۱.۴ 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 ۰، ۱ یا مقادیر دلخواه شما است:

```
# Load library
import numpy as np


# Generate a vector of shape (1,5) containing all zeros
vector = np.zeros(shape=5)


# View the matrix
print(vector)
array([0., 0., 0., 0., 0.])


# Generate a matrix of shape (3,3) containing all ones
matrix = np.full(shape=(3,3), fill_value=1)


# View the vector
print(matrix)
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

بحث

تولید آرایه‌هایی که از قبل با داده‌ها پر شده‌اند، برای چندین هدف مفید است، مانند عملکرد بهتر کد یا اسـتفاده از داده‌هـای مصنوعی برای آزمایش الگوریتم‌ها. در بسیاری از زبان‌های برنامه نویسی، پیش تخصیص‌یک آرایه از مقادیر پیش فرض (مانند ۰s)یک روش معمول در نظر گرفته می‌شود.

۲.۶ بارگذاری‌یک فایل پارکت[1]

مسئله

شما بایدیک فایل پارکت را بارگذاری کنید.

راه‌حل

تابع read_parquet از کتابخانه‌ی pandas به ما این امکان می‌دهد که فایل‌های Parquet را بخوانیم:

```
# Load library
import pandas as pd


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.parquet'


# Load data
dataframe = pd.read_parquet(url)


# View the first two rows
dataframe.head(2)
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

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]

---

[1] - Parquet
[2] - Apache Parquet documentation

۲.۷ بارگیری یک فایل Avro

مسئله

شما بایدیک فایل Avro را در دیتافرِیم pandas بارگذاری کنید.

راه‌حل

از تابع read_avro در کتابخانه‌ی pandavro استفاده کنید:

```
# Load library
import requests
import pandavro as pdx


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.avro'


# Download file
r = requests.get(url)
open('data.avro', 'wb').write(r.content)


# Load data
dataframe = pdx.read_avro('data.avro')


# View the first two rows
dataframe.head(2)
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

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[۳]

[3] - Apache Avro documentation

۲.۸ جستجو در پایگاه داده SQLite[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] SQL در پایگاه داده ایجاد کرده و آن را بارگذاری کنیم:

```
# Load libraries
import pandas as pd
from sqlalchemy import create_engine


# Create a connection to the database
database_connection = create_engine('sqlite:///sample.db')


# Load data
dataframe = pd.read_sql_query('SELECT * FROM data', database_connection)


# View first two rows
dataframe.head(2
```

	نام	نام خانوادگی	سن	امتیاز پیش‌آزمون
۰	Json	Miller	۴۲	۴
۱	Molly	Jaboson	۵۲	۲۴

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.

---

[4] - Querying a SQLite Database
[5] - query

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

- SQLite
- آموزش SQL در وبسایت Schools۳W

۲.۹ 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:

```
# Load library
import requests
import pandavro as pdx


# Create URL
url = 'https://machine-learning-python-cookbook.s3.amazonaws.com/data.avro'


# Download file
r = requests.get(url)
open('data.avro', 'wb').write(r.content)


# Load data
dataframe = pdx.read_avro('data.avro')


# View the first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

بحث

از بین تمام دستور العمل‌های ارائه شده در این فصل، احتمالاً این دستور العملی است کـه بیشـتر در دنیـای واقعـی اسـتفاده خواهیم کرد. در حالی که اتصال و خواندن ازیک پایگاه داده نمونه‌ی sqlite مفید است، احتمالاً نماینده‌ی جداولی نیست کـه

باید دریک محیط سازمانی به آنها متصل شوید. اکثر نمونه‌های SQL که به آن‌ها متصل می‌شوید، از شما می‌خواهند کـه بـه میزبان و پورت‌یک دستگاه از راه دور متصل شوید، ویک نام کاربری و رمز عبور برای احراز هویت مشخص کنید. این مثـال از شما می‌خواهد که‌یک نمونه SQL در حال اجرا را به صورت محلی راه‌اندازی کنید کـه ازیـک سـرور راه دور در لوکال‌هاسـت تقلید می‌کند تا بتوانید حسی از گردش کار دریافت کنید.

همچنین ببینید:

- مستندات PyMySQL
- مستندات Read SQL در کتابخانه‌ی Pandas

۲.۱۰ 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 کـه Google Sheet را بـه‌عنوان CSV صادر می‌کند:

```
# Import libraries
import pandas as pd


# Google Sheet URL that downloads the sheet as a CSV
url = "https://docs.google.com/spreadsheets/d/"\
"1ehC-9otcAuitqnmWksqt1mOrTRCL38dv0K9UjhwzTOA/export?format=csv"


# Read the CSV into a dataframe
dataframe = pd.read_csv(url)


# View the first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

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

۲.۱۱ بارگیری داده‌ها از سطل[6] ۳S

مسئله

شما بایدیک فایل CSV را ازیک سطل ۳S که به آن دسترسی دارید بخوانید.

راه‌حل

گزینه‌های ذخیره سازی را به Pandas اضافه کنید تا به شیء[7] ۳S دسترسی داشته باشد:

```
# Import libraries
import pandas as pd


# S3 path to CSV
s3_uri = "s3://machine-learning-python-cookbook/data.csv"


# Set AWS credentials (replace with your own)
ACCESS_KEY_ID = "xxxxxxxxxxxxx"
SECRET_ACCESS_KEY = "xxxxxxxxxxxxxxxx"


# Read the CSV into a dataframe
dataframe = pd.read_csv(s3_uri,storage_options={
                "key": ACCESS_KEY_ID,
                "secret": SECRET_ACCESS_KEY,
        }
)
#View first two rows
dataframe.head(2)
```

	عدد	تاریخ و زمان	دسته بندی
۰	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۰
۱	۵	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۰

[6] - S۳ Bucket
[7] - object

بحث

اکنـون بسـیاری از شـرکت‌ها داده‌هـا را در فروشـگاه‌های blob ارائه‌دهنـده ابـری ماننـدAmazon S۳ یـا Google Cloud Storage(GCS) نگهداری می‌کنند. معمولاً متخصصان یادگیری ماشینی برای بازیابی داده‌ها به این منابع متصـل می‌شـوند. اگرچه S۳s) URI ۳(//machine-learning-python-cookbook/data.csv: عمومی‌است، اما همچنان از شما می‌خواهـد که اعتبار دسترسی AWS خود را برای دسترسی به آن ارائـه دهیـد. شـایان ذکـر اسـت کـه اشـیاء عمـومی‌همچنین دارای URLهای HTTP هستند که می‌توانند فایل‌ها را از آن دانلود کنند، مانند این مورد برای فایل CSV.

همچنین ببینید:

- ۳Amazon S
- اعتبارنامه امنیتی AWS

۲.۱۲ 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:

```
# Import libraries
import requests

# URL to download the txt file from
txt_url = "https://machine-learning-python-cookbook.s3.amazonaws.com/text.txt"

# Get the txt file
r = requests.get(txt_url)

# Write it to text.txt locally
with open('text.txt', 'wb') as f:
f.write(r.content)


# Read in the file
with open('text.txt', 'r') as f:
text = f.read()

# Print the content
print(text)
Hello there!
```

	دسته بندی	تاریخ و زمان	عدد
۰	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۰	۵
۱	۰	۲۰۱۵-۰۱-۰۱ ۰۰:۰۰:۰۱	۵

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
- مدیران context در پایتون

۳.۲ 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:

```
# Load library
import pandas as pd


# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'


# Load data
dataframe = pd.read_csv(url)


# Show two rows
dataframe.head(2)
```

	نام	کلاس P	سن	جنسیت

۰	Allen, Miss Elisabeth Walton	اول	۲۹.۰	زن
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن

همچنین می‌توانیم به تعداد سطرها و ستون‌ها نگاهی بیندازیم:

```
# Show dimensions
dataframe.shape
```

```
(1313, 6)
```

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

```
# Show statistics
dataframe.describe()
```

	سن	زنده مانده	کد جنسیت
تعداد	۷۵۶.۰۰۰۰۰۰	۱۳۱۳.۰۰۰۰۰۰	۱۳۱۳.۰۰۰۰۰۰
میانگین	۳۰.۳۹۷۹۸۹	۰.۳۴۲۷۲۷	۰.۳۵۱۸۶۶
std	۱۴.۲۵۹۰۴۹	۰.۴۷۴۸۰۲	۰.۴۷۷۷۳۴
مینیمم	۰.۱۷۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۲۵%	۲۱.۰۰۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۵۰%	۲۸.۰۰۰۰۰۰	۰.۰۰۰۰۰۰	۰.۰۰۰۰۰۰
۷۵%	۳۹.۰۰۰۰۰۰	۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰
ماکزیمم	۷۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰	۱.۰۰۰۰۰۰

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

```
# Show info
dataframe.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1313 entries, 0 to 1312
Data columns (total 6 columns):
Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 1313 non-null object
1 PClass 1313 non-null object
2 Age 756 non-null float64
3 Sex 1313 non-null object
4 Survived 1313 non-null int64

5 SexCode 1313 non-null int64
dtypes: float64(1), int64(2), object(3)
memory usage: 61.7+ KB
```

بحث

پس از بارگیری برخی از داده ها، ایده خوبی است که بفهمیم ساختار آن چگونه است و چه نوع اطلاعـاتی در آن وجـود 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 خلاصـه داده‌هـا باید به کشیدن نمونه تکیه کنیم.

در راه حل خود، ما از مجموعه داده اسباب بازی مسافران تایتانیک استفاده می‌کنیم. با استفاده از head، می‌تـوانیم بـه چنـد ردیف اول (پنج به طور پیش فرض) داده‌ها نگاه کنیم. از طرف دیگر، می‌توانیم از tail برای مشاهده چند ردیف آخر اسـتفاده کنیم. با شکل می‌توانیم ببینیم که DataFrame ما شامل چند ردیف و سـتون اسـت. بـا توصـیف می‌تـوانیم برخـی از آمـار توصیفی اولیه برای هر ستون عددی را ببینیم. و در نهایت، اطلاعـات تعـدادی از نقـاط داده مفیـد را در مـورد 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 ۱ و ۰ هستند. با این حال، در این مورد مقـادیر عـددی نشـان دهنده دسته‌ها هستند. به عنوان مثال، اگر Survived برابر با ۱ 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 جنسـیتی (نشـانگر جنسیت مسافر).

۳.۳ 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:

```
# Load library
import pandas as pd


# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'


# Load data
dataframe = pd.read_csv(url)


# Select first row
dataframe.iloc[0]
Name        Allen, Miss Elisabeth Walton
PClass      1st
Age         29
Sex         female
Survived    1
SexCode     1
Name:       0, dtype: object
```

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

```
# Select three rows
dataframe.iloc[1:4]
```

	نام	کلاس P	سن	جنسیت
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن
۲	Allison, Mr Hudson Joshua Creighton	اول	۳۰.۰	مرد
۳	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	اول	۲۵.۰	زن

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

```
# Select four rows
dataframe.iloc[:4]
```

	نام	کلاس P	سن	جنسیت

۰	Allen, Miss Elisabeth Walton	اول	۲۹.۰	زن
۱	Allison, Miss Helen Loraine	اول	۲.۰	زن
۲	Allison, Mr Hudson Joshua Creighton	اول	۳۰.۰	مرد
۳	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	اول	۲۵.۰	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:

```
# Set index
dataframe = dataframe.set_index(dataframe['Name'])


# Show row
dataframe.loc['Allen, Miss Elisabeth Walton']
Name        Allen, Miss Elisabeth Walton
PClass      1st
Age         29
Sex         female
Survived    1
SexCode     1
Name:       Allen, Miss Elisabeth Walton, dtype: object
```

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.

۳.۵ 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:

```
# Load library
import pandas as pd


# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'


# Load data
dataframe = pd.read_csv(url)


# Sort the dataframe by age, show two rows
dataframe.sort_values(by=["Age"]).head(2)
```

	نام	کلاس P	سن	جنسیت
۷۶۳	Dean, Miss Elizabeth Gladys (Millvena)	سوم	۰.۱۷	زن
۷۵۱	Danbom, Master Gilbert Sigvard Emanuel	سوم	۰.۳۳	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.

۳.۱۶ 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:

```
# Load library
import pandas as pd

# Create URL
url =
'https://raw.githubusercontent.com/chrisalbon/sim_data/master/titanic.csv'

# Load data
dataframe = pd.read_csv(url)

# Get the minimum of every column
dataframe = pd.read_csv(url)
Name        Abbing, Mr Anthony
PClass      *
Age         0.17
Sex         female
Survived    0
SexCode     0
dtype:      object
```

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

```
# Mean Age, min and max SexCode
dataframe.agg({"Age":["mean"], "SexCode":["min", "max"]})
```

	سن	جنسیت
میانگین	۳۰.۳۹۷۹۸۹	Nan
مینیمم	Nan	۰.۰
ماکزیمم	Nan	۱.۰

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

```
# Number of people who survived and didn't survive in each class
dataframe.groupby(
        ["PClass","Survived"]).agg({"Survived":["count"]}
  ).reset_index()
```

کلاس P	زنده مانده		تعداد
۰	❋	۰	۱
۱	اول	۰	۱۲۹
۲	اول	۱	۱۹۳
۳	دوم	۰	۱۶۰

۴	دوم	۱	۱۱۹
۵	سوم	۰	۵۷۳
۶	سوم	۱	۱۳۸

بحث

توابع انبوه به ویژه در طول تجزیه و تحلیل داده‌های اکتشافی برای یادگیری اطلاعات در مورد زیرجمعیت‌های[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

۶.۸ 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] و مدل‌های پیش‌فرض شناسایی موجودیت با نام spaCy استفاده کنید:

[8] - subpopulations
[9] - pipeline

```
# Import libraries
import spacy


# Load the spaCy package and use it to parse the text
# make sure you have run "python -m spacy download en"
nlp = spacy.load("en_core_web_sm")
doc = nlp("Elon Musk offered to buy Twitter using $21B of his own money.")


# Print each entity
print(doc.ents)


# For each entity print the text and the entity label
for entity in doc.ents:
   print(entity.text, entity.label_, sep=",")
(Elon Musk, Twitter, 21B)
Elon Musk, PERSON
Twitter, ORG
21B, MONEY
```

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 آموزش‌دیده یا تنظیم شده هستند که به راحتی می‌توانند این اشیاء[10] را شناسایی کنند. در این مورد، ما از spaCy برای شناسایی یک شخص ("Elon Musk")، سازمان ("Twitter") و ارزش پولی ("B۲۱") 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] یا تجزیه و تحلیل داده‌ها استفاده کرد.

۶.۱۱ 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:

[10] - entities
[11] - downstream

```
# Load libraries
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel


# Create searchable text data
text_data = np.array(['I love Brazil. Brazil!',
                      'Sweden is best',

                      'Germany beats both'])


# Create the tf-idf feature matrix
tfidf = TfidfVectorizer()
feature_matrix = tfidf.fit_transform(text_data)


# Create a search query and transform it into a tf-idf vector
text = "Brazil is the best"
vector = tfidf.transform([text])


# Calculate the cosine similarities between the input vector and all other
  vectors
cosine_similarities = linear_kernel(vector, feature_matrix).flatten()


# Get the index of the most relevent items in order
related_doc_indicies = cosine_similarities.argsort()[:-10:-1]


# Print the most similar texts to the search query along with the cosine
  similarity
print([(text_data[i], cosine_similarities[i]) for i in related_doc_indicies])
[
   (
      'Sweden is best', 0.6666666666666666),
      ('I love Brazil. Brazil!', 0.5163977794943222),
      ('Germany beats both', 0.0
      )
]
```

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 [۰، ۱.۰] است که ۰ کمترین شباهت و ۱ 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

۶.۱۲ استفاده از یک طبقه بندی تحلیل احساسات[۱۲]

مسئله

شما می‌خواهید احساس برخی از متون را برای استفاده به عنوان یک ویژگی یا در تجزیه و تحلیل داده‌های پایین‌رونده[۱۳] طبقه بندی کنید.

راه‌حل

از طبقه‌بندی‌کننده احساسات کتابخانه transformers استفاده کنید.

[12] - Sentiment Analysis
[13] - downstream

```
# Import libraries
from transformers import pipeline


# Create an NLP pipeline that runs sentiment analysis
classifier = pipeline("sentiment-analysis")


# Classify some text
# (this may download some data and models the first time you run it)
sentiment_1 = classifier("I hate machine learning! It's the absolute worst.")
sentiment_2 = classifier(
        "Machine learning is the absolute"
        "bees knees I love it so much!"
)



#Print sentiment output
print(sentiment_1, sentiment_2)
[
        {
                'label': 'NEGATIVE',
                'score': 0.9998020529747009
        }
]

[
        {
                'label': 'POSITIVE',
                'score': 0.9990628957748413
        }
]
```

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 آموزش‌دیده است. ما در مورد NLP و این کتابخانه در فصل ۲۲ بیشتر صحبت خواهیم کرد، اما این مثال به عنوان مقدمه‌ای در سطح بالا برای قدرت استفاده از طبقه بندی کننده‌های از پیش آموزش‌دیده 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

۸.۱۵ استفاده از جاسازی‌های[۱۴] از پیش آموزش‌دیده به عنوان ویژگی‌ها

[14] - Embeddings

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:

```
# Load libraries
import cv2
import numpy as np
import torch
from torchvision import transforms
import torchvision.models as models

# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)

# Convert to pytorch data type
convert_tensor = transforms.ToTensor()
pytorch_image = convert_tensor(np.array(image_rgb))

# Load the pretrained model
model = models.resnet18(pretrained=True)

# Select the specific layer of the model we want output from
layer = model._modules.get('avgpool')

# Set model to evaluation mode
model.eval()

# Infer the embedding with the no_grad option
with torch.no_grad():
        embedding = model(pytorch_image.unsqueeze(0))

print(embedding.shape)
torch.Size([1, 1000])
```

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] استفاده کنیم که قبلاً از مدل‌های تصویری بزرگ از پیش آموزش‌دیده (مانند ResNet) آموخته‌ایم تا در مدل‌های یادگیری ماشین خودمان شروع به کار کنیم. به‌طور شهودی‌تر،

[15] - representations

می‌توانید درک کنید که چگونه می‌توانیم از وزن‌های یک مدل آموزش‌دیده 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:

```
# Load libraries
import cv2
import tensorflow as tf
import tensorflow_hub as hub


# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Convert to tensorflow data type
tf_image = tf.image.convert_image_dtype([image_rgb], tf.float32)


# Create the model and get embeddings using the inception V1 model
embedding_model = hub.KerasLayer(
        "https://tfhub.dev/google/imagenet/inception_v1/feature_vector/5"
)
embeddings = embedding_model(tf_image)


# Print the shape of the embedding
print(embeddings.shape)
(1, 1024)
```

همچنین ببینید:

- آموزش PyTorch: آموزش انتقال برای بینایی کامپیوتر
- TensorFlow Hub

۸.۱۵ تشخیص اشیا با OpenCV

مسئله

شما می‌خواهید اشیاء را در تصاویر با استفاده از طبقه بندی کننده‌های آبشاری از پیش آموزش‌دیده شده با OpenCV شناسایی کنید.

راه‌حل

یکی از طبقه‌بندی‌کننده‌های آبشاری Haar OpenCV را دانلود و اجرا کنید. در این مورد، ما از یک مدل تشخیص چهره از پیش آموزش‌دیده شده برای تشخیص و ترسیم مستطیل دور یک چهره در یک تصویر استفاده می‌کنیم:

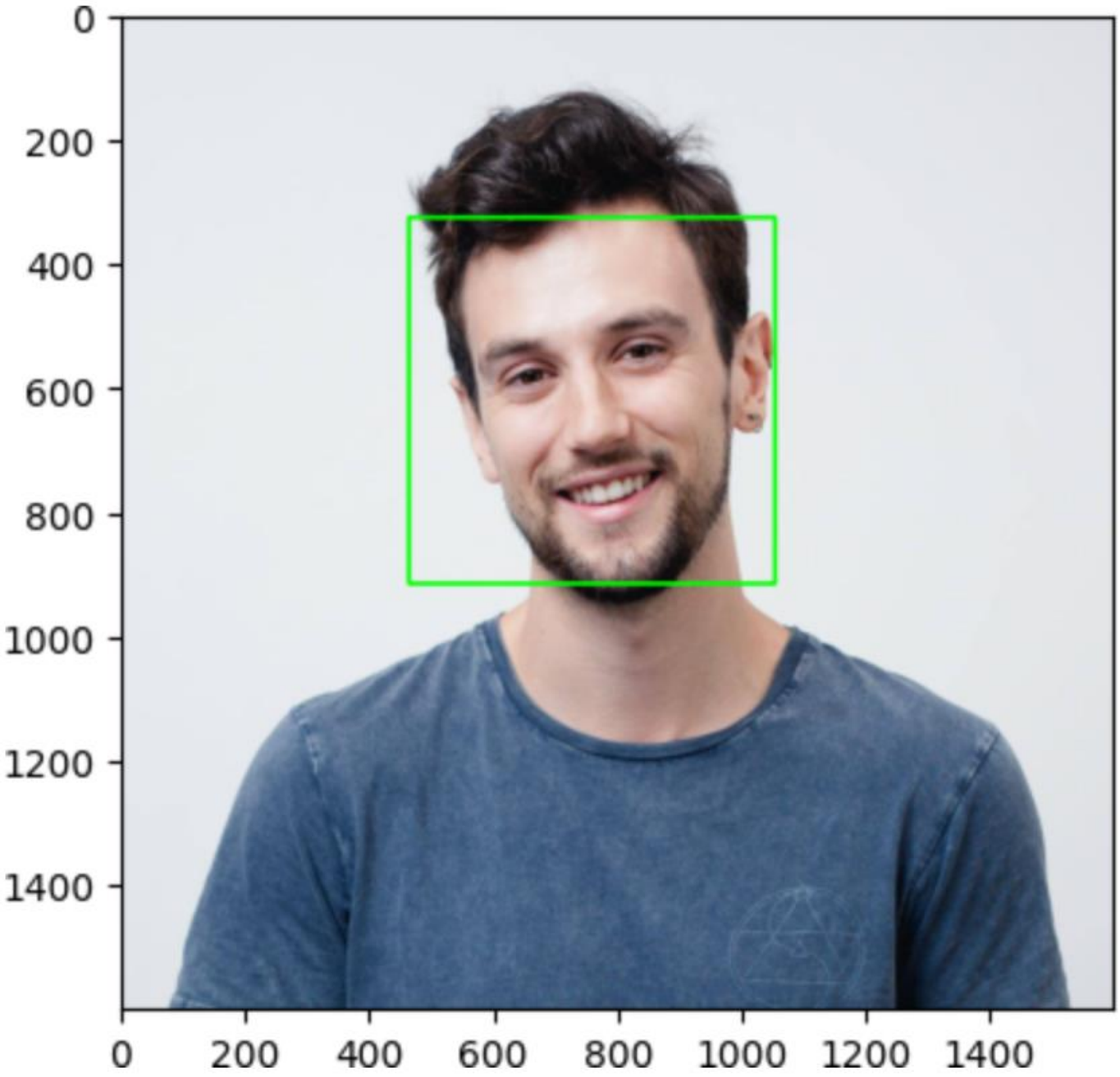
```
# Import libraries
import cv2
from matplotlib import pyplot as plt


# first run:
# mkdir models && cd models
# wget https://tinyurl.com/mrc6jwhp
face_cascade = cv2.CascadeClassifier()
face_cascade.load(
        cv2.samples.findFile(
                "models/haarcascade_frontalface_default.xml"
        )
)

# Load image
image_bgr = cv2.imread("images/kyle_pic.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Detect faces and draw a rectangle
faces = face_cascade.detectMultiScale(image_rgb)
for (x,y,w,h) in faces:
        cv2.rectangle(image_rgb, (x, y),
                (x + h, y + w),
                (0, 255, 0), 5)


# Show the image
plt.subplot(1, 1, 1)
plt.imshow(image_rgb)
plt.show()
```



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 آبشاری اعمال می‌شود.

در حالی که جزئیات این فرآیند خارج از محدوده این کتاب است، قابل توجه است که این مدل‌های از پیش آموزش‌دیده شده را می‌توان به راحتی از مکان‌هایی مانند OpenCV GitHub 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

۸.۱۵ طبقه بندی تصاویر با Pytorch

مسئله

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

راه‌حل

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

```
# Load libraries
import cv2
import json
import numpy as np
import torch
from torchvision import transforms
from torchvision.models import resnet18
import urllib.request


# Get imagenet classes
with urllib.request.urlopen(
        "https://raw.githubusercontent.com/raghakot/keras-vis/master/resources/"
        ):
        imagenet_class_index = json.load(url)

# Instantiate pretrained model
model = resnet18(pretrained=True)


# Load image
image_bgr = cv2.imread("images/plane.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)


# Convert to pytorch data type
convert_tensor = transforms.ToTensor()
pytorch_image = convert_tensor(np.array(image_rgb))


# Set model to evaluation mode
model.eval()
# Make a prediction
prediction = model(pytorch_image.unsqueeze(0))


# Get the index of the highest predicted probability
_, index = torch.max(prediction, 1)
# Convert that to a percentage value
percentage = torch.nn.functional.softmax(prediction, dim=1)[0] * 100


# Print the name of the item at the index along with the percent confidence
print(imagenet_class_index[str(index.tolist()[0])][1],
        percentage[index.tolist()[0]].item())
airship 6.0569939613342285
```

بحث

بسیاری از مدل‌های آموزش عمیق از پیش آموزش‌دیده 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 ۱۸ 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 ۸.۱۶)، که ما می‌توانیم از کلاس‌های تصویر پیش‌بینی‌شده به‌عنوان ویژگی‌های پایین‌رونده[۱۶] 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 آموزش‌دیده

۱۴.۶ ارزیابی جنگل‌های تصادفی[۱۷] با خطاهای خارج از کیسه[۱۸]

مسئله

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

راه‌حل

امتیاز خارج از کیف مدل را محاسبه کنید:

```
# Load libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn import datasets

# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# Create random forest classifier object
randomforest = RandomForestClassifier(
        random_state=0, n_estimators=1000, oob_score=True, n_jobs=-1)

# Train model
model = randomforest.fit(features, target)

# View out-of-bag-error
randomforest.oob_score_
0.9533333333333334
```

بحث

[16] - downstream
[17] - Random Forests
[18] - Out-of-Bag Erros

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.

۱۴.۱۲ باران مدل XGBoost

مسئله

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

راه‌حل

از کتابخانه xgboost پایتون استفاده کنید:

```
# Load libraries
import xgboost as xgb
from sklearn import datasets, preprocessing
from sklearn.metrics import classification_report
from numpy import argmax

# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target

# Create dataset
xgb_train = xgb.DMatrix(features, label=target)

# Define parameters
param = {
        'objective': 'multi:softprob',
        'num_class': 3
}

# Train model
gbm = xgb.train(param, xgb_train)

# Get predictions
predictions = argmax(gbm.predict(xgb_train), axis=1)

# Get a classification report
print(classification_report(target, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      0.96      0.98        50
           2       0.96      1.00      0.98        50

    accuracy                           0.99       150
   macro avg       0.99      0.99      0.99       150
weighted avg       0.99      0.99      0.99       150
```

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

۱۴.۶ 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:

```
# Load libraries
import lightgbm as lgb
from sklearn import datasets, preprocessing
from sklearn.metrics import classification_report
from numpy import argmax


# Load data
iris = datasets.load_iris()
features = iris.data
target = iris.target


# Create dataset
lgb_train = lgb.Dataset(features, target)


# Define parameters
params = {
        'objective': 'multiclass',
        'num_class': 3,
        'verbose': -1,
}



# Train model
gbm = lgb.train(params, lgb_train)


# Get predictions
predictions = argmax(gbm.predict(features), axis=1)


# Get a classification report
print(classification_report(target, predictions))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	50
2	1.00	1.00	1.00	50
accuracy			1.00	150
macro avg	1.00	1.00	1.00	150
weighted avg	1.00	1.00	1.00	150

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
- مستندات CatBoost (یکی دیگر از کتابخانه‌های بهینه شده برای GBM)

۱۵.۵ پیدا کردن نزدیکترین همسایه‌های تقریبی (ANN)

مسئله

می‌خواهید نزدیک‌ترین همسایگان را برای داده‌های بزرگ با تأخیر کم را واکِشی[19] کنید:

راه‌حل

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

19 - fetch

```
# Load libraries
import faiss
import numpy as np
from sklearn import datasets
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler

# Load data
iris = datasets.load_iris()
features = iris.data

# Create standardizer
standardizer = StandardScaler()

# Standardize features
features_standardized = standardizer.fit_transform(features)

# Set faiss parameters
n_features = features_standardized.shape[1]
nlist = 3
k = 2

# Create an IVF index
quantizer = faiss.IndexFlatIP(n_features)
index = faiss.IndexIVFFlat(quantizer, n_features, nlist)

# Train the index and add feature vectors
index.train(features_standardized)
index.add(features_standardized)

# Create an observation
new_observation = np.array([[ 1, 1, 1, 1]])

# Search the index for the 2 nearest neighbors
distances, indices = index.search(new_observation, k)

# Show the feature vectors for the two nearest neighbors
np.array([list(features_standardized[i]) for i in indices[0]])
array([[1.03800476, 0.55861082, 1.10378283, 1.18556721],
       [0.79566902, 0.32841405, 0.76275827, 1.05393502]])
```

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 کـه مـا بـه نتـایج کمتـر از ۱۰۰ میلی‌ثانیـه نیـاز داریـم، اجـرای KNN غیرممکن می‌شود.

ANN به ما کمک می‌کند تا با قربانی کردن مقداری از کیفیتِ جستجوی دقیقِ نزدیکترین همسایگان به نفع سرعت، بر ایـن مشکل غلبه کنیم. این بدان معناست که اگرچه ترتیب و موارد در ۱۰ همسایه اول یک جستجوی ANN ممکن اسـت بـا ۱۰ نتیجه اول از یک جستجوی دقیق KNN مطابقت نداشته باشد، ما آن ۱۰ 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 اسـتفاده از خوشـه بندی برای محدود کردن دامنه فضای جستجو برای جستجوی نزدیکترین همسـایگان مـا کـار می‌کنـد. IVF از تسسـلات[۲۰] Voronoi برای تقسیم فضای جستجوی ما به تعـدادی منطقـه (یـا خوشـه) مجـزا اسـتفاده می‌کنـد. و وقتـی بـرای یـافتن نزدیک‌ترین همسایگان می‌رویم، از تعداد محدودی از خوشه‌ها بازدید می‌کنیم تا مشاهدات مشـابهی را پیـدا کنـیم، بـرخلاف 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 کـه در شکل ۱۵-۱ نشان داده شده است، یک نمودار پراکنده از داده‌های تصادفی که در دو بعد مشاهده می‌شود، در نظر بگیرید.

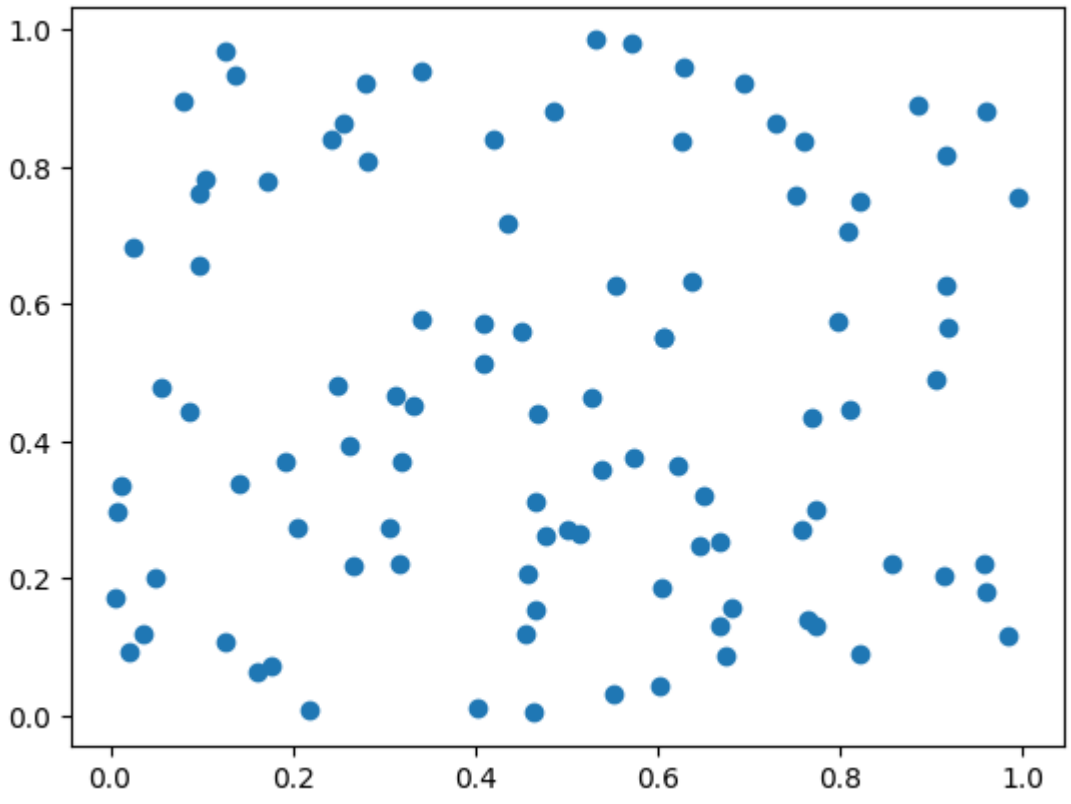


Figure 15-1. A scatter plot of randomly generated two-dimensional data

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 ۱۵-۲ نشان داده شده است.

[20] - tessellations

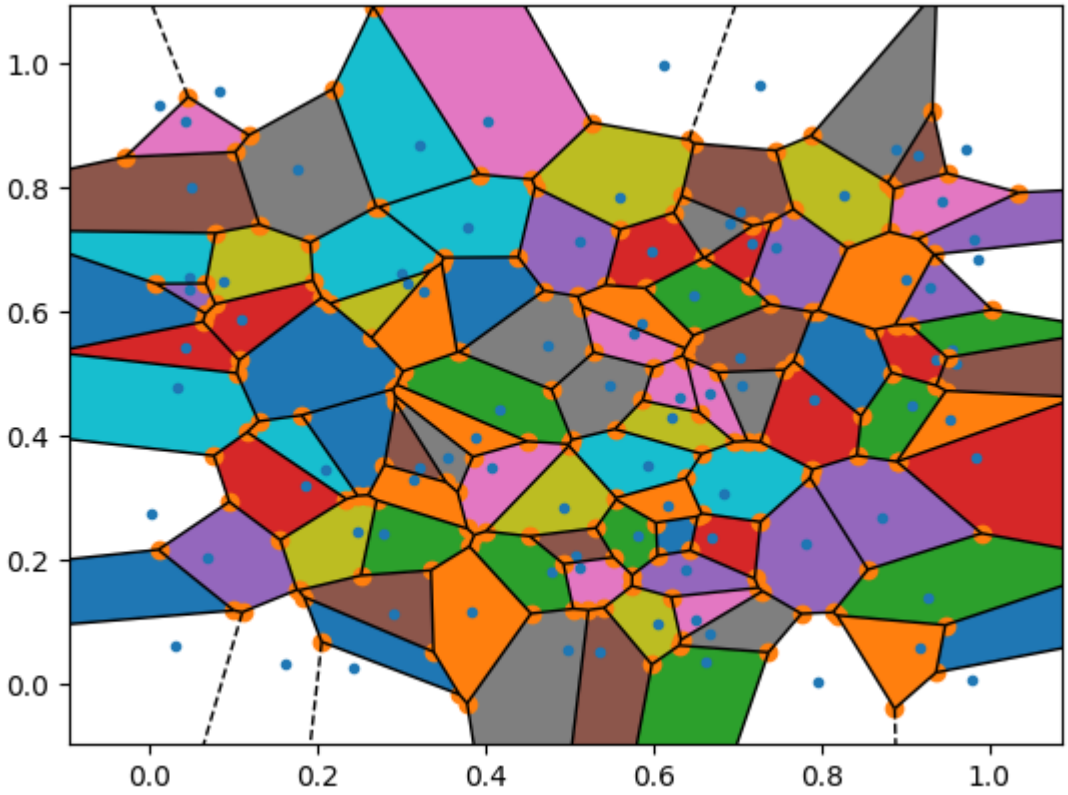


Figure 15-2. Randomly generated two-dimensional data separated into a number of different subspaces

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)

۱۵.۶ ارزیابی تقریبی نزدیکترین همسایگان

مسئله

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

راه‌حل

فراخوان[۲۱] k@ نزدیکترین همسایه ANN را در مقایسه با KNN محاسبه کنید:

[21] - recall

```
# Load libraries
import faiss
import numpy as np
from sklearn import datasets
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler

# Number of nearest neighbors
k = 10

# Load data
iris = datasets.load_iris()
features = iris.data

# Create standardizer
standardizer = StandardScaler()

# Standardize features
features_standardized = standardizer.fit_transform(features)

# Create KNN with 10 NN
nearest_neighbors = NearestNeighbors(n_neighbors=k).fit(features_standardized)

# Set faiss parameters
n_features = features_standardized.shape[1]
nlist = 3

# Create an IVF index
quantizer = faiss.IndexFlatIP(n_features)
index = faiss.IndexIVFFlat(quantizer, n_features, nlist)

# Train the index and add feature vectors
index.train(features_standardized)
index.add(features_standardized)
index.nprobe = 1

# Create an observation
new_observation = np.array([[ 1, 1, 1, 1]])

# Find distances and indices of the observation's exact nearest neighbors
knn_distances, knn_indices = nearest_neighbors.kneighbors(new_observation)

# Search the index for the two nearest neighbors
ivf_distances, ivf_indices = index.search(new_observation, k)
```

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 ۱۰ نزدیکترین همسایه، ما ۱۰۰٪ 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[۲۲] 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

فصل ۲۱- Tensorها با PyTorch

۲۰.۰ مقدمه

همانطور که NumPy یک ابزار اساسی برای دستکاری داده‌ها در پشته‌ی یادگیری ماشین[۲۳] 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 ۱ 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.

۲۰.۱ ساخت یک Tensor

مسئله

شما باید یک Tensor ایجاد کنید.

راه‌حل

از Pytorch برای ایجاد یک Tensor استفاده کنید:

---

[22] - recall

[23] - mechine learning stack

```
# Load library
import torch


# Create a vector as a row
tensor_row = torch.tensor([1, 2, 3])


# Create a vector as a column
tensor_column = torch.tensor(
        [
                [1],
                [2],
                [3]
        ]
)
```

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 ۱ 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

۲۰.۲ 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:

```
# Import libraries
import numpy as np
import torch


# Create a NumPy array
vector_row = np.array([1, 2, 3])


# Create a tensor from a NumPy array
tensor_row = torch.from_numpy(vector_row)
```

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

۲۰.۳ 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:

```
# Import libraries
import torch


# Create a tensor
tensor = torch.tensor(
[ [
        0, 0],
        [0, 1],
        [3, 0]
] )


# Create a sparse tensor from a regular tensor
sparse_tensor = tensor.to_sparse()
```

بحث

Tensorهای پراکنده، روش‌های کارآمد حافظه برای نمایش داده‌هایی هستند که عمدتاً از صفرها تشکیل شده‌اند. در فصل ۱ 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:

```
print(type(tensor))
print(type(sparse_tensor))
<class 'torch.Tensor'>
<class 'torch.Tensor'>
```

همچنین ببینید:

- مستندات PyTorch: Sparce Tensor

۲۰.۴ انتخاب عناصر در یک Tensor

مسئله

ما باید عناصر خاصی از یک Tensor را انتخاب کنیم.

راه‌حل

برای برگرداندن عناصر از نمایه سازی[۲۴] و برش[۲۵] NumPy استفاده کنید:

```
# Load library
import torch


# Create vector tensor
vector = torch.tensor([1, 2, 3, 4, 5, 6])


# Create matrix tensor
matrix = torch.tensor(
        [
                [1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]
        ]
)



#
Select third element of vector
vector[2]
tensor(3)


# Select second row, second column
matrix[1,1]
tensor(5)
```

بحث

[24] - indexing
[25] - slicing

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:

```
# Select all elements of a vector
vector[:]
array([1, 2, 3, 4, 5, 6])


# Select everything up to and including the third element
vector[:3]
tensor([1, 2, 3])


# Select everything after the third element
vector[3:]
tensor([4, 5, 6])


# Select the last element
vector[-1]
tensor(6)


# Select the first two rows and all columns of a matrix
matrix[:2,:]
tensor([[1, 2, 3],
        [4, 5, 6]])


# Select all rows and the second column
matrix[:,1:2]
tensor([[2],
        [5],
        [8]])
```

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:

```
# Reverse the vector
vector[::-1]
ValueError: step must be greater than zero
```

در عوض، اگر بخواهیم یک Tensor را معکوس کنیم، می‌توانیم از روش تلنگر[۲۶] استفاده کنیم:

```
vector.flip(dims=(-1,))
tensor([6, 5, 4, 3, 2, 1])
```

[26] - flip

همچنین ببینید:

- مستندات PyTorch: عملیات روی تنسورها

۲۰.۵ 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:

```
# Load library
import torch

# Create a tensor
tensor = torch.tensor([[1,2,3], [1,2,3]])

# Get the shape of the tensor
tensor.shape
torch.Size([2, 3])

# Get the data type of items in the tensor
tensor.dtype
torch.int64

# Get the layout of the tensor
tensor.layout
torch.strided

# Get the device being used by the tensor
tensor.device
device(type='cpu')
```

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.

۲۰.۶ اعمال عملیات بر روی عناصر

مسئله

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

راه‌حل

از قابلیت پخش[۲۷] با PyTorch استفاده کنید:

```
# Load library
import torch

# Create a tensor
tensor = torch.tensor([1, 2, 3])

# Broadcast an arithmetic operation to all elements in a tensor
tensor * 100
tensor([100, 200, 300])
```

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.

[27] - broadcasting

همچنین ببینید:

- مستندات PyTorch: معناشناسی Broadcasting
- Vectorization و Broadcasting با PyTorch

۲۰.۷ 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:

```
# Load library
import torch


# Create a tensor
torch.tensor([1,2,3])


# Find the largest value
tensor.max()
tensor(3)


# Find the smallest value
tensor.min()
tensor(1)
```

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:

```
# Create a multidimensional tensor
tensor = torch.tensor([[1,2,3],[1,2,5]])


# Find the largest value
tensor.max()
tensor(5)
```

۲۰.۸ 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:

```
# Load library
import torch


# Create 4x3 tensor
tensor = torch.tensor([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9],
                       [10, 11, 12]])


# Reshape tensor into 2x6 tensor
tensor.reshape(2, 6)
tensor([[ 1, 2, 3, 4, 5, 6],
        [ 7, 8, 9, 10, 11, 12]])
```

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.

۲۰.۹ انتقال یک Tensor

مسئله

شما باید یک Tensor را جابجا کنید.

راه‌حل

از تابع mT استفاده کنید:

```
# Load library
import torch


# Create a two-dimensional tensor
tensor = torch.tensor([[[1,2,3]]])


# Transpose it
tensor.mT
tensor([[1],
        [2],
        [3]])
```

بحث

انتقال با PyTorch کمی‌با NumPy متفاوت است. روش T مورد استفاده بـرای آرایـه‌های NumPy در PyTorch فقـط بـا Tensorهای دو بعدی پشتیبانی می‌شود و در زمان نوشتن برای Tensorهای اَشکال دیگر منسوخ شده است. روش mT کـه 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:

```
tensor.permute(*torch.arange(tensor.ndim - 1, -1, -1))
tensor([[1],
        [2],
        [3]])
```

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

۲۰.۱۰ صاف و مسطح کردن یک Tensor

مسئله

شما باید یک Tensor را به یک بعد تبدیل کنید.

راه‌حل

از تابع flatten استفاده کنید:

```
# Load library
import torch


# Create tensor
tensor = torch.tensor([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])


# Flatten tensor
tensor.flatten()
tensor([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

بحث

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

۲۰.۱۱ محاسبه نتایج نقطه ای

مسئله

شما باید حاصل ضرب نقطه‌ای دو Tensor را محاسبه کنید.

راه‌حل

از متد dot استفاده کنید:

```
# Load library
import torch


# Create one tensor
tensor_1 = torch.tensor([1, 2, 3])


# Create another tensor
tensor_2 = torch.tensor([4, 5, 6])


# Calculate the dot product of the two tensors
tensor_1.dot(tensor_2)
tensor(32)
```

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

۲۰.۱۲ ضرب Tensorها

مسئله

شما باید دو Tensor را ضرب کنید.

راه‌حل

از عملگرهای اساسی حساب پایتون استفاده کنید:

```
# Load library
import torch

# Create one tensor
tensor_1 = torch.tensor([1, 2, 3])

# Create another tensor
tensor_2 = torch.tensor([4, 5, 6])

# Multiply the two tensors
tensor_1 * tensor_2
tensor([ 4, 10, 18])
```

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:

```
tensor_1+tensor_2
tensor([5, 7, 9])
```

تفریق یک Tensor از دیگری:

```
tensor_1-tensor_2
tensor([-3, -3, -3])
```

تقسیم یک Tensor بر دیگری:

```
tensor_1/tensor_2
tensor([0.2500, 0.4000, 0.5000])
```

۲۱.۱ 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:

```
# Import libraries
import torch


# Create a torch tensor that requires gradients
t = torch.tensor([1.0, 2.0, 3.0], requires_grad=True)


# Perform a tensor operation simulating "forward propagation"
tensor_sum = t.sum()


# Perform back propagation
tensor_sum.backward()


# View the gradients
t.grad
tensor([1., 1., 1.])
```

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»[۲۸] به آرایه‌های NumPy تبدیل کنیم، عبارتی که برای توصیف عملیات‌هایی که از autograd پشتیبانی نمی‌کنند استفاده می‌شود:

[28] - breaking the graph

```
import torch


tensor = torch.tensor([1.0,2.0,3.0], requires_grad=True)
tensor.numpy()
RuntimeError: Can't call numpy() on Tensor that requires grad. Use
        tensor.detach().numpy() instead.
```

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

فصل ۲۲ - شبکه‌های عصبی برای داده‌های بدون ساختار

۲۲.۰ مقدمه

در فصل قبل، ما بر روی دستور العمل‌های شبکه عصبی برای داده‌های ساخت یافته، یعنی داده‌های جدولی تمرکز کردیم. بسیاری از بزرگترین پیشرفت‌ها در چند سال گذشته در واقع شامل استفاده از شبکه‌های عصبی و یادگیری عمیق برای داده‌های بدون ساختار، مانند متن یا تصاویر بوده است. کار با این مجموعه داده‌های بدون ساختار کمی متفاوت از کار با منابع داده‌های ساخت یافته است.

یادگیری عمیق به‌ویژه در فضای داده‌های بدون ساختار قدرتمند است، جایی که تکنیک‌های یادگیری ماشینی «کلاسیک» (مانند درخت‌های تقویت‌شده) معمولاً نمی‌توانند تمام پیچیدگی‌ها و تفاوت‌های ظریف موجود در داده‌های متنی، صدا، تصاویر، ویدیوها و غیره را ثبت کنند. ما در این فصل استفاده از یادگیری عمیق را به طور خاص برای داده‌های متن و تصویر بررسی خواهیم کرد.

در یک فضای یادگیری تحت نظارت برای متن و تصاویر، وظایف فرعی یا «انواع» یادگیری بسیاری وجود دارد. در زیر چند نمونه آورده شده است (اگرچه این فهرست جامعی نیست):

- طبقه بندی متن یا تصویر (مثال: طبقه بندی اینکه آیا یک تصویر، تصویر یک هات داگ است یا خیر)
- انتقال یادگیری (مثال: استفاده از یک مدل زمینه‌ای از پیش آموزش‌دیده‌شده مانند BERT و تنظیم دقیق آن در یک کار برای پیش بینی هرزنامه بودن یا نبودن ایمیل)
- تشخیص اشیا (به عنوان مثال: شناسایی و طبقه بندی اشیاء خاص در یک تصویر)
- مدل‌های مولد (مثال: مدل‌هایی که متن را بر اساس یک ورودی مشخص مانند مدل‌های GPT تولید می‌کنند)

با افزایش محبوبیت یادگیری عمیق و کالایی شدن روزافزون آن، راه حل‌های منبع باز و سازمانی برای مقابله با این موارد استفاده آسان تر شده اند. در این فصل، ما از چند کتابخانه کلیدی به عنوان نقطه‌ی ورود خود برای انجام این وظایف یادگیری عمیق استفاده خواهیم کرد. به طور خاص، ما از کتابخانه‌های PyTorch، Torchvision و Transformers در Python برای انجام مجموعه‌ای از وظایف و عملکردها در داده‌های متنی و تصویری ML استفاده می‌کنیم.

۲۲.۱ آموزش شبکه عصبی برای طبقه بندی تصاویر

مسئله

شما باید یک شبکه عصبی طبقه بندی تصویر را آموزش دهید.

راه‌حل

از یک شبکه عصبی کانولوشنال[۲۹] در PyTorch استفاده کنید:

[29] - Convolutional Neural Network (CNN)

```
# Import libraries
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms


# Define the convolutional neural network architecture
class Net(nn.Module):
        def __init__(self):
                super(Net, self).__init__()
                self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
                self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
                self.dropout1 = nn.Dropout2d(0.25)
                self.dropout2 = nn.Dropout2d(0.5)
                self.fc1 = nn.Linear(64 * 14 * 14, 128)
                self.fc2 = nn.Linear(128, 10)


        def forward(self, x):
                x = nn.functional.relu(self.conv1(x))
                x = nn.functional.relu(self.conv2(x))
                x = nn.functional.max_pool2d(self.dropout1(x), 2)
                x = torch.flatten(x, 1)
                x = nn.functional.relu(self.fc1(self.dropout2(x)))
                x = self.fc2(x)
                return nn.functional.log_softmax(x, dim=1)


# Set the device to run on
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")


# Define the data preprocessing steps
transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
])


# Load the MNIST dataset
train_dataset = datasets.MNIST('./data', train=True, download=True,
        transform=transform)
test_dataset = datasets.MNIST('./data', train=False, transform=transform)


# Create data loaders
batch_size = 64
train_loader = torch.utils.data.DataLoader(train_dataset,
        batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,
        shuffle=True)


# Initialize the model and optimizer
model = Net().to(device)
```

بحث

شبکه‌های عصبی کانولوشنال معمولاً برای عملکردهایی در تشخیص تصویر و بینایی کامپیوتری استفاده می‌شوند. آنها معمولاً از لایه‌های کانولوشن، لایه‌های ترکیبی و یک لایه کاملاً متصل تشکیل شده‌اند.

هدف از لایه‌های کانولوشن، یادگیری ویژگی‌های مهم تصویر است. لایه‌های کانولوشن با اعمال یک فیلتر در ناحیه خاصی از تصویر (به اندازه کانولوشن) کار می‌کنند. وزن‌های این لایه سپس یاد می‌گیرند که ویژگی‌های تصویری خاص را که در کار طبقه‌بندی حیاتی هستند، تشخیص دهند. به عنوان مثال، اگر مدلی را آموزش دهیم که دست فرد را تشخیص دهد، ممکن است فیلتر تشخیص انگشتان را بیاموزد.

هدف لایه ادغام معمولاً کاهش ابعاد ورودی‌ها از لایه قبلی است. این لایه همچنین از فیلتر اعمال شده بر روی بخشی از ورودی استفاده می‌کند، اما هیچ فعال سازی ندارد. در عوض، ابعاد ورودی را با انجام حداکثر ادغام (جایی که پیکسل را در فیلتر با بالاترین مقدار انتخاب می‌کند) یا ادغام متوسط (که در آن به طور متوسط از پیکسل‌های ورودی استفاده می‌شود) کاهش می‌دهد.

در نهایت، لایه‌ی کاملا متصل را می‌توان با چیزی مانند تابع فعال سازی softmax برای ایجاد یک طبقه بندی باینری استفاده کرد.

همچنین ببینید:

- شبکه‌های عصبی کانولوشنال (CNN)

۲۲.۲ آموزش شبکه عصبی برای طبقه بندی متن

مسئله

برای طبقه بندی داده‌های متنی باید یک شبکه عصبی آموزش دهید.

راه‌حل

از یک شبکه عصبی PyTorch استفاده کنید که اولین لایه آن به اندازه دایره لغات شما باشد:

```
# Import libraries
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


# Load the 20 newsgroups dataset
cats = ['alt.atheism', 'sci.space']
newsgroups_data = fetch_20newsgroups(subset='all', shuffle=True,
        random_state=42, categories=cats)


# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(newsgroups_data.data,
        newsgroups_data.target, test_size=0.2, random_state=42)


# Vectorize the text data using a bag-of-words approach
vectorizer = CountVectorizer(stop_words='english')
X_train = vectorizer.fit_transform(X_train).toarray()
X_test = vectorizer.transform(X_test).toarray()


# Convert the data to PyTorch tensors
X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_test = torch.tensor(y_test, dtype=torch.long)


# Define the model
class TextClassifier(nn.Module):
        def __init__(self, num_classes):
                super(TextClassifier, self).__init__()
                self.fc1 = nn.Linear(X_train.shape[1], 128)
                self.fc2 = nn.Linear(128, num_classes)


        def forward(self, x):
                x = nn.functional.relu(self.fc1(x))
                x = self.fc2(x)
                return nn.functional.log_softmax(x, dim=1)


# Instantiate the model and define the loss function and optimizer
model = TextClassifier(num_classes=len(cats))
loss_function = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
```

بحث

برخلاف تصاویر، داده‌های متنی ذاتا غیر عددی هستند. قبل از آموزش یک مدل، باید متن را به یک نمایش عددی تبدیل کنیم که مدل بتواند از آن استفاده کند تا بفهمد کدام کلمات و ترکیب‌های کلمه برای طبقه‌بندی مهم هستند. در این مثال، از CountVetorizer در scikit-learn برای رمزگذاری واژگان به عنوان بردار به اندازه‌ی کل واژگان استفاده می‌کنیم، جایی که هر کلمه به یک شاخص خاص در بردار اختصاص داده می‌شود، و مقدار در آن مکان، تعداد دفعات تکرار آن کلمه است که در یک پاراگراف مشخص ظاهر می‌شود. در این مورد، با نگاه کردن به مجموعه آموزشی خود می‌توانیم اندازه واژگان را ببینیم:

```
X_train.shape[1]
25150
```

ما از همین مقدار در اولین لایه شبکه عصبی خود برای تعیین اندازه لایه ورودی استفاده می‌کنیم: self.fc1 = nn.Linear(X_train.shape[1], 128). این عملکرد به شبکه ما اجازه می‌دهد تا آنچه را که به آن تعبیه‌های کلمه می‌گویند بیاموزد، یعنی بازنمایی برداری از کلماتی که از یک یادگیری تحت نظارت مانند کاری که در این بخش انجام دادیم آمده است. این کار به ما امکان می‌دهد جاسازی‌های کلمات با اندازه ۱۲۸ را یاد بگیریم، اگرچه این جاسازی‌ها در درجه اول برای این کار خاص و واژگان مفید خواهند بود.

۲۲.۳ تنظیم دقیق یک مدل از پیش آموزش‌دیده برای طبقه بندی تصویر

مسئله

شما می‌خواهید یک مدل طبقه بندی تصویر را با استفاده از آموخته‌های یک مدل از پیش آموزش‌دیده آموزش دهید.

راه‌حل

از کتابخانه transformers با torchvision برای تنظیم دقیق یک مدل از پیش آموزش‌دیده روی داده‌های خود استفاده کنید:

```
# Import libraries
import torch
from torchvision.transforms import(
        RandomResizedCrop, Compose, Normalize, ToTensor
        )
from transformers import Trainer, TrainingArguments, DefaultDataCollator
from transformers import ViTFeatureExtractor, ViTForImageClassification
from datasets import load_dataset, load_metric, Image


# Define a helper function to convert the images into RGB
def transforms(examples):
        examples["pixel_values"] = [_transforms(img.convert("RGB")) for img in
                examples["image"]]
        del examples["image"]
        return examples


# Define a helper function to compute metrics
def compute_metrics(p):
        return metric.compute(predictions=np.argmax(p.predictions, axis=1),
                references=p.label_ids)


# Load the fashion mnist dataset
dataset = load_dataset("fashion_mnist")


# Load the processor from the VIT model
image_processor = ViTFeatureExtractor.from_pretrained(
        "google/vit-base-patch16-224-in21k"
)


# Set the labels from the dataset
labels = dataset['train'].features['label'].names


# Load the pretrained model
model = ViTForImageClassification.from_pretrained(
        "google/vit-base-patch16-224-in21k",
        num_labels=len(labels),
        id2label={str(i): c for i, c in enumerate(labels)},
        label2id={c: str(i) for i, c in enumerate(labels)}
)


# Define the collator, normalizer, and transforms
collate_fn = DefaultDataCollator()
normalize = Normalize(mean=image_processor.image_mean,
        std=image_processor.image_std)
size = (
        image_processor.size["shortest_edge"]
        if "shortest_edge" in image_processor.size
        else (image_processor.size["height"], image_processor.size["width"])
)
```

بحث

در حوزه داده‌های بدون ساختار مانند متن و تصاویر، به‌جای شروع از ابتدا، به‌ویژه در مواردی که به داده‌های برچسب‌گذاری‌شده زیادی دسترسی نداریم، بسیار رایج است که از مدل‌های از پیش آموزش‌دیده‌شده روی مجموعه داده‌های بزرگ شروع کنیم. با استفاده از جاسازی‌ها و اطلاعات دیگر از مدل بزرگ‌تر، می‌توانیم مدل خود را برای یک کار جدید بدون نیاز به اطلاعات برچسب‌گذاری شده زیادی تنظیم کنیم. علاوه بر این، مدل از پیش آموزش‌دیده ممکن است اطلاعاتی داشته باشد که اصلاً در مجموعه داده آموزشی ما ثبت نشده باشد، که منجر به بهبود عملکرد کلی می‌شود. این فرآیند به یادگیری انتقالی معروف است.

در این مثال، وزن‌ها را از مدل ViT (Vision Transformer) گوگل بارگیری می‌کنیم. سپس، از کتابخانه transformers برای تنظیم دقیق آن برای یک کار طبقه‌بندی در مجموعه داده‌های مد MNIST، مجموعه داده‌ای ساده از اقلام لباس، استفاده می‌کنیم. این رویکرد را می‌توان برای افزایش عملکرد در هر مجموعه داده بینایی کامپیوتری اعمال کرد، و کتابخانه transformers یک رابط سطح بالایی را ارائه می‌دهد که می‌توانیم از آن برای تنظیم دقیق مدل خود از مدل‌های بزرگتر و از پیش آموزش‌دیده‌شده بدون نوشتن مقدار زیادی کد استفاده کنیم.

همچنین ببینید:

- وب‌سایت و مستندات Hugging Face

۲۲.۴ تنظیم دقیق یک مدل از پیش آموزش‌دیده‌شده برای طبقه بندی متن

مسئله

شما می‌خواهید یک مدل طبقه بندی متن را با استفاده از آموخته‌های یک مدل از پیش آموزش‌دیده‌شده آموزش دهید.

راه‌حل

از کتابخانه transformers استفاده کنید:

```python
# Import libraries
from datasets import load_dataset
from transformers import AutoTokenizer, DataCollatorWithPadding
from transformers import (
        AutoModelForSequenceClassification, TrainingArguments, Trainer
        )
import evaluate
import numpy as np


# Load the imdb dataset
imdb = load_dataset("imdb")


# Create a tokenizer and collator
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)


# Tokenize the imdb dataset
tokenized_imdb = imdb.map(
        lambda example: tokenizer(
                example["text"], padding="max_length", truncation=True
        ),
        batched=True,
)



# User the accuracy metric
accuracy = evaluate.load("accuracy")


# Define a helper function to produce metrics
def compute_metrics(eval_pred):
        predictions, labels = eval_pred
        predictions = np.argmax(predictions, axis=1)
        return accuracy.compute(predictions=predictions, references=labels)


# Create dictionaries to map indices to labels and vice versa
id2label = {0: "NEGATIVE", 1: "POSITIVE"}
label2id = {"NEGATIVE": 0, "POSITIVE": 1}


# Load a pretrained model
model = AutoModelForSequenceClassification.from_pretrained(
        "distilbert-base-uncased", num_labels=2, id2label=id2label,
        label2id=label2id
)



# Specify the training arguments
training_args = TrainingArguments(
        output_dir="my_awesome_model",
        learning_rate=2e-5,
        per_device_train_batch_size=16,
```

بحث

درست مانند استفاده از مدل‌های تصویری از پیش آموزش‌دیده شده، مدل‌های زبانی از قبل آموزش‌دیده شده، زمینه‌ی زیادی در مورد زبان دارند، زیرا معمولاً در طیف گسترده‌ای از منابع اینترنتیِ باز آموزش داده می‌شوند. هنگامی که از یک پایه مدل از پیش آموزش‌دیده شده شروع می‌کنیم، کاری که معمولا انجام می‌دهیم این است که لایه طبقه بندی شبکه موجود را با یکی از لایه‌های خود عوض می‌کنیم. این مورد به ما امکان می‌دهد تا وزن‌های شبکه را که قبلاً آموخته‌ایم، متناسب با وظایف خاص خود تغییر دهیم.

در این مثال، ما در حال تنظیم دقیق مدل DistilBERT هستیم تا تشخیص دهیم که آیا نقدهای فیلم IMDB مثبت (۱) یا منفی (۰) بوده است. مدل DistilBERT از قبل آموزش داده شده است که مجموعه بزرگی از کلمات و زمینه را در هر یک، علاوه بر وزن شبکه عصبی که از آموزشی قبلی آموخته شده است، ارائه می‌دهد. یادگیری انتقالی به ما این امکان را می‌دهد که از تمام کارهای اولیه انجام شده در آموزش مدل DistilBERT استفاده کنیم و آن را برای مورد استفاده خود، که در این مثال طبقه بندی نقدهای فیلم است، تغییر دهیم.

همچنین ببینید:

- طبقه بندی متن در کتابخانه transformers

۲۳.۲ ذخیره و بارگذاری یک مدل TensorFlow

مسئله

شما یک مدل TensorFlow آموزش دیده دارید و می‌خواهید آن را ذخیره کنید و در جای دیگری بارگذاری کنید.

راه‌حل

مدل را با استفاده از فرمت TensorFlow saved_model ذخیره کنید:

```
# Load libraries
import numpy as np
from tensorflow import keras

# Set random seed
np.random.seed(0)

# Create model with one hidden layer
input_layer = keras.Input(shape=(10,))
hidden_layer = keras.layers.Dense(10)(input_layer)
output_layer = keras.layers.Dense(1)(input_layer)
model = keras.Model(input_layer, output_layer)
model.compile(optimizer="adam", loss="mean_squared_error")

# Train the model
x_train = np.random.random((1000, 10))
y_train = np.random.random((1000, 1))
model.fit(x_train, y_train)

# Save the model to a directory called `save_model`
model.save("saved_model")
32/32 [==============================] - 1s 8ms/step - loss: 0.2056
INFO:tensorflow:Assets written to: saved_model/assets
```

سپس می‌توانیم مدل را در برنامه دیگری یا برای آموزش اضافی بارگذاری کنیم:

```
# Load neural network
model = keras.models.load_model("saved_model")
```

بحث

اگرچه در طول این کتاب به طور قابل توجهی از TensorFlow استفاده نکردیم، اما دانستن نحوه ذخیره و بارگذاری مدل‌های TensorFlow مفید است. برخلاف scikit-learn که از فرمت ترشی Python-native pickle استفاده می‌کند، TensorFlow روش خود را برای ذخیره و بارگذاری مدل‌ها ارائه می‌دهد. فرمت saved_model یک دایرکتوری ایجاد

می‌کند که مدل و تمام اطلاعات لازم برای بارگیری مجدد آن و همچنین پیش بینی کردن را در فرمت پروتکل بافر[30] (کـه از پسوند فایل pb. استفاده می‌کند) ذخیره می‌کند.

```
ls saved_model
assets fingerprint.pb keras_metadata.pb saved_model.pb variables
```

در حالی که ما به طور عمیق وارد این قالب نمی شویم، این روش استاندارد ذخیره، بارگیری و ارائه مدل‌های آموزش دیده در TensorFlow است.

همچنین ببینید:

- سریال سازی و ذخیره مدل‌های Keras
- قالب مدل ذخیره شده TensorFlow

۲۳.۳ ذخیره و بارگذاری یک مدل PyTorch

مسئله

شما یک مدل PyTorch آموزش دیده دارید و می‌خواهید آن را ذخیره کنید و در جای دیگری بارگذاری کنید.

راه‌حل

از توابع torch.save و torch.load استفاده کنید:

30 - protocol buffer

```
# Load libraries
import torch
import torch.nn as nn
import numpy as np
from torch.utils.data import DataLoader, TensorDataset
from torch.optim import RMSprop
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split


# Create training and test sets
features, target = make_classification(n_classes=2, n_features=10,
        n_samples=1000)
features_train, features_test, target_train, target_test = train_test_split(
        features, target, test_size=0.1, random_state=1)


# Set random seed
torch.manual_seed(0)
np.random.seed(0)


# Convert data to PyTorch tensors
x_train = torch.from_numpy(features_train).float()
y_train = torch.from_numpy(target_train).float().view(-1, 1)
x_test = torch.from_numpy(features_test).float()
y_test = torch.from_numpy(target_test).float().view(-1, 1)


# Define a neural network using `Sequential`
class SimpleNeuralNet(nn.Module):
        def __init__(self):
                super(SimpleNeuralNet, self).__init__()
                self.sequential = torch.nn.Sequential(
                        torch.nn.Linear(10, 16),
                        torch.nn.ReLU(),
                        torch.nn.Linear(16,16),
                        torch.nn.ReLU(),
                        torch.nn.Linear(16, 1),
                        torch.nn.Dropout(0.1), # Drop 10% of neurons
                        torch.nn.Sigmoid(),
                )


        def forward(self, x):
                x = self.sequential(x)
                return x


# Initialize neural network
network = SimpleNeuralNet()
# Define loss function, optimizer
criterion = nn.BCELoss()
optimizer = RMSprop(network.parameters())
```

بحث

اگرچه ما از فرمول مشابهی در فصل ۲۱ برای بررسی پیشرفت آموزش خود استفاده کـردیم، در اینجـا مـی‌بینیم کـه چگونـه می‌توان از همان رویکرد برای بارگذاری مجدد یک مدل در حافظه برای انجام پیش بینی استفاده کرد. model.pt که مدل را در آن ذخیره می‌کنیم در واقع فقط یک فرهنگ لغت (Dictionary) است که شامل پارامترهای مدل است. ما حالت مـدل را در کلیدی به نام model_state_dict در این فرهنگ لغت ذخیره کردیم. برای بارگذاری مجدد مدل، شبکه خـود را مجـدداً مقداردهی اولیه می‌کنیم و وضعیت مدل را با استفاده از تابع network.load_state_dict بارگذاری می‌کنیم.

همچنین ببینید:

- آموزش PyTorch: ذخیره و بارگذاری مدل ها

۲۳.۴ ارائه مدل‌های sikit-learn

مسئله

شما می‌خواهید با استفاده از یک وب سرور، مدل **sikit-learn** آموزش دیده خود را ارائه دهید.

راه‌حل

یک برنامه با استفاده از Python Flask بسازید که مدل آموزش داده شده در این فصل را بارگیری می‌کند:

```
# Import libraries
import joblib
from flask import Flask, request


# Instantiate a flask app
app = Flask(__name__)


# Load the model from disk
model = joblib.load("model.pkl")


# Create a predict route that takes JSON data, makes predictions, and
# returns them
@app.route("/predict", methods = ["POST"])
def predict():
        print(request.json)
        inputs = request.json["inputs"]
        prediction = model.predict(inputs)
        return {
        "prediction" : prediction.tolist()
        }


# Run the app
if __name__ == "__main__":
app.run()
```

مطمئن شوید که Flask را نصب کرده اید:

```
python3 -m pip install flask==2.2.3 joblib==1.2.0 scikit-learn==1.2.0
```

و سپس برنامه را اجرا کنید:

```
python3 app.py
        * Serving Flask app 'app'
        * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production
WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

اکنون، می‌توانیم پیش‌بینی‌هایی را در برنامه انجام دهیم و با ارسال داده با استفاده از curl، نتیجه را ببینیم:

```
curl -X POST http://127.0.0.1:5000/predict -H 'Content-Type: application/json'  -d '{"inputs":[[5.1,
3.5, 1.4, 0.2]]}'
{"prediction":[0]}
```

بحث

در این مثال، ما با استفاده از Flask، یک کتابخانه منبع باز محبوب برای ساخت برنامه‌های تحت وب در پایتون استفاده کردیم. ما یک مسیر به نام /predict تعریف می‌کنیم که داده‌های JSON را در یک درخواست POST می‌گیرد و یک dictionary حاوی پیش بینی‌ها را برمی گرداند. اگرچه این سرور به صورت production نیست (به هشدار Flask در مورد استفاده از سرور توسعه مراجعه کنید)، ما به راحتی می‌توانیم این کد را با یک برنامه تحت وب آماده‌تر برای production گسترش دهیم.

۲۳.۵ ارائه مدل‌های تنسورفلو

مسئله

شما می‌خواهید مدل TensorFlow آموزش دیده خود را با استفاده از یک وب سرور ارائه دهید.

راه‌حل

از فریمورک منبع باز TensorFlow Serving و Docker استفاده کنید:

```
docker run -p 8501:8501 -p 8500:8500 \
--mount type=bind,source=$(pwd)/saved_model,target=/models/saved_model/1 \
-e MODEL_NAME=saved_model -t tensorflow/serving
```

بحث

TensorFlow Serving یک راه حل ارائه منبع باز است که برای مدل‌های TensorFlow بهینه شده است. با استفاده از این مورد، به سادگی با ارائه مسیر مدل، یک سرور HTTP و gRPC با ویژگی‌های مفید اضافی برای توسعه‌دهندگان دریافت می‌کنیم.

دستور docker run یک کانتینر را با استفاده از image عمومی tensorflow/serving اجرا می‌کند و مسیر saved_model دایرکتوری فعلی ما ($(pwd)/saved_model) را در /models/saved_model/1 در داخل container ما در برنامه‌ی Docker نصب می‌کند. این عملکرد به طور خودکار مدلی را که قبلاً در این فصل ذخیره کرده بودیم در یک Container در Docker در حال اجرا بارگیری می‌کند که می‌توانیم کوئری‌های پیش بینی را به آن ارسال کنیم.

اگر در مرورگر وب خود به http://localhost:8501/v1/models/saved_model بروید، باید نتیجه JSON را در اینجا ببینید:

```
{
        "model_version_status": [
                {
                        "version": "1",
                        "state": "AVAILABLE",
                        "status": {
                                "error_code": "OK",
                                "error_message": ""
                        }
                }
        ]
}
```


مسیر /metadata در http://localhost:8501/v1/models/saved_model/metadata اطلاعات بیشتری در مـورد مـدل ارائه می‌دهد:

```
{"
model_spec":{
 "name": "saved_model",
 "signature_name": "",
 "version": "1"
},
"metadata": {"signature_def": {
 "signature_def": {
  "serving_default": {
   "inputs": {
    "input_8": {
     "dtype": "DT_FLOAT",
     "tensor_shape": {
      "dim": [
      {
       "size": "-1",
       "name": ""
      },
      {
       "size": "10",
       "name": ""
      }
     ],
      "unknown_rank": false
    },
    "name": "serving_default_input_8:0"
  }
},
"outputs": {
 "dense_11": {
  "dtype": "DT_FLOAT",
  "tensor_shape": {
   "dim": [
    {
     "size": "-1",
     "name": ""
    },
    {
     "size": "1",
     "name": ""
    }
   ],
   "unknown_rank": false
  },
  "name": "StatefulPartitionedCall:0"
  }
 },
 "method_name": "tensorflow/serving/predict"
},
"__saved_model_init_op": {
 "inputs": {},
 "outputs": {
  "__saved_model_init_op": {
   "dtype": "DT_INVALID",
```

ما می‌توانیم با استفاده از curl و دادن متغیرها به آن، با استفاده از برنامه REST پیش بینی انجام دهیم (این شبکه عصبی ۱۰ ویژگی دارد):

```
curl -X POST http://localhost:8501/v1/models/saved_model:predict
        -d '{"inputs":[[1,2,3,4,5,6,7,8,9,10]]}'
{
        "outputs": [
                [
                        5.59353495
                ]
        ]
}
```

همچنین ببینید:

- مستندات TensorFlow: مدل‌های Serving

۲۳.۶ ارائه مدل‌های PyTorch در سِلدون[۳۱]

مسئله

شما می‌خواهید یک مدل PyTorch آموزش دیده برای پیش بینی‌های زمان واقعی ارائه دهید.

راه‌حل

مدل را با استفاده از Seldon Core Python wrapper ارائه دهید:

31 - Seldon

```
# Import libraries
import torch
import torch.nn as nn
import logging


# Create a PyTorch model class
class SimpleNeuralNet(nn.Module):
        def __init__(self):
                super(SimpleNeuralNet, self).__init__()
                self.sequential = torch.nn.Sequential(
                        torch.nn.Linear(10, 16),
                        torch.nn.ReLU(),
                        torch.nn.Linear(16,16),
                        torch.nn.ReLU(),
                        torch.nn.Linear(16, 1),
                        torch.nn.Dropout(0.1), # Drop 10% of neurons
                        torch.nn.Sigmoid(),
                )


# Create a Seldon model object with the name `MyModel`
class MyModel(object):

        # Loads the model
        def __init__(self):
                self.network = SimpleNeuralNet()
                self.network.load_state_dict(
                        torch.load("model.pt")["model_state_dict"],
                        strict=False
                )
                logging.info(self.network.eval())


        # Makes a prediction
        def predict(self, X, features_names=None):
                return self.network.forward(X)
```

و آن را با Docker اجرا کنید:

```
docker run -it -v $(pwd):/app -p 9000:9000 kylegallatin/seldon-example
        seldon-core-microservice MyModel --service-type MODEL
2023-03-11 14:40:52,277 - seldon_core.microservice:main:578 -
  INFO:  Starting microservice.py:main
2023-03-11 14:40:52,277 - seldon_core.microservice:main:579 -
  INFO:  Seldon Core version: 1.15.0
2023-03-11 14:40:52,279 - seldon_core.microservice:main:602 -
  INFO:  Parse JAEGER_EXTRA_TAGS []
2023-03-11 14:40:52,287 - seldon_core.microservice:main:605 -
        INFO: Annotations: {}
2023-03-11 14:40:52,287 - seldon_core.microservice:main:609 -
        INFO: Importing MyModel
2023-03-11 14:40:55,901 - root:__init__:25 - INFO: SimpleNeuralNet(
        (sequential): Sequential(
                (0): Linear(in_features=10, out_features=16, bias=True)
                (1): ReLU()
                 (2): Linear(in_features=16, out_features=16, bias=True)
                (3): ReLU()
                (4): Linear(in_features=16, out_features=1, bias=True)
                (5): Dropout(p=0.1, inplace=False)
                (6): Sigmoid()
        )
)
2023-03-11 14:40:56,024 - seldon_core.microservice:main:640 -
  INFO:  REST gunicorn microservice running on port 9000
2023-03-11 14:40:56,028 - seldon_core.microservice:main:655 -
  INFO:  REST metrics microservice running on port 6000
2023-03-11 14:40:56,029 - seldon_core.microservice:main:665 -
  INFO:  Starting servers
2023-03-11 14:40:56,029 - seldon_core.microservice:start_servers:80 -
  INFO:  Using standard multiprocessing library
2023-03-11 14:40:56,049 - seldon_core.microservice:server:432 -
  INFO:  Gunicorn Config:  {'bind': '0.0.0.0:9000', 'accesslog': None,
  'loglevel': 'info', 'timeout': 5000, 'threads': 1, 'workers': 1,
  'max_requests': 0, 'max_requests_jitter': 0, 'post_worker_init':
  <function post_worker_init at 0x7f5aee2c89d0>, 'worker_exit':
  functools.partial(<function worker_exit at 0x7f5aee2ca170>,
  seldon_metrics=<seldon_core.metrics.SeldonMetrics object at
  0x7f5a769f0b20>), 'keepalive': 2}
2023-03-11 14:40:56,055 - seldon_core.microservice:server:504 -
  INFO:  GRPC Server Binding to 0.0.0.0:5000 with 1 processes.
2023-03-11 14:40:56,090 - seldon_core.wrapper:_set_flask_app_configs:225 -
  INFO:  App Config:  <Config {'ENV': 'production', 'DEBUG': False,
```

بحث

در حالی که روش‌های مختلفی برای ارائه یک مدل PyTorch وجود دارد، در اینجا ما Seldon Core Python wrapper را انتخاب می‌کنیم. Seldon Core یک چارچوب محبوب برای ارائه مدل‌ها در prodution است و دارای تعدادی ویژگی مفید وجود دارند که استفاده از آن را آسان‌تر و مقیاس‌پذیرتر از یک برنامه Flask می‌کند. این مورد به ما امکان می‌دهد یک کلاس ساده بنویسیم (در کد بالا از MyModel استفاده کردیم)، در حالی که کتابخانه Python به تمام اجزای سرور و آدرس‌های مقصد اهمیت می‌دهد. سپس می‌توانیم سرویس را با استفاده از دستور seldon-core-microservice اجرا کنیم، که سرور REST، سرور gRPC را راه‌اندازی می‌کند و حتی یک آدرس مقصد متریک را نشان می‌دهد. برای انجام پیش‌بینی در این سرویس، می‌توانیم سرویس را با آدرس مقصد زیر در پورت ۹۰۰۰ فراخوانی کنیم:

```
curl -X POST http://127.0.0.1:9000/predict application/json' -H 'Content-Type:
        -d '{"data": {"ndarray":[[0, 0, 0, 0, 0, 0, 0, 0, 0]]}}'
```

شما باید خروجی زیر را ببینید:

```
{"data":{"names":["t:0","t:1","t:2","t:3","t:4","t:5","t:6","t:7","t:8"],
        "ndarray":[[0,0,0,0,0,0,0,0,0]]},"meta":{}}
```

همچنین ببینید:

- پکیج Seldon Core Python
- مستندات TorchServe