"Deep feature selection"

Ayouaz, Rayan

ABSTRACT

Over the years, there has been an increase in importance regarding the procedure of selecting features in machine learning models. While the number of applications of deep learning are multiplying in a wide variety of disciplines, the need for interpretability grows as well, asked by experts who are confronted to the use of deep learning in their works. The intrinsic way that feature selection can prevent overfitting of statistical models by taking into consideration relevant features only can be viewed as a really important step for regularization and is probably one of the most powerful option to really make a model more accurate and relevant to model reality. In another perspective, feature selection can be used in a more pragmatic way to reduce dimensionality of a problem as part of a preprocessing procedure. In this work, we will mainly focus on embedded feature selection through deep models. Such models refer to deep neural networks achieving feature selection during the training process. In most cases, these models are simple to illustrate and intuitive to understand. These methods are still suffering from a lack of theoretical results and there is then still a lot of work and experiments to make before such structures will be integrate to state-of-the-art models. We will observe how these models behave with different tuning and compare them with the expectation that some of them will lead to a significant reducing of the dimensionality while maintaining the performance of the models.

CITE THIS VERSION

Ayouaz, Rayan. *Deep feature selection.* Ecole polytechnique de Louvain, Université catholique de Louvain, 2021. Prom. : Dupont, Pierre. http://hdl.handle.net/2078.1/thesis:32970

École polytechnique de Louvain

Deep feature selection

Author: **Rayan Ayouaz**
Supervisor: **Pierre Dupont**
Readers: **Victor Hamer, Nicolas Golenvaux**

Academic year 2020–2021
Master [60] in Computer Science

Master Thesis: Deep Feature Selection

A review over new embedded methods

Rayan Ayouaz

Contents

1. Abstract

Over the years, there has been an increase in importance regarding the procedure of selecting features in machine learning models. While the number of applications of deep learning are multiplying in a wide variety of disciplines, the need for interpretability grows as well, asked by experts who are confronted to the use of deep learning in their works. The intrinsic way that feature selection can prevent overfitting of statistical

models by taking into consideration relevant features only can be viewed as a really important step for regularization and is probably one of the most powerful option to really make a model more accurate and relevant to model reality. In another perspective, feature selection can be used in a more pragmatic way to reduce dimensionality of a problem as part of a preprocessing procedure. In this work, we will mainly focus on embedded feature selection through deep models. Such models refer to deep neural networks achieving feature selection during the training process. In most cases, these models are simple to illustrate and intuitive to understand. These methods are still suffering from a lack of theoretical results and there is then still a lot of work and experiments to make before such structures will be integrate to state-of-the-art models. We will observe how these models behave with different tuning and compare them with the expectation that some of them will lead to a significant reducing of the dimensionality while maintaining the performance of the models.

2. Introduction and literature review

2.1. Machine learning

Machine learning is a domain at the frontier of mathematics and computer science where we study computer algorithms that improve automatically by the use of data. [Mitchell 1997]

The use of a statistical model in order to predict data or more generally to make decision, is called inference. With that kind of algorithm we try to resolve two sort of problems: regression in which the model predict a real-value with respect to a set of inputs and classification in which the model assigns a class to a set of inputs.

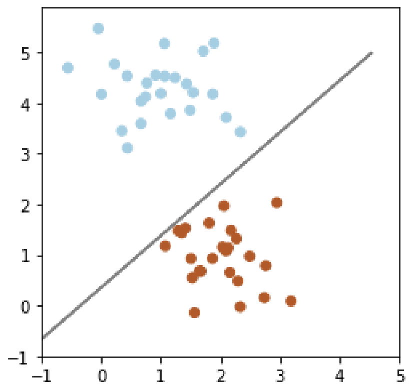


Figure (1) A linear classifier for a two-class problem in a two-dimension space

We can distinguish two problems: (1) unsupervised learning in which the model learns without knowing the value or the class associated with the inputs during the training and (2) supervised learning where the inputs are assigned to a class in the case of classification or to a real-value in the case of regression. During the learning process a model uses these information to asses its own performance and update its parameter in order to improve its performance.

In this work we will mainly focus on classification for supervised learning tasks. Figure 1 is an illustration of that particular task.

2.2. Deep learning

Due to its recent state-of-the-art achievement in various domains of research such as computer vision and language processing and the advancement in hardware equipment, deep learning – and more generally, neutral networks – has become a very important field in machine learning.

Neural networks have their own lexical in which we can find the terms: neurons, layers and connections. This lexical is as it seems partly inspired from biology but the comparison stops at the conceptual level as neural networks are first of all mathematical objects. The neural networks are distinguished by their structure which involves mainly the number of layers and the number of units for each layer. The first layer have as many units that there are inputs to the problem. The number of inputs units represent thus the original dimension of our data. The last layer have as many units as their are classes assigned to our problem in the case of classification.

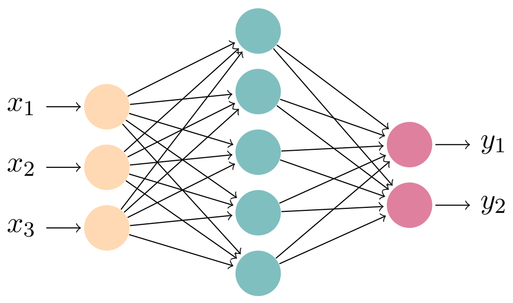


Figure (2) A simple neural network architecture with one hidden layer, a 3-dimensional input and two classes

Efficiency of neural networks are generally interpreted in terms of the universal approximation theorem [Haykin 1999] which stands for a network with an arbitrary number of hidden layers that can be capable of approximate any functions.

Shallow and deep networks The first general, working learning algorithm for supervised, deep feed-forward learning was the multi-layer perceptron. It was published by Alexey Ivakhnenko and Lapa in 1967. It comes itself from the original perceptron model with only one hidden layer. This original perceptron algorithm produce a linear discriminant in a feature space between two labels as it is described by this function:

$$f(x) = \begin{cases} 1 \text{ if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

Where w is a vector of real-valued weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product: $\sum_{i=1}^{m} w_i x_i$ (where m is the number of inputs to the perceptron) and b is the bias. The bias shifts the decision boundary and is not dependent of the inputs.

The main issue with a shallow and wide model, is that the networks have a strong memorization of the training examples but have no strong ability to generalize regarding the prediction of new unused during the training phase examples.

There has been empirical proof that adding layers could lead to a better generalization of new examples and thus, lead to an increase in accuracy. Multiple theoretical hypothesis has been advanced to understand these observations such as the manifold hypothesis [Fefferman u. a. 2013]

In modern deep learning algorithms, an activation function is added on each neuron in order to catch non linear representations of the data through the hidden layers. Recent works in the field massively use the ReLu (recitfied linear unit) activation function which have been adopted thanks to its performance confronted to the problem of the vanishing gradient. [Glorot u. a. 2011]

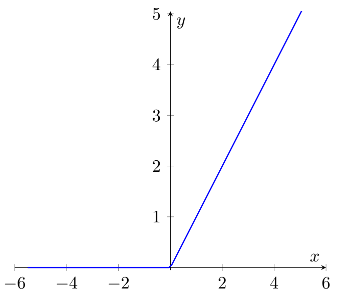


Figure (3) ReLu(x): max(0,x)

Such a model of neural network can be modeled like this:

$$f_k = act(b_k + W_k f_{k-1}) \tag{2}$$

Where $f_0 = X^G$ is the input of the neural net, f_k (for k > 0) is the output of the k_{th} hidden layer, which has weight matrix W_k and offset b_k of the k_{th} layer. "act" is the activation function of each neuron.

Notations In the next sections we will use the following notation:

- W_0 is the weight tensor of an additional structure at the entry of a dense structure.
- W_1 is the weight tensor of the dense structure.
- w_{ij} is the weight of the connection between the i_{th} input neuron connected to the j_{th} neuron of first hidden layer of the dense structure.
- X represent a set of input samples that we use to train or test the model.
- x_i represent a particular feature (a column in a matrix representation) of this set.

- Y is the label vector.
- $\widehat{Y}$ is the vector of predictions of the model.
- y_i is a particular label of the dataset.
- y_i' is a particular prediction.
- L is a loss function.
- N_{hid} is the number of neurons of the first hidden layer of the dense structure.
- N is the number of dimension of the training examples.
- M is the number of class.
- n is the number of training examples.
- s_i is a particular example from the training examples.
- S_k is the sum of all the weights that connect the neurons of the k_{th} layer to the next layer.
- $\odot$ is the element-wise multiplication.
- var denotes the variance function.
- P is the level of a particular layer of a neural network and w^P, b^P are the weights and bias of this layer.

Gradient descent The most standard method to adjust the weights of a neural network is the gradient descent. Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differential function.

It is proceeded with steps by calculating the gradient of a loss function of the model which can be evaluated with respect to various criteria. Many iterations are then required. This number of iteration is indeed a very important meta-parameter of our model as the gradient linearly approximate the loss function locally, a wise choice of the magnitude of the updating of the weights are required to avoid that the descent remains stuck in a local minima.

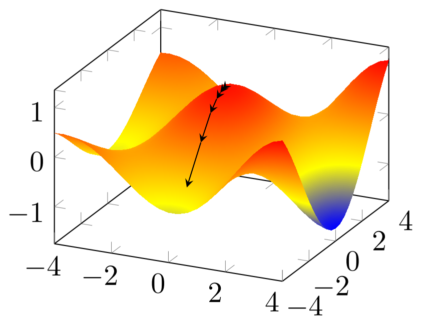


Figure (4) Representation of the gradient descend algorithm finding a local minima of a function.

The number of training examples that we take into account for computing the gradient is also an important meta-parameter. This number is set up in a process that is named mini-batch gradient descent.

Stochastic Gradient Descent [Bottou] and simulated annealing have been massively adopted in neural network models to avoid the local minima problem and increasing the chances of finding the global minima.

Gradient backpropagation Gradient backpropagation was introduced by [LeCun u. a. 1989] to have a general solving procedure to compute the gradient through a model with multiple layers. It uses the chain rule of calculus to calculate derivative. This procedure allows a lot of freedom to construct any architecture of deep learning model. Since, numerous architectures have been invented such as CNN, RNN and LSTM networks, some of them are standard today for solving particular machine learning problems.

$$C = \frac{1}{2}(y' - y)^2 \tag{3}$$

$$y' = \sigma(\underbrace{w^P a^{P-1} + b^L}_{z^P}) = \sigma(z^P) \tag{4}$$

$$\frac{\partial C}{\partial w^P} = \frac{\partial z^P}{\partial w^P}\frac{\partial a^P}{\partial z^P}\frac{\partial C}{\partial a^P} = a^{P-1}\sigma'(z^P)(a^P - y) \tag{5}$$

$$\frac{\partial C}{\partial a^{P-1}} = \frac{\partial z^P}{\partial a^{P-1}}\frac{\partial a^P}{\partial z^P}\frac{\partial C}{\partial a^P} = w^P\sigma'(z^P)(a^P - y) \tag{6}$$

$$\frac{\partial C}{\partial w^{L-1}} = \frac{\partial z^{P-1}}{\partial w^{P-1}}\frac{\partial a^{P-1}}{\partial z^{P-1}}\frac{\partial C}{\partial a^{P-1}} = a^{P-2}\sigma'(z^{P-1}) \times w^P\sigma'(z^P)(a^P - y) \tag{7}$$

Equation (7) shows how to compute the gradient of a specific parameter from (3) by retro-propagate it through (4), (5) and (6).

2.2.1. Regularization

Overfitting refers to a model that models the training data too well. It happens when a model learns the details and noise in the training data and therefore, the model will badly interpret new examples.

Regularization is a very important aspect of deep model as they are very affected by overfitting. For that, many methods have been proposed which often consists of the addition of terms in the loss function to constraint weights to small values.

Equation (8) define the l1-regularization where the l1-norms of the weights are added to the loss function. It has the effect of reducing the number of parameter used by a particular layer by pushing the weights to 0.
Equation (9) define the l2-regularization where the l2-norm of the weights are added to the loss function. It has the effect of avoiding that weights takes too large values which can conduct to overfitting.

$$Loss = L(Y - \widehat{Y}) + \lambda_1 \left\| S_k \right\|_1 \tag{8}$$

$$Loss = L(Y - \widehat{Y}) + \lambda_2 \left\| S_k \right\|_2 \tag{9}$$

Where k is a particular layer of the deep model. The value λ in both equations are then hyperparameters of the model.

2.2.2. Assessment of a model

Accuracy Accuracy is the ratio of correctly classified inputs in the context of classification.

Mean square error loss function Mean squared error is used in the context of regression problems and it is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y'_i - y_i)^2 \tag{10}$$

Cross-entropy loss function Cross-entropy is particularly useful in classification problems since it gives us an error express over all classes and tells us about the confidence of the prediction.

$$H_{y'}(y) := -\sum_{i}^{M} y'_i \log(y_i) \tag{11}$$

Where log is the natural logarithm.

BCR

- Precision is defined as the rate of true positive (TP) + the rate of false positive (FP)
- Recall is defined as the rate of true positive (TP) + the rate of false negative (FN) and represents the proportion of true positives retrieved amongst all the real positives
- Balanced classification rate,"BCR" is then defined by:

$$BCR = \frac{1}{2} * (\frac{TP}{\text{Recall}} + \frac{TN}{\text{Precision}}) \tag{12}$$

2.2.3. Curse of dimensionality

As deep learning models may use thousands or millions of inputs (for instance in computer vision) one of the main problem of high-dimensional data analysis is the curse of dimensionality. It refers to the fact that there are various phenomena that arise in high dimensions which do not occur in lower-dimension. Besides of being counter intuitive, these mathematical results brings a lot of difficulties. For instance since the volume of an hyper-sphere tends to zero as the dimensionality tends to infinity, it is easy to understand why such phenomena can be problematic for models which use metrics to compute distance between points in a space.

The struggle it implies is that there is never enough data to justify the robustness of a model when it comes to high-dimensional dataset. Another complexity of having a very large number of features is that if there are more features than observations then we run the risk of massively overfitting our model.

2.3. Feature selection

Feature selection is the method of selecting a subset of relevant features in order to reduce the dimensionality of our problem. It has other reasons to be applied, for interpretability of results for instance or when the model have a very high number of features and we want to isolate the relevant ones. [Kumar und Minz 2014]

Reducing the number of features can also lead to better accuracy on the test set as the irrelevant features can be considered as undesirable noise that perturb the training process. Reducing dimensionality help to prevent problems related to the curse of dimensionality.

Feature selection can be achieved separately as a preprocessing step or it can be computed during the training process. Such an approach is called an embedded method. Our work will focus particularly on this approach applied to deep leaning models and compare it with other methods such as we will detail.

2.3.1. Formal definition

Feature selection can be defined as the process of finding relevant features.[Blum und Langley 1997] A formal definition of relevance is then required.

Definition 2.1 (Relevance)**.** A variable $V \in X$ is relevant if and only if there exists a subset $B \in X$ such that $V \not\perp Y|B$. A variable is irrelevant if it is not relevant.

Relevant features can be further divided into strongly relevant and weakly relevant categories:

Definition 2.2 (Strong relevance)**.** A Variable V is strongly relevant if and only if $Y \not\perp V|X\backslash V$. A variable V is weakly relevant if it is relevant but not strongly relevant.

Feature selection can also be associated with two formal problems: the All-relevant problem and the minimal optimal problem. [Nilsson u. a. 2007]

Definition 2.3 (Relevance problem)**.** • All-relevant problem. Which consists in finding all relevant features. • Minimal optimal problem. Which consists in finding a subset $M \subseteq X$ such that $Y \perp X \setminus M|M$ and such that no proper subset of M satisfies this property.

2.3.2. Wrapper approach

A wrapper is a very intuitive solution to the problem of feature selection as it can be defined as a procedure in which we train the models multiple times with different subsets of features then we observe the difference between the scores obtained.

As the cardinal of all subset of a set equal 2^N (where N is the number of elements), exhaustively compute results for all the subsets is not desirable. It is indeed a np-hard problem. The problem mentioned here is the reason why the generation of subsets is often an heuristic search in which each state is a subset in the search space. These algorithms needs to establish a strategy to determine the successors of each states and the search organization.

This approach for achieving feature selection can then defined by four key steps:

- Subset Generation
- Evaluation of subset
- Stopping Criteria
- Result validation

The searches can be sequential (by selecting only one among all successors), the time complexity is then in $O(N)$ where the constant time unit is the time to fit the model once. The sequential search is not optimal and the trivial exhaustive search is the exponential search. Heuristics like Branch and Bound heuristic can help to reduce the search space and maintain its order as $O(N^2)$ for an exhaustive search.

[Kumar und Minz 2014]

Many independent criteria to evaluate a subset during the search procedure was proposed in the literature such as Distance measures [Almuallim und Dietterich 1994], Uncertainty measures [Verikas und Bacauskiene 2002], Dependency measure, Intercalss Distance Measures, etc..

A general method for a wrapper method is that the subset grows greedily with respect to the criterion used for subset generation. On each generation the model is fitted with this subset and the score obtained is saved. A stopping criterion could be to stop the search when the score stop to increase.

With such order of complexity, using a wrapper with deep models can be problematic, even with sequential search as the number of features can be very large.

2.3.3. Filter approach

A property of feature is to contain information about the different classes in the data. Such a property can be considered as feature relevance. Obviously and as [Law u. a. 2004] defined it "A feature can be regarded as irrelevant if it is conditionally independent of the class label".

In consequence, some statistical tests to evaluate the dependence of a feature to the targets labels can be used to perform a feature selection as a preprocessing step.

We call a filter the association of one particular statistical test and a threshold to discriminate which feature is relevant.

Following [Degeest u. a. 2019], such a filter may have six desirable property. The fist two relates to the performance of such filters:

- Ability to detect nonlinear relationships
- Ability to detect multivariate relationships

As the statistical criteria can usually not be evaluated exactly, the next properties refer to the quality of an estimator of this criteria.

- Estimator behaviour
- Estimator parameters: Filter estimator often requires an optimization parameter to be tuned. Therefore the influence of this parameter on the quality might be important.
- Estimator behaviour in small dataset
- Invariant estimator: the invariant should estimate each feature in a equivalent manner and not depend of the feature itself.

We will here describe three common filters as it will serve as a baseline evaluation for others approach.

2.3.3.1. Mutual information Mutual information comes from the information theory of Shannon [Shannon 1948]. It measures for two variables, how much knowing one of these variables reduces uncertainty about the other. It is define by (15).

$$MI(x_i, Y) = \log \frac{p(x_i, Y)}{p(x_i)p(Y)} \tag{13}$$

Where p is the density distribution of the join probabilities.

2.3.3.2. Pearson correlation The Pearson correlation coefficient [Guyon und Elisseeff 2003] is defined as

$$\rho = \frac{\mathrm{cov}(x_i, Y)}{\sigma_{xi}\sigma_Y} \tag{14}$$

Where cov(x_i, Y) is the covariance between the feature and Y and σ represent the standard deviation. It can only detect linear dependencies between variables and targets. An estimator of this coefficient can be:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{15}$$

Where $\overline{x}$ is the mean value of all observations for the feature.
It is mostly used in the context of regression problems.

2.3.3.3. RelieF Relief is a filter which assigns a score to each feature. In order to achieve this, this filter search through a neighborhood of samples of the same classes and for each feature, if a feature value difference is observed (in case of continuous data that difference is triggered with a particular threshold), that pair of sample of the same class is called a "hit" for this feature and the feature score decrease. On the other hand if a feature difference is observed and two samples do not share the same class the feature score of this particular feature increases.

ReliefF is a variant of Relief which search for k near misses from each different class and averages their contributions for updating the feature importance. The results are weighted with the prior probability of each class. It is particularly suitable for multi-classes problems.

2.3.4. Union of multiple filters

As [Mares u. a. 2016] shows taking the union of multiple feature selection methods (by adding their score to produce a general feature ranking) can lead to an improvement of selection sensitivity and thus performing a more robust selection. This can be explained by the fact that each method catch some relation that others don't catch. As filters are particularly submit to this kind of bias, we choose to use that assumption in our experiments as we will detail in the upcoming section.

2.3.5. Embedded approach with DNN

The idea behind an embedded approach is to compute the feature selection during the training process of the model. It often requires the adding of some extra penalties terms of the cost function.

In our work, we chose to focus our experiments to a very particular kind of embedded methods which exploit the structure of the deep models in order to achieve an embedded feature selection. We will focus on a few and unveil the concepts behind intuition.

2.3.5.1. Regularized network One of the most straight-forward method is the usage of specific penalties that use the L1-norm on the first (dense) layer of a DNN in order to minimize the contribution of some connections between the input layer and the first hidden layer. Such a modification of the cost function during back-propagation push the unnecessary weights to 0. These constrains will lead to a more accurate feature utilization of the network. [Verikas und Bacauskiene 2002]

A FS method in [Sun u. a. 2017] selects important variables on a feed-forward network by using L_1 regularization. Others methods use L_1 or $L_{1/2}$ penalty terms.

For achieving feature ranking on such a model, we can use a technique named activation potential analysis.

2.3.5.2. Activation potential analysis [Roy u. a. 2015] proposed a method to better interpret the trained weights of a model as a feature ranking. Their work shows that their method is more robust than another one named sensitivity analysis in which the effect on the accuracy is analysed after that each input is perturbed, in order to achieve feature ranking by selecting feature on which the network depend the most. Since this method is not accurate on networks with more than one or two layers they proposed to study the contribution of an input to the activation(averaged over all training values) of each hidden neuron in order to determine how much each input contribute to the predictions.

This method allows us to correctly interpret the trained weights of any network.

Remember that x_i is the i_{th} dimension of the input example x connected to j_{th} hidden neuron by w_{ij} and b_j the bias of hidden neuron j, the activation potential is calculated as

$$a_{ij} = w_{ij} * x_i + b_j \tag{16}$$

Then the average absolute activation potential contributed by the i_{th} dimension of M training examples $x^{(1)}, x^{(2)}, ..., x^{(k)}, ..., x^{(M)}$ connected to j_{th} hidden neuron is given by:

$$p_{ij} = \frac{1}{n} \sum_{k=1}^{n} |a_{ij}^{(k)}| \tag{17}$$

Where k represents k_{th} training example $x_{(k)}$. Afterwards, they compute the relative contribution of the i_{th} input dimension towards the activation potential of j_{th} hidden neuron using the following formula:

$$c_{ij} = \frac{a_{ij}}{\sum_{i=1}^{N} |p_{ij}|} \tag{18}$$

Where N is thus the dimension of input example x. The net positive contribution c_i^+ of an input dimension i over all hidden neurons is given as:

$$c_i^+ \sum_{i=1}^{N_{hid}} ReLU(c_{ij}) \tag{19}$$

The authors of the paper tested this technique in the context of a human gesture videos dataset. They observed that the results obtained, while reducing dimension with respect to the feature ranking of the activation potential technique, conducts to better results than applied a PCA transformation before training the same model. They did not provide a theoretical support for their analysis.

However, on their experiment the very most important features has a significantly higher action potential that the less important feature. The action potential of the top 10 features was five times larger than the action potential of the next features.

2.3.5.3. Pairwise regularized network [Li u. a.] use a sparse one-to-one layer between the input layer and the first hidden layer. The weights of each connection allows then to achieve feature ranking. The network is trained with elastic-net which is a convex combination of L1 and L2 regularization. This layer can also be trained with LASSO regularization as we will experiment next.

They proved that their model named DeepFS (or DFS) is not equivalent to a LASSO regularization of a first dense layer. They observed that their model can lead to the same accuracy as the LASSO regularized model while selecting less features. They assert that DFS is capable of recovering both linear and nonlinear features. They also constrained the weights to be positive as the sign of the weights is insignificant for the purpose because it can effectively be reversed by later layers.

Without taking in consideration activation function, the network can be modelled by:

$$y = W_1 \cdot W_0 \odot X \tag{20}$$

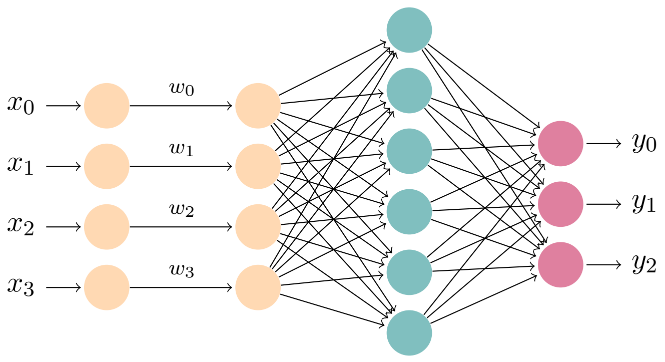


Figure (5) A dense network with a pairwise component.

They found that the model selected similar features as a random forest classifier while being as fast. Some difference were observed, they explained that the difference can be explained because the pairwise model considers the dependency of features while random forests independently measures how removing a feature affect the model.

They also questionned how much the influence of the purity (a indicator of the entropy) of the mini-batch used for learning is important and affects results but they left this answer for future works.

Their results were obtained in the context of a regression problem.

CancelOut The CancelOut layer introduced in [Borisov u. a.] works also on the principle of a one-to-one layer placed after the inputs. They added an activation function on these one-to-one neurons in order to easily constraint the weights (with a sigmoid function as it happens) and obtaining a more stable ranking. In order to stimulate the learning of the feature ranking they added a new regularization term in the loss function which uses the variance of the one-to-one weights to stimulate diversity. They also used L_1 penalty to introduce sparsity in the CancelOut layer.

The general loss function becomes:

$$\mathscr{L} = L(X, Y) - \lambda_1 var(\frac{W_0}{N_v}) + \lambda_2 \left\| \frac{W_0}{N_v} \right\| \tag{21}$$

Where λ_1 and λ_2 are meta-parameters of the model.

Without taking into consideration the activation function after the one-to-one layer the network can be modelled by:

$$y = W_1 \cdot g(W_0) \odot X \tag{22}$$

Where g is a function such that $g : R^n \implies R$.

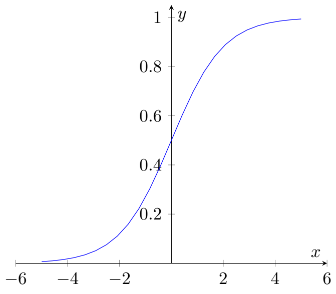


Figure (6) The sigmoid function

They experienced that their models was able to learn a model with equivalent accuracy of the simple pairwise structure with less feature. They also affirm that non-linear relationship between features and targets was learned more easily thank to the non-linearity of the sigmoid function.

2.3.5.4. Feature selection with a dense embedded component [Huang u. a. 2020] integrate the idea of achieving feature ranking into a deep model to select feature and they proposed their own architecture. Unlike prior sparse learning-based models with regularization terms, they used a deep model to map the features to their feature score and then use a pairwise layer to connect the two structures while training them at the same time.

In order to use the feature scores at the entry of the classification component, the outputs of the first model needs to be multiplied by their feature score. It allows the scoring to be consistent with the average value of a particular feature of the training set.

To introduce sparsity in the second component, the activation function of the last layer of the first component is designed as a linear threshold (since a thresholded version of ReLu is not suitable as it will filter negative inputs that the model may need in order to do predictions). The output of the first component can then be modelled as:

$$Z = TL(ReLu(XW_{01})W_{02})) \tag{23}$$

Where TL denotes the thresholded linear activation function.

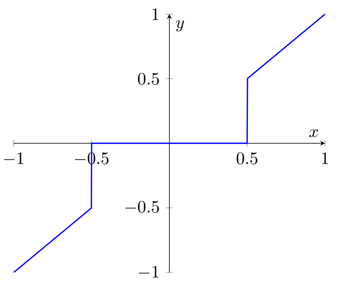


Figure (7) TL(x): $\begin{cases} x, x > \theta \text{ or } x < -\theta \\ 0, -\theta \leq x \leq \theta \end{cases}$

The loss function becomes then (for simplicity of the explanation we use here the MSE loss function):

$$loss = -\frac{1}{2n}\sum_{i-1}^{n} \|y_i - \hat{y}_i\|^2 = -\frac{1}{2n}\sum_{i-1}^{n} \|g(W_2(s_i \odot TL(W_1 ReLU(W_0 s_i))) - \hat{y}_i\|^2 \tag{24}$$

Where g is a unknown activation function of the classification layer (assuming there is only one hidden layer in the classification network).

The feature score can be obtained by forward propagation of the first fully-connected network. For n examples the feature score can be computed as

$$f_i = \frac{1}{n}\sum_{k=1}^{n} z_{k,i} \tag{25}$$

Where $z_{k,i}$ is the output of the first connected structure of the k_{th} sample for i_{th} feature.

They demonstrated the effectiveness of this method empirically. It also permits to not add additional parameters in the loss function (which can increase the difficulty of model optimisation) while still catching nonlinear relation between features. Following the authors, the features selected had a low redundancy rate. They proposed to analyse the benefit of a soft shrinking threshold function instead of the current hard-threshold. They observed that the structure was able to sometimes help to increase performance.

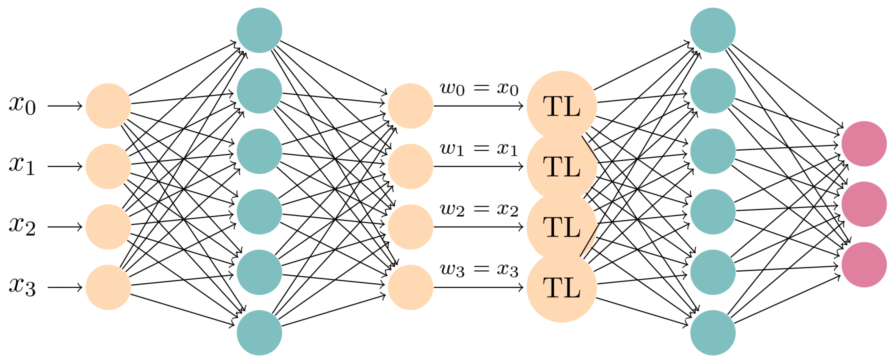


Figure (8) A dense network with an additional dense component to achieve feature selection.

2.3.5.5. Sparse layer with normalizing constraint Instead of introducing sparsity with a one-to-one layer, the work of [Bugata und Drotar 2020] uses a dense layer after the input layer with a special regularization in which the sum of each connection from an input is constrained to be less or equal than 1 and the variance of all incoming connections into each neurons of the so-called FSLayer is constrained to be more than 1.

If the FSLayer contains less neurons compared to the input layer, they proved that this layer operate as a feature selector since the weights are thus constrained to operate for this purpose.

The main constraints are then:

$$\forall_{k=1}^{M} var(A_k) < (\sum_i w_{ij})^2 \wedge \sum_i |w_{ij}| \leq 1 \wedge var(A_k) \geq 1 \tag{26}$$

Where A_k is the vector of all the weights that connect the k_{th} input to the next layer.

2.3.5.6. LassoNet The traditional Lasso regularization only applies to linear models. The architecture of lassoNet [Lemhadri u. a. 2021] however consist of a single residual connection that goes from the input to the output to by-pass a dense component which is used as an universal approximator.

The key idea is to budget the amount of linearity (with the M parameter) through this residual layer by adding a constraint in the loss function. The lassoNet objective function is then defined as:

$$\text{minimize } L(\theta, W) + \lambda \left\|\theta\right\|_1 \text{ s.t } \forall_{i=1}^{M} \left\|w_{ij}\right\| \leq M|\theta_i| \tag{27}$$

Where θ is the vector of the weights of the residual layer associated with each feature.

This has multiple advantages. It promotes the linear component of the signal above the nonlinear one and it serve to guide sparsity into the network. These components are then trained simultaneously.

In order to provide a feature ranking, the authors used a special procedure that they named proximal approximator that is run with respect to a parameter path in which they applied multiple times the traditional gradient descent while increasing the alpha parameter in order to make the regularization harder until no feature will be selected. The proximal approximator procedure update at each iteration the feature ranking and then provide at the end of the procedure the final one.

This procedure is computationally efficient and this so-called regularization path can be trained at a cost similar to the one of fitting a single model. Indeed, the complexity of the hierarichal proximator is controlled by $O(plog(p))$ where $p = N * N_{hid} + N$.

The authors claimed that this method outperformed significantly other state-of-the-art methods.

2.3.6. Stability of a feature selection method

Stability refers to the ability of a feature selection method to select the same subset of features among various subset of the training set examples. It is of course an important element to assert the relevance of a feature selection method.

[Meinshausen und Buhlmann 2010] showed a general procedure for resampling subsets of the original data and apply feature selection to them.

2.3.6.1. Jaccard The jaccard index is the rapport between the intersect and the union of two sets. We can use it as a measurement of the stability of our feature selection method by running it multiples time in order to produce multiple subsets of features and then taking the mean jaccard index of all combinations of these subsets.

$$Jaccard(U, V) = \frac{|U \cap V|}{|U \cup V|} \tag{28}$$

Where U and V represent a set.

2.3.6.2. A Better index A similar approach have been proposed by [Kuncheva] to achieve stability assessment.

$$K(\{S1, ..., S_k\}) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{|S_i \cap S_j| - \frac{s^2}{N}}{s - \frac{s^2}{N}} \tag{29}$$

Where k is the number of subsets we have to compare and S_i and S_j are two signatures built from different subsets of the training samples.

Of course stability alone can not be used to assess a feature selection method as a constant subset picked by a useless method will be indeed very stable. [Helleputte und Dupont 2009]

3. Methodology

3.1. Models in Keras

We implemented in keras the class PairwiseFS that take a base model (that we will call the baseline model) of DNN and add a pairwise layer between inputs and the first hidden layer. This layer have multiple arguments, l1 parameter for penalizing the weights of this layer with an l1 norm, l2 parameter for penalizing the weights with an l2 norm, l3 is a parameter to promote variance of the weights, the activation parameter allows to choose the activation function and the parameter weights_init let us choose between a constant initialisation of the weights or a randomized initialisation drawn from the normal distribution.

With that class we simulated the model inspired from [Li u. a.] and the model inspired from [Borisov u. a.] by setting the sigmoid activation function on the PairwiseFS model. Our implementation can be found in appendice.

We implemented another model inspired from [Huang u. a. 2020] that we called NeuralFS. We also implemented for that class a get_support method that followed the procedure they described in the paper to achieve feature ranking.

These models takes a base model as parameter. We also created a model identical to the base model with an additional dense layer after the input with the same number of neurons that the number of inputs neurons on which we applied l_1 penalties. We named this model: "regularized model".

3.2. Dataset used

3.2.1. Synthetic dataset

In order to asses the performance of our models we created a synthetic dataset in which the targets have a nonlinear relationship with the features. The features 0 to 4 are the ones used in order to compute the targets, the 5 to 9 are irrelevant and the features 10 to 13 are redundant with the first four.

The function used to create the targets is:

$$y = 10 * sin(\pi x_0 x_1) - 20 * (x_2 - 9.5)^2 + 5x_4 + \epsilon \tag{30}$$

Where five relevant features x_0 to x_4 are drawn from a uniform distribution and ϵ denotes the standard normal deviation $N(0, 1)$. The redundancy of other features is also created by adding a sample drawn from a uniform distribution from 0 to 1. Our dataset contains 5000 samples. The shape of it is then 5000 rows x 14 columns.

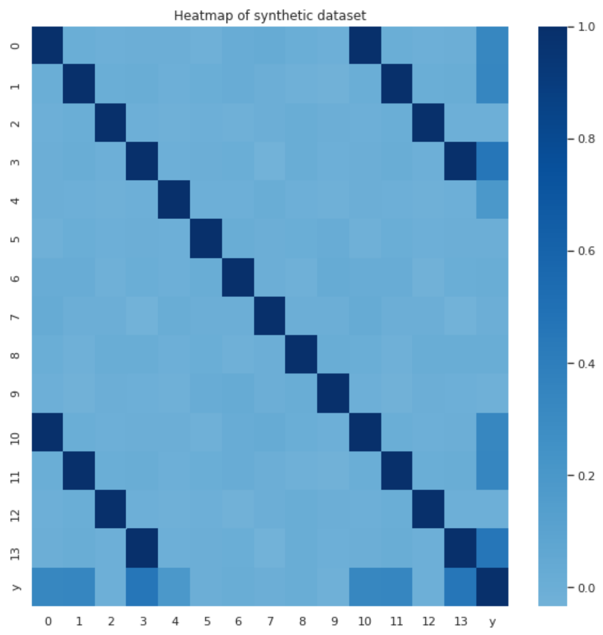


Figure (9) Heatmap (correlation matrix) of synthetic dataset

3.2.2. Arcene

The arcene dataset indicate the abundance of proteins in human sera having a given mass value. Based on those features, the models should predict the targets which discriminate healthy and cancer patient. The order of the features and patterns were randomized. More information can be found at: https://archive.ics.uci.edu/ml/datasets/Arcene

The shape of the dataset is 198 rows x 10000 columns.

3.2.3. Breast

The features of this dataset are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image such as the perimeter of the cell, etc..

The shape of the dataset is 568 rows x 30 columns. More information can be found at: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

3.2.4. Gastro

The features of this dataset are extracted from video data portraying gastrointestinal lesions. There are features vectors for 76 lesions, and there are 3 types of lesion: hyperplasic, adenoma and serrated adenoma.The problem can be transformed in a binary classification problem by combining adenoma and serrated adenoma in the same class.

In this configuration hyperplasic lesions would belong to the class 'benign' and the other 'malignant'.

The shape of the dataset is 76 rows x 698 columns. More information can be found at: `https://archive.ics.uci.edu/ml/datasets/Gastrointestinal+Lesions+in+Regular+Colonoscopy`

3.3. Choose of meta-parameter with Cross-Validation

We first split each dataset into a train set and a test set. The test-set is 1/3 of the size of the entire dataset.

We then searched the best metaparameters for the baseline model for each dataset. We computed a cross validation on the train set with 3 folds over these particular set of arguments in order to compute the best models on which we will make our experiments later.

The models are composed of a DNN in a pipeline after a scaling process made with the StandardScaler from the library sklearn.

The parameter grid in which we search for the best models with respect to its BCR score is the following:

Baseline	Value
Neurons	[10,50,100] for each layers (3)
11	[0.001,0.01,0.2,0.5]
l2	[0.001,0.01,0.25,0.4]
Regularized	Value
l1	[0.001,0.01,0.2,0.5]
Pairwise	Value
l1	[0.05,0.2,0.5]
12	[0.00,0.05,0.25,0.4]
l3	[0.00,0.005,0.1,0.5]
init	['glorout','const']
CancelOut	Value
l1	[0.05,0.2,0.5]
12	[0.00,0.05,0.25,0.4]
l3	[0.3,0.6,1]
init	['glorout','const']
NeuralFS	Value
12	[0.05,0.1,0.5]
threshold	[0.005,0.01,0.05,0.1,0.5]
LassoNet	Value
M	[0.05, 0.02, 0.5, 1, 5, 10, 20, 50, 100]

Table (1) Grid-search of meta-parameter

On table 1, init refer to the initialisation of the weights.

3.4. Feature ranking

To obtain feature ranking on the baseline DNN and the regularized DNN we used the technique described on the previous section: the activation potential analysis.

See Apendix C. for the function implementation

Obtaining the feature ranking for the pairwise (and cancelOut) model is straightforward. The weight of the pair layer is the feature ranking.

For the NeuralFS model we use the method we described in the previous section and we implemented it under the method get_support of the implemented class.

3.4.1. Feature ranking with filters

The feature ranking is computed by the normalized addition of the feature ranking obtained with the mutual information filter from the library sklearn and the feature ranking obtained with the filter RelieF from the the library skrebate.

3.5. Pseudo-code methodology test function

With the meta parameter obtained by searching the ones that lead to the best BCR score on the training set, we then test our models on the test set.

If embedded feature selection can be seen as an immediate way to process feature selection, it can also be viewed as a way to obtain a very accurate feature ranking that allows us to interpret the model but also to perform a posteriori the feature selection directly on the dataset based on the feature ranking. Proceeding in this way, often leads to even better results and we tested that assumption by reducing the test-set with different subsets of features from it, following the feature rankings. The part of the feature ranking evaluated in percentage represent the number of feature that we keep when adding the score associated with each feature until the sum reach a particular percentage of the total sum of all feature scores.

We also reduce the test-set by selecting half of the number of features by taking the n most important (where n is the total number of features divided by 2). We will refer to this particular run in the result section as "Mid".

In order to analyse stability we use the Jaccard index when we reduce the test-set with the percentage method (as the number of features kept between each folds may be different) and **the Kunsheva index** when fix the number of feature kept for the "Mid" run.

Algorithm 1: Evaluate a embedded feature selection method

```
Result: List of benchmarks
evaluate model on the test set;
if model is (Baseline or Regularized) then
    get feature ranking with activation potential analaysis;
else
    get feature ranking with the weights of the first layer (or via an
      implemented method) ;
end
applied feature selection with respect to the feature ranking of the FS model;
evaluate the baseline model on the reduced test set;
Evaluate stability of the feature ranking with a cross-validation of 4 folds on
  the test set (by compare the subset kept for each fold) ;
```

We applied this algorithm multiple times with different portions of the feature ranking.

As training a model implies a lot of stochastic parameters, we used the multiple runs of the test algorithm to save the different scores obtained with each model evaluated on the full test-set. We then compute the mean of these scores in order to have a more robust benchmark for each model. We then computed the mean score of 5 runs.

The notebooks that we used to make our experiments can be found at https://github.com/rayouaz/Deep-feature-selection and their seed are fixed in order to make the results reproducible.

3.6. Scaling consideration

As the arcene dataset contains a lot of features, computing the potential activation analysis of the regularized (which have a dense layer with as many neurons as inputs) was out of reach. The same problem comes for the neuralFS model but a simple solution to that is to reduce the number of neurons of the intermediate layer of the first part of the network. We reduced then the number of neurons for this layer by 3 (we divide it by 3) for the arcene dataset and choose to not compute anything for the regularized model for this dataset.

3.7. Hardware details

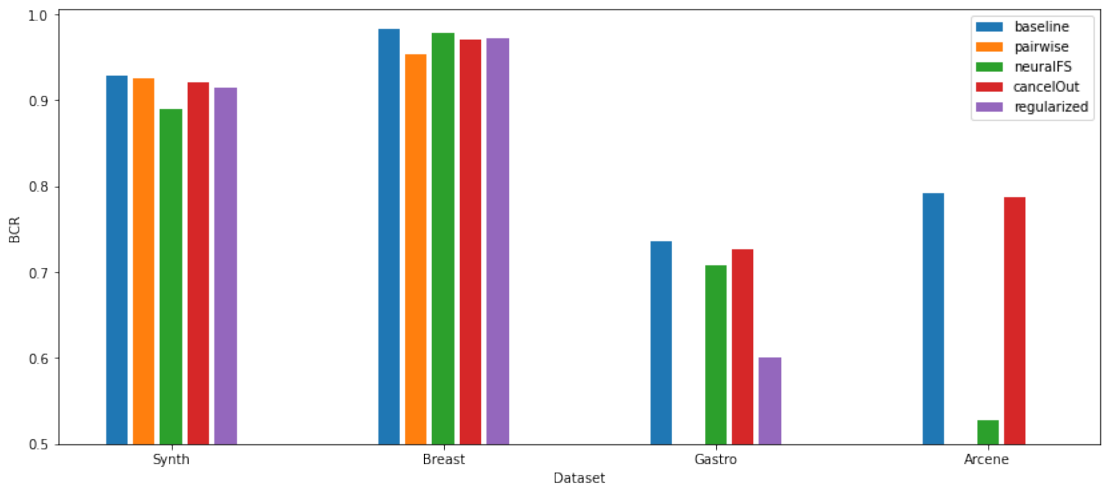
All experiment have been done using a Intel(R) Xeon(R) CPU @ 2.30GHz with 32GB of memory available. There was no use of GPU for computation.

4. Results

4.0.1. Scores on full dataset computed after 5 runs and confidence intervals

BCR Model				
Model	Synth	Breast	Gastro	Arcene
Baseline	0.928 +- 0.012	0.982 +- 0.019	0.736 +- 0.169	0.792 +- 0.098
Pairwise	0.925 +- 0.013	0.953 +- 0.030	0.5 +- 0.192	0.5 +- 0.121
NeuralFS	0.890 +- 0.015	0.979 +- 0.020	0.707 +- 0.175	0.528 +- 0.120
CancelOut	0.920 +- 0.013	0.971 +- 0.039	0.727 +- 0.171	0.788 +- 0.099
Regularized	0.914 +- 0.014	0.972 +- 0.024	0.6 +- 0.188	

p-value for confidence interval is 95%.



4.0.2. Scores with dataset reduced

4.0.2.1. From 80% of the feature ranking

BCR Model				
Ranking From	Synth	Breast	Gastro	Arcene
pairwise	0.927	0.952	0.727	0.806
cancelOut	0.872	0.981	0.727	0.75
lassonet	0.871	0.944	0.727	0.819
baseline	0.93	0.95	0.773	0.749
neuralFS	0.924	0.983	0.818	0.752
filter	0.852	0.959	0.636	0.788
regularized	0.924	0.961	0.727	0.0

4.0.2.2. From 60% of the feature ranking

BCR Model				
Ranking From	Synth	Breast	Gastro	Arcene
pairwise	0.876	0.964	0.727	0.788
cancelOut	0.867	0.876	0.727	0.819
lassonet	0.85	0.933	0.773	0.713
baseline	0.878	0.979	0.727	0.779
neuralFS	0.879	0.979	0.545	0.725
filter	0.852	0.968	0.773	0.748
regularized	0.786	0.946	0.727	0.0

4.0.2.3. Divide the number of feature by two

BCR Model				
Ranking From	Synth	Breast	Gastro	Arcene
pairwise	0.926	0.968	0.727	0.723
cancelOut	0.871	0.892	0.727	0.827
lassonet	0.927	0.971	0.727	0.806
baseline	0.876	0.976	0.727	0.779
neuralFS	0.874	0.968	0.864	0.782
filter	0.872	0.964	0.727	0.792
regularized	0.869	0.945	0.727	0.0

4.0.3. Stability of feature selection

4.0.3.1. With 80% of the feature ranking

BCR Model				
Model	Synth	Breast	Gastro	Arcene
pairwise	0.427	0.717	0.767	0.319
cancelOut	0.897	0.656	0.669	0.68
lassonet	1.0	1.0	0.542	0.553
baseline	0.852	0.815	0.766	0.764
neuralFS	0.463	0.608	0.289	0.333
filter	1.0	1.0	1.0	1.0
regularized	0.444	0.616	0.934	0.0

4.0.3.2. From 60% of the feature ranking

BCR Model				
Model	Synth	Breast	Gastro	Arcene
pairwise	0.2	0.619	0.753	0.174
cancelOut	0.745	0.44	0.43	0.429
lassonet	1.0	1.0	0.327	0.375
baseline	0.714	0.833	0.677	0.644
neuralFS	0.2	0.487	0.284	0.333
filter	1.0	1.0	1.0	1.0
regularized	0.484	0.821	0.855	0.0

4.0.3.3. Divide the number of feature by two

BCR Model				
Model	Synth	Breast	Gastro	Arcene
pairwise	0.429	0.511	1.0	-0.004
cancelOut	0.619	-0.022	0.009	-0.001
lassonet	0.81	0.689	0.769	0.589
baseline	0.81	0.778	0.916	0.749
neuralFS	0.524	0.644	0.566	0.999
filter	1.0	1.0	1.0	1.0
regularized	0.429	0.689	0.983	0.0

In the following sections, multiple graphs are representing two kinds of lines: a dotted one and a normal one. The normal one represent the score obtained by the full model without reduction and its confidence interval (p-value: 95%) and the dotted one represent the score obtained by the baseline model after reducing the dataset with respect to the ranking of the full model. The percentage in some titles represent the percentage of the feature ranking (with respect to the feature importance order) provided by the model. The parameters in the titles are the ones which had different values tested or the ones fixed to zero if precised. Without any counter-indication all models have been tested on 100 epochs (except the LassoNet one which have its own procedure).

4.0.4. Pairwise

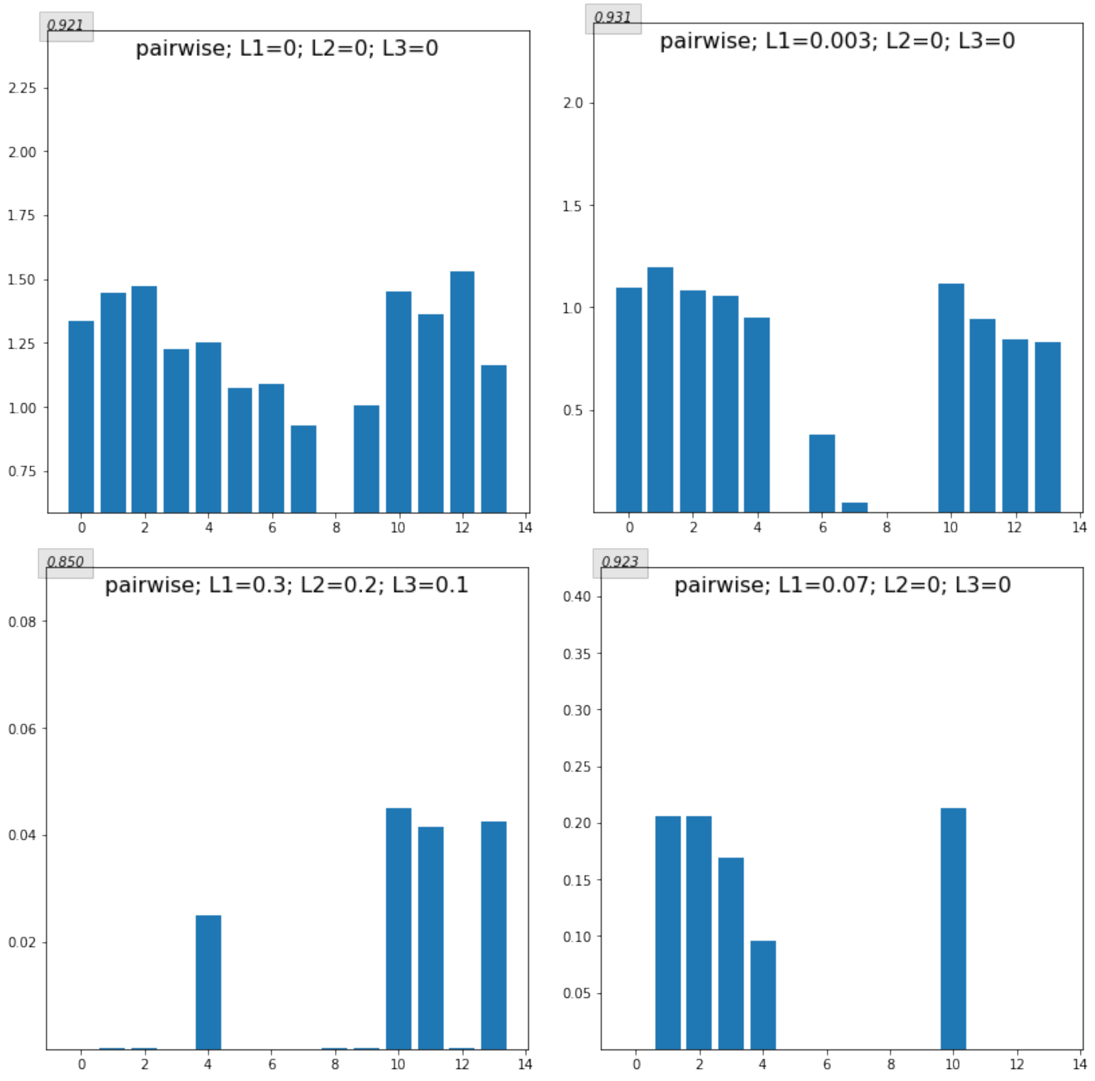


Figure (10) Feature rankings with different parameters for the pairwise model

The rankings confirms the intuition that the magnitude of the L1 penalty mainly controls the amount of features that will be selected.

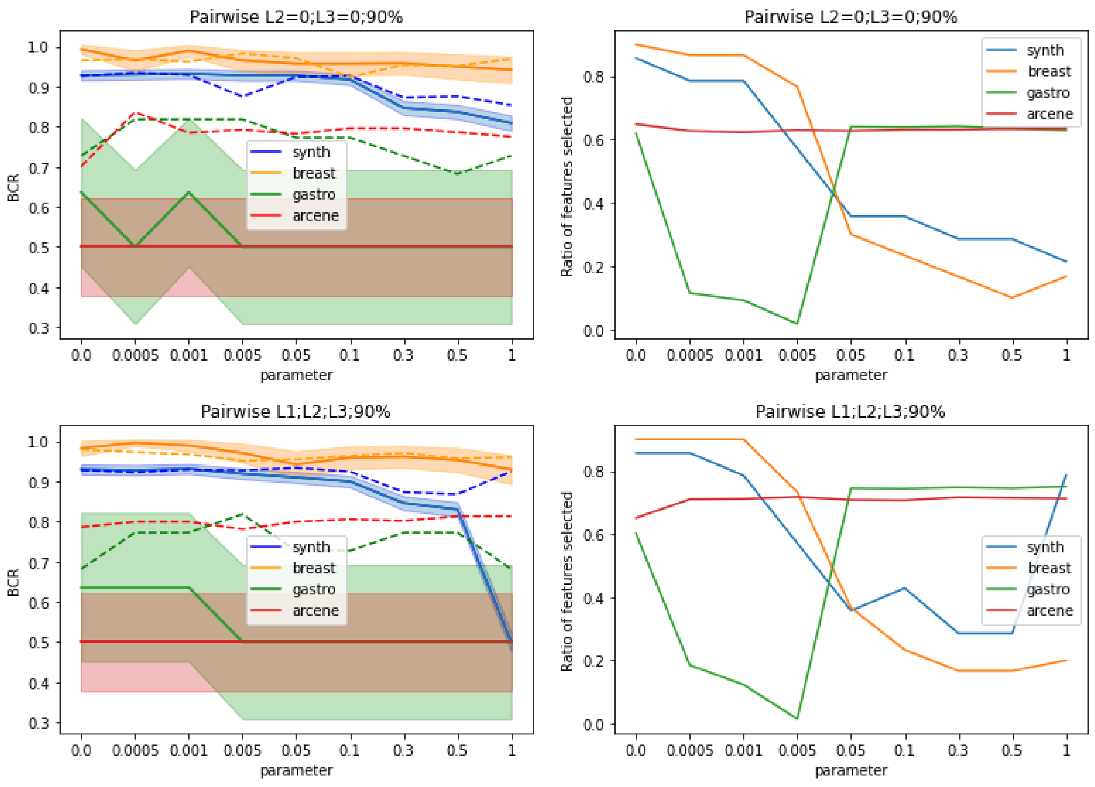


Figure (11) Tuning of multiple parameters for the pairwise model.

While varying the L1 penalty while the two others main parameters are set to 0 we observed that increasing the penalty did not lead every time to reduce the number of feature selected. When the penalty became too large for some dataset, the number of features selected rapidly increased. As the runs which suffers from this dysfunction are the ones with the higher number of dimension when penalties are numerous, these failures can be explained because the computing of the gradient descent becomes impossible as we can observe on the model scores.

A user of this model must mark attention to search the right magnitude of the parameters and start by testing very small values. This augmentation of the number of feature selected is translated to a dramatic decrease of the model accuracy for the pairwise model. On the other hand, the baseline model with the reduced dataset based on the pairwise ranking was not affected by this decrease because the ranking made then no distinction between features and selected most of them.

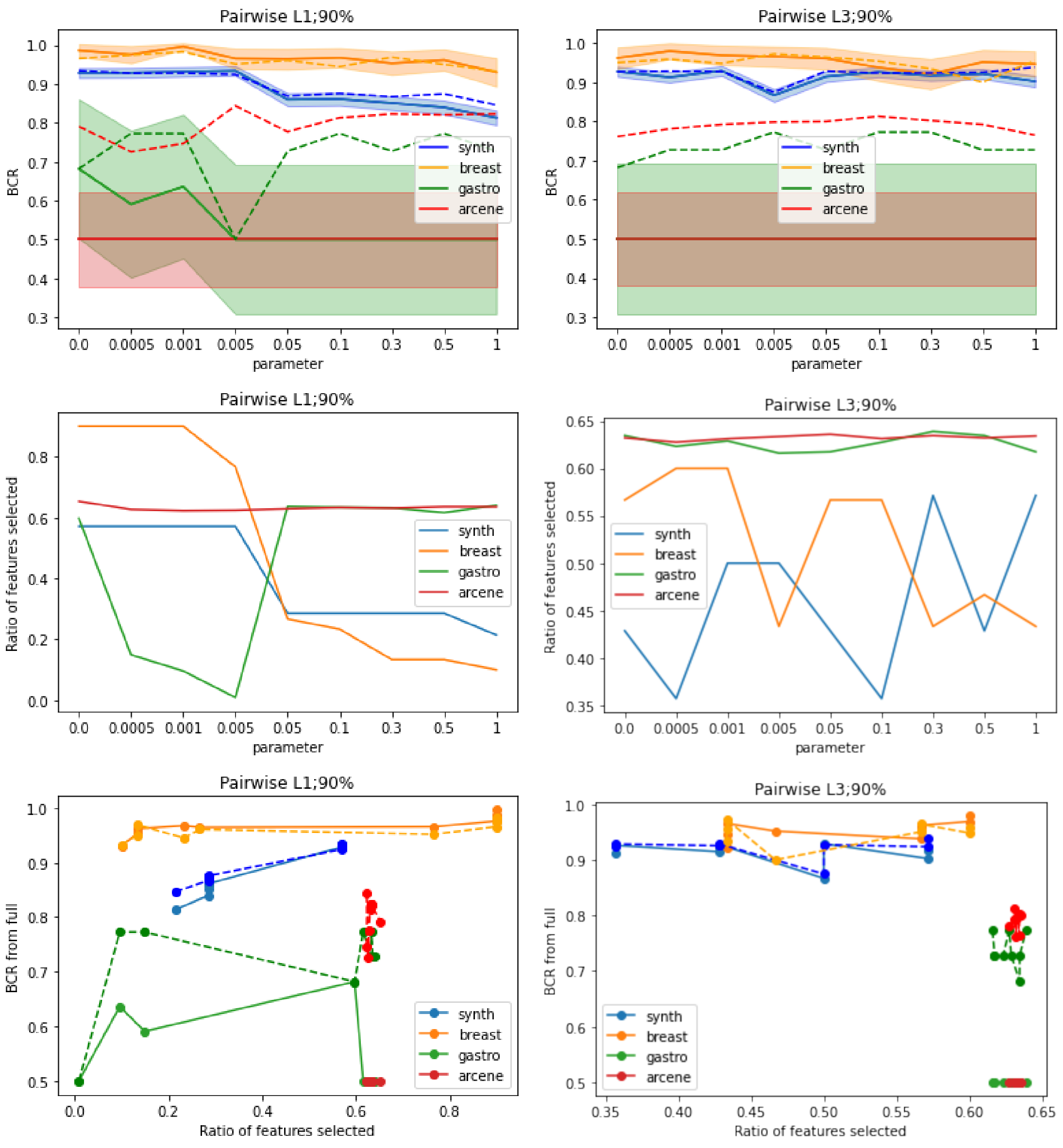


Figure (12) Tuning of l1 and l3 penalties on pairwise model.

Tuning the l1 parameter while the l2 and l3 penalties are not equal to zero provided the same results as the previous experience. However we did not observe a linear correlation between the penalty that promotes variance in the pairwise layer and the balanced accuracy even when training the model with less epochs. Yet we observed that a fine tuning of this parameter may increase the performance on the tested dataset.

4.0.5. Regularized

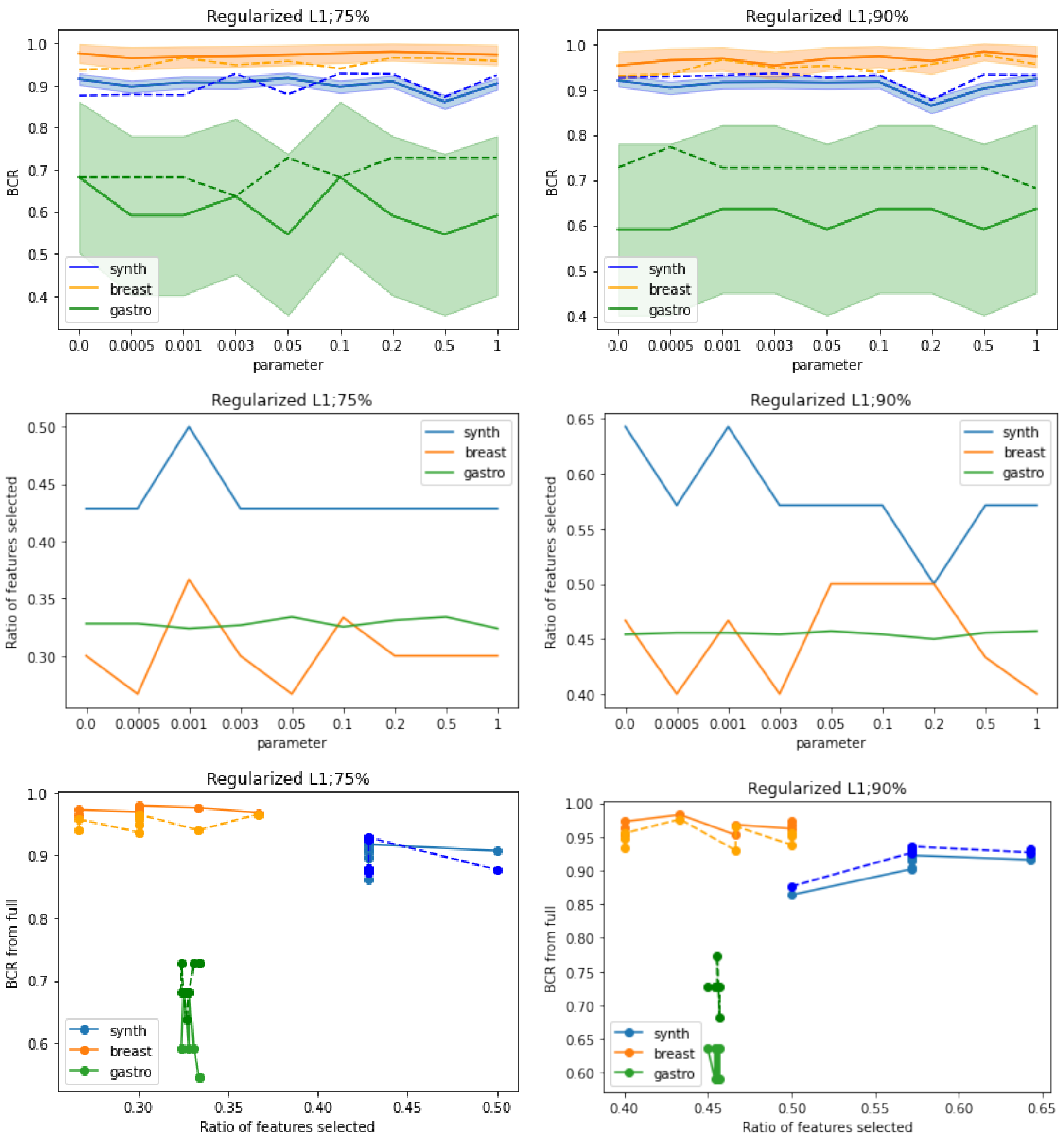


Figure (13) Tuning of l1 on regularized model.

The tuning of the regularized model did not provided any repeatable sign of influence on the feature ranking evaluated by the action potential analysis with 75% or 90% of the feature ranking.

4.0.6. Pairwise against regularized

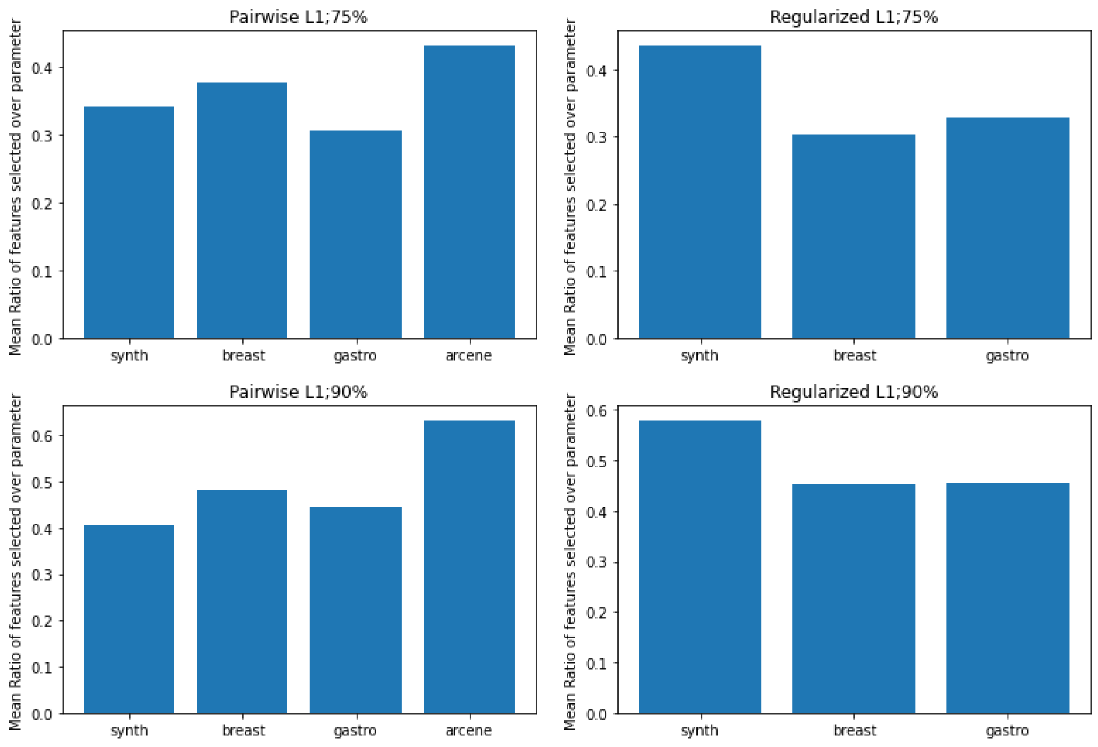


Figure (14) Ratio of feature selected when trying different L1 penalties

The authors of DeepFS claimed that the pairwise structure can lead to the same accuracy as the LASSO model (similar to our regularized model) while selecting less features. As we can observer on figure (15) our experiments confirms that affirmation. For instance, approximately 40% of features was selected for the synthetic dataset (in order to have 90% of the sum of all feature score). The regularized model selected at minimum 55% of the features while the performance remained as good.

4.0.7. CancelOut

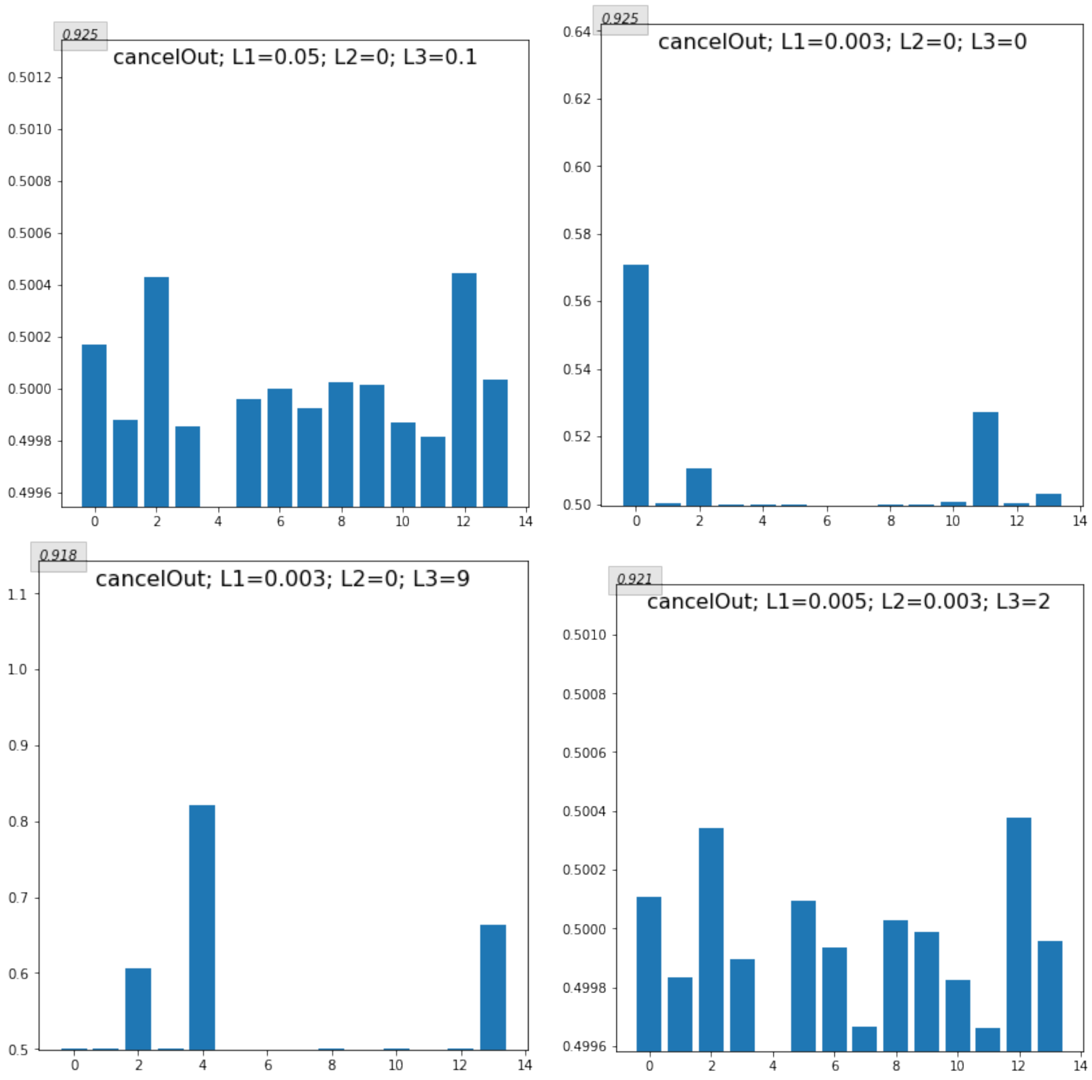


Figure (15) Feature rankings for the cancelOut model.

We can observe on figure (16) that as excepted the cancelOut model, unlike the simple one-to-one pairwise model, did not push the features weights to zero at the output of the activation function of the hidden layer. The model provides a ranking without erasing the signal of the potentially irrelevant features. At the output of the hidden layer if a signal is less than 0.5 the model had pay a cost to enhance the signal of this particular feature. Promoting variance helped to counter-balance this effect and discriminate relevant and irrelevant features until the magnitude of l3 penalty became too large which tend to vanish this effect.

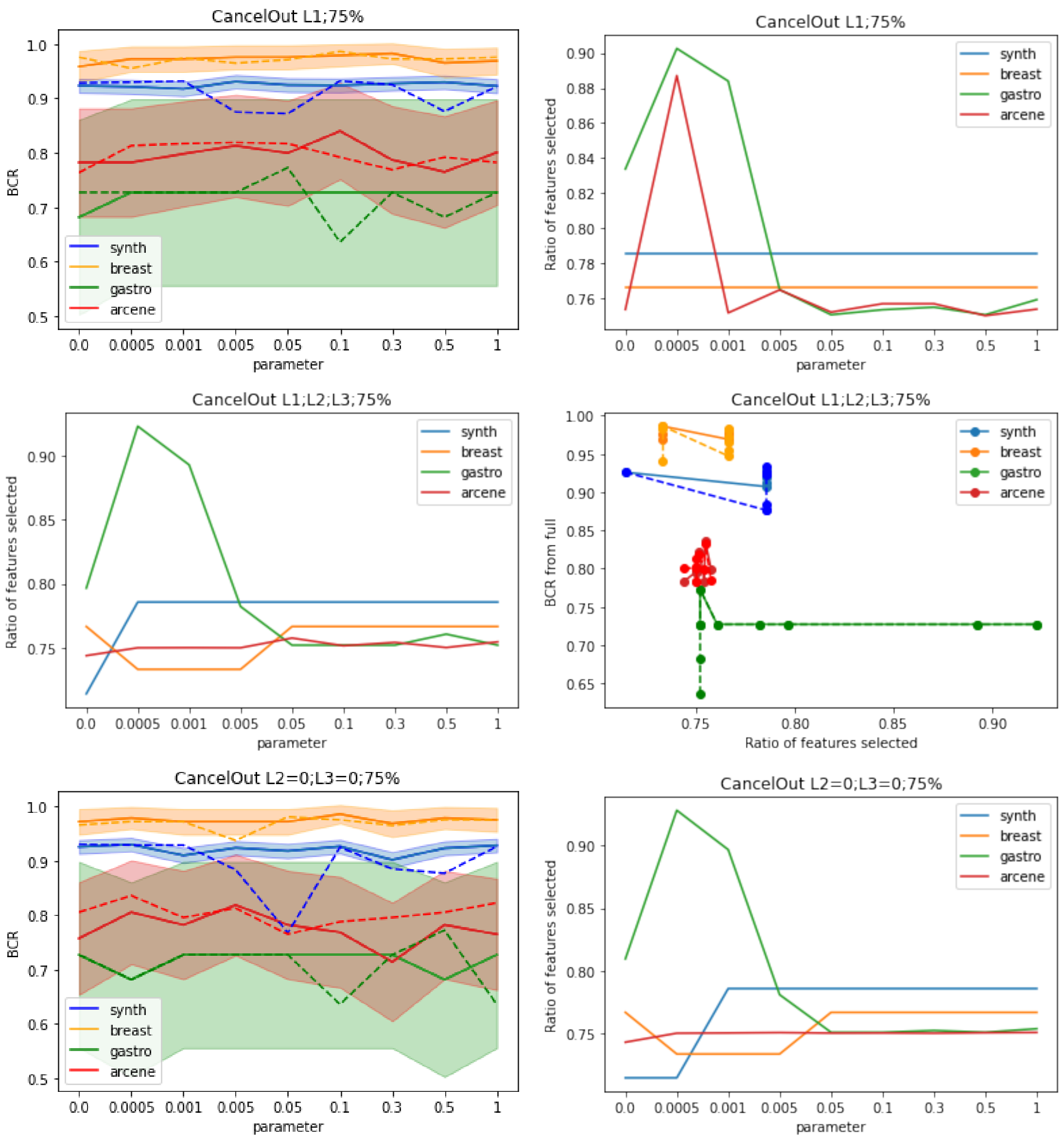


Figure (16) Tuning of CancelOut model.

A null l1 penalty tends to select less features than a small penalty in the cancelOut model. It can be explained by the fact that weights have no restrictions to be negative with no penalties and thus introduce more sparsity due to the sigmoid function. A small penalty tend to select more features as it is then more costly to cancel the features. However increasing this parameter allowed to select smaller amount of features. The BCR score did not change in a significant fashion through the iterations with different tunings.

We observed that the model tend to select the same ratio of feature as the percentage that we choose. It was excepted as the different feature scores are really close.

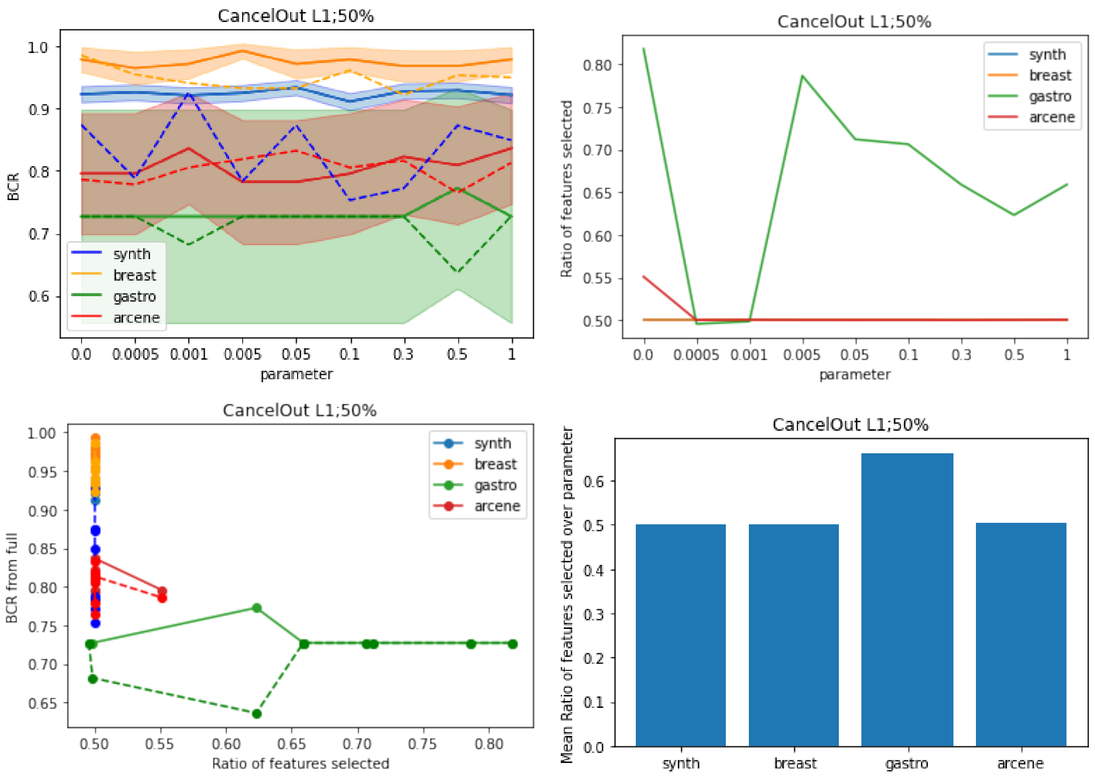
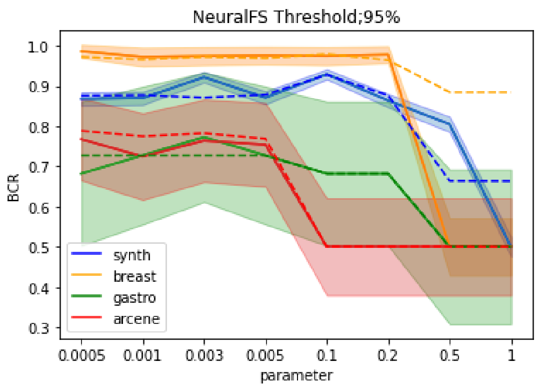
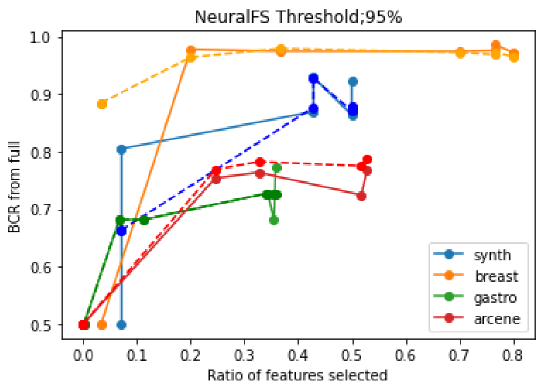


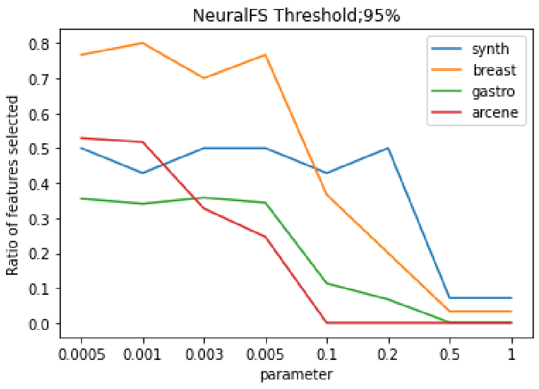
Figure (17) Tuning of CancelOut model.

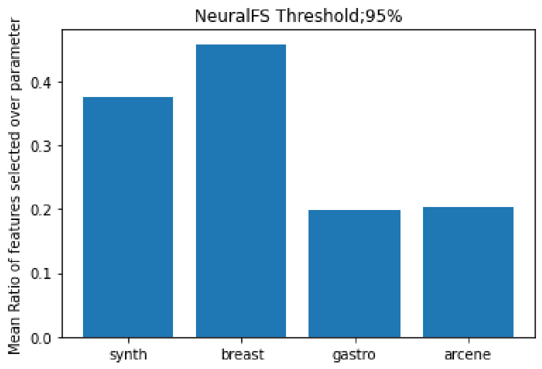
When we reduced the dataset with respect to 50% of the feature ranking the selection started to be less stable for the gastro dataset. This can be explained by the fact that most of the features were canceled. It can be used as a straight indicator of the amount of irrelevant feature in a dataset.

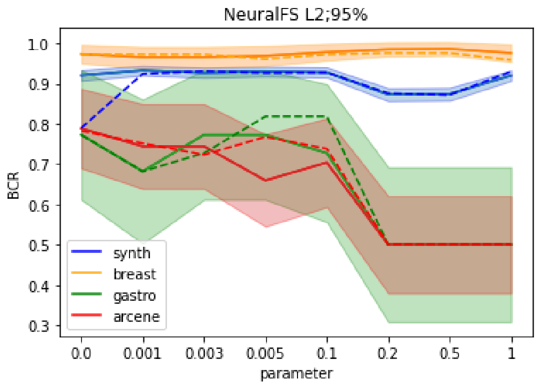
4.0.8. NeuralFS

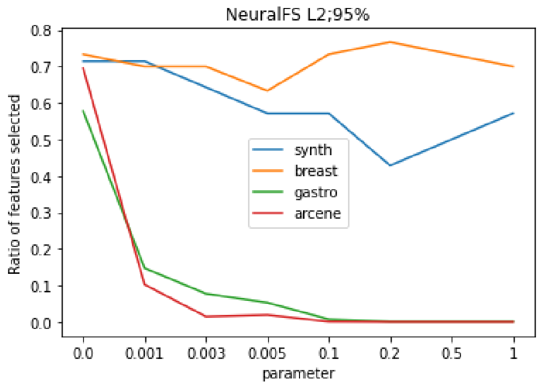

NeuralFS Threshold;95%
BCR
parameter
0.0005 0.001 0.003 0.005 0.1 0.2 0.5 1
synth
breast
gastro
arcene

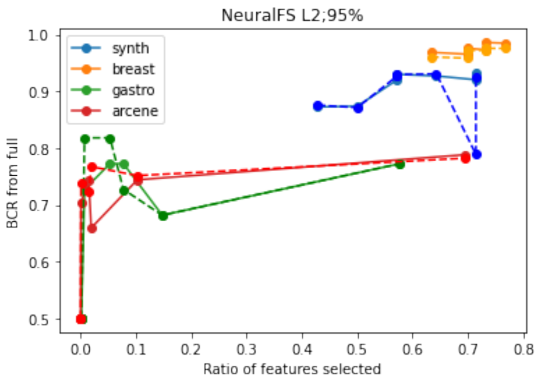

NeuralFS Threshold;95%
BCR from full
Ratio of features selected
synth
breast
gastro
arcene

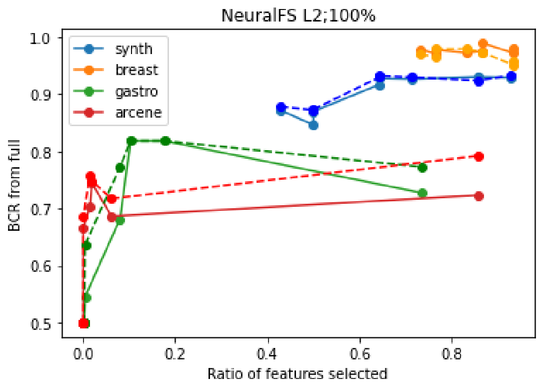

NeuralFS Threshold;95%
Ratio of features selected
parameter
0.0005 0.001 0.003 0.005 0.1 0.2 0.5 1
synth
breast
gastro
arcene


NeuralFS Threshold;95%
Mean Ratio of features selected over parameter
synth
breast
gastro
arcene


NeuralFS L2;95%
BCR
parameter
0.0 0.001 0.003 0.005 0.1 0.2 0.5 1
synth
breast
gastro
arcene


NeuralFS L2;95%
Ratio of features selected
parameter
0.0 0.001 0.003 0.005 0.1 0.2 0.5 1
synth
breast
gastro
arcene


NeuralFS L2;95%
BCR from full
Ratio of features selected
synth
breast
gastro
arcene


NeuralFS L2;100%
BCR from full
Ratio of features selected
synth
breast
gastro
arcene

The NeuralFS model, according to its authors should provide a very stable ranking and an easy-to-tune interface to choose the quantity of feature selected with the theshold parameter. We can observe that as expected, the threshold parameter was very useful to select the right proportion of feature. A strong correlation can be establish between the threshold parameter and the ratio of feature selected. On the other hand, we need to pay attention to avoid to choose a too large value for this parameter otherwise it will not select any feature and the model becomes useless. The performance of the model obviously declined when less features was selected. In opposition with previous models the numbers of feature selected can easily change following this parameters and affect the performance. For that reasons the authors of the papers associated to this structure proposed to create an adaptive threshold. This ability to control the quantity of feature selected is very desirable to avoid the intermediate phase of reducing the dataset according a provided ranking, putting at disposition a ready-to-use model that can easily be incorporated in a pipeline algorithm. The structure of the model is thus designed to be equivalent to the choice of extracting features from the feature ranking and use it as the input of the baseline model. The strong similarity between the scores obtained by the model and the baseline with reduced training set confirms that the two approach provides similar results as expected.

4.0.9. LassoNet

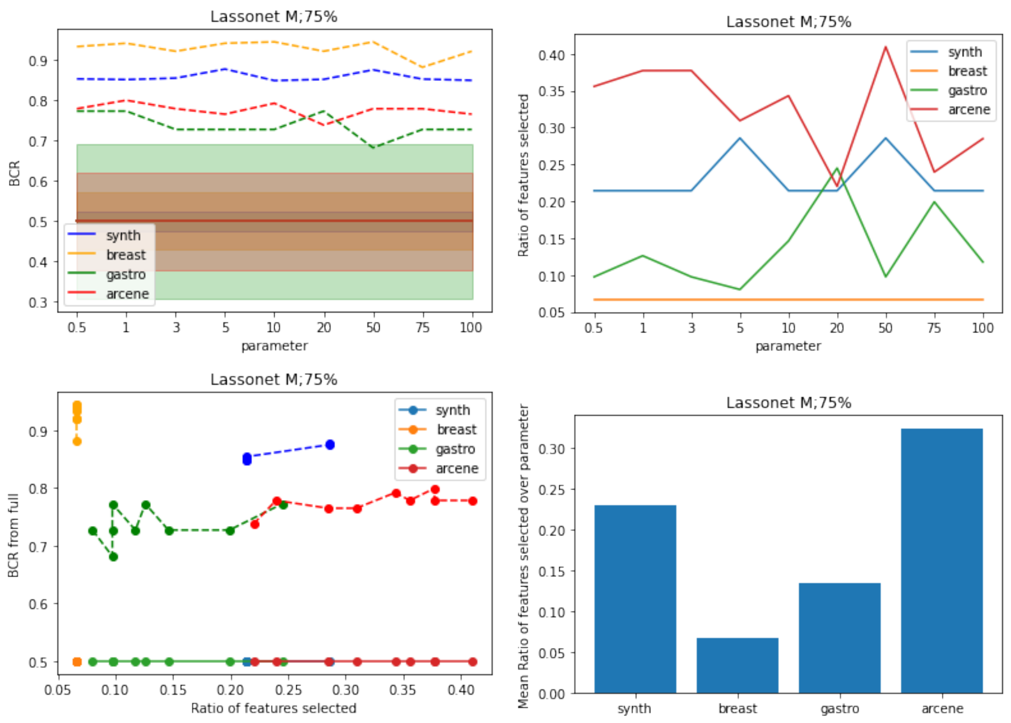


Figure (19) Tuning of LassoNet.

The model is made in order to provide the feature ranking at the end of the regularization path as we explained before. Still the procedure did not provide an accurate fitted model. This is the reason why the score obtained by the model at the end of the procedure failed to classify the datasets. The M parameter can be tuned to optimise the balanced accuracy but we did not find any correlation between the score and the magnitude of the M parameter.

4.0.10. Sparsity

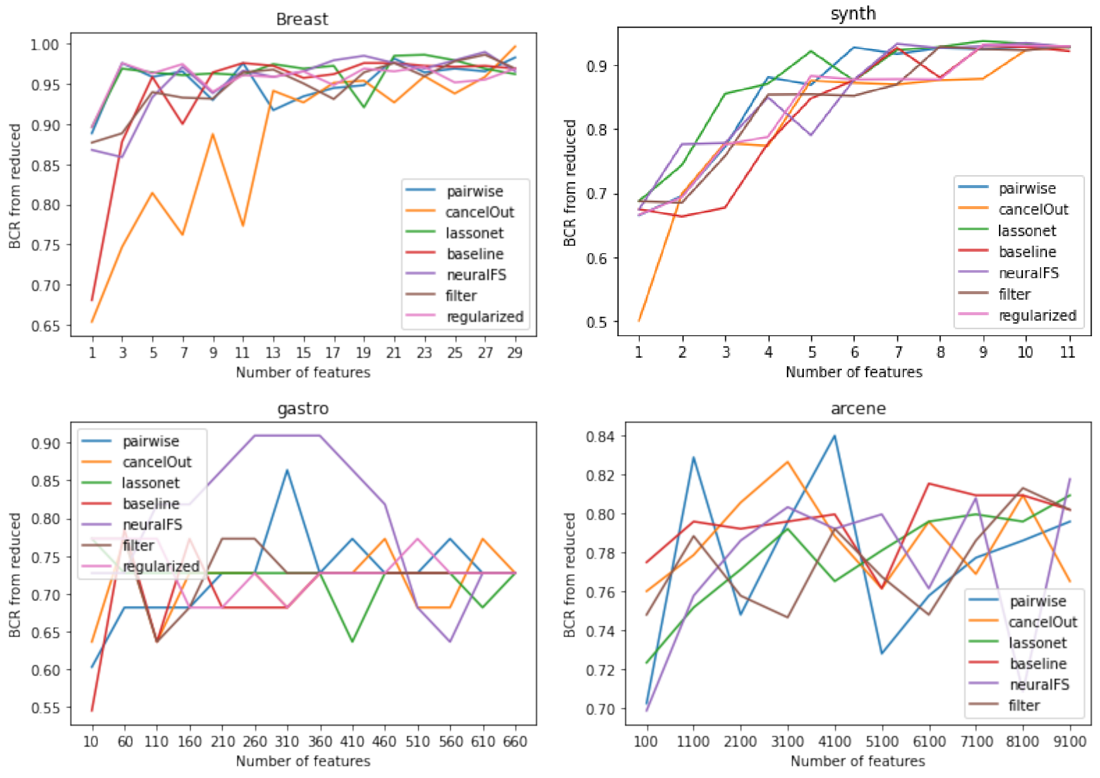


Figure (20) BCR when reducing the dataset with respect to the ranking privoded by a model

On the breast dataset the results of different models were similar when we used more than 50% of the features. When we used less features, especially when we used only a few of them, the baseline and the regularized model have lost accuracy as well as the cancelOut model. The lassonet, pairwise and filter approach did better, especially the LassoNet model which kept the accuracy at its maximum with only three features.

For the synthetic dataset the best model was distinctly the pairwise and the lassonet model. They reached the second best balanced accuracy with 5 features which is the number of relevant features in this case. The worst performance comes from the cancelOut model.

In the gastro dataset, the NeuralFS model outperformed the others and found clear benefit of the embedded mechanism of feature selection on the accuracy. The baseline approach failed to correctly classify this dataset with only 10 features while others still maintained good accuracy.

The cancelOut model and the activation potential analysis on the baseline model provided very robust results and was better to select fewer features than other models on the arcene dataset.

4.0.11. Performance

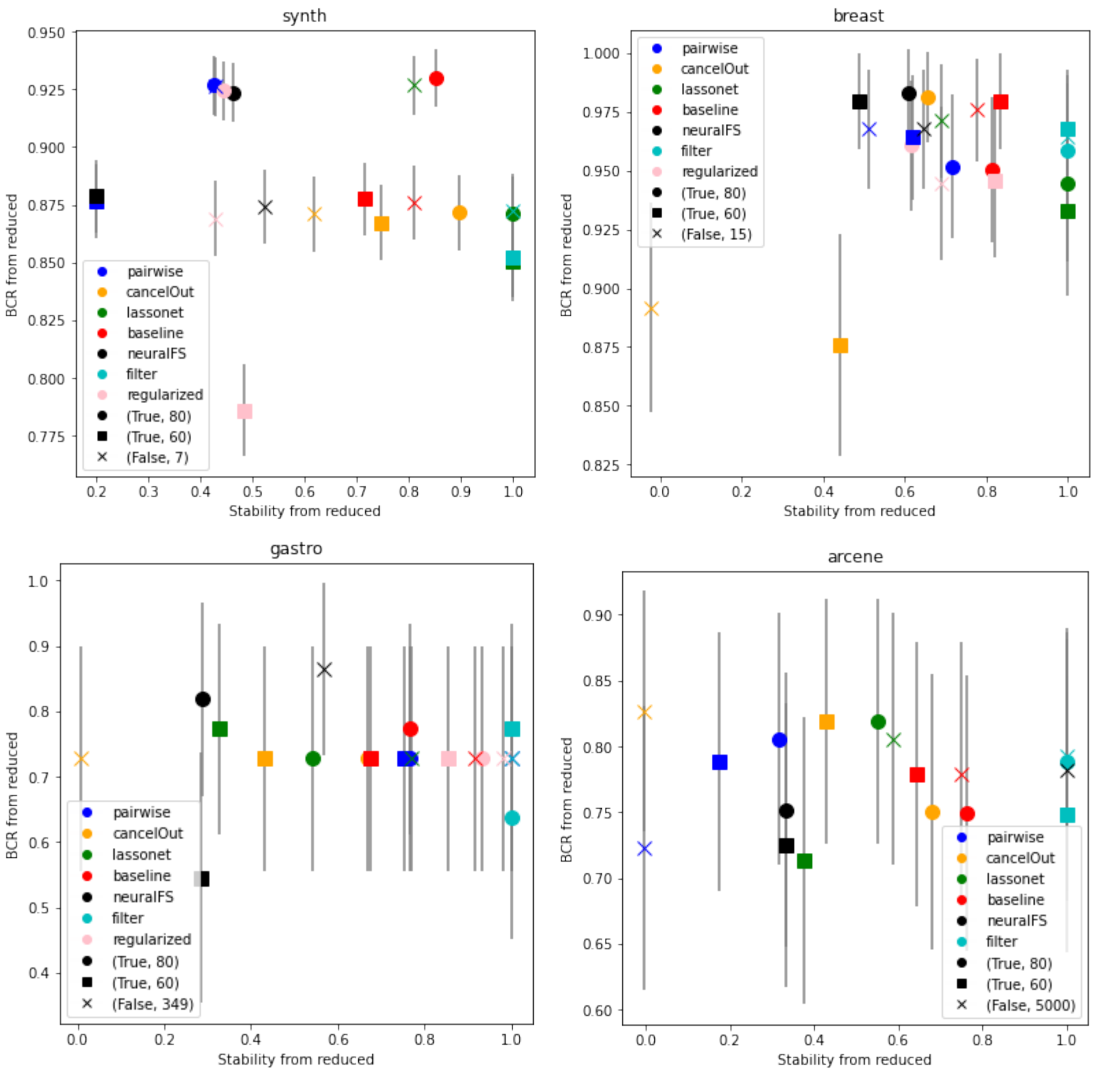


Figure (21) BCR on stability when reducing w.r to 80%, 60% and half of features of the feature ranking (boolean in legend indicate if the value is a percentage)

4.0.12. Feature ranking on synthetic dataset computed over 5 runs

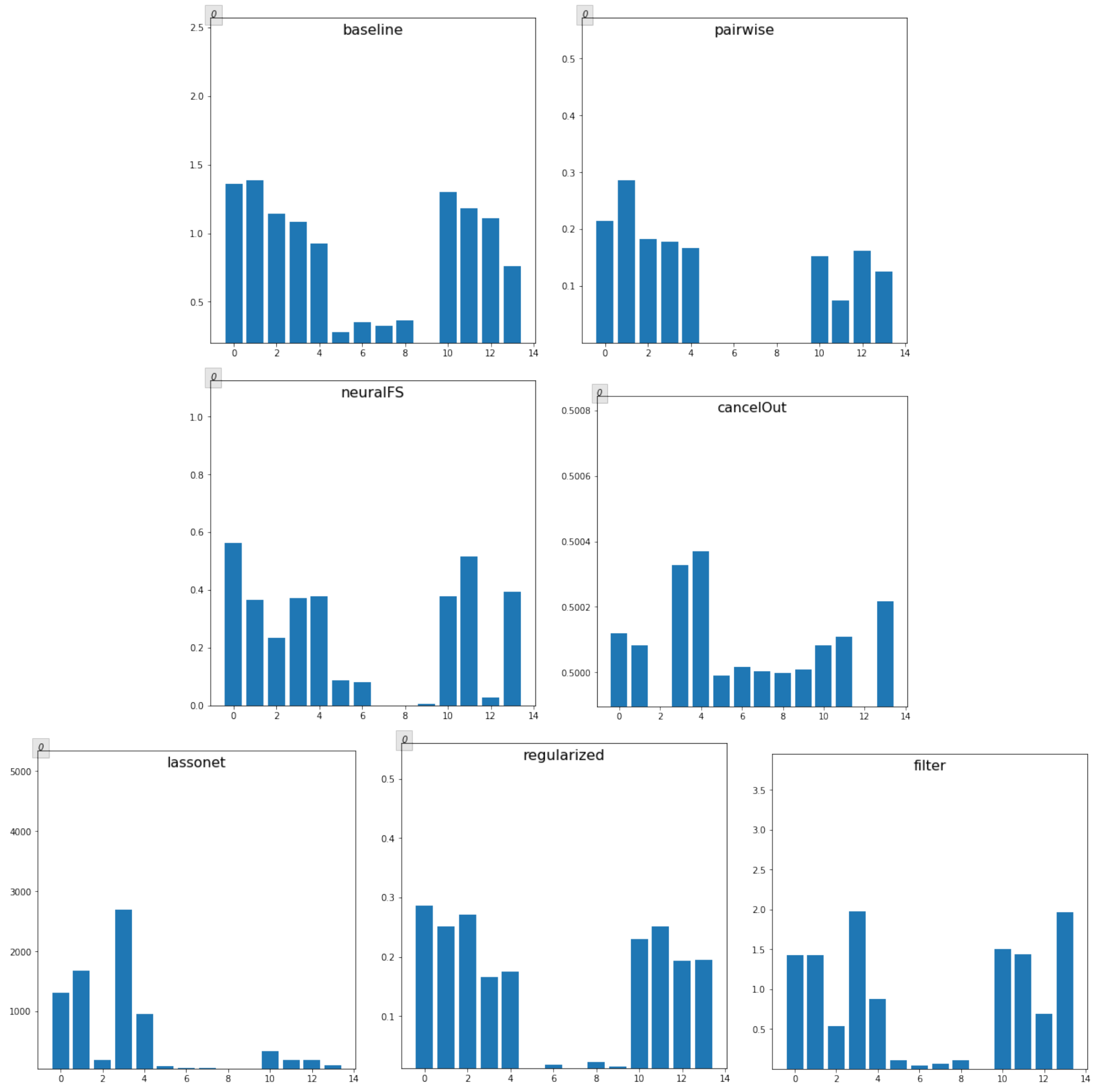


Figure (22) Feature rankings

5. Discussion

5.1. Benefits

We can observe than most of the time the special structure that we used for feature selection (Pairwise,NeuralFS,regularized,..) were outperformed by the baseline they were built from.

When it comes to train a reduced model provided by the feature ranking of an embedded selection, multiple model have competed the baseline run with all features. The pairwise and the LassoNet model provided a ranking for which the scores was as good as the baseline run with only half of the feature on the synthetic dataset. Finally the use of the ranking provided by NeuralFS outperformed baseline (by 13%) with only half of the features on the gastro dataset. As well as the cancelOut model which also outperformed the baseline run by 3% on the arcene dataset with only half of the features.

5.2. Feature ranking

All techniques used to achieve feature selection has selected accurately the useful features on the synthetic dataset but the rankings are not perfect. As we can see on figure (22), the only ranking that has almost eliminate all irrelevant features is the pairwise ranking but in figure (21) we can observe that the stability of the selection is generally not good. Consequently, we could make the hypothesis that such a models is efficient to identify irrelevant features but less efficient to identity the most important ones. The pairwise model fails to classify the two dataset with high dimensionnality (Gastro and Arcene) but still produce a coherent ranking during the training. The cancelOut model provided mitigate results. The model was less efficient to actually proceed the feature selection on the dataset by being unstable and provided light results except on the arcene dataset which has the greatest dimensionnality.

The activation potential analysis applied on the baseline network provided good results, never the best and never the worst and the ranking was one of the most stable. The analysis applied on the regularized provided also a very stable ranking but is less accurate as we can on see figure (21) (the scores when reducing dataset from the regularized ranking are correct but never outperform other models).

The scores of the NeuralFS model are really high on the Breast and Gastro dataset but provided low stability on the different datasets.

The LassoNet procedure was very stable, provided good benchmarks and was better than other models to select less features on the breast and synth dataset.

Lastly, it is important to notice that the runs done when selecting subsets of features from the filter ranking was outperformed by most of the others models on the synthetic dataset especially when we selected 60% of dataset.

5.3. Conclusion

Feature selection can leads to better performances of classifier but the most notable benefits here was the ability to maintain the scores almost as good with and without dimensionality reduction. We showed that the models we tested on a reduced test-set with respect to the feature ranking provided by the implemented models can provide better results than the runs with the test-set reduced from the simple application of some filters.

We demonstrated that these very simple-to-implement embedded methods can easily be use to help experts to interpret the models by reducing the dimensionality. Furthermore, on high-dimensional dataset, the gain of performance is sometimes important.

These models opens promising perspectives to the incorporation of embedded feature selection into future state-of-the-art neural networks models. Future works could be to observe these mechanisms on convolutionnal networks after the pooling layers.

A. Pairwise class

```
class PairwiseFS(Model):
    def __init__(self, inputs, base_model, l1, l2=0.001, l3=0.0,
activationCancel='sigmoid', initializer=None):
        self.l1 = l1
        self.inputDim = inputs.shape[1]
        dense = []
        inputLayer = Input(shape=(self.inputDim,))

        ##Construct pairwise
        cancel = PairLayer(activation=activationCancel, lambda_1=l1,
lambda_2=l2, lambda_3=l3, init=initializer) (inputLayer)
        second = clone_model(base_model)
        super().__init__(inputs=inputLayer, outputs=second (cancel))


    def get_support(self):
        weights = self.get_weights()[0]
        return weights
```

Figure (23) PairwiseFS class with Keras

```
    def build(self, input_shape):
        self.w = self.add_weight(
            shape=(input_shape[-1],),
            initializer=self.init,
            trainable=True)

    def call(self, inputs):
        if self.cancelout_loss:
            shap = self.w.shape[0]
            self.add_loss( self.lambda_1 * tf.norm(self.w, ord=1)  + self
.lambda_2 * tf.norm(self.w, ord=2))
            if self.lambda_3 != 0.0:
                self.add_loss(-self.lambda_3 * tf.math.reduce_variance(
self.w*(1/shap)))

        if self.activation == None: return inputs * self.w
        else: return inputs * self.activation(self.w)
```

Figure (24) Build and call method of the class PairLayer instance of the Layer class of Keras

B. Activation potential analysis

```python
def activation_potential_analysis(X, weights):
    weights_first = weights[0]
    bias = weights[1]
    support = []
    p_ij = np.zeros((X.shape[1], len(weights_first[0])))
    for dim in range(len(weights_first)):
        for hidden in range(len(weights_first[0])):
            for trains in range(X.shape[0]):
                p_ij[dim][hidden] += abs(weights_first[dim][hidden]*X
[trains][dim])
            p_ij[dim][hidden] = p_ij[dim][hidden]/X.shape[0]

    sum_j = np.sum(p_ij, axis=1)
    return sum_j

```

Figure (25) Activation potential analysis implementation in Python

References

[Almuallim und Dietterich 1994] Almuallim, H. ; Dietterich, T. G.: Learning boolean concepts in the presence of many irrelevant features. In: *Artificial intelligence* 69 (1994), Nr. 1-2, S. 279–305

[Blum und Langley 1997] Blum, Avrim L. ; Langley, Pat: Selection of relevant features and examples in machine learning. In: *Artificial Intelligence* 97 (1997), Nr. 1, S. 245–271. – URL https://www.sciencedirect.com/science/article/pii/S0004370297000635. – Relevance. – ISSN 0004-3702

[Borisov u. a.] Borisov, V. ; Haug, J. ; Kasneci, G. (2019 S.: Cancelout: A layer for feature selection in deep neural networks.

[Bottou] Bottou, L (2010 F.: Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*

[Bugata und Drotar 2020] Bugata, P. ; Drotar, P.: *Feature Selection Based on Sparse Neural Network Layer with Normalizing Constraints. arXiv.* 2020. – preprint

[Degeest u. a. 2019] Degeest, Alexandra ; Verleysen, Michel ; Frénay, Benoît: *Comparison Between Filter Criteria for Feature Selection in Regression.* S. 59–71, 09 2019. – ISBN 978-3-030-30483-6

[Fefferman u. a. 2013] Fefferman, Charles ; Mitter, Sanjoy ; Narayanan, Hariharan: *Testing the Manifold Hypothesis.* 2013

[Glorot u. a. 2011] GLOROT, Xavier ; BORDES, Antoine ; BENGIO, Yoshua: Deep Sparse Rectifier Neural Networks. In: GORDON, Geoffrey (Hrsg.) ; DUNSON, David (Hrsg.) ; DUDÍK, Miroslav (Hrsg.): *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* Bd. 15. Fort Lauderdale, FL, USA : PMLR, 11–13 Apr 2011, S. 315–323. – URL http://proceedings.mlr.press/v15/glorot11a.html

[Guyon und Elisseeff 2003] GUYON, I. ; ELISSEEFF, A.: An introduction to variable and feature selection. In: *Journal of machine learning research* 3 (2003), Mar, S. 1157–1182

[Haykin 1999] HAYKIN, Simon S.: Neural Networks: A Comprehensive Foundation. (1999)

[Helleputte und Dupont 2009] HELLEPUTTE, Thibault ; DUPONT, Pierre: Feature Selection by Transfer Learning with Linear Regularized Models, 09 2009, S. 533–547. – ISBN 978-3-642-04179-2

[Huang u. a. 2020] HUANG, Y. ; JIN, W. ; YU, Z. ; LI, B.: Supervised feature selection through Deep Neural Networks with pairwise connected structure. In: *Knowledge-Based Systems* 204 (2020), Nr. 10620, S. 2

[Kumar und Minz 2014] KUMAR, V. ; MINZ, S.: Feature selection: a literature review. In: *SmartCR* 4 (2014), Nr. 3, S. 211–229

[Kuncheva] KUNCHEVA, L. I. (2007 F.: A stability index for feature selection. In: *In Artificial intelligence and applications*

[Law u. a. 2004] LAW, M. H. ; FIGUEIREDO, M. A. ; JAIN, A. K.: Simultaneous feature selection and clustering using mixture models. In: *IEEE transactions on pattern analysis and machine intelligence* 26 (2004), Nr. 9, S. 1154–1166

[LeCun u. a. 1989] LECUN, Y. ; BOSER, B. ; DENKER, J. S. ; HENDERSON, D. ; HOWARD, R. E. ; HUBBARD, W. ; JACKEL, L. D.: Backpropagation applied to handwritten zip code recognition. In: *Neural computation* 1 (1989), Nr. 4, S. 541–551

[Lemhadri u. a. 2021] LEMHADRI, Ismael ; RUAN, Feng ; ABRAHAM, Louis ; TIBSHIRANI, Robert: LassoNet: A Neural Network with Feature Sparsity. In: *Journal of Machine Learning Research* 22 (2021), Nr. 127, S. 1–29. – URL http://jmlr.org/papers/v22/20-848.html

[Li u. a.] LI, Y. ; CHEN, C. Y. ; WASSERMAN, W. W. (2015 A.: Deep feature selection: Theory and application to identify enhancers and promoters.

[Mares u. a. 2016] MARES, M. A. ; WANG, S. ; GUO, Y.: *Combining multiple feature selection methods and deep learning for high-dimensional data.* 2016

[Meinshausen und Buhlmann 2010] Meinshausen, N. ; Buhlmann, P.: Stability selection. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72 (2010), Nr. 4, S. 417–473

[Mitchell 1997] Mitchell, Thomas M.: *Machine Learning.* 1. USA : McGraw-Hill, Inc., 1997. – ISBN 0070428077

[Nilsson u. a. 2007] Nilsson, Roland ; Peña, José ; Björkegren, Johan ; Tegner, Jesper: Consistent Feature Selection for Pattern Recognition in Polynomial Time. In: *Journal of Machine Learning Research* 8 (2007), 03, S. 589–612

[Roy u. a. 2015] Roy, D. ; Murty, K. S. R. ; Mohan, C. K. (2015 J.: Feature selection using deep neural networks. (2015), S. 1–6

[Shannon 1948] Shannon, C. E.: A mathematical theory of communication. In: *The Bell system technical journal* 27 (1948), Nr. 3, S. 379–423

[Sun u. a. 2017] Sun, Kai ; Huang, Shao-Hsuan ; Wong, David Shan-Hill ; Jang, Shi-Shang: Design and Application of a Variable Selection Method for Multilayer Perceptron Neural Network With LASSO. In: *IEEE Transactions on Neural Networks and Learning Systems* 28 (2017), Nr. 6, S. 1386–1396

[Verikas und Bacauskiene 2002] Verikas, A. ; Bacauskiene, M.: Feature selection with neural networks. In: *Pattern recognition letters* 23 (2002), Nr. 11, S. 1323–1335