Dr. Peter Arndt
Dr. Konrad Völkel

Heinrich-Heine-Universität Düsseldorf
Sommersemester 2022

Machine Learning
Exercise Sheet 3
(3 Exercises, 90 Points)
Due: 1.11.2022, 10:00

Exercise 1: (30 Points)

MLEs
Derive the maximum likelihood estimators for the following distributions. For this write down the log-likelihood function given n observations $x_1, \dots, x_n$ and determine the maximum with respect to the parameter.

1. Gaussian normal distribution $\mathcal{N}(x|\mu, \sigma^2)$ for μ given observations $x_1, \dots, x_n$.
2. Exponential distribution with probability density function $f(x|\lambda) = \lambda e^{-\lambda x}$ for $\lambda > 0$ given observations $x_i \geq 0$.

Exercise 2: (30 Points)

Throwing a (not necessarily fair) K-sided die n times allows us to infer posteriors for the unknown probabilities. The data is $\mathcal{D} = (x_1, \dots, x_K)$ with x_j being the number of times you have seen side j. Assume a Dirichlet prior (with (hyper-)parameter vector α) for the parameter vector $\theta = (\theta_1, \dots, \theta_K)$ with $0 \leq \theta_j \leq 1$ and $\sum_j \theta_j = 1$ and a multinomial likelihood for your data, i.e.,

$$p(\theta) = \mathrm{Dir}(\theta|\alpha) \qquad p(\mathcal{D}|\theta) = \mathrm{Mu}(x|n, \theta)$$

Show that the posterior is also Dirichlet, i.e., show

$$p(\theta|\mathcal{D}) = \mathrm{Dir}(\theta|\alpha + x)$$

Hint: You do not have to calculate the normalization constant, i.e., prove that the posterior is proportional to a Dirichlet distribution with parameter $\alpha + x$.

Exercise 3: (30 Points)

Recall the story from the lecture "Two sellers at Amazon have the same price. One has 90 positive and 10 negative reviews. The other one 2 positive and 0 negative. Who should you buy from?" Write down the posterior probabilities about the reliability (as in the lecture).

1. Calculate $p(\theta_1 > \theta_2|\mathcal{D}_1, \mathcal{D}_2)$ using quadrature, e.g., by using the function `dblquad` from `scipy.integrate`.
2. Calculate $p(\theta_1 > \theta_2|\mathcal{D}_1, \mathcal{D}_2)$ using Monte Carlo integration[1]. You can generate Beta distributed samples with the function `scipy.stats.beta.rvs(a,b,size)`.

[1] `https://en.wikipedia.org/wiki/Monte_Carlo_integration`