M6-L1 Problem 1

MinMax Scaling

MinMax scaling scales the data such that the minimum in each column is transformed to 0, and the maximum in each column is transformed to 1, using the formula $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$.

For example, for a training matrix X, MinMax scaling will give: $$ X =

$$\begin{bmatrix} 10 & 2 \\ 9 & 0 \\ 3 & 8 \\ 4 & 9 \\ 0 & 8 \\ -3 & 2 \\ -4 & -2 \\ 5 & 6 \end{bmatrix}$$

\overset{X}{\longrightarrow} \begin{bmatrix} 0.364\ 0.929 & 0.182\ 0.5 & 0.909\ 0.571 & 1.\ 0.286 & 0.909\ 0.071 & 0.364\ 0.\ 0.643 & 0.727\ \end{bmatrix} $$ Applying the same scaling to the given matrix of test data `A` should yield the following (notice we are scaling according to `X`, not `A`).

$$A = \begin{bmatrix} 2 & 1 \\ 5 & 5 \\ 11 & -1 \\ -3 & 2 \end{bmatrix} \overset{X}{\longrightarrow} \begin{bmatrix} 0.429 & 0.273 \\ 0.643 & 0.636 \\ 1.071 & 0.091 \\ 0.071 & 0.364 \end{bmatrix}$$

Implementing MinMax Scaling

A function to compute the minimum and maximum of each column is provided. Complete the scaling function `MinMax_scaler(X, Min, Max)` below to implement $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$.

Validate your results by comparing to the above.

In [4]:

```
import numpy as np
np.set_printoptions(precision=3)

def get_MinMax(X):
    Max = np.max(X,axis=0).reshape(1,-1)
    Min = np.min(X,axis=0).reshape(1,-1)
    return Min, Max

def MinMax_scaler(X, Min, Max):
    # YOUR CODE GOES HERE
    # Scale values such that Max --> 1 and Min --> 0
    return (X-Min)/(Max-Min)

# Loading train data X and test data A:
```

```
X = np.array([[10,9,3,4,0,-3,-4,5],[2,0,8,9,8,2,-2,6]]).T
A = np.array([[2,5,11,-3],[1,5,-1,2]]).T

# Getting the scaling constants for each column of X:
Xmin, Xmax = get_MinMax(X)

# Scaling X and A, and printing the results
print("X =\n", X, " -->\n", MinMax_scaler(X,Xmin,Xmax))
print("A =\n", A, " -->\n", MinMax_scaler(A,Xmin,Xmax))
```

```
X =
 [[10  2]
 [ 9  0]
 [ 3  8]
 [ 4  9]
 [ 0  8]
 [-3  2]
 [-4 -2]
 [ 5  6]]  -->
 [[1.     0.364]
 [0.929 0.182]
 [0.5   0.909]
 [0.571 1.   ]
 [0.286 0.909]
 [0.071 0.364]
 [0.    0.   ]
 [0.643 0.727]]
A =
 [[ 2  1]
 [ 5  5]
 [11 -1]
 [-3  2]]  -->
 [[0.429 0.273]
 [0.643 0.636]
 [1.071 0.091]
 [0.071 0.364]]
```

Standard Scaling

Standard scaling scales the data according to the mean (μ) and standard deviation (σ) of the training data. Scaling uses the formula $X' = \frac{X-\mu}{\sigma}$.

For example, for a training matrix X, Standard scaling will give: $$ X =

$$\begin{bmatrix} 10 & 2 \\ 9 & 0 \\ 3 & 8 \\ 4 & 9 \\ 0 & 8 \\ -3 & 2 \\ -4 & -2 \\ 5 & 6 \end{bmatrix}$$

\overset{X}{\longrightarrow} \begin{bmatrix} 1.46 & -0.547\ 1.251 & -1.061\

1. & 0.997\ 0.209 & 1.254\ -0.626 & 0.997\ -1.251 & -0.547\ -1.46 & -1.576\ 0.417 & 0.482\ \end{bmatrix} $$ Applying the same scaling to the given matrix of test data `A` should yield the following (notice we are scaling according to `X`, not `A`).

$$A = \begin{bmatrix} 2 & 1 \\ 5 & 5 \\ 11 & -1 \\ -3 & 2 \end{bmatrix} \xrightarrow{X} \begin{bmatrix} -0.209 & -0.804 \\ 0.417 & 0.225 \\ 1.668 & -1.318 \\ -1.251 & -0.547 \end{bmatrix}$$

Implementing Standard Scaling

A function to compute the `mu` and `sigma` of each column is provided. Complete the scaling function `Standard_scaler(X, Min, Max)` below to implement $X' = \frac{X-\mu}{\sigma}$.

Validate your results by comparing to the above.

In [6]:
```
import numpy as np
np.set_printoptions(precision=3)

def get_MuSigma(X):
    mu = np.mean(X,axis=0).reshape(1,-1)
    sigma = np.std(X,axis=0).reshape(1,-1)
    return mu, sigma

def Standard_scaler(X, mu, sigma):
    # YOUR CODE GOES HERE
    # Scale values such that mu --> 0 and sigma --> 1
    return (X-mu)/sigma

# Loading train data X and test data A:
X = np.array([[10,9,3,4,0,-3,-4,5],[2,0,8,9,8,2,-2,6]]).T
A = np.array([[2,5,11,-3],[1,5,-1,2]]).T

# Getting the scaling constants for each column of X:
Xmu, Xsigma = get_MuSigma(X)

# Scaling X and A, and printing the results
print("X =\n", X, " -->\n", Standard_scaler(X,Xmu,Xsigma))
print("A =\n", A, " -->\n", Standard_scaler(A,Xmu,Xsigma))
```

```
X =
 [[10  2]
 [ 9  0]
 [ 3  8]
 [ 4  9]
 [ 0  8]
 [-3  2]
 [-4 -2]
 [ 5  6]]  -->
 [[ 1.46  -0.547]
 [ 1.251 -1.061]
 [ 0.      0.997]
 [ 0.209  1.254]
 [-0.626  0.997]
 [-1.251 -0.547]
 [-1.46  -1.576]
 [ 0.417  0.482]]
A =
 [[ 2  1]
 [ 5  5]
 [11 -1]
 [-3  2]]  -->
 [[-0.209 -0.804]
```

```
[ 0.417  0.225]
[ 1.668 -1.318]
[-1.251 -0.547]]
```

In [ ]: