Filter-Based Information-Theoretic Feature Selection

Farinaz Pisheh
University of Houston
4800 Calhoun Rd.
Houston, TX, USA
zpisheh@uh.edu

Ricardo Vilalta
University of Houston
4800 Calhoun Rd.
Houston, TX, USA
rvilalta@uh.edu

ABSTRACT
Feature subset selection methods aim at identifying the smallest subset of features that maximize generalization performance, while preserving the true nature of the joint data distribution. In classification tasks, this is tantamount to finding an optimal subset of features relevant to the target class. A distinctive family of feature selection methods use a distance metric to identify relevant features, even under high feature interaction, by looking at the local class distribution. In this study we present EBFS: a new algorithm that is inspired by Relieff and uses an entropy-based metric to discover relevant features. Results on UCI data-sets show the effectiveness of our approach when compared to other filter-based feature selection methods.

CCS Concepts
• Computing methodologies → Feature selection;

Keywords
Feature selection; feature subset selection; supervised learning.

1. INTRODUCTION
Feature selection focuses on selecting only those features that will help to predict the target label with high generalization performance, e.g., with high accuracy. This task is essential in data-sets with a large number of features where often only a small subset of them is relevant. There are several approaches to select an optimal subset that does not include redundant and irrelevant features; in general, such approaches can be roughly divided into two categories: wrapper-based and filter-based [6]. The main difference between the two is that wrapper-based methods find an optimal set based on a specific classifier performance; as such, there is a strong dependency on the embedded classifier. Filter-based methods, on the other hand, evaluate features independently of any classifier; they rely on a metric of relevance that usually falls into one of several types, such as:

- Information-theoretic measures, distance-based measures, or consistency measures [20].

In this work, we introduce a new filter-based feature selection algorithm named EBFS, that resembles Relieff [10], a well-known distance-based method. Like Relieff, our method uses the nearest neighbors of each instance to find the local weight of each feature. The sum of the weights of a feature in all the selected neighborhoods gives the final weight of that feature. Unlike Relieff, our method uses an information-theoretical measurement to select features. While many filter-based methods focus on the relation between the target class and one or more features (mutual information, conditional mutual information, or interaction information), our approach looks at the entropy of feature values in local neighborhoods, effectively capturing information not used in previous distance-based methods. Experimental results on real data-sets show the effectiveness of our approach to detect and identify relevant features.

The rest of this paper is organized as follows. In Section 2, we provide background information on information-theoretic measures commonly used in feature selection algorithms. In Section 3, we review related work on filter-based methods. In Section 4, we describe our methodology. In Section 5, we show experimental results on several synthetic and real-life data-sets, comparing the performance of EBFS with other methods. To finalize, Section 6 provides conclusions and future work.

2. BACKGROUND INFORMATION
Many feature selection algorithms integrate information-theory (IT) in their evaluation method. The fundamental measure in IT is the entropy [12] of a discrete variable X:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$ (1)

where X is a random variable, n is the number of unique values for X, and P(x_i) is the probability of each possible value x_i. Entropy quantifies the amount of uncertainty in a random variable X (in bits). H(X) is maximum when all P(x_i) have the same value; in that case H(X) = log_2 n. The term \(\frac{H(X)}{\log_2 n}\) ∈ [0, 1] is referred to as normalized entropy.

Additional measures are defined next. Let Y be a random variable with m unique discrete values. The joint entropy [12, 2] of X and Y is defined as:

$$H(X,Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log_2 P(x_i, y_j)$$ (2)
where joint entropy is less than or equal to the sum of the individual entropy values of X and Y.

Conditional entropy[2] is defined using the definition of joint entropy:

\[ H(Y|X) = H(X, Y) - H(X) \]  

where the entropy of Y given X shows the decrease in the entropy of Y when knowing the entropy of X.

Mutual information[2] quantifies the amount of information shared by the two variables in terms of entropy and joint entropy:

\[ MI(X, Y) \equiv H(X) + H(Y) - H(X, Y) \]

\[ \equiv H(X) - H(X|Y) \equiv H(Y) - H(Y|X) \]  

Several forms of normalized MI have been proposed. One popular form in feature selection algorithm is as follows [2]:

\[ U(X, Y) = 2 \frac{MI(X, Y)}{H(X) + H(Y)} \]  

Conditional mutual information [2] can be defined for three discrete random variables. Having three variables, X, Y and Z, where Z has q unique discrete values, the following shows the conditional mutual information of X and Y given Z:

\[ MI(X; Y|Z) = \sum_{k=1}^{N} P(z_k) \sum_{i=1}^{m} \sum_{j=1}^{m} P(x_i, y_j|z_k) \log_2 \frac{P(x_i, y_j|z_k)}{P(x_i|z_k)P(y_j|z_k)} \]  

Conditional MI can be obtained using joint entropy, joint entropy, and conditional entropy:

\[ MI(X; Y|Z) = H(X|Z) + H(Y|Z) - H(X, Y|Z) \]

\[ \equiv H(X, Z) + H(Y, Z) - H(X, Y, Z) - H(Z) \]  

Another important concept is that of interaction information (II) [8]; it shows the amount of information that is common in more than two random variables, but cannot be found in any subset of those variables:

\[ II(X, Y, Z) = MI(X, Y) - MI(X, Y|Z) = MI(X, Y|Z) - MI(Y, Z) \]

Unlike other metrics, interaction information can have negative values, indicating overlapping information between variables, a sign of feature redundancy. On the other hand, positive values can be interpreted as feature synergy. This is important because under feature interaction, the individual information of X or Y is not enough to know the value of Z. However, the synergetic interaction between them can reduce the uncertainty of Z to zero [8]. In this paper, we only consider positive interaction.

3. RELATED WORK

Mutual information is commonly used to measure the average dependency between a feature and the target class; features with high MI have a greater chance to be included in the final subset of features. A popular forward feature selection technique using MI is called MRMR [11]; to avoid redundancy, the algorithm penalizes high values of MI between two features. Another algorithm is called CMIM [4]; it looks for features corresponding to high \( MI(X_i, Y) \) where \( X_i \) is a previously selected feature, Y is the target class, and \( X_i \) is a candidate feature; in this way, the algorithm discards features bearing high information about the target class, but that have similar information as previously selected features. Feature Selection Based on Joint Mutual Information (JMI) [17], evaluates the quality of a candidate feature \( X_i \), by joining it to each of the previously selected features \( \{X_j\} \) and measuring \( MI(X_i, X_j|Y) \).

Previous work has emphasized the importance of maintaining a balance between feature redundancy and relevancy to achieve competitive results [1]; as an example, the algorithm JMI provides a trade-off between accuracy and stability for small data-sets. CFS [7] uses Pearson’s correlation coefficient to find features bearing high correlation with the target class; to detect redundant features, the algorithm excludes features highly correlated to other features. FCBF [19] follows a similar approach by gathering a set of features exhibiting high symmetrical uncertainty with the class, removing redundant features within the selected features set.

None of the work above considers the feature interaction problem; each feature is simply assessed individually against the target class, while redundant features are discarded based on their correlation with other features. Unfortunately, interactive features might be removed in the process of discarding redundant ones. Employing the formula in equation 8 requires an exhaustive and computationally costly search for high-order interactive features.

3.1 Feature Interaction

Recent work advocates the search for interacting features jointly correlated with the target class. In [9], an interactive feature is called weakly relevant if it is not highly correlated with the target class, but belongs to a subset of features that together are strongly correlated with the target class. In INTERACT [22], this definition is used to propose a criterion to remove features following a backward search technique. In IWFS [21], a metric called interaction weight factor based on symmetrical uncertainty and conditional mutual information is proposed; the algorithm captures the interaction of up to three features and the target class. CMICOT [13] detects the interaction between multiple features under a greedy search and a sequential forward selection approach; the algorithm uses a binary feature representation to estimate the CMIM measure in high-dimensional problems, and can identify interactions of up to t features and the target class (t is an input parameter). In [16], a four-dimensional joint mutual information measure is used to detect high order interactions. In general, identifying high-order interactive features by using an efficient algorithm remains an open problem.

3.2 Relieff

Relieff is a distance-based feature selection algorithm [10] that works as follows. It first selects m instances and identifies the neighborhood around each instance. Using a distance-based measure (e.g., Euclidean distance), Relieff finds k nearest instances that are from the same class of the selected instance (nearest hits), and the k nearest instances from each of the different classes (nearest misses). The weight of each feature \( f_j, W[f_j] \), is then updated according to the following
where \( Z_i[f] \) is the value of feature \( f \) for the \( i \)th instance in the dataset, \( \text{Miss}_i \) and \( \text{Hit}_i \) are the \( j \)th nearest miss and the \( j \)th nearest hit respectively. \( \text{diff}(A[f], B[f]) \) is the distance between the values of \( f \) in A and B; for nominal features it is zero when the values are equal and one otherwise. For numerical attributes it is equal to \( (A[f] - B[f])/\Gamma \) where \( \Gamma \) is the range of possible values (max - min). Algorithm 1 shows pseudo code for the Relief algorithm.

**Algorithm 1: Relief**

Input : Data-set \( D = \{Z_i | 0 \leq i \leq M\} \) with features: \( F = \{f_j | 0 \leq j \leq n\} \) and \( C \) : Classes. \( k \) : Number of neighbors

Output: \( W[f_j] \) : Weight of each feature in the data-set

1. Set all \( W[f_j] = 0 \)
2. for \( i=1 \) to \( m \) do
   3. Pick a random instance \( Z_i \) from \( D \)
   4. Find \( k \) nearest Hits: \( \text{Hit}_i \)
   5. foreach \( C \neq \text{class}(Z_i) \) do
      6. Find \( k \) nearest Misses: \( \text{Miss}_i(C) \)
   7. end
   8. for \( j=1 \) to \( n \) do
      9. Update \( W[f_j] \) according to equation 9
   10. end
11. end

By looking at the local neighborhood of each instance, Relief is able to update the weight of each feature. The update compares the value of a feature in the current instance with the value of the feature in each of the nearest miss and nearest hit neighbors. Features with values similar to the nearest hits, and farther away from the nearest misses will score higher. Jakulin [8] states that the common weaknesses of algorithms that use information theory to detect feature interaction does not exist in Relief. By looking at the behavior of a feature in each neighborhood, Relief is capable of identifying interactive features. One limitation of Relief is its inability to remove feature duplicates. Several attempts have used the basic idea of Relief to come up with more effective methods [5, 15, 14].

Our contribution is to introduce an algorithm inspired by Relief that employs entropy to find relevant features. Experimental results on several data-sets of different sizes show that our algorithm is capable of identifying relevant features, including interactive features.

**4. METHODOLOGY**

While our method is inspired by the original Relief, we have used entropy to create a new metric called \( E_{\text{measure}} \). In this section we explain our new metric and our feature selection algorithm.

Relief is an extension to the original Relief algorithm. One of the critical properties of the Relief family is evaluating the relevancy of a feature based on the different values it takes on different classes. At its core, Relief estimates the following to obtain the final weights:

\[
W[f] \sim P(\text{diff. values of } f|\text{nearest misses}) - P(\text{diff. values of } f|\text{nearest hits})
\]

With this interpretation of Relief, a feature is expected to attain a high weight if it leads to a set of different feature values among instances of different classes, while it leads to a set of similar feature values among instances of the same class. In other words, the formula penalizes features that separate instances of the same class. Now let \( \text{Hits}_i \) and \( \text{Misses}_i \) represent two sets that contain the nearest hits and nearest misses of \( Z_i \) respectively. Let \( \text{val}(F_i, \text{Hits}_i) \) and \( \text{val}(F_i, \text{Misses}_i) \) be the set of values for feature \( F_i \) in the \( j \)th feature of all nearest hits and nearest misses of instance \( Z_i \) respectively. We can re-formulate the weight update formula for a feature \( f_j \) with respect to instance \( Z_i \) as follows:

\[
W[f_j, Z_i] \approx H(\text{val}(f_j, \text{Misses}_i)) - H(\text{val}(f_j, \text{Hits}_i))
\]

This formula shows the difference in uncertainty of the values of \( f_j \) in nearest misses and nearest hits. The formula shows two entropy terms: the first one corresponds to the amount of “choices” we have for the value of \( f_j \) in the set of nearest misses. In the case of non-binary classification problems, at least two classes are considered different from the class of instance \( Z_i \). In relevant features, we expect to have different values for each of the classes in the set of nearest hits; as a result we look for high entropy values where more choices of feature values lead to higher weight updates.

In the same way, the second term in equation 11 shows the amount of “choices” for \( f_j \) in the set of nearest hits. Here we look for low entropy values, meaning we expect most feature values to be almost the same. A value of zero means all nearest hits have the exact same value for \( f_j \).

While the formula in equation 11 can quantify the uncertainty of feature values in the neighborhood of an instance \( Z_i \), it fails to capture relevant information. Consider the following example. Assume a binary classification and neighborhoods of size \( k = 4 \). Assume the feature values for nearest misses is \{1,1,1,1\} and values for nearest hits is \{0,0,0,0\}. Although the feature separates nearest hits from nearest misses perfectly, both terms in equation 11 yield the same entropy value. To overcome this, we add the value of \( f_j \) of the current instance \( Z_i \), \( Z_i[f_j] \), to both terms of equation 11. We call this measurement \( E_{\text{measure}} \):

\[
E[f_j, Z_i] = H(\text{val}(f_j, \text{Misses}_i), Z_i[f_j]) - H(\text{val}(f_j, \text{Hits}_i), Z_i[f_j])
\]

By adding this additional term, we expect to reduce the value of \( \text{val}(f_j, \text{Misses}_i) \) in relevant features as we are adding a value that is anticipated to have higher probability of occurrence. On the other hand, adding it to \( \text{val}(f_j, \text{Hits}_i) \) leads to an increase of uncertainty as it is anticipated that this value will have lower chance of occurrence. We used normalized entropy in the formula to keep the range of entropy values between [0, 1]. Algorithm 2 shows pseudo code for EBFS.
Algorithm 2: EBFS

Input: Data-set \( D : \{ Z_i | 0 \leq i \leq M \} \)
with features: \( F : \{ f_j | 0 \leq j \leq n \} \) and
\( C : \) Classes.
\( k : \) Number of neighbors.

Output: \( W[f_j] : \) Weight of each feature in \( D \)

1. Set all \( W[f_j] = 0 \)
2. foreach \( Z_i \) in \( D \) do
   3. Hits = \{\}, Misses = \{
   4. Find \( k \) nearest Hits:
   5. \( \text{Hits}_i = \{\text{Hits}_i | 1 \leq i \leq k\} \)
   6. foreach \( C \neq \text{class}(Z_i) \) do
      7. Find \( k \) nearest Misses: \( \text{Miss}_i(C) \)
      8. Misses = Misses \( \cup \{\text{Miss}_i(C) | 1 \leq i \leq k\} \)
   9. end
   10. foreach \( f_j \) in \( F \) do
        11. Find values of \( f_j \) in Misses \( \cup \{\text{val}(f_j, \text{Misses})\} \)
        12. Find values of \( f_j \) in Hits \( \cup \{\text{val}(f_j, \text{Hits})\} \)
        13. \( W[f_j] = e \cdot \text{measure}(Z_i, F) \) according to equation 12
   14. end
15. end

5. EXPERIMENTS AND RESULTS

To assess the performance of our method, we compare it with five state-of-the-art feature selection algorithms: Relief [10], MMRR [11], CMIM [4], IWFS [21], and JMI [18]. We implemented EBFS in Python 3.0. For Relief, we used \( k = 3 \) and the built-in function from Matlab. IWFS was implemented in Matlab. The rest of the algorithms were provided by FEAST\(^1\).

The data-sets employed in the experiments can be found at the UCI repository\(^2\), except the Colon data-set\(^3\). We report on datasets having a variety of sample size, no. of features, and no. of classes. All the datasets were pre-processed to remove index values. On all the algorithms that use information theoretic measures, we discretized features with continuous features using the MDL method [3]. We used three classifiers in Scikit-learn\(^4\): Linear SVM, KNN, and Decision tree.

Similar to settings reported on recent publications [21, 13, 16], we planned our experiment as follows. We applied the feature selection algorithm on each of the datasets. Each algorithm was run \( n \) times on each dataset where \( 1 \leq n \leq \min(50, \# \text{features in dataset}) \). After the \( i \)th run, the top \( i \) feature(s) were selected and the rest were discarded. Then the dataset was fed to the three classifiers and run using ten-fold cross-validation. We report on average accuracy and standard deviation for the \( n \) runs. Results are shown in Table 2.

Each result was compared against EBFS for statistical significance using a two-tailed t-student test with \( p = 0.05 \). An (+) shows that EBFS is significantly better than the competitor, and an (-) shows the competitor is significantly better. For all the results with no signs, no significant difference were found. EBFS obtains the best results with decision tree classifier; no loss is seen against other feature selection algorithms in this classifier, and it achieved significantly better in fifteen cases.

In general, our method performed significantly better or equally well than other algorithms. In comparison to Relief, EBFS lost in just one dataset with the KNN classifier. Looking at the three classifiers results, EBFS achieved the most wins against IWFS; it won six, seven and eight times with Decision trees, KNN and SVM respectively. In comparison to JMI and CMIM, EBFS performed mostly equal or better than these algorithms in all the classifiers. EBFS obtained similar results in comparison to MMRR when using decision trees and KNN. In SVM, however, EBFS lost three times and won twice; making MMRR our strongest competitor.

6. CONCLUSIONS AND FUTURE WORK

Feature subset selection is a crucial task during the preprocessing phase of almost any classification task. Finding (high-order) interactive features is an important challenge that needs further attention. In this paper, we introduce EBFS, an entropy-based feature selection algorithm that analyzes the neighborhood of a random instance (as originally proposed in the well-known Relieff algorithm) to estimate feature quality. Different from previous work, EBFS focuses on the entropy of feature values for nearest hits and misses. In comparison with five popular feature selection algorithms, our method shows competitive results, and is capable of finding relevant features, including interactive features. In the future, we plan to improve our proposed \( e \cdot \text{measure} \) to relate more directly to generalization performance. In addition, we plan to investigate how to find neighborhoods in a selective manner, to increase the ability to identify interacting features.

7. REFERENCES


<table>
<thead>
<tr>
<th>Data-set</th>
<th>#Features</th>
<th>#Samples</th>
<th>#Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast</td>
<td>9</td>
<td>699</td>
<td>2</td>
</tr>
<tr>
<td>CMC</td>
<td>1473</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>Vehicle</td>
<td>846</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>KR</td>
<td>3196</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>Lung</td>
<td>32</td>
<td>56</td>
<td>3</td>
</tr>
<tr>
<td>audio</td>
<td>200</td>
<td>69</td>
<td>24</td>
</tr>
<tr>
<td>Musk</td>
<td>476</td>
<td>166</td>
<td>2</td>
</tr>
<tr>
<td>SCADI</td>
<td>70</td>
<td>205</td>
<td>7</td>
</tr>
<tr>
<td>Colon</td>
<td>62</td>
<td>2000</td>
<td>2</td>
</tr>
</tbody>
</table>

\(^1\)http://www.cs.man.ac.uk/~pococka4/FEAST/
\(^2\)https://archive.ics.uci.edu/ml/datasets/
\(^3\)http://genomics-pubs.princeton.edu/oncology/
\(^4\)https://scikit-learn.org/
Table 2: Average accuracy with standard deviations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>KNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Breast</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM</td>
<td>CM</td>
<td>CM</td>
</tr>
<tr>
<td>90.24±0.07</td>
<td>90.35±0.07</td>
<td>90.45±0.07</td>
</tr>
<tr>
<td>91.93±0.07</td>
<td>92.33±0.07</td>
<td>92.73±0.07</td>
</tr>
<tr>
<td>93.63±0.07</td>
<td>94.03±0.07</td>
<td>94.43±0.07</td>
</tr>
<tr>
<td>94.33±0.07</td>
<td>94.73±0.07</td>
<td>95.13±0.07</td>
</tr>
<tr>
<td>95.03±0.07</td>
<td>95.43±0.07</td>
<td>95.83±0.07</td>
</tr>
<tr>
<td>95.73±0.07</td>
<td>96.13±0.07</td>
<td>96.53±0.07</td>
</tr>
<tr>
<td>96.43±0.07</td>
<td>96.83±0.07</td>
<td>97.23±0.07</td>
</tr>
<tr>
<td>97.13±0.07</td>
<td>97.53±0.07</td>
<td>97.93±0.07</td>
</tr>
</tbody>
</table>

**Notes:**
- The results are presented as accuracy ± standard deviation.
- The table compares different datasets (Breast, CM, etc.) and algorithms (KNN, SVM).\n- The algorithms considered are KNN and SVM with various parameters.
- The datasets include binary classification tasks.
- The accuracy values range from 90.24% to 99.13%.