CSE474: Pattern Recognition Sessional

Offline on Channel Equalization

The task of channel equalization is to recover a sequence of transmitted bits at the receiver after they have been distorted by a communication channel. From the theory of data communication, there are two sources of distortion in the channel: (i) inter symbol interference where an information bit is affected by previously transmitted bits, and (ii) noise, i.e., addition of unwanted signals in the channel. If the transmitted bit is I_k and the received sample is x_k, we can model the effect of channel distortion as follows:

$$x_k = f(I_k, I_{k-1}, \ldots, I_{k-(n-1)}) + n_k \quad (1)$$

Here the function $f(.)$ is the channel's impulse response and n_k denotes the added noise. We assume $f(.)$, as a linear function, i.e.,

$$x_k = \sum_{i=0}^{n-1} h_i I_{k-i} + n_k$$

where h_i are constants and depend on the channel. We further assume that noise is normally distributed with zero mean and variance σ^2.

Now, the task of equalizer is to predict the transmitted bit I_k from l successive received samples x_k, x_{k-1}, ..., x_{k-l+1}, which are grouped as a vector denoted as $\underline{x}_k$.

$$\underline{x}_k = [x_k, x_{k-1}, \ldots, x_{k-l+1}]^T$$

We denote the predicted bit with $\hat{I}_k$. We can visualize the full process as described in the theory lecture slides or text book. **For simplicity, assume $n = l = 2$** in this assignment. However, h_i 's can be any value in range 0 to 1.0, that is,

$$0 < h_i < 1$$

There are several techniques for equalization, i.e., predicting I_k from $\underline{x}_k$ such as *Cluster based* method and *Context Dependent Classification* (CDC) method (which is actually a first order Markov chain model). In this assignment you are required to implement that the latter approach, i.e., you will model the equalizer

as a context dependent classification problem using suitable selection of classes (actually clusters according to your text book) and then use Viterbi algorithm to find the most likely sequence of clusters that $\underline{x}_k$'s belong to, and thereby determine the value of $\hat{I}_k$. For the definition of clusters and other details, please refer to your text book. Here, we only mention the heart of the Viterbi algorithm in terms of the following formulas,

$$D_{\max}(\omega_{i_k}) = \max_{i_{k-1}}\left[D_{\max}(\omega_{i_{k-1}}) + d(\omega_{i_k}, \omega_{i_{k-1}})\right]$$
$$i_k, i_{k-1} = 1,2,...,M \tag{2}$$

with

$$D_{\max}(\omega_{i_0}) = 0 \tag{3}$$

where,

$$d(\omega_{i_k}, \omega_{i_{k-1}}) = \ln \hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) = \ln P(\omega_{i_k} \left| \omega_{i_{k-1}} \right.).p(\underline{x}_k \left| \omega_{i_k} \right.) \tag{4}$$

and

$$d(\omega_{i_1}, \omega_{i_0}) = \ln \hat{d}(\omega_{i_1}, \omega_{i_0}) \equiv \ln P(\omega_{i_1}) p(\underline{x}_i \left| \omega_{i_1} \right.) \tag{5}$$

The CDC approach can be implemented in **two different ways** and **you need to implement both** of them. The approaches differ how you calculate $d(\omega_{i_k}, \omega_{i_{k-1}})$ in Equation (2). One way (**Method# 1**) is to estimate all transitional, prior and observation probabilities from training data file and use Equations (4) and (5). In this case, Equation (2) is maximized. The other approach (**Method# 2**) is to define $d(\omega_{i_k}, \omega_{i_{k-1}})$ as a cost function as follows

$$d(\omega_{i_k}, \omega_{i_{k-1}}) = d_{\omega_{i_k}}(\mathbf{x}_k) = \left\| \mathbf{x}_k - \mu_{i_k} \right\| \tag{6}$$

where, μ_{i_k} is a cluster center (centroid) which can be estimated from given training file. In this case, Equation (2) is searched for minimum. Though there are two methods, parameters of both methods can be calculated from a training file in a single pass.

Tasks:

You will be given a train file "train.txt". It is nothing but a very large bit string, with 0's and 1's appearing randomly. Your program will also take as **input** the channel co-efficients (h_i's), and the variance of the noise introduced in the channel. h_i's and variance can be loaded from another file. **While evaluating your program, the instructor may choose different values for h_i's and variance.**

Now follow the steps below.

1. Use the train file and h_i's and variance to build the CDC model for which we would need the following:

- Defining the states (e.g., clusters along with their centroids). h_i's and n and l indicate their number.
- Determining the prior and transition probabilities
- Determining the observation probabilities

You can assume that the train file is being transmitted bit-wise. Therefore for every overlapping sequence of n bits you can calculate an x_k using Equation (1). From every overlapping 2 x_k's, form vectors $\underline{x}_k$'s. Each $\underline{x}_k$ belong to a cluster. If you accumulate them, you will be able to determine clusters, cluster centroids, transition probabilities, initial probabilities. To determine observation probabilities, assume the noise is normally distributed, the observation probability should also follow multivariate normal distribution. Hence, you need only to find the mean and covariance of the distributions. The means are actually cluster centroids as observations are vectors, e.g., $\underline{x}_k$. If you accumulate the $\underline{x}_k$ vectors (observations) for each cluster, you can easily determine the covariance matrix using any library function. Now the probability density function $p(\underline{x}_k | \omega_{i_k})$ needed in Equation (4) and (5) can be trivially calculated using your knowledge gained in Assignment 1 (multivariate Bayesian classifier).

2. After the model is built, we need to test the model i.e., the test phase. Here you will transmit a sequence of bits (definitely from a test file), and from the received l sample valuess, x_k, x_{k-1}, ... x_{k-l+1} you need to estimate the transmitted bit sequence, $\hat{I}_k$ by using two implementation methods of Viterbi algorithm on your Markov chain model.

3. Compare the original signal and the estimated signals to find out the accuracy.

4. To simulate the channel, you need to define a class whose member will be the channel parameters: (i) the h_i's of the impulse response, (ii) variance of channel noise. All these parameters are to be taken as input from file. To simulate noise, you need to generate a normal random variable with the zero mean and given variance.

Inputs:

1. parameter.txt: **you will create this file.** This file contains co-efficients (h_i's), and the variance of the noise
2. train.txt: **we will provide this file.** Use this file to determine all necessary probability values, cluster centroids (means), covariance matrixes.
3. Test.txt: **we will provide this file.** This has the same structure as train.txt. You should pass the corresponding values of this file through the channel, and then form the received sample values (that has their own errors) using Equation (1). Use Method 1 and Method 2 to determine what was sent. Definitely these methods will **ONLY** use the received sample values and the parameters leaned in Step 2. Write the inferred bit-strings to two different files named **out1.txt** (for Method 1) and **out2.txt** (for Method 2). Obviously, if you implement correctly, out1.txt, ou2.txt and test.txt should be the same.

Outputs:

1. Out1.txt
2. Out2.txt2
3. % of accuracy

In sessional, we will test with various files and channel and noise parameters.

Submission deadline

11:55 pm 15 February 2022