RDFIA: Homework 1-a,b,c

GALMICHE Nathan PIRIOU Victor

October 14, 2022

Abstract

The goal is to produce an algorithm that is able to classify images from the dataset 15-Scenes. This algorithm belongs to the set of traditional methods that were constituting the state-of-the-art in computer vision until the end of the 2000s, before the deep learning era.

Training The training of the algorithm consists of:

1. Decomposing each of the images (of the datasets) into patches.
2. Encoding each of these patches using the SIFT algorithm.
3. Creating a visual dictionary, *i.e.* a list a recurrent patterns present in our dataset.
4. Encoding all the images of the dataset. What we have is an image of size Width × Height. What we want is an output (or an "encoding") vector whose size is equal to the number of class we can possibly find. What we have is an image of size Width × Height. What we want is an output (or an "encoding") vector whose size is equal to the number of class we can possibly find. This dimensionality reduction is done in three steps: First, from each image we deduce a set of descriptors based on the magnitude and the orientation of the gradients of the images, which is a relevant encoding since what characterizes the most an object are its contours. Second, each of the SIFT descriptors is encoded such that it is expressed in function of one or more words of the dictionary. Third, all the SIFT descriptors of each image are aggregated into a compact representation of the image, based on the frequency of detection of the features that can possibly appear in the image.
5. Training a classifier such that it learns the boundaries that separate the classes between each other.

Inference The classification of an image consists of:

1. Encoding the image in the same way that is described above.
2. Using a SVM classifier to assign an image to a class.

1 SIFT descriptor

1. **Show that kernels M_x and M_y are separable, i.e. that they can be written $M_x = h_y h_x^\top$ and $M_y = h_x h_y^\top$ with h_x and h_y two vectors of size 3 to determine.**

   We have

$$\frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = M_x,$$

   and

$$\frac{1}{2}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = M_y,$$

so $M_x = h_y h_x^\top$ and $M_y = h_x h_y^\top$ where $h_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ and $h_y = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$.

2. **Why is it useful to separate this convolution kernel?**
   The convolution product is associative.

   So, $I_x = I * M_x = I * (h_y * h_x^\top) = (I * h_y) * h_x^\top$ and $I_y = I * M_y = I * (h_x * h_y^\top) = (I * h_x) * h_y^\top$. This reduces the computational costs on an $N \times M$ image with a $m \times n$ filter from $\mathcal{O}(M \cdot N \cdot m \cdot n)$ down to $\mathcal{O}(M \cdot N \cdot (m+n))$. Here the complexity is deduced from the number of products involved in the convolution.

3. **What is the goal of the weighting by Gaussian mask?**
   In our case, we sample one 16×16 patch every 8 pixels. Hence, the sampled patches have some overlap. The weighting by a Gaussian mask gives a lower weight to the off-center pixels that belong to several samples.

   This weighting is also used because the center pixels are naturally more representative of the patch than the off-centers pixels.

4. **Explain the role of the discretization of the directions.**
   First of all, the discretization of the directions is needed because without a fixed number of directions we couldn't create the histogram. Second, this discretization increases the robustness to rotation. Indeed, if we slightly rotate a patch, we expect it to have the exact same "signature" vector than that of the original patch. The optimal trade-off between this robustness and the precision of the orientations depends on the context.

5. **Justify the interest of using the different post-processing steps.**
   If it is smaller than a threshold of 0.5, P_{enc} is set to a null vector and is immediately returned. This first thresholding is useful to desensitize the descriptor to the eventual noise of flat patches. Otherwise, the descriptor is normalized to have a unit euclidian norm to improve its robustness to global change of contrast (or "affine change in illumination") inside the image. Finally, values greater than 0.2 will be clamped to 0.2, and then the descriptor is L_2 normalized a second time. This final post-processing step allows the descriptor to deal with some non-affine change in illumination, which is desirable because two different patches that correspond to the same regions of a unique object must have exactly the same semantic representation even if the contrast is not the same in the two patches. In a nutshell, a patch just has to have a sufficiently high contrast in order to be meaningful.

6. **Explain why SIFT is a reasonable method to describe a patch of image when doing image analysis.**
   SIFT is not too computationally expensive and is a reasonable method to describe a patch with simple vectors between which it's very easy to compute a distance in order to compare them. Moreover, SIFT descriptors are robust to rotation, translation and illumination change. Finally, they just take into account the points of interest while ignoring the areas that don't provide significant information. This results in a sparse and spatial encoding.

7. **Interpret the results you got in this section.**
   It is hard to understand how a patch is built only by looking at the SIFT's descriptors. As we can see in this next image, even when the variations in the patch are not complex, it is hard to tell how the patch looks like only by peaks on the descriptor.

   Still, we can recognize breaks in the image that are constant in gradient modulus and orientation on the descriptor when peaks are of same size and evenly spaced.

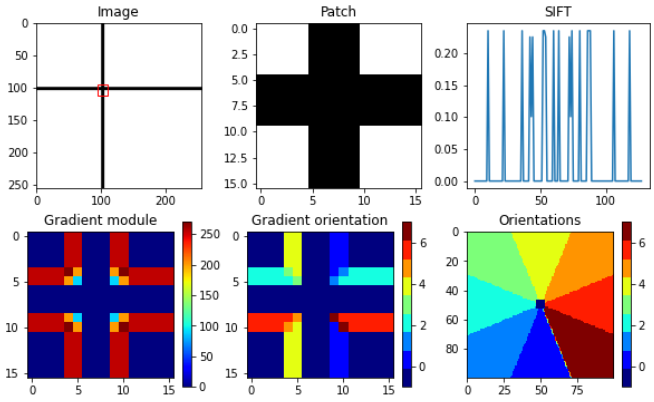
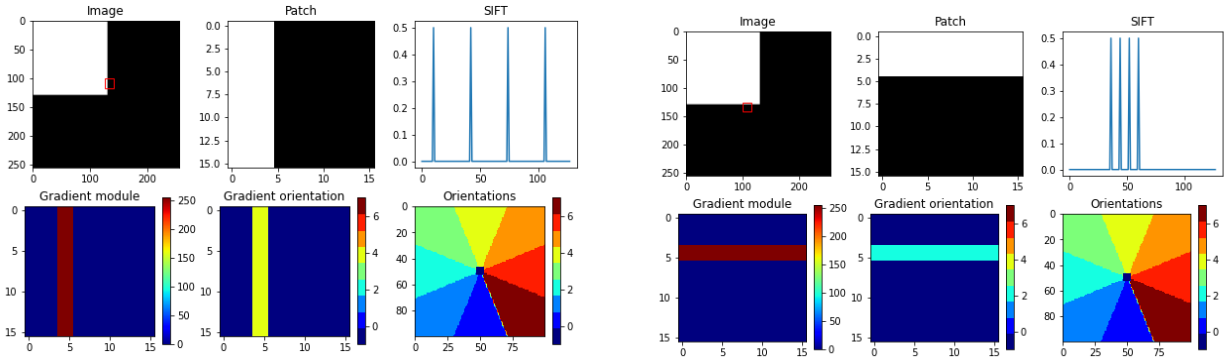


Figure (1) Graphs and SIFT descriptor obtained from a patch



(a) (b)

Figure (2) Example of patches containing a vertical break (a) and a horizontal break (b)

2 Visual dictionary

8. **Justify the need of a visual dictionary for our goal of image recognition that we are currently building.**
   We need to know what we can expect to recognize in our images. To do so, we have to do a clustering of all SIFT vectors to find words. Such a word is the descriptor that is the most representative of the descriptors belonging to a same class. Consequently, the visual dictionary is needed because the classification of a SIFT vector is done by assigning it to the corresponding class of the nearest word.

9. **Considering the points $\{x_i\}_{i=1..n}$ assigned to a cluster c, show that the cluster's center that minimize the dispersion is the barycenter (mean) of the points x_i:**

$$\text{Dispersion}(c) = \min_c \sum_i \|x_i - c\|_2^2$$

If the dispersion of the set of points belonging to the cluster c is minimized, then we have:

$$
\begin{aligned}
& \frac{\partial \text{Dispersion}(c)}{\partial c} = 0 \\
\Leftrightarrow \quad & \frac{\partial \sum_{i=1}^{n} \|x_i - c\|_2^2}{\partial c} = 0 \\
\Leftrightarrow \quad & \frac{\sum_{i=1}^{n} \partial((\sum_{k=1}^{n'} (x_{i,k} - c_k)^2)^{\frac{1}{2}})^2}{\partial c} = 0 \\
\Leftrightarrow \quad & \frac{\sum_{i=1}^{n} \partial(\sum_{k=1}^{n'} (x_{i,k} - c_k)^2)}{\partial c} = 0 \text{ where } n' \text{ is the number of components.} \\
\Leftrightarrow \quad & \frac{\sum_{i=1}^{n} \partial(x_{i,k} - c_k)^2}{\partial c_k} = 0 \\
\Leftrightarrow \quad & \sum_{i=1}^{n} -2(x_{i,k} - c_k) = 0 \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{i,k} = c_k \sum_{i=1}^{n} \\
\Leftrightarrow \quad & \sum_{i=1}^{n} x_{i,k} = c_k \cdot n \\
\Leftrightarrow \quad & c_k = \frac{1}{n} \sum_{i=1}^{n} x_{i,k}
\end{aligned}
$$

10. **In practice, how to choose the "optimal" number of clusters?**
    First of all, we need to keep in mind that for a specific dataset there is not necessarily a unique and obvious number of possible clusters. For instance, the clustering can be hierarchical; the eventual overlap between two clusters can be defined as a cluster, etc. It is also important to mention that generally we also want the number of cluster to be as low as possible because it's computationally expensive to deal with a high number of clusters.

    To choose the optimal number of clusters, there are two possibilities: either the number of cluster is known in advance, either we need to determine it experimentally. To do so, several techniques exist. Among the most commonly used, there are: the elbow method, the silhouette coefficient and the Gap statistic. Roughly speaking, the optimal number of clusters is the smallest one such that if we increase it, we don't obtain significant decrease of the intra-cluster distances.

11. **Why do we create a visual dictionary from the SIFTs and not directly on the patches of raw image pixels?**
    The clustering must be robust to small changes and perturbations applied on the image. Therefore, if we use directly the patches of raw image pixels, the clustering would be very unstable. On the contrary, a clustering done in the SIFT domain would be much more stable since the SIFT descriptors are invariant to such transformations, as we mentioned above.

    Moreover, the dimensionality of the space in which our vectors (describing our patches) are embedded has to be as low as possible in order to minimize the computational cost.

12. **Comment the results you get.**
    Results are satisfying as the images in the clusters shown seem very similar for the most part. However, when using a low number of clusters, some groups of images show pretty different images that have in common only a general scheme (grids for example). This happens much less when we use a big number of clusters.

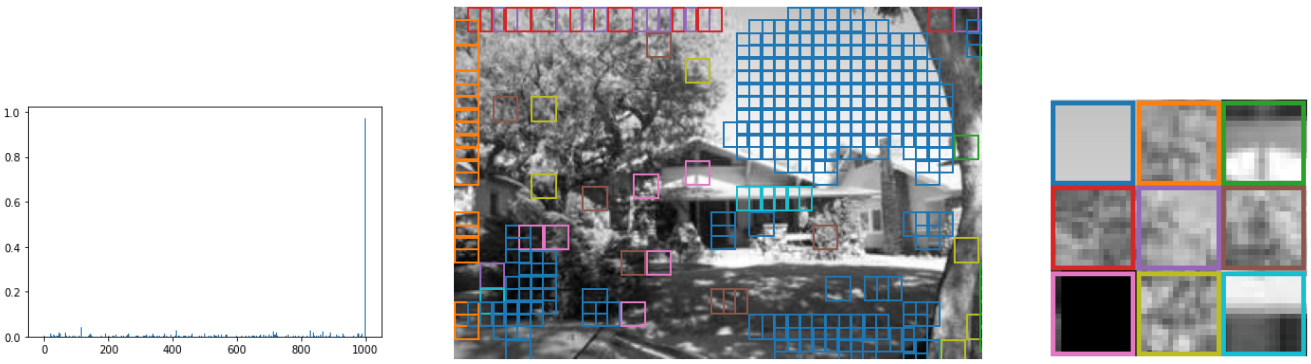
Figure (3) List of patches linked under a same word.

3 Bag of Words

13. **Concretely, what does the vector z represent of the image?**
z is an encoding of the image obtained by dividing, for each possible word, the number of times it was detected divided by the total number of patches of the image. In other words, z is a representation of the image based on the frequency of detection of the features that can possibly appear in the image.

14. **Show and discuss the visual results you got.**

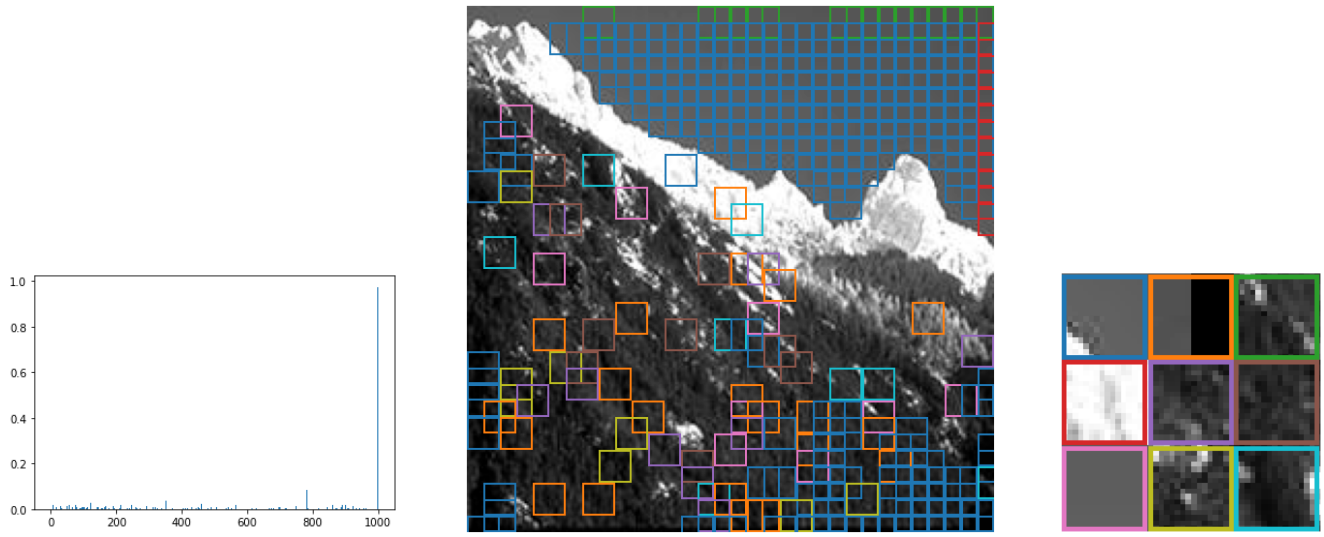
It appears our results are not perfect which is probably due to the fact that we had to limit the number of images used to reduce computation time. However, we can still analyse them.



(a) (b) (c)

Figure (4) (a) Repartition of the patches in the words, (b) an image and its most recognized words and (c) some patches

The first thing we note is that there almost always is a word that is widely found in the image. This word always corresponds to the plain patch. In most pictures we can find a plain wall, the sky or even an under/overexposed part of the image that is of constant color. As SIFT uses the gradient, we can expect those patches to be described the same way. This word corresponds to the null word we added for this purpose.



(a) (b) (c)

Figure (5) (a) Repartition of the patches in the words, (b) an image and its most recognized words and (c) some patches

As for other words, they seem to be evenly distributed. Even though some words seem pretty similar to the human eye, they are different when considering their SIFTs. As such, patches among a word are similar. Moreover, we experimented with 1000 clusters so we could expect to not find 1000 very different words.

We also note that some words are defined by the borders of the image. Those words can be then found inside the image when there is a vertical break in the patch.

15. **What is the interest of the nearest-neighbors encoding? What other encoding could we use (and why)?**
Nearest-neighbors encoding allows to assign each patch to a single word of the dictionary and then group comparable descriptors. The advantages of this encoding are the ease with which it can be implemented, and its robustness to small transformations applied to the descriptors.

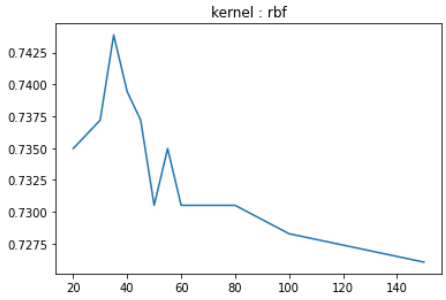
A more flexible solution would consist of doing a soft-assignment. As we saw in the fourth class of Matthieu Cord, several strategies exist (plausibility, uncertainty; cf slides 35 and 36). They are all based on the application of a kernel (typically an exponential one) on the distance between the descriptor and every centroid. Another way of doing a soft-assignment is to sparsely encode the descriptors (cf. slides 38 and 39).

16. **What the interest of the sum pooling? What other pooling could we use (and why)?**
The sum pooling keeps a lot of information in the image as we keep information on every feature in the image. Alternatively, we could do a max-pooling to just keep the feature that is the most present in the image. This would attenuate the noise effect.

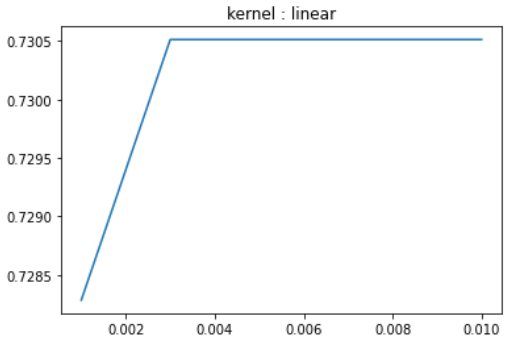
17. **What is the interest of the L_2 normalization? What other normalization could we use (and why)?**
The L_2 normalization is useful to make sure that the encoding of an image doesn't depend on the number of features present in it. We could also use the L_1 norm to limit the impact of the big values on the normalization.

4 SVM classifier

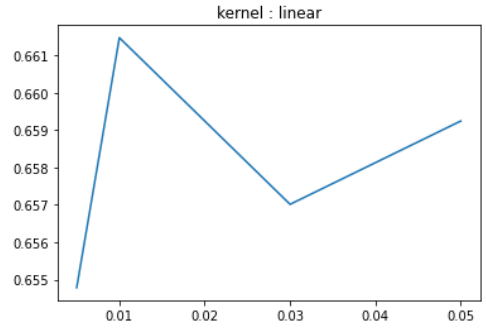
18. **Discuss the results, plot for each hyperparameters a graph with the accuracy in the y-axis.**



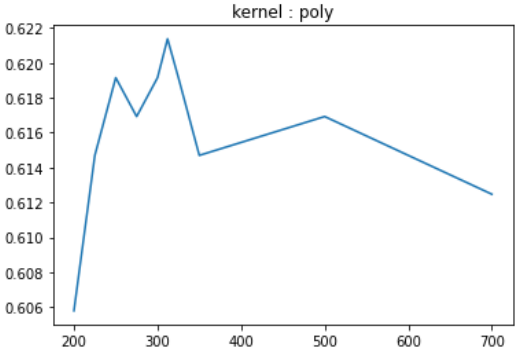
Accuracies obtained with the RBF kernel depending of C



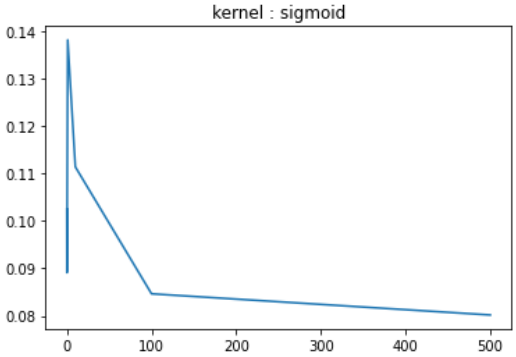
Accuracies obtained with the linear kernel depending of C



Accuracies obtained with the linear kernel but with a one vs all strategy, depending of C



Accuracies obtained with the polynomial kernel depending of C



Accuracies obtained with the Sigmoid kernel depending of C

First we tried the different kernel types. We found that for this data, the sigmoid kernel does not work at all. Whatever we set for the other hyperparameters, the score on the validation set does not go beyond 0.14.

The polynomial kernel is more interesting with scores on the validation set reaching 0.62. It is optimal when C is equal to about 312. The following graph was generated using the 'scale' kernel coefficient, which shows similar results as the 'auto' coefficient.

The second best kernel is the linear kernel using which reaches a score of 0.73. Changing C has pretty little impact as the score does not change with C greater or equal to 0.003.

Finally, the best kernel is the Radial Basis Function (RBF) kernel as its scores reach a bit more than 0.74.

All these scores were generated with the *one vs one* strategy. When we try to use the *one vs all* strategy with linear kernel, we find that the results really depend on the starting points which are chosen randomly.

19. **Explain the effect of each hyperparameter.**

The regularization parameter λ (or "C") It determines the trade-off between increasing the margin size and ensuring that the points lie on the correct side of the margin. In other words, it controls the trade off between achieving a low training error and a low testing error which is related to the ability to generalize to unseen data.

Kernel type Another hyperparameter is the kernel type. Kernels allow to transform the input space such that it becomes the most separable as possible after this transformation. Instead of computing the dot product $\langle \phi(X), \phi(W') \rangle$ of each pair of data points projected in a higher dimensional space, we just use a less costly function $K = \langle \phi(X), \phi(W') \rangle$ that is called a kernel. In *scikit-learn*, four kernels are available:

(a) The RBF is defined as $K(X, W') = \exp\left(-\gamma \cdot \|X - W'\|^2\right)$ where $\gamma > 0$ defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. If is too large, the radius of the area of influence only includes the support vector itself. Otherwise, it is very small, the boundary is not complex enough.

(b) The linear kernel is defined as $K(X, W') = \langle X \cdot W' \rangle$. This kernel is pretty popular as it is very effective with linearly separable data.

(c) The Sigmoid kernel is defined as $K(X, W') = \tanh\left(\langle X, W' \rangle + r\right)$. It is the first kernel to use when we don't really know what is the data like as it is the most efficient with non linearly separable data.

(d) The polynomial kernel is defined as $K(X, W') = \left(\gamma \cdot \langle X, W' \rangle + r\right)^d$ where $\gamma > 0$. Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. d is the degree while r is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial. The effect of γ is the same as the RBF kernel's γ.

Each kernel type is more adapted for certain data types. For example, as we saw in question 18, a Sigmoid kernel does not work for our data. In our case, the most efficient was the RBF kernel.

20. **Why the validation set is needed in addition of the test set ?**
The validation set is used to optimize the hyperparameters. We need this new set because the scores on the training set are already biased by the learning. Moreover, tuning the hyperparameters would bias the scores we get on the test set which is not desirable if we want to compare models.