CITS5508 Machine Learning Sample Exam paper
Semester One 2021

Tip for studying for the exam:

- Study the lecture notes. Also read the textbook – focus on the sections that are covered in the lectures.
- Try the exercises at the end of each chapter in the textbook. Answers to these exercises can be found in Appendix A of the textbook.

This sample exam paper has 6 questions.

This actual exam paper will have 10 questions, each of which contains two separate (unrelated) parts. Each part is worth 5 marks.
You should attempt ALL questions.
TOTAL: 100 MARKS

Your answers should be written on the question paper in the spaces provided under each question.

Calculators are not permitted.

On average, you should try to spend ∼12 minutes for each question. As we have a small number of students doing the online exam this year, no diagram drawing will be required in all the exam questions.

Note: the blue colour texts are the sample answers. The red texts (not parts of the answers) are extra explanation to help you.

Those texts shown in italics are for clarity only. In the exam, you don't have to write specific terms in italics. You can underline important words (or key words) but this is not necessary either.

---

1. (a) Explain what *feature scaling* is. Give an example for scaling features. Give an example of what could go wrong if feature scaling is not carried out. (5 marks)

   (b) What is the difference between Supervised Learning and Unsupervised Learning? Give an example for each. (5 marks)

   (a) Many ML algorithms do not perform well when the numerical input features have very different scales, so we do feature scaling to transform the data into numbers within ranges which are similar across the different features. For example: Min-max scaling: values shifted and rescaled so that they occupy the full range from 0 to 1 (and not outside that).

Another example: Standardisation: values shifted and rescaled to have zero mean and unit standard deviation.

If feature scaling is not performed, then a potential problem is (one of the following in the exam would suffice):

i) some algorithms such as SVM that are sensitive to different feature scales would be affected (the street that separate the two classes could be narrowed and rotated).

ii) the Gradient Descent optimization can be slowed down if the direction of steepest descent is not clear because the shape of a bowl in the cost surface is elongated along one dimension.

One of the examples (Min-max scaling or Standardisation) above would suffice.

(b) In Supervised Learning, the training data that you feed to the algorithm includes the desired output solutions, called the *labels*.

Example: supervised classification: training a SPAM filter to classify emails into SPAM and non-SPAM (or HAM) using a dataset of emails that have been manually labelled into these two categories.

In Unsupervised Learning, the training data is unlabelled so the tasks that we can do is to learn if the data contains certain patterns or forms a number of clusters.

Examples: using k-means clustering to reveal the clusters formed by the data; analyzing the pattern of the data for possible dimensionality reduction.

One example would suffice.

2. (a) Suppose that you have an N-class classification problem. What would be a suitable output function for your algorithm if $N = 2$? What would this function be if $N > 2$? Briefly explain your answers. (5 marks)

   (b) Explain what the OvO "one versus one" strategy is and what it is used for. (5 marks)

(a) If $N = 2$ then a suitable output function for the training algorithm is the sigmoid function, which produces an output in the range 0..1. This output can be treated as the probability for the target class, e.g., if the two classes in our problem are "dogs" and "cats" and our target class is "dogs", then if the output value is > 0.5, we can consider the output class to be "dogs" (otherwise "cats").

If $N > 2$ then softmax should be used. Suppose that we have 3 classes, then the softmax function outputs an array of 3 probabilities that sum to 1. We can take the largest value as the class predicted.

(b) This is a way of using binary classification techniques in a combined way to do multi-class classification.

For the OvO strategy, you train a binary classifier to distinguish between each pair of classes. So there are $n(n-1)/2$ binary classifiers for n classes. Then when you want to make a prediction on a new data instance you run it through all the classifiers and see which class wins the most duels.

3. (a) Briefly explain what *kernel trick* (that is commonly used in SVM) is. (5 marks)

   (b) Explain the difference between *bagging* and *pasting*. Under what classification and regression algorithms are they commonly used and why are these algorithms/methods so popular? (5 marks)

(a) SVM is a *linear* binary classification algorithm. We can incorporate a suitable kernel to augment our data to a higher-dimensional space, so while SVM produces a linear decision boundary in the new space, it effectively gives a non-linear decision boundary in the original space. However, we don't actually need to do this augmentation. When solving for the equation of the hyperplane in SVM, we can tackle the *dual problem* (instead of the *primal problem*) which involves computing the dot product of each pair of data vectors $(\mathbf{x}_i \cdot \mathbf{x}_j)$. Suppose that we use a 2nd order polynomial kernel Φ. We can equally compute $(\mathbf{x}_i \cdot \mathbf{x}_j)^2$ rather than $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. This is known as the Kernel Trick in that the SVM operates on a higher-dimensional space but we actually never need to augment our data to that space.

Note: the sample answer given above is more than what is required in the exam. It was written in length to help you understand the question in depth. In typing (for the online exam), you can replace $\mathbf{x}_i$ and $\mathbf{x}_j$ by xi and xj; and Φ by Phi.

(b) Both Bagging and Pasting are sampling techniques. The former is *sampling with replacement* and the latter is *sampling without replacement*. They are commonly used in the category of ensemble methods where, rather than having a diverse set of algorithms, we train the *same* algorithm (typically, a decision tree algorithm) using randomly sampled data. These methods are popular because:

1. the predictors can be trained in parallel using different CPUs, or GPUs or even on different servers.

2. the predictors can be tested also in parallel.

4. (a) Describe what a *random forest classifier* and *extra-trees* are. (5 marks)

   (b) Briefly explain what *manifold* and *manifold learning* mean. (5 marks)

(a) A random forest (RF) classifier is an ensemble of decision tree classifiers, each of which involves the construction of a binary tree for multiclass classification.

Different from the decision tree classifiers, extra randomness is introduced when growing the trees in RF classifiers: instead of searching for the best feature to split a node, the algorithm searches for the best feature among the subset of features that is randomly sampled from all the features.

Extra-Trees stands for "Extremely Randomized Trees". It is a random forest where even the threshold used for splitting a node is randomly generated.

(b) Suppose that we have data in an n-dimensional space. A d-dimensional manifold (where $d < n$) locally looks like a d-dimensional hyperplane in the n-dimensional space. e.g., for the Swiss roll example, $n = 3$ and $d = 2$.

Manifold Learning refers to a family of non-linear dimensionality reduction algorithms which work by modelling the manifold that the training instances lie on.

e.g., to reduce the dimension of the Swiss roll data from 3D to 2D, the algorithm learns to unroll the data.

5. (a) What is *explained variance ratio*? How do you use it? (5 marks)

   (b) What is a *multi-layer perceptron* (MLP)? Design a simple MLP (no need to draw a diagram) and describe the number of connection weights in your MLP that require training. (5 marks)

(a) When using Principal Component Analysis for dimensionality reduction, the Explained Variance Ratio (EVR) tells us the proportion of the data's variance that lies along each principal axis. e.g., we may have the EVR equal to $[0.80, 0.16, 0.04]$ along the 1st, 2nd, and 3rd principal axes for some 3D data. Notice that these numbers sum up to 1. If we want to reduce the dimension such that we retain at least 95% of the variance, then we can use the EVR to determine the reduced dimension. In the example here, we would project the data onto the 1st two principal axes, giving us 96% variance ($0.80 + 0.16 = 0.96$). The data is thus reduced from 3D to 2D.

(b) An MLP is composed of Perceptrons stacked together. It has an *input layer*, one or more layers of *threshold logic units* (TLUs), called the *hidden layers*, and an output layer (also a layer of TLUs).

You can design any MLP you like. Your answer for the number of connection weights of course will depend on the network that you design. Obviously, your network must have at least 1 hidden layer as otherwise your diagram would not be an MLP (it would be a Perceptron instead). Given the time limit in the exam, the best choice is to have just 1 hidden layer. Your MLP will need to have (of course) an input and an output layer.

Note: If your input layer only has 1 neuron then it would look too simple. So a minimum of 2 input neurons is recommended. Your hidden layer should have at least 2 neurons also; otherwise, there is no point having the hidden layer at all. Your output layer can have 1 or more neurons.

Suppose that your MLP is the same as Figure 10-7 (see textbook), which has 2 input neurons, 4 neurons in the hidden layer, and 3 output neurons. The number of connection weights for this MLP is: $(2+1) \times 4$ (from the *input* layer to the *hidden* layer) + $(4+1) \times 3$ (from the *hidden* layer to the *output* layer) $= 12 + 15 = 27$. Don't forget the bias terms!

If your MLP has 2 input neurons, 3 hidden neurons, and 2 output neurons, then the number of connection weights is: $(2+1) \times 3 + (3+1) \times 2 = 9 + 8 = 17$.

In your answer, you should describe the design of your MLP and how you obtain the number of connection weights (rather than just the final number).

6. (a) What is a *pooling* layer? In what network would you find pooling layers? What are the parameters that define a pooling layer? (5 marks)

(b) What is a *stacked autoencoder*? What is meant by "tying the weights" in a stacked autoencoder? (5 marks)

(a) A pooling layer is a layer of a deep network (DNN) where the input (actually the multi-channel feature map) is subsampled to reduce its spatial dimension. Pooling layers are commonly used in *convolutional neural networks* for inputs that are images. To define a pooling layer, we can specify

1. the *pool size*, e.g., a pool size of 2×2 will reduce the height and width of the input by a factor of 2.

2. the *strides* value. This is the stride step size for moving the pooling window along the row/column of each feature map, e.g., strides $= 3$ or strides $= 3 \times 3$ will further reduce the height and width of the input by a factor of 3.

3. the *padding* option, which is either "same" or "valid", for specifying whether pixels should be padded around the edges of the input for the pooling operation.

(b) A stacked autoencoder (or Deep Autoencoder) is an autoencoder having multiple hidden layers. Just like an autoencoder, it has an encoder part and a decoder part; the number of input neurons in the encoder part is the same as the number of output neurons in the decoder part. The typical architecture of a Stacked Autoencoder has a shape that looks like a sandwich: it has a central hidden layer known as the *coding layer*.

Tying the weights means that layers that are symmetrical around the coding layer have their weight matrices being the transpose of each other. This allows the number of trainable parameters to be reduced by almost a factor of 2, as only the weight matrices in the encoder part need to be learned.

The biase terms are not shared between the encoder and decoder. That is why the number of trainable parameters is not exactly halved.