# Classifying systems on the AVIRIS dataset

**Manuel Montoya Catalá**
Master in Multimedia and Communications
Carlos III of Madrid University
Spain, PA 2189
*mmontoya@ing.uc3m.es*

## Abstract

This paper analyses and compares the performance of several classifier systems for the AVIRIS image Indian Pine dataset. It preprocesses the data getting rid of noisy bands and meaningless samples and then tests several classier techniques such as LR, LDA, GNB, KNN, RF and SVM using crossvalidation for the tuning of its parameters if any. Also, ensemble techniques such as bagging and boosting are used.

## 1 Introduction

In this paper several a study of several classification schemes is developed for the AVIRIS image Indian Pine dataset [1][2]. The dataset is composed of the spectral sampling of a landscape corresponding to a 145x145 image over 220 different frequency bands. Our goal is to know which kind of terrain contains the image at every pixel. For doing so, preprocessing of the data will be performed and several classifier techniques will be tested in order to find the model that best fits the data.

## 2 Data preprocessing

The first step of almost every machine learning system is to preprocess the data. Preprocessing is used to transform the initial dataset so that it fulfills the requirements of the learning system and from which is easier to extract information, thus improving the overall performance of the system.

The AVIRIS image Indian Pine dataset is composed by the sampling of an optical sensor that delivers calibrated images of the upwelling spectral radiance in 220 contiguous spectral channels (bands). The aim of the AVIRIS dataset is to identify, measure, and monitor constituents of the Earth's surface and atmosphere based on molecular absorption and particle scattering signatures at different frequency bands. The dataset contains the value of the 220 contiguous channels for a single 145x145 image, each pixel of the image is a sample vector X, and each band is a feature, we end up with a dataset containing 21025x220.

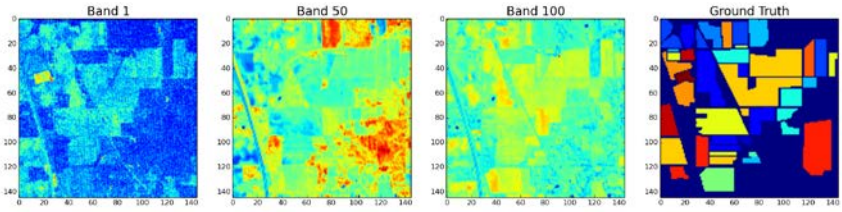


Figure 1: Spectral features examples and groundtruth.

There are particular preprocessing steps concerning this database, some of the bands are known to be noisy and are therefore removed so that the system performance is not impoverished by them. Also, the background class is excluded so that the system focuses on the different important terrains such as wheat, woods or stone. Then we apply the standard preprocessing steps of a machine learning system with numerical features, we split the database into subsets, one for training (20%) and the other for testing (80%) and normalize the dataset, transforming each feature to zero mean and standard deviation 1, using the training set as the source of the normalization.

# 3 Simple Classifiers

As a first approach to the problem, three basic non-parametric multiclass classifiers have been implemented using the sklearn library in Python [4]. The image below shows the accuracy results obtained for each of the classifiers Logistic Regression, Linear Discriminant Analysis and Gaussian Naïve Bayes. Quadrant Discriminant Analysis could not be carried out because of the lack of samples of some classes (class 8 has only 4 training samples).

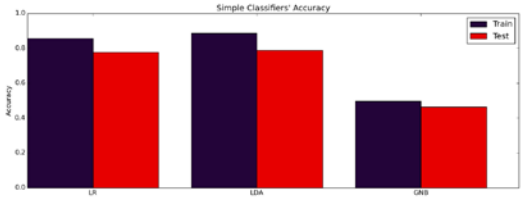


Figure 2: Simple Classifiers' Accuracy

Some of these classifiers make several assumptions, LR's maximizes a probability function which is equivalent to the one obtained if all classes are multivariate Gaussians with the same covariance matrix, LDA also makes this assumption and maximizes the separation among classes, both methods obtain very similar classification results of about 77% accuracy on testing which means the model fits the dataset nice enough, LDA makes a stronger assumption and therefore has more overfitting. GNB makes the assumption that all features have Gaussian distribution and are independent from each other; since it has poor accuracy, around 50%, we can conclude that the dataset does not have that independency property.

# 4 Parametric Classifiers

Parametric classifiers are those that have a set of parameters $\bar{\theta}$ affecting the classification properties of the learner. These parameters need to be tuned, we have to find their optimal value in order to obtain the optimal system and avoid overfitting. A simple approach for validating a parameter is to subdivide the training set in two sets, one for the actual training, and the other for validating the parameter; this second set acts as a test set in training time to measure the generalization of the selected hyperparameters.

A more refined technique called "crossvalidation" is usually used for selecting this hyperparameters, it consist in subdividing the training set in the data set in K subsets without replacement and successively train K models with all the subsets but on which is used to validate. In this paper crossvalidation with $K = 5$ has been used to tune the different classifiers. The parametric classifiers used are the KNN, SVM and RF.

## 4.1 K-Nearest Neighbors

The KNN classifier labels an input sample $\bar{X}^*$ according to the following procedure: first, it finds the K nearest training samples to $\bar{X}^*$, then it labels $\bar{X}^*$ as the majority vote class of these K closest neighbors. For validating the K parameter, a 5-fold have been performed, the figure below shows the training, validation and test error for different values of K.

As we can see in Figure 3(a), the training error for K = 1 is 0 since the nearest neighbor of the training set is itself, it will always be assigned its very same label. For small values of K,

the system makes local overfitting and for big values of K, the system chooses the majority class in the training dataset, validation finds K = 4 as the best value for this tradeoff. The accuracy for testing is about 73%, if more training samples were used, this method would obtain better results.

### 4.2 Support Vector Machine

The SVM is a binary classifier that aims to find the hyperplane over the feature space that maximizes the margin (separation between classes). If the classes are not linearly separable, we can allow some outliers by introducing penalization term C, any sample can be virtually moved in the feature space in order to make the problem linearly separable, the distance moved is multiplied by C and acts as a regularization term. If C is too small, any sample could be moved anywhere with no cost, and the system will not adjust to the real dataset, if C is too big the system will be too regularized and won´t find the optimal solution.

We can also transform the input feature space into another one that is more linearly separable by applying a transformation to them such as a polynomic extension or a Gaussian kernel. These kernel extensions allow using SVM in non-linearly separable datasets but can also cause overfitting. They also have hyperparameters; the polynomic extension needs to know the degree of the polynomial and the Gaussian kernel, the gamma parameter.

Figure 3(b) shows the validation accuracy for different degrees of the polynomic extension and C values. We can see the behavior previously described for the different values of C, also, as we increase the degree of the extension, the validation accuracy drops due to increasing overfitting. The testing error for the different kernels is about 82% for the linear, 80% for the polynomic extension of degree 2 and 85% for the Gaussian kernel. The training accuracy for all of them is about 95%, meaning that the polynomic kernel is the one that causes more overfitting, the C parameter is different for every kernel.

### 4.2 Decision Trees and Random Forest.

A decision tree is a classifier expressed as a recursive partition of the instance space. Starting from the root node, the classifier divides the feature space sequentially at every internal node based on a function of the features and a criteria to maximize (entropy, accuracy…). Our tree considers a single feature at each node for the partition of the space, thus creating hypercube decision boundaries. There are several hyperparameters to tune for every kind of tree; we tuned by crossvalidation the maximum depth of the tree and the criteria to optimize. This classifier tends to overfit due to the hard decision boundaries, Random Forest is an extension that generates a set of different trees by training them with random subsets of samples and featues and the combining the outputs, thus increasing the diversity, reducing the bias and overfitting.

Figure 3(c) shows the accuracy of the validated decision tree and its randomized extensions. The accuracy is similar to the one obtained by LR and LDA. Since our tree generates hypercube decision boundaries, it can be concluded that a single tree is not a good approach but a combination of them in the Random Forest softens the boundary increasing the test accuracy. Since training accuracy reaches 100%, but resting accuracy only reached 80%, we can see that even the combination of randomized trees has a lot of overfitting.

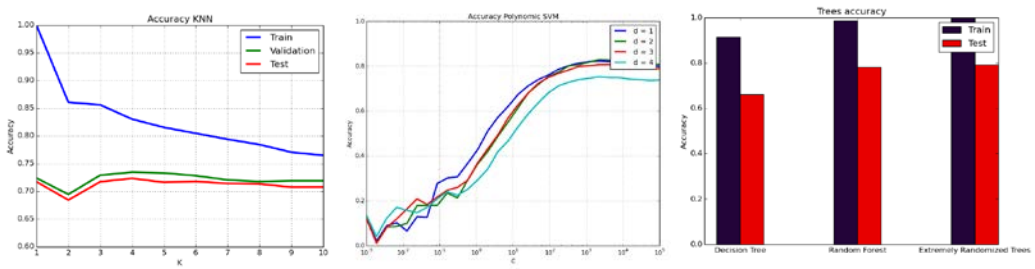


Figure 3: Validation of KNN, SVM and Trees

# 5 Ensembles

An ensemble system combines a set of weak learners to build a strong one, exploiting the diversity among these base learners. There are different approaches depending on how the learners are generated, trained and combined.

## 5.1 Bagging

Bagging methods form a class of algorithms which generate several independent instances of a base learner by training them with random subsets of the original training set and then combine their individual predictions. These methods are used as a way to reduce the variance of a base learner and avoid overfitting by introducing randomization in the training sets. The individual predictions are usually combined by averaging the individual outputs in the case of a soft output or by means of majority vote if the outputs are hard.

Figure 4 shows the accuracy of the bagging ensembles with 50 baggers and 80% of samples for different classifiers. The accuracy varies between $[-2,2]\%$ with respect to the base learners. Bigger number of baggers did not improve the results and lower number of samples impoverished the results. Bagging hasn`t improved a lot the system but at least it reduces the variance of the output of the pseudo-randomly trained classifiers.

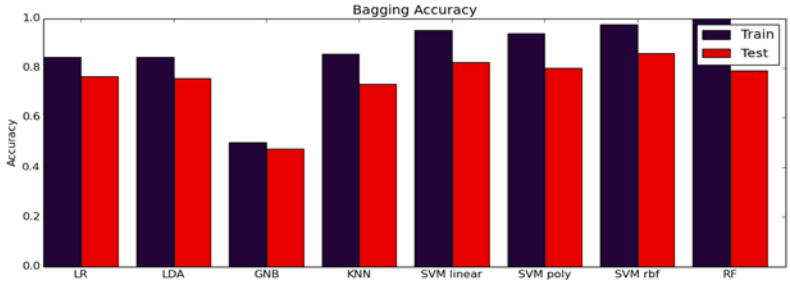


Figure 4: Accuracy of Bagging Ensembles

## 5.2 Boosting

Boosting is an ensemble technique used for creating a strong classifier as a linear combination of weak classifiers. Given a set of T weak classifiers $h_t(\cdot)$, $t = 1,2 \dots T$, the output of the system is obtained as a linear combination of the individual outputs. These weak classifiers $h_t(\cdot)$ are trained in a sequential manner, each weak learner focuses on a different region of the input space; this is achieved by using a weight vector $\overline{D}$ during the training phase. This $\overline{D}$ is a discrete distribution vector over the training samples space that tells the weak learner how much importance it has to give to every training sample. Samples that are poorly classified will have big weight values so the weak learner will focus on them

Boosting does not work well for the multiclass problem and the implementation of sklearn is not accurate so the results are not taken into account for this paper.

# 6 Conclusions and Future work

When working with a finite dataset, the better the dataset fits the model, the higher the accuracy, in this case, the classes seem to fit a multivariate Gaussian due to the good results obtained by the LR and LDA classifier and features are correlated due to the poorer results in the GNB. The SVM with Gaussian kernel is the best approach to the dataset, obtaining 86% of accuracy on test, this is due to the fact that SVM makes no assumptions about the data and maximizes margin. Random Forest also fits the data but causes high overtitting.

Future Work to be highlighted over this dataset is performing computer vision techniques to exploit the spatial relationship of the samples in the image and perform feature extraction and selection over the initial dataset.

## References


[1] C4.278.12996-1 MACHINE LEARNING APPLICATIONS 14/15-S2.

[2] AVIRIS dataset. http://aviris.jpl.nasa.gov/index.html

[3] Advanced Introduction to Machine Learning. 10715, Fall 2014. Eric Xing, Barnabas Poczos. School of Computer Science, Carnegie-Mellon University.

[4] scikit-learn library documentation http://scikit-learn.org/dev/index.html

[5] Chapter 9 DECISION TREES. Lior Rokach. Department of Industrial Engineering Tel-Aviv University