Chapter 3

Object Recognition Using Local Features

Object recognition has been one of the most active areas of computer vision and pattern recognition. But, despite this, it is widely known that there are several drawbacks when implementing the most famous recognition techniques because they can not deal with the large variety of possible object configurations and background situations. Under the typical scenario of object recognition, this chapter introduces some novelties that somehow improve previous classical techniques. A standard object recognition scheme is shown in figure (3.1). It must be said that this scheme is very elemental and most of the techniques presented in the previous chapter can easily be described in terms of the steps of this scheme.

Data Preprocessing: We start assuming that our objects are described in terms of pixels. These set of pixels in an image have some special meaning for us because we previously assume that some of them belong to an object. In contrast to these pixels, the remaining ones in the image belong to background and we are not interested in them. The data preprocessing step consists of all those operations that we can do with the image pixels that help us to extract information from the scene relevant for object recognition. As examples, we can delete image pixels corresponding to the background if we previously know that these pixels obey to some specific rule (i.e. black background). Also, in this step we can include all those operations that we can perform in an image in order to extract relevant and interesting points (keypoints) or some relevant information that can help us in subsequent steps.

Feature extraction/selection: In the previous chapter we reviewed several features that can be extracted from images in order to obtain relevant object representations. At this step, we will select an appropiate set of features depending on the images we have since they can suffer from a large variety of variations (illumination changes, occlusions, changes in object shape, etc). Once we selected an appropiate set of local features to apply to our objects, we will use the preprocessed image from the previous step as the starting point of this step.

Data preprocessing and feature extraction are two related steps that need to be thought at the same time since some features can only be extracted if our data is correctly preprocessed. As a result of this step, we generate a set of data vectors (also said feature vectors) for the next step. These data vectors contain all the data needed for further analysis and, in theory, they have to contain enough information to identify our objects.

Classifier Training: Given a set of data vectors (feature vectors) extracted from images containing the objects we want to identify, this step consists of obtaining a classifier that best characterizes our objects. Of course, depending on the features used to extract information we would need a different classifier. This classifier can be based on the relationship of data vectors, can be based on probability density functions that best describe our data or can be based on more complex criteria. Training model directly depend on the classification scheme. For example, if we use a k-nn (nearest neighbor) to classify new vectors we do not need to build a training classifier because we will only need to compare a testing vector with the whole set of training vectors without building a complex training model.

Testing: Once a training classifier is built from training data, the obtained classifier should be tested with new unseen data vectors in order to verify that it has learnt our objects. The point is to find a classifier that is able to recognize the same objects used in the learning step and, at the same time, a classifier that also deals with new unseen instances of the same objects. There are different ways to test a training classifier, but the most common way is to perform an analysis with unseen data which generates a set of the recognition rates. After this, we will take these recognition rates to produce a decision about the classifier and we will decide its acceptance or not.

One of the main problems of this scheme is that once we have extracted a relevant set of data vectors using a specific set of descriptors, we usually have a high-dimensional representation. That is, our data vectors are represented using a lot of components. Is it good or bad? In terms of object recognition, the more data we have about the objects, the easier it will be to distinguish them from others. But, in terms of computational time and resources, the more data we have about our objects, more complex models will be created. Here is where we have to find a tradeoff between the amount of data and computational requirements. In object recognition, we simply need the simplest model that guarantees a minimal recognition performance that is established a priori.

This high dimensional scheme could produce the well-known problem of *curse of dimensionality* [11]. The properties of high dimensional spaces often appear counterintuitive because of our experience with the physical world in a low-dimensional space. The main problem is that data representations in high dimensional spaces have a larger amount of surface area for a given volume than data representations in low-dimensional spaces [28]. In our particular scheme, the complexity of working with a high dimensional space can be alleviated if we can reduce the dimensionality of the problem. For the class of large problems where the benefits/efficiency of a vector

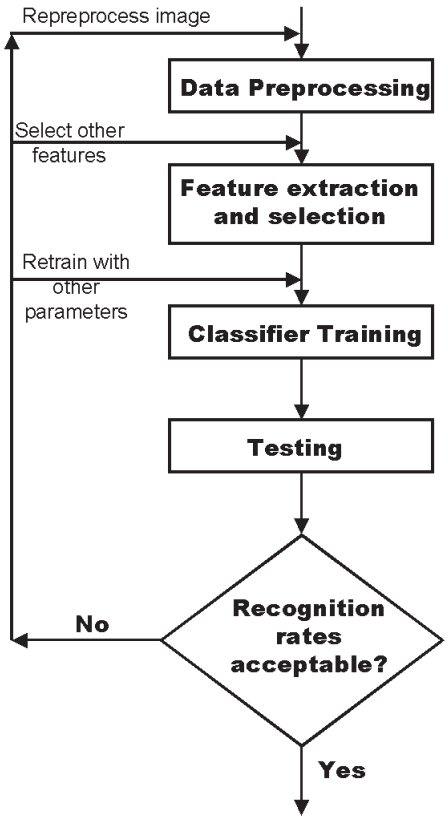


Figure 3.1: A standard object recognition scheme.

space model becomes critical, we often desire some form of compression (compaction) of the original data. Compression is advisable for two reasons. First, we wish to reduce the amount of physical disk space required to store data. Second, and often both more important and more difficult to implement, the memory required in performing computations on the original (uncompressed) data set prevent us from being able to explore the data. Furthermore, by reducing the dimensionality we are able to partially solve the *curse of dimensionality*.

One of the main aims of this chapter is the analysis of those techniques that can be used as linear transformations of data. Here, we want to analyze how a given set of data vectors can be transformed into another set of data vectors without losing a huge amount of information and improving the performance of the initial representation. Also, since all the data vectors that we will use are defined in a high dimensional space, we are interested in those linear transformations of data that can also reduce the dimensionality of the problem and alleviate the *curse of dimensionality* problem.

Unsupervised learning algorithms such as those we will study in this chapter are helpful for modeling low dimensional pattern structures present in high dimensional data. Linear transformations account for most of the feature extraction performed

in practice. Examples of spread out multivariate linear transformations are the Discrete Fourier transform, signal convolutions (linear filters), wavelets, Principal Component Analysis, Factor Analysis, Independent Component Analysis, Non-negative Matrix Factorization, Canonical Correlation Analysis, Linear Discriminant Analysis, etc. There are many reasons to restrict ourselves to linear mappings, mainly their simplicity. Along this thesis we will restrict ourselves to linear transformations for features extraction and/or representation, focusing on the Principal Component Analysis (PCA), Non-negative Matrix Factorization (NMF) and Independent Component Analysis (ICA).

This chapter and the whole thesis are totally focused on the recognition of objects using local features. Due to this fact, we briefly explain the reasons of this choice with a practical example which demonstrates that local data representations perform better than global ones without using additional information of objects. Then, we introduce the first unsupervised learning technique, the Principal Component Analysis (PCA). Principal Component Analysis is a widely used technique mainly chosen for reducing dimensionality of arbitrary data. Since PCA provides a reduced space, object recognition techniques can be applied on such a subspace and the complexity of high-dimensional spaces is reduced. It turns out that there are other techniques besides PCA to obtain reduced spaces. One example is the Non-negative Matrix Factorization (NMF) that is also introduced in this chapter. NMF pursues the same objectives as PCA, but assuming that the input data is positive defined. NMF has been introduced in the computer vision community as a technique that can extract positive descriptions of data but there are not studies showing what is the performance of this technique with respect to PCA. Under this new point of view, we tested the abilities of NMF to represent local color histograms and other positive representations. We also introduced a new technique called Weighted non-Negative Matrix Factorization (WNMF) which performs better than NMF in certain circumstances. Different object databases and case studies have been evaluated using PCA, NMF and WNMF. Each of these three linear algorithms are based on a set of particular assumptions. PCA assumes a Gaussian distribution of the original data, NMF and WNMF assume a Poisson distribution. So that, we explore these different sources of information in order to find different ways to combine these techniques. Also, we compare different classification frameworks to be used with them: (i) Reconstruction distances, (ii) a parametric model and (iii) a non-parametric model (k-nn). These classification techniques are all valid for recognition purposes. In the non-parametric classification framework, we used the k nearest neighbor technique [44]. So that, we also evaluated these algorithms with different metric distances in order to find the best combination of method and metric distance.

PCA, NMF and WNMF are three unsupervised algorithms which are based on linear transformations of data. They obtain subspaces which describe the original data and each technique assumes its own assumptions. Independent Component Analysis (ICA) is also presented in this framework because it is directly based on a linear transformation of data. However, ICA produces a statistical independent space which can be used to estimate probability density functions. We present ICA as a technique to estimate probability density functions in reduced subspaces which are previously obtained using PCA. Finally, since object recognition is performed with

sets of object classes, we present a new framework using ICA for such a problem. We extend ICA to work with different object classes. This extension is called: Class-Conditional Independent Component Analysis (CC-ICA). Then, an example using ICA to estimate probability density functions to classify between different object classes is presented.

3.1 Example with Local Features

As mentioned in chapter 2 several approaches have recently proposed the use of local window representations as a reliable solution to occlusions, complex backgrounds, scale changes, illumination changes, and different viewpoints or orientations [37, 110, 121, 123] within the problem of object recognition. When lighting conditions do not change severely, or color normalization methods can be used, the color distribution of an object is a simple kind of sensor data that has demonstrated to be an efficient signature for object-recognition in the appearance-based framework (see section (2.1.3)).

Data histograms are the most used positive representations in the computer vision literature. One can find histograms representing color [56, 137] or multidimensional receptive fields [120] or a mixture of different features [87]. Thus, it seems clear that histograms are relevant signatures for object recognition. In this section, we want to experimentally show the advantages of using a local approach with respect to a global one using local color histograms. It is clear that taking global histograms of objects, we are able to perform a reliable object classification if objects are not affected by ambient conditions or do not have occlusions but we will see that a local approach is even better.

Our database is composed of a set of 100 pharmaceutical products taken under 6 arbitrary positions. Figure (C.1) shows the entire object database. It can be seen that some of pharmaceutical products are very similar in terms of visual appearance because they have the same color tonalities. This fact explains an important number of misclassifications when we perform object recognition and classification with these products.

Part of this work is based on extracting perceptually relevant information of objects. In this sense, a common strategy to recognize an object in an image is to locate points where is probable to find discriminant features. For example, the human face consists of regions such as the eyes, nose, etc. These features assure a low misclassification rate. Such features can be selected by the system designer or can be chosen automatically. In our case, we are interested in detecting generic salient object features where to analyze local color distributions.

As noted in section (2.2), we used the Harris keypoint detector in our framework in order to find perceptually interesting points of our objects. Figure (3.2) shows all the interesting points (shown as grayvalues) of a pharmaceutical product which is also taken in other different poses. In this case, we detect 4538 local interesting points whose eigenvalues are greater than a predefined threshold (see section (2.2)). By extracting the local maxima of the eigenvalues (see section 2.2), the number of keypoints is reduced significantly to 22. These local maxima are shown in figure

(3.2.b) as white crosses. The other instances of the same product, figure (3.2.c) to (3.2.f) show that the local maxima are nearly the same.

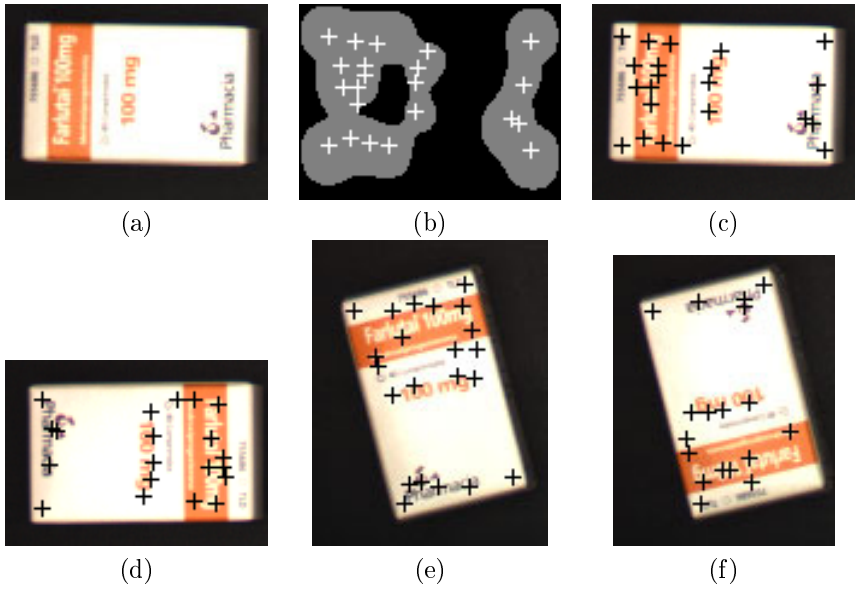


Figure 3.2: (a) A sample object image. (b) Gray values represent all the interesting points. White crosses represent local maxima. (c,d,e,f) Different instances of the same product of (a) taken in different poses and their interesting points.

Once we have selected a set of keypoints we extract the salient features, in our case local color histograms. Since we find desirable to be able to capture the objects at any rotation angle, the extracted local color histograms should be invariant to this. Extracting the histogram from a circular mask in the neighborhood of a keypoint minimizes the rotational effects that the image can suffer. Figure (3.3) shows which region is taken, given a set of detected keypoints.

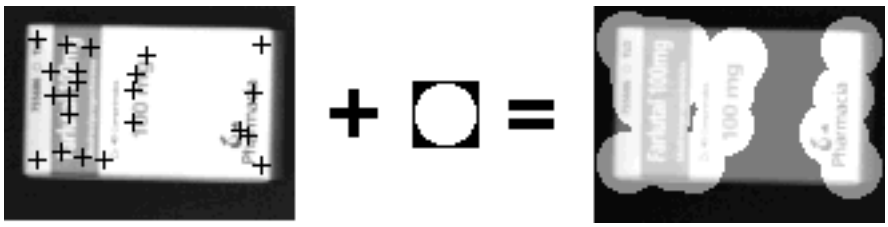


Figure 3.3: Given a set of detected keypoints, a circular region around each keypoint is used to extract its local color histogram representation achieving rotational invariance. It can be seen that the non homogeneous regions are used to identify an object.

For this specific application we reduced the number of local color histograms of objects because not all of them are really necessary. The Harris method assures that

all the local color histograms will be extracted from an image and these histograms will correspond to non homogeneous regions with edges, corners, lines, etc. But for classification purposes, not all of them are useful to identify objects. Furthermore, our object database is composed of similar objects or different objects but with specific local regions with the same color tonalities. These regions, of course, would lead to produce some difficulties when we try to recognize or classify objects. We reduced the number of this problematic points using a technique described by Ohba and Ikeuchi [99]. Using the Principal Component Analysis (PCA) to obtain an eigenspace of all the training set of points, we eliminate all those local color histograms that, once projected in this eigenspace, are very similar. PCA will be explained later in section 3.2 and, as we will see, it consists of finding a subspace representation that minimizes the mean squared error (MSE). It turns out that we eliminate some noise using this subspace representation and also, at the same time, we find a representation with less dimensions. Using this subspace representation and given two training histograms H_1 and H_2 and their respective projections in the subspace h_1 and h_2, we use the L_2 distance in the eigenspace (see section (2.1.3)). If this measure is less than a certain threshold, then these two histograms H_1 and H_2 are removed from the training set because we assume that are conflictive and can produce some confusions since they belong to two different pharmaceutical products.

Each pharmaceutical product of the object database is composed of 6 instances taken under different poses. We randomly select 2 instances for the training set and 4 for the testing set. We performed different series of experimental tests. The first one is to consider a global histogram of the whole image including the background. We should note that all images contain a pharmaceutical product and a black background. Also, we have tested the same technique but removing the background (since it is a black background this is an easy step). Up to this point, we can state that recognition based on global histograms performs very bad since we obtain recognition rates of 48.75% and 57% for both cases as seen in table (3.1). As predicted before, the main cause of this result is that a number of our objects present very similar color regions (i.e. large white regions are pervasive in the design of pharmaceutical product boxes!).

As a first step to local histogram analysis, we have extracted a set of local color histograms from each object as described before and we grouped all of them in a single histogram. That is, we obtained the mean color histogram from each object but only considering those regions that are relevant (i.e. homogeneous regions are not considered). Using this representation, we obtain a global representation of the interesting parts of the object. As seen in table (3.1) the recognition rate considering a mean color histogram of an object is 67%.

It seems clear that using a global histogram approach of objects can not be taken as a robust technique because we obtain low recognition rates. Here is where we see the need of introducing a local approach to improve the performance of our method. Given an object $\mathbf{H}$ and its set of L detected keypoints $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \dots \mathbf{h}_L]$, with $L = 100$. We create an eigenspace for all local color histograms for all the pharmaceutical products and we remove all the possible conflictive histograms as explained before (in the preprocessing phase). After this, once we have a test object, we obtain its representative local color histograms, we project all of them in the eigenspace found in the training phase and we compare the projected local color histograms (called

eigenhistograms in the literature [147]). In order to compare two projected histograms in an eigenspace, we use the L_2 distance. Object recognition is then based on a voting scheme. Each testing local color histogram votes for the most similar histogram in the database (for the object that contains the most similar training histogram). Thus, an object is recognized when its vote counter is the maximum among all the database objects.

Apart from considering a voting mechanism as described before, we also introduced a Bayesian approach in order to consider the a priori knowledge about histogram distribution of models. If we have K possible classes C^k, a Bayesian approach can be used to classify a set of local color histograms $\mathbf{h}_1, \ldots \mathbf{h}_L$ assigning the most probable target class C^T that describes the testing instance

$$C^T = \text{argmax}_k \;\; p(C^k|\mathbf{H}_1, \ldots \mathbf{H}_L) = p(\mathbf{H}_1, \ldots, \mathbf{H}_L|C^k)p(C^k) \tag{3.1}$$

Since these terms can not be estimated from our training data, we use the naive Bayes classifier that makes our task easier

$$C^T = \text{argmax}_k \;\; p(C^k) \prod_i p(\mathbf{H}_i|C^k) \tag{3.2}$$

and we estimate $p(\mathbf{H}_i|C^k)$ by summing the contribution from a mixture of m Gaussians

$$p(\mathbf{H}_i|C^k) = \sum_{i=1}^{m} w_i G(\mathbf{H}_i - \mu_i; \Sigma_i|C^k) \tag{3.3}$$

where this expression assumes that each sample is the center of a Gaussian and we use a kernel method for choosing the parameters. The kernel method positions Gaussians at all the samples in the distribution. In our case, the parameters are $m = N$, $w_i = \frac{1}{N}$, $\mu_i = \mathbf{h_i}$ and $\Sigma_i = [\frac{4}{N(D+2)}]^{\frac{1}{(D+4)}}$ [131], where D is the number of dimensions of our data. In all the experiments we have considered equiprobable priors.

We should note that the voting scheme is a local technique implemented in the subspace created during the training stage. This is done due to the high dimensional data of the original space (512 dimensions). In the testing stage we should search for the most similar training local color histogram. But this operation implies to compare the whole training local color histograms with this new unseen histogram. That is, this requires a prohibitive amount of computational resources if our database is composed of a huge number of pharmaceutical products. Working in the subspace representation, this operation is carried out faster because it is a low dimensional space. However, the Bayesian approach is used in the original space since we estimate a possible model that fits our data and is faster than the previous approach. Using this Bayesian approach, as we will see later on this thesis (see section 3.6), has another problem and is the estimation of a probability density function in a high dimensional space and this produces bad estimations for certain cases (specially when we do not have enough training samples to represent a certain pharmaceutical product).

As reflected in table (3.1), local approaches perform better than global ones. It must be said that removing the black background of all images greatly simplifies the

Method	Recognition rate
Global Histograms (NNL2 - with background)	58.75%
Global Histograms (NNL2 - no background)	57%
Mean Histogram (NNL2 - no background)	67%
Local voting scheme (NNL2 in eigenspace)	75.25%
Local Bayesian scheme (original space)	78.75%

Table 3.1: Experimental comparison between global and local techniques when recognizing a set of 100 pharmaceutical products. These results show the effectiveness of local approaches versus global ones. Notation: NNL2- Nearest Neighbor using L_2 metric distance.

complexity of the global representation and the performance of the method increases. After this, considering all those local color histograms from the relevant structures of our pharmaceutical products (not considering the homogeneous regions) implies to improve the global results. The outperformance obtained using a mean color histogram is due to the fact that we do not take into account those zones of the pharmaceutical products that are homogeneous and do not contain relevant information. Finally, we see that local approaches perform better than global ones with a significant amount of difference. A nearest neighbor technique using L_2 in the subspace performs reasonably good and the Bayesian approach applied in the original space of local color histograms performs a little bit better.

Even the good performance of local based techniques, there is a problem using these representations. If we increase the amount of pharmaceutical products, these techniques increase considerably the overhead needed to give an answer to the final user. It is clear that using a voting scheme we should store the whole training database in order to find the nearest neighbors. But this implies to have a prohibitive amount of local storage. Using a Bayesian scheme, we only need to obtain a probabilistic model for each new pharmaceutical product and it seems to be a better solution to this problem. Even a Bayesian approach can be applied to solve such a problem, we must say that working in a high dimensional space means to estimate a probability density function that, in certain objects, can not be reliable. Why not? Because certain objects are described using a few interesting local histograms (maybe 50) and we are estimating a probability density function in a high dimensional space of 512. Then, this means to have too few observations to estimate a reliable probability density function. This drawback has been exposed at the beginning of this chapter where we talked about the *curse of dimensionality* problem. Later on this thesis we will present the Independent Component Analysis (see section (3.6.1)) and we will propose a solution for this problem.

3.2 Data Representation using PCA

The above explained experiment where we experimented with local color histograms shows how we can use the Principal Component Analysis (PCA). The appearance-based visual learning of objects and scenes is commonly realized using Principal Com-

ponent Analysis. PCA is a classic technique in statistical data analysis, feature extraction, and data compression. The roots of PCA can be found in the early work of Pearson [104] and its first application shortly after in a classic paper by Spearman [133]. In the latter, the author considered data that consisted of school performance rankings corresponding to schoolchildren and determined a single linear combination of the data such that it explained for the maximum amount of variation in the results, claiming to have found a general factor of intelligence. After this, PCA was first formulated by Hotelling [59] in 1933 and since then it has been used in various applications in many areas.

Principal component analysis is a linear transformation from a high-dimensional input space to a low-dimensional feature space, which among all linear transformations guarantees the best possible representation of the high-dimensional input vectors in the low-dimensional feature space. It rotates the coordinate frame in a data-driven way, such that the variability of the input data can be efficiently described using only a small number of basis vectors.

PCA produces very good results, if the high-dimensional input vectors are correlated. This means that they contain redundant information. PCA removes the redundancy by decorrelating the input vectors; the new coordinates of the input vectors (principal components) are uncorrelated. As a consequence, the correlated high-dimensional input vectors can be efficiently represented as the uncorrelated low-dimensional vectors of principal components making PCA a very powerful tool for data compression.

The detailed formulation and description of PCA is found in appendix A.

3.3 Data Representation using NMF

In some computer vision problems extracted feature vectors are inherently non-negative. A typical example is the set of pixels that describe an image patch extracted from an interesting point of an object. If PCA is used to describe a non-negative data vector, the PCA bases would contain negative values, making it impossible to interpret the bases themselves. Since it is very natural to describe feature vectors using positive representations such as pixels or histograms, it is very interesting to think in a subspace description of data only using positive constraints. In the literature, there is a model observing this situation, the Non-negative Matrix Factorization (NMF) [79]. This model is a linear representation that minimizes the mean squared error (MSE criterion) as PCA but using non-negativity constraints. As seen later, the main and maybe the most important consequence of these non-negative constraints is that we obtain a parts-based representation. It is for that reason that we introduce NMF as a new alternative to the typical PCA model in order to find positive representations of subspaces that are related to positive space domains (pixels values or histograms).

In this section, we present the original Non-negative Matrix Factorization (NMF). NMF can be expressed using different approaches. For example, we can use an euclidean distance algorithm or we can use a divergence algorithm. In fact, these two approaches try to minimize the error obtained by different models but, at the end, the solutions are very close [157]. After this, we explain how to project new unseen

data vectors and we present a graphical example where we show the main objective of NMF.

3.3.1 Non-negative Matrix Factorization

Each learning algorithm has its own assumptions that bias generalization over particular problems and thus restrict the scope of applications. For instance, PCA assumes an interesting subspace should orthogonally span maximum variance directions or, equivalently, have low mean reconstruction error. Via these assumptions, PCA results in a popular technique useful for general dimensionality reduction, data compression, etc. Another consequence of these assumptions is that, through a global treatment of the input, PCA provides a holistic representation.

Could we think of a similar technique but for positive data? There is a model with such a situation and is called Non-negative Matrix Factorization (NMF) [79], and the main consequence of the non-negativity constraints is that only non-subtractive combinations of data are allowed. This ensures that the components are combined to form a whole in a non-subtractive manner. For this reason, NMF yields a parts-based representation opposed to the holistic representation obtained through other methods such as PCA [79]. Localized features offer several advantages in the context of object recognition, including stability to local deformations, lighting variations and partial occlusions. In addition there exists psychological and physiological evidence favouring a parts-based representation in the brain [84, 101, 150]. Notice also, that NMF is of straight application in several problems related to visual recognition where the features are naturally non-negative (pixel intensities, histogram values, etc.) and where non-negative components have a direct interpretation. For instance, if PCA were applied to a number of sample histogram, the PCA bases would contain negative values, making it impossible to interpret the bases themselves as histograms.

Given a non-negative matrix $\mathbf{X}$ of size $m \times n$, NMF algorithm seeks to find non-negative factors $\mathbf{W}$ and $\mathbf{H}$ such that

$$\mathbf{X} \approx \hat{\mathbf{X}} \equiv \mathbf{WH}, \quad \text{where } \mathbf{W} \in \Re^{m \times r} \text{ and } \mathbf{H} \in \Re^{r \times n} \tag{3.4}$$

Intuitively, we think of $\mathbf{W}$ as the matrix containing the NMF bases and $\mathbf{H}$ as the matrix containing the accompanying coefficients (weights).

We also desire that the resulting factorization require less storage than the original data set. If an $m \times n$ matrix can be thought of as requiring mn units of storage, NMF produces factors $\mathbf{W}$ and $\mathbf{H}$ requiring $r(m+n)$ units of storage. The reasonable assumption of keeping the precision of NMF factors equal to the precision of the original matrix ensures that storage will be reduced whenever the number of basis vectors, r, is chosen such that:

$$r < \frac{nm}{m+n} \tag{3.5}$$

In practice, r is usually chosen such that $r \ll \min(m, n)$ and its selection is analogous to choosing the desired rank in rank reduction problems[1] or the dimensionality of the

[1] Actually, the problem of choosing r for NMF is very difficult in contrast to choose a desired rank in traditional rank reduction techniques.

subspace created by PCA as seen in section 3.2. To see this we write the factorizations in terms of the columns of $\mathbf{X}$ and $\mathbf{H}$:

$$x_j \approx \hat{x}_j = \mathbf{W}h_j, \quad \text{where } x_j \in \Re^m \;\; h_j \in \Re^r \quad \text{for } j = 1, \ldots, n \tag{3.6}$$

Using this representation, we see that the left factor $\mathbf{W}$ contains a basis used for the linear approximation of $\mathbf{X}$. The right factor $\mathbf{H}$ is a coefficient matrix used to add up combinations of the basis vectors in $\mathbf{W}$. The non-negative constraint on $\mathbf{W}$ allows us to visualize the basis columns in the same manner as the columns in the original data matrix. This is the first benefit of NMF versus alternative factorization methods where the basis vectors conatain negative components that prevent similar visualization. If our data $\mathbf{X}$ are image collections, this basis consists of r representative images stored in the columns of $\mathbf{W}$.

The elements of $\mathbf{H}$ may be thought of as (non-negative) coefficients used to weigh the linear combination of basis vectors used to approximate each column in $\mathbf{X}$. The non-negative restriction on these coefficients result in the additive nature of NMF. For many types of data (see for example: houses and facial images in [79], handwriting samples in [77], and music tones in [70]), the additive property of NMF has been shown to result in bases that represent components of the original data (i.e. - doors and eyes, curves of letters and notes in a chord).

The non-negativity constraints on both $\mathbf{W}$ and $\mathbf{H}$ do not come without a cost. From a rank reduction standpoint, NMF is more computationally demanding and produces lower quality approximations when compared with other traditional alternatives. Non-negative matrix factorizations can be very difficult to compute. Lee and Seung have suggested an approach similar to that used in Expectation-Maximization (EM) algorithms (see [38]). This approach seeks to iteratively update the factorization based on a given objective function. We now introduce the two traditional objective functions that produce the NMF algorithms used in the literature. Both algorithms introduced here each seek to minimize a different objective function (distance measure) and each of these objective functions could be minimized with several different iterative procedures. The particular update strategies given here are shown because of their implementation ease and because they have been proven to monotonically decrease their respective objective function (see [79]).

3.3.2 Euclidean Distance Algorithm

First, we consider the Euclidean distance between each column of $\mathbf{X}$ and its approximation $\hat{\mathbf{X}} = \mathbf{WH}$. For computational purposes, we will use the sum of the squared distances between each column vector x_j in the original data matrix and its approximation $\hat{v}_j \approx \mathbf{W}h_j$. Using this distance measure, we arrive at the following objective function:

$$\Theta_{\mathrm{NMF}_E}(\mathbf{W}, \mathbf{H}) \equiv \sum_{j=1}^{n} \|x_j - \mathbf{W}h_j\| = \|\mathbf{X} - \mathbf{WH}\| \equiv \sum_{i=1}^{m}\sum_{j=1}^{n} \left(X_{ij} - \sum_{l=1}^{r} W_{il}H_{lj} \right)^2 \tag{3.7}$$

In doing NMF using the Euclidean distance algorithm, we wish to find the factors $\mathbf{W}$ and $\mathbf{H}$ that minimize the objective fucntion $\Theta_{\mathrm{NMF}_E}(\mathbf{W}, \mathbf{H})$. The lower bound of this

objective function is zero and will only be attained when a strict equality $\mathbf{X} = \mathbf{WH}$ is obtained. There are many ways to minimize this objective function, however, because of the lack of convexity in both variables $\mathbf{W}$ and $\mathbf{H}$, we can, at best, expect to achieve only local minima [79].

Thus far, researchers in this area have chosen to balance algorithm complexity and convergence speed by using the following update procedure:

$$H_{aj} \leftarrow H_{aj}\frac{(\mathbf{W}^T\mathbf{X})_{aj}}{(\mathbf{W}^T\mathbf{WH})_{aj}} \tag{3.8}$$

$$W_{ia} \leftarrow W_{ia}\frac{(\mathbf{XH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}} \tag{3.9}$$

When written this way, it is evident that the update consists of multiplying the current factors by a measure of the quality of the current approximation. Under this updates, the Euclidean distance objective function Θ_{NMF_E} has been proven [80] to be monotonically decreasing:

$$\Theta_{\mathrm{NMF}_E}\left(\mathbf{W}^{(t+1)},\mathbf{H}^{(t+1)}\right) \leq \Theta_{\mathrm{NMF}_E}\left(\mathbf{W}^{(t)},\mathbf{H}^{(t)}\right) \quad \text{for } t = 0,1,\ldots \tag{3.10}$$

3.3.3 Divergence Algorithm

The second objective function that is commonly used in practice is called the divergence, or entropy, measure:

$$\Theta_{\mathrm{NMF}_D}(\mathbf{W},\mathbf{H}) \equiv \mathrm{Div}(\mathbf{X}\|\mathbf{WH}) \equiv \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{\sum_{l=1}^{r}W_{il}H_{lj}} - X_{ij} + \mathbf{WH}_{ij}\right) \tag{3.11}$$

The objective function $\Theta_{\mathrm{NMF}_D}(\mathbf{W},\mathbf{H})$ is not a distance measure because, strictly speaking, it is not symmetric in $\mathbf{X}$ and $\mathbf{WH}$. Further motivation behind this objective function can be seen when the columns of $\mathbf{X}$ and the columns of the approximation $\mathbf{WH}$ sum to 1. In this case, $\Theta_{\mathrm{NMF}_D}(\mathbf{W},\mathbf{H})$ reduced to the Kullback-Leibler information measure used in probability theory [80]. This objective function is related to the likelihood of generating the columns in $\mathbf{X}$ from the bases $\mathbf{W}$ and encoding coefficients $\mathbf{H}$.

Again, as in the previous Euclidean distance algorithm, this objective function equals its lower bound of zero only when we have strict equality, $\mathbf{X} = \mathbf{WH}$. To balance complexity and speed, the following update rules are commonly used:

$$\begin{aligned} H_{aj} &\leftarrow H_{aj}\sum_i W_{ia}\frac{X_{ij}}{(\mathbf{WH})_{ij}} \\ W_{ia} &\leftarrow W_{ia}\sum_j \frac{X_{ij}}{(\mathbf{WH})_{ij}}H_{aj} \\ W_{ia} &\leftarrow \frac{W_{ia}}{\sum_j W_{ja}} \end{aligned} \tag{3.12}$$

Lee and Seung have proven in [80] that this algorithm monotonically decreases the objective function Θ_{NMF_D}:

$$\Theta_{\mathrm{NMF}_D}\left(\mathbf{W}^{(t+1)},\mathbf{H}^{(t+1)}\right) \leq \Theta_{\mathrm{NMF}_D}\left(\mathbf{W}^{(t)},\mathbf{H}^{(t)}\right) \quad \text{for } t=0,1,\ldots \tag{3.13}$$

We should point out that the divergence algorithm has some interesting properties that should be mentioned because they will be used in this thesis. The objective function defined in equation (3.11) can be expanded as:

$$\Theta_{\mathrm{N}MF_D}(\mathbf{W},\mathbf{H}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log X_{ij} - X_{ij}\log\sum_{l=1}^{r}W_{il}H_{lj} - X_{ij} + \mathbf{WH}_{ij}\right) \tag{3.14}$$

and as we can appreciate, term X_{ij} refers to the original data and always is a constant value. So that, it is very common to rewrite the objective function as:

$$\Theta_{\mathrm{N}MF'_D}(\mathbf{W},\mathbf{H}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log(\mathbf{WH})_{ij} - (\mathbf{WH})_{ij}\right) \tag{3.15}$$

When the objective function is written under this form, NMF can also be stated as a problem of likelihood maximization but with the assumption that $\mathbf{X}$ is drawn from a Poisson distribution with mean $\mathbf{WH}$ [80]. In this case, the distribution therefore takes the form:

$$P(X_{ij}|(\mathbf{WH})_{ij}) = e^{-(\mathbf{WH})_{ij}}\frac{(\mathbf{WH})_{ij}^{X_{ij}}}{X_{ij}!} \tag{3.16}$$

Now, if we take the logarithm of both sides, we can derive to such a formulation:

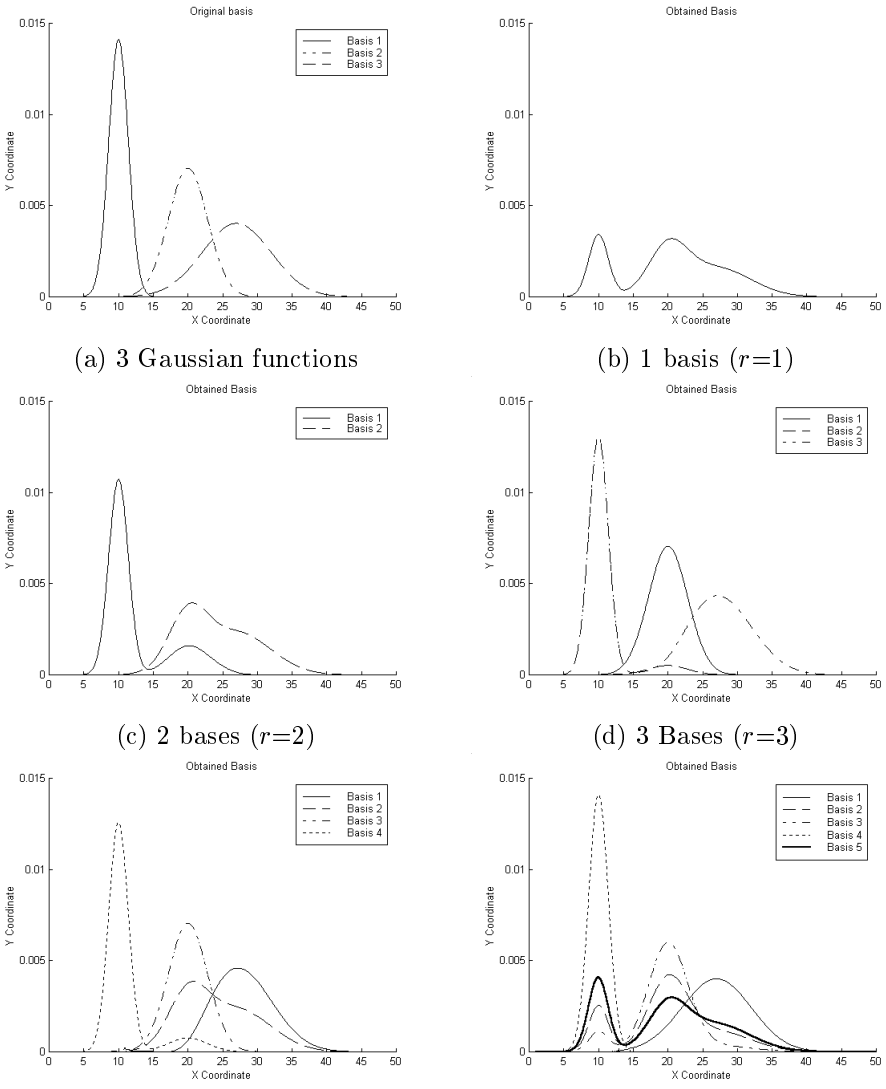
$$\log(P(X_{ij}|(\mathbf{WH})_{ij})) = X_{ij}\log(\mathbf{WH})_{ij} - \mathbf{WH}_{ij} - \log(X_{ij}!) \tag{3.17}$$

As we want to find out a good approximation of matrix $\mathbf{X}$, the term $\log(X_{ij}!)$ can be dropped because it is a function of $\mathbf{X}$ only, and makes no difference when optimizing with respect to $\mathbf{W}$ and $\mathbf{H}$. As can be seen, we have arrived to formulate expression (3.15). As all elements from all matrices (indexes ij) are conditionally independent, we can formulate expression (3.17) as a summation over ij as reflected in expression (3.15). Expression (3.17) takes the form of a $y\log x - x$ function and it is very easy to verify that the maximum of $y\log x - x$ with respect to x is at $x = y$, so maximizing the objective function should tend to make $\mathbf{X} \approx \mathbf{WH}$.

3.3.4 Graphical Example of NMF

We have generated a set of synthetic vectors to show how NMF is able to obtain the basis functions used to make up the original set. Using 3 Gaussian basis functions that can be seen in figure (3.4.a), we have created a set of 10 samples by weighting the original bases (see figure (3.5)).

Taking each generated vector of figure (3.5), we have created a 1000 dimensional histogram dividing the horizontal axis in 1000 partitions. Thus, each bin of this histogram contains the value of the corresponding vertical axis. Applying NMF over



(a) 3 Gaussian functions

(b) 1 basis (r=1)

(c) 2 bases (r=2)

(d) 3 Bases (r=3)

(e) 4 Bases (r=4)

(f) 5 Bases (r=5)

Figure 3.4: (a) 3 original basis functions used to make up the set of 10 training vectors used in our experiment. (b) to (f) are the different basis functions obtained after applying the NMF method with different configurations of the parameter r. If we expect to find a small amount of bases ($r = 1, 2$), the obtained bases are combinations of the global behaviours of the sample data. If we only want 3 basis functions ($r = 3$), the obtained bases are exactly the same as the original ones (a). Furthermore, if we want to extract a large amount of basis functions ($r = 4, 5$), the extracted basis functions are combinations of the specific behaviours of the sample data.

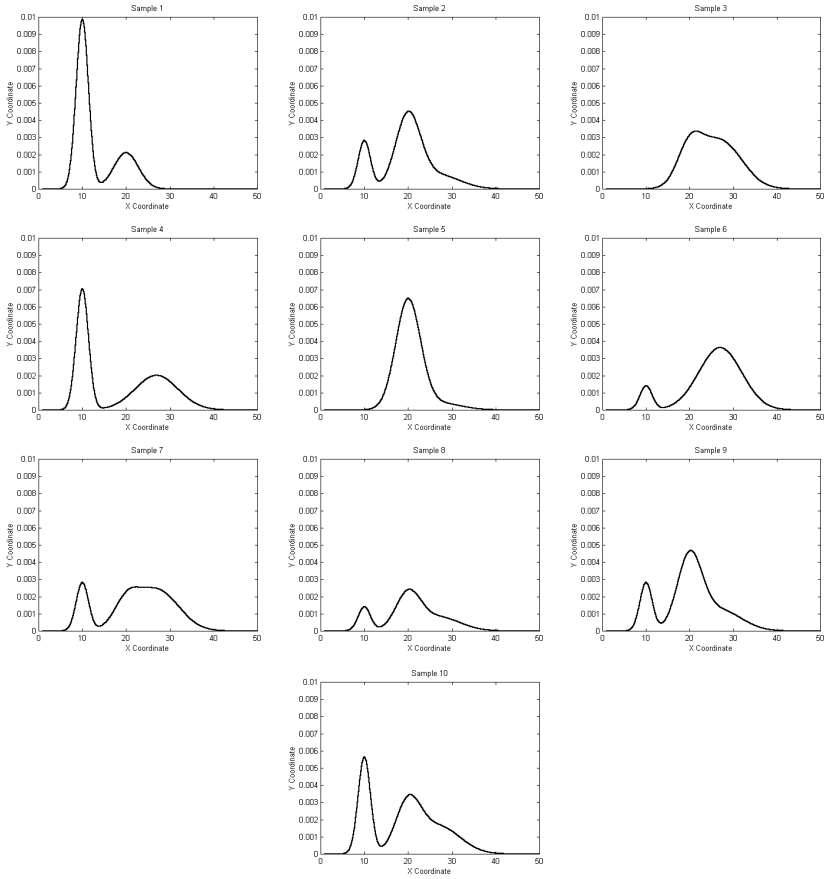


Figure 3.5: 10 differents samples created by weighting the 3 Gaussian functions defined in figure (3.4.a). As it can be seen, these samples contain mixed behaviours from the original data vectors.

this set of 10 1000 dimensional histograms and expecting to have a visual idea of the obtained r bases, the results are shown in figure (3.4). We can compare figure (3.4.a) with the extracted bases and since we have created this synthetic problem using 3 Gaussian functions, when we apply NMF with $r = 3$ (see figure (3.4.d), the obtained histograms of matrix $\mathbf{W}$ are nearly the same used for generating our original histograms.

In pattern recognition, the extraction of meaningful features from input signals is one of the most important problems. The synthetic problem exposed here in order to have a visual idea of what is doing NMF with our data can be seen under this point of view: We only know a set of samples and we want to know the intrinsic features used to create these samples. That is, when we tried to obtain $r = 3$ bases, we have discovered the real bases used to generate our data. But, in computer vision, since data is obtained from real world conditions and the number of bases is not clear, the

problem is more complicated.

In the literature one might find an application of NMF which is applied to Munsell color spectra [21]. In [21] they applied NMF to color spectra, an inherently non-negative data set, to investigate which and how many color categories are revealed and compared with other classical color categories. They applied NMF with a set of Munsell colors and they evaluated which colors are represented in the obtained bases. This application can be seen as a an analogy with the experiment we performed in this section as we recovered the original bases that were used to create the observed samples.

3.3.5 Projection of Unseen Vectors

The two previous explained NMF approaches to obtain non-negative factorizations of data is a guarantee that provide us with a stable solution (probably a local minimum solution). But up to this point we find that there is not a method to project new unseen data vectors using the set of bases $\mathbf{W}$. Usually, once we have a linear projection matrix, each time that a new unseen vector should be projected we can use the following expression

$$\mathbf{y} = \mathbf{W^{-1}x} \tag{3.18}$$

where $\mathbf{W}$ is the data matrix containing the bases of NMF. Here, the main problem is that the inverse of a positive matrix results in a matrix with positive and negative values. Inverting this matrix infringes the non-negativity constraint imposed by NMF. To overcome this drawback the projected vectors are found by running the same iterative update rules. Firstly, we use the input vector $\mathbf{X}$ with the new unseen vectors and then, we fix the bases (matrix $\mathbf{W}$) to the ones obtained in the training step. After this, we initialize matrix $\mathbf{H}$ with random positive values and we iterate our update rules trying to obtain a good solution of the problem. Using this approach, the resulting weights are always non-negative.

3.4 Data Representation using WNMF

Some data distributions can be slanted and may be not uniformly distributed. Local representations usually produce this kind of distributions. This problem is mentioned in [21] where they used NMF to recover basis functions of Munsell colors. But since Munsell colors do not have a well-balanced calibrated distribution, in terms of the color names, lightness, and saturation, basis functions of NMF contained repetitions. The original work of NMF [79] where they worked with faces, they had also this problem: several versions of ears and eyes emerge in several NMF bases. In order to reduce the effect of data that is not uniformly distributed, we introduce the use of a new technique, the Weighted Non-negative Matrix Factorization (WNMF). This technique is mainly based on the typical NMF but with a weight matrix that helps to reduce the redundancy of NMF when applied to data that is not uniformly distributed. Since we introduce this weighted version of NMF, we compare both techniques. We chose a local problem where data is expressed in terms of local color histograms and we compared the performance of both NMF and WNMF techniques in terms of their

reconstruction abilities. With this example, we see that WNMF is well suited to local data representations but only when the subspace dimension is correctly chosen. Finally, we compare both techniques in terms of the required number of iterations in order to find which technique is faster.

3.4.1 Weighted Non-negative Matrix Factorization

One of the main disadvantages of NMF arises when applied to local representations because this data is not uniformly distributed. In the field of visual object recognition, the earlier systems were focused on holistic object representations, the object as a whole. This approach has been successfully used in different applications such as face recognition or robot positioning, being its main advantage the ability to perform fast and reliably at low spatial resolutions and without any kind of prior knowledge. Still, there are some problems that prove difficult to solve for these kind of representations, like partial object occlusions and severe lighting changes. As mentioned in chapter 2, several approaches have recently been proposed using local window representations as reliable solutions to occlusions, complex backgrounds, scale changes, illumination changes and different viewpoints or orientations. This local perspective of a problem relies on domain specific knowledge to address the problem such as what to look for in an object and where that feature should be. This allows for a richer class representation and, consequently, a model that can be used to focus on more complex situations. Local models, representing a more specific and possibly less complex part of the object usually have high levels of redundancy among classes. Nevertheless this redundancy is dealt when the combination of parts is performed, it might present a problem to certain unsupervised learning techniques. Since NMF is a unsupervised learning technique that tries to represent our sample data using a fixed amount (r) of bases ($\mathbf{W}$), redundant local samples generate redundant bases. This situation does not arise with PCA due to the assumption of basis orthogonality. Such assumption is not possible within the NMF context since it would clearly break the nonnegativity constraints. Since we realized about this problem and we did a slight modification of NMF that, through the introduction of a sample weight matrix, we minimize the problem of generating redundant bases $\mathbf{W}$. We call this approach Weighted Non-negative Matrix Factorization (WNMF) and we performed several experiments in different contexts to illustrate its advantages over the classical approach.

The main reason NMF finds redundant bases in local representations is the fact this approach extracts several feature vectors from each object instead of a single data vector. Feature vectors can be redundant and strong similarities can exist among them. These similarities are not taken into account in the classical NMF approach: several identical feature vectors have more weight in the cost function than a few strongly different vectors and possibly relevant for the representation samples. We try to solve this problem by introducing a weight on each of the input samples, giving more weight to those vectors with low probability of appearing in the original dataset. Since we know that vectors with high probability of appearing in the dataset would be favoured by NMF, giving more weight to the other ones would tend to reduce redundancy in the model. This weighted model can be seen as the result of right-multiplying both sides of the factorization with a $n \times n$ diagonal weight matrix $\mathbf{Q}$ and

to estimate the bases and encodings for the new factorization model,

$$\mathbf{XQ} \approx \mathbf{WHQ} \text{ where } \mathbf{W} \in \Re^{m \times r}, \ \mathbf{H} \in \Re^{r \times n} \text{ and } \mathbf{Q} \in \Re^{n \times n} \tag{3.19}$$

and the diagonal elements q_i of matrix $\mathbf{Q}$ correspond to the weight of sample vectors $\mathbf{x}_i$, with $1 \leq i \leq n$. Now, with this new weight matrix, the modified objective function takes the form:

$$\Theta_{\text{WNMF}_D}(\mathbf{W}, \mathbf{H}) = \sum_{j=1}^{n} q_j \sum_{i=1}^{m} (X_{ij} \log(q_j (\mathbf{WH})_{ij}) - (\mathbf{WH})_{ij}) \tag{3.20}$$

By considering that $\mathbf{Q}$ is a diagonal matrix, then $q_j = Q_{jj}$. Comparing both objective functions from expressions (3.15) and (3.20) we can see that are nearly the same and the main difference between them is that we have included a weight for each sample vector. In this particular case, expression (3.20) comes out if we assume, again, a Poisson distribution of mean $\mathbf{WHQ}$. Now, the distribution is expressed as follows

$$P((\mathbf{XQ})_{ij} | (\mathbf{WHQ})_{ij}) = e^{-(\mathbf{WHQ})_{ij}} \frac{(\mathbf{WHQ})_{ij}^{(\mathbf{XQ})_{ij}}}{(\mathbf{XQ})_{ij}!} \tag{3.21}$$

and if we take the logarithms of both sides,

$$\log(P((\mathbf{XQ})_{ij} | (\mathbf{WHQ})_{ij})) = (\mathbf{XQ})_{ij} \log(\mathbf{WHQ})_{ij} - (\mathbf{WHQ})_{ij} - \log((\mathbf{XQ})_{ij}!) \tag{3.22}$$

Then, this expression can be rewritten as follows

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} Q_{jj} \log(Q_{jj} (\mathbf{WH})_{ij}) - Q_{jj} (\mathbf{WH})_{ij}) \tag{3.23}$$

So that, considering that $\mathbf{Q}$ is a diagonal matrix and $Q_{jj} = q_j$, we arrive to the expression defined in (3.20). We have to note that $\mathbf{Q}$ is a constant matrix, so that, the therm $\log((\mathbf{XQ})_{ij})$ can be dropped.

Now, the iterative update rules to obtain the new factored matrices subject to this new objective function are defined by:

$$H_{aj} \leftarrow H_{aj} \sum_{i} W_{ia} \frac{X_{ij}}{(\mathbf{WH})_{ij}} \tag{3.24}$$

$$W_{ia} \leftarrow W_{ia} \sum_{j} \frac{q_j X_{ij}}{(\mathbf{WH})_{ij}} H_{aj} \tag{3.25}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}} \tag{3.26}$$

These new update rules are easily derived from those defined in expression (3.12). Expression (3.12) can be rewritten in terms of two new matrices $\mathbf{X}'$ and $\mathbf{H}'$ that are defined as $\mathbf{X}' = \mathbf{XQ}$ and $\mathbf{H}' = \mathbf{HQ}$. Now, the update rule concerning to W_{ia} can be written as:

$$W_{ia} \leftarrow W_{ia} \sum_{j} \frac{X'_{ij}}{(\mathbf{WH}')_{ij}} H'_{aj} \tag{3.27}$$

so, considering that $\mathbf{X}' = \mathbf{XQ}$ and $\mathbf{H}' = \mathbf{HQ}$, this expression leads to this new one:

$$W_{ia} \leftarrow W_{ia} \sum_j \frac{X_{ij}Q_{jj}}{(\mathbf{WH})_{ij}Q_{jj}} H_{aj}Q_{jj} \tag{3.28}$$

So, the update rule concerning to $\mathbf{W}$ is nearly the same as in the usual NMF but considering the weight matrix $\mathbf{Q}$ as

$$W_{ia} \leftarrow W_{ia} \sum_j \frac{X_{ij}}{(\mathbf{WH})_{ij}} H_{aj}Q_{jj} \tag{3.29}$$

With the update rule concerning to the matrix $\mathbf{H}$ we see that expression

$$H'_{aj} \leftarrow H'_{aj} \sum_i W_{ia} \frac{X'_{ij}}{(\mathbf{WH}')_{ij}} \tag{3.30}$$

and replacing $\mathbf{H}' = \mathbf{HQ}$ and $\mathbf{X}' = \mathbf{XQ}$, we easily find that

$$H_{aj}Q_{jj} \leftarrow H_{aj}Q_{jj} \sum_i W_{ia} \frac{X_{ij}Q_{jj}}{(\mathbf{WH})_{ij}Q_{jj}} \tag{3.31}$$

Thus, update rule of matrix $\mathbf{H}$ remains exactly the same as in expression (3.12) because all Q_{jj} terms vanish.

3.4.2 Weight Matrix Selection

Once we derived all the whole set of update rules for the weighted version of NMF, we should know how to select the weight factors to build up matrix $\mathbf{Q}$. As explained before, classical NMF finds a redundant representation when the sample data is extracted from a local representation because it is a non-uniformly distribution of data. Then, we introduced the fact to give more weight to those vectors with low probability of appearing in the data set.

If we assume that we have a data set of n vectors (x_j for $j = 1, \ldots n$) and we assume that we know how to obtain the probability of a given vector to belong to the data set: $p(x_j|M)$ where M is a model of our data set. Then we are able to know the probability of each sample vector to belong to the whole data set as

$$p_j = p(x_j|M) \quad j = 1, \ldots n \tag{3.32}$$

So that, each sample vector x_j has its corresponding estimated probability to belong to the data set using a certain model. Then we invert these probabilities $p'_j = \frac{1}{p_j}$. Once having the inverted probabilities to give more importance to the less frequent sample vectors, we obtain the weights as

$$q_j = \frac{np'_j}{\sum_{l=1}^{n} p'_l} \tag{3.33}$$

We have to note that this is one possible way to obtain the weights to give more importance to the less frequent sample vectors and other possibilities can be considered.

In computational learning, the fact to give more importance to those vectors with low probability of appearance in the data set is not very common. Usually, the most important vectors are those that are only needed to obtain a good model of the whole data set. This is a different situation, with the usual NMF we find that matrix **W** contains redundant information (repeated bases in our case) and this is due to the fact that we are initializing our model with random values leading to converge to a stable solution through a minimization process. Here, the solution is local and since our data is extracted from a local representation, redundancy is also present in the original data.

3.4.3 Comparison of NMF versus WNMF

In this section, we present a graphical example that shows the performance obtained by both NMF and WNMF techniques. In the previous section, we mentioned that using NMF with local data representations tends to obtain redundant bases in matrix **W** and here we show what we meant by redundancy in this context.

We have selected 5 different color newspapers (see figure (3.6)) each of them containing particular local characteristics. Having 10 instances of each newspaper, we have divided all of them using a predefined grid obtaining a large amount of local regions (200 per image). Each local region is represented using the combination of one 8-bin histogram per spectral band (resulting dimensionality is 512) and all of them are used as input in the learning process of the NMF matrices (**W** and **H**). Because of the initial random conditions of the algorithm, we have initialized both approaches with the same random matrices, for accurate comparison. The selection of the weights (matrix **Q**) in the WNMF approach is performed using a leave one out technique that tests the probability of finding the corresponding histogram in the whole original set of histograms. Once we have estimated matrix **W** for both approaches, other local color histograms were projected. These additional histograms were extracted from the same objects but using a different grid. Finally, the reconstruction error of both approaches was compared using expression (3.7). Notice that, since NMF lacks restrictions, reconstruction error on the training set should be less than for WNMF. What we are actually measuring here is how well the reconstruction generalizes to a test set. This experiment was performed for a different number of bases $r = 30, 35, 40, 45, 50, 55$. Results are shown in figure (3.7) where we can see the evolution of the reconstruction error of the testing set against the number of iterations. We have to note that for the training stage, we learned our models using approximately 400 iterations of the algorithm, until it stabilized. For the test stage (the projection) we only made 50 iterations. Observing the figure we can deduce that much less iterations were needed until stabilization.

As noted in expression (3.5) the number of desired bases r is generally chosen so that $(n+m)r < nm$ where m is the dimension (512 in this particular case) and n the size of the dataset (10.000 in this particular case). With this rule, we can choose up to $r = 487$ bases. In figure (3.7), we obtain that WNMF cannot outperform NMF if it does not have a sufficient number of bases to represent specific tonalities (remember that our features are color histograms). If we consider $r \geq 40$ bases we have the opposite behaviour. To illustrate this situation we trained both models with a variable

Figure 3.6: Five different color newspapers each of them containing some characterstic local regions.

number of bases from 20 up to 200 in steps of 5 and a fixed number of iterations (50). Results are shown in figure (3.8) in terms of the number of bases against the difference between reconstruction errors. The WNMF error was subtracted to the NMF error so a positive value indicates a superior performance of the first.

Figure (3.8) confirms that with few bases, NMF generalizes better than its weighted version. Having a large amount of training vectors, NMF seeks the best bases that represent all this space and will unlikely generate redundant bases. Instead, WNMF gives more weight to specific color regions which might not be relevant in terms of reconstruction error, but that would be surely avoid redundancy with an increased number of bases. This is also confirmed. We observe that, depending on the variability of the input data NMF will start to generate redundant bases at some point in which simultaneously WNMF will improve its performance. In our specific problem, this point is found when $r = 40$ and it is useful to know this number if we have to choose the best representation for a given problem. This situation, in which WNMF outperforms NMF in reconstruction apparently is a contradiction with the fact that NMF is a direct attemp to minimize the reconstruction error. Reasons for this result should be found in the fact that the optimization landscape for our problem is non-convex, so both methods are likely to fall in local minima of the objective function. Since, from a theoretical standpoint, the only difference between both techniques is in the gradient directions and step, we can affirm that WNMF provides a better initialization for gradient descent, and consequently a local minimum nearer to the optimal solution.

Taking advantage of the nonnegativity of the solution we can also visualize the results: the basis vectors are themselves color histograms. From these histograms we can generate artificial images with the adequate proportion of colors. We chose the number of bases as 45 and generated such images for both approaches. The results are shown in figure (3.9). In both approaches we can already see the main characteristic of NMF: a specialized representation. This translates in sparse histograms: each basis vector accounts for a few colors, generally only one or maybe the combination of two colors. The differences between both approaches can also be observed. In the NMF bases there exist several repeated histograms. Notably white which is present alone, as well as mixed with other colors. This is understandable since white is the predominant color in the newspapers used for learning our representation. WNMF avoids this problem, localizing white in a single basis and diminishing its presence in

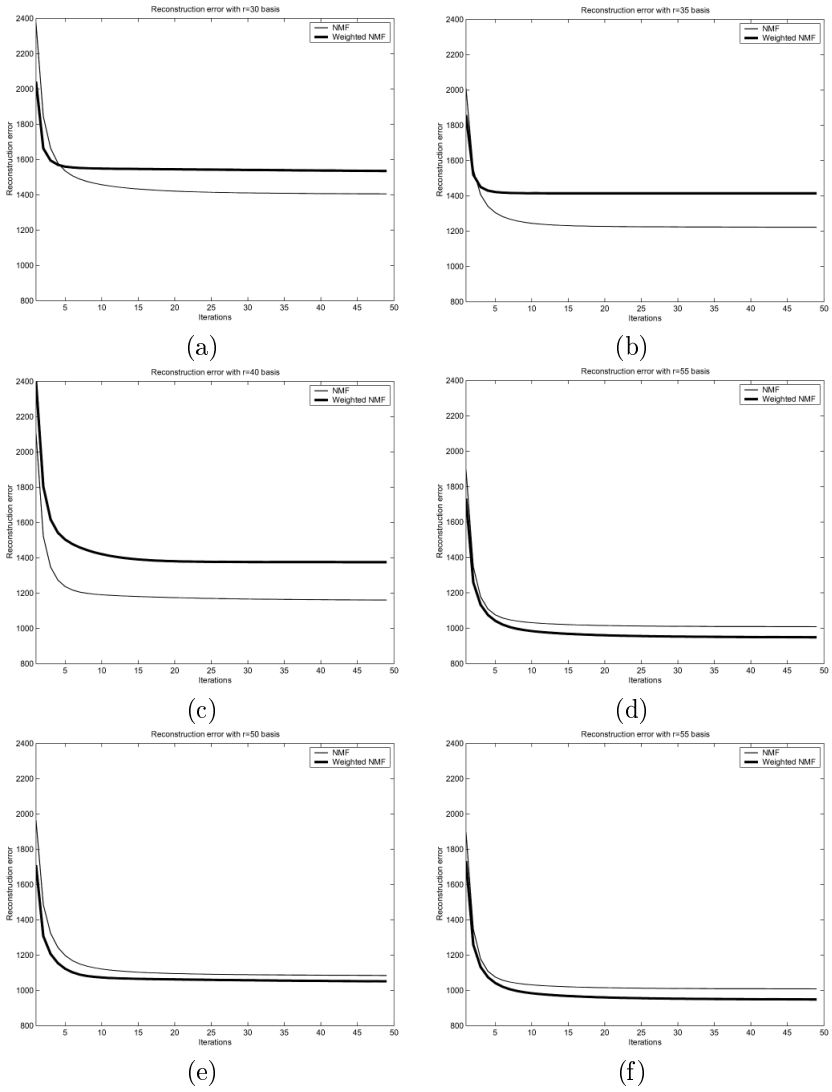


Figure 3.7: NMF (light line) and WNMF (solid line) reconstruction error against iterations on the test data. Using a small number of bases $r = 30, 35, 40$ from (a) to (c), NMF outperforms WNMF. When $r = 45, 50, 55$ from (d) to (f), WNMF outperforms NMF.

other bases. Also, WNMF contains 5 different green tonalities against 2 of the classical approach. This kind of specialization is more likely to provide a better discrimination among objects, i.e. if a newspaper (or ad, maybe) is associated with a particular green tonality, it will be simpler to discriminate it from other newspapers (ads).

Appendix B presents a full detailed set of experiments comparing NMF and WNMF under different frameworks. The above graphical comparison between NMF

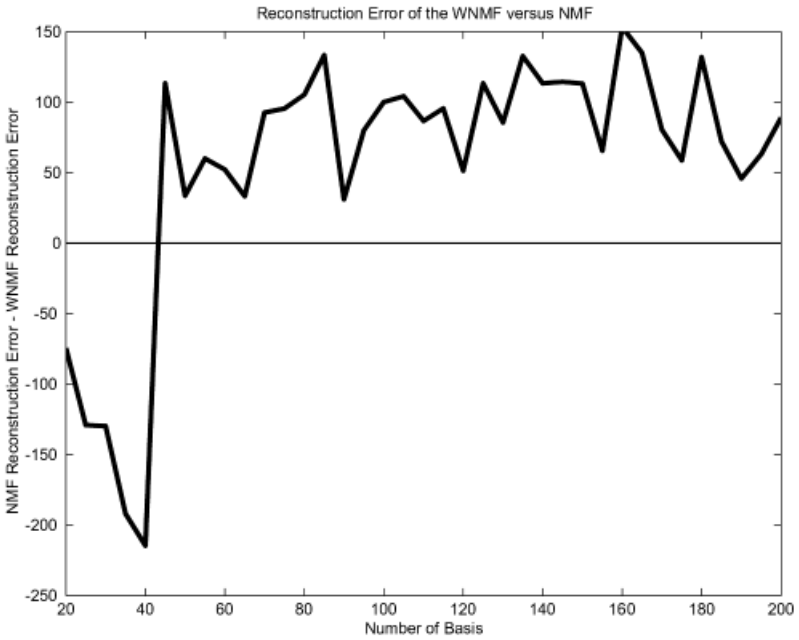
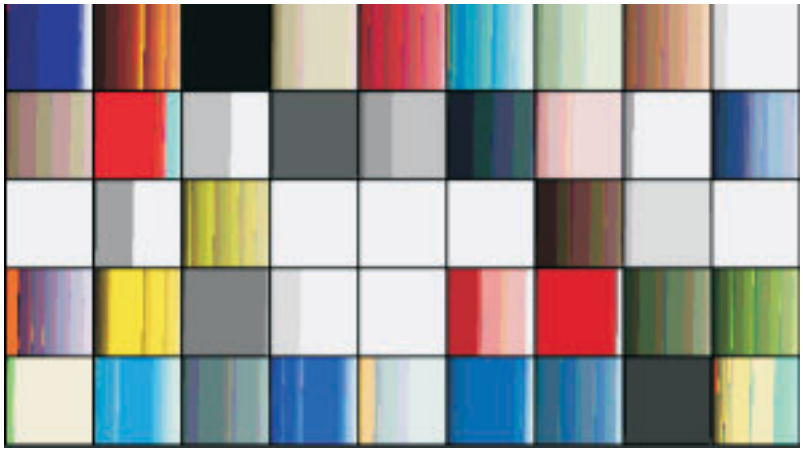


Figure 3.8: WNMF error subtracted to the NMF error against r, the number of bases of the factorization.

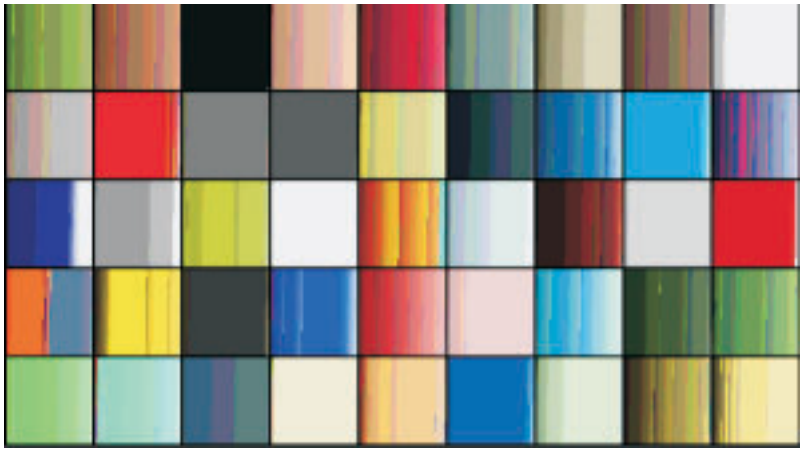
and WNMF has been used as a good example to distinguish the performance of WNMF in front of the classical NMF but other experiments should be carried out in order to do a final statement about the improvement of WNMF with respect to NMF. In appendix B, two different experiments related to the performance of WNMF with respect to NMF are exposed. The first experiment evaluates the performance of the Weighted version of NMF with respect the classical approach in the context of object recognition of natural patch classes. A second experiment evaluates the reconstruction error of WNMF and NMF in order to evaluate which technique performs faster in terms of the number of iterations.

3.5 Empirical Analysis of Linear Transformations using Local Features

Linear transformations of data produce a linear mapping of one high dimensional space to a low dimensional one (subspace). This linear mapping helps in reproducing the original data in a compact manner and the computational costs required to deal with the subspace are lower. We presented 3 linear techniques that are commonly used for obtaining reduced subspaces: Principal Component Analysis, Non-negative Matrix Factorization and its weighted version. However, it remains to know how we can use these techniques to classify new unseen instances of objects. In this section, we will show 3 different techniques that can be used to perform it. These methods



(a) 45 Histogram obtained using NMF



(b) 45 Histogram obtained using WNMF

Figure 3.9: Histogram bases obtained using (a) NMF and (b) WNMF. Note that the number of histogram bases containing the white color are reduced using WNMF and that WNMF contains 5 different histogram bases with green tonalities against 2 green tonalities of the classical NMF.

are:

1. **Reconstruction distances**: All techniques are based on finding a subspace description that minimizes the mean squared error (MSE). NMF/WNMF use only positive restrictions but PCA uses positive and negative ones. The first approach to be used in this context is: how well these techniques reproduce the original space? and the first measure to be used in this particular case is the reconstruction distance.

2. **Parametric model**: NMF and PCA are based on different assumptions. PCA comes out if we assume that the original data distribution is modeled using a Gaussian distribution. In contrast to this, NMF/WNMF are based on a Poisson distribution of data. Here, we want to perform classification using a parametric model of the original data distribution. However, we are only using the projected data. So that, we try to estimate the original parametric distribution using the projected coefficients.

3. **Non-parametric model**: The last, and the most common way to classify new unseen instances of objects is to use the projected coefficients of the subspace. Taking as a reference the projected coefficients, a usual technique to be used is the k-nearest neighbors. That is, given a test vector and a set of training vectors previously stored in the system, we search for the most similar training neighbor in the projected space and the new unseen data vector is classified according to this information.

These classification methods are compared using a set of local color histograms extracted from the corel image database. Some of the image samples of this image database are shown in section (C.2). For the non-parametric case, we evaluated several image databases: handwritten digits and face images.

3.5.1 Classification using Reconstruction Distances

In the PCA context (see appendix A), the reconstructed version of a projected vector is given by expression (A.18). Then, if we compare the original data vector and the reconstructed one, we will have an error measure as seen in expression (A.22). So that, PCA is based on finding a linear combination of the first k principal components (k is chosen by us) that best approximates the original data vector. Then, expression (A.22) evaluates the quality of this approximation.

In the NMF context (see section 3.3.2), the reconstructed version of a projected vector is given by $\mathbf{X} = \mathbf{WH}$ (when we iterate the same algorithm keeping $\mathbf{W}$ constant as explained in section (3.3.5)). Then, if we compare the original data vector with the reconstructed one (see expression (3.7)), we will also have a way to evaluate the quality of this approximation.

In this section we present an experimental test which compares PCA and NMF reconstruction abilities in terms of these reconstruction distances. It is clear that PCA should be better than NMF since NMF pursues the same objectives as PCA but in a restricted manner (only using positive data). However, we will see that we can take into account the different nature of NMF to merge both methods using reconstruction distances. A multiclass (10 classes) problem is exposed and we will see that NMF is well suited for certain classes and PCA for other ones. Then, we will analyze whether it is possible to guess a priori which technique (PCA, NMF or WNMF) represents better a certain data class. To do this, we will extract a set of statistics about our data classes. Then, we will analyze whether we can find a connection between the performance of each method and the extracted statistics (elongation of class).

We used the corel image database. 10 data classes are extracted as explained in appendix (C.2) and each data class is composed of a set of local color histograms.

Class Name	Error Mean	Error Variance
Tree	0.5194	0.0231
Rocky Mountains	0.6041	0.0262
Grass	0.6165	0.0401
Snow Mountains	0.6312	0.0398
Leaves	0.7020	0.0329
Water	0.7568	0.0402
Clouds	0.8125	0.0415
Ice	0.8136	0.0514
Sand	0.8480	0.0456
Sky	1.1450	0.0874

Table 3.2: Ordered list of data classes according to the mean error using the euclidean distance in the original space of local color histograms. Here, all vectors of a given class are compared with the other ones of the same class. The first class (Trees) is the most compact class and the last one (Sky) is the most dispersed one. This experiment has been done using 1000 local color histograms per class randomly selected from the whole database.

Then, as relevant statistics, we believe that the measure of dispersion of our data in the original space of color histograms should be of relevant importance. We chose the measure of dispersion of our data classes because as stated before, PCA is optimal in terms of the reconstruction error since it provides a subspace representation with low mean reconstruction error. But the problem of PCA is that assumes data gaussianity, so that, some data classes with no Gaussian behaviour may not be correctly classified using PCA. The measure of dispersion is somehow a measure of how elongated and how compact is our data class in the original space. NMF tries to find a subspace representation without orthogonality but assuming only positive data and this restriction implies to have a higher reconstruction error with respect to PCA. However, this does not mean that this non negative representation is not useful for recognition or classification.

In order to evaluate the level of compactness of a specific class, we used the euclidean metric (L_2) between the elements of each data class. Since PCA is optimal in terms when obtaining the minimum mean squared error (MSE), L_2 is the best metric distance to be used with PCA. Given a data class, say *Clouds* we evaluate how all the local color histograms of this class are related between them, so that, we obtain a measure as

$$\alpha_{\text{Compact}} = \sum_i \sum_j d_{L_2}(V^i_{\text{Clouds}}, V^j_{\text{Clouds}}) \quad \forall i,j \in \text{Class}_{\text{Clouds}},\ i \neq j \quad (3.34)$$

Where V^i_{Class} is element i of data class *Class*. It is expected that this measure of error gives us an idea of how each data class is distributed in the original 512 dimensional space of local color histograms. Table (3.2) shows the level of compactness of each data class in terms of the mean error and the mean variance.

By analyzing table (3.2) we are able to have an idea of the compactness of our data classes in the original space. Now, we know that the Sky class is a very dispersed

Class Name	Eigenvalue analysis
Tree	0.146
Rocky Mountains	0.195
Grass	0.209
Snow Mountains	0.219
Leaves	0.262
Water	0.306
Clouds	0.350
Ice	0.356
Sand	0.382
Sky	0.698

Table 3.3: Ordered list of data classes according to the sum of the eigenvalues of the corresponding covariance matrices of each data class. The first class of this table denotes a data class with low eigenvalues indicating a compact class. This ordered list is the same as the one presented in table (3.2).

class and the Tree class is the most compact one. Here, we use the term *compact class* in the sense that all the local color histograms of a class are concentrated in a specific region or not. We can also obtain this level of compactness analyzing the covariance matrix of data. Eigenvalues of the covariance matrix can help us to give an idea of the elongation of each axis of our data.

Since one of our techniques to evaluate is PCA, with this technique we are explicitly assuming that our data is a Gaussian in the original space. So that, if we assume that our data is modeled using a Gaussian distribution, eigenvalues of the covariance matrix will be a clear indicator of the elongation of each axis. We averaged all the eigenvalues of each data class and the results are shown in table (3.3) where we see that we obtain the same ordered list that the one presented in the previous table (3.2).

We are assuming that a dispersed class is the same as to consider a class with several local behaviours. This could be true or not and it does not depend on the level of compactness but if we have a data class that is dispersed in the original space it is expected that we have a complex class and maybe it could not be represented using a Gaussian. Since PCA is based on assuming that our data can be represented using a Gaussian, it would be expected that PCA can not be used properly and maybe other techniques can be used.

For image classification, we take 1000 local color histograms per class randomly selected from the whole database. 1000 data vectors have been used for training our models and 1000 data vectors for testing. So that, we learn 10 models using PCA and 10 models using NMF. We choose to use 60 dimensions to represent each class. Once we have a given class and both models representing this class, we decided to use the euclidean distance between the original data and the reconstructed data. That is, the reconstruction error. As expected, PCA is optimal in terms of this error (in average) but this does not mean that NMF can improve the representation of certain vectors (maybe those vectors far away from the general behaviour of the class). To know this, we compare the reconstruction error obtained using PCA and NMF for

Class Name	Mean error (PCA)	Mean error (NMF)	PCA > NMF
Grass	0.40	5.16	904
Tree	1.45	3.63	765
Sand	2.14	13.67	717
Ice	0.46	5.88	692
Rocky Mountains	8.32	21.65	683
Leaves	10.76	21.61	578
Water	6.34	22.47	563
Snow Mountains	3.74	14.47	541
Clouds	4.85	18.34	522
Sky	9.34	43.59	439

Table 3.4: Reconstruction error using PCA and NMF. Second and third columns show the mean error of the retroprojected histograms using PCA and NMF respectively. It can be seen that PCA can always represent the original data better than NMF in global terms but the fourth column shows the number of local color histograms that are better represented with PCA. This table is sorted according to the number reflected in the fourth column, so that, the grass class contains a large number of local vectors that are better represented using PCA than NMF and the sky class contains more vectors that are better represented using NMF than PCA.

each particular data class. Table (3.4) shows this experiment where it is clear that for some classes, the reconstruction error provided by NMF is better than the one provided by PCA when we consider each vector alone.

From table (3.4) we can state that the most dispersed class, the Sky class, can be better represented with NMF than PCA. The rest of data classes are better represented using PCA. But it has to be noted that it is difficult to predict which is the best representation for a given data class since we do not know whether the data class is Gaussian or not. Furthermore, we should not that table (3.4) shows the behaviour of each data class with respect its own data, i.e. the grass model can represent better the grass data when we use PCA instead of NMF, but, this does not mean that this model is good for classification since maybe it can not be used for discrimination purposes (classification against other data classes).

Here, we can observe that the behaviour of both techniques, PCA and NMF is different depending on the data class. Since we have differences between these techniques it seems natural to think that in terms of recognition these differences can be exploited somehow in order to take into account the advantages of both techniques at the same time. This is what we want to evaluate in the following.

The first classification experiment compares both PCA and NMF in terms of recognition rates. Given a new local color histogram that we want to classify and we know that belongs to one of our ten data classes, we project it in all the 10 models learned with PCA and NMF and we choose the model that better represents the original local color histogram. To compare the performances of boths methods, we show the confusion matrices obtained after classification in tables (3.5) and (3.6).

These two tables (3.5) and (3.6) can be understood as follows: Taking as a reference the table (3.5), given 1000 local color testing histograms of each data class

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**244**	1	358	0	4	32	325	18	0	18
Grass	0	**927**	0	14	0	33	0	0	26	0
Ice	17	0	**854**	0	0	3	90	10	2	24
Leaves	0	250	31	**662**	0	10	0	0	31	16
Rocky	26	47	29	0	**285**	363	35	29	155	31
Sand	21	26	22	0	25	**874**	7	0	23	2
Sky	18	0	351	0	0	16	**563**	25	0	27
Snow	89	0	411	0	3	33	160	**218**	5	81
Tree	1	207	3	20	14	39	1	2	**687**	26
Water	19	10	439	1	3	37	131	13	18	**329**
Total Recognition Rate: 56.43 %										

Table 3.5: Confusion matrix obtained using the PCA technique in a 60 dimensional subspace and using the euclidean distance of the retroprojected vectors.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**433**	2	141	0	7	33	217	110	0	57
Grass	0	**788**	0	53	15	28	0	0	116	0
Ice	115	3	**552**	1	5	1	89	119	3	112
Leaves	5	87	18	**793**	5	15	2	8	62	5
Rocky	22	24	17	0	**405**	285	33	71	131	12
Sand	42	10	8	0	116	**786**	2	12	22	2
Sky	213	6	160	1	8	13	**371**	54	0	174
Snow	112	0	196	0	17	15	82	**458**	3	117
Tree	0	68	2	30	58	21	1	1	**808**	11
Water	28	3	163	7	34	12	46	79	33	**595**
Total Recognition Rate: 59.89 %										

Table 3.6: Confusion matrix obtained using the NMF technique in a 60 dimensional subspace and using the euclidean distance of the retroprojected vectors.

(i.e. clouds), 244 local color histograms are classified as clouds, 1 vector as grass, 358 vectors as Ice, and so on. If we compare both confusion matrices, the best technique is the one that has higher values in the diagonal. It can be seen that clouds are better classified using NMF than using PCA, but grass is better classified using PCA than using NMF. At the end, NMF performs better than PCA since it has a total recognition rate of 59.89% in front of a recognition rate of 56.43% of the PCA. Our goal is to find a good model in the sense that it has to be able to represent the elements of its own data class without classifying elements of other classes to belong to its class.

In order to find a good model, we should analyze the confusion matrices in a vertical way. With this analysis, we will be able to know the best model for each kind of data and this is very interesting since we can select the best models of both algorithms and create a mixture of models according to the level of classification achieved using the testing data set. If we want to know which is the best model for each kind of data, we define a measure that weighs the correct classification results (diagonal entries of the confusion matrices) and the bad classification results in a vertical way. The measure selected to evaluate this behaviour is defined as:

$$\alpha_{\mathbf{CLASS}} = \frac{\text{Vectors correctly classified of } \mathbf{CLASS}}{\text{Number of vectors of other classes classified as } \mathbf{CLASS}} \tag{3.35}$$

Using this $\alpha_{\mathbf{CLASS}}$ as a reference, we evaluated both techniques (PCA and NMF) taking into account each data class and the results are the ones presented in table

(3.7).

	PCA	NMF
α_{Clouds}	**1.2775**	0.8063
α_{Grass}	1.7135	**3.8818**
α_{Ice}	0.5195	**0.7830**
α_{Leaves}	**18.91**	8.6196
α_{Rocky}	**5.819**	1.52
α_{Sand}	1.5442	**1.8582**
α_{Sky}	0.7517	**0.7860**
α_{Snow}	**2.2474**	1.0088
α_{Tree}	**2.6423**	2.1838
α_{Water}	**1.4622**	1.2143

Table 3.7: This table shows the $\alpha_{\mathbf{CLASS}}$ for each data class with respect to the reduction technique used (PCA or NMF). In bold face we can appreciate which technique performs better for all data classes.

Table (3.7) reflects that Grass, Ice, Sand and Sky classes are better classified using NMF than using PCA but all the other data classes are better classified using PCA. Now, it seems interesting to merge both classifiers but only considering the best models for each particular data class. Taking as a reference the best technique for each data class as reflected in table (3.7), we use NMF to represent Grass, Ice, Sand and Sky data classes and PCA for all the other ones. The confusion matrix obtained for this particular case is shown in table (3.8).

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**464**	1	182	0	10	32	201	60	1	49
Grass	0	**783**	0	61	22	19	0	0	113	2
Ice	59	0	**674**	1	1	0	109	45	4	107
Leaves	2	81	7	**825**	7	6	0	1	44	27
Rocky	44	18	9	0	**433**	204	30	36	177	49
Sand	69	8	10	0	116	**744**	1	2	42	8
Sky	62	3	261	0	1	17	**543**	50	1	62
Snow	213	0	256	0	13	6	47	**324**	6	135
Tree	2	56	1	36	24	8	0	2	**838**	33
Water	55	4	185	2	17	12	39	67	27	**592**
Total Recognition Rate: 62.20 %										

Table 3.8: Confusion Matrix for the combined PCA and NMF technique in $60D$ using the euclidean distance of the retroprojected vectors. Grass, Ice, Sand and Sky models are represented with a NMF model and the rest of classes are represented using the PCA technique.

As noted in table (3.8), recognition rates increase from 56.43% and 59.89% to 62.20% with this combined method.

Another approach to take into account when merging both methods is to consider a hierarchical point of view. That is, since it seems natural that some classes are more related to some other ones, we can try to create a classifier specialized to discriminate conflictive data classes in a hierarchical point of view. Since we are working on color

histograms, we can previously know which data classes are conflictive, i.e. data classes with blue tonalities would be conflictive since they can share some blue tonalities. Tables (3.9) and (3.10) show the number of times that a given test vector is bad classified. Analyzing these two tables, we would be able to know which data classes are conflictive and we can use specialized models.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	0	4	513	2	24	58	560	172	10	85
Grass	0	0	0	14	12	34	0	0	31	0
Ice	32	2	0	0	6	4	93	29	6	39
Leaves	20	271	45	0	27	36	14	15	100	40
Rocky	104	104	94	25	0	425	98	122	202	126
Sand	42	30	32	0	43	0	27	17	32	18
Sky	99	7	376	2	15	22	0	100	5	51
Snow	377	7	538	2	41	67	435	0	30	198
Tree	7	231	14	46	33	74	4	11	0	36
Water	181	34	527	9	61	57	344	217	46	0

Table 3.9: This matrix reflects all the testing vectors that are bad classified using PCA. When an original data vector is classified to belong to another data class, the matrix reflects this error with its corresponding value.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	0	10	248	10	46	57	302	270	22	136
Grass	4	0	2	63	46	42	3	2	151	3
Ice	201	8	0	7	21	14	143	231	16	141
Leaves	23	108	38	0	45	40	17	35	105	35
Rocky	101	58	78	13	0	324	68	155	167	78
Sand	66	26	34	12	160	0	21	71	65	26
Sky	377	33	320	26	36	45	0	246	26	351
Snow	220	4	292	5	51	37	156	0	20	182
Tree	5	89	8	42	74	33	4	16	0	17
Water	184	47	263	45	96	52	109	250	86	0

Table 3.10: This matrix reflects all the testing vectors that are bad classified using NMF. When an original data vector is classified to belong to another data class, the matrix reflects this error with its corresponding value.

Tables (3.9) and (3.10) are interpreted as follows: if a local vector to be tested that we previously know that belongs to the class clouds, for example, has a reconstruction distance using a model that is lower than the reconstruction distance using its own model (clouds), we add one error to the model because it is a conflictive model. Analyzing matrices of tables (3.9) and (3.10) we can state that Clouds, Ice, Sky, Snow and Water classes are really difficult to separate because they contain a high level of misclassifications. The main idea now is to create a classifier that works only using blue tonalities, so that, we will have a more specialized classifier. So, if we firstly classify our data vectors as to belong to a blue tonality against a green/brown tonality, we find a general classifier that can be refined for further classification tasks. Again, we can combine at each level of this hierarchical representation the best technique (PCA or NMF). Assuming this new architecture, we find that the best configuration is the one shown in figure (3.10).

Figure (3.10) shows the optimal hierarchical representation used to classify new testing vectors. The best technique to use (PCA or NMF) is depicted under the name

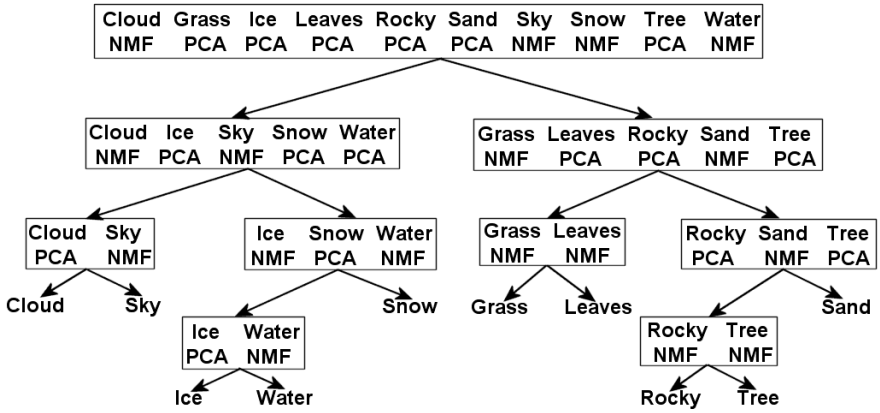


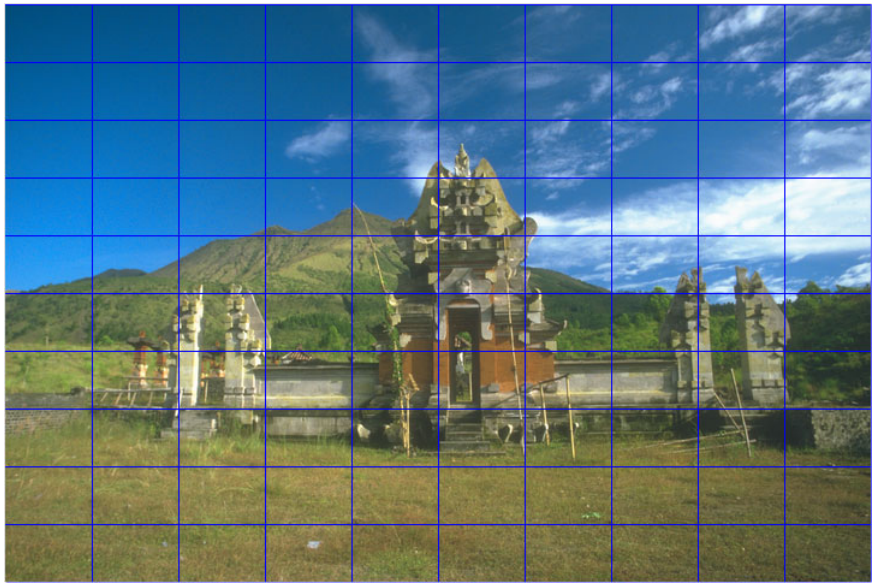
Figure 3.10: Optimal hierarchical representation in order to classify the original 10 data classes. Given a node in the tree and having a local color histogram to classify between the right or the left leaves, we obtain the reconstruction distance of this vector and we choose the leaf that contains the lower reconstruction distance. Under the name of each data class, we have the name of the technique used to represent the data.

of each data class and as it can be seen, depending on the data classes that we want to discriminate, the technique can change and this behaviour can be extracted from the matrices shown in tables (3.9) and (3.10). Now, the new confusion matrix using this hierarchical representation is shown in table (3.11).

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**370**	1	160	0	16	32	202	167	1	51
Grass	0	**795**	0	51	26	18	0	0	110	0
Ice	91	2	**625**	0	2	2	90	77	3	108
Leaves	0	91	22	**809**	6	6	0	8	49	9
Rocky	12	20	14	0	**507**	209	24	64	140	10
Sand	13	9	12	0	148	**780**	3	5	27	3
Sky	50	2	131	0	8	9	**618**	94	1	87
Snow	93	0	243	0	31	14	41	**517**	7	54
Tree	0	61	3	33	66	8	0	2	**822**	5
Water	19	4	173	7	35	15	26	122	39	**560**
Total Recognition Rate: 64.03 %										

Table 3.11: Confusion Matrix for the combined hierarchical PCA and NMF technique in $60D$ using the euclidean distance of the retroprojected vectors.

Up to this point, we can state that the combination of PCA and NMF techniques using a hierarchical scheme performs better than the other classifiers. Since we have extracted our initial local color histograms from images, we can use the neighborhood information of each local color histogram to correct some of the classification results obtained up to now. For example, if we decide to classify a cloud region, it is straightforward to think that a cloud region is surrounded by other cloud regions. So that, assuming this information can help us to improve the initial recognition results. Figure (3.11) shows an image of the database with its corresponding labeled regions.



(a) Original image

Sky	Sky	Sky	Sky	Cloud	Sky	Sky	Sky	Cloud	Cloud
Sky	Sky	Sky	Sky	Cloud	Cloud	Cloud	Cloud	Cloud	Cloud
Sky	Sky	Sky	Sky	Cloud	Cloud	Cloud	Cloud	Sky	Cloud
Sky	Sky	Sky	Sky			Cloud	Cloud	Cloud	Cloud
Sky	Sky	Rocky	Rocky			Cloud	Cloud	Cloud	Cloud
Grass	Grass		Tree						Tree
Grass									Tree
Grass	Grass	Grass							
Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass
Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass	Grass

(b) 100 local regions labeled according to our data classes.

Figure 3.11: An image of the database that is divided into $10 \times 10 = 100$ local regions and its corresponding labeled regions.

In order to take into account the neighborhood information, we propose two easy techniques to consider the local neighborhood of a certain zone to correct possible misclassifications. We named these two easy techniques as: *Highest neighborhood* and

Mean neighborhood.

- **Highest neighborhood:** Let us assume that we only have two models (X and Y) and we are using the same technique as described before: We want to classify a local vector as the model that reconstructs the vector with the minimal reconstruction distance. Table (3.12) shows two tables describing an imaginary scenario with some reconstruction distances obtained for our two models (X and Y). Following the assumptions mentioned in this present work, the classification results obtained for this particular situation are the ones presented in table (3.12.c). The problem of this classification approach is that some local regions can be classified completely different with respect to the whole neighborhood. For example, we can find a water region surrounded by sky regions and, of course, the probability of being a water region knowing that the rest of the neighborhood is composed of sky regions is very low.

  Knowing this, the first solution is to take a local color histogram and perform a classification as the most probable entity from its neighborhood. That is, evaluate the neighborhood of a color histogram and classify the vector as the most frequent entity of the neighborhood. For example, if we have a water region but all the neighborhood is composed of sky regions, classify the water region as sky. Table (3.13) shows the classification results for our imaginary situation of table (3.12).

- **Mean neighborhood:** Previous method assumes that we have a final classification decision for each local vector and according to this classification we want to correct some of the misclassifications. Instead of this, we can also work with the reconstructed distances. For example, if one local vector is classified different from the rest of its neighborhood, it means that its reconstruction distance is below than the rest. Taking the reconstruction distances of all the neighborhood vectors can help us to make a final decision about the entity of the local region. So that, given a local region and its neighborhood, we can use all the neighborhood reconstruction distances and obtain an averaged value to be used for the final decision. Table (3.12) is converted to the values of table (3.14).

0.1	0.3	0.7	0.1
0.4	0.2	0.9	0.1
0.6	0.3	0.1	0.2
0.9	0.6	0.3	0.3

(a) Model X

0.8	0.7	0.4	0.5
0.6	0.3	0.8	0.6
0.5	0.4	0.3	0.7
0.8	0.5	0.2	0.6

(b) Model Y

X	X	Y	X
X	X	Y	X
Y	X	X	X
Y	Y	Y	X

(c) Classification

Table 3.12: (a) and (b) are two imaginary situations of two models X and Y with their corresponding reconstruction distances. (c) Reflects the classification results considering the minimal reconstruction distances of both models X and Y of (a) and (b). Each local region is classified without taking into account its neighborhood.

In order to test these two new methods we should work with the whole database of color histograms instead of a random subset of them since we want to evaluate the

X	X	X	X
X	X	X	X
? → X	Y	X	X
Y	Y	X	X

Table 3.13: Classification results considering the most frequent model appeared in the neighborhood of each local region. These results are based on table (3.12). In this example, there is one ambiguous region labeled as ? that has 3 local regions with model X and 3 with model Y. In this particular case of having the same number of models in the neighborhood, we can classify the local region as the model that has the minimal distance by summing its reconstructed distances. Model X sums $0.4 + 0.2 + 0.3 = 0.9$ and model Y sums $0.5 + 0.8 + 0.5 = 1.8$, so, this ambiguous region is labeled as model X.

0.25	0.43	0.38	0.45
0.32	0.4	0.32	0.35
0.5	0.47	0.33	0.32
0.6	0.47	0.3	0.22

(a) Model X

0.6	0.6	0.55	0.57
0.55	0.53	0.52	0.55
0.52	0.48	0.48	0.53
0.55	0.45	0.45	0.45

(b) Model Y

X	X	X	X
X	X	X	X
X	X	X	X
Y	Y	X	X

(c) Classification

Table 3.14: (a) and (b) are the mean of the neighborhood reconstruction distances of each local region. These two tables are the result of considering the neighborhood of each model of table (3.12). (c) Shows the final classification results considering the tables of (a) and (b).

neighborhood of each local color histogram. Now, the following results are obtained taking into account the whole database of 50864 local regions. Results are shown in table (3.15).

Model Representation	Direct Implementation	Neighboorhood Mean	Neighboorhood Highest	Neighboorhood Highest + Mean
PCA	55.6	60.79	56.54	61.46
NMF	59.5	63.44	61.94	64.57
PCA + NMF Mixed	63.12	67.36	65.18	**69.21**
PCA + NMF Tree	64.87	**NO SENSE**	67.54	**NO SENSE**

Table 3.15: Experimental results using the whole database of 50864 testing vectors. It can be seen that the best configuration is achieved using a PCA + NMF mixed representation in conjunction with both neighborhood techniques obtaining a recognition rate of 69.21%. The particular case of using a hierarchical representation in conjunction with the neighborhood mean technique produces an error because we obtain the result of 55.6% (the same as using the PCA). This is due because in the hierarchical representation, all classes are represented at some level of the tree by the PCA model and this model leads to represent vectors with lower reconstruction distances. So, if we combine the reconstruction distances obtained by PCA using the neighborhood mean technique, this leads to obtain the same results as using the PCA technique.

Table (3.15) shows the improvement achieved using both neighborhood techniques. In this table, column *direct implementation* refers to the direct implementation of all

the four destribed methods. Column *neighborhood mean* refers to the implementation of all the algorithms when considering the mean of the neighborhood distances using eight-connectivity. Column *Highest neighborhood* refers to the technique that considers the most frequent class of the neighborhood of a local region. The last column considers the junction of both neighborhood techniques and it seems that we obtain the best recognition results.

3.5.2 Classification using a Parametric Model

The previous experiment compares PCA and NMF in terms of the distance from feature space (DFFS) as explained in section (A.3). This methodology is valid because both techniques reduce the dimensionality of the problem in terms of the L_2 norm. NMF uses only positive restrictions and PCA uses positive and negative values. As said, PCA assumes that the whole set of data vectors are described by a Gaussian distribution in the original space. However, NMF assumes that data is described using a Poisson distribution. Here is where we want to introduce the fact that both models can be described using a probabilistic point of view and both models can be compared in terms of probabilities.

In section (A.3) we show that PCA can be explained in terms of a Gaussian assumption. More precisely, expression (A.35) describes a good estimator of the probability density function of the original data space using the projected coefficients. This estimator is found by assuming that the original distribution is described by a Gaussian model. In contrast to this, NMF assumes a Poisson distribution and the probability density funcion is the one of expression (3.16). In this section, we explore how to merge PCA and NMF using these probability density function estimators. We repeat the same experiment as in the previous section. Additionally, we introduce the WNMF approach.

Table (3.16) shows the confusion matrix for PCA using a 60 dimensional subspace representation but this time using a probabilistic approach. As seen, comparing results with table (3.5), we obtain a slight improvement from 56.43% to 57.41%. Also, in table (3.17) we show the results obtained when we consider the probabilistic approach of NMF using a 60 dimensional subspace representation. Now, the improvement obtained using the probabilistic approach of NMF is better than the one obtained with PCA, from 59.89% to 62.11%. Finally, a first experiment with WNMF has been done using the probabilistic approach and we obtain a final recognition rate of 62.89%. This result can be seen in table (3.18).

As in the previous experiment, tables (3.16,3.17,3.18) show that there are some techniques that perform better for certain classes. So that, we try to combine these three techniques using the probabilistic approach in order to evaluate the final performance. In order to select the best technique for each data model, we use the definition of α_{CLASS} defined in expression (3.35) and the results are shown in table (3.19).

Comparing table (3.7) with table (3.19) we observe that nearly all the data classes that were represented using PCA are also represented with PCA. The other data classes, the ones represented with NMF, now, some of them are chosen to be represented with WNMF.

From all these experiments we can conclude stating that WNMF performs better

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**251**	1	354	0	4	31	323	18	0	18
Grass	0	**945**	0	8	0	29	0	0	18	0
Ice	12	0	**896**	0	0	3	68	3	2	16
Leaves	0	245	31	**668**	0	10	0	0	31	15
Rocky	24	44	29	0	**293**	360	35	29	155	31
Sand	24	28	23	0	27	**865**	8	0	23	2
Sky	18	0	342	0	0	16	**574**	23	0	27
Snow	92	0	413	0	3	33	162	**211**	5	81
Tree	1	202	3	20	14	32	1	2	**700**	25
Water	19	10	433	1	3	37	128	13	18	**338**
Total Recognition Rate: 57.41 %										

Table 3.16: Confusion matrix obtained using the PCA technique in a 60 dimensional subspace. This is the probabilistic approach of PCA.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**376**	1	228	0	11	30	218	97	0	39
Grass	0	**860**	0	19	1	21	0	0	99	0
Ice	49	1	**752**	0	1	1	65	69	4	58
Leaves	0	148	15	**779**	2	3	0	0	43	10
Rocky	28	30	15	2	**389**	298	20	39	155	24
Sand	15	14	13	3	56	**857**	10	5	23	4
Sky	76	2	228	1	111	14	**453**	71	0	44
Snow	141	0	281	0	21	23	60	**430**	1	43
Tree	0	91	3	22	9	20	0	0	**836**	19
Water	31	4	290	2	13	21	75	61	24	**479**
Total Recognition Rate: 62.11 %										

Table 3.17: Confusion matrix obtained using the NMF technique in a 60 dimensional subspace. This is the probabilistic approach of NMF.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**520**	1	136	0	10	54	168	53	0	58
Grass	0	**823**	0	32	9	28	0	0	108	0
Ice	215	1	**560**	0	7	2	106	51	2	56
Leaves	0	66	6	**851**	4	3	0	6	50	14
Rocky	23	20	17	2	**589**	179	20	46	87	17
Sand	21	12	1	0	158	**777**	5	6	17	3
Sky	136	0	195	0	7	19	**377**	48	0	218
Snow	210	0	177	0	16	22	29	**483**	0	63
Tree	0	145	1	39	42	9	0	1	**737**	26
Water	45	3	175	7	25	15	72	60	26	**572**
Total Recognition Rate: 62.89 %										

Table 3.18: Confusion matrix obtained using the WNMF technique in a 60 dimensional subspace. This is the probabilistic approach of WNMF.

than NMF. Also, a probabilistic scheme of all PCA, NMF and WNMF techniques is better than a direct approach based on reconstruction distances. Why is better to consider a probabilistic approach? We believe that is due to the fact that with a classification scheme based on reconstruction distances, it is not possible to take into account the internal structure of data. Since PCA assumes gaussianity and NMF/WNMF a Poisson distribution, the probabilistic approach considers more information related to the internal distribution of data of each technique and helps to classify vectors.

	PCA	NMF	WNMF
α_{Clouds}	**1.3210**	1.1058	0.8000
α_{Grass}	1.7830	2.9553	**3.3185**
α_{Ice}	0.5504	0.7008	**0.7909**
α_{Leaves}	**23.0345**	15.8979	10.6375
α_{Rocky}	**5.7451**	1.7289	2.1187
α_{Sand}	1.5699	1.9884	**2.3474**
α_{Sky}	0.7917	**1.0112**	0.9425
α_{Snow}	**2.3977**	1.2573	1.7823
α_{Tree}	**2.7778**	2.3954	2.5414
α_{Water}	1.5721	**1.9875**	1.2571

Table 3.19: α_{class} defined in expression (3.35) per each technique. Bold face indicates the best technique of each class.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**517**	1	169	0	8	25	175	57	1	47
Grass	0	**825**	0	53	19	15	0	0	86	2
Ice	53	0	**707**	1	1	0	100	41	4	93
Leaves	2	73	7	**849**	7	6	0	1	36	19
Rocky	37	17	9	0	**476**	186	28	35	167	45
Sand	55	7	10	0	87	**801**	1	2	29	8
Sky	57	3	236	0	1	17	**582**	46	1	57
Snow	198	0	242	0	13	6	42	**370**	6	123
Tree	2	39	1	24	15	7	0	2	**890**	20
Water	43	4	149	2	15	12	34	60	25	**656**
Total Recognition Rate: 66.73 %										

Table 3.20: Confusion Matrix for the combined classifier (PCA, NMF and WNMF) in $60D$ using a probabilistic approach.

	Model Clouds	Model Grass	Model Ice	Model Leaves	Model Rocky	Model Sand	Model Sky	Model Snow	Model Tree	Model Water
Clouds	**559**	1	146	0	8	22	163	53	1	47
Grass	0	**813**	0	55	20	15	0	0	95	2
Ice	49	0	**734**	1	1	0	84	41	4	86
Leaves	2	62	6	**871**	7	5	0	1	31	15
Rocky	36	17	9	0	**537**	155	27	32	148	39
Sand	49	7	7	0	78	**822**	1	2	26	8
Sky	52	3	205	0	1	16	**645**	35	1	42
Snow	179	0	219	0	13	6	41	**431**	6	105
Tree	2	34	1	22	14	7	0	2	**901**	17
Water	39	4	121	2	15	12	25	47	25	**710**
Total Recognition Rate: 70.23 %										

Table 3.21: Confusion Matrix for the optimal hierarchical classifier (PCA, NMF and WNMF) in $60D$ using a probabilistic approach.

3.5.3 Classification using a Nonparametric Model

As a nonparametric model, we used the k-nearest neighbours (k-nn) approach [44]. This approach is mainly based on finding k neighbors for a given vector. Then, we assign to this vector the class label majoritary among its k-nearest neighbors. k-nearest neighbors (k-nn) [44] has for long been probably the most intuitive classification rule