Image Processing and Pattern Recognition—Homework 2

Lukas Glaszner—01430043
Tobias Hamedl—11808141

December 2, 2020

1 Derivation

First, we need to derive the update of a mean shift iteration; the objective is to perform the following optimization:

$$\hat{x} = \arg\max_x \left[\exp\left(-\frac{1}{2\sigma^2} \|y - x\|^2 \right) \sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2} \left\| \frac{Dx - c_i^{\mathrm{dct}}}{h} \right\|^2 \right) \right] \tag{1}$$

$$= \arg\max p(x|y). \tag{2}$$

We can rewrite the posterior to be

$$p(x|y) = \sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2\sigma^2} \|y - x\|^2 - \frac{1}{2} \left\| \frac{Dx - c_i^{\mathrm{dct}}}{h} \right\|^2 \right), \tag{3}$$

calculating the gradient applying the chain rule and remembering that $\frac{dAx}{dx} = A^{\mathrm{T}}$ yields

$$\nabla p(x|y) = \sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2\sigma^2} \|y - x\|^2 - \frac{1}{2} \left\| \frac{Dx - c_i^{\mathrm{dct}}}{h} \right\|^2 \right) \left(\frac{1}{\sigma^2}(y - x) - \frac{1}{h} D^{\mathrm{T}} \cdot \frac{Dx - c_i^{\mathrm{dct}}}{h} \right). \tag{4}$$

We now slightly rewrite the last factor of this expression leading to

$$\nabla p(x|y) = \sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2\sigma^2} \|y - x\|^2 - \frac{1}{2} \left\| \frac{Dx - c_i^{\mathrm{dct}}}{h} \right\|^2 \right) \left(\frac{1}{h^2} D^{\mathrm{T}} c_i^{\mathrm{dct}} + \frac{1}{\sigma^2} y - \left(\frac{1}{h^2} D^{\mathrm{T}} D + \frac{1}{\sigma^2} I \right) x \right), \tag{5}$$

where we split up the exponent as in Eq. (1):

$$\nabla p(x|y) = \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)\sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2}\left\|\frac{Dx-c_i^{\mathrm{dct}}}{h}\right\|^2\right) \left(\frac{1}{h^2}D^{\mathrm{T}}c_i^{\mathrm{dct}} + \frac{1}{\sigma^2}y - \left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)x\right). \quad (6)$$

We now perform the following expansion:

$$\nabla p(x|y) = \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)\sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2}\left\|\frac{Dx-c_i^{\mathrm{dct}}}{h}\right\|^2\right) \left(\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)^{-1} \left(\frac{1}{h^2}D^{\mathrm{T}}c_i^{\mathrm{dct}} + \frac{1}{\sigma^2}y\right) - \left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)x\right), \quad (7)$$

which allows us to factor out an additional term, yielding

$$\nabla p(x|y) = \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right) \sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2}\left\|\frac{Dx-c_i^{\mathrm{dct}}}{h}\right\|^2\right) \left(\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)^{-1}\left(\frac{1}{h^2}D^{\mathrm{T}}c_i^{\mathrm{dct}} + \frac{1}{\sigma^2}y\right) - x\right). \quad (8)$$

Finally, we factor out the sum:

$$\nabla p(x|y) = \exp\left(-\frac{1}{2\sigma^2}\|y-x\|^2\right)\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right) \left[\sum_{i=1}^{N_\mathrm{c}} w_i \exp\left(-\frac{1}{2}\left\|\frac{Dx-c_i^{\mathrm{dct}}}{h}\right\|^2\right)\right] \underbrace{\left[\left(\frac{1}{h^2}D^{\mathrm{T}}D + \frac{1}{\sigma^2}I\right)^{-1}\left(\frac{1}{h^2}D^{\mathrm{T}}\frac{\sum_{i=1}^{N_\mathrm{c}} c_i^{\mathrm{dct}}\tilde{w}_i(x)}{\sum_{i=1}^{N_\mathrm{c}}\tilde{w}_i(x)} + \frac{1}{\sigma^2}y\right) - x\right]}_{m(x)}, \quad (9)$$

where $\tilde{w}_i(x) = w_i \exp\left(-\frac{1}{2h^2}\|Dx - c_i^{\mathrm{dct}}\|^2\right)$ is introduced to allow for a more compact notation and $m(x)$ is the so-called mean shift. The update is defined

to be $\tilde{m}(x) = m(x) - x$, i.e.

$$x \longleftarrow \left(\frac{1}{h^2} D^{\mathrm{T}} D + \frac{1}{\sigma^2} I\right)^{-1} \left(\frac{1}{h^2} D^{\mathrm{T}} \frac{\sum_{i=1}^{N_\mathrm{c}} c_i^{\mathrm{dct}} \tilde{w}_i(x)}{\sum_{i=1}^{N_\mathrm{c}} \tilde{w}_i(x)} + \frac{1}{\sigma^2} y\right). \tag{10}$$

Choosing reasonable parameters for h and σ will be vital during implementation, so let us take a moment to consider their meaning. Since we use an exponential kernel for KDE, the individual profiles resemble a Gaussian, where the bandwidth h is analogous to the variance. It thus controls the width of the kernels: if h is low, they are narrow; if it is large, they are broad.

In our model we assume Gaussian distributed noise, where σ is the variance. Given only a noisy input image, this parameter is unknown and we need find a suitable value by trial and error. Generally speaking, the noisier an image is, the larger σ will be.

2 Implementation

First, we need to implement the functions `mat1d(n)` and `mat2d(n)`, which calculate the matrices of the 1D and 2D discrete cosine transform (DCT), respectively. The DCT is most commonly defined as

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right], \quad k = 0, \ldots, N-1. \tag{11}$$

The elements of the corresponding $K \times N$ coefficient matrix are thus

$$c_{kn} = \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right]; \tag{12}$$

this is implemented with the outer product `torch.ger()`. In order to obtain an orthogonal matrix we need to multiply X_0 by $\frac{1}{\sqrt{2}}$ and the entire matrix by $\sqrt{\frac{2}{N}}$. The two-dimensional DCT coefficient matrix can subsequently be computed via the Kronecker product

$$C_{2\mathrm{D}} = C_{1\mathrm{D}} \otimes C_{1\mathrm{D}}. \tag{13}$$

We implement this equation by flattening the two one-dimensional matrices, applying the outer product on the resulting vectors and carefully putting back together the axes. To test whether our implementation actually yields

orthogonal coefficient matrices, we check whether $C_{1\mathrm{D}}^{\mathrm{T}} C_{1\mathrm{D}} = I$ and $C_{2\mathrm{D}}^{\mathrm{T}} C_{2\mathrm{D}} = I.$; this is indeed the case.

Implementing the mean shift algorithm is straightforward, however, we need to take care to use only `pytorch` functions and avoid `for`-loops or expanding tensors. For each iteration we save the PSNR for later visualization, as well as 10 random patches of the estimate in order to show the trajectory.

The function `KDE()` implements the KDE approximation using K-Means clustered DCT coefficients according to

$$\hat{p}(x) = \frac{1}{(2\pi)^{\frac{d}{2}} N_{\mathrm{c}} h^d} \sum_{i=1}^{N_{\mathrm{c}}} w_i \exp\left(-\frac{1}{2} \left\| \frac{Dx - c_i^{\mathrm{dct}}}{h} \right\|^2 \right). \tag{14}$$

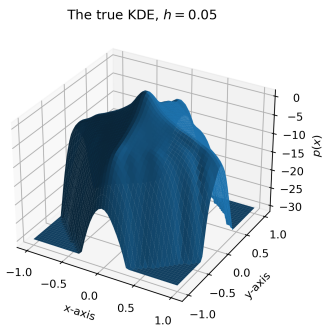
3 Visualization

3.1 K-Means Approximation

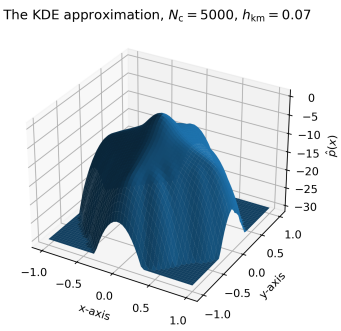
In order to make sure the estimated prior using $N_{\mathrm{c}} = 5000$ k-means centers is as accurate as possible, we plot both the given 'true' KDE and the estimate over a range of values h. The 'true' KDE was calculated using all—roughly 20 million—DCT coefficients of the image bank, the selection of the bandwidth is therefore not important.

Figure 1 shows the 'true' KDE as well as the result for the best parameter $h_{\mathrm{km}} = 0.07$ alongside one that is too large and one that is too small, where we have marginalized over dimension 2 in order to be able to plot in 3D. The spikes that occur if the bandwidth is too small are due to the fact that the individual kernels are very narrow, as explained above. If they are too broad however, they all cover the entire interval $[0, 1]$ in both directions, the KDE is thus nearly constant.

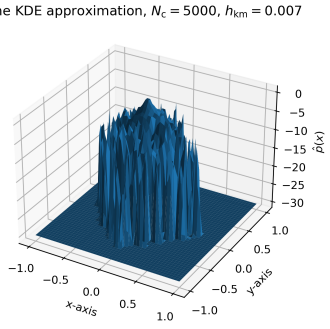
We also calculate the marginals along each dimension of both the 'true' and the approximated KDE, the results can be found in Figure 2. While they are definitely not the same, this is the best approximation we can achieve and it should suffice in the denoising task.



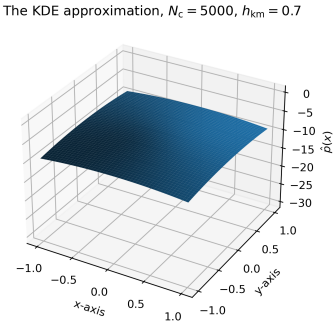
(a) $h = 0.05$



(b) $h_{\text{km}} = 0.07$

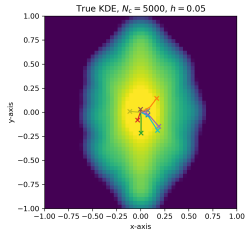


(c) $h_{\text{km}} = 0.007$

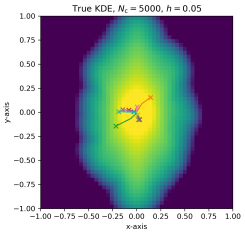


(d) $h_{\text{km}} = 0.7$

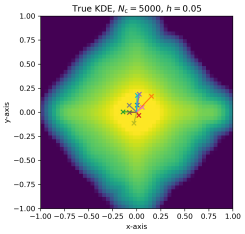
Figure 1: If the bandwidth h_{km} of the K-means approximation is too small, the resulting KDE is very spiky; if it is too large, it is almost flat and does not resemble the true KDE at all.



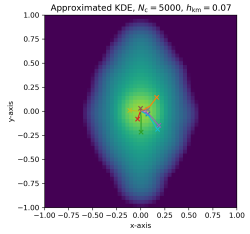
(a) Marginalization over dimension 0.



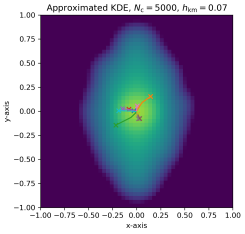
(b) Marginalization over dimension 1.



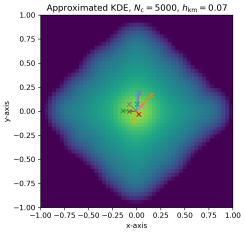
(c) Marginalization over dimension 2.



(d) Marginalization over dimension 0.



(e) Marginalization over dimension 1.



(f) Marginalization over dimension 2.

Figure 2: The shape of the approximation matches the ground truth rather well, but definately not exactly; moreover, the 'true' KDE features larger values.

3.2 Mean Shift Algorithm

While we could only use a patch size r of 2 while tuning the bandwidth, because otherwise we could not have marginalized over one dimension in order to plot the KDE; here we use a patch size of 4 and keeping $N_c = 5000$. The noise level of the additive noise is set to $\sigma_{\mathrm{n}} = \frac{25}{255}$; the noisy input image is shown in Figure 3.

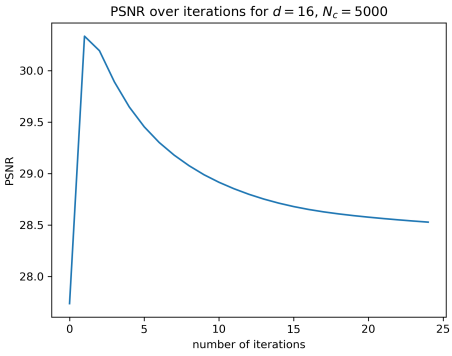
Figure 3: Noise is clearly present in the input image.

Figure 4, presents both the denoised output image, as well as the development of the PSNR over the number of iterations. We can see that the PSNR reaches a maximum very early on, thus we have rerun our algorithm with an additional stopping criterion: As soon as the PSNR value decreases the mean shift algorithm ends and the estimate is saved as the output, this already occurs after iteration 2. The result is shown in Figure 4a, the corresponding PSNR is 30.173.

We set the variance parameter in the mean shift algorithm to be $\sigma = 4\sigma_{\mathrm{n}}$; its choice generally depends on the selection of the bandwidth, which is $h = 0.07$ again. Generally speaking, we can say that for slightly larger or lower values of σ the PSNR that can be achieved does not vary significantly—it depends more strongly on the actual realization of the noise.

In Figure 2 we can also see the trajectory of 10 randomly selected coefficients in the prior. After saving 10 patches while performing the mean shift, we left-multiply them with D in order to transform them to DCT space and then plot the two dimensions also visualized in the corresponding KDE; the x's mark the starting points. We can see that they all move toward the center of the prior, which makes sense because the image features a lot of piecewise constant areas. Constant values correspond to $(0, 0, 0)$ in the 3-dimensional DCT-space and that is why the coefficient move towards the center.

(a) The noise has clearly been reduced, the image appears blurry and patchy, however.



(b) The PSNR rises rapidly at first, after which it deceases again, settling at around 28.5.

Figure 4: Note that the estimate does not originate from the same run as the PSNR plot.