Applied Machine Learning (874)

Post-block Assignment 3

Department of Industrial Engineering

Stellenbosch University

7 June 2021

Responsible Lecturer: Mr LE Burger (eldonburger@sun.ac.za)

Deadline: 20 June 2021 @ 23:59

Total: 180 marks

Instructions

1. The assignment tests your understanding of the concepts covered in topics 5, 7, 8, 9 and 10.
2. Your assignment should be submitted as **one pdf** document:
    a. Name the document as ???????PBA3.pdf, where you replace the question marks with your student number.
    b. In your final document, provide a heading for every question based on the notation of the document followed by your answer. For example, **Question 1** followed by your answer.
3. When completing the assignment, keep in mind that:
    a. Questions that ask for calculations will not be evaluated if only the answers are provided.
    b. When asked to provide an extract of your code, only provide the relevant code. The provided code should be readable, standalone, and consistent. Make use of proper spacing, variables names and include comments when applicable. You can use any programming language of your choice. Your code does not have to be optimal.
    c. All answers should be your own work. Plagiarism will not be tolerated.
    d. Of the total marks, ten marks will be allocated based on the overall quality of your report.
4. **Late submissions cannot be accepted** and no extensions to the deadline will be provided.

1. Radial basis function neural networks (RBFNN) [40]

Consider a radial basis function neural network (RBFNN) designed for a binary classification task. The RBFNN structure consists of:

- An input layer with three units (two input and one bias unit).
- A single hidden layer with thirteen units. Each unit implements the Gaussian function. The centre u_j and width σ_j of each Gaussian function has been predetermined and is provided in Table 1.
- An output layer with a single neuron. The single neuron implements the sigmoid activation function. The output of the network is converted to a classification label as follows:
    - When the output of the network is equal to or greater than 0.5, predict the label 1.
    - When the output of the network is less than 0.5, predict the label 0.

Note: The neuron in the output layer is connected to every unit in the hidden layer by the weight vector $\boldsymbol{w}$ (no bias unit)**.**

Table 1: Precalculated RBFNN weights

j	$\boldsymbol{u_{j,1}}$	$\boldsymbol{u_{j,2}}$	$\boldsymbol{\sigma_j}$
1	0.3	0.75	0.08
2	0.65	0.75	0.08
3	0.7	0.45	0.09
4	0.4	0.6	0.07
5	0.9	0.65	0.09
6	0.4	0.5	0.05
7	0.6	0.55	0.06
8	0.95	0.5	0.06
9	0.18	0.53	0.07
10	0.15	0.85	0.07
11	0.5	0.85	0.07
12	0.7	0.95	0.06
13	0.8	0.85	0.07

In this assignment, we will train the RBFNN on the "*RBFNN_train.csv*" data set using **batch gradient descent** so that the sum of squared error L_2 is minimised. Formally L_2 is defined as

$$L_2(\mathcal{M}_{w,\mu,\sigma}, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \mathcal{M}_{w,\mu,\sigma}(\boldsymbol{d}_i)\right)^2 \quad \text{(Eq.1)}$$

where the training set is composed of n training instances; each training instance is composed of descriptive features $\boldsymbol{d}$ and a target feature t; $\mathcal{M}_{w,u,\sigma}(\boldsymbol{d}_i)$ is the prediction made by the RBFNN for a training instance with descriptive features $\boldsymbol{d}_i$; and the RBFNN is defined by the model parameters w, μ and σ.

For the exercise assume that:

- The centre u_j and width σ_j of each Gaussian function is fixed. In other words, do not adjust these model parameters.
- The weight vector $\boldsymbol{w}$ is initialised to the values:
$$[0.1, -0.1, 0.1, -0.1, 0.1, -0.1, 0.1, -0.1, 0.1, -0.1, 0.1, -0.1, 0.1]$$
- Overfit the data set. You do not have to perform any form of cross validation

Question 1: Calculate the output of the fourth hidden neuron i.e. $j = 4$ for the input $z = [0.5, 0.5]$. Provide your final answer rounded to three decimal places. Provide your supporting calculations. [2]

Question 2: Calculate the output of the remaining hidden units for the input $z = [0.5, 0.5]$. Complete the table below, round your answers to three decimal places. [2]

Hidden unit	Output
y_1	
y_2	
y_3	
y_4	-----
y_5	
y_6	
y_7	
y_8	
y_9	
y_{10}	
y_{11}	
y_{12}	
y_{13}	

Question 3: Calculate the output of the RBFNN for the input $z = [0.5, 0.5]$. Provide your final answer rounded to three decimal places. Provide your supporting calculations. [2]

Question 4: Calculate the sum of squared error (defined by Eq.1) for the input $z = [0.5, 0.5]$ given that the label (target) for the input is one. Provide your final answer rounded to three decimal places. Provide your supporting calculations. [2]

Question 5: Calculate the sum of squared error (defined by Eq.1) for the "*RBFNN_train.csv*" data set. Provide your final answer rounded to three decimal places. [2]

Question 6: Generate a prediction for each instance of the "*RBFNN_train.csv*" data set. Use the results to complete the confusion matrix provided below. [2]

	Actual (1)	Actual (0)
Predict (1)		
Predict (0)		

Question 7: To train the RBFNN using **batch gradient descent**, the weight adjustment step size $\Delta w_j(t)$ needs to be calculated. Derive the weight adjustment step size, given that the network uses the sum of squared error (defined by Eq.1). [4]

Question 8: Develop a function to calculate the new weights for the RBFNN, after an iteration of batch gradient descent. **Provide an extract of your function which clearly indicates how each value with regards to the weight update is calculated**. [10]

Question 9: Apply the function developed in question 8 on the "*RBFNN_train.csv*" data set. Use a learning rate of 0.2. Use the output of your function to complete the provided table. [2]

Weights	Initial weights	New weights
w_1	0.1	
w_2	-0.1	
w_3	0.1	
w_4	-0.1	
w_5	0.1	
w_6	-0.1	
w_7	0.1	
w_8	-0.1	
w_9	0.1	
w_{10}	-0.1	
w_{11}	0.1	
w_{12}	-0.1	
w_{13}	0.1	

Question 10: Train the RBFNN for multiple epochs using an appropriate learning rate. Provide your final weights in the table provided together with the details of your training parameters i.e., learning rate and the number of epochs trained. [4]

Weights	Initial weights	Final weights
w_1	0.1	
w_2	-0.1	
w_3	0.1	
w_4	-0.1	
w_5	0.1	
w_6	-0.1	
w_7	0.1	
w_8	-0.1	
w_9	0.1	
w_{10}	-0.1	
w_{11}	0.1	
w_{12}	-0.1	
w_{13}	0.1	

Question 11: Generate a prediction for each instance of the "*RBFNN_train.csv*" data set (using the trained model). Use the results to complete the confusion matrix provided below. [2]

	Actual (1)	Actual (0)
Predict (1)		
Predict (0)		

Question 12: Create a scatter plot of the "*RBFNN_train.csv*" data set. In your plot differentiate between the instances labelled as 1 and the instances labelled as 0. Add the decision boundary of your trained RBFNN to the plot. [4]

Question 13: Is the following statement true or false: For extreme outliers, the RBFNN will always generate the label 1. Explain your answer. No marks will be allocated for a correct answer without the correct explanation. [2]

2. Self-organising maps [40]

Colour quantisation is used to reduce the number of distinct colours used in an image whilst keeping the new image visually similar to the original image. For example, an image containing thousands of unique colours can be compressed into an image that only contains a few unique colours. Colour quantisation is critical for displaying images that contain multiple distinct colours on devices that can only display a limited number of colours. Figure 1, illustrates an image where the number of distinct colours was reduced from ~30 000 distinct colours to only 25 distinct colours.

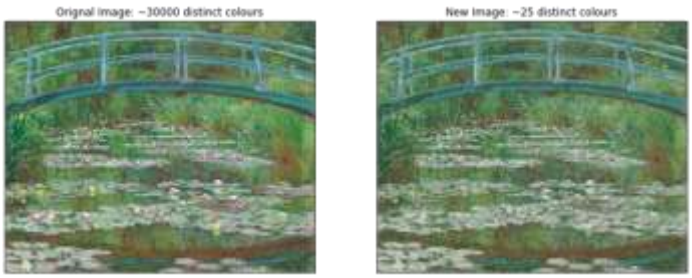

Figure 1: Colour quantisation of The Japanese Footbridge, 1899

We will consider the problem of applying colour quantisation to RGB images. In an RGB image, each pixel is defined by the combination of red, green, and blue intensities. The intensities values for each channel ranges from 0 to 255. By combining different intensity values, roughly 16.7 million unique colours can be represented. Given an RGB image, our goal is to find a smaller set of representative colours. To find a set of representative colours we will make use of self-organising maps.

The first step in training a self-organising map is to define a map structure. The map structure is usually a two-dimensional grid. For the assignment, we will consider a two-dimensional square grid consisting of $K = 5$ rows and $J = 5$ columns. Furthermore, each neuron has a I dimensional weight vector assigned to it. Our input data consist of three dimensions, thus $I = 3$.

The problem of colour quantisation has been broken down into a set of nine questions labelled question 1 to question 9 respectively. Before each question some information is provided as an aid to the student. The information provided is only an aid, and additional research might have to be conducted to answer the question.

Question 1: Create a self-organising map of $K = 5$ rows and $J = 5$ columns. Assign random colours to each neuron of the self-organising map. Display your self-organising map as an image where each neuron in the image is assigned a colour based on its weights. [2]

The initialisation of the weights of the neurons can occur in several different ways. Instead of using random weights, we can define a large enough hypercube to cover all the training patterns. The algorithm to define a hypercube will now be described.

The algorithm starts by finding the four extreme points of the map. The first two extreme points, the upper right corner (w_{1J}) and the bottom left corner (w_{K1}) is located at the two patterns with the largest inter-pattern Euclidean distance. The third extreme point, the upper left corner (w_{11}) is located at the pattern which is the furthest away (largest Euclidean distance) from the two extreme points found in the previous step. The last extreme point is located at the pattern which is the furthest away from all three extreme points.

Question 2: Table 2 contains a set of colour names along with their respective RGB values. Write a function that can determine the four extreme points using the data provided in Table 2. Provide an extract of your function and use the output of your function to complete the provided table. [4]

Table 2: RGB values

Colour	R	G	B
Red	255	0	0
Green	0	128	0
Blue	0	0	255
D Green	0	100	0
D Blue	0	0	139
Yellow	255	255	0
Orange	255	165	0
Purple	128	128	128

Extreme point	Colour
w_{11}	
w_{1J}	
w_{K1}	
w_{KJ}	

Given the four extreme points, the weights of the boundary neurons are initialised as:

$$w_{1j} = w_{11} + (j-1)\frac{w_{1J} - w_{11}}{J-1} \quad \text{(Eq.2)}$$

$$w_{Kj} = w_{K1} + (j-1)\frac{w_{KJ} - w_{K1}}{J-1} \quad \text{(Eq.3)}$$

$$w_{k1} = w_{11} + (k-1)\frac{w_{K1} - w_{11}}{K-1} \quad \text{(Eq.4)}$$

$$w_{kJ} = w_{1J} + (k-1)\frac{w_{KJ} - w_{1J}}{K-1} \quad \text{(Eq.5)}$$

for all $j = 2, \ldots, J-1$ and $k = 2, \ldots, K-1$.

The remaining codebook vectors are initialised as:

$$w_{kj} = w_{k1} + (j-1)\frac{w_{kJ} - w_{k1}}{J-1} \quad \text{(Eq.6)}$$

for all $j = 2, \ldots, J-1$ and $k = 2, \ldots, K-1$.

Question 3: Write a function that can determine the remaining initialisation values using the extreme point found in question 2. Use the output of your function to complete the provided table. Provide an extract of your function. [4]

Hypercube point	R	G	B
$w_{1,3}$			
$w_{3,1}$			
$w_{3,5}$			
$w_{3,3}$			

Question 4: Similar to question 1 create a self-organising map of $K = 5$ rows and $J = 5$ columns. Assign colours to each neuron of the self-organising map by using the hypercube approach. Display your self-organising map as an image where each neuron in the image is assigned a colour based on its weights. [2]

Given that we have defined a self-organising map and discussed how to initialise the weights, as a next step we need to decide how to train the self-organising map. For the task of colour quantisation, we will use the **fast batch map** algorithm discussed in topic 8: unsupervised learning. Like the stochastic training algorithm, the fast batch algorithm requires (i) a method to compute the winning neuron and (ii) a neighbourhood function. The winning neuron is defined as the neuron which has the smallest Euclidean distance between the neuron's weights and an instance.

Question 5: Table 3 represents a self-organising map where each cell of the table contains the weight values of the respective neuron. Write a function that can determine the coordinates (row and column indexes) of the winning neuron given the self-organising map provided in Table 2 and the instance [0, 128, 0]. Provide an extract of the function along with the coordinates of the self-organising map. Note: row and column indexes are indexed from 1 to 5 and not 0 to 4. [3]

Table 3: Self-organising map weights

	1	2	3	4	5
1	244,179,201	186,42,57	252,73,127	63,137,134	21,238,33
2	102,139,71	27,82,98	135,55,147	87,30,55	175,166,238
3	133,0,143	43,88,249	44,82,61	146,129,216	208,23,70
4	251,175,27	233,84,244	35,53,64	159,169,126	169,19,212
5	131,154,160	64,174,199	117,171,100	106,215,221	125,103,7

The neighbourhood function influences how much a neuron's weight is adjusted based on its proximity to the winning neuron. A popular choice for the neighbourhood function is the Gaussian function.

The Gaussian function is defined as:

$$h_{mn,kj} = \eta(t)e^{\frac{-\|c_{mn}-c_{kj}\|_2^2}{2\sigma(t)^2}} \quad \text{(Eq.7)}$$

where:

- $\eta(t)$ is the learning rate,
- $\sigma(t)$ is the width of the kernel,
- c_{mn} is the coordinates of the winning neuron
- c_{kj} is the coordinates of the neuron located at coordinates k, j.

Question 6: Write a function that calculates the output of the neighbourhood function given: (i) a learning rate, (ii) the width of the kernel (ii) the coordinates of the winning neuron and (iv) the coordinates of a given neuron c_{kj}. Apply your function to the self-organising map provided in Table 3, for a learning rate equal to one, a width equal to two and the winning coordinates [2,2]. Use the output of your function to populate the missing values in the table provided, given that each cell represents the evaluation of the neighbourhood function. Provide an extract of your function. [3]

	1	**2**	**3**	**4**	**5**
1	0.779	0.882	0.779	0.535	0.287
2	0.882		0.882	0.607	0.325
3	0.779	0.882		0.535	0.287
4	0.535	0.606	0.535		0.197
5	0.287	0.325		0.197	0.105

Since the learning process of a self-organising map is iterative, we require a stopping criterion. The quantisation error is usually used as an indication of how accurate the self-organising map is. The quantisation error is defined as the sum of Euclidean distances of all instances to the codebook vector of the winning neuron i.e.

$$\varepsilon_T = \sum_{p=1}^{p_T} \|z_p - w_{mn}(t)\|_2^2 \quad \text{(Eq.8)}$$

Question 7: Write a function that can calculate the quantisation error given a self-organising map and data instances. Apply your function to the self-organising map provided in Table 3 and the data set provided in Table 2. Provide an extract of your function along with the quantisation error rounded to two decimal units. [2]

With the basics defined we can now proceed to train a self-organising map for a specific image. Implement the fast batch-map algorithm with the following modifications:

- Weights should be initialised using random values.
- An appropriate strategy must be added to modify the learning rate and the width of the kernel per epoch.
- The quantisation error must be calculated per epoch.

- The learning rate, the width of the kernel and the quantisation error must be recorded per epoch.

Question 8: Apply your batch map algorithm on the image "*self-potrait_1998.74.5.jpg"* until a sufficient quantisation error is reached. Provide:

- An extract of your function.
- A graph plotting the quantisation error vs epoch number.
- A graph plotting the learning rate vs epoch number.
- A graph plotting the width of the sigma vs epoch number.
- An image of your self-organising map where each neuron in the image is assigned a colour based on its weights.
- An interpretation of the results. [16]

Given the trained self-organising map, as a final step, we can perform colour quantisation. A rather simple approach would be to replace each image value with the best matching unit in the self-organising map.

Question 9: Create a new image of the image "*self-potrait_1998.74.5.jpg"* by using the self-organising map created during question 8. To illustrate that your function works, provide a unique count of image intensities of the original image as well as a unique count of image intensities for the new image. Provide an extract of your function. [4]

3. Case Study: Document Classification Part 1 [60]

ArXiv (https://arxiv.org/) is a free distribution service and open-access archive. ArXiv hosts more than 1.8 million scholarly articles in various fields. When a paper is submitted to ArXiv, authors must select the most applicable field and subject area. For example, the field *computer science* and the subject area *artificial intelligence*. Before the paper is added to the archive, moderators must approve the paper by assessing whether the submission is topical to the subject area and if the paper is of scholarly value.

To speed up the moderation process ArXiv has approached you to develop a classification model that can predict the subject area of a paper using the title and/or abstract of the paper. An example of a paper is provided in Figure 2. In the future, papers can be automatically checked for topical relevance using your model, speeding up the moderation process.

Title: Neural Networks for Handwritten English Alphabet Recognition

Abstract: This paper demonstrates the use of neural networks for developing a system that can recognize hand-written English alphabets. In this system. each English alphabet is represented by binary values that are used as input to a simple feature extraction system. whose output is fed to our neural network system.

Subject Area: Artificial Intelligence

Figure 2: Example article, extracted from ArXiv

To develop the classification model, a data set consisting of approximately 100 000 papers published in the *computer science* field has been collected ("*arxiv2017.csv*"). For every paper, the published date, title, abstract and primary subject area are provided. To simplify the task, you will only have to build a binary classification model.

The specific label that your model must predict is based upon the last digit of your student number. For example, a student whose student number ends with the number zero must build a binary classification model that can predict the category *artificial intelligence*. The assigned categories are provided in Table 4.

Table 4: Assigned category

Code	Primary subject area	
0	AI	Artificial Intelligence
1	CL	Computation and Language
2	CR	Cryptography and Security
3	CV	Computer Vision and Pattern Recognition
4	DC	Distributed, Parallel, and Cluster Computing
5	DS	Data Structures and Algorithms
6	IT	Information Theory
7	LG	Machine Learning
8	LO	Logic in Computer Science
9	NI	Networking and Internet Architecture

Using the *arxiv2017.csv* data set, build a random forest classifier that can predict whether an unseen paper belongs to your assigned category or not. Create a report that covers the following information:

- **Introduction**: Provide a summary of the business objective. In your summary indicate which subject area is assigned to you. [5]
- **Objectives and validation strategy**: Provide a detailed description of how you intend to measure the performance of your model. You should include (i) the measures you intend to use and why you selected these measures as well as (ii) how you intend to evaluate whether your model will generalise to new data. [15]
- **Data pre-processing:** Provide a summary of how you extracted features from the provided data set. Indicate which features you believe would be relevant to the task, based on exploratory analysis. [15]
- **Initial model and evaluation:** Describe your initial model. Evaluate your model based on your validation strategy. Based on the results, provide a list of ideas that could potentially improve the performance of your model. Each idea should be justified based on the performance of your model. [15]
- **Optimal model:** Provide your final model. Describe how you obtained the optimal model. Evaluate your final model based on your validation strategy. Discuss the performance of your model with regards to the business problem. [10]

Your report should use the same sections e.g., *Introduction* and should not exceed four pages.

4. Case Study: Document Classification Part 2 [30]

The business owners of ArXiv are concerned that the model you developed in part 1 of the document classification task can become obsolete over time. To test whether the concern is valid we have collected additional papers published after the ("*arxiv2017.csv*") data set. The new data contains approximately 100 000 unseen papers ("*arxiv2019.csv*").

Question 1: Evaluate your final random forest classifier on the "*arxiv2019.csv*" data set. Plot the performance of the model at user-selected time-steps. [5]

Question 2: Provide a critical analysis of the performance obtained in question 1. Your analysis should include plausible reasons for the change in performance. [5]

Question 3: Develop a method that can be used to adjust the random forest model to automatically react to concept drift. Describe your method. Test your method on the "*arxiv2019.csv*" data set and compare your results to the results obtained in question 1 [20]

.