Using Artificial Neural Networks to Perform Feature Selection on Microarray Data

Giuliano Armano and Osvaldo Marullo
DMI, University of Cagliari, Italy
Email: armano@unica.it, osvaldo.marullo@gmail.com

***Abstract*—This article illustrates a feature selection technique that makes use of artificial neural networks. The problem being faced is the analysis of microarray expression data, which requires a mandatory feature selection step due to the strong imbalance between number of features and size of the training set. The proposed technique has been assessed on relevant benchmark datasets. All datasets report gene expression levels taken from female subjects suffering from breast cancer against normal subjects. Experimental results, with average accuracy of about 84% and very good balance between specificity and sensitivity, point to the validity of the approach.**

***Index Terms*—feature selection, artificial neural network, microarray data, cancer prediction**

I. Introduction

Cancer prediction is a main research topic in biomedicine, and microarrays are still the preferred tool for gathering relevant gene expression data. Due to their robustness and effectiveness, artificial neural networks have been extensively used to generate models able to deal with this kind of problems. To deepen this research topic the reader can consult, for example, the review work of Daoud and Mayo [1]. In this survey, the authors summarize the research topic into i) filtering methods, aimed at extracting representations that best describe the gene expressions, ii) predicting methods, aimed at maximizing the adopted performance measures (typically accuracy), and iii) clustering methods, aimed at dividing the genes or samples according to their similarities.

Typically, the analysis of data derived from microarrays turns out to be very difficult, as any reference dataset can always be thought of as a list of gene expression values. In fact, with M number of examples and N number of features, when M << N the so-called "curse of dimensionality" problem arises. In this case the negative impact due to the overwhelming number of features can be dealt with by performing feature reduction, feature selection, or both. However, in the field of microarray analysis feature selection is preferred, a relevant side-effect of the analysis being the identification of one or more clusters of genes that are responsible for the phenotype. A number of techniques have been devised and proposed over time to perform feature selection on microarray data. In particular, a review article has been recently issued on feature selection for cancer prediction by Hambali *et al.* [2]. Here the authors points out that achieving good performance starting with the complete set of genes remains a great challenge, due to the high dimensions, small sample size, and presence of noise in gene expression data. Then the authors recall the most relevant feature selection techniques that have been devised to overcome this problem, pointing that feature selection methods are typically less critical and sensitive than feature reduction ones. A comprehensive taxonomy of the various feature selection methods used for microarray cancer classification is also proposed and used as reference framework for discussing each specific method. Further surveys have also been issued on the same topic –see in particular Almugren and Alshamlan [3], and Gunavathi *et al.* [4]. As for the feature selection tools available off-the- shelf, the interested reader may also consult the work of Tadist *et al.* [5]. The perspective of the cited work is that feature selection techniques are believed to become a game changer that can help to reduce the complexity of genomic data, thus making it easier to analyse it and translating it into useful information. This work is also clearly influenced by the discussion on eXplainable AI [6], [7], which tipped the scales in favor of selecting features rather than transforming them into another space. Despite the growing amount of research that points to feature selection as the main tool for contrasting the curse of dimensionality, there is still room for devising ad hoc solutions, in particular for those research fields in which the order of magnitude of N/M is at least 10^2. This article illustrates a feature selection technique, based on multilayer perceptrons (MLPs, hereinafter), which is framed within the broad category of "wrappers" [8]. To prevent the occurrence of a selected feature set strongly biased by a specific run of the algorithm, a greedy selection strategy is enforced which repeatedly makes use of a number of independent run steps. In so doing, the possibility of disregarding relevant features is strongly reduced. Experiments have been performed on

Manuscript received December 29, 2021; revised March 22, 2022.

©2022 Int. J. Pharm. Med. Biol. Sci.

doi: 10.18178/ijpmbs.11.3.54-58

five datasets (available on GEO) derived from experiments on microarrays related to patients with breast cancer. One dataset was used for training and validation, and the others for testing. Experimental results point out that the accuracy, averaged on all datasets used for testing, is about 84%. Moreover, a very good balance between specificity and sensitivity has been observed. Notably, although devised for the specific research topic described in this article, the proposed technique is general enough to be used also for other application fields affected by the curse of dimensionality problem.

II. Methods

This section is aimed at illustrating the proposed feature selection algorithm, including the heuristics used for assessing feature importance. Before going into details, let us make some general consideration on the underlying framework.

The specific problem to be faced regards a dataset consisting of about 400 examples, distributed on five datasets. Each dataset is characterized by more than 22k features (of course, each feature accounts for a specific gene expression measured on a microarray experiment). No doubt that for this problem the curse of dimensionality holds, making it difficult for every learning algorithm to come up with a useful generalisation. For this reason a suitable feature selection algorithm has been devised and implemented.

A. *Overview of the Proposed Feature Selection Algorithm*

The insight that drawn the definition of the proposed algorithm was to perform feature selection in several steps by progressively reducing the set of selected features. Starting with the full set of features, at each step a small fraction of features is dropped, while the others are passed on to the next step. The algorithm ends when the wanted number of features is reached. The corresponding function is shown in Listing 1. Note that the algorithm iterates over a pair of functions, i.e., *make_steps and drop_features*. The former is entrusted with evaluating feature importance across multiple steps of feature selection run in parallel, whereas the latter is entrusted with dropping a fixed percent of features (default 5%), using a consensus-based strategy. Fig. 1 reports the dependencies between the main function and the subsidiary ones. The diamond icon used in the figure highlights that a "part-of" relation holds between the connected boxes –e.g., the connector that links *make_FS* and *make_steps* asserts that the former repeatedly calls the latter.

```
def make_FS ( dset , max_feat=20 , numruns=50 , ** kwargs ) :
    # Perform feature selection using a greedy strategy
    selected = dset.features.copy ( ) # start with all features
    while len ( selected ) > max feat :
        rankings = makesteps ( dset , selected , num runs , ** kwargs )
        selected = drop features ( selected , rankings )
    return selected
```

Listing 1: Algorithm for feature selection (simplified listing). The algorithm starts with the full set of features, whose number is then progressively decreased using a greedy strategy. The algorithm repeatedly calls the functions make steps and drop features.

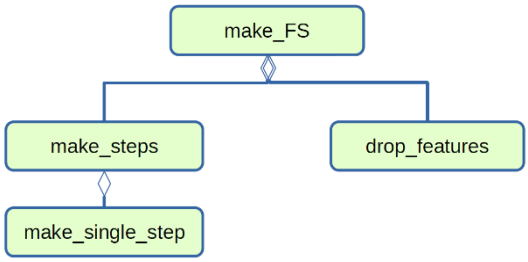

Figure 1. Function tree that highlights the role of each subsidiary function in the implementation of the proposed feature selection algorithm (here reported as *make_FS*).

The function *make_steps* embeds an iteration as well. This iteration is required to make the selection as much independent as possible from the "history" of training, as well as from the way training and validation set have been randomly split. In other words, multiple steps of feature importance evaluation are run to make the dropping of non relevant features more sound from a statistical perspective. A simplified version of the corresponding Python function is reported in Listing 2. The listing makes clear that a number of completely independent feature selection steps are run in parallel.[1] It is worth pointing out that each step gives rise to its specific ranking, so that all rankings are passed to the function drop features for further processing.

```
def make_steps ( dset , features , num_runs=50 , ** kwargs ) :
    #Perform multiple steps of feature selection ( default 50)
    rankings = [ None for k in range ( num_runs ) ]
    for k in range ( num runs ) :
        rankings [k] = make_single_step ( dset , features , ** kwargs )
    return rankings
```

Listing 2: Multiple steps of feature selection (simplified listing). Depending on the selected number of run steps (see the num runs parameter), a battery of feature selection steps is run. Each step gives rise to its specific ranking of features. All rankings are returned by the function for further processing.

The function *make_single_step* (see Listing 3) is the core of the computation. In fact, *make_FS* repeatedly calls make steps, which in turn repeatedly calls make single step. Given a dataset and the currently selected set of features,first *make_single_step* randomly splits the dataset into training and validation set (of course, only the selected features are preserved), then an MLP model

[1] In fact, this part of the algorithm has been run in parallel using the Ipyparallel Python library. A porting of the code to pySpark is on the way.

is created, characterized by a single hidden layer equipped with N*5% hidden nodes (with N number of selected features). After fitting the model with the training set, features are finally ranked according to an ad hoc strategy, which is described in the next subsection.

```
def make_single_step(dataset, features, **kwargs):
    #Perform a single step of feature selection
    train_set, valid_set = dataset.split(random=True)
    train_set = train_set.project(on=features)
    valid_set = valid_set.project(on=features)
    model = make_MLP_model(features, **kwargs).fit(train_set)
    return rank_features(model, valid_set)
```

Listing 3: Single step of feature selection (simplified listing). First, the given dataset is randomly split into training and validation set. Then an MLP model is created upon the training set and assessed on the specified features only. Finally, feature ranking is performed by looking at the difference between the full-fledged model and the performance observed after the removal of each specific feature.

B. *Assessing Feature Importance (on Single Step)*

To shed light on the adopted feature importance measure, let us consider the generic scenario in which only part of the full set of features has been retained, say *F*. Let us further assume that an MLP has been trained using part of the training samples, whilst the remaining ones are left for validation purposes (by default 30% of the training samples). Upon training, the output for each sample in the validation set is then calculated. Be *output(F)* the corresponding list of values. To evaluate the feature importance related to a generic feature *f* , one can clear any contribution of that feature inside the MLP. Be *output(F − f)* the corresponding list of values, which are the outputs obtained –sample by sample– by clearing up the contribution of *f* . The importance of each feature *f* can be easily calculated by summing up the absolute-value differences measured between *output(F)* and *output(F − f)*. Hence, with varying *f* , this function allows to establish the ranking of all features. In particular, a small value of the ranking function evaluated on a feature *f* would account for low feature importance, and vice versa. Listing 4 shows a simplified listing of the corresponding Python function.

```
def rank_features(model, valid_set):
    #Rank features according to their usefulness"
    outputs = model.eval_outputs(model.features, valid_set)
    ranking = dict()
    for f in model.features:
        fscore = model.eval_outputs(features.drop(f), valid_set)
        ranking[f] = sum(abs(outputs-fscore))  # L1 norm ...
    return sorted(ranking.items(),key=itemgetter(1),reverse=True)
```

Listing 4: Feature ranking for a single run is performed by looking at the difference between the full-fledged model and the performance observed after the removal of each specific feature. The function returns the sorted list of feature-ranking pairs, in descending order. Note that *features.drop(f)* is intended to denote the set of features in which f has been temporarily removed.

C. *Assessing Feature Importance (on Multiple Steps)*

In the previous subsection, the ad hoc strategy for assessing feature importance has been shown for a single feature selection step. However, the high-level feature selection algorithm must cope with multiple steps in parallel. To come up with an overall ranking built upon the individual rankings generated by each step, a consensus-based strategy had to be devised and implemented. Instead of listing out the Python source code, let us give an insight of the process commenting in words. Be *N* the number of parallel steps (default 50). Hence, a run of *make_steps* would generate *N* rankings, each evaluated using the previously described ad hoc strategy. For the sake of simplicity, let us concentrate on a single feature, say *f*. Looking for *f* across all available rankings, in general its position will vary, depending on the importance assigned to it in each specific ranking. However, less important features are expected to occur far from the first positions of each ranking and vice versa. The insight that stands behind the adopted consensus-based strategy is to sum up the positions in which *f* is found, being confident that big values would give strong support for disregarding the corresponding features and vice versa. Zooming out from the specific feature *f*, i.e., ranging over all selected features, one can easily draw the overall consensus-based ranking, to be used for dropping part of the available features. In so doing, the next multiple step can be run, starting over with a reduced set of features. Fig. 2 gives a snapshot of the consensus-based strategy devised for ranking features at each multiple step, which evaluates an overall ranking starting from all rankings generated by calling *make_step* (which in turn repeatedly calls *make_single_step*). Hence, the more a feature occurs on the upper side of each ranking on the left, the better will be its position in the overall ranking.

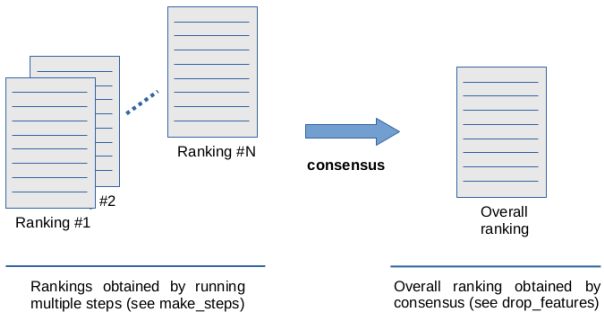

Figure 2. Snapshot of the proposed consensus-based strategy.

III. Experimental Setting and Results

Five datasets used in previous studies have been used as experimental work-bench. All datasets are downloadable from the Gene Expression Omnibus (GEO)

at NCBI [2]. Table I reports their characteristics and establishes a link between datasets and articles in which they have been analysed. The time to relapse of the disease and the grade of the tumour were considered for labelling. All datasets share the same set of features (which amount to 22,283). Note that only the first dataset has been used for feature selection and for training the classifier.

Considering a time span of five years and defining the degree of severity of the disease as S, individuals that have shown $S > 1$ have been labelled as positive, whereas those with degree $S \leq 1$ as negative. The dataset GSE4922 has been used as reference, being the largest and well balanced. In particular, this dataset has been used to perform feature selection and to train and validate each classification model. Feature selection and model training has been performed by taking a sample of GSE4922 for training and the remaining part as validation (with proportions of 70% and 30% respectively).

TABLE I. LIST OF THE MICROARRAY DATASETS USED TO PERFORM EXPERIMENTS

Dataset	Reference	TOT	POS	NEG	
GSE4922	Ivshina et al. [9]	117	61	56	FS + Train.
GSE1456	Pawitan et al. [10]	57	32	25	Test
GSE2990	Sotiriou et al. [11]	68	22	46	Test
GSE7390	Desmedt et al. [12]	59	34	25	Test
GSE6532	Loi et al. [13]	89	42	47	Test

TABLE II. EXPERIMENTAL RESULTS OBTAINED ON THE MICROARRAY BENCHMARK DATASETS USED FOR TEST

Dataset	Accuracy	Specifity	Sensitivity
GSE7390	0.78	0.64	0.88
GSE2990	0.85	0.87	0.82
GSE1456	0.88	0.92	0.84
GSE6532	0.84	0.87	0.81

The feature selection algorithm has been run once on the dataset GSE4922, giving rise to a small set of 25 selected features. Note that the order of magnitude regarding the ratio between the full set of features and the number of selected ones is about 10^3 (meaning that only 1 feature over 1000 has been selected). The selected features have been used as input to train a full model, still on GSE4922. In so doing, the system has been left completely unaware of the remaining datasets (i.e., GSE1456, GSE2990, GSE7390, and GSE6532), until testing. To leverage the statistical significance of results, in fact 10 MLPs have been trained (starting with the same set of 25 features found by the feature selection algorithm). Table II reports the average performances obtained on the benchmark test sets after repeatedly training an MLP with one hidden layer and equipped with 20 nodes on the GSE4922 dataset. The hidden layer uses ReLU as activation function, whereas the output layer used a sigmoid. The problem at hand being binary, each MLP was designed with a single output.

No significant variations on results has been evidenced along different runs. The table reports the averages of accuracy, specificity and sensitivity.

IV. CONCLUSIONS

This article has been focusing on a novel feature selection technique, to be used for improving the analysis of microarray expression data. The corresponding algorithm starts over with the full set of features and then drops part of them step-by-step, using a greedy strategy. To leverage the statistical significance of the selection, each step is in fact made up by several single independent steps run in parallel. Any such step uses the same ad hoc strategy to generate the corresponding feature ranking, whereas a consensus-based strategy has been devised and implemented to come up with an overall ranking at each "multiple" step. In so doing, feature removal is supported by a solid motivation, based on the ranking observed across many independent runs. The proposed algorithm allows to select only a small percent of features, with respect to the ones taken as input (the impressive feature selection ratio is about 99.9%). Downstream of the feature selection phase, an MLP has been trained using only the selected features and tested on relevant benchmark datasets. It is worth pointing out that different runs of training and testing have shown a robust behaviour, with an average accuracy steadily attested to 84%. A very good balance has been observed on each test set also for specificity and sensitivity. As for future work, we are planning to issue a release of the source code able to run with Apache pySpark. Additional ranking functions (e.g., based on the L2 norm) are also being experimented.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Osvaldo Marullo and Giuliano Armano have contributed to the research, data analysis and software programming. And both have approved the final version.

REFERENCES

[1] M. Daouda and M. Mayo, "A survey of neural network-based cancer prediction models from microarray data," *Artificial Intellgence in Medicine*, vol. 97, pp. 204-214, 2019.

[2] M. A. Hambali, T. O. Oladele, and K. S. Adewole, "Microarray cancer feature selection: Review, challenges and research directions," *International Journal of Cognitive Computing in Engineering*, vol. 1, pp. 78-97, 2020.

[2] https://www.ncbi.nlm.nih.gov/gds.

[3] N. Almugren and H. M. Alshamlan, "A survey on hybrid feature selection methods in microarray gene expression data for cancer classification," *IEEE Access*, vol. 7, pp. 78533-78548, 2019.
[4] C. Gunavathi, K. Premalatha, and K. Sivasubramanian, "A survey on feature selection methods in microarray gene expression data for cancer classification," *Research Journal of Pharmacy and Technology*, vol. 10, pp. 1395-1401, 2017.
[5] K. Tadist, S. Najah, N. S. Nikolov, F. Mrabti, and A. Zahi, "Feature selection methods and genomic big data: A systematic review," *Journal of Big Data*, vol. 6, no. 79, pp. 1-24, 2019.
[6] D. Gunning, Explainable artificial intelligence (XAI), Technical Report DARPA-BAA-16-3, Defense Advanced Research Projects Agency (DARPA), 2017.
[7] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *Proc. 41st International Convention on Information and Communication Technology, Electronics and Microelectronics*, Croatia, 2018, pp. 210-215.
[8] N. Kushmerick, D. S. Weld, and R. Doorenbos, "Wrapper induction for information extraction," in *Proc. IJCAI-97*, 1997.
[9] A. V. Ivshina, *et al*., "Genetic reclassification of histologic grade delineates new clinical subtypes of breast cancer," *Cancer Research*, vol. 66, pp. 10292-10301, 2006.
[10] Y. Pawitan, *et al*., "Gene expression profiling spares early breast cancer patients from adjuvant therapy: derived and validated in two population-based cohorts," *Breast Cancer Research*, vol. 7, no. 6, article R963, 2005.
[11] C. Sotiriou, *et al.*, "Gene expression profiling in breast cancer: Understanding the molecular basis of histologic grade to improve prognosis," *Journal of National Cancer Institute*, vol. 98, pp. 262-272, 2006.
[12] C. Desmedt, *et al.*, "Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the TRANSBIG multicenter independent validation series," *Clinic Cancer Research*, vol. 13, pp. 3207-3214, 2007.
[13] S. Loi, *et al*., "Definition of clinically distinct molecular subtypes in estrogen receptor-positive breast carcinomas through genomic grade," *Journal of Clinical Oncology*, vol. 25, no. 10, pp. 1239-1246, 2007.

Copyright © 2022 by the authors. This is an open access article distributed under the Creative Commons Attribution License (CC BY-NC-ND 4.0), which permits use, distribution and reproduction in any medium, provided that the article is properly cited, the use is non-commercial and no modifications or adaptations are made.

Osvaldo Marullo is PhD student in Information Technology at the University of Cagliari. His major research interests are on artificial neural networks applied to bioinformatics and on feature ranking/selection algorithms.

Giuliano Armano is associate professor of computer science at the University of Cagliari. His major research interests are on artificial neural networks applied to bioinformatics and on classifier / feature performance measures.