Assignment 4 – Linear and nonlinear equalization

Darko Zibar

December 2, 2024

Exercise 1 (15%)
The objective of this exercise is to implement linear discrete-time communication system model from the lecture slides, and show that the linear adaptive linear equalizer is effective in combating distortions induced by the channel. Linear discrete-time communication systems and the linear adaptive equalizer are shown on slide XX-XX. If this exercise is implemented correctly you should be able to reproduce figures similar to the ones shown on slide XX.

The data sequence, $x[k]$, consists of $+1$ and -1, i.e. $x[k] = \{\pm 1\}$ and is generated by drawing $L = 10000$ samples from the uniform distribution. The channel impulse response, $h[k]$, is described by the following function (raised cosine):

$$h[k] = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(k-2)\right)\right] & \text{for } k = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where the parameter W controls the amount of amplitude distortion produced by the channel, with the distortion increasing with W. The data sequence, $x[k]$ is passed through the channel, $h[k]$, by employing the convolution. The noise, $n[k]$ is then added to the output of the channel, i.e. $y[k] = h[k] * x[k] + n[k]$. The noise is generated by drawing samples from Gaussian distribution with zero mean and variance (noise power) of $\sigma_n^2 = 0.001$.

1. Generate and plot the data sequence
2. Use the convolution to obtain the output of the channel characterized by the impulse response $h[k]$. Set $W = 3$. Plot (stem) the output of the channel and comment on the graph. How does it compare to the input to the channel. Explain what happens with the data during convolution (hint: use the equation for convolution).
3. Implement the linear adaptive equalizer with $M = 11$ taps using the weight learning rules from the slides (gradient descent based learning). The error signal used to update the linear equalizer weights is defined as the difference between the equalizer output and the symbols, i.e. $e[k] = \hat{x}[k] - x[k - D]$. You will need to delay $x[k]$ with a certain number of samples, D, due to the response of the channel and the equalizer.
4. Plot the error squared $e^2[k]$ as a function of number of iterations. The maximum number of iterations should be 500. The learning rate should be chosen to $\eta = 0.075$. What do you observe?
5. Once the equalizer has converged and the weights of the equalizer has been learned, show that the distorted signal after the channel output can be equalized (be similar to the input data sequence $x[k]$).

6. Compute the number or errors between the equalized signal and the original data sequence $x[k]$. Show that the error rate can reach zero. Increase the noise variance to $\eta = 0.1$ and $W = 3$, and observe the error rate. Increase W to 3.5 and count the errors. (hint: use autocorrelation to find the delay between equalized and original data sequence)

Exercise 2 (25%)
In this exercise, we would like to demonstrate the capabilities of a single-layer neural network, trained by gradient descent (backpropagation), to model various functions. You will need to implement a single hidden layer neural network, including biases, with one input and one output. Use the lecture slides to see how gradient decent is used for weight updates. (Keep in mind that when training the neural network the gradient update is performed over the entire training data-set, slide. i.e. XX.)

Then, demonstrate that the implemented neural network can be used to model the following functions:

1. $f(x) = x^2$
2. $f(x) = x^3$
3. $f(x) = \sin(x)$
4. $f(x) = |x|$

To implement the training of the neural network, use the approach based the gradient descent described in the slides. Make sure that you also include bias connections at the input and after the hidden layer. The weights of the neural network should be initialized by drawing samples from a Gaussian distribution. The variance of the Gaussian distribution is a hyper-parameter that you will need to manually optimize in order for the gradient descent to converge. You will also need to do similar optimization for the number of hidden nodes and the learning rate μ. The training data should be generated by sampling $N = 300$ points from a uniform distribution between -1 and 1. i.e. $x \sim \text{Uniform}(-1, 1)$. The test data should be generated by having a regularly spaced interval between -1 and 1 in steps of 0.01 e.g. $x = [-1 : 0.01 : 1]$. The activation function should be $\tanh(\cdot)$.

In the same plot, show the training data and the modeling capabilities of the neural network implemented on the test data. Figures similar to Fig. 5.3, page 231 in the Bishop book should be produced. Also, plot the mean square error, MSE, as a function of number of iterations. By modifying initial conditions, find examples where the MSE converges but the neural network is not able to model the function. Explain what is going wrong.

Exercise 3 (35%)
The objective of this exercise is to implement the nonlinear discrete-time communication channel with the nonlinear equalizer from the lecture slides XX. You will demonstrate that the nonlinear equalizer, implemented as a multiple input single layer neural network, is effective in combating distortions induced by the nonlinear channel. If this exercise is implemented correctly you should be able to reproduce figures similar to the ones shown on slide XX, and show that the number of errors after the data has been transmitted through the nonlinear channel is is reduced when using the nonlinear equalizer compared to using the linear equalizer.

The data sequence, $x[k]$, consists of $+1$ and -1, i.e. $x[k] = \{\pm 1\}$ and is generated by drawing $L = 50000$ samples from the uniform distribution. The output from the nonlinear channel is expressed as:

$$y[k] = h[k] * x[k] + a \cdot (h[k] * x[k])^2 + n[k] \tag{2}$$

The channel impulse response, $h[k]$, is described by the following function (raised cosine):

$$h[k] = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(k-2)\right)\right] & \text{for } k = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where the parameter $W = 3$ and controls the amount of amplitude distortion produced by the channel. The parameter a controls the amount of nonlinear and it should be set to 0.8. The activation function for the neural network should be $\tanh(\cdot)$. The noise in this exercise is set to zero.

1. Generate the data sequence $x[k]$. Plot the data sequence after it has passed the nonlinear channel.

2. Implement the nonlinear adaptive equalizer, based on the multiple input neural network, with $M = 11$, i.e. the number of inputs to the neural network is 11. To implement the training of the adaptive equalizer based on the neural network, use the approach based the gradient descent described (the same as for exercise 2). The error signal used to update the linear equalizer weights is defined as the difference between the equalizer output and the symbols, i.e. $e[k] = \hat{x}[k] - x[k-D]$. You will need to delay $x[k]$ with a certain number of samples, D, due to the response of the channel and the equalizer.

3. Show that it is possible to reduce the number of errors by employing the nonlinear equalizer based on the neural network, compared to using the linear equalizer. For the nonlinear equalizer, you will need to vary the number of hidden nodes, the initialization of the neural network and the learning rate to find the best parameters. Also, since the error function used for the optimization of the neural network is nonlinear, due to the activation function, it will have many local minima. To avoid being trapped in a local minima, you will need to make sure to perform proper weight initialization. This means that you may need to run the nonlinear equalizer several times, each time with different weights initialization seed.

Exercise 4 (25%)
Implement the convolutional nonlinear equalizer from slide XX, and evaluate its performance on the nonlinear channel specified by the parameters form exercise 3.