Efficient Wrapper Feature Selection using AutoEncoder and Model Based Elimination

Journal:	*IEEE Letters of the Computer Society*
Manuscript ID	LOCS-2020-05-0016
Manuscript Type:	Letters
Keyword:	I.5.2.b Feature evaluation and selection < I.5.2 Design Methodology < I.5 Pattern Recognition < I Computing Methodologies, I.2.6.g Machine learning < I.2.6 Learning < I.2 Artificial Intelligence < I Computing Methodologies

Efficient Wrapper Feature Selection using AutoEncoder and Model Based Elimination

Sharan Ramjee, *Student Member, IEEE*, and Aly El Gamal, *Senior Member, IEEE*

Abstract—We propose a computationally efficient wrapper feature selection method - called Autoencoder and Model Based Elimination of features using Relevance and Redundancy scores (AMBER) - that uses a single ranker model along with autoencoders to perform greedy backward elimination of features, without requiring model retraining. The ranker model is used to prioritize the removal of features that are not critical to the classification task, while the autoencoders are used to prioritize the elimination of correlated features. We demonstrate the superior feature selection ability of AMBER on four well known datasets corresponding to different domain applications via comparing the accuracies with other computationally efficient state of the art feature selection techniques, and note how a surprisingly small number of features can lead to very high accuracies on some datasets.

Index Terms—AMBER, loss functional, ranker model, breast cancer, relevance and redundancy.

1 Introduction

Feature selection is an input pre-processing technique that eliminates features insignificant to the task at hand. As examined by [1], it is a powerful tool to alleviate the curse of dimensionality and reduce training time, as well as to improve data comprehensibility. For classification problems, [2] divides feature selection problems into two types: (a) given a fixed $k \ll d$, where d is the total number of features, find the k features that lead to the least classification error and (b) given a maximum expected classification error, find the smallest possible k along with a minimal set of features. In this letter, we will be focusing on problems of type (a). We can formalize this type of feature selection problems as follows. Given a set (x, y) of input vectors x and labels y, find a mapping of data $x \mapsto (x * \sigma)$, $\sigma \in \{0, 1\}^d$, along with a function f to minimize

$$\tau(f, \sigma) = \int L_y(f(x * \sigma))dP(x, y), \tag{1}$$

subject to $\|\sigma\|_0 = k$, where the distribution $P(x, y)$ is unknown, and can be inferred only from the training set, $x * \sigma = (x_1\sigma_1, \ldots, x_d\sigma_d)$ is an element-wise product, and $L_y(\cdot)$ is the loss functional.

Feature selection algorithms have three types: Filter, Wrapper, and Embedded methods. Filters rely on intrinsic data characteristics while wrappers measure the learning performance of a classifier to rank feature importance. [3] asserts that although filters are more computationally efficient than wrappers, the features selected by filters are not as good. Embedded methods incorporate feature selection into the underlying model to reconcile the efficiency advantage of filters with the learning algorithm interaction advantage of wrappers. However, embedded methods are model dependent because they perform feature selection during the training of the learning algorithm. This serves as a motivation for the use of wrapper methods that are computationally efficient and not model dependent. Wrapper methods can be further divided based on the search type for candidate features. We focus on Heuristic Search Wrappers that iteratively eliminate one feature at each iteration due to their computational efficiency advantages [4].

- *S. Ramjee and A. El Gamal are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. Email: {sramjee, elgamala}@purdue.edu.*

1.1 Motivation

1.1.1 Relevance and Redundancy

Our driving intuition is that irrelevant features are insignificant because their direct removal does not result in a drop in classification accuracy, while redundant features are insignificant because they are linearly or non-linearly dependent on other features and can be inferred - or approximated - from them. As detailed by [5], one does not necessarily imply the other. Filter methods are known to be better at identifying redundant features while wrapper methods are better at identifying irrelevant features, and hence, there is a need to incorporate a filter based technique to identify redundant features into wrapper methods, which we address using autoencoders.

1.1.2 Training the Classifier only once

Wrappers often have high computational complexity because the classifier needs to be trained for every considered feature set. For greedy backward elimination wrappers, like the Recursive Feature Elimination (RFE) method [6], the removal of k out of d features requires training the classifier for $\sum_{i=1}^{k}(d - i + 1)$ times, which is burdensome when d is large. Also, the saliency of the features selected is governed by how good the classifier that ranks the features is, and as such, we need to use state-of-the-art classifiers for ranking the features. These models often require large training times, which implies a trade-off between speed and quality of selected features. We address this issue by training the feature ranker model only once and relying on simulations of the loss functional that exploit a transferability property of neural networks.

2 STATE OF THE ART

We describe top-notch efficient feature selection methods that we will be comparing our proposed method to. With the exception of FQI, the implementations of these methods can be found in the scikit-feature package created by [3].

Fisher Score encourages selection of features that have similar values within the same class and distinct values across different classes. A precise definition is in [7].

Conditional Mutual Information Maximization (CMIM) is proposed in [8], [9], and iteratively selects features while maximizing the empirical Shannon mutual information function between the feature being selected and class labels, given already selected features.

Efficient and Robust Feature Selection (RFS) is an efficient feature selection method proposed by [10] that exploits the noise robustness property of the joint $\ell_{2,1}$-norm loss, by applying the $\ell_{2,1}$-norm minimization on both the loss and its associated regularization function. The value of the regularization coefficient for our experiments was chosen by performing RFS on a wide range of values and picking the value that led to the highest accuracy on the validation set.

Feature Quality Index (FQI) utilizes the output sensitivity of a learning model to changes in the input, to rank features [11]. Unlike the proposed method, FQI uses the Mean Square Error (MSE) instead of the model's loss, and uses a one-shot feature ranking without iterations.

3 AMBER

3.1 Sensitivity of weights to features

Following a gradient-based optimization of a deep neural network, the weights connected to the neurons in the input layer that correspond to more salient features tend to have larger magnitudes [12]. Similar to FQI, we measure the relevance of each feature by setting the value of the corresponding neuron to 0. Hence, all the weights from that neuron to the next layer will not have an impact on the output. Since more salient features possess weights of higher magnitude, these weights influence the output to a greater extent and setting their values to 0 in the input will result in a greater loss in the output layer. We will refer to the loss value of this ranker model as a feature's **Relevance Score**. We note that this is the basis of the Weight Based Analysis feature selection methods outlined by [13]. We further note that we normalize the training set before training by setting the mean of each feature to 0 and the variance to 1, so that our *simulation of feature removal* is effectively setting the feature to its mean value for all training examples.

3.2 Autoencoders Reveal Non-Linear Correlations

Weights connected to a feature can possess high magnitudes, even when this feature is redundant in presence of other features. We experimented with methods like PCA and correlation coefficients [14], but these methods reveal only linear correlations in data. Autoencoders, however, reveal non-linear correlations [15]. To eliminate one feature from a set of k features, we train the autoencoder with one dense hidden layer consisting of $k-1$ neurons using the normalized training set. We note that this hidden layer can also be convolutional, LSTM, or of other types depending on the data type. To evaluate a feature, we set its corresponding values in the training set to 0 and pass the set into the autoencoder. We then take the Mean Squared Error (MSE) between the output and the original input, and perform this for each of the k features separately. Lower MSE values indicate higher likelihood for redundancy. We refer to this MSE as a feature's **Redundancy Score**.

Algorithm 1 AMBER Algorithm for Feature Selection

Inputs: k: Number of features to be eliminated; trainSet: Training Dataset;
Outputs: featList: List of k eliminated features

function AMBER(k, trainSet)
Train state of the art RM using trainSet
Initialize featList to empty list
for $i = 1$ *to* k **do**
Set rmSet as trainSet with all features in featList set to 0
Set autoTrainSet as trainSet where all features in featList are removed
Train autoencoder with one hidden layer containing $d-i$ units using autoTrainSet
for j *in* $d-i+1$ *features not in featList* **do**
Record loss of RM when rmSet is evaluated after setting feature j to 0
Set cTrainSet as autoTrainset where feature j is set to 0
Record MSE of autoTrainSet and output of autoencoder when cTrainSet is evaluated
End for
Normalize RM losses and MSEs and add corresponding values
Sort and add lowest scoring feature to featList
End for
return (featList)

3.3 Loss functional simulations to prevent retraining

To eliminate k out of d features, we first train a state-of-the-art neural network model for the considered dataset, which we call the **Ranker Model** (RM). We then *simulate the loss functional* by setting the input for each of the d features in all the examples of the training set to 0 one at a time in a round-robin fashion to obtain a list of d Relevance Scores. Additionally, we train the autoencoder and pass the similarly modified training sets through the autoencoder to obtain d Redundancy Scores. We then divide the Relevance and Redundancy Scores by their corresponding ranges so that they both contribute equally to the final decision and add them to obtain a **Saliency Score**. The feature with the lowest Saliency Score is eliminated. In the context of the RM, elimination means that the feature is permanently set to 0 for all training examples in further iterations. In the context of the autoencoder, elimination means that that feature is permanently removed. This entire process, without retraining the RM, is done iteratively k times. Note that each Saliency Score is indeed obtained through simulations of the loss functionals - of both the RM and autoencoder - by evaluating at functions insensitive to the considered feature. The pseudocode for AMBER is described in Algorithm 1. We note that it is straightforward to modify AMBER for selecting minimal feature sets for a given accuracy guarantee.

4 Results

4.1 Experimental Setup

We used three Nvidia Tesla P100 GPUs, each with 16 GB of memory, and Keras with a TensorFlow backend. With the exception of the RadioML2016.10b dataset for which we used all three GPUs, we only used one GPU for training. The experiments were performed three times and the average accuracies were plotted at each feature count in Fig. 1. [1].

4.2 Datasets and Classifiers

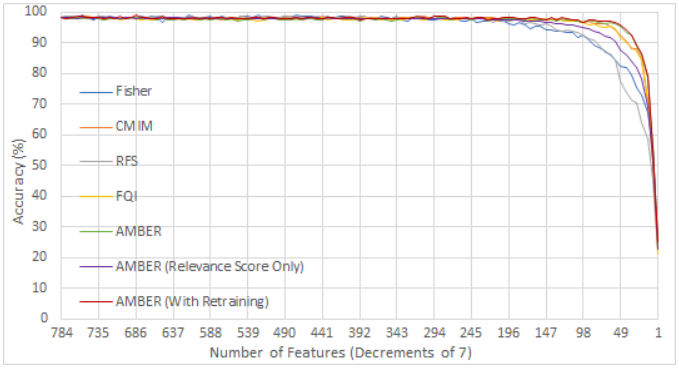
We consider different application domains to demonstrate AMBER's versatility. The final models, trained on the selected feature set, are common across all the feature selection methods that are compared and are trained until early stopping is achieved with a patience value of 5 to ensure that the comparisons are fair. For all classifier models, the softmax activation function is applied to the output layer with cross-entropy loss, and ReLU is applied to hidden layers, unless explicitly stated otherwise. All the autoencoders used have a dense hidden layer. We note that we obtained slightly better results with a convolutional and LSTM hidden layer for the MNIST and RadioML2016.10b datasets, respectively. The test split used for the Reuters and the Wisconsin Breast Cancer datasets is 0.2 while the test split used for the RadioML2016.10b dataset is 0.5. Some of the plots in Fig. 1 were jagged when feature counts in decrements of 1 were plotted and thus, we plotted them in larger feature count decrements. Finally, to demonstrate that the final model does not necessarily have to be the same as the RM used by AMBER, we used different models as the final model and the RM for the MNIST and RadioML2016.10b datasets.

MNIST is a handwritten digit recognition dataset consisting of 70000 28x28 grayscale images with 10 classes; 10000 of which represent the test set. The total number of features is 784. The RM is a CNN consisting of 2 convolutional layers, a max pooling layer, and 2 dense layers, in that order. The convolutional layers have 32 and 64 filters, in order of depth, with kernel sizes of (3x3) for both layers. The max pooling layer has a (2x2) pool and the dense layers have 128 and 10 (output layer) neurons. The final model used is an MLP model consisting of 3 fully connected layers with 512, 512, and 10 (output layer) neurons. Each of the hidden layers is followed by a 0.2 rate dropout layer.

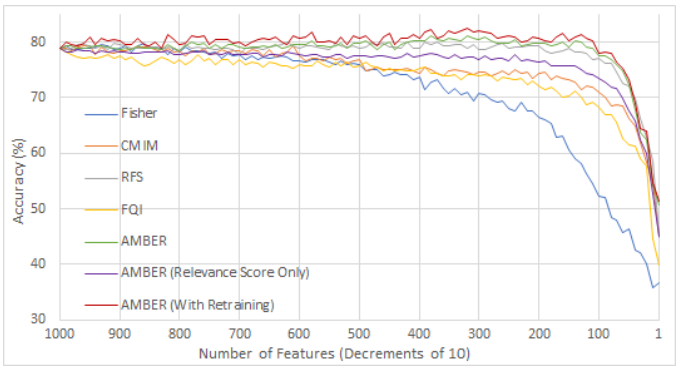
Reuters is a text dataset consisting of 11228 newswires with 46 classes, corresponding to different topics. Each wire is encoded as a sequence of word indices, where the index corresponds to a word's frequency in the dataset. For our demonstration, the 1000 most frequent words are used. The ranker and final models are the same MLP model consisting of 2 fully connected layers with 512 and 46 neurons.

Wisconsin Breast Cancer contains characteristics of cell nuclei, measured from an image of Fine Needle Aspirates (FNAs) of breast mass [16]. There are 569 examples with 30 features and 2 classes: malignant and benign. The ranker and final models are the same MLP having 4 fully connected layers with 16, 8, 6, and 1 (output layer) neurons, in order of depth. Sigmoid activation is used for the output layer.

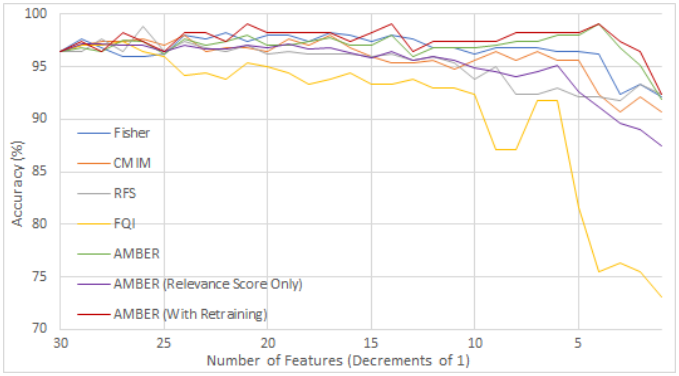
1. The source code for AMBER, links to the datasets considered, and the error bars for the comparison plots are available at https://github.com/alyelgamal/AMBER



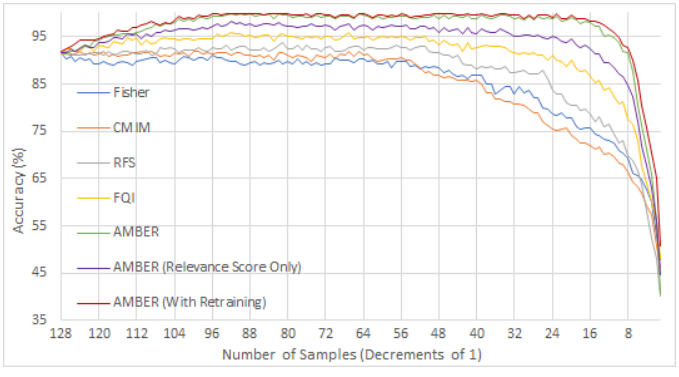
(a)



(b)



(c)



(d)

Fig. 1: Accuracy vs Feature Count plots for the final models trained with the selected features for the (a) MNIST, (b) Reuters, (c) Wisconsin Breast Cancer, and (d) RadioML2016.10b datasets.

TABLE 1: Accuracy Comparisons

Method	Avg. accuracy with top 10% features (%)			
	MNIST	**Reuters**	**Cancer**	**RadioML**
Fisher	88.37	51.21	92.40	73.42
CMIM	96.38	71.04	90.64	70.22
RFS	89.46	77.11	91.81	75.85
FQI	95.49	68.20	76.32	83.57
AMBER	**97.21**	**77.55**	**96.78**	**95.15**
- Relevance	93.29	73.45	89.65	89.54
- Retraining	97.21	78.11	97.37	97.49

TABLE 2: Time needed to rank all features in Seconds.

Method	MNIST	Reuters	Cancer	RadioML
AMBER	10552.24	21710.78	40.04	26417.53
- Retraining	24202.66	29005.08	739.01	42533.27

RadioML2016.10b consists of received wireless signal samples with 1200000 128-sample complex time-domain vectors and 10 classes, representing different modulation types [17]. It has 20 Signal to Noise Ratios (SNR), but we only show the results of the 18 dB data for better illustration. Each of the 128 samples consists of real and imaginary parts and thus, the input dimensions are 2x128, and the total number of features is 256. This dataset has the unique property that only pairs of features (belonging to the same sample) can be eliminated. AMBER, like FQI, is powerful in such situations as the pairs of features can be set to 0 to evaluate their collective rank. The other feature selection methods fail in this case because they account for feature interactions within the pairs of features as well, which is one reason for why AMBER outperforms them as it does not. For the other methods, to eliminate pairs of features belonging to the same sample, we simply added the scores belonging to the two features to obtain a single score for each sample. The Ranker Model used here is a CLDNN while the final model used is a ResNet, both of which are described in [18].

4.3 Classification Accuracies

The final models' classification accuracy plots using the selected features can be observed in Fig. 1. We observe the impressive performance delivered by AMBER that generally outperforms that of all four considered methods, particularly when the number of selected features becomes very low (about 99% average accuracy with 4 out of 30 features for the Cancer dataset and about 95% average accuracy with 16 out of 128 samples for the RadioML dataset). The comparisons of the accuracies of the final models using the top 10% of features are given in Table 1. The results in the last two rows refer to using a version of AMBER without the autoencoder's redundancy score, and another version of AMBER where the ranker model is retrained in every iteration, respectively. Note from the depicted results (purple curves in the figure) how solely relying on the RM significantly reduces AMBER's performance, which validates our intuition about the benefit of using the Autoencoder to capture correlations to reduce the generalization error. Further, we observe how negligible gains are achieved when retraining the RM in every iteration, that comes at a significant computational cost, as demonstrated in Table 2, which validates our intuition about simulating the loss functional without retraining for computational efficiency while maintaining good performance.

5 Conclusion

We presented a new wrapper feature selection method that relies on a task-based model and autoencoders to evaluate candidate combinations of features. The proposed method enjoys computational efficiency, as it avoids retraining the model in every iteration, unlike existing wrapper methods. We demonstrated the obtained superior performance in four different domain applications, and specially highlighted the added value of including autoencoders and negligible loss due to reusing the model without retraining.

References

[1] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.

[2] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Advances in neural information processing systems*, 2001, pp. 668–674.

[3] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–45, Dec 2017. [Online]. Available: http://dx.doi.org/10.1145/3136625

[4] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics*, vol. 2015, 2015.

[5] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[7] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[8] M. Vidal-Naquet and S. Ullman, "Object recognition with informative features and linear classification." in *ICCV*, vol. 3, 2003, p. 281.

[9] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine learning research*, vol. 5, no. Nov, pp. 1531–1555, 2004.

[10] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint $l_{2,1}$-norms minimization," in *Advances in neural information processing systems*, 2010, pp. 1813–1821.

[11] K. De Rajat, N. R. Pal, and S. K. Pal, "Feature analysis: Neural network and fuzzy set theoretic approaches," *Pattern Recognition*, vol. 30, no. 10, pp. 1579–1590, 1997.

[12] K. W. Bauer Jr, S. G. Alsing, and K. A. Greene, "Feature screening using signal-to-noise ratios," *Neurocomputing*, vol. 31, no. 1-4, pp. 29–44, 2000.

[13] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff, "Embedded methods," in *Feature extraction*. Springer, 2006, pp. 137–165.

[14] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[15] M. F. Balın, A. Abid, and J. Zou, "Concrete autoencoders: Differentiable feature selection and reconstruction," in *International Conference on Machine Learning*, 2019, pp. 444–453.

[16] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *Biomedical image processing and biomedical visualization*, vol. 1905. International Society for Optics and Photonics, 1993, pp. 861–871.

[17] T. O'Shea, J. Corgan, and T. Clancy, "Convolutional radio modulation recognition networks," in *Proc. International conference on engineering applications of neural networks*, 2016.

[18] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," *IEEE Machine Learning for Communications Emerging Technologies Initiatives, arXiv:1901.05850*, 2019.