A New Approach for Automated Feature Selection

Andreas Gocht, Christoph Lehmann, Robert Schöne
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden, 01062 Dresden, Germany
{andreas.gocht | christoph.lehmann | robert.schoene}@tu-dresden.de

***Abstract*—Feature selection or variable selection is an important step in different machine learning tasks. In a traditional approach, users specify the amount of features, which shall be selected. Afterwards, algorithm select features by using scores like the Joint Mutual Information (JMI). If users do not know the exact amount of features to select, they need to evaluate the full learning chain for different feature counts in order to determine, which amount leads to the lowest training error. To overcome this drawback, we extend the JMI score and mitigate the flaw by introducing a stopping criterion to the selection algorithm that can be specified depending on the learning task. With this, we enable developers to carry out the feature selection task before the actual learning is done. We call our new score Historical Joint Mutual Information (HJMI). Additionally, we compare our new algorithm, using the novel HJMI score, against traditional algorithms, which use the JMI score. With this, we demonstrate that the HJMI-based algorithm is able to automatically select a reasonable amount of features: Our approach delivers results as good as traditional approaches and sometimes even outperforms them, as it is not limited to a certain step size for feature evaluation.**

I. Introduction

While more and more data is collected for machine learning, validation and exploration of this data become increasingly challenging. For example, neuronal networks need more training data if the network is trained with more different features. To tackle this challenge, two methodologies are used in order to reduce the amount of features: *Feature extraction* and *feature selection* [1, Chapter 6]. Feature extraction reduces the amount of features by compressing their information. In consequence, information density increases. Alternatively, feature selection reduces the amount of features by selecting only the most informative features. Therefore, if the collection of feature data is expensive, or if the amount of features that can be used is limited, feature selection is the better approach. Let us consider two examples, one from the High Performance Computing (HPC) domain and another more general one:

In the HPC domain, we collect performance data in form of hardware performance monitoring counters (PMCs) [2]. PMCs enable users to count events that occur in a processor and are therefore suited to evaluate the performance of applications. An example for such data is the number of executed instructions or the number of Level 3 cache misses. The latter can be used as an indicator for how often fetched data is not present in caches but must be loaded from main memory. A high Level 3 cache miss rate indicates that the application can be memory bound and is not able to utilize the full computing performance of the processor. However, even though there are hundreds of countable processor events, only a limited amount of counters available to count these [3]. Therefore, a performance analyst needs to carefully analyze and select which events he intends to record. As the number of different events is too high for manually evaluating all available events, an automated approach for the selection would be helpful.

A more general example are production robots, which incorporate sensors that ensure a correct behavior. Here, each sensor increases the cost of the machine. Therefore, an engineer might be interested whether the amount of sensors can be reduced. This would be the case, if other sensors can provide the information of the sensor that is to be removed. Again, this is a classical task for feature selection, since it is interesting how many features (sensors) are really needed.

Both tasks can be solved by using feature selection, but not feature extraction. Therefore, we just consider feature selection in this paper.

Feature selection itself is often distinguished in three sections: embedded, wrapper, and filter methods [4]. The idea behind *embedded* methods is to add and remove features in order to minimize the prediction error of the whole learning algorithm [4]. Especially in machine learning where tasks have a long runtime, this approach is unpractical, as in theory, each combination of features needs to be evaluated. Like embedded methods, *wrapper* methods split data into training and validation data. But instead of evaluating the prediction error of the whole chain, wrapper methods are employed before the learning algorithm is triggered [4]. Both, the embedded and wrapper methods add hyperparameters to the optimization chain of modern machine learning. In contrast, *filter* methods employ the selection process before the actual learning is done [4]. Therefore, the amount of experiments is reduced significantly, as the features are chosen before the learning algorithm is used. Most filter methods require the user to specify the amount of features that the algorithm should select. The amount of features to select can be evaluated based on the error of the subsequent learning task: The lower the error, the closer the optimal amount of features. Unfortunately, this eliminates one main advantage of filter mechanisms. Especially for deep learning tasks, this is not a good approach, where the whole the network needs to be re-trained for each set of features.

Resulting from this observation and the two examples presented above, we infer the following properties for a good feature selection algorithm:

- Automatic stopping after all informative features are selected

- Alternative stopping after an application dependent maximal amount of features is selected
- No (re-)evaluation of the subsequent learning task

In order to implement such an algorithm, we extend the well-known Joint Mutual Information (JMI) based feature selection [5]. Our algorithm is capable of selecting the most relevant features and stops once the information added does not improve the quality of the result anymore. Moreover, it is possible to specify an upper limit of features and our algorithm is independent from any subsequent machine learning task.

The rest of the paper is organized as follows: Section II gives a short overview about the theoretical background and related work. In Section III, we explain our new feature selection methods. Section IV presents our results in comparison with JMI. Finally, we provide a summary and an outlook in Section V.

II. Theoretical Background and Related Work

In this section we summarize existing feature selection scores and algorithms in order to establish the theoretical background for our new approach, while Section IV focuses on the implementation and evaluation of the new approach.

The target of all feature selection algorithms is to select those features that provide most information about the class they predict. Therefore, the most-used algorithms utilize scores based on information theory.

A first approach in this direction was done by Lewis [6], who used the Mutual Information (MI) as score for feature selection. MI bases on entropy, cf. [7, p. 29] The entropy H for a variable U is defined by:

$$H(U) = -\sum_{u \in U} p(u) \log p(u)$$

while u denotes the possible value from the variable U, and $p(u)$ denotes the relative frequency of u[1]. The conditional entropy for a variable U depending on a variable V is defined by:

$$H(U|V) = -\sum_{v \in V} p(v) \sum_{u \in U} p(u|v) \log p(u|v)$$

while u and v denote the possible value for the variables U and V.

As an example for further use, assume a classification task where Y is a set containing different classes to be predicted. Let $X = \{X_1, X_2, X_3, ..., X_K\}$ be a set of K features. Now we can calculate the MI:

$$J_{MI}(X_k) = I(X_k; Y) = H(X_k) - H(X_k|Y)$$

for all $k = 1, \dots, K$. A possible graphical representation of the entropy and the MI is given in Figure 1. One of MIs properties is $I(X_k; Y) \in [0, 1]$, while $I(X_k; Y) = 0$ if X_k and Y are statistically independent, i.e., X_k has no information about Y. Moreover, $I(X_k; Y) = I(Y; X_k)$, cf. [7, p. 29].

For the description of the algorithm and for further use, let l denote the number of most informative features, whereby

[1]Note that $\sum_{u \in U} p(u) = 1$.

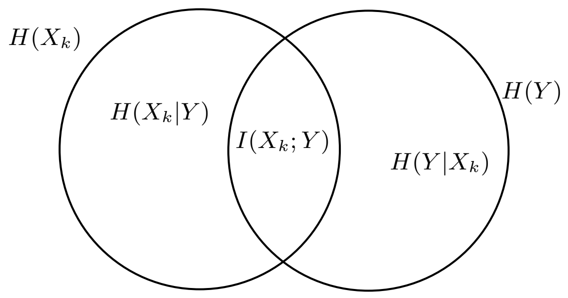

Figure 1: We can consider the entropy as the area of the circles, while the circles themselves represents the different classes in a feature. The Mutual Information, $I(X_k; Y)$ is constructed from the entropy, e.g., $H(X_k)$, and the conditional entropy, e.g., $H(X_k|Y)$. The area marked by $I(X_k; Y)$ represents the information that is shared among the feature and the predicted value Y.

$l \leq K$. In order to select the l most informative features for this task, the algorithm now calculates the MI score for all K features, and rank them according to the score in descending order. Finally, the algorithm collects l features with the largest $I(X_k; Y)$ into the set of selected features S.

The approach from Lewis was extended in different directions. For example, Yang and Moody used the Joint Mutual Information (JMI) as score to avoid redundant information between the selected features [5]. The JMI is based on the Conditional Mutual Information (CMI). Let us consider the CMI for $X_k \notin S$, Y and an already selected feature $X_j \in S$,

$$I(X_k; X_j|Y) = H(X_k|Y) - H(X_k|X_jY).$$

Note that the set S again contains all selected features. The JMI is now defined as:

$$\begin{aligned} J_{JMI}(X_k) &= \sum_{X_j \in S} \left(I(X_k; Y) + I(X_k; Y|X_j) \right) \\ &= \sum_{X_j \in S} I(X_k X_j; Y) \end{aligned}$$

Figure 2 illustrates the score.

The algorithm proposed by Yang can be written as follows:

1) Calculate $J_{MI}(X_k)$ for all features in $X_k \in X$
2) Add the feature with the largest $J_{MI}(X_k)$ to S
3) Calculate $J_{JMI}(X_k)$ for all features $X_k \in X \backslash S$
4) Add the feature with the largest $J_{JMI}(X_k)$ to S
5) Repeat 3) and 4) until $|S| = l$

The algorithm based on the JMI score has some advantages over traditionally used approaches for feature selection. Yang and Moody also show that it outperforms algorithms, which are based on PCA or CCA when it comes to data that is not Gaussian distributed. Furthermore, they demonstrated that four features, i.e., $l = 4$, out of 20, i.e., $K = 20$, which are selected by JMI, lead to a faster convergence of a neural network than using all available features.

In a study of different information theoretic based feature selection mechanisms, Brown et al. [7] recommended JMI as

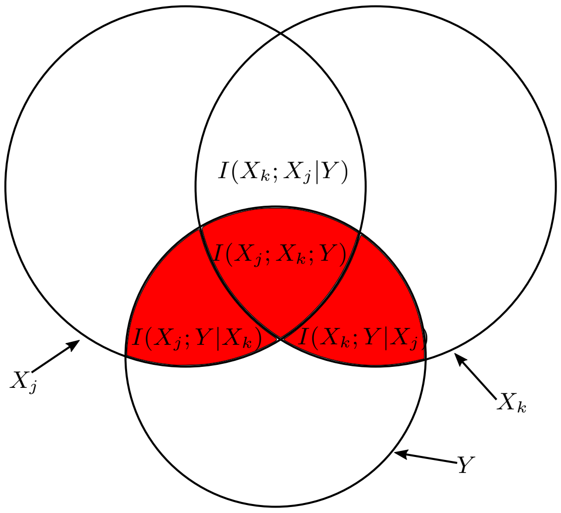

Figure 2: The goal of JMI is to find a new feature $X_k \in X$ that maximizes the red marked area, together with the already selected feature $X_j \in S$. Y represents the target class.

the best overall score for feature selection. They investigated other scores like Conditional Mutual Information Maximization or Mutual Information Maximization, but concluded that JMI gives a good trade-off between stability and accuracy. Moreover, they exposed the different feature selection mechanisms to a subset of benchmarks from the NIPS 2003 Feature Selection Challenge [8]. Brown et al. pointed out that the JMI-based algorithm performs better than most other feature selection methods. Therefore, we use JMI as base for our extension.

Hall [9] chose a conceptual different approach. He proposed the so-called Correlation-based Feature Selection (CFS). To calculate a score for currently selected features he used a heuristic derived by Ghiselli [10]. The metric optimizes the average feature-class correlation with respect to the average feature-feature correlation. CFS uses a best-first search to add features. Once the metric stops increasing, it drops back to the next unexplored subset, and explores this subset. After a specified number of steps, the search terminates and the feature space is re-evaluated. Now, all features are added, which have a correlation with the already chosen set of features that is higher than the highest correlation between the feature and all other already chosen features. CFS can handle continuous valued data using the Pearson correlation coefficient, which is a benefit against the information-theory-based features selection mechanisms. However, due to the final step of re-evaluation, it is hard to limit the amount of features that shall be added to the final set of features. Moreover, the Pearson correlation coefficient only works well on data with linear characteristics, but the dependency between features is not necessarily linear.

III. A New Score for Feature Selection

Similar to most filter mechanisms, the JMI-based feature selection method supports only a predefined number of features, as described previously. Therefore, if the optimal amount of features is unknown, it needs to be estimated. To do so, a certain amount of features is selected, and the error of the subsequent machine learning algorithm is evaluated. In a next step, a different amount of features is chosen, and the error of the algorithm is re-evaluated. For example, if we consider $l \in \{20, 40, 60, ..., 200\}$ relevant features, we need to evaluate the error of the subsequent machine learning algorithm for these $20, 40, 60, ..., 200$ features. From these results, we have to choose the amount of elements, i.e. l, which lead to the lowest error. Depending on the given amount of data, the machine learning algorithm, and the amount of features, such an approach can become very time consuming. Therefore, an automatic method to determine the required amount of features would be helpful.

A. *Description of the New Score and Algorithm*

As we mentioned in previous sections, we use JMI as basis for our new score. As reminder, the algorithm based on JMI, selects a new feature if the score of

$$J_{JMI}(X_k) = \sum_{X_j \in S} [I(X_j;Y) + I(X_k;Y|X_j)],$$

is larger than all other feature scores of the unselected features. In order to understand the interconnections between the JMI and our new score, we need to expand the JMI according to Brown et al. [7, p. 62]:

$$\begin{aligned} J_{JMI}(X_k) = \sum_{X_j \in S} \big[&I(X_j;Y) + I(X_k;Y) \\ &- \big(I(X_k;X_j) - I(X_k;X_j|Y)\big)\big]. \end{aligned}$$

According to Matsuda [11, Eq. 2.24], it is reasonable to rewrite $I(X_k;X_j) - I(X_k;X_j|Y)$ to $I(X_k;X_j;Y)$, which leads to

$$J_{JMI}(X_k) = \sum_{X_j \in S} \big[I(X_k;Y) + I(X_j;Y) - I(X_k;X_j;Y)\big]. \tag{1}$$

Even though we do not use Equation (1) for calculations, it helps us to explain the equation: The first term $I(X_k;Y)$ defines the mutual information between the potential feature to select X_k and the target class Y. This can be interpreted as the information that X_k holds about Y. The second term $I(X_j;Y)$ adds the mutual information between the already selected features X_j and the target class Y. This can also be understood as the information that one of the already selected values X_j holds about Y. However, if X_k and X_j hold similar information about Y, we would account for this information twice. Therefore, $I(X_k;X_j;Y)$ is subtracted, which is the mutual information of X_k, X_j and Y.

With this score, we can determine which information the joint features X_k and X_j hold about Y. We repeat this for all features in S. This allows us to select the one feature X_k

out of the previously not-selected features, which adds the most new information about Y, whereas $X_k \in X$ and $X_k \notin S$. However, this score does not account for development of the information, which our selected features in S accumulate over time. In order to overcome this issue, we propose a novel score, called Historical JMI (HJMI):

$$J_{HJMI}(X_k, S) = J_H + I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|} \tag{2}$$

where J_H represents the historical information about the already selected features. The function of J_H is best explained by the modified algorithm for the features selection, which is extended by a stopping criterion:

1) set $J_H = 0$
2) calculate $J_{HJMI}(X_k, S)$ for all $X_k \in X \backslash S$
3) save the largest result for $J_{HJMI}(X_k, S)$ as J_H and add the associated X_k to S
4) repeat 2) and 3) until stopping criterion is met or the maximal amount of features is reached

J_H is therefore similar to the second term of Equation (1) $I(X_j, Y)$ as it holds the information about already selected features. However, it holds even more information, as the last term of Equation (2) $\sum_{X_j \in S} I(X_k; X_j; Y)$ is included for all already selected features as well. As $I(X_k; Y)$ ranges between 0 and 1, it is reasonable to divide $\sum_{X_j \in S} I(X_k; X_j; Y)$ by the amount of already selected features $|S|$.

B. Computational Cost

For a fixed amount of features, e.g., $|S| = l$, it holds

$$J_{HJMI}(X_k) \propto I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|}. \tag{3}$$

This is similar to the JMI for a fixed amount of features, as outlined by Brown [7, p. 63]

$$J_{JMI}(X_k) \propto I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|}. \tag{4}$$

This indicates that the computational complexity of JMI and HJMI is the same for the same amount of features. However, for practical applications, the computational costs of HJMI will be smaller than those of JMI, since it stops once the stopping criterion is met. In contrast, JMI continues until a given amount of features is reached. Moreover, if the stopping criterion stops the algorithm after all relevant features are selected, it is not necessary to evaluate the prediction error of subsequent machine learning tasks to choose the optimal amount of features.

C. Stopping Criteria

Now that we have introduced the algorithm, we define the stopping criterion. As we described in the previous section, the algorithm adds a new term for each new feature. The question is now, when it should stop adding new terms and consequently new features. For ideal data, we can assume that there is no new information added if the first two terms reach zero, i.e.,

$$0 = I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|}. \tag{5}$$

This would be a reasonable stopping criterion. However, real life data is not ideal. Therefore, we define a threshold ε as stopping criterion

$$\left\| I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|} \right\| < \varepsilon. \tag{6}$$

Nevertheless, determining a suitable value for ε is not easy. Let us assume that we have a set of features which all result in $I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|} = 0.09$. If ε is too high, e.g., 0.1, all of these features will be ignored. If ε is too low, e.g., 0.01, they will be added. Therefore, ε must be adopted to the data it is applied to. This is a tedious task, which requires a-priori knowledge and is not generalizable.

Therefore, we propose a different approach. Since we have historical data available, our algorithm checks how much a new feature improves the overall quality of the metric.

$$\delta > \frac{I(X_k; Y) - \frac{\sum_{X_j \in S} I(X_k; X_j; Y)}{|S|}}{J_H}. \tag{7}$$

If the information added by a new feature X_k does not increase the information of the already selected features in S by more than a given δ, the algorithm will stop. In consequence, the selection will terminate at one point, independently of δ. For most benchmarks shown in Section IV, a threshold of $\delta = 3\%$ seems to be a good choice. However, it is reasonable to apply a smaller value if an application requires more precision or if the costs per features are low.

IV. Implementation and Validation

To evaluate our approach, we use the NIPS Feature Selection Challenge [8], [12]. We compare our results to that from Brown [7].

A. NIPS Feature Selection Challenge

The idea of the NIPS Features Selection Challenge is to provide a set of different classification challenges, which are supposed to be sufficiently solved using as few features as possible.

To do so, the authors chose different classification tasks from various fields: Mass spectrometry (ARCENE), text classification (DEXTER), drug discovery (DOROTHEA) and digit recognition (GISETTE). A fifth task (MADELON) was created artificially to represent the task of selecting features, where "no feature is informative by itself" [8]. The different tasks were enriched by features, which were drawn from a random distribution.

While Brown presented the results for GISETTE and MADELON, we chose to compute the results for all five classification benchmarks. As we did not upload our results, we just used the given training data, and evaluated it with the given validation data.

Listing 1: MATLAB inspired pseudocode for the algorithm utilizing HJMI

```
function select_features (X, Y, max_iterations)

select_features = [];
collect_hjmi = [];
j_h = 0

for n = 1:max_iterations
  for k = 1:features
    if ismember(k, selected_features)
      continue
    end
    jmi_1 = mi(X(k),Y)
    jmi_2 = 0
    for j = selected_features
      tmp1 = mi(X(k), X(j));
      tmp2 = cmi(X(k), X(j), Y);
      jmi_2 = jmi_2 + tmp1 - tmp2;
    end
    jmi(k) = j_h + jmi_1
    if n > 1
      jmi(k) = jmi(k) - jmi_2 / n - 1
  end

  [j_h, ind] = max(jmi);
  if ( ((j_h - hjmi)/hjmi) > 0.03 )
    and ( len(select_features) < max_features )
      hjmi = j_h
      select_features = [select_features ind];
      collect_hjmi = [collect_hjmi hjmi];
  else
    break
  end
end

return [select_features, collect_hjmi]
```

B. Classification Algorithm

Similar to Brown [7], we use a k nearest neighbor classifier, with $k = 3$. Moreover, we choose to discretize each feature into 10 bins. However, we did not succeed in reproducing the exact same numbers as Brown, but our validation error is close. The difference from our JMI results to the ones from Brown is less than 1% for the GISETTE benchmark and around 1% for MADELON, which seems acceptable. This difference might be related to differences in the discretization or the dataset.

C. Implementation

As we defined $I(X_k; X_j; Y)$ over $I(X_k; X_j) - I(X_k; X_j|Y)$, we now re-substitute $I(X_k; X_j|Y)$.

$$\begin{aligned} I(X_k; X_j; Y) =& I(X_k; X_j) - I(X_k; X_j|Y) \\ J_{HJMI}(X_k, S) =& J_H + I(X_k; Y) \\ & - \frac{\sum_{X_j \in S}[I(X_k; X_j) - I(X_k; X_j|Y)]}{|S|} \end{aligned}$$

For our implementation, we rely on the MIToolbox v3.0.1 [13] to calculate the mutual information $I(X_k; X_j)$ and $I(X_k; Y)$ as well as the conditional mutual information $I(X_k; X_j|Y)$. The first feature is selected according to the highest mutual information with the target class.

We chose MATLAB [14] as programming language. Listing 1 provides an overview on our implementation.

Benchmark	JMI Validation Error [%]	JMI Amount of Features (l)	HJMI Validation Error [%]	HJMI Amount of Features (l)
ARCENE	21.19	20	19.64	32
DEXTER	15.0	60	13.0	21
DOROTHEA	32.99	200	25.63	24
GISETTE	4.1	200	8.0	26
MADELON	10.67	20	10.67	20

Table I: Results for NIPS Feature Selection Challenge. The first column shows the smallest error for the validation set, with features selected using JMI. The second column shows the amount of features used to achieve this result. The third and fourth column present the same information for the newly introduced HJMI-based algorithm.

D. Results

Table I and Figure 3 provide an overview about our results. We compute the results for the JMI-based algorithm by evaluating the training and validation error for a different amount of features. For the results in Table I, we select the number of features, where the training error was the smallest. To train the model, we use training data from the features chosen by JMI as input. The error is calculated using the validation data and is presented in Table I. For the results of HJMI, we use the algorithm presented in the previous section. We calculate training and validation error in the same way as for the JMI algorithm.

The results, which we achieved for ARCNE, DEXTER, and DOROTHEA are significantly worse than one would expect for a good learning task. This can be related either to the selected learning algorithm, or to the discretization of the training data for the feature selection. However, in this paper, we focus on the algorithm and the presented stopping criterion. The challenge of finding a good solution for discretization or a learning algorithm that outperforms the current state-of-the-art is part of future work.

However, Table I also shows that in most cases HJMI-based algorithm delivers better results than the JMI-based one. This is related to the automated stopping criterion. It allows the algorithm to select the number of features outside of predefined steps. Only the GISETTE benchmark does not profit from this, which is expected behavior. As shown in Figure 3, the benchmark profits from a large amount of features, while the improvement for each added feature is quite small. Therefore, the increase of the HJMI score drops below 3 % and the algorithm stops. Here, a lower threshold would lead to a higher amount of selected features, and therefore to better results. Depending on the application, an additional error of 3.9 % could be acceptable if the effort to collect 174 features can be saved.

Figure 3 also shows the behavior of the MADELON benchmark described by Brown. As soon as the number of selected features increase above 20, the learning algorithm performs worse with every new feature that is added. Our novel HJMI-based algorithm is able to detect this boundary by itself.

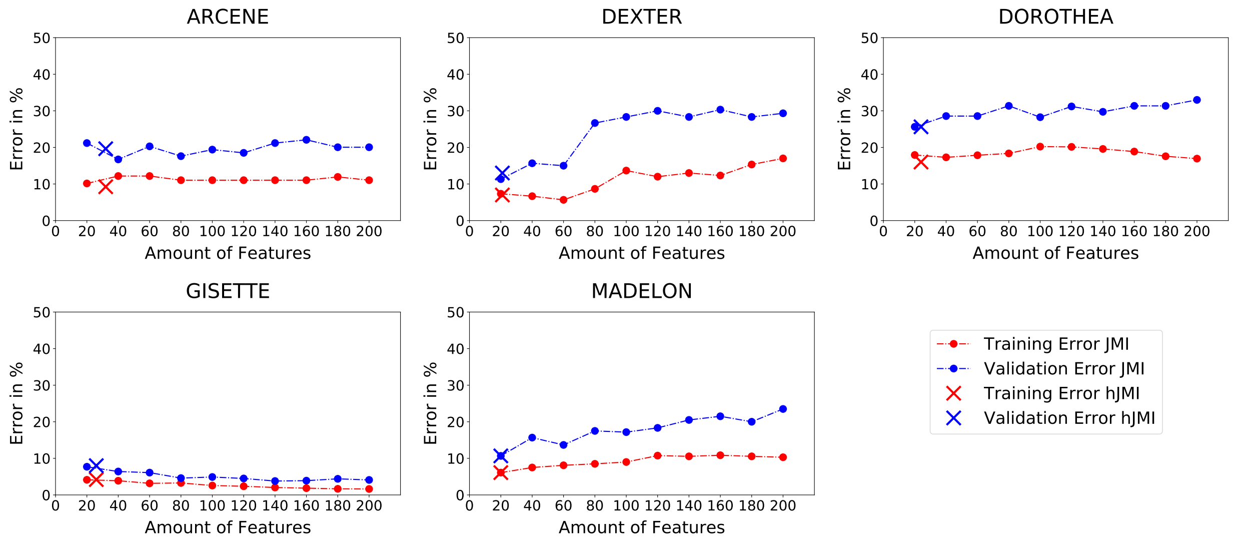

Figure 3: Comparison of the training and validation error archived with features selected using the JMI and HJMI score. For the JMI-based algorithm the different results for the different given amount of features are presented. For the HJMI-based algorithm only the automatically selected amount of features is shown.

V. Conclusion and Future Work

In this paper, we presented a novel score for feature selection called HJMI. The HJMI score allows an extension of the selection algorithm based on JMI by a stopping mechanism. Our results indicate that our approach performs well compared against the original in terms of feature selection capabilities: The new algorithm finds suitable features automatically if the learning task can be discretized. Hence, users no longer has to specify the amount of features needed for learning themselves. However, they still *can* specify an application depended upper limit of features Here, the new algorithm will select as many features as needed, but no more than permitted.

Unfortunately, if the training data is continuous valued and cannot be discretized, mutual-information-based metrics cannot be used. Under these circumstances, the entropy, which is used to determine the mutual information, can no longer be calculated. Therefore, further research is needed to exploit the properties of HJMI for such data. If being combined with a better learning algorithm, our approach will significantly improve the training and validation error, as we have shown.

Furthermore, the impact of the stopping threshold to the subsequent machine learning task needs to be evaluated. A possibility to estimate the improvement of the prediction quality based on the stopping threshold could be used to improve cases like the GISETTE benchmark. We also think about adding a cost to each feature in the equation. This would enable us to take the overhead of collecting each feature into account. To do so, the HJMI score could be extended by adding a feature specific cost term. This extension is interesting in mechanical engineering where each feature relates to a sensor. Here, it might be worthwhile to use multiple cheap sensors instead of an expensive one.

Acknowledgments

This work is supported by the European Union's Horizon 2020 program in the READEX project (grant agreement number 671657) and the German Federal Ministry of Education and Research (BMBF, 01IS14014A-D) by funding the competence center for Big Data "ScaDS Dresden/Leipzig".

References

[1] E. Alpaydin, *Introduction to Machine Learning*, ser. Adaptive computation and machine learning. MIT Press, 2014.

[2] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with papi-c," in *Tools for High Performance Computing 2009*, M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, Eds., 2010, DOI: 10.1007/978-3-642-11261-4_11.

[3] D. Molka, R. Schöne, D. Hackenberg, and W. E. Nagel, "Detecting memory-boundedness with hardware performance counters," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, 2017, DOI: 10.1145/3030207.3030223.

[4] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, 1997, DOI: 10.1016/S0004-3702(97)00063-5.

[5] H. H. Yang and J. Moody, "Data visualization and feature selection: New algorithms for nongaussian data," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, ACMID: 3009755.

[6] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the Workshop on Speech and Natural Language*, 1992, DOI: 10.3115/1075527.1075574.

[7] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, 2012, ACMID: 2188387.

[8] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in Neural Information Processing Systems*, 2004, ACMID: 2976109.

[9] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, ACMID: 657793.

[10] E. E. Ghiselli, *Theory of Psychological Measurement (Psychology S.)*, ser. Series in Psychology. McGraw-Hill Education, 1964.

[11] H. Matsuda, "Physical nature of higher-order mutual information: Intrinsic correlations and frustration," *Physical Review E*, 2000, DOI: 10.1103/PhysRevE.62.3096.

[12] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, "Feature selection challenge," 2018, https://competitions.codalab.org/competitions/3931.

[13] A. Pocock, "Mitoolbox," Online, accessed Feb 2017, https://github.com/Craigacp/MIToolbox/tree/v3.0.1.

[14] The MathWorks, Inc, "MATLAB 2018a," Online, accessed Oct 2018, https://www.mathworks.com.