**فصل 20- تنسورها با پایتورچ**

**20.0 مقدمه**

همانطور که NumPy یک ابزار اساسی برای دستکاری داده‌ها در پشته‌ی یادگیری ماشین[[1]](#footnote-1) 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[[2]](#footnote-2) و برش[[3]](#footnote-3) 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[[4]](#footnote-4) 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[[5]](#footnote-5) 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:



1. - mechine learning stack [↑](#footnote-ref-1)
2. - indexing [↑](#footnote-ref-2)
3. - slicing [↑](#footnote-ref-3)
4. - flip [↑](#footnote-ref-4)
5. - broadcasting [↑](#footnote-ref-5)