hw1

September 13, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sympy as sy

from IPython.display import display, Math, Latex

import log_reg
```

1 Create Dataset

The dataset is meant to represent whether a day is considered good or not, given temperature, wind, and humitidy.

[2]:
```
data = pd.read_csv("A1_Data_EliWeissler.csv")
data
```

[2]:

	temp	wind	humidity	LABEL
0	80	2	10	Good
1	100	5	80	Bad
2	90	10	100	Bad
3	70	1	25	Good
4	75	6	40	Good
5	85	15	70	Bad
6	80	10	85	Bad

Normalizing the data to be between 0-1, we get (rounding to two decimal places):

[3]:
```
X, Y = log_reg.normalize_data(data)
Math("X = " + sy.latex(sy.Matrix(np.round(X, 2))) +\
     "\quad Y = " + sy.latex(sy.Matrix(Y)))
```

[3]:

$$X = \begin{bmatrix} 0.33 & 0.07 & 0 \\ 1.0 & 0.29 & 0.78 \\ 0.67 & 0.64 & 1.0 \\ 0 & 0 & 0.17 \\ 0.17 & 0.36 & 0.33 \\ 0.5 & 1.0 & 0.67 \\ 0.33 & 0.64 & 0.83 \end{bmatrix} \quad Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

2 Linear Equations

For this problem, consider a linear equation with n variables. This gives us $w \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^1$, and $z \in \mathbb{R}^1$.

2.1 a)

In general, we will have:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

2.2 b)

In vector form, the sum is equivalent to a dot product between c and x:

$$z = w^T x + b$$

2.3 c)

The w values represent weights for each varible. This can be thought of as the slope of the hyperplane in R^n with respect to said variable (i.e., $\partial_{x_i} z = w_i$)

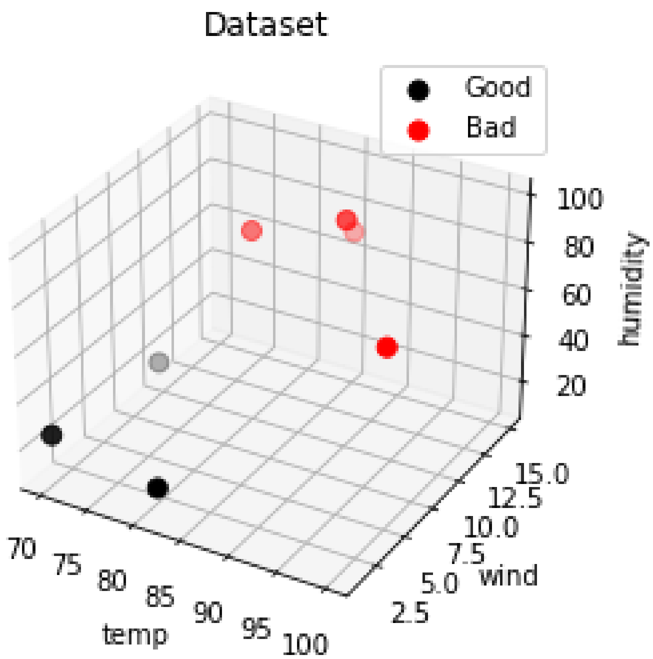
2.4 d)

The b value represents a bias, or offset. This is the value of z when all variables are 0 (i.e., y intercept in the 1-d case of $ y = mx + b$).

3 Plot Data

[4]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

# plotting
colors = ["k", "r"]
for i, is_good in enumerate(["Good","Bad"]):
    data_subset = data[data["LABEL"] == is_good]
    ax.scatter(data_subset["temp"], data_subset["wind"],␣
 ↪data_subset["humidity"],
               s=50, c=colors[i], label=data_subset["LABEL"].iloc[0])
ax.set_title('Dataset')
ax.set_xlabel("temp")
ax.set_ylabel("wind")
ax.set_zlabel("humidity")
ax.legend()
plt.show()
```



4 Data Clustered?

The data is easily linearly separable with respect to the labeling

5 Sigmoid

5.1 a)

The sigmoid function is

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

or

$$\sigma(x) = \frac{e^x}{e^x + 1}$$

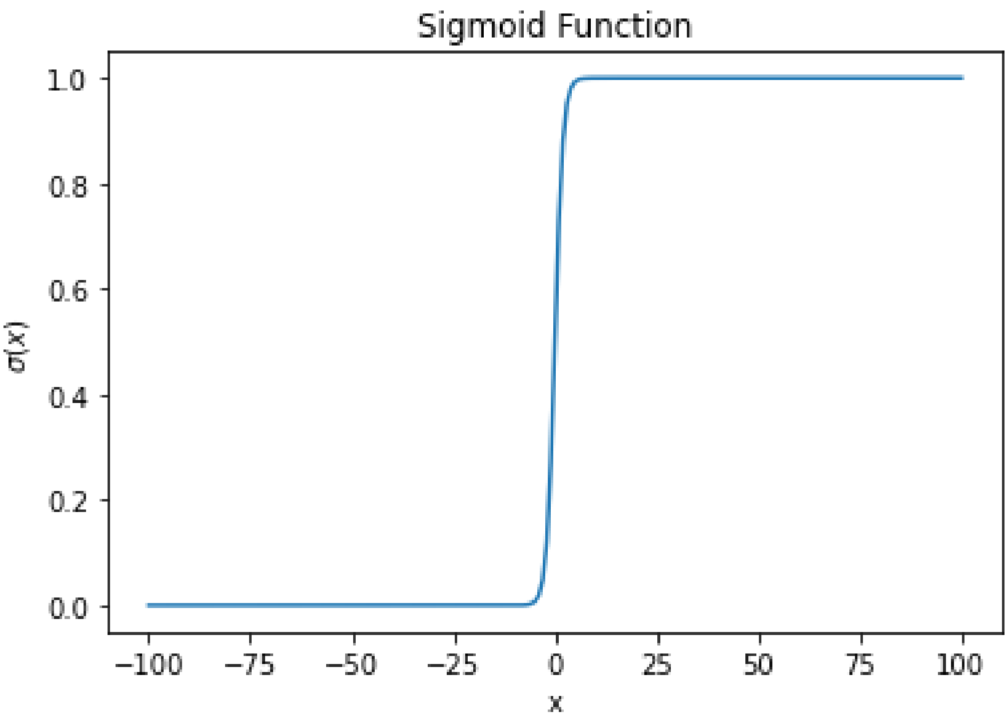
5.2 b)

To show that the two forms are equal, multiply the first form by e^x/e^x:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-x}} \cdot \frac{e^x}{e^x} = \frac{e^x}{e^x + 1}$$

5.3 c)

[5]:
```
def sigmoid(x):
    return 1/(1 + np.exp(-x))
x = np.arange(-100, 101)
plt.plot(x, sigmoid(x))
plt.xlabel("x")
plt.ylabel(r"$\sigma(x)$")
plt.title("Sigmoid Function");
```



5.4 d)

Using the quotient rule

$$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$$

we can see that:

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) = \frac{d}{dx}[\frac{1}{1+e^{-x}}] &= \frac{0\cdot(1+e^{-x}) - 1\cdot(-e^{-x})}{(1+e^{-x})^2} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}}(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}) \\
&= \boxed{\sigma(x)(1-\sigma(x))}
\end{aligned}
$$

6 Feed Forward

7 Loss Calculation

8 Changing W

9 Calculating Gradient

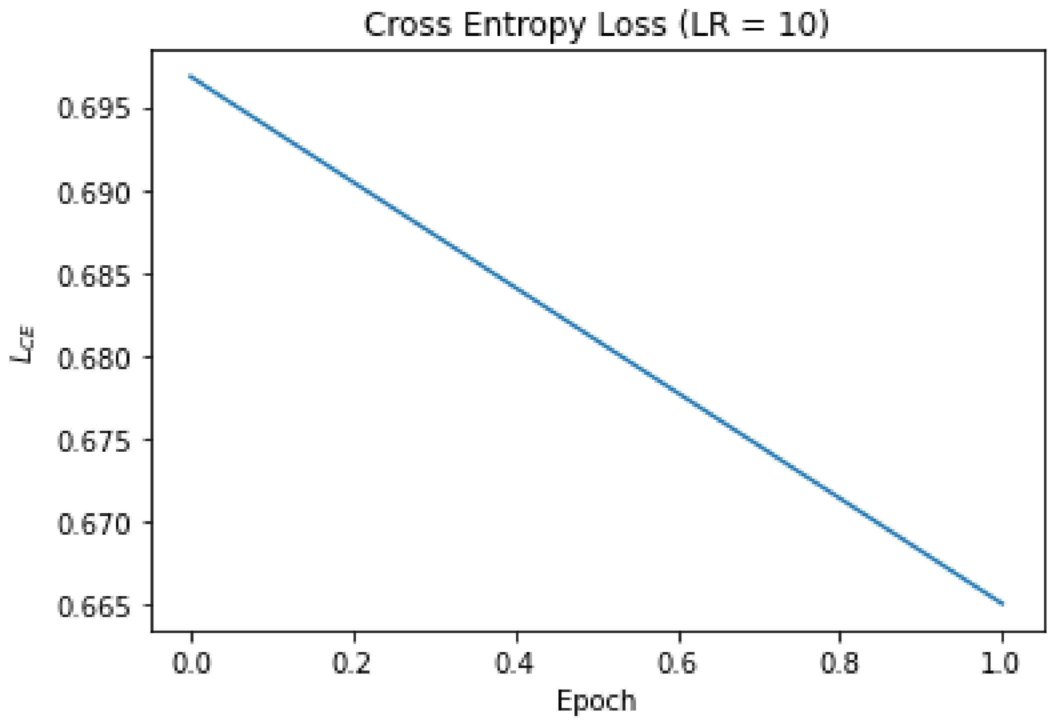
10 Gradient Descent Example

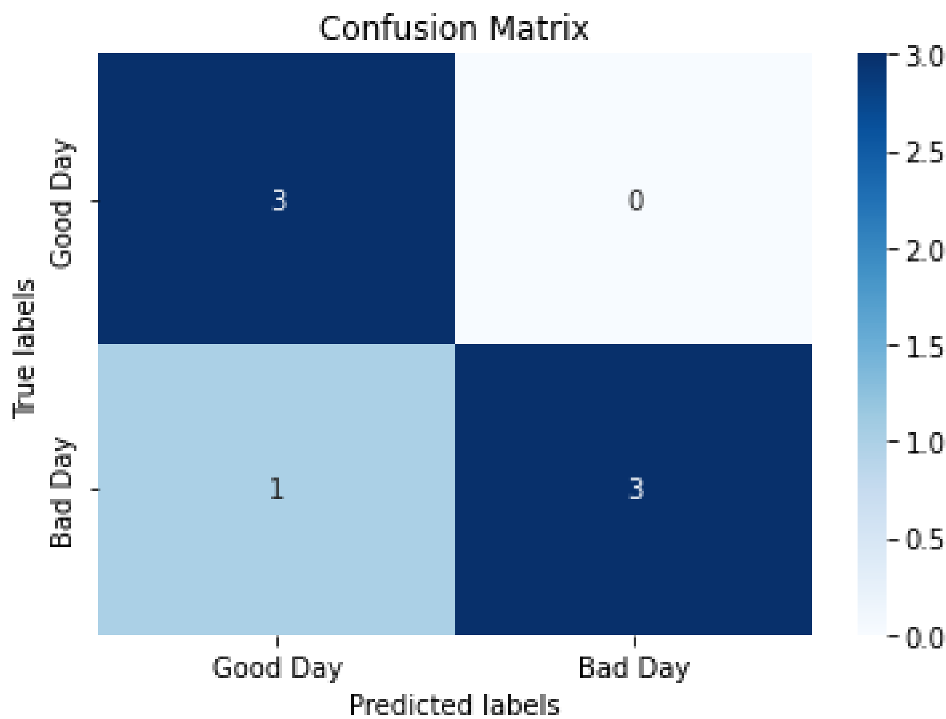
See math pdf.

11 Coding Logistic Regression

See log_reg.py for implementation details. Begin by trying to match the pen/paper results of one iteration:

[15]:
```
W, b, pred, loss = log_reg.logistic_reg("A1_Data_EliWeissler.csv", epochs=2,
    lr=10)
```


Cross Entropy Loss (LR = 10)
0.695
0.690
0.685
0.680
0.675
0.670
0.665
L_{CE}
0.0
0.2
0.4
0.6
0.8
1.0
Epoch

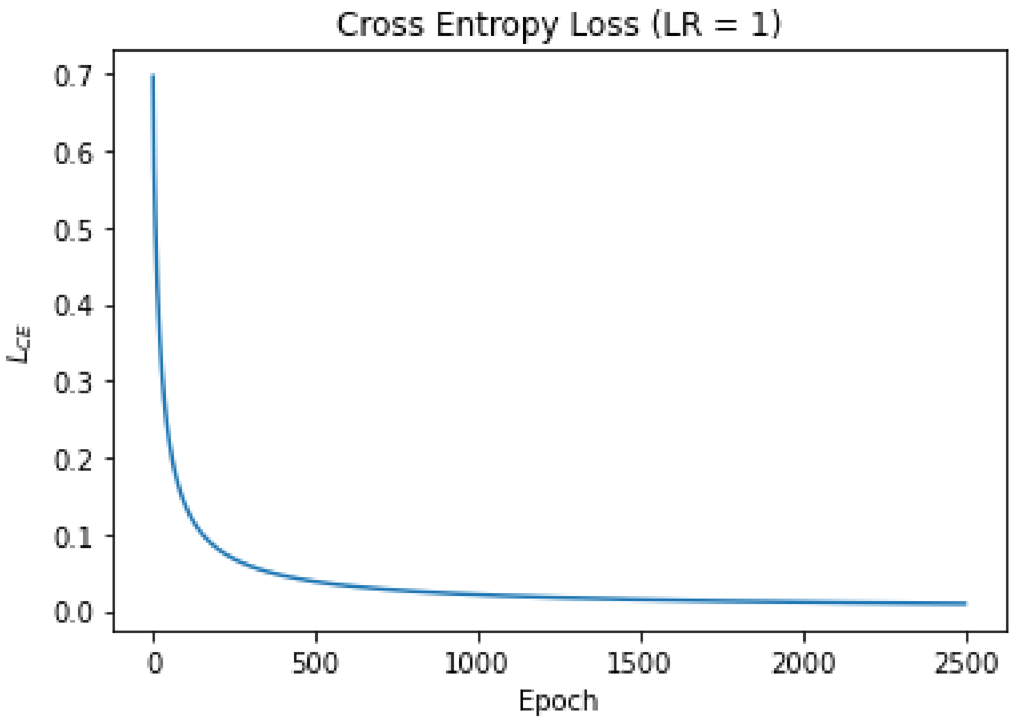

Confusion Matrix
3
0
1
3
Good Day
Bad Day
True labels
Good Day
Bad Day
Predicted labels
3.0
2.5
2.0
1.5
1.0
0.5
0.0

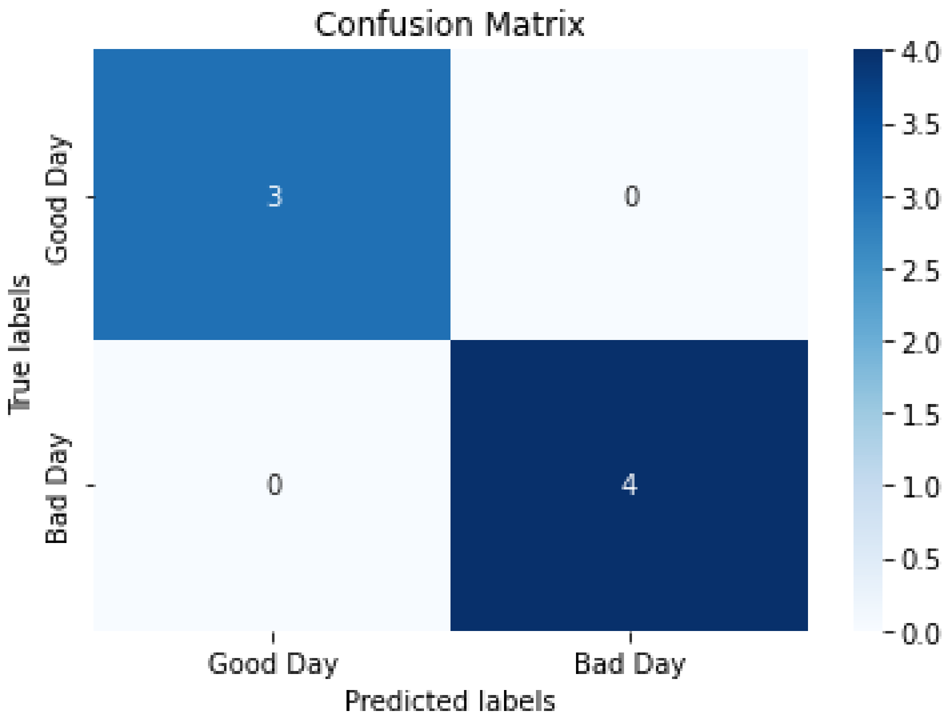
[16]:
```
W, b, loss
```

[16]:
```
(array([0.93465326, 0.98560302, 1.05639437]),
 -1.9956456075099909,
 array([0.69682832, 0.66509351]))
```

The weights and bias have updated as I had predicted with a learning rate of 10. The loss function is close, but we wouldn't expect it to be the same because of rounding and a non-zero bias update. Now let's run it for many more epochs:

[24]:
```
W, b, pred, loss = log_reg.logistic_reg("A1_Data_EliWeissler.csv", epochs=2500,␣
↪lr=1)
```





[28]:
```
print("Final W = ", W)
print("Final b = ", b)
print("Final Loss = ", loss[-1])
```

```
Final W =  [ 6.969755    6.06826653 10.17445771]
Final b =  -10.410035211162398
Final Loss =  0.009028718549532344
```

Yay, we were able to successfully predict the labels of our clearly separable data!