دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

استاد درس: دکتر فاطمه شاکری

طراح پروژه: امید سقط چیان

زمستان ۱۴۰۲

پیاده سازی MLP از پایه

درس مباحثی در علوم کامپیوتر

نحوه پیاده سازی

در این پروژه از شما انتظار داریم یک شبکه عصبی MLP را از پایه پیاده سازی کنید. در این پروژه شما مجاز نیستید از لایبری numpy استفاده کنید. یک دیتاست فرضی روی شبکه خود در نظر بگیرید و سعی کنید خطا را روی آن کاهش دهید. y ها را برای سادگی اسکالر در نظر بگیرید. در ادامه به صورت خیلی کلی ساختار کد به شما نشان داده خواهد شد.

```
class MLP:

    def __init__(self, input_size, layer_sizes):
        pass

    def forward(self, x):
        pass

    def __call__(self, x):
        pass

    def parameters(self):
        pass
```

این کلاس ۳ متود اصلی دارد که انتظار داریم با همین ساختار شما هم پیاده سازی خود را انجام دهید. متود forward محاسبات را روی ورودی انجام میدهد و خروجی را برمیگرداند. متود ___call___ تنها متود forward را صدا میزند. در اصل ما دوست داریم اینگونه model(x) ورودی را به مدل بدهیم. همچنین متود parameters تمامی وزن های شبکه را در لیست برمیگرداند. در نهایت ما بدین صورت شبکه خودمان را میسازیم.

```
input_size = 3  # Number of features in the input layer
layer_sizes = [2, 3, 1]  # Number of neurons in each hidden and output layer
model = MLP(input_size, layer_sizes)
```

کلاس MLP خودش شامل چندین لایه یا همان Layer میشود که دقیقا همان لایه های شبکه عصبی ما هستند. هر لایه باید بداند که دیتای ورودی چه ابعادی دارد و به چه ابعادی باید تبدیل شود. در اینجا ما ورودی هایمان همگی بردار هستند. هر Layer شامل چندین Neuron هست. مثلا اگر قرار باشد در یک لایه از شبکه عصبی یک بردار ۷ تایی دریافت کنیم (سایز ورودی) و خروجی یک بردار ۴ تایی داشته باشیم، باید در این لایه ۴ نورون بسازیم که هر نورون ۷ تا وزن و ۱ بایاس داشته باشد. جزییات و ساختار لایه ها و نورون ها را متناسب با توضیحات گفته شده پیاده سازی کنید.

پس از درست کردن شبکه ای که ساختیم باید فرایند یادگیری را بدین صورت تعریف کنیم:

```
n_epochs = 20

for _ in range(n_epochs):
    # Forward pass: Compute predictions for the entire dataset
    y_hats = ...

    # Compute the loss
    loss = ...

    # Zero the gradients to prevent accumulation from previous iterations
    optim.zero_grad()

    # Backward pass: Compute the gradient of the loss function with respect to model parameters
    loss.backward()

    # Update the model parameters using the optimizer
    optim.step()
```

- **محاسبه پیش بینی های مدل:** به ازای هر ورودی یا همان x ما مقدار خروجی توسط مدل یا همان y_hat را بدست می آوریم.
- **محاسبه خطا:** در اینجا شما باید خطای پیش بینی های خود را حساب کنید. برای راحتی کار تابع خطا را MSE در نظر بگیرید.
- **صفر کردن مشتق متغیر های درون شبکه:** در اینجا شما باید مشتق تمامی متغیرهای شبکه اتان را صفر کنید. پس از پیاده سازی قسمت مشتق گیری اتوماتیک دلیل این کار را متوجه خواهید شد.
- **محاسبه مشتق:** سخت ترین قسمت این پروژه پیاده سازی این قسمت میباشد. وقتی این تابع فراخوانی میشود باید مشتق loss بر اساس تمامی وزن ها و بایاس های شبکه محاسبه شود. برای پیاده سازی این قسمت شما باید از Automatic differentiation یا به اختصار diff auto استفاده کنید.
- **بهبود وزن های شبکه:** در این جا optimizer با توجه به داشتن مشتق loss نسبت به تمامی وزن ها و بایاس ها هر دو را در جهتی بروزرسانی میکند که خطای بعدی ما کمتر شود

مشتق گیری اتوماتیک

اینگونه مشتق گرفتن دو روش اصلی به نام forward mode و reverse mode دارد. در پایتورچ از روش دوم استفاده میشود و ماهم میخواهیم از روش دوم در این پروژه استفاده کنیم. برای یادگیری این موضوع تنها کافیست روی این لینک کلیک کنید و

تنها قسمت IV. An Example for Intuition را تا آخر مرحله شش بخوانید. (پس از خواندن منبع معرفی شده به خواندن توضیحات ادامه دهید)

در اصل شما باید داده ساختاری را در نظر بگیرید تا بتوانید گراف محاسباتی را تشکیل دهید و با صدا کردن تابع backward روی خروجی شبکه ، مشتق خروجی را نسبت به تمامی وزن ها و بایاس های شبکه حساب کنید. (در اینجا شبکه ما تنها یک خروجی دارد)

بهینه کننده

همانند پروژه اول optimzer باید به وزن های شبکه دسترسی داشته باشد و بتواند آن ها را این بار با توجه به مشتق شان و مقدار پارامتر lr یا همان learning_rate اپدیت کند. کارکرد متود step همانند متود update در پروژه قبل میباشد. همچنین متودی دیگری هم اضافه شده است به نام zero_grad که عملکرد آن توضیح داده شد.

```python
class Optimizer:

    def __init__(self, parameters, lr):
        # Initialize the optimizer with the model's parameters and the learning rate
        pass

    def zero_grad(self):
        # Reset the gradients of all parameters to zero
        pass

    def step(self):
        # Update the model's parameters based on the gradients and learning rate
        pass
```

معیارهای ارزیابی - نکات ارسال

- از شما انتظار میرود تمامی قسمت ها را مطابق با توضیحات گفته شده پیاده سازی کنید. پیاده سازی که در چهارچوب گفته شده نباشد نمره ای نخواهد داشت.
- برای قسمت ها و اجزای مختلف پروژه خود به **زبان فارسی** داکیومنت بنویسید و عملکرد آن را کامل و دقیق شرح دهید.
- نمره شما از این پروژه پس از دادن ارائه در عددی بین ۰ تا ۱ بسته به میزان تسلط شما ضرب میشود.
- فایل ارسالی خود را به صورت فشرده و با فرمت نامگذاری project_2_idGroup.zip نامگذاری کنید. فایل فشرده

شامل یک فایل ژوپیتر نوت بوک [1] و لینک گوگل کولب [2] میباشد. تنها ارسال لینک گوگل گولب **پذیرفته نیست**.

- همچنین می توانید گزارش تمرین خود را به صورت یک فایل ورد [3] ارسال کنید. در اینصورت باید فایل کامل کد خود را همراه با فایل ورد ارسال کنید و همچنین در توضیح هر بخش، عکسی از قطعه کدی که توضیح می دهید را در فایل ورد قرار دهید. توضیحات فایل ورد شما باید به **زبان فارسی** باشد.

مهلت تمرین

برای حل تمرین سری ۲ تا روز **۲۴ فروردین** زمان دارید. در نظر داشته باشید این پروژه مانند پروژه اول ساده نیست. لطفا برنامه ریزی مناسبی برای انجام آن در نظر بگیرید و به لحظات آخر موکول نکنید.

[1] Jupyter Notebook
[2] Google Colab
[3] Word