Feature Selection for Hyperspectral Images using Single-Layer Neural Networks

Mateus Habermann, Vincent Frémont, E H Shiguemori

Feature Selection for Hyperspectral Images using Single-Layer Neural Networks

M. Habermann[1,2], **V. Fremont**[1], **E.H. Shiguemori**[2]

[1]Sorbonne Universités, Université de Technologie de Compiègne,
CNRS, Heudiasyc UMR 7253, CS 60319, 60203 Compiègne cedex, France.
[2]Institute for Advanced Studies, Brazilian Air Force, Brazil.

Keywords: Feature Selection, Deep Learning, Overfitting Problem, Hyperspectral Images.

Abstract

Hyperspectral image classification by means of Deep Learning techniques is now a widespread practice. Its success comes from the abstract features learned by the deep architecture that are ultimately well separated in the feature space. The great amount of parameters to be learned requires the training data set to be very large, otherwise the risk of overfitting appears. Alternatively, one can resort to features selection in order to decrease the architecture's number of parameters to be learnt. For that purpose, this work proposes a simple feature selection method, based on single-layer neural networks, which select the most distinguishing features for each class. Then, the data will be classified by a deep neural network. The accuracy results for the testing data are higher for the lower dimensional data set when compared to the full data set, indicating less overfitting for the reduced data. Besides, a metric based on scatter matrices shows that the classes are better separated in the reduced feature space.

1 Introduction

Deep Learning (DL) approaches are undoubtedly a powerful tool for hyperspectral image classification [12][13][7]. Its strength comes from its deep architecture, which enables different levels of features abstraction. Besides, the fact that both the feature extractor and the classifier are embedded in the same architecture renders DL very powerful.

One should, however, be cautious when using a deep architecture. Considering, for instance, a deep neural network (DNN) —which will be the basis of our analysis —, the deeper the architecture, the more free parameters are needed. It is advisable that the ratio between training patterns and free classifier parameters be, at least, 10 [15]. For ratios bellow that threshold, the occurrence of overfitting is more likely.

One way to avoid this problem is to reduce the input data dimensionality. This would cause, as a direct consequence, the decrease on amount of the classifier parameters. Thus, the above-cited ratio would be smaller, when keeping the same training data amount.

There are two ways of reducing the data dimensionality: *i)* Feature extraction, and *ii)* feature selection. The former performs combination amongst features, generating new ones as the Principal Components Analysis does, for example. The latter creates a new feature set by choosing the original features that meet a certain criterion [15].

One positive aspect of feature selection is that it keeps the original information of data [5]. This can be valuable for some applications, especially the ones dealing with hyperspectral images, whose bands depict specific regions of electromagnetic spectra. Maintaining the original physical information provides a better model readability and interpretation [6].

In [8], the authors proceeded to feature selection by using a perceptron neural net with step function. After the training phase of the network, they discarded the features with the smallest interconnection weights. They claim that this approach reduces the processing time, compared to support vector machine-based features selection. It also avoids the evaluation of multiple feature subset combinations, what is common on wrapper approaches. In [2], the author lists some pros and cons of wrapper and filter methods for feature selection. They proposed a filter-based forward selection algorithm, which incorporates some aspects of wrapper method. More precisely, the proposed method uses boosted decision stumps[1]. In a successive processing, the features that correctly predicts the class label that other features could not are selected to be part of the reduced feature set. In [10], an embedded feature selection of hyperspectral bands with boosted decision trees is proposed. This method generated several decision trees, and the features most used by those trees are selected to be part of the reduced feature set.

This paper proposes a novel method for feature selection using a single-layer neural network, aiming at designing a reduced deep architecture less susceptible to overfitting. For each class, the features linked to the biggest weights of the network are selected. It is a filter-based method, which defines automatically the quantity of features to be selected.

The rest of the paper is organized as follows: In Section 2 some basic concepts of feature selection are presented. The proposed method is described in Section 3. The data set and the results are shown in Section 4. Finally, the Conclusion is

[1]A one-level decision tree.

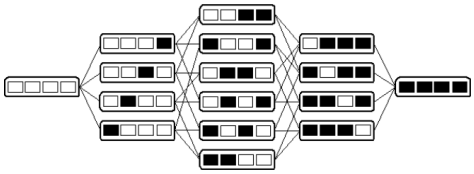

Figure 1. All possible combinations of a 4-feature search. Black boxes indicate the included features, whereas the white ones stand for the features not included [9].

found in Section 5.

2 Feature Selection

2.1 Definition

Feature Selection (FS) is the process of selecting a relevant subset of the original features. Concerning a classification problem, the aim of the chosen subset is twofold: *i)* To reduce the data dimensionality and *ii)* To retain as much information for class separability [15].

More formally, let A be the original set of features, with cardinality $|A| = m$. The feature selection is the assignment of weights w_i to each feature $a_i \in A$, defining their relative importance. Ultimately, the objective of FS is to identify relevant features according to a definition of relevance. Let $F = F_1 \times F_2 \times ... \times F_m$ be the feature space defined by unit vectors F_i, with $1 \leq i \leq m$. One seeks to define the function $f : F \longrightarrow L$, according to its relevant features, where L is the set of labels. Lastly, one definition for relevance could be: *A feature $a_i \in A$ is relevant to a function f if there are two elements α and β in space F with distinct labels such that α and β are distinct only with respect to a_i, and, consequently, $f(\alpha) \neq f(\beta)$* . That is, instances α and β can be distinguished only thanks to a_i [9].

2.2 Characterization

A FS method can be characterized under four aspects: *i)* Search organization; *ii)* Generation of features subset; *iii)* General schemes for feature selection ; and *iv)* Evaluation measure.

2.2.1 Search organization

The proposed algorithm is supposed to conduct the FS process using a specific strategy. Each state S in the search space determines the weights for the features $a_i \in A$. Figure 1 shows an example of feature search space, where each set with four squares is considered as an instance S_i.

In relation to the number of instances, a method can handle at a given instant, three possibilities: *i)* **Exponential search:** It can evaluate more than one state at a time. It achieves the optimal solution due to the exhaustive search. Sometimes, however, if the evaluation measure is monotonic, not all the possible combinations need be visited. In this case, a Branch and Bound algorithm [11] may be employed [15]. *ii)* **Sequential search:** This method selects one amongst all the candidates to the current state. It is an iterative procedure and, once one state is chosen, it is not possible to go back. *iii)* **Random search:** The intention is to use the randomness in order to avoid getting trapped in local minimum solution, and also to allow some movements towards states with worse results.

2.2.2 Generation of features subset

The output of a FS algorithm is the subset $A' \subset A$, containing the chosen features.

One can start with the original set A with m elements, and, at each step, one feature is dropped out from A until the desired number of features is achieved. This method is called **sequential backward selection**.

Another method, which is the reverse of the preceding procedure, is called **sequential forward selection**. It starts from an empty set, and the best feature —according to a specified criterion—is added to the set after each step.

The two afore-mentioned methods suffer from the *nesting effect*. That is, once a feature is discarded or chosen, depending on the method, it cannot be undone [15]. So, in **compound method** the idea is to use both forward and backward methods.

2.2.3 General schemes for features selection

The relationship between the FS method and the subsequent classifier can normally have two forms : *i)* Wrapper; and *ii)* Filter [15].

Wrapper : In this scheme, the FS method is used during the training phase of the classifier. For each feature a_i added to A' or discarded from it, the classifier should be trained again in order to assess the features subset A'. Thus, the main disadvantage is the heavy computational cost. Its advantage is the good overall classifier's accuracy [14] [9].

Filter: The FS method is used as a data preprocessing step. In this case, the feature selection method is independent of the classifier. The advantage of this scheme is its speed in relation to wrapper method. Its disadvantage is that the feature selection is not conducted by the classifier, yielding suboptimal results [14] [9].

2.2.4 Evaluation measures

There are some ways to assess how good a feature subset A' is. Most evaluation measures such as *Divergence* and *Chernoff Bound and Bhattacharyya Distance* take into account the probability distribution of the classes [9]. However, they are not easily computed.

Thus, in this work we will adopt a non-parametric measure called *Scatter Matrices* [15].

Scatter Matrices: Scatter Matrices are related to the way the samples are scattered in the feature space. Thus, the following three matrices should be defined:

Within-class scatter matrix:

$$M_w = \sum_{i=1}^{q} P_i \Sigma_i,$$

where q is the quantity of classes, Σ_i stands for the covariance matrix for class L_i, and P_i is the *a priori* probability of class L_i.

Between-class scatter matrix:

$$M_b = \sum_{i=1}^{q} P_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T,$$

where μ_i is the mean vector of class L_i, and μ_0 is the mean vector of the whole data.

Mixture scatter matrix: it can be expressed as

$$M_m = M_w + M_b.$$

The trace of M_m is the sum of variances of the features around their respective global mean. This way, the metric

$$J = \frac{trace(M_m)}{trace(M_w)}$$

has large values when the data are well clustered inside their respective classes, and the clusters of different classes are well separated [15].

Finally, J will be used in this work to assess the validity of a subset of selected features.

3 Proposed method

The FS method proposed in this paper can be categorized as *sequential forward selection*, with *sequential search*, and the evaluation criterion used for assessing it is *scatter matrix*. It is also a *filter method*, meaning that the features are selected before the employment of the classification algorithm.

The objective is to find a features subset $A' \subset A$, which will be used by the subsequent classifier. At first, A' is empty. After each iteration of the method, chosen features are added into A' until a stopping criterion is met.

3.1 Stopping criterion

In many approaches, the stopping criterion is user-defined. Normally, it is associated with the quantity of features the algorithms should find. For experienced users, it would pose no problem at all, however it is not always the case. Thus, in order to avoid such situations, the proposed method defines automatically the number of features to be retained.

Keeping in mind that the classifier to be used is a deep neural network (DNN), it is possible to define a deep architecture with the following restrictions: *i)* The amount of its learning parameters should be, at least, ten times smaller than the training data cardinality [15], in order to decrease the chances of having overfitting; and *ii)* the architecture should take a funnel-like disposition —to follow the encoding process of stacked autoencoders [16]. Therefore, knowing the output layer size and the number of hidden layers[2], one can easily come up with an architecture. One example of architecture is displayed in Figure 2.

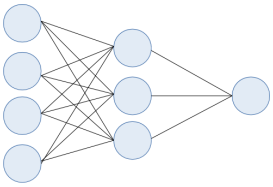

Figure 2. A funnel-shaped neural net, with one neuron in the output layer.

Let l be the quantity of hidden layers of the classifier, d the input layer size, and o the output layer size. The architecture used by the networks of this paper is defined according to Algorithm 1.

Algorithm 1 Funnel-shaped deep neural network architecture.

1: input : (d, o, l)
2: **input layer:** d neurons
3: j^{th} **hidden layer:** $d - j * r$ neurons, with $r = (d-o)/(l+1)$
4: **output layer:** o neurons
5: return: Neural net architecture

When taking into account the whole data without feature selection, the input layer size is $d = |A|$. After the feature selection process, $d = |A'|$.

By using the method described in Algorithm 2, one can determine the biggest —in terms of neurons quantity —architecture γ whose amount of parameters is less than $\frac{|X|}{10}$, where $|X|$ is the cardinality of the training data X.

Algorithm 2 Creation of an architecture with limited number of parameters.

1: $k = |A| + 1$
2: **do**
3: $\quad k = k - 1$
4: $\quad$ Create architecture γ_k according to Algorithm 1, using as input (k, o, l)
5: $\quad quantity\_parameters =$ amount of parameters of γ_k
6: **while** $quantity\_parameters > \frac{|X|}{10}$
7: return: γ_k

Finally, the input layer size of the architecture γ is the quantity of features to be selected by the proposed FS method. In reality, this quantity is the *stopping criterion*.

3.2 Feature selection

Partially based on the concept of Boosting [3], the proposed method gives the partitioning of the training data X into several subsets X'_i of equal cardinality. This approach was already used in [10], but in our work the novelty is the use of single-layer neural networks for feature selection.

[2] A user-defined hyperparameter.

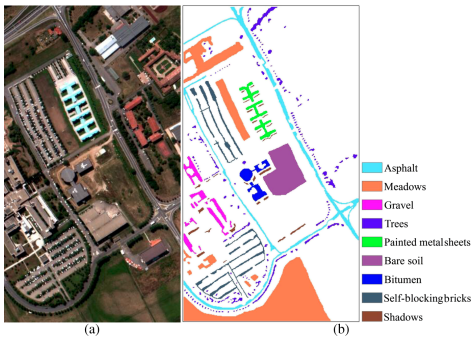

Figure 3. Pavia University image. (a) Pavia university. (b) Ground-truth map [4].

For the first subset X'_1, a single-layer neural network performs a binary classification using all the original features of the set A. The output layer has only one neuron, and its activation function is the sigmoid. After the training, the two features associated to the biggest and the smallest weights are added to the set A', which was initially empty. In the sequel, another single-layer neural network will perform a binary classification on the subset X'_2, but this time the features set will lack the two features previously chosen. This process repeats until the the cardinality of A' equals the input layer size of the reduced architecture γ calculated in Section 3.1.

In a classification problem with q classes, the method should be run q times. This way, it is possible to select the features that are more appropriate for each class.

4 Experiments

This section will show some results of the proposed method. For the training of the deep architectures used in this work, 20000 training epochs were used for both architectures. Theano library has been employed [1]. The data are composed of hyperspectral images.

4.1 Data set

The data used in this work are composed of two hyperspectral images. They were acquired by the ROSIS sensor, over Pavia, Italy. Pavia University is an image with 1096×1096 pixels, 103 spectral bands and 1.3m of spatial resolution. It is used as *training data*. Figure 3 shows Pavia University image and its ground-truth map.

The *testing data* is the Pavia Centre image. With 102 spectral bands, 610×610 pixels and 1.3m of spatial resolution. Figure 4 shows the image and its ground-truth.

According to Figures 3 and 4, each image has 9 classes. From those, 7 are present in both images. And among those 7, a total of 4 classes are considered here: *bare soil*, *meadows*, *self-blocking bricks* and *trees*. This choice was based on similarities of the class spectral signatures between training and testing data. Table 1 shows the classes and their respective cardinality. Both training and testing data are of the same size.

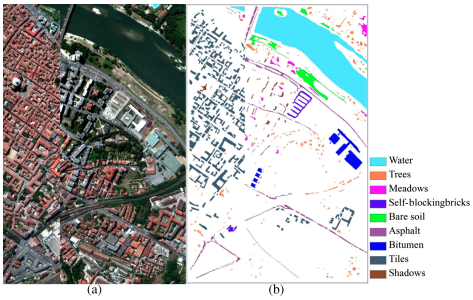

Figure 4. Pavia Centre image. (a) Pavia centre. (b) Ground-truth map [4].

Table 1. Classes and their cardinalities.

class	number of elements
bare soil	2863
meadows	3500
self-blocking bricks	2685
trees	3064
total	**12112**

4.2 Results

The main objective is the selection of features that will be used by a classifier, as shown in Figur 5.

Here, there are 4 classes to be classified, thus the proposed method should be run four times, as prescribed in Section 3.2.

At each execution, the algorithm selects the features for a class in particular. At the end, all the selected features compose the feature subset A', which will be used by the classifier.

4.2.1 Feature selection for each class

According to Table 1, there are in total 12112 instances for training the deep neural network. Thus, in order to be consistent with our pursuit of avoiding overfitting, the reduced architecture should have, at most, one-tenth of that amount as parameters quantity, *i.e.*, 1211 weights and biases. By using Algorithm 2, the calculated architecture $\gamma_{reduced}$ is $24 : 20 : 16 : 12 : 8 : 4 : 4$, where 24 is the quantity of neurons of the input layer. Thus, the amount of features to be selected is 24. For each one of the 4 classes its correspondent feature selection is tackled as a binary problem. Each class is supposed to contribute equally to the selected features set A'. Therefore, each class will provide A' with 6 features. At each run of the FS method, 2 features are selected. Thus, 3 executions for each class will be necessary.

Bare soil: For the *bare soil* class, the single-layer neural network described in Section 3.2 is trained to classify the fol-

Figure 5. Flowchart for the proposed method.

Table 2. Overall results for both architectures.

architecture	training acc.	testing acc.	J
γ_{full}	96.15%	51.57%	1.29
$\gamma_{reduced}$	95.84%	53.19%	1.36

lowing binary problem: *bare soil*, with 2863 instances, and *non-bare soil*, with 9249 instances. The selected features —or bands—for *bare soil* are 73, 81, 82, 83, 84 and 85.

Meadows: For this class, the same procedure used for *bare soil* class is adopted. The resulting selected features are: 40, 65, 68, 88, 96 and 102.

Self-blocking bricks: For the *self-blocking bricks* class, the selected features are: 28, 32, 55, 77, 87 and 100.

Trees: For *trees*, the selected features are: 23, 24, 26, 78, 79 and 80.

4.2.2 Reduced and full architectures results

In order to show the validity of our method, it is necessary to compare results between the full and reduced architectures. It is expected that the full architecture achieve a better accuracy in training data and worse accuracy in testing data, when compared to the reduced architecture.

The full architecture γ_{full} takes the original data set without reduction, that is, with 102 spectral bands. Its output layer size is 4 and in this work the number of hidden layers is always 5. According to Algorithm 1, the architecture of γ_{full} is $102 : 85 : 68 : 51 : 34 : 17 : 4$, where 102 and 4 are, respectively, the amount of input and output neurons.

Table 2 shows the overall results for both architectures.

Considering the full architecture γ_{full}, its accuracy for the training data was 96.15%. The accuracy for the testing data was 51.57%.

As for the reduced architecture $\gamma_{reduced}$, the accuracy for the training data was 95.84%, and the accuracy for the testing data was 53.19%.

Negative remarks: *i)* In both cases, the accuracy for the testing data was not good. There is a considerable difference between it and the accuracy for the training data. Maybe other architectures with different quantity of hidden layers and neurons may yield better results, however it is not the main concern of this paper. *ii)* The selected features, or bands, shown in Section 4.2.1 are, in many cases, contiguous. For a hyperspectral image, it means that they are strongly correlated, and strong correlation amongst selected features should be avoided by a FS method. According to the proposed method, the feature selection for any two classes L_i and L_j, with $i \neq j$, is independent, therefore, the inclusion of a same feature a_k in both A'_i and A'_j is perfectly possible, but undesirable. One way to solve this issue could the inclusion of one processing step after the feature selection operation, in order to discard repetitive and strongly correlated features. For this, our *stopping criterion* should be increased, permitting the selection of more features.

Positive remarks: *i)* Concerning the training data, the accuracy for γ_{full} was bigger than that for $\gamma_{reduced}$. In fact, it is expected, because the model with more parameters can *learn* finer details of the training data. However, getting excessively good at predicting the training data may degrade the prediction with other data sets, that is, its generalization power. In our experiments, it is very clear. Contrary to the training data, the testing data accuracy for the $\gamma_{reduced}$ architecture was better than that of the full architecture γ_{full}. The reduced architecture $\gamma_{reduced}$ uses less information from the data set than γ_{full}, and yet the former achieves better accuracy for the testing data. From this, we may conclude that there was less overfitting for the reduced architecture $\gamma_{reduced}$. *ii)* The metric J for $\gamma_{reduced}$ is bigger than that for γ_{full}, indicating that the classes are better separated in the reduced feature space F' induced by A'. Therefore, the proposed method is valid.

5 Conclusion

Deep Learning architectures have a large number of parameters to be learned. This fact *per se* represents no problem at all. The source of complications is the scarceness of the training data set, in terms of not possessing enough instances, creating a scenario where overfitting may arise.

One possible way to avoid this situation is decreasing the number of classifier's parameters. In a deep neural network, it may be achieved by decreasing the input layer size. Thus, feature selection methods enter the scene.

This work proposed a feature selection method based on single-layer neural networks. The selected features are the ones associated with the biggest network's absolute weights. Each class gives its contribution in order to compose the final subset of selected features. After the feature selection, the data were classified by two funnel-shaped deep neural networks. The reduced architecture, having as input data the selected features, achieved a better accuracy with the testing data when compared to the full architecture. Bearing in mind that in the reduced feature space the classes are better separated, this difference in performance may be credited to the presence of overfitting in the bigger architecture, whose accuracy for the training data was higher than that of the reduced architecture.

The problem of selecting correlated features is an issue that shall be addressed in further works. This way, it is possible to avoid the choice of bands bearing nearly the same information.

For the time being, the results indicate the validity of the proposed method.

Acknowledgements

This work was carried out in the framework of the Labex MS2T and DIVINA challenge team, which were funded by the French Government, through the program Investments for the Future managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

We are also thankful for the support provided by Brazilian Air Force and Institute for Advanced Studies (IEAv).

References

[1] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[2] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 74–81, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[3] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[4] Jianwei Gao, Qian Du, Lianru Gao, Xu Sun, and Bing Zhang. Ant colony optimization-based supervised and unsupervised band selections for hyperspectral urban data classification. *Journal of Applied Remote Sensing*, 8(1):085094, 2014.

[5] S. Khalid, T. Khalil, and S. Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*, pages 372–378, Aug 2014.

[6] J. Li and H. Liu. Challenges of feature selection for big data analytics. *IEEE Intelligent Systems*, 32(2):9–15, Mar 2017.

[7] P. Liu, H. Zhang, and K. B. Eom. Active deep learning for classification of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(2):712–724, Feb 2017.

[8] Arroyo G Meja-Lavalle M, Sucar E. Feature selection with a perceptron neural net. In *In: Proceedings of the international workshop on feature selection for data mining, pp 131135*, 2006.

[9] L. C. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: a survey and experimental evaluation. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 306–313, 2002.

[10] S. T. Monteiro and R. J. Murphy. Embedded feature selection of hyperspectral bands with boosted decision trees. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, pages 2361–2364, July 2011.

[11] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.*, 26(9):917–922, September 1977.

[12] B. Pan, Z. Shi, and X. Xu. R-vcanet: A new deep-learning-based hyperspectral image classification method. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(5):1975–1986, May 2017.

[13] B. Pan, Z. Shi, and X. Xu. R-vcanet: A new deep-learning-based hyperspectral image classification method. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(5):1975–1986, May 2017.

[14] A. H. Shahana and V. Preeja. Survey on feature subset selection for high dimensional data. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–4, March 2016.

[15] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, 4th edition, 2008.

[16] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.