section2

September 17, 2023

[6]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
```

[7]:
```
def fit_and_return_coef(x, y):
    model = LinearRegression()
    model.fit(x, y)  # x is (n_samples, n_features), y is (n_samples, n_targets)
    # print(model.score(x, y))
    return model.intercept_, model.coef_


def check_fitting(x, intercept, weights):
    return np.matmul(x, weights) + intercept


# q2.2:
x_2 = np.array([[-1, -1], [-1, +1], [+1, -1], [+1, +1]])
y_2 = np.array([+1, +1, +1, -1])
w0, coefs = fit_and_return_coef(x_2, y_2)

print(check_fitting(x_2, w0, coefs))
print(w0, coefs)
```

```
[ 1.5  0.5  0.5 -0.5]
0.5 [-0.5 -0.5]
```

[8]:
```
# q2.3:
x_3 = np.array([[-1, -1, -1],
                [-1, -1, +1],
                [-1, +1, -1],
                [-1, +1, +1],
                [+1, -1, -1],
                [+1, -1, +1],
                [+1, +1, -1],
                [+1, +1, +1]])
y_3 = np.array([-1, -1, -1, +1, -1, +1, +1, +1])
w0, coefs = fit_and_return_coef(x_3, y_3)

print(check_fitting(x_3, w0, coefs))
print(w0, coefs)
```

```
[-1.5 -0.5 -0.5  0.5 -0.5  0.5  0.5  1.5]
0.0 [0.5 0.5 0.5]
```

[9]:
```
# q2.4 A xor B = (A nand B) and (A or B)
x_4 = np.array([[-1, -1], [-1, +1], [+1, -1], [+1, +1]])
y_41 = np.matmul(x_4, np.array([-1, +1])) + 0.5
y_42 = np.matmul(x_4, np.array([+1, -1])) + 0.5
x_4final = np.concatenate((y_41, y_42)).reshape(2, 4).T

print('before masking (outputs from layer 1):\n' + str(x_4final))

x_4final[x_4final > 0] = +1
x_4final[x_4final < 0] = -1
y_4final = np.array([-1, +1, +1, -1])
# print(y_41, y_42)
print('after masking (outputs from layer 1):\n'  + str(x_4final))
intercept, arr = fit_and_return_coef(x_4final, y_4final)

# print('testing ' + str(np.matmul(x_4final, np.array(arr)) + intercept))

print('fitting xor (w20, w21, w22) = ' + str(fit_and_return_coef(x_4final,␣
 ↪y_4final)))
print(str(check_fitting(x_4final, *fit_and_return_coef(x_4final, y_4final))))

# print('fit xor ' + str(fit_and_return_coef(x_4_part_2, y_4final)))
# print('check xor ' + str(check_fitting(x_4_part_2,␣
 ↪*fit_and_return_coef(x_4_part_2, y_4final))))
```

```
before masking (outputs from layer 1):
[[ 0.5  0.5]
 [ 2.5 -1.5]
 [-1.5  2.5]
 [ 0.5  0.5]]
after masking (outputs from layer 1):
[[ 1.  1.]
 [ 1. -1.]
 [-1.  1.]
 [ 1.  1.]]
fitting xor (w20, w21, w22) = (1.0, array([-1., -1.]))
[-1.  1.  1. -1.]
```