Md. Mahin, Steve Aigbe and Christoph F. Eick

3

**Jason Mullen**

**4-1-23**

COSC 4368: Fundamentals of Artificial Intelligence Spring 2023

ProblemSet2 (Individual Tasks)

First Draft

Submission deadline: Task 3: April 1; Task 4: April 9, 2023

Last updated: March 13

Allocated points to ProblemSet2: Task3: 27 points; Task 4: 35 points.

Allocated points to ProblemSet2 are tentative and subject to change.

**3. Basic Supervised Learning Task: Predicting heart disease using SVM and MLP**

In this assignment, you will work with a dataset containing information on patients with and without heart disease. Your goal is to build and compare two models for predicting whether a patient has heart disease: one using SVM and one using MLP.

**Data:**

The dataset contains 303 instances and 14 attributes, including age, sex, chest pain type, resting blood pressure, serum cholesterol level, fasting blood sugar level, maximum heart rate achieved, and more. The target variable is a binary variable indicating whether the patient has heart disease or not.

You can download the dataset (*processed.cleveland.data*) from the UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

**Tasks:**

1. Split the data into train and test sets.
   1. **Completed**
2. Build an SVM model to predict whether a patient has heart disease. Train the SVM model on the train set using the given kernel functions (linear, rbf, sigmoid). You can use the same Gamma value for the non-linear kernel functions. Then, use the test set to obtain model prediction accuracy. Generate a table, as given below, for the obtained results.
   1. **Completed**

|  |  |
| --- | --- |
| **SVM Experiments** | |
| **Kernel Function** | **Accuracy** |
| linear | 0.75000 |
| rbf | 0.53333 |
| sigmoid | 0.53333 |

1. Build an MLP model to predict whether a patient has heart disease. Train the MLP model with the train set using the following hyperparameters: activation functions (relu, tanh), optimizers (SGD, Adam) and learning rates (0.01, 0.001). Then use the test set to obtain the model prediction accuracy. Generate a table that summarizes the obtained results as given below.
   1. **Completed**

|  |  |  |
| --- | --- | --- |
| **MLP Experiments - relu** | | |
| **Optimizer** | **Learning Rate** | **Accuracy** |
| SGD | 0.01 | 0.533333 |
| SGD | 0.001 | 0.550000 |
| Adam | 0.01 | 0.683333 |
| Adam | 0.001 | 0.666667 |

|  |  |  |
| --- | --- | --- |
| **MLP Experiments – than** | | |
| **Optimizer** | **Learning Rate** | **Accuracy** |
| SGD | 0.01 | 0.566667 |
| SGD | 0.001 | 0.550000 |
| Adam | 0.01 | 0.600000 |
| Adam | 0.001 | 0.716666 |

1. Interpret the tables you generated in steps 2 and 3; compare the performance of the SVM and MLP models. Which model performs better? Why do you think that is the case?
   1. **Interpretation of the table:** 
      1. **SVM table:**
         1. For that table we see that the linear Kernel has yielded the highest accuracy. This tells us that the data is more linearly separable than not. Similarly, linear Kernels are computationally more efficient and optimal when it comes to overfitting. Thus, we can tell based on the SVM Model that the data is set up in a particular way.
      2. **MLP Table:**
         1. **Relu:**
            1. We find that With Relu SDG, that the best results are produced when we decrease the learning rate. The difference is 2%.
            2. While with the Adam Optimizer we find the opposite. The Data shows us that the smaller learning rate produces better accuracy by around 2%.
         2. **Than:**
            1. From the table we can see that when to comes to SGD, the larger learning rate improves the accuracy but only marginally.
            2. While when it comes to the Adam optimizer, decreasing the learning rate size made a significant difference by about 10%
   2. **Comparison of SVM and MLP models:**
      1. When it comes to the comparisons of the tables, we see that SVM has the best accuracy. This is because SVM works best on small datasets that are linearly separable and contain a relatively small number of parameters. So as a result, Linear SVM is approximately 4% more accurate.
   3. **Which Performs better:**
      1. Linear SVM performs better than “Than MLP” with the Adam optimizer.
   4. **My thoughts:**
      1. Regarding the outcome of the Models the results are as expected. Considering it’s a small dataset, with a low number of parameters, and the data being relatively linearly separable it makes logical sense that Linear SVM yields the highest model Accuracy.

## **Deliverables:**

1. A Jupyter notebook with your code and analysis.
2. A brief report (1-2 pages) summarizing your findings and conclusions. This should include a discussion of the strengths and weaknesses of the SVM and MLP models, and recommendations for further improvement.

**Suggestions:**

You can use built-in functions in python and R.

For python, it is preferable to use Scikit-Learn for both SVM and MLP (see the scikit-learn documentation)

For R, we suggest the ‘mlp’ and ‘svm’ functions.

**4. Deep Learning and Generative Models**

**Learning Objectives**:

1. Learn to use deep learning and generative models such as VAE
2. Learn to use classifiers
3. Learn differn tools to create different deep learning models
4. Learning how to interpret quality of models

|  |  |
| --- | --- |
| **A: Variational Autoencoder** | **B: Normal Autoencoder** |
| C:\Users\mdmah\Google Drive\UH\TA\AI\Spring 2023\Picktures\VAE.png | C:\Users\mdmah\Google Drive\UH\TA\AI\Spring 2023\Picktures\autoencoder_header.png |

**Figure1:**

**A**: A sample Variational Auto-encoder. The VAE contains one encoder and one decoder part. Encoder starts from x,h and ends in z=(σ + μ). [(σ + μ) learns latent representation or key features of the images]. Decoder starts from z=(σ + μ) to h2 and ends in x2. Decoder utilizes learned important represntation from z=(σ + μ) and tries to regenerate the image in x2.

**B**: A sample normal auto-encoder. A normal autoenoder contains only a fully connected layer z instead of a pair of layers (σ + μ) to learn the hidden representation.

In this project we will use the MNIST computer vision digit dataset and experiment with auto-encoders such as Variational Auto-encoder(VAE) and simple autoencoder. The Jupiter notebook provided contains a VAE architecture and process of downloading the dataset.

**Task 4 Subtasks:**

1. Learn latent features from the digit dataset. Use the model given in reference [1]. Perform the following tasks:
   1. (\*\*) The given model has a three layer architecture for each encoder and decoder part. Can you modify the architecture into a four layer format. In this task, you need to convert encoder part into (x, h1, h2, z=(σ + μ) )= (784\*400\*200\*20) and decoder part into (z=(σ + μ), h3, h4, x2 )= (20, 200, 400, 784). Finally you need to compare the results based on their:
      1. Optimal loss after the model is fully trained, and
      2. Visually inspecting the output they generate using the images they generate and reconstruct. You can use plot\_generation() and plot\_reconstruction() function from the notebook.

Based on optimal loss and visual inspection write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

* 1. (\*) Take the base three layer architecture and check the performance of the model for six different configuration, where h\_dim and z\_dim is changed into following patterns: [(400,20), (400, 10), (400, 30), (300,20), (300, 30), (300,30)](Note: First one is the base architecture). Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and write down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.
  2. (\*\*) Take the base three layer architecture and convert it into a normal auto-encoder (figure 1.b)[2], e.g. replace z\_dim such a way that it will be single layer. As noraml autoencoder and variational auto-encoder have very different way of loss calculation, you need to modify loss function too. Now Perform the same type of comparison you have done in task **a** using optimal loss of the model and visual inspection and down your opinion which model is better and also try to give an explanation why a model is giving good performance over another.

**References:**

1. <https://github.com/dataflowr/notebooks/blob/master/HW3/VAE_clustering_empty.ipynb>
2. <https://www.analyticsvidhya.com/blog/2021/06/complete-guide-on-how-to-use-autoencoders-in-python/>