شد، گره را برگردان

fringe \leftarrow Queuing-Fn(fringe, Expand(node, Operators[problem]))

تابع Graph-Search(problem, Queuing-Fn) یک راه حل یا شکست را برمی گرداند

closed $\leftarrow \Phi$

fringe \leftarrow make-queue(make-node(initial-state[problem]))

در حلقه کارهای زیر را انجام بده

در صورتی که fringe خالی است عدم موفقیت را برگردان

node \leftarrow Remove-Front(fringe)

در صورتی که Goal-Test[problem] به کار رفته برای State(node) موفقیت آمیز بود node را برگردان

در صورتی که $node \notin closed$ آن گاه

closed را به node اضافه نما

\leftarrow Queuing-Fn(fringe, Expand(node, Operators[problem]))

fringe

پایان الگوریتم

سازگاری (یکنواحی) اکتشافات

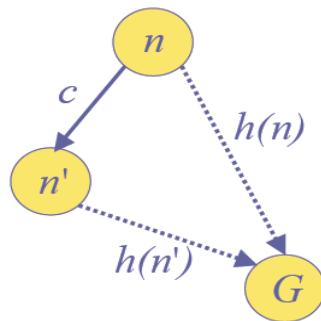
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در این مورد ، مطمین شوید که مسیر بهینه‌ی وضعیت‌های تکرار شده اولین باری است که توسعه داده می‌شود (به وجود می‌آید) . این کار می‌تواند با انتخاب اکتشاف صحیح انجام شود .

با توجه به نامساوی مثلثی داریم :



$$f(n') = g(n') + h(n') = g(n) + c + h(n') \geq g(n) + h(n) = f(n)$$

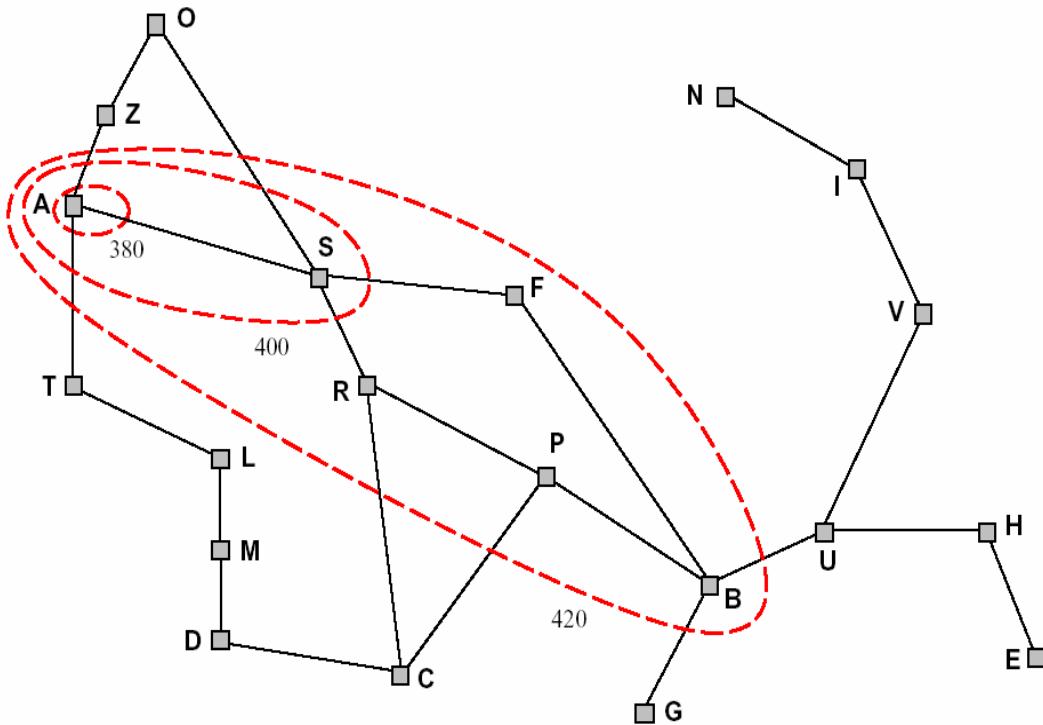
بهینه سازی A^* (مفیدتر)

روش A^* ، گره‌ها را براساس ترتیب افزایش مقدار f به وجود می‌آورد .

مترجم: سه راب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



خصوصیات A^*

کامل بودن ؟؟ بله ، مگر این که تعداد نامحدودی گره با $f(n) \leq f(G)$ وجود داشته باشد .

زمان ؟؟ به صورت نمایی می باشد .

فضا ؟؟ تمام گره ها در حافظه نگه می دارد و به صورت نمایی می باشد .

بهینه ؟؟ بله - تا زمانی که f_i تمام نشده است نمی تواند f_{i+1} را توسعه بدهد .

اگر C^* هزینه ی مسیر راه حل بهینه باشد ؟

• تمام گره های با $f(n) < C^*$ را توسعه می دهد .

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- بعضی از گره هایی که $f(n) = C^*$ است را توسعه می دهد.

- گره هایی که $C^* < f(n)$ است را توسعه نمی دهد.

روش A^* به طور موثری برای هرتابع ارزیابی بھینه می باشد. یک الگوریتم ممکن است که راه حل بھینه را در صورتی که همه ی گره های با $C^* < f(n)$ را توسعه ندهد، گم کند.

A^* بهتر کردن

یکی از الگوریتم های کلیدی در هوش مصنوعی می باشد، اما دارای اشکال در حافظه موردنیاز می باشد چون که تمام گره های ملاقات شده را در حافظه نگه می دارد و برای مسایل با ساختار بزرگ، عملی نمی باشد؛ برای رفع این مشکل از روش عمیق شونده تکراری استفاده می کنیم.

روش A^* عمیق شونده تکراری

در این روش، به جای محدود کردن عمق، به صورت تدریجی محدودیت تابع ارزیابی افزایش داده می شود و جستجوی اوّل عمق با هزینه محدود شده را برای تابع ارزیابی انجام می دهد. مشکل این روش، مقدار افزایش تدریجی محدوده ای ارزیابی است. برای رفع این مشکل یک راه این است که مقدار را در محدوده ای مقدار مرحله ای قبلی در نظر بگیریم و روش دیگر این است که محدوده را به وسیله ی یک مقدار ثابت به نام ^۱ افزایش دهیم.

اکتشافات (ابتكارات)

پیدا کردن یک تابع مکاشفه ای خوب برای جستجو کاری بسیار مهم است. تا کنون ما از فاصله ای اقلیدسی استفاده می کردیم که برای مسایل مسیریابی، خوب بود. اما مکاشفات را برای مسایل جدید چگونه

^۱ برگرفته از مطالب استاد دانشگاه ایالت پیتسبرگ کشور ایالات متحده ای آمریکا Milos Hauskrecht

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پیدا کنیم؟ جواب این است که راه استانداردی وجود ندارد، در بیش تر موارد باید مکافات به صورت دستی پیدا شوند.

ابتکارات قابل قبول^۱ - برای پازل ۸-تایی: $h_1(n)$ = تعداد کاشی های در جای نادرست گذاشته شده. $h_2(n)$ = مجموع فاصله تا جزیره‌ی مانهاتان^۲ (به تعداد کاشی (خانه) های در جای درست قرار داده شده توجه نمایید)

7	2	4
5		6
8	3	1

حالت اولیه

1	2	3
4	5	6
7	8	

حالت نهایی

$$h_1(S) = ??6$$

$$h_2(S) = ??4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14$$

خلاصه - توابع مکافه‌ای، هزینه‌ی کوتاه ترین مسیر را تخمین می‌زنند. ابتکارات خوب می‌توانند به طور چشمگیری هزینه‌ی جستجو را کاهش دهند. جستجوی اوّل بهترین و حریصانه‌ی اوّل - بهترین پایین ترین h را توسعه می‌دهد و کامل نیست و همیشه هم بهینه نیست. جستجوی A^* پایین ترین $h+g$ را

admissible heuristics^۱

Manhattan^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

توسعه می دهد و کامل و بهینه می باشد و همچنین بهینگی موثری دارد (تا حد شکستن گره ها و برای جستجوی مستقیم) و ابتکارات قابل قبول می توانند از راه حل مستقیم مسایل مجاز مشتق شوند.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل پنجم

الگوریتم های

جستجوی محلی^۱

local search algorithms^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

رؤوس مطالب

- تپه نورده (بالا رونده از تپه)^۱
- گرم و سرد کردن شبیه سازی شده^۲
- الگوریتم های ژنتیکی، پیدایشی یا تکوینی^۳ (به طور خلاصه^۴)
- جستجوی محلی در فضاهای پیوسته (خیلی به طور خلاصه)

مسایل بحینه سازی

Hill-climbing^۱
Simulated annealing^۲
Genetic algorithms(GA)^۳
briefly^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



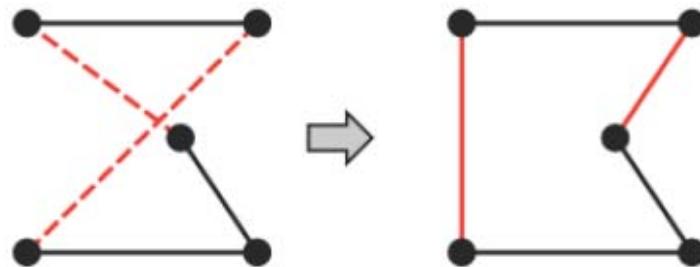
هوش مصنوعی

در برخی از ساختارهای ترکیبی که ما در تلاش برای بهینه سازی آن‌ها هستیم، محدودیت‌ها باید در نظر گرفته شوند و تابع هزینه باید بهینه شود. اغلب پیدا کردن راه حل آسان است، اما پیدا کردن بهترین راه حل سخت می‌باشد و پیدا کردن همه راه حل‌های ممکن هم غیر ممکن می‌باشد و ما دست کم، یک راه حل خوب را می‌خواهیم.

الگوریتم‌های بهبود یافته‌ی تکراری^۱ - در تعدادی از مسایل بهینه سازی، مسیر، نامربوط است؛ وضعیت (حالت) هدف، خودش راه حل است. بنابراین، برای فضای حالت، مجموعه‌ای از پیکربندی‌های کامل؛ پیکربندی بهینه، یا ارضای محدودیت‌ها، پیکربندی و برنامه‌ی زمانی را پیدا کنید. در موارد مشابه، شما می‌توانید از الگوریتم‌های بهبود یافته‌ی تکراری استفاده نمایید؛ یک حالت جاری تک را نگهداری نمایید و برای بهبود آن تلاش نمایید. در فضای ثابت، جستجوی برخط به اندازه‌ی جستجوی بروون خط مناسب است.

مثال: مسئله‌ی فروشنده‌ی دوره گرد - با هر مسیر کامل شروع کنید و معاوضه‌های دو

به دو را انجام دهید.



مثال: وزیر های شطرنج n - قایی^۲، وزیر را در یک صفحه‌ی شطرنج $n \times n$ بدون این که هیچ یک از دو وزیر در سطر^۱، ستون^۳ همانند قرار گیرند یا به طور مورب^۴، همانند قرار گیرند

iterative improvement algorithms^۱

n-queens^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

، قرار دهید . برای کاهش تعداد ناسازگاری ها ، یک وزیر را حرکت دهید . تقریبا همیشه مسایل وزیرهای چند گانه تقریبا به سرعت برای هر n بزرگ حل می شوند ، مثلا برای $n = 10$ میلیون

جستجو در مسئله ی بهینه

دو نوع روش وجود دارد : یکی روش سازنده^۴ که از یک وضعیت ابتدایی شروع کنید و به طرف یک راه حل حرکت نمایید ؛ به عنوان مثال ، در مورد مسئله ی مسافت شخص دوره گرد ، از یک شهر شروع نمایید و یک مسیر را با ملاقات همه ی شهرها به دست آورید . و دیگری روش تکراری که در این روش ، با یک راه حل شروع نمایید که می تواند غلط یا غیربهینه باشد و آن را به طرف یک راه حل بهینه ببرید ؛ به عنوان مثال ، در مورد مسافت شخص دوره گرد ، با یک مسیر شروع کنید و هزینه ی آن را با تعویض شهرها بهبود دهید .

از روش سازنده استفاده نمی کنیم ، چون که هزینه فقط به راه حل بستگی دارد ، نه به چگونگی آن ؛ روش جستجوی آگاهانه (مثل A^*) وقتی که ما یک راه حل برای همه ی مسیرها داریم خوب کار می کند . از جستجوی نآگاهانه هم استفاده نمی کنیم (چون فضای حالت در آن خیلی بزرگ می باشد)

جستجوی محلی

الگوریتم هایی که تا حالا ما بررسی کرده ایم ، تمام مسیر را از وضعیت اویلیه به وضعیت نهایی نگهداری می کنند و این باعث مصرف حافظه ی زیاد یا فضای زیاد در مورد مسایل بزرگ می شود . برای

row^۱

column^۲

diagonal^۳

constructive^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برخی از مسایل ، اطلاعات مسیر ، ضروری (حیاتی) می باشد ؛ مثل ، مسیر یابی و پازل ۸-تایی . در این مسایل ما هدف را می دانیم ، اما این که چگونه به آن برسیم را نمی دانیم . برای دسته ای دیگر از مسایل ، ممکن است ما نگران چگونگی رشته ای عملکردها نباشیم و پیدا کردن راه حل بهینه ، چیزی است که مهم می باشد مثل : زمانبندی ، طرح VLSI و پنهان شناسی (رمزنویسی)^۱ . در این موارد ، ما می توانیم با اطمینان ، برخی از اطلاعات مسیر را حذف نماییم . یک الگوریتم جستجو که فقط از وضعیت جاری استفاده می کند ، یک الگوریتم جستجوی محلی نام دارد . مزایای این روش عبارتند از :

به حافظه ای ثابت ، نیازمندیم و می توانیم راه حل هایی باورنکردنی را در مورد فضاهای بزرگ پیدا نماییم . و معایب این روش عبارتند از : معمولاً ، بهینه نمی باشد – فقط بهینه های محلی ، پیدا خواهند شد و می تواند به علت کمبود حافظه ، مردد باشد . در این روش ، فضای حالت شبیه روی زمین دارای تپه ها و دره ها می باشد و ارزیابی توسط تابع هزینه (ارزیابی) داده شده است .



cryptography^۱

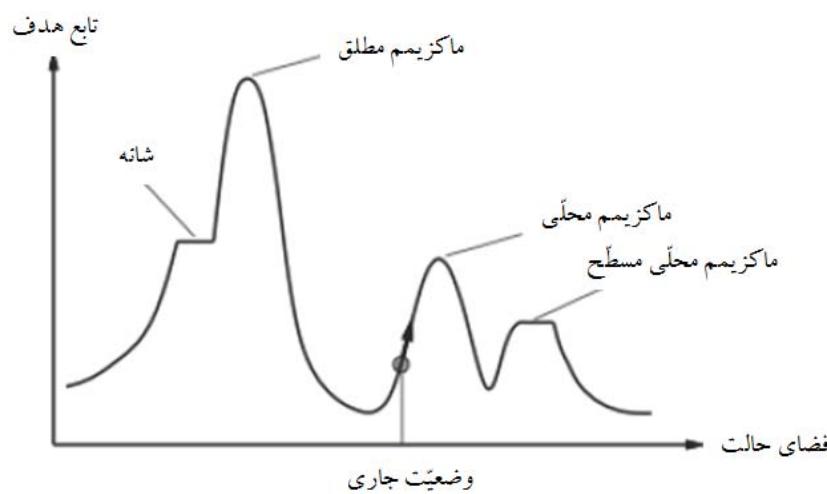
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

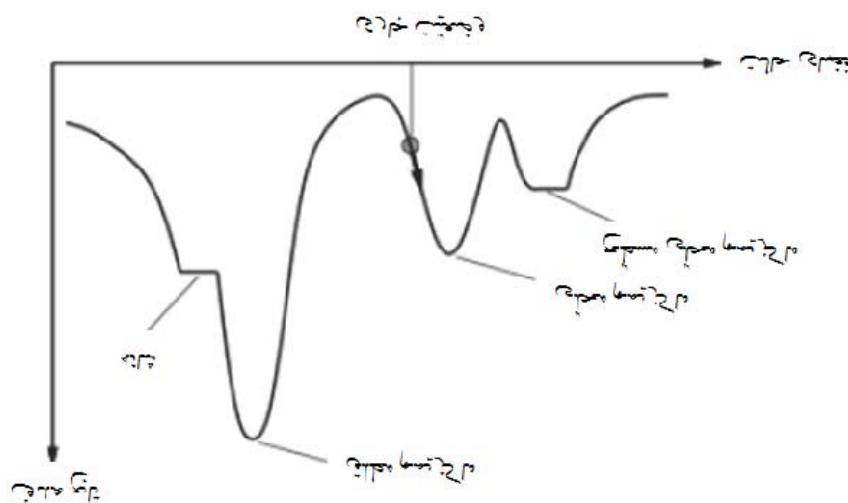


هوش مصنوعی

فضای حالت یک بعدی^۱ برای مثال ها، آسان تر نمایش داده می شود.



در این مورد، ماکزیمم و مینیمم قابل تعویض اند، برای این کار فقط مقدار را با استفاده از منفی کردن معکوس نمایید.



one-dimensional^۱

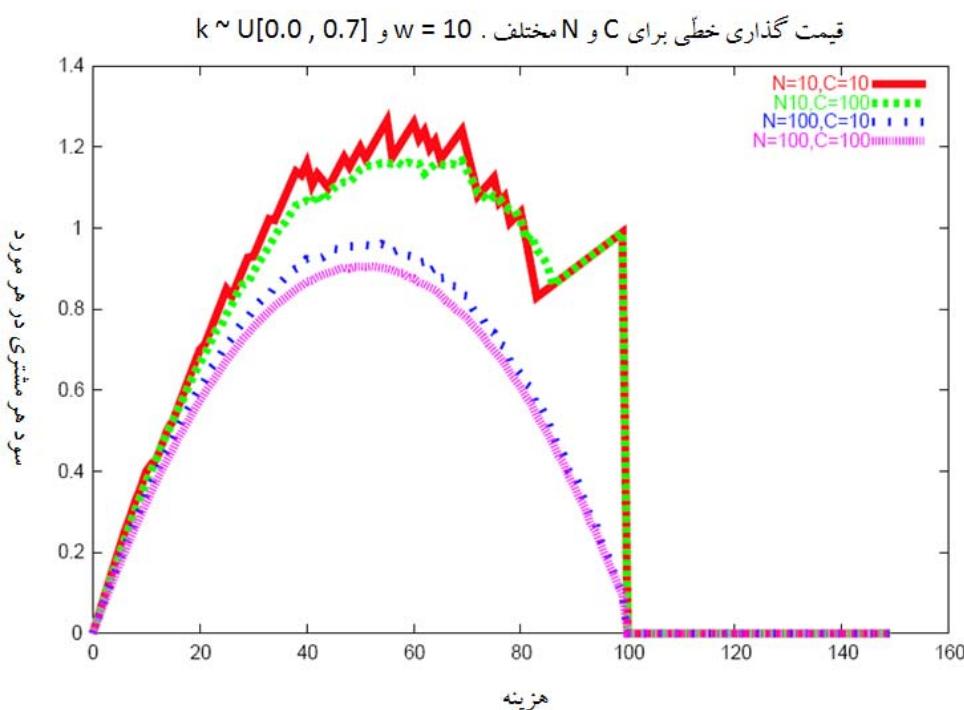
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دورنما (چشم انداز^۱) جستجو

جستجوی محلی ، معمولاً برای مسایل بهینه سازی ، مفید می باشد ؛ "پارامترهایی را پیدا نمایید که $O(x)$ بیشینه / کمینه باشد " ، این روش زمانی که فضای حالت ، ترکیبی از پارامترها می باشد ، مشکل است . در صورتی که n پارامتر وجود داشته باشد ، ما می توانیم یک فضای $n+1$ بعدی را تصوّر نماییم ، که n بعد اول ، پارامترهای تابع می باشند و $n+1$ امین بعد ، تابع هدف^۲ می باشد که ما این فضا را یک چشم انداز جستجو می نامیم . توجه کنید نقاط بهینه روی تپه ها هستند و گودی ها ، راه حل های ضعیف هستند . یک چشم انداز یک بعدی را در شکل زیر مشاهده می نمایید :



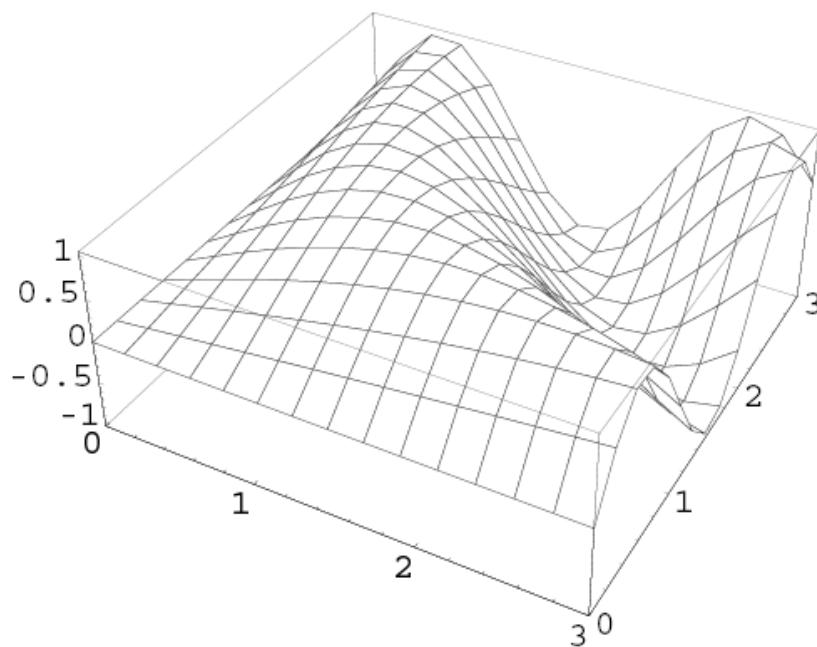
landscape^۱
objective function^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در شکل زیر یک چشم انداز دوبعدی نشان داده شده است:



دورنماها ، یک روش نمایش خوب برای الگوریتم های جستجوی محلی هستند . بالا رفتن از یک تپه را در نظر مجسم کنید ؛ این مورد یک راه را برای تشخیص مسایل آسان از مسایل سخت ، ارایه می کند .
حالت های آسان ؛ تعداد قله های کم ، سطوح صاف ، بدون برآمدگی (خط الرأس) / مسطح (فلات) هستند و
حالت های سخت : تعداد قله های زیاد ، سطوح دندانه دار یا ناپیوسته (گسسته) هستند . در شکل زیر که
شکل مسئله ۴- وزیر است وقتی که $h = 0$ می باشد ما هیچ حالت نادرستی را نداریم که در این مورد
کاملاً راحت هستیم ولی وقتی که $h = 2$ است ما دارای دو حالت نادرست هستیم (چون که دو وزیر در یک
ردیف قرار گرفته اند و دو وزیر دیگر به صورت مورب قرار گرفته اند) و کار ما نسبت به حالتی که $h = 0$
بود سخت تر است و در حالت $h = 5$ چون که پنج مورد نادرست داریم کار ما نسبت به دو مورد قبلی که

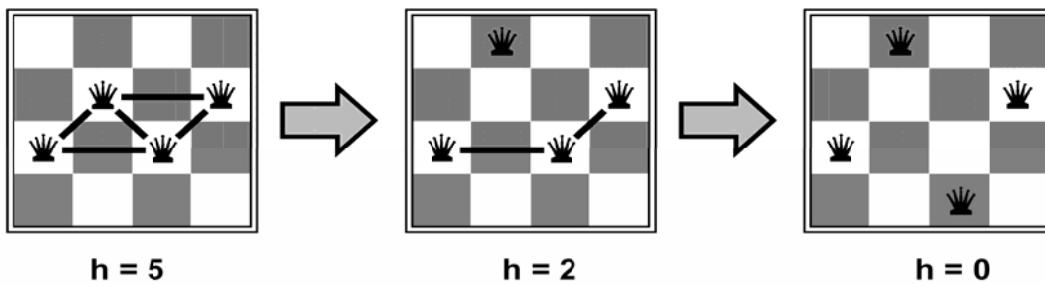
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$h = 0$ بود سخت تر است. در حالتی که $h = 0$ است می توانیم تصوّر کیم که در جایی مسطح قرار گرفته ایم و در حالتی که $h = 5$ است می توان تصوّر کرد که در شیب یک تپه قرار گرفته ایم.^۱



الگوریتم تپه نوردی

ساده ترین شکل جستجوی محلی، جستجوی تپه نوردی می باشد و دارای روشی ساده است به این ترتیب که در هر نقطه ، به جانشینان (همسایه ها) نگاه کنید و در جهت بهتر ، حرکت نمایید . این روش خیلی شبیه به جستجوی حریصانه می باشد و به حافظه ای خیلی کمی نیاز دارد . فلات می تواند سبب این شود که الگوریتم ، بی هدف ، سرگردان باشد .

جستجوی محلی در حساب دیفرانسیل و انتگرال^۲

ابتدا ریشه های یک معادله $f(x)=0$ را به دست آورید ، f ، تشخیص پذیر می باشد . با فرض یک x_1 ، $f(x_1)$ و $f'(x_1)$ را به دست آورید . از خط تانزانت برای $f(x_1)$ استفاده نمایید تا x_2 را انتخاب نمایید (شیب $= f'(x_1)$) . و برای سایر نقطه ها از فرمول $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_1)}$ استفاده نمایید ؛ این کاری که انجام دادید یک جستجوی تپه نوردی می باشد و برای موارد مسطح ، خیلی خوب می باشد .

^۱ مترجم
calculus^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

الگوریتم

تابع Hill-Climbing(problem) یک وضعیت را برمی گرداند

current \leftarrow Initial-State(problem)

در حلقه کارهای زیر را انجام بده

بالاترین جانشین مقداردهی شده i current را در neighbour قرار بده

در صورتی که $Value(neighbour) \leq Value(current)$ را بردان current \leftarrow neighbour

الگوریتم تپه نوردی (به بیان دیگر)

تابع Hill-Climbing(problem) یک حالت که یک ماکزیمم محلی است را برمی گرداند

ورودی ها ، problem که یک مسئله است می باشد

متغیرهای محلی current و neighbor که هر کدام یک گره هستند ، می باشند.

current \leftarrow Make-Node(Initial-State[problem])

کارهای زیر را انجام بده

بالاترین مقداردهی neighbor \leftarrow current از successor (مقدار بالاترین جانشین neighbor را در current قرار می دهد)

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اگر $\text{Value}[\text{neighbor}] \leq \text{Value}[\text{current}]$ می باشد $\text{State}[\text{current}]$ را برگردان

$\text{current} \leftarrow \text{neighbor}$

پایان حلقه

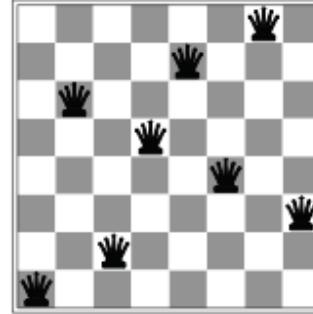
پایان الگوریتم

شروع مجلد تصادفی الگوریتم بالا رونده از تپه بر ماکزیمم محلی غلبه می کند.

مثال جستجوی تپه نورده

فرمول بندی کامل برای ۸- وزیر ، تابع جانشین : یک وزیر را به مربع دیگر در همان ستون ببرید (جانشین) که هزینه برابر است با تعداد جفت هایی که در جای نامناسب قرار دارند .

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	18	13	16	13	16
14	17	15	15	14	16	16	16
17	18	16	18	15	18	15	18
18	14	15	15	14	18	16	16
14	14	13	17	12	14	12	18



هزینه ای فعلی = ۱۷

هزینه ای کمینه ای محلی = ۱

خصوصیات تپه نورده

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پیاده سازی آن آسان می باشد؛ مقدار کمی از حافظه را استفاده می نماید ولی تپه نورده، زیاد تحت تأثیر شکل فضای حالت است در ماکزیمم محلی کم خوب می باشد ولی برای فضاهای جوچه تیغی (دندانه دار) مانند بد است. در شانه ها، ماکزیمم محلی و خط الراس^۱ ها دچار مشکل می شود؛ در ماکزیمم محلی نمی تواند قله ای بالاتر را بیند. در شانه، راهی به طرف بیرون را نمی تواند پیدا نماید و در خط الراس ها برای حرکت، شما باید به طرف پایین حرکت نماید و حرکت های بد را مجاز می داند.

تنوعات تپه نورده

حرکت های غیرمستقیم^۲ : روی کف^۳ (جای مسطح) حرکت می نماید و در حلقه نمی افتد
(یک تعداد بیشینه از حرکات را ثبت می کند)

تپه نورده اتفاقی : در میان حرکت های به طرف سربالایی یکی را به تصادف انتخاب می نماید

اوّلین انتخاب تپه نورده : اوّلین حرکت سربالایی را انتخاب می کند.

تپه نورده با شروع مجدد تصادفی : چند تپه نورده از وضعیت های اوّلیه ای انتخاب شده به صورت تصادفی است و تولید وضعیت های تصادفی می تواند غیر جزیی باشد.

بهبود روش تپه نورده

ridge^۴

sideways move^۵

plateau^۶

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تپه نوردهی ، می تواند خوشایند باشد ، برنامه نویسی آن آسان می باشد ، به فضای کمی نیازمند می باشد و در ضمن ، ممکن است ما ، روشی بهتر را نداشته باشیم . حال این سؤال مطرح می شود که چگونه این روش ، بهبود می یابد ؟ با استفاده از موارد زیر این روش بهبود می یابد :

- تپه نوردهی تصادفی - به صورت تصادفی حرکت های بالای تپه را انتخاب نمایید
- تپه نوردهی با شروع مجلد تصادفی
- تا زمانی که به یک بهینه برسید ، ادامه دهید
- یک وضعیت اوّلیه را به صورت تصادفی ، انتخاب نمایید
- مجلدآ ، اجرا نمایید .
- بعد از n تکرار ، بهترین راه حل رانگهداری نمایید .

روش شبیه سازی گرم و سرد کردن

ضعف روش تپه نوردهی ، این می باشد که هرگز به طرف پایین تپه حرکت نمی کند و شبیه به جستجوی حریصانه ، نمی تواند "پشتیبان"^۱ داشته باشد ؛ گرم و سرد کردن شبیه سازی شده ، تلاشی برای برطرف نمودن این موارد می باشد . ایده‌ی این روش ، گریختن از مانع‌یم محلی با اجازه دادن به بعضی از حرکت‌های بد یا نادرست می باشد . اما به تدریج تعداد و فراوانی حرکت‌های بد کاهش می یابد . در واقع این روش تپه نوردهی را با حرکت تصادفی ترکیب می نماید .

back up^۱
frequency^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گرم و سرد کردن: فلزات را با حرارت دادن آن ها و رساندن دمای آن ها به دمایی بالا و سرد کردن تدریجی آن ها سخت می کنند.

اگر یک توب پینگ را به عمیق ترین جای یک سطح دارای گودی (کمینه یا مینیمم مطلق) بیندازید، خارج کردن آن سخت ترازو قوتی است که آن را در یک کمینه ی محلی می اندازید.

شبیه سازی گرم و سرد کردن

وقتی که یک فلز مثل آهن را خیلی گرم می کنید دارای ارتعاش های زیادی است ولی وقتی که به تدریج دما کم می شود ارتعاش های آن هم به تدریج کم می شود در تپه نوردی این کار مثل این است که از کمینه ی محلی با اجازه دادن به برخی از حرکت های بد بگریزید، اما به تدریج تعداد و فراوانی آن ها را کاهش دهید.

الگوریتم شبیه سازی گرم و سرد کردن:

تابع (Simulated-Annealing(problem,schedule)) یک راه حل را بر می گرداند

وروودی ها، یک مسئله و schedule که یک نگاشت از زمان به دما می باشد، هستند.

متغیرهای محلی عبارتند از: current، که یک گره است و next، که یک گره است و T، که یک پروب کنترل کننده ی دما از مراحل پایین تر است.

current \leftarrow Make-Node(Initial-State[problem])

برای مقادیر $t = 1$ تا ∞ کارهای زیر را انجام بده:

$T \leftarrow \text{schedule}[t]$

در صورتی که $T=0$ بود گرهی current را برگردان

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک جانشین به تصادف انتخاب شده از گرهی $current$ را به گرهی $next$ نسبت بده

$$\Delta E \leftarrow \text{Value}[next] - \text{Value}[current]$$

در صورتی که $\Delta E > 0$ بود محتوای گرهی $next$ را در گرهی $current$ بروز

در غیر این صورت، محتوای گرهی $next$ را فقط با احتمال $e^{\Delta(E/T)}$ در گرهی $current$ کپی کن

پایان الگوریتم

روش شبیه سازی گرم و سرد کردن، همیشه حرکت های خوب را می پذیرد ولی در آن احتمال پذیرش یک حرکت بد وجود دارد.

خصوصیات شبیه سازی گرم و سرد کردن - در دمای ثابت T ، حالت کاری ممکن

است به توزیع بولتزمان^۱ بر سد.

$$p(x) = \alpha e^{\frac{E(x)}{kT}}$$

T با سرعت کمی کاهش داده می شود، در نتیجه، همیشه به بهترین حالت x^* می رسد. زیرا برای T های کوچک داریم:

$$e^{\frac{E(x^*)}{kT}} / e^{\frac{E(x)}{kT}} = e^{\frac{E(x^*) - E(x)}{kT}} \gg 1$$

آیا این الزاماً یک ضمانت قابل قبول است؟؟

Boltzman^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

از روش شبیه سازی گرم و سرد کردن در بسیاری از موارد مثل قالب بندی VLSI^۱، زمانبندی خطوط هوایی و غیره استفاده می شود. شبیه سازی گرم و سرد کردن در صورتی که T به اندازه‌ی کافی با سرعت کم تر، کاسته شود کامل و بهینه می باشد. به سادگی به الگوریتم تپه نورده اضافه می شود. خصوصیات این روش در انتخاب یک زمانبندی خنک کننده‌ی خوب می باشد.

جستجوی پرتو محلی^۲

ایده^۵ : به جای یک حالت k ، حالت های k را نگه می دارد و بالاترین مقدار k را در بین همه می آن ها (جانشینان یا نامزدها) انتخاب می کند. همانند جستجوهای k که در حالت موازی اجرا می شوند، نمی باشد. جستجو می کند تا حالت های خوب دیگر جستجو ها را پیدا کند و آن ها را به هم پیوند دهد. این روش اغلب کامل است، ولی تمام حالت های k در نهایت در بالای تپه می محلی قرار می گیرند؛ در این مورد راه حل این است که جانشینان k را به طور تصادفی انتخاب نماییم. این روش، یک روش بربنای مسیر است که به چند مسیر "اطراف" مسیر جاری نگاه می کند و k وضعیت را به جای یک وضعیت در حافظه نگهداری می نماید و با k گره می شود و در غیر این صورت تعداد k تا از بهترین جانشینان انتخاب می شوند. در این روش اطلاعات میان وضعیت ها می توانند مشترک شوند و به وعده داده شده ترین ناحیه ها حرکت می نماید. روش جستجوی پرتو محلی اتفاقی تعداد k جانشین را به صورت تصادفی انتخاب می نماید که این کار به وسیله می احتمالی که توسط تابع ارزیابی تشخیص داده می شود انجام می شود.

الگوریتم های ژنتیکی

^۱ مخفف Very Large Scale Integration می باشد؛ تعداد زیادی ترانزیستور که روی یک صفحه یا تراشه (chip) می باشد. کوچک قرار گرفته باشند (مترجم).

² local beam search^{*}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

الگوریتم های ژنتیکی می توانند به صورت یک شکل از جستجوی موازی تپه نورده ، تصوّر شوند . ایده ای اساسی این است که برخی از راه حل ها را به صورت تصادفی ، انتخاب نمایید و برای به دست آوردن راه حل های جدید ، بهترین تکه های راه حل ها را با هم ترکیب نمایید و این کار را تکرار نمایید . در این روش جانشین (کاندید یا نامزد) ها ، تابعی از دو وضعیت می باشند .

الگوریتم های ژنتیکی = جستجوی پرتوی محلی اتفاقی + تولید جایگزین های از

جفت حالت ها

جستجوی پرتو محلی به شکل "انتخاب طبیعی" : جانشینان (موجود تولید شده توسط والدین^۱) یک وضعیت (زیستمند^۲) نسل بعدی را با توجه به مقدارشان (سازگاری^۳ شان) تولید می کنند .

الگوریتم های ژنتیکی (GA ها) : الگوریتم های نمونه بعد از تکامل زیستی هستند .

در روش جستجوی پرتو اتفاقی ، وضعیت های جانشین به صورت تنوعاتی از دو وضعیت والد تولید می شوند ، نه فقط یکی .

اصطلاحات الگوریتم ژنتیکی

جمعیت^۴ ، به صورت اوّلیه مجموعه ای از k وضعیت تولید شده به صورت تصادفی است و مجموعه ای از وضعیت های مشتق شده است .

offspring ^۱
organism ^۲
fitness ^۳
population ^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نسل^۱، جمعیت در یک نقطه از زمان است و معمولاً، انتشار^۲ برای تمام جمعیت همزمان^۳ می شود

فرد^۴، یک عنصر از جمعیت است و به صورت یک رشته با الفبای محدود توصیف می شود.

تابع مناسب^۵، هر چقدر که تابع مناسب تری انتخاب کنیم منجر به شانس های بهتر برای تولید مجدد می شود و واژه‌ی مناسب بودن بیان کننده‌ی خوبی یک راه حل می باشد.

ژنوم^۶، یک راه حل یا وضعیت است.

ویژگی / ژن^۷ - یک پارامتر یا متغیر وضعیت است.

یک الگوریتم ژنتیکی پایه

یک جمعیت تصادفی را تولید کن و آن را در pop قرار بده

مادرامی که انجام نشده است، کارهای زیر را انجام بده

برای هر p در pop

p را مورد ارزیابی قرار بده

generation^۸

propagation^۹

synchronize^{۱۰}

individual^{۱۱}

fitness function^{۱۲}

genome^{۱۳}؛ تمامی ترکیب ژنتیکی یک فرد یا جمعیتی که بوسیله کروموزومها به ارث می رسد.

trait/gene^{۱۴}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برای $i=1$ تا اندازه i pop کارهای زیر را انجام بده

راه حل های تصادفی را از pop ، انتخاب نما

راه حل ها crossover نما

راه حل ها را تغییر بده

راه حل های جدید را در pop قرار بده

رمز کردن یک مسئله

فرض کنید که ما می توانیم مسئله امان را به صورت یک رشته بیتی ، رمز نماییم . مثلاً مسئله ۸ وزیر را در نظر بگیرید . برای هر ستون ، ما از سه بیت برای رمز نمودن ردیف وزیر استفاده می کنیم = ۲۴ بیت .

$$100 \ 101 \ 110 \ 000 \ 101 \ 001 \ 010 \ 110 = 4 \ 5 \ 6 \ 0 \ 5 \ 1 \ 2 \ 6$$

ما با تولید رشته های بیتی تصادفی شروع می کنیم ، سپس آن ها را با توجه به تابع مناسب (تابعی برای بهینه نمودن) مورد ارزیابی قرار می دهیم .

تولید راه حل های جدید

تابع جانشین ما با کاربر روی دو راه حل ، کار می کند که این کار ، crossover نام دارد . برای انجام عمل crossover دو راه حل را به صورت تصادفی انتخاب نمایید . برای انتخاب شایستگی مناسب ؛ تمام شایستگی ها را جمع نمایید ؛ داریم احتمال انتخاب $s =$ شایستگی s ، تقسیم بر مجموع شایستگی . سپس یک نقطه (مکان هندسی) تصادفی را در رشته های بیتی انتخاب نمایید و تکه های او را با تکه های دوم b_2 برای به وجود آوردن دو رشته بیتی جدید ، با هم ادغام نمایید .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

crossover مثال

$$s1: (100\ 101\ 110)\ (000\ 101\ 001\ 010\ 110) = 4\ 5\ 6\ 0\ 5\ 1\ 2\ 6$$

$$s2: (001\ 000\ 101)\ (110\ 111\ 010\ 110\ 111) = 1\ 0\ 5\ 6\ 7\ 2\ 6\ 7$$

مکان هندسی را برابر ۹ قرار دهید

$$s3 = (100\ 101\ 110)\ (110\ 111\ 010\ 110\ 111) \quad 4\ 5\ 6\ 6\ 7\ 2\ 6\ 7$$

$$s4 = (001\ 000\ 101)\ (000\ 101\ 001\ 010\ 110) \quad 1\ 0\ 5\ 0\ 5\ 1\ 2\ 6$$

سپس ، جهش را اعمال نمایید . با احتمال m (برای m کوچک) به صورت تصادفی ، یک بیت در راه حل را انتخاب نمایید . بعد از تولید یک جمعیّت جدید از همان اندازه به صورت جمعیّت قدیمی ، جمعیّت قبلی را حذف نمایید و دوباره شروع نمایید . crossover ، قطعات راه حل های موفق به صورت جزئی را با هم ترکیب می کند . ژن های نزدیک تر به هم ، برای ماندن با هم در رشته‌ی نسل ها ، بهتر می باشند . این کار ، رمز کردن را مهم می کند . جهش ، راه حل های جدید را به جمعیّت اعمال می کند . در صورتی که یک ژن از جمعیّت اولیه گم شده بود ، crossover نمی تواند آن را تولید کند .

اصول الگوریتم ژنتیکی

تولید مجدد توسط crossover : دو قسمت کردن ^۱ والدین در یک نقطه ای crossover داده شده و ترکیب شده ؛ این کار دو وضعیّت جدید را تولید می نماید .

نقطه ای crossover : برای جلو بردن کار استفاده می شود و اغلب به صورت تصادفی انتخاب می شود ؛ برای هر دو والد باید یکسان باشد و باید نسل های قابل قبول را تولید نماید .

split ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

جهش^۱: تولید مشخصه های جدید که در والدها وجود ندارد؛ تغییرات تصادفی اجزای رمز کننده در نسل که توسط احتمال کنترل شده است.

الگو^۲: اجزای مفید یک راه حل می توانند در میان نسل ها باقی بمانند.

انتخاب مناسب ترین فرضیه^۳ ها

انتخاب شایستگی مناسب با استفاده از فرموزیر می توانند منجر به تولید چند کپی منتشر شده شود.

$$P(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^k Fitness(h_j)}$$

انتخاب مسابقه ای^۴: دو فرد را به صورت تصادفی و با احتمال یکسان انتخاب نمایید سپس با یک احتمال ثابت، مورد مناسب تر را انتخاب نمایید.

انتخاب ردیف^۵: همه فرضیه ها را براساس شایستگی مرتب نمایید؛ احتمال انتخاب با ردیف مناسب است.

crossover

مرکب از تکه های ترکیب کننده ای افراد برای ساختن افراد جدید است.

mutation^۱

schema^۲

hypothesis^۳

tournament^۴

rank^۵

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تک نقطه ای Crossover را انتخاب نمایید و Crossover ها را

از آنجا بردارید و قطعه ها را تعویض نمایید؛ به عنوان مثال:

101|1110

0111110

crossover

011|0101

1010101

گام بعد:

101|1110

0111110

crossover

011|0101

1010101

گام بعدی:

101|1110

0111110

crossover

011|0101

1010101

گام بعدی:

101|1110

0111110

crossover

011|0101

1010101

گام بعدی:

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

101|1110

0111110

crossover011|**0101**101**0101**

گام بعدی :

101|1110

0111110

crossover

011|0101

1010101

پیاده سازی: از یک ماسک **crossover** (رشته‌ی دودویی) استفاده نمایید که در مثال 1110000 می‌باشد. دو والد h_1 و h_2 ارایه شده: $(h_1 \wedge m) \vee (h_2 \wedge \neg m), (h_1 \wedge \neg m) \vee (h_2 \wedge m)$ می‌باشند. با ماسک های اخنیاری به صورت زیر است:

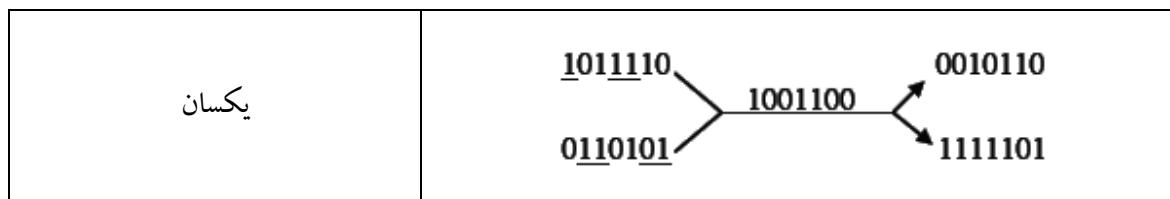
	اویله	ماسک	نسل
تک نقطه		$\begin{array}{c} \underline{1011110} \\ \quad \quad \quad \swarrow \\ \quad \quad \quad 1110000 \\ \quad \quad \quad \searrow \\ \underline{0110101} \end{array}$	$\begin{array}{c} 0111110 \\ \quad \quad \quad \swarrow \\ \quad \quad \quad 1010101 \\ \quad \quad \quad \searrow \\ 1010101 \end{array}$
دو نقطه		$\begin{array}{c} \underline{1011110} \\ \quad \quad \quad \swarrow \\ \quad \quad \quad 0011100 \\ \quad \quad \quad \searrow \\ \underline{0110101} \end{array}$	$\begin{array}{c} 1010110 \\ \quad \quad \quad \swarrow \\ \quad \quad \quad 0111101 \\ \quad \quad \quad \searrow \\ 0111101 \end{array}$

مترجم: سه راب جلوه گر

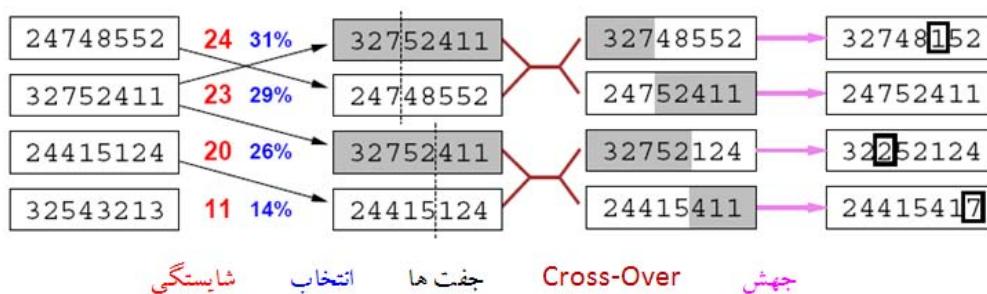
ویرایش دوم، بهار ۱۳۸۸



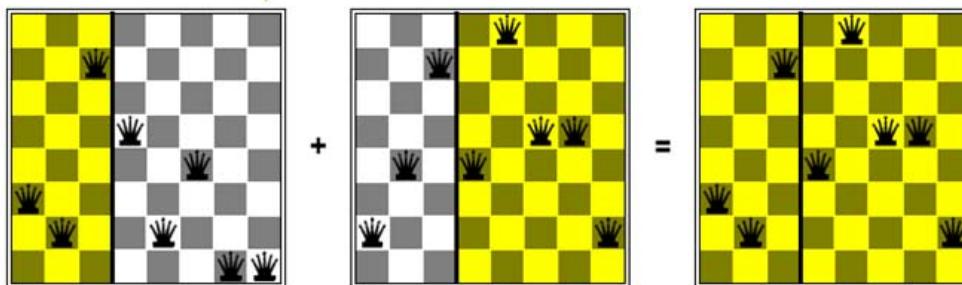
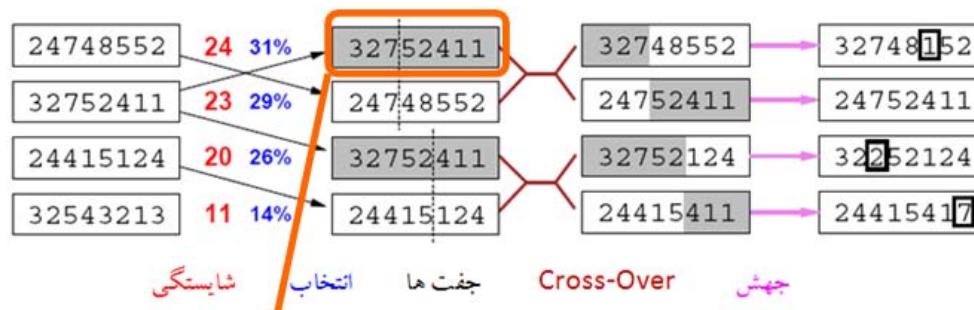
هوش مصنوعی



مثال : ۸ - وزیر



گام بعدی :



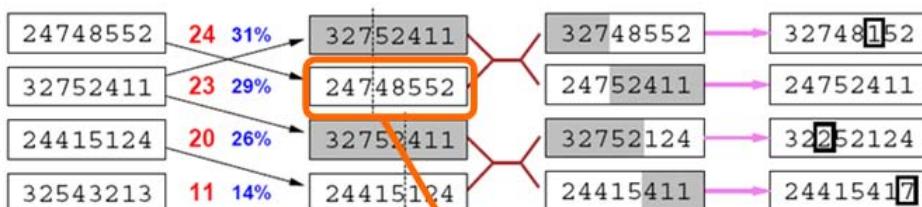
مترجم: سه راب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸

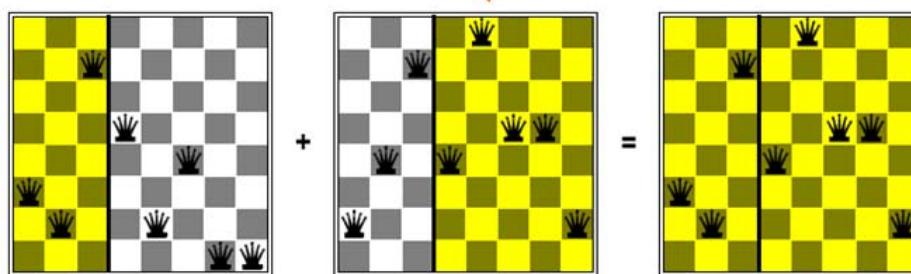


هوش مصنوعی

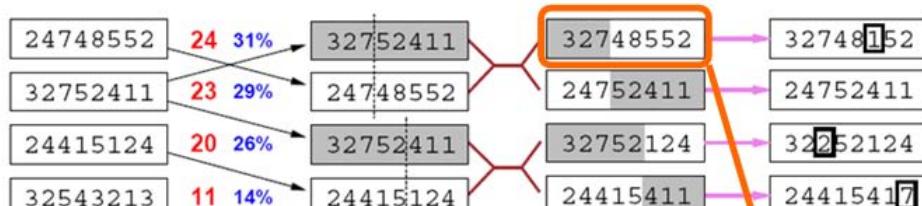
گام بعدی :



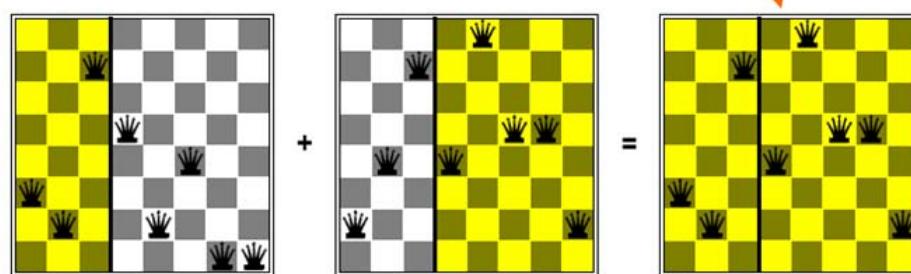
شایستگی انتخاب جفت ها Cross-Over جهش



گام بعدی :



شایستگی انتخاب جفت ها Cross-Over جهش



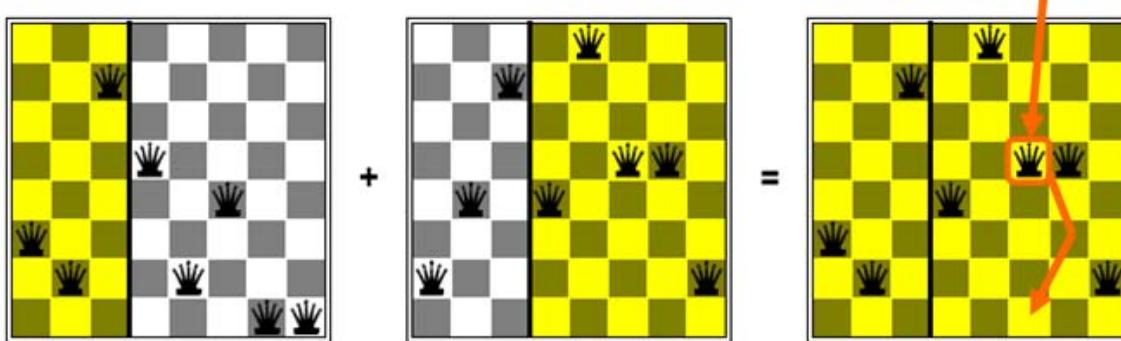
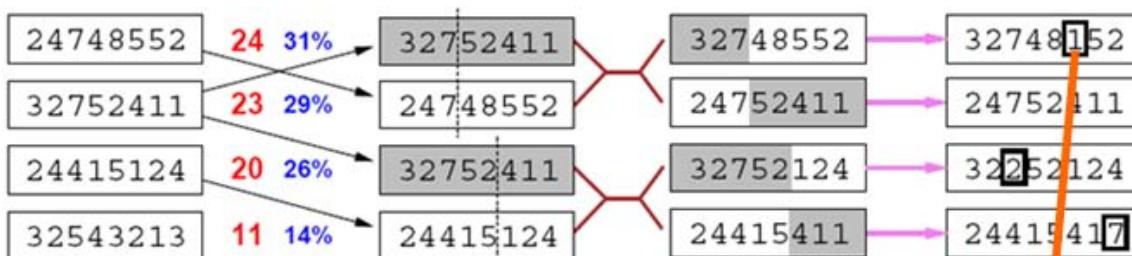
مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گام بعدی:



الگوریتم های ژنتیکی به حالت های رمزگذاری شده به صورت رشته نیاز دارند (GP ها از برنامه ها استفاده می کنند) . به زیر رشته های crossover iff ، که اجزایی معنی دار هستند کمک می کند. الگوریتم های ژنتیکی با تکامل مخالفند ؛ عوامل تکامل ^۱ واقعی ، تشکیلات ^۲ تکراری را رمز گذاری ^۳ می کند .

الگوریتم های ژنتیکی به صورت جستجو

genes ^۱
machinery ^۲
encode ^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

وضعیت ها ، راه حل های ممکن هستند . عملگرهای جستجو : جهش ، دورگه (crossover) و انتخاب هستند . از آنجایی که برای به وجود آوردن جمعیت ها چند راه حل به صورت موازی به کار گرفته می شوند . با انتخاب تابع مناسب ، مثل روش تپه نوردی ، بدون سراشیبی می باشد . جهش و دورگه باید به ما اجازه دهنده که از کمینه ی (مینیمم) محلی خارج شویم . وابسته به روش جستجوی شبیه سازی گرم و سرد کردن می باشد .

نکته : برنامه نویسی ژنتیکی وابسته به الگوریتم ژنتیکی می باشد و در آن ها فردها (رشته ها) همان برنامه ها هستند .

کاربردهای الگوریتم ژنتیکی

اغلب برای مسایل بهینه سازی مورد استفاده قرار می گیرد مثل : آرایش مدار ، طراحی سیستم و زمانبندی . به طور خلاصه باید بگوییم که به اندازه ی کافی برای پیدا کردن راه حل ، مناسب می باشد ولی بهبود قابل توجهی در مورد چند نسل ندارد و دارای محدودیت زمانی هستند .

فضاهای حالت پیوسته - تصویر کنید که می خواهیم جای سه فرودگاه را در کشور رومانی مشخص نماییم: فضای حالت شش بعدی توسط $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ تعریف می شود . تابع هدف : $f(x_1, y_1, x_2, y_2, x_3, y_3)$ = مجموعه ای از فواصل مربّعی از هر شهر به نزدیک ترین فرودگاه می باشد . متدهای مجزا^۱ ، فاصله های پیوسته را به فاصله های مجزا تبدیل می نمایند . گرادیان تجربی با تغییرات δ در هر وضعیت متناسب است . گرادیان از فرمول زیر محاسبه می شود:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$

discretization methods^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

با توجه به رابطه $x \leftarrow x + \alpha \nabla f(x)$ می توانیم مقدار f را افزایش (کاهش) دهیم ، گاهی از اوقات می توان این معادله را به درستی برای حالت $\nabla f(x) = 0$ در صورتی که فقط یک شهر داشته باشیم حل نماییم . در روش نیوتن - رفسون^۱ (1664-1690) برای حل $\nabla f(x) = 0$ در جایی که

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

از فرمول زیر استفاده می شود:

$$x \leftarrow x - H_f^{-1}(x) \nabla f(x)$$

Newton-Raphson^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل ششم

مسایل اراضی محدودیت^۱

Constraint Satisfaction Problems (CSPs)^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

رئوس مطالب

- مثال های مسایل اراضی محدودیت
- جستجوی معکوس لیست^۱ برای مسایل اراضی محدودیت
- ساختار مسئله و تجزیه^۲ ی مسئله
- جستجوی محلی برای مسایل اراضی محدودیت

مسایل اراضی محدودیت

مسئله ی جستجوی استاندارد : حالت ، یک " جعبه ی سیاه^۱" است - هر ساختمان داده ای قدیمی که از تست اهداف ، ارزیابی و جانشین پشتیبانی می کند .

backtracking^۱
decomposition^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مسئله‌ی اراضی محدودیت: حالت، توسط متغیرهای X_i و با مقدارهای درون دامنه

D_i تعریف می‌شود. تست هدف، مجموعه‌ای از محدودیت‌های معین کننده‌ی محدودیت‌های مجاز از مقادیر زیرمجموعه‌ی متغیرها می‌باشد. یک زبان ارایه‌ی رسمی^۲ مثال ساده‌ای از آن می‌باشد، الگوریتم‌های **همه منظوره**^۳ مفید را با توانایی بیشتر از الگوریتم‌های جستجوی استاندارد سبب می‌شود.

مثال نقشه‌ی رنگی: متغیرها عبارتند از: WA, NT, Q, NSW, V, SA, T: دامنه‌ها:

$$D_i = \{red, green, blue\}$$



^۱: نوعی از مهندسی معکوس است که هدف اصلی آن بررسی رفتار برنامه بدون توجه به کد برنامه‌ی آن است. مهندسی معکوس، از حالت کامپایل در آوردن یا تکه تکه کردن برنامه‌ی کاربردی بدون توجه به زبان برنامه نویسی ای که توسط آن به وجود آمده است می‌باشد بنابراین ممکن است به کد اصلی برنامه دسترسی پیدا نماید. (Babylon / Code)
² Analysis formal representation language
³ general-purpose

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

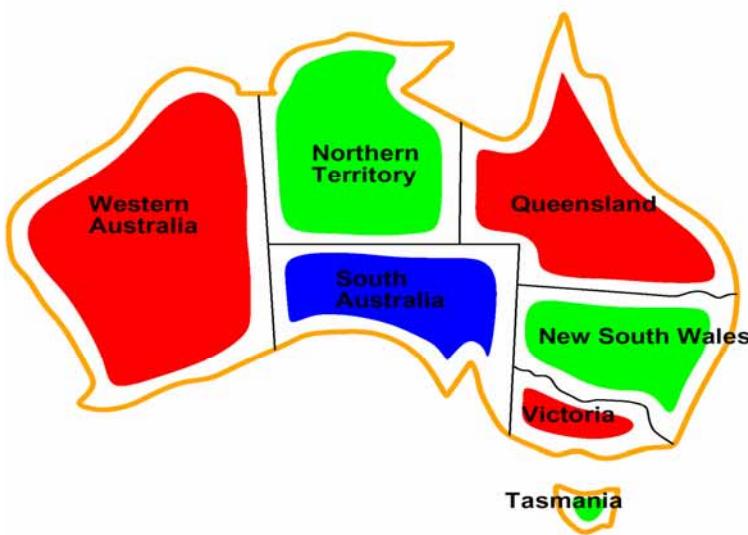


هوش مصنوعی

محدودیت ها: نواحی مجاور، باید رنگ های متفاوتی داشته باشند. به عنوان مثال $WA \neq NT$

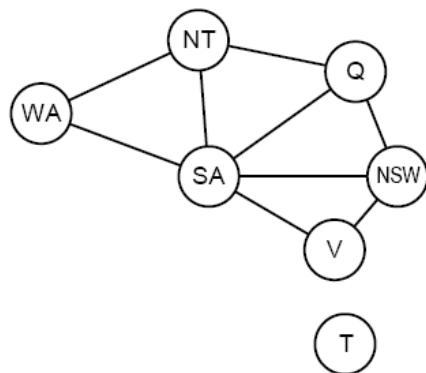
($WA, NT \in \{ (red, green), (red, blue), (green, red), (green, blue), \dots \}$)

گام بعدی:



راه حل ارایه شده همهی شرایط را لحاظ می کند، به عنوان مثال،

{ $WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green$ }



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گراف محدودیت^۱

مسئله‌ی اراضی محدودیت دودویی^۲: هر محدودیت حداکثر دو متغیر را تشریح می‌نماید.

گراف محدودیت: در این گراف، گره‌ها، متغیرها هستند، کمان‌ها^۳، محدودیت‌ها را نشان می‌دهند. الگوریتم‌های همه منظوره‌ی مسئله‌ی اراضی محدودیت از ساختار درختی برای بالا بردن سرعت جستجو استفاده می‌کنند. توجه نمایید که شهر تاسمانیا یک زیر مسئله‌ی مستقل است!

أنواع مسالی اراضی محدودیت

مسالی اراضی محدودیت با متغیرهای مجزا (گسسته)، در دامنه‌های محدود؛ اندازه d است \leftarrow (در نتیجه) $O(d^n)$ و انتساب‌ها کامل است. به عنوان مثال، مسالی اراضی محدودیت بولین^۴ (NP-کامل). در دامنه‌های نامحدود (integer) ها، string ها (رشته‌ها) و به عنوان مثال در زمانبند کار^۵، متغیرها روزهای شروع / پایان برای هر کار هستند و به یک زبان محدود^۶ نیازمندند، به عنوان مثال، $.StartJob_1 + 5 \leq StartJob_3$

مسالی اراضی محدودیت با متغیرهای پیوسته^۷: به عنوان مثال، زمان‌های شروع / پایان برای مشاهدات تلسکوپ^۸ هابل^۹

Constraint graph^۱

Binary CSP^۲

arcs^۳

Boolean CSPs^۴

job scheduling^۵

constraint language^۶

continuous variables^۷

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

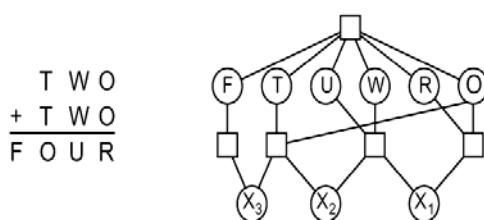


هوش مصنوعی

تنوع محدودیت ها

محدودیت های منحصر به فرد ^۳ ، شامل یک متغیر منفرد می شوند ، به عنوان مثال ، $SA \neq WA$ ؛ محدودیت های دودویی ، شامل جفت هایی از متغیرها می باشند ، به عنوان مثال ، $SA \neq WA$ و محدودیت های مرتبه ای بالاتر ، شامل تعداد سه یا بیش تر متغیر می باشند ، به عنوان مثال ، محدودیت های با ستون های پنهان (رمزی) ^۴ .

مثال : مسائل رمزی ، ما باید هر حرف را طوری تغییر دهیم که حاصل جمع به دست آید .



متغیرها : $X_3, X_2, X_1, F, T, U, W, R, O$. دامنه ها : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. محدودیت ها $O+O=R+10$ و $alldiff(F, T, U, W, R, O)$:

مسائل ارضای محدودیت در دنیای واقعی

مسائل انتساب ، به عنوان مثال ، چه کسی درس می دهد و در چه کلاسی ؟ مسائل برنامه زمانی ، کدام کلاس ارایه می شود ، چه زمانی و کجا ؟ ؛ پیکربندی سخت افزاری - صفحات گسترده ^۱

Telescope ^۱

Hubble ^۲

unary ^۳

cryptarithmetic column constraints ^۴

^۱ برنامه ای که امکان اجرای محاسبات روی چندین ستون از اعداد را برقرار کند ، spreadsheet نام دارد

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- زمان بند کننده‌ی انتقال - زمان بند کننده‌ی مرکز تولید. توجه کنید که تعدادی از مسایل دنیا واقعی شامل متغیرهای با مقدار واقعی هستند.

فرمول بندی جستجوی استاندارد، بیاید به صورت ساده شروع کنیم، برخورد ساده‌ای داشته باشیم و سپس آن را اثبات نماییم. تا کنون حالت‌ها توسط مقادیر نسبت داده شده تعریف می‌شدند: حالت اوّلیه را برابر با تهی در نظر می‌گیریم؛ تابع جایگزین، یک مقدار را به یک متغیر فاقد مقدار شده نسبت می‌دهد طوری که با مقدار فعلی تعارض یا ناسازگاری نداشته باشد. در نتیجه در صورتی که مقداردهی‌های مجاز وجود نداشته باشد، رد می‌شود. آزمایش یا تست هدف: بررسی می‌کند که انتساب جاری کامل باشد.

جستجوی معکوس لیست - انتساب متغیرها، جایجایی پذیر می‌باشد، توجه نمایید که، نمایید که انتساب‌ها برای یک متغیر منفرد در هر گره می‌باشد، در نتیجه $d=b=d^n$ گره وجود دارد. **جستجوی اول عمق برای مسایل اراضی محدودیت با انتساب های متغیر منفرد**، **جستجوی معکوس لیست نام دارد**. جستجوی معکوس لیست، اساس الگوریتم ناگاهانه برای مسایل اراضی محدودیت می‌باشد. می‌توان n - وزیر را برای $25 \approx n$ حل نمود.

الگوریتم

تابع (Backtracking-Search(csp)، راه حل یا عدم موقّیت را برمی‌گرداند

تابع (Recursive-Backtracking({ },csp) را برگردان

تابع (Recursive-Backtracking(assignment,csp) را برمی‌گرداند.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اگر انتساب (assignment) کامل است انتساب را برگردان

$\text{var} \leftarrow \text{Select-Unassigned-Variable}(\text{Variable}[\text{csp}], \text{assignment}, \text{csp})$

برای هر مقدار Order-Domain-Values(var, assignment, csp) در کارهای

زیر را انجام بده

در صورتی که value با $\text{assignment}[\text{csp}]$ ارایه شده در Constraint[csp] سازگار است

را به انتساب (assignment) اضافه نما $\{ \text{var} = \text{value} \}$

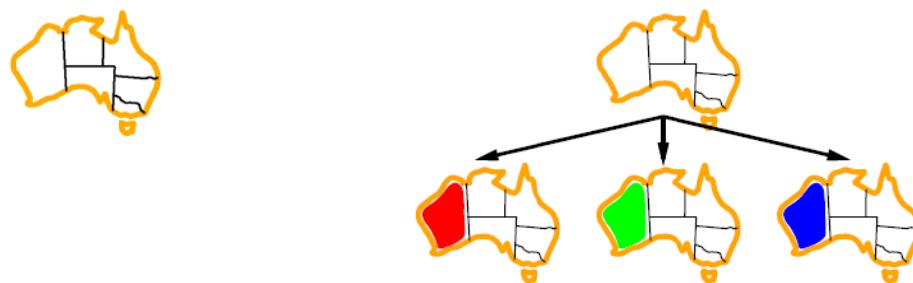
$\text{result} \leftarrow \text{Recursive-Backtracking}(\text{assignment}, \text{csp})$

در صورتی که result ≠ failure است، result را برگردان

را از انتساب $\{ \text{var} = \text{value} \}$ بردار

عدم موفقیت را برگردان

مثال لیست معکوس (تصاویر را از چپ به راست دنبال نمایید)

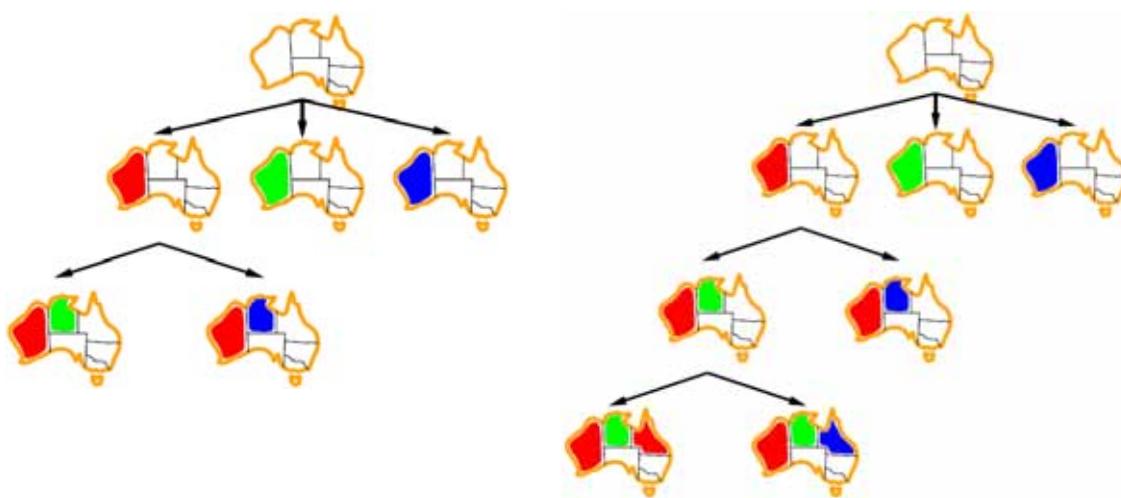


assignment¹

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



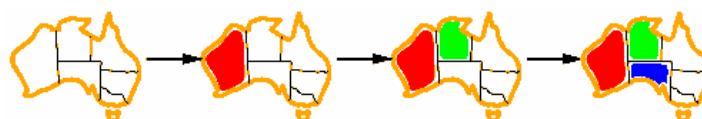
بهبود بخشیدن به کارایی لیست معکوس

متدهای همه منظوره می توانند افزایش زیادی در سرعت داشته باشند:

کم ترین مقادیر باقی مانده - کم ترین مقادیر باقی مانده^۱: متغیری را با کمترین مقادیر مجاز انتخاب نمایید.

قرمز دارای محدودترین مقدار می باشد .a

متغیری با کم ترین مقادیر مجاز را انتخاب نمایید .b



Minimum Remaining Values (MRV)^۱

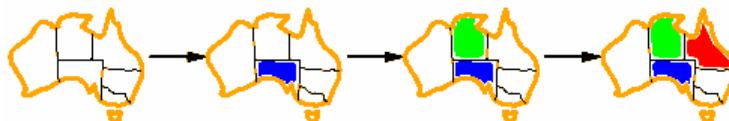
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



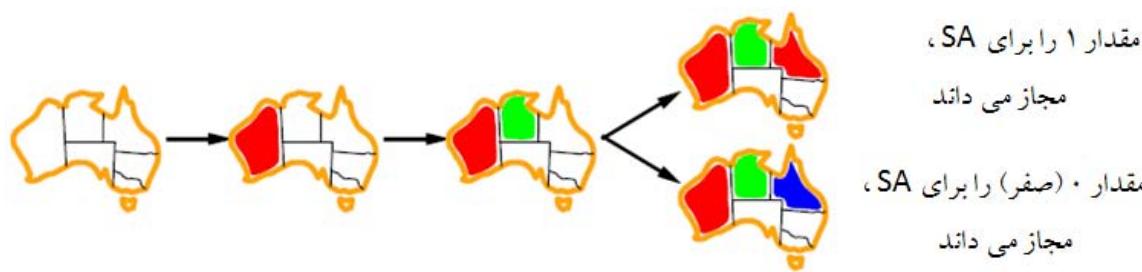
هوش مصنوعی

پیدا کردن درجه - متغیرها را در طول MRV به طور تصادفی بشکنید. کشف درجه: متغیری با بیشترین محدودیت در متغیرهای باقی مانده را انتخاب نماید.



کم ترین مقدار محدود کننده - یک متغیر ارایه شده، کم ترین مقدار محدود کننده را

انتخاب می نماید



ترکیب کردن این ابتکارات، داشتن ۱۰۰۰ وزیر را امکان پذیر می سازد.

بررسی مستقیم^۱ - مسیر باقی مانده با مقدارهای مجاز را برای متغیرهای انتساب داده نشده نگهداری نماید و جستجو را هنگامی که هر متغیر مقدارهای مجاز ندارد خاتمه دهید.



WA	NT	Q	NSW	V	SA	T
█	█	█	█	█	█	█

forward checking^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

: گام بعدی



WA	NT	Q	NSW	V	SA	T
Red	Green	Blue	Red	Green	Blue	Red
Red	White	Green	Blue	Red	Green	Blue

: گام بعدی



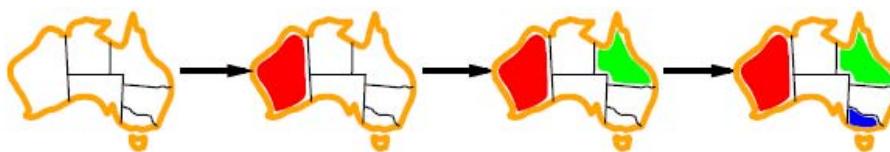
WA	NT	Q	NSW	V	SA	T
Red	Green	Blue	Red	Green	Blue	Red
Red	White	Green	Blue	Red	Green	Blue
Red	White	Blue	Green	Red	Blue	Red

: گام بعدی

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



توزيع محدود - بررسی اطلاعات توزیع های مستقیم از متغیر های منتب ب متغیر های منتب نشده ، به صورت سریع برای همه ی عدم موتفیت ها عمل نمی کند :



SA و NT هر دو نمی توانند آبی باشند ! توزیع محدود ، به صورت تکراری ، محدودیت ها را به صورت محلی اجرا می کند .

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

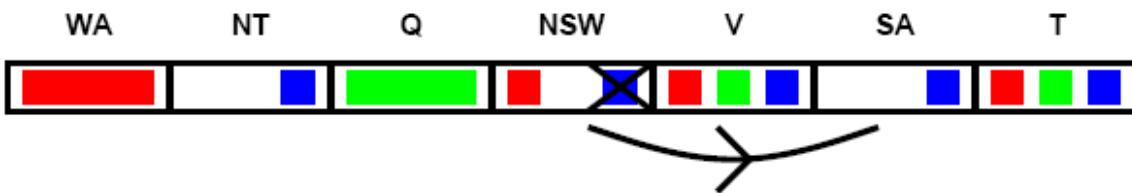


هوش مصنوعی

سازگاری قوس^۱ - ساده ترین توزیع، سازگاری هر قوس را سبب می شود. $X \rightarrow Y$ سازگار می باشد اگر برای هر مقدار X از X تعدادی برای Y مجاز باشند.



گام بعدی:



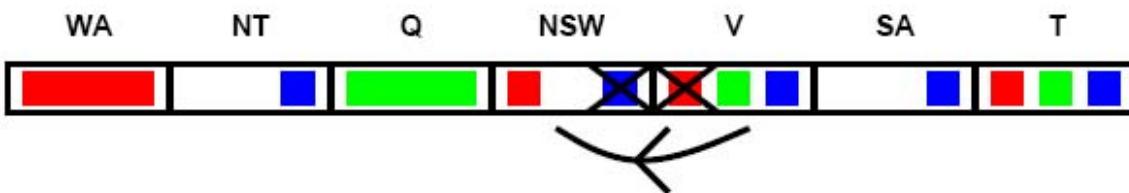
گام بعدی:

Arc consistency (AC)^۱

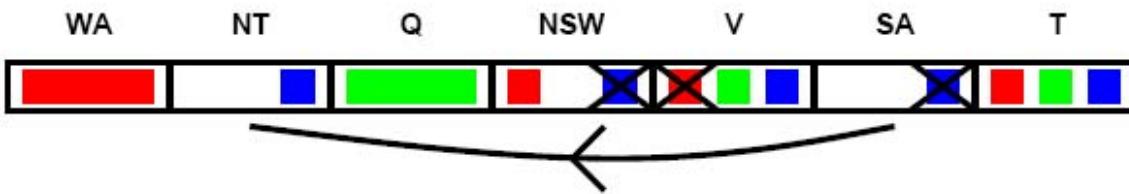
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



اگر X یک مقدار را از دست داد ، همسایه های X باید دوباره بررسی شوند .



اگر X یک مقدار را از دست داد ، همسایه های X باید دوباره بررسی شوند. سازگاری قوس زودتر از بررسی مستقیم ، عدم موافقیت پیدا می کند و می تواند به صورت یک پیش پردازنده و یا بعد از هر انتساب اجرا شود .

الگوریتم سازگاری قوس

تابع (CSP، AC-3(csp) و شاید دامنه های کاهش داده شده را برگرداند

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ورودی ها، CSP، که یک مسئله‌ی ارضای محدودیت دودویی با متغیرهای $\{X_1, X_2, \dots, X_n\}$ می‌باشد

متغیرهای محلی، queue، که یک صف از قوس‌ها که همه به صورت اولیه قوس‌هایی در csp هستند

مادامی که queue خالی نیست کارهای زیر را انجام بده

$$(X_i, X_j) \leftarrow \text{Remove-First}(queue)$$

اگر (X_i, X_j) Remove-Inconsistent-Values آن گاه

برای هر X_k در X_i Neighbours(کارهای زیر را انجام بده

$$(X_k, X_i) \text{ را به queue اضافه کن}$$

تابع (X_i, X_j) Remove-Inconsistent-Values را بر می‌گرداند

`remove ← false`

برای هر x در $Domain[X_i]$ کارهای زیر را انجام بده

در صورتی که هیچ مقداری در $Domain[X_j]$ اجازه نمی‌دهد که (x, y) محدودیت j را ارضاء کند آن گاه x را از $Domain[X_i]$ حذف کن؛ $removed \leftarrow true$

`removed` را برگردان

می‌تواند به $O(n^2 d^2)$ کاهش داده شود.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سازگاری قوس فقط مقادیر ناسازگار را برمی دارد و با این کار فضای جستجو را هرس می کند.

مثال: $X \neq Y$ و $X \neq Z$ و $Y \neq Z$ با دامنه $\{1, 2\}$ دارای سازگاری قوس می باشد اما راه حل

نیست.

سازگاری قوس هدایتی ^۱

سازگاری قوس، تجدید نظر را تکرار می کند (تعداد دفعات وابسته به قوس ها و دامنه ها می باشد) ولی سازگاری قوس هدایتی، فقط یک بار بازنگری می کند و یک ترتیب متغیر را بازبینی می کند ($<$)؛ مسئله ای اراضی محدودیت در صورتی یک سازگاری قوس هدایتی است اگر هر قوس (x_i, X_j) که $x_i < X_j$ سازگاری قوس باشد. توجه کنید که سازگاری قوس، کلی تراز سازگاری قوس هدایتی است و برای درخت ها مناسب می باشد.

سازگاری مسیر ^۲

یک مسیر (X_1, X_m) یک مسیر سازگار است اگر برای هر جفت از مقادیر x_1 و x_m راضی کننده ای همه ای محدودیت های x_1 و x_m باشد؛ یک انتساب برای X_2, X_{m-1} که راضی کننده ای محدودیت های (X_i, X_{i+1}) باشد وجود دارد. مسئله ای اراضی محدودیت با مسیر سازگار است در صورتی که هر مسیر، سازگار باشد. در واقع، شما فقط سازگاری همه ای مسیرهای با طول ۲ را احتیاج دارید. سازگاری مسیر جفت های مقادیر را برمی دارد و محدودیت ها را صریح می سازد (به عنوان مثال، $Y < X$ و $X+1 < Z \Leftarrow Y < Z$

K -سازگاری ^۱

Directional Arc Consistency (DAC)^۱
Path Consistency (PC)^۲

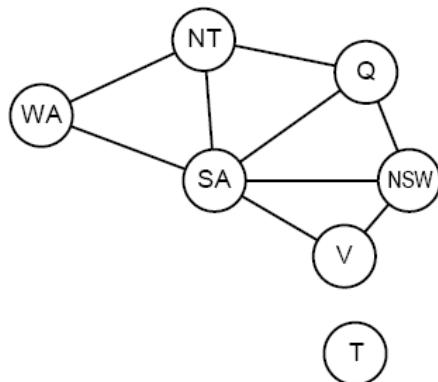
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- سازگاری ، آشکال سازگاری را تعمیم می دهد . مسئله ای ارضای محدودیت ، k -سازگار است اگر برای هر مجموعه i $k-1$ متغیری با انتساب سازگار و یک مقدار سازگار بتواند به هر k انتساب داده شود . در صورتی که هر متغیر با خودش سازگار باشد ، سازگار می باشد و دارای سازگاری قوس و سازگاری مسیر هم هست .



K - سازگار قوی - یک مسئله ای ارضای محدودیت k -سازگار قوی است در صورتی که k -سازگار و $k-1$ -سازگار و ... و ۱-سازگار باشد .

ساختم مسئله ^۲

تاسمانیا و مین لند^۳ زیر مسئله هایی مستقل هستند و به صورت اجزای متصل شده به گراف محدودیت قابل تعریف اند حال تصور کنید که هر زیر مسئله دارای متغیرهای C تا حد مجموع n باشد . در بدترین حالت ، هزینه ای راه حل برابر است با $n^{c.d^c}$ و به صورت خطی بر حسب مقدار n می

K-consistency^۱
problem structure^۲
mainland^۳

مترجم: سهراب جلوه گر

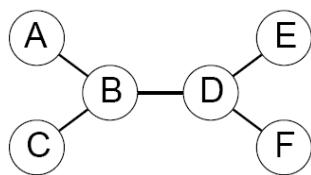
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

باشد. به عنوان مثال، برای $n = 80$, $d = 2$, $c = 20^{80}$ داریم: چهار بیلیون سال با سرعت ده میلیون گره بر ثانیه. 4.2^{20} چهاردهم ثانیه با سرعت ده میلیون گره بر ثانیه.

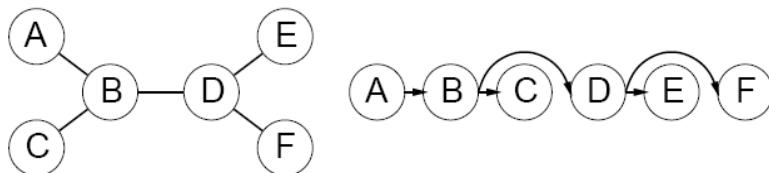
مسایل اراضی محدودیت درختی ساختیافته^۱



قضیه: در صورتی که گراف محدودیت هیچ حلقه ای نداشته باشد، مسئله ای اراضی محدودیت می تواند در زمان $O(nd^2)$ حل شود. این زمان را در جایی که در بدترین حالت، زمان برابر $O(d^n)$ است مقایسه نمایید. این خصوصیت همچنین برای استدلال های منطقی و احتمالاتی به کار برده می شود: یک مثال مهم، ارتباط میان محدودیت ها و پیچیدگی استدلال است.

الگوریتمی برای مسایل اراضی محدودیت درختی ساختیافته

۱. متغیری را به صورت ریشه انتخاب نمایید، متغیرها را از ریشه به برگ ها مرتب نمایید تا این که هر گره مقدم والد آن به صورت منظم در آید،



۲. برای j از n تا ۲ (به صورت کاهشی)، تابع $\text{RemoveInconsistent}(X_j)$ را به کار ببرید

Tree-structured CSPs^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

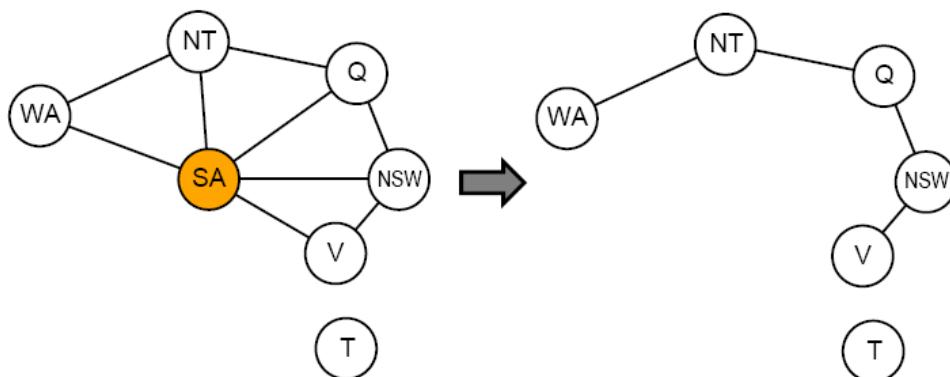


هوش مصنوعی

۳. برای j از ۱ تا n ، X_j را بدون تناقض با $Parent(X_j)$ متنسب نمایید.

مسایل ارضای محدودیت درختی ساختیافته‌ی تقریبی^۱

شاپسته سازی^۲: معرفی یک متغیر و هرس کردن دامنه‌ی همسایه‌های آن.



شاپسته سازی برش^۳: معرفی (در همه‌ی موارد) یک مجموعه از متغیرها تا این که گراف محدودیت، به یک درخت تبدیل شود. اگر اندازه‌ی برش C باشد، در نتیجه، زمان اجرای $O(d^c \cdot (n-C)d^2)$ را خواهیم داشت، برای C کوچک خیلی سریع است.

درخت تجزیه^۴

مسئله را به زیر مسئله‌های پیوسته تجزیه می‌نماید که زیر مسئله‌ها همان گره‌ها هستند. احتیاجات (ニヤزهای) ما عبارتند از: برای هر متغیر، لااقل یک گره باید داشته باشیم؛ متغیرهای پیوسته باید لااقل در یک

Nearly tree-structured CSPs^۱

Conditioning^۲

Cutset conditioning^۳

Tree decomposition^۴

مترجم: سه راب جلوه گر

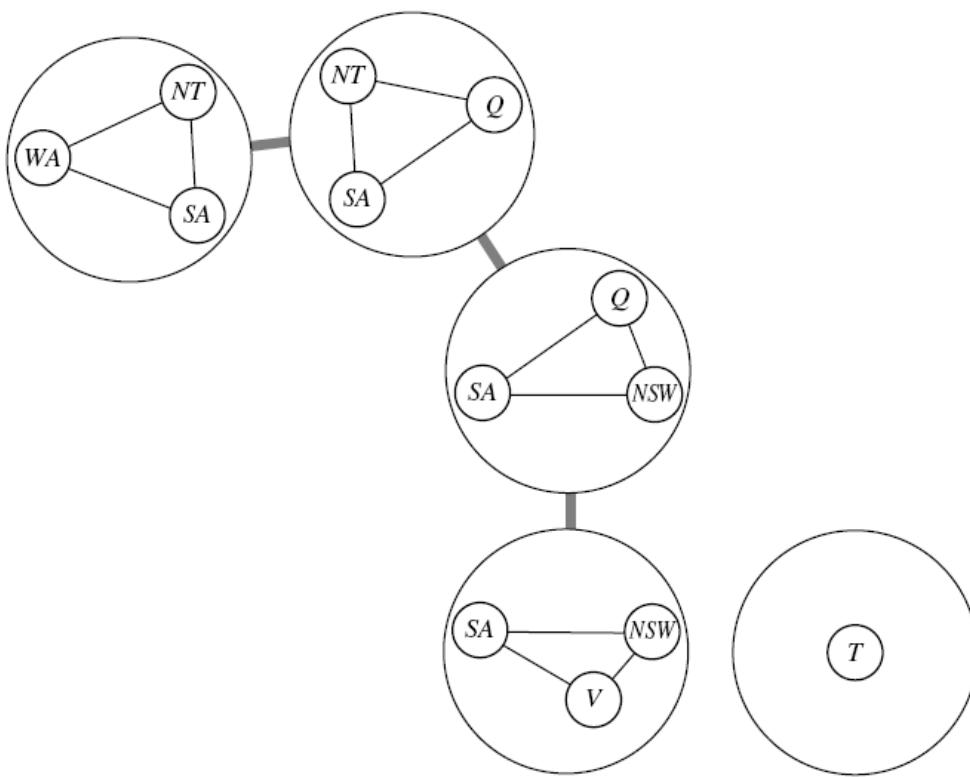
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گره با هم به نظر برسند و متغیری که به نظر می رسد در دو گره است باید در هر گره دیگر در مسیر وجود داشته باشد.

مثال درخت تجزیه



توجه کنید که محدودیت میان گره ها، سازگاری متغیرها را سبب می شود و تجزیه منحصر به فرد نمی باشد در ضمن زیر مسئله ها باید تا حد امکان کوچک باشند. زیرمسئله ها به صورت مستقل حل می شوند و سپس به وسیله ای متغیرهای مشترک به هم چسبانده^۱ می شوند. در صورتی که یک زیرمسئله دارای

¹ glued

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هیچ راه حلی نباشد آن گاه کل مسئله دارای هیچ راه حلی نمی باشد . در صورتی که اندازه بزرگ ترین مسئله W باشد ، آن گاه مسئله می تواند در $O(nd^w)$ حل شود .

الگوریتم های تکراری برای مسایل ارضای محدودیت - په نورده و شبیه سازی گرم و سرد کردن معمولی با حالت های کامل کار می باشد ، توجه کنید که همه مغایرها نسبت داده شده اند . برای به کاربردن با مسایل ارضای محدودیت ، باید حالت های مجاز با محدودیت های عملگرهای ناخوشایند ، دوباره به مقدار مغایرها نسبت داده شوند .

انتخاب مغایر : هر مغایر ناسازگار را به صورت تصادفی انتخاب نمایید .

ابتکار انتخاب مقدار با کم ترین ناسازگاری : مقداری را که از کم ترین محدودیت ها تخلف می کند را انتخاب نمایید . توجه نمایید ، در په نورده ، در $h(n)$ برابر با تعداد مجموع محدودیت های مختلف می باشد .

مثال : چهار وزیر

تصوّر نمایید که یک وزیر در هر ستون وجود دارد .

متغیرها : Q_1 و Q_2 و Q_3 و Q_4

دامنه ها : $D_i = \{1, 2, 3, 4\}$

محدودیت ها :

$Q_i \neq Q_j$ (نمی توانند در ردیف یکسان باشند)

$|Q_i - Q_j| \neq |i - j|$ (در یک قطر هم نمی توانند باشند)

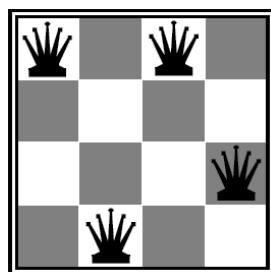
مترجم: سهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



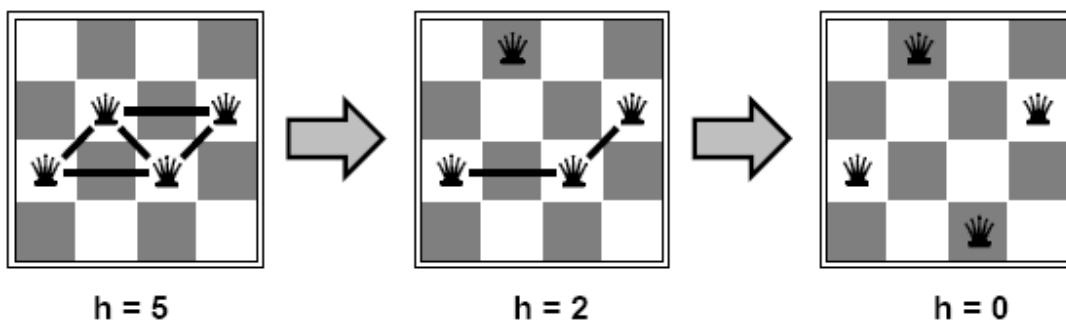
هوش مصنوعی

هر محدودیت را به مجموعه ای از مقادیر مجاز برای متغیر شان ترجمه نمایید :



به عنوان مثال ، برای (Q_1, Q_2) مقادیر عبارتند از (۳و۱) و (۴و۱) و (۴و۲) و (۱و۳) و (۱و۴) و (۲و۴)

حالت ها (وضعیت ها) : چهار وزیر در چهار ستون ($4^4 = 256$ حالت) | عملگرها : وزیر را در ستون حرکت دهید | تست هدف : هیچ حمله ای وجود نداشته باشد . | ارزیابی : $h(n) = \text{تعداد حملات}$



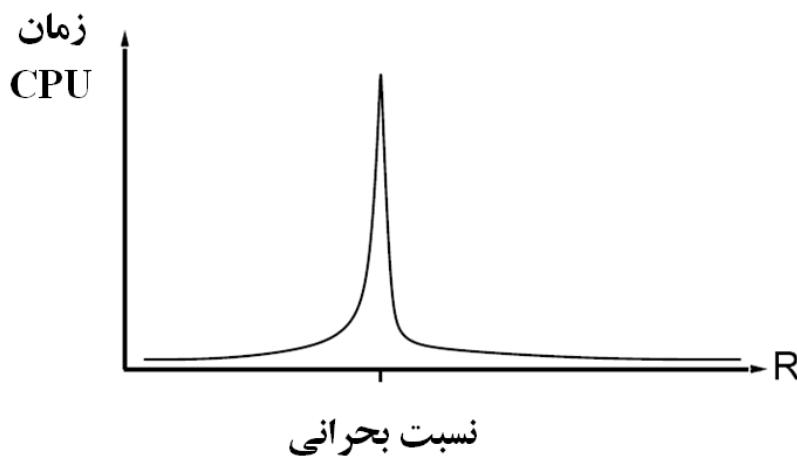
عمل با کمترین ناسازگاری - حالت اوّلیه ی تصادفی ارایه شده ، می تواند n - وزیر را در ثابت زمانی تقریبی $D(n)$ با احتمال زیاد حل کند ($n = 10,000,000$) . بیان های همانند می توانند برای هر مسئله ی اراضی محدودیت به صورت تصادفی صحیح باشند به جز در یک محدوده ی باریک از نسبت زیر : $R = \frac{\text{تعداد محدودیت ها}}{\text{تعداد متغیر ها}}$

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



محدودیت های دودویی

بیش تر تکنیک های گفته شده فقط با محدودیت های دودویی کار می کنند. تعدادی از مسائل به طور طبیعی به صورت محدودیت های n تایی^۱، رمزگذاری می شوند، به عنوان مثال: $X+Y=Z$. توجه نمایید که گره با سازگاری زوج، دودویی نمی باشد؛ به عنوان مثال $X \leq 12$ و محدودیت های یکانی به سادگی می توانند با کاهش دامنه به کار گرفته شوند:

$$D'_X = \{1,5\} \leftarrow X \leq 12 \text{ و } D_X = \{1,5,24\}$$

کاهش محدودیت های n تایی

هر مسئله ای ارضاي محدودیت می تواند به یک مسئله ای ارضاي محدودیت دودویی معادل تبدیل شود. برای این کار چند روش وجود دارد؛ در اینجا ما روش رمزگننده‌ی متغیر پنهان^۲ را بررسی می کنیم؛ در این روش محدودیت های n تایی با متغیرهای جدید ارایه می شوند؛ دامنه‌ی متغیرهای جدید،

¹n-ary

²hidden variable encoding

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چندتایی^۱ (n تایی) هستند؛ محدودیت های دودویی متغیرهای "اصلی" را به اجزای چندتایی ها بچسبانید.

مثال: $X+Y=Z$ ، با $X, Y, Z \in N$. در این صورت متغیر جدید U با دامنه ای

خواهد بود. از محدودیت $\pi_i(0,0)$ برای چسباندن مقدار $I^{\text{ام}}$

استفاده نمایید؛ به عنوان مثال:

$$\pi_2(U, Y) = \{(x_1, x_2, x_3), x_2\} \in N^3 \times N\}$$

سختی مسئله‌ی اراضی محدودیت

مسئله‌ی اراضی محدودیت، NP-کامل و سخت است، برای آن‌ها از الگوریتم‌های مکاشفه‌ای استفاده کنید (همان طوری که در درس مطرح شده است). در بدترین حالت، پیچیدگی به صورت نمایی می‌باشد، اما در مواردی دارای رفتار خوبی می‌باشد. الگوریتم‌های مکاشفه‌ای، کلاس‌هایی از مسایل نرم و رام شدنی را تعریف می‌نماید و دارای زمان چندجمله‌ای با محدودیت‌هایی در مسایل است. به عنوان مثال، مسایل با ساختار درختی.

برنامه نویسی براساس محدودیت

محدودیت‌ها، یک راه طبیعی برای بیان مسایل می‌باشند. مسئله‌ی اراضی محدودیت، با یک زبان برنامه نویسی حل می‌شود. برنامه نویسی با زبان‌های منطقی نظیر پرولوگ^۲ انجام می‌شود که دارای محیط اختصاصی می‌باشد (مثل وزارت^۳). پرولوگ به صورت‌های تجاری و رایگان ارایه شده است (به عنوان مثال SICStus یا GNU-Prolog). برای اطلاعات بیشتر نگاه کنید به راهنمای برخط برای برنامه

tuple^۱

Mozart^۴ نامی برای نسخه‌ی ۷.۵ سیستم عامل شرکت Apple برای کامپیوترهای MAC و PowerPC. نسخه‌های جدید دارای خصوصیات چند وظیفه‌ای، بهبود شیکه، پشتیانی چند رسانه‌ای بهتر و امکانات سیستم عامل‌های Dos و Windows (Babylon / Jensen's Technology Glossary) می‌باشند.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نویسی محدود با آدرس اینترنتی <http://kti.mff.cuni.cz/~Ebartak/constraints/> و آرشیو محدودیت ها با آدرس اینترنتی <http://www.cs.unh.edu/ccc/archive>.

خلاصه فصل

مسایل اراضی محدودیت، نوع خاصی از مسایل هستند که در آن ها حالت ها توسط مقدارهای یک مجموعه ای ثابت از متغیرها تعریف می شوند. و هدف آزمایش، توسط محدودیت هایی بر مقدار متغیرها تعریف می شود. پیماش معکوس، جستجوی اوّل - عمق با یک متغیر انتساب داده شده به هر گره می باشد. مرتب کردن متغیر و انتخاب ابتکاری مقدارها به طور بسیار مطلوبی به ما کمک می کند. بررسی مستقیم، از مقداردهی هایی که خرابی آینده را به وجود می آورند یا تضمین می کنند جلوگیری می کند. پخش محدود، کاری اضافی را برای مقدارهای محدود و کشف ناسازگاری ها، انجام می دهد. بیان مسئله به صورت مسئله ای اراضی محدودیت، تحلیل ساختار مسئله را اجازه می دهد. درخت ساختیافته ای مسایل اراضی محدودیت می تواند در زمانی که به صورت خطی می باشد، حل شود.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل هفتم

مسایل اراضی محدودیت و برنامه نویسی اراضی محدودیت^۱

Constraint Satisfaction Programming^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مسئله‌ی بهینه سازی محدودیت^۱

مسئله‌ی اراضی محدودیت استاندارد، مجموعه‌ای از متغیرهای مربوط به هم و یک مجموعه از محدودیت‌هاست. مسئله‌ی بهینه سازی محدودیت همان مسئله‌ی اراضی محدودیت استاندارد است به اضافه‌ی تابع هدف نسبت دهنده (نگاشت کننده)‌ی هر راه حل به یک مقدار عددی. راه حل این نوع مسایل، انتساب کامل محدودیت‌های ارضا کننده و بیشینه سازی یا کمینه سازی مسئله است. به عنوان مثال در رنگ آمیزی نقشه، هزینه، وابسته به رنگ‌های استفاده شده است و چاپ با چند رنگ دارای هزینه‌ی بیش تری می‌باشد.

الگوریتم‌های مسایل بهینه سازی محدودیت

الگوریتم ساده، همه‌ی راه حل‌ها را به حساب می‌آورد و از تمام روش‌هایی که تا کنون توضیح داده شده است استفاده می‌کند. الگوریتم در زمانی که تعداد زیادی راه حل وجود داشته باشد بد است (به عنوان مثال، n -وزیر). بهبودهای الگوریتم مسئله‌ی اراضی محدودیت برای اجتناب از جستجو در شاخه

Constraint Optimisation Problem (COP)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



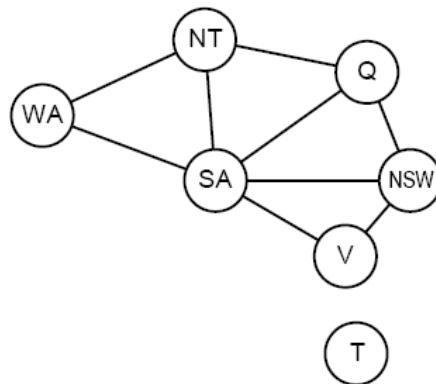
هوش مصنوعی

های بدون راه حل طراحی می شود. ایده‌ی کلیدی این است که شاخه‌ها را با زیر راه حل‌های بھینه هرس نمایید و از مکاشفه برای هرس فضای جستجو استفاده نمایید.

ابتکارها (مکاشفه‌ها)

تابع هدف، مقداردهی‌های کامل را ارزیابی می‌کند و تابع مکاشفه‌ای، انتساب‌های جزئی را ارزیابی می‌نماید و به مقادیر عددی، نسبت می‌دهد. در تخمین تابع هدف، به همه‌ی راه حل‌های توسعه دهنده‌ی یک انتساب جزئی نمایید باید بهترین مقدار این راه حل را تخمین بزنند. توجه کنید که مقداردهی‌های با مقادیر ابتکاری (مکاشفه‌ای) بد را توسعه ندهید. هرس اضافی و هرس استاندارد مسئله‌ی اراضی محدودیت هم می‌تواند به کار گرفته شود.

مسئله‌ی اراضی محدودیت به صورت برنامه نویسی منطقی



rgb(red).

rgb(green).

rgb(blue).

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

colourable([WA,SA,NT,Q,NSW,V,T]) :-

rgb(WA),rgb(SA),rgb(NT),rgb(Q),rgb(NSW),rgb(V),rgb(T),

$WA \equiv NT, NT \equiv Q, Q \equiv NSW, NSW \equiv V, SA \equiv WA, SA \equiv NT, SA \equiv Q, SA \equiv NSW, SA \equiv V.$

هر مسئله‌ی ارضای محدودیت با دامنه‌ی محدود می‌تواند به صورت یک شرط صریح تکی با برخی واقعیّات اضافه بیان شود. هر اتصال^۱، یک محدودیت در متغیرهای شامل شده می‌باشد. بنابراین، مسئله‌ی ارضای محدودیت را به زبان پرولوگ^۲ بنویسید و اجازه بدھید که مفسر^۳ آن را حل کند. مسئله‌این است که روش اول – عمق پرولوگ در بیشتر موارد، بهترین نمی‌باشد.

برنامه نویسی منطقی محدود

الگوریتم‌هایی از الگوریتم‌های مسئله‌ی ارضای محدودیت به پرولوگ استاندارد اضافه شده‌اند که معمولاً دارای نام گستره یا پسوند یا دامنه‌ی CLP می‌باشند (به صورت فایل‌های CLP در مورد عقلانیّات و CLP(FD) در دامنه‌های محدود؛ (B) در مورد بولین‌ها و (Q,R) در مورد عقلانیّات و واقعیّات می‌باشند. با استفاده از مکانیزم محلی^۴ پرولوگ برای توسعه‌ی زبان پیاده‌سازی شده‌اند.

مثال : مسئله‌ی رمزی : SEND+MORE=MONEY

:- use_module(library('clp/bounds')).

cryptarithmic(S,E,N,D,M,O,R,Y) :-

conjunct^۱

interpreter^۲

۳ مترجم

native^۴

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Digits = [S,E,N,D,M,O,R,Y],

Carries = [C1,C2,C3,C4],

Digits in 0..9,

Carries in 0..1,

M # = C4,

O + 10 * C4 # = M + S + C3,

N + 10 * C3 # = O + E + C2,

E + 10 * C2 # = R + N + C1,

Y + 10 * C1 # = E + D,

M # >= 1,

S # >= 1,

all_different(Digits),

label(Digits).

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



۱

فصل هشتم

تئوری بازی ها^۱



^۱ - تصویرهای بالا متعلق به گری کاسپاروف (Gary Kasparov)، قهرمان شطرنج روسی است.

Game playing^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

رئوس مطالب

بازی ها •

بازی کامل ^۱ •

- تصمیم گیری های مینیماکس ^۲

- هرس ^۳

محدودیت منابع و ارزیابی تقریبی ^۴ •

بازی های تصادفی ^۵ •

بازی های با اطلاعات غیر کامل ^۶ •

Perfect play ^۱

Minimax decisions ^۲

pruning ^۳

Resource limits and approximate evaluation ^۴

Games of chance ^۵

Games of imperfect information ^۶

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

محیط های چند عاملی

الگوریتم های جستجویی که ما تا حالا بررسی کرده ایم ، دارای یک محیط تک عاملی بودند . و عامل در حال کار در یک جهان بی اثر^۱ می باشد به عبارت دیگر ، دیگر عامل ها بر روی آن اثری نداشتند ؛ اما در مواردی که یک عامل باید دیگر عامل ها را به حساب بیاورد چه طور ؟ مثل محیط های رقابتی نظیر بازی های شطرنج ، چکرز ، go و ... ؛ یا حراجی ها^۲ ، فروشگاه های برخط و محیط های شراکتی^۳ مثل عامل های یک تیم فوتبال .

تئوری بازی ها – تئوری بازی ها ، شاخه ای از علم ریاضیات یا علم اقتصاد می باشد که به اثرات متقابل میان چند عامل می پردازد و روش هایی را برای تشخیص رفتار بهینه ، تعیین می نماید .

بازی های با اطلاعات کامل : عامل ها به همه ای دانش در مورد محیط ، دسترسی دارند . ما این محیط را محیط کاملاً قابل مشاهده می نامیم .

بازی های با اطلاعات ناقص : عامل باید در مورد جهان یا حریفش ، استنتاج نماید . ما /ین محیط را محیط قابل مشاهده به صورت جزئی می نامیم . مثال ها ، بازی های شانسی ، بیش تر برهمن کنش های دنیای واقعی و برخی از حراجی ها می باشند .

مفروضات تئوری بازی ها :

عقلانیت کامل – عملکردها ، همیشه کار درست را انجام خواهند داد .

neutral^۱

auctions^۲

cooperative^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

محاسبه‌ی نامحدود – عامل‌ها همیشه قادر به تشخیص این هستند که چه کاری برای انجام

دادن، درست می‌باشد.

همان طور که دیده ایم، این مفروضات در برخی از موارد، مصدق ندارند (درست نمی‌باشند) . به هر حال، ما هنوز از برخی از ایده‌های تئوری بازی‌ها برای کمک در مورد تصمیم‌گیری در مورد این که چه کاری برای انجام، درست می‌باشد، استفاده می‌نماییم. بنابراین، یک عامل چگونه در یک محیط چند عاملی در مورد این که چه کار باید بکند تصمیم‌گیری نماید؟

روش‌های بھینه

بیایید با ساده ترین نوع بازی‌ها شروع کنیم؛ مثل، بازی‌های دو نفره، بازی‌های با اطلاعات کامل و بازی‌های رقابتی به صورت کامل – که در آن‌ها یکی می‌برد و دیگری می‌بازد. ما می‌توانیم این بازی‌ها را به صورت یک مسئله‌ی جستجو تعریف نماییم. تئوری بازی‌ها یکی از قدیمی ترین مباحثی است که در هوش مصنوعی راجع به آن زیاد مطالعه شده است. دلیل این مطلب آن است که: اول، افراد بازی‌ها را دوست دارند! و خوب می‌توانند با بازی‌ها کار کنند. دوم، اغلب، بازی‌ها به صورت یک نماینده‌ی هوش دیده می‌شوند. سوم، بازی‌ها دارای یک توضیح واضح از محیط می‌باشند، ولی در آن‌ها فضاهای حالت، خیلی بزرگ و پیچیده‌اند، بنابراین برخی اوقات اطلاعات اتفاقی و ناکار آمد می‌باشند و این باعث به وجود آمدن چالش در مسئله‌می‌شود. وقتی که هوش مصنوعی خوب کار می‌کند بازی‌ها واضح و صریح می‌باشند. بازی‌ها برای هوش مصنوعی مانند جایزه‌ای بزرگ برای مسابقه‌ی طراحی اتومبیل هستند.

مسایل جستجوی بازی‌ها – در بازی‌ها حرکت حریف^۱، غیرقابل پیش‌بینی می‌باشد در نتیجه راه حل، یک روش مشخص کننده‌ی یک حرکت، برای هر جواب ممکن حریف است.

opposite^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

محدودیت های زمانی موجود برای پیدا کردن هدف ، نا مطلوب یا ناخوشایند هستند ، در نتیجه ، باید تخمین زده شوند .

تاریخچه : کامپیوتر به موارد ممکن در بازی توجه می کند (Babbage, 1846) الگوریتمی برای بازی کامل (Zermelo, 1912; Von Neumann, 1944) | وسعت محدود، ارزیابی تقریبی (Zuse , 1945 ; Wiener , 1948 ; Shannon , 1950) | برنامه ای شطرنج اوّلیه Samuel , 1951) | آموزش ماشینی^۱ برای بهبود بخشیدن به درستی ارزیابی (Turing , 1952- 57) (McCarthy , 1956) | بازیینی یا هرس برای اجازه دادن به جستجوی عمیق تر

انواع بازی ها

شانسی	قطعی	
تخنه نرد ^۴	شطرنج ^۲ ، بازی چکرز ^۳ ، go	اطلاعات کامل
پل ^۶ ، پوکر ^۷	تیک تا تو کور ^۵	اطلاعات ناقص

machine learning ^۱	chess ^۲
checkers ^۳	backgammon ^۴
blind tictactoe ^۵	bridge ^۶
poker ^۷	

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

شطرنج : بازی ای است که بر روی یک صفحه انجام می شود و برای دو بازیگر است که شانزده مهره را با توجه به قوانینی معین حرکت می دهند. هدف مات کردن شاه طرف مقابل می باشد.

بازی چکوز : صفحه‌ی بازی چکوز برای دو نفر می باشد که هر نفر دارای دوازده مهره می باشد؛ هدف پریدن و دستگیر نمودن مهره های حریف می باشد.

بازی go : بازی ای بر روی صفحه برای دو بازیگر می باشد که شمارنده هایی را روی یک شبکه قرار می دهند، هدف محاصره کردن و سپس دستگیر کردن شمارنده های حریف می باشد.^۱

تیک - تاک - تو کور : بازی تیک - تاک - تو دارای دو بازیگر می باشد. برای هر دو بازیگر هدف این است که اوئین کسی باشند که سه شیء همانند را در یک ردیف، ستون یا قطر قرار می دهند. صفحه‌ی این بازی دارای یک شبکه‌ی سه در سه می باشد. بنابراین دارای نه خانه می باشد.^۲

تحته فرد : بازی ای بر روی صفحه است و دو بازیگر دارد. هر بازیگر دارای پانزده مهره می باشد که آن ها را در بیست و چهار خانه‌ی مثلثی شکل با توجه به عدد ظاهر شده روی دو تاس حرکت می دهد.^۳

پل : یک بازی کارتی حقه‌ای برای چهار نفر می باشد که این چهار نفر به صورت دو گروه دو نفری با هم بازی می کنند و در هر طرف هر گروه مقابل هم می نشینند. بازی پل دارای دو مرحله می باشد: پیشنهاد و بازی.^۴

پوکر : بازی ای کارتی می باشد، محبوب ترین بازی از یک دسته از بازی ها به نام بازی های همچشمی می باشد که در آن بازیگران با کارت هایی کاملاً پنهان یا اندکی پنهان روی یک چیزی شرط

wordnet.princeton.edu/perl/webwn^۱

<http://www.seeingwithsound.com/tictactoe.htm>^۲

wordnet.princeton.edu/perl/webwn^۳

en.wikipedia.org/wiki/Bridge_game^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



بندی می کنند، سپس چیزی که شرط بندی روی آن انجام شده است به بازیگران باقی مانده ای که دارای بهترین ترکیب کارت ها هستند جایزه داده می شود.^۱ در زیر تصویر صفحه‌ی چند بازی ای که هم اکنون معرفی نمودیم مشاهده می نمایید:

نام بازی	چکرز	go	تیک - تاک - تو کور	تخته نرد
تصویر صفحه				

تئوری بازی به صورت جستجو - یک بازی دو نفره با اطلاعات کامل را در نظر بگیرید؛ آیا می توانیم این بازی را به صورت مسایل جستجو فرمول بندی نماییم؟؛ وضعیت صفحه‌ی بازی، وضعیت جستجوی مسئله می باشد؛ حرکت های مجاز، عملگرها هستند و وضعیت های پایانی، وضعیت هایی هستند که در آن ها بازی را برد ایم یا بازنده شده ایم یا بازی بی نتیجه مانده است. ما می توانیم به بردن عدد $+1$ ، به باختن عدد -1 و به بازی بی نتیجه مانده عدد 0 را نسبت دهیم. ما می خواهیم یک روش (یک راه برای انتخاب حرکت ها) که بازی را می برد پیدا نماییم.

در این مورد، یک حریف وجود دارد. که بداندیش است؟ یعنی، چیزهای خوب را برای خود می خواهد و چیز های بد را برای شما می خواهد. ما باید تصمیم گیری حریفمان را شبیه سازی نماییم. توجه

^۱en.wikipedia.org/wiki/Poker

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

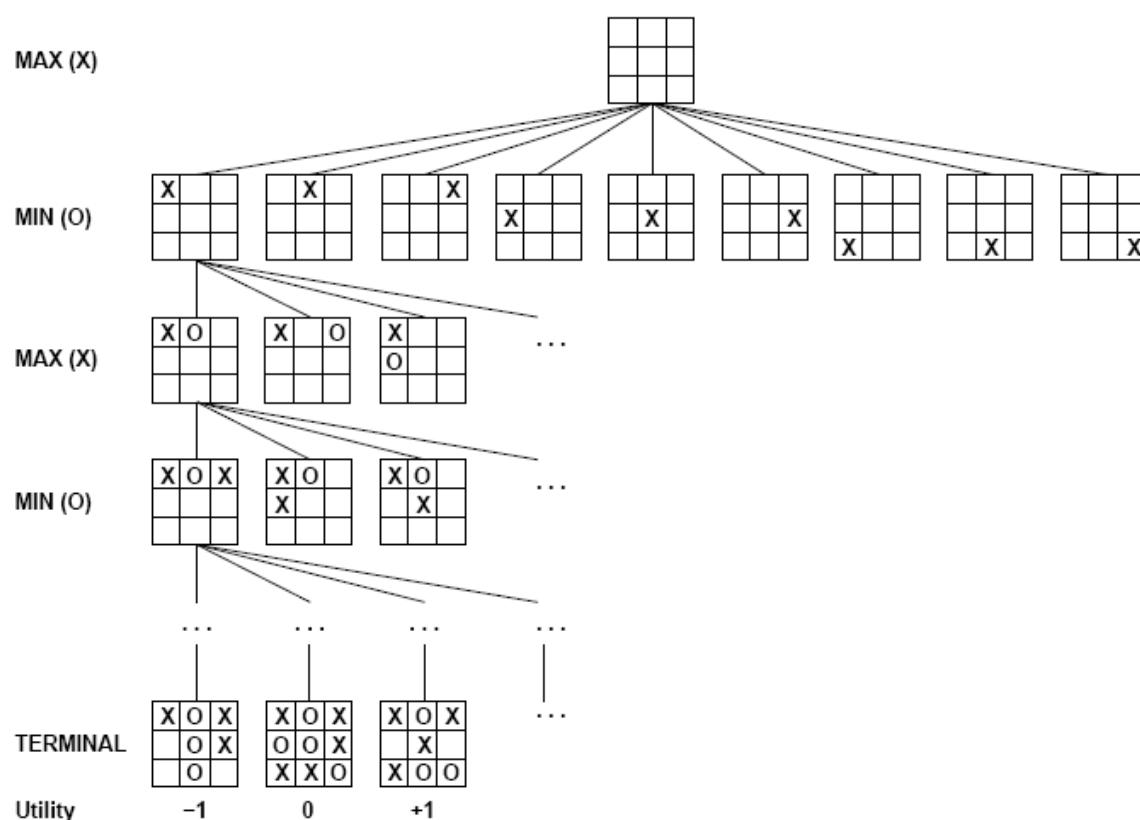


هوش مصنوعی

کنید که ، یک بازیگر بیشینه وجود دارد که بیش ترین نفع را برای خودش می خواهد و یک بازیگر کمینه وجود دارد که کم ترین نفع را برای خودش می خواهد .

درخت بازی - در حالت کلی ، ما می توانیم درخت بازی را برای هر بازی به صورت زیر رسم

نماییم .



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

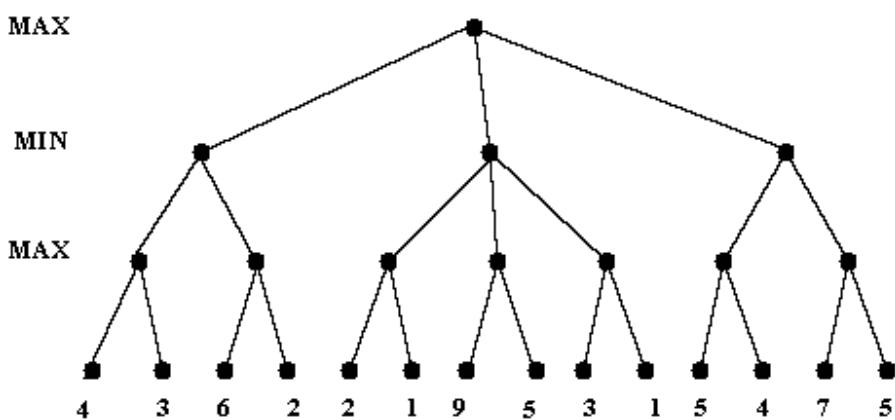


هوش مصنوعی

مینیماکس^۱ - MAX و MIN، دو بازیگر هستند که MAX می خواهد بازی را ببرد و MIN هم می خواهد بازی را ببرد و در واقع MIN و MAX، رقیب هم هستند. و هر دو بهترین بازی ممکن را انجام می دهند.

بازی کامل قطعی^۲ ، بازی های با اطلاعات کامل می باشند.

مثال زیر گویای این روش است:



داریم :

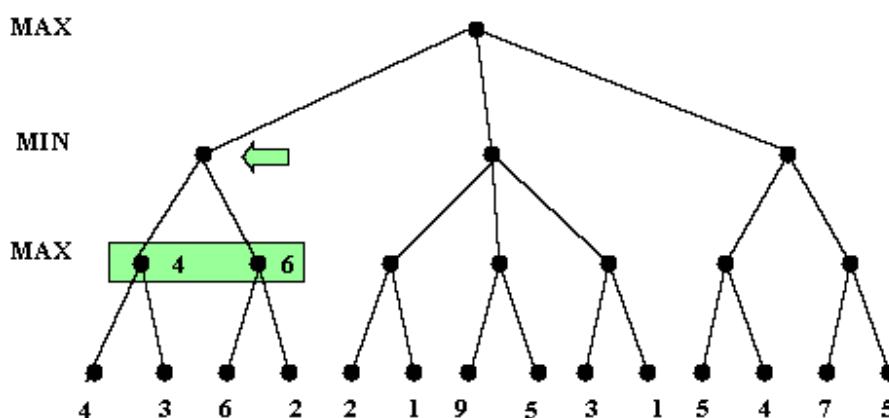
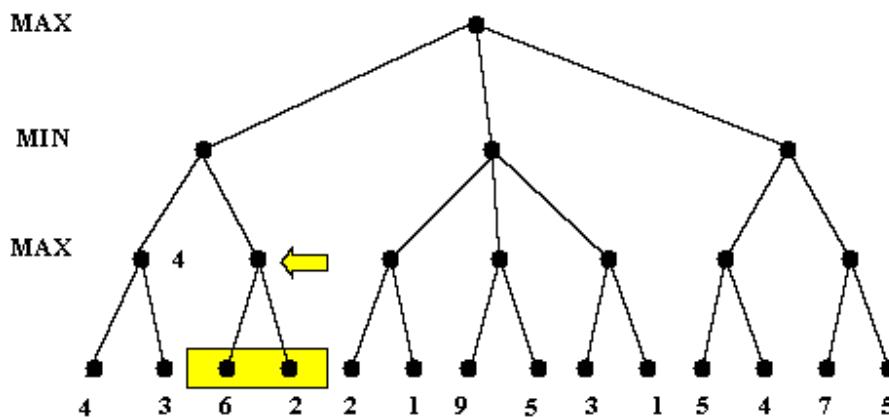
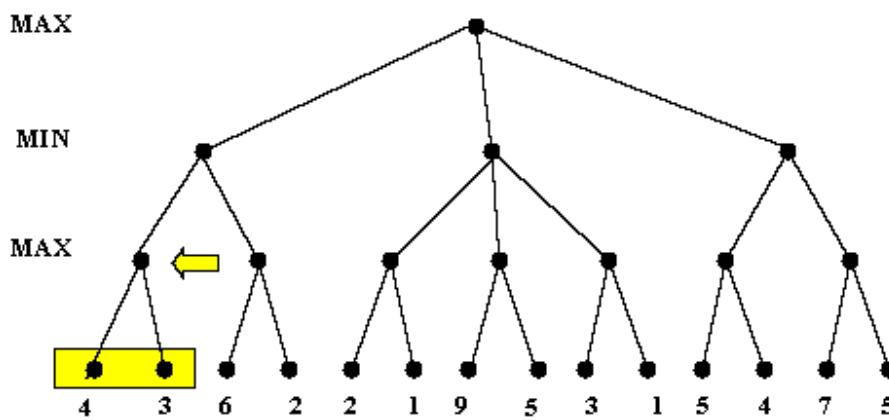
[www.informatics.susx.ac.uk/books/computers-and-thought/gloss/node1.html^۱](http://www.informatics.susx.ac.uk/books/computers-and-thought/gloss/node1.html)

^۱ Deterministic در کامپیوتر، نتیجه‌ی فرایندی که به موقعیت‌های اولیه ورودی‌ها بستگی دارد

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



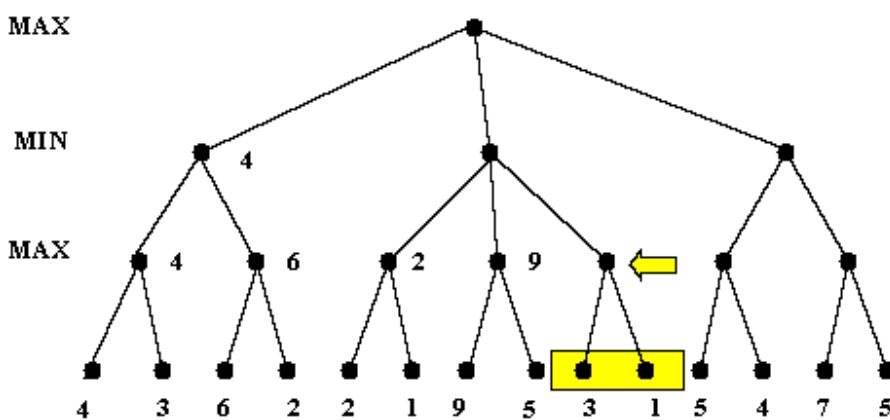
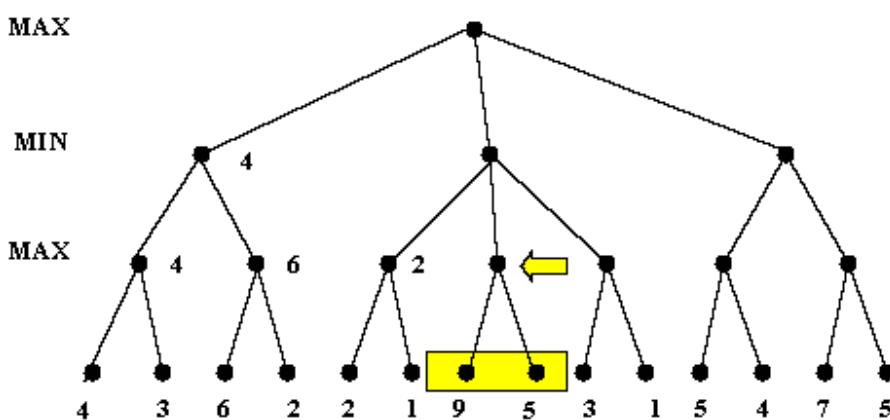
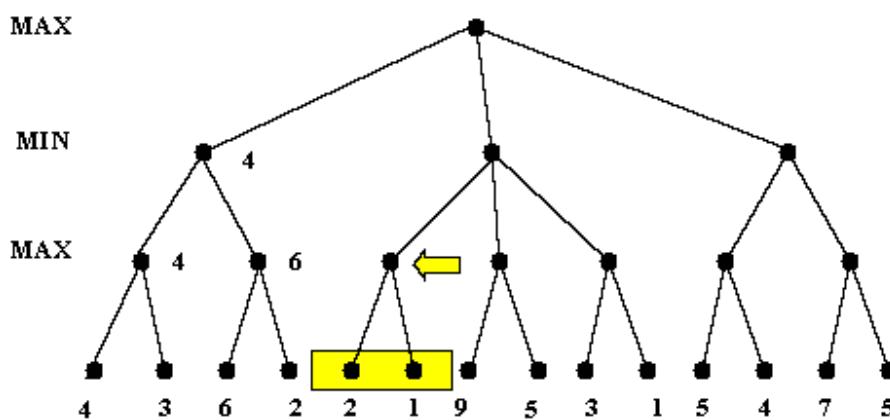
هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



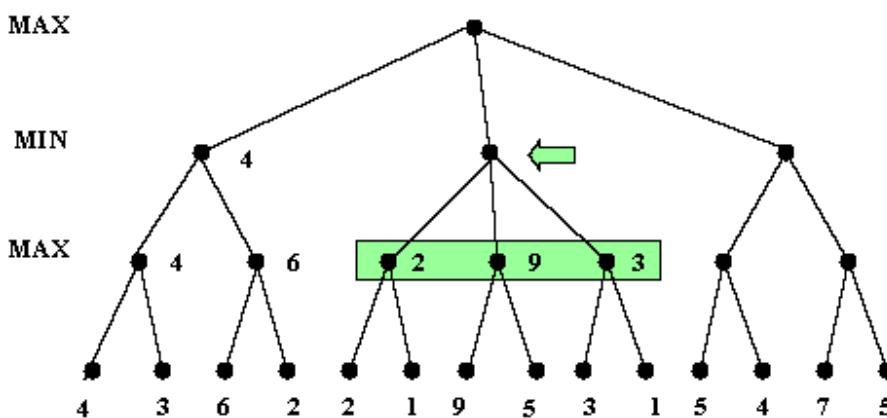
هوش مصنوعی



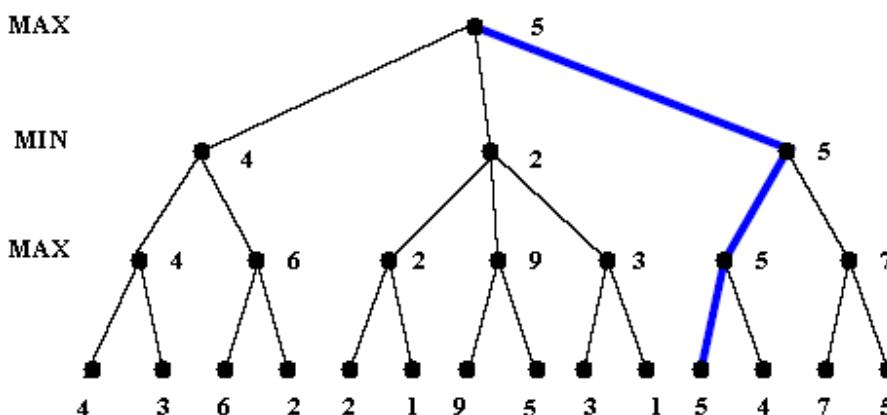
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



و به همین ترتیب موارد دیگر را هم به دست می آوریم تا در نهایت به شکل زیر برسیم :



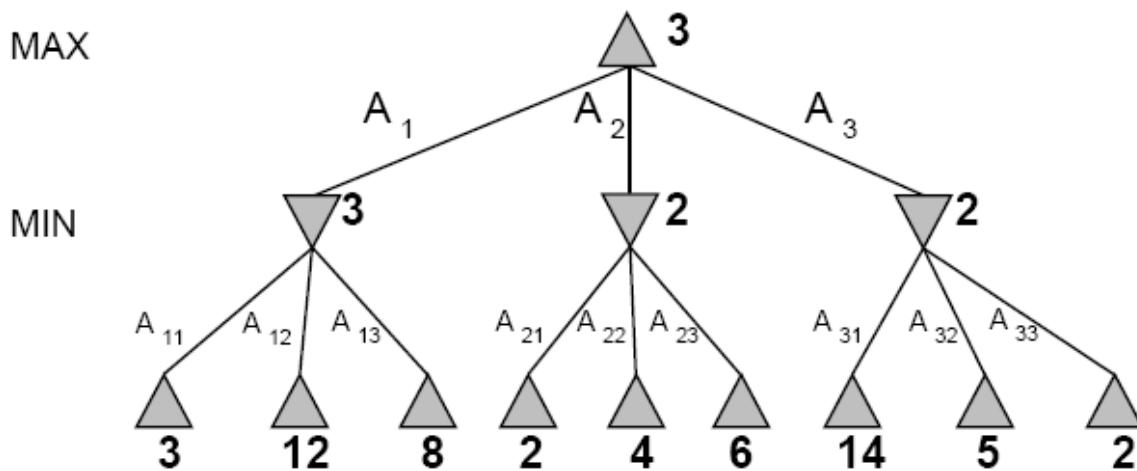
پس جواب این مثال عدد ۵ است.

به عنوان مثال دیگر ، برای بازی دو نفره داریم :

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



که در نتیجه حاصل برابر با ۳ خواهد بود.

الگوریتم مینیماکس

تابع $\text{Minimax-Decision(state)}$ یک عملکرد را بر می گرداند

ورودی : state ، که حالت جاری بازی می باشد

ای را که در Actions(state) بیشینه کننده ای $\text{Min-Value(Result(a, state))}$ می باشد را
برگردان

تابع Max-Value(state) یک مقدار مفید را بر می گرداند

در صورتی که تابع $\text{Terminal-Test(state)}$ دارای مقدار بازگشتی درست می باشد (درست) را
برگردان

$$v \leftarrow -\infty$$

برای a, s در تابع $v \leftarrow \text{Max}(v, \text{Min-Value}(s)), \text{Successor}(state)$

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مقدار v را بُرگردان

تابع Min-Value(state) یک مقدار مجاز را برمی گرداند

در صورتی که Utility(state) درست است، $\text{Terminal-Test(state)}$ را بُرگردان

$v \leftarrow \infty$

برای a, s درون Successors(state) ، $v \leftarrow \text{Min}(v, \text{Max-Value}(s))$

v را بُرگردان

خصوصیات مینیماکس

کامل بودن ?? فقط در صورتی که درخت محدود باشد، کامل است (شطرنج قوانین معینی برای این کار دارد) .

بهینه بودن ?? بله

پیچیدگی زمانی ?? $O(b^m)$

پیچیدگی فضای ?? $O(bm)$ (مثل جستجوی اوّل عمق)

برای شطرنج، $m \approx 100, b \approx 35$ ، (برای بازی های مستدل) \leftarrow راه حل دقیق کاملاً اجرا نشدنی است .

اما آیا ما احتیاج داریم که هر مسیر را پیدا نماییم ؟

$\alpha - \beta$ هرس

مترجم: سهراب جلوه گر

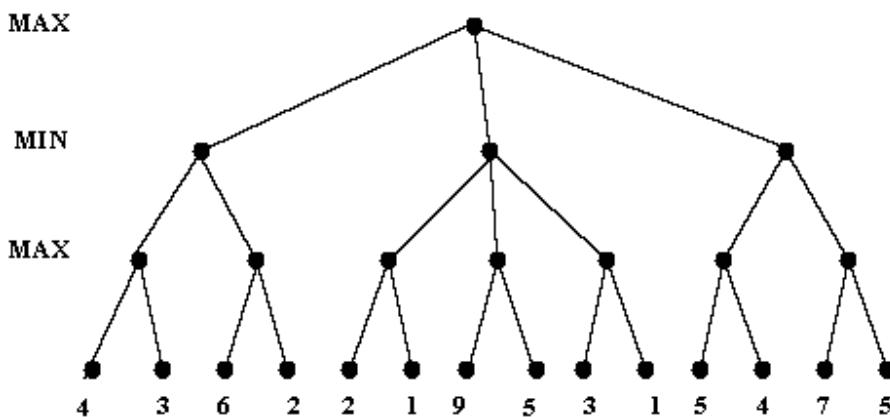
ویرایش دوم، بهار ۱۳۸۸



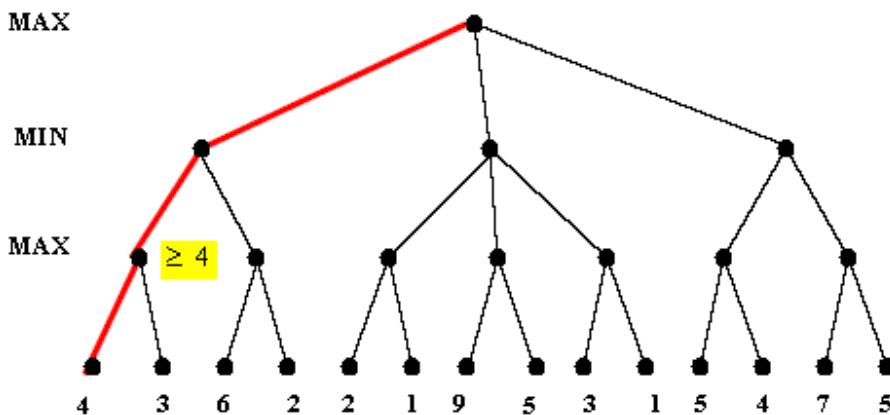
هوش مصنوعی

روشی استاندارد برای بازی های قطعی و با اطلاعات کامل می باشد؛ در این روش ، شاخه ای که هیچ وقت مورد استفاده قرار نمی گیرد را از درخت جستجو حذف می کنیم . توجه کنید که این شاخه ها هیچ وقت توسط یک بازیکن عاقل مورد استفاده قرار نمی گیرند ، در ضمن ، توسط بازیکن حریف هم مورد استفاده قرار نمی گیرند .

مثال هرس $\alpha - \beta$: به درخت زیر توجه کنید .



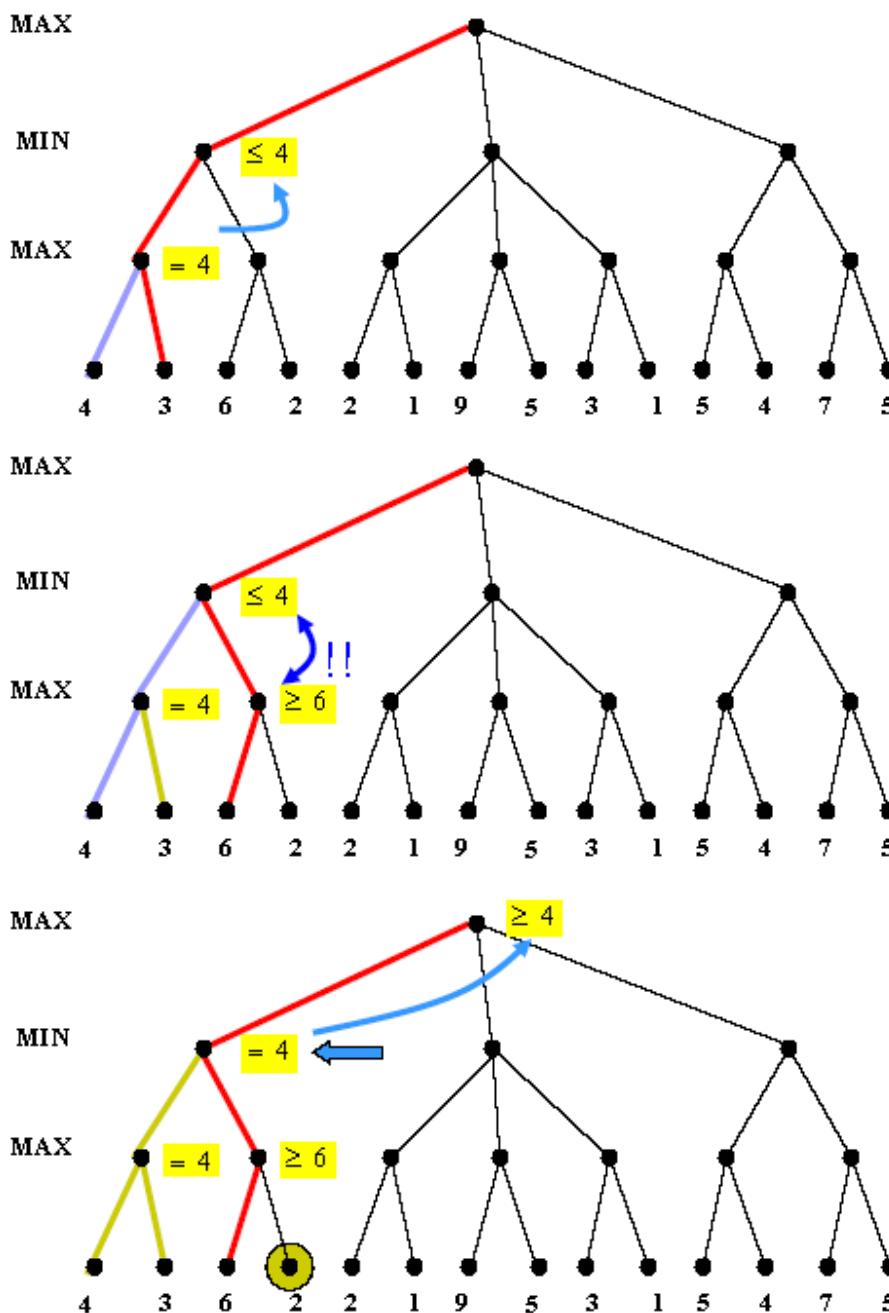
داریم (توجه کنید که گره های به شکل هیچوقت مورد استفاده قرار نمی گیرند) :



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



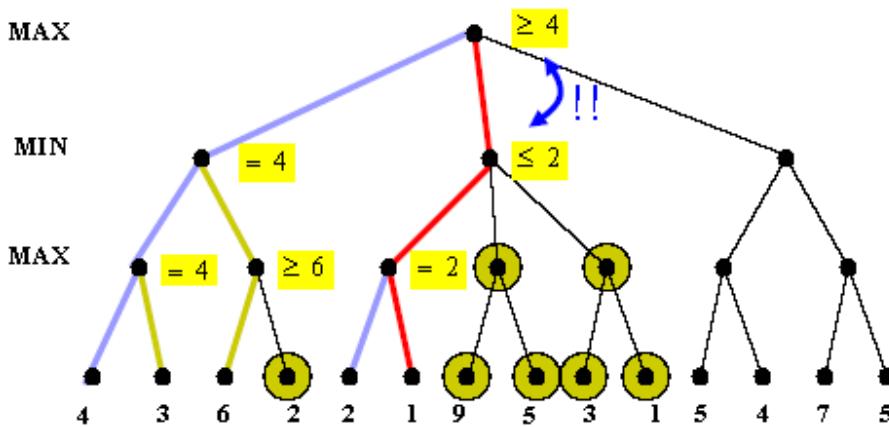
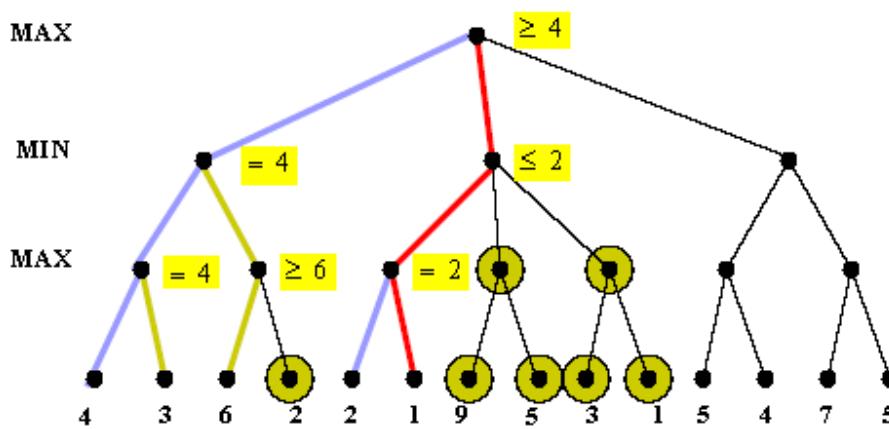
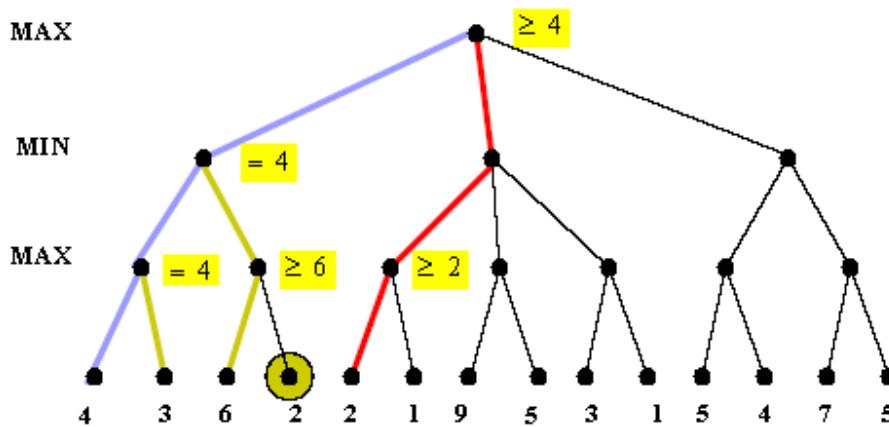
هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



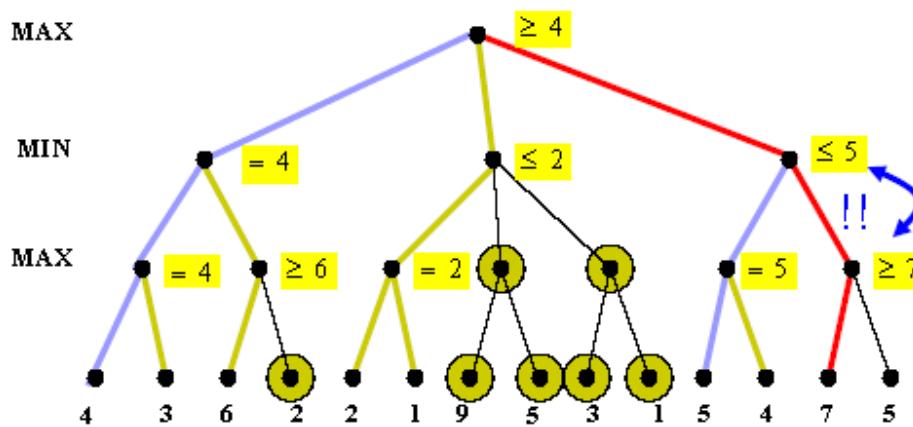
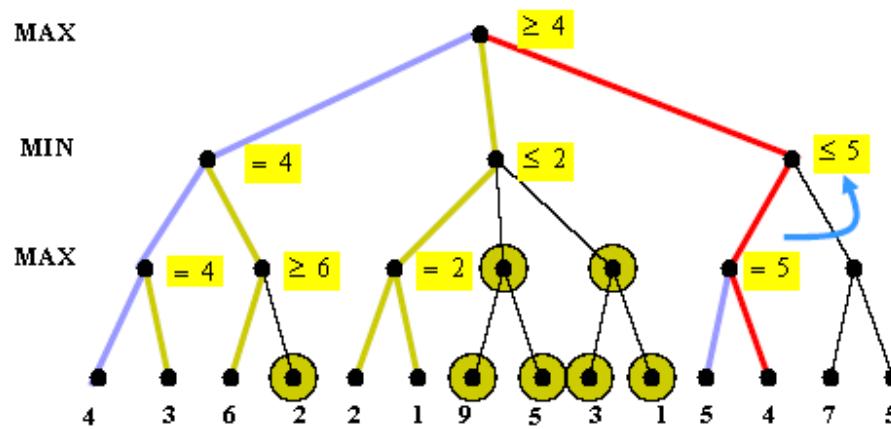
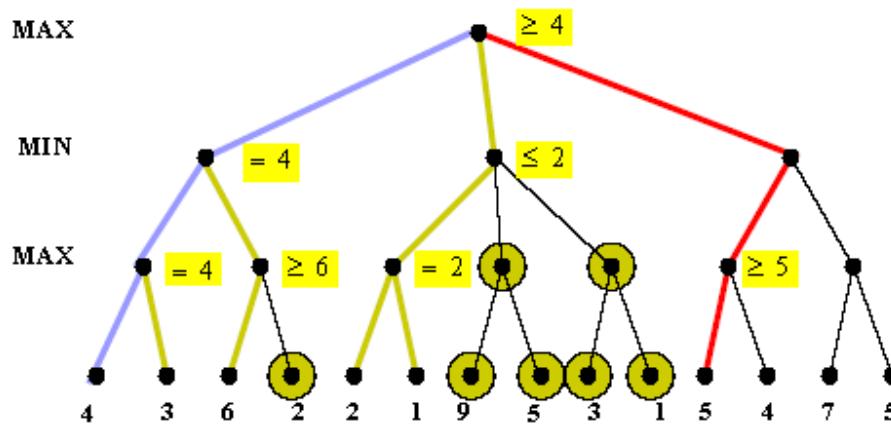
هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



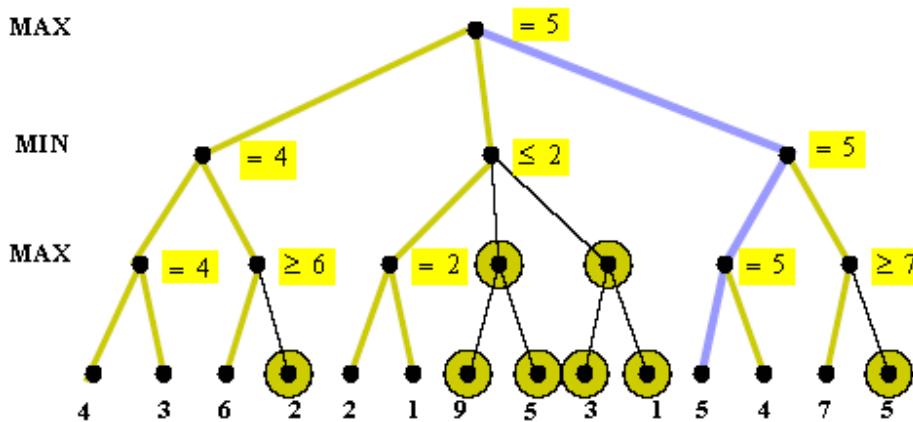
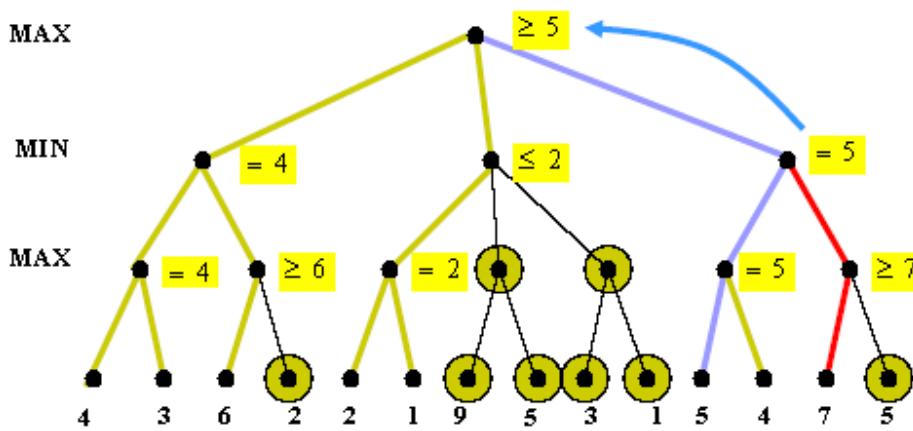
هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



α - β الگوریتم

تابع $\text{Alpha-Beta-Decision(state)}$ یک عمل را برابر می گرداند

ای را که در Actions(state) بیشینه کننده $\text{Min-Value}(a, \text{state})$ می باشد را برابر گردان

تابع $\text{Max-Value(state, } \alpha, \beta)$ یک مقدار مجاز را برابر می گرداند

ورودی ها ، state که حالت جاری در بازی می باشد

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

α مقدار بهترین برای MAX در طول مسیر به state می باشد .

β مقدار بهترین برای MIN در طول مسیر به state می باشد .

در صورتی که Utility(state) درست می باشد ، Terminal-Test(state) را برابر گردان

$$V \leftarrow -\infty$$

برای a, s در تابع Successors(state) کارهای زیر را انجام بده

$$v \leftarrow \text{Max}(v, \text{Min-Value}(s, \alpha, \beta))$$

در صورتی که $v \geq \beta$ باشد ، v را برابر گردان

$$\alpha \leftarrow \text{Max}(\alpha, v)$$

انتهای حلقه

$$v \text{ را برابر گردان}$$

تابع Min-Value(state, α, β) یک مقدار مجاز را برابر گرداند

شبیه مقدار Max-Value ، اما با قوانین رزرو شده α و β

خصوصیات α - β - بازیبینی یا هرس بر روی نتیجه ی نهایی اثری ندارد . حرکت خوب و منظم اثر هرس را بهبود می بخشد . با " منظم کننده ی کامل " ، پیچیدگی زمانی برابر با $O(b^{m/2})$ است ، درنتیجه عمق جستجو را مضاعف می کند . بنابراین ، به سادگی می تواند به عمق هشت برسد و بازی شطرنج خوبی را ارایه نماید . متأسفانه 35^{50} هنوز غیر ممکن است !

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

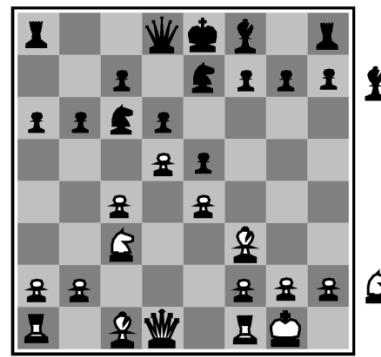
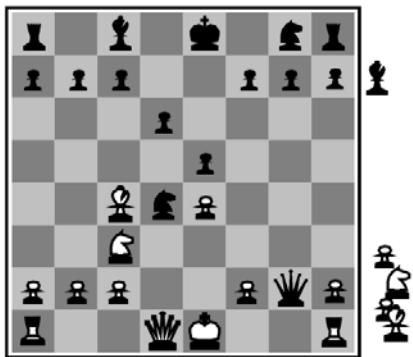


هوش مصنوعی

محدودیت منابع - فرض کنید ما صد ثانیه زمان داریم و می توانیم 10^4 گره در ثانیه را ملاقات کنیم درنتیجه در صد ثانیه 10^6 گره را می توانیم ملاقات نماییم.

توابع ارزیابی - یک تابع ارزیابی ، تابعی است که خوبی یک وضعیت را اندازه گیری می نماید (به عنوان مثال ، شانس برنده شدن در آن وضعیت را اندازه می گیرد) و این تابع می تواند توسط طراح ارایه شود یا با آزمایش به دست آید . اگر ترکیبات صفحه بتواند به صورت مستقل ارزیابی شود آن گاه یک انتخاب خوب یک تابع توزیع شده ی خطی می باشد :

انتخاب خوب یک صفحه s که $w_1f_1(s) + w_2f_2(s) + \dots + w_nf_n(s)$ نوع مهره بر روی صفحه ی بازی می باشد (مثلا فیل) و w_i ارزش مهره ها (مثلا یک برای سرباز و سه برای فیل) . توجه کنید که برنامه های پیشرفتی از ترکیبات غیرخطی هم استفاده می نمایند ، در ضمن ، ترکیبات و توزیع ها جزیی از قوانین شطرنج نمی باشند .



با حرکت سیاه ، وضعیت سفید بهتر می شود

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بازی های قطعی در عمل^۱

- بازی چکر : چینوک^۲ به چهل سال قهرمانی قهرمان جهان ماریون تیزلی^۳ در سال ۱۹۹۴ خاتمه داد. از پایگاه داده‌ی مشخص کننده‌ی پایان بازی شطرنج برای تعریف تمام بازی و برای وضعیت های شامل هشت یا کم تر مهره در صفحه استفاده شد، یک مجموعه از ۲۴۷ و ۴۰۱ و ۷۴۸ و ۴۴۳ حالت. Deep Blue قهرمان شطرنج جهان، گری کاسپاروف^۴ را در شش بازی در سال ۱۹۹۷ شکست داد. Deep Blue، دویست میلیون حالت را در هر ثانیه جستجو می‌کرد و از یک ارزیابی خیلی ماهرانه استفاده می‌کرد و از متدهای فاش نشده برای توسعه دادن تعدادی از خطوط جستجو تا ۴۰ تا استفاده می‌نمود.

اتلو^۵ : قهرمانانی که خیلی خوب هستند نمی‌پذیرند که با کامپیوتر رقابت نمایند.

Go : قهرمانان نمی‌پذیرند که با کامپیوتر رقابت کنند. در این بازی، $b > 300$ ، بنابراین بیشتر برنامه‌ها از الگوی براساس دانش برای پیشنهاد حرکت‌های ممکن استفاده می‌نمایند.

بازی های غیرقطعی در حالت کلی

در بازی های غیرقطعی مثل تخته نرد که تصویری از صفحه‌ی این بازی را در شکل زیر مشاهده می‌کنید، شанс به وسیله‌ی تاس^۶ و برهمن زدن کارت^۷ معرفی می‌شود.

deterministic games in practice^۱

Chinook^۲

Marion Tinsley^۳

Gary Kasparov^۴

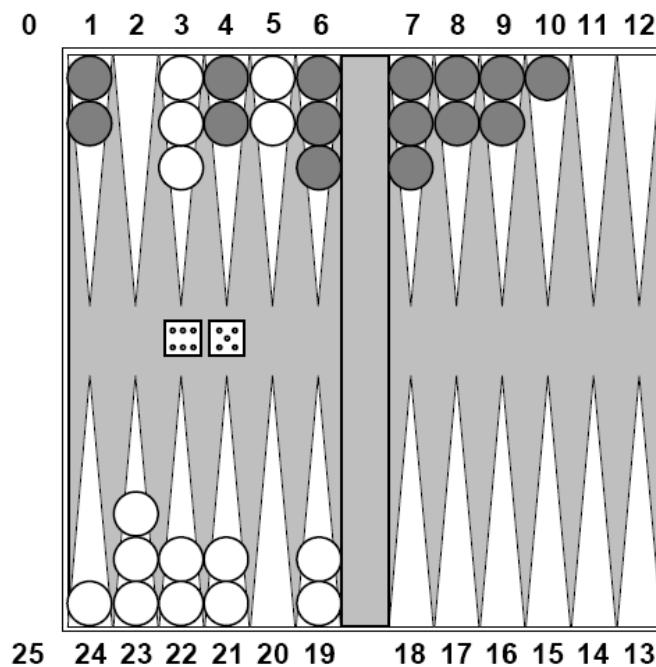
Othello^۵

dice^۶

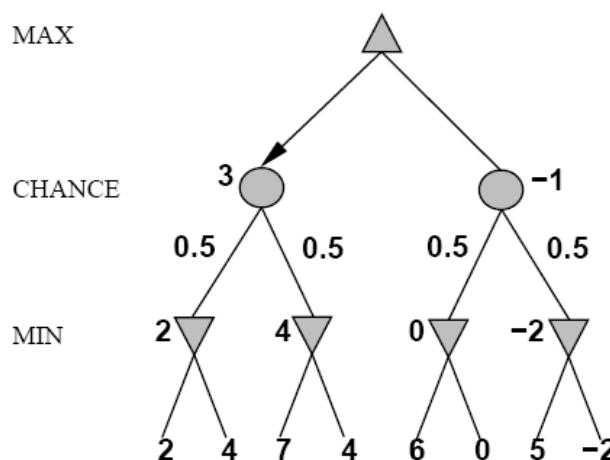
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



مثال ساده شده با سکه ی دورویه ^۲ :



card-shuffling
coin-flipping

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

الگوریتمی برای بازی های غیر قطعی

از آمار و احتمالات داریم ؛ مقدار مورد انتظار ار یک متغیر تصادفی X میانگینی از مقدارهای ممکن X می باشد و توسط احتمالات رخ داده $P(x)$ توزیع شده است :

$$E(x) = \sum_{x \in X} xP(x)$$

الگوریتم Minimax : شبیه Minimax می باشد ، تفاوت آن با Expectminimax است که از مقدارهای مورد انتظار در گره های تصادفی هم استفاده می نماید . بازی EXpectminimax کامل را ارایه می نماید ؛ مینیماکس هم بازی کامل را ارایه می کند در ضمن ما باید از گره های تصادفی هم استفاده نماییم :

...

در صورتی که حالت (state) یک گره ماکزیمم است ، بالاترین-ExpectMinimax Successors(state) را برگردان از Value

در صورتی که حالت (state) یک گره مینیمم است ، کمترین-ExpectMinimax Successors(state) را برگردان از Value

در صورتی که حالت (state) یک گره شانسی است ، میانگین-ExpectMinimax Successors(state) را برگردان از Value

...

توضیحی در مورد Deep Blue

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Deep Blue ، یک مثال خوب از ترکیب توان محاسباتی مدرن و تکنیک ها (روش ها) ای هوش مصنوعی می باشد . در سال ۱۹۹۷ میلادی ، Deep Blue گری گاسپاروف (بهترین شطرنج باز بشری) را مغلوب نمود . و این کار ، یک هدف هوش مصنوعی از دهه ۵۰ میلادی بود . Deep Blue یک ابررایانه ی RS/6000 32-Node می باشد . برنامه ی Deep Blue به زبان سی نوشته شده بود و از جستجوی آلفا – بتا استفاده می نمود و دارای سخت افزار مخصوص محاسبه کننده ی توابع ارزیابی بود . تعداد ارزیابی ها ، دویست میلیون حالت در ثانیه بود که $100 - 200 \text{ میلیون دقت در سه دقیقه}$ ، ارزیابی می کند . در ضمن میانگین فاکتور انشعاب در شطرنج ۳۵ می باشد و فضای حالت در آن حدود ^{۷۰} ۱۰ می باشد . مقدار زیادی از دانش با دامنه ی مخصوص افراد برای به وجود آوردن دقت در توابع ارزیابی به کار گرفته شد و از پایگاه داده های استادان بزرگ شطرنج در بازی های قبلی و همچنین از تعداد زیادی کتاب های پیشرفته استفاده شده بود .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل نهم

عامل های منطقی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- عامل ها براساس دانش^۱
- دنیای Wumpus
- منطق در حالت کلی - مدل ها و دارایی ها^۲
- منطق گزاره ای^۳ (بولین)
- تساوی^۴، صحت^۵، توانایی راضی کردن^۶
- قوانین استنتاج^۷ و اثبات قضیه^۸
- زنجیره ای مستقیم^۹
- زنجیره ای معکوس^{۱۰}
- تحلیل^{۱۱}

Knowledge-based agents^۱
 entailment^۲
 Propositional^۳
 Equivalence^۴
 validity^۵
 satisfiability^۶
 Inference rules^۷
 theorem proving^۸
 forward chaining^۹
 backward chaining^{۱۰}

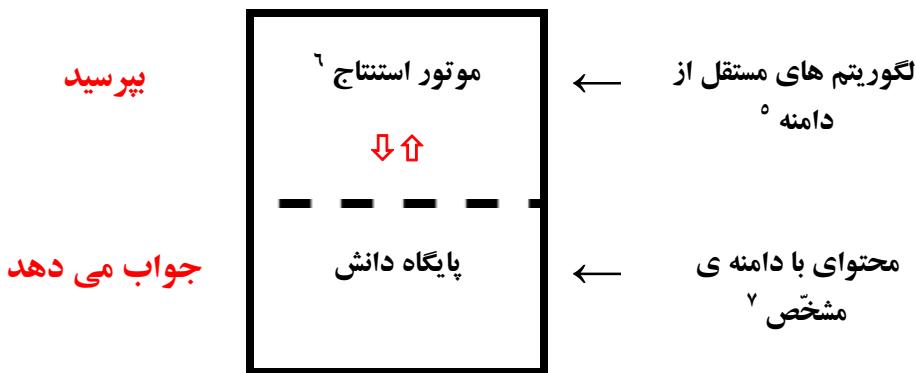
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پایگاه های دانش

یک پایگاه دانش^۲، تشکیل شده از جملاتی که واقعیّاتی را در مورد جهان [خود] ارایه می کنند. به بیان دیگر، پایگاه دانش، مجموعه ای از عبارت ها است که به وسیله ی یک زبان رسمی^۳ بیان شده اند. حال این سؤال مطرح می شود که تفاوت میان پایگاه دانش و پایگاه داده^۴ چیست؟: به طور کلی، در طرز بیان و کاربرد آن هاست و در عمل، یک پایگاه دانش ممکن است از یک پایگاه داده استفاده کند. جملات در مورد چیزهایی که در جهان وجود دارد مثل Wumpus ها، طلا، چاله ها، فضاهای و ارتباط میان عامل ها توضیح می دهند.



resolution^۱

Knowledge Base^۲

formal^۳

Database^۴

domain-independent algorithms^۵

inference engine^۶

domain-specific content^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

قانون استنتاج: وقتی کسی از پایگاه دانش می پرسد ، جواب باید از آن چه که قبل از پایگاه دانش گفته ایم بیايد .

شیوه ^۱ اعلانی یا اظهاری برای ساختن یک عامل (یا یک سیستم دیگر) :

به پایگاه دانش **بگویید** () ^۲ که می خواهید چه چیزی را بدانید . سپس از خودش **سؤال می کند** () ^۳ که چه کاری باید انجام دهد ؛ جواب هایی که می دهد باید از KB یا پایگاه دانش گرفته شوند . عامل ها می توانند در سطح دانش ^۴ دیده شوند . توجه نمایید که چه چیزی را آن ها می دانند ، بدون این که بدانند چگونه به کار گرفته می شوند ، یا در **محله** ^۵ کاربرد به ساختمان های داده ای ^۶ در پایگاه دانش و الگوریتم هایی که آن ها را به کار می برنند توجه می کنند .

یک عامل ساده ی براساس دانش

الگوریتم :

تابع KB یک عمل را برمی گرداند

متغیر KB که یک متغیر static است ، یک پایگاه دانش می باشد

t ، یک شمارنده با مقدار اوّلیه ی صفر و نشان دهنده ی زمان می باشد

approach^۱

Tell ()^۲

Ask ()^۳

knowledge level^۴

implementation level^۵

data structures^۶

مترجم: شهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Tell(KB,Make-Percept-Sentence(percept,t))

action \leftarrow Ask(KB,Make-Action-Query(t))

Tell(KB,Make-Action-Sentence(action,t))

$t \leftarrow t+1$

عمل (action) را برگردان

عامل باید قادر باشد: حالت ها ، عملکرد ها را ارایه نماید ؛ دریافت های ادراکی ^۱ جدید را یکی کند ^۲ ؛ ارایه های درونی جهان را به روز نماید ؛ خصوصیات پنهان جهان خود را استنباط ^۳ نماید و عملکردهای مناسب را استنباط نماید .

توضیح: عامل پایگاه دانش ، شبیه عامل های با وضعیت درونی می باشد . عامل ها می توانند در سطح های مختلف ، بیان شوند :

۱- سطح دانش : چه می دانند ، بدون توجه به پیاده سازی واقعی .

۲- سطح پیاده سازی : ساختمان های داده در پایگاه دانش و الگوریتم هایی که آن ها به کار می گیرند ؛ به عنوان مثال ، منطق گزاره ای و نتیجه ^۴ .

انواع دانش

percepts ^۱

Incorporate ^۲

deduce ^۳

resolution ^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

۱- **رویه ای**: به عنوان مثال ، تابع ها که مثل دانش می توانند فقط به یک روش و آن هم در زمان اجرا مورد استفاده قرار گیرند .

۲- **بیانی**: محدودیت ها و قانون ها که می توانند برای انجام انواع زیادی از استنباط ها مورد استفاده قرار گیرند .

Wumpus دنیا

Wumpus ، نام یک بازی کامپیوتری است ؛ در این بازی ، عامل یک غار را که از اتاق های متصل شده توسط راه را تشکیل شده کشف می کند . کمین گاه ، جایی است که در آن Wumpus قرار دارد و Wumpus ، جانوری است که هر عاملی را که به اتاقش وارد شود می خورد . همچنین ، برخی از اتاق ها دارای چاله های بدون کف هستند که هر عاملی را که در اتاق سرگردان است را به تله می اندازند . گاهی از وقت ها کپه ای از طلا هم در اتاق هست . هدف ما در بازی این است که طلا ها را جمع کنیم و بدون اینکه خورده شویم خارج شویم .

4	~~~~~ Stench	Breeze	PIT
3	Wumpus	Breeze ~~~~~ Stench Gold	Breeze
2	~~~~~ Stench	Breeze	
1	START	Breeze	Breeze
	1	2	3

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

wumpus دنیا PEAS توضیح

اندازه گیری عملکرد یا معیار کارایی: طلا (+1000)، مرگ (-1000)، (منفی یک) در هر گام، (۱۰-) برای استفاده از پیکان^۱

محیط: مربع های نزدیک به wumpus بدبو^۲ هستند؛ مربع های نزدیک به چاله^۳ خوش هوا^۴ هستند و اگر طلا در همان خانه^۵ خوش هوا باشد، درخشنان تر است؛ اگر به دیوار بخورید، ضربه خواهید خورد؛ اگر Wumpus، کشته شد بدبخت می شوید؛ تیراندازی^۶ wumpus را در صورتی که شما رویه روی آن باشید می کشد؛ تیراندازی فقط پیکان را مصرف^۷ می نماید؛ قلاب^۸، فقط طلا را در صورتی که در همان مربع باشد انتخاب می نماید. آزاد کردن، طلا را در همان خانه باقی می گذارد.

حسگرهای: بوی بد^۹، بوی خوب^{۱۰}، درخشش^{۱۱}، ضربه^{۱۲} و جیغ زدن^{۱۳}

حرکت ها: چپ گرد^۱، راست گرد^۲، حرکت مستقیم^۳، قلاب^۴، آزاد کردن^۵، تیراندازی^۶

arrow ^۱
smelly ^۲
pit ^۳
breezy ^۴
glitter ^۵
use up ^۶
grabbing ^۷
stench ^۸
breeze ^۹
glitter ^{۱۰}
bump ^{۱۱}
scream ^{۱۲}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دنیای Wumpus معمولی

عامل، معمولاً از خانه ۱ شروع می کند. کار عامل این است که طلا را پیدا کند، به خانه ۱ برگرد و از غار خارج شود.

خصوصیات دنیای wumpus

قابل مشاهده بودن نه – فقط دارای ادراک محلی است.

قطعیت نه – نتایج دقیقاً مشخص است.

دوره ای نه – دارای ترتیب در سطح عملکردها است.

ایستایی نه – wumpus و چاله ها نمی توانند حرکت نمایند.

گرسسته نه.

تک عاملی نه – wumpus در اصل یک طرح طبیعی می باشد.

حرکت عامل در دنیای Wumpus

Left turn^۱

Right turn^۲

Forward^۳

Grab^۴

Release^۵

Shoot^۶

مترجم: شهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

1.4	2.4	3.4	4.4
1.3	2.3	3.3	4.3
1.2	2.2	3.2	4.2
OK			
1.1 A OK	2.1	3.1	4.1

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1.4	2.4	3.4	4.4
1.3	2.3	3.3	4.3
1.2	2.2 P?	3.2	4.2
OK			
1.1 V OK	2.1 A B OK	3.1 P?	4.1

(b)

در خانه ۱ [۱] و ۲ [۲]، پایگاه دانش دارای قانون های محیط است. اوّلین دریافت یا ادراک [هیچ و هیچ و هیچ و هیچ] است، به یک خانه ای امن^۱ مثل [۱] و ۲ [۲] می رویم. خانه ۱ [۱] و ۲ [۲] چون خوش بو است نتیجه می گیریم که چاله در [۱] و ۲ [۲] یا [۱] و ۳ [۳] است؛ در نتیجه به خانه ۱ [۱] برمی گردیم و خانه ۱ امن بعدی را امتحان می نماییم.

1.4	2.4	3.4	4.4
1.3 W?	2.3	3.3	4.3
1.2 A S OK	2.2	3.2	4.2
1.1 V OK	2.1 B V OK	3.1 P?	4.1

(a)

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

1.4	2.4 P?	3.4	4.4
1.3 W?	2.3 A S G B	3.3 P?	4.3
1.2 S V OK	2.2 V OK	3.2	4.2
1.1 V OK	2.1 B V OK	3.1 P?	4.1

(b)

safe square^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در خانه $[1,2]$ ، از بوی تعفن خانه می فهمیم که Wumpus در خانه $[3,2]$ یا $[2,2]$ است. در خانه $[1,2]$ یا $[2,2]$ یا $[1,1]$ هم نیست. در نتیجه Wumpus در خانه $[3,2]$ است. بنابراین، خانه $[2,2]$ ، امن است چون بوی خوش در خانه $[2,2]$ نیست؛ بنابراین، چاله در خانه $[1,2]$ می باشد. بنابراین، به خانه $[2,2]$ می رویم.

1.4	2.4	3.4	4.4
1.3 W ₁	2.3	3.3	4.3
1.2 A S OK	2.2	3.2	4.2
1.1 V OK	2.1 B V OK	3.1 P ₁	4.1

(a)

A - Agent
B - Breeze
G - Glitter, Gold
OK - Safe square
P - Pit
S - Stench
V - Visited
W - Wumpus

1.4	2.4 P ₁	3.4	4.4
1.3 W ₁	2.3 A S G B	3.3 P ₁	4.3
1.2 S V OK	2.2 V OK	3.2	4.2
1.1 V OK	2.1 B V OK	3.1 P ₁	4.1

(b)

از خانه $[2,2]$ به خانه $[3,2]$ می رویم. در خانه $[3,2]$ درخشش طلا، بوی بد و بوی خوش را داریم، بنابراین طلا را برابر می داریم و از بوی خوش موجود در این خانه نتیجه می گیریم که چاله در خانه $[3,3]$ یا $[4,2]$ می باشد. سپس از مسیر امنی که قبلا آمده ایم بر می گردیم و در نهایت، بازی خاتمه می یابد.

منطق چیست؟

منطق، روشی کلاسیک برای ارایه ی دانش عامل می باشد و به ما اجازه می دهد که عامل ها را به صورت اعلانی (اظهاری)، برنامه ریزی و توصیف کنیم. به بیان دیگر، منطق، یک زبان رسمی است که

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دارای ظاهر^۱ و سmantیک^۲ می باشد ، ظاهر (نحو یا گرامر) ، می گوید که چه عبارت هایی مجازند مثلا در ریاضیات ، عبارت $x+2=5$ به صورت گرامری درست می باشد ، اما $x+3=3$ به صورت گرامری درست نمی باشد . به عنوان مثال دیگر ، در زبان ریاضی ، $x+2 \geq y$ یک جمله است ولی x^2+y یک جمله نمی باشد . و معنا یا سmantیک ، مشخص می نماید که یک عبارت در چه موقعی درست یا در چه موقعی ، غلط می باشد ؛ معنای $x+2=5$ در زمانی درست است که $x=3$ باشد و در غیر این صورت غلط می باشد . به عنوان مثال دیگر ، $x+2 \geq y$ درست است در صورتی که عدد $x+2$ کم تر از y نباشد . در جهان در صورتی که $x=7, y=1$ باشد صحیح می باشد . در جهان در صورتی که $x=0, y=6$ غلط است . توجه کنید که ، عبارت های منطقی ، باید یا درست باشند و یا غلط باشند ؛ به بیان دیگر ، "درجه ی درستی ، وجود ندارد" .

ارتباطات موجود در جهان

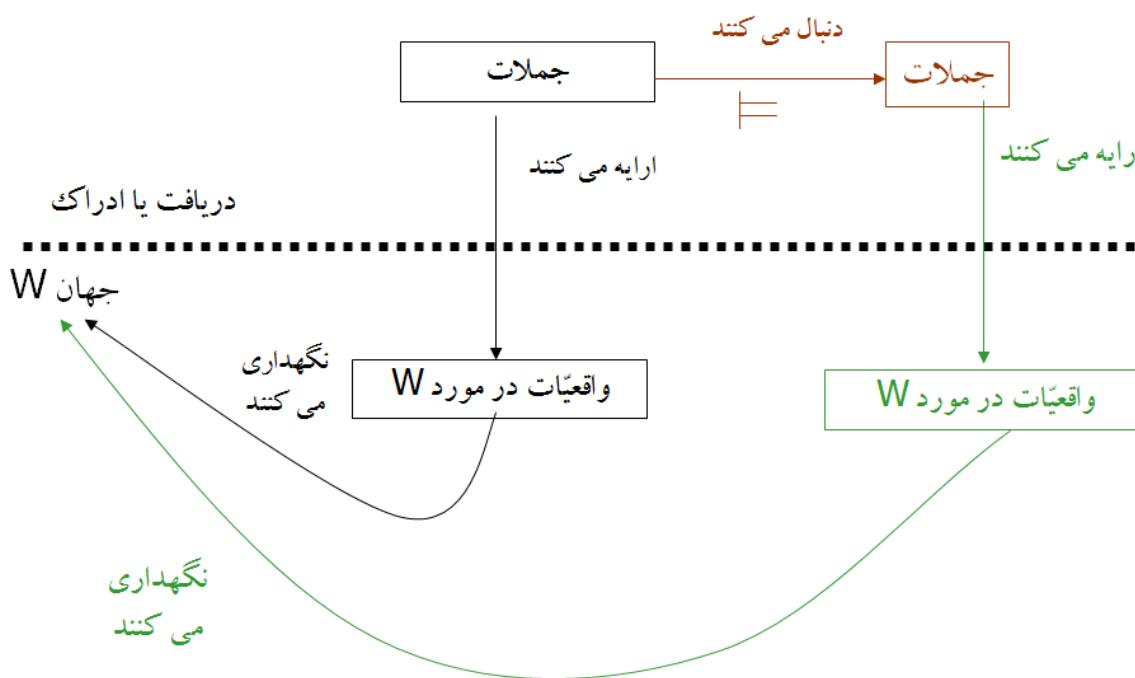
Syntax^۱
Semantics^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



دنبال کردن^۱ - معنای دنبال کردن این است که چیزی توسط دیگری دنبال شود: $\models \alpha$.

پایگاه دانش KB جمله α را دنبال می کند اگر و فقط اگر α در همه W درست باشد. به عنوان مثال، $x+y=4$ را دنبال می نماید.

دنبال کردن، ارتباطی میان جملات است که براساس سماتیک ها یا معناها می باشد. به عنوان مثال دیگر، اگر پایگاه دانش، دارای "پیراهن، سبز رنگ می باشد" و "پیراهن، خط دار (راه راه) می باشد" باشد، در این صورت "پیراهن، سبز رنگ است یا راه راه است را دنبال می نماید". به صورت تکنیکی (مطابق اصول فنی)، دنبال کردن، می گوید که: برای همه a مدل هایی که a درست است، b هم درست می باشد؛ مثلا، $a+2=5 \models a=3$. دنبال کردن به ما اجازه خواهد داد که استدلال نماییم و واقعیات جدیدی را به پایگاه دانش عامل مان اضافه نماییم.

^۱: استلزم یا ایجاب کردن

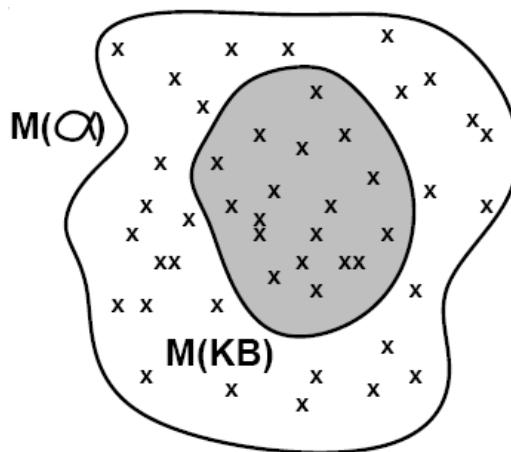
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مدل ها - مدل ها ، تعریف های رسمی وضعیت های ممکن جهان هستند. گوییم m مدلی از یک جمله α است در صورتی که α در m درست باشد. اگر $M(\alpha)$ مجموعه ای از تمام مدل های α باشد در این صورت $KB \models \alpha$ اگر و فقط اگر $M(KB) \subseteq M(\alpha)$ باشد.



دبال کودن در دنیای wumpus - بعد از اینکه در خانه ۱ [۱] هیچ چیزی را پیدا نکردیم حرکت بعدی ما ، حرکت به راست در خانه ۲ [۲] خوش هواست. مدل های ممکن برای؟ برای احتمال وجود چاله یا عدم وجود چاله چیستند؟ . سه انتخاب بولین وجود دارد و در نتیجه ۸ مدل ممکن است .

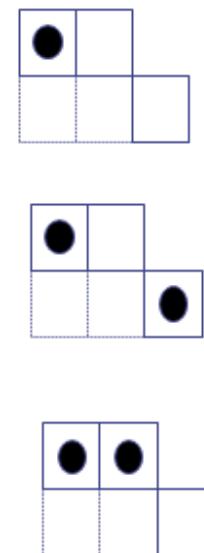
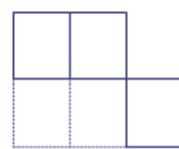
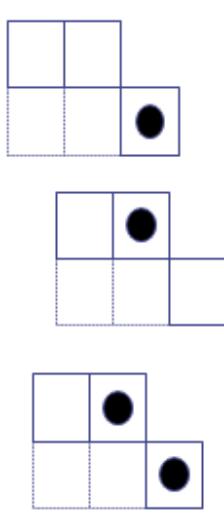
?	?		
A	B	A	?

مدل های Wumpus

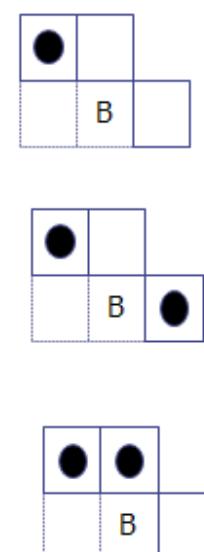
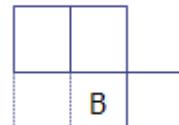
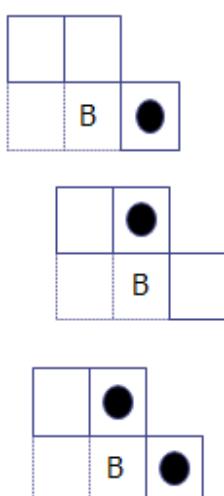
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



Wumpus مدل های

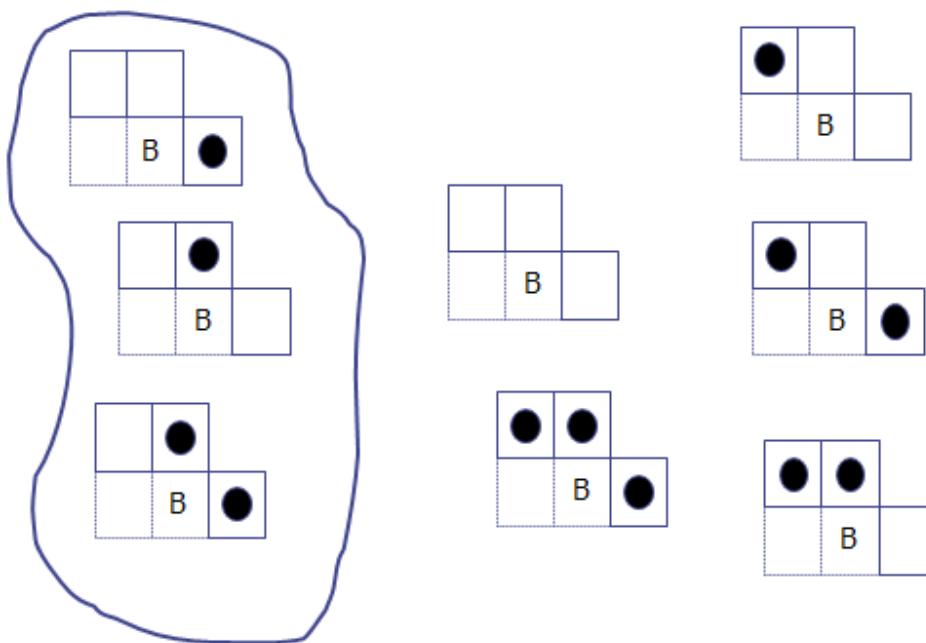


Wumpus مدل های

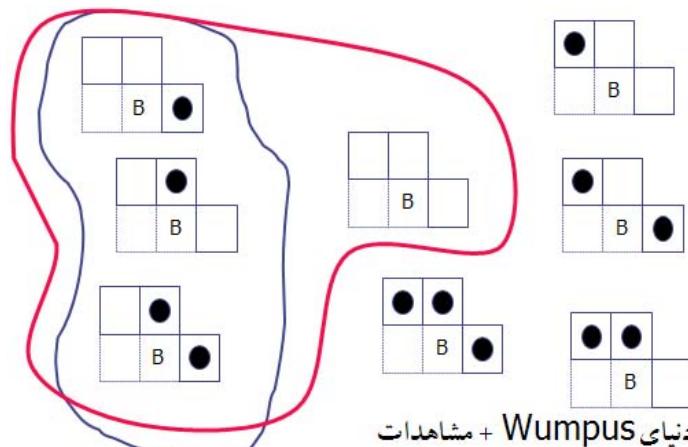
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



Wumpus های مدل های



پایگاه دانش (KB) = دنیای Wumpus + مشاهدات

α_1 = "[1 2]" [یعنی است]

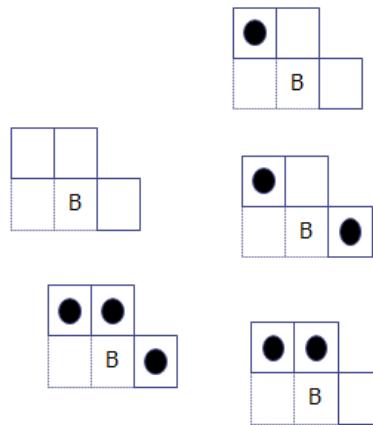
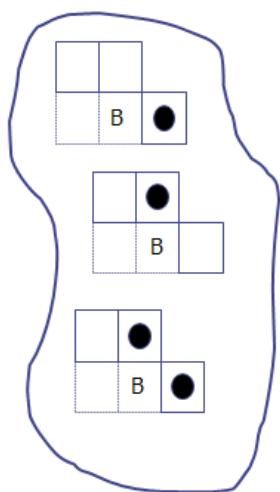
KB | = α_1

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Wumpus مدل های



پایگاه دانش (KB) = دنیای مشاهدات + Wumpus

α_2 = "خانه ی [۲] و [۲] امن است"

KB |= α_2 ??

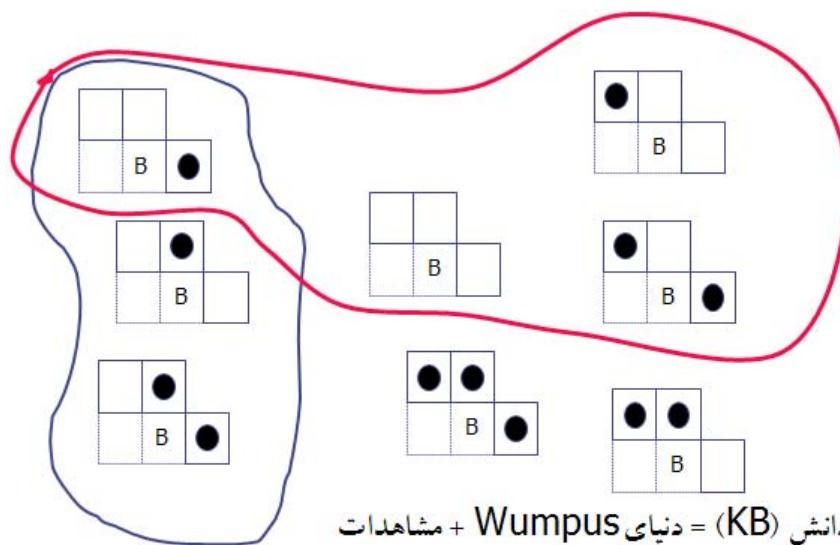
Wumpus مدل های

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



پایگاه دانش (KB) = دنیای Wumpus + مشاهدات

$\alpha_2 = \text{``خانه } i [2\text{ و } 2] \text{ امن است''}$

$KB \not\models \alpha_2 \quad \text{نیست!}$

استنتاج منطقی

نظریه‌ی دنبال کردن، که قبل آن را بررسی کردیم، می‌تواند برای استنتاج منطقی مورد استفاده قرار گیرد. بررسی مدل^۱ (مثال Wumpus را ببینید) همه‌ی مدل‌های ممکن را بررسی می‌کند و چک می‌کند که آیا α درست است.

در صورتی که یک الگوریتم فقط جمله‌های دنبال شده را به وجود می‌آورد، **نگهدارنده‌ی درستی^۲** یا **کامل^۱** نام دارد، در غیر این صورت فقط چیزها را به وجود می‌آورد. در صورتی کامل است که در جایی که $\alpha_i \models KB$ ، این هم درست باشد که α .

model checking^۱
truth preserving^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

کمال یا تمامیت^۲: الگوریتم می تواند هر عبارتی که دنبال می شود را به وجود آورد. ا در صورتی دارای تمامیت یا کمال است که در هر جایی که $KB / \neg_i \alpha$ ، این هم درست باشد که $KB / \neg_i \alpha$.

استنتاج^۳

KB؛ یعنی، جمله α می تواند از KB توسط روال^۴ مشتق شود (به وجود آید).

کامل بودن^۵: ا در صورتی کامل است که در جایی که $KB / \neg_i \alpha$ ، این هم درست باشد که $KB / \neg_i \alpha$.

کمال یا تمامیت^۶: ا در صورتی دارای تمامیت یا کمال است که در هر جایی که $KB / \neg_i \alpha$ ، این هم درست باشد که $KB / \neg_i \alpha$.

ما می خواهیم منطقی را که به اندازه‌ی کافی رساند برای گفتن تقریباً هر چیزی که می خواهیم و برای این که بفهمیم کدام یک صحیح است و نتیجه‌ی کدام زیر برنامه کامل می باشد، تعریف نماییم.

مثال‌هایی از منطق

منطق گزاره‌ای : $A \wedge B \Rightarrow C$

sound ^۷
completeness ^۸
inference ^۹
procedure ^{۱۰}
soundness ^{۱۱}
completeness ^{۱۲}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

($\forall x(\exists y \text{Mother}(y,x))$: منطق مرتبه ای اول :

$B(\text{John}, \text{Father}(\text{Zeus}, \text{Cronus}))$: منطق عقده ای یا زود باور^۱ :

منطق گزاره ای

ظاهر - منطق گزاره ای ، ساده ترین منطقی است که ایده های اساسی را توضیح می دهد .
نمادهای گزاره ای ، P_1, P_2 و می باشند . اگر S یک جمله باشد ؛ $\neg S$ ، عبارت نقیض ^۲ آن می باشد .

برای عبارات S_1 و S_2 :

$(S_1 \wedge S_2)$ AND (ترکیب فصلی ^۳ یا جدایی ؛ در صورتی که ورودی (ها) صحیح باشند ، نتیجه درست خواهد بود) این دو خواهد بود .

$(S_1 \vee S_2)$ OR (ترکیب عطفی ^۴ ؛ در صورتی که از S_1 و S_2 یکی درست باشد ، نتیجه درست خواهد بود) این دو خواهد بود .

$A \Rightarrow S_1 \Rightarrow S_2$ نتیجه گیری (استلزم یا استنباط ^۵) خواهد بود و در صورتی درست است که درست باشد و B هم درست باشد ؛ مثلا ، اگر الان هوا بارانی باشد ، آن گاه الان هوا ابری می باشد . استلزم به ما امکان استنتاج را می دهد .

Logic of Belief^۶
negation^۷
conjunction^۸
disjunction^۹

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نتنه: $S_1 \vee S_2 \Rightarrow S_2$ معادل است با $\neg S_1 \vee S_2$.

استنباط دو شرطی ($S_1 \Leftrightarrow S_2$) اگر و فقط اگر^۱ و ($S_1 \Rightarrow S_2$) خواهد بود. به عنوان مثال، $(A \Rightarrow B) \vee (\neg C) \wedge (P \wedge Q) \Rightarrow R$

ترتیب اولویت ها^۲

به ترتیب از چپ به راست عبارتند از \Rightarrow ، \wedge ، \vee ؛ به عنوان مثال، $\neg A \vee B \Rightarrow ((\neg A) \wedge B) \Rightarrow C$

مدل های موجود در منطق گزاره ای

مدل، نسبت دادن درست یا غلط به هر عبارت اتمیک (تجزیه تا پذیر) است؛ اگر A ، B ، C و D مدل های گزاره ای باشند؛ $m = \{A=true, B=false, C=false, D=true\}$ ، یک مدل است؛ همچنین، $m' = \{A=true, B=false, C=false\}$ هم یک مدل است. چند مدل می توانند برای n مدل گزاره ای تعریف شوند؟^۳

سمانیک ها یا معناهای منطق گزاره ای، هر مدل درست^۴ / غلط^۵ را برای هر نشان^۶ مشخص می نماید.

implication^۷

biconditional^۸

Order of Precedence^۹

true^{۱۰}

false^{۱۱}

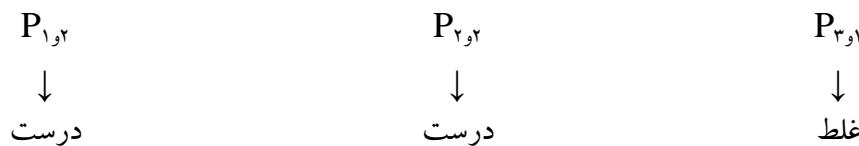
symbol^{۱۲}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



با این سه نشان، می توانیم هشت مدل داشته باشیم.

قوانينی برای ارزیابی درستی به وسیله‌ی یک مدل S : در صورتی که S غلط

باشد، صحیح است | $S_1 \wedge S_2$ در صورتی که S_1 و S_2 درست باشند، درست است | در $S_1 \vee S_2$

صورتی که S_1 یا S_2 درست باشند، درست است | $S_1 \Rightarrow S_2$ در صورتی که S_1 غلط باشد یا S_2

درست باشد، درست است؛ توجه نمایید که در صورتی که S_1 درست باشد و S_2 غلط باشد، غلط

است. | $S_1 \Leftrightarrow S_2$ در صورتی که $S_1 \Rightarrow S_2$ هم درست باشد، درست است

. مثال:

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{false} \vee \text{true}) = \text{true} \wedge \text{true} = \text{true}$$

جدول صحت برای رابطه ها

$P \Leftrightarrow Q$	$P \Rightarrow Q$	$P \vee Q$	$P \wedge Q$	$\neg P$	Q	P
درست	درست	غلط	غلط	درست	درست	غلط
غلط	درست	درست	غلط	درست	غلط	غلط
غلط	درست	درست	غلط	غلط	درست	درست
درست	غلط	غلط	درست	درست	غلط	درست

عبارت $A \Rightarrow B$ را به این صورت بخوانید؛ اگر A درست باشد آن‌گاه B درست خواهد بود

و در غیر این صورت ادعایی ندارم.

جملات دنیای wumpus

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

"چاله ها باعث ایجاد هوای خوش در خانه های اطراف می شوند"

$P_{i,j}$ در صورتی که یک چاله در $[i,j]$ وجود دارد، درست می باشد.

$B_{i,j}$ در صورتی که یک هوای خوش (breeze) در $[i,j]$ وجود داشته باشد، درست می باشد.

$$\neg P_{1,1}$$

$$\neg B_{1,1}$$

$$B_{2,1}$$

"چاله ها باعث ایجاد هوای خوش در خانه های اطراف می شوند"

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

"یک خانه خوش هوا است اگر و فقط اگر یک چاله در اطرافش وجود داشته باشد"

جداوی صحّت برای استنتاج

KB	R_5	R_4	R_3	R_2	R_1	$P_{3,1}$	$P_{2,2}$	$P_{2,1}$	$P_{1,2}$	$P_{1,1}$	$B_{2,1}$	$B_{1,1}$
غلط	غلط	درست	درست	درست	درست	غلط						
غلط	غلط	درست	درست	غلط	درست	درست	غلط	غلط	غلط	غلط	غلط	غلط
.
.
.
غلط	درست	درست	درست	غلط	درست	درست	غلط	غلط	غلط	غلط	درست	غلط
درست	درست	درست	درست	درست	درست	درست	غلط	غلط	غلط	غلط	درست	غلط
درست	درست	درست	درست	درست	درست	غلط	درست	غلط	غلط	غلط	درست	غلط
درست	درست	درست	درست	درست	درست	درست	درست	غلط	غلط	غلط	درست	غلط
غلط	درست	درست	درست	غلط	درست	غلط	غلط	درست	غلط	غلط	درست	غلط
.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

.
غلط	درست	درست	غلط	درست	درست	غلط	درست							

در شمارش سطراها (با انتساب های مختلف به سطراها) ، در صورتی که KB در سطر درست است ، از α هم صرف نظر نمایید .

استنتاج با شمارش

شمارش Depth-first برای همه مدل ها صحیح و کامل است .

تابع $TT\text{-Entails?}(KB, \alpha)$ درست یا غلط را برمی گرداند

ورودی ها : KB ، که یک پایگاه دانش می باشد و یک جمله در منطق گزاره ای می باشد و α که یک صفت می باشد و به صورت یک جمله ای منطق گزاره ای می باشد
یک لیست از سمبول های گزاره ای در KB و $\alpha \leftarrow symbols$

تابع $TT\text{-Check-All}(KB, \alpha, symbols, [])$

تابع $TT\text{-Check-All}(KB, \alpha, symbols, model)$ درست یا غلط را برمی گرداند

در صورتی که $Empty?(symbols)$ صحیح می باشد ،

در صورتی که $PL\text{-True?}(\alpha, model)$ دارای ارزش درست است ، $PL\text{-True?}(KB, model)$ را برگردان .

در غیر این صورت درست (true) را برگردان

در غیر این صورت

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$P \leftarrow \text{First(symbols)}; rest \leftarrow \text{Rest(symbols)}$

TT-Check-
و
TT-Check-All(KB,α,rest,Expand(P,true,model))
All(KB,α,rest,Expand(P,false,model)) را برگردان

برای n سمبول می باشد؛ مسئله به صورت $O(2^n)$ co-NP-complete می باشد.

تساوی های منطقی

دو عبارت در صورتی معادل منطقی می باشند که در مدل های یکسان برابر باشند:

$\alpha \equiv \beta \vdash \alpha \models \beta$ و فقط اگر

$\wedge^1 : \text{جایی پذیری}$ $(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$

$\vee^1 : \text{جایی پذیری}$ $(\alpha \vee \beta) \equiv (\beta \vee \alpha)$

$\wedge^2 : \text{شرکت پذیری}$ $((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$

$\vee^2 : \text{شرکت پذیری}$ $((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$

$\neg\neg^3 : \text{حذف نقیض دوگانه}$ $\neg(\neg\alpha) \equiv \alpha$

$\neg\neg\neg^1 : \text{مفهوم مخالف}$ $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$

¹ commutativity

² associativity

³ double-negation elimination

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

^۱: حذف استنتاج $(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$

^۲: حذف شرط دو طرفه $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$

^۳: قانون دمورگان $\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$

^۴: قانون دمورگان $\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$

^۵: توزیع پذیری $\alpha \wedge (\beta \vee \gamma) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$

^۶: توزیع پذیری $\alpha \vee (\beta \wedge \gamma) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

صحّت و قابلیت ارضاء - یک عبارت در صورتی درست است که در همهٔ مدل‌ها درست باشد. مثلاً، $True, A \vee \neg A, A \Rightarrow A, (A \wedge (A \Rightarrow B)) \Rightarrow B$. صحّت مربوط می‌شود به نتیجهٔ گیری از راه قضیّهٔ قیاس $\models \alpha$: KB $\models \alpha$ اگر و فقط اگر $(KB \Rightarrow \alpha)$ معجاز باشد. یک عبارت ناراضی کننده است اگر در برخی از مدل‌ها درست باشد؛ مثل، $A \vee B, C$ و یک جمله ناراضی کننده است اگر در هیچ مدلی درست نباشد؛ مثل، $A \wedge \neg A$.

قابلیت ارضاء وابسته به نتیجهٔ بودن درست آمده از راه زیر است: $\models \alpha$: KB اگر و فقط اگر $\models \alpha$: KB ناراضی کننده باشد. α را به وسیلهٔ برهان خلف^۷ ثابت نمایید.

^۱ contraposition

^۲ implication

^۳ biconditional elimination

^۴ De Morgan

^۵ distributivity

Deduction Theorem^۹

^۷ reductio ad absurdum

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

متدهای اثبات ۱

به طور کلی متدهای اثبات به دو دسته تقسیم می شوند:

- ✓ استفاده از قوانین استنتاج : در این روش ، معمولاً به ترجمه‌ی عبارات به صورت یک فرم نرمال نیازمندیم و برای اثبات ، یک سری از قوانین استنتاج را به کار می بریم .
- ✓ بردسی مدل ^۱ : که در این روش ، از جدول صحت استفاده می نماییم .

زنجیره‌ی مستقیم و معکوس

زنجیره‌ی مستقیم : از قوانین و چیزهای موجود استفاده می کنیم تا چیزهای اضافی را به دست آوریم و اگر به عبارت مورد نظر رسیدیم ، دیگر این کار را ادامه نمی دهیم . این روش ، استنتاج مشتق شده از داده ها ^۲ هم نام دارد و برای به دست آوردن واقعیّات یا قوانین جدید ، بسیار مفید می باشد .

$$P \Rightarrow Q, L \wedge M \Rightarrow P, B \wedge L \Rightarrow M, A \wedge P \Rightarrow L, A \wedge B \Rightarrow L, A, B$$

Proof methods ^۱

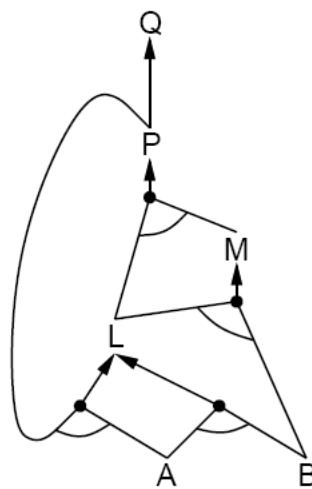
Model checking ^۲

data-driven reasoning ^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



الگوریتم زنجیره‌ی مستقیم

تابع $\text{PL-FC-Entails}(\text{KB}, q)$ درست یا غلط را بر می‌گرداند

ورودی‌ها، KB ، که پایگاه دانش، یک مجموعه از شروط گزاره‌ای شیپوری^۱

q ، صفات، که یک سمبول گزاره‌ای می‌باشد

متغیرهای محلی^۲: count ، یک جدول شاخص گذاری شده توسط شرط، به صورت اولیه حاوی تعداد permis ‌ها می‌باشد

^۱ یک عبارت شیپوری، عبارت‌هایی "یا بی" هستند که در آن‌ها حداقل یک لفظ مثبت وجود دارد؛ مثل، $\neg A \vee \neg B$ و $\neg A \vee \neg B \vee \neg C \vee D$ ؛ عبارت‌های شیپوری می‌توانند به صورت استنتاج‌هایی با یک نتیجه (تالی) نوشته شوند؛ مثلا، $A \wedge B \wedge C \Rightarrow D$ می‌تواند به صورت $\neg A \vee \neg B \vee \neg C \vee D$ نوشته شود. عبارت‌های شیپوری، مبنای پایه‌ی برنامه نویسی منطقی هستند.

local variables^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک جدول شاخص گذاری شده توسط سمبل، هر ورودی دارای مقدار اوّلیه‌ی inferred می‌باشد

agenda، لیستی از سمبل‌ها، به صورت اوّلیه سمبل‌ها در KB شناخته می‌شوند

مادامی که agenda خالی نمی‌باشد کارهای زیر را انجام بده

$p \leftarrow \text{Pop}(\text{agenda})$

و گرنه برای inferred[p] کارهای زیر را انجام بده

$\text{inferred}[p] \leftarrow \text{true}$

برای هر شرط شیپوری C که در p قضیّه به دست می‌آید کارهای زیر را انجام بده

$\text{count}[c] \leftarrow \text{کاهش بده}$

در صورتی که count[c]=0 می‌باشد کارهای زیر را انجام بده

در صورتی که Head[c]=q می‌باشد true را برگردان

$\text{Push}(\text{Head}[c], \text{agenda})$

را برگردان false

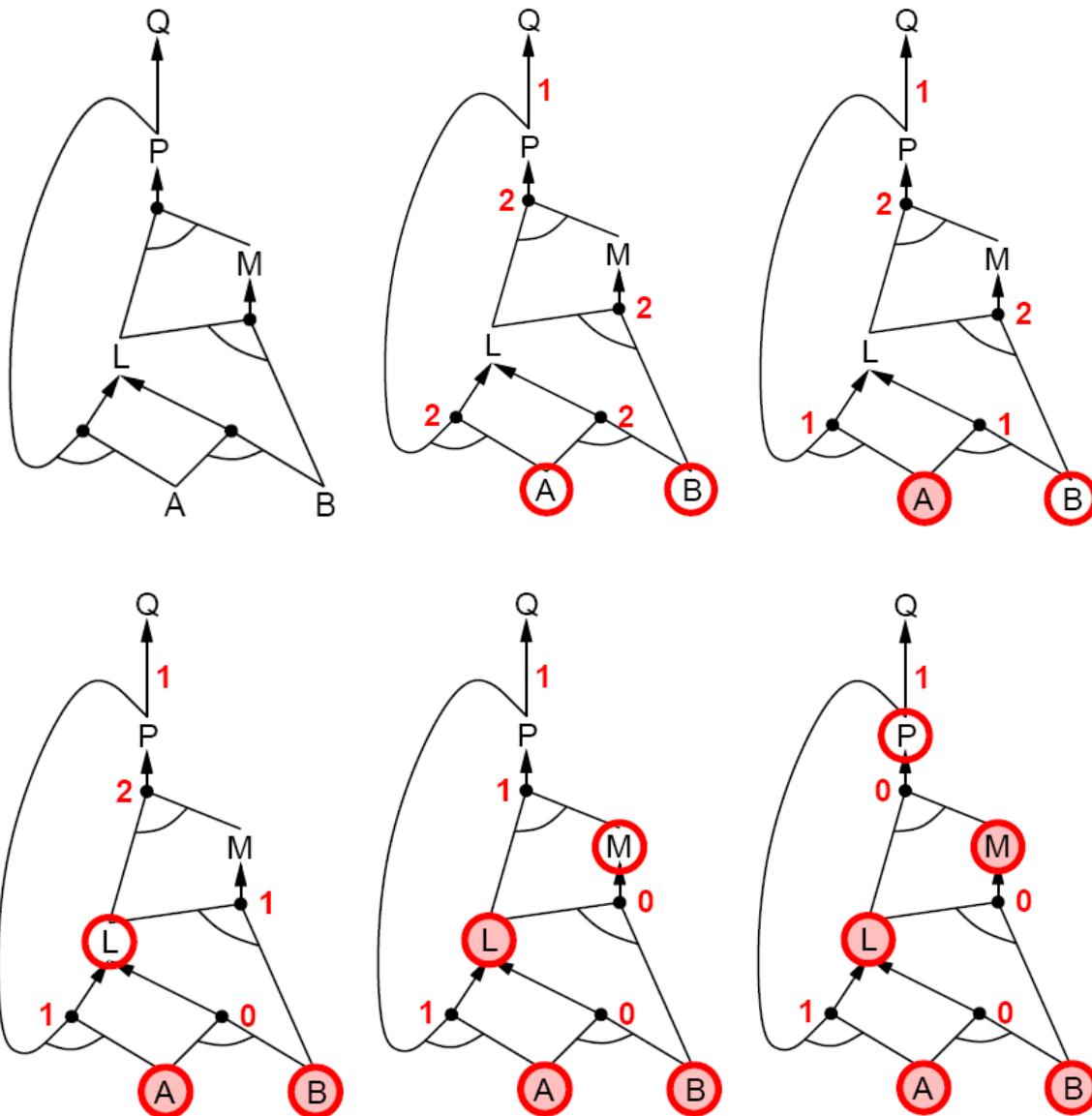
مثال زنجیره‌ی مستقیم

$$P \Rightarrow Q, L \wedge M \Rightarrow P, B \wedge L \Rightarrow M, A \wedge P \Rightarrow L, A \wedge B \Rightarrow L, A, B$$

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

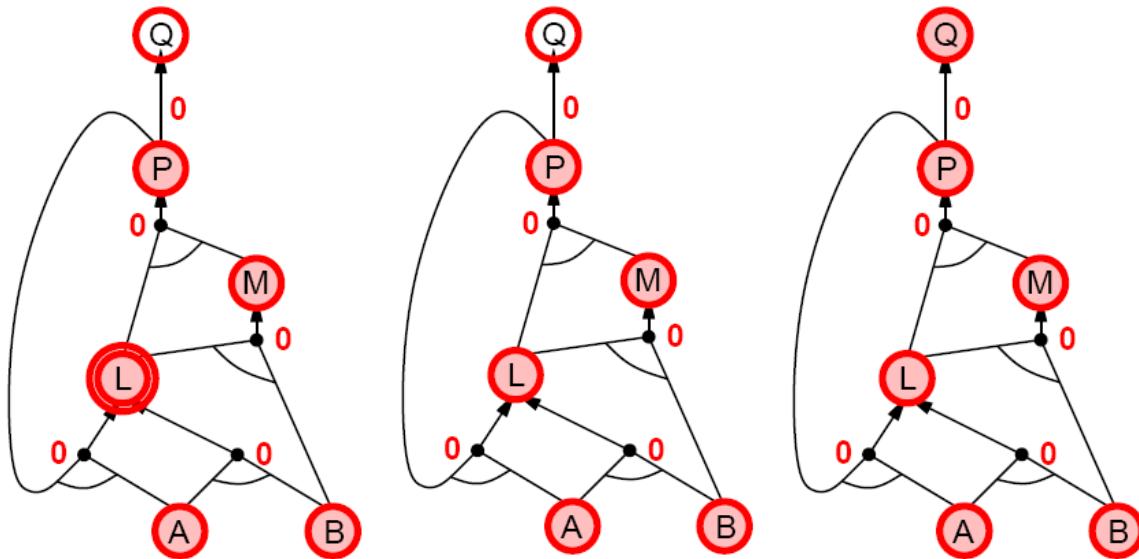


مترجم: سهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی



اثبات تمامیت

یا زنجیره‌ی مستقیم ، هر عبارت اتمیک که توسط KB به وجود آمده است را نتیجه می‌دهد ؛ FC در جایی که هیچ عبارت اتمیک جدیدی مشتق نمی‌شود به یک نقطه‌ی ثابت^۱ می‌رسد ؛ توجه کنید که حالت نهایی به صورت یک مدل m ، نسبت دهنده‌ی درست / غلط به سمبول‌ها می‌باشد ؛ هر شرط در KB اصلی ، در m درست است ؛ از این رو m مدلی از KB می‌باشد ؛ در صورتی که $|KB|=q$ در هر مدل KB ، شامل m درست می‌باشد .

زنجیره‌ی معکوس

از نتیجه ، با پیدا کردن قانون‌هایی که می‌توانند نتیجه را به وجود بیاورند به عقب بر می‌گردد ؛ سپس تلاش می‌کند تا مقدمات قانون‌ها را به دست آورد . این روش برای حل مسأله مناسب است و برخی اوقات ،

fixed point^۱

مترجم: سه راب جلوه گر

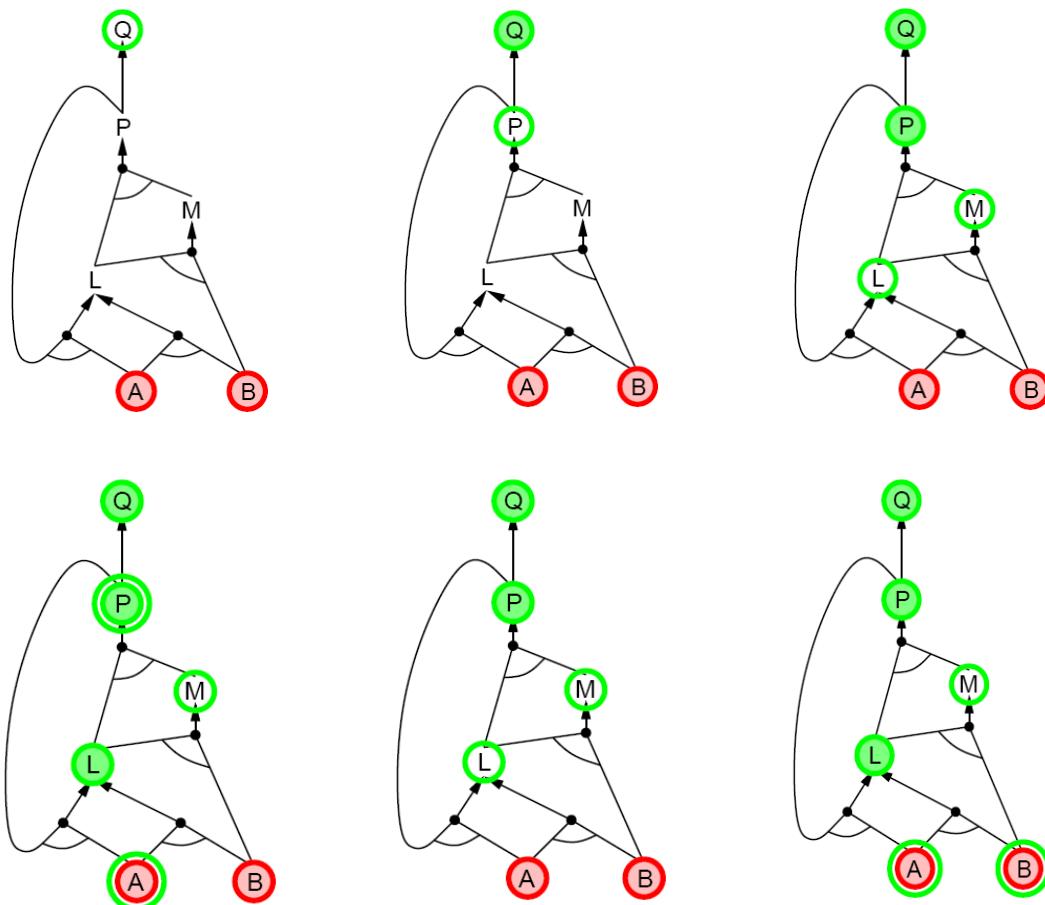
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

استنتاج مشتق شده از پرس و جو^۱ نامیده می شود. در این روش ، احتمال به وجود آوردن اطلاعات جدید و ناشناخته ، کم تر می باشد . زبان برنامه نویسی پرولوگ هم از زنجیره ای معکوس استفاده می نماید .

مثال زنجیره ای معکوس (شکل ها را از چپ به راست دنبال نمایید .)



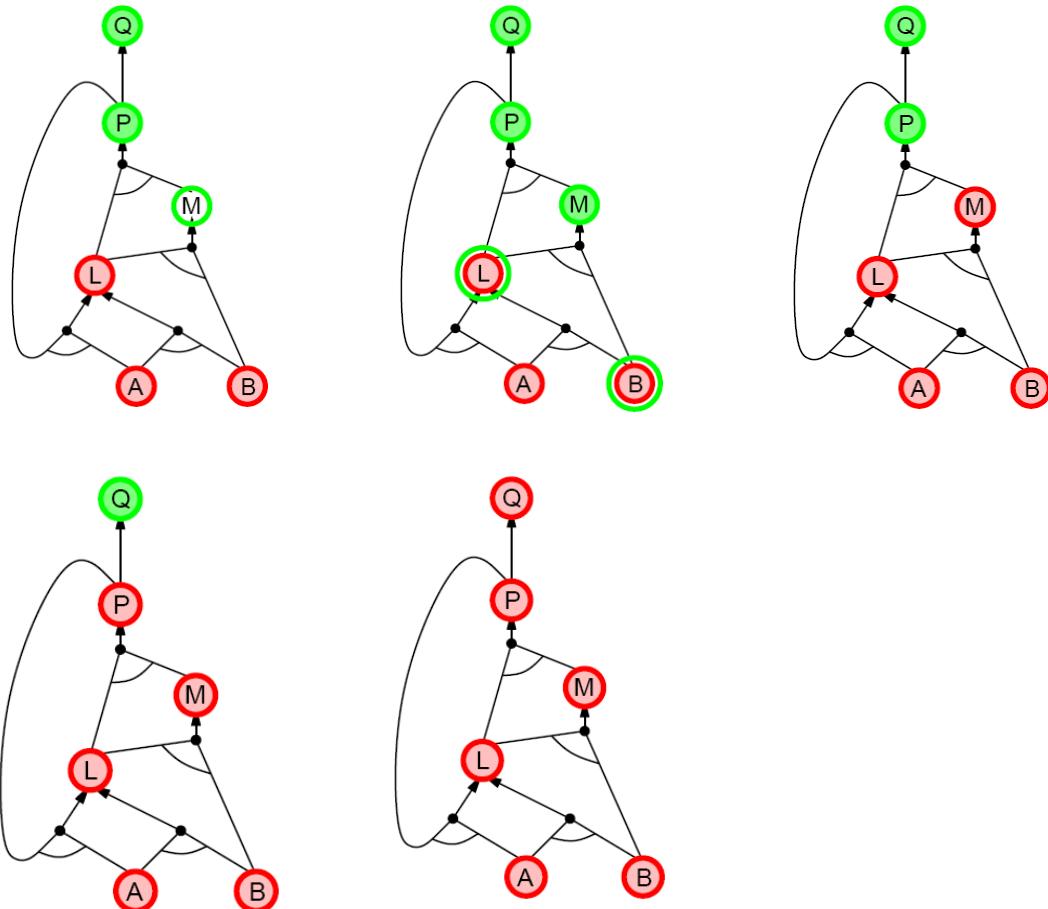
query-driven reasoning^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



زنجیره‌ی مستقیم (FC) و معکوس (BC)

در FC از داده‌ها استفاده می‌کنیم و ممکن است تعدادی کار را که به هدف نامربوطند انجام دهیم در BC از نتیجه‌ها شروع می‌کنیم و برای حل مسئله مناسب می‌باشد.

تحلیل

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

قوانين قبلی ، درست هستند ولی الزاماً کامل نمی باشند . خوشبختانه ، قانون کاملی برای استنتاج وجود دارد که این قانون ، تحلیل نام دارد . قانون تحلیل پایه شبیه این می باشد ، $A \vee B \wedge C$ به ما اجازه می دهد که $B \vee C$ را استنتاج کنیم . در موقع تحلیل ، باید پایگاه دانش ما به صورت CNF باشد .

صورت نرمال ربط دهنده^۱ (CNF)

اغلب ، مفید است که فرمول ها را به صورت های نرمال بنویسیم . صورت های نرمال با آنچه که قبل از نرمال کردن بودند فرقی ندارند و تفاوت آن ها در ظاهر آن هاست و برای استدلال ، مناسب ترند . یک حرف^۲ ، یک چیز تجزیه ناپذیر^۳ یا نقیض آن است مثل ، P یا نقیض آن که $\neg P$ است . یک فرمول به صورت CNF است اگر به صورت $A_1 \wedge A_2 \wedge \dots \wedge A_k$ باشد که A_i تشکیل شده از یا (OR) گزاره ها یا نقیض آن ها . به عنوان مثال ؟

$(p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ به صورت CNF است .

$(\neg p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ به صورت CNF نمی باشد .

$(p \vee q) \wedge r \wedge (p \Rightarrow (\neg r \vee s))$ به صورت CNF نمی باشد .

نکته : هر عبارت منطق گزاره ای می تواند به فرم نرمال ربط دهنده تبدیل شود .

CNF یک عبارت به

مثلا ، $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

CNF یا Conjunctive Normal Form^۴

literal^۵

atom^۶

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

۱. حذف کردن \Leftrightarrow ؛ جایگزین کردن $\alpha \Leftrightarrow \beta$ با $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

۲. حذف کردن \Leftarrow ؛ جایگزین کردن $\alpha \Leftarrow \beta$ با $\neg\alpha \vee \beta$

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

۳. \rightarrow را با استفاده از قوانین دمورگان و قرینه سازی دوگانه به درون عبارات، حرکت دهید:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

قوانین توزیع و گسترش را روی \wedge و \vee به کار ببرید:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

مثال :

فرمول اوّلیه	فرمول نهایی (به صورت CNF)
$A \vee (B \wedge C)$	$(A \vee B) \wedge (A \vee C)$
$(B \wedge C) \vee A$	$(B \vee A) \wedge (C \vee A)$

مزیّت های منطق گزاره ای

- اعلانی یا اظهاری بودن این روش - دانش می تواند از استنتاج، مجزا باشد.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- می تواند اطلاعات جزئی را به کار بگیرد.

- می تواند عبارت های پیچیده تر را به صورت ساده تر تولید نماید.

- مکانیزم های صحیح و کامل دارد (برای عبارت های شیپوری، کارآمد می باشد)

معایب منطق گزاره ای

- افزایش تشریحی در تعداد لفظ ها

- راهی برای تشریح ارتباط های میان اشیا وجود ندارد.

- راهی برای بیان کیفیت، در مورد اشیا وجود ندارد.

منطق مرتبه ی اول، روشی برای رسیدگی کردن به این مسائل می باشد.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل دهم

منطق مرتبه‌ی اول^۱ به

بیان ساده

First-Order Logic(FOL)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

منطق مرتبه‌ی اول، تمام خصوصیات منطق گزاره‌ای را دارد. منطق مرتبه‌ی اول براساس این ایده است که جهان مرکب از دو نوع موجودیت^۱ می‌باشد: اشیا^۲ و ارتباطات^۳; اشیا، معمولاً به نام‌ها و چیزها اشاره می‌نمایند، به عنوان مثال، جورج بوش^۴ و خورشید و ارتباطات معمولاً به خصوصیات^۵ و ویژگی‌ها^۶ اشاره می‌کنند، به عنوان مثال، جورج بوش زنده است. خورشید، داغ است. لیلی پاتر^۷، مادر هری پاتر^۸ است.

در تفسیر زبان طبیعی از "هر ..." یا "همه ..." استفاده می‌کنیم؛ برای مثال، "هر شخصی دارای یک والد است. "، "همه‌ی پرنده‌گان پرواز می‌نمایند. "؛ این مطالب اظهار می‌کنند که همه‌ی اشیا دارای یک ویژگی معین هستند. همچنین از "برخی ..." هم استفاده می‌نماییم؛ برای مثال، "برخی از پرنده‌گان

entity^۱

objects^۲

relations^۳

George Bush^۴

properties^۵

attributes^۶

Lily Potter^۷

Harry Potter^۸

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نمی توانند پرواز نمایند" ، "برخی از افراد از شترنج لذت می بردند" ؛ این مطالب اظهار می کنند که لاقل ، یک شی دارای یک خصوصیت معین می باشد .

یک زبان مرتبه‌ی اوّل شامل موارد زیر است :

۱- یک مجموعه از ثابت‌ها^۱ ؛ که با حروف کوچک مشخص می شوند می باشد ، نظیر a ، b

، c و ... ؛ به طور حسّی ، یک ثابت نام یک شی می باشد .

۲- یک مجموعه از متغیرها ، معمولاً با حروف کوچک مشخص می شوند ، نظیر x ، y ، z و

...

۳- یک مجموعه از سمبل‌های تابعی ، که معمولاً با حروف کوچک نمایش داده می شوند ،

نظیر f ، g ، h و ... ؛ هر سمبل تابعی دارای یک مقدار صحیح و مثبت می باشد .

مجموعه‌ای از واژگان با قوانین^۲ زیر تعریف می شود :

۱. یک ثابت ، یک واژه می باشد

۲. یک متغیر ، یک واژه می باشد

۳. یک عبارت^۳ f(t_1, \dots, t_n) در صورتی یک واژه است که یک سمبل

تابعی و هر t_i واژه باشند

۴. هیچ چیز دیگری واژه نمی باشد

گزاره‌ها^۱

constants^۱

rules^۲

expression^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک زبان مرتبه‌ی اول همچنین شامل یک مجموعه از گزاره‌ها که معمولاً با حروف بزرگ مشخص می‌شوند است، نظیر P ، Q ، R می‌باشد که گزاره‌ها خصوصیات اشیا را نمایش می‌دهند.

رابطه‌ها^۲ و کمیت سنج‌ها (سورها)^۳

رابطه‌های "استاندارد" منطق گزاره‌ای عبارتند از: \rightarrow ، \neg ، \wedge ، \vee و ... دو کمیت سنج عبارتند از: یکی \forall ، بخوانید "برای همه"؛ که سور عمومی^۴ نامیده می‌شود و دیگری \exists ، بخوانید "وجود دارد"؛ که سور وجودی^۵ نام دارد. به طور معمول، سورها با متغیرها و متغیرها با سورها به کار می‌روند.

فرمول مرتبه‌ی اول

در صورتی که P یک سمبول گزاره‌ای دارای ارزش n و t_1, t_2, \dots, t_n واژگان باشند، آن گاه $P(t_1, \dots, t_n)$ یک فرمول منطق مرتبه‌ی اول می‌باشد؛ در صورتی که φ و ψ فرمول باشند، آن گاه $\varphi \vee \psi$ ، $\varphi \wedge \psi$ ، $\neg \varphi$ و $\psi \rightarrow \varphi$ هم فرمول هستند؛ در صورتی که φ یک فرمول باشد و x یک متغیر باشد، آن گاه (φ) و $\forall x(\varphi)$ هم فرمول هستند و فرمول دیگری به غیر از موارد گفته شده وجود ندارد.

متغیرهای آزاد و محدود^۶

ساختار فرمول مرتبه‌ی اول، کامل نمی‌باشد زیرا فرمول‌هایی نظیر $\forall x: P(a)$ و $(\forall x) P(x)$ که x یک متغیر آزاد نامیده می‌شود را مجاز می‌داند.

predicates^۷

Connectives^۸

Quantifiers^۹

universal quantifier^{۱۰}

existential quantifier^{۱۱}

Free and Bound variables^{۱۲}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

معنی یک فرمول منطق مرتبه ی اول

در منطق مرتبه ی اول ، معنی \neg ، معنی \rightarrow مثل معنی آن ها در منطق گزاره ای می باشد ، به عنوان مثال در صورتی که φ و ψ دو فرمول باشند آن گاه $\psi \rightarrow \varphi$ در صورتی درست می باشد که لاقل یکی از φ یا ψ درست باشند . در زمانی که سورها وجود دارند ، صحّت یک فرمول با توجه به دامنه ای سخن تشخیص داده می شود ؟ یک دامنه که معمولاً با حرف D نمایش داده می شود فقط یک مجموعه می باشد ، یک مجموعه می تواند مثل اعداد طبیعی $\{1, 2, 3, \dots\}$ یا مجموعه ای از افراد باشد . به فرمول $\varphi : \forall x : \varphi$ و یک دامنه D توجه نمایید ، فرمول $\varphi : \forall x : \varphi$ در دامنه ای D درست می باشد اگر و فقط اگر $(x | d) \in \varphi$ برای هر $d \in D$ درست باشد ، که $(x | d) \in \varphi$ فرمولی گرفته شده از φ بعد از جایگزین نمودن هر وقوع x در φ با d می باشد . به عنوان مثال ، به فرمول $\forall x : has_chair(x)$ و یک دامنه ای D تشکیل شده از دانش آموزانی که در حال حاضر در OMB 31 هستند توجه نمایید . فرمول $\forall x : has_chair(x)$ در دامنه ای D درست است اگر و فقط اگر $(x | d) \in has_chair(d)$ درست باشد ، حال به فرمول $has_chair(John)$ درست باشد و همین طور این مورد برای هر دانشجوی درون 31 OMB هم درست باشد . حال به فرمول $\exists x : \varphi$ و دامنه ای D توجه نمایید ؛ فرمول $\exists x : \varphi$ در دامنه ای D درست است اگر و فقط اگر $(x | d) \in \varphi$ برای برخی از $d \in D$ درست باشد که $(x | d) \in \varphi$ فرمولی گرفته شده از φ بعد از جایگزین نمودن هر وقوع x در φ با d می باشد به عنوان مثال ، به فرمول $\exists x : s \tan ding(x)$ و یک دامنه ای D مرکب از افرادی که در حال حاضر در OMB 31 هستند توجه نمایید ؛ فرمول $\exists x : s \tan ding(x)$ در دامنه ای D درست است اگر و فقط اگر لاقل یک شخص در OMB 31 ایستاده باشد و بنابراین $standing(Lecturer)$ درست و داریم :

$\exists x : s \tan ding(x)$ درست می باشد . توجه کنید که یک فرمول ممکن است در برخی از دامنه ها درست باشد اما در برخی دیگر نادرست باشد ؛ به عنوان مثال ، به فرمول $(x = y^2) \rightarrow \forall x \exists y : (x = y^2)$ توجه نمایید ؛ این فرمول در دامنه ای اعداد حقیقی مثبت درست می باشد ولی در دامنه ای همه ای اعداد حقیقی این عبارت غلط می باشد ؛ اگر x منفی باشد ، x نمی تواند با y^2 ای برابر باشد ، از آن جایی که $0 \geq y^2$ می باشد .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تعريف - یک فرمول که در همه‌ی دامنه‌ها درست است یک همانگویی^۱ (همیشه درست) می‌باشد.

تعريف - یک فرمول که در همه‌ی دامنه‌ها غلط است یک خلاف گویی^۲ (تناقض) است.

بیان نسبت خانوادگی

منطق مرتبه‌ی اوّل یک زبان ارایه‌ی قدرتمند برای دانش می‌باشد؛ به ارایه‌ی نسبت خانوادگی به زبان انگلیسی توجه نمایید، گزاره‌ها ممکن است شامل موارد زیر باشند:

^r Parent(_, _) ○

^f Child(_, _) ○

^d Grandparent(_, _) ○

^s Sibling(_, _) ○

^v Male(_) ○

^l Female(_) ○

tautology^۱

contradiction^۲

^۳ پدر یا مادر

^۴ بچه، فرزند

^۵ پدر بزرگ یا مادر بزرگ

^۶ برادر یا خواهر

^۷ مرد

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$- = (_ , _) \quad \circ$$

راهی کلی برای تبدیل عبارت های انگلیسی به منطق مرتبه ی اوّل وجود ندارد؛ در این مورد نظریه های^۲ غیردقیق^۳ سادگی^۴ و اختصار^۵ وجود دارد. در زیر جملاتی بیان شده و معادل آن ها در منطق مرتبه ی اوّل هم آورده شده است:

$$\text{"Harry پدر James"} \quad \circ$$

$$\text{Parent(James, Harry) } \wedge \text{Male(James)} \quad \blacksquare$$

$$\text{"Harry مادر Lily"} \quad \circ$$

$$\text{Parent(Lily, Harry) } \wedge \text{Female(Lily)} \quad \blacksquare$$

ارتباطات خانوادگی

$$\bullet \text{ "مردان" و زنان" کاملاً متمایز هستند }$$

$$\forall x : (\text{Male}(x) \leftrightarrow \neg \text{Female}(x)) \quad \circ$$

$$\text{خوانده می شود "p" معادل با } q \leftrightarrow p \text{ است و کوتاه نویسی برای } \circ$$

$$\text{می باشد } ((p \rightarrow q) \wedge (q \rightarrow p))$$

^۱زن

notions^۲

imprecise^۳

simplicity^۴

conciseness^۵

males^۶

females^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- برادر و خواهر^۱ افرادی دقیقاً متمایز هستند که دارای یک والد مشترک می‌باشند

$$\forall x \forall y (Sibling(x, y) \leftrightarrow (\neg=(x, y) \wedge \exists p (Paren(p, x) \wedge Paren(p, y)))) \quad \circ$$

منطق مرتبه‌ی اول و منطق گزاره‌ای

در منطق مرتبه‌ی اول \forall لازم است؛ به فرمول $(P(x) : \forall x)$ توجه نمایید، برای یک دامنه‌ی محدود $D = \{a_1, \dots, a_n\}$ این عبارت دقیقاً در زمانی درست است که $P(a_i)$ برای $1 \leq i \leq n$ درست باشد: $\forall x : P(x) \equiv P(a_1) \wedge \dots \wedge P(a_n)$ و برای یک دامنه‌ی نامحدود $D = \{a_1, \dots, a_n, \dots\}$ این عبارت دقیقاً در زمانی درست است که $P(a_i)$ برای هر $1 \leq i \leq n$ در عبارت زیر درست باشد: $\forall x : P(x) \equiv P(a_1) \wedge \dots \wedge P(a_n) \wedge \dots$ اما یک فرمول نمی‌تواند دارای یک طول بی‌نهایت باشد.

در منطق مرتبه‌ی اول، \exists هم لازم است؛ به فرمول $(P(x) : \exists x)$ توجه نمایید، برای یک دامنه‌ی محدود $D = \{a_1, \dots, a_n\}$ این عبارت دقیقاً در زمانی درست است که $P(a_i)$ برای $1 \leq i \leq n$ درست باشد: $\exists x : P(x) \equiv P(a_1) \vee \dots \vee P(a_n)$. برای یک دامنه‌ی نامحدود $D = \{a_1, \dots, a_n, \dots\}$ این عبارت دقیقاً در زمانی درست است که $P(a_i)$ برای هر $i \in D$ در عبارت زیر درست باشد: $\exists x : P(x) \equiv P(a_1) \vee \dots \vee P(a_n) \vee \dots$ اما یک فرمول نمی‌تواند دارای یک طول بی‌نهایت باشد.

نقیض و سورها

داریم:

$$\neg \forall x : P(x) \equiv \exists x : \neg P(x) \quad \circ$$

$$\forall x : \neg P(x) \equiv \neg \exists x : P(x) \quad \circ$$

^۱siblings

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$\neg \forall x : \neg P(x) \equiv \exists x : P(x) \quad \circ$$

$$\forall x : P(x) \equiv \neg \exists x : \neg P(x) \quad \circ$$

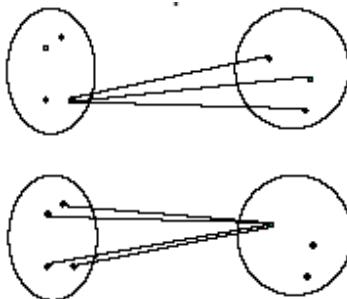
مثال برای نقیض و سورها

به گزاره‌ی $\neg \forall x : male(x)$ در دامنه‌ی دانش آموزان درون OMB 31 توجه نمایید. معنای $\neg \forall x : male(x)$ این است که موردی وجود ندارد که در آن هر کس در OMB 31، مرد باشد؛ این مطلب معادل با این عبارت است که کسی در OMB مرد نمی‌باشد، ظاهراً معادل $\exists x : \neg male(x)$ می‌باشد. سایر معادل های عبارت $\neg \exists x : P(x, y) \equiv \exists y \exists x : P(x, y) \equiv \exists y \forall x : P(x, y) \equiv \forall y \forall x : P(x, y)$ و

$$\neg \exists x \forall y : P(x, y) \equiv \forall y \neg \exists x : P(x, y)$$

به عنوان مثال دیگر، به دامنه‌ی $G1 \cup G2$ از دو گروه افراد G1 و G2 توجه نمایید. Group1() و Group2() گزاره‌های یگانی هستند که نشان می‌دهند که به ترتیب کسی در گروه G1 و G2 هست. هم یک گزاره‌ی دودویی نشان دهنده‌ی این که دو نفر هم‌دیگر را می‌شناسند می‌باشد:

$$\forall x \forall y (Group1(x) \wedge Group2(y) \rightarrow Know(x, y)) \quad \circ$$



$$\forall y \forall x (Group1(x) \wedge Group2(y) \rightarrow Know(x, y))$$

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل یازدهم

منطق مرتبه‌ی

اول

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- چرا منطق مرتبه ی اول ؟
- ظاهر و معنای منطق مرتبه ی اول
- بازی با عبارات
- دنیای wumpus در منطق مرتبه ی اول

موافقین و مخالفین منطق گزاره ای

موافقین

☺ منطق گزاره ای ، اظهاری (اعلانی) است : اجزای گرامری با واقعیات برابرند . ☺ منطق گزاره ای ، به اطلاعات جزئی / فصلی / منفی شده اجازه می دهد (برخلاف بیشتر ساختارهای داده ای

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

و پایگاه های داده ای) .^۱ منطق گزاره ای ، **ترکیبی**^۲ می باشد ؛ به عنوان مثال ، مفهوم $B_{1,1} \wedge P_{1,2}$ از $B_{1,1}$ و از $P_{1,2}$ به وجود می آید .^۳ مفهوم در منطق گزاره ای ، **مستقل از متن**^۴ است . (برخلاف زبان طبیعی ، که مفهوم وابسته به متن می باشد) .

مخالفین

نمط گزاره ای محدودیت زیادی در قدرت بیان دارد (برخلاف زبان طبیعی) . مثال ، ما نمی توانیم بگوییم " چاله ها باعث هوای مطلوب در خانه های مجاور می شوند " به جز با نوشتن یک عبارت برای هر خانه .

منطق مرتبه ای اول - از آنجایی که منطق گزاره ای ، محتویات واقعی دنیا را به خود می گیرد ، منطق گزاره ای (مثل زبان طبیعی) محتویات جهان را به خود می گیرد .

اشیاء : مردم ، خانه ها ، اعداد ، تئوری ها ، رونالد مک دونالد^۵ ، رنگ ها ، بازی های بیسبال^۶ ، جنگ ها ، قرن ها و می باشند .

وابستگی ها (ارتباطات یا روابط) : قرمز ، گرد^۷ ، ساختگی^۸ (جعلی یا قلابی) ، اول^۹ (عده) ، ساختمان چند طبقه^{۱۰} و می باشند .

compositional^۱

context-independent^۲

Ronald McDonald^۳

baseball^۴

round^۵

bogus^۶

prime^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

براذر ... ، بزرگ تر از ، درون ... ، قطعه ای از ... ، به رنگ ، اتفاق افتاده بعد از ، داشتن ، آمدن میان و می باشند .

عملکردها ^۱(تابع ها) : پدر ... ، بهترین دوست ، تصدی ... ، یکی بیش تر از ، پایان می باشند .

منطق در حالت کلی

معرفت شناسی ^۴	هستی شناسی ^۳	زبان
درست / غلط / ناشناخته	واقعیات	منطق گزاره ای
درست / غلط / ناشناخته	واقعیات ، اشیاء ، رابطه ها	منطق مرتبه ای اوّل
درست / غلط / ناشناخته	واقعیات ، اشیاء ، رابطه ها ، زمان ها	منطق زمانی ^۵
اندازه ای اعتقاد ^۶	واقعیات	تیوری احتمال ^۷
مقدار فاصله ای شناخته شده ^{۱۰}	واقعیات + درجه ای درستی ^۹	منطق فازی ^۸

multistoried ^۱

functions ^۲

Ontological Commitment ^۳

Epistemological Commitment ^۴

Temporal logic ^۵

Probability logic ^۶

degree of belief ^۷

Fuzzy logic ^۸

degree of truth ^۹

known interval value ^{۱۰}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ظاهر منطق مرتبه‌ی اول - عناصر اساسی عبارتند از :

- ثابت‌ها مثل : King, John, 2, UCB, Koblenz, C, ...

- گزاره‌ها مثل : Brother, >, =, ...

- توابع مثل : Sqrt, LeftLegOf, ...

- متغیرها مثل : x, y, a, b, ...

- رابط‌ها مثل : $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

- تساوی : =

- کمیت سنج‌ها (سورها) : \exists, \forall

عبارات اتمیک^۱

- عبارت اتمیک $term_1 = term_2$ یا $predicate(term_1, \dots, term_n)$ =

. واژه $function(term_1, \dots, term_n)$ = که یا ثابت است و یا متغیر می‌باشد.

: مثال

**Brother(KingJohn,RichardTheLionheart) >
(Length(LeftLegOf(Richard)),Length(LeftLegOf(KingJohn)))**

Atomic^۱

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در عبارت $\text{Brother} \cdot \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart})$ گزاره،
KingJohn ثابت، RichardTheLionheart ثابت، KingJohn واژه،
RichardTheLionheart واژه و کل عبارت یک عبارت اتمیک می باشد.

$> (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn})))$

در عبارت فوق، Length تابع، LeftLegOf گزاره تابع،
 Length ثابت، KingJohn تابع، LeftLegOf تابع، Length ثابت،
 $\text{Length}(\text{LeftLegOf}(\text{KingJohn}))$ واژه و $\text{Length}(\text{LeftLegOf}(\text{Richard}))$ واژه و
کل عبارت $>$
 $(\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn})))$
)، عبارت اتمیک می باشد.

عبارات پیچیده - عبارات پیچیده، همانند منطق گزاره ای، از عبارات اتمیک با استفاده از رابط ها ساخته می شوند، مثل:

$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$$

مثال:

$\text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn})$

$> (1,2) \vee \leq (1,2)$

$> (1,2) \wedge \neg > (1,2)$

در عبارت فوق، Sibling و واژه Richard ، گزاره می باشد. KingJohn واژه است. Sibling طرف راست گزاره است. Richard طرف راست واژه است. KingJohn است. Sibling طرف راست واژه است. $\text{Sibling}(\text{KingJohn}, \text{Richard})$ ، عبارت اتمیک است. $\text{Sibling}(\text{Richard}, \text{KingJohn})$ عبارت اتمیک است و کل عبارت، عبارت پیچیده می باشد.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

صحّت در منطق مرتبه‌ی اول - عبارت‌ها با رعایت یک مدل و یک تفسیر^۱ ،

درست می‌باشند. مدل، شامل اشیای بزرگ‌تر یا مساوی یک (عناصر دامنه) و رابط‌های میان آن‌ها می‌باشد ($\text{contains} \geq 1$). تفسیر موارد مراجعه را برای موارد زیر تایین می‌نماید:

سمبل‌های ثابت^۲ \longleftrightarrow اشیاء

سمبل‌های گزاره‌ای^۳ \longleftrightarrow رابط‌ها

سمبل‌های تابعی^۴ \longleftrightarrow رابط‌های تابعی^۵

یک عبارت اتمیک^۶ ($\text{predicate}(term_1, \dots, term_n)$) در صورتی درست است که اشیاء ارجاع شده به آن توسط $term_1, \dots, term_n$ با ارتباط‌های ارجاع شده توسط گزاره در ارتباط باشند.

مثالی برای مدل‌های منطق مرتبه‌ی اول

interpretation^۱

constant symbols^۲

predicate symbols^۳

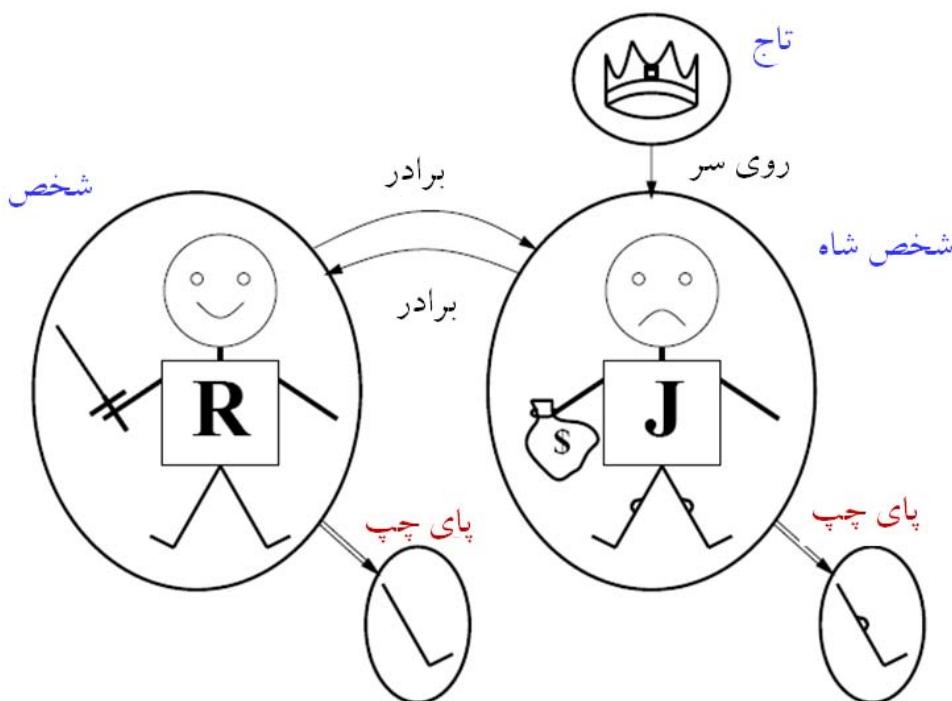
function symbols^۴

functional relations^۵

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



مثال صحت - به تفسیری به صورت زیر توجه نمایید:

ریچارد^۱ ← فردی دلیر^۲ است

جان^۳ ← پادشاهی بد

برادر ← ارتباط برادری^۴

Richard^۱

Lionheart^۲

John^۳

brotherhood relation^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تحت این تفسیر، Brother(Richard,John) فقط در مورد ریچارد فردی دلیر

است درست است و جان پادشاهی بد است درست است، که در این مدل دارای ارتباط برادری هستند.

سور عمومی - همه در بر کلی ^۱ زرنگ هستند: $\forall x : At(x, Berkeley) \Rightarrow Smart(x)$

$\forall x$ در یک مدل m درست است در صورتی که P به ازای هر x ممکن در مدل درست باشد.

$(At(KingJohn, Berkeley) \Rightarrow Smart(KingJohn)) \wedge (At(Richard, Berkeley) \Rightarrow Smart(Richard)) \wedge (At(Berkeley, Berkeley) \Rightarrow Smart(Berkeley)) \wedge \dots$

$\forall x : At(x, Berkeley) \wedge Smart(x)$ معنای این عبارت این است که " همه در بر کلی هستند و همه با هوش هستند ".

تعیین خواص وجودی ^۲

شخصی در با هوش است را به این صورت می توانیم بیان نماییم:

که با مثال های زیر برابر است: $\exists x At(x, S tan ford) \wedge Smart(x)$

$(At(KingJohn, S tan ford) \wedge Smart(KingJohn))$
 $\vee (At(Richard, S tan ford) \wedge Smart(Richard))$
 $\vee (At(S tan ford, S tan ford) \wedge Smart(S tan ford))$

خصوصیات کمیت سنج ها (سورها)

Berkely ^۱
smart ^۱
Existential quantification ^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$\forall x \forall y$ مثل $\exists x \exists y$ می باشد . $\exists x \forall y$ مثل $\forall y \exists x$ می باشد . $\forall x \forall y$ مثل $\forall y \forall x$ نمی باشد .

" $\exists x \forall y Loves(x, y)$ " یعنی : " شخصی وجود دارد که همه را در جهان دوست دارد "

" $\forall y \exists x Loves(x, y)$ " یعنی : " همه در جهان با لاقل یک فرد دوست می شوند "

دوگانگی کمیت سنج ها^۱ - هر چیزی می تواند با استفاده از دیگری بیان شود :

$$\forall x Likes(x, IceCream) \equiv \neg \exists x \neg Likes(x, IceCream)$$

$$\exists x Likes(x, Broccoli) \equiv \neg \forall x \neg Likes(x, Broccoli)$$

بازی با عبارات

برادران ، هم نژاد ^۲ هستند را به این صورت بیان می نماییم :

$$\forall x, y Brother(x, y) \Rightarrow Sibling(x, y)$$

"برادری (هم نژادی)" [رابطه ای]^۳ متقارن ^۴ است ؛ یعنی :

$$\forall x, y : Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

مادر یکی والده ی یکی دیگر است :

$$\forall x, y : Mother(x, y) \Leftrightarrow (Female(x) \wedge Parent(x, y))$$

Quantifier duality ^۱

Sibling ^۲

مترجم ^۳

symmetric ^۴

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اولین عموزاده بچه ای هم نژاد با والد است:

$$\forall x, y : FirstCousin(x, y) \Leftrightarrow \exists p, psParent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



و پایگاه های داده ای \mathcal{D} . متعلق گزاره ای، توکیپی (نمی باشد) به عنوان مثال، مفهوم $P_{1,2} \wedge P_{1,1}$ از $P_{1,1}$ و از $P_{1,2}$ به وجود نمی آید. مفهوم در منطق گزاره ای بدلستقل از هست است. (برخلاف زبان طبیعی، که مفهوم وابسته به متن نمی باشد).

مخالفین

فصل دوازدهم

استنتاج در منطق

مرتبه‌ی اول^۱

Inference in First-order Logic^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- کاهنده‌ی استنتاج منطق مرتبه‌ی اول به استنتاج گزاره‌ای
- یک شکلی^۱
- طرز عمل کلی^۲
- زنجیره‌ی مستقیم و معکوس
- برنامه نویسی منطقی^۳
- تحلیل^۴

unification^۱

Generalized Modus Ponens^۲

Logic programming^۳

Resolution^۴

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تاریخچه‌ی مختصری از استدلال

در سال ۴۵۰ قبل از میلاد مسیح، پیروان فلسفه‌ی رواقیون^۱، منطق‌گزاره‌ای و (شاید) استدلال را بنیان نهادند. | در سال ۳۲۲ قبل از میلاد مسیح، ارسسطو، قیاس منطقی^۲ (قوانین استدلال) و سورها را بیان کرد. | در سال ۱۵۶۵، کاردانو^۳، تیوری احتمال^۴ (منطق‌گزاره‌ای^۵ + نامعلومی^۶) را به وجود آورد. | در سال ۱۸۴۷، بول^۷، دوباره منطق‌گزاره‌ای را بررسی کرد. | سال ۱۸۷۹، فرگ^۸، منطق مرتبه‌ی اوّل را به وجود آورد. | در سال ۱۹۲۲، ویتجنستین^۹، اثبات توسط جدول صحت را بررسی کرد. | در سال ۱۹۳۰، گدل^{۱۰}، گفت: الگوریتم کاملی برای منطق مرتبه‌ی اوّل وجود دارد. | در سال ۱۹۳۰، هربرند^{۱۱}، الگوریتم کاملی برای منطق مرتبه‌ی اوّل به وجود آورد. | در سال ۱۹۳۱، گدل گفت: الگوریتم کاملی برای محاسبه وجود

Stoics^۱

syllogisms^۲

Cardano^۳

probability theory^۴

propositional logic^۵

uncertainty^۶

Boole^۷

Frege^۸

Wittgenstein^۹

Godel^{۱۰}

Herbrand^{۱۱}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ندارد. | در سال ۱۹۶۰، دیویس-پوتنم^۱، الگوریتمی کاربردی برای منطق گزاره ای ارایه کرد. | در سال ۱۹۶۵، راینسن^۲ الگوریتمی کاربردی برای منطق مرتبه ای اوّل - تحلیل به وجود آورد.

استنتاج در منطق مرتبه ای اوّل

می توانیم انواع استنتاج را با منطق مرتبه ای اوّل داشته باشیم.

برداشتن سورها

نمونه سازی (معرفی) عمومی^۳

ما همیشه می توانیم یک واژه ای زمینه را با یک متغیر جانشین نماییم مثلا در مثال زیر؛
 $\forall x LivesIn(x, Springfield) \Rightarrow knows(x, Homer)$
 می باشد، مامی توانیم $x=Bart$ را جایگزین نماییم:

$$\frac{\forall v: \alpha}{Subst(\{v/g\}, \alpha)} \text{ برای هر متغیر } v \text{ و} \quad LivesIn(Bart, Springfield) \Rightarrow knows(Bart, Homer)$$

 واژه ای زمینه ای g برقرار می باشد.

مثال، نتیجه های $\forall x : King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ عبارتند از:

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$

$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$

$$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$$

⋮

Davis/Putnam^۱

Rabinson^۲

Universal instantiation (UI)^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نمونه سازی (معرفی) وجودی^۱

در مورد $\exists x : LivesIn(x, Springfield) \wedge knows(x, Homer)$ ، ما می دانیم که این باید برای لاقل یک شی، درست باشد. بایاید این شی را k بنامیم؛ داریم، $LivesIn(k, Springfield) \wedge knows(k, Homer)$ که در این مورد، یک ثابت اسکلِم نامیده می شود. پس، برای هر عبارت a و متغیر v و سمبول ثابت k که در جای دیگری در پایگاه دانش وجود ندارد داریم:

$$\frac{\exists v\alpha}{Subst(\{v/k\}, \alpha)}$$

مثال: نتیجه های $\exists x Crown(x) \wedge OnHead(x, John)$ **بـ صـورـت**

تواند باشد که C_1 به وجود آمده از یک سمبول ثابت جدید به نام $Crown(C_1) \wedge OnHead(C_1, John)$ ثابت اسکلِم^۲ می باشد.

مثالی دیگر: از $d(e^y)/dy = e^y$ مـا نـتـیـجـهـ مـیـ گـیرـیـمـ،ـ $d(x^y)/dy = x^y$ (ای کـهـ بـهـ وجود آمده است یک سمبول ثابت جدید مـیـ باـشـدـ).

گزاره بندی^۳

ما می توانیم هر عبارت دارای سور وجودی را با یک نوع اسکلمایز شده جایگزین نماییم. برای عبارت های با سور عمومی، می توانید هر جایگزین ممکن را جایگزین نمایید؛ که این کار به ما اجازه می

Existential instantiation (EI)^۱

Skolem constant^۲

propositionalization^۳

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دهد که از قانون های استنتاج گزاره ای استفاده نماییم . البته این کار خیلی ناکارآمد می باشد ؛ تا سال ۱۹۶۰ از همین روش استفاده می کردند .

تبديل به استدلال گزاره ای

تصور نمایید که پایگاه دانش فقط شامل عبارات زیر باشد :

$$\forall x: \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x), \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Brother}(\text{Richard}, \text{John})$$

معرفی عبارت عمومی در همه ی موارد ممکن است و داریم :

$$\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}), \text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}), \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Brother}(\text{Richard}, \text{John})$$

که پایگاه دانش جدید ، گزاره ای می باشد ، سمبول های گزاره ای عبارتند از :

$\text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Evil}(\text{John}), \text{King}(\text{Richard})$

و

تبديل

- **ادعا** : یک عبارت زمینه به وسیله ی پایگاه دانش جدید در صورتی که به وسیله ی پایگاه دانش اصلی به وجود آمده باشد ، موجود می شود .

- **ادعا** : هر پایگاه دانش منطق مرتبه ی اوّل می تواند برای حفظ موجودیت ، به صورت گزاره ای بیان شود .

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- **ایده:** پایگاه دانش را به صورت گزاره ای و پرس و جو بیان نمایید، تحلیل را به کار ببرید و نتایج را برگردانید.

- **مسئله:** با سمبول های توابع ، واژگان زمینه به صورت نامحدود وجود خواهند داشت .

مثال : Father(Father(Father(John)))

قضیه ی هربوند (۱۹۳۰) : در صورتی که عبارت α توسط یک پایگاه دانش منطق مرتبه ی اوّل به وجود بیاید ، توسط یک زیرمجموعه ی محدود از پایگاه دانش های گزاره ای موجود می شود .

ایده : برای $n = 0$ تا بینهایت کارهای زیر را انجام بده

یک KB (پایگاه دانش) گزاره ای با عمق n بسازید

بررسی نمایید که آیا α توسط این KB تولید می شود

قضیه ی تورینگ و چورچ^۱ (۱۹۳۶) : به وجود آمدن^۲ در منطق مرتبه ی اوّل تقریباً قابل تصمیم گیری^۳ می باشد .

ایرادهای گزاره ها

گزاره ، برای تولید تعداد زیادی عبارات که به هم نامربوط هم هستند به کار می رود .

مثال :

Turing,Church^۱
entailment^۲
semidecidable^۳

مترجم: شهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی

$\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$, $\text{King}(\text{John})$, $\forall y \text{ Greedy}(y)$, $\text{Brother}(\text{Richard},\text{John})$

حاصل به نظر می رسد که Evil(John) باشد ، اما گزاره بندی ، تعداد زیادی واقعیات نظیر Greedy(Richard) که نامربوط هستند را به وجود می آورد . با گزاره های k-تایی p و n های ثابت ، $p.n^k$ معرفی وجود خواهد داشت . با سمبول های تابعی ، معرفی های با مقادیر به مراتب بدتری وجود خواهد داشت !

یک شکلی

یک شکلی ، این است که ما فقط می خواهیم جایگزین هایی برای عباراتی که به ما کمک می کنند چیزهایی را ثابت نماییم پیدا کنیم . برای مثال ، اگر ما بدانیم :

$\forall x : \text{LivesIn}(x, \text{Springfield}) \wedge \text{WorksAt}(x, \text{PowerPlant}) \Rightarrow \text{know}(x, \text{Homer})$

$\forall y : \text{LivesIn}(y, \text{Springfield})$

$\text{WorksAt}(\text{MrSmithers}, \text{PowerPlant})$

آن گاه ، ما باید قادر باشیم مستقیما نتیجه بگیریم knows(MrSmithers,Homer) با جایگزینی :
. {x/MrSmithers,y/MrSmithers}

پس در مثال قبلی ، در صورتی که بتوانیم یک زیر وضعیت θ را پیدا کنیم ، می توانیم بلا فاصله استدلال نماییم (x) King و (y) Greedy و (x) King با (y) Greedy مطابقت می کنند .

$$\alpha\theta = \beta\theta \wedge \theta = \{x/\text{John}, y/\text{John}\} , \text{Unify}(\alpha, \beta) = \theta$$

p	q	θ
---	---	----------

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	fail

استاندارد کردن از روی هم افتادن^۱ متغیرها جلوگیری می کند، مثال: $Knows(z_{17}, OJ)$

طرز عمل کلی^۲

ایده‌ی اساسی این است که فرض کنیم یک استنتاج به شکل $P_1 \wedge P_2 \wedge \dots \wedge P_i \Rightarrow Q$ را داریم و عبارات به صورت P'_1, P'_2, \dots, P'_i می باشند، اگر مجموعه ای از جانشین‌ها به صورت $P'_1 = P'_1, P'_2 = P'_2, \dots, P'_n = P'_n$ وجود داشته باشند، سپس ما می توانیم جانشین را به کار بگیریم و روش طرز عمل کلی را برای به دست آوردن Q به کار بگیریم. این تکنیک استفاده از جانشین‌ها برای جست عبارات بالا، برای به دست آوردن استنتاج، یک شکلی نام دارد. پردازه‌ی استنتاج ما حالا پیدا کردن جانشین‌هایی که به ما اجازه خواهند داد عبارات جدید را به وجود آوریم می باشد.

الگوریتم یگانی: دو عبارت را می پذیرد و مجموعه ای از جانشین‌هایی که عبارات را به صورت

یکتا می سازند برمی گرداند.

overlap^۱

Generalized Modus Ponens (GMP)^۲

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$\{x/Homer\}$ ، WorksAt(x,PowerPlant), WorksAt(Homer,PowerPlant) را تولید می نماید.

$\{x/Homer,y/PowerPlant\}$ ، WorksAt(x,PowerPlant), WorksAt(Homer,y) می نماید.

، WorksAt(x,PowerPlant), WorksAt(FatherOf(Bart),y) $\{x/FatherOf(Bart),y/PowerPlant\}$ را تولید می کند.

Homer غلط می باشد – x نمی تواند هم به WorksAt(x,PowerPlant), WorksAt(Homer,x) اشاره کند و هم به PowerPlant.

آخرین عبارت یک اشتباه است؛ فقط به خاطر این که ما خواسته ایم x را در هر دو عبارت استفاده نماییم. ما می توانیم x را با یک متغیر منحصر به فرد (x_{21}) که یک عبارت است جایگزین نماییم که این کار استانداردسازی نامیده می شود. اگر بیش از یک جانشین وجود داشته باشد که بتوانند دو عبارت که به نظر می رسد یکسان هستند را بسازند چطور؟؛ مثلا:

Sibling(y,z) و Sibling(Bart,x) می تواند $\{x/Bart,y/Bart,z/Bart\}$ یا $\{Bart/y,x/z\}$ را تولید نماید که اوّلین یکسان سازی، کلی تراز دومی است - چون الزام های کم تری را سبب می شود. پس، ما می خواهیم کلی ترین یکسان کننده ها را در زمانی که استنتاج را انجام می دهیم پیدا نماییم.

الگوریتم یکسان سازی

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برای یگانه کردن دو عبارت ، به صورت بازگشته جلو بروید . اگر هر دو عبارت تک متغیری است ، یک یگانه سازی که متغیر را به یک ثابت متصل می کند را پیدا نمایید . در غیر این صورت یکانگی درون واژه ای اوّل را ، که توسط باقی مانده ای هر عبارت دنبال شده است را صدا بزنید .

مثال :

و $Sibling(x, Bart)$ و $Sibling(Lisa, y)$ و $Sibling(x, Bart) \wedge PlaysSax(x)$ که مامی توانیم $x/Lisa, y/Bart$ را با $\{x/Lisa, y/Bart\}$ یگانه نماییم . پردازه ای پیدا کردن مجموعه ای کاملی از جانشین ها ، یک پردازه ای جستجو می باشد که وضعیت ، لیستی از جانشین ها می باشد و تابع جانشین ، لیستی از یکسان سازی های بالقوه و لیست های جانشین جدید است . پس در مورد یکسان سازی داریم :

$$\frac{p_1, p_2, \dots, p_n, (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta}$$

در جایی که $p_i\theta = p_i\theta$ برای همه i ها برقرار باشد . به عنوان مثال ، اگر p_1 ، King(John) و p_2 ، Greedy(x) می باشد . $p_1\theta = p_1\theta$ برای همه i ها برقرار باشد . به عنوان مثال ، اگر p_1 ، King(x) و p_2 ، Greedy(y) می باشد . $p_1\theta = p_1\theta$ برای همه i ها برقرار باشد . $p_2\theta = p_2\theta$ برای همه i ها برقرار باشد . $(p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge p_2\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$ می باشد و $\{x/John, y/John\}$ می باشد .

روش طرز عمل کلی ، با شرط های نامحدود پایگاه دانش استفاده می شود (دقیقاً با یک لفظ مثبت) . تمام متغیرها ، کمیت های عمومی را به خود می گیرند .

اثبات روش طرز عمل کلی - باید نشان بدھیم که : $p_1, \dots, p_n, (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models q\theta = p_i\theta$ برای همه i ها موجود ، $p_i\theta$ باشد .

مشروط بر این که برای همه i ها موجود ، $p_i\theta$ باشد .

اثبات : با استفاده از نمونه سازی عمومی (UI) برای هر شرط نامحدود p داریم

$$1.(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$$

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$2. p_1^{'}, \dots, p_n^{'} |= p_1^{'} \wedge \dots \wedge p_n^{'} |= p_1^{'} \theta \wedge \dots \wedge p_n^{'} \theta$$

۳. از ۱ و ۲ نتیجه می شود که $q\theta$ توسط روش های معمولی دنبال می شود.

مثال : پایگاه دانش - قانون می گوید که این یک جنایت است که یک آمریکایی به ملت های دشمن ، اسلحه بفروشد . کشور نونو ^۱ یک دشمن آمریکا می باشد و دارای تعدادی موشک می باشد و تمام این موشک ها توسط کلنل وست ^۲ که یک آمریکایی است به این کشور فروخته شده است . ثابت کنید که کلنل وست یک جنایت کار می باشد .

... برای یک آمریکایی این جنایت است که به ملل دشمن اسلحه بفروشد :

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

کشور نونو دارای تعدادی موشک می باشد ... توجه کنید :

$$\exists x \text{Owns}(\text{Nono}, x) \wedge \text{Missle}(x) : \text{Missle}(M_1), \text{Owns}(\text{Nono}, M_1)$$

... تمام این موشک ها توسط کلنل وست فروخته شده اند .

$$\forall x \text{Missle}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$$

موشک ها اسلحه هستند : $\text{Weapon}(x) \Rightarrow \text{Missle}(x)$

$$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$$

وست آمریکایی است ... $\text{American}(\text{West})$

Nono ^۱

Colonel West ^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

کشور نونو یک دشمن آمریکا است : Enemy(Nono,America)

زنجیره‌ی مستقیم

ایده‌ی اساسی این است که ، با واقعیات و قوانین شروع کنید (استنباط‌ها) و دایما روش طرز عمل کلی (GMP) را به کار ببرید تا واقعیات جدید بتوانند به وجود بیايند .

الگوریتم زنجیره‌ی مستقیم (به بیان دیگر)

تابع $FOL-FC-Ask(KB,\alpha)$ یک تعویض یا غلط را بر می‌گرداند

مادرامی که new خالی است کارهای زیر را انجام بده

$$new \leftarrow \{\}$$

برای هر عبارت r در KB موارد زیر را انجام بده

$$(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow Standardize-Apart(r)$$

برای هر θ در KB $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$ برای برخی از p_1, \dots, p_n در

$$q' \leftarrow Subst(\theta, q)$$

در صورتی که q' تغییر نام یک عبارت موجود در KB یا new نبود کارهای زیر را انجام بده

را به new اضافه کن q'

$$\phi \leftarrow Unify(q', \alpha)$$

در صورتی که ϕ fail نمی‌باشد ، ϕ را برگرددان

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

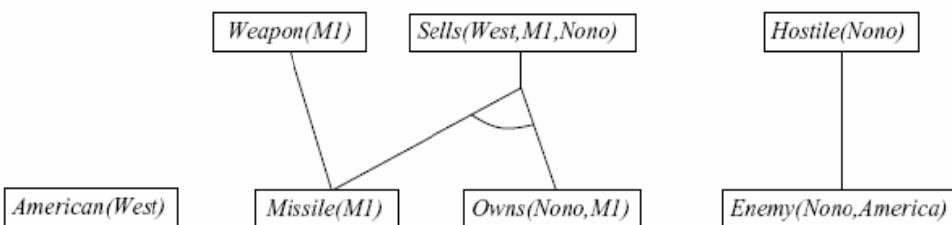
را به KB اضافه کن new

را برگردان false

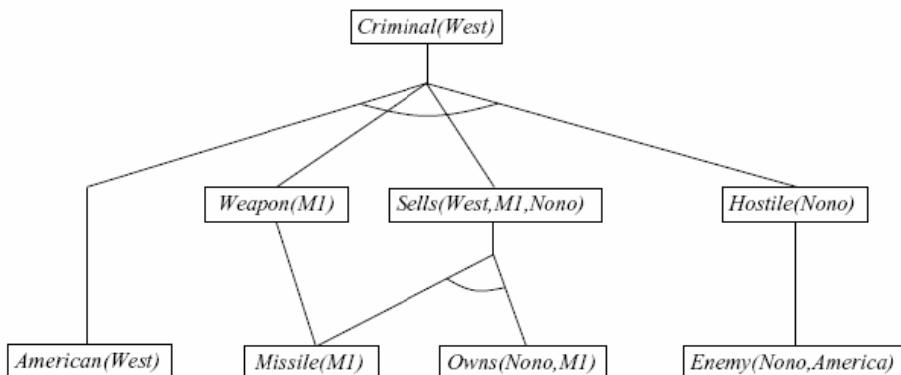
اثبات به روش زنجیره‌ی مستقیم

American(West) *Missile(MI)* *Owns(Nono, MI)* *Enemy(Nono, America)*

گام بعدی :



گام بعدی :



خصوصیات زنجیره‌ی مستقیم

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



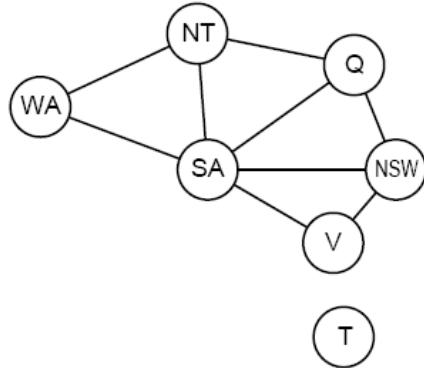
هوش مصنوعی

زنجیره‌ی مستقیم، بی عیب می‌باشد، چون که از روش طرز عمل کلی (GMP) استفاده می‌کند.

برای شرط‌های محدود، زنجیره‌ی مستقیم، کامل می‌باشد. خیلی شبیه به جستجوی اوّل-بهترین عمل می‌کند. این الگوریتم اساسی، خیلی هم کارآمد نیست، چون: اوّل، همه‌ی یکی کننده‌های ممکن را پیدا می‌کند ولی گران است. دوم، هر قانون، در هر تکرار، مجدداً چک می‌شود و سوم، این که واقعیاتی که به هدف منجر نمی‌شوند، تولید می‌شوند.

زنجیره‌ی مستقیم به طور گسترده‌ای در **پایگاه داده‌های استقرایی یا استنتاجی یا قیاسی**^۱ استفاده می‌شود. یک سیستم پایگاه داده‌ی قیاسی، سیستمی پایگاه داده‌ای است که می‌تواند قیاس را براساس قوانین و واقعیات نگهداری شده پایگاه دانش قیاسی انجام دهد. سیستم‌های پایگاه دانش قیاسی، اساساً به قوانین و واقعیات رسیدگی می‌کنند، از یک زبان اعلانی (نظیر پرولوگ) برای مشخص کردن این قوانین و واقعیات استفاده نمایند و از یک موتور استنتاج که می‌تواند قوانین و واقعیات جدید را از آن‌ها بیکاری شده‌اند قیاس نماید.^۲

مثال تطبیق دهنده‌ی سخت^۳



deductive databases^۱

en.wikipedia.org/wiki/Deductive_database^۲

hard matching^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$Diff(wa, nt) \wedge Diff(wa, sa) \wedge Diff(nt, q) \wedge Diff(nt, sa) \wedge Diff(q, nsw) \wedge Diff(q, sa) \wedge Diff(nsw, v)$
 $\wedge Diff(nsw, sa) \wedge Diff(v, sa) \Rightarrow Colorable$

Diff(Red,Blue)	Diff(Red,Green)	Diff(Green,Red)
Diff(Green,Blue)	Diff(Blue,Red)	Diff(Blue,Green)

در صورتی که CSP راه حلی برای CSP های شامل 3SAT در مورد خاص Colorable () داشته باشد استنتاج می شود ، با وجودی که تطبیق به صورت NP-hard می باشد .

زنجیره‌ی معکوس

این روش ، به طرف معکوس ، از هدف به طرف واقعیاتی که باید برای هدف نشان داده شوند حرکت می کند . از روش طرز عمل کلی ، به صورت معکوس برای تمرکز دادن جستجو روی پیدا کردن شرط هایی که می توانند منجر به هدف شوند استفاده می کند ؛ همچنین از شرط های نامحدود استفاده می نماید و جستجو به روش اوّل - عمق جلو می رود .

الگوریتم زنجیره‌ی معکوس

تابع () FOL-BC-Ask(KB,goals,θ) مجموعه ای از تعویض ها را برمی گرداند

ورودی ها : KB ، که یک پایگاه دانش می باشد

goals ، لیستی از پیوندهای شکل دهنده‌ی یک صفت (θ قبلا به کار گرفته شده است)

{ } ، تعویض جاری ، به صورت اوّلیه ، تعویض خالی می باشد { }

متغیرهای محلی : answers ، که مجموعه ای از تعویض ها می باشد ، و به صورت اوّلیه تهی ({}) می باشد .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در صورتی که q تهی است مجموعه $\{\theta\}$ را برگردان

$$q' \leftarrow Subst(\theta, First(goals))$$

برای هر عبارت r موجود در KB

در جایی که $\theta' \leftarrow Unify(q, q')$ و $Stan dardize - Apart(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$ برقرار می باشد.

$$new-goals \leftarrow [p_1, \dots, p_n \mid Rest(goals)]$$

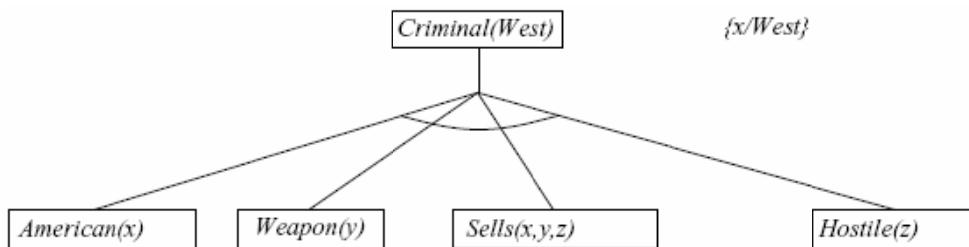
$$answers \leftarrow FOL-BC-Ask(KB, new-goals, Compose(\theta', \theta)) \cup answers$$

answers را برگردان

مثال زنجیره‌ی معکوس

Criminal(West)

گام بعدی:

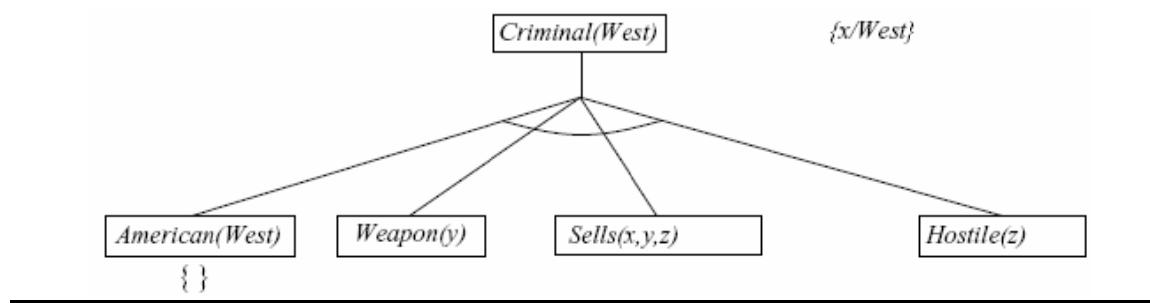


گام بعدی:

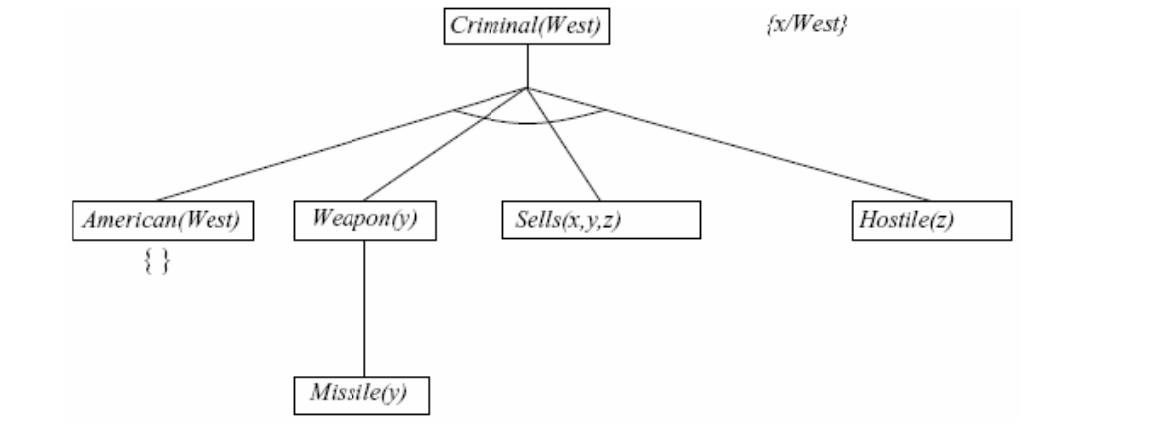
مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



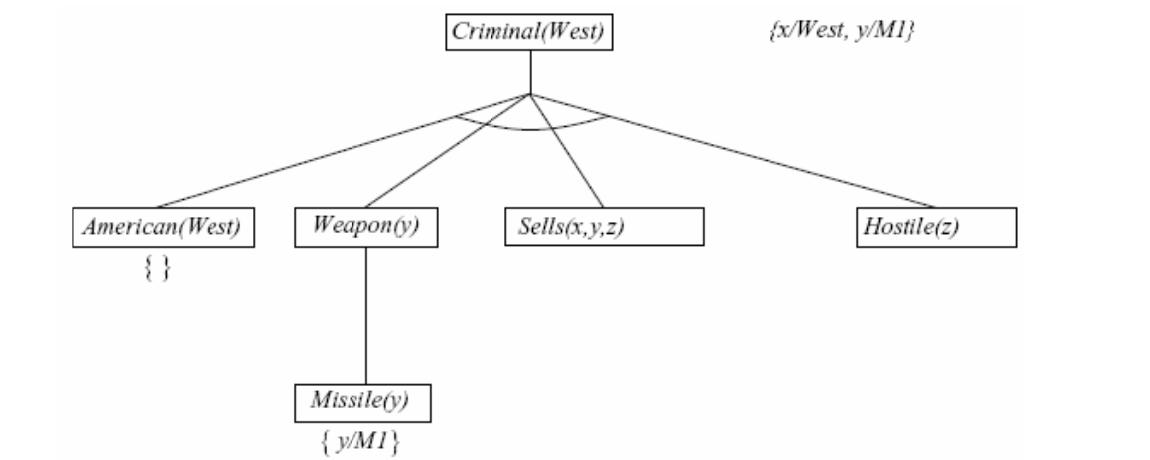
هوش مصنوعی



گام بعدی:



گام بعدی:

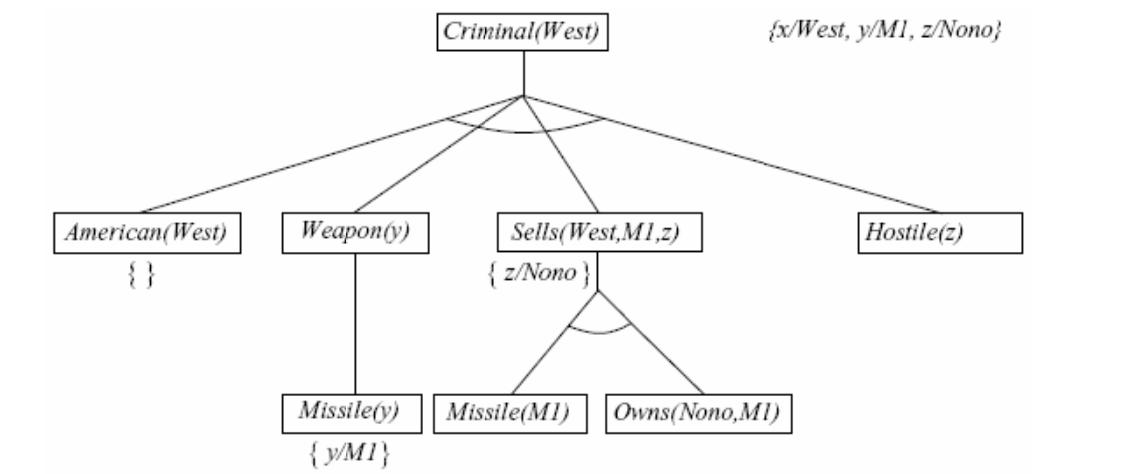


مترجم: سهرا ب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

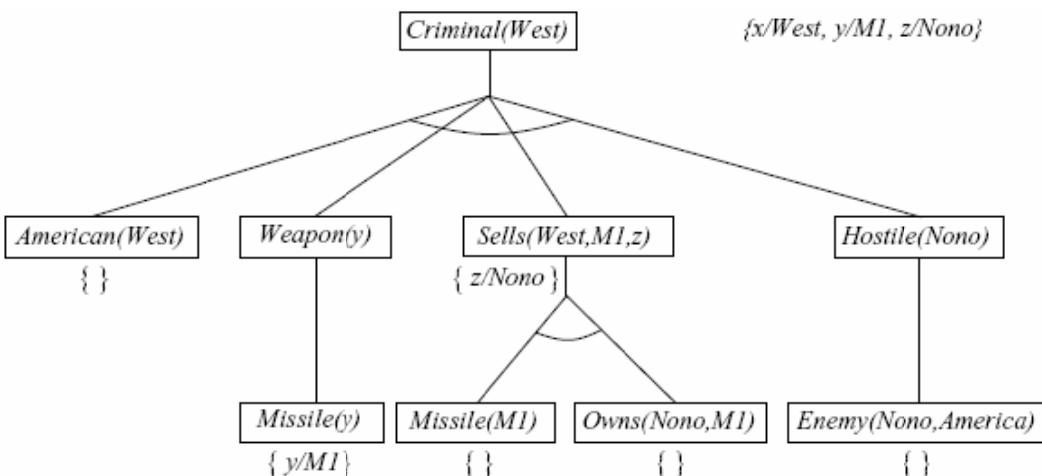


هوش مصنوعی

گام بعدی:



گام بعدی:



خصوصیات زنجیره‌ی معکوس

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

زنگیره‌ی معکوس از روش اوّل – عمق استفاده می‌کند؛ این به این معنی است که از وضعیت‌های تکرار شده رنج می‌برد. همچنین این روش کامل نمی‌باشد و می‌تواند برای سیستم‌های برنامایی پرس و جو^۱ خیلی موثر باشد در ضمن، این روش برای برنامه‌نویسی منطقی^۲ به طور گسترده‌ای استفاده می‌شود.

تحلیل: خلاصه

تحلیل را در منطق گزاره‌ای به یاد بیاورید: $(A \vee C) \wedge (\neg A \vee B) \Rightarrow (B \vee C)$ ؛ تحلیل در منطق مرتبه‌ی اوّل به طریقی مشابه عمل می‌کند. توجه کنید که لازم است عبارات به فرم CNF باشند.

تبديل به صورت فرمال CNF - هر کس که همه‌ی حیوانات را دوست می‌دارد کسی او را دوست می‌دارد:

$$\forall x[\forall y Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y Loves(y, x)]$$

۱. استلزم^۳‌ها را حذف کنید

$$\forall x[\neg\forall y \neg Animal(y) \vee Loves(x, y)] \vee [\exists y Loves(y, x)]$$

۲. \rightarrow را به درون حرکت دهید:

query – based systems^۱
logic programming^۲

³ implication

در کامپیوتر، عملی منطقی با ساختار
IF-THEN
اگر A,B درست باشند، آن‌گاه تابع AND این دو هم درست است

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$\neg\forall x, p \equiv \exists x\neg p, \neg\exists x, p \equiv \forall x\neg p :$
 $\forall x[\exists y\neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y Loves(y, x)]$
 $\forall x[\exists y\neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y Loves(y, x)]$
 $\forall x[\exists y Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y Loves(y, x)]$

متغیرهای استاندارد : هر کمیت ، باید از یک متغیر جدا استفاده نماید ؛ ۳

$\forall x[\exists y Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z Loves(z, x)]$ عبارت های وجودی را اسکلمایز نمایید ۴

$\forall x[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$ سورهای عمومی را رها نمایید ۵

$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$ \wedge را روی \vee توزیع نمایید ۶

$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

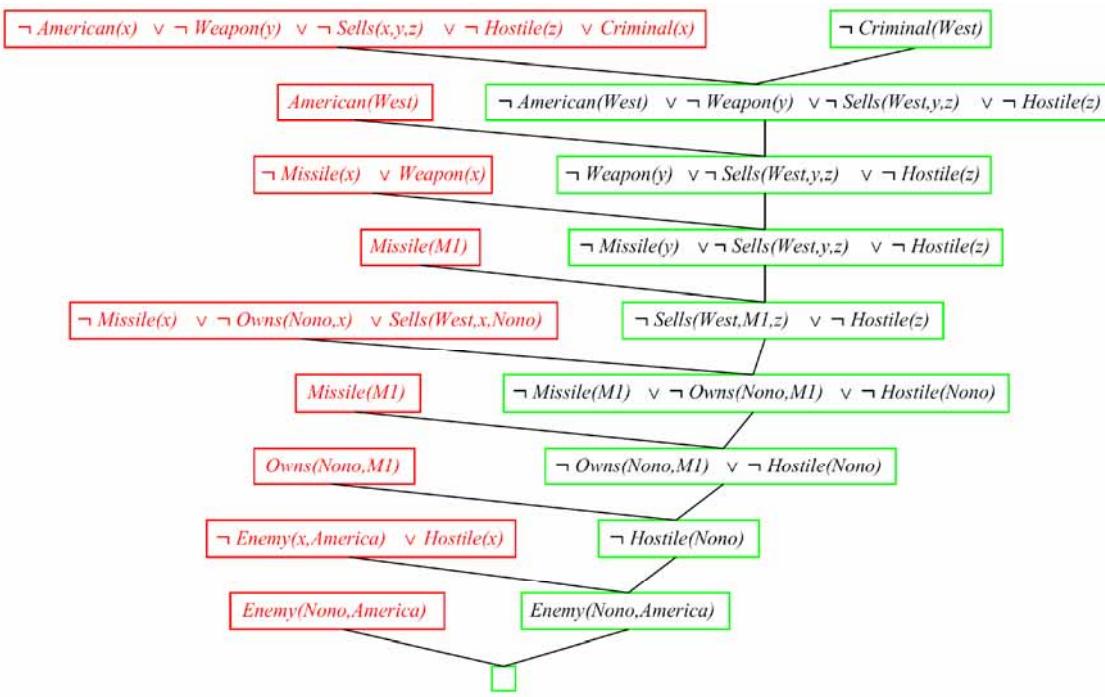
اثبات تحلیل : شرط های نامحدود

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل سیزدهم

نامعلومی^۱

(عدم قطعیت)

^۱ uncertainty یا ابهام یا عدم قطعیت

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- نامعلومی
- احتمال^۱
- ظاهر و معناها
- استدلال (استنتاج)
- استقلال^۲ و قانون بیز^۳

Probability^۱
Independence^۲
Bayes' Rule^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

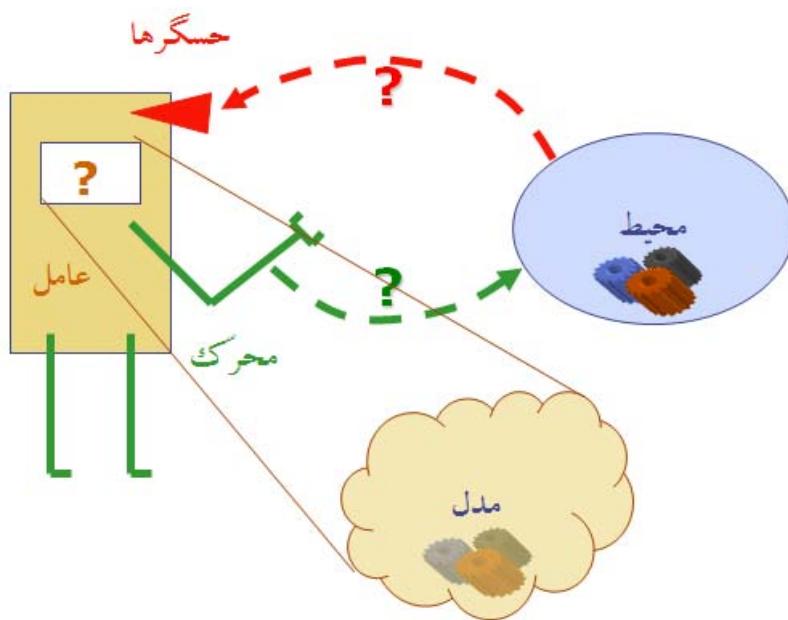


هوش مصنوعی

نامعلومی

در تعدادی از محیط های جالب عامل ها، نا معلوم بودن یا عدم قطعیت دارای یک نقش زیاد می باشد؛ به عبارت دیگر؛ عملکردها ممکن است دارای اثرات نامعلومی باشند؛ مثل، پرتاپ یک پیکان به هدف. عامل ها ممکن است وضعیت درست جهان را ندانند. در ضمن ارتباط های میان واقعیت ها ممکن است به صورت معلوم نباشند. مثلاً گاهی اوقات، در زمانی که هوا ابری می باشد، باران می بارد؛ به عبارت دیگر، هر وقت که هوا ابری باشد، باران نمی بارد!. عامل های هوشمند باید نامعلوم بودن یا ابهام را بررسی نمایند.

عامل نامعلوم^۱



انواع نامعلومی

uncertain agent¹

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

نامعلومی در دانش قبلی

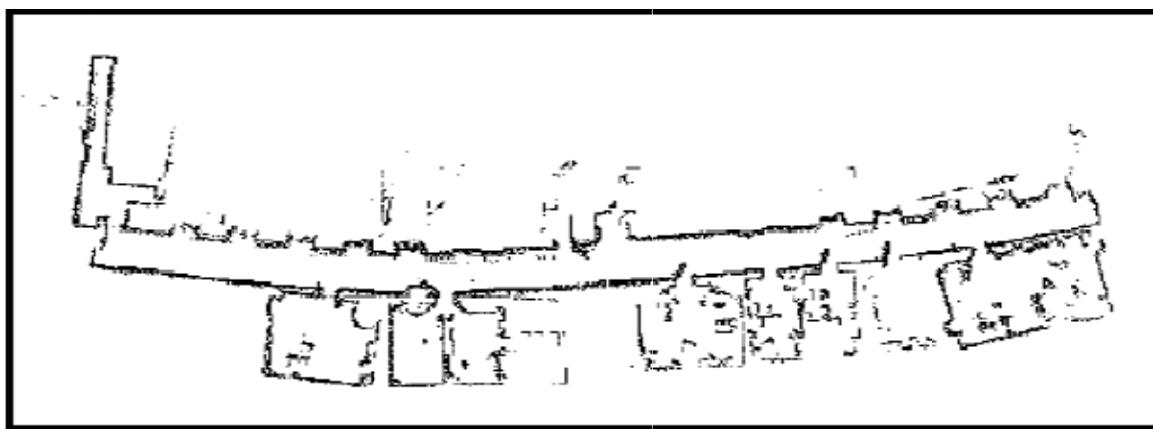
به عنوان مثال، بعضی از دلایل بیماری ناشناخته اند و در دانش یک عامل دستیار پزشک بیان نشده اند.

نامعلومی در عملکردها

مثلاً، کارهایی که باید برای انجام آن ها پیش شرط هایی برقرار باشند؛ شاید هم این پیش شرط ها طولانی هم باشند؛ به عنوان مثال، برای رانندگی کردن با اتومبیل خودم در صبح باید اتومبیل من در طول شب دزدیده نشده باشد؛ تایر یا تایرهای آن پنچر نباشند؛ باید در مخزن (باک) آن، گاز وجود داشته باشد؛ با تری آن از کار نیفتاده باشد؛ باید موتور آن روشن شود؛ کلیدهای آن را گم نکرده باشم؛ کامیون مسیر رانندگی را مسدود نکرده باشد؛ نباید به طور ناگهانی، من کور یا فلچ شوم و البته نمی توانیم همه موارد را لیست کنیم.

نامعلومی در ادراک

مثلاً ممکن است حسگرها اطلاعات کامل یا دقیق را در مورد جهان بزنگردانند؛ مثلاً ممکن است یک ربات مسیر درست را پیدا نکند:



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چگونه با نامعلومی، برخورد کنیم؟

به صورت نآشکار (ضمنی یا تلویحی)

به مواردی که نامعلوم هستند، در زمانی که شما می توانید کار خود را انجام دهید، بی اعتماد باشید و عملیاتی را به وجود بیاورید که در مقابل نامعلومی، مقاوم هستند.

به صورت آشکار

مدلی از جهان را بسازید که نامعلومی را در وضعیت ها، حرکت ها و مشاهده هایش در نظر می گیرد و در مورد اثر عملیاتی که مدل انجام می دهد دلیل بیاورید.

منطق و نامعلومی

ما به زودی خواهیم دید که چگونه منطق را با نامعلومی به کار ببریم؛ مثال هایی از منطق را در زیر مشاهده می نمایید:

Studies(Bart) \vee WatchesTV(Bart)

Hungry(Homer) \Rightarrow Eats(Homer, HotDog) \vee Eats(Homer, Pie)

$\exists x : \text{Hungry}(x)$

متاسفانه، نگرش منطقی دارای برخی اشکالات می باشد.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ضعف های منطق^۱

نمی توانیم همه‌ی نتیجه‌های ممکن را تایین کنیم؛ مثلاً، نمی‌دانیم که اگر الان از خانه خارج شویم، آیا به موقع در محل مورد نظر خواهیم بود یا نه؟. یا اگر پدال گاز اتومبیل خود را زیاد‌تر فشار دهیم آیا باعث زودتر رسیدن ما به مقصد می‌شود یا اینکه باعث می‌شود که ما تصادف کنیم؟!.

ما ممکن است همه‌ی نتیجه‌های ممکن را ندانیم. مثلاً، "در صورتی که یک بیمار دارای دندان درد باشد، وی ممکن است پوسیدگی^۲ در دندان خود داشته باشد، یا ممکن است دارای بیماری لته^۳ باشد، یا ممکن است دلیلی دیگر داشته باشد که ما چیزی در مورد آن نمی‌دانیم."

ما هیچ راهی برای صحبت در مورد احتمال حوادث نداریم. مثلاً، "امروز ممکن است که من با تیر چراغ برق برخورد کنم."

کمیت و کیفیت

منطق، یک نگرش کمی به نامعلومی را ارایه می‌دهد؛ ما می‌توانیم بگوییم که یک اتفاق، بیشتر از دیگری روی می‌دهد یا این که ممکن است یک اتفاق روی دهد. این روش در مواردی که ما آماری نداریم، مفید می‌باشد. احتمال به ما اجازه می‌دهد که به صورت کیفی بحث کنیم؛ ما احتمال رخ دادن یک اتفاق را با یک عدد بیان می‌کنیم.

نامعلومی و عقلانیت (هوشمند بودن)

weaknesses with logic^۱

cavity^۲

gum^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تعريف ما از عقلانیت را به یاد بیاورید ؟ یک عامل هوشمند (عقلانی) ، عاملی است که با بیشینه نمودن معیار کارایی عمل می نماید . ما چگونه این عامل را در یک جهان دارای نامعلومی تعریف نمایم ؟ . می گوییم ، عملکرد عامل دارای نتیجه هایی است و هر نتیجه را با یک عدد که نشان دهنده ای احتمال آن است بیان می نماییم . سپس یک عامل می تواند به هر کدام از نتایج ممکن و عدد آن ها و احتمال رخدادن نتیجه توجه نماید و عملی را که دارای بالاترین سود مورد انتظار می باشد را انتخاب نماید .

تیوری ترکیب کننده ای برتری ها با احتمال رخداد یک نتیجه ، تیوری تصمیم گیری ^۱ نام دارد .

مثال : بیایید عملکرد A را ترک کردن به مقصد فرودگاه ، t دققه قبل از پرواز قرار دهیم . آیا در زمان A به موقع آن جا خواهم بود ؟

مشکلات :

- (۱) مشاهده پذیری جزئی ^۲ ؛ مثل ، حالت جاده ، برنامه ای رانندگان دیگر اتومبیل ها و
- (۲) حس گرهای پر پارازیت ^۳ (اغتشاش) ؛ مثلا ، گزارش های ترافیک ممکن است درست نباشند .
- (۳) نامعلومی در نتایج عملکرد ^۴ ؛ مثلا ممکن است تایر اتومبیل مان پنچر شود و
- (۴) پیچیدگی بیش از حد ^۵ طرح ریزی ^۱ و پیش بینی ترافیک ^۲

decision theory ^۱

partial observability ^۲

noisy sensors ^۳

uncertainty in action outcomes ^۴

immense ^۵

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بنابراین یک شیوه‌ی منطقی یا با ریسک‌های دروغ^۳ است؛ مثلاً، [این که بگوییم]^۴ با A_{25} به موقع خواهم رسید^۵ یا به نتایجی منجر می‌شود که خیلی برای تصمیم‌گیری ضعیفند؛ " A_{25} در صورتی مرا به موقع خواهد رساند که تصادفی در راه رخ ندهد و باران نبارد و لاستیک اتومبیل من بی عیب باشد و"
" A_{1440} معقولانه است که مرا به موقع برساند اما من باید در طول شب در فرودگاه بممانم!")

روش‌هایی برای به کار گیری نامعلومی

منطق پیش‌فرض یا غیر یکنواخت^۶؛ فرض کنید که اتومبیل من دارای تایر پنچر نباشد.
یا فرض کنید که A_{25} کار می‌کند مگر این که با تناقض^۷ ثابت شود که کار نمی‌کند.

احتمال

یک چارچوب مشهور و قابل فهم برای نامعلومی می‌باشد که دارای معنی‌های واضح است و برای ترکیب شواهد، استدلال در مورد پیشگویی و تشخیص و ترکیب شواهد جدید، جواب‌های اساسی را مهیا می‌کند. احتمال، برخی از سطح‌های انسان‌های ماهر (خبره‌های انسانی) را شناسایی می‌کند.

اساس احتمال

به عنوان مثال، احتمال‌های $P(\text{Hungry}(\text{Homer})) = 0.99$ و $P(\text{BartStudied}) = 0.01$ را در نظر بگیرید. توجه کنید که خود گزاره ممکن است درست یا غلط باشد و فرق دارد با اینکه بگوییم، عبارت به صورت جزیی، درست است.

modelling^۸

predicting traffic^۹

risks falsehood^{۱۰}

nonmonotonic^{۱۱}

contradiction^{۱۲}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مقادیر تصادفی^۱

یک متغیر تصادفی ، متغیر یا گزاره ای است که مقدارش ناشناخته است و دارای دامنه ای از مقادیر است که می تواند آن ها را به خود بگیرد ؛ این متغیرها می توانند از موارد زیر باشند :

○ بولین (درست و غلط) ؛ مثل : isRaining و Hungry(Homer)

▪ گسسته ؛ مقادیر از یک دامنه ی قابل شمارش گرفته می شوند ؛ مثل ، دما^۲ : < گرم ^۳ ، خنک ^۴ ، ملایم ^۵ و پیش بینی وضع هوا : < آفتابی ^۶ ، ابری ^۷ ، بارانی ^۸ >

▪ پیوسته ؛ مقادیر ، می توانند از یک فاصله (دامنه ی غیرقابل شمارش) گرفته شوند ، نظیر : [او^۹] ؛ مثل ، سرعت ^{۱۰} ، زمان ، مکان ^۹

در اینجا بیشتر تمرکز ما بر روی مورد گسسته خواهد بود .

پس ، یک متغیر تصادفی ، یکتابع از فضای نمونه برای برای بعضی از محدوده ها می باشد ، مثل :

Odd(1) = true ها . مثل Boolean ها یا real

random values ^۱
temperature ^۲
hot ^۳
cool ^۴
mild ^۵
sunny ^۶
overcast ^۷
velocity ^۸
position ^۹

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

رویدادها (رخدادها) ی تجزیه ناپذیر یا اتمیک^۱

ما می توانیم گزاره ها را با استفاده از رابط های منطقی استاندارد و به کارگیری اجتماع و اشتراک با هم ترکیب نماییم :

$P(\text{Hungry}(\text{Homer}) \wedge \neg \text{Study}(\text{Bart}))$

$P(\text{Brother}(\text{Lisa}, \text{Bart}) \vee \text{Sister}(\text{Lisa}, \text{Bart}))$

یک عبارت که یک مقدار ممکن را برای هر متغیر نامعین مشخص می کند یک رویداد اتمیک^۲ نام دارد؛ رویدادهای اتمیک دو به دو ناسازگار^۳ (مستقل از هم) هستند. مجموعه ای از همه ای رویدادهای اتمیک، شامل تمام جزییات^۴ هستند. یک رویداد اتمیک، درستی یا نادرستی هر گزاره را پیش بینی می کند. توجه کنید که رویدادهای اتمیک در تشخیص موارد دارای چند متغیر نامعین، مفید می باشند.

نکته : منطق فازی^۵ از درجه ی قطعیت (درجه ی درستی)^۶ استفاده می کند نه نامعلومی^۷.

مثال : نمونه هایی از احتمال $P(\text{Weather}=\text{sunny}) = 0.72$ و $P(\text{Cavity}'=\text{true}) = 0.1$

غیرشرطی هستند. توزیع احتمال، مقادیر را برای همه ای انتساب های ممکن، ارایه می دهد:

Atomic Events^۸

atomic event^۹

mutually exclusive^{۱۰}

exhaustive^{۱۱}

: یک شکل از منطق ریاضی است که در آن مقادیر بین صفر و یک دارای ارزش درست هستند.

Fuzzy logic^{۱۲}

(WordNet 2.0)

degree of truth^{۱۳}

uncertainty^{۱۴}

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$P(\text{Weather}) = \langle 0.72, 0.1, 0.08, 0.1 \rangle$$

توجه نمایید که مجموع اعداد برابر یک می باشد ($0.72 + 0.1 + 0.08 + 0.1 = 1.00$) . توزیع پیوسته ی احتمال برای مجموعه ای از متغیرهای تصادفی ، احتمال هر رویداد اتمیک را در متغیرهای تصادفی ارایه می کند . به عنوان مثال ، $P(\text{Weather}, \text{Cavity})$ برابر است با یک ماتریس 4×2 از مقادیر :

Weather =	sunny	rain	cloudy	snow
Cavity = true	0.144	0.02	0.016	0.02
Cavity = false	0.576	0.08	0.064	0.08

هر پرسش درباره ی یک دامنه ، می تواند به وسیله ی توزیع پیوسته ، پاسخ داده شود ، زیرا هر رویداد ، مجموعه ای از فضای نمونه می باشد .

اگر داشته باشیم ، $P(\text{Sunny})=0.5$ ، $P(\text{Overcast})=0.4$ ، $P(\text{Rain})=0.1$ ، در این صورت می توانیم احتمالات ترکیب های متغیرها را لیست نماییم :

$$\begin{aligned} P(\text{Overcast} \wedge \text{Humid}) &= 0.2 & P(\text{Rain} \wedge \neg \text{Humid}) &= 0.1 & P(\text{Rain} \wedge \text{Humid}) &= 0.1 \\ P(\text{Sunny} \wedge \neg \text{Humid}) &= 0.25 & P(\text{Sunny} \wedge \text{Humid}) &= 0.15 & P(\text{Overcast} \wedge \neg \text{Humid}) &= 0.2 \end{aligned}$$

این ، یک توزیع احتمال توأم^۳ نام دارد . برای متغیرهای پیوسته ، ما نمی توانیم مقادیر را بشماریم ؛ در عوض ، ما از یکتابع پارامتری به نام توزیع نرمال^۱ استفاده می نماییم :

^۱ کرم خوردگی دندان

^۲ ابری

joint probability distribution^۴

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$P(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dz \quad (\text{توزيع نرمال})$$

تصمیم گیری تحت شرایط نامعلومی - در مثال رفتن به فرودگاه که قبلا در مورد آن صحبت کردیم ، تصور نمایید موارد زیر را داریم :

$P(A_{25}) = 0.04$ مرا به موقع برساند

$P(A_{90}) = 0.74$ مرا به موقع برساند

$P(A_{120}) = 0.95$ مرا به موقع برساند

$P(A_{1440}) = 0.999$ مرا به موقع برساند

کدام مورد فوق را باید انتخاب نماییم ؟ انتخاب ، وابسته به صلاح دید های ما ، برای از دست ندادن پرواز ، وضعیت فرودگاه و می باشد .

قضیه ی سودمندی ^۲ ، برای ارایه و استدلال در مورد صلاح دیدها استفاده می شود .

قضیه ی تصمیم گیری ^۳ برابر است با ، قضیه ی سودمندی + قضیه ی احتمال .

مبانی احتمال - با یک مجموعه از فضای ساده Ω شروع می کنیم ؛ مثال ، شش حالت ممکن یک تاس . اگر $\omega \in \Omega$ باشد .

normal distribution ^۱

Utility theory ^۲

Decision theory ^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



در یک بار پرتاپ یک تاس، احتمال آمدن هر کدام از اعداد ۱ تا ۶ روی شش وجه تاس، بین ۰ تا ۱ است؛ پس داریم: $P(\omega) \leq 1$. در ضمن مجموع احتمال های آمدن هر وجه تاس، برابر ۱ است؛
 $\sum_{\omega} P(\omega) = 1$ یعنی،

مثال: $P(1)=P(2)=P(3)=P(4)=P(5)=P(6)=1/6$ می بینید که احتمال آمدن هر وجه تاس برابر با $\frac{1}{6}$ می باشد.

یک رویداد A زیرمجموعه ای از Ω می باشد:

$$P(A) = \sum_{\{\omega \in A\}} P(\omega)$$

مثلا، احتمال این که در یک بار پرتاپ یک تاس، ظاهر می شود کمتر از ۴ باشد، برابر است با: احتمال ۱ آمدن + احتمال ۲ آمدن + احتمال ۳ آمدن = $1/2 = 1/6 + 1/6 + 1/6$

استنتاج با توزیع های پیوسته ای احتمال

آسان ترین راه انجام استنتاج احتمالی این است که یک جدول ارایه کننده ای توزیع پیوسته ای احتمال را داشته باشیم. به هر متغیر مستقل نگاه کنید و احتمال وابسته را پیدا نمایید.

	Humidity'=High	Humidity=High	Humidity=Normal	Humidity=Normal
--	----------------	---------------	-----------------	-----------------

^۱ رطوبت

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

	Outlook'=Overcast	Outlook=Sunny	Outlook=Overcast	Outlook=Sunny
Rain	0.1	0.05	0.15	0.05
\neg Rain	0.2	0.15	0.1	0.2

می توانیم توزیع پیوسته‌ی احتمال را برای تشخیص احتمال نهایی^۱ متغیر وابسته ، با جمع کردن همه
ی موارد وابسته به متغیر که می توانند درست باشند به دست آوریم :

$$P(\text{Rain}) = 0.1 + 0.05 + 0.15 + 0.05 = 0.35$$

اگر P ، یک توزیع تصادفی را برای هر متغیر تصادفی نظری X ، بیان نماید ؛ داریم :

$$P(X = x_i) = \sum_{\{\omega : X(\omega) = x_i\}} P(\omega)$$

مثال : در پرتاپ یکبار یک تاس ، احتمال این که عدد فرد ، ظاهر شود برابر است با :

$$P(\text{Odd}=\text{true}) = P(1) + P(3) + P(5) = 1/6 + 1/6 + 1/6 = 1/2$$

گزاره ها - یک گزاره را به صورت رویداد (مجموعه‌ای از فضاهای نمونه) در جایی که گزاره صحیح می باشد ، تصور نمایید . برای متغیرهای تصادفی بولین A و B ، اگر رویداد a ، مجموعه‌ای از فضاهای نمونه ای به شرط آن که $A(\omega)=\text{true}$ باشد ، آن گاه رویداد $\neg a$ ، مجموعه‌ای از فضاهای نمونه ای به شرط آن که $A(\omega) = \text{false}$ خواهد بود و رویداد $a \wedge b$ = مواردی هستند که $(A(\omega) \text{ و } B(\omega))$ [هر دو] ، می باشند . True

^۱ پیش بینی

marginal probability^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

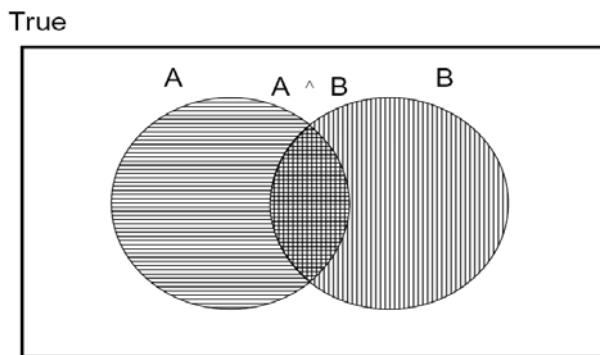


هوش مصنوعی

اغلب در هوش مصنوعی ، فضای نمونه ای ، به وسیله ی مقادیر یک مجموعه از متغیرهای تصادفی تعریف می شود . توجه نمایید که فضای نمونه ای حاصلضرب کارتزین محدوده هایی از متغیرها می باشد . با متغیرهای بولین ، فضای نمونه با مدل منطق گزاره ای ، برابر می باشد .

مثال :

$$(a \vee b) \equiv (\neg a \wedge b) \vee (a \wedge \neg b) \vee (a \wedge b) \Rightarrow P(a \vee b) = P(\neg a \wedge b) + P(a \wedge \neg b) + P(a \wedge b)$$



چرا از احتمال استفاده می کنیم ؟ تعریف ، بر این دلالت می کند که رویدادهای منطقی وابسته ای معین ، باید روابط احتمالاتی داشته باشند .

$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$: مثال

یک فرمول :

$$P(A \vee \neg A) = P(A) + P(\neg A) - P(A \wedge \neg A) \Rightarrow P(\text{True}) = P(A) + P(\neg A) - P(\text{False}) \Rightarrow 1 = P(A) + P(\neg A) \Rightarrow P(A) = 1 - P(\neg A)$$

ظاهر گزاره ها

مترجم: سهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی

متغیرهای تصادفی گزاره ای یا بولین : مثلاً اگر Cavity برای آیا من کرم خورده‌گی دندان دارم ؟ ، باشد ؛ آن گاه ، $Cavity = \text{true}$ یک گزاره می باشد .

متغیرهای تصادفی گستته (محدود یا نامحدود)

مثال : هوا (وضعیت جوی)^۱ ، دارای یکی از حالت های آفتابی ، بارانی ، ابری و برفی^۲ می باشد .
هوا = بارانی ، یک گزاره می باشد .

متغیرهای تصادفی پیوسته^۳ ، محدوده دار^۴ یا فاقد محدوده^۵ می باشند . مثال : دما = ۲۱.۶ ، یا
. Temp < 22.0

احتمال شرطی^۶

قبل دیدیم که اگر هوا ابری باشد ، احتمال بارندگی بالا می رود ؛ که این یک احتمال شرطی نام دارد و به صورت $P(\text{Rain}|\text{Cloudy})$ نوشته می شود . برای توزیع های شرطی توجه نمایید که اگر $P(\text{Cavity}|\text{Toothache})$ مورد نظر باشد و ما بیش تر بدانیم ، مثلا cavity ارایه شده باشد ، در این صورت داریم : $P(\text{cavity}|\text{toothache}, \text{cavity}) = 1$:

تعریف احتمال شرطی

Weather^1

snow^2

Continuous random variables^۷

bounded^۸

unbounded^۹

conditional probability^۹

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$P(b) \neq 0, \text{ با شرط } P(a | b) = \frac{P(a \wedge b)}{P(b)}$$

با استفاده از فرمول بالا، قانون ضرب^۱ را به صورت زیر خواهیم داشت:

$$P(a \wedge b) = P(a | b)P(b) = P(b | a)P(a)$$

مثال:

$$P(\text{Weather}, \text{Cavity}) = P(\text{Weather} | \text{Cavity})P(\text{Cavity})$$

مثال: اگر $P(\text{Rain} | \text{Cloudy})$ باشد؛ $P(\text{Cloudy}) = 0.3$ و $P(\text{Rain} \wedge \text{Cloudy}) = 0.15$ چه
قدر است؟

$$P(\text{Rain} | \text{Cloudy}) = \frac{P(\text{Rain} \wedge \text{Cloudy})}{P(\text{Cloudy})} = \frac{0.15}{0.3} = 0.5$$

قانون زنجیره ای^۲ - برای کاربرد موافقیت آمیز قانون ضرب به وجود آمده است:

$$\begin{aligned} P(X_1, \dots, X_n) &= P(X_1, \dots, X_{n-1})P(X_n | X_1, \dots, X_{n-1}) \\ &= P(X_1, \dots, X_{n-2})P(X_{n-1} | X_1, \dots, X_{n-2})P(X_n | X_1, \dots, X_{n-1}) \\ &= \dots = \prod_{i=1}^n P(X_i | X_1, \dots, X_{i-1}) \end{aligned}$$

توزیع پیوسته^۳

Product rule^۱

Chain rule^۲

joint distribution^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برای k متغیر تصادفی X_k, X_1, \dots ، توزیع پیوسته‌ی این متغیرها، جدولی است که در آن هر عدد، احتمال یک ترکیب از مقدارهای X_k, X_1, \dots را ارایه می‌نماید.

: مثال

	Toothache	\neg Toothache
Cavity	0.04	0.06
\neg Cavity	0.01	0.89

$$P(\neg Cavity \wedge \text{Toothache}) \quad P(Cavity \wedge \neg \text{Toothache})$$

با توجه به جدول بالا داریم :

$$\begin{aligned} P(\text{Toothache}) &= P((\text{Toothache} \wedge \text{Cavity}) \vee (\text{Toothache} \wedge \neg \text{Cavity})) \\ &= P(\text{Toothache} \wedge \text{Cavity}) + P(\text{Toothache} \wedge \neg \text{Cavity}) = 0.04 + 0.01 = 0.05 \\ P(\text{Toothache} \vee \text{Cavity}) &= P((\text{Toothache} \wedge \text{Cavity}) \vee (\text{Toothache} \wedge \neg \text{Cavity}) \\ &\vee (\neg \text{Toothache} \wedge \text{Cavity})) = 0.04 + 0.01 + 0.06 = 0.11 \end{aligned}$$

می‌توانیم همچنین احتمالات شرطی را محاسبه نماییم :

$$\begin{aligned} P(\text{Cavity} | \text{Toothache}) &= P(\text{Cavity} \wedge \text{Toothache}) / P(\text{Toothache}) \\ P(\text{Cavity} \wedge \text{Toothache}) &=? \end{aligned}$$

با توجه به جدول، برابر است با ۰.۰۴

$$P(\text{Toothache}) = ?$$

حاصل این احتمال را هم که قبلاً محاسبه کردیم، برابر بود با: ۰.۰۵؛ درنتیجه داریم:

$$P(\text{Cavity} | \text{Toothache}) = 0.04 / 0.05 = 0.8$$

نرم‌افزاری

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در فرمول احتمال شرطی ، مقسوم علیه^۱ می تواند به صورت یک ثابت نرمال سازی α دیده شود :

$$P(\text{Cavity}|\text{Toothache}) = P(\text{Cavity} \wedge \text{Toothache}) / P(\text{Toothache}) = \alpha [P(\text{Cavity} \wedge \text{Toothache})] = \alpha \times 0.04 = \frac{1}{0.05} \times 0.04 = 20 \times \frac{4}{100} = 0.8$$

تعمیم

$$P(A \wedge B \wedge C) = P(A|B,C) P(B|C) P(C)$$

استدلال با شمارش

تصور نمایید X ، همه ی متغیرها باشد . معمولاً ، ما توزیع پیوسته ی متغیرهای پرس و جوی^۲ Y را می خواهیم که توسط مقادیر e برای متغیرهای ثابت^۳ E ارایه شده اند . تصور نمایید متغیرهای مخفی باشند . در این صورت ، مجموع تمام پیوند ها با جمع کردن متغیرهای مخفی به دست می آید :

$$P(Y | E = e) = \alpha P(Y, E = e) = \alpha \sum_h P(Y, E = e, H = h)$$

واژگان درون مجموعه ، تماماً متصل می باشند زیرا Y و E و H با هم ، خروجی آن ها ، مجموعه ای از متغیرهای تصادفی می باشد .

مسایل آشکار :

پیچیدگی زمانی در بدترین حالت در جایی که d بزرگترین می باشد

برابر است با $(O(d^n))$

denominator^۱

query variables^۲

evidence variables^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پیچیدگی فضای نگهداری توزیع پیوسته برابر است با $O(d^n)$ (۲)

چگونه می توان تعداد را برای ورودی های $O(d^n)$ پیدا نماییم (۳)

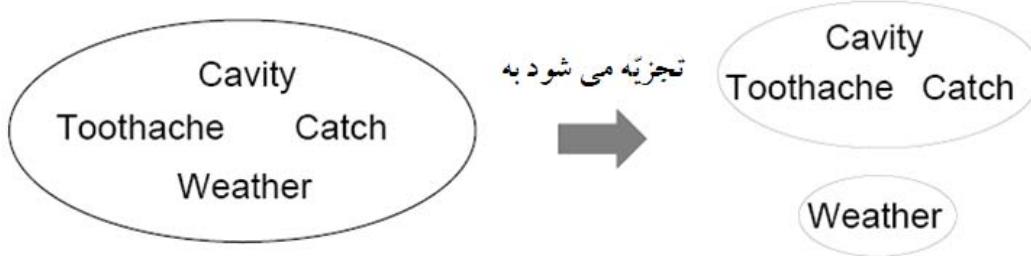
استقلال^۱

در برخی از موارد ، ما می توانیم چیزها را با توجه به این که یک متغیر دارای هیچ اثری بر روی دیگری نمی باشد ساده نماییم . برای مثال ، اگر ما یک متغیر چهارم DayOfWeek را به محاسبه بارندگی اضافه نماییم چه تغییری به وجود می آید ؟ ؛ از آنجایی که روز هفته بر احتمال باران تاثیری نخواهد داشت ، ما داریم :

$$P(\text{Rain}|\text{Cloudy}, \text{Monday}) = P(\text{Rain}|\text{Cloudy}, \text{Tuesday}) \dots = P(\text{Rain}|\text{Cloudy})$$

ما گفتیم که Rain و DayOfWeek مستقل می باشند . بنابراین می توانیم توزیع پیوسته ای احتمال بزرگ تر را به زیر جدول هایی مجزا تقسیم نماییم ؛ استقلال به ما کمک خواهد کرد که دامنه را به تکه های مجزا تقسیم نماییم .

تعریف - A و B در صورتی مستقل هستند که ، $P(B|A) = P(B)$ یا $P(A|B) = P(A)$ یا $P(A,B) = P(A)P(B)$. با توجه به این تعریف برای مثال دندان درد داریم :



Independence^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$P(\text{Toothache}, \text{Catch}, \text{Cavity}, \text{Weather}) = \\ P(\text{Toothache}, \text{Catch}, \text{Cavity})P(\text{Weather})$$

2^n ورودی به ۱۲ تا کاهش داده می شود؛ برای n سکه‌ی مستقل داریم:

استقلال مطلق، خیلی مفید است ولی نادر یا کمیاب^۱ می باشد. مثلا، دندانپزشکی، زمینه‌ی بزرگی با صد ها متغیر می باشد که هیچ‌کدام مستقل نیستند. در این مورد چه کاری انجام بدھیم؟

استقلال شرطی^۲

اگر A و B مستقل شرطی باشند و C هم داده شده باشد، داریم:

$$P(A, B|C) = P(A|C)P(B|C)$$

$$P(A|C, B) = P(A|C)$$

در بیشتر موارد، استفاده از استقلال شرطی، اندازه‌ی ارایه‌ی توزیع پیوسته را از حالت نمایی n به حالت خطی n کاهش می دهد. استقلال شرطی، پایه‌ای ترین و نبیرون‌ترین دانش درباره‌ی محیط‌های نامعین می باشد.

مثال اتومبیل - سه پارامتر گاز (Gas)، باتری (Battery) و استارت (Start) را برای اتومبیل در نظر بگیرید؛ داریم:

$$P(\text{Battery}| \text{Gas}) = P(\text{Battery})$$

چون که گاز و باتری، مستقل از هم هستند. ولی

$$P(\text{Battery}| \text{Gas, Starts}) \neq P(\text{Battery}| \text{Starts})$$

rare^۱
conditional independence^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

چون گاز و باتری با وجود استارت ، مستقل از هم نمی باشند .

قانون بیز

با استفاده از قانون ضرب داریم : $P(a \wedge b) = P(b | a)P(a)$ و $P(a \wedge b) = P(a | b)P(b)$ ، در نتیجه ما می توانیم این تساوی ها را برابر هم قرار دهیم :

: در نتیجه قانون بیز را به صورت زیر خواهیم داشت :

$$P(a | b) = \frac{P(b | a)P(a)}{P(b)}$$

یا به شکل نرم ام داریم :

$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)} = \alpha P(X | Y)P(Y)$$

مثال برای قانون بیز - این مفروضات را در نظر بگیرید : منتظر ^۱ باعث خشکی گردن ^۱ در ۵۰٪ از بیماران می شود ؛ $P(\text{stiffNeck} | \text{Meningitis}) = 0.5$ ؛ احتمال منتظر قدمی برابر است با $P(\text{meningitis}) = 0.00002$ ؛ احتمال خشکی گردن قدمی برابر است با $1/20000$ و $P(\text{stiffNeck}) = 0.05$. حال یک بیمار با خشکی گردن مراجعه می نماید . احتمال این که وی دارای منتظر باشد چقدر است ؟

$$P(\text{meningitis} | \text{stiffNeck}) = \frac{P(\text{stiffNeck} | \text{meningitis})P(\text{meningitis})}{P(\text{stiffNeck})} = \frac{0.5 \times 0.00002}{0.05} = 0.0002$$

meningitis ^۱
stiff neck ^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مثال : جدول زیر را در نظر بگیرید ،

	Toothache	\neg Toothache
Cavity	0.04	0.06
\neg Cavity	0.01	0.89

اگر داشته باشیم : $P(\text{Cavity}|\text{Toothache})=0.8$ و $P(\text{Toothache})=0.05$ ، $P(\text{Cavity})=0.1$ در این صورت $P(\text{Toothache}|\text{Cavity})$ را به دست آورید .

$$P(\text{Toothache}|\text{Cavity}) = (0.8 \times 0.05) / 0.1 = 0.4$$

تعمیم

$$P(A \wedge B \wedge C) = P(A \wedge B | C) P(C) = P(A | B, C) P(B | C) P(C)$$

$$P(A \wedge B \wedge C) = P(A \wedge B | C) P(C) = P(B | A, C) P(A | C) P(C)$$

$$P(B | A, C) = \frac{P(A | B, C) P(B | C)}{P(A | C)}$$

مثال : مدل ساده‌ی بیز ^۱

Naïve Bayes Model ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در یک مدل قدیمی برای تشخیص بیماری؛ علایمی که به صورت مستقل شرطی هستند برای بیماری داده شده اند، بنابراین، اگر X_1, \dots, X_n علایمی باشند که مریض گفته؛ مثل، سردرد، تب بالا و غیره و H ، فرضیه هایی برای سلامتی بیمار باشد، در این صورت، داریم:

$$P(X_1, \dots, X_n, H) = P(H)P(X_1|H)\dots P(X_n|H)$$

این مدل ساده‌ی بیزی، اجازه‌ی ارایه‌های فشرده را می‌دهد و فرضیه‌های مستقل نیرومندی دارد.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



۱

فصل چهاردهم

شبکه های بیزی^۲

^۱ تصویر مربوط به توماس بیز (Thomas Bayes) می باشد.

Bayesian networks^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- ساختار
- معنا
- توزیعات پارامتربرندی شده^۱

استنتاج احتمالی

در گذشته، ما در مورد سیستم های برمبنای قانونی که می توانند استنتاج های منطقی را انجام دهند صحبت کردیم. مثلاً اگر کبوتر خانگی^۲، گرسنه شود به بازار می رود:

parametrized distributions^۱
Homer^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Hungry(Homer) \Rightarrow GoesTo(Homer,Quickie-mart)

ما تمایل داریم این نوع از عملکرد را به جهان هایی با نامعلومی توسعه دهیم . مثلا :

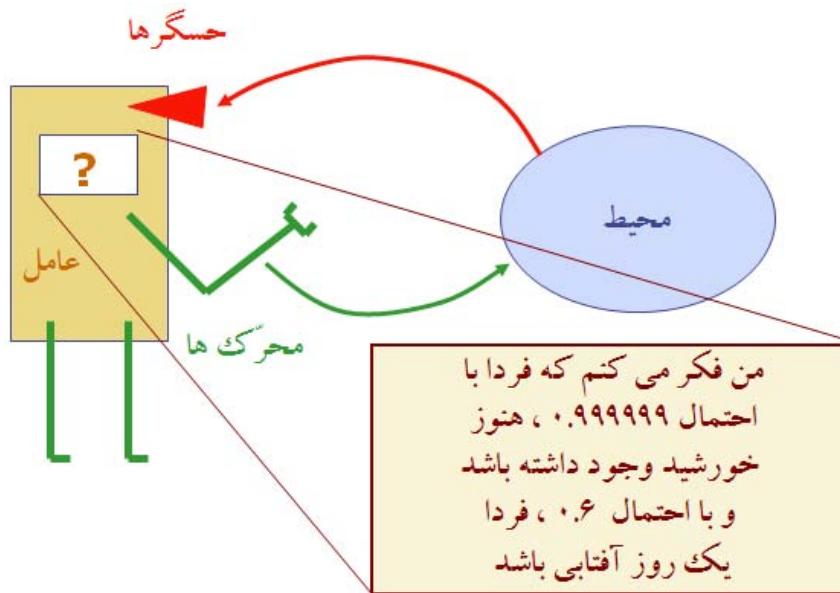
$$P(\text{Hungry}(\text{Homer}))=0.98$$

$$P(\text{GoesTo}(\text{Homer},\text{Quickie-mart})|\text{Hungry}(\text{Homer}))=0.5$$

$$P(\text{GoesTo}(\text{Homer},\text{Quickie-mart}))=?$$

در کل ، ما می توانیم از قانون بیز برای این کار استفاده نماییم ، در این مورد ، مشکل در کار کردن با توزیع پیوسته ای احتمال می باشد ، چون دارای جدولی بزرگ به صورت نمایی می باشد . ما به ساختمان داده ای که در آن تعدادی متغیر بر هم تاثیر نمی گذارند نیازمندیم . به عنوان مثال ، رنگ کلاه بارت بر این که Homer گرسنه می باشد تاثیری نمی گذارد ؛ ما این ساختار را یک شبکه بیزی می نامیم .

عامل احتمالی



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مسئله

در زمان مشخص t ، پایگاه دانش (KB) یک عامل، مجموعه ای از حدس ها است؛ در زمان t ، حسگرهای عامل، یک دید را به وجود می آورند که احتمال یکی از حدس ها را افزایش می دهد؛ حال سوال این است که در این موقع عامل باید چگونه احتمال موارد دیگر را به روز نماید؟

هدف شبکه های بیزی

توصیف یک مجموعه از حدس ها را با به وجود آوردن ارتباط های صریح میان حدس ها و پیدا کردن استقلال شرطی میان حدس ها آسان می کند و یک روش مناسب تر از استفاده از جدول های توزیع پیوسته را برای به روز کردن قدرت حدس ها در زمانی که شواهد جدید مشاهده می شوند، به وجود می آورد.

نام های دیگر شبکه های بیزی

نام های دیگر شبکه های بیزی عبارتند از: شبکه های حدسی^۱، شبکه های احتمالی^۲ و شبکه های بیان کننده ی علت^۳.

شبکه های بیزی

یک روش ساده و گرافیکی برای ادعاهای مستقل شرطی در یک نمایش فشرده برای توزیع پیوسته کامل است. به عبارت دیگر، یک شبکه بیزی، یک گراف مستقیم است که در آن هر گره با اطلاعات احتمالی، تفسیر می شود. یک شبکه دارای موارد زیر است:

belief networks^۱

probabilistic networks^۲

causal networks^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- یک مجموعه از متغیرهای تصادفی که با گره های درون شبکه برابر است.

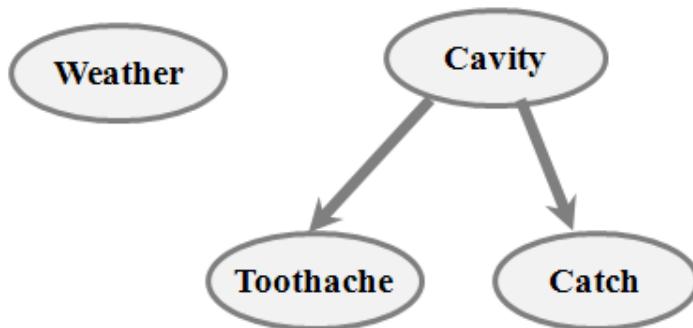
- یک مجموعه از یال ها (لبه ها یا پیکان ها) که گره ها را متصل می کنند. یال ها، تاثیرات را نمایش می دهند. اگر پیکانی از X به طرف Y وجود داشته باشد، آن گاه X ، پدر Y می باشد.

هر گره یک توزیع شرطی احتمال را که نشان دهنده ای احتمال هر مقداری که گره می تواند بگیرد و شرطی که مقادیر والدهایش می توانند بگیرند را نگهداری می کند. در این شبکه، هیچ حلقه ای وجود ندارد؛ به عبارت دیگر، این شبکه به صورت یک گراف مستقیم بدون دور می باشد.

پس، یک شبکه بیزی، یک نمایش ساده و گرافیکی برای موارد مستقل شرطی می باشد.

ساختار شبکه بیزی: در این ساختار هر متغیر دارای یک گره است و شکل آن به صورت یک گراف غیر حلقه ای مستقیم می باشد. در ضمن، یک توزیع شرطی برای هر گره ای ارایه شده توسط والدهای خودش به صورت $P(X_i | Parents(X_i))$ می باشد. در ساده ترین حالت، توزیع شرطی ارایه شده به صورت یک جدول احتمال شرطی^۱، توزیع را برمبای X_i برای هر ترکیب از مقادیر والد ارایه می دهد.

مثال - توپولوژی شبکه، موارد استقلال شرطی را بیان می کند:



conditional probability table (CPT)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

هوا مستقل از دیگر متغیرها می باشد . دندان درد و دچار شدن به آن به صورت شرطی مستقل از پوسیدگی دندان می باشند .

مثال دزدی^۱ - من سرکار هستم و همسایه ام ، جان با من تماس می گیرد و می گوید آژیر خانه ی من فعال است ، اما همسایه ی دیگرم ، مری^۲ این کار را انجام نمی دهد . بعضی از وقت ها آژیر با کم ترین لرزه ای فعال می شود . آیا در خانه ی من واقعاً دزد است ؟

متغیرها در این مثال عبارتند از : دزد ، لرزه^۳ ، آژیر^۴ ، تماس های جان^۵ ، تماس های مری^۶ .

توپولوژی شبکه آن چه که اتفاق افتاده است را برگشت می دهد (معکس می کند) :

- یک دزد می تواند آژیر را در وضعیت خاموش قرار دهد .

- یک لرزه ی زمین می تواند آژیر را خاموش نماید .

- آژیر می تواند سبب تماس مری شود .

- آژیر می تواند سبب تماس جان شود .

burglar^۱

Mary^۲

earthquake^۳

alarm^۴

JohnCalls^۵

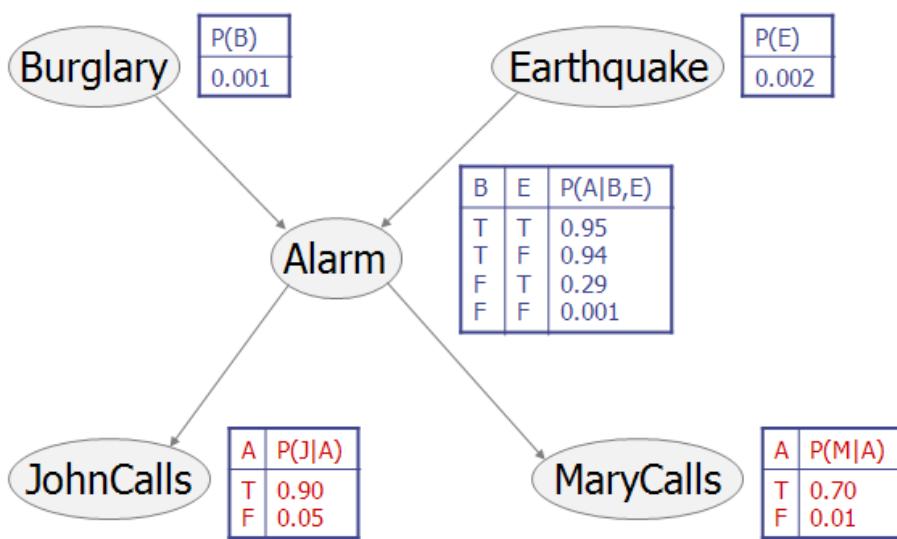
MaryCalls^۶

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



تراکم^۱

شبکه های بیزی ، ارایه ی شبکه های بزرگ را امکان پذیر می سازند ؛ در این شبکه ها ، اطلاعات اضافی (تکراری)^۲ برداشته می شود .

یک CPT برای X_i بولین با والدهای بولین k دارای 2^k سطر برای ترکیب مقدارهای والد می باشد .
هر سطر یک عدد p را برای $X_i = \text{true}$ نیاز دارد (عدد مورد نیاز برای $X_i = \text{false}$ $1-p$ می باشد).
در صورتی که هر متغیر دارای بیش از k والد نباشد ، شبکه ی کامل به تعداد $O(n.2^k)$ سطر نیاز دارد . توجه کنید که [تراکم] به صورت خطی ، با n رشد می کند و دارای مرتبه $O(2^n)$ برای توزیع با اتصال کامل می باشد .

معانی عمومی

compactness^۱
redundant^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

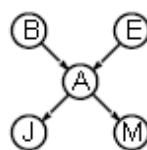


هوش مصنوعی

معانی کلی، توزیع پیوسته‌ی کامل را به صورت ضرب توزیعات شرطی محلی، تعریف می‌نماید:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents } (X_i))$$

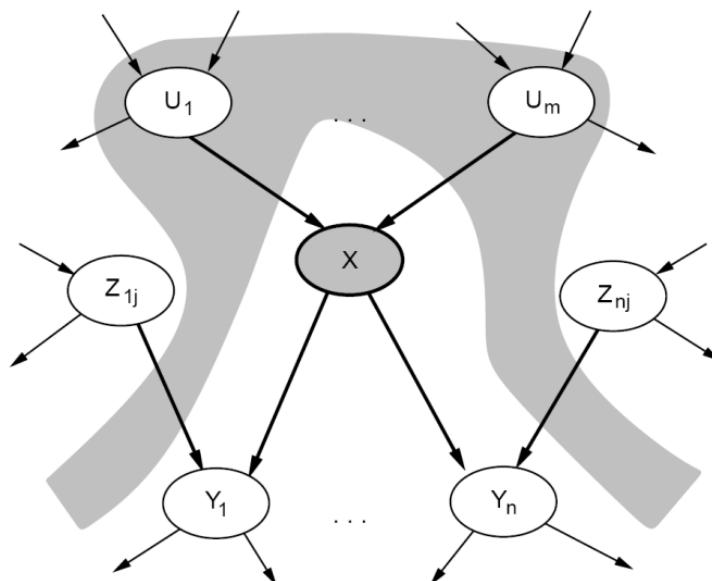
مثال:



$$\begin{aligned} P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \approx 0.00063 \end{aligned}$$

معانی محلی

هر گره، به صورت شرطی مستقل از فرزندانی است که توسط پدرشان به وجود آمده‌اند.



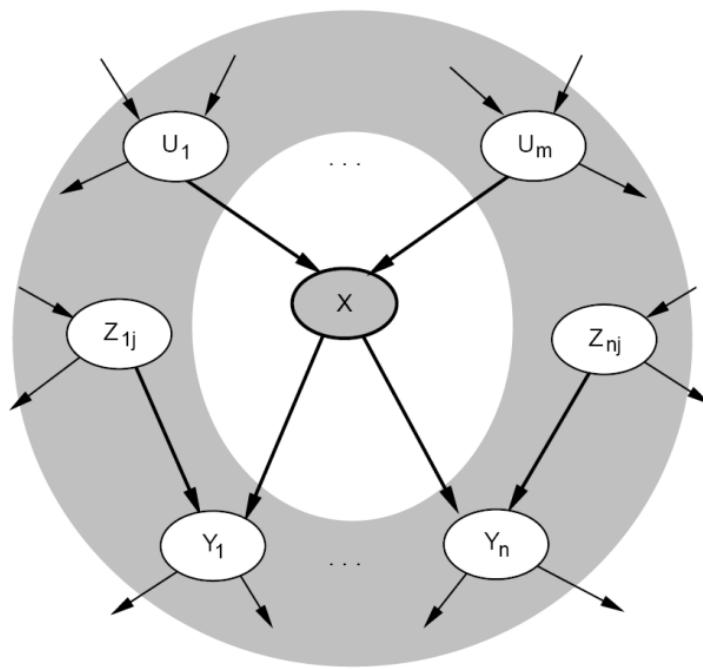
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

قضیّه: معانی محلی \Leftrightarrow معانی عمومی

پوسته‌ی مارکف^۱ - هر گره به صورت شرطی، مستقل از همه‌ی گره‌های دیگری است که توسط پوسته‌ی مارکوف ارایه می‌شوند؛ پوسته‌ی مارکوف تشکیل شده از: والدین + فرزندان + والدین فرزندان



ساختار شبکه

هر گره دارای یک جدول احتمال شرطی است که این جدول، احتمال هر مقدار آن گره و مقادیر والدهای آن را ارایه می‌کند.

بیان نامعلومی (عدم قطعیت) به طور خلاصه

Markov blanket^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

توجه نمایید که ما نیازی به داشتن گره هایی برای همه مواردی که مری ممکن است تماس نگیرد نداریم . یک روش احتمالی به ما اجازه می دهد که این اطلاعات را به صورت M - خلاصه نماییم . این روش ، به یک عامل ساده اجازه می دهد که به جهان های بزرگی که دارای تعداد زیادی از نتایج نامعلوم ممکن هستند رسیدگی کند .

ارایه‌ی ضمنی توزیع پیوسته‌ی کامل

به یاد بیاورید که توزیع پیوسته‌ی کامل به ما اجازه می داد که احتمال هر متغیر و همه آن‌ها یابی که ارایه شده اند را محاسبه نماییم . رویدادهای مستقل می توانند به جدول‌های مستقل تقسیم شوند و این رویدادها ، CPT‌هایی هستند که در شبکه‌ی بیزی دیده می شوند . بنابراین ما می توانیم از این اطلاعات برای انجام محاسبات استفاده نماییم . داریم :

$$P(x_1, x_2, \dots, x_n) = \prod P(x_i | \text{parent}(x_i))$$

مثال :

$$P(A \wedge \neg E \wedge \neg B \wedge J \wedge M) = P(J | A)P(M | A)P(A | \neg B \wedge \neg E)P(\neg B)P(\neg E) = \\ 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$

ساخت شبکه‌های بیزی

اغلب ، چند روش برای ساخت یک شبکه‌ی بیزی وجود دارد . در این مورد ، مهندسی دانش به کشف ارتباطات مستقل شرطی نیازمند است . والدهای یک گره باید آن‌ها یابی باشند که به طور مستقیم بر مقدار آن برتری دارند ؛ مثلا ، JohnCalls توسط لرزه دارای برتری می شود ، اما نه به طور مستقیم و تماس جان و ماری دارای برتری بر یکدیگر نمی باشند . به صورت صریح ، ما فرض می کنیم :

$$P(\text{MaryCalls} | \text{JohnCalls}, \text{Alarm}, \text{Earthquake}, \text{Burglary}) = P(\text{MaryCalls} | \text{Alarm})$$

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در زیر، روشی برای ساخت شبکه های بیزی آمده است:

(۱) یک مجموعه ای منظم از متغیر های X_1, \dots, X_n را انتخاب نمایید.

(۲) برای $i=1$ تا n کارهای زیر را انجام بدی:

را به شبکه اضافه کن . والد هایی از X_1, \dots, X_{i-1} را انتخاب کن که X_i

$$P(X_i | Parents(X_i)) = P(X_i | X_1, \dots, X_{i-1})$$

این نوع از انتخاب والد ها معانی عمومی زیر را به وجود می آورد:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | X_1, \dots, X_{i-1}) \quad (\text{قانون زنجیری})$$

$$= \prod_{i=1}^n P(X_i | Parents(X_i))$$

مثال - تصور کنید که ما مجموعه ای M, J, A, B, E را به ترتیب ، انتخاب می نماییم :



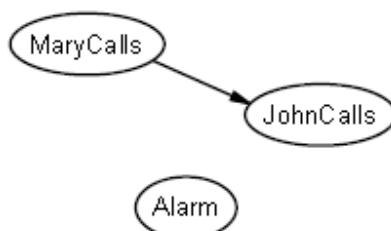
$$P(J|M) = P(J)? \text{ نه}$$

chain rule^۱

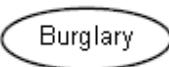
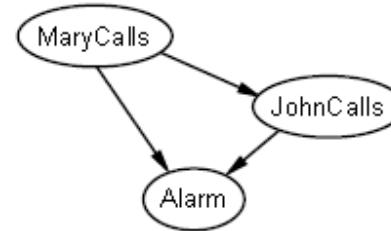
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

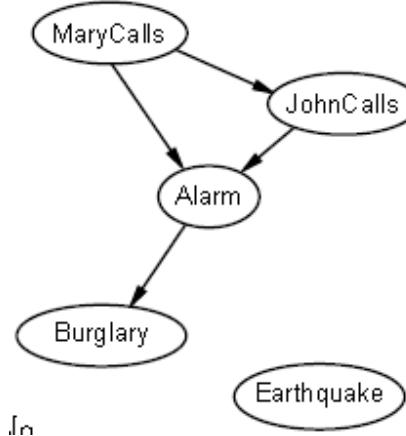


$P(A|J,M) = P(A|J)$? $P(A|J,M) = P(A)$? نه



$P(B|A,J,M) = P(B|A)$ ؟ نه

$P(B|A,J,M) = P(B)$ ؟ نه



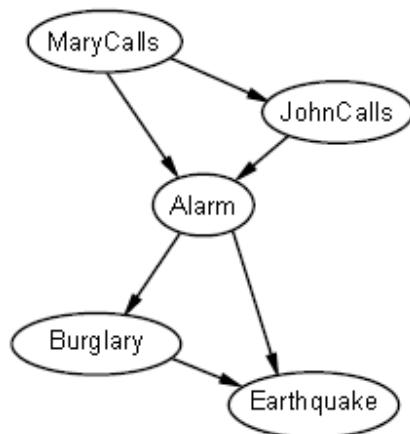
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$P(E|B,A,J,M) = P(E|A)$ ؟ نه

$P(E|B,A,J,M) = P(E|A,B)$ ؟ بله



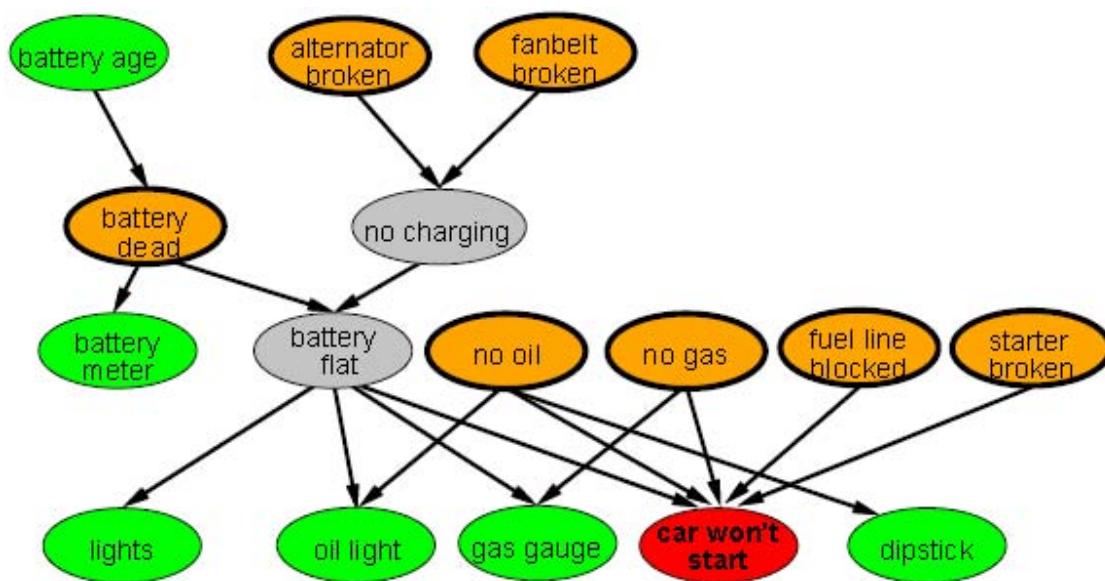
مثال : عیب اتومبیل

دلیل اوّلیه : ماشین روشن نمی شود . متغیرهای قابل آزمایش در شکل زیر با رنگ سبز نشان داده شده اند ؛ متغیرهایی که "شکسته اند و بنابراین آن ها را تعمیر نماییم " با رنگ نارنجی در شکل زیر نمایش داده شده اند ؛ برای متغیر های مخفی (خاکستری) ، قطعات وابسته را چک نمایید و به این طریق پارامترها را کاهش دهید .

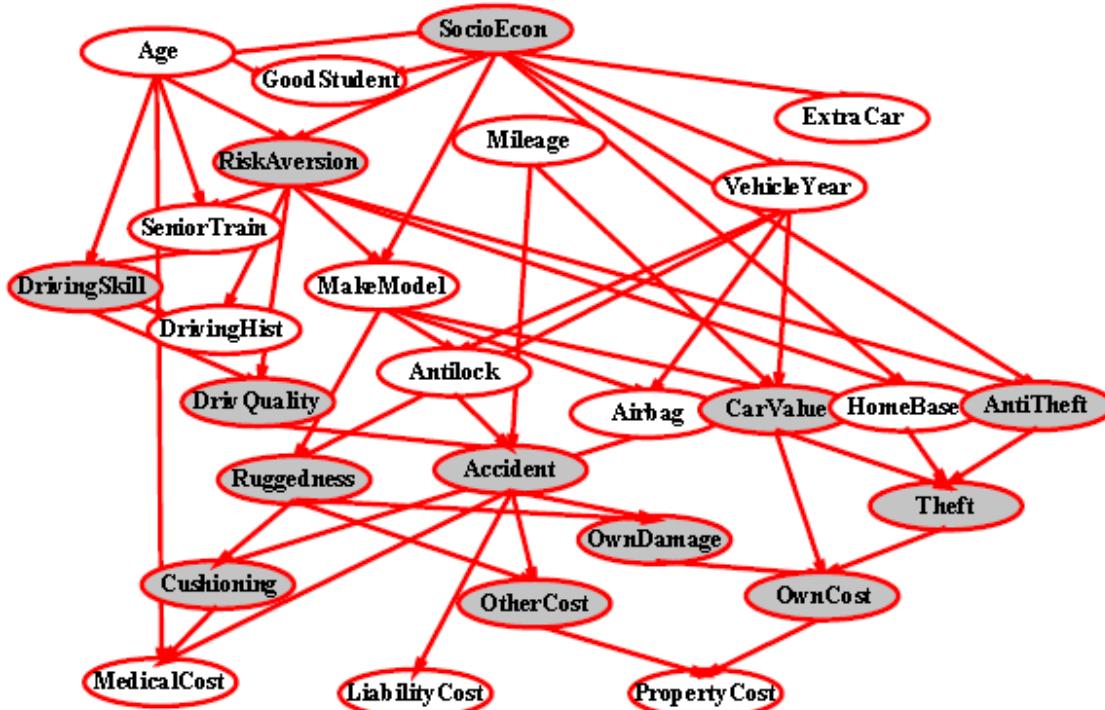
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



مثال : بیمه ای اتومبیل



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ساخت یک شبکه

اول، ریشه ها را به وجود بیاورید؛ دوم، متغیرهای مستقیم را اضافه نمایید؛ سوم، فرزندان مستقیم آن را اضافه نمایید؛ به این ترتیب، شما یک مدل سبی^۱ را به وجود آورده اید. تخمین ها با این روش به مراتب آسان تر به دست می آیند. برای ساخت می توانیم از اثر به سبب هم تلاش نماییم، اما در این حالت، شبکه به مراتب پیچیده تر خواهد بود.

کاربردهای شبکه های بیزی

عبارتند از: تشخیص، که به طور گسترده ای در محصولات میکروسافت استفاده می شود؛ تشخیص طبی؛ فیلتر نمودن نامه های الکترونیکی که ناشناس می باشند^۱؛ کاربرد در سیستم های خبره (کنترل وسائل، مانیتورینگ) و کنترل روبوتیک.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل پانزدهم

استنتاج در شبکه های بیزی

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

. {JohnCalls,MaryCalls} را متغیرهای مدرکی که دیده شده اند قرار دهید ؛ به عنوان مثال { JohnCalls,MaryCalls } را متغیر پرس و جو قرار دهید ؛ به عنوان مثال ، Burglary و فرض کنید می خواهیم $P(X|E)$ را به دست آوریم .

J	M	$P(B ...)$
T	T	?

کاربردهای شبکه های بیزی

کاربرد اصلی یک شبکه بیزی : ارایه ی مشاهدات جدید و محاسبه ی توانایی های جدید برخی (یا همه ی) تصویرات ^۱ .

beliefs ^۱

مترجم: سهراب جلوه گر

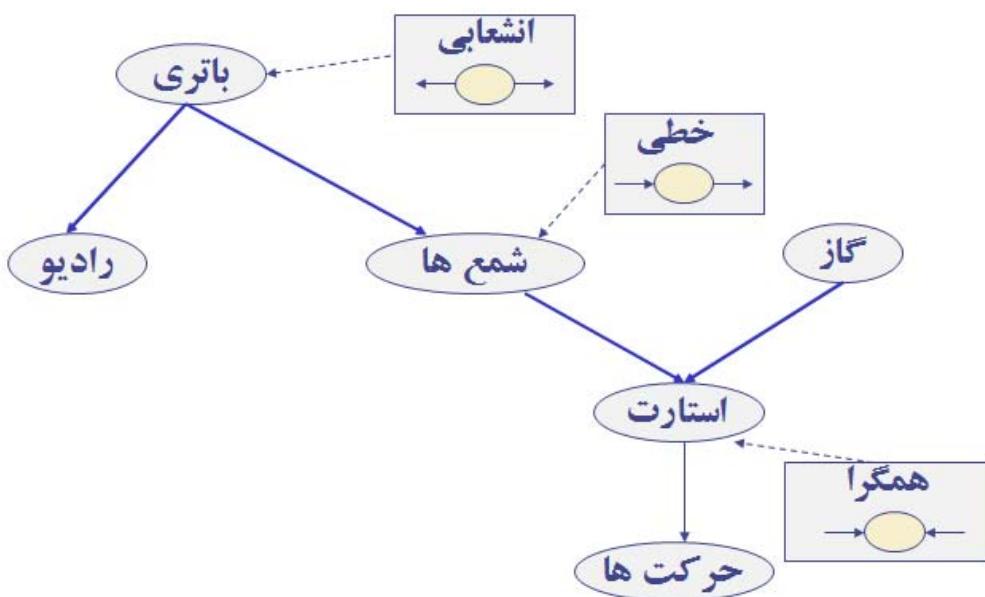
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

کاربرد دیگر: ارایه‌ی توانایی یک تصور و این که کدام مشاهده را باید برای به وجود آوردن بیشترین تغییر در توانایی تصور به کار ببریم.

انواع گره‌های موجود در یک مسیر



با داشتن یک مجموعه‌ی E از گره‌های مدرک، دو گره که توسط یک مسیر غیر مستقیم به هم وصل شده‌اند در صورتی مستقل هستند که دارای یکی از سه شرط زیر باشند:

- ۱- یک گره‌ی موجود در مسیر به صورت خطی^۱ و در E باشد.
- ۲- یک گره‌ی موجود در مسیر به صورت انشعابی^۲ و در E باشد.
- ۳- یک گره‌ی موجود در مسیر به صورت همگرا باشد و نه این گره و نه هیچ یک از فرزندانش در E باشند.

linear^۱

diverging^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گاز و رادیو با توجه به مدرک شمع ها^۱ مستقل هستند. گاز و رادیو با توجه به مدرک بانتری مستقل هستند.

استنتاج در شبکه های بیزی

در ساده ترین حالت داریم:



$$P(B) = P(a)P(B|a) + P(\sim a)P(B|\sim a)$$

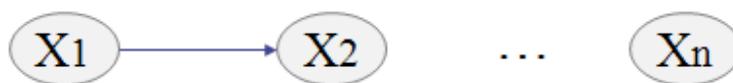
$$P(B) = \sum_A P(A)P(B|A)$$

حال سوال این جاست که اگر شکل زیر را داشته باشیم، آن گاه $P(C)$ برابر چیست؟



اگر یک زنجیره به صورت زیر داشته باشیم پیچیدگی زمانی برای محاسبه $P(X_n)$ چه قدر است

؟



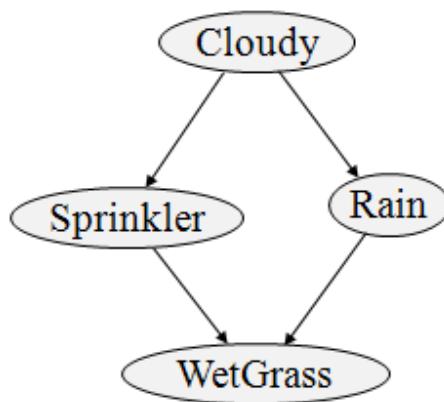
مثال:

Spark plugs^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



$$P(W) = \sum_{R,S,C} P(w|r,s)P(r|c)P(s|c)P(c)$$

$$= \sum_{R,S} P(w|r,s) \sum_C P(r|c)P(s|c)P(c)$$

اگر $f_c(R,S)$ را در نظر بگیریم داریم :

$$P(W) = \sum_{R,S} P(w|r,s) f_c(r,s)$$

الگوریتم در حال محاسبه ای احتمال ها به صورت تکی نمی باشد ، بلکه در حال محاسبه ای تمام موارد است . دو ایده برای جلوگیری از انفجار نمایی محاسبه ها وجود دارد :

- ۱ - با توجه به ساختار شبکه ای بیزی ، برخی از زیر عبارت ها در محل های اتصال فقط به تعداد کمی از متغیرها وابسته اند .
- ۲ - با یکبار محاسبه ای آن ها و نگهداری نتایج ، ما می توانیم از به وجود آمدن آن ها به صورت نمایی در بسیاری از موارد جلوگیری نماییم .

روش حذف متغیر

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

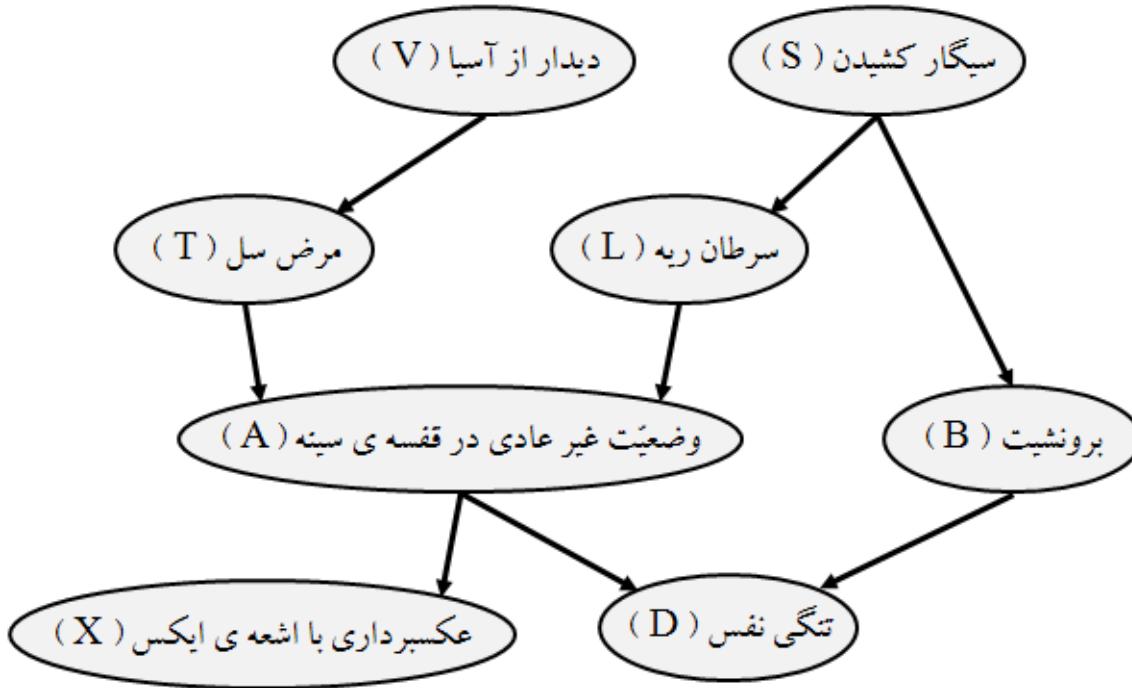
ایده‌ی اصلی این است که پرس و جو را به صورت زیر بنویسیم:

$$P(X_n, e) = \sum_{x_k} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i)$$

و به صورت تکراری کارهای زیر را انجام دهیم:

- ۱- تمام عبارت‌های نامریوط را به خارج از داخلی ترین جمع انتقال دهیم.
- ۲- داخلی ترین جمع را انجام دهیم و عبارت جدید را به دست آوریم.
- ۳- عبارت جدید را در ضرب وارد نماییم.

یک مثال پیچیده تر



مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ما می خواهیم احتمال تنگی نفس ($P(D)$) را محاسبه نماییم . باید v ، s ، x ، t ، l ، a ، b را حذف نماییم . عامل های اوّلیه عبارتند از :

$$P(v)P(s)P(t | v)P(l | s)P(b | s)P(a | t, l)P(x | a)P(d | a, b)$$

برای حذف v داریم :

$$f_v(t) = \sum_v P(v)P(t | v)$$

در نتیجه داریم :

$$f_v(t)P(s)P(l | s)P(b | s)P(a | t, l)P(x | a)P(d | a, b)$$

توجه کنید که $f_v(t) = P(t)$. در حالت کلی ، نتیجه ای حذف لزوماً یک عبارت احتمالی نمی باشد

برای حذف s داریم :

$$f_s(b, l) = \sum_s P(s)P(b | s)P(l | s)$$

در نتیجه :

$$f_v(t)f_s(b, l)P(a | t, l)P(x | a)P(d | a, b)$$

در جمع بندی بر روی نتایج s که دارای یک عامل و دو آرگومان $f_s(b, l)$ است ، در حالت کلی نتیجه ای حذف ممکن است یک تابع با چند متغیر باشد .

برای حذف X داریم :

مترجم: شهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$f_x(a) = \sum_x P(x | a)$$

در نتیجه داریم:

$$f_v(t) f_s(b, l) f_x(a) P(a | t, l) P(d | a, b)$$

توجه کنید که $f_x(a)$ برای همه مقدارهای a برابر با یک می باشد.

برای حذف t داریم:

$$f_t(a, l) = \sum_t f_v(t) P(a | t, l)$$

در نتیجه داریم:

$$f_s(b, l) f_x(a) f_t(a, l) P(d | a, b)$$

برای حذف l داریم:

$$f_l(a, b) = \sum_l f_s(b, l) f_t(a, l)$$

در نتیجه داریم:

$$f_l(a, b) f_x(a) P(d | a, b)$$

برای حذف a و b داریم:

$$f_a(b, d) = \sum_a f_l(a, b) f_x(a) P(d | a, b) \quad f_b(d) = \sum_b f_a(b, d)$$

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ما حالا فهمیدیم که حذف متغیر به صورت یک رشته از عملیات **بازنویسی** می باشد ، محاسبه می واقعی در مرحله ای حذف انجام می شود و محاسبه ، وابسته به ترتیب حذف می باشد .

رسیدگی به مدارک

$P(L, V = t, S = f, D = V = t, S = f, D = t)$ را داریم و می خواهیم t را محاسبه نماییم ؛ با نوشتن عامل ها شروع می کنیم :

$$P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b)$$

از آنجایی که ما می دانیم که $V = t$ می باشد ، نیازی به حذف V نداریم و به جای این کار می توانیم عامل های $P(V)$ و $P(T|V)$ را به صورت زیر جایگزین نماییم :

$$f_{P(V)} = P(V = t) \quad f_{p(T|V)}(T) = P(T | V = t)$$

که این ها اجزای مناسب عامل های اصلی را که مدارک بیان می کنند انتخاب می کنند . توجه کنید که $f_{p(V)}$ یک ثابت است و بنابراین در حذف دیگر متغیرها ظاهر نمی شود .

عامل های اوّلیه پس از تنظیم مدرک ها به صورت زیر خواهند بود :

$$f_{P(v)}f_{P(s)}f_{P(t|v)}(t)f_{P(l|s)}(l)f_{P(b|s)}(b)P(a|t,l)P(x|a)f_{P(d|a,b)}(a,b)$$

پس از حذف x داریم :

$$f_{P(v)}f_{P(s)}f_{P(t|v)}(t)f_{P(l|s)}(l)f_{P(b|s)}(b)P(a|t,l)f_x(a)f_{P(d|a,b)}(a,b)$$

با حذف t داریم :

$$f_{P(v)}f_{P(s)}f_{P(l|s)}(l)f_{P(b|s)}(b)f_t(a,l)f_x(a)f_{P(d|a,b)}(a,b)$$

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پس از حذف a داریم:

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_a(b, l)$$

با حذف b داریم:

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_b(l)$$

الگوریتم حذف متغیر

تصوّر کنید X_1, X_2, \dots, X_m رشته‌ای از متغیرهای غیر پرس و جو باشند:

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j P(X_j | Parents(X_j))$$

برای I مساوی با m تا ۱ کارهای زیر را انجام بده:

- در مجموع یابی برای X_i فقط عامل‌هایی که X_i را بیان می‌کنند، نگهدارید.
- عامل‌ها را ضرب کنید، یک عامل را که شامل یک شماره i برای هر مقدار متغیرهای مذکور X_i هستند را به دست آورید.
- جمع را انجام دهید و یک عامل f را که شامل یک عدد برای هر مقدار متغیرهای ذکر شده که شامل X_i نمی‌باشند را به دست آورید.
- عامل ضرب شده را در جمع جایگزین نمایید.

پیچیدگی حذف متغیر

فرض کنید در یک مرحله‌ی حذف ما عبارت زیر را محاسبه می‌کنیم:

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$f_x(y_1, \dots, y_k) = \sum_x f'(x, y_1, \dots, y_k)$$

$$f'_x(x, y_1, \dots, y_k) = \prod_{i=1}^m f_i(x, y_{1,1}, \dots, y_{1,l_i})$$

که به $m \cdot |\text{Val}(X)| \cdot \prod_i |\text{Val}(Y_i)|$ ضرب نیاز دارد. برای هر مقدار x, y_1, \dots, y_k ، ما به m ضرب نیاز داریم.

روش های استنتاج

به دو دسته ای استنتاج دقیق^۱ و استنتاج تقریبی^۲ تقسیم می شوند. استنتاج در زنجیره های ساده ، حذف متغیر و الگوریتم های طبقه بندی درختی^۳ ، جزو روش های استنتاج دقیق هستند و شبیه سازی احتمالی یا تصادفی^۴ یا روش های نمونه^۵ برداری و روش های زنجیره ای مارکوف مونت کارلو^۶ جزو روش های استنتاج تقریبی می باشند.

شبیه سازی تصادفی مستقیم

فرض کنید که شما دارای مقادیری برای برخی از زیرمجموعه های متغیرها (G) هستید و می خواهید مقدارهایی را برای متغیرهای ناشناخته (U) به دست آورید. تعداد زیادی از نمونه ها را از شبکه ای

Exact inference^۱

Approximate inference^۲

Clustering tree algorithms^۳

Stochastic simulation^۴

sampling methods^۵

Markov chain Monte Carlo methods^۶

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



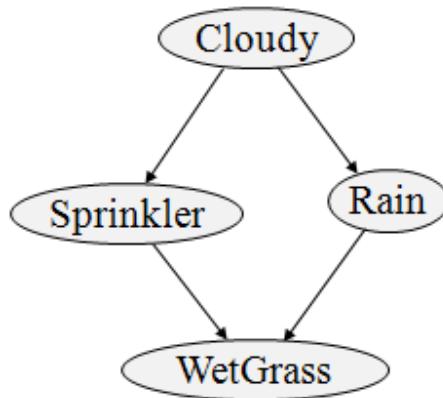
هوش مصنوعی

بیزی به صورت تصادفی به وجود بیاورید؛ نمونه ها را برای همه ای متغیرها به وجود آورید؛ از متغیرهای ریشه شروع کنید و این کار را به صورت مستقیم (رو به جلو) ^۱ ادامه دهید.

رد نمونه برداری: نمونه هایی که با مقدارهای G سازگارند را نگه دارید.

مقدارها را برای U به کار ببرید تا احتمال های تخمین زده شده را به دست آورید. دقیقت نتیجه ها به تعداد نمونه ها بستگی دارد.

مثال: با داشتن شکل زیر، $P(\text{WetGrass} | \text{Cloudy})$ چیست؟



$$P(\text{WetGrass} | \text{Cloudy}) = P(\text{WetGrass} \wedge \text{Cloudy}) / P(\text{Cloudy})$$

۱. کارهای زیر را N بار انجام دهید:

 - ۱.۱. فرض کنید به تصادف هوا ابری ^۲ است.
 - ۱.۲. برای هر فرض ابری بودن هوا، آپاش ^۱ و باران ^۲ را فرض کنید و سپس خیس بودن چمن ^۳ را فرض نمایید.

forward ^۱
Cloudy ^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

۲. نسبت تعداد مواردی که خیس بودن چمن و ابری بودن هوا درست است را در صورتی ابری بودن هوا درست باشد محاسبه نمایید.

وزن احتمال ^۴

ایده‌ی این روش این است که نمونه‌هایی که لازم است در مرحله‌ی اوّل رد شوند را تولید ننمایید؛ فقط از متغیرهای ناشناخته‌ی Z نمونه بگیرید و هر نمونه را با داشتن مدرک E با توجه به احتمالی که رخ می‌دهد مورد ارزیابی قرار دهید.

الگوریتم زنجیره‌ی مارکوف مونت کارلو

در زنجیره‌ی مارکوف هر مورد تولید شده در نمونه، وابسته به نمونه‌ی قبلی می‌باشد و در روش مونت کارلو از روش نمونه برداری آماری استفاده می‌شود.

یک پیمایش تصادفی را در میان فضای متغیر انجام دهید و آمارها را در موقع جلو رفتن نگه دارید؛ برای این کار با یک نمونه‌ی تصادفی که با متغیرهای مدرک سازگار است شروع کنید و در هر مرحله، برای برخی از متغیرهای غیر مدرک که با دیگر انتساب‌های جاری سازگارند، به صورت تصادفی از مقدارش نمونه برداری نمایید. با داشتن نمونه‌های کافی، زنجیره‌ی مارکوف مونت کارلو (MCMC) یک تخمین دقیق از توزیع درست مقدارها را ارایه می‌کند.

Sprinkler ^۱

Rain ^۲

WetGrass ^۳

Likelihood weighting ^۴

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل شانزدهم

شناخت سخن یا

سخن شناسی^۱

(به طور خلاصه)

Speech recognition^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- سخن به صورت استدلال احتمالی^۱
- صدای سخن^۲

سخن شناسی

عملیات لازم برای توانمندسازی یک کامپیوتر برای شناسایی و واکنش دادن به صدای سخن به وجود آمده در سخن انسان می باشد^۳. به عنوان تعریفی دیگر، سخن شناسی یا تشخیص صدا^۴، توانایی سیستم های کامپیوتری برای دریافت سخن به صورت ورودی و پردازش بر روی آن یا بیان آن به صورت نوشته می

speech as probabilistic inference^۱

speech sounds^۲

Babylon / آکسفورد^۳

voice recognition^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

باشد. کاربردهای عملی سخن شناسی، شامل سیستم های پایگاه داده - پرس و جو^۱ و سیستم های بازیابی اطلاعات^۲ می باشد. سخن شناسی دارای کاربرد در روبوتیک و مخصوصاً توسعهٔ روبوت هایی که می توانند "بشنوند" می باشد.^۳

سخن به صورت استدلال احتمالی

سیگال های سخن، پارازیت دار (اغتشاش دار)^۴، متغیر و مبهم^۵ می باشند. شبیه ترین ترتیب کلمات و سیگنال سخن ارایه شده چیست؟، برای این کار از قانون بیز استفاده نمایید:

$$P(\text{Words}|\text{signal}) = \alpha P(\text{signal}|\text{Words})P(\text{Words})$$

توجه کنید که، سخن، به مدل صوتی^۶ و مدل زبانی^۷، تجزیه می شود. **کلمات**^۸، ترتیب وضعیت های پنهان می باشند و **سیگنال**^۹، ترتیب مشاهده می باشد. در مورد اصوات^{۱۰}، باید بدانیم که، تمام سخنان بشر ترکیبی از ۴۰ الی ۵۰ صوت می باشد. برای یک سطح میانی وضعیت های مخفی شدهٔ

database-query systems^۱

information retrieval systems^۲

^۳ دایره المعارف بریتانیکا

noisy^۴

ambiguous^۵

acoustic model^۶

language model^۷

words^۸

signal^۹

phones^{۱۰}

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

میان کلمات و سیگنال ها داریم ، مدل صوتی^۱ = مدل تلفظ^۲ + مدل صوت^۳. آرپابت^۴ تعیین شده برای انگلیسی آمریکایی به صورت زیر می باشد :

[iy]	<u>beat</u>	[b]	<u>bet</u>	[p]	pet
[ih]	bit	[ch]	<u>Chet</u>	[r]	<u>rat</u>
[ey]	<u>bet</u>	[d]	<u>debt</u>	[s]	<u>set</u>
[ao]	<u>bought</u>	[hh]	<u>hat</u>	[th]	<u>thick</u>
[ow]	<u>boat</u>	[hv]	<u>high</u>	[dh]	<u>that</u>
[er]	B <u>ert</u>	[l]	<u>let</u>	[w]	<u>wet</u>
[ix]	ros <u>es</u>	[ng]	<u>sing</u>	[en]	<u>button</u>
:	:	:	:	:	:

برای مثال ، برای کلمه " ceiling " داریم [s iy l ih ng] / [s iy l ix ng] / [s iy l en] :

acoustic model^۱

pronunciation model^۲

phone model^۳

ARPAbet^۴

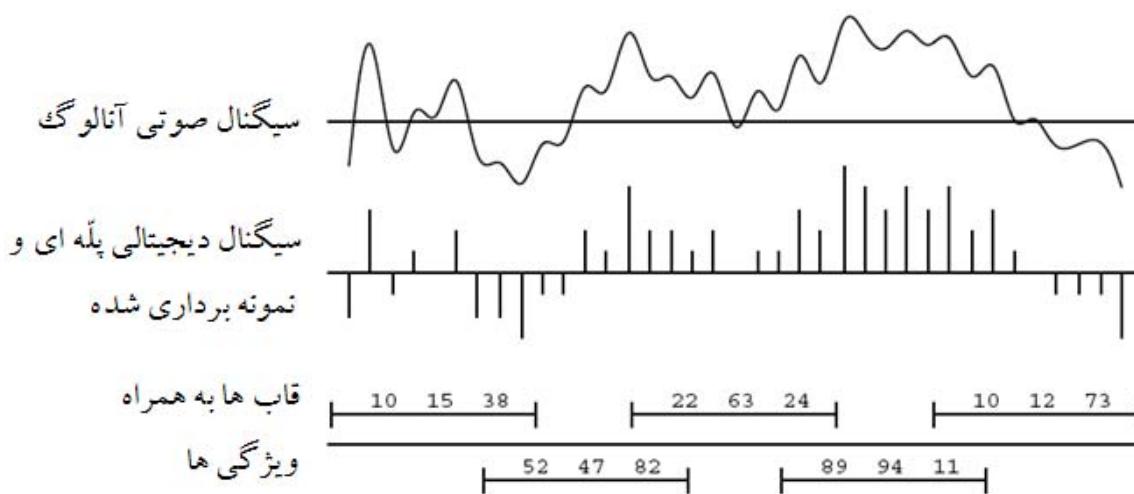
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

صدای سخن - سیگال خام میکروفون به صورت یک تابع یا عملکرد از زمان می باشد ؟ در پردازش ، قاب های ۳۰ میلی ثانیه ای روی هم می افتد و همگی به وسیله ای پستی و بلندی اشان توصیف می شوند .



پستی و بلندی های قاب با فرمت های معمولی می باشند و قله^۱ ها دارای طیف^۲ قوی می باشند .

اصوات سه حالته^۳ : هر صوت دارای سه وجه می باشد (آغاز^۴ ، وسط^۵ ، پایان^۶) ، به عنوان مثال ، حرف [t] دارای ابتدای آرام^۷ ، وسط قوی^۸ ، انتهای خشن^۹ می باشد .

peak^۱

spectrum^۲

Three-state phones^۳

Onset^۴

Mid^۵

End^۶

silent Onset^۷

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل هفدهم

تصمیم گیری های عاقلانه^۳ (تیوری تصمیم گیری^۴)

explosive Mid^۱

hissing End^۲

Rational decisions^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

• تقدم های عاقلانه ^۱

• تسهیلات ^۲

• پول ^۳

• تسهیلات چند خصوصیتی ^۰

• شبکه های تصمیم گیری ^۴

• ارزش اطلاعات ^۱

Decision theory ^۱

Rational preferences ^۰

utilities ^۰

Money ^۰

Multiatribute utilities ^۰

Decision networks ^۰

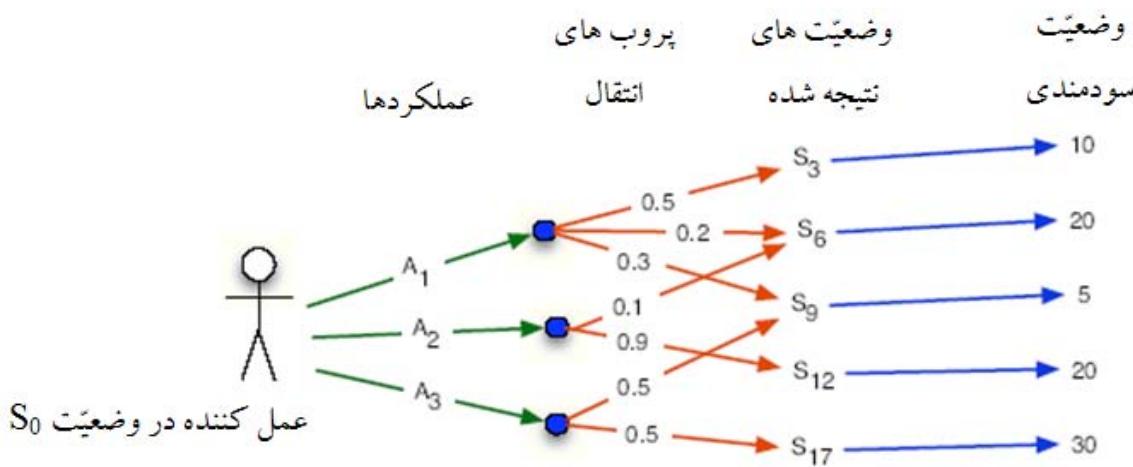
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تصمیم گیری عاقلانه

تصمیم گیری ای عاقلانه است که منجر به انجام کار درست می شود؛ کار درست را انجام دهید؛ یعنی، عملی که سود مورد انتظار را بیشینه می نماید. اصل بیش ترین سودمندی مورد انتظار^۱، می گوید که یک عامل هوشمند باید کاری که سودمندی مورد انتظار^۲ عامل را بیشینه می کند انتخاب نماید.



$$EU(A_1) = (0.5)(10) + (0.2)(20) + (0.3)(5) = 10.5$$

$$EU(A_2) = (0.1)(20) + (0.9)(20) = 20, EU(A_3) = (0.5)(5) + (0.5)(30) = 17.5$$

Value of information^۱

Maximum Expected Utility (MEU)^۲

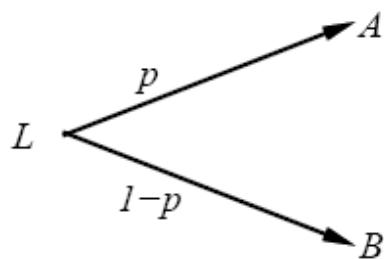
Expected Utility (EU)^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تقدّم ها – تیوری سودمندی^۱ به تقدّم ها توجه می کند. در این مورد ، حالت ها ، جایزه ها (A ، B و غیره) می باشند. نتیجه های ممکن ، مثل قرعه کشی هایی با جایزه های نامعلوم می باشند.



قرعه کشی L برابر است با : $[p,A; (1-p),B]$

یادداشت : عبارت $A \succ B$ مقدم است ; عبارت $A \sim B$ ~ معنی ، بی تفاوتی میان A و B و عبارت $\overset{\curvearrowleft}{A} \sim \overset{\curvearrowright}{B}$ معنی ، B بر A مقدم نمی باشد .

اولویّت های عاقلانه

اولویّت های عامل هوشمند ،تابع محدودیّت ها می باشد که محدودیّت ها عبارتند از :

$$(A \succ B) \vee (B \succ A) \vee (A \sim B) \quad ^2$$

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C) \quad ^3$$

$$A \succ B \succ C \Rightarrow \exists p [p, A; 1-p, C] \sim B \quad ^1$$

utility theory^۱
Orderability^۲
Transitivity^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

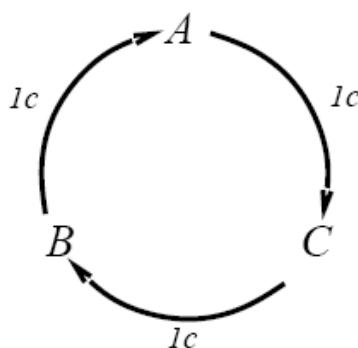
$$A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$$

تعویض پذیری ^۱

$$A \succ B \Rightarrow (p \geq q \Leftrightarrow [p, A; 1-p, B] \succsim [q, A; 1-q, B])$$

یکنواختی ^۲

تخلّف از محدودیت ها، منجر به نمودها یا آشکارسازی نامعقولة نه می شود، برای مثال: یک عامل با اولویت های لازم می تواند منجر به از دست دادن همه ی پول شود:



در صورتی که $C \succ B$ باشد، آن گاه یک عامل که دارای C می باشد باید یک سنت ^۴ برای دریافت B پرداخت نماید. در صورتی که $A \succ B$ باشد، آن گاه یک عامل که دارای B می باشد باید یک سنت برای دریافت A پردازد. در صورتی که $C \succ A$ باشد، آن گاه یک عامل که دارای A می باشد باید یک سنت برای دریافت C پردازد.

بیشینه کردن سودمندی (تسهیلات) مورد انتظار

Continuity ^۱

Substitutability ^۲

Monotonicity ^۳

cent که معادل یک صدم دلار آمریکایی می باشد ^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اصل های تیوری سودمندی در مورد اولویت ها صحبت می کنند. تابع سودمندی از اصل های سودمندی پیروی می کند.

قضیه (رمزی^۱، ۱۹۳۱؛ ون نیومن^۲ و مورگنسترن^۳، ۱۹۴۴)؛ اگر اولویت های یک عامل از اصل های سودمندی، پیروی کند، در این صورت یک تابع با مقدار حقیقی U وجود دارد که:

اصل بیش ترین سودمندی مورد انتظار^۴: سودمندی یک قرعه کشی، مجموع سودمندی های هر نتیجه و مطابق با احتمال زیر می باشد:

$$U([p_1, S_1; \dots; p_n, S_n]) = \sum_i p_i U(S_i)$$

که با قانون تصمیم گیری با بیش ترین سودمندی مورد انتظار، هم تراز است.

قانون بیش ترین سودمندی مورد انتظار (MEU): عملکردی را که سود مورد نیاز را بیشینه می کند را انتخاب نمایید.

نکته : یک عامل می تواند کاملاً عاقلانه باشد (سازگار با قانون MEU باشد) بدون حتی ارایه یا به کارگیری سود و احتمالات. مثلاً، یک جدول انتخابی^۵ برای تیکتاکتوی کامل.

Ramsey^۱

von Neumann^۲

Morgenstern^۳

MEU principle^۴

lookup table^۵

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سودمندی پول

تیوری سودمندی دارای ریشه هایی در اقتصاد است. آیا پول می تواند به صورت یک معیار سودمندی مورد استفاده قرار گیرد؟.

به عنوان مثال ، فرض کنید که شما در یک مسابقه تلویزیونی مبلغ یک میلیون دلار ، برنده می شوید . برای ادامه ی بازی شما باید یک سکه را به هوا پرتاب نمایید؛ اگر شیر آمد ، مبلغ سه میلیون دلار می گیرید ، ولی اگر خط آمد تمام پولی را که قبلاً برده اید از دست می دهید . در این موقع ، شما چه کاری را انجام خواهید داد؟ . شما می توانید در این مورد ، بازی را به صورت زیر بیان نمایید :

$$L : [0.5, \text{Heads}(triple); 0.5, \text{Tails}(loose)]$$

$$EU(L) = 0.5 \times 3,000,000 + 0.5 \times 0 = 1,500,000$$

استاندارد نزدیک به صرفه های بشر

اگر در یک بازی قمار روسی^۱ شانس شما برای کشته شدن ، یک میلیونیم باشد ، آیا در این بازی شرکت می کنید . قمار روسی ، یک بازی خطرناک شانسی است که در آن یک نفر گلوله ای را با استفاده از یک هفت تیر (که به سوی سر یک شخص نشانه رفته است) شلیک می نماید .^۲

پول - پول به صورت یک تابع سودمند رفتار نمی کند . با یک L که به صورت شانسی ارایه شده ، با مقدار مالی مورد انتظار $EMV(L)$ معمولاً $U(EMV(L)) < U(L)$ می باشد ، توجه کنید که افراد مخالف ریسک هستند .

^۱ Russian roulette

^۲ فرنگ آکسفورد

مترجم: سهراب جلوه گر

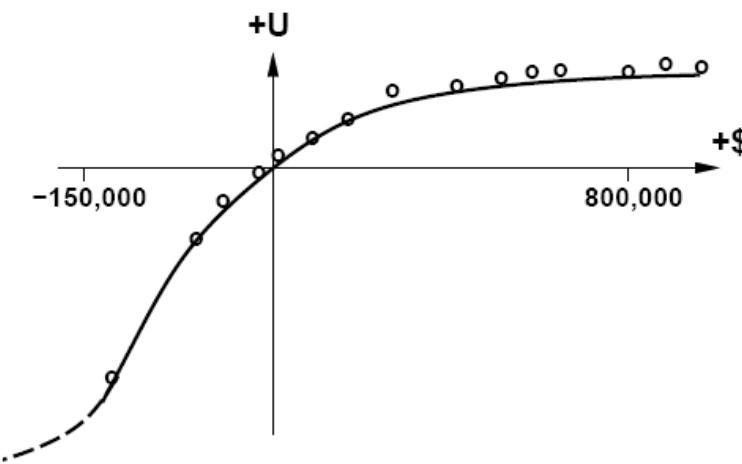
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

منحنی سودمند^۱: در یک مسابقه، با چه مقدار احتمال p ، در بی تفاوتی میان یک پاداش x و دارای شансی برای بردن M بزرگ هستم؟

داده های تجربی^۲ معمولی و قیاسی با ریسک متعادل به صورت زیر رفتار می کنند:



اهمیت پول

ارزش (سودمندی) پول، وابسته به مقدار پولی است که شما دارید. هزار دلار برای یک فرد فقیر ارزشی بیش تر از ارزش همان مقدار پول برای یک فرد ثروتمند دارد. اگر S اندازه ای ارزش پول برای یک فرد باشد، $U(S+\$1000)-U(S)$ برای فرد فقیر ارزشی به مراتب بزرگ تر از ارزشی که برای یک فرد پولدار دارد خواهد داشت. توجه کنید که $U(Money)$ یک تابع غیرخطی می باشد که شیب آن برای همه مقادیر S یکسان نمی باشد.

برنولی^۱: $U(Money) \propto \log(Money)$ تقریباً برابر است با

utility curve^۱
empirical^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بیزاری از خطر (ریسک) و بیمه^۲

یک شанс بخت آزمایی $L = [p, \$M; (1-p), \$0]$ داده شده است، چه مقدار پول شما می گیرید تا در این بخت آزمایی شرکت نکنید؟ در صورتی که $X < EMV(L)$ باشد، آن گاه شما از ریسک، بیزار خواهید بود؛ شما یک مقدار کم تراز بخت آزمایی را دریافت خواهید کرد تا از ریسک بخت آزمایی اجتناب نمایید. X ، برابری حتمی^۳ بخت آزمایی نام دارد. در این مورد، اجرت بیمه^۴ برابر است با $.EMV(L) - X$.

فرض نمایید که شما دارای یک خانه به ارزش یک میلیون دلار هستید، فرض کنید که احتمال از دست دادن خانه براثر آتش سوزی برابر یک هزارم باشد. بنابراین شанс شامل مقادیر ممکن که شما باید پردازید می باشد. $EMV(L_h) = \$1,000$ و $L_h = [p = .001, \$M; (1 - p) = .999, \$0]$. شما چه مقدار X حاضرید به بیمه پردازید تا از این شанс آتش سوزی جلوگیری نمایید؟ باقی مانده $X - EMV(L_h)$ مقداری است که شما به بیمه خواهید پرداخت. اگر شما خیلی از ریسک متنفر باشید، آن گاه X کم خواهد بود و شما هزینه‌ی زیادی برای اجرت بیمه جهت جلوگیری از عواقب اقتصادی فردی خواهید پرداخت.

یک مثال واقعی بخت آزمایی

در یک مسابقه‌ی بخت آزمایی اگر داشته باشیم:

$$L = [p = .02, \$10; q = .000005, \$1,000,000; (1 - p - q) = .9799995, \$0]$$

برابر یک دلار می باشد. آن گاه $EMV(L)$ چه قدر است؟ چه هنگام معقولانه است که یک بليط بخت آزمایی را بخریم؟

Bernoulli^۱

insurance^۲

certainty equivalent^۳

insurance premium^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

جواب: در ابتدا، هزینه‌ی بلیط را در عواقب بخت آزمایی به کار می‌بریم:

$$L = [p=.02; q=.0000005, 999,999; (1-p-q)=.9799995, -1]$$

$$EMV(L) = (.02)(9) + (.0000005)(999,999) + (.9799995)(-1) = \$-0.3$$

در صورتی که ما از هزینه‌ی بلیط، چشم پوشی نماییم

عقلانیت و سود بخت آزمایی

فرض کنید S_k ، وضعیت دارایی‌های جاری افراد باشد. در این صورت، سود بخت آزمایی براساس $U(S_{k+n})$ برای مقادیر مختلف n می‌باشد. با صرف نظر از هزینه‌ی بلیط:

$$U(L) = (.02)U(S_{k+10}) + (.0000005)U(S_{k+1,000,000})$$

(سود پس انداز هزینه‌ی بلیط) است یا نه، و این وابسته به U و K می‌باشد. در صورتی که $U(S_{k+1}) > U(L) > U(S_{k+10})$ باشد، خرید بلیط کاری معقولانه می‌باشد.

مزیت چند خصوصیتی بودن^۱

چگونه می‌توانیم توابعی سودمند با تعدادی متغیر X_1, X_n را به کار بگیریم؟ برای مثال، $U(DNC)$ چیست؟ چگونه می‌توانیم توابع سودمندی که دارای تقدّم در رفتار هستند را ترکیب کنیم؟

ایده‌ی ۱: وضعیت‌هایی را تحت تصمیم‌گیری‌هایی که می‌توانند بدون شناسنامه‌ی کامل

$$U(x_1, \dots, x_n)$$

Multiattribute utility^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



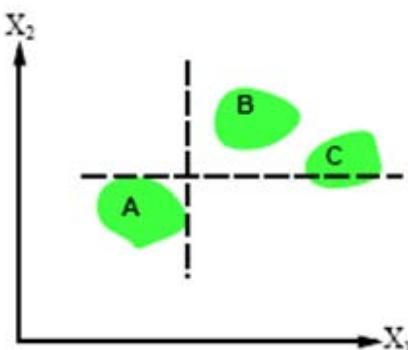
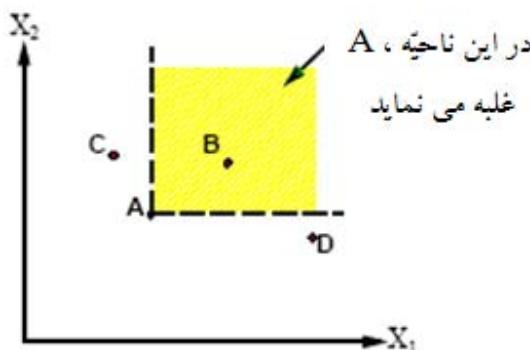
هوش مصنوعی

ایده‌ی ۲: انواع مختلف مستقل در اولویت‌ها را شناسایی نماییم و فرم‌های استاندارد نتیجه را برای $U(x_1, \dots, x_n)$ به دست آوریم.

تسلط (نفوذ) سخت^۱

معمولًاً خصوصیات را به گونه‌ای تعریف می‌کنیم که U یکنواخت باشد.

تعریف تسلط (نفوذ سخت): انتخاب B در صورتی نفوذ شدید بر انتخاب A دارد که $U(B) \geq U(A)$ و بنابراین $\forall i X_i(B) \geq X_i(A)$.



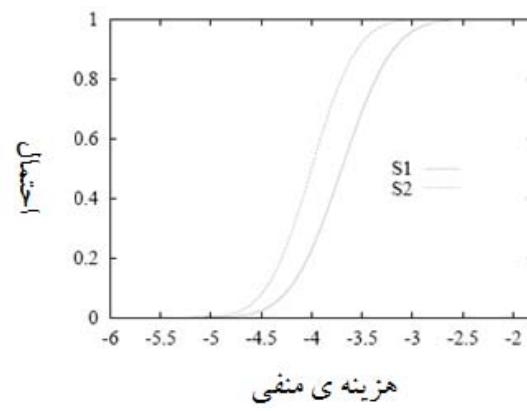
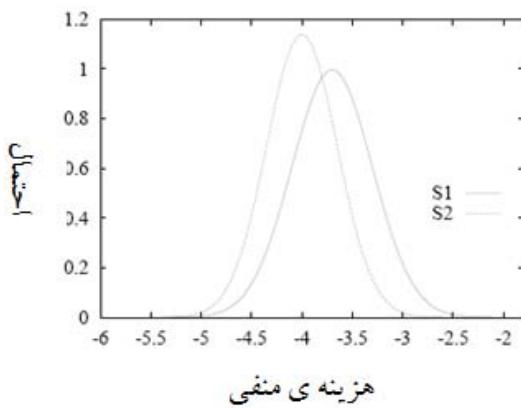
نفوذ اتفاقی

strict dominance^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



توزیع p_1 در صورتی بر توزیع p_2 اثر اتفاقی می گذارد که :

$$\forall t : \int_{-\infty}^t p_1(x) dx \leq \int_{-\infty}^t p_2(t) dt$$

در صورتی که U دارای اثر یکنواخت بر X باشد ، در این صورت A_1 با توزیع به دست آمده از p_1 ، بر A_2 ، با توزیع به دست آمده از p_2 اثر اتفاقی می گذارد اگر :

$$\int_{-\infty}^{\infty} p_1(x) U(x) dx \geq \int_{-\infty}^{\infty} p_2(x) U(x) dx$$

کاربرد چند خصوصیتی : در صورت وجود اثر تصادفی در همه‌ی خصوصیات ، چند خصوصیتی ، بهینه خواهد بود .

اثر اتفاقی ، اغلب می تواند بدون توزیعات دقیق و با استفاده از چگونگی منطق ، تعیین شود . به عنوان مثال ، هزینه‌ی تولید با افزایش فاصله شهر S_2 از شهر S_1 افزایش می یابد ، در نتیجه ، S_1 دارای اثر تصادفی بر S_2 از نظر هزینه می باشد . به طور مثال ، آسیب دیدگی (صدمه) با افزایش سرعت در لحظه‌ی تصادف افزایش می یابد . می توان رفتار شبکه‌ها را با اطلاعات اثر تصادفی تفسیر نمود : $X \xrightarrow{+} Y$ (X دارای اثرات مثبت بر

مترجم: سهراب جلوه گر

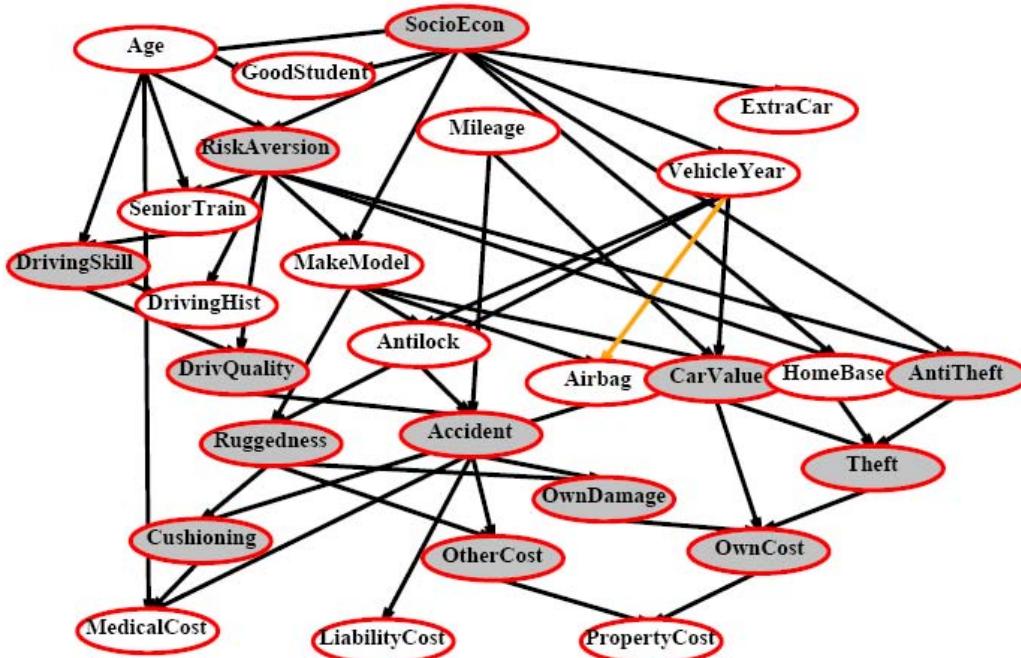
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

Y می باشد) دارای این معنی است که : برای هر مقدار Z از سایر والدهای Z متعلق به Y می باشد) دارای اثر اتفاقی بر (x_1, x_2) باشد . $P(Y | x_1, x_2) \Rightarrow P(Y | x_1, z)$

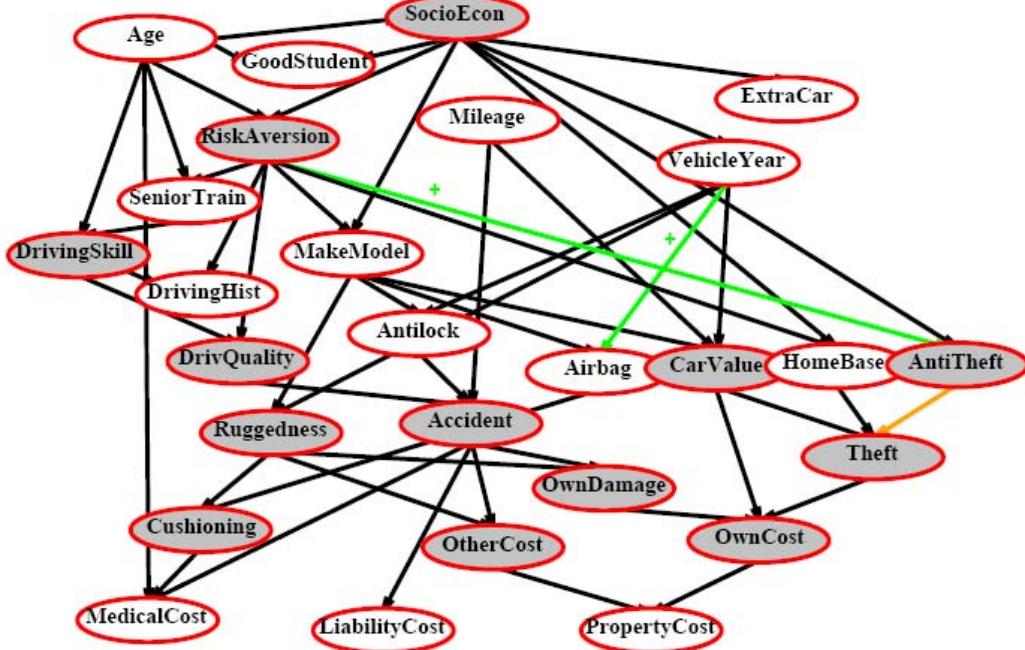
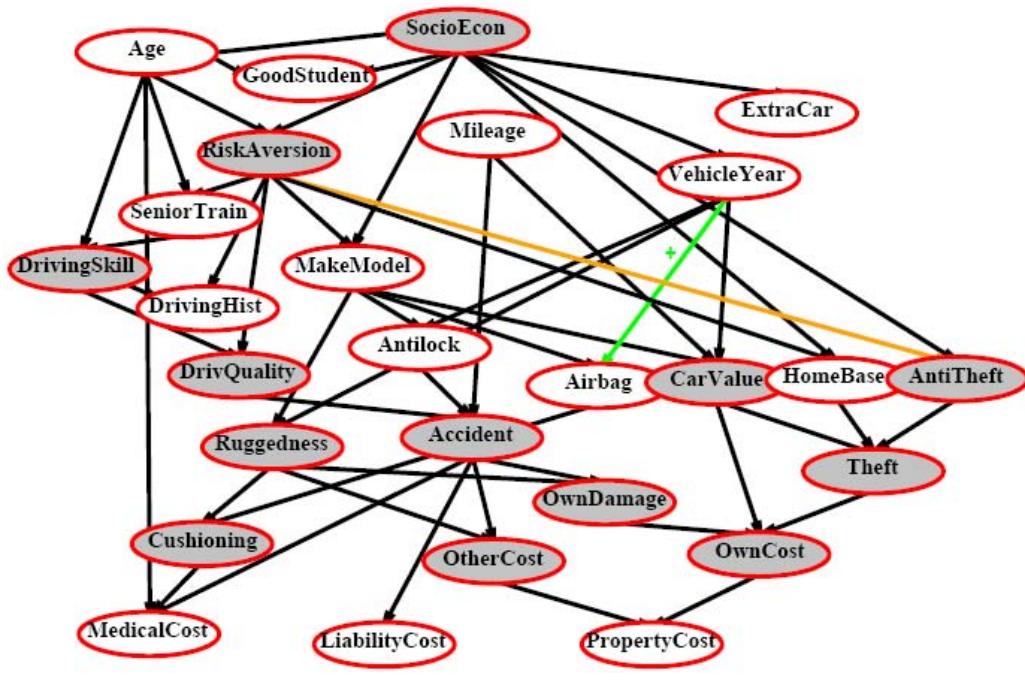
بر چسب قوس های + یا -



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

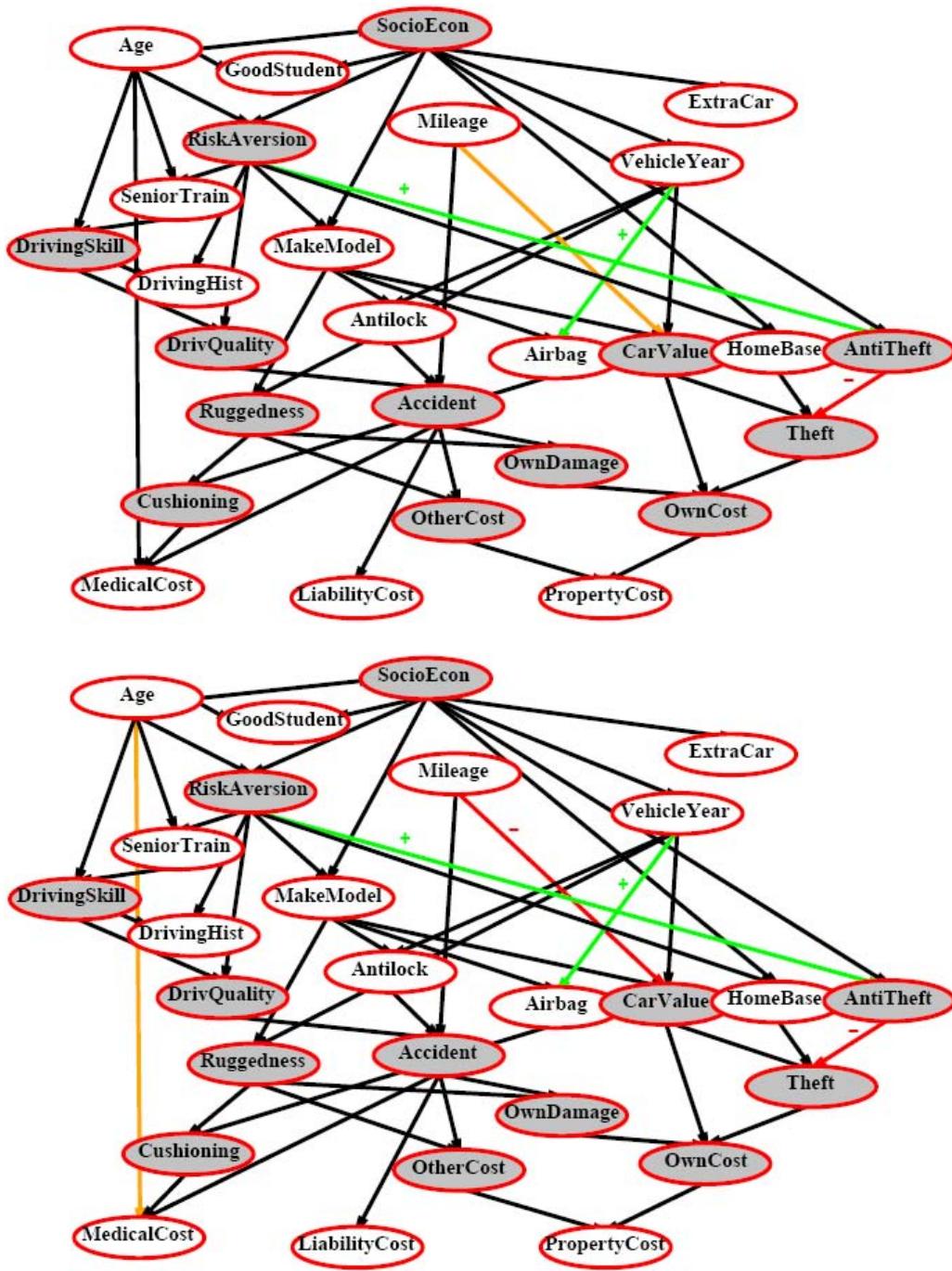
هوش مصنوعی



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی

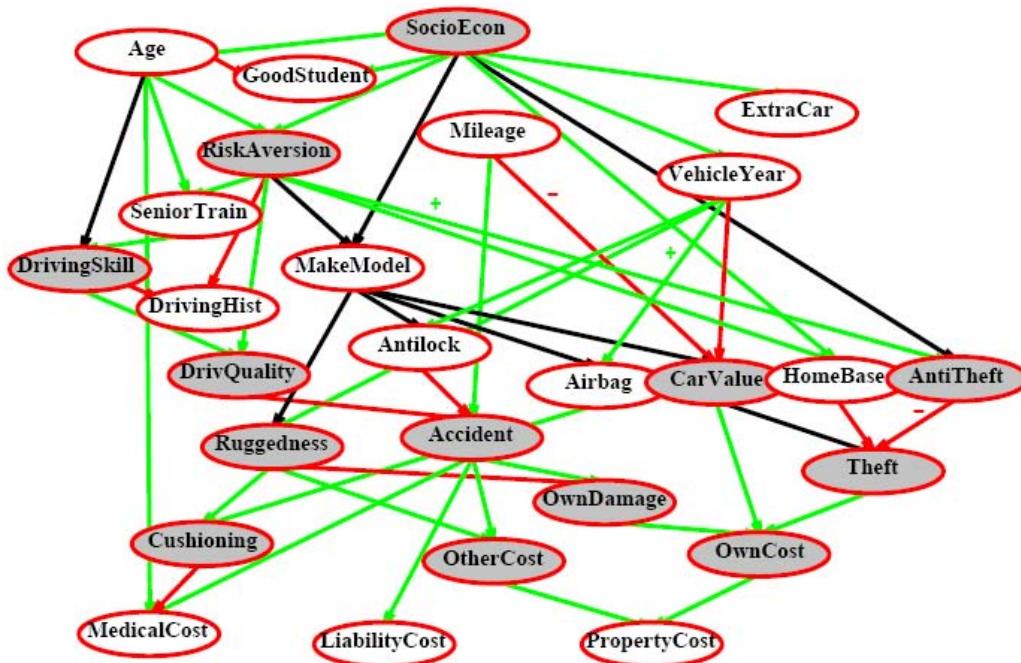


مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



اولویت ساختار: قطعی

از نظر اولویت، مستقل از X_3 هستند در صورتی که: اولویت میان $\langle x_1, x_2, x_3 \rangle$ و $\langle x'_1, x'_2, x'_3 \rangle$ وابسته به X_3 نباشد. مثلاً در $\langle \text{Noise}, \text{Cost}, \text{Safety} \rangle$

$\langle 4.6, 4.2, 20000 \rangle$ و $\langle 20000, 2006, 70000 \rangle$.

قضیه مستقل از لحاظ اولویت دو طرفه (لیون تیف^۱، ۱۹۴۷): اگر هر جفت از خصوصیات از لحاظ اولویت مستقل از متمم خود باشد، بنابراین هر زیر مجموعه از خصوصیات از لحاظ اولویت مستقل از متمم خود می باشد.

Leontief^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

قضیّه (دبرو^۱، ۱۹۶۰): اگر قضیّه‌ی از لحاظ اولویّت مستقل بودن دو طرفه برقرار باشد، در نتیجه تابعی به صورت جمع وجود دارد که:

$$V(S) = \sum_i V_i(X_i(S))$$

بنابراین به دست آوردن توابع تک خصوصیتی^۲، n تایی؛ اغلب با تقریب خوبی انجام می‌شود.

اولویّت ساختار: اتفاقی

- لازم است که اولویّت‌های بخت آزمایی را بدانیم: X دارای استقلال سودمندی^۳، Y می‌باشد، در صورتی که اولویّت‌های شانس در X وابسته به Y نباشند.

مستقل از لحاظ سودمندی بودن دو طرفه: هر زیر مجموعه مستقل از لحاظ سودمندی از مکمل خود می‌باشد، در نتیجه، تابع سودمندی به صورت ضرب وجود دارد که:

$$U = k_1 U_1 + k_2 U_2 + k_3 U_3 + k_1 k_2 U_1 U_2 + k_2 k_3 U_2 U_3 + k_3 k_1 U_3 U_1 + k_1 k_2 k_3 U_1 U_2 U_3$$

روال زیر برنامه‌ها و بسته‌های نرم افزاری برای تولید تست‌های اولویّت، خانواده‌هایی با استاندارد های متفاوت توابع سودمند را معرفی می‌نماید.

شبکه‌های تصمیم‌گیری

گره‌های عملکرد^۱ و گره‌های سودمند^۲ را به شبکه‌های رفتاری برای فعال ساختن تصمیم‌گیری منطقی اضافه نمایید.

Debreu^۱

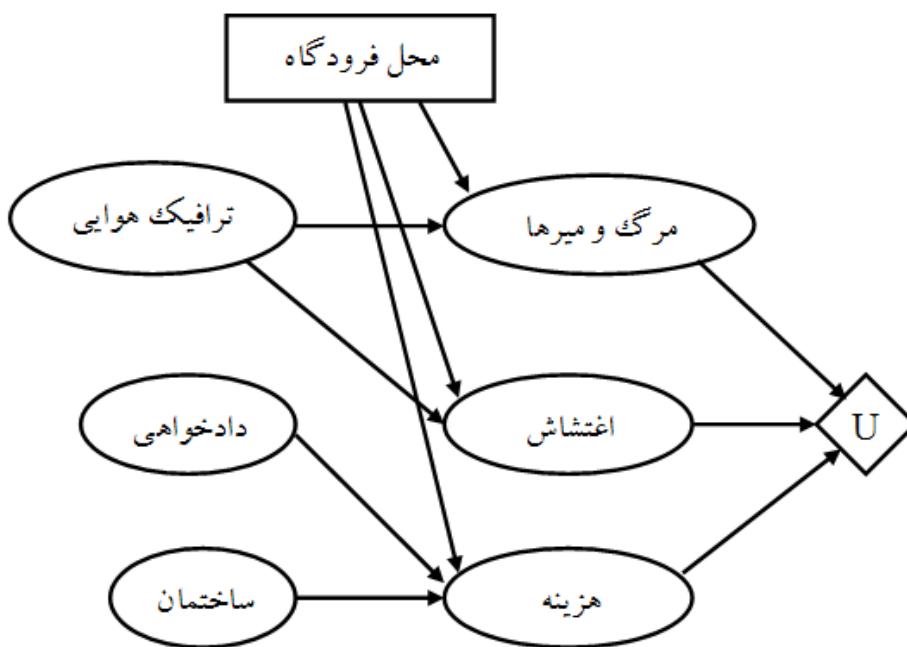
single-attribute^۲

utility-independent^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



الگوریتم :

برای هر مقدار گرهی عملکرد ،

مقدار مورد انتظار سودمندی عملکرد و ثبات (پایداری) گرهی ارایه شده را با استفاده از استنتاج

محاسبه نمایید

عملکرد MEU را برگردان

پایان الگوریتم

شبکه‌ی تصمیم‌گیری روبوت

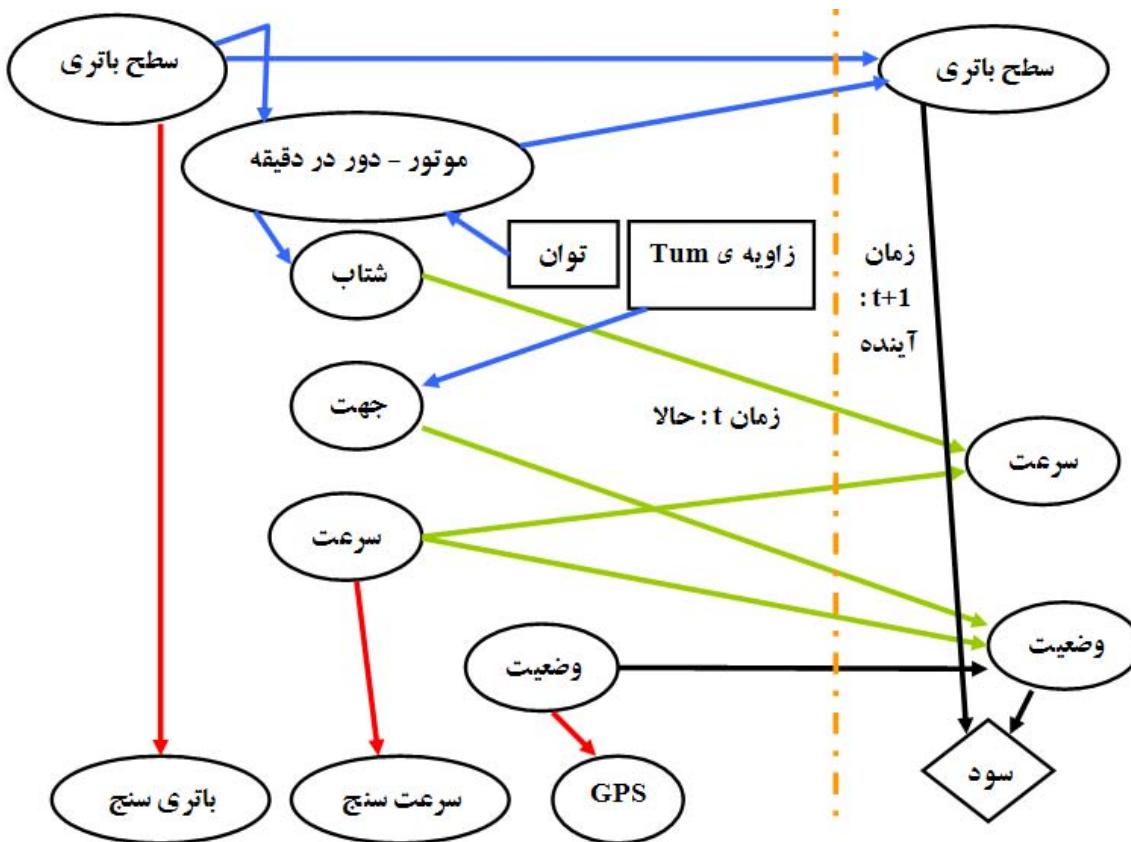
action nodes^۱
utility nodes^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



ارزیابی شبکه‌ی تصمیم‌گیری

۱. متغیرهای مدرک را برای وضعیت جاری سیستم قرار دهید.
 ۲. برای هر بردار انتساب های مقدار برای متغیرهای تصمیم‌گیری (به عنوان مثال ، فرمان یا توان) :
- (۱) گره های تصمیم‌گیری را به مقادیر آن ها انتساب دهید .

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

(۲) از استنتاج بیزی (دقیق یا تقریبی) برای محاسبه ای احتمال های قبلی همه
والدهای گره سود استفاده نمایید.

(۳) سود را براساس توزیع احتمال روی مقادیر والدها محاسبه نمایید.

بردار تصمیم گیری دارای بیش ترین سود را برگردان .۳

تصمیم گیری روبوت

۱. مقادیر شناخته شده را به متغیرهای مدرک نسبت دهید:

باتری سنج ، سرعت سنج و GPS^۱ (همه در زمان t)

۲. برای هر ترکیب ممکن از توان و زاویه ی چرخش ^۲ :

(۱) از استنتاج بیزی برای محاسبه ای توزیع احتمال برای سطح باتری و
وضعیت در زمان $t+1$ استفاده نمایید.

(۲) سپس آن توزیع ها را برای محاسبه ای سود به کار ببرید.

.۳. بهترین توان را به دست آورید.

نکته : از سود مقادیر آنلده ای طرح ^۱ سطح باتری و وضعیت به صورت پایه ای برای انتخاب
عملکردهای جاری ، استفاده نمایید.

^۱ مخفف Global Positioning System است و سیستمی ناوبری [مکان نمایی] است که با شبکه ای گستردۀ از ماهواره ها کار می کند .
^۲ turn-angle

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ارزش اطلاعات

- **ایده ۵:** مقدار به دست آمده برای هر جزء ممکن مدرک را محاسبه نمایید. این کار به طور مستقیم از شبکه‌ی تصمیم‌گیری می‌تواند انجام شود.

مثال: تمرین خرید صحیح روغن. از دو قوطی A و B، دقیقاً یکی محتوی روغن می‌باشد، قیمت را k در نظر بگیرید. احتمال هر کدام را 0.5 در نظر بگیرید. قیمت فعلی هر قوطی $k/2$ می‌باشد.
"متخصص" پیشنهاد بررسی A را می‌دهد. آیا A مناسب است؟

راه حل: ارزش مورد انتظار اطلاعات را محاسبه نمایید، که برابر است با، مقدار مورد انتظار بهترین عملکرد اطلاعات ارایه شده منهای مقدار مورد انتظار بهترین عملکرد بدون اطلاعات. عملکردها خرید A (Buy(A)) و خرید B (Buy(B)) می‌باشند. بدون اطلاعات، ارزش مورد انتظار هر عملکرد برابر است با:

بررسی ممکن است بگوید که "روغن در A است" یا "روغن در A نمی‌باشد" و احتمال 0.5 . برای هر کدام داده شده است! با داشتن اطلاعات، مقدار مورد انتظار برای بهترین عملکرد، برابر است با:

[$0.5 * \text{ارزش داده شده} \text{ی خرید A} + 0.5 * \text{ارزش خرید B}$] با فرض این که "روغن درون A نمی‌باشد" که برابر است با:

$$= (0.5 * k/2) + (0.5 * k/2) - 0 = k/2$$

بنابراین ارزش اطلاعات مورد انتظار برابر است با:

PROJECTED FUTURE^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

انتخاب اندازه گیری^۱

اطلاعات معمولاً مستقل نمی باشند . به دست آوردن اندازه گیری ها می تواند پرهزینه باشد : آزمایش های طبی ، تحلیل های شیمیایی اعماق دریا و احتیاج به اولویت بندی اندازه ها یا آزمایش ها داریم . برای این کار ، آن هایی که منجر به وضعیت های بالاترین سود مورد انتظار می شوند را انتخاب نمایید .

اندازه گیری ها ← اطلاعات وضعیت ها ← انتخاب عملکردها ← وضعیت های کلی جدید ← ارزشیابی سود^۲

از آنجایی که تعدادی از این اتصالات به صورت اتفاقی می باشند ، ما به توزیع احتمالی که هم نتایج اندازه گیری و هم عملکردهای منظم را دارا هستند رسیدگی می نماییم .

Measurement Selection^۱
Utility Assessment^۲

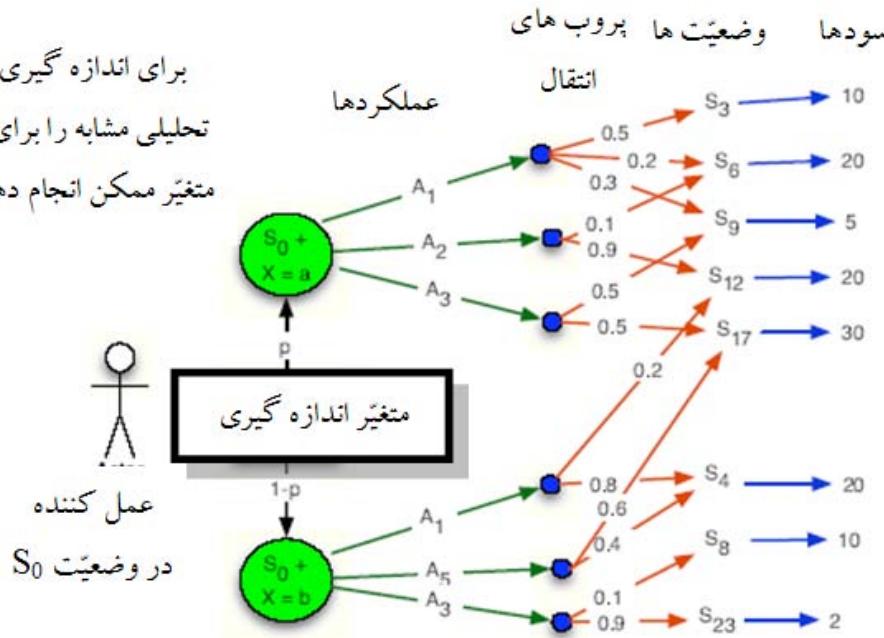
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برای اندازه گیری ،
تحلیلی مشابه را برای هر
متغیر ممکن انجام دهید .



فرمول کل - اگر برای مدرک جاری E ، بهترین عملکرد جاری، α باشد . نتیجه های عملکرد ممکن S_i و توان بالقوه i (پتانسیل) مدرک جدید E_j برابر است با :

$$EU(\alpha | E) = \max_a \sum_i U(S_i) P(S_i | E, a)$$

تصور نمایید که ما می دانیم $\alpha_{e_{jk}}$ را انتخاب می کنیم .

$$EU(\alpha_{e_{jk}} | E, E_j = e_{jk}) = \max_a \sum_i U(S_i) P(S_i | E, a, E_j = e_{jk})$$

E_j ، یک متغیر تصادفی است که مقدار فعلی آن ناشناخته می باشد ، در نتیجه ، باید مقادیر آن را حول تمام ممکن آن محاسبه نماییم :

$$VPI_E(E_j) = (\sum_k P(E_j = e_{jk} | E) EU(\alpha_{e_{jk}} | E, E_j = e_{jk})) - EU(\alpha | E)$$

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مخفف *Value of Perfect Information* VPI می باشد ، به معنی ارزش اطلاعات کامل می باشد .

خصوصیات ارزش اطلاعات کامل

- **فا منفی^۱** می باشد :

- **جمع ناپذیری^۲** ، در مثال ، توجه کنید که E_j دوبار آمده است :

$$VPI_E(E_j, E_k) \neq VPI_E(E_j) + VPI_E(E_k)$$

مستقل از نظم بودن^۳ :

$$VPI_E(E_j, E_k) = VPI_E(E_j) + VPI_{E, E_j}(E_k) = VPI_E(E_k) + VPI_{E, E_k}(E_j)$$

توجه: در زمانی که بیش تر از یک بخش مدرک می تواند جمع آوری شود ، بیشینه کردن ارزش اطلاعات کامل برای هر قسمت ، برای انتخاب یکی از آن ها که همیشه هم بهینه نمی باشد منجر به این می شود که جمع کردن مدرک ها تبدیل به یک مسئله ای تصمیم گیری ترتیبی شود .

کیفیت (چگونگی) رفتارها

اگر انتخاب ، قابل مشاهده می باشد ، اطلاعات دارای ارزش کمی می باشند . اگر انتخاب ، غیرقابل مشاهده می باشد ، اطلاعات ، به دلیل بالا بودن گوناگونی سودمندی (U) ، دارای ارزش زیادی می باشند .

nonnegative^۱

nonadditive^۲

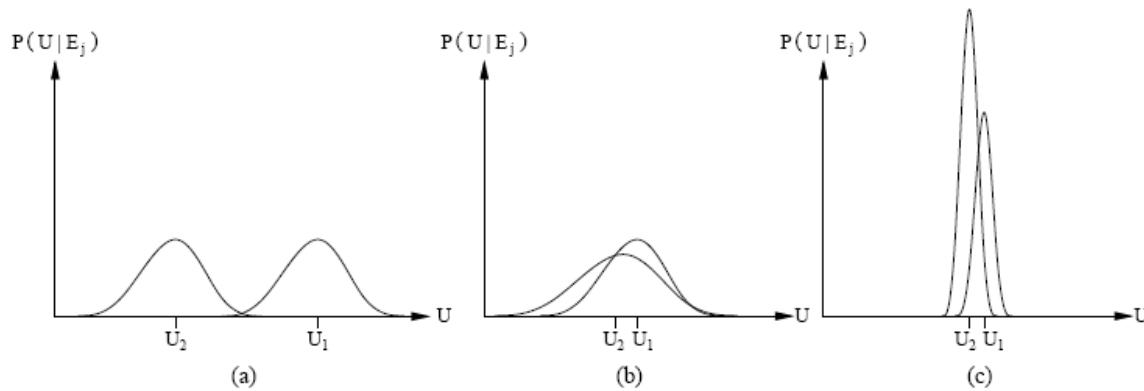
order-independent^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اگر انتخاب غیرقابل مشاهده می باشد ، اطلاعات ، به دلیل پایین بودن گوناگونی سودمندی (U) ، دارای ارزش کمی می باشند .



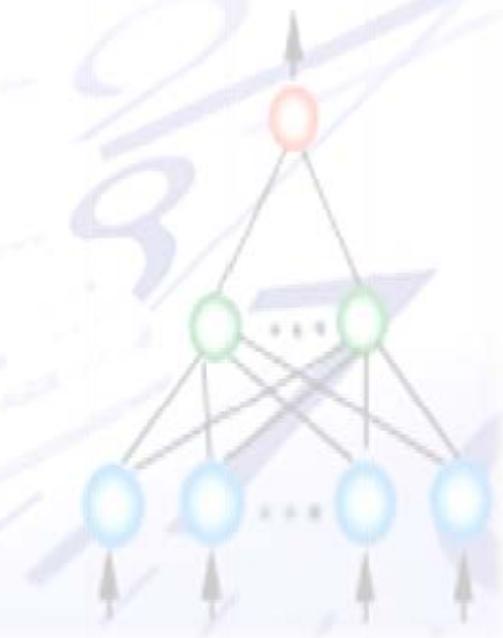
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فصل هیجدهم

شبکه های عصبی^۱



neural networks^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

دیوس مطالب

- مغز^۱
- شبکه های عصبی
- پرسپترون^۲ ها
- پرسپترون های چند سطحی
- کاربردهای شبکه های عصبی

شبکه های عصبی

brains^۱
Perceptron^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بسیاری از آنچه که تا کنون ما مطالعه کرده ایم می تواند به صورت هوش مصنوعی نمادین یا نشانه ای^۱ طبقه بندی شود؛ در هوش مصنوعی نمادین، تأکید بر سمبول‌ها و ارتباطات میان آن‌ها می‌باشد. به عنوان مثال، جستجو، منطق، درخت‌های تصمیم‌گیری و برنامه‌ریزی جزء هوش مصنوعی نمادین می‌باشد. سمبول‌ها برای رفتار هوشمند، کلیدی می‌باشند. شبکه‌های عصبی بر رفتار نمادین تمرکز می‌نماید. توجه کنید که رفتار هوشمند از ارتباط اجزای ساده به وجود می‌آید.

زیست‌شناسی و علم کامپیووتر

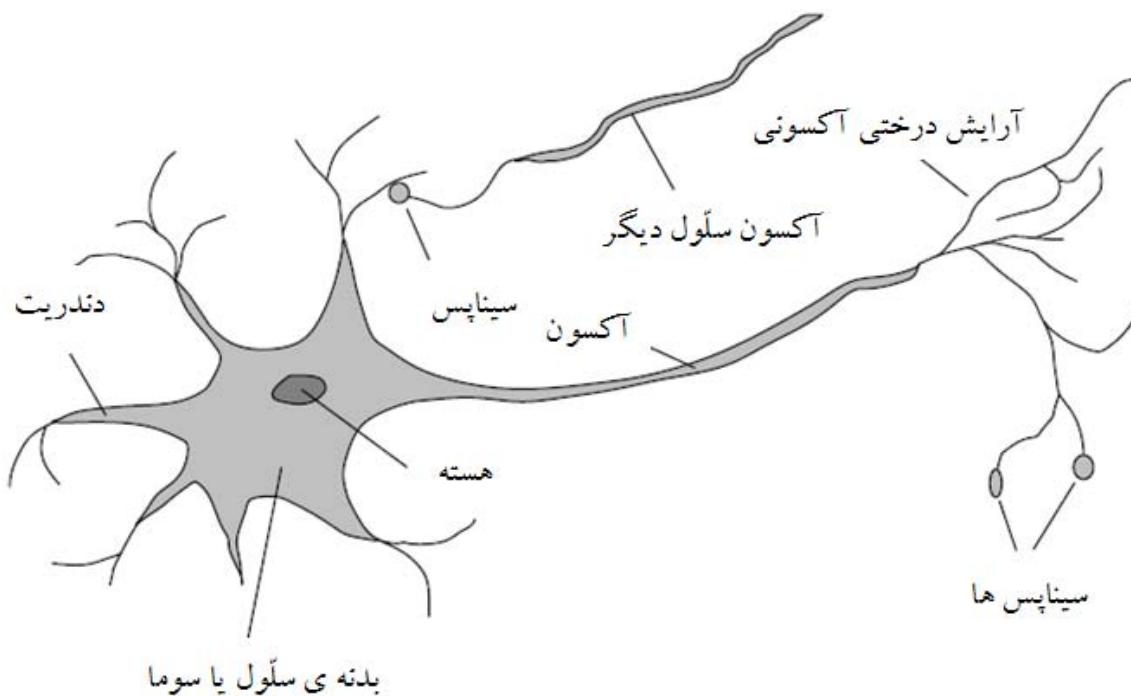
در نورون‌های^۲ زیست‌شناسی، سیگنال‌های دریافت شده توسط دندانهای^۳ دریافت می‌شوند و از طریق آکسون^۴ به دیگر نورون‌ها می‌رسند. [در انتقال سیگنال‌ها]، علامت دهنده^۵ و شلیک^۶، خیلی پیچیده می‌باشند. فکر و رفقار در تعامل هزاران نورون به وجود می‌آیند. محققان هوش مصنوعی اغلب جذب توسعه‌ی الگوریتم‌های مؤثر می‌باشند؛ مانند الگوریتم‌های ژنتیکی، ما بر روی ایده‌هایی که کاملاً طبیعی هستند کار می‌کنیم و آن‌هایی که مفید هستند را به کار می‌بندیم.

symbolic AI ^۱
neurons ^۲
dendrites ^۳
axon ^۴
signaling ^۵
firing ^۶

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



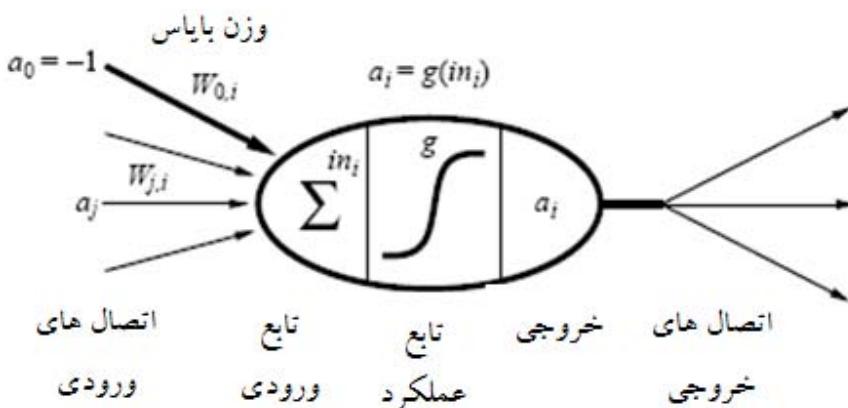
شبکه های عصبی محاسباتی

شبکه های عصبی از گره ها تشکیل شده اند. این گره ها توسط اتصال هایی جدا از آکسون ها به هم متصل می باشند. هر اتصال دارای یک فشار است که قدرت سیگنال (علامت) را مشخص می کند. هر گره دارای یکتابع عملکرد غیرخطی است. خروجی گره به صورت یکتابع توزیع شده، جمع ورودی ها را مدیریت می نماید.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



تابع عملکرد

در حالت کلی، از هر تابع غیرخطی می‌توان استفاده نمود؛ عمومی‌ترین تابع، دو تابع به صورت زیر می‌باشند:

- ۱- **تابع مرحله‌ای (تابع آستانه)** – خروجی‌ها، اگر ورودی مثبت باشد، یک می‌باشد و در غیر این صورت صفر می‌باشد.
- ۲- **تابع S شکل (سیگموید)**^۱ یا **نقلي**^۲: $\frac{1}{1+e^{-x}}$ ؛ که به صورت مشتق پذیر و پيوسته^۳ می‌باشد و دارای تغییر سریع در نزدیکی آستانه می‌باشد و به صورت تدریجی به سمت بی‌نهایت می‌رود.

مثال‌ها:

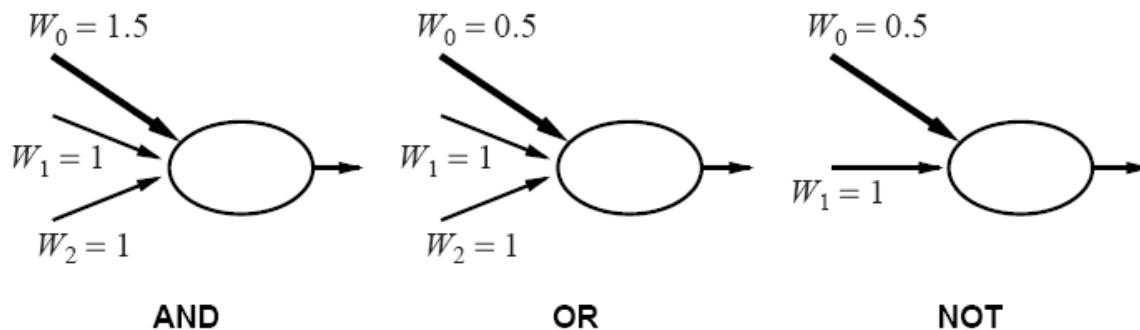
sigmoid^۱
logistic^۲
continuously differentiable^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



شبکه های عصبی می توانند به سادگی برای انجام برخی از عملیات استاندارد منطقی با استفاده از تابع عملکرد آستانه ای ساخته شوند؛ که در این مورد، باید شما بسته به تابعی که می خواهید، آستانه را تغییر دهید.

انواع گره ها

ما می توانیم سه نوع گره را تشخیص دهیم:

- ۱- گره های ورودی
- ۲- گره های خروجی
- ۳- گره های پنهان

ما همچنین می توانیم انواع شبکه های زیر را داشته باشیم:

- ۱- شبکه های با تغذیه ای مستقیم^۱: علامت (سیگنال) ها در یک جهت حرکت می کنند و بدون دور یا سیکل می باشند. شبکه های تغذیه ای مستقیم، توابع را تکمیل می کنند و وضعیت داخلی ندارند.
- ۲- شبکه های بازگشت کننده^۱: در انتشار علامت (سیگنال)، دور یا سیکل وجود دارد.

feed-forward networks^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ما در ابتدا روی شبکه های تغذیه ای مستقیم تمرکز می نماییم.

شبکه های تغذیه ای مستقیم به شکل تخمین زنده ای توابع

شبکه های عصبی با تغذیه ای مستقیم در یک خانواده از الگوریتم ها به نام تخمین زنده های تابع غیرخطی^۱ قرار می گیرند؛ خروجی یک شبکه ای عصبی تابعی از ورودی هایش می باشد. تابع فعال سازی غیرخطی، ارایه ای توابع پیچیده را اجازه می دهد. با تطبیق دادن وزن ها، ما تابعی که ارایه می شود را تغییر می دهیم. شبکه های عصبی اغلب برای تخمین توابع پیچیده از داده ها استفاده می شوند.

طبقه بندی با شبکه های عصبی

شبکه های عصبی، طبقه بندی را هم خیلی خوب انجام می دهند؛ ورودی ها را به یک یا بیش تر خروجی تبدیل می نمایند و دامنه ای خروجی به کلاس هایی مجزا تقسیم می شوند و برای کارهای آموزشی در جایی که نمی دانیم "در جستجوی چه هستیم" ، خیلی مفید می باشد، مثل: صورت شناسی^۲، دست خط شناسی^۳ و رانندگی یک اتومبیل.

مثال تغذیه ای مستقیم

recurrent networks^۱

nonlinear function approximators^۲

face recognition^۳

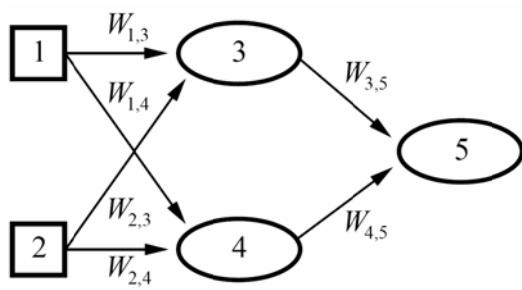
handwriting recognition^۴

مترجم: سهراب جلوه گر

ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی



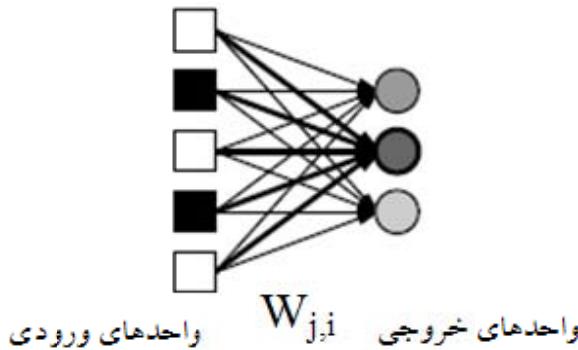
شبکه‌ی تغذیه‌ی مستقیم، یک خانواده‌ی پارامتریندی شده از توابع غیرخطی است، داریم:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

پرسپترون‌ها

پرسپترون، شبکه‌ای از نورون‌های است که در آن یک ارتباط دارای وزن میان خروجی برخی از نورون‌ها و ورودی دیگر نورون‌ها وجود دارد؛ به عنوان تعریفی دیگر، ابزاری است که می‌تواند الگوهای یادگیری (آموزشی) را تشخیص دهد.^۱

پرسپترون‌های تک لایه‌ای

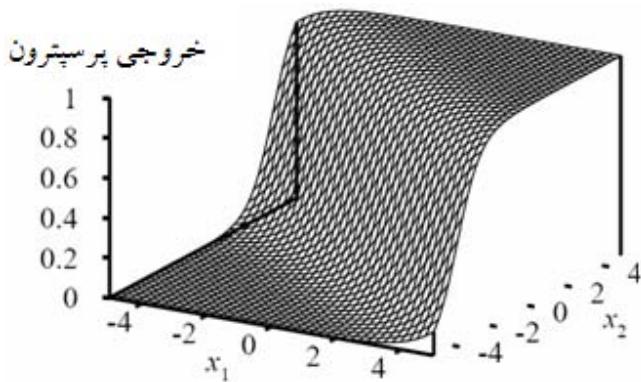


Babylon / English - English^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



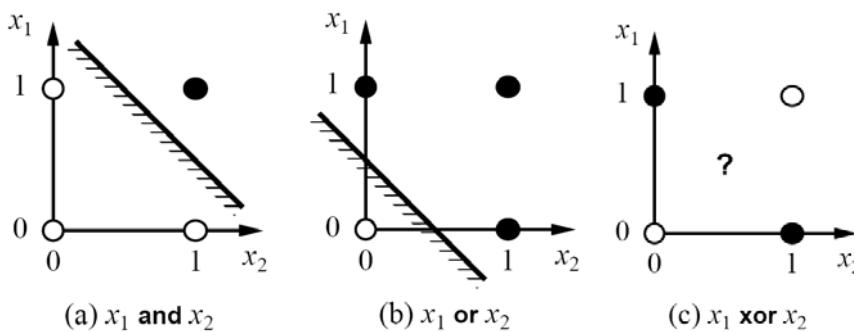
هوش مصنوعی



پرسپترون های تک لایه ای ، ساده ترین شکل شبکه‌ی عصبی می‌باشد ؟ برای فهم ، ساده می‌باشد ولی برای محاسبه ، محدود می‌باشد و در آن‌ها هر واحد ورودی به صورت مستقیم به یک یا بیش تر واحد خروجی متصل شده است . اگر $w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0$ باشد ، آن‌گاه $o(x_1, \dots, x_n) = 1$. همه‌ی واحدهای خروجی به طور مستقل عمل می‌نمایند – نه به صورت اشتراکی .

توجه نمایید که یک پرسپترون با تابع مرحله‌ای g ، می‌تواند AND ، OR ، NOT و ... را ارایه نماید ولی XOR را نمی‌تواند بیان نماید و در فضای ورودی ، یک مجزاً کننده‌ی خطی^۱ را ارایه می‌نماید :

$$W \cdot x > 0 \text{ یا } \sum_j W_j x_j > 0$$



linear separator^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

توان نمایش پرسپترون ها

خروجی شبکه، $\sum_{j=0}^n W_j i_j > 0$ می باشد. پرسپترون ها قادر به ارایه هر تابع خطی جداسدنی هستند، ولی متأسفانه، تعدادی از توابع مثل XOR به دلیل این که به صورت خطی جدا شدنی نمی باشند قابل نمایش تو سط پرسپترون ها نمی باشند. در روزهای ابتدایی هوش مصنوعی، پرسپترون ها ساده بودند، به علت این واقعیت که وزن آن ها می توانست به سادگی پیدا شود.

آموزش پرسپترون

اگر سیگنال خروجی، خیلی بزرگ باشد، اوزان باید کاهش یابند. برای این کار، وزن هایی را که در خروجی شرکت دارند را کاهش دهید، توجه کنید که وزن های با ورودی صفر، مؤثر نمی باشند. اگر خروجی خیلی کوچک باشد، اوزان باید افزایش یابند. برای این کار، وزن هایی را که در خروجی شرکت دارند را افزایش دهید، در این مورد هم توجه کنید که وزن های با ورودی صفر، مؤثر نمی باشند. آموزش، در واقع بهینه سازی وزن ها می باشد و به طور متناوب ما در حال انجام یک جستجوی تپه نورده در میان فضای وزن هستیم.

پرسپترون های چند لایه ای

در پرسپترون های چند لایه ای، لایه ها معمولاً به صورت کامل متصل می شوند و به طور معمول، تعدادی از واحدهای مخفی به صورت دستی انتخاب می شوند. پرسپترون ها دارای این مزیت هستند که یک الگوریتم آموزشی ساده دارند ولی عیب آن ها در این است که دارای محدودیت های محاسباتی هستند. حال سؤال این است که اگر ما یک لایه مخفی دیگر اضافه نماییم چه اتفاقی می افتد؟ پاسخ این است که توان محاسباتی افزایش می یابد؛ با یک لایه مخفی پنهان، می تواند هر تابع پیوسته را نمایش دهد و با دو لایه مخفی، می تواند هر تابعی را نمایش دهد. در این مورد، مشکل این است که چگونه وزن های صحیح را برای گره های پنهان، پیدا نماییم؟

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

واحدهای ورودی

a_i

واحدهای مخفی

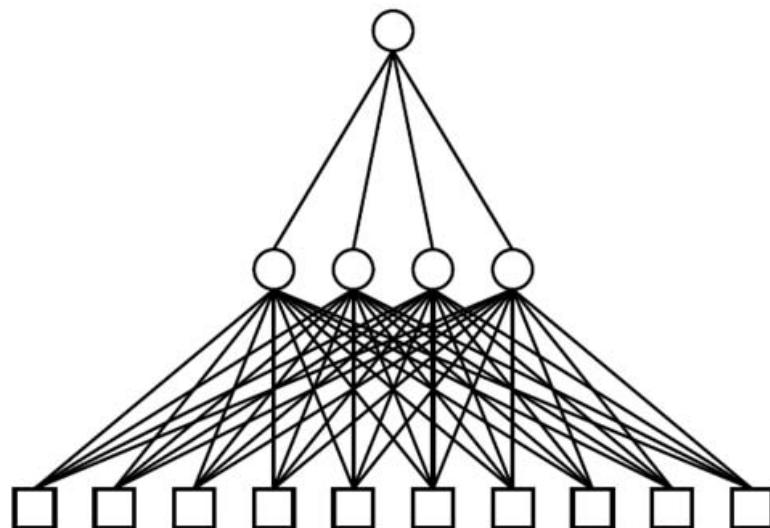
$W_{j,i}$

a_j

واحدهای ورودی

$W_{k,j}$

a_k



انتشار معکوس^۱

انتشار معکوس، شاخه ای از الگوریتم آموزشی پرسپترون برای رسیدگی کردن به چند لایه ی گره هاست. گره ها از تابع فعال سازی سیگموید استفاده می نمایند، داریم:

$$g(\text{input}_i) = \frac{1}{1 + e^{-\text{input}_i}}$$

$$g'(\text{input}_i) = g(\text{input}_i)(1 - g(\text{input}_i))$$

یادداشت:

$h_w(x)$ - بردار خروجی های شبکه ○

y - خروجی مطلوب برای یک مثال آموزشی ○

backpropagation^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

a_j - خروجی j امین واحد مخفی

o_i - خروجی i امین واحد خروجی

t_i - خروجی برای i امین مثال آموزشی

خطای خروجی برای گره i خروجی i :

$$\Delta_i = (t_i - o_i) * g(input_i) * (1 - g(input_i))$$

به روز رسانی وزن (خروجی پنهان):

$$W_{j,i} = W_{j,i} + \alpha * a_j * \Delta_i$$

به روز رسانی وزن های با ورودی پنهان:

ایده: هر گرهی پنهان دارای یک کسر از خطا به صورت Δ_i می باشد. Δ_i را با توجه به شدّت اتصال میان گره پنهان و خروجی تقسیم نمایید: $Delta_j = g(input)(1 - g(input)) \sum_i W_{j,i} \Delta_i$ ، قانون را برای وزن های با ورودی پنهان به روز نمایید: $W_{k,j} = W_{k,j} + \alpha * input_k * \Delta_j$

الگوریتم انتشار معکوس

مادامی که انجام نشده است کارهای زیر را انجام بده:

برای d در مجموعه i آموزشی

ورودی های d و انتشار مستقیم را به کار ببر.

برای گرهی i در لایه ورودی

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$$\Delta_i = output * (1 - output) * (t_{\text{exp}} - output)$$

برای هر گرهی پنهان

$$\Delta_j = output * (1 - output) * \sum W_{k,i} \Delta_i$$

برای هر وزن

$$W_{j,i} = W_{j,i} + \alpha * \Delta_i * input_j$$

متوقف کردن شرط ها

سؤال : چه موقع، آموزش را متوقف نماییم؟

جواب : وقتی که تعداد تکرارها ثابت شده باشد، مجموع خطای زیر یک مجموعه‌ی آستانه باشد و همگرایی^۱ وجود داشته باشد؛ یعنی، هیچ تغییری در وزن‌ها به وجود نیاید.

صورت شناسی با استفاده از شبکه‌های عصبی

یکی از مزایای شبکه‌های عصبی این است که آن‌ها به یک توضیح صریح از ارتباط‌های میان بعدهای نیاز ندارند. آن‌ها واقعاً توضیحات را یاد نمی‌گیرند، برای مسائلی که یک جواب نمادین ندارند به خوبی مناسب می‌باشند. شما می‌توانید یک شبکه‌ی عصبی موجود را برای تشخیص صورت‌ها بازنگری نمایید.

نکاتی در مورد انتشار معکوس

convergence^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

همیشه برای چند لایه‌ی پنهان کار می‌کنند. انتشار معکوس فقط برای گرایش دادن (همگرایی) به یک کمینه‌ی محلی ضمانت شده است و ممکن است مجموعه‌ی وزن‌های کامل را پیدا نکنیم. وزن‌های اولیه‌ی پایین می‌توانند در کارکردن شبکه به صورت به مراتب خطی تربه‌ما کمک نمایند، همچنین می‌توانند از شروع مجدد تصادفی استفاده نمایند.

شبکه‌های با عملکرد شعاعی^۱

یک مسئله‌ی انتشار معکوس این است که هر گره با خروجی یک راه حل همکاری می‌نماید؛ این به این معنی است که همه‌ی وزن‌ها باید برای کمینه کردن خطای عمومی میزان شوند. اغتشاش در یک بخش از داده‌ها می‌تواند سبب یک ضربه در همه‌ی خروجی شبکه شود. همچنین رشته‌های زمانی طولانی می‌باشند. شبکه‌های با عملکرد شعاعی، یک راه حل را برای این مورد بیان می‌نمایند.

بینش: هر گره در شبکه یک جزء از فضای ورودی را نمایش می‌دهد و دنبال طبقه‌بندی نمونه‌هایی که در اطراف آن رخ می‌دهند می‌باشد.

نگرش وانیلی^۲: برای هر نقطه‌ی آموزشی $x_i, f(x_i)$ ، گره‌ای را بسازید که در وسط x_i قرار دارد. خروجی این گره برای یک ورودی جدید x ، $\phi(|x - x_i|) W^*$ خواهد بود. که W ، وزن می‌باشد و ϕ که تابع اساسی است برابر است با $\phi = \exp\left(-\frac{x^2}{2\sigma^2}\right)$.

هر گره دارای یک "ناحیه‌ی تأثیر"^۳ است که می‌تواند نمونه‌های نزدیک را طبقه‌بندی نماید. آموزش با طبقه‌بندی غلط، فقط بر روی گره‌هایی که نزدیک نمونه‌هایی که به صورت نادرست طبقه‌بندی شده‌اند تأثیر خواهد گذاشت. همچنین، این شبکه، تک لایه‌ای می‌باشد؛ وزن‌ها می‌توانند با یک معادله‌ی

Radial Basis Function networks^۱

vanilla approach^۲

zone of influence^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

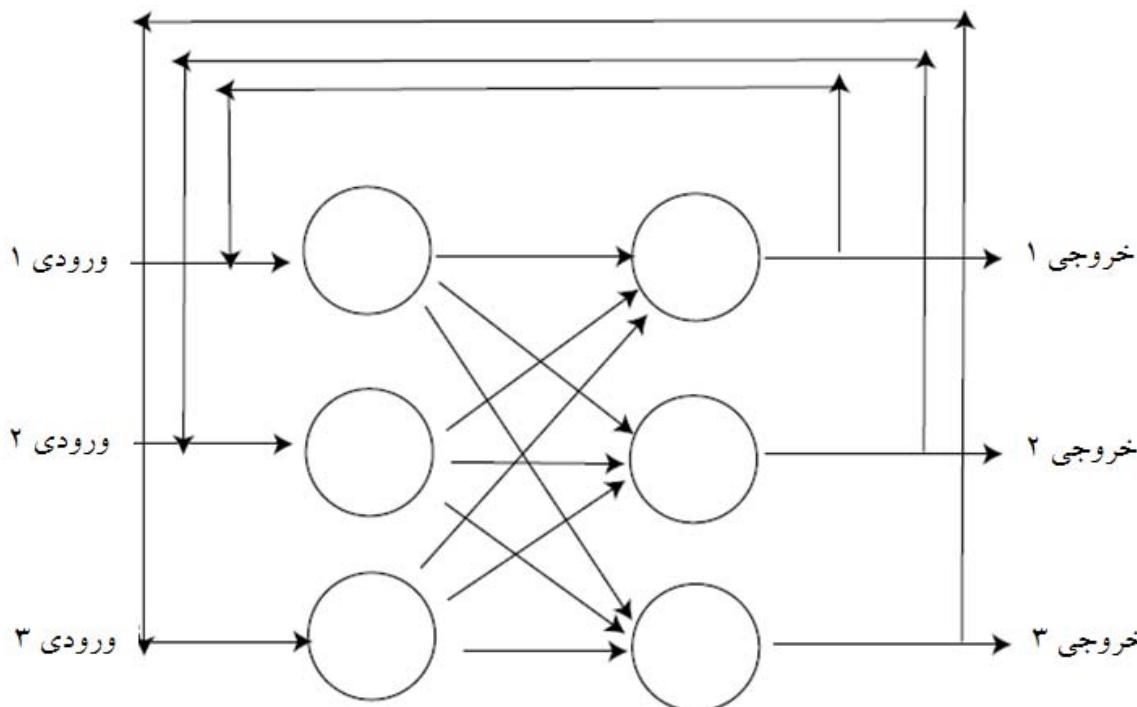


هوش مصنوعی

ماتریسی به دست آیند، داریم: $\Phi W = t$. در نتیجه، $W = \Phi_t^{-1}$. توجه کنید که معکوس کردن یک ماتریس، به مراتب سریع تر از ساختن آن با انتشار معکوس می باشد.

شبکه های عصبی برگشت کننده^۱

تا کنون ما فقط در مورد شبکه های با تغذیه‌ی مستقیم صحبت کرده ایم. در این شبکه ها، علامت ها (سیگنال ها) در یک جهت منتشر می شوند، خروجی بلا فاصله در دسترس می باشد و دارای الگوریتم هایی هستند که به سادگی قابل فهم می باشند. تعداد زیادی کار می تواند با شبکه های عصبی برگشت کننده انجام شود. لاقل برخی از خروجی ها به پشت ورودی ها متصل شده اند. در زیر یک شبکه عصبی برگشت کننده‌ی تک لایه‌ای را می بینید:



recurrent NNs(Neural Networks)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

شبکه های هوپفیلد^۱

یک شبکه ای هوپفیلد دارای هیچ گره معین ورودی یا خروجی نمی باشد؛ هر گره یک ورودی و روال های یک خروجی را دریافت می نماید؛ هر گره به گره های دیگر متصل شده است و معمولاً، از توابع آستانه ای استفاده می شود؛ شبکه بلا فاصله یک خروجی را تولید نمی نماید و مردّم می باشد؛ تحت برخی از وضعیت های به آسانی قابل دسترس، سرانجام به حالت موازنہ می رسد؛ وزن ها با استفاده از روش گرم و سرد کردن شبیه سازی شده به دست می آیند. شبکه های هوپفیلد می توانند برای ساختن یک حافظه ای شرکت پذیر^۲ به کار روند، در این شبکه ها، یک قسمت از یک الگو برای شبکه ارایه می شود و شبکه تمام الگو را به یاد می آورد. برای حرف شناسی^۳ هم به کار می رود، همچنین برای مسایل بهینه سازی هم به کار می رود و اغلب برای مدل فعالیت مغز استفاده می شود.

Hopfield networks^۱
associative memory^۲
letter recognition^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فصل نوزدهم



الگوریتم های ژنتیکی^۱

Genetic Algorithms^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تاریخچه

در دهه‌ی هفتاد میلادی توسط جان هنلند^۱ بیان شد؛ در دهه‌ی هشتاد میلادی عمومیت یافت و براساس تئوری تکاملی داروین (باقی ماندن شایسته ترین) بود. الگوریتم‌های ژنتیکی می‌توانند برای حل انواعی از مسایل که برای حل، آسان نمی‌باشند با استفاده از روش‌های دیگر مورد استفاده قرار گیرند.

تکامل در دنیای واقعی

هر سلول یک موجود زنده دارای کروموزوم^۲ هایی می‌باشد. هر کروموزوم شامل یک مجموعه از ژن^۳‌ها می‌باشد. هر ژن برخی از خواص موجود زنده (مانند رنگ چشم) را تعیین می‌کند. یک مجموعه از ژن‌ها در برخی از موارد ژنوتیپ (نوع معرف و نماینده یک جنس)^۴ نامیده می‌شود. یک مجموعه از

John Holland^۱
chromosome^۲
gene^۳
genotype^۴

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

خصوصیات (نظیر رنگ چشم) گاهی فتوتیپ^۱ نامیده می شود. انتشار، شامل بازتر کیب^۲ زن ها از والدین و سپس مقدار کوچکی جهش^۳ (خطاهای) در کپی کردن می باشد. شایستگی^۴ یک موجود زنده مقداری است که می تواند قبل از مردنش انتشار دهد. تکامل، براساس "بقای مناسب ترین"^۵ می باشد.

شروع با یک تصوّر

فرض کنید که شما مشکلی دارید و نمی دانید که چگونه آن را حل نمایید. برای حل این مشکل چه کار می توانید بکنید؟ آیا می توانید به طریقی از یک کامپیوتر برای پیدا کردن یک راه حل استفاده نمایید؟ این کار باید خوب باشد! می توانید آن را انجام دهید؟

یک راه حل گنگ

یک الگوریتم "تولید و تست گنگ":

تکرار کن

یک راه حل ممکن تصادفی را تولید کن

راه حل را امتحان کن و خوب بودن آن را بسنج

تا موقعی که راه حل به اندازه‌ی کافی خوب باشد

phenotype^۱

recombination^۲

mutation^۳

fitness^۴

survival of the fittest^۵

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

آیا ما می توانیم از این ایده‌ی گنگ استفاده نماییم؟

برخی اوقات بله: در صورتی که تعداد راه حل‌های کمی وجود داشته باشد و شما به اندازه‌ی کافی زمان در اختیار داشته باشید، آن گاه روش‌های این فرمی می‌توانند استفاده شوند. اما برای بیشتر مساله‌ها مانع از این راه حل‌ها استفاده نماییم: در زمانی که راه حل‌های ممکن، زیاد باشند و شما به اندازه‌ی کافی زمان برای امتحان همه‌ی آن‌ها نداشته باشید، این راه حل‌ها نمی‌توانند استفاده شوند.

ایده‌ای که کمتر دارای گنگ بودن می‌باشد (الگوریتم ژنتیکی)

یک مجموعه‌ی تصادفی از راه حل‌ها را تولید نمایید

تکرار کن

هر راه حل را در مجموعه امتحان کن و آن‌ها را رتبه بندی کن

برخی از راه حل‌های بد را از مجموعه بردار

برخی از راه حل‌های خوب را تکثیر کن

برخی از تغییرات کوچک را در مورد آن‌ها اعمال کن

تا زمانی که بهترین راه حل به اندازه‌ی کافی خوب شود

چگونه شما یک راه حل را کد می‌کنید؟

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بدیهی است که این وابسته به مسئله می باشد! الگوریتم های ژنتیکی ، اغلب راه حل ها را به صورت رشته های بیتی^۱ باطول ثابت ، کد می کنند (به عنوان مثال ، ۰۱۱۱۰ ، ۱۰۱۱۰ ، ۱۱۱۱۱ ، ۰۰۰۱۰۱). هر بیت برخی از خصوصیات راه حل های ارایه شده برای مسئله را ارایه می کند . برای این که الگوریتم های ژنتیکی کار کنند ، ما نیاز به این داریم که هر رشته را تست نماییم و به آن امتیازی بدهیم که نشان دهنده ی چگونگی خوب بودن آن باشد .

مثال نادان - حفر برای روغن

تصوّر کنید که شما باید برای روغن جایی را در طول یک جاده‌ی بیابانی یک کیلومتری ، حفر نمایید .

مسئله : بهترین مکان در جاده را که بیش ترین مقدار روغن را در روز تولید می کند ، انتخاب نمایید

ما می توانیم هر راه حل را به صورت یک وضعیت در جاده نمایش دهیم . تصوّر نمایید که تمام اعداد بین [۰ ، ۱۰۰۰] باشند .

کجا را برای روغن ، حفر نماییم ؟

bitstrings^۱

مترجم: سهرا ب جلوه گر



هوش مصنوعی

راه حل ۱ = ۳۰۰



راه حل ۲ = ۹۰۰



جاده (راه)

مجموعه‌ی ممه‌ی راه حل های ممکن [۱۰۰۰]، هصای جستجو یا هصای حات کام دارد. در این مورد، این فقط یک عدد است اما می‌تواند تعداد زیادی عدد یا سمتی باشد. اغلب، الگوریتم‌های ژنتیکی اعداد را به صورت دودویی (با اینتری) کد می‌کنند. در مورد این مثال، ده بیت را که برای نمایش ۰...۱۰۰۰ کافی است را انتخاب می‌نماییم.

تیددا، له دشته هی، یانه هی

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

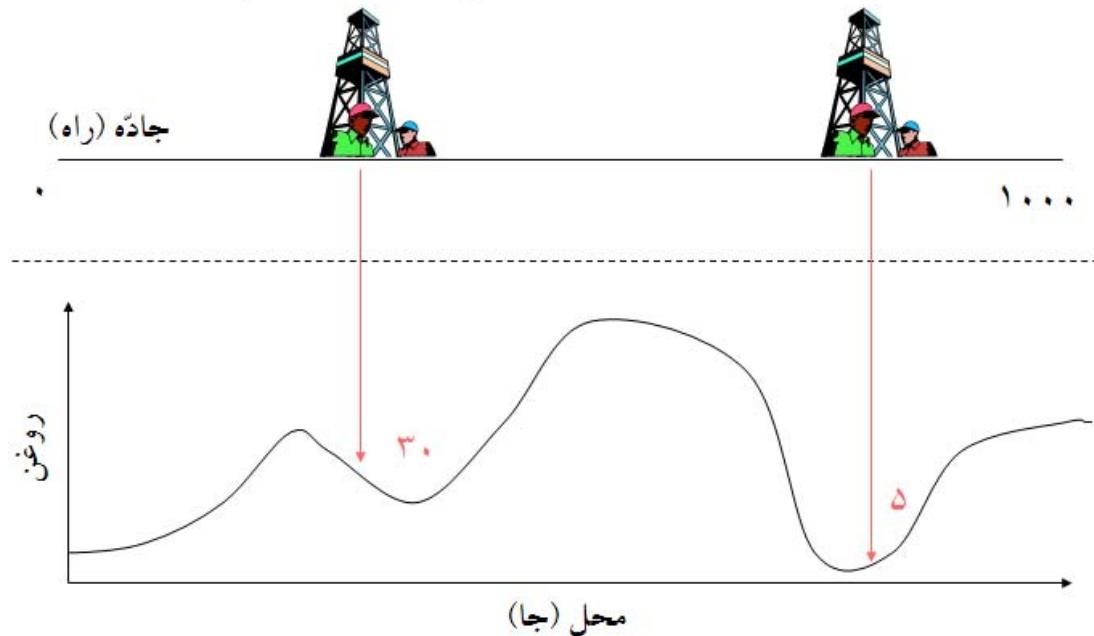
در الگوریتم های زنگنه، این رشته های کد شده گاهی ژنتیک ها یا کروموزوم ها نامیده می شوند و بیت های تکی گاهی اوقات ژن ها نامیده می شوند.

$$\text{راه حل } 1 = 300$$

(0100101100)

$$\text{راه حل } 2 = 900$$

(1110000100)



خلاصه

ما تا کنون دیدیم که چگونه راه حل ها را به صورت یک عدد بیان نماییم. یک عدد را به صورت یک رشته زنگنه کد کردیم. یک رتبه یا درجه را برای هر عدد داده شده به منظور تعیین میزان خوب بودن آن راه به آن عدد نسبت دادیم که اغلب تابع شایستگی^۱ نامیده می شود. مثال روغن در واقع بهینه سازی با استفاده از یک تابع $f(x)$ است که ما باید پارامتر X را تطبیق دهیم.

fitness function^۱

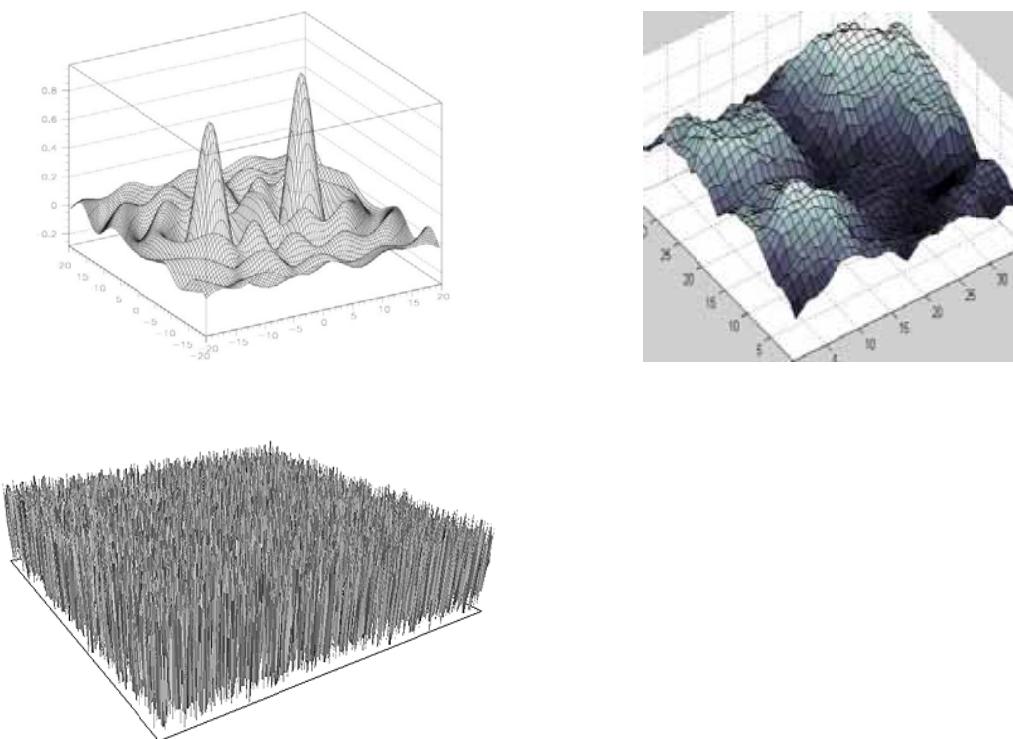
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فضای جستجو

برای یک تابع ساده‌ی $f(x)$ ، فضای جستجو یک بعدی است. اما با کد کردن چند مقدار به کروموزوم، ابعاد زیادی می‌تواند به وجود آید به عنوان مثال برای حالت دو بعدی تابع f به صورت $f(x,y)$ خواهد بود. فضای جستجو می‌تواند به صورت یک سطح باشد. هر زنوتایپ ممکن، یک نقطه در فضا است. یک الگوریتم ژنتیکی تلاش می‌کند که نقطه‌ها را به جاهای بهتر (با شایستگی بالاتر) در فضا منتقل نماید. در زیر سه نمونه از منظره‌های شایستگی را مشاهده می‌نماییم.



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بدیهی است که نوع فضای جستجو تعیین می کند که چگونه یک الگوریتم ژنتیکی کار کند. یک فضای کاملاً تصادفی برای یک الگوریتم ژنتیکی، بد خواهد بود. همچنین الگوریتم های ژنتیکی، اگر فضاهای جستجو شامل تعداد زیادی از موارد تصادفی باشند، ممکن است در ماکریزم محلی بیفتد.

اضافه کردن جنسیت - Crossover

اگرچه که الگوریتمی که ما گفتیم برای فضاهای جستجوی ساده کار می کند اما این الگوریتم هنوز خیلی ساده است. این الگوریتم به جهش های تصادفی برای یافتن یک راه حل خوب، وابسته می باشد. با اضافه نمودن جنسیت به این الگوریتم می توانیم نتایج بهتری را به دست آوریم. این کار با انتخاب دو والد در موقع تکثیر^۱ و ترکیب ژن های آن ها برای تولید فرزند انجام می شود. دو رشته ای بیتی (کروموزوم) والد با امتیاز بالا انتخاب می شوند و با نرخ Crossover با هم ترکیب می شوند. دو فرزند (رشته ای بیتی) به وجود می آیند و هر فرزند ممکن است به صورت تصادفی تغییر داده شود (جهش).

انتخاب والدها

روش های زیادی برای انتخاب کروموزوم های با امتیاز بهتر وجود دارد. امتیاز، اغلب شایستگی^۲ نامیده می شود. از روش "چرخش رولت"^۳ می توان به صورت زیر استفاده نمود:

- شایستگی همه ای کروموزوم ها را اضافه نمایید.
- یک عدد تصادفی R را در محدوده ای آن به وجود آورید.
- اوّلین کروموزوم موجود در جمعیّت را با این شرط که زمانی که همه ای شایستگی های قبلی اضافه می شوند، لااقل مقدار R را می دهد انتخاب نمایید.

¹ reproduction

² fitness

³ Roulette Wheel

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

جمعیّت نمونه

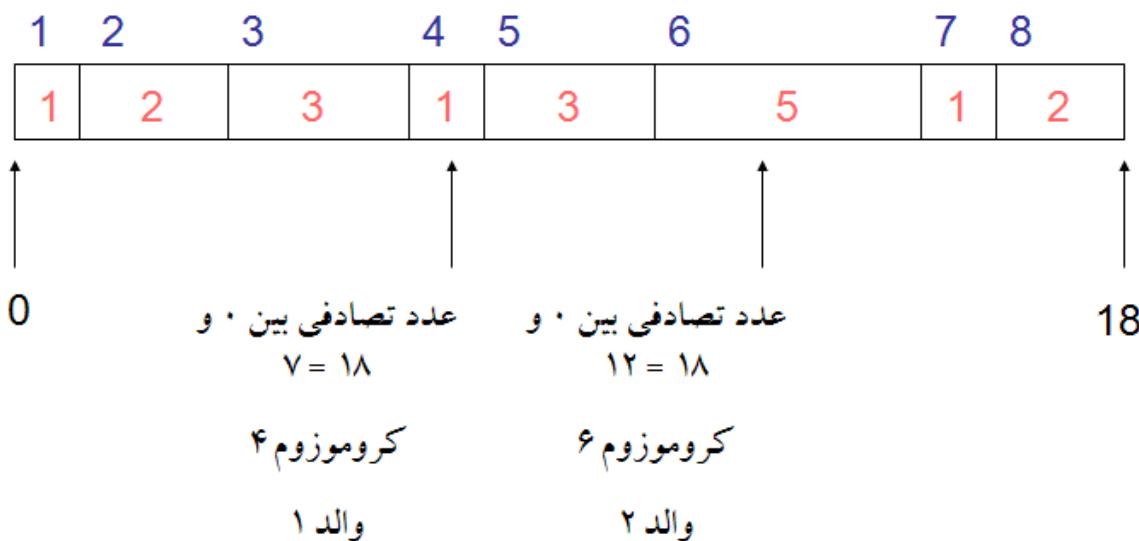
شماره	کروموزوم	شاپستگی
1	1010011010	1
2	1111100001	2
3	1011001100	3
4	1010000000	1
5	0000010000	3
6	1001011111	5
7	0101010101	1
8	1011100111	2

انتخاب چرخش رولت

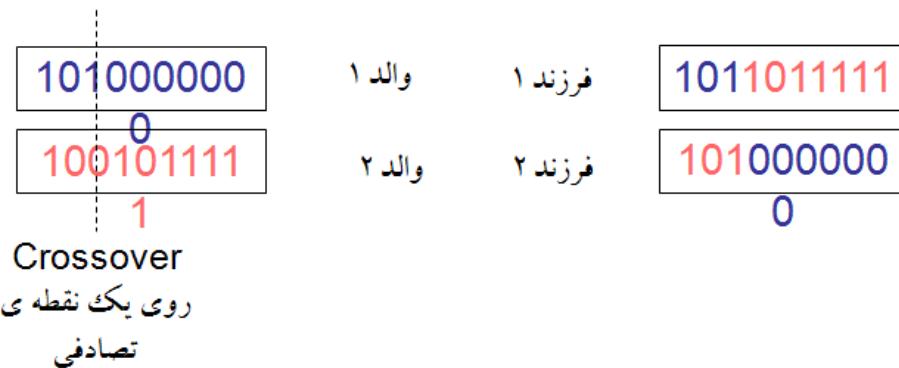
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



– بازترکیب (جفت‌گیری) ^۱ Crossover



با احتمال بالا (نرخ Crossover ، (Crossover را به والدها اعمال نمایید . (مقدارهای عادی بین ۰ . ۹۵ تا ۰ . ۰ می باشند)

جهش

Recombination ^۱: ترکیب جدیدی از فنوتیپها در نتیج حاصل از والدین با فنوتیپی متفاوت .

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

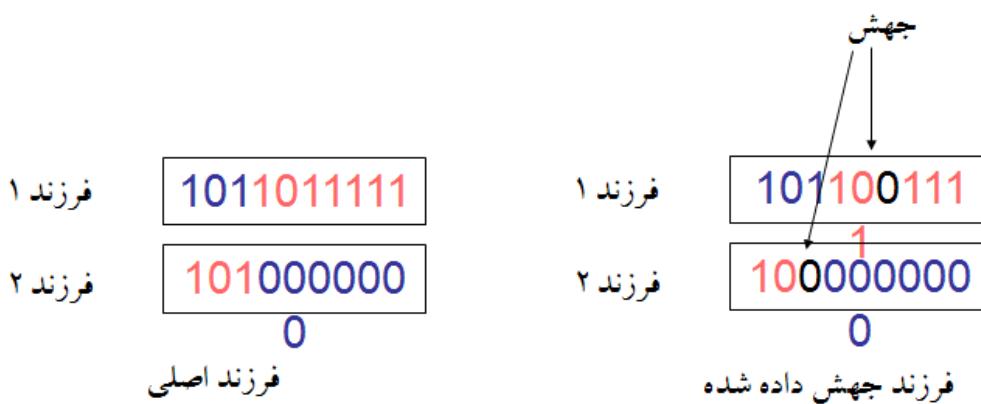


هوش مصنوعی

یک یا بیش تر زن به صورت تصادفی به یک مقدار تصادفی، جهش داده می شوند، مثلاً:

$$1111111111 \rightarrow 1101011111$$

در مورد مثال قبلی داریم:



برگشت به الگوریتم ژنتیکی

یک جمعیت را با استفاده از کروموزوم های تصادفی، تولید کن

برای هر نسل کارهای زیر را انجام بده

شاپستگی هر کروموزوم را محاسبه کن

کارهای زیر را تکرار کن

انتخاب رولت را برای انتخاب جفت های والد ها به کار ببر

فرزنده را با استفاده از Crossover و جهش تولید نما

تا زمانی که جمعیت جدید تولید شود

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

تا زمانی که بهترین راه حل به اندازه‌ی کافی خوب شود

تنوع های زیاد الگوریتم های ژنتیکی

به خاطر موارد زیر ، الگوریتم های ژنتیکی دارای تنوع زیادی هستند :

- انواع مختلف انتخاب که رولت نیستند ؛ مثل ، مسابقه^۱ ، نخبه سالاری^۲ و غیره .
- تفاوت در جفت گیری ؛ مثل ، Crossover چند نقطه‌ای .
- انواع مختلف کد کردن رشته‌ی بیتی ؛ مثل ، مقدارهای صحیح و مجموعه‌ی منظم از سمبل‌ها
- انواع مختلف جهش

پارامترهای زیاد برای تنظیم

هر پیاده سازی الگوریتم ژنتیکی به تصمیم گیری در مورد تعدادی از پارامترها نیاز دارد ؛ مثل ، اندازه‌ی جمعیت (N) ، نرخ جهش (m) ، نرخ Crossover (c) . که اغلب ، این‌ها باید براساس نتیجه‌های به دست آمده تنظیم شوند . مقدارهای معمولی شاید $N = 50$ ، $m = 0.05$ و $c = 0.9$ باشند .

چگونه Crossover کار می‌کند ؟

تعداد زیادی نظریه در این مورد وجود دارد البته برخی مخالفت‌ها هم با این نظریه‌ها وجود دارد . آقای هُلَنْد نظریه‌ی " الگو " را معرفی کرد ؛ این ایده می‌گوید که Crossover ، بیت‌های خوب والدین مختلف را نگهداری می‌کند و آن‌ها را برای تولید راه حل‌های بهتر ترکیب می‌نماید . بنابراین ، یک الگوی خوب کد کننده باید بیت‌های خوب را در طول Crossover و جهش نگهداری نماید .

Tournament^۱

Elitism^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



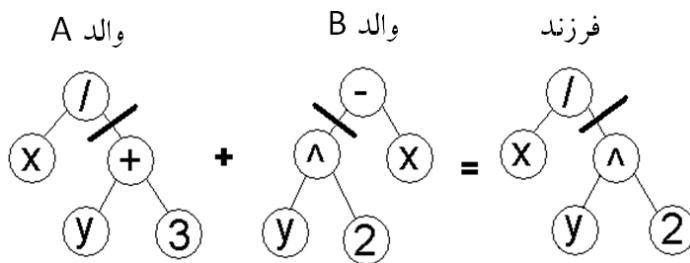
هوش مصنوعی

کاربردهای الگوریتم های ژنتیکی

الگوریتم های ژنتیکی در بسیاری از موارد و زمینه های تحقیقاتی، نظیر مسایل عددی، مسئله مسافت شخص دوره گرد، مدار همیلتونی، مسیر اویلری، پیدا کردن شکل مولکول های پروتئین، مسایل NP سخت، طراحی شبکه های عصبی، توابع برای به وجود آوردن تصاویر، مدلسازی شناختی و تئوری بازی ها به کار می روند.

برنامه نویسی ژنتیکی

زمانی که کروموزوم یک برنامه یا تابع را کد می کند، این کار برنامه نویسی ژنتیکی نامیده می شود. برای انجام این کار معمولاً از ارایه‌ی درختی استفاده می شود. Crossover، باعث جابه جا شدن زیر درخت ها میان والدین می شود.



این کار برای برنامه های کوچک، مفید است ولی برای برنامه های بزرگ با توابع پیچیده، ناکارآمد می باشد.

توابع شایستگی ناآشکار (ضمونی)

اغلب الگوریتم های ژنتیکی از توابع شایستگی ضمنی استفاده می کنند (همانند آنچه که در مورد مثال روغن دیدید). برخی از الگوریتم های ژنتیکی (نظیر زندگی مصنوعی یا روبوتیک تکاملی) از توابع شایستگی ضمنی و پویا استفاده می کنند.

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مسئله

در مسئله‌ی مسافت شخص دوره گرد، یک دوره گرد باید کوتاه ترین مسیر را که یک مجموعه از شهرها را طی می‌کند، پیدا نماید. فرض کنید که ما فاصله‌ی میان هر شهر را می‌دانیم. این مسئله‌ای مشکل می‌باشد، زیرا تعداد مسیرهای ممکن برابر $N!$ می‌باشد، که N ، برابر با تعداد شهرها می‌باشد. الگوریتم ساده‌ای که در این مورد بهترین جواب را به سرعت ارایه نماید وجود ندارد. یک کد کننده‌ی کروموزوم، یک عملکرد جهش و یک تابع Crossover را برای مسئله‌ی مسافت شخص دوره گرد پیدا می‌نماید. فرض کنید تعداد شهرها (N) برابر ۱۰ می‌باشد. بعد از همه‌ی عملیات، کروموزوم‌های تولید شده باید همیشه مسافت‌های ممکن مجاز را ارایه نمایند (هر شهر فقط باید یک بار ملاقات شود). برای این مسئله، یک راه حل وجود ندارد و تعداد زیادی الگوهای مختلف، قبلًا استفاده شده‌اند.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل بیستم

سیستم های خبره^۱

Expert systems^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سیستم های خبره، برنامه هایی طراحی 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 مرکز می کنند. در این سیستم ها، برنامه نویسان، واقعیات و قوانین را اضافه می کنند و سیستم، پردازهای استنتاج را به صورت خودکار انجام می دهد و می توانند یا از زنجیرهای مستقیم (سیستم های تولید) یا از زنجیرهای معکوس (ثابت کننده های قضیه) 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 اطلاعات کلی ترا را به نمایش نمی گذارد و در یک محدوده ای نسبی و با تخصص نسبی کار می کند. اما در آن زمینه عملکردش در سطح خبره می باشد. این سیستم ها ممکن است برای طبقه بندی، تشخیص عیب، پیدا کردن خرابی، تفسیر اطلاعات، طراحی، 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 ای تجاری سیستم های خبره شامل این موارد می باشد: یک سیستم استفاده شده برای ارایه ای

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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 عالیم 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)^۱ ؛ 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 خطای کنند و مناسب تر هستند و رک تر هستند ، یا در مقایسه با خبره های بشری ، بی طرف تر می باشند . البته ، سیستم های خبره معايب زیادی هم دارند ، که باید استفاده از آن ها را به دامنه های مشخص ، محدود نماییم . سیستم های خبره دارای بینش ، دلسوزی ، فهم انگیزه ی بشری ، توانایی حدس ، توانایی یادگیری (معمولًا)، دانش قضاوت صحیح (عقل سليم) کمی می باشند . در این مورد ، کاربر می تواند دلسوزی و قضاوت صحیح را اضافه نماید .

خصوصیات 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

^۱ 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 دوم، بهار ۱۳۸۸



هوش مصنوعی

باید طبیعی باشد . سوالات 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 باورگردنی و نه الزاماً صحیح باشند ؛ به عنوان مثال ، یک سیستم خبره که یک وضعیت بهداشتی را از یک مجموعه از علائم ، استنتاج می کند ؛ بعید است که وضعیت ، رشته ای منطقی از علائم (نشانه ها) باشد . باید قادر به کار با دامنه ای دانش ناقص و مواردی که اطلاعات ناقص هستند ، باشد . اغلب ، دانش و / یا مورد اطلاعات ، ناکامل می باشند و دانش و / یا داده ها ممکن است قابل اعتماد نباشند (مثلاً ، ممکن است دارای خطای 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 اطلاعات جدید را یکی نمایند (یا از طریق مهندسی دانش بیشتر 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 . اما ، اکثر سیستم های خبره که تا کنون ساخته شده اند سیستم های قانون گرا می باشند ؛ یعنی ، پایگاه دانش ، شامل واقعیت ها و قانون ها می باشد .

پایگاه دانش

maintainability^۱
rule-based^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پایگاه دانش ، شامل واقعیت ها و قوانین می باشد . تصوّر کنید که ما در حال ساخت یک سیستم خبره‌ی قانون گرا برای تشخیص پزشکی بودیم . قانون‌ها ارتباط‌های میان علایم و عبارت‌های پزشکی را کد خواهند کرد . واقعیت‌ها دانش در مورد بیماری جاری را کد خواهند کرد . در سیستم‌های قانون گرا ، اغلب به صورت اگر ... آن گاه ... نوشته می شوند ، اما به صورت برابر می توانند به یکی از شکل‌هایی که برای موارد زیر استفاده می شوند نوشته شوند :

if p_1 AND p_2 ... AND p_n then q

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

$q \vee \neg p_1 \vee \dots \vee \neg p_n$

برای سیستم‌های قانون گرا ، استفاده از منطق گزاره‌ای ، زمانی که گزاره‌ها دارای هیچ آرگومانی نمی باشند ، کاملاً معمولی است . بنابراین متغیرها و سمبول‌های تابعی در واقعیت‌ها و قانون‌ها وجود ندارند . ما در مثال‌های زیر خودمان را با این وضعیت ، محدود نموده ایم . نتیجه‌ی یک قانون شاید یک قلم تجزیه ناپذیر باشد که در قانون قبلی آمده است . ما می توانیم این زنجیره را به شکل گرافیکی به صورت یک گراف AND-OR نمایش دهیم . برای مثال ، برای قانون‌های 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

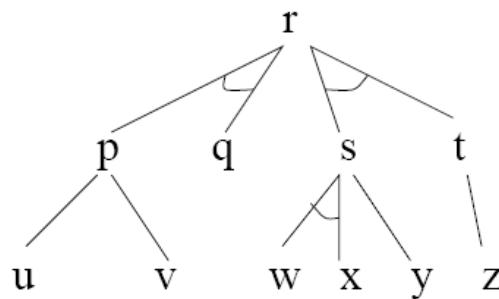
if z then t

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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 *animalChews²Cud³* **then** *animalIsUngulate⁴*

if *animalIsUngulate* AND *animalHasLongNeck⁵* **then** *animalIsGiraffe⁶*

if *animalIsUngulate* AND *animalIsStriped⁷* **then** *animalIsZebra⁸*

ما از این مثال، بعداً استفاده خواهیم کرد.

¹ پستاندار

² می جَوَد

³ نُشخوار می کند

⁴ سُم دار

⁵ گَرْدَن

⁶ زرَافَه

⁷ راه راه یا خط دار

⁸ گُورِ خَر

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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 مؤثر، ما به دنبال یک مسیر که ریشه و برگ ها را به هم متصل می کند و این موارد را اجرا می کنند می گردیم: در صورتی که گره ای، یک گرهی OR می باشد، برای نمایش این که یک مسیر برای فقط یکی از نسل ها وجود دارد، مناسب می باشد؛ اگر گره ای، یک گرهی 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 واقعیت را که باید درست باشد را نمایش می دهد.

در سیستم های خبره‌ی محاوره‌ای، ما فرض نمی کنیم که همه‌ی واقعیات شناخته شده اند و قبل از در پایگاه دانش وجود داشته اند. بنابراین، در زمانی که ما در تلاش برای این هستیم که بینیم که آیا یک گرهی برگ (واقعیت) درست است یا نه، ما پایگاه دانش را بررسی خواهیم نمود، اما اگر در پایگاه دانش نباشد، در این صورت ما از کاربر پرسش می کنیم که آیا واقعیت 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).

زنگیره‌ی معکوس - بیشتر استدلال قانون گرا به روشی انجام می شود که شبیه تحلیل SLD می باشد (همان طوری که در پاراگراف قبل گفته شد، دلایل وجود دارد که به ندرت با تحلیل SLD برابر می باشند). اما، عبارت های زنگیره‌ی معکوس، استدلال مشتق شده از هدف می باشند و استدلال مشتق شده از فرض به صورت عمومی تری در جامعه‌ی سیستم های خبره مورد استفاده قرار می گیرند. در عبارت

Rule-Based Reasoning (RBR)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



های گراف AND-OR، این نوع از استدلال در ریشه‌ی گراف شروع می‌شود و برای پیدا کردن مسیری از ریشه به برگ‌ها تلاش می‌نماید.

زنجیره‌ی مستقیم - این روش، استدلال مشتق شده از داده‌ها هم نامیده می‌شود. در عبارات گراف AND-OR، این نوع از استدلال از برگ‌ها شروع می‌کند و تلاش می‌کند که مسیری را از برگ‌ها به طرف ریشه پیدا نماید.

زنجیره‌ی معکوس و مستقیم برگ برگ شده^۱ - برخی از سیستم‌ها منحصرآ از یکی از این روش‌ها یا هر دوی این روش‌ها استفاده می‌کنند. اما تعداد زیادی از سیستم‌های برگ برگی از هر دو روش استفاده می‌کنند که این کار خیلی طبیعی می‌تواند باشد. به یک مشاوره با یک دکتر بشری توجه نماید. بیمار برخی از عالیم را تشریح می‌کند. نتایج با استفاده از زنجیره‌ی مستقیم به دست می‌آید (شاید فقط به طور آزمایشی). دکتر یک فرض را انتخاب می‌کند و با استفاده از زنجیره‌ی معکوس به پرسشی می‌رسد که برای بیمار ارایه شده است. بیمار، دوباره صحبت می‌کند، و شاید به پرسشی دیگر جواب دهد و یا از او اطلاعات دیگری خواسته شود. و این کار (پروسه) باز هم تکرار شود.

توضیح‌های استدلال

ما قبلاً اهمیت داشتن توضیح روان را در یک سیستم خبره بیان کردیم. سیستم‌های قانون گرا معمولاً دو امکان را برای توضیحات ارایه می‌کنند. ممکن است یا پرسند چرا و یا پرسند چگونه. سیستم، سؤالات را با نمایش برخی از قوانین مربوط، جواب می‌دهد. برای مثال، تصوّر نماید که سیستم خبره‌ی شناسایی حیوانات از کاربر این پرسش را می‌پرسد که آیا حیوان، نشخوار می‌کند. در این مورد، کاربر می‌تواند به جای جواب دادن به این سؤال پرسد که: چرا شما این سؤال را می‌پرسید؟ برای این کار، سیستم خبره با نمایش یک گراف AND-OR پاسخ می‌دهد. برای مثال، ممکن است سیستم پاسخ دهد: من از شما پرسیدم که آیا حیوان نشخوار می‌کند، زیرا این در تشخیص این که حیوان، جانور سرم دار است کمک می‌

Interleaved Backwards- and Forwards-Chaining^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



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 زرآفه است کمک می کند. در صورتی که حیوان سم دار باشد و دارای گردن بلند باشد، زرآفه 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 پرسد؛ یعنی این که بپرسد: چگونه به این نتیجه رسیدی؟ برای این کار، سیستم خبره با نمایش قسمت های موفق 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.

بیشتر تحقیقات انقلابی در سیستم های قانون گرا در مورد سیستمی به نام MYCIN انجام شده است. ایده های درون MYCIN دارای برتری بر همه سیستم های خبره ای قانون گرا هستند. MYCIN برای استفاده ی یک پزشک برای تشخیص عفونت های باکتریایی خون طراحی شد. در حدود ۴۵۰ قانون دارد و تقریباً به طور کامل با استفاده از زنجیره ای معکوس کار می کند. بازده MYCIN و طبیعی بودن مکالمه ای آن با مشارکت قطعه های خیلی کوچک دانش که شاید در مدت زنجیره ای معکوس به کار گرفته شوند، بهبود یافت. یکی از پرمغایر ترین این موارد، استفاده از یک مجموعه از فرآقانون 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 ای مشهور، PROSPECTOR (برای ارزیابی مکان های معدنی دارای استعداد برای استخراج) و R1 (a.k.a. XCON) برای بررسی، کامل کردن و پیکربندی تقاضاهای تجهیزات کامپیوترا مشتری، می باشند. بعد از برخی از آزمایش ها با ساخت چند

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سیستم خبره‌ی قانون گرا، محققان هوش مصنوعی دریافتند که فقط پایگاه دانش، وابسته به دامنه می‌باشد. موتور استنتاج و رابط کاربر (نسبتاً) مستقل از دامنه بودند. به طور کلی، برای ساخت یک سیستم خبره‌ی جدید برای یک دامنه‌ی مختلف به یک پرسه‌ی مهندسی دانش برای دریافت کردن قانون‌ها نیاز داریم.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



۱



فصل بیست و یکم

پردازش های تصمیم

گیری مارکوف^۲

^۱ تصویر بالا متعلق به آندری ای. مارکوف (Andrei A. Markov) (۱۸۵۶ – ۱۹۲۲)، دانشمند و ریاضیدان معروف روسی است.

Markov Decision Processes(MDP)^۳

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

پردازش های تصمیم گیری مارکوفی

یک مدل گسسته برای تصمیم گیری تحت نامعلومی می باشد . چهار جزء این مدل عبارتند از : وضعیت ها ؛ که جهان به وضعیت هایی تقسیم شده است . عملکردها ؛ هر وضعیت دارای یک تعداد محدود از عملکردها می باشد . تابع انتقال ؛ ارتباطی احتمالی میان وضعیت ها و عملکردهای قابل دسترسی برای هر وضعیت و تابع پاداش ؛ پاداش مورد انتظار انجام یک کار ، تحت وضعیت s می باشد .^۱

به وجود آوردن تصمیم گیری های ترتیبی

قبلآ ما در مورد این موارد صحبت کردیم : به وجود آوردن تصمیم گیری های به یکباره در یک محیط قطعی ؛ به وجود آوردن تصمیم گیری های ترتیبی در یک محیط قطعی ، مثل : جستجو – استنتاج – برنامه ریزی ؛ به وجود آوردن تصمیم گیری های به یکباره در یک محیط اتفاقی ، مثل : شبکه های باور و

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

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

سودمندی مورد انتظار

به یاد بیاورید که سودمندی مورد انتظار یک عملکرد ، سودمندی هر نتیجه‌ی ممکن است و توسعه احتمالی که نتیجه اتفاق یافتدارزیابی (وزن) می شود ؛ به طور صریح تر ، از وضعیت S ، یک عامل ممکن است عملکردهای a_1, a_2, \dots, a_n را بگیرد . هر عملکرد a_i می تواند منجر به وضعیت های $s_{i1}, s_{i2}, \dots, s_{im}$ با احتمال $P_{i1}, P_{i2}, \dots, P_{im}$ 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 وضعیت های وابسته را مدل انتقال وضعیت^۱ می نامیم . عامل باید وضعیت 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 ، مسئله این است که چگونه احتمال وضعیت ها را تشخیص دهیم ؟ ؛ احتمال رسیدن به وضعیت 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 وضعیت ها فقط وابسته به یک تعداد محدودی از والدها می باشد . ساده ترین ، پردازه‌ی مرتبه‌ی اوّل مارکوف در این صورت است که احتمال تغییر وضعیت ها فقط وابسته به وضعیت قبلی باشد . ما نیز روی پردازش مرتبه‌ی اوّل مارکوف تمرکز می نماییم .

توزیع های ثابت^۲

state transition model^۱
Stationary distributions^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

حل مسایل ترتیبی

در این مورد ، سودمندی ، وابسته به یک رشته از وضعیت های s_1, s_2, \dots, s_n می باشد و برای هر وضعیت ، 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 ای تصمیم گیری مارکوفی داریم : یک وضعیت اویلیه s_0 ؛ یک مجموعه از وضعیت ها و عملکردهای گستته ؛ یک مدل انتقال : $T(s,a,s')$ که احتمال رسیدن به وضعیت s' از s وقتی که عمل a انجام می شود را نشان می دهد و یکتابع پاداش $R(s)$

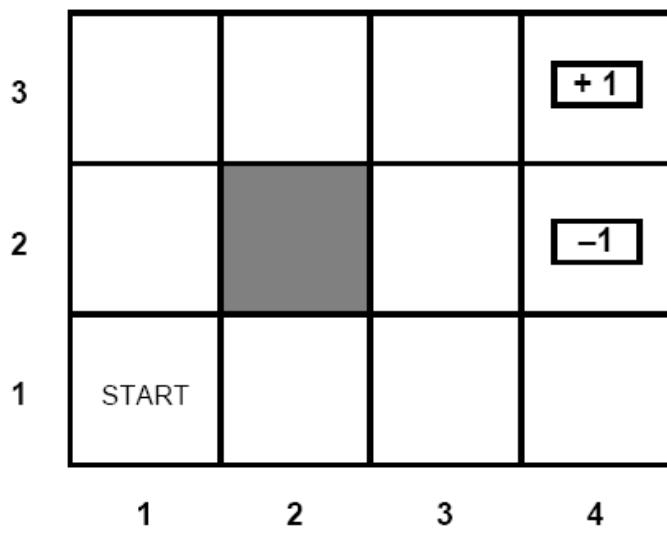
مثال : مسئله ی شبکه

مترجم: سهراب جلوه گر

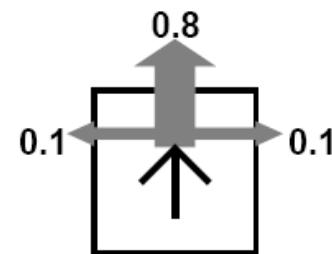
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



(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 وضعیت ، 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 اتفاقی می باشد ، یک راه حل به صورت رشته ای از عملکرد نمی باشد . در عوض ، ما باید معین نماییم که یک عامل باید چه کاری در هر وضعیت ممکن انجام دهد . ما این خصوصیت را یک روش ^۱ می نامیم : "اگر شما زیر هدف می باشید بالا باید و اگر شما در ستون سمت چپ می باشید ، به راست حرکت نمایید . " ما یک روش را با π نشان می دهیم و (S, π) روش را برای وضعیت S ، مشخص 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

π^*
 $\pi^{optimal}$

مترجم: سهراب جلوه گر

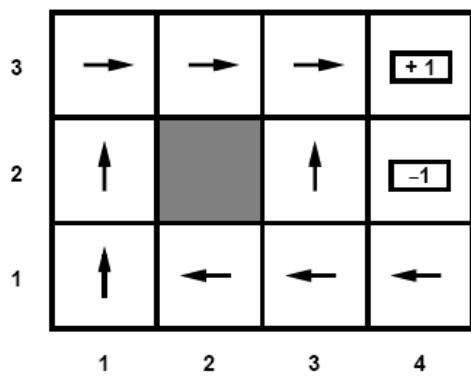
ویرایش دوم، بهار ۱۳۸۸



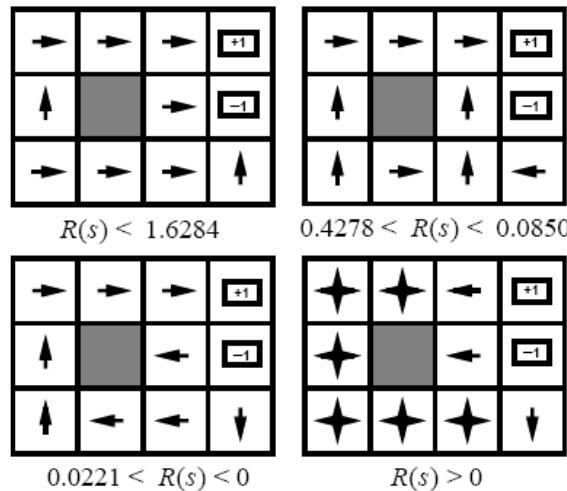
هوش مصنوعی

نام دارد. اولین باری که یک روش پیدا می شود، عامل فقط می تواند بهترین عملکرد را برای هر وضعیت پیدا نماید.

مثال: مسئله ی شبکه



(a)



(b)

توجه کنید که در شکل سمت چپ، $R(s)$ برابر است با 0.400 . و در شکل، تمام پاداش های غیر صفر، باید منفی باشند. از آنجایی که هزینه های تغییر در وضعیت های غیربایانی تغییر می یابد، بنابراین 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 جلوگیری نماید و در پاداش مثبت، عامل تلاش نمی کند که خارج شود.

مطلوب بیش تری در مورد توابع پاداش

در حل یک پردازه ی تصمیم گیری مارکوفی، یک عامل باید به مقدار عملکردهای آینده توجه نماید. انواع مختلفی از مسایل که باید به آن ها توجه نماید، وجود دارد:

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

• وسعت - آیا جهان برای همیشه ادامه می یابد؟

○ وسعت محدود: بعد از N عملیات، جهان متوقف می شود و هیچ پاداشی دریافت نمی شود.

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

• ما همچین 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:

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

در صورتی که γ بزرگ باشد، ما وضعیت 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 $1 - \gamma$. تخفیف به ما اجازه می دهد که مقدار محسوسی، مسایل با وسعت نامحدود را داشته باشیم. در غیر این صورت، همه می 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 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 دوم، بهار ۱۳۸۸



هوش مصنوعی

تکرار مقدار

چگونه یک روش بهینه را پیدا نماییم؟ ما با محاسبه‌ی سودمندی مورد انتظار هر وضعیت و سپس انتخاب عملکردهایی که سودمندی مورد انتظار را بیشینه می‌کنند شروع می‌نماییم. در یک مسئله‌ی ترتیبی، سودمندی یک وضعیت، سودمندی مورد انتظار همه‌ی رشته‌های وضعیتی که از آن دنبال می‌شوند، می‌باشد. این به روشی که اجرا می‌شود وابسته می‌باشد. در اصل، $U(s)$ سودمندی مورد انتظار یک روش بهینه از وضعیت s می‌باشد.

سودمندی وضعیت‌ها

	0.812	0.868	0.918	+1
2	0.762		0.660	-1
1	0.705	0.655	0.611	0.388

1 2 3 4

توجه کنید که سودمندی‌ها دارای بیشترین مقدار برای وضعیت‌های نزدیک به خروجی +1 هستند. سودمندی یک وضعیت، پاداش فوری برای آن وضعیت به اضافه‌ی سود تخفیف داده شده‌ی مورد انتظار وضعیت بعدی می‌باشد و فرض براین است که عامل، عملکرد بهینه را انتخاب می‌نماید.

معادله‌ی بلمن:

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

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

معادله‌ی بلمن، براساس برنامه نویسی پویا می‌باشد. در یک گراف انتقال بدون حلقه، شما می‌توانید این‌ها را به صورت بازگشتی توسط معکوس کار کردن از وضعیت نهایی به وضعیت‌های اویله حل نمایید. شما این کار را برای گراف‌های انتقال با حلقه، به صورت مستقیم نمی‌توانید انجام دهید.

تکرار مقدار

از آنجایی که سودمندی وضعیت‌ها وابسته به سودمندی سایر وضعیت‌ها می‌باشد، چگونه یک راه حل نزدیک را پیدا نماییم؟ ما می‌توانیم از یک نگرش تکراری به صورت زیر استفاده نماییم:

- هر وضعیت را به صورت تصادفی مقداردهی اویله نمایید.

- برای یک وضعیت، طرف سمت چپ را با توجه به مقدار های همسایگانش محاسبه نمایید.

- این کار را برای به روز رسانی طرف سمت راست دیگر وضعیت‌ها ادامه دهید.

- قانون به روز رسانی به صورت زیر است:

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

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

الگوریتم تکرار مقدار

انجام دهید^۱

برای s های موجود در وضعیت ها

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

تا هنگامی که^۲

تمام سودمندی ها با مقدار کم تراز 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	0.15

do^۱
until^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

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:

مترجم: سهراب جلوه گر
ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی

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 :

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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:

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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	-0.01

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

تشریح مطلب

نقاط قوت 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 اطلاعات نیاز نداریم و فقط به این نیاز داریم که بدانیم در هر وضعیت چه کاری را انجام دهیم .

تکرار روش ^۱ (بهبود روش تکرار مقدار)

Policy iteration ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

الگوریتم

سودمندی همه ی وضعیت ها را برابر با صفر قرار بده

رشته ی روش های تصادفی شاخص گذاری شده توسط وضعیت را در P_i قرار بده

انجام دهید^۱

سودمندی هر وضعیت را برای P_i ارزیابی نماید و آن را در U قرار دهید .

برای S موجود در وضعیت ها

عملکردی که سودمندی مورد انتظار را برای آن وضعیت ، بیشینه می نماید پیدا

نماید و در a قرار دهید

را در $P_i(S)$ بروزیزد

مدادامی^۲ که برخی از عملکردها تغییر کردند

مثال :

do^۱

while^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

1	-0.04 	2	-0.04 	3	0.04 	+1	
4	-0.04 			5	-0.68 	-1	
6	-0.04 	7	-0.04 	8	-0.04 	9	-0.12

روش های تصادفی را به وجود آورید ، سودمندی وضعیت ها را براساس این روش ها ، مورد ارزیابی قرار دهید .

1	-0.04 	2	-0.04 	3	0.04 	+1	
4	-0.04 			5	-0.68 	-1	
6	-0.04 	7	-0.04 	8	-0.04 	9	-0.12

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

1 -0.07	2 -0.02	3 0.55	+1
4 -0.07		5 -0.14	-1
6 -0.07	7 -0.07	8 -0.12	9 -0.12

Arrows indicate transitions between states:

- From state 1 to state 2: down arrow
- From state 2 to state 3: right arrow
- From state 3 to state 4: right arrow
- From state 4 to state 5: left arrow
- From state 5 to state 6: up arrow
- From state 6 to state 7: right arrow
- From state 7 to state 8: left arrow
- From state 8 to state 9: left arrow

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

1 -0.04	2 -0.02	3 0.55	+1
4 -0.04		5 -0.14	-1
6 -0.04	7 -0.04	8 -0.12	9 -0.12

Arrows indicate transitions between states:

- From state 1 to state 2: right arrow
- From state 2 to state 3: right arrow
- From state 3 to state 4: right arrow
- From state 4 to state 5: left arrow
- From state 5 to state 6: up arrow
- From state 6 to state 7: right arrow
- From state 7 to state 8: left arrow
- From state 8 to state 9: down arrow

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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 مجدد 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
→	←	↑	↓

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

1 0.15	2 0.41	3 0.67	+1
4 0.04		5 0.30	-1
6 -0.11	7 -0.11	8 0.08	9 -0.13

مجدداً، از روش های به دست آمده برای تخمین مجدد سودمندی ها استفاده نمایید.

1 0.15	2 0.41	3 0.67	+1
4 0.04		5 0.30	-1
6 -0.11	7 -0.11	8 0.08	9 -0.13

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

1 0.23	2 0.45	3 0.68	+1
4 0.06		5 0.33	-1
6 -0.03	7 -0.01	8 0.13	9 -0.07

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. حسن تکرار روش در این است که نزدیک شدن یا همگرایی سریع تر می باشد.

هوش مصنوعی



مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

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 کیم [و کدام را اجرا نمایم]، ما بعداً به این مورد خواهیم پرداخت.

طبقه بندی کننده ها

الگوریتم های ژنتیکی، به صورت بالقوه، به ما یک روش برای استنتاج و انتخاب قانون ها را ارایه می دهند. گرچه قانونی که دارای بیش ترین شایستگی می باشد را انتخاب نمایند، توسعه ۱، و «^۱» شکلاتی که در مورد وجود دارند عبارتند از اینکه: چگونه شایستگی را به یک قانون انتساب دهیم؟ و چگونه عملکرد برخط بالا رانگهداری نماییم؟

اجزای یک سیستم طبقه بندی

سیستم های طبقه بندی یک سیستم جزوی از قانون و سیستم انتساب اعتبار و یک الگوریتم ژنتیکی برای تولید قانون های جدید است.

قانون و سیستم پیام - حسگرهای عارا، اطلاعات را که به صورت یک رشته می بینی کند شاهد اند دریافت می نمایند. از محاسبات این سیستم، این پیام به بندهای کننده ها (قانون ها) را با تطبیق شرط ها فعال می نماید. طبقه بندی کننده ها پیام ارسال می نمایند؛ این پیام ها

Classifier Systems^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سیستم های طبقه بندی کنندۀ

الگوریتم های ژنتیکی برای مسائل بهینه سازی، خیلی خوب کار می کنند و داریم $f(x_1, x_2, x_3, \dots, x_n) = R$ در مورد مسائلی 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 بادقت تر باشیم. طبقه بندی، پردازه ای انتساب یک ورودی به یکی از کلاس های چندگانه می باشد.

سیستم های تولید^۱

یک سیستم تولید، از یک مجموعه از تولیدات (قوانين)، حافظه ای که در آن واقعیّات قرار می گیرند و یک الگوریتم که با استفاده از روش زنجیره ای 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

Production system^۱
Babylon / FOLDOC^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

از اگر - آن گاه یا قانون های شرط - عملکرد استفاده می شود . مثلاً در دنیای جاروبرقی داریم : در صورتی که خانه ی [۱،۱] 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 بالقوه ، 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 $1,0,0,*$ قانون ها را به صورت یک رشته ی بیتی کد نمایید . بیت های وضعیت ، 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 ، اطلاعات را که به صورت یک رشته ی بیتی کد شده اند دریافت می نمایند ؛ این اطلاعات ، پیامی از محیط می باشد . این پیام طبقه بندی کننده ها (قانون ها) را با تطبیق شرط ها فعال 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 فعال نمایند .

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

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

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

اگر پیام ۱۱۱ از محیط دریافت می شود . در ابتدا ، قانون یک اجرا می شود ، ۰۰۰ در لیست پیام قرار می گیرد . سپس ، قانون های دو و چهار اجرا می شود ، ۱۱۰۰ و ۱۱۰۱ در لیست پیام قرار می گیرند . بعد ، قانون سه اجرا می شود ، ۱۰۰۰ در لیست پیام قرار می گیرد ، که با قانون شماره 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 قوت 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 فعال کرده (اند) فرستاده می شود .

مثال : اگر به صورت اوّلیه (t=0) داشته باشیم : $M = 111$

- 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

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

S=0 محیط

:t = 1 در

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

S=20 محیط

:t = 2 در

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

S=20 محیط

:t = 3 در

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

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

S=20 محیط

: t = 4 در

1. 01## : 0000 S=220 0001
2. 00#0 : 1100 S=208
3. 11## : 1000 S=196
4. ##00 : 0001 S=156

S=20 محیط

و در $t = 5$ - درخواست به سیستم می آید و به آخرین طبقه بندی کننده‌ی فعال نسبت داده می شود .

- 1) 01## : 0000 S=220
- 2) 00#0 : 1100 S=208
- 3) 11## : 1000 S=196
- 4) ##00 : 0001 S=206

S=20 محیط

تولید قوانین جدید - دسته سطل ، یک روش انتخاب قوانین و انتساب اعتبار را ارایه می نماید .

چگونه قوانین جدید را به دست بیاوریم ؟ الگوریتم ژنتیکی پایه‌ی ما از یک مدل جمعیت که با هم اشتراک ندارند استفاده می نماید ؛ همه‌ی جمعیت در زمان t ، در زمان $t+1$ جایگزین می شود . به خوبی برای بهینه سازی عمل می کند ، ولی برای یادگیری ، زیاد مناسب نمی باشد . در عوض ، ما از نخبه سالاری¹ برای حفظ

elitism¹

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

برخی از قانون ها استفاده می نماییم ، از روش انتخاب رولت^۱ برای حفظ قانون ها استفاده نمایید ، البته این روش ، کندتر استنتاج می کند .

برنامه نویسی ژنتیکی - فراتر از قوانین - روش های تکاملی برای تولید برنامه ها به کار گرفته شده اند . ایده ای این روش این است که یک برنامه می تواند به صورت یک S- عبارتی بیان شود ، مثلاً :

$$(+ 3 (* 4 x))$$

ما می توانیم این را به صورت یک درخت بکشیم . ما سپس تکامل را روی یک جمعیت از برنامه ها انجام می دهیم . شایستگی ، کارایی یک برنامه برای یک عملکرد ارایه شده است . زیر Crossover درخت ها را با هم عوض می نماید . جهش ، عملگرها را عوض می نماید و برای تکامل استفاده می شوند ، مثال ها برنامه های فوتbal روبوت ، مدارهای کنترل آنالوگ ، فیلترها و مدارهای پیچیده می باشند . در این مورد چالش ها عبارتند از ، فضای پهناور (نامحدود) برنامه ها و کمبود ساختارهای سطح بالاتر برنامه .

roulette selection^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل بیست و سوم

درخت های

تصمیم گیری

Decision trees^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در گذشته، ما فرض کردیم که دانش اولیه توسط خبره‌ها به ما داده شده است و بر چگونگی استفاده از این دانش تمرکز کردیم. حال می‌خواهیم در این مورد صحبت کنیم که چگونه دانش را از راه مشاهده به دست آوریم و بر قانون‌های گزاره‌ای تمرکز می‌کنیم. مثلاً اگر هوا آفتابی و گرم باشد، آن گاه تنیس بازی کنید و می‌نویسیم:

sunny \wedge warm \rightarrow PlayTennis

یا اگر هوا خنک باشد و بارانی باشد یا باد شدید بوزد، آن گاه تنیس بازی نکنید و می‌نویسیم:

cool \wedge (rain \vee strongWind) $\rightarrow \neg$ PlayTennis

یادگیری

برای یک عامل، یادگیری به چه معنی می‌باشد؟ عامل، دانش جدید را دریافت می‌نماید، دانش جدید را به کار می‌گیرد، رفتارش را تغییر می‌دهد و در یک کار معین، معیار کارایی خود را بهبود می‌بخشد.

عامل‌های آموزشی

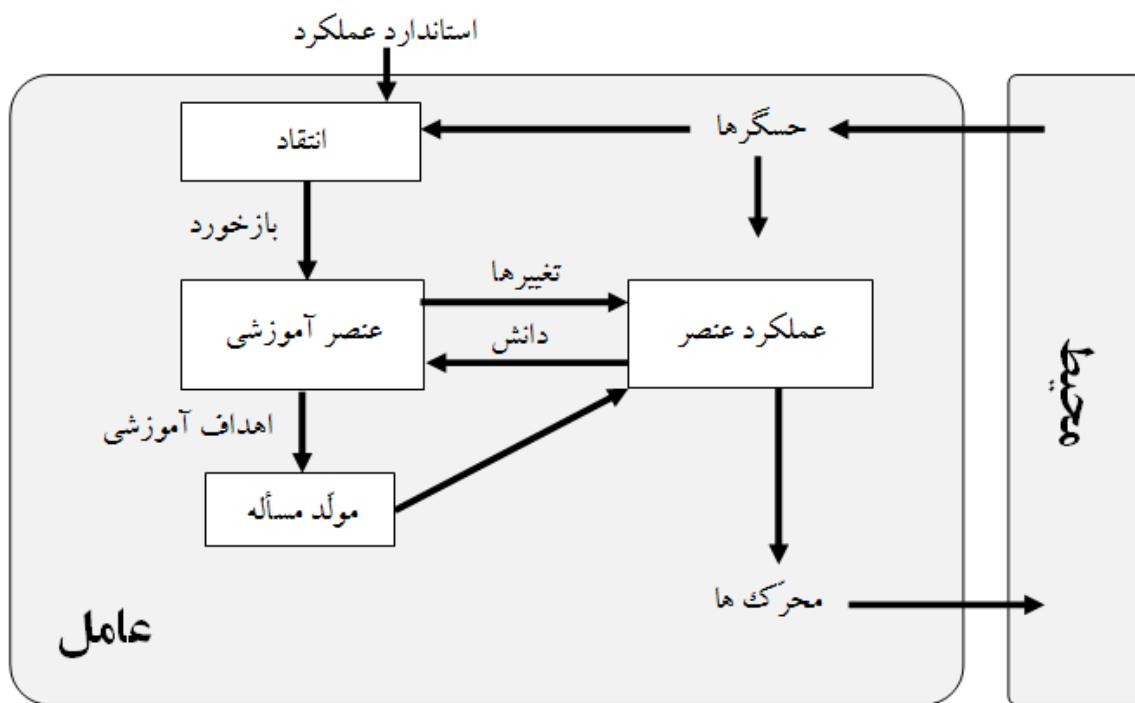
مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

به یاد بیاورید که ما قبلاً در مورد عامل های آموزشی صحبت کردیم:



یک عامل آموزشی دارای یک عنصر کارایی^۱ و یک عنصر آموزشی^۲ می باشد . عنصر کارایی ، چیزی است که یک عامل برای تصمیم گیری در مورد این که چه کاری انجام دهد از آن استفاده می کند و این چیزی است که ما تا کنون مطالعه کرده ایم . عنصر آموزشی ، چیزی است که به عامل برای بازنگری عنصر کارایی اجازه می دهد ، این ممکن است به معنی اضافه نمودن یا تغییر قانون ها یا واقعیات ، بازنگری

performance element^۱
learning element^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یک مکاشفه و تغییر یک تابع جانشین^۱ باشد. یک عامل ، برای بازنگری کردن رفتارش ، به اطلاعاتی که به عامل بگوید چگونه به خوبی کارش را انجام می دهد نیازمند می باشد ، این اطلاعات ، بازخورد^۲ نام دارد.

أنواع بازخورد

در اصل سه نوع عملکرد آموزشی وجود دارد که هر کدام بازخورد متفاوتی را دارد :

یادگیری نظارت شده^۳ (کنترل شده) که در این مورد ، یک منبع خارجی (که اغلب آموزش دهنده^۴ نام دارد) ، عامل را بآنموزه های برچسب زده شده^۵ ارایه می نماید . عامل ، موارد / عملکردهای معینی را در طول طبقه بندی اشان می بیند .

یادگیری بدون نظارت^۶ (کنترل نشده) که در این مورد ، آموزش دهنده ای برای ارائه ی نمونه ها وجود ندارد و عامل ، معمولاً برای پیدا کردن الگوهایی در داده ها تلاش می نماید .

یادگیری تقویتی^۷ که این ، یک نوع مخصوص از آموزش است که در آن عامل فقط یک پاداش را برای انجام یک عمل دریافت می نماید . ممکن است نداند یک پاداش بهینه چگونه می باشد و "بهترین" عملکرد را برای انجام نمی داند .

یادگیری نظارت شده (کنترل شده)

Successor function^۱

feedback^۲

Supervised learning^۳

teacher^۴

labeled examples^۵

unsupervised learning^۶

reinforcement learning^۷

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یکی از عمومی ترین شکل های آموزش می باشد . عامل با مجموعه ای از داده های برچسب زده شده ارایه می شود که باید از این داده ها برای تشخیص قانون های کلی تراستفاده نماید . نمونه ها ، لیست بیماران و ویژگی ها ، چه عامل هایی مرتبط با سرطان می باشند ؟ چه عواملی ، فردی را دارای خطر می داند ؟ بهترین سؤالات برای طبقه بندی حیوانات چیست ؟ صورت چه کسی در این تصویر می باشد ؟ این پردازش یادگیری قانون های کلی از واقعیات مشخص ، استنتاج^۱ نام دارد .

تعريف یک مسئله ی یادگیری - ما می توانیم مسئله ی یادگیری را با تخمین ، به صورت یک تابع f که به ما می گوید چگونه یک مجموعه از ورودی ها را طبقه بندی نماییم تفسیر کنیم . یک مثال در این مورد یک مجموعه از ورودی های x و $f(x)$ متناظر می باشد :

<<Mammal,Eats-Meat,Black-Stripes,Tawny>,Tiger>

ما می توانیم یک عملکرد آموزشی را به این صورت ، تعريف نماییم : یک مجموعه ای ارایه شده از نمونه های f ، یک تابع H که را برای مثال ما تخمین می زند پیدا نماید ، H ، فرض^۲ نام دارد .

استنتاج - ما ترجیح می دهیم H را تعمیم بدهیم ، این به این معنی است که H به درستی نمونه های دیده نشده را طبقه بندی می کند . در صورتی که فرض به درستی بتواند همه ی نمونه های آموزشی را طبقه بندی نماید ، ما آن را فرض سازگار^۳ می نامیم . هدف ، پیدا کردن یک فرض سازگار که در نمونه های دیده نشده هم به خوبی کار کند . ما می توانیم یادگیری را به صورت جستجو در میان یک فضای از فرض ها تصوّر نماییم .

induction^۱

hypothesis^۲

consistent hypothesis^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

بایاس استنتاجی - توجه نمایید که استنتاج، بی عیب نمی باشد. در انتخاب یک فرض، ما یک گمان آموزش داده شده را به وجود می آوریم. روشی که ما با استفاده از آن، این گمان را به وجود می آوریم بایاس نام دارد. نمونه ها، اصل اکام^۱، معین ترین فرض، کلی ترین فرض و تابع خطی می باشند.

مشاهده ی داده ها - عامل ها، ممکن است دارای ابزار مختلف مشاهده کننده ی نمونه های یک فرض باشند. یک الگوریتم یادگیری دسته ای^۲، یک مجموعه ی بزرگ داده ها که همه با هم ارایه می شوند و یک فرضیه ی منفرد را انتخاب می کنند، می باشد. یک الگوریتم یادگیری افزایشی، نمونه های همه در یک زمان را دریافت می کند و دائمًا فرض آن را بازنگری می نماید؛ دسته معمولاً با دقّت تر می باشد، اما افزایشی ممکن است با محیط عامل، مناسب تر باشد. یک عامل یادگیری فعال می تواند نمونه ها را انتخاب نماید. یک عامل یادگیری غیرفعال^۳، دارای نمونه های ارایه شده با آن به وسیله ی یک منبع خارجی هستند می باشد؛ یادگیری فعال توانمندتر می باشد، اما ممکن است با محدودیت های دامنه، مناسب نباشد.

درخت های تصمیم گیری یادگیری - درخت های تصمیم گیری، ساختمان داده هایی هستند که یک عامل را با ابزارهای طبقه بندی کننده ی نمونه ها ارایه می نمایند. در هر گره در درخت، یک ویژگی، تست می شود.

^۱Occam's razor، برای دو توصیف ارایه شده، آسان ترین آن ها را انتخاب نماید - توضیحی که به مفروضات کمتری نیاز دارد (لغت نامه ی وب برنامه ی babylon)

^۲batch

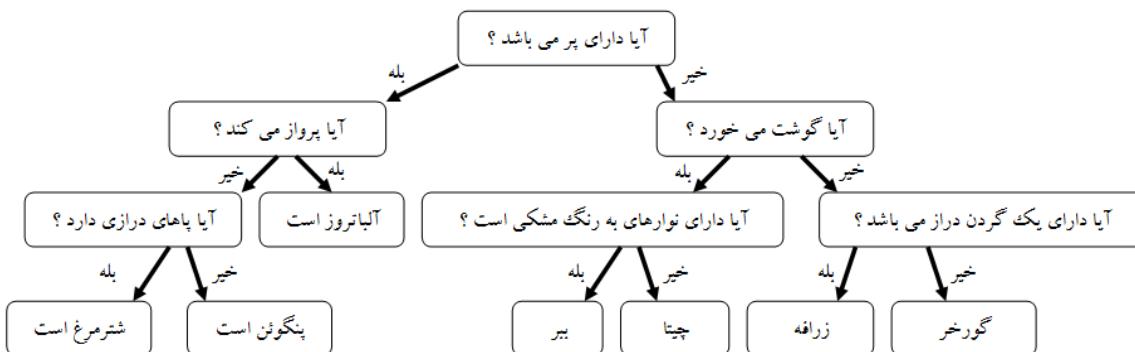
^۳passive

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



اطلاعات

در مورد دنیای جاروبرقی ، اتاق ها ممکن است تمیز یا کثیف باشند که برای بیان این مورد به یک بیت نیاز داریم . اگر یک اتاق دارای چهار وضعیت باشد یا هشت وضعیت چه طور ؟ اگر یک اتاق فقط دارای یک وضعیت باشد چه طور ؟ به چند بیت برای بیان نیاز داریم ؟

تئوری اطلاعات

به صورت رسمی تر ، باید بگوییم که n پاسخ ممکن v_1, v_2, \dots, v_n برای یک سؤال وجود دارد و هر جواب دارای احتمال رویداد $P(v_i)$ می باشد . محتوای اطلاعات ^۱ جواب برای سؤال به این صورت می باشد :

$$I = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$

برای یک سگ ، این برابر است با :

$$-\frac{1}{2} \log_2 \frac{1}{2} + -\frac{1}{2} \log_2 \frac{1}{2} = 1$$

information content ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

سؤالاتی با یک احتمال زیاد جواب ، دارای محتوای اطلاعاتی کمی خواهد بود . (در صورتی که سگه در یک بار $\frac{99}{100}$ شیر بیاید ، $I=0.08$ خواهد بود .) ، محتوای اطلاعاتی ، برخی اوقات بی نظمی (آنتروپی^۱) نامیده می شود و اغلب در فشرده سازی و الگوریتم های انتقال داده ها استفاده می شود .

استفاده از تئوری اطلاعات

برای درخت های تصمیم گیری ، ما می خواهیم بدانیم ارزشمندی هر مورد ممکن چگونه می باشد ، یا چه میزان از اطلاعات را نتیجه می دهد . ما می خواهیم احتمال جواب های ممکن از یک مجموعه مورد آموخت را تخمین بزنیم . معمولاً ، یک مورد تکی برای به طور کامل مجزا نمودن نمونه های مثبت و منفی کافی نمی باشد . در عوض ، ما نیاز داریم در مورد این فکر کنیم که چه مقدار بهتری بعد از سؤال وجود خواهد داشت و این ، سود اطلاعات^۲ نام دارد .

سود اطلاعات

در صورتی که یک مجموعه مورد آزمایش دارای p نمونه مثبت و n نمونه منفی باشد ، بی نظمی یا آنتروپی به صورت زیر خواهد بود :

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

ما می خواهیم خصوصیتی که برای مجزا کردن نمونه های مثبت و منفی ، نزدیک ترین می باشد را پیدا نماییم . ما با محاسبه باقی مانده شروع می کنیم - این اطلاعاتی است که هنوز در داده ها بعد از این که ما ویژگی A را تست نمودیم وجود دارد . می گوییم ویژگی A می تواند مقادیر ممکن 7 را به کار بگیرد . این تست ،

entropy^۱

information gain^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

V زیر مجموعه‌ی جدید از داده‌ها را به نام E_1, E_2, \dots, E_v خواهد ساخت. باقی مانده، مجموع اطلاعات موجود در هر کدام از این زیرمجموعه‌ها می‌باشد.

$$\text{Re } \text{mainder } (A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} * I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

سود اطلاعات سپس می‌تواند به صورت تفاوت میان اطلاعات اصلی (قبل از آزمایش) و اطلاعات جدید (بعد از آزمایش) بیان شود.

$$\text{Gain}(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{Re } \text{mainder } (A)$$

مکاشفه: ویژگی با بیش ترین سود اطلاعات را انتخاب نماید.

سؤال: این کدام نوع از جستجو می‌باشد؟

مثال:

روز	پیش‌بینی	دما	رطوبت	باد	تنیس بازی
D1	آفتابی	گرم	بالا	ملایم	نه
D2	آفتابی	گرم	بالا	شدید	نه
D3	ابری	گرم	بالا	ملایم	بله
D4	بارانی	متوسط	بالا	ملایم	بله
D5	بارانی	خنک	معمولی	ملایم	بله
D6	بارانی	خنک	معمولی	شدید	نه
D7	ابری	خنک	معمولی	شدید	بله
D8	آفتابی	متوسط	بالا	ملایم	نه

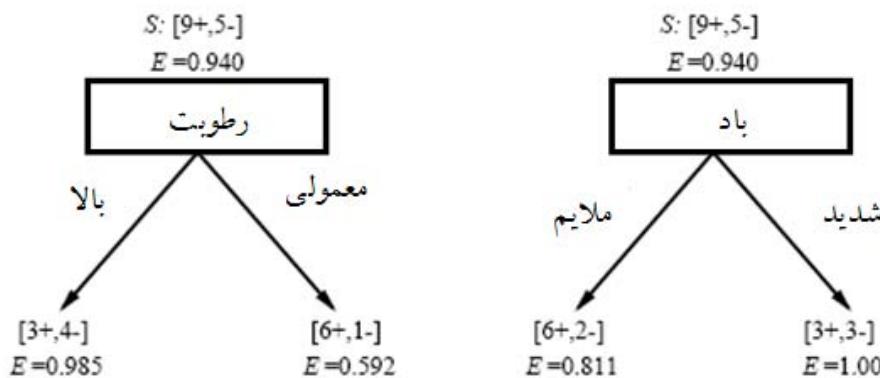
مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

D9	آفتایی	خنک	معمولی	ملايم	بله
D10	بارانی	متوسط	معمولی	ملايم	بله
D11	آفتایی	متوسط	معمولی	شدید	بله
D12	ابری	متوسط	بالا	شدید	بله
D13	ابری	گرم	معمولی	ملايم	بله
D14	بارانی	متوسط	بالا	شدید	نه

کدام ویژگی، بهترین طبقه بندی می باشد؟



$$\text{Grain}(S, \text{رطوبت}) = 0.940 - (7/14) \cdot 0.985 - (7/14) \cdot 0.592 = .151$$

$$\text{Gain}(S, \text{باد}) = 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1.0 = .048$$

اغتشاش یا پارازیت ^۱

Noise ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

در صورتی که دو نمونه‌ی دارای ویژگی‌های یکسان ولی مقدارهای مختلف وجود داشته باشد، یک درخت تصمیم‌گیری قادر به طبقه‌بندی آن‌ها به صورت مجزاً نخواهد بود و در این صورت می‌گوییم که این داده‌ها دارای اغتشاش می‌باشند.

ورودی‌های پیوسته‌ی مقداردهی شده - درخت‌های تصمیم‌گیری همچنین می‌توانند برای کار با ورودی‌های صحیح و مقداردهی شده به صورت پیوسته توسعه داده شوند. در این مورد برای پیدا کردن مقداری که بالاترین سود اطلاعات را نتیجه می‌دهد از جستجوی تپه نورده استفاده نمایید.

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

فصل بیست و چهارم

یادگیری Q ای'

Q - learning '

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

ما به محیط های تصادفی جذب شده ایم . ما می توانیم این مسئله را به صورت یک پروسه ی تصمیم گیری مارکوفی فرمول بندی نماییم ؛ مسئله دارای یک حالت اولیه s_0 ، یک مجموعه ی گستره از وضعیت ها و عملکردها ، یک مدل انتقالی $T(s,a,s')$ که احتمال رسیدن وضعیت s' از s در موقع انجام عمل a می باشد و یک تابع پاداش $(R(s))$ می باشد .

روش ها^۱

یک روش ، راه حلی برای یک پردازه ی تصمیم گیری مارکوفی می باشد و عملکرد بهینه را برای انجام در هر وضعیت ، مشخص می نماید . عملکرد بهینه ، عملکردی است که سودمندی مورد انتظار کاهاش داده شده از این وضعیت را بیشینه می نماید . بنابراین ، سودمندی یک وضعیت ، پاداشی برای آن وضعیت ، به اضافه ی پاداش کم شده ی مورد انتظار برای دنبال کردن روش بهینه از این وضعیت می باشد . داریم :

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

policies^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

که عبارت فوق معادله‌ی بلمن^۱ نام دارد.

تکرار مقدار^۲

به صورت مستقیم نمی‌توان معادله‌ی بلمن را حل نمود. ما می‌توانیم از تکرار مقدار برای حل یک سیستم معادلات بلمن استفاده نماییم: به سودمندی‌های غیرپایانی، مقدارهای تصادفی نسبت دهید. برای هر وضعیت، عملکرد بهینه‌ی ارایه کننده‌ی این مقادیر را محاسبه نمایید. از این روش برای به روز رسانی سودمندی تخمین زده شده‌ی وضعیت استفاده نمایید و این کار را تا زمان نزدیک شدن به وضعیت مطلوب (همگرایی)، ادامه دهید.

تکرار روش^۳

در روش تکرار مقدار؛ حتی در زمانی که روش تغییر نمی‌کند، همگرایی، کند می‌باشد و ما واقعاً نمی‌توانیم نسبت به سودمندی یک وضعیت، بی تفاوت باشیم. تکرار روش، روش‌ها را به صورت مستقیم به روز رسانی می‌نماید.

یادگیری یک روش

تا حالا، ما فرض کردیم که دارای دانش زیادی هستیم. در حالت به خصوص، ما فرض کرده‌ایم که مدلی از جهان شناخته شده است؛ این، مدل انتقال وضعیت می‌باشد. اگر ما یک مدل نداشته باشیم چطور؟ همه‌ی چیزی که ما می‌دانیم این است که مجموعه‌ای از وضعیت‌ها و یک مجموعه از عملکردها وجود دارند. ما هنوز می‌خواهیم یک روش بهینه را یاد بگیریم.

Bellman equation^۱

Value Iteration^۲

Policy Iteration^۳

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

یادگیری Q ای

آموزش یک روش به صورت مستقیم، مشکل می باشد . مسأله این است که داده های ما به یک شکل نمی باشند : $\langle \text{وضعیت} , \text{عملکرد} \rangle$. در عوض ، به شکل $R, s_1, s_2, s_3, \dots, s_d$ می باشد . از آنجایی که ما تابع انتقال را نمی دانیم ، مشکل است که سودمندی یک وضعیت را یاد بگیریم . در عوض ، ما یک تابع $Q(s,a)$ را یاد خواهیم گرفت که "سودمندی" انجام عملکرد a را در وضعیت s ، تخمین می زند . به طور صریح تر ، $Q(s,a)$ مقدار a را در وضعیت s را ارایه می کند و بعد از این به صورت بهینه عمل می کند .

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

روش بهینه این است که عملکرد با بالاترین مقدار Q را در هر وضعیت ، به دست آوریم . اگر عامل بتواند $Q(s,a)$ را یاد بگیرد ، می تواند عملکردهای بهینه را حتی بدون دانستن تابع پاداش یا تابع انتقال داشته باشد .

یادگیری تابع Q

برای یادگیری Q ، ما باید بتوانیم مقدار یک عملکرد را در یک وضعیت ، حتی اگر پاداش ها در طول زمان پخش شده اند تخمین بزنیم . ما می توانیم این کار را به صورت تکراری انجام دهیم . توجه کنید که $U(s) = \max_a Q(s,a)$. سپس ما می توانیم معادله را برای Q به صورت زیر بازنویسی نماییم :

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a')$$

بیایید تخمین $Q(s,a)$ را به صورت $\hat{Q}(s,a)$ ، مشخص نماییم . ما یک جدول لیست کننده i هر جفت وضعیت - عملکرد و مقدار تخمین زده شده i Q را نگهداری می نماییم . عامل ، وضعیت s را در کمی کند ، عملکرد a را انتخاب می کند و سپس در کمی کند که پاداش برابر با $R(s,a) = r$ می باشد ، که

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

آن را دریافت می کند و وضعیت جدید S' می باشد . سپس جدول Q را با توجه به فرمول زیر به روز می کند :

$$\hat{Q}(s, a) = r + \gamma \max_{a'} \hat{Q}(s', a')$$

عامل ، تخمینی از \hat{Q} را برای s' برای تخمین \hat{Q} برای s ، استفاده می نماید . توجه کنید که عامل نیاز به هیچ دانشی در مورد R یاتابع انتقال برای اجرای این کار ندارد . آموزش Q ای برای نزدیک شدن (همگرایی) ضمانت شده است به شرطی که پاداش ها محدود شده باشند ، عامل جفت وضعیت - عملکرد های این شکلی را به روشنی که معمولاً نامحدود می باشد انتخاب می کند . این به این معنی است که یک عامل باید دارای احتمالی غیر صفر از انتخاب هر a در هر s به صورت رشتہ ای از جفت های وضعیت - عملکرد با نگرش نامحدود باشد .

۱. کتشاف

بنابراین چگونه آن را به وجود بیاوریم ؟ یادگیری Q ای دارای یک تفاوت مشخص از دیگر الگوریتم هایی که تا کنون دیده ایم می باشد و آن این است که عامل می تواند عملکردها و دیدی که آن ها به دست می آورند را انتخاب نماید و این یادگیری پویا ^۱ نام دارد . عامل ، مایل به بیشینه کردن عملکرد می باشد ؛ این به این معنی است که کاری را که در حال حاضر به نظر می رسد بهترین باشد را انجام می دهد ، اما اگر عامل هیچ گاه عملکردهای با " دید بد " نداشته باشد ، نمی تواند از خطاهای بیرون بیابد . \hat{Q} خیلی بادقت نمی باشد ، بنابراین ، عملکردهای غیر بهینه را امتحان خواهیم کرد . از این پس ، در صورتی که \hat{Q} بهتر شود ، ما عملکردهای بهینه را انتخاب خواهیم کرد .

active learning ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

اکتشاف بولتزمان^۱

یک راه برای انجام این کار استفاده از اکتشاف بولتزمان است. ما یک عمل با احتمال زیر را انجام می دهیم:

$$P(a | s) = \frac{k^{\hat{Q}(s,a)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

که k پارامتر دما می باشد. این فرمول ، شبیه به فرمولی است که ما در روش شبیه سازی گرم و سرد کردن استفاده کردیم.

یادگیری مستحکم^۲

یادگیری Q ای نمونه ای از آن چه که ما یادگیری مستحکم می نامیم می باشد. در یادگیری مستحکم ، عامل ها نمونه هایی از این که چگونه عمل نمایند را نمی بینند. در عوض ، آن ها عملکردها را انتخاب می نمایند و پاداش ها یا مجازات ها را دریافت می نمایند. یک عامل می تواند یاد بگیرد ، حتی در زمانی که عمل غیر بهینه ای را انجام می دهد. تعیین ها به پاداش تأخیر داده شده و پردازه های غیرقطعی تصمیم گیری مارکوفی رسیدگی می کنند.

خلاصه

یادگیری Q ای برخی اوقات به صورت یادگیری مستقل از مدل^۱ شناخته می شود. در این روش ، عامل فقط نیاز به انتخاب عملکردها و دریافت پاداش ها دارد. مشکلات این روش ، چگونه تعیین بدهد؟ ،

Boltzmann exploration^۱
reinforcement learning^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مقیاس پذیری^۲ و سرعت نزدیک شدن (همگرایی) می باشد. یادگیری Q ای برای یک الگوریتم مفید تجربی به سطح مطلوبی رسیده است.

model-free learning^۱
scalability^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸

هوش مصنوعی



فصل بیست و پنجم

برنامه ریزی^۱

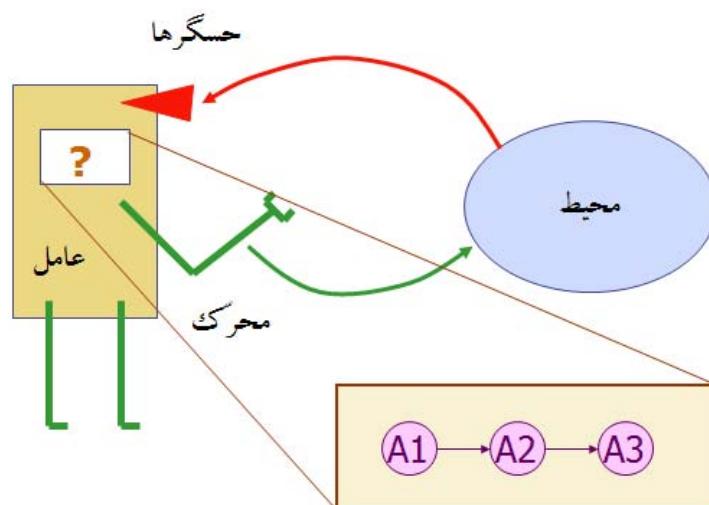
planning^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عامل برنامه ریزی



برنامه ریزی

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸

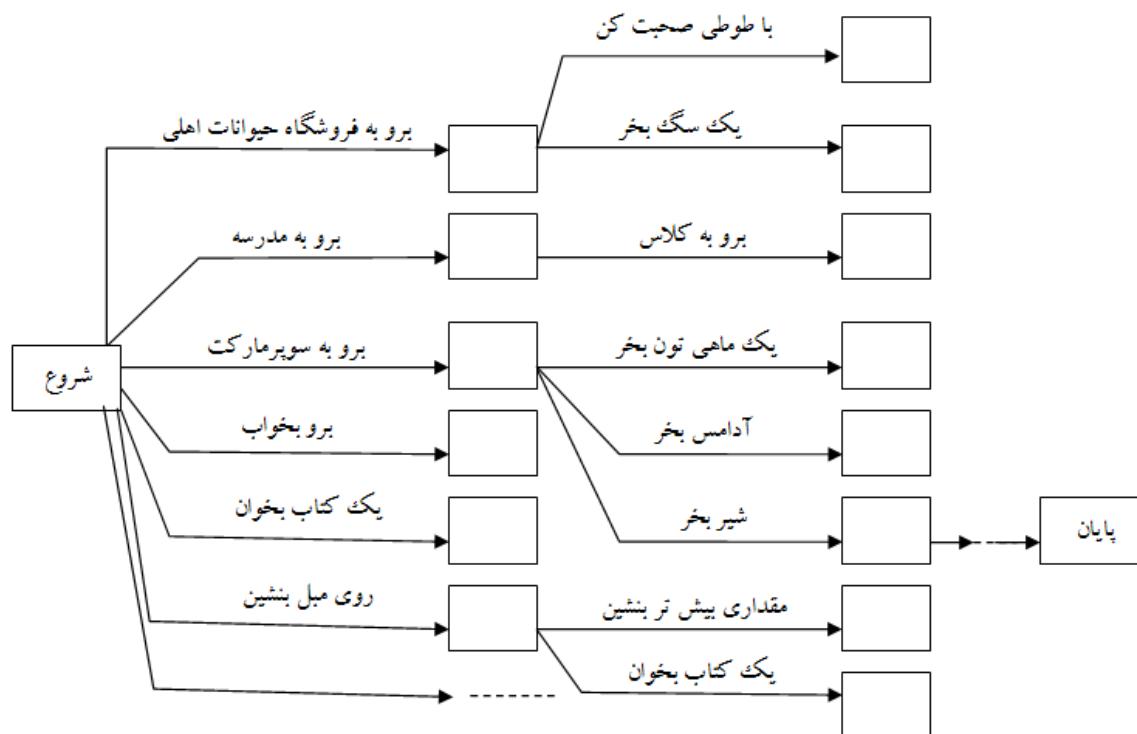


هوش مصنوعی

برنامه ریزی ، تعبیه کردن یک رشته از عملکردها برای دسترسی به یک هدف می باشد . برنامه ریزی کلاسیک ، کاملاً قابل مشاهده ، قطعی ، محدود ، ثابت^۱ و گستته است . زبان مشخصی برای ارایه ی مسایل برنامه ریزی وجود دارد .

جستجو و برنامه ریزی

الگوریتم های جستجوی استاندارد به نظر می رسد که به سختی شکست بخورند :



static^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مشکلاتی که در روش جستجو وجود دارد عبارتند از: تعداد زیادی عملکردهای نامرتب وجود دارد، پیدا کردن مکافه های خوب مشکل می باشد و از تجزیه‌ی مسئله نمی توان سودی برد. به صورت خودکار مکافه ها از ارایه مشتق می شوند؛ مانند مسئله‌ی اراضی محدودیت.

مقایسه‌ی برنامه ریزی با منطق

برنامه ریزی	جستجو	
عبارت های منطقی	ساختمان داده های جاوا	وضعیت ها
پیش شرط ها ^۱ / نتیجه ها	کد جاوا	عملکردها
عبارت منطقی	کد جاوا	هدف
محدودیت های روی عملکردها	رشته ای از S_0	برنامه

زبانی برای مسایل برنامه ریزی

حل کننده‌ی مسئله‌ی مؤسسه‌ی تحقیق استنفورد^۲ (STRIPS) : جهان توسعه شرط های منطقی توصیف می شود، وضعیت به صورت اجتماع حرف های مثبت می باشد و می تواند به صورت

preconditions^۱
STanford Research Institute Problem Solver (STRIPS)^۲

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

گزاره ای ؟ به عنوان مثال ، Happy \wedge Hungry برای نمایش وضعیت عامل و یا منطق مرتبه ای اول و عبارت های مستقل از تابع باشد ؟ به عنوان مثال ،

$$At(Plane_1, Verona) \wedge At(Plane_2, Malpensa)$$

فرض جهان بسته ؟ هر شرطی که نام برده نشده غلط می باشد .

هدف ، وضعیتی است که به صورت جزیی مشخص شده است . اگر محدودیت ها همه لفظ هایی از هدف باشند یک وضعیت یک هدف را ارضا می نماید . به عنوان مثال ،

$$At(Plane_1, Verona) \wedge At(Plane_2, Malpensa)$$

هدف ($At(Plane_2, Malpensa)$) را ارضا می نماید .

عملکردهای استریپس

عملکردها تو سط موارد زیر مشخص می شوند :

پیش شرط ها ^۱ : زمانی را مشخص می کنند که عملکرد می تواند به کار گرفته شود .

اثرات : وضعیت تو سط عملکردها تغییر می یابد . در این مورد ، لیست اضافه ، گزاره هایی هستند که صحیح از آب در می آیند و لیست حذف ، گزاره هایی که غلط از آب در می آیند .

عملکردها شامل متغیرها می باشند ، یک طرح عملکرد منفرد عملکردهای مختلف را نمایش می دهد (نمونه ای از متغیرها) . توصیف عملکردها به صورت منظم می باشد و زبان ، محدود شده است .

طرح عملکرد :

preconditions ^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$At(p) \wedge Sells(p, x)$

Buy(x)

Have(x)

Buy(x)

پیش شرط ، $At(p) \wedge Sells(p, x)$ می باشد و اثر ، $Buy(x)$ است و باید به این نکته توجه کنیم که ، هیچ اطلاعاتی در مورد چگونگی اجرای عملکرد وجود ندارد ! . چون که زبان محدود شده است پس الگوریتم کارا می باشد . عملکرد ، نام و لیست پارامتر را مشخص می نماید . پیش شرط ، اجتماع لفظ های مثبت و اثر ، اجتماع لفظ ها (مثبت یا منفی) است . یک مجموعه ای کامل از عملگرهای استریپس می تواند به یک مجموعه از اصول جانشین وضعیت^۱ ترجمه شود .

معانی^۲

با داشتن یک وضعیت (اجتماع لفظ ها) ، اگر انتساب یک متغیر ، در ارتباط با لفظ های شامل شده ی وضعیت باشد ، پیش شرط ارضا می شود ؛ به عنوان مثال ، وضعیت $At(HW) \wedge Sell(HW, Drill)$ ، پیش شرط $(At(p) \wedge Sell(p, x)) / Drill$ را با انتساب p/HW و $x/Drill$ راضی می کند .

عملکردهای با پیش شرط های ارضا شده می توانند در این موارد به کار گرفته شوند : حذف عناصر از لیست حذف ، اضافه نمودن عناصر به لیست اضافه ، وضعیت جدید :

$At(HW) \wedge Sell(HW, Drill) \wedge Have(Drill)$

مثال : خرید

successor – state axioms^۱
semantics^۲

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

عملکردها عبارتند از :

Buy(x)

پیش شرط : Sells(store,x) و At(store)

اثر : Have(x)

Go(x,y)

پیش شرط : At(x)

اثر : $\neg At(x)$ و $At(y)$

شروع :

$At(Home) \wedge Sells(SM, Milk) \wedge Sells(SM, Banana) \wedge Sells(HWS, Drill)$

هدف :

$Have(Milk) \wedge Have(Banana) \wedge Have(Drill)$

مثال : انتقال محموله‌ی هوایی

عملکردها :

Load(c,p,a)

پیش شرط : $At(c, a) \wedge At(p, a) \wedge \text{Cargo}(c) \wedge Plane(p) \wedge Airport(a)$

اثر : $\neg At(c, a) \wedge In(c, p)$

Unload(c,p,a)

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

$In(c, p) \wedge At(p, a) \wedge \text{Cargo}(c) \wedge \text{Plane}(p) \wedge \text{Airport}(a)$: پیش شرط :

$At(c, a) \wedge \neg In(c, p)$: اثر :

$\text{Fly}(p, \text{from}, \text{to})$

$At(p, \text{from}) \wedge \text{Plane}(p) \wedge \text{Airport}(\text{from}) \wedge \text{Airport}(\text{to})$: پیش شرط :

$\neg At(p, \text{from}) \wedge At(p, \text{to})$: اثر :

عملکردها عبارتند از Unload ، Load ، Fly . گزاره ها ، $\text{In}(0,0)$ و $\text{At}(0,0)$ نتیجه ی Load و Unload می باشند . انواع گزاره ها هم $\text{Cargo}(0)$ ، $\text{Plane}(0)$ و $\text{Airport}(0)$ هستند .

شروع :

$At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
 $\wedge \text{Cargo}(C_1) \wedge \text{Cargo}(C_2) \wedge \text{Plane}(P_1) \wedge \text{Plane}(P_2)$
 $\wedge \text{Airport}(JFK) \wedge \text{Airport}(SFO)$

. $At(C_1, JFK) \wedge At(C_2, SFO)$: هدف :

یک راه حل به صورت زیر می باشد :

$Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$
 $Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)$

برنامه ریزی استریپس

مسئله ی برنامه ریزی استریپس : یک رشته از عملکردهایی که به هدف منجر می شوند را پیدا نمایید . وضعیت ها و هدف ها توسط یک اجتماع از لفظ ها تعریف می شوند .

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

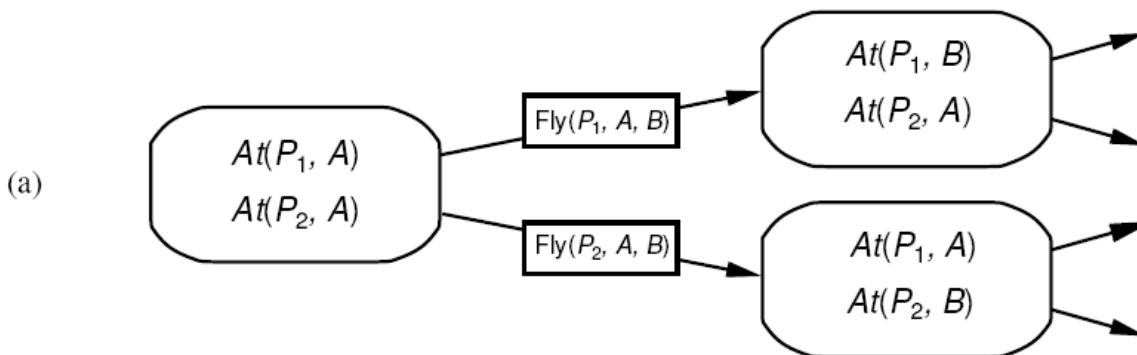
فضای حالت جستجو: جستجوی مستقیم از وضعیت اولیه تلاش می کند تا به هدف برسد.

جستجوی معکوس از هدف تلاش و طرح ریزی می کند که به وضعیت ابتدایی برسد.

فضای برنامه‌ی جستجو: برنامه ریزی با مرتبه‌ی جزیی^۱، فضای جزیی ساخته شده توسط طرح ها را جستجو می کند.

فضای حالت جستجو

مسئله‌ی برنامه ریزی، مسئله‌ی جستجو را تعریف می کند. وضعیت اولیه، وضعیت شروع می باشد، آزمون هدف، بررسی می کند که آیا هدف برآورده (ارضا) می شود یا نه، عملکردها، عملگرها را تعریف می کنند، هزینه مرحله، معمولاً^۱ می باشد. در زیر، شکل a، جستجوی مستقیم و شکل b، جستجوی معکوس می باشد:



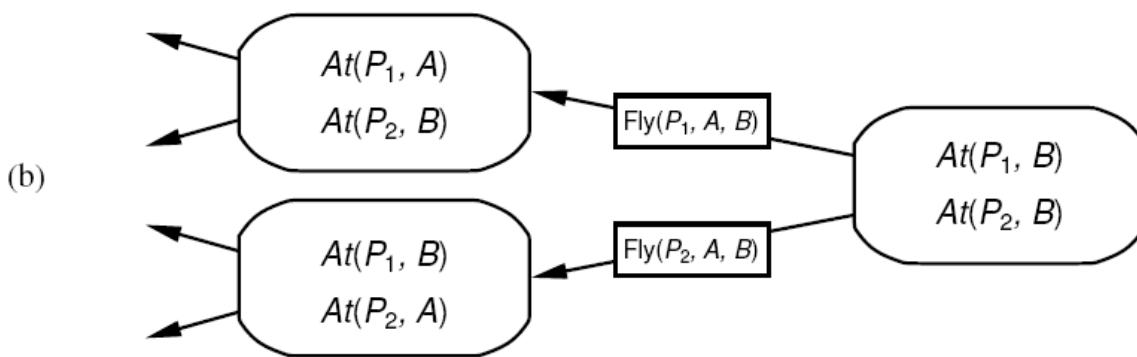
Partial-Order Planning(POP)^۱

مترجم: سهراب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی



جستجوی مستقیم - حلقه‌ی جستجوی اصلی : یک عملکرد را انتخاب نماید و پیش شرط را با وضعیت یکی نمایید . در صورتی که پیش شرط برآورده شد ، عملکرد تولید کننده‌ی یک وضعیت جدید را به کار ببرید . بررسی کنید که آیا وضعیت جدید ، هدف را برآورده می نماید یا نه . در صورتی که فضای حالت ، محدود باشد ، برای الگوریتم های جستجوی کامل ، کامل می باشد . این روش به دلیل عملکردهای نامربوط ، ناکارآمد می باشد و به مکافهه های خوب ، نیازمند می باشد .

جستجوی معکوس - ایده‌ی این روش این است که عملکردهای مربوط را فقط با شروع از هدف انتخاب نمایید . عملکرد ، در زمانی که به یکی از متّحدانش دست می یابد (لیست اضافه) ، مربوط می باشد . در زمانی که هر لفظ مربوط را بی اثر^۱ نمی کند (لیست حذف) ، سازگار می باشد .

یک عملکرد مربوط و سازگار را انتخاب نماید و وضعیت جدید را تولید نمایید : لیست اضافه را پاک نمایید و پیش شرط ها را اضافه نمایید .

این روش با یک وضعیت برآورده شده توسعه وضعیت ابتدایی خاتمه می یابد و روی وضعیت های مربوط و سازگار منشعب می شود .

فضای حالت و فضای برنامه

undo^۱

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

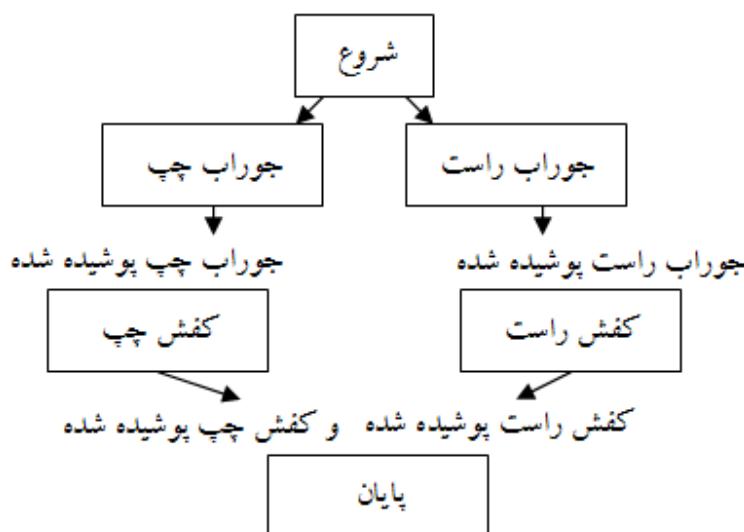
جستجوی مستقیم و معکوس، رشته های خطی عملکردها را اکتشاف می نماید که این کار لازم نمی باشد؛ به راه حل انتقال هوایی محموله توجه نمایید:

$Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$
 $Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)$

تنها ترتیب لازم، میان $Unload$ ، Fly و $Load$ می باشد که توسط اثرات سببی به هم متصل شده اند؛ به عنوان مثال، ($In(C_1, P_1)$ به $Load(C_1, P_1, SFO)$) استفاده شده توسط پیش شرط $Unload$ دست می یابد. نیازی نیست که ($Load(C_2, P_2, JFK)$) بعد از ($Load(C_1, P_1, JFK)$) قرار دهیم.

طرح با مرتبه ای جزیی^۱: گراف عملکردها شامل شروع و پایان می باشد (گراف \equiv مرتبه ای جزیی \succ)

مثال: طرح با مرتبه ای جزیی:



partial-order plan^۱

مترجم: سه راب جلوه گر

ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

طرح های با مرتبه ای کلی :



طرح های منظم شده به صورت جزئی^۱ - مجموعه ای از مراحل با مدخله ای شروع^۲

دارای توصیف وضعیت اولیه به صورت اثرش می باشد، مرحله ای پایانی^۳ دارای توصیف هدف به صورت پیش شرطش می باشد، اتصالات سببی^۴ از نتیجه های یک مرحله به پیش شرط دیگری به وجود می آیند و مرتب سازی زمانی^۵ میان جفت های مراحل می باشد.

شرط باز^۶، پیش شرط یک گام که هنوز به صورت سببی متصل نشده است می باشد. طرح در صورتی کامل است که هر پیش شرط در دسترس باشد، پیش شرط در صورتی قابل دسترس می باشد که اثر یک

start step^۱

finish step^۲

causal links^۳

temporal ordering^۴

open condition^۵

مترجم: سهراب جلوه گر

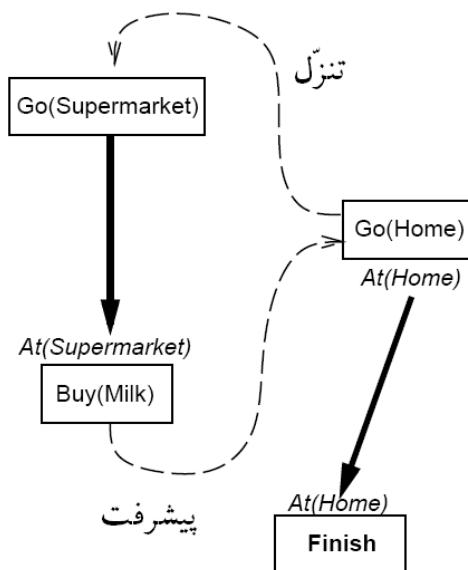
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

مرحله قبل تر باشد و هیچ مرحله‌ی میانی، آن را بی اثر ننماید (تناقض نداشته باشند). طرح‌ها نمی‌توانند دارای حلقه باشند.

ناسازگاری‌ها و پیشرفت^۱ / تنزل^۲ - یک عملکرد ناسازگار یک مرحله‌ی میانی بالقوه است که شرط قابل دسترس توسط اتصال سبیی را خراب می‌نماید. به عنوان مثال، Go(Home)، At(Supermarket) را حذف می‌نماید:



تنزل: قبل از Go(Supermarket) قرار دهید

پیشرفت: بعد از Buy(Milk) قرار دهید

promotion^۱
demotion^۲

مترجم: سهراب جلوه گر
ویرایش دوم ، بهار ۱۳۸۸



هوش مصنوعی

فهرست برخی از منابع و مآخذ

بیش تر مطالب استفاده شده برای ترجمه‌ی این کتاب ، مطالب به زبان انگلیسی موجود در پایگاه های اینترنتی دانشگاه های کشور ایالات متحده‌ی آمریکا می باشد . در زیر ، فهرست برخی از پایگاه های اینترنتی ای که در ترجمه‌ی این کتاب از آن ها استفاده شده را مشاهده می نمایید :

۱- مطالب موجود در پایگاه اینترنتی دانشگاه سانفرانسیسکو ایالات متحده‌ی آمریکا ، به نشانی www.cs.usfca.edu اینترنتی

۲- مطالب موجود در پایگاه اینترنتی دانشکده‌ی برکلی دانشگاه کالیفرنیا ایالات متحده‌ی آمریکا ، با آدرس اینترنتی <http://www.cs.berkeley.edu> (مطالب استوارت راسل و پیتر نورویگ)

۳- مطالب موجود در پایگاه اینترنتی دانشکده‌ی علوم کامپیوتر دانشگاه دولتی تگزاس ایالات متحده‌ی آمریکا ، به آدرس اینترنتی www.cs.txstate.edu

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

- ۴- مطالب موجود در پایگاه اینترنتی <http://www-formal.stanford.edu> (مطالب جان مک کارتی، استاد دانشگاه استنفورد)
- ۵- مطالب گردون اس. نواک جی. آر^۱ استاد دانشگاه تگزاس ایالت متحده ای آمریکا به نشانی www.cs.utexas.edu اینترنتی
- ۶- مطالب Milos Hauskrecht استاد دانشگاه شهر پیتسبرگ کشور ایالت متحده ای آمریکا به نشانی اینترنتی www.cs.pitt.edu
- ۷- مطالب موجود در پایگاه اینترنتی دانشگاه مریلند کشور ایالت متحده ای آمریکا با آدرس <http://www.cs.umd.edu> اینترنتی
- ۸- مطالب موجود در پایگاه اینترنتی مدرسه علوم کامپیوتر و مهندسی، به نشانی اینترنتی www.cse.unsw.edu.au
- ۹- مطالب موجود در پایگاه اینترنتی جامعه هوش محاسباتی، به نشانی اینترنتی <http://ebrains.la.asu.edu/~jennie/tutorial/index.htm>
- ۱۰- مطالب موجود در پایگاه اینترنتی مهندسان سیاهپوست، به آدرس اینترنتی www.nsbe.org
- ۱۱- مطالب موجود در پایگاه اینترنتی دانشگاه آزاد بوزن ایتالیا، به نشانی اینترنتی <http://www.inf.unibz.it>
- ۱۲- مطالب موجود در پایگاه اینترنتی دانشگاه کوبلنц آلمان، به نشانی اینترنتی <http://www.uni-koblenz.de>

Gordon S. Novak Jr. - ^۱

مترجم: سهراب جلوه گر
ویرایش دوم، بهار ۱۳۸۸



هوش مصنوعی

۱۳- مطالب موجود در پایگاه اینترنتی دانشکده علوم کامپیوتر دانشگاه کورک ایرلند، با آدرس

اینترنتی <http://www.cs.ucc.ie>

۱۴- مطالب موجود در پایگاه اینترنتی دانشکده علوم کامپیوتر و اطلاعات نروژ، به آدرس

اینترنتی www.idi.ntnu.no

۱۵- مطالب موجود در پایگاه اینترنتی دانشگاه ملی تایوان، به آدرس اینترنتی

www.csie.ntu.edu.tw

۱۶- مطالب موجود در پایگاه اینترنتی دانشگاه لیورپول^۱ انگلستان به نشانی اینترنتی

<http://www.csc.liv.ac.uk>

۱۷- مطالب موجود در پایگاه اینترنتی دانشگاه خصوصی سینساي^۲ کشور ترکیه

۱۸- مطالب موجود در پایگاه اینترنتی دانشگاه شمال غربی^۳ کشور ایالات متحده ای آمریکا به نشانی

اینترنتی <http://www.eecs.northwestern.edu>

۱۹- دانشکده علوم کامپیوتر دانشگاه بولوگنا کشور ایتالیا به نشانی اینترنتی

<http://www.cs.unibo.it>

Liverpool - ^۱

Sabancı University - ^۲

NORTHWESTERN UNIVERSITY - ^۳