بنام خدا

رگرسیون با mlp و تنسورفلو

۱. استفاده از شبکه عصبی mlp برای رگرسیون

از شبکه‌های mlp می‌توان برای رگرسیون استفاده کرد .مثلا برای تخمین قیمت یک خانه، ویژگی‌های آن خانه را به شبکه می‌دهیم. سپس شبکه تخمین می‌زند که قیمت آن خانه چقدر است. در چنین مسائلی ما فقط یک نورون خروجی خواهیم داشت. که خروجی آن نورون، مقدار پیش‌بینی شده برای قیمت خانه است .در رگرسیون چندمتغیره به ازای هر متغیر یک نورون خروجی خواهیم داشت .به عنوان مثال برای تخمین مرکز یک دایره به ۲ عدد نیاز است. که مختصات مرکز در فضای دوبعدی مشخص شود .چنین مسئله‌ای نیاز به شبکه‌ای دارد که دو نورون خروجی داشته باشد.

به طور کلی برای استفاده از mlp در رگرسیون، نیاز نیست نورن‌های خروجی تابع فعالساز داشته باشند .در این صورت نورون‌های خروجی آزاد هستند که هر رنج عددی را تخمین بزنند .اگر بخواهید که خروجی شبکه همیشه مثبت باشد، آنگاه می‌توانید از یک تابع فعالسازی مانند ReLU استفاده کنید .اگر هم می‌خواهید خروجی در یک رنج خاص باشد، می‌توانید از توابع فعالسازی مانند سیگموید استفاده کنید .این توابع فعالسازی رنج خروجی را محدود به یک بازه می‌کنند. مثلا سیگموید رنج خروجی را بین ۰ و ۱ قرار خواهد داد.

۱,۱ طرح مسئله

در این بخش ما می‌خواهیم یک شبکه رگرسیون با mlp و تنسورفلو بسازیم. وظیفه شبکه‌ای که می‌سازیم این است که ویژگی‌های یک خانه را بگیرد و قیمت آن را تخمین بزند .ویژگی خانه می‌تواند عمر ساختمان، تعداد اتاق‌ها و ... باشد. برای آموزش این شبکه از پایگاه داده California Housing Prices استفاده خواهیم کرد. برای پیاده‌سازی این پروژه از فریمورک تنسورفلو ۲ و کراس استفاده می‌کنیم. همچنین کدنویسی در گوگل کولب انجام خواهد شد.

فراخوانی پایگاه داده

گفتیم پایگاه داده‌ای که استفاده خواهیم کرد، California Housing Pricesخواهد بود. در این پایگاه داده ویژگی‌های هر خانه به شکل زیر مشخص شده است:

- طول جغرافیایی
- عرض جغرافیایی

- قدمت بنا
- تعداد اتاق‌ها
- تعداد اتاق خواب‌ها
- میران آلودگی منطقه
- صاحب خانه‌ها
- میزان درآمد
- قیمت بنا
- نزدیکی به اقیانوس

گفتیم ما می‌خواهیم یک شبکه رگرسیون با mlp و تنسورفلو بسازیم. که با استفاده از ویژگی‌هایی که در بالا گفتیم، مقدار قیمت بنا را تخمین بزنند. برای این کار ابتدا باید این پایگاه داده را دانلود و فراخوانی کنیم .کتابخانه معروف یادگیری ماشین یعنی Scikit-Learnامکان دانلود این پایگاه داده را فراهم آورده است. برای فراخوانی داده با Scikit-Learn، ابتدا باید این کتابخانه را import کنیم. ما کُل کتابخانه را فراخوانی نمی‌کنیم. بلکه فقط ابزارهایی از کتابخانه که به آن‌ها نیاز داریم را فراخوانی می‌کنیم:

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow import keras
```

پس از import کردنScikit learn ، نوبت به فراخوانی داده می‌رسد. ابتدا داده‌ها را فراخوانی می‌کنیم:

```
houses = fetch_california_housing()
xTrain, xTest, yTrain, yTest = train_test_split(houses.data, houses.target)
xTrain, xValid, yTrain, yValid = train_test_split(xTrain, yTrain)
print(xTrain.shape, yTrain.dtype)
```

مشاهده می‌کنیده که در پایگاه دادهCalifornia Housing Prices ، داده‌های آموزش ، اعتبارسنجی و تست جدا نشده‌اند .پس همین اول با کمک دستورtrain_test_split ، داده‌ها را به train و تست تقسیم می‌کنیم .سپس داده‌های trainرا باز هم با کمک دستور train_test_split به train و validation تقسیم می‌کنیم .

نکته دستور train_test_split یک ورودی اضافی نیز قبول می‌کند. این ورودی اضافی مشخص می‌کند که چند درصد از داده‌ها به آموزش و چند درصد دیگر به ارزیابی اختصاص داده شود. مثلا اگر بخواهید ۵۰ درصد از داده‌ها را به ارزیابی اختصاص دهید کافی است بنویسید . train_test_split(x_train, y_train, test_size=0.5) :مقدار پیش‌فرض برای**test_size** ، مقدار ۰.۲۵ است .یعنی اگر test_size اصلا در ورودی نباشد، ۲۵ درصد از داده‌ها برای ارزیابی جدا خواهند شد.

اگر داده‌ها را print کنید، متوجه خواهید شد که ویژگی‌ها رنج‌های متفاوتی دارند. بنابراین باید مقدار ویژگی‌ها را استاندارد کنیم. مقادیر را با استفاده از Scikit-Learnبه میانگین صفر و انحراف معیار یک استاندارد می‌کنیم:

```
scaler = StandardScaler()
xTrain = scaler.fit_transform(xTrain)
xTest = scaler.transform(xTest)
xValid = scaler.transform(xValid)
#print(xTrain)
#print(xTest)
```

مشاهده می‌کنید که ما به کمک توابع StandardScaler و fit_transform از کتابخانه Scikit-Learnتوانستیم مقادیر را به میانگین صفر و انحراف معیار یک استاندارد کنیم .حتما مقادیر داده‌ها را قبل و بعد از استانداردسازی print کنید. و ببینید که چه تغییراتی در داده‌ها رخ می‌دهد.

ساختن شبکه mlp برای رگرسیون

بعد از import داده‌ها و استانداردسازی، نوبت به تعریف شبکه می‌رسد. فرآیند تعریف شبکه با استفاده از model.sequentialانجام خواهد شد .در شبکه رگرسیون، لایه خروجی هیچ تابع فعالسازی نخواهد داشت. زیرا ما می‌خواهیم که شبکه یک عدد را تخمین بزند .همچنین در مسئله رگرسیون از تابع اتلاف MSE استفاده خواهیم کرد. این تابع اتلاف میزان فاصله از مقدار واقعی را به ما نشان می‌دهد. شبکه رگرسیون را به شکل زیر تعریف می‌کنیم:

```
from numpy.ma.core import shape
model = keras.models.Sequential([keras.layers.Dense(30, activation="relu" , input_shape = xTrain.shape[1:]),keras.layers.Dense(1)])
```

همان‌طور که مشاهده می‌کنید، شبکه، یک لایه fully connected با ۳۰ نورون دارد .این لایه با دستور keras.layers.Denseساخته شده است. که در آن ورودی اول، تعداد نورون‌ها را نشان می‌دهد .ورودی دوم نشان می‌دهد که تابع فعالسازی ReLU برای این لایه انتخاب شده است .ورودی سوم اما ابعاد ورودی لایه را نشان می‌دهد . متغیر x_train تعداد ۱۱۶۱۰ سطر و ۸ ستون دارد. در واقع ۸ ستون از داده داریم و باید این ۸ ستون را به شبکه بدهیم. عبارت input_shape=x_train.shape[1:]می‌گوید که ابعاد ورودی ۸ است! درست همان چیزی که می‌خواستیم. در نهایت لایه خروجی هم یک نورون دارد.

قدم بعدی نوبت به compile کردن شبکه است. یعنی باید تعیین کنیم که شبکه چه تابع اتلافی دارد .برای این شبکه از MSEاستفاده خواهیم کرد. همچنین باید تعیین شود الگوریتم بهینه‌سازی آن چیست .برای این شبکه از گرادیان کاهشی استفاده می‌کنیم. در کلاس‌بندی معیار ارزیابی را هم مشخص کردیم .وقتی بحث رگرسیون می‌شود، معیار ارزیابی را همان فاصله می‌توان در نظر گرفت .در اینجا ما معیار ارزیابی تعریف نمی‌کنیم. زیرا تابع اتلاف یعنی همان MSE، معیار ارزیابی نیز هست. برای compile کردن شبکه کد زیر را بنویسید:

```
model.compile(loss = "mean_squared_error", optimizer = "sgd")
```

آموزش شبکه رگرسیون

تا اینجا توانستیم یک شبکه رگرسیون با mlp و تنسورفلو ۲ بسازیم و کامپایل کنیم. الان نوبت به آموزش دادن شبکه است. برای آموزش شبکه، کد زیر را بنویسید:

```
history = model.fit(xTrain, yTrain, epochs = 20, validation_data=(xValid, yValid))
```

مشاهده می‌کنید دستوری که برای آموزش شبکه نوشتیم، model.fitاست .ابتدا داده‌های آموزشی و برچسب آن‌ها را به عنوان ورودی به model.fit می‌دهیم .سپس با کمک epochs=20 مشخص می‌کنیم که تعداد تکرارها ۲۰ تا باشد .در نهایت هم داده‌های اعتبارسنجی را به صورت validation_data=(X_valid, y_valid) به دستور model.fit می‌دهیم

رسم نمودار loss

برای رسم نمودار اتلاف کافی است بنویسید:

```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```

ارزیابی شبکه رگرسیون

در این قسمت می‌خواهیم ببینیم شبکه روی داده‌های تست چه عملکردی دارد .برای این کار کافی است از دستور model.evaluateاستفاده کنید:

```
evaluateModel = model.evaluate(xTest, yTest)
```

با اجرای کد بالا مقدار اتلاف **MSE** برای داده‌های به دست خواهد آمد. **(حسین مهدوی فر – ۰۹۳۸۳۷۳۹۶۰۴)**