

Multi-stage Redundancy Reduction: Effective Utilisation of Small Protein Data Sets

John Hawkins

Mikael Bodén

School of Information Technology and Electrical Engineering
The University of Queensland,
St Lucia, QLD, Australia,
Email: jhawkins@itee.uq.edu.au

Abstract

In many important bioinformatics problems the data sets contain considerable redundancy due to the evolutionary processes which generate the data and biases in the data collection procedures. The standard practice in bioinformatics involves removing the redundancy such that there is no more than at most forty percent similarity between sequences in a data set. For small data sets this can dilute the already impoverished data beyond the boundary of practicality. One can choose to include all available data in the process by just ensuring that only the training and test samples have the required redundancy gap. However, this encourages overfitting of the model by exposure to a highly redundant training sets. We outline a process of multi-stage redundancy reduction, whereby the paucity of data can be effectively utilised without compromising the integrity of the model or the testing procedure.

Keywords: Redundancy Reduction, Generalisation Estimation, Cross Validation

1 Introduction

An essential part of protein data set development for machine learning applications in bioinformatics involves removing redundancy so that the bias in the data is minimised (Hobohm, Scharf, Schneider & Sander 1992). The redundancy reduction helps prevent a model over fitting to the bias in the data collection processes, and prevents predictive accuracy being overestimated.

However, there are a number of biological problems where the data sets are comparatively small simply due to the fact that they relate to subtle aspects of cellular life. In these instances we are faced with a problem: "How to train and test our models so that we best utilise the available data and do not bias our tests?".

One solution that has been proposed to this problem involves giving a weighting to each of the data points to correct for biases in the training data (Krogh & Mitchison 1995, Eddy, Mitchison & Durbin 1995). The prime difficulty posed by such an approach is that not all learning algorithms are conducive to using weighted samples. In order to employ this technique one needs to restrict the type of model used, or modify an existing model to accommodate the weightings. Recently the weighting of samples has been extended

to the calculation of performance metrics (Budagyan & Abagyan 2006). The authors concluded that one need not perform an artificial reduction of the data set if the testing is performed using a weighted metric. Their results indicated that inclusion of redundant data can improve the performance of the model.

We present a simple solution to the problem of making effective use of small data sets without compromising testing rigour. The technique takes the form of a regime for gradual data set reduction as the model moves from training to testing. The redundancy reduction occurs in three stages, allowing small amounts of redundancy within the training sets, but rigorously excluding it between training and test sets. Furthermore, we provide a clear indication of generalisation improvement offered by redundancy reduction by showing that performance is optimal when not using all available data, but by allowing only small amounts of redundancy within the training sets.

2 Background

The effective application of machine learning techniques to problems of classification is not simply a matter of training a model on all available data. Although a model produced in this fashion will perform very well on data with similarity to the training data it will tend to fail on genuinely novel data. Hence, the performance statistics produced by cross-validation on data sets containing redundancy will not be a reliable guide to their ability to generalise.

The problem stems from two sources, firstly used a data set that over represents some region of the problem space encourages the model to over fit this data. This is less of a problem if the bias is present in the real world, however often these biases are due to the collection of data. The second problem with redundant data is that when used to estimate the performance of the model it will bias those estimates due to the fact that the test points are very close to repetitions of the training points. When a model that is trained and tested on redundant data "the apparent predictive performance may be overestimated, reflecting the method's ability to reproduce its own particular input rather than its generalization power" (Baldi & Brunak 2001) (page 6).

Considerable effort has gone into developing algorithms to minimise the redundancy within the data yet maximise the amount left with which to build models (Hobohm et al. 1992). A *de facto* standard has emerged in bioinformatics to perform a redundancy reduction of data sets using sequence homology scores, such that no two sequences have greater than 25%-40% identical residues across a specified length. This threshold is no doubt due to the fact that it is the so called 'twilight' region in which sequence alignments become a poor indicator of homology (Rost 1999).

However, what remains unaddressed is whether the two reasons for redundancy reduction require the same level of reduction in order to mitigate their respective problems. This issue forms the central question of this study and the answer to which suggests the technique of multi-stage redundancy reduction as a method of effectively utilising small data sets.

3 The Method

The essence of the technique is as follows: we perform an initial redundancy reduction of the data in order to remove the sequences that are most similar. The redundancy reduction is performed using BLASTCLUST to generate clusters with a specified level of similarity. From these clusters a single sample is chosen as the representative of the cluster. The *initial reduction threshold*, Θ_i , is a permissive threshold, between 40% and 90% similarity. The sequences remaining after the initial reduction comprise the data set for the purposes of training the model.

In order to ensure that the testing procedure is not compromised by the existence of some redundancy in the data, we perform a second clustering at the *final reduction threshold*, Θ_f , set to a low enough level as to guarantee a rigorous testing. These clusters are then used to generate the subsets of data for the cross-validations. Each of the clusters is allocated to one of the subsets. So that all the redundancy exists within the cross validation subsets. This ensures that there is no redundancy between the sequences that are used for training and those used for testing, such that the cross-validation is assured to be an adequate test of generalisation.

The cross-validation is then performed such that the partially redundant data is used in the training of the models. However, in each iteration, the set that has been allocated for testing undergoes a further reduction at the *final reduction threshold* Θ_f , to remove the remaining redundancy. In this way we allow the models to utilise some redundancy in the training data, but remove all redundancy from the testing procedure.

The procedure is demonstrated schematically in Figure 1.

4 Data sets

For the purposes of demonstrating the utility of the technique we apply it to two different data sets of proteins. Both data sets revolve around the importation of proteins into the peroxisome, which is a small but important organelle in eukaryotic cells. The two mechanisms are named after the sequence signals which the proteins rely on for recognition and import. Peroxisomal Targeting Signal One (PTS1) and Peroxisomal Targeting Signal Two (PTS2), are both subtle distinct sequences within the protein that allow import into the organelle through distinct protein pathways (Baker & Sparkes 2005, Michels, Moyersoen, Krazy, Galland, Herman & Hannaert 2005). The crucial aspect of these data sets for the current paper is that the peroxisome has a small protein complement, hence the data sets are small.

The PTS1 pathway relies on an C-terminal tripeptide and sequence of nine preceding residues that support its recognition. We have outlined the extraction of this data set in previous studies (Wakabayashi, Hawkins, Maetschke & Bodén 2005, Hawkins & Bodén 2005). The data set derivation relies on a biologically informed template for the import signal. This template fits all known positives, but due to its generality fits a larger number of negatives.

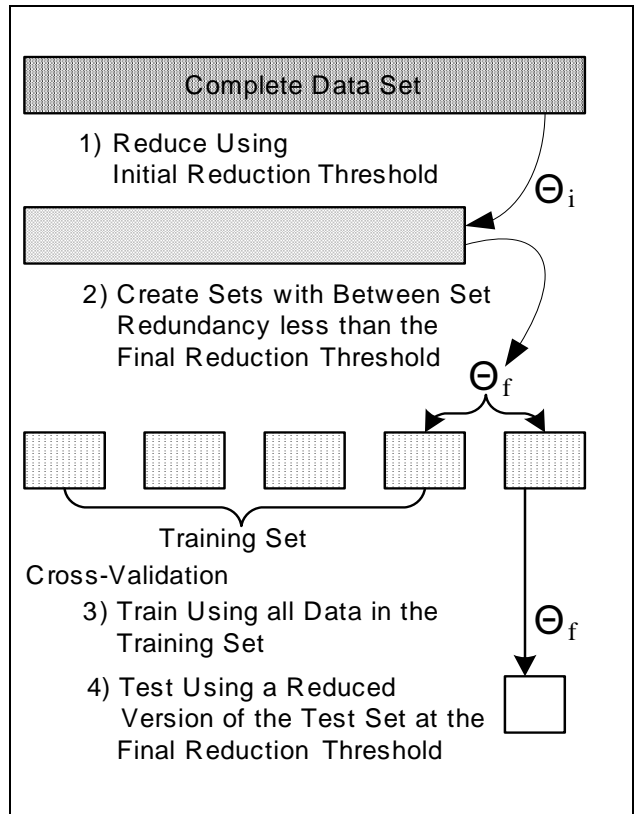


Figure 1: Schematic Representation of the Multi-Stage Redundancy Reduction Training and Testing Procedure.

The PTS2 pathway relies on a 9-mer with an unspecified position, although most instances occur in the N-terminal region. There are far fewer known instances of PTS2 proteins, and to make matters worse the template signal is less specific. Hence, the training sets for differentiating between real and fallacious PTS2 instances are highly unbalanced. We have outlined the extraction of this data set in a previous study (Bodén & Hawkins 2006). Due to the size of the negative set we maintain the heavily reduced version (2799 proteins reduced at a 10% threshold) and instead focus the multi-stage redundancy reduction on the positive set only.

The numbers of proteins that result from the varying levels of redundancy reduction are shown in Table 1.

5 Simulations

In order to demonstrate the effectiveness of the technique and identify a threshold for the initial redundancy reduction we perform a range of simulations over each of the data sets. For each data set we produce a set of versions each of which have been through an initial redundancy reduction. For these initial reductions we use a set of thresholds varying from 100 (for no reduction) down to 30. The number of proteins in the data sets for each of these threshold is shown in Table 1.

As a model to train on these problems we have chosen a Support Vector Machine with a spectrum kernel. The spectrum kernel is a general purpose sequence kernel that is efficient to run and has proven effective on a wide range of bioinformatics problems. The spectrum kernel takes one parameter, k which defines the length of the sequence segments considered in the spectrum.

For a given sequence, the spectrum of the sequence

Data set		Redundancy Reduction Threshold									
		100	95	90	85	80	70	60	50	40	30
PTS1	Positives	139	131	114	97	83	66	62	56	50	47
	Negatives	291	256	230	208	193	180	171	165	164	163
PTS2	Positives	97	91	81	69	62	56	51	41	37	35

Table 1: Numbers of Proteins in each data set as the initial redundancy reduction threshold is varied. At the upper limit, a threshold of 100 means no reduction is performed. At the lower end of 30% the threshold is identical to that used to distinguish the test sets, hence the process is equivalent to the standard process of a single redundancy reduction prior to training and testing.

is the set of all k -mers it contains. The Spectrum kernel compares any two sequences by considering the number of these k -mers that two sequences share (Leslie, Eskin & Grundy 2002). More specifically, the kernel calculates the dot product between the vectors holding all k -mer counts for any pair of sequences. If two sequences share a large number of k -mers they produce a large spectrum kernel value.

For both data sets we explore k values ranging between 1 and 5. We run these on each of the data sets, such that the initial reduction threshold ranges from 100% (No Reduction) to 30% (Single Stage Reduction). We run each configuration as a ten-fold cross-validation, ten times using a different seed to split the data for the cross validations.

We use our own implementation of the spectrum kernel that runs with a modified version of the LIBSVM package (Chang & Lin 2001). The C value of an SVM is commonly called the regularisation constant and indicates the penalty that is applied to samples that are positioned on the wrong side of the decision boundary. For the PTS1 problem we use a general C value of 0.5, however for the PTS2 problem, due to the massive imbalance of the data, we modified the code such that the positive samples use $C = 1000$ and the negative samples use $C = 0.002$. These values were found through a number of trial runs as a method of forcing the learning to give equal emphasis to both classes.

6 Results

The results for each of the spectrum kernels on the PTS1 problem are shown in Table 2. In each case we show the mean Matthews' Correlation Coefficient MCC over the ten independent runs. The MCC is calculated using the formula:

$$r(c) = \frac{tp_c tn_c - fp_c fn_c}{\sqrt{(tp_c + fn_c)(tp_c + fp_c)(tn_c + fp_c)(tn_c + fn_c)}} \quad (1)$$

The data are shown graphically in Figure 2 with a standard error bar showing the estimated standard deviation of the mean. This is calculated with the formula:

$$stderr = \frac{\sigma}{\sqrt{N}} \quad (2)$$

Where σ is the standard deviation of the test statistic, in this case the MCC, and N is the number of samples, in this case 10.

For three of the four k values tried the multi-stage redundancy reduction technique produced the best model. The best overall model produced used an initial redundancy reduction threshold of 85% and a k value of 2. As we can see in Figure 2, the results are somewhat variable across the different kernels and thresholds. In spite of the lack of a consistent trend, if we compare the results of the best model at threshold 30 with the best overall model, the standard error bar

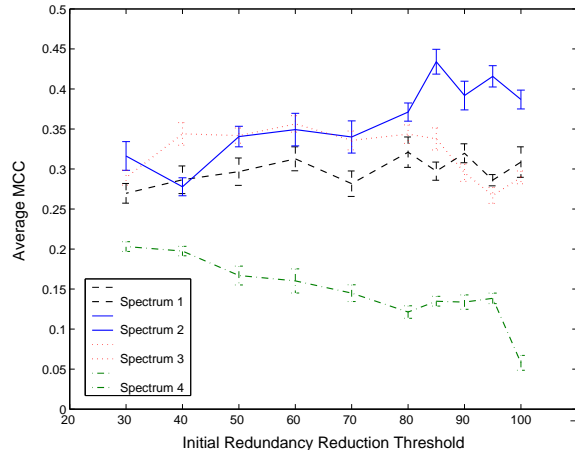


Figure 2: The average MCC for the Spectrum Kernel on the PTS1 problem, plotted against the initial threshold of redundancy reduction. The error bars depict one standard error on either side of the mean. Data generated from ten runs of ten-fold cross-validation.

indicates that using some redundancy in the training produces significantly better results than not. However, due to the overlap of the standard error distributions it is not possible to say whether the apparent improvement offered by the multi-stage redundancy reduction is significant.

Similarly the results for each of the spectrum kernels on the PTS2 problem are shown in Table 3.

For the PTS2 problem we see that for all four of the spectrum values used the best performing kernel was produced under multi-stage redundancy reduction. The best overall model produced used an initial redundancy reduction threshold of 85% and a k value of 4. As we can see in Figure 3, the results are much more consistent across the different kernel settings. Three of the four performing best with an initial threshold between 80–85%. The effect is most pronounced with $k = 4$, where under the standard practice or using all data the kernel performs very poorly. In this case the best model performs significantly better under the multi-stage redundancy reduction scheme than either using all data, or single-stage redundancy reduction.

It is interesting to note that in both case studies the majority of the kernels perform best when allowed to use some of the redundant data. In some cases profoundly better than if all that data was used or it was simply discarded. What we see clearly in both figures is that the best model for each problem was created using the multi-stage redundancy reduction procedure with an initial reduction of 85%.

PTS1 Simulation Results

k-value	Initial Redundancy Reduction Threshold									
	100	95	90	85	80	70	60	50	40	30
1	0.3087	0.2861	0.3197	0.2973	0.3210	0.2817	0.3127	0.2967	0.2866	0.2697
2	0.3868	0.4158	0.3918	0.4341	0.3711	0.3401	0.3492	0.3405	0.2778	0.3163
3	0.2896	0.2669	0.2954	0.3376	0.3436	0.3357	0.3563	0.3418	0.3439	0.2892
4	0.0578	0.1384	0.1337	0.1348	0.1213	0.1449	0.1602	0.1669	0.1974	0.2031

Table 2: Average MCC values for the Spectrum Kernel run with different k-values and different thresholds for the initial redundancy reduction. The first column with an initial threshold of 100% involves using all available data to train the models. The final column, at 30%, involves a single redundancy reduction prior to training and testing. Intermediate values are the result of the two stage redundancy reduction procedure.

PTS2 Simulation Results

k-value	Initial Redundancy Reduction Threshold									
	100	95	90	85	80	70	60	50	40	30
1	0.0823	0.0812	0.0832	0.0873	0.0842	0.0846	0.0902	0.0868	0.0973	0.0903
2	0.0886	0.0872	0.1107	0.1291	0.1281	0.1045	0.0858	0.0812	0.0533	0.0507
3	0.0797	0.0818	0.1212	0.1369	0.1444	0.0648	0.0336	0.0205	0.0116	0.0356
4	0.0271	0.0292	0.0295	0.1588	0.1419	-0.0035	-0.0040	-0.0031	-0.0030	-0.0029

Table 3: Average MCC values for the Spectrum Kernel run with different k-values and different thresholds for the initial redundancy reduction. The first column with an initial threshold of 100% involves using all available data to train the models. The final column, at 30%, involves a single redundancy reduction prior to training and testing. Intermediate values are the result of the two stage redundancy reduction procedure.

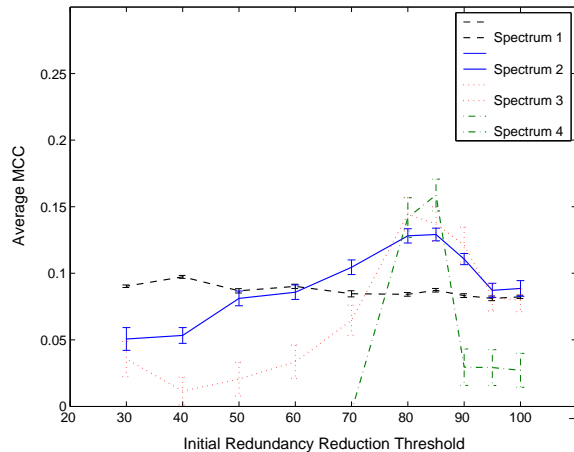


Figure 3: The average MCC for the Spectrum Kernel on the PTS2 problem, plotted against the initial threshold of redundancy reduction. The error bars depict one standard error on either side of the mean. Data generated from ten runs of ten-fold cross-validation.

7 Conclusion

Some key biological problems have only small amounts of data available for the building of models. It is therefore crucial to make effective use of the paucity of available data. Data sets typically undergo a process of redundancy reduction for two reasons: To prevent the model from over fitting to the bias present in the data, and to ensure that our testing of the model’s generalisation ability is rigorous. Typically the redundancy reduction is done once as the data set is curated with the implicit assumption that the threshold for reduction should be identical for both of these purposes.

We have shown that by treating these two purposes of redundancy reduction separately, we are able to increase the amount of data available to train our models. By using rigorous testing procedures we have shown that the optimal thresholds of redundancy for these two purposes are not identical. I.e the reduction required in order to prevent over fitting is less than that required to perform rigorous testing. It appears that one can produce a superior model by allowing it to train on data with a mild amount of redundancy.

References

- Baker, A. & Sparkes, I. A. (2005), ‘Peroxisome protein import: some answers, more questions’, *Current Opinion in Plant Biology* **8**(6), 640–647.
- Baldi, P. & Brunak, S. (2001), *Bioinformatics : the machine learning approach, Second Edition*, MIT Press, Cambridge, Mass.
- Bodén, M. & Hawkins, J. (2006), Evolving discriminative motifs for recognizing proteins imported to the peroxisome via the pts2 pathway, in ‘Proceedings of the IEEE Congress on Evolutionary Computation’, IEEE, Canada, pp. 9300–9305.
- Budagyan, L. & Abagyan, R. (2006), ‘Weighted quality estimates in machine learning’, *Bioinformatics* p. btl458.

- Chang, C.-C. & Lin, C.-J. (2001), *LIBSVM: a library for support vector machines*. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Eddy, S. R., Mitchison, G. J. & Durbin, R. (1995), 'Maximum discrimination hidden markov models of sequence consensus.', *Journal of Computational Biology* **2**(1), 9–23.
- Hawkins, J. & Bodén, M. (2005), Predicting peroxisomal proteins, in 'Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology', IEEE, Piscataway, pp. 469–474.
- Hobohm, U., Scharf, M., Schneider, R. & Sander, C. (1992), 'Selection of representative protein data sets', *Protein Science* **1**(3), 409–417.
- Krogh, A. & Mitchison, G. (1995), Maximum entropy weighting of aligned sequences of proteins or dna, in 'Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology', pp. 215–221.
- Leslie, C., Eskin, E. & Grundy, W. S. (2002), The spectrum kernel: A string kernel for svm protein classification, in R. B. Altman, A. K. Dunker, L. Hunter, K. Lauerdale & T. E. Klein, eds, 'Proceedings of the Pacific Symposium on Biocomputing', World Scientific, pp. 564–575.
- Michels, P., Moyersoen, J., Krazy, H., Galland, N., Herman, M. & Hannaert, V. (2005), 'Peroxisomes, glyoxysomes and glycosomes', *Molecular Membrane Biology* **22**(1 - 2), 133–145.
- Rost, B. (1999), 'Twilight zone of protein sequence alignments', *Protein Eng.* **12**(2), 85–94.
- Wakabayashi, M., Hawkins, J., Maetschke, S. & Bodén, M. (2005), Exploiting targetting signal dependencies in the prediction of pts1 peroxisomal proteins, in M. Gallagher, J. Hogan & F. Maire, eds, 'Intelligent Data Engineering and Automated Learning - IDEAL 2005: 6th International Conference', Vol. 3578 of *Lecture Notes in Computer Science*, Springer, pp. 454–461.