Solution 1: Multivariate Regression

Consider the multivariate regression setting on $\mathcal{X} \subset \mathbb{R}^p$ without target features, i.e., $\mathcal{Y} = \mathbb{R}$ and $\mathcal{T} = \{1, \ldots, m\}$. Furthermore, consider the approach of learning a (simple) linear model $f_j(\mathbf{x}) = \mathbf{a}_j^\top \mathbf{x}$ for each target j independently. For this purpose, we would face the following optimization problem:

$$\min_A \|Y - \mathbf{X}A\|_F^2,$$

where $\|B\|_F^2 = \sqrt{\sum_{i=1}^n \sum_{j=1}^m B_{i,j}^2}$ is the Frobenius norm for a matrix $B \in \mathbb{R}^{n \times m}$ and

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^\top \\ \vdots \\ (\mathbf{x}^{(n)})^\top \end{bmatrix}, \qquad A = [\mathbf{a}_1 \quad \cdots \quad \mathbf{a}_m], \qquad Y = \begin{bmatrix} \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(n)} \end{bmatrix}.$$

(a) Show that $\hat{A} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top Y$ is the optimal solution in this case (provided that $\mathbf{X}^\top \mathbf{X}$ is invertible).

Solution:

Note that for a matrix $B = (\boldsymbol{b}_1 \ldots \boldsymbol{b}_m) \in \mathbb{R}^{n \times m}$, it holds that

$$\|B\|_F^2 = \sum_{i=1}^n \sum_{j=1}^m B_{i,j}^2 = \sum_{j=1}^m \sum_{i=1}^n B_{i,j}^2 = \sum_{j=1}^m \boldsymbol{b}_j^\top \boldsymbol{b}_j.$$

Thus, the function $f(A) = \|Y - \mathbf{X}A\|_F^2$ we want to minimize can be written as

$$\begin{aligned} \|Y - \mathbf{X}A\|_F^2 &= \sum_{j=1}^m (\boldsymbol{y}_j - \mathbf{X}\boldsymbol{a}_j)^\top (\boldsymbol{y}_j - \mathbf{X}\boldsymbol{a}_j) \\ &= \sum_{j=1}^m \boldsymbol{y}_j^\top \boldsymbol{y}_j - 2\boldsymbol{y}_j^\top \mathbf{X}\boldsymbol{a}_j + \boldsymbol{a}_j^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{a}_j, \end{aligned}$$

where $\boldsymbol{y}_j$ is the j-th column of Y. Therefore, we can write $f(A) = \sum_{j=1}^m f_j(\mathbf{a}_j)$, where

$$f_j(\mathbf{a}) = \boldsymbol{y}_j^\top \boldsymbol{y}_j - 2\boldsymbol{y}_j^\top \mathbf{X}\boldsymbol{a} + \boldsymbol{a}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{a}$$

and we can minimize each f_j separately (w.r.t. to $\mathbf{a}_j$). We compute the gradient of f_j and set it to $\mathbf{0}$ and solve w.r.t. $\mathbf{a}$:

$$\begin{gathered} \nabla f_j(\mathbf{a}) = -2\boldsymbol{y}_j^\top \mathbf{X} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{a} \stackrel{!}{=} \mathbf{0} \\ \Leftrightarrow \mathbf{a} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}_j. \end{gathered}$$

We check that this is indeed a minimum, by checking that the Hessian matrix is positive semi-definite: The Hessian matrix is

$$\nabla \nabla^\top f_j(\mathbf{a}) = 2\mathbf{X}^\top \mathbf{X}.$$

It is positive semi-definite, since for any $\boldsymbol{z} \in \mathbb{R}^p$ it holds that

$$\boldsymbol{z}^\top (2\mathbf{X}^\top \mathbf{X})\boldsymbol{z} = 2\boldsymbol{z}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{z} = 2(\mathbf{X}\boldsymbol{z})^\top \mathbf{X}\boldsymbol{z} = 2\|\mathbf{X}\boldsymbol{z}\|_2^2 \geq 0.$$

Consequently, the minimizer of f_j is $\hat{\mathbf{a}}_j = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}_j$, so that the minimizer of f is

$$\hat{A} = (\hat{\mathbf{a}}_1 \ \ldots \ \hat{\mathbf{a}}_m) = \big((\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}_1 \ \ldots \ (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}_m \big) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top Y.$$

In particular, the gradient of f w.r.t. matrix $A = (\mathbf{a}_1 \ \dots \ \mathbf{a}_m)$ is

$$\begin{aligned}
\nabla f(A) &= (\nabla f_1(\mathbf{a}_1) \ \dots \ \nabla f_m(\mathbf{a}_m)) \\
&= \left(-2\boldsymbol{y}_1^\top \mathbf{X} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{a}_1 \quad \dots \quad -2\boldsymbol{y}_m^\top \mathbf{X} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{a}_m\right) \\
&= -2Y^\top \mathbf{X} + 2\mathbf{X}^\top \mathbf{X}A.
\end{aligned}$$

Hence, a gradient descent routine with (fixed) step size α for f would iterate as follows:

$$\hat{A} \leftarrow \hat{A} - 2\alpha\left(-Y^\top \mathbf{X} + \mathbf{X}^\top \mathbf{X}\hat{A}\right).$$

(b) Assume that the data $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{X} \times \mathcal{Y}^m$ is generated[1] according to the following statistical model

$$(y_1, \dots, y_m) = \mathbf{y} = (\mathbf{x}^{(i)})^\top A^* + \boldsymbol{\epsilon}^\top,$$

where $A^* \in \mathbb{R}^{p\times m}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. Show that the maximum likelihood estimate for A^* coincides with the estimate in (a).

Solution:

Under the statistical model it holds that $\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)} \sim \mathcal{N}\left((\mathbf{x}^{(i)})^\top A^*, \boldsymbol{\Sigma}\right)$, i.e.[2],

$$\begin{aligned}
p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}, A^*) &= (2\pi)^{-m/2}|\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A^*)\boldsymbol{\Sigma}^{-1}(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A^*)^\top\right] \\
&\propto \exp\left[-\frac{1}{2}\left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A^*\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A^*\right)^\top\right]. \qquad (1)
\end{aligned}$$

So the log-likelihood for A is

$$\begin{aligned}
l(A \mid \mathcal{D}) &= \log\left(\prod_{i=1}^n p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}, A)\right) \\
&\propto \log\left(\exp\left[-\frac{1}{2}\sum_{i=1}^n \left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)^\top\right]\right) \\
&= -\sum_{i=1}^n \left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)^\top.
\end{aligned}$$

So, we want to maximize the function

$$\begin{aligned}
g(A) &= -\sum_{i=1}^n \left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A\right)^\top \\
&= -\sum_{i=1}^n \mathbf{y}^{(i)}\boldsymbol{\Sigma}^{-1}(\mathbf{y}^{(i)})^\top - 2(\mathbf{x}^{(i)})^\top A\boldsymbol{\Sigma}^{-1}(\mathbf{y}^{(i)})^\top + (\mathbf{x}^{(i)})^\top A\boldsymbol{\Sigma}^{-1}A^\top \mathbf{x}^{(i)}.
\end{aligned}$$

We compute the gradient of g and set it to $\mathbf{0}_{p\times m}$ and solve w.r.t. A :

$$\begin{aligned}
\nabla g(A) &= \sum_{i=1}^n 2\mathbf{x}^{(i)}\mathbf{y}^{(i)}\boldsymbol{\Sigma}^{-1} - 2\mathbf{x}^{(i)}(\mathbf{x}^{(i)})^\top A\boldsymbol{\Sigma}^{-1} \overset{!}{=} \mathbf{0}_{p\times m} \\
&\Leftrightarrow \mathbf{X}^\top Y\boldsymbol{\Sigma}^{-1} - \mathbf{X}^\top \mathbf{X}A\boldsymbol{\Sigma}^{-1} \overset{!}{=} \mathbf{0}_{p\times m} \\
&\Leftrightarrow A = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top Y,
\end{aligned}$$

where we used for computing the gradient that

$$\begin{aligned}
\frac{\partial \boldsymbol{z}^\top \boldsymbol{B}\tilde{\boldsymbol{z}}}{\partial \boldsymbol{B}} &= \boldsymbol{z}\tilde{\boldsymbol{z}}^\top, \qquad \forall \boldsymbol{z} \in \mathbb{R}^n, \tilde{\boldsymbol{z}} \in \mathbb{R}^m, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \\
\frac{\partial \boldsymbol{z}^\top \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^\top \tilde{\boldsymbol{z}}}{\partial \boldsymbol{B}} &= 2\boldsymbol{z}\tilde{\boldsymbol{z}}^\top \boldsymbol{B}\boldsymbol{C}^\top, \qquad \forall \boldsymbol{z} \in \mathbb{R}^n, \tilde{\boldsymbol{z}} \in \mathbb{R}^m, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \boldsymbol{C} \in \mathbb{R}^{n\times n}.
\end{aligned}$$

Moreover, any matrix product $\mathbf{X}^\top Y$ can be written as the sum of outer products of the column and row vectors: $\sum_{i=1}^n \mathbf{x}^{(i)}\mathbf{y}^{(i)}$. Thus, the MLE coincides with the OLS in (a).

[1] Of course, in an iid fashion and the $\mathbf{x}$'s are independent of the $\boldsymbol{\epsilon}$'s.

[2] Note that $\mathbf{y}^{(i)} - (\mathbf{x}^{(i)})^\top A^*$ is a row vector.

(c) Write a function implementing a gradient descent routine for the optimization of this linear model.

(d) 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 the estimated parameter matrices $\hat{A}$ using the mean squared error, i.e.,

$$\frac{1}{m \cdot p} \sum_{i=1}^{p} \sum_{j=1}^{m} (\mathbf{a}^*_{i,j} - \hat{\mathbf{a}}_{i,j})^2$$

  and summarize the difference over all 100 simulation repetitions.

- Compare the estimation also with the James-Stein estimate of A^*, which is given by

$$A^{JS} = \left(\mathbf{a}_1^{JS} \dots \mathbf{a}_m^{JS}\right),$$

  where

$$\mathbf{a}_j^{JS} = \left(1 - \frac{(m-2)\sigma^2}{n\|\hat{\mathbf{a}}_j - \mathbf{a}_j^*\|_2^2}\right)(\hat{\mathbf{a}}_j - \mathbf{a}_j^*) + \mathbf{a}_j^*, \quad j = 1, \dots, m.$$

  and $\hat{\mathbf{a}}_j$ is the MLE for the jth target parameter.

```r
#' @param step_size the step_size in each iteration
#' @param X the feature input matrix X
#' @param Y the score matrix Y
#' @param A the parameter matrix
#' @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 eps.

#' @return a set of optimal parameter matrix A

gradient_descent <- function(
                        step_size,
                        X,
                        Y,
                        A = matrix(rep(0,ncol(X)*m),ncol=m),
                        eps = 1e-8){

    change <- 1 # something larger eps

    XtX <- crossprod(X)
    XtY <- crossprod(X,Y)
    while(sum(abs(change)) > eps) {
        # Use standard gradient descent:
        change <- + step_size * (XtY - XtX%*%A)
        # update A in the end
        A <- A + change
    }
    return(A)
}

# make it all reproducible
set.seed(123)

# settings
n <- 10000
p <- 100

m <- 6
nr_sims <- 20

# define mse
mse <- function(x,y) mean((x-y)^2)
```

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

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

js_estimate <- function(A,A_truth) {
    A_JS = A
    for(i in 1:ncol(A)) {

        A_JS[,i]= (1-4*(m-2)/(n*sum((A[,i]-A_truth[,i])^2)))* (A[,i]-A_truth[,i])+
    A_truth[,i]
    }
    A_JS
}

for(sim_nr in seq_len(nr_sims)){
    # create response
    Y <- f_truth + rnorm(n*m, sd = 2)

    time_lm <- system.time( coef_lm <- coef(lm(Y~-1+X)) )["elapsed"]

    time_js <- system.time(
                coef_js <- js_estimate(coef_lm,A_truth) )["elapsed"]
    time_js = time_js + time_lm

    time_gd_1 <- system.time(
                    coef_gd_1 <- gradient_descent(
                        step_size = 0.0001,
                        X = X,
                        Y = Y)
                )["elapsed"]

    time_gd_2 <- system.time(
                    coef_gd_2 <- gradient_descent(
                        step_size = 0.00005,
                        X = X,
                        Y = Y)
                )["elapsed"]

    mse_lm <- mse(coef_lm, A_truth)
    mse_js <- mse(coef_js, A_truth)
    mse_gd_1 <- mse(coef_gd_1, A_truth)
    mse_gd_2 <- mse(coef_gd_2, A_truth)

    # save results in list (performance, time)
    result_list[[sim_nr]] <- data.frame(
                                mse_lm = mse_lm,
                                mse_js = mse_js,
                                mse_gd_1 = mse_gd_1,
                                mse_gd_2 = mse_gd_2,
                                time_lm = time_lm,
                                time_js = time_js,
                                time_gd_1 = time_gd_1,
                                time_gd_2 = time_gd_2)
}

library(ggplot2)
library(dplyr)
library(tidyr)
```

```r
do.call("rbind", result_list) %>%
    gather() %>%
    mutate(what = ifelse(grepl("mse", key), "MSE", "Time"),
        algorithm = gsub("(mse|time)\\_(.*)","\\2", key)) %>%
    ggplot(aes(x = algorithm, y = value)) + geom_boxplot() + theme_bw() +
    facet_wrap(~ what, scales = "free")
```