Statistical Machine Learning: Exercise 2

Parametric and Non-parametric Density Estimation, Expectation Maximization

Prof. Stefan Roth, Dr. Simone Schaub-Meyer

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2021

Due date: Jun 15th, 23:59 PM

Task 1: Optimization (25)

1a) Numerical Optimization (15 Points)

Rosenbrock's function is a benchmark function for optimization and defined as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right].$$

To solve the corresponding optimization problem, we would like to find

$$\arg\min_{(x)} f((x)).$$

In Python, write a simple gradient descent algorithm and run it for 10,000 steps on Rosenbrock's function with $n = 20$. To do so, determine the derivative of the function and write a function which evaluates it. Attach a snippet of your GD algorithm, discuss the effects of the learning rate and attach a plot of your learning curve with your best learning rate.

(5 pt. for clear code of Rosenbrocks function and derivative)
(3 pt. for correct gradient descent)
(2 pt. for good documentation)
(2 pt. for clear and complete graph [title, axis, legend])
(3 pt. for discussion of learning rate)

https://github.com/syahrulhamdani/Gradient-Descent-for-Rosenbrock-Function/blob/master/rosenbrock.py

https://github.com/cjtonde/optimize_rosenbrock/blob/master/src/optimize_rosenbrock.py

https://zhuanlan.zhihu.com/p/161547509

1b) Gradient Descent Variants (10 Points)

Throughout this class we will see that gradient descent is one of the most used optimization techniques in machine learning. This task asks you to get a more in-depth understanding of the topic by conducting some research by yourself.

1) There are several variants of gradient descent, namely *vanilla* (also known as *batch*), *stochastic* and *mini-batch*. Each variant differs in how much data we use to compute the gradient of the objective function. Discuss the differences among them, pointing out pros and cons of each one.

2) Many gradient descent optimization algorithms use the so-called *momentum* to improve convergence. What is it? Is it always useful?

Task 2: Density Estimation - MLE (8 Points)

2a) Maximization Likelihood Estimate of the Exponential Distribution (8 Points)

Derive the maximum likelihood estimate for the parameter $s \neq 0$ of the following probability density function.

$$p(x \mid s) = \begin{cases} \frac{1}{s} \exp\left(-\frac{x}{s}\right) & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Start your derivations with the likelihood function and state the assumptions you make during your derivations.

Task 3: Density Estimation (20 Points)

In this exercise, you will use the datasets `densEst1.txt` and `densEst2.txt`. The datasets contain 2D data belonging to two classes, C_1 and C_2.

3a) Prior Probabilities (2 Points)

Compute the prior probability of each class from the dataset.

3b) Biased ML Estimate (5 Points)

Define the bias of an estimator and write how we can compute it. Then calculate the biased and unbiased estimates of the conditional distribution $p(x|C_i)$, assuming that each class can be modeled with a Gaussian distribution. Which parameters have to be calculated? Show the final result and attach a snippet of your code. Do not use existing functions, but rather implement the computations by yourself!

3c) Class Density (5 Points)

Using the unbiased estimates from the previous question, fit a Gaussian distribution to the data of each class. Generate a single plot showing the data points and the probability densities of each class. (Hint: use the contour function for plotting the Gaussians.)

3d) Posterior (8 Points)

In a single graph, plot the posterior distribution of each class $p(C_i|x)$ and show the decision boundary.

Task 4: Non-parametric Density Estimation (20 Points)

In this exercise, you will use the datasets `nonParamTrain.txt` for training and `nonParamTest.txt` for evaluating the fidelity of your model.

4a) Histogram (4 Points)

Compute and plot the histograms using bins of size 0.02, 0.5, and 2.0. Intuitively, indicate which bin size performs the best and explain why. Knowing only these three trials, would you be sure that the one you picked is truly the best one? Attach the plot of the histograms and a snippet of your code.

4b) Kernel Density Estimate (6 Points)

Compute the probability density estimate using a Gaussian kernel with $\sigma = 0.03$, $\sigma = 0.2$, and $\sigma = 0.8$. Compute the log-likelihood of the training data for each case, compare them and show which parameter performs the best. Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval $x \in [-4, 8]$, attach a snippet of your code and comment the results.

4c) K-Nearest Neighbors (6 Points)

Estimate the probability density with the K-nearest neighbors method with $K = 2, K = 8, K = 35$. Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval $x \in [-4, 8]$, attach a snippet of your code and comment the results.

4d) Comparison of the Non-Parametric Methods (4 Points)

Estimate the log-likelihood of the testing data using the KDE estimators and the K-NN estimators. Why do we need to test them on a different data set? Compare the log-likelihoods of the estimators w.r.t. both the training and testing sets in a table. Which estimator would you choose?

Task 5: Expectation Maximization (20 Points)

In this exercise, you will use the datasets `gmm.txt`. It contains data from a Gaussian Mixture Model with four 2-dimensional Gaussian distributions.

5a) Gaussian Mixture Update Rules (2 Points)

Define the model parameters and the update rules for your model. Specify the E- and M-steps of the algorithm.

5b) EM (18 Points)

Implement the Expectation Maximization algorithm for Gaussian Mixture Models. Initialize your model uniformly. To compute the multivariate normal probability density function you can use the function `multivariate_normal` from `scipy.stats`. Generate plots at different iterations $t_i \in \{1, 3, 5, 10, 30\}$, showing the data and the mixture components, and plot the log-likelihood for every iteration $t_i = 1 : 30$. Attach a snippet of your code.