Class Label Specific Mutual Information Feature Selection Method Paper Review

Randall Fowler, *Department of ECE, UC Davis*

***Abstract*—Modern data processing is a complex field in which the number of dimensions in a dataset may be relatively large. Information theory techniques can be used to estimate the amount of information within a feature of the dataset to select a reduced feature space while maximizing the accuracy and speed of the model. Selection of features should follow a process that minimizes complexity and maximizes qualities defined by constraints. Commonly defined constraints would be to minimize redundancy, maximize importance to each feature, and maximize information between features. Class-label specific mutual information (CSMI) is a feature selection method to optimize features over each class label rather than over all classes. Normally, information theory-based approaches would follow a general framework to maximize over all features, and this does provide a further reduced feature space. The complexity of CSMI is larger than other methods, but it is also capable of achieving a better classification accuracy than most other techniques.**

***Index Terms*—Feature seletion, class label specific mutual information, information theory, estimating mutual information.**

I. Introduction

THIS is a review of a paper called "A General Framework for Class Label Specific Mutual Information Feature Selection Method" [1]. Information theory-based feature selection (ITFS) is one of the best methods of selecting optimal features for classification of data. The objective for ITFS is to find a subset of the original feature space to maximize a user defined value function. Most modern applications provide a very large, or high-dimensional, feature space in which the larger the feature space is, the more complexity added to data processing. Examples of datasets evaluated in this paper contain hundreds or thousands of collected features relating to text data, facial images, and speech recognition. There are many feature selection methods to estimate the importance of individual features, however, ITFS methods compare the importance of each feature along with optimizing the information collected between features. Some features within the original feature space can contradict one another, repeat information, or may not even contribute to labeling data.

The general framework for most ITFS methods contains terms to maximize relevancy, minimize redundancy, and maximize conditional redundancy. Relevancy between a feature and a label is the measurement of importance on how well a feature will contribute to classification. This is the foundation for all ITFS methods as each term of the value function can be weighted for varying the importance. Relevancy, or importance estimation, should be the term with the most weighting and typically given a weight of one where all other terms will be given a weight less than or equal to one. Redundancy will only add complexity as it pertains to an additional feature that could be removed. "The inclusion of correlated features are beneficial if the correlation across the classes is stronger than the overall correlation" [1]. This means that with given class information, redundancy can be helpful in maximizing information.

While many ITFS methods share the same general framework for calculating the value of the features, CSMI approaches the feature space from a new angle. Instead of maximizing the value of features over all classes, CSMI attempts to maximize the features over each label individually. The intention is to increase information for predicting each class, but this does add complexity to the data processing as the number of features and machine learning (ML) models will increase. Normal ITFS methods would only have one selected feature space and model, but CSMI will have the same number of selected feature spaces, models, and labels.

II. Methodology

Information theory is one of the most effective tools for selecting optimal features for maximizing accuracy in a model. Within this field of study, the amount of information or relevancy between two variables can be calculated using mutual information (MI). When the distribution of a dataset is unknown, the MI must be estimated to quantify the value within the collected data.

A. Estimating Mutual Information

A common method to estimate entropy and mutual information is to utilize a k-th nearest neighbor (kNN) algorithm for measuring distances between each sample. Optimizing these estimator techniques is still a progressing topic, and a python library, sklearn, provides a function for estimating MI while distinguishing discrete versus continuous cases [2]. Within the library function, it calculates MI using an equation similar to

$$\begin{aligned}\hat{I}(X;Y) = \Psi(N) + \Psi(k) - \frac{1}{N}\sum_{i=1}^{N}\Psi(n_x(i)+1) \\ -\frac{1}{N}\sum_{i=1}^{N}\Psi(n_y(i)+1)\end{aligned} \tag{1}$$

where n_x and n_y are samples within the neighborhood of k distance [3]. This distance is found by using kNN on a

This is a paper review completed as a class project for Information Theory, EEC 266, taught at UC Davis by Dr. Lifeng Lai.

Written on December 5th, 2023.

joint distribution between x and y. The Ψ function refers to a digamma function defined as

$$\Psi(t) = \frac{\Gamma'(t)}{\Gamma(t)} \tag{2}$$

and

$$\Gamma(t) = \int_0^\infty u^{t-1} e^{-u} du \tag{3}$$

For conditional MI, a similar approach can be made by using

$$\begin{aligned} \hat{I}(X;Y|Z) = \Psi(k) - \frac{1}{N}\sum_{i=1}^{N}\Psi(n_{xz}(i)+1) \\ -\frac{1}{N}\sum_{i=1}^{N}\Psi(n_{yz}(i)+1) + \frac{1}{N}\sum_{i=1}^{N}\Psi(n_z(i)+1) \end{aligned} \tag{4}$$

to provide given information to the MI estimate [4]. Again, n_z would be the number of samples within the kth distance of a given sample, and n_{xz} and n_{yz} would be the count for joint distances. While (1) and (4) were implemented for the paper review, the challenges faced would be the large runtime, and for this reason, the library was used for MI estimates.

Since the python library for MI estimation does not contain a function for conditional MI or joint variables, functions for this must be implemented separately. Conditional MI has a known relationship with joint MI that can be used to separate terms and reduce complexity.

$$I(X;Y|Z) = I(X;Y,Z) - I(X;Z) \tag{5}$$

In (5), $I(X;Z)$ can be calculated using the library function for MI, but the joint MI, $I(X;Y,Z)$, would still require an additional method. To find the joint MI, kNN is used on a 3-dimensional space to find the k-th neighbor distance for each point. A K-dimensional Tree (KDTree) speeds up the process of counting the number of samples within a neighborhood, and then, (1) can be used with n_{yz} being the number of samples in the neighbor between the joint of two variables. This process will differ for discrete and continuous variable combinations, and this could be a source of differences between the results reproduced in this paper compared to the results in the reviewed paper.

B. General Framework for ITFS Methods

The general framework for all ITFS methods focuses on maximizing the value of each feature f_k in a subset feature space S.

$$J(f_k) = I(f_k;C) - \alpha \sum_{f_j \in S} I(f_j;f_k) + \beta \sum_{f_j \in S} I(f_j;f_k|C) \tag{6}$$

Each ITFS method will select a different value function for determining optimal features in a smaller feature space. Most ITFS methods compared to in the reviewed paper utilize (6) with different selections of α and β hyperparameters. The different hyperparameters will vary the weighting given to the redundancy and conditional redundancy values.

The only ITFS method in the reviewed paper that has a slightly different value function than (6) is Max-Relevancy and Max-Independence (MRI).

$$\begin{aligned} J_{MRI}(f_k) = I(f_k;C) + \sum_{f_j \in S} I(f_k;C|f_j) \\ + \sum_{f_j \in S} I(f_j;C|f_k) \end{aligned} \tag{7}$$

MRI has a value function with terms that maximize on the relevancy of selected features f_j to all labels C given a target feature f_k as well as relevancy of the target feature to all labels given the selected features [5].

All the compared ITFS methods, except MI Maximization (MIM), will have a time complexity of $O(TFN)$, given that MI is calculated with $O(N)$, J_{MIM} is calculated with $O(FN)$ for F number of features in the entire feature space, and T being the threshold number of features to be selected. In the implementation for this paper, estimating the conditional MI is not optimal and has a larger time complexity than expected.

C. CSMI implementation

To optimize selected features for each label rather than over all labels, the dataset is replicated m times for m number of labels. Each of the replicated datasets will be dedicated towards a single class and given a new label of true or false, true referring to the data with the original class label and false for all other classes. This is known as Class Binarization as each of the new datasets will have a true label and false label, and each new dataset will be used to find an optimal subset of features for maximizing each of the original labels individual.

One issue provided is that each binary dataset could have a very unbalanced number of true and false labels. Maximizing the defined value function could have a large bias towards one of the two labels. To reduce this issue, an oversampling technique is applied to lesser of the two labels to provide an equal number of classes. The reviewed paper mentions different oversampling and undersampling methods being applied, but the best results were observed with a simple oversampling technique of repeating samples [1]. With balanced binary datasets, an ITFS method can be applied to each for gathering m selected feature spaces.

The CSMI framework differs from the general framework for ITFS methods as it maximizes the amount of information to the target class label P_i where i is the label for the original class labels C.

$$\begin{aligned} J(f_k) = \sum_{f_j \in S_{c_i}} [I(f_k;P_i|f_j) - I(f_k;C|f_j) \\ + I(f_j;P_i|f_k) - I(f_k;f_j)] \end{aligned} \tag{8}$$

It has a similar expression to (7) where conditional information given target or selected features is maximized, and redundancy is still minimized. For all ITFS methods discussed in this paper, each will start by finding J_{MIM} for selecting the first feature. The J_{MIM} method will continue this for all selected features, but all other ITFS methods will use the selected feature space for selecting the next feature. Implementation of the CSMI algorithm can be seen in algorithm 2 of the reviewed

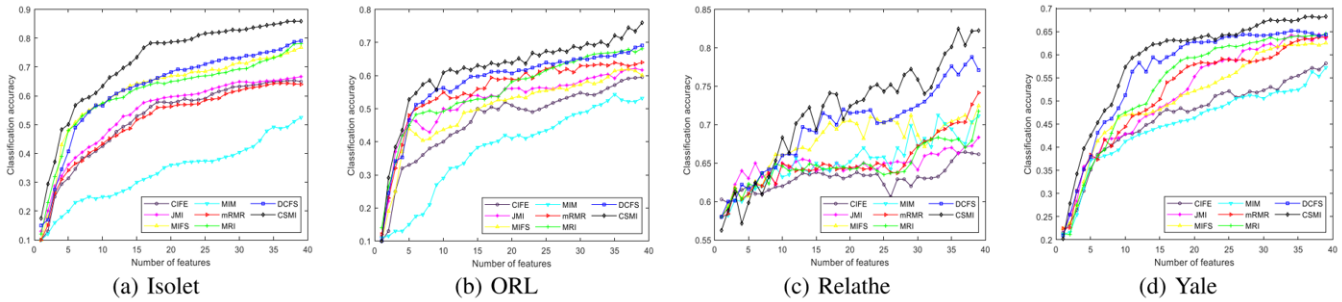


Fig. 1. Average classification accuracy achieved on four representative data sets.

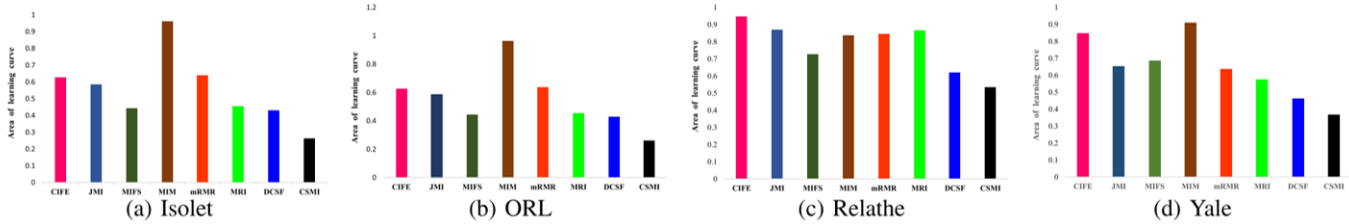


Fig. 2. Average classification accuracy achieved on four representative data sets.

paper, and MI can be estimated using methods discussed in section II-A [1].

Time complexity does increase to $O(mTFN)$ since the CSMI algorithm will be applied to m datasets. While this may not be a concern for extracting features only once, the classification with selected features also increases in complexity. Each selected feature space will be used as input to a dedicated model, and each model will be trained using the original class label space, not the binary labels.

With m models being utilized, a voting system decides the final classification. Majority voting will be used to output a final classification, but a larger weight will be given to the votes that voted for their target label. Target label for each model would be the label in which the corresponding selected feature space was optimized for classifying. Since the reviewed paper did not specify how to decide on a tie between votes, a tie will go to the class with the smaller label value, but in reality, this would be a 50-50 decision.

III. Experimental Analysis

For the experimental analysis of the CSMI method versus seven other ITFS methods, the reviewed paper used 16 datasets to evaluate the performances. After extracting features for each ITFS method, 4 different ML models were used: Naïve Bayes (NB), kNN, Support Vector Machine (SVM), and eXtreme Gradient Boosting (XGBoost). Tables 7 to 17, in the reviewed paper, display results and comparisons of the varies ITFS methods, datasets, and ML models [1]. The majority of these results show that CSMI actually does have better performance than previous methods. Over an average of 100 runs, CSMI does better on all the datasets and with each ML model.

The selected figures to reproduce are figures 10 and 11 in the reviewed paper, and these are shown in figures 1 and 2 in this paper. These figures use 4 datasets, Isolet, ORL, Relathe, and Yale, with an unknown ML model used to produce these figures. It is assumed that XGBoost is used to produce these results.

Figure 1 shows the classification accuracy of the ML model with a given number of selected features on the x-axis. For each ITFS method shown in the legend, the selected features in the order of selection will be used for training the model. This means that each ITFS method will require being trained 40 times with 1 feature, 2 features, and so on till 40 features are given to the model. These results are averaged over 100 runs and CSMI appears to nearly always have the better classification accuracy at almost all the selected number of features.

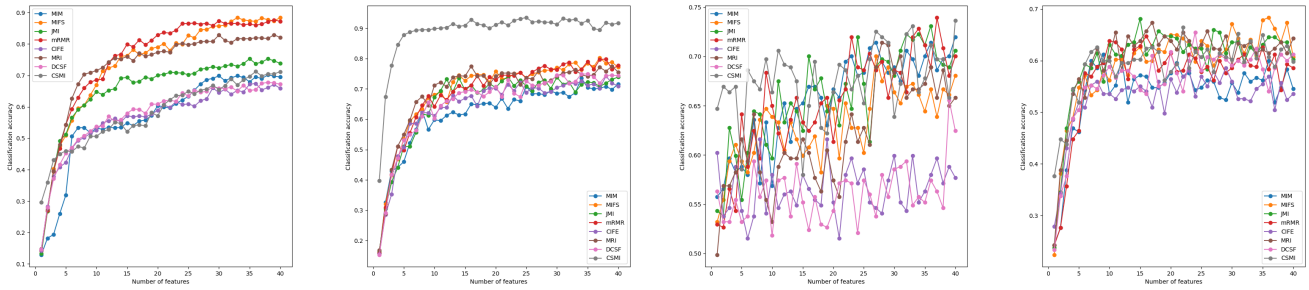
One concern in figure 2 is how goodness-of-fit was calculated for CSMI as it uses m number of models while the other methods only use 1. It is assumed that an average is taken over all CSMI models, but this is not addressed for evaluating the training time. The focus of the figure could be the quick learning handled by each ITFS method, but either way, CSMI does outperform all the other methods shown.

IV. Replicated Project

A. Time Complexity

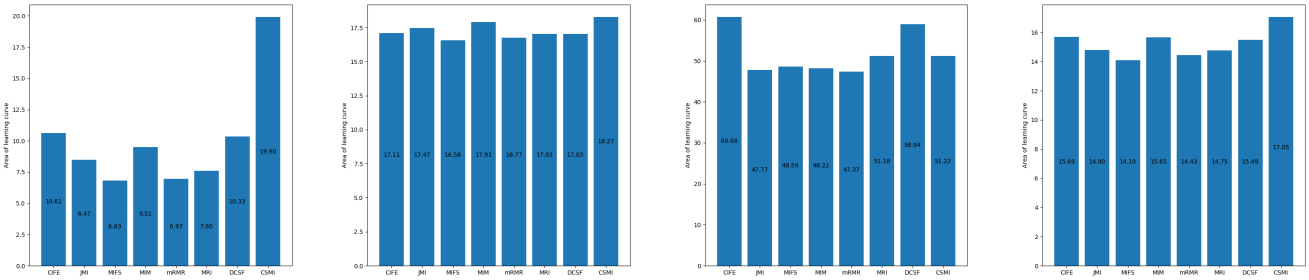
For the experiment of the project, a Lambda server with a A1000 GPU was used for training all ML models. To reduce complexity of the project, only XGBoost was implemented for the replicated results as the ML model that the reviewed paper used was not specified. A dedicated library for XGBoost was selected for optimizing the training and evaluating time.

One of the greatest challenges experienced by reproducing the reviewed paper would be the large time complexity. The first version of the project required nearly two weeks to select features from all the datasets. Luckily, the sklearn library does



(a) Isolet (b) ORL (c) Relathe (d) Yale

Fig. 3. Reproduced results of figure 1 with averaging of 10 iterations.



(a) Isolet (b) ORL (c) Relathe (d) Yale

Fig. 4. Reproduced results of figure 2 with averaging of 10 iterations.

optimize the MI estimations, but conditional MI was still an issue. After revising the project multiple times, the time requirement was reduced to 40 hours with Isolet and ORL requiring 13.5 and 18 hours respectively. Selected features for all the feature spaces were printed as logs and saved as a numpy array when the algorithm finished a dataset.

All other ITFS methods were conducted in one algorithm and required 8.7 hours to complete all datasets. Ralethe required 5.6 hours to select features for all ITFS methods, except CSMI, and this should be due to it being the largest of the datasets. However, this dataset only has two labels which should not significantly differ the time between CSMI and other ITFS methods. An explanation for this would be the optimization in the numpy library when computing all ITFS methods rather than just CSMI.

Model training and evaluation varied greatly depending on the dataset. Ralethe did seem to greatly benefit from the reduced feature space as one session of training and evaluation took 0.2 to 5 seconds for all ITFS methods. Yale also managed to be trained fairly quickly with 30 to 50-second sessions. Isolet and ORL provided timing concerns as sessions lasted for several minutes, and these datasets constrained the number of averages used in this paper. Averaging 10 runs for all datasets, the results shown in figure 3 and 4 required 26 hours to reproduce after feature selection. Timing variations between 1 run or 10 runs is due to the XGBoost library invoking processes for the models.

B. Results

The figures 3 and 4 contain the results of reproducing figures 1 and 2. While figures 1 and 2 were created with an averaging over 100 sessions, time constraints only allowed for completing 10 sessions in the reproduced results. Accuracy seen in figure 3 was collected from an $mlogloss$ and $logloss$ error functions depending on the number of labels, but only Ralethe required $logloss$ since it had binary classes.

Figure 3b has the most similar results to figure 1 as the sequence of ITFS trends match; the reproduced results for ITFS actually showed a significant increase in performance versus the other methods. Yale seems to have a similar trend, and with an average of 100 runs, the compared plots of 3d and 1d might match. However, all other plots in figure 3 look different and might not improve with more averaging.

For the goodness-of-fit measurement, the learning curve was integrated, and the produced results were not normalized. Figure 2 appears to be normalized or possibly using a different error function, but the size differences between the reviewed paper and this paper's results can still be compared. CSMI goodness-of-fit was averaged over all models used in classifying, but this did not seem to help with the results. Ralethe in figure 4c appears to have the most similar results, but it is difficult to compare.

Differences in the results might come from the reduced number of averaging, but it is more likely that some of the feature selection was not conducted properly. Estimating

conditional mutual information was a main concern for this project, and it is unsure if all cases are being handled correctly. With differences in discrete versus continuous MI estimating, some of the datasets might not be handled correctly. ORL did seem to have good results in reproducing accuracy, but the goodness-of-fit did not. A mistake in calculating goodness-of-fit might have occurred, but further averaging would clear up uncertainty in quality of all results.

V. Conclusion

Many challenges were seen as this project progressed. Estimating mutual information was the biggest influence as this defined the $O(N)$ time complexity that would be used in other algorithms. MI could be estimated using a python library that followed similar techniques to recent publications [2], [3]. However, conditional MI or joint MI encouraged further investigation into kNN estimation for using most of the ITFS methods described in this paper. Without an optimized implementation, the time requirement was too large to make changes and check for better results, and this is described in detail in section IV-A.

Results of the paper were not ideal, but with an increase in averaging, the results could be closer to the reviewed paper. The accuracy for the dataset ORL did have the most similar results, but Relathe and Yale would certainly improve with additional runs.

A. Future Work

This paper review has provided insight on estimation methods for dataset distributions. One issue with the CSMI method is that the increase in complexity is large compared to others as the number of labels will increase the number of feature spaces. With a large number of features, CSMI quickly becomes unusable as an ITFS method. However, one possible solution for this increase in complexity would be to group class labels by relevancy. Instead of performing Class Binarization, new labels spaces could be created as all relevant labels would have their own unique label and irrelevant labels would be given a single shared label. To find these reduced label spaces would be the challenge of this proposed work, but using a similar method of selecting features with kNN estimating could be applied to selecting relevant labels. The number of selected feature spaces and ML modes would reduce from m to l as l would be the number of grouped labels.

With ITFS methods being applicable to most high-dimensional datasets, CSMI could be applied to research with the LEPS projects as the transabdominal fetal oximeter (TFO) device does extract many signals that can be analyzed to have a large number of features. Adapting the CSMI algorithm to select features on regressions, rather than discretize class labels, could be beneficial to the TFO project. Information theory could also be applied to selecting weights for maximizing the amount of information gathered between signals.

References

[1] D. K. Rakesh and P. K. Jana, "A General Framework for Class Label Specific Mutual Information Feature Selection Method," IEEE Trans. Info. Theory, vol. 68, no. 12, Dec. 2022.

[2] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Nov. 2011.

[3] P. Zhoa and L. Lai, "Analysis of KNN Information Estimators for Smooth Distributions," IEEE Trans. Info. Theory, vol. 66, no. 6, Jun. 2020.

[4] O. C. Mesner and C. R. Shalizi, "Conditional Mutual Information Estimation for Mixed, Discrete and Continuous Data," IEEE Trans. Info. Theory, vol. 67, no. 1, Jan. 2021.

[5] J. Wang, J. Wei, Z. Yang, and S. Wang, "Feature Selection by Maximizing Independent Classification Information," IEEE Trans. Info. Theory, vol. 29, no. 4, Apr. 2017.