مشروع نهاية الفصل 5

مقرر Deep Learning — Optimization — SGD و Momentum و Adam

الهدف: تقارن ٣ optimizers من الصفر على نفس مسألة الـ softmax regression.

الأدوات: Python مع numpy وscikit-learn، وتقدر تشغّل الكود في Google Colab المجاني.

الداتا: sklearn.datasets.load_digits (١٧٩٧ صورة ٨×٨، ١٠ أرقام). مصدرها UCI. مدمجة في scikit-learn.

الأسئلة العملية

1 جهّز (Prepare)

- نفس إعداد الفصل ٢ (softmax regression).

2 SGD (SGD)

- ١٠٠ خطوة بـ lr = 0.1.

3 Momentum (Momentum)

- v = 0.9 v + g ثم W -= lr v.

4 Adam (Adam)

- m و v بـ beta1 = 0.9 و beta2 = 0.999، مع تصحيح الانحياز.

5 قارن (Compare)

- اطبع الـ loss النهائي لكل واحد بنفس عدد الخطوات.

بونس

- بونس: جرّب lr أكبر وأصغر.
- بونس: جرّب mini-batch حجمه 64.

أسئلة تفكير (اكتب إجابة في سطرين)

1. إيه اللي بيعمله الـ momentum؟
2. ليه Adam بيحتاج تصحيح الانحياز في الأول؟

ماذا تسلّم؟

- ملف الكود (أو notebook) بيشتغل من الأول للآخر.
- فقرة قصيرة (٥ أسطر) بتلخّص النتيجة وإيه اللي اتعلمته.

الحل النموذجي

جرّب بنفسك الأول، وبعدين قارن. الأرقام الحقيقية في النتيجة تحت.

الكود

```
# dataset ships inside scikit-learn (no download needed)
import numpy as np
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

X, y = load_digits(return_X_y=True)
X = X / 16.0
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=0)
Ytr = np.eye(10)[ytr]

def softmax(Z):
    E = np.exp(Z - Z.max(1, keepdims=True))
    return E / E.sum(1, keepdims=True)

def grad_loss(W):
    P = softmax(Xtr @ W)
    return Xtr.T @ (P - Ytr) / len(ytr), -np.log(P[np.arange(len(ytr)), ytr]).mean()

def run(name, steps=100, lr=0.1):
    W = np.zeros((64, 10)); v = np.zeros_like(W); m = np.zeros_like(W)
    for t in range(1, steps + 1):
        g, loss = grad_loss(W)
        if name == 'sgd':
            W -= lr * g
        elif name == 'momentum':
            v = 0.9 * v + g; W -= lr * v
        else:
            m = 0.9 * m + 0.1 * g; v = 0.999 * v + 0.001 * g * g
            W -= lr * (m / (1 - 0.9 ** t)) / (np.sqrt(v / (1 - 0.999 ** t)) + 1e-8)
    return grad_loss(W)[1]

for name in ('sgd', 'momentum', 'adam'):
    print(name.ljust(9), 'final loss', round(run(name), 4))
```

النتيجة

```
sgd       final loss 1.1117
momentum  final loss 0.2545
adam      final loss 0.0481
```

ملاحظات على الإجابات

- الـ momentum بيجمّع اتجاه الـ gradients السابقة فبيمشي أسرع في الاتجاه الثابت.
- في أول الخطوات m و v قريبين من صفر فبنصحّحهم بالقسمة على (1 - beta^t). الأرقام النهائية فوق هي اللي عندنا، مش قاعدة عامة لكل مسألة.

Credits: topic follows the matching chapter of Dive into Deep Learning (d2l.ai, CC BY-SA 4.0). Questions, code and explanations are original.