Why to scale the features?

- A dataset may contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use **Eucledian distance** between two data points in their computations, this is a problem.
- Feature scaling can affect the condition number and thus improve the performance/convergence. Imagine a very narrow and a very broad contour for cost function. In the link below, l_2 normalization makes the condition number nearly 1, making linear models much faster to train. https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/ch04.html (https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/ch04.html).
- One of the reasons that scaling or normalization reduces condition number might be they suppress the relative difference of the numerical values among different features. If the numerical values of a feature is given in terms of a column, then scaling / normalization along columns, will give rows whose elements are not so different along each row. This will generally reduce the condition number of the matrix.
- Sometimes there might be very big difference among the values of the same feature. So the scaling or normalization within columns will not help. It is not enough to have only not-so-different values along each row. If the values of different rows along a column are still quite different, then the matrix might still be ill-conditioned, **as the dimensions/rank of the row space and column space of a matrix is always the same**. Therefore, we sometimes **log transform or square root transform** the data for the same feature, in order to suppress the huge difference within columns and thus reduce the condition number of the relevant matrix.

When to scale the features?

- Rule of thumb I follow here is any algorithm that computes distance or assumes normality, scale your features.
- k-nearest neighbors with an Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
- Scaling is critical, while performing Principal Component Analysis(PCA). PCA tries to **get the features with maximum variance and the variance is high for high magnitude features**. This skews the PCA towards high magnitude features. **Furthermore doing PCA is to diagonalize the covariance matrix. However, in the definition of covariance matrix there are subtractions of mean from each value. After centering the data, we don't have to subtract the mean anymore.**

When not necessary to scale the features?

- Tree based models are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modeling trees.
- Algorithms like Linear Discriminant Analysis(LDA), Naive Bayes are by design equipped to handle this and gives weights to the features accordingly. Performing a features scaling in these algorithms may not have much effect.
- As introduced above, scaling/normalization or log/square root transformation can help reduce the condition number or increase the effective rank of a matrix and thus increase the stability of numerical calculations. However, it is not the case that the better the condition number, the better the solution. This depends on the specific situation.
- The key advantage of either increasing increasing the effective rank or reducing the condition number is to reduce the variance in applications such as machine learning. Or, generally, we would achieve numerical stability in our calculations. However, if the problem is not ill-conditioned, then normalization or scaling should not give the better results. Scaling or normalization will anyway modify the the original data. If the benefits (e.g. reducing variance) of these modification cannot overpass the disadvantages (meaning the modification itself), then we should not do either scaling or normalization.

A strategy
Because it is impossible to have a general well-determined rule for whether we should perform feature scaling or not, a better strategy is always try either case.

How to scale the features?

- Standardization: Standardization replaces the values by their Z scores. $x' = \frac{x-\mu}{\sigma}$. This redistributes the features with their mean $\mu = 0$ and standard deviation $\sigma = 1$.
- Mean Normalization: $x' = \frac{x-\text{mean}(x)}{\max(x)-\min(x)}$. This distribution will have values between -1 and 1 with $\mu = 0$.

Standardization and Mean Normalization can be used for algorithms that assumes zero centric data like Principal Component Analysis(PCA).

1. Min-Max Scaling: $x' = \frac{x-\min(x)}{\max(x)-\min(x)}$. This scaling brings the value between 0 and 1.
2. Unit Vector: $x' = \frac{x}{\|x\|}$.

Scaling is done considering the whole feature vector to be of unit length. Min-Max Scaling and Unit Vector techniques produces values of range `[0,1]`. When dealing with features with hard boundaries this is quite useful. For example, when dealing with image data, the colors can range from only 0 to 255.

An example is shown below.

In [1]:
```python
import numpy as np
def feature_scaling(array):
    std = np.std(array)
    mean = np.mean(array)
    if(std > 0):
        array = (array-mean)/std
    else:
        print("std are zero for this array");
    return array

X_train = np.array([[1.,3.,2.,5.],[2.,4.,3.,1.],[5.,2.,6.,7.]])
# If I write X_train = np.array([[1,3,2,5],[2,4,3,1],[5,2,6,7]]), then due to the error of integer division
# the normalized matrix will be zero. So whenever division involved, make sure all the data are in float type.

print(X_train)
m, n = X_train.shape
print(m,n)
for i in range (n): # Normally it should be (1,n) if the first column is 1.
    X_train[:,i] = feature_scaling(X_train[:,i])

print(X_train)
```

```
[[1. 3. 2. 5.]
 [2. 4. 3. 1.]
 [5. 2. 6. 7.]]
3 4
[[-0.98058068  0.         -0.98058068  0.26726124]
 [-0.39223227  1.22474487 -0.39223227 -1.33630621]
 [ 1.37281295 -1.22474487  1.37281295  1.06904497]]
```

References

https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e (https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e)
https://stats.stackexchange.com/questions/189652/is-it-a-good-practice-to-always-scale-normalize-data-for-machine-learning (https://stats.stackexchange.com/questions/189652/is-it-a-good-practice-to-always-scale-normalize-data-for-machine-learning)
https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/ch04.htmlTf-idf (https://www.oreilly.com/library/view/feature-engineering-for/9781491953235/ch04.htmlTf-idf). https://stats.stackexchange.com/questions/121886/when-should-i-apply-feature-scaling-for-my-data (https://stats.stackexchange.com/questions/121886/when-should-i-apply-feature-scaling-for-my-data)