تمرین کوتاه 5 شبکه های عصبی – مازندرانیان - 830402066

سوال 1) تجمع گرادیان روی mini batch های متعدد برای بروزرسانی وزن های مدل در مواقعی که کارت گرافیک محدود داریم

وقتی با محدودیت حافظه گرافیکی روبرو هستیم، استفاده از مینی‌بچ‌های بزرگ در آموزش مدل‌های یادگیری عمیق می‌تواند چالش‌برانگیز باشد. تجمع گرادیان روشی است که به ما اجازه می‌دهد تا از مینی‌بچ‌های کوچک‌تر استفاده کنیم و در عین حال به نتایج مشابهی با مینی‌بچ‌های بزرگ برسیم.

نحوه کار:

1. **تعیین اندازه مینی‌بچ موثر و تعداد مراحل تجمع :**
   - اندازه مینی‌بچ موثر: اندازه‌ای که می‌خواهیم به آن دست یابیم (مثلاً 128).
   - تعداد مراحل تجمع: تعداد دفعاتی که گرادیان‌ها جمع می‌شوند قبل از به‌روزرسانی وزن‌ها (مثلاً 4).
   - اندازه مینی‌بچ واقعی: اندازه مینی‌بچ موثر تقسیم بر تعداد مراحل تجمع.(128 / 4 = 32)
2. **ایجاد بافر برای جمع‌آوری گرادیان‌ها :**
   - یک بافر ایجاد می‌کنیم که در آن گرادیان‌های هر مینی‌بچ را جمع می‌کنیم.
3. **تکرار روی مینی‌بچ‌ها :**
   - برای هر مینی‌بچ :
     - داده‌های مینی‌بچ را بارگذاری می‌کنیم.
     - پاس رو به جلو را انجام می‌دهیم و خطا را محاسبه می‌کنیم.
     - گرادیان‌ها را محاسبه می‌کنیم.
     - گرادیان‌های محاسبه‌شده را به بافر اضافه می‌کنیم.
4. **به‌روزرسانی وزن‌ها :**
   - پس از پردازش تعداد مشخصی مینی‌بچ (برابر با تعداد مراحل تجمع) :
     - گرادیان‌های جمع‌شده را بر تعداد مراحل تجمع تقسیم می‌کنیم تا میانگین‌گیری شود.
     - وزن‌های مدل را با استفاده از گرادیان‌های میانگین‌شده به‌روزرسانی می‌کنیم.
     - بافر گرادیان‌ها را صفر می‌کنیم.

مزایای تجمع گرادیان:

- **کاهش مصرف حافظه :**با استفاده از مینی‌بچ‌های کوچک‌تر، فشار کمتری به حافظه گرافیکی وارد می‌شود.
- **بهبود پایداری آموزش :**تجمع گرادیان‌ها می‌تواند به پایداری آموزش کمک کند، خصوصاً در شبکه‌های عصبی عمیق.
- **امکان استفاده از مینی‌بچ‌های بزرگ‌تر موثر :**با تجمع گرادیان‌ها، می‌توان به نتایج مشابهی با مینی‌بچ‌های بزرگ‌تر دست یافت.

مثال از کتابخانه ی Pytorch (منبع: ChatGPT):

```
import torch

effective_batch_size = 128
accumulation_steps = 4
actual_batch_size = effective_batch_size // accumulation_steps

model = YourModel()
optimizer = torch.optim.Adam(model.parameters())

accumulated_grads = None

for epoch in range(num_epochs):
    for i, (inputs, labels) in enumerate(dataloader):
        outputs = model(inputs)
        loss = loss_fn(outputs, labels)

        loss.backward()

        if accumulated_grads is None:
            accumulated_grads = [g.clone() for g in
model.parameters()]
        else:
            for g, acc_g in zip(model.parameters(), accumulated_grads):
                acc_g += g.grad

        if (i + 1) % accumulation_steps == 0:
            for g in model.parameters():
                g.grad = g.grad / accumulation_steps
            optimizer.step()
            optimizer.zero_grad()
            accumulated_grads = None
```

سوال 2) مقایسه ی تکنیک های batch Normalization

نرمال‌سازی دسته‌ای (Batch Normalization) و انواع آن، به عنوان تکنیک‌های اساسی در یادگیری عمیق، به بهبود پایداری آموزش و تعمیم‌پذیری مدل‌ها کمک شایانی کرده‌اند. در این بخش، به مقایسه رایج‌ترین تکنیک‌های نرمال‌سازی می‌پردازیم:

.نرمال‌سازی دسته‌ای استاندارد(Batch Normalization)

مزایا:

- **پایداری بیشتر در آموزش :**با کاهش شیفت کوواریانس داخلی، روند آموزش را آسان‌تر می‌کند.
- **اثر منظم‌سازی :**به عنوان نوعی منظم‌سازی عمل کرده و از بیش‌برازسازی جلوگیری می‌کند.
- **همگرایی سریع‌تر :**می‌تواند به همگرایی سریع‌تر در طول آموزش منجر شود.

محدودیت‌ها:

- **حساس به اندازه‌های کوچک دسته :**با اندازه‌های کوچک دسته، عملکرد آن ممکن است کاهش یابد.
- **برای شبکه‌های عصبی بازگشتی (RNN) مناسب نیست :**در مدیریت ماهیت ترتیبیRNN ها با مشکل مواجه می‌شود.

.نرمال‌سازی لایه‌ای(Layer Normalization)

مزایا:

- **مستقل از اندازه دسته :**با اندازه‌های کوچک دسته یا حتی نمونه‌های تکی نیز به خوبی عمل می‌کند.
- **برایRNN ها مناسب است :**به دلیل استقلال از اندازه دسته، می‌تواند به طور موثر درRNN ها استفاده شود.

محدودیت‌ها:

- **برای شبکه‌های عصبی کانولوشنی (CNN) کمتر موثر است :**ممکن است اطلاعات مکانی را به خوبی BN درک نکند.

.نرمال‌سازی نمونه‌ای(Instance Normalization)

مزایا:

- **حساس به انتقال سبک :**می‌تواند برای انتقال سبک با نرمال‌سازی در هر نمونه استفاده شود.
- **مستقل از اندازه دسته و لایه :**در سناریوهایی که اطلاعات اندازه دسته یا لایه مرتبط نیست، به خوبی عمل می‌کند.

محدودیت‌ها:

- **برای وظایف عمومی کمتر موثر است :**ممکن است به اندازه BN برای طبقه‌بندی یا رگرسیون عمومی موثر نباشد.

.نرمال‌سازی گروهی(Group Normalization)

مزایا:

- **بین BN و LN قرار دارد :**ترکیبی از جنبه‌های BN و LN است و تعادلی بین نرمال‌سازی مبتنی بر دسته و لایه ایجاد می‌کند.
- **با اندازه‌های کوچک دسته به خوبی عمل می‌کند :**نسبت به BN در برابر اندازه‌های کوچک دسته مقاوم‌تر است.

محدودیت‌ها:

- **تنظیم ابرپارامتر :**نیاز به تنظیم تعداد گروه‌ها دارد که می‌تواند چالش‌برانگیز باشد.

انتخاب تکنیک مناسب

بهترین تکنیک به وظیفه و معماری خاص بستگی دارد:

- **نرمال‌سازی دسته‌ای استاندارد :**معمولاً نقطه شروع خوبی برای بسیاری از وظایف است.
- **نرمال‌سازی لایه‌ای :**برایRNN ها یا زمانی که با اندازه‌های کوچک دسته سروکار دارید، در نظر بگیرید.
- **نرمال‌سازی نمونه‌ای :**برای انتقال سبک یا وظایفی که نرمال‌سازی در سطح نمونه مفید است، بررسی کنید.
- **نرمال‌سازی گروهی :**اگر به تعادل بین نرمال‌سازی مبتنی بر دسته و لایه نیاز دارید، با آن آزمایش کنید.