COMP9417 18s1 Assignment 2 - project topics

0: Self-proposed

The objective of this topic is to propose a machine learning problem, source the dataset(s) and implement a method to solve it. This will typically come from an area of work or research of which you have some previous experience.

Topic 0: Propose your own topic

Task description
Up to you. The basic conditions are:

- it must involve some practical work with some implementation of machine learning
- you must send an email to one of the course coordinators with a description of what you are planning (a couple of paragraphs would usually be enough) that needs to be approved in an emailed reply before you start
- it must not involve double-dipping, i.e., be part of project for another course, or for research postgrads it must include a statement to the effect that it is not part of the main work planned for the thesis (although it can be related, e.g. if your research is on behavioural cloning, it could be on reinforcement learning, which is a related but different approach)

Team 1 to 4 person team

Difficulty assigned per topic (often 5/5).

1: Competitions & Challenges

The objective of these topics is to solve a machine learning competition or challenge where the dataset(s) are provided and you must implement a method to solve problem.

Topic 1.1: Machine learning and data mining competitions

Task description
A number of sites now host regular competitions. You can enter a live competition or work on the dataset from a past competition. The main site hosting machine learning competitions is Kaggle. Current competitions are here.

Note: only competitions "for prizes and point" are acceptable. However, you can select one from either Active or Completed competitions to work on. There are now 8

years worth of completed competitions.

Many research-oriented competitions are also hosted on Kaggle. However, there are some long-running competitions that pre-date Kaggle. For example, the ACM KDD Cup competitions have been running annually since 1997. These are challenging machine learning tasks, and are chosen to be representative of problems faced in industry and academia. However, note that in these competitions it is often the case that, as in many real-world data mining applications, it is usually as important to get to know the data, deal with issues in setting up the problem, etc. as it is to work out how to actually do the learning. A full list of KDD Cup competitions is available in the KDD Cup Archives.

For any of the competitions, active or completed, the choice of competition is up to you, but you should select only **one** of the competitions and work on that. The basic conditions are:

- assess carefully the time you will need to understand the competition requirements, get familiar with the data and run the algorithm(s) you plan to use
- you must send an email to one of the course admins with a description of what you are planning (a couple of paragraphs would usually be enough) that needs to be approved before you start
- your report and software must be submitted as for the other topics; for live competitions you can include your submission's placing on the leaderboard at submission time!

You must follow the task description as closely as possible. In the competitions that have closed, the test data are available, but you should avoid testing against these until you are really sure you understand the performance of your method(s). Test set performance should be evaluated and appear in your report, as well as an indication of how your team's performance matches those in the original competition (if available, but there usually is a leaderboard or published papers by the top-placed teams available after the competition closes).

Team 1 to 4 person team

Difficulty depends on the topic, but often 4/5 or 5/5.

Topic 1.2: Coda Lab datasets

Task description
Coda Lab hosts datasets, code and papers on two areas that may be of interest: Question Answering and Image Classification. Question Answering is a task in the area of Natural Language Processing that requires mapping a natural language question along with some knowledge source (text or a structured database) into an answer. A number of Question Answering datasets are here. This includes a live competition

using the Stanford Question Answering Dataset (SQuAD), which can be accessed here. The Coda Lab worksheet on Image Classification contains some popular image classification datasets, such as ImageNet.

Team 1 to 4 person team

Difficulty 5/5.

Topic 1.3: Deep Reinforcement Learning / Transfer Learning

Task description
This task is pretty much state-of-the-art in reinforcement learning. *It is recommended that you do not attempt this unless you have experience in both deep learning and reinforcement learning*. This is a current contest (finishes June 5, 2018) using the Sonic The Hedgehog series of games for SEGA Genesis. In this contest, participants try to create the best agent for playing custom levels of the Sonic games - without having access to those levels during development.

Team 1 to 4 person team

Difficulty 5/5.

Topic 1.4: Learning for Artificial General Intelligence

Task description
Microsoft researchers have made available environments for agents based on Minecraft called Project Malmo. Can you select a task for agents in this environment and define a machine learning problem to enable the agents to improve with experience on this task ?

Team 1 to 4 person team

Difficulty 5/5.

Topic 1.5: Learning to predict "stance" of news articles

Task description
The problem of automatically detecting "fake news", that is, news stories that are made up with an intention to deceive, typically for the purpose of some secondary gain, often financial, is known to be difficult. Another way of thinking about this problem is that it is an automated version of *fact checking*, i.e., establishing the truth or otherwise of some assertion. Not surprisingly, in general this is a hard problem problem for AI.

A contest last year had the aim of using machine learning to tackle an early stage of this problem, that of predicting the "stance" of the text of a news article relative to a headline. In this setting the text may be said to agree, disagree, discuss or be unrelated to the headline.

Although the contest is now over, the publicly available datasets can be used as a text classification problem. You should follow the competition methodology and scoring system (a Python script is provided for this purpose) as fara as possible. Note: the supplied *test* datasets are **NOT** to be used for tuning or tweaking your algorithms, but only for the final evaluation of whatever methods you use!

Team 1 to 4 person team

Difficulty 5/5.

Topic 1.6: Data repositories

Task description
As an alternative to following the rules of past competitions you can choose to apply machine learning to the datasets in new and interesting ways, or you could do the same with datasets from some of the many repositories now on the Web. The original such repository is the UC Irvine Machine Learning Repository here. These datasets are usually formatted for easy import into machine learning software.

MIT's Reality Commons project has some datasets on human behaviour obtained from mobile phones and other devices, such as the one on Human Activity Recognition Using Smartphones here.

The U.S. Government's open data initiative Data.gov contains many data sets. The problem is that many are simply raw data and individual data sets do not represent a well-defined learning problem; however, the hope is that by linking several data sets some interesting mining may be possible. Other governments doing similar things are in the U.K., N.Z and Australia. A more complete list is available here.

Many of the above, plus more, are included in the list of repositories at KDnuggets.

Pick a data set, review its publication history (if any) and see if you can think of an interesting angle to explore to make a project. The basic conditions are

- assess carefully the time you will need to specify a well-posed problem and set up the dataset(s) and algorithm(s)
- give particular attention to defining a target function and ways in which you can evaluate the learning
- you must send the course admins an email with a description of what you are planning (a couple of paragraphs would usually be enough) that needs to be

epproved before you start

Team 1 to 4 person team

Difficulty assigned per topic (often 5/5).

2: Core methods

The objective of these topics is to give you an opportunity to get a deeper understanding of some of the ideas we have covered in the course. Note: these are "***implement from scratch***" projects, that is, you must not use any machine learning libraries, apart from utilities to read data from files. This means that, for example, if you are implementing in Python, you cannot use any code from scikit-learn (except for reading data from a file), although you can use NumPy or SciPy code to implement your algorithm.

Topic 2.1: Perceptron / Linear Unit

Task description
Write a program to implement learning for numeric prediction / classification based on an *n*-input linear unit / perceptron architecture. First, implement the linear unit architecture and the gradient descent learning method. Test your system on the standard UCI data set autoMpg. Note that you will have to think about the representation of input data for this task: some of the attributes are real-valued, but others have discrete, ordered values. One possibility is to use a Boolean variable (an "indicator" variable) for each discrete value for such attributes, where the value is 1 if the instance has that value for the attribute, and 0 otherwise.

Second, implement the perceptron training algorithm to learn *n*-input perceptron classifiers for Boolean functions. Test on examples from at least two different **8-input** Boolean functions (note: should include linearly-separable and non-linearly-separable functions).

Third, implement the "Pocket Algorithm" which is a straightforward extension adding a kind of "weight memory" to the perceptron training algorithm which can improve learning performance. Pseudo-code for the algorithm is here. Compare its learning performance with that of the perceptron training algorithm (for example, by plotting the values of the pocket's accuracy variables compared to those of the perceptron's during learning on training data with different levels of added noise).

There is an option in this project to add graphical output showing how the system is learning. This may require adding an extra person to the team. The graphics do not have to be very complicated, but should show key features of the learning algorithms, such as weight update, or changes to the variables in the pocket algorithm.

Team 1 to 3 person team

Difficulty 3/5

Topic 2.2: Nearest Neighbour

Task description
Implement *k*-Nearest Neighbour (kNN) for both classification and numeric prediction.

For classification, test your version of kNN for range of values of k on the standard UCI data set ionosphere. Evaluate your system by leave-one-out cross-validation.

For numeric prediction, test your version of kNN for range of values of k on the standard UCI data set autos. For this data set you can remove examples with missing attribute values, which will reduce the number of examples to 159. The task is to predict the price given the 14 continuous and 1 integer attributes. You can also experiment with encoding the other attributes to allow their use in the distance function to see if this affects predictive accuracy. Evaluate your system by leave-one-out cross-validation.

If time permits you can extend your methods to implement distance-weighted Nearest Neighbour (WNN).

There is an option in this project to add graphical output showing what the system "learns". This requires adding an extra person to the team. The graphics does not have to be very complicated, but needs to show key features of the algorithms, such as the effect of varying the *k* parameter.

Team 1 to 3 person team

Difficulty 3/5

3: Legacy methods

The objective of these topics is to broaden coverage by including options that are no longer included in the course material, but may be of interest for further study. For each topic, versions of past lecture materials are provided as background.

Topic 3.1: Genetic Algorithm

Task description
Implement the basic Genetic Algorithm method as described here, including the point

mutation rule and at least two of these crossover rules - single-point, two-point and uniform. Test on the standard UCI data sets balance-scale and mushroom.

You will need to consider carefully the representations you use for the classifiers to be learned for both data sets. Also note that the mushroom data set is quite large.

There is an option in this project to add graphical output showing how the system is learning. This requires adding an extra person to the team (from 2 to 3 people). The graphics does not have to be very complicated, but needs to show key features of the algorithm, for example, the selection process to construct a new generation of hypotheses.

Team 2 or 3 person team

Difficulty 3/5

Topic 3.2: Rule Learning by Sequential Covering

Task description
Implement a Sequential Covering rule learning algorithm based on the method described on slides 21-25 of these notes. Your program should use a version of *information gain* from decision tree learning to implement the "Performance()" measure to select the attribute-value test to specialize the rule's condition part.

Test on these versions of the UCI data sets mushroom and splice.

Note that you do not need the instance name (first attribute) in the splice data set, and that the mushroom data set is quite large !

There is an option in this project to add graphical output showing how the system is learning. This requires adding an extra person to the team (from 1 to 2 people). The graphics does not have to be very complicated, but needs to show key features of the algorithm, for example, the coverage and accuracy of the process of learning a rule.

Team 1 or 2 person team

Difficulty 3/5

Topic 3.3: Inductive Logic Programming

Task description
Implement an algorithm to compute the Relative Least General Generalization (RLGG) of pairs of ground instances of the target predicate, as outlined on slides 27-36 in these lecture notes.

Data set: In a typical Bongard problem there are 6 positive and 6 negative examples of the hidden rule that correctly classifies a scene. Select a small number (at least 2) of the Bongard problems from here (you could also make your own, or contact me if you need more information).

You will need to represent the scenes in such a way that the RLGG can be represented as a single clause. The RLGG is constructed from a pair of positive examples, so in a typical Bongard problem there would be 15 possible pairs to choose to initialize RLGG construction.

Once constructed, the RLGG clause (which is unique for a pair of positive examples and fixed set of background clauses) can be further generalized to ensure it covers all the positive examples and none of the negative examples by dropping literals or turning constants to variables.

Represent the examples (pos and neg) and the background knowledge in Prolog. It may be easiest to implement the algorithm in Prolog too. There are several high-quality open-source Prologs. SWI Prolog has the best documentation, with built-in help, whereas in general YAP Prolog enables faster running compiled programs.

Team 1 or 2 person team

Difficulty 4/5

Topic 3.4: Recommender system using collaborative filtering

Task description
Pick **one** of the the following problems for recommendation and apply collaborative filtering as described in these lecture notes (or an alternative). The data is from a collection collected by the GroupLens research group.

The suggested problems are either: learning to predict the ratings of books on the BookCrossing dataset, or movie ratings on the MovieLens Data Sets (these are of different sizes, so start with the 100K). However, other options are also available from the GroupLens site.

Team 1 or 4 person team

Difficulty 5/5

Topic 3.5: Face recognition using neural networks

Task description
This is Mitchell's face recognition task using neural networks. The task is explained here. See also Chapter 4 in Mitchell's book "Machine Learning" which is in the library.

We showed an example of this neural network in lectures, and this will allow you to get experience of neural networks without the use of modern software libraries which hide a lot of the details.

Note that although the task description is clearly dated, the problem remains an interesting one. You need to complete the second part of the task as well (ignore the ``optional'' heading !). This is an open-ended assignment, so you will need to consider how far you can take this this.

Team 2 or 3 person team

Difficulty 3/5

Topic 3.6: Reinforcement Learning

Task description
Implement a traffic light simulator for a single intersection and a reinforcement learning system to learn how to switch the traffic lights to reduce the delay for vehicles at the intersection. You can use Q-learning, as described in these notes, or an alternative.

The details are explained here and there is a short movie showing the traffic light controller in operation here.

Implementing graphics will help considerably in debugging this project.

Team 3 or 4 person team

Difficulty 4/5 to 5/5 depending on extensions.

4: Miscellaneous

These options do not fit anywhere else, but may be of interest.

Topic 4.1: Machine Learning for Bioinformatics

Task description
Since these applications depend on domain knowledge, a team should include at least one student majoring BE Bioinformatics, and you will need to contact LiC if you are interested.

Team 1 to 4 person team

Difficulty assigned per topic (probably 5/5).

Topic 4.2: Learning to play "20 Questions"

Task description
The traditional game often called "20 Questions" has been implemented in at least two online versions that learn from user interaction, 20Q and Akinator. It is not clear how the learning is implemented for either of these systems, but there appears to be a patent application on the technology behind 20Q by its author, Robin Burgener, which outlines a form of neural network. In this project, which is very open-ended, the objective is to implement a simple version of the twenty questions game that can interact with users and learn which questions to ask, thereby (hopefully) improving with experience. Some general background and information about radio and TV versions is available on Wikipedia. There are various pointers on the web to different implementations, for example, in Python, or suggestions about the way in which this kind of system could use machine learning, but these should be treated with caution. This project is only for those with experience in the rapid development of interactive systems who are looking for a challenging application of machine learning in this context.

Team 1 to 4 person team

Difficulty 5/5

Report and assessment criteria

Assignment criteria.

Brief guide to writing the report.

Sample of a report.

Submission

Submission is via `give`. You need to submit the project files and your report.

The project files should be submitted as a compressed `tar` or `zip` archive. If your submission exceeds the current limit (2MB) contact the course admins to arrange an alternative method of submission for the excess materials, such as a download link.

The report must be in PDF format.

Combinations accepted are (if you are running from the Linux command-line):

```
$ give cs9417 ass2 files.tgz report.pdf
```

```
$ give cs9417 ass2 files.zip report.pdf
```

Deadline

Sunday June 3 23:59:59 (contact course admins if you need an extension).

Last modified Mon May 7 09:12:45 AEST 2018