Machine Learning

Assignment 1

March 22, 2022

Deadline: April 6, 2022 at 23:55h.
Submission: Upload your report (`report.pdf`), your implementation (`denoising.py`) and your figure file (`figures.pdf`) to the TeachCenter. Please do not zip your files. Please use the provided framework-file for your implementation.

1 Bayesian Denoising (3P)

The goal in this assignment is to use Bayes' rule for two different denoising settings. We therefore require a *training* set $\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, where $\mathbf{y}_n \in \mathbb{R}^D$ are clean, noise-free samples. Further, we assume a *test* set $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, where $\mathbf{x}_m \in \mathbb{R}^D$ are noisy samples.

To frame this in a Bayesian setting, we use two D-dimensional random variables $\mathbf{X}$ and $\mathbf{Y}$ representing the noisy and the clean sample, respectively. Next, we can model the noisy sample with

$$\mathbf{X} = \mathbf{Y} + \mathbf{N}, \tag{1}$$

where $\mathbf{N} \sim \mathcal{N}(0, \mathbf{\Sigma})$ is the D-dimensional random variable representing the noise. Furthermore, we assume that the noise is independent in each of the D-dimensions features, i.e.

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2 & & \\ & \ddots & \\ & & \sigma^2 \end{pmatrix} \in \mathbb{R}^{D \times D}. \tag{2}$$

In terms of probabilities, adding noise as shown in eq. (1) can be described by the likelihood

$$\begin{aligned} p(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}) &= \frac{1}{\sqrt{(2\pi)^D |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{\Sigma}^{-1} (\mathbf{y} - \mathbf{x})\right) \\ &\overset{eq.\ (2)}{=} \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{x}\|^2\right) \end{aligned} \tag{3}$$

where the lower-case letters $\mathbf{x}$ and $\mathbf{y}$ are one specific instance of the upper-case random variables $\mathbf{X}$ and $\mathbf{Y}$, respectively.

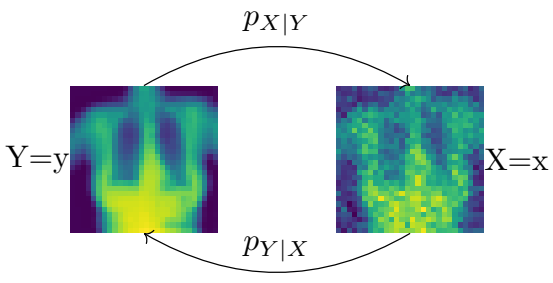

Figure 1: Relation between noisy image $\mathbf{x}$ and noise-free image $\mathbf{y}$.

Figure 1 shows a visualization of eq. (3) for the setting of image denoising, where noise is added to a specific instance $\mathbf{Y} = \mathbf{y}$. Moreover, the process of denoising a noisy image $\mathbf{X} = \mathbf{x}$ can be described by the posterior

$$p(\mathbf{Y}|\mathbf{X} = \mathbf{x}). \tag{4}$$

Since we don't know the complete distribution over the target space $p(\mathbf{y})$ we need to make some assumptions given our finite training dataset $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$. We want to compare two options to estimate our prior distribution $p(\mathbf{y})$, see also Figure 2:

1. use the given discrete data $\mathbf{y}$ with a discrete Dirac measure

$$p(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \delta_{\mathbf{y}_n}(\mathbf{y}), \tag{5}$$

2. use kernel density estimation (KDE), a non-parametric approach for density estimation with a kernel k and a bandwidth h

$$p(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{y} - \mathbf{y}_n}{h}\right), \tag{6}$$

where any kernel function k can be used, however, a smooth kernel, such as the Gauss kernel is desirable to avoid discontinuities in the estimated probability density function and yields the following

$$p(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(\sqrt{2\pi}h)^D} \exp\left(-\frac{\|\mathbf{y} - \mathbf{y}_n\|^2}{2h^2}\right). \tag{7}$$

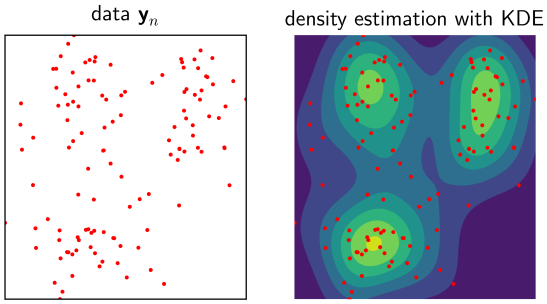

Figure 2: Given a finite numer of 2D samples (red markers) we can either use the data itself (Dirac measure) or a kernel density estimation (KDE). Note that the latter allows for a smooth prior distribution and for $h \to 0$ it approaches the Dirac measure.

Having a model for the likelihood and the prior distribution, we now have two possibilities to use the posterior eq. (4) for making predictions, as shown in the lecture notes and slides. The first possibility is the so called *conditional mean*, also know as *regression function*, and is given by

$$\hat{\mathbf{y}}_{CM}(\mathbf{x}) = \mathbb{E}_{\mathbf{Y}}[\mathbf{Y}|\mathbf{X} = \mathbf{x}] = \frac{\sum_{n=1}^{N} \mathbf{y}_n p(\mathbf{Y} = \mathbf{y}_n) p(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}_n)}{\sum_{n=1}^{N} p(\mathbf{Y} = \mathbf{y}_n) p(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}_n)}, \tag{8}$$

where $\hat{\mathbf{y}}_{CM}(\mathbf{x})$ minimizes the expected quadratic loss for the noisy sample $\mathbf{x}$.

The second option is to compute the maximum a posteriori (MAP) prediction, which is given by

$$\hat{\mathbf{y}}_{MAP}(x) = \arg\max_{\mathbf{y}_n} p(\mathbf{Y} = \mathbf{y}_n) p(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}_n). \tag{9}$$

Tasks

1. Verify analytically that the conditional mean $\hat{\mathbf{y}}_{CM} = \mathbb{E}_{\mathbf{Y}}[\mathbf{Y}|\mathbf{X} = \mathbf{x}]$ is given by eq. (8)
2. Verify analytically that the MAP $\hat{\mathbf{y}}_{MAP}$ is given by eq. (9)

2 Bayesian Denoising – 2D Toytask (10P)

The goal of this task is to apply the two approaches for obtaining the posterior, namely the conditional mean eq. (8) and the MAP eq. (9), to denoise 2D out-of-distribution data points. Therefore, we are going to use a dataset of 2D points ($D = 2$) that are drawn from different normal distributions.

Tasks

1. Construct a 2D training set $\mathbf{y}$ consisting of $N = 900$ samples, originating from 3 different multivariate normal distributions (each normal distribution should give rise to 300 samples). Use the following parameters to set up the three distributions:

$$\mu_{\mathbf{1}} = (0, 0)^T; \mu_{\mathbf{2}} = (0, 1.5)^T, \mu_{\mathbf{3}} = (1.5, 1.5)^T; \mathbf{\Sigma_1} = \mathbf{\Sigma_2} = \mathbf{\Sigma_3} = \begin{pmatrix} 0.075 & 0 \\ 0 & 0.075 \end{pmatrix}$$

2. Implement kernel density estimation based on the training data $\mathbf{y}$ using a Gaussian kernel (7). What is the influence of the bandwidth h?
3. Construct a test dataset $\mathbf{x}$ by selecting one well-chosen point that is out-of-distribution, i.e. somewhere in the 2D plane such that it is clearly distinguishable from the underlying distributions of the training dataset $\mathbf{y}$.
4. Implement the conditional mean model in eq. (8).
5. Implement the MAP model in eq. (9).
6. Test your implementation of both the conditional mean and the MAP model with the test point $\mathbf{x}$ by assuming $\sigma^2 = 1$ with the prior distributions, i.e. Dirac measure and KDE.
7. For both the dirac prior and the KDE prior, plot the training samples $\mathbf{y}$, the estimated prior density and the test sample together with its conditional mean and MAP solution. Choose 3 different bandwidths h for the KDE prior. Describe the plots.

Implementation details

- Using `numpy.random.multivariate_normal` random samples can be drawn from a multivariate normal distribution.
- Using `numpy.meshgrid` you can construct a finely discretized grid to evaluate your estimated density. Filled contour plots can be obtained using `matplotlib.pyplot.contourf`.

Implement this task in `task2` of `denoising.py`. Please do not use any other libraries than the ones that are already included.

3 Bayesian Denoising – Image Dataset (12P)

Finally, we want to apply our model to image denoising. Therefore, use the ChestMNIST dataset from MedMNIST[1] inspired by the well-known MNIST dataset.
Hint: for numerical stability we want to implement the product of the likelihood eq. (3) and the prior that are needed to compute the conditional mean eq. (8) and the MAP eq. (9) in the log-domain, where the multiplication will subsequently be expressed as an addition.

Therefore, we need to rewrite the likelihood and the prior in the log-domain. For the likelihood in eq. (3), this can be expressed as

$$\log p(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}) = -\frac{D}{2}\log(2\pi\sigma^2) - \frac{\|y - x\|^2}{2\sigma^2}. \tag{10}$$

Here, we only will use the KDE prior anymore, which can be re-written as

$$\log p(\mathbf{y}) = \log\left(\sum_{n=1}^{N} \exp\left(-\log(N) - \frac{D}{2}\log(2\pi) - D\log(h) - \frac{\|\mathbf{y} - \mathbf{y}_n\|^2}{2h^2}\right)\right). \tag{11}$$

[1] https://github.com/MedMNIST/MedMNIST

Tasks

1. Give an analytic expression of the conditional mean given in eq. (8) and eq. (9), where the likelihood and the prior are evaluated in the log-domain.
2. Verify analytically that the likelihood in the log-domain is given by eq. (10).
3. Verify analytically that the KDE prior in the log-domain is given by eq. (11).
4. Load the clean training and test images provided by the module `medmnist`. Randomly select $N = 1000$ training images for $\mathbf{y}$ and $M = 25$ test images for $\mathbf{x}$ for all experiments.
5. Construct the test dataset $\mathbf{x}$ by adding pixel-wise zero-mean Gaussian noise with variance $\sigma^2 = \{0.1, 1\}$.
6. Implement the KDE prior distribution analogously to the previous task. Ensure that it is given in the log-domain and use the numerically stable log-sum-exp trick to implement eq. (11)

$$\log \sum_{j=1}^{J} \exp(x_j) = \max(x_j) + \log \sum_{j=1}^{J} \exp(x_j - \max(x_j)).$$

7. Implement the conditional mean eq. (8) and the MAP model eq. (9), and ensure the numerical stability of both implementations.
8. Test your implementation of the conditional mean and the MAP model for the KDE prior with $\sigma^2 = \{0.1, 1\}$.
9. Plot the $M = 25$ results together in one plot for each $\sigma^2 = \{0.1, 1\}$. To this end make one large image each with size $(5\sqrt{D}) \times (5\sqrt{D})$ containing all the sub-images. Don't forget to convert the result-vectors back to images before plotting.
10. Describe the plots: What are the differences wrt. σ^2? How and why are the two posterior models different from each other?

Implementation details

- Note that the images of size $(H = \sqrt{D}) \times (W = \sqrt{D})$ are assumed to be flattend to a vector of length D. In NumPy the flattening operation can be done with `flat = img.flatten()` and the inverse operation, i.e. from vector to image can be done with `img = flat.reshape(H,W)`.
- Additionally, all images are assumed to be normalized to the interval $[0, 1]$. This can be simply achieved by dividing the images by 255 after loading the data.
- In order to avoid numerical problems, we need to implement the sum in both the numerator and the denominator in the computation of the conditional mean robustly with $z = \max_n r_n$:

$$\sum_{n=1}^{N} \exp(r_n) = \exp(z) \sum_{n=1}^{N} \exp(r_n - z). \tag{12}$$

Implement the Bayesian Denoising in `task3` of `denoising.py`. Again, do not use any other libraries than `numpy`, `matplotlib` and `medmnist`. To obtain the dataset, install `MedMNIST` via `pip install medmnist`, and load the train/test data with

```
train_data = medmnist.ChestMNIST('train', download=True).imgs
test_data = medmnist.ChestMNIST('test', download=True).imgs
```