انتخاب خودکار اندازه سودآور بلاک‌ها در کرنل‌های تسریع داده شده

ترجمه: سید علی حسینی
شماره دانشجویی: ۹۸۷۳۷۲۲۳۶۳
۰۹۱۵۰۷۱۸۰۱۸

فهرست مطالب

چکیده

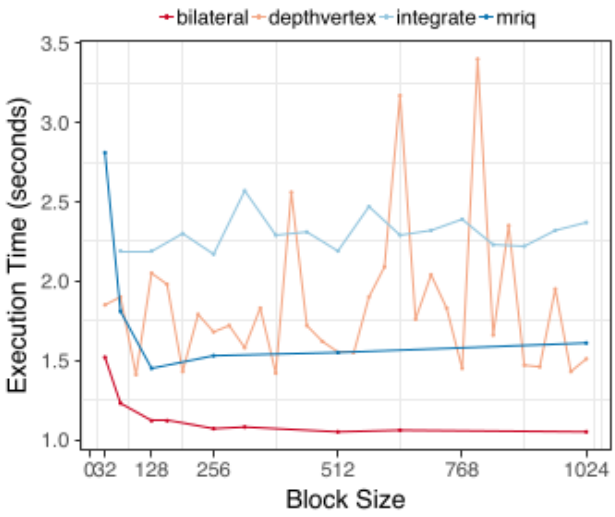
چکیده واحدهای پردازش گرافیکی (GPU) عملکرد بالایی را در مصرف توان پایین تا زمانی که منابع به خوبی مورد استفاده قرار می‌گیرند، فراهم می‌کنند. اندازه بلوک‌های یک نخ یک عامل در تعیین میزان اشغال هسته است و معیاری برای اندازه‌گیری بهره‌برداری از GPU می‌باشد. یک رویکرد کلی این است که اندازه بلوک را بیشترین میزان قابل اشغال در نظر بگیریم. با این حال، بسیاری از ترکیبات بلوک و اندازه شبکه می‌توانند بالاترین میزان اشغال را تامین کنند، اما عملکرد می‌تواند به طور قابل‌توجهی بین پیکربندی‌های مختلف متفاوت باشد. این به این دلیل است که تغییر در ساختار نخ، استفاده متفاوتی از منابع سخت‌افزاری را ایجاد می‌کند. بنابراین، بهینه‌سازی برای اشغال به تنهایی کافی نیست و ساختار نخ نیز باید در نظر گرفته شود. این مسئولیت برنامه‌نویس است که اندازه بلوک‌ها را تنظیم کند، اما انتخاب اندازه مناسب همیشه شهودی نیست. در این مقاله، ما از یادگیری ماشین برای انتخاب خودکار اندازه‌های بلوک سود آور استفاده می‌کنیم. علاوه بر این، ما نشان می‌دهیم که تکنیک‌های یادگیری ماشین به کارایی شمارنده‌ها وابسته است و کارایی می‌تواند به دلایلی متغیر باشد.

علاوه بر این، ما نشان می‌دهیم که تکنیک‌های یادگیری ماشین به کارایی شمارنده‌ها وابسته است و کارایی می‌تواند به دلایلی متغیر باشد.

۱.۰ مقدمه

واحدهای پردازش گرافیک (GPU) می‌توانند عملکرد خوبی را در مصرف توان پایین تا زمانی که استفاده خوبی از منابع وجود دارد، فراهم کنند. اندازه بلوک‌های یک نخ یک عامل در تعیین میزان اشغال هسته است. میزان اشغال یک نسبت بین تعداد ریسمان‌های فعال و حداکثر تعداد ریسمان‌های قابل برنامه‌ریزی است. میزان اشغال مشخص می‌کند که چگونه یک هسته موازی از GPU استفاده می‌کند و ارتباط نزدیکی با تخصیص منابع دارد. یک دستورالعمل کلی، پیدا کردن پیکربندی رشته‌ای است که به بالاترین میزان اشغال منجر می‌شود. با این حال، نشان‌داده شده‌است که برای برخی هسته‌ها بالاترین درصد اشغال همواره بهترین عملکرد را نمی‌دهد [۲۲]. اشغال بالا منجر به افزایش رقابت منابع می‌شود، چرا که رشته‌های بیشتری برای محدود کردن منابع سخت‌افزاری مانند ثبات‌ها و حافظه اشتراکی رقابت می‌کنند. ضریب اشغال پایین هر نخ را با منابع بیشتری فراهم می‌کند اما این می‌تواند اثر منفی ناشی از تاخیر پایین داشته باشد. ضریب اشغال پایین هر نخ را با منابع بیشتری فراهم می‌کند اما این می‌تواند اثر منفی ناشی از تاخیر پایین داشته باشد. علاوه بر این، اندازه‌های بلوک چند گانه می‌توانند بیش‌ترین اشغال را برای یک هسته مشخص داشته باشند، اما عملکرد آن‌ها می‌تواند در این تنظیمات متفاوت متفاوت باشد. بنابراین، بهینه‌سازی برای اشغال به تنهایی کافی نیست و هندسه نخ نیز باید در نظر گرفته شود. بنابراین، بهینه‌سازی برای اشغال به تنهایی کافی نیست و هندسه نخ نیز باید در نظر گرفته شود.

این کار توسط موسسه علوم ملی از طریق دریافت جایزه CNS ـ ۱۳۰۵۳۰۲ و CNS ـ ۱۲۵۳۲۹۲ مورد حمایت قرار گرفت.



شکل ۱: زمان اجرا چهار کاربرد با اندازه‌های بلوک مختلف

عملکرد فعلی نشان می‌دهد که برنامه نویسان شبکه و اندازه بلوک را برای بهینه‌سازی کاربردهای GPU انتخاب می‌کنند. انتخاب یک پیکربندی مسیر خوب همیشه شهودی نیست. تغییرات کوچک در اندازه بلوک رشته‌ای می‌تواند تاثیر قابل‌توجهی بر عملکرد داشته باشد.

تغییرات عملکرد چهار مغز نشان‌داده‌شده در شکل شکل را در نظر بگیرید. ۱. کارایی یک هسته می‌تواند نسبت به اندازه‌های مختلف بلوک‌ها تغییر کند. اگر چه اندازه بلوک بزرگ‌تر عملکرد بهتری دارد، اما بزرگ‌ترین اندازه بلوک لزوما بهترین نتایج را تولید نمی‌کند. به عنوان مثال، انتخاب یک اندازه بلوک کوچک‌تر از ۱۲۸.

انتخاب گزینه‌های مختلف برای پیکربندی بلوکی رشته می‌تواند زمان زیادی را از برنامه‌نویس طلب کند. شاید لازم باشد که برنامه‌نویس پیکره بندی رشته را به صورت دستی تغییر دهد، برنامه را دوباره اجرا کند، و نتایج عملکرد را برای هر تغییر تا زمان رسیدن سطح عملکرد مورد نظر جمع‌آوری

کند. علاوه بر این برای پیدا کردن یک اندازه بلوک بهینه چند بعدی در نظر گرفته شود. این جستجوی پیچیده فضای زیادی را طلب میکند.

ارزیابی دشوار است. همانطور که اندازه این فضای جستجو افزایش می‌یابد، انجام یک جستجوی کامل دشوار می‌شود و بسیاری از جستجوهای اکتشافی ممکن است به راحتی در فضای op-tima گیر کنند. ویژگی‌های این فضای جستجو شامل اندازه داده ورودی و تعداد ثبات‌ها تخصیص‌یافته است که هر دو با عملکرد هسته تحت یک اندازه بلوک مشخص همبستگی دارند.

عامل دیگری که می‌تواند باعث ایجاد تفاوت شود اندازه شبکه است
اندازه شبکه نشان‌دهنده مقدار کلی کاری است که باید از نظر تعداد موضوعات آغاز شود. هنگامی که یک اندازه شبکه بزرگ مورد استفاده قرار می‌گیرد، این منجر به کار کم‌تر برای هر رشته برای انجام و افزایش رقابت برای منابع سخت‌افزاری محدود می‌شود. بااستفاده از یادگیری ماشینی (ML)، عملکرد و مصرف توان هسته‌های GPU را می توان از طریق انتخاب خودکار پیکره بندی‌های thread سودآور بهبود بخشید. این کاهش تعداد هسته لازم را کاهش می‌دهد و امکان ارزیابی موثرتر فضای جستجوی پیچیده را فراهم می‌کند. علاوه بر این، تکنیک‌های ML همراه با شمارنده‌ها و عملکرد سخت‌افزاری، می‌تواند به ایجاد بینشی در مورد علل اصلی تفاو عملکرد بین پیکربندی‌های مختلف نخ کمک کند. در این مقاله ما یک استراتژی برای انتخاب اندازه بلوک‌های سودآور در هسته‌های GPU با استفاده از ML نظارت شده ارایه می‌کنیم. مدل ML از رویدادها عملکردی پویا به عنوان ویژگی‌های استفاده می‌کند. با توجه به هسته ،GPU چارچوب ما هسته را پروفایل کرده و ویژگی‌های دینامیکی مربوطه را استخراج می‌کند. چارچوب ما عملکرد کرنل را دریافت می‌کند و سپس ویژگی‌های دینامیکی مربوطه را استخراج می‌کند. سپس این مدل پیش‌بینی می‌کند که آیا تغییر در اندازه بلوک عملکرد هسته داده‌شده را بهبود خواهد بخشید.
چارچوب ما تمام مراحل اصلی یادگیری ماشین را بررسی می‌کند جریان کار، شامل استخراج ویژگی، انتخاب ویژگی، و برچسب گذاری داده آموزشی است. به منظور اطمینان از اندازه نمونه کافی برای داده‌های آموزشی، ما چندین متغیر کد را از یک برنامه ایجاد می‌کنیم. این متغیرها رفتار متمایزی را در پلتفرم هدف نشان می‌دهند، که اجازه می‌دهد طیف وسیعی از ویژگی‌های برنامه برای مدل یادگیری ماشین از آن وجود داشته باشد.
به طور خلاصه، سهم اصلی این ساخت یک روش ابتکاری مبتنی بر یادگیری ماشین جدید برای انتخاب اندازه‌های قطعه قطعات که در چند دهه اخیر مورد بررسی قرار گرفته‌اند. یک چارچوب کلی برای تهیه خودکار طبقه‌بندی نظارت شده به طور خودکار برای مدل‌سازی عملکرد ویژه سکو و خودکارسازی کار یادگیری ماشین
تحلیل علل زمینه‌ساز آنومالی های عملکردی ناشی از تغییرات قطعه نخ

۲.۰ پیش‌زمینه

GPU یک پردازنده موازی بسیار موازی است که به طور سنتی برای رندر کردن گرافیک کامپیوتری مورد استفاده قرار می‌گیرد. با این حال، GPU های مدرن معمولا برای انجام محاسبات در کاربردهای علمی و مهندسی مورد استفاده قرار می‌گیرند. یک GPU متشکل از مجموعه‌ای از Streaming multiprocessors (SMs) است و هر SM شامل تعدادی از واحدهای اجرایی به نام Processors

Stream (SPs) است. GPU های مدرن حاوی هزاران SP هستند. SM برای اجرای صدها رشته به صورت همزمان طراحی شده‌است و به دنبال دستورالعمل واحد، مدل چند داده (SIMD) اجرا می‌شود. قابلیت محاسبه GPU های NVIDIA ویژگی‌های مورد پشتیبانی سخت‌افزار GPU را شناسایی می‌کند [۲].

۱.۲.۰ cuda

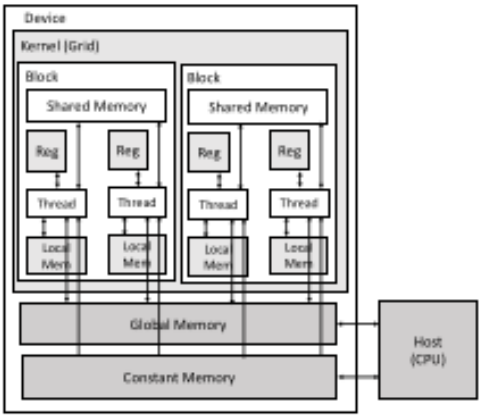
cuda یک واسط برنامه‌نویسی است که به ما امکان برنامه‌ریزی مستقیم GPU را می‌دهد. cuda C یک افزونه به زبان برنامه‌نویسی C است که به توسعه دهندگان اجازه می‌دهد تا توابع موازی را برای اجرا روی GPU بنویسند. در مدل cuda، GPU می‌تواند به وسیله اجرا روی نخ‌های موازی به کارایی بالایی دست پیدا کند.

۲.۲.۰ سلسله‌مراتب نخ‌ها

اساسی‌ترین واحد اجرای در cuda نخ است. ریسمان، که مجموعه‌ای از ۳۲ رشته هستند، به طور همزمان با هم اجرا می‌شوند به قطعات حلقوی تقسیم می‌شوند. نخه به طور مستقل از یکدیگر اجرا می‌شوند و به آن‌ها اجازه می‌دهد تا به هر ترتیبی در هر تعداد هسته قرار گیرند. ریسمان در همان بلوک یکسان بر روی یک multiprocessor یکسان اجرا می‌شود و به همان واحد حافظه اشتراکی دسترسی دارند. هر نخ از بلاک در طول اجرای یک هسته به یک SM محول می‌شود. یک شبکه مجموعه‌ای از بلوک‌های نخی است. تعداد بلوک‌های نخی در یک شبکه معمولا براساس اندازه داده پردازش می‌شود. بلوک‌های نخ‌ها در یک شبکه در عرض چند SMs نقشه‌برداری شده‌اند. حداکثر تعداد رشته‌هایی که می‌توانند برای هر بلوک تعیین شوند بسته به معماری GPU و توانایی محاسبه تغییر می‌کند. همچنین، حداکثر تعداد بلوک‌ها در هر SM و حداکثر threads در هر SM نیز بستگی قابلیت محاسبه بستگی دارد.
عوامل محدودکننده شامل تعداد ثبات‌ها و حافظه اشتراکی مورد نیاز هسته و تعداد و مقدار حافظه اشتراکی موجود بر روی میکروپروسسور می‌شود [۲].

۳.۲.۰ سلسله‌مراتب حافظه

یک GPU فعال شده دارای شش جز حافظه متفاوت است: رجیستر، حافظه اشتراکی، حافظه محلی، حافظه سراسری، حافظه بافت و حافظه ثابت. هر رشته حافظه خصوصی خودش را دارد. هر بلوک حافظه اشتراکی خود را دارد، که در میان همه نخ‌ها در آن بلوک سهیم است. حافظه سراسری، حافظه ثابت، و حافظه بافت را می توان توسط همه رشته‌ها به دست آورد.
حافظه بافت و حافظه فقط خوانده می‌شوند، در حالی که سایر انواع حافظه قابل خوانده / نوشتن هستند. یک نمودار تعمیم‌یافته از این سلسله‌مراتب حافظه در شکل نشان‌داده شده‌است.



شکل ۲: دیاگرام GPU مرکزی

۴.۲.۰ یادگیری ماشین

یادگیری ماشین (ML)روشی برای تجزیه و تحلیل داده‌ها است که از الگوریتم‌هایی استفاده می‌کند که به طور تکراری از داده استفاده می‌کنند تا بتواند از آن‌ها یادبگیرد و در آن‌ها الگوهای پنهانی را پیدا کند.
اگر یک برنامه کامپیوتری قادر به بهبود عملکرد خود در انجام یک کار با استفاده از تجربیات قبلی باشد، گفته می‌شود که آموخته‌است [۱۷].
یکی از بزرگ‌ترین نقاط ضعف ML این توانایی است که به طور خودکار محاسبات پیچیده ریاضی را به مجموعه بزرگی از داده‌ها با حداقل تلاش کاربر اعمال کند.
رایج‌ترین روش‌های ML تحت نظارت و یادگیری بدون نظارت هستند. در یادگیری بدون نظارت، داده ورودی با خروجی درست برچسب نمی‌شود. هدف یادگیری بدون نظارت، کشف تشابهات و یافتن ساختار در داده‌ها است.
الگوریتم‌های یادگیری ML با استفاده از نمونه‌های برچسب دار آموزش‌دیده اند. الگوریتم یادگیری با مجموعه‌ای از ورودی‌ها و خروجی‌های صحیح متناظر، به طور معمول به عنوان مجموعه آموزشی، ارایه می‌شود. هدف از الگوریتم یادگیری استنباط یک تابع است که خطا را با توجه به این ورودی‌ها به حداقل می‌رساند. به طور خلاصه، هدف الگوریتم‌های یادگیری با نظارت، یادگیری نگاشت اگر X آن‌گاه Y است، که در آن X یک نمونه است و Y یک برچسب کلاس است. یک مرز تصمیم‌گیری، سطحی بیش از حد است که فضای یادگیری را به مجموعه‌ها، یکی برای هر طبقه تقسیم‌بندی می‌کند. محدوده تصمیم‌گیری منطقه‌ای از فضای یادگیری است که در آن برچسب خروجی مبهم است. فضای یادگیری به صورت خطی قابل جداسازی است در صورتی که کلاس‌های فضایی را می توان با یک سطح خطی منفرد از هم جدا کرد.

۳.۰ کارهای مرتبط

۴.۰ بهینه‌سازی برای پیکربندی نخ‌ها

al et Seo و همکارانش یک روش ابتکاری برای انتخاب اندازه گروه کاری برای هسته‌های OpenCL که در پردازنده‌های چندهسته‌ای کار می‌کنند را توسعه دادند [۲۰]. آن‌ها از تخمین ایستا و زمان اجرا برای تنظیم اندازه گروه کاری برای بهبود موقعیت و بالانس بار استفاده می‌کنند. آن‌ها تعداد آن‌ها را به یک جستجوی کامل از تمام پیکره بندی‌های کاری ممکن مقایسه می‌کنند. این نتایج نشان می‌دهند که استراتژی آن‌ها می‌تواند یک عملکرد مشابه داشته باشد
مثلا هزینه پایین‌تر. آن‌ها این تکنیک را به GPU گسترش نمی‌دهند که در آن مسائل عملکرد بسیار متفاوت هستند.

Tran و همکارانش یک مدل تنظیم برای محاسبه گرید و اندازه بلوک برای دستیابی به عملکرد بهینه براساس بالاترین اشغال پیشنهاد دادند [۲۱]. رویکرد آن‌ها قادر است مجموعه‌ای از شبکه انتخابی و اندازه بلوک‌ها را سریع‌تر از استفاده از جستجوی کامل محاسبه کند. با این حال، مدل آن‌ها تنها متکی بر آستانه بلوک و اندازه‌های شبکه‌ای است که توسط معماری GPU اجرا می‌شوند.
آن‌ها ویژگی‌های هسته را در نظر نمی‌گیرند که در تعیین پیکربندی بهینه نخ ضروری است. مدل آن‌ها عمدتا برای کاهش فضای جستجو به جای استفاده از یک پیشگوی ML مورد استفاده قرار

می‌گیرد و ممکن است فهرستی از پیکربندی‌های چند کاندید را تولید کند.
al. Magniet در این مقاله با استفاده از یک کامپایلر OpenCL مبتنی بر ،llvm تبدیل نخ- درشت به کار گرفته شده‌است. علاوه بر این، آن‌ها از درخت‌های رگرسیون و شمارنده کارایی سخت‌افزاری برای شناسایی ویژگی‌ها استفاده کردند که تحت‌تاثیر درشت سازی نخ قرار می‌گیرند. آن‌ها تاثیر عامل بزرگ شدن بر عملکرد را ارزیابی کردند و دریافتند که درختان رگرسیون می‌توانند ویژگی‌های سخت‌افزاری مرتبط با عملکرد را شناسایی کنند. al. Magniet مطالعه دیگری که در آن از دستورها سطح منبع برای تنظیم پارامترهای پیکربندی نخ استفاده می‌کنند، انجام شد. با این حال، روش تنظیم آن‌ها به طور آشکار استفاده از منابع هسته را مدل نمی‌کند [۱۵]. Gupta و همکاران STATuner را طراحی کردند که مجموعه‌ای از ویژگی‌های استاتیک را شناسایی می‌کند که هسته cuda را مشخص می‌کند و یک طبقه‌بندی کننده ماشین بردار پشتیبان را می‌سازد تا پیش‌بینی کنند که کدام اندازه بلوک بهترین عملکرد را فراهم می‌کند [۹]. معیارهای ایستا با استفاده از هسته‌های cuda در llvm بدست می‌آیند. تحلیل استاتیک کد دودویی تولید شده و IR برای گرفتن معیارهایی برای ترکیب دستورالعمل، حلقه‌ها، کاربرد ثبت، حافظه اشتراکی در هر بلوک و همزمانی thread انجام می‌شود. رویکرد ما در این است که چارچوب ما از ویژگی‌های هسته پویا به عنوان ورودی مدل ML استفاده می‌کند.

۵.۰ دستگاه یادگیری مدل‌سازی غیرکارا

مطالعه کاربردهای اخیر تکنیک‌های ML در مدل‌سازی عملکرد و تنظیم در HPC الگویی از چالش‌های ورودی و اینکه متخصصان آل ام به آن‌ها رسیدگی کرده‌اند را نشان می‌دهد. کاربرد اولیه مدل یادگیری ماشین و تنظیم (mlmt)به عنوان پاسخی به تنظیم زمان طولانی تنظیم برای autotuning مبتنی بر جستجو ظاهر شد. به این ترتیب، برخی از کارهای اولیه در این حوزه مبتنی بر استفاده از مدل‌سازی اکتشافی، هرس و جستجوی تجربی به منظور کاهش فضای پارامتر و یافتن معیارهای توقف اولیه هستند چون شبکه‌های عصبی و مدل‌های رگرسیون لجستیک محبوبیت پیدا کردند، برای حل مشکلات در HPC به کار گرفته شدند. Cava - zos و همکارانش کار خود را در شناسایی توالی بهینه کامپایلر با استفاده از مدل‌های رگرسیون لجستیک چندگانه پیش بردند. [۳]. برآورد افزایش یا کاهش عملکرد با استفاده از یک بهینه‌سازی خاص به عنوان کاهش مشکل بزرگ‌تر یافتن مجموعه بهینه‌ای از بهینه‌سازی برای بسیاری از سناریوها به خوبی کار می‌کند. با این حال در، این تکنیک، امکان هم‌افزایی بیشتری نسبت به رفتارهای مخالف بهینه‌سازی‌های چندگانه وجود دارد.
علاوه بر این، همانطور که تعداد بهینه‌سازی‌های موجود در دسترس هنوز زیا است، زمان ایجاد داده‌های آموزشی و تعداد طبقه بندها مورد نیاز نیز زیاد باقی می‌ماند. به عنوان مثال، GCC ۲.۸.۴ ۱۹۳ بهینه‌سازی و انتخاب توالی بهینه اساسا به معنای ایجاد یک آرایه از ۱۹۳ طبقه‌بندی کننده و مجموعه داده آموزشی برای هر طبقه‌بندی کننده است. علاوه بر این، معماری‌های مختلف در حال تغییر در HPC چالش سازگاری را مطرح کرده‌اند. Fursin و همکارانش برای پرداختن به

این چالش با جمع‌آوری دانش بهینه‌سازی جمعی در میان معماری، به منبع یابی انبوه روی‌آورده اند [۸].

مشابه بسیاری از مشکلات ،ML موفقیت تکنیک‌های ML به مشخصه سازی دقیق ورودی بستگی دارد. محققان تلاش کرده‌اند تا برنامه‌ها را با استفاده از نمودار جریان کنترل برنامه [۶] مشخصه‌های برنامه ایستا [۸] و شمارنده‌های عملکرد سخت‌افزاری ارزیابی کنند [۱۹]. شمارنده‌های عملکرد سخت‌افزار مزایای افزوده پویا بودن را دارند و می‌توانند پاسخ سیستم خاص معماری را ثبت کنند. با این حال، تعداد زیادی از اتفاقات عملکرد وجود دارد و انتخاب موثرترین دشوار است.

بسیاری از آن‌ها به دست‌چین کردن روی آوردن شده‌اند [۱۶] در حالی که برخی روش‌های آماری برای انتخاب رویدادهایی بکار گرفته‌اند که در میان اجرای برنامه‌های مختلف متفاوت است [۱۱]، [۱۹]. وو و سایرین مدلی را طراحی کردند که از شبکه‌های عصبی و خوشه‌بندی k ـ means برای تخمین عملکرد و قدرت هسته بر روی سایر تنظیمات سخت‌افزاری استفاده می‌کند [۲۴]. مقادیر شمارنده عملکرد سخت‌افزار جمع‌آوری‌شده از یک پیکره بندی سخت‌افزار به عنوان ورودی مدل برای پیش‌بینی عملکرد هسته در تنظیمات دیگر مورد استفاده قرار می‌گیرند. تمرکز بر روی بهینه کردن یک هسته برای یک سیستم مشخص نیست، بلکه مشخص کردن چگونگی عملکرد دانه در سیستم‌های دیگر است.

به رغم چالش‌های پژوهشگران HPC در کاربرد ،ML تکامل ML در HPC چشمگیر بوده‌است. بسیاری از انواع روش‌های ML به طور موفقیت آمیزی به شاخه‌های مختلفHPC در بهینه‌سازی عملکرد از طریق تغییرات کد [۵]، تنظیمات زمان اجرا [۷]، [۱۳]، شناسایی تنگناهای عملکرد [۱۰]، [۱۱] و اخیرا، در مدیریت کارآمد انرژی [۴] اعمال شده‌اند.

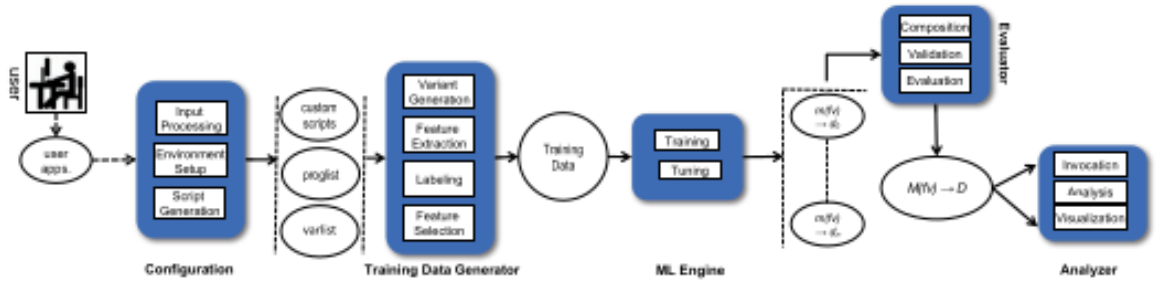
برای شروع، چارچوب ما دستورالعمل‌های سفارشی ایجاد می‌کند که وظایف استخراج ویژگی، انتخاب ویژگی، تولید داده آموزشی، آموزش مدل، ارزیابی و انتخاب را هدایت می‌کند. مدل ایجادشده به عنوان R ذخیره می‌شود و واسط برای کاربر فراهم می‌کند تا آن را در برنامه‌های نامریی اجرا کند. یک حالت تعاملی نیز برای انجام وظایف فرعی به طور انتخابی پشتیبانی می‌شود.

۱.۵.۰ پیکربندی

ما یک رابط ساده فراهم می‌کنیم که به کاربران اجازه می‌دهد تا دایرکتوری برنامه‌ها را مشخص کنند تا به عنوان ورودی برای نسل داده‌های آموزشی مورد استفاده قرار گیرند. این رابط پیکربندی مسیر محیطی را تعیین می‌کند، دستگاه‌های فعال cuda را شناسایی می‌کند، و ساخت و اجرای سفارشی می‌سازد که متناسب با محیط کاربر هستند. در این مرحله دستورالعمل‌ها برای تولید داده‌های آموزشی در فایلی به نام proglist مشخص می‌شوند.

۲.۵.۰ آموزش داده‌های آموزشی

پس از ایجاد سفارشی و اجرای دستورالعمل‌ها، یک فایل Proglist و Configurer ایجاد می‌کند که اطلاعات لازم برای ایجاد داده‌های آموزشی در پلتفرم هدف را فراهم می‌کند. هر سطر از فایل حاوی اطلاعاتی برای اجرای هر برنامه است که قرار است در مجموعه آموزشی مورد استفاده قرار گیرد. این فایل به عنوان ورودی به یک اسکریپت به نام varlist.gen.sh عمل می‌کند.



شکل ۳: دیاگرام GPU مرکزی

varlist.gen.sh هر خط از proglist را می‌خواند و یک فایل با نام varlist را می‌خواند که شامل دستورالعمل‌ها برای ایجاد انواع برنامه برای هر برنامه پایه است که در proglist فهرست شده‌است. این متغیرها شامل تغییر پرچم ـ ، پیکربندی بلوکی رشته در راه‌اندازی هسته، و داده‌های ورودی برنامه متفاوت (در زمان دردسترس)می‌شوند. لیست متغیرها به یک اسکریپت فرستاده می‌شود که هر نوع برنامه را تولید می‌کند، می‌سازد و اجرا می‌کند. در این مرحله، ویژگی‌های زمان اجرای هر برنامه حاوی لیست متغیرها با استفاده از nvprof جمع‌آوری شده‌است. جمع‌آوری و پردازش داده‌ها در این مرحله با جزییات بیشتر در بخش X توضیح داده می‌شود.

۳.۵.۰ موتور ML

در مرحله موتور یادگیری ماشین (ML)داده‌های آموزشی به یک اسکریپت R ارسال می‌شود. در این سناریو، مجموعه داده آموزشی به طور تصادفی به مجموعه‌های آموزشی و آزمایش تقسیم می‌شود. این فرآیند ۱۰ بار تکرار می‌شود، تنظیم پارامترهای تنظیم‌کننده در هر زمان، و مدل که بالاترین صحت را در طول اعتبار دارد انتخاب می‌شود. عملکرد مدل نهایی با استفاده از مجموعه تست ارزیابی می‌شود تا اطمینان حاصل شود که overfitting رخ نداده است.

۴.۵.۰ تحلیل گر

چارچوب حاضر در حال حاضر از سه نوع تصویرسازی تحلیل حمایت می‌کند تا اطلاعاتی را در مورد داده‌های آموزشی و مدل ایجاد شده ارایه دهد.

قطعات PCA

قطعات PCA برای بررسی ویژگی‌های داده‌های آموزشی استفاده می‌شوند. خوشه‌بندی به روش k-means بر روی فضای ویژگی اعمال می‌شود، جایی که مقدار k از طریق روش سیلوئت تعیین می‌شود. ما مولفه اصلی را اجرا می‌کنیم.
تجزیه و تحلیل (PCA)در فضای ویژگی و نتایج خوشه‌بندی بر روی نمودارهای پراکندگی با استفاده از دو مولفه اصلی (PC ها)مشاهده می‌شوند که بیش‌ترین تغییرات را در داده‌ها توضیح می‌دهند. یک نقطه در نمودار نشان‌دهنده یک متغیر کد است. نکاتی که می‌توانند برای نشان دادن برنامه پایه، برچسب گذاری کلاس، و ارزش‌های مبتنی بر آستانه بکار گرفته شوند. Ellipses نشان‌دهنده خوشه و دو نقطه سقوط در همان خوشه است که نشان می‌دهد که آن‌ها رفتار مشابهی را نشان می‌دهند. نقشه‌های خوشه‌ای ـ PCA می‌تواند

اطلاعاتی را در مورد داده‌های آموزشی به چندین روش ارایه کند. تعداد خوشه‌ها انعکاسی از تعداد انواع مختلف کدها هستند.

قطعات قطعات VR ـ PCA

اگر چه pcas در درجه اول برای کاهش ابعاد استفاده می‌شوند، در mlmt می‌توانند به روش‌های دیگری مفید باشند. ما می‌توانیم به یک کامپیوتر به عنوان یک ویژگی ترکیبی که یک الگوی عملکرد گسترده را توصیف می‌کند فکر کنیم. به عنوان مثال، یک PC ممکن است دارای رفتار مقید به حافظه باشد و دارای ویژگی‌های مرتبط مانند نرخ miss، DRAM LLC و چرخه‌های توقف باشد. با این حال، به طور کلی، رابطه بین بسیاری از رویداده‌ای عملکرد یا نامعلوم است و یا مشخص نیست.

شناسایی رویداده‌ای مهم عملکرد که یک کامپیوتر PC را تشکیل می‌دهند می‌تواند بینش ارزشمندی درباره عملکرد برنامه و ویژگی‌های معماری ارایه دهد. بااینحال، چالش این است که کامپیوترها قادر به تصویرسازی مستقیم نیستند. برای پرداختن به این موضوع، ما چرخش Varimax را بر روی خرده ـ که از طریق PCA کشف شد، اعمال کردیم و سپس از یک نمودار قطعه‌ای برای تجسم سهم هر ویژگی برای رایانه‌های شخصی k ام استفاده کردیم. این قطعه قطعات یک راه سریع برای شناسایی ویژگی‌های مرتبط فراهم می‌کند (اگرچه ماهیت رابطه در فضای ویژگی آشکار نشد). این دانش می‌تواند برای بهینه‌سازی کد مستقل مدل تولید شده مورد استفاده قرار گیرد.

)

تجزیه درخت تصمیم‌گیری) درخت‌های تصمیم‌گیری به overfitting متمایل هستند و توانایی آن‌ها برای یادگیری فضاهای مختلط محدود است. با وجود این نواقص، درخت‌های تصمیم‌گیری آسان هستند و می‌توانند راهی شهودی برای درک یادگیری پشت پیش‌بینی‌ها فراهم کنند.

۶.۰ فرموله کردن مدل‌سازی

مدل ما از یادگیری ماشین برای فراهم کردن کاربر استفاده می‌کند پیشنهاداتی در زمینه نحوه اصلاح اندازه بلوک رشته‌ای کد آن‌ها ارایه شده‌است. با توجه به یک هسته، مدل ما مشخص می‌کند که آیا اندازه بلوک رشته‌ای باید افزایش یابد یا کاهش یابد تا به عملکرد بهتر دست یابد

۷.۰ ابعاد بلوک رشته‌ای قانونی

عوامل متعددی باید مورد توجه قرار گیرند:
محدودیت‌های سخت‌افزاری :GPU
ابعاد بلوک رشته‌ای اصلی
درستی نتایج هسته
اغلب هسته به گونه‌ای کدگذاری شده‌است که صحت نتایج به اندازه بلوک بستگی دارد. این بدان معناست که در حالی که یک اندازه بلوک مشخص در cuda قانونی است، ممکن است در زمینه برنامه مورد نظر معتبر نباشد. می‌توان مشخص کرد که آیا اندازه بلوک با بررسی نتایج یک برنامه با استفاده از اندازه بلوک جدید با نتایج اصلی معتبر است یا نه. توجه داشته باشید که این رویکرد تنها برای برنامه‌هایی که خروجی آن‌ها قطعی است کار می‌کند.

به این دلایل، ما انتخاب کردیم که مدلی که تغییرات نسبی در اندازه بلوک را به جای دادن اعداد مطلق نشان می‌دهد. سپس برنامه‌نویس می‌تواند اندازه بلوک معتبر بعدی را در جهت تغییر انتخاب کند.

۸.۰ الگوریتم انتخاب یادگیری ماشین

برای پیش‌بینی جهت تغییر در اندازه بلوک برای یک هسته مشخص، ما از ماشین‌های بردار پشتیبان (SVM) استفاده می‌کنیم. در انتخاب کدام الگوریتم یادگیری ماشین برای استفاده، ما در نظر می‌گیریم که چه نوع مرزه‌ای تصمیم‌گیری در فضای ویژگی ما انتظار داریم. به طور خاص، این که آیا فضای ویژگی به صورت خطی قابل‌تفکیک است، در انتخاب کدام مدل یادگیری ماشین برای استفاده مهم است.

اگر چه داده‌های ما فقط سه طبقه‌بندی را شامل می‌شود، فضای ویژگی بسیار پیچیده‌تر است. به همین دلیل، داده‌های ما یک مرز تصمیم‌گیری مجزای خطی را نشان نمی‌دهد. بنابراین، ما تصمیم گرفتیم از الگوریتمی توانمند برای یادگیری فضاهای مختلط استفاده کنیم که بطور خطی قابل‌تفکیک نیستند. ما SVM را به خاطر دقت و توانایی بالا برای یادگیری جستجوی پیچیده انتخاب کردیم فاصله دارد.

دیگر نقاط قوت SVM این است که آن‌ها بیش از حد تحت‌تاثیر داده‌های نویزی قرار نمی‌گیرند و مستعد overfitting نیستند. ما از ترفند کرنل استفاده می‌کنیم، که فضای ویژگی را به فضای بیشتر بعدی نگاشت تا یادگیری مرزه‌ای تصمیم‌گیری غیر خطی غیر خطی را فعال کند.

ما همچنین استراتژی تمام ـ را بکار می‌بریم که چندین SVM دوتایی را ترکیب می‌کند تا امکان پیش‌بینی چند طبقه را فراهم کند.

علاوه بر این، ما بر خوشه‌بندی برای ارزیابی فضای ویژگی و درخت‌های تصمیم برای ارایه بینش معنادار به دلایلی تکیه می‌کنیم که چرا برخی هسته‌ها با اندازه‌های بلوک کوچک‌تر در اندازه‌های بلوک بزرگ‌تر عمل می‌کنند.

۹.۰ نسل یادگیری داده

چارچوب ما قادر است تا حداکثر تخصیص ثبت و اندازه بلوک مغز cuda را به منظور تولید انواع کده‌ای چندگانه از همان برنامه پایه، دستکاری کند. سپس معیارهای دینامیکی هر کدام از تغییرات هسته با استفاده از counters عملکرد جمع‌آوری می‌شوند. این مجموعه از معیارها، بردار ویژگی ورودی برای مدل ML است.

۱۰.۹.۰ کلرینه شدن مشخصه

چارچوب ما از رویداده‌ای زمان اجرا به عنوان ویژگی‌های استفاده می‌کند. برای جمع‌آوری رویداده‌ای زمان اجرا، مقادیر از counters عملکرد سخت‌افزاری با استفاده از nvprof را مطالعه می‌کنیم. ما یک اسکریپت shell ایجاد کردیم که لیستی از وقایع انتخاب‌شده از پرونده متنی را می‌خواند و آن‌ها را به nvprof منتقل می‌کند. برای کاهش زمان مورد نیاز برای جمع‌آوری این رویدادها، ما از تسهیم و تقسیم رویدادها به گروه‌هایی استفاده می‌کنیم که می‌توانند در طول یک برنامه منفرد بدون ایجاد درگیری در counters سخت‌افزاری اندازه‌گیری شوند. براساس دانش و تحلیل کارشناسی،

رویدادهایی را انتخاب می‌کنیم که ارتباط نزدیکی با اندازه بلوک رشته‌ای دارند. رویداده‌ای انتخابی در جدول I نشان‌داده شده‌اند. علاوه بر در نظر گرفتن کاربرد منابع، ما ویژگی‌های برنامه زیر را مدلسازی می‌کنیم.

حافظه حافظه

Events Collected	
gld_request gst_request	fb_read_sectors fb_write_sectors
l1_local_load_hit l1_local_store_hit l1_global_load_hit	l1_local_load_miss l1_local_store_miss l1_global_load_miss
uncached_gld_transaction	global_store_transaction
gld_inst_32bit gst_inst_32bit	inst_issued inst_executed
not_predicated _off_thread_inst_executed	thread_inst_executed
l2_write_sector_misses l2_read_l1_hit_sectors l2_total_write_sector_queries	l2_read_sector_misses l2_total_read_sector_queries shared_load_replay
global_ld_mem _divergence_replays	global_st _mem_divergence_replays

شکل ۴: مجموعه رویدادهای NVPROF

دسترسی به حافظه می‌تواند تاثیر زیادی بر عملکرد مغز داشته باشد. coalescing یک تکنیک دسترسی به حافظه است که در آن درخواست‌های حافظه در یک خط حافظه نهان با هم گروه‌بندی می‌شوند تا یک تراکنش واحد ایجاد کنند. coalescing معمولا در سطح پیچ انجام می‌شود. درخواست‌های مرتب شده برای مکان‌های مجاور حافظه از نخ‌ها در همان حالت پیچ در یک تراکنش مرکب، به میزان زیادی ترافیک حافظه را کاهش می‌دهد. با تعداد بیشتری از warps، دسترسی به حافظه بیشتر می‌تواند در یک زمان رخ دهد. با این حال، تعداد زیادی از warps در هر SM می‌تواند عملکرد را به دلیل افزایش رقابت منابع تنزل دهد. به این دلایل، هسته‌ها که وابسته به حافظه هستند تمایل بیشتری به تغییرات در اندازه بلوک دارند.

انتقال کنترل

واگرایی کنترل عامل دیگری است که می‌تواند عملکرد مغز را تحت‌تاثیر قرار دهد. دستورها شاخه مکرر و واگرایی شاخه می‌تواند عملکرد را کاهش دهد. یک دستورالعمل کنترل واگرا است اگر آن را در یک پیچ درگیر کند تا مسیرهای اجرایی مختلف را به دست بگیرد. در cuda، واگرایی منجر به serialization مسیرهای اجرایی می‌شود در نتیجه تعداد کل دستورالعمل‌ها را افزایش می‌دهد. علاوه بر این، نخ‌ها در یک پیچ نمی‌توانند تا زمانی که همه نخ‌ها از مسیر شرطی خارج شده‌اند، ادامه پیدا کند. اندازه بلوک کوچک‌تر می‌تواند سربار واگرایی کنترل را با کاهش تعداد دستورها اجرا شده در هر پیچ و محدود کردن تعداد رشته‌هایی که باید به خاطر واگرایی صبر کنند، کاهش دهد.

با این‌حال اگر تعداد کمی از رشته‌ها راه‌اندازی شوند، ممکن است برای پنهان کردن تاخیر در آموزش کافی نباشد.

۲.۹.۰ مجموعه رویداد

نسخه پایه هر هسته با استفاده از اندازه بلوک پیش‌فرض نخ و فشار ثبت در نظر گرفته می‌شود. برای هر نسخه خط پایه، ما اندازه بلوک را در پیکربندی پرتاب هسته کد تغییر دادیم. ما اساس و همه متغیرها را اجرا کردیم و زمان اجرا و زمان اجرای هسته را با استفاده از nvprof جمع‌آوری کردیم. سپس سرعت هر نمونه را بر روی نسخه خط پایه محاسبه کردیم. برچسب‌ها به هر نمونه در مجموعه داده براساس سرعت و اندازه بلوک اضافه شدند.

۳.۹.۰ کوواریانس داده‌ها

برای آموزش مدل ML، ما باید آن را با نمونه‌های برچسب دار فراهم کنیم که از آن‌ها یاد گرفته خواهد شد. برچسب دستی هر نمونه در مجموعه داده آموزشی وقت گیر است. برای کم کردن کاربر این وظیفه، چارچوب ما فرآیند را با استفاده از متن‌ها و یک الگوریتم ساده برای مشخص کردن برچسب‌ها تغییر می‌دهد.
برای هر نمونه در مجموعه داده آموزشی، ابتدا خط پایه محاسبه می‌شود. یک سرعت را در نظر می‌گیریم که بد است، یک سرعت = ۱ تا خنثی و یک سرعت ۱ تا خوب باشد. سپس اندازه بلوک متغیر با اندازه بلوک پایه مقایسه می‌شود و برچسب با استفاده از الگوریتم ۱ تعیین می‌شود.

Algorithm 1 Labeling algorithm

```
for all d ∈D do
  if speedup < 1 and newBlock < origBlock then
    new.Label ← increase.
    orig.Label ← noChange.
  else if speedup < 1 and newBlock > origBlock then
    new.Label ← decrease.
    orig.Label ← noChange.
  else if speedup > 1 and newBlock < origBlock then
    new.Label ← noChange.
    orig.Label ← decrease.
  else if speedup > 1 and newBlock > origBlock then
    new.Label ← no Change.
    orig.Label ← increase.
  end if
end for
```

۴.۹.۰ انتخاب ویژگی

انتخاب ویژگی برای بهبود دقت مدل ML اهمیت دارد. ویژگی‌های اضافی که اضافی هستند یا هیچ اطلاعات اضافی برای مدل ارایه نمی‌کنند باید حذف شوند. اول، هر حادثه‌ای که مقدار صفر برای تمامی برنامه‌ها داشت را حذف کردیم. سپس، رابطه بین ویژگی‌های باقیمانده را با محاسبه ضرایب همبستگی با استفاده از فرمول همبستگی پیرسون بررسی کردیم، که یک وابستگی خطی بین دو متغیر، X و Y را اندازه‌گیری می‌کند:
ویژگی‌های با ضریب همبستگی بیشتر از ۰.۹ از

$$r = \frac{\sum(X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum(X - \overline{X})^2 \sum(Y - \overline{Y})^2}}$$

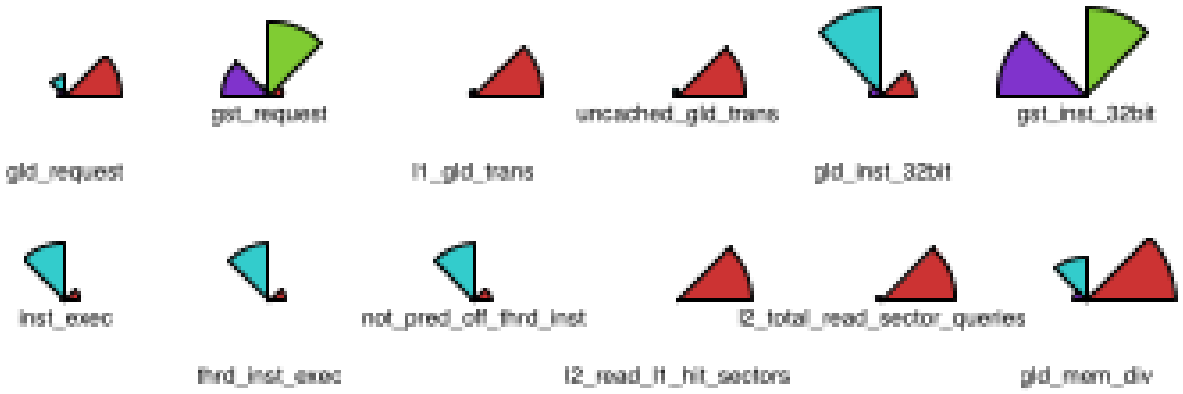
مجموعه حذف شدند. خصوصیاتی که به شدت با همه ویژگی‌های دیگر هم‌بسته هستند هیچ اطلاعات اضافی را به داده‌ها اضافه نمی‌کنند در نتیجه اضافی هستند و می‌توانند دقت پیش‌بینی مدل را کاهش دهند. ویژگی‌های باقی مانده و همبستگی آن‌ها در شکل ۴ نشان‌داده شده‌است. تخمین زده می‌شود که این ویژگی‌ها بالاترین قدرت پیش‌بینی را دارند.

۱۰.۰ امتحان تجربی

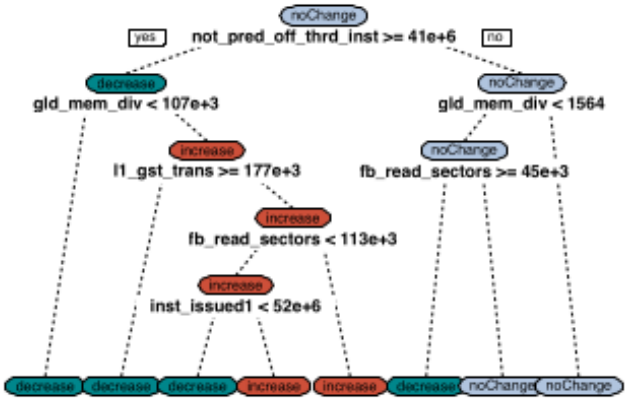
ما مدل خود را با استفاده از a تسلا K۴۰c بر روی سیستم linux که ۵.۷ installed نصب کرده بود، ارزیابی کردیم. The دارای قابلیت محاسباتی ۵.۳ است، حداکثر ۱۰۲۴ رشته در هر بلوک و حداکثر ۲۰۴۸ رشته در هر SM را پشتیبانی می‌کند.
.A کلرینه شدن ما از هسته‌ها از suites [۱] و SLAMBench [۱۸] برای تولید داده‌های آموزشی استفاده کردیم.

۱۱.۰ نتایج

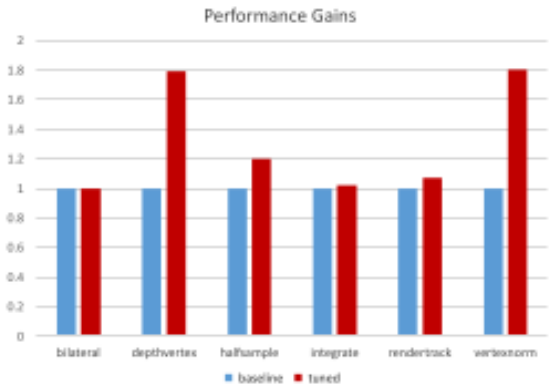
.A آموزش ویژگی فضایی پیچیدگی فضای ویژگی را می توان با اجرای تحلیل مولفه اصلی (PCA) و خوشه‌بندی K ـ means در مجموعه داده آموزشی ما ارزیابی کرد. همانطور که در شکل می‌بینید ۵ که در آن بسیاری از اندازه‌های بلوک در داخل یک خوشه قرار می‌گیرند، بهترین اندازه بلوک همواره آسان نیست.



شکل ۵: ماتریس همبستگی از ویژگی های باقی مانده پس از انتخاب ویژگی ها



شکل ۶: تجزیه و تحلیل نقشه Cluster-PCA در مجموعه داده های آموزش ما



شکل ۷: معیارهای تقسیم درخت تصمیم.