CSE 523 Machine Learning
Weekly Report Progress
(01-04-23)

Topic - Password strength checker

Group name - Predictors_4.0

NAME	ENROLLMENT NUMBER
Zalak Shah	AU2040217
Freyal Shah	AU2040228
Freya Modi	AU2040222
Avinash Raval	AU2040031

This week we started implementing logistic regression code for password strength detection.

We did the following things:

- From the password, we extracted features like length, and arrays containing boolean values for has_upper, has_digit, has_specialchar columns.

Data Preprocessing

```
[10] #Defining functions
     def has_special_char(s):
         return any(c in string.punctuation for c in s)

     data.dropna(inplace=True)
     data['length'] = data['password'].apply(len)
     data['has_upper'] = data['password'].apply(lambda x: any(c.isupper() for c in x))
     data['has_digit'] = data['password'].apply(lambda x: any(c.isdigit() for c in x))
     data['has_specialchar'] = data['password'].apply(lambda x: any(has_special_char(x) for c in x))
```

Define features and target variable

```
[16] X = np.hstack((np.ones((len(data), 1)), data[['length', 'has_upper', 'has_digit','has_specialchar']].values))
     y = data['strength'].values
```

0s

data

	password	strength	length	has_upper	has_digit	has_specialchar
0	kzde5577	1.0	8	False	True	False
1	kino3434	1.0	8	False	True	False
2	visi7k1yr	1.0	9	False	True	False
3	megzy123	1.0	8	False	True	False
4	lamborghin1	1.0	11	False	True	False
...	...	...	...	...	...	...
564046	uglyand123	1.0	10	False	True	False
564047	knzaojdi8	1.0	9	False	True	False
564048	kei55553	1.0	8	False	True	False
564049	viniabde2	1.0	9	False	True	False
564050	rangaj777	1.0	9	False	True	False

564050 rows × 6 columns

- Then we defined the sigmoid function and cost function for the logistic regression.

Defining functions

```
#sigmoid function
def sigmoid(a):
    return 1 / (1 + np.exp(-a))

#cost function
def cost_function(theta, X, y):
    m = len(y)
    h = sigmoid(X @ theta)
    J = (-1/m) * (y @ np.log(h) + (1-y) @ np.log(1-h))
    grad = (1/m) * (X.T @ (h - y))
    return J, grad
```

- After this we started Training the model using the technique of gradient descent.

Initializing theta

```
[20] theta = np.zeros(X.shape[1])
```

Training logistic regression model using gradient descent

```
[24] theta = np.zeros(X.shape[1])
     alpha = 0.01
     iterations = 1000
     m = len(y)
     for i in range(iterations):
         h = X.dot(theta)
         loss = h - y
         gradient = X.T.dot(loss) / m
         theta = theta - alpha * gradient
     print(theta)
```

```
[-0.16247568398086581 0.12747363033865153 0.18953044206187775
 -0.1410874960361894 0.006525865016034045]
```

Code link:
https://colab.research.google.com/drive/1a3KzKSMWzbop7tYvZwn-B_rezTcps-0w?usp=sharing