Md. Mahin, Theodoros Tavoulareas and Christoph F. Eick

COSC 4368: Fundamentals of Artificial Intelligence Spring 2021

ProblemSet2 (Individual Tasks)

Submission deadline: Task 4: Monday, April 5; Task 5: Wednesday, April 21

Last updated: March 30

Allocated points to ProblemSet2: TBDL

Allocated points to ProblemSet2 are tentative and subject to change.

**4. Using SVM and NN Tools** *Theodoros*

The goal of this task is to apply different classification approaches to a challenging dataset to compare the results, to enhance the accuracy of the learnt models via selecting better parameters/preprocessing/using kernels/incorporating background knowledge and to summarize your findings in a report. For this problem, we will use the [UCI Machine Learning Repository](http://archive.ics.uci.edu/ml/index.php) and particularily the [Heart failure clinical records](https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records) dataset. This is a classification problem. The goal of this task is to learn binary classification models that distinguish the two classes (in this case mortality event or not). Please read carefully the dataset description on the UCI website as it gives information about the different attributes.

As far as classification algorithms are concerned, we will use:

1. Neural Networks (Multi Layer Perceptron – MLP)

2. Support Vector Machines.

You will use 2 “variations” of each approach:

* For the SVM, you should use 2 different kernels (any kernel is fine, you can use the linear kernel as one kernel)
* For the MLP, you should use two of the following activation functions: 1. Logistic/sigmoid 2. Tanh and 3. Relu

Accuracy of the four classification algorithms, you compare, should be measured using 10-fold cross validation. In your report after comparing the experimental results, write a paragraph or two trying to explain/speculate why, in your opinion one classification algorithm outperformed the other. Finally, at the end of your report write two paragraphs which summarize the most important findings of this task.

**Deliverables:**

Please submit both the report and the source code file.

**Suggestions:**

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

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

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

**5. Generating MNIST Handwritten Digits with GANs** *Theodoros*

The purpose of this taks is to get familiarized with the concepts of Generative Adversarial Networks (GANs). In this task, you are asked to apply the GAN pipeline that is described in this [article](https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/). You will discover how to develop a generative adversarial network with deep convolutional networks for generating handwritten digits using the MNIST dataset. Please follow the steps of the article very thoroughly. In general, you will learn: (i) How to define and train the standalone discriminator model for learning the difference between real and fake images; (ii) How to define the standalone generator model and train the composite generator and discriminator model;

(iii) How to evaluate the performance of the GAN and use the final standalone generator model to generate new images.

**Questions:**

1. **TanH Activation and Scaling**: Update the example to use the tanh activation function in the generator and scale all pixel values to the range [-1, 1].
2. **Change Latent Space**: Update the example to use a larger or smaller latent space and compare the quality of the results and speed of training
3. **Label Smoothing**: Update the example to use one-sided label smoothing when training the discriminator, specifically change the target label of real examples from 1.0 to 0.9, and review the effects on image quality and speed of training.
4. A common issue in GANs is the following: if a generator produces an especially plausible output, the generator may learn to produce only that output. In fact, the generator is always trying to find the one output that seems most plausible to the discriminator. How can we make the generator broaden its scope?
5. GANs frequently fail to converge. What are some methods that can improve GAN convergence?

**Deliverables:**

* Code
* Answers to the questions

**Bonus Questions:**

1. **Batch Normalization**: Update the discriminator and/or the generator to make use of batch normalization, recommended for DCGAN models.
2. **Model Configuration**: Update the model configuration to use deeper or more shallow discriminator and/or generator models, perhaps experiment with the UpSampling2D layers in the generator.