m3-l1-p1

February 9, 2024

1 M3-L1 Problem 1 (5 points)

[13]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1.1 Sigmoid function

Define a function, `sigmoid(h)`, which computes and returns the sigmoid `g(h)` given an input `h`. Recall the mathematical formulation of sigmoid:

$$g(h) = \frac{1}{1 + e^{-h}}$$

[14]:
```
def sigmoid(h):
    # YOUR CODE GOES HERE
    sigmoid_h = 1/(1+np.exp(-h))
    return sigmoid_h
```

1.2 Transformation function

In logistic regression, we transform the input before applying the sigmoid function. This transformation can take many forms, but here let's define a function `transform_quadratic(x,w)` that takes in an input `x`, and a weight vector `w`, and returns the sum $ w_0 1 + w_1 x + w_2 x^2 $.

[15]:
```
def transform_quadratic(x, w):
    # YOUR CODE GOES HERE
    transform_sum = w[0] * 1 + w[1] * x + w[2] * x**2
    return transform_sum
```

1.3 Example

Now, we will use both `sigmoid()` and `transform_quadratic()` in a logistic regression context.

Suppose a logistic regression model states that:

$$P(y = 1 \mid x) = g(\mathbf{w}'x),$$

for $g(h)$ the sigmoid function and $\mathbf{w} = [4, -3, 2]$.

Use the functions you wrote to compute $P(y = 1 \mid x = 1.2)$ and $P(y = 1 \mid x = 7)$. Print these probabilities.

[16]:
```
w = [4,-3,2]
for x in [1.2, 7.]:
    P = sigmoid(transform_quadratic(x,w))
    print(f"x = {x:3} --> P(y=1) = {P}")
```

```
x = 1.2 --> P(y=1) = 0.9637362836253517
x = 7.0 --> P(y=1) = 1.0
```