Exercise 1: Risk Minimization and Gradient Descent

This exercise builds upon the previous exercise sheet.

(a) Write a function implementing a gradient descent routine for the optimization of the linear model defined in the previous exercise sheet. Start with:

```r
#' @param step_size the step_size in each iteration
#' @param X the feature input matrix X
#' @param y the outcome vector y
#' @param beta a starting value for the coefficients
#' @param eps a small constant measuring the changes in each update step.
#' Stop the algorithm if the estimated model parameters do not change
#' more than \code{eps}.

#' @return a set of optimal coefficients beta
gradient_descent <- function(step_size, X, y, beta, eps = 1e-8){

  # >>> do something <<<

  return(beta)

}
```

(b) Run a small simulation study by creating 20 data sets as indicated below and test different step sizes α (fixed across iterations) against each other and against the state-of-the-art routine for linear models (in R, using the function `lm`, in Python, e.g., `sklearn.linear_model.LinearRegression`).

- Compare the difference in estimated coefficients $\beta_j, j = 1, \dots, p$ using the mean squared error, i.e.

$$p^{-1}\sum_{j=1}^{p}(\beta_j^{truth} - \hat{\beta}_j)^2$$

  and summarize the difference over all 20 simulation repetitions.
- Compare the run times of your implementation and the one given by the state-of-the-art method by wrapping the function calls into a timer (e.g., `system.time()` in R).

```r
# settings
n <- 10000
p <- 100
nr_sims <- 20

# create data (only once)
X <- matrix(rnorm(n*p), ncol=p)
beta_truth <- runif(p, -2, 2)
f_truth <- X%*%beta_truth

# create result object
result_list <- vector("list", nr_sims)

for(sim_nr in nr_sims)
```

```r
{

  # create response
  y <- f_truth + rnorm(n, sd = 2)

  # >>> do something <<<


  # save results in list (performance, time)
  result_list[[sim_nr]] <- add_something_meaningful_here

}
```

(c) Why is gradient descent maybe not the best option for optimization of a linear model with $L2$ loss? What other options exist? Name at least one and describe how you would apply this for the above problem.

(d) Can we say something about the algorithm's consistency w.r.t. $\mathbb{P}_{xy}$, if $f^* \notin \mathcal{H}$?