In [ ]:

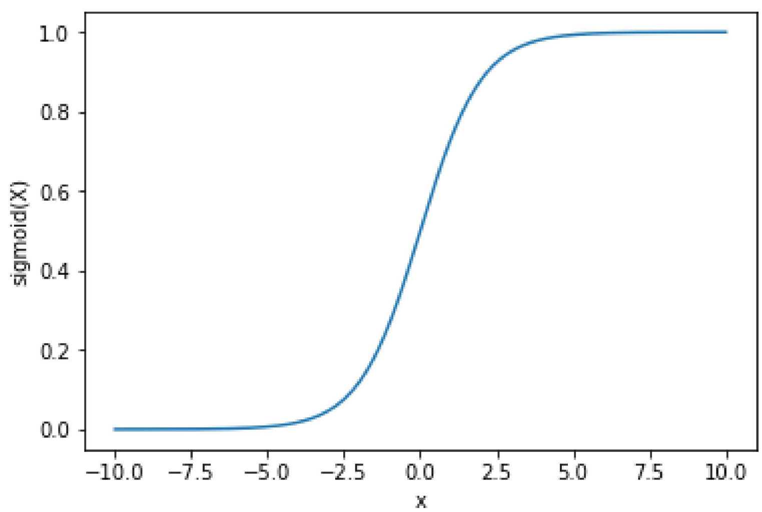
225229140

part 1

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt


def sigmoid(x):
    return np.where(x < 0, np.exp(x) / (1 + np.exp(x)), 1 / (1 + np.exp(-x)))


val = np.linspace(start=-10, stop=10, num=200)
sigmoid_values = sigmoid(val)
plt.plot(val, sigmoid_values)
plt.xlabel("x")
plt.ylabel("sigmoid(X)")
plt.show()
```



In [8]:
```
import numpy as np
import math
def sigmoid(x):
    return math.exp(x)
sigmoid(4)
```

Out[8]: 54.598150033144236

In [7]:
```
def mysigmoid(x):
    return np.exp(x)
arr=np.array([1,2,3])
print(mysigmoid(arr))
```

```
[ 2.71828183  7.3890561  20.08553692]
```

part 2

In [23]:
```
def sig_derivative(s):
    s=math.exp(s)
    s_derivative=(s*(1-s))
    return s_derivative

sig_derivative(4)
```

Out[23]:
```
-2926.359837008584
```

part 3

In [27]:
```
import numpy as np

def image_to_vector(image):
    vector = image.reshape((image.shape[0] * image.shape[1] * image.shape[2], 1
    return vector
image = np.array([[[255, 0], [0, 255], [128, 128]],
                  [[0, 128], [128, 0], [255, 255]],
                  [[64, 64], [192, 192], [0, 0]]])

vector = image_to_vector(image)
print(vector)
```

```
[[255]
 [  0]
 [  0]
 [255]
 [128]
 [128]
 [  0]
 [128]
 [128]
 [  0]
 [255]
 [255]
 [ 64]
 [ 64]
 [192]
 [192]
 [  0]
 [  0]]
```

part 4

In [29]:
```
import numpy as np

def normalizeRows(x):

    row_norms = np.linalg.norm(x, axis=1, keepdims=True)
    normalized_x = x / row_norms
    return normalized_x

x = np.array([[0, 3, 4],
              [2, 6, 4]])

normalized_x = normalizeRows(x)
print(normalized_x)
```

```
[[0.         0.6        0.8       ]
 [0.26726124 0.80178373 0.53452248]]
```

part 5

In [37]:
```
import numpy as np

x1 = np.array([9, 2, 5])
x2 = np.array([7, 2, 2])

mul_result = np.multiply(x1, x2)
print("Multiplication:", mul_result)


dot_result = np.dot(x1, x2)
print("Dot product:", dot_result)
```

```
Multiplication: [63  4 10]
Dot product: 77
```

In [33]:
```
import numpy as np

x1 = np.array([9, 2, 5, 0, 0, 7, 5, 0, 0, 0, 9, 2, 5, 0, 0, 4, 5, 7])
x2 = np.array([7, 2, 2, 9, 0, 9, 2, 5, 0, 0, 9, 2, 5, 0, 0, 8, 5, 3])

mul_result = np.multiply(x1, x2)
print("Multiplication:", mul_result)


dot_result = np.dot(x1, x2)
print("Dot product:", dot_result)
```

```
Multiplication: [63  4 10  0  0 63 10  0  0  0 81  4 25  0  0 32 25 21]
Dot product: 338
```

In [35]:
```
import numpy as np
import time

N = 1000000


x1 = np.random.random(N)
x2 = np.random.random(N)


start_time = time.time()
mul_result = np.multiply(x1, x2)
end_time = time.time()
mul_time = end_time - start_time


start_time = time.time()
dot_result = np.dot(x1, x2)
end_time = time.time()
dot_time = end_time - start_time

print("Multiplication time:", mul_time)
print("Vectorization (dot product) time:", dot_time)
```

```
Multiplication time: 0.0029916763305664062
Vectorization (dot product) time: 0.0009708404541015625
```

In [38]:
```
#part 6
```

In [39]:
```
import numpy as np

actual = np.array([1, 0, 0, 1, 1])
prediction = np.array([.9, 0.2, 0.1, .4, .9])
```

In [40]:
```
l1_loss = np.sum(abs(actual - prediction))
print(l1_loss)
```

```
1.1
```

In [41]:
```
y = np.array([1, 0, 0, 1, 1])
ypred = np.array([.9, 0.2, 0.1, .4, .9])
l2_loss = np.sum((y - ypred) ** 2)
print(l2_loss)
```

```
0.43
```

In [ ]: