Problem 2.1 (2.5 points)

In the last exercise we showed, that the naive least squares approach is not well suited for classification due to the lack of a squashing function. The perceptron introduces a different classification objective for a given dataset $T = \{\mathbf{x}_{(i)}, y_{(i)}\}_{i=1}^{N}$:

$$\begin{aligned} \mathbf{x}_{(i)}^T\mathbf{w} + b &\geq 0 \qquad \text{if } y_{(i)} = 1 \\ \mathbf{x}_{(i)}^T\mathbf{w} + b &< 0 \qquad \text{if } y_{(i)} = -1. \end{aligned} \tag{2.1.1}$$

As such, the loss function is given by:

$$\tilde{J}(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \max\big(0, -y_{(i)}\big(\mathbf{x}_{(i)}^T\mathbf{w} + b\big)\big). \tag{2.1.2}$$

Due to the fact, that $\mathbf{w} = \mathbf{0}$ and $b = 0$ are trivial solutions to the minimization of Equation (2.1.2), we typically optimize the following approximation:

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \log\Big(1 + e^{-y_{(i)}\left(\mathbf{x}_{(i)}^T\mathbf{w}+b\right)}\Big). \tag{2.1.3}$$

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:

$$\hat{y}_{(i)} = \begin{cases} 1, & \text{if } \tilde{y}_{(i)} \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{2.1.4}$$

$\boxed{2.1.1}$ In Equation (2.1.3) the maximization from (2.1.2) 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?

$\boxed{2.1.2}$ Find an analytical expression for the gradient $\nabla J(\mathbf{w}, b)$ of the approximated loss function from Equation (2.1.3). For that, assume a generic weight vector with two entries $\mathbf{w} = [w_0, w_1]^T$.

$\boxed{2.1.3}$ Use the result from Task 2.1.2 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:

$$\begin{bmatrix} \hat{\mathbf{w}}^{(k)} \\ \hat{b}^{(k)} \end{bmatrix} \leftarrow \begin{bmatrix} \hat{\mathbf{w}}^{(k-1)} \\ \hat{b}^{(k-1)} \end{bmatrix} - \alpha \cdot \nabla J(\hat{\mathbf{w}}^{(k-1)}, \hat{b}^{(k-1)}). \tag{2.1.5}$$

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 $k = [1, \ldots, K]$.

$\boxed{2.1.4}$ Use the code from Task 2.1.3 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 $\hat{b}^{(K)}$. What is the accuracy of your classifier on the training dataset?

$\boxed{2.1.5}$ Repeat the optimization from Task 2.1.4 and save the history of the estimates $\hat{w}_0^{(k)}$, $\hat{w}_1^{(k)}$ and $\hat{b}^{(k)}$. Plot each value over the gradient descent iterations $k = [1, \ldots, K]$. Do the same for the loss $J(\hat{\mathbf{w}}^{(k)}, \hat{b}^{(k)})$.

$\boxed{2.1.6}$ Provide a scatterplot of the training dataset and draw the decision surface of your trained classifier ontop.

$\boxed{2.1.7}$ 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 last exercise?

$$s_1 = \{[-100,0]^T, 1\} \qquad s_2 = \{[-1,0]^T, -1\} \tag{2.1.6}$$

Problem 2.2 (2.5 points)

SVCs[1] are obtained by adjusting the loss function of the perceptron to include a buffer zone between the separating hyperplane $\mathbf{x}_{(i)}^T\mathbf{w} + b = 0$ and the symmetric translated versions of itself given by $\mathbf{x}_{(i)}^T\mathbf{w} + b = \pm 1$. This leads to the following formulation of the classification objective for a given dataset $T = \{\mathbf{x}_{(i)}, y_{(i)}\}_{i=1}^N$:

$$\begin{aligned} \mathbf{x}_{(i)}^T\mathbf{w} + b &\geq 1 \qquad \text{if } y_{(i)} = 1 \\ \mathbf{x}_{(i)}^T\mathbf{w} + b &\leq -1 \qquad \text{if } y_{(i)} = -1. \end{aligned} \tag{2.2.1}$$

This criterion is reflected in the SVC loss function:

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \max\bigl(0{,}1 - y_{(i)}\bigl(\mathbf{x}_{(i)}^T\mathbf{w} + b\bigr)\bigr) + \lambda\|\mathbf{w}\|^2 \tag{2.2.2}$$

where $\lambda \geq 0$ is a hyperparameter. The term $\lambda\|\mathbf{w}\|^2$ in Equation (2.2.2) encodes the objective, that the margin around the hyperplane should be maximized.

SVCs are of particular interest because they offer an efficient extension to a non-linear regime. By using the kernel trick, they can map a given input dataset to a higher dimension $d_2 > d_1$, in which the data is again linearly separable.

$$\phi : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, \quad \mathbf{x}_{(i)} \mapsto \phi(\mathbf{x}_{(i)}) \tag{2.2.3}$$

$\boxed{2.2.1}$ Download the file *blobs.csv* from TUWEL and generate a scatter plot that highlights the two classes in color. Fit a SVC[2] with a linear kernel and $C = 1$ to the dataset and extract the weights $\mathbf{w}$ and the bias b from the trained model.

$\boxed{2.2.2}$ Plot the separating hyperplane as well as the two equidistant translations running through the support vectors ontop of the scatter plot from Task 2.2.1. Calculate the margin of the separating hyperplane.

$\boxed{2.2.3}$ Repeat Tasks 2.2.1 and 2.2.2 for the dataset *circles.csv*. Is this dataset lineraly separable in the given space?

$\boxed{2.2.4}$ Find a transformation ϕ so that the *circles.csv* dataset is linearly separable in the new space $\mathbf{x}'$.

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3, \quad \phi(\mathbf{x}) = \mathbf{x}' = [x_0, x_1, x_2']^T \tag{2.2.4}$$

Provide a scatterplot of the transformed data showing x_0 and x_2' with $x_1 = 0$. Repeat the training of the SVC on the transfomed dataset using a linear kernel and $C = 100$. Again plot the hyperplane and the two equidistant translations ontop of the 2D scatter plot.

[1]Support Vector Classifiers
[2]sklearn.svm.SVC(C=1, kernel='linear')

$\boxed{2.2.5}$ Now fit a SVC with a polynomial kernel[3] of degree 2 to the original 2D *circles.csv* dataset and plot a heatmap of the predicted labels $\hat{y}$ for the interval $x_0 \in [-2, 2]$ and $x_1 \in [-2, 2]$. Is the separating hyperplane a linear function of $\mathbf{x}$?

$\boxed{2.2.6}$ The loss function of the SVC can be reformulated in a way, that the input data $\mathbf{x}$ only enters in the form of the inner product $\langle \mathbf{x}_{(i)}, \mathbf{x}_{(j)} \rangle$ between two samples i and j. As such, the transformation $\phi(\mathbf{x})$ is implicitly defined via the kernel function k:

$$k\big(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}\big) = \big\langle \phi\big(\mathbf{x}_{(i)}\big), \phi\big(\mathbf{x}_{(j)}\big) \big\rangle. \tag{2.2.5}$$

The polynomial kernel of degree 2 used in Task 2.2.5 is for instance given by:

$$k_{\text{poly},d=2}\big(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}\big) = \big(\mathbf{x}_{(i)}^{\top}\mathbf{x}_{(j)} + 1\big)^2. \tag{2.2.6}$$

Find an expression for the transformation $\mathbf{x}' = \phi(\mathbf{x})$ that is implicitly applied to $\mathbf{x} = [x_0, x_1]^T$ via the polynomial kernel in Equation (2.2.6). What is the dimension d_2 after the transformation?

[3]sklearn.svm.SVC(C=1, kernel="poly", degree=2)

Problem 2.3 (2.5 points)

Most machine learning algorithms are based on the assumption that the dataset at hand is balanced. I.e., that the targets y of the samples are equally distributed. In the following we will examine how common evaluation metrics such as *accuracy* are misleading, whenever this assumption is not fulfilled.

$\boxed{2.3.1}$ Download the files *imbalanced_ train.csv* and *imbalanced_ test.csv* from TUWEL and extract T_{train} and T_{test} using *pandas*. Addionally, evaluate the class distribution by plotting histograms of y for each the training and test dataset.

$\boxed{2.3.2}$ Fit a support vector classifier[4] with an RBF kernel and a regularization parameter of $C = 0.01$ to T_{train}. What is the accuracy of your model on T_{test}?

$\boxed{2.3.3}$ Calculate the confusion matrix — see Table 2.3.1 — of your classifier on T_{test}. If possible, provide the values for *Precision*, *Recall* and *F1-Score*. What do you observe? *Hint: The Precision is given by* $\frac{TP}{TP+FP}$, *while the Recall is defined as* $\frac{TP}{TP+FN}$. *We denote the harmonic mean of both measures as the F1-Score.*

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	True Negatives (TN)	False Positives (FP)
$y = 1$	False Negatives (FN)	True Positives (TP)

Table 2.3.1: Confusion Matrix

$\boxed{2.3.4}$ *Resampling* is a commonly used technique to tackle class imbalances in the training dataset. In *undersampling* we randomlly select samples (without replacement) from the majority class until we obtain a balanced problem. Implement *undersampling* for T_{train} and repeat Tasks 2.3.2 and 2.3.3.
Hint: Note, that we train on all samples from the minority class in undersampling.

$\boxed{2.3.5}$ *Oversampling* refers to a scheme, where we sample (with replacement) from the minority class until the training dataset is equally split. Implement *oversampling* for T_{train} and again repeat Tasks 2.3.2 and 2.3.3. Which of the two *resampling* variants achieves the highest *F1-Score*?
Hint: Note, that we train on all samples from the majority class in oversampling.

[4]sklearn.svm.SVC()