Ca' Foscari University of Venice

Department of Environmental Sciences, Informatics and Statistics

Master Degree in Computer Science
Second cycle (D.M. 270/2004)

Final Thesis

# Feature Selection Using Neural Network Pruning

**Supervisor**
Ch. Prof. Marcello Pelillo

**Graduand**
Alberto Scalco
Matriculation Number: 846175

**Academic year**
2017-2018

**Abstract**

Feature selection is a well known technique for data prepossessing with the purpose of removing redundant and irrelevant information with the benefits, among others, of an improved generalization and a decreased curse of dimensionality. This paper investigates an approach based on a trained neural network model, where features are selected by iteratively removing a node in the input layer. This pruning process, comprise a node selection criterion and a subsequent weight correction: after a node elimination, the remaining weights are adjusted in a way that the overall network behaviour do not worsen over the entire training set. The pruning problem is formulated as a system of linear equations solved in a least-squares sense. This method allows the direct evaluation of the performance at each iteration and a stopping condition is also proposed. Finally experimental results are presented in comparison to another feature selection method.

# Contents

## 1 Introduction

In machine learning, while is fundamental that the information contained in the data is sufficient to infer the output, the presence of irrelevant or redundant features can lead to loss of performance and burden the entire process since classifiers and regressors are sensitive to the features used to construct it. Under the latter assumption, feature reduction aims to select a subset of features that retains most of the intrinsic information in the data. feature reduction can be separated in *feature extraction*, which transforms the original features and select a subset of it (PCA and discriminant analysis are well-known examples of this class), while *feature selection* directly select a subset from the initial features.
Exhaustive search is generally impractical so, many algorithms has been proposed in order to obtain reliable solutions. The choice of optimality criteria is also difficult as there are multiple objectives in a feature selection task. Many popular search approaches use greedy hill climbing, which iteratively evaluate a candidate subset of features, then modify the subset and evaluate if the new subset provide an improvement over the previous. Evaluation of the subsets requires a scoring metric that grades a subset of features. Some proxy measures like information theory based scoring, can be employed like in [1] where a combination of the concept of mutual information between features and outputs and pairwise mutual information between features is applied. Genetic algorithms are search heuristic that mimics the process of natural evolution and can be used as framework for the extraction of relevant features ([2] and [3]). Other graph based methods includes clustering approaches like dominant sets, where nodes represent features and the similarity are computed through mutual information and entropy([4]).
This thesis investigates a feature selection method based on neural network. The main idea is to apply a pruning method, which are generally applied to hidden layers, to the input performing a backward feature selection by iteratively removing input nodes. In this context, the pruning can be interpreted as feature selection since the elimination of an input nodes implies the removal of a feature. After each iteration, the pruning algorithm used [5], adjusts the remaining weights minimizing the performance loss of the network. The procedure allows to be iterated until the required trade-off between feature reduction and performance is reached. The performance metrics can be detected at each step providing flexibility on the choice of the stopping criterion. It is also possible to adopt any stopping criterion based directly on the network performance, instead of using an estimating measure of quality, thus having a more reliable evaluation.
In the following sections, first the notation and definitions are presented in Section 2, The proposed algorithm is discussed in Section 3, an introduction to a comparison method is shown in Section 4, while in in Section 5 experimental results are provided. Lastly in Section 6 conclusions are reported.

## 2 Definitions and notations

A neural network (or model) can be represented a directed weighted graph $N = (V, E, w)$ where $V = \{0, 1, \ldots, n\}$ is the set of $n + 1$ units (or neurons), $E \subseteq V \times V$ is the set of connections and $w : E \rightarrow R$ is a weight function that assigns to any connection $(i, j) \in E$ a real value $w(i, j)$ (or $w_{ij}$ for ease of notation).
For each unit $i \in V$ the *projective field*, which is the set of unit fed by $i$, can be defined as follow

$$P_i = \{j \in V : (i, j) \in E\} \tag{1}$$

In the same way the *receptive field* of $i$, which is the set of units that feed $i$

$$R_i = \{j \in V : (j, i) \in E\} \tag{2}$$

Where the cardinalities of the sets are respectively denoted as $p_i$ and $r_i$.
Considering a fully connected network, the receptive field of a given node is the preceding layer, while the projective field is the subsequent layer.
Being $V = V_I \cup V_H \cup V_O$ partitioned in the sets of input, hidden and output nodes, input nodes $j \in V_I$ receive their input $x_j$ from the external that also corresponds to their output ($x_j = y_j$), while every non-input unit $i \in (V_H \cup V_O)$ instead, receive from its own receptive field $R_i$ an input given by

$$\xi_i = \sum_{j \in R_i} w_{ji} y_j \tag{3}$$

where $y_j$ represents the output value of the unit $j$, and provide as output

$$y_i = f(\xi_i) \tag{4}$$

where $f$ is an arbitrary differentiable activation function.

## 3 The feature selection algorithm

The variable selection method is presented as an extension of an iterative pruning method already presented in [5] and [6]. The algorithm consists in iteratively selecting a input unit to remove after the network has been satisfactory trained performance-wise. It is assumed that the network is trained over the set of $M$ patterns $x^m = (x_1, \dots, y_n)\, m = 1, \dots, M$ by means of an arbitrary learning procedure from which the pruning algorithm is completely independent.
Suppose that an input node $h \in V_I$ is chosen to be removed (the elimination criterion will be addressed below). Removing $h$ implies the removal of all its outgoing connection. The new pruned network will have the following connection set

$$E_{new} = E_{old} - (\{h\} \times P_h) \tag{5}$$

After node removal, it is required to adjust the weights of $h$'s projective field $P_h$ to preserve the network behaviour over the training set. For each unit $i \in P_h$, its net input upon presentation of pattern $\mu$ is

$$\xi_i^{(\mu)} = \sum_{j \in R_i} w_{ji} y_j^{(\mu)} \tag{6}$$

where $y_j^{(\mu)}$ denotes the the output of the input unit $j$ corresponding to pattern $\mu$. After removal of $h$, each $i$ unit in the $P_h$ will receive its input from $R_i - \{h\}$. In order to let the pruned model be as close as possible to previous, need to hold the following relation:

$$\forall \mu \qquad \sum_{j \in R_i} w_{ji} y_j^{(\mu)} = \sum_{j \in R_i - \{h\}} (w_{ji} + \delta_{ji}) y_j^{(\mu)} \tag{7}$$

where $\mu = 1, \dots, M$ and $i \in P_h$, where $\delta_{ij}$'s are the adjusting factors to be determined. After an algebraic manipulation the latter expression can be written as

$$\forall \mu \qquad \sum_{j \in R_i - \{h\}} \delta_{ij} y_j^{(\mu)} = w_{hi} y_h^{(\mu)} \tag{8}$$

(8) is a system of $MP_h$ linear equations in the $\kappa_h = \sum_{i \in P_h} (r_i - 1)$ unknowns $\{\delta_{ji}\}$, where $\kappa_h$ represent the number of incoming connections into $h$'s projective field $P_h$ after $h$ removal.
The system can be conveniently represented in a matrix notation in the form $\boldsymbol{A\delta = b}$: for each unit in the input layer $j \in V_I$ their input vectors

$$\overline{y}_j = (y_j^{(1)}, \dots, y_j^{(M)})^T \tag{9}$$

where $y_j^{(\mu)}$ are the inputs (and the outputs) of the node $j$ on presentation of pattern $\mu \in M$.
Also, being $r_i$ the cardinality of $R_i$, let $Y_{i,h}$ denote the $M \times (r_i - 1)$ matrix, whose columns are the output vectors of $i$'s new receptive field $R_i - \{h\}$:

$$Y_{i,h} = [\overline{y}_{j1} \quad \overline{y}_{j2} \quad \dots \quad \overline{y}_{j_{r_i - 1}}] \tag{10}$$

where the indexes $j_k$, for all $k = 1, \dots, r_i - 1$, vary in $R_i - \{h\}$.
The weight adjustment can be performed by solving the $p_h$ disjoint system:

$$Y_{i,h}\overline{\delta}_i = \overline{z}_{i,h} \tag{11}$$

for every $i \in P_h$, where $\overline{\delta}_i$ is the unknown vector and:

$$\overline{z}_{i,h} = w_{hi}\overline{y}_h \tag{12}$$

Putting these systems together, we obtain

$$\overline{z}_h - Y_h\overline{\delta} \tag{13}$$

where

$$Y_h = diag(Y_{i_1,h}, Y_{i_2,h}, \ldots, Y_{i_{p_h},h}) \tag{14}$$

$$\overline{\delta} = (\overline{\delta}_{i_1}^T, \overline{\delta}_{i_2}^T, \ldots, \overline{\delta}_{i_{p_h}}^T)^T \tag{15}$$

$$\overline{z}_h = (\overline{z}_{i_1,h}^T, \overline{z}_{i_2,h}^T, \ldots, \overline{z}_{i_{p_h},h}^T)^T \tag{16}$$

One iteration of the pruning algorithm consists on solving the following linear system in the least-square sense:

$$minimize \left\|\overline{z}_h - Y_h\overline{\delta}\right\|_2 \tag{17}$$

To solve the system of equations (17), In [5] is applied a conjugate gradient least-square method called CGPCNE algorithm (details in [7]) to perform this minimization problem on whole $Y_h$ matrix. However even for discrete dataset size, the computation may become unfeasible in terms of space, since the $Y_h$ matrix dimension is $Mp_h \times \kappa_h$ where $\kappa_h = \sum_{i \in P_h}(r_i - 1)$.
Because of the structure of the $Y_h$ matrix which is block diagonal, each block can be computed independently from the others, so the definition of $\overline{\delta}$ is computed iteratively on each $Y_{i,h}$, thus:

$$\forall i \in P_h \qquad minimize \left\|\overline{z}_{i,h} - Y_{i,h}\overline{\delta_i}\right\|_2 \tag{18}$$

## 3.1 Elimination criterion

Ideally the best choice of the input unit $h$ will be the one among all input nodes which will lead to the smallest final residual of the system (17). This will guarantees that the weights adjustment will have the minimal impact on the overall network behaviour. This approach however, implies solving as many systems as there are input nodes and it would become impractical even for small number of units.
The identification of the input unit $h$, is instead based on a property of the the CGPCNE method adopted in [5]. The latter method solves the linear system by starting with an initial $\delta_0$ and iteratively produces a sequence of $\{\delta_k\}_{k=1,2,\ldots}$ that decrease the residuals $\rho_h(\overline{\delta}_k) = \left\|\overline{z}_h - Y_h\overline{\delta}_k\right\|_2^2$. A suboptimal approach is to chose the unit $h$ that has the smallest initial residual

$$h = \arg\min_{h \in V_I} \rho_h(\overline{\delta}_0) \tag{19}$$

Since the initial solution $\overline{\delta}_0$ is chosen to be the null vector, (19) can be written as

$$h = \arg\min_{h \in V_I} \sum_{i \in P_h} w_{hi}^2 \left\| \overline{y}_h \right\|_2^2 \tag{20}$$

The latter selection criterion have an interesting interpretation. The quantity defined as $\sum_{i \in P_h} w_{hi}^2 \left\| \overline{y}_h \right\|_2^2$ is nearly identical to the measure of "goodness" of an individual unit activity proposed in [8]. This can be interpreted as a criteria that select among all input units the one with the smallest synaptic activity.

## 3.2 DGELSD - the least square algorithm

The employed algorithm for solving systems in the least square sense is the DGELSD routine in LAPACK package. By means of different approaches the algorithm solve generic systems in the form

$$minimize \left\| b - Ax \right\|_2 \tag{21}$$

However, for overdetermined systems where $A \in \mathbb{R}^{m \times n}$ and $m \geq n$ (such as those accounted in the experimental results section), the routine first perform a QR factorization, namely $A = QR$ where $Q$ is an orthogonal matrix and $R$ an upper triangular matrix and subsequently computes the vector $x$ by solving

$$x = R^{-1} Q^T b \tag{22}$$

The QR factorization is performed via *Householder transformations* leading to $Q$ and $R$ as follows: select the first $m$-dimensional column vector $x$ of matrix $A$, computes

$$u = x - \left\| x \right\|_2 e_1 \tag{23}$$

where $e_1$ is a $m$-dimensional vector $(1, 0, \ldots, 0)^T$. The vector $u$ is normalized

$$v = \frac{u}{\left\| u \right\|_2} \tag{24}$$

and the Householder matrix $Q1$ computed as

$$Q_1 = I - 2vv^T \tag{25}$$

where $I$ is the $m$-by-$m$ identity matrix. Computing the following, is it possible to gradually transform $A$ to upper triangular form

$$Q_1 A = \begin{bmatrix} \left\| x \right\|_2 & * & \ldots & * \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{bmatrix} \tag{26}$$

This operations can be repeated for A$'$ (obtained from $Q_1 A$ by deleting the first row and first column), resulting in a Householder matrix $Q_2$. The number of iterations $t$ is defined as $t = min(m - 1, n)$.

Note that $Q_2$ is smaller than $Q_1$. Since it is wanted to operate on $Q_1 A$ instead of $A'$ it is required to expand it to the upper left, filling in a 1, or in general:

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & Q'_k \end{pmatrix} \tag{27}$$

When $t$ iterations are reached $Q$ and $R$ can be derived as follow

$$Q = Q_1^T Q_2^T \dots Q_k^T \tag{28}$$

$$R = Q^T A \tag{29}$$

Thus leading to a numerically stable QR factorization of $A$, and permitting to solve $x$ via (22).

## 3.3 Algorithm definition

The algorithm is defined as follows: starting from a satisfactorily trained network $N^{(0)} = (V^{(0)}, E^{(0)}, w^{(0)})$, with $V_I^{(0)}$ equivalent to the complete set of feature, the procedure iteratively construct a sequence of networks $\{N^{(k)}\}$ identifying the unit $h \in V_I$ to remove with the already defined selection criterion, resolving the system (13) in the least square sense to compute the weights adjustment $\overline{\delta}^{(k)}$, removing the unit $h$ along with its incoming and outgoing connections and finally adjust the remaining weights. More explicitly the algorithm can be express as follows.

**Algorithm 1** Pruning algorithm

```
k := 0
repeat
    choose $h \in V_I^{(k)}$ according to rule (20)
    compute $\overline{\delta}$ that solves (17) in the least square sense
    construct $N^{(k+1)} = (V^{(k+1)}, E^{(k+1)}, w^{(k+1)})$ as follows:
        $V^{(k+1)} := V^{(k)} - \{h\}$
        $E^{(k+1)} := E^{(k)} - (\{h\} \times P_h^{(k)})$
        $w^{(k+1)} := \begin{cases} w_{ji}^{(k)}, & \text{if } i \notin P_h \\ w_{ji}^{(k)} + \delta_{ji}, & \text{if } i \in P_h \end{cases}$
        k := k + 1

until the stopping condition on $N^{(k+1)}$ is met
```

Given the iterative structure of the algorithm, a certain flexibility is allowed in the choice of the stopping condition to better fit the requirements. At each iteration the network performance can be evaluated and compared with the original, for instance, in terms of loss o recognition rate if it is a classification problem. Moreover is not necessary to retrain the network after pruning since the weights are already updated.

### 3.3.1 Fully connected algorithm optimization

In the algorithm definition, the $\overline{\delta}$ computation by solving (17) in the least square sense can be further improved under the assumption that a fully connected network is employed. As shown earlier, solving (17) is equivalent to solving (18). Recalling (11), the latter equation can be written as follows

$$\forall i \in P_h \qquad Y_{i,h}\overline{\delta}_i = \overline{y}_h w_{hi} \tag{30}$$

Assuming the network fully connected, each unit $i \in P_h$ have the same receptive field $R_i = R_k$ where $\forall i, k \in P_h$, thus the outputs $\overline{y}_j$ for all $j \in R_i - \{h\}$ are the same for each unit $i$. Namely

$$\forall i, k \in P_h \qquad Y_{i,h} = Y_{k,h} \tag{31}$$

Being $\overline{y}_h$ independent from unit $i$ and $w_{hi}$ a scalar, by exploiting algebraic properties, it is possible to simply solve a single system of equation in the least square sense.
Considering the new system of equations

$$Y_{i,h}\overline{\delta}_x = \overline{y}_h \tag{32}$$

and its least square sense minimization problem

$$minimize \left\|\overline{y}_h - Y_{i,h}\overline{\delta}_x\right\|_2 \tag{33}$$

where $\overline{\delta}_x$ is a generic resulting vector, and considering the vector of the outgoing weights from $h$ to any node $i \in P_h$ so defined

$$w_h = (w_{hi_1}, w_{hi_2}, \dots, w_{hi_{p_h}})^T \qquad i_k = 1, \dots, p_h \tag{34}$$

each correction vector $\overline{\delta}_i$ can be later computed by the outer product with the vector $w_h$

$$\Delta = \overline{\delta}_x \otimes w_h = \overline{\delta}_x w_h^T \tag{35}$$

and then selecting the $i$-th column vector of the $\Delta$ matrix

$$\forall i \in P_h \qquad \overline{\delta}_i = \Delta(:, i) \tag{36}$$

## 3.4 stopping condition

In order to enhance both the generalization and the feature reduction of the final model, the feature selection algorithm is stopped when a decrease of 5% on the validation loss from the original model is reached. The use of the validation, instead of the training set, provide more accurate estimation of the loss function over new unseen data, while the threshold of 5% is a good compromise between feature number and drop of the loss function. Moreover the validation loss is a always defined metric regardless the problem under examination.
Note that any any stopping condition can be defined by the experiment designer, for instance a predefined number of selected feature or a recognition rate threshold can be employed if the problem allows it.

## 4 MIFS - Comparison algorithm

The proposed feature selection algorithm was confronted with another called MIFS presented in [1]. The procedure is based on the concept of mutual information between the features $F$ and the outputs (or classes) $C$. It identifies the best feature to keep and then further picks the remaining by putting in relation the mutual information between the feature $f$ and the class set $C$ $I(C;f)$, with the mutual information of any $f \in F$ and the features already chosen $s \in S$, $I(f;s)$.
First, It computes the mutual information for each feature $f \in F$ with the class set $C$ defined as $I(C,f)$. Then it chooses the first feature to be kept in $S$ as the one that maximizes the mutual information $\arg\max_{f \in F} I(C;f)$. Afterwards, it applies a greedy iterative approach for the subsequent features: at each step, computes the mutual information among all couples of features in $f \in F$ and $s \in S$, namely $I(f;s)$ and then chose the new feature $f \in F$ to be moved in the feature to keep $s \in S$ by solving the following

$$\arg\max_{f \in F} I(C;f) - \beta \sum_{s \in S} I(f;s) \tag{37}$$

The greedy approach is performed until the cardinality of $S$ reaches the required number of features.
In the following experiments, although parameter $\beta$ was suggested in [1] to be in range $[0.5, 1]$, it was fine-tuned in the range $[0.1, 1]$ to give the best results over the chosen dataset.

# 5 Experimental results

To evaluate the effectiveness of the feature selection algorithm, analysis are conducted on three different problems, two of which can be addressed as classification problems and the last as a regression problem. For each of the three experiments, 10 independent model were trained each of which are composed of three fully connected layers (input, hidden and output layer). The training stage was performed by means of a early stopping criterion based of the validation loss function: the training was performed until there were an improvement on the validation loss function with a patience parameter set to 10 (the loss can degrades up to 10 epochs) and the best model over all epochs was retained as final. The parameters such as the number of hidden nodes and activation functions are properly chosen until a good validation loss value were reached. Moreover the Adam optimizer was adopted for the training phase and Xavier initialization for the initial weights.

## 5.1 Reuters newswire topics classification

The dataset is comprised of 11'228 newswires from Reuters, labeled over 46 topics. Each wire is encoded as a sequence of word indexes. For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the entire dataset. The top 2000 words are considered and each newswire is represented as 'bag of words' frequency vector. each pattern is thus transformed into a $1 \times 2000$ vector, while the classes are represented as a one-hot vector of dimension $1 \times 46$. Training set is composed of 7186, validation set of 1796 and test set of 2246. Ten independently initialized 2000-128-46 models where trained. Hidden layer uses a *sigmoid* activation while *softmax* is applied to output layer, with a *categorical cross-entropy* loss function.

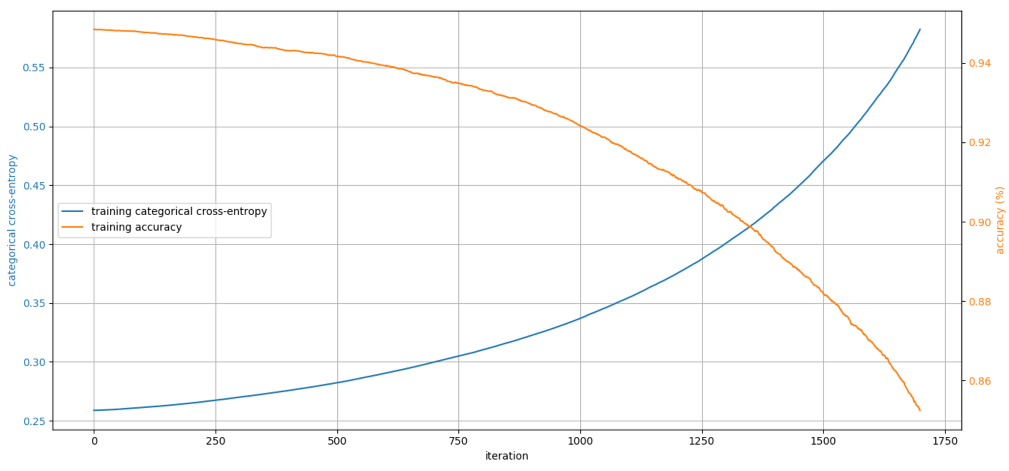

Figure 1: Reuters average training categorical cross-entropy and accuracy for the 10 models at each iteration.

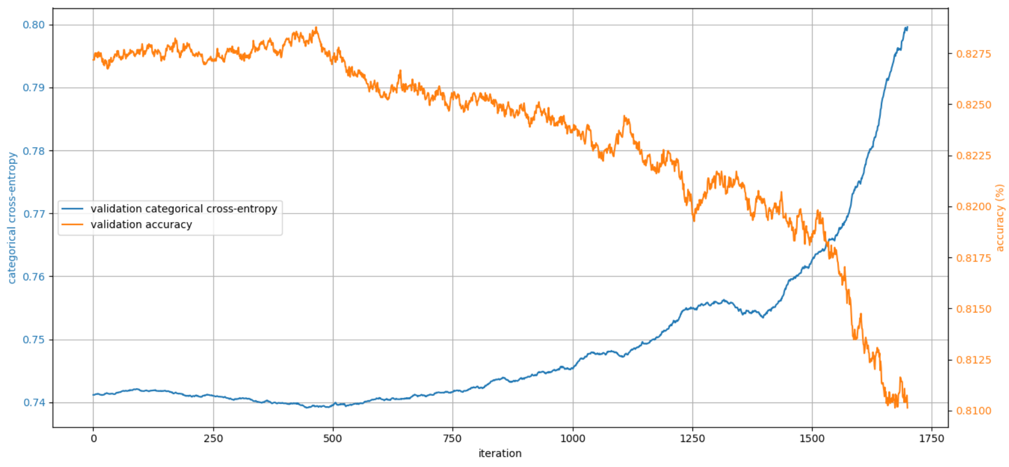

Figure 2: Reuters average validation categorical cross-entropy and accuracy for the 10 models at each iteration.

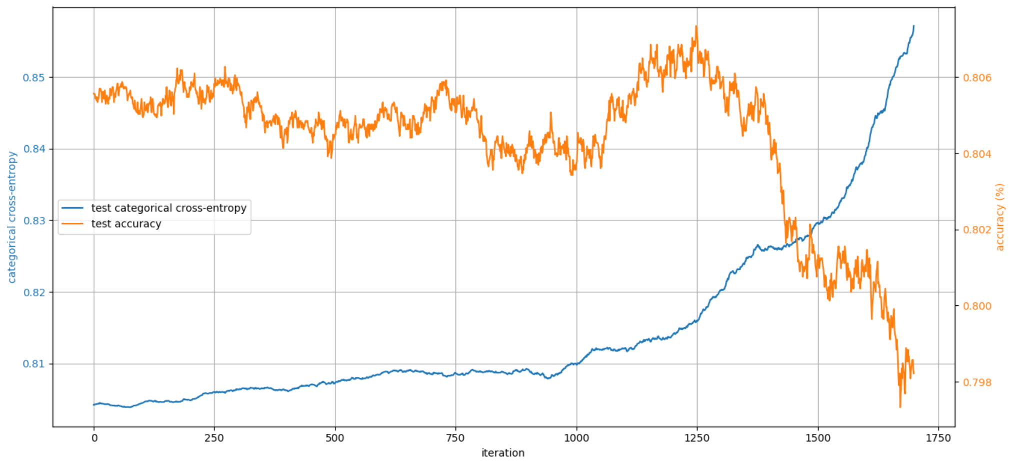

Figure 3: Reuters average test categorical cross-entropy and accuracy for the 10 models at each iteration.

In Figure 1, Figure 2 and Figure 3 The average of the 10 models of loss and recognition rate are shown for training, validation and test set respectively. As can be intuitively deducted, as the number of iterations increases, the accuracy decreases and loss function degrades. Accuracy and loss function are more affected in the training set, compared with validation and test set, where the variations are more restrained. The average number of selected feature is of 388 (Table 1), corresponding to iteration 1612.

| | | test categorical cross-entropy | | | test accuracy | | |
|---|---|---|---|---|---|---|---|
| model | features | original | proposed method | MIFS | original | proposed method | MIFS |
| 1 | 364 | 0.8007 | 0.8404 | 0.9412 | 0.8085 | 0.8032 | 0.7761 |
| 2 | 390 | 0.8004 | 0.8319 | 0.9407 | 0.8077 | 0.8032 | 0.7757 |
| 3 | 392 | 0.8080 | 0.8410 | 0.9409 | 0.8032 | 0.7988 | 0.7753 |
| 4 | 381 | 0.7945 | 0.8375 | 0.9387 | 0.8032 | 0.8041 | 0.7751 |
| 5 | 388 | 0.7999 | 0.8328 | 0.9399 | 0.8077 | 0.8014 | 0.776 |
| 6 | 398 | 0.8016 | 0.8376 | 0.9308 | 0.8054 | 0.8050 | 0.7777 |
| 7 | 377 | 0.8146 | 0.8514 | 0.9375 | 0.8014 | 0.7970 | 0.7762 |
| 8 | 373 | 0.8083 | 0.8576 | 0.9388 | 0.8077 | 0.7988 | 0.7768 |
| 9 | 411 | 0.8055 | 0.8388 | 0.9353 | 0.8054 | 0.7992 | 0.7777 |
| 10 | 401 | 0.8091 | 0.8430 | 0.9314 | 0.8054 | 0.8014 | 0.7787 |
| average | 387.5 | 0.8043 | 0.8412 | 0.9375 | 0.8056 | 0.8012 | 0.7766 |
| standard deviation | 14.09 | 0.00587 | 0.00794 | 0.00381 | 0.00237 | 0.00268 | 0.00118 |

Table 1: The Reuters test loss function and test accuracy for each of the 10 models. MIFS values are averaged among 5 trials ($\beta = 0.5$).

After the application of the proposed stopping condition for the feature selection, to evaluate the quality of the results, a comparison with the MIFS algorithm ha been done. A number of features $k_m$ were extracted for the 10 models. For each of them, MIFS where subsequently applied with a number of features equal to its corresponding model. The MIFS resulting features were used for training 5 independently initialized network with the same model structure ($k_m$-128-46) with training procedure explained earlier. Detailed results are presented in Table 1.
The proposed algorithm provided a great reduction on number of features, on average, 19% of the total amount were retained. The final pruned models provided a decrease of the 4.6% for the categorical cross-entropy and 0.44% for the recognition rate. MIFS instead had drop of 16.5% and 2.9% for the loss function and accuracy respectively. MIFS performed slightly worse for both loss function and recognition rate for all the 10 trials.
Recalling that in Reuters the words are represented with an index that rank the word by its frequency over the entire dataset, to provide an overall view of how the algorithms behave, Figure 4 shows three examples of the feature selected by the proposed algorithm and Figure 5 those selected by MIFS. Both algorithms tend to recognize as import the most frequent words. MIFS feature choice just follow the word frequency with some exceptions where certain frequent words like the "said","is", "he", "but", "cash" are excluded. The proposed method instead, select other less frequent words (like "bureau", "Venezuela", "brokerage", "bankruptcy") and exclude some of the most frequent. This choice can be considered favorable by

the fact that while some very frequent words might by irrelevant on the choice of belonging class, some less frequent can be very descriptive of the newswire topics.

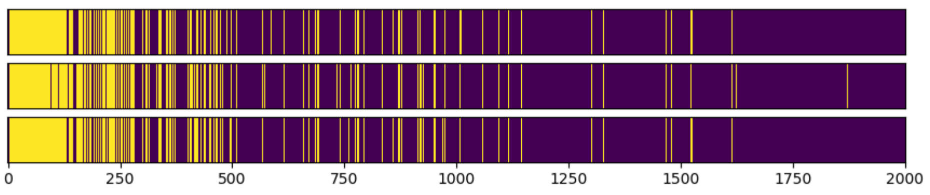

Figure 4: Visualizations of 3 trails results for Reuters with the proposed method with respectively {364, 390, 392} features. The abscissa represent the word index. Yellow vertical lines depict the selected features.

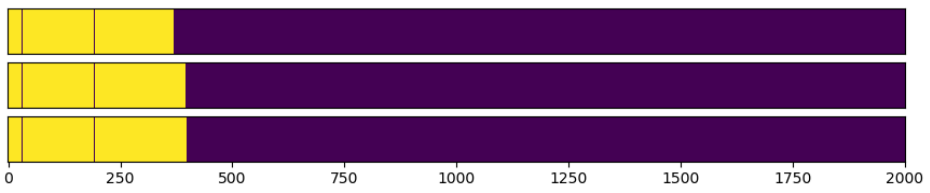

Figure 5: Visualizations of 3 trails results for Reuters with the MIFS Algorithm with respectively {364, 390, 392} features. The abscissa represent the word index. Yellow vertical lines depict the selected features.

## 5.2 MNIST handwritten digits

MNIST is a well-known dataset of grayscale $28 \times 28$ pixel images consisting of handwritten digits (from 0 to 9). The aim is to classify the greatest amount of images in its corresponding class $|C| = 10$. The number of training patterns are 48'000, the validation set is composed of 12'000 images, while those in the test set are 10'000. The intensity values that range from 0 to 255 where normalized, and the inputs linearized as vector of $1 \times 784$ size. The classes where represented as a one-hot vector of dimension $1 \times 10$.
Ten independent 784-256-10 models where trained with a *categorical cross-entropy* loss function. Hidden layer employs a *sigmoid* activation, while the output uses *softmax*. In order to analyze the behaviour of the proposed feature selection, the algorithm was applied until all but a single feature was left.

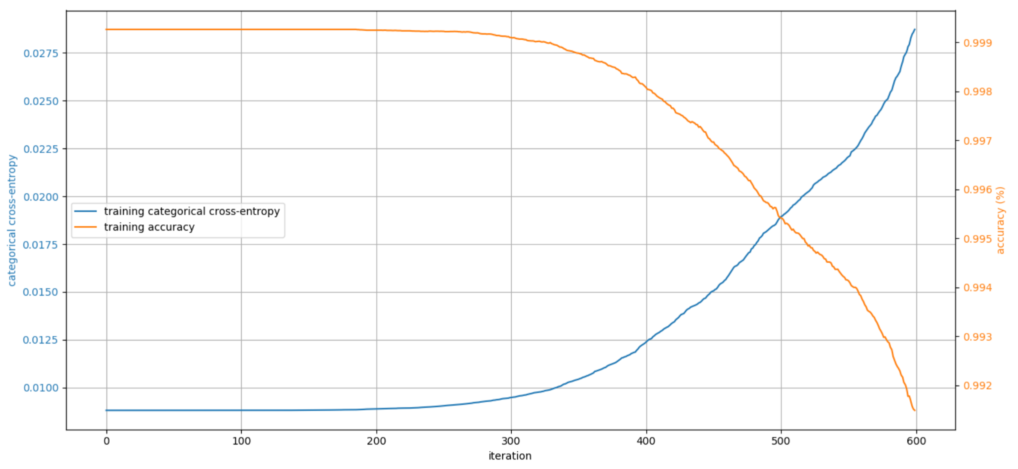

Figure 6: MNIST average training categorical cross-entropy and accuracy for the 10 models at each iteration.

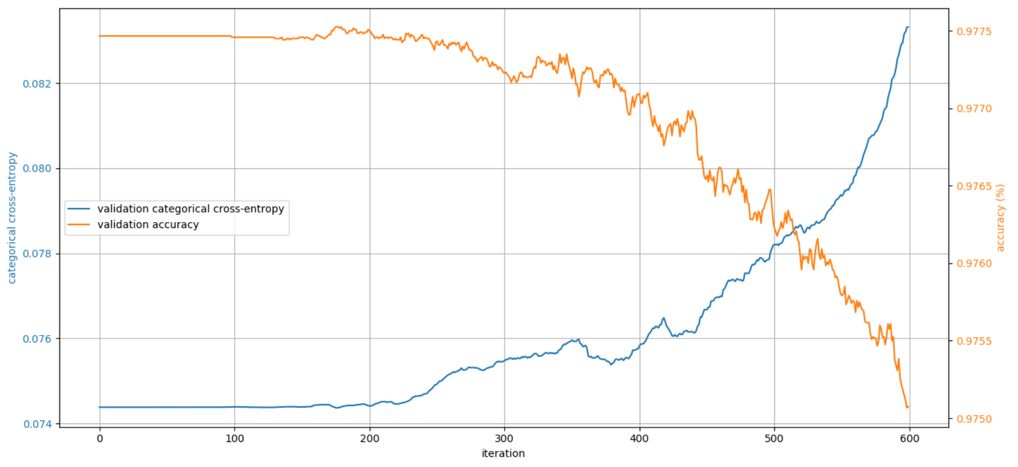

Figure 7: MNIST average validation categorical cross-entropy and accuracy for the 10 models at each iteration.

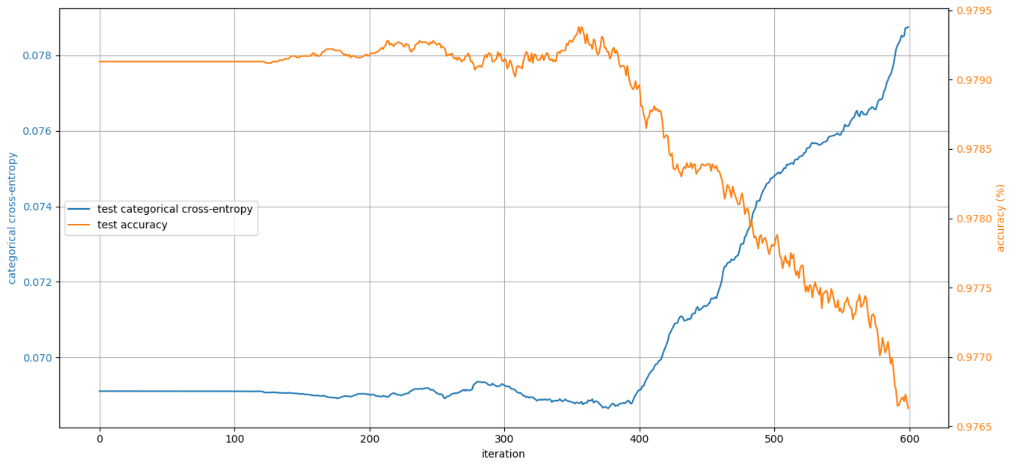

Figure 8: MNIST average test categorical cross-entropy and accuracy for the 10 models at each iteration.

In Figure 6, Figure 7 and Figure 8 training, validation and test set metrics as the average of loss and recognition rate for the ten models are reported. The proposed algorithm shows a stable behaviour without significant deterioration over the training set until iteration 300. Despite the behaviour over the test data is affected by more variance, the loss function fluctuate around the value of the original network up to 400 iterations, before worsening. From Table 2 the average number of feature extracted with the application of the stopping condition provided on average 284 features, corresponding to iteration 500.

| | | test categorical cross-entropy | | | test accuracy | | |
|---|---|---|---|---|---|---|---|
| model | features | original | proposed method | MIFS | original | proposed method | MIFS |
| 1 | 321 | 0.0678 | 0.0711 | 0.1130 | 0.9785 | 0.9782 | 0.9671 |
| 2 | 277 | 0.0700 | 0.0768 | 0.1105 | 0.9784 | 0.9761 | 0.9667 |
| 3 | 233 | 0.0717 | 0.0767 | 0.1037 | 0.9791 | 0.9783 | 0.9678 |
| 4 | 302 | 0.0682 | 0.0728 | 0.1104 | 0.9794 | 0.9793 | 0.9667 |
| 5 | 288 | 0.0701 | 0.0753 | 0.1070 | 0.9790 | 0.9775 | 0.9679 |
| 6 | 246 | 0.0679 | 0.0741 | 0.1053 | 0.9792 | 0.9781 | 0.9676 |
| 7 | 307 | 0.0692 | 0.0729 | 0.1100 | 0.9796 | 0.9777 | 0.9671 |
| 8 | 272 | 0.0674 | 0.0759 | 0.1095 | 0.9798 | 0.9784 | 0.9672 |
| 9 | 310 | 0.0693 | 0.0716 | 0.1094 | 0.9790 | 0.9777 | 0.967 |
| 10 | 283 | 0.0693 | 0.0755 | 0.1111 | 0.9793 | 0.9778 | 0.9666 |
| average | 283.9 | 0.0691 | 0.0743 | 0.1090 | 0.9791 | 0.9779 | 0.9672 |
| standard deviation | 28.19 | 0.00130 | 0.00206 | 0.00282 | 0.00044 | 0.00082 | 0.00045 |

Table 2: The MNIST test loss function and test accuracy for each of the 10 models. MIFS values are averaged among 5 trials ($\beta = 0.1$).

As performed in the previous experiment, a comparison between the proposed algorithm and MIFS is presented in Table 2. The application of the proposed stopping condition for the feature selection, on average provides a number of 284 features, that is roughly the 36% of the total number features. The average difference of the recognition rate from the original network is 0.12% for the proposed method and 1,2% for MIFS, while the drop of the loss function is 7.5% and 57.7% respectively. It is clear that the proposed method, with equal number of features, performed better than MIFS for both metrics.
Since the features of the problem in consideration have a graphical interpretation, provided the *model 1* as example, Figure 9 for the proposed method and Figure 10 for MIFS, both show the original image of 784 pixels, and the feature selected at some sample iterations. In Figure 9 can be clearly seen how the algorithm first removes pixels on the edges of the image, which can be considered as the *irrelevant* information and only subsequently removing the *redundant* information in the center area of the image. Peculiar is how during the process of *redundant* information removal, the algorithm has the tendency to maintain the locality information (which is characteristic of images) by selecting a pixel and eliminating its neighbors. In fact, despite the decreasing number of feature, digits are still recognizable even with 64 pixels.
MIFS instead even with a small $\beta$ (which represent to some extend the importance of removing *redundant* information) removes both *redundant* and *irrelevant* information as the removal progresses by excluding pixels in the inner part of the image

along with those on the contour of the digits. This is behaviour is derived by the fact that, when the number of selected features in $S$ becomes large, in (37) the second term $\sum_{s \in S} I(f;s)$ overcomes the first $I(C;f)$. In fact, although single values of $I(f;s)$ might result small, an increasing cardinality of $S$ lead to greater values of the summation. This makes gradually more preferable the removal of *redundant* features against those that are *irrelevant*.

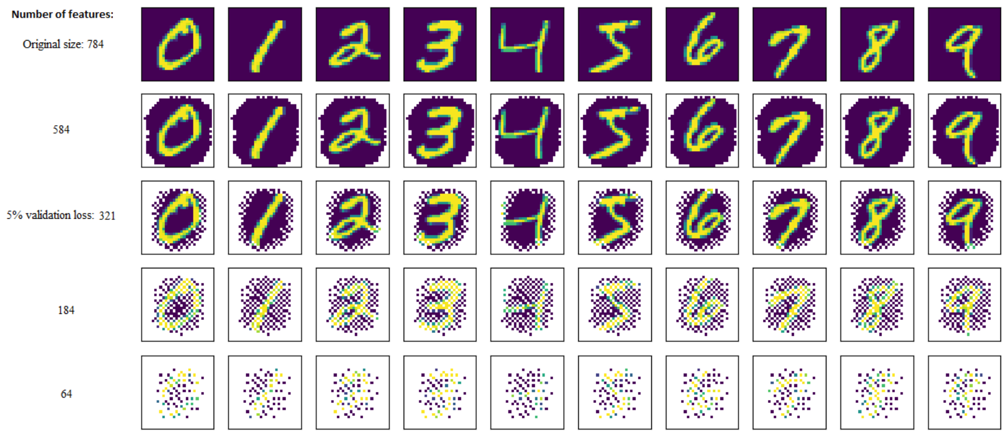

Figure 9: Proposed algorithm image reconstructions for *model 1* with {784, 584, 351, 184, 64} features selected.

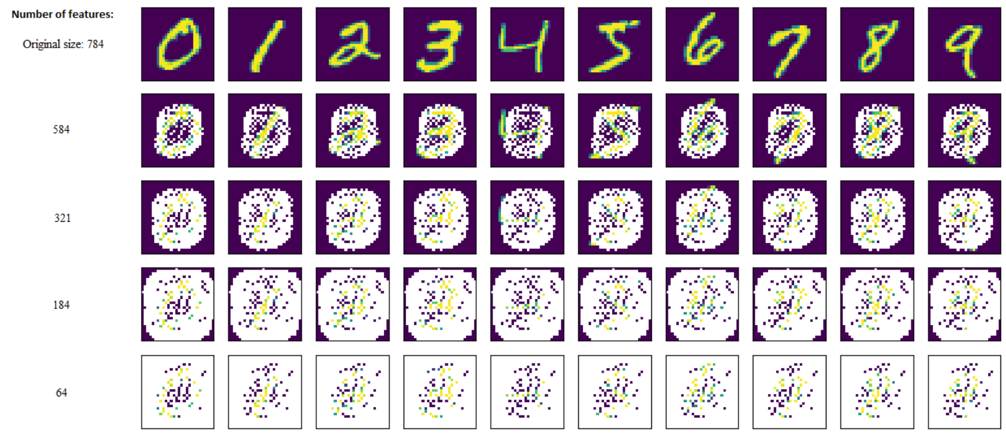

Figure 10: MIFS image reconstructions with {784, 584, 351, 184, 64} features selected ($\beta = 0.1$).

## 5.3 UJIIndoorLoc indoor localization

UJIIndoorLoc is dataset for indoor localization based on WLAN fingerprints. The data covers three buildings of Universitat Jaume I with 4 or more floors and almost 110m$^2$. Even though the dataset can be used for classification (e.g. actual building and floor identification), regression on actual longitude and latitude has been performed. Each WiFi fingerprint can be characterized by the detected Wireless Access Points (WAPs) and the corresponding Received Signal Strength Intensity (RSSI). The intensity values are represented as negative integer values ranging -104dBm (extremely poor signal) to 0dBm. The values are scaled from 0 to 105, where the minimum signal strength is 1 and the maximum is 105. The 0 value denote those WAP that were not detected. The number of features are 520 which represent the different WAPs, thus, the WiFi fingerprint is composed by 520 intensity values. The latitude and the longitude were normalized in the range $[0, 1]$. Other dataset information like building and user id were ignored for the ease of problem representation. The dataset is comprise of 12'760 training, 3'190 validation and 3'987 test patterns. The model topology is 520-400-2 with *sigmoid* activation functions for all layers.

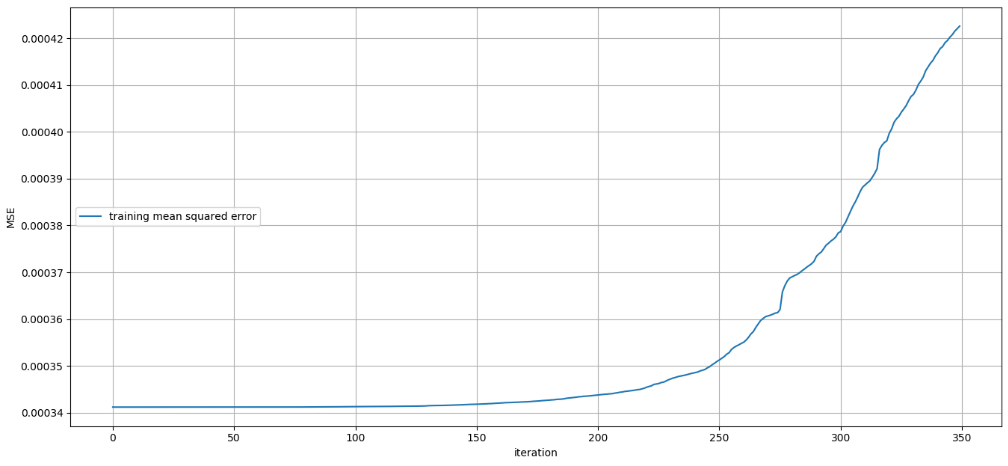

Figure 11: UJIIndoorLoc average training mean squared error for the 10 models at each iteration.

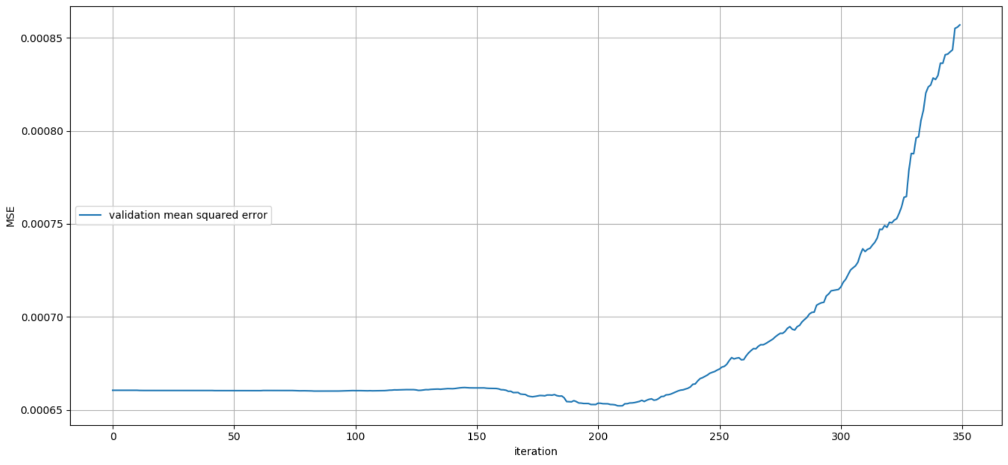

Figure 12: UJIIndoorLoc average validation mean squared error for the 10 models at each iteration.

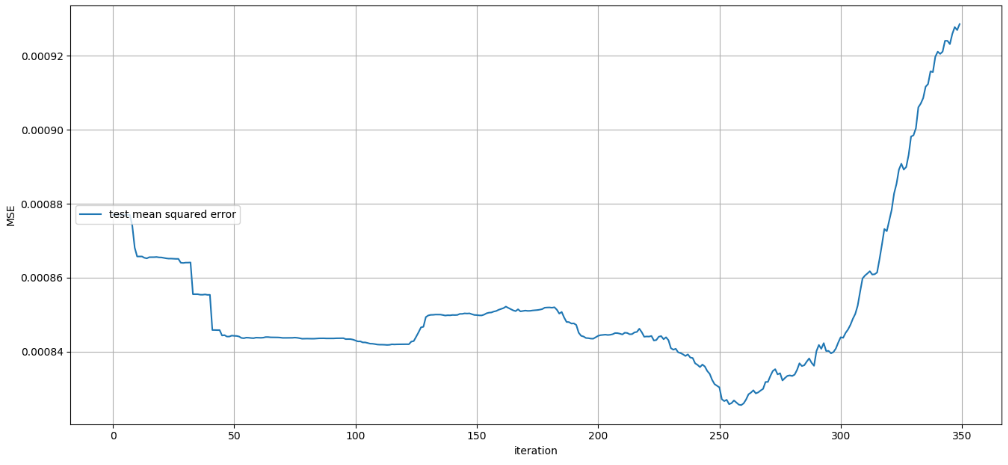

Figure 13: UJIIndoorLoc average test mean squared error for the 10 models at each iteration.

In order to evaluate the behaviour of the proposed algorithm was stressed until a single feature remained. In Figure 11 and Figure 13 the MSE loss function averaged among 10 models on training, validation and test set are presented. While training and validation MSE are constant up to 250 iterations, in the first 50 iterations, the test set shows an improvement of MSE and no significant increase were observed until 300 iterations. As Table 3 shows, on average the stopping condition proposes 243 features (equivalent to 277 iterations). At this iteration, the MSE in the test set is even smaller than the original network.

| | | test mean squared error | | |
|---|---|---|---|---|
| model | features | original | proposed method | MIFS |
| 1 | 262 | 0.00091 | 0.00084 | 0.00133 |
| 2 | 233 | 0.00107 | 0.00099 | 0.00121 |
| 3 | 239 | 0.00085 | 0.00083 | 0.00126 |
| 4 | 231 | 0.00085 | 0.00081 | 0.00120 |
| 5 | 250 | 0.00082 | 0.00080 | 0.00129 |
| 6 | 264 | 0.00089 | 0.00084 | 0.00123 |
| 7 | 231 | 0.00085 | 0.00082 | 0.00119 |
| 8 | 238 | 0.00083 | 0.00078 | 0.00123 |
| 9 | 226 | 0.00090 | 0.00083 | 0.00119 |
| 10 | 255 | 0.00081 | 0.00078 | 0.00122 |
| average | 242.9 | 0.00088 | 0.00083 | 0.00124 |
| standard deviation | 13.80 | 0.000076 | 0.000059 | 0.000044 |

Table 3: The Reuters test loss function and test accuracy for each of the 10 models. MIFS values are averaged among 5 trials ($\beta = 0.1$ and $t = 16$).

Since MIFS works with output classes, using it on regression problems requires to discretize the output values in classes. One way to perform this operation is to rearrange each output value into intervals. Assuming $t$ intervals, having $l = 2$ output values (latitude and longitude), the total amount of classes will be $t^l$.
After a tuning process, the number of intervals was set to $t = 16$ leading to 256 distinct classes.
As can be seen, the networks pruned by the proposed method generalize slightly better than the original models. There were a decrease of the MSE of 5,7%. MIFS instead had a MSE increase of 40.9% on average.

# 6 Conclusions

In this thesis a method for variable selection using neural networks has been analyzed with real-world problems experiments. The key idea consists on iteratively removing features by means on pruning the input layer of a trained network, subsequently adjusting the remaining weights to preserve the network behaviour on training data. The weights adjustment can be formulated in terms of system of linear equations to be solved in the least-squares sense with a performing algorithm based on QR factorization. An optimization for the system of linear equation was also presented when fully connected network are adopted, based on the general idea that a single system is required to be solved for distinct receptive fields. A removal criterion with low computational cost is also proposed which happens to be proportional to an independently developed "goodness" metric for neural units. However, provided the flexibility of the feature selection method, both the latter criterion and algorithm for solving the linear system can be freely chosen without altering the entire process, providing an interesting cue for further investigations.
Moreover, the iterative nature of the algorithm, allow the experiment designer to monitor the network performance at each step and employ a stopping condition that is more suitable for the problem resolution. Another benefit of this method is that do not require any parameter tuning process that might result tedious. The experimental results obtained also in comparison to the other feature selection technique, prove that the method does a very good job of reducing the feature set size while preserving excellent performance.

## References

[1] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions On Neural Networks*, 5(4):537–550, 1994.

[2] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49, 1998.

[3] F.Z. Brill, D.E. Brown, and W.N. Martin. Fast genetic selection of features for neural-network classifiers. *IEEE Transactions on Neural Networks*, 3(2):324–328, 1992.

[4] Z. Zhang and Edwin R. Hancock. Mutual information criteria for feature selection. In *Similarity-based pattern recognition: fisrt international workshop, SIMBAD 2011*, volume 7005 LNCS, pages 235–249, Berlin, 2011. Springer-Verlag Berlin.

[5] G. Castellano, A.M. Fanelli, and M. Pelillo. An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions On Neural Networks*, 8(3):519–531, 1997.

[6] R. Reed. Pruning algorithms—a review. *IEEE Transactions On Neural Networks*, 4(1):740–747, 1991.

[7] A. Björck and T. Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT*, 19(2):145–163, 1979.

[8] K. Murase, Y. Matsunaga, and Y. Nakade. A backpropagation algorithm which automatically determines the number of association units. In *Proc. Int. J. Conf. Neural Networks*, pages 783–788, Singapore, 1991.

[9] E.D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[10] G. Castellano and A.M. Fanelli. Variable selection using neural-network models. *Neurocomputing*, 31:1–13, 2000.

[11] F. Ling, D. Manolakis, and J. Proakis. A recursive modified gram-schmidt algorithm for least- squares estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):829–836, 1986.

[12] J. Stoerand and R. Bulirsch. Introduction to numerical analysis. chapter 4, pages 223–230. Springer, 3rd edition, 2002.

[13] R. P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22, 1987.

[14] LAPACK: Linear Algebra PACKage. Dgelsd. `http://www.netlib.org/lapack/explore-html/d7/d3b/group__double_g_esolve_ga94bd4a63a6dacf523e25ff617719f752.html`, 2017. Online; accessed 16 February 2019.

[15] A. Jović, K. Brkić, and N. Bogunović. A review of feature selection methods with applications. In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2015.