Redundant Feature Selection using Permutation Methods

Phillip Taylor PHIL@DCS.WARWICK.AC.UK
Nathan Griffiths NATHAN@DCS.WARWICK.AC.UK
Abhir Bhalerao ABHIR@DCS.WARWICK.AC.UK
The University of Warwick, CV4 7AL, UK

Abstract

Automatic feature selection aims to select the features with highest performance when used in a classifier. One popular measure for estimating feature relevancy and redundancy is Mutual Information (MI), although it is biased toward features with multiple values. Permutation methods have been successfully applied in normalizing for numerous biases including that of MI; however they are computationally expensive and complete redundancy computation is infeasible. In this paper, we introduce a measure that can be used to approximate all m^2 redundancies between m features, while performing only m permutation methods for their relevancies. We then show using simulated data that this permutation redundancy measure holds similar properties to normalized MI and apply it in selecting features from example datasets using minimal Redundancy Maximal Relevancy (mRMR).

1. Introduction

Feature selection aims to select features that provide the highest performance when used in models and is an integral part of the machine learning and data mining processes (Kohavi and John, 1997; Guyon and Elisseeff, 2003). Relying on human experts to select good features is often sub-optimal because of human error or personal biases, and so efforts have been made to automate the feature selection process. One popular filter for feature selection is minimal Redundancy Maximal Relevance (mRMR) (Peng et al., 2005), although several other filter methods exist (e.g. (Li et al., 2008; Chen, 2011)). In many implementations these selection methods use Mutual Information (MI) as a measure of both relevancy and redundancy (Herman et al., 2013), although it is biased towards features with multiple values (Jensen and Cohen, 2000). One way to reduce this bias is to normalize MI by entropy, as in Symmetric Uncertainty (SU) (Witten and Frank, 2011; Taylor et al., 2012), but this is imperfect since it does not account for other potential biases in the data or feature selection process. An alternative approach is to use permutation methods to normalize for biases (Good, 2000; Altmann et al., 2010), but they are computationally very expensive and a complete redundancy analysis between all features is infeasible. In this paper, we introduce a measure based on permutation methods that can be used to approximate all m^2 normalized redundancies while performing only m permutation methods for the relevancies, where m is the number of input features.

2. Background

The permutation method is a statistical test that can be used to assign a significance to a correlation between two variables such as $MI(x, y)$, or to normalize it for biases (Good, 2000; Hapfelmeier and Ulm, 2013). It operates by computing the correlation statistic for several different permutations (1000 in this paper) of the variables. Permuting either variable gives the same permutation distribution, and it is computationally more efficient to permute the class labels when computing relevancies rather than each individual feature.

The significance, or p-value, is the proportion of the permutation distribution that is at least as large as $MI(x, y)$. Rather than compute a p-value, Wang et al. (2009) assume that the distribution is normally distributed and use the standard score of $MI(x, y)$,

$$\mathrm{Z_{MI}(x, y)} = \frac{MI(x, y) - \{y' \in \Psi(y) : MI(x, y')\}_\mu}{\{y' \in \Psi(y) : MI(x, y')\}_\sigma}, \tag{1}$$

where $\Psi(y)$ is the set of computed permutations of y, and μ and σ represent the mean and standard deviation. Radivojac et al. (2004), use Z_{MI} to rank features with p-values below a threshold. The other features are ranked by their p-value and below those ranked by Z_{MI}.

A popular filter for feature selection introduced by Peng et al. (2005) is mRMR. In general, mRMR aims to maximize the difference or ratio between the mean relevancy, $Rel(\cdot)$, and redundancy, $Red(\cdot)$, of selected features (Herman et al., 2013). In this paper we use a forward greedy search to select the feature that satisfies,

$$\max_{x \in X \setminus S} Rel(x_i, y) - \frac{1}{|S \cup \{x\}|^2} \sum_{x_i, x_j \in S \cup \{x\}} Red(x_i, x_j), \tag{2}$$

where $X = \{x_0, x_1, \ldots, x_n\}$ is the complete set of input features, y is the target variable, and $S \subset X$ is the set of currently selected features. Where both $Rel(\cdot)$ and $Red(\cdot)$ is given by MI, it is referred to as $MImRMR$ in this paper.

3. A redundant permutation feature selector

Using Z_{MI} as a measure of both relevancy and redundancy in mRMR is prohibitive for even small feature sets, and has a worst case of $m + m^2$ permutation methods when computing a full ranking. Therefore, we propose a redundancy metric that is calculated directly from the permutation distributions produced in computing the relevancies, requiring exactly m permutation methods to compute all redundancies. Specifically, we suggest that the similarity of the relevancy permutation distributions be used to estimate redundancy.

If two binary features, x_1 and x_2, are mutually redundant and $MI(x_1, x_2) \approx 1$, then we can say that their relevancies are similar; $MI(x_1, y) \approx MI(x_2, y)$ for any target y. A corollary of this is that dissimilar relevancies, $MI(x_1, y) \not\approx MI(x_2, y)$, imply that the features are not redundant; $MI(x_1, x_2) \not\approx 1$. Unfortunately, similar relevancies, $MI(x_1, y) \approx MI(x_2, y)$, do not guarantee that the features are redundant, and there may be unrelated features with similar relevancies. Knowledge of relevancies does, however, provide some insight into the feature redundancy relationship. For instance, if the two relevancies, $MI(x_1, y)$ and $MI(x_2, y)$, are similar then the features are more likely to be redundant than if the relevancies are very different. Furthermore, if it is known that the features have similar relevancies

with many different targets, the likelihood of their redundancy is increased. This is the basis of the proposed permutation redundancy measure.

The permutation redundancy measure is computed by performing the permutation method for several features simultaneously, permuting only the target at each iteration. For a given permutation, y', the permutation correlations, $MI(x_i, y')$ are recorded for all features $x_i \in X$. Imagine that for all computed permutations of y, $\Psi(y)$, the permutation correlations for features x_1 and x_2 are similar, i.e. $MI(x_1, y') \approx MI(x_2, y') \forall y' \in \Psi(y)$. In this case it is reasonable to conclude that x_1 and x_2 are related and redundant features. If they were not related, some proportion of the permutation correlations would be dissimilar.

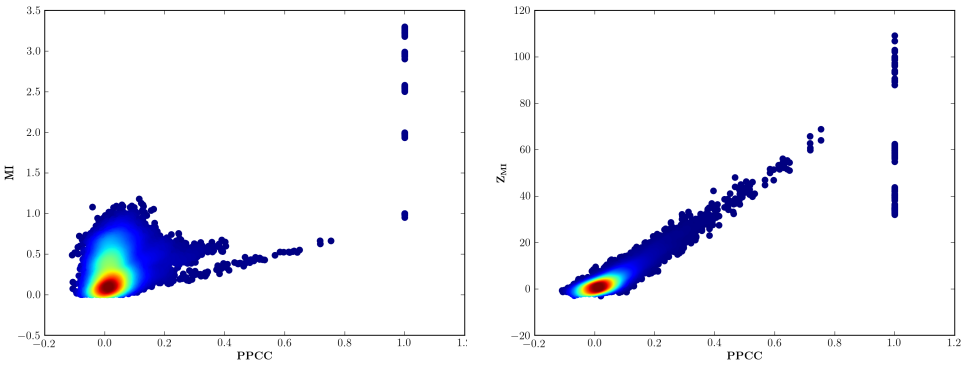
Permutation distributions are not directly comparable by a measure such as mean absolute difference, because the permutation correlations share the same bias found in MI and increase with the dimensionality of a feature. Instead, to successfully compare distributions of different ranges, we use Pearson's Correlation Coefficient,

$$PC_{MI}(x_1, x_2, y) = PCC(MI(x_1, y'), MI(x_2, y') : \forall y' \in \Psi(y)). \tag{3}$$

Simulated data is used to show the relationship between MI, Z_{MI} and PC_{MI}. The data is simulated by generating a uniform binary string of 100 independent samples which is taken to be the target, y. A total of 125 features are then generated by copying this target and changing their sample values randomly to decrease their relationship with y and their cardinalities to increase their entropy and bias their MI with other features. The features are separated into five sets of 25, each of which has a different percentage of the sample values altered. Specifically, the percentages of changed samples are 5%, 10%, 20%, 30%, 40%; producing features varying in levels of relevancy and redundancy. Each set of 25 features with the same number of value changes is split once more into 5 sub-sets. In the first subset, the features are kept the same and remain binary. In the second, each of the feature values are divided uniformly at random into two, creating features of cardinality 4. The third subset has each of the feature values divided into three, while the fourth and fifth subsets have features of cardinality 8 and 10 respectively. This creates a simulated dataset with 5 features for each value change and value split combination, totalling 125 features.

The scatter plots in Figure 1 show that PC_{MI} is highly related to Z_{MI}, but not to MI. This provides evidence that the PC_{MI} redundancy measure does not exhibit the bias in MI, which is rectified by Z_{MI}. Therefore, using this measure may be beneficial to redundant feature selection with the permutation method, as it can be used as a surrogate for permutation normalized MI so that m^2 permutation methods do not have to be performed.

Finally, Z_{MI}, and its surrogate PC_{MI}, can be used in a feature selection filter such as mRMR. Specifically, Z_{MI} can be used as a measure of feature relevancy (in place of $Rel(\cdot)$ in Equation 2), and PC_{MI} as a measure of redundancy (in place of $Red(\cdot)$ in Equation 2). This approach assigns more importance to relevancy than redundancy, however, as the range of Z_{MI} is much larger than PC_{MI}. Rather than use a weighting parameter for the relevancy and redundancy (Vinh et al., 2010) and to consider them of equal importance, they can be normalized between 0 and 1 at each selection step. After this normalization the most relevant feature that is not yet selected will have a relevancy score of 1, and the least redundant unselected feature will have a redundancy score of 0. This feature selection filter is referred to as $PmRMR$ in this paper.



(*a*) MI x PC_{MI} (0.567) (*b*) Z_{MI} x PC_{MI} (0.893)

Figure 1: Scatter plots of MI (a) and Z_{MI} (b) against PC_{MI}. Lighter red points indicate higher density regions. The correlations of the measures are shown in braces.

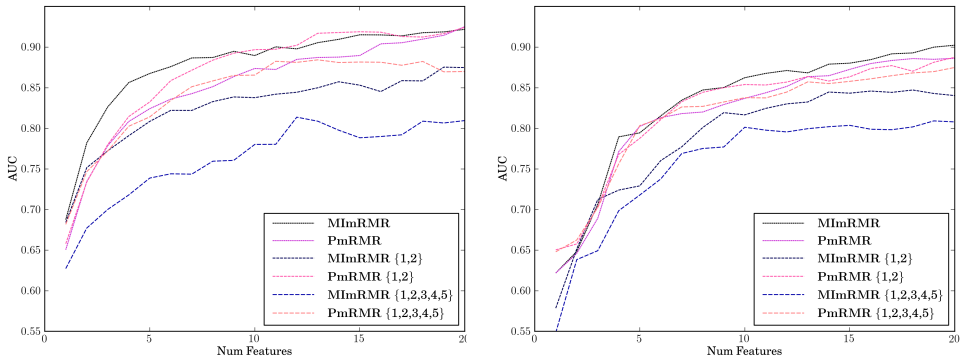
4. Evaluation

To evaluate the $MImRMR$ and $PmRMR$ feature selection methods, we used the Arrhythmia, Chess, Congress, Credit, Fertility, Madelon, Musk 1, Parkinsons, Promoters, Soybean (small), Soybean (large), Spambase, Splice, TR11, TR12, TR21, TR23, Vehicles, Wine, and Yeast that are available in the UCI[1] and Tuned IT[2] repositories. These datasets were chosen because of their range in size and features, as well as their use in previous feature selection literature (Herman et al., 2013). All samples with missing values were first removed from the dataset, before numeric or real valued features were discretized using the minimum descriptive length method (Fayyad and Irani, 1993). At this point, features with only one discrete value were discarded as they contain no information. This is so features can be generated from existing ones, while changing their sample values to worsen their predictive performance and increasing their dimensionality to bias MI.

From each dataset, 5 new datasets were generated by copying original features and increasing their dimensionalities. In all cases, before increasing the dimensionality of a feature, 5% of the values were changed to worsen their predictive abilities. The target variable was not copied or altered in any of the new datasets. The 5 datasets, referred to as $\{1\}$, $\{1,2\}$, $\{1,2,3\}$, $\{1,2,3,4\}$, and $\{1,2,3,4,5\}$, had different numbers of features added to the original ones with different numbers of splits in their values. Dataset $\{1\}$ had double the number of features as the original, and the values of each added feature were split once to double its dimensionality. All of the features present in $\{1\}$ were also in $\{1,2\}$, with one extra copy of the original features having two splits in their values to triple their dimensionalities. In each of the subsequent datasets an extra copy was added on top of the previous, with one extra split in values applied. In the fourth, fifth and sixth datasets therefore, there were four, five, and six times as many features as in the original dataset, with dimensionalities multiplied by four, five and six respectively.

For each of the datasets a random subset validation procedure with ten train-test iterations was performed. In each iteration 50% of the samples were taken uniformly at

1. http://archive.ics.uci.edu/ml, 2. http://tunedit.org/repo/Data/



(*a*) Random Forest, Musk 1 (*b*) SVM, TR 21

Figure 2: Mean AUCs over ten evaluations with between one and twenty features from Musk 1 (a) and TR 11 (b) datasets and using Random Forest and SVM respectively.

random as training data, from which features were ranked using forward selection with both $MImRMR$ and $PmRMR$. To consider relevancy and redundancy of equal importance and for a fair comparison, the relevancies and redundancies in both $MImRMR$ and $PmRMR$ were normalized between 0 and 1, before choosing each feature. Twenty classifiers were then built with increasing numbers of features (between one and twenty) taken from the top of these rankings. The classification algorithms used were Naïve Bayes, Decision Tree, Random Forest, and Support Vector Machine (SVM), which are all available in the WEKA (Witten and Frank, 2011) library. The remaining 50% of the samples in each iteration were used as testing data to produce a performance measure in the form of a weighted Area Under the ROC (Receiver Operating Characteristic) Curve (AUC). Finally, because features were ranked using the same training samples for both ranking methods, the AUC performances produced during each testing iteration can be compared directly.

For illustration, the mean AUC performances over the ten iterations of the TR 21 and Musk 1 datasets, using the Random Forest and SVM classifiers respectively are shown in Figure 2. The plots are representative of using other classifiers with different datasets, and show that AUC decreases as more features with higher dimensionalities present. It also shows that performance decreases less when features are selected using $PmRMR$ than with $MImRMR$. In some cases, and mainly with the Decision tree classifier, where AUCs for the original features were close to 1, the mean AUC performance was affected less by the added features than when the AUC for the original dataset was small.

Table 1 shows the number of times features selected by $MImRMR$ outperformed those selected by $PmRMR$, and vice versa, for each classifier over the 4000 train-test iterations. These results show that $MImRMR$ outperformed $PmRMR$ slightly more often for all classifiers with the original datasets. As more copied features are injected into the datasets with more value splits, $PmRMR$ tends to outperform $MImRMR$ more often. This trend is the same for all four classifiers, but is most clear with Random Forest. In other experiments we also added features with different amounts of sample value changes, but did not find this to significantly affect the performance of either feature ranking method.

Classifier	Original		{1}		{1, 2}		{1, 2, 3}		{1, 2, 3, 4}		{1, 2, 3, 4, 5}	
	MI	**P**	**MI**	**P**	**MI**	**P**	**MI**	**P**	**MI**	**P**	**MI**	**P**
Naïve Bayes	**1805**	986	1533	**1771**	1348	**2160**	1446	**2181**	1278	**2333**	1429	**2208**
Decision Tree	**1222**	1178	1103	**1653**	1062	**1818**	900	**2087**	876	**2133**	1090	**1963**
Random Forest	**1774**	1292	1346	**2193**	1220	**2532**	1175	**2687**	1198	**2673**	1221	**2677**
SVM	**1546**	1035	1418	**1778**	1334	**2035**	1311	**2183**	1216	**2301**	1356	**2259**
Total	**6347**	4491	5400	**7395**	4964	**8545**	4832	**9138**	4568	**9440**	5096	**9107**

Table 1: Number of times features selected by $MImRMR$ (MI) outperformed those selected by $PmRMR$ (P), and vice versa, for each classifier over all train-test iterations.

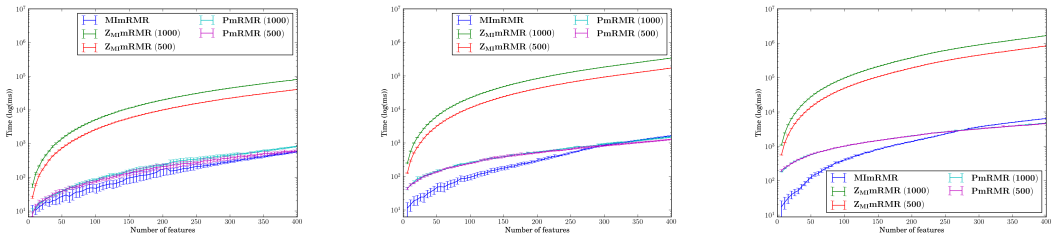
Dataset	{1}		{1, 2}		{1, 2, 3}		{1, 2, 3, 4}		{1, 2, 3, 4, 5}	
	MI	**P**	**MI**	**P**	**MI**	**P**	**MI**	**P**	**MI**	**P**
Datasets where better	3	**15**	1	**17**	1	**16**	2	**16**	1	**17**
Total features	684	**848**	636	**838**	609	**814**	616	**825**	614	**819**

Table 2: Number of original features ranked by in top five by $MImRMR$ (MI) and $PmRMR$ (P) over all train-test iterations.

A good feature ranking method should rank the original features higher than the injected ones, as randomizing values in the copies means that they are worse predictors of the target. The total number of times an original feature was ranked in the top five by $MImRMR$ and $PmRMR$ for the datasets with extra features are shown in Table 2. Detailed results for each dataset are omitted for space reasons, but the number of datasets where one outperformed the other in this task are presented. Overall, as features were copied more and with more splits, fewer original features were ranked in the top five by both $MImRMR$ and $PmRMR$. In the majority of cases, $PmRMR$ outperformed $MImRMR$, and $MImRMR$ was again more affected by increasing the dimensionality of features than was $PmRMR$. One notable case where $MImRMR$ outperformed $PmRMR$ is with the Congress dataset, which is small and simple in structure. In fact, when the top ten or twenty features in the rankings are considered, $PmRMR$ outperforms $MImRMR$ less often, with $MImRMR$ performing better for several smaller datasets including Credit, Fertility, and Soybean (small).

Finally, in Figure 3 we present the times taken to rank all features from simulated binary datasets for different numbers of samples, features and permutations. The datasets are generated to have 100, 1000, 5000 samples with the same percentages of value changes as outlined in Section 3, but with increasing numbers of feature copies. The number of features in the dataset is shown on the x-axis and the y-axis shows the log time taken to rank all features using the $MImRMR$ and $Z_{MI}mRMR$ (with 500 and 1000 permutations) and PC_{MI} (with 500 and 1000 permutations) ranking methods. The time taken to produce the full ranking increased exponentially with the number of features for all selection approaches. The $Z_{MI}mRMR$ selection method, where a new permutation distribution is generated for each redundancy calculation, was by far the slowest method and the computation times of this method increased fastest with the number of features. The computation required by PC_{MI} increased more slowly with respect the size of the data, and was quicker than $MImRMR$ when there were over 275 features and 1000 or 5000 samples.

These computation times are consistent with a complexity analysis of the selection methods. The redundancy computation time of $MImRMR$ and $Z_{MI}mRMR$ is dependent on both the number of features and the sample size, and is $O(nm+n^2m^2)$ for n samples and



(*a*) 100 samples (*b*) 1000 samples (*c*) 5000 samples

Figure 3: Computation times for the $MImRMR$, $Z_{MI}mRMR$ (with 500 and 1000 permutations) and PC_{MI} (with 500 and 1000 permutations) methods to rank all features from a simulated binary datasets with 100, 1000 and 5000 samples and increasing numbers of features.

m features. The $PmRMR$ method on the other hand is dependent on the number of features and the number of permutations computed for the relevancies, and is $O(Pnm + P^2m^2)$. Although the overhead for computing relevancies is larger ($O(Pnm) > O(nm)$), $PmRMR$ is computationally more efficient than $MImRMR$ for large numbers of features and sample sizes when the number of permutations is fixed. This was reflected in our results, where there only is a small difference between computation times of $PmRMR$ with 500 and 1000 permutations with larger numbers of features and sample sizes.

5. Conclusion

This paper investigated redundant feature selection using permutation normalized correlations. We showed with simulated data that permutation normalized MI can be estimated accurately by comparing permutation distributions computed from a common target. A normalized variant of mRMR was used to successfully select features from example datasets with extra features to increase the difficulty of the selection problem. Our approach can automatically select features and requires no parameters other than the number of permutations – which should be as large as is computationally reasonable.

As future work we intend to investigate the relationship between the number of permutations used, and the performance of the features selected using the permutation redundancy measure. We also intend to investigate different feature selection filters with the redundant permutation measure, such as feature clustering (Li et al., 2008).

References

A. Altmann, L. Toloşi, O. Sander, and T. Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

S. Chen. Redundant feature selection based on hybrid GA and BPSO. In *International Conference on Communication Software and Networks*, pages 414–418, 2011.

U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. *Machine Learning*, 1993.

P. Good. *Permutation tests: a practical guide to resampling methods for testing hypotheses*, volume 2. Springer New York, 2000.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

A. Hapfelmeier and K. Ulm. A new variable selection approach using random forests. *Computational Statistics & Data Analysis*, 60:50–69, 2013.

G. Herman, B. Zhang, Y. Wang, G. Ye, and F. Chen. Mutual information-based method for selecting informative feature sets. *Pattern Recognition*, 46(12), 2013.

D. Jensen and P. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97 (1-2):273–324, 1997.

G. Li, X. Hu, X. Shen, X. Chen, and Z. Li. A novel unsupervised feature selection method for bioinformatics data sets through feature clustering. In *IEEE International Conference on Granular Computing*, pages 41–47, 2008.

H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

P. Radivojac, Z. Obradovic, K. Dunker, and S. Vucetic. Feature selection filters based on the permutation test. In *European Conference on Machine Learning*, pages 334–346. Springer, 2004.

P. Taylor, F. Adamu-Fika, S. Anand, A. Dunoyer, N. Griffiths, and T. Popham. Road type classification through data mining. In *Proceedings of the International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 233–240, 2012.

L. Vinh, N. Thang, and Y. Lee. An improved maximum relevance and minimum redundancy feature selection algorithm based on normalized mutual information. In *, 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, pages 395–398, 2010.

J. Wang, W. Lee, and M. McKeown. A novel segmentation, mutual information network framework for EEG analysis of motor tasks. *BioMedical Engineering OnLine*, 8(1):1–19, 2009.

I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Series in Data Management Systems. Morgan Kaufmann, 2011.