RDFIA Homework-1 :

Image representation with SIFT encoding, Bag of Words and SVM classifier

Andrew Caunes

ZHAO Yunfei

October 15, 2021

In this homework we aim at programming and testing an image representation using a Bag-of-Word method with a dictionary of SIFT encodings of local patches. We then use a support vector machine to classify images in the 15-Scenes dataset.

Contents

1 SIFT **0**
1.1 Computing the gradient of an image 0
1.2 Computing the SIFT representation of a patch 0

2 Visual Dictionary **3**

3 Bag of Words **5**

4 SVM classifier **8**

List of Figures

1 left: pixel gradient norm right: pixel gradient orientation 1
2 Illustration of SIFT encoding of a selected region in an image. The red rectangle in the first picture on top is the region we select and it is zoomed in the second picture on top. The third picture on top shows the values of our 128 dimensions encoded vector for the 16×16 patch. We have 16 mini patches of 4×4 for each patch and each mini patch is encoded into a vector of dimension 8 and these are 8 discretized orientation showed in the third figure on the bottom. Then these 16 vectors are concatenated to a vector of dimension 128. The first and second figure are respectively tho norm and orientation of pixel gradients. 1
3 left: patch corresponds to center vector of no gradient right: random regions . 5
4 Top 100 patches for 4 centers, we can see relevant patterns appearing 5
5 Illustration of the presentation of an image by bag of words with "nearest-neighbors" encoding 6
6 Bag of Word encoding using euclidean distance instead of nearest-neighbor . . . 8
7 Parallel coordinates view of hyper parameters selection for SVM 9
8 Scatter plot matrix view of hyper parameters selection for SVM 10

1 SIFT

1.1 Computing the gradient of an image

To compute the gradient of an image at a pixel (x, y):

$$G(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}^\top = \begin{bmatrix} I_x & I_y \end{bmatrix}^\top \tag{1}$$

We use Sobel kernal for a convolution of the image.

$$I_x = I * M_x \qquad\qquad I_y = I * M_y$$

$$M_x = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad\qquad M_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2}$$

Questions

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

Let

$$h_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad\qquad h_y = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

We have $Mx = h_y h_x^T$ and $My = h_x h_y^T$.

2. Why is it useful to separate this convolution kernel?

We have $I * Mx = I * h_y * h_x^T$, but convolving according to the right term uses 6 products instead of 9 as respect to the term on the left, thus it is more computationally efficient. This stems from the fact that the matrices are rank 1.

1.2 Computing the SIFT representation of a patch

In practice

An example of pixel gradient norm and pixel gradient orientation is showed in figure 1. An example of SIFT encoding of a selected patch of pixel in an image is showed in figure 2.

Questions

3. What is the goal of the weighting by gaussian mask?

We assume that the important content to see in a given patch will be in its center. If some interesting area is not centered in one patch, it will often be centered in the next patch.

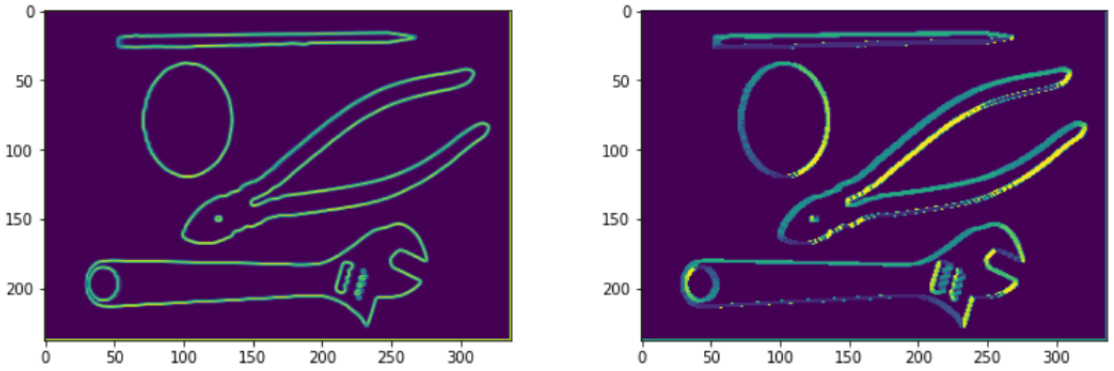


Figure 1: left: pixel gradient norm right: pixel gradient orientation

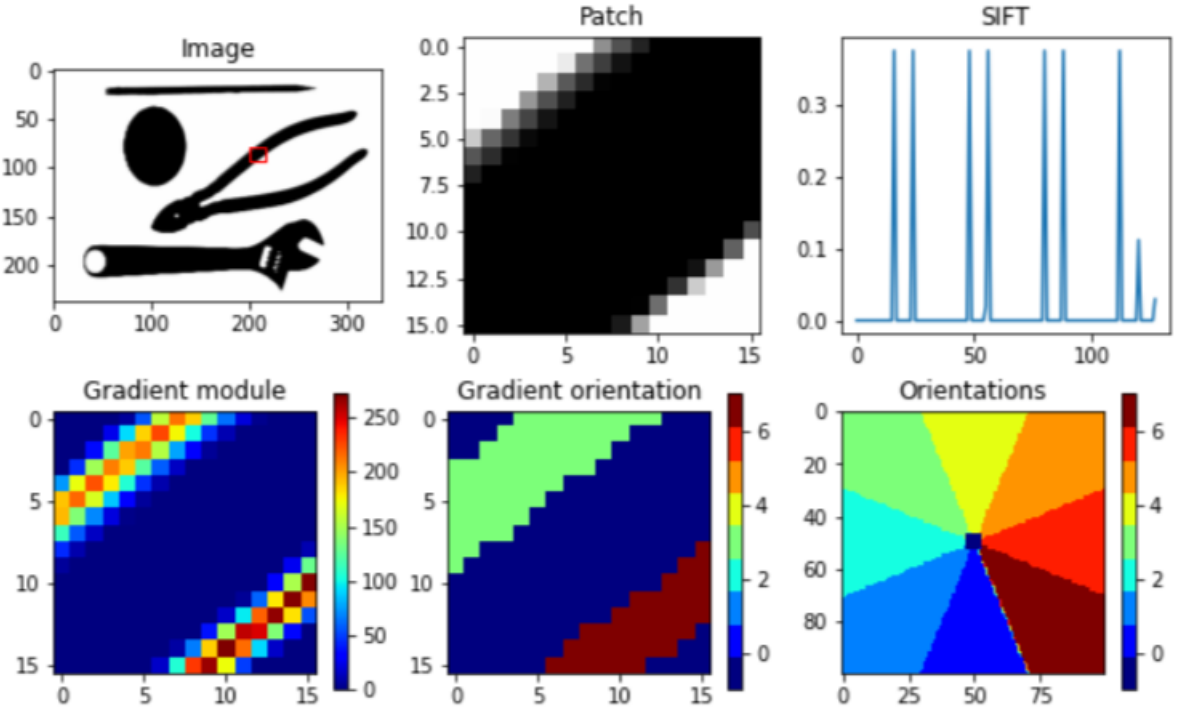


Figure 2: Illustration of SIFT encoding of a selected region in an image. The red rectangle in the first picture on top is the region we select and it is zoomed in the second picture on top. The third picture on top shows the values of our 128 dimensions encoded vector for the 16×16 patch. We have 16 mini patches of 4×4 for each patch and each mini patch is encoded into a vector of dimension 8 and these are 8 discretized orientation showed in the third figure on the bottom. Then these 16 vectors are concatenated to a vector of dimension 128. The first and second figure are respectively tho norm and orientation of pixel gradients.

4. Explain the role of the discretization of the directions.

There are infinitely many directions, so we need to discretize them in order to store the gradient information in a compact vector.

5. Justify the interest of using the different post-processing steps.

We don't want to keep the less informative data, that is when the patches contain little to no change, so we set their encoding to zero.
We normalize the SIFT vectors so that they all contribute the same for classification.
We also clip the smaller values in each SIFT vector since this information tends to be less informative and coming from noise.

6. Explain why SIFT is a reasonable method to describe a patch of image when doing image analysis.

The SIFT encoding allows us to identify the patches of the image which contain more variation like the corners and borders, which tend to be useful to identify objects and other important structure in images.

7. Interpret the results you got in this section.

The encoding seems effective in detecting corners and borders in the image on simple images with high contrast and clear lines. We can now use it to encode an entire image and later use the extracted features in a Bag of Word setting.

1.3. Computing SIFTs on the image dataset

In practice

```
def compute_sift_image(I):
    x, y = dense_sampling(I)
    im = auto_padding(I)
    m = gaussian_mask()
    patch_size=16

    # Here, compute on the global image (norm, gradients)
    Gn, Go = compute_grad_mod_ori(im)

    sifts = np.zeros((len(x), len(y), 128))
    for i, xi in enumerate(x):
        for j, yj in enumerate(y):
            Gn_patch = Gn[xi : xi+patch_size, yj : yj+patch_size]
            Go_patch = Go[xi : xi+patch_size, yj : yj+patch_size]
            sifts[i, j, :] = compute_sift_region(Gn_patch, Go_patch, m)

    return sifts
```

2 Visual Dictionary

```python
def compute_visual_dict(sift, n_clusters=1000, n_init=1, verbose=1):
    # reorder data
    dim_sift = sift[0].shape[-1]
    sift = [s.reshape(-1, dim_sift) for s in sift]
    sift = np.concatenate(sift, axis=0)
    # remove zero vectors
    keep = ~np.all(sift==0, axis=1)
    sift = sift[keep]
    # randomly pick sift
    ids, _ = compute_split(sift.shape[0], pc=0.05)
    sift = sift[ids]

    kmeans = KMeans(n_clusters=n_clusters).fit(sift)
    # add cluster for zeros
    vdict = np.vstack((kmeans.cluster_centers_, np.zeros(dim_sift)))

    return vdict
```

Questions

8. Justify the need of a visual dictionary for our goal of image recognition that we are currently building.

We want to extract relevant features from the images which will allow us to classify the image accurately. Using the raw data of pixels or even all SIFT vectors from each patch of each image would be too much data and make the learning step pretty slow. This is why we process the data by using the SIFT encodings and by extracting some structure from them with clustering. We obtain features that will be less exhaustive but more relevant and easier to learn from. We hope that this structure for the data will be representative of various features of the images such as objects, shapes which are simple patterns, better generalized with less variance.

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 :

$$\min_c \sum_i \|x_i - c\|^2$$

We have :

$$\begin{aligned}\frac{\partial}{\partial c}\sum_{i=1}^{n}\|x_i - c\|^2 &= 2\sum_{i=1}^{n}(x_i - c)\\ &= 2(\sum_{i=1}^{n}x_i - nc)\\ &= 0\\ \iff c &= \frac{1}{n}\sum_{i=1}^{n}x_i\end{aligned}$$

Therefore the barycenter of the points x_i is the center that minimizes the dispersion.

10. In practice, how to choose the "optimal" number of clusters?

It's an hyperparameter that we can set using a validation set during training.

11. Why do we create a visual dictionary from the SIFTs and not directly on the patches of raw image pixels?

The SIFTs encode potentially relevant information by focusing on patches of images with more variation, such as corners and borders. The features we cluster with SIFTs already containe somewhat high level information that can help to classify images, while raw pixels only contain very low level information about a pixel's color. A visual dictionary using the pixels would probably only differentiate between the colors which wouldn't help classifying images. For example a little change on image or a little change of brightness will cause a great change on raw pixel.

12. Comment the results you get.

In figure 4, we can see that the clustering found some structure in the SIFT patches, for example identifying some windows or barriers, which will be useful to classify images. Some of the clusters seem less relevant as they only show blank patches or seemingly unrelated ones.

In figure 4, we select 4 centers of vector of SIFT and we show the top 100 most related patches and we see that they are all showing similar shapes and structures.

Figure 3: left: patch corresponds to center vector of no gradient right: random regions

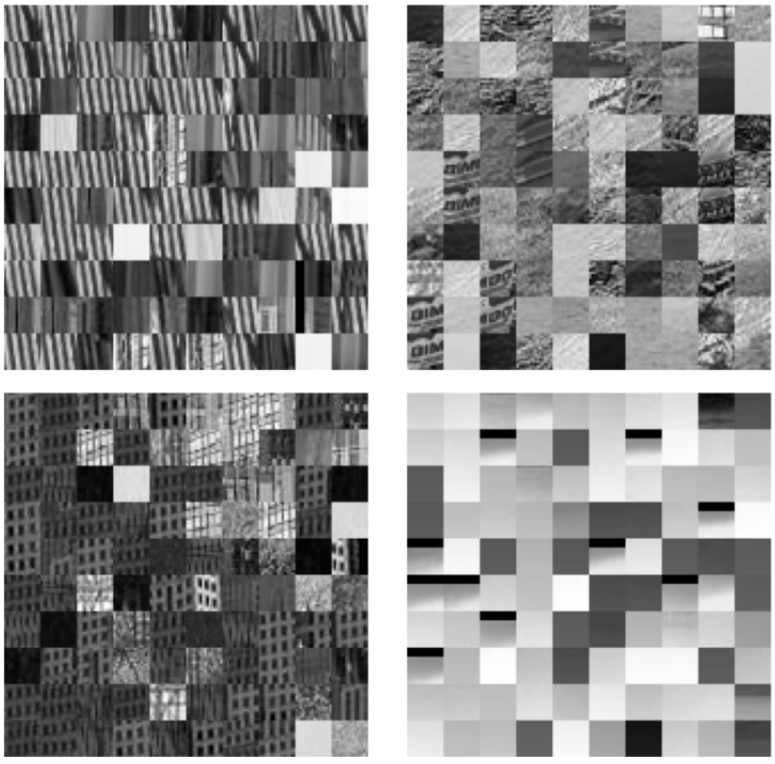


Figure 4: Top 100 patches for 4 centers, we can see relevant patterns appearing

3 Bag of Words

In practice

```
def compute_feats(vdict, image_sifts):
    [...]
    # compute distance from image sifts to vdict
    for sift in sifts:
```

```
        ind = np.argmin(distance_matrix(vdict, sift.reshape(1,128)))
        feats[ind] += 1
    feats = feats/np.linalg.norm(feats)

    return feats
```

Questions

13. Concretely, what does the vector z represent of the image?

For some image, the vector z represents for each "word" in the visual dictionary the number of patches that are closest to that word. z should thus approximate a vector of occurrences of each word of the dictionary in the image.
It is only an approximation since some words (SIFT of a patch) in an image may not have a definition in the dictionary.

14. Show and discuss the visual results you got.

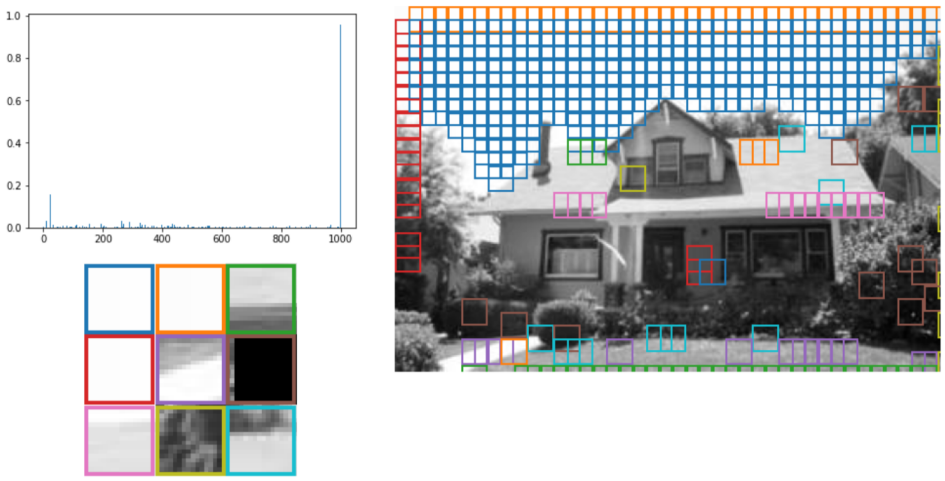


Figure 5: Illustration of the presentation of an image by bag of words with "nearest-neighbors" encoding

In the image **??** the sky seems to have been correctly identified as some word, as well as part of the border of the roof and other high level features. This is promising because we can imagine some classifier taking advantage of these features to accurately identify buildings for example.
It seems however that some relevant features were not identified as efficiently, such as the windows of the house. This might be addressed by using more training data or modifying various hyperparameters like the size of the patches.

15. What is the interest of the nearest-neighbors encoding? What other encoding could we use (and why)?

The NN encoding means we choose to attribute one specific word to each patch of the image. Thus we assume that each word in the image (SIFT of a patch) has a corresponding definition in the dictionary and that its definition is the closest in L2-norm to it.
This might not be the case and we could use a simple euclidean distance encoding instead, storing in z the cumulative sums of the distances of the words in the image to each word in the dictionary.
Using the same notation as for the NN encoding :

$$h_i \in \mathbb{R}^M \qquad \hat{z} \in \mathbb{R}^M$$

$$h_i[j] = -\|x_i - c_j\| \qquad \hat{z} = \sum_{i=1}^{n_{patch}} h_i$$

$$z = \frac{\hat{z}}{\|\hat{z}\|}$$

The higher $z_i \in \mathbb{R}_-$, $i \in \{1 \dots M\}$, the more there are words close to the i-th center of the dictionary in the image. This gives a continuous metric of how many of each words in the dictionary there are in one image.
See one implementation and the results below :

```
def compute_feats(vdict, image_sifts):
    ...

    # compute distance from image sifts to vdict
    for sift in sifts:
        dist = distance_matrix(vdict, sift.reshape(1,128)).reshape(1001,)
        feats += -dist
    feats = feats / np.linalg.norm(feats)

    return feats
```

16. What is the interest of the sum pooling? What other pooling could we use (and why)?

Sum pooling allows to reduce the number of features by combining all the h_i vectors, without loosing information (unless we include a feature of the positions of the patches).
There are other types of pooling such as mean pooling (which would produce the same result here) or max pooling, which would keep only the word occurring the most in each image, thus loosing a lot of information in the process. Sum pooling is probably the best option here since we already have rather simple features to deal with (frequencies of occurrence of words).

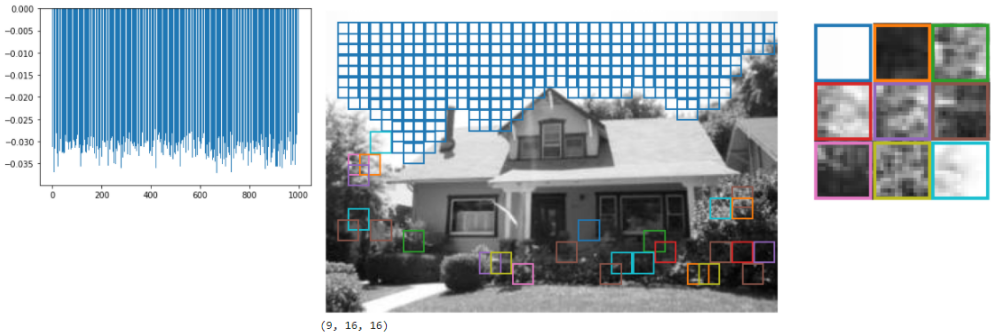


Figure 6: Bag of Word encoding using euclidean distance instead of nearest-neighbor

17. What is the interest of the L2 normalization? What other normalization could we use (and why)?

The L2-Normalization is used in order to make each Bag of Word vector similarly influential towards a classifier. For example we can reduce the influence caused by the size of image, etc.
We could use other norms to do this such as the L1-norm, which would reduce the weight of vectors with uniformly distributed words, or even the $L\infty - norm$ which would give less weight to vectors with one coordinate (word) standing out above the others.

4 SVM classifier

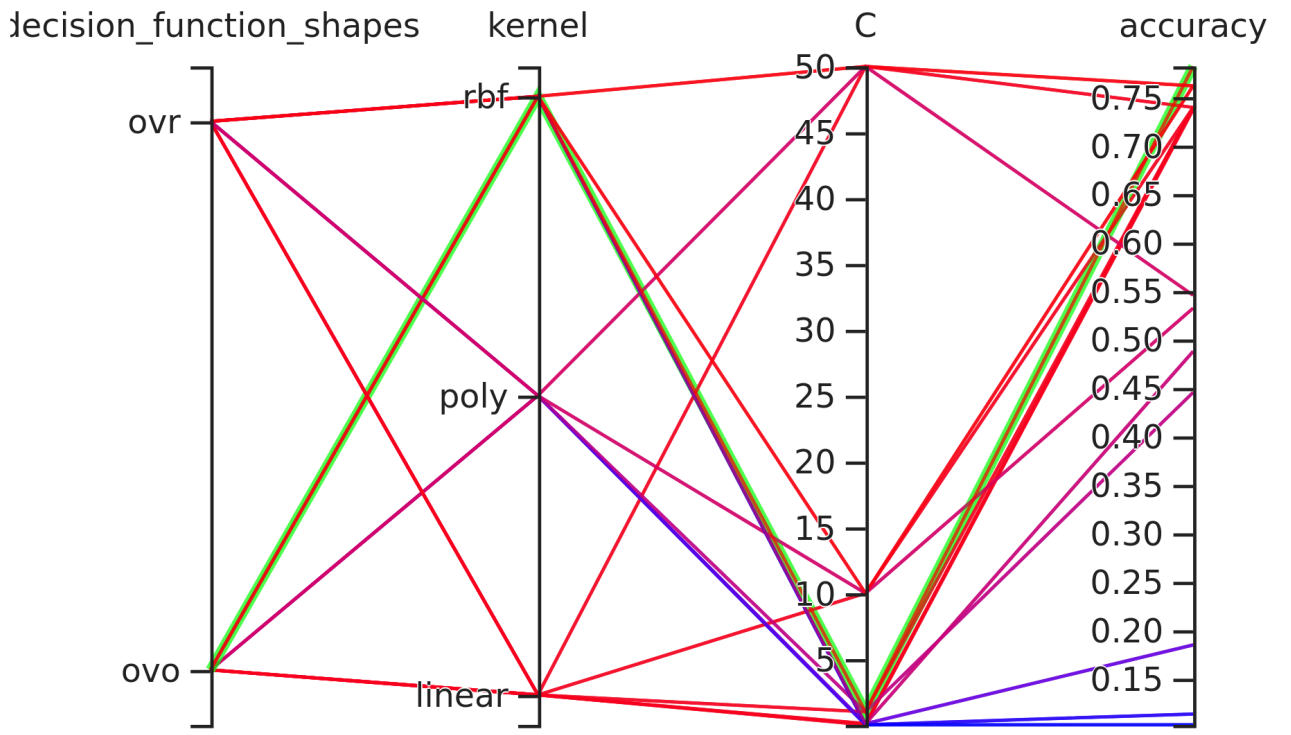
In this final part, we finish our pipeline with a support vector machine classifier. From the previous sections, we transform each image into a vector of dimension 1001. And we are going to train our final classifier by the following steps:

1. Split dataset into a **training set**, a **validation set** and a **test set**.
2. By training the model with different hyper-parameters and evaluating it on validation dataset, we select the optimal hyper-parameters.
3. We train the model with the selected hyper-parameters on both the **training set** and the **validation set** as our final model.
4. We evaluate the final model on the **test set**.

We split the original dataset into **train set**, **validation set**, **test set** with respect to ratio 70%, 10%, 20%. As for hyper-parameters, We choose decision function for multi-class of SVM between ***OVR: one vs rest*** and ***OVO: one vs one***. Then, for value ***C***, we choose value from **0.01, 0.1, 1, 10, 50**. Finally, We also try different kernel among ***Radial Basis Function (RBF) kernel, Linear kernel, polynomial kernel***.

From figure 7 we can conclude that the best hyper parameters we obtain from validation dataset is:

- decision function: one vs one



Trial ID	decision_function_shapes	kernel	C	accuracy
1634058471.29...	ovo	rbf	1.0000	0.78174

Figure 7: Parallel coordinates view of hyper parameters selection for SVM

- kernel: RBF
- c: 1

The SVM model with these parameters achieve an accuracy of 0.78 on validation dataset.

Now we use these hyper parameters to train a new SVM model on the whole train dataset and make an evaluation on test dataset.

The final evaluation score is: **0.7324414715719063**

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

We can see from figure 7 and figure 8 that for this dataset, the types of kernel and the value of C have a bigger influence on result than the type of decision.

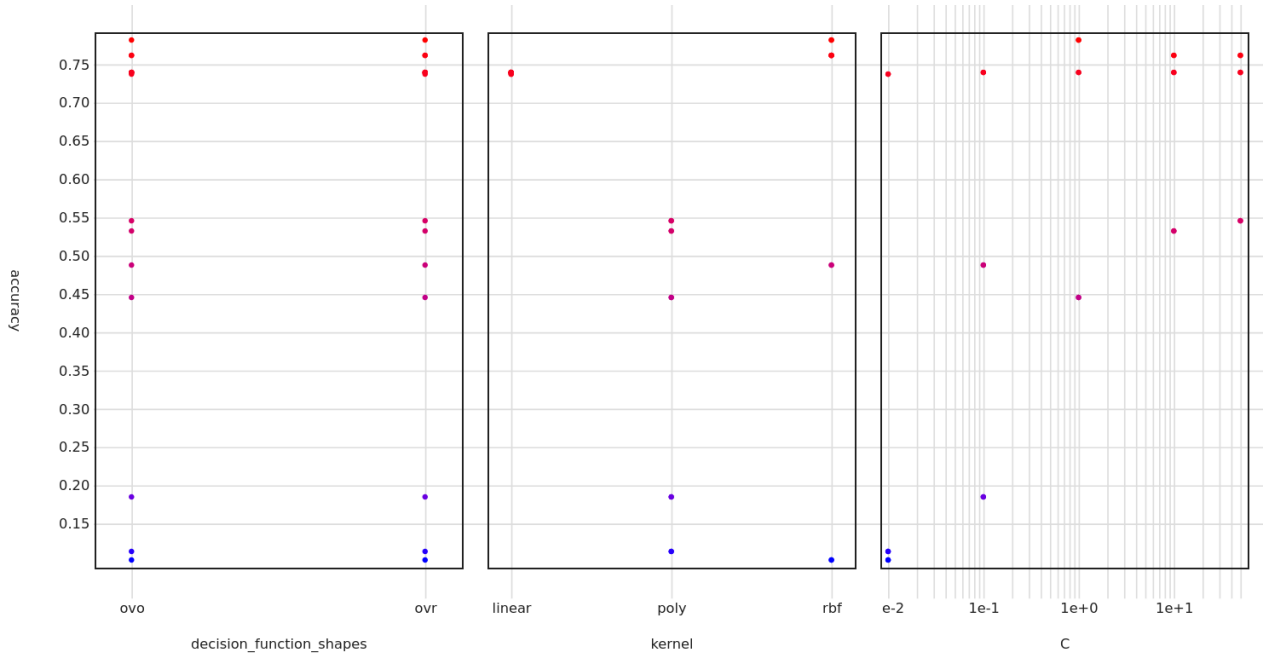


Figure 8: Scatter plot matrix view of hyper parameters selection for SVM

19. Explain the effect of each hyper parameter.

Firstly, For decision function of multi-class classification problem, the OVO and OVR help to extend a binary classifier to multi-classifier, OVO will create a classifier for each couple of class and the result is the class with the highest score and OVR will create a classifier for each class versus all the others.

Secondly, the kernel help to present the variable in a new space, and in this space, the sample which can not be linearly separated in original space might be able to be separated linearly there and the new space could be of infinite dimension and it is efficient to calculate by kernel.

Thirdly, the C is a variable to balance the margin of hyper plan and misclassification on training data. With larger C value, the classifier will try to separate the samples with less misclassifications and a smaller-margin hyperplane which may lead to over-fitting and may be influenced by some noise samples.

20. Why the validation set is needed in addition of the test set ?

In supervised learning, one of the most important rule is that none of the test samples should be used during training process, even for hyper parameters selection. So we split validation set from training set to help us select our hyper parameters for SVM and then the final evaluation is done on the test set.