دانشکده علوم ریاضی
گروه ریاضی محض

پروژه کارشناسی ریاضی گرایش هندسه جبری محاسباتی

عنوان

یادگیری تقویتی در الگوریتم بوخبرگر

پژوهشگر

ستاره غفوری

استاد راهنما

دکتر امیر هاشمی

تقدیم به:

پدر و مادرم

چکیده

پایه‌های گربنر ابزارهای قدرتمندی در جبر جابجایی و هندسه جبری هستند که برای حل طیف گسترده‌ای از مسائل در علوم مختلف به کار می‌روند. الگوریتم بوخبرگر، الگوریتم اصلی برای محاسبه این پایه‌ها است، اما هزینه محاسباتی بالای آن، یک چالش مهم محسوب می‌شود. در این پروژه، به دنبال بهبود کارایی و بهینه‌سازی الگوریتم بوخبرگر با بهره‌گیری از روش‌های یادگیری تقویتی هستیم.

واژگان کلیدی پایه گربنر، یادگیری تقویتی، الگوریتم بوخبرگر

فهرست مطالب

فصل ۱

مقدمه

ظهور جبر جابجایی و هندسه جبری، توانایی ما را در حل سیستم‌های پیچیده چندجمله‌ای به‌طور قابل توجهی افزایش داده است و راه را برای طیف گسترده‌ای از کاربردهای علمی در حوزه‌های مختلف هموار کرده است. یکی از ابزارهای مرکزی در این زمینه، مفهوم پایه‌های گربنر است که مجموعه‌ای از چندجمله‌ای‌ها هستند که امکان حل سوالات بنیادی مربوط به این سیستم‌ها را فراهم می‌کنند. از حذف متغیرها تا پارامترسازی راه حل‌ها و مطالعه خواص هندسی مجموعه‌های راه حل، پایه‌های گربنر به عنوان سنگ بنای بسیاری از الگوریتم‌ها عمل می‌کنند. کاربردهای آن‌ها در حوزه‌های مختلفی مانند بینایی رایانه، رمزنگاری، شبکه‌های بیولوژیکی، رباتیک، آمار و بسیاری دیگر گسترده است.

در قلب محاسبه پایه‌های گربنر، الگوریتم بوخبرگر قرار دارد، یک فرایند تکرار بنیادی طراحی شده برای استخراج این پایه‌های ضروری. با این حال، در حالی که قدرتمند است، هزینه محاسباتی الگوریتم – هم از نظر زمان و هم حافظه – اغلب یک گلوگاه در کاربردهای عملی است. در نتیجه، تحقیقات زیادی بر بهبود کارایی الگوریتم بوخبرگر متمرکز شده است، جایی که پیاده‌سازی‌های فعلی معمولاً بر روی روش‌های ابتکاری و استراتژی‌های تعریف شده توسط تخصص انسانی متکی هستند. متأسفانه، این انتخاب‌ها اغلب زیر بهینه هستند و عملکرد آن‌ها فقط بر روی کتابخانه‌های محدودی از معیارها تأیید می‌شود.

پیشرفت‌های اخیر در یادگیری ماشین فرصتی متحول کننده برای رفع این چالش‌ها ارائه می‌دهند. توانایی آموزش مدل‌ها بر روی مجموعه داده‌های گسترده، امکان کشف روش‌های ابتکاری و استراتژی‌های تصمیم‌گیری مؤثرتر را فراهم می‌کند که می‌توانند عملکرد الگوریتم را بهبود بخشند. از میان این پیشرفت‌ها، یادگیری تقویتی به

عنوان یک روش امیدوارکننده برای کاوش و تدوین استراتژی‌های جدید برجسته می‌شود که می‌تواند نتایجی فراتر از توانایی‌های طراحی سنتی الگوریتم ارائه دهد. این رویکرد شباهت‌هایی با موفقیت‌های چشمگیر مشاهده شده در بازی‌هایی مانند شطرنج دارد، جایی که سیستم‌های هوش مصنوعی استراتژی‌هایی را توسعه داده‌اند که به طور قابل توجهی از رقبای انسانی پیشی می‌گیرند.

این پروژه به دنبال ایجاد ارتباط بین دنیای کلاسیک محاسبه پایه‌های گربنر و بینش‌های مدرن یادگیری عمیق تقویتی است، با تمرکز ویژه بر یادگیری استراتژی‌های جدید انتخاب S- زوج در الگوریتم بوخبرگر. با چارچوب‌بندی این فرایند انتخاب به عنوان یک وظیفه یادگیری تقویتی، هدف ما نشان دادن نه تنها امکان‌پذیری بلکه برتری این استراتژی‌های جدید نسبت به پیاده‌سازی‌های پیشرفته در حوزه‌های مختلف است. در حالی که کارهای قبلی کاربرد تکنیک‌های یادگیری ماشین را در انتخاب الگوریتم و پارامتر در ارتباط با پایه‌های گربنر بررسی کرده‌اند، تمرکز هدفمند بر استراتژی‌های انتخاب S- زوج و تأثیر آن‌ها بر الگوریتم بوخبرگر تا حد زیادی مورد بررسی قرار نگرفته است.

این پایان نامه به چهار فصل سازماندهی شده است، فصل‌های دوم و سوم دانش بنیادی در یادگیری تقویتی و پایه‌های گربنر را ارائه می‌دهند که به عنوان زمینه برای بحث‌های بیشتر عمل می‌کنند [۵، ۷] . سهم اصلی این پروژه در فصل چهار ارائه می‌شود، جایی که ما چالش انتخاب S- زوج را به عنوان یک مسئله یادگیری تقویتی مشخص می‌کنیم و کارایی روش‌های پیشنهادی خود را نشان می‌دهیم. [۴] به طور خلاصه، این کار تلاش می‌کند تا با ادغام الگوهای نوآورانه یادگیری ماشین، به گفتگوی جاری در مورد پایه‌های گربنر و جبر محاسباتی کمک کند و در نهایت هدف آن پیشبرد هم نظریه و هم عمل در این حوزه مطالعاتی است.

فصل ۲

مبانی هوش مصنوعی

۱.۲ مقدمه

هوش مصنوعی در سال‌های اخیر پیشرفت‌های چشمگیری داشته و به یکی از مهم‌ترین حوزه‌های تحقیقاتی و فناوری تبدیل شده است. هوش مصنوعی به کامپیوترها اجازه می‌دهد تا وظایفی را انجام دهند که قبلاً تصور می‌شد فقط انسان‌ها می‌توانند از پس آن‌ها بربیایند، مانند یادگیری، حل مسئله، تصمیم‌گیری و درک زبان طبیعی. همچنین هوش مصنوعی می‌تواند در حلقه‌های چندجمله‌ای برای وظایفی مانند پیش‌بینی زمان اجرا و حافظه مورد نیاز الگوریتم بوخبرگر و بهینه‌سازی الگوریتم بوخبرگر با انتخاب پارامترها و طراحی الگوریتم‌های هیبریدی کارآمد باشد.[۱، ۳، ۲]

۲.۲ انواع یادگیری

یادگیری ماشین شاخه‌ای از هوش مصنوعی است که به کامپیوتر اجازه می‌دهد بدون برنامه‌ریزی صریح از داده‌ها یاد بگیرد. در ادامه انواع یادگیری ماشین معرفی می‌شوند.

۱.۲.۲ یادگیری نظارت شده

یادگیری نظارت‌شده یکی از زیرشاخه‌های اصلی یادگیری ماشین است که در آن، الگوریتم‌ها از طریق مجموعه‌ای از داده‌های برچسب‌گذاری شده آموزش داده می‌شوند. این داده‌ها شامل نمونه‌هایی از ورودی و خروجی مورد نظر هستند. الگوریتم یادگیری با تجزیه و تحلیل این داده‌ها، الگویی را برای پیش‌بینی خروجی برای ورودی‌های جدید یاد می‌گیرد.

اولین قدم در یادگیری نظارت‌شده، جمع‌آوری مجموعه‌ای از داده‌ها است که شامل نمونه‌هایی از ورودی و خروجی مورد نظر باشد. این داده‌ها باید به طور دقیق برچسب‌گذاری شده باشند، به این معنی که باید مشخص باشد که هر ورودی مربوط به چه خروجی است. در ادامه باید الگورتیم مناسب را انتخاب کنیم.الگوریتم‌های مختلفی برای یادگیری نظارت‌شده وجود دارد. انتخاب الگوریتم مناسب به نوع داده‌ها و وظیفه‌ای که می‌خواهید الگوریتم انجام دهد بستگی دارد. الگوریتم انتخاب شده با استفاده از داده‌های آموزشی آموزش داده می‌شود. در طول این فرآیند، الگوریتم الگویی را در داده‌ها پیدا می‌کند که می‌تواند برای پیش‌بینی خروجی برای ورودی‌های جدید استفاده شود. پس از آموزش الگوریتم، عملکرد آن با استفاده از داده‌های آزمایشی ارزیابی می‌شود. این کار برای اطمینان از اینکه الگوریتم می‌تواند به طور دقیق خروجی را برای ورودی‌های جدید پیش‌بینی کند، انجام می‌شود.

الگوریتم‌های مختلفی برای یادگیری نظارت‌شده وجود دارد، از جمله الگوریتم‌های طبقه بندی، الگوریتم‌های رگرسیون، الگوریتم‌های پیشبینی و غیره. یادگیری نظارت شده همچنین طیف وسیعی از کاربرد را در موضوعات مختلف دارد ازجمله آنها میتوان به تشخیص بیماری در پزشکی، تشخیص تقلب (برای شناسایی معاملات)، بازاریابی (برای پیشبینی رفتار مشتری)، رباتیک و غیره اشاره کرد.

مثال ۱.۲.۲. می‌توان از یادگیری نظارت شده برای آموزش یک الگوریتم برای طبقه بندی تصاویر بافت‌ها به عنوان سرطانی یا غیر سرطانی استفاده کنید. برای این کار، به مجموعه ای ازبافت‌های تصویربرداری شده با

برچسب‌های صحیح (سرطانی یا غیرسرطانی) نیاز دارید. الگوریتم یادگیری نظارت شده با تجزیه و تحلیل این داده‌ها، الگویی را برای تشخیص تصاویر جدید به عنوان سرطانی یا غیرسرطانی یاد می‌گیرد.

از معایب این یادگیری می‌توان به نیاز به داده‌های برچسب‌گذاری شده اشاره کرد. همانطور که گفته شد آموزش الگوریتم‌های یادگیری نظارت شده به مجموعه ای از داده‌های برچسب‌گذاری شده نیاز دارند که جمع‌آوری و برچسب‌گذاری‌آن‌ها می‌تواند پر هزینه و زمان‌بر باشد.

۲.۲.۲ یادگیری بدون نظارت

در این روش، الگوریتم‌ها به طور خودکارو بدون نیاز به داده‌های برچسب‌گذاری شده از داده‌ها یاد می‌گیرند و می‌توانند الگوها و ساختارهای پنهان در داده‌ها را کشف می‌کنند.

اولین قدم در یادگیری بدون نظارت، جمع‌آوری مجموعه‌ای از داده‌ها است. این داده‌ها می‌توانند شامل تصاویر، متن، صدا یا هر نوع داده دیگری باشند. قبل از اینکه الگوریتم‌های یادگیری بدون نظارت را روی داده‌ها اجرا کنید، باید داده‌ها را پیش پردازش کنید. این کار شامل تمیز کردن داده‌ها، حذف مقادیر گمشده و رمزگذاری داده‌های دسته‌ای است. حال الگوریتم مناسب را انتخاب میکنیم که انتخاب الگوریتم مناسب به نوع داده‌ها و وظیفه‌ای که می‌خواهید الگوریتم انجام دهد بستگی دارد.الگوریتم انتخاب شده با استفاده از داده‌ها آموزش داده می‌شود. در طول این فرآیند، الگوریتم الگوها و ساختارهای پنهان در داده‌ها را کشف می‌کند. پس از آموزش الگوریتم، عملکرد آن با استفاده از معیارهای مختلف ارزیابی می‌شود.

الگوریتم‌های مختلفی برای یادگیری نظارت‌شده وجود دارد، از جمله الگوریتم‌های خوشه بندی، الگوریتم‌های کاهش بعد، الگوریتم‌های کشف وابستگی و غیره. از کاربردهای این یادگیری میتوان به تجزیه و تحلیل داده‌ها (کشف الگوها و روندهای پنهان در داده‌ها)،بازاریابی (شناسایی فرصت‌های جدید بازاریابی و دسته‌بندی مشتریان)، کشف داروهای جدید در پزشکی و غیره اشاره کرد.

۳.۲.۲ یادگیری تقویتی

یادگیری تقویتی نوعی تکنیک یادگیری ماشین است که به نحوه عملکرد یک عامل هوشمند در محیطی پویا برای به حداکثر رساندن پاداش تجمعی می‌پردازد. فرایند یادگیری تقویتی به این صورت است که عامل یا هوش مصنوعی ما در یک وضعیت خاص در محیط شروع می‌شود و بر اساس وضعیت درک شده و سیاست خود، یک عمل

را انتخاب می‌کند و آن را در محیط اجرا می‌کند. محیط بر اساس عمل اجرا شده، پاداش یا جریمه ای به عامل می‌دهد و عامل از تجربه (حالت، عمل، پاداش یا جریمه) یاد می‌گیرد و سیاست خود را بر مبنای آن تنظیم می‌کند.

برای فهم ساده‌تر فرایند یادگیری تقویتی فرض کنید سگ نقش عامل را داشته باشد و عمل حرکاتی هستند که سگ میتواند انجام دهد (دویدن، پریدن و پارس کردن).حالت، موقعیت فعلی سگ در محیط است (در کجا قرار دارد یا مثلا توپ را می‌بیند یا نه) مفاهیم پاداش و جریمه در این مثال مشخص هستند. سیاست نیز استراتژی سگ برای انتخاب حرکات بعدی خود است. در این میان سگ با انجام دادن عمل مناسب (آوردن توپ برای شما) و گرفتن تشویق این کار را یاد میگیرد و درصورت قرارگرفتن در حالت مشابه احتمال اینکه این کار را تکرار کند بالا می‌رود.

یادگیری تقویتی در دو مورد از یادگیری نظارت شده و بدون نظارت متمایز است. اول درعدم نیاز به داده‌های برچسب‌گذاری شده. برخلاف یادگیری نظارت شده، یادگیری تقویتی به داده‌های پیش‌برچسب‌گذاری شده با ورودی‌ها و خروجی‌های مطلوب خاص نیاز ندارد. عامل از طریق آزمون و خطا، تعامل با محیط و دریافت پاداش (مثبت) یا جریمه (منفی) برای اعمال خود یاد می‌گیرد. دوم اینکه هدف‌محور است و مشابه یادگیری بدون نظارت، "پاسخ صحیح" از پیش تعریف شده‌ای ندارد. با این حال، یادگیری تقویتی بر خلاف یادگیری بدون نظارت که بر یافتن الگوها در داده‌ها تمرکز دارد، هدف روشنی دارد که آن به حداکثر رساندن پاداش تجمعی در طول زمان است.

همانطور که اشاره شد یادگیری تقویتی حاصل تعامل ۳ رکن اصلی محیط‌ها و عامل و بازخورد است پس برای درک بهتر، هرکدام را جداگانه مورد بررسی قرار خواهیم داد.

۱.۳.۲.۲ محیط‌ها

محیط‌ها دنیایی را که با آن تعامل می‌کنیم یا بازی ای که انجام می‌دهیم را نشان می‌دهند. محیط‌ها به عنوان فرایندهای تصمیم‌گیری مارکوف مدل سازی می‌شوند.

تعریف ۱.۲.۲. فرایند تصمیم گیری مارکوف مجموعه ای از حالات S و اعمال A با دینامیک انتقال به صورت زیر است:

$$p : \mathcal{S} \times \mathbb{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [۰, ۱]$$

$$p(\acute{s}, r|s, a) = Pr[S_{t+۱} = \acute{s}, R_{t+۱} = r|S_t = s, A_t = a]$$

و توزیع اولیه:

$$\rho : \mathcal{S} \longrightarrow [۰, ۱]$$

$$\rho(s) = Pr[S_۰ = s]$$

همانطور که از نام آن پیداست، فرایندهای تصمیم‌گیری مارکوف بسیار شبیه به زنجیره‌های مارکوف هستند. هر دو شامل مجموعه‌ای از حالات و احتمالات گذار هستند، اما فرایندهای تصمیم گیری مارکوف، همچنین شامل اعمال و پاداش نیز می‌شوند. به طور خاص، هر حالت در یک زنجیره مارکوف دارای توزیع احتمالی روی حالات بعدی است، در حالی که در یک فرایند تصمیم‌گیری مارکوف، هر حالت دارای انتخابی از اعمال است که هر کدام توزیع احتمالی خاص خود را روی حالات بعدی دارند. علاوه بر این، هر انتقال حالت در یک فرایند تصمیم‌گیری مارکوف شامل یک پاداش عددی است. حالت جاری و عمل انتخاب شده به طور کامل دینامیک انتقال را تعیین می‌کنند، که خاصیت مارکوف است که نام فرایند تصمیم‌گیری مارکوف را به آنها داده است.

در عمل، یک محیط یک فرایند تصمیم گیری مارکوف را با محاسبه $p(۰, ۰|s, a)$برای حالت جاری s و عمل a که توسط عامل ارائه شده است و سپس نمونه گیری از توزیع حاصل برای بازگرداندن یک حالت جدید $s_۰$و پاداش r پیاده سازی می‌کند. به طور مشابه، حالت اولیه محیط با نمونه گیری از $\rho(۰)$به دست می‌آید.

مثال ۲.۲.۲. تصور کنید یک بازی ساده داریم که در آن باید یک میله را روی یک چرخ دستی به صورت ایستاده نگه دارید. چرخ دستی فقط می‌تواند به چپ یا راست حرکت کند و موقعیت آن و سرعتش جزئی از بازی است. همچنین وضعیت میله و سرعت حرکت آن هم مهم است. در این بازی شما می‌توانید چرخ دستی را به چپ یا راست هل دهید. هر بار که میله به صورت ایستاده (با زاویه مشخصی نسبت به حالت عمودی) قرار دارد، امتیاز ۱ دریافت می‌کنید. بازی زمانی تمام می‌شود که میله دیگر به صورت ایستاده نباشد.

این مثال یک مسئله استاندارد برای یادگیری تقویتی است. در یادگیری تقویتی، یک عامل (مثلا شما در این بازی) با محیط (بازی ما) که شامل قوانین و شرایط است، تعامل می‌کند تا یاد بگیرد بهترین تصمیم را برای رسیدن به هدف (ایستاده نگه داشتن میله) بگیرد.

مثال ۳.۲.۲. بازی شطرنج، محیط یک فرایند تصمیم گیری مارکوف است. حالت‌های بازی تمام چینش‌های معتبر مهره‌ها روی صفحه هستند. عمل‌های ممکن، حرکات مجاز هستند که با توجه به وضعیت بازی تغییر می‌کنند. پاداش برای برد لحظه‌ای ۱ است و برای هر حرکت دیگری ۰ می‌باشد. در این حالت، استراتژی حریف

دینامیک انتقال را تعیین می‌کند، اما روش‌های یادگیری تقویتی برای مسائل یادگیری تقویتی چند عامله وجود دارد که شطرنج را به عنوان بازی بین دو عامل نشان می‌دهد.

۲.۳.۲.۲ عامل

در یادگیری تقویتی، عامل نماینده بازیگر یا بازیکن بازی است. این عامل به صورت توابع سیاست مدل سازی می‌شود.

تعریف ۲.۲.۲. با در نظر گرفتن فرایند تصمیم‌گیری مارکوف یک تابع سیاست مانندπ به صورت زیر است

$$\pi : \mathcal{A}.\mathcal{S} \to [۰, ۱]$$

$$\pi(a|s) = Pr[A_t = a|S_t = s]$$

در این تعریف، سیاست‌ها (رویکردهای تصمیم‌گیری) تصادفی هستند. این مفهوم کلی‌تر است، زیرا هر سیاست قطعی (که همیشه یک عمل را انتخاب می‌کند)، یک سیاست تصادفی است که به تمام اعمال انتخاب‌نشده، احتمال صفر اختصاص می‌دهد. علاوه بر این، گاهی اوقات سیاست بهینه تصادفی است.

۳.۳.۲.۲ بازخورد

هدف عامل این است که در درازمدت، تاحد امکان پاداش بیشتری از محیط جمع‌آوری کند. این موضوع به عنوان بازخورد در طول مسیرها نمایش داده می‌شود.

تعریف ۳.۲.۲. مسیر τ را دنباله‌ای از حالات، اعمال و پاداش‌ها گوییم که با دنبال کردن سیاست π در فرایند تصمیم‌گیری مارکوف به دست می‌آید.این دنباله با

$$\tau = (S_۰, A_۰, R_۱, S_۱, A_۱, R_۲, S_۲, \ldots, S_T)$$

نشان داده می‌شود که در آن $S_۰$ اولین حالت در دنباله، $A_۰$ اولین عمل انجام شده و $R_۱$ اولین پاداش دریافتی بعد از انجام اولین عمل است. و همینطور تا انتهای دنباله ادامه دارد.

تعریف ۴.۲.۲. به مجموع پاداش‌هایی که در مسیر τ دریافت می‌شود، بازخورد $R(\tau)$ گوییم. و آن را با فرمول

$$R(\tau) = \sum_{t=۱}^{T} R_t$$

محاسبه می‌کنیم.

در مسئله یادگیری‌تقویتی، هدف ما پیدا کردن یک سیاست π است به طوری که میانگین بازخورد مسیرهایی که بر اساس این سیاست طی می‌شوند به حداکثر برسد. به عبارتی دیگر، می‌خواهیم سیاستی را پیدا کنیم که به طور متوسط در طول زمان، بیشترین پاداش ممکن را برای عامل به ارمغان بیاورد.

به طور کلی دو نوع مسئله یادگیری‌تقویتی داریم. اگر دینامیک انتقال محیط را به طور کامل بدانیم، با مسئله برنامه‌ریزی مواجه هستیم. که در این حالت می‌توانیم با محاسبه دقیق بازخورد مسیرهای مختلف، بهترین سیاست را انتخاب کنیم. اما در نوع دوم، دینامیک انتقال محیط ناشناخته است و یا محاسبه آن غیرعملی است. در این حالت مسئله یادگیری کامل مطرح می‌شود. برای حل این مسئله، تنها راه این است که عامل صرفا از طریق تعامل با محیط، سیاست‌های خوبی را بیاموزد. بنابراین، اگرچه در حالت ایده‌آل ممکن است دینامیک انتقال محیط را بدانیم (مسئله برنامه‌ریزی)، اما در اغلب موارد با مسئله یادگیری کامل مواجه هستیم که در آن عامل باید صرفا از طریق تعامل با محیط یاد بگیرد.

مثال ۴.۲.۲. برای مثال به سه مسیر شبیه سازی‌شده زیر توجه کنید.

$$\tau_۱ = (s_۱, a_۱, ۲, s_۱, a_۱, ۰, s_۲, a_۲, ۵, s_۳)$$

$$\tau_۲ = (s_۱, a_۱, ۰, s_۲, a_۱, ۰, s_۱, a_۱, ۰, -۵, s_۳)$$

$$\tau_۳(s_۱, a_۱, ۲, s_۱, a_۱, ۰, s_۲, a_۱, -۵, s_۳)$$

که بازخوردهای‌آنها به ترتیب $R(\tau_۱) = ۷, R(\tau_۲) = -۵, R(\tau_۳) = -۳$ است. همانطور که مشاهده می‌شود، هر مسیر شامل دنباله‌ای از حالات، اعمال و پاداش‌ها است. بازده هر مسیر نیز با جمع پاداش‌ها در آن مسیر محاسبه شده است.

مسیر ۱ یک مسیر مطلوب است، زیرا بازخورد آن ۷ است که نشان‌دهنده دریافت پاداش مثبت در طول مسیر

است. مسیر ۲ یک مسیر نامطلوب است، زیرا بازخورد آن −۵ است که نشان‌دهنده دریافت پاداش منفی در طول مسیر است. مسیر ۳ یک مسیر متوسط است، زیرا بازخورد آن −۳ است که نشان‌دهنده دریافت پاداش‌های مثبت و منفی در طول مسیر است.

در بسیاری از موارد، در نظرگرفتن پاداش‌های آینده به اندازه پاداش‌های حال حاضر ارزشمند نیستند. این موضوع معمولا با استفاده از یک شاخص تخفیف بیان می‌شود. به عبارت دیگر این شاخص می‌تواند اثرگذاری گذارهای آینده را بر روی پاداش مورد انتظار ما پس از انجام یک عمل، کاهش دهد.

تعریف ۵.۲.۲. بازده تنزیل شده $R_\gamma(\tau)$، مجموع پاداش‌هایی است که در طول مسیر τ به دست می‌آید، اما با این تفاوت که پاداش‌های آینده با در نظر گرفتن عامل تخفیف، ارزش کمتری نسبت به پاداش‌های حال حاضر پیدا می‌کنند.

$$R_\gamma(\tau) = \sum_{t=1}^{T} \gamma^{t-1} R_t$$

. که در آن γ شاخص تخفیف است و $0 \leq \gamma \leq 1$.

۳.۲ یادگیریQ

یادگیری تقویتی مدرن اغلب با یادگیری تقویتی عمیق هم‌معنی در نظر گرفته‌می‌شوند. یادگیری تقویتی عمیق به استفاده از شبکه‌های تقویتی عمیق به عنوان ابزاری برای تقریب زدن توابع در یادگیری تقویتی اشاره دارد. با وجود موفقیت‌های اخیر در یادگیری عمیق، یادگیری تقویتی سابقه‌ای طولانی دارد. اگوریتم‌ها و تکنیک‌های متعددی وجود دارند که در ابتدا خارج از این چارچوب (یادگیری عمیق) توسعه یافته‌اند. یادگیریQ به عنوان یک الگوریتم یادگیری تقویتی که ابتدا در سال ۱۹۹۷ معرفی شد، به عنوان یک مثال انگیزشی ذکر شده‌است. در ابتدا باید توجه شود که برخی حالات در یک محیط (مانند یک بازی) نسبت به حالات دیگر مطلوب‌تر هستند. برای مثال، موقعیت‌های خاصی در صفحه شطرنج، شانس بیشتری برای برد به ما می‌دهند. این مفهوم ترجیح بین حالات را می‌توان با دقت بیشتری با استفاده از تابع ارزش تعریف کرد.

تعریف ۱.۳.۲. فرض کنید یک فرایند تصمیم‌گیری مارکوف، یک عامل تخفیف γ و سیاست π را داشته باشیم

تابع ارزش-حالت $v_{\pi,\gamma}$ به صورت زیر تعریف می‌شود.

$$v_{\pi,\gamma}(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{T} \gamma^{t-1} R_t | S_0 = s]$$

تعریف ۲.۳.۲. فرض کنید یک فرایند تصمیم‌گیری مارکوفی و یک عامل تخفیف داشته باشیم. تابع ارزش-حالت بهینه، نمایش داده شده توسط $v_{\pi,\gamma}$، برابر با حداکثر تابع ارزش-حالت برای همه سیاست‌های ممکن در حالت s است.

$$v_{*,\gamma}(s) = \max v_{\pi,\gamma}(s)$$

به عبارت دیگر، این مقدار، بهترین بازده تنزیل‌شده مورد انتظار ما را در صورتی که از حالت s شروع کرده و بهترین سیاست را دنبال کنیم، نشان می‌دهد. پس بهترین سیاست سیاستی است که بالاترین بازده تنزیل‌شده مورد انتظار را به همراه دارد.

قضیه ۱.۳.۲. فرض کنید یک فرایند تصمیم‌گیری مارکوف و یک عامل تخفیف γ داشته باشیم. آنگاه معادلات زیر نتیجه می‌شوند:

$$v_{\pi,\gamma}(s) = \sum_{s',r,a} \pi(a|s) p(s', r|s, a)(r + \gamma v_{\pi,\gamma}(s'))$$

$$v_{*,\gamma}(s) = \max_a \sum_{s',r} p(s', r|s, a)(r + \gamma v_{*,\gamma}(s'))$$

معادلات در قضیه پیشین به ما این امکان را می‌دهند که برای هر حالت s یک سیستم معادله با متغیرهای $v_{*,\gamma}$ یا $v_{\pi,\gamma}$ بنویسیم. اگر بتوانیم نسخه‌ی مربوط به $v_{*,\gamma}$ را حل کنیم، آنگاه می‌توانیم سیاست بهینه را با معادله زیر محاسبه کنیم.

$$\pi_*(s) = \arg\max \sum_{s',r} p(s', r|s, a)(r + \gamma v_*(s'))$$

این روش، مسئله فرایند تصمیم‌گیری مارکوف را حل می‌کند. با این حال، این روش دو مشکل اساسی دارد. اول اینکه سیستم معادلات ممکن است بزرگ و پیچیده باشد و یافتن راه‌حل‌های دقیق غیرقابل اجرا بشود. و دوم

اینکه، درمسئله یادگیری کلی، ما دینامیک انتقال (p) محیط را در اختیار نداریم و بنابراین نمی‌توانیم این معادلات را بنویسیم. درواقع، حتی اگر تابع v_* را در اختیار داشته‌باشیم، برای استفاده از این معادله و تعیین سیاست، همچنان به شناخت دینامیک انتقال نیاز داریم. بنابراین، هنگامی که دینامیک انتقال ناشناخته یا محاسبه‌ی آن غیرقابل‌اجرا باشد، نمی‌توان از توابع ارزش-حالت برای حل مسئله استفاده کرد. به جای آن، باید توابع ارزش-عمل (تابع Q) را در نظر بگیریم. تابع ارزش-عمل ،(تابعQ) ارزش مورد انتظار انجام یک عمل خاص (a) در یک حالت خاص (s)را نشان می‌دهد. با در نظر گرفتن تمام حالات بعدی ممکن $(s_۰)$ و پاداش‌های مرتبط، این تابع به ما می‌گوید که کدام عمل در هر حالت، بازده بلندمدت بهتری (بازده تنزیل‌شده) به همراه خواهد داشت.

تعریف ۲.۳.۳. فرض کنید یک فرایند تصمیم‌گیری مارکوف، یک عامل تخفیف γ و سیاست π را داشته باشیم تابع ارزش-عمل $q_{\pi,\gamma}(s,a)$ به صورت زیر تعریف می‌شود.

$$q_{\pi,\gamma}(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=۱}^{T} \gamma^{t-۱} R_t | S_۰ = s, A_۰ = a]$$

به عبارت دیگر، این تابع، بازده تنزیل‌شده مورد انتظار ما را در صورتی که از حالت s شروع کرده، عمل a را انجام دهیم و سپس در تمامی مراحل بعدی از سیاست π تبعیت کنیم، نشان می‌دهد.

تعریف ۲.۳.۴. فرض کنید یک فرایند تصمیم‌گیری مارکوفی و یک عامل تخفیف γ داشته باشیم. تابع ارزش-عمل بهینه، نمایش داده شده توسط $q_{\pi,\gamma}$، برابر با حداکثر تابع ارزش-عمل برای همه سیاست‌های ممکن در حالت s است.

$$q_{*,\gamma}(s,a) = \max q_{\pi,\gamma}(s,a)$$

قضیه ۲.۳.۲. توابع ارزش-عمل رابطه زیر را برآورده می‌کنند.

$$q_{\pi,\gamma}(s,a) = \sum_{s',r} p(s',r|s,a)\left(r + \gamma \sum_{a'} \pi(a'|s') q_{\pi,\gamma}(s',a')\right)$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma \max_{a'} q_{*,\gamma}(s',a'))$$

اگر تابع ارزش-عمل را داشته‌باشیم. دراین صورت، سیاست بهینه $\pi_*(s)$ با معادله زیر بدست می‌آید.

$$\pi_*(s) = \arg\max_a q_*(s,a)$$

معادله بالا به ما این امکان را می‌دهد تا بدون نیاز به دینامیک انتقال، سیاست بهینه را تعیین کنیم. دلیل این امر آن است که انتخاب عمل درون تابع q_* گنجانده شده‌است. با این حال، همچنان با این مشکل مواجه هستیم که بدون در اختیار داشتن دینامیک انتقال، نمی‌توانیم سیستم معادلات ارائه شده در قضیه ۲.۳.۲ را ایجاد کنیم. اما از آنجایی که:

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma \max_{a'} q_{*,\gamma}(s',a')) = \mathbb{E}_{s',r}[r + \gamma \max_{a'} q_*(s',a')]$$

مقدار $q_*(s,a)$ یک امید ریاضی است و با استفاده از روش‌های مونت‌کارلو قابل تقریب است. این روش شامل نمونه‌گیری از گذارهای $s,a \to s'$ از محیط با دنبال کردن هر سیاستی و محاسبه میانگین ارزش $r + \gamma \max_{a'} q_*(s',a')$ می‌شود. این ایده منجر به الگوریتم یادگیری Q می‌شود. این الگوریتم شامل نگه‌داشتن یک جدول $Q[s,a]$ برای هر جفت از حالت و عمل، برقراری تعامل با محیط برای نمونه‌گیری از گذارها و به‌روزرسانی تدریجی جدول برای تطابق بهتر با معادله اخیر است.

در صورتی که بازدید از همه حالت‌ها و اعمال همچنان ادامه داشته باشد و اندازه گام α به طور مناسب به صفر کاهش یابد، الگوریتم یادگیری Q با احتمال به تابع ارزش-عمل بهینه واقعی، q_* همگرا می‌شود. در عمل، α روی مقدار کوچکی ثابت می‌شود و اکتشاف (بررسی فضا) با دنبال کردن یک سیاست ϵ-حریصانه تضمین می‌شود. این سیاست با احتمالϵ به صورت تصادفی عمل می‌کند و در غیر این صورت، عملی را با بالاترین مقدار پیش‌بینی‌شده فعلی از $Q[s,a]$ انتخاب می‌کند.

مثال ۳.۳.۲. فرایند تصمیم‌گیری مارکوف ساده ای را فرض کنید که دارای سه حالت (که یکی از آنها انتهایی است) و دو عمل است، بنابراین جدولی به نام $Q[s,a]$ را با مقداردهی اولیه زیر ایجاد می‌کنیم:

$$Q[s,a] = \begin{bmatrix} ۰/۶۵ & ۰/۳۸ \\ ۰/۳۸ & ۰/۱۱ \\ ۰/۰۰ & ۰/۰۰ \end{bmatrix}$$

ستون‌های جدول به اعمال $a_۱$ و$a_۲$ و ردیف‌ها به حالت‌های $s_۱$و $s_۲$ و $s_۳$ اختصاص دارند. مقادیر جدول به

الگوریتم ۱.۲ Q-learning

parameters: $\epsilon \in (0,1], \alpha \in (0,1]$

initialize table $Q[s,a]$**arbitrarily,except**$Q[s,0] = 0$ **for terminal** s

loop

select start state S

while S is not terminal **do**

$A \leftarrow$ select action using ϵ-greedy with Q

$S', R \leftarrow$observe transition after taking action A

$Q[S,A] \leftarrow (1-\alpha)Q[S,A] + \alpha(R + \gamma \max_a Q[S',a])$

$S \leftarrow S'$

end while;

end loop;

جز ردیف آخر به صورت تصادفی انتخاب شده‌اند، زیرا $s_۳$ حالت انتهایی است. سپس در یک اجرای نمونه با $\epsilon = ۰/۵, \alpha = ۰/۰۵, \gamma = ۰/۹۹$ ، جدول به صورت زیر تغییر می‌کند:

$$\begin{bmatrix} ۰/۶۵ & ۰/۳۸ \\ ۰/۳۸ & ۰/۱۱ \\ ۰/۰۰ & ۰/۰۰ \end{bmatrix} \rightarrow \begin{bmatrix} ۸/۸۰ & ۷/۹۵ \\ -۱/۳۷ & ۵/۰۰ \\ ۰/۰۰ & ۰/۰۰ \end{bmatrix} \rightarrow \begin{bmatrix} ۹/۳۵ & ۷/۹۵ \\ -۰/۱۵ & ۵/۰۰ \\ ۰/۰۰ & ۰/۰۰ \end{bmatrix} \xrightarrow{Some} \begin{bmatrix} ۸/۹۷ & ۷/۹۵ \\ -۱/۶۵ & ۵/۰۰ \\ ۰/۰۰ & ۰/۰۰ \end{bmatrix}$$

به طوری که سیاست نهایی پیش‌بینی شده $\pi(s) = \arg\max_a Q[s,a]$ برابر است با:

$$\pi(a_۱|s_۱) = ۱/۰$$

$$\pi(a_۲|s_۱) = ۰/۰$$

$$\pi(a_۱|s_۲) = ۰/۰$$

$$\pi(a_۲|s_۲) = ۱/۰$$

که به صورت تصادفی استاندارد نمایش داده‌شده‌است.

$$A \xrightarrow[SomeMoreText]{SomeText} B$$

۴.۲ تابع تقریب

در مثال ۲.۳.۳، ما مسئله یادگیری تقویتی را با Q یادگیری حل کردیم. با این حال، ما نیاز به حفظ جدولی داشتیم که برای هر حالت و عمل ممکن ورودی داشته باشد و برای به روز رسانی ورودی جدول مربوطه، باید از هر حالت و عملی بازدید کنیم. این امر با افزایش تعداد حالات و اعمال به سرعت غیرقابل اجرا می‌شود که مشکل اصلی اکثر الگوریتم‌های یادگیری تقویتی کلاسیک است: آنها به فضای حالت و عمل بزرگ یا با ابعاد بالا تعمیم داده نمی‌شوند.

یک راه حل طبیعی جایگزینی جدول با یک تابع تقریب است که در مثال ۲.۳.۳ یک مدل یادگیری ماشینی آموزش داده شده برای پیش‌بینی $q_*(s,a)$ خواهد بود. این مدل انعطاف پذیری بسیار کمتری نسبت به یک جدول در تطبیق تجربه عامل خواهد داشت، اما این در واقع یک مزیت است. به عبارتی با نگاشت حالات مشابه به مقادیر مشابه، مدل می‌تواند از حالات مشابه اما نه یکسان یاد بگیرد و اطلاعات جدول را در قالبی بسیار کوچکتر فشرده کند.

استفاده از تابع تقریب مستلزم دانستن نحوه مدل سازی یک تابع و نحوه آموزش آن مدل است. در حالی که گزینه‌های دیگری نیز وجود دارد، یادگیری تقویتی عمیق مدرن بر اساس موفقیت‌های اخیر در یادگیری با نظارت با استفاده از گرادیان نزولی برای تناسب شبکه‌های عصبی ساخته شده است. این خود به خود یک زمینه بسیار گسترده است، بنابراین در اینجا فقط ایده‌ها را به اندازه کافی پوشش می‌دهیم تا مقداری شهود به دست آوریم.

۱.۴.۲ گرادیان نزولی

فرض کنید مجموعه داده‌ای به شکل $(x_1,y_1),\ldots,(x_n,y_n)$ داریم و می‌خواهیم تابع f را روی این داده‌ها برازش دهیم(fit) به گونه‌ای که $y_i \approx f(x_i)$. ما حدس می‌زنیم که داده‌ها خطی باشند، بنابراین سعی می‌کنیم مقادیری برای $\theta = \begin{bmatrix}\theta_0\\ \theta_1\end{bmatrix}$ پیدا کنیم که در آن $f_\theta(x)=\theta_1 x+\theta_0$ برازش خوبی داشته باشد. برای سنجش کیفیت تقریب خود، می‌توانیم میانگین مربعات خطا را پیش‌بینی و مقادیر واقعی را به صورت زیر محاسبه کنیم.

$$J(\theta)=\frac{1}{2n}\sum_{i=1}^{n}(y_i-f(x_i))^2=\frac{1}{2n}\sum_{i=1}^{n}(y_i-(\theta_1 x_i+\theta_0))^2.$$

با این دیدگاه، تابع خطا J تابعی از پارامترها است. می‌توانیم گرادیان تابع خطا را نسبت به پارامترها محاسبه کنیم:

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \end{bmatrix} = -\frac{1}{n} \begin{bmatrix} \sum_{i=1}^{n} (y_i - (\theta_1 x_i + \theta_0)) \\ \sum_{i=1}^{n} (y_i - (\theta_1 x_i + \theta_0)) x_i \end{bmatrix}.$$

ما می‌خواهیم تابع خطا کم باشد. با حرکت دادن پارامترها در جهت منفی گرادیان، می‌توانیم تابع خطا را کاهش دهیم. بنابراین، $\alpha > 0$ را به عنوان یک نرخ یادگیری کوچک در نظر می‌گیریم. با شروع از هر حدس اولیه θ_0 ، بارها حدس را به صورت زیر تنظیم می‌کنیم:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta J(\theta)|_{\theta_k}.$$

این روش گرادیان نزولی است که برای بهینه‌سازی هر تابع خطای افتراقی خروجی یک مدل که نسبت به پارامترهایش قابل افتراق است، استفاده می‌شود. الگوریتم‌های بهینه‌سازی مبتنی بر گرادیان پیچیده‌تری مانند RMSProp و Adam وجود دارند که برای اکثر مسائل استفاده می‌شوند.

۲.۴.۲ شبکه عصبی

طراحی شبکه‌های عصبی مصنوعی با الهام از نورون‌های بیولوژیکی صورت گرفت، اما یادگیری عمیق آنقدر از این انگیزه اولیه فاصله گرفته است که هرگونه تشابهی به سختی قابل درک است. به جای آن، یک ماتریس $N \times M$ به نام W و یک بردار $N \times 1$ به نام $\vec{b}$ را در نظر بگیرید. سپس تابع $f : \mathbb{R}^M \to \mathbb{R}^N$ را به صورت زیر تعریف می‌کنیم:

$$f(\vec{x}) : W\vec{x} + \vec{b}$$

با تغییر مقادیر داخل W و $\vec{b}$ می‌توانیم این تابع را تغییر دهیم، بنابراین این مقادیر پارامترهای مدل هستند. با داشتن داده‌های $\vec{x_i} \in \mathbb{R}^M$ و $\vec{y_i} \in \mathbb{R}^N$ ، می‌توانیم با تنظیم پارامترها با استفاده از گرادیان نزولی، مدل را برازش دهیم. البته تابع f بالا فقط یک تبدیل خطی آفین است و آنچه توصیف می‌کنیم صرفاً گرادیان نزولی برای برازش یک مدل خطی چند متغیره است.

اضافه کردن غیرخطی بودن به صورت توابع فعال‌سازی، کلید اصلی این مبحث است. این توابع را می‌توان

به هر ورودی در خروجی f اعمال کرد. توابع فعال‌سازی معمولاً با σ نشان داده می‌شوند و انتخاب‌های رایج شامل سیگموئید $\sigma(z) = \frac{۱}{۱+e^{-z}}$ ، تانژانت هذلولوی $\sigma(z) = \frac{e^z-e^{-z}}{e^z+e^{-z}}$ و واحد خطی اصلاح‌شده $\sigma(z) = \max(۰, z)$ هستند.

سپس تابع غیرخطی $g : \mathbb{R}^M \to \mathbb{R}^N$ به صورت زیر تعریف می‌شود:

$$g(\vec{x}) : \sigma(W\vec{x} + \vec{b})$$

این یک مدل غیرخطی است که تا زمانی که توابع فعال‌سازی، توابع افتراقی باشند، همچنان می‌توان آن را با گرادیان نزولی برازش داد.

علاوه بر این، ما می‌توانیم با انجام مکرر تبدیل‌های آفین و فعال‌سازی‌ها، تابعی مانند تابع h را به صورت زیر تعریف کنیم:

$$h : \mathbb{R}^M \to \mathbb{R}^N$$

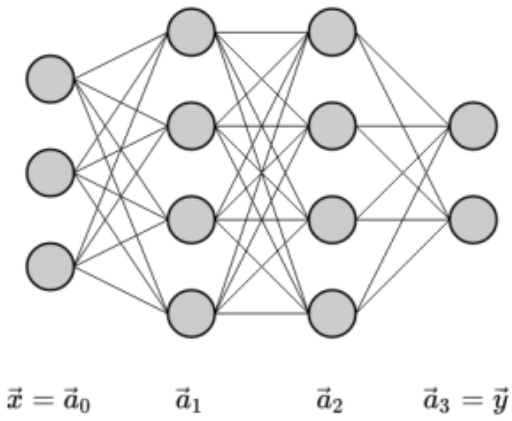
$$h(\vec{x}) : \sigma(W_n \cdots \sigma(W_۲\sigma(W_۱\vec{x} + \vec{b_۱}) + \vec{b_۲}) + \cdots + \vec{b_n}$$

.

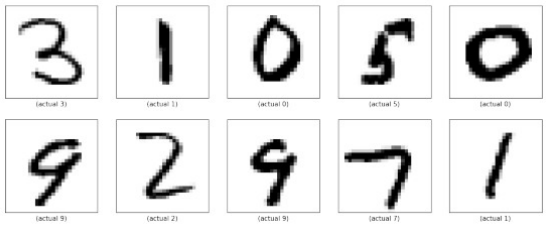
این تابع که اغلب به صورت مجموعه‌ای از لایه‌ها نشان داده می‌شود، یک شبکه عصبی متراکم پایه، که گاهی اوقات به عنوان پرسپترون چندلایه شناخته می‌شود، است. ورودی‌های شبکه $(\vec{x} = \vec{d_۰})$ از سمت چپ وارد می‌شوند و محاسبه هر لایه بعدی از چپ به راست را می‌توان با انجام تبدیل خطی آفین $\vec{d_i} = \sigma(W_i\vec{d_{i-۱}} + \vec{b_i})$ و فعال‌سازی بر روی خروجی لایه قبلی $(\vec{d_{i-۱}})$ انجام داد و به این ترتیب خروجی لایه جاری $(\vec{d_i})$ را به دست آورد.

گرادیان‌های تابع زیان را نیز می‌توان برای هر لایه به طور مؤثر از راست به چپ با استفاده از قانون زنجیره‌ای محاسبه کرد. قانون زنجیره‌ای الگوریتم مهمی است که به آن پس‌انتشار گفته می‌شود. از آنجایی که محاسبات در شبکه‌های عصبی اساسا عملیات ماتریسی هستند، هم ارزیابی پیش‌رو و هم پس‌انتشار قابلیت موازی‌سازی بالایی دارند. این قابلیت به این معنی است که گرادیان نزولی روی شبکه‌های عصبی را می‌توان با نرم‌افزار و سخت‌افزار مدرن به صورت بسیار کارآمد انجام داد.

بسیاری از انواع دیگر شبکه‌های عصبی وجود دارند، مانند شبکه‌های عصبی کانولوشنال و شبکه‌های عصبی



شکل ۱.۲: یک شبکه عصبی پایه. واحدها یا "نرون‌ها" مقادیر اسکالر (عددی) هستند که به صورت عمودی در لایه‌ها سازماندهی شده‌اند و وابستگی‌های آن‌ها با خطوط نشان داده می‌شود.



شکل ۲.۲: چند نمونه داده ،MNIST همراه با برچسب صحیح آن‌ها

بازگشتی، که از انواع مختلفی از لایه‌ها و اتصالات تشکیل شده‌اند، اما همه آنها از ایده اولیه ساخت یک تابع غیرخطی با پارامتر بالا که قابل تفاضل(به این معنی است که می‌توان از تابع در هر نقطه مشتق گرفت.) است، پیروی می‌کنند. شبکه‌های عصبی در یادگیری طیف گسترده‌ای از توابع برای کارها و کاربردهای مختلف بسیار مؤثر بوده‌اند.

مثال ۱.۴.۲. پایگاه داده MNIST مجموعه‌ای از ۷۰۰۰۰ رقم دست‌نویس است که توسط دانش‌آموزان دبیرستانی ایالات متحده و کارمندان اداره آمار نوشته شده است. این تصاویر به صورت ۲۸ در ۲۸ پیکسل در مقیاس خاکستری هستند و با رقم صحیح برچسب‌گذاری شده‌اند (شکل ۲.۲ را ببینید). MNIST به طور گسترده به عنوان یک مجموعه داده ساده در یادگیری ماشین استفاده می‌شود، جایی که هدف پیش‌بینی برچسب با توجه به تصویر رقم دست‌نویس است.

فرض کنید هر تصویر خاکستری ۲۸ در ۲۸ پیکسلی در پایگاه داده MNIST را به عنوان یک نقطه در فضای

$\mathbb{R}^{۷۸۴}$ بعدی در نظر بگیریم. هدف ما در این مثال، یادگیری تابعی به نام $f : \mathbb{R}^{۷۸۴} \to \mathbb{R}^{۱۰}$ است که این تصاویر را به اعدادی بین ۰و ۱نگاشت می‌کند. خروجی این تابع برای هر عدد خروجی (۱۰ خروجی وجود دارد، چون می‌خواهیم ۱۰ رقم را تشخیص دهیم)، نشان‌دهنده احتمال این است که تصویر مربوط به آن رقم خاص باشد. (شبیه توابع سیاست‌گذاری، خروجی به صورت توزیع احتمال است تا یک مقدار واحد، چرا که این کار باعث تفاضل‌پذیری بهتر تابع می‌شود).

می‌توانیم یک شبکه عصبی ساده با یک لایه پنهان میانی و تابع فعال‌سازی خطی اصلاح‌شده را با استفاده از ماتریس‌ها و بردارها نمایش دهیم. این شبکه در کل به ۵۳۰،۲۰۳ پارامتر (وزن‌ها و سوگیری‌ها) نیاز دارد.

با آموزش این مدل با استفاده از الگوریتم گرادیان نزولی روی ۶۰۰۰۰ تصویر از مجموعه داده MNIST تنها برای چند دقیقه روی یک لپ‌تاپ معمولی، به تابعی دست پیدا می‌کنیم که می‌تواند ۹۷/۶درصد از ۱۰۰۰۰ تصویر باقی‌مانده (که در طول آموزش ندیده بود) را به درستی تشخیص دهد. شبکه‌های پیچیده‌تر می‌توانند به دقت بالاتری دست پیدا کنند و در واقع، تشخیص ساده‌ی ارقام با ابزارهای یادگیری عمیق به یک مسئله حل‌شده تبدیل شده است.

۳.۴.۲ بهینه‌سازی سیاست تقریبی(PPO)

اندازه گام γ در الگوریتم گرادیان سیاست (یا در عمل، بهینه‌ساز نزول گرادیان استفاده‌شده در طول آموزش) کوچک است زیرا ما فقط در حال تقریب زدن گرادیان در مقدار فعلی پارامترهای θ هستیم و گرادیان ممکن است با دور شدن از آن نقطه تغییر کند. به عبارت دیگر، γ کوچک است تا حرکت ما را در فضای پارامتر در هر بار تنظیم سیاست محدود کنیم. با این حال، حرکات کوچک در فضای پارامتر لزوماً مربوط به حرکات کوچک در فضای سیاست نیست. در واقع، یک مشکل کلیدی با گرادیان سیاست پایه این است که بدون تنظیم بسیار دقیق اندازه گام‌ها، الگوریتم ممکن است حرکات فاجعه‌بار بزرگی در فضای سیاست انجام دهد و تکنیک‌های مختلفی را برای محدود کردن یا تغییر پویای اندازه گام‌ها پیاده کند. به یاد بیاورید که گام گرادیان سیاست پایه را می‌توان به عنوان یک گام گرادیان نزولی برای کمینه کردن تابع زیان جایگزین دید

$$L^{PG}(\theta) = \mathbb{E}_{\tau\sim\pi}[(\log \pi_\theta(A_t|S_t)\Phi_t)]$$

که مقدار منفی مربوط به دیدگاه کمینه کردن یک زیان به جای بیشینه کردن عملکرد است. یک روش برای

محدود کردن تغییرات در سیاست با تابع زیان جایگزین مرتبط زیر شروع می‌شود.

$$L^{CPI}(\theta) = \mathbb{E}[\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)}\Phi_t]$$

از تکرار سیاست محافظه‌کارانه. برای درک این زیان، توجه کنید که کمینه کردن $L^{CPI}(\theta)$ شامل تنظیم نسبت احتمال‌های سیاست جدید $\pi_\theta(A_t|S_t)$ به احتمال‌های سیاست قدیمی $\pi_{\theta_k}(A_t|S_t)$ است. هنگامی که t مثبت است (به این معنی که عمل خوب بوده است)، می‌توانیم با بزرگ‌تر کردن این نسبت از ۱، یعنی با احتمال بیشتر برای انتخاب مجدد عمل، زیان را کاهش دهیم. هنگامی که ϕ_t منفی است، می‌توانیم به طور مشابه با کوچک‌تر کردن نسبت، یعنی با کاهش احتمال انتخاب مجدد عمل، زیان را کاهش دهیم. ما هنوز محدودیتی برای سیاست نداریم، اما حضور نسبت سیاست جدید به قدیمی در این زیان یک اصلاح ساده را پیشنهاد می‌کند: قرار دادن یک محدودیت بر روی اینکه گام بهینه‌سازی چقدر در حرکت دادن این نسبت پیش می‌رود. با برش زدن مقدار نسبت، به یک زیان می‌رسیم

$$L^{PPO}(\theta) = \mathbb{E}[\min(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)}, clip(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_k}(A_t|S_t)}, ۱-\epsilon, ۱+\epsilon))\phi - t]$$

که فقط با بهبود نسبت در محدوده کوچک $(۱-\epsilon, ۱+\epsilon)$ کوچک‌تر می‌شود. خارج از این محدوده، زیان کاهش نمی‌یابد (اگرچه می‌تواند افزایش یابد)، بنابراین بهینه‌ساز استفاده‌شده برای انجام برازش‌ها برای حرکت دادن سیاست جدید خیلی دور از سیاست قدیمی پاداش نمی‌گیرد. این منجر به تغییرات بسیار تدریجی‌تر در سیاست و وابستگی بسیار کمتر به انتخاب اندازه گام می‌شود. کمینه کردن مقدار L^{PPO} الگوریتم بهینه‌سازی سیاست تقریبی (PPO) را ارائه می‌دهد که در [۸۳] معرفی شده است (به‌ویژه، این شکل برش‌خورده PPO است). PPO یک الگوریتم استاندارد در یادگیری تقویتی عمیق است، اغلب انتخاب پیش‌فرض برای یک روش گرادیان سیاست است و انتخاب ما برای تمام نتایج فصل ۴ خواهد بود.

فصل ۳

پایه گربنر و الگوریتم بوخبرگر

۱.۳ مقدمه

پایه گربنر، ابزاری قدرتمند در جبر محاسباتی است که برای مطالعه ایده‌آل‌ها و حلقه‌های چندجمله‌ای به کار می‌رود. این مفهوم توسط بوخبرگر در رساله دکترای خود تحت راهنمایی گربنر معرفی شد . بوخبرگر در سال ۱۹۶۵، پایه گربنر را به عنوان ابزاری برای نمایش فضای خارج‌قسمتی متناظر با یک ایده‌آل صفر بعدی به عنوان یک فضای برداری معرفی کرد . او همچنین الگوریتم بوخبرگر را با پیچیدگی زمانی $d^{2^{O(n)}}$ را برای محاسبه پایه گربنر که در آن d بزرگترین درجه چندجمله‌ای وردی و n تعداد متغیر است، را معرفی کرد. در سال ۱۹۸۳، لازارد با استفاده از تکنیک‌های جبر خطی، الگوریتمی با پیچیدگی زمانی کمتری برای محاسبه پایه گربنر ارائه کرد. این الگوریتم نقطه عطفی در روند محاسبه پایه گربنر بود و منجر به طراحی الگوریتم های جدیدی شد که به دلیل استفاده از ابزارهای جبر خطی، پیچیدگی کمتری داشت و سرعت آن‌ها به طور قابل توجهی بالا تر بود. در سال ۱۹۹۲، مولر، مورا و تراورسو با استفاده از تکنیک "کاهش به صفر" در الگوریتم بوخبرگر، سرعت آن را تا حدی بهبود بخشیدند. فوژر در سال‌های ۱۹۹۹ و ۲۰۰۲، دو الگوریتم F_4 و F_5 را بر اساس ایده‌های قبلی ارائه کرد. الگوریتم F_5 تا کنون سریع‌ترین الگوریتم برای محاسبه پایه گربنر شناخته می‌شود.[۷]

۲.۳ حلقه چندجمله‌ای‌های تک‌متغیره

در این بخش به چند تعریف و قضیه در ارتباط با حلقه چندجمله‌ای‌های تک‌متغیره می‌پردازیم.

تعریف ۱.۲.۳. یک ترکیب خطی متناهی به صورت $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ را که درآن $a_n, a_{n-1}, \ldots, a_0 \in K$ یک چندجمله‌ای تک‌متغیره بر حسب x می‌نامیم. حلقه همه چندجمله‌ای‌های بر حسب x با ضرایب در K را با $K[x]$ نمایش می‌دهیم.

می‌توان دید که $K[x]$ یک فضای برداری بر روی K با پایه $\{1, x, x^2, \ldots\}$ است.

تعریف ۲.۲.۳. اگر ضریب هر جمله از $f(x)$ صفر باشد، آنگاه $f(x)$ را یک چندجمله‌ای صفر می‌نامیم.

تعریف ۳.۲.۳. اگر فرض کنیم $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ یک چندجمله‌ای در $K[x]$ باشد آنگاه :

۱) در صورتی که $a_i \neq 0$ درجه $f(x)$ همان ماکزیممi در آن است که با$\deg(f)$ نشان‌داده می‌شود.

۲) درجه چندجمله‌ای صفر را با $-\infty$ نمایش می‌دهند.

تعریف ۴.۲.۳. اگر فرض کنیم چندجمله‌ای $f(x)$ چندجمله‌ای دلخواه در$K[x]$ باشد، آنگاه:

۱) $x^{\deg(f)}$ را تک‌جمله‌ای پیشروی $f(x)$ می‌نامیم و آن را با $LM(f)$ نمایش می‌دهیم.

۲) $a_{\deg(f)}$ را ضریب پیشروی $f(x)$ می‌نامیم وآن را با $LC(f)$ نمایش می‌دهیم.

۳)$LM(f).LC(f)$را جمله پیشروی $f(x)$ می‌نامیم و آن را با $LT(f)$ نمایش می‌دهیم.

اگر فرض کنیم f, g دو چندجمله‌ای غیر صفر باشند آنگاه $\deg(f) \leqslant \deg(g)$ اگر و تنها اگر $LT(f)|LT(g)$.

قضیه ۱.۲.۳. فرض کنیم K یک میدان باشد و $f, g \in K[x]$ باشند و $g \neq 0$. در این صورت چندجمله‌ای‌های یکتای r, q وجود دارند به طوری که $f = qg + r$ که در آن $\deg(r) < \deg(f)$ یا $r = 0$ است.

تعریف ۵.۲.۳. فرض کنیم$I \subseteq R$ و R حلقه جابه‌جایی باشد. I را ایده‌آل R گوییم هرگاه: $0 \in I$ و در صورتی که $a, b \in I$ و$r \in R$ آنگاه $a + b \in I$ و $ra \in I$.

تعریف ۶.۲.۳. فرض کنیم R یک حلقه جابه‌جایی باشد و$a \in R$. $\langle a \rangle := \{ra | a \in R\}$ را ایده‌آل تولید شده توسط a تعریف می‌کنیم.

هرگاه به ازای یک $a \in R$، $I = \langle a \rangle$ آنگاه I ایده‌آل اصلی حلقه‌ی R است.

تعریف ۷.۲.۳. هرگاه حاصل ضرب دو عضو ناصفر R، مخالف صفر باشد آنگاه حلقه جابجایی R را دامنه صحیح گوییم. دامنه صحیح R را که هر ایده‌آل آن اصلی است دامنه ایده‌آل اصلی گوییم.

قضیه ۲.۲.۳. اگر K یک میدان باشد، آنگاه هر ایده‌آل از $K[x]$ یک ایده‌آل اصلی است.

۳.۳ حلقه چندجمله‌ای‌های چندمتغیره

تعریف ۱.۳.۳. یک تک‌جمله‌ای چندمتغیره برحسب $x_1, x_2, \ldots, x_n$ به صورت ${x_1}^{a_1}{x_2}^{a_2}\ldots{x_n}^{a_n}$ است که درآن $a_1, \ldots, a_n$ اعداد صحیح و نامنفی هستند. درجه این چندجمله‌ای برابر است با $a_1 + a_2 + \cdots + a_n$. همچنین ${x_1}^{a_1}{x_2}^{a_2}\cdots{x_n}^{a_n}$ را با x^a نمایش می‌دهیم که در آن $x = (x_1, \ldots, x_n)$ و $a = (a_1, \ldots, a_n)$ و درجه x^a را با $|a|$ نمایش می‌دهیم.

تعریف ۲.۳.۳. یک چندجمله‌ای f را می‌توان به شکل $\sum_{\alpha} a_\alpha x^\alpha$ بنویسیم که a_α ها متعلق به K هستند و تعداد اندیس‌های α متنهاهی است. مجموعه همه چندجمله‌ای ها بر حسب $x_1, x_2, \ldots, x_n$ با ضرایب در K را با $R = K[x_1, x_2, \ldots, x_n]$ نیز یک حلقه جابه‌جایی است.

تعریف ۳.۳.۳. فرض کنیم $\sum_{\alpha} a_\alpha x^\alpha$ یک چندجمله‌ای در R باشد.

۱) a_α را ضریب تک‌جمله‌ای x^α می‌نامیم.

۲)اگر ضریب تک جمله‌ای مخالف صفر باشد آنگاه $a_\alpha x^\alpha$ را یک جمله از f گوییم.

۳)درجه f با نماد $\deg(f)$ برابر است با ماکزیمم$|a|$ به قسمی که a_α مخالف صفر باشد.

تعریف ۴.۳.۳. هرگاه درجه هر جمله از f برابر با d باشد چندجمله‌ای $f \in R$ را همگن از درجه d گوییم.

قضیه ۱.۳.۳. اگر K یک دامنه صحیح باشد آنگاه $K[x_1, x_2, \ldots, x_n]$ نیز یک دامنه صحیح است.

تعریف ۵.۳.۳. ایده‌آل تولید شده توسط چندجمله‌ای‌های $f_1, \ldots, f_s \in R$ را به صورت زیر تعریف می‌کنیم:

$$\langle f_1, f_2, \ldots, f_s\rangle := \{p_1 f_1 + \cdots + p_s f_s | p_1, \ldots, p_s \in R\}$$

تعریف ۶.۳.۳. یک ترتیب تک‌جمله‌ای روی R یک ترتیب کلی $\prec$ روی $\mathbb{N}^n$ است، هرگاه برای هر

$\alpha, \beta, \gamma \in \mathbb{N}^n$

۱) اگر $\alpha \prec \beta$ آنگاه $\alpha + \gamma \prec \beta + \gamma$.

۲) $\prec$ روی $\mathbb{N}^n$ خوش‌ترتیب باشد.

توجه می‌کنید که هرتک‌جمله‌ای $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ از یک‌n-تایی $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ ساخته می‌شود، یعنی یک تناظر یک‌به‌یک بین $\mathbb{N}^n$ و تک‌جمله‌ای‌های R وجود دارد. با استفاده از این تناظر، برای هر ترتیب $\prec$ روی $\mathbb{N}^n$ می‌نویسیم $x^\alpha \prec x^\beta$ هرگاه $\alpha \prec \beta$.

ترتیب‌های زیادی روی تک‌جمله‌ای وجود داردو ما تنها سه ترتیب تک‌جمله‌ای مهم را در ادامه معرفی می‌کنیم.

تعریف ۷.۳.۳. (ترتیب الفبایی) فرض کنیم $\alpha, \beta \in \mathbb{N}^n$. گوییم $\alpha \prec_{lex} \beta$ هرگاه اولین مؤلفه‌ی ناصفر $\beta - \alpha$ مثبت باشد.

مثال ۲.۳.۳. اگر $\alpha = (1, 2, 3)$ و $\beta = (1, 3, 7)$ آن‌گاه $\beta - \alpha = (0, 1, 4)$. بنابراین $\alpha \prec_{lex} \beta$، یعنی در $K[x, y, z]$ داریم $xy^2z^3 \prec_{lex} xy^3z^7$.

تعریف ۸.۳.۳. (ترتیب الفبایی مدرج) فرض کنیم $\alpha, \beta \in \mathbb{N}^n$. گوییم $\alpha \prec_{dlex} \beta$ هرگاه

$$|\alpha| = \sum_{i=1}^{n} \alpha_i < \sum_{i=1}^{n} \beta_i = |\beta|$$

یا اگر $|\alpha| = |\beta|$ آن‌گاه $\alpha \prec_{dlex} \beta$.

مثال ۳.۳.۳. توجه می‌کنیم که $(0, 2, 5, 0) \prec_{dlex} (1, 2, 3, 0)$ و $(1, 2, 5, 2) \prec_{dlex} (1, 2, 3, 4)$. پس در $K[x, y, z, w]$ داریم $y^2z^5 \prec_{dlex} xy^2z^3$ و $xy^2z^5w^2 \prec_{dlex} xy^2z^3w^4$.

تعریف ۹.۳.۳. (ترتیب الفبایی معکوس مدرج) فرض کنیم $\alpha, \beta \in \mathbb{N}^n$. گوییم $\alpha \prec_{drl} \beta$ هرگاه $|\alpha| < |\beta|$ یا اگر $|\alpha| = |\beta|$ آن‌گاه آخرین مؤلفه‌ی ناصفر $\beta - \alpha$ منفی باشد.

مثال ۴.۳.۳. اگر $\alpha = (3, 2, 5)$ و $\beta = (2, 3, 5)$ آن‌گاه $\alpha - \beta = (1, -1, 0)$. بنابراین $\beta \prec_{drl} \alpha$. پس در $K[x, y, z]$ داریم $x^3y^2z^5 \prec_{drl} x^2y^3z^5$.

تعریف ۱۰.۳.۳. فرض کنیم $f = \sum_\alpha a_\alpha x^\alpha$ یک چندجمله‌ای غیرصفر در R و $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد.

۱) درجه چندگانه‌ی f بانماد $multideg(f)$ برابر است با ماکزیمم $\alpha \in \mathbb{N}^n$ نسبت به $\prec$ به قسمی که $a_\alpha \neq ۰$.

۲) $a_{multideg(f)}$ را ضریب پیشروی f می‌نامیم و آن را با $LC(f)$ نمایش می‌دهیم.

۳) $x^{multideg(f)}$ را تک‌جمله‌ای پیشروی f می‌نامیم و آن را با $LM(f)$ نمایش می‌دهیم.

۴) $LC(f).LM(f)$ را جمله پیشروی f می‌نامیم و آن را با $LT(f)$ نمایش می‌دهیم.

مثال ۳.۳.۵. فرض کنید $f = ۴xy^۲z + ۴z^۲ - ۵x^۳ + ۷x^۲z^۲$ و $z \prec_{lex} y \prec_{lex} x$ در این صورت

$$multideg(f) = (۳, ۰, ۰), LC(f) = -۵, LM(f) = x^۳, LT(f) = -۵x^۳$$

قضیه ۳.۳.۶. (الگوریتم تقسیم در $R = K[x_۱, \ldots, x_n]$) فرض کنیم $\prec$ یک ترتیب چندجمله‌ای روی R باشد و $f_۱, \ldots, f_s \in R$. در این صورت هرچندجمله‌ای $f \in R$ را می‌توان به صورت $f = q_۱f_۱ + \cdots + q_sf_s + r$ نوشت که $q_۱, \ldots, q_s, r \in R$ و $r = ۰$ یا هیچ جمله‌ای از r توسط $LT(f_۱), \ldots, LT(f_s)$ عاد نمی‌شود. دراین حالت $q_۱, \ldots, q_s$ را خارج‌قسمت و r را یک باقی‌مانده تقسیم f به $f_۱, \ldots, f_s$ می‌نامیم.

اثبات. با ارائه‌ی یک الگوریتم و اثبات پایان‌پذیری و درستی آن، نشان می‌دهیم که $q_۱, \ldots, q_s, r$ با ویژگی‌های بالا وجود دارند.

ابتدا نشان می‌دهیم الگوریتم پایان‌پذیر است. فرض کنیم $q_{۱i}, \ldots, q_{si}, r_i, p_i$ نتایج بدست آمده برای $q_۱, \ldots, q_s, r, p$ درمرحله i -ام اجرای الگوریتم باشند. پس $p_۰ = f$ و $q_{۱۰} = \cdots = q_{s۰} = r_۰ = ۰$.
در هربار اجرای حلقه $while$ دوم، یکی از حالات زیر اتفاق می‌افتد:

حالت اول: $LT(f_i)|LT(p)$. درنتیجه $p_{j+۱} = p_j - \frac{LT(p_j)}{LT(f_i)}f_i$. از طرفی

$$LT(\frac{LT(p_j)}{LT(f_i)}f_i) = \frac{LT(p_j)}{LT(f_i)}LT(f_i) = LT(p_j)$$

. بنابراین به ازای هر j ، $LT(p_{j+۱}) \prec LT(p_j)$.

حالت دوم: $LT(f_i) \nmid LT(p)$. پس $p_{j+۱} = p_j - LT(p_j)$ لذا به ازای هر j ، $LT(p_{j+۱}) \prec LT(p_j)$.

اگر الگوریتم نزولی نباشد یک دنباله نزولی نامتناهی به صورت $\cdots \prec LT(p_۱) \prec LT(p_۰)$ خواهیم‌داشت که با فرض خوش‌ترتیب بودن $\prec$ در تناقض است. لذا پس از متنهاهی‌بار اجرای الگوریتم $p = ۰$ و الگوریتم پایان

الگوریتم ۱.۳ الگوریتم تقسیم در $k[x_1, \ldots, x_n]$

Input: $f_1, \ldots . f_s$ and $\prec$

Output: $q_1, q_2, \ldots, q_s, r$

$q_1 := 0; \ldots; q_s := 0; r := 0; p := f;$

while $p \neq 0$ **do**

$i := 1;$

$flag := false;$

while $i \leq s$ **and** $flag = false$ **do**

if $\mathrm{LT}(f_i) \mid \mathrm{LT}(p)$ **then**

$q_i := q_i + \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)};$

$p := p - \frac{\mathrm{LT}(p)}{\mathrm{LT}(f_i)} f_i;$

$flag := true;$

else

$i := i + 1;$

end if;

end while;

if $flag = false$ **then**

$r := r + \mathrm{LT}(p);$

$p := p - \mathrm{LT}(p);$

end if;

end while;

return $(q_1, \ldots, q_s, r);$

end;

می‌پذیرد برای اثبات درستی الگوریتم با استفاده از استقرا نشان می‌دهیم در مرحله‌ی j-ام الگوریتم

$$f = q_{1j} f_1 + \cdots + q_{sj} f_s + p_j + r_j. \tag{۱.۳}$$

به وضوح برای $j = ۰$ ، حکم برقرار است. فرض کنیم رابطه پیشین در مرحله $(j + ۱)$-ام برقرار باشد و حکم را برای مرحله $(j + ۱)$-ام ثابت می‌کنیم. اگر در مرحله $(j + ۱)$-ام در حلقه‌ی $while$ دوم، حالت اول روی دهد آن‌گاه $LT(f_i)|LT(p_{j+۱})$. درنتیجه

$$q_{۱(j+۱)}f_۱ + \cdots + q_{i(j+۱)}f_i + \cdots + q_{s(j+۱)}f_s + p_{j+۱} + r_{j+۱}$$

$$= q_{۱j}f_۱ + \cdots + (qij + \frac{LT(p_j)}{LT(f_i)})f_i + \cdots + q_{sj}f_s + (p_j - \frac{LT(p_j)}{LT(f_i)}f_i) + r_j$$

$$= q_{۱j}f_۱ + \cdots + q_{sj}f_s + p_j + r_j = f$$

و اگر حالت دوم روی دهد آن‌گاه

$$q_{۱(j+۱)}f_۱+\cdots+q_{s(j+۱)}f_s+p_{j+۱}+r_{j+۱} = q_{۱j}f_۱+\cdots+q_{sj}f_s+p_j+LT(p_j)+r_j+LT(p_j) = f.$$

بنابر رابطه قبل برقرار است. اما چون الگوریتم پایان‌پذیر است، در آخرین مرحله اجرای الگوریتم $p = ۰$. پس $f = q_۱f_۱ + \cdots + q_sf_s + r$. همچنین با توجه به ساختار الگوریتم، یا هیچ جمله ای به r اضافه نمی‌شود که در این صورت $r = ۰$ و یا جملاتی به r اضافه می‌شوند که هیچ‌کدام توسط $LT(f_i)$ عاد نمی‌شوند؛ یعنی هیچ جمله‌ای از r توسط $LT(f_i)$ ها عاد نمی‌شود. □

باقیمانده تقسیم یک چندجمله‌ای به دنباله‌ای از چندجمله‌ای‌ها در حالت کلی یکتا نیست.

مثال ۳.۳.۷. فرض کنیم $f = x^۲y + xy^۲ + y^۲$ ، $f_۱ = y^۲ - ۱$ و $f_۲ = xy - ۱$ چندجمله‌ای‌هایی در $K[x, y]$ باشند و $y \prec_{lex} x$ با تقسیم f به $(f_۱, f_۲)$ داریم

$$f = (x + ۱)(y^۲ - ۱) + x(xy - ۱) + (۲x + ۱)$$

و با تقسیم f به $(f_۲, f_۱)$ خواهیم‌داشت

$$f = (x + y)(xy - ۱) + (y^۲ - ۱) + (x + y + ۱).$$

فرض کنیم $f, f_1, \ldots, f_s$ چندجمله‌ای‌هایی در R باشند. اگر باقیمانده‌ی تقسیم f به $f_1, \ldots, f_s$ صفر باشد آن‌گاه $f \in \langle f_1, \ldots, f_s \rangle$.

صفر بودن باقیمانده تقسیم، شرط کافی برای عضویت f در ایده‌آل $I = \langle f_1, \ldots, f_s \rangle$ است. در مثال زیر نشان می‌دهیم $r = 0$ شرط لازم برای عضویت f در I نیست.

مثال ۸.۳.۳. فرض کنیم $f = xy^2 - x$ ، $f_1 = xy + 1$ و $f_2 = y^2 - 1$ چندجمله‌ای‌هایی در $K[x, y]$ باشند و $y \prec_{lex} x$. با تقسیم f به (f_1, f_2) داریم

$$f = y \cdot (xy + 1) + 0 \cdot (y^2 - 1) + (-x - y)$$

و با تقسیم f به (f_2, f_1) خواهیم‌داشت

$$f = x \cdot (y^2 - 1) + 0 \cdot (xy + 1) + 0 \in \langle f_1, f_2 \rangle.$$

در بخش بعد، با معرفی پایه‌ی گربنر خواهیم دید که تحت چه شرایطی صفر شدن باقیمانده شرط لازم و کافی برای عضویت f در $\langle f_1, f_2, \ldots, f_s \rangle$ است.

۴.۳ پایه گربنر

برای معرفی پایه گربنر ابتدا ایده‌آل‌های تک جمله‌ای در $R = K[x_1, \ldots, x_n]$ را تعریف می‌کنیم و لم دیکسون را برای آن‌ها بیان می‌کنیم.

تعریف ۱.۴.۳. ایده‌آل $I \subset R$ را یک ایده‌آل تک‌جمله‌ای گوییم هرگاه توسط مجموعه‌ای از تک جمله‌ای‌ها تولید شود. در این حالت می‌نویسیم $I = \langle x^\alpha | \alpha \in A \rangle$ که $A \subset \mathbb{N}^n$.

مثال ۱.۴.۳. $I = \langle x^4y^2, x^3y^4, x^2y^5 \rangle \subset K[x, y]$ یک ایده‌آل تک‌جمله‌ای است.

حال لم زیر را بدون اثبات بیان می‌کنیم که در اثبات لم دیکسون مورد استفاده است.

لم ۲.۴.۳. فرض کنیم $A \subset \mathbb{N}^n$ ، $I = \langle x^\alpha | \alpha \in A \rangle$ یک ایده‌آل تک‌جمله‌ای و $x^\beta \in R$ یک تک‌جمله‌ای باشد. در این صورت $x^\beta \in I$ اگر و تنها اگر $\alpha_0 \in A$ وجود داشته باشد که $x^{\alpha_0} | x^\beta$.

در اینجا لم دیکسون و اثبات آن را می‌آوریم که بیان می‌کند هر ایده‌آل تک‌جمله‌ای R یک مجموعه مولد متناهی دارد.

قضیه ۳.۴.۳. (لم دیکسون) فرض کنیم $A \subset \mathbb{N}^n$ و $I = \langle x^\alpha | \alpha \in A \rangle \subset K[x_1, \dots, x_n]$ یک ایده‌آل تک‌جمله‌ای باشد. دراین صورت I را می‌توان به صورت $I = \langle x^{\alpha(1)}, \dots, x^{\alpha(s)} \rangle$ نوشت که $\alpha(1), \dots, \alpha(s) \in A$، یعنی I دارای یک مجموعه متناهی از تک‌جمله‌ای‌هاست.

اثبات. اثبات را با استقرا روی n انجام می‌دهیم. اگر $n = 1$ آن‌گاه $I = \langle x_1^\alpha | \alpha \in A \rangle$ که $A \subset \mathbb{N}$. فرض کنیم β کوچکترین عضو A باشد. پس برای هر $\alpha \in A$ داریم $\beta \leq \alpha$. بنابراین $x_1^\beta | x_1^\alpha$. درنتیجه $I = \langle x_1^\beta \rangle$. حال فرض کنیم حکم برای $n-1$ برقرار باشد و $I \subset K[x_1, \dots, x_{n-1}, y]$ یک ایده‌آل تک‌جمله‌ای باشد. در این صورت $J = I|_{y=1}$ یک ایده‌آل تک‌جمله در $K[x_1, \dots, x_{n-1}]$ است. لذا بنا بر فرض استقرا می‌توان نوشت $J = \langle x^{\alpha(1)}, \dots, x^{\alpha(s)} \rangle$. از طرفی برای هر $1 \leq i \leq s$، $0 \leq m_i$ وجود دارد که $x^{\alpha(1)} y^{m_i} \in I$. قرار می‌دهیم $m = \max\{m_1, \dots, m_s\}$ وبرای هر $0 \leq l \leq m-1$ تعریف می‌کنیم $J_l = \langle x^\beta \in k[x_1, \dots, x_{n-1}] | x^\beta y^l \in I \rangle \subset K[x_1, \dots, x_{n-1}]$ دوباره طبق فرض استقرا، برای هر l می‌توان نوشت $J_1 = \langle x^{\alpha_l(1)}, \dots, x^{\alpha_l(s_l)} \rangle$. ادعا می‌کنیم که I توسط عناصر زیر تولید می‌شود

$$
\begin{gathered}
x^{\alpha(1)} y^m, \dots, x^{\alpha(s)} y^m \\
x^{\alpha_0(1)}, \dots, x^{\alpha_0(s_0)} \\
x^{\alpha_1(1)} y, \dots, x^{\alpha_1(s_1)} y \\
\vdots \\
x^{\alpha_{m-1}(1)} y^{m-1}, \dots, x^{\alpha_{m-1}(s_{m-1})} y^{m-1}.
\end{gathered}
$$

برای اثبات ادعا، فرض کنیم $x^\alpha y^p \in I$ یک تک‌جلهای باشد که $x^\alpha \in K[x_1, \dots, x_{n-1}]$. دو حالت امکان پذیر است:

حالت اول: اگر $p \geq m$ آنگاه $x^\alpha \in J$. لذا بنا بر لم ۲.۴.۳، $1 \leq i \leq s$ وجود دارد که $x^{\alpha(i)} | x^\alpha$. درنتیجه $x^{\alpha(i)} y^m | x^\alpha y^p$.

حالت دوم: اگر $p < m$ آنگاه $x^\alpha \in J_p$. لذا بنابر لم ۲.۴.۳، $1 \leq i \leq s_p$ وجود دارد که $x^{\alpha_p(i)} | x^\alpha$. در نتیجه $x^{\alpha_p(i)} y^p | x^\alpha y^p$. بنابراین حکم ثابت می‌شود. □

در ادامه قضیه اساسی هیلبرت را که بیان می‌کند هر ایده‌آل، یک مجموعه مولد متناهی دارد بیان می‌کنیم. ابتدا چند تعریف و نکته را می‌آوریم.

تعریف ۲.۴.۳. فرض کنیم $I \subset R$ یک ایده‌آل ناصفر و $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد. مجموعه جملات پیشروی اعضای I را با $LT(I)$ نمایش می‌دهیم، یعنی

$$LT(I) = \{cx^\alpha | \exists f \in I : LT(f) = cx^\alpha\}$$

و $\langle LT(I)\rangle$ را ایده‌آل تولید شده توسط اعضای $LT(I)$ تعریف می‌کنیم. پس $\langle LT(I)\rangle = \langle LT(f)|f \in I\rangle$.

تذکر ۱.۴.۳. فرض کنیم $f_1, \ldots, f_s$ چندجمله‌ای‌هایی در R باشند و $I = \langle f_1, \ldots, f_s\rangle$ در این صورت طبق تعریف $\langle LT(f_1), \ldots, LT(f_s)\rangle \subset \langle LT(I)\rangle$ ، اما عکس رابطه شمول در حالت کلی برقرار نیست. برای مثال فرض کنیم $f_1 = x^3 - 2xy$ ، $f_2 = x^2y - 2y^2 + x$ ، $I = \langle f_1, \ldots, f_s\rangle$ و $y \prec_{dlex} x$. در این صورت

$$x^2 = x \cdot (x^2y - 2y^2 + x) - y \cdots (x^3 - 2xy) \in I$$

. بنابراین $x^2 = LT(x^2) \in \langle LT(I)\rangle$. درحالتی که $LT(f_1) = x^3$ و $LT(f_2) = x^2y$ ، x^2 را عاد نمی‌کنند. لذا بنابر لم ۲.۴.۳ داریم $x^2 \notin \langle LT(f_1), LT(f_2)\rangle$.

قضیه ۴.۴.۳. فرض کنیم I ایده‌آلی در R باشد. در این صورت

الف) $\langle LT(I)\rangle$ یک ایده‌آل تک‌جمله‌ای است.

ب) چندجمله‌ای‌های $g_1, \ldots, g_t \in I$ وجود دارند به طوری که $\langle LT(I)\rangle = \langle LT(g_1)\rangle, \ldots . \langle LT(g_t)\rangle$.

اثبات. (الف) واضح است. برای اثبات قسمت (ب) چون $\langle LT(I)\rangle = \langle LT(g)|g \in I\rangle$ لذا بنابر لم دیکسون، چندجمله‌ای‌های $g_1, \ldots, g_t \in I$ وجود دارند که $\langle LT(I)\rangle = \langle LT(g_1)\rangle, \ldots . \langle LT(g_t)\rangle$ و حکم برقرار است. □

قضیه ۵.۴.۳. (قضیه اساسی هیلبرت) هر ایده‌آل $I \subset R$ دارای یک جموعه مولد متناهی است.

اثبات. اگر $I = \{0\}$ آنگاه $\{0\}$ مجموعه مولد متناهی مورد نظر است. در غیر این صورت، بنابر گزاره قبل چندجمله‌ای‌های $g_1, \ldots, g_t \in I$ وجود دارند به طوری که $\langle LT(I)\rangle = \langle LT(g_1)\rangle, \ldots, \langle LT(g_t)\rangle$.

ادعا می‌کنیم $I = \langle g_1, \ldots, LT(g_t)\rangle$. به وضوح $\langle g_1, \ldots, g_t\rangle \subset I$ ، زیرا $g_1, \ldots, LT(g_t) \in I$. برعکس، فرض کنیم $f \in I$. با استفاده از الگوریتم تقسیم در R ، چند جمله‌ای‌های $q_1, \ldots, q_t$ وجود $r \in R$ دارند که $f = q_1g_1 + \cdots + q_tg_t + r$ که $r = 0$ یا هیچ جمله ای از r توسط $LT(g_1), \ldots, LT(g_t)$ عاد نمی‌شود. توجه می‌کنیم که $r = f - q_1g_1 + \cdots + q_tg_t$. اگر $r \neq 0$ آنگاه $LT(r) \in \langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$. لذا بنابر لم ۲.۴.۳، $LT(r)$ باید توسط یکی از $LT(g_i)$ ها عاد شود که در تناقض با باقیمانده بودن r است. درنتیجه $r = 0$ و $f = q_1g_1 + \cdots + q_tg_t \in \langle g_1, \ldots, g_t\rangle$. □

پایه $\{g_1, \ldots, g_t\}$ در قضیه اساسی هیلبرت دارای این ویژگی است که $\langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$. می‌دانیم که همه مولدها این ویژگی را ندارند . این پایه های خاص را به صورت زیر تعریف می‌کنیم.

تعریف ۳.۴.۳. فرض کنیم $I \subset R$ یک ایده‌آل و $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد. زیرمجموعه متناهی از $G = \{g_1, \ldots, g_t\}$ را یک پایه گربنر I نسبت به $\prec$ گوییم هرگاه $\langle LT(I)\rangle = \langle LT(g_1), \ldots, LT(g_t)\rangle$.

پس اگر $G = \{g_1, \ldots, g_t\}$ یک پایه گربنر I نسبت به ترتیب $\prec$ باشد آنگاه $I = \langle g_1, \ldots, g_t\rangle$.
در نتیجه پایه گربنر یک ایده‌آل یکتا نیست.

تعریف ۴.۴.۳. فرض کنیم R یک حلقه باشد. R را نوتری گوییم هرگاه هر زنجیر صعودی $I_1 \subset I_2 \subset I_3 \subset \cdots$ از ایده‌آل‌های R پایان‌پذیر باشد.

قضیه ۶.۴.۳. حلقه $R = K[x_1, \ldots, x_n]$ نوتری است.

اثبات. فرض کنیم $I_1 \subset I_2 \subset I_3 \subset \cdots$ یک زنجیر صعودی از ایده‌آل‌های R باشد. قرار می‌دهیم $I = \bigcup_{i=1}^{\infty} I_i$ که ایده‌آلی در R است. بنابر قضیه اساسی هیلبرت می‌توانیم بنویسیم $I = \langle f_1, \ldots, f_s\rangle$. از طرفی برای هر $1 \leq i \leq s$، عدد j_i وجود دارد که $f_i \in I_{j_i}$. قرار می‌دهیم $N = \max\{j_1, \ldots, j_s\}$. بنابراین $f_1, \ldots, f_s \in I_N$ و در نتیجه $I = \langle f_1, \ldots, f_s\rangle \subset I_N \subset I_{N+1} \subset \cdots \subset I$.

□

تعریف ۵.۴.۳. مجموعه متناهی $G \subset R$ را یک پایه گربنر گوییم هرگاه G یک پایه گربنر برای $\langle G\rangle$ باشد.

گزاره ۱.۴.۳. فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R ، $f \in R$ و $G = \{g_1, \dots, g_t\}$ یک پایه گربنر برای ایده‌آل $I \subset R$ باشد. در این صورت $r \in R$ یکتا با ویژگی‌های زیر وجود دارد.

الف) $r = 0$ یا $LT(g_1), \dots, LT(g_t)$ هیچ جمله‌ای از r را عاد نمی‌کنند.

ب)چندجمله‌ای $g \in I$ وجود دارد که $f = g + r$.

اثبات. بنابر الگوریتم تقسیم می‌توان نوشت $f = q_1 g_1 + \cdots + q_t g_t + r$ که r در شرایط (الف) صدق می‌کند. همچنین شرایط (ب) برقرار است؛ زیرا $g = q_1 g_1 + \cdots + q_t g_t \in I$. حال برای اثبات گزاره کافیست اثبات کنیم r یکتاست. فرض کنیم $f = g + r = g' + r'$ که شرایط (الف)و(ب) برقرارند. پس $r - r' = g' - g \in I$. اگر $r \neq r'$ آنگاه $LT(r - r') \in \langle LT(I) \rangle = \langle LT(g_1), \dots, LT(g_t) \rangle$ در نتیجه بنابر لم ۲.۴.۳ ، برای یک $1 \leq i \leq t$ داریم $LT(g_i) | LT(r - r')$ از آنجایی که $LT(r - r')$ جمله‌ای از r یا r' است، پس $LT(g_i)$ یکی از جملات r یا r' را عاد می‌کند که غیر ممکن است. لذا $r = r'$.

□

تعریف ۶.۴.۳. فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R ، $f \in R$ و G یک پایه گربنر باشد.باقیمانده تقسیم f بر G شکل متعارف f می‌نامیم و آن را با نماد $NF_G(f)$. نمایش می‌دهیم.

نتیجه ۱.۴.۳. فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R و G یک پایه گربنر ایده‌آل $I \subset R$ باشد. در این صورت $f \in I$ اگر و تنها اگر $NF_G(f) = 0$.

تعریف ۷.۴.۳. فرض کنیم $f, f_1, \dots, f_s \in R$ و r یک باقیمانده تقسیم f بر $F = \{f_1, \dots, f_s\}$ باشد. در این صورت r را با نماد $rem(f, F)$ نمایش می‌دهیم و می‌نویسیم $f \to_F^* r$.

قضیه ۷.۴.۳. فرض کنیم $I \subset R$ یک ایده‌آل، $G = \{g_1, \dots, g_t\} \subset I$ و $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد. در این صورت عبارات زیر معادل‌اند.

الف) G یک پایه گربنر برای I است.

ب) $f \in I$ اگر و تنها اگر $rem(f, G) = 0$.

پ) $f \in I$ اگر و تنها اگر $f = \sum_{i=1}^{t} h_i g_i$ به قسمی که $LM(f) = \max_{\prec}\{LM(h_i)LM(g_i) | 1 \leq i \leq t\}$.

اثبات. اثبات بخش (الف)به(ب)همان نتیجه ۱.۴.۳ است.

اثبات (ب) به(پ) اگر $f \in I$ آنگاه طبق فرض $rem(f,G) = ۰$ لذا طبق الگوریتم تقسیم داریم $f = \sum_{i=۱}^{t} h_i g_i$ که با توجه به روند الگوریتم $LM(f) = \max_{\prec}\{LM(h_i)LM(g_i) | ۱ \leq i \leq t\}$ برعکس واضح است.

اثبات (پ) به (الف) با توجه به تعریف پایه گربنر کافیست ثابت کنیم $\langle LT(I)\rangle = \langle LT(G)\rangle$ واضح است که $\langle LT(G)\rangle \subset \langle LT(I)\rangle$ برای اثبات عکس رابطه شمول باید ثابت کنیم برای هر $f \in I$ ،

$LT(f) \in \langle LT(G)\rangle$ طبق فرض $f = \sum_{i=۱}^{t} h_i g_i$ که

$LM(f) = \max_{\prec}\{LM(h_i)LM(g_i) | ۱ \leq i \leq t\}$. پس $۱ \leq i \leq t$ وجود دارد که $LM(f) = LM(h_i)LM(g_i)$ و $LM(g_i) | LM(f)$.

□

در ادامه به دنبال پیدا کردن یک پایه گربنر یکتا برای یک ایده‌آل داده شده هستیم. با استفاده از لم زیر می‌توانیم با حذف مولدهای غیرضروری، پایه گربنر کوچک‌تری برای یک ایده‌آل پیدا کنیم.

لم ۱.۴.۳. فرض کنیم G یک پایه گربنر ایده‌آل $I \subset R$ باشد و $g \in G$ اگر $LT(g) \in \langle LT(G\backslash\{g\})\rangle$ آنگاه $G\backslash\{g\}$ نیز یک پایه گربنر برای ایده‌آل I خواهد بود.

اثبات. می‌دانیم $\langle LT(I)\rangle = \langle LT(G)\rangle$ اگر $LT(g) \in \langle LT(G\backslash\{g\})\rangle$ آنگاه $\langle LT(G\backslash\{g\})\rangle = \langle LT(G)\rangle$. لذا طبق تعریف، $G\backslash\{g\}$ نیز یک پایه گربنر برای I است. □

تعریف ۸.۴.۳. یک پایه گربنر G برای ایده‌آل $I \subset R$ را یک پایه گربنر مینیمال I گوییم هرگاه برای هر $g \in G$

۱) $LC(G) = ۱$

۲) $LT(g) \notin \langle LT(G\backslash\{g\})\rangle$

مثال ۸.۴.۳. ایده‌آل I در تذکر ۱.۴.۳ را در نظر می‌گیریم. در بخش بعد خواهیم دید مجموعه

$$G = \{f_۱ = x^۳ - ۲xy, f_۲ = x^۲y - ۲y^۲ + x, f_۳ = -x^۲, f_۴ = -۲xy, f_۵ = -۲y^۲ + x\}$$

یک پایه گربنر I نسبت به ترتیب $y \prec_{dlex} x$ است. همچنین

$$LT(f_۱) = x^۳, LT(f_۲) = x^۲y, LT(f_۳) = -x^۲, LT(f_۴) = -۲xy, LT(f_۵) = -۲y^۲$$

چون $LT(f_۳)|LT(f_۱)$ و $LT(f_۴)|LT(f_۲)$، لذا با حذف $f_۱$ و $f_۲$ مجموعه $G' = \{x^۲, xy, y^۲ - ۱/۲x\}$ یک پایه گربنر مینیمال برای I است. همچنین برای هر $a \in k$ مجموعه $G" = \{x^۲, axy, xy, y^۲ - ۱/۲x\}$ نیز یک پایه گربنر مینیمال برای I است.

در مثال قبل دیدیم که پایه گربنر مینیمال یکتا نیست. در تعریف زیر، یک پایه گربنر یکتا را معرفی می‌کنیم.

تعریف ۹.۴.۳. یک پایه گربنر G برای ایده‌آل $I \subset R$ را یک پایه گربنر کاهش‌یافته برای I گوییم هرگاه برای هر $g \in G$

۱) $LC(g) = ۱$.

۲) هیچ جمله ای از g متعلق به $\langle LT(G\backslash\{g\})\rangle$ نباشد.

گزاره ۲.۴.۳. فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد. در این صورت هر ایده‌آل $I \subset R$ دارای یک پایه گربنر کاهش‌یافته یکتا نسبت به $\prec$ است.

مثال ۹.۴.۳. در مثال ۸.۴.۳ مجموعه G' پایه گربنر کاهش‌یافته یکتای ایده‌آل I است.

۵.۳ برخی کاربردهای پایه گربنر

فرض کنیم $R = K[x_۱, \ldots, x_n]$ و $I \subset R$ یک ایده‌آل باشد. در این بخش، فضای خارج‌قسمتی R/I و ایده‌آل صفر بعدی را تعریف می‌کنیم و سپس قضیه محک تناهی را برای k- فضای برداری R/I بیان می‌کنیم.

تعریف ۱.۵.۳. فرض کنیم I ایده‌آلی ار حلقه R باشد. $f, g \in R$ را همنهشت به پیمانه I گوییم و می‌نویسیم $f \equiv g \mod I$ هرگاه $f - g \in I$ توجه می‌کنیم که $\equiv$ یک رابطه هم‌ارزی است.

تعریف ۲.۵.۳. فرض کنیم I ایده‌آلی از حلقه R باشد. مجموعه کلاس‌های هم‌ارزی به پیمانه I را خارج‌قسمت R به پیمانه I می‌نامیم و آن را به صورت $R/I = \{f + I | f \in R\}$ تعریف می‌کنیم. برای هر $g, f \in R$ و R/I همراه با جمع و ضرب زیر یک حلقه جابه‌جایی هست که به آن حلقه خارج‌قسمتی می‌گوییم.

$$(f, I) + (g + I) = (f, g) + I, (f + I)(g + I) = fg + I.$$

تذکر ۱.۵.۳. R/I یک فضای برداری روی میدان K است.

تعریف ۳.۵.۳. فرض کنیم $I \subset R$ یک ایده‌آل باشد. $U = \{u_1, \ldots, u_t\} \subset \{x_1, \ldots, x_n\}$ را مستقل به پیمانه I گوییم هرگاه $I \cap k[u_1, \ldots, u_t] = \{0\}$.

تعریف ۴.۵.۳. بعد ایده‌آل $I \subset R$ را به صورت زیر تعریف می‌کنیم

$$dim(I) := \max\{|U||U \subset \{x_1, \ldots, x_n\}\}.$$

مثال ۱.۵.۳. مجموعه $G = \{xz+z, yz+z\}$ یک پایه گربنر نسبت به ترتیب تک‌جمله‌ای $z \prec_{lex} y \prec_{lex} x$ است. همچنین $\{x\}$ ، $\{y\}$ ، $\{z\}$ و $\{x, y\}$ مستقل به پیمانه $\langle G\rangle$ هستند، در حالی که $\{x, z\}$ ، $\{y, z\}$ و $\{x, y, z\}$ مستقل به پیمانه $\langle G\rangle$ نیستند. پس $dim(\langle G\rangle) = 2$.

تعریف ۵.۵.۳. ایده‌آل $I \subset R$ را صفر بعدی گوییم هرگاه $dim(I) = 0$.

تذکر ۲.۵.۳. اگر ایده‌آل $I \subset R$ صفر بعدی باشد آنگاه برای هر $1 \leq i \leq n$ داریم $I \cap K[x_i] \neq \{0\}$ ، یعنی یک چندجمله‌ای غیر صفر بر حسب x_i در I وجود دارد. برعکس، اگر برای هر $1 \leq i \leq n$ ، یک چندجمله‌ای غیر صفربر حسب x_i در I وجود داشته باشد آنگاه $I \cap K[x_i] \neq \{0\}$. در نتیجه $dim(I) = 0$. لذا قضیه زیر را داریم.

قضیه ۲.۵.۳. ایده‌آل I صفر بعدی است اگر و تنها اگر برای هر $1 \leq i \leq n$ داشته باشیم $I \cap K[x_i] \neq \{0\}$.

مثال ۳.۵.۳. فرض کنیم $I = \langle x^2, y^2\rangle$ ایده‌آلی در $K[x, y]$ باشد. در این صورت $x^2 \in I \cap K[x]$ و $y^2 \in I \cap K[x]$. در نتیجه $dim(I) = 0$

قضیه ۴.۵.۳. (قضیه مکولی) فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R و $G = \{g_1 \ldots, g_t\}$ یک پایه گربنر ایده‌آل $I \subset R$ باشد. در این صورت مجموعه

$$U = \{[u]|\forall i = 1, \ldots, t : LT(g_i) \nmid u\}$$

یک پایه برای k- فضای برداری R/I است.

اثبات. ابتدا نشان می‌دهیم هر عضو R/I توسط U تولید می‌شود. فرض کنیم $[f] = f + I \in R/I$ با تقسیم f بر G طبق الگوریتم تقسیم داریم $f = q_۱g_۱ + \cdots + q_tg_t + r$ که در آن $r = a_۱u_۱ + \cdots + a_mu_m$ و

$$\forall i = ۱, \ldots, t, \forall j = ۱, \ldots, m : LT(g_i) \nmid u_j$$

که $[u_۱], \ldots, [u_m] \in U$ از آن جایی که $q_۱g_۱ + \cdots + q_tg_t \in I$ ، پس

$$[f] = [q_۱g_۱ + \cdots + q_tg_t + a_۱u_۱ + \cdots + a_mu_m] = a_۱[u_۱] + \cdots + a_m[u_m].$$

حال ثابت می‌کنیم U یک مستقل خطی است. فرض کنیم $\{[u_۱], \cdots, [u_m]\} \subset U$ و $a_۱[u_۱] + \cdots + a_m[u_m] = [۰]$. در نتیجه $[a_۱u_۱] + \cdots + [a_mu_m] = [۰]$ لذا $f = a_۱u_۱ + \cdots + a_mu_m \in I$ اگر برای یک $۱ \leq j \leq m$ داشته باشیم $a_j \neq ۰$ آنگاه $f \neq ۰$ و لذا f دارای تک‌جمله‌ای پیشرو است. پس $g_i \in G$ وجود دارد که $LT(g_i)|LM(f_i)$ بنابراین $۱ \leq l \leq m$ وجود دارد که $LT(g_i)|u_l$ که در تناقض با فرض قضیه است و U مستقل خطی است. □

مثال ۵.۵.۳. فرض کنیم $I = \langle x^۲ + x, y^۲ + y\rangle$ ایده‌آلی در $K[x, y]$ باشد. در این صورت $G = \{x^۲ + x, y^۲ + y\}$ یک پایه گربنر I نسبت به ترتیب $x \prec_{lex} y$ است. همچنین $LT(I) = \langle x^۲, y^۲\rangle$ لذا $U = \{[۱], [x], [y], [xy]\}$ یک پایه $K-$ فضای برداری R/I است.

قضیه ۶.۵.۳. (محک تناهی) فرض کنیم $I \subset R$ یک ایده‌آل باشد. در این صورت عبارات زیر معادل هستند.

الف) $dim(I) = ۰$

ب) $k-$ فضای برداری R/I از بعد متناهی است

پ)برای هر ترتیب تک جمله ای $\prec$ روی R ، هر پایه گربنر G برای I نسبت به $\prec$ و هر $۱ \leq i \leq n$ ، چندجمله‌ای $g_i \in G$ و عدد صحیح مثبت v_i وجد دارد که $LM(g_i) = x_i^{v_i}$

ت) ترتیب تک‌جمله‌ای $\prec$ روی R و پایه گربنر G برای I نسبت به ترتیب $\prec$ وجود دارد به قسمی که برای هر $۱ \leq i \leq n$ چندجمله‌ای $g_i \in G$ و عددصحیح مثبت v_i وجود دارد که $LM(g_i) = x_i^{v_i}$

اثبات. (الف) $\Leftarrow$ (ب) فرض کنیم $dim(I) = ۰$ پس برای هر $۱ \leq i \leq n$ ، چندجمله‌ای $g_i \in I \cap K[x_i]$ وجود دارد. فرض کنیم $LM(g_i) = x_i^{v_i}$ لذا برای هر i داریم $LM(g_i) = LT$ بنا بر این اعضای R/I به

صورت $[x_1^{\alpha_1} \cdots x_n^{\alpha_n}]$ هستند که برای هر i ، $0 \leq a_i \leq v_i$ پس $dim_k(\frac{R}{I}) \leq v_1 \cdots v_n < \infty$.

(ب) $\Leftarrow$ (الف) فرض کنیم $dim_k(\frac{R}{I}) < \infty$ در این صورت برای هر $1 \leq i \leq n$ مجموعه $\{[1], [x], [y], [xy], \ldots\}$ وابسته خطی است. پس a_j هایی که همگی صفر نیستند وجود دارند به طوری که $\sum_j a_j[x_i^j] = [0]$ یعنی $\sum_j a_j x_i^j \in I \cap K[x_i]$ لذا برای هر i ، $I \cap K[x_i] \neq 0$. بنا بر این ایده‌آل I صفر بعدی است.

(الف) $\Leftarrow$ (پ) فرض کنیم $dim(I) = 0$ ، $\prec$ یک ترتیب تک‌جمله‌ای روی R و G یک پایه گربنر I نسبت به $\prec$ باشد. چون $dim(I) = 0$ پس برای هر $1 \leq i \leq n$ چندجمله‌ای $h_i \in I \cap K[x_i]$ وجود دارد. فرض کنیم $LM(h_i) = x_i^{t_i}$. از آنجایی که G یک پایه گربنر I است، لذا چندجمله‌ای $g_i \in G$ وجود دارد به طوری که $LM(g_i) | x_i^{t_i}$ یعنی $LM(g_i) = x_i^{v_i}$ که $v_i \leq t_i$. (پ) $\Leftarrow$ (الف) برای هر $1 \leq i \leq n$ ترتیب تک‌جمله‌ای $\prec_{lex}$ را به قسمی در نظر می‌گیریم که x_i از بقیه متغیرها کوچک‌تر باشد. همچنین فرض کنیم G یک پایه گربنر I نسبت به $\prec_{lex}$ باشد. در این صورت طبق فرض، چندجمله‌ای $g_i \in G$ و عدد صحیح مثبت v_i وجود دارد که $LM(g_i) = x_i^{v_i}$ ، یعنی $g_i \in I \cap K[x_i]$ در نتیجه $dim(I) = 0$

(پ) $\Leftarrow$ (ت) بدیهی است.

(ت) $\Leftarrow$ (ب) مشابه حالت (الف) $\Leftarrow$ (ب) است.

□

۶.۳ محاسبه پایه گربنر

در این بخش قضیه محک بوخبرگر را بیان می‌کنیم و با استفاده از آن، الگوریتم بوخبرگر را برای محاسبه پایه گربنر ارائه می‌دهیم.

تعریف ۱.۶.۳. فرض کنیم $f, g \in R$ دو چندجمله‌ای ناصفر باشند، $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد، $multideg(f) = \alpha$ و $multideg(g) = \beta$ قرار می‌دهیم $\gamma = (\gamma_1, \ldots, \gamma_n)$ که برای هر i ، $\gamma_i = max(\alpha_i, \beta_i)$ (x^γ را کوچکترین مضرب مشترک $LM(f)$ و $LM(g)$ می‌نامیم و می‌نویسیم $x^\gamma = lcm(LM(f), LM(g))$). در این صورت چندجمله‌ای-s f و g را به صورت زیر تعریف می‌کنیم

$$Spoly(f, g) := \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g.$$

مثال ۱.۶.۳. فرض کنیم $y \prec_{lex} x$ یک ترتیب تک‌جمله‌ای روی $R = \mathbb{R}[x, y]$ باشد و $f = x^۳y^۲ - x^۲y^۳ + x, g = ۳x^۴ * y + y^۲ \in R$. در این صورj $\gamma(۴, ۲)$

$$Spoly(f, g) = \frac{x^۴y^۲}{x^۳y^۲}.f - \frac{x^۴y^۲}{۳x^۴y}.g = -x^۳y^۳ + x^۲ - \frac{۱}{۳}y^۳.$$

در ادامه به اثبات لم زیر می‌پردازیم که برای اثبات قضیه محک بوخبرگر به آن نیازمندیم.

لم ۲.۶.۳. فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R باشد و $f_۱, \dots, f_k \in R$ چندجمله‌ای‌های مخالف صفر باشند که برای هر $۱ \leqslant i \leqslant k$، $LM(f_i) = x^\alpha$. در این صورت اگر $f = c_۱f_۱ + \cdots + c_kf_k$ که $c_i \in K$ و $LM(f) < x^\alpha$، آن‌گاه f را می‌توان به صورت ترکیب خطی از $spoly(f_i, f_j)$ نوشت که $۱ \leqslant i, j \leqslant k$.

اثبات. فرض کنیم برای هر i، $LM(f_i) = x^\alpha$، $LC(f_i) = d_i$ و $LC(c_if_i) = c_id_i$. ادعا می‌کنیم اگر $f = c_۱f_۱ + \cdots + c_kf_k$ و $LM(f) < x^\alpha$، آن‌گاه $c_۱d_۱ + \cdots + c_kd_k = ۰$. به همین منظور برای هر $۱ \leqslant j, l \leqslant k$ داریم:

$$Spoly(f_j, f_l) = \frac{x^\alpha}{d_jx^\alpha}f_j - \frac{x^\alpha}{d_lx^\alpha}f_l = \frac{f_j}{d_j} - \frac{f_l}{d_l}.$$

حال اگر به‌جای اولین جمله f یعنی $c_۱f_۱$ قرار دهیم $c_۱d_۱(\frac{f_۱}{d_۱} - \frac{f_۲}{d_۲} + \frac{f_۲}{d_۲})$، آن‌گاه $f = c_۱d_۱(\frac{f_۱}{d_۱} - \frac{f_۲}{d_۲} + \frac{f_۲}{d_۲}) + c_۲f_۲ + \cdots + c_kf_k$ که با یک محاسبه ساده نتیجه می‌شود:

$$f = c_۱d_۱(\frac{f_۱}{d_۱} - \frac{f_۲}{d_۲}) + (c_۱d_۱\frac{f_۲}{d_۲} + c_۲f_۲) + c_۳f_۳ + \cdots + c_kf_k. \qquad (۲.۳)$$

همچنین اگر به‌جای $c_۲f_۲$ در (۲.۳) قرار دهیم $c_۲d_۲(\frac{f_۲}{d_۲} - \frac{f_۳}{d_۳} + \frac{f_۳}{d_۳})$، آن‌گاه داریم:

$$f = c_۱d_۱(\frac{f_۱}{d_۱} - \frac{f_۲}{d_۲}) + (c_۱d_۱ + c_۲d_۲)(\frac{f_۲}{d_۲} - \frac{f_۳}{d_۳}) + (c_۱d_۱ + c_۲d_۲ + c_۳d_۳)\frac{f_۳}{d_۳} + c_۴f_۴ + \cdots + c_kf_k.$$

این روند را تا مرحله $k - ۱$ ادامه می‌دهیم، پس داریم:

$$f = c_۱d_۱(\frac{f_۱}{d_۱} - \frac{f_۲}{d_۲}) + \cdots + (c_۱d_۱ + \cdots + c_{k-۱}d_{k-۱})(\frac{f_{k-۱}}{d_{k-۱}} - \frac{f_k}{d_k}) + (c_۱d_۱ + \cdots + c_kd_k)\frac{f_k}{d_k}.$$

در نتیجه چون $LM(f) < x^\alpha$ پس $c_۱d_۱ + \cdots + c_kd_k = ۰$ و این یعنی f یک ترکیب خطی از $spoly(f_j, f_l)$-هاست. □

قضیه ۳.۶.۳. (محک بوخبرگر) فرض کنیم $\prec$ یک ترتیب تک‌جمله‌ای روی R و $G = \{g_1, \ldots, g_t\} \subset R$ مجموعه‌ای از چندجمله‌ای‌های غیر صفر باشند. در این صورت G یک پایه نسبت به $\prec$ است اگر و تنها اگر برای هر $g_i, g_j \in G$ که $i \neq j$، $spoly(g_i, g_j) \longrightarrow^*_G 0$.

اثبات. فرض کنیم $G = \{g_1, \ldots, g_t\}$ یک پایه گربنر برای $I = \langle G \rangle$ باشد. چون برای هر $g_i, g_j \in I$، داریم

$$spoly(g_i, g_j) = \frac{x^\alpha}{LT(g_i)} g_i - \frac{x^\alpha}{LT(g_j)} g_j.$$

بنابراین $spoly(g_i, g_j) \in I$. از طرفی G یک پایه گربنر برای I است پس باقیمانده تقسیم هر چندجمله‌ای در I نسبت به G صفر است. بنابراین $spoly(g_i, g_j) \longrightarrow^*_G 0$.

برعکس، فرض کنیم برای هر $g_i, g_j \in G$، $spoly(g_i, g_j) \longrightarrow^*_G 0$. نشان می‌دهیم G یک پایه گربنر $I = \langle G \rangle$ است. کافیست نشان دهیم برای هر $f \in I$، $g_i \in G$ وجود دارد که $LT(g_i) | LT(f)$. بنابراین فرض کنیم $f \in I$ پس $h_1, \ldots, h_t \in R$ وجود دارند که $f = \sum_{i=1}^{t} h_i g_i$. با توجه به این که h_iها منحصر بفرد نیستند پس این ترکیب یکتا نیست. بنابراین ترکیب $f = \sum_{i=1}^{t} h_i g_i$ را طوری انتخاب می‌کنیم که $X = \max\{LM(h_i g_i) \mid 1 \leqslant i \leqslant t\}$ کمترین نسبت به $\prec$ باشد. برای X سه حالت امکان پذیر است:

حالت اول: اگر $X \prec LM(f)$، آن‌گاه برای هر i داریم $LM(h_i g_i) \preceq X \prec LM(f)$. پس $LM(h_1 g_1 + \ldots + h_t g_t) \prec LM(f)$ و این یعنی $LM(f) \prec LM(f)$ که یک تناقض است.

حالت دوم: اگر $X = LM(f)$، آن‌گاه $h_1, \ldots, h_t \in R$ وجود دارند که $f = h_1 g_1 + \cdots + h_t g_t$ و $LM(f) = X = \max\{LM(h_1 g_1), \ldots, LM(h_t g_t)\}$. بنابراین برای i-ای داریم $LM(h_i g_i) = LM(f)$ و این معادل با $LM(h_i) LM(g_i) = LM(f)$ است. در نتیجه $LM(g_i) \mid LM(f)$ که در این حالت حکم مسئله ثابت می‌شود.

حالت سوم: اگر $LM(f) \prec X$، آن‌گاه با نشان دادن یک تناقض ثابت می‌کنیم این حالت امکان پذیر نیست. پس فرض کنیم $f = h_1 g_1 + \cdots + h_t g_t$ و $LM(f) \prec X = \max\{LM(h_1 g_1), \ldots, LM(h_t g_t)\}$. قرار می‌دهیم $S := \{i \mid LM(h_i g_i) = X\}$ و برای هر i فرض می‌کنیم $LT(h_i) = c_i X_i$ که $c_i \in K$ و $X_i \in R$ یک تک‌جمله‌ای است. همچنین قرار می‌دهیم $g = \sum_{i \in S} c_i X_i g_i$ پس برای هر i، $LM(X_i g_i) = LM(h_i g_i) = X$. چون $LM(f) \prec X$ بنابراین $LM(g) \prec X$. در نتیجه شرایط لم قبل برقرار است پس داریم $g = \sum_{i \in S} d_{ij} spoly(X_i g_i, X_j g_j)$ که $d_{ij} \in K$. به عبارتی دیگر می‌توان نوشت

$$spoly(X_ig_i, X_jg_j) = \frac{X}{LT(X_ig_i)} \cdot X_ig_i - \frac{X}{LT(X_jg_j)} \cdot X_jg_j.$$

همچنین فرض کنیم $X_{ij} = lcm(LM(g_i), LM(g_j))$ پس

$$spoly(X_ig_i, X_jg_j) = \frac{X_{ij} \cdot \frac{X}{X_{ij}}}{LT(X_ig_i)} \cdot X_ig_i - \frac{X_{ij} \cdot \frac{X}{X_{ij}}}{LT(X_jg_j)} \cdot X_jg_j = \frac{X}{X_{ij}} spoly(g_i, g_j) \quad (۳.۳)$$

از طرفی طبق فرض قضیه داریم $spoly(g_i, g_j) \longrightarrow^*_G ۰$. بنابراین $spoly(g_i, g_j) = \sum_{k=۱}^{t} f_kg_k$ و $LM(f_kg_k) \prec X_{ij}$. در نتیجه $\frac{X}{X_{ij}} \cdot LM(f_kg_k) \prec X$. پس با استفاده از (۳.۳) ثابت کردیم برای هر i, j می‌توانیم $spoly(X_ig_i, X_jg_j)$ را به صورت ترکیبی از $g_۱, \ldots, g_t$ با حداکثر جمله کوچکتر از X بنویسیم. بنابراین g را می‌توان با ترکیبی از $g_۱, \ldots, g_t$ با حداکثر جمله کوچکتر از X جایگزین کرد. در نتیجه f را می‌توان به صورت ترکیبی از جملات کوچکتر X نوشت که این در تناقض با کمینه بودن X است. □

در ادامه با توجه به محک بوخبرگر، الگوریتم بوخبرگر را ارائه می‌دهیم سپس درستی و پایان‌پذیری آن را اثبات می‌کنیم.

اثبات. ابتدا نشان می‌دهیم الگوریتم پایان‌پذیر است سپس درستی آن را اثبات می‌کنیم. فرض کنیم G_i نتیجه اجرای الگوریتم در مرحله iام باشد (منظور از مرحله iام، مرحله‌ای است که iامین چندجمله‌ای مخالف صفر محاسبه می‌شود) و $g_{i+۱}$، $i+۱$امین چندجمله‌ای مخالف صفر باشد. چون $g_{i+۱} \neq ۰$ و باقیمانده تقسیم یک چندجمله‌ای بر G_i است پس $LT(g_{i+۱}) \notin \langle LT(G_i) \rangle$. از طرفی چون $g_{i+۱}$ به $G_{i+۱}$ اضافه می‌شود پس $LT(g_i) \in \langle LT(G_{i+۱}) \rangle$. در نتیجه یک زنجیر اکیدا صعودی به صورت

$$\langle LT(G) \rangle \subsetneq \langle LT(G_۱) \rangle \subsetneq \cdots$$

داریم. چون R نوتری است پس این زنجیر ایستاست و در نتیجه الگوریتم پایان می‌پذیرد.

برای اثبات درستی الگوریتم فرض کنیم G پایه گربنر محاسبه شده در آخرین مرحله الگوریتم باشد. ابتدا نشان می‌دهیم $\langle G \rangle = \langle F \rangle$. واضح است که $\langle F \rangle \subseteq \langle G \rangle$ به همین منظور با استفاده از استقرا برای هر i نشان می‌دهیم $\langle G_i \rangle \subseteq \langle F \rangle$. اگر قرار دهیم $i = ۰$، آن‌گاه داریم $G_۰ = F$ و $\langle G_۰ \rangle \subseteq \langle F \rangle$. پس فرض کنیم $\langle G_i \rangle \subseteq \langle F \rangle$ نشان می‌دهیم $\langle G_{i+۱} \rangle \subseteq \langle F \rangle$. با توجه به الگوریتم بوخبرگر $g, g' \in G_i$ وجود دارند که $spoly(g, g') \longrightarrow^*_{G_i} g_{i+۱}$ پس $G_{i+۱} = G_i \cup \{g_{i+۱}\}$. چون $g, g' \in \langle G_i \rangle$ داریم $spoly(g, g') \longrightarrow^*_G g_{i+۱}$

Buchberger **3.2**

Require: $F = \{f_1, \ldots, f_k\} \subset R$ and $\prec$ a monomial ordering;

Ensure: G a Gröbner basis for $\langle F \rangle$;

$G := F$;

$P := \{\{g_i, g_j\} \mid g_i, g_j \in G\,,\ g_i \neq g_j\}$;

while $P \neq \emptyset$ **do**

Select $\{g_i, g_j\} \in P$;

$P := P \setminus \{\{g_i, g_j\}\}$;

$spoly(g_i, g_j) \longrightarrow^*_G h$;

if $h \neq 0$ **then**

$P := P \cup \{\{g, h\} \mid g \in G\}$;

$G := G \cup \{h\}$;

end if;

end while;

return (G);

$g_{i+1} \in \langle G_i \rangle$. این یعنی این که $\langle G_{i+1} \rangle \subseteq \langle G_i \rangle$، بنابراین $\langle F \rangle \subseteq \langle G_{i+1} \rangle$. در نتیجه داریم $\langle F \rangle = \langle G \rangle$. همچنین با توجه به محک بوخبرگر باید نشان دهیم برای هر $g, g' \in G$، $spoly(g, g') \longrightarrow^*_G 0$. بنابر برهان خلف فرض می‌کنیم $g, g' \in G$ وجود داشته باشند که $spoly(g, g') \longrightarrow^*_G r$ و $r \neq 0$ در نتیجه این در تناقض با پایان‌پذیری الگوریتم است پس G یک پایه گربنر $\langle F \rangle$ است. □

مثال ۴.۶.۳. فرض کنیم $I = \langle f_1, f_2 \rangle = \langle x^3 - 2xy, x^2y - 2y^2 + x \rangle$ ایده‌آلی در $K[x, y]$ باشد و $lexx \prec_d y$. دقت می‌کنیم $G = f_1, f_2$ یک پایه گربنر برای I نیست زیرا $LT(Spoly(f_1, f_2)) = .\langle x^3 - 2xy, x^2y - -2y^2 + x \rangle$ دقت می‌کنیم $G = \{f_1, f_2\}$ یک پایه گربنر برای I نیست، زیرا $-x^2 \notin \langle LT(f_1), LT(f_2) \rangle$ پس G را به $LT(Spoly(f_1, f_2)) = -x^2 \notin \langle LT(f_1), LT(f_2) \rangle$ اضافه $f_3 = Spoly(f_1, f_2) = -x^2$

می‌کنیم. لذا $G = \{f_۱, f_۲, f_۳\}$ داریم

$$Spoly(f_۱, f_۳) = (x^۳ - ۳xy) - (-x)(-x^۲) = -۲xy.$$

اما $rem(-۲xy, G) \neq ۰$. بنابراین $f_۴ = -۲xy$ را به G اضافه میکنیم. همچنین

$$Spoly(f_۲, f_۳) = (x^۲y - ۲y^۲ + x) - (-y)(-x^۲) = -۲y^۲ + x$$

که $rem(-۲y^۲+x, G) \neq ۰$ پس $f_۵ = -۲y^۲+x$ را به G اضافه می‌کنیم. بنابراین $G = \{f_۱, f_۲, f_۳, f_۴, f_۵\}$ محاسبات نشان می‌دهند برای هر $۱ \leq i, j \leq ۵, rem(Spoly(f_i, f_j), G) = ۰$. در نتیجه بنابر محک بوخبرگر مجموعه G یک پایه گربنر برای I است.

فصل ۴

راهبردهای انتخاب زوج در الگوریتم بوخبرگر

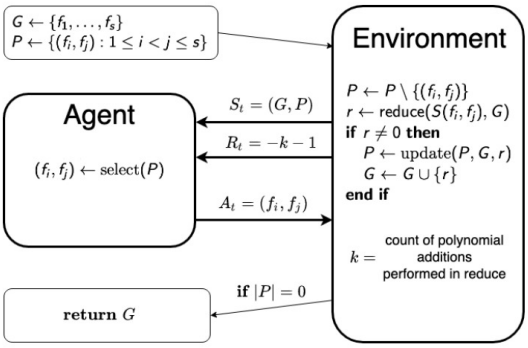
در الگوریتم بوخبرگر چندین نقطه وجود دارد که به انتخاب‌هایی بستگی دارد که بر صحت عملکرد تاثیر نمی‌گذارد، اما می‌تواند تاثیر قابل توجهی بر عملکرد داشته باشد. هر پیاده سازی واقعی الگوریتم در یک سیستم جبر رایانه‌ای نیازمند مشخص کردن برخی راهبردها یا اکتشافات برای این انتخاب‌ها است. اکتشافاتی که در حال حاضر در پیاده‌سازی‌های پیشرفته استفاده می‌شود بهینه نیستند و عمدتاً بر اساس شهود انسانی و نتایج روی مجموعه‌ای از مسائل معیارسنجی خاص بنا شده‌اند. در این فصل بر روی راهبرد انتخاب زوج که یک انتخاب کلیدی در الگوریتم بوخبرگر است، تمرکز می‌کنیم. ما ابتدا نشان می‌دهیم که مشکل انتخاب زوج به طور طبیعی در چارچوب یادگیری تقویتی جای می‌گیرد و توصیف دقیقی از مسئله یادگیری تقویتی مربوطه را ارائه می‌دهیم. سپس نشان می‌دهیم که یک مدل شبکه عصبی ساده برای انتخاب زوج، که با بهینه‌سازی سیاست نزدیک آموزش دیده است، از راهبردهای انتخاب پیشرفته در توزیع‌های ایده‌آل دوجمله‌ای تصادفی غیرهمگن ۹ تا ۳۷ درصد عملکرد بهتری دارد. در حالی که تعمیم و تفسیر این مدل چالش‌هایی را به همراه دارد، این نشان‌دهنده مسیری است که در آن پیشرفت‌های مدرن در یادگیری ماشین می‌تواند عملکرد الگوریتم‌های حیاتی در محاسبات نمادین را بهبود بخشد. در نهایت، به طور خلاصه چند ساختار شبکه دیگر را برای این مسئله در نظر می‌گیریم.[۶، ۴]

۱.۴ مسئله یادگیری تقویتی

الگوریتم بوخبرگر در فصل سوم معرفی شد، با تمرکز بر استراتژی‌های انتخاب زوج در بخش یادگیری تقویتی در فصل دوم مورد بحث قرار گرفت، با فرآیندهای تصمیم گیری مارکوف که در بخش ۱.۱.۲ توصیف شد. در این بخش، ما سرانجام این دو موضوع را با مدل‌سازی الگوریتم بوخبرگر به عنوان یک فرایند تصمیم‌گیری مارکوف ترکیب می‌کنیم که در آن یک عامل با محیط برای انجام انتخاب زوج تعامل می‌کند.

ساختار الگوریتم بوخبرگر را به یاد بیاورید، شکل ۱.۴ خلاصه‌ای از تفکیک این الگوریتم به عامل و محیط را نشان می‌دهد. ایده اصلی این است که هر بار عبور از حلقه while یک گام زمانی است. در گام زمانی t ، حالت محیط $S_t = (G_t, P_t)$ از مجموعه تولید کننده فعلی $G_t = G$ و مجموعه زوج فعلی $P_t = P$ تشکیل شده است. عامل باید یک زوج را از مجموعه فعلی انتخاب کند، بنابراین مجموعه اعمال مجاز $\mathcal{A}_t = P_t$ است.

هنگامی که عامل یک عمل $A_t \in \mathcal{A}_t$ را انتخاب می‌کند، محیط با حذف زوج از مجموعه زوج، کاهش S- چندجمله‌ای مربوطه و در صورت لزوم به‌روزرسانی مجموعه مولد و زوج به‌روزرسانی می‌شود. پس از به‌روزرسانی محیط، عامل پاداش R_t را دریافت می‌کند که منفی ۱ برابر تعداد اضافه شدن چندجمله‌ای در کاهش زوج A_t ، از جمله تفریق منفردی که S- چندجمله‌ای را تولید می‌کند، است. این همان پروکسی مستقل از سخت افزار برای هزینه محاسباتی است که برای ارزیابی استراتژی های انتخاب استفاده کردیم. این به عنوان هزینه عمل رد می‌شود زیرا اکنون آن را به عنوان هزینه عمل در نظر می‌گیریم. بنابراین فضای حالت $\mathcal{S}$ مجموعه ای از چندجمله‌ای‌ها و زوج های بین آنها است. فضای عمل $\mathcal{A}$ شامل زوج‌های (f_i, f_j) است، با مجموعه متغیر اعمال موجود در هر حالت که توسط P_t داده شده است. دینامیک انتقال p از مارکوف عبارتند از:



شکل ۱.۴: مسئله یادگیری تقویتی الگوریتم بوخبرگر

عملکرد محیط بر اساس مجموعه ای از قوانین از پیش تعریف شده برای عملیات کاهش و بروزرسانی چندجمله‌ای‌ها در طی محاسبه، به صورت قطعی (قابل پیش‌بینی) است. به عبارت دیگر، خروجی هر عمل از پیش تعیین شده و تحت تاثیر عوامل تصادفی قرار نمی‌گیرد. توزیع اولیه (ρ) توزیعی بر روی مجموعه‌هایی از مولدهای ایده‌آل اولیه و زوج‌های متناظر است. این توزیع می‌تواند یک مجموعه شروع ثابت باشد، اما در آزمایشات ما یک توزیع ایده‌آل خواهد بود.

محیط با نمونه برداری از توزیع اولیه و مقداردهی اولیه مجموعه زوج‌ها، شروع به کار می‌کند. توجه داشته باشید که هنگام استفاده از استراتژی‌های حذف زوج (همانطور که برای همه نتایج در این پروژه وجود دارد)، مجموعه اولیه زوج‌ها ممکن است مجموعه کامل $\{(f_i, f_j) : ۱ \leq i < j \leq s\}$ نباشد.

هر مسیر $\tau = (s_۰, a_۰, r_۱, s_۱, \ldots, r_T, s_T)$ توالی ای از گام‌ها در الگوریتم بوخبرگر است و زمانی که مجموعه زوج ها خالی باشد، الگوریتم با یک پایه گربنر خاتمه می‌یابد و در یک حالت نهایی به پایان می‌رسد.

هدف عامل به حداکثر رساندن "بازده مورد انتظار" E در طول یک مسیر است که با فرمول

$$\mathbb{E}[\sum_{t=۱}^{T} \gamma^{t-۱} r_t]$$

نمایش داده می‌شود. که در آن $۰ \leq \gamma \leq ۱$ یک عامل تخفیف است. با $\gamma = ۱$ ، این معادل کمینه کردن تعداد مورد انتظار اضافه شدن چندجمله‌ای برای تولید پایه گربنر است. این مسئله از دیدگاه یادگیری تقویتی چندین چالش جالب را ایجاد می‌کند

۱) فضای حالت نامحدود: فضای حالت در چند بعد نامحدود است (تعداد متغیرها، درجه و طول مولدها، تعداد مولدها، اندازه ضرایب) و می‌تواند در طول یک تکرار به طور قابل توجهی تغییر کند.

۲) فضای عمل پویا: اندازه مجموعه عمل با هر گام زمانی تغییر می‌کند و می‌تواند بسیار بزرگ یا بسیار کوچک باشد.

۳) واریانس بالای دشواری: می‌تواند در مقدارهای اولیه مختلف از یک توزیع، واریانس بالایی در سختی وجود داشته باشد.

محیط همچنین می‌توانست آزمایش‌های تقسیم تک‌جمله‌ای یا محاسبات حذف زوج را اندازه‌گیری و جریمه کند. اما برای سادگی، چنین قابلیت‌هایی در این پروژه پیاده‌سازی نشده است. استراتژی‌های مختلف برای زیرمجموعه‌های کاهش و بروزرسانی می‌توانند مسئله را تغییر دهند. گزینه‌های موجود در این زیرمجموعه‌ها

نیز می‌توانند به عنوان عملگر برای عامل در نظر گرفته شوند. به عبارت دیگر، می‌توان کل فرایند تصمیم‌گیری در زیرمجموعه‌های کاهش و بروزرسانی را خود به عنوان مسائل جداگانه یادگیری تقویتی در نظر گرفت.

۲.۴ ساختار حالت و شبکه

اگرچه در بخش قبلی، پیاده‌سازی الگوریتم بوخبرگر به عنوان یک مسئله یادگیری تقویتی از نظر تئوری کافی به نظر می‌رسد، اما ملاحظات عملی استفاده از آن را دشوار می‌کنند. به طور خاص، تغییر شکل حالت و نامحدود بودنِ ابعاد برخی جوانب، به کارگیری شبکه‌های عصبی برای حل این مسئله را با مشکل مواجه می‌کند. برای آموزش عامل‌ها، مجبور شدیم انتخاب‌هایی در مورد چگونگی نمایش حالت به مدل انجام دهیم. ادعا نمی‌کنیم که این نمایش، بهترین حالت است. در واقع، توسعه و آزمایش ساختارهای دیگر برای شبکه و حالت، یک پروژه برای آینده خواهد بود.

برای شروع پروژه، ابتدا تعداد متغیرها (n) را ثابت می‌کنیم و فرض می‌کنیم که همه ایده‌آل‌ها حداکثر برحسب n متغیر باشند. این یک محدودیت قوی است، اما برای محدود کردن ابعاد نامحدود حالت در این راستا، ضروری بود.

سپس، هر چندجمله‌ای را به صورت الحاق بردارهای توان برای k جمله اصلی آن نمایش می‌دهیم، که در آن k ثابت دیگری است. به این صورت که جملات چند جمله‌ای را برحسب ترتیب مورد نظرمان مرتب و با K جمله اول آن کار می‌کنیم. در تقریباً همه آزمایش‌ها، $k = ۲$ را انتخاب می‌کنیم که دو جمله اول را نمایش می‌دهد. اگر یک چندجمله‌ای کمتر از k جمله داشته باشد، ورودی‌ها را با صفر پر می‌کنیم.

حالت $S_t = (G_t, P_t)$ را به صورت یک ماتریس نمایش می‌دهیم که سطرهای آن با الحاق بردارهای مربوط به چندجمله‌ای‌های هر زوج به دست می‌آیند. برای n متغیر و p زوج، این منجر به یک ماتریس با ابعاد $p \times ۲kn$ می‌شود. که در مثال بعدی این ماتریس توصیف و تفسیر شده است.

اکنون محیط، به طور جزئی مشاهده می‌شود، زیرا ماتریس شامل ضرایب یا لیست کامل اعضای چندجمله‌ای‌ها نمی‌شود. با این حال، بر اساس تجربه، مهم‌ترین اطلاعات در مورد یک چندجمله‌ای در طول محاسبه یک پایه گربنر، تک‌جمله‌های پیشرو هستند. برای مثال، استراتژی‌های انتخاب درجه و نرمال فقط به تک‌جمله‌های پیشرو در هر زوج نگاه می‌کنند. انتخاب ما برای $k = ۲$ کمی اطلاعات بیشتر را در اختیار قرار می‌دهد. در واقع، اگر همه ورودی‌ها، ایده‌آل‌های دودویی باشند، همه چندجمله‌ای‌ها در محاسبه نهایی دودویی باقی خواهند ماند. با

$k = ۲$ ، عامل همه تک‌جمله‌های درگیر در هر ژنراتور را می‌بیند و فقط ضرایب تصادفی را از دست می‌دهد.

مثال ۱.۲.۴. فرض کنید $n = ۳$ و $k = ۲$ باشد. حالتی را در نظر بگیرید که با $G = \{xy^۶ + ۹y^۲z^۴, z^۴ + ۱۳z, xy^۳ + ۹۱xy^۲\}$ نمایش داده می‌شود و در آن اعضای هر دوجمله‌ای بر اساس ترتیب الفبایی معکوس مدرج مرتب شده‌اند. همچنین، فرض کنید $p = \{(۱, ۲), (۱, ۳), (۲, ۳)\}$ باشد که نشان‌دهنده زوج‌های چندجمله‌ای‌های درگیر است. با نشان دادن هر زوج در یک سطر، ماتریس زیر به دست می‌آید:

$$\begin{bmatrix} ۱ & ۶ & ۰ & ۰ & ۲ & ۴ & ۰ & ۰ & ۴ & ۰ & ۰ & ۱ \\ ۱ & ۶ & ۰ & ۰ & ۲ & ۴ & ۱ & ۳ & ۰ & ۱ & ۲ & ۰ \\ ۰ & ۰ & ۴ & ۰ & ۰ & ۱ & ۱ & ۳ & ۰ & ۱ & ۲ & ۰ \end{bmatrix}$$

برای مثال، سه درایه اول در اولین سطر، $x^۱y^۶z^۰$ را نمایش می‌دهند که اولین عضو در اولین ژنراتور درگیر در اولین زوج است.

با وجود ثابت بودن عرض ماتریس، ارتفاع آن در حالات مختلف تغییر می‌کند. برای مدیریت این ورودی، عامل ما از یک شبکه سیاست استفاده می‌کند که هر سطر را به صورت جداگانه در نظر می‌گیرد و با استفاده از مجموعه‌ای از لایه‌های متراکم ، هر سطر را به یک امتیاز ترجیحی واحد نگاشت می‌کند. این را می‌توان به عنوان یادگیری یک تابع $f : \mathbb{R}^{۲kn} \to \mathbb{R}$ در نظر گرفت که سپس به صورت سطر به سطر بر روی ماتریس اعمال می‌شود. سیاست عامل، که یک توزیع احتمال روی مجموعه زوج‌های فعلی است، بر اساس این امتیازات ترجیحی است. در آزمایش‌های اولیه، به نظر نمی‌رسید که عمق شبکه به طور قابل توجهی بر عملکرد تأثیر بگذارد، بنابراین ما روی معماری پایه زیر توافق کردیم:

$$\boxed{p \times ۲kn} \xrightarrow{denserelu} \boxed{p \times ۱۲۸} \xrightarrow{denselinear} \boxed{p \times ۱} \xrightarrow{softmax} \boxed{p \times ۱}$$

به دلیل سادگی پیاده‌سازی، به‌کارگیری این مدل در یک پیاده‌سازی عملی از الگوریتم بوخبرگر به صورت تئوری آسان خواهد بود. امتیازهای ترجیحی که توسط شبکه تولید می‌شوند، می‌توانند به عنوان کلیدهای مرتب‌سازی برای مجموعه زوج‌ها استفاده شوند. هر زوج تنها یک بار پردازش می‌شود و انتظار می‌رود ضرب ماتریس‌های نسبتاً کوچک در این مدل، سربار اضافی کمی را در یک پیاده‌سازی دقیق به همراه داشته باشد.

با این حال، با توجه به این که عملکرد لحظه‌ای الگوریتم بوخبرگر وابستگی شدیدی به جزئیات پیچیده

پیاده‌سازی دارد، ما به طور انحصاری بر معیارهای مستقل از پیاده‌سازی تمرکز می‌کنیم و آزمایش بهبود عملکرد لحظه‌ای را به کارهای آینده موکول می‌کنیم.

این سازوکار، یک نمونه خاص از یک مسئله یادگیری تقویتی کلی است که در آن، حالت به عنوان مجموعه‌ای از اعمال توصیف می‌شود و هر عمل با یک بردار ویژگی با اندازه ثابت مرتبط است. شبکه ما یاد می‌گیرد که بردارهای ویژگی را به امتیازهای ترجیحی نگاشت کند، اما هرگز نمی‌تواند اعمال را در متن سایر اقداماتی که حالت کامل را تشکیل می‌دهند، ارزیابی کند.

شبکه‌های پیچیده‌تری که می‌توانند از زمینه استفاده کنند در بخش‌های بعد مورد بررسی قرار می‌گیرند، اما ما پیشرفت قابل توجهی را با استفاده از آن‌ها برای مسئله بوخبرگر مشاهده نکردیم.

۳.۴ روش‌های آموزشی

عامل‌های ما با استفاده از بهینه‌سازی سیاست تقریبی (PPO) آموزش داده شدند. آموزش طی یک سری مرحله انجام می‌شود. در هر مرحله، ابتدا اپیزودها را با دنبال کردن سیاست فعلی نمونه‌برداری می‌کنیم، که به معنی تولید ایده‌آل‌ها از یک توزیع ثابت و محاسبه پایه‌های گربنر با انتخاب زوج با نمونه‌برداری از خروجی شبکه فعلی است. مسیرها ذخیره می‌شوند و عملکرد ثبت می‌شود. سپس، تخمین مزیت تعمیم‌یافته (GAE) برای محاسبه مزیت هر عمل در هر مسیر استفاده می‌شود که در طول آن مرحله نرمال می‌شوند. تابع ارزش ما $V(G,P)$ تعداد ضرب‌های چندجمله‌ای مورد نیاز برای تکمیل الگوریتم بوخبرگر با شروع از حالت (G,P) با استفاده از استراتژی درجه) زوج ها دو به دو محاسبه و زوجی انتخاب می‌شود که کمترین s- چندجمله‌ای را دارد.) تخمین می‌زند. محاسبه این تابع در هر حالت هزینه‌بر است اما عملکرد را به طور قابل توجهی بهبود می‌بخشد.

همانطور که در یادگیری تقویتی استاندارد هست، مجموعه‌های ثابت آموزش یا تست وجود ندارد. در عوض، داده‌های آموزشی و آزمایش به صورت آنلاین توسط یک تابع که به طور تصادفی یک ایده‌آل را از توزیع می‌سازد، تولید می‌شوند. اندازه بزرگ توزیع‌ها از هرگونه بیش‌برازش به یک زیرمجموعه خاص از ایده‌آل‌ها جلوگیری می‌کند. برای مثال، حتی با نادیده گرفتن ضرایب، تعداد کل ایده‌آل‌ها در ۳-۲۰-۱۰-وزنی تقریباً $۱۰^{۵۵}$ است. پس از آموزش، عامل‌ها با اجرای روی ایده‌آل‌های جدیدی که به همین روش تولید شده‌اند، آزمایش خواهند شد.

۱.۳.۴ تنظیم ابرپارامترها

قبل از اجرای ارزیابی اصلی، ما یک جستجوی مختصر برای ابرپارامترها را در چهار مرحله انجام دادیم. هر مرحله با توزیع وزنی ۳-۲۰-۱۰ آموزش داده شد، زیرا این توزیع سریعترین اجرای آموزش را ارائه می‌دهد. اجرای ابرپارامترها برای ۱۵۰۰ دوره برای هر مرحله انجام شد و بر اساس عملکرد آموزش، قضاوت شدند.

در مرحله اول، نرخ یادگیری را در $\{۱۰^{-۵}, ۱۰^{-۴}, ۱۰^{-۳}\}$ ، اندازه دسته را در $\{۳۲, ۲۵۶, None\}$ (که در آن None نشان‌دهنده تمام داده‌های دوره در یک دسته است) و اپیزودها در هر دوره را در $\{۱۰, ۱۰۰\}$ تغییر دادیم. این پارامترها بر سرعت آموزش و تعادل بین تولید تجربه جدید و آموزش مکرر بر روی تجربه قدیمی تأثیر می‌گذارند. سه اجرا بر روی هر مجموعه پارامتر انجام شد که در مجموع ۵۴ اجرا بود. نتایج به طور کلی بسیار مشابه بودند، اگرچه زمان‌های آموزش متفاوت بود. نرخ‌های یادگیری

$۱۰^{-۴}$ تمایل به داشتن بهترین عملکرد داشتند و بنابراین برای همه آزمایش‌های بعدی انتخاب شدند. اندازه دسته None معمولاً بهترین بود، اما تفاوت‌ها جزئی بود و زمان آموزش و استفاده از حافظه بسیار بدتر بود، بنابراین ما در عوض اندازه دسته ۵۱۲ را ثابت کردیم که بالاترین مقداری بود که از نظر محاسباتی معقول بود. استفاده از ۱۰۰ اپیزود در هر دوره به طور کلی بهتر از ۱۰ بود، بنابراین این مقدار نیز ثابت شد.

در مرحله دوم، پارامتر λ را در $\{۰/۹۷, ۱/۰\}$ و ضریب تخفیف γ را در $\{۰/۹۹, ۱/۰\}$ تغییر دادیم. این پارامترها بر محاسبه مزیت‌ها تأثیر می‌گذارند. سه اجرا بر روی هر مجموعه پارامتر انجام شد که در مجموع ۱۲ اجرا بود. مقادیر $\lambda = ۰/۹۷$ و $\gamma = ۰/۹۹$ به طور مداوم بهترین بودند، بنابراین از این مقادیر برای همه آزمایش‌های بعدی استفاده کردیم. در مرحله سوم، پارامتر ϵ را در $\{۰/۱, ۰/۲\}$ و محدودیت KL- واگرایی را در $\{۰/۰۱, ۰/۱\}$ تغییر دادیم. این پارامترها اثر محدودکننده بر اندازه گام در PPO را تغییر می‌دهند. سه اجرا بر روی هر مجموعه پارامتر انجام شد که در مجموع ۱۲ اجرا بود. تفاوت‌ها ناچیز بود، با احتمال بهبود جزئی با استفاده از $\epsilon = ۰/۲$ ، بنابراین این مقدار و محدودیت KL- واگرایی برابر با ۰/۰۱ انتخاب شد.

در مرحله چهارم، فقط شکل شبکه را تغییر دادیم. شبکه‌های تک لایه پنهان با ۴، ۳۲، ۱۲۸ و ۲۵۶ واحد و شبکه‌های دو لایه پنهان با ۴، ۳۲، ۱۲۸ و ۲۵۶ واحد پنهان در هر لایه آزمایش شدند. سه اجرا بر روی هر شبکه انجام شد که در مجموع ۲۴ اجرا بود. نتایج بهبود قابل توجهی در استفاده از حداقل ۳۲ واحد پنهان و بدون تفاوت عمده بین یک و دو لایه پنهان یا بین ۱۲۸ و ۲۵۶ واحد پنهان نشان داد. بنابراین تصمیم گرفتیم ساختار شبکه را به یک لایه پنهان با ۱۲۸ واحد ثابت کنیم. بسیاری از این جستجوهای هایپرپارامتر به دلیل

hyperparameter	value
γ for GAE	0.99
λ for GAE	0.97
ϵ for PPO	0.2
optimizer	Adam
learning rate	0.0001
batch size	512
max policy updates per epoch	40
policy KL-divergence limit	0.01
hidden layers	[128]
value function	degree
epochs	3000
episodes per epoch	100
max episode length	500

شکل ۲.۴: ابرپارامترها برای اجرای آموزش اولیه

محدودیت‌های محاسباتی محدود بودند و تنظیمات بیشتر می‌تواند منجر به عملکرد بهتر شود. به عنوان مثال، پس از محاسبه نتایج اصلی، جستجوی دقیق‌تر مرحله دوم با $\gamma =$ و $\lambda = \{۰/۹, ۰/۹۵, ۰/۹۷, ۰/۹۹, ۱/۰\}$ $\{۰/۹, ۰/۹۵, ۰/۹۷, ۰/۹۹, ۱/۰\}$ نشان داد که $\lambda = ۰/۹$ منجر به بهبود ۲درصدی عملکرد در طول آموزش نسبت به انتخاب اولیه $\lambda = ۰/۹۷$ می‌شود، اگرچه γ ، که در آن ۰/۹۹ هنوز به طور مداوم بهترین بود، به طور کلی تأثیرگذارتر بود. همچنین مهم است که توجه داشته باشیم که بهترین عملکرد آموزش داده شده ما نسبت به خطوط پایه در نه توزیع مختلف در ۳-۲۰-۱۰-وزنی رخ داد، همان توزیعی که در آن جستجوهای هایپرپارامتر را انجام دادیم. شاید بتوان با استفاده از هایپرپارامترهای مناسب در توزیع‌های خودشان، عملکرد بهتری در موارد دیگر به دست آورد.

۲.۳.۴ اجراهای آموزشی

برای هر تنظیم، پنج اجرای کامل را با استفاده از پارامترهای جدول ۲.۴ انجام دادیم. وزن‌های مدل هر ۱۰۰ دوره ذخیره می‌شد و یک مدل واحد از هر تنظیم بر اساس بهترین عملکرد آموزشی صاف شده از بین اجراها و نقاط ذخیره موجود انتخاب می‌شد. زمان‌های آموزش بین ۱۱ تا ۱۰۱ ساعت در هر اجرا با استفاده از Opteron ۶۳۷۶ با ۵۱۲ گیگابایت رم متغیر بود.

شکل‌های ۳.۴، ۴.۴ و ۵.۴ عملکرد عامل‌های نمونه از هر توزیع را در طول آموزش نشان می‌دهند. تمام عامل‌ها در توزیع‌های وزنی به سرعت به نتایج بهتری نسبت به بهترین استراتژی‌های شناخته شده دست یافتند و

عامل‌ها در توزیع‌های یکنواخت نیز در نهایت از آن‌ها پیشی می‌گیرند. با این حال، عامل‌ها در توزیع‌های حداکثر در بهترین حالت به عملکرد مشابه با بهترین استراتژی‌ها نزدیک می‌شوند. برخی از عامل‌ها هنوز در حال بهبود هستند، اما به نظر می‌رسد که اکثر آن‌ها در بقیه دوره آموزش به عملکرد ثابت یا بسیار کند افزایشی رسیده‌اند. به جز ۵-۵-۱۰-حداکثر، عملکرد آموزشی در دانه‌های تصادفی مختلف به طرز شگفت‌آوری ثابت بود، شاید به این دلیل که واریانس بالای سختی ایده‌آل‌ها اثر تنظیم‌کننده دارد.

۴.۴ نتایج

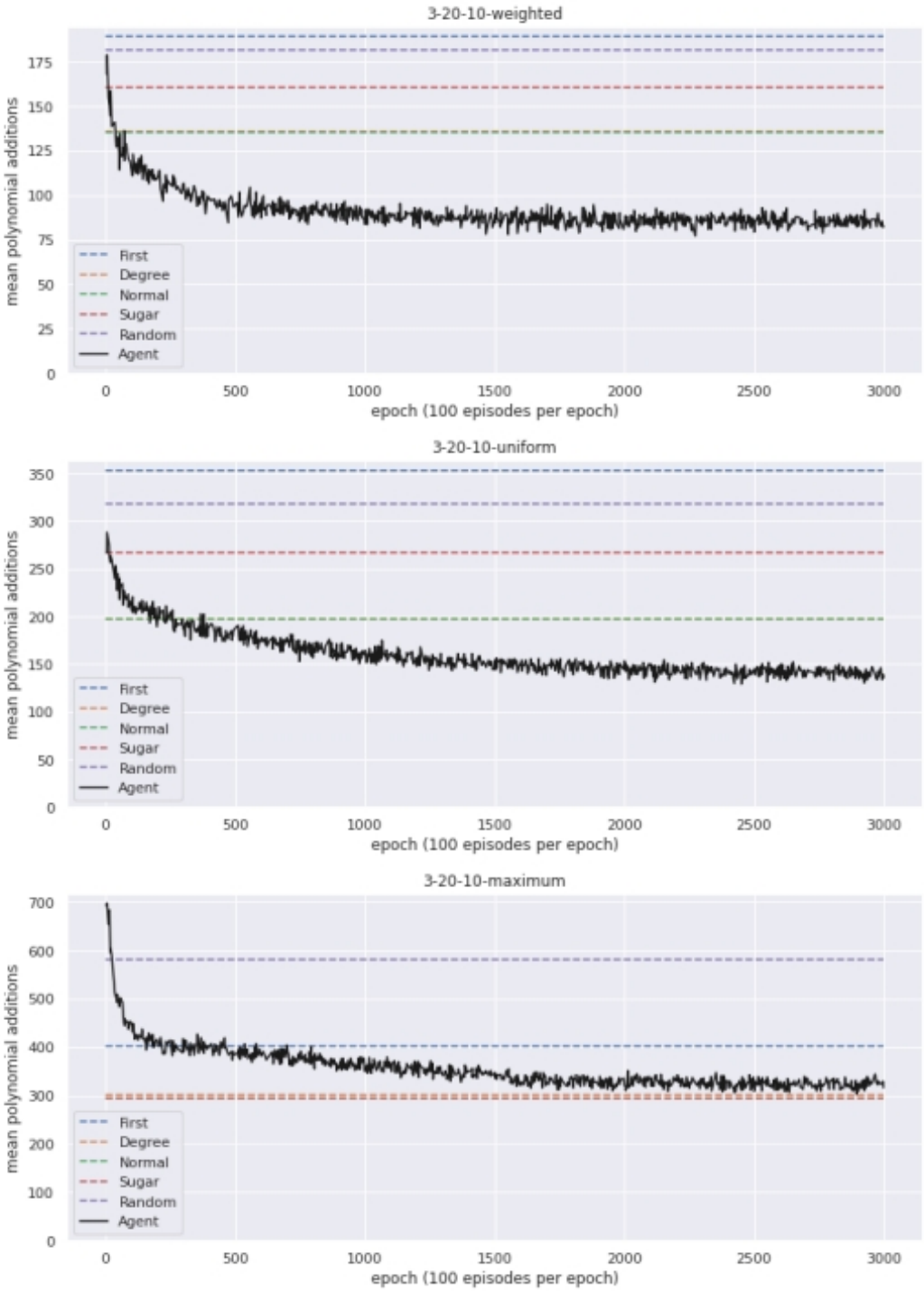
عملکرد متوسط در طول آموزش که در شکل‌های ۳.۴، ۴.۴ و ۵.۴ مشاهده می‌شود، از قبل نشان می‌دهد که عامل‌های ما استراتژی‌هایی را آموخته‌اند که برتری نسبت به اکتشافات (روش‌های) پیشرفته در حوزه‌های وزن‌دار و یکنواخت دارند. در این بخش، ما این نتایج را بیشتر کمی می‌کنیم و سعی می‌کنیم بفهمیم که شبکه‌ها چه چیزی را کشف کرده‌اند. همچنین بررسی می‌کنیم که استراتژی‌های آن‌ها چقدر در برابر توزیع‌های مختلف پس از آموزش مقاوم هستند.

۱.۴.۴ عملکرد عامل

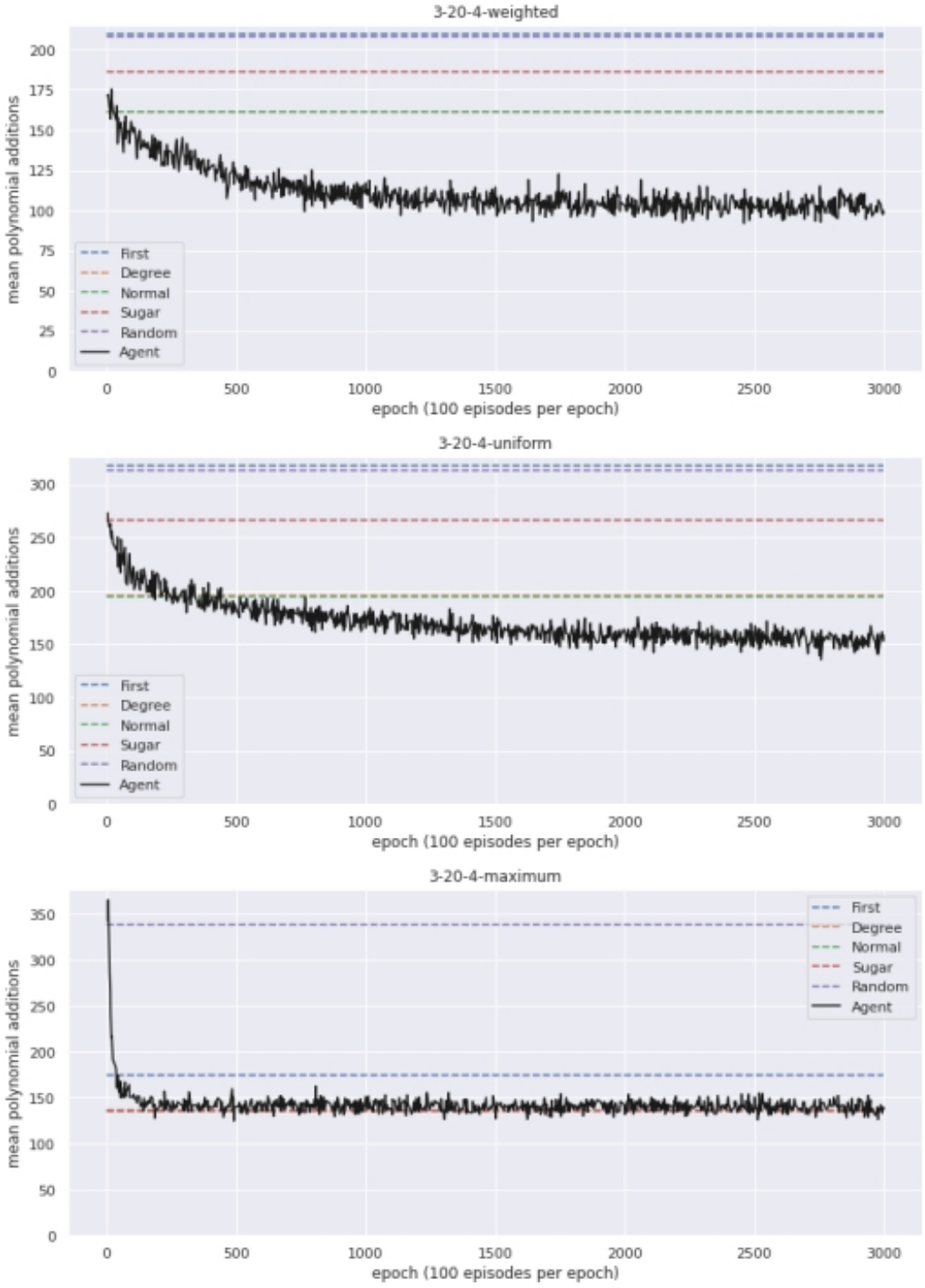
جدول‌های ۶.۴، ۷.۴ و ۸.۴ عملکرد نهایی عامل‌هایی را که روی نه توزیع اصلی آموزش دیده‌اند، نشان می‌دهند. عامل‌ها در توزیع‌های وزن‌دار به‌طور میانگین بین ۲۹ تا ۳۷ درصد کمتر از بهترین استراتژی‌های معیار، جمع‌های چندجمله‌ای انجام می‌دهند. همچنین آن‌ها موفق به کاهش انحراف استاندارد در تعداد جمع‌های چندجمله‌ای بین ۳۶ تا ۴۶ درصد شده‌اند. این نتایج حاکی از آن است که عامل‌ها توانسته‌اند استراتژی‌های موثری را برای حل مسائل در این نوع توزیع‌ها بیاموزند.

بهبود عملکرد در توزیع‌های یکنواخت که عموماً با مسائل دشوارتری همراه هستند، نسبت به استراتژی‌های پایه همچنان مشهود است. عامل‌های آموزش‌دیده در این توزیع‌ها، ۹ تا ۲۹ درصد کمتر جمع چندجمله‌ای انجام می‌دهند که نشان‌دهنده یادگیری آنهاست، اما به اندازه توزیع‌های وزن‌دار چشمگیر نیست.

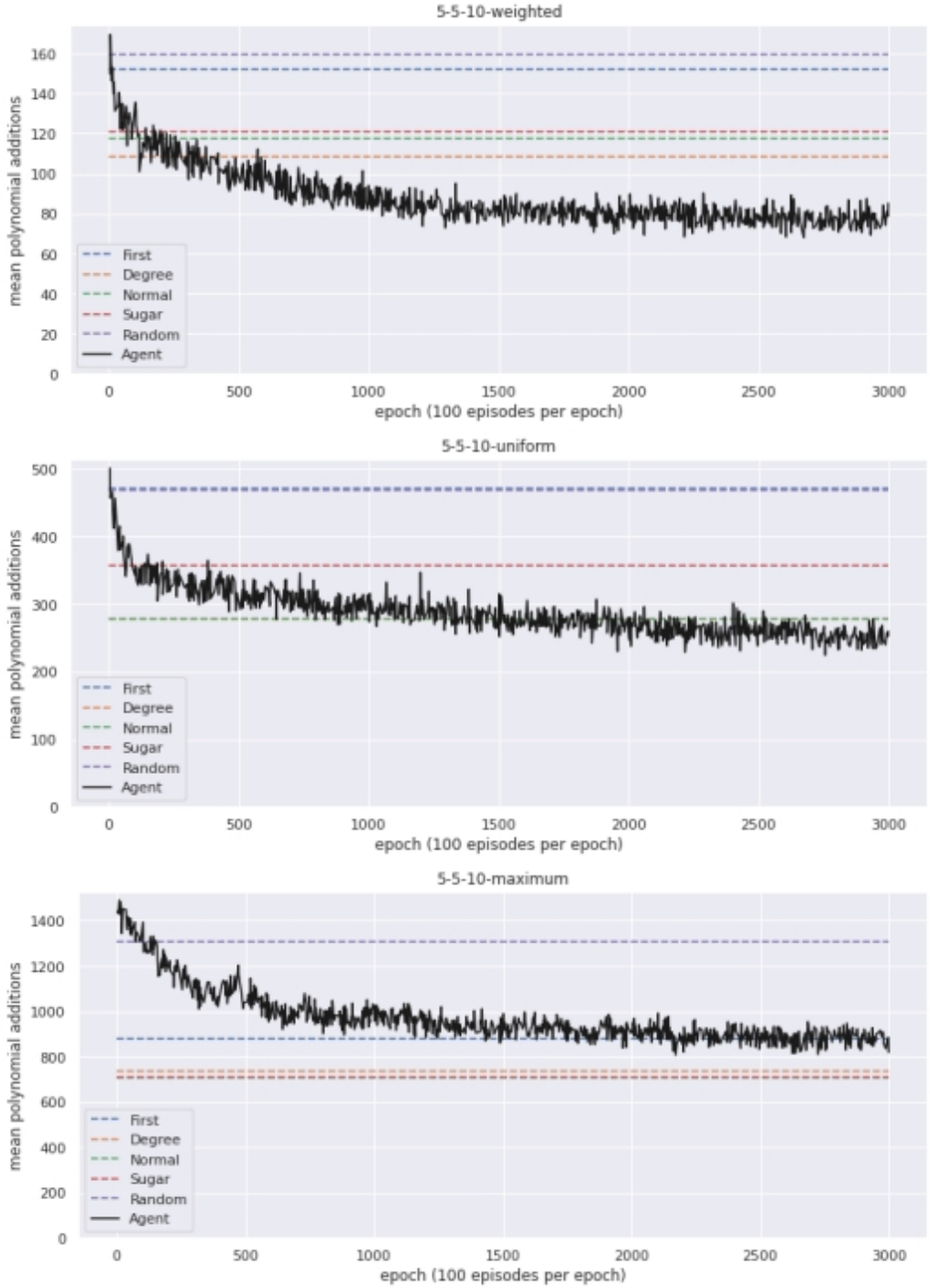
عملکرد در توزیع‌های حداکثر ۹۴ که به عنوان دشوارترین توزیع‌ها شناخته می‌شوند، از حالت تصادفی بهتر است، اما همچنان از بهترین استراتژی‌های معیار پیشی نمی‌گیرد. در واقع، عملکرد آن‌ها بین ۳ تا ۲۲ درصد ضعیف‌تر است. این نتایج نشان می‌دهد که عامل‌ها برای حل مسائل در این توزیع‌های بسیار دشوار، نیاز به



شکل ۳.۴: میانگین تعداد اضافات چندجمله‌ای در هر دوره از آموزش روی توزیع های ۳-۲۰-۱۰. خطوط چین دار نشان دهنده عملکرد متوسط استراتژی های معیار روی ۱۰۰۰۰ ایده آل تصادفی است. هرچه تعداد کمتر باشد، بهتر است.



شکل ۴.۴: میانگین تعداد اضافات چندجمله‌ای در هر دوره از آموزش روی توزیع‌های ۳-۲۰-۴. خطوط چین‌دار نشان‌دهنده عملکرد متوسط استراتژی‌های معیار روی ۱۰،۰۰۰ ایده‌آل تصادفی است. هرچه تعداد کمتر باشد، بهتر است.



شکل ۵.۴: میانگین تعداد اضافات چندجمله‌ای در هر دوره از آموزش روی توزیع‌های ۵-۵-۱۰. خطوط چین‌دار نشان‌دهنده عملکرد متوسط استراتژی‌های معیار روی ۱۰،۰۰۰ ایده‌آل تصادفی است. هرچه تعداد کمتر باشد، بهتر است.

Strategy	3-20-10-weighted	3-20-10-uniform	3-20-10-maximum
First	189.07[74.17]	353.05[115.30]	401.38[105.71]
Degree	135.67[50.88]	197.16[57.02]	300.86[74.68]
Normal	135.15[50.61]	197.32[57.95]	294.45[73.03]
Sugar	160.40[66.22]	266.80[88.56]	294.45[73.03]
Random	181.30[68.01]	318.05[102.32]	580.36[166.72]
Agent	84.85[27.15]	139.44[42.65]	325.37[79.73]
Improvement	37%[46%]	29%[25%]	-11%[-9%]

شکل ۶.۴: عملکرد عامل در توزیع‌های ۳–۲۰–۱۰ روی ۱۰۰۰۰ نمونه. مقادیر جدول میانگین [انحراف استاندارد] تعداد جمع‌های چندجمله‌ای را نشان می‌دهد. هر ستون نشان‌دهنده یک عامل مجزا است و بهبود عملکرد نسبت به بهترین استراتژی معیار است.

آموزش بیشتر یا بهبود الگوریتم دارند.

شکل‌های ۱۰.۴ و ۹.۴ جزئیات بیشتری در مورد توزیع تعداد جمع‌های چندجمله‌ای برای هر مسئله (ایده‌آل) ارائه می‌دهند. با بررسی این شکل‌ها می‌توان الگوهای بیشتری در عملکرد عامل‌ها و مقایسه دقیق‌تر آنها با استراتژی‌های پایه استخراج کرد.

۲.۴.۴ انواع مدل

ما متوجه شدیم که عملکرد مدل در موارد زیر کاهش می‌یابد:

۱) زمانی که تنها مجاز باشد جمله پیشرو هر زوج را ببیند.

۲) زمانی که تابع ارزش در طول آموزش حذف شود.

۳) زمانی که از شبکه‌ای با تنها ۴ واحد پنهان استفاده شود.

برای مقایسه عملکرد در توزیع وزنی ۳–۲۰–۱۰، به جدول ۱۱.۴ مراجعه کنید. با این حال، تمام مدل‌های آموزش‌دیده همچنان عملکرد بهتری نسبت به بهترین استراتژی معیار که در این مورد با میانگین عملکرد ۱۵.۱۳۵ (از جدول ۶.۴) استراتژی نرمال می‌باشد، به دست می‌آورند.

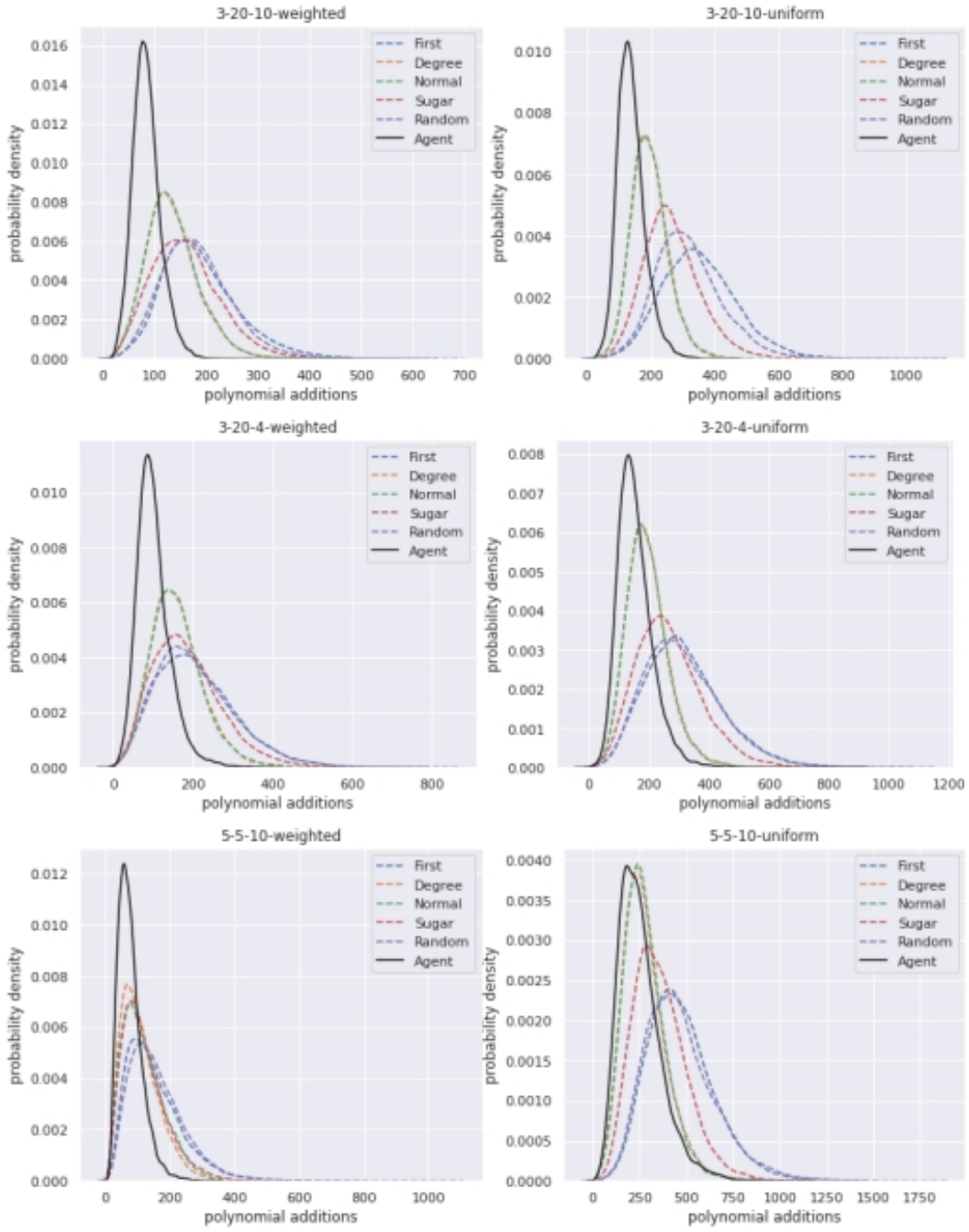
اگرچه مدل بدون تابع ارزش، ضعیف‌ترین عملکرد را در جدول ۱۱.۴ دارد، اما شکل ۱۲.۴ نشان می‌دهد که این مدل همچنان در ۳۰۰۰ دوره آموزشی در حال بهبود است. تابع ارزش عمدتا برای افزایش سرعت آموزش، به ویژه در دوره‌های اولیه که عامل به طور تصادفی حدس می‌زند، مفید است.

Strategy	3-20-4-weighted	3-20-4-uniform	3-20-4-maximum
First	209.43[101.08]	317.40[128.90]	174.40[73.47]
Degree	161.20[66.04]	195.61[70.80]	136.26[47.23]
Normal	161.18[67.47]	194.56[71.34]	135.41[46.91]
Sugar	185.91[87.86]	266.33[109.08]	135.41[46.91]
Random	207.85[100.35]	312.92[129.71]	337.98[182.42]
Agent	102.67[42.75]	154.07[58.71]	139.54[48.59]
Improvement	36%[35%]	21%[17%]	-3%[-4%]

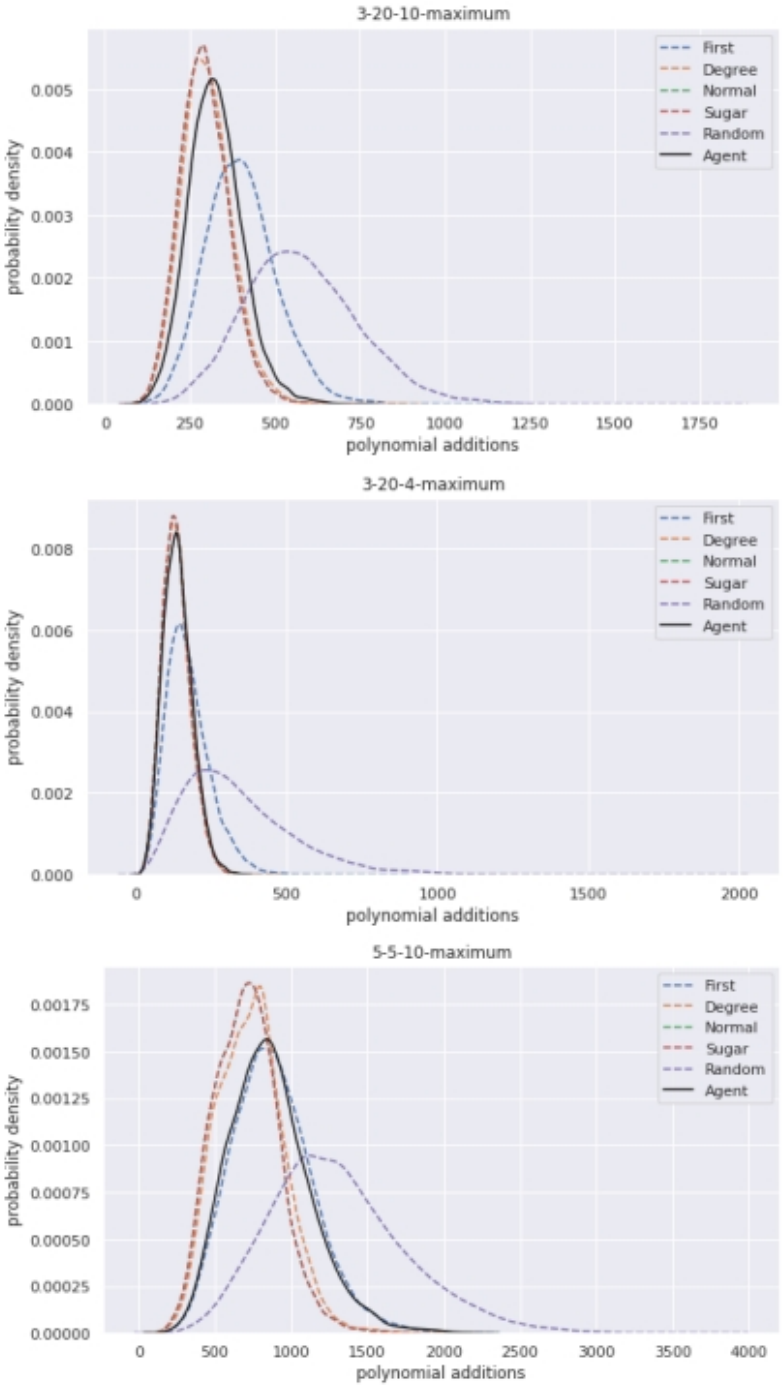
شکل ۷.۴: عملکرد عامل در توزیع‌های ۳-۲۰-۴ روی ۱۰۰۰۰ نمونه. مقادیر جدول میانگین [انحراف استاندارد] تعداد جمع‌های چندجمله‌ای را نشان می‌دهد. هر ستون نشان‌دهنده یک عامل مجزا است و بهبود عملکرد نسبت به بهترین استراتژی معیار است.

Strategy	5-5-10-weighted	5-5-10-uniform	5-5-10-maximum
First	152.06[86.66]	470.96[176.94]	878.11[268.28]
Degree	108.45[58.72]	277.84[118.13]	735.96[220.53]
Normal	117.46[64.96]	278.17[113.46]	707.46[210.50]
Sugar	120.84[68.35]	356.95[142.24]	707.46[210.50]
Random	159.46[85.51]	468.42[190.03]	1306.78[448.69]
Agent	76.87[37.87]	253.63[116.70]	861.55[270.96]
Improvement	29%[36%]	9%[-3%]	-22%[-29%]

شکل ۸.۴: عملکرد عامل در توزیع‌های ۵-۵-۱۰ روی ۱۰۰۰۰ نمونه. مقادیر جدول میانگین [انحراف استاندارد] تعداد جمع‌های چندجمله‌ای را نشان می‌دهد. هر ستون نشان‌دهنده یک عامل مجزا است و بهبود عملکرد نسبت به بهترین استراتژی معیار است.



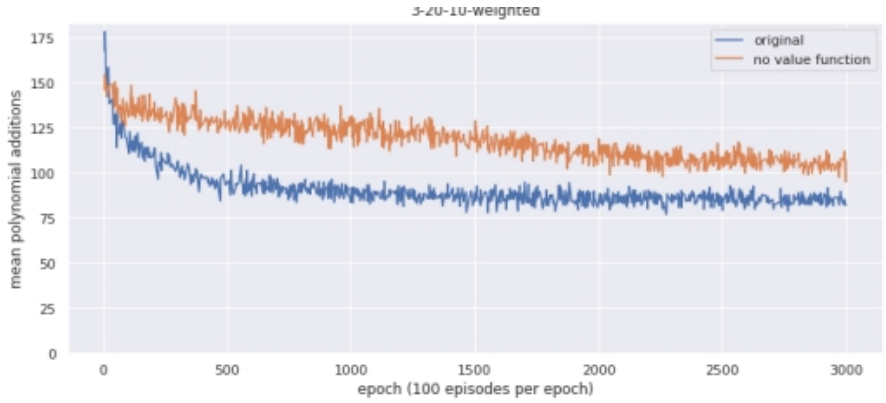
شکل ۹.۴: تخمین‌های چگالی هسته‌ای برای تعداد جمع‌های چندجمله‌ای به ازای هر ایده‌آل در توزیع‌های یکنواخت و وزن‌دار برای عامل کاملاً آموزش‌دیده و استراتژی‌های معیار روی ۱۰۰۰۰ نمونه. هرچه اعداد کوچکتر باشند، بهتر است.



شکل ۱۰.۴: تخمین‌های چگالی هسته‌ای برای تعداد جمع‌های چندجمله‌ای به ازای هر ایده‌آل در توزیع‌های ماکزیمم برای عامل کاملاً آموزش‌دیده و استراتژی‌های معیار روی ۱۰،۰۰۰ نمونه. هرچه اعداد کوچکتر باشند، بهتر است.

Agent	Polynomial Additions	Drop
original	84.85[27.15]	-
lead monomial only	90.60[29.52]	7%[9%]
no value function	103.89[34.40]	22%[27%]
4 unit hidden layer	88.70[28.66]	5%[6%]

شکل ۱۱.۴: این جدول نتایج اجرای مدل‌های مختلف را نشان می‌دهد. در هر سطر، میانگین [انحراف استاندارد] تعداد جمع‌های چندجمله‌ای انجام شده توسط مدل و همچنین افت عملکرد نسبت به مدل اصلی بر روی ۱۰۰۰۰ نمونه ایده‌آل از توزیع ۳-۲۰-۱۰ وزن‌دار نمایش داده شده است.



شکل ۱۲.۴: میانگین تعداد جمع‌های چندجمله‌ای انجام شده در هر دوره از آموزش را برای مدل اصلی و مدل بدون تابع ارزش در توزیع وزن‌دار ۳-۲۰-۱۰ مقایسه می‌کند. به خاطر داشته باشید که در این زمینه، کم‌تر بودن تعداد جمع‌های چندجمله‌ای نشان‌دهنده عملکرد بهتر است.

۳.۴.۴ خروجی شبکه عصبی آموزش دیده

به طور کلی، تفسیر نحوه تولید خروجی توسط یک شبکه عصبی آموزش‌دیده، به خصوص برای شبکه‌هایی با لایه‌های زیاد و انواع اتصالات، کار دشواری است. در این مورد خاص، ساختار شبکه‌ای که ما استفاده کرده‌ایم، برای هر زوج یک مقدار اسکالر خروجی می‌دهد. سپس این مقدار وارد یک تابع نرم‌سازی روی مجموعه زوج‌ها می‌شود تا توزیع احتمالی برای زوج انتخاب‌شده را تعیین کند. به این معنی که می‌توانیم مقدار اسکالر اختصاص داده‌شده به هر زوج را به عنوان امتیاز ترجیح در نظر بگیریم، به طوری که امتیاز ترجیح بالاتر تقریباً با احتمال بسیار بیشتر انتخاب شدن زوج مطابقت دارد. با بررسی این امتیازهای ترجیحی، چندین مؤلفه استراتژی‌های عامل را شناسایی کرده‌ایم:

۱) عامل‌ها زوج‌هایی را ترجیح می‌دهند که توسط معیار درجه انتخاب می‌شدند به عبارتی درجه زوج از بقیه کمتر باشد. (درجه $lcm(LM(f_i), LM(f_j))$ کوچک‌تر است).

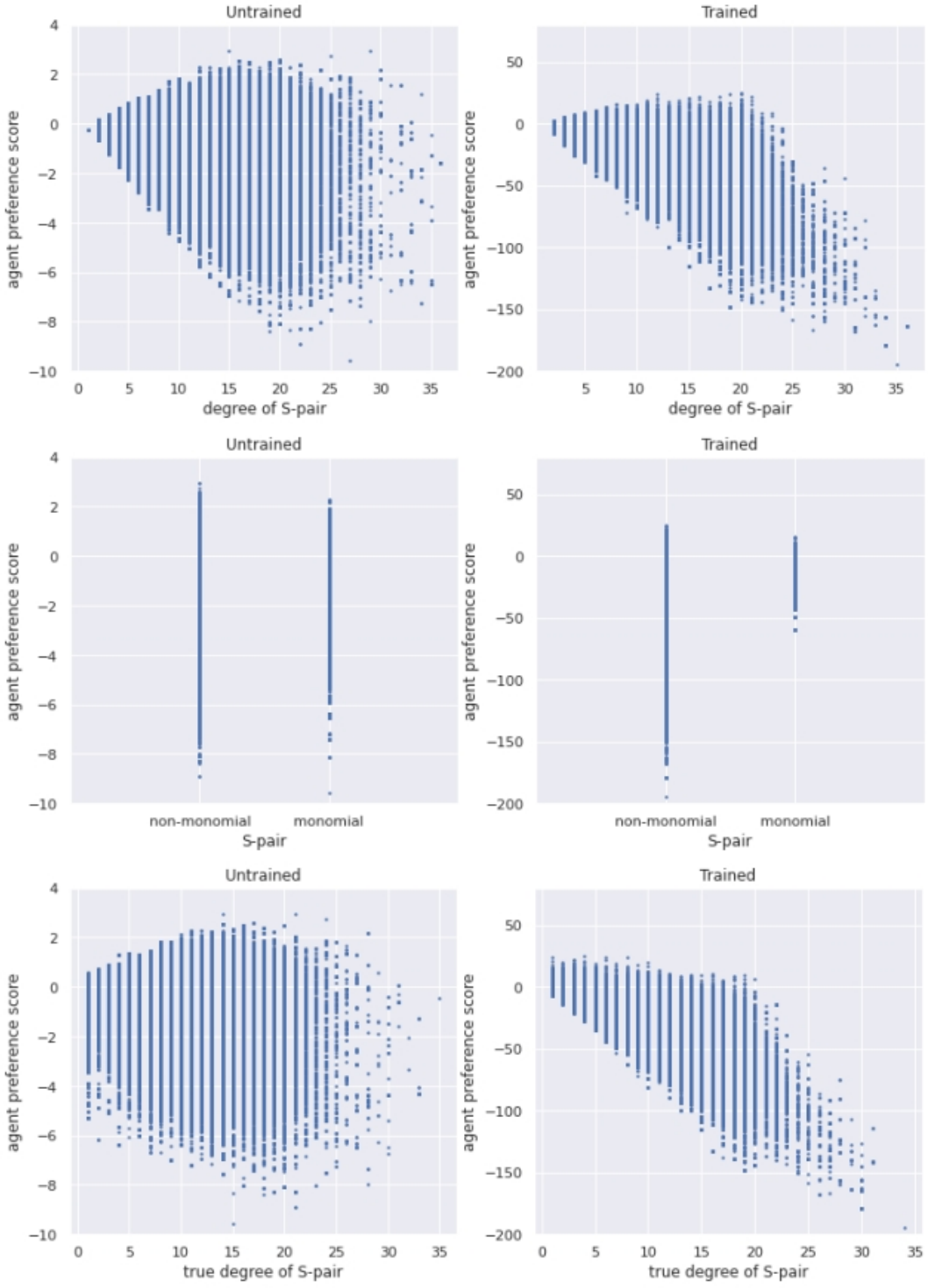
۲) عامل‌ها زوج‌هایی را ترجیح می‌دهند که s- چندجمله‌ای آن‌ها یک تک‌جمله‌ای باشد. ($S(f_i, f_j) = cx^a$).

۳) عامل‌ها زوج‌هایی را ترجیح می‌دهند که s- چندجمله‌ای آن‌ها کم‌درجه باشد (یعنی درجه $LM(S(f_i, f_j))$ که ما آن را درجه حقیقی زوج می‌نامیم، کوچک باشد).

توجه داشته باشید که این ویژگی را معمولاً می‌توان تنها با دسترسی به دو جمله اول تولیدکننده‌های چندجمله‌ای محاسبه کرد و همیشه هنگامی که با ایده‌آل‌های تولید شده توسط دوجمله‌ای‌ها سروکار داریم، با دو جمله اول قابل محاسبه است. تغییرات در امتیاز ترجیح برای این ویژگی‌ها پس از آموزش، در شکل ؟؟ قابل مشاهده است. عامل اولیه آموزش‌ندیده طیف وسیع‌تری از امتیاز ترجیح را برای درجات بالاتر (جمله‌ی پیشرو) دارد، زیرا بردار صفی‌ای که نشان‌دهنده زوجی با درجه بزرگ است، بزرگی بیشتری دارد و نتایج صرفاً به دلیل تصادفی بودن خاص ناشی از این مقداردهی اولیه وزن‌ها، منفی‌تر از مثبت هستند.

پس از آموزش، عامل امتیاز ترجیح پایین‌تری برای زوج‌های با درجه بالا، امتیاز ترجیح بالاتری برای زوج‌هایی که منجر به s- چندجمله‌ای می‌شوند، و امتیاز ترجیح پایین‌تری برای زوج‌هایی که منجر به s- چندجمله‌ای با درجه حقیقی بالا می‌شوند، اختصاص می‌دهد. این روابط تنها به طور متوسط درست هستند، بنابراین نمی‌توانند نتایج را به طور کامل توضیح دهند، اما همچنین منطقی هستند، زیرا یافتن تولیدکننده‌ای با درجه پایین یا یک تک‌پارچه برای کاهش سریع‌تر در مراحل بعدی و خاتمه سریع الگوریتم بوخبرگر مفید است.

قسمت اول استراتژی یعنی انتخاب درجه، یک روش ساده است. بخشی از انگیزه ما برای ساختار شبکه این بود که می‌دانستیم می‌تواند انتخاب درجه را تقلید کند، بنابراین تعجب‌آور نیست که می‌تواند این ویژگی را یاد بگیرد و استفاده کند. برعکس، قسمت‌های دوم و سوم استراتژی‌های انتخاب استاندارد نیستند. وقتی استراتژی انتخاب یک جفت با کمترین درجه s- چندجمله‌ای را که به آن درجه حقیقی می‌گوییم، به صورت دستی اجرا کردیم، میانگین تعداد جمع‌ها در ۱۰۰۰۰ نمونه از توزیع ۳-۲۰-۱۰ وزنی ۱۳.۱۲۰ بود که ۱۱ درصد بهبود نسبت به استراتژی درجه است. از طرف دیگر، برای استراتژی‌ای که درجه را دنبال می‌کند اما ابتدا هر s- چندجمله‌ای که تک‌جمله‌ای است را انتخاب می‌کند، میانگین تعداد جمع‌ها ۱۳۴٫۲۰ بود که ۱ درصد بهبود نسبت به درجه است. در حالی که هیچ یک از استراتژی‌های دستی به بهبود ۳۷ درصدی عامل نسبت به درجه دست نمی‌یابد، قابل توجه است که این بینش‌ها از مدل منجر به استراتژی‌های قابل فهمی شده است که استراتژی‌های مرجع ما را در این حوزه شکست می‌دهند.



شکل ۱۳.۴: این نمودار امتیاز ترجیح یک عامل را برای ویژگی‌های خاص قبل و بعد از آموزش نشان می‌دهد. زوج‌های داده برای این نمودار طی ۱۰۰ دوره آموزشی در توزیع وزن‌دار ۳-۲۰-۱۰ تولید شده‌اند. توجه داشته باشید که مقیاس محور y (عمودی) تغییر کرده است.

پیوست ۱

در این بخش الگوریتم پایانی آموزش مورد نیاز در فصل ۴ پروژه که به زبان پایتون .نوشته شده ارائه شده است

Prog-I:

```
————————————————————————————————————————
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, Input

# Define the necessary polynomial functions as you have provided
def lex_order(x):
    if x[0] > x[2]:
        lm0, lm1, lm2, lm3 = x[0], x[1], x[2], x[3]
    elif x[2] > x[0]:
        lm0, lm1, lm2, lm3 = x[2], x[3], x[0], x[1]
    elif x[1] > x[3]:
        lm0, lm1, lm2, lm3 = x[0], x[1], x[2], x[3]
    elif x[3] > x[1]:
        lm0, lm1, lm2, lm3 = x[2], x[3], x[0], x[1]
    else:
```

```
        return x
    return [lm0, lm1, lm2, lm3]


def lcm_fg(x, y):
    x = lex_order(x)
    y = lex_order(y)
    lm1 = max(x[0], y[0])
    lm2 = max(x[1], y[1])
    return [lm1, lm2]


def spoly_fg(x, y):
    x = lex_order(x)
    y = lex_order(y)
    lcmfg = lcm_fg(x, y)
    s = [
        lcmfg[0] - x[0] + x[2],
        lcmfg[1] - x[1] + x[3],
        lcmfg[0] - y[0] + y[2],
        lcmfg[1] - y[1] + y[3],
    ]
    if s[0] == s[2] and s[1] == s[3]:
        return 0
    return lex_order(s)


def div_reduce(spolyfg, F):
    spolyfg = lex_order(spolyfg)
```

```
    red = spolyfg
    rewards = 0
    for f in F:
        if f[0] <= red[0] and f[1] <= red[1]:
            red = spoly_fg(red, f)
            rewards += 1
            if red == 0:
                return 0, rewards
            red = lex_order(red)
    return red, rewards

class BuchbergerEnv:
    def __init__(self, F):
        self.original_F = F
        self.current_F = F[:]

    def reset(self):
        self.current_F = self.original_F[:]
        return self.current_F[0]

    def step(self, action):
        i, j = action
        spoly = spoly_fg(self.current_F[i], self.current_F[j])

        if spoly == 0:
            return -1, True  # End episode if zero
```

```
        reducespoly, rewards = div_reduce(spoly, self.current_F)
        if reducespoly != 0:
            self.current_F.append(reducespoly)
            return -rewards, False
        return 0, False

class PPOAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.policy_model = self.build_model()
        self.value_model = self.build_model()

    def build_model(self):
        model = tf.keras.Sequential()
        model.add(Input(shape=(self.state_size,)))
        model.add(layers.Dense(128, activation='relu'))
        model.add(layers.Dense(self.action_size, activation='softmax'))
        return model

    def choose_action(self, state):
        state = np.array(state).reshape([1, self.state_size])
        probabilities = self.policy_model.predict(state).flatten()
        return np.random.choice(self.action_size, p=probabilities)
```

```
def create_random_dataset(num_samples):
    return
    [np.random.randint(1, 10, size=(4,)).tolist() for _ in range(num_samples)]


def train_agent():
    num_samples = 20
    F_samples = create_random_dataset(num_samples)
    env = BuchbergerEnv(F_samples)
    agent = PPOAgent(state_size=4, action_size=num_samples)

    for episode in range(1000):
        state = env.reset()
        done = False
        total_reward = 0

        while not done:

            i = np.random.randint(0, num_samples)
            j = np.random.randint(0, num_samples)
            if i == j:  # Make sure not to choose the same index
                j = (j + 1) % num_samples

            (for demonstration)
            action = (i, j)
            reward, done = env.step(action)
            total_reward += reward
```

```
        print(f'Episode: {episode}, Total Reward: {total_reward}')

if __name__ == "__main__":
    train_agent()
```

کتاب‌نامه

[1] Achiam, Joshua. Spinning up in deep reinforcement learning.

[2] Alvarez, Alejandro Marcos, Louveaux, Quentin, and Wehenkel, Louis. A machine learning-based approximation of strong branching. *INFORMS J. Comput.*, 29(1):185–195, 2017.

[3] Bengio, Yoshua, Lodi, Andrea, and Prouvost, Antoine. Machine learning for combinatorial optimization: a methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021.

[4] Dylan Peifer, Michael Stillman, Daniel Halpern-Leistner. Learning selection strategies in buchberger's algorithm.

[5] Huang, Zongyan, England, Matthew, Wilson, David J., Bridge, James, Davenport, James H., and Paulson, Lawrence C. Using machine learning to improve cylindrical algebraic decomposition. *Math. Comput. Sci.*, 13(4):461–488, 2019.

[6] Ian Goodfellow, Yoshua Bengio and Courville, Aaron. Deep learning.

[7] Truong, Tuyen Trung. Bounded birationality and isomorphism problems are computable. *Beitr. Algebra Geom.*, 65(1):129–155, 2024.