**1.4 پیش تخصیص آرایه‌های NumPy**

**مسئله**

شما باید آرایه‌هایی با اندازه معین را با مقداری از قبل تخصیص دهید.

**راه‌حل**

NumPy دارای توابعی برای تولید بردارها و ماتریس‌هایی با هر اندازه با استفاده از 0، 1 یا مقادیر دلخواه شما است:



**بحث**

تولید آرایه‌هایی که از قبل با داده‌ها پر شده‌اند، برای چندین هدف مفید است، مانند عملکرد بهتر کد یا استفاده از داده‌های مصنوعی برای آزمایش الگوریتم‌ها. در بسیاری از زبان‌های برنامه نویسی، پیش تخصیص‌یک آرایه از مقادیر پیش فرض (مانند 0s)‌یک روش معمول در نظر گرفته می‌شود.

**2.6 بارگذاری‌یک فایل پارکت[[1]](#footnote-1)**

**مسئله**

شما باید‌یک فایل پارکت را بارگذاری کنید.

**راه‌حل**

تابع read\_parquet از کتابخانه‌یpandas به ما این امکان می‌دهد که فایل‌های Parquet را بخوانیم:



|  |  |  |  |
| --- | --- | --- | --- |
|  | دسته بندی | تاریخ و زمان | عدد |
| 0 | 0 | 00:00:00 2015-01-01 | 5 |
| 1 | 0 | 00:00:01 2015-01-01 | 5 |

**بحث**

Parquet‌یک فرمت محبوب ذخیره سازی داده در فضای بزرگ داده است. اغلب با ابزارهای داده‌های بزرگ مانند Hadoop و Spark مورد استفاده قرار می‌گیرد. در حالی که PySpark خارج از تمرکز این کتاب است، به احتمال زیاد شرکت‌هایی که در مقیاس بزرگ فعالیت می‌کنند از‌یک قالب ذخیره‌سازی داده کارآمد مانند پارکت استفاده می‌کنند، و دانستن نحوه خواندن آن در‌یک دیتافریم و دستکاری آن بسیار ارزشمند است.

**همچنین ببینید:**

* [مستندات پارکت آپاچی[[2]](#footnote-2)](https://parquet.apache.org/)

**2.7 بارگیری‌یک فایل Avro**

**مسئله**

شما باید‌یک فایل Avro را در دیتافرِیم pandas بارگذاری کنید.

**راه‌حل**

از تابع read\_avro در کتابخانه‌ی pandavro استفاده کنید:



|  |  |  |  |
| --- | --- | --- | --- |
|  | دسته بندی | تاریخ و زمان | عدد |
| 0 | 0 | 00:00:00 2015-01-01 | 5 |
| 1 | 0 | 00:00:01 2015-01-01 | 5 |

**بحث**

Apache Avro‌یک فرمت داده باینری و منبع باز است که برای ساختار داده بر طرح‌واره‌ها متکی است. در زمان نوشتن، به اندازه پارکت رایج نیست. با این حال، فرمت‌های داده‌های باینری بزرگ مانند Avro، Thrift و Protocol Buffer به دلیل ماهیت کارآمدشان در حال افزایش محبوبیت هستند. اگر با سیستم‌های داده بزرگ کار می‌کنید، احتمالاً در آینده نزدیک با‌یکی از این فرمت‌ها مواجه خواهید شد.

**همچنین ببینید:**

* [مستندات پارکت آپاچی[[3]](#footnote-3)](https://avro.apache.org/docs/1.2.0/)

**2.8 جستجو در پایگاه داده SQLite[[4]](#footnote-4)**

**مسئله**

شما باید داده‌ها را از‌یک پایگاه داده با استفاده از زبان SQL بارگیری کنید.

**راه‌حل**

read\_sql\_query از کتابخانه‌ی Pandas به ما این امکان را می‌دهد که‌یک دستور کوئری[[5]](#footnote-5) SQL در پایگاه داده ایجاد کرده و آن را بارگذاری کنیم:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | نام | نام خانوادگی | سن | امتیاز پیش‌آزمون |
| 0 | Json | Miller | 42 | 4 |
| 1 | Molly | Jaboson | 52 | 24 |

**بحث**

Apache SQL زبانی برای استخراج داده‌ها از پایگاه‌های داده است. در این دستور، ابتدا از create\_engine برای تعریف اتصال به موتور پایگاه داده SQL به نام SQLite استفاده می‌کنیم. در مرحله بعد ما از read\_sql\_query در کتابخانه‌ی Pandas برای کوئری زدن روی پایگاه داده با استفاده از زبان SQL و قرار دادن نتایج در‌یک DataFrame استفاده می‌کنیم.

SQL به خودی خود‌یک زبان است و اگرچه فراتر از محدوده این کتاب است، اما مطمئناً برای هر کسی که می‌خواهد در مورد یادگیری ماشینی بیاموزد، ارزش دارد. کوئری SQL ما، SELECT \* FROM data، از پایگاه داده می‌خواهد که تمام ستون‌های (\*) جدول را به نام عنوان داده به ما بدهد.

توجه داشته باشید که این‌یکی از معدود دستور العمل‌های این کتاب است که بدون کد اضافی اجرا نمی‌شود. به طور خاص، create\_engine('sqlite:///sample.db') فرض می‌کند که‌یک پایگاه داده SQLite از قبل وجود دارد.

* [SQLite](https://www.sqlite.org/index.html)
* [آموزش SQL در وبسایت W3Schools](https://www.w3schools.com/sql/)

**2.9 کوئری زدن روی‌یک پایگاه داده‌ی SQL از راه دور**

**مسئله**

شما باید به‌یک پایگاه داده SQL از راه دور متصل شوید و داده‌ها را از آن بخوانید.

**راه‌حل**

یک اتصال با pymysql ایجاد کنید و آن را با Pandas در‌یک DataFrame بخوانید:



|  |  |  |  |
| --- | --- | --- | --- |
|  | عدد | تاریخ و زمان | دسته بندی |
| 0 | 5 | 00:00:00 2015-01-01 | 0 |
| 1 | 5 | 00:00:01 2015-01-01 | 0 |

**بحث**

از بین تمام دستور العمل‌های ارائه شده در این فصل، احتمالاً این دستور العملی است که بیشتر در دنیای واقعی استفاده خواهیم کرد. در حالی که اتصال و خواندن از‌یک پایگاه داده نمونه‌ی sqlite مفید است، احتمالاً نماینده‌ی جداولی نیست که باید در‌یک محیط سازمانی به آنها متصل شوید. اکثر نمونه‌های SQL که به آن‌ها متصل می‌شوید، از شما می‌خواهند که به میزبان و پورت‌یک دستگاه از راه دور متصل شوید، و‌یک نام کاربری و رمز عبور برای احراز هویت مشخص کنید. این مثال از شما می‌خواهد که‌یک نمونه SQL در حال اجرا را [به صورت محلی راه‌اندازی کنید](https://github.com/kylegallatin/mysql-db-example) که از‌یک سرور راه دور در لوکال‌هاست تقلید می‌کند تا بتوانید حسی از گردش کار دریافت کنید.

**همچنین ببینید:**

* [مستندات PyMySQL](https://pymysql.readthedocs.io/en/latest/user/examples.html)
* [مستندات Read SQL در کتابخانه‌ی Pandas](https://pandas.pydata.org/docs/reference/api/pandas.read_sql.html)

**2.10 بارگیری داده‌ها از Google Sheet**

**مسئله**

شما باید داده‌ها را مستقیماً از Google Sheet بخوانید.

**راه‌حل**

از read\_CSV در کتابخانه‌ی pandas استفاده کنید و نشانی اینترنتی ارسال کنید که Google Sheet را به‌عنوان CSV صادر می‌کند:



|  |  |  |  |
| --- | --- | --- | --- |
|  | عدد | تاریخ و زمان | دسته بندی |
| 0 | 5 | 00:00:00 2015-01-01 | 0 |
| 1 | 5 | 00:00:01 2015-01-01 | 0 |

**بحث**

در حالی که Google Sheets را می‌توان به راحتی دانلود کرد، گاهی اوقات مفید است که بتوانید آنها را مستقیماً در پایتون بدون هیچ مرحله میانی بخوانید. پارامتر کوئری /export?format=csv در انتهای URL بالا‌یک نقطه پایانی ایجاد می‌کند که می‌توانیم فایل را دانلود کنیم یا در Pandas بخوانیم.

**همچنین ببینید:**

* [Google Sheets API](https://developers.google.com/sheets/api)

**2.11 بارگیری داده‌ها از سطل[[6]](#footnote-6) S3**

**مسئله**

شما باید‌یک فایل CSV را از‌یک سطل S3 که به آن دسترسی دارید بخوانید.

**راه‌حل**

گزینه‌های ذخیره سازی را به Pandas اضافه کنید تا به شیء[[7]](#footnote-7) S3 دسترسی داشته باشد:



|  |  |  |  |
| --- | --- | --- | --- |
|  | عدد | تاریخ و زمان | دسته بندی |
| 0 | 5 | 00:00:00 2015-01-01 | 0 |
| 1 | 5 | 00:00:01 2015-01-01 | 0 |

**بحث**

اکنون بسیاری از شرکت‌ها داده‌ها را در فروشگاه‌های blob ارائه‌دهنده ابری مانندAmazon S3 یاGoogle Cloud Storage(GCS) نگهداری می‌کنند. معمولاً متخصصان یادگیری ماشینی برای بازیابی داده‌ها به این منابع متصل می‌شوند. اگرچه S3 URI (s3://machine-learning-python-cookbook/data.csv) عمومی‌است، اما همچنان از شما می‌خواهد که اعتبار دسترسی AWS خود را برای دسترسی به آن ارائه دهید. شایان ذکر است که اشیاء عمومی‌همچنین دارای URL‌های HTTP هستند که می‌توانند فایل‌ها را از آن دانلود کنند، مانند [این مورد برای فایل CSV](https://oreil.ly/byelc).

**همچنین ببینید:**

* [Amazon S3](https://aws.amazon.com/s3/)
* [اعتبارنامه امنیتی AWS](https://docs.aws.amazon.com/IAM/latest/UserGuide/security-creds.html)

**2.12 بارگذاری داده‌های بدون ساختار**

**مسئله**

شما باید داده‌های بدون ساختار مانند متن یا تصاویر را بارگیری کنید.

**راه‌حل**

از تابع باز پایتون برای بارگذاری اطلاعات استفاده کنید:



|  |  |  |  |
| --- | --- | --- | --- |
|  | دسته بندی | تاریخ و زمان | عدد |
| 0 | 0 | 00:00:00 2015-01-01 | 5 |
| 1 | 0 | 00:00:01 2015-01-01 | 5 |

**بحث**

در حالی که داده‌های ساختاریافته را می‌توان به راحتی از CSV، JSON یا پایگاه‌های داده مختلف خواند، داده‌های بدون ساختار می‌توانند چالش برانگیزتر باشند و ممکن است نیاز به پردازش سفارشی داشته باشند. گاهی اوقات باز کردن و خواندن فایل‌ها با استفاده از تابع باز اصلی پایتون مفید است. این مورد به ما امکان می‌دهد فایل‌ها را باز کنیم و سپس محتوای آن فایل را بخوانیم.

**همچنین ببینید:**

* [تابع باز پایتون](https://docs.python.org/3/library/functions.html#open)
* [مدیران context در پایتون](https://docs.python.org/3/library/contextlib.html)

**3.2 دریافت اطلاعات در مورد داده ها**

**مسئله**

شما می‌خواهید برخی از ویژگی‌های یک DataFrame را مشاهده کنید.

**راه‌حل**

یکی از ساده‌ترین کارهایی که می‌توانیم پس از بارگذاری داده‌ها انجام دهیم، مشاهده چند ردیف اول با استفاده از head است:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | نام | کلاس P | سن | جنسیت |
| 0 | Allen, Miss Elisabeth Walton | اول | 29.0 | زن |
| 1 | Allison, Miss Helen Loraine | اول | 2.0 | زن |

همچنین می‌توانیم به تعداد سطرها و ستون‌ها نگاهی بیندازیم:



ما می‌توانیم آمار توصیفی برای هر ستون عددی را با استفاده از describe بدست آوریم:



|  |  |  |  |
| --- | --- | --- | --- |
|  | سن | زنده مانده | کد جنسیت |
| تعداد | 756.000000 | 1313.000000 | 1313.000000 |
| میانگین | 30.397989 | 0.342727 | 0.351866 |
| std | 14.259049 | 0.474802 | 0.477734 |
| مینیمم | 0.170000 | 0.000000 | 0.000000 |
| 25% | 21.000000 | 0.000000 | 0.000000 |
| 50% | 28.000000 | 0.000000 | 0.000000 |
| 75% | 39.000000 | 1.000000 | 1.000000 |
| ماکزیمم | 71.000000 | 1.000000 | 1.000000 |

**علاوه بر این، روش info می‌تواند اطلاعات مفیدی را نشان دهد:**



**بحث**

پس از بارگیری برخی از داده ها، ایده خوبی است که بفهمیم ساختار آن چگونه است و چه نوع اطلاعاتی در آن وجود دارد. در حالت ایده آل، ما داده‌های کامل را مستقیماً مشاهده می‌کنیم. اما در بیشتر موارد دنیای واقعی، داده‌ها می‌توانند هزاران تا صدها هزار تا میلیون‌ها سطر و ستون داشته باشند. در عوض، برای مشاهده برش‌های کوچک و محاسبه آمار خلاصه داده‌ها باید به کشیدن نمونه تکیه کنیم.

در راه حل خود، ما از مجموعه داده اسباب بازی مسافران تایتانیک استفاده می‌کنیم. با استفاده از head، می‌توانیم به چند ردیف اول (پنج به طور پیش فرض) داده‌ها نگاه کنیم. از طرف دیگر، می‌توانیم از tail برای مشاهده چند ردیف آخر استفاده کنیم. با شکل می‌توانیم ببینیم که DataFrame ما شامل چند ردیف و ستون است. با توصیف می‌توانیم برخی از آمار توصیفی اولیه برای هر ستون عددی را ببینیم. و در نهایت، اطلاعات تعدادی از نقاط داده مفید را در مورد DataFrame نشان می‌دهد، از جمله انواع داده‌های شاخص و ستون، مقادیر غیر تهی و میزان استفاده از حافظه.

شایان ذکر است که آمار خلاصه، همیشه داستان کامل را بیان نمی‌کند. به عنوان مثال، Pandas با ستون‌های زنده‌مانده و کد جنسیتی به عنوان ستون‌های عددی رفتار می‌کنند زیرا دارای 1 و 0 هستند. با این حال، در این مورد مقادیر عددی نشان دهنده دسته‌ها هستند. به عنوان مثال، اگر Survived برابر با 1 باشد، نشان می‌دهد که مسافر از فاجعه جان سالم به در برده است. به همین دلیل، برخی از آمار خلاصه‌ی ارائه شده منطقی نیستند، مانند انحراف استاندارد ستون کد جنسیتی (نشانگر جنسیت مسافر).

**3.3 برش DataFrame ها**

**مسئله**

شما باید یک زیر مجموعه داده یا برش‌هایی از یک DataFrame را انتخاب کنید.

**راه‌حل**

از loc یا iloc برای انتخاب یک یا چند ردیف یا مقدار استفاده کنید:



می‌توانیم از : برای تعریف برش ردیف‌هایی که می‌خواهیم، استفاده کنیم. مانند انتخاب ردیف‌های دوم، سوم و چهارم:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | نام | کلاس P | سن | جنسیت |
| 1 | Allison, Miss Helen Loraine | اول | 2.0 | زن |
| 2 | Allison, Mr Hudson Joshua Creighton | اول | 30.0 | مرد |
| 3 | Allison, Mrs Hudson JC (Bessie Waldo Daniels) | اول | 25.0 | زن |

حتی می‌توانیم از آن برای به دست آوردن تمام ردیف‌ها تا یک نقطه استفاده کنیم. مانند همه ردیف‌ها تا ردیف چهارم:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | نام | کلاس P | سن | جنسیت |
| 0 | Allen, Miss Elisabeth Walton | اول | 29.0 | زن |
| 1 | Allison, Miss Helen Loraine | اول | 2.0 | زن |
| 2 | Allison, Mr Hudson Joshua Creighton | اول | 30.0 | مرد |
| 3 | Allison, Mrs Hudson JC (Bessie Waldo Daniels) | اول | 25.0 | زن |

**DataFrame‌ها نیازی به ایندکس شدن عددی ندارند. ما می‌توانیم ایندکس یک DataFrame را روی هر مقداری که مقدار آن برای هر ردیف منحصر به فرد باشد، تنظیم کنیم. برای مثال، می‌توانیم فهرست را به‌عنوان نام مسافران تنظیم کنیم و سپس ردیف‌ها را با استفاده از یک نام انتخاب کنیم:**



**بحث**

همه سطرها در DataFrame در کتابخانه‌ی Pandas دارای یک مقدار شاخص منحصر به فرد است. به طور پیش فرض، این شاخص یک عدد صحیح است که موقعیت ردیف را در DataFrame نشان می‌دهد. با این حال، لازم نیست که حضور داشته باشد. شاخص‌های DataFrame را می‌توان به صورت رشته‌های الفبایی منحصر به فرد یا شماره مشتری تنظیم کرد. Pandas برای انتخاب ردیف‌ها و تکه‌های ردیف‌ها دو روش ارائه می‌دهد:

* loc زمانی مفید است که نمایه DataFrame یک برچسب (به عنوان مثال، یک رشته) باشد.
* iloc با جستجوی موقعیت در DataFrame کار می‌کند. برای مثال، iloc[0] بدون در نظر گرفتن اینکه شاخص یک عدد صحیح است یا یک برچسب، ردیف اول را برمی‌گرداند.

استفاده زیاد و راحت بودن با loc و iloc مفید است زیرا هنگام پاکسازی داده‌ها زیاد ظاهر می‌شوند.

**3.5 مرتب سازی مقادیر**

**مسئله**

شما باید یک دیتافریم را بر اساس مقادیر یک ستون مرتب کنید.

**راه‌حل**

از تابع sort\_values در کتابخانه‌ی pandas استفاده کنید:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | نام | کلاس P | سن | جنسیت |
| 763 | Dean, Miss Elizabeth Gladys (Millvena) | سوم | 0.17 | زن |
| 751 | Danbom, Master Gilbert Sigvard Emanuel | سوم | 0.33 | مرد |

**بحث**

در طول تجزیه و تحلیل و کاوش داده ها، مرتب کردن یک DataFrame بر اساس یک ستون یا مجموعه‌ای از ستون‌های خاص اغلب مفید است. آرگومان by در sort\_values فهرستی از ستون‌ها را می‌گیرد که براساس آن DataFrame مرتب می‌شود و این مرتب سازی بر اساس ترتیب نام ستون‌ها در لیست مرتب است.

به‌طور پیش‌فرض، آرگومان accending روی True تنظیم شده است، بنابراین مقادیر پایین‌ترین به بالاترین را مرتب می‌کند. اگر ما مسن‌ترین مسافران را به جای جوان‌ترین مسافران می‌خواستیم، می‌توانیم آن را روی False تنظیم کنیم.

**3.16 تجمیع عملیات و آمار**

**مسئله**

شما باید یک عملیات را روی هر ستون (یا مجموعه‌ای از ستون ها) در یک دیتافریم جمع کنید.

**راه‌حل**

از روش agg در کتابخانه‌ی pandas استفاده کنید. در اینجا، ما به راحتی می‌توانیم حداقل مقدار هر ستون را بدست آوریم:



گاهی اوقات، ما می‌خواهیم توابع خاصی را به مجموعه‌های خاصی از ستون‌ها اعمال کنیم:



|  |  |  |
| --- | --- | --- |
|  | سن | جنسیت |
| میانگین | 30.397989 | Nan |
| مینیمم | Nan | 0.0 |
| ماکزیمم | Nan | 1.0 |

همچنین می‌توانیم توابع انبوه را برای گروه‌ها اعمال کنیم تا آمار توصیفی و خاص‌تری به دست آوریم:



|  |  |  |  |
| --- | --- | --- | --- |
| کلاس P | زنده مانده |  | تعداد |
| 0 | \* | 0 | 1 |
| 1 | اول | 0 | 129 |
| 2 | اول | 1 | 193 |
| 3 | دوم | 0 | 160 |
| 4 | دوم | 1 | 119 |
| 5 | سوم | 0 | 573 |
| 6 | سوم | 1 | 138 |

**بحث**

توابع انبوه به ویژه در طول تجزیه و تحلیل داده‌های اکتشافی برای یادگیری اطلاعات در مورد زیرجمعیت‌های[[8]](#footnote-8) مختلف داده‌ها و رابطه بین متغیرها مفید هستند. با گروه‌بندی داده‌ها و اعمال آمار انبوه، می‌توانید الگوهایی را در داده‌ها مشاهده کنید که ممکن است در طی فرآیند یادگیری ماشین یا مهندسی ویژگی مفید باشند. در حالی که نمودارهای بصری نیز مفید هستند، اغلب مفید است که چنین آمار توصیفی خاصی نیز به عنوان مرجع برای درک بهتر داده‌ها وجود داشته باشد.

**همچنین ببینید:**

* [مستندات agg در کتابخانه‌ی Pandas](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.agg.html)

**6.8 انجام شناسایی موجودیت با نام**

**مسئله**

می‌خواهید شناسایی موجودیت نام‌گذاری شده را در متن (مانند «شخص»، «دولت» و غیره) انجام دهید.

**راه‌حل**

برای استخراج موجودیت‌ها از متن، از خط لوله[[9]](#footnote-9) و مدل‌های پیش‌فرض شناسایی موجودیت با نام spaCy استفاده کنید:



**بحث**

شناسایی موجودیت نامی، فرآیند شناسایی موجودیت‌های خاص از متن است. ابزارهایی مانند spaCy ارائه دهنده‌ی روش‌هایی مانند خطوط لوله از پیش پیکربندی شده و حتی مدل‌های یادگیری ماشینی از پیش آموزش دیده یا تنظیم شده هستند که به راحتی می‌توانند این اشیاء[[10]](#footnote-10) را شناسایی کنند. در این مورد، ما از spaCy برای شناسایی یک شخص ("Elon Musk")، سازمان ("Twitter") و ارزش پولی ("21B") از متن خام استفاده می‌کنیم. با استفاده از این اطلاعات، می‌توانیم اطلاعات ساختاریافته را از داده‌های متنی بدون ساختار استخراج کنیم. سپس این اطلاعات را می‌توان در مدل‌های یادگیری ماشین پایین‌رونده[[11]](#footnote-11) یا تجزیه و تحلیل داده‌ها استفاده کرد.

**6.11 استفاده از بردارهای متن برای محاسبه شباهت متن در یک عبارت جستجو**

**مسئله**

شما می‌خواهید از بردارهای tf-idf برای پیاده سازی یک تابع جستجوی متن در پایتون استفاده کنید.

**راه‌حل**

شباهت کسینوس بین بردارهای tf-idf را با استفاده از scikit-learn محاسبه کنید:



**بحث**

بردارهای متنی برای موارد استفاده از NLP مانند موتورهای جستجو بسیار مفید هستند. پس از محاسبه بردارهای tf-idf برای مجموعه‌ای از جملات یا اسناد، می‌توانیم از همان شیء tfidf برای بردار کردن مجموعه‌های متن آینده استفاده کنیم. سپس، می‌توانیم شباهت کسینوس را بین بردار ورودی و ماتریس بردارهای دیگر محاسبه کرده و بر اساس مرتبط‌ترین اسناد مرتب کنیم.

شباهت کسینوس در محدوده [0، 1.0] است که 0 کمترین شباهت و 1 بیشترین شباهت را دارد. از آنجایی که ما از بردارهای tf-idf برای محاسبه شباهت بین بردارها استفاده می‌کنیم، فراوانی وقوع یک کلمه نیز در نظر گرفته می‌شود. با این حال، با یک مجموعه کوچک (مجموعه‌ای از اسناد) حتی کلمات "متداول" ممکن است اغلب ظاهر نشوند. در این مثال، "سوئد بهترین است" مرتبط ترین متن با عبارت جستجوی ما "برزیل بهترین است" است. از آنجایی که عبارت کوئری به برزیل اشاره می‌کند، ممکن است انتظار داشته باشیم "من عاشق برزیل هستم. برزیل!» مرتبط ترین مورد باشد. با این حال، "سوئد بهترین است" به دلیل کلمات "است" و "بهترین" شبیه ترین است. با افزایش تعداد اسنادی که به مجموعه خود اضافه می‌کنیم، کلماتی که اهمیت کمتری دارند، وزن کمتری خواهند داشت و تأثیر کمتری بر محاسبات شباهت کسینوس ما خواهند داشت.

**همچنین ببینید:**

* [**شباهت کسینوس، GeeksForGeeks**](https://www.geeksforgeeks.org/cosine-similarity/)

**6.12 استفاده از یک طبقه بندی تحلیل احساسات[[12]](#footnote-12)**

**مسئله**

شما می‌خواهید احساس برخی از متون را برای استفاده به عنوان یک ویژگی یا در تجزیه و تحلیل داده‌های پایین‌رونده[[13]](#footnote-13) طبقه بندی کنید.

**راه‌حل**

از طبقه‌بندی‌کننده احساسات کتابخانه transformers استفاده کنید.



**بحث**

کتابخانه Transformers یک کتابخانه‌ی بسیار محبوب برای وظایف NLP است و شامل تعدادی API با استفاده آسان برای مدل‌های آموزشی یا استفاده از مدل‌های از پیش آموزش دیده است. ما در مورد NLP و این کتابخانه در فصل 22 بیشتر صحبت خواهیم کرد، اما این مثال به عنوان مقدمه‌ای در سطح بالا برای قدرت استفاده از طبقه بندی کننده‌های از پیش آموزش دیده در خطوط لوله یادگیری ماشین شما برای تولید ویژگی ها، طبقه بندی متن یا تجزیه و تحلیل داده‌های بدون ساختار آورده شده است.

**همچنین ببینید:**

* [تور سریع Hugging Face Transformers](https://huggingface.co/docs/transformers/quicktour)

**8.15 استفاده از جاسازی‌های[[14]](#footnote-14) از پیش آموزش دیده به عنوان ویژگی‌ها**

**مسئله**

می‌خواهید جاسازی‌های پیش‌آموزش‌شده را از یک مدل موجود در PyTorch بارگیری کنید و از آنها به عنوان ورودی یکی از مدل‌های خود استفاده کنید.

**راه‌حل**

از torchvision.models برای انتخاب یک مدل و سپس بازیابی یک جاسازی از آن، برای یک تصویر معین، استفاده کنید:



**بحث**

در فضای ML، یادگیری انتقال اغلب به عنوان گرفتن اطلاعات آموخته شده از یک کار و استفاده از آن به عنوان ورودی برای کار دیگر تعریف می‌شود. به‌جای شروع از صفر، می‌توانیم از نمایش‌هایی[[15]](#footnote-15) استفاده کنیم که قبلاً از مدل‌های تصویری بزرگ از پیش آموزش دیده (مانند ResNet) آموخته‌ایم تا در مدل‌های یادگیری ماشین خودمان شروع به کار کنیم. به‌طور شهودی‌تر، می‌توانید درک کنید که چگونه می‌توانیم از وزن‌های یک مدل آموزش دیده برای تشخیص گربه‌ها به عنوان شروع خوبی برای مدلی که می‌خواهیم برای تشخیص سگ‌ها آموزش دهیم، استفاده کنیم. با به اشتراک گذاشتن اطلاعات از یک مدل به مدل دیگر، می‌توانیم اطلاعاتی را که از سایر مجموعه داده‌ها و معماری‌های مدل به دست می‌آیند، بدون هزینه‌های سربار آموزش یک مدل از ابتدا به کار ببریم.

کل کاربرد یادگیری انتقالی در بینایی کامپیوتر خارج از محدوده این کتاب است. با این حال، راه‌های مختلفی وجود دارد که بتوانیم نمایش‌های مبتنی بر جاسازی‌ها را از تصاویر خارج از PyTorch استخراج کنیم. در TensorFlow، یکی دیگر از کتابخانه‌های رایج برای یادگیری عمیق، می‌توانیم از tensorflow\_hub استفاده کنیم:



**همچنین ببینید:**

* [آموزش PyTorch: آموزش انتقال برای بینایی کامپیوتر](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html#sphx-glr-beginner-transfer-learning-tutorial-py)
* [TensorFlow Hub](https://www.tensorflow.org/hub)

**8.15 تشخیص اشیا با OpenCV**

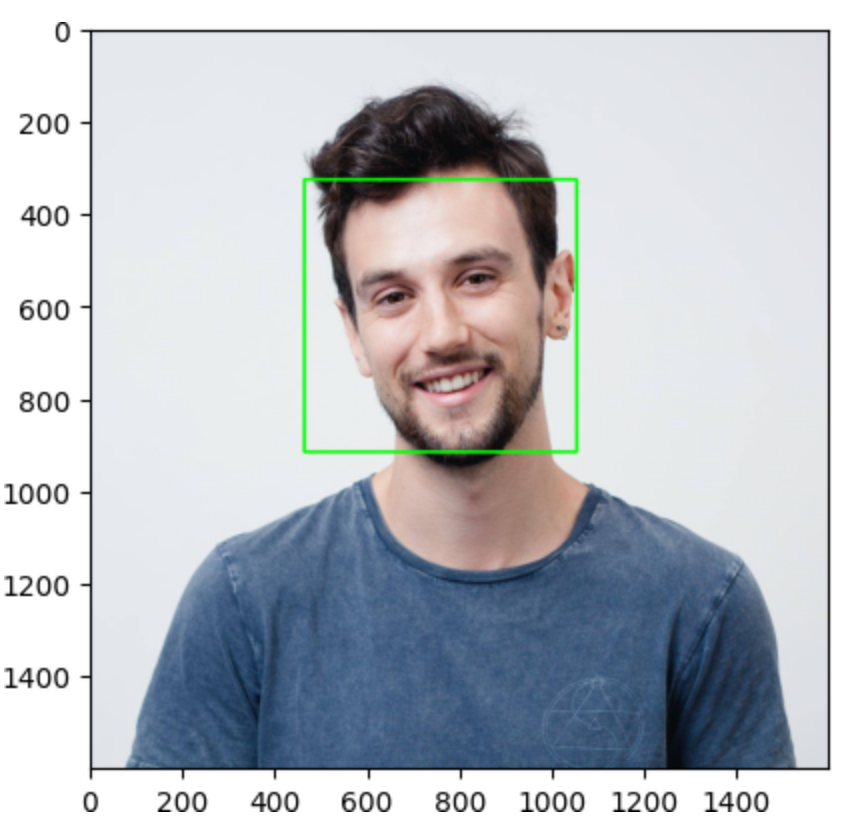
**مسئله**

شما می‌خواهید اشیاء را در تصاویر با استفاده از طبقه بندی کننده‌های آبشاریی از پیش آموزش دیده شده با OpenCV شناسایی کنید.

**راه‌حل**

یکی از طبقه‌بندی‌کننده‌های آبشاری [Haar OpenCV](https://github.com/opencv/opencv/tree/4.x/data/haarcascades) را دانلود و اجرا کنید. در این مورد، ما از یک مدل تشخیص چهره از پیش آموزش دیده شده برای تشخیص و ترسیم مستطیل دور یک چهره در یک تصویر استفاده می‌کنیم:



****

**بحث**

طبقه‌بندی‌کننده‌های آبشاری Haar مدل‌های یادگیری ماشینی هستند که برای یادگیری مجموعه‌ای از ویژگی‌های تصویر (به‌ویژه ویژگی‌های Haar) استفاده می‌شوند که می‌توانند برای شناسایی اشیاء در تصاویر استفاده شوند. خود ویژگی‌ها، ویژگی‌های مستطیلی ساده‌ای هستند که با محاسبه تفاوت در مجموع بین مناطق مستطیلی تعیین می‌شوند. پس از آن، یک الگوریتم تقویت گرادیان برای یادگیری مهم ترین ویژگی‌ها و در نهایت ایجاد یک مدل نسبتا قوی با استفاده از طبقه بندی کننده‌های آبشاریی اعمال می‌شود.

در حالی که جزئیات این فرآیند خارج از محدوده این کتاب است، قابل توجه است که این مدل‌های از پیش آموزش دیده شده را می‌توان به راحتی از مکان‌هایی مانند [OpenCV GitHub](https://github.com/opencv/opencv/tree/4.x/data/haarcascades) به عنوان فایل XML دانلود کرد و بدون آموزش مدلی، روی تصاویر اعمال کرد. این در مواردی مفید است که می‌خواهید ویژگی‌های ساده تصویر باینری مانند contain\_face (یا هر شیء دیگری) را به داده‌های خود اضافه کنید.

**همچنین ببینید:**

* [آموزش OpenCV: طبقه بندی آبشاری](https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html)

**8.15 طبقه بندی تصاویر با Pytorch**

**مسئله**

شما می‌خواهید تصاویر را با استفاده از مدل‌های آموزش عمیق از پیش آموزش دیده شده در Pytorch طبقه بندی کنید.

**راه‌حل**

از torchvision.models برای انتخاب یک مدل طبقه بندی تصویر از پیش آموزش دیده شده و تغذیه تصویر از طریق آن استفاده کنید:



**بحث**

بسیاری از مدل‌های آموزش عمیق از پیش آموزش دیده شده، برای طبقه بندی تصاویر به راحتی از طریق PyTorch و TensorFlow در دسترس هستند. در این مثال، ما از ResNet18، یک معماری شبکه عصبی عمیق استفاده کردیم که بر روی مجموعه داده ImageNet با عمق 18 لایه آموزش داده شده است. مدل‌های عمیق‌تر ResNet، مانند ResNet101 و ResNet152 نیز در Pytorch در دسترس هستند - و فراتر از آن، مدل‌های تصویری زیادی برای انتخاب وجود دارند. مدل‌های از پیش آموزش‌دیده شده، بر روی مجموعه داده ImageNet می‌توانند احتمالات پیش‌بینی‌شده را برای همه کلاس‌های تعریف‌شده در متغیر imagenet\_class\_index در قطعه کد قبلی، که ما از GitHub دانلود کرده‌ایم، به صورت خروجی نمایش دهند.

به طور مثال تشخیص چهره در OpenCV (نگاه کنید به دستور العمل 8.16)، که ما می‌توانیم از کلاس‌های تصویر پیش‌بینی‌شده به‌عنوان ویژگی‌های پایین‌رونده[[16]](#footnote-16) برای مدل‌های آینده ML یا تگ‌های فراداده مفیدی که اطلاعات بیشتری را به تصاویر ما اضافه می‌کنند، استفاده کنیم.

**همچنین ببینید:**

* [مستندات PyTorch: مدل‌ها و وزن‌های از پیش آموزش دیده](https://pytorch.org/vision/stable/models.html)

**14.6 ارزیابی جنگل‌های تصادفی[[17]](#footnote-17) با خطاهای خارج از کیسه[[18]](#footnote-18)**

**مسئله**

شما باید یک مدل جنگل تصادفی را بدون استفاده از اعتبارسنجی متقاطع ارزیابی کنید.

**راه‌حل**

امتیاز خارج از کیف مدل را محاسبه کنید:



**بحث**

در جنگل‌های تصادفی، هر درخت تصمیم با استفاده از زیرمجموعه‌ای از مشاهدات راه‌اندازی، آموزش داده می‌شود. این بدان معناست که برای هر درخت یک زیر مجموعه جداگانه از مشاهدات وجود دارد که برای آموزش آن درخت استفاده نمی‌شود. به این مشاهدات خارج از کیسه (OOB) می‌گویند. ما می‌توانیم از مشاهدات OOB به عنوان یک مجموعه تست برای ارزیابی عملکرد جنگل تصادفی خود استفاده کنیم.

برای هر مشاهده، الگوریتم یادگیری، ارزش واقعی مشاهدات را با پیش‌بینی زیرمجموعه‌ای از درختانی که با استفاده از آن مشاهده آموزش ندیده‌اند، مقایسه می‌کند. امتیاز کلی محاسبه می‌شود و یک معیار واحد از عملکرد یک جنگل تصادفی را ارائه می‌دهد. تخمین امتیاز OOB جایگزینی برای اعتبارسنجی متقابل است.

در scikit-learn، می‌توانیم امتیازهای OOB یک جنگل تصادفی را با تنظیم oob\_score=True در شیء جنگل تصادفی (یعنی RandomForestClassifier) محاسبه کنیم. امتیاز را می‌توان با استفاده از oob\_score\_ بازیابی کرد.

**14.12 باران مدل XGBoost**

**مسئله**

شما باید یک مدل درختی با قدرت پیش بینی بالا آموزش دهید.

**راه‌حل**

از کتابخانه xgboost پایتون استفاده کنید:



**بحث**

XGBoost (که مخفف Extreme Gradient Boosting است) یک الگوریتم تقویت گرادیان بسیار محبوب در فضای یادگیری ماشینی است. اگرچه همیشه یک مدل مبتنی بر درخت نیست، اما اغلب برای مجموعه‌ای از درخت‌های تصمیم استفاده می‌شود. به دلیل موفقیت گسترده در وب سایت مسابقه یادگیری ماشین Kaggle، محبوبیت زیادی به دست آورد و از آن زمان به بعد الگوریتمی‌قابل اعتماد برای بهبود عملکرد فراتر از جنگل‌های تصادفی معمولی یا ماشین‌های تقویت شده گرادیان بوده است.

اگرچه XGBoost به دلیل محاسباتی فشرده شناخته شده است، بهینه سازی عملکرد محاسباتی (مانند پشتیبانی از GPU) در چند سال گذشته، تکرار سریع با XGBoost را به طور قابل توجهی آسان کرده است، و زمانی که عملکرد آماری الزامی‌است، به عنوان یک الگوریتم رایج انتخاب می‌شود.

**همچنین ببینید:**

* [مستندات XGBoost](https://xgboost.readthedocs.io/en/stable/)

**14.6 بهبود عملکرد بلادرنگ با LightGBM**

**مسئله**

شما باید یک مدل مبتنی بر درخت تقویت شده با گرادیان را آموزش دهید که از نظر محاسباتی بهینه شده باشد.

**راه‌حل**

از کتابخانه ماشینی تقویت شده با گرادیان lightgbm استفاده کنید:



**بحث**

کتابخانه lightgbm برای ماشین‌های تقویت‌شده‌ی گرادیان استفاده می‌شود و برای زمان آموزش، استنتاج و پشتیبانی GPU بسیار بهینه شده است. به عنوان یک نتیجه از کارایی محاسباتی آن، اغلب در تولید و در تنظیمات در مقیاس بزرگ استفاده می‌شود. اگرچه استفاده از مدل‌های Sikit-Learn معمولا آسان‌تر است، برخی از کتابخانه‌ها، مانند lightgbm، می‌توانند زمانی مفید باشند که داده‌های بزرگ داشته باشیم یا از لحاظ زمان‌های آموزشی دقیق برای مدل، محدود شده‌اید.

**همچنین ببینید:**

* [مستندات LightGBM](https://lightgbm.readthedocs.io/en/latest/)
* [مستندات CatBoost (یکی دیگر از کتابخانه‌های بهینه شده برای GBM)](https://catboost.ai/en/docs/)

**15.5 پیدا کردن نزدیکترین همسایه‌های تقریبی (ANN)**

**مسئله**

می‌خواهید نزدیک‌ترین همسایگان را برای داده‌های بزرگ با تأخیر کم را واکِشی[[19]](#footnote-19) کنید:

**راه‌حل**

از جستجوی تقریبی نزدیکترین همسایگان (ANN) با کتابخانه faiss فیس بوک استفاده کنید:

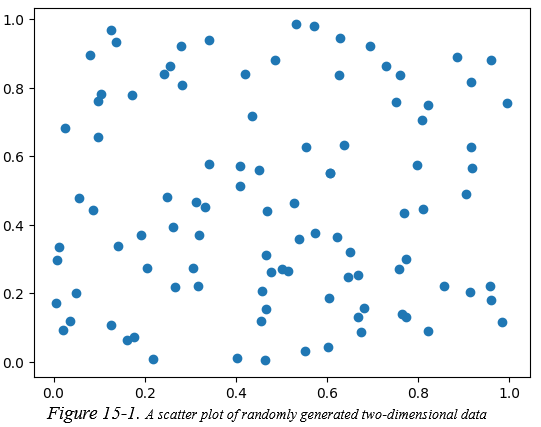


**بحث**

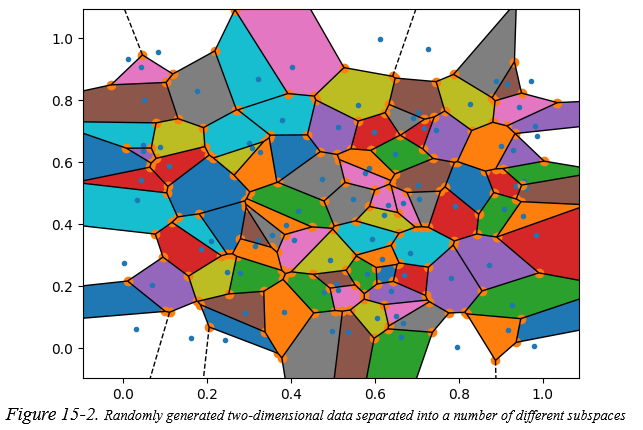
KNN یک رویکرد عالی برای یافتن مشابه ترین مشاهدات در مجموعه‌ای از داده‌های کوچک است. با این حال، با افزایش اندازه داده‌های ما، زمان محاسبه فاصله بین هر مشاهده و سایر نقاط مجموعه داده‌ی ما نیز افزایش می‌یابد. سیستم‌های ML در مقیاس بزرگ مانند موتورهای جستجو یا توصیه، اغلب از نوعی معیار تشابه برداری برای بازیابی مشاهدات مشابه استفاده می‌کنند. اما در مقیاس بزرگ در زمان واقعی، جایی که ما به نتایج کمتر از 100 میلی‌ثانیه نیاز داریم، اجرای KNN غیرممکن می‌شود.

ANN به ما کمک می‌کند تا با قربانی کردن مقداری از کیفیتِ جستجوی دقیقِ نزدیکترین همسایگان به نفع سرعت، بر این مشکل غلبه کنیم. این بدان معناست که اگرچه ترتیب و موارد در 10 همسایه اول یک جستجوی ANN ممکن است با 10 نتیجه اول از یک جستجوی دقیق KNN مطابقت نداشته باشد، ما آن 10 همسایه اول را بسیار سریعتر دریافت می‌کنیم.

در این مثال، ما از یک رویکرد ANN به نام شاخص فایل معکوس (IVF) استفاده می‌کنیم. این رویکرد با استفاده از خوشه بندی برای محدود کردن دامنه فضای جستجو برای جستجوی نزدیکترین همسایگان ما کار می‌کند. IVF از تسسلات[[20]](#footnote-20) Voronoi برای تقسیم فضای جستجوی ما به تعدادی منطقه (یا خوشه) مجزا استفاده می‌کند. و وقتی برای یافتن نزدیک‌ترین همسایگان می‌رویم، از تعداد محدودی از خوشه‌ها بازدید می‌کنیم تا مشاهدات مشابهی را پیدا کنیم، برخلاف مقایسه بین هر نقطه از مجموعه داده‌هایمان با همدیگر.

نحوه ایجاد مجموعه‌های Voronoi از داده‌ها به بهترین وجه با استفاده از داده‌های ساده، قابل مشاهده است. همانطور که در شکل 15-1 نشان داده شده است، یک نمودار پراکنده از داده‌های تصادفی که در دو بعد مشاهده می‌شود، در نظر بگیرید.

با استفاده از Tessellations Voronoi، می‌توانیم تعدادی زیرفضا ایجاد کنیم که هر کدام از آنها فقط شامل یک زیرمجموعه کوچک از کل مشاهداتی است که می‌خواهیم در آن جستجو کنیم، همانطور که در شکل 15-2 نشان داده شده است.



پارامتر nlist در کتابخانه Faiss به ما امکان می‌دهد تعداد خوشه‌هایی را که می‌خواهیم ایجاد کنیم تعریف کنیم. یک پارامتر اضافی، nprobe، می‌تواند در زمان پرس و جو برای تعریف تعداد خوشه‌هایی که می‌خواهیم جستجو کنیم تا نزدیک‌ترین همسایه‌ها را برای یک مشاهده مشخص بازیابی کنیم، استفاده شود. افزایش هر دو nlist و nprobe می‌تواند منجر به همسایگان با کیفیت بالاتر به قیمت تلاش محاسباتی بزرگتر و در نتیجه زمان اجرای طولانی تر برای شاخص‌های IVF شود. کاهش هر یک از این پارامترها، اثر معکوس خواهد داشت و کد شما سریعتر اجرا می‌شود اما همچنین خطر بازگشت نتایج با کیفیت پایین تر را خواهید داشت.

توجه داشته باشید که این مثال دقیقا همان خروجی دستور اول در این فصل را برمی‌گرداند. این به این دلیل است که ما با داده‌های بسیار کوچک کار می‌کنیم و فقط از سه خوشه استفاده می‌کنیم، که باعث می‌شود نتایج ANN ما به طور قابل‌توجهی با نتایج KNN ما متفاوت نباشد.

**همچنین ببینید:**

* [نزدیکترین نمایه‌های همسایه برای جستجوی شباهت (انواع شاخص ANN مختلف)](https://www.pinecone.io/learn/series/faiss/vector-indexes/)

**15.6 ارزیابی تقریبی نزدیکترین همسایگان**

**مسئله**

می‌خواهید ببینید ANN شما چگونه با نزدیک‌ترین همسایگان (KNN) مقایسه می‌شود:

**راه‌حل**

فراخوان[[21]](#footnote-21) @k نزدیکترین همسایه ANN را در مقایسه با KNN محاسبه کنید:



**بحث**

Recall @k به سادگی به عنوان تعداد آیتم‌هایی است که توسط ANN در برخی از k نزدیک‌ترین همسایه‌هایی که در نزدیک‌ترین همسایه‌ها در همان k، تقسیم بر k ظاهر می‌شوند؛ بازگردانده می‌شود. در این مثال، در 10 نزدیکترین همسایه، ما 100% فراخوانی داریم، به این معنی که ANN ما، همان شاخص‌های KNN ما را در k=10 برمی‌گرداند (البته نه لزوماً به همان ترتیب).

یادآوری یا فراخوان[[22]](#footnote-22) یک معیار رایج برای استفاده در هنگام ارزیابی شبکه‌های عصبی مصنوعی در برابر نزدیکترین همسایگان است.

**همچنین ببینید:**

* [یادداشت Google در ANN برای سرویس موتور تطبیق Vertex](https://cloud.google.com/vertex-ai/docs/vector-search/overview#why_does_ann_perform_approximate_matches_instead_of_exact_matches)

**فصل 21- Tensorها با PyTorch**

**20.0 مقدمه**

همانطور که NumPy یک ابزار اساسی برای دستکاری داده‌ها در پشته‌ی یادگیری ماشین[[23]](#footnote-23) است، PyTorch نیز ابزاری اساسی برای کار با Tensorها در پشته‌ی یادگیری عمیق است. قبل از اینکه به یادگیری عمیق بپردازیم، باید خود را با Tensorهای PyTorch آشنا کنیم و بسیاری از عملیات مشابه آنچه با NumPy در فصل 1 انجام شده است ایجاد کنیم.

اگرچه PyTorch تنها یکی از چندین کتابخانه یادگیری عمیق است، اما از محبوبیت قابل توجهی هم در دانشگاه و هم در صنعت برخوردار است. Tensorهای PyTorch بسیار شبیه به آرایه‌های NumPy هستند. با این حال، آنها همچنین به ما اجازه می‌دهند تا عملیات Tensor را روی GPUها (سخت افزار تخصصی برای یادگیری عمیق) انجام دهیم. در این فصل، ما با اصول اولیه Tensorهای PyTorch و بسیاری از عملیات رایج سطح پایین آشنا خواهیم شد.

**20.1 ساخت یک Tensor**

**مسئله**

شما باید یک Tensor ایجاد کنید.

**راه‌حل**

از Pytorch برای ایجاد یک Tensor استفاده کنید:



**بحث**

ساختار اصلی داده در PyTorch یک Tensor است و از بسیاری جهات Tensorها دقیقاً مانند آرایه‌های چند بعدی در NumPy هستند که در فصل 1 استفاده شدند. درست مانند بردارها و آرایه‌ها، این Tensorها می‌توانند به صورت افقی (یعنی ردیف‌ها) یا عمودی (یعنی ستون‌ها) نمایش داده شوند).

**همچنین ببینید:**

* [مستندات PyTorch: Tensorها](https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html)

**20.2 ایجاد یک Tensor از NumPy**

**مسئله**

شما باید Tensorهای PyTorch را از آرایه‌های NumPy ایجاد کنید.

**راه‌حل**

از تابع from\_numpy در PyTorch استفاده کنید:



**بحث**

همانطور که می‌بینیم، PyTorch از نظر نحوی، بسیار شبیه به NumPy است. علاوه بر این، به راحتی به ما اجازه می‌دهد تا آرایه‌های NumPy را به Tensorهای PyTorch تبدیل کنیم که می‌توانیم روی پردازنده‌های گرافیکی و دیگر سخت‌افزارهای شتاب‌دهنده استفاده کنیم. در زمان نگارش، NumPy به طور مکرر در اسناد PyTorch ذکر شده است، و PyTorch خود حتی راهی را ارائه می‌دهد که Tensorهای PyTorch و آرایه‌های NumPy می‌توانند حافظه یکسانی را برای کاهش سربار به اشتراک بگذارند.

**همچنین ببینید:**

* [مستندات PyTorch: پل با NumPy](https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#bridge-to-np-label)

**20.3 ایجاد یک Tensor پراکنده**

**مسئله**

با توجه به داده‌هایی با مقادیر غیر صفر بسیار کم، می‌خواهید آن‌ها را به‌طور کارآمد با یک Tensor نشان دهید.

**راه‌حل**

از تابع PyTorch to\_sparse استفاده کنید:



**بحث**

Tensorهای پراکنده، روش‌های کارآمد حافظه برای نمایش داده‌هایی هستند که عمدتاً از صفرها تشکیل شده‌اند. در فصل 1 ما از scipy برای ایجاد یک ماتریس ردیف پراکنده فشرده (CSR) استفاده کردیم که دیگر یک آرایه NumPy نیست.

کلاس torch.Tensor به ما این امکان را می‌دهد که ماتریس‌های منظم و پراکنده را با استفاده از یک شی ایجاد کنیم. اگر انواع دو Tensorی را که ایجاد کرده‌ایم بررسی کنیم، می‌بینیم که در واقع هر دو از یک کلاس هستند:



**همچنین ببینید:**

* [مستندات PyTorch: Sparce Tensor](https://pytorch.org/docs/stable/sparse.html)

**20.4 انتخاب عناصر در یک Tensor**

**مسئله**

ما باید عناصر خاصی از یک Tensor را انتخاب کنیم.

**راه‌حل**

برای برگرداندن عناصر از نمایه سازی[[24]](#footnote-24) و برش[[25]](#footnote-25) NumPy استفاده کنید:



**بحث**

مانند آرایه‌های NumPy و بیشتر مسائل دیگر در پایتون، Tensorهای PyTorch دارای شاخص صفر هستند. هر دو ویژگی نمایه سازی و برش نیز در Tensorها پشتیبانی می‌شوند. یک تفاوت کلیدی در Tensorها این است که نمایه سازی یک Tensor در PyTorch برای برگرداندن یک عنصر واحد، بر خلاف مقدار خود شی (که به شکل یک عدد صحیح یا اعشاری خواهد بود) همچنان یک Tensor برمی‌گرداند. نحوه‌ی برش نیز همانند NumPy است و اشیائی از نوع Tensor را در PyTorch برمی‌گرداند:



یک تفاوت کلیدی این است که Tensorهای PyTorch هنوز از مراحل منفی هنگام برش پشتیبانی نمی‌کنند. بنابراین، تلاش برای معکوس کردن یک Tensor با استفاده از برش، یک خطا ایجاد می‌کند:



در عوض، اگر بخواهیم یک Tensor را معکوس کنیم، می‌توانیم از روش تلنگر[[26]](#footnote-26) استفاده کنیم:



**همچنین ببینید:**

* [مستندات PyTorch: عملیات روی تنسورها](https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html#operations-on-tensors)

--

**20.5 توصیف یک Tensor**

**مسئله**

شما می‌خواهید شکل، نوع داده و قالب یک Tensor را همراه با سخت افزاری که استفاده می‌کند، توصیف کنید.

**راه‌حل**

شکل، نوع داده طرح و ویژگی‌های دستگاه Tensor را بررسی کنید:



**بحث**

Tensorهای PyTorch تعدادی ویژگی مفید برای جمع آوری اطلاعات در مورد یک Tensor مشخص ارائه می‌دهند، از جمله:

**شکل - Shape**

ابعاد Tensor را برمی‌گرداند

**نوع داده‌ای - Dtype**

نوع داده اشیاء درون Tensor را برمی‌گرداند

**چیدمان - Layout**

طرح‌بندی حافظه را برمی‌گرداند (متداول‌ترین حالت استفاده از راه‌حل است برای Tensorهای متراکم)

**دستگاه - Device**

سخت افزاری را که Tensor روی آن ذخیره می‌شود برمی‌گرداند (CPU/GPU)

باز هم، تمایز اصلی بین Tensorها و آرایه‌ها ویژگی‌هایی مانند دستگاه است، زیرا Tensorها گزینه‌های شتاب‌دهنده سخت‌افزاری مانند GPU را در اختیار ما قرار می‌دهند.

**20.6 اعمال عملیات بر روی عناصر**

**مسئله**

شما می‌خواهید یک عملیات را برای تمام عناصر یک Tensor اعمال کنید.

**راه‌حل**

از قابلیت پخش[[27]](#footnote-27) با PyTorch استفاده کنید:



**بحث**

عملیات پایه در PyTorch از مزیت پخش(broadcasting) برای موازی کردن عملیات‌ها با استفاده از سخت افزارهای شتاب یافته مانند GPU استفاده می‌کند. این عملیات برای عملگرهای ریاضی پشتیبانی شده در پایتون (+، -، ×، /) و سایر توابع ذاتی PyTorch صادق است. برخلاف NumPy، PyTorch یک روش برداری برای اعمال یک تابع بر روی تمام عناصر یک Tensor ندارد. با این حال، PyTorch مجهز به تمام ابزارهای ریاضی لازم برای توزیع و تسریع عملیات معمول مورد نیاز برای جریان‌های کاری یادگیری عمیق است.

**همچنین ببینید:**

* [مستندات PyTorch: معناشناسی Broadcasting](https://pytorch.org/docs/stable/notes/broadcasting.html)
* [Vectorization و Broadcasting با PyTorch](https://blog.paperspace.com/pytorch-vectorization-and-broadcasting/)

**20.7 یافتن مقادیر حداکثر و حداقل**

**مسئله**

شما باید حداکثر یا حداقل مقدار را در یک Tensor پیدا کنید.

**راه‌حل**

از متدهای max و min در PyTorch استفاده کنید:



**بحث**

توابع min و max در یک Tensor به ما کمک می‌کند بزرگترین یا کوچکترین مقادیر را در آن Tensor پیدا کنیم. این توابع در Tensorهای چند بعدی نیز یکسان عمل می‌کنند:



**20.8 تغییر شکل Tensorها**

**مسئله**

شما می‌خواهید شکل (تعداد سطرها و ستون ها) یک Tensor را بدون تغییر مقادیر عناصر تغییر دهید.

**راه‌حل**

از متد reshape در PyTorch استفاده کنید:



**بحث**

دستکاری شکل یک Tensor می‌تواند در زمینه یادگیری عمیق رایج باشد، زیرا نورون‌ها در یک شبکه عصبی اغلب به Tensorهایی با شکل بسیار خاص نیاز دارند. از آنجایی که شکل مورد نیاز یک Tensor می‌تواند بین نورون‌ها در یک شبکه عصبی مشخص تغییر کند، خوب است که درک سطح پایینی از ورودی‌ها و خروجی‌های خود در یادگیری عمیق داشته باشیم.

**20.9 انتقال یک Tensor**

**مسئله**

شما باید یک Tensor را جابجا کنید.

**راه‌حل**

از تابع mT استفاده کنید:



**بحث**

انتقال با PyTorch کمی‌با NumPy متفاوت است. روش T مورد استفاده برای آرایه‌های NumPy در PyTorch فقط با Tensorهای دو بعدی پشتیبانی می‌شود و در زمان نوشتن برای Tensorهای اَشکال دیگر منسوخ شده است. روش mT که برای جابجایی دسته‌های Tensor استفاده می‌شود به باقی روش‌ها ترجیح داده می‌شود، زیرا مقیاس آن به بیش از دو بعد نیز می‌رسد.

یک راه اضافی برای جابجایی Tensorهای PyTorch با هر شکلی، استفاده از روش permute است:



این روش برای Tensorهای یک بعدی (که مقدار Tensor جابجا شده همان Tensor اصلی است) نیز کار می‌کند.

**20.10 صاف و مسطح کردن یک Tensor**

**مسئله**

شما باید یک Tensor را به یک بعد تبدیل کنید.

**راه‌حل**

از تابع flatten استفاده کنید:



**بحث**

مسطح کردن یک Tensor یک تکنیک مفید برای کاهش بعد یک Tensor چند بعدی به یک بعد است.

**20.11 محاسبه نتایج نقطه ای**

**مسئله**

شما باید حاصل ضرب نقطه‌ای دو Tensor را محاسبه کنید.

**راه‌حل**

از متد dot استفاده کنید:



**بحث**

محاسبه حاصل ضرب نقطه‌ای دو Tensor یک عملیات معمولی است که در فضای یادگیری عمیق و همچنین فضای بازیابی اطلاعات، مفید است. شاید قبلاً در کتاب به یاد داشته باشید که از حاصل ضرب نقطه‌ای دو بردار برای انجام یک جستجوی مبتنی بر شباهت کسینوس استفاده کردیم. انجام این کار در PyTorch در GPU (به جای NumPy یا scikit-learn در CPU) می‌تواند مزایای عملکردی چشمگیری در مشکلات بازیابی اطلاعات داشته باشد.

**همچنین ببینید:**

* [برداری کردن و پخش با PyTorch](https://blog.paperspace.com/pytorch-vectorization-and-broadcasting/)

**20.12 ضرب Tensorها**

**مسئله**

شما باید دو Tensor را ضرب کنید.

**راه‌حل**

از عملگرهای اساسی حساب پایتون استفاده کنید:



**بحث**

PyTorch از عملگرهای اساسی حسابی مانند ×، +، - و / پشتیبانی می‌کند. اگرچه ضرب Tensorها احتمالاً یکی از رایج‌ترین عملیات مورد استفاده در یادگیری عمیق است، اما دانستن اینکه Tensorها می‌توانند جمع، تفریق و تقسیم شوند نیز مفید است.

جمع یک Tensor با دیگری:



تفریق یک Tensor از دیگری:



تقسیم یک Tensor بر دیگری:



**21.1 استفاده از Autograd با PyTorch**

**مسئله**

می‌خواهید از ویژگی‌های خودکار PyTorch برای محاسبه و ذخیره گرادیان‌ها پس از انتشار به جلو و انتشار به عقب استفاده کنید.

**راه‌حل**

Tensorها را با گزینه requires\_grad که روی True تنظیم شده است ایجاد کنید:



**بحث**

Autograd یکی از ویژگی‌های اصلی PyTorch و عامل مهمی‌در محبوبیت آن به عنوان یک کتابخانه یادگیری عمیق است. توانایی محاسبه، ذخیره و تجسم آسان گرادیان ها، PyTorch را برای محققان و علاقه مندان به ساخت شبکه‌های عصبی از ابتدا، بسیار شهودی می‌کند.

PyTorch از یک گراف غیر چرخه‌ای جهت دار (DAG) برای نگه داشتن رکوردی از تمام داده‌ها و عملیات محاسباتی انجام شده بر روی آن داده استفاده می‌کند. این امر فوق‌العاده مفیدی است، اما به این معنی است که ما باید مراقب باشیم که چه عملیاتی را روی داده‌های PyTorch خود اعمال می‌کنیم که به گرادیان نیاز دارند. هنگام کار با autograd، نمی‌توانیم به راحتی Tensorهای خود را بدون «شکستن نمودار»[[28]](#footnote-28) به آرایه‌های NumPy تبدیل کنیم، عبارتی که برای توصیف عملیات‌هایی که از autograd پشتیبانی نمی‌کنند استفاده می‌شود:



برای تبدیل این Tensor به یک آرایه NumPy، باید متد ()detach را روی آن فراخوانی کنیم، که نمودار را شکسته و در نتیجه توانایی ما برای محاسبه خودکار گرادیان‌ها را می‌شکند. در حالی که این کار قطعا می‌تواند مفید باشد، ارزش دانستن این را دارد که جدا کردن Tensor، از محاسبه خودکار گرادیان PyTorch جلوگیری می‌کند.

1. - Parquet [↑](#footnote-ref-1)
2. - Apache Parquet documentation [↑](#footnote-ref-2)
3. - Apache Avro documentation [↑](#footnote-ref-3)
4. - Querying a SQLite Database [↑](#footnote-ref-4)
5. - query [↑](#footnote-ref-5)
6. - S3 Bucket [↑](#footnote-ref-6)
7. - object [↑](#footnote-ref-7)
8. - subpopulations [↑](#footnote-ref-8)
9. - pipeline [↑](#footnote-ref-9)
10. - entities [↑](#footnote-ref-10)
11. - downstream [↑](#footnote-ref-11)
12. - Sentiment Analysis [↑](#footnote-ref-12)
13. - downstream [↑](#footnote-ref-13)
14. - Embeddings [↑](#footnote-ref-14)
15. - representations [↑](#footnote-ref-15)
16. - downstream [↑](#footnote-ref-16)
17. - Random Forests [↑](#footnote-ref-17)
18. - Out-of-Bag Erros [↑](#footnote-ref-18)
19. - fetch [↑](#footnote-ref-19)
20. - tessellations [↑](#footnote-ref-20)
21. - recall [↑](#footnote-ref-21)
22. - recall [↑](#footnote-ref-22)
23. - mechine learning stack [↑](#footnote-ref-23)
24. - indexing [↑](#footnote-ref-24)
25. - slicing [↑](#footnote-ref-25)
26. - flip [↑](#footnote-ref-26)
27. - broadcasting [↑](#footnote-ref-27)
28. - breaking the graph [↑](#footnote-ref-28)