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

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

(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).

(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

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

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).

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

```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 a starting value for 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 \code{eps}.

#' @return a parameter matrix A
gradient_descent <- function(step_size, X, Y, A, eps = 1e-8){

  # >>> do something <<<

  return(A)

}
```

Hint: You have computed the gradient in (a).

(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`).

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

- 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
# settings
n <- 10000
p <- 100
m <- 6
nr_sims <- 20

# 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)

for(sim_nr in nr_sims)
{

  # create response
  Y <- f_truth + rnorm(n*m, sd = 2)

  # >>> do something <<<


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

}
```

Exercise 2: Conditional Random Fields vs. Structured SVMs

Similar to probabilistic classifier chains, conditional random fields try to model the conditional distribution $\mathbb{P}(\mathbf{y} \mid \mathbf{x})$ by means of

$$\pi(\mathbf{x}, \mathbf{y}) = \frac{\exp(s(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}^m} \exp(s(\mathbf{x}, \mathbf{y}'))},$$

where $x \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ with $\mathcal{Y}$ being a finite set (e.g., multi-label classification), and $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ being a scoring function. Training of a conditional random field is based on (regularized) empirical risk minimization using the negative log-loss:

$$\ell_{log}(\mathbf{x}, \mathbf{y}, s) = \log\left(\sum_{\mathbf{y}' \in \mathcal{Y}^m} \exp(s(\mathbf{x}, \mathbf{y}'))\right) - s(\mathbf{x}, \mathbf{y}).$$

Predictions are then made by means of

$$h(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}^m} s(\mathbf{x},\mathbf{y}). \tag{1}$$

Structured Support Vector Machines (Structured SVMs) are also using scoring functions for the prediction, but use the structured hinge loss for the (regularized) empirical risk minimization approach:

$$\ell_{shinge}(\mathbf{x},\mathbf{y},s) = \max_{\mathbf{y}'\in\mathcal{Y}^m} \left(\ell(\mathbf{y},\mathbf{y}') + s(\mathbf{x},\mathbf{y}') - s(\mathbf{x},\mathbf{y})\right),$$

where $\ell : \mathcal{Y}^m \times \mathcal{Y}^m \to \mathbb{R}$ is some target loss function (e.g., Hamming loss or subset 0/1 loss).

Show that if we use scoring functions s of the form

$$s(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{m} s_j(\mathbf{x}, y_j),$$

where $s_j : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are scoring functions for the j-th target, then

(a) conditional random fields are very well suited to model the case, where the distributions of the targets $y_1, \ldots, y_m$ are conditionally independent,

(b) the structured hinge loss corresponds to the multiclass hinge loss for the targets if we use the (non-averaged) Hamming loss for $\ell(\mathbf{y},\mathbf{y}') = \sum_{j=1}^{m} \mathbb{1}_{[y_j \neq y'_j]}$, i.e.,

$$\ell_{shinge}(\mathbf{x},\mathbf{y},s) = \sum_{j=1}^{m} \max_{y'_j\in\mathcal{Y}} \left(\mathbb{1}_{[y_j\neq y'_j]} + s_j(\mathbf{x}, y'_j) - s_j(\mathbf{x}, y_j)\right).$$