UCL COMPUTER SCIENCE, MSC MACHINE LEARNING

Bioinformatics: Mini-Project

COMPGI10

Edward Brown: 16100321

March 24, 2017

A method for predicting the sub-cellular location of eukaryotic proteins. It details a Support Vector Machine Classifier (SVC) on features derived purely from the sequence of the protein.

Using 3-fold cross-validation on a labelled dataset of cytosolic, secreted, nuclear and mitochondrial proteins, the method achieves a weighted-average F1-score of 0.68. It goes on to suggest how to improve the accuracy of the method as well alternative methods.

CONTENTS

1 Study Motivation

It is difficult to assess the function of a protein in a cell. However, the sub-cellular location of a protein in a cell can give us clues to its function [2]. Therefore, a model to predict the location of a protein a cell may prove useful in this effort. Although there are more divisions in the cell, the sub-cellular locations could be broadly split into the following:

Cytosolic: Inside the cell itself but not within any organelle.

Secreted: Proteins that are secreted to outside of the cell.

Nuclear: Proteins found inside the cell nucleus.

Mitochondrial: Proteins taken to the cell's mitochondria. [2]

Given the sequences of proteins known to belong to these categories, we can create a model that predicts the location of proteins for which we do not know the location.

2 Methodology

2.1 Model Choice

For the purposes of this report, the Support Vector (Machine) Classifier was chosen due to its simple implementation and lightweight applicability. Its implementation and cross-validation was performed using the SVC package from sklearn. The documentation for which can be found here:

`http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

2.2 SVM Formulation

The method for predicting the sub-cellular location is performed using a Support Vector (Machine) Classifier (SVC). SVCs are normally used in binary classification, so a one-vs-one procedure is used whereby the confidence for each class' prediction is comprised of a comparison of each class versus each other class [1].

SVM classifiers find a hyper-plane than separates classes of data and try to maximise the margin that this hyper-plane separates.

There are an infinite amount of hyperplanes separating the points, the SVM aims to max-

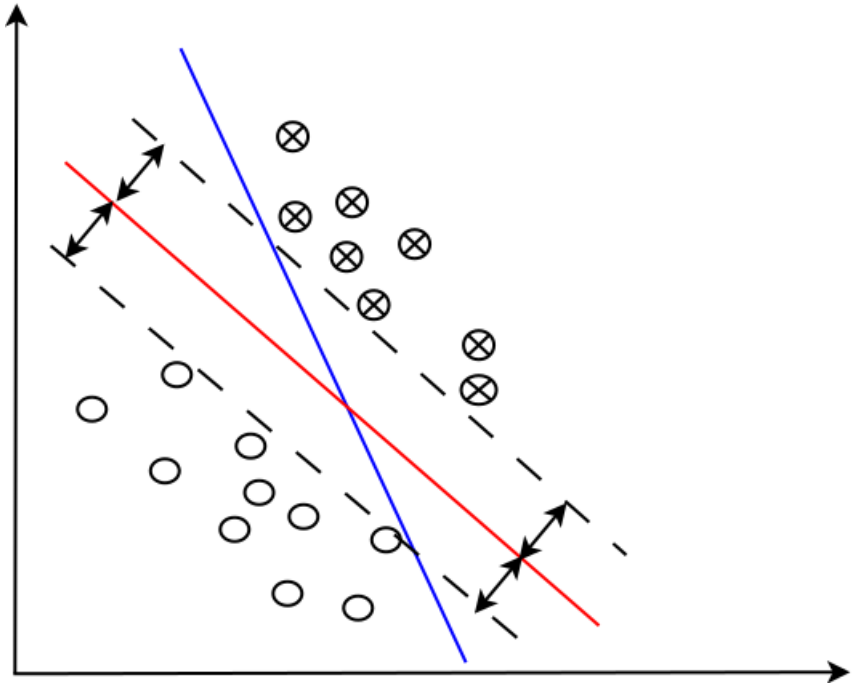

Figure 1: Separating Hyperplanes [7]

imises the width of the margin. Here, the blue line is a separating hyper plane but the red line does a better job at maximising the margin.

After optimising, we find our prediction function as:

$$y_{\text{predict}}(\mathbf{x}) = \text{sgn}(w_0 + \mathbf{w}^T\mathbf{x})$$

where (w_0 is a constant and w is the weight vector [1].

By the Representer Theorem that **w** can be represented by a linear combination of the training data [8].

$$\mathbf{w} = \sum_i^N \alpha_i y_i \mathbf{x}_i$$

Then performing the 'Kernel Trick' [1], we can rewrite the classification as:

$$y_{\text{predict}}(x) = \text{sgn}\big(\alpha_0 + \sum_i^N \alpha_i \kappa(\mathbf{x}_i\mathbf{x})\big)$$

subject $0 \leq \alpha_i \leq C$ and κ is the choice of kernel function. [1]

This dual form of the Support Vector Machine means we can use a kernel that effectively

carries out a feature mapping to radial basis functions, without having to explicitly reform our data [1].

2.3 Feature Choices

The features chosen for the SVC to use for classification were as follows:

Features	**Description**
Sequence Length	Simply the number of components in the sequence.
Global Amino Acid Composition	The Percentage of each amino acid in the whole sequence.
Local Amino Acid Composition (first and last 50)	The Percentage of each amino acid in the first 50 or last 50 of the sequence.
Isoelectric Point	The isoelectric point is the pH at which a particular molecule carries no net electrical charge in the statistical mean.[4] [5]
Molecular Weight	The weight of the protein in terms of hydrogen atoms.

These features are easily extracted using Biopython which can read and analyse protein sequences. The documentation for which can be found here:

`http://biopython.org/wiki/Documentation`

2.4 Experimental Procedure/Model Evaluation

The training data for this model was comprised of a training/test split of 4:1. This split is simply performed by a random shuffle of the dataset followed by the fractional split. The testing data is then held out for final model evaluation.

To optimise the hyper-parameter choice, a 3-fold cross validation method is employed where the weighted f1 score is the metric of performance. The model is trained on 2/3 of the training set and then evaluated on the remaining 1/3, the validation set. The performance (f1 score) of this split is then evaluated. This is repeated holding out each 1/3 of the training set (three iterations). The performance of that particular hyper-parameter choice is then evaluated as a simple average of the f1-score over those three iterations.

Once the optimal hyper-parameter set is arrived at, the model is retrained on the full training-set using those parameters. This trained model is the final model whose performance is evaluated on the test data.

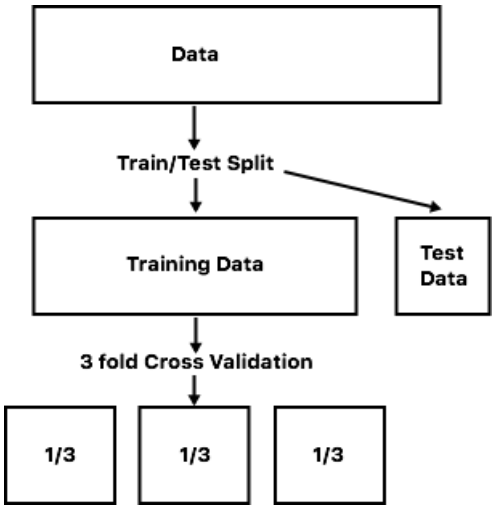

Figure 2: Experimental Procedure

3 Results

3.1 Hyper-parameter Search Results

The hyper parameter search for the SVM's 'C' parameter was searched through the range $[0.7, 0.75, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.20]$ and the kernel choices were [linear,radial basis functions]. The optimal cross-validated f1-score hyper-parameters are summarised below:

C	Kernel
1.1	Radial Basis Function

3.2 Performance Metrics

When this trained model is applied to the held out test set we achieve the following metrics of performance:

Protein	Precision	Recall	F1-score	Accuracy	No. Proteins
Cytosolic	0.58	0.60	0.59	0.59	605
Mitochondrial	0.75	0.65	0.69	0.65	268
Nuclear	0.63	0.65	0.64	0.65	645
Secreted	0.81	0.80	0.81	0.80	327
Weighted Average/Total	0.66	0.66	0.66	0.66	1845

This table shows us that the model performed very well with secreted proteins and the poorest performance was for the cytosolic proteins.

3.3 Confusion Matrix

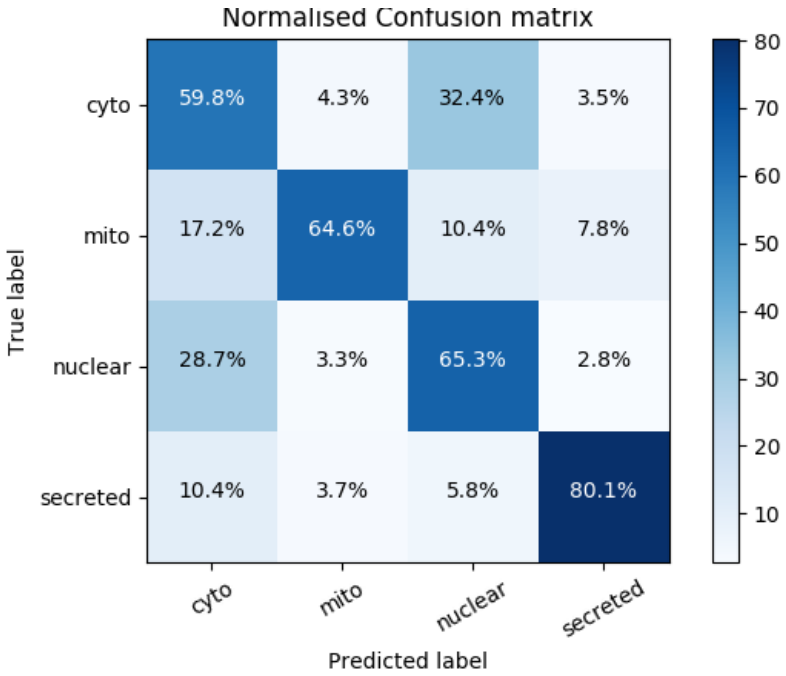

Figure 3: Normalised Confusion Matrix

Figure 3 shows the confusion matrix for the results on the test set. Cytosolic proteins were incorrectly classified as Nuclear proteins 32.4% of the time, while Nuclear proteins were misclassified as as Cytosolic Proteins 28.7% of the time. These were responsible for the largest loss in accuracy across the whole model's performance.

4 Blind Test

The trained model is applied to a dataset of unknown location, for which their sub-cellular location is predicted with the confidence of that predictions accuracy.

Sequence	Location	Confidence
SEQ677	Secr	38.3%
SEQ231	Secr	83.5%
SEQ871	Secr	79.6%
SEQ388	Cyto	47.0%
SEQ122	Cyto	47.4%
SEQ758	Nucl	69.3%
SEQ333	Cyto	56.9%
SEQ937	Cyto	64.9%
SEQ351	Cyto	46.9%
SEQ202	Cyto	41.1%
SEQ608	Mito	61.2%
SEQ402	Nucl	53.6%
SEQ433	Secr	94.5%
SEQ821	Secr	81.6%
SEQ322	Nucl	75.7%
SEQ982	Nucl	87.1%
SEQ951	Nucl	57.9%
SEQ173	Cyto	50.2%
SEQ862	Cyto	52.8%
SEQ224	Cyto	60.2%

5 Conclusions

The model performed reasonably, although there was notable confusion between Cytosolic and Mitochondrial proteins. This is perhaps indicative of a lack of distinguishing features. In order to improve the accuracy of the model, it would perhaps be prudent to add more features to the data. Currently, we somewhat arbitrarily only use the first 50 and last 50 percentages as a measure of local amino acid percentages. Obviously, we could add any number instead of the first and last 50, up to simply the global percentages. Indeed further features such as the aromaticity [3] could have been used.

It naturally occurs at this point that there might be algorithms which would exploit this. Recurrent neural networks for example are perfect for sequence classification. Stacked LSTM and GRU networks could be a candidate for increased accuracy [6].

In terms of adding more features, there are many potential features that could be used for classification. For example, aromaticity [3] [5] or flexibility.

REFERENCES

[1] Murphy, Kevin P., *Machine Learning: A Probabilistic Perspective.*, Cambridge, Mass.: MIT, 2013.

[2] Jones, D., *COMPGI10 Coursework* UCL, `http://www0.cs.ucl.ac.uk/staff/D.Jones/coursework/`

[3] Vihinen, *Accuracy of protein flexibility predictions*, 1994

[4] Bjellqvist, B.; Hughes, G. J.; Pasquali, C.; Paquet, N.; Ravier, F.; Sanchez, J. C.; Frutiger, S.; Hochstrasser, D. (1993-10-01).*Electrophoresis*, 1993.

[5] *Isoelectric Point*, `https://en.wikipedia.org/wiki/Isoelectric_point`

[6] Recurrent Neural Network Tutorial, Part 4 — Implementing a GRU/LSTM RNN with Python and Theano — WildML

[7] Brown E, *Numerical Optimisation, Assignment 2*, 2017

[8] SchÃűlkopf, Bernhard; Herbrich, Ralf; Smola, Alex J. (2001). *A Generalized Representer Theorem*. Computational Learning Theory. Lecture Notes in Computer Science. 2111: 416âĂŞ426. doi:10.1007/3-540-44581-1_27. ISBN 978-3-540-42343-0.