COMP 3105 – Assignment 2 Report
– Fall 2025 –

Due: Sunday October 19, 2025 23:59.

Group 51
Andrew Wallace - 101210291
Christer Henrysson - 101260693

Getting started

Note that Python 3.10 is used for this assignment. Please install requirements using virtual environment via:

```
python3.10 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Question 1 (5%) Binary Classifier (Primal Form)

In this question, you will implement binary classification with different losses from scratch, in Python using NumPy/SciPy, and evaluate their performances on the synthetic datasets from above with different regularization hyper-parameters. You will learn some essential built-in functions like `scipy.optimize.minimize` to solve unconstrained problems and `cvxopt.solvers.qp` to solve quadratic programmings. The input vectors are assumed to be **un-augmented** in this question (i.e. we do not add a constant feature of 1 to it). All of the following functions must be able to handle arbitrary $n > 0$ and $d > 0$. The vectors and matrices are represented as NumPy arrays. Your functions shouldn't print additional information to the standard output.

(a) (1%) Implement a Python function

```
w, w0 = minExpLinear(X, y, lamb)
```

Please see `A2codes.py` for the implementation of `minExpLinear(X, y, lamb)`

(b) (1%) Implement a Python function

```
w, w0 = minHinge(X, y, lamb, stablizer=1e-5)
```

Please see `A2codes.py` for the implementation of `minHinge(X, y, lamb, stabilizer=1e-5)`

(c) (1%) Implement a Python function

```
yhat = classify(Xtest, w, w0)
```

Please see `A2codes.py` for the implementation of `classify(Xtest, w, w0)`

(d) (1%) In this part, you will evaluate your implementation with different regularization hyper-parameters. Implement a Python function

```
train_acc, test_acc = synExperimentsRegularize()
```

Please see `A2codes.py` for the implementation of `synExperimentsRegularize()`
The

(e) (1%) Looking at your tables from above, analyze the results and discuss any findings you may have and the possible reasons behind them.

Question 2 (5%) Binary Classification (Adjoint Form)

In this question, you will implement binary classification with different losses (again) using the adjoint formula coming from the representer theorem, and evaluate their performances on the synthetic datasets from above with different kernels. The input vectors are assumed to be **un-augmented** in this question (i.e. we do not add a constant feature of 1 to it). All of the following functions must be able to handle arbitrary $n > 0$ and $d > 0$. The vectors and matrices are represented as NumPy arrays. Your functions shouldn't print additional information to the standard output.

(a) (1%) Implement a Python function

```
a, a0 = adjExpLinear(X, y, lamb, kernel_func)
```

(b) (1%) Implement a Python function

```
a, a0 = adjHinge(X, y, lamb, kernel_func, stabilizer=1e-5)
```

(c) (1%) Implement a Python function

```
yhat = adjClassify(Xtest, a, a0, X, kernel_func)
```

(d) (1%) In this part, you will evaluate your implementation with different kernels. Implement a Python function

```
train_acc, test_acc = synExperimentsKernel()
```

(e) (1%) Looking at your tables from above, analyze the results and discuss any findings you may have and the possible reasons behind them.

Question 3 (5%) Binary Classification (SVM Dual Form)

In this question, you will implement binary classification with the hinge loss (yet again) using the dual formula, and choose the best hyper-parameter and kernel for some real-world problems via cross-validation. The input vectors are assumed to be **un-augmented** in this question (i.e. we do not add a constant feature of 1 to it). All of the following functions must be able to handle arbitrary $n > 0$ and $d > 0$. The vectors and matrices are represented as NumPy arrays. Your functions shouldn't print additional information to the standard output.

(a) (1%) Implement a Python function

```
a, b = dualHinge(X, y, lamb, kernel_func, stabilizer=1e-5)
```

(b) (1%) Implement a Python function

```
yhat = dualClassify(Xtest, a, b, X, y, lamb, kernel_func)
```

(c) (2%) The A2files.zip includes an image dataset, A2train.csv, of handwritten digits taken from the MNIST dataset. Each image is either digit 4 or digit 9 (once loaded, you can call the `plotDigit` function to see some samples of the images as in Fig. 4). The first column of the csv file is the class label. Treat digit 4 as the -1 class and digit 9 as the +1 class, your task is to use your `dualHinge` function to learn a good binary classifier. In this part, you need to perform cross-validation and select the best hyperparameters and kernels for this dataset. Implement a Python function

```
cv acc, best lamb, best kernel = cvMnist(dataset_folder, lamb_list,
                                         kernel_list, k=5)
```

(d) (1%) We will evaluate your choices from (c) on a new test dataset (that you don't have access to). You will get full mark here if your chosen hyper-parameters, kernel function, dualHinge and dualClassify can achieve acceptable performance on the test dataset.