Machine Learning

Assignment 1

March 24, 2020

Deadline: April 14, 2020 at 23:55h.
Submission: Upload your report and your implementation to the TeachCenter. Please do not zip your files.

1 Probability Refresher

4% of our population have a wheat allergy. However, there are another 1% that are affected by celiac disease, which is an autoimmune disease. Let us define D as the random variable representing the disease state with $D = \{0, 1, 2\}$ for healthy, wheat allergic and celiac patients, respectively. A serological test can be done to test for celiac disease, returning either a positive ($T = 1$) or a negative ($T = 0$) result. For a celiac, the test returns a positive result with 98%, however, it sometimes fails with 1% reporting that a healthy person has celiac disease. Since the test is not designed to diagnose wheat allergy, there is also the possibility that a positive test result is due to wheat allergy which happens with a probability of 20%.

Complete the following table by calculating all joint probabilities and marginal probabilities using Bayes' rule, sum rule and product rule.

T \ D	healthy $D = 0$	allergy $D = 1$	celiac $D = 2$	$p(T_i)$
pos. $T = 1$				
neg. $T = 0$				
$p(D_i)$				

2 Mutual Information

The aim of this exercise is to compute the mutual information (MI) of the marginals of a 2D Gaussian distribution. Let $\mathbf{X} = (X_1, X_2)^T$ be a normally distributed 2D random vector

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\mu} = 0, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2 & \alpha\sigma^2 \\ \alpha\sigma^2 & \sigma^2 \end{pmatrix},$$

where $\sigma > 0$ and $|\alpha| < 1$ is a free parameter which is bounded in order to ensure positive definiteness of $\boldsymbol{\Sigma}$. Complete the following tasks:

1. Compute the MI $I[X_1, X_2]$ between the random variables X_1 and X_2 in dependence of α.
2. For which setting of α is the MI minimal or maximal?
3. Plot the Gaussian distribution $p(\mathbf{x})$ with 3 different values for σ^2 and α, respectively. Compare and interpret the plots.
4. Plot the marginal distributions $p(x_1)$ and $p(x_2)$ for the same σ^2 and α as in the previous task and describe your results.

3 Empirical Risk Minimization

In many supervised machine learning tasks the goal is to find a predictor y from a function space $\mathcal{Y}$, that maps variables from an input space $\mathcal{X}$ to a target space $\mathcal{T}$. Based on a loss function $L(\cdot,\cdot) : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$, the performance of such a predictor is commonly measured by the *true risk* (or expected loss)

$$R(y) = \mathbb{E}_{X,T}[L(T, y(X))],$$

which is evaluated on the joint probability distribution $P_{X,T}$ of two random variables X, T. The goal in machine learning is always to solve the following minimization problem

$$y^*(x) = \arg\min_{y \in \mathcal{Y}} R(y),$$

where $y^*(x)$ is the true optimal predictor. Unfortunately $P_{X,T}$ is usually unknown due to the limitation of finite data. With a given finite dataset of N input values $\mathbf{x} = (x_1, ..., x_N)$ and corresponding N target values $\mathbf{t} = (t_1, ..., t_N)$ the risk minimization problem is replaced by the *empirical risk* minimization problem

$$\hat{y}(x) = \arg\min_{y \in \mathcal{Y}} \left\{ \hat{R}(y) := \frac{1}{N} \sum_{n=1}^{N} L(t_n, y(x_n)) \right\},$$

which of course only serves as an approximation to the true risk minimization problem.

In this example we investigate empirical risk minimization for regression. We assume that $x_n \in [0, 2\pi]$ and $t_n = \sin(x_n) + \eta$ with $\eta \sim \mathcal{N}(0, \sigma^2)$. Moreover, we assume a quadratic loss function i.e. $L(t, y(x)) = (t - y(x))^2$. For the class of potential functions $y(x) \in \mathcal{Y}$, we consider the family of linear functions

$$y_p(x, \mathbf{w}) = w_0 x^0 + w_1 x^1 + w_2 x^2 + ... + w_p x^p,$$

where $p \in \mathbb{N}$ is the degree of the polynomial in x and $\mathbf{w} = (w_0, ..., w_p)^T \in \mathbb{R}^{p+1}$ is the corresponding weight vector. The empirical risk minimization problem can therefore be written as the least squares optimization problem

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} (y_p(x_n, \mathbf{w}) - t_n)^2 = \arg\min_{\mathbf{w}} \frac{1}{N} \|\mathbf{\Phi w} - \mathbf{t}\|^2 \tag{1}$$

$$\mathbf{\Phi} = \begin{pmatrix} x_1^0 & x_1^1 & x_1^2 & \dots & x_1^p \\ x_2^0 & x_2^1 & x_2^2 & \dots & x_2^p \\ \vdots & & \ddots & & \vdots \\ x_N^0 & x_N^1 & x_N^2 & \dots & x_N^p \end{pmatrix} \in \mathbb{R}^{N \times (p+1)} \qquad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{pmatrix} \in \mathbb{R}^{p+1} \qquad \mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix} \in \mathbb{R}^N$$

The optimal empirical predictor is finally given by $\hat{y}_p(x, \mathbf{w}^*)$.

Tasks

1. What is the true optimal predictor (regression function) $y^*(x)$?
2. Analytically derive the solution $\mathbf{w}^*$ for minimizing (1).
3. Create a random data set with $N = 10$ data points $x_n \in [0, 2\pi]$ and t_n computed as stated above with $\sigma^2 = 0.1$.
4. Compute the optimal empirical predictors $\hat{y}_p(x, \mathbf{w}^*)$ for linear models with increasing model complexity $p = 1, 2, ..., 8$ and plot it together with $y^*(x)$ over the entire interval $[0, 2\pi]$. Also include the samples (x_n, t_n) from your data set in the plot.
5. Compute and plot the *empirical risk* $\hat{R}(\hat{y}_p(x, \mathbf{w}^*))$ in dependence of the model complexity p.
6. Compute and plot the *true risk* $R(\hat{y}_p(x, \mathbf{w}^*))$ by making use of the decomposition of the expected quadratic loss [PRML, eq. 1.90]. Approximate the first integral by uniformly sampling x on the interval $[0, 2\pi]$. The second integral can be evaluated in closed form.

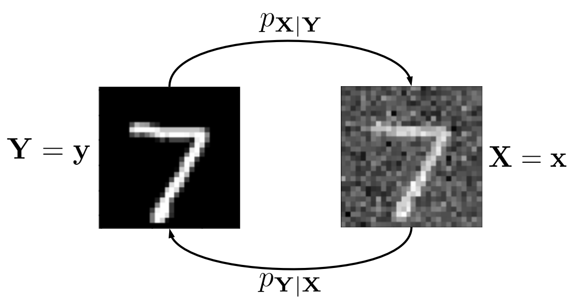
$p_{\mathbf{X}|\mathbf{Y}}$
$\mathbf{Y} = \mathbf{y}$
$\mathbf{X} = \mathbf{x}$
$p_{\mathbf{Y}|\mathbf{X}}$

Figure 1: Relation between noisy and noise-free images.

7. For which model is the *true risk* minimal? What can you observe for increasing model complexity p?

8. Recompute the previous plots using $N = 100$ samples. How and why do the results change?

Implementation details Implement the tasks with Python and name the file `erm.py`. Please do not use any other libraries than `numpy` and `matplotlib`.

4 MNIST - Bayesian Denoising

The goal of this task is to use Bayes' rule for image denoising. We are going to use the well known MNIST dataset[1] in this task. We assume we have given a *training* set $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$, where $\mathbf{y}_n \in \mathbb{R}^D$ are the clean (=noise-free) images. Similarly, we assume we have given a *test* set $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$, where $\mathbf{x}_m \in \mathbb{R}^D$ are noisy images.

To put everything into a Bayesian setting, we use two D-dimensional random variables $\mathbf{X}$ and $\mathbf{Y}$ representing the noisy and the clean image, respectively. Next, we can model the noisy image with

$$\mathbf{X} = \mathbf{Y} + \mathbf{N}, \tag{2}$$

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 over the pixels, i.e.

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2 & & \\ & \ddots & \\ & & \sigma^2 \end{pmatrix} \in \mathbb{R}^{D \times D}. \tag{3}$$

In terms of probabilities, adding noise as shown in eq. (2) can be described by the likelihood

$$\begin{aligned} p(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}) &= \frac{1}{\sqrt{(2\pi)^D |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{\Sigma}^{-1} (\mathbf{y} - \mathbf{x})\right) \\ &\overset{eq.\ (3)}{=} \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{x}\|^2\right) \end{aligned} \tag{4}$$

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 shows a visualization of eq. (4), where a specific instance noise is added to a specific instance $\mathbf{Y} = \mathbf{y}$. Furthermore, fig. 1 shows that the process of removing the noise from a noisy image can be described by the posterior

$$p(\mathbf{Y} | \mathbf{X} = \mathbf{x}), \tag{5}$$

where $\mathbf{X} = \mathbf{x}$ is a specific instance of a noisy image.

As we have seen in the lecture notes and slides, we have two possibilities to use the posterior eq. (5) for making predictions. The first possibility is the so called *conditional mean*, also know as *regression*

[1] http://yann.lecun.com/exdb/mnist/

function. Under the assumption that $\mathbf{Y}$ is uniformly distributed with $p(\mathbf{Y}) \sim \mathcal{U}([0,1]^D)$, the conditional mean is given by

$$\hat{\mathbf{y}}_{CM}(\mathbf{x}) = \mathbb{E}_{\mathbf{Y}}[\mathbf{Y}|\mathbf{X}=\mathbf{x}] = \frac{\sum_{n=1}^{N} \mathbf{y}_n p(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}_n)}{\sum_{n=1}^{N} p(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}_n)}, \tag{6}$$

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

The second option is to compute the maximum a posteriori (MAP) prediction. Again, by assuming $p(\mathbf{Y}) \sim \mathcal{U}([0,1]^D)$ the MAP prediction is given by

$$\hat{\mathbf{y}}_{MAP}(x) = \arg\max_{\mathbf{y}_n} p(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}_n). \tag{7}$$

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. (6)
2. Verify analytically that the MAP $\hat{\mathbf{y}}_{MAP}$ is given by eq. (7)
3. Load the clean training images and the clean test images with the provided functions in Python.
4. Construct the training dataset **y** by loading the training dataset as provided.
5. Construct the test dataset **x** by adding pixel-wise random Gaussian noise with variance σ to obtain $\mathbf{x}_n$.
6. Implement the conditional mean model eq. (6)
7. Implement the MAP model eq. (7)
8. Randomly select $N = 10000$ images from **y** and $M = 100$ images from **x** for all your experiments.
9. Test your implementation of the conditional mean and the MAP model with $\sigma = \{0.25, 0.5, 1\}$
10. Plot the $M = 100$ results for all σ together in one plot. To this end make one large image with size $10D \times 10D$ containing all the sub-images. Don't forget to convert the result-vectors back to images before plotting.
11. Describe the plots: What are the differences wrt. σ? How and why differ the two models from each other?

Implementation details

- Note that eqs. (2) to (7) assume the images of size $(H = \sqrt{D}) \times (W = \sqrt{D})$ 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 enumerator and the denominator in eq. (6) robustly with

$$\sum_{n=1}^{N} \exp(r_n) = \exp(z) \sum_{n=1}^{N} \exp(r_n - z) \tag{8}$$

  where $r_n = -\frac{1}{2\sigma^2}\|\mathbf{y_n} - \mathbf{x}\|^2$ and $z = \max_n r_n$.

Implement the Bayesian Denoising in Python and name the file `denoising.py`. Please do not use any other libraries than `numpy` and `matplotlib`. You can use the provided file `mnist.py` to load the MNIST dataset with Python. You can load the training data with
`Y = mnist.load_data("train-images-idx3-ubyte.gz")` and the test data with
`X = mnist.load_data("t10k-images-idx3-ubyte.gz")`.