"هو"

موضوع پژوهش:

مقدمات کاربرد جبرخطی در یادگیری ماشین

پژوهنده: کیمیا اسماعیلی

استاد: دکتر سیامک یاسمی

استاد حل تمرین: فرزانه ادیبی

فهرست مطالب:

چکیده:

در این پژوهش سعی بر آن است تا کاربرد تئوری جبرخطی در یادگیری ماشین را به طور مقدماتی بررسی کرده و مثالی از آن را در بخش عملی و برنامه نویسی ببینیم.

مقدمه:

همان طور که میدانیم جبرخطی، شاخه مهمی از ریاضیات است که با نام "ریاضیات داده ها" نیز شناخته میشود. (ریاضیات را میتوان به عنوان یک زبان، فلسفه و علم از جنبه های مختلف برررسی کرد) در کل ریاضیات به عنوان زبانی برای بیان مسائل علمی ، کارکرد های پیشرفته و استدلال، در همه جا دیده می شود پس کاملا واضح است که استفاده از شاخه ای از ریاضیات به اهمیت و گستردگی جبرخطی در برنامه نویسی، اجتناب ناپذیر و در واقع بسیار کاربردی است.

یادگیری ماشین، یکی از زمینه های مرتبط با هوش مصنوعی است که عملکرد ادراک یک موجود هوشمند را شبیه سازی میکند (مثال ساده آن الگوریتم هایی است که اسپم یا هرزنامه را در نامه های الکترونیکی تشخیص داده و جدا مینمایند.) و پایه‌ی عملیاتِ یادگیری‌ماشین و یادگیری‌عمیق، اعداد هستند.

برای اینکه با مفاهیم پایه‌ای یادگیری ماشین آشنا شوید، نیاز به یادگیری بیشتر مفاهیم ریاضی خواهید داشت. شما باید نحوه کارکرد الگوریتم‌های مختلف، محدودیت‌های آن‌ها و فرضیات اساسی اتخاذشده برای آن‌ها را درک کنید. در ابتدای کار، حوزه‌های بسیار زیادی از قبیل جبر، حسابان، آمار، هندسه سه‌بعدی و ... برای مطالعه وجود دارند.

با توجه به اینکه کامپیوتر های امروزی برای پردازش دو عدد 0 و1 طراحی شده اند، برای انجام عملیات مختلف، به ابزاری مانند ماتریس ها و بردار ها نیاز داریم تا بتوان پردازش های پیچیده تر و هوشمندانه تر داشته باشیم. در ادامه به نمونه های نیازمندی به جبرخطی بیشتر میپردازیم.

❖ چرا جبرخطی در یادگیری ماشین؟

مثال هایی از کاربرد آن:

: (I)

به تصویر بالا نگاه کنید. انسان هایی با بینایی سالم، اثر "شب پرستاره" از ونگوگ را میبینند. دیدن یک تصویر و پردازش و تشخیص آن برای چشم انسان سالم، دشوار نیست اما اگر بخواهیم کدی بنویسیم تا کامپیوتر، تصویر مشابهی را نمایش دهد، کار بسیار سختی پیش روی ما خواهد بود (حتی بیان آن نیز کار .سختی است)

ما می‌توانیم تصویر و جزئیات آن را تشخیص دهیم زیرا مغز انسان در طی میلیون‌ها سال تکامل یافته و تربیت شده است و اکنون قادر به تشخیص چنین چیزی است. ما از عملیاتی که در پس‌زمینه مغزمان صورت می‌گیرد و ما را قادر می‌سازد رنگ‌های درون تصویر را تشخیص دهیم، اطلاعی نداریم. مغز ما، به نحوی آموزش دیده است که به صورت خودکار، این کار را برای ما انجام دهد.

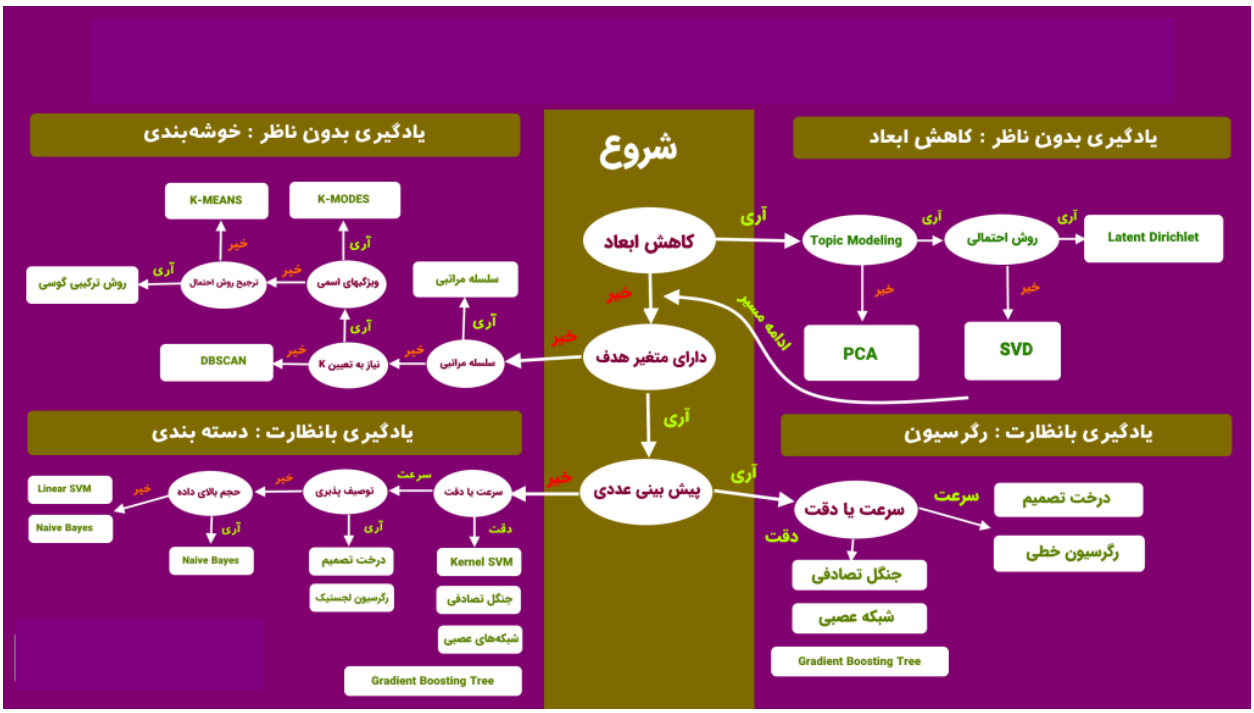
اگر بخواهیم کاری کنیم که یک کامپیوتر نیز قادر به انجام چنین چیزی باشد، آسان نیست. این مسئله، یکی از حوزه‌های تحقیقاتی فعال در علم دادها و یادگیری ماشین است. قبل از کار بر روی تشخیص ویژگی‌های یک تصویر، بگذارید بر روی یک سؤال خاص تمرکز کنیم: چگونه تصویری مانند مثال بالا به همراه ویژگی‌های مختلف، می‌تواند در یک کامپیوتر ذخیره شود؟ این کار با ذخیره‌سازی شدت پیکسل‌ها در ساختاری به نام «ماتریس» صورت می‌گیرد. به این ترتیب، هر عملی که بخواهید بر روی این تصویر انجام دهید، به احتمال زیاد از قواعد جبر خطی و ماتریس‌ها در پس‌زمینه خود استفاده خواهد کرد.

: (II)

الگوریتم ها و کتابخانه های متعددی در زبانهای برنامه نویسی بر پایه جبرخطی کار میکنند.

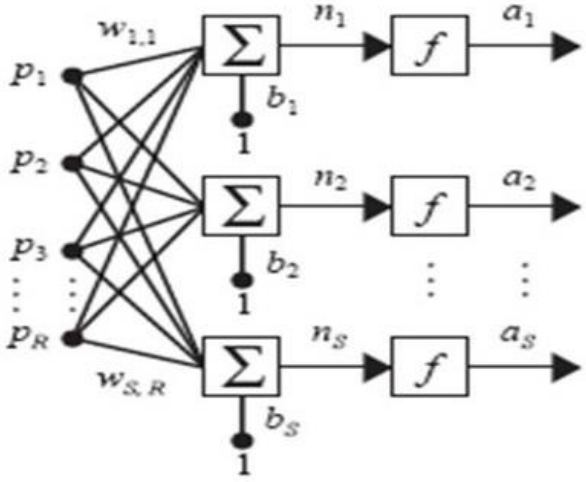
اگر به نحوی با حوزه علم داده‌ها آشنا باشید، ممکن است نام الگوریتم XGBOOST

را شنیده باشید که توسط بسیاری از افراد موفق در علم داده‌ها، به کار گرفته می‌شود و یک الگوریتم یادگیری ماشین براساس تقویت درخت موازی است که به عنوان فریم وورکی برای تقویت گرادیان (الگوریتم دیگری که برای مسائل رگرسیون و طبقه بندی به کار میرود) استفاده میشود و برای ارائه پیش‌بینی‌ها، داده‌های عددی را در فرم ماتریسی ذخیره می‌کند. این کار، باعث تقویت سرعت پردازش داده و نتایج دقیق‌تر می‌شود. به علاوه، بسیاری از الگوریتم‌های مختلف دیگر نیز از ماتریس‌ها برای ذخیره و پردازش داده‌ها استفاده می‌کنند.



: (III)

یادگیری عمیق، یک عبارت فراگیر جدید در حوزه به کارگیری ماتریس‌ها به منظور ذخیره ورودی‌هایی مانند تصویر، گفتار یا متن و ارائه یک راه‌حل برای مسائل این‌چنینی است. وزن‌های به‌دست‌آمده توسط یک شبکه عصبی نیز در ماتریس‌ها ذخیره می‌شوند. در تصویر زیر، نمونه‌ای از نمایش گرافیکی وزن‌های ذخیره‌شده در یک ماتریس نشان داده شده است.



$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

سوالی که پیش می آید: هوش مصنوعی (AI) ، یادگیری عمیق و یادگیری ماشین ؟

این اصطلاحات گاهی با یکدیگر هم‌پوشانی پیدا می‌کنند و گیج‌کننده می‌شوند و افراد هرکدام را به‌ازای دیگری به کار می‌برند.

AI یعنی کامپیوتری که به نحوی رفتار انسان را تقلید کند.

یادگیری ماشین زیرمجموعه‌ای از AI است و شامل تکنیک‌هایی می‌شود که کامپیوتر را قادر می‌سازد تا خودش داده را دریابد و از چیزها سردربیاورد و برنامه‌های کاربردی AI را ارائه دهد.

یادگیری عمیق زیرمجموعه‌ای ازیادگیری ماشین است که کامپیوترها را قادر می‌سازد تا مسائل پیچیده‌تری را حل کنند.

: (IV)

یکی دیگر از زمینه‌های تحقیقاتی فعال در حوزه یادگیری ماشین، نحوه کار با متن است. رایج‌ترین تکنیک‌های مورد استفاده در این زمینه، "کیسه کلمات"، "ماتریس لغت - سند" و... هستند.

تمام این تکنیک‌ها، به صورت بسیار مشابهی عمل می‌کنند. به این صورت که شمارش کلمات در اسناد را انجام داده و به منظور اجرای عملیاتی مانند تحلیل معنایی، ترجمه زبان، خلق زبان و غیره، تعداد تکرار کلمات را در یک فرم ماتریسی ذخیره می‌کنند.

پس دیدیم که جبرخطی ابزار قدرتمندی برای حل مسائل، شبیه سازی و ساده سازی مسائل در اختیار ما میگذارد (در ادامه مثال هایی از استفاده جبرخطی برای شبیه سازی و حل مسائل مختلف، مشاهده میکنیم)اکنون می‌توانید میزان اهمیت جبر خطی در یادگیری ماشین را درک کنید. مشاهده کردیم که تصویر، متن یا هر داده دیگری، از ماتریس‌ها برای ذخیره‌سازی و پردازش داده‌ها استفاده می‌کنند. همین موضوع می‌تواند انگیزه کافی برای یادگیری جبر خطی را در افراد علاقه مند به این زمینه به وجود بیاورد و اهمیت این شاخه از ریاضیات را در این زمینه نشان دهد.

❖ نمایش مسائل با استفاده از جبرخطی:

جبرخطی ابزار متعددی برای حل مسائل و شبیه سازی آنها در اختیار ما میگذارد. این ابزار عبارتند از: اسکالر ها، بردار ها، صفحه ها، ماتریس ها، تبدیلات خطی ، تنسور (عنصری هندسی است که در ریاضی و فیزیک به منظور گسترش
مفاهیم اسکالرها، بردارها و ماتریس‌ها به ابعاد بالاتر معرفی می‌شوند)و...

: (I)

بیایید تعریف مسائل جبر خطی را با یک مثال ساده شروع کنیم. فرض کنید قیمت یک توپ و دو راکت یا یک راکت و دو توپ، 100 واحد باشد. حال باید قیمت یک توپ و یک راکت را بیابیم. قیمت یک راکت را با «x» و قیمت یک توپ را «y» نشان می‌دهیم. مقدار x و y بسته به شرایط می‌تواند هر چیزی باشد. این یعنی x و y متغیر هستند.

فرم ریاضی این مسئله را به صورت زیر می‌نویسیم:

$$(1)\ 2x + y = 100$$

حالت دوم را نیز به همین صورت می‌نویسیم:

$$(2)\ x + 2y = 100$$

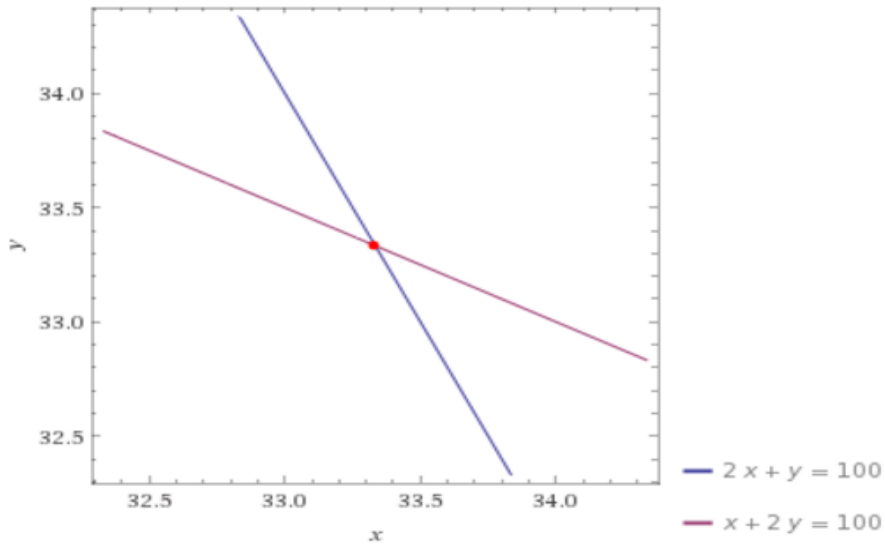
حال برای پیدا کردن قیمت توپ و راکت، باید مقادیر x و y را طوری به دست بیاوریم که در هر دو معادله صدق کنند. مسئله اصلی در جبر خطی، پیدا کردن مقادیر x و y است (حل یک دستگاه معادلات خطی). به طور کلی در جبر خطی، داده‌ها به صورت معادلات خطی نشان داده می‌شوند. این معادلات خطی نیز در فرم ماتریسی و برداری نشان داده می‌شوند. تعداد متغیرها و همچنین تعداد معادلات با توجه به شرایط تغییر می‌کنند اما نمایش آن‌ها به صورت ماتریسی و برداری خواهد بود.

: (II)

تصویرسازی مسئله

معمولاً رسم گرافیکی و تصویرسازی، به حل تصویرسازی مسائل کمک می‌کند. معادلات خطی، اشیا مسطح را نشان می‌دهند. برای درک بهتر، از ساده‌ترین نوع این اشیا شروع می‌کنیم. خط مربوط یک معادله، مجموعه‌ای از نقاطی است که در معادله داده‌شده صدق می‌کنند. به عنوان مثال، نقاطی هستند که در معادله 1 ما صدق می‌کنند. در نتیجه، این نقاط باید بر روی خط مرتبط با معادله 1 ما قرار گیرند. به همین صورت نقاطی هستند که در معادله 2 صدق می‌کنند.

حال می‌خواهیم نقاطی را پیدا کنیم که در هر دو معادله صدق کنند. برای این کار، باید نقطه‌ای را پیدا کنیم که در هر دو معادله صدق می‌کند. از نظر گرافیکی، ما به دنبال تقاطع دو خط معرف معادلات 1 و 2 هستیم (تصویر پایین).



بگذارید با استفاده از عملیات ابتدایی جبری مانند جمع، تفریق و جایگذاری، این مسئله را حل کنیم:

(1) 2x + 2y = 100

حالت دوم را نیز به همین صورت می‌نویسیم:

(2) X + 2y = 100

از معادله 1 داریم:

y = (100 – x) / 2

مقدار y را درون معادله 2 قرار می‌دهیم:

(3) X + 2 * (100 – x) / 2 = 100

از آنجایی که معادله 3 یک معادله یک مجهولی است، برای x و y قابل حل است.

این مسئله آسان بود. اکنون به یک مرحله بالاتر می‌رویم.

: (III)

معرفی مسائل پیچیده‌تر

فرض کنید یک دستگاه با سه معادله و سه مجهولی (مانند زیر) به شما داده شده است و می‌خواهید مقادیر تمام متغیرها را به دست بیاورید. بیایید این مسئله را با هم حل کنیم:

(4) $x + y + z = 1$

(5) $2x + y = 1$

(6) $5x + 3y + 2z = 4$

از معادله 4 داریم:

(7) $z = 1 – x – y$

در معادله 6 داریم z با جایگذاری:

$5x + 3y + 2(1 – x – y) = 4$

(8) $3x + y = 2$

اکنون می‌توان معادله 8 و 5 را برای به دست آوردنx وy به صورت یک دستگاه با دو معادله و دو مجهول، مانند مثال توپ و راکت حل کرد. بعد از بدست آوردن مقادیر xوy می‌توان از معادله 7 برای به دست آوردن مقدار z استفاده کرد.

همان‌طور که مشاهده کردید، با اضافه کردن یک متغیر دیگر، تلاش برای پیدا کردن حل مسئله به میزان زیادی افزایش یافت. حل همزمان 10 معادله، بسیار دشوار، خسته‌کننده و زمان‌بر خواهد بود

فرض کنید، میلیون‌ها داده در یک مجموعه داده واقعی داریم. استفاده از رویکرد بالا برای به دست آوردن جواب، بیشتر شبیه به یک کابوس خواهد بود تا یک راه‌حل. حال فرض کنید که قرار است این کار را چندین و چند بار انجام دهیم. مطمئناً، سال‌های طول می‌کشد تا این مسئله را حل کنیم. این حجم از کار تنها بخشی از مسائل مرتبط با علوم داده و یادگیری ماشین است. چه کار باید انجام دهیم؟ به نظر شما، باید حل این‌گونه مسائل را به کلی کنار بگذاریم؟ پاسخ منفی است.

برای حل یک دستگاه معادلات خطی بزرگ، از ماتریس استفاده می‌کنیم. قبل از اینکه به سراغ تعریف ماتریس و ویژگی‌های آن برویم، بیایید تصویر فیزیکی مسئله خود را در نظر بگیریم. برای این کار، نیاز به تعاریفی داریم که در بخش های بعدی به آن‌ها می‌پردازیم.

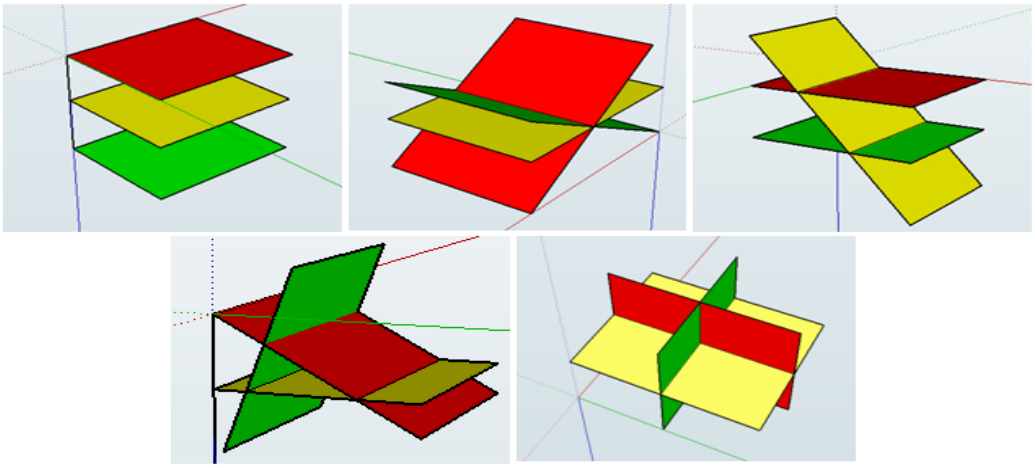
: (IV)

صفحه

یک معادله خطی با سه متغیر (سه مجهولی)، مجموعه‌ای از نقاطی را نشان می‌دهد که در معادلات مربوط به خود صدق می‌کنند. آیا می‌توانید، شیء فیزیکی معرف چنین معادله‌ای را حدس بزنید؟ سعی کنید شیء معرف یک معادله با دو متغیر را در نظر بگیرید (خط) و سپس، متغیر سوم را به آن اضافه کنید. باید متوجه شده باشید که با اضافه شدن متغیر سوم، معادل سه‌بعدی خط به وجود می‌آید.

اساساً، یک معادله خطی با سه متغیر، معرف یک صفحه است. به طور ساده، یک صفحه، شیء هندسی مسطحی است که تا بی‌نهایت گسترش می‌یابد. برای پیدا کردن جواب‌های دستگاه معادلات خطی سه مجهولی، باید تقاطع بین صفحات معرف هر معادله را پیدا کنیم (مانند کاری که برای خط انجام دادیم). به طور کلی، چهار حالت برای نحوه تقاطع بین سه صفحه وجود دارد:

1. تقاطعی با هم ندارند.
2. هر سه در یک خط تقاطع می‌یابند
3. در یک صفحه باهم تقاطع می‌یابند.
4. در یک نقطه باهم تقاطع می‌یابند.



انسان‌ها، تنها توان تصور سه بعدی اشیاء مختلف را دارند و تصور اشیاء در 4 یا 10,000 بعد، برایشان غیرممکن است. با این حال، ریاضیدانان توسط به کارگیری ترفندهایی، با داده‌هایی با ابعاد بالا به راحتی کار می‌کنند؛ یکی از این ترفندها، استفاده از ماتریس است.

: (IV)

ماتریس

ماتریس، روشی برای نوشتن مؤلفه‌های مشابه در کنار یکدیگر است تا بتوان با توجه به موارد مورد نیاز، آن‌ها را به راحتی بررسی و تغییرات لازم را ایجاد کرد. به طور کلی در علم داده‌ها، به منظور ذخیره اطلاعاتی مانند وزن‌های شبکه عصبی مصنوعی در حین آموزش الگوریتم‌های مختلف، از ماتریس استفاده می‌شود.

اساساً ماتریس، یک آرایه دوبعدی از اعداد است (البته تا جایی که به بحث ما مرتبط می شود). میدانیم که سطرها را با «i» و ستون‌ها را با «j» علامت‌گذاری می‌کنند. درایه‌ها با سطر i ام و ستون j ام نمایه می‌شوند. نام‌گذاری ماتریس‌ها نیز به وسیله حروف الفبای لاتین صورت می‌گیرد. مثال: ماتریس A با درایه‌های Aij.

جمع دو ماتریس، ضرب دو ماتریس، ضرب یک اسکالر در ماتریس، ماتریس ترانهاده و ... از مباحث پرکاربرد ماتریس در یادگیری ماشین هستند.

- ❖ استفاده از جبرخطی در یادگیری ماشین به طور عملی:

حال جبرخطی را به طور ملموس در یک زبان برنامه نویسی بررسی میکنیم. یکی از مزایای زبان پایتون، متن باز(open source) بودن آن است که قابلیت اضافه شدن کتابخانه های جدید و متنوع را به این زبان می دهد. لیست زیر، کتابخانه هایی از پایتون هستند که برای برنامه نویسی در زمینه یادگیری ماشین، با استفاده از ابزار ریاضی پیشرفته، وجود دارند:

سایکیت لِرن(scikit-learn)

کِرَس(Keras)

XGBoost

StatsModels

LightGBM

CatBoost

PyBrain

Eli5

Numpy

در این قسمت رابطه تنگاتنگ یادگیری ماشین و جبرخطی را به طور خاص در زبان برنامه نویسی پایتون و کتابخانه NumPy بررسی میکنیم.

: (I)

:NumPy

همان طور که قبل تر هم گفتیم، یادگیری ماشین شاخه‌ای از علوم کامپیوتر است که در آن سعی می‌کنیم توانایی یادگیری را در ماشین (کامپیوتر) پیاده‌سازی کنیم؛ به این ترتیب یک کامپیوتر می‌تواند با دیدن یک سری الگوها، آن الگو را یاد بگیرد و در تشخیص موارد بعدی، موثر عمل کند. البته اگر بخواهیم یادگیری ماشین را دقیق‌تر تعریف کنیم، باید بگوییم: یادگیری ماشین یک حوزه مطالعاتی است که به کامپیوترها امکان یادگیری را می‌دهد بدون آن که نیاز باشد به شکل صریح و واضح کدنویسی شوند.

نامپای یا numpy یک کتابخانه پایتون است که برای کار با آرایه‌ها (تعدادی متغیر از یک نوع داده و تحت یک نام می‌باشد.) به وجود آمده است. کتابخانه numpy همچنین توابعی برای انجام عملیات‌های گوناگون در جبر خطی، تبدیل فوریه و ماتریس‌ها دارد. نامپای در سال ۲۰۰۵ توسط تراویس الیفانت (Travis Oliphant) و به صورت یک پروژه متن باز (open source) ایجاد شد .Numpy سرواژه‌ی عبارت Numerical Python به معنای پایتون عددی یا پایتون محاسباتی است.

یک آرایه NumPy ویژگی هایی از قبیل محاسبات ریاضی، محاسبات منطقی، تغییر شکل آرایه ها، مرتب سازی، جبر خطی، محاسبات آماری و ... را نیز در اختیار شما می گذارد. از ویژگی های دیگر کتابخانه NumPy می توان به موارد زیر اشاره نمود:

1. تمامی پارامترهای یک آرایه در کتابخانه NumPy قابلیت امکان تعریف داده های متفاوت در یک آرایه را دارا می باشند.
2. آرایه NumPy بر خلاف ساختمان داده (List) لیست در پایتون (Python) اندازه ثابتی در هنگام ساخته شدن دارد، و تغییر در اندازه این آرایه منجر به ساخته شدن یک آرایه جدید و پاک شدن آرایه قبلی خواهد شد.
3. استفاده از کتابخانه NumPy در نهایت منجر به بهینه شدن و کاهش زمان اجرای برنامه شما خواهد شد زیر این کتابخانه در زبان پایتون به عنوان یکی از کتابخانه های بهینه جهت کار با آرایه ها تنظیم شده است. آرایه NumPy قادر به انجام بسیاری از عملیات آماری و ریاضی بر روی داده ها زیاد، به صورت کاملا کارامد و بهینه با خطوط بسیار کمی از کد هستند.

هسته کتابخانه NumPy در زبان برنامه نویسی پایتون ndarray (Python) است. این آرایه چند بعدی که انوع مختلفی از داده را می‌تواند ذخیره نماید، بسیار کارآمد و بهینه طراحی شده که در بالا به تفاوت های اصلی آن با ساختمان داده لیست در پایتون اشاره شد.

چرا باید از numpy استفاده کنیم؟

در پایتون چیزی به عنوان آرایه وجود ندارد؛ با این حال می‌توان از لیست (list) به عنوان آرایه استفاده کرد. مشکل لیست آن است که سرعت پردازش داده‌ها در آن بسیار پایین است Numpy . تلاش دارد شیئی را به عنوان آرایه ارائه دهد که ۵۰ برابر از لیست سریع‌تر است. شیئی که به عنوان آرایه در numpy موجود است، ndarrayنام دارد. نامپای توابع زیادی دارد که کار با ndarrayرا بسیار راحت کرده‌اند. توجه داشته باشید که با توجه به استفاده‌ی گسترده از آرایه‌ها در علوم داده و با توجه به حجیم بودن داده‌ها، سرعت مقوله‌ی بسیار مهمی برای ماست.

چرا کتابخانه numpy از لیست سریع‌تر است؟

کتابخانه نامپای، داده‌های موجود در آرایه را در خانه‌هایی پشت سر هم از حافظه ذخیره می‌کند؛ در مقابل، ذخیره‌سازی داده‌ها در لیست این گونه نیست؛ در واقع لیست، هر یک از آیتم‌ها را در محلی ذخیره می‌کند و آیتم‌های مختلف موجود در لیست، الزاما در خانه‌های پشت سر هم حافظه

قرار ندارند. بنابراین، پردازنده می‌تواند به داده‌های موجود در آرایه‌ی ساخته‌شده به وسیله‌ی numpy، به سرعت دست یابد و به طور بهینه، عملیات‌های مورد نیاز را انجام دهد. در علوم کامپیوتر، به چنین رفتاری محلی بودن مرجع یا locality of reference گفته می‌شود. مطلبی که بیان شد، علت اصلی سریع‌تر بودن numpy از لیست در پایتون است؛ علل دیگری هم وجود دارند؛ به عنوان مثال، کتابخانه numpy به گونه‌ای ساخته شده است که برای کار با آخرین معماری پردازنده‌های مدرن، بهینه باشد.

: (II)

مفاهیم پایه:

هدف اصلی NumPy فراهم ساختن امکان کار با آرایه‌های چندبعدی همگن است. این آرایه‌ها جدولی از عناصر (معمولاً اعداد) هستند که همگی از یک نوع می‌باشند و با یک چندتایی، از اعداد صحیح مثبت اندیس‌گذاری می‌شوند. در NumPy ابعاد به نام محور (axe) شناخته می‌شوند. تعداد محورها رتبه (rank) نامیده می‌شود.

برای مثال، مختصات یک نقطه در فضای 3 بعدی [1, 2, 1] یک آرایه با رتبه 1 است زیرا یک محور دارد. این محور طولی به‌اندازه 3 دارد. در مثال زیر آرایه رتبه 2 دارد (2 بعدی است). بعد (محور) نخست طولی به اندازه 2 دارد، بعد دوم طول 3 دارد.

```
[[ 1., 0., 0.],
[ 0., 1., 2.]]
```

ndarray.ndim: تعداد محور (ابعاد) آرایه است. در دنیای پایتون تعداد ابعاد به‌صورت رتبه نامیده می‌شود.

ndarray.shape: ابعاد یک آرایه است. این خصوصیت از یک چندتایی اعداد صحیح تشکیل یافته است که نشان‌دهنده اندازه هر بعد آرایه هستند. برای یک ماتریس با n ردیف و m ستون، شکل (shape) به‌صورت (n,m)خواهد بود. بدین ترتیب طول چندتایی shape برابر با رتبه آرایه یا تعداد ابعاد ndim است.

ایجاد آرایه

چند روش برای ایجاد آرایه وجود دارند. برای مثال، می‌توان با استفاده از تابع array یک آرایه را از فهرست معمولی پایتون یا چندتایی‌ها ایجاد کرد. نوع آرایه حاصل، برابر با نوع عناصر موجود در دنباله‌های تشکیل دهنده آن خواهد بود .

```
>>> from numpy import *
>>> a = array( [2,3,4] )
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int32')
>>> b = array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

یکی از خطاهای رایج در کار کردن با آرایه‌های چندبعدی زمانی رخ می‌دهد که قصد داریم array را با چند آرگومان عددی فراخوانی کنیم، در حالی که باید از فهرست منفردی از اعداد به عنوان آرگومان استفاده کنیم .

```
>>> a = array(1,2,3,4) # اشتباه
>>> a = array([1,2,3,4]) # صحیح
```

arrayدنباله‌ای از دنباله‌ها را به آرایه‌های چندبعدی تبدیل می‌کند، دنباله‌ای از دنباله‌های دنباله‌ها به آرایه‌های سه‌بعدی تبدیل می‌شود و همین‌طور تا آخر .

```
>>> b = array( [ (1.5,2,3), (4,5,6) ] )
>>> b
array([[ 1.5, 2. , 3. ],
[ 4. , 5. , 6. ]])
```

پرینت کردن آرایه‌ها

زمانی که یک آرایه را پرینت می‌کنید NumPy آن را به صوت یک فهرست تودرتو نمایش می‌دهد که طرح کلی آن به‌صورت زیر است:

- آخرین محور از چپ به راست پرینت می‌شود.
- محور ماقبل آخر از بالا به پایین پرینت می‌شود.

- باقی محورها نیز از بالا به پایین پرینت و هرکدام با یک خط خالی از قبلی جدا می‌شوند.

بدین ترتیب آرایه‌های تک‌بعدی به‌صورت ردیفی، آرایه‌های دوبعدی به‌صورت ماتریس و آرایه‌های سه‌بعدی به‌صورت فهرستی از ماتریس‌ها پرینت می‌شوند.

```
>>> a = arange(6) # 1d آرایه
>>> print a
[0 1 2 3 4 5]
>>>
>>> b = arange(12).reshape(4,3) # 2d آرایه
>>> print b
[[ 0 1 2]
 [ 3 4 5]
 [ 6 7 8]
 [ 9 10 11]]
>>>
```

```
>>> c = arange(24).reshape(2,3,4) # 3d آرایه
>>> print c
[[[ 0 1 2 3]
  [ 4 5 6 7]
  [ 8 9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

: (III)

عملیات پایه در این کتابخانه

عملیات‌های حسابی بر روی آرایه‌ها در سطح عناصر انجام می‌یابند. درنتیجه اجرای عملیات حسابی یک آرایه جدید ایجاد و مقادیر آن پر می‌شود .

```
>>> a = array( [20,30,40,50] )
>>> b = arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10*sin(a)
array([ 9.12945251, -9.88031624, 7.4511316 , -2.62374854])
>>> a<35
array([True, True, False, False], dtype=bool)
```

برخلاف بسیاری از زبان‌های ماتریسی عملگر * در آرایه‌های NumPy به‌صورت عنصر به عنصر، عمل ضرب را انجام می‌دهد. ضرب ماتریسی را می‌توان با استفاده از تابع dot یا ایجاد اشیای matrix انجام داد .

```
>>> A = array( [[1,1],
... [0,1]] )
>>> B = array( [[2,0],
... [3,4]] )
>>> A*B # ضرب در سطح عناصر
array([[2, 0],
[0, 4]])
>>> dot(A,B) # ضرب در سطح ماتریس
array([[5, 4],
[3, 4]])
```

برخی عملیات‌ها مانند =+ و =* به‌جای ایجاد یک آرایه جدید بر روی همان ماتریس موجود عمل می‌کنند .

```
>>> a = ones((2,3), dtype=int)
>>> b = random.random((2,3))
>>> a *= 3
>>> a
array([[3, 3, 3],
[3, 3, 3]])
>>> b += a
>>> b
array([[ 3.69092703, 3.8324276 , 3.0114541 ],
[ 3.18679111, 3.3039349 , 3.37600289]])
>>> a += b # به مقدار صحیح تبدیل می‌شود b
>>> a
array([[6, 6, 6],
[6, 6, 6]])
```

این عملیات‌ها به طور پیش‌فرض طوری بر روی آرایه‌ها اجرا می‌شوند که صرف‌نظر از شکلشان، گویی آرایه‌ها فهرستی از اعداد هستند. بااین‌حال با تعیین پارامتر axis می‌توان یک عملیات را در راستای یک محور تعیین شده در یک آرایه اجرا کرد:

```
>>> b = arange(12).reshape(3,4)
>>> b
array([[ 0, 1, 2, 3],
[ 4, 5, 6, 7],
[ 8, 9, 10, 11]])
>>>
>>> b.sum(axis=0) # sum of each column
array([12, 15, 18, 21])
>>>
>>> b.min(axis=1) # min of each row
array([0, 4, 8])
>>>
>>> b.cumsum(axis=1) # cumulative sum along each row
array([[ 0, 1, 3, 6],
[ 4, 9, 15, 22],
[ 8, 17, 27, 38]])
```

: (IV)

مزایای دیگر کتابخانه نامپای:

همانطور که میدانیم، زمان اجرای برنامه و سرعت آن در برنامه های حجیم(مانند برنامه های یادگیری ماشین) از اهمیت بسیار بالایی برخوردار است. یکی از مزایایی که کتابخانه نامپای را گزینه مناسبی برای کار کردن در زمینه یادگیری ماشین میکند، استفاده از الگوریتم کوپراسمیت-وینوگارد است که یکی از سریع ترین الگوریتم های ضرب ماتریسی است (در سال 2010 توسط اندرو استوترز ، 2012 توسط ویرجینیا ویلیامز و 2014 توسط فرانسوا لوگل الگوریتم هایی ارائه شد که با اختلاف کمی سریع تر بودند) الگوریتم کوپراسمیت-وینوگارد دارای اردر زمانی $\mathcal{O}(n^{2.375477})$ است که در مقایسه با الگوریتم ضرب

عادی با اردر زمانی $\mathcal{O}(n^3)$ و الگوریتم استراسن با اردر زمانی $\mathcal{O}(n^{2.807355})$ ، زمان بهتری را در اختیار ما قرار میدهد.

علاوه بر این همانطور که دیدیم، این کتابخانه ابزار قوی، متنوع و پیشرفته ای را برای کار با اعداد با استفاده از جبرخطی برای ما فراهم می آورد و کار ما را در برنامه نویسی برای یادگیری ماشین، روان و ساده میکند.(کتابخانه های متنوعی در پایتون برای کار با اعداد وجود دارد اما رویکرد این کتابخانه بسیار عمیق تر و دارای موارد پیشرفته تر و متنوع تر است.)

جمع بندی:

پس میتوان گفت استفاده از جبرخطی، مزایا و استفاده های زیادی در برنامه نویسی و به طور خاص، یادگیری ماشین دارد.

مراجع و منابع:

-Basics of Linear Algebra for Machine Learning by Jason Brownlee

-https://www.analyticsvidhya.com/blog/2017/05/comprehensive-guide-to-linear-algebra/

-https://dl.acm.org/doi/abs/10.1145/2213977.2214056

-Coppersmith, Don; Winograd, Shmuel (1990), "Matrix multiplication via arithmetic progressions"

-Stothers, Andrew (2010), *On the Complexity of Matrix Multiplication* (Ph.D thesis), University of Edinburgh.

-Williams, Virginia Vassilevska (2011), Breaking the Coppersmith-Winograd barrier

-"Le Gall, François (2014), "Powers of tensors and fast matrix multiplication", Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (*ISSAC* 2014)

-https://www.kdnuggets.com/2018/10/top-python-machine-learning-libraries.html