COMP 3105 – Assignment 3 Report
– Fall 2025 –

Due: Sunday November 16, 2025 23:59.

Group 51
Andrew Wallace - 101210291
Christer Henrysson - 101260693

Getting started

Note that Python 3.11 is used for this assignment. Please install requirements using virtual environment via:

```
python3.11 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Question 1 (4%) Linear Multi-Class Classifier

(a) (1%) Implement a Python function
`W = minMulDev(X, Y)`
Please see `A3codes.py` for the implementation of `W = minMulDev(X, Y)`

(b) (1%) Implement a Python function
`Yhat = classify(Xtest, W)`
Please see `A3codes.py` for the implementation of `Yhat = classify(Xtest, W)`

(c) (1%) Implement a Python function
`acc = calculateAcc(Yhat, Y)`
Please see `A3codes.py` for the implementation of `acc = calculateAcc(Yhat, Y)`

(d) (1%) In this part, you will evaluate your implementation on the synthetic datasets from above. The results from the synthetic classification experiment using seed 101210291 report the following training accuracies:

n	Model 1	Model 2
16	0.93125	1
32	0.88125	0.984375
64	0.86875	0.9609375
128	0.85390625	0.94296875

Table 1: Training accuracies with different number of training dataset sizes

The results from the synthetic classification experiment using seed 101210291 report the following test accuracies:

n	Model 1	Model 2
16	0.7317	0.8963
32	0.7928	0.8857
64	0.8228	0.8976
128	0.844	0.9178

Table 2: Test accuracies with different number of training dataset sizes

$$W^* = \text{argmin}_W \in \mathbb{R}^{d \times k} \frac{1}{n} \sum_{i=1}^{n} \log(1_k^{\mathrm{T}} \exp^{W^T x_i}) - y_i^{\mathrm{T}} W^{\mathrm{T}} x_i$$

Our results show that the training for our linear classifier is less accurate for data generated with data model 1 compared to data generated with model 2. This is because the data in model 1 has more overlap compared to model 2. This makes the data classes

more difficult to linearly separate. We also see a trend towards lower accuracies as the number of data points increases. Again, this is possibly due to the difficulty of linearly separating classes when there is more overlapping data points that belong to different classes. However, this reduces overfitting in the learned classifier, generalizing better to the test dataset. That is, as the number of data points increases in the test dataset, the accuracies increase due to this generalization in the classifier.

Question 2 (7%) Principle Component Analysis

(a) (1%) Implement a Python function
`U = PCA(X, k)`
Please see `A3codes.py` for the implementation of `PCA(X, k)`

(b) (0.5%) Implement a Python function
`Xproj = projPCA(Xtest, mu, U)`
Please see `A3codes.py` for the implementation of `projPCA(Xtest, mu, U)`

(c) (2%) Implement a Python function
`A = kernelPCA(X, k, kernel func)`
Please see `A3codes.py` for the implementation of `kernelPCA(X, k, kernel func)`

(d) (2%) Implement a Python function
`Xproj = projKernelPCA(Xtest, Xtrain, kernel func, A)`
Please see `A3codes.py` for the implementation of `projKernelPCA(Xtest, Xtrain, kernel func, A)`

(e) (1%) In this part, you will evaluate your implementation on the synthetic datasets from above. Implement a Python function
`train_acc, test_acc = synClsExperimentsPCA()`
Please see `A3codes.py` for the implementation of `synClsExperimentsPCA()`
The results from the synthetic PCA classification experiment using seed 51 report the following training accuracies:

Dim k	Model 1	Model 2
1	0.8503125	0.64242188
2	0.85789063	0.94179687

Table 3: Training accuracies for PCA classification with different dimension sizes

The results from the synthetic PCA classification experiment using seed 51 report the following test accuracies:

Dim k	Model 1	Model 2
1	0.84541	0.63417
2	0.83993	0.91801

Table 4: Test accuracies for PCA classification with different dimension sizes

(f) (0.5%) Looking at your tables from above, analyze the results and discuss any findings you may have and the possible reason behind them.

Question 3 (4%) k-means

(a) (1%) Implement a Python function
`Y, U, obj val = kmeans(X, k, max iter=1000)`

(b) (1%) Implement a Python function
`Y, U, obj val = repeatKmeans(X, k, n runs=100)`

(c) (1%) Implement a Python function
`obj val list = chooseK(X, k candidates=[2,3,4,5,6,7,8,9])`

(d) (2%) Implement a Python function
`Xproj = projKernelPCA(Xtest, Xtrain, kernel func, A)`

References

λ	Linear	Poly(d=2)	Gauss(σ=1.0)
0.001	0.958	**0.978**	0.493
0.01	0.958	**0.978**	0.493
0.1	0.958	**0.978**	0.493

Table 5: Q3(c) average validation accuracies for MNIST (4 vs 9). Best setting: λ=0.001, Poly(d=2).