Technical University of Munich
Institute for Communications Engineering
Prof. Dr. sc. techn. Gerhard Kramer

Machine Learning for Communications

Bonus Project

Guidelines

- The project consists of 3 problems.
- Please hand in the project in a group of at most 2 students.
- Deadline: **31.01.2024 23.59 CET**.
- The solutions to the following tasks should be uploaded as one zip file to the Moodle submission form.
- The naming scheme of the zip file must be `mlcomm_project_NAME1_NAME2.zip`, where `NAMEX` should be replaced by the family name of each contributor. Additionally, include a file names.txt in your zip, which lists the names of all contributors and their matriculation numbers (one name per line and the matriculation number separated by a semicolon).
- The code for each of the three tasks should go into individual files.
- The project will be graded as either passed or failed. If it is passed a bonus of 0.3 will be given to your exam grade. However, no bonus will be given if your exam grade is worse than 4.0.
- To pass the project, you must provide a significantly good attempt to solve all the following tasks.
- Additional information for problems 1 and 2:
  - You're not allowed to add any additional libraries (everything you need is already imported).
  - Provide the solutions to theory questions in a typed pdf file (e.g., Latex). Graphs may be drawn by hand and imported. Always provide your solution first and the derivation afterwards.

– For the coding tasks, place to following files and folders (including all subfiles) into a folder called *problem_1_2* : *nn.py*, *detectors.py*, *detector_nn.py*, *trained_networks/*.

Problem 1: Neural Network Based Soft-Output Detector

Let $\boldsymbol{Y} = [\underline{y}_1, \dots, \underline{y}_m]$ be a $k \times m$ matrix containing m vectors $\underline{y}_i$ in each column. Every vector $\underline{y}_i \in \mathbb{R}^k$ is the output sequence of a communication system with corresponding input $\underline{x}_i \in \{-1, 1\}^k$ (more details on the system can be found in problem 2).
The transmit vector $\underline{x}_i$ is drawn from

$$\mathcal{C} = \{\underline{c}_1, \dots, \underline{c}_{2^k}\} \tag{1}$$

where $\underline{c}_1 = [-1, \dots, -1]^T$, $\underline{c}_2 = [-1, \dots, -1, 1]^T$, $\underline{c}_3 = [-1, \dots, 1, -1]^T$ and so on all the way to $\underline{c}_{2^k} = [1, \dots, 1]^T$.
We also define a sparse matrix $\boldsymbol{L} \in \{0, 1\}^{2^k \times m}$ where each column contains a 1 in the j-th row if $\underline{x}_i = \underline{c}_j$ and 0 elsewhere. For example, if $\underline{x}_i = [-1, \dots, 1, -1]^T$, then $(\boldsymbol{L})_{i,j}$ is 1 for $j = 3$ and 0 for $j \neq 3$. This is a so called one-hot encoding.
We would like to build a neural network which for any input vector $\underline{y}$ gives an output of length 2^k where the j-th element corresponds to the probability that $\underline{c}_j$ was transmitted. Note that we once again stack multiple input vectors into a matrix as discussed in the tutorials.

a) Open the file *nn.py*. Copy your solution from tutorial 2 wherever indicated.

b) Implement the method `__call__()` in the class `CELossSoftmax`. For input matrices $\boldsymbol{O}$ (output of neural network) and $\boldsymbol{L}$ (corresponding labels), it calculates

$$CE(\boldsymbol{L}, \boldsymbol{O}) = -\frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{2^k} (\boldsymbol{L})_{i,j} \ln\left((\boldsymbol{O})_{i,j}\right). \tag{2}$$

Don't forget to register the object with the network (see tutorial 2).
Hint: If you use matrix multiplication and `np.sum()` or `np.mean()`, you can compute this without using for-loops.

c) The softmax function for 2^k inputs $z_i \in \mathbb{R}$ has 2^k outputs a_i where

$$a_i = \frac{\mathrm{e}^{z_i}}{\sum_{j=1}^{2^k} \mathrm{e}^{z_j}}. \tag{3}$$

Argue why the outputs correspond to a probability mass function.

d) Implement the method `__call__()` in the class `Softmax`. For a $2^k \times m$ input matrix $\boldsymbol{Z}$ (output of previous linear layer), it calculates the output $\boldsymbol{A}$ where each element is

$$(\boldsymbol{A})_{i,j} = \frac{\mathrm{e}^{(\boldsymbol{Z})_{i,j}}}{\sum_{l=1}^{2^k} \mathrm{e}^{(\boldsymbol{Z})_{l,j}}} \tag{4}$$

Don't forget to register the object with the network (see tutorial 2).

Hint: Using `np.exp()` and `np.sum()`, you can implement this without any for-loop.

e) Run the script *detector_nn.py* to train your neural network and store the weights. This script calculates the achievable information rate.[1] Tweak the parameters indicated in the beginning of the file to improve the performance of your network. After submission, we will test your network with a fresh test dataset, but set the parameter `retrain_network` to false. We then use the weights and biases stored in the folder *trained_networks* (the script *detector_nn.py* stores them automatically).

[1] For those not familiar with information theory: The information rate describes the information transmitted with each channel use given the transmit constellation and detector used. Higher is better.

Problem 2: Sum-Product Algorithm Based Soft-Output Detector

Let $\underline{X} = [X_1, \ldots, X_k]^T \in \mathbb{R}^k$ be the input to a channel and $\underline{Y} = [Y_1, \ldots, Y_k]^T \in \mathbb{R}^k$ be the output. Assume

$$Y_i = H_i X_i + N_i \tag{5}$$

where N_i is white noise with $N_i \sim \mathcal{N}(0, \sigma_n^2)$ and H_i is a real-valued channel coefficient. Note that the receiver does not know the channel coefficient, but its statistics. Let $H_1 - H_2 - \ldots - H_k$ form a Markov Chain and assume Gaussian increments, that is

$$H_i | H_{i-1} = h_{i-1} \sim \mathcal{N}\left(h_{i-1}, \sigma_h^2\right) \tag{6}$$

and $H_1 \sim \mathcal{N}\left(\mu_h, \sigma_h^2\right)$. We also assume i.i.d. 2-ASK signalling $X_i \in \{-1, 1\}$ with $P_{X_i}(-1) = P_{X_i}(1) = 1/2$ for all i.

Given a received sequence $\underline{y}$, the receiver wishes to give an estimate of the transmitted sequence $\hat{\underline{x}}$ by estimating each element individually.

Throughout this problem, we will use

$$\mathcal{N}\left(x; \mu, \sigma^2\right) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right). \tag{7}$$

a) State the sequence-wise a posteriori estimate $\hat{\underline{x}}$.

b) State the marginalization and factorization required to compute the sequence-wise maximum a posteriori estimate $\hat{\underline{x}}$.

c) Draw the factor graph for $k = 3$. Use

$$f_i(\tilde{h}_i) = P(x_i) p(y_i | x_i, \tilde{h}_i)$$
$$g_i(\tilde{h}_i \tilde{h}_{i-1}) = p(\tilde{h}_i | \tilde{h}_{i-1}).$$

d) Use Bayes' Rule and the Sum-Product Algorithm to show that

$$m_{g_i \to h_i}\left(\tilde{h}_i\right) = p\left(\underline{x}_1^{i-1}, \underline{y}_1^{i-1}, \tilde{h}_i\right). \tag{8}$$

e) Show that

$$\int_{-\infty}^{\infty} \prod_{i=1}^{3} \mathcal{N}\left(x; \mu_i, \sigma_i^2\right) \mathrm{d}x = \mathcal{N}\left(\mu_1; \mu_2, \sigma_1^2 + \sigma_2^2\right) \mathcal{N}\left(\frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}; \mu_3, \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} + \sigma_3^2\right). \tag{9}$$

This will be helpful later on.

f) Show that

$$\mathcal{N}\left(ax;\mu,\sigma^2\right) = \mathcal{N}\left(a\mu;x,\sigma^2\right) \tag{10}$$

for $a \in \{-1, 1\}$.

g) Find a way to calculate $p(\underline{x}, \underline{y})$ using parts of the Sum-Product Algorithm. Calculate your algorithm analytically using $k = 3$.
Hint: Make use of the result of subtask d).

h) You can now define an algorithm for any length k by observing the impact the nodes have on incoming messages. Write down an algorithm to calculate $p(\underline{x}, \underline{y})$ for a given transmit vector $\underline{x} \in \{-1, 1\}^k$ and receive vector $\underline{y}$ as well as the channel and noise statistics, that is σ_h^2, μ_h and σ_n^2.

i) Extend the algorithm to matrix notation to efficiently compute $p(\underline{x}|\underline{y})$ for all $\underline{x} \in \{-1, 1\}^k$. You can use $\boldsymbol{X}$ to denote a matrix with all vectors in $\{-1, 1\}^k$ stored in individual columns.
Hint: You can use $p(\underline{x}, \underline{y})$ to calculate $p(\underline{x}|\underline{y})$ through

$$p(\underline{x}|\underline{y}) = \frac{p(\underline{x}, \underline{y})}{\sum_{\underline{x}' \in \{-1,1\}^k} p(\underline{x}', \underline{y})}. \tag{11}$$

j) In the file *detectors.py*, implement your algorithm in the function `detector_spa()`. Note that the input `const` corresponds to what we have defined as $\boldsymbol{X}$.
Hint: Feel free to use the function `normpdf()` which you can find in the file *detectors.py*, too.

k) Run the file *detector_spa.py* which calculates the achievable information rate for the soft-output detector. How does this compare to the achievable information rate using the neural network?

Problem 3: Expectation Maximization

You are given a dataset `EM_data.npy` (load with `np.load`) of noisy receive data points $y_i \in \mathcal{C}, i = 1, \ldots, 10000$ after transmission over an optical channel. The transmission over the optical fiber is modeled as

$$Y = \Delta X + N$$

where $\Delta \in \mathbb{C}$ and X is from a M-QAM constellation $\mathcal{X}$ of unknown size M. The noise N is zero mean circularly-symmetric complex Gaussian with unknown variance σ^2. In this task, you should do the following:

- Use the expectation maximization (EM) algorithm to determine the unknown model parameters σ^2, Δ and the distribution P_X on the constellation symbols. Print those results after performing this estimation.
- Your solution should be contained in a python file or a jupyter notebook named `mlcomm_project_problem3`.
- You can only use the following libraries: *numpy*, *matplotlib.pyplot*, *scipy.stats*.
- *Hint 1:* A scatter plot of the transmit and receive data is very helpful to gain first insights.
- *Hint 2:* Think carefully about how to initialize the algorithm. As pointed out in the lecture, an initialization with K-Means may be beneficial to get a good initial starting point for EM.