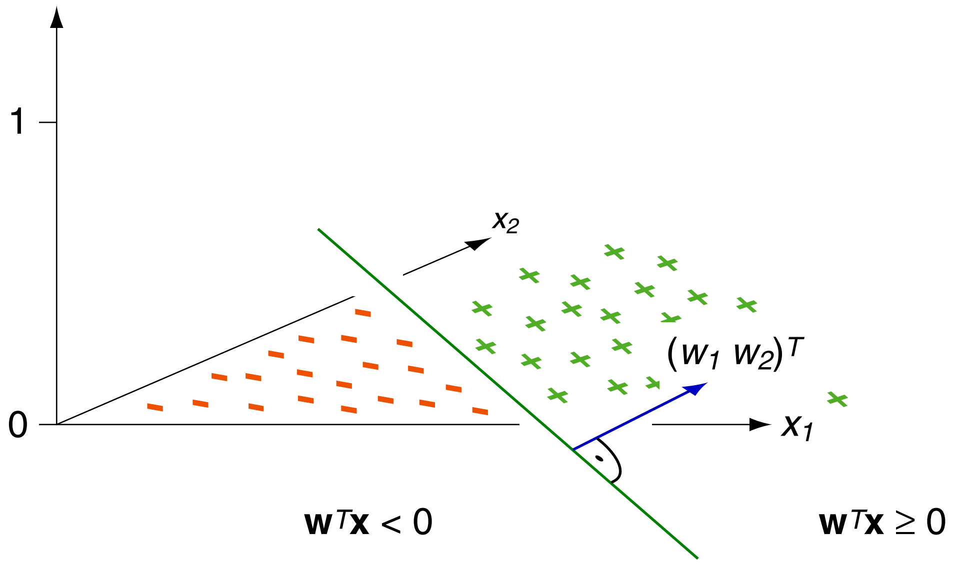
Until Wednesday, Dec. 6th, 2023, 11:59 pm CET, solutions to the following exercises must be submitted as one zip-file named `ml23-ex3-group<your-group-number>.zip` via Moodle:
1, 2, 3a,b,d,e,f, 4, and 5.

Exercise 1 : Linear Models (0.5+0.5+0.5+0.5+0.5+0.5=3 Points)

(a) Name these concepts: $l(c, y(\mathbf{x}))$, $L(\mathbf{w})$, $\mathcal{L}(\mathbf{w})$, $\mathbf{w}$, $\vec{\mathbf{w}}$

(b) How would the figure below change if w_0 is halved?



(c) What is the difference (if any) between decision boundaries for linear and logistic regression?

(d) The lecturenotes slides state that a key difference between ridge ($R_{||\vec{\mathbf{w}}||_2^2}$) and lasso ($R_{||\vec{\mathbf{w}}||_1}$) regression is that, with lasso regression, parameters can be reduced to zero. Explain why.

(e) Why can the gradient descent method not be applied for $L_{0/1}(\mathbf{w})$?

Exercise 2 : Pointwise Loss Functions (2+1=3 Points)

In the lecturenotes, slide ML:III-63 on loss computation for logistic regression in detail, the rightmost plot “Loss over hyperplace distance” shows the pointwise logistic and 0/1 loss for a logistic regression model for $l_\sigma(1, y(\mathbf{x}))$, that means, for examples with $c = 1$. In this exercise you will investigate the case of examples with $c = 0$.

(a) Show that $l_\sigma(0, y(\mathbf{x})) = \log(1 + e^{\mathbf{w}^T\mathbf{x}})$. Hint: $\sigma(-a) = 1 - \sigma(a)$.

(b) Draw the plot “Loss over hyperplace distance” for examples with $c = 0$ (both logistic and 0/1 loss).

Exercise 3 : Gradient Descent (1.5+0.5+0+0.5+0.5+1+0+0=4 Points)

In this exercise you will be calculating one iteration of the LMS algorithm, slide ML:I-42.

The set D contains the following three examples of one-dimensional vectors with the two classes $\{-1, 1\}$:

	x_1	c
$\mathbf{x}_1$	4	1
$\mathbf{x}_2$	1	-1
$\mathbf{x}_3$	5	1

Assume the weight vector $\mathbf{w}$ was randomly initialized to $\mathbf{w} := (1, 2)^T$ and $\mathbf{x}_1$ was randomly selected for the first iteration of the algorithm.

(a) Plot the line defined by $\mathbf{w}$ and all examples from D into one coordinate system.

(b) Compute the squared loss w.r.t. $\mathbf{x}_1$ and $\mathbf{w}$. The squared loss is defined as

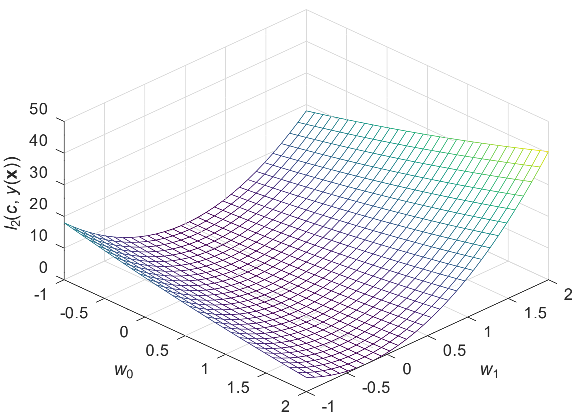
$$l_2(c, y(\mathbf{x})) = \frac{1}{2} \cdot (c - y(\mathbf{x}))^2.$$

(c) Show that the loss gradient, $\left(\frac{\partial l_2}{\partial w_0}, \frac{\partial l_2}{\partial w_1}\right)^T$, is indeed equal to $-\delta \cdot \mathbf{x}$.

(d) Derive the loss gradient for $\mathbf{x}_1$ and $\mathbf{w}$.

(e) Calculate $\Delta\mathbf{w}$ with a learning rate $\eta = 0.03$ for $\mathbf{x}_1$ and $\mathbf{w}$.

(f) The following plot shows the loss landscape defined by l_2 for $\mathbf{x}_1$. Mark the location of the model for $\mathbf{w}$ and for its update $\mathbf{w} + \Delta\mathbf{w}$.



(g) Compute the squared loss w.r.t. $\mathbf{x}_1$ and the updated $\mathbf{w}$. Could it be possible that this loss is now larger than it was before the update?

(h) Repeat for $\mathbf{x}_2$ and $\mathbf{x}_3$.

Exercise 4 : Regularization (1+1=2 Points)

Suppose we are estimating the regression coefficients in a linear regression model by minimizing the objective function $\mathcal{L}$.

$$\mathcal{L}(\mathbf{w}) = \mathsf{RSS}_{tr}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

The term $\mathsf{RSS}_{tr}(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in D_{tr}} \left(y_i - \mathbf{w}^T \mathbf{x}_i \right)^2$ refers to the residual sum of squares computed on the set D_{tr} that is used for parameter estimation. Assume that we can also compute an RSS_{test} on a separate set D_{test} that we don't use during training.

When we vary the hyperparameter λ, starting from 0 and gradually increase it, what will happen to the following quantities? Explain your answers.

(a) The value of $\mathsf{RSS}_{tr}(\mathbf{w})$ will...

- ☐ remain constant.
- ☐ steadily increase.
- ☐ steadily decrease.
- ☐ increase initially, then eventually start decreasing in an inverted U shape.
- ☐ decrease initially, then eventually start increasing in a U shape.

(b) The value of $\mathsf{RSS}_{test}(\mathbf{w})$ will...

- ☐ remain constant.
- ☐ steadily increase.
- ☐ steadily decrease.
- ☐ increase initially, then eventually start decreasing in an inverted U shape.
- ☐ decrease initially, then eventually start increasing in a U shape.

Exercise 5 : [P] Implementing Logistic Regression Classifier (2+1+2+2+1+1 Points)

In this exercise, you will implement a logistic regression model for predicting binary argument quality. Note that logistic regression performs very poorly on this dataset: you might even get a worse misclassification rate than the random model.

Download and use these files from Moodle:

- `features-train-cleaned.tsv`: Feature vectors for each example in the training set.
- `features-test-cleaned.tsv`: Feature vectors for each example in the test set.
- `quality-scores-train-cleaned.tsv`: Quality scores for each example in the training set.
- `programming_exercise_logistic_model.py`: Template program for writing your implementation. It contains function stubs for each function mentioned below. It should be used like this with the files above:

```
python3 programming_exercise_logistic_model.py
  features-train-cleaned.tsv quality-scores-train-cleaned.tsv
  features-test-cleaned.tsv quality-scores-test-predicted.tsv
```

- `requirements.txt`: Requirements file for the template; can be used to install dependencies.

(a) Implement two functions to load the dataset:

`load_feature_vectors` reads feature vectors from a `features-*-cleaned.tsv` and returns the contained multiset of feature vectors X as an n-by-$(p+1)$ matrix.[1]

`load_class_values` reads the *overall quality* ratings from the `quality-scores-train-cleaned.tsv` as one list of 0s (overall quality score is 1) and 1s (overall quality score is 2 or 3).

How many examples of each class are in the data set?

(b) Implement a function `misclassification_rate` to measure the misclassification rate of the model's predictions.

What is the misclassification rate of a random classifier on the training set?

(c) Implement a function `logistic_function` to calculate the output of the logistic (sigmoid) function w.r.t. input $\mathbf{x}$ parameterized by $\mathbf{w}$ and a function `logistic_prediction` to predict the class of $\mathbf{x}$ accordingly.

(d) Implement a function `train_logistic_regression_with_bgd` that fits a logistic regression model using the Batch Gradient Descent algorithm. A parameter of the function specifies the fraction of training examples to not use for training but for validation. The function returns the trained weights as $p+1$-vector and three lists containing the training loss, misclassification rate on the training examples, and the misclassification rate on the validation examples after each iteration.

(e) Plot the training loss, misclassification rate on the training examples, and the misclassification rate on the validation examples after each iteration.

Are loss and misclassification rate correlated?

(f) Use the trained model to predict the overall quality for each example in the test set (`features-test-cleaned.tsv`). Write the prediction as one column to a file and submit that along with your other solutions.

If you would like to improve your model, here are some hints of what you could try:

- The features you have implemented may have vastly different value scales, which can harm the performance of linear models. For details and a rationale on how that influences the training process, you can watch this video by Andrew Ng. Try to implement one of the feature scaling methods explained in the video.
- Experiment with further hyperparameters and model variants to get the best results, e.g., tweaking the learning rate and number of iterations, selecting subsets of the features, computing additional features as (non-)linear combinations of the existing ones, adding a regularization term, etc.
- Consider implementing k-fold cross-validation instead of using a single hold-out in order to get a more robust estimate of your model's performance on the unseen data.

[1] Recall: $n = |D|$ is the number of examples and p is the number of features. If you can not remember, see the lecture slides on why X has $p+1$ and not p columns.