مشروع نهاية الفصل 3

مقرر Deep Learning — Multilayer Perceptrons — شبكة MLP بـ backprop من الصفر

الهدف: تبني شبكة 64-32-10 بـ ReLU وتحسب الـ gradients بإيدك بالـ backpropagation.
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)

- قسّم digits ٨٠/٢٠ وابدأ أوزان عشوائية صغيرة.

2 Forward (Forward)

- احسب $Z1 = X\,W1 + b1$، $H = \mathrm{ReLU}(Z1)$، ثم $\mathrm{softmax}(H\,W2 + b2)$.

3 Backward (Backward)

- احسب gradient الخرج $(P - Y)$، وارجع للخلف للطبقة الأولى مع قناع الـ ReLU.

4 درّب (Train)

- ٣٠٠ خطوة بـ $lr = 0.5$، واطبع الـ loss كل ١٠٠ خطوة.

5 قارن (Compare)

- قارن دقة الـ test بالـ Softmax regression من الفصل ٢.

بونس

- بونس: جرّب ٦٤ unit وقارن.
- بونس: اعمل gradient check بفرق محدود على وزن واحد.

أسئلة تفكير (اكتب إجابة في سطرين)

1. ليه من غير activation الشبكة تفضل خطية؟
2. إيه دور قناع الـ ReLU في الـ backward؟

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]

rng = np.random.default_rng(0)
W1, b1 = rng.normal(0, 0.2, (64, 32)), np.zeros(32)
W2, b2 = rng.normal(0, 0.2, (32, 10)), np.zeros(10)

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

def forward(A):
    Hd = np.maximum(0, A @ W1 + b1)
    return Hd, softmax(Hd @ W2 + b2)

for step in range(1, 301):
    Hd, P = forward(Xtr)
    G2 = (P - Ytr) / len(ytr)
    G1 = (G2 @ W2.T) * (Hd > 0)
    W2 -= 0.5 * Hd.T @ G2; b2 -= 0.5 * G2.sum(0)
    W1 -= 0.5 * Xtr.T @ G1; b1 -= 0.5 * G1.sum(0)
    if step % 100 == 0:
        print('step', step, 'loss', round(-np.log(P[np.arange(len(ytr)), ytr]).mean(),
4))
print('test acc', round((forward(Xte)[1].argmax(1) == yte).mean(), 3))
```

النتيجة

```
step 100 loss 0.1617
step 200 loss 0.0954
step 300 loss 0.0686
test acc 0.958
```

ملاحظات على الإجابات

- من غير activation كل الطبقات بتتجمّع في ضرب مصفوفات واحد، يعني نموذج خطي.
- القناع (Hd > 0) بيمرّر الـ gradient بس للـ units اللي كانت شغالة.

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