FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO, UNIVERZA V LJUBLJANI

MATHEMATICAL MODELLING

FIRST HOME ASSIGNMENT

Report

Author

Jernej Vivod

Lecturer

Dr. Neža Mramor Kosta

March 22, 2018

Contents

1 Problem

Problem Statement

We have an unknown system, which responds with an output signal y(t) to an input signal x(t). An output value at time t can be expressed as a linear combination of current and possibly also past inputs x.

Let n denote the size of vector of coefficients h and let L denote the size of the input vector X. Since we have to look $n-1$ inputs back for each output y, we will only be able to predict $L-(n-1)$ outputs.

MATLAB/Octave Implementation - Goal

Our task is to write two MATLAB/Octave functions.

The first function ***movavg*** computes the vector of coefficients h for given inputs x and outputs y. The parameter n tells how large the vector of coefficients h should be.

The second function called ***prediction*** computes the output values y given input values x and vector of coefficients h.

The function headers are defined as:

```
function [y] = prediction(x, h)

function [h] = movavg(x, y, n)
```

2 Solution

Bulding the Matrix A

The matrix corresponding to the system of equations will contain $L-(n-1)$ rows and n columns, where L is the number of measurements ($L = N-1$). Each row contains the past inputs and the current input that are used to compute y_k, where $k \in [n-1,\, N]$. This means that, for example, if $n = 2$, y_k will be a linear combination of x_{k-1} and x_k. The matrix A will therefore contain 2 columns and $L-1$ rows. In this specific case we will not be able to compute y_0 as there are not enough x values to form the corresponding linear combination.
The matrix A can be written as such:

$$\begin{bmatrix} x(n-1-n+1) & .. & x(n-1) \\ \vdots & & \vdots \\ x(N-n+1) & .. & x(N) \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} y(n-1) \\ \vdots \\ y(N) \end{bmatrix}$$

the value of output y at time t can be described as a linear combination of past and present input values x as
$y(t) = h_1 x(t-n+1) + h_2 x(t-n+2) + \cdots + h_{n-1} x(t-1) + h_n x(t)$
or written equivalently as
$y_k = h_1 x_{k-n+1} + h_2 x_{k-n+2} + \cdots + h_{n-1} x_{k-1} + h_n x_k$.

Computing the Vector y from Known A and h

If we know the vector of coefficients h (which has size n), we can use it to compute exactly $L-(n-1)$ outputs y, where L is the number of inputs x. This is true because we have to look $n-1$ inputs back to compute the value y at current time t. There are not enough inputs x to form the corresponding linear combinations for outputs y with indices smaller than $n-1$. The values for output y are computed by simply solving the system of equations $Ah = y$ for y by multiplying the matrix A with vector of coefficients h.

The Accuracy of Predicting y as the Size of h Changes

The following graph shows how the accuracy of prediction for output y differs from the actual output values as the size of the vector h changes. The mean prediction error was computing by averaging the absolute values of the differences between predicted and actual y values. The coefficients vector h was computed from the training set train-io.txt and was then used to try to predict the output vector y in io-test.txt. The constant n was taken from $n \in [1,\ 50]$.

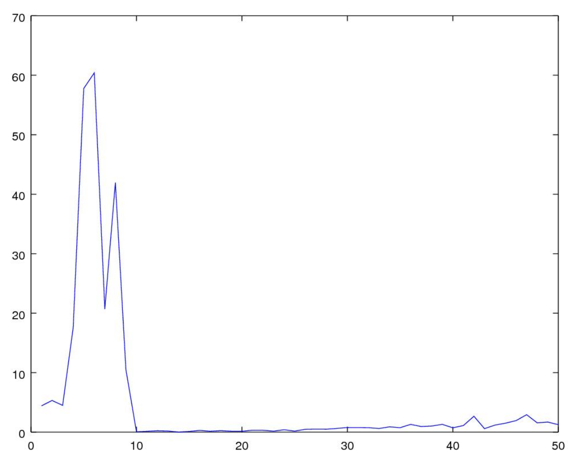

Figure 1: Mean Prediction Error with respect to n

We can see that the error reaches its lowest point at around $n \in [10, 20]$ and then slowly begins to increase again as n increases.

Computing Coefficients h if Input and Output Values are Constant

What would the vector of coefficients look like if every input and output value were a constant? Let a be a constant value and let $x_i = y_i = a,\ i \in [0,\ N]$. The matrix representation of the system of linear equations $Ah = y$ can be represented as:

$$\begin{bmatrix} a & .. & a \\ \vdots & & \vdots \\ a & .. & a \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} a \\ \vdots \\ a \end{bmatrix}$$

The linear combination of an arbitrary y value can be written as follows:

$ah_1 + \cdots + ah_n = a$

From this it follows that

$a(h_1 + \cdots + h_n) = a$

$h_1 + \cdots + h_n = 1.$

This shows that every vector of coefficient h whose components add up to 1 will represent a solution of the system.

If we require that $\forall i, j \in [1, n] : h_i = h_j$, we can write that $\forall i \in [1, n] : h_i = \frac{1}{n}$

3 Implementation in MATLAB/Octave

This section lays out a well-commented MATLAB/Octave implementation of the movavg and prediction functions described in the problem statement in section 1.

movavg

```
function [h] = movavg(x, y, n)

  % initialize empty matrix A
  A = [];
  % Initialize empty matrix for storing the next column to add to matrix A.
  next_column = [];

  % Outer loop runs on [n - 1, 0].
  % (the quantities subtracted from current t)
  for sub = n - 1 : -1 : 0

    % Inner loop runs on [n, N].
    % (quantities representing the current t)
    for k = n : size(x)(2)
      % Add appropriate x value to next_column vector.
      next_column = [next_column; x(k - sub)];
    end
    % Put constructed column into the A matrix.
    A = [A, next_column];
    % Prepare the next_column vector for next iteration (empty it).
    next_column = [];
  end
  % Before solving, vector y must be truncated as
  % we can only predict values y(t) where t >= n - 1.
  y = y(:, n:end);

  % Make y a column vector and solve system of equations for h.
  h = A\y';
  h = h';
endfunction
```

prediction

```
function [h] = movavg(x, y, n)

  % initialize empty matrix A
  A = [];
  % Initialize empty matrix for storing the next column to add to matrix A.
  next_column = [];

  % Outer loop runs on [n - 1, 0].
  % (the quantities subtracted from current t)
  for sub = n - 1 : -1 : 0

    % Inner loop runs on [n, N].
    % (quantities representing the current t)
    for k = n : size(x)(2)
      % Add appropriate x value to next_column vector.
      next_column = [next_column; x(k - sub)];
    end
    % Put constructed column into the A matrix.
    A = [A, next_column];
    % Prepare the next_column vector for next iteration (empty it).
    next_column = [];
  end
  % Before solving, vector y must be truncated as
  % we can only predict values y(t) where t >= n - 1.
  y = y(:, n:end);

  % Make y a column vector and solve system of equations for h.
  h = A\y';
  h = h';
endfunction
```