Weighted Protein-Protein Interaction Networks and Community Detection

Gavin Gray

Master of Science by Research

Neuroinformatics

DTC in Neuroinformatics and Computational Neuroscience

School of Informatics

University of Edinburgh

2014

Abstract

Acknowledgements

Firstly, I would like to acknowledge Colin Mclean as the main supervisor to this project and provided extensive input to both the implementation, planning and execution of all parts of the project. Finlay Maguire provided Biological expertise during this project, evaluating the theoretical basis behind many features and checking some of the code. As with the nature of this project, there were many databases and public resources that were used. These are described throughout the report and referenced appropriately. The project also made use of the Scikit-learn(**pedregosa_scikit-learn:_2011**) package throughout. Finally, this report was written using code developed by Chris Brown, Alex Shearn and Danilo Orlando.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Gavin Gray*)

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

It is estimated that disorders of the brain cost Europe €798 billion Euros in 2010(**olesen__economic__2012**). Specifically, depression is estimated to cost Europe €91.9 billion in 2004 and schizophrenia along with associated psychotic disorders is estimated at €93.9 billion. There are many diseases with currently limited treatment options in the brain, and a deeper understanding of the brain is required to treat them.

These diseases are very likely to act at the synaptic level(**chua__architecture__2010**; **synsys**). However, the exact proteins or genes involved in these diseases are unknown. If it is possible to even gain slightly more information about associated proteins at the synapse it may be possible to develop new treatments(**li__interaction__2010**).

In addition, as a project this includes work from various areas, such as Bioinformatics, Machine Learning and Software Development that made it a good learning exercise. It is a worthwhile investment in the fundamentals of constructing PPI networks.

1.2 Outline

This project involved combining both direct and indirect data sources to create weighted edges in a PPI graph to affect the communities detected by a Community Detection algorithm. A flow chart describing this process and the elements involved is shown in figure 1.1. There are three main input sources shown,

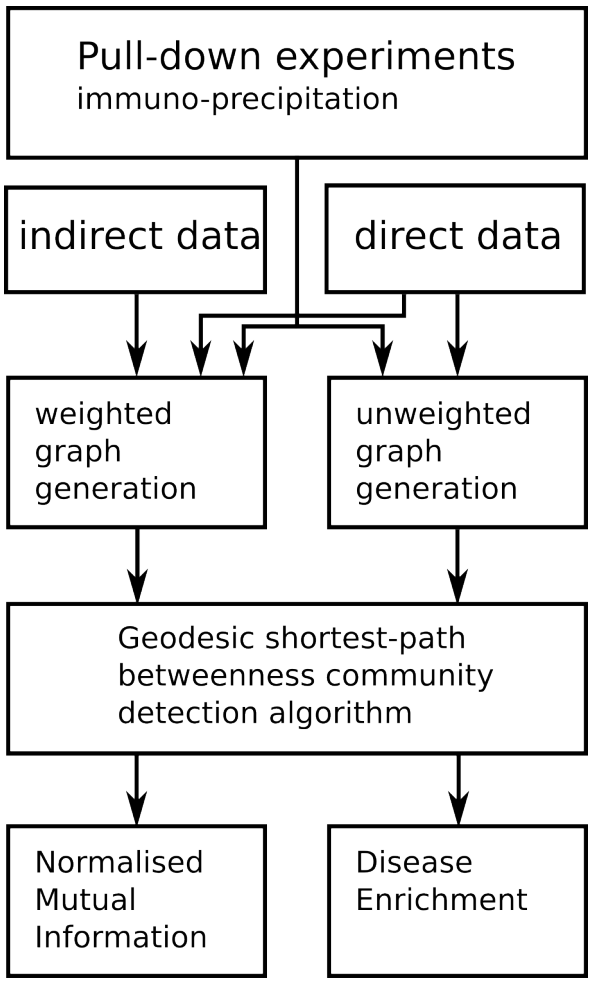


Figure 1.1: A flow chart describing how the elements of the project as a whole, as shown in the project proposal.

the pull-down experiments, direct data and indirect data. Pull-down experiments were used to identify the proteins of interest, which were used to build an interaction network using direct data sources, such as interaction databases. The indirect data sources, such as functional annotations of proteins, were used along with direct data sources in the process to generate weighted graphs.

The resulting weighted and unweighted PPI graphs were then separated into communities using a Community Detection algorithm. These communities could then be compared using the two methods named in figure 1.1: Disease Enrichment and NMI. The results of these tested the hypothesis of the project.

The hypothesis of this project was that the communities detected in each case would differ and that these differences would be relevant to the conclusions each network could illustrate about a particular disease. NMI was able to show simply

that the two sets of communities detected was different. Disease enrichment refers to testing the likelihood that a given community is involved in a particular disease. This allowed us to show the differences in the predictions for disease involvement given by each network due to the weighting of the networks.

The report first describes the background knowledge driving the hypothesis of the project in chapter 2. In chapter 3 the methods involved in executing the project are described. Finally, the results obtained are described in chapter 4. In addition, the execution of all commands required to reproduce the project were recorded in notebooks described in section A. This project was stored in a git repository and is publicly available in a repository described in Appendix .1.

Conclusion

The following report covers the background information necessary to understand the methods of the project and the results which were obtained. It was approached as an opportunity to use a variety of new tools, and each will be described as the report continues.

Chapter 2

Background

This project involved the use of protein interaction prediction to build weighted PPI networks improve the performance of a Community Detection algorithm on a PPI network. The aim of the Community Detection algorithm was to gain insight into structure of the interactions between proteins at the synapse to aid disease research. In the following chapter these different components, and how they fit together, will be described.

2.1 The synapse and protein interaction

Cells consist largely of proteins. Each of these proteins are carefully tuned molecules which fit into machinery of a cell within the human body. Functions of these cells include almost all cellular functions; there are proteins capable of pumping ions, reshaping DNA and fluorescing(**alberts_molecular_2008**). A crude model of the cell is to map the interactions between these molecular machines to try to guess about the functioning of the cell. These models are protein-protein interaction (PPI) networks and can be useful for disease research.

The proteins at the synapse drive synaptic communication, which in turn defines the functioning of the brain. As these proteins define the functioning of the brain any disorders which affect the brain are very likely to involve these proteins. Disorders which affect the brain are also very common and poorly understood, affecting one in three people in the developed world. Curing these diseases therefore may be possible through a greater understanding of the interactions of proteins at the synaptic level(**synsys**; **chua_architecture_2010**).

Synapses are the contacts between nerve cells where the vast majority of communication between nerve cells occurs, the only exceptions being through signalling molecules that can cross the cell membrane. There are two types of synapses in the nervous system, electrical and chemical(**kandel_principles_2000**). Electrical synapses form a simple electrical connection through an ionic substrate between two neurons. Chemical synapses are involved in a much more complex system of neurotransmitter release and reception.

Synapses are therefore important to the functioning of the nervous system. A problem with synapse function will likely cause large problems to the nervous system, so diseases of the nervous system are likely to involve problems with synapse function. As the cell is composed of proteins, so is the synapse composed of proteins. Investigating the functioning of these proteins will help to explain the functioning of the synapse and hopefully provide insight into the diseases of the synapse.

Physical interaction between proteins can be inferred from a range of different experiments. Typical contemporary protein interaction networks rely on databases of confirmed interactions from a variety of experiments, for example in **kenley_detecting_2011** several well-known interaction databases were used. By forming a network from these individual interactions as edges and clustering this network the example paper was able to predict complexes and functional associations. If these functional associations are involved in disease it is possible to associate proteins with diseases, as will be shown in section 3.

Originally, two papers, **ito_comprehensive_2001** and **uetz_comprehensive_2000** were able to leverage large volumes of recent interaction data and build interaction networks. These papers were able to make interesting discoveries about the network of interactions in yeast simply by investigating subnetworks in the network that was produced.

The interaction network we are investigating in this work is referred to throughout as the active zone network in the synapse. These proteins are part of the pre-synapse and are illustrated in figure 2.1. Proteins identified as part of this network were used as baits in the pull-down experiments whose results are used in this project to build the PPI network which is the focus of the weighted and unweighted Community Detection.

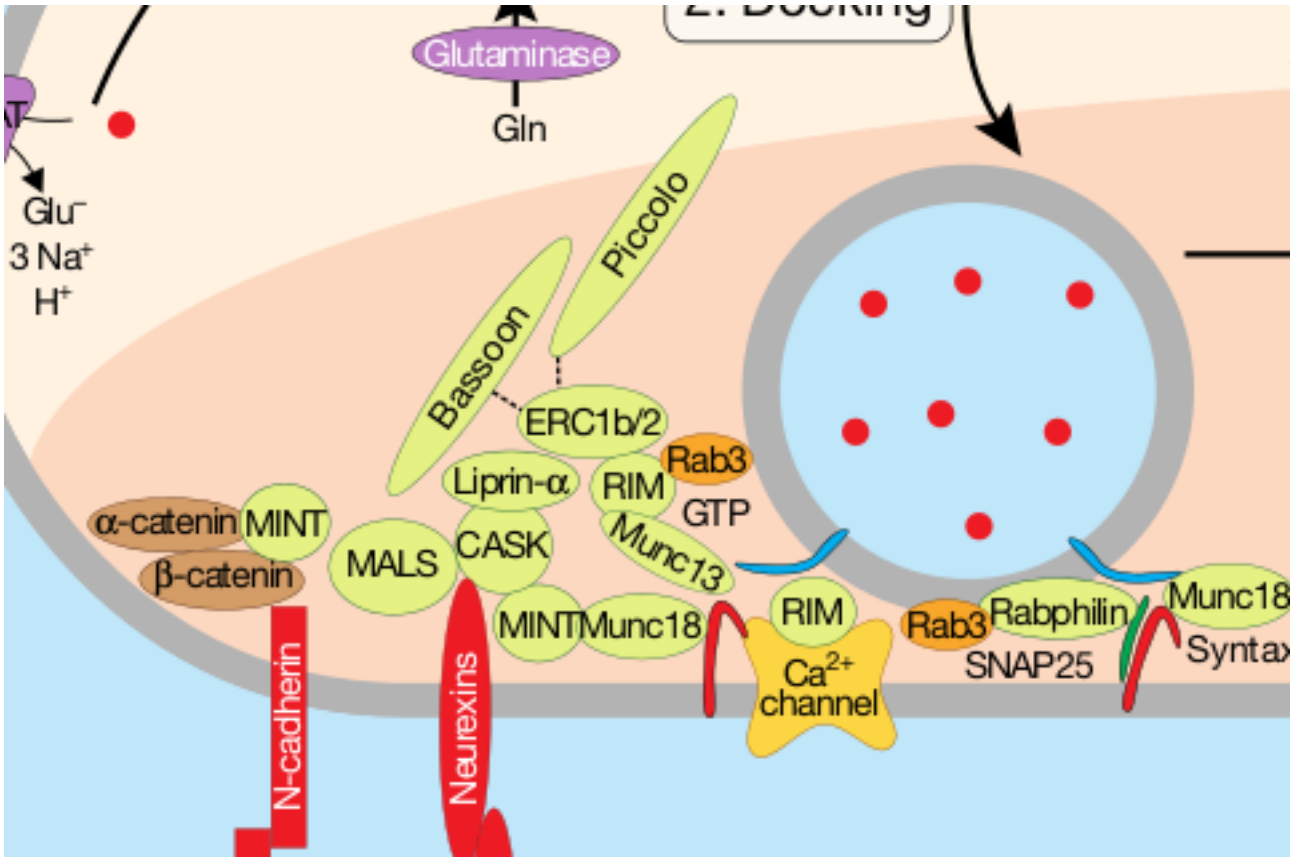


Figure 2.1: An illustration of the proteins identified to be involved in the active zone network(**chua_architecture_2010**).

2.2 Protein complexes and community detection

As mentioned in the previous section it is possible to analyse PPI networks to detect protein complexes and functional groups. This has recently been achieved through use of Community Detection(**chen__identifying__2013**; **wang__recent__2010**), which uses various methods to find community structure in graphs.

Community structure is described as a characteristic of graphs which have many connections within sub-groups but few connections outside that group(**newman__commu**). Unfortunately, this description is not specific on exact measures for a graph to have community structure. Community detection algorithms are simply tested on graphs that are agreed to exhibit community structure with the aim of finding the pre-defined communities.

There are two main approaches to the problem of Community Detection: traditional hierarchical methods and more recent optimization based methods(**newman__commu**). Hierarchical methods were developed in the field of sociology and involves grading nodes by how highly connected they are in the network and then using this value to group nodes into communities. Optimization based methods involves a different measure known as betweenness, which is analogous to the current flowing along edges if the graph were an electric circuit, and then allows a reductive technique where edges are removed iteratively to reveal sub-graphs without connections between them.

2.3 Protein-protein interaction prediction

Protein interaction prediction was developed to solve the problem of incomplete and unreliable interaction data by combining both direct and indirect information(**qi__learning__2008**). Direct information are the result of experiments, such as yeast two-hybrid, intended to directly find protein-protein interactions. Indirect information includes biological data that was not gathered directly to find interactions, such as gene expression data. More information on the data sources can be found in the following section 2.4.

To predict a protein interaction we need to have a value or sequence of values from which to make our guess as to the existence of an interaction. For each interaction this set of values are known as features. The bulk of the work in this project involved obtaining these values for every feature necessary to train the

Data source type	Examples
Primary interaction databases	DIP(**xenarios__dip__2002**)
	HIPPIE(**schaefer__hippie:__2012**)
	BioGRID(**stark__biogrid:__2006**)
Associated features	Features derived from Gene Ontology(**ashburner__gene__**
	Those used in **rodgers-melnick__predicting__2013**
Other PPI prediction resources	STRING(**vonMering__string__2005**)
	InterologWalk(**gallone__bio::homology::interologwalk__**

Table 2.1: A table summarising the different sources of data used in the course of the project.

classifier and classify the interactions of the synaptic network.

The classifier is a machine learning algorithm that can learn from a labelled training set how to sort these vectors of features into the appropriate category. However, these algorithms cannot make predictions unless the training data is informative. Also, the training data must be an accurate representation of the case the algorithm is planned to be applied to.

Completing the interactome of a given organism from incomplete data is a major goal for some works in the protein interaction field, such as **rodgers-melnick__predicting__** The goal in this project is to appropriately weight interactions in a PPI network to improve the performance of a Community Detection algorithm.

Weakly interacting proteins will have a lower confidence in their interacting at all, as it will have been observed less frequently. Therefore, by weighting the interactions in a PPI network according to our confidence we can also make the PPI network reflect more closely the true situation.

2.4 Data sources and networks

Many different data sources were considered for inclusion in this project. The full list can be found in Appendix A.4. These different data sources fall into categories described in table 2.1.

The indirect sources of data were chosen based on usage in the literature, such as in the case of Gene Ontology(**qi__evaluation__2006**). Direct data sources were listed by investigating all of the available databases which could be of use

and choosing from these.

Conclusion

The goal of this project involved obtaining weights for a PPI network correlated with the strength of different protein interactions to improve the performance of a Community Detection algorithm. Improving the performance in this way, it was hoped would produce new insight into protein interactions that could cause disease.

Chapter 3

Methodology

Feature extraction was the predominant component in this project and involved turning the various data sources involved into usable features in a machine learning framework. This allowed the Classifier to be trained in a traditional way. The chosen Community Detection method, a Spectral Modularity algorithm, is also described.

3.1 Feature extraction

Features are the processed form of the data we use to make predictions about a given interaction between pairs of proteins and is defined mathematically in the following text. The process of taking a retrieving these from biological databases is referred to as feature extraction. This commonly involved mapping between protein identifiers and indexing large tables automatically, but also involved many other small pieces of scripting.

In supervised learning problems we wish to learn a mapping between input variables and output variables given a training set. Defining this training set rigorously, it consists of input variables $\boldsymbol{x}$ which are typically vectors of values known as features. The output variables in a classification problems are a set of labels(**murphy_machine_2012**): y. In the case of binary classification these are simply either 0 or 1. Therefore, given N training vectors $\boldsymbol{x}_i$ and training labels y_i we can define our training set $\mathcal{D}$ for N data points as:

$$\mathcal{D} = ((\boldsymbol{x}_i, y_i))_{i=1}^{N} \tag{3.1}$$

Our problem involves taking various types of biological data, such as entries from biological databases indicating that proteins are involved in the same part of a cell and using these as features. The training labels are either an interaction (a one) or a non-interaction (a zero). Interactions are taken to be any interactions in the iRefIndex(**razick_irefindex_2008**) database with over 50% confidence. Non-interactions are random binary combinations of Entrez protein IDs, which is a method applied in other works(**qi_evaluation_2006**) to create negative training examples. These are also checked against the iRefIndex database to ensure they are not accidentally known interactions. There are approximately 600 non-interactions for every true interaction.

What we would like to estimate is the posterior probability of an interaction existing given a new feature vector after training our classifier. For any model $\mathcal{H}$ and a new feature vector $\boldsymbol{x}^*$ we can express this using Bayes theorem:

$$p(y^* = 1|\boldsymbol{x}^*, \mathcal{D}, \mathcal{H}) = \frac{p(\boldsymbol{x}^*|y^* = 1, \mathcal{D}, \mathcal{H})p(y^* = 1|\mathcal{D}, \mathcal{H})}{\sum y^* p(\boldsymbol{x}^*|y^*, \mathcal{D}, \mathcal{H})} \tag{3.2}$$

The posterior probability is $p(y^* = 1|\boldsymbol{x}^*, \mathcal{D}, \mathcal{H})$. The likelihood is $p(\boldsymbol{x}^*|y^* = 1, \mathcal{D}, \mathcal{H})$ The prior is $p(y^* = 1|\mathcal{D}, \mathcal{H})$. The marginal likelihood is $\sum y^* p(\boldsymbol{x}^*|y^*, \mathcal{D}, \mathcal{H})$. Where y^* and $\boldsymbol{x}^*$ are a new label and feature vector, respectively.

We explicitly apply a prior to the probability of interaction based on the expected ratio of interactions to non-interactions stated in (**qi_evaluation_2006**) of 1/600. This will be described in more detail in section 3.2.3.

3.1.1 Protein identifier mapping

Mapping from one protein identifier to another became a significant problem in this project. Unfortunately, most Biological databases maintain their own indexing method to identify different genes and proteins. New data sources being integrated into this project would often be using a different identification scheme to the NCBI Entrez GeneID originally chosen to use in the PPI network.

Genes are defined by their amino acid sequence, but this is a long series of letters and the number of genes is much smaller than the possible combinations of these letters. For the sake of posterity databases containing information about genes typically apply an identifier for each gene that is much shorter and can encode other information about the gene. The Entrez GeneID identifier is relat-

ively simple, just consisting of a number generated when the gene was added to the database(**maglott_entrez_2007**).

Other popular schemes include the Ensembl identifier from the Ensembl database(**ensembl_**), Uniprot identifiers from the Uniprot database(**uniprot_website**) and even those used only for specific databases such as DIP identifiers(**dip_website**). Mapping between these different identifiers is difficult as each identifier may map to none or many in another database. The reason this happens is due to isoforms of different proteins; different amino acid sequences can code for a protein with the same name.

Various tools exist to map from one protein identifier to another: Ensembl's BioMart(**smedley_biomart_2009**) is a versatile web-based tool, for example. In this project, simple conversion tables from NCBI's Gene(**maglott_entrez_2007**) ftp server were primarily used. Another tool used was the Uniprot(**consortium_universal_2**) online service web service.

Unfortunately, using any of these services there will be a number of IDs which cannot be converted and many IDs mapping to the same Entrez ID as different protein isoforms are picked up. One way to avoid this problem is to only refer to a single canonical form of any given protein and find this protein in other databases through its amino acid sequence. This ensures that when referring to an interaction between two identifiers the interaction is always simply between two proteins.

Otherwise, as in this project, the interaction is detected between two Entrez IDs; which corresponds to an interaction between genes - possibly only a single interaction between combinations of the isoforms of each gene. Unfortunately, this means that this project is only concerned with gene interaction prediction until the Entrez IDs have been carefully canonicalized. This is not really a problem, as we are only aiming to provide a weighting to a graph, rather than provide an accurate prediction of interaction between proteins.

A solution to this problem is provided by the iRefIndex(**razick_irefindex:_2008**) database, which combines many databases and stores canonicalized entries. Using this database, it would be possible to ensure that the proteins used in a future project would be reliable canonical proteins. Additionally, each protein of interest should ideally be stored with reference to its sequence in, for example, FASTA format.

3.1.2 Dedicated code: ocbio.extract

To keep track of the various different data sources and assemble the features into vectors to be used in the classifiers a dedicated piece of code was required. The code developed is written as a python module called ocbio.extract. Usage and development notes for this program are referenced in Appendix A.3.

3.1.3 Gold standard datasets

The database of interacting proteins (DIP) is a database of interactions proven by small-scale experiments(**xenarios__dip__2002**). Each interaction added is hand curated so it was expected as a reliable training set. Also, this database was used as a training set in **qi__evaluation__2006**

Unfortunately, problems were found with DIP as a training set. Features derived from interaction databases would win out in importance versus all indirect features as shown in figure 3.1. HIPPIE was also tested as a training set, but this lead to the same problem.

It was decided that only indirect features should be used in the trained supervised classifier and direct evidence integrated into the final weightings in an explicit Bayesian method described in section 3.2.3; the results of which are described in section 4.2.5. Once these databases were removed the performance of the classifier was drastically lower. However, all of the available features had more closely distributed importances in the final classifier, as shown in section 4.2.4.

To train the final classifier the iRefIndex(**razick__irefindex:__2008**) database was used to find positive interactions due to its effective protein identifier mapping ability, as described above. Again, non-interactions were generated as random combinations of Entrez Gene IDs. These were also checked against the positive interactions to ensure known interactions were not present.

3.1.4 PPI prediction features

PPI prediction features are a set of values for each interaction considered, if it is a real or non-interaction. This arrangement is illustrated in figure 3.2. The features used are described by index in table 3.1.

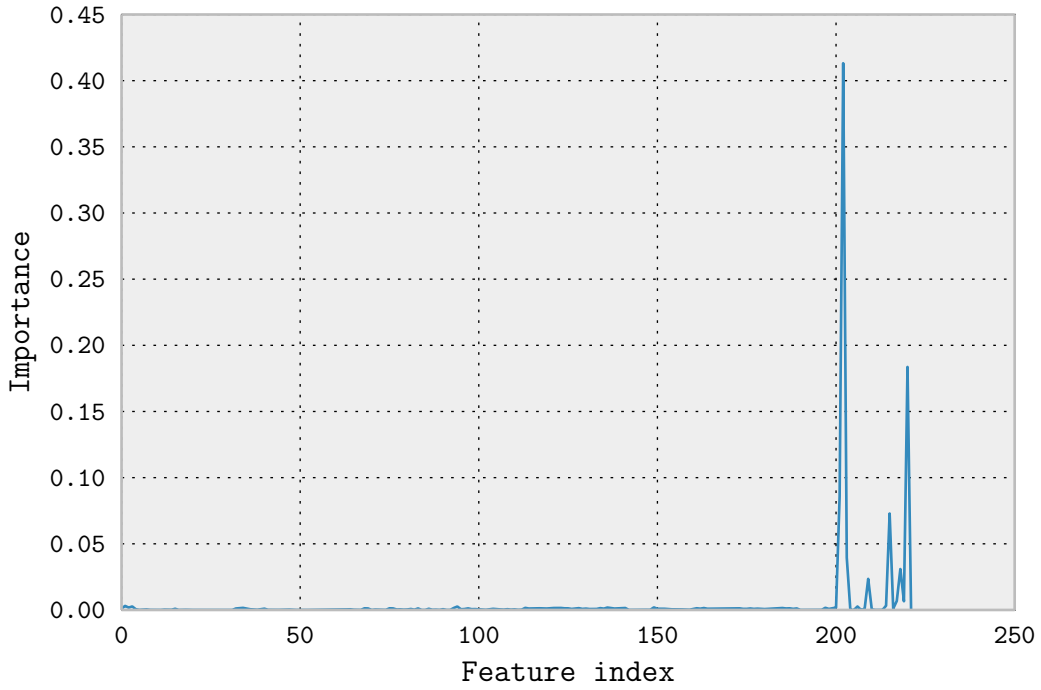


Figure 3.1: An example of an unbalanced set of feature importances plotted after fitting a Random Forest classifier to a dataset containing interaction database derived features.

Feature	Indices	Descript
Gene Ontology	1-90	Described in section 4.1.1, with individual
Y2H	91	Y2H experimentally
ENTS derived	92-198	Features used by the ENTS classifier, described by index in supp
ENTS summary	199	Prediction result of the classifier in **rodg**

Table 3.1: Each feature used in the final classifier is described by index.

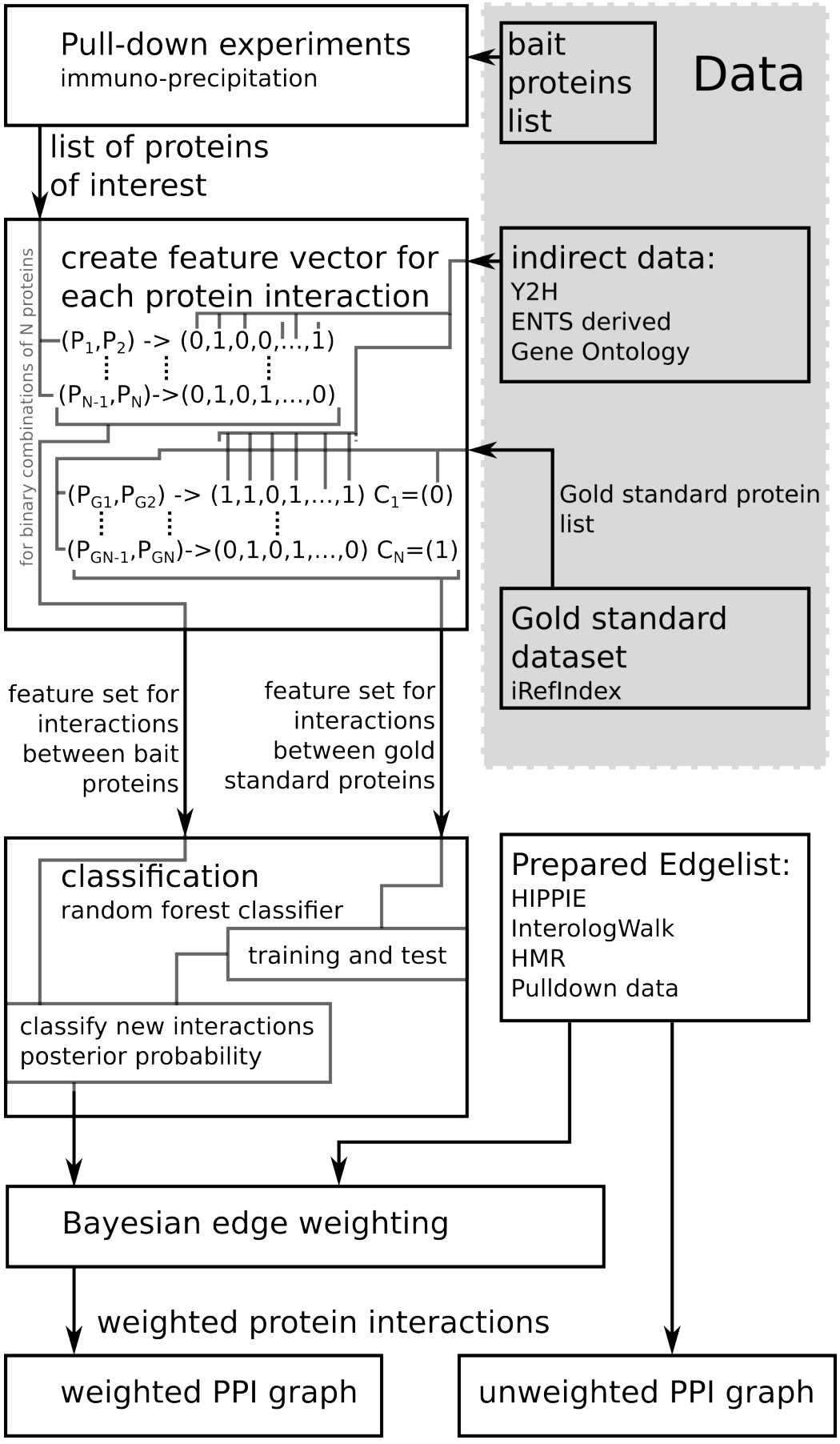


Figure 3.2: A diagram showing the structure of feature vectors and their relationship to the project as a whole. This figure is based on a similar flow chart shown in the project proposal.

Classifier	Hyper-parameters	
Logistic Regression	C	Inv
	kernel	T
Support Vector Machine	Gamma(γ)	
	C	
Random Forest	N estimators	
	Max features	Number of features considered when look
Extremely Randomized Trees	N estimators	
	Max features	

Table 3.2: Summary of the hyper-parameters described in the Scikit-learn**pedregosa_scikit-learn:_2011** documentation to be tuned for each of the models considered.

3.2 Weighting Protein Interactions

The classification problem we are solving is different in that we are really trying to obtain a realistic weighting of interactions for use in a PPI network. Classification is normally concerned about picking a decision threshold to classify examples into categories. However, in this case the output of our process is the posterior probability of the model given a new example.

To produce this output the chosen tool was the Python package Scikit-learn(**pedregosa_sci**). Each classifier implemented in this package has a similar interface allowing modular code to be written. In addition, this package is actively developed with all the required classifiers having efficient implementations.

The following sections describe the three classifiers chosen. These were a logistic regression model, a random forest model and a support vector machine. Each of these involved tuning a number of hyper-parameters.

Hyper-parameters

Each of the Classifiers used has hyper-parameters which will affect its performance. These hyper-parameters are described in table 3.2 for each of the models used in the project. The optimal values for these found can be found in section 4.2.1, table 4.2.

Logistic Regression

Logistic regression is a linear model being used for classification. It is equivalent to a linear regression model transformed through a sigmoid function(**murphy_machine_2012**), denoted here by σ:

$$p(c = 1|\boldsymbol{x}) = \sigma(b + \boldsymbol{x}^T * \boldsymbol{w}) \tag{3.3}$$

In equation 3.3 $\boldsymbol{x}$ is the vector of features and c is the class label - in our case 1 is a real interaction, 0 is a non-interaction. The weights and biases are the parameters of this model, expressed in the above equation as b and $\boldsymbol{w}$, respectively.

This divides the points in the dataset by a hyperplane, classifying the points on each side into different classes. For data that is linearly separable, this produces a classifier that will make no mistakes on the test data. Unfortunately, the data we are working with is not linearly separable as shown in section 4.1.3.

To find the parameters the log likelihood of this model must be maximised; corresponding to the maximum likelihood solution. The log likelihood for this model, for N feature vectors, is:

$$L(\boldsymbol{w}, b) = \sum_{n=1}^{N} c^n \log \sigma(b + \boldsymbol{w}^T \boldsymbol{x}^n) + (1 - c^n) \log(1 - \sigma(b + \boldsymbol{w}^T \boldsymbol{x}^n)) \tag{3.4}$$

Regularisation of the weights represents a prior belief that the weights should not increase without bound. In a case where the data is linearly separable and where regularisation is not applied the weights will increase without bound to produce extremely confident classifications(**barber_bayesian_2013**). To stop this from happening we apply a penalty term, α, to the size of the weights:

$$L'(\boldsymbol{w}, b) = L(\boldsymbol{w}, b) - \alpha \boldsymbol{w}^T \boldsymbol{w} \tag{3.5}$$

Tuning this hyper-parameter is the goal of a grid search when training a Logistic Regression model.

Support Vector Machines

Logistic regression can be generalised to apply kernel functions to the input features to obtain better classifications. Support Vector Machines exploit this while

also applying a different objective function intended to avoid overfitting(**murphy_machine_2**). The objective in placing the hyperplane for a Support Vector Machine is a "maximum margin" in that is attempts to maintain the same distance from the closest opposing class points.

These are often successful classifiers in practice. Applications include text categorisation, hand-written character recognition, image classification and biosequences analysis(**cristianini_introduction_2000**).

The hyper-parameters for a Support Vector Machine control the kernels, along with the regularisation parameter as described for logistic regression in section 3.2. Two hyper-parameters which can be tuned during a grid search operation are the degree of polynomial kernels if chosen and the gamma coefficient of the kernels.

Random Forest

Random forests operate as a combination of many decision trees. Decision trees are intuitively simple in that it consists of a series of comparisons arranged in a tree as shown in figure 3.3.

In this way decision trees are both simple and tractable methods for using a feature vector to classify inputs and there are many automated ways to generate effective decision trees. The problem in the design of a decision tree is which comparison to choose at each node, described as how to partition the data(**murphy_machine_2012**). There are several implementations of algorithms to achieve this and a full description can be found in **murphy_machine_2012** Other advantages of decision trees include the ability to handle a mixture of discrete and continuous inputs, automatic variable selection, scaling to large datasets readily and the ability to handle missing inputs, with modification.

Unfortunately, despite the strengths of decision trees there are still problems with stability: small changes in the input data can produce large changes in the output(**murphy_machine_2012**). The random forest algorithm addresses this problem by providing redundancy; multiple trees are grown and their results averaged. For M trees trained on different subsets of the data(**murphy_machine_2012**):

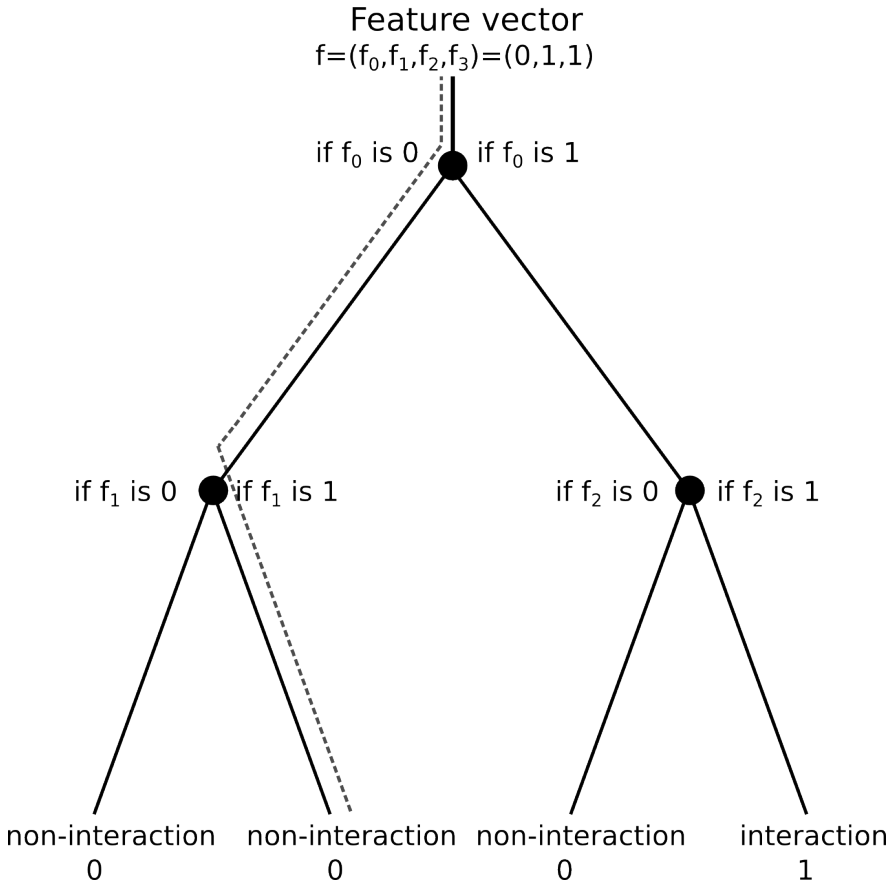


Figure 3.3: A simple example decision tree, illustrating the process of sequential, dependent decisions. Based on an image obtained through Wikimedia(**wkmdacommons**).

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x) \tag{3.6}$$

Where $f_m(x)$ is mth tree. This is simply averaging the results of the trees and is known as bagging.

With the ability to work on large datasets and mixing continuous and discrete data these types of classifiers would appear to already be well suited to this problem. This is what has been observed in the literature, with these classifiers achieving the best performance despite different types of biological data being used(**qi__evaluation__2006**; **rodgers-melnick__predicting__2013**). Due to these reports in the literature it appeared that this classifier would be the best choice for our protein interaction prediction task.

Other options

Other options considered for our classification problem, but not included in the project due to time constraints included Feedforward neural networks, Naive Bayes and Beta regression. Naive Bayes in particular would have required modifying the code from Scikit-learn to deal with data from multiple different distributions or implementing Weka's solution of kernel density estimated distributions for each different feature(**john__estimating__1995**). Beta regression would have been very suitable for the task and is suggested as future work, described in the following section.

Beta regression

Beta regression is a generalised linear model is one in which the output variable is distributed according to a Beta distribution(**smithson__better__2006**). A maximum-likelihood solution to this model can be found and the resulting model fit in the same way as a standard regression model.

True protein interactions are not something which can be assumed in a protein interaction prediction task. Many proteins are known to be very likely, but others are less confidently classified, as reflected in the HIPPIE database's confidence scoring system(**schaefer__hippie:__2012**). To train a classifier a training set of true and false interactions is required and this cannot be supplied without thresholding the database at an arbitrary confidence value.

Beta regression avoids this problem as it allows the confidence values to remain a part of the regression process. Using this we can build a model and update our belief on the likelihood of interaction in a Bayesian framework much more easily.

3.2.1 Classifier testing

Various measures were applied to each classifier to estimate their performance in different ways. These tests included simple accuracy measures applied over learning curve and grid searches along with plotting ROC and precision-recall curves. Grid searches of parameter values were used to find optimal hyper-parameters.

Cross-validation was applied to tests during parameters searches and when plotting learning curves to get a statistical estimate of the reliability of the metrics applied to the classifier(**witten_data_2011**). As the training set is relatively large, these were applied as random sub-samples of the full data set, but stratified to maintain the same proportion of zeros to ones as in the full training set. This is important as it must reflect the expected ratio for real protein interactions to non-interactions.

Pipeline

A pipeline is a combination of algorithms intended to run on the data in sequence after the data has been split into training and test. In the case of this project the pipeline involved three components: a mean value filling imputer, a standard scaler and the classifier itself. The imputer simply replaced missing values in the training data with the corresponding mean value for that column. Scikit-learn's standard scaler centers the data at zero mean and unit variance.

Learning Curves

Learning curves, as in the notebook referenced in Appendix A.2, were used in this project to ensure that the number of samples used in a grid search was sufficient. The learning curve plots the accuracy of a classifier after it has been trained using cross-validation on a varying number of samples.

Grid search

A grid search is a cross-validated test measuring accuracy testing the classifier with a variety of different hyper-parameters. Classifiers, such as a Logistic Re-

gression model, have some number of hyper-parameters. A Logistic Regression model, for example, has a single hyper-parameter so that the grid search for this model simply involves varying this parameter to obtain the optimum performance on the test set.

Accuracy

The accuracy value plotted in the learning curve and used in grid searches of parameters values is simply the proportional of correctly classified instances in a training or validation set. Typically, the protein interaction prediction problem is sparse, in that there are very few interactions for the large number of non-interactions. This is reflected in the accuracy in that a classifier that simply always predicts zero can still achieve a very high accuracy. Using this accuracy value is therefore problematic and this requires the other measures employed, such as ROC and precision-recall AUC values.

ROC curve

A Receiver Operating Characteristic, or ROC, curve plots the variation in true positive to false positive rate as the threshold of classification is varied. This makes it useful as an illustration of the tradeoff possible with this particular classifier. The Area Under Curve, or AUC, value is the area under this line. A higher AUC value corresponds to a better classifier, although there is some controversy surrounding this(**hanczar_small-sample_2010**). These concerns center on the problems of small sample sizes, which are not the case in this project.

Precision-recall curve

A precision-recall curve plots the precision of a classifier against its recall as the threshold of classification is varied. The precision is defined as the number of true positive results over the number of total positive results. Recall is defined as the number of true positive results against the number of available positive examples The area under the curve is used to gauge the classifier's effectiveness.

3.2.2 Missing data

Before classification could be performed the missing data in the feature vectors had to be imputed. This was performed by filling the missing values with the mean value of that feature. This is a common technique that is applied if it is likely the data is missing at random. In this case the data that is missing is due to mismatches in protein identifier mapping dictionaries, which is likely random and independent of the interaction prediction task.

3.2.3 Bayesian weighting

Weighting interactions using the posterior distribution of a probabilistic model requires that the model accurately represents beliefs about the system being modelled. Supervised classification requires that true interactions are known in order to find the parameters of the model. Unfortunately, in this problem we cannot know for certain whether an interaction is real or false. Therefore, when fitting a supervised model we only obtain, at best, an accurate predictor for the interaction database used to fit the model.

As the PPI network edges we plan to weight have already been defined through combining different interaction databases we already have a strong prior on the existence of these interactions. Using the classifier on its own can then, at best, reproduce one of the databases used to create this network and many of the interactions will be incorrectly weighted much lower than expected. The solution is to treat both the result of the classifier and the edgelist inclusion as observable events dependent on the latent "interaction" variable. Along with these variables, we can also include other protein interaction prediction resources, such as STRING(**von_mering_string:_2005**).

The model we have chosen to update our prior belief using assumes conditional independence of the observable variables given the hidden interaction variable. This equates to a Naive Bayes model with a belief network as shown in figure 3.4. An advantage of this model is the simplicity of its analysis. The disadvantage is that our variables could be dependent, as the classifier is trained on some of the same interaction databases that the HIPPIE database is composed of.

The class label, interaction, is not observed in this model so we cannot solve to find the parameters. To define the model, the only way to proceed is to manually define them by conservatively estimating the conditional true positive and false

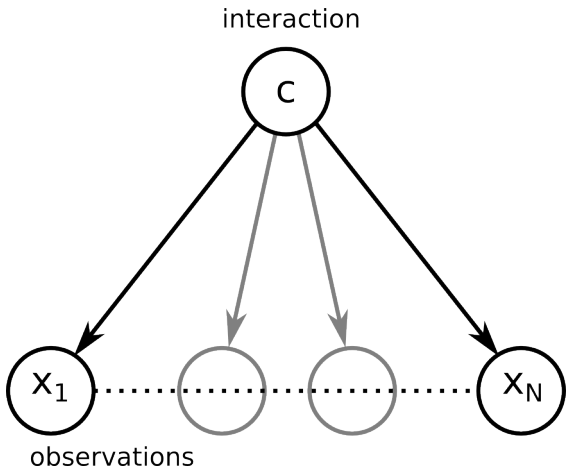


Figure 3.4: A belief network illustrating the Naive Bayes model, equivalent to that used for inference when weighting the interactions of the PPI network.

positive rates of the Bernoulli distributions. Continuous distributions, such as the classifier predictions or prediction databases, must be estimated using KDE.

Using KDE in conjunction with Naive Bayes is the approach used by Weka to deal with arbitrary conditional distributions(**john_estimating_1995**). Unfortunately, this distribution is estimated using samples labeled using the protein interaction databases, so we cannot be fully confident in its predictions. As before, a conservative estimate of its accuracy is applied through increasing the smoothing bandwidth.

3.3 Measures applied to weighted and unweighted PPI networks

It is hoped that a weighted graph will provide new insight into the interactions of proteins in the active zone. For this reason, comparing the unweighted and weighted cases of the graph produced is a major goal of this project. The following sections describe this process and the measures applied to both graphs.

3.3.1 Community detection

Three algorithms were identified for use in this project for community detection, the first two are described in **newman_finding_2004** Geodesic edge betweenness and random edge betweenness are based on betweenness measures

to partition the graph, making them optimisation approaches. The third, spectral modularity, was the chosen algorithm in this project.

The original advantages of spectral modularity techniques were better results in less time than competing methods in **newman_modularity_2006** In a recent paper, this method was compared favourably to other methods in terms of CPU time(**mcleanunpub**). Although it detected fewer communities, it maintained a higher modularity score.

3.3.2 Normalised Mutual Information

Mutual information intuitively is the reduction in uncertainty about one random variable by observing another. Defined in terms of entropy it is(**mackay_information_2003**):

$$I(X;Y) = H(X) - H(X|Y) \tag{3.7}$$

Where $H(X)$ is the entropy of the random variable X and $H(X|Y)$ is the conditional entropy of X given Y.

In the case of the function we are using to perform this from Scikit-learn the mutual information is normalised by $\sqrt{H(X) \times H(Y)}$(**pedregosa_scikit-learn:_2011**). This produces a value between zero and one which reflects the redundancy of the distributions - 1.0 being exactly the same and 0.0 being independent.

3.3.3 Disease Enrichment

By linking the proteins in a cluster to known disease annotations it is possible to estimate the likelihood that a given community is involved in a particular disease. The code being used produces a p-value to indicate this likelihood. In the case of this project, due to the aims of the SYNSYS project and for simplicity, only two diseases were investigated: Schizophrenia and Alzheimer's.

Conclusion

This project involved a large number of varied techniques and algorithms being used on large data sets. Many problems had to be solved in order to piece together the full project. As each of these steps were sequential, the results

section describes the results as they were gathered, in the same structure as seen above.

Chapter 4

Results

As the project progressed the intended approach was found to be flawed and some changes were made. The first of these was the change from a DIP-based training set to HIPPIE-based training set. After the classification had been performed the use of a supervised classifier with this training set was found to be a poor method by itself for weighting interactions. Using all of the data available it was possible to run a simple Bayesian alternative to continue with the weighting for the Community detection algorithm.

4.1 PPI feature vectors

Many features were identified for extraction and these are described in Appendix A.4. Of these, only a small subset were succesfully processed into a usable form. These can be found in appendix A.4.

Of these, a smaller proportion were used in the final classifier, which neglected features directly derived from interaction databases. The features used to train the final classifier are shown in table 4.1. A brief description of these features is given below.

4.1.1 Gene Ontology

The Gene Ontology(**ashburner_gene_2000**) is a resource of annotations for genes to indicate various characteristics in a hierarchical manner, such as cellular localisation or function. This resource has been used in past papers(**qi_evaluation_2006**) and in databases such as STRING(**von_mering_string:_2005**) to predict protein interactions. Intuitively, it can be used to detect when, for example, two

Feature	Size	Type	Coverage on training set	Coverage on active z
Gene Ontology	90	Binary categorical	100.0%	100.0%
Yeast two-hybrid	1	Numerical	100.0%	100.0%
ENTS derived	107	Numerical	38.39%	42.74%

Table 4.1: A table summarising the components of the feature vectors used in the final classifier.

proteins are localised in the same area of the cell - as this would increase the probability that these two proteins interact. Details on exactly how this feature was generated can be found in the notebook reference in Appendix A.4.1.

4.1.2 Features derived from ENTS

These features were obtained through analysis and modification of the bundled code and data downloaded from the website of **rodgers-melnick__predicting__2013** In turn, most of these features were generated through the Multiloc2 program of **blum__multiloc2:__2009** The remaining features are pairwise combinations of conserved protein domains, which are conserved "modules" of proteins described in **janin__domains__1985**

4.1.3 Data visualisation

For all graphs, two cases were investigated: in proportion and out of proportion. In proportion refers to the case where the proportion of interactions to non-interactions is correct; specifically, 1 interaction to 600 non-interactions. Out of proportion maintains the same number of interactions to non-interactions. This is important as it is often easier to separated the data when the classes are equally split.

Reducing dimensionality

If the data were two dimensional we would like to plot it as a scatter plot and see if there was a clear grouping of the points. This would indicate that a classification algorithm would be able classify the data. As the data has in excess of one hundred dimensions it is necessary to reduce the dimensionality before plotting.

Two methods were tested to reduce the dimensionality of the data to two dimensions so that it could be easily plotted. The first of these is PCA, which is relatively simple with a fast implementation, and the second is t-SNE, which is more complicated but has achieved better performance in recent works. Both of these methods are described below in more detail.

In PCA the method to express a high-dimensional point $\boldsymbol{x}$ as a low dimensional point relies on finding an approximation for this point according to(**barber_bayesian_2013**):

$$\boldsymbol{x}^n \approx \boldsymbol{c} + \sum_{j=1}^{M} y_j^n \boldsymbol{b}^j \equiv \tilde{\boldsymbol{x}}^n \tag{4.1}$$

Where the low dimensional points are $\boldsymbol{y}^n$, $\boldsymbol{c}$ is a constant and $\boldsymbol{b}_j$ are the principal components. The algorithm to find these principal components is described in **barber_bayesian_2013**

The resulting plot produced using this technique is shown in figures 4.1 and 4.2 for the in proportion and out cases, respectively. In the case of the data being out of proportion it appears that the groups can be separated. However, this is not the case for the in proportion case.

A more complex method to reduce the dimensionality of the data is t-SNE(**van_der_maate**). This technique won the Merck Visualization Challenge. It differs from PCA in the objective function in that it aims to maintain a similarity measure between points between the high and low-dimensional cases.

In addition to using this technique it was also recommended to use Truncated Singular Value Decomposition(SVD) to reduce the dimensionality of the vector to 50 beforehand. This was for implementation based reasons in Scikit-learn.

Unfortunately, this was no more successful than PCA on this dataset, as shown in figures 4.3 and 4.4. It should also be noted that the number of points plotted in these graphs is much lower than in the case of PCA as it is much more computationally intensive.

Both of the methods above suggest that this classification problem is difficult, as the points are not significantly separated in any of the plots.

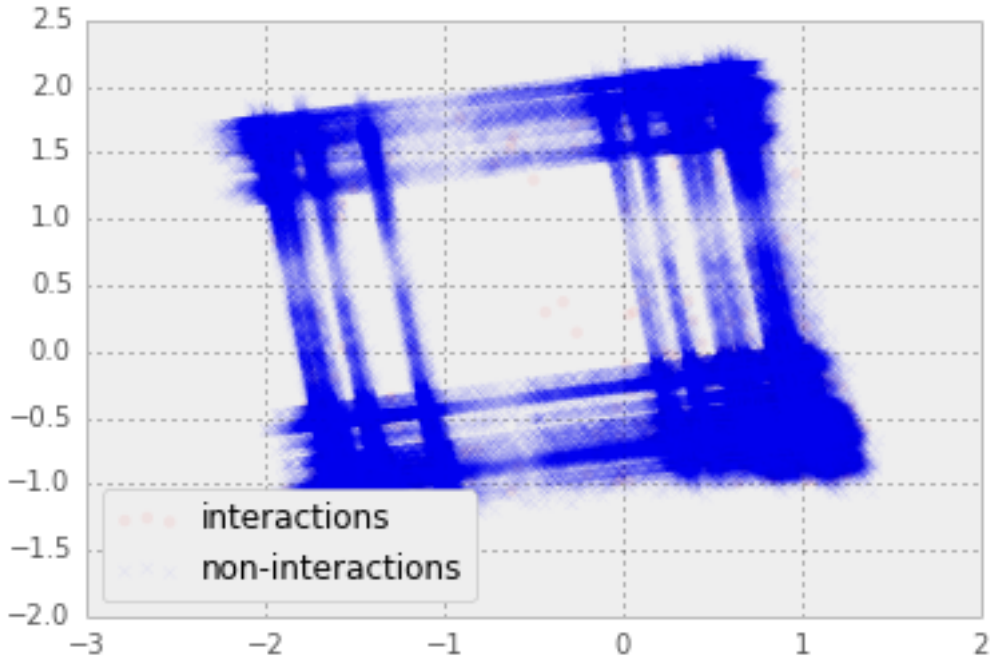


Figure 4.1: In proportion plot of the data reduced to two dimensions using PCA.

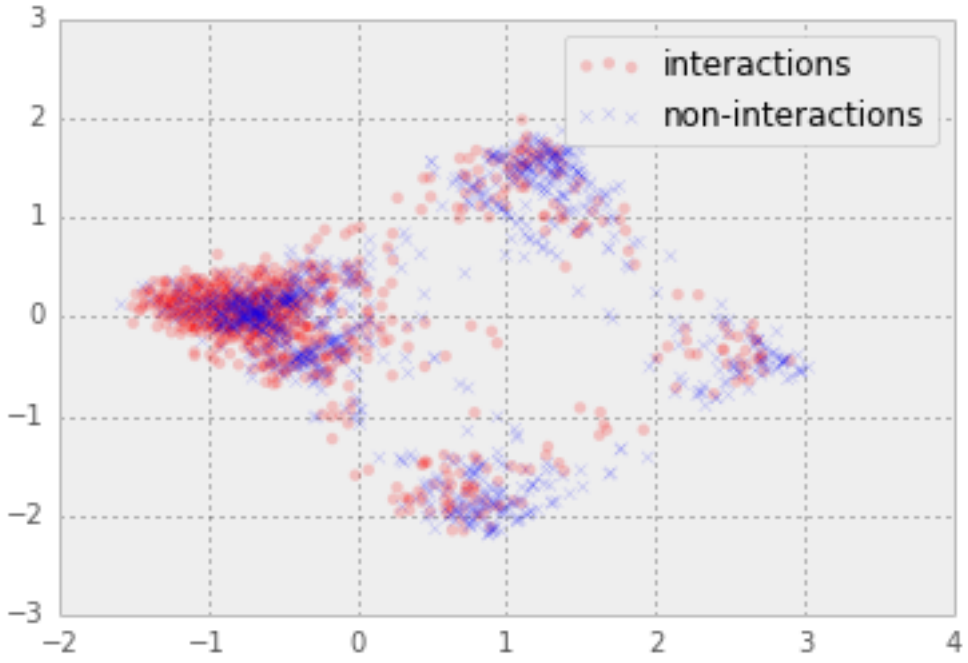


Figure 4.2: Out of proportion plot of the data reduced to two dimensions using PCA.

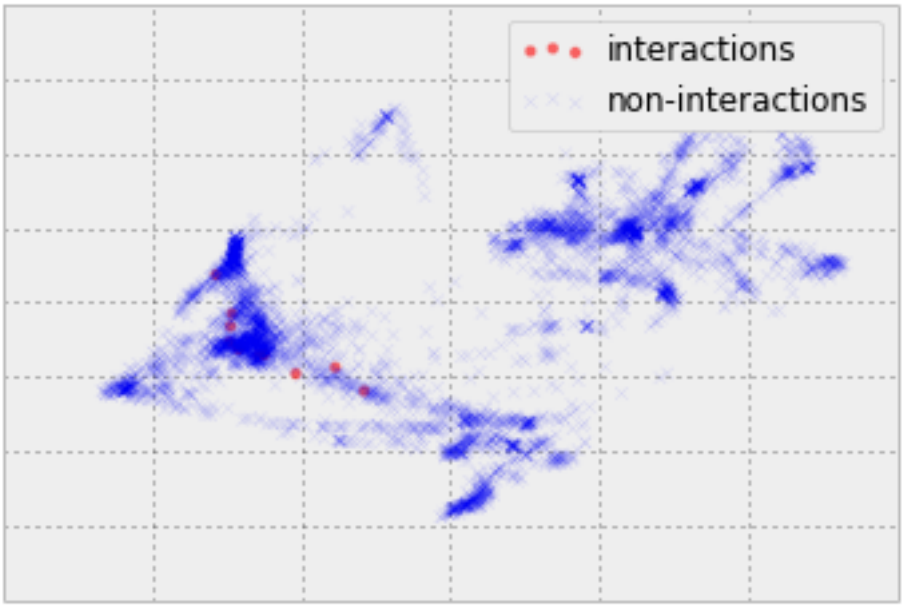


Figure 4.3: In proportion plot of the data reduced to two dimensions using t-SNE. Axes are not labeled as they are meaningless after the transformation.

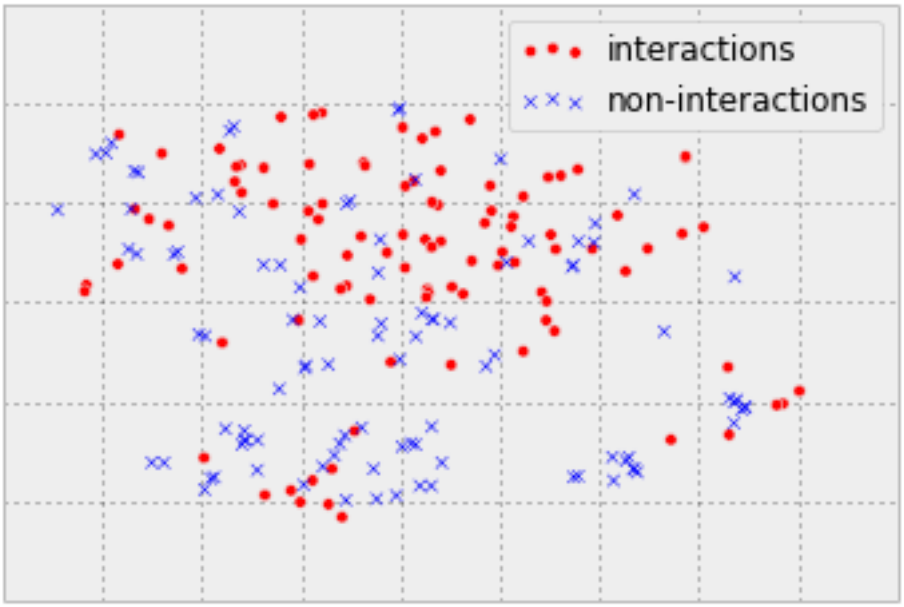


Figure 4.4: Out of proportion plot of the data reduced to two dimensions using t-SNE. Axes are not labeled as they are meaningless after the transformation.

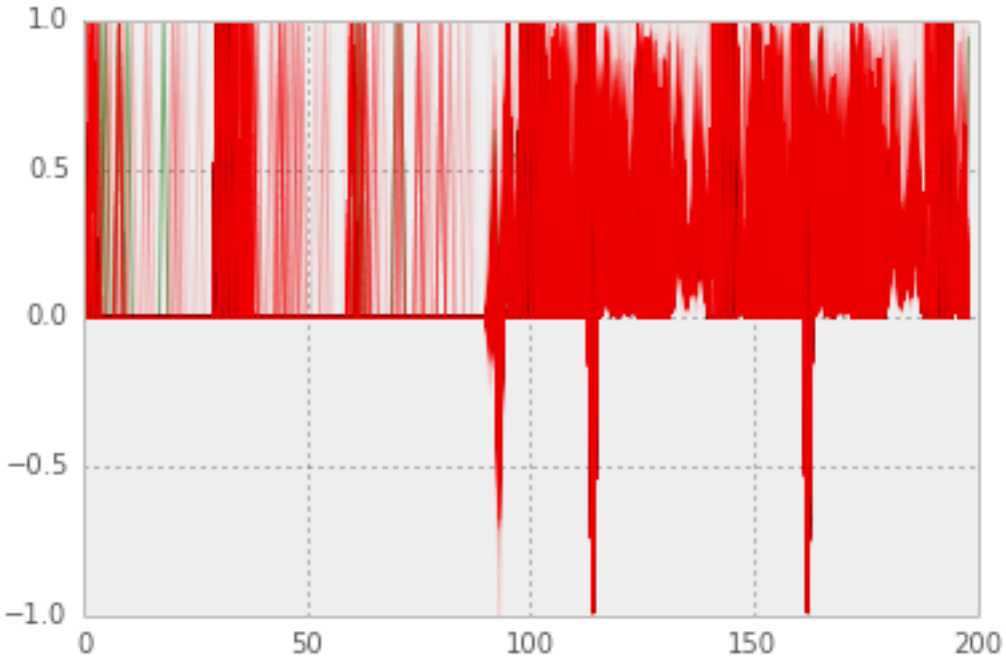


Figure 4.5: In proportion parallel line plot of the data used to train the final classifier. Each feature vector is scaled into the graph interval and plotted, overlapping.

4.1.4 High dimensional plots

Neglecting reducing the dimensionality of the data there are some plots which are able to illustrate a very large number of dimensions. Two which have been applied in this project are parallel lines plots and Andrew's curves(**andrews_plots_1972**). Parallel lines plots are relatively simple in that each feature is simply scaled and plotted along the x axis at its index in the vector, producing a number of overlapping lines. Andrew's curves are more complex, and the method is described below along with the results obtained.

The parallel lines plots are shown in proportion in figure 4.5 and out of proportion in figure 4.6. In either case, it is not clear whether the data easily separates into two classes. The in proportion case in particular shown in figure **??** is very difficult to separate into interactions and non-interactions.

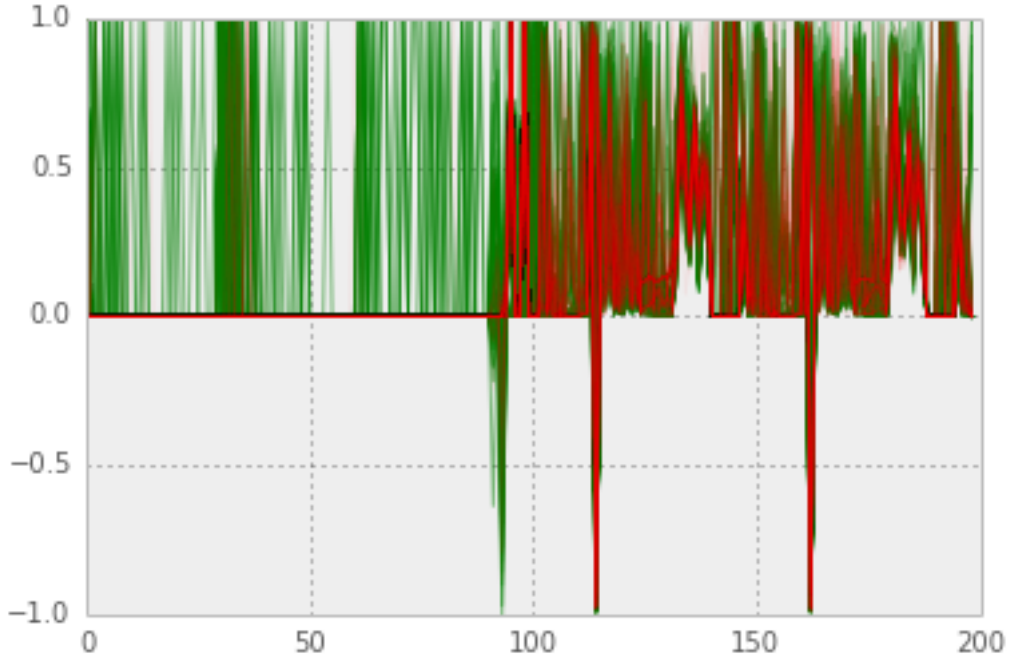


Figure 4.6: Out of proportion parallel line plot of the data used to train the final classifier. Each feature vector is scaled into the graph interval and plotted, overlapping.

Classifier	Hyper-parameter values	Test set accuracy		Training set acc
Logistic Regression	C : 0.0215	0.9984	$\pm 3 \times 10^{-5}$	0.99847
Support Vector Machine	kernel: RBF Gamma: 10.0 C: 10.0	0.9985	N/A	0.99933
Random Forest	N estimators: 44 Max features: 25	0.99837	$\pm 3 \times 10^{-5}$	0.99921
Extremely Randomized Trees	N estimators: 94 Max features: 25	0.99837	$\pm 3 \times 10^{-5}$	0.99921

Table 4.2: Summary of the hyper-parameter combinations found by grid search and associated statistical accuracy measures.

4.2 Classification in weighted PPI networks

In a classification task there are two common components: a training set that is labeled and a set of data which the classifier is to be applied to, without labels. This task is similar in that we have created a training set with labels and the active zone network forms a set of feature vectors that have no labels. However, this does not accurately encode our prior knowledge of the active zone network interactions, which were picked for their likelihood to be true interactions. For this reason, in this application classification alone is not sufficient to solve the problem of creating accurate weights, driving the development of the technique described in section 3.2.3.

This section describes both the results of training the classifier as planned on the labeled training set and the results of the Bayesian method to weight the interactions of the active zone network.

4.2.1 Classifier accuracy and best parameters

Grid searches of hyper-parameter values were used to find the optimal combination. The results of this search are shown in table 4.2.

Although the results of the Support Vector Machine appear to be good, it was trained on a relatively small dataset and its performance in later tests deemed it unsuitable to our classification task. The random forest can be seen to be

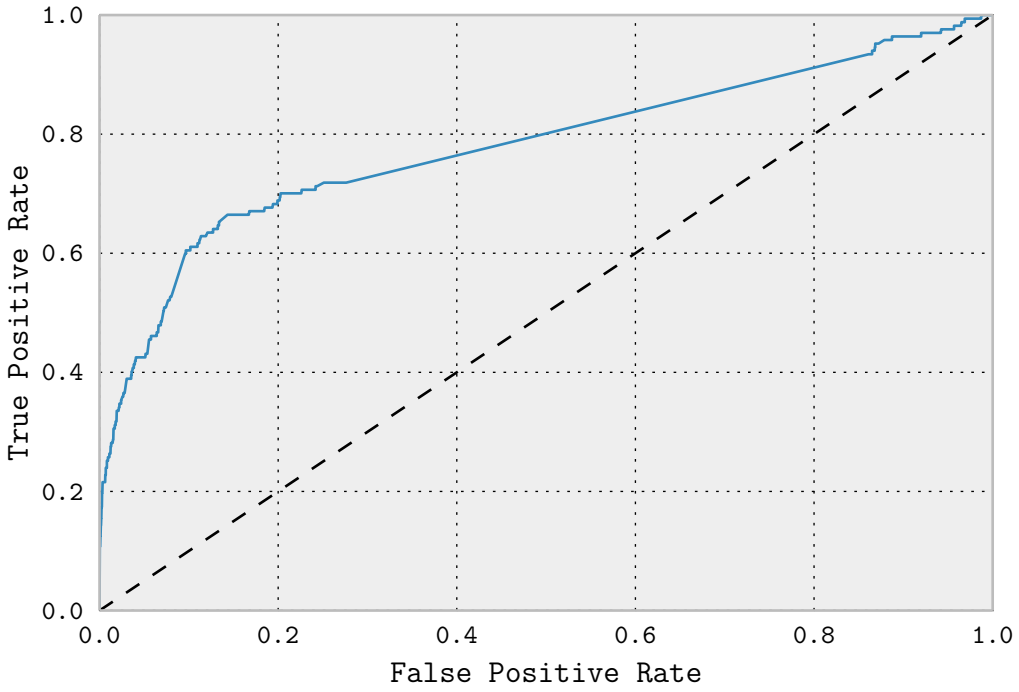


Figure 4.7: The ROC curve produced by a logistic regression model on the test data set using the best parameters shown in table 4.2 with an AUC of 0.781.

performing worse than the logistic regression model in this test, with a slightly lower accuracy, but within the standard error boundaries.

4.2.2 ROC curves

As described in section 3.2.1 ROC curves were applied to all classifiers to characterise the performance of each. The ROC curve produced by the logistic regression model is shown in figure 4.7. The positive AUC value is desirable and the curve of the ROC curve suggests this model can achieve a relatively good true positive rate at low false positive rate, which is likely what it would be used at - to avoid false positive protein interactions being classified.

However, the ROC curve shown in figure 4.8 is very similar to that created by the logistic regression model except it has slightly better performance at low false positive rates. Neither show very good performance and it clear that only at very high false positive rates will the classifiers reliably find the true interactions.

Unfortunately, it was not possible to train the support vector machine on a large enough training set to plot a realistic ROC curve due to time constraints,

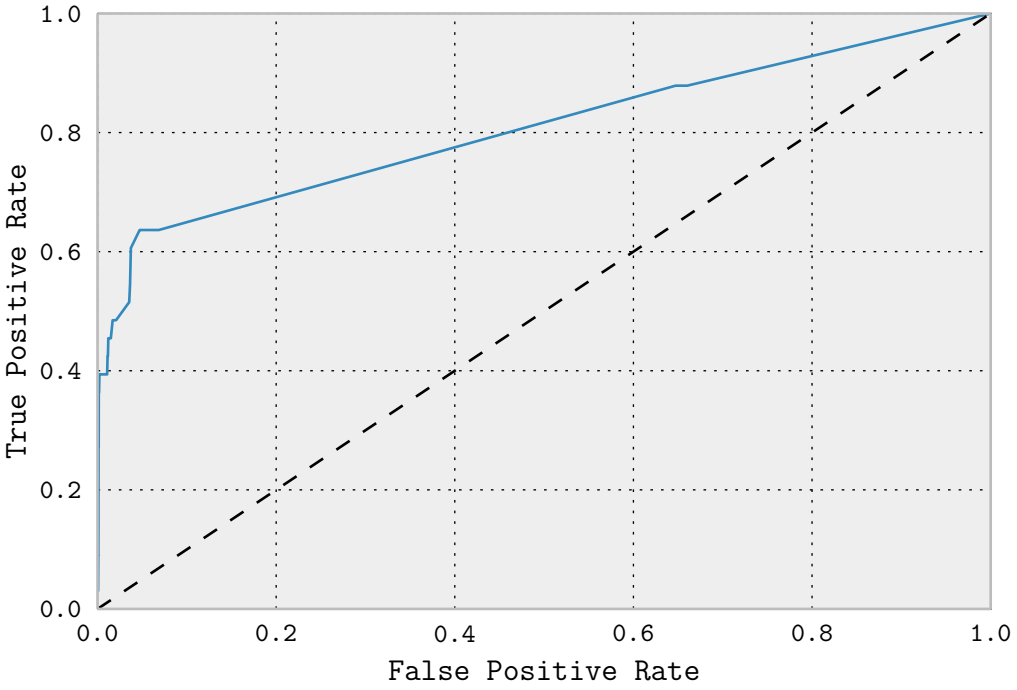


Figure 4.8: The ROC curve produced by a random forest model on the test data set using the best parameters shown in table 4.2 with an AUC of 0.806.

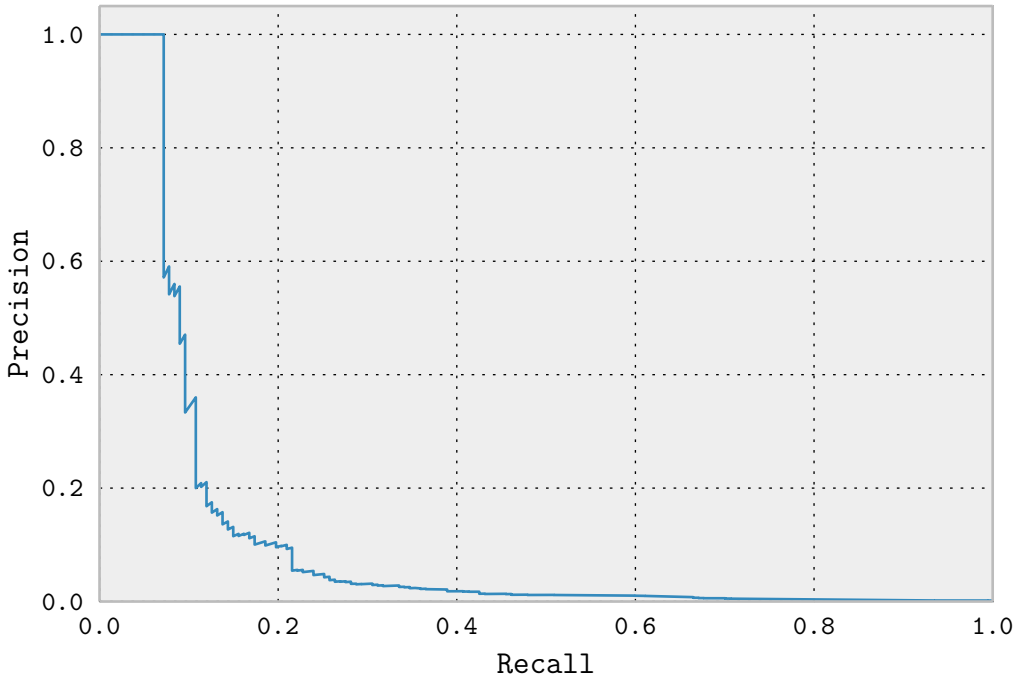


Figure 4.9: The precision-recall curve produced by a logistic regression model on the test data set using the best parameters shown in table 4.2 with an AUC of 0.11.

so it is not shown here. The notebook referenced in appendix A.2 provides the results obtained. This is also true of the precision-recall graph in the following section.

4.2.3 Precision-recall curves

As described in section 3.2.1 the classifiers were also characterised through plotting a precision-recall curve. This is shown for the random forest and logistic regression models in figures 4.10 and 4.9, respectively. Although the random forest model achieves a better AUC value, the curve demonstrates is unable to recall as many samples with perfect precision as the logistic regression model. As with the ROC curve, neither classifier shows particularly good performance.

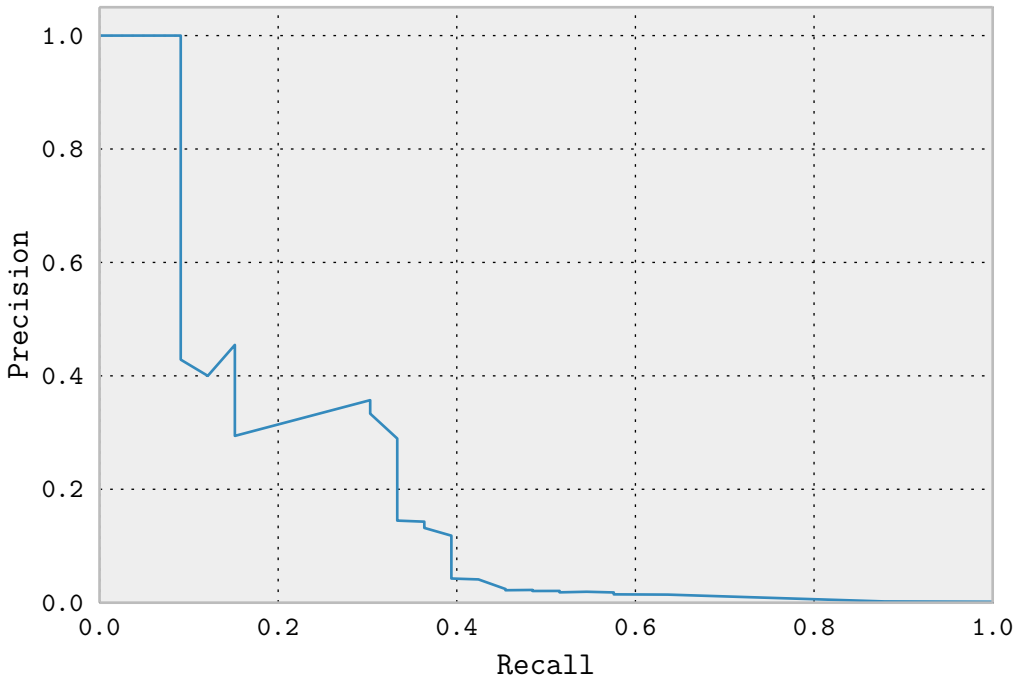


Figure 4.10: The precision-recall curve produced by a random forest model on the test data set using the best parameters shown in table 4.2 with an AUC of 0.12. This graph shows some artefacts due to the extremely small number of positive interactions even in relatively large sample sizes.

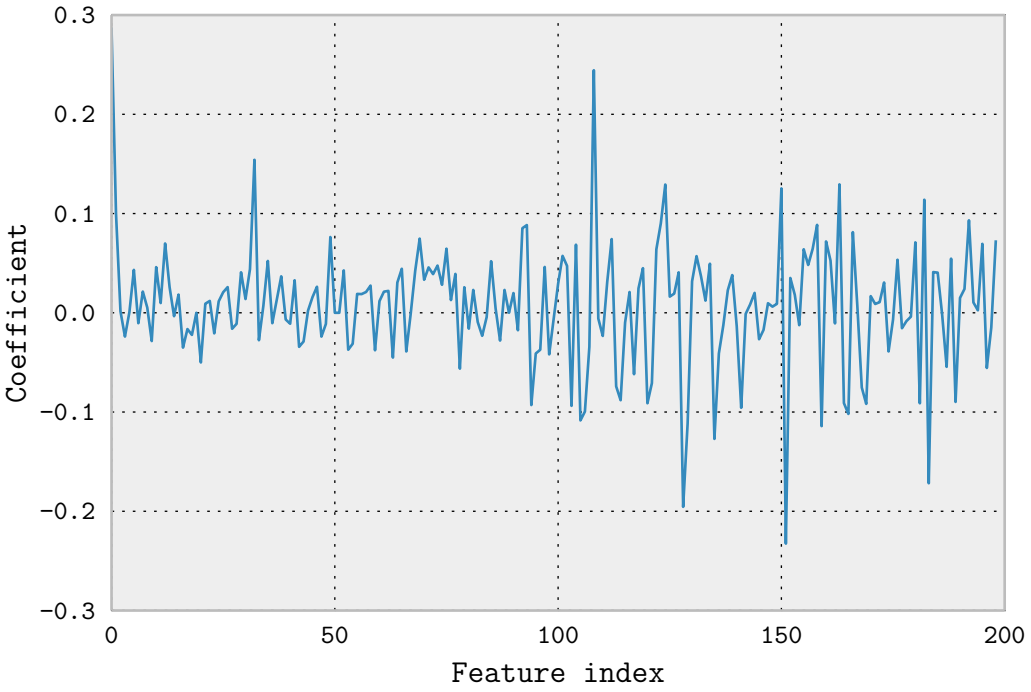


Figure 4.11: The internal weighting of features in a logistic regression model trained using the parameters described in table 4.2. The coefficients plotted on the y axis have no unit.

4.2.4 Feature importances

Both logistic regression models and random forests can quickly return feature importances. In the case of logistic regression, this is as simple as looking at the weights applied to each input feature. Random forests store the importances of each feature internally in Scikit-learn and this can be easily queried. These importances measures are plotted in figures 4.11 and 4.12.

4.2.5 Bayesian weighting of interactions

Confidence values were chosen to be conservative but otherwise were arbitrary to the task and could be more carefully chosen in future work. Estimating the error rates of the relevant databases was beyond the scope of this project. The values chosen are shown in table 4.3. As the Edgelist has been reviewed and because iRefIndex represents a combination of many databases a TPR of 0.9 seemed like a fair estimate of the accuracy of positive predictions. Negative predictions were

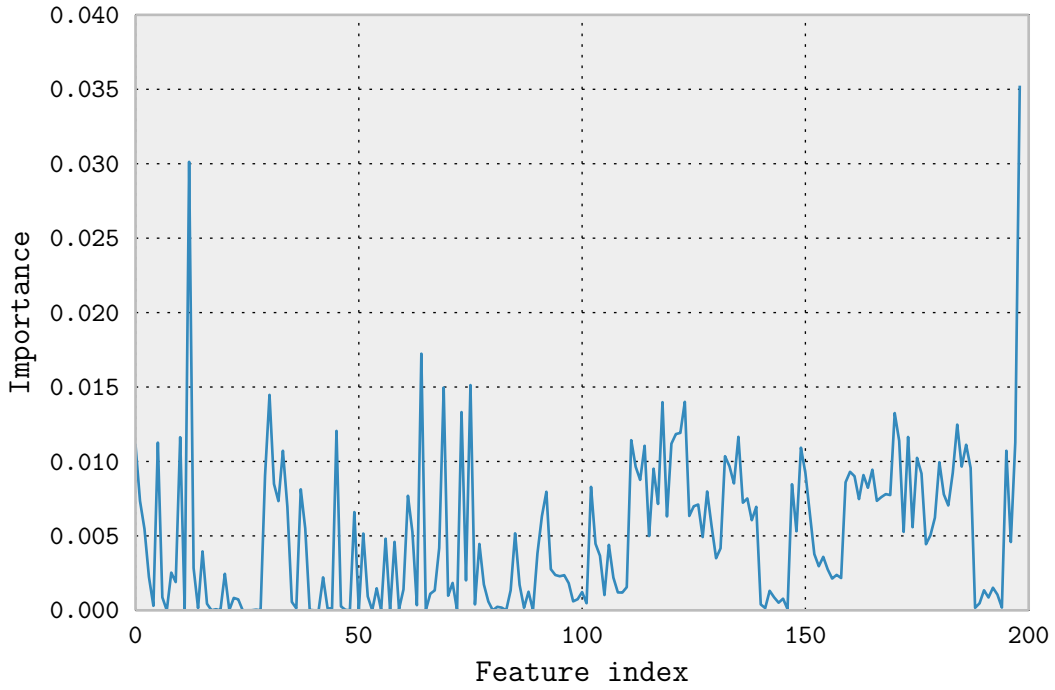


Figure 4.12: The importances of each input feature as reported by the random forest model trained using the parameters described in table 4.2. The importances plotted on the y axis have no unit.

Parameter	Value
Edgelist TPR	0.9
Edgelist TNR	0.5
iRefIndex TPR	0.9
iRefIndex TNR	0.5

Table 4.3: A table summarising the parameters chosen in the probabilistic model used two weight the interactions.

more difficult to estimate. Both TNRs were set to be 0.5, which is very likely to be an underestimate as most interactions are false.

Once the Bayesian updating of the weights was completed it was necessary to check the distribution of the weights produced. This distribution is shown in figure 4.13 Unfortunately, the distribution produced forms two dense groups due to the majority of binary features involved.

4.3 Comparison of weighted and unweighted PPI networks

Once the weighted edges had been generated the communities in both the weighted and unweighted cases were found using a Spectral Modularity algorithm. The resulting graphs of the active zone network, divided into communities, are shown in figures 4.14 and 4.15 for the unweighted and weighted cases, respectively. By eye, it is easy to see that the clusters detected differ.

This observation is mirrored in the results of the NMI test. Both NMI test implementations returned a value of 0.604, indicating that the clustering is quite similar.

4.3.1 Disease enrichment

The communities detected to be involved in disease differ by number between weighted and unweighted. As these numbers are arbitrary, it is necessary to compare these communities between the two graphs to see if both graphs contain similar communities involved in disease. The significant disease communities discovered in each graph can be found in Appendix **??**.

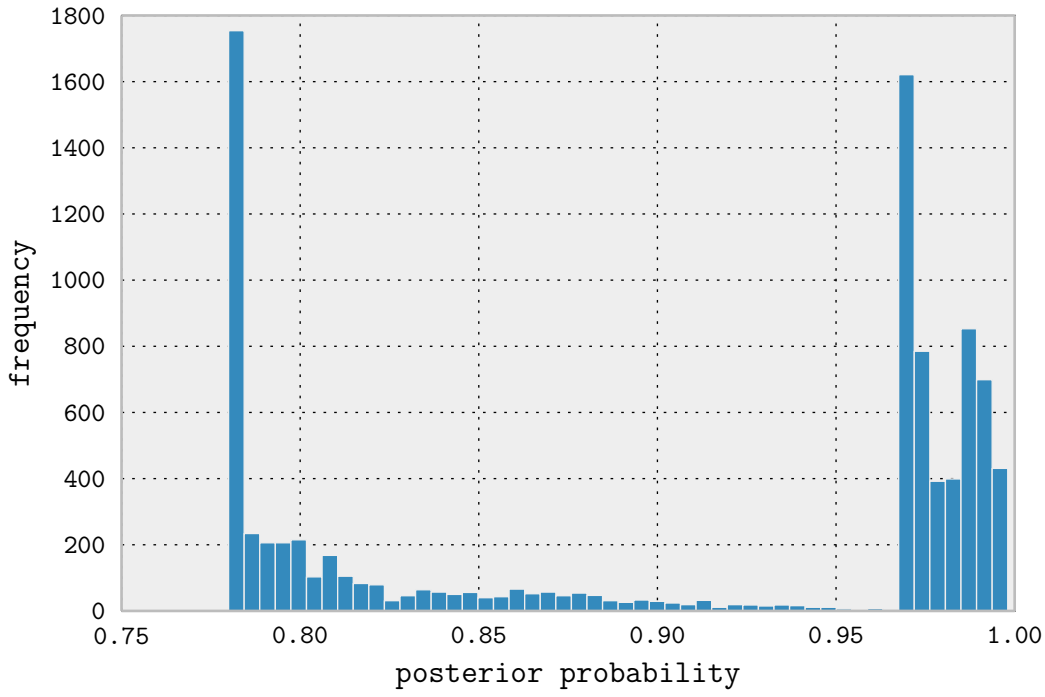
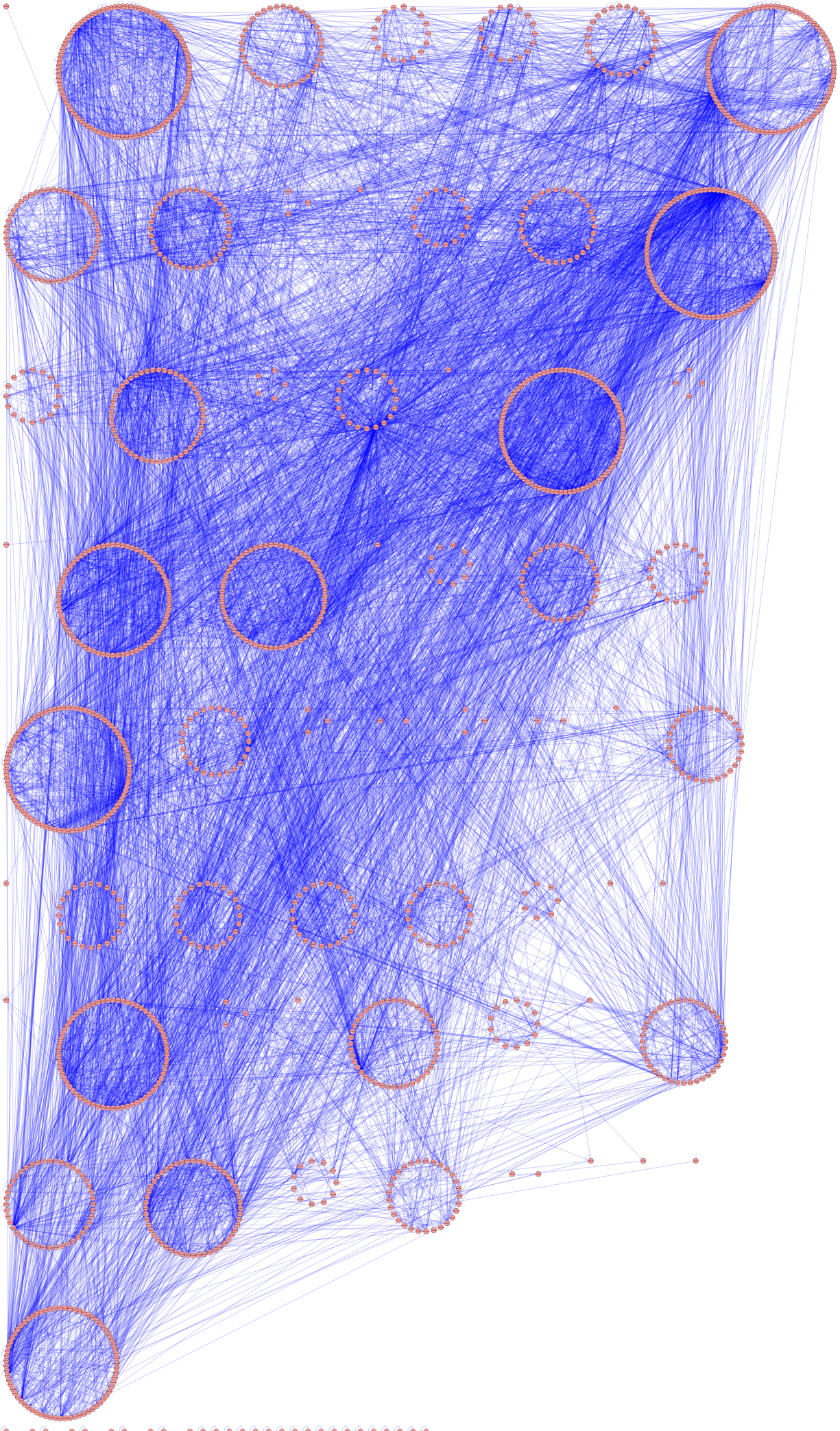
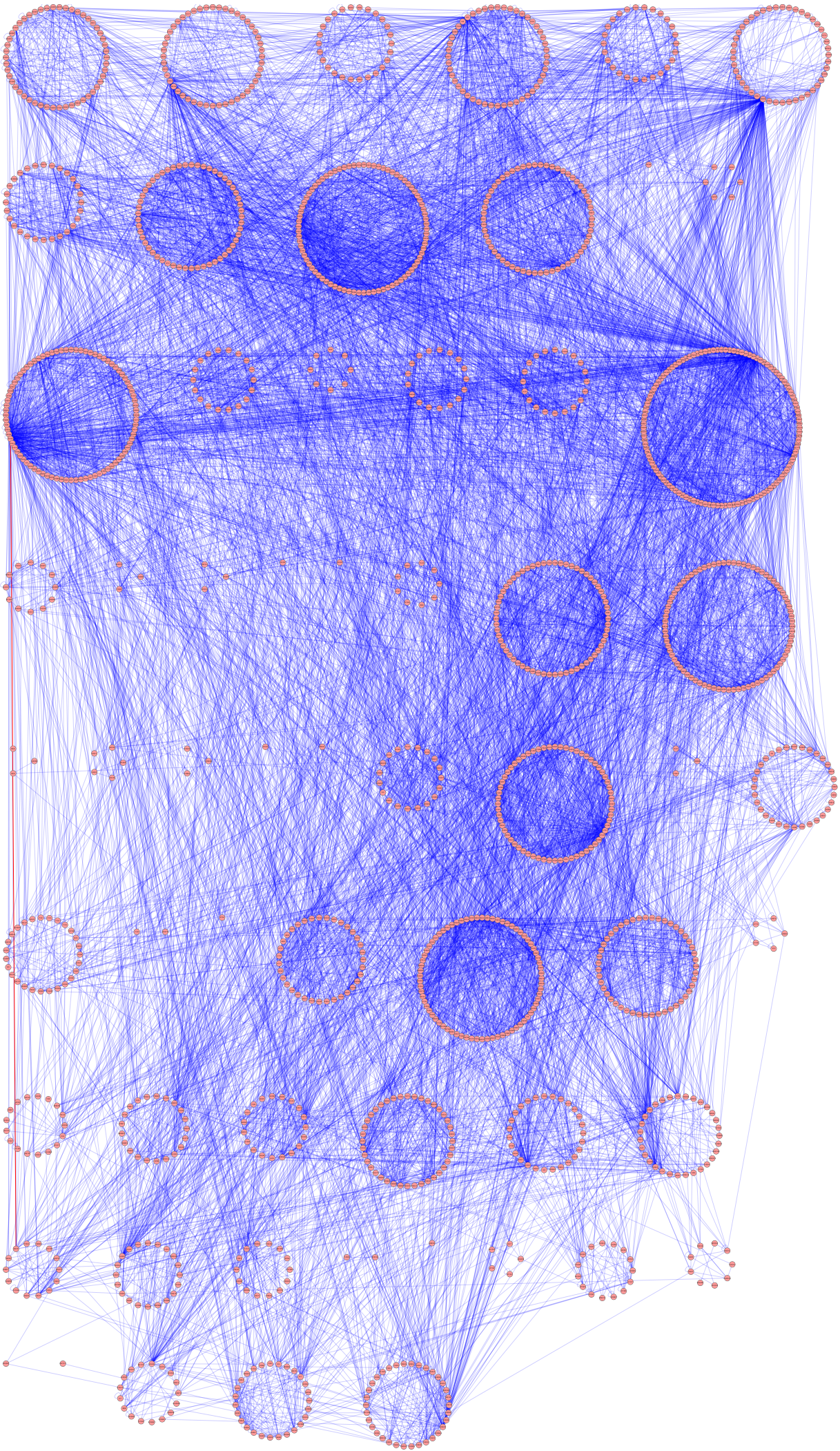


Figure 4.13: A histogram of the weights produced using the method of Bayesian weight generation described in section 3.2.3.





_33.pgf **!! Missing graphics !!** _29.pgf **!! Missing graphics !!**

Figure 4.16: The communities 29 and 33 from the unweighted and weighted graphs, respectively are plotted. In each plot the nodes not shared in each community are plotted separated from the main community on the right.

_44.pgf **!! Missing graphics !!** _64.pgf **!! Missing graphics !!**

Figure 4.17: The communities 64 and 44 from the unweighted and weighted graphs, respectively are plotted. In each plot the nodes not shared in each community are plotted separated from the main community on the right.

Looking at the most likely community for Schizophrenia in the unweighted graph, it is possible to compare the members of this community to the members of all the communities in the weighted graph to find the closest match. Community 29 in the unweighted graph was found to be the most likely to be involved in Scizophrenia. The membership test used equates to the number of members shared between two communities over the number of members in either community. For this community it was found that community 33 in the weighted graph most closely matched 29 with a value of 0.45.

These graphs were then plotted to investigate the reason for this difference as shown in figure 4.16. It can be seen that there are many more points found in the unweighted graph that are also in the weighted graph than vice versa. In this graph it is not clear why many of these proteins have been separated, there are no clear weak connections holding together the unweighted case. However, in figure 4.17 we can see that the weighted graph contains most of the proteins of the unweighted community minus a set that are very weakly connected.

Conclusion

Although it is easy to demonstrate that weighting the graph has had an effect on the communities detected, it is much more difficult to discern what this effect has been or how the weighting has brought it about. However, many of the results presented here are positive, showing that the various techniques involved can be used to produce a result.

Chapter 5

Conclusions

This project involved an array of different tools and data. Despite many difficulties and mistakes during the planning of the project, all of the aims of the project as stated in the proposal have been met. However, the results obtained are not conclusive or useful.

5.1 Deliverables

During the course of the project a large number of data sources were tested and extracted into a machine learning workflow. These formed large feature vector files which could be used for classification. Three different classifiers were then trained on this data and compared. Of these, the best was used to provide predictions to weight edges. A Bayesian method was used to combine this with other data sources and prior knowledge to generate the edge weights. These edge weights were then used to create a weighted PPI graph which was then compared to its unweighted counterpart.

5.2 Future work

As a basis for future work, this work illustrates many difficulties in working with varied publicly available data sources. However, it also provides insight into the correct method for weighting interaction edges. These edges should be weighted directly as an estimate of interaction strength. The premise of this project was that the posterior probability of the interaction existing would correlate well with the strength of the interaction as interactions which are strong will be observed

more often.

However, if a full probabilistic model was to be designed the latent variable - which was interaction in our model - could be a continuous variable in the unit interval defined at a Beta distribution. The problem then becomes one of estimating interaction strength, which is difficult to observe in order to obtain the training set required to create a probabilistic model. Using the array of biological databases available it would be possible to link different observations based on strong biological prior knowledge.

Conclusion

Despite the results of this project, the resources and insight into better solutions generated made it worthwhile. In the future it, building on the results of this project it will be possible to create a PPI network that accurately summarises our knowledge of interactions and interaction strength using all of the available data.

.1 Repository

Git is version control software and was used extensively in this project in combination with Github(**github**) and git-annex(**gitannex**). The project is built from two repositories, the first inside the second: a large git-annex based repository containing all the data and a smaller git repository stored on github containing all the code and documentation for the report(**opencast-bio**). History for all the code and documentation, every previous version, is available publicly online but the data cannot be made publicly available.

Hosting the code online was intended to aid remote collaborators. It is also useful in order to maintain an accurate log of the work involved in the project. The repository also includes a wiki(**opencastbiowiki**) providing documentation on some of the code available in the repository and weekly reports covering the entire project.

The repository consists of several directories:

- notebooks:

  - Contains IPython notebooks covering code executed during the project.
  - Each notebook includes inline documentation on what is being done, and why.

- ocbio:
  - This contains the Python module of code developed during the project.
  - The major component of this is the extract.py file, which deals with writing feature vectors for use in classification.

- proposal:
  - Only contains the original proposal for the project.

- report:
  - Contains this report and all the required files to compile it.
  - Based on a repository for Masters project templates(**ug4template**) with modifications by Danilo Orlando.

- scripts:
  - Contains scripts, but these were not used during the bulk of the project.

The code in this repository may be useful to a future project, but could also be substantially improved upon. It is more likely that the notes on how to go about a protein interaction prediction task, and this report, will be more useful to future work.

Weekly reports were kept as a summary of the work completed every week for supervisors and to maintain a log of the project. These can be found on the project wiki(**opencastbiowiki**).

.1.1 Parallel processing with IPython.parallel

To take maximum advantage of the available computing facilities and because the sample sizes in the project exceed one million the decision was made to prepare the code for parallel processing on a remote server. Particularly, grid

search operations to optimize performance of the classifier were considered to be processor intensive and vital to the success of the prediction task. The easiest way to set up these interactive parallel processing operations was the parallel processing model in IPython(**parallel_python_webpage**).

The notebooks using parallel processing are the notebooks on classifier training, which are described in Appendix A.2. This usage depended on code from a parallel processing tutorial(**ogrisel_parallel**) to distribute memory to the cores using Numpy's memmap methods. The code to do this has been integrated into the ocbio module in the project repository and can be used as shown in the notebooks. Potentially, and as described in the tutorial, this code could be used to run the classifier training on cloud services using Starcluster.

Appendix A

Notebooks

The job of quickly running arbitrary processing on a variety of different data sources, each of which are being encountered for the first time was approached using IPython notebooks. Quick interactive programming was useful as unexpected problems could be quickly solved. Also, a detailed log, with inline comments, could be kept to track exactly what was done.

It would be possible for anyone with access to the data to run this code again to verify it. The code was run with Python version 2.7.7 and Scikit-learn 0.15.0. The notebooks can be found at the following locations:

- The root directory for these notebooks can be found here: `https://github.com/ggray1729/opencast-bio/tree/master/notebooks`
- These can be viewed here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/tree/master/notebooks/`

A.1 Feature Extraction Notebooks

Of the feature extraction notebooks in the repository, not all of them were successful. Only those that were successful are listed here.

A.1.1 Gene Ontology

In total 90 features were extracted from the Gene Ontology as binary values. The following notebook describes how the features applied were generated:

- The notebook in the opencast-bio repository can be found here: `https://github.com/ggray1729/opencast-bio/blob/master/notebooks/Extracting%20Gene%20Ontology%20features%200.2.ipynb`
- This notebook can be viewed here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/blob/master/notebooks/Extracting%20Gene%20Ontology%20features%200.2.ipynb`

A.1.2 Features derived from ENTS

107 features were generated derived from the work in **rodgers-melnick__predicting__2013** These were found by analysing the code provided on the papers web page as described in the following notebook:

- The notebook in the opencast-bio repository can be found here: `https://github.com/ggray1729/opencast-bio/blob/master/notebooks/Inspecting%20ENTS%20code.ipynb`
- This notebook can be viewed here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/blob/master/notebooks/Inspecting%20ENTS%20code.ipynb`

A.2 Classifier Training

This notebook contains all the code that was run to train and test the classifier used in this project. It involves model selection, grid search of parameters and various plots describing the performance of different classifiers, such as ROC curves and Precision Recall curves. Links are provided to the code and an online service to view the notebook:

- The notebook in the opencast-bio repository can be found here: `https://github.com/ggray1729/opencast-bio/blob/master/notebooks/Classifier%20Training%20HIPPIE.ipynb`
- This notebook can be viewed here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/blob/master/notebooks/Classifier%20Training%20HIPPIE.ipynb`

A.3 ocbio.extract Usage

This notebook describes how to use the code developed in the project to build feature vectors from the various data sources. It can be found in the following locations:

- The notebook in the opencast-bio repository can be found here: `https://github.com/ggray1729/opencast-bio/blob/master/notebooks/ocbio.extract%20usage%20notes.ipynb`
- This notebook can be viewed here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/blob/master/notebooks/ocbio.extract%20usage%20notes.ipynb`

A.4 Data sources

These data sources were gathered on the project wiki, which can be found here: `https://github.com/ggray1729/opencast-bio/wiki/Feature-extraction`. The vast majority could not be included in the project due to time constraints but the list may be useful for future work. Those that were extracted into usable features are listed below:

- HIPPIE database(**schaefer__hippie:__2012**)
- Pulldown derived features: affinity and abundance
- Gene Ontology(**ashburner__gene__2000**), also described in section 4.1.1
- Yeast Two-Hybrid, also described in section **??**
- ENTS derived features(**rodgers-melnick__predicting__2013**), also described in section 4.1.2
- iRefIndex database(**razick__irefindex:__2008**)
- STRING database(**von__mering__string__2005**)
- HMR database
- InterologWalk results(**gallone__bio::homology::interologwalk__2011**)

A.4.1 Gene Ontology Features

Each feature used in the Gene Ontology feature represents an event where both proteins in the interaction have the same Gene Ontology term active. If both are active the feature is 1, otherwise it is zero. To select effective features, commonly occurring features in the active zone network were selected from each Gene Ontology domain. These are summarised by domain and in the order used in the feature vectors in table A.1 for the first 10 of each domain. The remaining terms are listed in table A.2.

A.5 Disease enrichment results

This appendix contains the results for disease enrichment between the weighted and unweighted graphs for the diseases Schizophrenia and Alzheimer's. These can be found in tables A.3 and A.4 for the weighted and unweighted results of the disease enrichment test, respectively.

Domain	Term	Occurrences in active zone	Index
Molecular Function	protein binding	960	1
	poly(A) RNA binding	308	2
	ATP binding	232	3
	metal ion binding	137	4
	identical protein binding	100	5
	RNA binding	99	6
	protein homodimerization activity	97	7
	calcium ion binding	88	8
	zinc ion binding	79	9
	GTP binding	78	10
Cellular Component	cytoplasm	614	31
	cytosol	575	32
	nucleus	554	33
	extracellular vesicular exosome	550	34
	plasma membrane	449	35
	mitochondrion	265	36
	integral component of membrane	252	37
	nucleolus	235	38
	nucleoplasm	143	39
	membrane	130	40
Biological Process	small molecule metabolic process	266	61
	gene expression	184	62
	viral process	144	63
	signal transduction	130	64
	cellular protein metabolic process	125	65
	synaptic transmission	114	66
	RNA metabolic process	104	67
	mRNA metabolic process	99	68
	transmembrane transport	98	69
	blood coagulation	97	70

Table A.1: The features used in the Gene Ontology feature, by domain, and in the order used in the feature vector. The observed frequency in the active zone network is also shown. These are only listed for the first 10 of each domain. The remaining features, in order, are shown in table A.2.

Molecular Function	Cellular Component	Domain Indices
1-30	31-60	
protein binding	cytoplasm	
poly(A) RNA binding	cytosol	
ATP binding	nucleus	
metal ion binding	extracellular vesicular exosome	
identical protein binding	plasma membrane	
RNA binding	mitochondrion	
protein homodimerization activity	integral component of membrane	
calcium ion binding	nucleolus	
zinc ion binding	nucleoplasm	
GTP binding	membrane	
nucleotide binding	cell junction	
protein kinase binding	perinuclear region of cytoplasm	
DNA binding	Golgi apparatus	
structural constituent of ribosome	endoplasmic reticulum	
GTPase activity	integral component of plasma membrane	
actin binding	extracellular space	
enzyme binding	mitochondrial inner membrane	
molecular_function	mitochondrial matrix	
protein complex binding	endoplasmic reticulum membrane	
calmodulin binding	extracellular region	nuclear
protein serine/threonine kinase activity	dendrite	
receptor binding	intracellular membrane-bounded organelle	
protein domain specific binding	cytoskeleton	
structural molecule activity	microtubule	
protein heterodimerization activity	Golgi membrane	
protein C-terminus binding	neuronal cell body	
SH3 domain binding	protein complex	
structural constituent of cytoskeleton	ribonucleoprotein complex	
transporter activity	centrosome	SR
signal transducer activity	actin cytoskeleton	

Table A.2: The full list of GO terms used in the feature vector, by index, is shown.

Community	Size	disease	p-value	{p-value}	
44	35	schizophrenia	0.00344584	0.564032	0.00
45	73	schizophrenia	0.00389962	0.5285770000000001	
48	17	Alzheimer's_disease	0.00757007	0.6198670000000001	
33	64	Alzheimer's_disease	0.00839002	0.563842	
33	64	schizophrenia	0.0233474	0.541118	
14	18	schizophrenia	0.0418727	0.5903689999999999	
2	48	schizophrenia	0.046719300000000005	0.563059	
15	8	Alzheimer's_disease	0.049793199999999996	0.666429	

Table A.3: Disease enrichment results for the weighted communities detected in the active zone network.

Community	Size	disease	p-value	p-value	Sig
29	88	schizophrenia	0.00169105	0.549829	0.0
2	99	schizophrenia	0.00432129	0.547038	0.00
30	26	Alzheimer's_disease	0.00537398	0.597882	0.00
61	24	schizophrenia	0.019590700000000003	0.562575	0.00
63	23	Alzheimer's_disease	0.0350298	0.5739609999999999	0.00
61	24	Alzheimer's_disease	0.0425309	0.593381	0.00
63	23	schizophrenia	0.046159399999999996	0.589334	0.00

Table A.4: Disease enrichment results for the unweighted communities detected in the active zone network.