Reading Asst

Review of RLS:

given: $\{u(j), d(j)\}\Big|_{j=1}^{i-1}$

$$w(i-1) \Rightarrow \min_{w} \sum_{j=1}^{i-1} |d(j) - u(j)^T w|^2 \rightarrow (1)$$

$u(j) \Rightarrow$ input at j^{th} instant $\in L \times 1$

$d(j) \Rightarrow$ Desired response

Soln:

let $U(i-1) = [u(1), \ldots, u(i-1)]_{L \times i-1}$

$d(i-1) = [d(1), \ldots d(i-1)]^T$ } $\in 1 \times i-1$

cost fn

$\Rightarrow$ $J(w) = \| d(i-1) - U^T w \|^2$

$\frac{\partial J}{\partial w} = 2(d(i-1) - U^T w)$

$= -2 . U (d(i-1) - U^T w) = 0$

$\Rightarrow$ $\boxed{w = (U U^T)^{-1} U d}$

$\Rightarrow$ $\boxed{w(i-1) = [U(i-1) U^T(i-1)]^{-1} U(i-1) d(i-1)}$

But there's a problem, for every 'i',

$\Rightarrow$ The task become tedious

Hence, we need to come up with a recursive algo.

$\hookrightarrow$ Hence, $P(i-1) = \underbrace{[U(i-1) U^T(i-1)]^{-1}}_{L \times L}$

Need to find a relation b/w $P(i)$ & $P(i-1)$

As proved in earlier class

$$\Rightarrow P(i) = \left[P(i-1) - \frac{P(i-1)\,u(i)\,u(i)^T P(i-1)}{1 + u(i)^T P(i-1)\,u(i)} \right]$$

$w(i) = P(i)\,U(i)\,d(i)$

And thus, it's reduced to

$$\Rightarrow w(i) = w(i-1) + \underbrace{\frac{P(i-1)\,u(i)}{1 + u(i)^T P(i-1)\,u(i)}}_{\text{gain}} \underbrace{\left[d(i) - u(i)^T w(i-1) \right]}_{\text{posterior error}}$$

Algo of RLS

$\Rightarrow$ Init: $w(0) = 0$, $P(0)$.

$\Rightarrow$ Iteration: $i \geq 1$:-

$r(i) = 1 + u(i)^T P(i-1)\,u(i)$

gain :- $k(i) = P(i-1)\,u(i) / r(i)$

post. error :- $d(i) - u(i)^T w(i-1) = e(i)$

$\Rightarrow$ ~~w(i)~~

- Weight updation:

$$w(i) = w(i-1) + k(i)\, e(i)$$

$$P(i) = P(i-1) - (k(i)\, k^T(i)\, r(i))$$

---

* Kernel RLS *

* We use Mercer thm:

$$u(i) \longrightarrow \phi(u(i))$$

given; $\{d(1), d(2), \ldots\}$ & $\{\phi(1), \phi(2), \ldots\}$

Here, wt vector,

$$\min_{w} \sum_{j=1}^{i} |d(j) - w^T \phi(j)|^2 + \underbrace{\lambda \|w\|^2}_{\text{Regularisation term.}}$$

* Since, $\phi(.) \Rightarrow$ can be of high dimensions we need a regulariser so that weights don't shoot up.

$d(i) = [d(1) \ldots d(i)]^T \} \; i \times 1$

$\phi(i) = [\phi(1), \ldots \phi(i)]$ $\quad L \times i$

Similar to regularised least-squares

$$J(w) = \sum_{j=1}^{i} |d(j) - w^T \phi(j)|^2 + \lambda \|w\|^2$$

$$\frac{\partial J}{\partial w} = \cancel{J(w) = \| d(i) - w^T \phi}$$

$w \rightarrow L \times 1$

$$\Rightarrow J(w)_i = \| d(i) - \phi^T w \|^2 + \lambda \|w\|^2$$

$$\frac{\partial J}{\partial w_i} = \cancel{-(d(i)} \; -2\phi(d(i) - \phi^T w) + 2\lambda w = 0$$

$$= (\lambda I + \phi \phi^T) w = \phi d(i)$$

$$\Rightarrow \boxed{w = (\lambda I + \phi \phi^T)^{-1} \phi d(i)}$$

$$\boxed{w(i) = (\lambda I + \phi(i) \phi(i)^T)^{-1} \phi(i) d(i)}$$

Proving through Matrix inversion lemma

To prove:

$$(\lambda I + \phi\phi^T)^{-1}\phi = \phi(\lambda I + \phi^T\phi)^{-1}$$

Using

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

→ Let $A = \lambda I$, $B = \phi$, $C = I$, $D = \phi^T$

LHS $= [\lambda^{-1}I - \lambda^{-2}\phi(I + \lambda^{-1}\phi^T\phi)^{-1}\phi^T]\phi$

$$= \lambda^{-1}\phi - \lambda^{-2}\phi(I + \lambda^{-1}\phi^T\phi)^{-1}\phi^T\phi \rightarrow (1)$$

RHS $=$ $A: \lambda I$, $B = \phi^T$, $C = I$, $D = \phi$

$$\rightarrow \phi[\lambda^{-1}I - \lambda^{-2}\phi^T[I + \lambda^{-1}\phi\phi^T)]^{-1}\phi]$$

$$= \lambda^{-1}\phi - \lambda^{-2}\phi\phi^T[I + \lambda^{-1}\phi\phi^T]^{-1}\phi \rightarrow (2)$$

⇒ Consider (1) - (2)

$$\Rightarrow \lambda^{-1}\phi - \lambda^{-2}\phi(I + \lambda^{-1}\phi^{T}\phi)^{-1}\phi^{T}\phi - \cancel{\lambda^{-1}\phi}$$

$$+ \lambda^{-2}\phi\phi^{T}[I + \lambda^{-1}\phi\phi^{T}]^{-1}\phi$$

$$\Rightarrow \lambda^{-2}\phi[\phi^{T}[I + \lambda^{-1}\phi\phi^{T}]^{-1} - [I + \lambda^{-1}\phi^{T}\phi]^{-1}\phi^{T}]\phi$$

$$= \lambda^{-1}\phi[\phi^{T}[\lambda I + \phi\phi^{T}]^{-1} - [\lambda I + \phi^{T}\phi]^{-1}\phi^{T}]\phi \quad \rightarrow (3)$$

$\Rightarrow$ Set $(3) \leq T$

$\Rightarrow$ Let $T = (\lambda I + \phi\phi^{T})^{-1}\phi - \phi(\lambda I + \phi^{T}\phi)^{-1}$

$$\Rightarrow T^{T} = \phi^{T}[\lambda I + \phi\phi^{T}]^{-1} - [\lambda I + \phi^{T}\phi]^{-1}\phi^{T}$$

$\Rightarrow$ we have $T = \lambda^{-1}(\phi T^{T}\phi)$

$\Rightarrow$ this to hold true,

$$\boxed{T = 0}$$

$\Rightarrow$ Hence $(1) - (2) = 0$

$\Rightarrow$ LHS = RHS

we transformed $\phi\phi^T \to \phi^T\phi$

where we can find using the Kernel trick. {done using matrix inversion}

$$\Rightarrow w(i) = \phi(i)\,[\lambda I + \underbrace{\phi(i)^T \phi(i)}_{\text{Kernel trick}}]^{-1} d(i)$$

dim

let $w(i) = \phi(i)\,a(i)$ — } $L \times 1$

$\Rightarrow a(i) = [\lambda I + \phi^T(i)\,\phi(i)]^{-1} d(i)$ } $i \times 1$

let $Q(i) : [\lambda I + \phi^T(i)\,\phi(i)]^{-1}$ } $i \times i$

$\Rightarrow$ $Q(i)$ } $i \times i$ dim

$Q(i-1)$ } $i-1 \times i-1$ dim

$$\Rightarrow [Q(i)]^{-1} = \begin{bmatrix} Q(i-1)^{-1} & h(i) \\ h^T(i) & \lambda + \phi^T(i)\,\phi(i) \end{bmatrix}$$

i.e., we need to add a col' & row from $Q(i)$ to $Q(i-1)$.

$Q(i)^{-1} = \lambda I + \phi^T(i)\phi(i)$

$Q(i-1) = \lambda I + \phi^T(i-1)\phi(i-1)$

$$\phi(i-1) = \begin{bmatrix} \phi(1) & \cdots & \phi(i-1) \end{bmatrix}_{L \times i-1}$$

$$\Rightarrow \phi^T\phi = \begin{bmatrix} \phi(1) \\ \vdots \\ \phi(i-1) \end{bmatrix} \begin{bmatrix} \phi(1) & \cdots & \phi(i-1) \end{bmatrix}$$

If another data point adds i.e.,

To find $\phi^T(i)\phi(i) \Rightarrow \begin{bmatrix} \phi^T_{(i-1)}\phi_{(i-1)} & \phi^T(i-1)\phi(i) \\ \phi^T(i)\phi(i-1) & \|\phi(i)\|^2 \end{bmatrix}$

$$\Rightarrow [Q(i)]^{-1} = \begin{bmatrix} \lambda I_{(i-1 \times i-1)} + \phi^T_{(i-1)}\phi_{(i-1)} & \phi^T(i-1)\phi(i) \\ \phi^T_{(i)}\phi(i-1) & \lambda + \|\phi(i)\|^2 \end{bmatrix}$$

$$\Rightarrow [Q(i)]^{-1} = \begin{bmatrix} Q(i-1)^{-1} & h(i) \\ h^T(i) & \lambda + \phi^T(i)\phi(i) \end{bmatrix} \rightarrow (1)$$

Here, $h(i) = \phi^T(i-1)\phi(i)$

* But we need to find a relation

b/w $Q(i)$ & $Q(i-1)$

⇒ **Block matrix inversion method:**

$$\Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A-BD^{-1}C)^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & (D-CA^{-1}B)^{-1} \end{bmatrix}$$

From (1):

$A \rightarrow Q(i-1)^{-1}$

$B \rightarrow h(i)$

$C \rightarrow h(i)^T$

$D \rightarrow \lambda + \phi(i)^T \phi(i)$

$$\Rightarrow \left[(Q(i))^{-1}\right]^{-1} = Q(i) = r(i)^{-1} \begin{bmatrix} Q(i-1)r(i) + z(i)z(i)^T & -z(i) \\ -z(i)^T & 1 \end{bmatrix}$$

$r(i) = \lambda + \phi(i)^T \phi(i) - z(i)^T h(i)$

$z(i) = Q(i-1)h(i)$

$a(i) = Q(i)\, d(i) \rightarrow d(i) = \begin{bmatrix} d(i-1) \\ d(i) \end{bmatrix}$

$$a(i) = \begin{bmatrix} a(i-1) - z(i)\, r(i)^{-1} e(i) \\ r(i)^{-1} e(i) \end{bmatrix}$$

$e(i)$ is the posterior error

→ $\boxed{w(i) = \phi(i)\, a(i)}$

⇒ We now have a recursive relation

↳ $e(i) = d(i) - f_{i-1}(u(i))$

$= d(i) - h(i)^T a(i-1)$

Points to remember

↳ Unlike RLS, as we iterate, our the dimensions of $a(i)$, $Q(i)$ increases,

→ this is due to our transformation from $\phi\phi^T \rightarrow \phi^T\phi$, to use kernel trick,

* Diff. b/w KLMS, KRLS :

→ KLMS & KRLS is similar

But, for every iter.

→ we add $u(i)$ with coeff $r(i)^{-1} e(i)$

→ KRLS updates by $-z(i) r(i)^{-1} e(i)$

But KLMS, never updates prev. coeff.

Let f : estimate of i/p – o/p map.

$$f_i = f_{i-1} + r(i)^{-1} \left[k(u(i), \cdot) - \sum_{j=1}^{i-1} z_j(i)\, k(u(j), \cdot) \right]$$

⇒

Algorithm

Init:

$Q(1) = (\lambda + K(u(1), u(1))^{-1}$

$a(1) = Q(1)\, d(1)$

→ Computation:

$i > 1$:

$h(i) = [K(u(i), u(1)), \ldots K(u(i), u(i-1))]^T$

↳ distance vec

$z(i) = Q(i-1)\, h(i)$

$r(i) = \lambda + K(u(i), u(i)) - z^T(i)\, h(i)$

$$Q(i) = r(i)^{-1} \begin{bmatrix} Q(i-1)\, r(i) + z(i) z^T(i) & -z(i) \\ -z^T(i) & 1 \end{bmatrix}$$

$e(i) = d(i) - h^T(i)\, a(i-1)$

$$a(i) = \begin{bmatrix} a(i-1) - z(i)\, r(i)^{-1} e(i) \\ r(i)^{-1} e(i) \end{bmatrix}$$

⇒ At iter i:, a test input u_*

$$f(u_*) = \sum_{j=1}^{i} a_j(i)\, K(u(j), u_*)$$