To scale or not to scale, that is the question

January 4, 2022

Scaling a dataset $\mathbf{X}$ means to bring all the columns to a comparable range. There are two main scaling methods: min-max scaling and standard scaling. We can safely say that scaling never hurts: whatever model you build, if you scale the dataset before, you never get worse performance (in terms of final accuracy or speed of convergence).

However, we can distinguish three cases:

- Scaling improves model performance (in terms of accuracy or convergence speed)
- Scaling does not change anything and
- Scaling does not improve performance but improves explainability

1 When scaling improves performance

1.1 K-means clustering

In the simplest case, K-means clustering is based on Euclidean distance between the samples and the centroids. If some features have much bigger scale than others, they will dominate in the computation of the distance. As a consequence, it is like "ignoring" the smaller scale features when deciding how to cluster, which is a pity.

K-means clustering can also use other types of distances, but the most common ones always suffer from the problem above.

1.2 All the models with regularization

With regularization, we force all the parameters to be small. Suppose a feature j of your dataset has a very small scale. Its parameter θ_j will be forced to be small (as for the other parameters). Therefore, for any sample $\mathbf{x}^{(i)}$, the quantity $\theta_j \cdot x_j^{(i)}$ will be small as well. As a consequence, the contribution of this feature will be ignored, which is a pity.

Therefore, any time you use regularization, your dataset must be scaled.

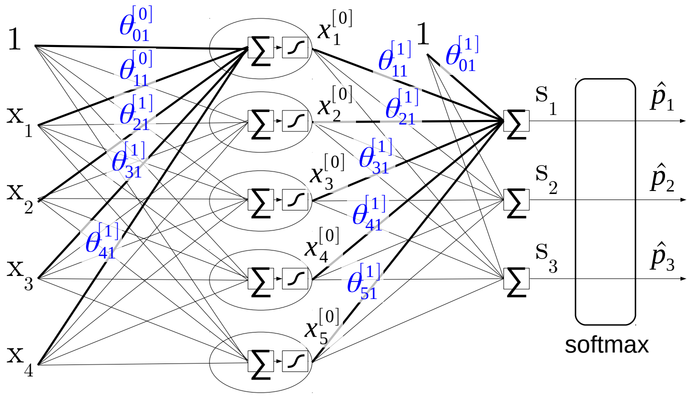
1.3 Neural networks

With **neural networks** scaling generally improves convergence speed. Why?

Let us consider a neural network and the neurons of the input layer. The output of the q-th neuron of the input layer is

$$\sigma\left(\sum_{j=1}^{N} \theta_{jq}^{[0]} \cdot x_j^{(i)}\right) \tag{1}$$

where σ is a non-linear function, e.g., sigmoid or reLu.



Suppose now the values of feature 1 are much bigger than the values of all the other features. When we perform training, we first initialize weights randomly. For any sample $\mathbf{x}^{(i)}$ given as input to our neural network, the 1st component $x_1^{(i)}$ is likely to be much bigger than all the other components $x_j^{(i)}$. Therefore, the term $\theta_{1q}^{[0]} \cdot x_1^{(i)}$ is likely to be much bigger all the other terms $\theta_{jq}^{[0]} \cdot x_j^{(i)}$. Therefore, if we look at equation (1), the argument of $\sigma(\cdot)$ is dominated by $\theta_{1q}^{[0]} \cdot x_1^{(i)}$. As a consequence, the output of that neuron is mainly dictated by feature 1, which "crushes" the contribution of all the other features.

In other words, we are starting in a very bad way our training: from the beginning we risk to erase the contribution of features with smaller magnitude. Therefore, before we fix this false start, we need a lot of iterations
$\Rightarrow$ Convergence is slow.

2 When scaling does not change anything

With all tree-based learning models (decision trees, random forests, extra-trees, isolation forest, etc.), scaling does not change anything. Indeed, when a tree is grown, we have to decide how to split the various nodes. In order to do so, we take different features, for each feature we choose a split point between its minimum and maximum value and compute the information gain of that split.

We repeat the same for several features and we choose the one that provides the highest information gain. Since the splitting point is always a number between the minimum and the maximum value, we do not need any scaling.

3 Singular Value Decomposition

(Ignore this section if we did not study Singular Value Decomposition this year)
Suppose your dataset have columns with different magnitudes, e.g., columms are in different unities of measurement, for instance bytes, gygabytes, 5-stars-ratings and audio bitrates all in the same dataset. If you apply Singular Value Decomposition (SVD) right away, the columns with the highest magnitude risk to monopolize the decomposition.

In such cases, it is advisable to scale your dataset before applying SVD. [1]

4 When scaling does not improve performance but improves explainabiity

4.1 Linear regression with no regularization

Suppose your linear regression model is

$$h(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)} \tag{2}$$

Suppose that you run Ordinary Least Squares (OLS) and find the optimal parameters $\theta_0^* = 0$, $\theta_1^* = 10$ and $\theta_2^* = 20$, such that the prediction for sample i is

$$\hat{y}^{(i)} = 10 \cdot x_1^{(i)} + 20 \cdot x_2^{(i)}$$

Let us now scale the dataset, for simplicity just multiplying by 5 the first column and by 4 the second column. If we now run again OLS, what will be the optimal parameters?

...

Well, we saw before that the best prediction for i was $\hat{y}^{(i)}$. Therefore we just need to adjust the parameters so that we obtain the same prediction on the scaled dataset. Try to fill the empty spaces in the following formula with the new optimal parameters:

$$\hat{y}^{(i)} = \_\_\_ \cdot \mathbf{5}x_1^{(i)} + \_\_\_ \cdot \mathbf{4}x_2^{(i)}$$

The answer is

$$\hat{y}^{(i)} = \frac{10}{\mathbf{5}} \cdot \mathbf{5}x_1^{(i)} + \frac{20}{\mathbf{4}} \cdot \mathbf{4}x_2^{(i)}$$

[1] To know more about scaling and dimensionality reduction, check Sec.11.5 of Härdle, W. K., and Simar, L. (2015). Applied multivariate statistical analysis, fourth edition. Applied Multivariate Statistical Analysis, Fourth Edition (4th ed.)

In other words, if we change the scale of some columns, a linear regression will scale accordingly the optimal parameters and produces the same prediction!

For this reason, scaling the dataset does not affect the performance of linear regression.

However, it may still be good to scale the dataset before applying linear regression if we want **more explainability**.

Suppose your dataset is scaled and the coefficient θ_1 is higher (in absolute value) than θ_1. If you look at equation (2), since $x_1^{(i)}$ and $x_2^{(i)}$ are on the same scale, you can affirm that feature 1 has more impact on the prediction $\hat{y}^{(i)}$. In other words, it has a higher **importance**.

If instead your dataset were not scaled, you could not have concluded anything about importance. Indeed, the coefficient θ_1 could be high but $\mathbf{x}^{(i)}1$ could be on a very small scale. As a result, the contribution of $\theta_1 \cdot \mathbf{x}^{(i)}1$ could be negligible.

To summarize, scaling does not improve performance of non-regularized linear regression, but improves explainability.

For this reason, when using recursive feature elimination (RFE) with a linear model, since RFE removes the *least important features*, the dataset must always been scaled.

4.2 Logistic Regression with no regularization

For the same reasons as before, scaling does not improve the performance of logistic regression but improves explainability.

You may say: wait a moment, logistic regression is a simple form of neural network. Therefore, if we look at the arguments of Sec. 1.3, we would expect that that logistic regression converges more slowly if we do not scale the dataset, as it was the case for neural networks. This is in theory true, but the difference in convergence speed is usually negligible with logistic regression. Indeed, recall that the loss function of logistic regression is a convex function of model parameters. Convex functions are easy to minimize applying gradient descent, and a theorem ensures that gradient descent always converges to the optimal parameters. On the contrary, the loss function of neural networks can be complicated and is not convex. Convergence toward the optimal parameters is more difficult there and is not guaranteed. This is why, if we do not scale the dataset and use neural networks, we may have troubles. While if we do the same with logistic regression we may not notice any visible difference.