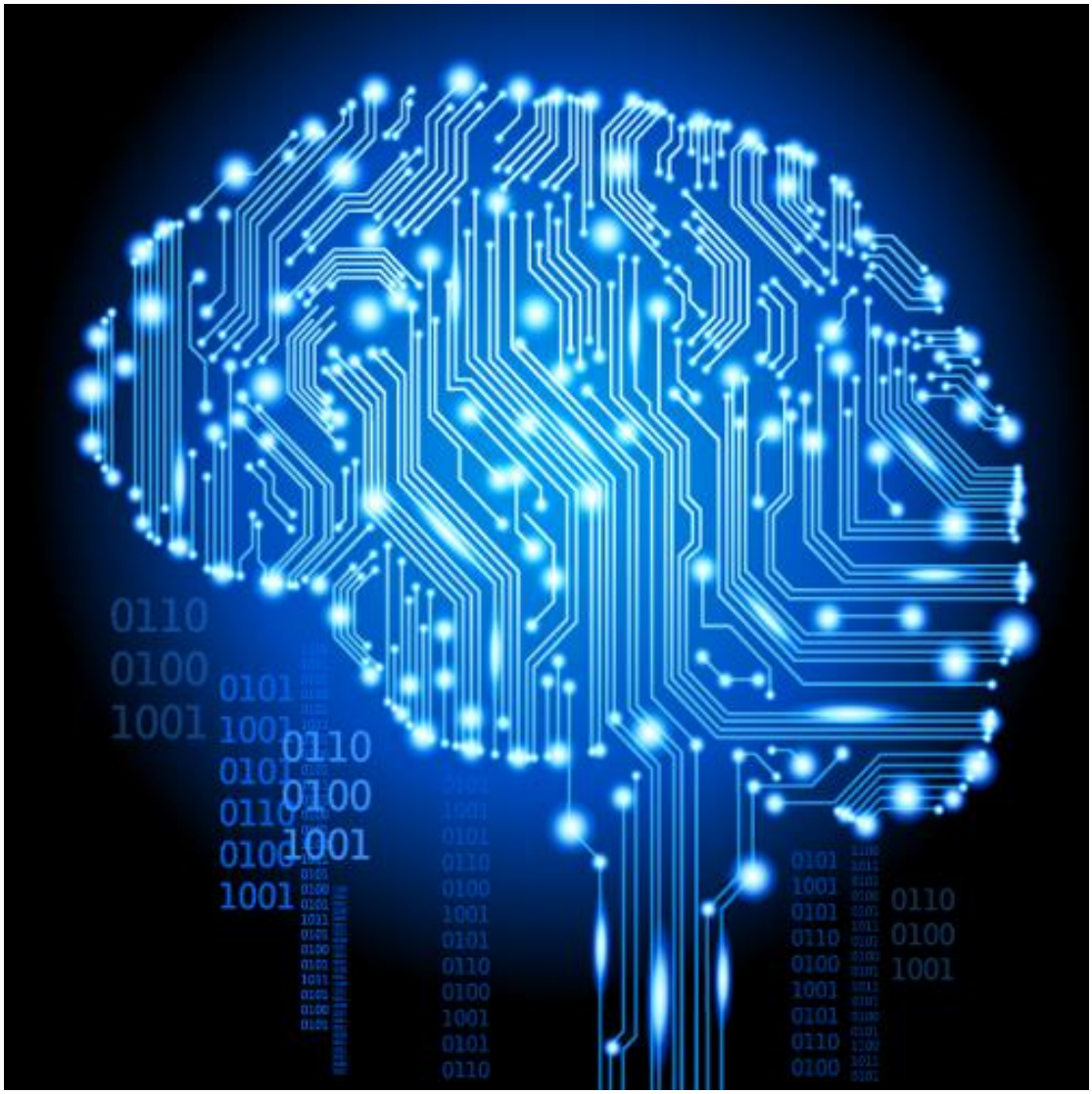
COMP3009 Machine Learning

Computer Based Coursework Manual – Autumn 2021



The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Table of Contents

1. Introduction

The purpose of this Computer-Based Coursework (LAB) is to provide you with hands-on experience in implementing and testing basic machine learning techniques. The techniques that will be assessed are Support Vector Machines (SVM), Decision Trees (DT), Artificial Neural Networks (ANN) and Evaluating Hypotheses. Each of these techniques will be used in order to solve two problems: binary classification and regression. You will pick two datasets from a list of public toy machine learning datasets, and each group will use a different combination of datasets.

2. Organization

2.1 Working Method

Implementation of the algorithms will be done in MATLAB or Python (for TensorFlow). You will work in groups of 5 students (where possible). You are expected to work together on implementation of each machine learning technique and the related emotion recognizer. The groups and the lecture schedule are available on Moodle. The implementation will be either done from scratch (for Decision Trees, and large aspects of Evaluating Hypotheses) or by using dedicated toolboxes (Matlab's SVMs and TensorFlow's ANNs). After an assignment is completed, the Lab Assistants will evaluate the generated code of each group. Your code will have to be submitted to the CS Git repository.

In addition, each group must hand in via Moodle a report of approximately 1,000 words plus results matrices and graphs, explaining details of the implementation process of each algorithm, for each of the two problems, along with comments on the acquired results. The link to your Git repository must also be included so we can mark the code. Detailed deliverable details for every assignment will be described at the end of every section describing the assignment in question.

For each assignment, a zip file with the name of the group should be submitted to Moodle. The zip should include (at least), the following items:

- The written report. It should be in PDF format.
- The corresponding trained models
- The link to your code used to solve each of the assignments
- Code must include data pre-processing, training, and evaluation code

Each group will be responsible to a great extend for the way in which the tasks and the reports are prepared and presented. These reports will provide feedback on the performance of the group as a whole. The individual performance of each group member will be evaluated using peer review following the end of the last lab session.

2.2 Role of the Lab Assistants

The role of the Lab Assistants is to monitor your implementation of the assignments. The Lab Assistants, however, will not make any substantive contribution to the implementation process. Final grading will be exclusively done by the convenor of the course, who will ask for the recommendations of the Lab Assistants concerning the group progress.

2.3 Communication

Communication between the students and the Lab Assistants is very important and will be done in labs during the lab sessions (where possible due to COVID) or via MS Teams.

Please mention your group name and number in any communication; this makes it easier for us to divide the work. In addition, students should visit the Moodle page of the course, which will contain links to the required data files as well as various MATLAB functions needed to complete the assignments of this lab and also many other useful links and information.

2.4 Peer Assessment

After the three assignments are completed so-called *peer assessments* will be held. The members of each group will be asked to complete a form with anonymous feedback on the performance of their group members. This information will be used to make changes to the group mark to obtain an individual grade for every group member.

The Lab Assistant assigned to your group will participate in the peer assessment as an extra group member, using your performance in the interviews to inform their judgment for the peer assessment.

Peer reviews will be disseminated and should be handed in through Moodle after the final coursework hand-in deadline.

2.5 Time Management

In total, there are 3 assessed assignments to be completed. As mentioned before, after the completion of each assignment a 1000 word report must be handed in and a discussion between the Lab Assistants and the group members will take place. The deadlines for handing in assignments can be found on Moodle.

Missing any of the deadlines above will reflect on the final lab grade. Late submissions will result in 5% penalty per day (days rounded up to the next integer). Reports will have to be handed in through Moodle. Only one report needs to be submitted per group. The Report should be accompanied by your source code, so hand in a single .zip archive.

2.6 Grading

After all the assignments are completed and the reports are handed in, every group member will be assigned a grade. Due to peer assessment the final grade can be different for every student. In this lab, we expect each group member to actively participate in the implementation of the algorithms, the reports and the interviews between them and the Lab Assistants. Each individual assignment will be graded based on the code that was developed and the report that was delivered. The final grade m_i for each group member will be a product of the group grade m_g and the group member's personal contribution α_i to the group progress, according to the following formula:

$$m_g = \frac{1}{3}\sum_{l=1}^{3} m_l$$

$$m_i = \alpha_i * m_g$$

Under the constraint that:

$$\frac{1}{n}\sum_{i}^{n} \alpha_i = 1$$

where m_l is the mark for lab *l*, and *n* the number of students in your group.

Attendance to the lab is mandatory and accounts for 30% of the **final grade for the Machine Learning Course**. Failure to attend the lab due to medical or other extenuating circumstances should be registered through the school office as a request for ECF.

3.2 Data

For the first three (unassessed) labs, we provide the following three toy datasets:

- Forest cover type dataset in cov1.mat
- OldFaithful dataset in OldFaithful.mat
- Texas temperature dataset in texas_temp.mat

Please note that the above 3 datasets will be used only for the introductory exercises of the first three lab sessions.

4. System Evaluation

In this section, the basic system evaluation concepts that will be used throughout this tutorial are given. These include:

- K-fold Cross Validation
- The Confusion Matrix
- Recall and Precision Rates
- The F_α-measure

a) Cross Validation

The concept of cross-validation is closely related to overfitting, a concept very important to machine learning. Usually a learning algorithm is trained using some set of training examples, i.e., exemplary situations for which the desired output is known. The learner is assumed to reach a state where it will also be able to predict the correct output for other examples, thus generalizing to situations not presented during training. However, in cases where learning was performed for too long or where training examples are not representative of all situations that can be encountered, the learner may adjust to very specific random features of the training data that have no causal relation to the target function. In the process of overfitting, the performance on the training examples still increases while the performance on unseen data becomes worse. An example of overfitting can be seen in Fig.3. The point where the error on the test set increases is the point where overfitting occurs.

In order to avoid overfitting, it is necessary to use additional techniques, one of which is cross-validation. Cross-validation is the statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, while the other subset(s) are retained for subsequent use in confirming and validating the initial analysis. The initial subset of data is called the *training set*; the other subset(s) are called *validation* or *testing sets*.

The process of cross-validation is schematically depicted in Fig.4, where the initial dataset is split *N* times (*N-fold* cross validation) in order to give *N* error estimates. The final error will be the average of these *N* estimates, as shown in the figure.

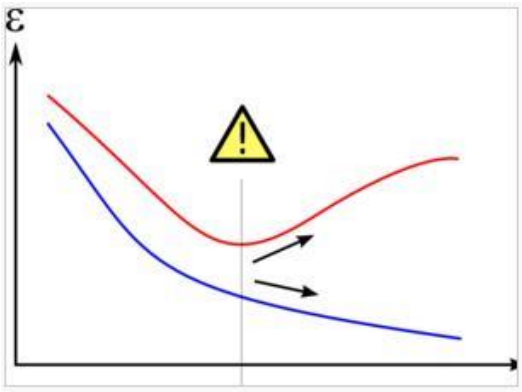


Fig.3: Overfitting/Overtraining in supervised learning (e.g. in a neural network). Training error is shown in blue, validation error in red. If the validation error increases while the training error steadily decreases, then a situation of overfitting may have occurred.

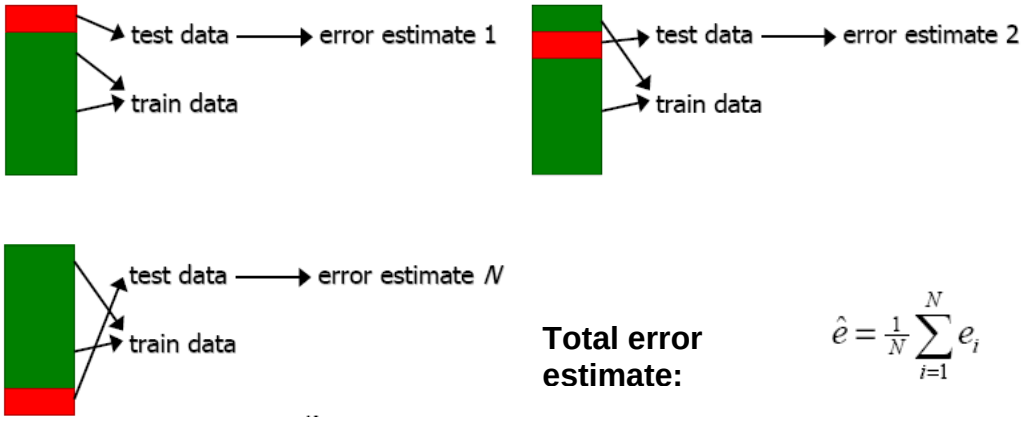


Fig.4: *N-fold* cross validation process. The data is split each time into training and testing datasets. The final prediction error is the mean average error.

b) Confusion Matrix

A confusion matrix is a visualization tool typically used to present the results attained by a hypothesis. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabelling one as other). In the example confusion matrix below (Table 2), of the 8 actual cats, the system predicted that three were dogs, and of the six dogs, it predicted that one was a rabbit and two were cats. We can see from the matrix that the system in question has trouble distinguishing between cats and dogs, but can make the distinction between rabbits and other types of animals pretty well.

	Cat	Dog	Rabbit
Cat	5	3	0
Dog	2	3	1
Rabbit	0	2	11

Table 2: A simple confusion matrix

c) Recall and Precision Rates

Consider a binary classification problem, that is, containing only positive or negative examples, and a classifier that classifies these examples into two possible classes. A positive example assigned to the positive/negative class is called *True Positive/False Negative*. In the same way the *True Negative/False Positive* terms are defined. Based on this classification, the most obvious way to measure the performance of a classifier is to calculate the correct classification rate, defined as the sum of True Positives and True Negatives, divided by the total number of the examples.
There are cases, however, where the classification rate can be misleading. Consider the two following cases, depicted in the following tables:

Classifier	TP	TN	FP	FN	Recognition Rate
A	25	25	25	25	50%
B	37	37	13	13	74%

Classifier	TP	TN	FP	FN	Recognition Rate
A	25	75	75	25	50%
B	0	150	0	50	75%

Table 3: Two different classification scenarios. The top matrix depicts a classification problem using a balanced dataset (same number of positive and negative examples), while in the bottom matrix, the dataset is unbalanced.

It is clear, from the first table, that classifier B is better than A, since its classification rate is significantly larger and the number of positive and negative examples is the same. For the second table however, it is clear that, while classifier B correctly classifies the negative examples, it misses all the positive ones, in contrast to classifier A, whose classification performance is more balanced between the two classes. In order to overcome this ambiguity, and to be able to compare the two classifiers, the recall and precision rates are used instead.

Recall and Precision measure the quality of an information retrieval process, e.g., a classification process. Recall describes the completeness of the retrieval. It is defined as the portion of the positive examples retrieved by the process versus the total number of existing positive examples (including the ones not retrieved by the process). Precision describes the actual accuracy of the retrieval, and is defined as the portion of the positive examples that exist in the total number of examples retrieved. A schematic representation of the recall and precision rates is given in Fig.5. *A* represents the set of True Positives, *B* is the set of positive examples not retrieved, that is, False negatives and *C* is the set of negative examples that were wrongly classified as positives, that is, False Positives. Based on the recall and precision rates, we can justify if a classifier is better than another, i.e. if its recall and precision rates are significantly better.

Fig.5: Schematic representation of the recall and precision rates.

d) F_α measure

While recall and precision rates can be individually used to determine the quality of a classifier, it is often more convenient to have a single measure to do the same assessment. The F_α measure combines the recall and precision rates in a single equation :

$$F_\alpha = (1+\alpha)\frac{precision * recall}{\alpha * precision + recall},$$

where α defines how recall and precision will be weighted. In case recall and precision are evenly weighted then the F_1 measure is defined as follows:

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

You are highly encouraged to write their own script to compute the Confusion Matrix, from which they would be able to compute the recall and precision rates, as well as the F1 score. You will be asked during lab interviews to produce a confusion matrix, as well as the F1 score and precision and recall rates, for a given test data. Failing to produce the Confusion Matrix and the scores will incur in a noticeable drop in the group's mark for that assessment.

e) Binary vs. Multi-class Classification

So far we have only talked about classifiers using datasets with only positive or negative examples. These classifiers are called binary classifiers, since the class labels can only take two values: ±1 (positive/negative). Many real-world problems, however, have more than two classes. In these cases, we are talking about multi-class classification. To get M-class classifiers, it is common to construct a set of binary classifiers $f^1 \ldots f^M$, each trained to separate one class from the rest, and combine them for doing the multi-class classification according to the maximal output, that is, by taking:

$$\arg\max_{j=1\ldots M} g^j(x),$$

where $g^j(x) = \text{sgn}(f^j(x))$, and sgn() is the sign function. In case of a tie, i.e. two or more classifiers output +1, then the class is randomly chosen among the classes which were proposed by the classifiers. This is called one versus the rest classification, where

each separate classifier is trained on one particular class, the positive class, while all the examples that belong to the rest of the classes are pooled together in order to form the negative class. One possible drawback of this method is that every classifier is trained on a small set of positive examples and a large set of negative ones, leading to an unbalanced training set, as described in section (c).

Pairwise classification trains a classifier for each possible pair of classes. For M classes, this results in (M-1)M/2 binary classifiers. This number is usually larger than the number of one-versus-the-rest classifiers; for instance, if M=10, we need to train 45 binary classifiers rather than 10 as in the method above. When we try to classify a test pattern, we evaluate all 45 binary classifiers, and classify according to which of the classes gets the highest number of votes. A vote for a given class is defined as a classifier putting the pattern into that class. The individual classifiers, however, are usually smaller in size than they would be in the one-versus-the-rest approach. This is for two reasons: first, the training sets are smaller, and second, the problems to be learned are usually easier, since the classes have less overlap.