وزارة التعليم العالي والبحث العلمي

جامعة أم البواقي

كلية العلوم الاقتصادية والعلوم التجارية وعلوم التسيير

مقياس: تطبيقات التعلم العميق قسم: العلوم الاقتصادية

سلسلة التمارين رقم: 04

التمرين الأول:

خوارزمية تعلم بيرسبترون

1. ابدأ متجه الوزن w بقيم عشوائية صغيرة.
2. حتى يتقارب البيرسيبترون:
   أ. قم بعمل حلقة على كل متجه سمة x_j وقم بتسمية الفئة الفعلية d_j في مجموعة التدريب.
   ب. خذ x وقم بتمريرها عبر الشبكة وحساب قيمة الإخراج:

$$y_j = f(w(t).x_j)$$

   ج. قم بتحديث الأوزان w:

$$w: w_i(t+1) = w_i(t) + \alpha(d_j - y_j)x_{j,i}$$

ET (AND)		
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

OU (OR)		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

XOU (XOR)		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

```
# import the necessary packages
import numpy as np
class Perceptron:
  def __init__(self, N, alpha=0.1):
    # initialize the weight matrix and store the learning rate
    self.W = np.random.randn(N + 1) / np.sqrt(N)
    self.alpha = alpha

  def step(self, x):
    return 1 if x > 0 else 0

  def fit(self, X, y, epochs=10):
    X = np.c_[X, np.ones((X.shape[0]))]
    # loop over the desired number of epochs
    for epoch in np.arange(0, epochs):
      # loop over each individual data point
      for (x, target) in zip(X, y):
        p = self.step(np.dot(x, self.W))
        if p != target:
          error = p - target
          self.W += -self.alpha * error * x

  def predict(self, X, addBias=True):
    X = np.atleast_2d(X)
    if addBias:
      X = np.c_[X, np.ones((X.shape[0]))]
    return self.step(np.dot(X, self.W))
```

بعد تدريب Perceptron الخاص بنا، نحتاج إلى تقييمه على البيانات للتأكد من أنه قد تعلم بالفعل دالة OR:

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [1]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
print("[INFO] testing perceptron...")
for (x, target) in zip(X, y):
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(x, target[0], pred))
```

بعد تنفيذ الكود أعلاه يتم عرض المخرجات على النحو التالي:

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=0
[INFO] data=[0 1], ground-truth=1, pred=1
[INFO] data=[1 0], ground-truth=1, pred=1
[INFO] data=[1 1], ground-truth=1, pred=1
```

كما يتضح ، كان Perceptron لدينا قادرًا على تعلم عملية OR ،

الآن دعنا ننتقل إلى دالة ، ANDأدخل الكود التالي:

[42]
✓ 0s

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [0], [0], [1]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
print("[INFO] testing perceptron...")
for (x, target) in zip(X, y):
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(x, target[0], pred))
```

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=0
[INFO] data=[0 1], ground-truth=0, pred=0
[INFO] data=[1 0], ground-truth=0, pred=0
[INFO] data=[1 1], ground-truth=1, pred=1
```

أخيرًا ، دعنا نلقي نظرة على وظيفة XORغير الخطية باستخدام perceptron.أدخل الكود في الأسفل:

[43]
✓ 0s

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
print("[INFO] testing perceptron...")
for (x, target) in zip(X, y):
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(x, target[0], pred))
```

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=1
[INFO] data=[0 1], ground-truth=1, pred=0
[INFO] data=[1 0], ground-truth=1, pred=0
[INFO] data=[1 1], ground-truth=0, pred=0
```

لا يهم عدد المرات التي تجري فيها هذه التجربة بمعدلات تعلم مختلفة أو طرق تهيئة مختلفة ،

حيث لا يمكنك أبدًا نمذجة دالة XOR باستخدام Perceptronأحادي الطبقة.

بدلاً من ذلك، ما **نحتاجه هو المزيد من الطبقات ذات دوال التنشيط غير الخطية**