TP LDA

Rool Sara, 5 ModIA

June 6, 2025

1 Introduction

The objective of this lab is to classify simulations of the interstellar medium using Linear Discriminant Analysis (LDA). Several statistical estimators will be employed in the process, including:

- Empirical mean
- Empirical autocovariance matrix
- Power spectrum

The performance of each method will be evaluated and compared to determine their effectiveness in distinguishing between the simulated classes.

2 Loading data

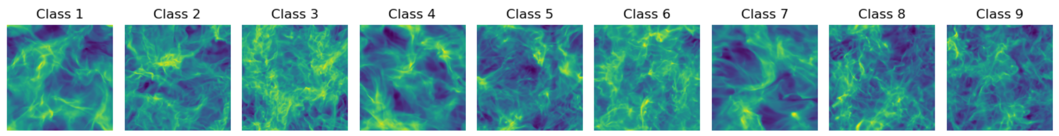

Figure 1: Display for train data

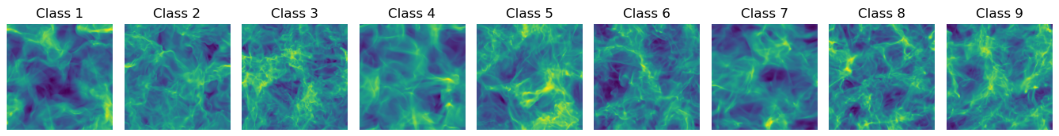

Figure 2: Display for test data

After plotting the data, it is difficult to clearly distinguish the classes, whether in the training or test sets. The textures and patterns are visually similar, and any differences are subtle and not obvious without further analysis.

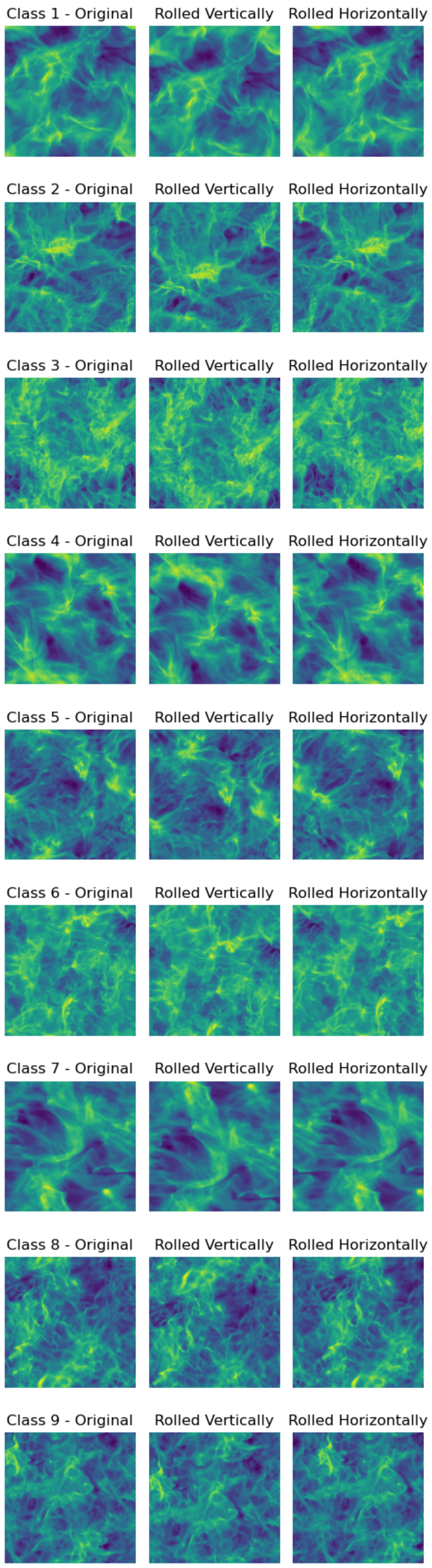

Figure 3: Display for rolled train data

When we use `np.roll` to shift the images, they still look smooth and connected, without any sharp edges. This means the image data uses periodic boundary conditions. It shows that the image stays consistent when shifted, which means it has a kind of symmetry.

3 LDA with the empirical mean

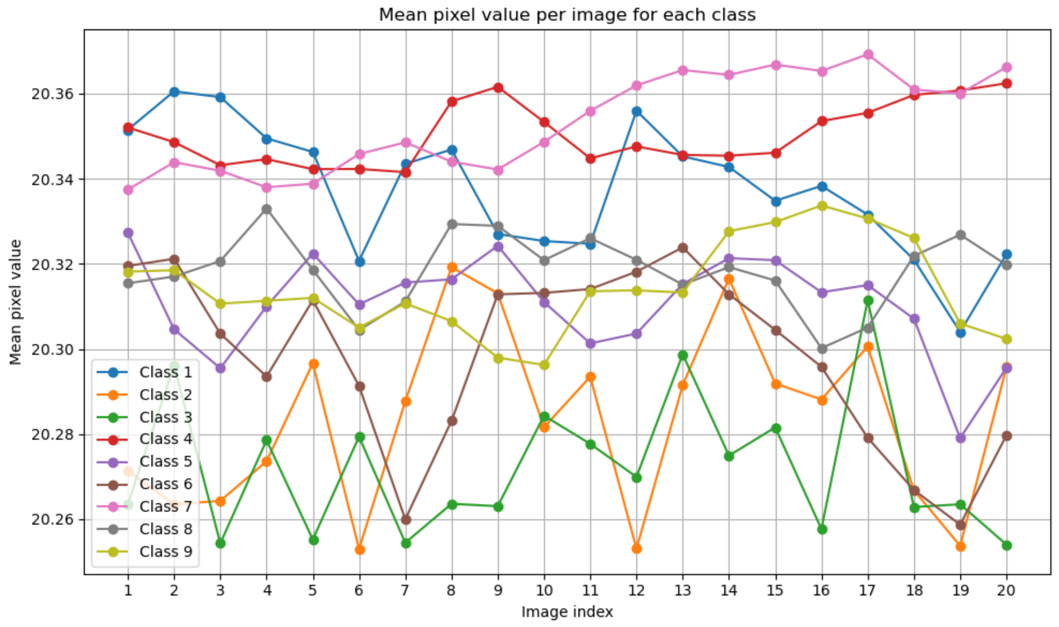

Figure 4: Mean pixel value per image for each class (train data)

The empirical mean does not appear to be a good statistical estimator for distinguishing between classes. The mean pixel values for each class are very close, ranging only between 20.26 and 20.36. This small difference makes it difficult to clearly separate the classes based on the mean alone. As a result, the separation between classes is not visually or statistically significant using this estimator.

I applied the `classify_by_LDA()` function on the mean pixel values as features. The obtained accuracies are as follows:

- Train Accuracy: 0.33
- Test Accuracy: 0.37

As we can see, these accuracy values are relatively low, around 0.3 to 0.4, which indicates poor classification performance. This result is consistent with the previous plot, where we observed that the empirical mean values are very similar across classes. Therefore, the mean is not a good feature for distinguishing between them. These findings are further supported by the confusion matrices shown below.

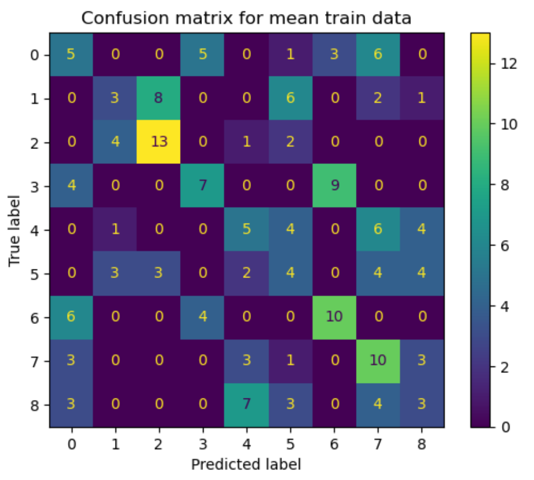

Figure 5: Confusion matrix for mean train data after LDA

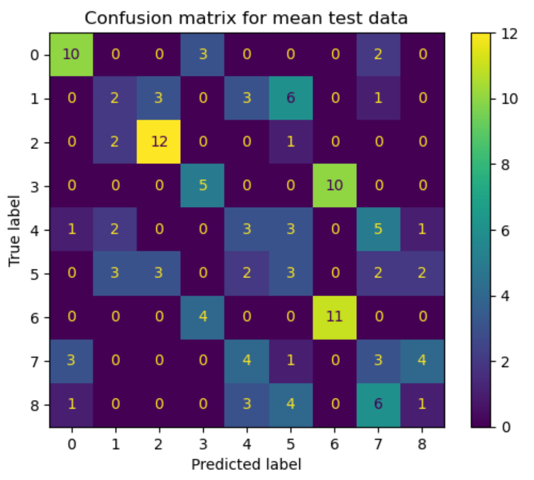

Figure 6: Confusion matrix for mean test data after LDA

4 LDA with autocovariance matrix

Let the translation vector be $\tau = (\tau_1, \tau_2)$ with $\tau_1 = 0, 1, ...dn$ and $\tau_2 = -dn, ..., -1, 0, 1, ...dn$. We note $X(u)$ the value of the image X for the pixel u. The autocovariance of the process X (assumed to be real-valued) is defined by:

$$\phi(u, \tau) = \mathbb{E}[(X(u) - \mu_u)(X(u - \tau) - \mu_{u-\tau})]$$

- What is meant by $\mathbb{E}$ in such a definition?

In this definition, $\mathbb{E}$ is the expectation taken over all realizations of the process X. It captures the average statistical relationship between two pixel values. The two positions involved are u, a 2D coordinate representing a pixel location in the image, and $u - \tau$, which is another pixel shifted by a translation vector $\tau = (\tau_1, \tau_2)$.

- What do μ_u and $\mu_{u-\tau}$ represent?

The terms $\mu_u = \mathbb{E}[X(u)]$ and $\mu_{u-\tau} = \mathbb{E}[X(u - \tau)]$ represent the local means of the image at these respective positions (u and $u - \tau$).

- Under what condition does the autocovariance depend only on τ?

The autocovariance $\phi(u, \tau)$ depends only on τ when the random process X is second-order stationary. This means that the mean value $\mu_u = \mathbb{E}[X(u)]$ is constant across all spatial locations, and the covariance between any two pixels depends only on their relative displacement τ, not on their positions in the image. In practice, images have finite size, so exact stationarity is not possible because of edge effects. To get around this, we often assume periodic boundary conditions like in our case.

- Then justify the name "autocovariance matrix" in our case.

When the autocovariance depends only on τ (which is our case due to the periodic boundaries), we can organize the values $\phi(\tau)$ for all relevant translations τ into a matrix, where each entry corresponds to the covariance between two spatial points separated by τ. That's why we call it the autocovariance matrix.

- Give its dimensions as a function of the choice of dn.

Given that $\tau_1 = \{0, 1, ...dn\}$ and $\tau_2 = \{-dn, ..., -1, 0, 1, ...dn\}$, the total number of unique translation vectors $\tau = (\tau_1, \tau_2)$ is:

$$\text{(number of } \tau_1 \text{ values)} \times \text{(number of } \tau_2 \text{ values)} = (dn+1) \times (2dn+1)$$

Thus, the autocovariance matrix has dimensions: $(dn+1) \times (2dn+1)$

The estimator implemented in the function `compute_features_cov` computes the empirical autocovariance matrix of a single image for a given maximum shift dn. This estimator is based on the assumption that the image is a realization of a second-order stationary random process. For each displacement vector $\tau = (\tau_1, \tau_2)$, where $\tau_1 \in \{0, \ldots, dn\}$ and $\tau_2 \in \{-dn, \ldots, dn\}$, the empirical autocovariance is estimated by:

$$\hat{\phi}(\tau) = \frac{1}{N_\tau} \sum_{u \in \text{valid}} \left(X(u) - \bar{X}\right) \left(X(u-\tau) - \bar{X}\right),$$

where:

- $X(u)$ is the intensity at pixel position u,
- $\bar{X}$ is the mean intensity of the image,
- $X(u-\tau)$ is the pixel value at the shifted position,
- The sum is taken over all valid pixels such that both u and $u-\tau$ are inside the image (i.e., wrapped-around values are excluded),
- N_τ is the number of valid pixels for the shift τ.

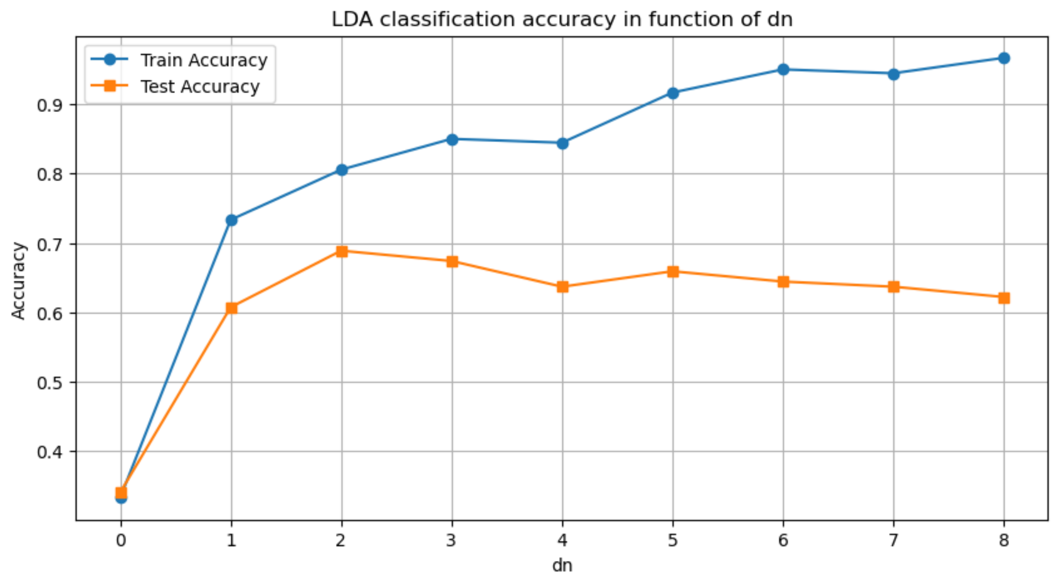

Figure 7: LDA classification accuracy in function of dn

When using the empirical mean of each image as features, the LDA classifier achieved relatively poor performance, with a train accuracy of 0.33 and a test accuracy of 0.37. This was expected, as the mean values were very similar across classes. In contrast, when using the autocovariance matrix as features, the classification results improved significantly. The training accuracy increased steadily with higher values of dn, reaching over 0.95. Although the test accuracy peaked around $dn = 2$ and then slightly decreased, it remained clearly higher than with the mean features, staying between 0.62 and 0.69. This shows that autocovariance is a much better feature for classification.

5 LDA using power spectrum (periodogram)

The autocovariance function and the power spectrum of a signal are linked through the Fourier transform. In the context of a stationary random process, the power spectral density is defined as the Fourier transform of the autocovariance function. This is the Wiener–Khinchin theorem.

Let $\phi(\tau)$ the autocovariance function, with τ a spatial displacement. The power spectral density $S(f)$ is then given by:

$$S(f) = \mathcal{F}[\phi(\tau)](f),$$

where $\mathcal{F}$ is the 2D Fourier transform, and f is the spatial frequency. Similarly, the autocovariance function can be recovered by applying the inverse Fourier transform to the power spectrum:

$$\phi(\tau) = \mathcal{F}^{-1}[S(f)].$$

By looking at the power spectrum, we propose to study values of *dom* between 20 and 80. When *dom* is below 20, the extracted region is too small and does not capture enough useful frequency information. On the other hand, values above 80 include parts of the spectrum that add little relevant information for classification.

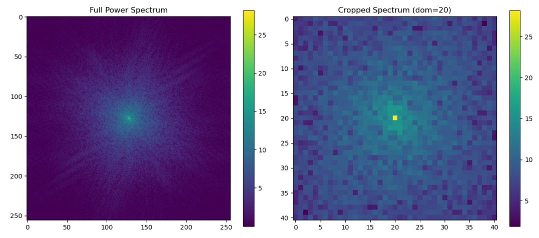

Figure 8: Power spectrum of a training image with $dom = 20$.

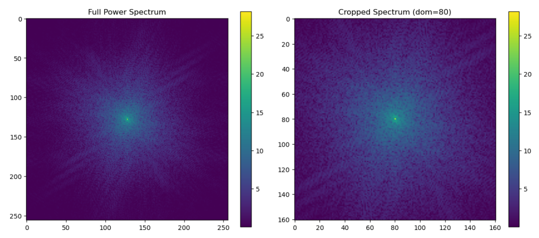

Figure 9: Power spectrum of a training image with $dom = 80$.

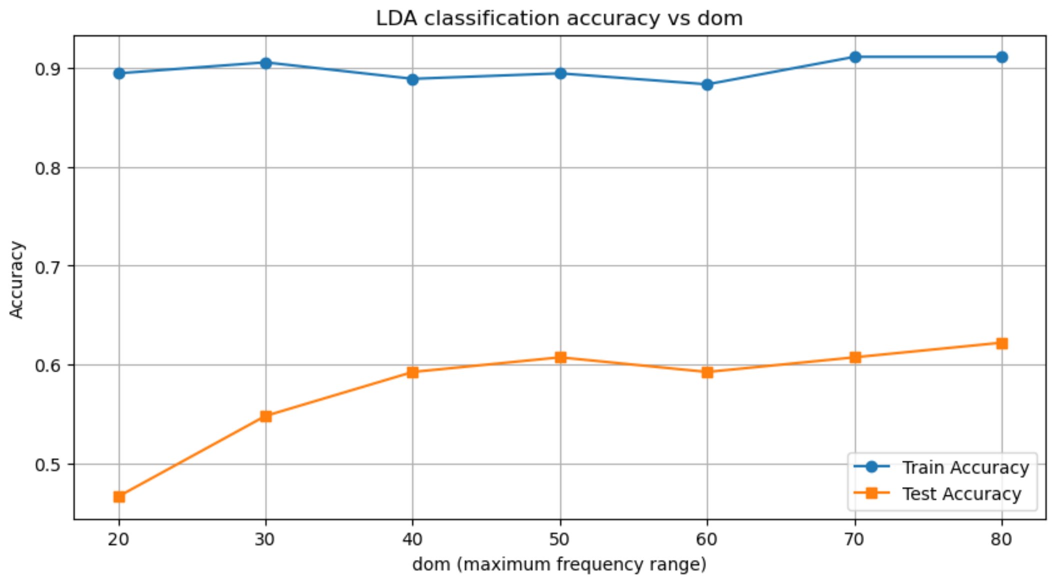

Figure 10: LDA classification accuracy in function of *dom*

The results obtained using the power spectrum as features show a significant improvement over the empirical mean approach and are comparable to, though slightly below, those obtained using the autocovariance matrix. To recall, the empirical mean yielded poor classification performance (around 33% train and 37% test accuracy). In contrast, the power spectrum leads to much higher accuracies, consistently above 90% for training and between 47% and 63% for test depending on the value of *dom*. This indicates that the power spectrum captures more informative and discriminative features related to the spatial frequency content of the images. Although training accuracy is high across all values of *dom*, the test accuracy suggests some overfitting. However, compared to the autocovariance matrix (with a training accuracy of 80% and test accuracy slightly under 70% for $dn = 2$), the power spectrum performs slightly worse.

6 Performance analysis

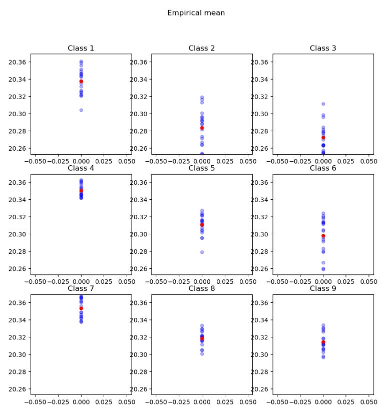

Figure 11: Empirical mean of each image per class.

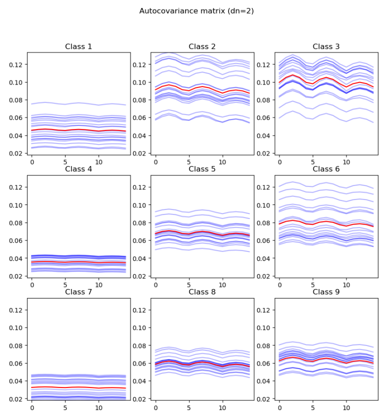

Figure 12: Autocovariance matrix of each image per class.

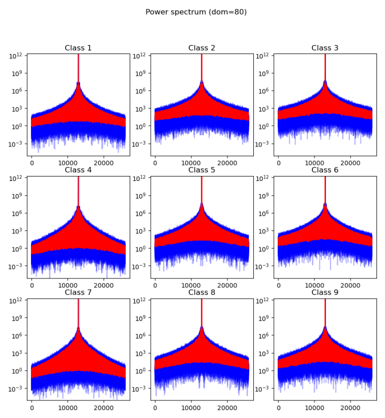

Figure 13: Power spectrum of each image per class.

The three plots made using the `plot_mean_per_class` function help to compare how well different types of features can separate the image classes. In the first plot, which shows the empirical mean, we see that the points for each class are very close to each other. This means that the average value of the features does not change much between classes, so it is not very helpful to tell them apart.

In the third plot, we see the power spectrum, which gives information about the frequency content of the images. Although this gives more detail than the mean, the shapes of the curves are still very similar across all classes. This makes it hard to clearly separate the classes based on this feature alone.

The second plot, which shows the autocovariance matrix, gives much better results. The curves are more different from one class to another. This means that the relationships between features contain useful information to tell the classes apart.

Overall, the autocovariance features are more helpful for classification than the empirical mean or the power spectrum.

These results are confirmed by the following graphs. They display the average features for each class and how much they vary.

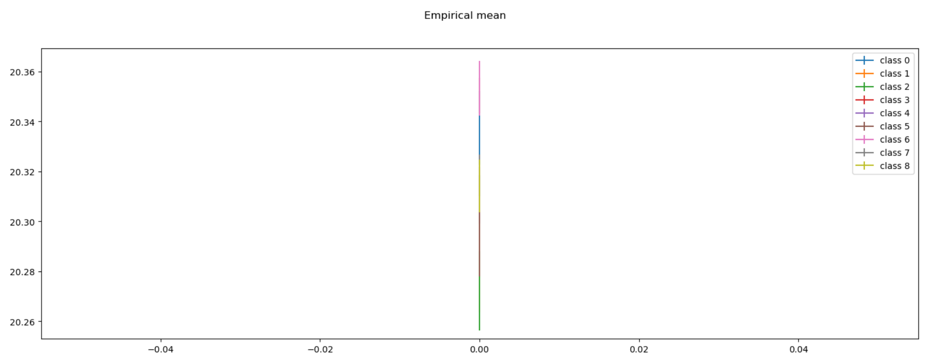

Figure 14: Mean feature vectors per class (empirical mean)

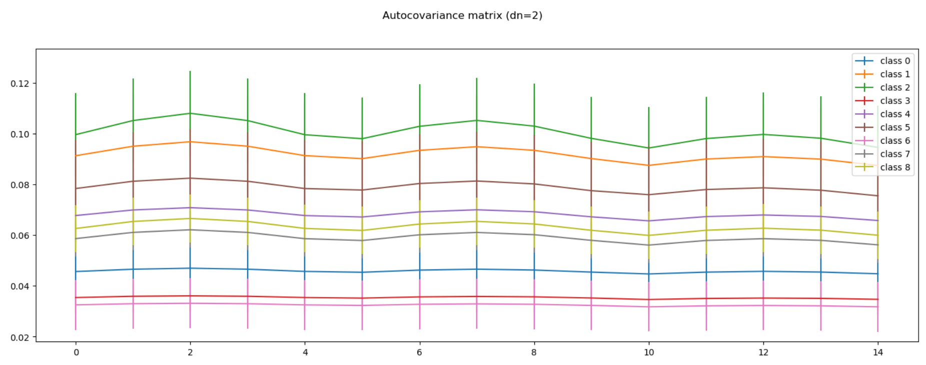

Figure 15: Mean feature vectors per class (autocovariance matrix)

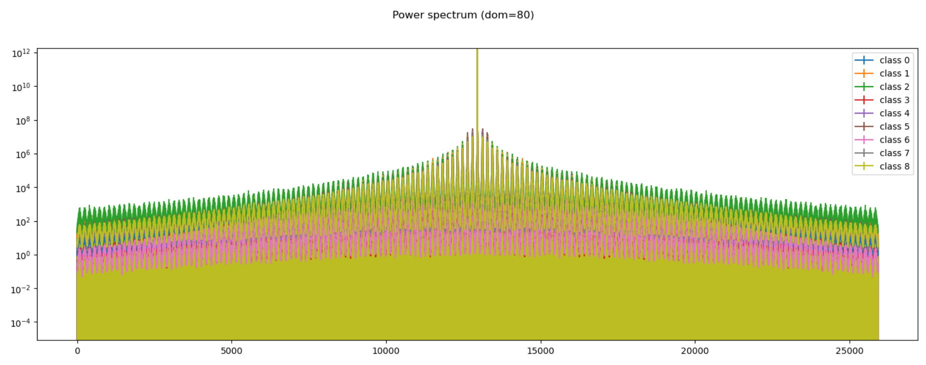

Figure 16: Mean feature vectors per class (power spectrum)

In the last two features (the autocovariance matrix and the power spectrum), the rotational invariance was not used. The power spectrum is invariant to translations, which is not the case with the autocovariance matrix. Not using rotational invariance can make classification harder when textures appear at different angles.

7 Proposition of features

7.1 Second-order scattering coefficients (invariant by rotation)

Based on the previous analysis and the course, I propose to use the second-order scattering coefficients with rotational invariance as a feature. The scattering transform constructs features using cascaded wavelet transforms followed by modulus and averaging operations. For an image $X(u)$, the zeroth-order coefficient is a global average:

$$S_0x = x * \phi_J,$$

where ϕ_J is a low-pass filter at scale 2^J.

The first-order coefficients are computed by convolving the image with a set of directional wavelets ψ_{λ_1} (indexed by scale and orientation $\lambda_1 = (j_1, \theta_1)$), followed by a modulus and low-pass filtering:

$$S_1x(\lambda_1) = |x * \psi_{\lambda_1}| * \phi_J.$$

The second-order coefficients capture interactions between wavelet responses at different scales and orientations:

$$S_2x(\lambda_1, \lambda_2) = \big||x * \psi_{\lambda_1}| * \psi_{\lambda_2}\big| * \phi_J.$$

These coefficients are stable to deformations and invariant to translations due to the averaging by ϕ_J. To achieve rotational invariance, we further average the scattering coefficients over the angular variable:

$$Sx_{\text{rot}} = \int Sx(R_\theta u)\, d\theta,$$

where R_θ denotes a rotation by angle θ. In practice, this averaging is approximated by computing the mean across orientations in the scattering output.

This construction provides features that are invariant to translations and small deformations, due to the low-pass filtering and modulus operations. The features are also invariant to rotations, by averaging across angular responses.

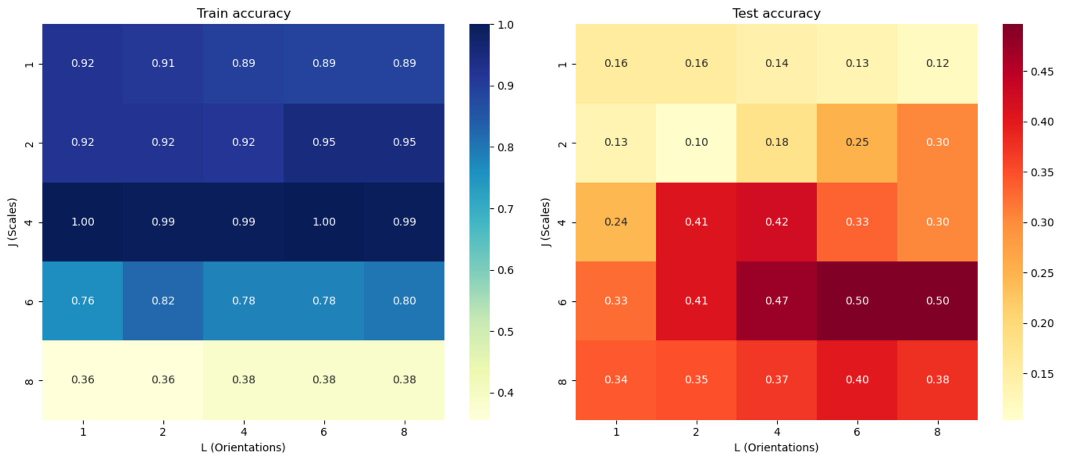

Figure 17: Accuracy scores of the LDA using 2nd-order scattering coefficients invariant by rotation as features for different scales j and orientations λ

The heatmaps show the LDA classification accuracy using second-order scattering coefficients invariant to rotation, for different values of scales j and orientations λ. We observe that training accuracy is generally high (close to 1.0) for small and moderate values of j (particularly $j = 4$), indicating that the model fits well on the training set. However, the test accuracy remains much lower, with the best values (up to 0.50) reached for $j = 6, \lambda = 6$ or 8, suggesting that larger j and λ capture more discriminative information for generalization.

To compare these results with the previous features, we see that the second-order scattering coefficients invariant to rotation offer an interesting trade-off between invariance and discriminability. While they do not reach the high training accuracies of the power spectrum (above 90%) or the autocovariance matrix (80%), their test accuracy, peaking around 50%, remains acceptable. Notably, they outperform the empirical mean, which yielded poor performance (around 33% train and 37% test). However, the power spectrum still achieves better generalization (up to 63% test accuracy) and the autocovariance matrix performs best overall (near 70% test accuracy).

These results are further confirmed by the plots below, which show the scattering coefficients (with $j = 6$ and $\lambda = 8$) for each images of each class. We can observe that some classes have very similar feature profiles, making them difficult to distinguish. This overlap in feature distributions explains the relatively low test accuracy, as the model struggles to separate these visually or structurally similar classes.

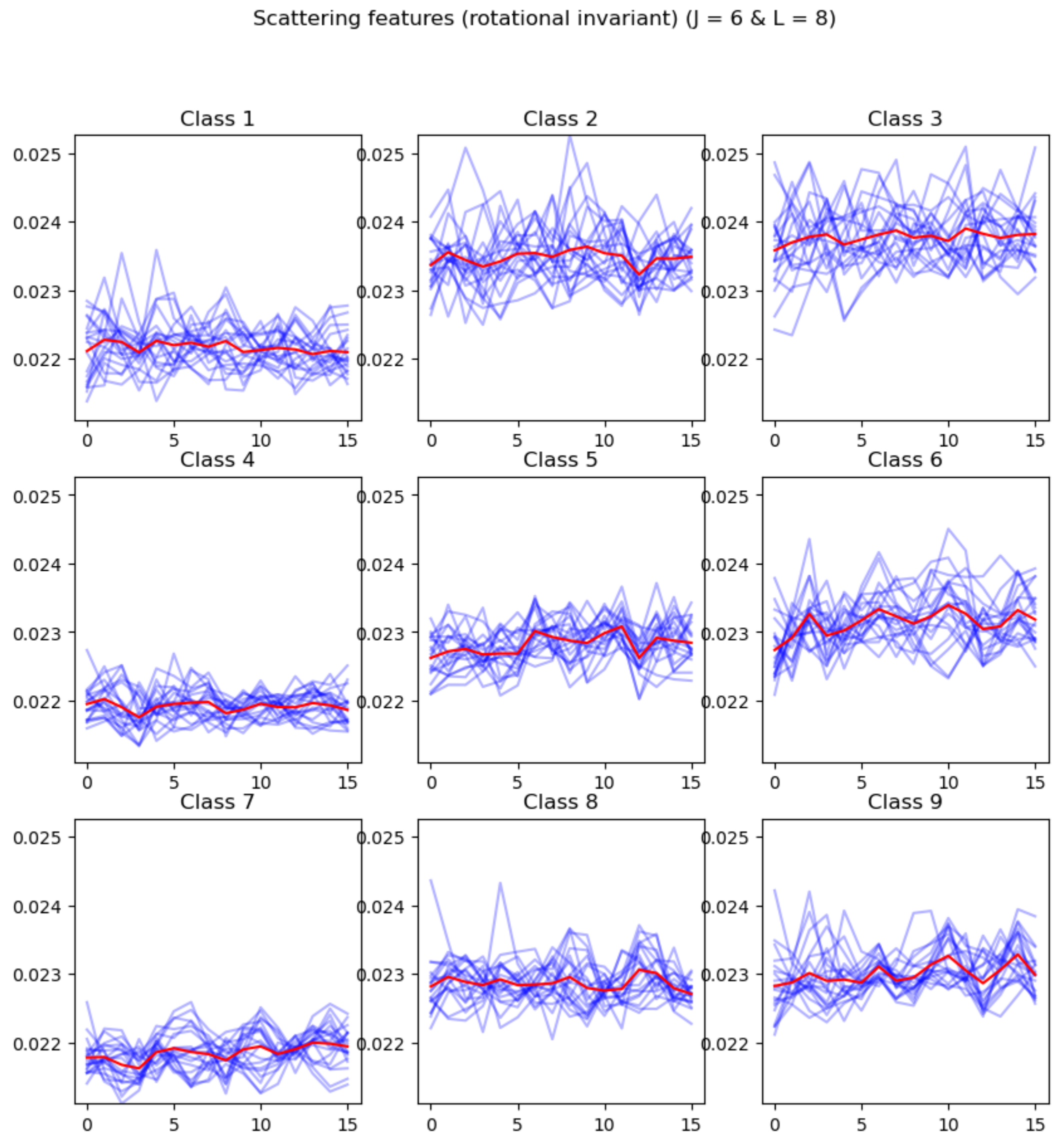

Figure 18: Second-order scattering coefficients (invariant by rotation) of each image per class.

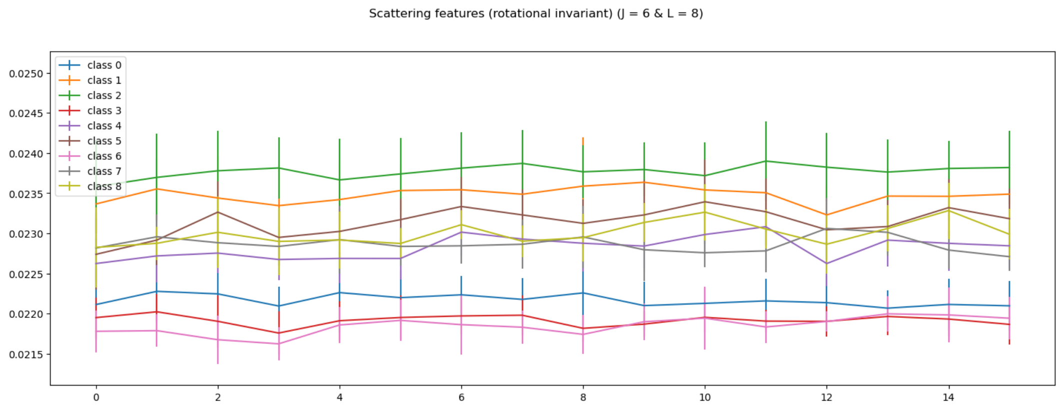

Figure 19: Mean feature vectors per class (second-order scattering coefficients invariant by rotation)

7.2 Second-order scattering coefficients (invariant by translation)

As an additional experiment, I was curious to see how the second-order scattering coefficients would perform without applying rotational invariance. So, I used the raw scattering coefficients directly, without averaging over the orientation axis. And we obtain the following heatmaps:

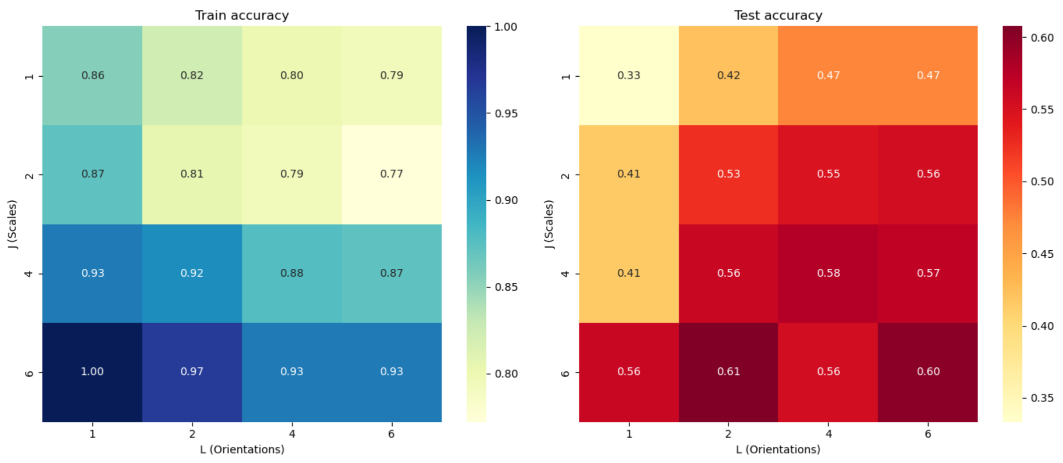

Figure 20: Accuracy scores of the LDA using 2nd-order scattering coefficients invariant by translation as features for different scales j and orientations λ

Compared to the previous results using rotation-invariant scattering features, the classification performance has clearly improved, especially on the test set. The test accuracies now reach up to 61%, and are consistently above 50% for $j \geq 2$, with little sensitivity to the number of orientations λ. This suggests that preserving orientation information (i.e., not averaging over angles) provides richer, more discriminative features, which help the LDA classifier better separate classes.

These results are interesting because the input images are theoretically invariant under rotation (meaning their class identity should not depend on orientation) the experimental results show that preserving orientation details during feature extraction still leads to better classification.

On the training set, the performance remains high across the board (above 79%), and even reaches 100% accuracy for $j = 6$ and $\lambda = 1$. This indicates that the model can fit the training data very well and maybe too well (overfitting).