MLM: Assessed Exercise Report

Tom Wallis

Introduction

Problem Outline

For the Machine Learning (Masters) coursework at Glasgow University, an assessed exercise was given, where students were to utilise machine learning concepts for the purpose of identifying areas of a epithelial and stromal regions in data provided pertaining to Tissue Microarrays.[1]

Some things about this data are worth noting:

- 112 features were provided. It is probable, however, that these may not *all* be useful in the classification of the stromal and epithelial regions. Therefore, identification of useful features in the data would dramatically reduce the time taken to classify large datasets.
- It might be that, for the given data,[2] certain classification algorithms may not lend themselves as well to performant, accurate classification as others. Therefore, it is important to identify which classification algorithms are most performant for the given dataset.

[1] The data provided included classifications for the datapoints; our assigned task was to create classifiers, from algorithms covered in the course, trained on this data.

[2] and for Tissue Microarray-related data in general

Research Question

Given these important questions regarding the dataset given, two research questions naturally arose, to be answered through the coursework:

1. Compare the performance of two classification algorithms of our choosing
2. Find out whether better performance can be obtained using a subset of the features

Research Approach

Two classification algorithms were tested:

- Gaussian Naive-Bayes[3]
- K Nearest Neighbours[4]

These classification algorithms were chosen for a few reasons:

Firstly, GNB predicts classes in a probabilistic fashion; KNN does not. Therefore, choosing to compare these two algorithms gives some insight (though clearly does not solve the matter completely) as to whether a probabilistic classifier might be more performant than a non-probabilistic classifier for this dataset.

Another reason for choosing these two particular algorithms is their relative simplicity. Support Vector Machines,[5] for example, have relatively complex classification mechanisms compared to a KNN classifier. The choice of especially simple probabilistic and non-probabilistic methods meant that particular sophistications from either focus would not give either an advantage in performance.[6]

[3] Referred to through this report as "GNB"

[4] Referred to through this report as "KNN"

[5] Referred to through this report as "SVM"

[6] One can imagine comparing a well-tuned, domain-specific classification algorithm against an unsophisticated method: one would naturally expect the more specialised algorithm to perform better, when applied to its specific domain. To avoid this and similar imbalances, a particularly simple approach was selected from both probabilistic and non-probabilistic offerings.

Model Details

The model was implemented in Python, using an object-oriented approach, and tested via a small script to interact with the model. The KNN classifier was implemented by hand; as a result of unforeseen time constraints, the GNB algorithm provided by the `sklearn` Python package was used.

KNN's classification system is to make use of geometric similarities between different datapoints with known classes. Should it be provided with a datapoint with an unknown class, it checks this datapoint's K closest geometric neighbours for their classes.

An assumption is made here in KNN's classification philosophy: a datapoint is likely to be of the class of other data which it is geometrically close to.[7] Therefore, KNN simply assigns the new datapoint a class determined by the most common class in its K nearest neighbours.

[7] While clearly having some issues, such as outlier detection, this approach does make sense if considering that classes of data will tend to cluster when limiting features to those which characterise the classes.

GNB's classification philosophy is also simple, though slightly more complex than KNN's. It finds its foundations in Bayes' Rule:

$$P(\theta|D) = P(\theta)\frac{P(D|\theta)}{P(D)}$$

However, any Naive Bayesian classifier makes an additional assumption, being that the components of a datapoint are independent for a given class:

$$p(x_{new}|t_{new} = k, \mathbf{X}, \mathbf{t}) = \prod_{d=1}^{D}(p(x_d^{new}|t_{new} = k, \mathbf{X}, \mathbf{t}))$$

...where D is the number of dimensions, and x_d^{new} is the value of the dth one.

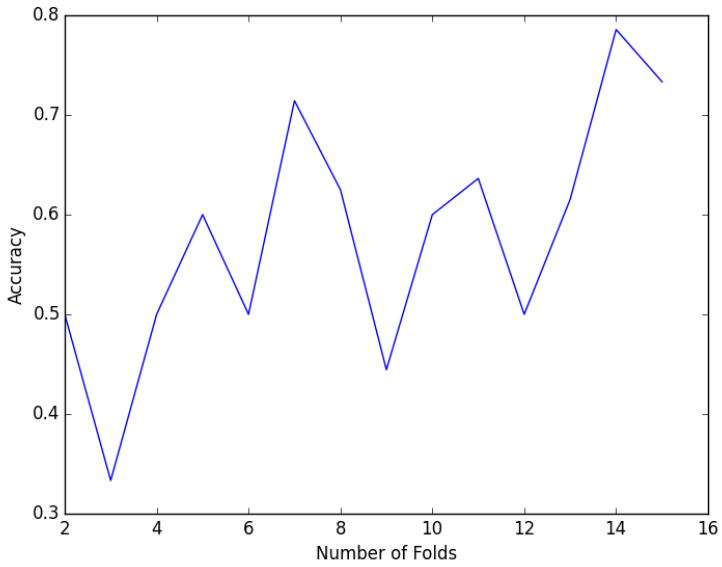
GNB then makes use of an assumption of Gaussian distribution around a class' mean geometric point. For each class, therefore, the chance of a new datapoint's being in a given class can be calculated by finding:

$$class = \underset{k \in 1,\ldots,K}{\operatorname{argmax}}\, p(x_{new}|t_{new} = k, \mathbf{X}, \mathbf{t}) \prod_{d=1}^{D}(N(\mu_{kd}, \sigma_{kd}^2))$$

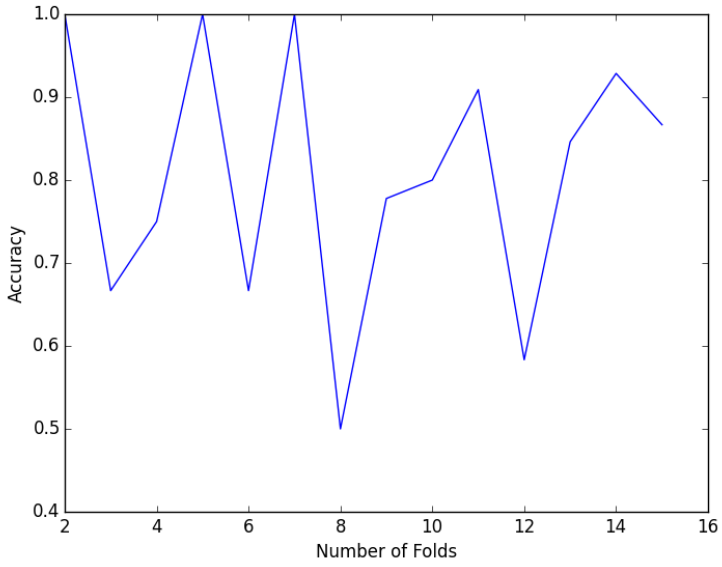
...for all classes k in K. Though the model's mathematical nature can make it seem complex, it ultimately relies on a small amount of conditional probability encasing reasoning about the distribution of the data in the model.

Findings

A factor used to optimise the performance of both models was the size of the folds in the cross-validation procedure: both models were tested for cross validation with between two and fifteen folds. Results varied considerably:



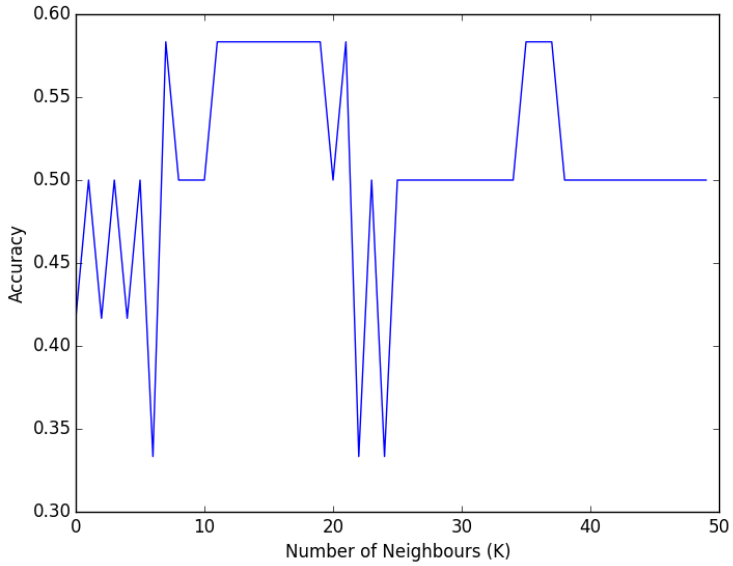
Values from cross-validation with different fold sizes for the KNN classifier.



Values from cross-validation with different fold sizes for the GNB Classifier.

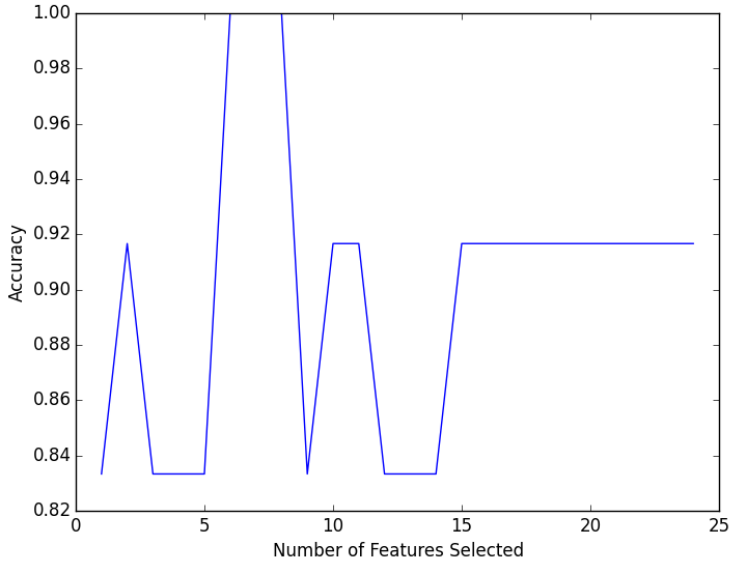
In an effort to maximise the efficacy of both classifiers, understanding their performance on different fold sizes was important; KNN was more performant when run through cross-validation with 12 folds — GNB proved to perform better going through cross-validation with 9 folds. More folds would ordinarily improve the performance of the classifiers, as they would have more data to classify using during the testing. To limit running times, the number of folds was capped at their most performant number ≤ 15 folds.

Another variable to be optimised was the number of neighbours KNN considered. Taking the previously determined number of folds, an optimal K value was found at 7. This was capped at 50 neighbours, as one can see that higher K values correlate to a more successful classifier (as is to be expected, as the classification can take more data into account):

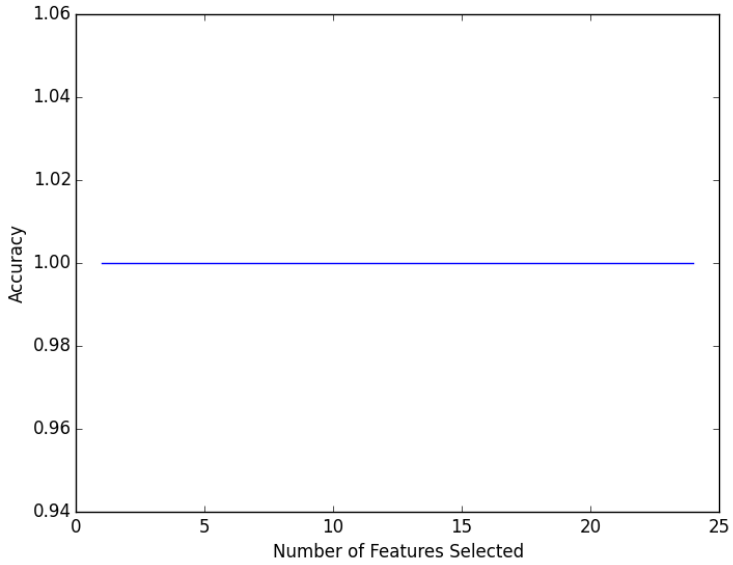


Optimising the value of K at the optimal number of folds for KNN

Finally, the number of features was to be evaluated. Features were selected via sklearn's `SelectKBest` built-in class. SelectKBest scores features according to their apparent utility in classification, by comparing the entire dataset. SelectKBest was chosen for implementation simplicity; the ability to supply one's own scoring function made the class versatile, yet very simple to integrate to the model constructed.



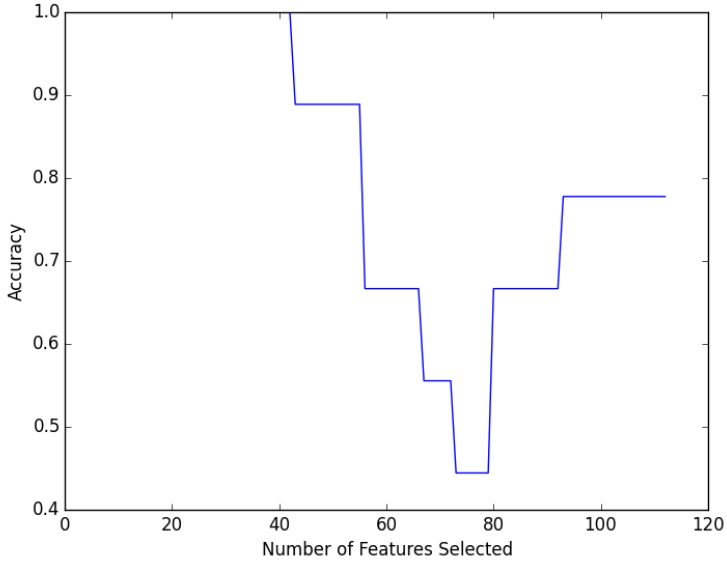
Performance over features chosen for KNN



Performance over features chosen for GNB

KNN becomes even more performant, even at low feature counts, and frequently achieves higher than 90% accuracy with the number of cross validation folds and the value of K having been optimised already.

GNB, however, sits at 100% accuracy without noisy features. Instead of limiting the experiment to the first 25 features, we might graph out how it performs on *all* numbers of features selected:



Performance over features chosen for GNB — considering all 113 features

We now see that GNB doesn't begin to lose performance until noise is introduced at feature 43. GNB is particularly performant, therefore, once optimised — beating KNN by a significant degree.[8]

[8] Though KNN is also rather performant.

Discussion

It is evident — particularly when looking at the classifiers' performances with an optimised feature set — that GNB is more performant than KNN for this dataset. However, some subtleties stand out: both algorithms performed relatively poorly before optimisation, and both GNB and KNN performed very well once fully optimised for the dataset.

This result indicates that a probabilistic classification algorithm is better suited to this dataset than a non-probabilistic one. This might be a result of GNB's slight sophistication in its classification philosophy as compared to KNN; the results of this experiment do not determine scientifically which classification algorithm is "better" — nor does it explore why that might be — but we can determine that, *for this dataset*, GNB is definitively more performant than KNN.

Literature Review

In Fuchs & Buhmann, 2011 [9], the authors note that the field of computational pathology can be split into two three distinct stages: *Data*, which feeds into *Imaging*, and culminates in *Statistics*.

All of these sections are important in the classification process, as they make clear. As Fuchs et al. note, however, some results in the field are still to be properly established by the research community. One example of this is "*Multiple Expert Learning*", which can operate successfully in certain scenarios, but is a technique which requires further analysis by the field at large. One example of further research in the area can be found in Dekel & Shamir, 2009.[10]

[9] Thomas J Fuchs and Joachim M Buhmann. Computational pathology: Challenges and promises for tissue analysis. *Computerized Medical Imaging and Graphics*, 35(7):515–530, 2011

[10] Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *COLT*, 2009

Dekel & Shamir produce a series of theorems to assist in crowd-sourced data research. Practically, the work serves two purposes: it demonstrates practical usage of the theorems it proves, and also demonstrates how researchers can explore their theorems in future empirical work with tools such as Amazon's *Mechanical Turk*.

One can imagine, therefore, that data collection and pruning will only advance as tools to generate data become more sophisticated, and make use of modern techniques such as crowdsourcing. Another important component of the computational pathology field is found in imaging, however, producing and processing additional forms of data. One dichotomy, as explained in Gurcan et al.[11], lies in the differences between Histopathology and Cytopathology.

[11] Metin N Gurcan, Laura Boucheron, Ali Can, Anant Madabhushi, Nasir Rajpoot, and Bulent Yener. Histopathological Image Analysis: A Review. *IEEE Reviews on Biomedical Engineering*, 1:147–171, 2009

Histopathology is the study of how a disease manifests in tell-tale signs under microscopic examination. Cytopathology, by contrast, studies how the cell physically changes in terms of its composition and function.

Imaging techniques exist for these methods can differ; for example, Histopathological study frequently makes use of staining techniques to simplify component identification, which lends itself to some multi-spectrum imaging methods (as noted in [12]). Cytopathology, in contrast, identifies simpler components in its imaging process. The components are therefore less visually complex, but also lend themselves less readily to emerging imaging technologies.

[12] Thomas J Fuchs and Joachim M Buhmann. Computational pathology: Challenges and promises for tissue analysis. *Computerized Medical Imaging and Graphics*, 35(7):515–530, 2011

In discussing these emerging technologies, the topic of statistics — the actual learning mechanism — is yet unexplored. One particularly important note on statistical work is its application: specific statistical methods are developed for tackling expectation of cancer survival, but other techniques, such as mixtures of predictions from survival experts via a technique such as kernel mixture survival models [13], take similar approaches to the combinations of predictions as [14]'s

[13] Tomohiro Ando, Seiya Imoto, and Satoru Miyano. Kernel mixture survival models for identifying cancer subtypes, predicting patient's cancer types and survival probabilities. *Genome informatics. International Conference on Genome Informatics*, 15(2):201–210, 2004

[14] Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *COLT*, 2009

crowdsourced data approach.

The field is clearly diverse, and an understanding of biological aspects, statistics and imaging techniques are a brief overview of the diverse fields one must be acquainted with when exploring computational pathology. However, the field is also expanding rapidly and continues to innovate. Interesting future work might compare the crowdsourced data techniques in computational pathology to similar techniques in software engineering, such as planning poker[15].

[15] J Grenning. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, (April):1–3, 2002