More on Training Strategies for Critic and Action Neural Networks in Dual Heuristic Programming Method

George G. Lendaris[1], Christian Paintz[2], Thaddeus Shannon[3]

1) Professor of Systems Science and Electrical Engineering
2) Graduate Student, Electrical Engineering
3) Graduate Student, Systems Sciencd Ph.D. Program
Portland State University
Portland, Oregon 97207

IEEE SMC'97, Orlando

Context:

Adaptive Critic Design (ACD)

A methodology for adaptively designing an (approximately) optimal controller for a given plant according to a stated criterion.

We use a NN as the controller [actionNN], and another NN [criticNN] to update (assist in the design of) the controller.

"Plant" is represented via state vector R(t).

Context:

The ACD method entails the user defining a (primary) utility function $U(t)$ for the specific application, and then maximizing a new utility function (Bellman Eqn.):

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k)$$

[We note: $J(t) = U(t) + \gamma J(t+1)$**]**

Context: Family of Adaptive Critic Designs

The criticNN approximates either $J(t)$ or gradient of $J(t)$ wrt state vector $R(t)$ $[\nabla J(R)]$

- ❖**Heuristic Dynamic Programming (HDP)**

  CriticNN approximates $J(t)$

- ❖**Dual Heuristic Programming (DHP)**

  CriticNN approximates $\nabla J(R) \equiv \lambda$

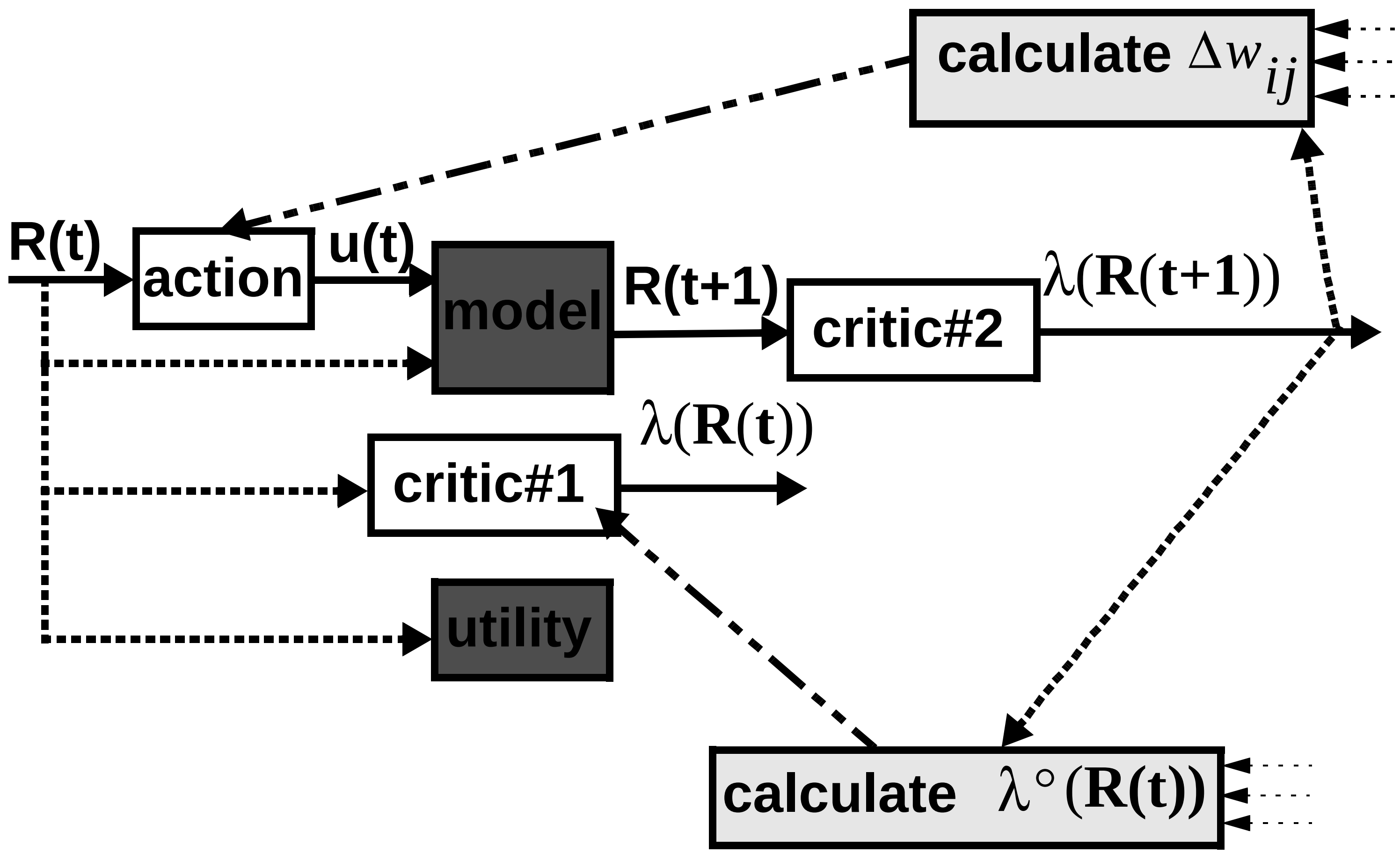
- ❖**Generalized Dual HeuristicProgramming(GDHP)**

  CriticNN approximates $J(t)$ and $\nabla J(R)$

- ❖**Action Dependent ....**

  CriticNN also inputs $u(t)$ and outputs $\nabla J(u)$

Computing Schema for discussing Strategies



Weights in actionNN are updated with objective of maximizing J(t):

$$\Delta w_{ij}(t) = lcoef \bullet \frac{\partial}{\partial w_{ij}(t)} J(t)$$

where $$\frac{\partial}{\partial w_{ij}(t)} J(t) = \sum_{k=1}^{a} \frac{\partial}{\partial u_k(t)} J(t) \bullet \frac{\partial}{\partial w_{ij}(t)} u_k(t)$$

and $$\frac{\partial}{\partial u_k(t)} J(t) = \frac{\partial}{\partial u_k(t)} U(t) + \frac{\partial}{\partial u_k(t)} J(t+1)$$

and $$\frac{\partial}{\partial u_k(t)} J(t+1) = \sum_{s=1}^{n} \frac{\partial}{\partial R_s(t+1)} J(t+1) \bullet \frac{\partial}{\partial u_k(t)} R_s(t+1)$$

call this term $\lambda(t+1)$ **(to be output of critic)**

CriticNN output is λ.

For training criticNN, "desired output" is λ°.

(cf. Eqn. (6) in paper)

Paraphrase of Eqn. (6) [cf. Eqn. (7) in paper]:

$$\lambda_s^\circ(t) = [\sim\text{Utility}] + \sum_{j=1}^{a} ([\sim\text{Utility}] \bullet [\sim\text{Action}])$$

$$+ \sum_{k=1}^{n} ([\sim\text{Critic(t+1)}] \bullet [\sim\text{Plant}])$$

$$+ \sum_{k=1}^{n} \left\{ \sum_{j=1}^{a} ([\sim\text{Critic(t+1)}] \bullet [\sim\text{Plant}] \bullet [\sim\text{Action}]) \right\}$$

Today focus on "solving" Eqn. (7)

Strategies to solve Eqn. (6) [and (7)]

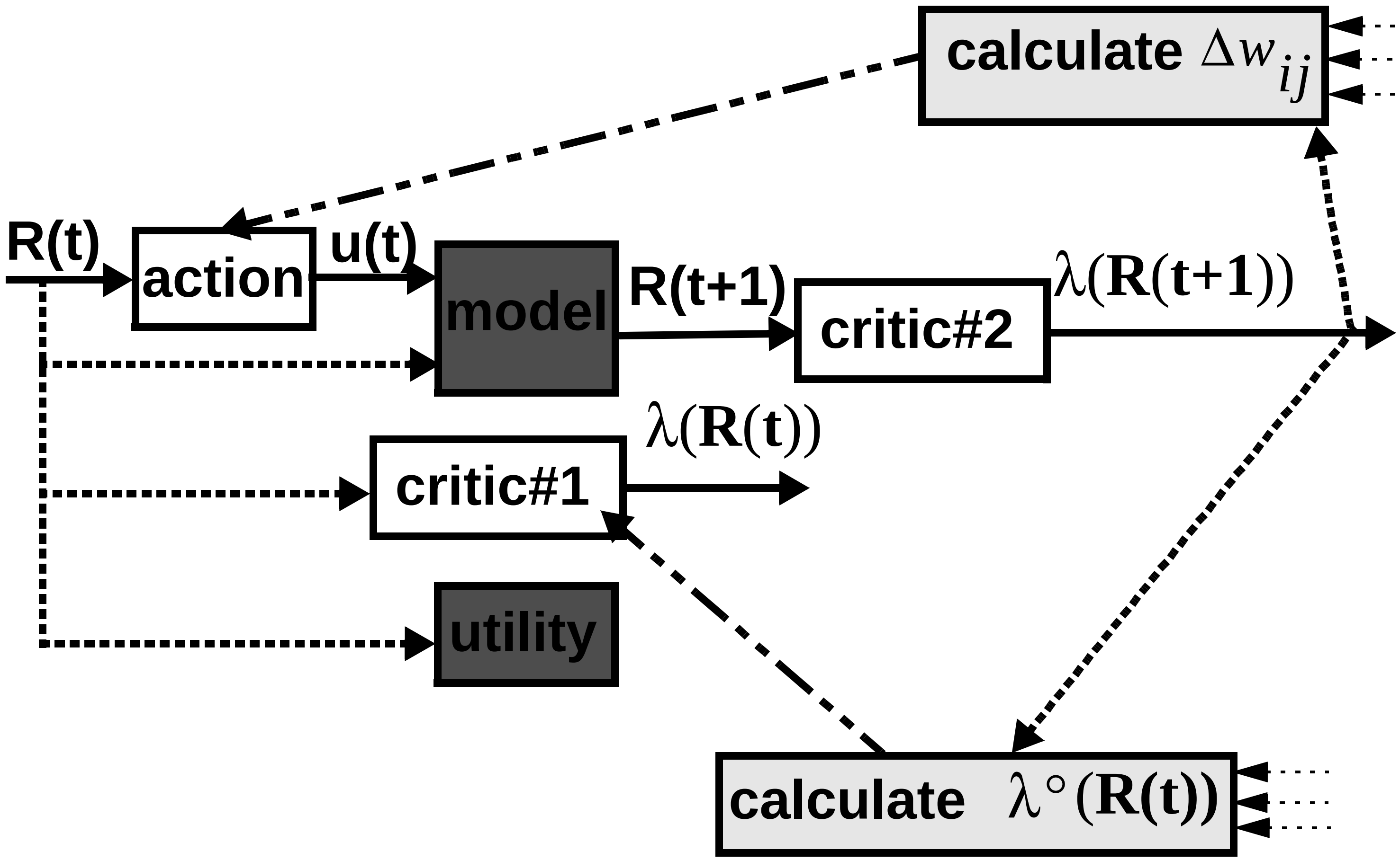
Strategy 1. Straight application of the equation.

Strategy 2. Basic 2-stage process [“flip/flop”].
[e.g., Santiago/Werbos, Prokhorov/Wunsch]
During stage 1, train criticNN, not actionNN;
During stage 2, train actionNN, not criticNN.

Strategy 3. Modified 1st stage of 2-stage process.
While train criticNN during stage 1, keep parameters constant in module that calculates critic’s desired output $\lambda^{\circ}(\mathbf{R})$. Then adjust weights all at once at end of stage 1.

Strategy 4. Single-stage process, using modifications introduced in Strategy 3.

Computing Schema for discussing Strategies



Experimental Procedures

Train 3 passes through sequence
(5, -10, 20, -5, -20, 10) [degrees from vertical].
Train 30 sec. on each angle.
Accumulate absolute values of U: C(1), C(2), C(3).

Test pass through train sequence
(30 sec. each angle). Accumulate U values: C(4).

Generalize pass through sequence
(-23, -18, -8, 3, 13, 23) [degrees from vertical].
Accumulate U values: C(5).

Generalize pass through sequence
(-38, -33, 23, 38) [degrees from vertical].
Accumulate U values: C(6).

Theta-only version of Pole-Cart problem

Strategy	via Strategy 1 Edge Gains	via corresp. Edge Gains	$D(1)_{S1}/D(1)_{TEG}$
1	206 ± 53	206 ± 53	18 / 18
2a	239 ± 2.5	226 ± 3.5	45 / 42
4a	115 ± 2.0	49 ± 1.0	16 / 6
4b	128 ± 2.5	48 ± 1.0	17 / 6

Theta-X version of Pole-Cart problem

Strategy	via Strategy 1 Edge Gains	via corresp. Edge Gains	$D(1)_{S1}/D(1)_{TEG}$
1	349 ± 11	349 ± 11	58 / 58
2a	not run	none found	-----
4a	394 ± 7.5	207 ± 9.5	62 / 33
4b	390 ± 10.5	230 ± 9.5	67 / 41

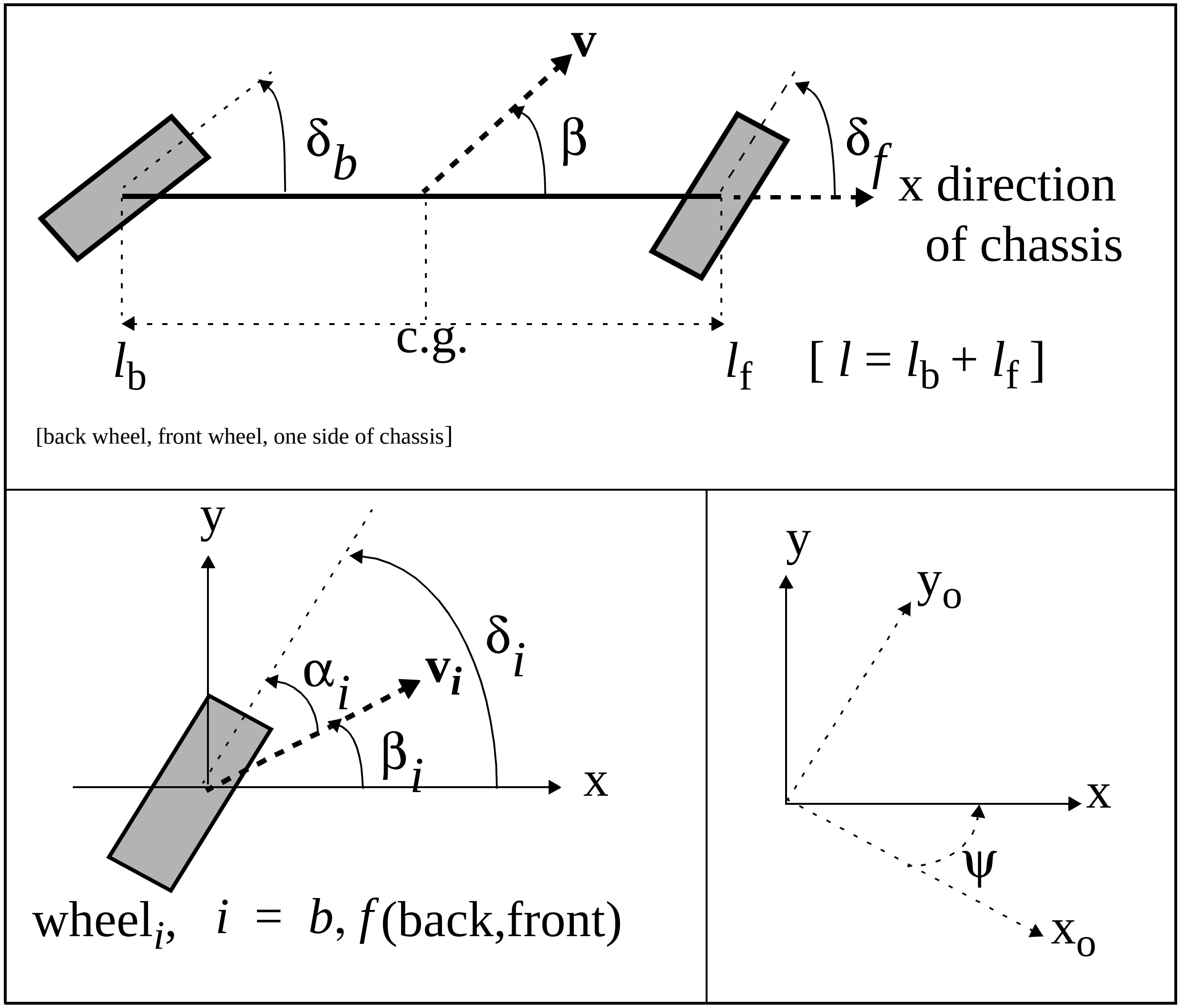
[Columns 2 & 3 are total cost (mean ± std. dev.)]
[Column 4 gives ave. number of drops for the two gains]

Characterization of Results

Edge Gains (learning coefficients) for Pole-Cart

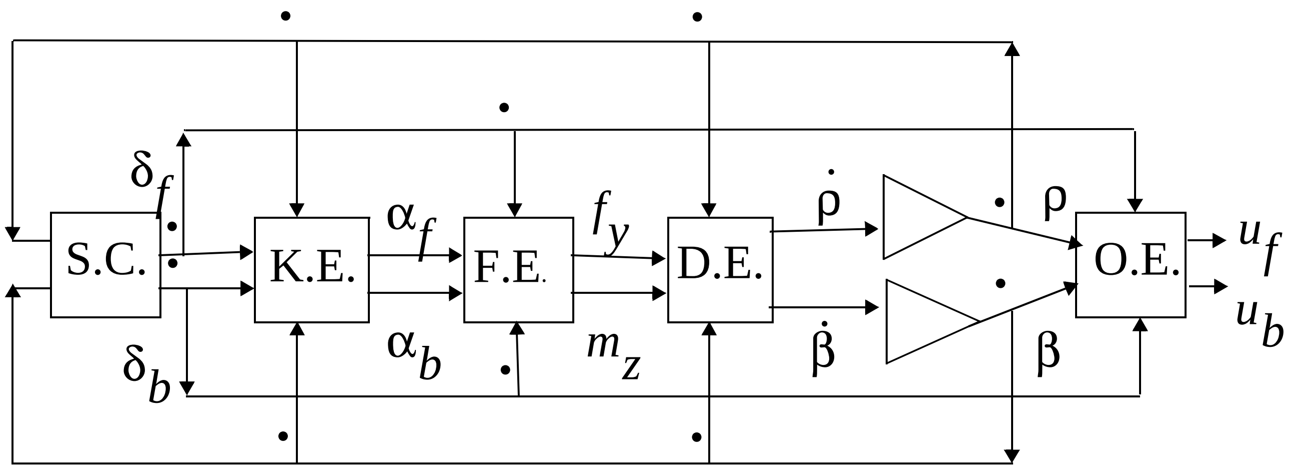
Strategy	Theta-Only critic/action gains	Theta-X critic/action gains
1	.07/.25	.02/.2
2a	.07/1.0	none
4a	.3/.9	.04/.4
4b	.3/.9	.03/.4

Learning rates for actionNN and criticNN determine speed of convergence of DHP.



v
δ_b
β
δ_f
x direction of chassis
c.g.
l_b
l_f
$[\, l = l_b + l_f \,]$
[back wheel, front wheel, one side of chassis]
y
δ_i
α_i
$\mathbf{v}_i$
β_i
x
wheel$_i$, $i = b, f$ (back,front)
y
y_o
x
ψ
x_o

Block Diagram for Bicycle Steering Model [1]



S.C.= Steering Controller
K.E.=Kinematic Equations
F.E.= Force Equations
D.E.=Dynamic Equations
O.E.=Output Equations

Recap re. criticNN:

Performs mapping: $\lambda(\mathbf{R})$

Desired output for training purposes: $\lambda^\circ(\mathbf{R})$

Solution (not known) of Bellman equation: $\lambda\hat{}(\mathbf{R})$

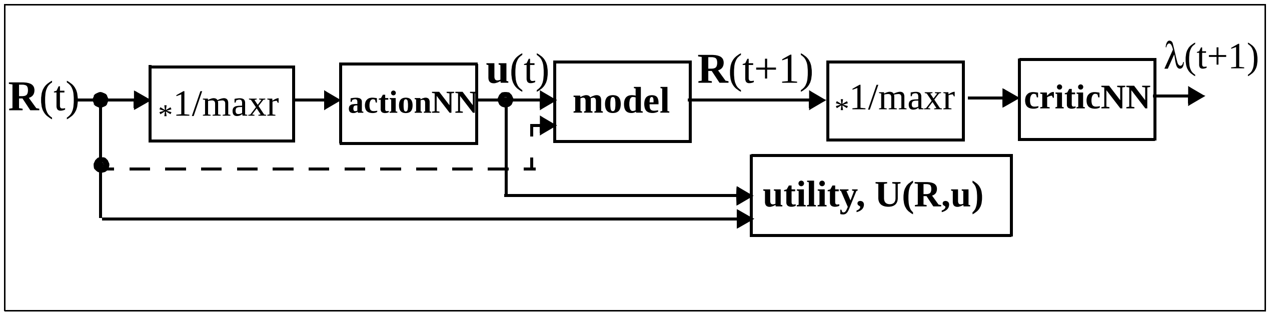
Learn process: $\lambda(\mathbf{R})$ **is to converge to** $\lambda^\circ(\mathbf{R})$;

$\lambda^\circ(\mathbf{R})$ **is to converge to** $\lambda\hat{}(\mathbf{R})$**.**

i.e., $\lambda(\mathbf{R}) \longrightarrow \lambda^\circ(\mathbf{R}) \longrightarrow \lambda\hat{}(\mathbf{R})$

[The better the criticNN "solves" the Bellman eqn., the better the actionNN will approximate an optimal controller.]

Dual Heuristic Programming (DHP)



For state R(t), actionNN -->control signal u(t).

Apply u(t), plant/model changes state to R(t+1).

Calculate utility U(R(t), u(t)).

CriticNN is used to adapt the actionNN.

[The CriticNN itself must be adapted.]

Step Responses of 6-1-1 Controller, 1m pole

[Trained w/ Θmax $\pm$ 10^o]

[No explicit X training]

Trained: 7.5^{o}displ.

Tested: 38^{o}displ.

Tested: -6.6m displ.

"Desired Output" for CriticNN:

$$\lambda_s^{\circ}(t) = \frac{d}{dR_s(t)}U(t) + \sum_{j=1}^{a}\left(\frac{\partial}{\partial u_j(t)}U(t) \bullet \frac{\partial}{\partial R_s(t)}u_j(t)\right)$$

$$+ \sum_{k=1}^{n}\left(\frac{\partial}{\partial R_k(t+1)}J(t+1) \bullet \frac{d}{dR_s(t)}R_k(t+1)\right)$$

$$+ \sum_{k=1}^{n}\left\{\sum_{j=1}^{a}\left(\frac{\partial}{\partial R_k(t+1)}J(t+1) \bullet \frac{\partial}{\partial u_j(t)}R_k(t+1) \bullet \frac{\partial}{\partial R_s(t)}u_j(t)\right)\right\}$$