Machine Learning H/M Coursework 2018

Anders Uhrenholt[#], Grimur Hjorleifsson Eldjarn[#], Ting Su[#], Ubong Etuk[#], Ke Yuan
{a.uhrenholt.1, 2332462H, 2279173S, u.etuk.1, ke.yuan}@glasgow.ac.uk
[#] lead authors
07-02-2018

Introduction

1. This coursework is a group project.
    1.1. You are asked to form 3- or 4-member groups.
    1.2. Each group should form a team on Kaggle and submit their answers using the team name thereafter.
    1.3. The final report and code should be submitted in group unit.
    1.4. Each person should also submit a delta score doc to evaluate each member in the group.

2. The coursework consists of two tasks: 1) regression, 2) classification.
    2.1. You are asked to construct 2 algorithms for each task. That is 4 algorithms in total.
    2.2. You are encouraged to write your own code, but use of 3rd party implementations is allowed, as long as you demonstrate an understanding of the algorithms in the accompanying report.
    2.3. You should test your algorithm using Kaggle page provided for each task.

3. Instructions for working with Kaggle (Group)
    3.1. For each task, there is a Kaggle page where you can download your training set and testing set.
    3.2. You should submit your answers to each Kaggle page following the description.
    3.3. You should also use the feedback from Kaggle to refine your algorithms.
    3.4. You are allowed to submit twice a day for each task. Use your submissions wisely.
    3.5. By the end of the competition on Kaggle, you are asked to submit **only your best result for each task**, no matter which algorithm you are using.

4. Moodle submission
    4.1. Every student is asked to submit three documents through moodle: A report (PDF), a zip folder containing the 4 algorithm implementations, and one delta score report.
    4.2. Report: This is an overall report, where you should discuss both of the tasks according to the task descriptions below. The purpose for the report is to demonstrate why a particular algorithm is selected as the final Kaggle submission.
    4.3. Code submission
        4.3.1. The 4 algorithm implementations should be in separate files, using names like:
            - "CM-your_algorithm_name.format"
            - "RM-your_algorithm_name.format"
        4.3.2. Submit one zip folder containing all these 4 implementations.
    4.4. Delta score **(Grade distribution)**

4.4.1. You are asked to submit your own version of delta score doc to evaluate members in the group.
4.4.2. This should include scores you assign to each member, and the reason for your assignment.
4.4.3. These scores must be in the range [0-30] for each member, but the total score of the group should add up to 30.

For M student only:
If you are in a level M group, you are also asked to include a literature review of 3 or 4 papers from the list below in your report. The requirement can be found in the making scheme.
- Blei, D., Ng, A. j., Jordan, M. I. (2003) Latent Dirichlet Allocation. Journal of Machine Learning Research. 3 (4–5): pp. 993–1022
- Kingma, D. P., Welling, M. (2013) Auto-Encoding Variational Bayes. arXiv:1312.6114
- Mikolov, T. et al (2013) Distributed Representations of Words and Phrases and their Compositionality. NIPS
- Goodfellow, I. J. et al (2014) Generative Adversarial Nets. arXiv:1406.2661
- Teh, Y. W. (2006). Hierarchical Dirichlet Processes. Journal of the American Statistical Association. 101: pp. 1566–1581.
- Buettner, F. et al (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature Biotechnology* v33, pp. 155–160.

Regression task
Over the years various methods have been proposed for evaluating CPU performance. One of the older ones is from the 1980's where the magazine Computerworld would compare new CPUs to a well-known base model (IBM 370/158 Model 3). The metrics included amongst others benchmark speeds, user experience, and expert reviews. Based on these factors a relative score was then calculated indicating to the consumer how much better or worse than the IBM model the CPU was expected to be.

Authors at the time examined whether the relative score for new models could be predicted directly from hardware specifications such as cache size and cycle time. Your job is to do the same. On Kaggle you will find a training set containing 168 CPUs with 6 features each: Cache memory size, minimum and maximum number of I/O channels, machine cycle time, and minimum and maximum main memory. From these you are tasked with predicting the performance scores for 41 unseen CPUs.
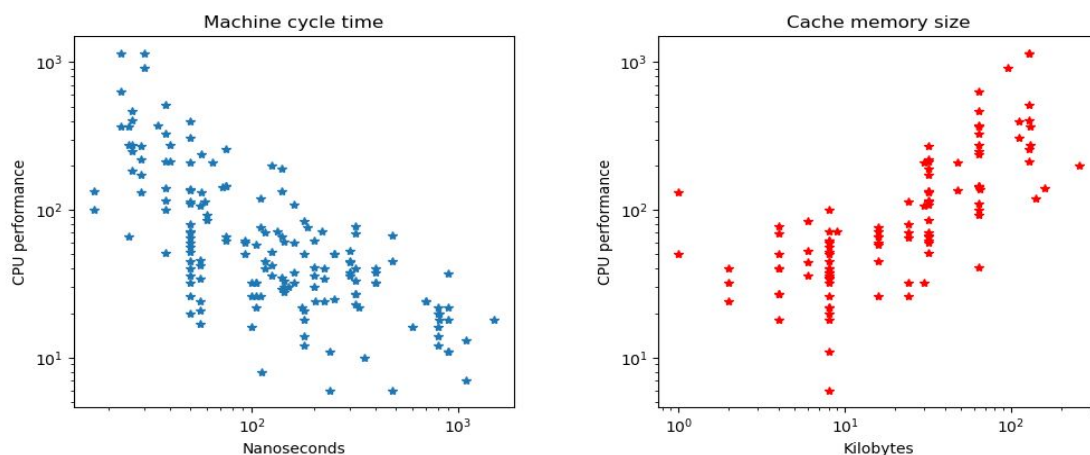

Figure 1: Two features in log-scale.

Your mission:

1. Construct **TWO** regression models for the CPUs, and test your models against the dataset on Kaggle. You're free to split data into training- and test however you like.

2. Use your data and the result you get using the dataset on Kaggle to answer the following questions in your report:

   • What are the models behind the two classification algorithms you chose and implemented?
   • What is your experiment setup for training these methods, and how are you going to measure the performance of your regression algorithms?
   • Compare the performance of these two algorithms (You can use the metrics you learned in the course, or any other measurements appropriate).
   • Can you obtain better performance by using only a subset of the features? Note: Be careful about using test data to train.

Note that it might be a good idea to transform one or more features into new features using basis functions. For instance, could the minimum and maximum memory features meaningfully be combined to one?

Classification task

Tissue microarray (TMA) is a recent innovation in the field of pathology. A TMA (Figure 1, left) contains many small representative tissue samples from hundreds of different cases assembled on a single histologic slide, and therefore allows high throughput analysis of multiple specimens at the same time. It is possible to build a classifier (Figure 1, right) that distinguish between epithelial (where cancer cells live) and stromal (where immune cells and other normal cells live) regions. The classifier identifies epithelial and stromal regions from images in large patient cohorts, allowing of quantification of the interaction between cancer cells and normal cells.
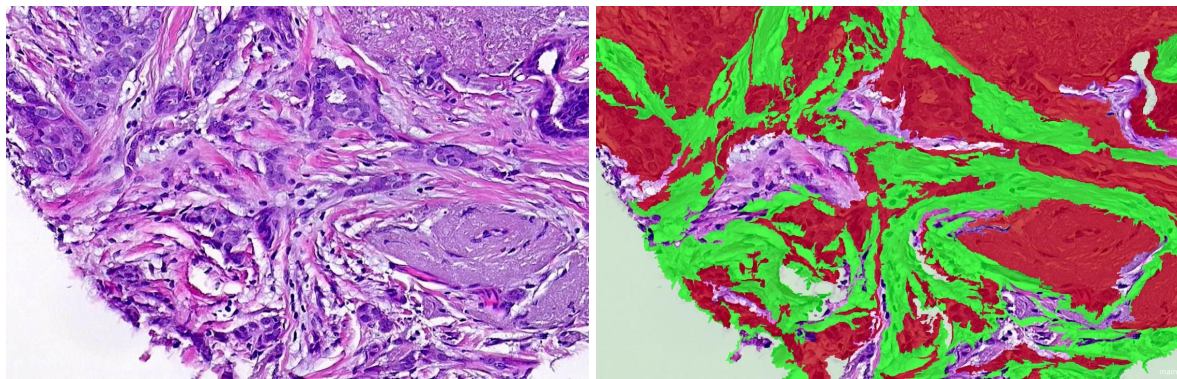

Figure 2: Left: a example of HE stained TMA from a breast tumour tissue. Right: The same image overlaid with identified epithelial (red) and stromal (green) regions. Images from Beck et al (2011).

The training file in Kaggle contains 600 data points and 112 features. Each data point is constructed from small regions of coherent appearance known as superpixel. The first column EpiOrStroma is the class for each object with 1 representing epithelial region (red in Figure 2, right) 2 representing stromal region (green in Figure 2, right). The remaining 112 columns are features extracted from TMAs using standard computer vision pipelines (e.g. image segmentation, edge detection, texture features, etc). Figure 3 shows the data in features GLCM.Ang..2nd.moment..quick.8.11..Layer.1..all.dir.. and Ratio.Layer.3.
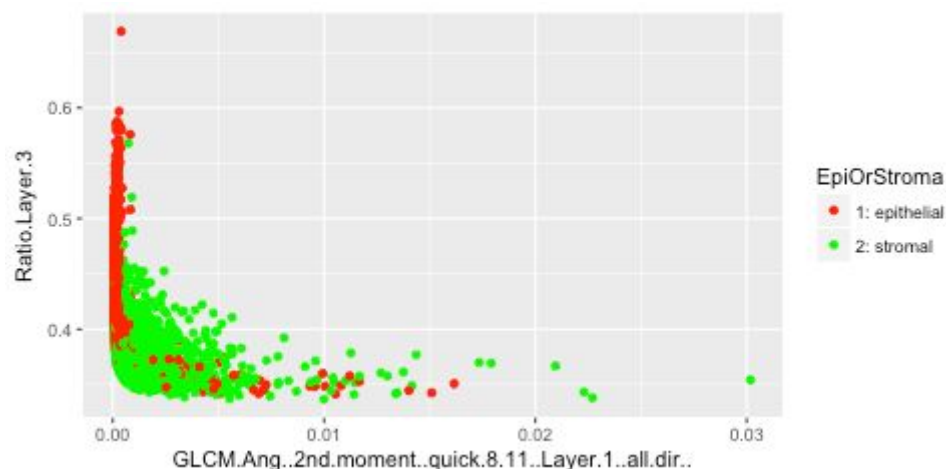
Figure 3: Scatter plot of features `GLCM.Ang..2nd.moment..quick.8.11..Layer.1..all.dir..` and `Ratio.Layer.3`.

Your mission:

1. Construct **TWO** classifiers for epi vs stroma superpixels task, and test your classifiers against the dataset on Kaggle. You're free to split data into train and test however you like.

2. Use your data and the result you get using the dataset on Kaggle to answer following questions in your report:

   • What are the models behind the two classification algorithms you chose and implemented?
   • What is your experiment setup for training these classifiers, and how are you going to measure the performance of your classification algorithms?
   • Compare the performance of these two algorithms (You can use the metrics you learned in the course, or any other measurements appropriate).
   • Can you obtain better performance by using only a subset of the features? Note: Be careful about using test data to train.

Hints:

   • Try to make good use of **numpy**, or other libraries that can help you with maths problems.
   • Third party libraries that have fully implemented classification algorithms are allowed. But you will **NOT** get the full implementation score if you use these libraries.
   • Think carefully when choosing your measurement metrics. Your choice needs to be explained in your report.

Marking scheme

● **10% (Kaggle and Moodle) Participation:** Submit to kaggle for both tasks. Submit report and code to moodle.
● **20% (Kaggle) Regression performance: average public and private score**
   ○ **⅓:** 3-member group: <150; 4-member group: <100
   ○ **⅔:** 3-member group: <55; 4-member group: <50
   ○ **Full:** 3-member group: <45; 4-member group: <42
● **20% (Kaggle) Classification performance: average public and private score**
   ○ **⅓:** 3-member group: >50%; 4-member group: >60%
   ○ **⅔:** 3-member group: >75%; 4-member group: >80%

- ○ **Full:** 3-member group: >85%; 4-member group: >88%
- **10% (Report) Regression methods benchmark and selection:** Quality and rationale of experiments.
- **10% (Report) Classification methods benchmark and selection:** Quality and rationale of experiments.
- **H: 10%, M: 5% (Report) Demonstrate good understanding of benchmarked regression methods:** General idea and mathematical details.
- **H: 10%, M: 5% (Report) Demonstrate good understanding of benchmarked classification methods:** General idea and mathematical details.
- **10% (Code) Implementation:** code clarity, documentation, experiment reproducibility.
- **M: 10% (Report) Literature review:** Should include 3 papers for 3 member group, 4 papers for 4 member group. What is the problem; what is the state-of-the-art before the paper; what is new about the solution; quality of supporting evidence (what experiment did the author conduct to demonstrate what).