
**دانشکده مهندسی کامپیوتر**

**مباحث ویژه ۱ (یادگیری عمیق)**

**تمرین ۱**

علی صداقی

۹۷۵۲۱۳۷۸

## سوال ۱)

**AI Winter:** در سال های ۱۹۷۰ پیشبینی وجود داشت که که بین ۳ تا ۸ سال ماشین های ما دانش عمومی انسان های معمولی را خواهند داشت. اما همچنان (سال ۲۰۱۶) این هدف بسیار دور به نظر می رسد. در صورتی که در بین سال های ۱۹۶۰ و ۱۹۷۰ محققان آن را نزدیک خود می دیدند. چندین سال بعد که انتظارات برآورده نشد، محققان و دولت مردان هزینه و زمان و بودجه خود را از این فیلد برداشتند و ما اولین زمستان هوش مصنوعی را تجربه کردیم.

در سال ۱۹۸۰ هم که صحبت Symbolic AI بود سیستم های نخبه شروع به جلب توجه شرکت های بزرگ کردند. و چندین موفقیت باعث شد سرمایه گذاری دوباره به این فیلد برگردد. اما در سال ۱۹۹۰ ثابت شد این سیستم ها هزینه زیادی دارند و scale کردن آن ها سخت می‌باشد همچنین اسکوپ کوچکی دارند. در نتیجه علاقه مندی ها کاسته شد و هوش مصنوعی دومین زمستان خود را تجربه کرد.

**Backpropagation:** یک راه و شیوه تا بتوانیم زنجیره هایی از عملیات پارامتری را به صورت بازگشتی انجام بدهیم. (با استفاده از روش های بهینه سازی مانند gradient descent). در واقع در این روش با استفاده از قاعده مشتق زنجیره ای (chain rule) تاثیر خطا در لایه آخر در میان لایه های میانی و قبلی منتشر میشود و میتوان پارامتر های آن هارا نیز به صورت آگاهانه بهینه کنیم.

**Objective Function:** در واقع در بحث ما همان loss function میشود. کمیتی که باید در زمان آموزش الگوریتم بهینه و کمینه کنیم. بیانگر مقدار موفقیت وظیفه (task) ای است که در حال انجام هستیم. (تابع هدف)

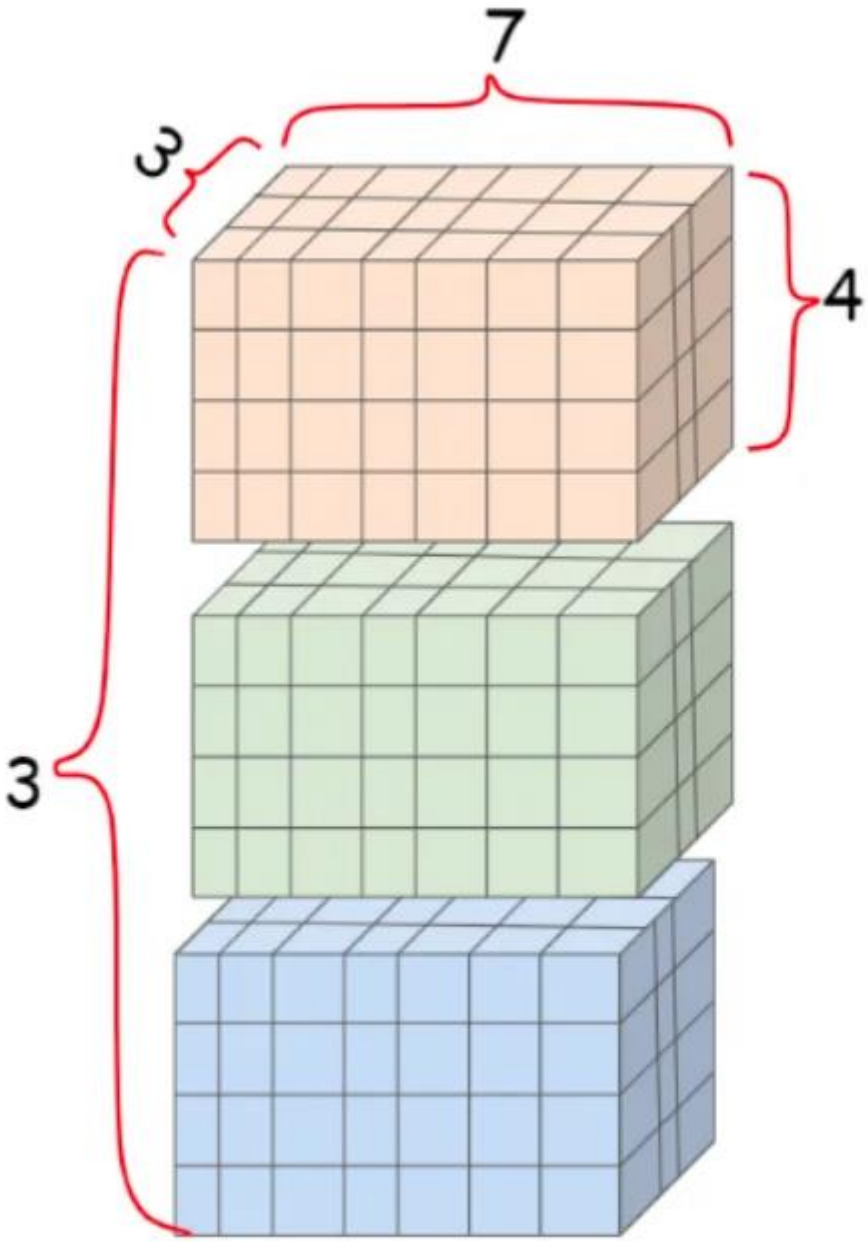
Kernel Methods: در سال های ۱۹۹۰ که شبکه های عصبی شروع به کسب احترام میان محققان کردند، یک روش جدید در یادگیری ماشین به نام روش کرنل محبوب شد. kernel method ها یک گروه از الگوریتم های دسته بندی هستند. معروف ترین آن ها الگوریتم Support Vector Machine(SVM) است. نام آن از kernel trick ها برداشته شده است. تابع کرنل یک عملیات محاسباتی قابل ردیابی است که دو نقطه در فضای اولیه را به فضای ثانویه نمایشی نگاشت می کند (فضای هدف). نتیجه آن دستیابی به نمایش جدید و مطلوب از داده قبلی است.

**4D tensor vs. 4-dimensional vector:**

```
np.array([4, 6, 8, 12])
```

مثال بالا یک 4-d vector است. در واقع در این حالت ما یک axis داریم و در آن ۴ عضو موجود می باشد. rank آن برابر ۱ می‌باشد و shape آن به صورت (4,). به آن rank 1 array نیز میگویند.

در تنسور ۴ بعدی در واقع rank برابر ۴ می‌باشد در نتیجه تاپل مربوط به shape آن دارای ۴ عضو است. در شکل زیر یک نمونه از تنسور ۴ بعدی آورده شده است.




Shape of a 4-D tensor

مثلا در سوال ۳ عبارت x_train یک تنسور ۴ بعدی است.

**Element-wise product vs. Tensor product:** در ضرب element-wise با استفاده از کاراکتر * هر درایه از ماتریس در درایه متناظرش در ماتریس دیگر ضرب می شود و شکل دو ماتریس باید یکی باشد (یا قابل broadcast

باشد). اما منظور از ضرب تنسور همان ضرب ماتریس ها در ریاضیات است که که درایه های موجود در تنسور ورودی به نوعی با هم ترکیب می شوند. نام دیگر آن dot product یا ضرب نقطه ای است که در هندسه تحلیلی با آن آشنا شدیم. در واقع در ضرب تنسور ها عملیات زیر به صورت vectorized صورت می گیرد:

```
z = 0.
for i in range(x.shape[0]):
    z += x[i] * y[i]
return z
```

## سوال ۲)

$$Sample\ Variance: \sigma^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

$$Gaussian\ Distribution = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-u)^2}{2\sigma^2}\right)$$

- هنگامی که واریانس برابر صفر شد احتمال را برابر یک در نظر می‌گیریم.

| Class | mean(F1) | var(F1) | mean(F2) | var(F2) | mean(F3) | var(F3) |
|---|---|---|---|---|---|---|
| Spam | 0.16 | 0.16 | 0.83 | 0.16 | 0.66 | 0.26 |
| Not Spam | 1.00 | 0.00 | 0.25 | 0.25 | 0.25 | 0.25 |

نتایج جدول بالا با کد نیز چک شده است.

```
{0: {'prob': 0.6, 'mean': array([0.16666667, 0.83333333, 0.66666667]), 'var': array([0.16666667, 0.16666667, 0.26666667])},
1: {'prob': 0.4, 'mean': array([1.  , 0.25, 0.25]), 'var': array([0.  , 0.25, 0.25])}}
```

حالا با توجه به فرمول Naïve Bayes احتمال هر کلاس را محاسبه می‌کنیم:

$$p(class|data) = \frac{p(data|class) \times p(class)}{p(data)}$$

برای محاسبه $p(data|class)$ از توزیع گاوسی استفاده می‌کنیم.

**ورودی [1 1 0]:**

| Class | prob | p(F1\|class) | p(F2\|class) | p(F3\|class) |
|---|---|---|---|---|
| Spam | 0.60 | 0.12 | 0.89 | 0.33 |
| Not Spam | 0.40 | 1.00 | 0.25 | 0.70 |

حالا برای به دست آوردن صورت کسر Posterior جدول بالا را به صورت سطری ضرب میکنیم.

Posterior Numerator (Spam) = 0.0211

Posterior Numerator (Not Spam) = 0.07

**ورودی [1 1 1]:**

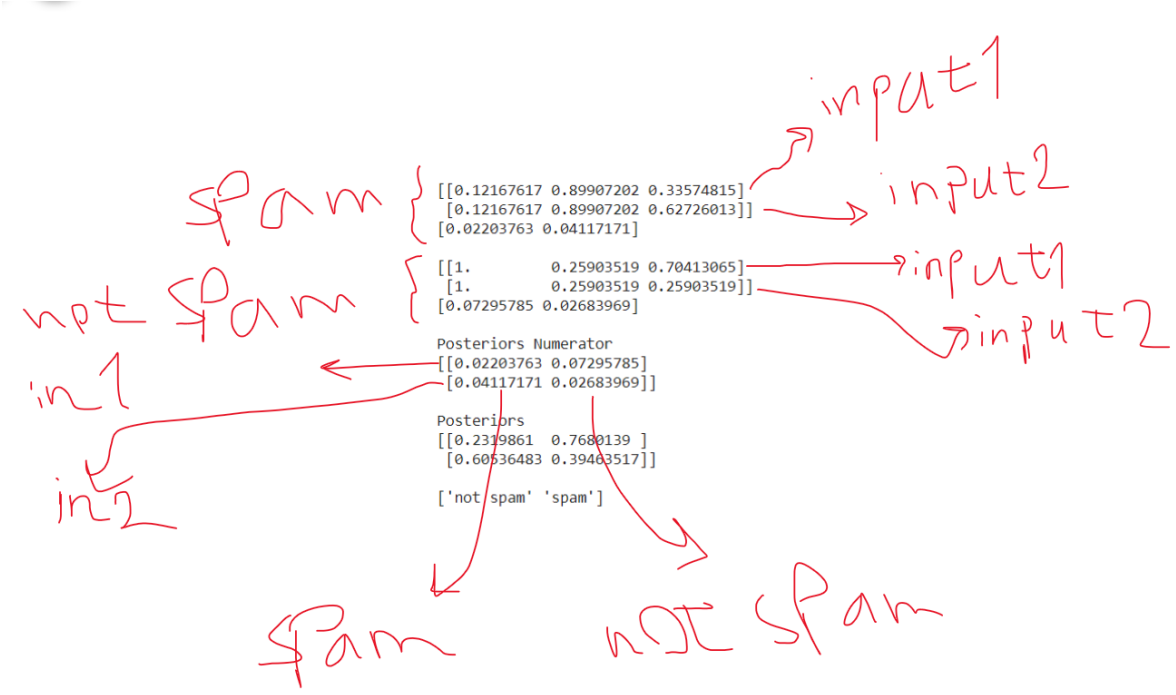
| Class | prob | p(F1\|class) | p(F2\|class) | p(F3\|class) |
|---|---|---|---|---|
| Spam | 0.60 | 0.12 | 0.89 | 0.62 |
| Not Spam | 0.40 | 1.00 | 0.25 | 0.25 |

حالا برای به دست آوردن صورت کسر Posterior جدول بالا را به صورت سطری ضرب میکنیم.

Posterior Numerator (Spam) = 0.0397296

Posterior Numerator (Not Spam) = 0.025

توسط کد نیز مراحل بالا ارزیابی شد. (تفاوت رقم ها به دلیل در نظر گرفتن تا دو رقم اعشار است.)



## سوال ۳)

1) Data Type:

نوع المان های (Elements) داخل یک آرایه نامپای را نشان می‌دهد. در این مثال دیتا تایپ المان ها عدد ۸ بیتی بدون علامت (Unsigned) می‌باشد که می‌تواند مقادیر ۰ تا ۲۵۵ را به خود بگیرد.

```
# Data Type
print(x_train.dtype)
print(y_train.dtype)
print(x_test.dtype)
print(y_test.dtype)
```

```
uint8
uint8
uint8
uint8
```

2) Rank: Number of array dimensions. The number of axes is rank.

تعداد بعد های یک آرایه نامپای را بر میگرداند. در واقع طول تاپلی است که تابع shape بر میگرداند.

3D space [7, 8, 3] is an array of rank 1

```
# Rank
print(x_train.ndim)
print(y_train.ndim)
print(x_test.ndim)
print(y_test.ndim)
```

```
4
2
4
2
```

3) Shape: The elements of the shape tuple give the lengths of the corresponding array dimensions.

طول و تعداد المان های هر بعد از یک آرایه نامپای را بر میگرداند.

```
[593] # Shape
      print(x_train.shape)
      print(y_train.shape)
      print(x_test.shape)
      print(y_test.shape)

      (50000, 32, 32, 3)
      (50000, 1)
      (10000, 32, 32, 3)
      (10000, 1)
```

مثلا در این دیتاست ۵۰۰۰۰ تصویر برای آموزش موجود است که هر یک دارای طول و عرض ۳۲ پیکسل است و ۳ کانال دارد.

این ۵۰۰۰۰ تصویر هر یک دارای یک لیبل هستند.

۱۰۰۰۰ تصویر نیز برای تست وجود دارد که طول و عرض هر تصویر ۳۲ پیکسل است و ۳ کانال دارد. همچنین برای هر تصویر یک label موجود است.

**سوال ۴)**

ابتدا دیتاست را با استفاده از تابع separate_classes بر اساس کلاس‌ها دسته بندی می‌کنیم.

سپس در تابع classes_info اطلاعات زیر مربوط به هر کلاس را به دست می‌آوریم:

- prob: احتمال رویداد آن کلاس
- mean: میانگین هر ویژگی در آن کلاس
- var: واریانس هر ویژگی در آن کلاس

تابع fit یک wrapper بر روی تمامی توابع بالا می باشد.

تابع gaussian_distribution با گرفتن نمونه جدید و داشتن mean و std مقدار احتمال گاوسی هر ورودی در آن کلاس را حساب می‌کند.

تابع predict یک wrapper بر روی توابع بالا است، و در نهایت صورت کسر posterior محاسبه می‌شود.

تابع normalize نیز صورت کسر و مخرج کسر (evidence) را تقسیم میکند و مقدار مهایی posterior به دست می آید.

✓ کد بالا بر روی مثال درون اسلاید تست شد و نتایج آن مطابق اسلاید بود.

```
Posteriors Numerator
[[6.19707184e-09 5.37790918e-04]]

Posteriors
[[1.15230663e-05 9.99988477e-01]]

['female']
```

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

✓ همچنین این کد بر روی سوال ۲ نیز تست شد که نتایج آن در سوال ۲ موجود است.

❖ **نتایج بر روی دیتاست Iris:**

دیتاست:

```
(150, 4)
(150, 3)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

نتیجه:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Accuracy: 96.0%

**منابع:**

داکیومنت نامپای

کتاب مرجع

https://medium.com/analytics-vidhya/data-representations-for-neural-networks-tensor-vector-scaler-basics-4beae5910398