COMP90051 Statistical Machine Learning
Project 2 Description

Due date: 4:00pm Thursday, 27^{th} May 2021 **Weight:** 25%[1]

Most machine learning models are based on supervised learning, applied to a given training dataset which includes instances and their labels. In many practical settings, however, labelled data is not readily available. While unsupervised learning is one way of dealing with this situation, a popular alternative that can still make use of powerful discriminative machine learning methods is *Active Learning*. This describes a machine learning framework whereby data is labelled in an incremental manner, and a machine learning model is used to decide what data instances should be annotated next. This allows the model to focus the labelling effort—which can often be slow and expensive, e.g., requiring manual human labour—on highly informative instances which can correct weaknesses in the model. There are several different strategies for choosing (or synthesising) data for annotation, and this project will require you to understand and implement several such techniques.

In this project, you will work *individually* (not in teams) to implement several active learning methods. For all of these methods you will have to refer to the literature, and read and understand it yourself.

By the end of the project you should have developed

ILO1. An understanding Active Learning, and how it relates to other learning paradigms in machine learning;

ILO2. Better understanding of how machine learning concepts like uncertainty can support machine learning;

ILO3. Demonstrable ability to implement ML approaches in code; and

ILO4. An ability to pick up recent machine learning publications in the literature, understand their focus, contributions, and algorithms enough to be able to implement and apply them. (And being able to ignore other presented details not needed for your task.)

Overview

We will be working largely from the following book which presents a detailed survey of the field of active learning:

> Burr Settles, 'Active Learning', *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan Claypool, June 2012.
> `http://cat.lib.unimelb.edu.au/record=b5410513~S42` (downloadable ebook)

in particular chapters 2, 3 and 5. While the above book does a superb job of covering many of the algorithms and theory, you will also want to read other papers, such as the original papers presenting the algorithms, as well as papers with a stronger empirical focus like (Schein and Ungar, 2007) as cited in the above book.

Your task is to develop pool-based active learning methods to implement an active learning framework, and using this to support uncertainty sampling, query by committee and hierarchical cluster-based sampling. In all cases logistic regression is to be used as the component classification model.

You will be using the dataset to simulate active learning of a classifier which detects the language given a handwritten character image. We will use part of the OmniGlot dataset[2] which comprises the alphabets of 30 different languages, with a total of 19,280 images of hand-drawn characters. This will be used for simulating active learning by starting from a small *seed set* of 300 labelled images,[3] then progressively revealing more labels over the course of an active learning run, retraining a classifier at each stage. The testing set will be used for

[1] Forming a combined hurdle with project 1.

[2] `https://github.com/brendenlake/omniglot`

[3] Stratified such that every label is equally represented.

reporting of results. Unlike the previous project, we have provided you with all the data, so you will need to be careful you use these resources in an appropriate manner.

Required Resources The LMS page for project 2 comprises

- `project2.pdf` this spec;
- `project2.ipynb` Jupyter notebook skeleton in Python;
- `project2_dataset.ipynb` Jupyter notebook illustrating the dataset and its processing (purely FYI);
- `images.npy` numpy file of vector embeddings of the dataset instances; and
- `labels.npy` numpy file with a vector of labels, one per instance.

You will implement code in Python Jupyter notebooks, which after running on your machine you will submit via LMS. Further detailed rules about what is expected with code are available towards the end of this spec. We appreciate that while some have entered COMP90051 with little prior Python experience, many workshops so far have exercised and built up basic Python and Jupyter knowledge.

Dataset: The LMS page for project 2 contains a dataset suitable for validating your AL algorithms. You should download these files and familiarise yourself with their content:

- `images.npy`: a matrix of 19,280 x 1,000 encoding images of handwritten characters, each embedded in 1000d
- `labels.npy`: a vector of 19,280 string valued labels, encoding to the alphabet name of the corresponding image

The notebook includes code for processing these files into a training pool, seed set, and testing set. Note that the seed set lists the indices of examples from the training pool that are to be treated as labelled, while all other instances are to be initially considered unlabelled. Over the course of an active learning simulation additional instances will have their labels revealed, and become part of the labelled set.

Part 1: Implementing Logistic Regression component classifier [4 marks]

Implement Python functions `train_logistic_regression` and `evaluate_logistic_regression_accuracy` which train a logistic regression classifier, and evaluate it against a test set. You are welcome to use python libraries, e.g., those you've encountered in the workshops, for this task. Use these two functions to train logistic regression over the *seed set*, and train on the full pool. Report the accuracy on the test set. This will serve as a rough guide for the range of performance you might expect to see the active learning models, below.

Part 2: Active Learning algorithm with Random heuristic [6 marks]

For the next stage, you will need to develop the meta-algorithm for pool based active learning. The pseudocode for the algorithm is presented in Figure 1, which is designed to support all manner of active learning algorithms, including those in the subsequent parts of this project. You may wish to compare this to Settles Figure 2.3, which is closely related. Here we consider the faster setting of batched active learning, rather than the online version where $b = 1$ instance is selected in every round, and we also treat selection as a ranking step followed by an argmax, which lends itself to simpler and more modular implementation.

Your task is to implement the pool based active learning algorithm, following the `pool_based_active_learning` function supplied in the notebook, and the random selection heuristic in the method `random_select` (such that

1: $\mathcal{U}$ = pool of unlabelled instances, $\{\mathbf{x}\}$
2: $\mathcal{L}$ = set of initial labelled instances, $\{\langle \mathbf{x}, y \rangle\}$
3: b = number of instances to label in each step
4: **for** $t = 1, 2, \ldots, T$ **do**
5: $\quad \theta^{(t)} = \textbf{train}(\mathcal{L})$
6: $\quad$ score all instances in pool, $\mathbf{r} = \textbf{select}(\mathcal{U})$
7: $\quad$ **for all** $j \in \text{argmax}(b, \mathbf{r})$ **do**
8: $\quad\quad$ reveal label y_j
9: $\quad\quad$ add $\langle \mathbf{x}_j, y_j \rangle$ to $\mathcal{L}$
10: $\quad\quad$ remove $\mathbf{x}_j$ from $\mathcal{U}$
11: $\quad$ **end for**
12: **end for**
13: **return** $\{\theta^{(t)}\}_{t=1}^{T}$

Figure 1: Generic active learning algorithm, adapted from Figure 2.3 of Settles (2012). Requires a train function and select function, and $\text{argmax}(b, \mathbf{r})$ returns the indices of the b maximising elements of $\mathbf{r}$ (tied values to be broken at random).

lines 6-7 end up sampling uniformly without replacement from the pool). Apply this active learning function along with the random selection function and the training function from Part 1. You should use run active learning such that it starts with the supplied *seed set*, and then trains on progressively larger datasets growing by $b = 60$ instances in each step, and stopping after reaching 3000 instances, i.e., $T = 45$ steps. Present the results of your experiment in a plot showing test accuracy compared to number of training instances.

Part 3: Uncertainty sampling [4 marks]

Now that you have a working infrastructure, it is time to develop the first active learning algorithm. For this, you will use uncertainty sampling, one of the simplest and most enduring methods for active learning. This works by selecting instances that have the most predictive uncertainty under the model, and accordingly one would expect the model to learn the most from their labelling.

You will need to review Settles Ch 2, particularly "Entropy" on pp 14–15. You may also find this reference useful:

> A.I. Schein and L.H.Ungar. 'Active learning for logistic regression: An evaluation'. *Machine Learning*, 68(3):235265, 2007.
> `https://repository.upenn.edu/cgi/viewcontent.cgi?article=1378&context=cis_papers`

Your task is to implement the entropy selection heuristic in the method `logistic_regression_entropy_select`. Evaluate your selection function in the active learning framework and present a plot of the accuracy over the active learning run, based on the same run parameters from Part 2.

Part 4: Query by committee (QbC) [6 marks]

The next active learning algorithm is based on *training an ensemble of several models* and measuring the extent of disagreement within the ensemble as the selection heuristic. The idea is that instances that are predicted differently by each ensemble member are likely to be highly informative, once labelled. You will need to review Settles chapter 3, specifically section 3.4 on 'Query by Committee', and the several cited papers on pages 28-29. Your task is to implement the QbC method. You should set the size of the ensemble to two, and use bagging as the ensembling method.

You should implement three techniques for measuring disagreement: vote entropy, soft vote entropy and the KL-based method (Settles eq 3.1-3.3, page 29), implementing each of these as a separate selection function in methods `query_by_committee_xxx` where `xxx` is replaced with the name of each of the three methods. You will also need to complete the specialised training function, `train_committee` to learn an bagged ensemble of classifiers in each step.

Evaluate the results of these three methods using your active learning framework. Present a plot with a curve for each of these three methods, using the same run parameters as for Parts 2 & 3. Ensure that you report the accuracy of training a standard logistic regression classifier on the selected datasets from each iteration of active learning, not the accuracy of a bagged ensemble, which may be different.

Part 5: Hierarchical sampling [5 marks]

The last algorithm required in the project, is a hierarchical sampling method presented in the following paper:

> S.Dasgupta and D.J.Hsu. 'Hierarchical sampling for active learning'. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 208215. ACM, 2008.
> Paper: `https://icml.cc/Conferences/2008/papers/324.pdf`
> **Updated 21/5** Talk: `http://videolectures.net/icml08_hsu_hsa`

See also the detailed coverage of this algorithm in Settle section 5.2. This method uses an initial hierarchical clustering of the data, which it then prunes based on empirical measurements of the purity of each cluster, i.e., how concentrated points in the cluster are with respect to the class label. You will need to make several decisions, such as how to cluster your data, and which selection mechanism to use (you shouldn't use the random method, but the other methods presented are all good choices; note the connection between the UB method and UCB1 from the MAB lecture.) Justify any decisions you make in your submission, using short inline comments.

Note that this algorithm doesnt require repeated retraining in the active learning loop, it only requires storing of counts, and thus it makes sense to process one instance at a time. You may wish to implement this as a completely separate function, rather than using the `pool_based_active_learning` method. Report the performance of hierarchical sampling through presenting a plot as for the previous methods, showing the test accuracy versus training data size, based on an active learning run with the same parameters as before. **Updated 21/5**: Note that the algorithm in Dasgupta & Hsu (2008) includes a 'batch' parameter B, for which you are free to try various values, but $B = 60$ is a sensible choice.

Project Submission

Preserving its structure, you must (1) rename `project2.ipynb` as `username.ipynb` using your username;[4] (2) flesh out with your project solutions with cells, (3) run on your local machine prior to submission so that outputs and plots are preserved (you are strongly recommended to open your notebook again prior to upload to double check. We may not run your notebook; given your environment might subtly differ to ours, it is your responsibility to ensure results are contained), and then (4) submit in LMS.

Marks: graders will perform code reviews of your implementations. In general a portion will be available for correctness, a portion will be available for code structure and style (primarily the former). Code should have necessary commenting to understand interfaces, and major points of inner working, basic checks of well-formed input, clear variable names and readable statements.

[4] LMS/UniMelb usernames look like `tcohn`, not to be confused with email such as `trevor.cohn`. Please also put your name and student id in the top text cell of the notebook.

Further Rules. You may discuss Python at a high-level with others, but do not collaborate on solutions. You may consult resources to understand active learners conceptually, but do not make any use of online code *whatsoever*. (We will run code comparisons against online partial implementations to enforce these rules.)

You must use the standard python3 environment as used in the labs (Anaconda3, Python 3.6 or higher) You may use the packages already imported in the provided `project2.ipynb` notebook, but are welcome to add additional imports to standard python libraries and those for scientific computing (this does not include python libraries that are explicitly designed for active learning). You should use `matplotlib` for plotting. Late submissions will be accepted up to 4 days with -3 penalty per day.