Problem 1.1 (25 %)

Consider the following polynomial regression scheme of degree m, with the predictions $\hat{y}_{(i)}$ for a scalar input $x_{(i)}$ given by:

$$\hat{y}_{(i)} = w_0 + w_1 x_{(i)} + w_2 x_{(i)}^2 + \cdots + w_m x_{(i)}^m. \tag{1.1.1}$$

We can fit this regressor to a given training dataset $T_{train} = \{x_{(i)}, y_{(i)}\}_{i=1}^n$, by minimizing the loss $J_{\text{LS}}(\mathbf{w})$ through the corresponding least squares problem:

$$\begin{bmatrix} \hat{y}_{(1)} \\ \hat{y}_{(2)} \\ \hat{y}_{(3)} \\ \vdots \\ \hat{y}_{(n)} \end{bmatrix} = \begin{bmatrix} 1 & x_{(1)} & x_{(1)}^2 & \cdots & x_{(1)}^m \\ 1 & x_{(2)} & x_{(2)}^2 & \cdots & x_{(2)}^m \\ 1 & x_{(3)} & x_{(3)}^2 & \cdots & x_{(3)}^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{(n)} & x_{(n)}^2 & \cdots & x_{(n)}^m \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \tag{1.1.2}$$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmin}}\ J_{\text{LS}}(\mathbf{w}) = \underset{\mathbf{w}}{\text{argmin}}\ \|X\mathbf{w} - \mathbf{y}\|^2. \tag{1.1.3}$$

While least squares provides a weight configuration $\hat{\mathbf{w}}$ that minimizes the error on T_{train}, it lacks a dedicated scheme to control the *generalization* to unseen data.

Ridge regression is an extension to least squares that introduces an additional regularization parameter $\lambda \geq 0$. The adapted loss is then given by:

$$J_{\text{Ridge}}(\mathbf{w}) = \|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2. \tag{1.1.4}$$

In the following we will examine the role this *regularization* parameters plays in addressing *over-&underfitting*, by evaluating how least squares and Ridge regression perform on the test dataset $T_{test} = \{x_{(i)}, y_{(i)}\}_{i=n+1}^N$.

$\boxed{1.1.1}$ Download the files *regression_train.csv* and *regression_test.csv* from TUWEL and use *pandas*[1] to extract the data. Generate a scatterplot of T_{train} and T_{test} combined and highlight each dataset through a distinct color.

$\boxed{1.1.2}$ Implement the code to fit the weight vector $\hat{\mathbf{w}}$ to the training dataset T_{train}. I.e., minimize (1.1.3) for an arbitrary degree m. Use the trained regressor and calculate the MSE[2] of your predictions over T_{test} with $m = [0,1,\ldots,10]$. Provide a stem plot showing the values over m. Which degree m leads to the smallest error?
Hint: Use the numpy function pinv for numerically stable matrix inversions.

$\boxed{1.1.3}$ Additionally compute the predictions $\hat{y}$ of the regressor for inputs in the interval $x \in [0,2]$ for all the configurations of m from Task 1.1.2. Depict the predictions as a line plot ontop of the scatter plots from Task 1.1.1. What do you observe for high and low values of m?

[1] https://pandas.pydata.org/
[2] Mean Squared Error

$\boxed{1.1.4}$ Use matrix calculus rules to derive the analytical solution for the weight vector, which minimizes the Ridge regression loss given in (1.1.4). I. e., find $\hat{\mathbf{w}}$, such that $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, J_{\text{Ridge}}(\mathbf{w})$. Use your result, to extent the code from Task 1.1.2 to Ridge regression. *Hint: Note, that the results should be equivalent to least squares for $\lambda = 0$.*

$\boxed{1.1.5}$ Fit a Ridge regressor with $m = 5$ to T_{train} and use the trained model to calculate the MSE on T_{test}. Plot the MSE over λ for $\lambda = [0, 0.1, 1, 5, 10, 100, 500, 1000]$. Again depict the predictions for $x \in$ [0,2] ontop of the scatter plots from Task 1.1.1. What to you observe for high and low values of λ?

The role of the regularization parameter also becomes apparent when considering a probabilistic interpretation of least squares. Let's consider the following Bayesian approach

$$p(\mathbf{y}|\mathbf{w}) = \mathcal{N}\big(\mathbf{y}; \mathbf{X}\mathbf{w}, \mathbf{I}\sigma_n^2\big) \qquad p(\mathbf{w}) = \mathcal{N}\big(\mathbf{w}; \mathbf{0}, \mathbf{I}\sigma_w^2\big), \tag{1.1.5}$$

with a Gaussian likelihood and an i.i.d. zero-mean Gaussian prior for $\mathbf{w}$ with variance σ_w^2. Using Bayes's rule we can then optimize for $\hat{\mathbf{w}}$ by minimizing $J(\mathbf{w}) = -\log p(\mathbf{w}|\mathbf{y})$.

$\boxed{1.1.6}$ Find an analytical expression for $J(\mathbf{w})$ under the Bayesian interpretation given in (1.1.5) and compare the result to the loss function from (1.1.4). Then, repeat the step for a Laplacian prior with $p(\mathbf{w}) = \prod_{i=0}^{m} \frac{1}{2\pi\sigma_w} e^{-\frac{|w_i|}{\sigma_w}}$. Briefly discuss which solutions for $\hat{\mathbf{w}}$ are promoted by the different priors. *Hint: Note, that you can discard all terms not depending on* $\mathbf{w}$.

Problem 1.2 (25 %)

In the following, we will evaluate the use of linear least squares for a classification problem. As such, we are given a dataset $T_{data} = \{\mathbf{x}_{(i)}, y_{(i)}\}_{i=1}^{N}$ consisting of inputs $\mathbf{x} = [x_0, x_1]^T$ and their respective labels $y \in \{-1, 1\}$. For each input $\mathbf{x}_{(i)}$, the classifier generates a soft-label prediction $\tilde{y}_{(i)}$ according to:

$$\tilde{y} = w_0 x_0 + w_1 x_1 + w_2. \tag{1.2.1}$$

Following a least squares approach, the model can be trained by minimizing the corresponding loss function to obtain the weight estimate $\hat{\mathbf{w}}$:

$$J_{\mathrm{LS}}(\mathbf{w}) = \|X\mathbf{w} - \mathbf{y}\|^2 = \|\tilde{\mathbf{y}} - \mathbf{y}\|^2 \qquad \hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, J_{\mathrm{LS}}(\mathbf{w}). \tag{1.2.2}$$

Subsequently, we use the following rule to transform the soft-label output $\tilde{y}_{(i)}$ into a hard-label $\hat{y}_{(i)}$:

$$\hat{y}_{(i)} = \begin{cases} 1, & \text{if } \tilde{y}_{(i)} \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{1.2.3}$$

We can then examine the performance of the trained classifier by comparing the predictions $\hat{y}_{(i)}$ with the given labels $y_{(i)}$.

$\boxed{1.2.1}$ Download the files *data_ blob_ train.csv* and *data_ blob_ test.csv* from TUWEL and generate a scatterplot of each T_{train} and T_{test}. Highlight the respective class labels by color.

$\boxed{1.2.2}$ Fit the classifier to the training data, i.e., obtain the least squares solution for $\hat{\mathbf{w}}$ with X and $\mathbf{y}$ from the training dataset. Provide the accuracy of your trained classifier for T_{train} and T_{test}. Further, generate a heatmap of the soft-label output $\tilde{y}$ in the interval $x_0 \in [-3{,}3]$ and $x_1 \in [-3{,}3]$.

$\boxed{1.2.3}$ Find an expression for the decision surface, i.e., the line separating the two classes $\hat{y} = 1$ and $\hat{y} = -1$. Plot this line ontop of the scatter plots from Task 1.2.1.

$\boxed{1.2.4}$ Calculate the least squares error of your trained classifier for the samples s_1 & s_2. I.e., obtain the loss $J_{\mathrm{LS}}(\hat{\mathbf{w}})$ between the given label y and the soft-label $\tilde{y}$ for each of the two samples. Which of the samples contributes the higher loss to the optimization?

$$s_1 = \{[-10{,}10]^T, 1\} \qquad s_2 = \{[0{,}1.5]^T, -1\} \tag{1.2.4}$$

Now assume, that the hard-labels $\hat{y}$ are used for the loss calculation and compare the results. I.e., again calculate $J_{\mathrm{LS}}(\hat{\mathbf{w}})$ for both samples, but replace $\tilde{y}$ with $\hat{y}$. Why is the optimization of the soft-labels via least squares not ideal?

$\boxed{1.2.5}$ Download the file *data_ moon_ train.csv* and *data_ moon_ test.csv* from TUWEL and again provide a scatterplot highlighting the respective class labels. Then repeat Task 1.2.2-1.2.3. Is a linear classifier an appropriate choice for this dataset?

Problem 1.3 (25 %)

The california housing dataset was derived from the 1990 US census and consists of the features shown in table 1.3.1. As such, each entry includes data for a distinct californian district with the median house value in 100 000 USD as the target variable.

Input: $\mathbf{x}$								Target: y
MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal

Table 1.3.1: California Housing Data

In the following, you will develop a regressor operating on this real world dataset. I.e., you will train a model that can predict the median house price for a given configuration of features. For this, we will use the Python machine learning library *scikit-learn*. The dataset is available via:

from sklearn.datasets import fetch_ california_ housing

Note, that you might want to set the following optional parameters:

as_ frame=True, download_ if_ missing=False

$\boxed{1.3.1}$ Download the california housing dataset and collect the inputs and targets in a *pandas* dataframe. Examine the output of $.describe()$ and $.corr()$. Which feature has the highest correlation with the target value?

$\boxed{1.3.2}$ Conduct basic visual data analysis by generating histograms for single features. Further, expose the relations between different features through scatter plots. Include a selection of plots in your report and briefly discuss your findings. *Hint: You might want to use the function relplot from the seaborn library.*

$\boxed{1.3.3}$ Use the pandas function $.sample(frac = 1, random\_state = 1)$ to shuffle the dataset and split it into training and test data. Use the first 15 000 elements of the shuffled dataset as T_{train} and the remaining ones as T_{test}.

$\boxed{1.3.4}$ Fit a Ridge regressor[3] with $alpha = 1$ to the training dataset T_{train}. Extract the weight vector and the intercept of your trained regressor and provide a list of the features sorted by their weights. Which features have a positive/negative impact on the house price? *Hint: Check the scikit-learn documentation for details.*

$\boxed{1.3.5}$ Compute the MAE[4] in USD between the predictions $\hat{y}$ of your model and the targets y for T_{train} and T_{test} separately. Visualize the error $(\hat{y}_i - y_i)$ through histograms for both subsets. Further generate a scatter plot of your predictions $\hat{y}$ and the respective ground truth values y for the samples in T_{test}. What would the plot look like for a model without any error?

[3] from sklearn.linear_model import Ridge

[4] Mean Absolute Error

$\boxed{1.3.6}$ Implement 10-fold cross-validation and compute mean and standard deviation of your estimation error over all evaluation runs. Select MAE as the metric and make sure to use all available data by concatenating T_{train} and T_{test}. What do the results tell you about your regressor? *Hint: An additional resource on cross-validation is provided on TUWEL.*

$\boxed{1.3.7}$ Fit a random forest regressor[5] from *scikit-learn* to the training dataset and repeat Task 1.3.5 and 1.3.6. Briefly discuss how the outcomes for the random forest regressor differ from Ridge regression.

[5] sklearn.ensemble.RandomForestRegressor(n_estimators=50)

Problem 1.4 (25 %)

In a previous problem we showed, that the naive least squares approach is not well suited for classification due to the lack of a squashing function. For a given dataset $T = \{\mathbf{x}_{(i)}, y_{(i)}\}_{i=1}^{N}$ with $y_{(i)} \in \{-1, 1\}$, the perceptron considers the classification objective:

$$\begin{aligned} \mathbf{x}_{(i)}^T\mathbf{w} + b \geq 0 \qquad & \text{if } y_{(i)} = 1 \\ \mathbf{x}_{(i)}^T\mathbf{w} + b < 0 \qquad & \text{if } y_{(i)} = -1, \end{aligned} \tag{1.4.1}$$

which is enforced through the following loss function:

$$\tilde{J}(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \max\big(0, -y_{(i)}\big(\mathbf{x}_{(i)}^T\mathbf{w} + b\big)\big). \tag{1.4.2}$$

Due to the fact, that $\mathbf{w} = \mathbf{0}$ and $b = 0$ are trivial solutions to the minimization of (1.4.2), the following approximated loss function is typically considered in practice:

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \log\Big(1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T\mathbf{w}+b\right)}\Big). \tag{1.4.3}$$

The hard-label outputs $\hat{y}_{(i)}$ of the perceptron are then calculated from the soft-label predictions $\tilde{y}_{(i)} = \mathbf{x}_{(i)}^T\mathbf{w} + b$ according to:

$$\hat{y}_{(i)} = \begin{cases} 1, & \text{if } \tilde{y}_{(i)} \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{1.4.4}$$

$\boxed{1.4.1}$ In (1.4.3) the maximum expression from (1.4.2) is approximated by the function $f(t) = \log(1 + e^t)$. Plot $f(t)$ in the interval $t \in [-10, 10]$. For which combinations of $y_{(i)}$ and $\tilde{y}_{(i)}$ does $J(\mathbf{w}, b)$ saturate?

$\boxed{1.4.2}$ Find an analytical expression for the gradient $\nabla J(\mathbf{w}, b)$ of the approximated loss function from (1.4.3). For that, assume a generic weight vector with two entries $\mathbf{w} = [w_0, w_1]^T$.

$\boxed{1.4.3}$ Use the result from Task 1.4.2 and implement gradient descent to find a weight $\hat{\mathbf{w}} = [\hat{w}_0, \hat{w}_1]^T$ and bias $\hat{b}$ estimate, that minimizes the loss $J(\mathbf{w}, b)$ over a given training dataset. In gradient descent the update equation for the estimates is given by:

$$\begin{bmatrix} \hat{\mathbf{w}}^{(k)} \\ \hat{b}^{(k)} \end{bmatrix} \leftarrow \begin{bmatrix} \hat{\mathbf{w}}^{(k-1)} \\ \hat{b}^{(k-1)} \end{bmatrix} - \alpha \cdot \nabla J(\hat{\mathbf{w}}^{(k-1)}, \hat{b}^{(k-1)}). \tag{1.4.5}$$

Implement the scheme in a way, that allows for generic values of $\hat{\mathbf{w}}^{(0)}$, $\hat{b}^{(0)}$, the step size α as well as the overall iteration duration K. Note, that $k = [1, \ldots, K]$.

$\boxed{1.4.4}$ Use the code from Task 1.4.3 to fit your perceptron to the dataset *classification.csv* from TUWEL. For that use $\hat{\mathbf{w}}^{(0)} = [0, -2]$, $\hat{b}^{(0)} = 2$, $K = 600$ and $\alpha = 0.01$. Report the parameters $\hat{\mathbf{w}}^{(K)}$ and $\hat{b}^{(K)}$ and visualize the decision surface of your trained classifier. What is the accuracy of your classifier on the training dataset?

$\boxed{1.4.5}$ Save the history of the weights $\hat{w}_0^{(k)}$, $\hat{w}_1^{(k)}$ and the bias term $\hat{b}^{(k)}$ during the optimization and visualize them over the gradient descent iterations $k = [1, \ldots, K]$. Also generate a plot of the optimization history of $J(\hat{\mathbf{w}}^{(k)}, \hat{b}^{(k)})$.

$\boxed{1.4.6}$ Finally, use your trained model to calculate the loss $J(\mathbf{w}, b)$ for each of the samples s_1 & s_2. Why is the the given training objective superior to the least squares classifier from the previous problem?

$$s_1 = \{[-100,0]^T, 1\} \qquad s_2 = \{[-1,0]^T, -1\} \tag{1.4.6}$$