Handling Missing Features with Boosting Algorithms for Protein–Protein Interaction Prediction

Fabrizio Smeraldi[1]*, Michael Defoin-Platel[2], and Mansoor Saqi[2]

[1] School of Electronic Engineering and Computer Science,
Queen Mary University of London,
Mile End Road,
London UK-E14NS
fabri@dcs.qmul.ac.uk
http://www.dcs.qmul.ac.uk

[2] Biomathematics and Bioinformatics,
Rothamsted Research,
Harpenden, UK-AL52JQ
michael.defoin-platel@bbsrc.ac.uk
mansoor.saqi@bbsrc.ac.uk
http://www.rothamsted.bbsrc.ac.uk

Abstract. Combining information from multiple heterogeneous data sources can aid prediction of protein-protein interaction. This information can be arranged into a feature vector for classification. However, missing values in the data can impact on the prediction accuracy. Boosting has emerged as a powerful tool for feature selection and classification. Bayesian methods have traditionally been used to cope with missing data, with boosting being applied to the output of Bayesian classifiers. We explore a variation of Adaboost that deals with the missing values at the level of the boosting algorithm itself, without the need for any density estimation step. Experiments on a publicly available PPI dataset suggest this overall simpler and mathematically coherent approach may be more accurate.

1 Introduction

One of the goals of systems biology is to understand the roles of proteins at various levels of biological organisation, from molecular function through to cellular and physiological function. The identification of networks of interacting proteins is a step to suggesting higher levels of organisation [4]. Experimental data for protein-protein interactions is available for a number of organisms in repositories such as Biogrid [5] and Intact [12]. However protein interaction datasets often contain many false positives [17, 7, 6] and are for most organisms largely incomplete. This has prompted the development of a number of machine learning

* Part of this work was done while F. Smeraldi was visiting Rothamsted Research.

approaches for the prediction of protein-protein interactions. Although information from sequence alone has been used [19, 25], several approaches combine heterogeneous sources of data. Information from such sources can form components of a feature vector associated with a pair of proteins. Given a suitable Gold Standard comprising pairs of proteins that are known to interact and pairs known not to interact, a classifier can be constructed that attempts to discriminate interacting from non-interacting pairs. The features obtained from various data sources are widely heterogeneous and could include, for example, the extent to which the two proteins share similar patterns of co-expression, or the existence of sequence–similar proteins known to interact in another organism. The performance of the classifier can be assessed and the classifier can then be used to predict interactions between pairs of proteins for which no experimental information is yet available. Previous studies have constructed predicted interactomes for various organisms including yeast, *Arabidopsis* and human, using machine learning methods such as Bayes classifiers [11, 15, 24] and Support Vector Machines [3, 25, 13]. However, experimental measurements of each feature are usually available for different subsets of proteins pairs. Therefore, as more information is combined from multiple heterogeneous sources, the way that missing observations are treated becomes increasingly important. Arguably, this somewhat limits the available choices of classifiers, unless non-trivial pre-processing steps (typically Bayesian in nature) are applied. For instance, kernel methods offer no straightforward way to handle missing data, as the implicit mapping carried out by the kernel depends non-linearly on all components of the input vector (see for instance [20]); typical applications of ensemble methods such as AdaBoost [8] also rely on preprocessing.

In [11] a Naive Bayesian approach was used to predict protein-protein interactions in yeast by integrating a few genomic features. A larger number of features as well as a boosted classifier were used to assess the limits of data integration in [15]. Here we explore a variant of the Adaboost algorithm that is able to deal explicitly with missing feature values. Our approach matches or betters the results obtained by Bayesian preprocessing and is overall easier to implement. More importantly, it deals away with the arbitrariness inherent in density estimation and fits into the same solid theoretical framework as AdaBoost, especially with regard to convergence guarantees.

1.1 Dealing with missing features

Several strategies have been proposed for dealing with missing features (for an overview, see [20]). At least two typologies of approaches have been applied to the database we study (see Section 3). The first, most obvious strategy is simply to discard the examples for which data is missing. This approach has been investigated in [14], but complete information for each feature was only available for a small subset of the protein pairs.

Another classical technique that can be employed to deal with missing data is to complement the dataset with an estimate or a default value when a particular feature is missing. One way in which this has been done is by learning Naive

Bayes classifiers from (subsets of) the raw features [11]. Since Naive Bayes classifiers can fall back on the prior, missing data are no longer an issue (cf Section 6.2); a more advanced classifier can then be applied to their output [15].

However, training a Naive Bayes classifier involves a density estimation step. Density estimation is an ill–posed and difficult problem [9]; the variety of the techniques available (Parzen windows, histograms, Expectation Maximisation to name a few) is indicative of the element of arbitrariness that this process entails. Indeed, some of the most successful classification algorithms such as Support Vector Machines [26] and Adaboost [8] owe their effectiveness at least in part to the fact that they avoid estimating densities for their input, but rather optimise the margin or a bound on the empirical error; from this point of view, introducing a preliminary density estimation step seems somewhat incongruous.

In this work we investigate an effective alternative to Naive Bayes classifiers for dealing with missing feature values when Adaboost [8] is used. Adaboost is an adaptive strategy for combining multiple binary classifiers with slightly better–than–random performance (the so-called *Weak Learners*) into a highly accurate *strong classifier*. The algorithm works by maintaining a list of weights over the training examples, so that "difficult" examples (that are misclassified by many Weak Learners) become more important over time. Adaboost iteratively chooses the optimal Weak Learner (WL) with respect to the weights, adds it to the strong classifier with an appropriate coefficient, and updates the weights of the training examples (for details, see Section 2). When each WL is a function of a single feature (for instance a simple threshold on the feature value, also known as a *decision stump*), Adaboost iterations essentially perform feature selection.

Training Bayesian classifiers on the raw feature values and then boosting them [15] amounts to dealing with missing data in the WLs, so that the missing values are hidden from Adaboost. The alternative we here explore is to use WLs that *abstain* (i.e. do not return a decision) on missing data, and let the boosting algorithm deal with the problem.

Although variants of AdaBoost for WLs that abstain have been introduced early on [23], they have not been as widely applied as the standard algorithm; specifically, to the best of our knowledge, they have not yet been applied to PPI data. One of the reasons for this may be that these algorithms have been introduced in the slightly different context of confidence-rated predictions, i.e. under the assumption that the WLs provide a graded output instead than a binary decision. Also, while a considerable computational simplification of standard Adaboost has made the implementation of the algorithm straightforward [2], to the best of our knowledge this has not yet been generalised to the case of WLs that abstain.

In this paper we introduce in detail the simplified version of the algorithm for classifiers that abstain and we apply it to a widely investigated set of features for PPI in yeast [15]. Our results show that avoiding density estimation and dealing with missing features at a late stage may indeed be the better option, at least when Boosting algorithms are used.

```
Adaboost for weak learners that abstain (simplified):

Input:
  1. Labelled training vectors $\{(\boldsymbol{x}_i, y_i)\}$, with $y_i \in \{-1, +1\}$.
  2. Weak learners $\{h_j(x) : \mathsf{X} \to \{-1, 0, +1\}\}$.

Initialise the weights $d_{1,i} = \frac{1}{m}$.

For $t = 1, \ldots, T$:
  1. Train each weak learner $h_j(\boldsymbol{x})$. Using the current distribution of weights $d_{t,i}$, compute the total weight of the training examples on which it abstains ($W_a$), that it classifies correctly ($W_c$) and that it misclassifies ($W_m$).
  2. Select the weak learner $h_t$ that minimises $Z = W_a + 2\sqrt{W_c W_m}$. Check that $Z < 1$; otherwise quit.
  3. Compute $\alpha_t = \frac{1}{2}\log(W_c/W_m)$ and update the weights:
     $$d_{t+1,i} = \begin{cases} d_{t,i}/(W_a\sqrt{W_m/W_c} + 2W_m) & \text{if } h_t \text{ misclassifies } \boldsymbol{x}, \\ d_{t,i}/(W_a\sqrt{W_c/W_m} + 2W_c) & \text{if } h_t \text{ classifies } \boldsymbol{x} \text{ correctly}, \\ d_{t,i}/Z & \text{if } h_t \text{ abstains on } \boldsymbol{x}. \end{cases}$$

Output the strong classifier: $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$
```

Table 1. Simplified version of AdaBoost for weak learners that abstain (Ada-ABS). This algorithm is equivalent to the one introduced in [23] and reduces to standard Adaboost in the case of weak learners that never abstain, as is the case with Naive Bayes classifiers.

2 Boosting weak learners that abstain

We use the AdaBoost algorithm for classifiers that abstain given in [23], since the choice of parameters in this algorithm guarantees the tightest bound on the training error. However, instead of following the traditional formulation of Adaboost, we extend to the case of classifiers that abstain the simplification presented in [2]. As shown in [16], such simplification can be seen as a direct solution of the dual formulation of the Adaboost minimisation problem. Extension to the case of WLs that abstain is straightforward, and leads to the algorithm outlined in Table 1 (henceforth Ada-ABS). We emphasise that this simplified version computes, step by step, the same weights and the same final decision function as specified in [23]. The main advantage is its simplicity; by comparison, in the original formulation of Adaboost [8] the weight update procedure is

$$d_{t+1,i} = \frac{1}{Z_t} d_{t,i} \exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right), \tag{1}$$

with

$$Z_t = \sum_i d_{t,i} \exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right) \tag{2}$$

The algorithm for classifiers that abstain is very similar to classical Adaboost, with the following main differences:

1. the value of α is $\log(W_c/W_m)/2$ (different variants of the algorithm specify other choices for α);
2. at each iteration the weak learner that minimises Z is chosen, as opposed to the learner that minimises the weighted classification error W_c.

The reason for points 2. and 3. above is that, at each iteration, Z multiplies a bound on the training error; the choice of α minimises Z given W_c and W_m. When $W_a = 0$, i.e. the weak learner never abstains, this is equivalent to the standard choice $\alpha = 1/2\log((1-W_m)/W_m)$. It is easy to see that if no weak learners abstain also the weight update equations, and hence the entire algorithm, revert to standard Adaboost.

Notice that, since $Z = W_a + 2\sqrt{W_c W_m}$, weak learners that abstain on a large number of training examples (or on training examples carrying a large weight) are penalised irrespective of their performance on the examples on which they do provide a decision. Also, since $Z_t < 1$, training examples on which a weak learner abstains see their weight increased for the following iteration. Therefore, the algorithm will eventually select one or more weak learners to classify each of the training examples.

3 Dataset

In this study we use a Gold Standard dataset[3] for the prediction of protein-protein interactions as described in [11] and [15]. This is based on the MIPS (Munich Information Centre for Protein Sequences) hand curated catalogue. We report experimental results over a subset of 3161 proteins, as already described in [15]. This subset contains 2,711,441 interacting or non-interacting protein pairs, for which at least one of the 16 genomic features defined in Table 2 is available. For a complete definition of the features and of their relationship to protein-protein interactions see [15]. In the whole dataset, a sample consists of two protein names, 16 genomic features and a label that specifies if the proteins interact or not. The distribution of the features is shown in Figure 1. The dataset has a large number of missing features, even after discarding from the dataset the samples for which no data at all is available; the percentage of missing data by feature is shown in Table 2.

4 Results

We compare the performance of the three *strong classifiers* obtained by boosting three different sets of WLs. These are implemented as threshold-based classifiers

[3] Available online at http://networks.gersteinlab.org/intint

Features	Description	Missing data (%)			IG Rank	
		Positive	Negative	Total	Orig.	Compl.
F1	mRNA Co-expression	7.71	1.03	1.05	7	4
F2	MIPS Functional Similarity	2.41	51.41	51.26	6	1
F3	GO Functional Similarity	8.85	76.06	75.86	5	2
F4	Co-Essentiality	73.85	78.78	78.76	8	8
F5	Absolute mRNA Expression	5.62	0.27	0.28	9	5
F6	Marginal Essentiality	6.21	4.25	4.26	10	7
F7	Absolute Protein Abundance	37.07	43.97	43.95	11	11
F8	Co-regulation	52.15	97.79	97.65	16	16
F9	Phylogenetic Profiles	88.92	99.03	99.00	4	6
F10	Gene Neighbourhood	96.22	99.96	99.95	3	10
F11	Rosetta Stone	98.63	99.95	99.95	2	9
F12	Synthetic Lethality	98.75	99.97	99.97	15	15
F13	Gene Cluster	99.98	99.98	99.98	14	14
F14	Threading Scores	98.75	99.96	99.95	13	13
F15	Co-evolution Scores	99.98	99.99	99.99	12	12
F16	Interologs in other Organism	54.65	99.85	99.71	1	3

Table 2. Percentage of missing values by feature over the $8,250$ interacting protein pairs, $2,703,191$ non-interacting pairs (and $2,711,441$ pairs total) listed in the Gold Standard after removing the pairs for which no feature values at all are given. Also shown is a rank of the feature according to information gain [18] for the original dataset and a dataset complemented by the mean.

(*decision stumps*), a common choice for WLs (see Section 6.1). The WLs are trained (*i*) on the log odds of interaction for a particular feature (Bayes WLs), (*ii*) on the raw feature values (WL-Abs) or (*iii*) on the raw feature values augmented by the Naive Bayes score (Section 6.2), considered as an additional feature (WL-Abs+Naive Bayes). WLs trained on the raw feature values abstain (return a value of zero) when the feature value is missing. We use the Ada-ABS algorithm outlined in Table 1. It should be noted that, as detailed in Section 2, Ada-ABS reverts to standard Adaboost when the weak learners do not in fact abstain.

A Naive Bayes Classifier (NBC) trained on all features is used to give a baseline performance. A common objection to boosting approaches based on decision stumps is that correlations between feature values or higher order statistical moments that might be exploited, for instance, by a Support Vector Machine are treated very poorly, as each WL bases its binary decision on a single feature.

As a coarse estimate of this limitation, we augment the set of WLs that abstain with a decision stump based on the posterior estimate provided by the baseline NBC. As we will see, when the percentage of missing data is limited this combination of WLs that abstain with an NBC improves performance well beyond the level achieved by the classical approach of applying Bayesian estimation to the single features separately and then boosting them. As the percentage

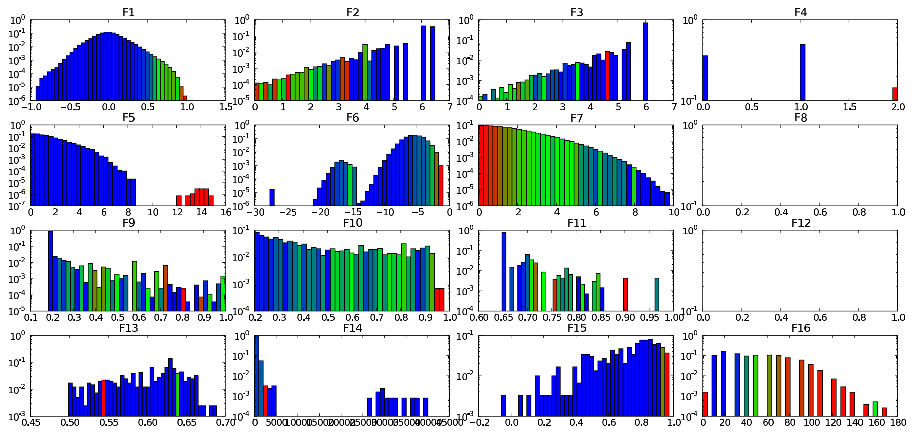

Fig. 1. Distributions of the feature values (on a log scale). Colour indicates the fraction of interacting protein pairs in each bin. Blue indicates that the bin mainly corresponds to non-interacting pairs; red bins predominantly account for interacting pairs.

EER (%)	All features	F1–F4	F5–F16	w/o F2,F3
Naive Bayes Class.	5.0%	5.3%	27.3%	12.2%
Bayes WLs	3.2%	3.5%	18.6%	13.0%
WLs that Abstain	3.0%	3.3%	**17.0%**	**11.7%**
WL-Abs + NBC	**2.0%**	**2.5%**	**17.0%**	12.2%

Table 3. Summary of the EER for different choices of classifiers and of features, averaged over 4 cross-validation rounds. The database and the first two algorithms listed were explored in [11, 15]. Bold values show the minimal EER by set of features. Boosting WLs that abstain is always better than both Naive Bayes and the boosted Bayesian WLs. Complementing the WLs that abstain with the NB classifier can improve performance when multiple observations with low percentages of missing data are used.

of missing data increases our results suggest that the NBC becomes less informative; weak learners that abstain still provide optimal results in this case.

4.1 Classification accuracy

We perform a series of four-fold cross-validation experiments on the dataset described in Section 3 with an increasing number n of boosting iterations, using different subsets of the features listed in Table 2. A schematic of the results is given in Table 3.

Figure 2 shows the Equal Error Rate averaged across the four cross-validation runs as a function of n, when all the 16 features are used for training.

The minimum error rate when boosting Bayes classifiers is in this case achieved after 207 iterations (average EER=3.2%). For WLs that abstain, a slightly lower

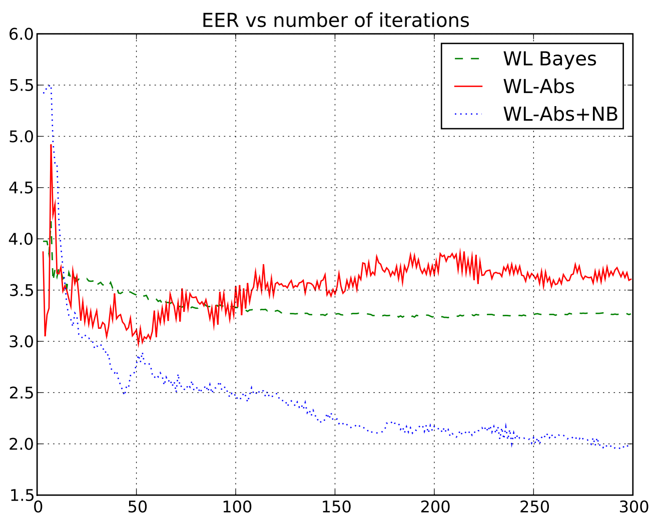

Fig. 2. Equal Error Rate as a function of the number of iterations for AdaBoost using Bayes WLs, WLs that abstain and the combination of WL-Abs and the Naive Bayes Classifier trained on all features.

EER is obtained after 51 iterations (average EER=3.0%). If we consider the estimate of the NBC as a new feature and boost it with Ada-ABS together with the WLs that abstain, the average EER is lowered significantly to 2.0% (at 293 iterations). This is more that 30% better than boosting the Bayes WLs and less than half the average EER of the baseline Naive Bayes Classifier (which is 5.0%).

The ROCs corresponding to the optimal average EERs listed above are displayed in Figure 3, for all four cross-validation rounds separately. As can be seen, all boosting approaches widely improve on the Naive Bayes. Weak Learners that abstain yield slightly better performance than boosted Bayes WLs, with a more marked advantage in the high sensitivity part of the curve. The best ROC curves are obtained by adding the Naive Bayes Classifier as a weak learner, arguably because this captures the correlation between the features — a point that is confirmed by our results in Section 4.3 below.

In predictive usage, one would need to set the number of iterations using a validation set and extrapolate. This applies both to Adaboost with Bayes WLs and to the algorithms we introduce. Convergence and generalisation ability of Adaboost as a function of the number of iterations is a widely investigated topic [22, 21], that is largely beyond the scope of this work. Empirically from Figure 2 we notice that, after the decision function stabilises in the first 30 iterations, the boosted classifiers are fairly insensitive to the choice of the number of iterations: the average EER oscillates in a band narrower than 1% as n is increased from 30 to 300. However, Ada-ABS has a slight tendency to over-fit

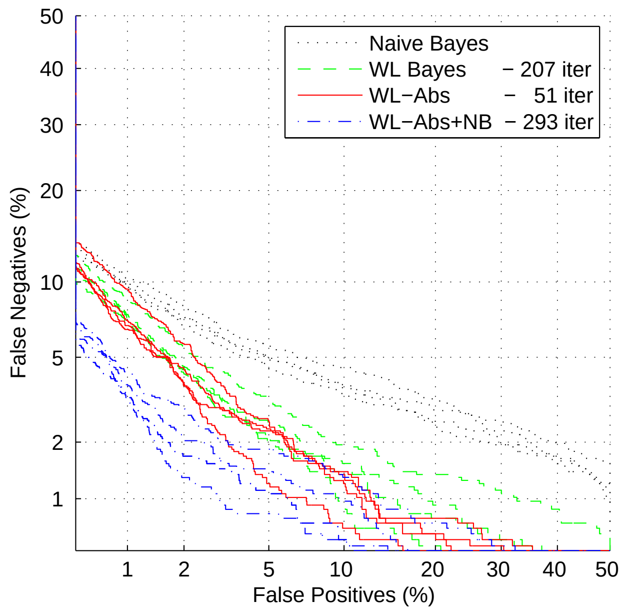

Fig. 3. Superposed ROC curves for all 4 cross validation rounds. The curves displayed correspond to Bayes WLs (207 iterations), WLs that abstain (51 iterations) and a combination of WL-Abs and the Naive Bayes classifier (293 iterations).

after about 60 iterations, while Bayesian WLs are more stable. This is likely due to the fact that density estimation with a histogram (Section 6.2) effectively reduces the choices for the weak learners by limiting the number of different thresholds available for each feature to the number of bins in the histogram. In turn, this seems to regularise the decision function, albeit at the expense of classification accuracy.

However, it should be noted that while the curves in Figure 2 tell the entire story for WL-Abs, of which the number of iterations is the only parameter, Adaboost with Bayesian (weak) classifiers includes in addition many more parameters hidden in the density estimation step; this should be kept into account when assessing the relative stability of the algorithms.

4.2 Structure of the decision function

In Table 4 we list the features that contribute to the decision function for each of the ROC curves in Figure 3. Features are listed in the order in which the corresponding WLs are first selected by the boosting algorithm; the number of times a particular feature is chosen (possibly with different thresholds) is indicated in parentheses.

Classifier	**Weak learners (occurrences)**
WLs that Abstain (51 iter.)	F5(3,5,6) – F2(15,18,19) – F1(7,9,11) – F3(4,5,7) – F6(5,9,7,11) – F16(2) – F4(2),F7(1,2) – F8,12(1), F4(2) – F8,12(1) – F7,8(0,1) – F14(0,1)
Bayes WLs (207 iter.)	F16(31,33) – F2(30,37,38) – F3(25,27,30) – F1(19,23,27,34) – F5(17,18,25,34) – F4(6,7,9) – F6(20,24,29,30) – F8(4,5) – F7(3,4,6,7) – F12(9,13-15) – F14(0,2,6,7) – F11(0,3) – F9(0,1,2)
WLs that Abstain plus Naive Bayes (293 iter.)	NB(64,78,80) – F1(10,16,25) – F2(91,108,109,112) – F5(11,18,19), F16(2) – F5(28),F16(1,2) – F3(37,40-42) – F4(2),F6(21,25,30) – F4(2,3),F6(18) – F8(2),F12(1) – F8(1,2),F12(1) – F14(2,3),F7(8) – F7(5,10),F9(2),F11(1) – F7(1),F9(1,2),F11(1) – F11(1),F14(0,1)

Table 4. Features used by the WLs appearing in the decision functions that yield the ROC curves in Figure 3. WLs are listed in order of first appearance. The number of times each feature is chosen is given in parentheses. Multiple comma–separated entries correspond to variations across the four cross-validation runs. NB indicates the WL corresponding to the Naive Bayes Classifier.

As can be seen, the order in which the features are chosen is fairly stable for each set of WLs, with variations between the cross-validation rounds occurring for higher numbers of iterations. By comparison with Table 2 we see that the first few features are chosen among the most discriminative features and among those with the lowest percentage of missing values. Feature F5, the first feature when using Weak Learners that Abstain, has the lowest total percentage of missing data, while F16 (the first Bayesian WL) has the third highest information gain on the complemented dataset (and the highest on the original dataset).

Probably as an effect of the unbalance between positive and negative training examples, overlap with the positive training data seems to be more important than the coverage of the set of non-interacting pairs. Feature 16 for instance has 99.71% missing data but covers about 45% of the positive training set. Similarly Feature 3 has 75.86% of missing data overall, but only just short of 9% of the positive training data are not covered. Conversely, features 10, 13 and 15 do not appear at all in the optimal decision functions (except insofar as they contribute to the NBC). As shown in Table 2, these features are unavailable for at least 99.95% of the Gold Standard data, with very limited coverage of both interacting and non-interacting pairs.

Table 4 provides a picking order, but it does not specify the weight of each feature in the decision function nor the dependence on the feature value. These data are displayed in Figure 4 for the decision function resulting from 51 iterations of Ada-ABS on WLs that abstain.

More specifically, with reference to the strong decision function $H(\boldsymbol{x})$ defined in Table 1, for each feature j the figure shows

$$H_j(\boldsymbol{x}) = \sum_{t \in \mathsf{S}_j} \alpha_t h_t(x_j) \tag{3}$$

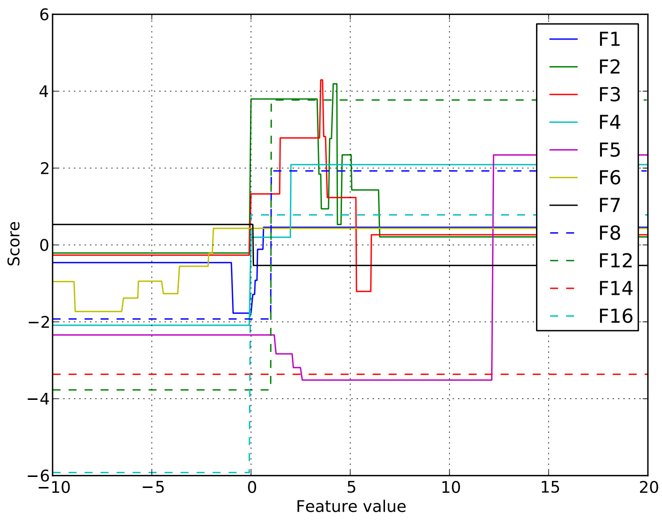

Fig. 4. Contribution of the single features to the decision function for Ada-ABS, 51 iterations. F6 has an additional threshold at -27.63, F14 at +287.00, and F16 at +50.00.

where S_j is the set of iterations in which a weak learner that is a function of feature j has been chosen . As can be seen, the functions H_J corresponding to the 11 distinct features all have similar range, suggesting that they all contribute significantly to the final decision. This confirms that the classifier can effectively extract information from features with high percentages of missing data.

Figure 4 also shows how boosting can give different weights to specific intervals of feature values by combining simple step-wise weak learners with different thresholds. The type of these intervals (open or close) also depends on the parity of the number of weak learners for the specific feature. As the first row of Table 4 shows, a few features have a marked preference for an odd or an even number of occurrences. This explains the jagged appearance of the EER curves for WLs that abstain in Figure 2 .

4.3 Exploring subsets of features

According to [15], near-optimal performance should be achieved in the case of boosting Bayes WLs even when only the first four features are used (mRNA co-expression, MIPS functional similarity, GO functional similarity and Co-essentiality). We test this by limiting the boosting algorithm to select among weak learners trained on the first four features only (both for Bayes WLs and WLs that abstain). The resulting average EERs are reported in the third column of Table 3. Indeed, for both the baseline Naive Bayes Classifier and boosted Bayes WLs, the average EERs increase by less than 10%. The lowest average

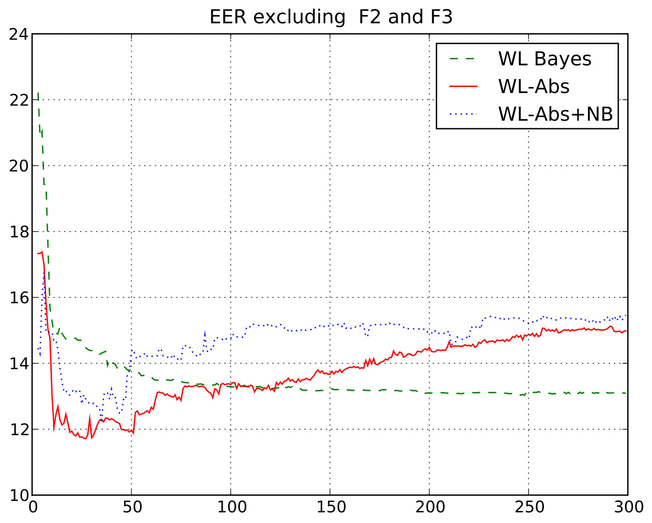

Fig. 5. Equal Error Rate as a function of the number of iterations when boosting over all features except for F2 and F3, that are based on functional similarity.

EER of 2.5%, however, is obtained using a combination of WLs that abstain with a WL that thresholds the NBC; this is less than half the EER of the baseline NBC (5.3%), and actually improves on both the NBC and boosted Bayes WLs even when those are allowed to use all features (see the second column of Table 3). It should be noted that, when we allow our algorithm to use all features, the EER is reduced by a further 25%. This suggests that F1 to F4 do not actually capture all the information available, and that our approach is better at extracting information from the remaining features (F5 to F16).

To further test the behaviour of our algorithms on features that mostly have high percentages of missing data and have previously been found to be less informative, we perform cross-validation tests on the database using features F5 to F16 only. Average EERs are displayed in the fourth column of Table 3. As can be seen, WLs that abstain outperform the other approaches. Adding a WL based on the Naive Bayes classifier does not in this case improve accuracy; this is, in our view, a consequence of the high number of missing feature values and of the consequent overall poor performance of the NBC itself.

Following [15] we also explore the effect of removing two strong features, namely F2 and F3, that are based on functional similarity and dominate the prediction performance. Since the accurate assignment of a functional category to a protein generally involves a manual curation step, prediction performance without these features may also better reflect a more general application case.

In Figure 5 we report the average EER as a function of the number of iterations after removing features F2 and F3 from the database. The minimum

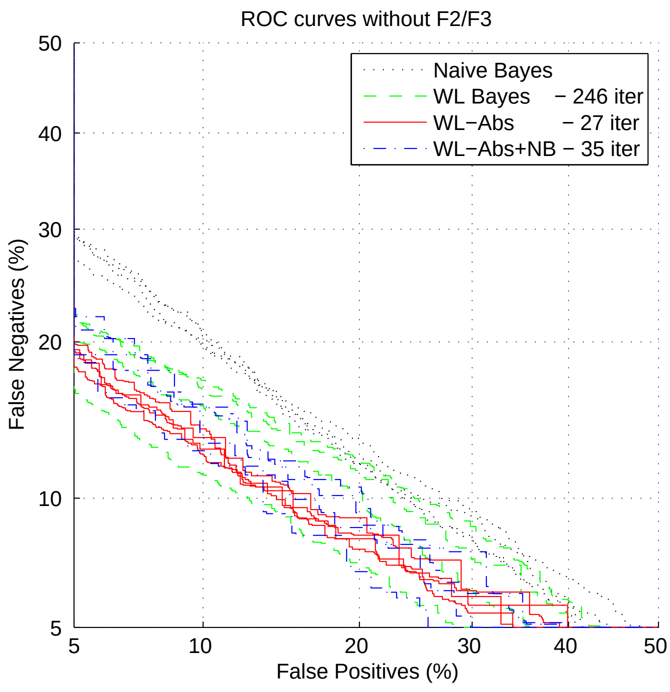

Fig. 6. ROC curves for Naive Bayes classifiers (33 iterations) and Raw Features (27 iterations). Curves for the four cross-validation rounds are superposed. Note how boosting the Raw Features leads to higher precision as well as increased stability.

average EER for boosted Bayes WLs is 13.0%, while for WLs that abstain it is 11.7% (see Table 3, last column). These values are achieved after 246 and 27 iterations respectively. The average EER of the baseline Naive Bayes Classifier is 12.2%; adding its output as a WL to the WLs that abstain does not in this case improve accuracy. The corresponding ROC curves for all four cross-validation round and the best choice of the number of iterations are displayed in Figure 6.

5 Conclusions

The integration of multiple heterogeneous data sources is important in systems biology as it has the potential to impact on prediction. In order to exploit information that may be available only on parts of the dataset, effective strategies for handling missing data are required.

We explore the use of a variant of Adaboost (originally derived in the context of confidence–rated predictions) that allows the weak learners to “abstain” , i.e. not to return a decision whenever a feature value is missing. We introduce an exact simplified mathematical formulation of this algorithm along the lines of an existing simplification of AdaBoost. The simplified algorithm is hardly more complicated to implement than Adaboost, and reverts to standard Adaboost when no data are missing.

We test our algorithm over different subsets of features from a widely used database of yeast PPI data, that includes measurements with high numbers of missing features. Experimental results show that our approach can handle large percentages of missing data effectively, consistently outperforming the common alternative of boosting single-feature Bayesian classifiers. Besides being overall easier to implement, our approach deals away with the theoretical and practical complications of density estimation.

Our results also indicate that a Naive Bayes classifier trained on all features, although overall far less effective on its own, may capture correlations between the features that escape an approach based on decision stumps; when this happen, this information can be integrated in the boosting framework by considering the Naive Bayes classifier as an additional weak learner. In future work we propose to investigate the use of other choices for the weak learners that may better account for dependency between the features.

6 Appendix

6.1 Weak learners that abstain

We use decision stumps that abstain to deal with missing feature values in the dataset. More in detail, given training vectors $\boldsymbol{x}$ with missing components we define a weak learner for each of the N features (components) in the following way:

$$h_j(\boldsymbol{x}) = \begin{cases} +p_j & \text{if } x_j \geq \tau_j \\ -p_j & \text{if } x_j < \tau_j \\ 0 & \text{if } x_j \text{ is missing.} \end{cases} \tag{4}$$

The "polarity" constant $p_j \in \{-1, +1\}$ allows us to cater for features representing similarities and dissimilarities in the same framework. Both p_j and τ_j are optimised at each iteration to minimise Z as defined in Table 1.

6.2 Naive Bayes classifiers

A Naive Bayes Classifier is based on the Bayes Rule and on the assumption that the features are conditionally independent given the class. It assigns the most probable label value $\hat{y}$ to the vector $\boldsymbol{x} = \{x_1, \dots, x_N\}$ according to:

$$\hat{y} = \operatorname{sign}\left(\frac{P(+1)}{P(-1)} \prod_j^N \frac{P(x_j| + 1)}{P(x_j| - 1)} - \tau \right) \tag{5}$$

where $P(y)$ is the prior probability of class y, $P(x_j|y)$ are the class-conditional densities and τ is a threshold used to set the operation point of the classifier.

NBCs provide a straightforward way of dealing with missing feature values: when a given feature x_j is missing the corresponding likelihood ratio $P(x_j|+1)/P(x_j|-1)$ in Equation 5 is set to 1. In the limit case that all the features of $\boldsymbol{x}$ are missing, the classifier can still return a prediction relying on the *a-priori* odds $P(+1)/P(-1)$.

Arguably the most critical part in Naive Bayes classification is the computation of robust estimates for the densities $P(y)$ and $P(x_j|y)$. In this study, the prior probabilities $P(y)$ are estimated as the proportion of each class in the training set. The conditional probabilities $P(x_j|y)$ are more difficult to estimate, mainly because many of the the x_j are continuous features. The seminal work of [10] has shown that a simple discretization technique by histograms can be used for this dataset. When known (see for example [1]), the numbers and ranges of the bins used in former studies have been employed in this work; otherwise, similar binning strategies have been used.

In order to obtain smooth estimates, the m-estimate method was used with $m = 2$ to compute the probabilities from frequencies of labels according to

$$P(x_j = b|y) = \frac{\text{count}(b, y) + mP(y)}{\text{count}(b) + m} \tag{6}$$

where $\text{count}(b, y)$ is number of samples with label y in bin b, $\text{count}(b)$ is the total number of samples in bin b and $P(y)$ is the prior probability of label y.

In this study, the classifier defined by Equation 5 is referred to as the Naive Bayes Classifier. The single–feature Bayes WLs are obtained by thresholding the likelihood ratio $P(x_j|+1)/P(x_j|-1)$ for each individual feature $x_j \in \boldsymbol{x}$.

References

1. Azuaje, F., Dopazo, J.: Data Analysis and Visualization in Genomics and Proteomics. John Wiley & Sons (2005)
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Machine Learning 36(1–2), 105–139 (July 1999)
3. Ben-Hur, A., Noble, W.S.: Kernel methods for predicting protein-protein interactions. Bioinformatics 21 Suppl 1, i38–46 (2005)
4. Bork, P., Jensen, L.J., von Mering, C., Ramani, A.K., Lee, I., Marcotte, E.M.: Protein interaction networks from yeast to human. Curr Opin Struct Biol 14(3), 292–9 (2004)
5. Breitkreutz, B.J., Stark, C., Reguly, T., Boucher, L., Breitkreutz, A., Livstone, M., Oughtred, R., Lackner, D.H., Bhler, J., Wood, V., Dolinski, K., Tyers, M.: The bioGRID interaction database: 2008 update. Nucleic Acids Res 36(Database issue), D637–D640 (Jan 2008)
6. Deane, C.M., Salwiski, L., Xenarios, I., Eisenberg, D.: Protein interactions: two methods for assessment of the reliability of high throughput observations. Mol Cell Proteomics 1(5), 349–356 (May 2002)
7. Edwards, A.M., Kus, B., Jansen, R., Greenbaum, D., Greenblatt, J., Gerstein, M.: Bridging structural biology and genomics: assessing protein interaction data with known complexes. Trends Genet 18(10), 529–536 (Oct 2002)

8. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Science 55(1) (1997)
9. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2001)
10. Jansen, R., Greenbaum, D., Gerstein, M.: Relating whole-genome expression data with protein-protein interactions. Genome Res 12(1), 37–46 (Jan 2002)
11. Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N.J., Chung, S., Emili, A., Snyder, M., Greenblatt, J.F., Gerstein, M.: A Bayesian networks approach for predicting protein-protein interactions from genomic data. Science 302(5644), 449–453 (Oct 2003)
12. Kerrien, S., Alam-Faruque, Y., Aranda, B., Bancarz, I., Bridge, A., Derow, C., Dimmer, E., Feuermann, M., Friedrichsen, A., Huntley, R., Kohler, C., Khadake, J., Leroy, C., Liban, A., Lieftink, C., Montecchi-Palazzi, L., Orchard, S., Risse, J., Robbe, K., Roechert, B., Thorneycroft, D., Zhang, Y., Apweiler, R., Hermjakob, H.: Intact–open source resource for molecular interaction data. Nucleic Acids Res 35(Database issue), D561–D565 (Jan 2007)
13. Lin, M., Hu, B., Chen, L., Sun, P., Fan, Y., Wu, P., Chen, X.: Computational identification of potential molecular interactions in Arabidopsis. Plant Physiol 151(1), 34–46 (Sep 2009)
14. Lin, N., Wu, B., Jansen, R., Gerstein, M., Zhao, H.: Information assessment on predicting protein-protein interactions. BMC Bioinformatics 5, 154 (Oct 2004)
15. Lu, L.J., Xia, Y., Paccanaro, A., Yu, H., Gerstein, M.: Assessing the limits of genomic data integration for predicting protein networks. Genome Res 15(7), 945–953 (Jul 2005)
16. Malacaria, P., Smeraldi, F.: On Adaboost and optimal betting strategies. In: Proceedings of the 5th international conference on data mining (dmin/worldcomp). pp. 326–332. CSREA Press (July 2009)
17. von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., Bork, P.: Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417(6887), 399–403 (May 2002)
18. Michalski, R.S., Carbonell, J.G., Mitchell, T.M.: Machine Learning: An Artificial Intelligence Approach. Tioga Publishing Company (1983)
19. Najafabadi, H.S., Salavati, R.: Sequence–based prediction of protein–protein interaction by means of codon usage. Genome Biology 9(5) (2008)
20. Pelckmans, K., Brabanter, J.D., Suykens, J.A.K., Moor, B.D.: Handling missing values in support vector machine classifiers. Neural Networks 18, 684–692 (2005)
21. Rätsch, G., Warmuth, M.: Efficient margin maximizing with boosting. Journal of Machine Learning Research 6, 2131–2152 (2005)
22. Rudin, C., Schapire, R.E., Daubechies, I.: On the dynamics of boosting. In: Advances in neural information processing systems. vol. 16 (2004)
23. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3) (1999)
24. Scott, M.S., Barton, G.J.: Probabilistic prediction and ranking of human protein-protein interactions. BMC Bioinformatics 8, 239 (2007)
25. Shen, J., Zhang, J., Luo, X., Zhu, W., Yu, K., Chen, K., Li, Y., Jiang, H.: Predicting protein-protein interactions based only on sequences information. Proc Natl Acad Sci U S A 104(11), 4337–4341 (Mar 2007)
26. Vapnik, V.N.: The nature of statistical learning theory. Springer–Verlag (1995)