

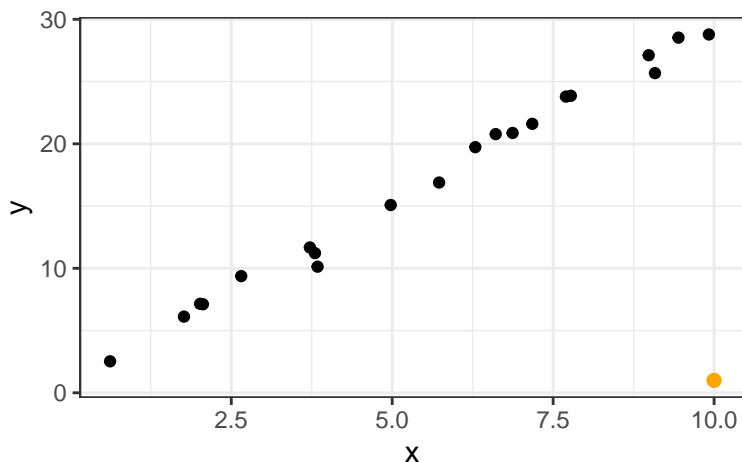
Exercise 1: HRO in sklearn

Throughout the lecture, we will frequently use the Python package **sklearn** and its descendants, providing an integrated ecosystem for all common machine learning tasks. Let's recap the HRO principle and see how it is reflected in **sklearn**. An overview of the most important objects and their usage, illustrated with numerous examples, can be found at <https://scikit-learn.org/stable/index.html>.

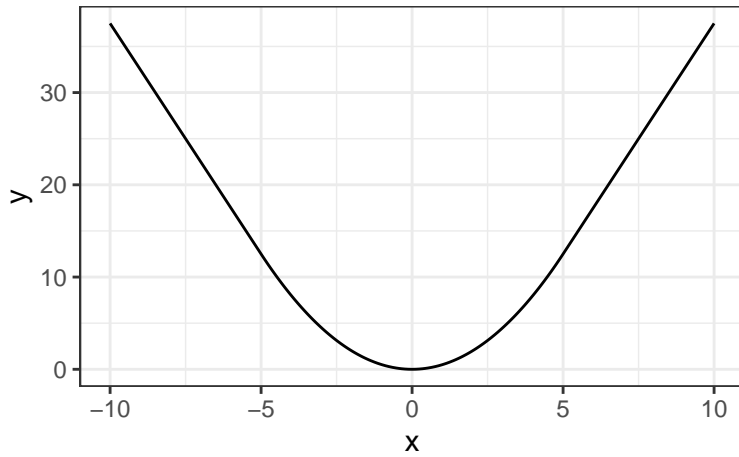
- a) How are the key concepts (i.e., hypothesis space, risk and optimization) you learned about in the lecture videos implemented in **sklearn**?
- b) Have a look at the function `from sklearn.datasets import load_iris`. What attributes does this resulting `utils.Bunch` object store?
- c) Pick an **sklearn** module for classification or regression of your choice. What are the different settings for this learner?
(Hint: Import the specific module and use `get_params()` to see all available settings.)

Exercise 2: Loss Functions for Regression Tasks

In this exercise, we will examine loss functions for regression tasks somewhat more in depth.



- a) Consider the above linear regression task. How will the model parameters be affected by adding the new outlier point (orange) if you use
 - i) L_1 loss
 - ii) L_2 lossin the empirical risk? (You do not need to actually compute the parameter values.)



b) The second plot visualizes another loss function popular in regression tasks, the so-called *Huber loss* (depending on $\epsilon > 0$; here: $\epsilon = 5$). Describe how the Huber loss deals with residuals as compared to $L1$ and $L2$ loss. Can you guess its definition?

c) Derive the least-squares estimator, i.e., the solution to the linear model when using $L2$ loss, analytically via

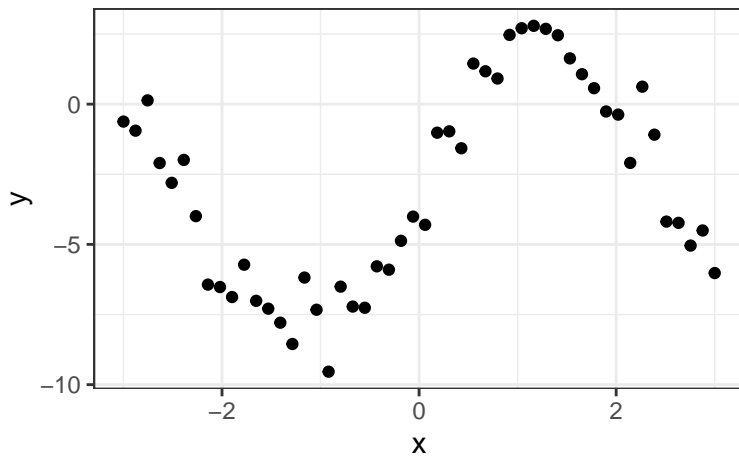
$$\hat{\theta} = \arg \min_{\theta \in \Theta} \|\mathbf{y} - \mathbf{X}\theta\|_2^2.$$

Exercise 3: Polynomial Regression

Assume the following (noisy) data-generating process from which we have observed 50 realizations:

$$y = -3 + 5 \cdot \sin(0.4\pi x) + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, 1)$.



- We decide to model the data with a cubic polynomial (including intercept term). State the corresponding hypothesis space.
- Demonstrate that this hypothesis space is simply a parameterized family of curves by plotting in `Python` curves for 3 different models belonging to the considered model class.
- State the empirical risk w.r.t. θ for a member of the hypothesis space. Use $L2$ loss and be as explicit as possible.
- We can minimize this risk using gradient descent. In order to make this somewhat easier, we will denote the transformed feature matrix, containing x to the power from 0 to 3, by $\tilde{\mathbf{X}}$, such that we can express our model by $\tilde{\mathbf{X}}\theta$ (note that the model is still linear in its parameters, even if \mathbf{X} has been transformed in a non-linear manner!). Derive the gradient of the empirical risk w.r.t θ .

- e) Using the result from d), state the calculation to update the current parameter $\theta^{[t]}$.
- f) You will not be able to fit the data perfectly with a cubic polynomial. Describe the advantages and disadvantages that a more flexible model class would have. Would you opt for a more flexible learner?

Exercise 4: Predicting abalone

We want to predict the age of an abalone using its longest shell measurement and its weight.

See <https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/> for more details.

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
abalone = pd.read_csv(url, sep=",",
                      names=["sex", "longest_shell", "diameter", "height", "whole_weight",
                             "shucked_weight", "visceral_weight", "shell_weight", "rings"])

abalone = abalone[["longest_shell", "whole_weight", "rings"]]
```

- a) Plot LongestShell and WholeWeight on the x - and y -axis, respectively, and color points according to Rings.

Using sklearn:

- b) Initiate a linear regression learner (for this you will need to import the `from sklearn.linear_model import LinearRegression` extension package first) and use it to train a linear model on the abalone data.
- c) Compare the fitted and observed targets visually.
(Hint: use `import matplotlib.pyplot as plt`.)
- d) Assess the model's training loss in terms of MAE.
(Hint: The MAE metric is retrieved by calling `from sklearn.metrics import mean_absolute_error`.)



<https://en.wikipedia.org/wiki/Abalone#/media/File:LivingAbalone.JPG>