بسمه تعالی

دانشکده مهندسی کامپیوتر

پاییز ۱۴۰۰

یادگیری عمیق

نام استاد: دکتر محمدی

تمرین اول

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

۱. پاسخ سوال اول

- AI winter: در تاریخ هوش مصنوعی، بازه‌های زمانی وجود دارد که سرمایه‌گذاری و علایق مردم نسبت به هوش مصنوعی کاهش یافته بود. این اطلاح را به آن دوره‌ها می‌گوییم.
- Backpropagation: الگوریتمی از یادگیری supervised است که در شبکه‌های عصبی و با استفاده از gradient decent کار می‌کند. هدف از این الگوریتم اعمال تغییرات روی وزن‌های لایه‌های مختلف شبکه عصبی بر اساس گرادیان خطای خروجی نهایی شبکه می‌باشد.
- Objective function: تابعی که رابطه آن را می‌خواهیم حداقل یا حداکثر کنیم تا با استفاده از آن سود را زیاد و ضرر را تا جای ممکن کاهش دهیم. البته معمولا محدودیت های مختلفی روی متغیرها وجود دارد و در روابط باید لحاظ شوند.
- Kernel methods: دسته‌ای از الگوریتم‌های classification هستند که در SVM بسیار کاربرد دارند. هدف از این الگوریتم‌ها این است که داده‌ها را به بعدهای بالاتر ببریم تا classification ساده‌تر شود. اساس آن‌ها این است که در فضای نمونه‌ای جدید، صرفا محاسبه‌ی فاصله‌ی بین نقاط کفایت می‌کند و تخمین زدن دقیق مقدار نیاز نیست.
- 4D tensors vs. 4-dimensional vector: این دو را نباید باهم اشتباه بگیریم. بردارهای ۴ بعدی، فقط یک محور دارند که خود این محور ۴ بعد دارد. در حالی که tensor ۴ بعدی، ۴ محور دارد و هر کدام از این محور ها می‌توانند چندین بعد داشته باشند.
- Element-wise product vs. Tensor product: این دو را نباید باهم اشتباه بگیریم. ضرب element-wise یا hadamard، همان ضرب ماتریس هاست که درواقع حاصل ضرب دو ماتریس به اندازه m*n و n*h، یک ماتریس m*h می‌شود. نماد آن در کتابخانه های مختلف numpy و keras و Theano و tensorflow، "*" است. اما ضرب tensor، که در ریاضی نمادش "." است و در numpy آن را با "dot" می شناسیم، خروجی عدد می دهد و المان های دو بردار را ضرب میکند و جمع آن ها را برمی‌گرداند.

۲. پاسخ سوال دوم

اگر هر کدام از ستون های بردار x را به ترتیب x1، x2، x3 بنامیم، داده‌های زیر را طبق ورودی سوال خواهیم داشت:

	Mean(x1)	Variance(x1)	Mean(x2)	Variance(x2)	Mean(x3)	Variance(x3)
Spam	0.17	0.1667	0.83	0.1667	0.67	0.403
Nspam	1	0	0.25	0.25	0.25	0.25

حال می توانیم طبق فرمول gaussian naïve bayes، احتمالات زیر را برای X1 = [1 1 0] محاسبه کنیم:

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

P(x1|spam)= 0.124, P(x2|spam)=0.902, P(x3|spam)=0.36, P(spam)=0.6

P(x1|Nspam)=1, P(x2|Nspam)=0.26, P(x3|Nspam)=0.706, P(Nspam)=0.4

سپس می توانیم طبق فرمول naïve bayes برای ورودی‌های مختلف احتمال اسپم بودن یا نبودن را بررسی کنیم: (/ علامت تقسیم است)

P(X1 is spam) = P(x1|spam)*P(x2|spam)*P(x3|spam)*P(spam)/evidence = 0.124 * 0.902 * 0.36 * 0.6 = 0.024

P(X1 is not spam) = P(x1|spam)*P(x2|spam)*P(x3|spam)*P(spam)/evidence = 1 * 0.26 * 0.706 * 0.4 = 0.073

پس پیشبینی میکنیم که X1، اسپم نیست.

و به طور مشابه برای X2 = [1 1 1] خواهیم داشت:

P(X2 is spam) =0.124 * 0.902 * 0.546 * 0.6 = 0.037

P(X2 is not spam) = 1 * 0.26 * 0.26 * 0.4 = 0.027

پس پیشبینی میکنیم که X2، اسپم است.

۳. پاسخ سوال سوم

پس از دریافت dataset مطابق توضیحات صورت سوال، به این ترتیب موارد خواسته شده را به دست می آوریم:

1. data type of sets:

```
[14] print("x_train data type: ", x_train.dtype)
     print("y_train data type: ", y_train.dtype)
     print("x_test data type: ", x_test.dtype)
     print("x_test data type: ", y_test.dtype)
```

```
x_train data type:  uint8
y_train data type:  uint8
x_test data type:  uint8
x_test data type:  uint8
```

که این خروجی نشان دهنده این است که تمامی دیتاهای موجود در این ۴ آرایه از نوع uint8 هستند. (چون درواقع rgb پیکسل های مختلف هستند و اعدادی ۸ بیتی بین ۰ تا ۲۵۵ هستند.)

2. rank of sets:

```
[19] print("x_train rank: ", x_train.ndim)
     print("y_train rank: ", y_train.ndim)
     print("x_test rank: ", x_test.ndim)
     print("x_test rank: ", y_test.ndim)
```

```
x_train rank:  4
y_train rank:  2
x_test rank:  4
x_test rank:  2
```

این خروجی نشان‌دهنده تعداد ابعاد هر کدام از این آرایه‌ها می‌باشد. یعنی x_train و x_test آرایه های ۴ مولفه‌ای و y_train و y_test آرایه های ۲ مولفه‌ای هستند. فعلا نمی‌توانیم حدس خوبی از این ابعاد بزنیم اما در بخش بعدی همین سوال متوجه می‌شویم.

3. shape of sets:

```
print("x_train shape:", np.shape(x_train))
print("y_train shape:", np.shape(y_train))
print("x_test shape:", np.shape(x_test))
print("y_test shape:", np.shape(y_test))
```

```
x_train shape: (50000, 32, 32, 3)
y_train shape: (50000, 1)
x_test shape: (10000, 32, 32, 3)
y_test shape: (10000, 1)
```

این خروجی ابعاد دقیق هر کدام از این آرایه‌ها را نمایش می‌دهد. یعنی می‌فهمیم که در x_train، ۵۰۰۰۰ عکس وجود دارم که ابعاد ۳۲*۳۲ دارند و سه مولفه رنگی آن‌ها را در اختیار داریم. X_test هم مشابه همان است و فقط تعداد عکس ها ۱۰۰۰۰ تاست. همچنین y_train، به ازای هر کدام از آن ۵۰۰۰۰ عکس یک label دارد و y_test هم به همین صورت برای ۱۰۰۰۰ عکس موجود در داده های تست.

** فایل Q3.ipynb پاسخ این بخش می‌باشد و ضمیمه شده است.

۴. پاسخ سوال چهارم

پیاده سازی این سوال در فایل Q4.ipynb انجام شده و پیوست شده است. همچنین در لینک زیر قابل مشاهده و اجراست:

https://colab.research.google.com/drive/1XxfrB7pDcnkh0i8EI-qNcOjQHOhp_VMm?usp=sharing

همانطور که می بینید به دقت ۹۶ درصد با الگوریتم navie bayes می‌رسیم.