Assignment 4.docx
DOCX 61.26KB

answer question

Thought for 123 seconds

- First, the problem is about system identification using a convex combination of two systems. The overall predicted output is $\hat{y}(n) = \mu\hat{y}_1(n) + (1-\mu)\hat{y}_2(n)$. I need to develop algorithms for updating the weights $w_1(n)$, $w_2(n)$, and the mixing parameter $\mu$.

  From the diagram, it seems like there are two adaptive filters or systems, System 1 and System 2, each with their own weights, $w_1$ and $w_2$. The output of each system is $\hat{y}_1(n)$ and $\hat{y}_2(n)$. The true output is $y(n)$, and we have errors: local errors $e_1(n) = y(n) - \hat{y}_1(n)$ and $e_2(n) = y(n) - \hat{y}_2(n)$, and overall error $e(n) = y(n) - \hat{y}(n)$.

  The goal is to update $w_1$, $w_2$, and $\mu$ to minimize the overall error or something similar. The problem says "develop algorithm for the updating", so I need to derive update equations.

  I recall that in adaptive filtering, we often use gradient-based methods like LMS (Least Mean Squares) for updating weights. Here, we have two sets of weights and a mixing parameter.

  Let me denote the input vector as $x(n)$. Then, $\hat{y}_1(n) = w_1^T(n)x(n)$ and $\hat{y}_2(n) = w_2^T(n)x(n)$. But the diagram shows $x(n)$ going to both systems, so yes.

  The overall output is $\hat{y}(n) = \mu\hat{y}_1(n) + (1-\mu)\hat{y}_2(n)$.

  Now, we need to update $w_1$, $w_2$, and $\mu$. Probably, we want to minimize the mean square error, $E[e^2(n)]$, where $e(n) = y(n) - \hat{y}(n)$.

  Since it's an adaptive setting, we might use instantaneous gradient for LMS.

  Let me start with updating $w_1$ and $w_2$.

  Consider the cost function for $w_1$. If we think about it, $w_1$ affects $\hat{y}_1(n)$, which affects $\hat{y}(n)$, and thus $e(n)$. So, to update $w_1$, we can use the gradient of $e^2(n)$ with respect to $w_1$.

  Similarly for $w_2$.

  And for $\mu$, we need the gradient with respect to $\mu$.

  So, let's compute the gradients.

  First, the overall error: $e(n) = y(n) - \hat{y}(n) = y(n) - [\mu\hat{y}_1(n) + (1-\mu)\hat{y}_2(n)]$

  Now, the instantaneous cost is $J(n) = \frac{1}{2}e^2(n)$ (often we use 1/2 for convenience in gradient).

  But for LMS, we usually don't include 1/2, but it doesn't matter since step size can adjust.

Let me define the update for $w_1$.

The gradient of $J(n) = \frac{1}{2}e^2(n)$ with respect to $w_1$ is:

$$\frac{\partial J(n)}{\partial w_1} = e(n)\frac{\partial e(n)}{\partial w_1}$$

Now, $\frac{\partial e(n)}{\partial w_1} = \frac{\partial [y(n) - \hat{y}(n)]}{\partial w_1} = -\frac{\partial \hat{y}(n)}{\partial w_1}$

And $\frac{\partial \hat{y}(n)}{\partial w_1} = \frac{\partial [\mu \hat{y}_1(n) + (1-\mu)\hat{y}_2(n)]}{\partial w_1} = \mu \frac{\partial \hat{y}_1(n)}{\partial w_1} + (1-\mu)\frac{\partial \hat{y}_2(n)}{\partial w_1}$

But $\hat{y}_2(n)$ does not depend on $w_1$, so $\frac{\partial \hat{y}_2(n)}{\partial w_1} = 0$. Similarly, $\hat{y}_1(n) = w_1^T x(n)$, so $\frac{\partial \hat{y}_1(n)}{\partial w_1} = x(n)$.

Thus, $\frac{\partial \hat{y}(n)}{\partial w_1} = \mu x(n)$

Therefore, $\frac{\partial e(n)}{\partial w_1} = -\mu x(n)$

So, $\frac{\partial J(n)}{\partial w_1} = e(n)(-\mu x(n)) = -\mu e(n) x(n)$

In LMS, we update weights in the negative gradient direction, so:

$$w_1(n+1) = w_1(n) - \eta_1 \frac{\partial J(n)}{\partial w_1} = w_1(n) + \eta_1 \mu e(n) x(n)$$

Where $\eta_1$ is the step size for $w_1$.

Similarly, for $w_2$:

$$\frac{\partial J(n)}{\partial w_2} = e(n)\frac{\partial e(n)}{\partial w_2}$$

$\frac{\partial e(n)}{\partial w_2} = -\frac{\partial \hat{y}(n)}{\partial w_2}$

$\frac{\partial \hat{y}(n)}{\partial w_2} = \frac{\partial [\mu \hat{y}_1(n) + (1-\mu)\hat{y}_2(n)]}{\partial w_2} = \mu \frac{\partial \hat{y}_1(n)}{\partial w_2} + (1-\mu)\frac{\partial \hat{y}_2(n)}{\partial w_2}$

Since $\hat{y}_1(n)$ does not depend on $w_2$, $\frac{\partial \hat{y}_1(n)}{\partial w_2} = 0$, and $\hat{y}_2(n) = w_2^T x(n)$, so $\frac{\partial \hat{y}_2(n)}{\partial w_2} = x(n)$

Thus, $\frac{\partial \hat{y}(n)}{\partial w_2} = (1-\mu)x(n)$

So, $\frac{\partial e(n)}{\partial w_2} = -(1-\mu)x(n)$

Therefore, $\frac{\partial J(n)}{\partial w_2} = e(n)(-(1-\mu)x(n)) = -(1-\mu)e(n)x(n)$

Then, update for $w_2$:

$$w_2(n+1) = w_2(n) - \eta_2 \frac{\partial J(n)}{\partial w_2} = w_2(n) + \eta_2(1-\mu)e(n)x(n)$$

Now, for $\mu$, we need to update it. Similarly, we take gradient with respect to $\mu$.

$$\frac{\partial J(n)}{\partial \mu} = e(n)\frac{\partial e(n)}{\partial \mu}$$

$\frac{\partial e(n)}{\partial \mu} = -\frac{\partial \hat{y}(n)}{\partial \mu}$

Now, $\frac{\partial \hat{y}(n)}{\partial \mu} = \frac{\partial [\mu \hat{y}_1(n) + (1-\mu)\hat{y}_2(n)]}{\partial \mu} = \hat{y}_1(n) - \hat{y}_2(n)$

So, $\frac{\partial e(n)}{\partial \mu} = -(\hat{y}_1(n) - \hat{y}_2(n)) = -(\hat{y}_1(n) - \hat{y}_2(n))$

Thus, $\frac{\partial J(n)}{\partial \mu} = e(n)[-(\hat{y}_1(n) - \hat{y}_2(n))] = -e(n)(\hat{y}_1(n) - \hat{y}_2(n))$

Now, we update $\mu$ in the negative gradient direction. But note that $\mu$ is a mixing parameter, and it should be between 0 and 1 because it's a convex combination. So, we need to ensure that after update, $\mu$ remains in [0,1]. One way to handle this is to use a sigmoid function or to project onto [0,1]. Alternatively, we can use a gradient update and then clip the value.

Let me denote the update for $\mu$. Let $\mu(n)$ be the value at time n.

Then, $\mu(n+1) = \mu(n) - \eta_\mu \frac{\partial J(n)}{\partial \mu} = \mu(n) + \eta_\mu e(n)(\hat{y}_1(n) - \hat{y}_2(n))$

Where $\eta_\mu$ is the step size for $\mu$.

But after this update, $\mu$ might be outside [0,1]. So, we can clip it: if $\mu(n+1) > 1$, set to 1; if $\mu(n+1) < 0$, set to 0.

Alternatively, we can define $\mu$ as a function of another parameter, say $\alpha$, such that $\mu = \sigma(\alpha) = \frac{1}{1+e^{-\alpha}}$, which always between 0 and 1. Then we update $\alpha$ instead.

In many papers, for convex combinations, they often use the sigmoid approach to keep $\mu$ in [0,1].

But let's see what the problem expects. The diagram shows $\mu$ and $1 - \mu$, so probably $\mu$ should be between 0 and 1.

In the update for $w_1$ and $w_2$, we have $\mu$ in the update equations, so if $\mu$ is not constrained, it might cause issues.

I think for simplicity, we can assume that we update $\mu$ directly and then clip it.

But let me check the literature. I recall that in some combination schemes, they use the LMS on the mixing parameter.

Another thing: in the diagram, there is "Algorithm1" and "Algorithm2" for updating $w_1$ and $w_2$, and probably another for $\mu$.

Also, from the diagram, there is $v(n)$ which might be noise, but I think we can ignore that for now.

So, to summarize, the algorithms for updating:

For $w_1$:

$$w_1(n+1) = w_1(n) + \eta_1 \mu e(n) x(n)$$

For $w_2$:

$$w_2(n+1) = w_2(n) + \eta_2 (1-\mu) e(n) x(n)$$

For $\mu$:

$$\mu(n+1) = \mu(n) + \eta_\mu e(n)(\hat{y}_1(n) - \hat{y}_2(n))$$

Then, clip $\mu(n+1)$ to [0,1] if necessary.

But in the update for $\mu$, we have $e(n)(\hat{y}_1(n) - \hat{y}_2(n))$, which seems correct.

We can also think about the local errors. Notice that $e(n) = y(n) - \hat{y}(n)$, but from the local errors, $e_1(n) = y(n) - \hat{y}_1(n)$ and $e_2(n) = y(n) - \hat{y}_2(n)$, so $\hat{y}_1(n) - \hat{y}_2(n) = -(e_1(n) - e_2(n))$, because $\hat{y}_1(n) - \hat{y}_2(n) = [y(n) - e_1(n)] - [y(n) - e_2(n)] = -e_1(n) + e_2(n) = e_2(n) - e_1(n)$

So, $\hat{y}_1(n) - \hat{y}_2(n) = e_2(n) - e_1(n)$

Thus, the update for $\mu$ can be written as:

$$\mu(n+1) = \mu(n) + \eta_\mu e(n)(e_2(n) - e_1(n))$$

Which might be useful.

Now, in the diagram, there is "Algorithm" pointing to $\mu$, and "Algorithm1" and "Algorithm2" for $w_1$ and $w_2$, so probably we have three update rules.

Also, the diagram shows $w^o$ which is the true weight, but we don't know that.

So, I think this is the algorithm.

But let me write it properly.

The algorithm should be iterative for each time n.

At each time n:

1. Get input x(n) and desired output y(n).
2. Compute outputs:
   $\hat{y}_1(n) = w_1^T(n)x(n)$
   $\hat{y}_2(n) = w_2^T(n)x(n)$
3. Compute overall output:
   $\hat{y}(n) = \mu(n)\hat{y}_1(n) + (1 - \mu(n))\hat{y}_2(n)$
4. Compute errors:
   $e_1(n) = y(n) - \hat{y}_1(n)$
   $e_2(n) = y(n) - \hat{y}_2(n)$
   $e(n) = y(n) - \hat{y}(n)$
5. Update weights:
   $w_1(n+1) = w_1(n) + \eta_1 \mu(n) e(n) x(n)$
   $w_2(n+1) = w_2(n) + \eta_2 (1 - \mu(n)) e(n) x(n)$
6. Update mixing parameter:
   $\mu(n+1) = \mu(n) + \eta_\mu e(n)(\hat{y}_1(n) - \hat{y}_2(n))$
   Or equivalently, $\mu(n+1) = \mu(n) + \eta_\mu e(n)(e_2(n) - e_1(n))$
7. If necessary, clip $\mu(n+1)$ to [0,1].

This seems complete.

But in the update for $w_1$ and $w_2$, we are using the overall error e(n), not the local errors. That makes sense because we want to minimize the overall error.

Also, note that in the update for $w_1$, the step size is scaled by $\mu$, so if $\mu$ is small, the update for $w_1$ is small, which is intuitive because if $\mu$ is small, $w_1$ contributes less to the output.

> Similarly for $w_2$.
>
> For $\mu$, the update depends on the difference between the outputs or errors.
>
> I think this is correct.
>
> So, for the answer, I should write the update equations as above.
>
> Now, to present it neatly.

The system identification problem involves estimating an unknown system using a convex combination of two adaptive systems. The overall predicted output is given by $\hat{y}(n) = \mu\hat{y}_1(n) + (1-\mu)\hat{y}_2(n)$, where $\hat{y}_1(n)$ and $\hat{y}_2(n)$ are the outputs of System 1 and System 2, respectively. The local errors are $e_1(n) = y(n) - \hat{y}_1(n)$ and $e_2(n) = y(n) - \hat{y}_2(n)$, and the overall error is $e(n) = y(n) - \hat{y}(n)$. The weights $w_1(n)$ and $w_2(n)$ for the two systems and the mixing parameter $\mu(n)$ are updated using gradient-based algorithms to minimize the mean square error.

## Algorithm for Updating $w_1(n)$, $w_2(n)$, and $\mu(n)$

At each time step $n$:

1. **Compute outputs**:
   - $\hat{y}_1(n) = w_1^T(n)x(n)$
   - $\hat{y}_2(n) = w_2^T(n)x(n)$
   - $\hat{y}(n) = \mu(n)\hat{y}_1(n) + (1-\mu(n))\hat{y}_2(n)$
2. **Compute errors**:
   - $e_1(n) = y(n) - \hat{y}_1(n)$
   - $e_2(n) = y(n) - \hat{y}_2(n)$
   - $e(n) = y(n) - \hat{y}(n)$
3. **Update weights**:
   - $w_1(n+1) = w_1(n) + \eta_1\mu(n)e(n)x(n)$
   - $w_2(n+1) = w_2(n) + \eta_2(1-\mu(n))e(n)x(n)$
     where $\eta_1$ and $\eta_2$ are step sizes for $w_1$ and $w_2$, respectively.
4. **Update mixing parameter**:
   - $\mu(n+1) = \mu(n) + \eta_\mu e(n)\left(\hat{y}_1(n) - \hat{y}_2(n)\right)$
     or equivalently,
     $\mu(n+1) = \mu(n) + \eta_\mu e(n)\left(e_2(n) - e_1(n)\right)$
     where $\eta_\mu$ is the step size for $\mu$. To ensure $\mu$ remains in $[0,1]$, clip $\mu(n+1)$ after update:
   - If $\mu(n+1) > 1$, set $\mu(n+1) = 1$

- If $\mu(n+1) < 0$, set $\mu(n+1) = 0$

## Explanation

- The updates for $w_1(n)$ and $w_2(n)$ are derived from the gradient of the instantaneous cost function $J(n) = \frac{1}{2}e^2(n)$ with respect to the weights. The step sizes $\eta_1$ and $\eta_2$ control the convergence rate.
- The update for $\mu(n)$ is derived from the gradient of $J(n)$ with respect to $\mu$. Clipping ensures the convex combination remains valid.
- The algorithm uses the overall error $e(n)$ for updates, ensuring minimization of the overall prediction error.

This approach allows adaptive combination of two systems to model the unknown system effectively.