Title Pending

Gavin Gray

Master of Science by Research
Neuroinformatics
DTC in Neuroinformatics and Computational Neuroscience
School of Informatics
University of Edinburgh
2014

Abstract

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

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

The motivation for this work is as it was stated in the proposal(**proposal**):

> Disorders of the central nervous system, such as major depression or schizophrenia, affect 1 in 3 people in the developed world. At the synaptic level these diseases are likely related to proteins and their interactions found inside the synapse(**synsys**; Chua et al., 2010). The diseases of the synapse are related to the proteins in the synapse(Chua et al., 2010), therefore understanding these proteins may help to treat disease(Li, Klemmer and Smit, 2010).

It is estimated that disorders of the brain cost Europe €798 billion Euros in 2010(Olesen et al., 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(**synsys**; Chua et al., 2010). 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, Klemmer and Smit, 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

1.2.1 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.

1.2.2 Planning

The project was split between five tasks:

- Feature Extraction
- Classifier Training
- Community Detection and analysis
- Writing the report

The original plan for this project placed feature extraction as the dominant task. This was correct, in that the project only found time for training the classifier and performing Community Detection within the last five weeks. However, the reasons for delays were unknown at the time of beginning and were due largely to inexperience with Bioinformatics.

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**).

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 attempt to 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 section 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(**citation_needed**). Curing these diseases therefore may be possible through a greater understanding of the interactions of proteins at the synaptic level(**synsys**; Chua et al., 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 and Cho (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 et al. (2001) and Uetz et al. (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.

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 et al., 2013; Wang et al., 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, 2012). 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, 2012). 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, 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 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, Culp and DiFazio (2013). 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 **??**. These different data sources fall into categories:

- Primary interaction databases, such as DIP(Xenarios et al., 2002), HIPPIE(Schaefer et al., 2012) or BioGRID(Stark et al., 2006).
- Associated features, such as features derived from Gene Ontology(Ashburner et al., 2000) or those used in Rodgers-Melnick, Culp and DiFazio (2013).
- Other PPI prediction resources, such as STRING(**vonMering_string_2005**) or InterologWalk(Gallone et al., 2011).

The indirect sources of data were chosen based on usage in the literature, such as in the case of Gene Ontology(Qi, Bar-Joseph and Klein-Seetharaman, 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

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 section 3.1.2. 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.

3.1.1 IPython 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.

3.1.2 Supervised binary classification

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, 2012, p. 2). 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 $\mathcal{D}$ 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 HIPPIE(Schaefer et al., 2012) database with over 50% confidence. Negative interactions are three million random binary combinations of Entrez protein IDs, which is a method applied in other works(Qi, Bar-Joseph and Klein-Seetharaman, 2006) to create negative training examples. There are many more negative that positive interactions and it is unlikely that any of the negative examples are real interactions.

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

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

These expressions are defined:

- The posterior probability:

$$p(y^* = 1|\boldsymbol{x}^*, \mathcal{D}, \mathcal{H}) \tag{3.3}$$

- The likelihood:

$$p(\boldsymbol{x}^*|y^* = 1, \mathcal{D}, \mathcal{H}) \tag{3.4}$$

- The prior:

$$p(y^* = 1|\mathcal{D}, \mathcal{H}) \tag{3.5}$$

- The marginal likelihood:

$$\sum y^* p(\boldsymbol{x}^*|y^*, \mathcal{D}, \mathcal{H}) \tag{3.6}$$

We do not apply explicitly apply a prior to the probability of interaction, meaning that we implicitly apply a uniform prior.

3.1.3 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 that originally chosen to use in PPI network work at Edinburgh: the NCBI Entrez identifier.

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 relatively simple, just consisting of a number generated when the gene was added to the database(**maglott_entrez_2006**).

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 et al., 2009):
  - Able to map from various identifiers to others based on different databases.
  - Produces a csv which can be used in a variety of different tools.
  - Requires a list of protein identifiers.
- NCBI's Gene(**maglott_entrez_2006**) provides conversion tables on it's ftp servers:
  - Simple tab-separated text file converting all known GeneIDs between different formats.
- The Uniprot(**consortium_universal_2007**) online service:

  - Similar to BioMart, but with a simpler interface.
  - Converts only to or from Uniprot identifiers.
  - Requires a list of protein identifiers.

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.

3.1.4 Dedicated code: ocbio.extract

3.1.5 Gold standard datasets

The database of interacting proteins (DIP) is a database of interactions proven by small-scale experiments(Xenarios et al., 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, Bar-Joseph and Klein-Seetharaman (2006).

3.1.6 PPI prediction features

3.1.7 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.

However, the gains from parallel processing were mitigated by trying to run the notebook server on the remote server. The clusters could have been initialised and the jobs submitted locally to the remote server operating as a cluster, thus taking advantage of all the available resources. This would also be a better system if the parallel processing were to be scaled up to use other processing clusters.

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 et al., 2011). 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.

3.2.1 Classification algorithms

Three classification algorithms were chosen to test on the data:

- Logistic regression
- Support vector machine (SVM)
- Random forest

The reasons for choosing these and brief description of each algorithm are given below.

3.2.1.1 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, 2012, p. 376):

$$p(c = 1|\boldsymbol{x}) = \sigma(b + \boldsymbol{x}^T * \boldsymbol{w}) \tag{3.7}$$

Where $\boldsymbol{x}$ is the vector of features. The weights and biases are the parameters of this model, expressed in the above equation as b and $\boldsymbol{w}$.

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.

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 is:

$$L(\boldsymbol{w}, b) = \sum_{n=1}^{N} c^n \log \sigma(b + \boldsymbol{w}^T \boldsymbol{x}^n) + (1 - c^n) \log(1 - \sigma(b + \boldsymbol{w}^T \boldsymbol{x}^n)) \tag{3.8}$$

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(Murphy, 2012, p. 381). To stop this from happening we apply a penalty term to the size of the weights:

$$L'(\boldsymbol{w}, b) = L(\boldsymbol{w}, b) - C\boldsymbol{w}^T\boldsymbol{w} \tag{3.9}$$

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

3.2.1.2 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, 2012, p. 383). 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_an_2000**).

The hyper-parameters for a Support Vector Machine control the kernels, along with the regularisation parameter as described for logistic regression in section 3.2.1.1. 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.

3.2.1.3 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 3.1.

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, 2012, p. 544). There are several implementations of algorithms to achieve this and a full description can be found in Murphy (2012, p. 544). 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.

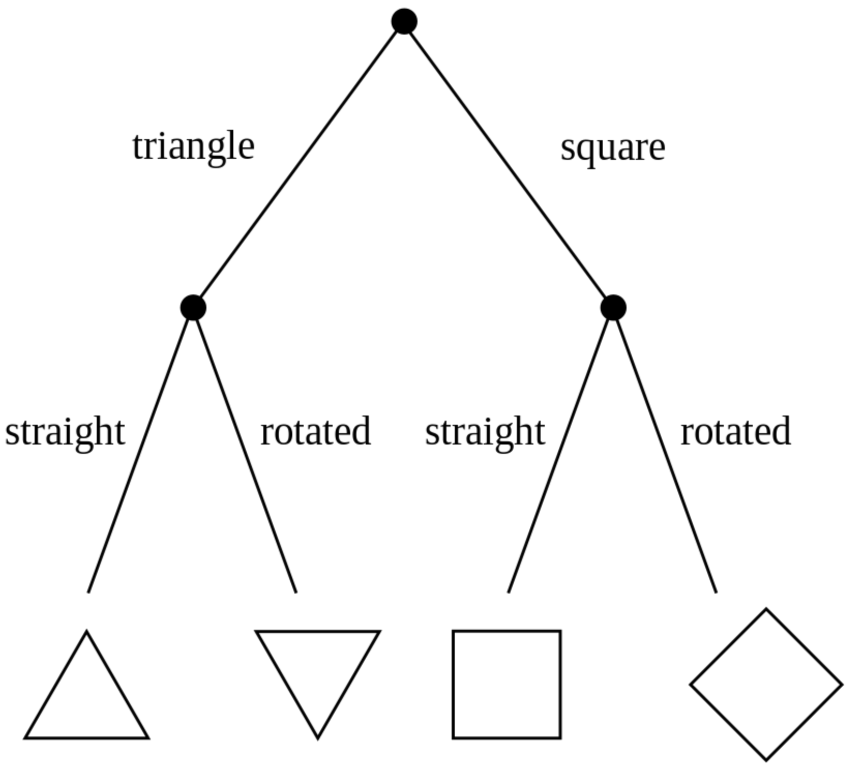

Figure 3.1: A simple example decision tree, illustrating the process of sequential, dependent decisions.

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, 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, 2012):

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x) \tag{3.10}$$

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, Bar-Joseph and Klein-Seetharaman, 2006; Rodgers-Melnick, Culp and DiFazio, 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
- 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 and Langley, 1995). Beta regression would have been very suitable for the task and is suggested as future work, described in the following section.

3.2.1.4 Beta regression

A generalised linear model is one in which the output variable is distributed according to a given target density(Murphy, 2012). In the case of Beta regression(Smithson and Verkuilen, 2006) this target model is the Beta distribution, which has the convenient characteristic of being bounded between 0 and 1. 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 et al., 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.

3.2.2 Classifier verification

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, Frank and Hall, 2011, p. 152). 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.

3.2.2.1 Pipeline

3.2.2.2 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.

3.2.2.3 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 Regression 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.

3.2.2.4 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.

3.2.2.5 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 et al., 2010). These concerns center on the problems of small sample sizes, which are not the case in this project.

3.2.2.6 Precision-recall curve

A precision-recall curve plots the precision of a classifier against its recall as the threshold of classification is varied. These terms are defined:

- Precision - the number of true positive results over the number of total positive results

- Recall - the number of true positive results against the number of available positive examples

Again, the area under the curve can gauge a classifier's effectiveness.

3.2.3 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.

3.2.4 Bayesian weighting

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 and Girvan (2004). All three are based on betweenness measures to partition the graph, making them optimisation approaches, but each calculates their betweenness measure in a different way:

- Geodesic edge betweenness - find the shortest path between all pairs of nodes and count the number of these paths which run along each edge.

- Random edge betweenness - expected net number of times that a random walk between two nodes will pass through an edge summed over all node pairs.

- Spectral betweenness - pending

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, 2003):

$$I(X;Y) = H(X) - H(X|Y) \tag{3.11}$$

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 et al., 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.

Code produced at Edinburgh to calculate NMI was also used. The problem with the Scikit-learn code is that it assumes the class labels are correct, but they have just been arbitrarily numbered as the communities are found. This code takes that into account, ensuring that the class labels are accurate to the IDs in the community.

3.3.3 Disease Enrichment

Conclusion

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 ??. Of these, only a small subset were succesfully processed into a usable form. These are listed below and more information about each can be found in Appendix ??:

- HIPPIE database
- Pulldown derived features: affinity and abundance
- Gene Ontology, also described in section 4.1.1
- Yeast Two-Hybrid, also described in section 4.1.3
- ENTS derived features, also described in section 4.1.2
- iRefIndex database
- STRING database

Feature	Size	Type	Coverage on training set	Coverage on active
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.

- HMR database
- InterologWalk results

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 et al., 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, Bar-Joseph and Klein-Seetharaman, 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 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.1.1.

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, Culp and DiFazio (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**

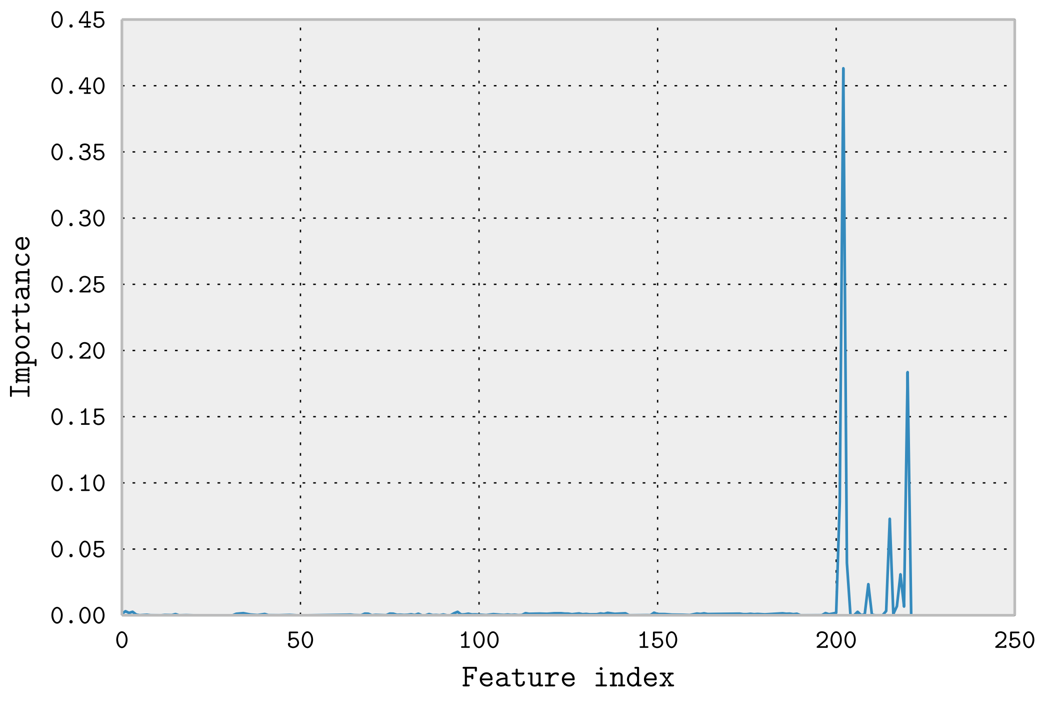

Figure 4.1: An example of an unbalanced set of feature importances plotted after fitting a Random Forest classifier to a dataset containing interaction database derived features.

4.1.3 Yeast Two-Hybrid results

4.1.4 Removed features

As listed above there were originally many features used in the supervised classification that were derived from interaction databases. These were very effective in predicting interactions on the training set, as expected, but their importance in the task outweighed any other features, as shown in figure 4.1. It was decided that 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 3.2.4; 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. The classifier was then integrated

4.1.5 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.

4.1.5.1 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, 2013, p. 330):

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

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 (2013, p. 333).

The resulting plot produced using this technique is shown in figures 4.2 and 4.3 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_maat**
). This technique won the Merck Visualization Challenge. It differs from PCA in

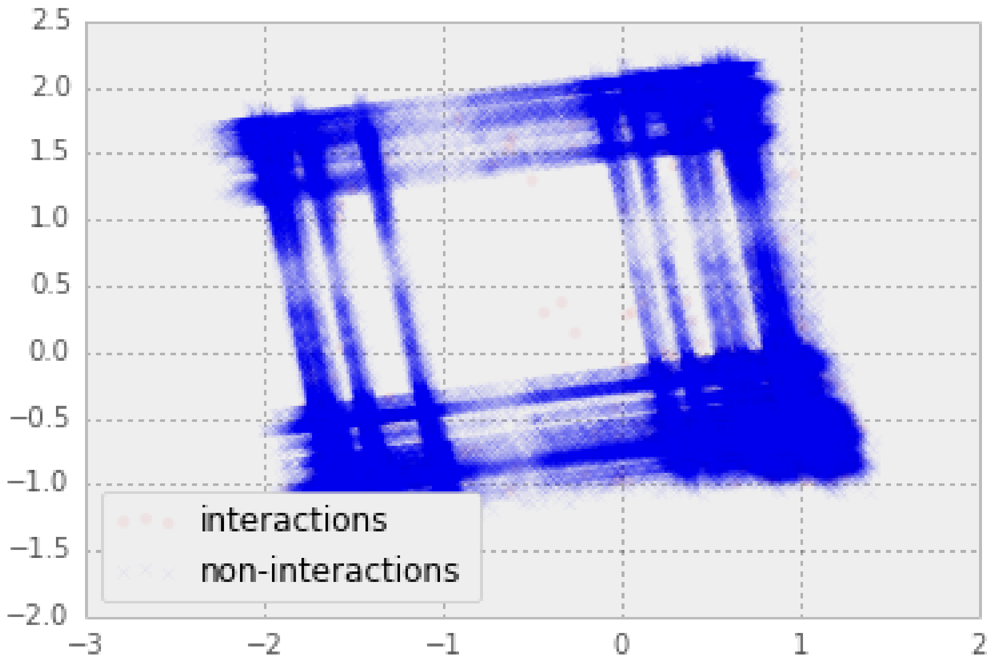

Figure 4.2: In proportion plot of the data reduced to two dimensions using PCA.

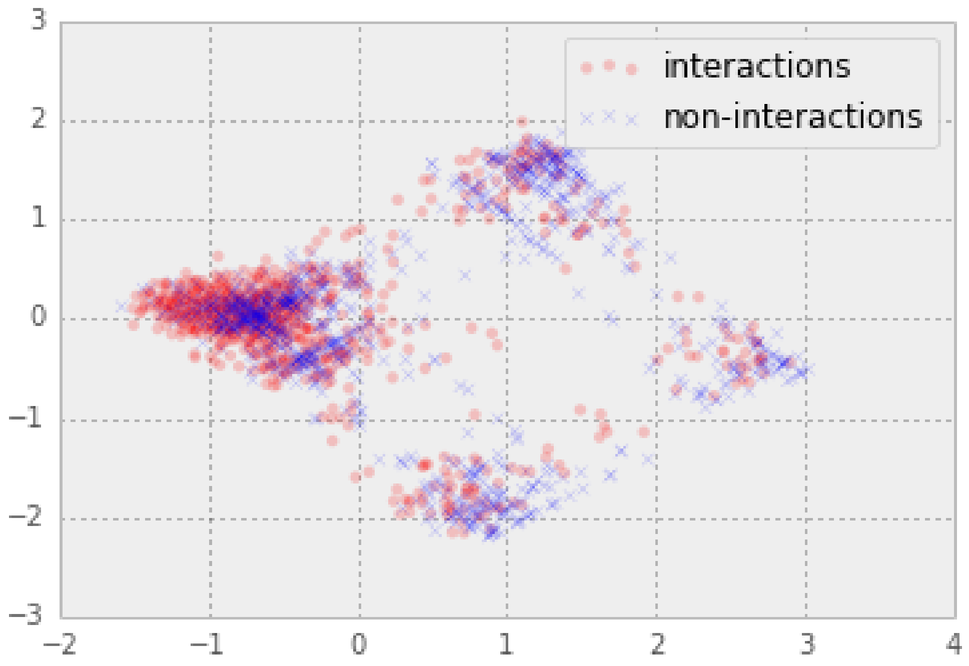

Figure 4.3: Out of proportion plot of the data reduced to two dimensions using PCA.

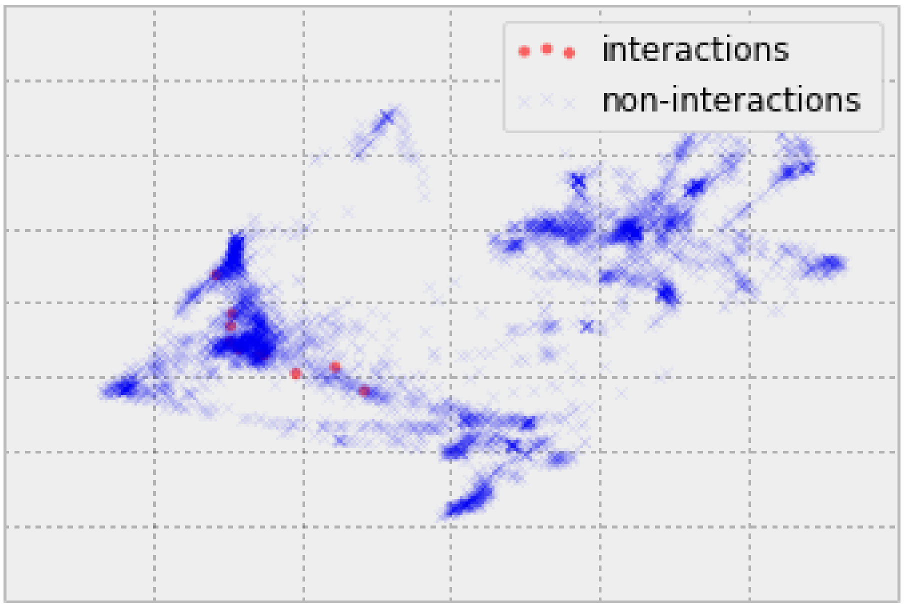

Figure 4.4: In proportion plot of the data reduced to two dimensions using t-SNE.

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 4.4 and 4.5. 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.

4.1.6 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

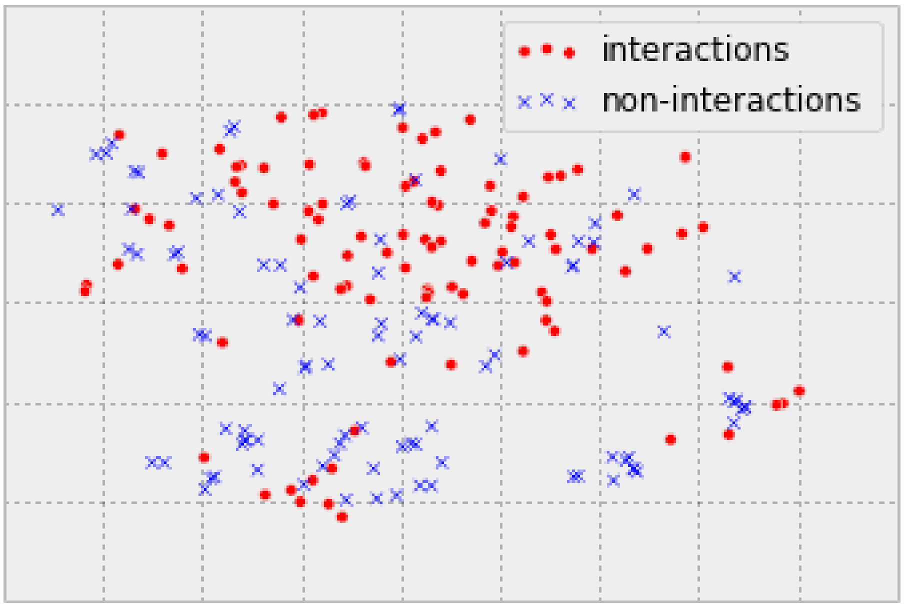

Figure 4.5: Out of proportion plot of the data reduced to two dimensions using t-SNE.

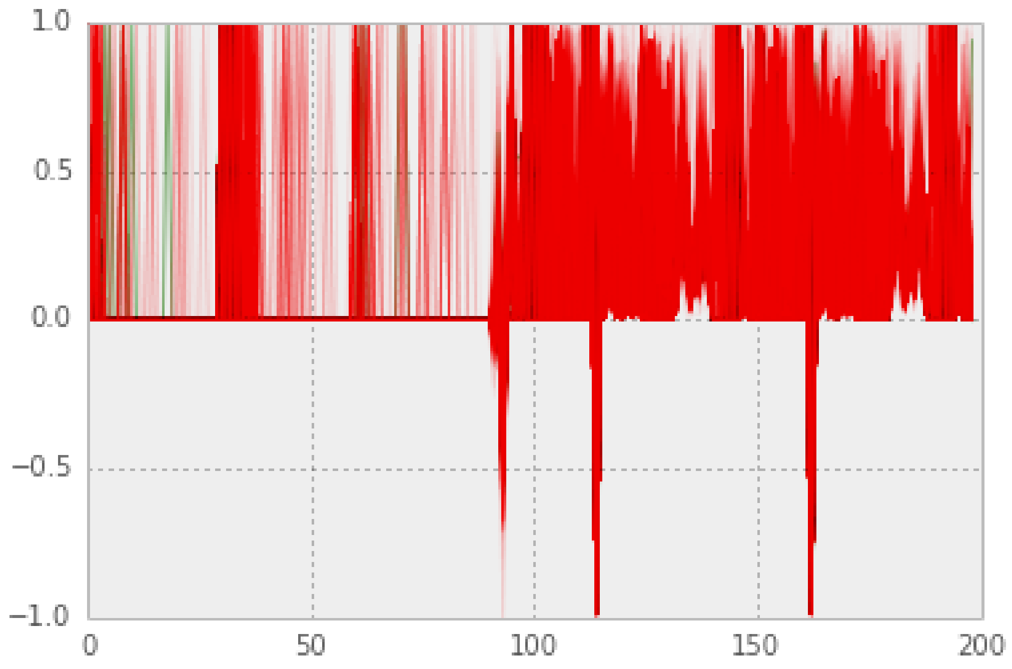

Figure 4.6: In proportion parallel line plot of the data used to train the final classifier.

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 4.6 and out of proportion in figure 4.7. 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.

Andrew's curves are similar to parallel line plots in that they consist of overlapping lines. The lines in Andrew's curves are generated by mixing waves at different frequencies weighted based on the values of the features in the vector(**andrews_plots_197**
). The result is a periodic wave that is modulated based on the values of the feature vectors. If the feature vectors of each class have reliably different values this should be reflected in the grouping of these curves. The two cases of in proportion and out are shown in figures 4.8 and 4.9, respectively.

In neither graph is it easy to see a grouping of the curves being plotted. This means that it is likely to be very difficult to accurately classify the vectors.

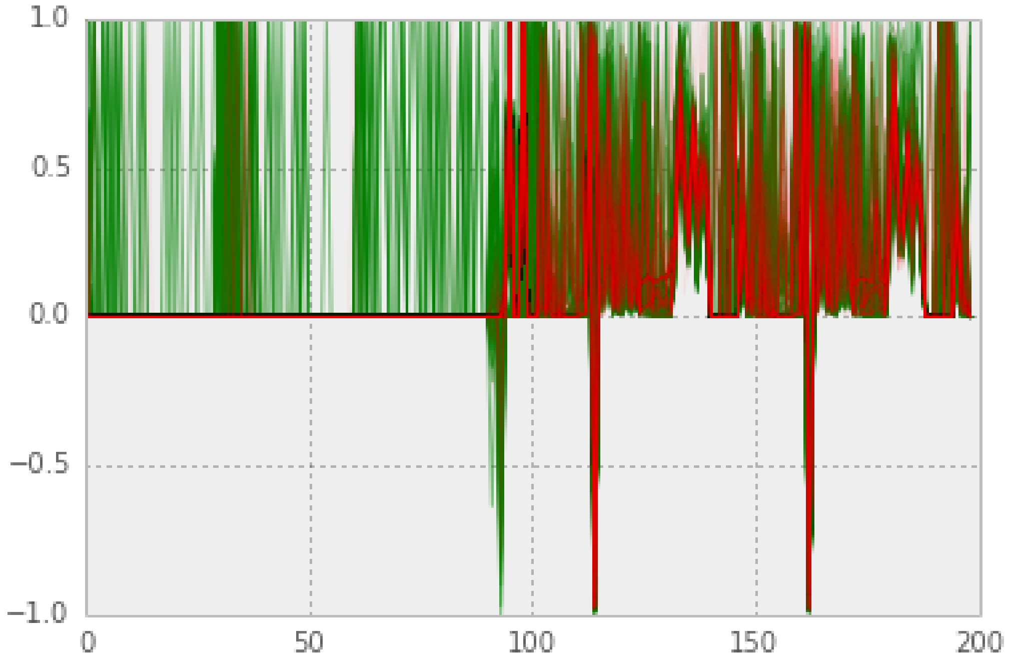

Figure 4.7: Out of proportion parallel line plot of the data used to train the final classifier.

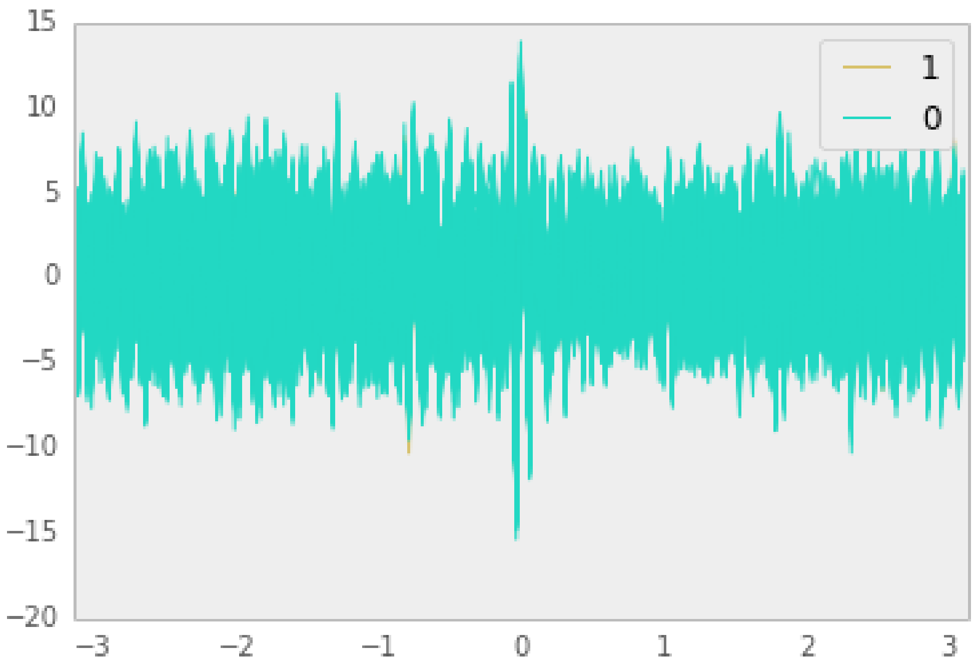

Figure 4.8: In proportion Andrew's curves plot of the data used to train the final classifier.

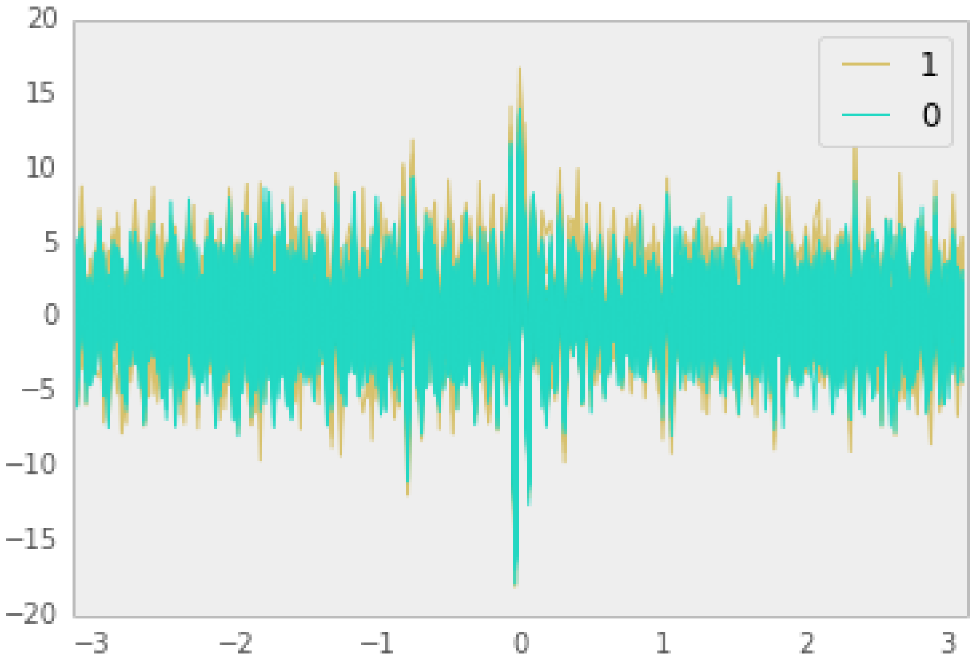

Figure 4.9: Out of proportion Andrew's curves plot of the data used to train the final classifier.

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

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 3.2.4.

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 performing worse than the logistic regression model in this test, with a slightly

Figure 4.10: 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.

lower accuracy, but within the standard error boundaries.

4.2.2 ROC curves

As described in section 3.2.2 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 4.10. 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 4.11 is very similar to that created by the logistic regression model except it has slightly better performance at low false positive rates. In addition, it has a higher AUC value.

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, 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

Figure 4.11: 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 4.12: 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.

!! Missing graphics !!

Figure 4.13: 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.

section.

4.2.3 Precision-recall curves

As described in section 3.2.2 the classifiers were also characterised through plotting a precision-recall curve. This is shown for the random forest and logistic regression models in figures 4.13 and 4.12, 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.

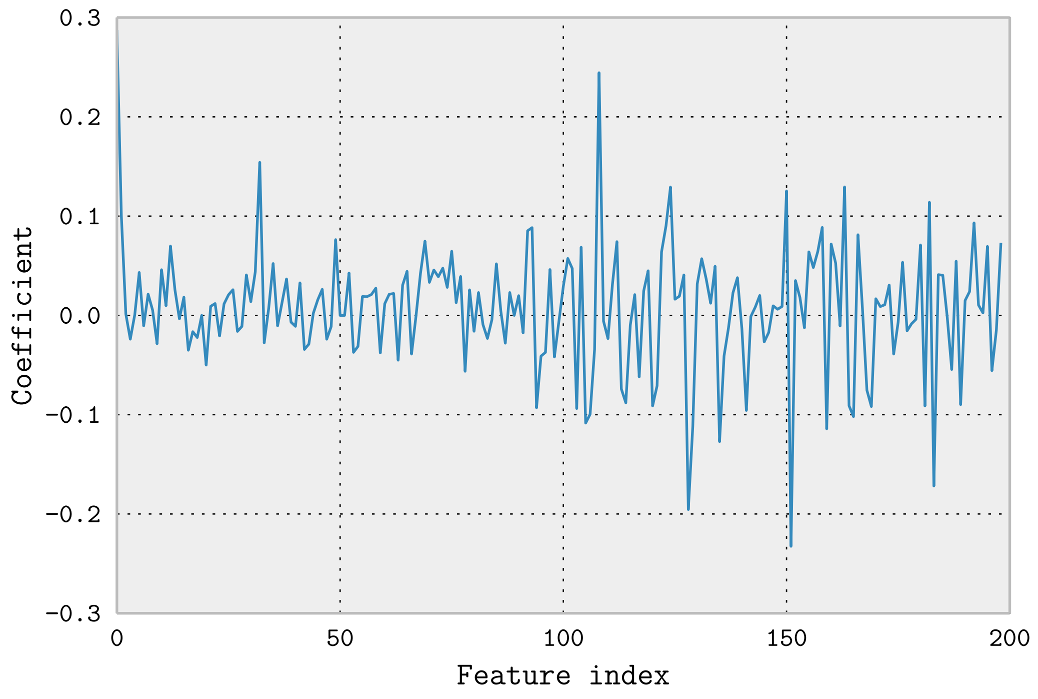

Figure 4.14: The internal weighting of features in a logistic regression model trained using the parameters described in table 4.2.

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 4.14 and 4.15.

4.2.5 Bayesian weighting of interactions

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 4.16 Unfortunately, the distribution produced forms two dense groups due to the majority of binary features involved.

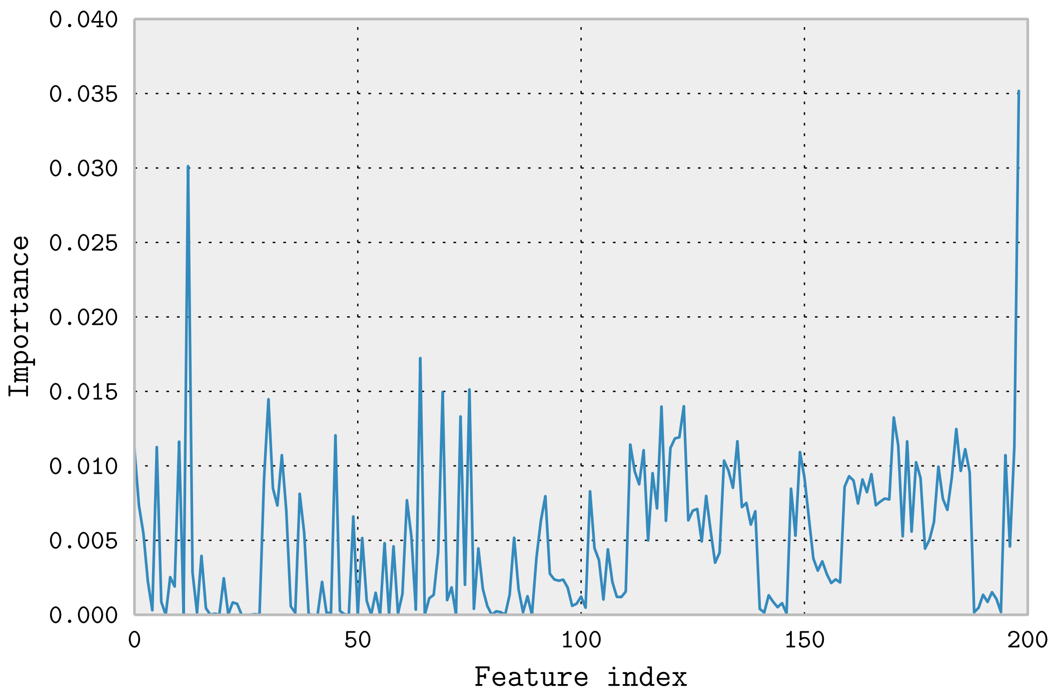

Figure 4.15: The importances of each input feature as reported by the random forest model trained using the parameters described in table 4.2.

!! Missing graphics !!

Figure 4.16: A histogram of the weights produced using the method of Bayesian weight generation described in section 3.2.4.

4.3 Comparison of weighted and unweighted PPI networks

4.3.1 Graph comparison

4.3.2 Disease enrichment

4.3.3 The effect of weighting

Conclusion

Chapter 5

Conclusions

Appendix A

Notebooks

This project took advantage of IPython notebooks to run interactive scripting for the diverse tasks necessary to organise the data. Additionally, this allowed the project to maintain comprehensive records of work 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.

A.1 Feature Extraction Notebooks

Of the feature extraction notebooks in the repository, not all of them were succesful. Only those that were succesful 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`
- A viewable version of this notebook can be found 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

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`
- A viewable version of this notebook can be found here: `http://nbviewer.ipython.org/github/ggray1729/opencast-bio/blob/master/notebooks/Classifier%20Training%20HIPPIE.ipynb`

Bibliography

Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M. and Sherlock, G. (2000) ‘Gene Ontology: tool for the unification of biology’, en, *Nature Genetics* 25 (1), pp. 25–29, ISSN: 1061-4036, DOI: `10.1038/75556`, URL: `http://www.nature.com/ng/journal/v25/n1/abs/ng0500_25.html` (visited on 22/04/2014).

Barber, D. (2013) ‘Bayesian Reasoning and Machine Learning’.

Chen, B., Fan, W., Liu, J. and Wu, F.-X. (2013) ‘Identifying protein complexes and functional modules—from static PPI networks to dynamic PPI networks’, en, *Briefings in Bioinformatics*, bbt039, ISSN: 1467-5463, 1477-4054, DOI: `10.1093/bib/bbt039`, URL: `http://bib.oxfordjournals.org/content/early/2013/06/18/bib.bbt039` (visited on 20/03/2014).

Chua, J.J.E., Kindler, S., Boyken, J. and Jahn, R. (2010) ‘The architecture of an excitatory synapse’, en, *Journal of Cell Science* 123 (6), pp. 819–823, ISSN: 0021-9533, 1477-9137, DOI: `10.1242/jcs.052696`, URL: `http://jcs.biologists.org/content/123/6/819` (visited on 23/04/2014).

Gallone, G., Simpson, T.I., Armstrong, J.D. and Jarman, A.P. (2011) ‘Bio::Homology::InterologW - A Perl module to build putative protein-protein interaction networks through interolog mapping’, en, *BMC Bioinformatics* 12 (1), p. 289, ISSN: 1471-2105, DOI: `10.1186/1471-2105-12-289`, URL: `http://www.biomedcentral.com/1471-2105/12/289/abstract` (visited on 09/04/2014).

Hanczar, B., Hua, J., Sima, C., Weinstein, J., Bittner, M. and Dougherty, E.R. (2010) ‘Small-sample precision of ROC-related estimates’, *Bioinformatics* 26 (6), pp. 822–830, URL: http://bioinformatics.oxfordjournals.org/content/26/6/822.short (visited on 05/08/2014).

Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M. and Sakaki, Y. (2001) ‘A comprehensive two-hybrid analysis to explore the yeast protein interactome’, en, *Proceedings of the National Academy of Sciences* 98 (8), pp. 4569–4574, ISSN: 0027-8424, 1091-6490, DOI: 10.1073/pnas.061034498, URL: http://www.pnas.org/content/98/8/4569 (visited on 25/07/2014).

John, G.H. and Langley, P. (1995) ‘Estimating continuous distributions in Bayesian classifiers’, *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., pp. 338–345, URL: http://dl.acm.org/citation.cfm?id=2074196 (visited on 29/07/2014).

Kenley, E.C. and Cho, Y.-R. (2011) ‘Detecting protein complexes and functional modules from protein interaction networks: A graph entropy approach’, en, *PROTEOMICS* 11 (19), pp. 3835–3844, ISSN: 1615-9861, DOI: 10.1002/pmic.201100193, URL: http://onlinelibrary.wiley.com/doi/10.1002/pmic.201100193/abstract (visited on 17/03/2014).

Li, K.W., Klemmer, P. and Smit, A.B. (2010) ‘Interaction proteomics of synapse protein complexes’, en, *Analytical and Bioanalytical Chemistry* 397 (8), pp. 3195–3202, ISSN: 1618-2642, 1618-2650, DOI: 10.1007/s00216-010-3658-z, URL: http://link.springer.com/article/10.1007/s00216-010-3658-z (visited on 20/03/2014).

Mackay, D.J.C. (2003) *Information Theory, Inference, and Learning Algorithms*, ISBN: 0521642981.

Murphy, K.P. (2012) *Machine Learning: A Probabilistic Perspective*, London: The MIT Press, ISBN: 978-0-262-01802-0, URL: http://dl.acm.org/citation.cfm?id=2380985.

Newman, M.E.J. (2012) ‘Communities, modules and large-scale structure in networks’, en, *Nature Physics* 8 (1), pp. 25–31, ISSN: 1745-2473, DOI: 10.1038/nphys2162, URL: http://www.nature.com/nphys/journal/v8/n1/abs/nphys2162.html (visited on 17/03/2014).

Newman, M.E.J. and Girvan, M. (2004) ‘Finding and evaluating community structure in networks’, *Physical Review E* 69 (2), p. 026113, DOI: 10.1103/

PhysRevE.69.026113, URL: http://link.aps.org/doi/10.1103/PhysRevE.69.026113 (visited on 17/03/2014).

Olesen, J., Gustavsson, A., Svensson, M., Wittchen, H.-U., Jönsson, B., CDBE2010 study group and European Brain Council (2012) 'The economic cost of brain disorders in Europe', eng, *European Journal of Neurology: The Official Journal of the European Federation of Neurological Societies* 19 (1), pp. 155–162, ISSN: 1468-1331, DOI: 10.1111/j.1468-1331.2011.03590.x.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011) 'Scikit-learn: Machine learning in Python', *The Journal of Machine Learning Research* 12, pp. 2825–2830, URL: http://dl.acm.org/citation.cfm?id=2078195 (visited on 29/07/2014).

Qi, Y. (2008) 'Learning of protein interaction networks', PhD thesis, Universitat Pompeu Fabra, Spain, URL: http://www-2.cs.cmu.edu/~qyj/paper_CMU/qyj-disertation-v17-final.pdf (visited on 09/04/2014).

Qi, Y., Bar-Joseph, Z. and Klein-Seetharaman, J. (2006) 'Evaluation of different biological data and computational classification methods for use in protein interaction prediction', en, *Proteins: Structure, Function, and Bioinformatics* 63 (3), pp. 490–500, ISSN: 1097-0134, DOI: 10.1002/prot.20865, URL: http://onlinelibrary.wiley.com/doi/10.1002/prot.20865/abstract (visited on 25/03/2014).

Rodgers-Melnick, E., Culp, M. and DiFazio, S.P. (2013) 'Predicting whole genome protein interaction networks from primary sequence data in model and non-model organisms using ENTS', en, *BMC Genomics* 14 (1), p. 608, ISSN: 1471-2164, DOI: 10.1186/1471-2164-14-608, URL: http://www.biomedcentral.com/1471-2164/14/608/abstract (visited on 24/06/2014).

Schaefer, M.H., Fontaine, J.-F., Vinayagam, A., Porras, P., Wanker, E.E. and Andrade-Navarro, M.A. (2012) 'HIPPIE: Integrating Protein Interaction Networks with Experiment Based Quality Scores', *PLoS ONE* 7 (2), ISSN: 1932-6203, DOI: 10.1371/journal.pone.0031826, URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3279424/ (visited on 08/04/2014).

Smedley, D., Haider, S., Ballester, B., Holland, R., London, D., Thorisson, G. and Kasprzyk, A. (2009) 'BioMart – biological queries made easy', en, *BMC Genomics* 10 (1), p. 22, ISSN: 1471-2164, DOI: `10.1186/1471-2164-10-22`, URL: `http://www.biomedcentral.com/1471-2164/10/22/abstract` (visited on 29/07/2014).

Smithson, M. and Verkuilen, J. (2006) 'A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables.', *Psychological methods* 11 (1), p. 54, URL: `http://psycnet.apa.org/journals/met/11/1/54/` (visited on 05/08/2014).

Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A. and Tyers, M. (2006) 'BioGRID: a general repository for interaction datasets', *Nucleic acids research* 34 (suppl 1), pp. D535–D539, URL: `http://nar.oxfordjournals.org/content/34/suppl_1/D535.short` (visited on 02/08/2014).

Uetz, P., Giot, L., Cagney, G., Mansfield, T.A., Judson, R.S., Knight, J.R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S. and Rothberg, J.M. (2000) 'A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae', en, *Nature* 403 (6770), pp. 623–627, ISSN: 0028-0836, DOI: `10.1038/35001009`, URL: `http://www.nature.com/nature/journal/v403/n6770/abs/403623a0.html` (visited on 25/07/2014).

Wang, J., Li, M., Deng, Y. and Pan, Y. (2010) 'Recent advances in clustering methods for protein interaction networks', en, *BMC Genomics* 11 (Suppl 3), S10, ISSN: 1471-2164, DOI: `10.1186/1471-2164-11-S3-S10`, URL: `http://www.biomedcentral.com/1471-2164/11/S3/S10/abstract` (visited on 20/03/2014).

Witten, I.H., Frank, E. and Hall, M.A. (2011) *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*, en, Elsevier, ISBN: 9780080890364.

Xenarios, I., Salwinski, L., Duan, X.J., Higney, P., Kim, S.-M. and Eisenberg, D. (2002) 'DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions', *Nucleic Acids Research* 30 (1), pp. 303–305, ISSN: 0305-1048, URL: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC99070/` (visited on 10/04/2014).