Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Manuscript Category

Protein Classification

Predicting the subcellular location of eukaryotic proteins

Eric Hambro [1,*]

[1] eric.hambro.17@ucl.ac.uk, SN: 17096751, Machine Learning MSc, UCL

Abstract

Motivation: Predicting subcellular localization from raw sequence data is a important research problem, with applications in drug discovery and medecine research. This paper presents a comparison of two machine learning classifiers to do this: Random Forests and AdaBoost. Using features derived from the sequence, these models were trained to classify non homologuous protein sequences into four localizations: cytosolic, nuclear, secretory, mitochondrial. These models were compared to each other and a baseline, and the best was chosen to predict localizations for 20 unknown proteins.

Results: The Random Forest classifier performed best in the comparison, with a test accuracy of 67.5%, and F1 score of 68.3% . This outperformed AdaBoost's 64.4% test accuracy and F1 of 62.6%. Both beat the benchmark significantly, and the Random Forest model was chosen to make the final predictions.

Contact: eric.hambro.17@ucl.ac.uk

1 Introduction

Discovering subcellular localization of proteins is a vital piece of information in many applied fields in cell biology. Not only is it important to help discover and catalogue the complex chemical pathways and signals that occur inside the cell, but it has vital medical applications, in particular for drug discovery. Furthermore, in cases where no homology is evident for a gene, subcellular localization of the protein can be used an vital indicator of the potential function of that protein.

Beginning in the late 90s, machine learning started to be used in subcellular localization prediction in an attempt to reduce the cost and time taken to process protein sequences. Traditional methods of determining subcellular location, such as tagging proteins with fluorescent markers, were laborious and time intensive, and had been unable to keep up with the rapidly proliferating amount of genetic and protein sequence data.

The first of these such machine learning models was PSORT by Nakai *et al*. (1999), who used handcrafted sequence features and the taxon of origin to predict localization, using a k-nearest neighbours model. Since then a number of improvements have made to the field, through use of a variety of different machine learning models, and specialising in different organism's cells. For instance, recent models such as REALoc (Tung *et al*. (2017)) combine SVMs and Generative Adversarial Neural Networks in layers, to predict 6 subcellular localizations in human proteins, achieving a test accuracy on an independent database surpassing 6 other models by over 10%.

In our particular experiment we are presented with a labeled data set of 9222 proteins and we predict and evaluate both a Random Forests model and AdaBoost model on these data, comparing both to a baseline random model. The proteins themselves each correspond to one of four subcellular localizations - cytosolic, secretory, nuclear or mitochondrial - and after training the models, we make predictions of locations for 20 unknown proteins in these classes, before analysing our own test errors and models.

Section 2 of this paper contains a brief background of biological and machine learning elements used in the experiment, whereas Sections 3 and 4 presents the methods and results of the experiments. In Section 5 a discussion of the results is presented, along with a comparison of the models, their features, and analysis of the errors. A short conclusion is presented in Section 6.

2 Background

2.1 Subcellular Localization & Transport

In eukaryotic cells, proteins are often highly adapted to fulfilling their specialist functions in the cell. These functions can be very localized within organelles or other parts of the cell, and so the proteins are adapted to the specific subcellular location in the cell. In order to faciliate transport of protein to its location, often a location signifier is in some way encoded into the structure and sequence of the protein. These are known as target peptides or signal peptides. These peptides are sometimes removed by a signal peptidase when they reach their destinations, and so have little other functional significance. They can vary greatly in structure, depending on their final destination, and can be explored in greater depth in Chapter 12 of Alberts, *et al*. (2008). Below are some of the known features of signal and target peptides, as mentioned in the above book.

1. **Targeting Secretion**
   Proteins headed toward secretory pathways often contain short amino acid encodings (approximately 16-30 amino acids long), occuring near the N-terminus of the protein. Often they include a domain or sequence that is particularly hydrophobic, and these hydrophobic residues may form an α-helix.
2. **Targeting Mitochondria**
   In the case of proteins headed to mitochondria, a *mitochondrial targeting signal* of a peptide signal is often found at the N-terminus of the protein. These often consist of alternating positively charged and hydrophobic amino acids, that come together to form an amphiphilic α-helix. This pattern (of positively charged residues on one side, and uncharged hydrophobic residues on another) is what is primarily recognised by the receptor proteins, rather than a specific sequence.
3. **Into The Nucleus**
   Proteins that require importing into the nucleus often contain a *Nuclear Localization Signal*, which may consist of a sequence amino acids that are positively charged (often arginine and lysine) exposed on the surface of the protein. In contrast with the other target peptides, it appears these signals can occure almost anywhere in the chain, and often form loops. Many of these are still uncharacterised.
4. **Out of the Nucleus (Into The Cytosol)**
   Proteins that are are being exported out of the nucleus and into the cytosol are often tagged with a *Nuclear Export Signal*. A known example of *Nuclear Export Signal* consists of a chain of several (hydrophobic) leucines, interspersed with other amino acids, between them. Since some proteins frequently move between the nucleus and the cytosol, it is common for some proteins to contain both *Nuclear Export Signals* and *Nuclear Localization Signals*, which are effectively controlled by the cell to direct movement.

2.2 Bagging & Boosting

Our investigation compares two machine learning models to a benchmark random classifier, selecting the best. These two models (Random Forests and AdaBoost) were selected on account of being good are examples of common ensembling techniques: *bagging* and *boosting*.

2.2.1 Bagging & Random Forests

Bootstrap aggregation (or *bagging*) is an ensemble method to the reduce the variance of a prediction when using high variance classifiers. It involves, selecting many subsamples of the training data, and generating estimates from the average performance of classifiers trained on this data. Thanks to this averaging of predictions, overfitting our training data becomes less of a risk, since each classifier can overfit to it's subsection of the data without impacting the average too greatly in the end.

In a Random Forest, our high-variance classifier is a decision tree, and we adapt the *bagging* algorithm to improve performance. Specifically we only give each individual classifier access to a random subset of the features of the data, at training time. The aim of this is to stop any correlation on the decision trees that may arise from their greedy behaviour, as they look through variables to select the best split point. By only having access to a random subset of features, the decision trees become less correlated, and the Random Forest algorithm produces a better averaged estimate.

2.2.2 Boosting & AdaBoost

The AdaBoost algorithm is a different method that uses a ensemble technique known as *boosting*. Instead of a combining votes from many high variance normal classifiers (like a decision tree), *boosting* involves learning by combining lots of weak learners (like a decision stump), by training the on particularly chosen subsections of the data, and combining votes in a particular way. Specifically, each successive weak-learner is trained on selections of the data that aim to emphasise the most misclassified data points. These learners are then weighted by their error, to provide the overall classification.

Table 1. Experimental Data Set By Class

Cytosolic	Mitochondrial	Nucleic	Secreted
3004	1299	3314	1605

The total available labeled data used in this experiment.

3 Method

3.1 Preprocessing

Data was obtained from four files containing protein sequences in Fasta format. Each file contained proteins according to its type of subcellular localization: cytosolic, secreted, nuclear and mitochondrial. The number of proteins in each file is presented in Table 1. These data files contained no homologues, and therefore could be assimilated, shuffled and split into a training and validation set randomly. The data was then split into a test set that would be held out for final evalution (20%), and a train and validation set (80%) that could be used to tune hyperparameters.

The data was then fed through a pipeline to extract a number of features from the sequence. These features all corresponded to floating point numbers, and so were concatenated to form a 75-dimensional feature vector, that would be used to train the models.

3.2 Features

The open source BioPython module (Cock *et al.* (2009)) was used to extract features from the sequence data, allowing us to augment raw residue sequence data with important characteristics of the amino acids, such as isoelectronic point or molecular weight. In total 75 features were created, although many of these could be grouped into the same category. These features were:

- **Protein Length** - An integer representing the length of the protein sequence, in residues.
- **Amino Acid Composition** - A float created for each amino acid, corresponding to the percentage composition of that amino amino acid in the protein.
- **Aromaticity** - A float corresponding to a measure of the aromaticity of the molecule according to Lobry *et al.* (1994). This is measured by adding the frequencies of residues phenylalanine, tryptophan and tyrosine, in the sequence of the protein.
- **Isoelectric Point** - A float representing the isoelectric point of the protein according to values taken by Bjellqvist *et al.* (1994).
- **Molecular Weight** - A float simply molecular weight of the protein.
- **Secondary Structure Features** - Three separate features were created indicating the fraction of amino acids which tend to be either α-helixes, β-sheets, or turns/loops. These amino acids were: α - Val, Ile, Tyr, Phe, Trp, Leu; β - Glu, Met, Ala, Leu; *loops* - Asn, Pro, Gly, Ser
- **Start/End Amino Acid Composition** - Two floats were created for each amino acid, according to the percentage composition of that amino amino acid in the first and last 20 residues of the protein respectively.

These features were chosen as they had a high impact on the accuracy of the classifier. The importance of each of these features to various classifications is examined in section 4.

3.3 Classifier Training & Tuning

Three classifiers were trained to investigate the best possible model in predicting the subcellular location. These models were:

- Categorical Distribution Classifier (Random Benchmark)
- Random Forests Classifier
- Adaboost Classifier

All classifiers were implemented using the open source *scikit-learn* package (Pedregosa *et al*. (2011)). In all cases, model parameters were fit with a 5-fold cross validation, on the train and validate set mentioned previously. Hyperparameters were tuned using a grid search, and chosing the parameters that had the best 5-fold cross validation score of accuracy. The best of these 5-fold models was then taken for each classifier, and was tested on the held out test set, evaluating both the overall accuracy, and the precision and recall of individual categories.

It is worth noting that in the Random Forests Model, multiple models were trained, each solely to predict one class, instead of a single multi-class classifier. Although this is strictly unnecesssary for a Random Forests Classifier, it provides the additional advantage of allowing for interpretable feature importances, and more interpretable analysis of the individual class classifiers. To ensure a fair analysis of the classifier, a single multi-class Random Forests Model was also implemented, though this was found to have the same accuracy as the individual models, to within experimental error. In this multi-model case, *scikit-learn* provided the best means for creating the 4 mutually exclusive classifiers from Random Forests, using it's One-Vs-Rest Classifier wrapper.

3.4 Benchmarks

A raw, bottom level benchmark for this investigation was a random classifier, here referred to as the Categorical Classifier. This classifer is so called because it simply predicted categories for data points randomly following a simple categorical distribution, with the probability vector $\mathbf{p}$ learnt from the calculating frequency of each class in it's training set. This model was implemented by hand, using the inheritance properties of *scikit-learn*'s BaseEstimator class, to allow easy integration into the *scikit-learn* pipeline.

4 Results

4.1 Final Hyperparameters

Table 2 shows results of the hyperparameter grid search for each of the models, presenting the best hyperparameters according to the mean cross-validation accuracy. These hyperparameters were the ones used for all following data analysis.

Table 2. Hyperparameters of Model

Model	Hyperparameter	Settings
Categorical (Benchmark)	N.A.	N.A.
Random Forests Classifier	estimators	450
Adaboost Classifier	estimators	200
	learning rate	0.25

4.2 Model Scores

With the hyperparameters selected and 5-fold cross-validation performed, the most accurate models from the cross-validation were chosen as "final models", and evaluated on the held out test data-set. Table 3 shows both the results of the cross-validation accuracy (both train and validation), and the accuracy and F1 score of the best model on the held out test set. Further interrogation of these models in then presented in Table 4, where the precision, recall and F1 score for each classes of each model is presented.

Table 3. Accuracy of Models

Model	CV Train Acc	CV Valid Acc	Test Acc	Test F1
Categorical (Benchmark)	0.283 ± 0.004	0.291 ± 0.011	0.277	0.273
Random Forests	1.00 ± 0.000	0.682 ± 0.005	0.675	0.683
AdaBoost	0.650 ± 0.003	0.634 ± 0.013	0.643	0.626

Mean cross-validation accuracies with test accuracies and F1 scores.

Table 4. Classification Metrics

Model	Class	Precision	Accuracy	F1	Support
Categorical	Cyto	0.326	0.333	0.329	604
	Mito	0.144	0.140	0.142	271
	Nucl	0.363	0.363	0.363	653
	Secr	0.210	0.206	0.208	316
Random Forests	Cyto	0.602	0.556	0.578	604
	Mito	0.804	0.668	0.730	271
	Nucl	0.639	0.714	0.674	653
	Secr	0.837	0.880	0.858	316
AdaBoost	Cyto	0.506	0.495	0.500	604
	Mito	0.694	0.661	0.677	271
	Nucl	0.589	0.640	0.613	653
	Secr	0.895	0.807	0.849	316

Metrics calculated on test set, using best trained model

Table 5. Predictions of Test Proteins

Protein ID	Location	Probabilty	Protein ID	Location	Probabilty
SEQ677	Cyto	0.416	SEQ608	Cyto	0.472
SEQ231	Cyto	0.307	SEQ402	Mito	0.628
SEQ871	Nucleus	0.381	SEQ433	Secreted	0.804
SEQ388	Cyto	0.360	SEQ821	Secreted	0.795
SEQ122	Cyto	0.430	SEQ322	Nucleus	0.680
SEQ758	Nucleus	0.560	SEQ982	Nucleus	0.813
SEQ333	Cyto	0.494	SEQ951	Nucleus	0.483
SEQ937	Cyto	0.686	SEQ173	Cyto	0.495
SEQ351	Cyto	0.517	SEQ862	Mito	0.662
SEQ202	Mito	0.694	SEQ224	Cyto	0.381

Predictions made with Random Forest model

From these tables it is clear that not only did the Random Forests Model perform best overall but it also performed best in every single individual class. AdaBoost performed only slightly worse in accuracy, 0.643 compared to Random Forests 0.675, although it performed worse under the F1 metric (0.626 compared to 0.683), where the F1 metric was an average of class F1s, weighted by the number of classes present in the test set. Both models significantly outperformed the categorical benchmark, with its accuracy of 0.277.

Furthermore in all cases, the test set accuracies were very close to the mean cross-validation accuracies, which themselves had low variance. This indicates little overfitting of the hyperparameter choice to the validation set - as desired.

4.3 Predictions

Since the Random Forest Model was selected as our best model, it was subsequently used to to make predictions of 20 hitherto unclassified proteins. These prediction, along with probability confidence scores, are given in Table 5.

4.4 Plots

Finally, to investigate the importance of various features, the ROC curves were plotted for our Random Forests model (Figure 1), along with the confusion matrix for its classifications (Figure 2). Furthermore the top 25 importance weights of the individual classifiers were also plotted, to aid in this investigation. These are presented in Figures 3 & 4 .

5 Discussion

5.1 Classifier Comparison

It is unsurprising that both boosting and bagging methods far outperformed the benchmark method. The woeful performance of 0.277 accuracy is to be expected for a model that doesn't take into account any of the sequence data, or underlying features. However, the significant discrepancy between Random Forests and AdaBoost is more difficult to explain, and probably comes down to the high variance nature of the underlying data. In general, boosting methods struggle on very noisy data, as in the best cases bagging acts to reduce variance, whereas boosting tends to reduce bias. It is interesting to note that the AdaBoost classifier only really approached Random Forest performance in the case of the Secreted label (F1 - RF 0.858 vs Ada 0.849), which is probably the class with the least 'noise', or strongest 'signal', judging by the strength of the leading feature importances.

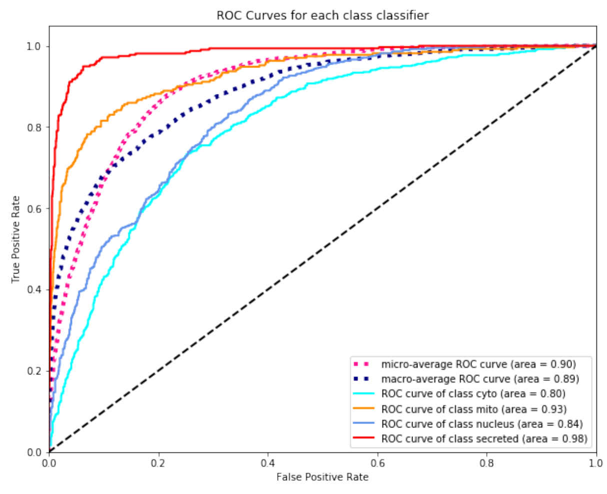

Fig. 1. Returning Operator Characteristic (ROC) curves for each classifier in Random Forests

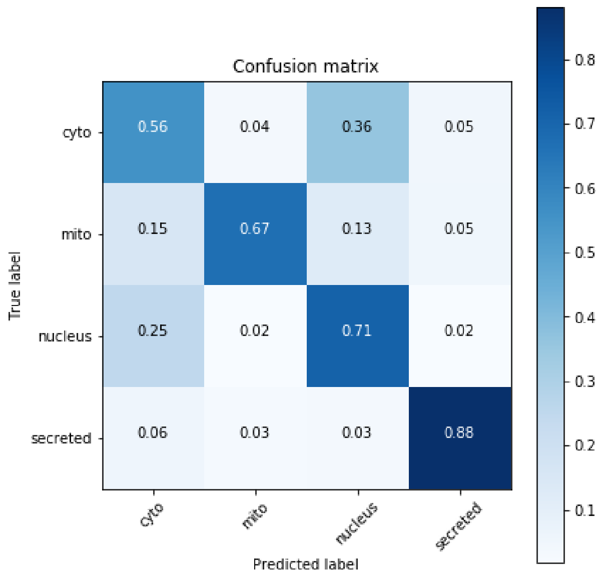

Fig. 2. Confusion Matrix for Random Forest Classifier

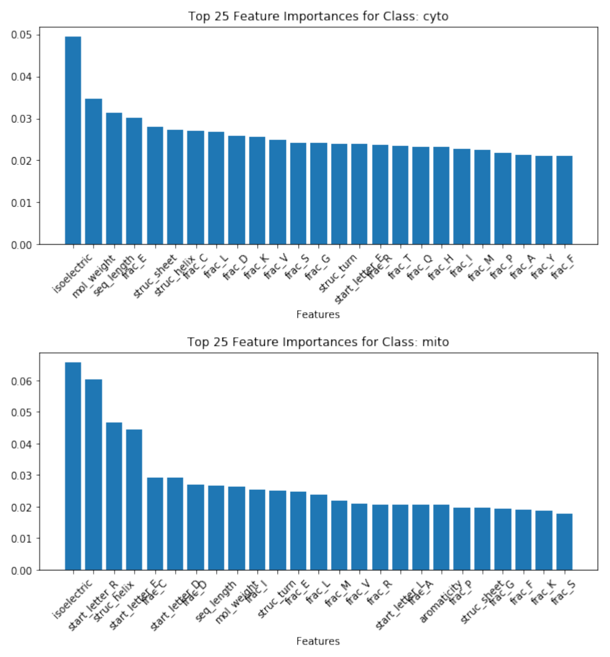

Fig. 3. Top 25 Feature Importances for Cytosol and Mitochondrial classes in Random Forests Classifier

5.2 Error Analysis

Although both the feature-based models performed to different levels on the test set, it is striking that both models found the same classifications difficult. Both models' F1 scores for classes run from lowest to highest in the same order: cytosolic, nuclear, mitochondrial, secreted. This seems to suggest an underlying challenge with distinguishing these former classes, either due to true biological similarity, or simply lack of features.

This is also supported by examining the confusion matrix for the Random Forests classifier (Figure 2). It appears that the greatest single sources of error come from misclassifying nuclear labels for cytosolic

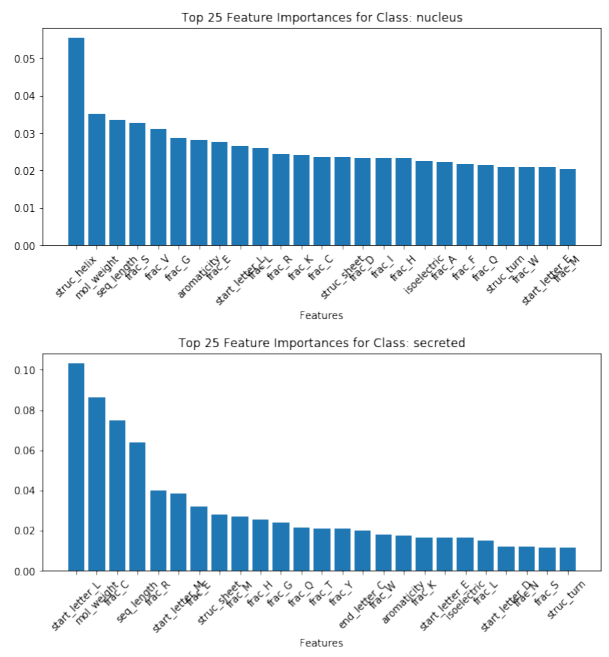

Fig. 4. Top 25 Feature Importances for Nucleus and Secreted classes in Random Forests Classifier

ones, and vice versa. Almost 25% of nuclear proteins are misclassified as cytosolic, and 36% as cytosolic as nuclear.

This misclassification is made all the more evident by looking at the Returning Order Characteristic (ROC) curves, where all the classifiers are plotted together. From these graphs, it is immediately clear that the best classifier is the secreted class, with the area under curve (AUC) of 0.98, followed by the mitochondrial class of 0.93. The worst performer is the cytosolic class (0.80) followed by the nuclear class (0.84), as also visible from the curves.

From a biological perspective, there are also a number of reasons why it is conceivable that these particular protein classifiers (cytosolic and nuclear) should perform worse than the other two categories (secreted and mitochondrial). For one thing, since a number of proteins frequently move between the nucleus and the cytosol, it is possible that many proteins may posses both *Nuclear Export Signals* and *Nuclear Localization Signals*. This mixup of very distinct features (in the form of target peptides) would likely lead the classifiers to struggle to distinguish the correct class in ambiguous cases.

Furthermore, it is likely that mitochondrial proteins and secreted proteins may have important macroscopic properties (aside from target peptides) that still show up in our choice of features. For instance there may exist restrictions on mitochondrial proteins, that need to be perhaps a certain size or make up, help in the respiration process.

5.3 Features Analysis

Perhaps more interesting than the comparison of the two different models is the close comparison of the features that have had a significant impact on the model. By inspecting Random Forest model's feature importances, we have a natural way of interpreting the significance of each features to each classifier.

The first thing to notice is the relative scales of the four graphs, in comparison to one another. The secreted classifier (Fig 4) has a much stronger set of "significant" features, than all the other classes. In the case of comparison to nucleus and cytoplasmic classes, secreted's top feature is over twice the importance of the latter classes' highest feature. Furthermore its top 4 features features all have higher importances than the highest single feature for these classes. This indicates that there are distinct 'killer features' for this class, which explains why both AdaBoost and Random Forests scored particularly highly on this class. From the importances, it appears these features are highly tied to the fraction of leucines and cysteines at the start of the sequence, and the molecular weight and sequence length.

Next to notice is that, as mentioned in the initial background, mitochondrial proteins also have importances of features matching the description of the *mitochondrial targeting signals*. These target peptides sometimes consist of alternating positively charged and hydrophobic amino acids forming a helix, so correspondingly our strongest features include features noting isoelectricity, structural helices, and the presence of positively charged arginine at the start of the sequences.

When it comes to nuclear and cytosolic proteins, it's worth noting that the second and third best importance features - molecular weight and sequence length - are shared. This indicates some of the difficulty these classes have in distinguishing themselves. Furthermore these features are broad macroscopic features that do not really indicate a particular target sequence, also making it harder to classify these proteins. In fact, even the "strongest feature" for each of these classes is a macroscropic feature - isoelectric point, and helix fraction - indicating our set of features haven't been able to pick up the nuclear target peptides.

It is finally worth noting that none of the most important features correspond to peptides at the end of the sequence, only the start. This is fairly good evidence that something important happens at the N-terminus of a protein, and supports background theory that this is where target peptides are often found.

5.4 Further Improvements

Although this model performed well in distinguishing the secreted and mitochondrial proteins there are a number of additional features that may particularly improve future implementations of this model.

First, some measure of hydrophobicity would be a valuable additional feature. As mentioned in section 2, often the hydrophobic characteristic of signal peptides is what is detected by receptors, but this model only captures these features indirectly, through looking at frequencies of individual amino acids. A specific feature for hydrophobicity wasn't added due to the lack of there being an available hydrophobicity scale in the BioPython module, but it is likely some measure of this (both over the whole molucule and just the N-terminus), would be of some value.

Secondly, it might be worth exploring and searching for specific target patterns within the model. When exploring potential features, I attempted to search for these by creating a feature for all possible "bigrams" and "trigrams" of residues, and examining importances after training, but the high dimensionality made learning difficult and slow. Instead, if one had specific examples of known target peptides this feature would likely greatly help classification, without adding too much to computational complexity. A drawback would be that these examples may be very incomplete, as many signals are still unknown, so there may be a limited benefit.

Finally it may be worth exploring the use of neural networks, as classifiers. Neural networks have shown significant promise in various fields in bioinformatics including protein function prediction (eg Fa *et al.* (2018)), and could theoretically be applied to this problem. In this case, one could use a richer set of features and feed through a multiclass deep neural network, or even augment the sequence data with features and the feed augmented sequence into recurrent neural network, such as an LSTM. This latter architecture might have the advantage of allowing investigators to detect new target peptides, through spotting sequences that radically increase likelihoods of a certain classification. However, a drawback is that these neural methods often require a great deal of data, and the dataset here is relatively small so a neural model may not generalise well. This method should be considered, if more training sequence is avaliable.

6 Conclusions

In conclusion, it was clear that the Random Forests algorithm performed best when predicting the subcellular locations of proteins, achieving an F1 score of 0.683, and test accuracy of 67.5%. This beat the AdaBoost classifier, with F1 of 0.626 and test accuracy of 64.3%, although both models far surpassed a random benchmark based on a categorical distribution, with its F1 of 0.273 and accuracy 27.7%.

The outperformance of the Random Forests algorithm, compared to AdaBoost, is likely due to the very noisy data that occurs in this field, and performance was only really comparable when 'killer features' were present for a class. However, both models seemed to struggle with same class classifications, which indicated an underlying difficulty with biological features, rather than the algorithms themselves.

In particular, cytosolic and nuclear proteins were easily mixed up, likely due to the fact that both *Nuclear Export Signals* and *Nuclear Localization Signals* can be present on the same protein, and also are not restricted to only appearing at the N-terminus, making them harder to pick out. Secreted and mitochondrial class classifiers performed much better, and had feature importances that supported the current biological understanding of signal peptides such as *mitochondrial targeting signals*, and their structure.

Finally I concluded that the above experiment, while carried out well (with a hold out test accuracy performing very similarly to the cross-validation accuracy), could be improved by the addition of a number of extra features, including hydrophobicity or counting of known target peptide subsequences. Also in the case of being presented with more labelled data, I suggest neural methods could be used to tackle this problem with some success in future.

References

Alberts, *et al*. (2008) Molecular Biology of the Cell (5th ed.), *Book Title*, 2nd edn. Publisher, Location, Vol. 1, pp. 702, 704-708, 713-14.

Nakai, K. and Horton, P., PSORT: a program for detecting the sorting signals of proteins and predicting their subcellular localization, *Trends Biochem. Sci*, 24(1) 34-35 (1999).

Tung C-H, Chen C-W, Sun H-H, Chu Y-W (2017) Predicting human protein subcellular localization by heterogeneous and comprehensive approaches. PLoS ONE 12(6): e0178832. https://doi.org/10.1371/journal.pone.0178832.

Lobry JR, Gautier C. (1994) Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 Escherichia coli chromosome-encoded genes. *Nucleic Acids Res.* Aug 11;22(15):3174-80.

Bjellqvist, B., Basse, B., Olsen, E. and Celis, J.E. Reference points for comparisons of two-dimensional maps of proteins from different human cell types defined in a pH scale where isoelectric points correlate with polypeptide compositions. *Electrophoresis* 1994, 15, 529-539.

Cock PA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B and de Hoon MJL (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25, 1422-1423

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python *Journal of Machine Learning Research* 12, pp 2825–2830

Rui Fa, Domenico Cozzetto, Cen Wan, David T. Jones, (2018) Predicting Human Protein Function with Multi-task Deep Neural Networks *[Under Review] - https://doi.org/10.1101/256420*

Appendix

Code and Implementation

Repository

All code for implementation and plots can be found at:

https://github.com/condnsdmatters/bioinformatics

Repository Structure

The code repository has the following layout, where *italics* imply a file, and **bold** as directory.

- *investigate.ipynb* - an ipython notebook used to conduct experiments and generate plots. Final results are take from here.
- *features.py* - code used to extract and convert sequence data into feature vectors.
- *categorical.py* - code for implementation of the categorical benchmark, using inheritance from sci-kit learn.
- *aux.py* - code for auxiliary subroutines such as loading files, and turning tuples of data into matrices.
- *run_model.py* - code used to orchestrate the running of experiments, especially in development stage.
- *requirements.txt* - a list of packages to be installed by pip, the python package manager
- **write_up** - the directory for all LaTeXrelated to this report