NORMALIZATION

Units

Real variables x (random or continous) are sampled from points on the real number line

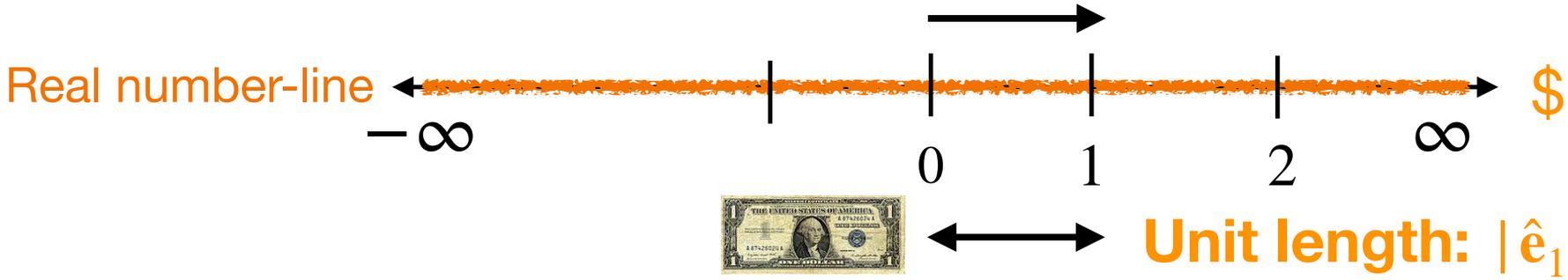
- A “unit” defines what “length"=1 means in the mathematical "space" in which the variable exists (i.e. what is a unit length)

$x =$ **distance (unit=meters)** $\quad x = 1 \rightarrow$ 

$x =$ **bank balance (unit=USD)** $\quad x = 1 \rightarrow$ 

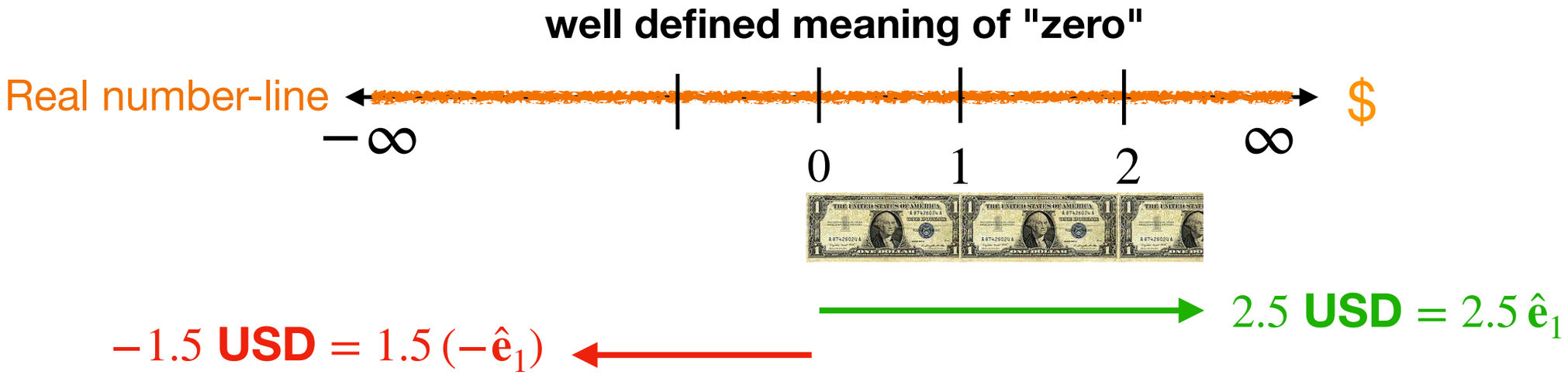
- Typically it is helpful to introduce a “unit vector” with length 1 which points in the positive direction
  - Recall that vectors have both magnitude (length) and directions

unit vector: $\hat{e}_1 = 1$



- Any point x along the number line is a scalar multiple of the unit vector

well defined meaning of "zero"



Function transformations

Transformation Rules for Functions		
Function Notation	Type of Transformation	Change to Coordinate Point
f(x) + d	Vertical translation up d units	(x, y) → (x, y + d)
f(x) – d	Vertical translation down d units	(x, y) → (x, y – d)
f(x + c)	Horizontal translation left c units	(x, y) → (x – c, y)
f(x – c)	Horizontal translation right c units	(x, y) → (x + c, y)
–f(x)	Reflection over x-axis	(x, y) → (x, –y)
f(–x)	Reflection over y-axis	(x, y) → (–x, y)
af(x)	Vertical stretch for |a|> 1	(x, y) → (x, ay)
	Vertical compression for 0 < |a| < 1	
f(bx)	Horizontal compression for |b| > 1	$(x, y) \rightarrow \left(\frac{x}{b}, y\right)$
	Horizontal stretch for 0 < |b| < 1	

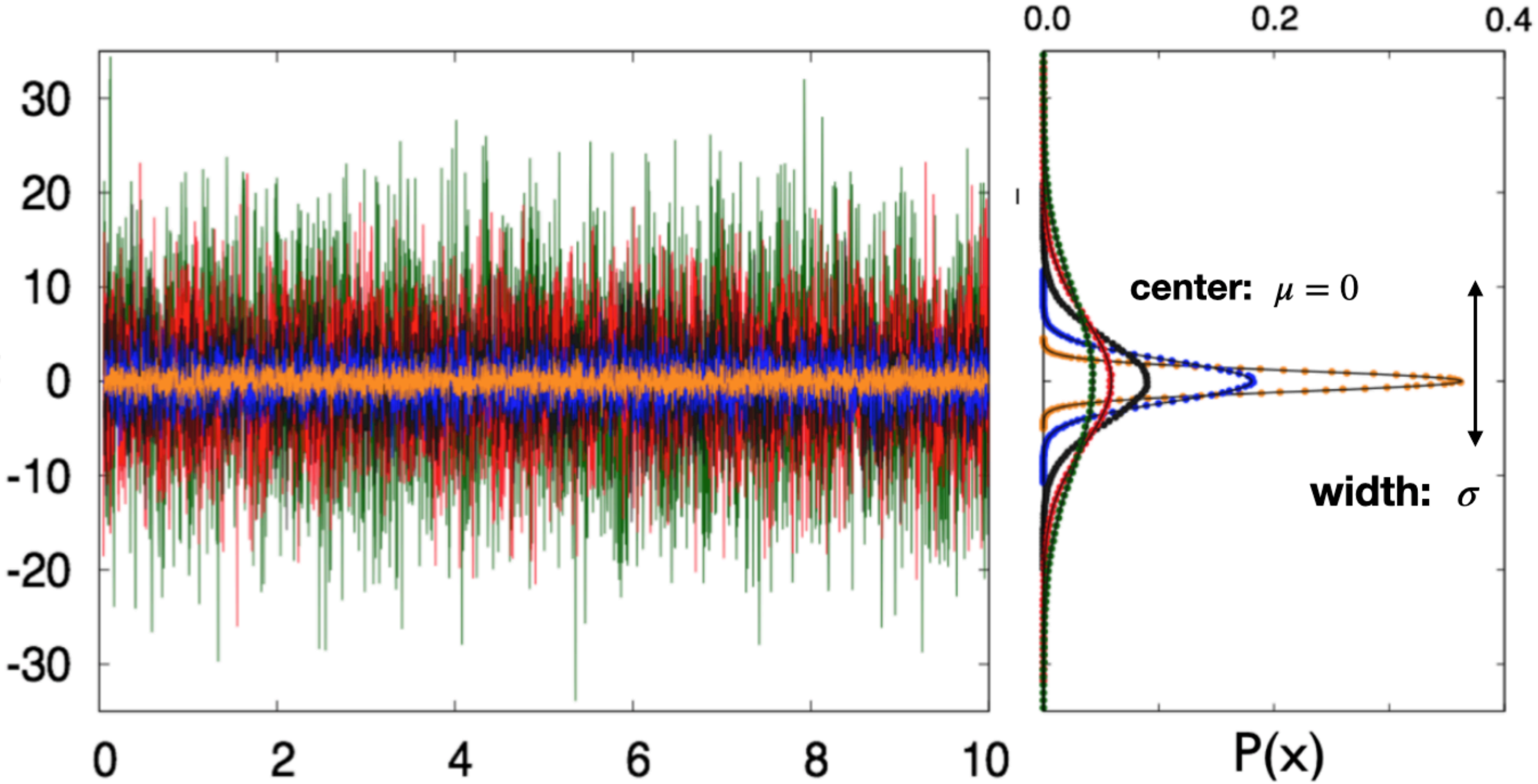
Rigid

Non-Rigid

https://www.onlinemathlearning.com/parent-functions.html

Parameterizing a function: $y = f(x \mid \mathbf{p}) = Af\left(\frac{x - x_o}{w}\right) + S$

Normal distributions examples



Normal distributions with same mean increasing standard deviation

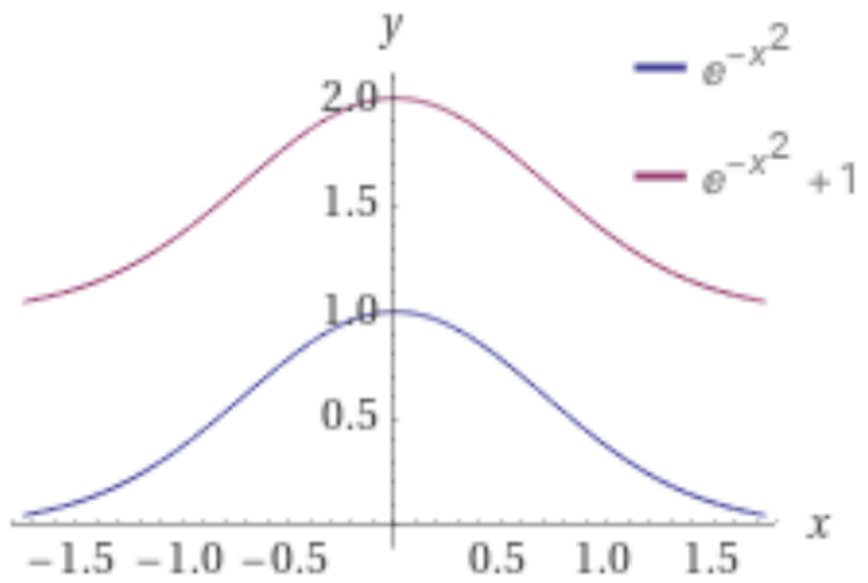
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

two parameters: (μ, σ)

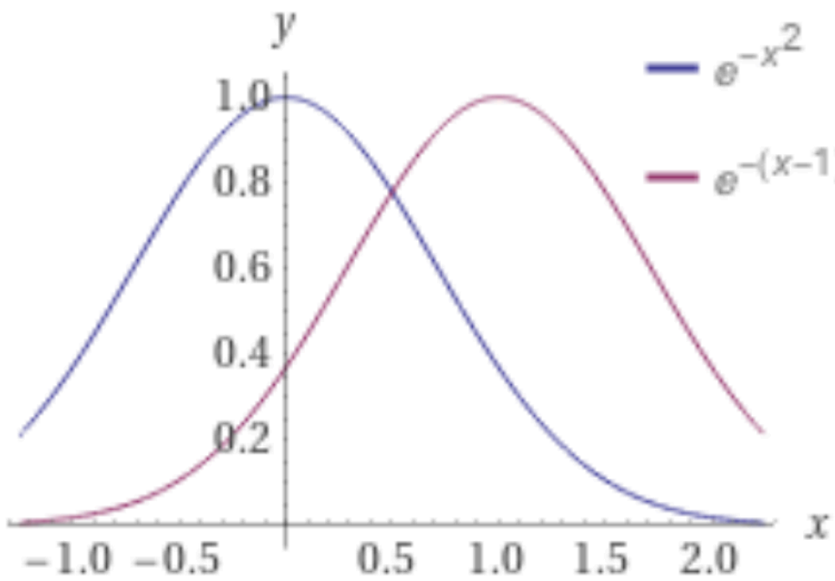
Example: Gaussian

$$y = f(x \mid \mathbf{p}) = A f\left(\frac{x - x_o}{w}\right) + S$$

addition outside (rigid verticle shift): $f(x) + a$



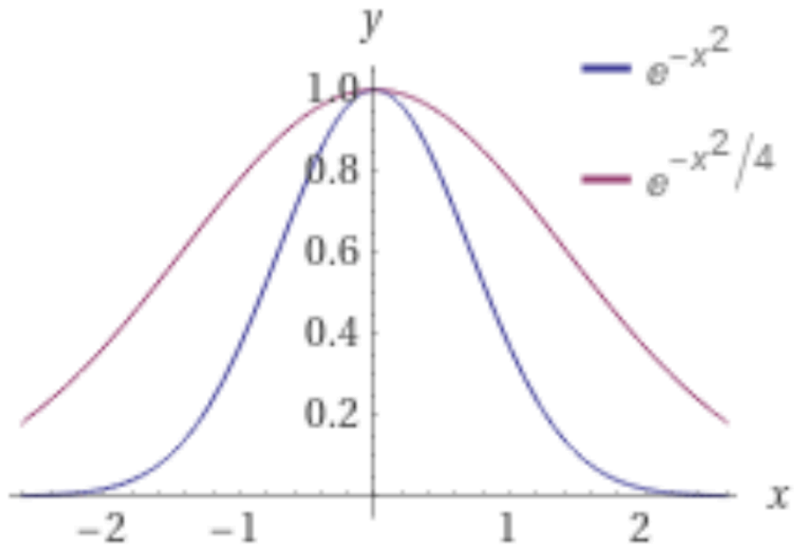
addition inside (rigid horizontal shift): $f(x - a)$



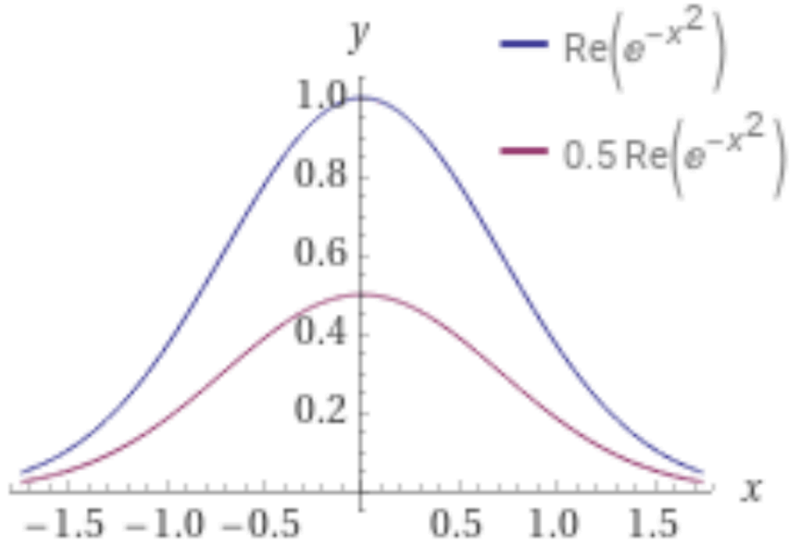
$$f(x) = e^{-x^2}$$

Parent function:

multiplication inside (non-rigid horizontal scaling): $f(a\,x)$

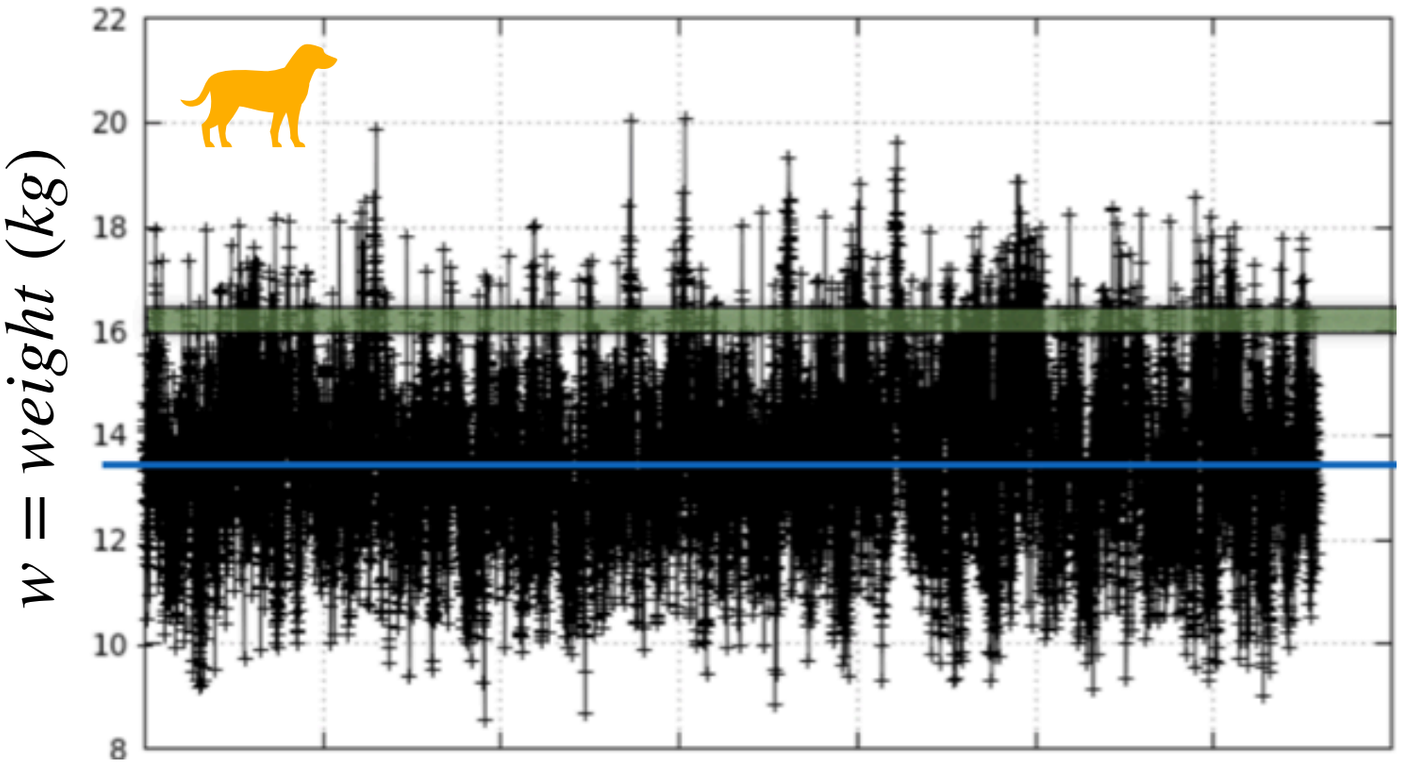


multiplication outside (non-rigid verticle scaling): $a f(x)$

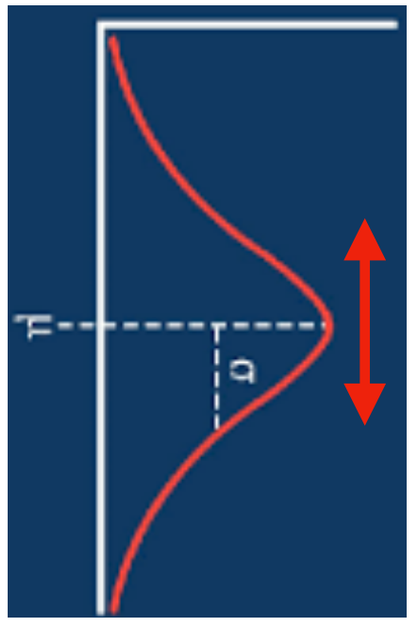


Standard score

Original Un-normalized space



$$p(w) = \frac{1}{\sigma_w\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\mu_w}{\sigma_w}\right)}$$



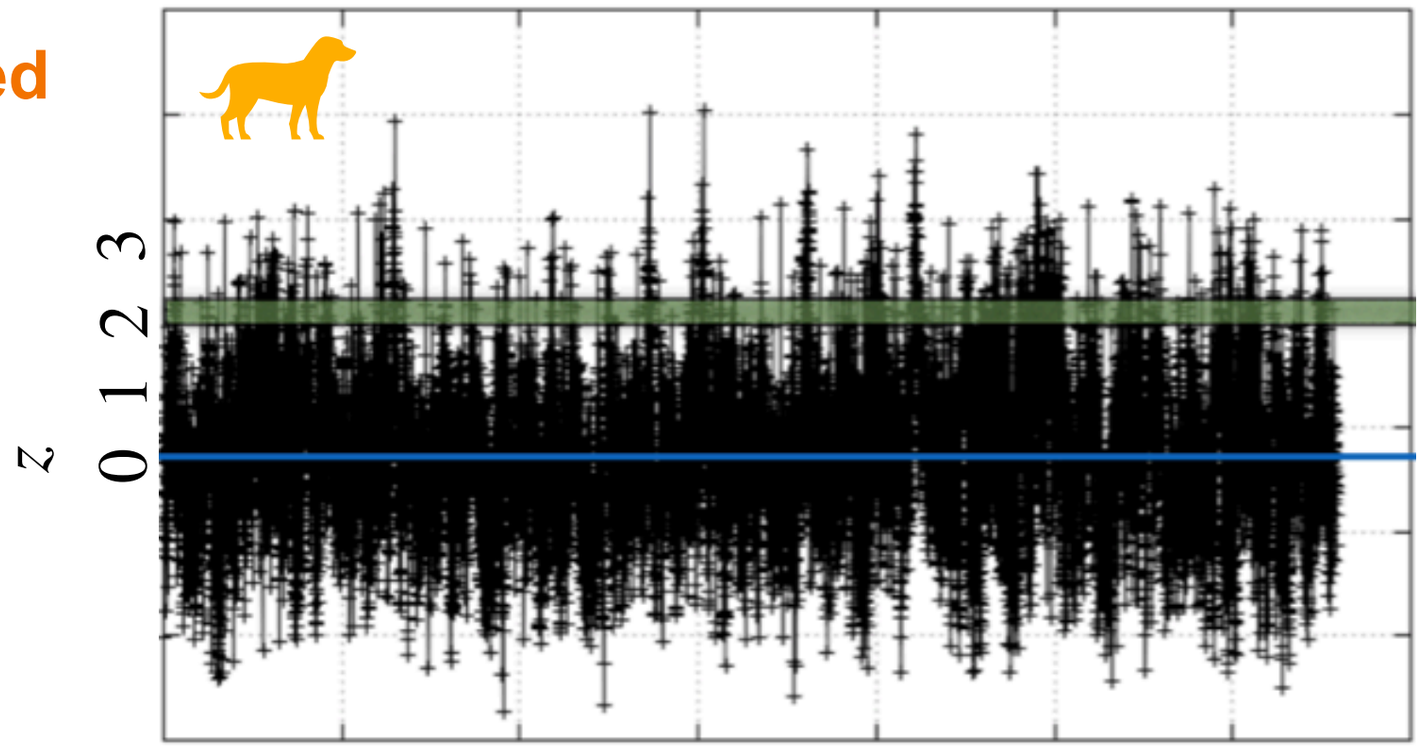
$\mu_w \approx 13.5\,kg$

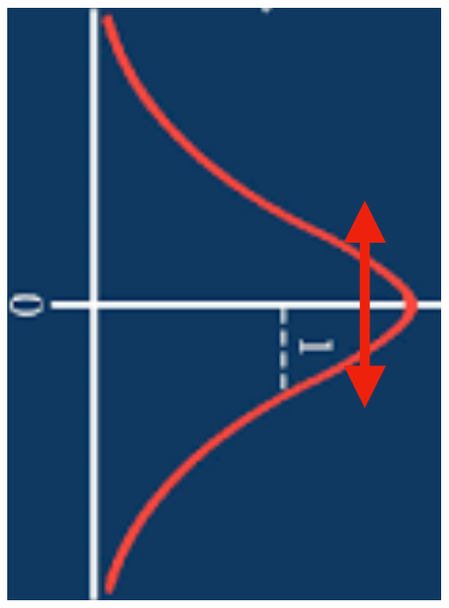
$\sigma_w \approx 2.2\,kg$

$$z = \frac{w - \mu_w}{\sigma_w}$$

Normalization

Normalized space

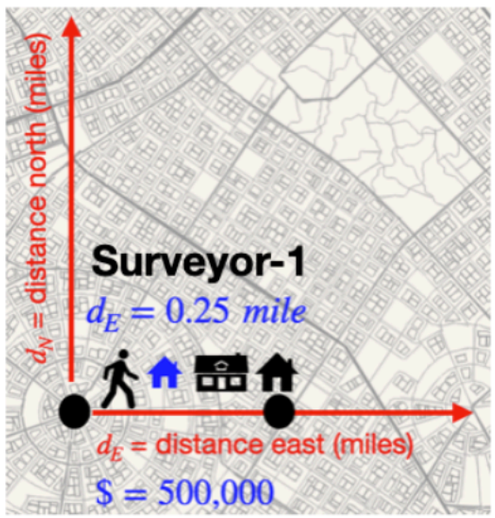




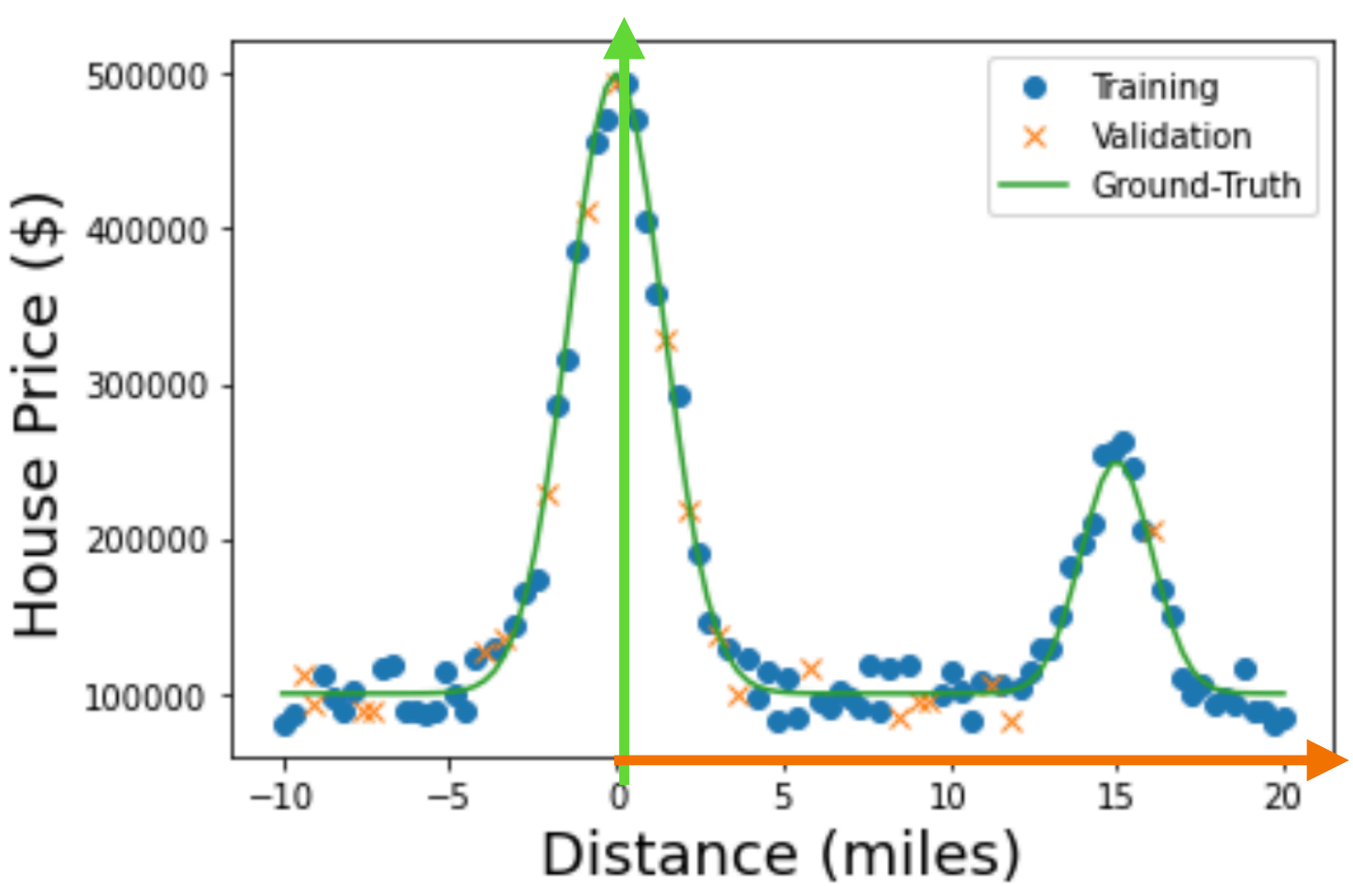
$\mu_z \approx 0$

$\sigma_z \approx 1$

Example



Gaussian



$$\hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix}$$

$\sigma_y \quad \mu_y$

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix}$$

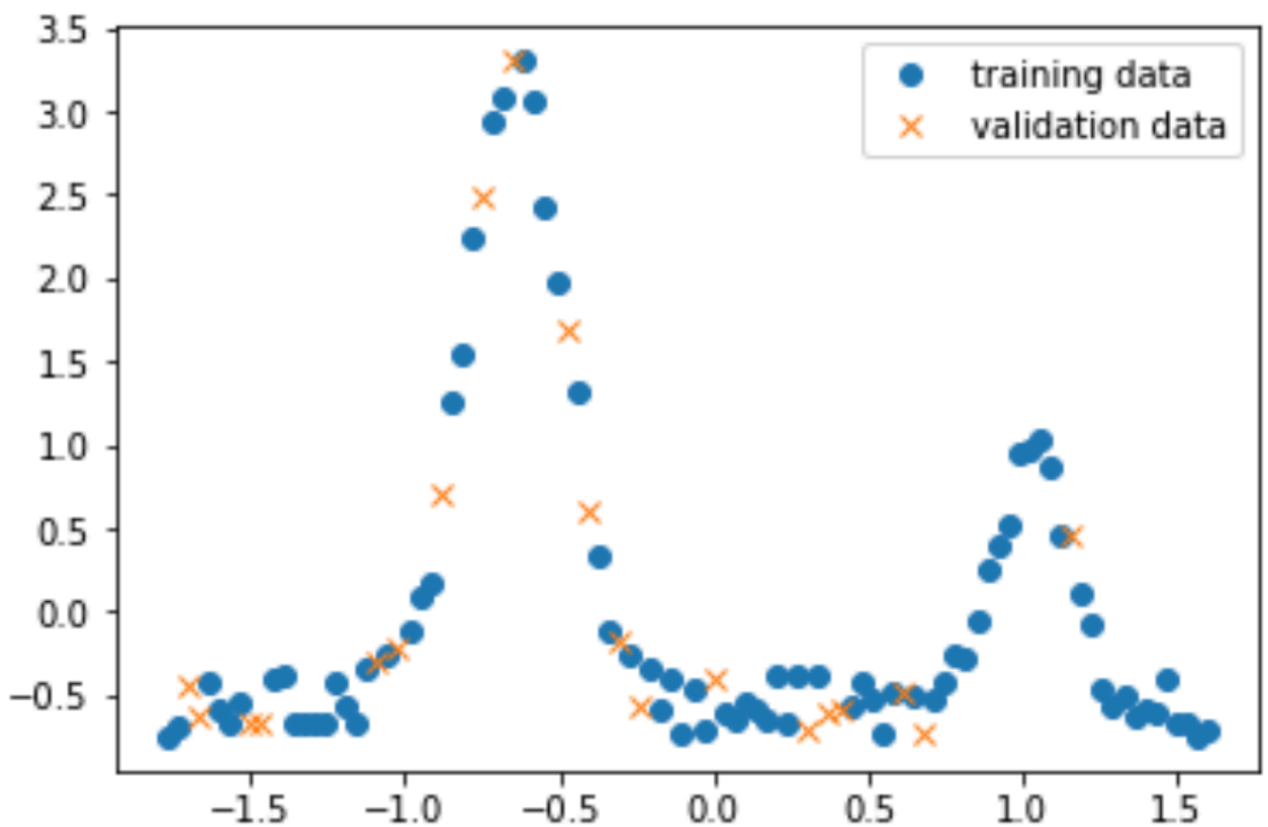
$\sigma_x \quad \mu_x$

Normalize

$$x_i \to \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \qquad y_i \to \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

```
ux=np.mean(x_train)
sx=np.std(x_train)
x_train = (x_train-ux)/sx
y_train = (y_train-uy)/sy
```

Transformed that data Into a dimensionless space that runs from roughly -3 to 3 (i.e. unit-less space)

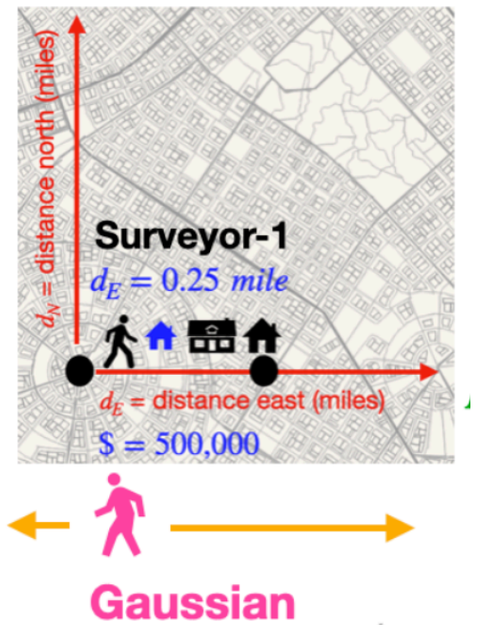


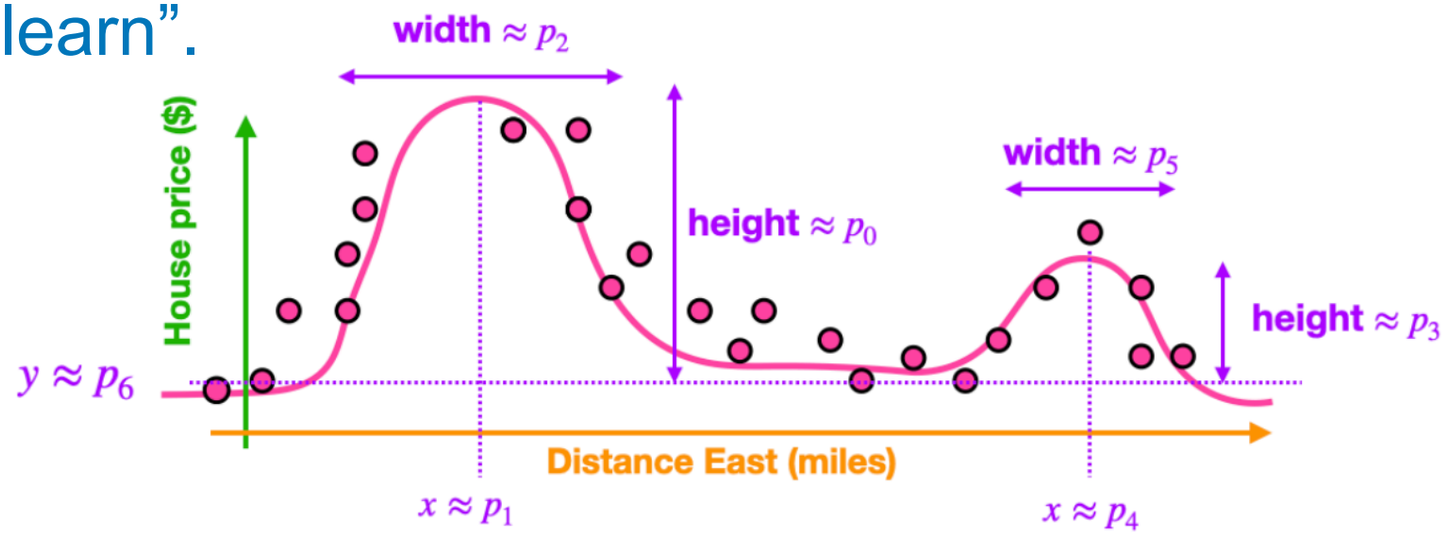
Why do we normalize?

1) Normalization forces the data into a standardized space, centered near the origin (0,0), with a data range ~ (-3 to 3). This effectively "gets rid" of units. However, this range depends on the statistical distribution of the original data

2) Normalization makes visualization more effective, since all numerical variables span roughly the same range

3) In this normalized space it is MUCH easier for machine learning algorithms to "learn" the data. For example, it parametric modeling we try to "learn" a curve that fits the data by adjusting (i.e. learning) the model parameters. In the normalized space, these parameters will also tend to be in the range (-3 to 3), and therefore easier to "learn".





$$M(x \mid \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \mid \mathbf{p}) = p_0 e^{-\left(\frac{x-p_1}{p_2}\right)^2} + p_3 e^{-\left(\frac{x-p_4}{p_5}\right)^2} + p_6$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6)$$

"The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model"

https://developers.google.com/machine-learning/data-prep/transform/normalization

Scaling to range

Source: https://developers.google.com/machine-learning/data-prep/transform/normalization

Recall from MLCC that **scaling** means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range:

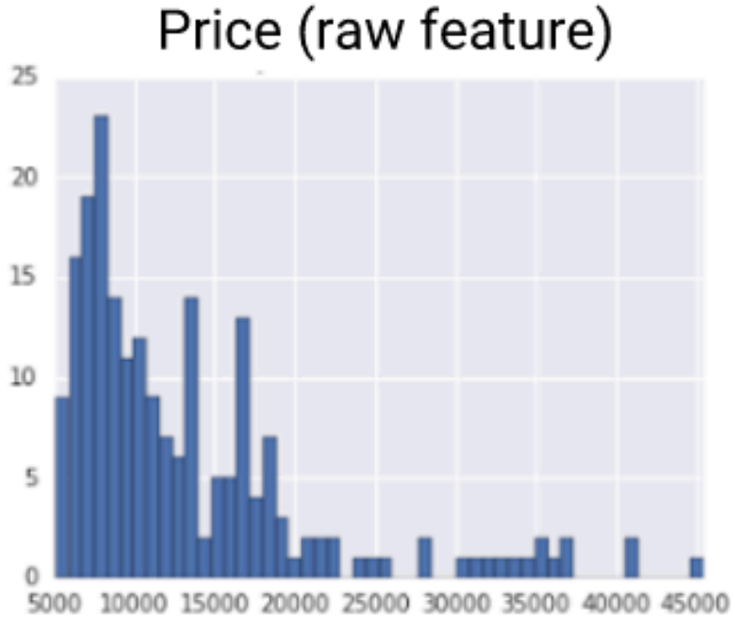
$$x' = (x - x_{min})/(x_{max} - x_{min})$$

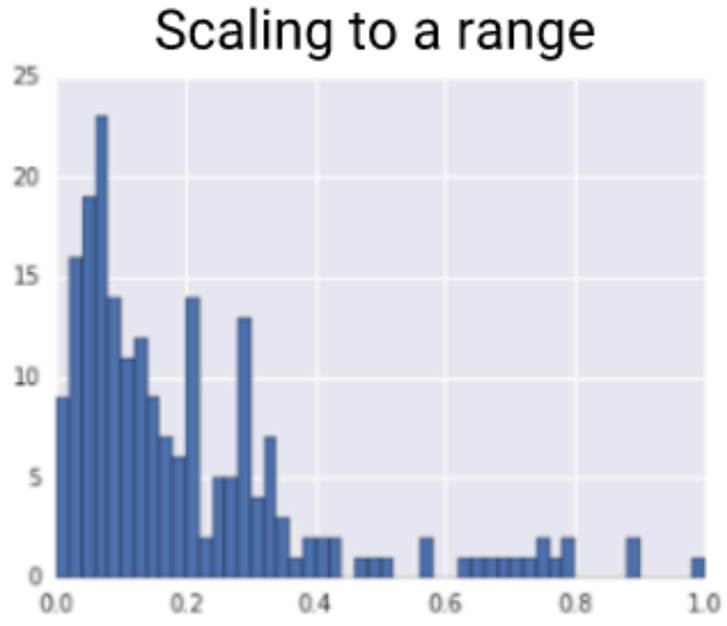
Scaling to a range is a good choice when both of the following conditions are met:

- You know the approximate upper and lower bounds on your data with few or no outliers.
- Your data is approximately uniformly distributed across that range.

A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.

In contrast, you would *not* use scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.





Feature clipping

Source: https://developers.google.com/machine-learning/data-prep/transform/normalization

If your data set contains extreme outliers, you might try feature clipping, which caps all feature values above (or below) a certain value to fixed value. For example, you could clip all temperature values above 40 to be exactly 40.

You may apply feature clipping before or after other normalizations.

Formula: Set min/max values to avoid outliers.

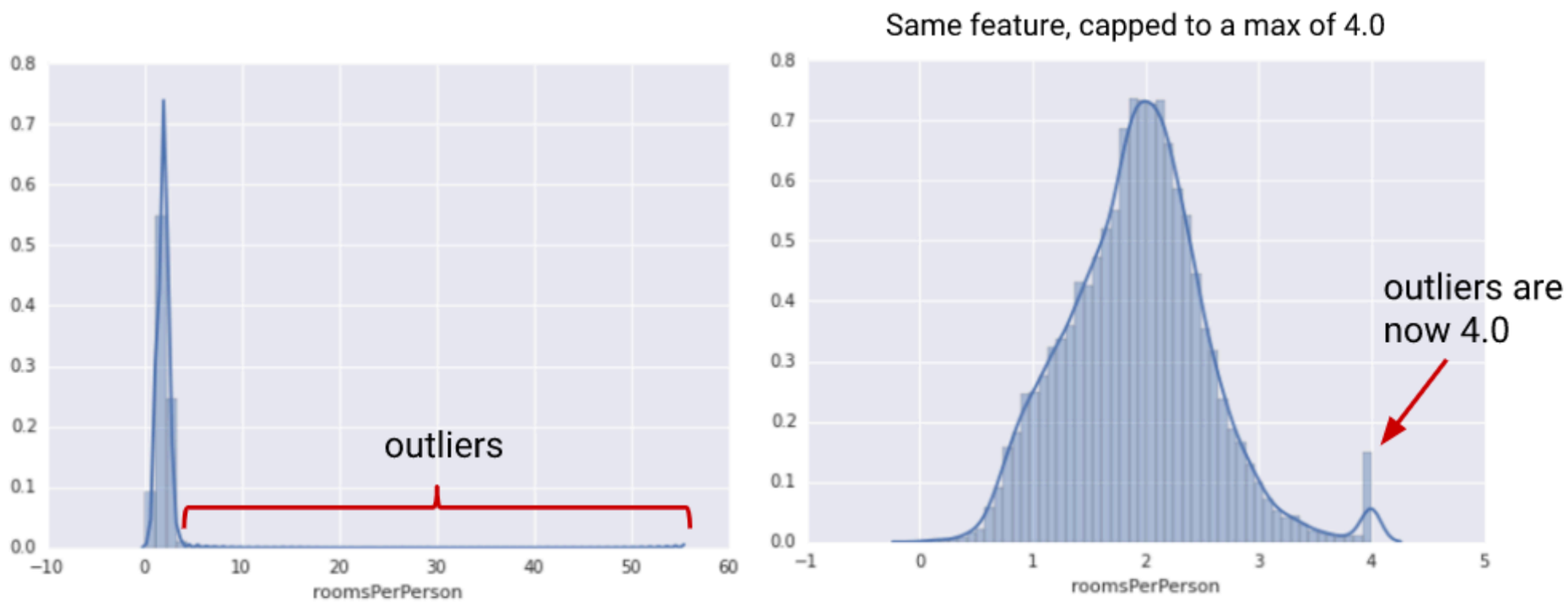


Figure 2. Comparing a raw distribution and its clipped version.

Another simple clipping strategy is to clip by z-score to +-Nσ (for example, limit to +-3σ). Note that σ is the standard deviation.

Z-score normalization

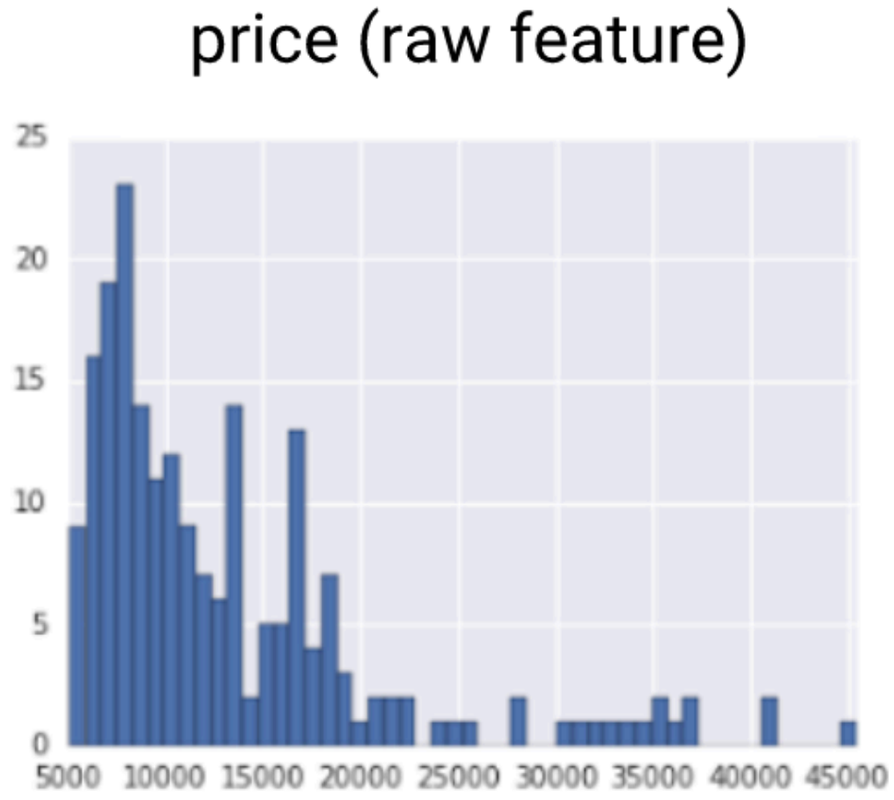
Source: https://developers.google.com/machine-learning/data-prep/transform/normalization

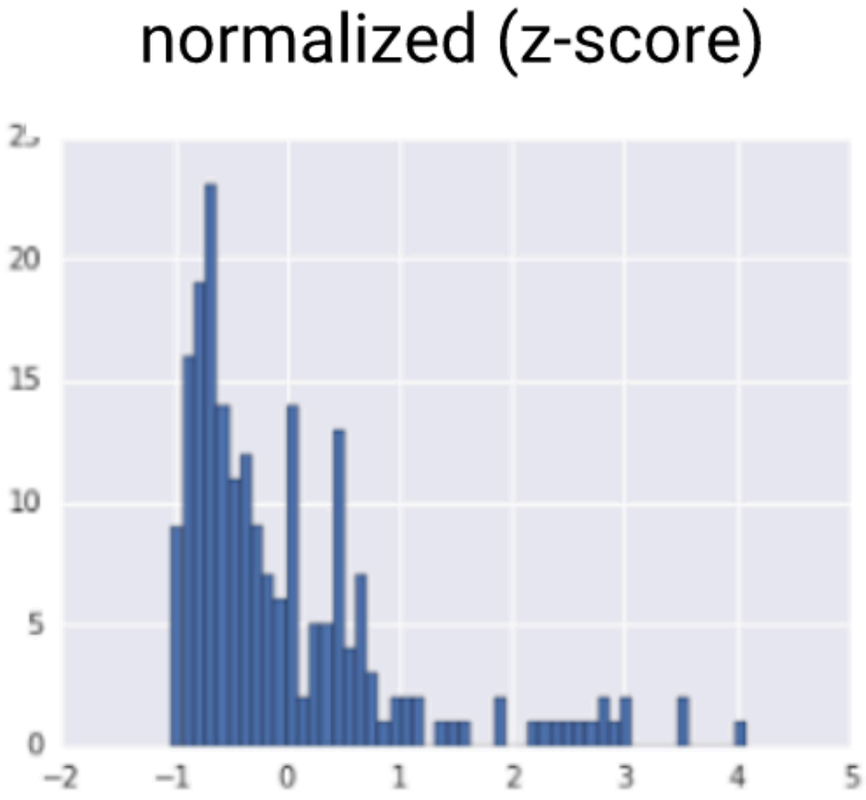
Z-score is a variation of scaling that represents the number of standard deviations away from the mean. You would use z-score to ensure your feature distributions have mean = 0 and std = 1. It's useful when there are a few outliers, but not so extreme that you need clipping.

The formula for calculating the z-score of a point, *x*, is as follows:

$$x' = (x - \mu)/\sigma$$

★ **Note:** μ is the mean and σ is the standard deviation.





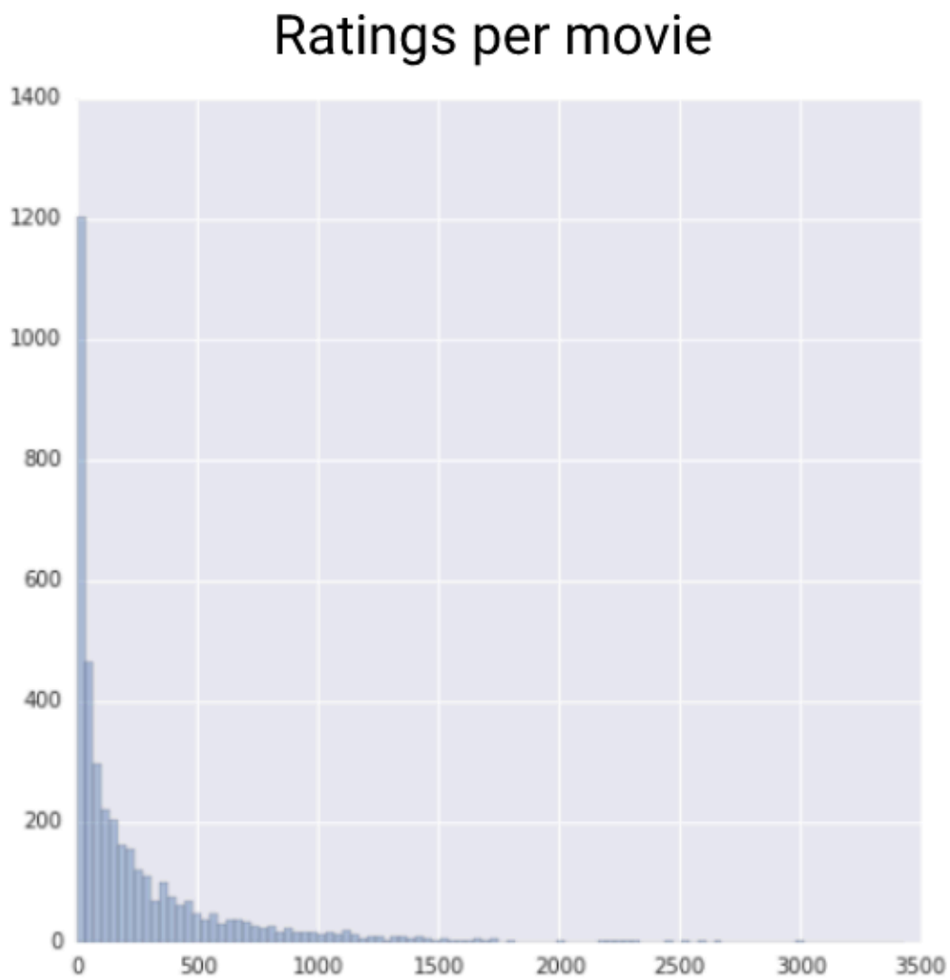
Log scaling

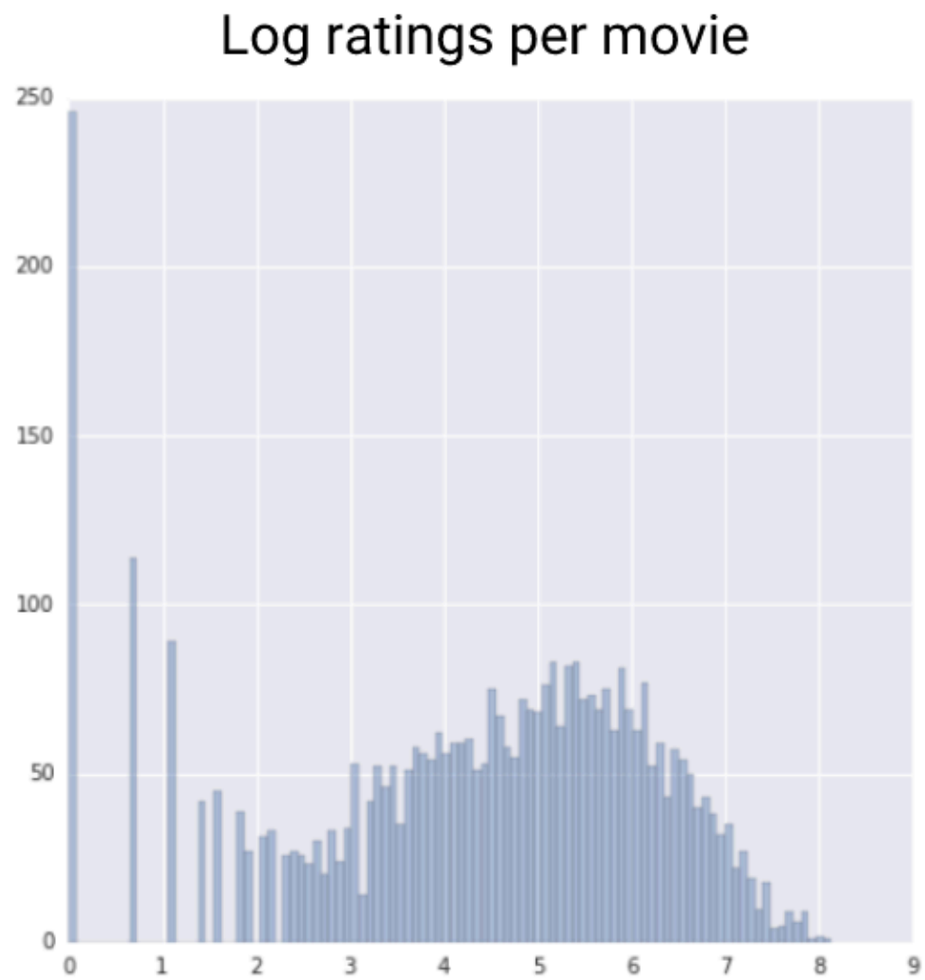
Source: https://developers.google.com/machine-learning/data-prep/transform/normalization

Log scaling computes the log of your values to compress a wide range to a narrow range.

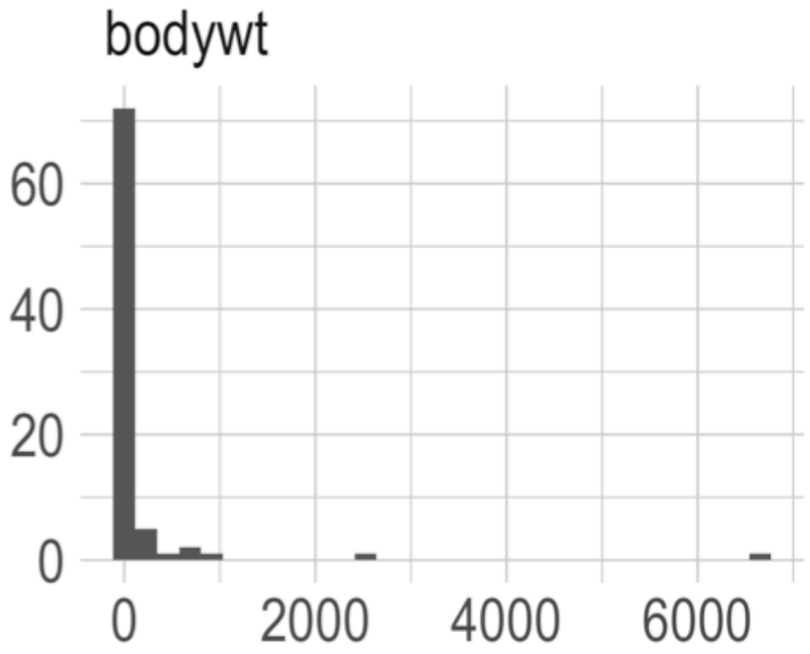
$$x' = log(x)$$

Log scaling is helpful when a handful of your values have many points, while most other values have few points. This data distribution is known as the *power law* distribution. Movie ratings are a good example. In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.





Coordinate transform

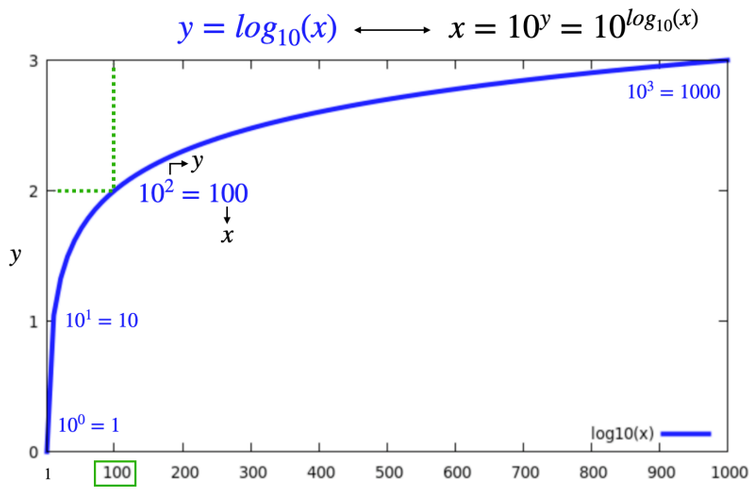


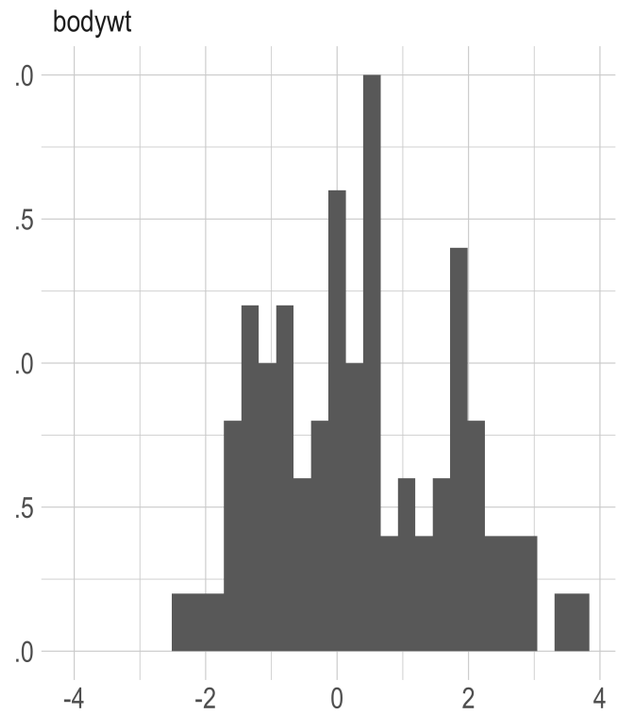
heavily skewed distribution

Indications are that we should transform the two weight variables, since they are both very skew and long-tailed. The usual transformation for this kind of distribution is a *log-transform*

Transform

$$x \rightarrow x' = log_{10}(x)$$

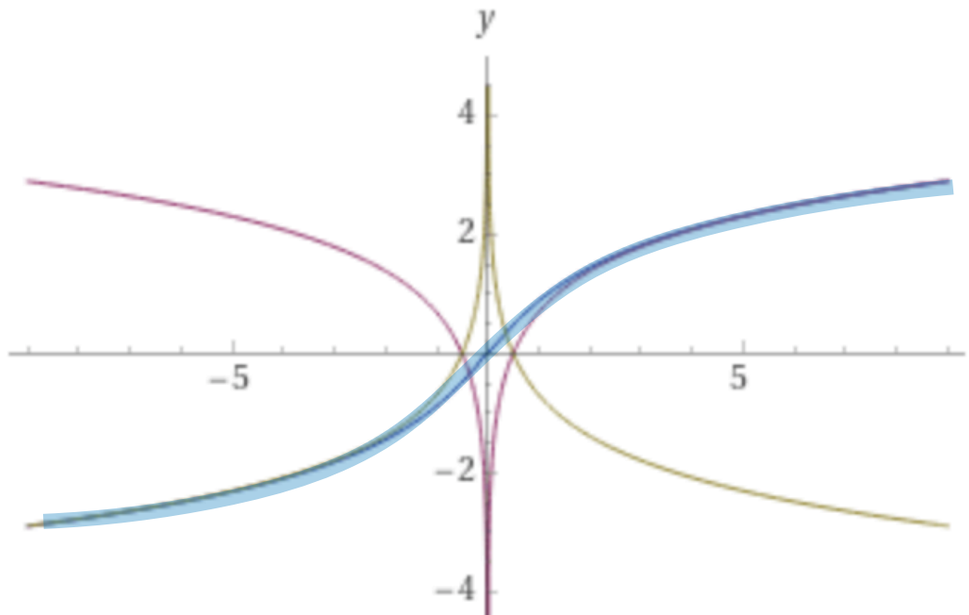




more symetric/even distribution

NOTE: INPUTS TO log(x) MUST BE POSITIVE NUMBERS

Alternative transform: $x \rightarrow x' = sinh^{-1}(x)$



plot	$\sinh^{-1}(x)$
	$\log(2x)$
	$-\log(2x)$

Logarithms

$$y = log_{10}(x) \longleftrightarrow x = 10^y = 10^{log_{10}(x)}$$

$10^3 = 1000$

$10^2 = 100$

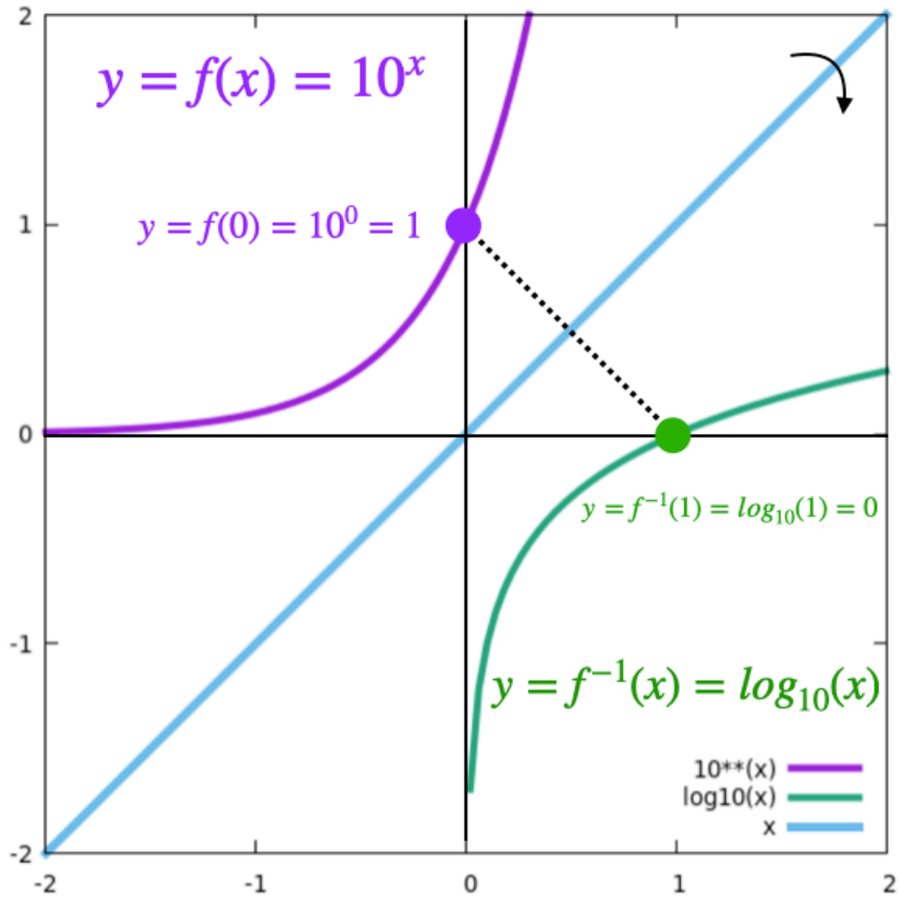
$10^1 = 10$

$10^0 = 1$

y

x

$log_{10}(10^0) = 0$ $log_{10}(10^1) = 1$

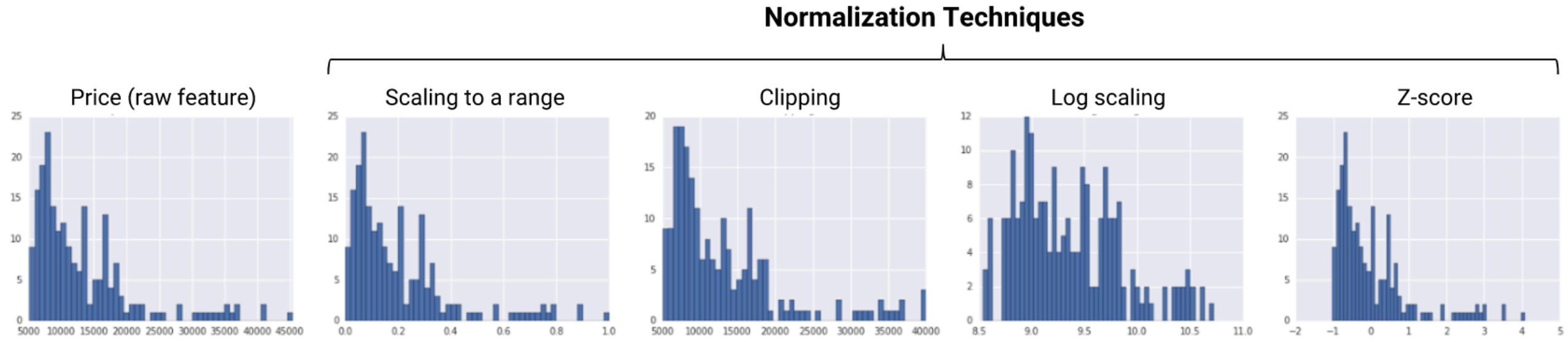


logarithm = inverse of exponential

set ytic 1; set key bottom; set xrange [1:1000]; plot log10(x) lw 4 lc rgb 'blue'

Summary-1

Source: https://developers.google.com/machine-learning/data-prep/transform/normalization



The best normalization technique is one that empirically works well, so try new ideas if you think they'll work well on your feature distribution.

Normalization Technique	Formula	When to Use
Linear Scaling	$x' = (x - x_{min})/(x_{max} - x_{min})$	When the feature is more-or-less uniformly distributed across a fixed range.
Clipping	if x > max, then x' = max. if x < min, then x' = min	When the feature contains some extreme outliers.
Log Scaling	x' = log(x)	When the feature conforms to the power law.
Z-score	x' = (x - μ) / σ	When the feature distribution does not contain extreme outliers.

Summary-2

Normalization of a vector is the process of making the vector have a unit length (i.e. $|\mathbf{v}| = 1$)

$$\mathbf{v}_{norm} = \frac{\mathbf{v}}{|\mathbf{v}|}$$

We can do a similar thing with a vector of data (e.g. h=heights with units ft).

This is done using the following normalization process

$$\mathbf{h}_{norm} = \frac{\mathbf{h} - \mu_h}{\sigma_h}$$

Where μ_h, σ_h are the mean (center) and standard deviation (width) of the height distribution.

This process makes a new vector $\mathbf{h}_{norm}$ which is a dimensionless quanity, meaning that it doesn't have units.

The units cancel in the division arising during the normalization equation.

This also forces the data into a standard range of roughly [-3 to 3] while still preserving the "shape" of the data.

Often, when training machine learning models, it is important to normalize the data first.

The model will have a much easier time "fitting" if every input is in a standard range of [-3 to 3]

You can always "un-do" the normalization and re-assign the units by algebraically re-arranging the formula.

$$\mathbf{h} = \mathbf{h}_{norm}\sigma_h + \mu_h$$

Un-normalize by un-doing the operations and re-scaling/shifting back to the original space