

# تمرین اول یادگیری عمیق

97521423

محمدعلی فراهت

## سوال (1)

- **Al winter** : به طور کلی به زمان‌هایی که در آن، دنیا از هوش مصنوعی ناامید و سرمایه‌گذاری‌ها بر روی آن کاهش زیادی داشته، **Al winter** می‌گویند. تابحال دو بار در تاریخ این اتفاق افتاده و برخی معتقدند که در حال حاضر هم در سومین **Al winter** هستیم. عملاً وقتی از هوش مصنوعی انتظارات زیادی دارند و تصور می‌کنند که در آینده کوتاه مدت این انتظارات برآورده می‌شود و در واقعیت این اتفاق نمی‌افتد، از آن ناامید میشوند. برای مثال در سال 1960 که هوش مصنوعی تازه شناخته شده بود و در حال پیشرفت بود، همه فکر میکردند که تا آینده‌ای نزدیک هوش مصنوعی‌ای نزدیک هوش بشر تولید میشود. اما چند سال بعد که این اتفاق نیوفتاد، امیدها بر باد رفت و وارد اولین **Al winter** شدیم. اتفاقی مشابه در سال 1980 برای **expert systems** افتاد و این دومین **Al winter** تاریخ بود.
- **Backpropagation** : یک الگوریتم برای یادگیری شبکه‌های عصبی چند لایه است که طی تحقیقات همزمان و موازی محققان توسعه یافت. این الگوریتم از **loss function** در لایه آخر استفاده میکند و لایه به لایه و به ترتیب وزن‌های شبکه را آپدیت میکند تا برای نمونه‌های بعدی **loss function** کمتر شود و یادگیری کامل‌تر شود. نام دیگر این الگوریتم **reverse-mode differentiation** است.
- **Objective function** : نام دیگر و آشناتر این فانکشن **loss function** است و درواقع میزان خطای شبکه عصبی را نشان میدهد. کار آن مقایسه خروجی شبکه و جواب درست است که از قبل میدانیم. تمام تلاش ما این است که آن را کاهش دهیم و به سمت صفر ببریم.

- **Kernel methods** : گروهی از الگوریتم های طبقه بندی است که با نام support vector machine (SVM) شناخته میشود. SVM ها مرزهای جداسازی کلاس ها را پیدا میکنند. SVM ها در دیتا های بزرگ خوب عمل نمیکنند چون پیدا کردن ویژگی های آن ها به صورت دستی غیر ممکن است و در این روش kernel function ها را باید به صورت دستی پیدا کنیم.
- **4D tensors vs. 4-dimensional vector** : فرق آن ها این است که بردار (vector) چهاربعدی فقط 4 عدد ورودی دارد دارد که روی یک محور هستند ، مانند array([1,2,3,4]) ، اما tensor چهاربعدی دارای 4 محور است و هر کدام هم میتوانند چند بعدی باشند. پس قابل مقایسه با vector نیست.
- **Element-wise product vs. Tensor product** : در element-wise product ، عملیات ها به صورت جداگانه و روی هر entry جدا جدا انجام می شوند. این عملیات ها میتوانند به صورت موازی و یا vectorized انجام شوند. ولی tensor product به صورت ماتریسی با هم انجام میشوند و به آن دات هم میگویند. هر کدام از این عملیات ها در keras و numpy و ... با علامت خاصی مثل \* یا dot استفاده میشوند.

$$P(\text{spam}) = 6/10$$

(2

$$P(\text{not spam}) = 4/10$$

$P(x_i)$	spam	not spam
$x_1$	$1/6$	$1$
$x_2$	$5/6$	$1/4$
$x_3$	$4/6$	$1/4$

$$P(x_1=1) = 5/10, \quad P(x_2=1) = 6/10, \quad P(x_3=1) = 5/10$$

$$X_1 = [1, 1, 0] \Rightarrow P(\text{spam} | x_1, x_2, x_3) = P(\text{spam}) P(x_1, x_2, x_3 | \text{spam})$$

$$= P(\text{spam}) P(x_1 | \text{spam}) P(x_2 | \text{spam}) (1 - P(x_3 | \text{spam}))$$

$$= 6/10 \times 1/6 \times 5/6 \times 2/6 = 1/36$$

$$\Rightarrow P(\overline{\text{spam}} | x_1, x_2, x_3) = P(\overline{\text{spam}}) P(x_1, x_2, x_3 | \overline{\text{spam}})$$

$$= P(\overline{\text{spam}}) P(x_1 | \overline{\text{spam}}) P(x_2 | \overline{\text{spam}}) (1 - P(x_3 | \overline{\text{spam}}))$$

$$= 4/10 \times 1 \times 1/4 \times 3/4 = 3/40$$

not spam  $\leftarrow$  not spam is less

$$X_2 = [1, 1, 1] \Rightarrow P(\text{spam} | x_1, x_2, x_3) = P(\text{spam}) P(x_1, x_2, x_3 | \text{spam})$$

$$= P(\text{spam}) P(x_1 | \text{spam}) P(x_2 | \text{spam}) P(x_3 | \text{spam})$$

$$= 6/10 \times 1/6 \times 5/6 \times 4/6 = 1/18$$

$$\Rightarrow P(\overline{\text{spam}} | x_1, x_2, x_3) = P(\overline{\text{spam}}) P(x_1, x_2, x_3 | \overline{\text{spam}})$$

$$= P(\overline{\text{spam}}) P(x_1 | \overline{\text{spam}}) P(x_2 | \overline{\text{spam}}) P(x_3 | \overline{\text{spam}})$$

$$= 4/10 \times 1 \times 1/4 \times 1/4 = 1/40$$

spam  $\leftarrow$  spam is more

### سوال 3

در این بخش مراحل خواسته شده را با عکس از محیط colab نشان خواهیم داد.

#### دانلود

```
1 from keras.datasets import cifar10
2 (x_train, y_train), (x_test, y_test) = cifar10.load_data()

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [=====] - 2s 0us/step
170508288/170498071 [=====] - 2s 0us/step
```

حالا مولفه‌های dtype و ndim و shape را برای x\_train پرینت میکنیم :

```
[2] 1 print(x_train.dtype)
      2 print(x_train.ndim)
      3 print(x_train.shape)

uint8
4
(50000, 32, 32, 3)
```

جنس ورودی ما از نوع عدد و uint8 است (int 8 بیتی) و همچنین ورودی 4 بعد (محور) دارد. خط آخر شکل ورودی را نشان میدهد که در اینجا یعنی 50 هزار تا عکس ورودی داریم که هر کدام 32 \* 32 پیکسل دارد و هر پیکسل دارای 3 مقدار برای rgb می‌باشد.

حالا مولفه‌های dtype و ndim و shape را برای y\_train پرینت میکنیم :

```
[3] 1 print(y_train.dtype)
      2 print(y_train.ndim)
      3 print(y_train.shape)

uint8
2
(50000, 1)
```

جنس ورودی مانند قبل، int است و دو محور دارد. خط آخر به این معنی است که 50 هزار تا ورودی داریم و هر کدام یک برچسب هستند ( که جواب داده های x\_train هستند)

حالا مولفه‌های dtype و ndim و shape را برای x\_test پرینت میکنیم :

```
[4] 1 print(x_test.dtype)
      2 print(x_test.ndim)
      3 print(x_test.shape)

uint8
4
(10000, 32, 32, 3)
```

این داده‌ها دقیقا مانند x\_train هستند ولی با این تفاوت که تعداد عکس‌ها 10 هزار تا است، که برای تست کردن مدل استفاده خواهند شد.

---

حالا مولفه‌های dtype و ndim و shape را برای x\_train پرینت میکنیم :

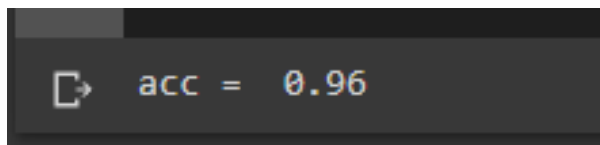
```
[5] 1 print(y_test.dtype)
      2 print(y_test.ndim)
      3 print(y_test.shape)

uint8
2
(10000, 1)
```

این داده‌ها هم درست مانند y\_train هستند و درواقع خروجی‌های مورد انتظار ما از داده‌های ورودی x\_test هستند که تعداد آن‌ها هم 10 هزار تا برچسب هست.

#### سوال 4)

برای حل این سوال ابتدا دیتا را از لینک صورت سوال دانلود کردم، سپس آنها را پارس کرده و دیتای آموزش و دیتای تست را یکی در نظر گرفتم ( طبق گفته دستیاران آموزشی) حالا برای هر گل ( در کل 3 مدل گل در این دیتاست وجود داشت) مقدار میانگین و واریانس را حساب کردم. سپس در تابع predict با استفاده از فرمول های زیر احتمال پیشین و احتمال شرطی هر کدام را محاسبه کردم و سپس از این تابع روی تست ها استفاده کردم و جواب های درست را شمردم و بر تعداد کل تست ها ( 150 تا ) تقسیم کردم تا دقت این الگوریتم را نشان دهم. دقت در پایان ، 96 درصد بود.



```
acc = 0.96
```

برای سهولت در تصحیح تمرین ، عکس کد زده شده را در صفحه بعد میگذارم:

```

1 data = {}
2 test_data = []
3
4 data['Iris-virginica'] = []
5 data['Iris-setosa'] = []
6 data['Iris-versicolor'] = []
7
8 f = open("iris.data")
9 for e in f:
10     toks = e.split(',')
11     if len(toks) == 5:
12         flower = toks[4].strip()
13         a = [float(i) for i in toks[:4]]
14         attributes = np.array(a)
15         data[flower].append(attributes)
16         test_data.append((flower, attributes))
17
18 variance = {}
19 mean = {}
20
21 variance['Iris-virginica'] = np.var(np.array(data['Iris-virginica']), axis=0)
22 variance['Iris-setosa'] = np.var(np.array(data['Iris-setosa']), axis=0)
23 variance['Iris-versicolor'] = np.var(np.array(data['Iris-versicolor']), axis=0)
24
25 mean['Iris-virginica'] = np.mean(np.array(data['Iris-virginica']), axis=0)
26 mean['Iris-setosa'] = np.mean(np.array(data['Iris-setosa']), axis=0)
27 mean['Iris-versicolor'] = np.mean(np.array(data['Iris-versicolor']), axis=0)
28
29
30 pos = {}
31
32 pos['Iris-virginica'] = len(data['Iris-virginica']) / 150
33 pos['Iris-setosa'] = len(data['Iris-setosa']) / 150
34 pos['Iris-versicolor'] = len(data['Iris-versicolor']) / 150
35
36
37 def predict(flowerr, xi):
38     cp = (1/np.sqrt(2* np.pi* variance[flowerr]))* np.exp(-1*((xi - mean[flowerr])**2)/(2*variance[flowerr]))
39     return (np.prod(cp) * pos[flowerr])
40
41 flowers = ['Iris-virginica','Iris-setosa','Iris-versicolor']
42 i = 0
43
44 for test in test_data:
45     maxx = -9999999
46     for f in flowers:
47         ps = predict(f, test[1])
48         if ps > maxx:
49             maxx = ps
50             winner = f
51     if winner == test[0]:
52         i += 1
53 print('acc = ', i/150)
54

```