Deadline: Th. Nov. 29, 10:00 am Drop your printed or legible handwritten submissions into the boxes at Samelsonplatz, or upload them as `.pdf` or `.ipynb` files onto the LearnWeb. **On this sheet you can earn up to 4 bonus points.**

Exercise 1 (Optimization - 10+2 points)**.**

1. (4) Describe the differences between Coordinate Descent, Gradient Descent and the Newton Method. Under which conditions is each method appropriate?
2. (4) Write down the parameter updates of all 3 methods in the case of linear regression with MSE loss $\ell(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$
3. (4) Write down the parameter updates of all 3 methods in the case of logistic regression with log-likelihood loss $\ell(\beta) = \log L_{\mathcal{D}}^{\text{cond}}(\beta) = y^T \log \widehat{y} + (\mathbf{1} - y)^T \log(\mathbf{1} - \widehat{y})$ where $\widehat{y} = \text{logistic}(X\beta)$

Exercise 2 (Variable Selection - 10+2 points)**.** Recall that in the second part of exercise 4.1, we fitted a quadratic function of the form $\widehat{y}(x) = ax_1^2 + 2bx_1x_2 + cx_2^2$ to the data. Note that the ground truth in this case was

$$y = x_1^2 - x_2^2 + \epsilon \qquad \epsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$$

i.e. $a = 1, b = 0, c = -1$. Implement the forward search algorithm and test it on the `tutorial4.2.dat` dataset with a general second degree polynomial model

$$\widehat{y}(x) = \widehat{\theta}_0 + \widehat{\theta}_1 x_1 + \widehat{\theta}_2 x_2 + \widehat{\theta}_3 x_1^2 + \widehat{\theta}_4 x_1 x_2 + \widehat{\theta}_5 x_2^2$$

At the start of each outer loop, print out the currently selected variables V as well as the test error e_{best}. Run your algorithm 5 times, in each case on a freshly randomized, approximate 50/50 training/test data split.

Are the results always the same? Which parameters are added first to the model? Can you think of any improvements to the algorithm such that it would only select the true variables in this example?

Hint: The `scikit-learn` repository[1] is a python package for machine learning. It offers some useful functions to deal with polynomial regression. Especially the class `sklearn.preprocessing.PolynomialFeatures`[2]can be useful for solving this exercise.

[1] https://scikit-learn.org/stable/
[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html