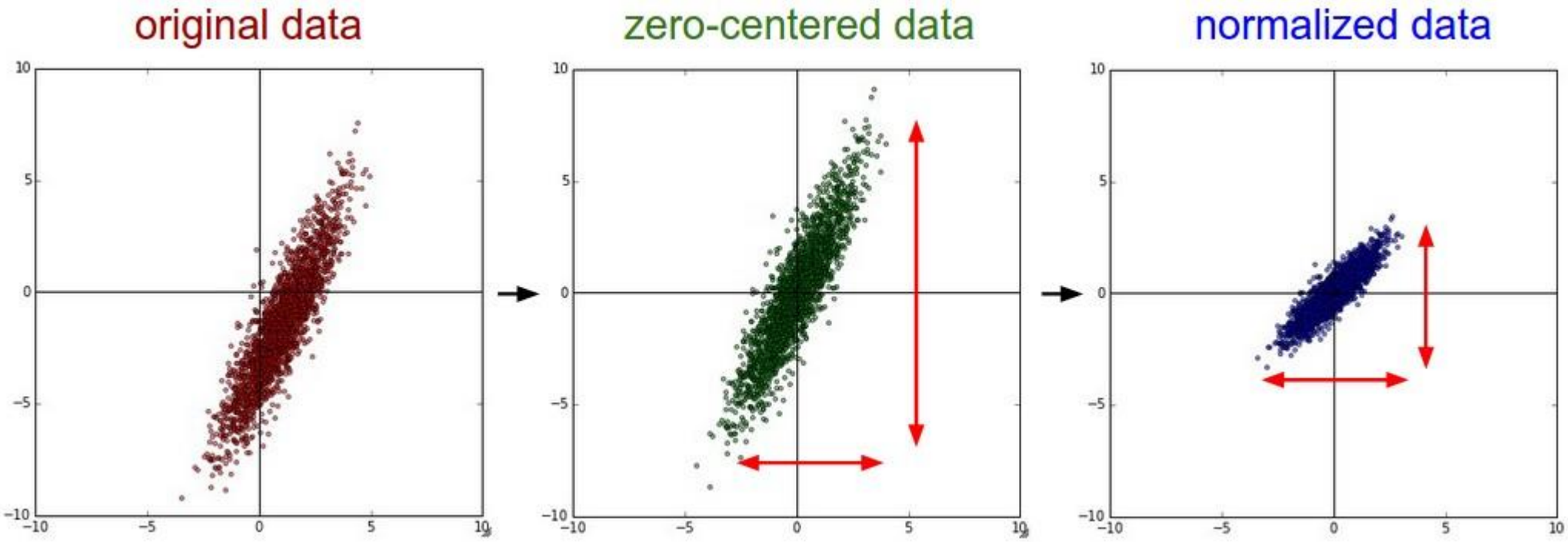
Lesson 5

Scaling and Normalization

Kush Kulshrestha

Topics

1. Scaling of Data
2. Data Normalization
3. Difference between Scaling and Normalization
4. Min-max scalar and Box-Cox Transformation



Scaling of data

- Variable scaling requires taking values that span a specific range and representing them in another range.
- The standard method is to scale variables to [0,1].
- This may introduce various distortions or biases into the data but the distribution or shape remains same.
- Depending on the modeling tool, scaling variable ranges can be beneficial or sometimes even required.

One way of doing this is:

Linear Scaling Transform

- First task in this scaling is to determine the minimum and maximum values of variables.
- Then applying the transform:

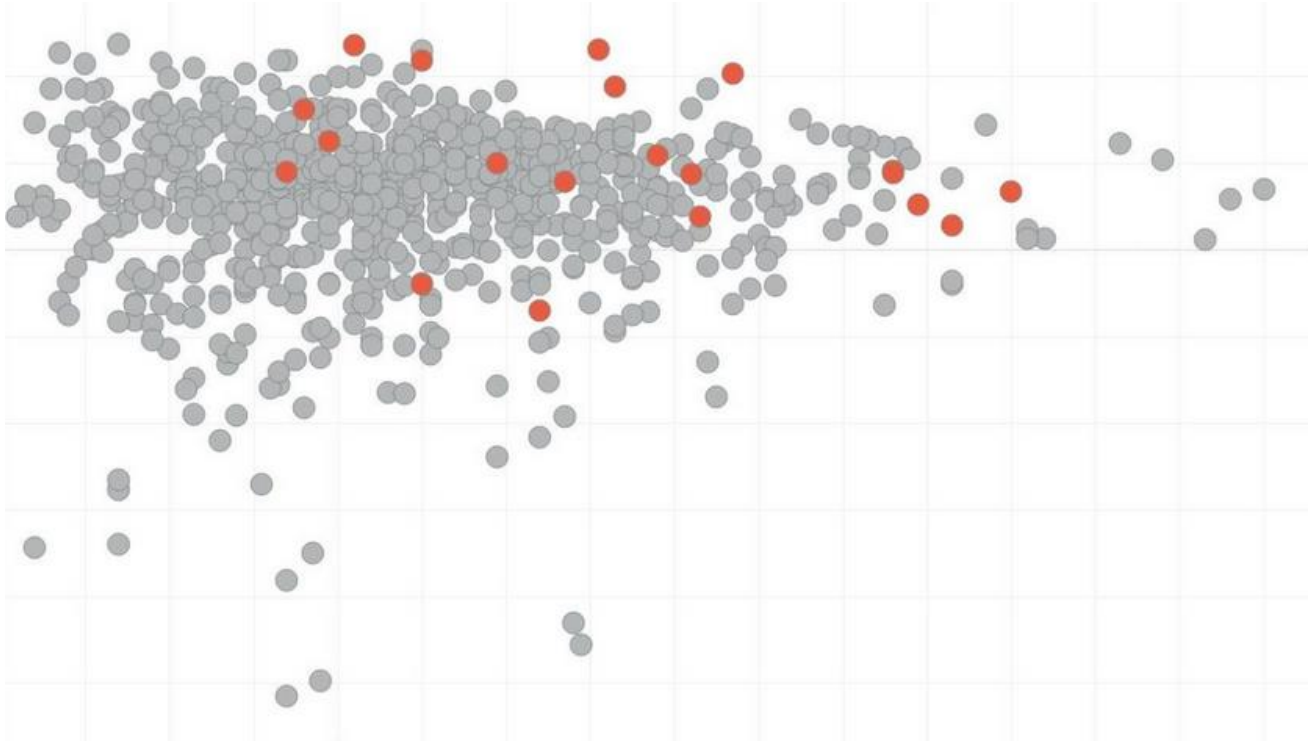
$$(x - \min\{x_1, x_N\}) / (\max\{x_1, x_N\} - \min\{x_1, x_N\})$$

- This introduces no distortion to the variable distribution.
- Has a one-to-one relationship between the original and scaled values.

Scaling of data

Out of Range values

- In data preparation, the **data used is only a sample of the population**.
- Therefore, it **is not certain** that the **actual minimum and maximum values** of the variable have been **discovered** when scaling the ranges.
- If some values that turn up later in the mining process are outside of the limits discovered in the sample, they are called *out-of-range* values.



Scaling of data

Dealing with Out of Range values

- After range scaling, all variables should be in the range of [0,1].
- Out-of-range values, however, have values like -0.2 or 1.1 which can cause unwanted behavior.

Solution 1: Ignore that the range has been exceeded.

- Most modeling tools have (at least) some capacity to handle numbers outside the scaling range.
- Important question to ask: Does this affect the quality of the model?

Solution 2: Exclude the out of range instances.

- One problem is that reducing the number of instances reduces the confidence that the sample represents the population.
- Another problem: **Introduction of bias**. Out-of-range values may occur with a certain pattern and ignoring these instances removes samples according to a pattern introducing distortion to the sample

Scaling of data

Dealing with Out of Range values

Solution 3: Clip the out of range values

- If the value is greater than 1, assign 1 to it. If less than 0, assign 0.
- This approach assumes that out-of-range values are somehow equivalent with range limit values.
- Therefore, the information content on the limits is distorted by projecting multiple values into a single value.
- This also introduces some bias.

Solution 4: Making room for out of range values

- The linear scaling transform provides an undistorted normalization but suffers from out-of-range values.
- Therefore, we should modify it to somehow include also values that are out of range.
- Most of the population is inside the range so for these values the normalization should be linear.
- The solution is to ***reserve some part of the range for the out-of-range values*****.**
- Reserved amount of space depends on the confidence level of the sample:
  e.g. - 98% confidence linear part is [0.01, 0.99]

Scaling of data

Dealing with Out of Range values

Squashing the out of range values

- Now the problem reduces to fitting the out-of-range values into the space left for them.
- The greater the difference between a value and the range limit, the less likely any such value is found.
- Therefore, the transformation should be such that as the distance to the range grows, the smaller the increase towards one or decrease towards zero.
- One possibility is to use functions of the form $y = 1/x$ and attach them to the ends of the linear part.

Its difficult to carry out the scaling in pieces depending on the nature of the data point.
We can do all of the above steps using one function called **Softmax scaling.**

Scaling of data

Dealing with Out of Range values

Softmax Scaling –

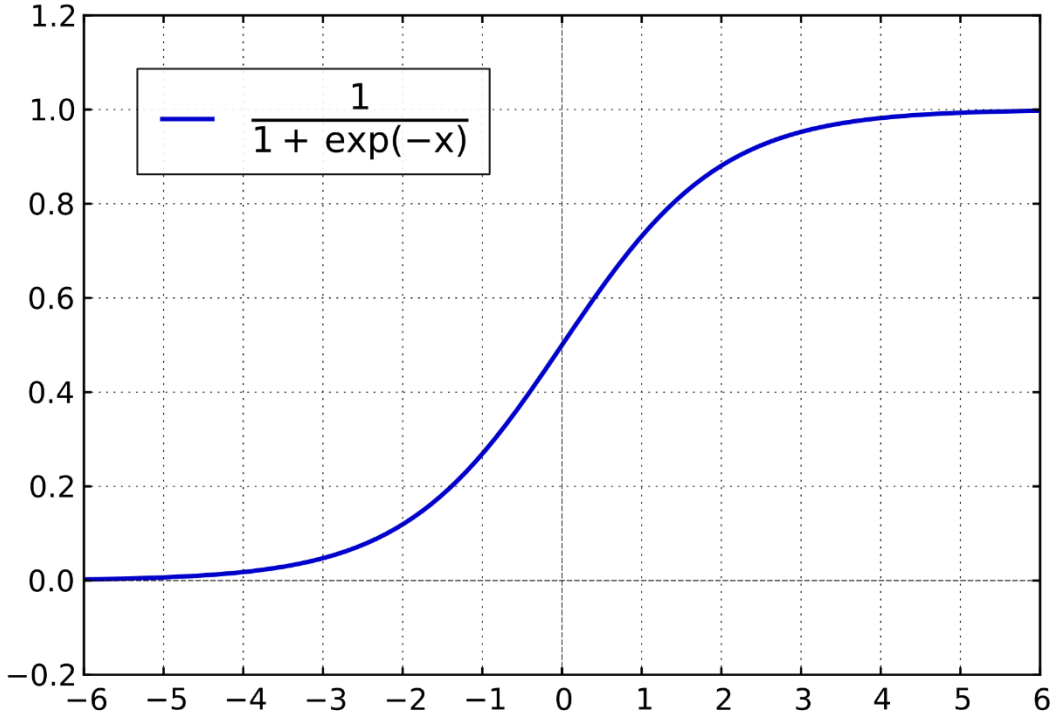
- The extent of the linear part can be controlled by one parameter.
- The space assigned for out-of-range values can be controlled by the level of uncertainty in the sample.
- Non identical values have always different normalized values.

Softmax scaling is based on the **logistic function:**

$$y = 1 / (1 + e^{-x})$$

Where y is the scaled value and x is the input value.

- The logistic function transforms the original range of [-∞,∞] to [0,1] and also has a linear part on the transform.



Scaling of data

Min-Max Scaler In scikit-learn

```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_train_norm = mms.fit_transform(X_train)
X_test_norm = mms.transform(X_test)
```

```
X_train_norm[0]
```

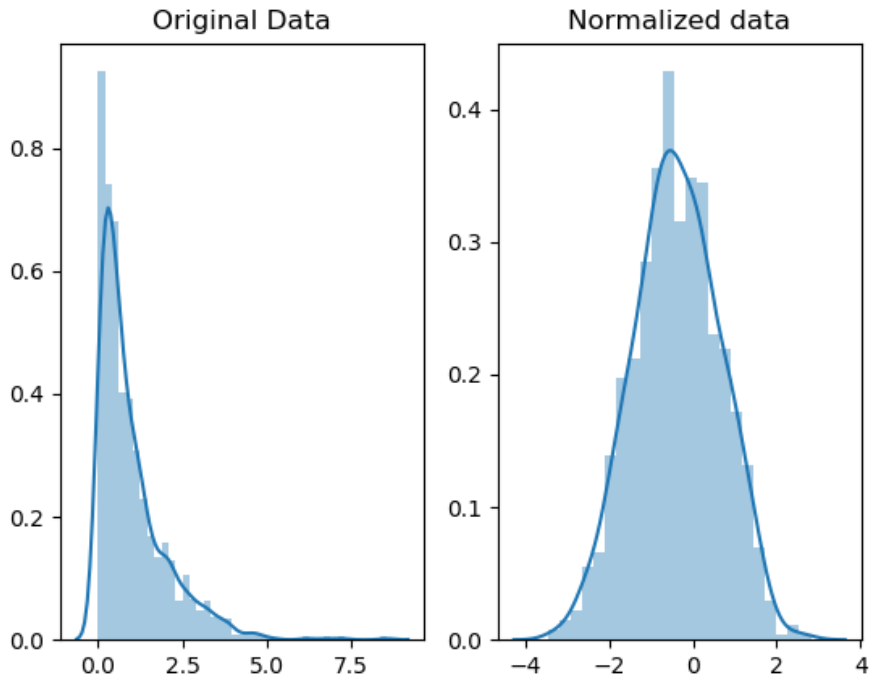
```
array([ 0.72043011,  0.20378151,  0.53763441,  0.30927835,  0.33695652,
        0.54316547,  0.73700306,  0.25      ,  0.40189873,  0.24068768,
        0.48717949,  1.        ,  0.5854251 ])
```

```
X_test_norm[0]
```

```
array([ 0.72849462,  0.16386555,  0.47849462,  0.29896907,  0.52173913,
        0.53956835,  0.74311927,  0.13461538,  0.37974684,  0.4364852 ,
        0.32478632,  0.70695971,  0.60566802])
```

Data Normalization

- Applying a function to each data point z in the data: $y_i = f(z_i)$
- Unlike scaling, not only the data is distorted, but the shape or distribution of the data changes as well.
- The point of normalization is to change your observations so that they can be described as a normal distribution.
- In general, you'll only want to normalize your data if you're going to be using a machine learning or statistics technique that assumes your data is normally distributed.
- Or before using any technique containing “Gaussian” in its name.

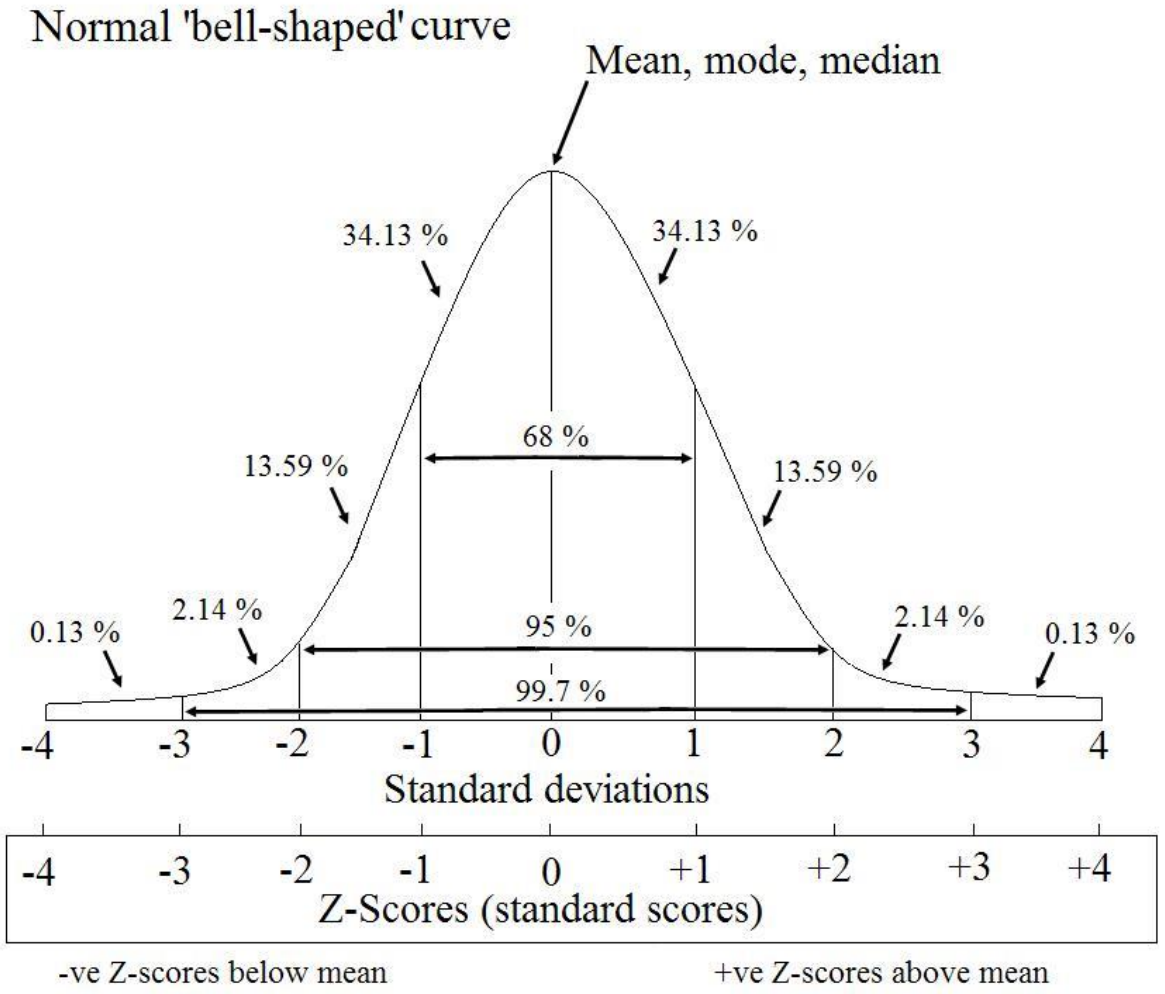


Data Normalization

Data Standardization

- Also called z-scores.
- The terms *normalization* and *standardization* are sometimes used interchangeably.
- *Standardization* transforms data to have a mean of zero and a standard deviation of 1.
- Done by subtracting the mean and dividing by the standard deviation for each data point.

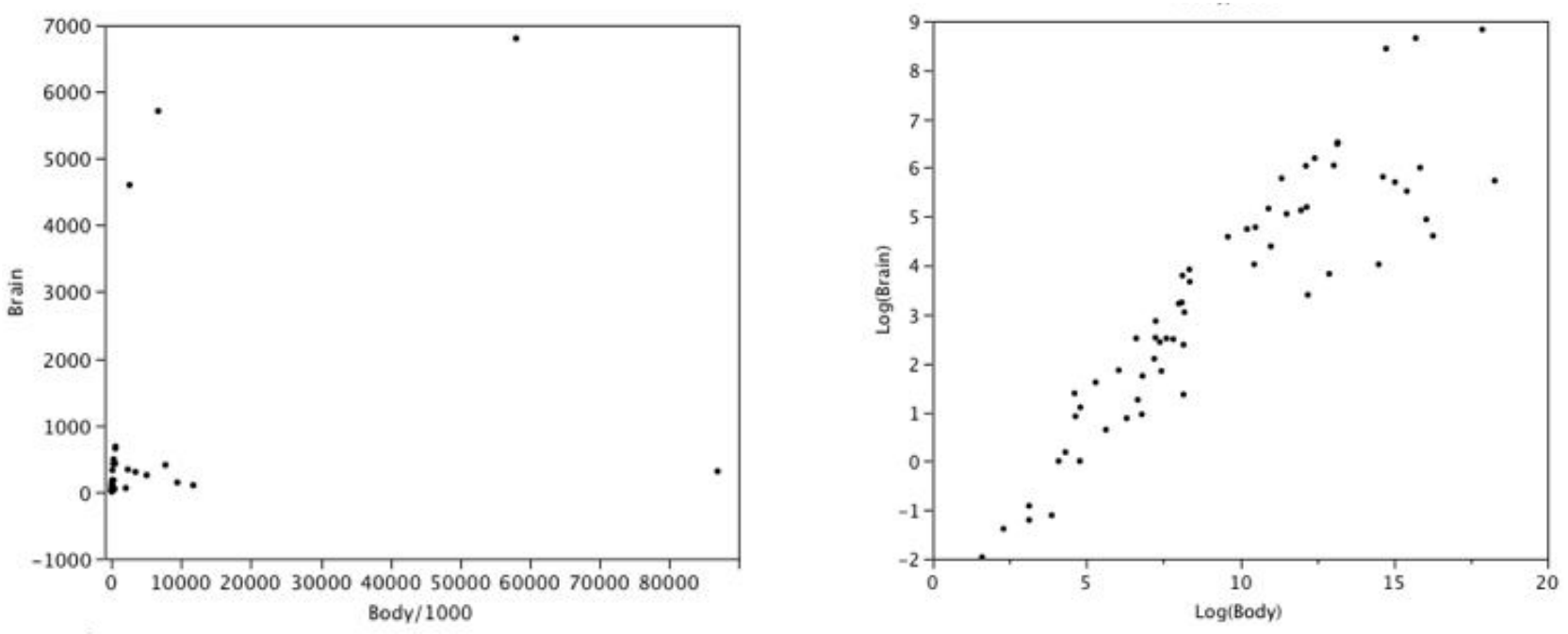
$$X_{i,\,1\sigma} = \frac{X_i - \bar{X}_S}{\sigma_{X,\,S}}$$



Data Normalization

Log Transformation

- The log transformation can be used to make highly skewed distributions less skewed.
- This can be valuable both for making patterns in the data more interpretable and for helping to meet the assumptions of inferential statistics.
- Figure shows an example of how a log transformation can make patterns more visible. Both graphs plot the brain weight of animals as a function of their body weight. The raw weights are shown in the left panel; the log-transformed weights are plotted in the right panel.



Data Normalization

Box-Cox Transformation

The Box-Cox transformation of the variable x is also indexed by λ, and is defined as:

$$x'_\lambda = \frac{x^\lambda - 1}{\lambda}$$

- Box-Cox transformations cannot handle negative values.
- One way to deal with this is by adding the minimum data point value to add the data points.
- We want to use the same lambda generated from the training set of box cox transformation in the test set.
- Another way to write this equation:

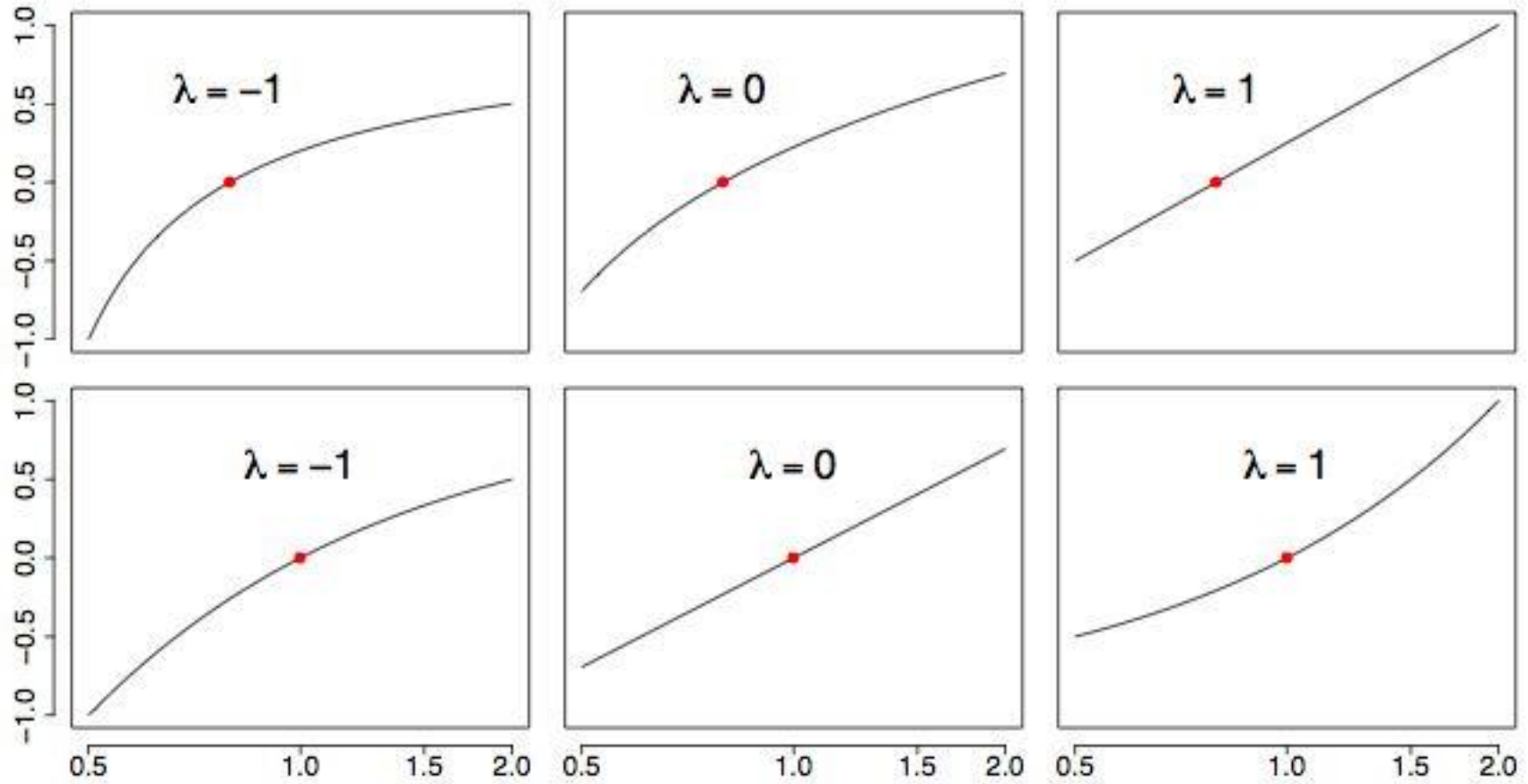
$$x'_\lambda = \frac{e^{\lambda \log(x)} - 1}{\lambda} \approx \frac{\left(1 + \lambda \log(x) + \frac{1}{2}\lambda^2 \log(x)^2 + \cdots\right) - 1}{\lambda} \to \log(x)$$

as λ̃ turns to be 0

Data Normalization

Box-Cox Transformation

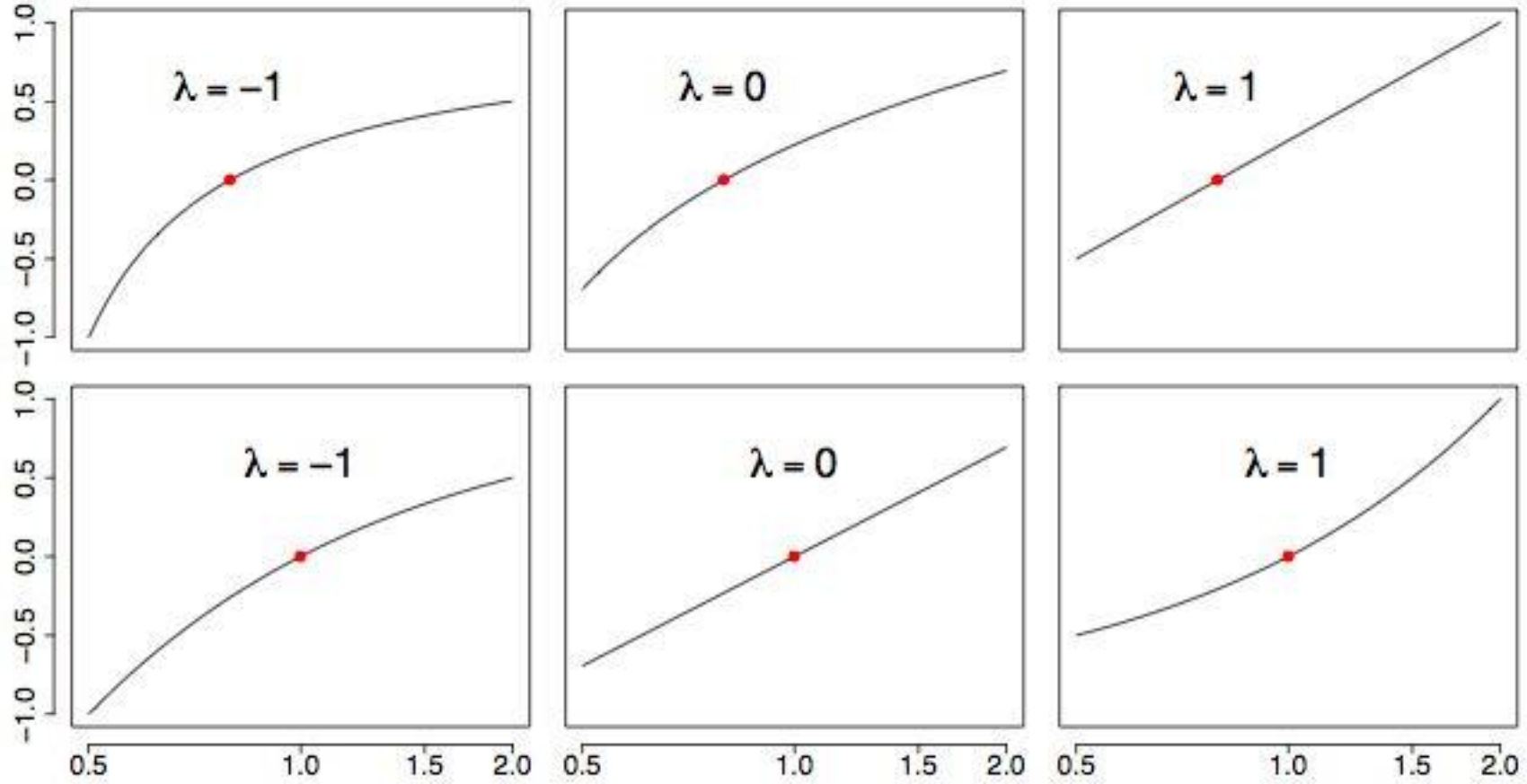
Notice with this definition of x'_λ that x = 1 always maps to the point x'_λ = 0 for all values of λ. To see how the transformation works, look at the examples in Figure.



Data Normalization

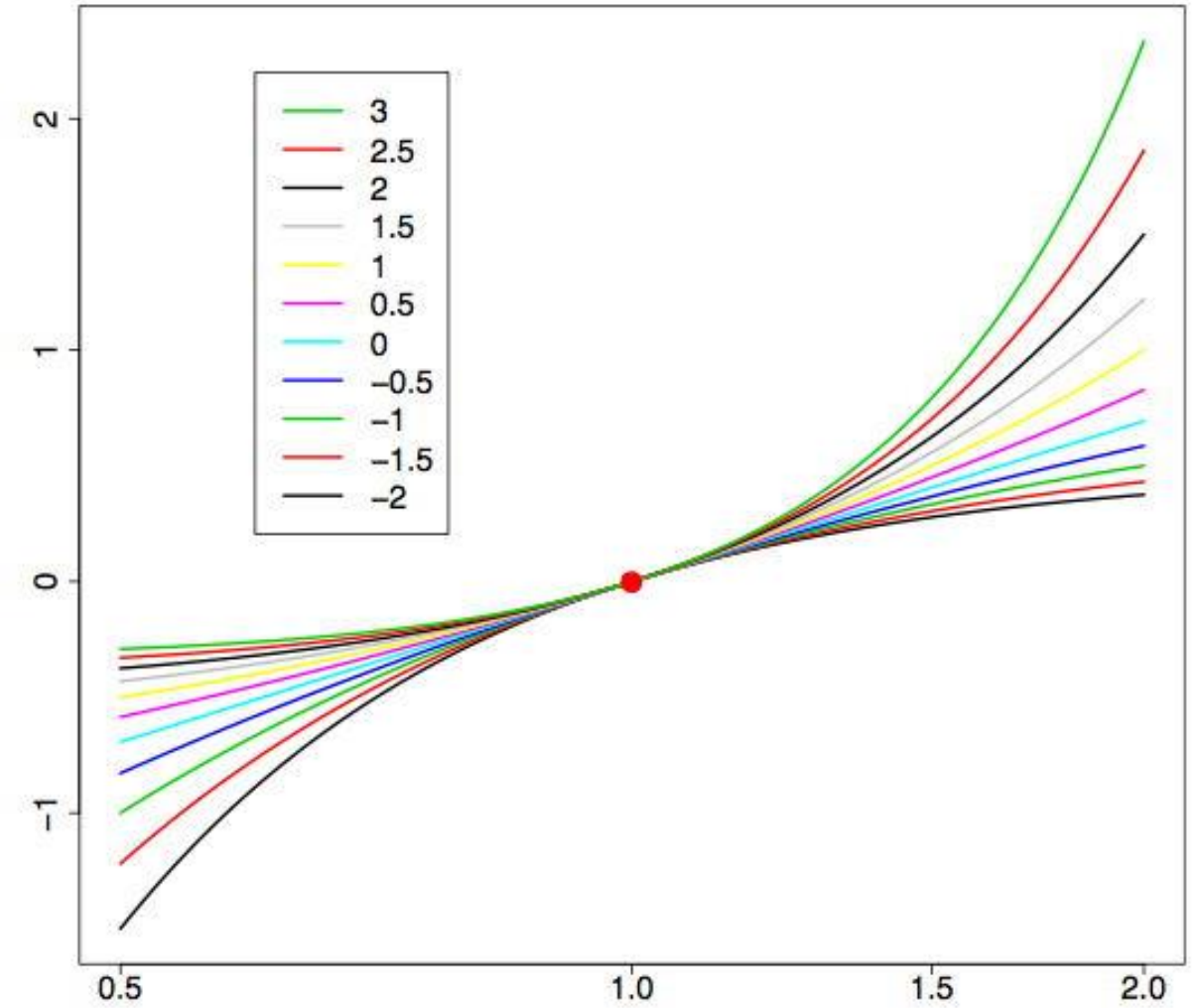
Box-Cox Transformation

In the top row, the choice λ = 1 simply shifts x to the value x−1, which is a straight line. In the bottom row (on a semi-logarithmic scale), the choice λ = 0 corresponds to a logarithmic transformation, which is now a straight line. We superimpose a larger collection of transformations on a semi-logarithmic scale in Figure 2.



Data Normalization

Box-Cox Transformation



Data Normalization

Why do we need normalization?

1. Some algorithms need normally distributed data as input, otherwise the results are not reliable.
2. Easy comparison of values.
3. Interpretation of results makes more sense.
4. Objective function works faster in some cases if the data is normalized, i.e., speed of the algorithm increases.
5. Can create complex features that may improve the model (or make it non-linear)