Mạng nơ ron đơn tầng, 2 lớp (Tensorflow): thực hành 1, 2, 3 slides 3

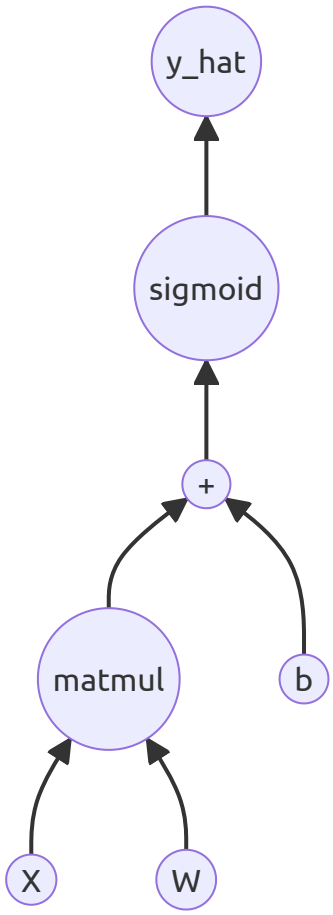
- Họ tên: Kim Minh Thắng
- Mã số sinh viên: B2007210

Thực hành 1

1.1 Import tensorflow

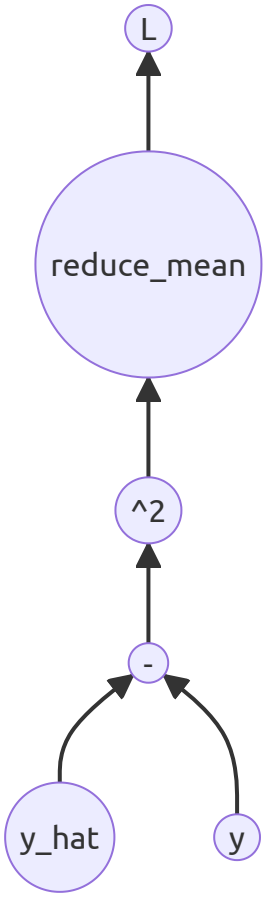
In [ ]:
```
import tensorflow as tf
```

1.2 Đồ thị tính toán



In [ ]:
```
@tf.function
def predict(X, W, b):
    return tf.sigmoid(tf.matmul(X, W) + b)
```

1.3 Hàm lỗi



In [ ]:
```
@tf.function
def loss(y, y_hat):
    return tf.reduce_mean((y - y_hat)**2)
```

1.4 Tính đạo hàm riêng tự động với `GradientTape`

In [ ]:
```
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])
W = tf.Variable(tf.random.normal((2, 1)))
b = tf.Variable(tf.random.normal(()))

alpha = 0.1
```

In [ ]:
```
loss_his = []

for it in range(500):
    with tf.GradientTape() as tape:
        current_loss = loss(y, predict(X, W, b))

    loss_his.append(current_loss)

    dW, db = tape.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
```

1.5 Dự đoán

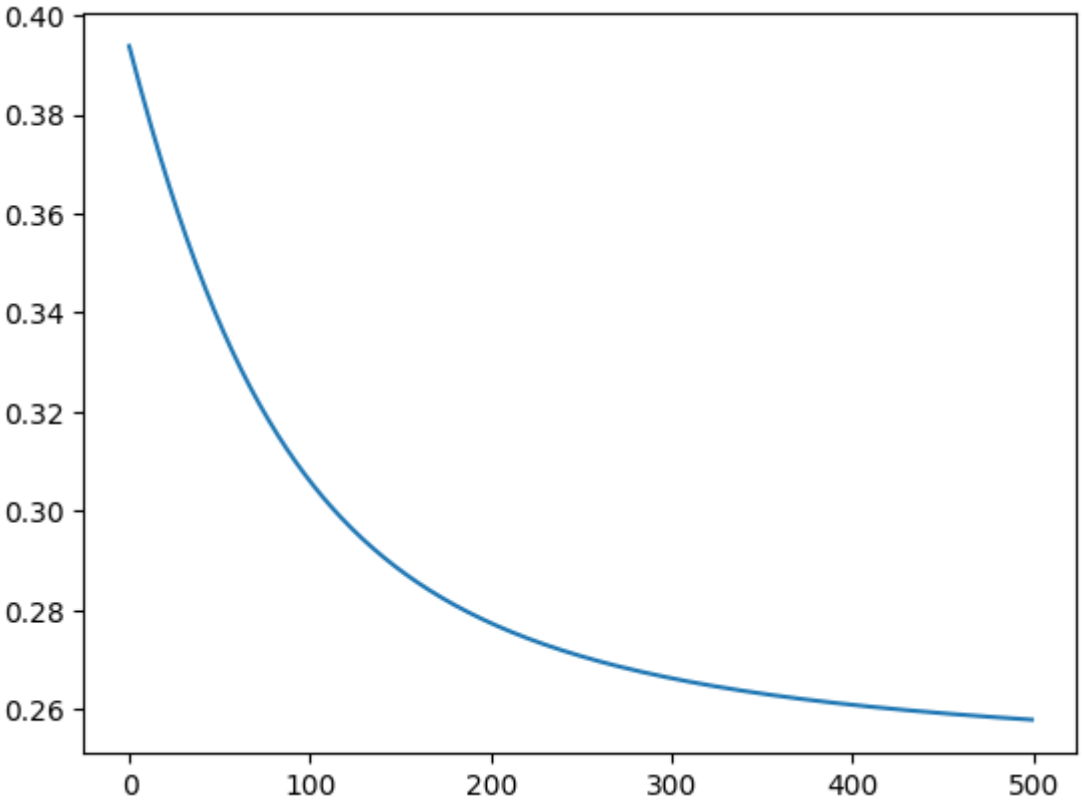
In [ ]:
```
y_hat = predict(X, W, b)
y_hat
```

Out[ ]:
```
<tf.Tensor: shape=(4, 1), dtype=float32, numpy=
array([[0.19804282],
       [0.02177161],
       [0.0694706 ],
       [0.00668346]], dtype=float32)>
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(loss_his)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f106fc3cb80>]
```



Thực hành 2

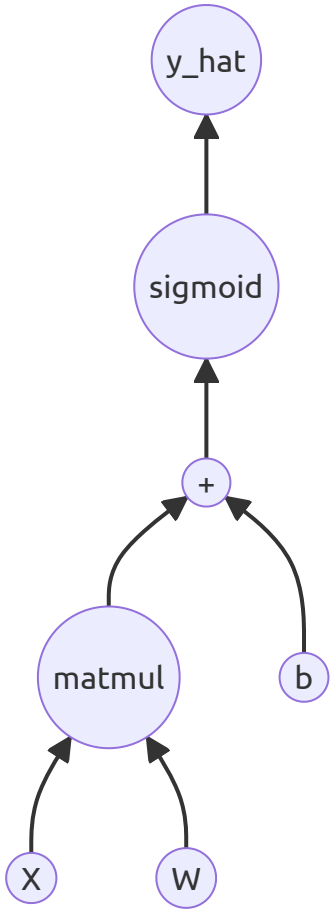
- Làm lại bài thực hành 1 với hàm lỗi `binary crossentropy` được định nghĩa như sau:

$$\mathrm{L}(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

2.1 Import tensorflow

In [ ]:
```
import tensorflow as tf
```

2.2 Đồ thị tính toán



In [ ]:
```
@tf.function
def predict(X, W, b):
    return tf.sigmoid(tf.matmul(X, W) + b)
```

2.3 Hàm lỗi

$$\mathrm{L}(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

In [ ]:
```
@tf.function
def loss(y, y_hat):
    return - tf.reduce_sum(y * tf.math.log(y_hat) + (1 - y) * tf.math.log
```

2.4 Tính đạo hàm riêng tự động với `GradientTape`

In [ ]:
```
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])
W = tf.Variable(tf.random.normal((2, 1)))
b = tf.Variable(tf.random.normal(()))

alpha = 0.1
```

In [ ]:
```
loss_his = []
for it in range(500):
    with tf.GradientTape() as tape:
        current_loss = loss(y, predict(X, W, b))
```

```
    loss_his.append(current_loss)

    dW, db = tape.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
```

2.5 Dự đoán

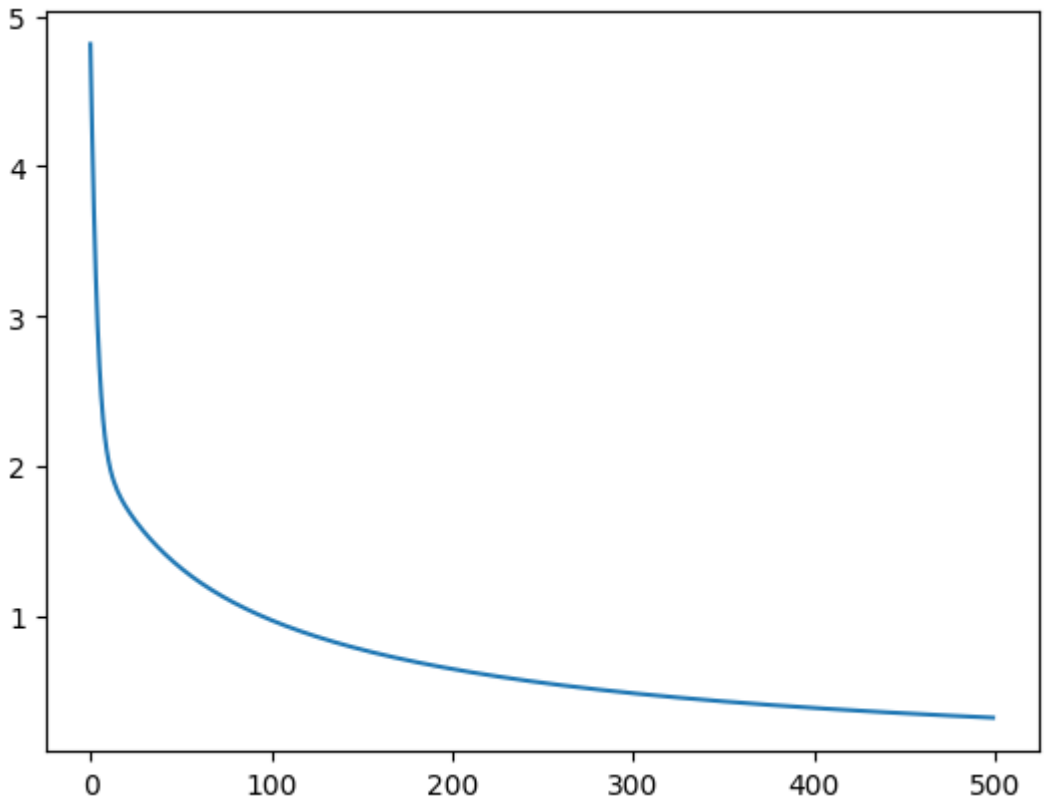
In [ ]:
```
y_hat = predict(X, W, b)
y_hat
```

Out[ ]:
```
<tf.Tensor: shape=(4, 1), dtype=float32, numpy=
array([[0.00133216],
       [0.08831336],
       [0.08836398],
       [0.8756035 ]], dtype=float32)>
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(loss_his)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f106fe33bb0>]
```



Thực hành 3

Làm lại bài phân loại hoa iris (2 lớp) với tensorflow thay vì dùng keras

- Sử dụng hàm lỗi `binary crossentropy`
- Cần phân ngưỡng kết quả đầu ra để có nhãn chính xác
- Tính độ chính xác phân lớp bằng cách so sánh nhãn dự báo và nhãn mong muốn

Có thể dùng hàm `Tensor.numpy()` để lấy giá trị của tensor về dạng numpy để hậu xử lý.

Import các thư viện cần thiết

In [ ]:
```
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

3.1 Đọc và dữ liệu

In [ ]:
```
df = pd.read_csv('./Iris.csv')
df = df.head(100)
```

In [ ]:
```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
X
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
95	5.7	3.0	4.2	1.2
96	5.7	2.9	4.2	1.3
97	6.2	2.9	4.3	1.3
98	5.1	2.5	3.0	1.1
99	5.7	2.8	4.1	1.3

100 rows × 4 columns

In [ ]:
```
le = LabelEncoder()

y = df['Species'].values

le.fit(y)

y = le.transform(y)

pd.DataFrame(y)
```

Out[ ]:

	0
0	0
1	0
2	0
3	0
4	0
...	...
95	1
96	1
97	1
98	1
99	1

100 rows × 1 columns

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [ ]:
```
x_train = tf.convert_to_tensor(x_train, dtype=tf.float32)
x_test = tf.convert_to_tensor(x_test, dtype=tf.float32)
y_train = tf.constant(y_train, dtype=tf.float32)
y_test = tf.constant(y_test, dtype=tf.float32)
```

3.2 Định nghĩa hàm dự đoán

In [ ]:
```
@tf.function
def predict(X, W, b):
    return tf.sigmoid(tf.matmul(X, W) + b)
```

3.3 Định nghĩa hàm lỗi (binary crossentropy)

In [ ]:
```
@tf.function
def loss(y, y_hat):
    return - tf.reduce_mean(y * tf.math.log(y_hat) + (1 - y) * tf.math.lo
```

3.4 Train model

In [ ]:
```
W = tf.Variable(tf.random.normal((4, 1)))
b = tf.Variable(tf.random.normal(()))
alpha = 0.1
```

In [ ]:
```
loss_his = []

for it in range(500):
    with tf.GradientTape() as tape:
        current_loss = loss(y_train, predict(x_train, W, b))

    loss_his.append(current_loss)
```

```
    dW, db = tape.gradient(current_loss, [W, b])

    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
```

3.5 Dự đoán

In [ ]:
```
y_hat = predict(x_test, W, b)

y_hat = y_hat.numpy()

y_hat = [1 if i >= 0.5 else 0 for i in y_hat]

y_hat = tf.convert_to_tensor(y_hat, dtype=tf.float32)

print(f'Loss: {abs(y_hat - y_test).numpy().sum() / len(y_test) * 100}%')
```

```
Loss: 50.0%
```

In [ ]:
```
plt.plot(loss_his)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f106fc3f100>]
```