Exercise 1:

Throughout the lecture, we will frequently use the `R` package `mlr3` 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 `mlr3`. An overview on the most important objects and their use, illustrated with numerous examples, can be found at `https://mlr3book.mlr-org.com/basics.html`.

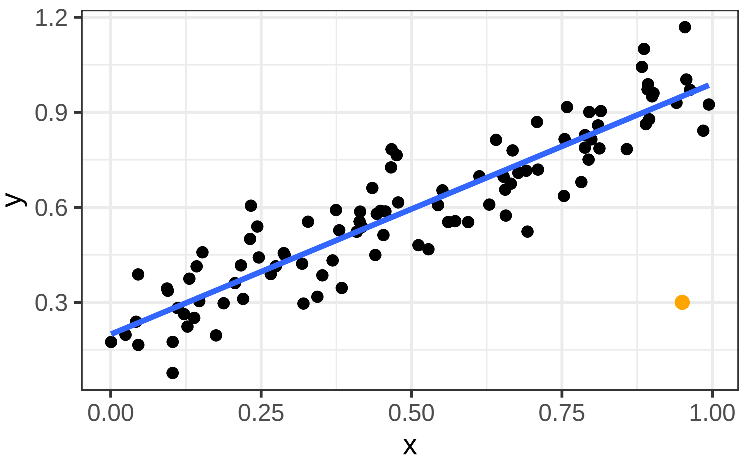
a) How are the key concepts you learned about in the lecture videos implemented in `mlr3`?

b) Have a look at `mlr3::tsk("iris")`. What attributes does this `task` object store?

c) Pick an `mlr3` learner of your choice. What are the different settings for this learner?
(Hint: Use `mlr3::mlr_learners$keys()` to see all available learners.)

Exercise 2:

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

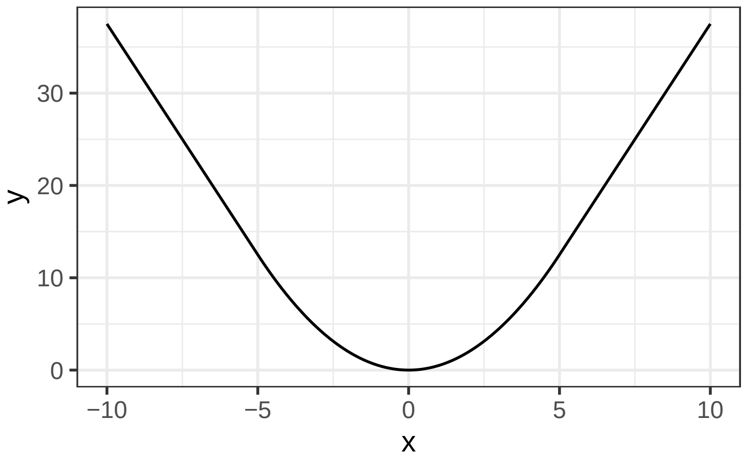


a) Consider the above linear regression problem. How will the model parameters be affected by adding the new outlier point (orange) if you use

   i) $L1$ loss

   ii) $L2$ loss

   in your 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 $\delta > 0$; here: $\delta = 5$). Describe how the Huber loss deals with residuals as compared to $L1$ and $L2$ loss. Can you guess its definition?

c) Zu schwer? Show that $median(y)$ is the optimal constant prediction c when using $L1$ loss.
Hint:

- Employing the law of total expectation, we can find c via

$$\arg\min_c \mathbb{E}\left[|y-c|\right] = \arg\min_c \int_{-\infty}^{\infty} |y-c|\ p(y)\mathrm{d}y = \arg\min_c \int_{-\infty}^{c} -(y-c)\ p(y)\ \mathrm{d}y + \int_{c}^{\infty} (y-c)\ p(y)\ \mathrm{d}y.$$

- Setting the derivative of this expression, found by application of Leibniz's rule, to zero yields

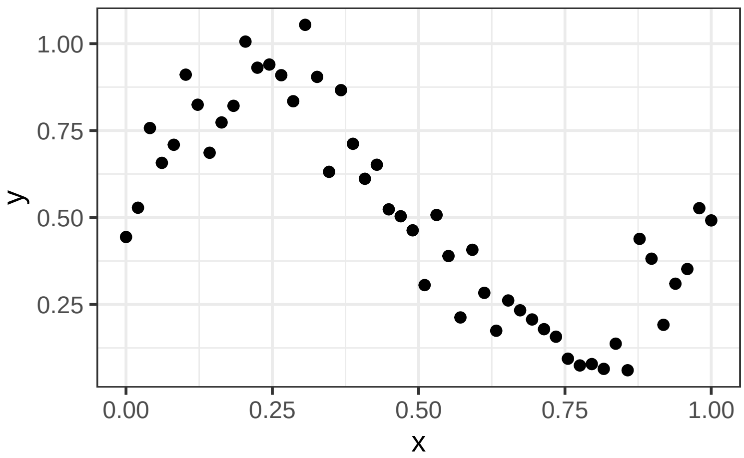
$$0 \overset{!}{=} \int_{-\infty}^{c} p(y)\ \mathrm{d}y - \int_{c}^{\infty} p(y)\ \mathrm{d}y.$$

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

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2.$$

Exercise 3:

Assume the following (noisy) data-generating process: $y = 0.5 + \sin(2\pi x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.1)$.



a) We decide to model the data with a cubic polynomial (including intercept term). State the corresponding hypothesis space.

b) Demonstrate that this hypothesis space is simply a parameterized family of curves by plotting in `R` curves for 3 different models belonging to the considered model class.

c) State the empirical risk w.r.t. $\boldsymbol{\theta}$ for a member of our hypothesis space. Use $L2$ loss and be as explicit as possible.

d) 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 $\tilde{\mathbf{X}}\boldsymbol{\theta}$ (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 $\boldsymbol{\theta}$.

e) Using the result from d), state the calculation to update the current parameter $\boldsymbol{\theta}^{[t]}$.

f) You will not be able to fit the data perfectly with a cubic polynomial. Should you opt for a more flexible model class (i.e., a hypothesis space with higher capacity)? What might be disadvantageous about this?

Exercise 4:

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

See `http://archive.ics.uci.edu/ml/datasets/Abalone` for more details.

```
url <-
  "http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
abalone <- read.table(url, sep = ",", row.names = NULL)
colnames(abalone) <- c(
  "sex", "longest_shell", "diameter", "height", "whole_weight",
  "shucked_weight", "visceral_weight", "shell_weight", "rings")
```

a) Plot `longest_shell` and `whole_weight` on the x- and y-axis, respectively, and color points with `rings`.

Using `mlr3`:

b) Define a linear regression learner (for this you will need to load the `mlr3learners` extension package first) and use it to fit a linear model to the `abalone` data. For this, first create a `task` like so (necessary if they have heard the intro to mlr3 by then?):

```
data <- abalone[, c("longest_shell", "shell_weight", "rings")]
task_abalone <- mlr3::TaskRegr$new(
  id = "abalone",
  backend = data,
  target = "rings")
```

c) Compare the fitted and observed targets visually.
(Hint: use `autoplot()`.)

d) Assess the model's training loss in terms of MAE.
(Hint: losses are retrieved by calling `$score()`, which accepts different `mlr_measures`, on the prediction object.)

Recall that you can find the official `mlr3` manual at `https://mlr3book.mlr-org.com/`.