به نام خدا

ملیکا احمدی رنجبر تمرین اول

❖

1. AI Winter: رویدادی که پس از افزایش غیرمنطقی انتظارات در بازه کوتاه و بیش از حد توان واقعی AI رخ دهد و کاهش سرمایه‌گذاری را به دلیل برآورده نشدن آن انتظارات همراه داشته باشد.
2. Backpropagation: روشی برای آموزش عملیات پی‌درپی شامل پارامتر که بر اساس Gradient-Descent است و در واقع برای قسمت بهینه‌سازی در شبکه مورد استفاده قرار می‌گیرد.
3. Objective Function: تابعی که خروجی پیش‌بینی شده شبکه را با خروجی واقعی مقایسه و فاصله این دو را محاسبه می‌کند. در واقع عملکرد شبکه را ارزیابی می‌کند که Loss Function نیز نام دارد.
4. Kernel Methods: گروهی از الگوریتم‌های دسته‌بندی است که بهترین الگوریتم شناخته شده آن SVM است. مفهوم الگوریتم آن است که برای دسته‌بندی نیازی به محاسبه مختصات دقیق هر نقطه در فضای جدید نیست و تنها لازم است فاصله بین نقاط در آن فضا مشخص شود. آن نیز به راحتی توسط Kernel Function قابل اجرا است.
5. 4D tensors vs. 4-dimensional vector: Vector اشاره به یک آرایه یک بعدی دارد که بسته به تعداد ورودی آرایه (X) آن را X-dimensional vector می‌گویند. برای مثال آرایه یک بعدی روبه‌رو [8, 19, 22, 20, 4, 0] شامل6 عدد است اما همه در کنار هم و یک بعدو بنابراین 6-dimensional vector است در حالی که همچنان در دسته‌بندی 1D tensor است. اما 4D tensor شامل 4 بعد است. برای مثال ماتریسی از ماتریس را در نظر بگیرید. در اینجا تعداد ورودی در هر بعد تفاوتی ندارد و تاثیری در نام‌گذاری ندارد. نکته مهم آن است که مفهوم Dimension در هر بخش Tensor و Vector درست فهمیده شود. بهتر است برای جلوگیری از اشتباه از Rank برای Tensor استفاده شود. Rank همان بعد Tensor را مشخص می‌کند (Axis).
6. Element-wise product vs. Tensor product: در واقع همان ضرب Tensor product داخلی است به این معنا که المان هر سطر و ستون ابتدا ضرب و سپس تمامشان جمع می‌شوند و یک المان را برای Tensor جدید ایجاد می‌کند. در حالی که Element-wise product تک تک ا المان‌ها را فقط ضرب می‌کند مثلا : [4, 6, 8] * [1, 3] -> [[4, 6, 8], [12, 18, 24]]
اما در Tensor product داریم : [4] dot [2, 5] -> 4 * 2 + 4 * 5 = 28

❖ در این سوال باید در نهایت دو احتمال زیر را برای هر تست داده شده محاسبه کنیم و سپس احتمال بیشتر را به عنوان کلاس برای آن تست انتخاب کنیم:

P (Spam | x = [1 1 0])

P (Not spam | x = [1 1 0])

P (Spam | x = [1 1 1])

P (Not spam | x = [1 1 1])

حال با توجه به اینکه Feature 3 داریم (هر کدام از اعداد) و روش نیز Naïve Bayes است باید این 3 را Independent در نظر بگیریم و احتمالات را حساب کنیم.

فرمول به این صورت است :

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(X|Y) = \prod P(X^k|Y)$$

P (Spam) = 0.6, P (Not spam) = 0.4, Evidence = 10

P (First place 1 | Spam) = 1/6, P (Second place 1 | Not spam) = 1

P (Second place 1 | Spam) = 5/6, P (Second place 1 | Not spam) = 1/4

P (Third place 1 | Spam) = 4/6, P (Third place 1 | Not spam) = 1/4

P (Third place 0 | Spam) = 2/6, P (Third place 0 | Not spam) =3/4

x = [1 1 0]

P (Spam | x = [1 1 0]) = (P (Spam) * P (First place 1 | Spam) * P (Second place 1 | Spam) * P (Third place 0 | Spam)) / Evidence

P (Spam | x = [1 1 0]) = 0.6 * 1/6 * 5/6 * 2/6 / 10 = 0.0027

P (Not spam | x = [1 1 0]) = 0.4 * 1 * 1/4 * 3/4 / 10 = 0.0075 This sample is Not spam

x = [1 1 1]

P (Spam | x = [1 1 1]) = (P (Spam) * P (First place 1 | Spam) * P (Second place 1 | Spam) * P (Third place 1 | Spam)) / Evidence

P (Spam | x = [1 1 1]) = 0.6 * 1/6 * 5/6 * 4/6 / 10 = 0.0037 This is a spam

P (Not spam | x = [1 1 1]) = 0.4 * 1 * 1/4 * 1/4 / 10 = 0.0025

❖ به طور کلی Data type نوع ساختار استفاده شده را نشان می‌دهد, مثلا عدد استفاده شده float32 است یا uint8.

Rank بعد Tensor را بیان می‌کندبرای مثال 10 داده عکس 20 *20 رنگی دارای Rank 4 است به شکل زیر: (10, 20, 20, 3)

و در نهایت Shape تعداد ورودی هر بعد (Axes) را نشان می‌دهد همچون مثال بالا (10, 20, 20, 3).

حال برای هر خروجی به شکل زیر است:

```
---------- Train Dataset ----------
Train dataset data type:  uint8
Train dataset rank:  4
Train dataset shape:  (50000, 32, 32, 3)
```

- *برای نمونه‌های آموزشی: به این معنا که جنس اعداد داده uint8 است و Tensor دارای 4 بعد است که می‌توان تعداد ورودی را در هر بعد در خط آخر مشاهده کرد. به طور دقیق, 50000 نمونه آموزشی داریم که هر کدام 32 * 32 پیکسل است و دارای معیار رنگ RGB است.*

```
---------- Train Label ----------
Train label data type:  uint8
Train label rank:  2
Train label shape:  (50000, 1)
```

- *برای پاسخ‌های آموزشی: به این معنا که جنس اعداد داده uint8 است و Tensor دارای 2 بعد است که می‌توان تعداد ورودی را در هر بعد در خط آخر مشاهده کرد. به طور دقیق, 50000 پاسخ برای نمونه‌های آموزشی داریم که هر کدام یک عدد است که کلاس آن نمونه را مشخص کند.*

```
---------- Test Dataset ----------
Test dataset data type:  uint8
Test dataset rank:  4
Test dataset shape:  (10000, 32, 32, 3)
```

- *برای نمونه‌های تست: به این معنا که جنس اعداد داده uint8 است و Tensor دارای 4 بعد است که می‌توان تعداد ورودی را در هر بعد در خط آخر مشاهده کرد. به طور دقیق, 10000 نمونه برای تست داریم که هر کدام 32 * 32 پیکسل است و دارای معیار رنگ RGB است.*

```
---------- Test Label ----------
Test label data type:  uint8
Test label rank:  2
Test label shape:  (10000, 1)
```

- *برای پاسخ‌های تست: به این معنا که جنس اعداد داده uint8 است و Tensor دارای 2 بعد است که می‌توان تعداد ورودی را در هر بعد در خط آخر مشاهده کرد. به طور دقیق, 10000 پاسخ برای نمونه‌های تست داریم که هر کدام یک عدد است که کلاس آن نمونه را مشخص کند.*

❖ در این سوال پیاده‌سازی Naïve Bayes را برای داده‌های پیوسته داریم و برای همین با کمک محاسبه واریانس و میانگین مسئله را حل می‌کنیم با استفاده از همین فرمول:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(X|Y) = \prod P(X^k|Y)$$

با توجه به پیوسته بودن مقدار ویژگی در این دسته اطلاعات از توزیع نرمال برای محاسبه احتمال استفاده کردیم که به شکل زیر است:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

در کد نیز, محاسبه این احتمال به شکل زیر است:

```
def CalculateConditionalProbability(X, Var, Mean):

    Prob = (1 / math.sqrt(2 * np.pi * Var)) * math.exp(-1 * (pow(X - Mean, 2) / (2 * Var)))
    return Prob
```

بعد از محاسبه این قسمت که برای هر Feature باید حساب شود, طبق Naïve Bayes در هم ضربشان می‌کنیم و همچنین در احتمال آن کلاس خاص که به صورت زیر است:

```
# We Already Have This, 1/3
def CalculateClassProbability(Label, Class):

    ClassProb = np.count_nonzero(Label == Class) / Evidence
    return ClassProb
```

اگرچه می‌دانیم از هر 3 دسته به شکل مساوی داده داریم.

بعد از آنکه این احتمالات برای هر دسته حساب شد, بیشترین آن را به عنوان Label برای داده انتخاب می‌کنیم:

```
def SelectFinalClass(Classes):

    Class = np.argmax(Classes)
    return Class
```

نکته دیگر برای محاسبه دقت است که اگر Label واقعی و Prediction برابر بود عدد یک را به آرایه و در غیر این صورت 0 را اضافه می‌کنیم. در انتها نیز تعداد 1 را برای دقت حساب می‌کنیم.

```
def Accuracy(SelectedCategory, ActualAnswer):
    if (SelectedCategory == 0 and ActualAnswer == 'setosa') or (SelectedCategory == 1 and ActualAnswer == 'versicolor') or (SelectedCategory == 2 and ActualAnswer == 'virginica'):
        AccuracyNum.append(1)
    else:
        AccuracyNum.append(0)
```

اعداد دسته نیز به شکل بالا است.

آموزش بر روی تمام داده‌هاست و تست نیز به همین شکل. حلقه بر روی کل داده. حلقه برای احتمال کلاس‌ها. و در نهایت حلقه برای هر Feature تا احتمال شرطی حساب شود.

```
# Test
for Sample in Data:
    ProbabilitiesForClasses = []
    for Class in range(0, 3):
        Up = EachClassProbability[Class]
        for Feature in range(0, 4):
            Up *= CalculateConditionalProbability(Sample[Feature], Var[Class][Feature], Mean[Class][Feature])
        FinalClassProb = Up / Evidence
        ProbabilitiesForClasses.append(FinalClassProb)
    Predicted = SelectFinalClass(ProbabilitiesForClasses)
    Accuracy(Predicted, Label[It])
    It += 1
```

```python
def CalculateVarMean(Data):
    Var = [0] * 3
    Mean = [0] * 3
    # First Class
    Var[0] = np.var(Data[0:49], axis=0)
    Mean[0] = np.mean(Data[0:49], axis=0)

    # Second Class
    Var[1] = np.var(Data[50:99], axis=0)
    Mean[1] = np.mean(Data[50:99], axis=0)

    # Third Class
    Var[2] = np.var(Data[100:149], axis=0)
    Mean[2] = np.mean(Data[100:149], axis=0)
    return Var, Mean;
```

Variance و Mean برای هر دسته هم به این شکل است. با توجه به Index تقسیم‌بندی شده.

❖ منابع:

- ✓ Deep Learning with Python
- ✓ https://numpy.org/doc/stable/reference/generated/numpy.var.html
- ✓ https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html