مشروع نهاية الفصل 1

مقرر Introduction — Deep Learning — الخلية العصبية الواحدة

الهدف: تبني خلية عصبية واحدة (sigmoid) من الصفر بـ numpy وتدرّبها بالـ gradient descent وتقارنها بـ LogisticRegression.

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

الداتا: sklearn.datasets.load_breast_cancer (٥٦٩ عينة، ٣٠ خاصية). مصدرها UCI. مدمجة في scikit-learn.

الأسئلة العملية

1 جهّز (Prepare)

- حمّل الداتا، اعمل standardize، وقسّمها ٧٠/٣٠ بـ random_state=0.

2 الخلية (The neuron)

- اكتب $p = \text{sigmoid}(X\,w + b)$ والـ loss (cross-entropy).

3 التدريب (Training)

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

4 قيّم (Evaluate)

- اطبع دقة الـ test وقارنها بـ LogisticRegression.

بونس

- بونس: جرّب lr = 5 واكتب إيه اللي بيحصل.
- بونس: اطبع أكبر ٣ أوزان بالقيمة المطلقة.

أسئلة تفكير (اكتب إجابة في سطرين)

1. ليه بنستخدم sigmoid في الخرج؟
2. ليه الـ standardize مهم للـ gradient descent؟

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_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

X, y = load_breast_cancer(return_X_y=True)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3, random_state=0)
sc = StandardScaler().fit(Xtr); Xtr, Xte = sc.transform(Xtr), sc.transform(Xte)

sigmoid = lambda a: 1 / (1 + np.exp(-a))
w, b = np.zeros(Xtr.shape[1]), 0.0
for step in range(1, 301):
    p = sigmoid(Xtr @ w + b)
    w -= 0.5 * Xtr.T @ (p - ytr) / len(ytr)
    b -= 0.5 * (p - ytr).mean()
    if step % 100 == 0:
        loss = -np.mean(ytr * np.log(p + 1e-9) + (1 - ytr) * np.log(1 - p + 1e-9))
        print('step', step, 'loss', round(loss, 4))
print('neuron test acc :', round(((sigmoid(Xte @ w + b) > 0.5) == yte).mean(), 3))
print('sklearn test acc:', round(LogisticRegression(max_iter=2000).fit(Xtr,
ytr).score(Xte, yte), 3))
```

النتيجة

```
step 100 loss 0.0705
step 200 loss 0.0618
step 300 loss 0.0574
neuron test acc : 0.977
sklearn test acc: 0.977
```

ملاحظات على الإجابات

- الخلية الواحدة هي نفسها Logistic Regression: وزن لكل feature وانحياز وsigmoid.
- الـ sigmoid بتحوّل الخرج لاحتمال بين صفر وواحد.
- الـ standardize بيخلّي كل الـ features بنفس المقياس فالـ gradient descent بيمشي بثبات.

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